Open Platform Software
開放平台軟體

Assignment #7: Chat Room (3)

Semester: 106-2

Offered: 2018

Dr. Bo-Hao Chen

Department of Computer Science and Engineering

Yuan Ze University, Taoyuan, Taiwan

bhchen at saturn.yzu.edu.tw

What is MySQL?

- MySQL is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation.

- Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access.

What is MongoDB?

- MongoDB is an open-source, non-relational database developed by MongoDB, Inc. MongoDB stores data as documents in a binary representation called BSON (Binary JSON).

- Related information is stored together for fast query access through the MongoDB query language.

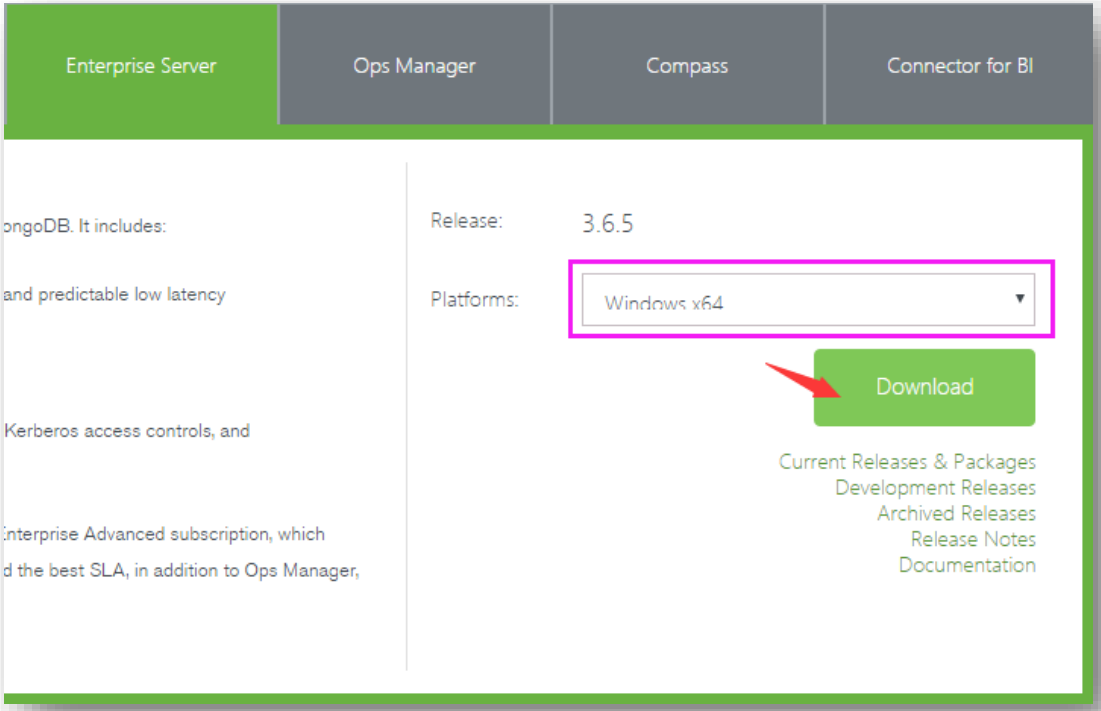
Terminology and Concepts

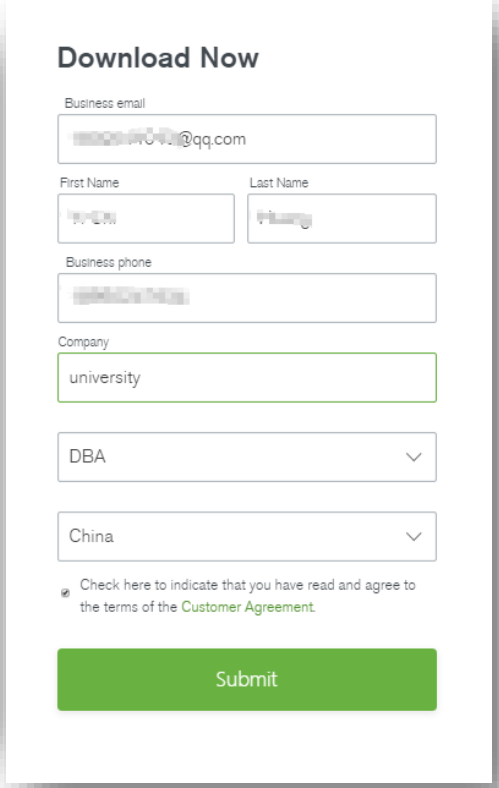
- Many concepts in MySQL have close analogs in MongoDB. The table below outlines the common concepts across MySQL and MongoDB

MySQL	MongoDB
ACID Transactions	ACID Transactions*
Table	Collection
Row	Document
Column	Field
Secondary Index	Secondary Index
JOINs	Embedded documents $lookup & $graphLookup
GROUP_BY	Aggregation Pipeline

