

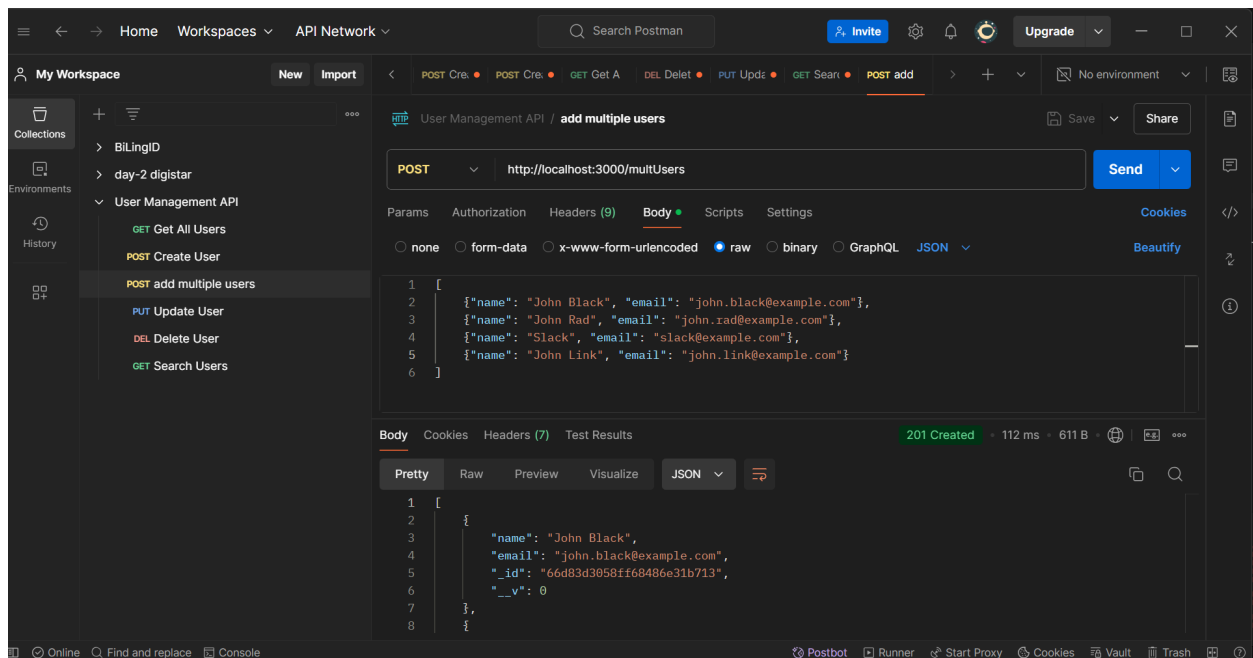
Nama: Dhafa Nur Fadhillah

Kelas: Hacker Back-end 2

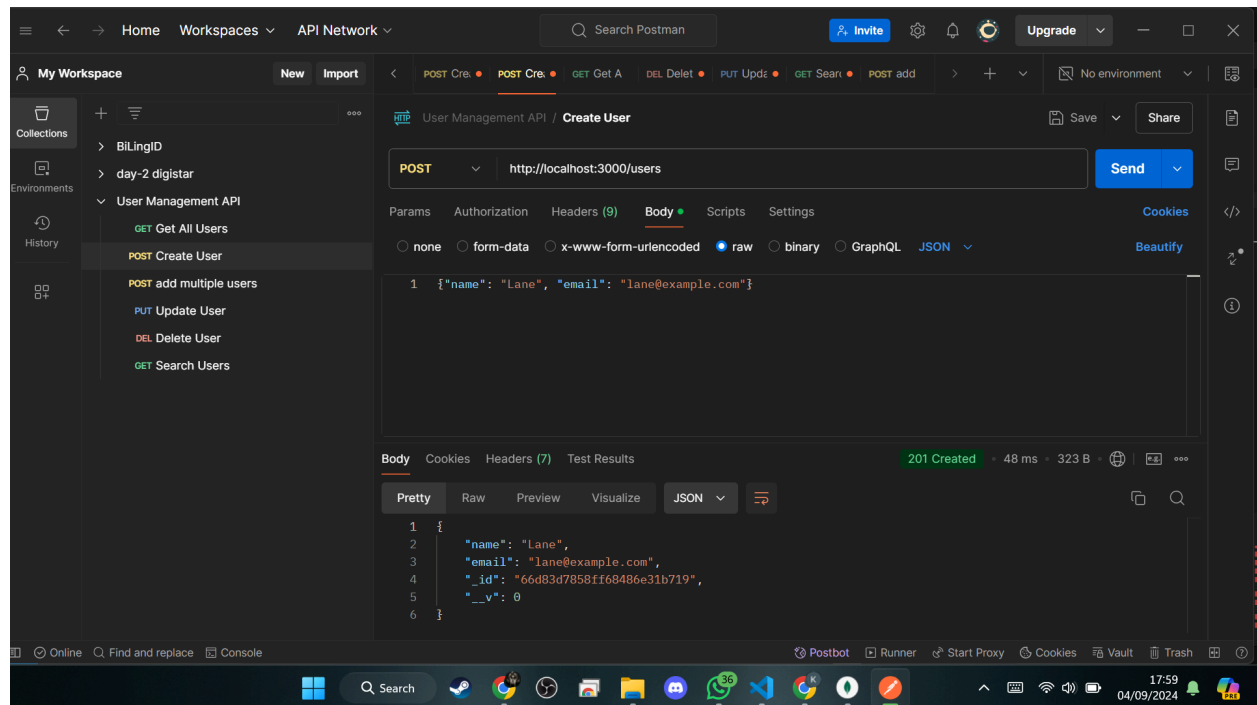
Run server.js

```
PS C:\Users\User\Downloads\nodejs\day-4> node server.js
Server running on port 3000
MongoDB connected successfully
```

