

Name of Student: Pushkar Sane		
Roll Number: 45		Lab Assignment Number: 3
Title of Lab Assignment: To study MongoDB architecture and implement CRUD operations.		
DOP: 06-09-2024		DOS: 06-09-2024
CO Mapped: CO3	PO Mapped: PO1, PO2, PO3, PO4, PO5, PSO1	Signature:

Practical No. 3**Aim:**

To study MongoDB architecture and implement CRUD (Create, Read, Update, Delete) operations along with following:

1. Installation
2. Query the Sample Database using MongoDB querying commands:
 - a. Insert Document
 - b. Query Document
 - c. Indexing

Theory:

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and functions which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

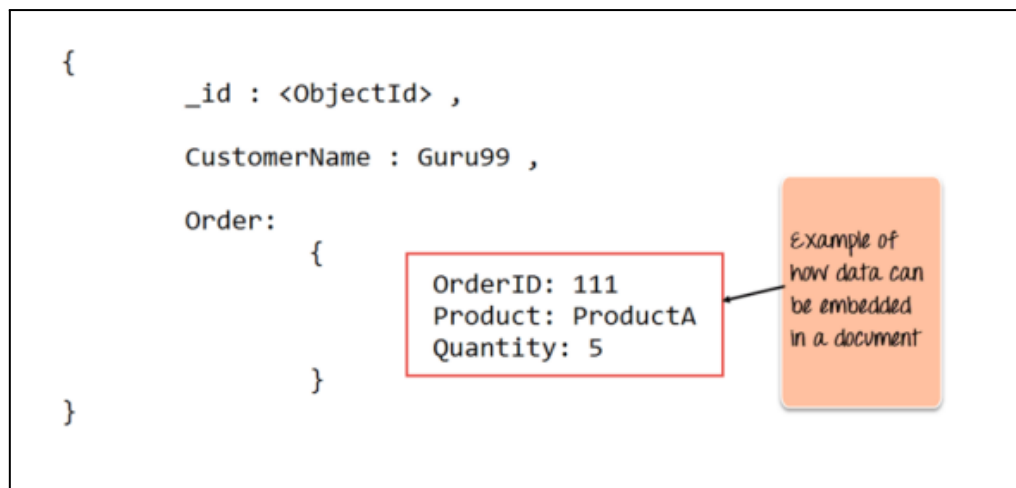
MongoDB Features:

1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) don't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability - The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database.

MongoDB Example -

The below example shows how a document can be modelled in MongoDB.

1. The **_id** field is added by MongoDB to uniquely identify the document in the collection.
2. What you can note is that the Order Data (OrderID, Product, and Quantity) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modelled in MongoDB.