Install MongoDB Enterprise on Windows

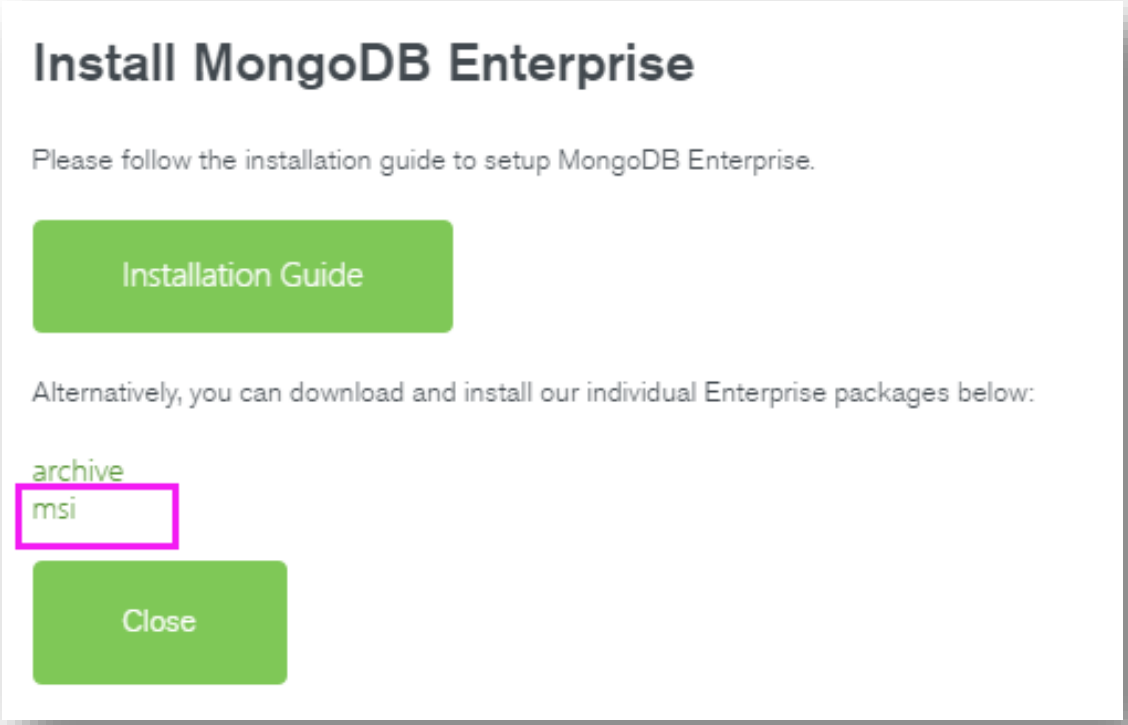
- Download MongoDB Enterprise Server
  - https://www.mongodb.com/download-center#enterprise





Download MongoDB Enterprise Server

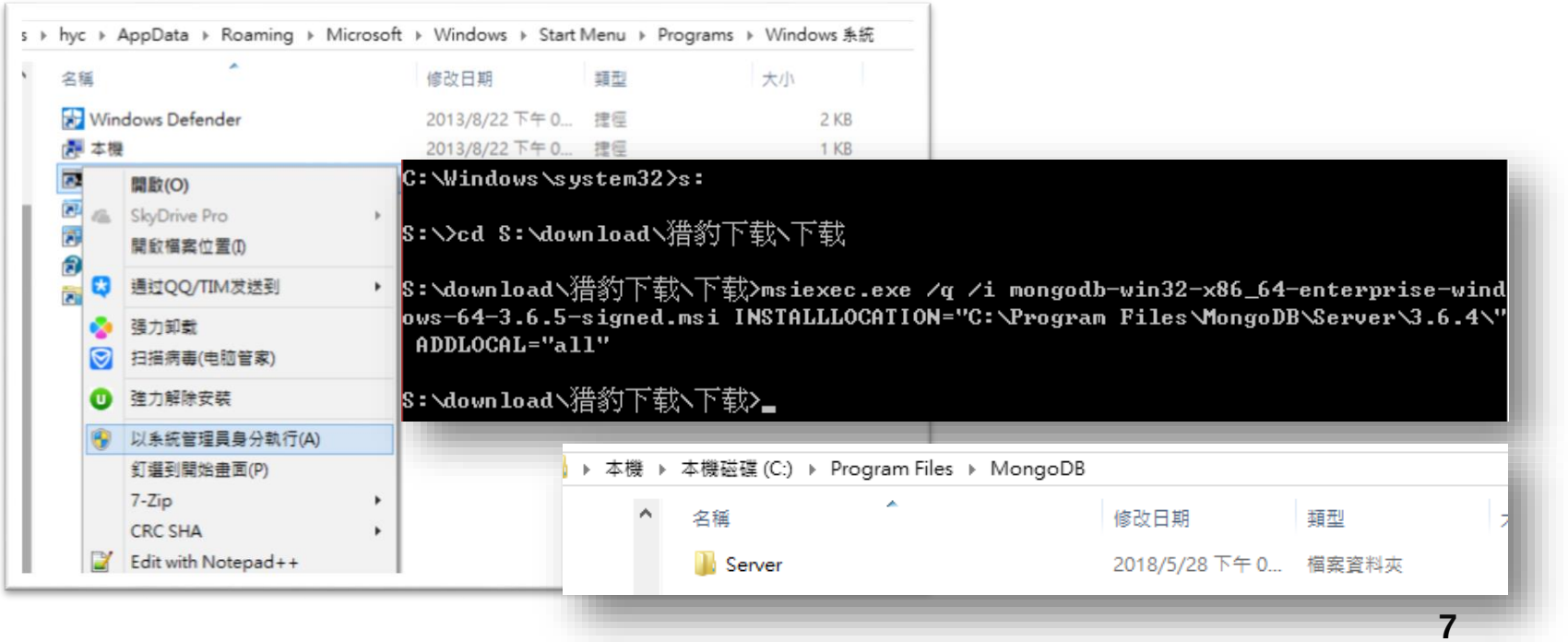
- http://gofile.me/3trS5/b8NioPPQq
- http://gofile.me/3trS5/uqgeYZwn4