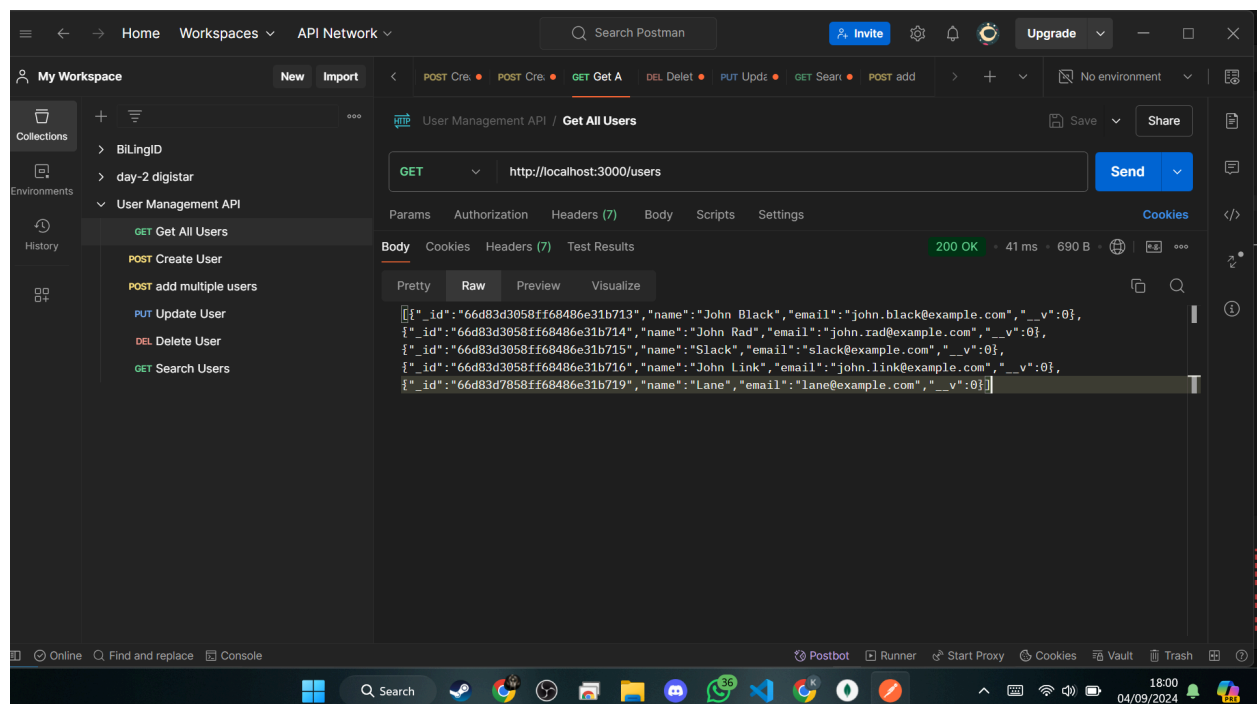
Add multiple users



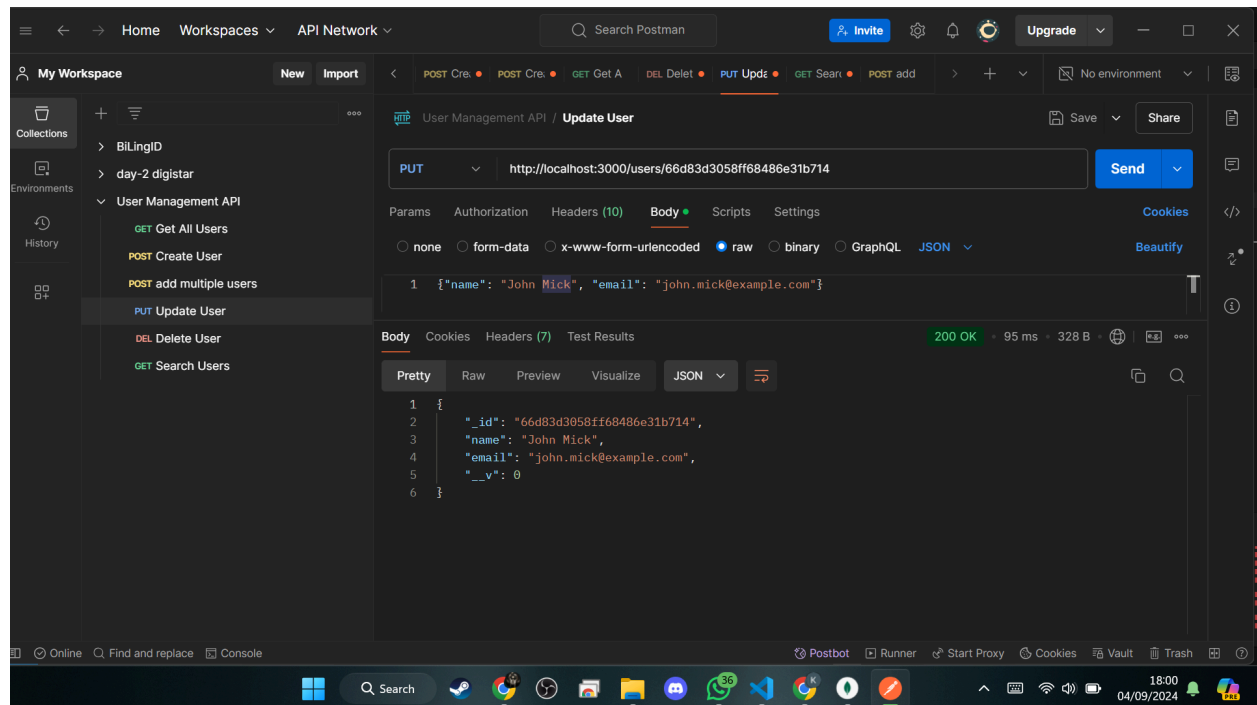
Tes tambah user 1



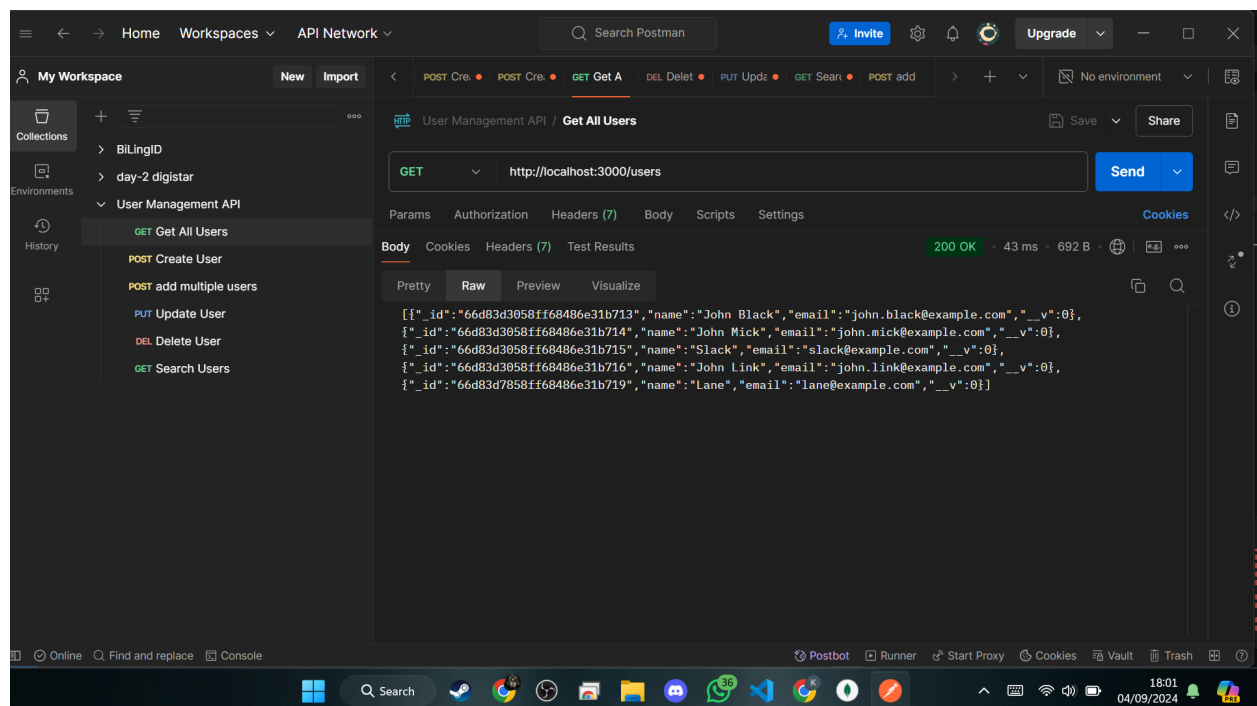
Hasil tambah user



Update user



hasil



Delete user

A screenshot of the Postman application interface. The top bar shows 'Home', 'Workspaces', and 'API Network'. The left sidebar has 'My Workspace' selected. The main panel shows a 'DELETE' request to 'http://localhost:3000/users/66d83d3058ff68486e31b714'. The 'Send' button is highlighted. Below the request, the 'Body' tab is selected, showing a JSON response:

```
{ 1: { 2: "message": "User with id 66d83d3058ff68486e31b714 deleted successfully" 3: }
```

hasil

A screenshot of the Postman application interface. The top bar shows 'Home', 'Workspaces', and 'API Network'. The left sidebar has 'My Workspace' selected. The main panel shows a 'GET' request to 'http://localhost:3000/users'. The 'Send' button is highlighted. Below the request, the 'Body' tab is selected, showing a JSON response:

```
