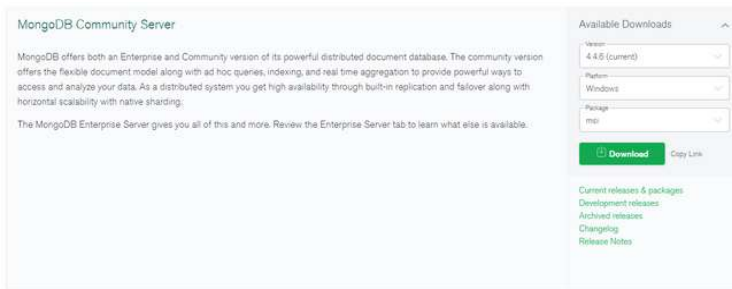

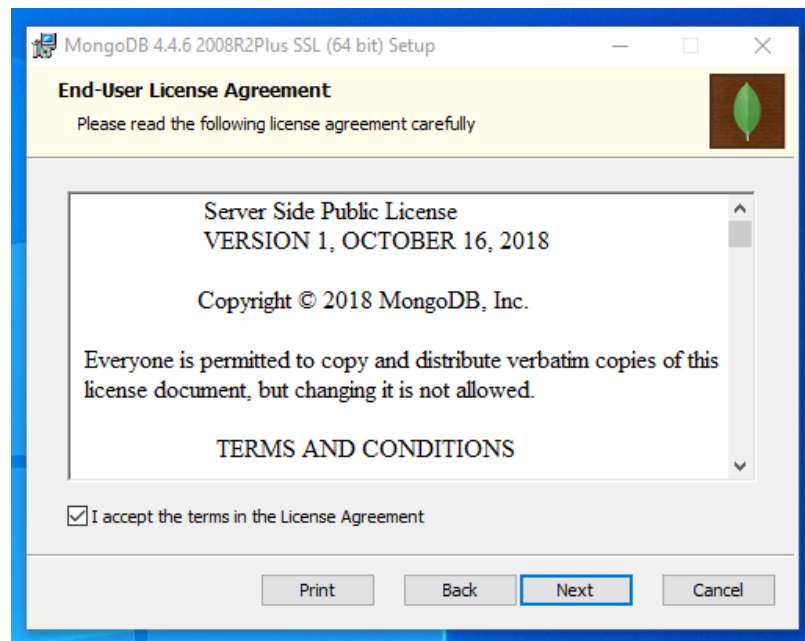


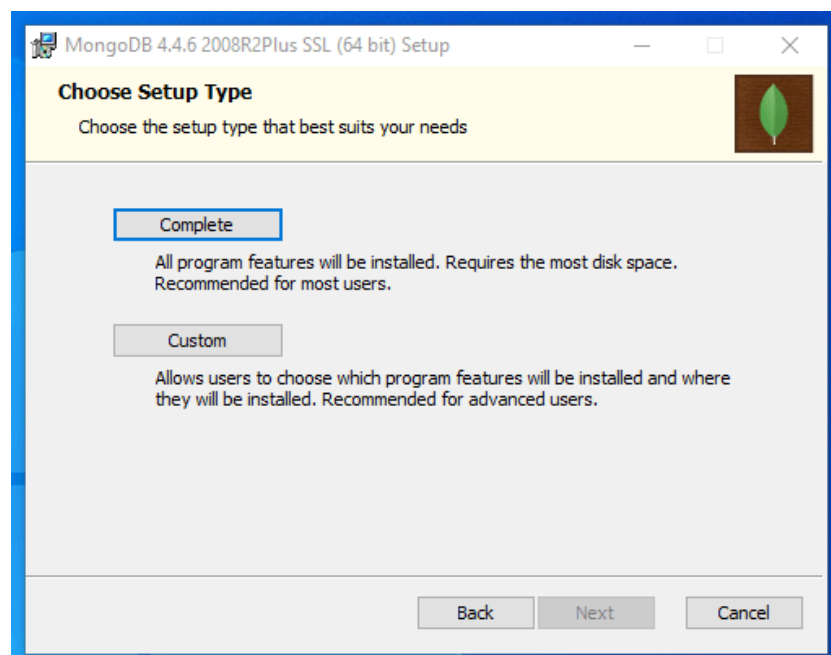
Experiment No: 5

Topic:	To install and configure MongoDB to execute NoSQL commands.
Prerequisite:	Basic knowledge of database, SQL, NoSQL
Mapping With COs:	CSL702.5
Objective:	Able to collect, manage, store, query and analyze various forms of Big Data.
Outcomes:	Students will be able to install and manage big data using NoSQL database (Mongodb).
Instructions:	This experiment is a compulsory experiment. All the students are required to perform this experiment individually.
Deliverables:	<p>Submission:</p> <p>1. Installation steps snapshots.</p>  