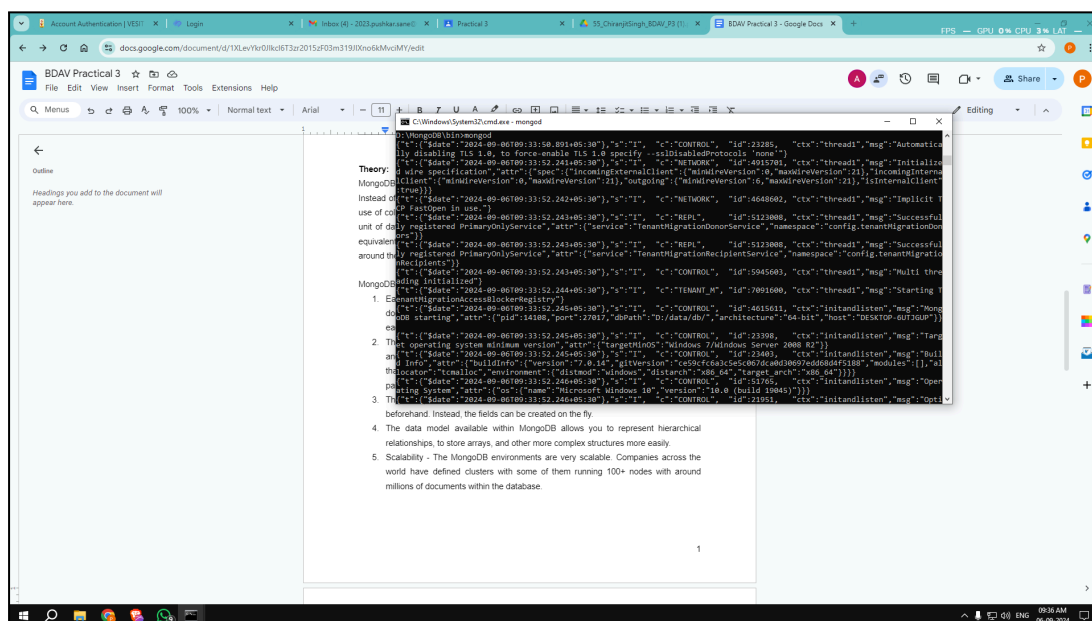


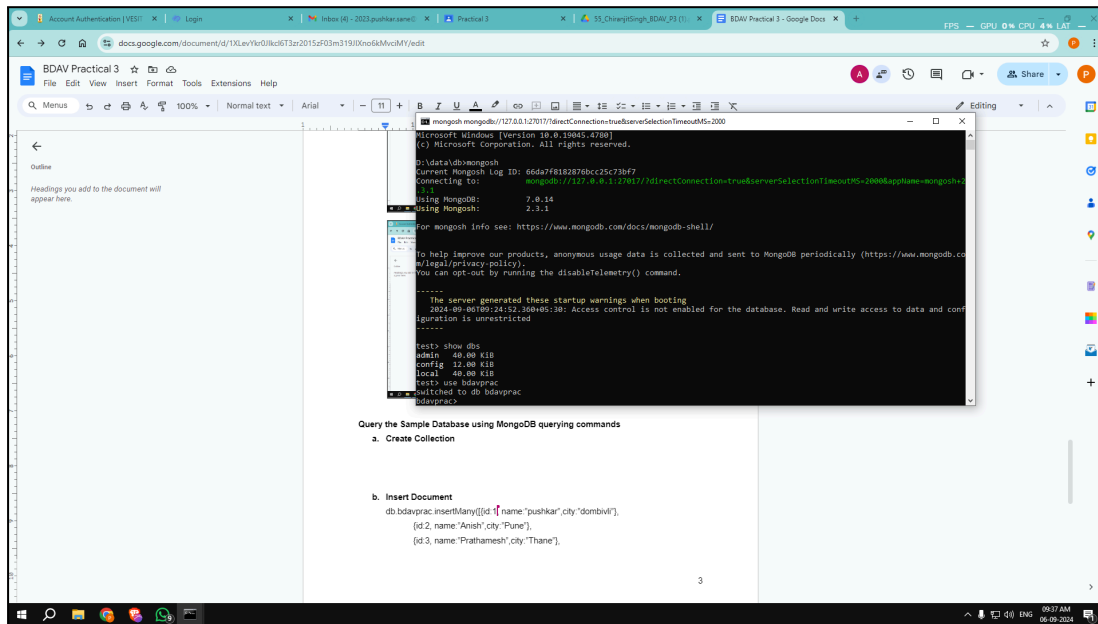
Code:

Database Creation

```
show dbs
```

use bdavprac

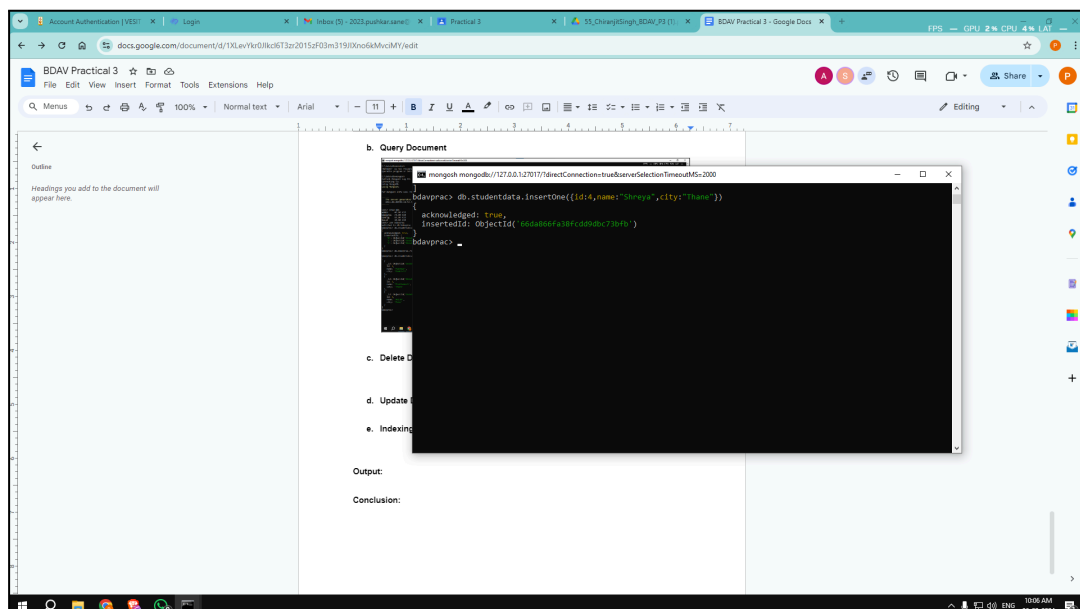




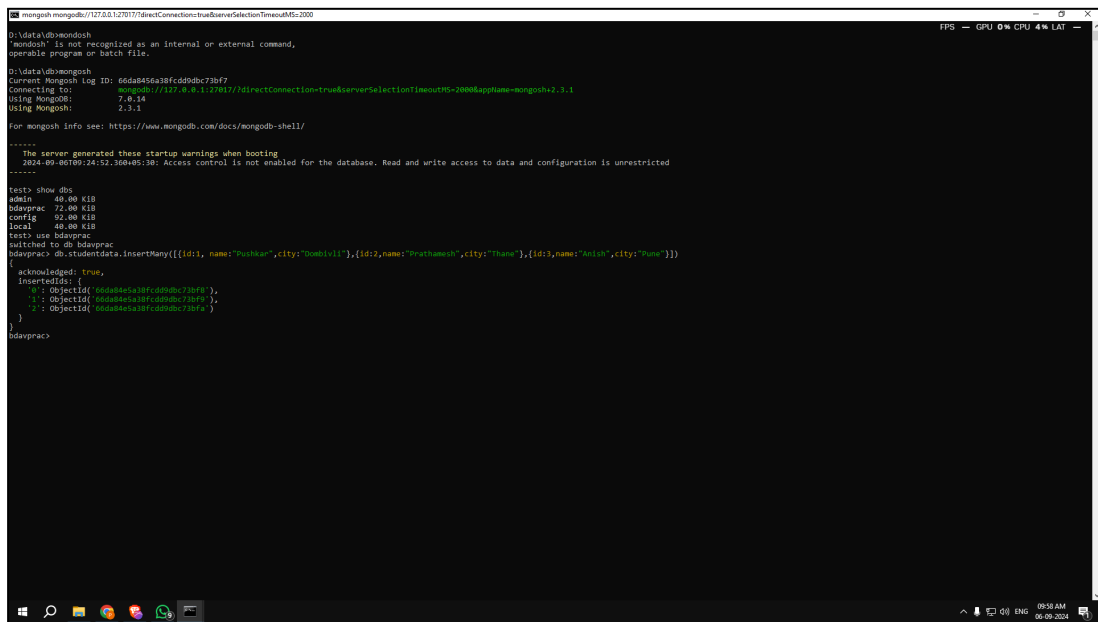
Query the Sample Database using MongoDB querying commands

a. Insert Document

```
db.studentdata.insertMany([
  {id:1, name:"pushkar",city:"dombivli"},
  {id:2, name:"Anish",city:"Pune"},
  {id:3, name:"Prathamesh",city:"Thane"}])
```



db.studentdata.insertOne({id:4,name:"Shreya",city:"Thane"})

