

MongoDB Granular Access Control - Testing Guide

This document provides a step-by-step guide on how to test the MongoDB database with granular access control. It includes login credentials for different users, expected results, and explanations of access levels. Ensure that MongoDB is running inside Docker before proceeding.

1. Prerequisites

The project is on Docker Hub and needs to be pulled from there before you can start. To pull the project run this command

```
```sh
docker pull oliroat/my-mongo
```
```

and then run this command to run the image you just pulled inside a container

```
```sh
docker run -d -p 27017:27017 --name mongo-container oliroat/my-mongo
```
```

Before testing, make sure MongoDB is running inside Docker with authentication enabled. Use the following command to check if the container is running:

```
```sh
docker ps
```
```

2. Database-Level Access Control

At this level, access is granted or restricted to entire databases.

Admin User - Full Access

Login as the admin user to verify access to all databases:

```
```sh
docker exec -it mongo-container mongosh -u admin -p adminPassword
--authenticationDatabase admin
```
```

Once inside MongoDB, run:

```
```sh
show dbs
```
```

✓ Expected Result: You should see ****all**** databases: `admin`, `customerDB`, `employeeDB`, and `config`.

Customer Admin - Full Access to customerDB

before you can login you need to exit the db using:

```
exit
```

then you can run this to login as Customer Admin:

```
```sh
docker exec -it mongo-container mongosh -u customerAdmin -p customerPass
--authenticationDatabase customerDB
```
```

```
```sh
show dbs
```
```

✓ Expected Result: Only `customerDB` should be listed.

Employee Admin - Full Access to employeeDB

before you can login you need to exit the db using:

```
exit
```

```
```sh
docker exec -it mongo-container mongosh -u employeeAdmin -p employeePass
--authenticationDatabase employeeDB
```
```

```
```sh
show dbs
```
```

✓ Expected Result: Only `employeeDB` should be listed.

3. Collection-Level Access Control

Customer Reader - Read-Only Access to customerDB

before you can login you need to exit the db using:

```
exit
```

```
```sh
docker exec -it mongo-container mongosh -u customerReader -p readerPass
--authenticationDatabase customerDB
```
```

```
```sh
use customerDB
show collections
```
```

✓ Expected Result: The `customers` collection should be listed.

```
```sh
db.customers.find()
```
```

✓ Expected Result: All customer records should be displayed. John doe, Jane smith and Alice brown

```
```sh
db.customers.insertOne({ name: 'Unauthorized', email: 'fail@example.com' })
```
```

✗ Expected Result: ****Unauthorized error**** should be returned.

Employee Reader - Read-Only Access to employeeDB

before you can login you need to exit the db using:

```
exit
```

```
```sh
docker exec -it mongo-container mongosh -u employeeReader -p readerPass
--authenticationDatabase employeeDB
```
```

```
```sh
use employeeDB
show collections
```
```

✓ Expected Result: The `employees` collection should be listed.

```
```sh
db.employees.find()
```
```

✓ Expected Result: All employee records should be displayed. Alice, Bob and Charlie

4. Document-Level Access Control

before you can login you need to exit the db using:


```
exit
```

Sales Rep - Restricted to Assigned Customers

```
```sh
docker exec -it mongo-container mongosh -u salesRep101 -p repPass101
--authenticationDatabase customerDB
```
```

```
```sh
use customerDB
```
```

```
```sh
db.customers.find()
```
```

 Expected Result: ****Unauthorized error**** should be returned.

Instead, use the restricted view:

```
```sh
db.salesRep101_view.find()
```
```

 Expected Result: Only customers with `salesRepId: 101` should be visible. so Bob should not be displayed

5. Field-Level Access Control

before you can login you need to exit the db using:

```
exit
```

HR Limited - Restricted from Seeing Salaries (Cell-Level)

This demonstrates Field level access, where certain fields in a document are hidden from specific users. In this case, HR users can only see `name` and `department`, but ****not salary****.

Login as the HR user:


```
```sh
docker exec -it mongo-container mongosh -u hrLimited -p hrLimitedPass
--authenticationDatabase employeeDB
```
```

Switch to `employeeDB`:

```
```sh
use employeeDB
```
```


Try reading the full `employees` collection (should fail):

```
```sh
db.employees.find()
```
```

 Expected Result: **Unauthorized error** should be returned since HR users do not have direct access to `employees`.

Instead, query the restricted view `employeeHR_view`:

```
```sh
db.employeeHR_view.find()
```
```

 Expected Result: Only `name` and `department` fields should be visible, while `salary` remains hidden:

```
```json
[
 { "_id": 1, "name": "Alice Manager", "department": "HR" },
 { "_id": 2, "name": "Bob Engineer", "department": "IT" }
 { "_id": 3, "name": "Charlie sales", "department": "Sales" }
]
```
```

Sales Rep - Insert-Only Access

Sales reps can add new customers but **cannot update or delete existing customers**.

before you can login you need to exit the db using:

```
exit
```

Login as the sales rep writer:

```
```sh
docker exec -it mongo-container mongosh -u salesRepWriter -p writePass
--authenticationDatabase customerDB
```
```

Switch to `customerDB`:

```
```sh
use customerDB
```
```

Try inserting a new customer (should succeed):

```
```sh
db.customers.insertOne({ name: "New Client", email: "client@example.com", salesRepId:
103 })
```
```

✓ Expected Result: Insert operation succeeds.

Try updating an existing customer (should fail):

```
```sh
db.customers.updateOne({ name: "John Doe" }, { $set: { email: "newemail@example.com" }
})
```
```

✗ Expected Result: Unauthorized error.

Try deleting a customer (should fail):

```
```sh
db.customers.deleteOne({ name: "John Doe" })
```
```

✗ Expected Result: Unauthorized error.

Verifying Inserted Customers

Since Sales Reps do not have read access to the `customers` collection, we need to log in as `customerAdmin` to verify that the new customer was added.

Before logging in, exit the Sales Rep session:

```
```sh
exit
```
```

Login as Customer Admin to verify the inserted record:


```
```sh
docker exec -it mongo-container mongosh -u customerAdmin -p customerPass
--authenticationDatabase customerDB
```
```

Switch to `customerDB`:

```
```sh
use customerDB
```
```

Check if the newly inserted customer exists:

```
``sh
db.customers.find({ name: "New Client" })
``
```

 Expected Result: The inserted customer should be displayed in the result:

```
``json
[
  { "_id": ObjectId("..."), "name": "New Client", "email": "client@example.com", "salesRepId":
103 }
]
``
```