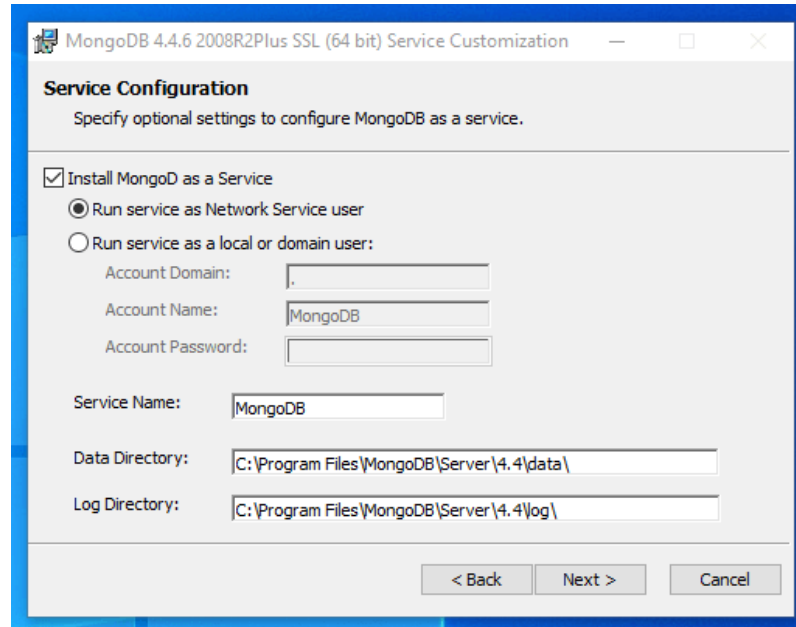
Click **Next** on the initial page to continue.



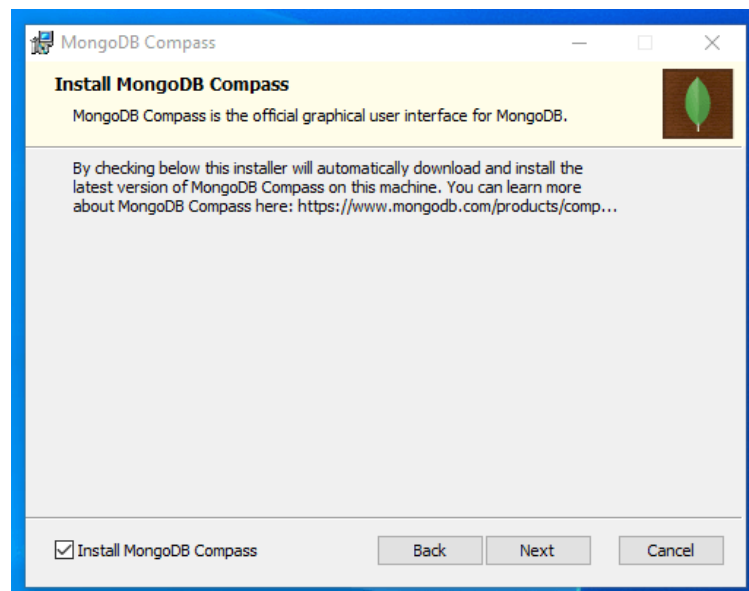
Click **Next** to continue.



