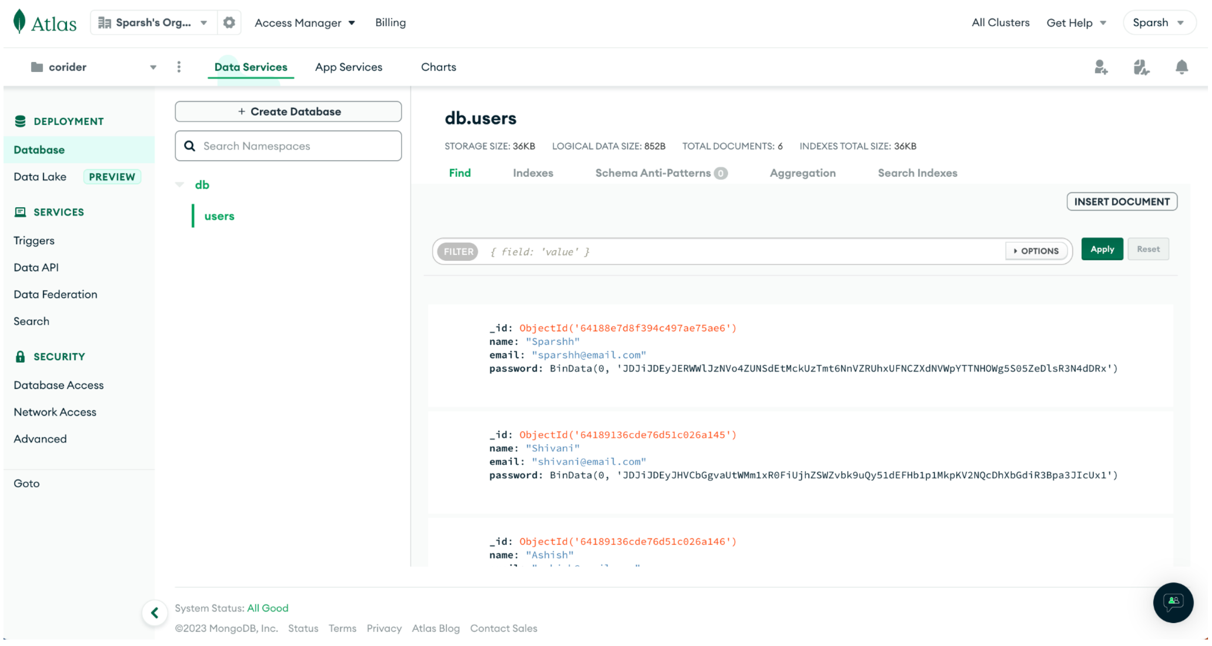
TESTING for APIs using POSTMAN

DATABASE: MongoDB



db.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 852B TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB

