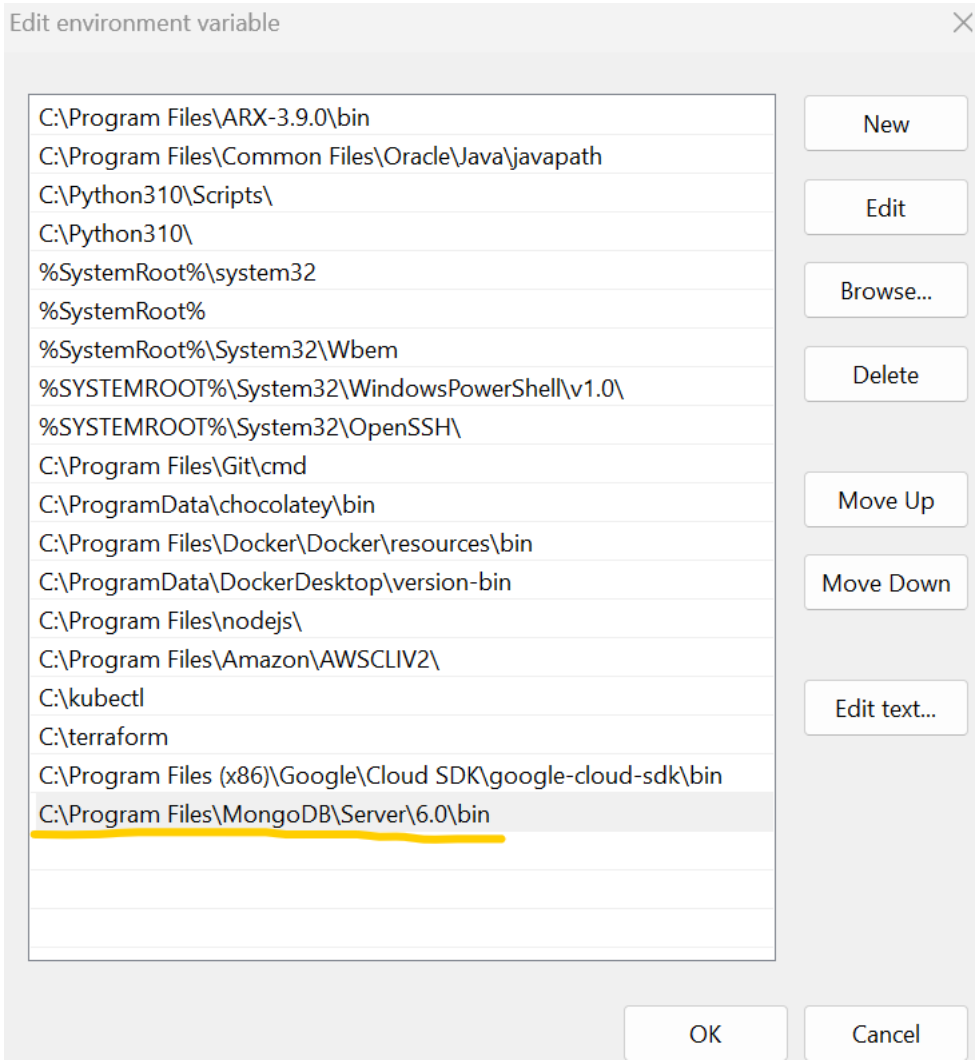
9.1P: Adding a database to your application

