



SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]

[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]

[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]

[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]

VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NAME : _____

ROLL NO : _____

BRANCH : _____

YEAR : _____

SEMESTER : _____



SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service : SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CLASS: III B.E. (CSE)

SEMESTER: VI

Certified that this is the bonafide record of work done by
Mr./Ms. _____ in the
20CS2E32 MEAN STACK DEVELOPMENT of this institution for VI
Semester during the Academic year 2022-2023.

Faculty In-Charge
Mrs.M.Indira Priyadharshini
Assistant Professor/CSE

Professor & Head

Date: _____.

ROLL NUMBER

Submitted for the VI semester B.E. Practical Examination on
_____ 2022-2023.

Internal Examiner

Subject Expert

INDEX

Ex.No	Date	Program Name	Page No	Mark	Signature
1	13/12/2022	Study experiment of MongoDB			
MongoDB Compass					
2.1	20/12/2022	Installation of MongoDB Compass			
2.2	20/12/2022	Creating and Querying data using MongoDB Compass			
MongoDB Shell					
3.1	27/12/2022	Installation of MongoDB Shell			
3.2	27/12/2022	Scripting using MongoDB Shell			
3.3	3/1/2023	Building and utilizing an index in MongoDB			
MongoDB Atlas					
4.1	3/1/2023	Installation of MongoDB Atlas			
4.2	10/1/2023	Running a MongoDB cluster in MongoDB Atlas			
Angular JS					
5	24/01/2023	Communicating Components Through Binding in AngularJS			
6	31/01/2023	Http Service in Angular JS			
Node JS					
7	7/2/2023	Installation Node.js			
8.1	14/02/2023	Module creation in Node.js			
8.2	21/02/2023	Interaction with the file system in Node.js			
Express Js					
9	28/02/2023	Installation of Express Js			
10.1	7/3/2023	ExpressJs middleware functions			
10.2	14/03/2023	ExpressJs Scaffold			
CONTENT BEYOND THE SYLLABUS					
	21/03/2023	Web application Using TypeScript and ASP.Net			

AIM:

To study about the NoSQL database i.e., MONGO DB.

DESCRIPTION:**DATABASE:**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

TYPES OF DATABASES:

There are many different types of databases. The best database for a specific organization depends on how the organization intends to use the data.

- Relational databases.
- Object-oriented databases.
- Distributed databases.
- Data warehouses.
- NoSQL databases.
- Graph databases.

MONGO DB:

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System (RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

RDBMS VS MONGODB:

- RDBMS has a typical schema design that shows number of tables and the relationship between these tables whereas MongoDB is document-oriented. There is no concept of schema or relationship.
- Complex transactions are not supported in MongoDB because complex join operations are not available.
- MongoDB allows a highly flexible and scalable document structure. For example, one data document of a collection in MongoDB can have two fields whereas the other document in the same collection can have four.
- MongoDB is faster as compared to RDBMS due to efficient indexing and storage techniques.
- There are a few terms that are related in both databases. What's called Table in RDBMS is called a Collection in MongoDB. Similarly, a Table is called a Document and A Column is called a Field. MongoDB provides a default '_id' (if not provided explicitly) which is a 12-byte hexadecimal number that assures the uniqueness of every document. It is similar to the Primary key in RDBMS.

FEATURES OF MONGODB:

- **Document Oriented:** MongoDB stores the main subject in the minimal number of documents and not by breaking it up into multiple relational structures like RDBMS. For example, it stores all the information of a computer in a single document called Computer and not in distinct relational structures like CPU, RAM, Hard disk, etc.
- **Indexing:** Without indexing, a database would have to scan every document of a collection to select those that match the query which would be inefficient. So, for efficient searching Indexing is a must and MongoDB uses it to process huge volumes of data in very less time.
- **Scalability:** MongoDB scales horizontally using sharding (partitioning data across various servers). Data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that reside across many physical servers. Also, new machines can be added to a running database.
- **Replication and High Availability:** MongoDB increases the data availability with multiple copies of data on different servers. By providing redundancy, it protects the database from hardware failures. If one server goes down, the data can be retrieved easily from other active servers which also had the data stored on them.
- **Aggregation:** Aggregation operations process data records and return the computed results. It is similar to the GROUPBY clause in SQL. A few aggregation expressions are sum, avg, min, max, etc.

WHERE DO WE USE MONGODB?

MongoDB is preferred over RDBMS in the following scenarios:

- **Big Data:** If you have huge amount of data to be stored in tables, think of MongoDB before RDBMS databases. MongoDB has built-in solution for partitioning and sharding your database.
- **Unstable Schema:** Adding a new column in RDBMS is hard whereas MongoDB is schema-less. Adding a new field does not effect old documents and will be very easy.
- **Distributed data:** Since multiple copies of data are stored across different servers, recovery of data is instant and safe even if there is a hardware failure.

LANGUAGE SUPPORT BY MONGODB:

MongoDB currently provides official driver support for all popular programming languages like C, C++, Rust, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go, and Erlang.

GENERAL PACKAGES:

- Core Processes
- Windows Services
- Binary Import and Export Tools
- Data Import and Export Tools
- Diagnostic Tools
- GridFS
- MongoDB Compass

RESULT:

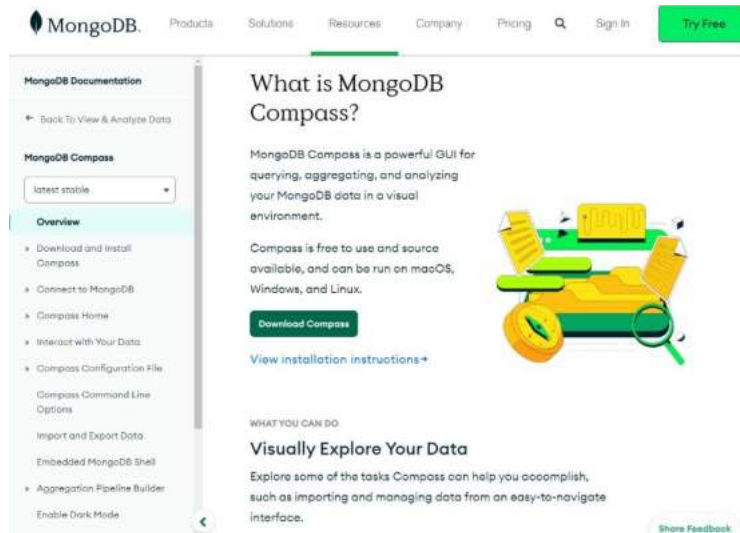
Thus, the study on MONGO DB is made successfully.

AIM:

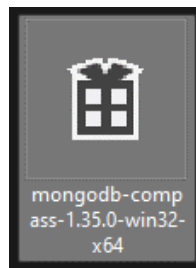
To install MongoDB Compass in our machine.

PROCEDURE:

Step 1: Firstly, go to MongoDB website and download MongoDB Community Server.



Step 2: Double click the installer icon.

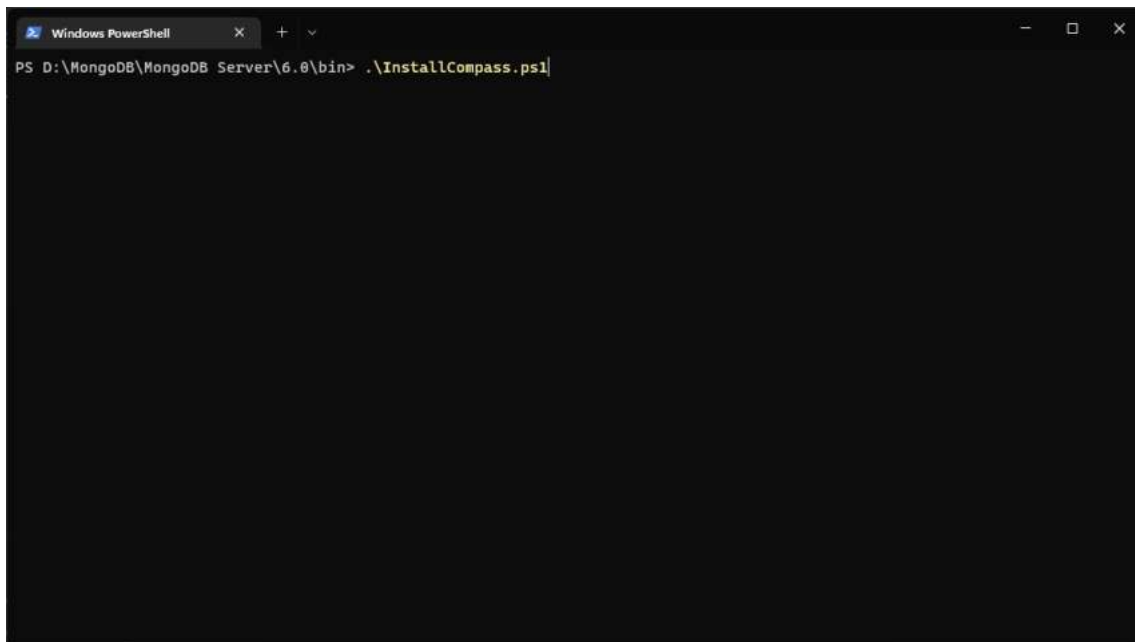


Step 3: Follow the installation prompts and customize the installation according to your need.