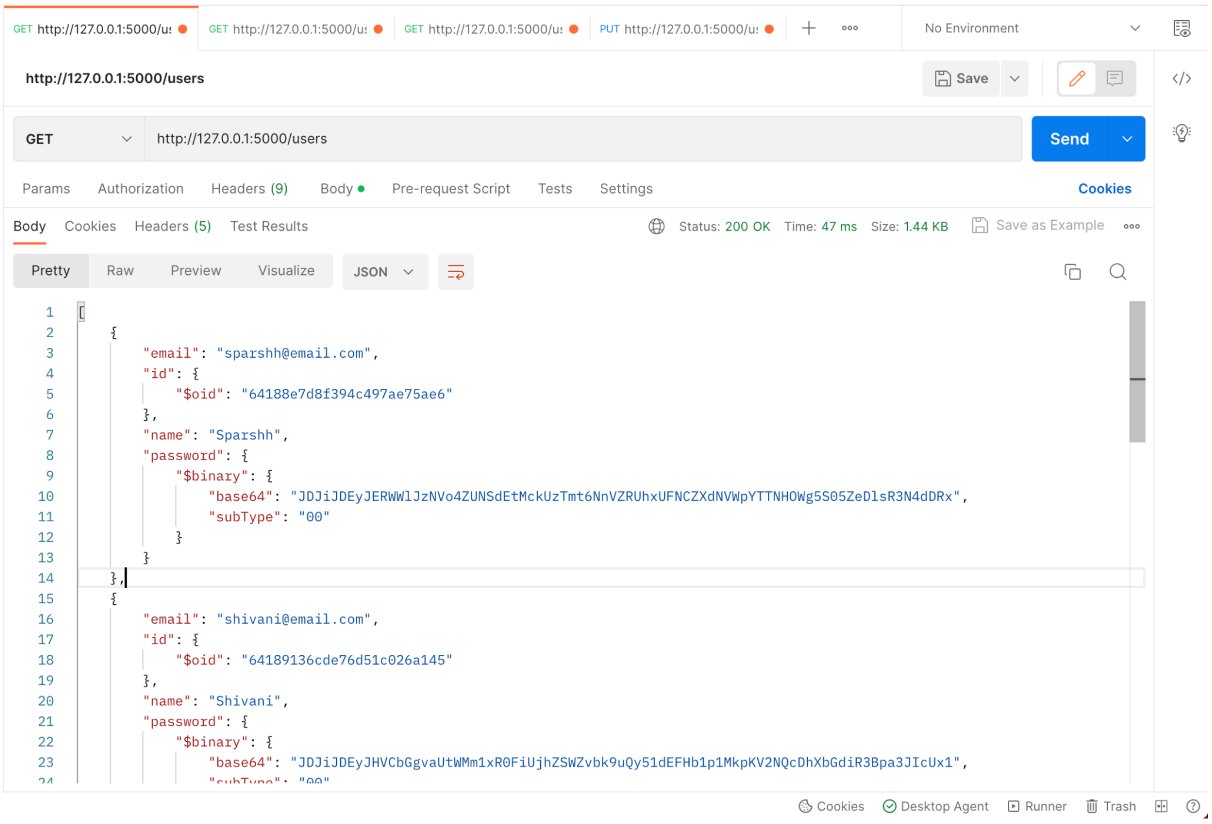
```
{ field: 'value' }
```

```
_id: ObjectId('64188e7d8f394c497ae75ae6')
name: "Sparshh"
email: "sparshh@email.com"
password: BinData(0, 'JDJiJDEyJERWWlJzNVo4ZUNSdEtMckUzTmt6NnVZRUhxUFNCZXdNVWpYTTNHOWg5S05ZeDlsR3N4dDRx')

_id: ObjectId('64189136cde76d51c026a145')
name: "Shivani"
email: "shivani@email.com"
password: BinData(0, 'JDJiJDEyJHVCbGgvaUtWMm1xR0FiUjhZSWZvbk9uQy51dEFHb1p1MkpKV2NQcDhXbGdiR3Bpa3JIcUx1')

_id: ObjectId('64189136cde76d51c026a146')
name: "Ashish"
```