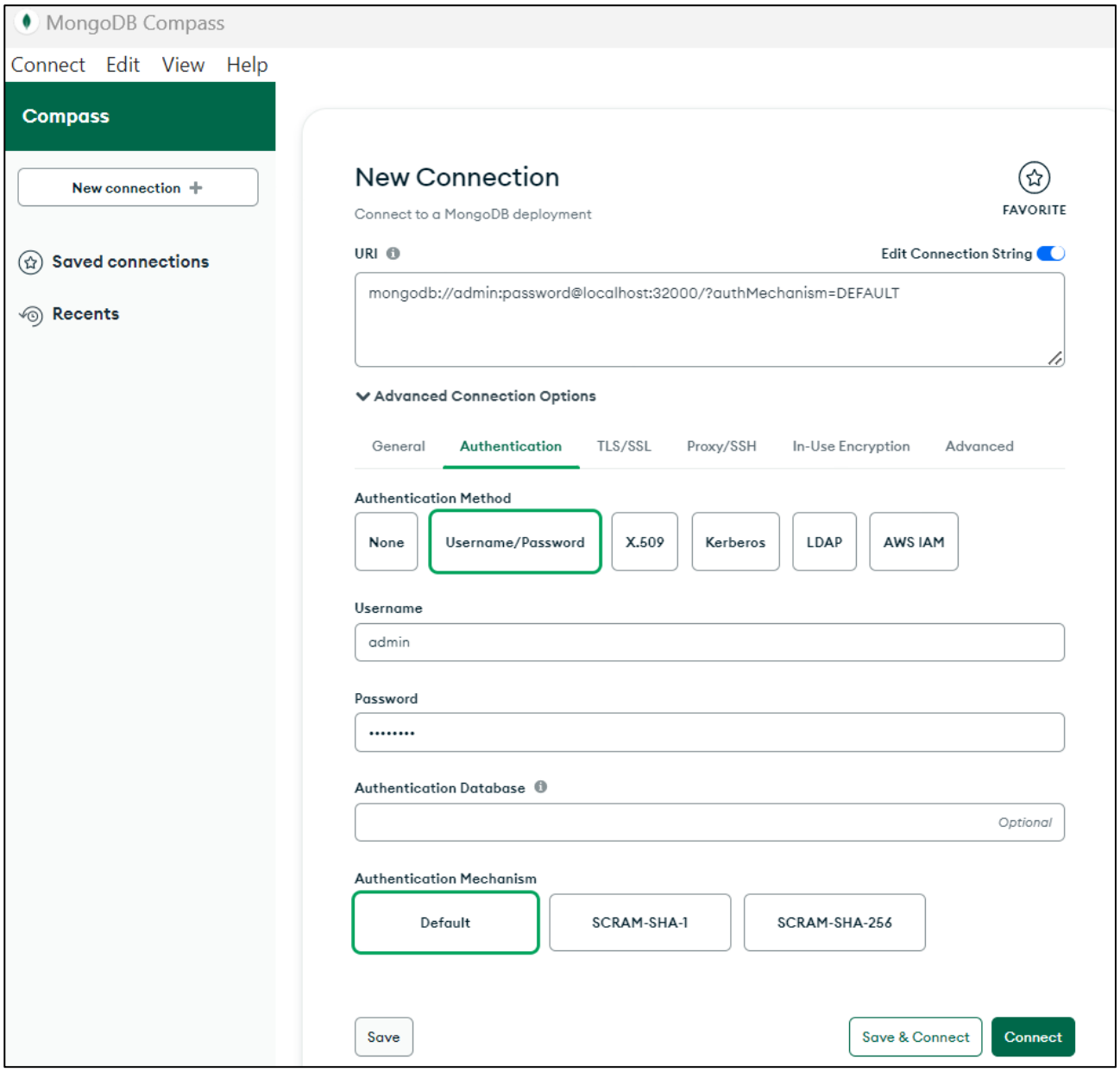
1. Installed MongoDB

- Downloaded version 6.0 of Mongo db and installed the complete version.
- Set the environment variable path to add mongo db's bin folder location



2. Create a MongoDB user with appropriate permissions for your application.

- Created new connection in MongoDB Compass by using New Connection > Authentication > username as admin and password as password



3. Configurations and commands –

- Configured **persistent** storage for the MongoDB database by creating a Persistent Volume and Persistent Volume Claim.
- Ran the command **kubectl apply -f .** to configure all the yamls at once
- Configured Kubernetes Secret file for the MongoDB user credentials and added them to the deployment manifest. Secret is an object that contains sensitive information such as password etc. There is a small yaml to configure this –

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
immutable: false
type: Opaque
data:
  password: cGFzc3dvcmQ
```

- Added MongoDB **secret** to deployment manifest.

```
        env:
          - name: MONGO_INITDB_ROOT_USERNAME
            value: "admin"
          - name: MONGO_INITDB_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mongodb-secret
                key: password
```

- **MongoDb** is configured in server.js by -

  Imported mongodb library as –->
  const MongoClient = require('mongodb').MongoClient;
  and then used the connection url shown as 'uri' below. Database name is 'crud' -

```
const uri = 'mongodb://admin:password@localhost:32000/?authMechanism=DEFAULT';
MongoClient.connect(uri, (err, client) => {
    if (err) return console.log(err);
    db = client.db('crud');

    app.use('/api/v1/users', usersCtrl);

    app.listen(3000, function() {
        console.log('server running on port 3000', '');
    });
});
```

4. Below are the screenshots for all the created configurations explained above and execurted with command **kubectl apply -f .**
   These screenshot demonstrate successful creation of pv, pvc, etc that happened by running the kubectl get commands–

   Pv and pvc –

```
C:\Users\Karthik>kubectl get pvc
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mongo-pvc   Bound    pvc-b201d266-7066-4f07-a7ac-e326761e76ef   500M       RWX            hostpath       40m

C:\Users\Karthik>kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   REASON   AGE
mongo-pv                                   500M       RWX            Retain           Available                                                 40m
pvc-b201d266-7066-4f07-a7ac-e326761e76ef   500M       RWX            Delete           Bound       default/mongo-pvc   hostpath               40m
```

   Svc-

