FINÁLNÍ PROJEKT č.1

Autor: Alexandra Viktorenko

Contents

ZADÁNÍ

Cílem finálního projektu je otestovat funkčnost aplikace, která slouží k manipulaci s daty o studentech. Aplikace má rozhraní REST-API, které umožňuje vytvoření, smazání a získání dat..

Přístupové údaje:

Databáze	database: Host: Username: Password:
REST-API	http://

Poznamky:

Nezapomeňte, že v IT se data musí někde uložit a poté získat. Proto ověřte, že data jsou správně uložena a získávána z databáze.
Nezapomeňte do testovacích scénářů uvést testovací data, očekávaný výsledek včetně těla odpovědi a stavových kódů.

INITIAL ANALYSIS

The task is to test a REST API mediating access to student information database. As specified in requirements, API accepts requests in **3 methods**:

1. GET – returns student information stored in DB
2. POST – submits new student entry
3. DELETE – removes a student entry

Investigation has shown that GET request can be sent in **two modes**:

1. Calling the endpoint directly (this returns all stored students data)
2. ID (integer) parameter is appended to endpoint path separated by „ / “ (this returns the data of a student with specified ID)

POST request can be sent with or without providing **„id“** field.

- If the user with specified ID exists, their data is edited to the values provided in the request.
- If not, ID field is ignored.

DELETE request can be sent only in {{path}}/{{id}} format.

As REST API practice dictates, GET and DELETE requests are sent without payload. Request and response payloads are in JSON format.

This leaves us with 5 distinct features:

1. GET request without student ID
2. GET request with student ID
3. POST request with payload containing new student data and no/new ID
4. POST request with payload containing updated student data and existing ID
5. DELETE request with student ID

The following risks have been identified per feature and for the tested product in general:

Risk	Probability	Severity	Mitigation
Correct request does not succeed (e.g., **POST** request with all mandatory fields returns Unexpected error)	Low	Critical	Team: design discussion, documentation
			QA: full coverage of happy paths
Correct request succeeds, but incorrectly (e.g., **GET** request with student ID returns wrong student data)	High	Critical	Team: design discussion, documentation
			QA: full coverage of happy paths, detailed test assertions
Incorrect request succeeds (e.g., **GET** request with non-existent ID returns some student data)	Medium	Critical	Team: unhappy paths, misformated inputs and fraud cases discussed and documented, sanity checks ran before submitting to QA
			QA: full coverage of discussed cases
Incorrect request fails, but user is not properly notified (e.g., **GET** request with non-existent ID returns Unexpected error or Bad request with incorrect error message)	High	Medium	Team: well-written validations
			QA: validation cases covered, data format exploratory tests ran
If too much data is present, **GET** request takes too long to execute	High	Low	Team: non-functional requirements well defined, pagination is implemented
			QA: performance test is ran

Risk	Probability	Severity	Mitigation
If request rate is too high, correct requests throttle or fail	Medium	Low	Team: non-functional requirements well defined, performance alerts in place
			QA: performance test is ran
Correct request with successful response makes incorrect changes to the DB (e.g., old data gets rewritten, wrong entry gets deleted)	Low	High	Team: integration tests
			QA: every step verified against DB
API is misused	Low	High	Team: security standards implemented, data safety ensured, authentication implemented
			QA: run tests within security standards
Failure on the side of hosting or DB provider	Very low	Critical	Team: infrastructure alerts in place
			QA: Failover tests run
Application is not user-friendly	Medium	Low	Team: UX design, A/B tests
			QA: UI tests, focus tests Not applicable
If new feature is introduced, failure in old fuctionality occurs	High	High	Team: Unit tests
			QA: Regression tests, automation

Given there is no documentation and no other technical requirements specified, the rest will be discovered in exploratory testing.
Happy path for the API has to be assumed.

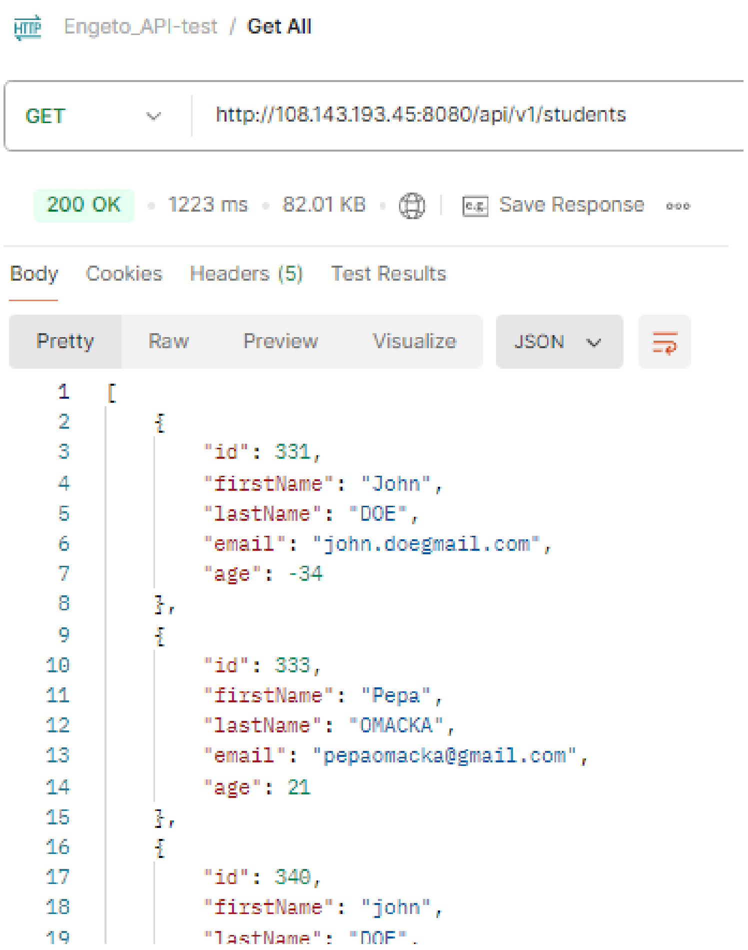
TEST SCENARIOS

The following scenarios were executed to verify application functionality, performance and security. Every test step was verified against the database.

GET

Happy path

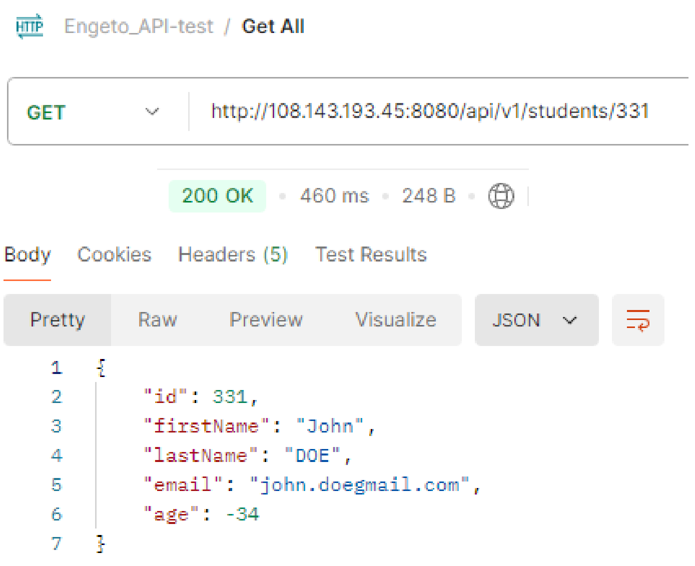
<u>Scenario:</u> Return all users by calling GET without user ID
GIVEN there is 889 entries in the datablase
WHEN GET request is sent to the endpoint URL
THEN 889 entries are returned
    AND response status is 200 OK
    AND students are returned from the smallest ID to the largest



http://108.143.193.45:8080/api/v1/students

```
[
    {
        "id": 331,
        "firstName": "John",
        "lastName": "DOE",
        "email": "john.doegmail.com",
        "age": -34
    },
    {
        "id": 333,
        "firstName": "Pepa",
        "lastName": "OMACKA",
        "email": "pepaomacka@gmail.com",
        "age": 21
    },
    {
        "id": 340,
        "firstName": "john",
        "lastName": "DOE",
```

Scenario: Return existing user by calling GET with user ID
GIVEN there is user with ID 331
WHEN GET request is sent to the endpoint URL with ‚/331' at the end
THEN Correct information for user with ID 331 is returned
AND response status is 200 OK



GET http://108.143.193.45:8080/api/v1/students/331

200 OK • 460 ms • 248 B

```
{
    "id": 331,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -34
}
```

Scenario: No users are displayed if no users are in the DB
GIVEN there is no users in the database
WHEN GET request is sent to the endpoint URL
THEN empty payload is returned
AND response status is 200 OK

Performance (theoretical bags)

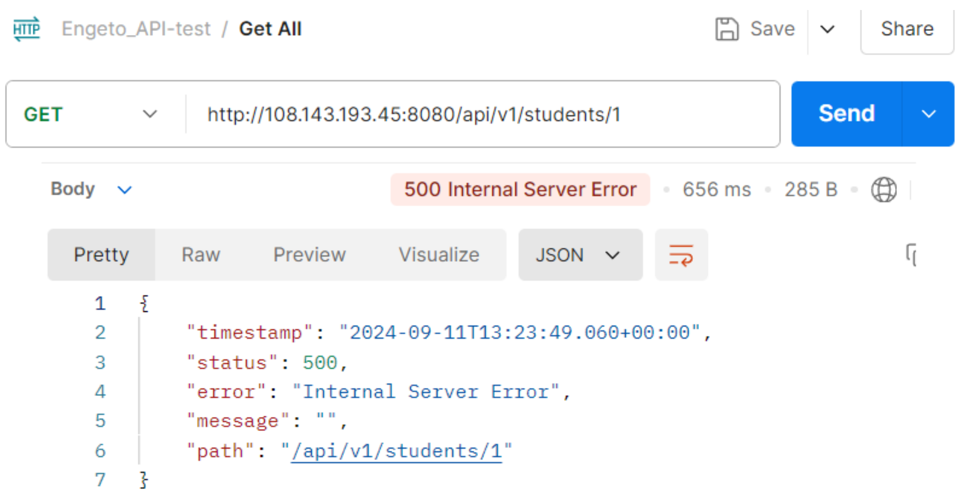
Performance tests were not executed, listed here for illustration purposes

Scenario: Return all users is not delayed if many users are present
GIVEN there is X entries in the datablase (X to be determined during team discussions)
WHEN GET request is sent to the endpoint URL
THEN X entries are returned
AND response status is 200 OK
AND response time is below N ms (N to be determined during team discussions)

Scenario: Return all users is not delayed if many requests are sent
GIVEN there is 889 entries in the datablase
WHEN X GET requests are sent to the endpoint URL within 10 minutes
THEN 889 entries are returned
AND response status is 200 OK
AND response time is below N ms (N to be determined during team discussions)

Unhappy path

<u>Scenario</u>: Non-existing student ID is requested
GIVEN there is no student with ID 1
WHEN GET request is sent to the endpoint URL with ‚/1‘ at the end
THEN error message about non-existent student is returned
AND response status is 404 Not found