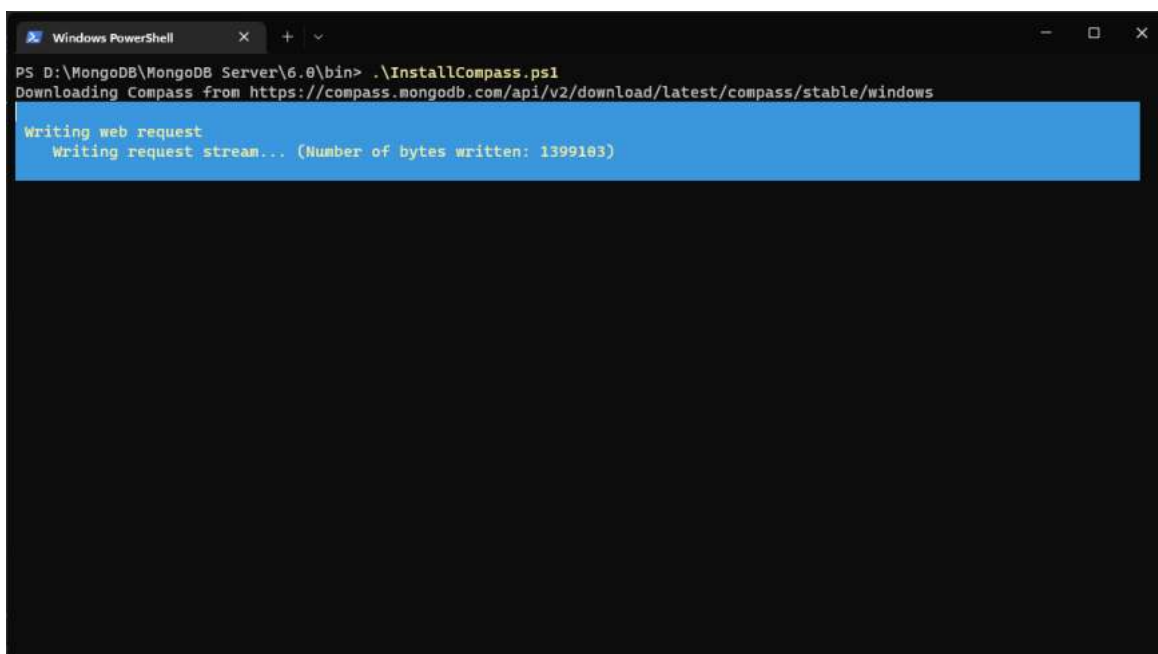
Step 4: After selecting the path for installation, the installation process for the MongoDB server will start and the required files will be installed. After that open the location of installed MongoDB server to the **bin** folder and open the same in Windows PowerShell.

Now in CMD, type **.\InstallCompass.ps1**.



```
Windows PowerShell
PS D:\MongoDB\MongoDB Server\6.0\bin> .\InstallCompass.ps1
```

Now this .ps1 file will install the required files for using MongoDB Compass.

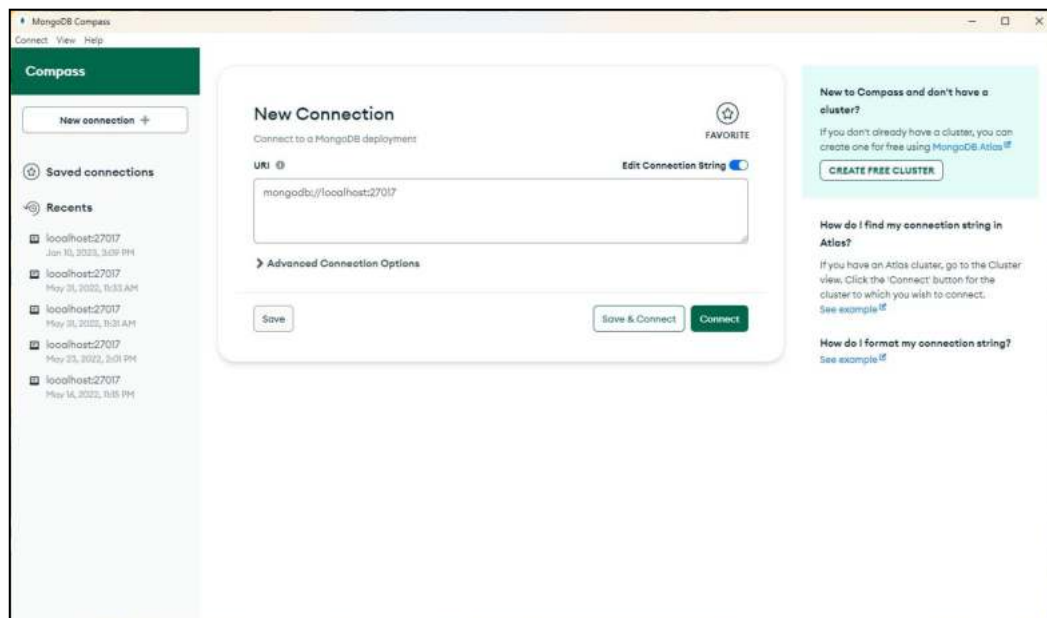


```
Windows PowerShell
PS D:\MongoDB\MongoDB Server\6.0\bin> .\InstallCompass.ps1
Downloading Compass from https://compass.mongodb.com/api/v2/download/latest/compass/stable/windows
Writing web request
Writing request stream... (Number of bytes written: 1399103)
```

Now, the Compass will be installed.


```
Windows PowerShell
PS D:\MongoDB\MongoDB Server\6.0\bin> .\InstallCompass.ps1
Downloading Compass from https://compass.mongodb.com/api/v2/download/latest/compass/stable/windows
Installing Compass
Successfully installed Compass
PS D:\MongoDB\MongoDB Server\6.0\bin> |
```

Step 5: After the above step, all installation process is done. now you can work with your databases.



After installation, you will find the MongoDB compass icon on your desktop.

RESULT:

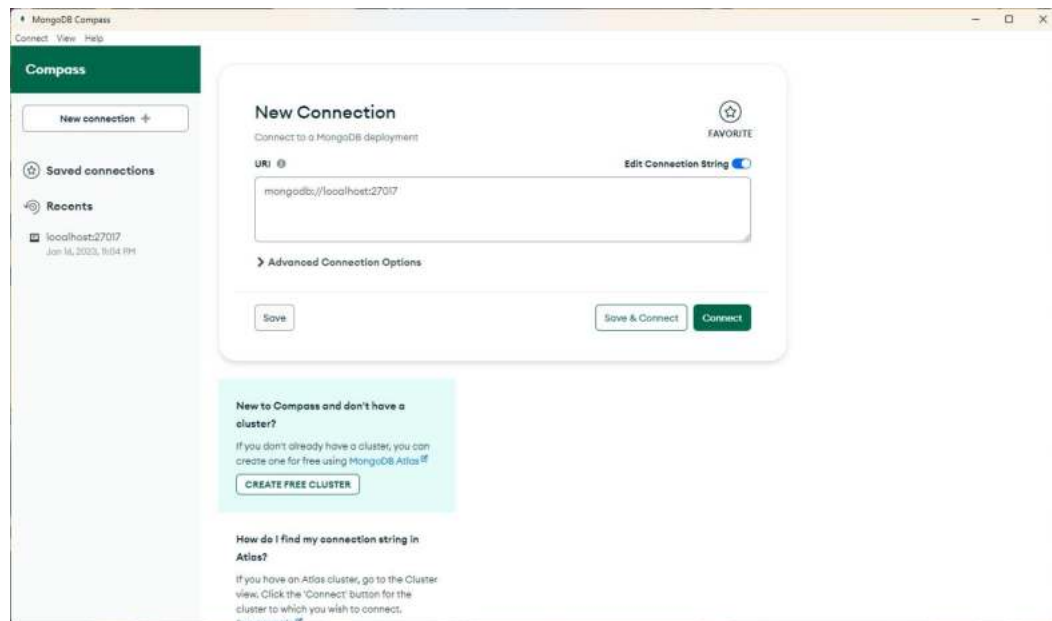
Thus, MongoDB Compass has been successfully installed in our machine.

AIM:

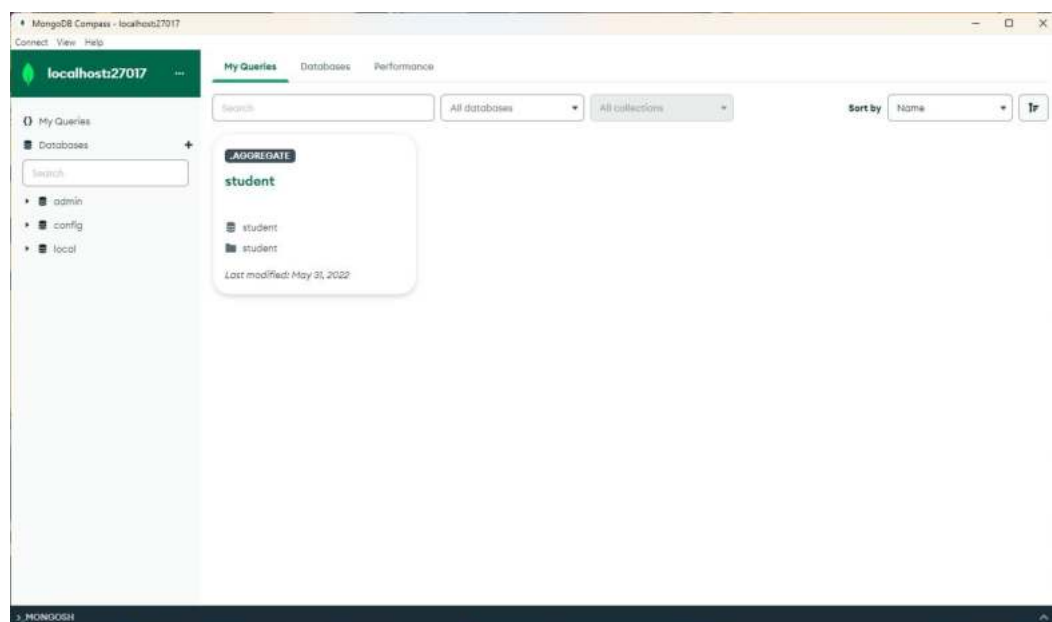
To create a database and to query data using MongoDB Compass.

