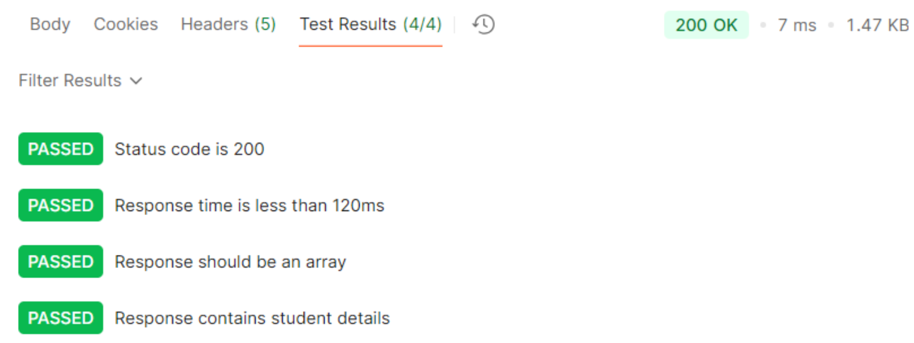
Route 1: (/student)

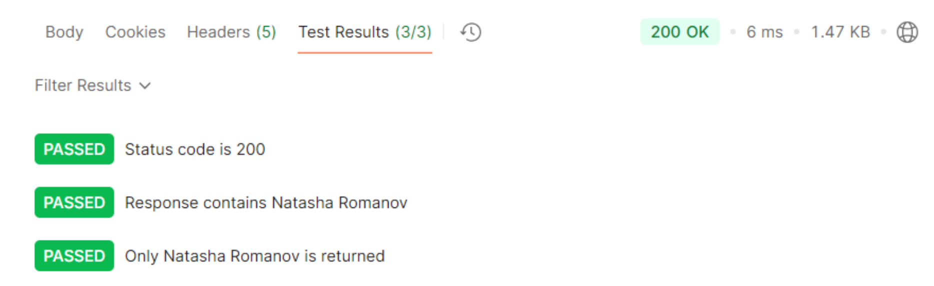
Test Case	**01**
Description	Retrieve all students
URL	http://127.0.0.1:5000/student
Body	None
Method	GET
Choices	Fetch the student list
Representative Values	List of students
Constraints	Test fails if response time > 120ms. Response must be a JSON array with id, name, and personnummer.

Post-response Scripts	pm.test("Status code is 200", function () {
	    pm.response.to.have.status(200);
	});
	
	pm.test("Response time is less than 120ms", function () {
	    pm.expect(pm.response.responseTime).to.be.below(120);
	});
	
	pm.test("Response should be an array", function () {
	    pm.expect(pm.response.json()).to.be.an("array");
	});
	
	pm.test("Response contains student details", function () {
	    const students = pm.response.json();
	    students.forEach(student => {
	        pm.expect(student).to.have.property("id");
	        pm.expect(student).to.have.property("name");
	        pm.expect(student).to.have.property("personnummer");
	    });
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (4/4) 200 OK • 7 ms • 1.47 KB Filter Results PASSED Status code is 200 PASSED Response time is less than 120ms PASSED Response should be an array PASSED Response contains student details
Comments (if any)	Successfully achieved the goal

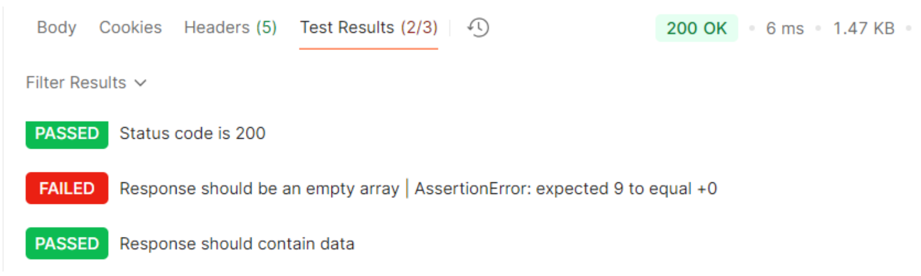
Test Case	02
Description	Retrieve students filtered by name
URL	http://127.0.0.1:5000/student?name=Natasha Romanov
Body	None
Method	GET
Choices	Filter by exact name match
Representative Values	Name: Natasha Romanov
Constraints	API must return the student whose name matches Natasha Romanov.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Response contains Natasha Romanov", function () {`
	`    const students = pm.response.json();`
	`    const natasha = students.find(student => student.name === "Natasha Romanov");`
	`    pm.expect(natasha).to.not.be.undefined;`
	`});`
	
	`pm.test("Only Natasha Romanov is returned", function () {`
	`    const students = pm.response.json();`
	`    const natasha = students.filter(student => student.name === "Natasha Romanov");`
	`    pm.expect(natasha.length).to.equal(1);`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 200 OK • 6 ms • 1.47 KB Filter Results PASSED Status code is 200 PASSED Response contains Natasha Romanov PASSED Only Natasha Romanov is returned
Comments (if any)	None

Test Case	**03**
Description	Verify behavior with an empty student list
URL	http://127.0.0.1:5000/student
Body	None
Method	GET
Choices	Return an empty list
Representative Values	Empty student list
Constraints	API should return an empty array with status code `200` when no students exist.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Response should be an empty array", function () {`
	`    const responseData = pm.response.json();`
	`    pm.expect(responseData).to.be.an("array");`
	`    pm.expect(responseData.length).to.equal(0);`
	`});`
	`pm.test("Response should contain data", function () {`
	`    const responseData = pm.response.json();`
	`    pm.expect(responseData.length).to.be.greaterThan(0);`
	`});`

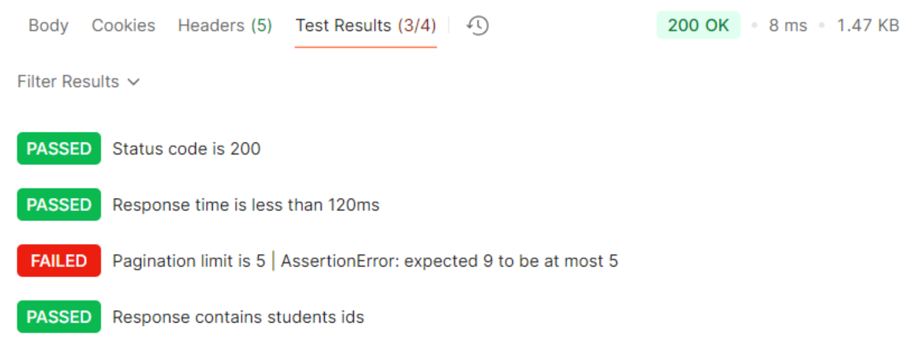
Pass/Fail	 Body Cookies Headers (5) Test Results (2/3) 200 OK • 6 ms • 1.47 KB Filter Results PASSED Status code is 200 FAILED Response should be an empty array | AssertionError: expected 9 to equal +0 PASSED Response should contain data

Comments (if any)	The student list is not empty

<table>
<tr><th>Test Case</th><th>04</th></tr>
<tr><td>Description</td><td>Retrieve students with pagination</td></tr>
<tr><td>URL</td><td>http://127.0.0.1:5000/student?page=1&limit=5</td></tr>
<tr><td>Body</td><td>None</td></tr>
<tr><td>Method</td><td>GET</td></tr>
<tr><td>Choices</td><td>Paginated student list</td></tr>
<tr><td>Representative Values</td><td>Page: 1
Limit: 5</td></tr>
<tr><td>Constraints</td><td>Response must contain ≤ 5 students.
If more than 5 students are returned, expect an error.
Response time must be < 120ms.</td></tr>
<tr><td>Post-response Scripts</td><td>

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response time is less than 120ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(120);
});
pm.test("Pagination limit is 5", function () {
    const students = pm.response.json();
    pm.expect(students.length).to.be.at.most(5);
});
pm.test("Response contains students ids", function () {
    const students = pm.response.json();
    students.forEach(student => {
        pm.expect(student).to.have.property("id");
    });
});
```

</td></tr>
<tr><td>Pass/Fail</td><td>



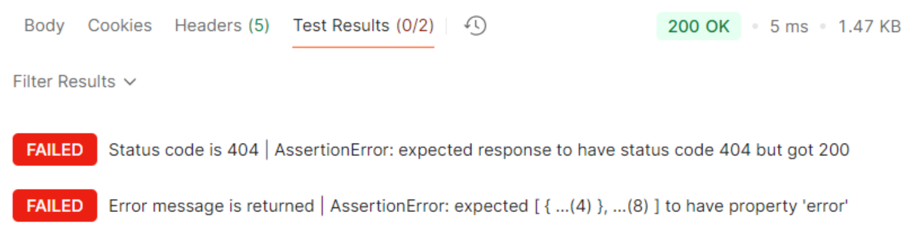
PASSED Status code is 200
PASSED Response time is less than 120ms
FAILED Pagination limit is 5 | AssertionError: expected 9 to be at most 5
PASSED Response contains students ids

</td></tr>
</table>

Comments (if any)	Backend pagination logic does not respect the limit parameter

Test Case	**05**
Description	List students with an invalid filter
URL	http://127.0.0.1:5000/student?program=CSE
Body	None
Method	GET
Choices	Use invalid filter
Representative Values	program=CSE
Constraints	API should return 404 Not Found for an invalid filter. Response should include an error message: Invalid program filter.

Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message is returned", function () {`
	`    const response = pm.response.json();`
	`    pm.expect(response).to.have.property("error", "Invalid program filter");`
	`});`

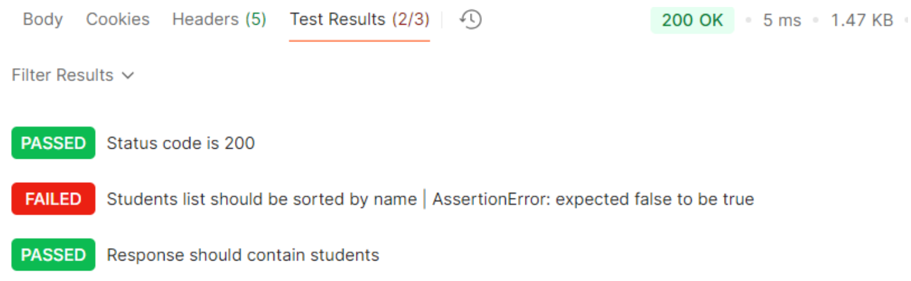
Pass/Fail	 Body Cookies Headers (5) Test Results (0/2) 200 OK 5 ms 1.47 KB Filter Results FAILED Status code is 404 | AssertionError: expected response to have status code 404 but got 200 FAILED Error message is returned | AssertionError: expected [{ …(4) }, …(8)] to have property 'error'
Comments (if any)	Backend is not handling the invalid filter correctly

Test Case	**06**
Description	Verify student list sorting by name
URL	http://127.0.0.1:5000/student?sort=name
Body	None
Method	GET
Choices	Sort by name
Representative Values	sort=name

<table>
<tr><td>Constraints</td><td>API should return students sorted by name (either ascending or descending).
The list should be in alphabetical order (A to Z or Z to A).</td></tr>
<tr><td>Post-response Scripts</td><td>

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Students list should be sorted by name", function () {
    const students = pm.response.json();
    let sortedAsc = true;
    let sortedDesc = true;
    for (let i = 0; i < students.length - 1; i++) {
        if (students[i].name > students[i + 1].name) {
            sortedAsc = false;
            break;
        }
    }
    for (let i = 0; i < students.length - 1; i++) {
        if (students[i].name < students[i + 1].name) {
            sortedDesc = false;
            break;
        }
    }
    pm.expect(sortedAsc || sortedDesc).to.be.true;
});
pm.test("Response should contain students", function () {
    const students = pm.response.json();
    pm.expect(students.length).to.be.above(0);
});
```