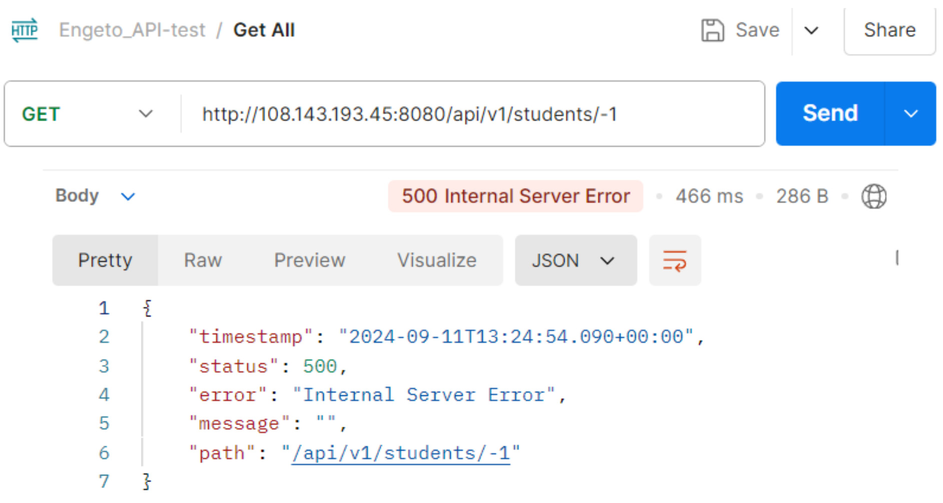
http://108.143.193.45:8080/api/v1/students/1

500 Internal Server Error • 656 ms • 285 B

```
{
    "timestamp": "2024-09-11T13:23:49.060+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/1"
}
```

Validation

<u>Scenario</u>: Invalid student ID is requested
GIVEN ID field is a non-negative integer
WHEN GET request is sent to the endpoint URL with ‚/-10‘ at the end
OR GET request is sent to the endpoint URL with ‚/-1‘ at the end
OR GET request is sent to the endpoint URL with ‚/a‘ at the end
OR GET request is sent to the endpoint URL with ‚?id=331‘ at the end
THEN error message about invalid ID is returned
AND response status is 404 Not found or 400 Bad Request



http://108.143.193.45:8080/api/v1/students/-1

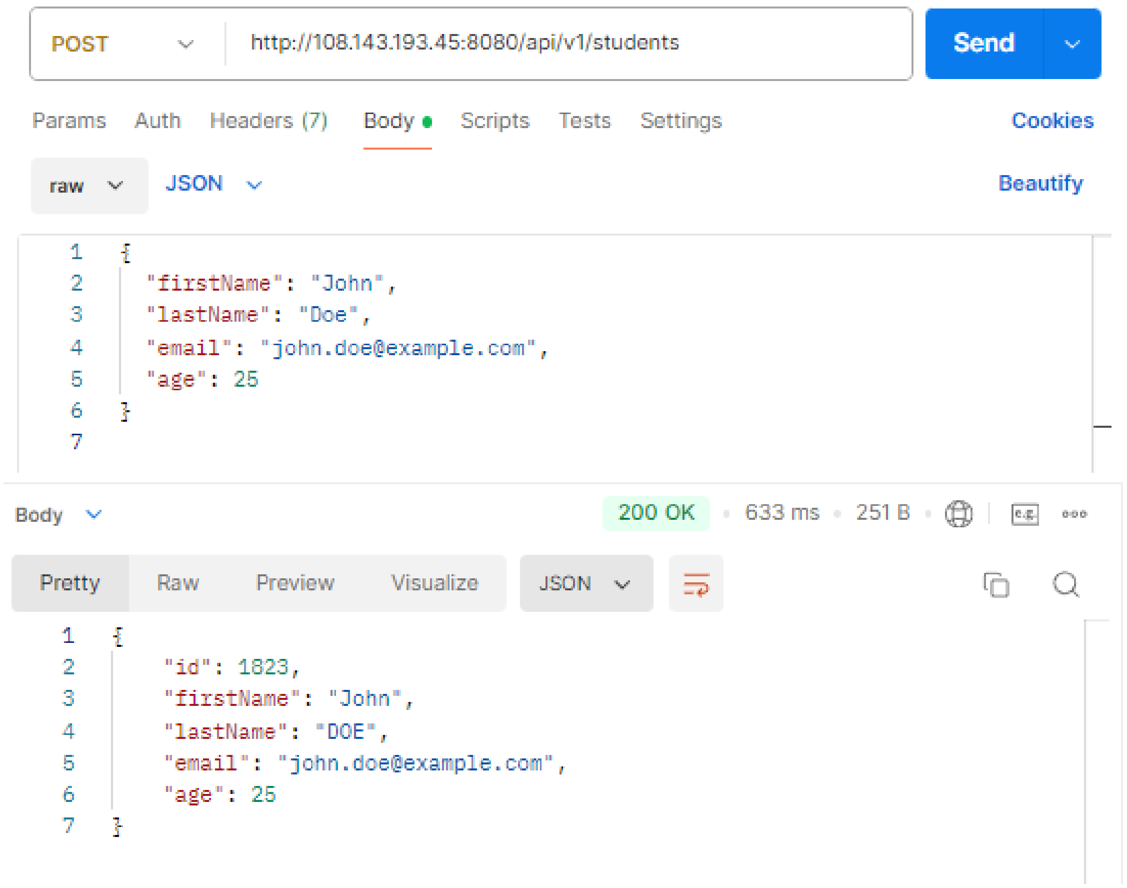
500 Internal Server Error • 466 ms • 286 B

```
{
    "timestamp": "2024-09-11T13:24:54.090+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/-1"
}
```

POST

Happy path

<u>Scenario:</u> New student is created with all fields
GIVEN the last student in the database has ID 1822
WHEN POST request is sent
AND firstName and lastName set to a random name
AND email is set to random email in format „{{alphanumerical_characters}}@{{domain}}.{{top_level_domain}}“
AND age is set to random integer
AND id is missing
THEN response contains firstName, lastName, email and age as provided
AND id is 1823
AND response status is 200 OK or 201 Created



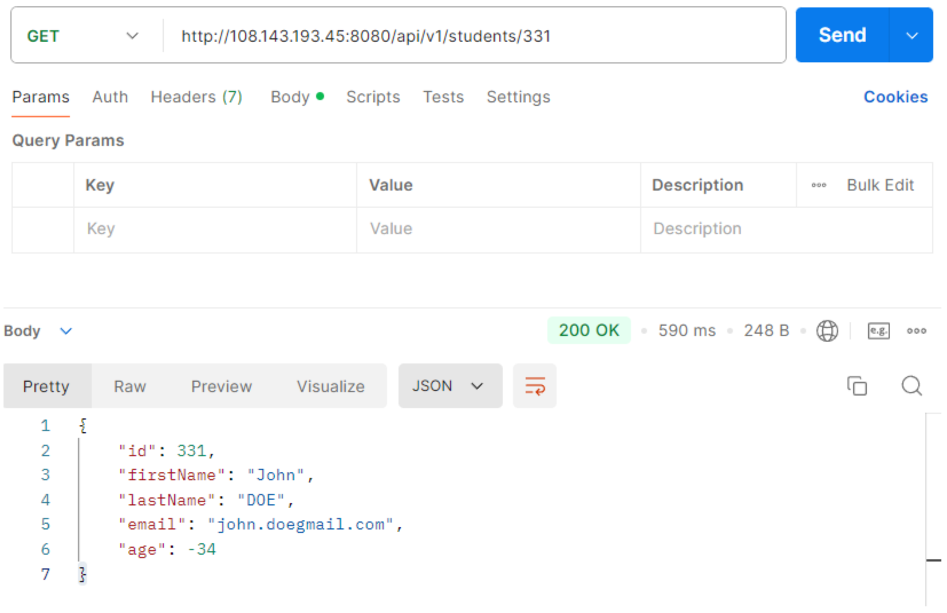
http://108.143.193.45:8080/api/v1/students

```
{
"firstName": "John",
"lastName": "Doe",
"email": "john.doe@example.com",
"age": 25
}
```

200 OK • 633 ms • 251 B

```
{
    "id": 1823,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doe@example.com",
    "age": 25
}
```

<u>Scenario</u>: Existing student is edited
GIVEN there is student with ID 331
WHEN POST request is sent with id:331 and firstName is different
    OR lastName is different
    OR email is different
    OR age is different
THEN response contains id, firstName, lastName, email and age as provided
    AND response status is 200 OK

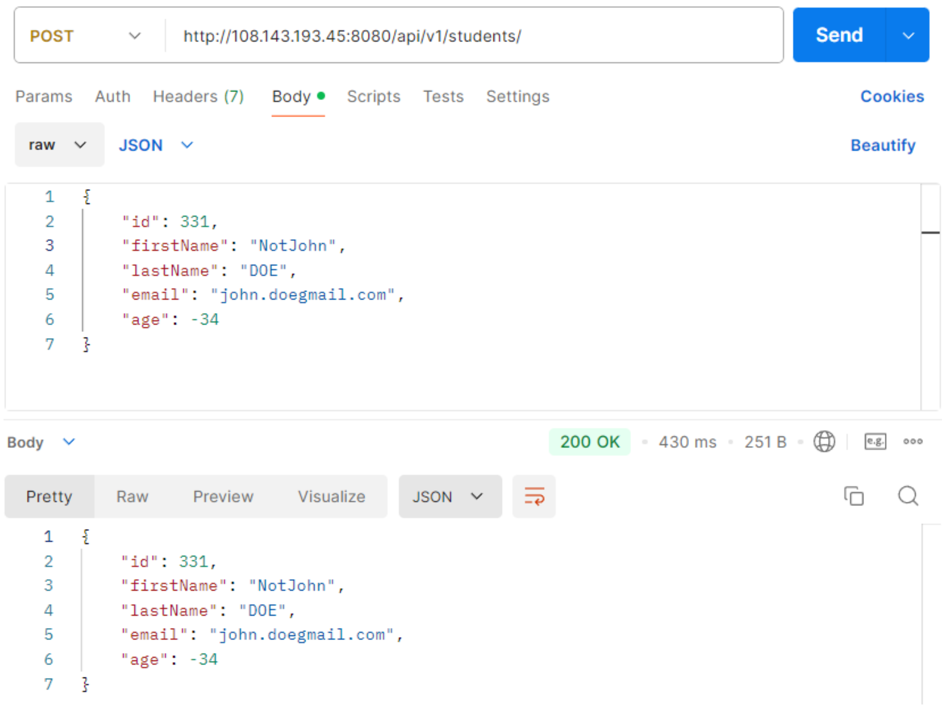


GET http://108.143.193.45:8080/api/v1/students/331

Body 200 OK 590 ms 248 B

JSON

```
{
    "id": 331,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -34
}
```



POST http://108.143.193.45:8080/api/v1/students/

raw JSON

```
{
    "id": 331,
    "firstName": "NotJohn",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -34
}
```

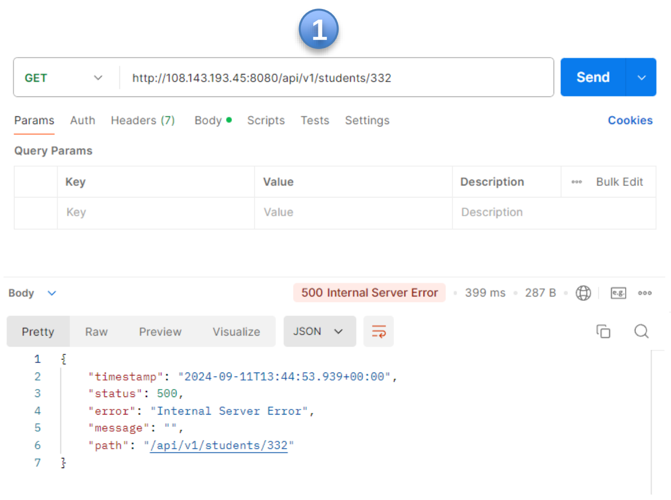
Body 200 OK 430 ms 251 B

JSON

```
{
    "id": 331,
    "firstName": "NotJohn",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -34
}
```

