

Introduction

This assignment is based on developing a web coordination policy between the server and the client, where the server is responsible for checking the order of operations the client performs. The server has 5 endpoints namely login, display, placeOrder, returnOrder, and logout. I am using a laptop dataset from [kaggle](#). The client will interact with all these endpoints in sequential order or out of order. The way I am trying to implement the coordination policy is by allowing only logged-in users to query the laptop details, and place or return laptops. Also, I am allowing a client to return a laptop only if they have placed an order.

Technologies/components used

Technology	Usage
Fast API	For creating endpoints(web services) for the client to interact.
MongoDB	The database is used for storing laptop details, user credentials, and user-application communication stages
Python	Client-side application testing

Web service Design

For **identification of a conversation**, I am using the **clients username** from header.

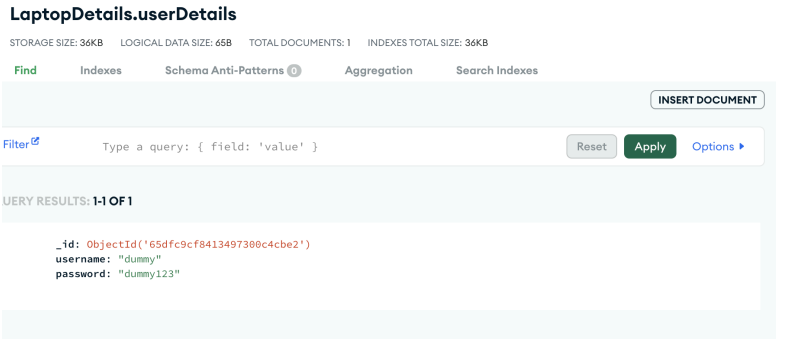
Endpoints

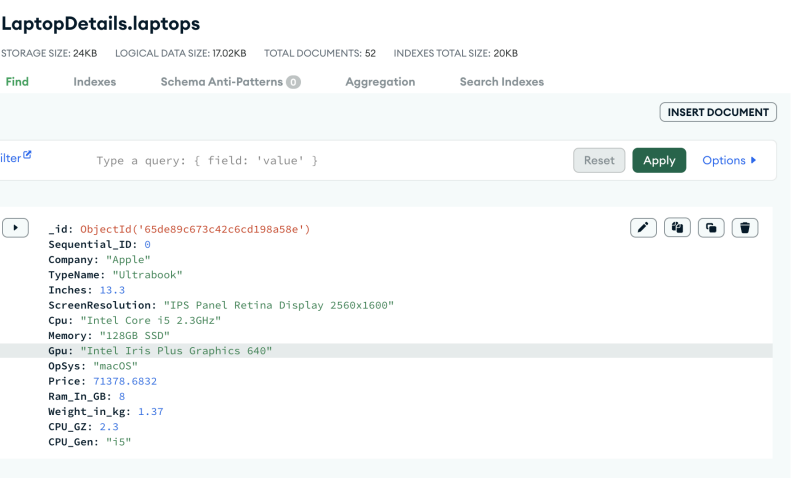
Web service	Server endpoint	Client Usage
Login - To allow the client to login to the application	http://127.0.0.1:8000/login	"http://127.0.0.1:8000/login", headers={"username": "Shree", "password": "shree123"}
Display - List all the laptops of different brands and their details.	http://127.0.0.1:8000/displayLaptops	"http://127.0.0.1:8000/displayLaptops", headers=headers, params=data
Place Order - Place order for laptop with a particular ID.	http://127.0.0.1:8000/placeOrder	"http://127.0.0.1:8000/placeOrder", headers=headers, params=data)
Return Order - Return the order of the placed laptop.	http://127.0.0.1:8000/returnOrder	"http://127.0.0.1:8000/returnOrder", headers=headers, params=data)
Logout - Client logs out	http://127.0.0.1:8000/logout	"http://127.0.0.1:8000/logout", headers={"username": "Shree",

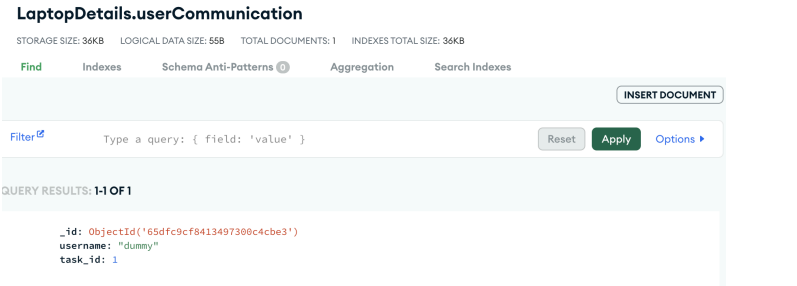
		"password": "shree123"}})
--	--	---------------------------

Backend Database

I have used MongoDB for storing the laptop details, user credentials, and user-application communication stages.

