BCSE302L-DatabaseManagement System

A2+TA2

MongoDB Application

Learning Management System (LMS)

Names and registration numbers:

Aastha – 22BCE0026

Shruti Anil Gaikwad – 22BDS0298

Kritika Bansal – 22BDS0304

Video recording link:

https://drive.google.com/file/d/115qcXt6hqSGJ-U7_O1qu2YZsisoVB4Lf/view?usp=drive_link

Introduction

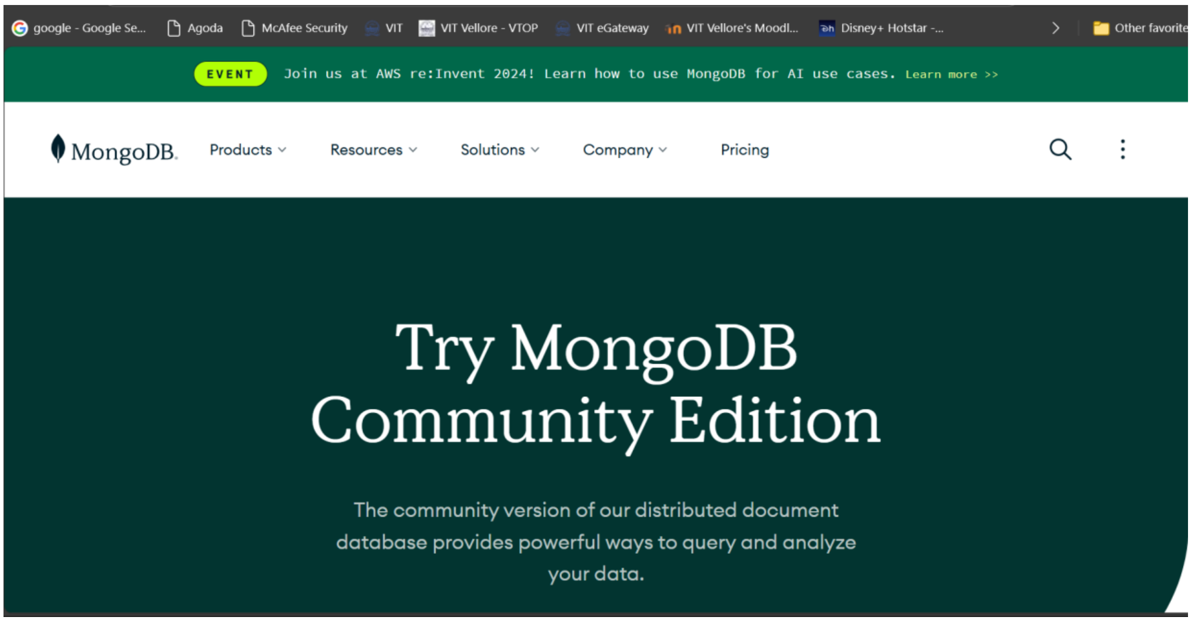
MongoDB is a **powerful and flexible solution for handling modern data needs.** As a leading NoSQL database, MongoDB offers a dynamic schema design, enabling developers to store and manage data in a way that aligns seamlessly with contemporary application requirements.

Unlike traditional relational databases, **MongoDB's document-oriented architecture allows for greater agility and scalability,** making it a preferred choice for businesses and developers aiming to handle large volumes of unstructured or semi-structured data.

Installation steps:

Step 1: Download MongoDB

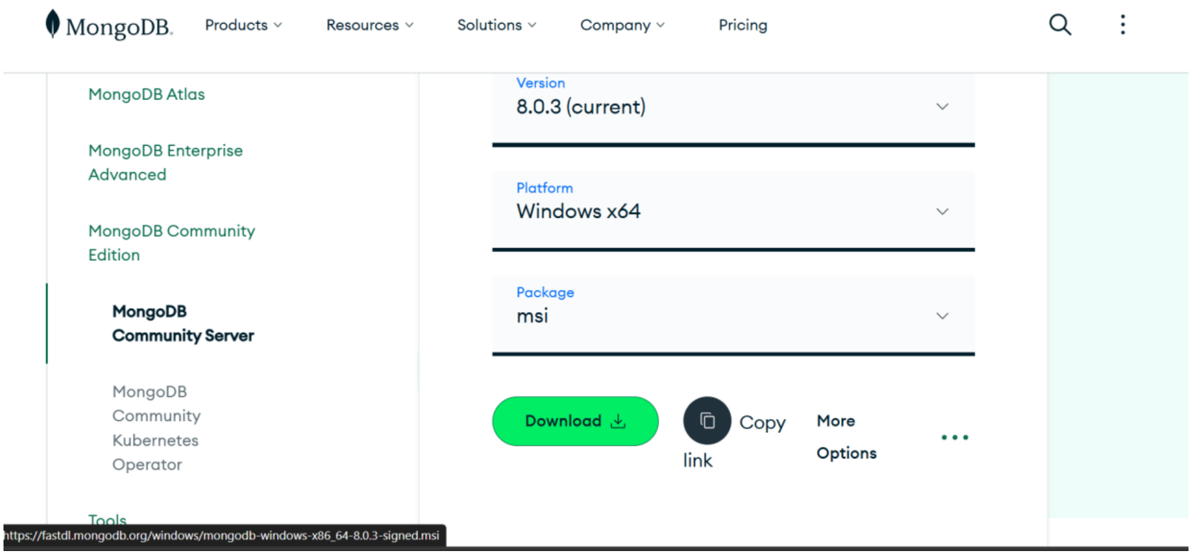
1. Visit the official MongoDB Download Center.



Try MongoDB Community Edition

The community version of our distributed document database provides powerful ways to query and analyze your data.

2. Under **MongoDB Community Server**, select the latest release for Windows.
3. Choose your platform as **Windows** and package as **MSI**.



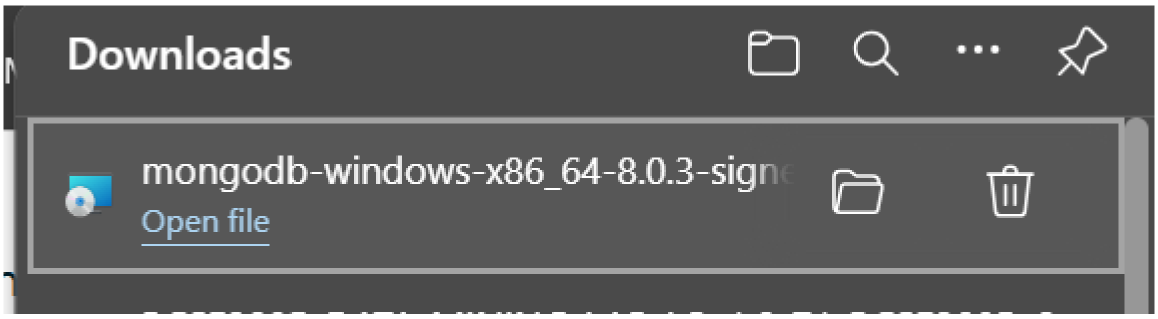
Version: 8.0.3 (current)

Platform: Windows x64

Package: msi

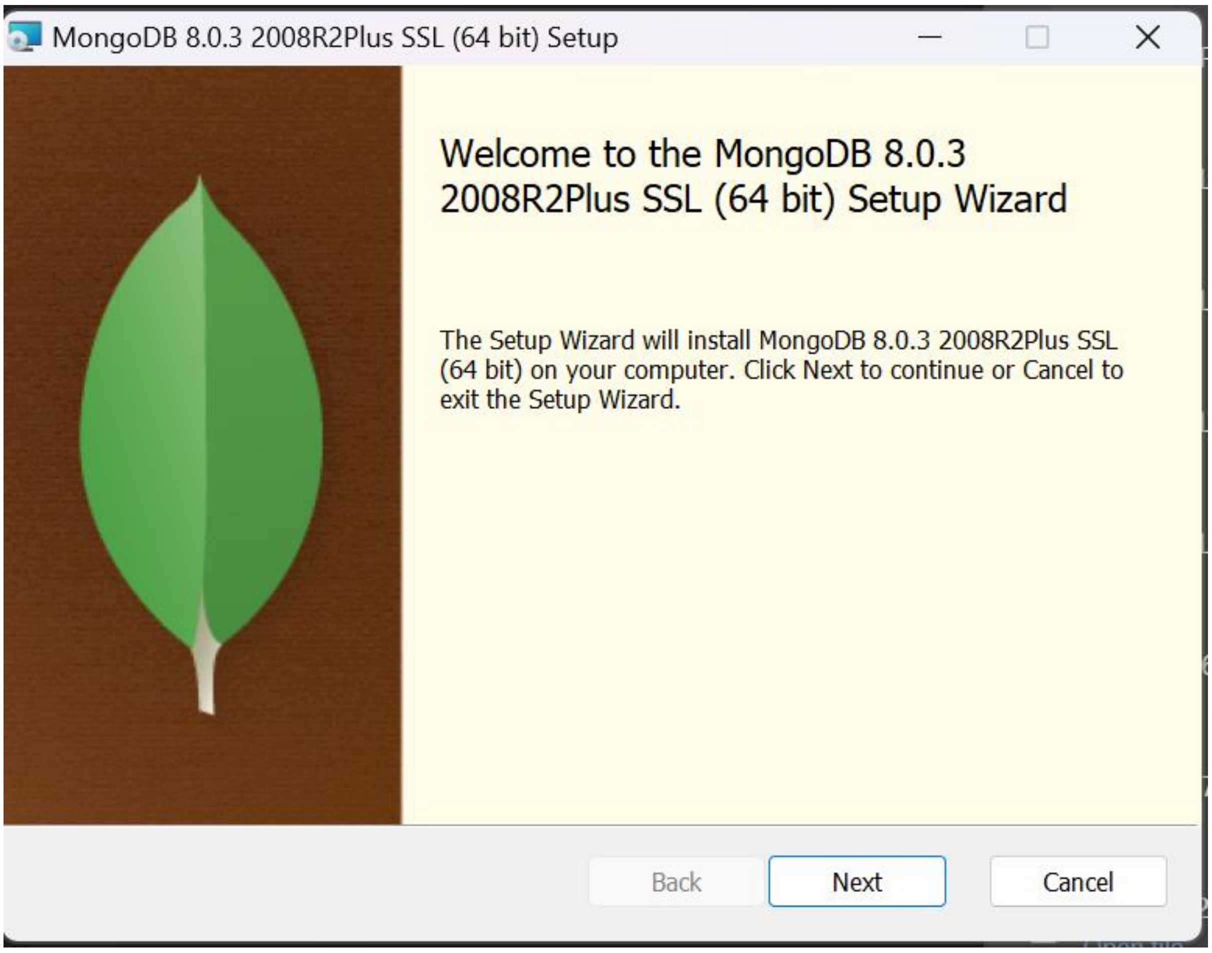
https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-8.0.3-signed.msi

4. Click the **Download** button to get the installer.



Step 2: Install MongoDB

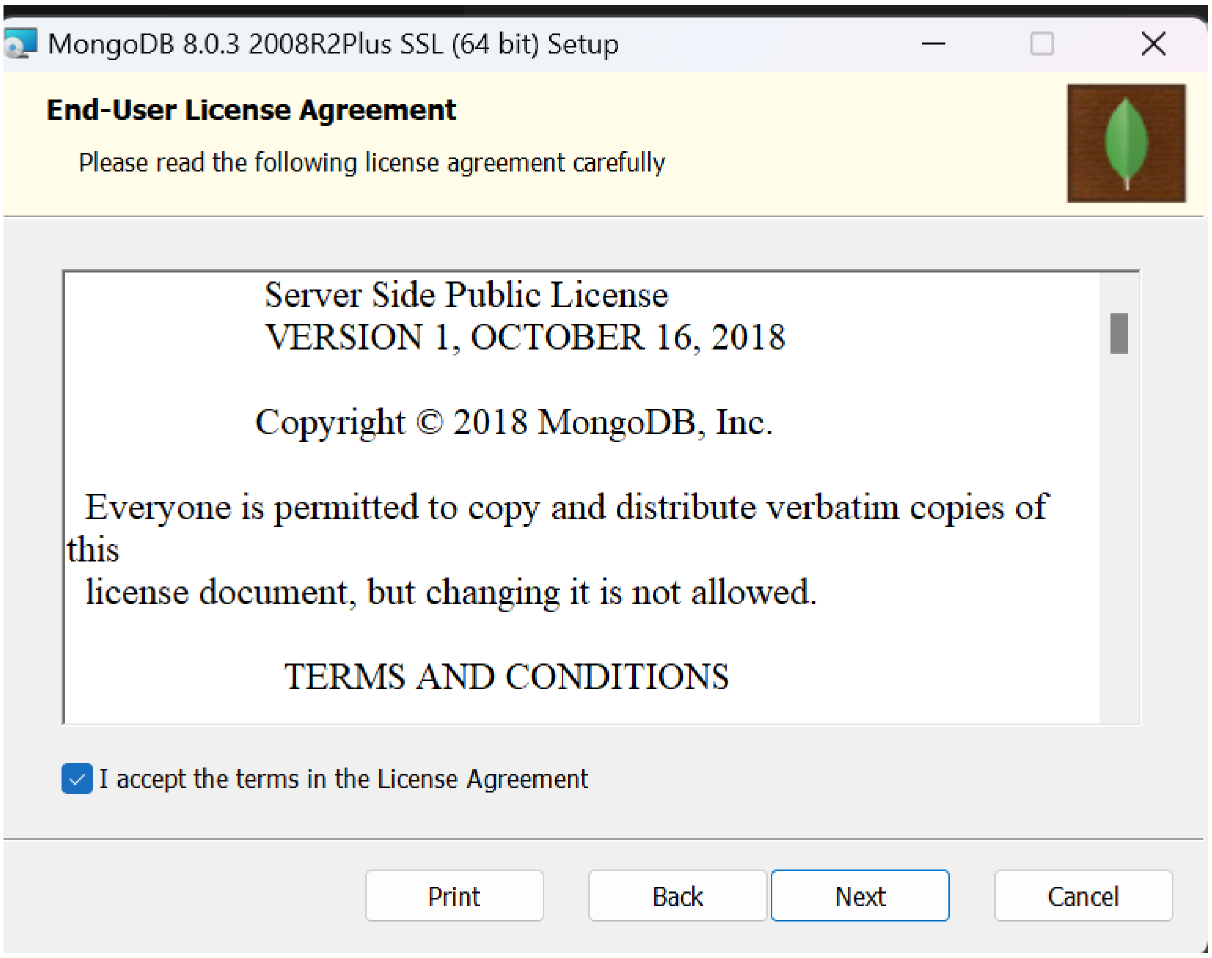
1. Run the **MSI** installer you just downloaded.



MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 8.0.3 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.


MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

Server Side Public License
VERSION 1, OCTOBER 16, 2018

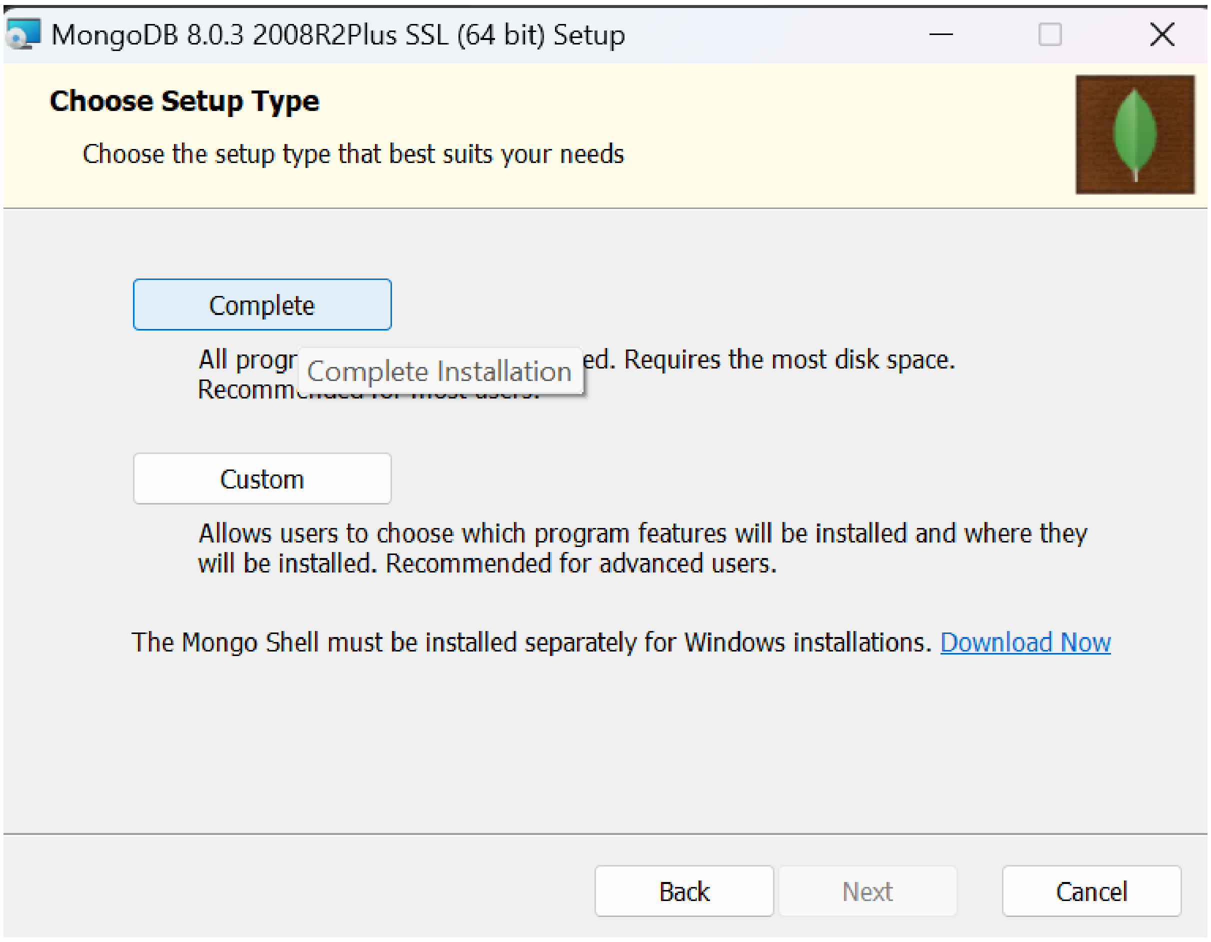
Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