1. GET /users - Returns a list of all users.



http://127.0.0.1:5000/users

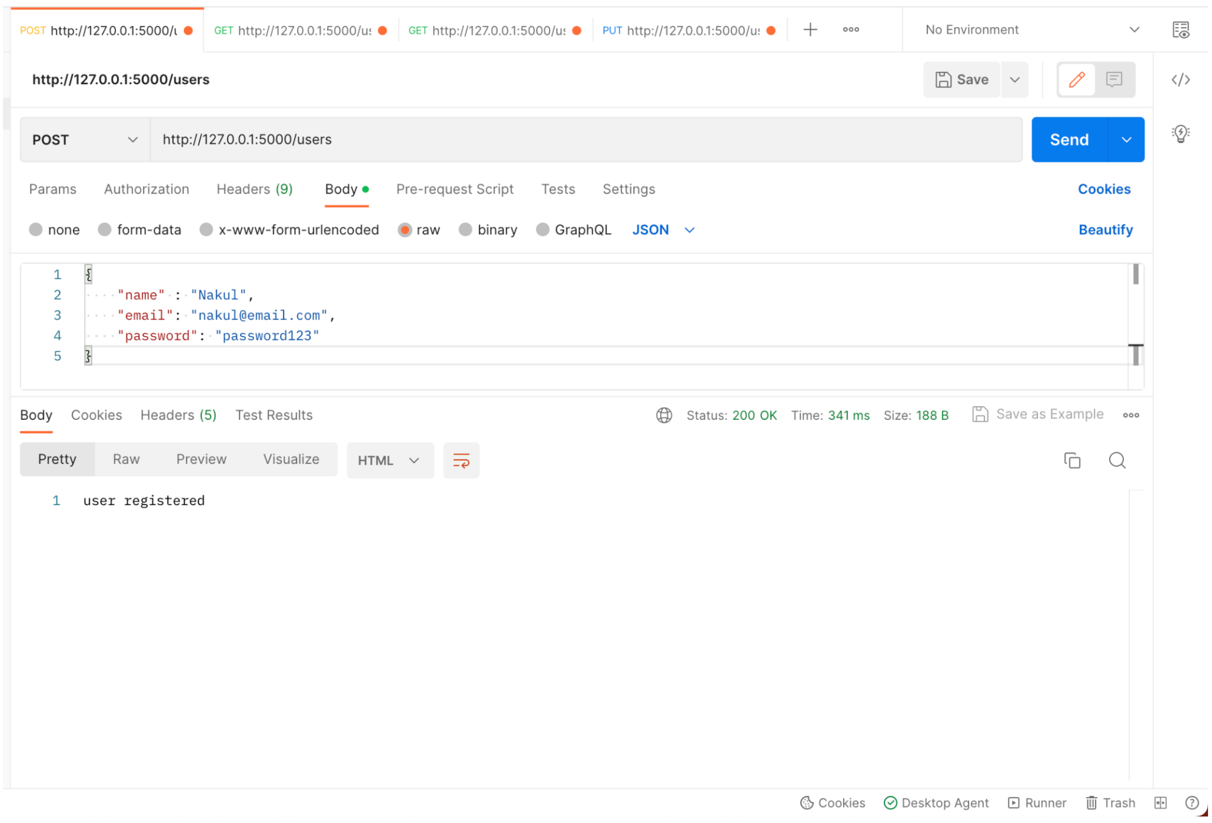
GET http://127.0.0.1:5000/users

Status: 200 OK Time: 47 ms Size: 1.44 KB

```
[
    {
        "email": "sparshh@email.com",
        "id": {
            "$oid": "64188e7d8f394c497ae75ae6"
        },
        "name": "Sparshh",
        "password": {
            "$binary": {
                "base64": "JDJiJDEyJERWWlJzNVo4ZUNSdEtMckUzTmt6NnVZRUhxUFNCZXdNVWpYTTNHOWg5S05ZeDlsR3N4dDRx",
                "subType": "00"
            }
        }
    },
    {
        "email": "shivani@email.com",
        "id": {
            "$oid": "64189136cde76d51c026a145"
        },
        "name": "Shivani",
        "password": {
            "$binary": {
                "base64": "JDJiJDEyJHVCbGgvaUtWMm1xR0FiUjhZSWZvbk9uQy51dEFHb1p1MkpKV2NQcDhXbGdiR3Bpa3JIcUx1",
                "subType": "00"
```