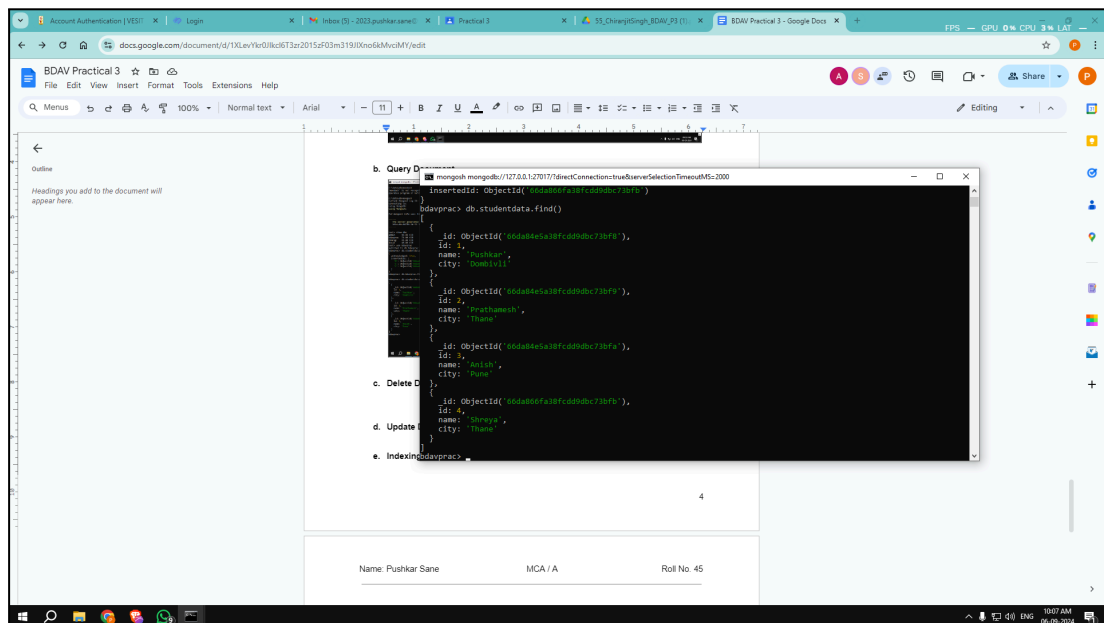


```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
D:\data\db>mongosh
'mongosh' is not recognized as an internal or external command,
operable program or batch file.
D:\data\db>mongosh
Current Mongosh Log ID: 66da8456a38fcd9dbc73bf7
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.1
Using Mongosh:
2.3.1
For mongosh info see: https://www.mongodb.com/docs/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-09-09T09:24:52.306+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

tests show dbs
admin 48.00 KiB
bdavprac 72.00 KiB
config 92.00 KiB
local 48.00 KiB
tests use bdavprac
switched to db bdavprac
bdavprac> db.studentdata.insertMany([({id:1, name:"Pushkar",city:"Dombivli"},{id:2,name:"Prathamesh",city:"Thane"},{id:3,name:"Anish",city:"Pune"})])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66da8456a38fcd9dbc73bf7'),
    '1': ObjectId('66da8456a38fcd9dbc73bf7'),
    '2': ObjectId('66da8456a38fcd9dbc73bf7')
  }
}
bdavprac>
```

b. Query Document



b. Query Document

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
bdavprac> db.studentdata.find()
{
  '_id': ObjectId('66da8456a38fcd9dbc73bf7'),
  'id': 1,
  'name': 'Pushkar',
  'city': 'Dombivli'
},
{
  '_id': ObjectId('66da8456a38fcd9dbc73bf7'),
  'id': 2,
  'name': 'Prathamesh',
  'city': 'Thane'
},
{
  '_id': ObjectId('66da8456a38fcd9dbc73bf7'),
  'id': 3,
  'name': 'Anish',
  'city': 'Pune'
},
{
  '_id': ObjectId('66da8456a38fcd9dbc73bf7'),
  'id': 4,
  'name': 'Shreya',
  'city': 'Thane'
}
```

c. Delete Document