Unhappy path

<u>Scenario:</u> Student is edited with non-existent ID – student created instead
GIVEN the last student in the database has ID 1810
WHEN POST request is sent
AND firstName and lastName set to a random name
AND email is set to random email in format „{{alphanumerical_characters}}@{{domain}}.{{top_level_domain}}“
AND age is set to random integer
AND id is 9999
THEN expected result needs to be discussed with the team
WHEN GET request is sent with ‚/9999‘
THEN response status is 404 Not found


1

GET http://108.143.193.45:8080/api/v1/students/332 Send

Params Auth Headers (7) Body Scripts Tests Settings Cookies

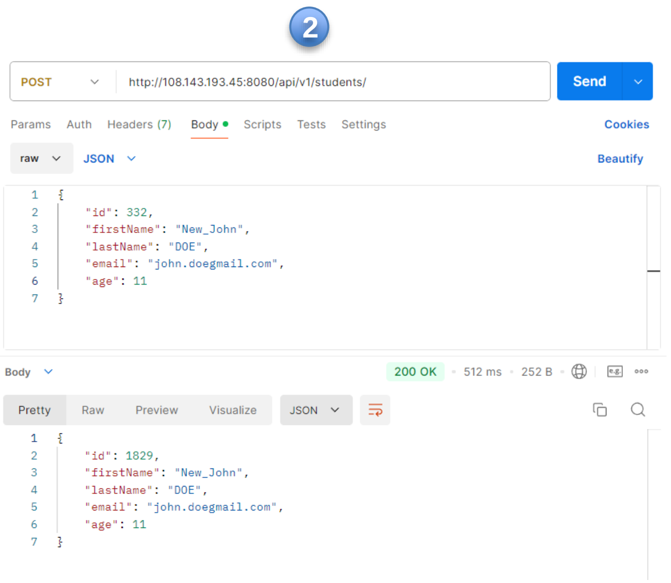
Query Params

Key	Value	Description
Key	Value	Description

Body 500 Internal Server Error · 399 ms · 287 B

Pretty Raw Preview Visualize JSON

```
{
    "timestamp": "2024-09-11T13:44:53.939+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/332"
}
```


2

POST http://108.143.193.45:8080/api/v1/students/ Send

Params Auth Headers (7) Body Scripts Tests Settings Cookies

raw JSON Beautify

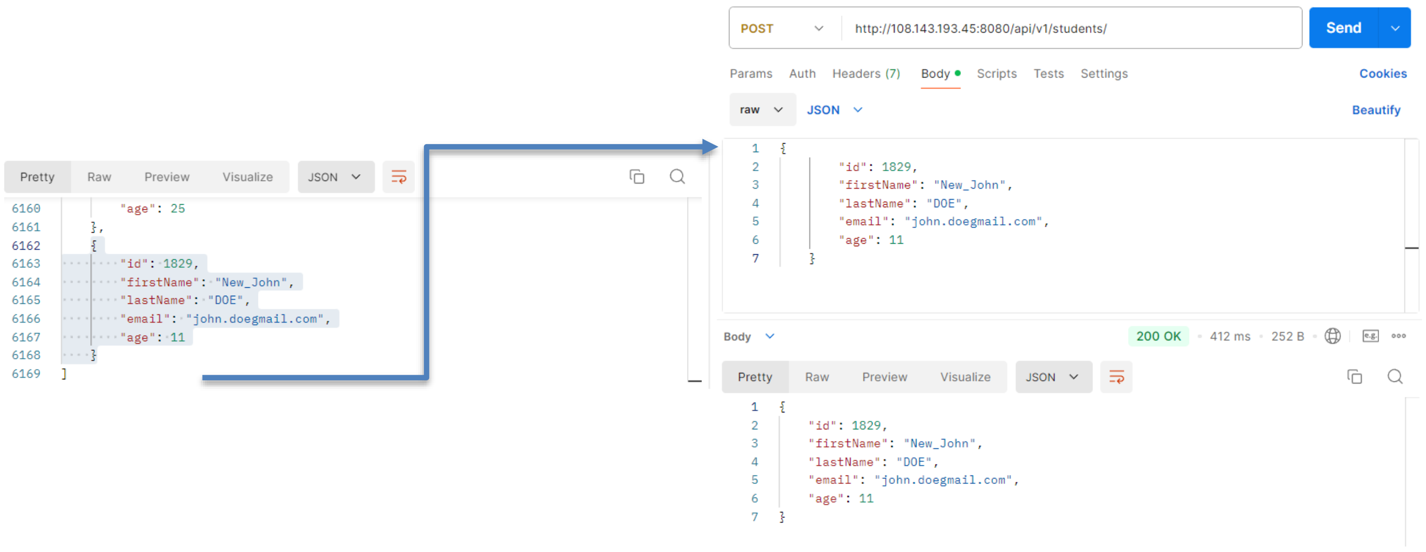
```
{
    "id": 332,
    "firstName": "New_John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

Body 200 OK · 512 ms · 252 B

Pretty Raw Preview Visualize JSON

```
{
    "id": 1829,
    "firstName": "New_John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

<u>Scenario</u>: Student is edited with no changes
GIVEN the last student in the database has ID 1829
WHEN POST request is sent
AND all fields contain current values for student 1829
THEN expected result needs to be discussed with the team


Pretty Raw Preview Visualize JSON

```
        "age": 25
    },
    {
        "id": 1829,
        "firstName": "New_John",
        "lastName": "DOE",
        "email": "john.doegmail.com",
        "age": 11
    }
]
```

POST http://108.143.193.45:8080/api/v1/students/ Send

Params Auth Headers (7) Body Scripts Tests Settings Cookies

raw JSON Beautify

```
{
        "id": 1829,
        "firstName": "New_John",
        "lastName": "DOE",
        "email": "john.doegmail.com",
        "age": 11
    }
```

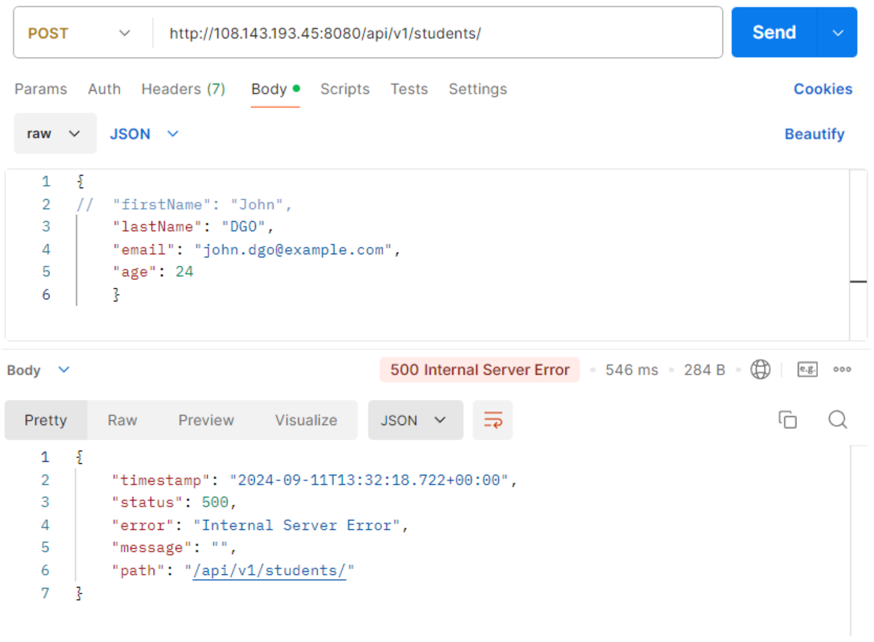
Body 200 OK · 412 ms · 252 B

Pretty Raw Preview Visualize JSON

```
{
    "id": 1829,
    "firstName": "New_John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

Validation

Scenario: New student is created with some fields missing
GIVEN all fields are mandatory
WHEN POST request is sent with firstName missing
    OR lastName is missing
    OR email is missing
    OR age is missing
THEN expected result needs to be discussed with the team
Error message is displayed and responsed status is 400 Bad Request



POST http://108.143.193.45:8080/api/v1/students/

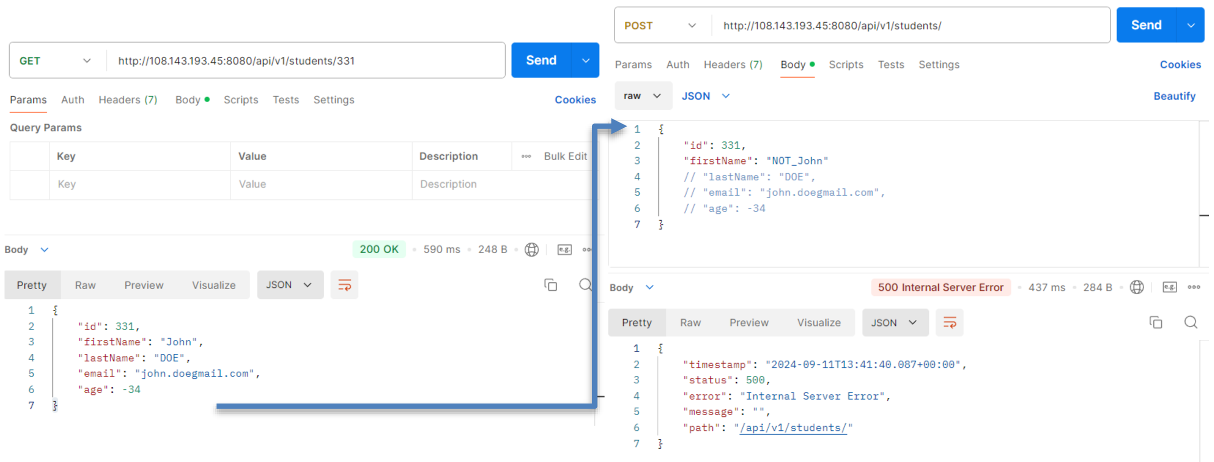
Body: raw, JSON

```
{
//  "firstName": "John",
    "lastName": "DGO",
    "email": "john.dgo@example.com",
    "age": 24
    }
```

Body — 500 Internal Server Error · 546 ms · 284 B

```
{
    "timestamp": "2024-09-11T13:32:18.722+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/"
}
```

Scenario: One field for existing student is edited
GIVEN there is student with ID 331
WHEN POST request is sent with id:331 and no other fields except for firstName present, firstName and is different
    OR only lastName is present and is different
    OR only email is present and is different
    OR only age is present and is different
THEN error message is displayed and responsed status is 400 Bad Request



GET http://108.143.193.45:8080/api/v1/students/331

Query Params

	Key	Value	Description
	Key	Value	Description

Body — 200 OK · 590 ms · 248 B

