



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

OCTOBER 2021-22

**APPOINTY INTERNSHIP TECHNICAL
TASK**

INSTAGRAM BACKEND API

SUBMITTED BY:

NAME: Shashwat Jha

REG ID: 19BCE0996.

GITHUB: github.com/shashwatjha018

HOW TO RUN THIS API

- I) This API is programmed using GOLANG and MongoDB is used as a database.
- II) API-Testing has been done with the help of POSTMAN.
- III) First install GOLANG on your local machine and download the Go Package from Visual Studio Code extensions and install all the necessary packages.
- IV) Then download the code from repository and save it anywhere on your local machine.
- V) Enter the directory using “CD” command in your VS Code terminal
- VI) Set up init file inside the directory by typing “go mod init github.com/<your github username>insta-api-appointy” in the VS code terminal.
- VII) Download the GO MONGO-DB package by typing “go get go.mongodb.org/mongo-driver/mongo”
- VIII) Now download and install the MongoDB Compass on your local machine and start its default server.
- IX) Now type “go run main.go” in the VS Code terminal to run the API
- X) Test it using POSTMAN by entering it’s URL.

CODE REPOSITORY:

You can find the full source code of this API here:

<https://github.com/shashwatjha018/insta-backend-api-19BCE0996-SHASHWAT-JHA>

SOME OF THE REQUIRED MODULES CODES OTHER THAN THE ENDPOINTS:

Passwords should be securely stored such they can't be reverse engineered (PASSWORD HASHING)

```
//Password Hashing
data := []byte(user.Password)
b := md5.Sum(data)
user.Password = hex.EncodeToString(b[:])
```

PAGINATION:

```
func Pagation(r *http.Request, limit int) (int, int) {
    keys := r.URL.Query()
    if keys.Get("page") == ""{
        return 1, 0
    }
    page, _ := strconv.Atoi(keys.Get("page"))
    if page < 1{
        return 1, 0
    }
    begin := (limit * page) - limit
    return page, begin
}
```

Make the server thread safe

This API uses the concept of concurrency to make the server thread safe by implementing Goroutines. A Goroutine is a [function](#) or method which executes independently and simultaneously in connection with any other Goroutines present in your program

```
func handleRequests(){
    go http.HandleFunc("/", HomePage)
    go http.HandleFunc("/users", CreateUser)
    go http.HandleFunc("/userlist", UserList)
    go http.HandleFunc("/users/", GetUser)
    go http.HandleFunc("/posts", CreatePost)
    go http.HandleFunc("/posts/", GetPost)
    go http.HandleFunc("/posts/users/", GetUserPost)
    go log.Fatal(http.ListenAndServe(":8081", nil))
}
```

UNIT TESTS FOR ENDPOINTS:

Following are some testcases for each function implemented

OPERATION 1:

AIM: Create a user
METHOD: POST request
URL: http://localhost:8081/user

INPUT (using JSON request body):