CREATING AND QUERYING DATA IN DATABASE:

Step 1: Create a new connection with localhost or connect to a cluster if any from the below page



Step 2: After a connection is created the databases from the cluster will be displayed



Step 3: To create a database select the “+” icon on the left side of the navigation menu, a pop up menu will be displayed

Create Database ×

Database Name

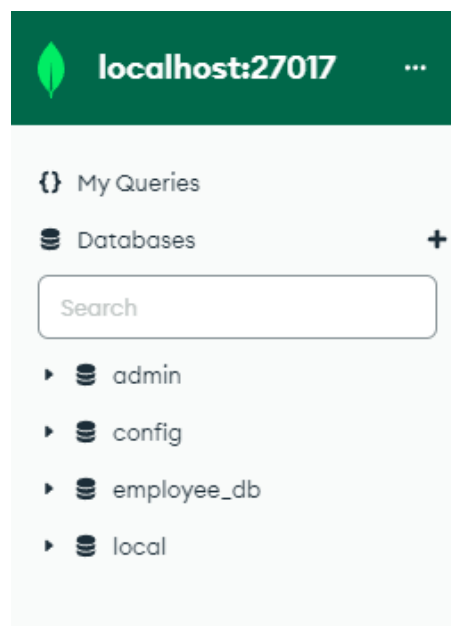
Collection Name

➤ **Advanced Collection Options** (e.g. Time-Series, Capped, Clustered collections)

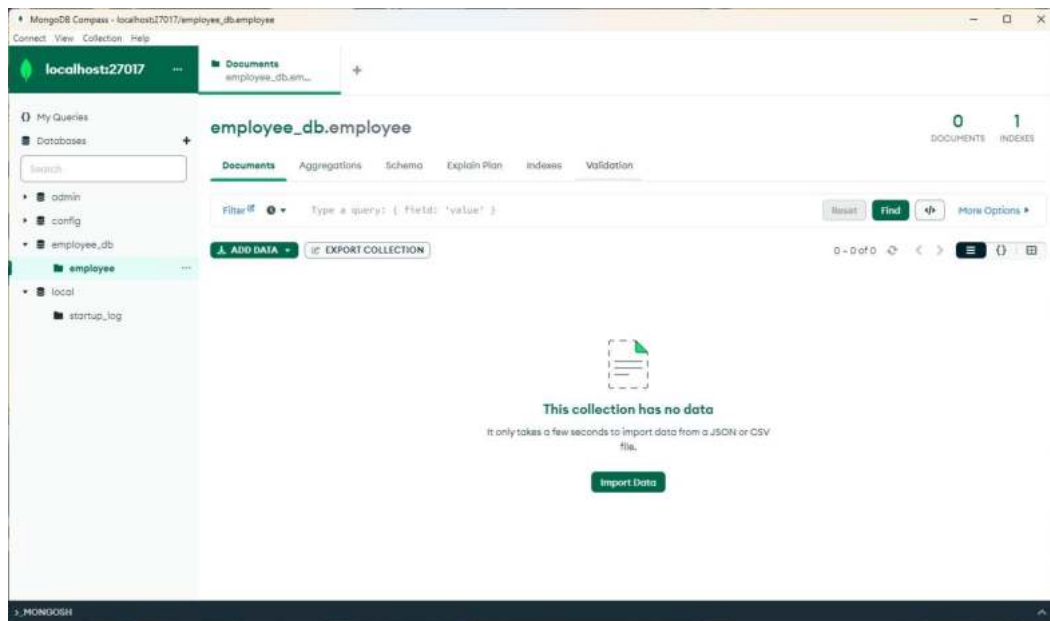
Cancel

Create Database

Step 4: Now a database will be created which will be displayed on the top left corner.



Step 5: To insert new documents to the created database navigate to the database created and select add data and select insert document.



Enter the details of the fields and values and then select insert to insert the document into the database.