```
bdavprac> db.studentdata.deleteOne({id:4})
{
  acknowledged: true,
  deletedCount: 1
}
```

d. Update Document

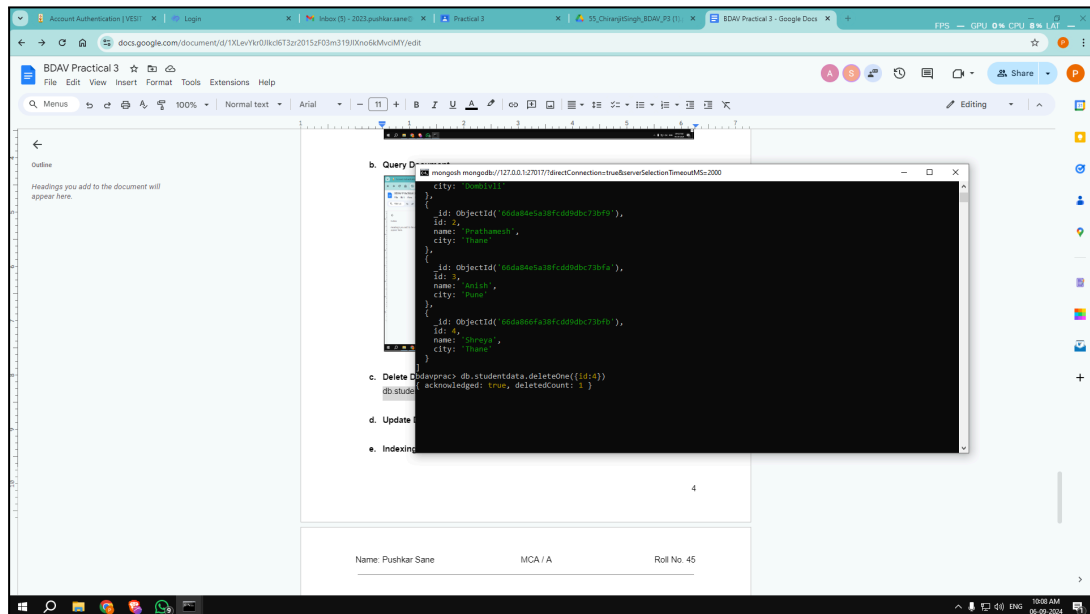
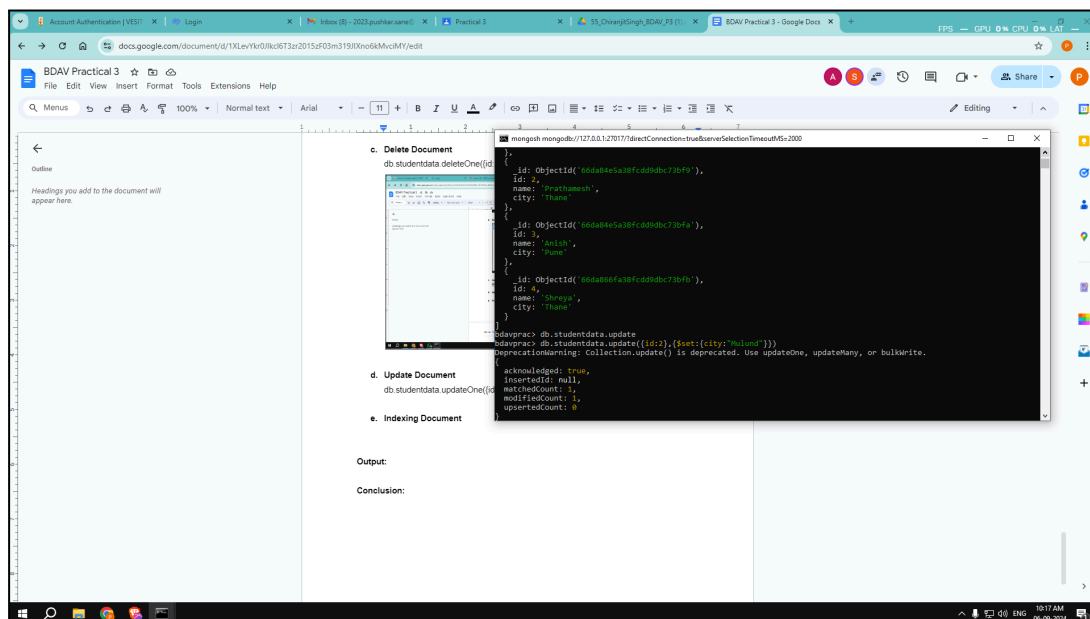
```
bdavprac> db.studentdata.updateOne({id:4}, {name:"Shreya",city:"Thane"})
{
  acknowledged: true,
  insertedId: ObjectId('66da8456a38fcd9dbc73bf7'),
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

e. Indexing

```
bdavprac> db.studentdata.createIndex({id:1})
{
  ok: 1,
  msg: "Index created"
}
```