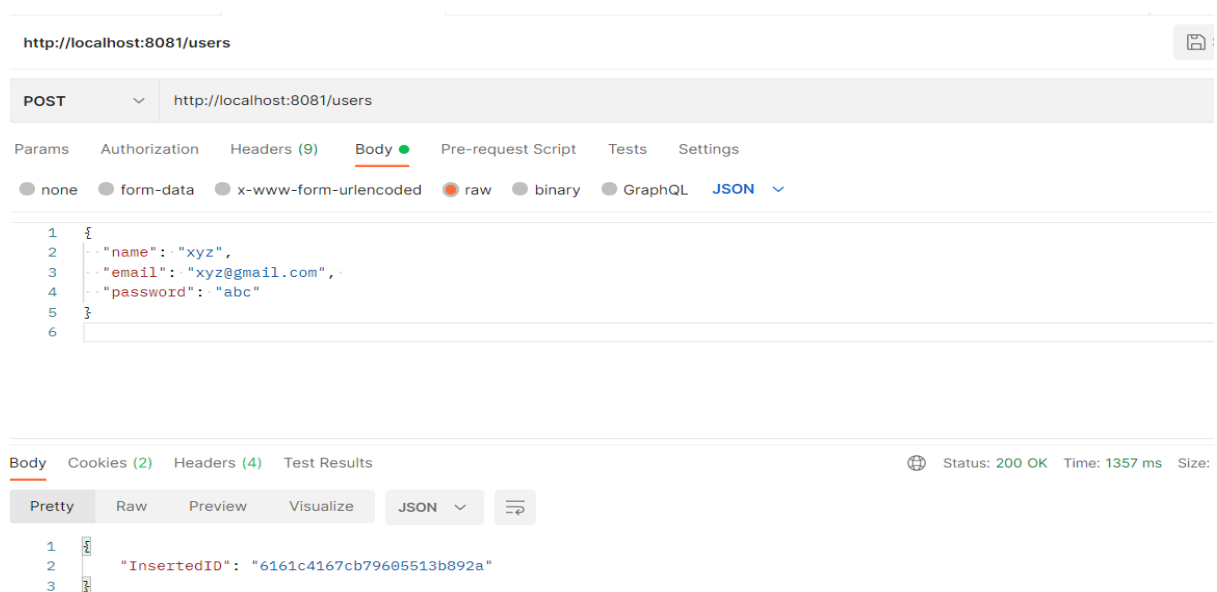
```
{  
  "name": "xyz",  
  "email": "xyz@gmail.com",  
  "password": "abc"  
}
```

EXPECTED OUTPUT:

- Message showing User inserted.
- Generate User Id
- Save user details in MongoDB database.

ACTUAL OUTPUT (Screenshot):

POSTMAN



MONGODB:

```
_id: ObjectId("6161a949f3e6f4b4fcf97c9d")
name: "Shashwat Dummy Test 1"
email: "dummy1@gmail.com"
password: "46f05eee7da43518f8d131b7f753a658"
```

```
_id: ObjectId("6161a96ff3e6f4b4fcf97c9e")
name: "Shashwat Dummy Test 2"
email: "dummy2@gmail.com"
password: "67addf0388110031d86e0a2f39d66618"
```

```
_id: ObjectId("6161a97bf3e6f4b4fcf97c9f")
name: "Shashwat Dummy Test 3"
email: "dummy3@gmail.com"
password: "8a50542f41d8e1b9132f6a484a1b15d7"
```

```
_id: ObjectId("6161aa7df3e6f4b4fcf97ca3")
name: "New Shashwat Dummy Test "
email: "newdummy@gmail.com"
password: "ee20e35ae49290df22441c91b345f4e5"
```

```
_id: ObjectId("6161c4167cb79605513b892a")
name: "xyz"
email: "xyz@gmail.com"
password: "900150983cd24fb0d6963f7d28e17f72"
```

Operation completed successfully.

OPERATION 2:

AIM: Get a user using id

METHOD: GET Request

INPUT URL: <http://localhost:8081/users/6161c4167cb79605513b892a>

EXPECTED OUTPUT:

- Display user details
- Password should be hashed.

```
{
  "_id": "6161c4167cb79605513b892a",
  "name": "xyz",
  "email": "xyz@gmail.com",
  "password": "900150983cd24fb0d6963f7d28e17f72"
}
```

ACTUAL OUTPUT (Screenshot):

The screenshot displays a REST client interface with the following details:

- URL:** <http://localhost:8081/users/6161c4167cb79605513b892a>
- Method:** GET
- Response Status:** 200 OK
- Response Time:** 1358 ms
- Response Size:** 227 B
- Response Format:** JSON (Pretty)
- Response Body:**

```
{
  "_id": "6161c4167cb79605513b892a",
  "name": "xyz",
  "email": "xyz@gmail.com",
  "password": "900150983cd24fb0d6963f7d28e17f72"
}
```

REMARKS:

Operation completed successfully.