</td></tr>
<tr><td>Pass/Fail</td><td>



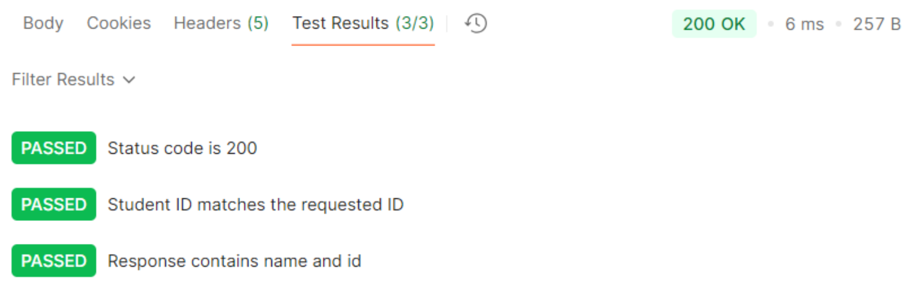
PASSED Status code is 200
FAILED Students list should be sorted by name | AssertionError: expected false to be true
PASSED Response should contain students

</td></tr>
<tr><td>Comments (if any)</td><td>Sorting functionality not working correctly</td></tr>
</table>

Route 2: (/student/{student_id})

Test Case	07
Description	Fetch a student with a valid ID
URL	http://127.0.0.1:5000/student/9
Body	None
Method	GET
Choices	Fetch student by ID
Representative Values	9 (Valid student ID)
Constraints	API should return the student with ID 9. Response must include id and name fields.

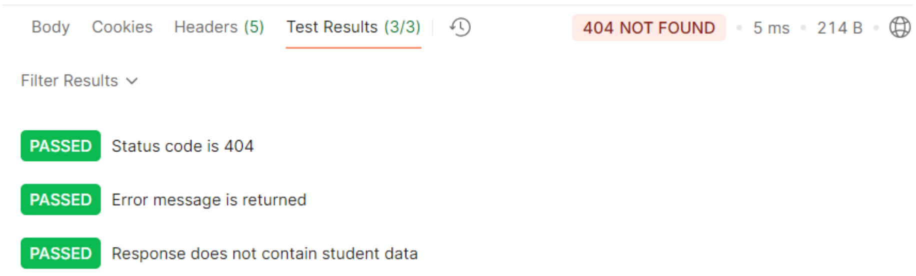
Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Student ID matches the requested ID", function () {`
	`    const student = pm.response.json();`
	`    pm.expect(student).to.have.property("id", 9);`
	`});`
	`pm.test("Response contains name and id", function () {`
	`    const student = pm.response.json();`
	`    pm.expect(student).to.have.property("name");`
	`    pm.expect(student).to.have.property("id");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 200 OK 6 ms 257 B Filter Results PASSED Status code is 200 PASSED Student ID matches the requested ID PASSED Response contains name and id
Comments (if any)	Successfully achieved the goal

Test Case	08
Description	Fetch a student with an invalid ID
URL	http://127.0.0.1:5000/student/100

Body	None
Method	GET
Choices	Invalid ID
Representative Values	100 (Invalid student ID)
Constraints	API should return 404 Not Found for an invalid ID. Response must include error message.

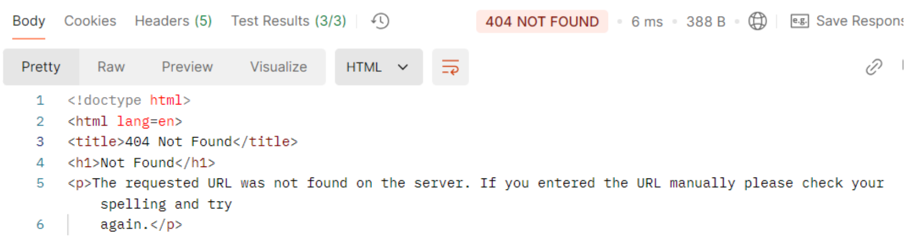
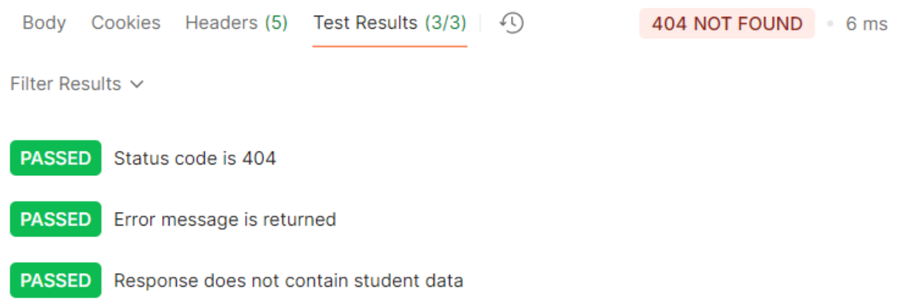
Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message is returned", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Student ID 100 does not exist");`
	`});`
	`pm.test("Response does not contain student data", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.not.have.property("id");`
	`    pm.expect(responseBody).to.not.have.property("name");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) Filter Results 404 NOT FOUND • 5 ms • 214 B PASSED Status code is 404 PASSED Error message is returned PASSED Response does not contain student data
Comments (if any)	Successfully achieved the goal

Test Case	**09**
Description	Fetch student with missing student ID
URL	http://127.0.0.1:5000/student/
Body	None
Method	GET
Choices	Missing ID
Representative Values	No value after /student/

Constraints	Return 404 Not Found for missing ID. Response should contain error message.

Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	
	`pm.test("Error message is returned", function () {`
	`    const responseBody = pm.response.text();`
	`    pm.expect(responseBody).to.include("Not Found");`
	`});`
	
	`pm.test("Response does not contain student data", function () {`
	`    const responseBody = pm.response.text();`
	`    pm.expect(responseBody).not.to.include("id");`
	`    pm.expect(responseBody).not.to.include("name");`
	`});`

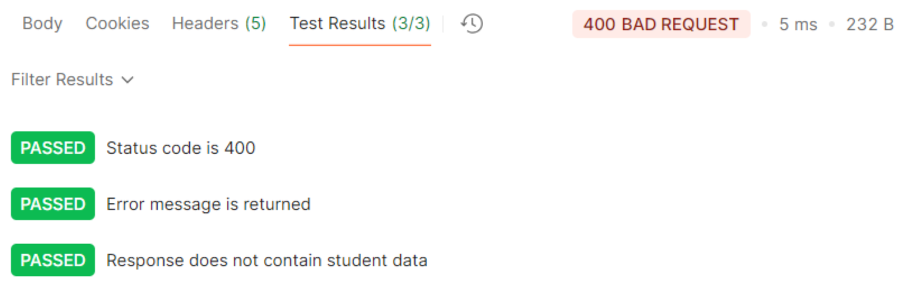
Pass/Fail	 Body
	Cookies
	Headers (5)
	Test Results (3/3)
	404 NOT FOUND
	6 ms
	388 B
	Save Respons
	Pretty
	Raw
	Preview
	Visualize
	HTML
	`<!doctype html>`
	`<html lang=en>`
	`<title>404 Not Found</title>`
	`<h1>Not Found</h1>`
	`<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try`
	`again.</p>`
	 Body
	Cookies
	Headers (5)
	Test Results (3/3)
	404 NOT FOUND
	6 ms
	Filter Results
	PASSED Status code is 404
	PASSED Error message is returned
	PASSED Response does not contain student data

Comments (if any)	Successfully achieved the goal

Test Case	**10**
Description	Fetch student with non-numeric ID
URL	http://127.0.0.1:5000/student/abbabc
Body	None
Method	GET

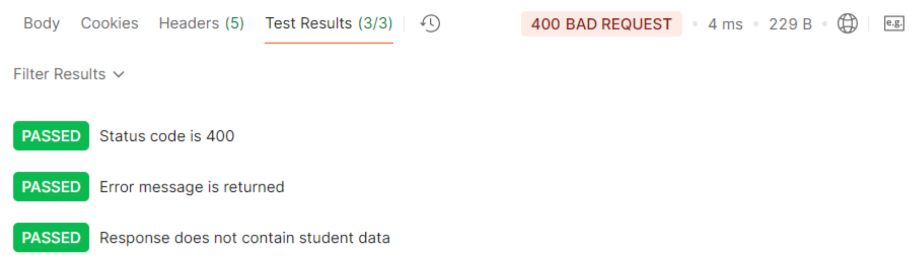
Choices	Non-numeric ID
Representative Values	Non-numeric ID: abbabc
Constraints	Return 400 Bad Request for non-numeric ID. Response should contain a JSON error message.