4

Name: Pushkar Sane MCA / A Roll No 45

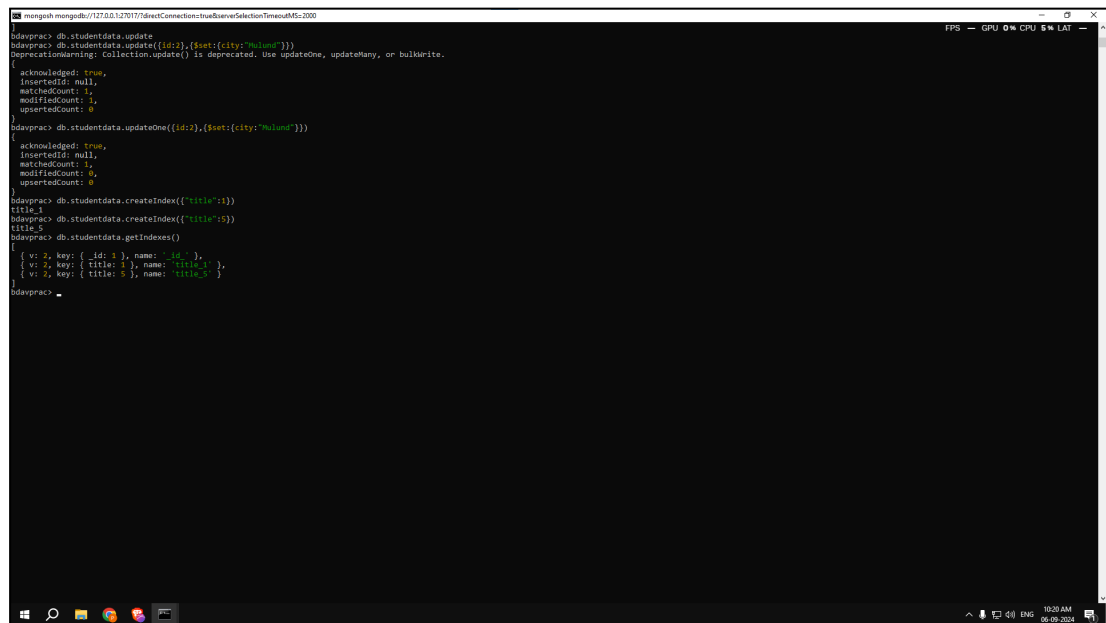
c. Delete Document`db.studentdata.deleteOne({id:4})`**d. Update Document**`db.studentdata.updateOne({id:2},{ $set:{city:"Mulund"}})`

e. Indexing Document

```
db.studentdata.createIndex({"title":1})
```

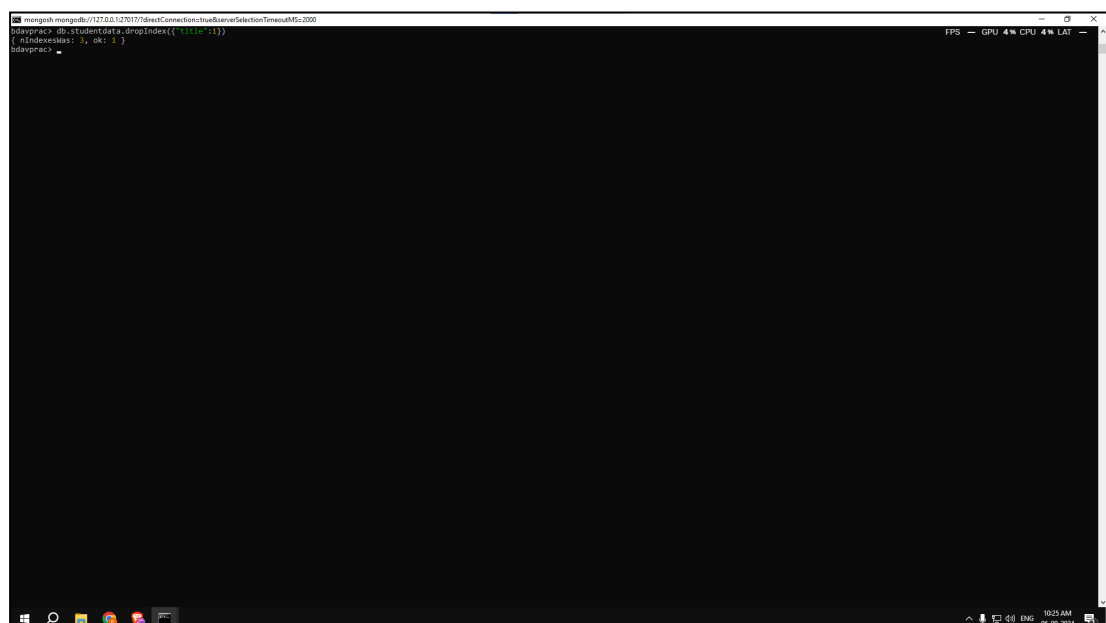
```
db.studentdata.createIndex({"title":6})
```

```
db.studentdata.getIndexes()
```

