

Joshua Atmadjaya

535230052

TI – B

Back-End Programming

Exercise API Project

1. Prevent Duplicate Email

GET USERS POST AUTHEN... POST POST US... x +

POST http://localhost:5000/api/users

Query Body Headers¹ Auth Vars Script Assert Tests Docs

JSON Prettify

```
1 {
2   "name" : "MakotoMachiYuki",
3   "email" : "MakotoMachiYuki@gmail.com",
4   "password": "123456",
5   "confirm_password": "123456"
6 }
```

Response Headers Timeline Tests

200 OK 4.99s 62B

```
1 {
2   "name": "MakotoMachiYuki",
3   "email": "MakotoMachiYuki@gmail.com"
4 }
```

GET USERS POST AUTHEN... POST POST US... x +

POST http://localhost:5000/api/users

Query Body Headers¹ Auth Vars Script Assert Tests Docs

JSON Prettify

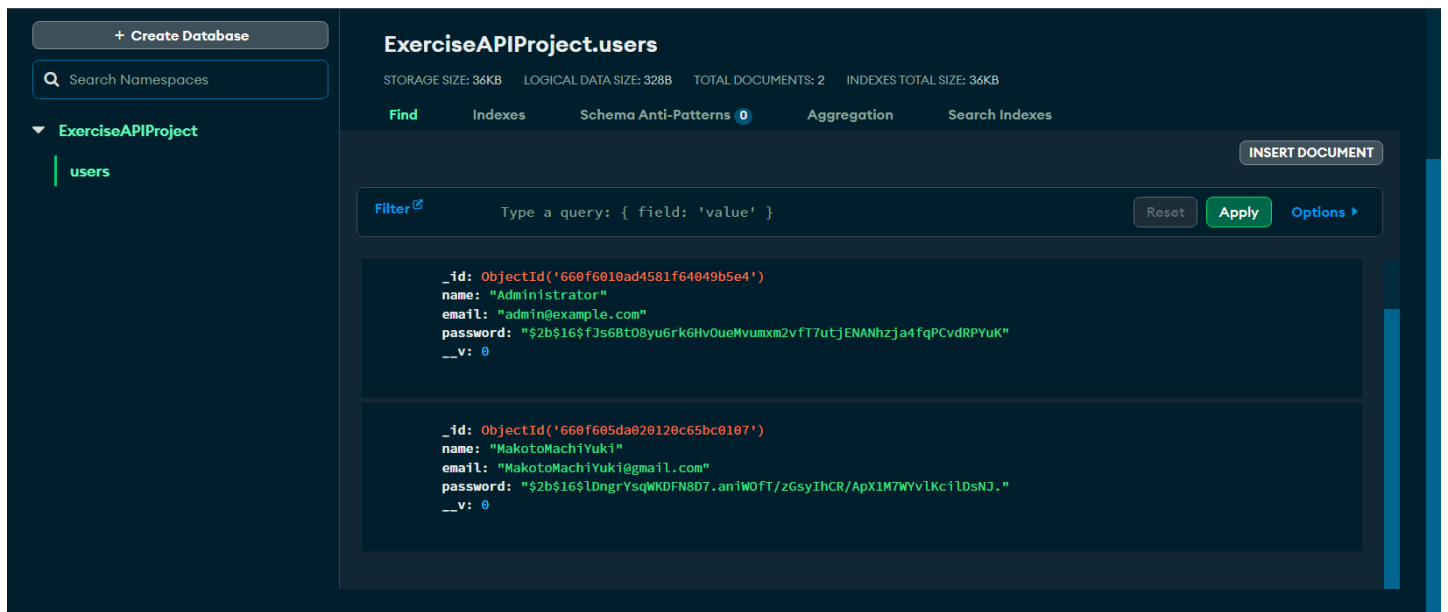
```
1 {
2   "name" : "MakotoMachiYuki",
3   "email" : "MakotoMachiYuki@gmail.com",
4   "password": "123456",
5   "confirm_password": "123456"
6 }
```