Post-response Scripts	`pm.test("Status code is 400", function () {`
	`    pm.response.to.have.status(400);`
	`});`
	`pm.test("Error message is returned", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Student ID abbabc is not formatted correctly.");`
	`});`
	`pm.test("Response does not contain student data", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.not.have.property("id");`
	`    pm.expect(responseBody).to.not.have.property("name");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 400 BAD REQUEST 5 ms 232 B Filter Results PASSED Status code is 400 PASSED Error message is returned PASSED Response does not contain student data
Comments (if any)	Goal achieved

Test Case	**11**
Description	Fetch student with special characters
URL	http://127.0.0.1:5000/student/@$%
Body	None
Method	GET
Choices	Special characters in ID
Representative Values	Special Character ID: @$%
Constraints	Return 400 Bad Request for IDs with special characters. Response should contain a JSON error message.

Post-response Scripts	pm.test("Status code is 400", function () {
	    pm.response.to.have.status(400);
	});
	pm.test("Error message is returned", function () {
	    const responseBody = pm.response.json();
	    pm.expect(responseBody).to.have.property("error", "Student ID @$% is not formatted correctly.");
	});
	pm.test("Response does not contain student data", function () {
	    const responseBody = pm.response.json();
	    pm.expect(responseBody).to.not.have.property("id");
	    pm.expect(responseBody).to.not.have.property("name");
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 400 BAD REQUEST 4 ms 229 B Filter Results PASSED Status code is 400 PASSED Error message is returned PASSED Response does not contain student data
Comments (if any)	Goal achieved

Route 3: (/create)

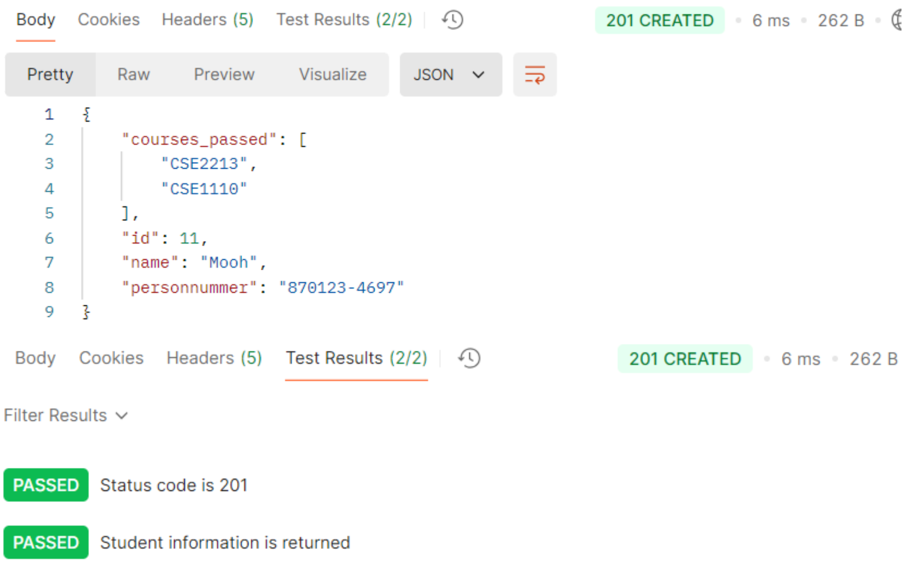
Test Case	12
Description	Valid Student Creation
URL	http://127.0.0.1:5000/create

Body	{
	  "name": "Mooh",
	  "personnummer": "870123-4697",
	  "courses_passed": ["CSE2213", "CSE1110"]
	}

Method	POST
Choices	Valid student creation
Representative Values	Name: Mooh Personnummer: 870123-4697 Courses: ["CSE2213", "CSE1110"]

Constraints	All fields must be provided. personnummer must be unique.

Post-response Scripts	`pm.test("Status code is 201", function () {`
	`    pm.response.to.have.status(201);`
	`});`
	
	`pm.test("Student information is returned", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("name", "Mooh");`
	`    pm.expect(responseBody).to.have.property("personnummer",`
	`"870123-4697");`
	
	`pm.expect(responseBody).to.have.property("courses_passed").that`
	`.is.an("array").and.includes("CSE2213", "CSE1110");`
	`});`

Pass/Fail	 Body
	Cookies
	Headers (5)
	Test Results (2/2)
	201 CREATED
	6 ms
	262 B
	Pretty
	Raw
	Preview
	Visualize
	JSON`{`
	`    "courses_passed": [`
	`        "CSE2213",`
	`        "CSE1110"`
	`    ],`
	`    "id": 11,`
	`    "name": "Mooh",`
	`    "personnummer": "870123-4697"`
	`}`
	 Body
	Cookies
	Headers (5)
	Test Results (2/2)
	201 CREATED
	6 ms
	262 B
	Filter ResultsPASSED Status code is 201
	PASSED Student information is returned

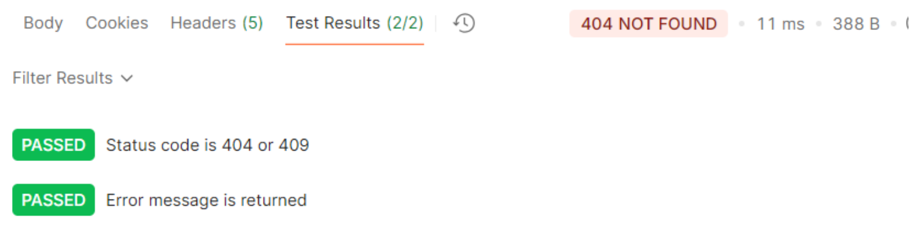
Comments (if any)	Goal Achieved

Test Case	**13**
Description	Create a student with an existing personnummer
URL	http://127.0.0.1:5000/create

Body	`{`
	`  "name": "Niaz",`
	`  "personnummer": "980401-9999",`

	"courses_passed": ["CSE2213", "CSE1110"] }
Method	POST
Choices	Duplicate student creation attempt
Representative Values	Personnummer: 980401-9999 (already exists for id4)
Constraints	personnummer must be unique.

Post-response Scripts	pm.test("Status code is 404 or 409", function () {
	    pm.expect(pm.response.code).to.be.oneOf([404, 409]);
	});
	
	pm.test("Error message is returned", function () {
	    const responseBody = pm.response.text();
	    if (pm.response.code === 409) {
	        pm.expect(responseBody).to.include("Student with this personnummer already exists");
	    } else if (pm.response.code === 404) {
	        pm.expect(responseBody).to.include("Not Found");
	    }
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) Filter Results404 NOT FOUND • 11 ms • 388 B PASSED Status code is 404 or 409 PASSED Error message is returned
Comments (if any)	Backend team should pay more attention

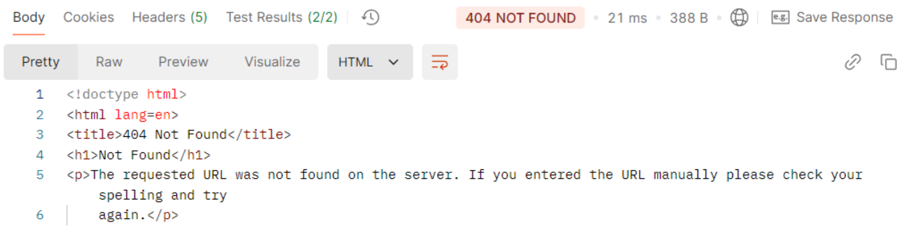
Test Case	**14**
Description	Create a student with invalid data
URL	http://127.0.0.1:5000/create

Body	{
	  "name": "",
	  "personnummer": "abcsd",
	  "courses_passed": ["CSE2213", "CSE1110"]
	}

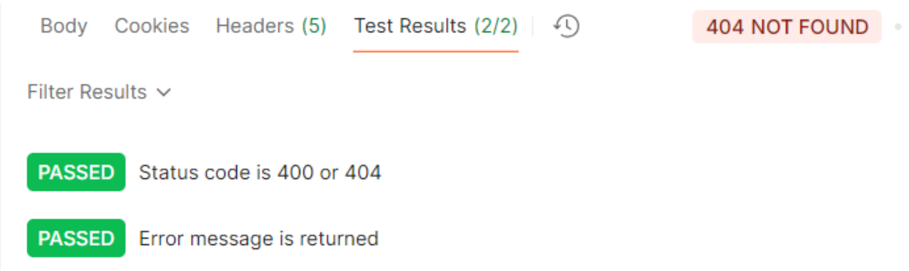
<table>
<tr><td>Method</td><td>POST</td></tr>
<tr><td>Choices</td><td>Invalid input data</td></tr>
<tr><td>Representative Values</td><td>Name: Empty string
Personnummer: abcsd (non-numeric)</td></tr>
<tr><td>Constraints</td><td>Name, Personnummer, and Courses must be valid and non-empty.
Personnummer must be numeric.</td></tr>
<tr><td>Post-response Scripts</td><td>

```
pm.test("Status code is 400 or 404", function () {
    pm.expect(pm.response.code).to.be.oneOf([400, 404]);
});

pm.test("Error message is returned", function () {
    const responseBody = pm.response.text();

    if (pm.response.code === 400) {
        pm.expect(responseBody).to.include("Invalid input data");
    } else if (pm.response.code === 404) {
        pm.expect(responseBody).to.include("Not Found");
    }
});
```

</td></tr>
<tr><td>Pass/Fail</td><td>



```
<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your
    spelling and try
    again.</p>
```



PASSED Status code is 400 or 404

PASSED Error message is returned

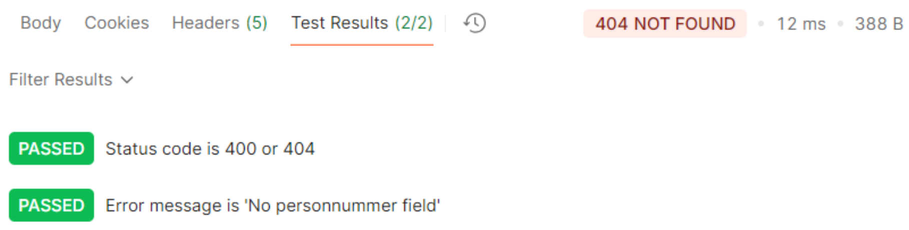
</td></tr>
<tr><td>Comments (if any)</td><td>Backend team should pay more attention</td></tr>
</table>

Test Case	**15**
Description	Create a student with missing required fields
URL	http://127.0.0.1:5000/create

Body	`{`
	`  "name": "Razia",`
	`  "courses_passed": ["CSE2213","SE1110","SE1111", "CSE1110"]`
	`}`

Method	POST
Choices	Missing required fields (personnummer)
Representative Values	Name: Razia Personnummer: Missing Courses: [CSE2213, SE1110, SE1111, CSE1110]
Constraints	All required fields must be provided, including personnummer.

