

SQA Assignment 1

Submitted By:

- Md. Ehsan Khan 011201122
- Shofi Rayhan Siyam 011201117
- Muhammad Rahat Ahasan 011201112

Section: B

Course: CSE 4495

Submitted To: Md. Mohaiminul Islam

Date: 28 Dec 2024

Abstract

This report summarizes the testing of a Flask-based API for managing students and programs. The tests validated the API's functionality, reliability, and error handling through various scenarios, including edge cases. The results confirm the API's readiness for deployment while highlighting areas for potential improvement.

Links

- Report (Markdown): [Report](#)
- Postman Collection Export: [SQA Assignment 1.postman_collection.json](#)
- Postman Collection Online: [SQA Assignment 1](#)

Test Report

API Endpoint 1: `/student`

Method: `GET`

Request Body: `NONE`

Response

```
[
  {
    "courses_passed": [
      "CSE2213",
      "CSE1110",
      "CSE1111",
      "CSE1114",
      "CSE1115",
      "CSE1116",
      "CSE2117",
      "CSE2118",
      "CSE3119",
      "CSE4010"
    ],
    "id": 1,
    "name": "Abdul Jabbar",
    "personnummer": "920223-9999"
  },
  {
    "courses_passed": [
      "CSE2213",
      "CSE1110",
      "CSE1111",
      "CSE1114",
      "CSE1115",
      "CSE1116",
      "CSE2117"
    ],
    "id": 2,
```

```
    "name": "Abdel Ahmed",
    "personnummer": "990111-9999"
  },
  {
    "courses_passed": [],
    "id": 3,
    "name": "Joel Stevansson ",
    "personnummer": "011030-9999"
  },
  {
    "courses_passed": [
      "DSE1001",
      "DSEE1002",
      "DSE1003",
      "DSE1004",
      "DSE2105",
      "DSE2116",
      "DSE3217",
      "DSE3218",
      "DSE3239",
      "DSE4100"
    ],
    "id": 4,
    "name": "Yue Sakamoto",
    "personnummer": "980401-9999"
  },
  {
    "courses_passed": [
      "DSE1001",
      "DSE1002",
      "DSE1003",
      "DSE3217"
    ],
    "id": 5,
    "name": "Ivan Faknamovich",
    "personnummer": "951128-9999"
  },
  {
    "courses_passed": [
      "CSE2213",
      "CSE1110",
      "CSE1111",
      "CSE1114",
      "CSE1115",
      "CSE1116",
      "CSE2117",
      "CSE2118",
      "CSE3119",
      "CSE4010",
      "DSE1004",
      "DSE2105"
    ],
    "id": 6,
```

```

    "name": "Amy Pond",
    "personnummer": "020201-9999"
  },
  {
    "courses_passed": [
      "CSE1115",
      "CSE4010",
      "DSE1001",
      "DSE1002",
      "DSE1003",
      "DSE1004",
      "DSE2105",
      "DSE2116",
      "DSE3217",
      "DSE3218",
      "DSE3239",
      "DSE4100"
    ],
    "id": 7,
    "name": "Rory Williams",
    "personnummer": "010203-9999"
  },
  {
    "courses_passed": [
      "CSE1115",
      "CSE4010",
      "DSE1001",
      "DSE1004",
      "DSE2105",
      "DSE1004"
    ],
    "id": 8,
    "name": "Steve Rogers",
    "personnummer": "07045-9999"
  },
  {
    "courses_passed": [
      "CSE2213"
    ],
    "id": 9,
    "name": "Natasha Romanov",
    "personnummer": "861231-9999"
  }
]

```

Scripts

```

pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});

```

```

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('array');
  responseData.forEach(function(student) {
    pm.expect(student).to.have.property('courses_passed');
    pm.expect(student).to.have.property('id');
    pm.expect(student).to.have.property('name');
    pm.expect(student).to.have.property('personnummer');
  });
});

pm.test("Courses_passed is an array", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('array');
  pm.expect(responseData[0].courses_passed).to.be.an('array');
});

pm.test("Personnummer is in a valid format", function () {
  const responseData = pm.response.json();

  responseData.forEach(function(student) {
    pm.expect(student.personnummer).to.match(/^\d{6}-\d{4}$/);
  });
});

