

# Instagram Backend API :- Documentation

Requirements:Go 1.10 or higher.

MongoDB 2.6 and higher.

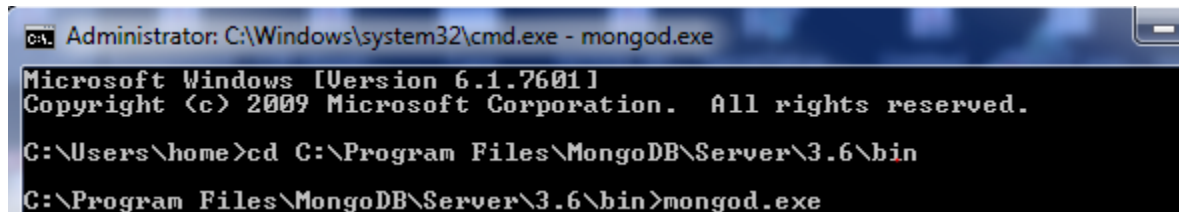
Operations to be performed:

- Create an User
  - Should be a POST request
  - Use JSON request body
  - URL should be '/users'
  - Fields are \_id,name,email,password.
- Get a user using id
  - Should be a GET request
  - Id should be in the url parameter
  - URL should be '/users/<id here>'
- Create a Post
  - Should be a POST request
  - Use JSON request body
  - URL should be '/posts'
  - Fields are \_id,caption,image\_url,postedAt,user\_id
- Get a post using id
  - Should be a GET request
  - Id should be in the url parameter
  - URL should be '/posts/<id here>'
- List all posts of a user
  - Should be a GET request
  - URL should be '/post/users/<Id here>'

Execution:

Stating mongod locally:

1.open command prompt and execute "**mongod.exe**" in the directory where it is present.as shown below:



```
Administrator: C:\Windows\system32\cmd.exe - mongod.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\home>cd C:\Program Files\MongoDB\Server\3.6\bin
C:\Program Files\MongoDB\Server\3.6\bin>mongod.exe
```

2.open a new command prompt and execute “**mongo.exe**” in the directory where it is present.as shown below:

```
Administrator: C:\Windows\system32\cmd.exe - mongo.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\home>cd C:\Program Files\MongoDB\Server\3.6\bin
C:\Program Files\MongoDB\Server\3.6\bin>mongo.exe
```

Running the api file:

- 1.execute “**go build**”. In the directory where the main.go is present.
- 2.execute “**go run ./**”.In the directory where the main.go is present.

As shown below:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

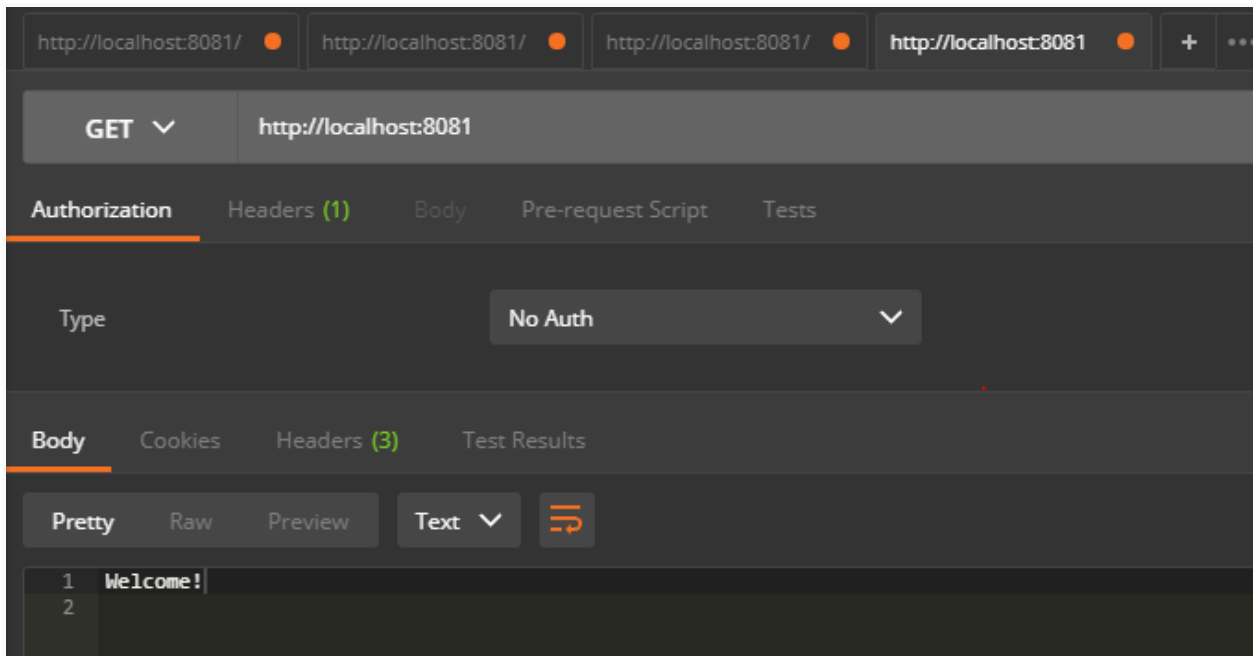
C:\Users\home\Desktop\Instagram-Backend-API>go build