Response Headers Timeline Tests

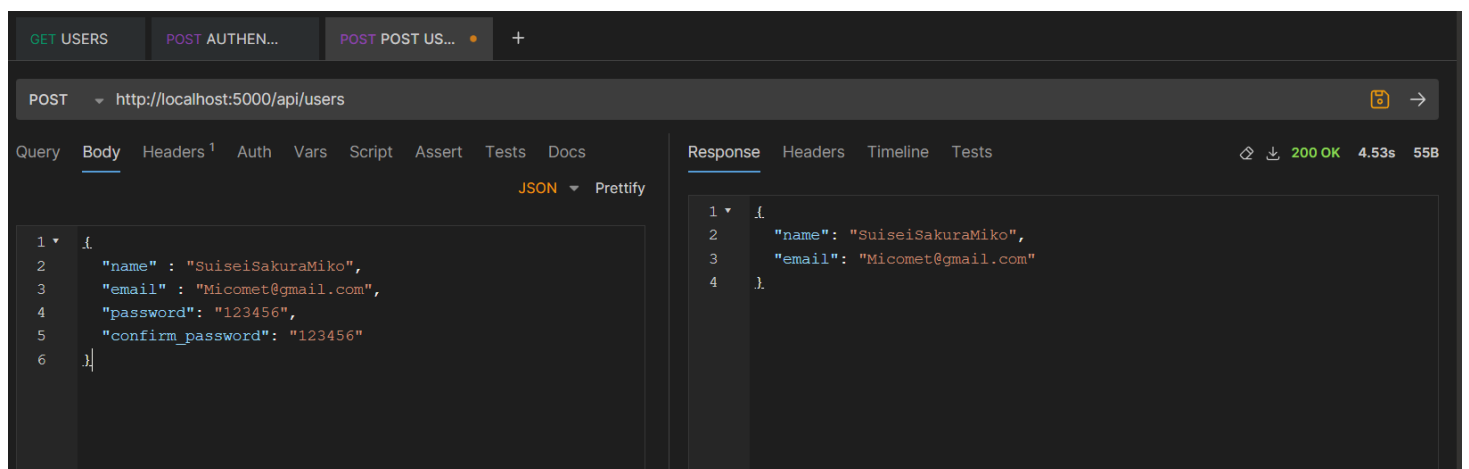
409 Conflict 82ms 147B

```
1 {
2   "statusCode": 409,
3   "error": "EMAIL_ALREADY_TAKEN_ERROR",
4   "description": "This email already taken, try use another",
5   "message": "Email is already taken"
6 }
```

