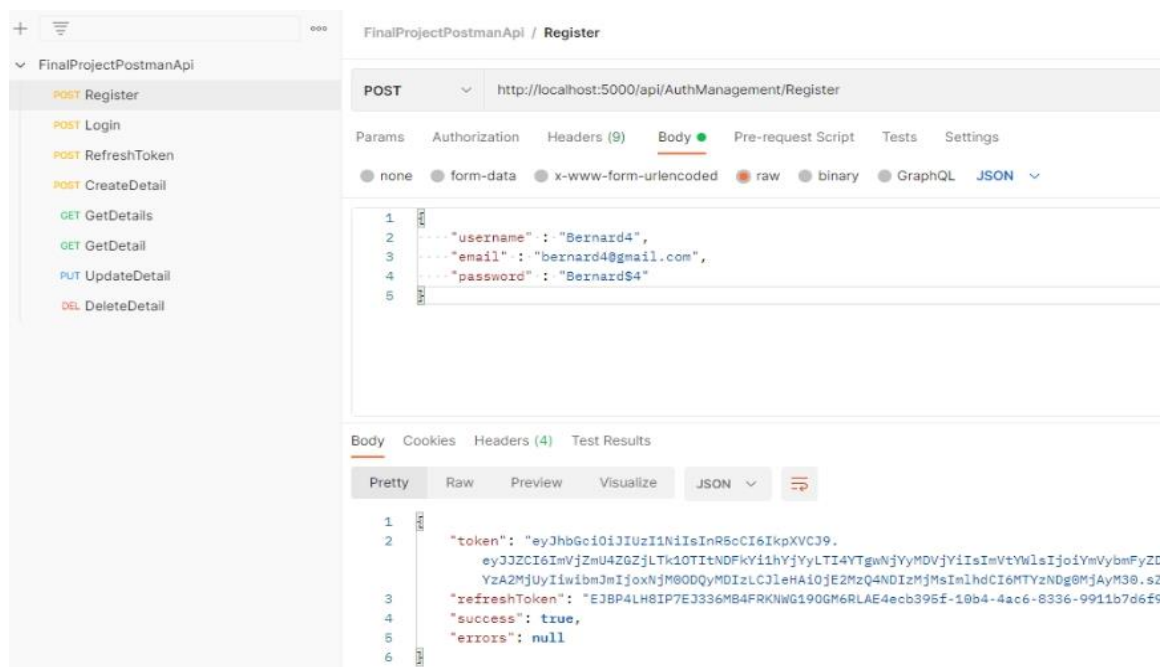


Tugas : Final Project

- Local MySQL, melalui dotnet run, f5 ataupun run iis express pada vs studio (hidupkan mysql dan apache pada xampp) kemudian buka <https://localhost:5001/swagger/index.html>
- Docker menggunakan database local ataupun online Remotemysql, melalui run container docker dan testing melalui postman
- Paymentapi28.herokuapp.com ( setelah push Heroku, juga testing melalui postman )

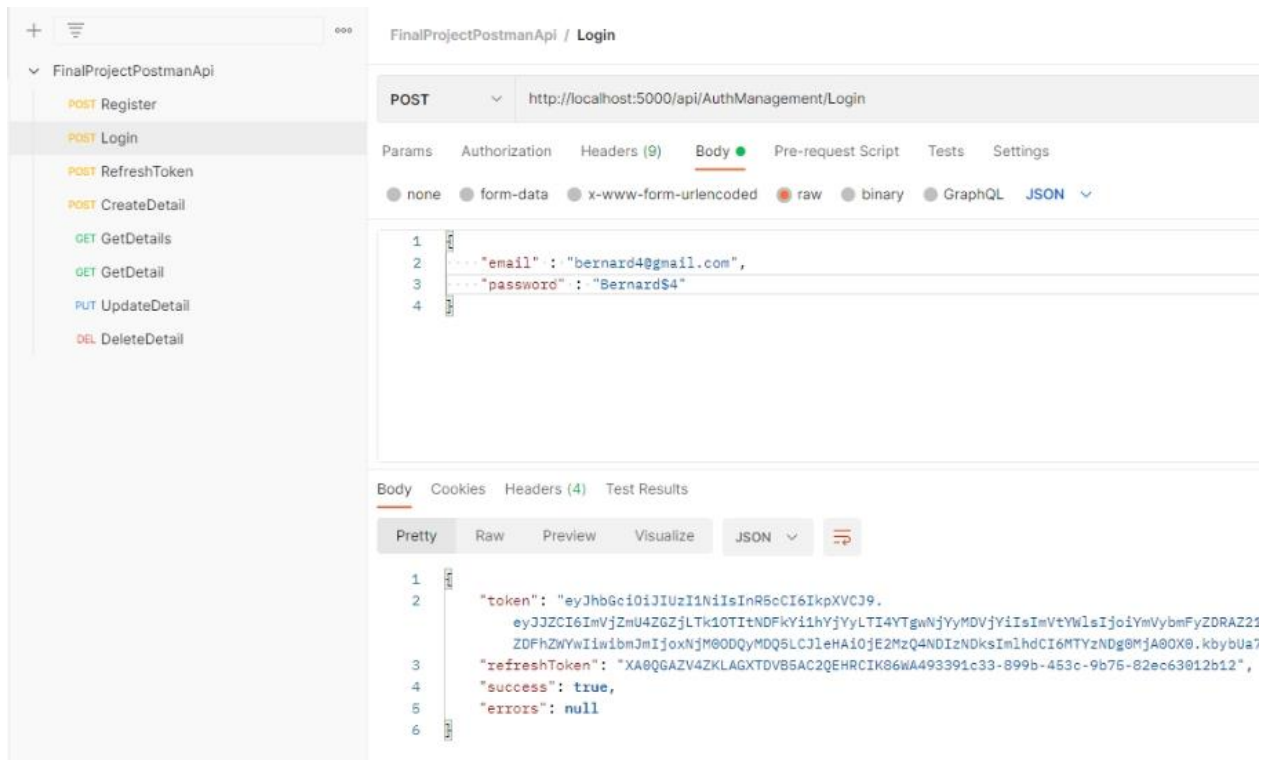
Pada endpoint ini user akan menekan tombol body dan mengubah format text menjadi “JSON”, kemudian user akan memasukkan 3 input yaitu “username”, “email”, “password” pada body dengan format JSON dan akan menekan tombol send. Jika berhasil akan muncul output berupa “token”, “refreshToken”, status “success” dan juga “errors” jika ada. Jika tidak berhasil maka “errors” akan menampilkan status error yang diterima.