Post-response Scripts	`pm.test("Status code is 400 or 404", function () {`
	`    pm.expect(pm.response.code).to.be.oneOf([400, 404]);`
	`});`
	
	`pm.test("Error message is 'No personnummer field'", function () {`
	`    const responseBody = pm.response.text();`
	`    if (pm.response.code === 400) {`
	`        const jsonResponse = JSON.parse(responseBody);`
	`        pm.expect(jsonResponse.error).to.eql("No personnummer field");`
	`    } else if (pm.response.code === 404) {`
	`        pm.expect(responseBody).to.include("Not Found");`
	`    }`
	`});`

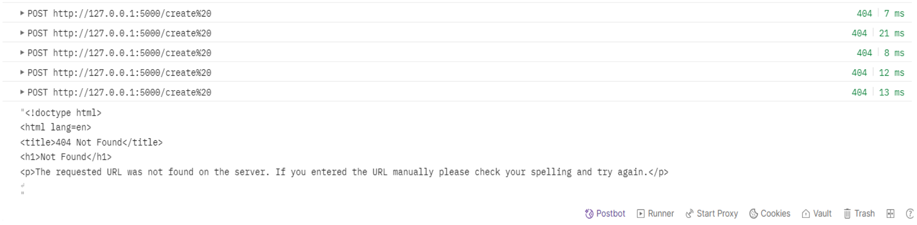
Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND • 12 ms • 388 B Filter Results PASSED Status code is 400 or 404 PASSED Error message is 'No personnummer field'
Comments (if any)	Goal Achieved

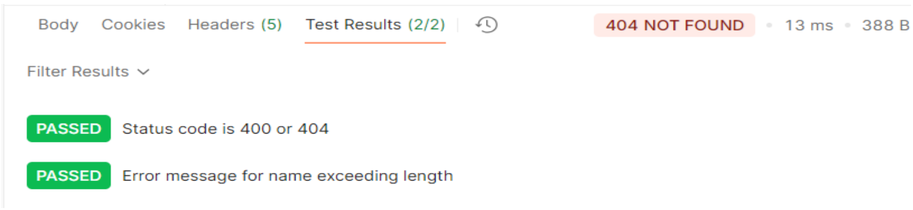
Test Case	**16**
Description	Create a student with a long name
URL	http://127.0.0.1:5000/create

Body	{
	  "name": "Razia Sultana Mohsina Mehnaj Niaz Sami Jahid Hossain Mohsina Razia Rahman Niaz Mohsina Razia",
	  "personnummer": "870123-4697",
	  "courses_passed": ["CSE2213", "CSE1158"]
	}

Method	POST
Choices	Input exceeding length constraints
Representative Values	Name: "Razia Sultana Mohsina Mehnaj Niaz Sami Jahid Hossain Mohsina Razia Rahman Niaz Mohsina Razia" (a long name) Personnummer: "870123-4697" Courses: "CSE2213", "CSE1158"
Constraints	Maximum length of name should be enforced by the API.

Post-response Scripts	pm.test("Status code is 400 or 404", function () {
	    pm.expect(pm.response.code).to.be.oneOf([400, 404]);
	});
	
	pm.test("Error message for name exceeding length", function () {
	    const responseBody = pm.response.text();
	    if (pm.response.code === 400) {
	        const jsonResponse = JSON.parse(responseBody);
	        pm.expect(jsonResponse).to.have.property("error", "Name exceeds maximum allowed length");
	    } else if (pm.response.code === 404) {
	        pm.expect(responseBody).to.include("Not Found");
	    }
	});
	console.log(pm.response.text());

Pass/Fail	 ▸ POST http://127.0.0.1:5000/create%20 404 7 ms ▸ POST http://127.0.0.1:5000/create%20 404 21 ms ▸ POST http://127.0.0.1:5000/create%20 404 8 ms ▸ POST http://127.0.0.1:5000/create%20 404 12 ms ▸ POST http://127.0.0.1:5000/create%20 404 13 ms "<!doctype html> <html lang=en> <title>404 Not Found</title> <h1>Not Found</h1> <p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p> ↲ "

	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND 13 ms 388 B Filter Results PASSED Status code is 400 or 404 PASSED Error message for name exceeding length
Comments (if any)	Goal Achieved

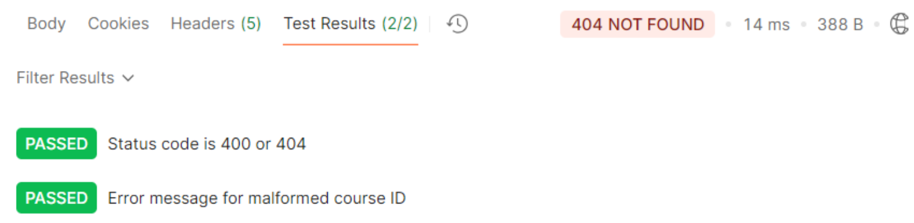
Test Case	**17**
Description	Invalid Course ID Format
URL	http://127.0.0.1:5000/create

Body	`{`
	`  "name": "Niaz Sami",`
	`  "personnummer": "870123-6697",`
	`  "courses_passed": ["CSE2213", "CSEuiux"]`
	`}`

Method	POST
Choices	Invalid course ID
Representative Values	Name: Niaz Sami Personnummer: 870123-6697 Course ID: CSE2213, CSEuiux
Constraints	The last part of the course ID should consist of numeric values ("CSE2213" is valid, but "CSEuiux" is invalid).

Post-response Scripts	`pm.test("Status code is 400 or 404", function () {`
	`    pm.expect(pm.response.code).to.be.oneOf([400, 404]);`
	`});`
	
	`pm.test("Error message for malformed course ID", function () {`
	`    const responseBody = pm.response.text();`
	
	`    if (pm.response.code === 400) {`
	`        try {`
	`            const jsonResponse = JSON.parse(responseBody);`
	
	`pm.expect(jsonResponse).to.have.property("error").that.includes("Malformed course ID");`
	`        } catch (e) {`

	`pm.expect.fail("Expected JSON response but got HTML: " + responseBody);`
	`}`
	`} else if (pm.response.code === 404) {`
	`pm.expect(responseBody).to.include("Not Found");`
	`}`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) Filter Results 404 NOT FOUND · 14 ms · 388 B PASSED Status code is 400 or 404 PASSED Error message for malformed course ID
Comments (if any)	Goal Achieved

Route 4: (/update/{student_id})

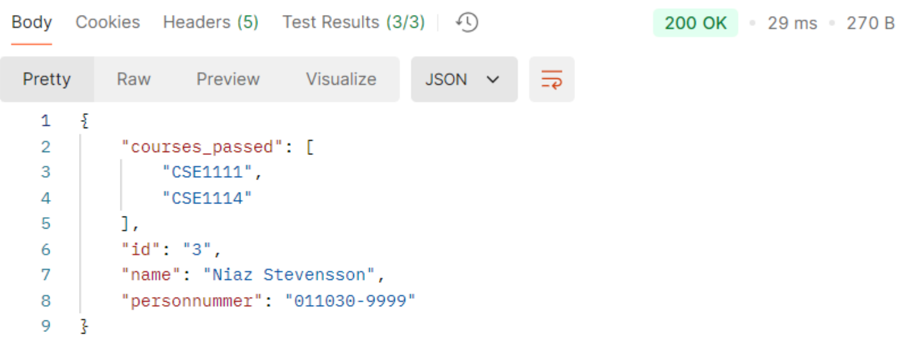
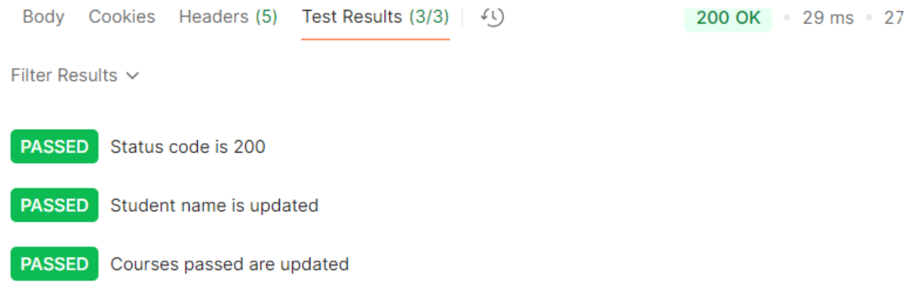
Test Case	**18**
Description	Update a student name and courses
URL	http://127.0.0.1:5000/update/3

Body	`{`
	`"name": "Niaz Stevensson",`
	`"personnummer": "011030-9999",`
	`"courses_passed": ["CSE1111", "CSE1114"]`
	`}`

Method	PUT
Choices	Update student name and courses
Representative Values	Student ID: 3 Name: "Niaz Stevensson" Personnummer: "011030-9999" Courses Passed: ["CSE1111", "CSE1114"]
Constraints	The personnummer cannot be modified. The name and courses_passed can be updated.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`pm.response.to.have.status(200);`
	`});`
	`pm.test("Student name is updated", function () {`

	const responseBody = pm.response.json();
	pm.expect(responseBody).to.have.property("name", "Niaz Stevensson");
	});
	pm.test("Courses passed are updated", function () {
	const responseBody = pm.response.json();
	pm.expect(responseBody.courses_passed).to.include("CSE1111");
	pm.expect(responseBody.courses_passed).to.include("CSE1114");
	});