```
C:\Users\Karthik>kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)           AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP           8h
mongo-svc    NodePort    10.105.103.50   <none>        27017:32000/TCP   41m
```

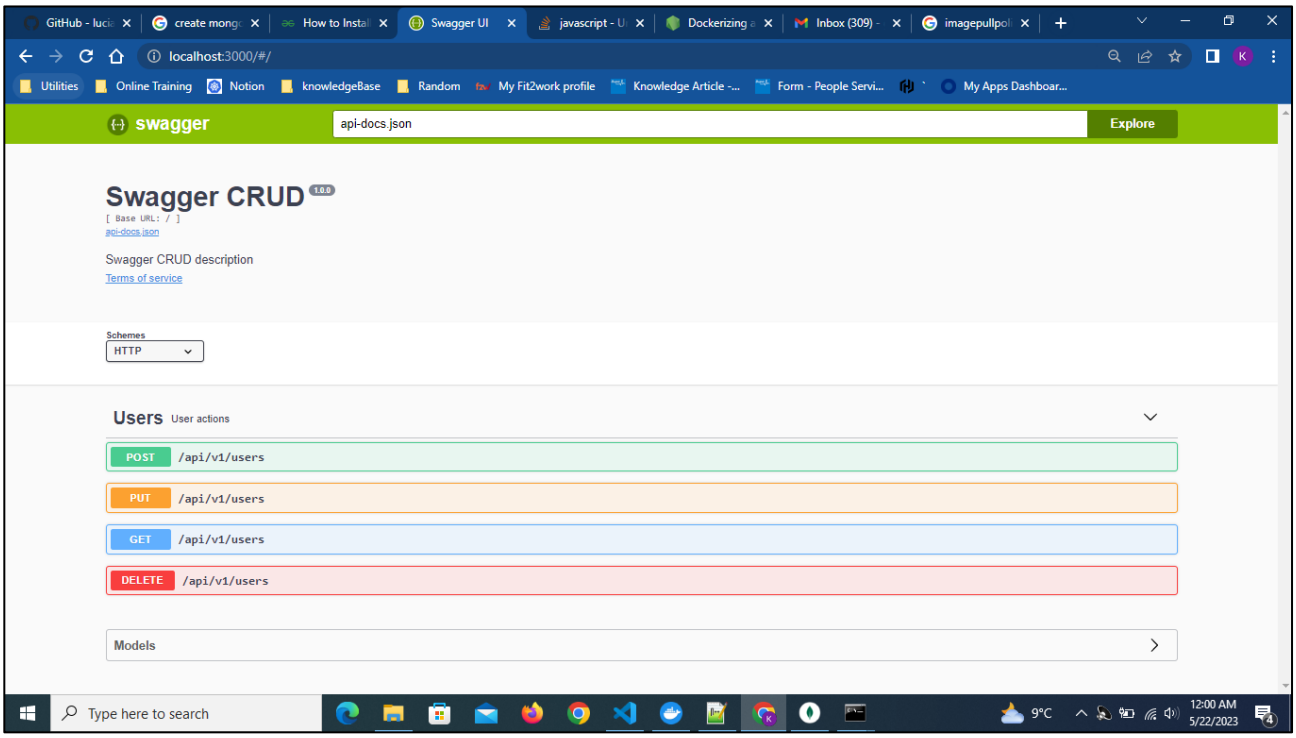
   Pods-