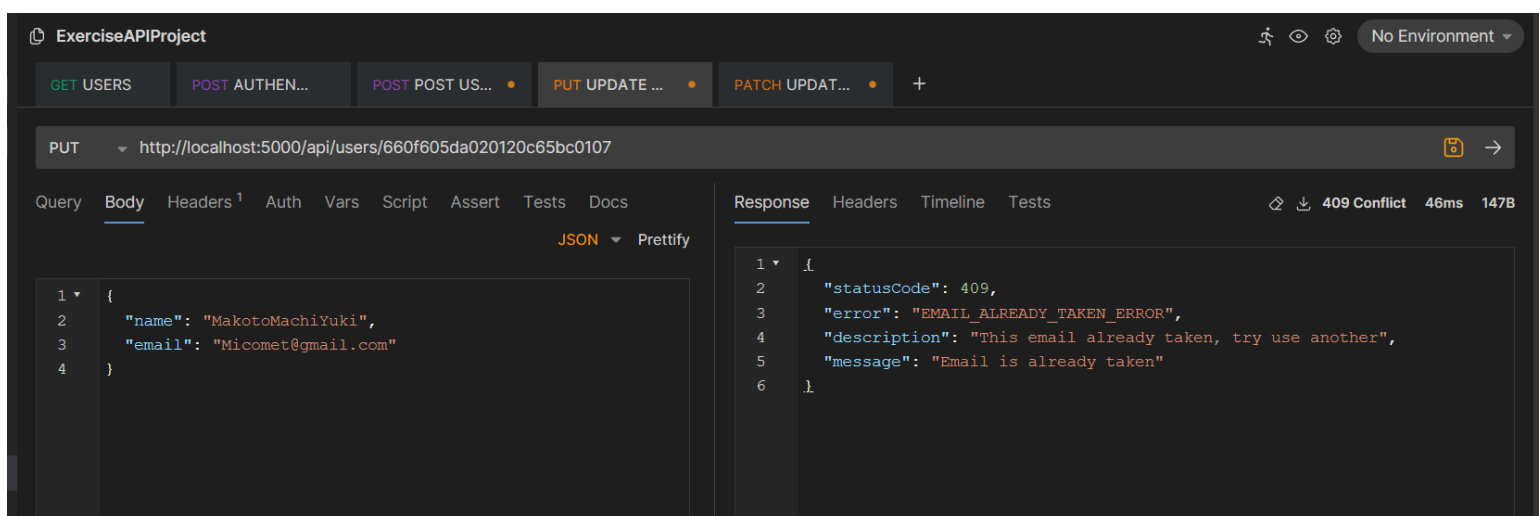
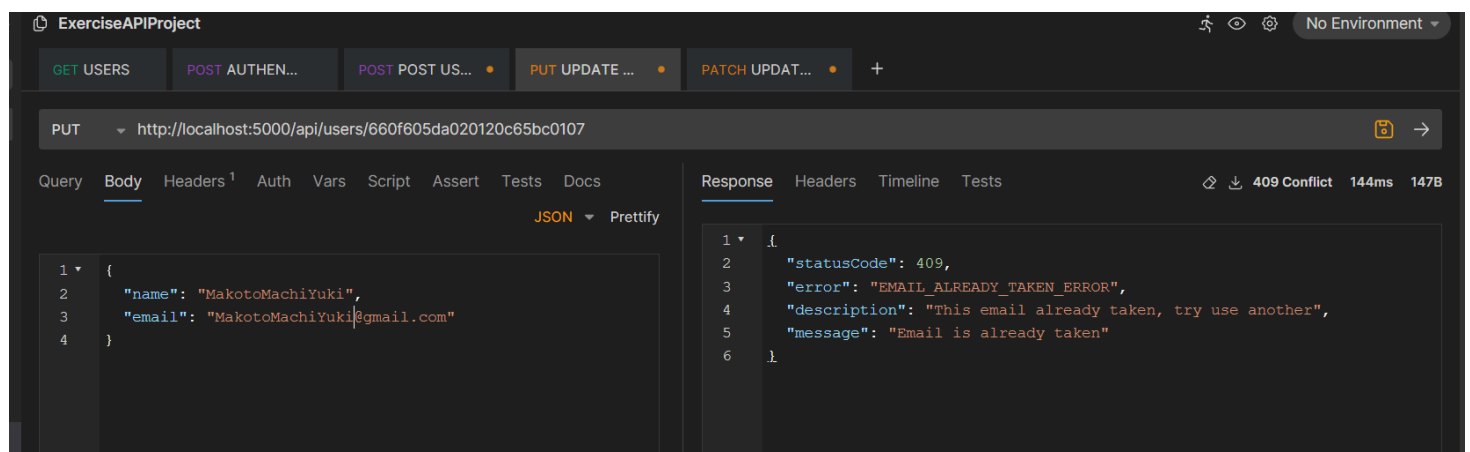
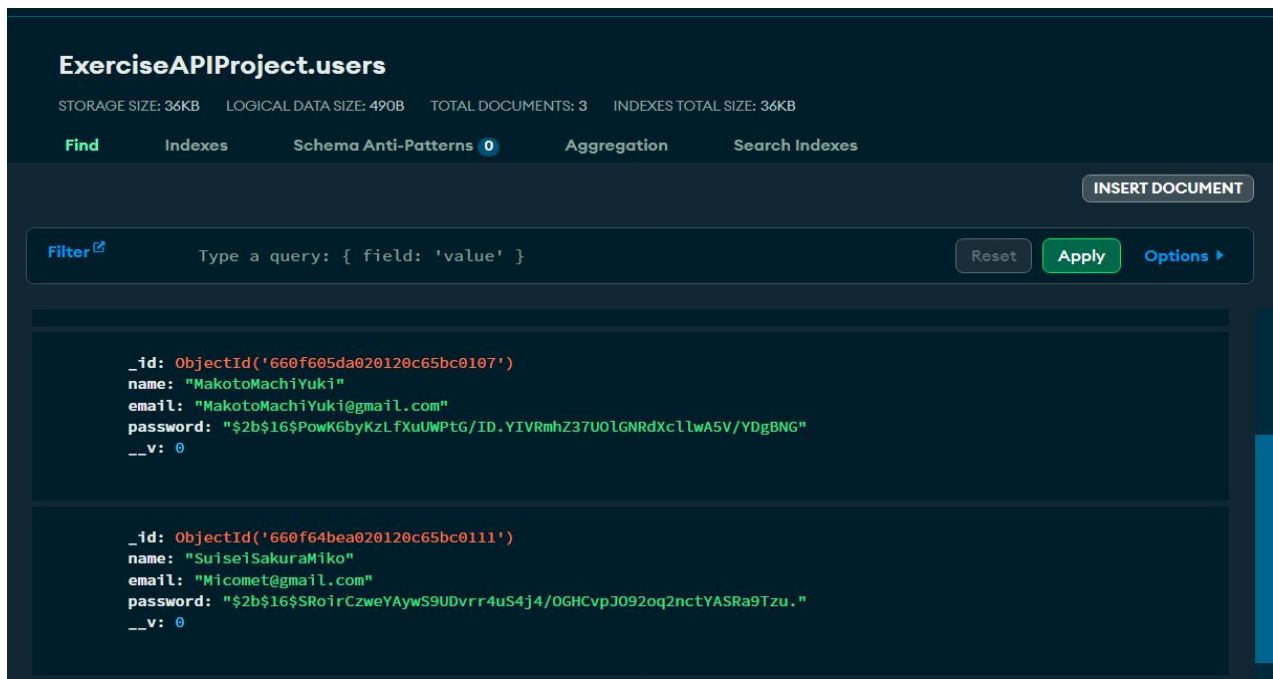
Ketika email yang sudah dicreate dan masuk terdalam database mongodbnnya.. Maka email yg sama tidak akan bisa digunakan lagi karena akan mengembalikan error message 409 dan tidak akan masuk kedalam databasenya