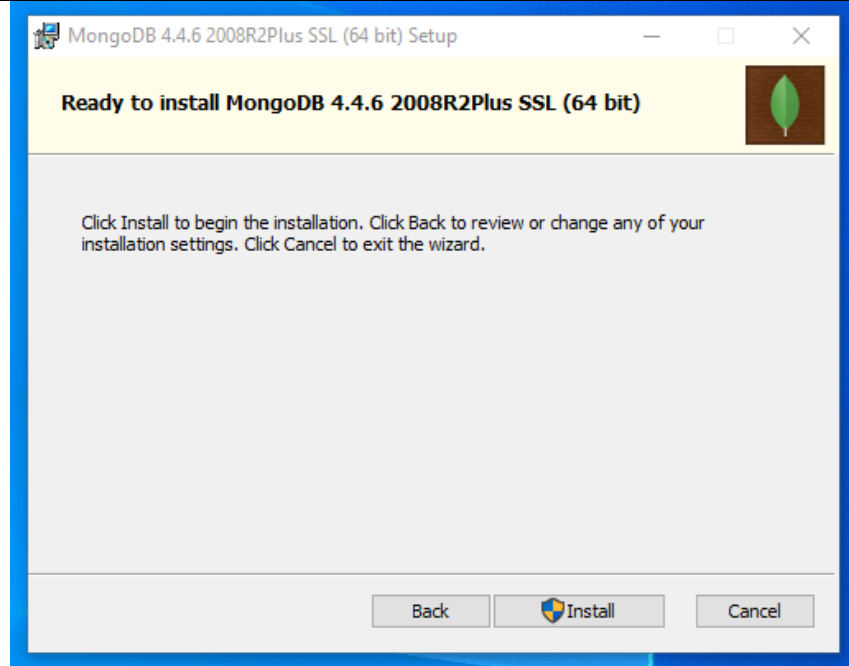
Choose the **Complete** installation to install all of the MongoDB components.



The default values should work well for most scenarios. Click **Next** when you are satisfied with your selections.



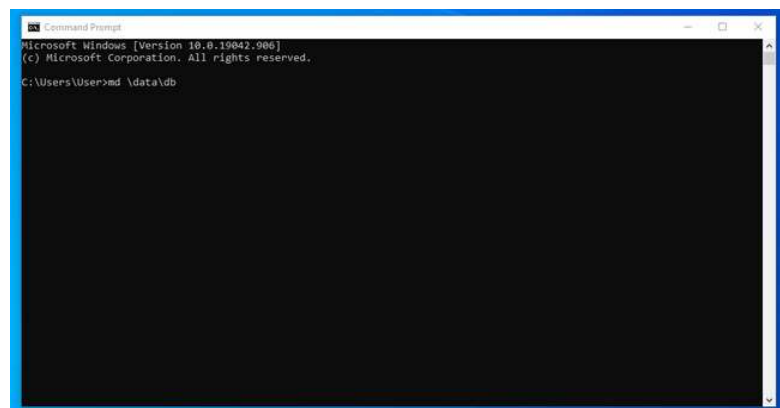
Click **Next** after making your decision.



Click **Install** to begin installing all of the MongoDB components on your computer.

Before you run the server, you need to create the default directory where MongoDB stores its data: `\data\db`. You can create that directory by typing:

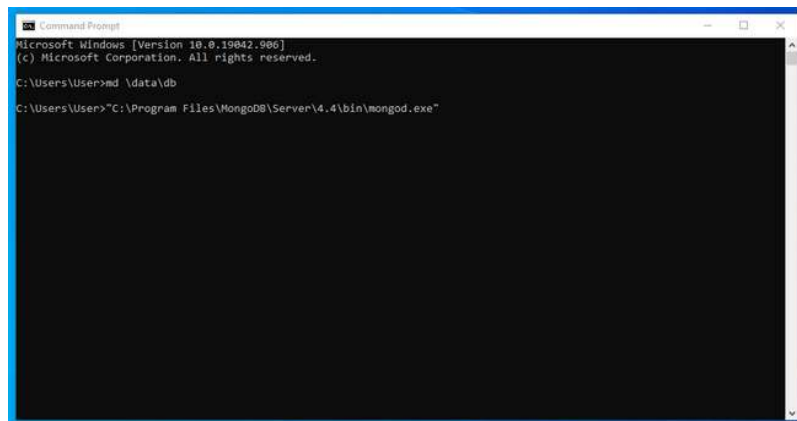
```
md \data\db
```



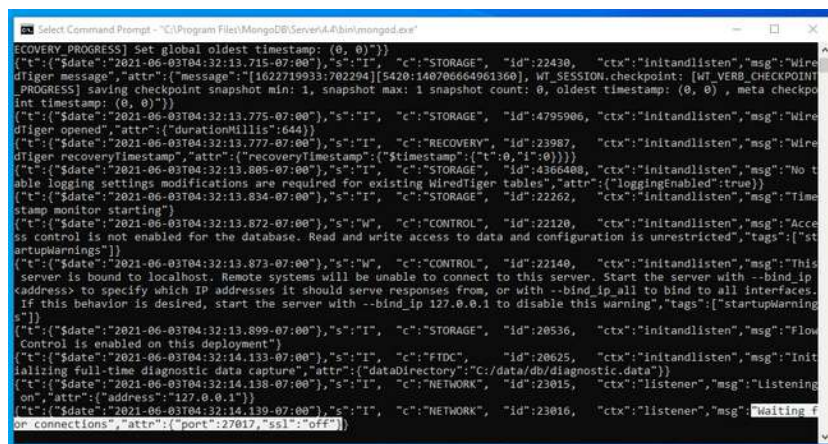
Part of the path contains the MongoDB version number that you installed, so your installation path may be slightly different than the one used below:

```
C:\Program
```

Files\MongoDB\Server\4.4\bin\mongod.exe



If everything is functioning correctly, the server will start up and output diagnostic information to the console. To verify that the startup was successful, look for a message that indicates that it is now accepting connections from clients:



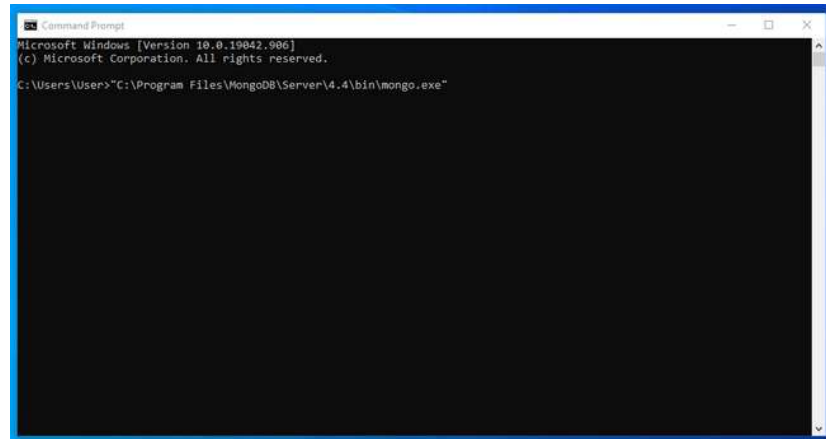
To connect to your running MongoDB server, open another Command Prompt window. Similar to before, we need to type in the absolute path to the executable file.

In this case, we are trying to run the `mongo.exe` executable so, taking into account the differences in version numbers, the command should

look something like this:

C:\Program

Files\MongoDB\Server\4.4\bin\mongo.exe

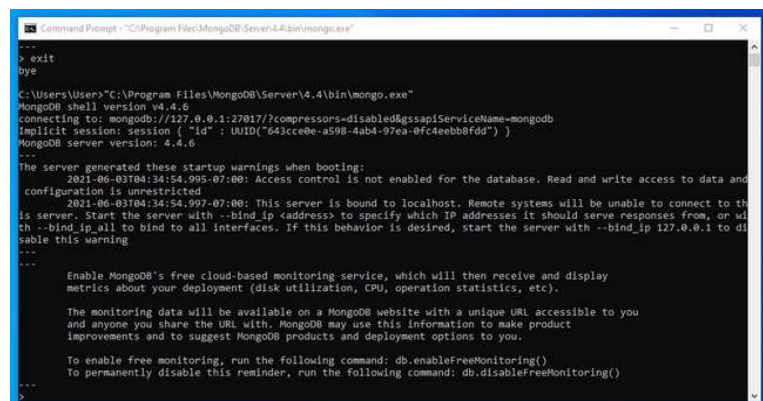


```
Microsoft Windows [Version 10.0.19042.986]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe

MongoDB shell version v4.4.6
```

Once the shell connects to the server, it will print information about the connection and drop you into a MongoDB prompt:

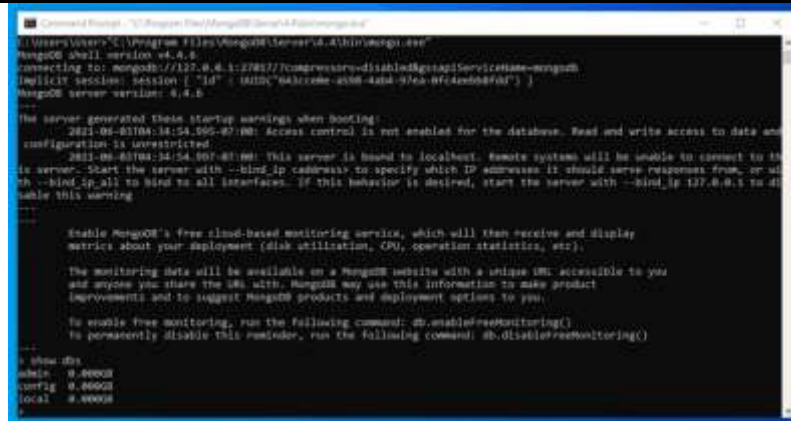


```
---
> exit
bye
C:\Users\User>C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("643c0e0e-a598-4ab4-97ea-0fc4eebb8fdd") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
2021-06-03T04:34:54.995-07:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
2021-06-03T04:34:54.997-07:00: This server is bound to localhost. Remote systems will be unable to connect to th
is server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or w
th --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to di
sable this warning
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

To verify that the server is responding to commands, run the show dbs command:



```

C:\Users\Sana> cd "C:\Program Files\MongoDB\Server\4.4\bin"
C:\Program Files\MongoDB\Server\4.4\bin> mongo
MongoDB shell version: 4.4.6
connecting to: mongodb://127.0.0.1:27027/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : "MIDC-balxcmk-4508-4404-07ea-0f64e0b0f0d4" }
MongoDB server version: 4.4.6

The server generated these startup warnings when booting:
  2023-09-03T04:34:54.505-07:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted.
  2023-09-03T04:34:54.507-07:00: This server is bound to localhost. Remote systems will be unable to connect to the
server. Start the server with --bind_ip address to specify which IP addresses it should serve responses from, or use
--bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to dis
able this warning.

---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---
> use admin
admin> use admin
config> use admin
local> use admin

```

2. Problem statement (for which domain database is going to maintained)

Introduction:

In the fast-paced world of healthcare, efficient management of patient information and doctor records is paramount for ensuring quality patient care and streamlining hospital operations. To address this need, we are tasked with developing a Hospital Management System using MongoDB as the database system. The system will encompass two main entities: Patients and Doctors. This problem statement outlines the purpose and objectives of the database, as well as the utilization of MongoDB to create a robust Hospital Management System.

Problem Statement:

The primary objective of this project is to design a MongoDB database for a Hospital Management System, which will facilitate efficient storage, retrieval, and manipulation of data related to patients and doctors in a hospital setting. The database should meet the following requirements:

Patient Management:

Store detailed information about patients, including but not limited to their personal details (name, contact information, date of birth), medical history, and admission details.
Enable efficient search and retrieval of patient records for quick access by medical staff, such as nurses and doctors.
Support the addition, modification, and deletion of patient records as needed.

Doctor Management:

Maintain a comprehensive database of doctors working in the hospital, including their contact information, specialization, and availability.
Enable quick access to doctor profiles for appointment scheduling and patient referrals.
Allow the hospital administration to manage doctor records efficiently.

Patient-Doctor Relationship:

Establish a connection between patients and their attending doctors. Each patient should be associated with a doctor responsible for their treatment.

Enable easy assignment and reassignment of doctors to patients as required.

Store records of appointments and treatments associated with specific patient-doctor relationships.

3. Apply various basic operations and CRUD operations (snapshots of each)

1. Show Databases -

Command -

show dbs

```
-----  
test> show dbs  
admin    40.00 KiB  
config   12.00 KiB  
local    40.00 KiB  
test>
```

2. Create Database -

Command -

use <database_name>

```
test> use hospital  
switched to db hospital  
hospital> show dbs  
admin    40.00 KiB  
config   60.00 KiB  
local    40.00 KiB  
hospital>
```

3. Create Tables -

Command -

db.createCollection("<tablename>")