TERMS AND CONDITIONS

[✓] I accept the terms in the License Agreement

2. Follow the installation instructions:
    - **Choose Setup Type**: Select "Complete" for a full installation.


MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

Choose the setup type that best suits your needs

Complete

All progr Complete Installation ed. Requires the most disk space.
Recomm

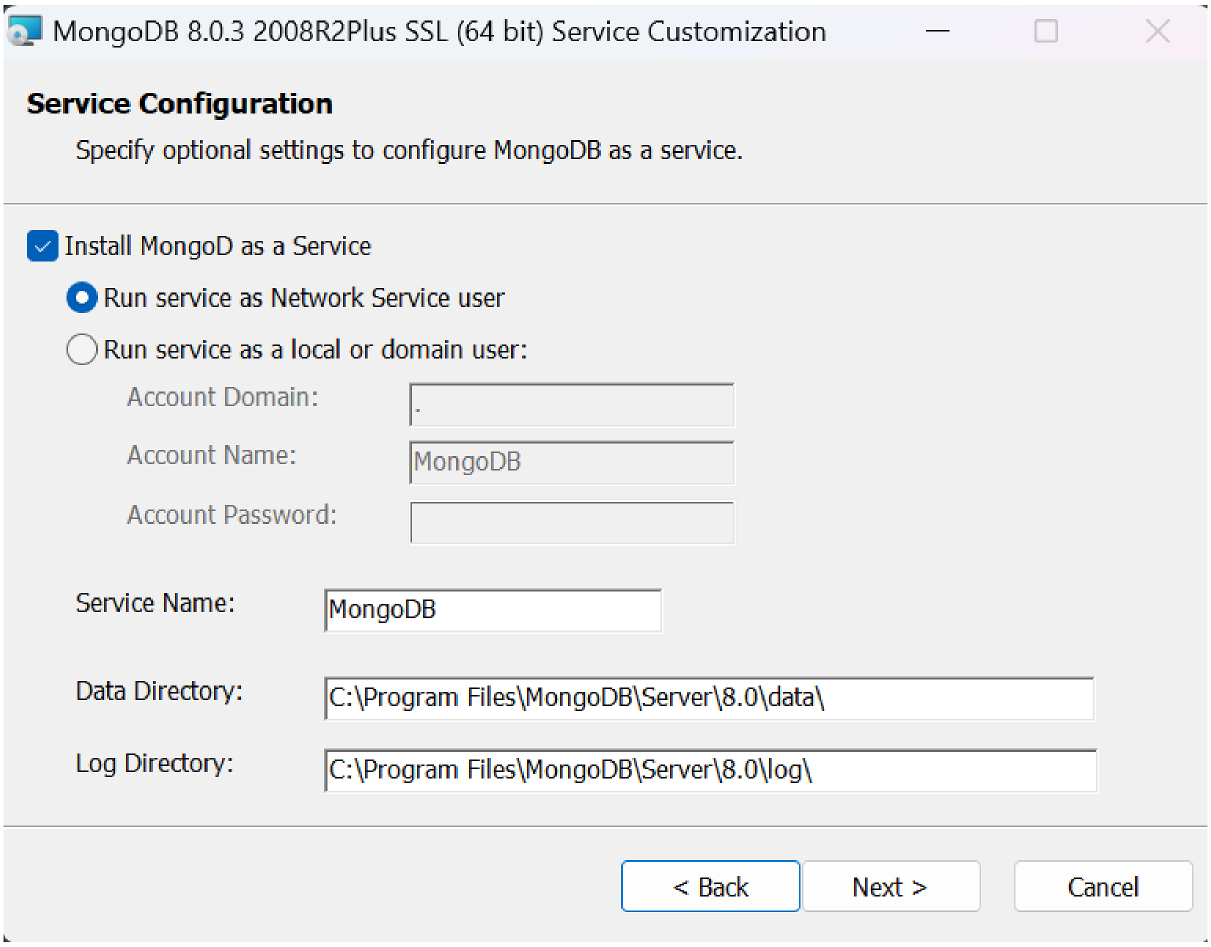
Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

The Mongo Shell must be installed separately for Windows installations. Download Now

Back Next Cancel

- **Custom Installation**: You can choose a custom installation if you want to install MongoDB in a specific directory, but "Complete" is recommended for most users.
- **Service Configuration**: Ensure "Run MongoDB as a Service" is checked. This allows MongoDB to start automatically when your system starts. You can also configure it to run manually later.
    - **Service Name**: Keep the default name (MongoDB).
    - **Data Directory**: The directory where MongoDB stores data (C:\Program Files\MongoDB\Server\{version}\data).
    - **Log Directory**: The directory where MongoDB stores logs (C:\Program Files\MongoDB\Server\{version}\log).


MongoDB 8.0.3 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

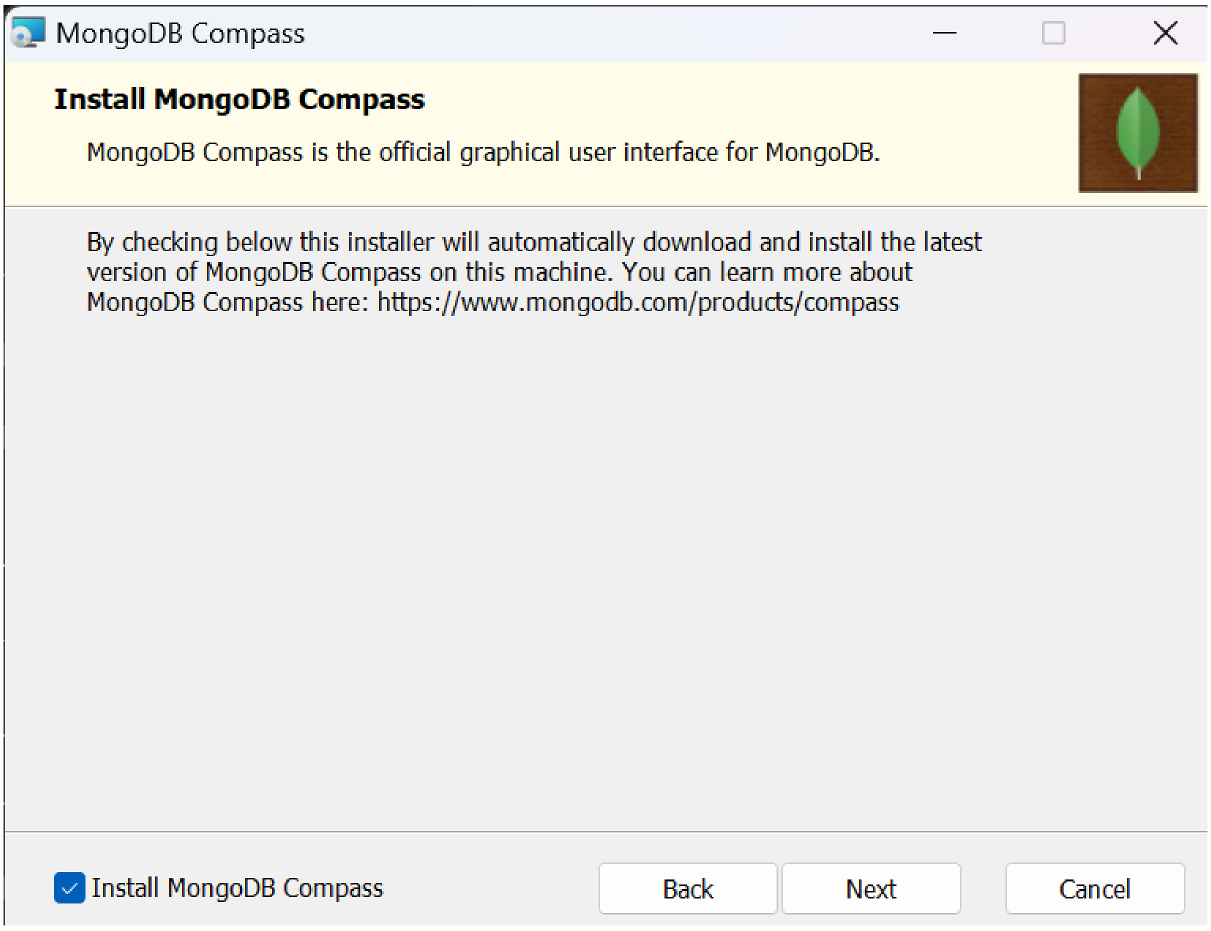
[✓] Install MongoD as a Service

[●] Run service as Network Service user

○ Run service as a local or domain user:

Account Domain:	.
Account Name:	MongoDB
Account Password:	
Service Name:	MongoDB
Data Directory:	C:\Program Files\MongoDB\Server\8.0\data\
Log Directory:	C:\Program Files\MongoDB\Server\8.0\log\

- **Install MongoDB Compass**: MongoDB Compass is a GUI for MongoDB. You can install it or skip if you prefer to use the command line.


MongoDB Compass

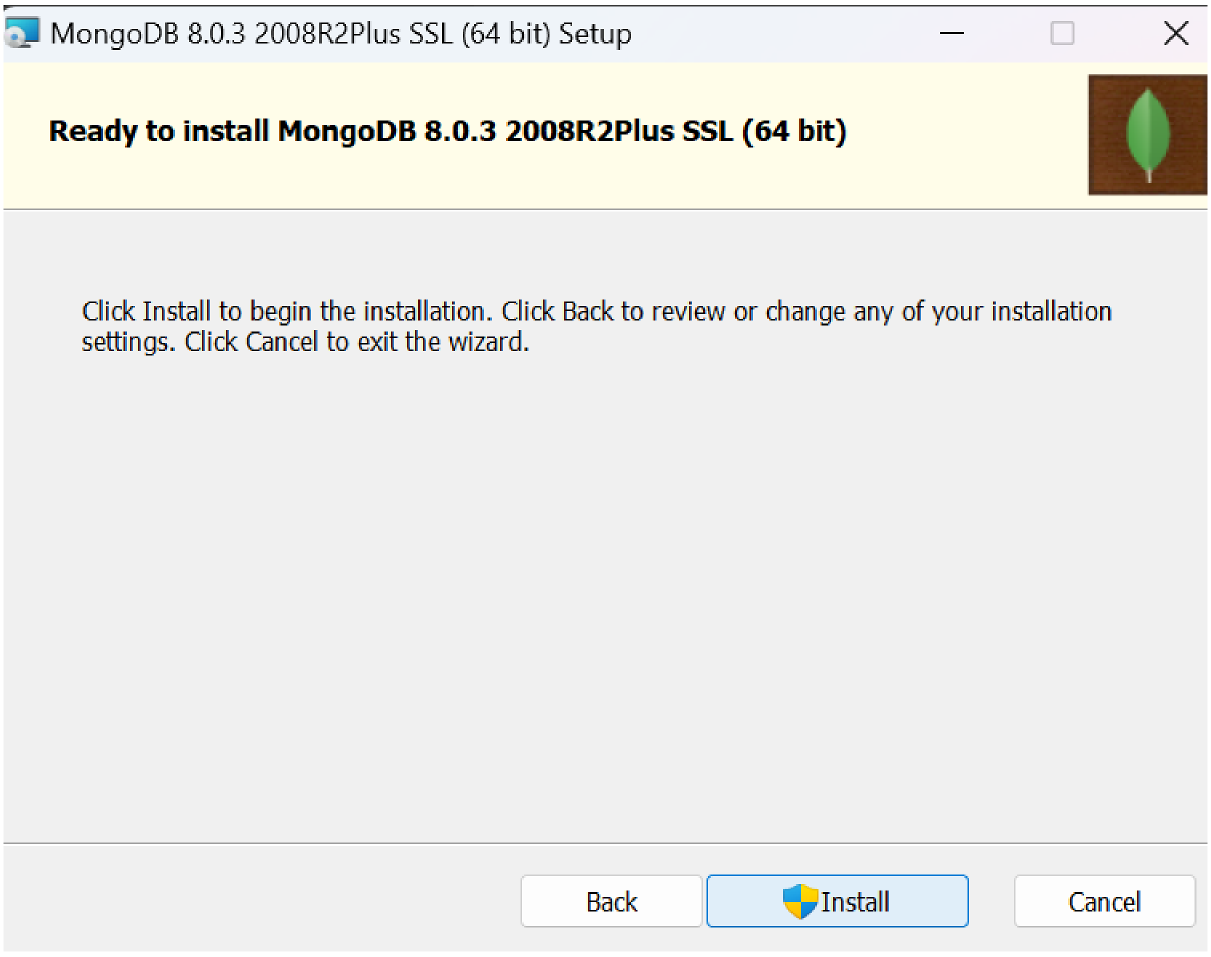
Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

[✓] Install MongoDB Compass

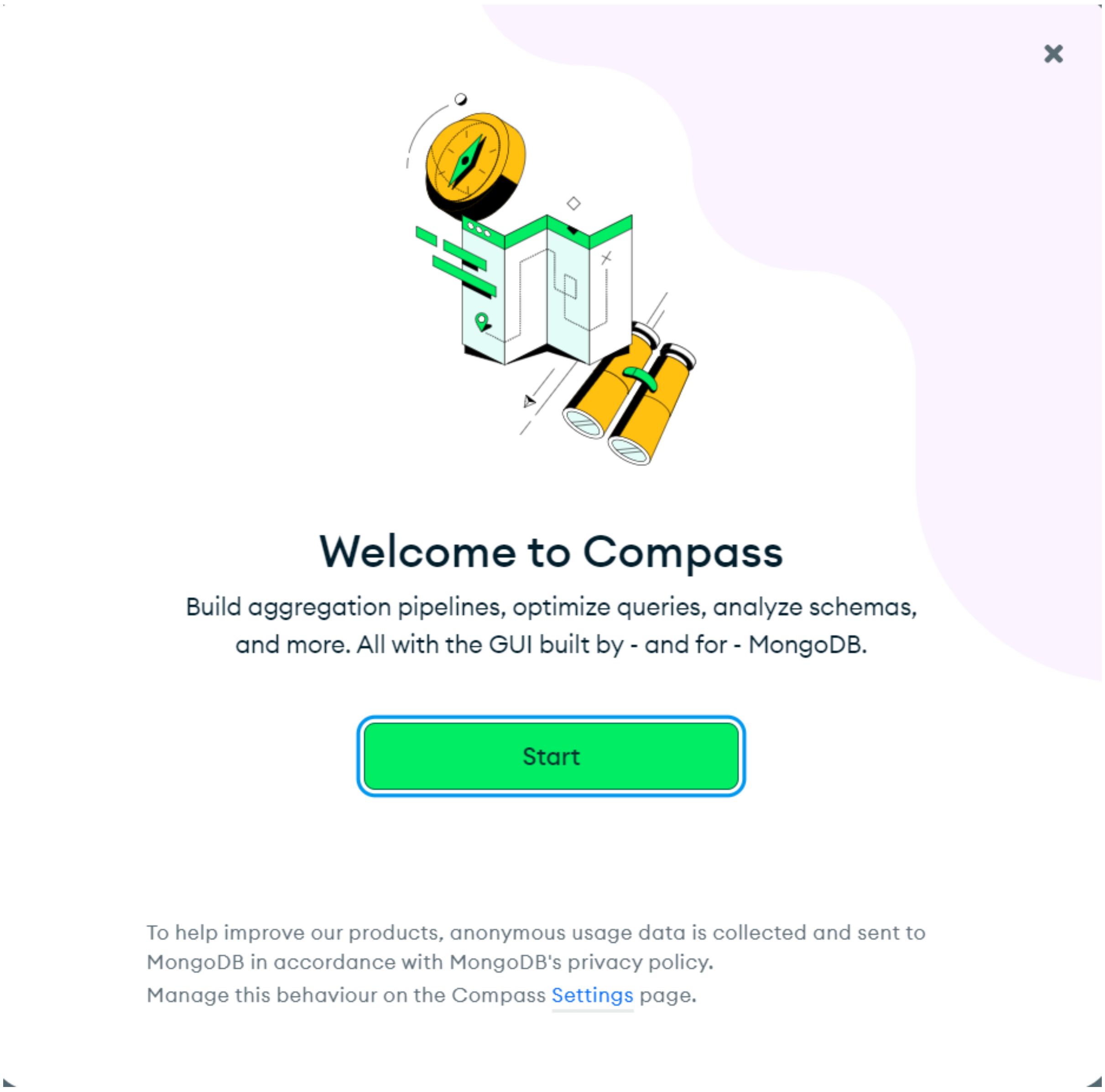
3. Click **Install** and complete the installation.


MongoDB 8.0.3 2008R2Plus SSL (64 bit) Setup

Ready to install MongoDB 8.0.3 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

4. GO TO MONGOCOMPASS AND ADD NEW CONECTION

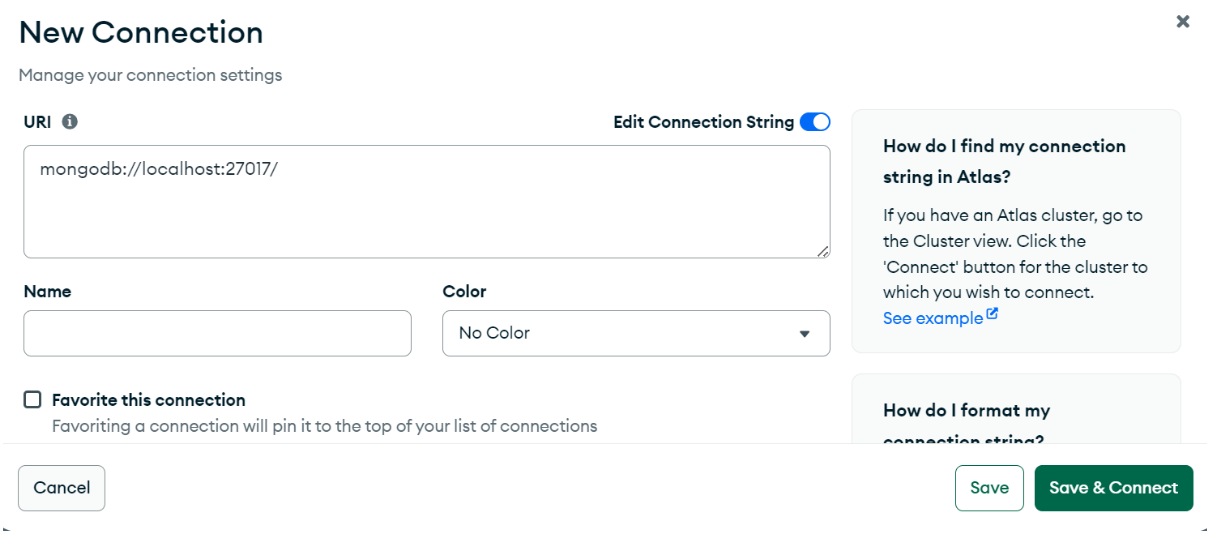


Welcome to Compass

Build aggregation pipelines, optimize queries, analyze schemas, and more. All with the GUI built by - and for - MongoDB.

To help improve our products, anonymous usage data is collected and sent to MongoDB in accordance with MongoDB's privacy policy.
Manage this behaviour on the Compass Settings page.

5. SAVE AND CONNECT TO ESTABLISH CONNECTION.


New Connection

Manage your connection settings

URI

Edit Connection String

```
mongodb://localhost:27017/
```

Name

Color: No Color

Favorite this connection

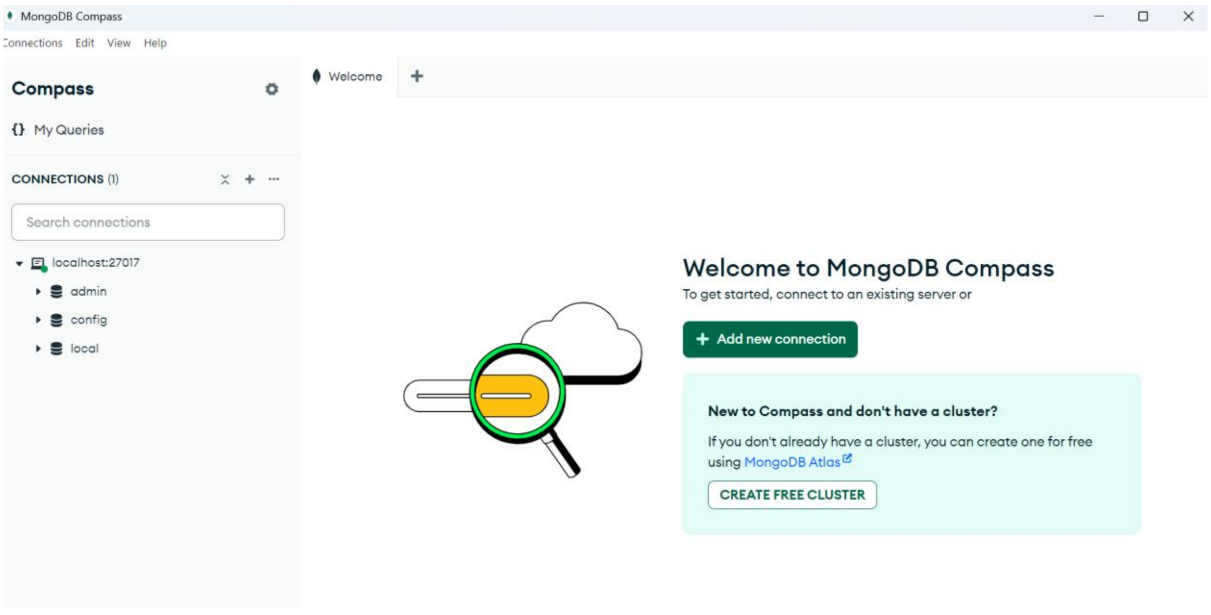
Favoriting a connection will pin it to the top of your list of connections

How do I find my connection string in Atlas?