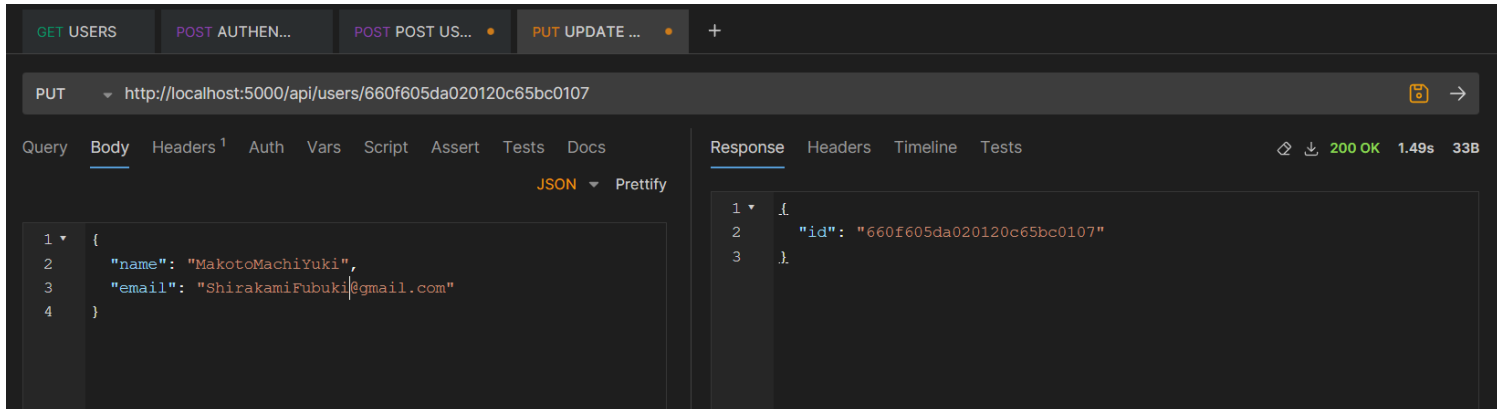
Dan tentu saja email yg berbeda bisa dicreate



Dan juga ketika emailnya mau diupdate melewati put dengan email sebelumnya maupun email yg sudah terdaftar di databasenya tidak akan bisa diupdate



Dan jika berhasil maka akan return id nya dan emailnya sudah terupdate di databasenya (User yg diupdate adalah user yg email sebelumnya MakotoMachiYuki@gmail.com menjadi ShirakamiFubuki@gmail.com)



The screenshot shows a REST client interface with a PUT request to `http://localhost:5000/api/users/660f605da020120c65bc0107`. The request body is a JSON object with `name` and `email` fields. The response is a JSON object with an `id` field.

```
PUT http://localhost:5000/api/users/660f605da020120c65bc0107
```