<Returns all the records from the users collection>

2. POST /users - Creates a new user with the specified data

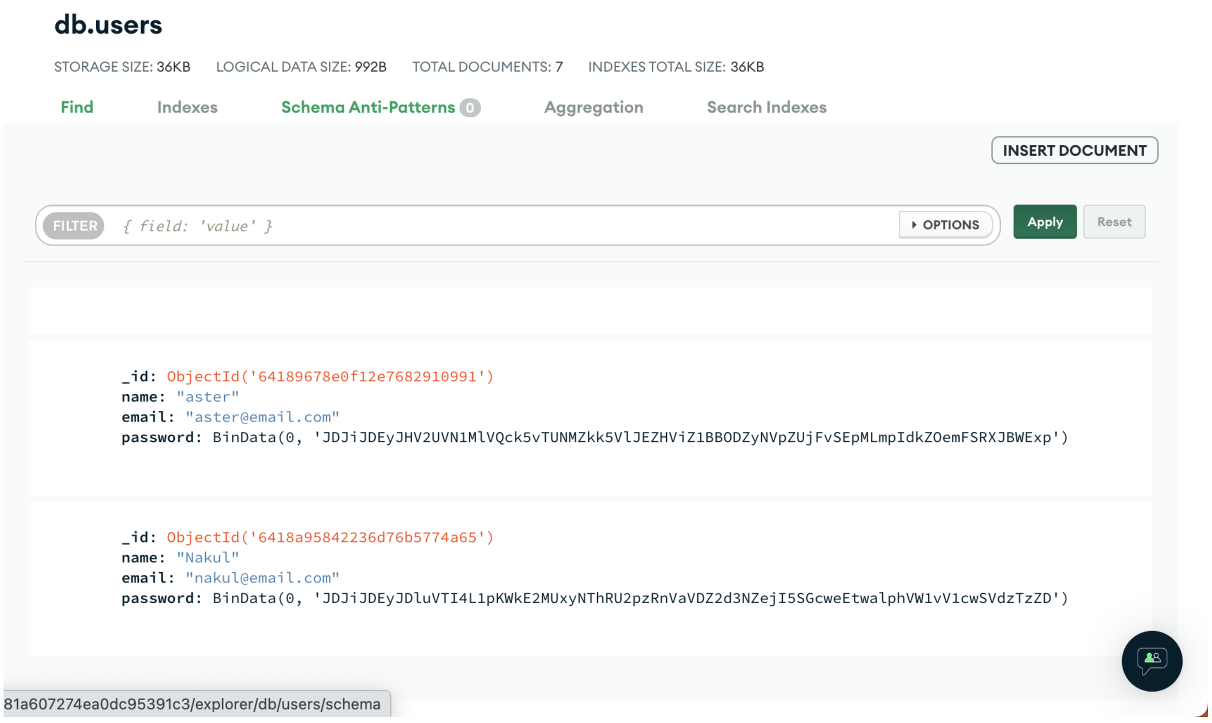


POST http://127.0.0.1:5000/users

```
{
    "name" : "Nakul",
    "email": "nakul@email.com",
    "password": "password123"
}
```

Status: 200 OK Time: 341 ms Size: 188 B

```
user registered
```



db.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 992B TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 36KB

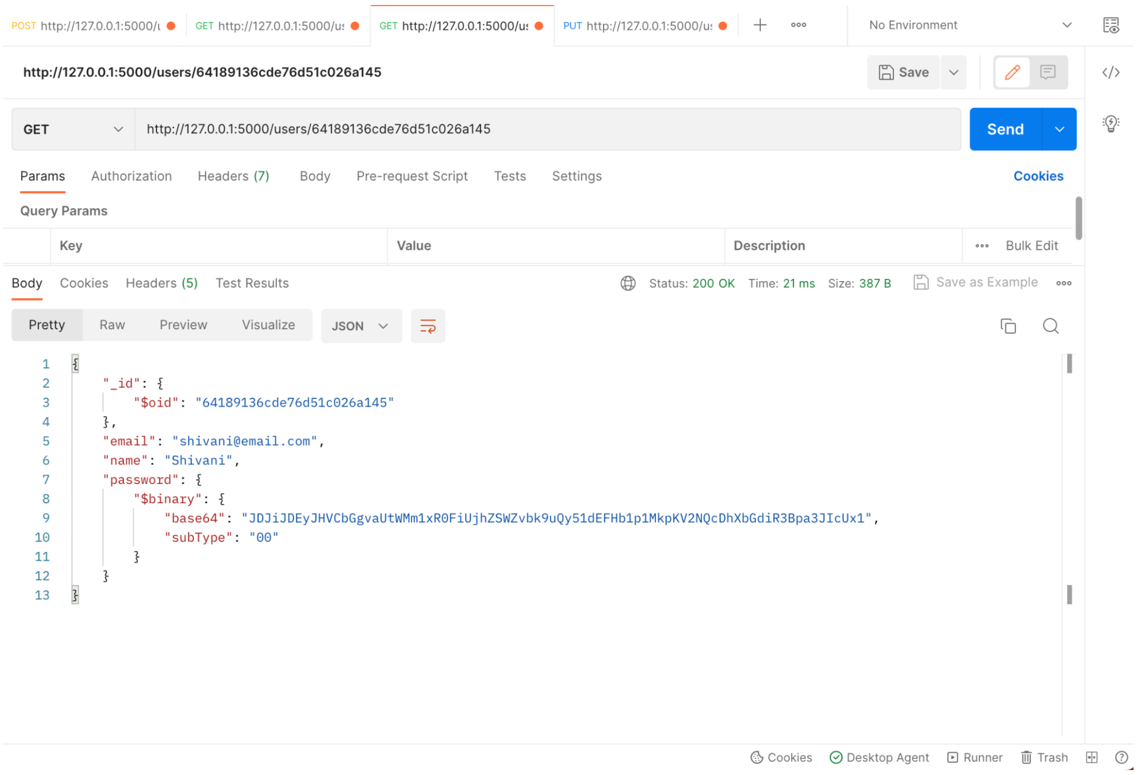
```
_id: ObjectId('64189678e0f12e7682910991')
name: "aster"
email: "aster@email.com"
password: BinData(0, 'JDJiJDEyJHV2UVN1MlVQck5vTUNMZkk5VlJEZHViZ1BBODZyNVpZUjFvSEpMLmpIdkZOemFSRXJBWExp')

_id: ObjectId('6418a95842236d76b5774a65')
name: "Nakul"
email: "nakul@email.com"
password: BinData(0, 'JDJiJDEyJDluVTI4L1pKWkE2MUxyNThRU2pzRnVaVDZ2d3NZejI5SGcweEtwalphVW1vV1cwSVdzTzZD')
```