Install MongoDB Enterprise

- Change to the directory containing the .msi installation binary of your choice and invoke:

```
msiexec.exe /q /i mongodb-win32-x86_64-enterprise-windows-64-3.6.5-signed.msi INSTALLLOCATION="C:\Program Files\MongoDB\Server\3.6.4\" ADDLOCAL="all"
```



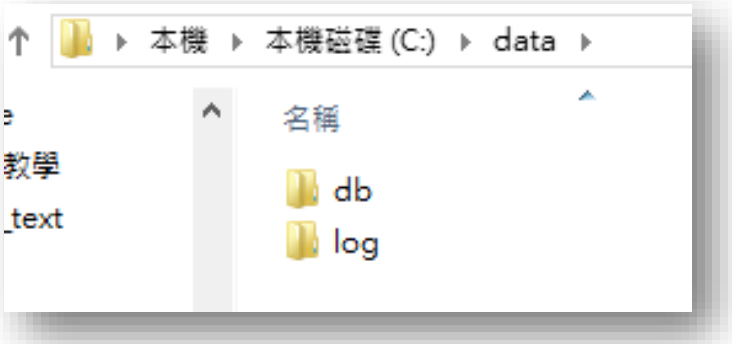
Set up the MongoDB environment

- MongoDB requires a data directory to store all data.
  - MongoDB's default data directory path is the absolute path \data\db on the drive from which you start MongoDB.
  - Create this folder by running the following command in a Command Prompt:

```
md \data\db
```

- MongoDB requires a log directory to record logs.

```
md \data\log
```



Start MongoDB

- Add system path

```
C:\Users\hyc>mongod
```

- or

```
C:\Users\hyc>"C:\Program Files\MongoDB\Server\3.6.4\bin\mongod.exe"
```

```
C:\Users\hyc>mongod
2018-05-28T16:11:13.255+0800 I CONTROL  [initandlisten] MongoDB starting : pid=9
152 port=27017 dbpath=C:\data\db\ 64-bit host=WIN-L5EQM1SC8NB
2018-05-28T16:11:13.256+0800 I CONTROL  [initandlisten] targetMinOS: Windows 7/W
```

Verify that MongoDB has started successfully

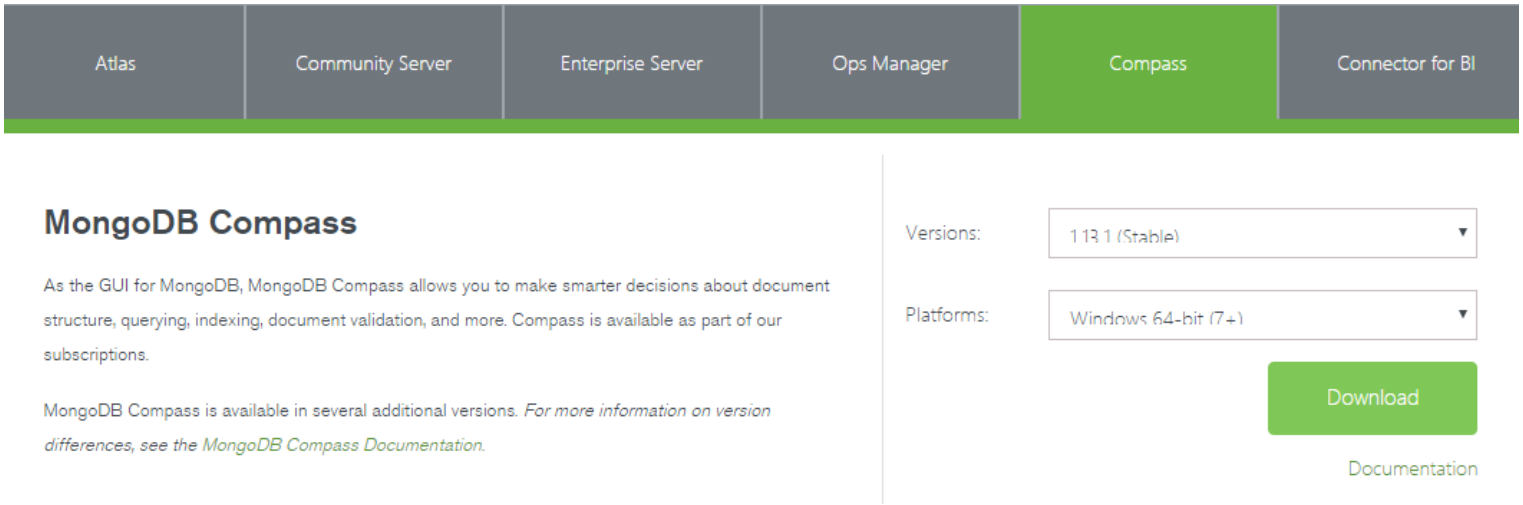
- Verify that MongoDB has started successfully by checking the process output for the following line:
  - 1.

```
iagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-05-28T16:11:13.938+0800 I NETWORK  [initandlisten] waiting for connections
on port 27017
```