User Credentials	Database
Collection name: userDetails Fields: _id, username, password Usage: To store the user credentials every time new user logs in	 <p>LaptopDetails.userDetails STORAGE SIZE: 36KB LOGICAL DATA SIZE: 65B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB</p> <p>Find Indexes Schema Anti-Patterns Aggregation Search Indexes</p> <p>INSERT DOCUMENT</p> <p>Filter Type a query: { field: 'value' } Reset Apply Options</p> <p>QUERY RESULTS: 1-1 OF 1</p> <pre>{ "_id": "ObjectId('65dfc9cf8413497308c4cbe2')", "username": "dummy", "password": "dummy123" }</pre>

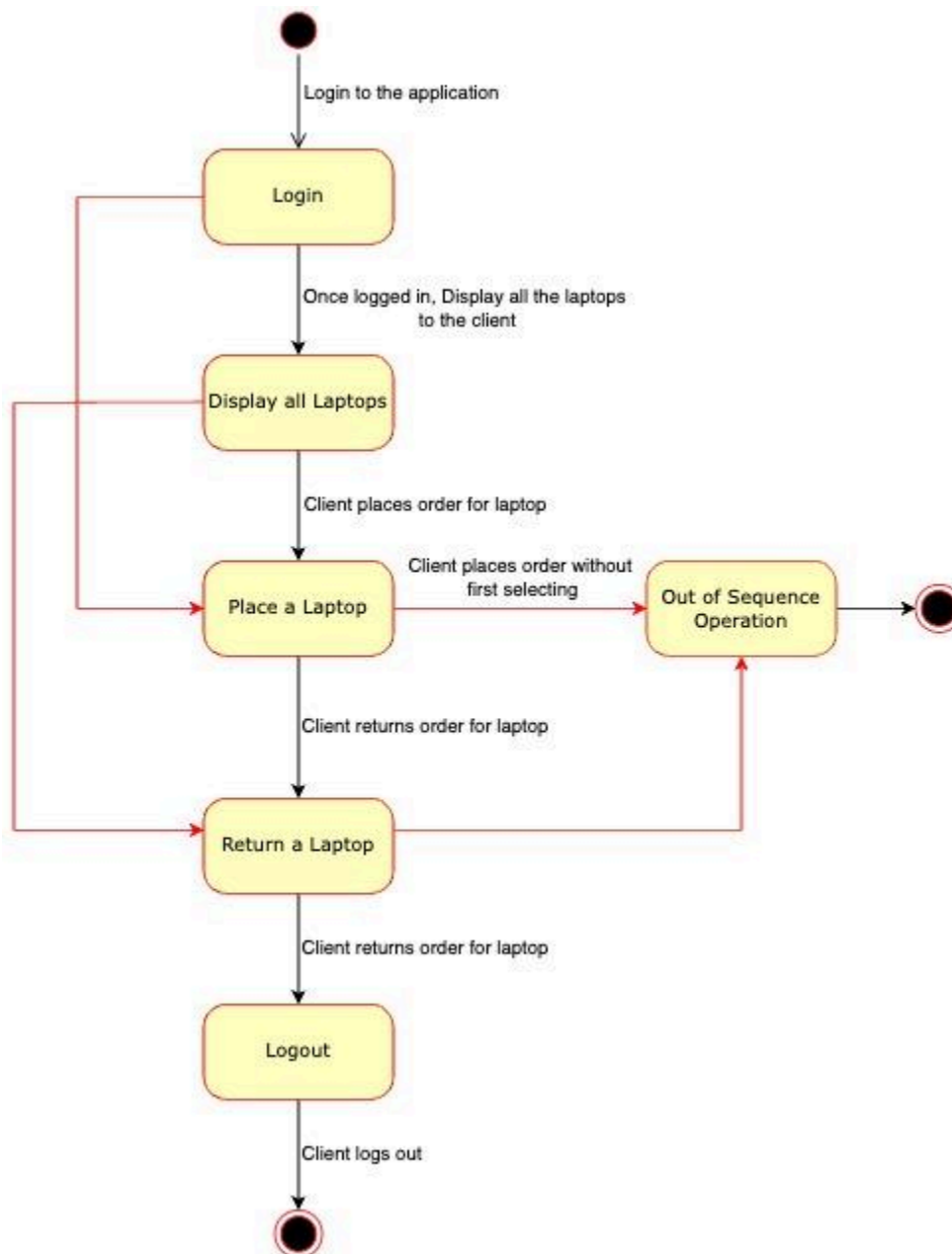
Laptop details	Database
Collection name: Laptops Fields: _id, Sequential_id, Company, TypeName, Inches, ScreenResolution, Cpu, Memory, Gpu, OpSys, Price, Ram_In_GB, Weight_in_kg, CPU_GZ, CPU_Gen Usage: To store the details of laptops	 <p>LaptopDetails.laptops STORAGE SIZE: 24KB LOGICAL DATA SIZE: 17.02KB TOTAL DOCUMENTS: 52 INDEXES TOTAL SIZE: 20KB</p> <p>Find Indexes Schema Anti-Patterns Aggregation Search Indexes</p> <p>INSERT DOCUMENT</p> <p>Filter Type a query: { field: 'value' } Reset Apply Options</p> <p>QUERY RESULTS: 1-1 OF 1</p> <pre>{ "_id": "ObjectId('65de89c673c42c6cd198a58e')", "Sequential_ID": 0, "Company": "Apple", "TypeName": "Ultrabook", "Inches": 13.3, "ScreenResolution": "IPS Panel Retina Display 2560x1600", "Cpu": "Intel Core i5 2.3GHz", "Memory": "128GB SSD", "Gpu": "Intel Iris Plus Graphics 640", "OpSys": "macOS", "Price": 71378.6832, "Ram_In_GB": 8, "Weight_in_kg": 1.37, "CPU_GZ": 2.3, "CPU_Gen": "i5" }</pre>