81a607274ea0dc95391c3/explorer/db/users/schema

<Adds record to the users collection>

3. GET /users/<id> - Returns the user with the specified ID.



http://127.0.0.1:5000/users/64189136cde76d51c026a145

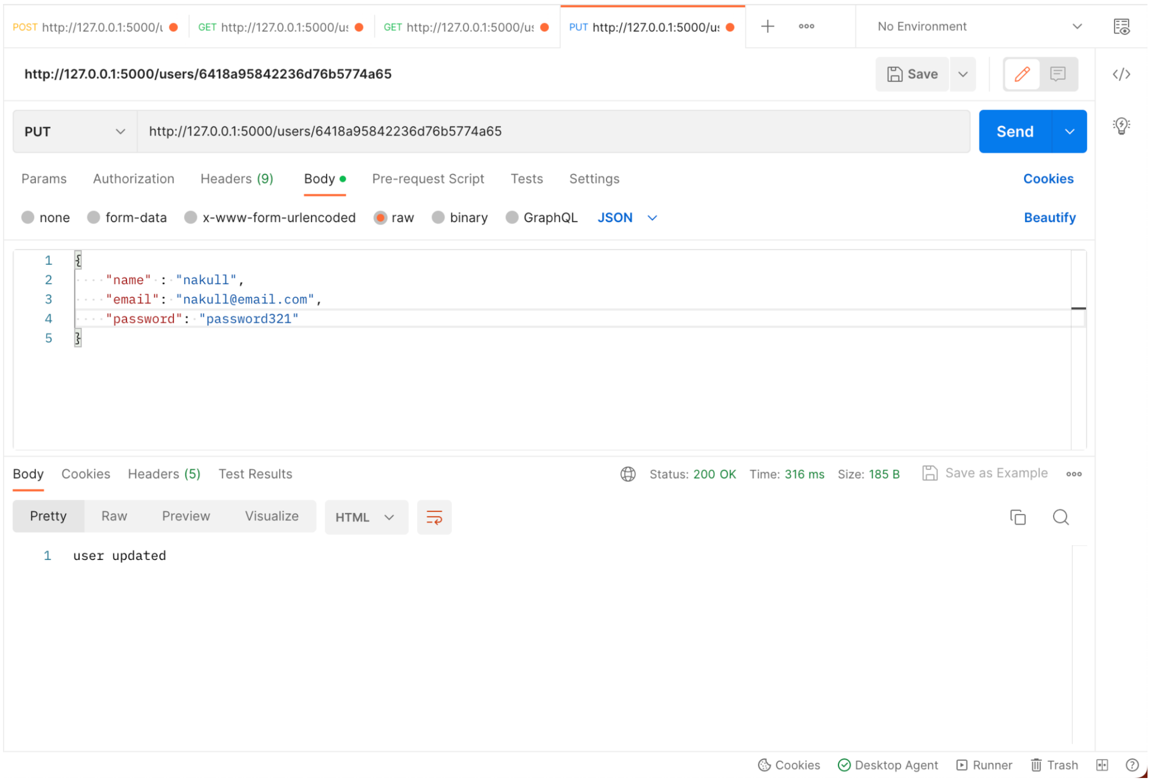
GET http://127.0.0.1:5000/users/64189136cde76d51c026a145