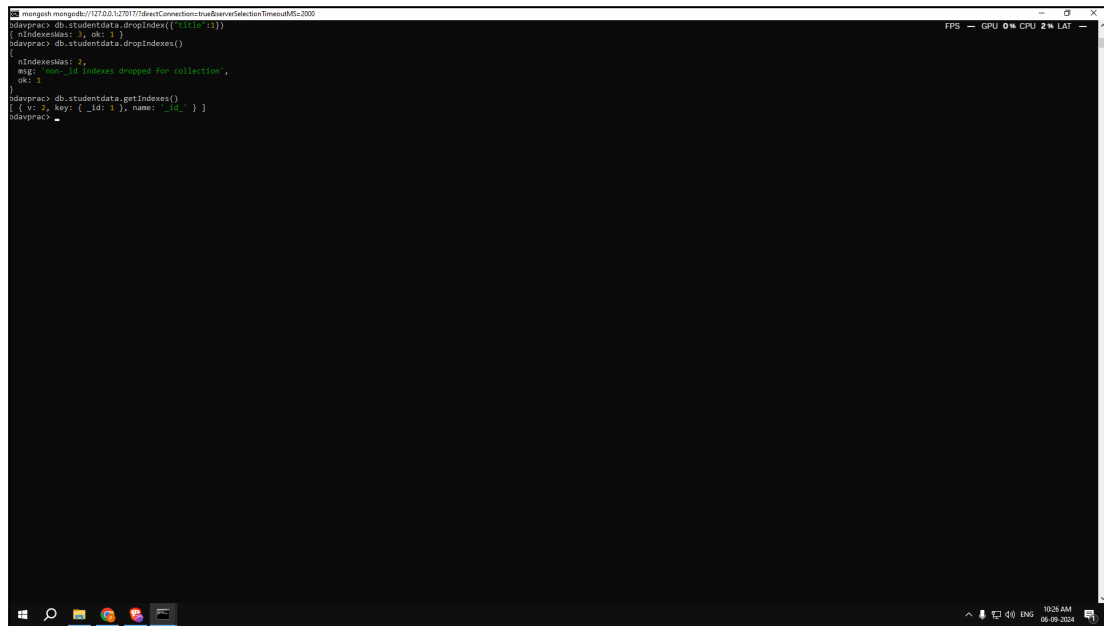


```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
> use studentdata
> db.studentdata.update(
  { _id: 1 }, { $set: { city: "Mumbai" } })
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.studentdata.updateOne(
  { _id: 2 }, { $set: { city: "Mumbai" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
> db.studentdata.createIndex({"title":1})
title_1
> db.studentdata.createIndex({"title":5})
title_5
> db.studentdata.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { title: 5 }, name: 'title_5' }
]
```

```
db.studentdata.dropIndex({"title":1})
```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
> use studentdata
> db.studentdata.dropIndex({"title":1})
{ "ok": 1, "errmsg": "", "code": 0 }
>
```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
> use studentdata
> db.studentdata.dropIndex({'_id':1})
{ nIndexesWas: 1, ok: 1 }
> db.studentdata.dropIndexes()
{ nIndexesWas: 1,
  msg: 'non-_id indexes dropped for collection',
  ok: 1 }
> db.studentdata.getIndexes()
[ { v: 1, key: { '_id': 1 }, name: '_id' } ]
>
```

Conclusion:

Successfully implemented CRUD operations and querying in MongoDB