If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.

See example

How do I format my connection string?



Welcome to MongoDB Compass

To get started, connect to an existing server or

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas

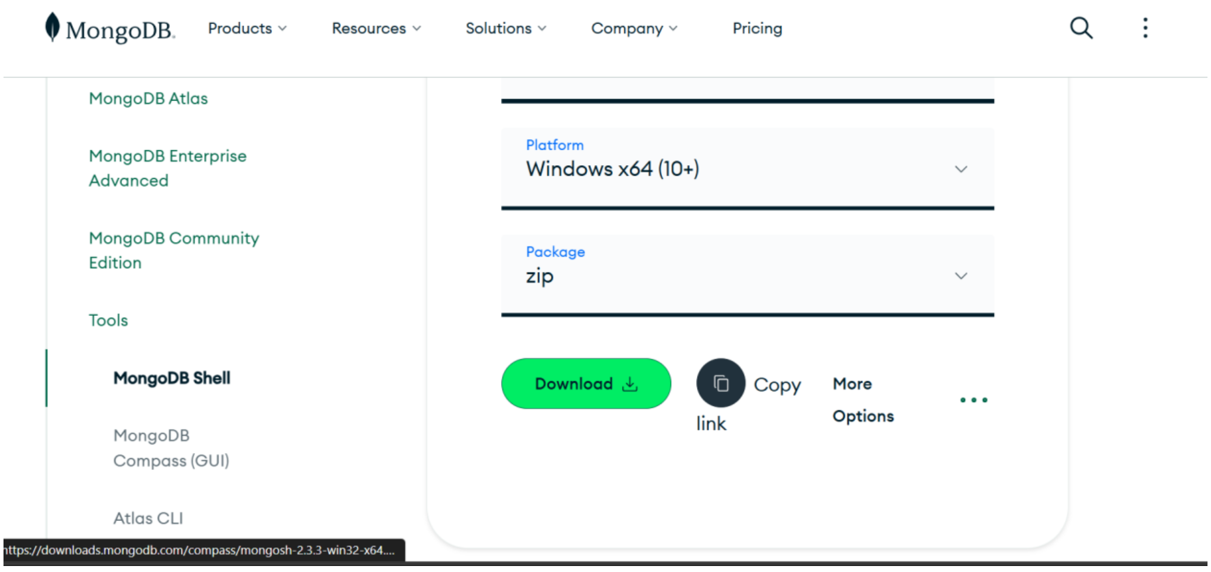
DOWNLOADING MONGO SHELL

If you specifically want to download the **MongoDB Shell (mongosh)** without installing the full MongoDB server, here's how to do it:

Step 1: Download MongoDB Shell (mongosh)

1. Go to the official MongoDB Download Center.
2. Under **MongoDB Shell**, choose your platform as **Windows**.

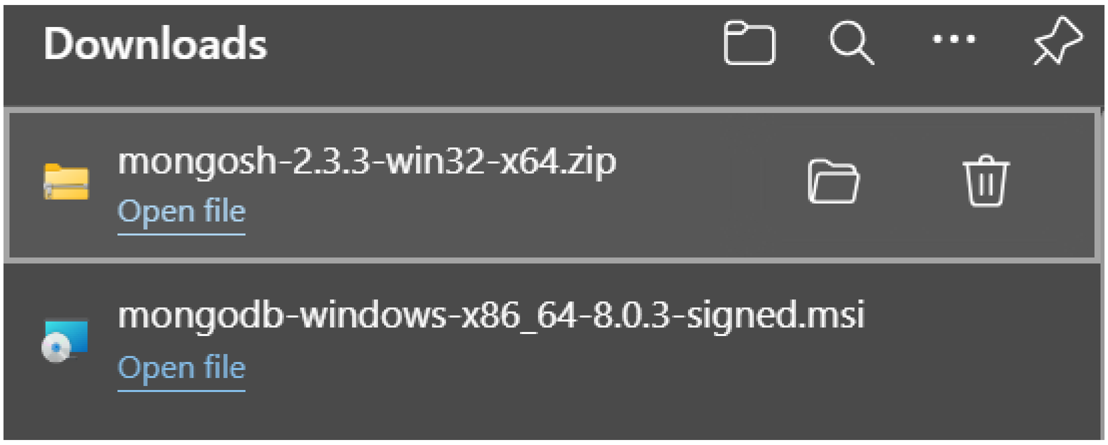
3. Click the **Download** button to download the shell package (MSI file).



Platform: Windows x64 (10+)

Package: zip

https://downloads.mongodb.com/compass/mongosh-2.3.3-win32-x64....

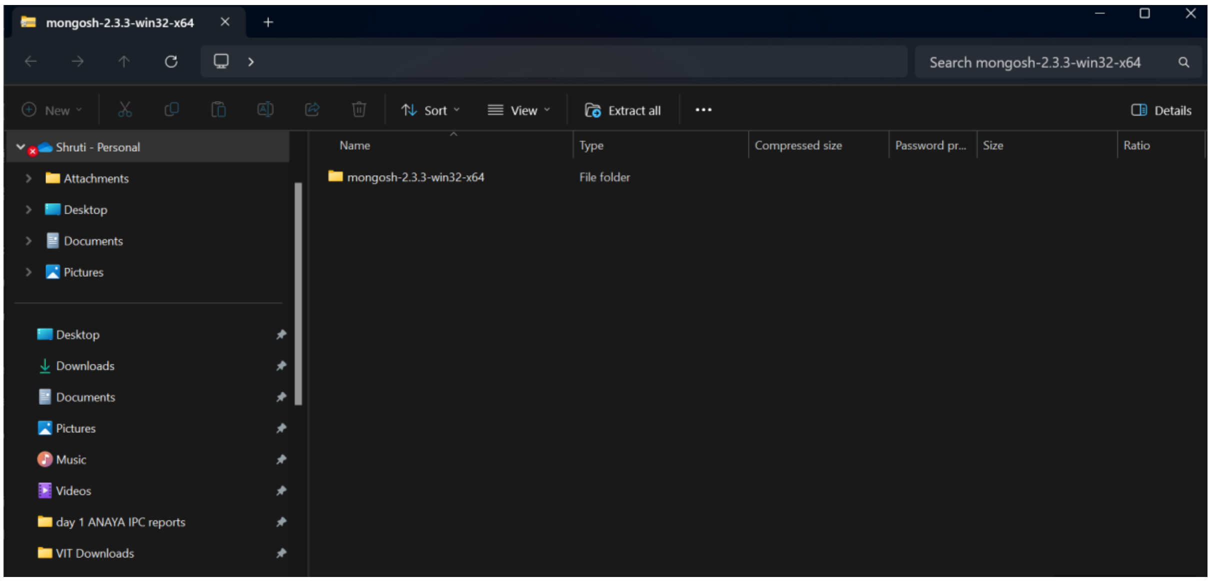


mongosh-2.3.3-win32-x64.zip

mongodb-windows-x86_64-8.0.3-signed.msi

Step 2: Install MongoDB Shell

1. Run the downloaded **MSI** installer.
2. Follow the installation instructions:
   - Accept the terms and conditions.
   - Choose the installation folder or use the default.
   - Click **Install** to complete the installation.

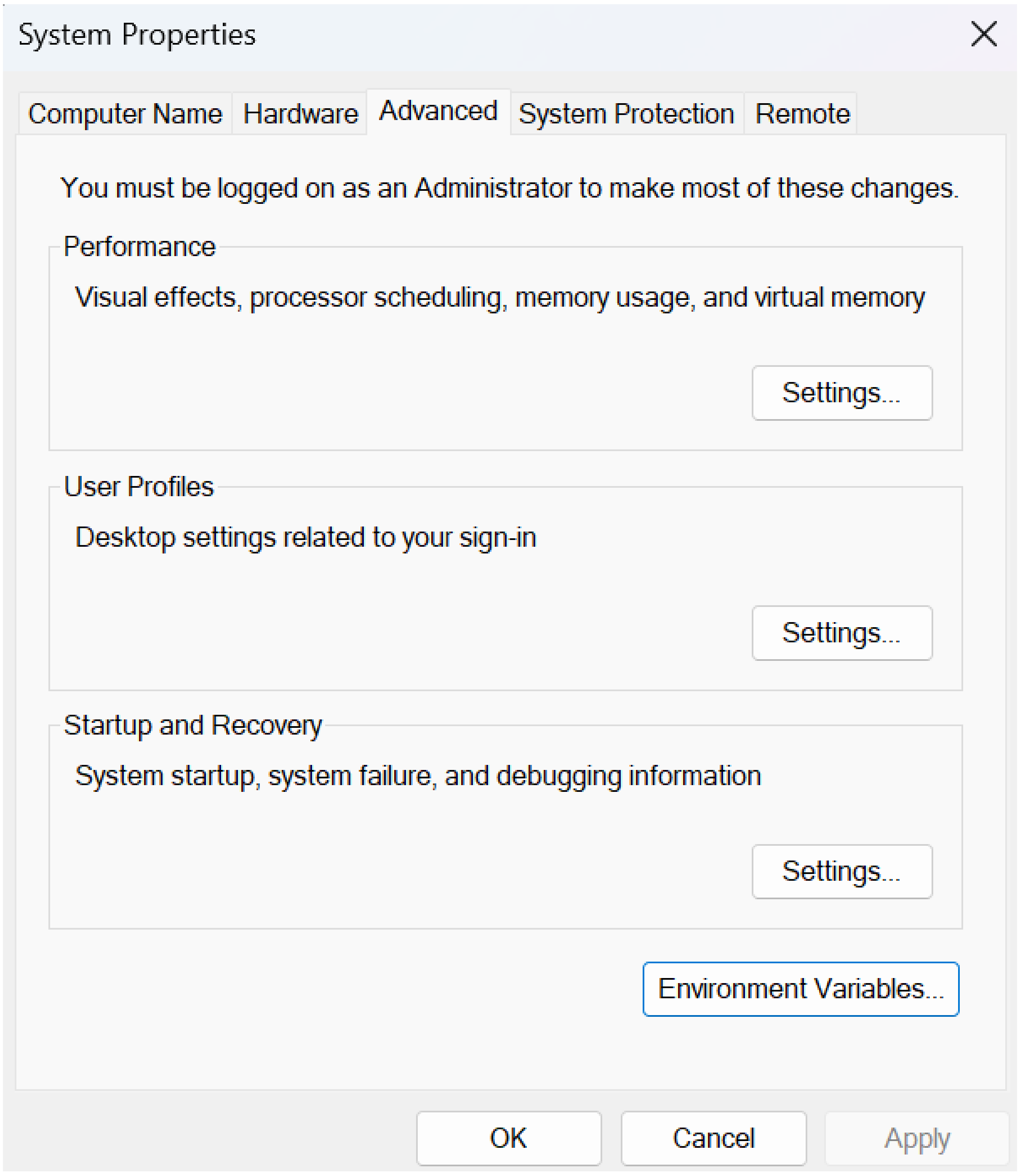


Name	Type	Compressed size	Password pr...	Size	Ratio
mongosh-2.3.3-win32-x64	File folder				

Step 3: Add mongosh to the System PATH (Optional)

To make sure you can run mongosh from any command prompt, you might need to add it to your system PATH:

1. Go to **Start** → **Control Panel** → **System** → **Advanced System Settings** → **Environment Variables**.


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

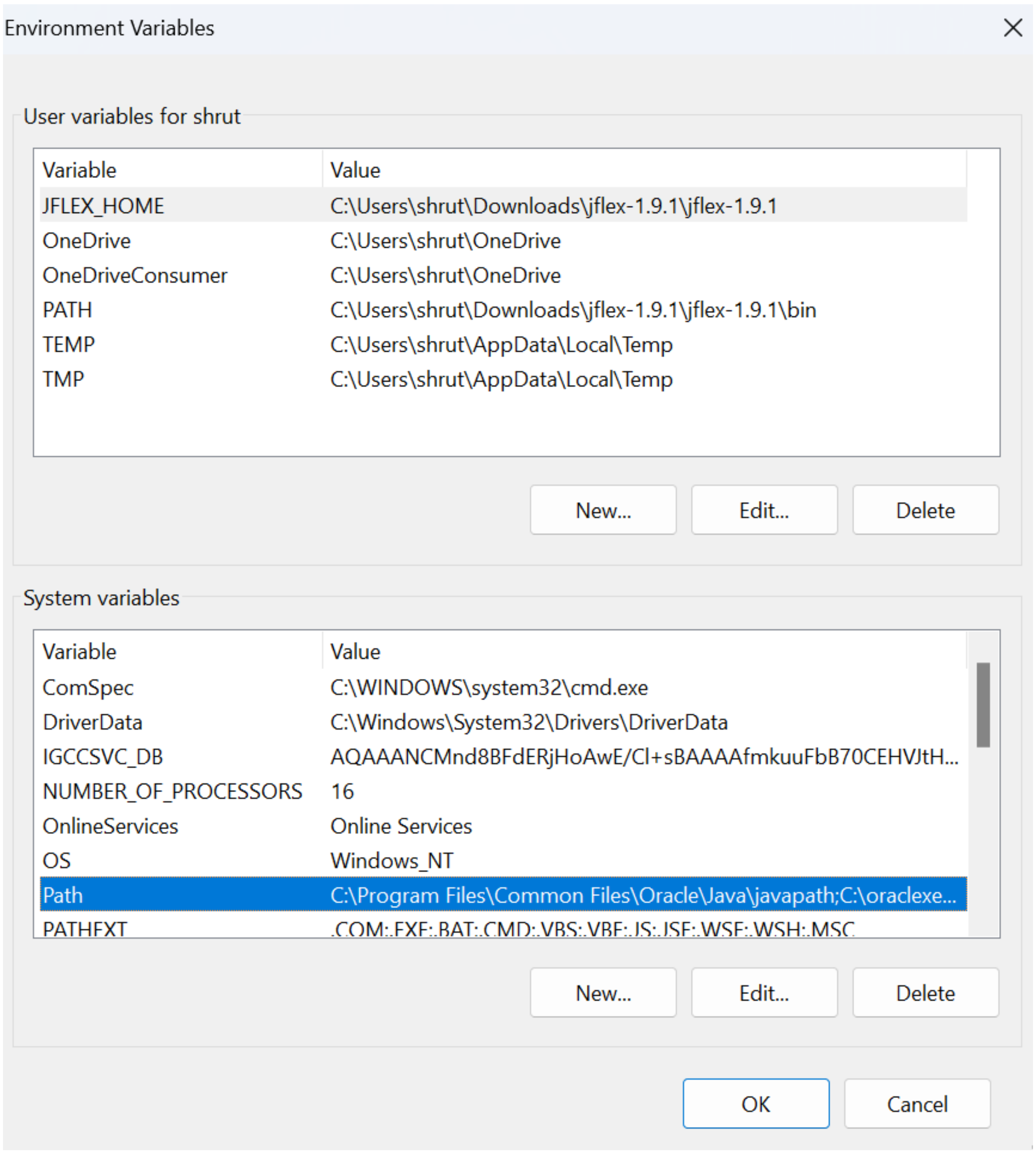
Startup and Recovery

System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply


Environment Variables

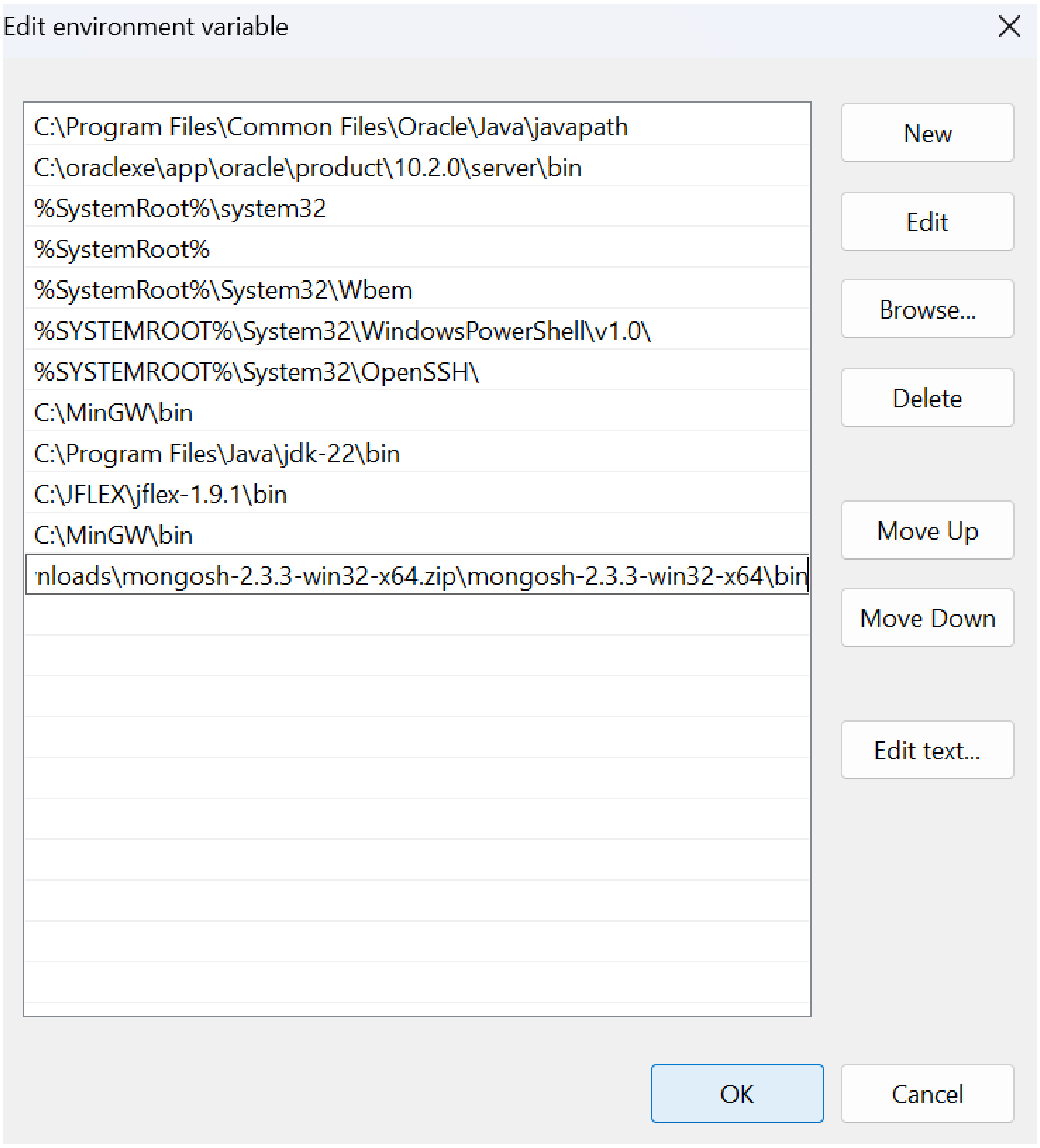
User variables for shrut

Variable	Value
JFLEX_HOME	C:\Users\shrut\Downloads\jflex-1.9.1\jflex-1.9.1
OneDrive	C:\Users\shrut\OneDrive
OneDriveConsumer	C:\Users\shrut\OneDrive
PATH	C:\Users\shrut\Downloads\jflex-1.9.1\jflex-1.9.1\bin
TEMP	C:\Users\shrut\AppData\Local\Temp
TMP	C:\Users\shrut\AppData\Local\Temp