## 2. Login (<http://localhost:5000/api/AuthManagement/Login>)

Pada endpoint ini user akan memasukkan 2 input yaitu “email” dan “password” yang telah didaftarkan sebelumnya pada body dengan format JSON dan akan menekan tombol send.

Jika berhasil akan muncul output berupa “token”, “refreshToken”, status “success” dan juga “errors” jika ada. Jika tidak berhasil maka “errors” akan menampilkan status error yang diterima.



## 3. RefreshToken (<http://localhost:5000/api/AuthManagement/RefreshToken>)

Pada endpoint ini user akan memasukkan 2 input yaitu “token” dan “refreshToken” yang telah didapatkan saat melakukan login, dan akan diinput pada body dengan format JSON dan akan menekan tombol send.

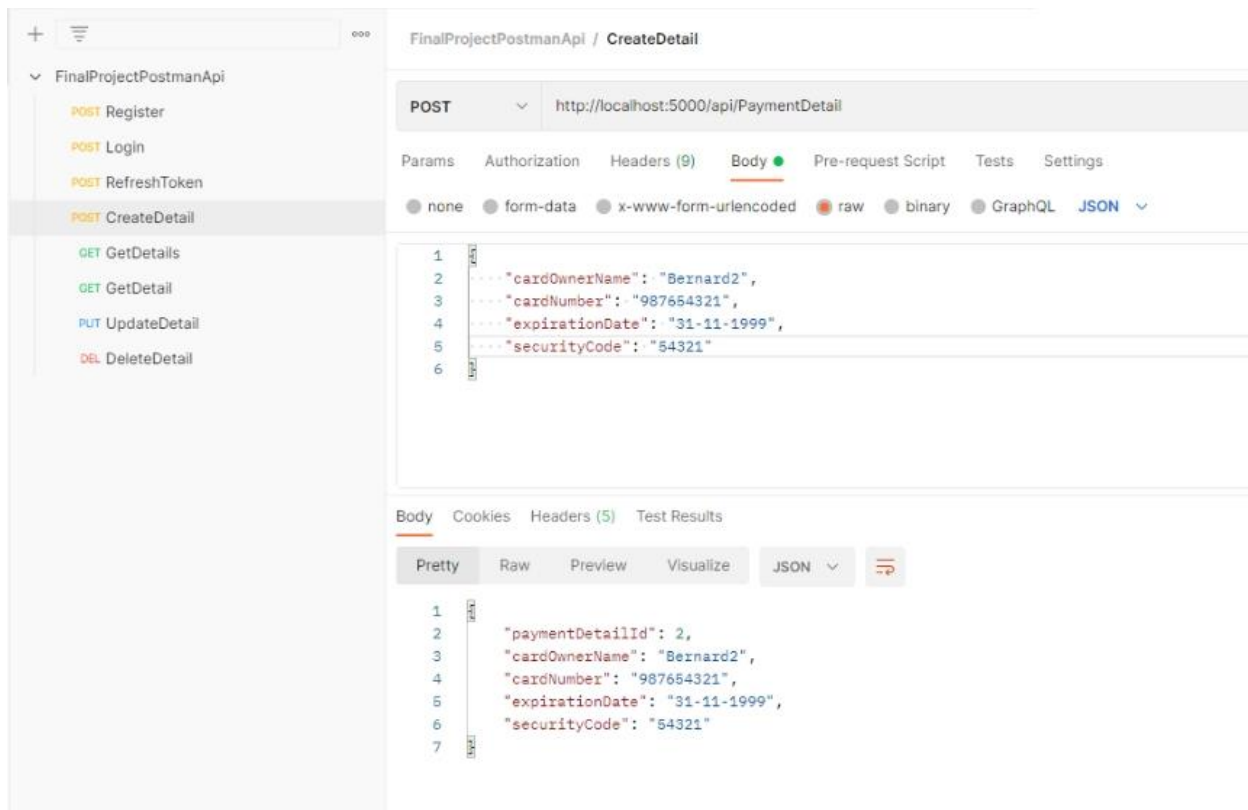
Jika berhasil akan muncul output berupa “token” yang baru, “refreshToken”, status “success” dan juga “errors” jika ada. Jika tidak berhasil maka “errors” akan menampilkan status error yang diterima.