Insert Document

To Collection employee_db.employee

VIEW

1

`_id: ObjectId('63e12007d00be1f11f34f0fb')`

ObjectId

2

`employee_id: 197`

Int32

3

`first_name: "Alan"`

String

4

`last_name: "Walker"`

String

5

`email: "AWALKER"`

String

6

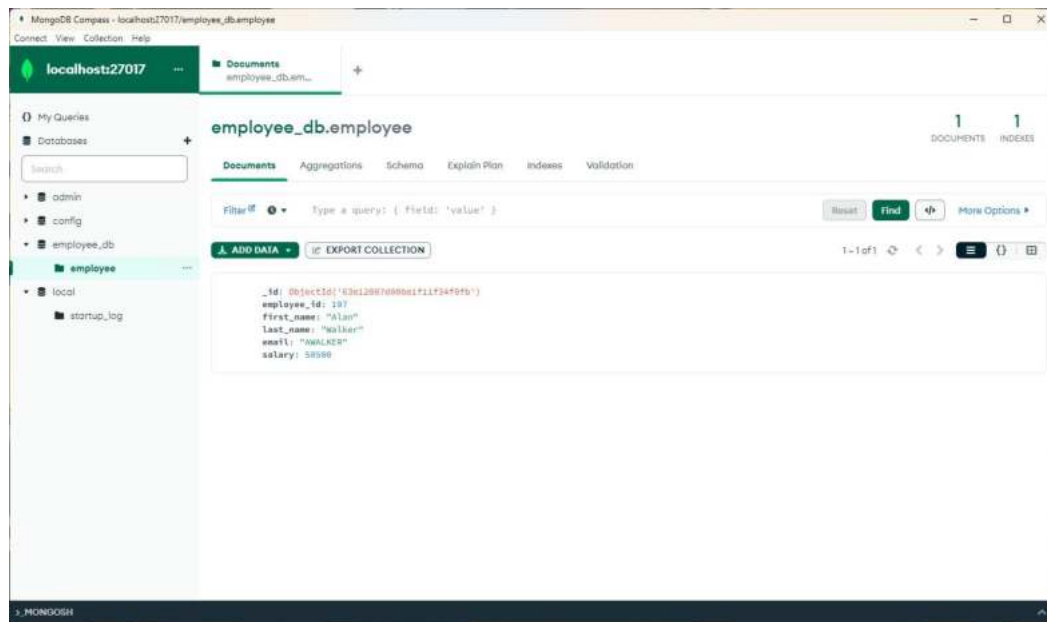
`salary: 50500`

Int32

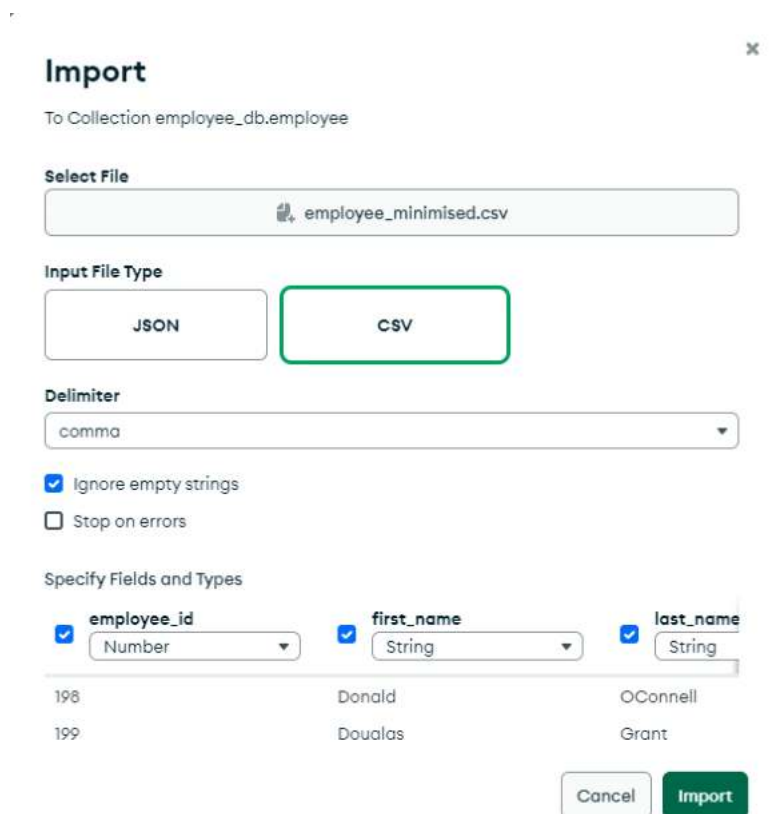
Cancel

Insert

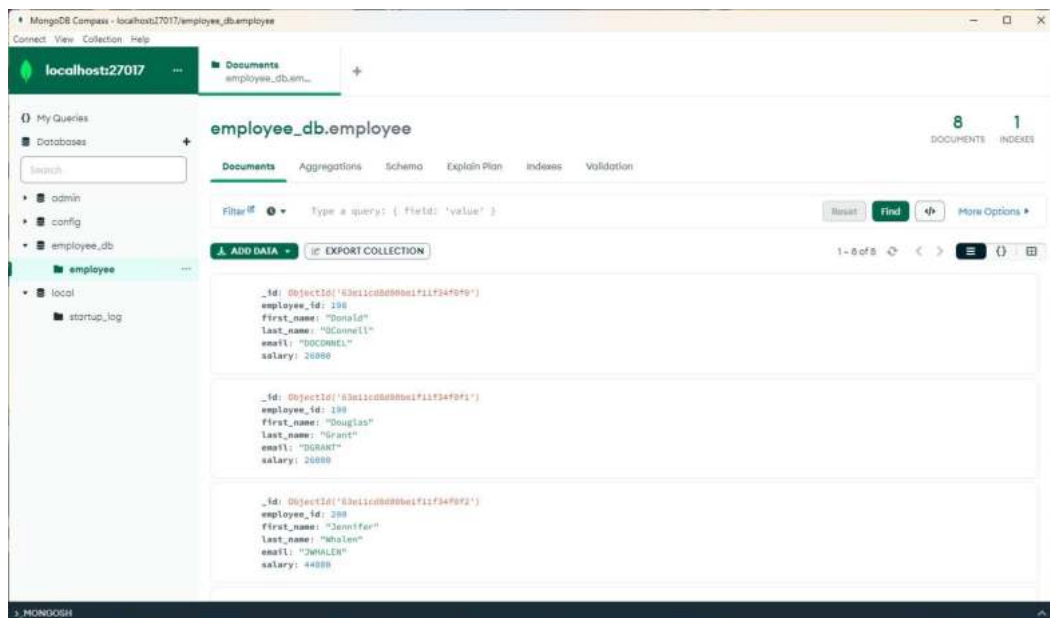
Now the document will be inserted to the database.



Step 6: Documents can also be inserted with CSV and JSON files. It can be done by selecting add data and then selecting insert file.



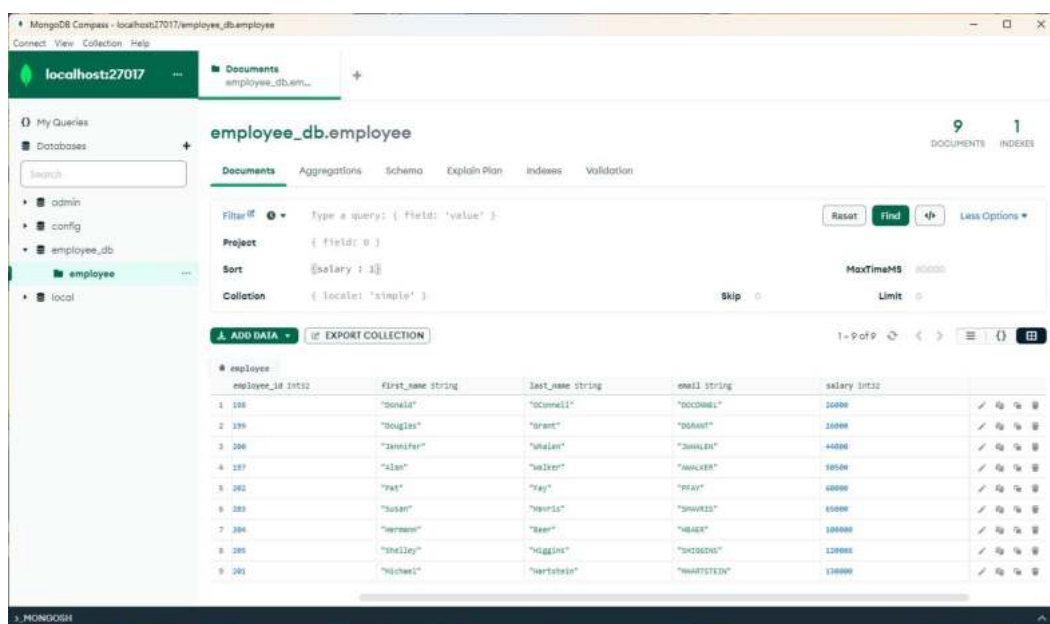
Now, select the type of fields to be imported i.e., for name “String” for salary “Number” etc. and then import the data.



Now the contents of the file will be imported and will be displayed on the documents field.

Step 7: There are many queries which can be used to filter out the documents of a database.

For example, the documents can be sorted with the query “{field: -1}” or “{field: 1}” -1 will sort the documents in descending order and 1 will sort in ascending order.



Here the documents are sorted in ascending order of the field salary.

RESULT:

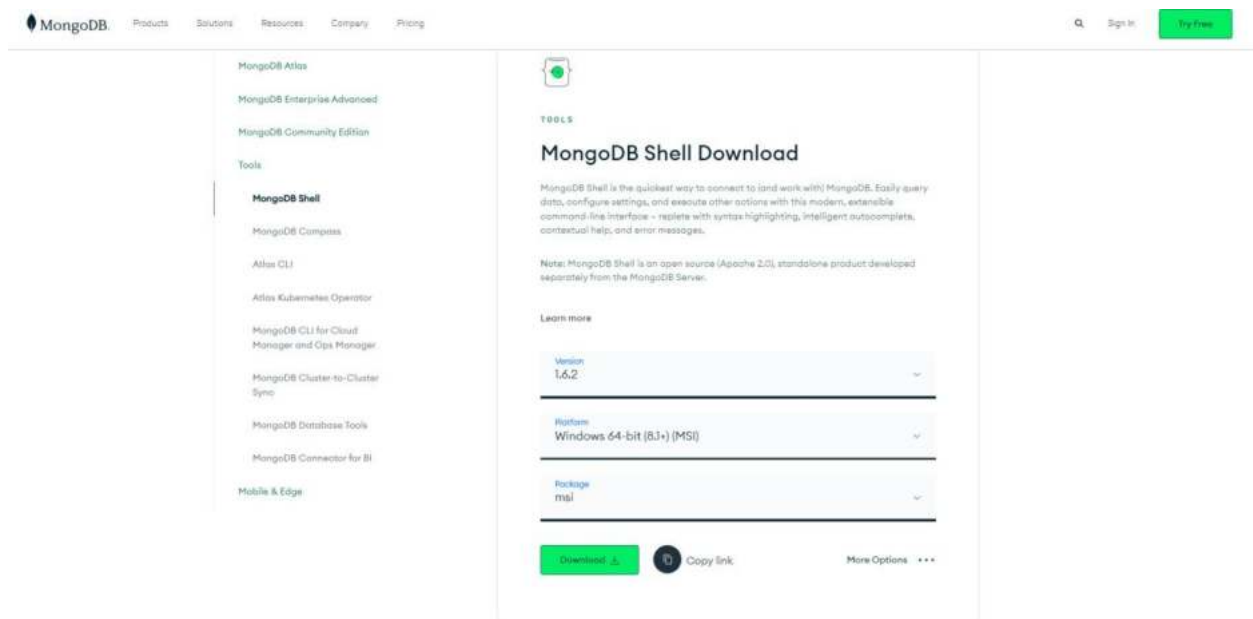
Thus, a database was created and the data for the database are queried using MongoDB compass.

AIM:

To install MongoDB shell in the local system.

PROCEDURE:

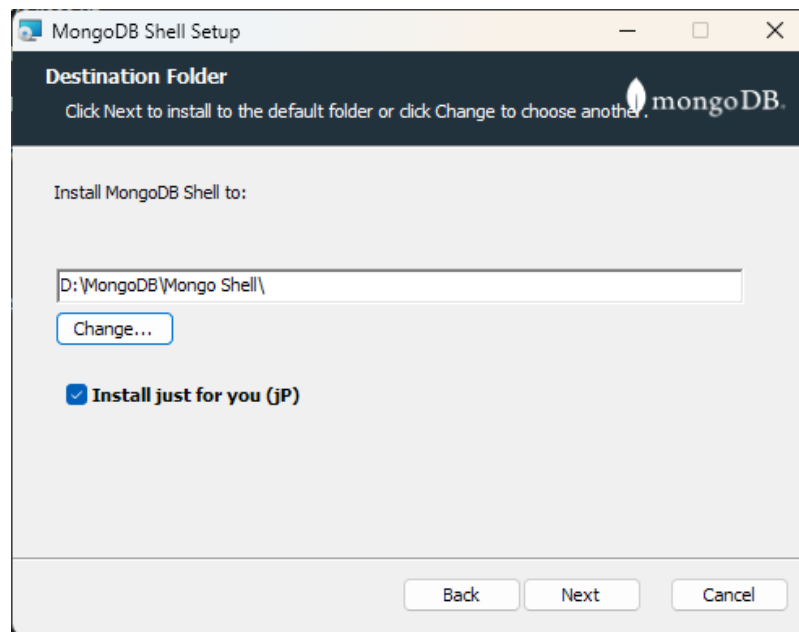
Step 1: To install MongoDB shell, go to the site <https://www.mongodb.com/try/download/shell> and select the file type to download it, it can be either a ZIP file or a MSI file. Here we'll go with an MSI file. Now select on download to start the download.