[{"_id": "66d83d3058ff68486e31b713", "name": "John Black", "email": "john.black@example.com", "__v": 0}, {"_id": "66d83d3058ff68486e31b715", "name": "Slack", "email": "slack@example.com", "__v": 0}, {"_id": "66d83d3058ff68486e31b716", "name": "John Link", "email": "john.link@example.com", "__v": 0}, {"_id": "66d83d7858ff68486e31b719", "name": "Lane", "email": "lane@example.com", "__v": 0}]
```

Search user by name

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections including 'BilingID', 'day-2 digistar', and 'User Management API'. The 'User Management API' collection is expanded, showing endpoints like 'GET Get All Users', 'POST Create User', 'POST add multiple users', 'PUT Update User', 'DEL Delete User', and 'GET Search Users'. The 'GET Search Users' endpoint is selected. The main panel displays the request details for the 'GET' method with the URL 'http://localhost:3000/search?name=John'. The 'Params' tab is active, showing a single parameter 'name' with the value 'John'. The 'Body' tab is also visible. The response is shown in the 'Body' tab, indicating a '200 OK' status with a response time of 39 ms and a body size of 427 B. The response body is a JSON array of two user objects.

```
GET http://localhost:3000/search?name=John
```

Key	Value	Description
name	John	Description

```
[{"_id": "66d83d3958ff68486e31b713", "name": "John Black", "email": "john.black@example.com", "__v": 0}, {"_id": "66d83d3958ff68486e31b716", "name": "John Link", "email": "john.link@example.com", "__v": 0}]
```

The screenshot shows the Postman application interface with the same setup as the first image. The 'GET Search Users' endpoint is selected, and the URL is 'http://localhost:3000/search?name=Lane'. The 'Params' tab shows the parameter 'name' with the value 'Lane'. The 'Body' tab shows a '200 OK' status with a response time of 36 ms and a body size of 320 B. The response body is a JSON array of one user object.

```
GET http://localhost:3000/search?name=Lane
```

Key	Value	Description
name	Lane	Description

```
[{"_id": "66d83d7858ff68486e31b719", "name": "Lane", "email": "lane@example.com", "__v": 0}]
```