#### 4. CreateDetail (<http://localhost:5000/api/PaymentDetail>)

Pada endpoint ini user akan memasukkan 4 input yaitu “cardOwnerName”, “cardNumber”, “expirationDate”, dan “securityCode” sesuai dengan informasi ataupun data diri yang dipunya, kemudian akan menekan tombol send.

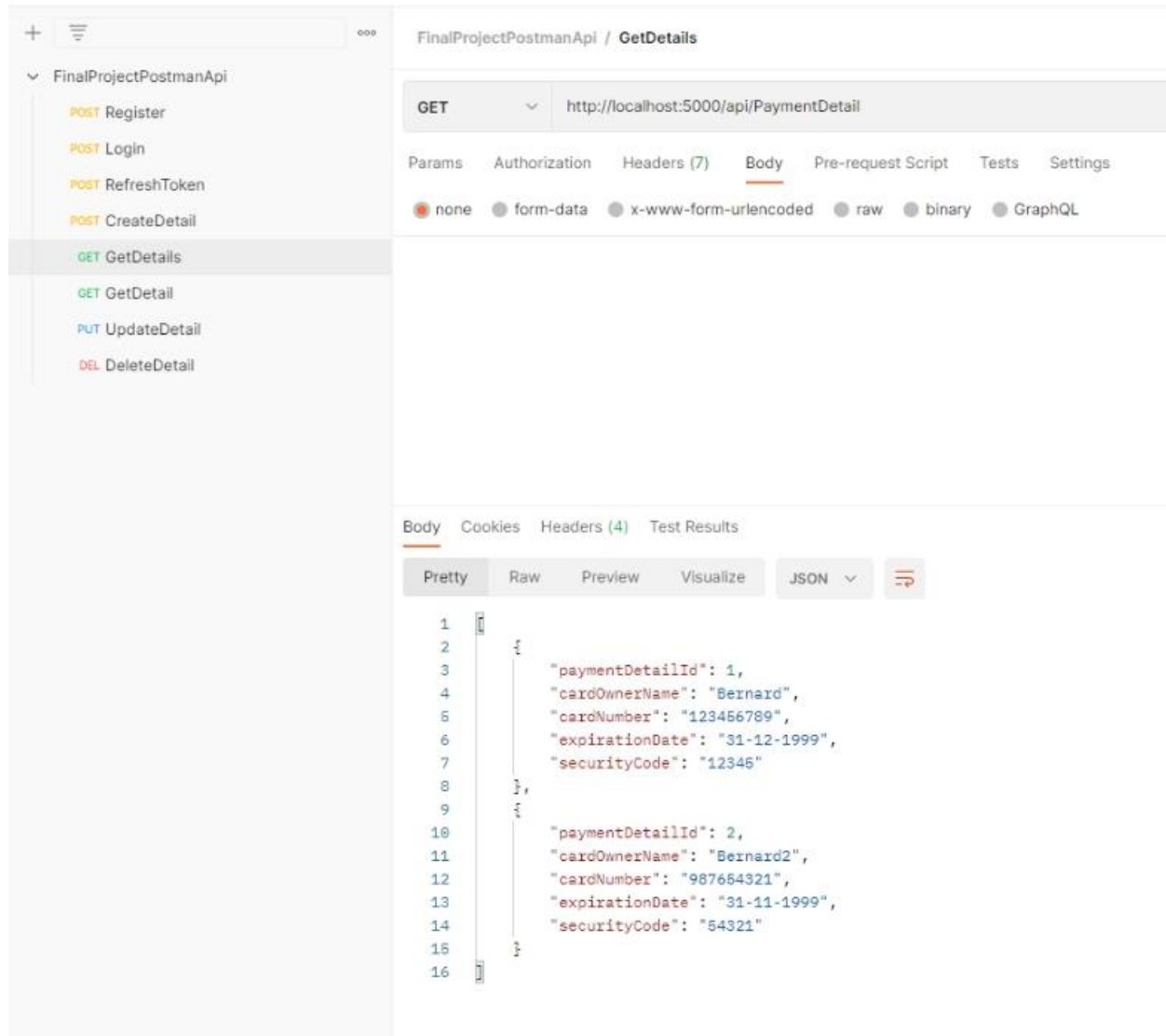
Jika berhasil akan muncul output berupa data yang berhasil terdaftar seperti “paymentDetailId”, “cardOwnerName”, “cardNumber”, “expirationDate”, dan “securityCode”. Jika tidak berhasil maka akan muncul output “errors” yang menampilkan status error yang diterima.



#### 5. GetDetails (<http://localhost:5000/api/PaymentDetail>)

Pada endpoint ini user akan memasukkan 5 input yaitu “cardOwnerName”, “cardNumber”, “expirationDate”, dan “securityCode” sesuai dengan informasi ataupun data diri yang dipunya, kemudian akan menekan tombol send.