Query Body Headers¹ Auth Vars Script Assert Tests Docs

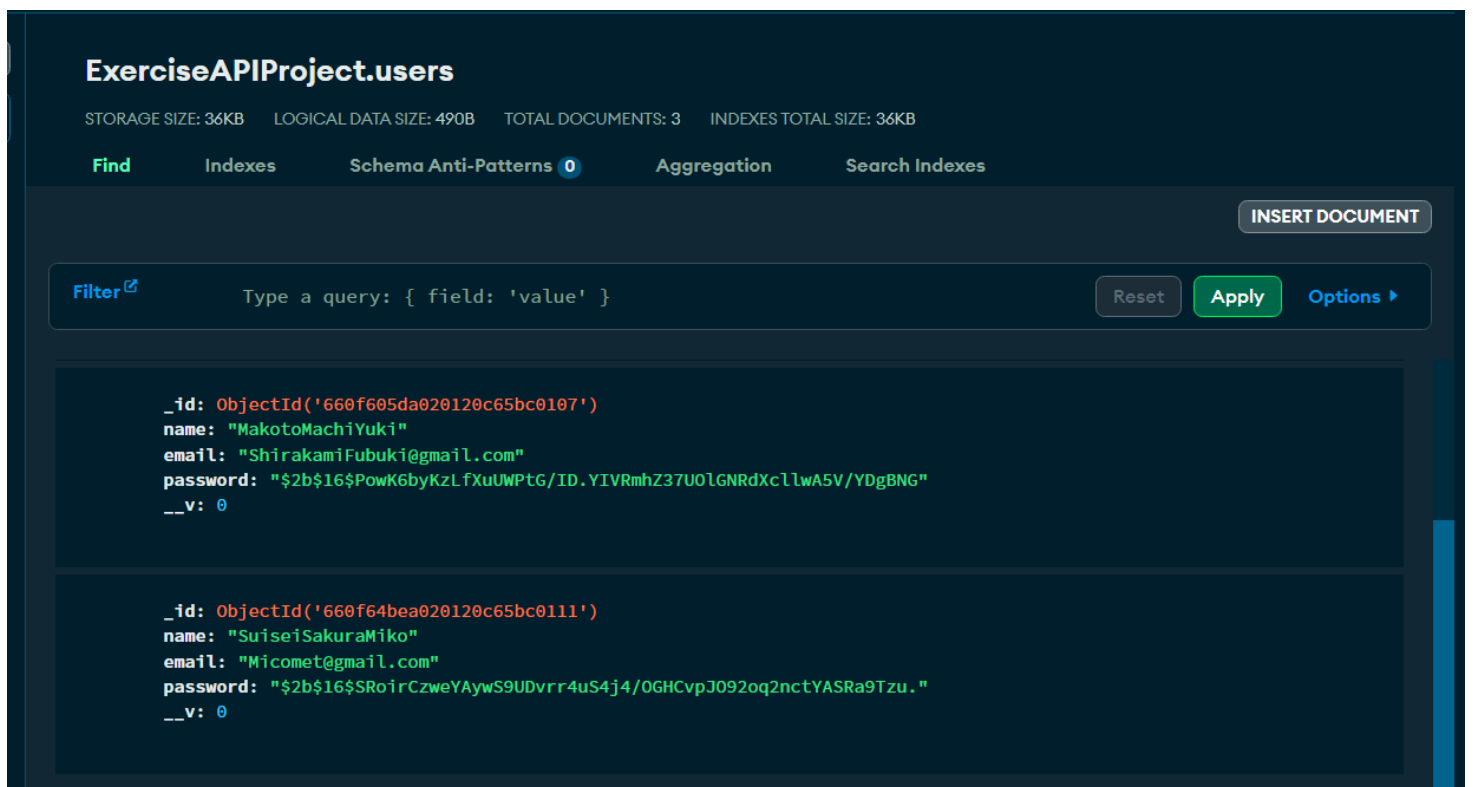
JSON Prettify

```
1 {
2   "name": "MakotoMachiYuki",
3   "email": "ShirakamiFubuki@gmail.com"
4 }
```

Response Headers Timeline Tests

```
1 {
2   "id": "660f605da020120c65bc0107"
3 }
```

200 OK 1.49s 33B



The screenshot shows the MongoDB Atlas interface for the `ExerciseAPIProject.users` collection. It displays the storage size, logical data size, total documents, and indexes total size. The `Find` tab is active, showing a filter query and two documents.