OPERATION 3:

AIM: Create a post

METHOD: POST Request

URL: http://localhost:8081/posts

INPUT (using JSON request body):

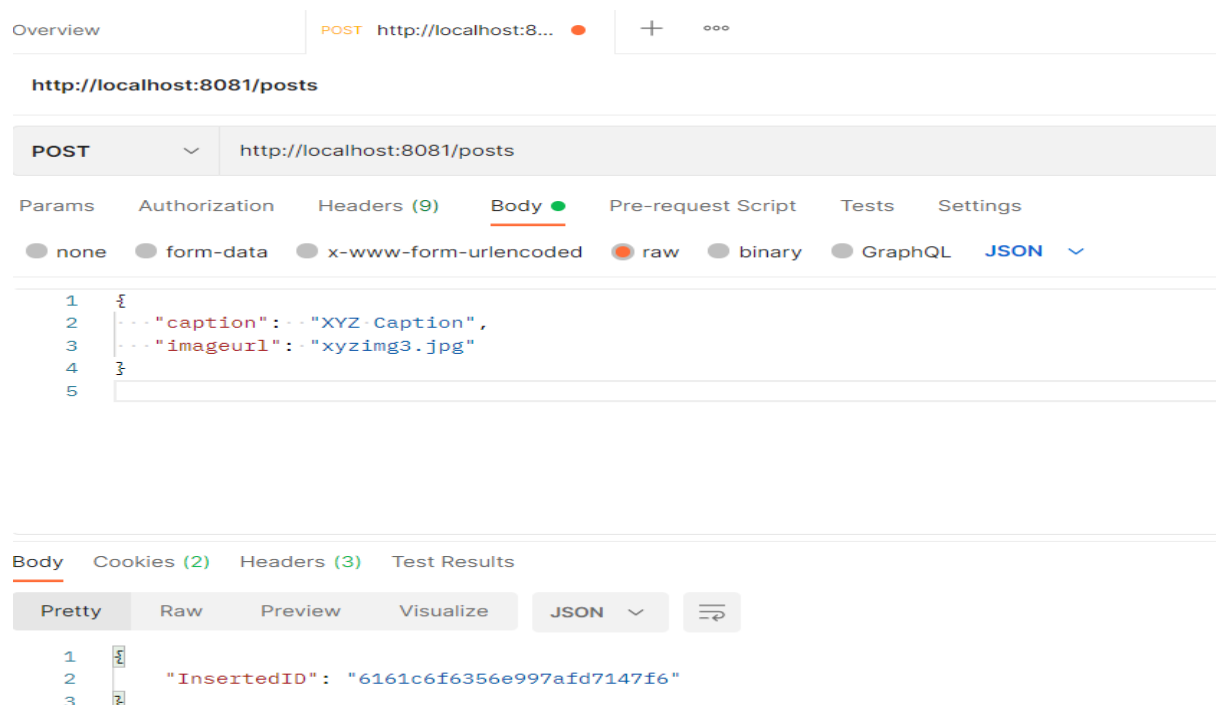
```
{  
  "caption": "XYZ Caption",  
  "imageurl": "xyzimg3.jpg"  
}
```

EXPECTED OUTPUT:

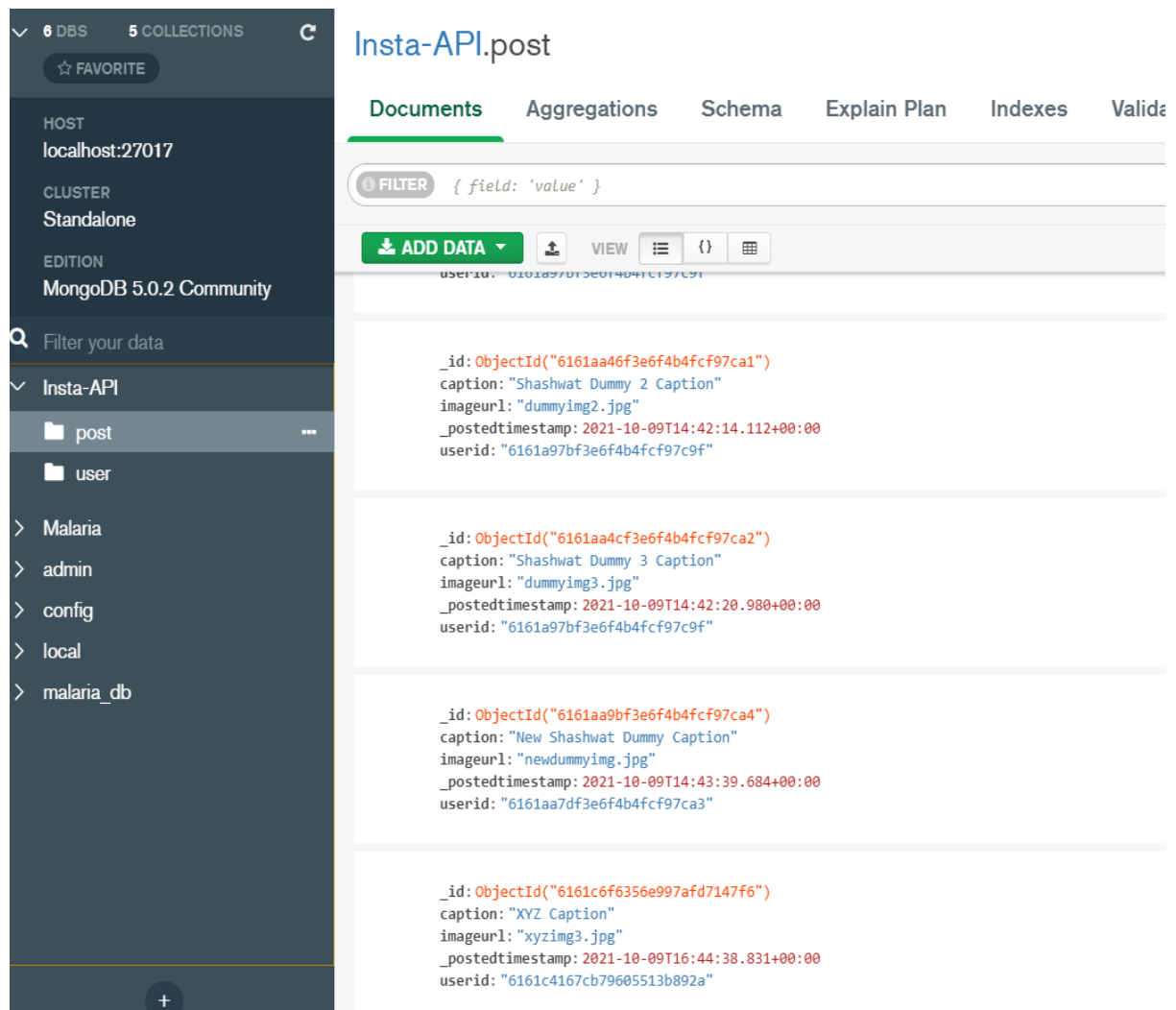
- Message showing post inserted.
- Generate Post Id
- Store Posted Timestamp
- The post belongs to the user who logged in previously.
- Save post details in MongoDB database.

ACTUAL OUTPUT (Screenshot):

POSTMAN



MONGODB



The screenshot shows the MongoDB Compass interface. On the left sidebar, the database 'Insta-API' is selected, and the 'post' collection is highlighted. The main area displays the 'Documents' tab for the 'post' collection. A filter bar at the top shows a filter: `{ field: 'value' }`. Below the filter bar, there are buttons for 'ADD DATA', 'VIEW', and a grid icon. The documents are listed as follows:

Document 1	Document 2	Document 3	Document 4
<pre>{ "_id": ObjectId("6161aa46f3e6f4b4fcf97ca1"), "caption": "Shashwat Dummy 2 Caption", "imageUrl": "dummyimg2.jpg", "_postedtimestamp": 2021-10-09T14:42:14.112+00:00, "userid": "6161a97bf3e6f4b4fcf97c9f" }</pre>	<pre>{ "_id": ObjectId("6161aa4cf3e6f4b4fcf97ca2"), "caption": "Shashwat Dummy 3 Caption", "imageUrl": "dummyimg3.jpg", "_postedtimestamp": 2021-10-09T14:42:20.980+00:00, "userid": "6161a97bf3e6f4b4fcf97c9f" }</pre>	<pre>{ "_id": ObjectId("6161aa9bf3e6f4b4fcf97ca4"), "caption": "New Shashwat Dummy Caption", "imageUrl": "newdummyimg.jpg", "_postedtimestamp": 2021-10-09T14:43:39.684+00:00, "userid": "6161aa7df3e6f4b4fcf97ca3" }</pre>	<pre>{ "_id": ObjectId("6161c6f6356e997afd7147f6"), "caption": "XYZ Caption", "imageUrl": "xyzimg3.jpg", "_postedtimestamp": 2021-10-09T16:44:38.831+00:00, "userid": "6161c4167cb79605513b892a" }</pre>

REMARKS:

Operation completed successfully.