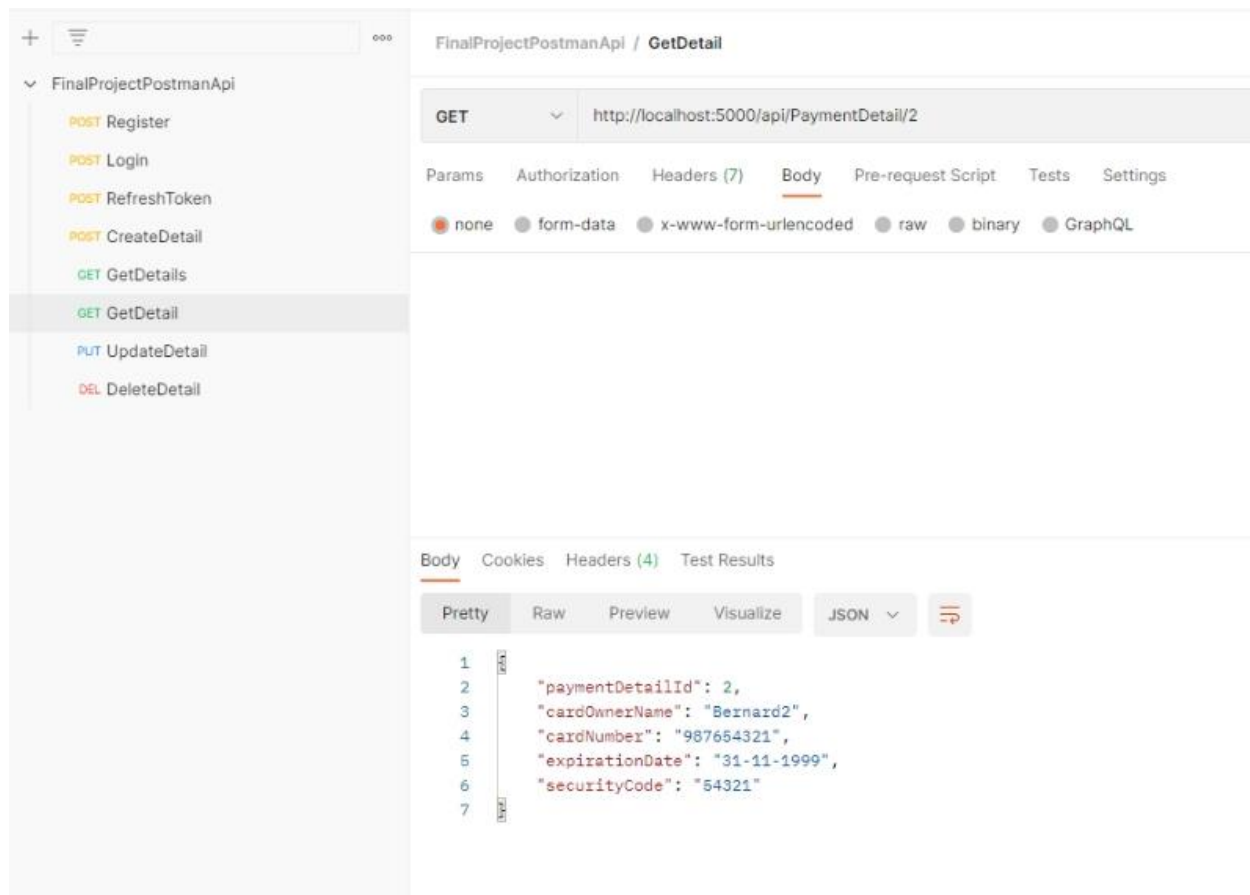
Jika berhasil akan muncul output berupa seluruh data yang ada dengan atribut lengkap seperti “paymentDetailId”, “cardOwnerName”, “cardNumber”, “expirationDate”, dan “securityCode”. Jika tidak berhasil maka akan muncul output “errors” yang menampilkan status error yang diterima.



## 6. GetDetail (<http://localhost:5000/api/PaymentDetail/{id}>)

Pada endpoint ini user akan memasukkan 1 input pada { } pada url endpoint yang dapat diisi dengan angka/id dari "paymentDetailId" yang ingin dicari, kemudian akan menekan tombol send.

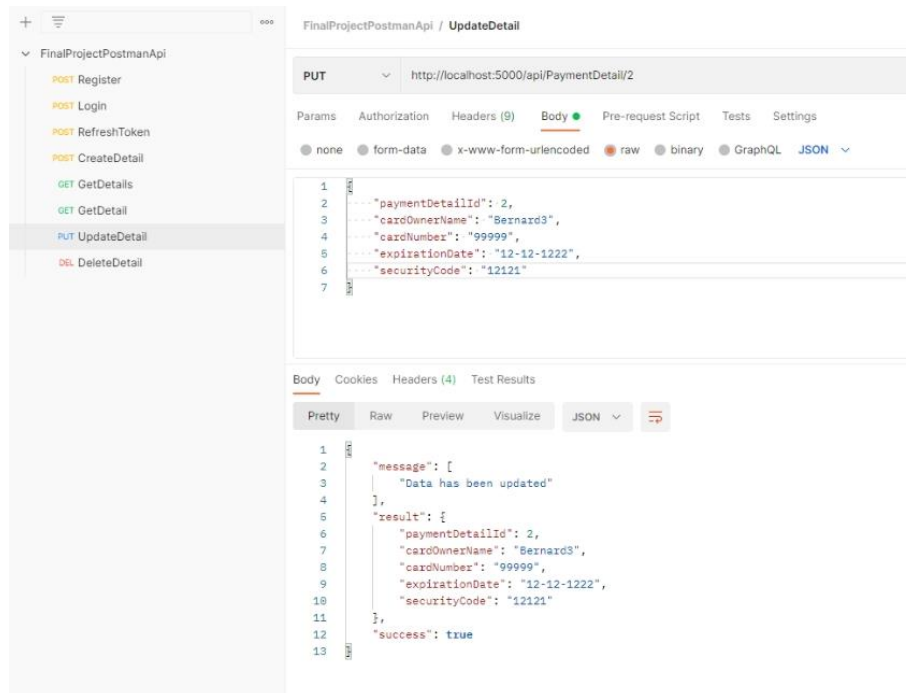
Jika berhasil akan muncul output berupa data dengan atribut lengkap seperti "paymentDetailId", "cardOwnerName", "cardNumber", "expirationDate", dan "securityCode". yang sesuai dengan "paymentDetailId" yang diinput pada { }. Jika tidak berhasil maka akan muncul output "errors" yang menampilkan status error yang diterima.