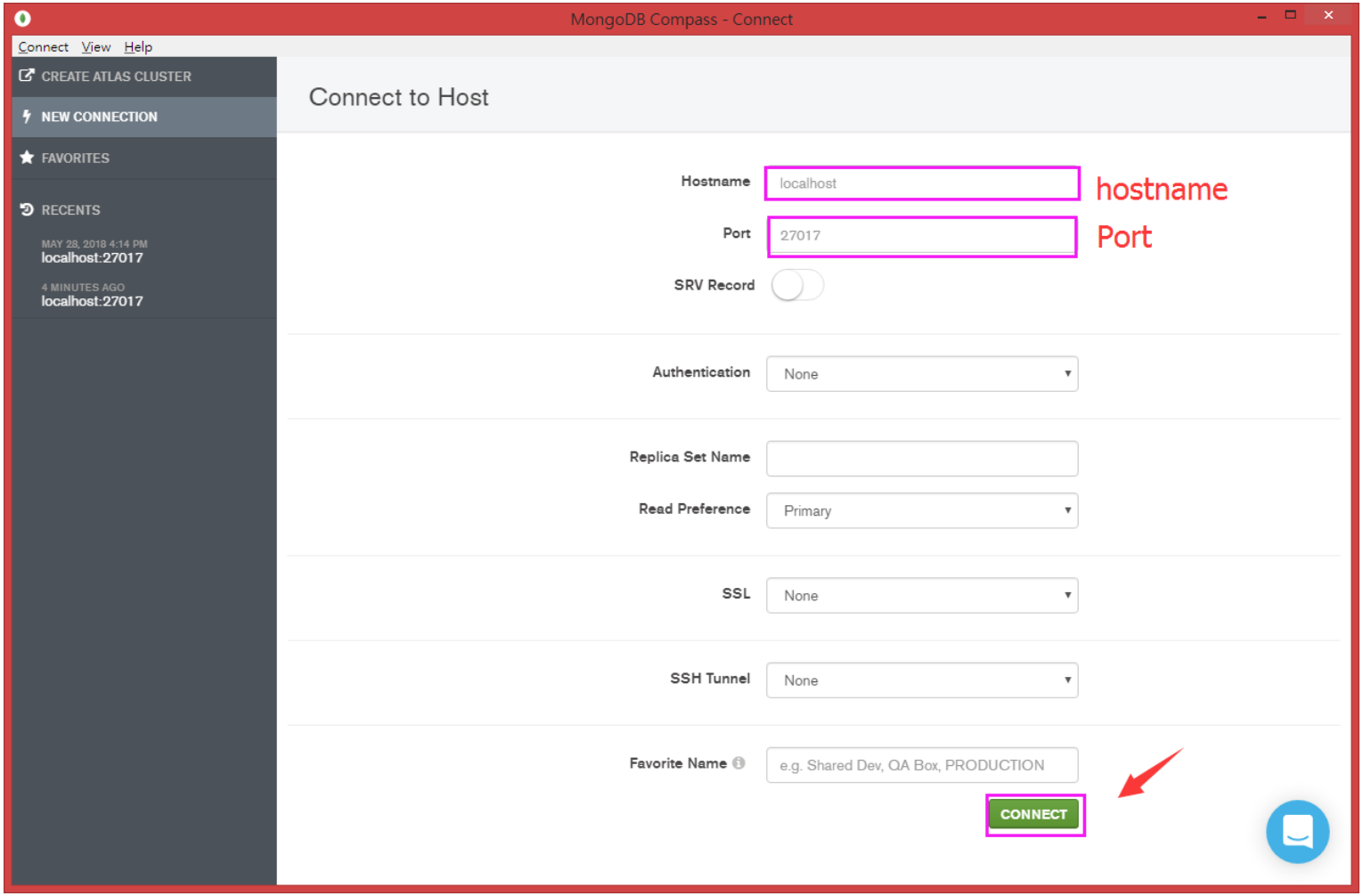
  - 2.