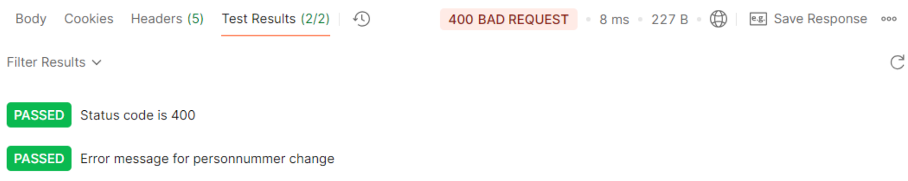
Pass/Fail	 Body
	Cookies
	Headers (5)
	Test Results (3/3)
	200 OK
	29 ms
	270 B
	Pretty
	Raw
	Preview
	Visualize
	JSON
	{
	"courses_passed": [
	"CSE1111",
	"CSE1114"
	],
	"id": "3",
	"name": "Niaz Stevensson",
	"personnummer": "011030-9999"
	}
	 Body
	Cookies
	Headers (5)
	Test Results (3/3)
	200 OK
	29 ms
	27
	Filter Results
	PASSED Status code is 200
	PASSED Student name is updated
	PASSED Courses passed are updated

Comments (if any)	None

Test Case	**19**
Description	Update a student personnummer
URL	http://127.0.0.1:5000/update/4
Body	{ "name": "Yue Sakamoto", "personnummer": "980401-9997", "courses_passed": ["DSE1001", "DSE1003", "DSE1004", "DSE2105", "DSE2116", "DSE3217", "DSE3218", "DSE3239", "DSE4100"]

	}
Method	PUT
Choices	Update personnummer
Representative Values	Student ID: 4 Name: "Yue Sakamoto" Personnummer: "980401-9997" Courses Passed: ["DSE1001", "DSE1003", "DSE1004", "DSE2105", "DSE2116", "DSE3217", "DSE3218", "DSE3239", "DSE4100"]
Constraints	The personnummer cannot be modified, and an error should be returned if attempted.

Post-response Scripts	pm.test("Status code is 400", function () {
	    pm.response.to.have.status(400);
	});
	pm.test("Error message for personnummer change", function () {
	    const responseBody = pm.response.json();
	    pm.expect(responseBody).to.have.property("error", "Changes to personnummer are not allowed.");
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST 8 ms 227 B Save Response Filter Results PASSED Status code is 400 PASSED Error message for personnummer change
Comments (if any)	None

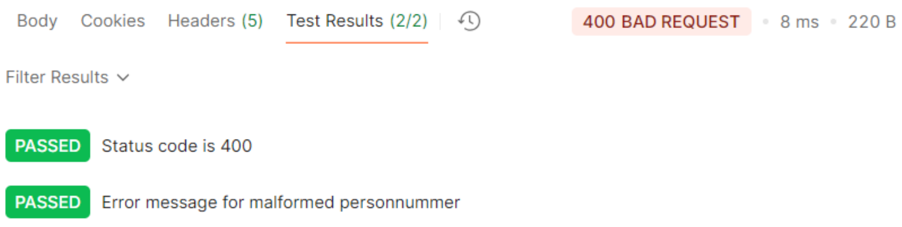
Test Case	**20**
Description	Update a non-existent student record
URL	http://127.0.0.1:5000/update/15

Body	{
	  "name": "Abul",
	  "personnummer": "980401-100",
	  "courses_passed": ["DSE1001","DSE4100"]
	}

Method	PUT
Choices	Update Non-existent
Representative Values	Student ID: 15 (does not exist) Name: "Abul"

	Personnummer: "980401-100" Courses Passed: ["DSE1001", "DSE4100"]
Constraints	The student ID must exist in the database. If it does not, an error should be returned.

Post-response Scripts	`pm.test("Status code is 400", function () {`
	`    pm.response.to.have.status(400);`
	`});`
	
	`pm.test("Error message for malformed personnummer", function ()`
	`{`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error",`
	`"Malformed personnummer 980401-100");`
	`});`

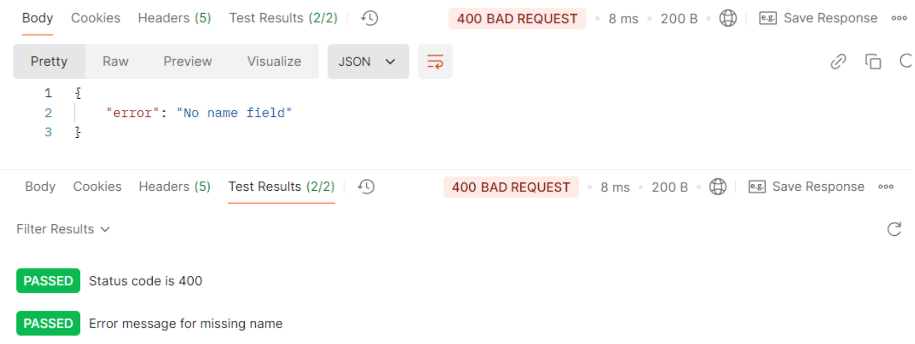
Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST 8 ms 220 B Filter Results PASSED Status code is 400 PASSED Error message for malformed personnummer
Comments (if any)	None

Test Case	**21**
Description	Update a student record with missing name field
URL	http://127.0.0.1:5000/update/2

Body	`{`
	`  "personnummer": "990111-9999",`
	`  "courses_passed": ["CSE2213", "CSE1110"]`
	`}`

Method	PUT
Choices	Update with missing name
Representative Values	Student ID: 2 Personnummer: "990111-9999" Courses Passed: ["CSE2213", "CSE1110"] Missing Name field
Constraints	The name field is mandatory. If it is missing, an error should be returned.
Post-response Scripts	`pm.test("Status code is 400", function () {`

	`    pm.response.to.have.status(400);`
	`});`
	`pm.test("Error message for missing name", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "No name field");`
	`});`

Pass/Fail	
	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST 8 ms 200 B Save Response
	Pretty Raw Preview Visualize JSON
	`{`
	`    "error": "No name field"`
	`}`
	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST 8 ms 200 B Save Response
	Filter Results
	PASSED Status code is 400
	PASSED Error message for missing name

Comments (if any)	None

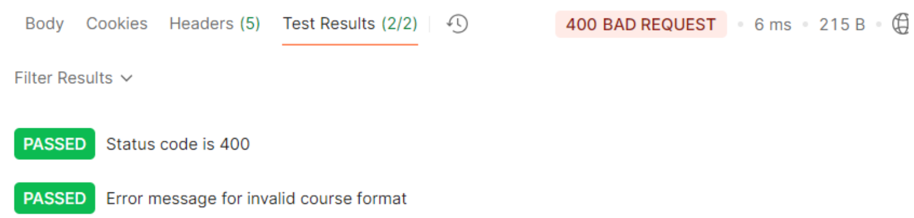
Test Case	**22**
Description	Update a student record with an invalid course format
URL	http://127.0.0.1:5000/update/6

Body	`{`
	`  "name": "Amy Pond",`
	`  "personnummer": "020201-9999",`
	`  "courses_passed": ["CSE1111", "CSEaaaa"]`
	`}`

Method	PUT
Choices	Update with invalid course format
Representative Values	Student ID: 6 Name: Amy Pond Personnummer: "020201-9999" Courses Passed: ["CSE1111", "CSEaaaa"]
Constraints	Courses must follow the correct format, with the last part being numeric. If invalid, an error should be returned.

Post-response Scripts	`pm.test("Status code is 400", function () {`
	`    pm.response.to.have.status(400);`
	`});`

	pm.test("Error message for invalid course format", function () {
	    const responseBody = pm.response.json();
	    pm.expect(responseBody).to.have.property("error", "Malformed course ID: CSEaaaa");
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST • 6 ms • 215 B Filter Results PASSED Status code is 400 PASSED Error message for invalid course format
Comments (if any)	None

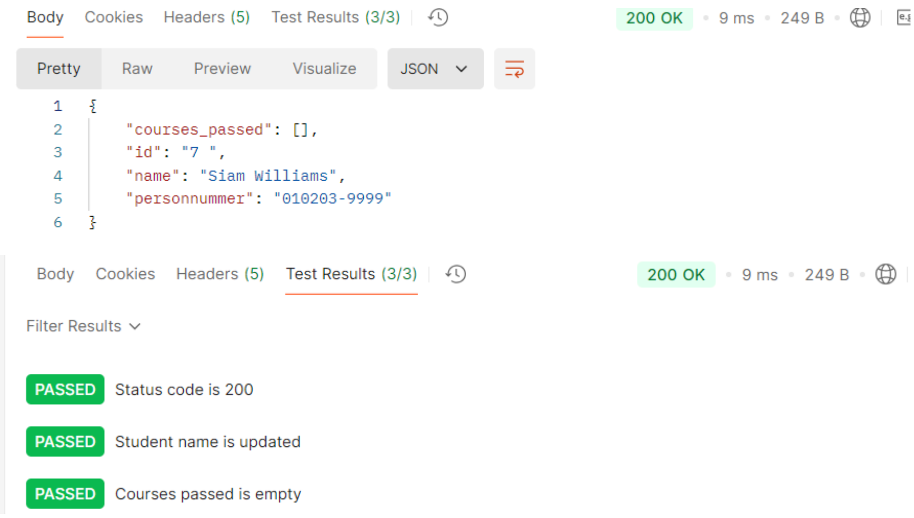
Test Case	23
Description	Update a student record with an empty courses_passed array
URL	http://127.0.0.1:5000/update/7

Body	{
	  "name": "Siam Williams",
	  "personnummer": "010203-9999",
	  "courses_passed": []
	}

Method	PUT
Choices	Update courses to an empty array
Representative Values	Student ID: 7 Name: Siam Williams Personnummer: "010203-9999" Courses Passed: []
Constraints	The courses_passed array can be empty.