```
{
    "id": 331,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -34
}
```

POST http://108.143.193.45:8080/api/v1/students/

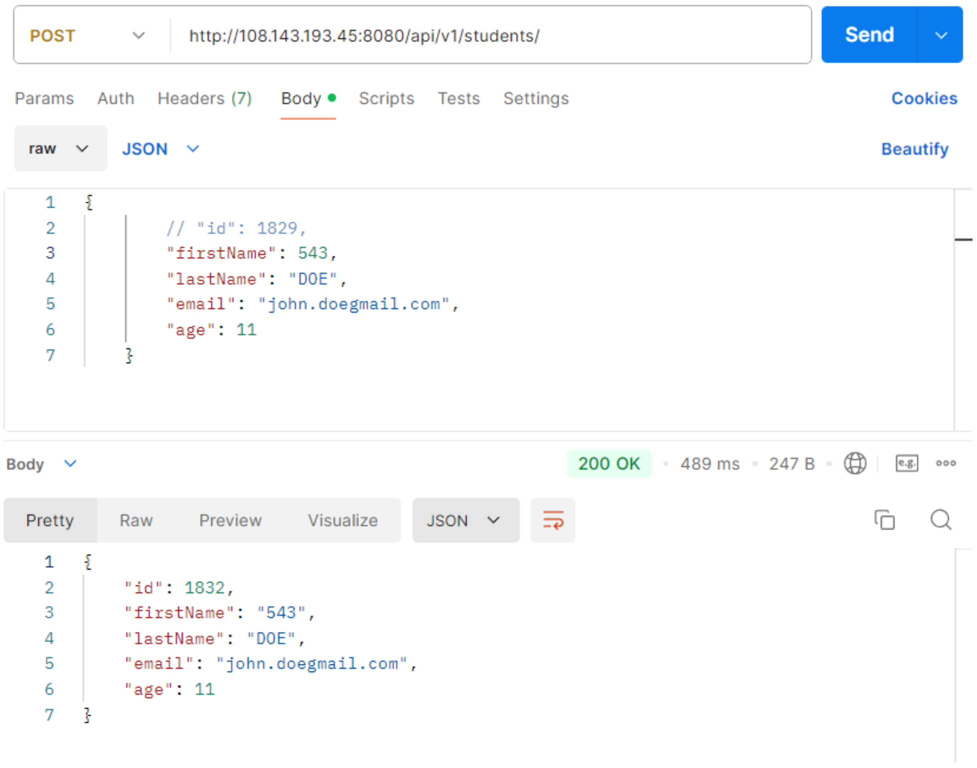
Body: raw, JSON

```
{
    "id": 331,
    "firstName": "NOT_John"
    // "lastName": "DOE",
    // "email": "john.doegmail.com",
    // "age": -34
}
```

Body — 500 Internal Server Error · 437 ms · 284 B

```
{
    "timestamp": "2024-09-11T13:41:40.087+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/"
}
```

<u>Scenario</u>: First/Last name fields are validated
Exact expected behaviour to be discussed with the team
Things to investigate: minimal length, maximal length, capitalization, supported characters, number of whitespaces, trailing/leading whitespaces, number, boolean, empty string



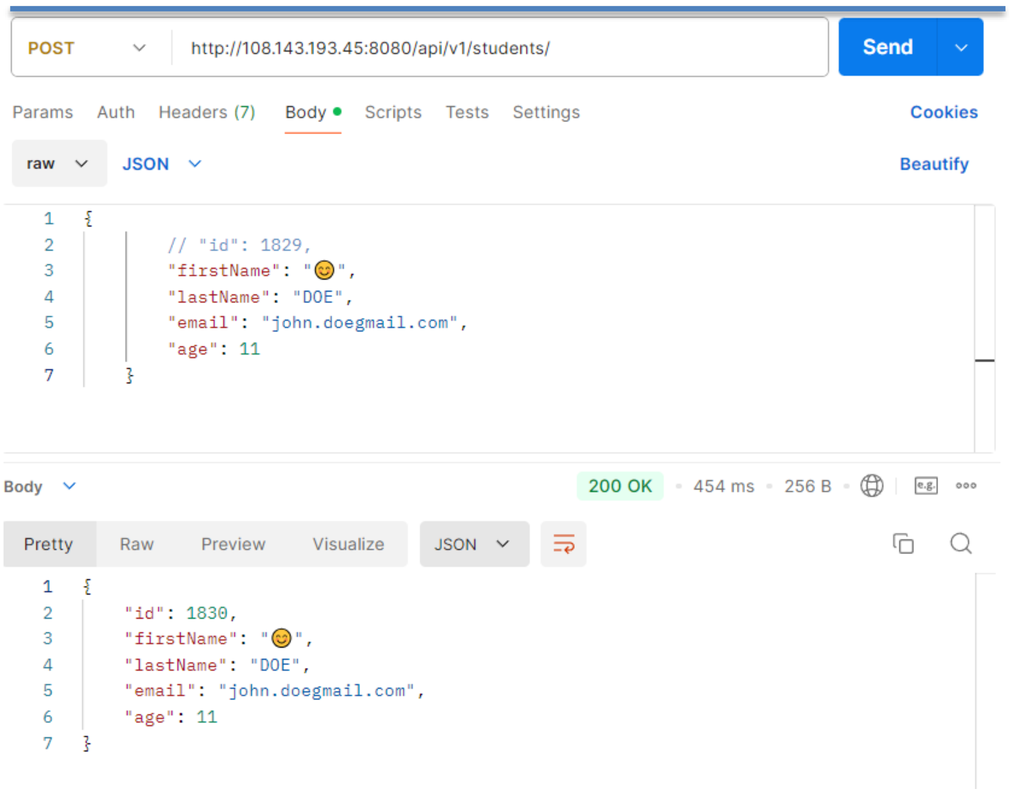
POST http://108.143.193.45:8080/api/v1/students/

Body: raw JSON

```
{
    // "id": 1829,
    "firstName": 543,
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
  }
```

200 OK · 489 ms · 247 B

```
{
    "id": 1832,
    "firstName": "543",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```



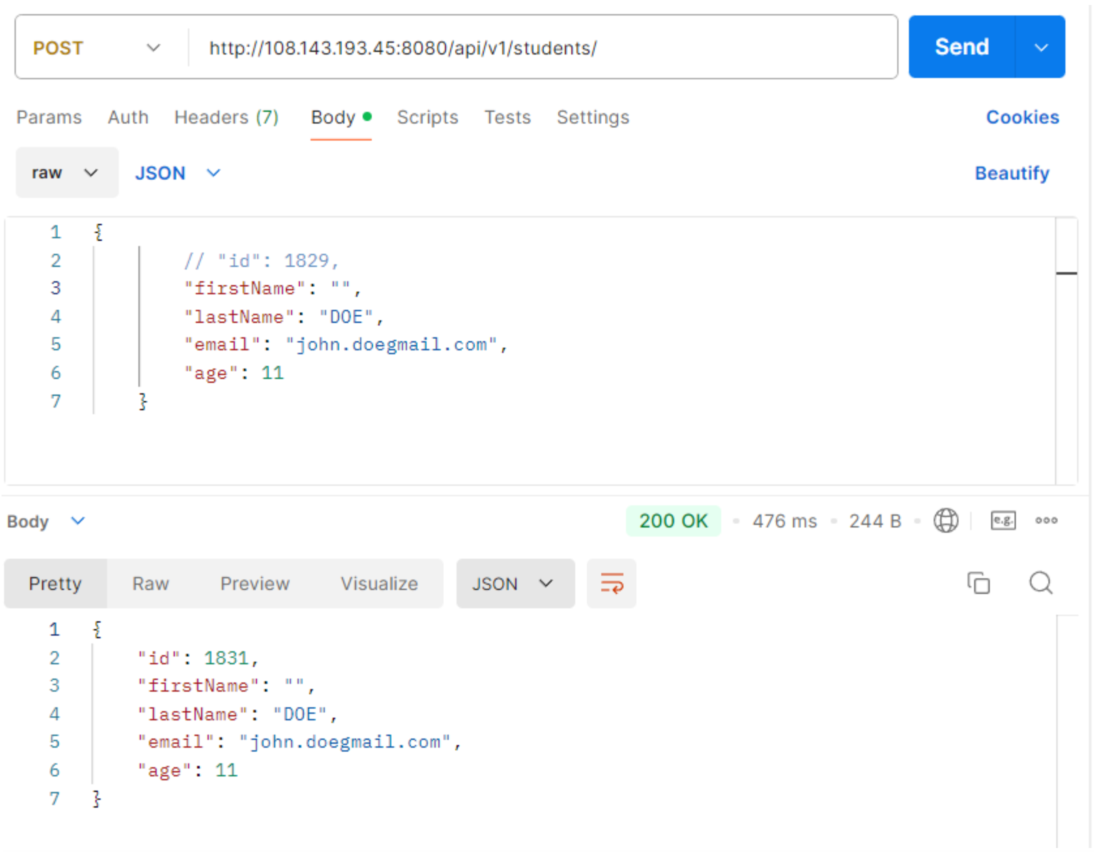
POST http://108.143.193.45:8080/api/v1/students/

Body: raw JSON

```
{
    // "id": 1829,
    "firstName": "😊",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
  }
```

200 OK · 454 ms · 256 B

```
{
    "id": 1830,
    "firstName": "😊",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

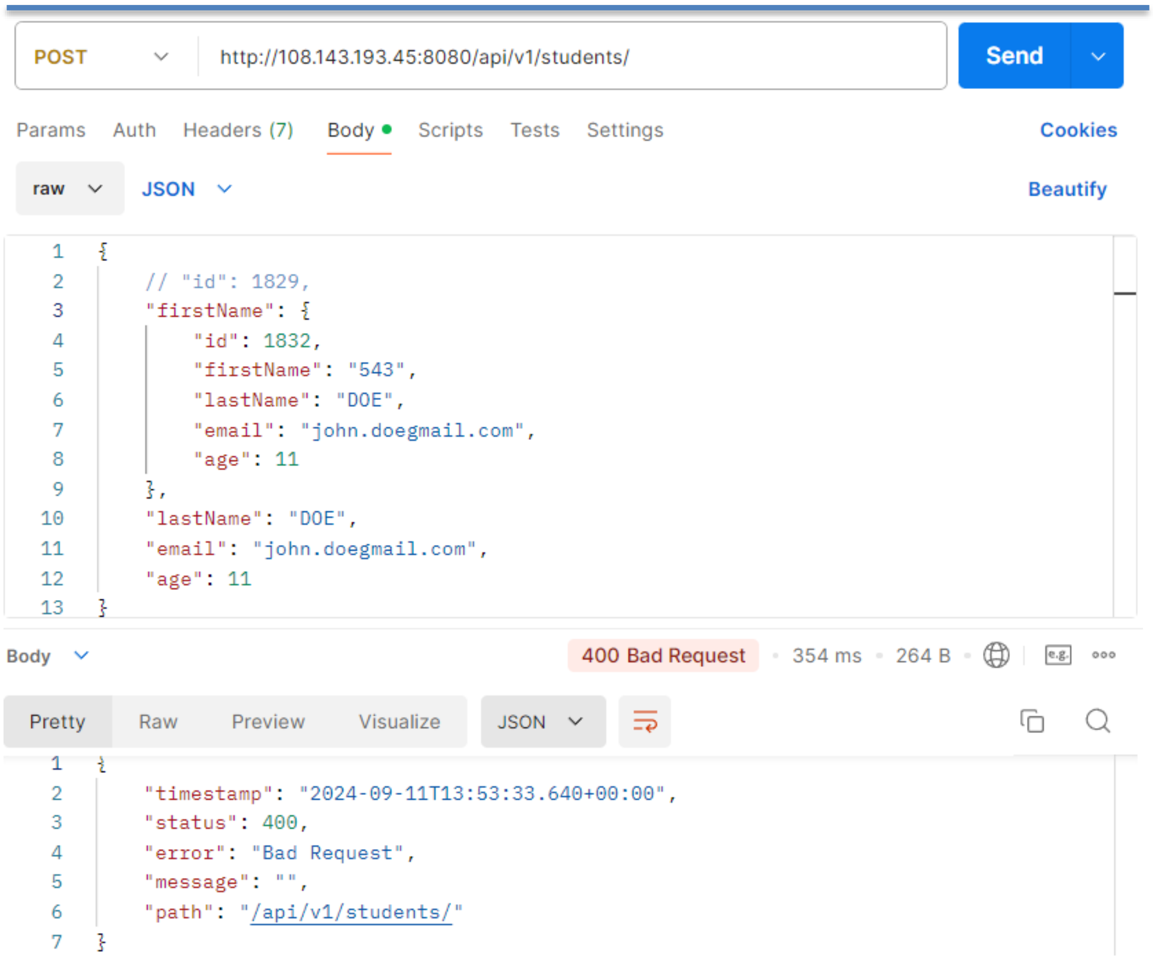


POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

Body 200 OK 476 ms 244 B