## 7. UpdateDetail (http://localhost:5000/api/PaymentDetail/{})

Pada endpoint ini user akan memasukkan 1 input pada {} pada url endpoint yang dapat diisi dengan angka/id dari "paymentDetailId" dari data yang ingin diupdate ataupun diganti, kemudian akan menekan tombol send. Kemudian pada bagian body user juga akan menginput detail dari perubahan data yang akan diterapkan, dan kemudian akan menekan tombol send.

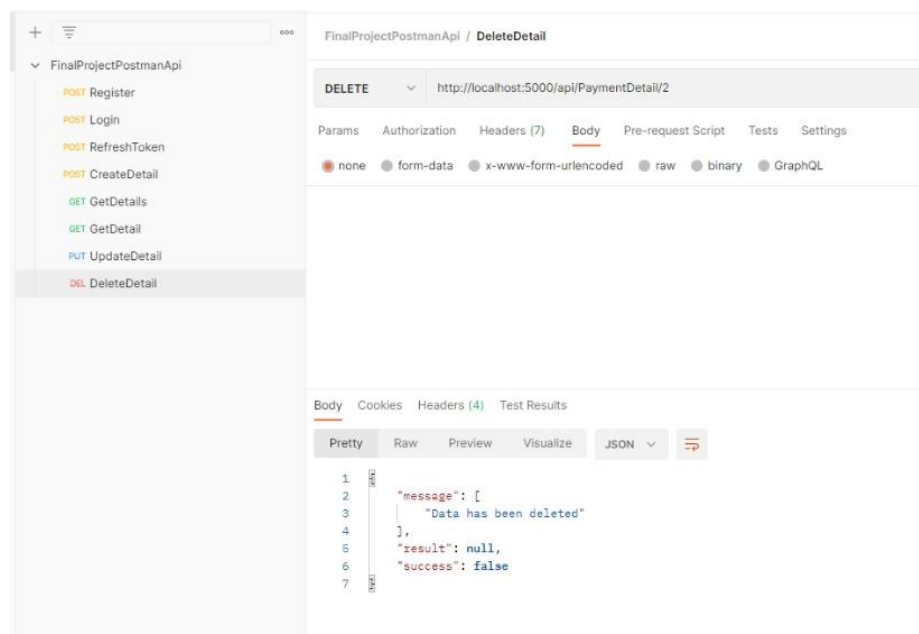
Jika berhasil akan muncul output berupa data dengan atribut lengkap seperti "paymentDetailId", "cardOwnerName", "cardNumber", "expirationDate", dan "securityCode". yang telah diubah. Jika tidak berhasil maka akan muncul output "errors" yang menampilkan status error yang diterima.



## 8. DeleteDetail (<http://localhost:5000/api/PaymentDetail/{id}>)

Pada endpoint ini user akan memasukkan 1 input pada `{}` pada url endpoint yang dapat diisi dengan angka/id dari `paymentDetailId` dari data yang ingin dihapus, kemudian akan menekan tombol send.

Jika berhasil akan muncul output berupa pemberitahuan bahwa data telah berhasil dihapus. Jika tidak berhasil maka akan muncul output `"errors"` yang menampilkan status error yang diterima.



Sama halnya dengan menggunakan postman user akan memasukkan input sesuai dengan contoh format JSON yang ada, bedanya user akan menekan tombol try it out terlebih dahulu, kemudian memasukkan input dan menekan tombol Execute.

```
{
  "username": "string",
  "email": "user@example.com",
  "password": "string"
}
```

```
{
  "username" : "Bernard5",
  "email" : "bernard5@gmail.com",
  "password" : "Bernard%5"
}
```

```
response body
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Username 'Bernards' is already taken."
  ]
}
```

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Passwords must have at least one non alphanumeric character.",
    "Passwords must have at least one digit ('0'-'9').",
    "Passwords must have at least one uppercase ('A'-'Z')."
  ]
}
```

```
{
  "username": "Bernard6",
  "email": "bernard6@gmail.com",
  "password": "Bernard*6"
}
```

**Output Jika memasukkan data dengan benar**



## 2. Login