Post-response Scripts	pm.test("Status code is 200", function () {
	    pm.response.to.have.status(200);
	});
	pm.test("Student name is updated", function () {
	    const responseBody = pm.response.json();
	    pm.expect(responseBody).to.have.property("name", "Siam Williams");
	});

	`pm.test("Courses passed is empty", function () {`
	`    const responseBody = pm.response.json();`
	
	`pm.expect(responseBody.courses_passed).to.be.an('array').that.is.empty;`
	`});`

Pass/Fail	
	 Body Cookies Headers (5) Test Results (3/3) 200 OK 9 ms 249 B
	Pretty Raw Preview Visualize JSON
	`{`
	`    "courses_passed": [],`
	`    "id": "7 ",`
	`    "name": "Siam Williams",`
	`    "personnummer": "010203-9999"`
	`}`
	 Body Cookies Headers (5) Test Results (3/3) 200 OK 9 ms 249 B
	Filter Results
	PASSED Status code is 200
	PASSED Student name is updated
	PASSED Courses passed is empty

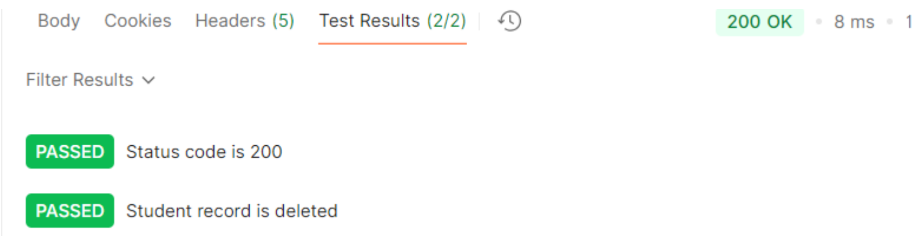
Comments (if any)	None

Route 5: (/delete/{student_id})

Test Case	24
Description	Delete an existing student record
URL	http://127.0.0.1:5000/delete/8
Body	None
Method	DELETE
Choices	Delete a student record
Representative Values	Student ID: 8 (Existing student)
Constraints	The student ID must exist in the database for deletion.

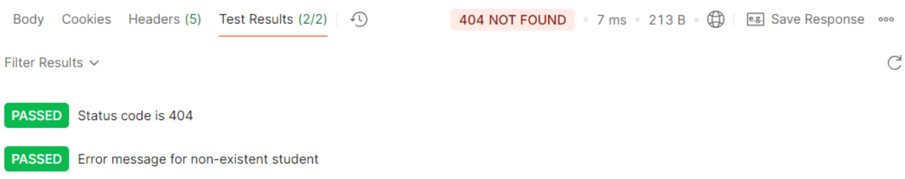
Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Student record is deleted", function () {`
	`    const responseBody = pm.response.json();`

	`pm.expect(responseBody).to.have.property("deleted", "8");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 200 OK • 8 ms • 1 Filter Results PASSED Status code is 200 PASSED Student record is deleted
Comments (if any)	Goal Achieved

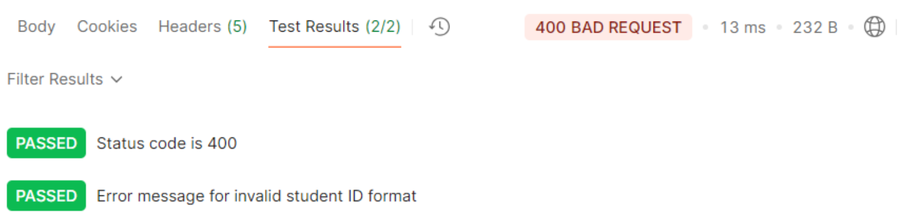
Test Case	**25**
Description	Delete a non-existent student record
URL	http://127.0.0.1:5000/delete/12
Body	None
Method	DELETE
Choices	Delete a student record
Representative Values	Student ID: 12 (Non-existing student)
Constraints	The student ID must exist in the database for deletion.

Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message for non-existent student", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Student ID 12 does not exist");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND • 7 ms • 213 B • Save Response Filter Results PASSED Status code is 404 PASSED Error message for non-existent student
Comments (if any)	Goal Achieved

Test Case	**26**
Description	Attempt to delete a student record with invalid ID format
URL	http://127.0.0.1:5000/delete/lala
Body	None
Method	DELETE
Choices	Delete a student record using invalid ID format
Representative Values	Student ID: lala (non-numeric)
Constraints	Student ID should be numeric.

Post-response Scripts	`pm.test("Status code is 400", function () {`
	`    pm.response.to.have.status(400);`
	`});`
	`pm.test("Error message for invalid student ID format", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody.error).to.include("not formatted correctly");`
	`});`

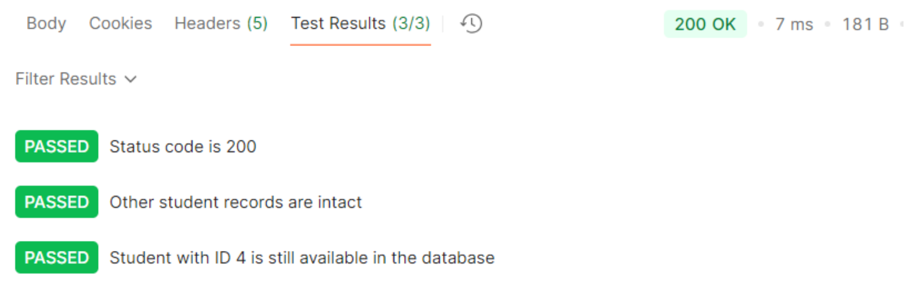
Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 400 BAD REQUEST • 13 ms • 232 B Filter Results PASSED Status code is 400 PASSED Error message for invalid student ID format
Comments (if any)	Goal Achieved

Test Case	**27**
Description	Delete a student record but check the integrity of the remaining records
URL	http://127.0.0.1:5000/delete/3
Body	None
Method	DELETE
Choices	Integrity check of Student ID
Representative Values	Student ID: 3 (existing student)
Constraints	Deleting one student should not affect other records.

<table>
<tr><td>Post-response Scripts</td><td>

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Other student records are intact", function () {
    const responseBody = pm.response.json();
    pm.expect(responseBody).to.have.property("deleted", "3");
});

pm.sendRequest({
    url: 'http://127.0.0.1:5000/student/4',
    method: 'GET'
}, function (err, res) {
    pm.test("Student with ID 4 is still available in the
database", function () {
        pm.expect(res).to.have.property('code', 200);
        const responseBody = res.json();
        pm.expect(responseBody).to.have.property("id", 4);
        pm.expect(responseBody).to.have.property("name");