```

Purpose

- To test the API endpoint `/student` with a valid request

Assertion

- PASSED Response status code is 200
- PASSED Response has the required fields
- PASSED Courses_passed is an array
- FAILED Personnummer is in a valid format | AssertionError: expected '07045-9999' to match `/^\d{6}-\d{4}$/`

Verdict: FAILED

API Endpoint 2: `/student/1`

Method: GET

Request Body: NONE

Response

```
{
  "courses_passed": [
    "CSE2213",
    "CSE1110",
    "CSE1111",
    "CSE1114",
    "CSE1115",
    "CSE1116",
    "CSE2117",
    "CSE2118",
    "CSE3119",
    "CSE4010"
  ],
  "id": 1,
  "name": "Abdul Jabbar",
  "personnummer": "920223-9999"
}
```

Scripts

```
pm.test("Response status code is 200", function () {
  pm.expect(pm.response.to.have.status(200));
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.courses_passed).to.exist;
  pm.expect(responseData.id).to.exist;
  pm.expect(responseData.name).to.exist;
  pm.expect(responseData.personnummer).to.exist;
});

pm.test("Courses_passed is an array", function () {
  const responseData = pm.response.json();
  pm.expect(responseData.courses_passed).to.be.an('array');
});

pm.test("ID is a non-negative integer", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.id).to.be.a('number').and.to.satisfy((val) => val >= 0, "ID should");
});

pm.test("Personnummer is in a valid format", function () {
  const responseData = pm.response.json();
```

```
pm.expect(responseData.personnummer).to.match(/^d{6}-d{4}$/);
});
```

Purpose

- To test the API endpoint `/student/1` with a valid student ID

Assertion

- PASSED Response status code is 200
- PASSED Response has the required fields
- PASSED Courses_passed is an array
- PASSED ID is a non-negative integer
- PASSED Personnummer is in a valid format

Verdict: PASSED

API Endpoint 3: `/student/10`

Method: GET

Request Body: NONE

Response

```
{
  "error": "Student ID 10 does not exist"
}
```

Scripts

```
pm.test("Response status code is 404", function () {
  pm.expect(pm.response.code).to.equal(404);
});

pm.test("Response is in JSON format", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
```

Purpose

- To test the API endpoint `/student/10` with a non-existent student ID

Assertion

- PASSED Response status code is 404
- PASSED Response is in JSON format

Verdict: FAILED

API Endpoint 4: /create

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response

```
{
  "courses_passed": [
    "CSE2213",
    "CSE1110"
  ],
  "id": 10,
  "name": "Fakibaz Student",
  "personnummer": "990101-1234"
}
```

Scripts

```
pm.test("Response status code is 201", function () {
  pm.expect(pm.response.code).to.equal(201);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData).to.have.property('courses_passed');
  pm.expect(responseData).to.have.property('id');
  pm.expect(responseData).to.have.property('name');
  pm.expect(responseData).to.have.property('personnummer');
});
```



```

pm.test("Courses passed array is present and has expected number of elements", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData.courses_passed).to.exist.and.to.be.an('array');
    pm.expect(responseData.courses_passed).to.have.lengthOf(2);
});

pm.test("Id is a non-negative integer", function () {
    const responseData = pm.response.json();
    pm.expect(responseData.id).to.be.a('number').and.to.be.at.least(0, "Id should be a non-neg
});

pm.test("Name and personnummer are non-empty strings", function () {
    const responseData = pm.response.json();
    pm.expect(responseData.name).to.be.a('string').and.to.have.lengthOf.at.least(1, "Name shou
    pm.expect(responseData.personnummer).to.be.a('string').and.to.have.lengthOf.at.least(1, "P
});

```

Purpose

- To test the API endpoint `/create` with a valid request body

Assertion

- PASSED Response status code is 201
- PASSED Response has the required fields
- PASSED Courses passed array is present and has expected number of elements
- PASSED Id is a non-negative integer
- PASSED Name and personnummer are non-empty strings



Verdict: PASSED

API Endpoint 5: `/create`

Method: POST

Request Body

```

{
    "personnummer": "990101-1234",
    "courses_passed": ["CSE2213", "CSE1110"]
}