```
C:\Users\Karthik>kubectl get all -A
NAMESPACE     NAME                                         READY   STATUS    RESTARTS         AGE
default       pod/mongo-5fc8bb68b-ck5jx                    1/1     Running   0                40m
default       pod/nodejs-dbd4998f6-qgzc5                   1/1     Running   2 (17s ago)      37s
kube-system   pod/coredns-565d847f94-69jmg                 1/1     Running   0                8h
kube-system   pod/coredns-565d847f94-78dpb                 1/1     Running   0                8h
kube-system   pod/etcd-docker-desktop                      1/1     Running   0                8h
kube-system   pod/kube-apiserver-docker-desktop            1/1     Running   0                8h
kube-system   pod/kube-controller-manager-docker-desktop   1/1     Running   0                8h
kube-system   pod/kube-proxy-84llr                         1/1     Running   0                8h
kube-system   pod/kube-scheduler-docker-desktop            1/1     Running   2 (6h43m ago)    8h
kube-system   pod/storage-provisioner                      1/1     Running   2 (6h43m ago)    8h
kube-system   pod/vpnkit-controller                        1/1     Running   24 (7m46s ago)   8h
```

Both mongodb pod and nodejs pods are running

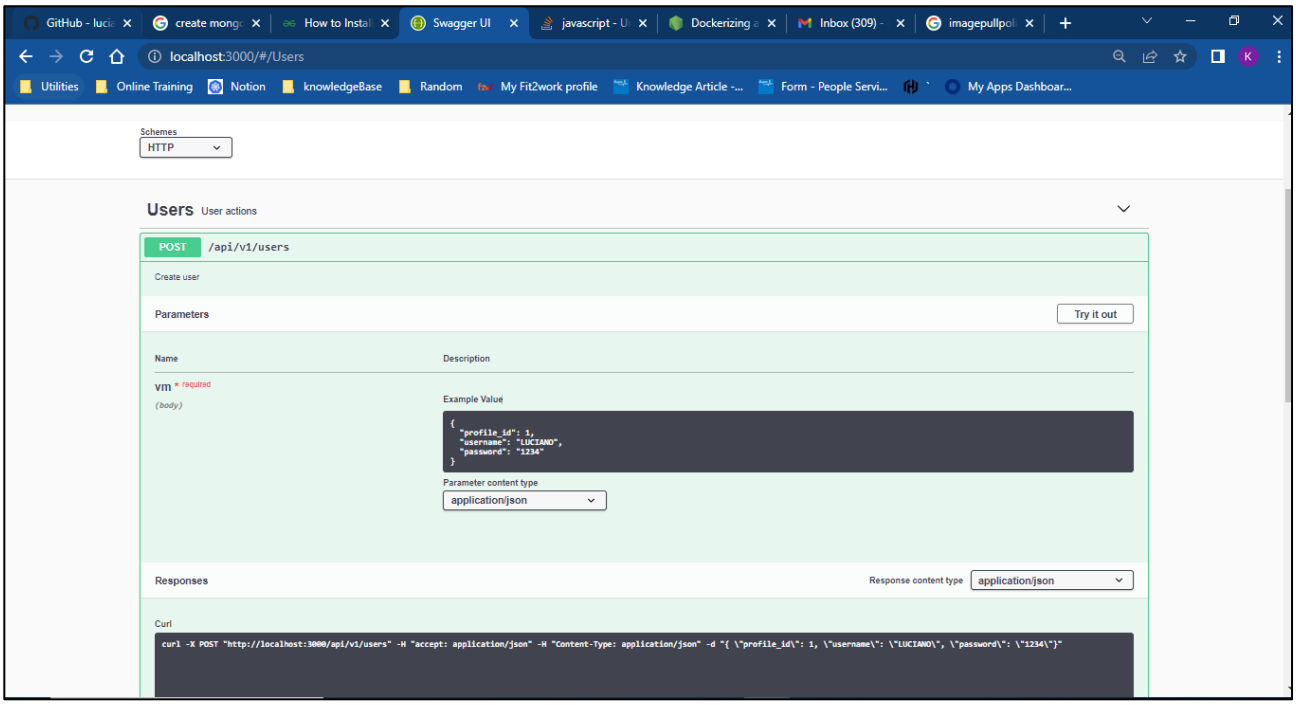
5. Tested the deployment for CRUD (Create, Read, Update, Delete) operations

Launched localhost:3000 as –

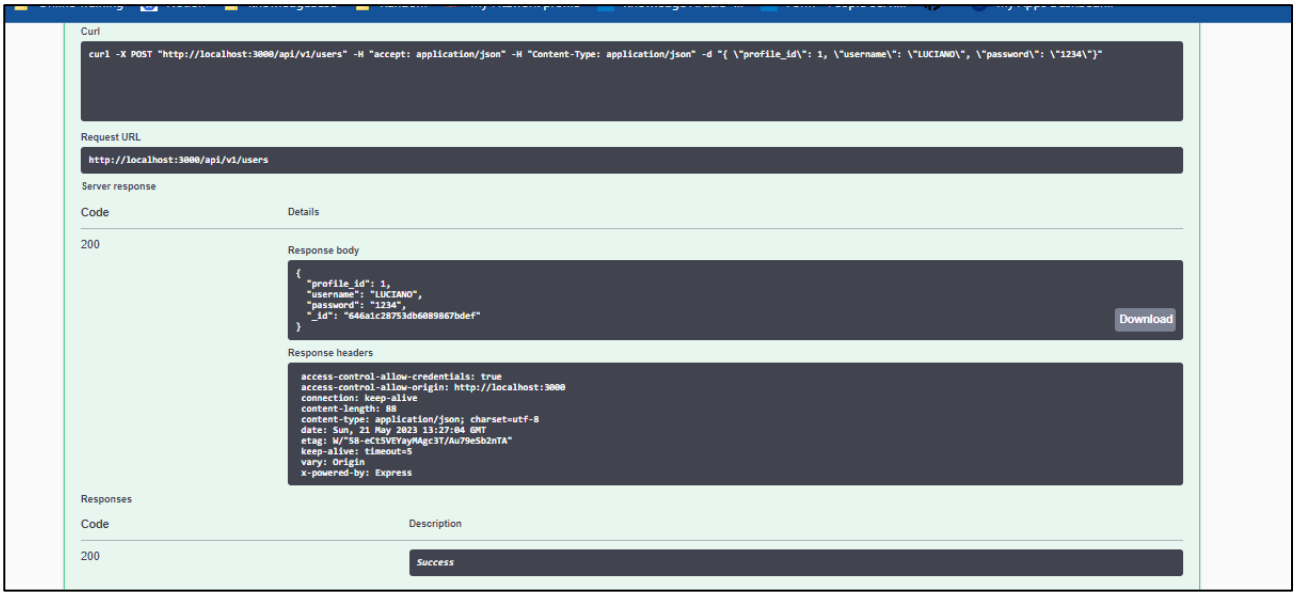


Step 1. Testing POST - /Create endpoint. The input and output screenshots are -