System variables

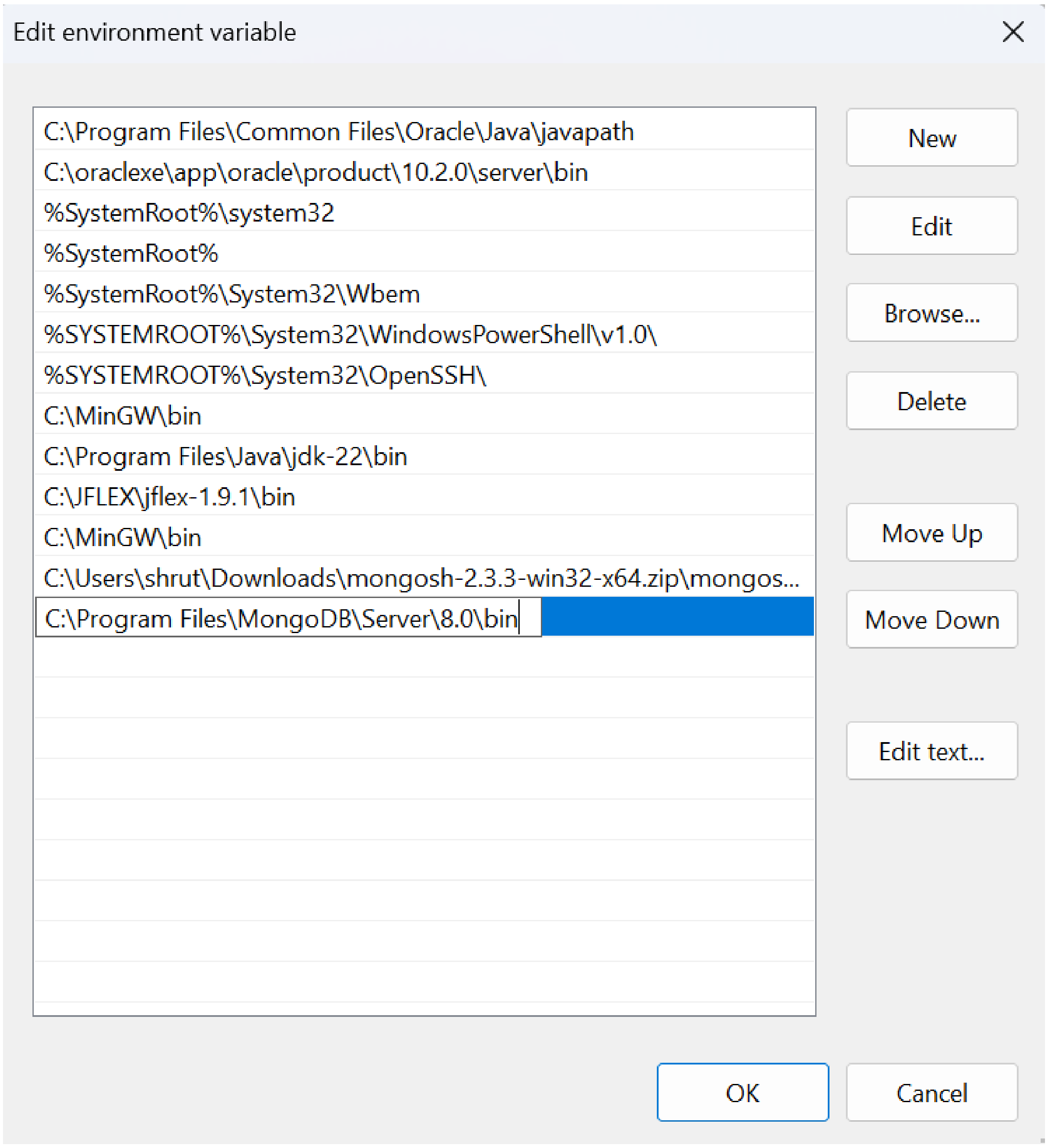
Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
IGCCSVC_DB	AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAfmkuuFbB70CEHVJtH...
NUMBER_OF_PROCESSORS	16
OnlineServices	Online Services
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\oraclexe...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

2. In the **System Variables** section, find the variable named Path and click **Edit**.


Edit environment variable

```
C:\Program Files\Common Files\Oracle\Java\javapath
C:\oraclexe\app\oracle\product\10.2.0\server\bin
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\MinGW\bin
C:\Program Files\Java\jdk-22\bin
C:\JFLEX\jflex-1.9.1\bin
C:\MinGW\bin
nloads\mongosh-2.3.3-win32-x64.zip\mongosh-2.3.3-win32-x64\bin
```

3. Click **New**, then add the path to the folder where **mongosh** was installed (e.g., C:\Program Files\MongoDB\mongosh\bin).



Edit environment variable

```
C:\Program Files\Common Files\Oracle\Java\javapath
C:\oraclexe\app\oracle\product\10.2.0\server\bin
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\MinGW\bin
C:\Program Files\Java\jdk-22\bin
C:\JFLEX\jflex-1.9.1\bin
C:\MinGW\bin
C:\Users\shrut\Downloads\mongosh-2.3.3-win32-x64.zip\mongos...
C:\Program Files\MongoDB\Server\8.0\bin
```

4. Click **OK** to save.

Step 4: Run MongoDB Shell

1. Open **Command Prompt** (you can search for "cmd" in the Start menu).
2. Type mongosh and hit **Enter** to launch the MongoDB shell.
   - If MongoDB is running locally on port 27017 (default), the shell will automatically connect to it.

Step 5: Start Interacting with MongoDB

Once the shell is running, you can begin issuing commands, like connecting to a MongoDB instance, querying, and more.

TOOLS TO BE DOWNLOADED

Step 1: Download MongoDB Database Tools

1. Go to the MongoDB Database Tools Download Page.
2. Select **Windows** as your operating system.
3. Click **Download** to download the tools as a zip file.

Step 2: Extract the Downloaded Zip File

1. Extract the downloaded .zip file to a directory of your choice. For example, C:\MongoDBTools.
2. The extracted folder will contain various tools like mongodump, mongorestore, mongoimport, mongoexport, etc.

Step 3: Add MongoDB Tools to System PATH (Optional)

To easily use the tools from any Command Prompt, you can add them to your system's PATH.

1. Go to **Start** → **Control Panel** → **System** → **Advanced System Settings** → **Environment Variables**.
2. In the **System Variables** section, find the variable named Path and click **Edit**.
3. Click **New**, then add the path to the folder where MongoDB tools were extracted (e.g., C:\MongoDBTools\bin).
4. Click **OK** to save the changes.

Step 4: Verify Installation

1. Open **Command Prompt** and type:

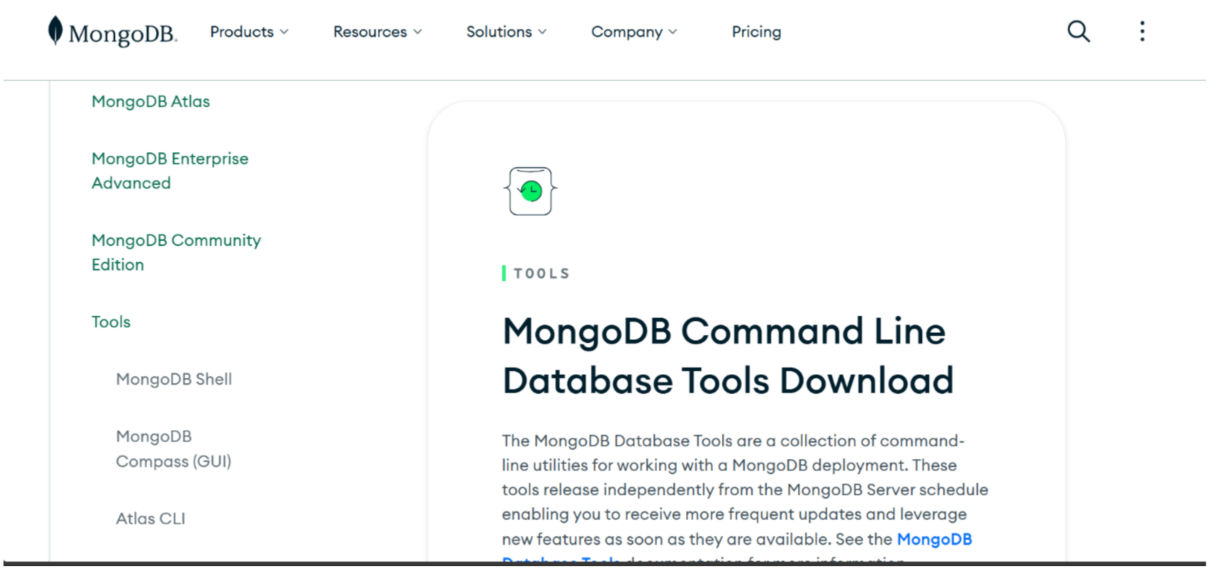
mongodump --version

This should show the version information if the tool is correctly installed.

2. Similarly, check for mongorestore:

mongorestore --version

Now you're ready to use mongodump and mongorestore

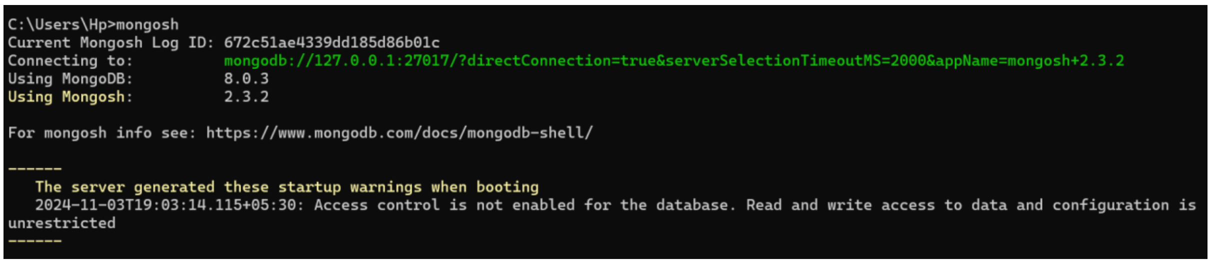


TOOLS

MongoDB Command Line Database Tools Download

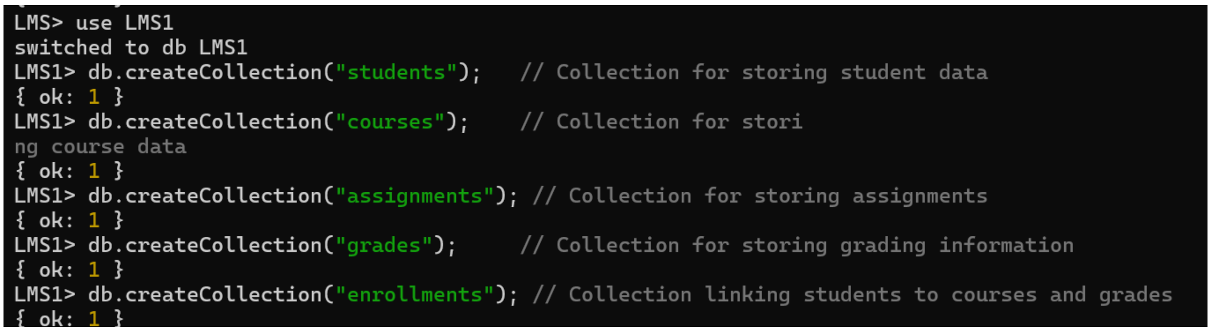
The MongoDB Database Tools are a collection of command-line utilities for working with a MongoDB deployment. These tools release independently from the MongoDB Server schedule enabling you to receive more frequent updates and leverage new features as soon as they are available. See the MongoDB

Type mongosh in command line to check whether installed properly

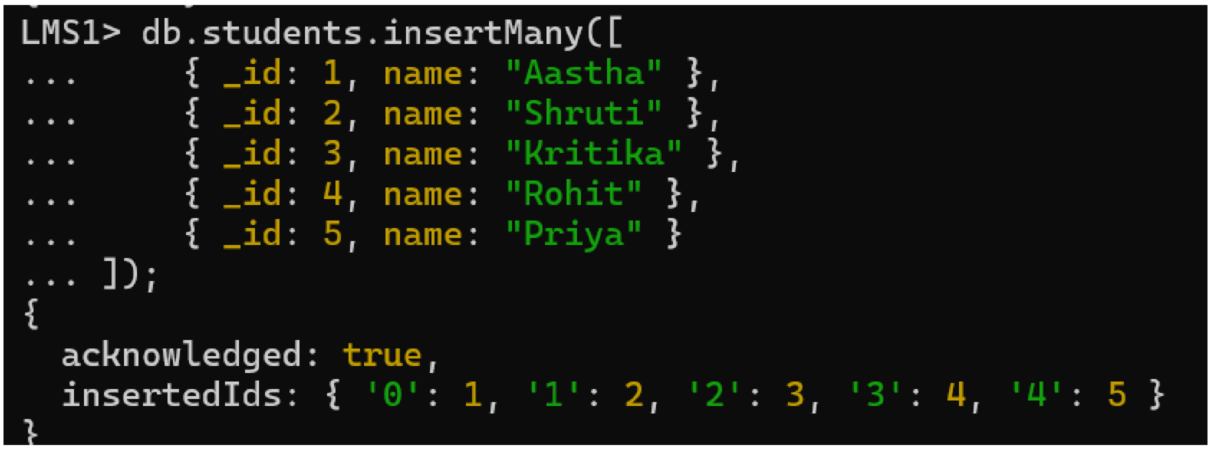


```
C:\Users\Hp>mongosh
Current Mongosh Log ID: 672c51ae4339dd185d86b01c
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.2
Using MongoDB:          8.0.3
Using Mongosh:          2.3.2

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-11-03T19:03:14.115+05:30: Access control is not enabled for the database. Read and write access to data and configuration is
unrestricted
------
```

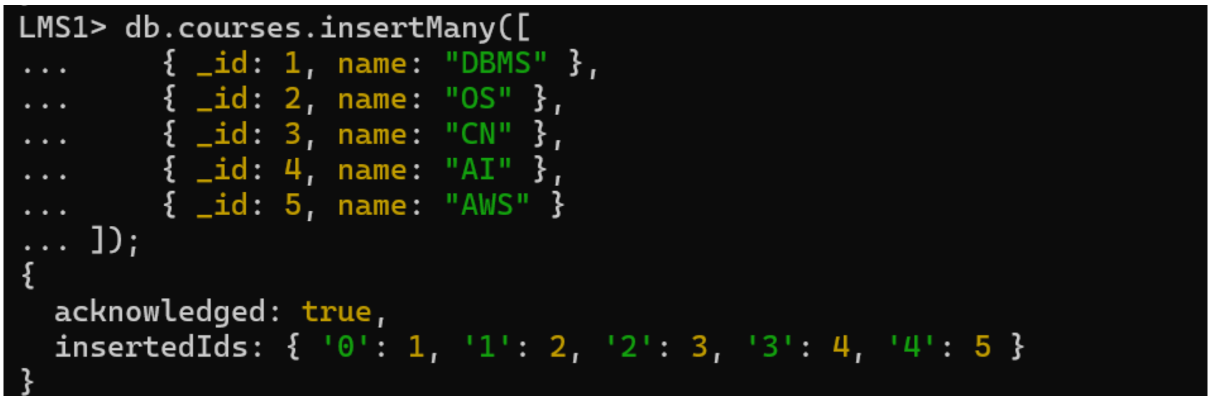
1. **Schema Design**: Design collections for courses, students, assignments, and grades.



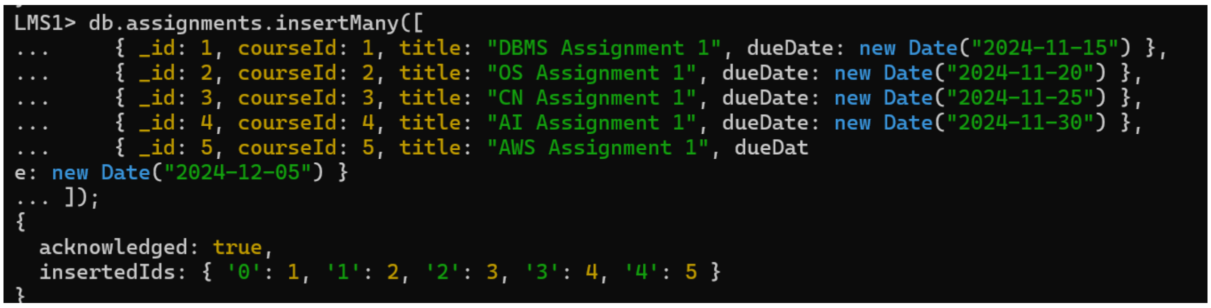
```
LMS> use LMS1
switched to db LMS1
LMS1> db.createCollection("students");   // Collection for storing student data
{ ok: 1 }
LMS1> db.createCollection("courses");    // Collection for stori
ng course data
{ ok: 1 }
LMS1> db.createCollection("assignments"); // Collection for storing assignments
{ ok: 1 }
LMS1> db.createCollection("grades");     // Collection for storing grading information
{ ok: 1 }
LMS1> db.createCollection("enrollments"); // Collection linking students to courses and grades
{ ok: 1 }
```



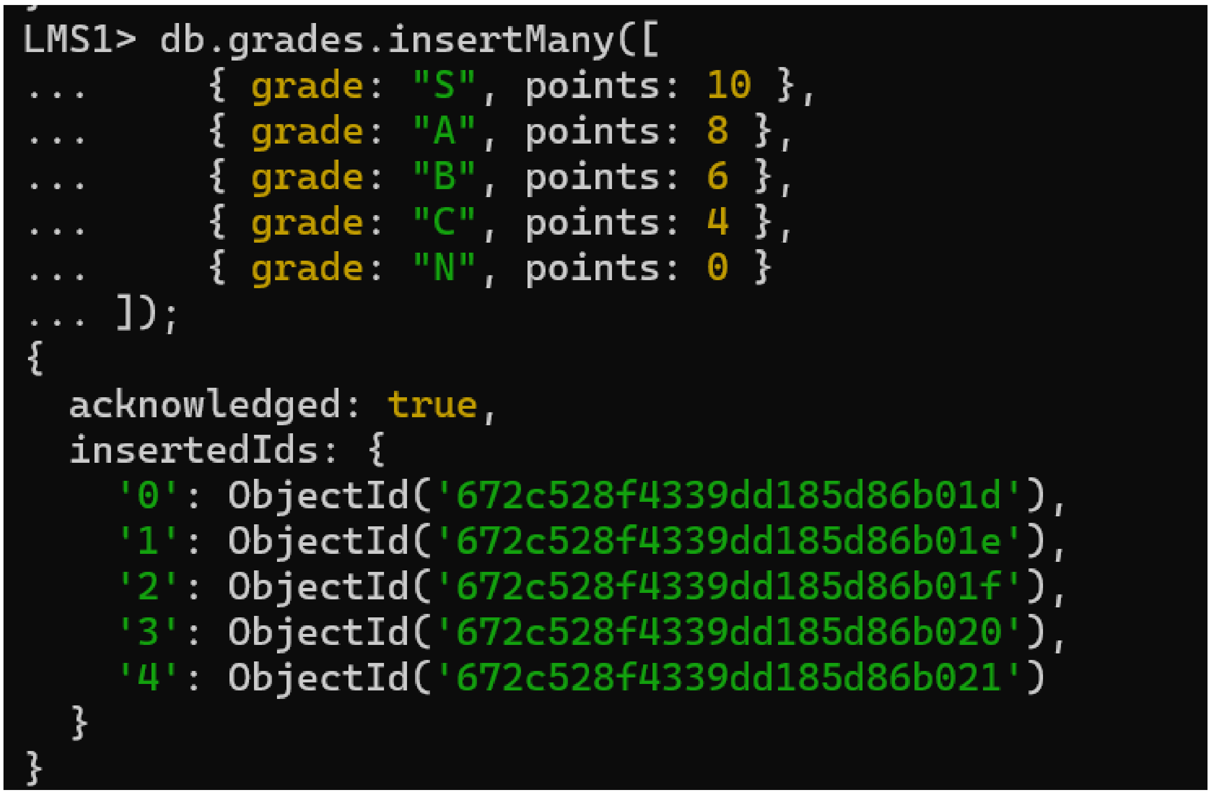
```
LMS1> db.students.insertMany([
...     { _id: 1, name: "Aastha" },
...     { _id: 2, name: "Shruti" },
...     { _id: 3, name: "Kritika" },
...     { _id: 4, name: "Rohit" },
...     { _id: 5, name: "Priya" }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
```