```

Response

```
{  
  "error": "No name field"  
}
```

Scripts

```
pm.test("Response status code is 400", function () {  
  pm.expect(pm.response.code).to.equal(400);  
});
```

```
pm.test("Response has the required field - error", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('object');  
  pm.expect(responseData.error).to.exist;  
});
```

```
pm.test("Error is a non-empty string", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error should be a non-empty string");  
});
```

```
pm.test("Content-Type header is application/json", function () {  
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");  
});
```

```
pm.test("Verify the error message 'No name field'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "No name field";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```



Purpose

- To test the API endpoint `/create` with a missing `name` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error

- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'No name field'

Verdict: PASSED

API Endpoint 6: /create

Method: POST

Request Body

```
{
  "name": "",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Blank or null name"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});
```

```
pm.test("Error is a non-empty string", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error shou
});
```

```
pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Verify the error message 'Blank or null name'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Blank or null name";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/create` with a blank `name` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'Blank or null name'

Verdict: PASSED

API Endpoint 7: `/create`

Method: POST

Request Body

```
{
  "error": "No personnummer field"
}
```

Response Body

```
{
  "error": "No personnummer field"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error is a non-empty string", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error shou
});

pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Verify the error message 'No personnummer field'", function () {
  const responseData = pm.response.json();
  const errorMessage = "No personnummer field";
  pm.expect(responseData.error).to.include(errorMessage);
});
```



Purpose

- To test the API endpoint `/create` with a missing `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'No personnummer field'

Verdict: PASSED

API Endpoint 8: `/create`

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Blank or null personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});
```

```
pm.test("Error is a non-empty string", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error shou
});
```

```
pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
```

```
pm.test("Verify the error message 'Blank or null personnummer'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Blank or null personnummer";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

```
});
```

Purpose

- To test the API endpoint `/create` with a blank `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'Blank or null personnummer'

Verdict: PASSED

API Endpoint 9: `/create`

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "0101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Malformed personnummer 0101-1234"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();
```

```

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData.error).to.exist;
  });

  pm.test("Error is a non-empty string", function () {
    const responseData = pm.response.json();

    pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error shou
  });

  pm.test("Content-Type header is application/json", function () {
    pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
  });


  pm.test("Verify the error message 'Malformed personnummer'", function () {
    const responseData = pm.response.json();
    const errorMessage = "Malformed personnummer";
    pm.expect(responseData.error).to.include(errorMessage);
  });

```

Purpose

- To test the API endpoint `/create` with a malformed `personnummer` field in the request body

Assertion

-  PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'Malformed personnummer'

Verdict: PASSED

API Endpoint 10: `/create`

Method: POST

Request Body

```

{
  "name": "Fakibaz Student",
  "personnummer": "991301-1234",

```



```
    "courses_passed": ["CSE2213", "CSE1110"]
  }
```

Response Body

```
{
  "error": "Invalid month in personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});
```

```
pm.test("Error is a non-empty string", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error should be a non-empty string");
});
```

```
pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
```

```
pm.test("Verify the error message 'Invalid month in personnummer'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Invalid month in personnummer";
  pm.expect(responseData.error).to.include(errorMessage);
});
```



Purpose

- To test the API endpoint `/create` with an invalid month in the `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'Invalid month in personnummer'

Verdict: PASSED

API Endpoint 11: /create

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "990132-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Invalid day in personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
pm.test("Response has the required field - error", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});
```

```
pm.test("Error is a non-empty string", function () {
  const responseData = pm.response.json();
```

```
pm.expect(responseData.error).to.be.a('string').and.to.have.lengthOf.at.least(1, "Error shou
});

pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Verify the error message 'Invalid day in personnummer'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Invalid day in personnummer";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/create` with an invalid day in the `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required field - error
- PASSED Error is a non-empty string
- PASSED Content-Type header is application/json
- PASSED Verify the error message 'Invalid day in personnummer'

◀  ▶

Verdict: PASSED

API Endpoint 12: `/create`

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{  
  "error": "Personnummer already belongs to student 10"  
}
```