OPERATION 4:

AIM: Get a post using id

METHOD: GET request

INPUT URL: <http://localhost:8081/posts/6161c6f6356e997afd7147f6>

EXPECTED OUTPUT:

- Display post details
- Realtime Datetime stored automatically


```
{
  "_id": "6161c6f6356e997afd7147f6",
  "caption": "XYZ Caption",
  "imageurl": "xyzimg3.jpg",
  "_postedtimestamp": "2021-10-09T22:14:38.831+05:30",
  "userid": "6161c4167cb79605513b892a"
}
```

ACTUAL OUTPUT (Screenshot):

The screenshot displays a REST client interface with the following details:

- Overview** tab selected. URL: `http://localhost:8081/posts/6161c6f6356e997afd7147f6`
- Method**: GET
- Params** tab selected, showing **Query Params** with one entry:

KEY	VALUE
Key	Value
- Body** tab selected, showing the response in **JSON** format:

```
1 {
2   "_id": "6161c6f6356e997afd7147f6",
3   "caption": "XYZ Caption",
4   "imageurl": "xyzimg3.jpg",
5   "_postedtimestamp": "2021-10-09T22:14:38.831+05:30",
6   "userid": "6161c4167cb79605513b892a"
7 }
```

REMARKS:

Operation completed successfully.

OPERATION 5:

AIM: List all posts of a user (Pagination added)

METHOD: GET Request

INPUT URL: <http://localhost:8081/user/post/6161a97bf3e6f4b4fcf97c9f>

EXPECTED OUTPUT:

- Display user Id
- Display the number of posts displayed
- Display all the post details too.
- If the posts exceeds a certain limit(2), it should display remaining output on the next page.

PAGE 1:

6161a97bf3e6f4b4fcf97c9f

Displayed results: 2

```
[
  {
    "_id": "6161aa10f3e6f4b4fcf97ca0",
    "caption": "Shashwat Dummy 1 Caption",
    "imageurl": "dummyimg1.jpg",
    "_postedtimestamp": "2021-10-09T20:11:20.862+05:30",
    "userid": "6161a97bf3e6f4b4fcf97c9f"
  },
  {
    "_id": "6161aa46f3e6f4b4fcf97ca1",
    "caption": "Shashwat Dummy 2 Caption",
    "imageurl": "dummyimg2.jpg",
    "_postedtimestamp": "2021-10-09T20:12:14.112+05:30",
    "userid": "6161a97bf3e6f4b4fcf97c9f"
  }
]
```

PAGE 2:

6161a97bf3e6f4b4fcf97c9f

Displayed results: 1

```
[
```

```
{
  "_id": "6161aa4cf3e6f4b4fcf97ca2",
  "caption": "Shashwat Dummy 3 Caption",
  "imageurl": "dummyimg3.jpg",
  "_postedtimestamp": "2021-10-09T20:12:20.98+05:30",
  "userid": "6161a97bf3e6f4b4fcf97c9f"
}
```

ACTUAL OUTPUT (Screenshot):