Visualize and explore

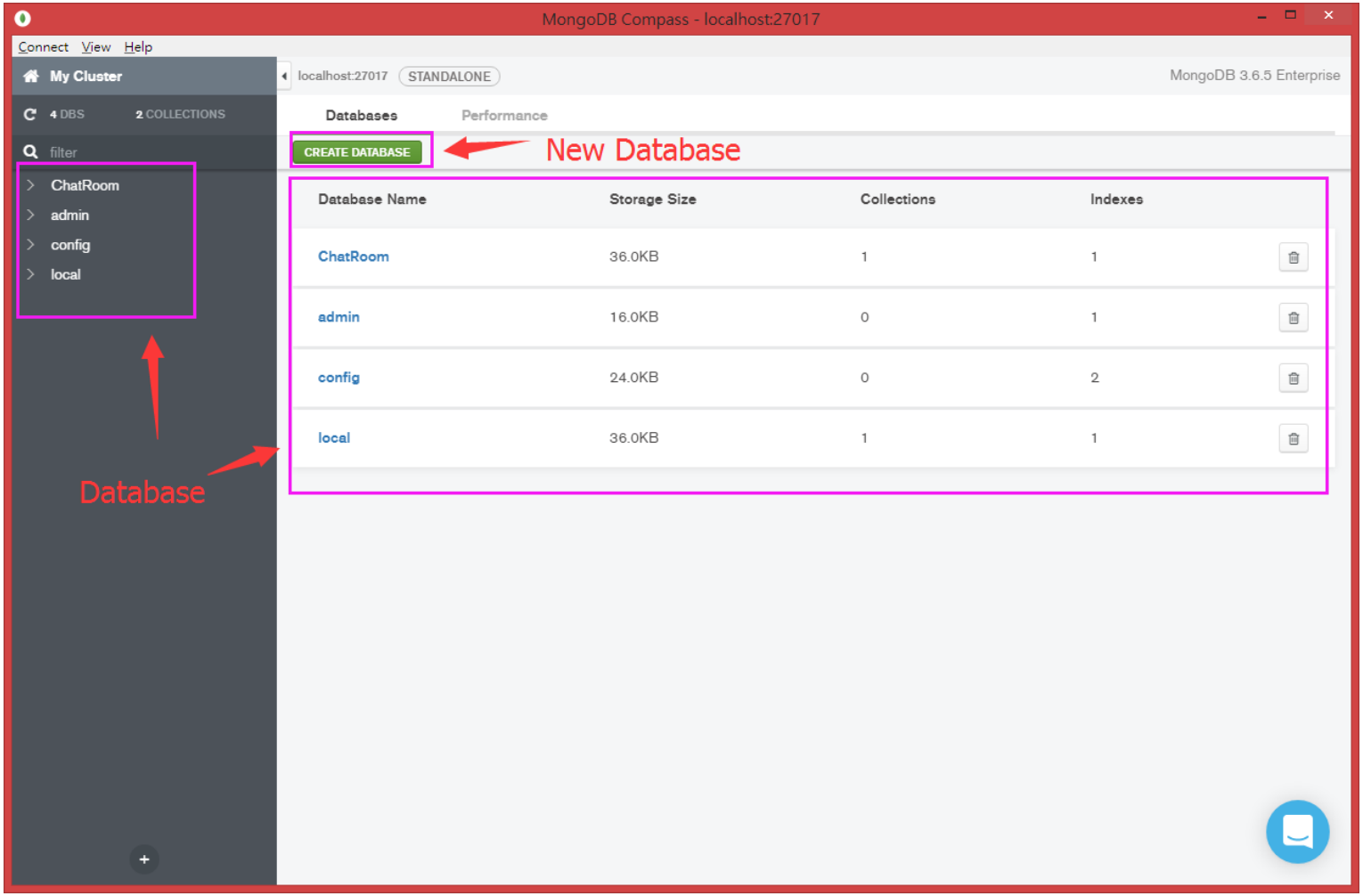
- The GUI for MongoDB, Visually explore your data.
- View and optimize your query performance.
  - https://www.mongodb.com/download-center?jmp=hero#compass