ExerciseAPIProject.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 490B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

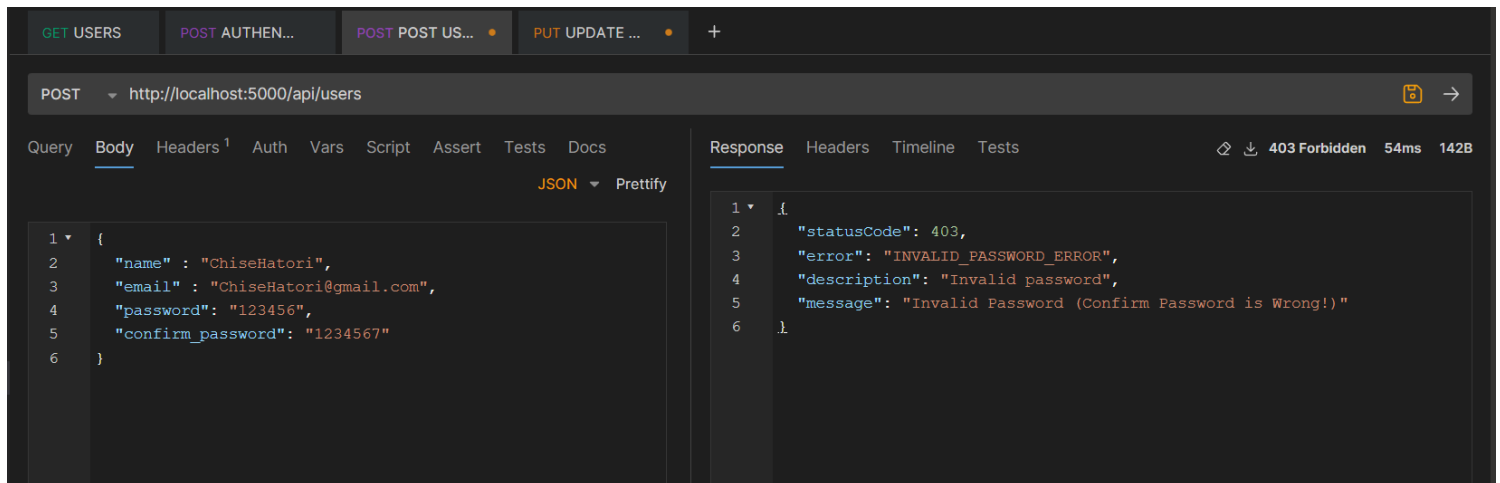
INSERT DOCUMENT

Filter [Filter](#) Type a query: { field: 'value' } Reset Apply Options

```
_id: ObjectId('660f605da020120c65bc0107')
name: "MakotoMachiYuki"
email: "ShirakamiFubuki@gmail.com"
password: "$2b$16$PowK6byKzLfXuUWPtG/ID.YIVRmhZ37U0lGNRdXcLlwA5V/YDgBNG"
__v: 0
```

```
_id: ObjectId('660f64bea020120c65bc0111')
name: "SuiSeiSakuraMiko"
email: "Micomet@gmail.com"
password: "$2b$16$SRoirCzweYaywS9UDvrr4uS4j4/0GHCvpJ092oq2nctYASRa9Tzu."
__v: 0
```

2. Confirm Password



POST `http://localhost:5000/api/users`

Query Body Headers¹ Auth Vars Script Assert Tests Docs

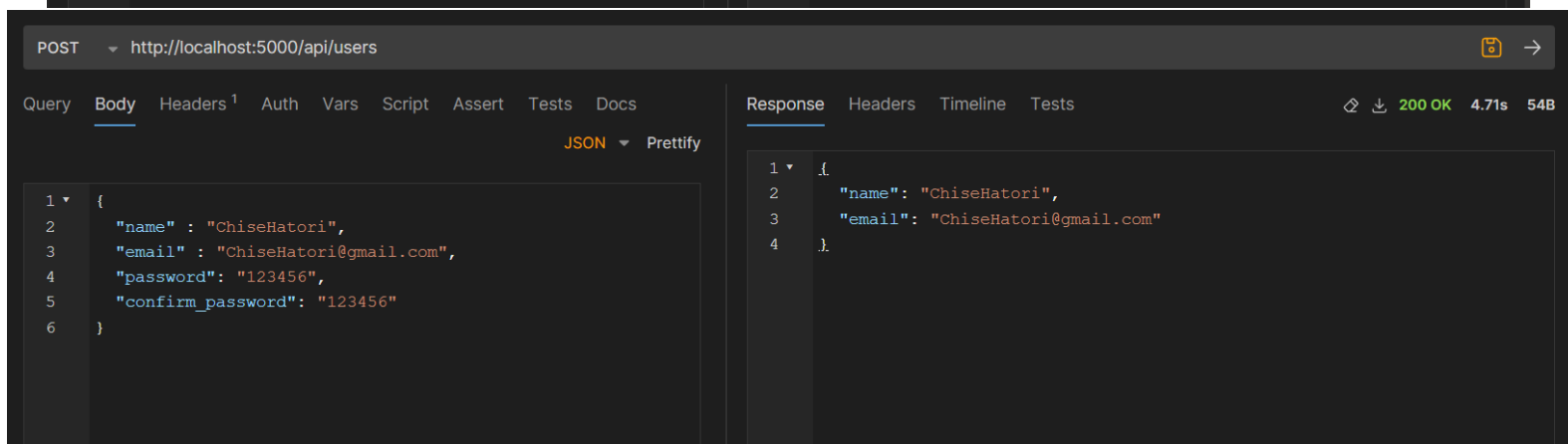
JSON ▾ Prettify

```
1 {
2   "name" : "ChiseHatori",
3   "email" : "ChiseHatori@gmail.com",
4   "password": "123456",
5   "confirm_password": "1234567"
6 }
```