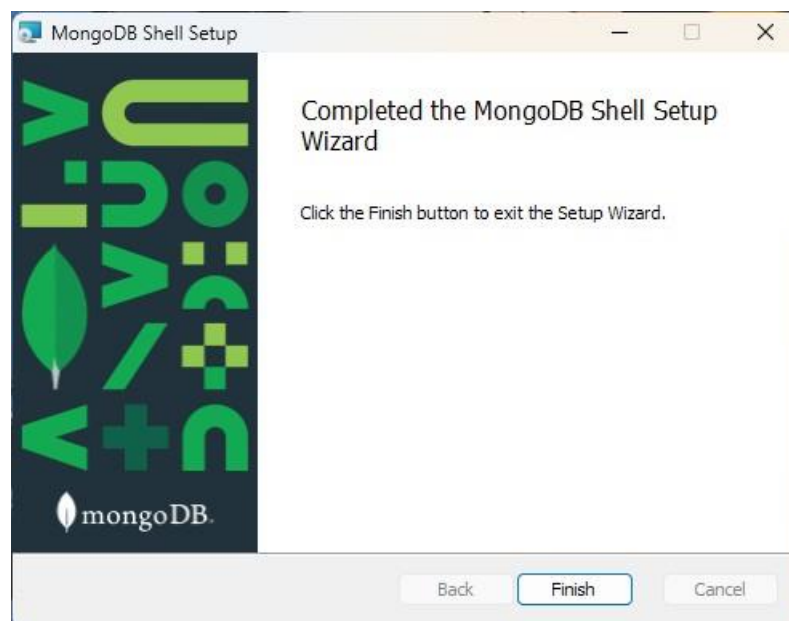
Step 2: After downloading the file, select the file to install, an installation prompt will be displayed.



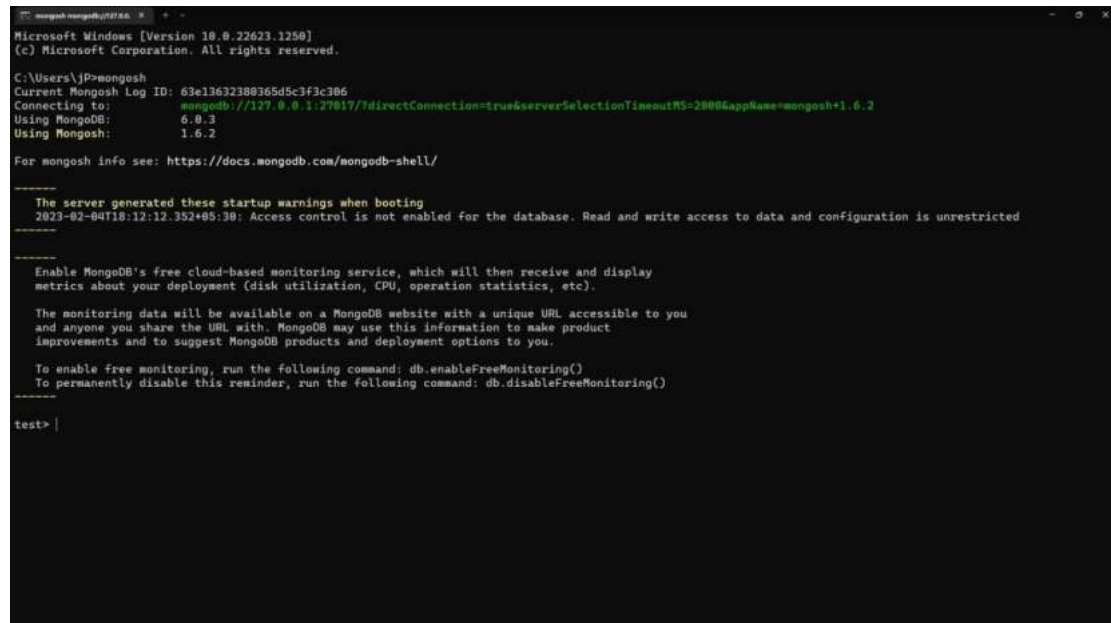
Step 3: Now choose the destination folder to where the shell should be installed and start the installation by selecting next and install button.



Step 4: Now after installation is complete, click on finish.



To check whether the installation is legit, go to the command prompt and type the command “mongosh” and run the command.



```
Microsoft Windows [Version 10.0.22623.1250]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JP>mongosh
Current Mongosh Log ID: 63c13632388765d5c3f3c386
Connecting to:  mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.6.2
Using MongoDB:  6.0.3
Using Mongosh:   1.6.2

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

-----
The server generated these startup warnings when booting
2023-02-04T18:12:12.352+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

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()
-----

test> |
```

RESULT:

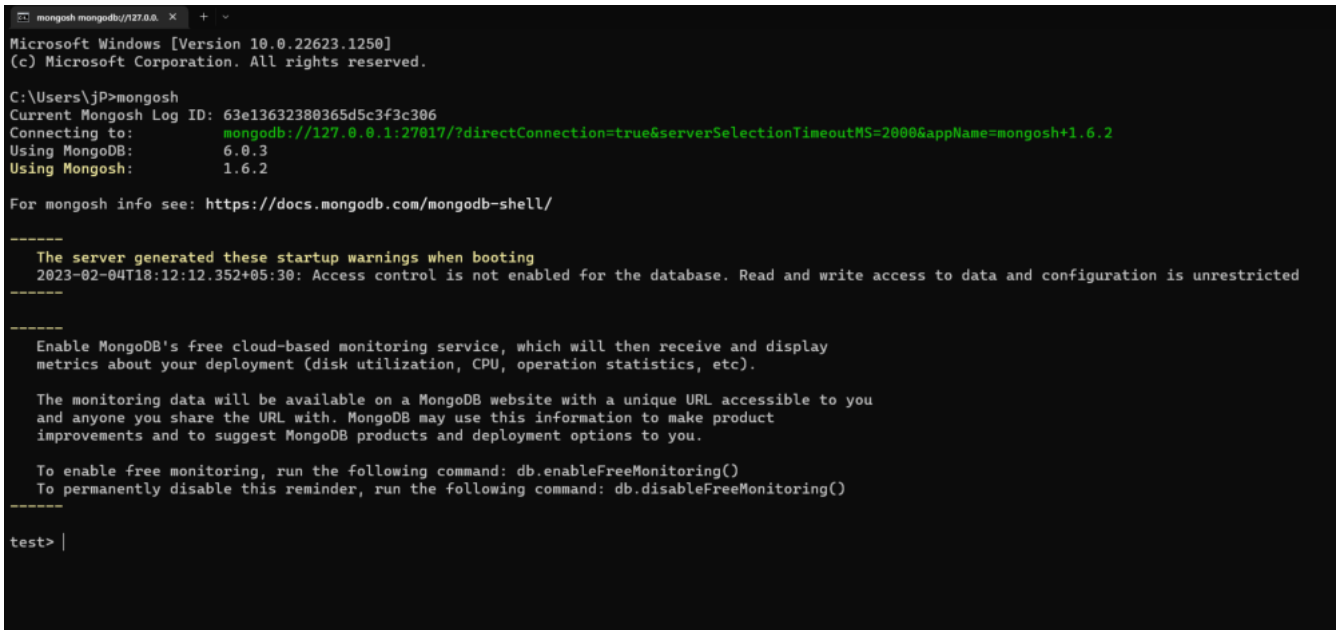
Thus, the installation of MongoDB shell is successfully completed.

AIM:

To connect MongoDB using Mongosh in CMD and explore its terminal.

PROCEDURE:

Step 1: Use 'mongosh' command to run the shell



```
Microsoft Windows [Version 10.0.22623.1250]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jp>mongosh
Current Mongosh Log ID: 63e13632380365d5c3f3c306
Connecting to:  mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.6.2
Using MongoDB:  6.0.3
Using Mongosh:  1.6.2

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

-----
The server generated these startup warnings when booting
2023-02-04T18:12:12.352+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

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()
-----

test> |
```

Step 2: Now you are in the mongo shell. You can run the mongo and mongod without command prompt.

Step 3: MongoDB server runs on default port, 27027. If your MongoDB server runs on a different port you have to explicitly specify it in the command.

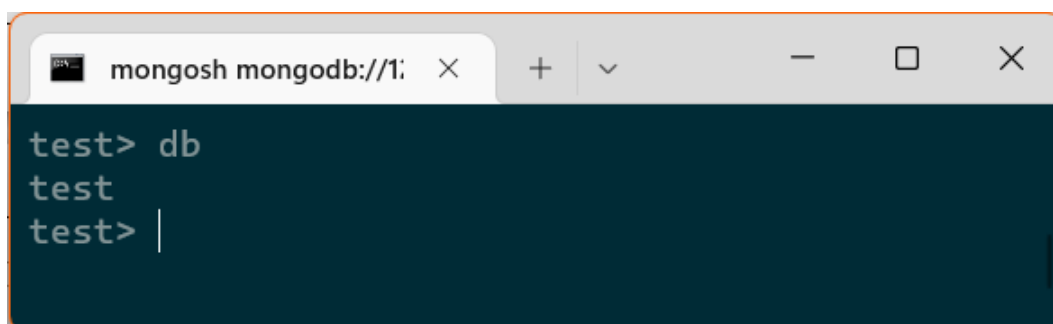
`mongosh --port 28010`

Step 4: If you want to connect to a remote server, use the '-host' option with the mongo command.