User Communication	Database
Collection name: userCommunication Fields: _id, username, task_id Usage: To store the current state of user communication with the application	 <p>LaptopDetails.userCommunication STORAGE SIZE: 36KB LOGICAL DATA SIZE: 55B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB</p> <p>Find Indexes Schema Anti-Patterns Aggregation Search Indexes</p> <p>INSERT DOCUMENT</p> <p>Filter Type a query: { field: 'value' } Reset Apply Options</p> <p>QUERY RESULTS: 1-1 OF 1</p> <pre>{ "_id": "ObjectId('65dfc9cf8413497308c4cbe3')", "username": "dummy", "task_id": 1 }</pre>

Coordination Protocol

In this diagram, the coordination policy is user login and returning an order. This means that the server will display the list of laptops only if the user has logged in. The server also won't allow them to place or return an order until the client has logged in. Another coordination policy will be that, the server won't allow the client to return an order unless the order has been previously placed. In the state diagram below following along the red lines leads to error.

State activity diagram



in detail from the above state diagram.

Below are the steps described

1. The first step for the client is to login to the application using the login endpoint. This is where the client's credentials will be stored in the database along with the first communication ID as 1 in the userCommunication table. If after logging in the client tries **to place an order for the laptop**, then the client has performed an out of sequence operation. The application will then exit.
2. Next step includes the client viewing all the available laptops using the endpoint displayLaptops. This communication will be stored in the userCommunication table with ID 2. If after display, the client tries **to place a return order** then the client has performed an out of sequence operation. The application will then exit.
3. After selecting the laptop id, the client will now place an order using the endpoint placeOrder. This communication will be stored in the userCommunication table with ID 3.
4. Now that the client has ordered the laptop they can return it. This communication will be stored in the userCommunication table with ID 4.
5. The last operation includes the client logging out of the application.

Web Service Implementation

1. **Initialize the database** with user-related collections and laptop collections.
Step 1: Create the laptop collection and then load the data into it.
 - **Run the script loadData.py**Step 2: Create the userCredentials collection and the userCommunication collection.
 - **Run the createTables.py**
2. **Server-side** logic:
 1. Initially, a sequence dictionary (**sequence**) is used to define the order of operations: login(1), displayLaptops(2), placeOrder(3), and returnOrder(4).
 2. After that, I have initialized connection to my collections in the database. The userCommunication task ID is updated for each operation to enforce the correct order.
 3. I have also defined a base model for the user credentials as part of input validation in the server side.
 4. Access to MongoDB is enabled using the motor.motor_asyncio library for asynchronous interaction. The code uses asynchronous patterns, making use of async and await for MongoDB interactions.
 5. Operations performed and how the states are handled are given as below.
 1. **/login**
Handles user authentication. Checks if the user exists in the database. If not, adds the user to the database and updates task_id in userCommunication table to state 1.
 2. **/displayLaptops**
Displays all laptops if the user is logged in and in the correct order. Checks the user's current task_id, this should be 1. Only then update it for

the current sequence order which is 2 and store it in userCommunication table.

3. /placeOrder

Places an order if the user is logged in and in the correct order.

Verifies the availability of the selected laptop in the database. Checks the user's current task_id, this should be 2. Only then updates the task_id in userCommunication table for the current order which is 3.

4. /returnOrder

Places a return order if the user is logged in and in the correct order.

Checks the user's current task_id, this should be 3. Only then update the task_id in userCommunication table for the current sequence order which is 4 and store it in the database.

5. /logout

Logs the user out by deleting entries from user details and user communication from the database.

Client Side/Testing

This client-side code interacts with the FastAPI web service by making HTTP requests to different endpoints. It performs a series of operations such as logging in, displaying laptops, placing orders, returning orders, and logging out. The three test scenarios to check the coordination policy enforcement are as below.

a. runOperationsInOrder_test1

This test will run all the operations in order namely log in, display laptops, place an order, return the order, and then log out. This test function will not run into any error as all the functions are getting executed in order.