Status: 200 OK Time: 21 ms Size: 387 B

```
{
    "_id": {
        "$oid": "64189136cde76d51c026a145"
    },
    "email": "shivani@email.com",
    "name": "Shivani",
    "password": {
        "$binary": {
            "base64": "JDJiJDEyJHVCbGgvaUtWMm1xR0FiUjhZSWZvbk9uQy51dEFHb1p1MkpKV2NQcDhXbGdiR3Bpa3JIcUx1",
            "subType": "00"
        }
    }
}
```

<Returns record on passing id in flask route through GET request>

4. PUT /users/<id> - Updates the user with the specified ID with the new data.



http://127.0.0.1:5000/users/6418a95842236d76b5774a65

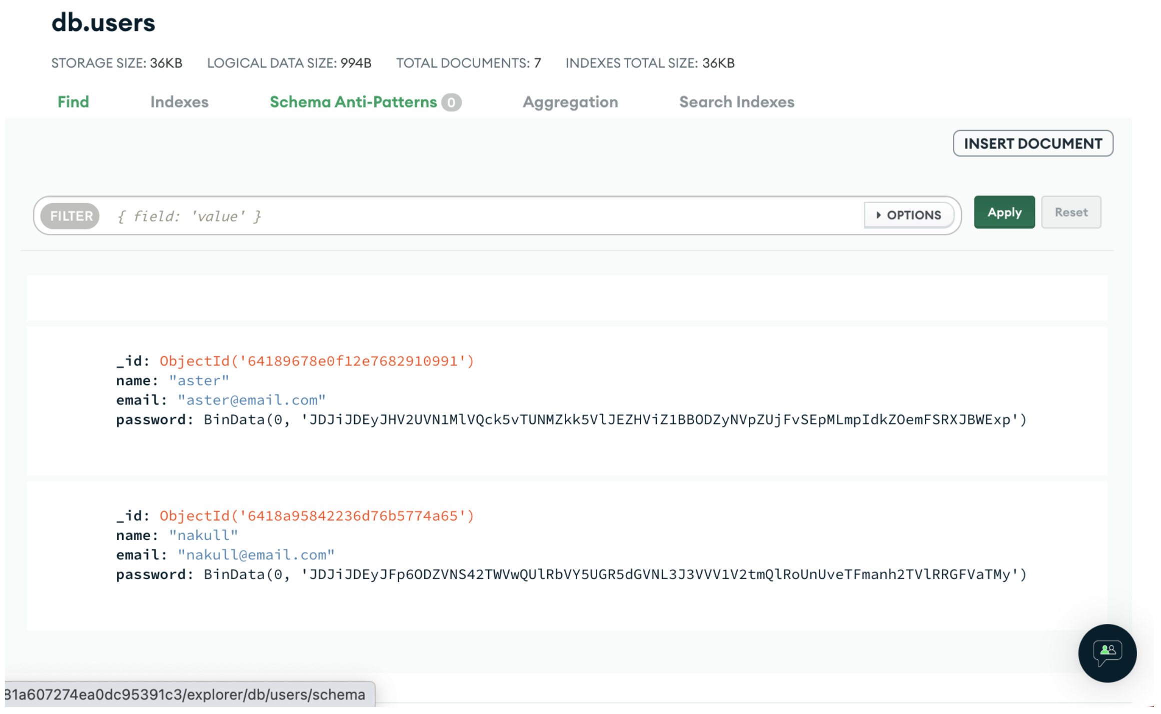
PUT http://127.0.0.1:5000/users/6418a95842236d76b5774a65

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "name" : "nakull",
    "email": "nakull@email.com",
    "password": "password321"
}
```

Status: 200 OK Time: 316 ms Size: 185 B