```
{
    "id": 1831,
    "firstName": "",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```



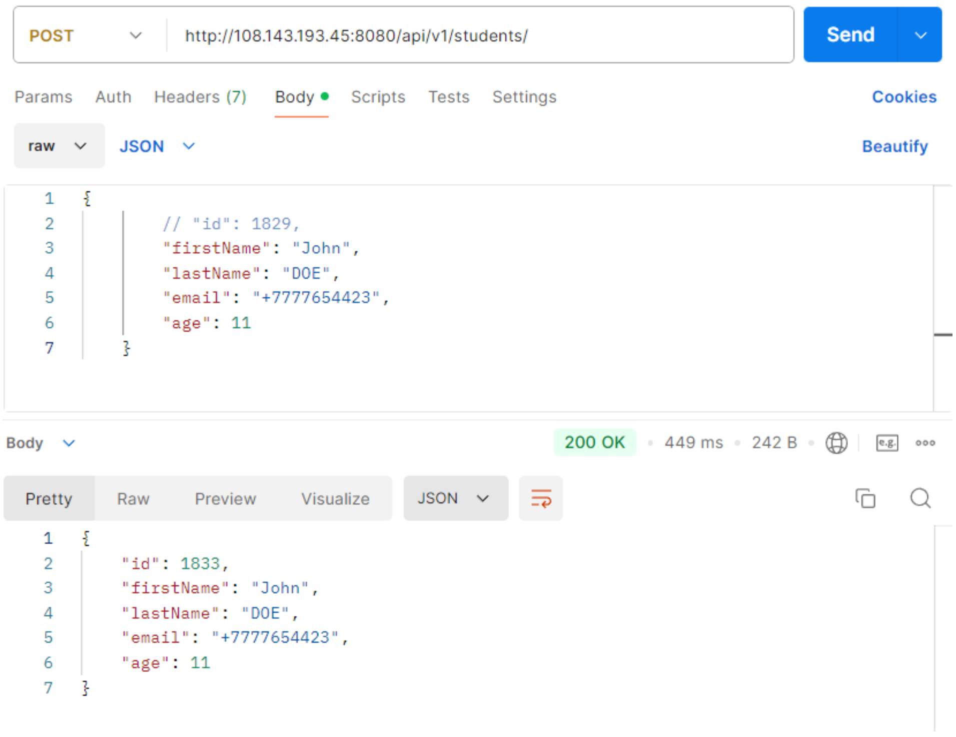
POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": {
        "id": 1832,
        "firstName": "543",
        "lastName": "DOE",
        "email": "john.doegmail.com",
        "age": 11
    },
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 11
}
```

Body 400 Bad Request 354 ms 264 B

```
{
    "timestamp": "2024-09-11T13:53:33.640+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "",
    "path": "/api/v1/students/"
}
```

Scenario: Email field is validated
Exact expected behaviour to be discussed with the team
Things to investigate: supported TLDs, non-email strings, trailing/leading whitespaces, number, boolean,



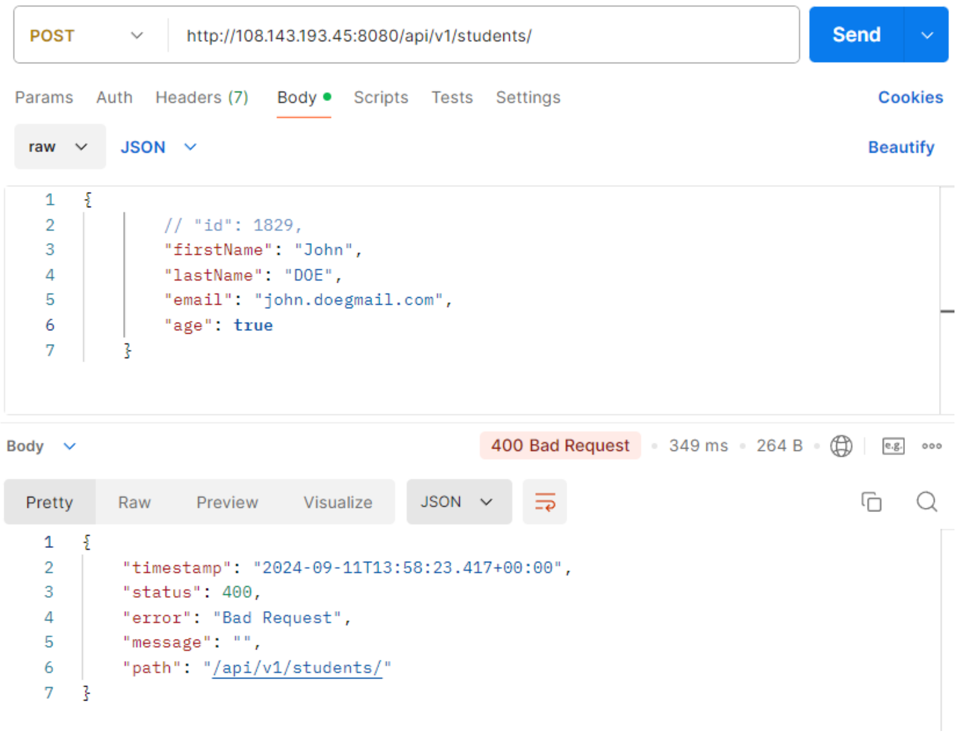
POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "+7777654423",
    "age": 11
}
```

200 OK • 449 ms • 242 B

```
{
    "id": 1833,
    "firstName": "John",
    "lastName": "DOE",
    "email": "+7777654423",
    "age": 11
}
```

Scenario: Age field is validated
Exact expected behaviour to be discussed with the team
Things to investigate: minimal age, maximal age, 0, negative numbers, decimal, operation (e.g. 2+2), string, boolean, null, string containing number (can be cast into number)

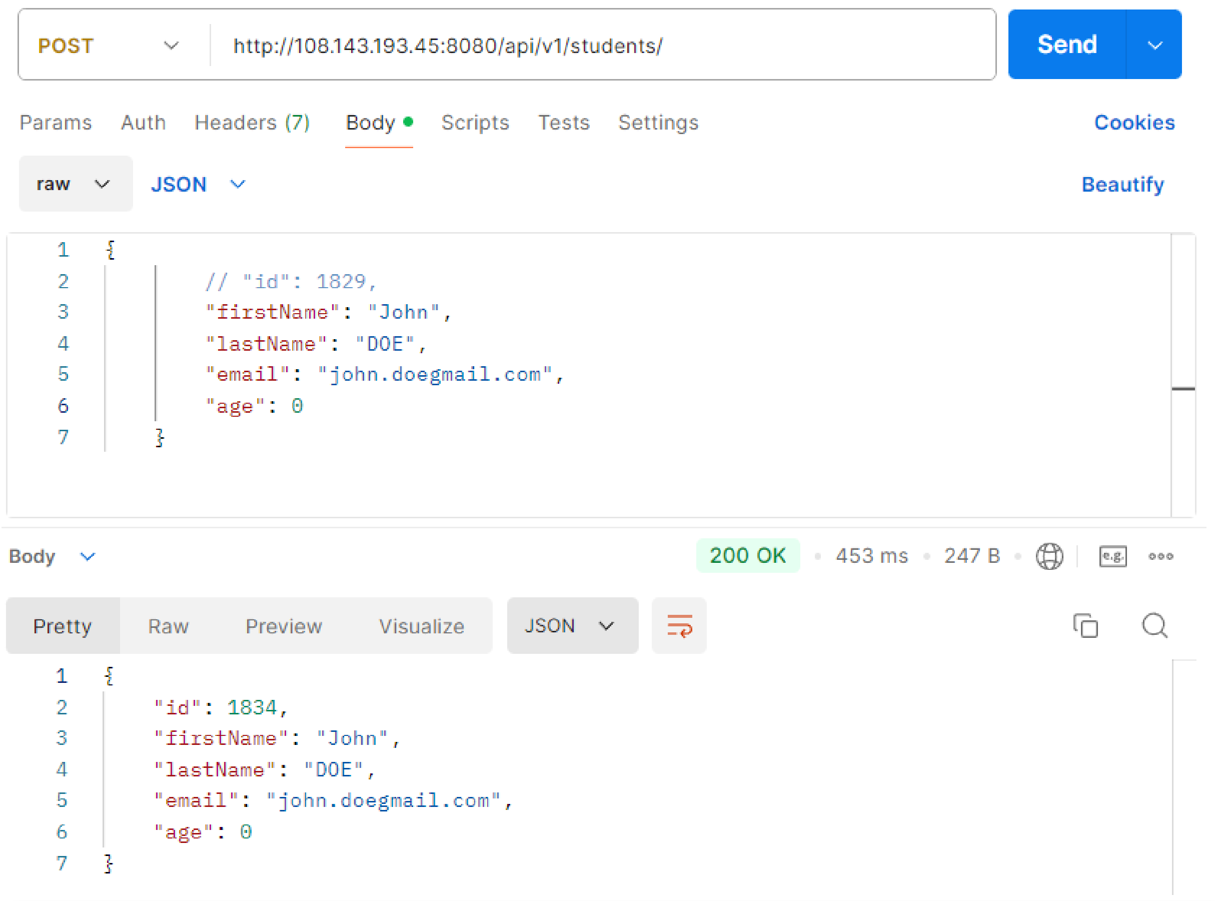


POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": true
}
```

400 Bad Request • 349 ms • 264 B