Visualize and explore



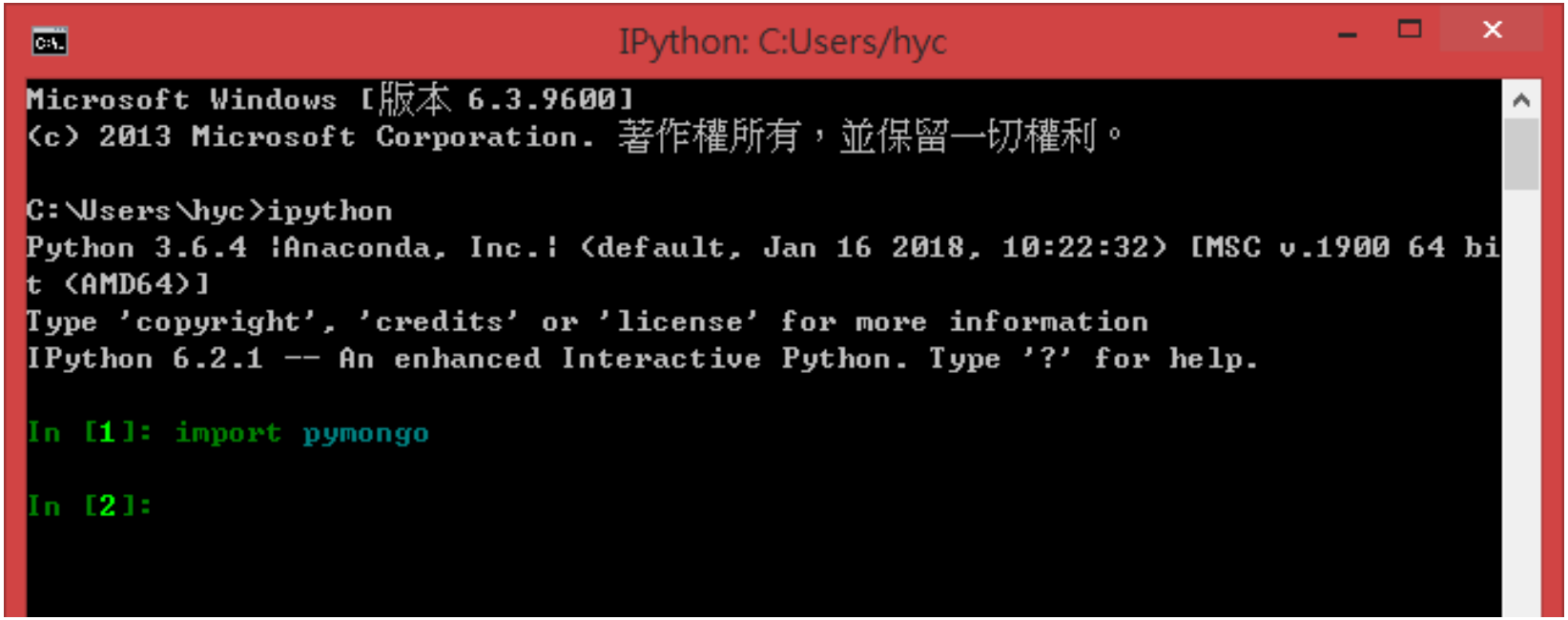
Visualize and explore



Install pymongo

- Pymongo is a library used to manipulate MongoDB in python

```
C:\Users\hyc> pip install pymongo
```



Pymongo: establish connection

```
from pymongo import MongoClient
client = MongoClient()
```

- or

```
client = MongoClient("localhost", 27017)
```

- or

```
client = MongoClient("mongodb://localhost:27017/")
```

Pymongo: database and collection

```
# connection
db = client["ChatRoom"]
collection = db["user"]

# test if connection success
print(collection.stats)
```

SQL	NOSQL
server	server
database	database
table	collection
row	dictionary

IPython: C:Users/hyc

```
In [6]: db = client["ChatRoom"]

In [7]: collection = db["user"]

In [8]: collection.stats
Out[8]: Collection(Database(MongoClient(host=['localhost:27017'], document_class
=dict, tz_aware=False, connect=True), 'ChatRoom'), 'user.stats')

In [9]:
```

Pymongo: insert

```
# insert one
collection.insert_one({'uname': 'E', 'upwd': 'E'})
# insert many
userList = []
userList.append({'uname': 'A', 'upwd': 'A'})
userList.append({'uname': 'B', 'upwd': 'B'})
userList.append({'uname': 'C', 'upwd': 'C'})
collection.insert_many(userList)
```