Output:

```
(practice) (base) shreenidhi@Shreenidhis-MacBook-Pro practice % python client.py
Order your Laptop
Test 1: All the operations run in order
"You are ready to shop!"
({'Sequential_ID':0,"Company":"Apple","Type Name":"Ultrabook","Inches":13.3,"ScreenResolution":"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 2.3GHz","Memory":"128GB '
'SSD","Gpu":"Intel Iris Plus Graphics '
'640","OpSys":"macOS","Price":71378.6832,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":2.3,"CPU_Gen":"i5"},{"Sequential_ID":1,"Company":"Apple",
Type Name":"Ultrabook","Inches":13.3,"ScreenResolution":"1440x900","Cpu":"Intel '
'Core i5 1.8GHz","Memory":"128GB Flash Storage","Gpu":"Intel HD Graphics '
'6000","OpSys":"macOS","Price":47895.5232,"Ram_In_GB":8,"Weight_in_kg":1.34,"CPU_GZ":1.8,"CPU_Gen":"i5"},{"Sequential_ID":2,"Company":"HP","Ty
pe Name":"Notebook","Inches":15.6,"ScreenResolution":"Full '
'HD 1920x1080","Cpu":"Intel Core i5 7200U 2.5GHz","Memory":"256GB '
'SSD","Gpu":"Intel HD Graphics 620","OpSys":"No '
'

.....
'10","Price":58767.84,"Ram_In_GB":8,"Weight_in_kg":1.49,"CPU_GZ":1.8,"CPU_Gen":"i7"},{"Sequential_ID":51,"Company":"Acer","Type Name":"Noteboo
ok","Inches":13.3,"ScreenResolution":"Full '
'HD 1920x1080","Cpu":"Intel Core i7 8550U 1.8GHz","Memory":"512GB '
'SSD","Gpu":"Intel UHD Graphics 620","OpSys":"Windows '
'10","Price":58767.84,"Ram_In_GB":8,"Weight_in_kg":1.49,"CPU_GZ":1.8,"CPU_Gen":"i7"},{"Sequential_ID":51,"Company":"Acer","Type Name":"Notebook
","Inches":15.6,"ScreenResolution":"1366x768","Cpu":"Intel '
'Core i3 7100U 2.4GHz","Memory":"1TB HDD","Gpu":"Intel HD Graphics '
'620","OpSys":"Windows '
'10","Price":20459.52,"Ram_In_GB":4,"Weight_in_kg":2.4,"CPU_GZ":2.4,"CPU_Gen":"i3"}])
"Order Placed"
"Return Order Placed"
"User logged out!"
```

b. runOperationsInOrder_test2

This test will run the operations like login, place an order, and then log out. This test function will result in an error **“Wrong sequence, you need to display all the laptops first”**, because the user is trying to place an order without first selecting the product i.e. displaying.

Output:

```
Test 2: User tries to place order before displaying the laptops
"You are ready to shop!"
"Wrong sequence, you need to display all the laptops first"
"User logged out!"
```

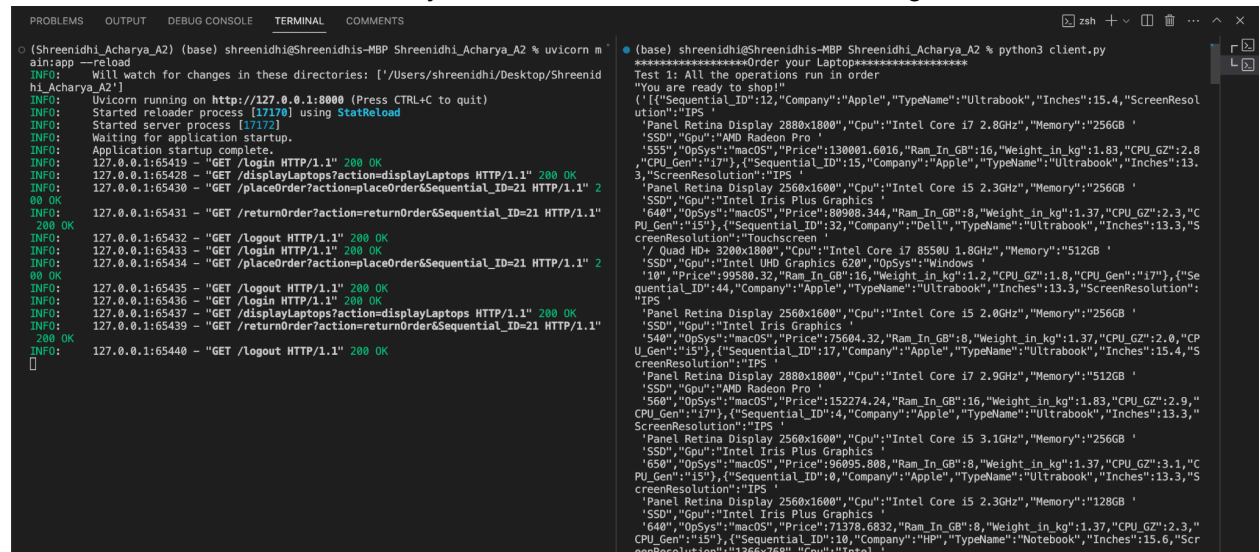
c. runOperationsInOrder_test3

This test will run operations like logging in, displaying laptops, returning an order, and then logging out. This test function will also result in an error **“Wrong sequence, you need to place the laptop order first”**, because the user is trying to return an order without first placing an order for the same product.