Scripts

```
pm.test("Response status code is 400", function () {  
  pm.expect(pm.response.code).to.equal(400);  
});  
  
pm.test("Response has the required fields", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('object');  
  pm.expect(responseData.error).to.exist;  
});  
  
pm.test("Error field is non-empty", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");  
});  
  
pm.test("Verify the error message 'Personnummer already belongs to student'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "Personnummer already belongs to student";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```



Purpose

- To test the API endpoint `/create` with a `personnummer` field that already belongs to a student

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Personnummer already belongs to student'

Verdict: PASSED

API Endpoint 13: `/create`

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "880101-1234",
  "courses_passed": ["", "CSE1110"]
}
```

Response Body

```
{
  "error": "Empty or null course ID"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'Empty or null course ID'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Empty or null course ID";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/create` with an empty or null course ID in the request body

Assertion

- PASSED Response status code is 400

- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Empty or null course ID'

Verdict: PASSED

API Endpoint 14: /create

Method: POST

Request Body

```
{
  "name": "Fakibaz Student",
  "personnummer": "200101-1234",
  "courses_passed": 12
}
```

Response Body

```
<!doctype html>
<html lang=en>
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either th
```



Scripts

```
pm.test("Response status code is 500", function () {
  pm.expect(pm.response.code).to.equal(500);
});

pm.test("Response content type is HTML", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("text/html");
});
```

Purpose

- To test the API endpoint /create with an unhandled exception in the server

Assertion

- PASSED Response status code is 500

- PASSED Response content type is HTML

Verdict: PASSED

API Endpoint 15: /update/10

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "courses_passed": [
    "CSE2213",
    "CSE1110"
  ],
  "id": "10",
  "name": "Fakibaz Student Updated",
  "personnummer": "990101-1234"
}
```

Scripts

```
pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});
```

Purpose

- To test the API endpoint /update/10 with valid request body

Assertion

- PASSED Response status code is 200

Verdict: PASSED

API Endpoint 16: /update/11

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "980101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Student ID 11 does not exist"
}
```

Scripts

```
pm.test("Response status code is 404", function () {
  pm.response.to.have.status(404);
});
```

Purpose

- To test the API endpoint /update/11 with a non-existent student ID

Assertion

- PASSED Response status code is 404

Verdict: PASSED

API Endpoint 17: /update/10

Method: PUT

Request Body

```
{
  "personnummer": "990101-1234",
```



```
    "courses_passed": ["CSE2213", "CSE1110"]
  }
```

Response Body

```
{
  "error": "No name field"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'No name field'", function () {
  const responseData = pm.response.json();
  const errorMessage = "No name field";
  pm.expect(responseData.error).to.include(errorMessage);
});
```



Purpose

- To test the API endpoint `/update/10` with a missing `name` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'No name field'

Verdict: PASSED

API Endpoint 18: /update/10

Method: PUT

Request Body

```
{
  "name": "",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Blank or null name"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'Blank or null name'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Blank or null name";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/update/10` with a blank `name` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Blank or null name'

Verdict: PASSED

API Endpoint 19: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Blank or null personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
```

```
const responseData = pm.response.json();

pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should
});

pm.test("Verify the error message 'Blank or null personnummer'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Blank or null personnummer";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/update/10` with a blank `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Blank or null personnummer'

Verdict: PASSED

API Endpoint 20: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "No personnummer field"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});


pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'No personnummer field'", function () {
  const responseData = pm.response.json();
  const errorMessage = "No personnummer field";
  pm.expect(responseData.error).to.include(errorMessage);
});
```



Purpose

- To test the API endpoint `/update/10` with a missing `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'No personnummer field'

Verdict: PASSED

API Endpoint 21: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "0101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

```
}
```

Response Body

```
{  
  "error": "Malformed personnummer 0101-1234"  
}
```

Scripts

```
pm.test("Response status code is 400", function () {  
  pm.expect(pm.response.code).to.equal(400);  
});  
  
pm.test("Response has the required fields", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('object');  
  pm.expect(responseData.error).to.exist;  
});  
  
pm.test("Error field is non-empty", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");  
});  
  
pm.test("Verify the error message 'Malformed personnummer'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "Malformed personnummer";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```



Purpose

- To test the API endpoint `/update/10` with a malformed `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Malformed personnummer'

Verdict: PASSED

API Endpoint 22: /update/10

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "991301-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Invalid month in personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'Invalid month in personnummer'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Invalid month in personnummer";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/update/10` with an invalid month in the `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Invalid month in personnummer'