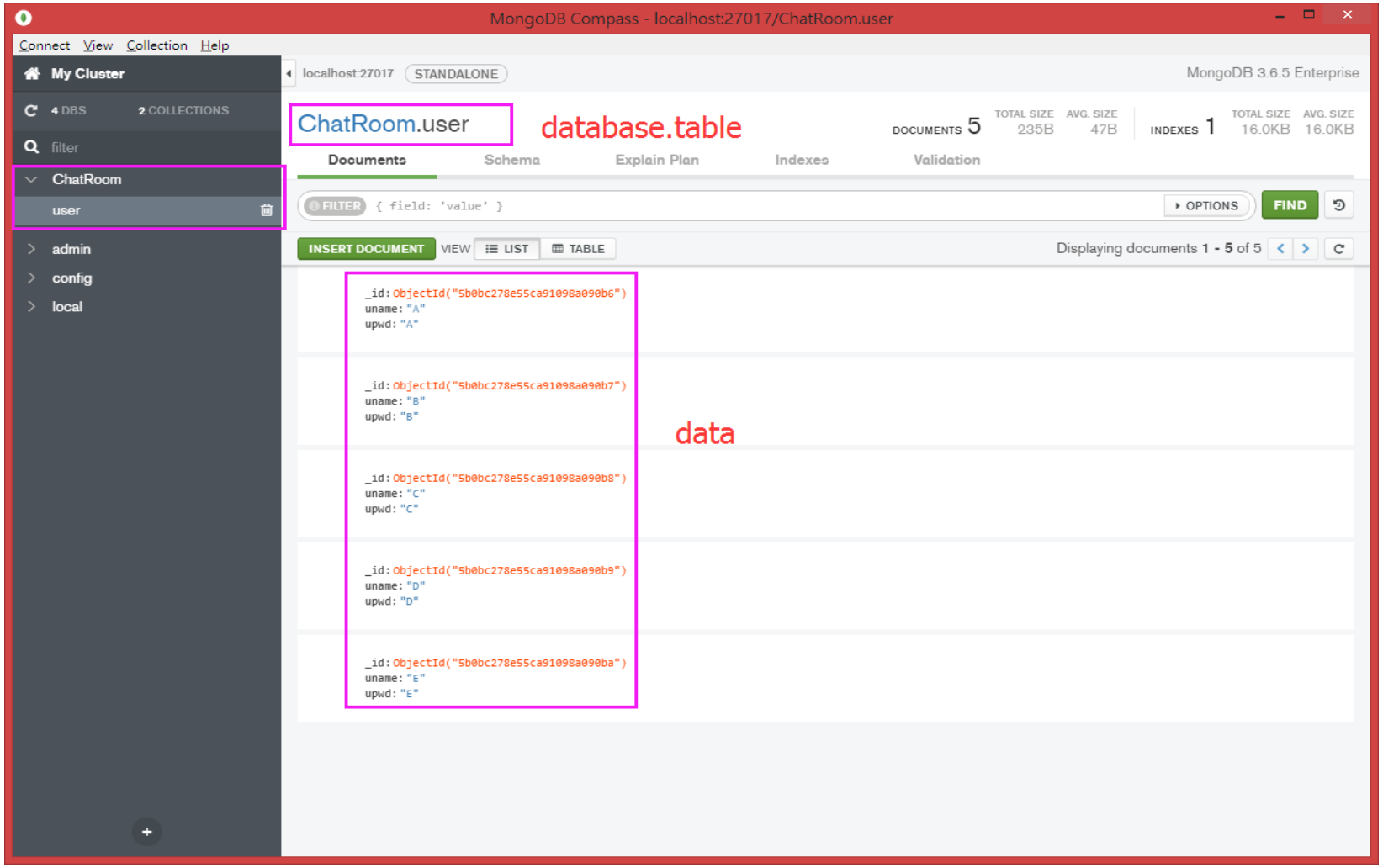
IPython: C:Users/hyc

```
In [14]: collection.insert_one({'uname': 'E', 'upwd': 'E'})
Out[14]: <pymongo.results.InsertOneResult at 0xf66042a888>

In [15]: userList = []
    ...: userList.append({'uname': 'A', 'upwd': 'A'})
    ...: userList.append({'uname': 'B', 'upwd': 'B'})
    ...: userList.append({'uname': 'C', 'upwd': 'C'})
    ...: collection.insert_many(userList)
    ...:
Out[15]: <pymongo.results.InsertManyResult at 0xf660670188>

In [16]:
```

Visualize


MongoDB Compass - localhost:27017/ChatRoom.user
Connect View Collection Help
My Cluster
4 DBS 2 COLLECTIONS
filter
ChatRoom
user
admin
config
local
localhost:27017 STANDALONE
MongoDB 3.6.5 Enterprise
ChatRoom.user
database.table
DOCUMENTS 5 TOTAL SIZE 235B AVG. SIZE 47B
INDEXES 1 TOTAL SIZE 16.0KB AVG. SIZE 16.0KB
Documents Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS
FIND
INSERT DOCUMENT
VIEW LIST TABLE
Displaying documents 1 - 5 of 5
_id: ObjectId("5b0bc278e55ca91098a090b6")
uname: "A"
upwd: "A"
_id: ObjectId("5b0bc278e55ca91098a090b7")
uname: "B"
upwd: "B"
_id: ObjectId("5b0bc278e55ca91098a090b8")
uname: "C"
upwd: "C"
_id: ObjectId("5b0bc278e55ca91098a090b9")
uname: "D"
upwd: "D"
_id: ObjectId("5b0bc278e55ca91098a090ba")
uname: "E"
upwd: "E"
data

Pymongo: query

```
# find_one() returns a single document matching a query (or None if
# there are no matches).
collection.find_one({'uname':'A'})
collection.find_one({'uname':'A','upwd':'A'})

# find by id
from bson.objectid import ObjectId
collection.find_one({'_id':ObjectId('5b0e6e4ae55ca90304d33773')})
```

```
In [27]: collection.find_one({'uname':'A'})
Out[27]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'
}

In [28]: collection.find_one({'uname':'A','upwd':'A'})
Out[28]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'
}

In [29]: from bson.objectid import ObjectId

In [30]: collection.find_one({'_id':ObjectId('5b0e6e4ae55ca90304d33773')})
Out[30]: {'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'
}
```

Pymongo: query

```
for ii in collection.find({'uname': 'A', 'upwd': 'A'}):
        print(ii)

print(collection.find().count())
```

IPython: C:Users/hyc

```
In [16]: for ii in collection.find({'uname': 'A', 'upwd': 'A'}):
    ...:     print(ii)
    ...:
{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}

In [17]: print(collection.find().count())
5
```

Pymongo: query

```
for ii in collection.find().sort('uname'):
        print(ii)

for ii in collection.find().sort('uname',pymongo.DESCENDING):
        print(ii)
```

```
In [18]: for ii in collection.find().sort('uname'):
    ...:     print(ii)
    ...:
{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'}

In [19]: for ii in collection.find().sort('uname',pymongo.DESCENDING):
    ...:     print(ii)
    ...:
{'_id': ObjectId('5b0e6e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'}
{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}
```

Pymongo: query

```
# query all data
cursor = collection.find({})
data = [d for d in cursor]
print(data)
```