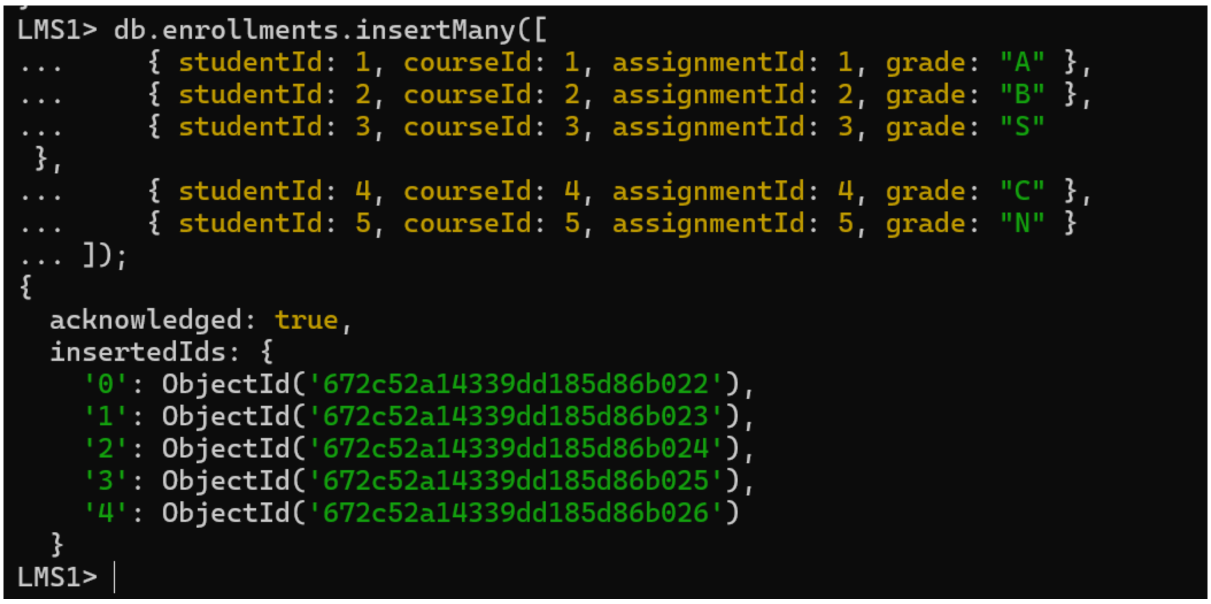
```
LMS1> db.courses.insertMany([
...     { _id: 1, name: "DBMS" },
...     { _id: 2, name: "OS" },
...     { _id: 3, name: "CN" },
...     { _id: 4, name: "AI" },
...     { _id: 5, name: "AWS" }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
```



```
LMS1> db.assignments.insertMany([
...     { _id: 1, courseId: 1, title: "DBMS Assignment 1", dueDate: new Date("2024-11-15") },
...     { _id: 2, courseId: 2, title: "OS Assignment 1", dueDate: new Date("2024-11-20") },
...     { _id: 3, courseId: 3, title: "CN Assignment 1", dueDate: new Date("2024-11-25") },
...     { _id: 4, courseId: 4, title: "AI Assignment 1", dueDate: new Date("2024-11-30") },
...     { _id: 5, courseId: 5, title: "AWS Assignment 1", dueDat
e: new Date("2024-12-05") }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
```



```
LMS1> db.grades.insertMany([
...     { grade: "S", points: 10 },
...     { grade: "A", points: 8 },
...     { grade: "B", points: 6 },
...     { grade: "C", points: 4 },
...     { grade: "N", points: 0 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('672c528f4339dd185d86b01d'),
    '1': ObjectId('672c528f4339dd185d86b01e'),
    '2': ObjectId('672c528f4339dd185d86b01f'),
    '3': ObjectId('672c528f4339dd185d86b020'),
    '4': ObjectId('672c528f4339dd185d86b021')
  }
}
```



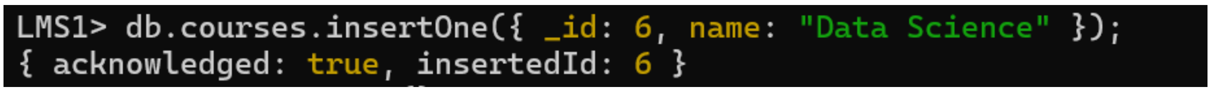
```
LMS1> db.enrollments.insertMany([
...     { studentId: 1, courseId: 1, assignmentId: 1, grade: "A" },
...     { studentId: 2, courseId: 2, assignmentId: 2, grade: "B" },
...     { studentId: 3, courseId: 3, assignmentId: 3, grade: "S"
 },
...     { studentId: 4, courseId: 4, assignmentId: 4, grade: "C" },
...     { studentId: 5, courseId: 5, assignmentId: 5, grade: "N" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('672c52a14339dd185d86b022'),
    '1': ObjectId('672c52a14339dd185d86b023'),
    '2': ObjectId('672c52a14339dd185d86b024'),
    '3': ObjectId('672c52a14339dd185d86b025'),
    '4': ObjectId('672c52a14339dd185d86b026')
  }
LMS1> 
```

2. **CRUD Operations:** Implement CRUD operations for managing courses and student assignments.

CRUD stands for **Create**, **Read**, **Update**, and **Delete**. Here's how each operation works in MongoDB.

a) Create

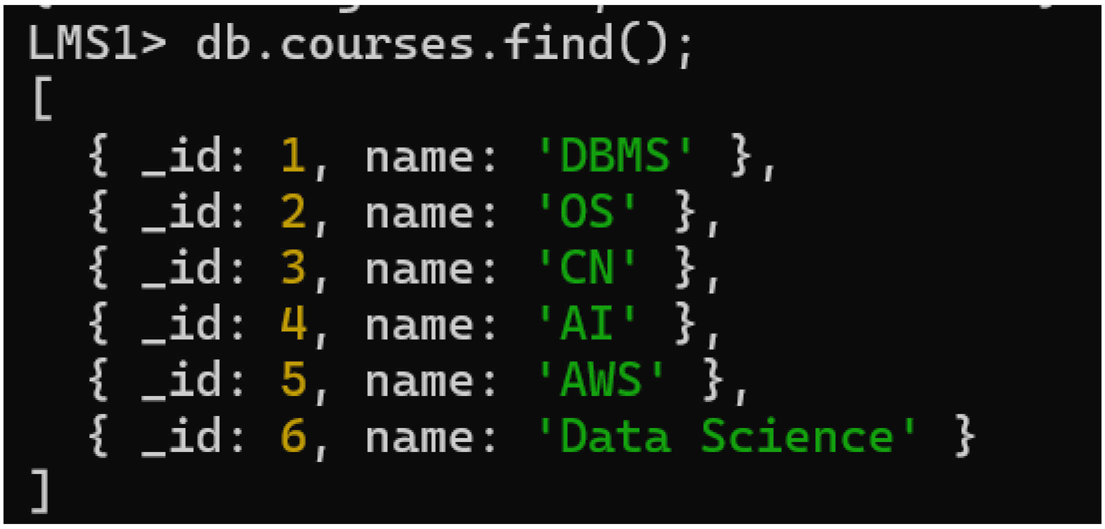
Inserting a new course into the courses collection.



```
LMS1> db.courses.insertOne({ _id: 6, name: "Data Science" });
{ acknowledged: true, insertedId: 6 }
```

b) Read

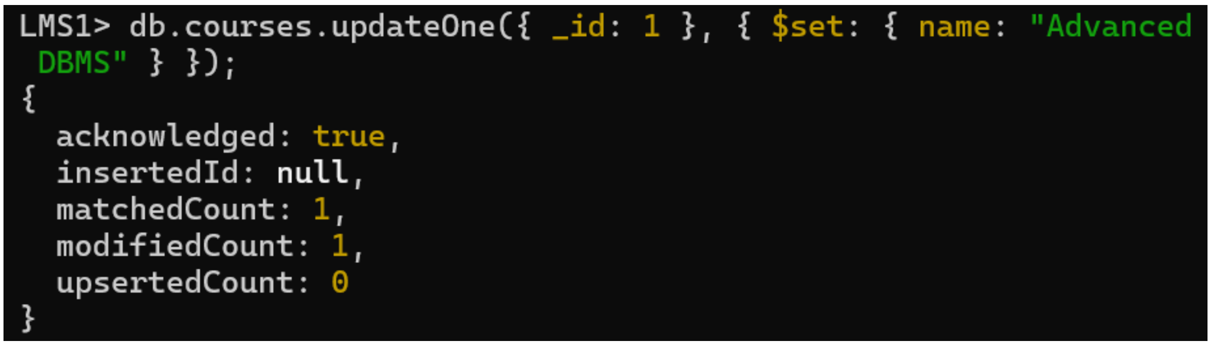
Retrieve all documents in the courses collection.



```
LMS1> db.courses.find();
[
  { _id: 1, name: 'DBMS' },
  { _id: 2, name: 'OS' },
  { _id: 3, name: 'CN' },
  { _id: 4, name: 'AI' },
  { _id: 5, name: 'AWS' },
  { _id: 6, name: 'Data Science' }
]
```

c) Update

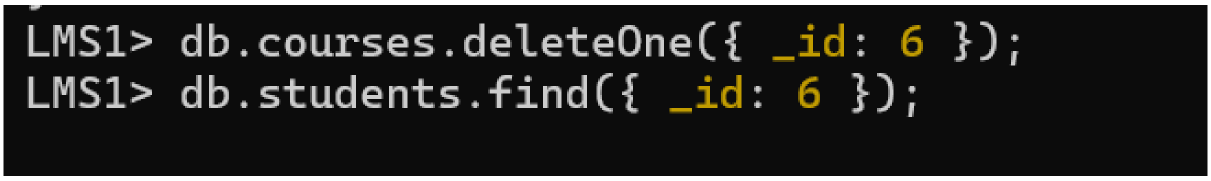
Change the name of a course with _id: 1.



```
LMS1> db.courses.updateOne({ _id: 1 }, { $set: { name: "Advanced
 DBMS" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

d) Delete

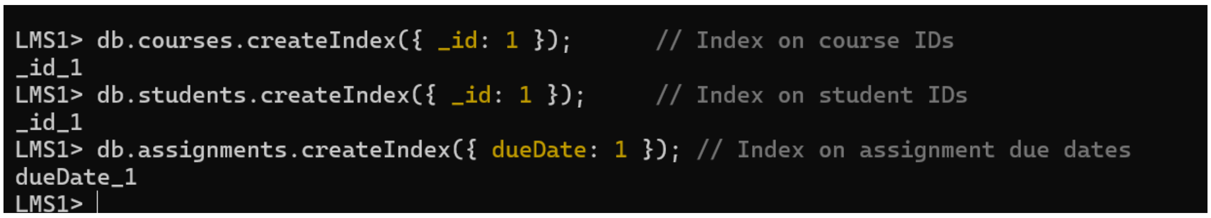
After deleting we check if it is successfully deleted by using find



```
LMS1> db.courses.deleteOne({ _id: 6 });
LMS1> db.students.find({ _id: 6 });
```

3. **Indexing:** Create indexes on course IDs, student IDs, and assignment due dates.

Indexes speed up search operations. Here, we'll create indexes on specific fields.



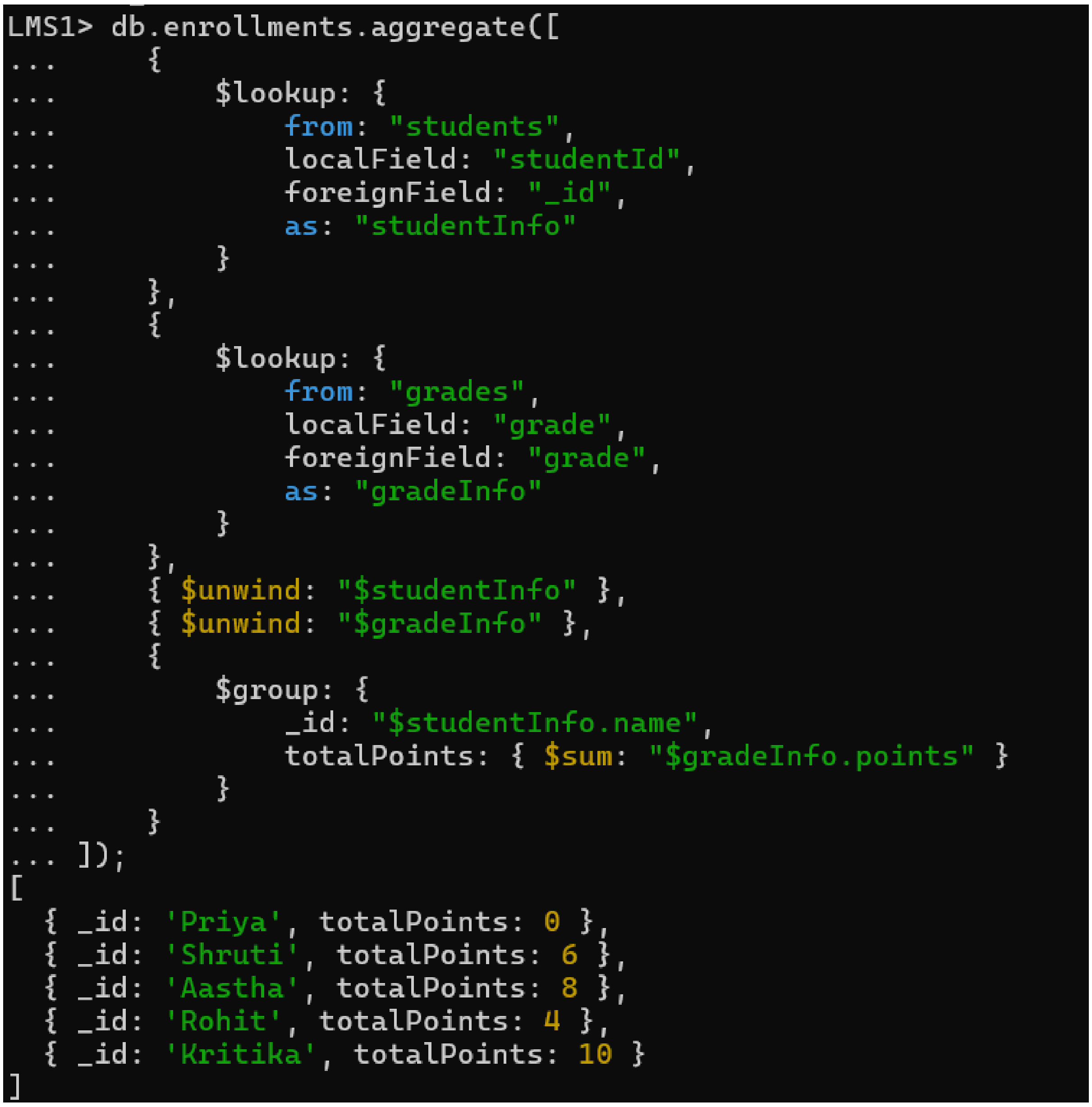
```
LMS1> db.courses.createIndex({ _id: 1 });      // Index on course IDs
_id_1
LMS1> db.students.createIndex({ _id: 1 });     // Index on student IDs
_id_1
LMS1> db.assignments.createIndex({ dueDate: 1 }); // Index on assignment due dates
dueDate_1
LMS1> |
```

4. **Aggregation:** Aggregate data to generate reports on student performance and course enrolment.

Aggregation allows you to process data and return calculated results. This example calculates the total points each student has earned based on grades.

Explanation:

- $lookup joins collections.
- $unwind converts arrays to single objects for easier processing.
- $group groups by student name and calculates totalPoints.