Verdict: PASSED

API Endpoint 23: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "990132-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Invalid day in personnummer"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();
```



```
pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should  
});  
  
pm.test("Verify the error message 'Invalid day in personnummer'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "Invalid day in personnummer";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```

Purpose

- To test the API endpoint `/update/10` with an invalid day in the `personnummer` field in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty

◀ ◀ PASSED Verify the error message 'Invalid day in personnummer' ▶ ▶

Verdict: PASSED

API Endpoint 24: `/update/10`

Method: PUT

Request Body

```
{  
  "name": "Fakibaz Student Updated",  
  "personnummer": "990130-1234",  
  "courses_passed": ["CSE213", "CSE1110"]  
}
```

Response Body

```
{  
  "error": "Malformed course ID: CSE213"  
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});


pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'Malformed course ID'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Malformed course ID";
  pm.expect(responseData.error).to.include(errorMessage);
});
```



Purpose

- To test the API endpoint `/update/10` with a malformed course ID in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Malformed course ID'

Verdict: PASSED

API Endpoint 25: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "990130-1234",
  "courses_passed": [ "", "CSE1110" ]
```

```
}
```

Response Body

```
{  
  "error": "Empty or null course ID"  
}
```

Scripts

```
pm.test("Response status code is 400", function () {  
  pm.expect(pm.response.code).to.equal(400);  
});  
  
pm.test("Response has the required fields", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('object');  
  pm.expect(responseData.error).to.exist;  
});  
  
pm.test("Error field is non-empty", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");  
});  
  
pm.test("Verify the error message 'Empty or null course ID'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "Empty or null course ID";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```



Purpose

- To test the API endpoint `/update/10` with an empty or null course ID in the request body

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Empty or null course ID'

Verdict: PASSED

API Endpoint 26: /update/10

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "980101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Changes to personnummer are not allowed."
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
});

pm.test("Verify the error message 'Changes to personnummer are not allowed.'", function () {
  const responseData = pm.response.json();
  const errorMessage = "Changes to personnummer are not allowed.";
  pm.expect(responseData.error).to.include(errorMessage);
});
```

Purpose

- To test the API endpoint `/update/10` with an attempt to change the `personnummer` field in the request body which are not allowed.

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Changes to personnummer are not allowed.'

Verdict: PASSED

API Endpoint 27: `/update/10`

Method: PUT

Request Body

```
{
  "name": "Fakibaz Student Updated",
  "personnummer": "990101-1234",
  "courses_passed": ["CSE2213", "CSE1110"]
}
```

Response Body

```
{
  "error": "Personnummer already belongs to student 10"
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData.error).to.exist;
});

pm.test("Error field is non-empty", function () {
  const responseData = pm.response.json();
```

```
pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should  
});  
  
pm.test("Verify the error message 'Personnummer already belongs to student'", function () {  
  const responseData = pm.response.json();  
  const errorMessage = "Personnummer already belongs to student";  
  pm.expect(responseData.error).to.include(errorMessage);  
});
```

Purpose

- To test the API endpoint `/update/10` with a `personnummer` field that already belongs to a student

Assertion

- PASSED Response status code is 400
- PASSED Response has the required fields
- PASSED Error field is non-empty
- PASSED Verify the error message 'Personnummer already belongs to student'

Verdict: PASSED

API Endpoint 28: `/update/10`

Method: PUT

Request Body

```
{  
  "name": "Fakibaz Student Updated",  
  "personnummer": "990101-1234",  
  "courses_passed": 12  
}
```

Response Body

```
<!doctype html>  
<html lang=en>  
<title>500 Internal Server Error</title>  
<h1>Internal Server Error</h1>  
<p>The server encountered an internal error and was unable to complete your request. Either th  
  there is an error in the application.</p>
```

Scripts

```
pm.test("Response status code is 500", function () {  
    pm.expect(pm.response.code).to.equal(500);  
});  
  
pm.test("Response content type is HTML", function () {  
    pm.expect(pm.response.headers.get("Content-Type")).to.include("text/html");  
});
```

Purpose

- To test the API endpoint `/update/10` with an unhandled exception in the server

Assertion

- PASSED Response status code is 500
- PASSED Response content type is HTML

Verdict: PASSED

API Endpoint 29: `/delete/10`

Method: DELETE

Request Body: NONE

Response Body