pm.expect(responseBody).to.have.property("courses_passed").that
.is.an("array");
    });
});
```

</td></tr>
<tr><td>Pass/Fail</td><td>



PASSED Status code is 200

PASSED Other student records are intact

PASSED Student with ID 4 is still available in the database

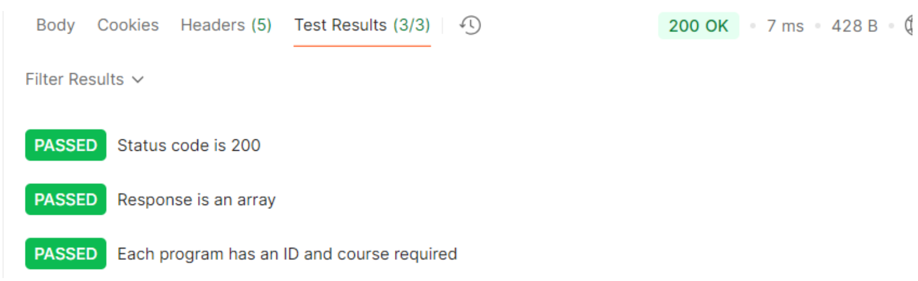
</td></tr>
<tr><td>Comments (if any)</td><td>None</td></tr>
</table>

Route 6: (/program)

Test Case	**28**
Description	Retrieve a list of all programs successfully
URL	http://127.0.0.1:5000/program
Body	None

Method	GET
Choices	Fetch all programs
Representative Values	Programs
Constraints	The response should return an array of objects, each containing id and courses_required.

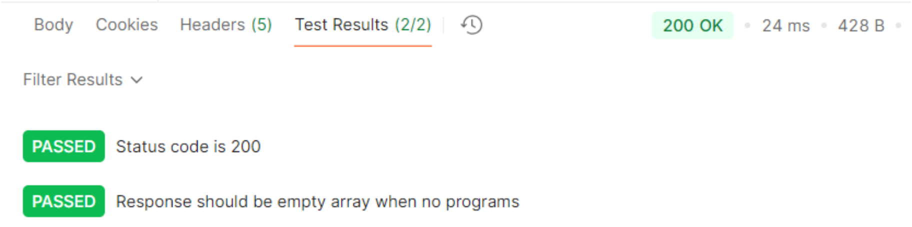
Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Response is an array", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.be.an('array');`
	`});`
	`pm.test("Each program has an ID and course required", function`
	`() {`
	`    const responseBody = pm.response.json();`
	`    responseBody.forEach(program => {`
	`        pm.expect(program).to.have.property("id");`
	
	`pm.expect(program).to.have.property("courses_required");`
	`    });`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 200 OK 7 ms 428 B Filter Results PASSED Status code is 200 PASSED Response is an array PASSED Each program has an ID and course required
Comments (if any)	None

Test Case	**29**
Description	Verify response when no programs are available
URL	http://127.0.0.1:5000/program
Body	None
Method	GET
Choices	Return an empty array if no programs exist

Representative Values	No programs available in the database
Constraints	The response should be an empty array when no programs exist.

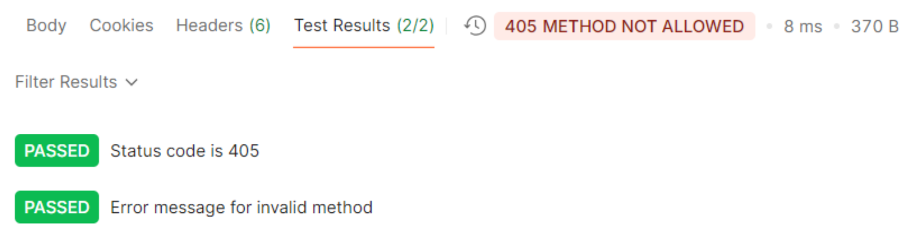
Post-response Scripts	pm.test("Status code is 200", function () {
	    pm.response.to.have.status(200);
	});
	
	pm.test("Response should be empty array when no programs",
	function () {
	    const responseBody = pm.response.json();
	    if (responseBody.length === 0) {
	
	pm.expect(responseBody).to.be.an('array').that.is.empty;
	    } else {
	        pm.expect(responseBody.length).to.be.above(0);
	    }
	});

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 200 OK 24 ms 428 B Filter Results PASSED Status code is 200 PASSED Response should be empty array when no programs
Comments (if any)	Successfully achieved the goal

Test Case	**30**
Description	Verify response for an invalid request method
URL	http://127.0.0.1:5000/program
Body	None
Method	POST
Choices	Return an error for unsupported HTTP methods
Representative Values	Unsupported HTTP method POST instead of GET
Constraints	Only the GET method should be allowed for this endpoint.

Post-response Scripts	pm.test("Status code is 405", function () {
	    pm.response.to.have.status(405);
	});

	pm.test("Error message for invalid method", function () {
	    const responseBody = pm.response.text();
	    try {
	        const jsonResponse = JSON.parse(responseBody);
	        pm.expect(jsonResponse).to.have.property("error", "Method not allowed");
	    } catch (e) {
	        pm.expect(responseBody).to.include("Method Not Allowed");
	    }
	});

Pass/Fail	 Body Cookies Headers (6) Test Results (2/2) 405 METHOD NOT ALLOWED 8 ms 370 B Filter Results PASSED Status code is 405 PASSED Error message for invalid method
Comments (if any)	Backend team should pay more attention

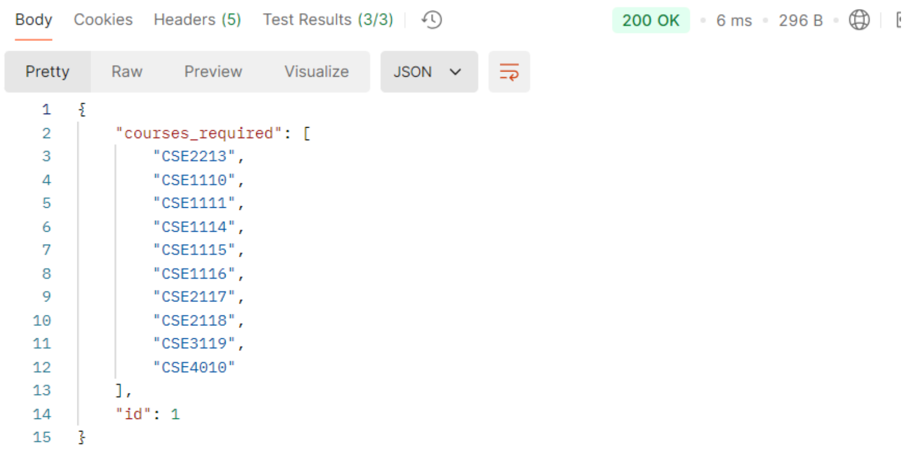
Route 7: (/program/{program_id})

Test Case	**31**
Description	Retrieve required courses for program ID 1
URL	http://127.0.0.1:5000/program/1
Body	None
Method	GET
Choices	Retrieve courses for program ID 1
Representative Values	Program ID: 1 Courses: ["CSE2213", "CSE1110", "CSE1111", "CSE1114", "CSE1115", "CSE1116", "CSE2117", "CSE2118", "CSE3119", "CSE4010"]
Constraints	Program ID must exist. Response should list required courses.

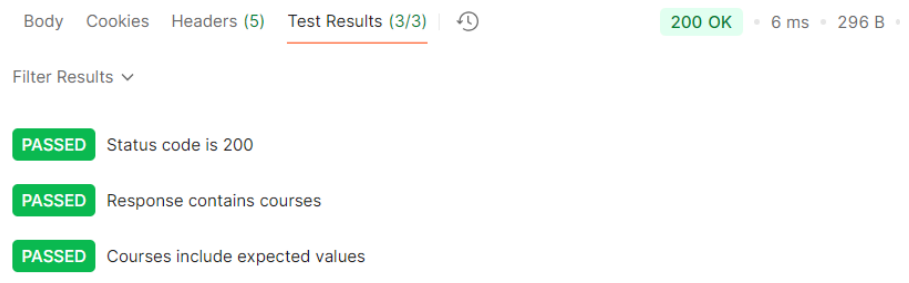
Post-response Scripts	pm.test("Status code is 200", function () {
	    pm.response.to.have.status(200);
	});
	pm.test("Response contains courses", function () {

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

pm.expect(responseBody).to.have.property("courses_required").that.is.an("array").that.is.not.empty;
});
pm.test("Courses include expected values", function () {
    const responseBody = pm.response.json();
pm.expect(responseBody.courses_required).to.include.members(["CSE2213", "CSE1110", "CSE1111", "CSE1114", "CSE1115", "CSE1116", "CSE2117", "CSE2118", "CSE3119", "CSE4010"]);
});
```

Pass/Fail



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



PASSED Status code is 200

PASSED Response contains courses

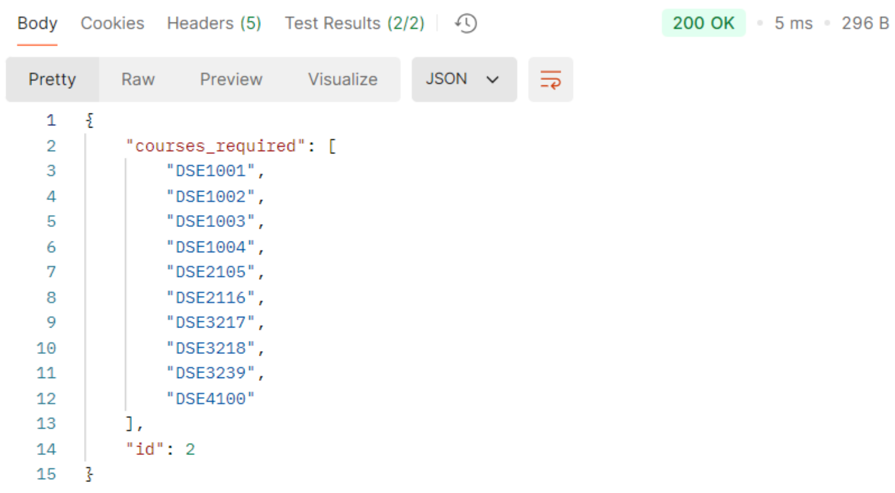
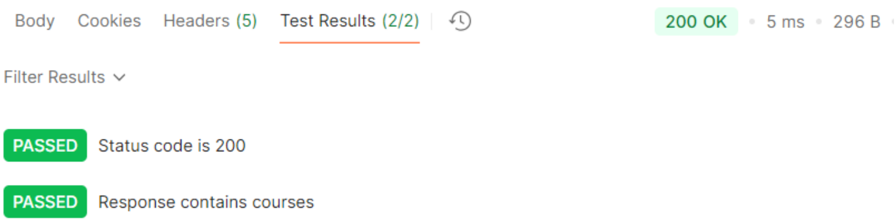
PASSED Courses include expected values

Comments (if any)	None

Test Case	**32**
Description	Retrieve required courses for program ID 2
URL	http://127.0.0.1:5000/program/2
Body	None
Method	GET
Choices	Retrieve courses for program ID 2

Representative Values	Program ID: 2
Constraints	Program ID must exist. Response should list required courses.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Response contains courses", function () {`
	`    const responseBody = pm.response.json();`
	
	`pm.expect(responseBody).to.have.property("courses_required").that.is.an("array").that.is.not.empty;`
	`});`

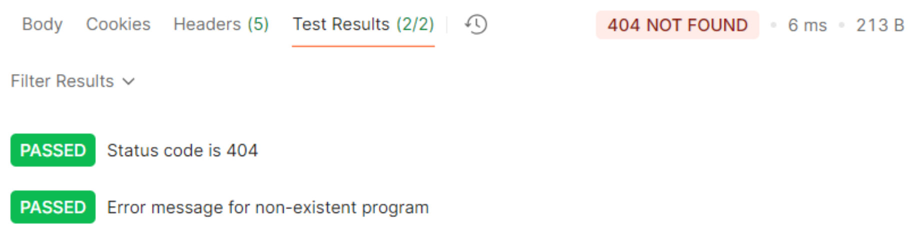
Pass/Fail	
	 Body Cookies Headers (5) Test Results (2/2)
	
	 Pretty Raw Preview Visualize JSON
	`{`
	`    "courses_required": [`
	`        "DSE1001",`
	`        "DSE1002",`
	`        "DSE1003",`
	`        "DSE1004",`
	`        "DSE2105",`
	`        "DSE2116",`
	`        "DSE3217",`
	`        "DSE3218",`
	`        "DSE3239",`
	`        "DSE4100"`
	`    ],`
	`    "id": 2`
	`}`
	
	
	PASSED Status code is 200
	PASSED Response contains courses

Comments (if any)	None

Test Case	**33**
Description	Retrieve required courses for a non-existent program ID
URL	http://127.0.0.1:5000/program/22
Body	None
Method	GET

Choices	Attempt to retrieve courses for program ID 22 (non-existent)
Representative Values	Program ID: 22 (non-existent) Expected Error: "Program ID 22 does not exist"
Constraints	Program ID must exist.