```
user updated
```

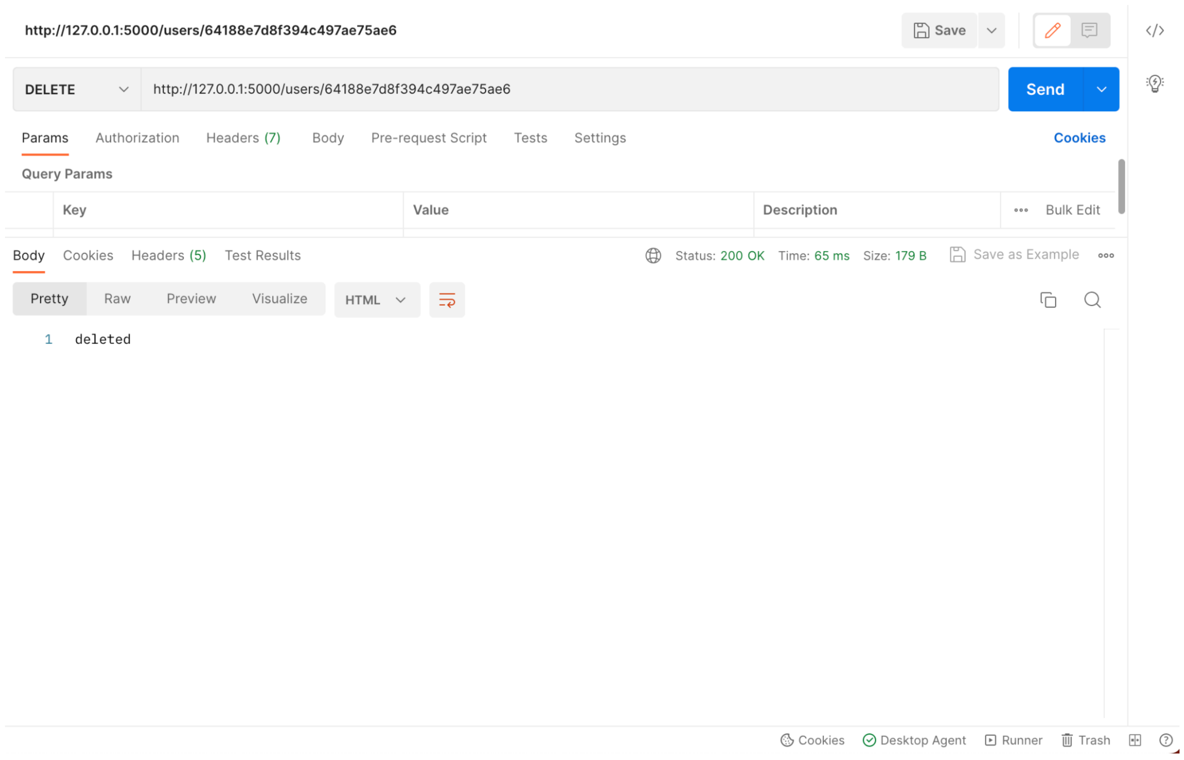

db.users
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 994B TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER
{ field: 'value' }
OPTIONS
Apply
Reset
31a607274ea0dc95391c3/explorer/db/users/schema

```
_id: ObjectId('64189678e0f12e7682910991')
name: "aster"
email: "aster@email.com"
password: BinData(0, 'JDJiJDEyJHV2UVN1MlVQck5vTUNMZkk5VlJEZHViZ1BBODZyNVpZUjFvSEpMLmpIdkZOemFSRXJBWExp')

_id: ObjectId('6418a95842236d76b5774a65')
name: "nakull"
email: "nakull@email.com"
password: BinData(0, 'JDJiJDEyJFp6ODZVNS42TWVwQUlRbVY5UGR5dGVNL3J3VVV1V2tmQlRoUnUveTFmanh2TVlRRGFVaTMy')
```

<Using data of user we registered earlier. User record updated and changes reflected in mongoDB>

5. DELETE /users/<id> - Deletes the user with the specified ID.



```
deleted
```

<Deleted record from collection by passing id in flask route through DELETE request>