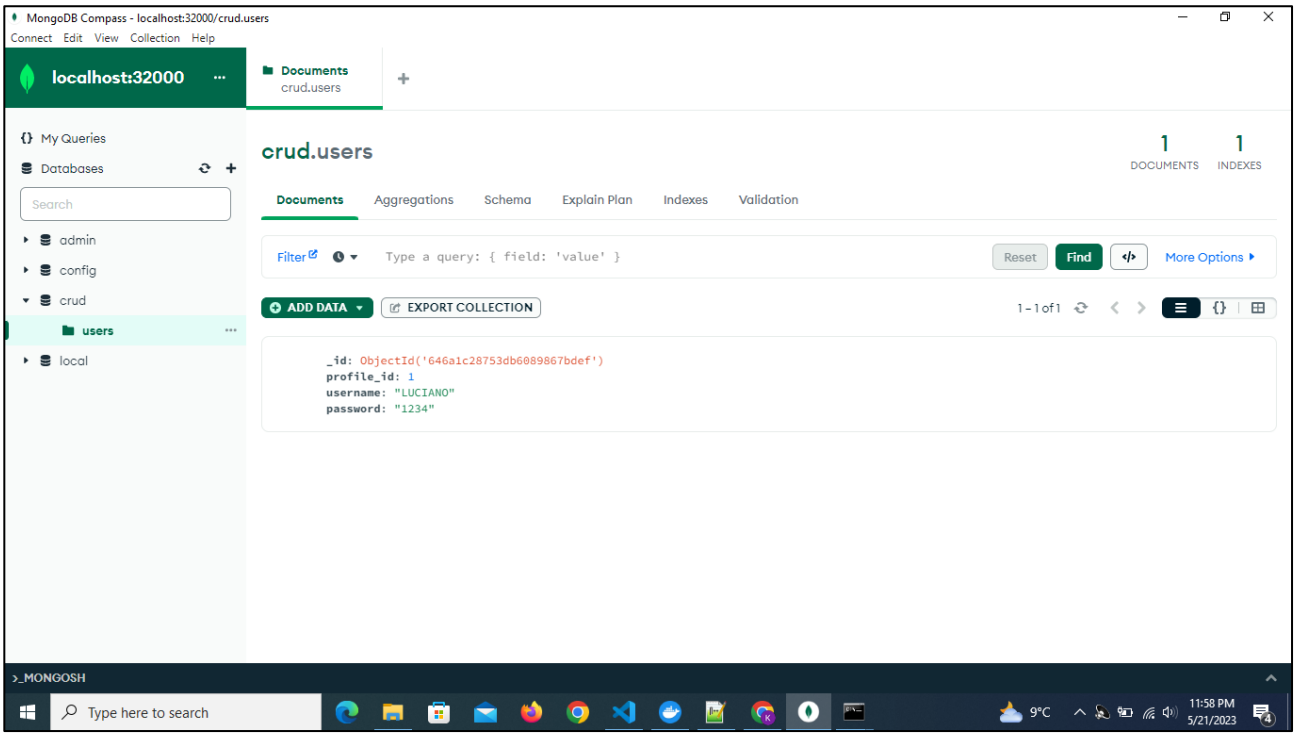
Inputted the body in POST as follows-



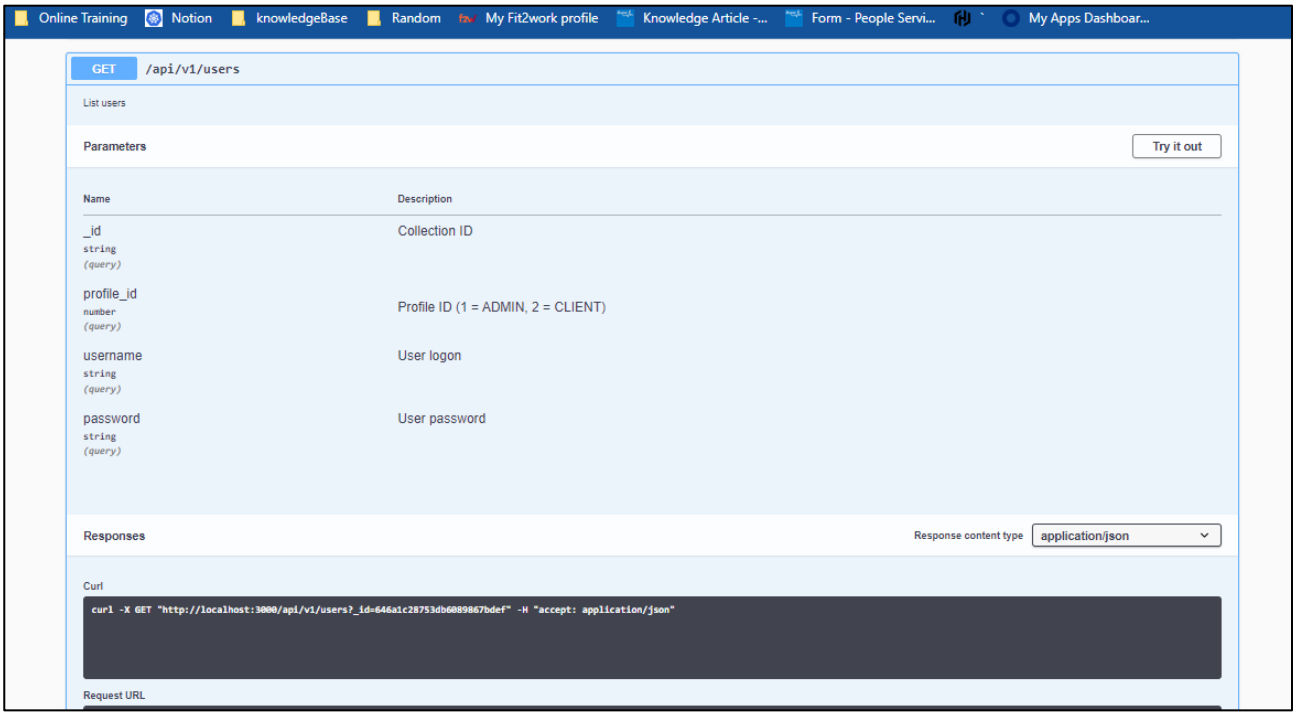
The output 200 was received as –

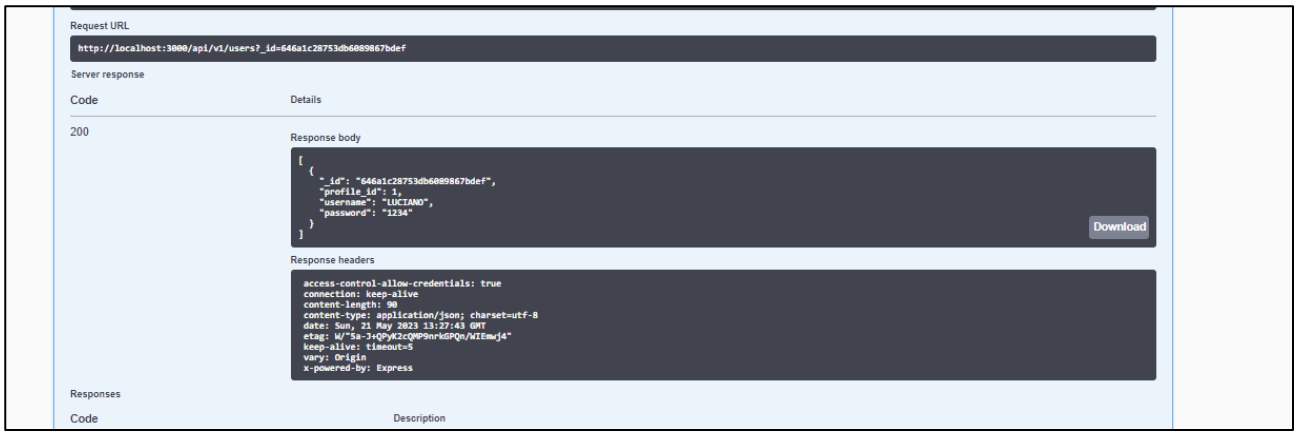


The Mongo db compass resultantly showed –



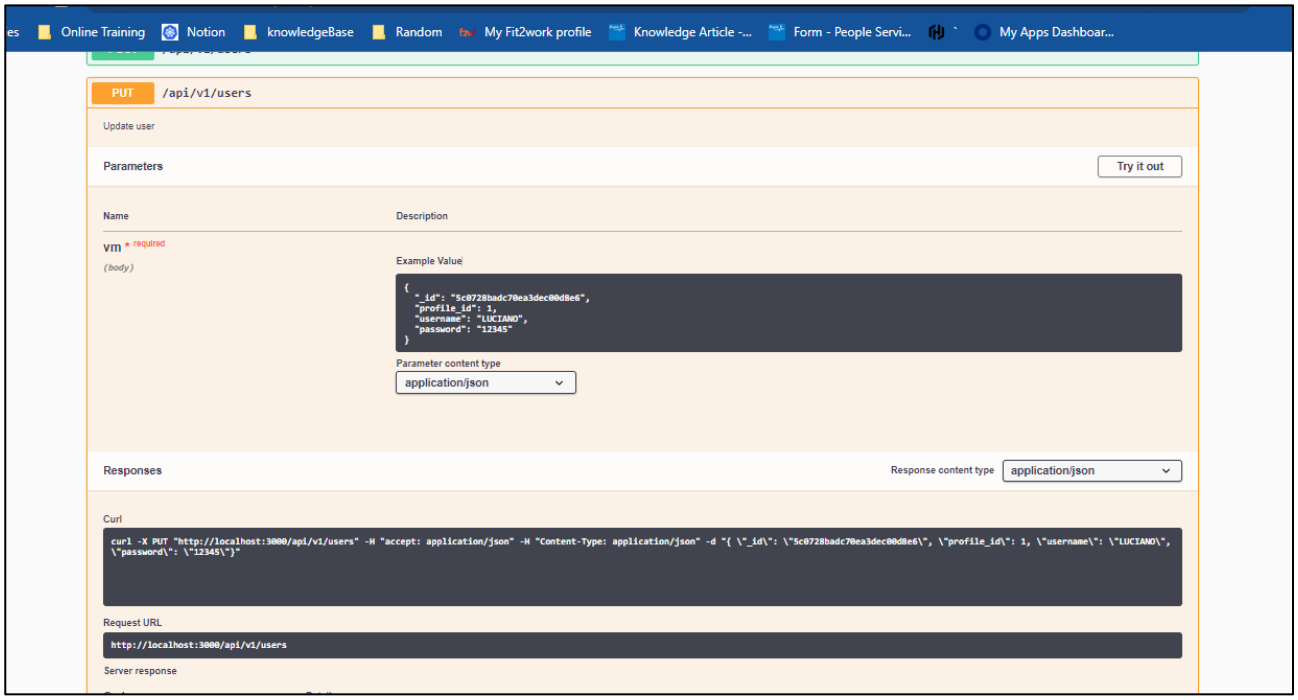
Step 2. GET operation –





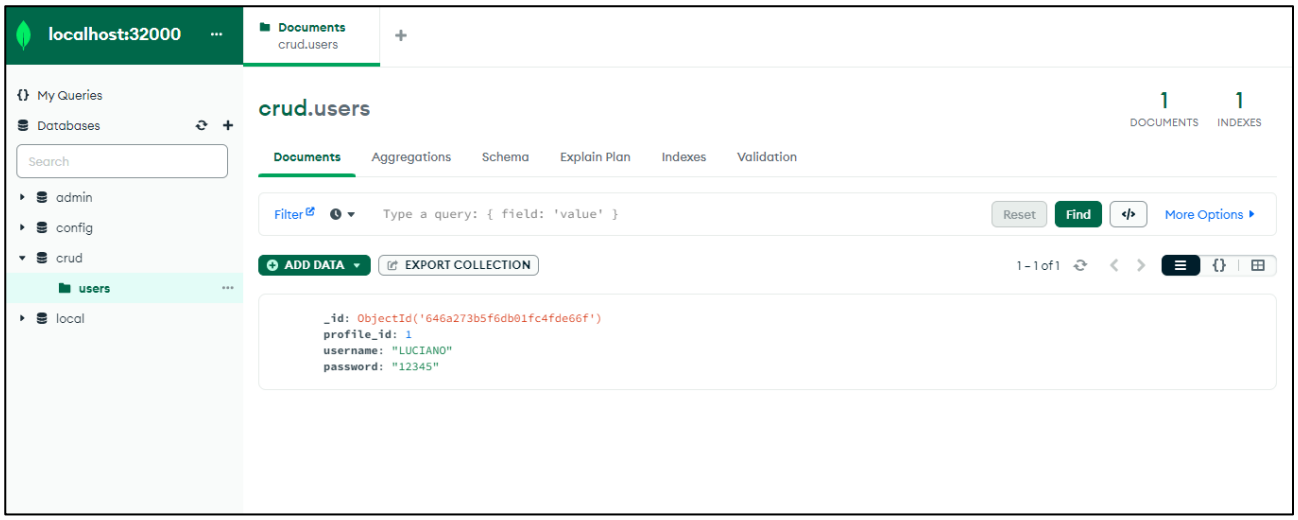
Step 3. PUT operation – Changed id from 1234 to 12345 for the object id created in the output of POST operation.

Input –