I) Since we uploaded only 1 post from this user id, it will show only 1 post

http://localhost:8081/posts/users/6161c4167cb79605513b892a

GET http://localhost:8081/posts/users/6161c4167cb79605513b892a

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

Query Params

KEY	VALUE
Key	Value

Body Cookies (2) Headers (3) Test Results

Pretty Raw Preview Visualize JSON

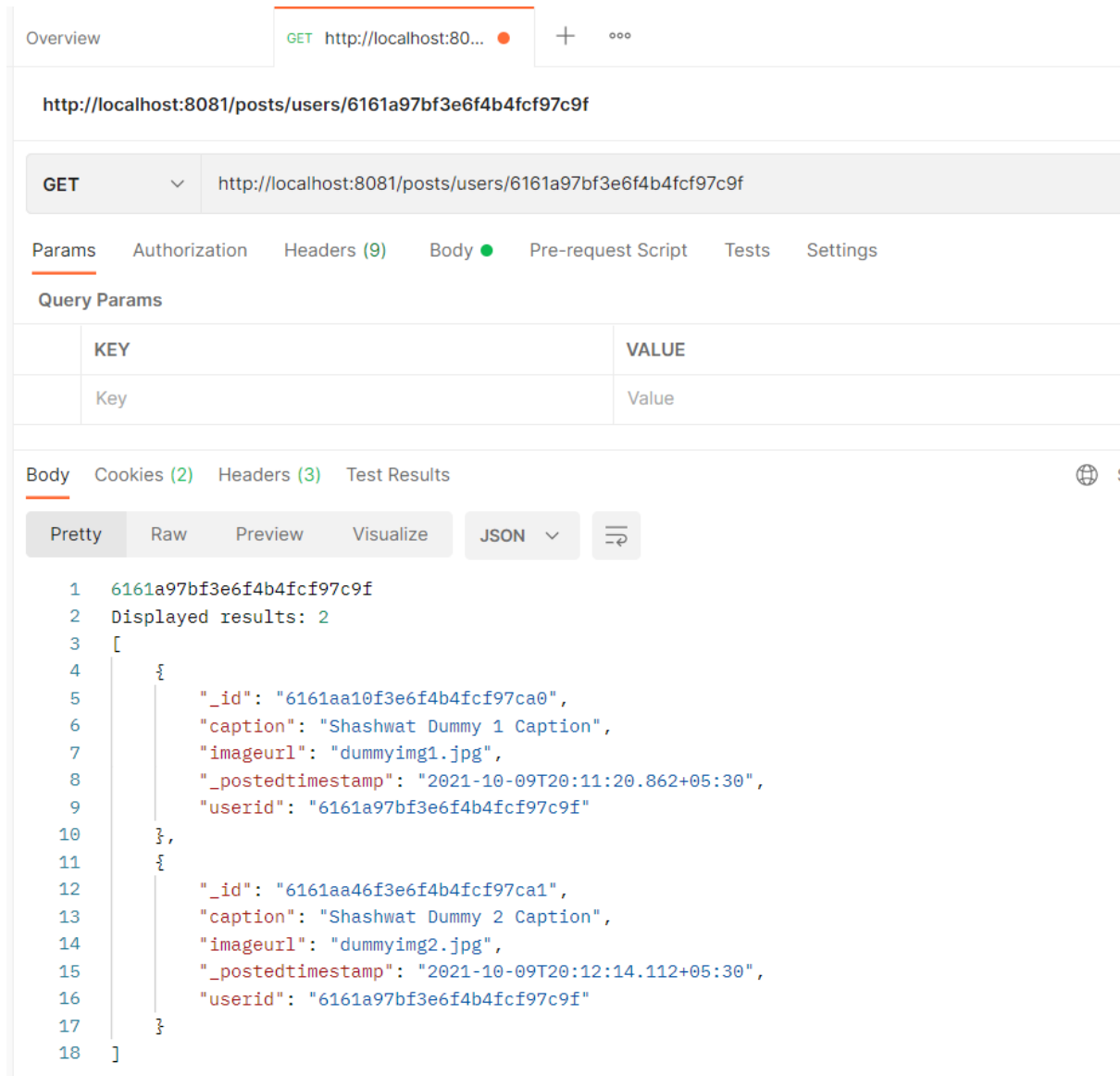
```

1 6161c4167cb79605513b892a
2 Displayed results: 1
3 [
4   {
5     "_id": "6161c6f6356e997afd7147f6",
6     "caption": "XYZ Caption",
7     "imageurl": "xyzimg3.jpg",
8     "_postedtimestamp": "2021-10-09T22:14:38.831+05:30",
9     "userid": "6161c4167cb79605513b892a"
10  }
11 ]

```

2) User Id “ 6161a97bf3e6f4b4fcf97c9f “ has posted 3 images. So first two post will be displayed on the first page and then we have to give the query parameter(?page=2) to see the remaining posts.