Output:

```
Test 3: User tries to return the laptop before placing the order
"You are ready to shop!"
({{"Sequential_ID":0,"Company":"Apple","TypeName":"Ultrabook","Inches":13.3,"ScreenResolution":"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 2.3GHz","Memory":"128GB '
'SSD","Gpu":"Intel Iris Plus Graphics '
'640","OpSys":"macOS","Price":71378.6832,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":2.3,"CPU_Gen":"i5"},{"Sequential_ID":1,"Company":"Apple","
TypeName":"Ultrabook","Inches":13.3,"ScreenResolution":"1440x900","Cpu":"Intel '
'Core i5 1.8GHz","Memory":"128GB Flash Storage","Gpu":"Intel HD Graphics '
'6000","OpSys":"macOS","Price":47895.5232,"Ram_In_GB":8,"Weight_in_kg":1.34,"CPU_GZ":1.8,"CPU_Gen":"i5"},{"Sequential_ID":2,"Company":"HP","Ty
peName":"Notebook","Inches":15.6,"ScreenResolution":"Full '
'HD 1920x1080","Cpu":"Intel Core i5 7200U 2.5GHz","Memory":"256GB '
'SSD","Gpu":"Intel UHD Graphics 620","OpSys":"Windows '
'10","Price":58767.84,"Ram_In_GB":8,"Weight_in_kg":1.49,"CPU_GZ":1.8,"CPU_Gen":"i7"},{"Sequential_ID":51,"Company":"Acer","TypeName":"Notebook
","Inches":15.6,"ScreenResolution":"1366x768","Cpu":"Intel '
'Core i3 7100U 2.4GHz","Memory":"1TB HDD","Gpu":"Intel HD Graphics '
'620","OpSys":"Windows '
'10","Price":20459.52,"Ram_In_GB":4,"Weight_in_kg":2.4,"CPU_GZ":2.4,"CPU_Gen":"i3"]})
"Wrong sequence, you need to place the laptop order first"
"User logged out!"
```

d. The overall screenshot when you run both client and server side together will look like below.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
o (Shreenidhi_Acharya_A2) (base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 % uvicorn m
ain:app --reload
INFO: Will watch for changes in these directories: ['/Users/shreenidhi/Desktop/Shreenid
hi_Acharya_A2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reload process [17170] using StatReload
INFO: Started server process [17172]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:65419 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:65428 - "GET /displayLaptops?action=displayLaptops HTTP/1.1" 200 OK
INFO: 127.0.0.1:65438 - "GET /placeOrder?action=placeOrder&Sequential_ID=21 HTTP/1.1" 2
00 OK
INFO: 127.0.0.1:65431 - "GET /returnOrder?action=returnOrder&Sequential_ID=21 HTTP/1.1"
200 OK
INFO: 127.0.0.1:65432 - "GET /logout HTTP/1.1" 200 OK
INFO: 127.0.0.1:65433 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:65434 - "GET /placeOrder?action=placeOrder&Sequential_ID=21 HTTP/1.1" 2
00 OK
INFO: 127.0.0.1:65435 - "GET /logout HTTP/1.1" 200 OK
INFO: 127.0.0.1:65436 - "GET /login HTTP/1.1" 200 OK
INFO: 127.0.0.1:65437 - "GET /displayLaptops?action=displayLaptops HTTP/1.1" 200 OK
INFO: 127.0.0.1:65439 - "GET /returnOrder?action=returnOrder&Sequential_ID=21 HTTP/1.1"
200 OK
INFO: 127.0.0.1:65440 - "GET /logout HTTP/1.1" 200 OK
[]