```
{
    "timestamp": "2024-09-11T13:58:23.417+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "",
    "path": "/api/v1/students/"
}
```

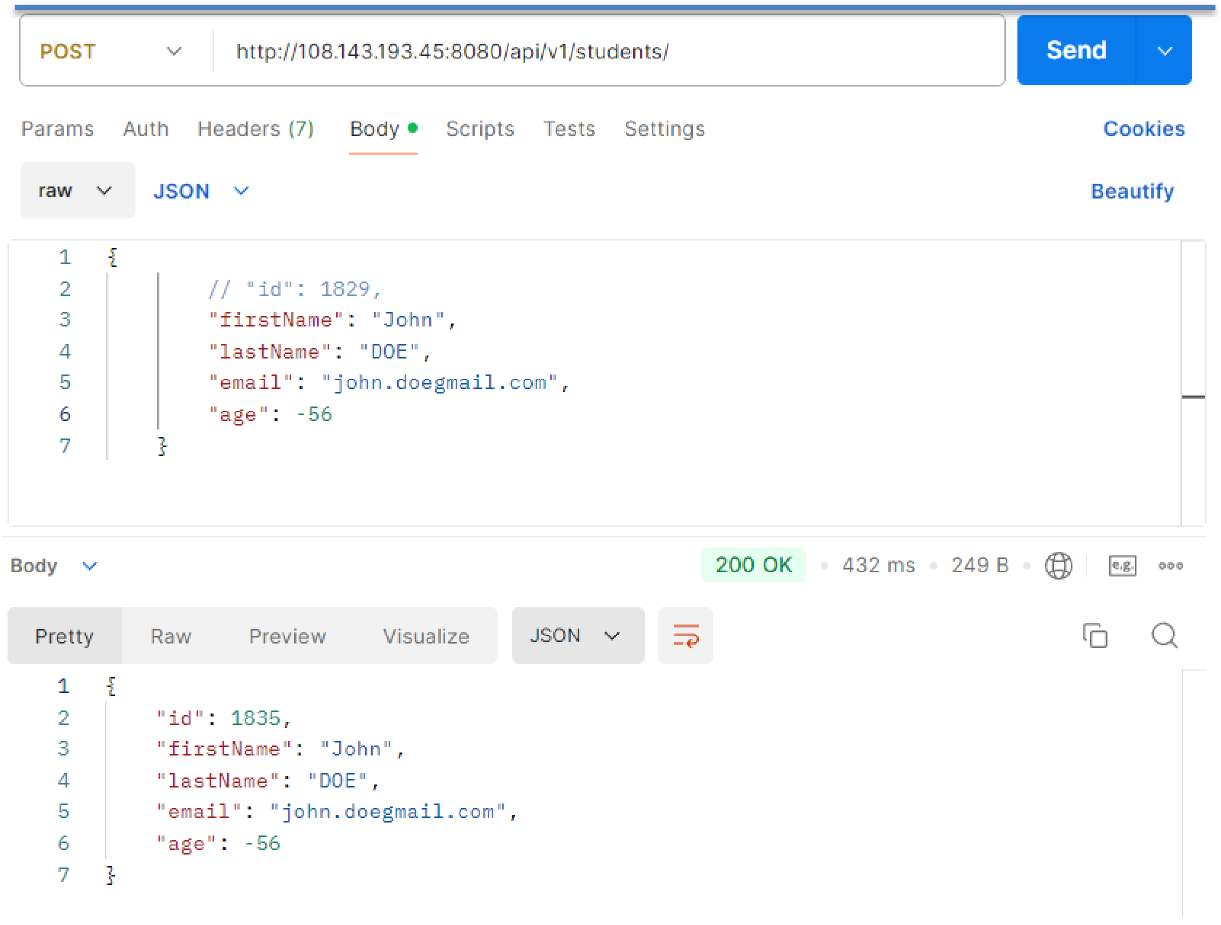


POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 0
}
```

200 OK • 453 ms • 247 B

```
{
    "id": 1834,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 0
}
```

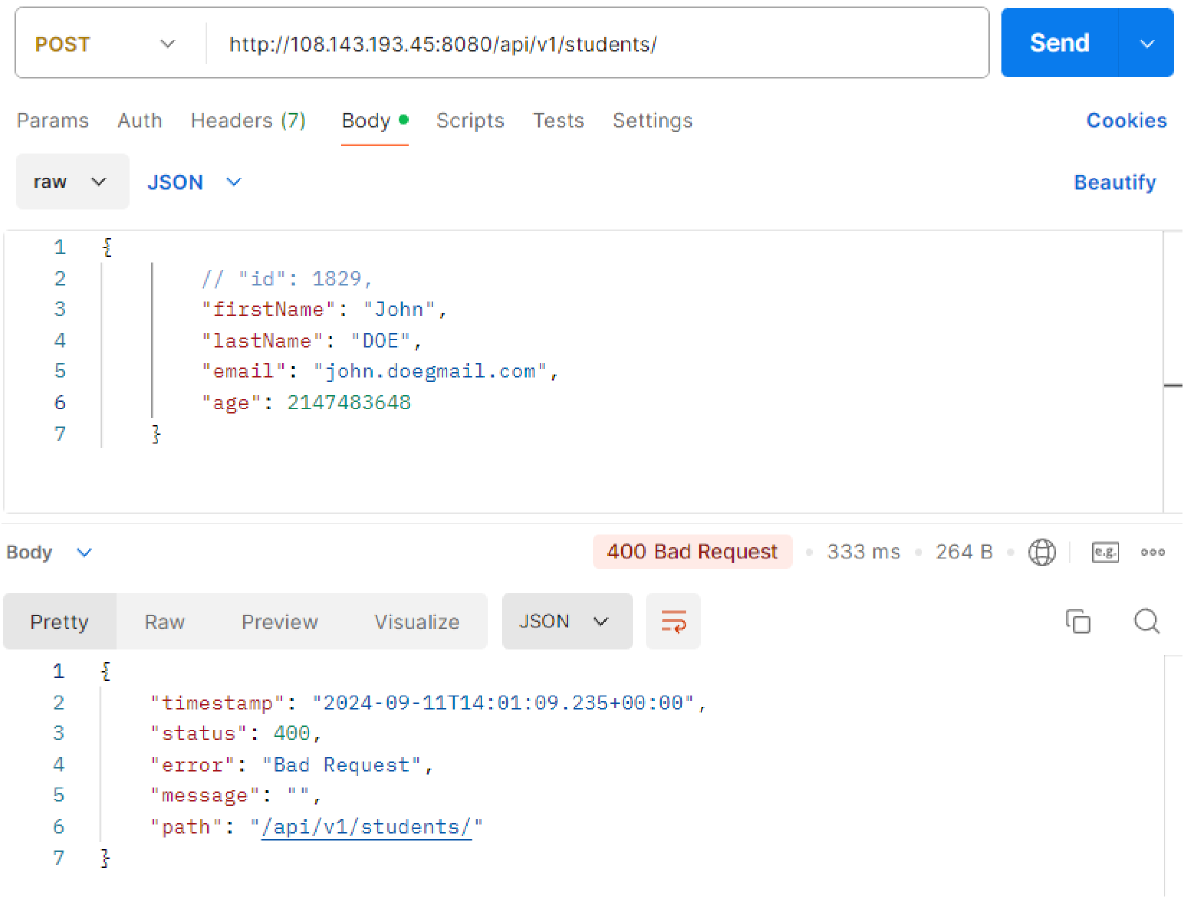


POST http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -56
}
```

200 OK • 432 ms • 249 B

```
{
    "id": 1835,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": -56
}
```

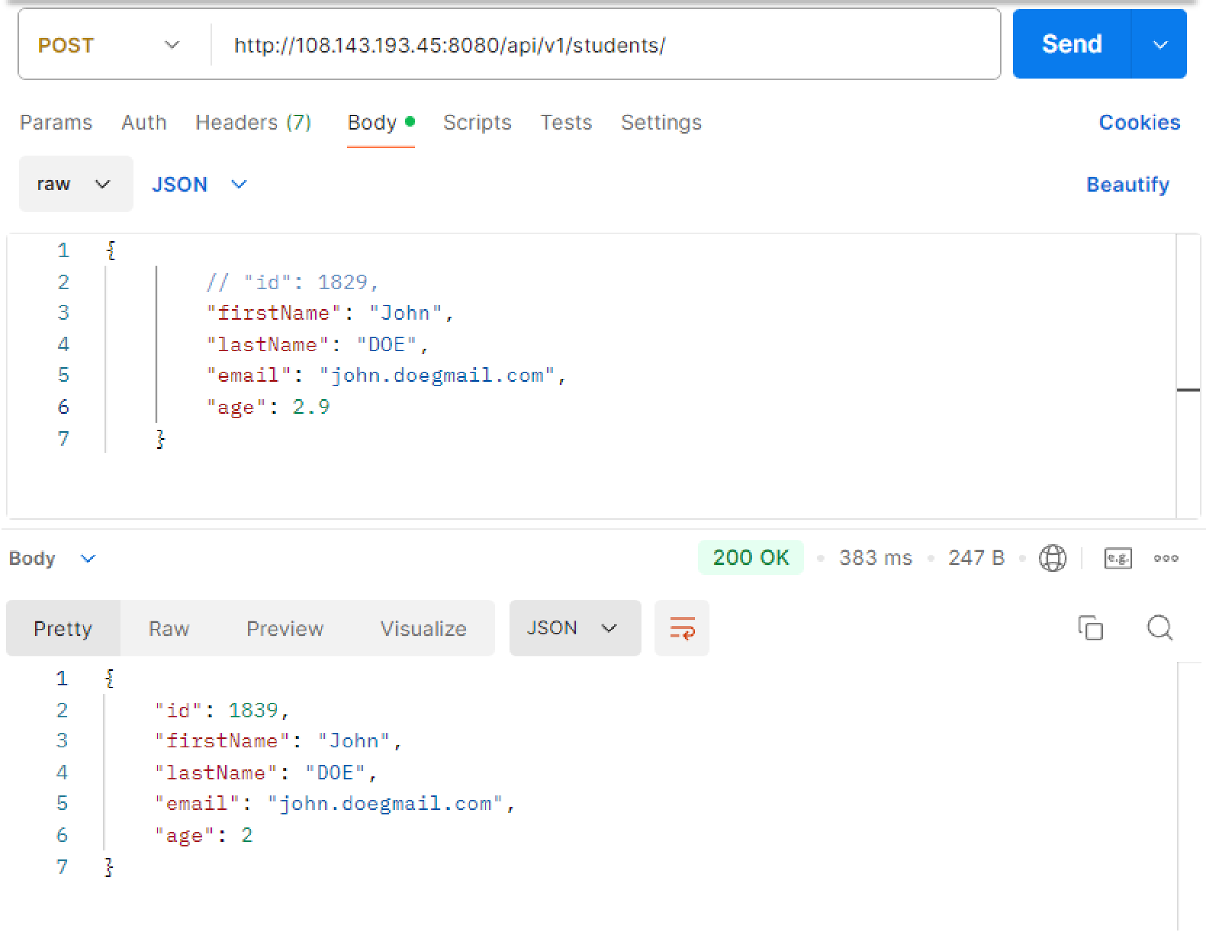


http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 2147483648
}
```

400 Bad Request • 333 ms • 264 B

```
{
    "timestamp": "2024-09-11T14:01:09.235+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "",
    "path": "/api/v1/students/"
}
```