Page 1:



Overview GET http://localhost:80... + ...

http://localhost:8081/posts/users/6161a97bf3e6f4b4fcf97c9f

GET http://localhost:8081/posts/users/6161a97bf3e6f4b4fcf97c9f

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

Query Params

KEY	VALUE
Key	Value

Body Cookies (2) Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 6161a97bf3e6f4b4fcf97c9f
2 Displayed results: 2
3 [
4   {
5     "_id": "6161aa10f3e6f4b4fcf97ca0",
6     "caption": "Shashwat Dummy 1 Caption",
7     "imageurl": "dummyimg1.jpg",
8     "_postedtimestamp": "2021-10-09T20:11:20.862+05:30",
9     "userid": "6161a97bf3e6f4b4fcf97c9f"
10  },
11  {
12    "_id": "6161aa46f3e6f4b4fcf97ca1",
13    "caption": "Shashwat Dummy 2 Caption",
14    "imageurl": "dummyimg2.jpg",
15    "_postedtimestamp": "2021-10-09T20:12:14.112+05:30",
16    "userid": "6161a97bf3e6f4b4fcf97c9f"
17  }
18 ]
```

Page 2:

Overview GET http://localhost:80... + ...

http://localhost:8081/posts/users/6161a97bf3e6f4b4fcf97c9f?page=2

GET http://localhost:8081/posts/users/6161a97bf3e6f4b4fcf97c9f?page=2

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

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	page	2	
	Key	Value	Description

Body Cookies (2) Headers (3) Test Results Status: 200 OK Time: 101ms

Pretty Raw Preview Visualize JSON ↕

```
1 6161a97bf3e6f4b4fcf97c9f
2 Displayed results: 1
3 [
4   {
5     "_id": "6161aa4cf3e6f4b4fcf97ca2",
6     "caption": "Shashwat Dummy 3 Caption",
7     "imageurl": "dummyimg3.jpg",
8     "_postedtimestamp": "2021-10-09T20:12:20.98+05:30",
9     "userid": "6161a97bf3e6f4b4fcf97c9f"
10  }
11 ]
```

REMARKS:

Operation completed successfully.

OPERATION 6: (Additional operation I added)

AIM:	List all users
METHOD:	GET Request
INPUT URL:	http://localhost:8081/userlist

EXPECTED OUTPUT:

- Display all users stored in the database
- Display all the post details too.
- Password should be hashed.

```
[
{
  "_id": "6161a949f3e6f4b4fcf97c9d",
  "name": "Shashwat Dummy Test 1",
  "email": "dummy1@gmail.com",
  "password": "46f05eee7da43518f8d131b7f753a658"
},
{
  "_id": "6161a96ff3e6f4b4fcf97c9e",
  "name": "Shashwat Dummy Test 2",
  "email": "dummy2@gmail.com",
  "password": "67addf0388110031d86e0a2f39d66618"
},
{
  "_id": "6161a97bf3e6f4b4fcf97c9f",
  "name": "Shashwat Dummy Test 3",
  "email": "dummy3@gmail.com",
  "password": "8a50542f41d8e1b9132f6a484a1b15d7"
},
{
  "_id": "6161aa7df3e6f4b4fcf97ca3",
  "name": "New Shashwat Dummy Test ",
  "email": "newdummy@gmail.com",
  "password": "ee20e35ae49290df22441c91b345f4e5"
},
{

```

```
    "_id": "6161c4167cb79605513b892a",
    "name": "xyz",
    "email": "xyz@gmail.com",
    "password": "900150983cd24fb0d6963f7d28e17f72"
  }
]
```

ACTUAL OUTPUT (Screenshot):

The screenshot displays a REST client interface. At the top, the URL bar shows 'http://localhost:8081/userlist' with a status of 'GET' and a red circle icon. Below the URL bar, the 'Body' tab is selected, showing a JSON array of four user objects. The JSON is formatted in a 'Pretty' view. The first three objects are dummy users with IDs starting with '6161a', and the fourth object is the user 'xyz' with ID '6161c4167cb79605513b892a'.