```
{
  "email": "user@example.com",
  "password": "string"
}
```

## Format JSON

```
{
  "email": "bernard5@gmail.com",
  "password": "Bernard^6"
}
```

### Input Data yang salah

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Invalid login request"
  ]
}
```

### Validasi untuk input data yang tidak sesuai

```
{
  "email": "bernard6@gmail.com",
  "password": "Bernard^6"
}
```

### Input Data yang benar

[illegible]

**Output jika data yang dimasukan sesuai / benar**

### 3. RefreshToken

```
{
  "token": "string",
  "refreshToken": "string"
}
```

## Format JSON

[illegible]

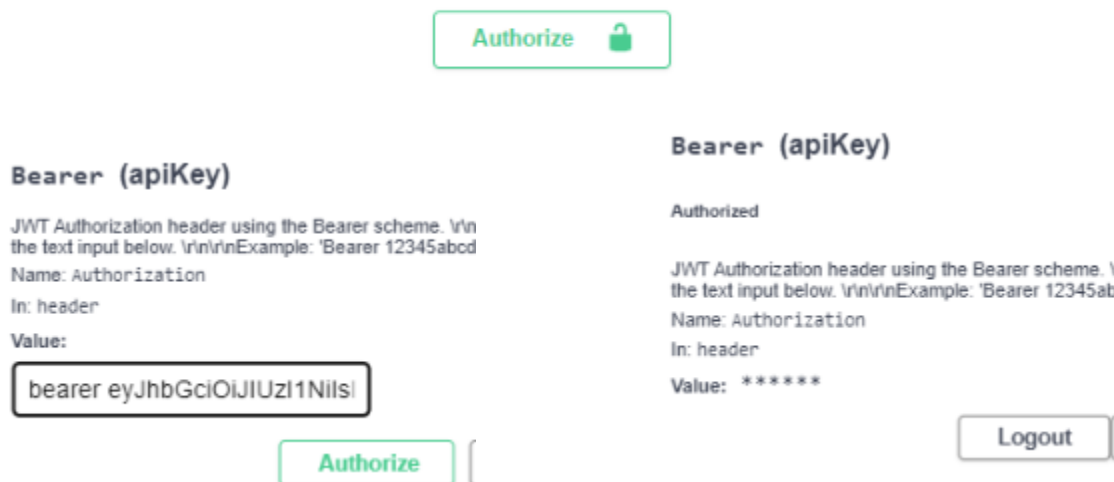
### Input Token yang didapat dari sesi login

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Token has not yet expired"
  ]
}
```

**Output yang dikarenakan sesi login yang belum expired ( 10 menit )**

## PaymentDetail

Pada **PaymentDetail** di swagger user harus terotorisasi terlebih dahulu sebelum dapat mengakses endpoint PaymentDetail dengan cara menekan tombol authorization berwarna hijau dan kemudian memasukkan token dengan format “Bearer” + “token”, dimana token yang harus dimasukkan adalah token yang didapat dari sesi Login.



#### 4. GetDetails

```
[
  {
    "paymentDetailId": 1,
    "cardOwnerName": "Bernard",
    "cardNumber": "123456789",
    "expirationDate": "31-12-1999",
    "securityCode": "12345"
  }
]
```

Tampilan data yang ada pada database

#### 5. CreateDetail

```
{
  "paymentDetailId": 0,
  "cardOwnerName": "string",
  "cardNumber": "string",
  "expirationDate": "string",
  "securityCode": "string"
}
```

Format JSON

POST	/api/PaymentDetail
Parameters	
No parameters	
Request body	
<pre>{   "paymentDetailId": 0,   "cardOwnerName": "Bernard",   "cardNumber": "2233445566",   "expirationDate": "10-9-2008",   "securityCode": "12345" }</pre>	

Input data detail baru

```
{
  "paymentDetailId": 0,
  "cardOwnerName": "Bernard",
  "cardNumber": "2233445566",
  "expirationDate": "10-9-2008",
  "securityCode": "12345"
}
```

Output setelah data diinput oleh user

Response body

```
[
  {
    "paymentDetailId": 1,
    "cardOwnerName": "Bernard",
    "cardNumber": "123456789",
    "expirationDate": "31-12-1999",
    "securityCode": "12345"
  },
  {
    "paymentDetailId": 8,
    "cardOwnerName": "Bernard",
    "cardNumber": "2233445566",
    "expirationDate": "10-9-2008",
    "securityCode": "12345"
  }
]
```

Output database saat user melakukan GetDetails, setelah CreateDetail dilakukan

## 6. GetDetail

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Kolom untuk menginput paymentDetailId yang ingin diGET

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Input 8 pada kolom paymentDetailId

Response body

```
{
  "paymentDetailId": 8,
  "cardOwnerName": "Bernard",
  "cardNumber": "2233445566",
  "expirationDate": "10-9-2008",
  "securityCode": "12345"
}
```

Output data dengan paymentDetailId 8 yang telah diGET

```
{
  "message": [
    "Data Not Found"
  ],
  "result": {
    "statusCode": 404
  },
  "success": false
}
```

Jika paymentDetailId yang diinput salah / tidak ada

## 7. UpdateDetail

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Kolom untuk menginput paymentDetailID yang ingin diUpdate