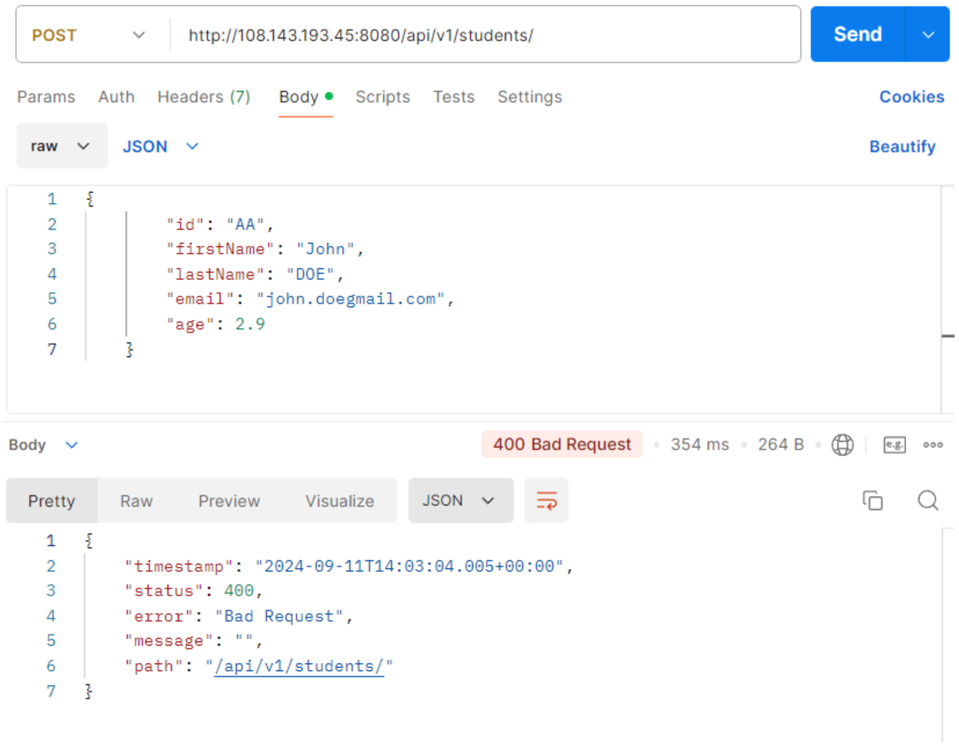
http://108.143.193.45:8080/api/v1/students/

```
{
    // "id": 1829,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 2.9
}
```

200 OK • 383 ms • 247 B

```
{
    "id": 1839,
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 2
}
```

Scenario: ID field is validated
Exact expected behaviour to be discussed with the team
Things to investigate: minimal age, maximal age, 0, negative numbers, decimal, operation (e.g. 2+2), string, boolean, null, string containing number (can be cast into number)



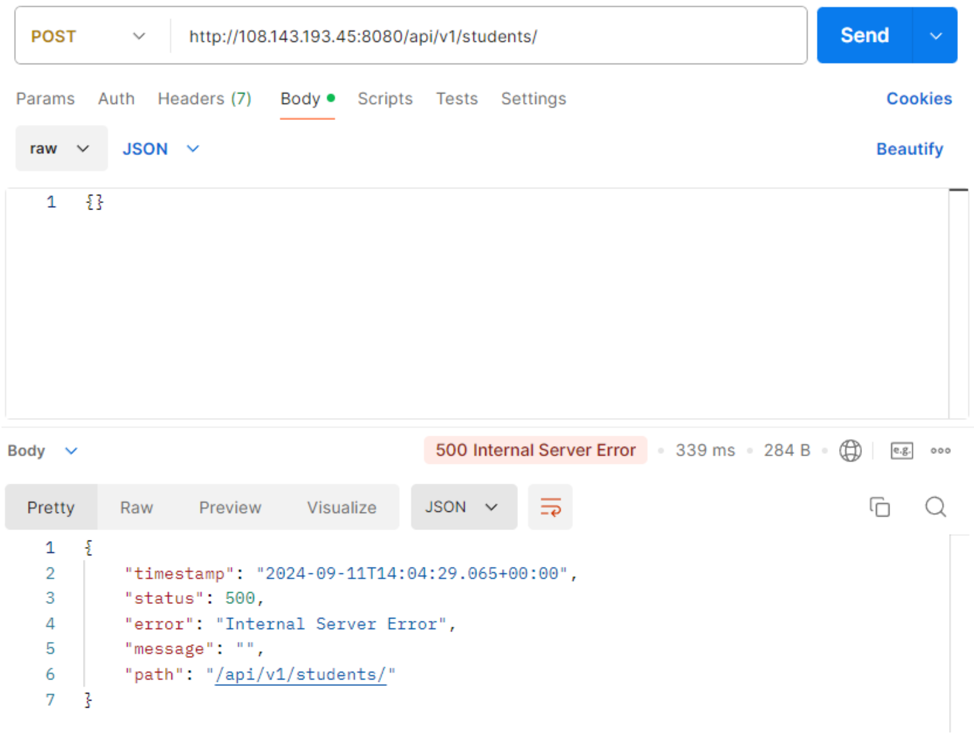
http://108.143.193.45:8080/api/v1/students/

```
{
    "id": "AA",
    "firstName": "John",
    "lastName": "DOE",
    "email": "john.doegmail.com",
    "age": 2.9
}
```

400 Bad Request • 354 ms • 264 B

```
{
    "timestamp": "2024-09-11T14:03:04.005+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "",
    "path": "/api/v1/students/"
}
```

Scenario: Invalid JSON is sent
Exact expected behaviour to be discussed with the team
Things to investigate: empty JSON, missing { or }, missing comma, missing key, missing value



http://108.143.193.45:8080/api/v1/students/

```
{}
```

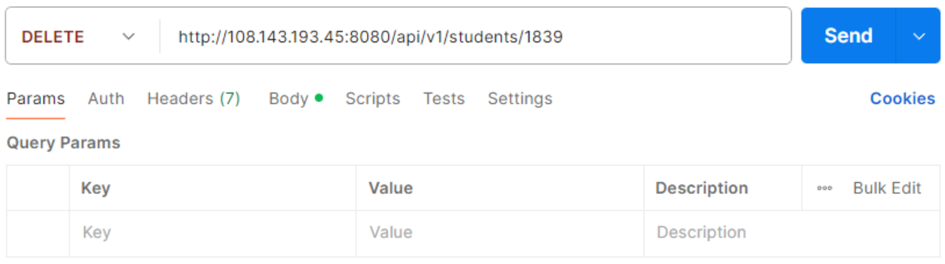
500 Internal Server Error • 339 ms • 284 B

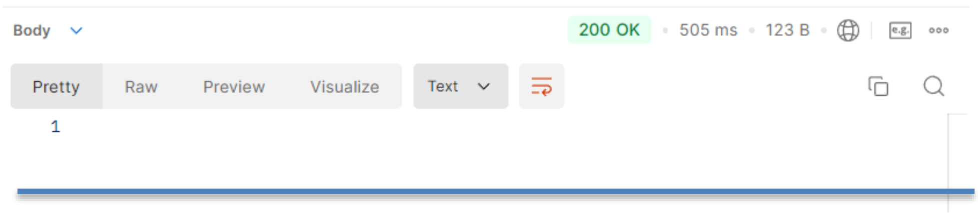
```
{
    "timestamp": "2024-09-11T14:04:29.065+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/"
}
```

DELETE

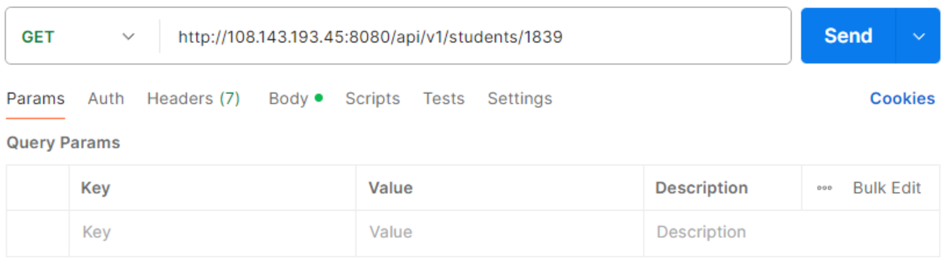
Happy path

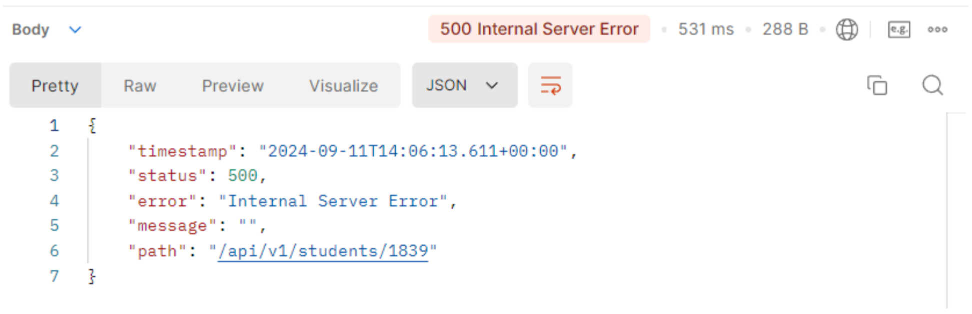
Scenario: Student is deleted
GIVEN there is student with ID 1839
WHEN DELETE request it sent to the endpoint with ‚/1819' at the end
THEN response status 200 OK is returned
WHEN GET request it sent to the endpoint with ‚/1819' at the end
THEN response status 404 Not Found is returned


DELETE http://108.143.193.45:8080/api/v1/students/1839


Body 200 OK • 505 ms • 123 B

```
1
```

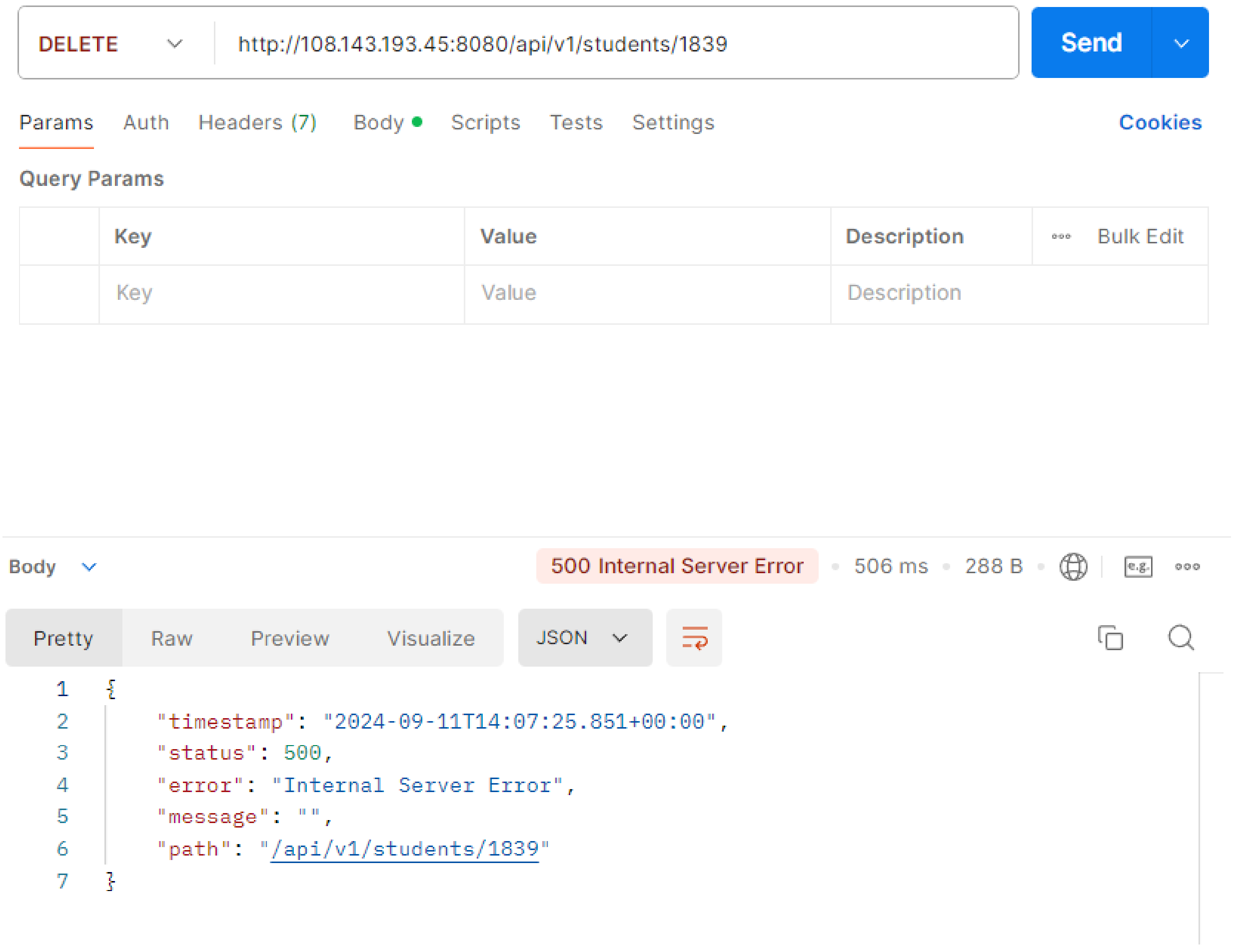

GET http://108.143.193.45:8080/api/v1/students/1839


Body 500 Internal Server Error • 531 ms • 288 B

```
{
    "timestamp": "2024-09-11T14:06:13.611+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/1839"
}
```