Response Headers Timeline Tests

🔗 ⬇ 403 Forbidden 54ms 142B

```
1 {
2   "statusCode": 403,
3   "error": "INVALID_PASSWORD_ERROR",
4   "description": "Invalid password",
5   "message": "Invalid Password (Confirm Password is Wrong!)"
6 }
```



POST `http://localhost:5000/api/users`

Query Body Headers¹ Auth Vars Script Assert Tests Docs

JSON ▾ Prettify

```
1 {
2   "name" : "ChiseHatori",
3   "email" : "ChiseHatori@gmail.com",
4   "password": "123456",
5   "confirm_password": "123456"
6 }
```

Response Headers Timeline Tests

🔗 ⬇ 200 OK 4.71s 54B

```
1 {
2   "name": "ChiseHatori",
3   "email": "ChiseHatori@gmail.com"
4 }
```

Dan jika `confirm_password` tidak sama dengan `password` maka akan mengembalikan error message “Invalid Password (Confirm Password is Wrong!) dengan kode 403

3. Change Password

Password awalnya adalah 123456 yg dimana ketika old_passwordnya diisi dengan password yg salah maka akan mengembalikan error message 403 dengan message “Old Password is Wrong”

The screenshot shows a REST client interface with a PATCH request to `http://localhost:5000/api/users/660f605da020120c65bc0107/change-password`. The request body is a JSON object with the following fields:

```
{
  "old_password": "1234567",
  "new_password": "12345689",
  "confirm_password": "12345689"
}
```

The response is a JSON object with the following fields:

```
{
  "statusCode": 403,
  "error": "INVALID_PASSWORD_ERROR",
  "description": "Invalid password",
  "message": "Invalid Password (Old Password is Wrong!)"
}
```

The status bar indicates a 403 Forbidden response with a 4.70s duration and 138B body size.

Begitu pula jika confirm_password tidak sama dengan new_password akan mengembalikan message confirm_password is wrong

The screenshot shows a REST client interface with a PATCH request to `http://localhost:5000/api/users/660f605da020120c65bc0107/change-password`. The request body is a JSON object with the following fields:

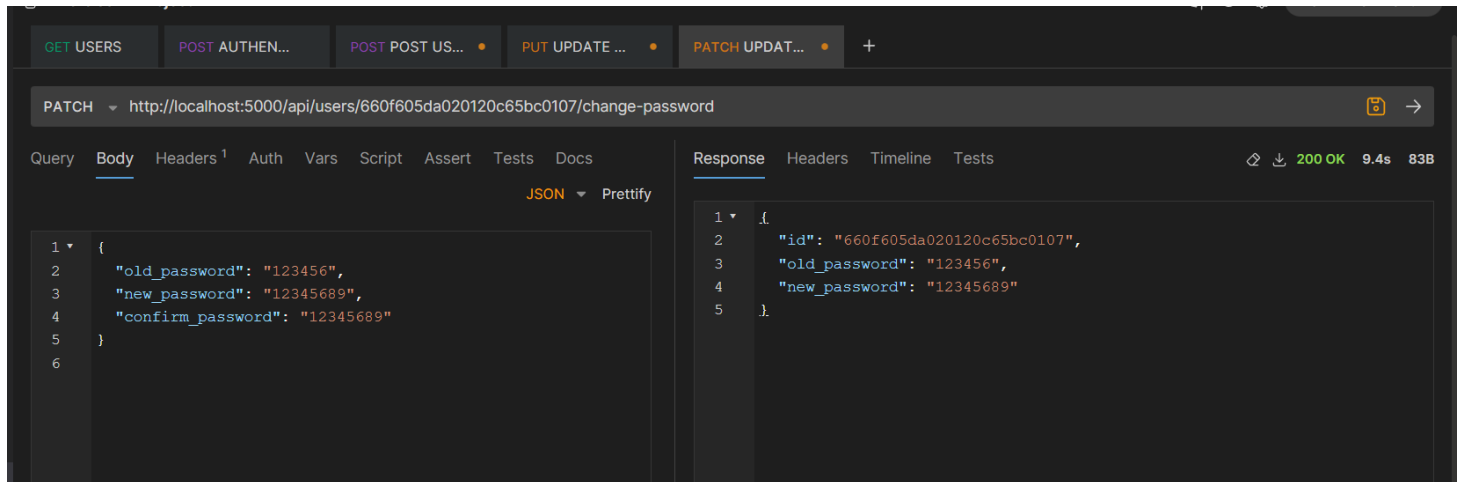
```
{
  "old_password": "123456",
  "new_password": "12345689",
  "confirm_password": "1234568"
}
```

The response is a JSON object with the following fields:

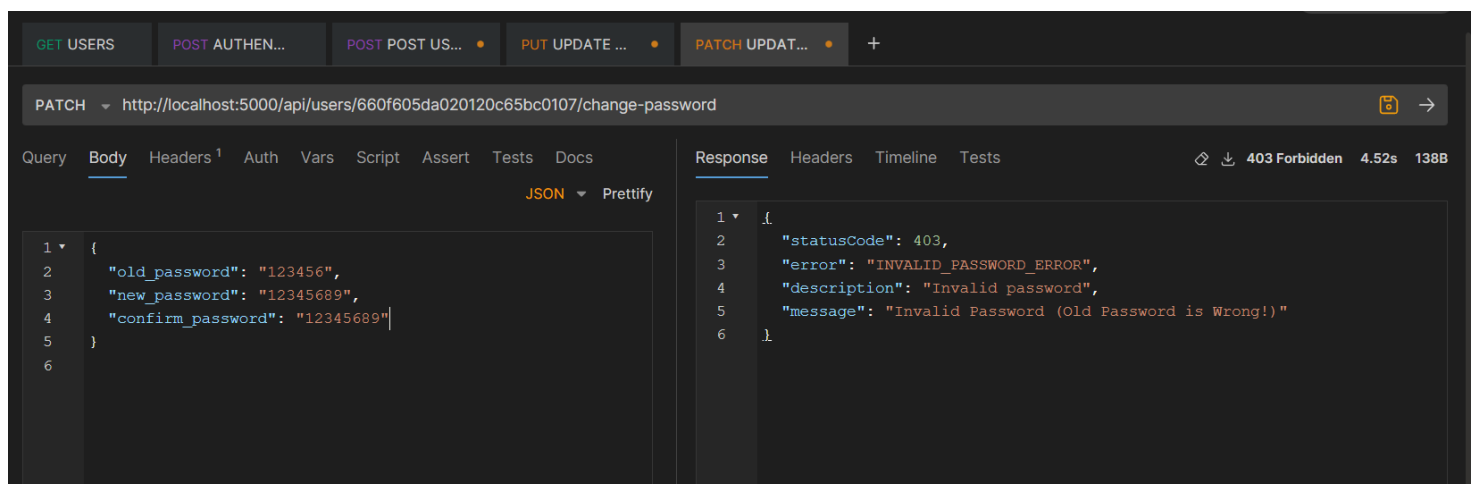
```
{
  "statusCode": 403,
  "error": "INVALID_PASSWORD_ERROR",
  "description": "Invalid password",
  "message": "Invalid Password (Confirm Password is Wrong!)"
}
```

The status bar indicates a 403 Forbidden response with a 2.84s duration and 142B body size.

Dan ketika semua kriteria terpenuhi, `old_password` benar dan `confirm_password` benar maka akan return di response `id`, `old_password` dan `new_password`nya yg akan jadi password sekarang ini



Buktinya adalah ketika body yg sama di send maka akan error karena passwordnya sudah terganti didalam databasenya



Dan ketika `old_password`nya menggunakan password yg barusan diganti (12345689) maka akan sukses