IPython: C:Users/hyc

```
In [31]: cursor = collection.find({})

In [32]: data = [d for d in cursor]

In [33]: print(data)
[{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'A'}, {'_id
': ObjectId('5b0e6e4ae55ca90304d33774'), 'uname': 'B', 'upwd': 'B'}, {'_id': Obj
ectId('5b0e6e4ae55ca90304d33775'), 'uname': 'C', 'upwd': 'C'}, {'_id': ObjectId(
'5b0e6e4ae55ca90304d33776'), 'uname': 'D', 'upwd': 'D'}, {'_id': ObjectId('5b0e6
e4ae55ca90304d33777'), 'uname': 'E', 'upwd': 'E'}]

In [34]:
```

Pymongo: update

```
temp = collection.find_one({‘uname’ : ’A’})
temp[‘upwd’] = ‘K’

# update method 1
collection.save(temp)
# update method 2
temp2 = temp.copy()
collection.update(temp, temp2)
```

```
In [34]: temp = collection.find_one({"uname":"A"})

In [35]: temp['upwd']='K'

In [36]: collection.save(temp)
S:\software\anaconda\Scripts\ipython:1: DeprecationWarning: save is deprecated.
Use insert_one or replace_one instead
Out[36]: ObjectId('5b0e6e4ae55ca90304d33773')

In [37]: temp2 = temp.copy()
    ...: collection.update(temp, temp2)
    ...:
S:\software\anaconda\Scripts\ipython:2: DeprecationWarning: update is deprecated
. Use replace_one, update_one or update_many instead.
Out[37]: {'n': 1, 'nModified': 0, 'ok': 1.0, 'updatedExisting': True}

In [38]: print(collection.find_one({'uname':'A'}))
{'_id': ObjectId('5b0e6e4ae55ca90304d33773'), 'uname': 'A', 'upwd': 'K'}
```

Pymongo: delete

```
collection.remove({'uname':'E'})
collection.delete_one({'uname':'D'})
collection.delete_many({'uname':'C'})
print([d for d in collection.find({})])

# delete all data
collection.delete_many({})
```

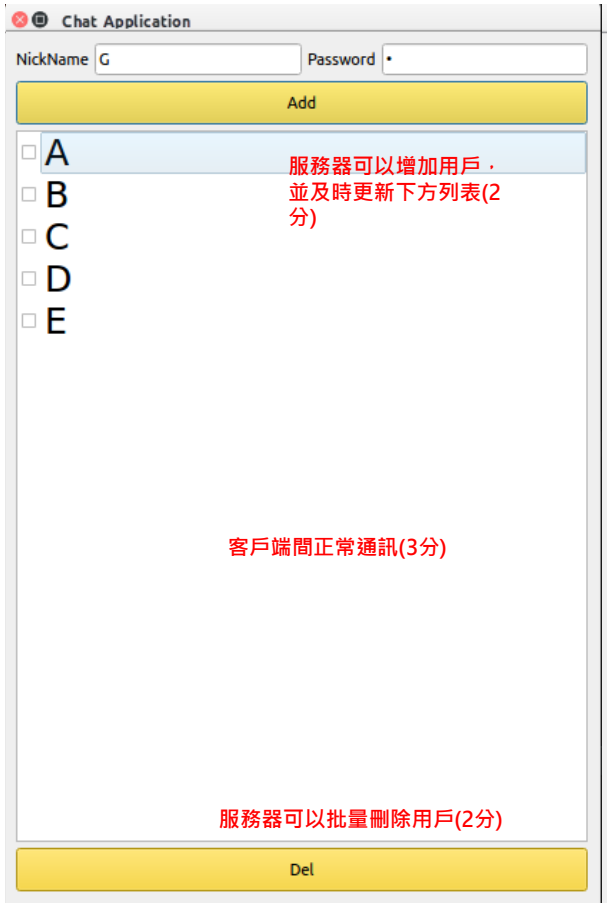
```
In [46]: collection.remove({'uname':'E'})
    ...: collection.delete_one({'uname':'D'})
    ...: collection.delete_many({'uname':'C'})
    ...: print([d for d in collection.find({})])
    ...:
S:\software\anaconda\Scripts\ipython:1: DeprecationWarning: remove is deprecated
. Use delete_one or delete_many instead.
[{'_id': ObjectId('5b0e9f12e55ca90f58a8fbed'), 'uname': 'A', 'upwd': 'K'}, {'_id
': ObjectId('5b0e9f12e55ca90f58a8fbee'), 'uname': 'B', 'upwd': 'B'}]

In [47]: collection.delete_many({})
Out[47]: <pymongo.results.DeleteResult at 0xf660940bc8>

In [48]: print([d for d in collection.find({})])
[]
```

Assignment

服務器端介面(1分)



服務器可以增加用戶，並及時更新下方列表(2分)

客戶端間正常通訊(3分)

服務器可以批量刪除用戶(2分)

客戶端端介面(1分)

客戶端及時顯示在綫人數(2分)

密碼顯示為點(1分)

客戶端可修改密碼並可成功用新密碼登錄(2分)

用戶離開通知，並更新在綫人數(2分)

使用mongodb作為數據庫(3分)

上週完成的加2.5分