`mongosh --host mongodb0.example.com --port 28010`

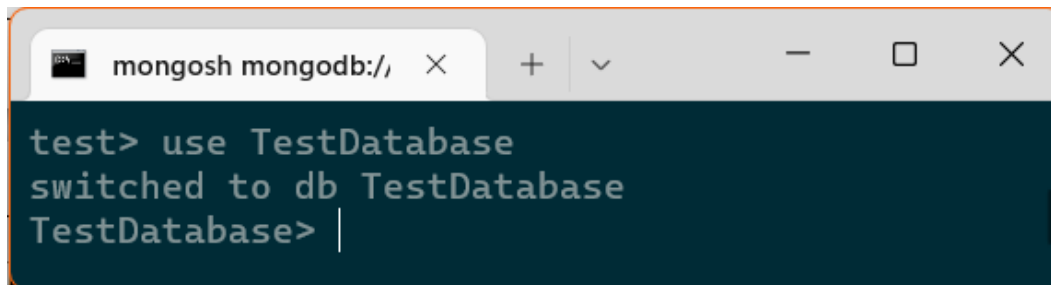
COMMANDS IN MONGOSH

1) Db: Use db command to see the database you are current working with



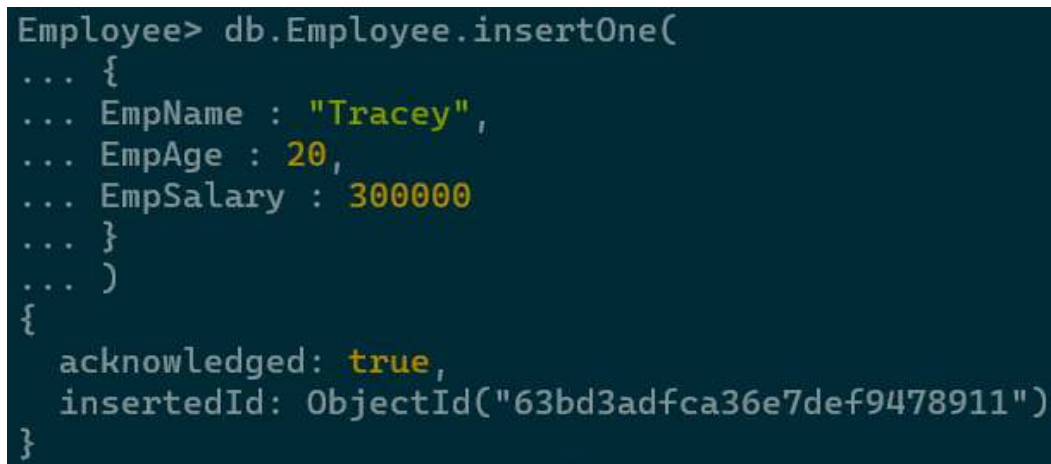
```
test> db
test
test> |
```

- 2) Use : The use command is used to switch to a different database. If you don't have a database, it creates a database.



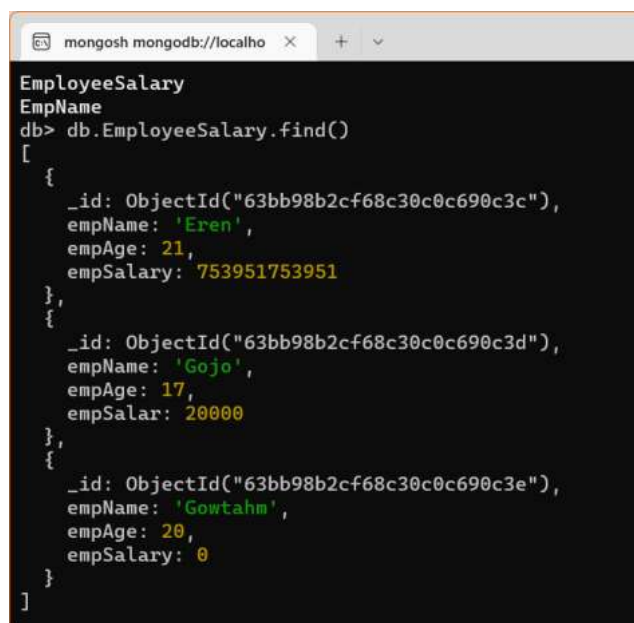
```
mongosh mongodb://
test> use TestDatabase
switched to db TestDatabase
TestDatabase> |
```

- 3) insertOne: It is used to create collections and insert data.



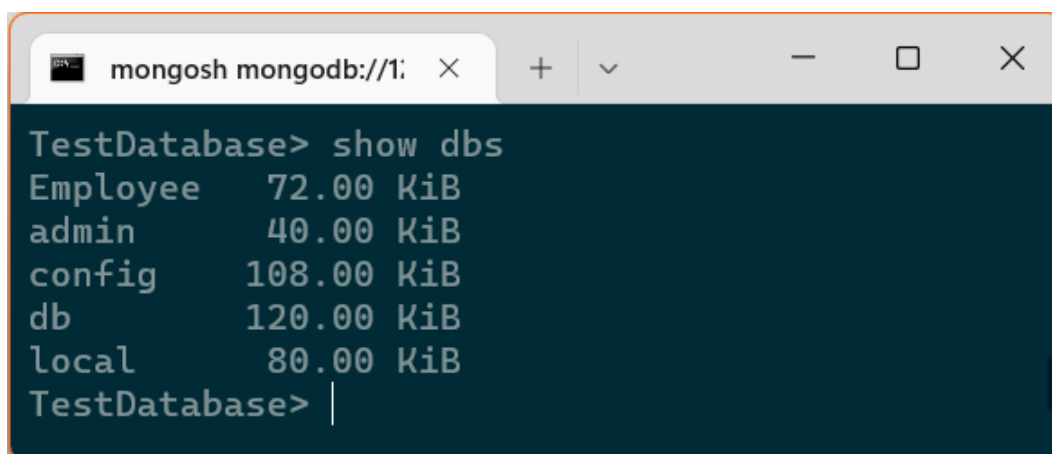
```
Employee> db.Employee.insertOne(
... {
...   EmpName : "Tracey",
...   EmpAge : 20,
...   EmpSalary : 300000
... }
... )
{
  acknowledged: true,
  insertedId: ObjectId("63bd3adfca36e7def9478911")
}
```

- 4) insertMany: It is used to insert more than one document to a collection



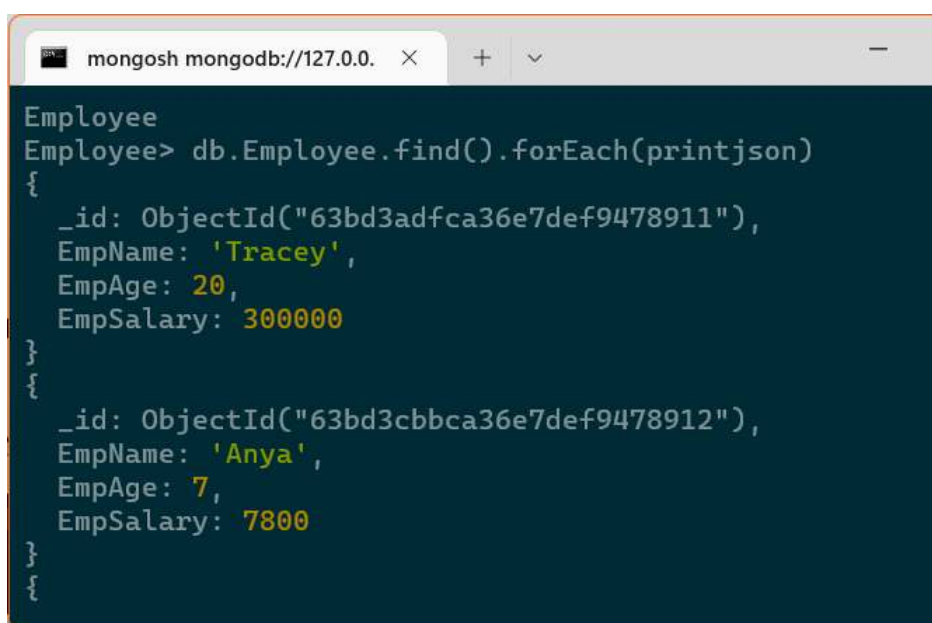
```
EmployeeSalary
EmpName
db> db.EmployeeSalary.insertMany([
  {
    empName: 'Eren',
    empAge: 21,
    empSalary: 753951753951
  },
  {
    empName: 'Gojo',
    empAge: 17,
    empSalary: 20000
  },
  {
    empName: 'Gowtahn',
    empAge: 20,
    empSalary: 0
  }
])
db> db.EmployeeSalary.find()
[
  {
    _id: ObjectId("63bb98b2cf68c30c0c690c3c"),
    empName: 'Eren',
    empAge: 21,
    empSalary: 753951753951
  },
  {
    _id: ObjectId("63bb98b2cf68c30c0c690c3d"),
    empName: 'Gojo',
    empAge: 17,
    empSalary: 20000
  },
  {
    _id: ObjectId("63bb98b2cf68c30c0c690c3e"),
    empName: 'Gowtahn',
    empAge: 20,
    empSalary: 0
  }
]
```

5) show dbs: It display the all databases in the connected server.