```
{
  "paymentDetailId": 0,
  "cardOwnerName": "string",
  "cardNumber": "string",
  "expirationDate": "string",
  "securityCode": "string"
}
```

Format JSON

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Request body

```
{
  "paymentDetailId": 8,
  "cardOwnerName": "Bernard",
  "cardNumber": "2233445566",
  "expirationDate": "10-9-2008",
  "securityCode": "12345"
}
```

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "Bad Request",
  "status": 400,
  "traceId": "00-e227320a4ec15d4cacc99f192ec0adc9-0195deadf5bb524e-00"
}
```

Output yang muncul jika user menginput paymentDetailId yang tidak sesuai di kedua kolom

```
{
  "message": [
    "Data has been updated"
  ],
  "result": {
    "paymentDetailId": 8,
    "cardOwnerName": "Bernard",
    "cardNumber": "12345",
    "expirationDate": "10-9-2100",
    "securityCode": "54321"
  },
  "success": true
}
```

Output yang muncul jika user menginput data yang sesuai



## 8. DeleteDetail

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Kolom untuk menginput paymentDetailID yang ingin diDelete

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Input 1 pada kolom paymentDetailId

```
{
  "message": [
    "Data has been deleted"
  ],
  "result": null,
  "success": true
}
```

Output yang muncul jika paymentDetailID sesuai / ada

**paymentDetailId** \* required  
integer(\$int32)  
(path)

Input 2 pada kolom paymentDetailID

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
  "title": "Not Found",
  "status": 404,
  "traceId": "00-e226cad04972ef488b8b54e1fc2992-ed7d0bdfd9e12d45-00"
}
```

Output yang muncul jika paymentDetailID tidak sesuai / tidak ada

GET /api/PaymentDetail/{paymentDetailId}

Parameters

Name	Description
<b>paymentDetailId</b> * required integer(\$int32) (path)	<input type="text" value="1"/>

Input Kembali 1 sebagai paymentDetailId pada kolom GetDetail setelah paymentDetailId 1 diDelete

```
{
  "message": [
    "Data Not Found"
  ],
  "result": {
    "statusCode": 404
  },
  "success": false
}
```

Output yang akan muncul adalah “Data Not Found” karena sudah terDelete

The screenshot shows a REST client interface with the following sections:

- Method:** GET
- URL:** /api/PaymentDetail
- Parameters:** No parameters
- Execute:** A blue button to execute the request.
- Responses:**
  - Curl:**

```
curl -X 'GET' \
'https://localhost:5001/api/PaymentDetail' \
-H 'accept: */*' \
-H 'Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV'
```
  - Request URL:** https://localhost:5001/api/PaymentDetail
  - Server response:**

Code	Details
200	<p><b>Response body</b></p> <pre>{   {     "paymentDetailId": 8,     "cardOwnerName": "Bernard",     "cardNumber": "12345",     "expirationDate": "10-9-2100",     "securityCode": "54321"   } }</pre>

Output GetDetails setelah data dengan paymentDetailId 1 diDelete



Sama halnya dengan menggunakan postman melalui localhost, bedanya hanya mengganti url endpoint pada bagian localhost:5000 menjadi paymentapi28.herokuapp.com.

The screenshot shows a REST client interface with a POST request to `https://paymentapi28.herokuapp.com/api/authmanagement/register`. The request body is a JSON object with the following fields:

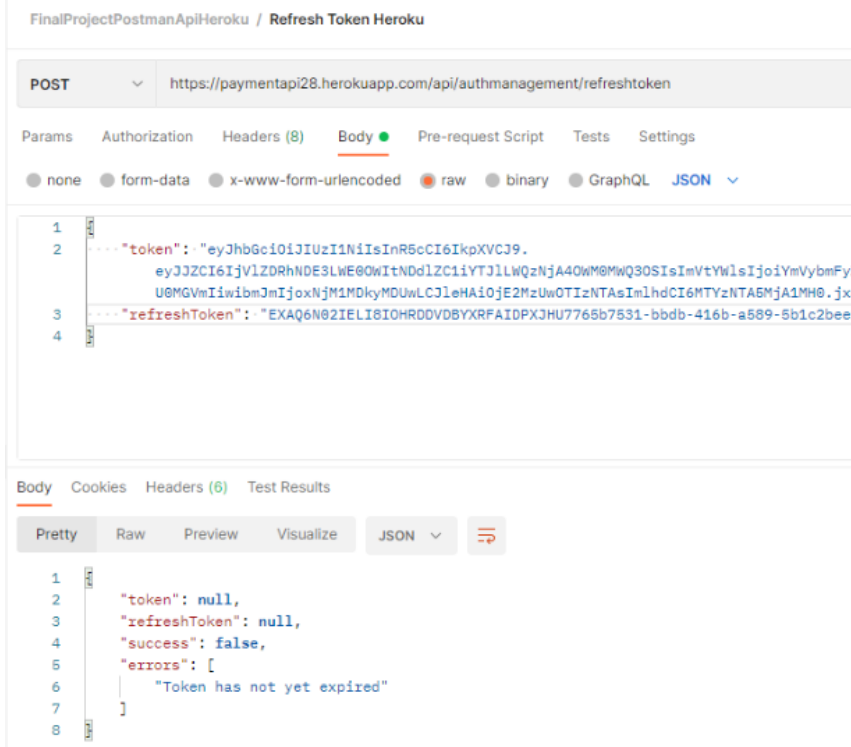
```
{  "username": "Bernard5",  "email": "bernard5@gmail.com",  "password": "Bernard%5"}
```

The response is also shown in JSON format:

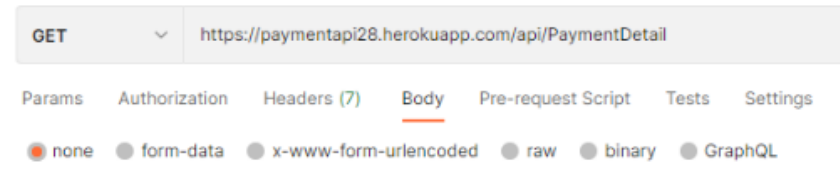
```
{  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJC2I6IjVlZDRhNDc3LWEOOWItNDdlZC1iYTJlLWQzNjA4OWMwMwQ3ODSiImVtYWlsIjoIYmVybmFyZDZVZ21haWwvY29tIiwic3VlZHMtZiIiwibmJmIjoxNjM1MDkyMDI4LCJleHAiOjE2MzUwOTIzMjgsImhhdCI6MTYzNTA5MjAyOH0uShLvryW1bQyAo-gWErfjJxZds",  "refreshToken": "IEW2QRZOW2N0Z5LXRTLXC6AJBLX202NKESU9e0b27c4-f760-4300-86b2-0fda6e31313c",  "success": true,  "errors": null}
```

The screenshot shows a REST client interface with a POST request to `https://paymentapi28.herokuapp.com/api/authmanagement/login`. The request body is in JSON format, containing `"email": "bernard5@gmail.com"` and `"password": "Bernard%5"`. The response body is also in JSON format, containing `"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJJ2ZCI6IjVlZDRhNDE3LWE0OWItNDdlZC1iYTJlLWQzNjA4OWM0MWQ3OSIsImVtYWlsIjoiYmVybU0MGVmIiwibmJmIjoxNjM1MDkyMDUwLCJleHAiOjE2MzUwOTIzNTAsIm1hdCI6MTYzNTA5MjA1MH0"`, `"refreshToken": "EXAQ6N02IELI8IOHRDDVDBYXRFAIDPXJHU7765b7631-bbdb-416b-a589-5b1c2"`, `"success": true`, and `"errors": null`.

### 3. RefreshToken (<https://paymentapi28.herokuapp.com/api/authmanagement/refreshToken>)



#### 4. GetDetails (<https://paymentapi28.herokuapp.com/api/paymentdetail>)



## 5. CreateDetail (<https://paymentapi28.herokuapp.com/api/paymentdetail>)

FinalProjectPostmanApiHeroku / CreateDetail Heroku


POST ▼ <https://paymentapi28.herokuapp.com/api/PaymentDetail>

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☒ JSON ▼