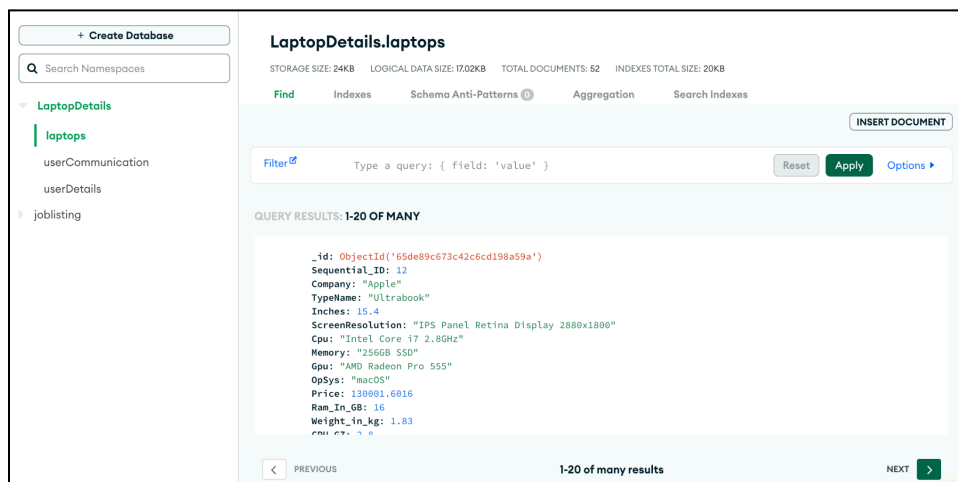
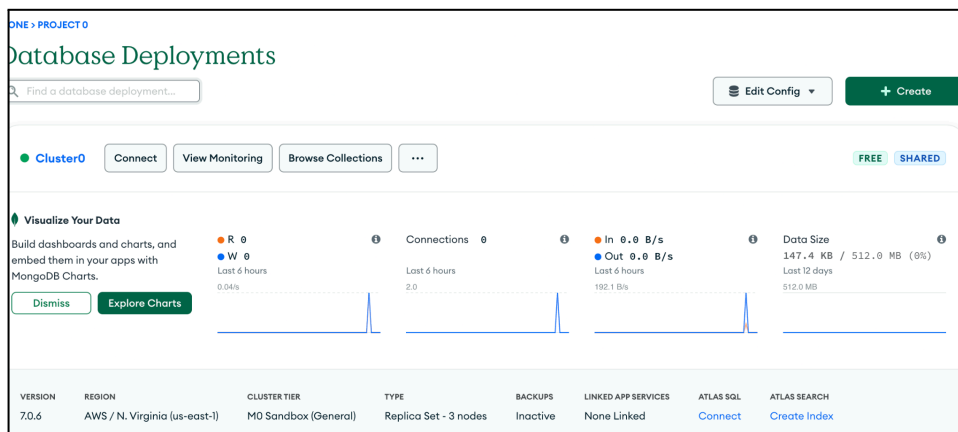
(base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 % python3 client.py
*****Order your Laptop*****
Test 1: All the operations run in order
"You are ready to shop!"
({{"Sequential_ID":12,"Company":"Apple","TypeName":"Ultrabook","Inches":15.4,"ScreenResol
ution":"IPS '
'Panel Retina Display 2880x1800","Cpu":"Intel Core i7 2.8GHz","Memory":"256GB '
'SSD","Gpu":"AMD Radeon Pro '
'555","OpSys":"macOS","Price":130001.6016,"Ram_In_GB":16,"Weight_in_kg":1.83,"CPU_GZ":2.8
,"CPU_Gen":"i7"},{"Sequential_ID":15,"Company":"Apple","TypeName":"Ultrabook","Inches":13.
3,"ScreenResolution":"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 2.3GHz","Memory":"256GB '
'SSD","Gpu":"Intel Iris Plus Graphics '
'640","OpSys":"macOS","Price":80908.344,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":2.3,"C
PU_Gen":"i5"},{"Sequential_ID":32,"Company":"Dell","TypeName":"Ultrabook","Inches":13.3,"S
creenResolution":"Touchscreen '
'Quad HD+ 3200x1800","Cpu":"Intel Core i7 8550U 1.8GHz","Memory":"512GB '
'SSD","Gpu":"Intel UHD Graphics 620","OpSys":"Windows '
'10","Price":99580.32,"Ram_In_GB":16,"Weight_in_kg":1.2,"CPU_GZ":1.8,"CPU_Gen":"i7"},{"Se
quential_ID":44,"Company":"Apple","TypeName":"Ultrabook","Inches":13.3,"ScreenResolution":
"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 2.0GHz","Memory":"256GB '
'SSD","Gpu":"Intel Iris Graphics '
'540","OpSys":"macOS","Price":75604.32,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":2.0,"CP
U_Gen":"i5"},{"Sequential_ID":17,"Company":"Apple","TypeName":"Ultrabook","Inches":15.4,"S
creenResolution":"IPS '
'Panel Retina Display 2880x1800","Cpu":"Intel Core i7 2.9GHz","Memory":"512GB '
'SSD","Gpu":"AMD Radeon Pro '
'560","OpSys":"macOS","Price":152274.24,"Ram_In_GB":16,"Weight_in_kg":1.83,"CPU_GZ":2.9,"
CPU_Gen":"i7"},{"Sequential_ID":4,"Company":"Apple","TypeName":"Ultrabook","Inches":13.3,"
ScreenResolution":"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 3.1GHz","Memory":"256GB '
'SSD","Gpu":"Intel Iris Plus Graphics '
'650","OpSys":"macOS","Price":96095.808,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":3.1,"C
PU_Gen":"i5"},{"Sequential_ID":0,"Company":"Apple","TypeName":"Ultrabook","Inches":13.3,"S
creenResolution":"IPS '
'Panel Retina Display 2560x1600","Cpu":"Intel Core i5 2.3GHz","Memory":"128GB '
'SSD","Gpu":"Intel Iris Plus Graphics '
'640","OpSys":"macOS","Price":71378.6832,"Ram_In_GB":8,"Weight_in_kg":1.37,"CPU_GZ":2.3,"