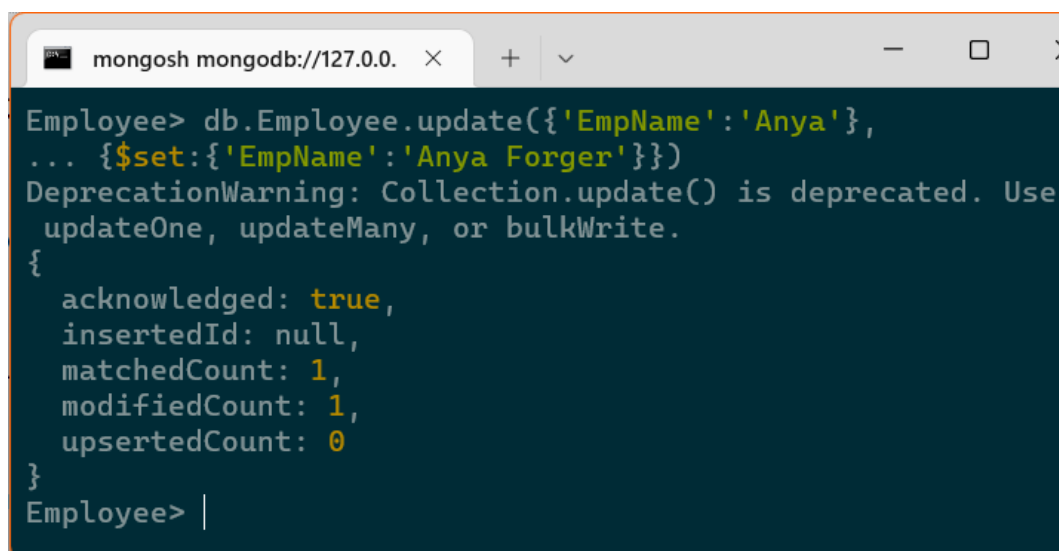
```
mongosh mongodb://1: x + v - □ ×
TestDatabase> show dbs
Employee    72.00 KiB
admin       40.00 KiB
config      108.00 KiB
db          120.00 KiB
local       80.00 KiB
TestDatabase> |
```

6) Find: It is used to fetch data in a collection. The forEach(printjson) method prints them with JSON formatting



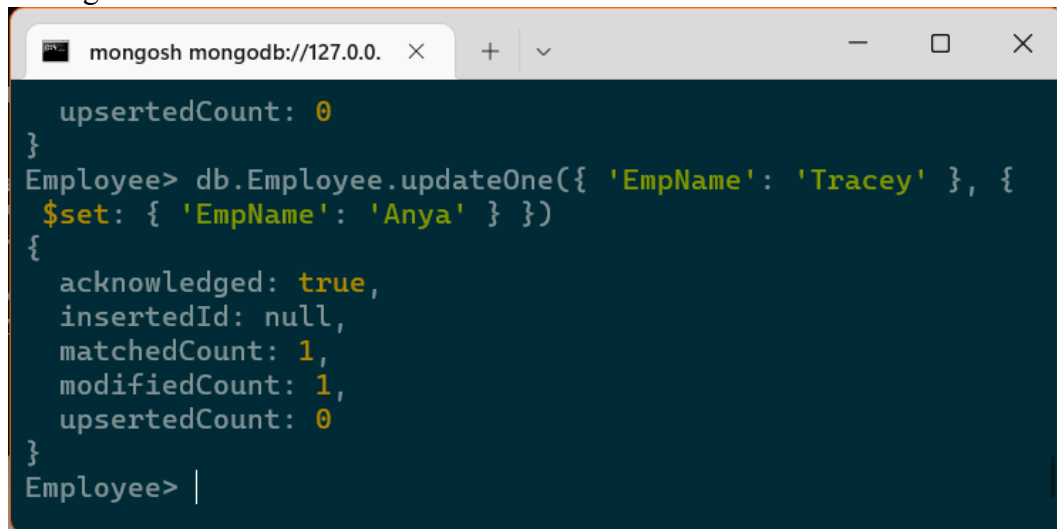
```
mongosh mongodb://127.0.0. x + v -
Employee
Employee> db.Employee.find().forEach(printjson)
{
  _id: ObjectId("63bd3adfca36e7def9478911"),
  EmpName: 'Tracey',
  EmpAge: 20,
  EmpSalary: 300000
}
{
  _id: ObjectId("63bd3cbbca36e7def9478912"),
  EmpName: 'Anya',
  EmpAge: 7,
  EmpSalary: 7800
}
{
}
```

7) Update: The update() method updates the values in the existing document.



```
mongosh mongodb://127.0.0. x + v -
Employee> db.Employee.update({'EmpName': 'Anya'},
... {$set: {'EmpName': 'Anya Forger'}})
DeprecationWarning: Collection.update() is deprecated. Use
updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Employee> |
```

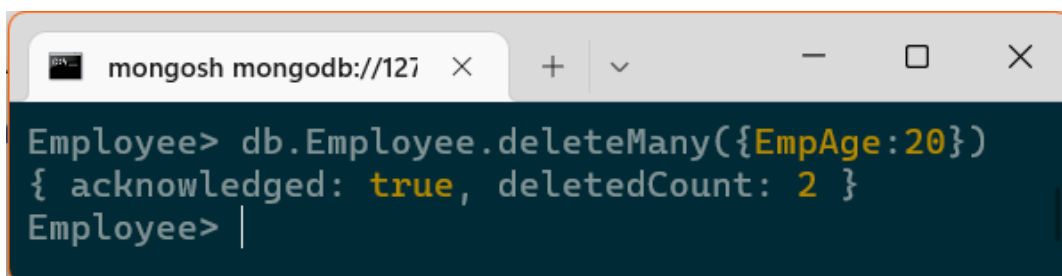
- 8) `updateOne`: It updates the first matching document in the collection. Use `{multi: true}` to update all the matching document in the collection

A screenshot of a MongoDB shell window titled 'mongosh mongodb://127.0.0.0'. The terminal shows the following commands and output:

```
upsertedCount: 0
}
Employee> db.Employee.updateOne({ 'EmpName': 'Tracey' }, {
  $set: { 'EmpName': 'Anya' } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Employee> |
```

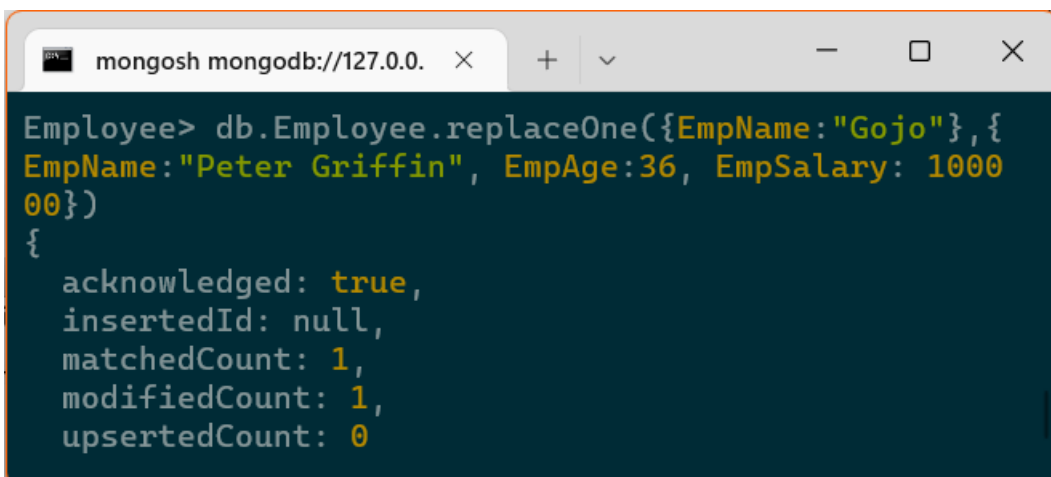
- 9) `deleteOne`: This command deletes the first matched document in the collection

- 10) `deleteMany`: This command deletes all the matching documents in the collection

A screenshot of a MongoDB shell window titled 'mongosh mongodb://127.0.0.0'. The terminal shows the following commands and output:

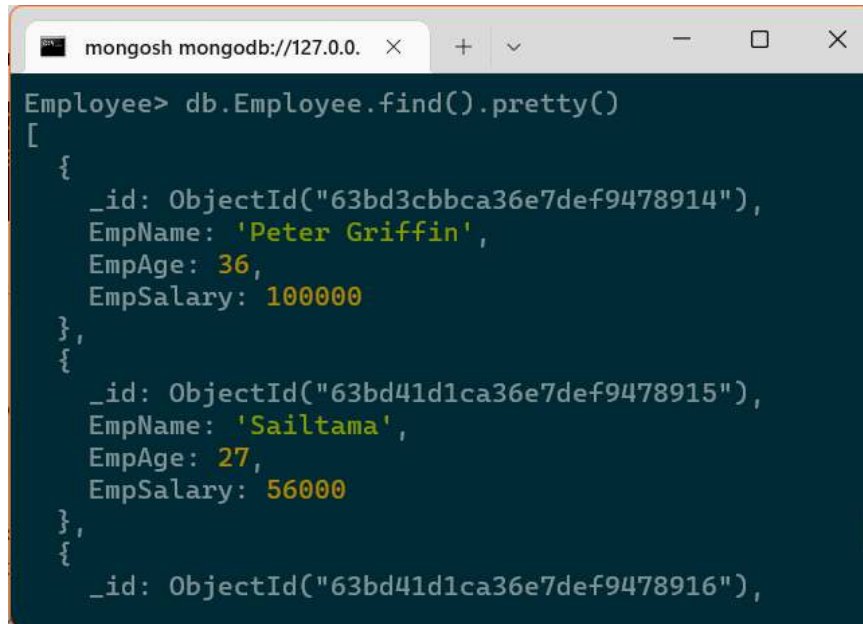
```
Employee> db.Employee.deleteMany({EmpAge:20})
{ acknowledged: true, deletedCount: 2 }
Employee> |
```

- 11) `replaceOne`: It is a mongo shell command, which only replaces one document at a time. By default it replaces the first document. Desired documents can be deleted using filters.