```
LMS1> db.enrollments.aggregate([
...     {
...         $lookup: {
...             from: "students",
...             localField: "studentId",
...             foreignField: "_id",
...             as: "studentInfo"
...         }
...     },
...     {
...         $lookup: {
...             from: "grades",
...             localField: "grade",
...             foreignField: "grade",
...             as: "gradeInfo"
...         }
...     },
...     { $unwind: "$studentInfo" },
...     { $unwind: "$gradeInfo" },
...     {
...         $group: {
...             _id: "$studentInfo.name",
...             totalPoints: { $sum: "$gradeInfo.points" }
...         }
...     }
... ]);
[
  { _id: 'Priya', totalPoints: 0 },
  { _id: 'Shruti', totalPoints: 6 },
  { _id: 'Aastha', totalPoints: 8 },
  { _id: 'Rohit', totalPoints: 4 },
  { _id: 'Kritika', totalPoints: 10 }
]
```

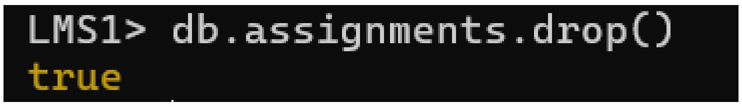
5. **Data Validation:** Set up validation rules for assignments and grades.

Set rules to ensure data integrity. MongoDB provides schema validation at the collection level. We will set up validation rules for the assignments and grades collections.

Create Validation Rules for assignments Collection

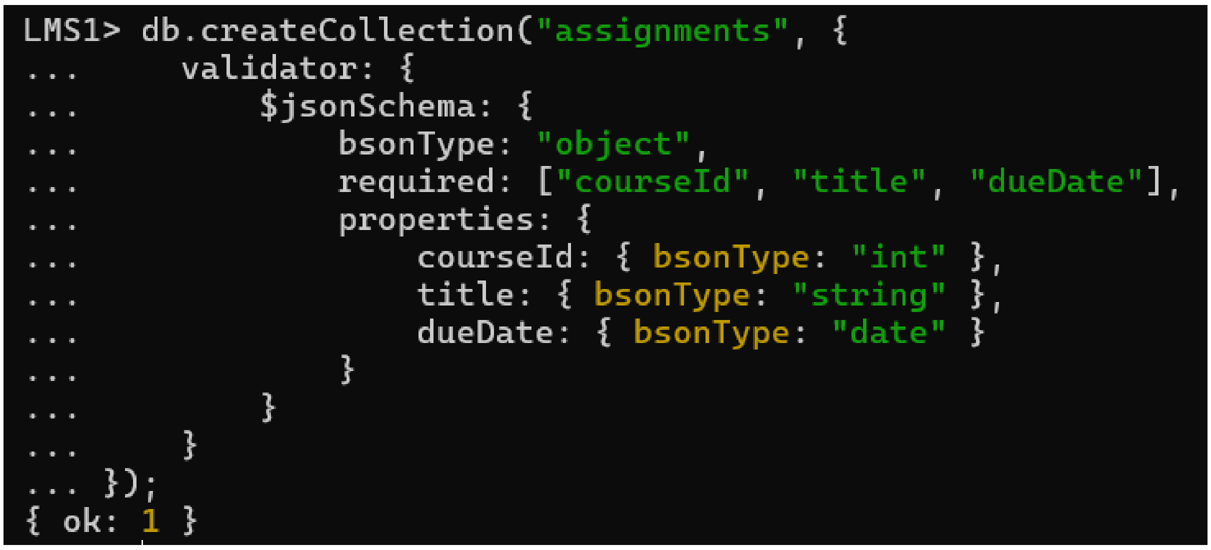
We will ensure that every assignment has the correct fields, and the dueDate must be a valid date in the future.

1. **Drop the existing assignments collection** (since validation rules can only be applied when creating a collection):



```
LMS1> db.assignments.drop()
true
```

2. **Create the assignments collection with validation rules**:

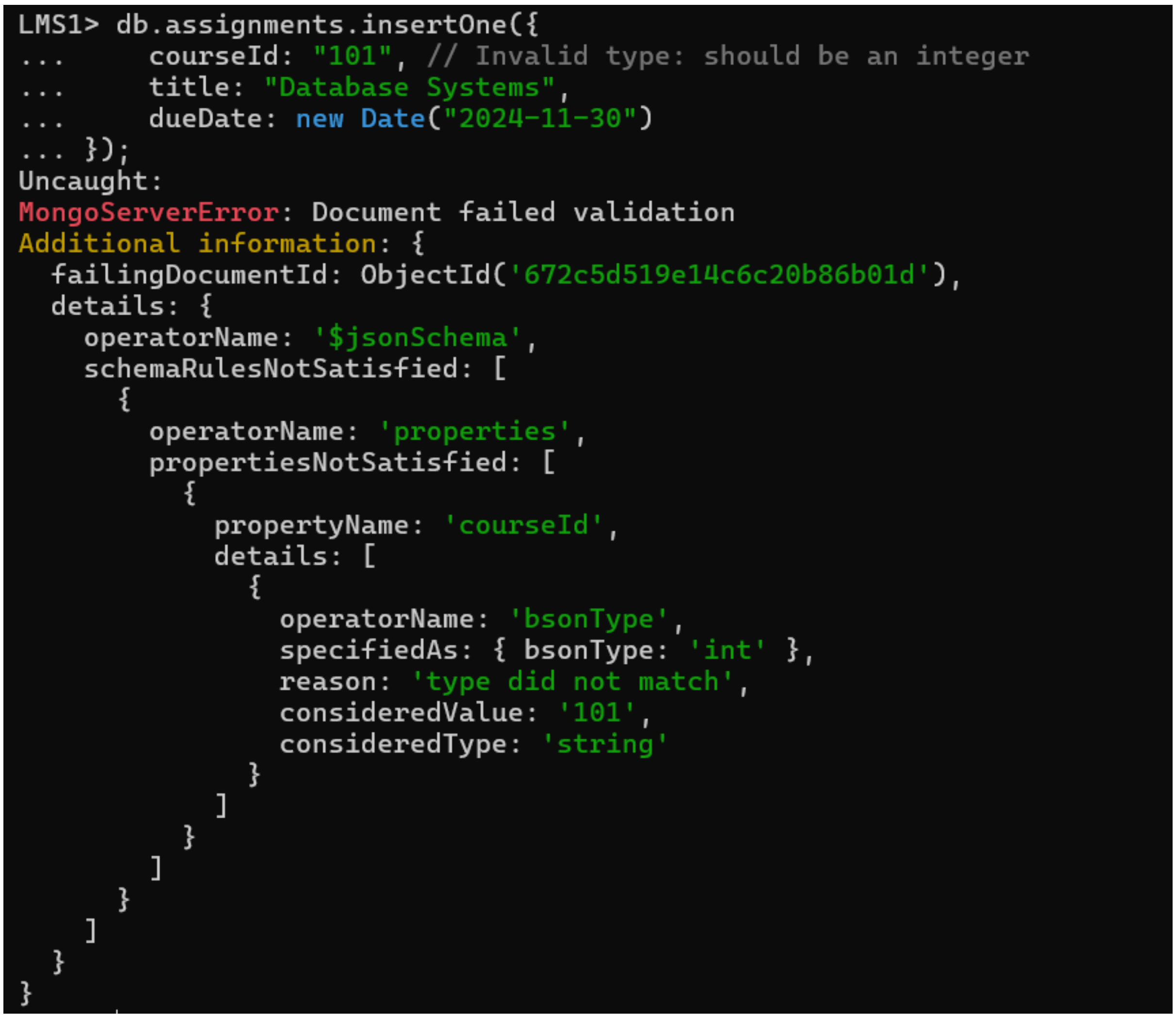


```
LMS1> db.createCollection("assignments", {
...     validator: {
...         $jsonSchema: {
...             bsonType: "object",
...             required: ["courseId", "title", "dueDate"],
...             properties: {
...                 courseId: { bsonType: "int" },
...                 title: { bsonType: "string" },
...                 dueDate: { bsonType: "date" }
...             }
...         }
...     }
... });
{ ok: 1 }
```

- **Validator**: This is the main part of the code that tells MongoDB to enforce data validation rules.
- **$jsonSchema**: This specifies that the validation rules are based on a JSON schema, which is a standard format for defining the structure of JSON data. **MongoDB uses JSON schema to validate the types, required fields, and format of each document.**
- **bsonType**: This specifies the data type each field should have:
  - **"object"** for the whole document, meaning each document in this collection should be a JSON object (not, for example, an array or a simple string).
  - Inside the document, each field (courseId, title, dueDate) also has a bsonType:
    - courseId must be an **integer** ("int").
    - title must be a **string**.
    - dueDate must be a **date**.

- **Required Fields**: The required keyword specifies fields that must be present in every document inserted into the collection. In this example:
  - Every document in assignments must contain courseId, title, and dueDate fields.
  - If any of these fields are missing in a document, MongoDB will reject the document during insert or update operations.
- **Properties**: The properties keyword defines the rules for individual fields:
  - Inside properties, each field (such as courseId, title, and dueDate) is listed with its own bsonType validation.
  - This ensures that each field has the correct data type. If a document doesn't match these rules, MongoDB will reject it.

To check if Data Validation working or not



```
LMS1> db.assignments.insertOne({
...     courseId: "101", // Invalid type: should be an integer
...     title: "Database Systems",
...     dueDate: new Date("2024-11-30")
... });
Uncaught:
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId('672c5d519e14c6c20b86b01d'),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'properties',
        propertiesNotSatisfied: [
          {
            propertyName: 'courseId',
            details: [
              {
                operatorName: 'bsonType',
                specifiedAs: { bsonType: 'int' },
                reason: 'type did not match',
                consideredValue: '101',
                consideredType: 'string'
              }
            ]
          }
        ]
      }
    ]
  }
}
```

As courseID is int and given datatype is string so data validation not applied therefore MongoDB gives error.

6. **Backup and Restore:** Manage backups of course and student data.

Purpose

Backing up your database is essential for data safety and disaster recovery. If data is accidentally deleted, corrupted, or if there's a hardware failure, backups allow you to restore data to its previous state.

How to Backup

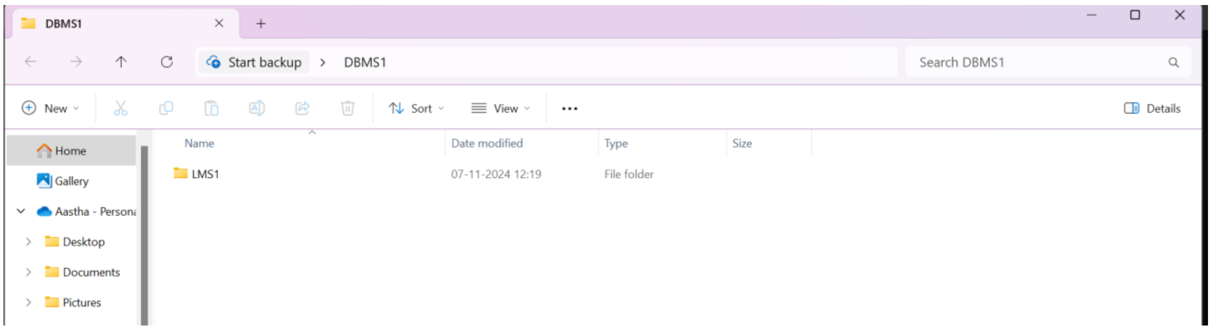
In MongoDB, there are different ways to perform backups. Here's how you can do it:

Using mongodump Command:

- mongodump is a MongoDB tool that creates binary backups of your database.

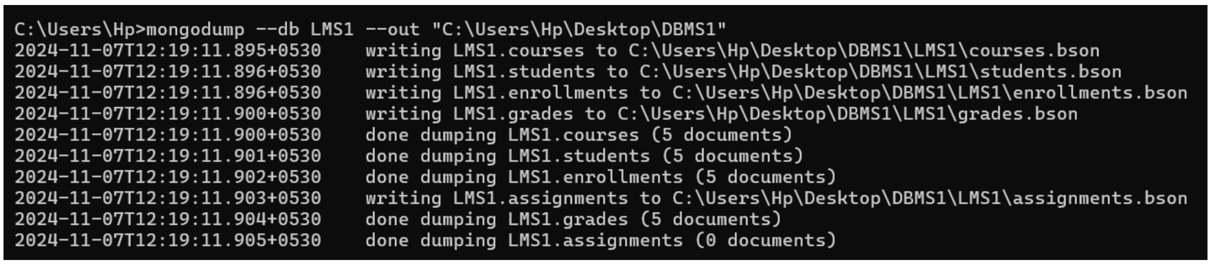
Here's how to use mongodump in your **command prompt** or **PowerShell**:

1. **Exit the MongoDB Shell** by typing exit and pressing Enter, if you're still in the MongoDB Shell



Name	Date modified	Type	Size
LMS1	07-11-2024 12:19	File folder	

Run the following command to create a backup of your LMS1 database. Make sure to include double quotes around the output path if it contains spaces:



```
C:\Users\Hp>mongodump --db LMS1 --out "C:\Users\Hp\Desktop\DBMS1"
2024-11-07T12:19:11.895+0530    writing LMS1.courses to C:\Users\Hp\Desktop\DBMS1\LMS1\courses.bson
2024-11-07T12:19:11.896+0530    writing LMS1.students to C:\Users\Hp\Desktop\DBMS1\LMS1\students.bson
2024-11-07T12:19:11.896+0530    writing LMS1.enrollments to C:\Users\Hp\Desktop\DBMS1\LMS1\enrollments.bson
2024-11-07T12:19:11.900+0530    writing LMS1.grades to C:\Users\Hp\Desktop\DBMS1\LMS1\grades.bson
2024-11-07T12:19:11.900+0530    done dumping LMS1.courses (5 documents)
2024-11-07T12:19:11.901+0530    done dumping LMS1.students (5 documents)
2024-11-07T12:19:11.902+0530    done dumping LMS1.enrollments (5 documents)
2024-11-07T12:19:11.903+0530    writing LMS1.assignments to C:\Users\Hp\Desktop\DBMS1\LMS1\assignments.bson
2024-11-07T12:19:11.904+0530    done dumping LMS1.grades (5 documents)
2024-11-07T12:19:11.905+0530    done dumping LMS1.assignments (0 documents)
```

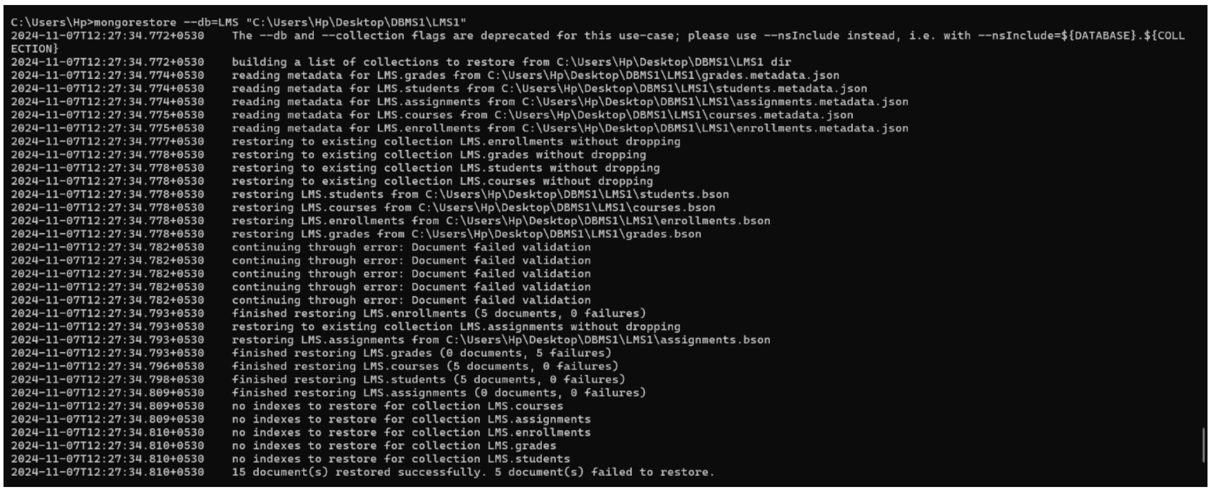
Here LMS1 Folder stored in specified location that is dbms

<u>How to Restore</u>

To restore a backup:

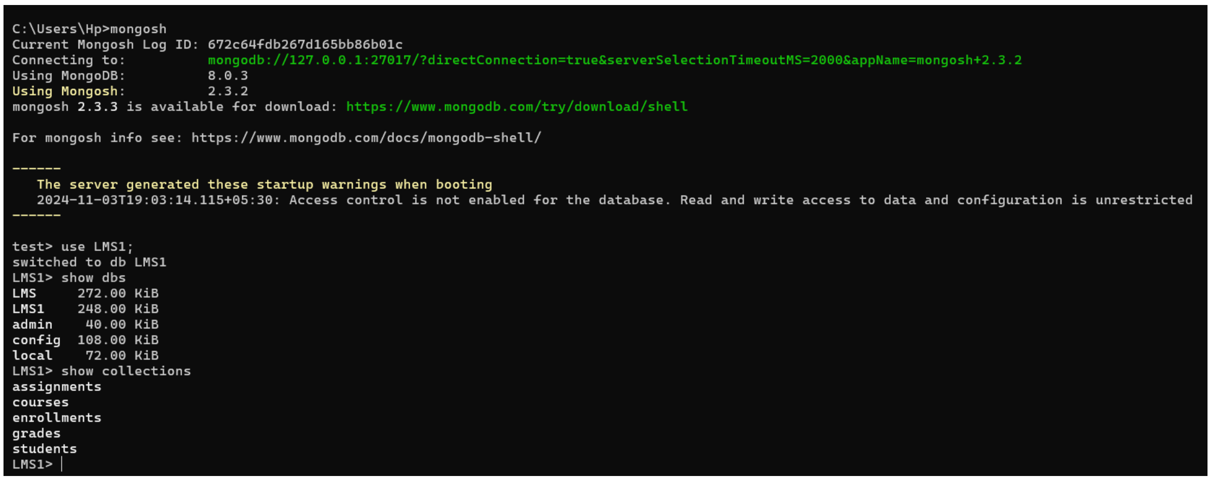
Using mongorestore Command:

- mongorestore is a MongoDB tool that restores data from a backup created by mongodump.
- To restore the LMS1 database, use the following command:



```
C:\Users\Hp>mongorestore --db=LMS "C:\Users\Hp\Desktop\DBMS1\LMS1"
2024-11-07T12:27:34.772+0530    The --db and --collection flags are deprecated for this use-case; please use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLL
ECTION}
2024-11-07T12:27:34.772+0530    building a list of collections to restore from C:\Users\Hp\Desktop\DBMS1\LMS1 dir
2024-11-07T12:27:34.774+0530    reading metadata for LMS.grades from C:\Users\Hp\Desktop\DBMS1\LMS1\grades.metadata.json
2024-11-07T12:27:34.774+0530    reading metadata for LMS.students from C:\Users\Hp\Desktop\DBMS1\LMS1\students.metadata.json
2024-11-07T12:27:34.774+0530    reading metadata for LMS.assignments from C:\Users\Hp\Desktop\DBMS1\LMS1\assignments.metadata.json
2024-11-07T12:27:34.775+0530    reading metadata for LMS.courses from C:\Users\Hp\Desktop\DBMS1\LMS1\courses.metadata.json
2024-11-07T12:27:34.775+0530    reading metadata for LMS.enrollments from C:\Users\Hp\Desktop\DBMS1\LMS1\enrollments.metadata.json
2024-11-07T12:27:34.777+0530    restoring to existing collection LMS.enrollments without dropping
2024-11-07T12:27:34.778+0530    restoring to existing collection LMS.grades without dropping
2024-11-07T12:27:34.778+0530    restoring to existing collection LMS.students without dropping
2024-11-07T12:27:34.778+0530    restoring to existing collection LMS.courses without dropping
2024-11-07T12:27:34.778+0530    restoring LMS.students from C:\Users\Hp\Desktop\DBMS1\LMS1\students.bson
2024-11-07T12:27:34.778+0530    restoring LMS.courses from C:\Users\Hp\Desktop\DBMS1\LMS1\courses.bson
2024-11-07T12:27:34.778+0530    restoring LMS.enrollments from C:\Users\Hp\Desktop\DBMS1\LMS1\enrollments.bson
2024-11-07T12:27:34.778+0530    restoring LMS.grades from C:\Users\Hp\Desktop\DBMS1\LMS1\grades.bson
2024-11-07T12:27:34.782+0530    continuing through error: Document failed validation
2024-11-07T12:27:34.782+0530    continuing through error: Document failed validation
2024-11-07T12:27:34.782+0530    continuing through error: Document failed validation
2024-11-07T12:27:34.782+0530    continuing through error: Document failed validation
2024-11-07T12:27:34.782+0530    continuing through error: Document failed validation
2024-11-07T12:27:34.793+0530    finished restoring LMS.enrollments (5 documents, 0 failures)
2024-11-07T12:27:34.793+0530    restoring to existing collection LMS.assignments without dropping
2024-11-07T12:27:34.793+0530    restoring LMS.assignments from C:\Users\Hp\Desktop\DBMS1\LMS1\assignments.bson
2024-11-07T12:27:34.793+0530    finished restoring LMS.grades (0 documents, 5 failures)
2024-11-07T12:27:34.796+0530    finished restoring LMS.courses (5 documents, 0 failures)
2024-11-07T12:27:34.798+0530    finished restoring LMS.students (5 documents, 0 failures)
2024-11-07T12:27:34.809+0530    finished restoring LMS.assignments (0 documents, 0 failures)
2024-11-07T12:27:34.809+0530    no indexes to restore for collection LMS.courses
2024-11-07T12:27:34.809+0530    no indexes to restore for collection LMS.assignments
2024-11-07T12:27:34.810+0530    no indexes to restore for collection LMS.enrollments
2024-11-07T12:27:34.810+0530    no indexes to restore for collection LMS.grades
2024-11-07T12:27:34.810+0530    no indexes to restore for collection LMS.students
2024-11-07T12:27:34.810+0530    15 document(s) restored successfully. 5 document(s) failed to restore.
```

Verifying if restored performed successfully



```
C:\Users\Hp>mongosh
Current Mongosh Log ID: 672c64fdb267d165bb86b01c
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.2
Using MongoDB:          8.0.3
Using Mongosh:          2.3.2
mongosh 2.3.3 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-11-03T19:03:14.115+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use LMS1;
switched to db LMS1
LMS1> show dbs
LMS     272.00 KiB
LMS1    248.00 KiB
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
LMS1> show collections
assignments
courses
enrollments
grades
students
LMS1> |
```

The **mongorestore** function is used to restore MongoDB data from a backup (usually in BSON and metadata format) into a running MongoDB instance.

What does mongorestore do?

- **Restores data**: It takes the BSON files (which contain the actual data) and metadata files from the backup directory and inserts them into the corresponding MongoDB collections.
- **Does not delete the backup**: The backup files in the folder you specify (e.g., C:\Users\Hp\Desktop\DBMS1) **will not be deleted** after running mongorestore. The tool **only restores data** into the database and does not alter or remove any files in the backup directory.

What happens during the restore?

- If you use the **--drop** flag, it will drop (delete) the existing collections in the target database before restoring the data. This is useful for avoiding conflicts like duplicate keys.
- If you don't use the --drop flag, it will attempt to insert data into the existing collections. If there are duplicate _id values, you'll see errors unless handled otherwise (e.g., using --noIndexRestore).

Example:

Let's say you execute the command:

mongorestore --drop --nsInclude=LMS1.* "C:\Users\Hp\Desktop\DBMS1"

- This will drop (delete) any existing collections in the LMS1 database before restoring data from the backup directory.
- The backup folder C:\Users\Hp\Desktop\DBMS1 will remain unchanged, and the files will not be deleted or modified.

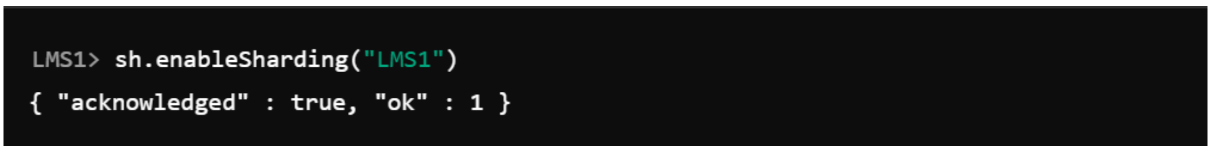
7. Sharding: Configure sharding to handle large volumes of student and course data.

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations. Database systems with large data sets or high throughput applications can challenge the capacity of a single server.

Step-by-Step Guide to Shard the assignments Collection:

1. **Enable Sharding for Database LMS1:**

First, enable sharding on the LMS1 database. This allows MongoDB to distribute collections from this database across multiple shards.