CPU_Gen":"i5"},{"Sequential_ID":10,"Company":"HP","TypeName":"Notebook","Inches":15.6,"Scr
eenResolution":"1366x768","Cpu":"Intel '
'Core i3 7100U 2.4GHz","Memory":"1TB HDD","Gpu":"Intel HD Graphics '
'620","OpSys":"Windows '
'10","Price":20459.52,"Ram_In_GB":4,"Weight_in_kg":2.4,"CPU_GZ":2.4,"CPU_Gen":"i3"]})
"Wrong sequence, you need to display all the laptops first"
"User logged out!"
```

Readme

The setup for this assignment is divided into two parts. Database setup and Testing.

Database:

1. Since the database is on cloud, you dont need to load the collection with laptop data again. I have already loaded it for you to test. However the script that loads and create the collections are present in the program directory. The script **loadData.py**, will load all the data of the laptop into the laptops collection. The script **createTables.py** will create the collections for storing the user credentials and user communication and load these with some dummy starter values. Again, **you dont need to run both of these files**.
2. For verification purpose you can go to the link as below and check the collections
<https://cloud.mongodb.com/v2/64a55d3f8829345b54db94ba#/clusters>
Username: shreenidhiacharya7@gmail.com
Password: admin1234pq
3. On log in you will see the below home page and collections.

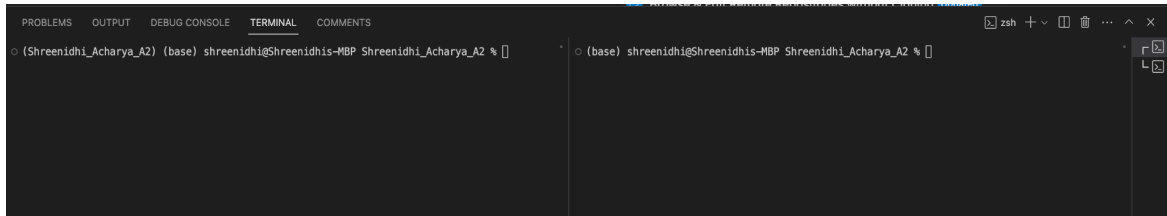


4. This database is accessible from any ip address so you dont need to add your ip address to access the database.

Testing:

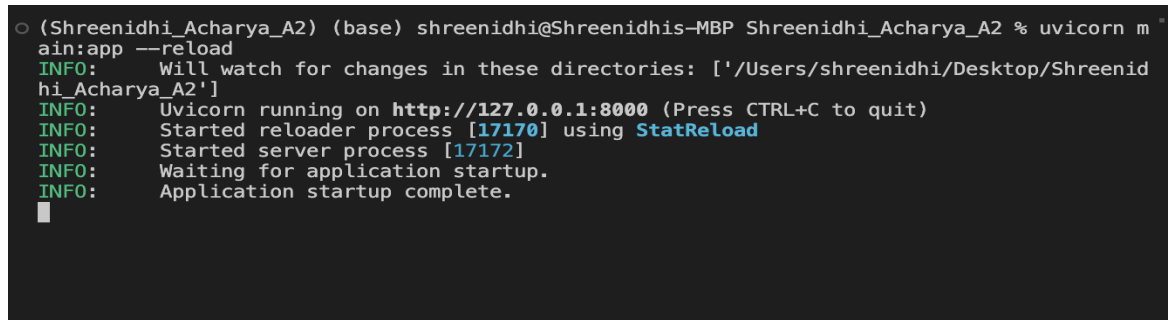
Run the below commands after folder extraction to deploy the application from your environment.

5. Open the folder "**Shreenidhi_Acharya_A2**" in visual studio code or pycharm.
6. Run the command "**python -m venv .**" in a terminal (**CTRL+~**) to create a virtual environment for this application.
Note: If you get error in the above command, you can use **python3 -m venv .**
7. Run the command "**source bin/activate**" to activate the created environment.
8. Run the command "**pip install -r requirements.txt**" to install all dependencies.
Note: if you encounter error with pip command use "**pip3 install -r requirements.txt**" instead.
9. Open two terminals. One for running the server side and the other for running the client side.



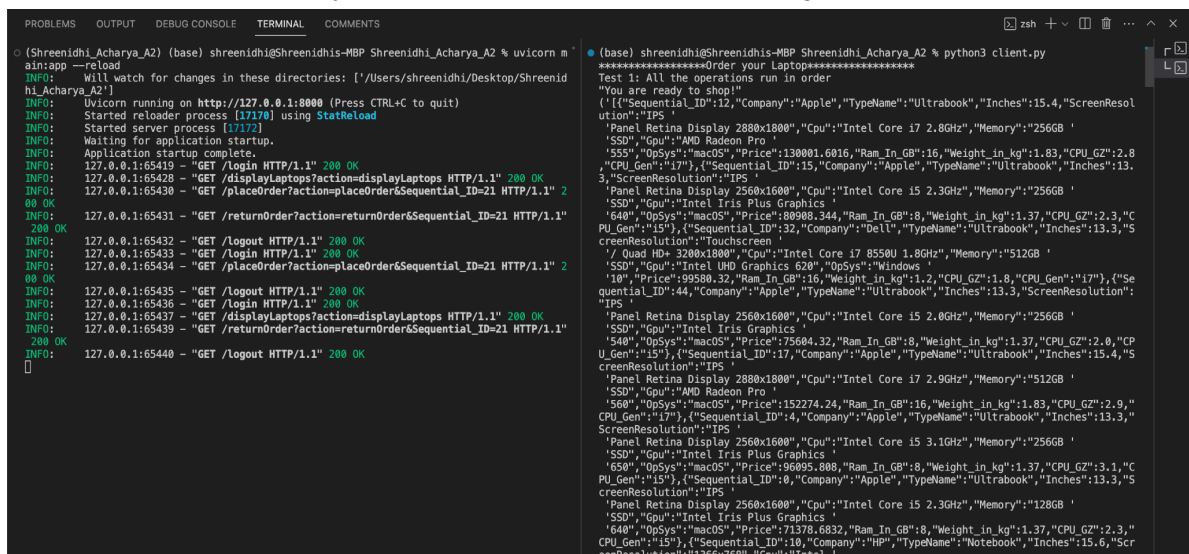
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
(Shreenidhi_Acharya_A2) (base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 %
(base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 %
```