```
{  
    "deleted": "10"  
}
```

Scripts

```
pm.test("Response status code is 200", function () {  
    pm.expect(pm.response.code).to.equal(200);  
});  
  
pm.test("Response has the required field 'deleted'", function () {  
    const responseData = pm.response.json();  
  
    pm.expect(responseData).to.be.an('object');  
    pm.expect(responseData.deleted).to.exist;  
});
```

```
pm.test("Deleted field should not be empty", function () {
  const responseData = pm.response.json();

  pm.expect(responseData.deleted).to.exist.and.to.not.be.empty;
});

pm.test("Content type is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
```

Purpose

- To test the API endpoint `/delete/10` with a valid request

Assertion

- PASSED Response status code is 200
- PASSED Response has the required field 'deleted'
- PASSED Deleted field should not be empty

Verdict: PASSED

API Endpoint 30: `/delete/fds`

Method: DELETE

Request Body: NONE

Response Body

```
{
  "error": "Student ID fds is not formatted correctly."
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
```



```

    pm.expect(responseData.error).to.exist;
  });

  pm.test("Error field is non-empty", function () {
    const responseData = pm.response.json();

    pm.expect(responseData.error).to.exist.and.to.have.lengthOf.at.least(1, "Error field should");
  });

  pm.test("Verify the error message 'Student ID fds is not formatted correctly.'", function () {
    const responseData = pm.response.json();
    const errorMessage = "Student ID fds is not formatted correctly.";
    pm.expect(responseData.error).to.include(errorMessage);
  });
}

```

Purpose

- To test the API endpoint `/delete/fds` with an invalid student ID

Assertion

- **PASSED** Response status code is 400
- **PASSED** Response has the required fields
- **PASSED** Error field is non-empty
- **PASSED** Verify the error message 'Student ID fds is not formatted correctly.'

Verdict: **PASSED**

API Endpoint 31: `/delete/100`

Method: **DELETE**

Request Body: **NONE**

Response Body

```

{
  "error": "Student ID 100 does not exist"
}

```

Scripts

```

pm.test("Response status code is 404", function () {
  pm.expect(pm.response.code).to.equal(404);
});

```

```
});
```

Purpose

- To test the API endpoint `/delete/100` with a non-existent student ID

Assertion

- PASSED Response status code is 404

Verdict: PASSED

API Endpoint 32: `/program`

Method: GET

Request Body: NONE

Response Body

```
[
  {
    "courses_required": [
      "CSE2213",
      "CSE1110",
      "CSE1111",
      "CSE1114",
      "CSE1115",
      "CSE1116",
      "CSE2117",
      "CSE2118",
      "CSE3119",
      "CSE4010"
    ],
    "id": 1
  },
  {
    "courses_required": [
      "DSE1001",
      "DSE1002",
      "DSE1003",
      "DSE1004",
      "DSE2105",
      "DSE2116",
      "DSE3217",
      "DSE3218",
      "DSE3239",
      "DSE4100"
    ]
  }
]
```

```
    ],  
    "id": 2  
  }  
]
```

Scripts

```
pm.test("Response status code is 200", function () {  
  pm.response.to.have.status(200);  
});  
  
pm.test("Response has the required fields", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('array');  
  responseData.forEach(function(item) {  
    pm.expect(item).to.have.property('courses_required');  
    pm.expect(item).to.have.property('id');  
  });  
});  
  
pm.test("Courses_required is an array", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('array');  
  pm.expect(responseData[0].courses_required).to.be.an('array');  
});  
  
pm.test("Id is a non-negative integer", function () {  
  const responseData = pm.response.json();  
  
  pm.expect(responseData).to.be.an('array');  
  responseData.forEach(function(item) {  
    pm.expect(item.id).to.be.a('number').and.to.be.at.least(0);  
  });  
});
```

Purpose

- To test the API endpoint `/program` with a valid request

Assertion

- PASSED Response status code is 200
- PASSED Response has the required fields
- PASSED Courses_required is an array

- PASSED Id is a non-negative integer

Verdict: PASSED

API Endpoint 33: /program/1

Method: GET

Request Body: NONE

Response Body