```
LMS1> sh.enableSharding("LMS1")
{ "acknowledged" : true, "ok" : 1 }
```

2. **Choose a Shard Key and Index on assignments Collection:**

Select a shard key for the assignments collection. For example, let’s use courseId as the shard key. Ensure that an index exists on this field.

```
LMS1> use LMS1
switched to db LMS1

LMS1> db.assignments.createIndex({ courseId: "hashed" })
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

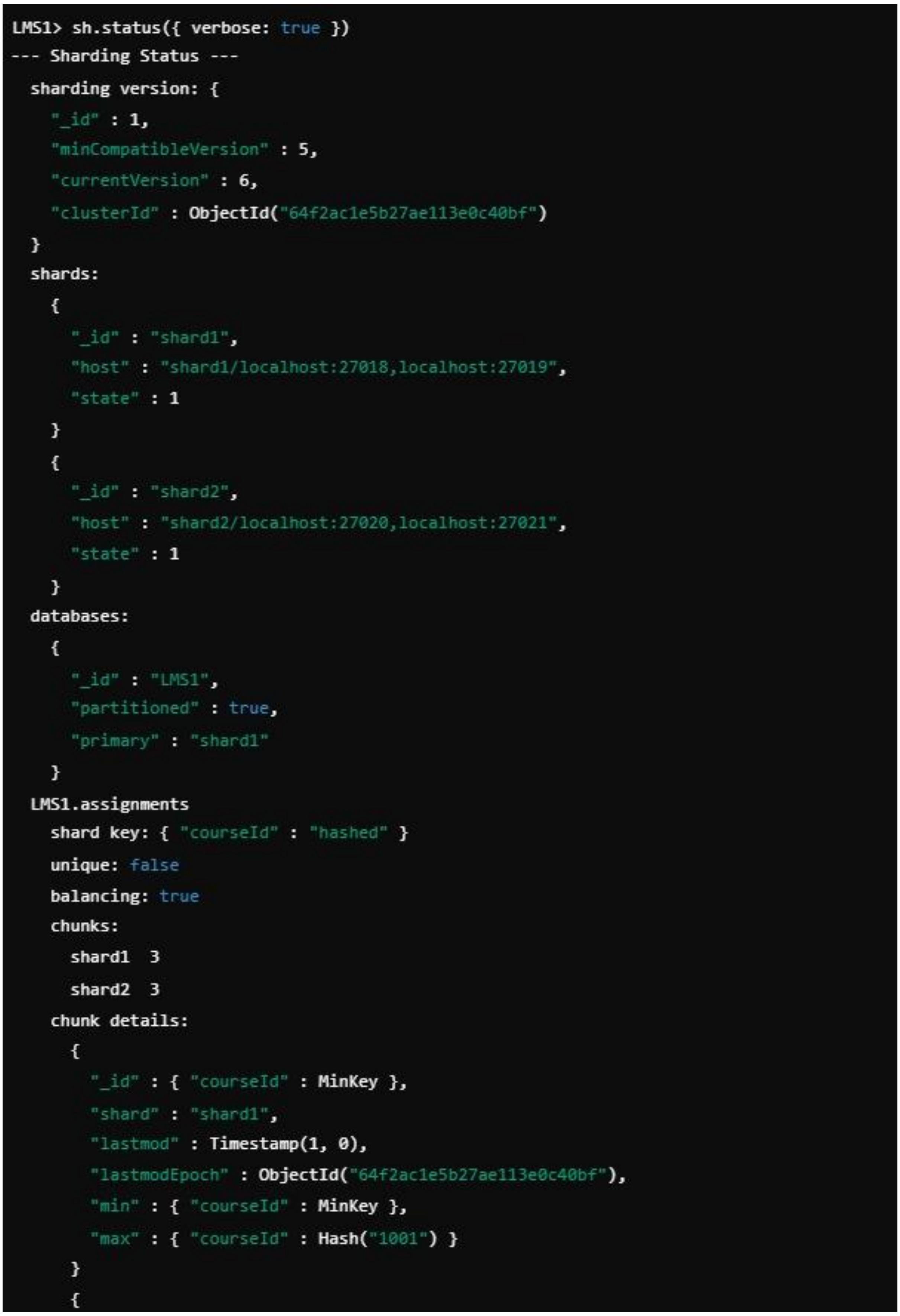
3. **Shard the Collection:**

Now, shard the assignments collection using the courseId field as the shard key. A hashed shard key will distribute documents evenly across shards.

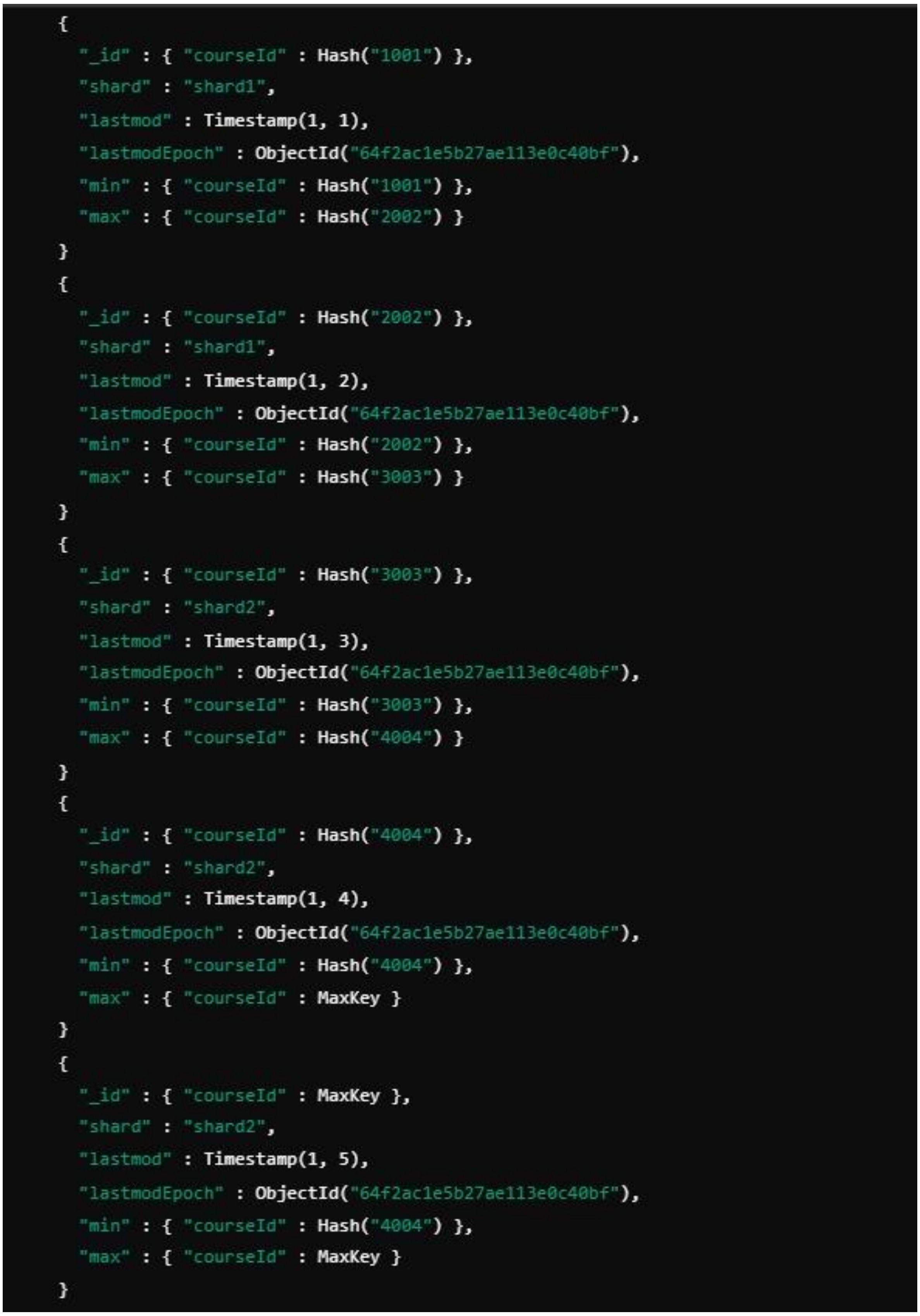
```
LMS1> sh.shardCollection("LMS1.assignments", { courseId: "hashed" })
{ "collectionsharded" : "LMS1.assignments", "ok" : 1 }
```

4. **Verify Sharding Status:**

Check the sharding status to confirm the configuration. This will display the shard distribution and the collections that are sharded.



```
LMS1> sh.status({ verbose: true })
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("64f2ac1e5b27ae113e0c40bf")
  }
  shards:
    {
      "_id" : "shard1",
      "host" : "shard1/localhost:27018,localhost:27019",
      "state" : 1
    }
    {
      "_id" : "shard2",
      "host" : "shard2/localhost:27020,localhost:27021",
      "state" : 1
    }
  databases:
    {
      "_id" : "LMS1",
      "partitioned" : true,
      "primary" : "shard1"
    }
  LMS1.assignments
    shard key: { "courseId" : "hashed" }
    unique: false
    balancing: true
    chunks:
      shard1  3
      shard2  3
    chunk details:
      {
        "_id" : { "courseId" : MinKey },
        "shard" : "shard1",
        "lastmod" : Timestamp(1, 0),
        "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
        "min" : { "courseId" : MinKey },
        "max" : { "courseId" : Hash("1001") }
      }
      {
```



```
{
  "_id" : { "courseId" : Hash("1001") },
  "shard" : "shard1",
  "lastmod" : Timestamp(1, 1),
  "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
  "min" : { "courseId" : Hash("1001") },
  "max" : { "courseId" : Hash("2002") }
}
{
  "_id" : { "courseId" : Hash("2002") },
  "shard" : "shard1",
  "lastmod" : Timestamp(1, 2),
  "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
  "min" : { "courseId" : Hash("2002") },
  "max" : { "courseId" : Hash("3003") }
}
{
  "_id" : { "courseId" : Hash("3003") },
  "shard" : "shard2",
  "lastmod" : Timestamp(1, 3),
  "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
  "min" : { "courseId" : Hash("3003") },
  "max" : { "courseId" : Hash("4004") }
}
{
  "_id" : { "courseId" : Hash("4004") },
  "shard" : "shard2",
  "lastmod" : Timestamp(1, 4),
  "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
  "min" : { "courseId" : Hash("4004") },
  "max" : { "courseId" : MaxKey }
}
{
  "_id" : { "courseId" : MaxKey },
  "shard" : "shard2",
  "lastmod" : Timestamp(1, 5),
  "lastmodEpoch" : ObjectId("64f2ac1e5b27ae113e0c40bf"),
  "min" : { "courseId" : Hash("4004") },
  "max" : { "courseId" : MaxKey }
}
```

8. **Change Streams:** Monitor changes to assignment submissions and grades.

Change Streams in MongoDB allow applications to access real-time data changes without the complexity and overhead of polling the database. They provide a powerful and efficient way to listen to changes in collections, databases, or entire clusters.

To monitor real-time changes using MongoDB Compass and execute a Change Stream on the assignments collection in your LMS1 database, you can follow these steps and view the output directly in Compass:

Steps to Monitor Changes in MongoDB Compass:

1. Launch MongoDB Compass on your system.
2. Enter your MongoDB connection string (URI) in the Compass connection dialog and click Connect.

   Example URI for local MongoDB: mongodb://localhost:27017.

3. Select the LMS1 Database
4. Once connected, you'll see the list of databases on the left-hand sidebar. Click on LMS1 to select it.
5. Open the assignments Collection. In the list of collections under LMS1, click on the assignments collection.
6. Go to the Aggregations Tab. Click on the Aggregations tab at the top of the assignments collection view.
7. Set Up the Change Stream Aggregation:

In the Aggregations tab, you'll need to create an aggregation pipeline that activates the change stream.

In the pipeline builder, add the following stage:

```
{ "$changeStream": { "fullDocument": "updateLookup" } }
```

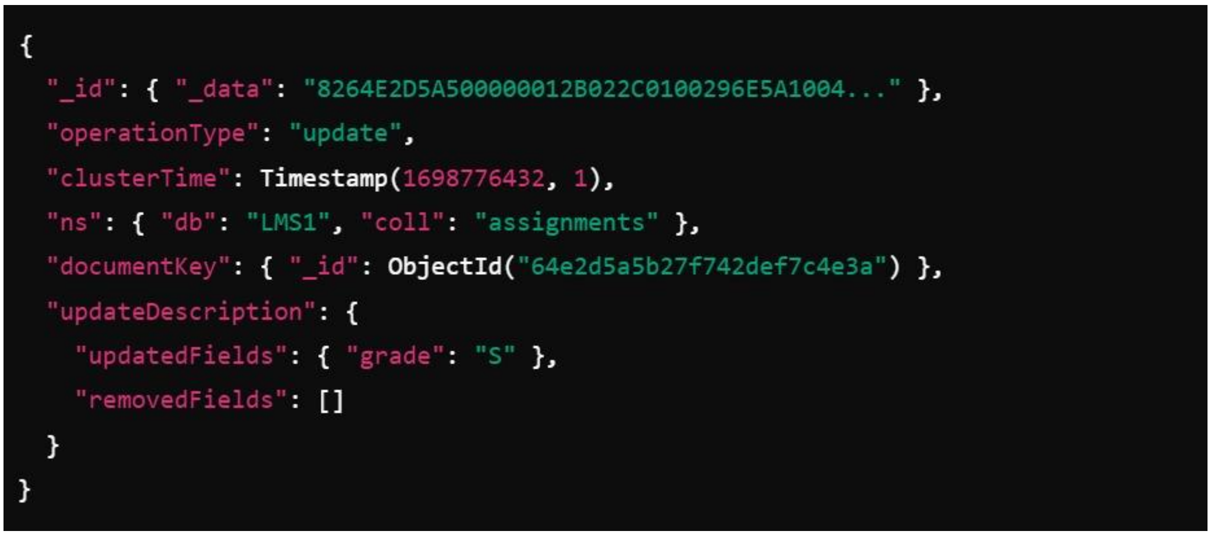
This pipeline opens a change stream to monitor insert, update, and delete operations in the assignments collection.

Run the Aggregation:

Click Run. The aggregation will start listening for changes on the assignments collection.

Example Output in MongoDB Compass:

After running the pipeline, as soon as a change (like an insert, update, or delete) occurs, Compass will display the change event in real-time. Below is an example of how the output might look when a grade is updated:



```
{
  "_id": { "_data": "8264E2D5A500000012B022C0100296E5A1004..." },
  "operationType": "update",
  "clusterTime": Timestamp(1698776432, 1),
  "ns": { "db": "LMS1", "coll": "assignments" },
  "documentKey": { "_id": ObjectId("64e2d5a5b27f742def7c4e3a") },
  "updateDescription": {
    "updatedFields": { "grade": "S" },
    "removedFields": []
  }
}
```

This example shows:

An update operation in the assignments collection.

The field "grade" was updated to S.

The specific document that was updated is referenced by its _id.

9. **Text Search:** Enable full-text search for course descriptions and assignment details.

MongoDB Text Search is a powerful feature designed to find specified text within string content in **MongoDB** collections. This capability is essential for applications that require **efficient** and effective searching capabilities.

Step 1: Switching to the Target Database

```
use LMS1
```

This command switches to the database named LMS1. If it does not exist, MongoDB creates the database implicitly once you add data to it.

Step 2: Creating a Text Index on a Collection

```
db.courses.createIndex({ name: "text" })
```

Here, we are creating a text index on the name field in the courses collection. Text indexes are required for performing text searches in MongoDB. The field being indexed here is name, and the index type is text.

Explanation of the Result:

"createdCollectionAutomatically": false: - This means that the collection already exists, so MongoDB did not create it automatically.

"numIndexesBefore": 1: - The number of indexes that existed before adding this new text index.

"numIndexesAfter": 2: - The number of indexes after the text index was created.

"ok": 1: - Indicates the success of the index creation operation.