```
GET http://localhost:8081/userlist

[
  {
    "_id": "6161a96ff3e6f4b4fcf97c9e",
    "name": "Shashwat Dummy Test 2",
    "email": "dummy2@gmail.com",
    "password": "67addf0388110031d86e0a2f39d66618"
  },
  {
    "_id": "6161a97bf3e6f4b4fcf97c9f",
    "name": "Shashwat Dummy Test 3",
    "email": "dummy3@gmail.com",
    "password": "8a50542f41d8e1b9132f6a484a1b15d7"
  },
  {
    "_id": "6161aa7df3e6f4b4fcf97ca3",
    "name": "New Shashwat Dummy Test ",
    "email": "newdummy@gmail.com",
    "password": "ee20e35ae49290df22441c91b345f4e5"
  },
  {
    "_id": "6161c4167cb79605513b892a",
    "name": "xyz",
    "email": "xyz@gmail.com",
    "password": "900150983cd24fb0d6963f7d28e17f72"
  }
]
```

REMARKS:

Operation completed successfully.

DATABASE SCREENSHOTS:

Insta-API.post

DOCUMENTS 5 TOTAL SI
744

[Documents](#) [Aggregations](#) [Schema](#) [Explain Plan](#) [Indexes](#) [Validation](#)

FILTER { field: 'value' }

ADD DATA

VIEW



{ }

Displayi

```
_id: ObjectId("6161aa10f3e6f4b4fc97ca0")
caption: "Shashwat Dummy 1 Caption"
imageurl: "dummyimg1.jpg"
_postedtimestamp: 2021-10-09T14:41:20.862+00:00
userid: "6161a97bf3e6f4b4fc97c9f"
```

```
_id: ObjectId("6161aa46f3e6f4b4fc97ca1")
caption: "Shashwat Dummy 2 Caption"
imageurl: "dummyimg2.jpg"
_postedtimestamp: 2021-10-09T14:42:14.112+00:00
userid: "6161a97bf3e6f4b4fc97c9f"
```

```
_id: ObjectId("6161aa4cf3e6f4b4fc97ca2")
caption: "Shashwat Dummy 3 Caption"
imageurl: "dummyimg3.jpg"
_postedtimestamp: 2021-10-09T14:42:20.980+00:00
userid: "6161a97bf3e6f4b4fc97c9f"
```

```
_id: ObjectId("6161aa9bf3e6f4b4fc97ca4")
caption: "New Shashwat Dummy Caption"
imageurl: "newdummyimg.jpg"
_postedtimestamp: 2021-10-09T14:43:39.684+00:00
userid: "6161aa7df3e6f4b4fc97ca3"
```

```
_id: ObjectId("6161c6f6356e997afd7147f6")
```


Insta-API.user

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

ADD DATA  **VIEW**   

```
_id: ObjectId("6161a949f3e6f4b4fcf97c9d")
name: "Shashwat Dummy Test 1"
email: "dummy1@gmail.com"
password: "46f05eee7da43518f8d131b7f753a658"
```

```
_id: ObjectId("6161a96ff3e6f4b4fcf97c9e")
name: "Shashwat Dummy Test 2"
email: "dummy2@gmail.com"
password: "67addf0388110031d86e0a2f39d66618"
```

```
_id: ObjectId("6161a97bf3e6f4b4fcf97c9f")
name: "Shashwat Dummy Test 3"
email: "dummy3@gmail.com"
password: "8a50542f41d8e1b9132f6a484a1b15d7"
```

```
_id: ObjectId("6161aa7df3e6f4b4fcf97ca3")
name: "New Shashwat Dummy Test "
email: "newdummy@gmail.com"
password: "ee20e35ae49290df22441c91b345f4e5"
```

```
_id: ObjectId("6161c4167cb79605513b892a")
name: "xyz"
email: "xyz@gmail.com"
password: "900150983cd24fb0d6963f7d28e17f72"
```