```
{
  "courses_required": [
    "CSE2213",
    "CSE1110",
    "CSE1111",
    "CSE1114",
    "CSE1115",
    "CSE1116",
    "CSE2117",
    "CSE2118",
    "CSE3119",
    "CSE4010"
  ],
  "id": 1
}
```

Scripts

```
pm.test("Response status code is 200", function () {
  pm.expect(pm.response.code).to.equal(200);
});

pm.test("Content-Type header is application/json", function () {
  pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Response schema contains required fields - courses_required and id", function () {
  const responseData = pm.response.json();

  pm.expect(responseData).to.be.an('object');
  pm.expect(responseData).to.have.property('courses_required');
  pm.expect(responseData).to.have.property('id');
});
```

Purpose

- To test the API endpoint `/program/1` with a valid request

Assertion

- PASSED Response status code is 200
- PASSED Content-Type header is application/json
- PASSED Response schema contains required fields - `courses_required` and `id`

Verdict: PASSED

API Endpoint 34: `/program/fdg`

Method: GET

Request Body: NONE

Response Body

```
{
  "error": "Program ID fdg is not formatted correctly."
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

Purpose

- To test the API endpoint `/program/fdg` with an invalid program ID

Assertion

- PASSED Response status code is 400

Verdict: PASSED

API Endpoint 35: `/program/500`

Method: GET

Request Body: NONE

Response Body

```
{
  "error": "Program ID 500 does not exist"
}
```

Scripts

```
pm.test("Response status code is 404", function () {
  pm.expect(pm.response.code).to.equal(404);
});
```

Purpose

- To test the API endpoint `/program/500` with a non-existent program ID

Assertion

- PASSED Response status code is 404

Verdict: PASSED

API Endpoint 36: `/finished/1/1`

Method: GET

Request Body: NONE

Response Body

```
{
  "completed_courses": 10,
  "status": true
}
```

Scripts

```
pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});
```

```
pm.test("Response has the required fields - completed_courses and status", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData).to.have.property('completed_courses');
    pm.expect(responseData).to.have.property('status');
});

pm.test("Response content type is application/json", function () {
    pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Completed_courses is an integer and status is a boolean", function () {
    const responseData = pm.response.json();

    pm.expect(responseData.completed_courses).to.be.a('number');
    pm.expect(responseData.status).to.be.a('boolean');
});
```

Purpose

- To test the API endpoint `/finished/1/1` with a valid request

Assertion

- PASSED Response status code is 200
- PASSED Response has the required fields - completed_courses and status
- PASSED Response content type is application/json
- PASSED Completed_courses is an integer and status is a boolean

Verdict: PASSED

API Endpoint 37: `/finished/1/4`

Method: GET

Request Body: NONE

Response Body

```
{
  "error": "Program ID 4 does not exist"
}
```

Scripts

```
pm.test("Response status code is 404", function () {  
    pm.expect(pm.response.code).to.equal(404);  
});
```

Purpose

- To test the API endpoint `/finished/1/4` with a non-existent program ID

Assertion

- PASSED Response status code is 404

Verdict: PASSED

API Endpoint 38: `/finished/100/1`

Method: GET

Request Body: NONE

Response Body

```
{  
    "error": "Student ID 100 does not exist"  
}
```

Scripts

```
pm.test("Response status code is 404", function () {  
    pm.expect(pm.response.code).to.equal(404);  
});
```

Purpose

- To test the API endpoint `/finished/100/1` with a non-existent student ID

Assertion

- PASSED Response status code is 404

Verdict: PASSED

API Endpoint 39: `/finished/dgfs/1`

Method: GET

Request Body: NONE

Response Body

```
{
  "error": "Student ID dgfs is not formatted correctly."
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

Purpose

- To test the API endpoint `/finished/dgfs/1` with an invalid student ID

Assertion

- PASSED Response status code is 400

Verdict: PASSED

API Endpoint 40: `/finished/1/sdfds`

Method: GET

Request Body: NONE

Response Body

```
{
  "error": "Program ID sdfds is not formatted correctly."
}
```

Scripts

```
pm.test("Response status code is 400", function () {
  pm.expect(pm.response.code).to.equal(400);
});
```

```
});
```

Purpose

- To test the API endpoint `/finished/1/sdfds` with an invalid program ID

Assertion

- `PASSED` Response status code is 400

Verdict: `PASSED`