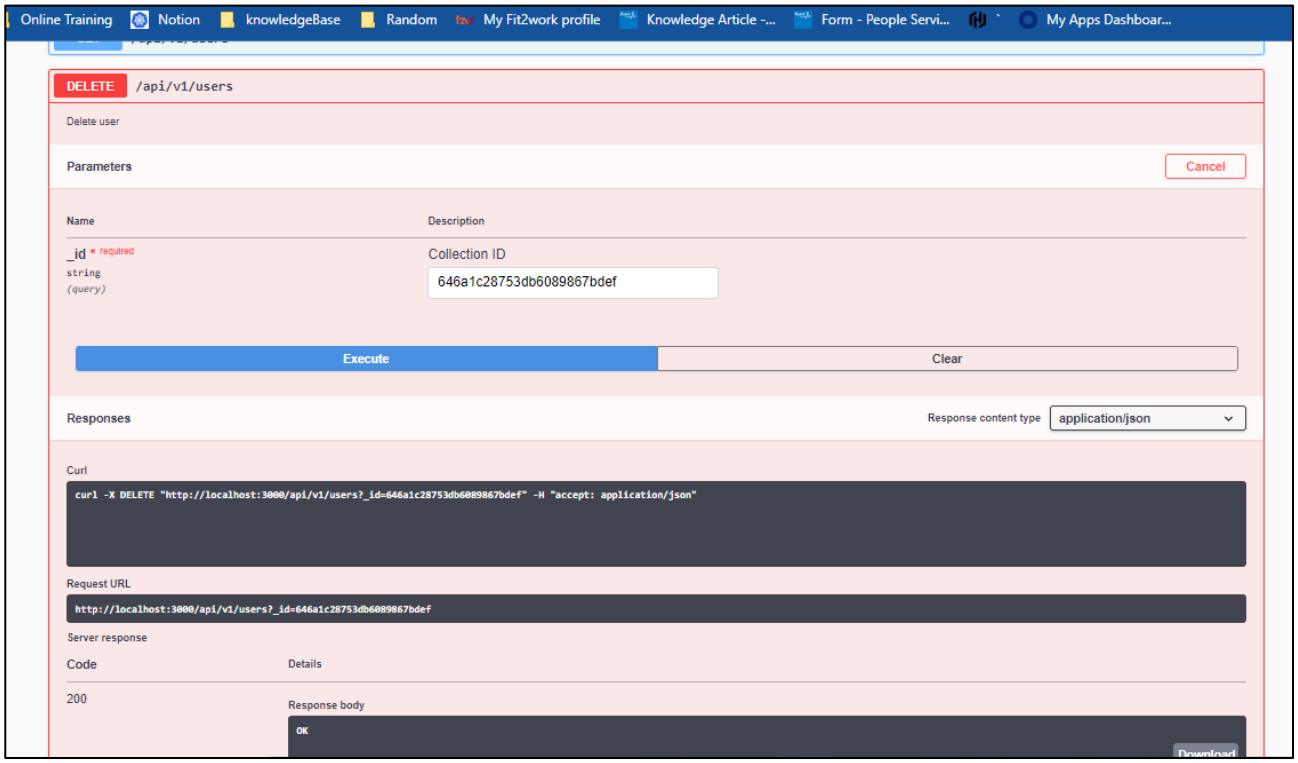


Output –

Corresponding change in db –

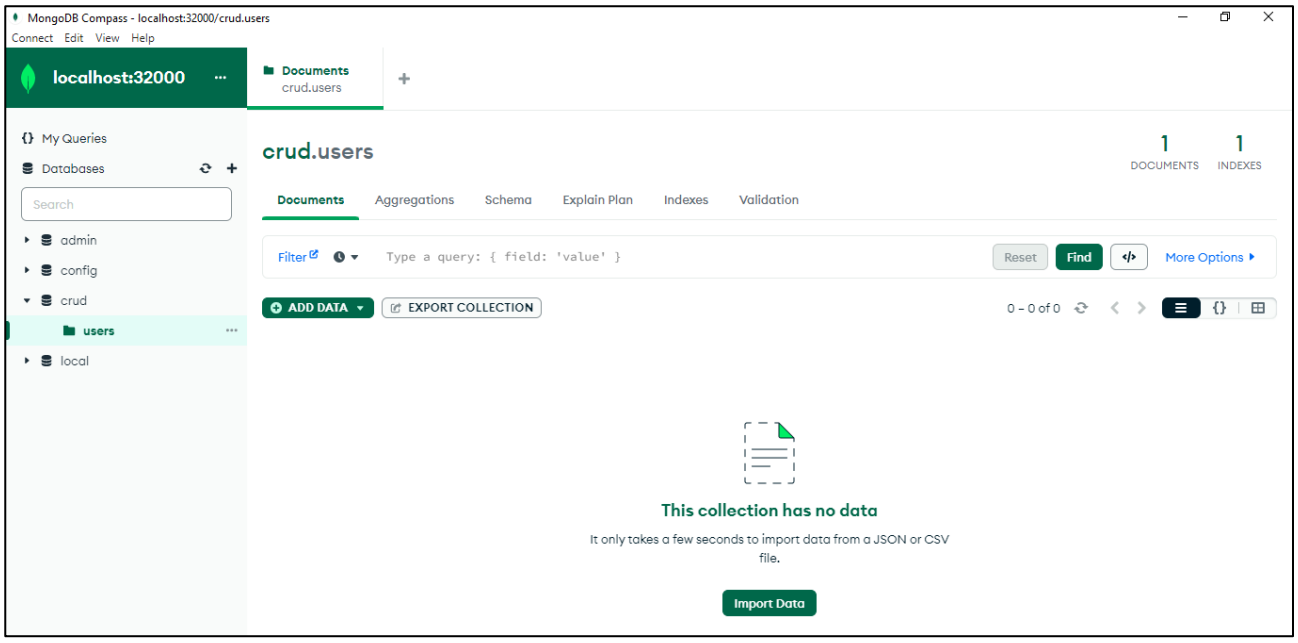


Step 4. DELETE operation using the id



Deletion success code 200 -

Effect on mongodb compass –



Note:
In the code submitted, all the Kubernetes code and configurations are done as per my understanding from the workshop content.
CRUD operations are learnt from research, mainly from the source – [https://github.com/lucianopereira86/NodeJS-MongoDB-Kubernetes/tree/master]