Unhappy path

Scenario: Non-existent user is deleted
GIVEN there is no student with ID 1839
WHEN DELETE request it sent to the endpoint with ‚/1819' at the end
THEN response status 400 Bad Request or 404 Not Found is returned



http://108.143.193.45:8080/api/v1/students/1839

Key	Value	Description
Key	Value	Description

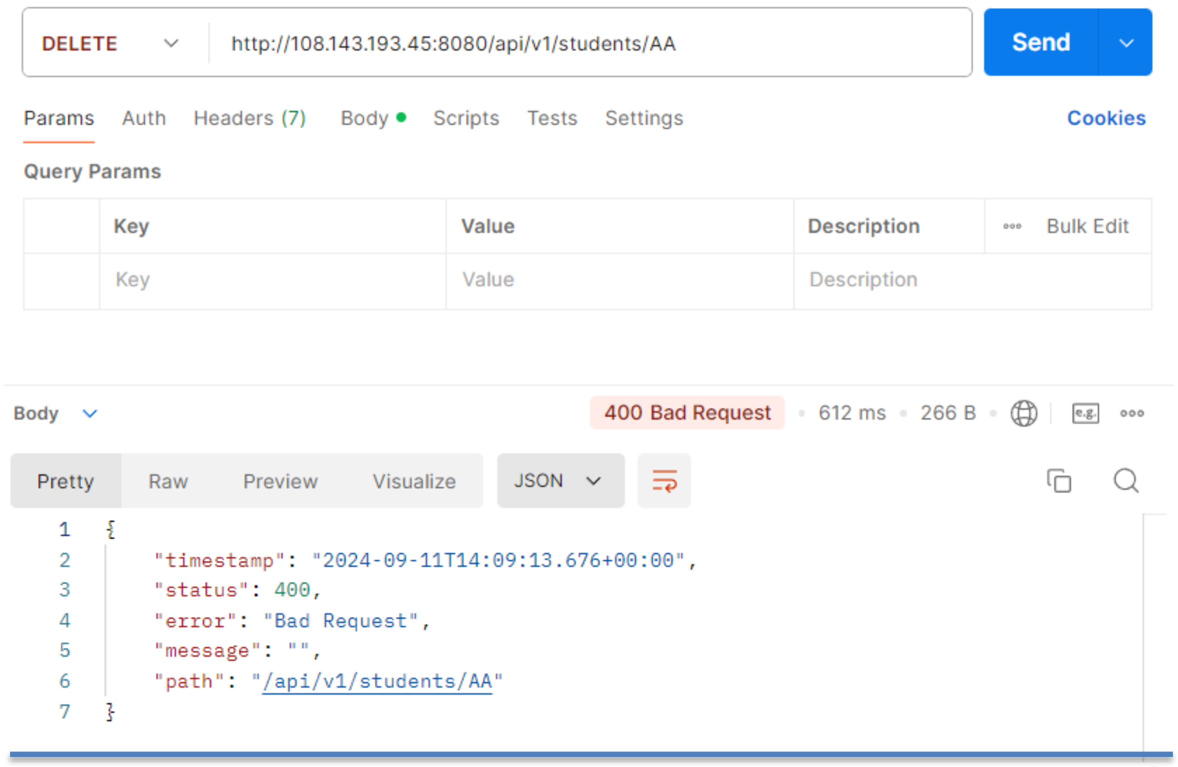
500 Internal Server Error • 506 ms • 288 B

```
{
    "timestamp": "2024-09-11T14:07:25.851+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "",
    "path": "/api/v1/students/1839"
}
```

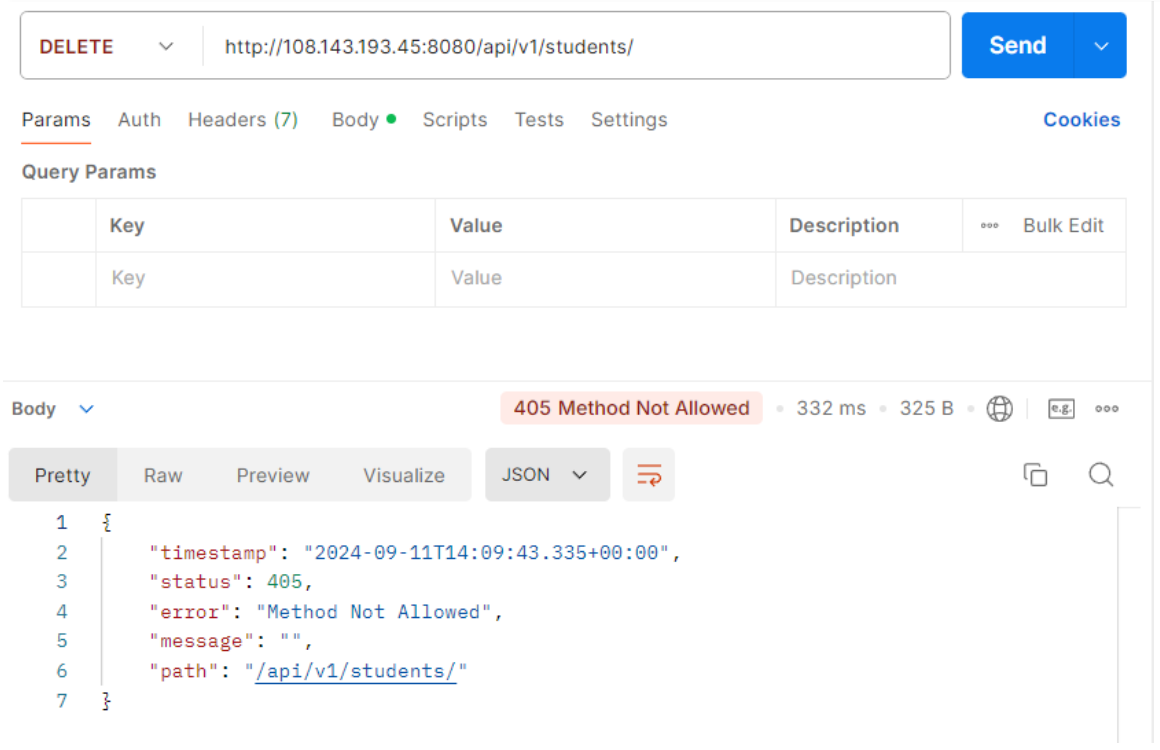
Validations

Scenario: Deleted ID validation

Exact expected behaviour to be discussed with the team. Things to investigate: missing ID, not a number, 0, negative number



```
{
    "timestamp": "2024-09-11T14:09:13.676+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "",
    "path": "/api/v1/students/AA"
}
```



```
{
    "timestamp": "2024-09-11T14:09:43.335+00:00",
    "status": 405,
    "error": "Method Not Allowed",
    "message": "",
    "path": "/api/v1/students/"
}
```

Easy to forget things

- Verify that new data is correctly stored and retrieved.
- Ensure that error messages are user-friendly and informative.
- Confirm that authorization and authentication mechanisms are in place and effective.
- Test API responses for various edge cases and unexpected inputs.
- Check if the API handles large data sets and high load scenarios effectively.

Strategies and methods

Test strategy or method	Planned to be used?	Already used?
Requirements, use cases, design notes covered appropriately in this test plan	Y	N
Code coverage is sufficient	N	N
The white-box knowledge is reflected in the test plan	N	N
Previous test plans for similar features checked to inspire	Y	N
Exploratory testing before completing the final version of the test plan	Y	Y
CRUD testing appropriately applied	Y	Y
Positive and negative testing appropriately applied	Y	Y
Boundary inputs appropriately covered	Y	Y
Test plan reviewed for correctness and completeness by the relevant team members	Y	N

BUG REPORT

Authorization not required

Steps:

1. Send any correct request without authentication

Expected: status 401 or 403
Actual: status 200 OK

GET with invalid or non-existent ID returns 500

Steps:

1. Send GET to path with ‚/9999' at the end

Expected: status 400 or 404 with message
Actual: status 500 „Unexpected error"

GET with incorrect param convention returns 500

Steps:

1. Send a GET Request with Incorrect Parameter Convention:
    - Construct a GET request with parameters that do not follow the expected format (e.g., *GET /students?id=abc*, *GET /students?id=123*)

Expected: status 400
Actual: status 500 Internal Server Error

POST with existing ID and all fields the same is not validated

Steps:

1. Send a POST request to the endpoint with an existing ID
2. Include all fields (*firstName*, *lastName*, *email*, *age*) with the exact same values as currently stored

Expected: status 400 Bad Request
Actual: status 200 OK

Email is not validated

Steps:

1. Send a POST request to create or update a student with various invalid email formats in the email field (e.g., *12345*, *true*, *notanemail*)

Expected: status 400 Bad Request.
Actual: status 200 OK

When non-integer age is sent, it is rounded down without notifying user

Steps:
1. Send POST request with age: 29.9

Expected: response with age:29.9 OR response with age 29 and message "Your age was rounded down"
Actual: response with age:29, input data modified without message

Validations (age, corrupt JSON) have no message

Steps:
1. Send any incorrect request

Expected: specific error messages
Actual: status 200

POST with Empty payload returns 500

Steps:
1. Send a POST request to the API with an empty body: { }.

Expected: status 400 Bad Request
Actual: status 500 Internal Server Error

Duplicate students can be created

Steps:
1. Send POST request with data identical to already existing student

Expected: Status 400, warning message about duplicate data
Actual: Status 200 OK

ID of deleted student is not reused (create, receive ID 1819, delete, create, receive ID 1820)

Steps:
1. Create a Student
    - Send a POST request to create a new student (*1819*).
2. Delete the Student
    - Send a DELETE request with the ID *1819*
    - Confirm that the student is successfully deleted and verify that 1819 is no longer in the database.
3. Create Another Student
    - Send another POST request to create a new student after the deletion.
    - Receive a new ID, e.g., 1820.

Expected:
- The ID 1819 should be reused for the new student if IDs are intended to be recycled after deletion.
- The new student should receive the same ID 1819 that was previously deleted.

Actual:
- The new student receives a different ID, e.g., 1820, even though the ID 1819 was available after the deletion.

- The API does not reuse IDs from deleted records.

DELETE with invalid or non-existent ID returns 500

Steps:

1. Send a DELETE request to the API with an invalid or non-existent ID in the URL

Expected: status 400 or 404
Actual: status 500 Internal Server Error