```
hospital> db.createCollection("doctors")
{ ok: 1 }
hospital> db.createCollection("patients")
{ ok: 1 }
```

4. Insert Values in DB

Command (insertOne)-

db.<table_name>.insertOne({<key>:"<value>"});

```
hospital> db.Patient.insertOne({
...   ID: 101,
...   Name: "Ella Davis",
...   Place: "Houston",
...   Condition: "Infection"
... });
{
  acknowledged: true,
  insertedId: ObjectId("6527ed561ecdbbd61542461a")
}
```

Command (insertMany)-

db.<table_name>.insertMany(
{<key1>:"<value1>"},
{<key2>:"<value2>"}
);

```
hospital> db.Patient.insertMany([
...   {
...     ID: 102,
...     Name: "John Doe",
...     Place: "New York",
...     Condition: "Fever"
...   },
...   {
...     ID: 103,
...     Name: "Jane Smith",
...     Place: "Los Angeles",
...     Condition: "Fracture"
...   },
...   {
...     ID: 104,
...     Name: "Alice Johnson",
...     Place: "Chicago",
...     Condition: "Allergy"
...   }
... ])
hospital>
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6527ed791ecdbbd61542461b"),
    '1': ObjectId("6527ed791ecdbbd61542461c"),
    '2': ObjectId("6527ed791ecdbbd61542461d")
  }
}
```

5. Display -

Command - db.<table_name>.find()

```
hospital> db.Patient.find()
[
  {
    _id: ObjectId("6527ed561ecdbbd61542461a"),
    ID: 101,
    Name: 'Ella Davis',
    Place: 'Houston',
    Condition: 'Infection'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461b"),
    ID: 102,
    Name: 'John Doe',
    Place: 'New York',
    Condition: 'Fever'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461c"),
    ID: 103,
    Name: 'Jane Smith',
    Place: 'Los Angeles',
    Condition: 'Fracture'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461d"),
    ID: 104,
    Name: 'Alice Johnson',
    Place: 'Chicago',
    Condition: 'Allergy'
  }
]
```

6. Update -

Command -

`db.<table_name>.updateOne(&set:{<key>:<new_value>})`

```
hospital> db.Patient.updateOne(
...   { ID: 101 },
...   { $set: { Name: "Michael Jackson" } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
hospital> db.Patient.find()
[
  {
    _id: ObjectId("6527ed561ecdbbd61542461a"),
    ID: 101,
    Name: 'Michael Jackson',
    Place: 'Houston',
    Condition: 'Infection'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461b"),
    ID: 102,
    Name: 'John Doe',
    Place: 'New York',
    Condition: 'Fever'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461c"),
    ID: 103,
    Name: 'Jane Smith',
    Place: 'Los Angeles',
    Condition: 'Fracture'
  },
  {
    _id: ObjectId("6527ed791ecdbbd61542461d"),
    ID: 104,
    Name: 'Alice Johnson',
    Place: 'Chicago',
    Condition: 'Allergy'
  }
]
```

Delete Table -

`db.<table_name>.drop()`

```
hospital> db.Doctor.drop()
true
```

	<p>Drop Database -</p> <pre>temp> db.dropDatabase() { ok: 1, dropped: 'temp' } temp></pre>
Conclusion:	Thus students will be able to successfully installed mongodb and applied CRUD operations to manage the big data.
References:	<p>https://www.mongodb.com/</p> <p>https://docs.mongodb.com/</p> <p>https://university.mongodb.com/</p>

Don Bosco Institute of Technology

Department of Computer Engineering

Academic year – 2023-2024

Big Data Analytics

Assessment Rubric for Experiment No. :

05

Performance Date :

Submission Date :

Title of Experiment : To install and configure MongoDB to execute NoSQL commands.

Year and Semester : IVth Year and VIIth Semester

Batch : A

Name of Student : Alston Fernandes

Roll No. : 19

Performance	Poor	Satisfactory	Good	Excellent
	1 point	2 points	3 points	4 points
Results and Documentations	Poor	Satisfactory	Good	Excellent
	1 point	2 points	3 points	4 points
Viva	Poor	Satisfactory	Good	Excellent
	1 point	2 points	3 points	4 points
Timely Submission	Submission beyond 7 days of the deadline	Late submission till 7 days	Submission on time	
	1 points	2 points	3 points	

Faculty Incharge : Ms. Sana Shaikh