C:\Users\home\Desktop\Instagram-Backend-API>go run ./
```

To test the api i am using postman:

- 1.for url:<http://localhost:8081>

Output:



2.for url:http://localhost:8081/users/

Output:

The screenshot displays a REST client interface with a tab for `http://localhost:8081/`. The selected method is **POST** for the endpoint `http://localhost:8081/users`. The **Body** tab is active, showing the request body as a JSON object:

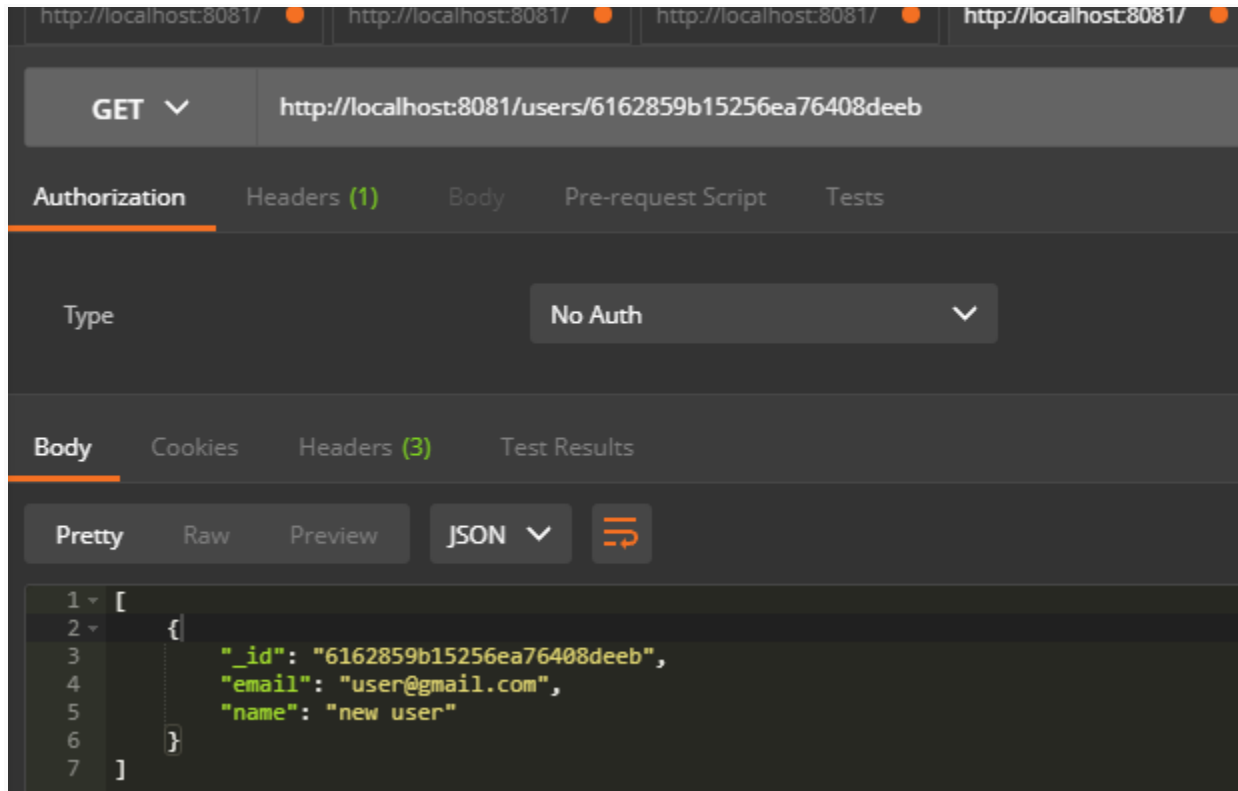
```
1 {  
2   "name": "new user",  
3   "email": "user@gmail.com",  
4   "password": "user123"  
5 }  
6
```

Below the request body, the **Body** tab of the response is active, showing the response body as a JSON object:

```
1 {  
2   "InsertedID": "6162859b15256ea76408deeb"  
3 }
```

3..for url:http://localhost:8081/users/<\_id>

output:



Note: password will not be displayed.

Output with password to show how password is saved in database:

The screenshot shows a web browser with a REST client interface. The address bar displays `http://localhost:8081/`. The request method is `GET` and the URL is `http://localhost:8081/users/6162859b15256ea76408deeb`. The `Authorization` tab is selected, showing `No Auth`. The `Body` tab is also selected, showing the response in `JSON` format. The response body is a JSON array containing one object:

```
[
  {
    "_id": "6162859b15256ea76408deeb",
    "email": "user@gmail.com",
    "name": "new user",
    "password": "$2a$10$a9CfhPbWL44nJDScM/4s..ASCB rJUkk174/VM0v7XRfMDppCu1gSG"
  }
]
```

4..for url:http://localhost:8081/posts

Output:

The screenshot displays a REST client interface with a dark theme. At the top, there are four tabs for different URLs, all pointing to `http://localhost:8081/`. The main area is titled `POST http://localhost:8081/posts`. Below this, there are tabs for `Authorization`, `Headers (1)`, `Body` (which is selected), `Pre-request Script`, and `Tests`. Under the `Body` tab, there are radio buttons for `form-data`, `x-www-form-urlencoded`, `raw` (selected), and `binary`, followed by a dropdown menu set to `JSON (application/json)`. The request body is a JSON object: 

```
{  "userid": "6162859b15256ea76408deeb",  "caption": "post1 by new user",  "url": "http://img.com/1"}
```

. Below the request body, there are tabs for `Body` (selected), `Cookies`, `Headers (3)`, and `Test Results`. Under the `Body` tab, there are buttons for `Pretty`, `Raw`, and `Preview`, followed by a dropdown menu set to `JSON` and a button with a JSON icon. The response body is a JSON object: 

```
{  "InsertedID": "616288a8aec800f101ffa74d"}
```

5..for url:http://localhost:8081/posts/<\_id>

Output:

The screenshot shows a web browser's developer tools interface. At the top, there are four tabs for the URL `http://localhost:8081/`. Below this, the **GET** method is selected, and the URL `http://localhost:8081/posts/616288a8aec800f101ffa74d` is entered. The **Authorization** tab is active, showing a dropdown menu set to **No Auth**. Below this, the **Body** tab is active, showing the response body in **JSON** format. The response is a JSON array with one object:

```
1 [
2   {
3     "_id": "616288a8aec800f101ffa74d",
4     "caption": "post1 by new user",
5     "image_url": "http://img.com/1",
6     "postedat": "2021-10-10T12:01:04.137+05:30",
7     "user_id": "6162859b15256ea76408deeb"
8   }
9 ]
```

6..for url:http://localhost:8081/post/users/<user\_id>

Output:

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/post/users/6162859b15256ea76408deeb`. The response is displayed in JSON format, showing an array of two post objects.

```
[
  {
    "_id": "616288a8aec800f101ffa74d",
    "caption": "post1 by new user",
    "image_url": "http://img.com/1",
    "postedat": "2021-10-10T12:01:04.137+05:30",
    "user_id": "6162859b15256ea76408deeb"
  },
  {
    "_id": "61628933aec800f101ffa750",
    "caption": "post2 by new user",
    "image_url": "http://img.com/2",
    "postedat": "2021-10-10T12:03:23.667+05:30",
    "user_id": "6162859b15256ea76408deeb"
  }
]
```