Step 3: Performing a Text Search

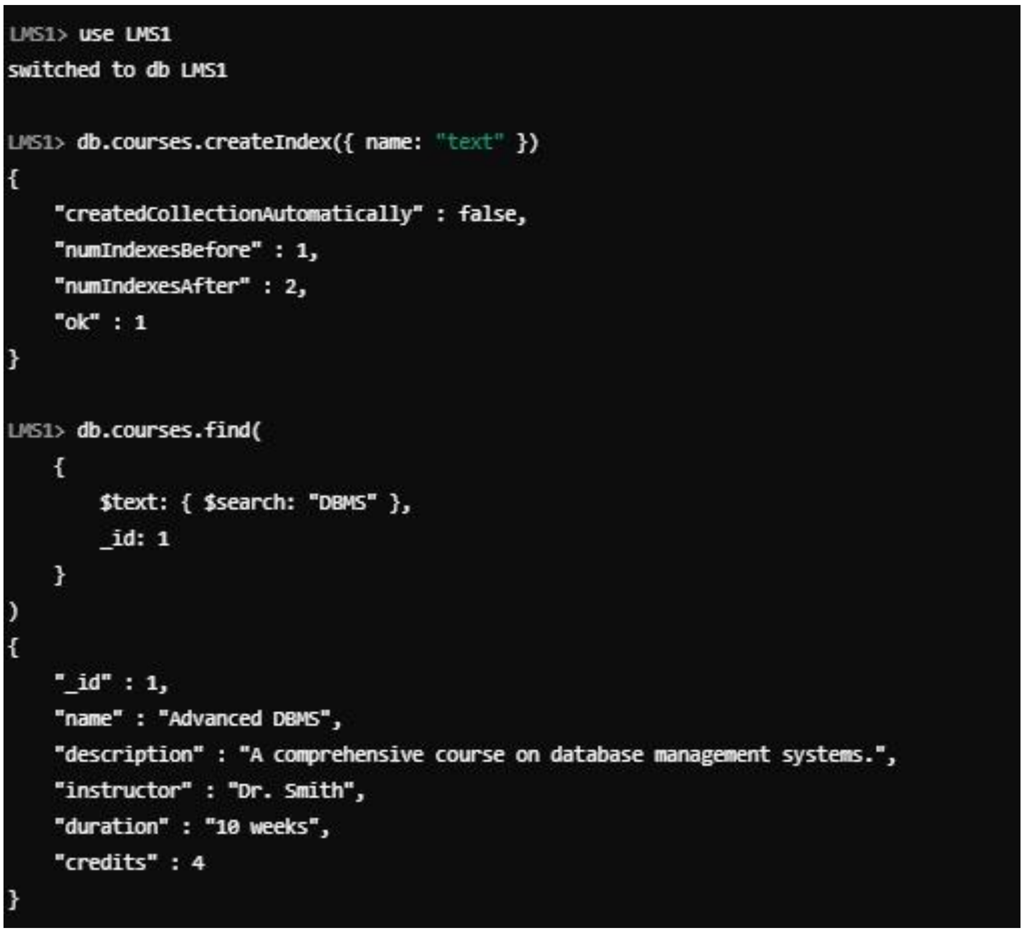
The $text operator is used to perform a text search on the indexed field (name).

The $search operator specifies the string to search for, which in this case is "DBMS". MongoDB searches for documents that contain the word "DBMS" in the name field.

The _id: 1 is included to only return documents with _id equal to 1.

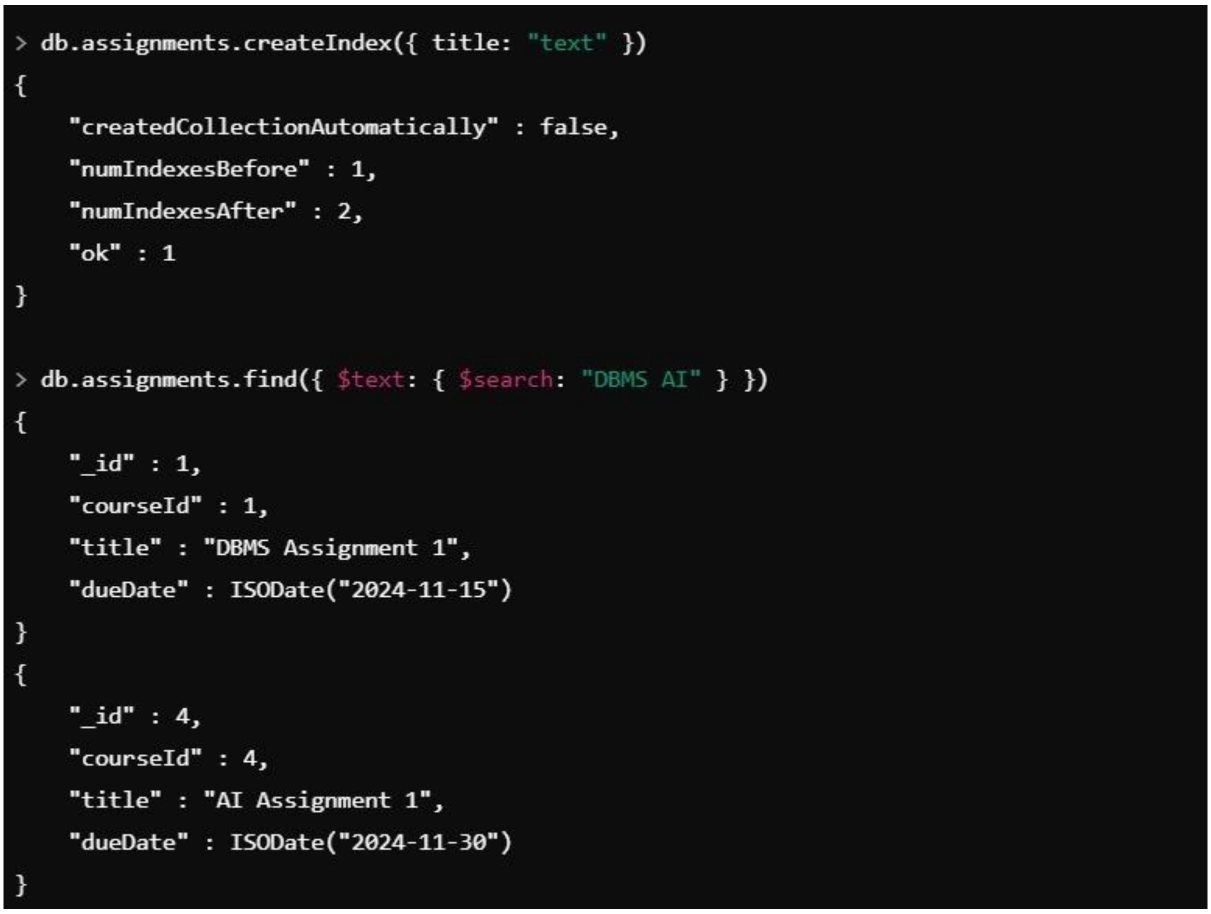
Step 4: Output of the Query

The query returns the following document:



```
LMS1> use LMS1
switched to db LMS1

LMS1> db.courses.createIndex({ name: "text" })
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}

LMS1> db.courses.find(
    {
        $text: { $search: "DBMS" },
        _id: 1
    }
)
{
    "_id" : 1,
    "name" : "Advanced DBMS",
    "description" : "A comprehensive course on database management systems.",
    "instructor" : "Dr. Smith",
    "duration" : "10 weeks",
    "credits" : 4
}
```

This is the document where the name field contains the word "DBMS." MongoDB found this document by scanning the text index created on the name field.



```
> db.assignments.createIndex({ title: "text" })
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}

> db.assignments.find({ $text: { $search: "DBMS AI" } })
{
    "_id" : 1,
    "courseId" : 1,
    "title" : "DBMS Assignment 1",
    "dueDate" : ISODate("2024-11-15")
}
{
    "_id" : 4,
    "courseId" : 4,
    "title" : "AI Assignment 1",
    "dueDate" : ISODate("2024-11-30")
}
```

find with $text: Searches for documents with "DBMS" or "AI" in the title, returning the matching assignments.

10. **Security:** Implement authentication and authorization for accessing course and student information.

MongoDB provides various features, such as authentication, access control, encryption, to secure your MongoDB deployments.

- **Enable Authentication**:

  Authentication was enabled by creating a root user in the admin database with db.createUser(). This root user has the highest privileges for managing MongoDB.

- **Start MongoDB with Authentication**:

  After creating the root user, MongoDB was restarted, and authentication was enforced. The service was started using net start MongoDB.

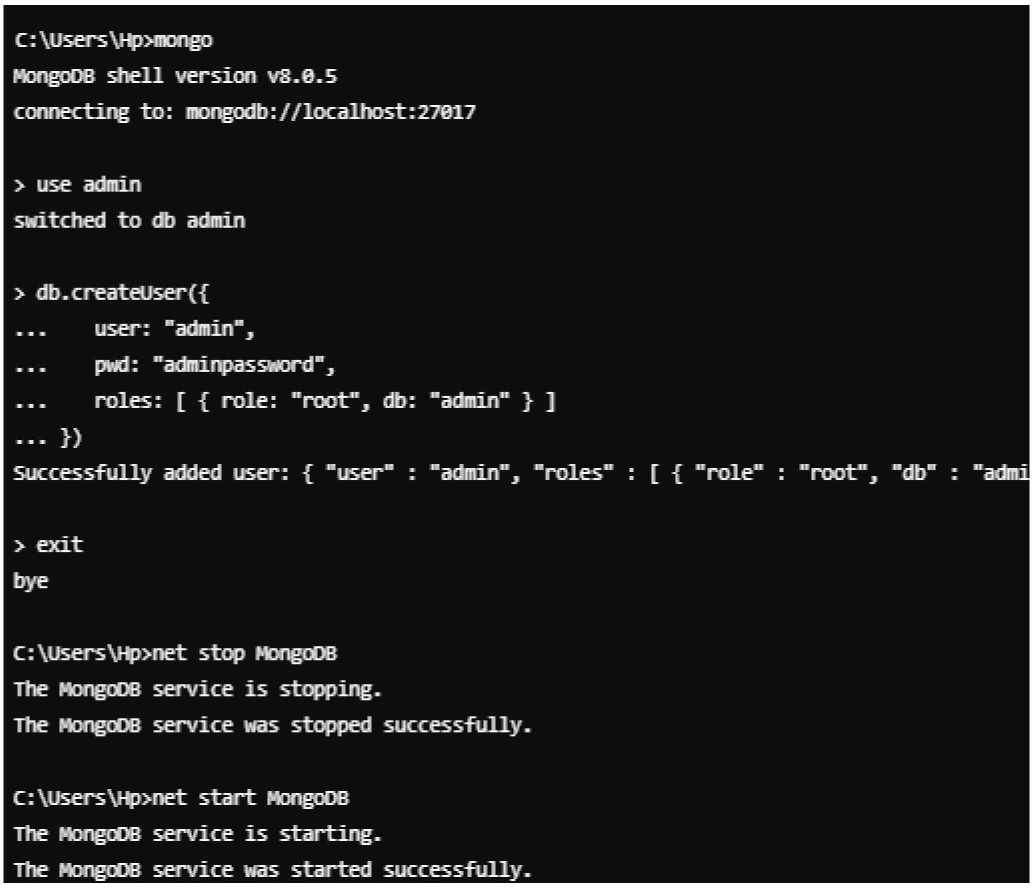
- **Create Additional Users**:

  Two additional users were created within the LMS1 database:

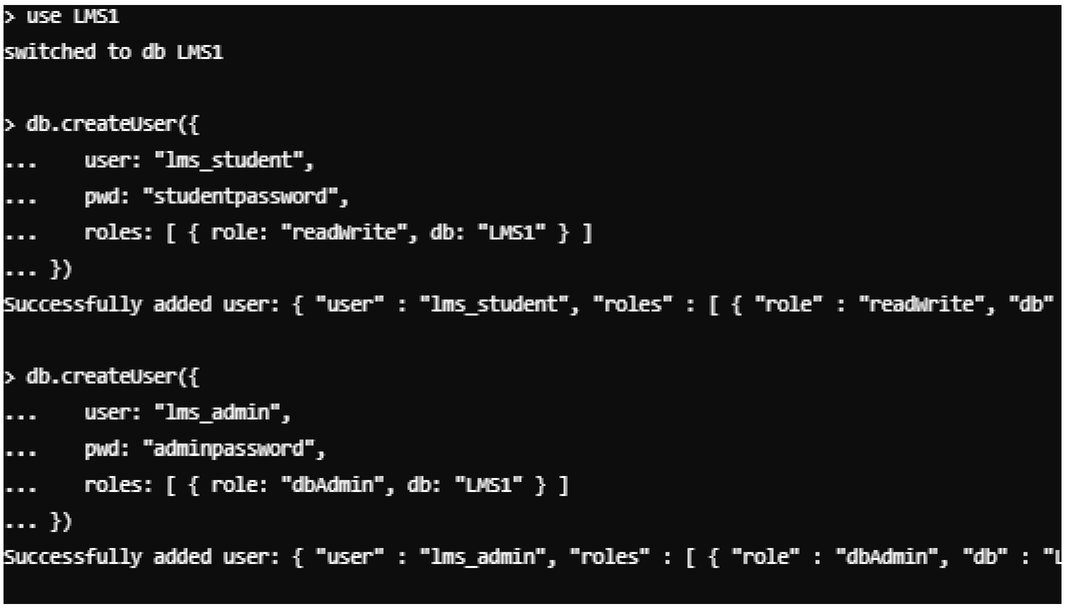
  - lms_student with readWrite role, allowing them to read and write data in the LMS1 database.
  - lms_admin with dbAdmin role, allowing them to manage database operations but not perform administrative tasks on the MongoDB instance itself.

- **Verification**:

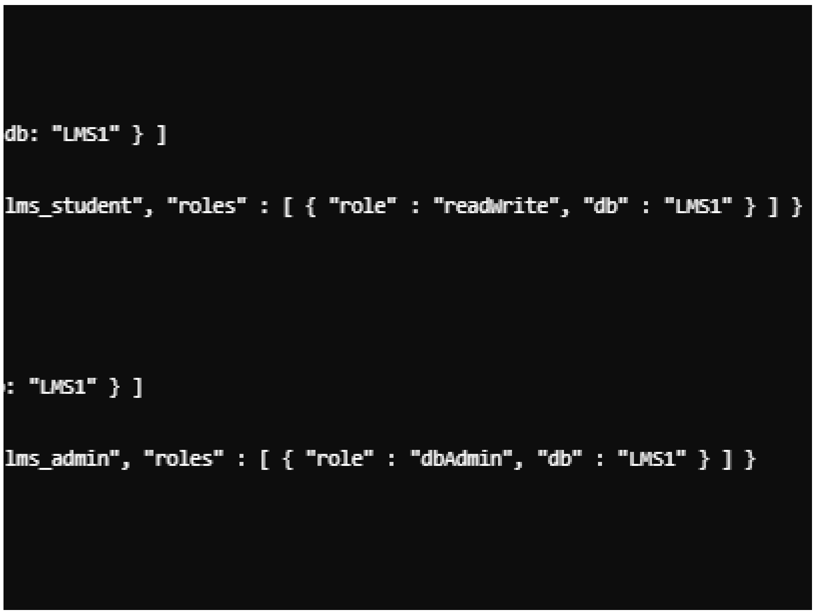
  The db.getUsers() command was run to verify that the users were successfully created with the correct roles.



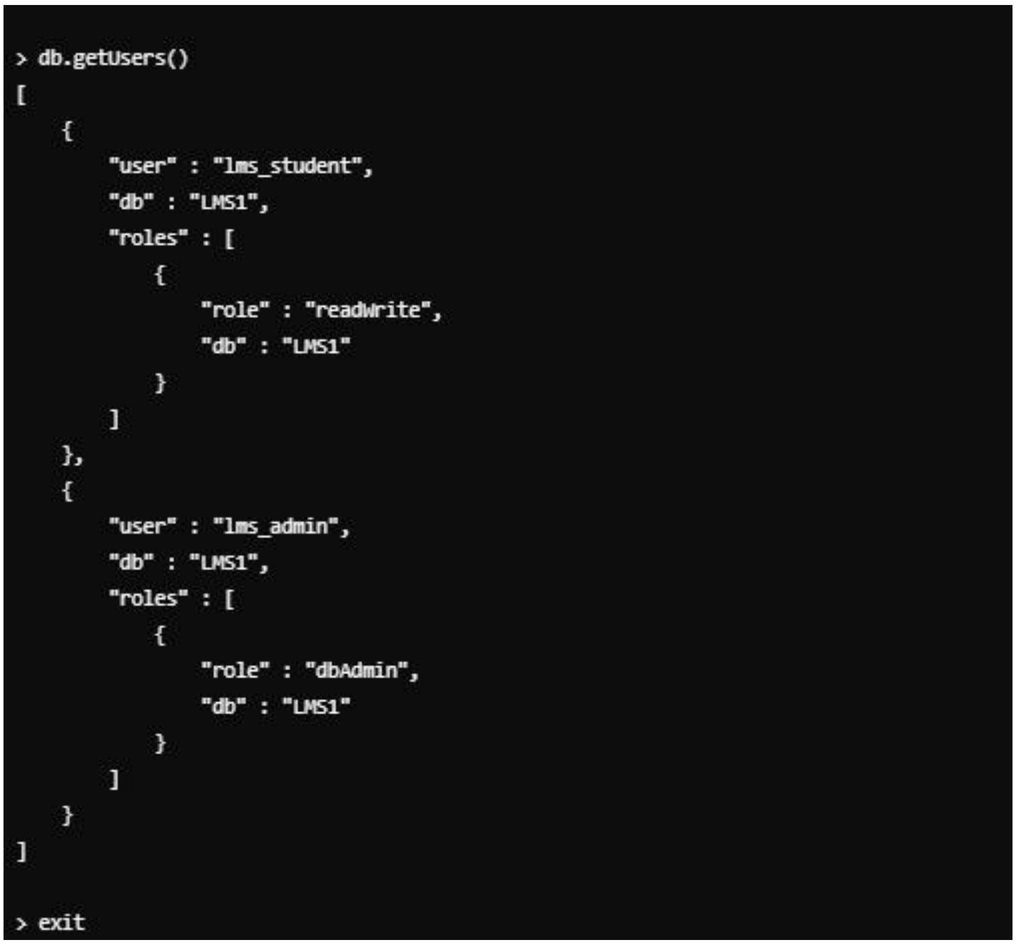
```
C:\Users\Hp>mongo
MongoDB shell version v8.0.5
connecting to: mongodb://localhost:27017

> use admin
switched to db admin

> db.createUser({
...    user: "admin",
...    pwd: "adminpassword",
...    roles: [ { role: "root", db: "admin" } ]
... })
Successfully added user: { "user" : "admin", "roles" : [ { "role" : "root", "db" : "admi

> exit
bye

C:\Users\Hp>net stop MongoDB
The MongoDB service is stopping.
The MongoDB service was stopped successfully.

C:\Users\Hp>net start MongoDB
The MongoDB service is starting.
The MongoDB service was started successfully.
```



```
> use LMS1
switched to db LMS1

> db.createUser({
...    user: "lms_student",
...    pwd: "studentpassword",
...    roles: [ { role: "readWrite", db: "LMS1" } ]
... })
Successfully added user: { "user" : "lms_student", "roles" : [ { "role" : "readWrite", "db"

> db.createUser({
...    user: "lms_admin",
...    pwd: "adminpassword",
...    roles: [ { role: "dbAdmin", db: "LMS1" } ]
... })
Successfully added user: { "user" : "lms_admin", "roles" : [ { "role" : "dbAdmin", "db" : "L
```



```
db: "LMS1" } ]

lms_student", "roles" : [ { "role" : "readWrite", "db" : "LMS1" } ] }




: "LMS1" } ]

lms_admin", "roles" : [ { "role" : "dbAdmin", "db" : "LMS1" } ] }
```



```
> db.getUsers()
[
    {
        "user" : "lms_student",
        "db" : "LMS1",
        "roles" : [
            {
                "role" : "readWrite",
                "db" : "LMS1"
            }
        ]
    },
    {
        "user" : "lms_admin",
        "db" : "LMS1",
        "roles" : [
            {
                "role" : "dbAdmin",
                "db" : "LMS1"
            }
        ]
    }
]

> exit
```