```
1  {
2    "cardOwnerName": "Testing",
3    "cardNumber": "19283",
4    "expirationDate": "05-05-2005",
5    "securityCode": "5555"
6  }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
1  {
2    "paymentDetailId": 4,
3    "cardOwnerName": "Testing",
4    "cardNumber": "19283",
5    "expirationDate": "05-05-2005",
6    "securityCode": "5555"
7  }
```

## 6. GetDetail (<https://paymentapi28.herokuapp.com/api/paymentdetail/{}>)


FinalProjectPostmanApiHeroku / GetDetail Heroku

GET ▼ <https://paymentapi28.herokuapp.com/api/PaymentDetail/4>

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
1  {
2    "paymentDetailId": 4,
3    "cardOwnerName": "Testing",
4    "cardNumber": "19283",
5    "expirationDate": "05-05-2005",
6    "securityCode": "5555"
7  }
```

## 7. UpdateDetail (<https://paymentapi28.herokuapp.com/api/paymentdetail/4>)

The screenshot shows a REST client interface with a PUT request to `https://paymentapi28.herokuapp.com/api/PaymentDetail/4`. The 'Body' tab is selected, and the request body is a JSON object. The response body is also shown in the 'Body' tab, indicating a successful update.

```
PUT https://paymentapi28.herokuapp.com/api/PaymentDetail/4
```

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "paymentDetailId": 4,
3   "cardOwnerName": "Hacktiv8",
4   "cardNumber": "88888888",
5   "expirationDate": "08-08-2008",
6   "securityCode": "87654"
7 }
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": [
3     "Data has been updated"
4   ],
5   "result": {
6     "paymentDetailId": 4,
7     "cardOwnerName": "Hacktiv8",
8     "cardNumber": "88888888",
9     "expirationDate": "08-08-2008",
10    "securityCode": "87654"
11  },
12  "success": true
13 }
```

## 8. DeleteDetail (<https://paymentapi28.herokuapp.com/api/paymentdetail/4>)

The screenshot shows a REST client interface with a DELETE request to `https://paymentapi28.herokuapp.com/api/PaymentDetail/4`. The 'Body' tab is selected, and the response body is shown, indicating a successful deletion.

```
DELETE https://paymentapi28.herokuapp.com/api/PaymentDetail/4
```

Params Authorization Headers (7) **Body** Pre-request Script Tests S

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": [
3     "Data has been deleted"
4   ],
5   "result": null,
6   "success": false
7 }
```