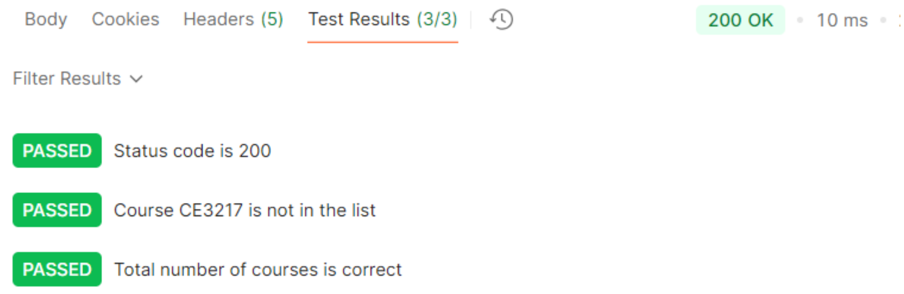
Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message for non-existent program", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Program ID 22 does not exist");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND 6 ms 213 B Filter Results PASSED Status code is 404 PASSED Error message for non-existent program
Comments (if any)	None

Test Case	**34**
Description	Ensure that the course CE3217 is not included in the required courses for program 2
URL	http://127.0.0.1:5000/program/2
Body	None
Method	GET
Choices	Verify CE3217 is not in the required courses list
Representative Values	Program ID: 2 Course: CE3217
Constraints	CE3217 should not be in the required courses.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	`pm.test("Course CE3217 is not in the list", function () {`
	`    const responseBody = pm.response.json();`

	`pm.expect(responseBody.courses_required).to.not.include("CE3217");`
	`});`
	`pm.test("Total number of courses is correct", function () {`
	`    const responseBody = pm.response.json();`
	`pm.expect(responseBody.courses_required.length).to.equal(10);`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (3/3) 200 OK 10 ms Filter Results PASSED Status code is 200 PASSED Course CE3217 is not in the list PASSED Total number of courses is correct
Comments (if any)	None

Route 8: (/finished/{student_id}/{program_id})

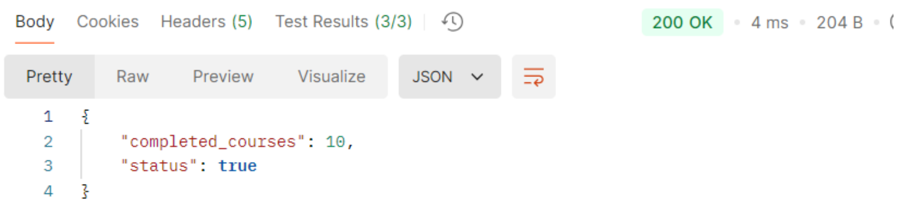
Test Case	**35**
Description	Check if student 1 is eligible to graduate from program 1
URL	http://127.0.0.1:5000/finished/1/1
Body	None
Method	GET
Choices	Check graduation eligibility for student 1 in program 1
Representative Values	Student ID: 1 Program ID: 1
Constraints	The student must meet the graduation requirements.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`    pm.response.to.have.status(200);`
	`});`
	
	`pm.test("Completed courses and status are returned", function () {`
	`    const responseBody = pm.response.json();`

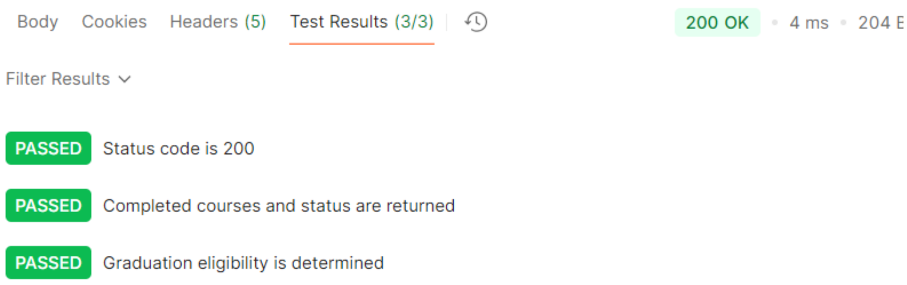
<table>
<tr>
<td></td>
<td>

```
pm.expect(responseBody).to.have.property("completed_courses");
    pm.expect(responseBody).to.have.property("status");

pm.expect(responseBody.completed_courses).to.be.a("number");
    pm.expect(responseBody.status).to.be.a("boolean");
});
pm.test("Graduation eligibility is determined", function () {
    const responseBody = pm.response.json();
    if (responseBody.status === true) {
        pm.expect(responseBody.completed_courses).to.equal(10);
    } else {

pm.expect(responseBody.completed_courses).to.be.below(10);
    }
});
```

</td>
</tr>
<tr>
<td>Pass/Fail</td>
<td>



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



PASSED Status code is 200

PASSED Completed courses and status are returned

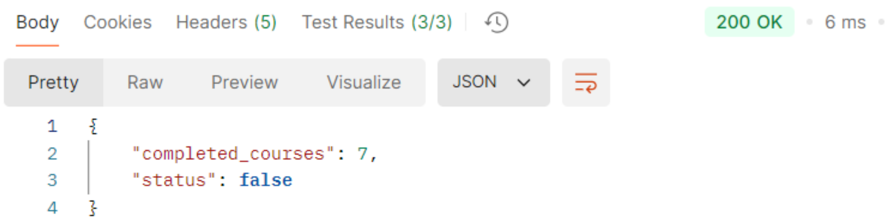
PASSED Graduation eligibility is determined

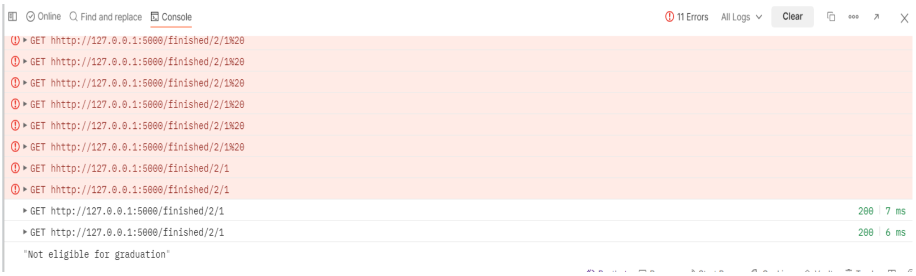
</td>
</tr>
<tr>
<td>Comments (if any)</td>
<td>None</td>
</tr>
</table>

Test Case	**36**
Description	Check if a student isn’t eligible for graduation due to incomplete courses
URL	http://127.0.0.1:5000/finished/2/1
Body	None
Method	GET
Choices	Check if student 2 is not eligible due to incomplete courses

Representative Values	Student ID: 2 Program ID: 1
Constraints	Student must complete all courses to graduate.

Post-response Scripts	`pm.test("Status code is 200", function () {`
	`pm.response.to.have.status(200);`
	`});`
	
	`pm.test("Completed courses and status are returned", function`
	`() {`
	`const responseBody = pm.response.json();`
	
	`pm.expect(responseBody).to.have.property("completed_courses");`
	`pm.expect(responseBody).to.have.property("status");`
	
	`pm.expect(responseBody.completed_courses).to.be.a("number");`
	`pm.expect(responseBody.status).to.be.a("boolean");`
	`});`
	
	`pm.test("Graduation eligibility is determined", function () {`
	`const responseBody = pm.response.json();`
	`if (responseBody.status === true) {`
	`console.log("Eligible for graduation");`
	`pm.expect(responseBody.completed_courses).to.equal(10);`
	`} else {`
	`console.log("Not eligible for graduation");`
	
	`pm.expect(responseBody.completed_courses).to.be.below(10);`
	`}`
	`});`

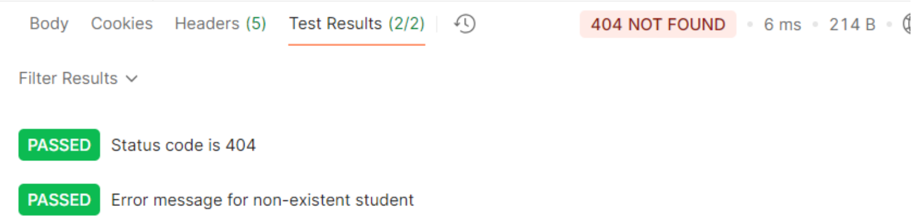
Pass/Fail	 Body
	Cookies
	Headers (5)
	Test Results (3/3)
	200 OK
	6 ms
	Pretty
	Raw
	Preview
	Visualize
	JSON
	`{`
	`"completed_courses": 7,`
	`"status": false`
	`}`

	
	Online Find and replace Console 11 Errors All Logs Clear
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1%20`
	`GET hhttp://127.0.0.1:5000/finished/2/1`
	`GET hhttp://127.0.0.1:5000/finished/2/1`
	`GET http://127.0.0.1:5000/finished/2/1    200 7 ms`
	`GET http://127.0.0.1:5000/finished/2/1    200 6 ms`
	`"Not eligible for graduation"`

Comments (if any)	Backend team should pay more attention

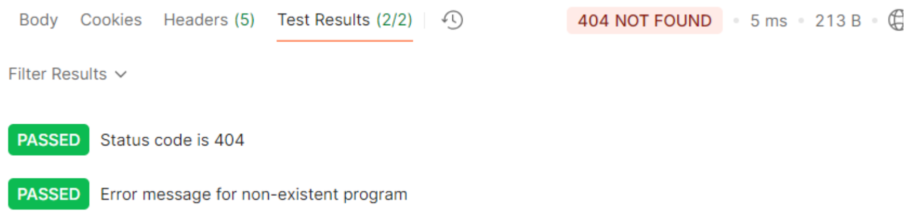
Test Case	**37**
Description	Check for graduation status of a non-existent student
URL	http://127.0.0.1:5000/finished/100/2
Body	None
Method	GET
Choices	Request status for non-existent student (ID: 100) in program 2
Representative Values	Student ID: 100 (non-existent) Program ID: 2
Constraints	Student ID must exist.

Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message for non-existent student", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Student ID 100 does not exist");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND 6 ms 214 B Filter Results PASSED Status code is 404 PASSED Error message for non-existent student
Comments (if any)	Backend team should pay more attention

Test Case	38
Description	Check eligibility for a non-existent program
URL	http://127.0.0.1:5000/finished/6/99
Body	None
Method	GET
Choices	Check eligibility for student 6 in non-existent program 99
Representative Values	Student ID: 6 Program ID: 99 (non-existent)
Constraints	Program ID must exist.

Post-response Scripts	`pm.test("Status code is 404", function () {`
	`    pm.response.to.have.status(404);`
	`});`
	`pm.test("Error message for non-existent program", function () {`
	`    const responseBody = pm.response.json();`
	`    pm.expect(responseBody).to.have.property("error", "Program ID 99 does not exist");`
	`});`

Pass/Fail	 Body Cookies Headers (5) Test Results (2/2) 404 NOT FOUND 5 ms 213 B Filter Results PASSED Status code is 404 PASSED Error message for non-existent program
Comments (if any)	None

—-----------------------------------THE END—----------------------------------