10. Run the command "**uvicorn main:app --reload**" to start the server on the first terminal.



```
(Shreenidhi_Acharya_A2) (base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 % uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['/Users/shreenidhi/Desktop/Shreenidhi_Acharya_A2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [17170] using StatReload
INFO: Started server process [17172]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

11. Run the command "**python client.py**" on the second terminal after the server is running. On the second terminal you will see all the test cases running at once.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
(Shreenidhi_Acharya_A2) (base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 % uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['/Users/shreenidhi/Desktop/Shreenidhi_Acharya_A2']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [17170] using StatReload
INFO: Started server process [17172]
INFO: Waiting for application startup.
INFO: Application startup complete.
127.0.0.1:65431 - "GET / HTTP/1.1" 200 OK
127.0.0.1:65432 - "GET /login HTTP/1.1" 200 OK
127.0.0.1:65428 - "GET /displayLaptops?action=displayLaptops HTTP/1.1" 200 OK
127.0.0.1:65430 - "GET /placeOrder?action=placeOrder&Sequential_ID=21 HTTP/1.1" 200 OK
127.0.0.1:65431 - "GET /returnOrder?action=returnOrder&Sequential_ID=21 HTTP/1.1" 200 OK
127.0.0.1:65432 - "GET /logout HTTP/1.1" 200 OK
127.0.0.1:65433 - "GET /login HTTP/1.1" 200 OK
127.0.0.1:65434 - "GET /placeOrder?action=placeOrder&Sequential_ID=21 HTTP/1.1" 200 OK
127.0.0.1:65435 - "GET /logout HTTP/1.1" 200 OK
127.0.0.1:65436 - "GET /login HTTP/1.1" 200 OK
127.0.0.1:65437 - "GET /displayLaptops?action=displayLaptops HTTP/1.1" 200 OK
127.0.0.1:65439 - "GET /returnOrder?action=returnOrder&Sequential_ID=21 HTTP/1.1" 200 OK
127.0.0.1:65440 - "GET /logout HTTP/1.1" 200 OK

(base) shreenidhi@Shreenidhis-MBP Shreenidhi_Acharya_A2 % python3 client.py
*****Order your Laptop*****
Test 1: All the operations run in order
('Sequential_ID':12,'Company':'Apple','Type':'Ultrabook','Inches':15.4,'ScreenResolution':'IPS')
'Panel Retina Display 2880x1800','Cpu':'Intel Core i7 2.8GHz','Memory':'256GB'
'SSD','Gpu':'AMD Radeon Pro'
'555','OpSys':'macOS','Price':130001.6016,'Ram_In_GB':16,'Weight_in_kg':1.83,'CPU_G2':2.8
'PU_Gen':'i7'),('Sequential_ID':32,'Company':'Apple','Type':'Ultrabook','Inches':13.3,'ScreenResolution':'IPS')
'Panel Retina Display 2560x1600','Cpu':'Intel Core i5 2.3GHz','Memory':'256GB'
'SSD','Gpu':'Intel Iris Plus Graphics'
'648','OpSys':'macOS','Price':80988.344,'Ram_In_GB':8,'Weight_in_kg':1.37,'CPU_G2':2.3,'CPU_Gen':'i5'),('Sequential_ID':32,'Company':'Dell','Type':'Ultrabook','Inches':13.3,'ScreenResolution':'Touchscreen')
'Quad HD+ 3200x1800','Cpu':'Intel Core i7 8550U 1.8GHz','Memory':'512GB'
'SSD','Gpu':'Intel UHD Graphics 620','OpSys':'Windows'
'10','Price':99580.32,'Ram_In_GB':16,'Weight_in_kg':1.2,'CPU_G2':1.8,'CPU_Gen':'i7'),('Sequential_ID':44,'Company':'Apple','Type':'Ultrabook','Inches':13.3,'ScreenResolution':'IPS')
'Panel Retina Display 2560x1600','Cpu':'Intel Core i5 2.0GHz','Memory':'256GB'
'SSD','Gpu':'Intel Iris Graphics'
'560','OpSys':'macOS','Price':75684.32,'Ram_In_GB':8,'Weight_in_kg':1.37,'CPU_G2':2.0,'CPU_Gen':'i5'),('Sequential_ID':17,'Company':'Apple','Type':'Ultrabook','Inches':15.4,'ScreenResolution':'IPS')
'Panel Retina Display 2880x1800','Cpu':'Intel Core i7 2.9GHz','Memory':'512GB'
'SSD','Gpu':'AMD Radeon Pro'
'560','OpSys':'macOS','Price':152274.24,'Ram_In_GB':16,'Weight_in_kg':1.83,'CPU_G2':2.9,'CPU_Gen':'i7'),('Sequential_ID':14,'Company':'Apple','Type':'Ultrabook','Inches':13.3,'ScreenResolution':'IPS')
'Panel Retina Display 2560x1600','Cpu':'Intel Core i5 3.1GHz','Memory':'256GB'
'SSD','Gpu':'Intel Iris Plus Graphics'
'650','OpSys':'macOS','Price':9695.808,'Ram_In_GB':8,'Weight_in_kg':1.37,'CPU_G2':3.1,'CPU_Gen':'i5'),('Sequential_ID':8,'Company':'Apple','Type':'Ultrabook','Inches':13.3,'ScreenResolution':'IPS')
'Panel Retina Display 2560x1600','Cpu':'Intel Core i5 2.3GHz','Memory':'128GB'
'SSD','Gpu':'Intel Iris Plus Graphics'
'648','OpSys':'macOS','Price':71378.6832,'Ram_In_GB':8,'Weight_in_kg':1.37,'CPU_G2':2.3,'CPU_Gen':'i5'),('Sequential_ID':10,'Company':'HP','Type':'Notebook','Inches':15.6,'ScreenResolution':'1366x768','Cpu':'Intel'
```