A screenshot of a MongoDB shell window titled 'mongosh mongodb://127.0.0.0'. The terminal shows the following commands and output:

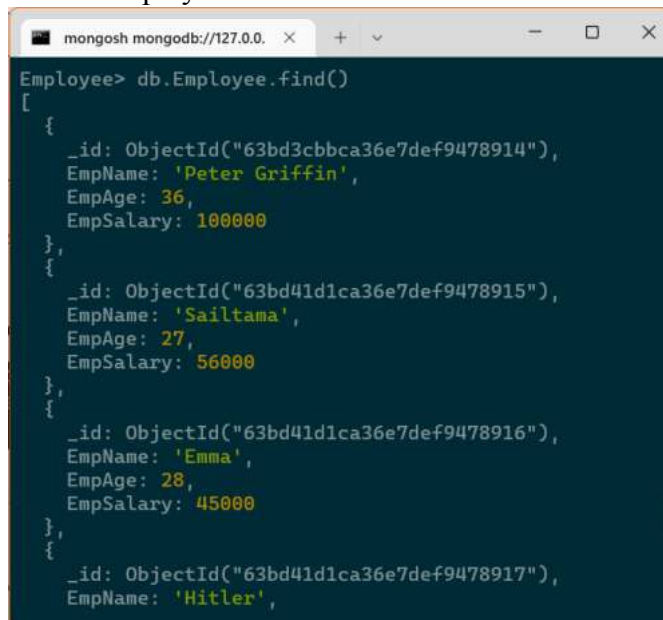
```
Employee> db.Employee.replaceOne({EmpName:"Gojo"},{
  EmpName:"Peter Griffin", EmpAge:36, EmpSalary: 1000
  00})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

12) pretty: This method displays the result in formatted way.



```
Employee> db.Employee.find().pretty()
[
  {
    _id: ObjectId("63bd3cbbca36e7def9478914"),
    EmpName: 'Peter Griffin',
    EmpAge: 36,
    EmpSalary: 100000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478915"),
    EmpName: 'Sailtama',
    EmpAge: 27,
    EmpSalary: 56000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478916"),
```

13) Find: This method will display all the documents in a collection.

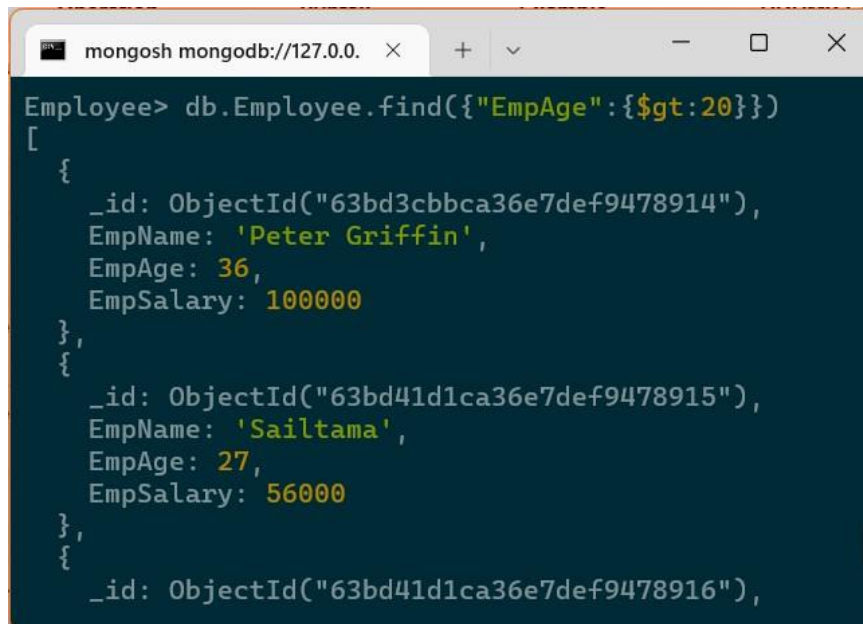


```
Employee> db.Employee.find()
[
  {
    _id: ObjectId("63bd3cbbca36e7def9478914"),
    EmpName: 'Peter Griffin',
    EmpAge: 36,
    EmpSalary: 100000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478915"),
    EmpName: 'Sailtama',
    EmpAge: 27,
    EmpSalary: 56000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478916"),
    EmpName: 'Emma',
    EmpAge: 28,
    EmpSalary: 45000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478917"),
    EmpName: 'Hitler',
```

14) To query the document, use the following operations with find command to make them act as filters

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:{\$eq:<value>}}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{\$lt:<value>}}	db.mycol.find({"likes":{\$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{\$lte:<value>}}	db.mycol.find({"likes":{\$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{\$gt:<value>}}	db.mycol.find({"likes":{\$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{\$gte:<value>}}	db.mycol.find({"likes":{\$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{\$ne:<value>}}	db.mycol.find({"likes":{\$ne:50}}).pretty()	where likes != 50
Values in an array	{<key>:{\$in:[<value1>,<value2>,...,<valueN>]}}	db.mycol.find({"name":{\$in:["Raj", "Ram", "Raghu"]}}).pretty()	Where name matches any of the value in :["Raj", "Ram", "Raghu"]
Values not in an array	{<key>:{\$nin:<value>}}	db.mycol.find({"name":{\$nin:["Ramu", "Raghav"]}}).pretty()	Where name values is not in the array :["Ramu", "Raghav"] or, doesn't exist at all

This query finds the document/documents whose age is greater 20.



```

Employee> db.Employee.find({"EmpAge":{$gt:20}})
[
  {
    _id: ObjectId("63bd3cbbca36e7def9478914"),
    EmpName: 'Peter Griffin',
    EmpAge: 36,
    EmpSalary: 100000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478915"),
    EmpName: 'Sailtama',
    EmpAge: 27,
    EmpSalary: 56000
  },
  {
    _id: ObjectId("63bd41d1ca36e7def9478916"),

```

RESULT:

Thus, scripting using mongo shell is done successfully and the output is verified.

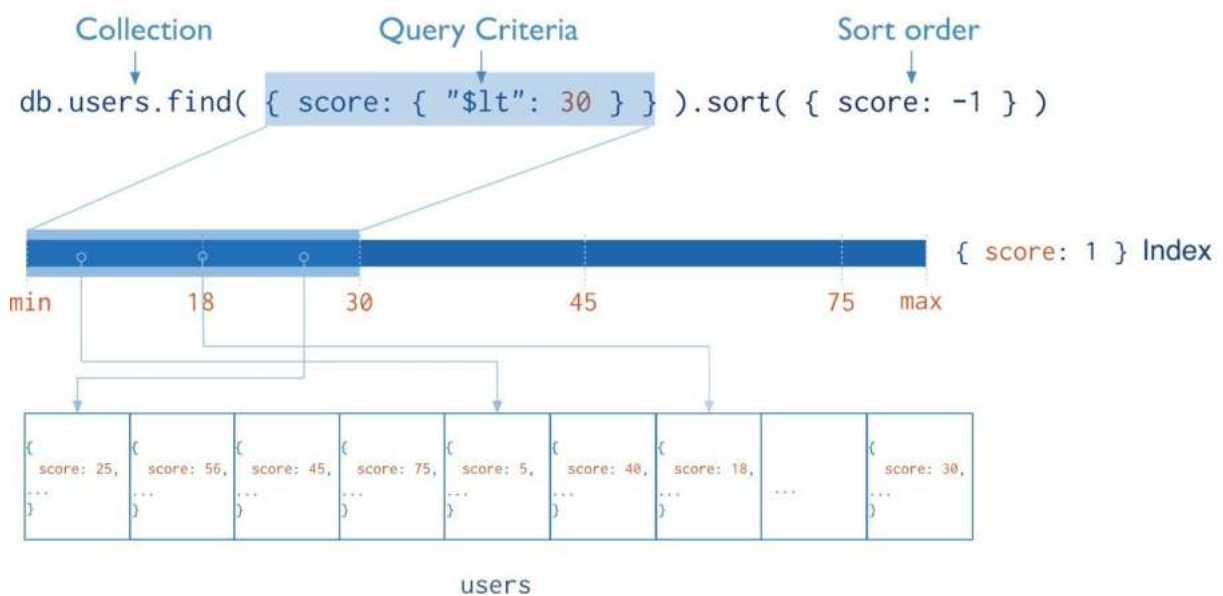
AIM:

To create and utilize indexes in MongoDB

DESCRIPTION:

Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a *collection scan*, i.e., scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

The following diagram illustrates a query that selects and orders the matching documents using an



index:

DEFAULT _ID INDEX:

MongoDB creates a unique index on the `_id` field during the creation of a collection. The `_id` index prevents clients from inserting two documents with the same value for the `_id` field. You cannot drop this index on the `_id` field.

INDEX NAMES:

The default name for an index is the concatenation of the indexed keys and each key's direction in the index (i.e., 1 or -1) using underscores as a separator.

For example, an index created on

`{ item : 1, quantity: -1 }` has the name `item_1_quantity_-1`.

INDEX TYPES:

- Single Field
- Compound Index
- Multikey Index
- Geospatial Index
- Text Search Indexes
- Hashed Indexes
- Clustered Indexes

CREATING AN INDEX:

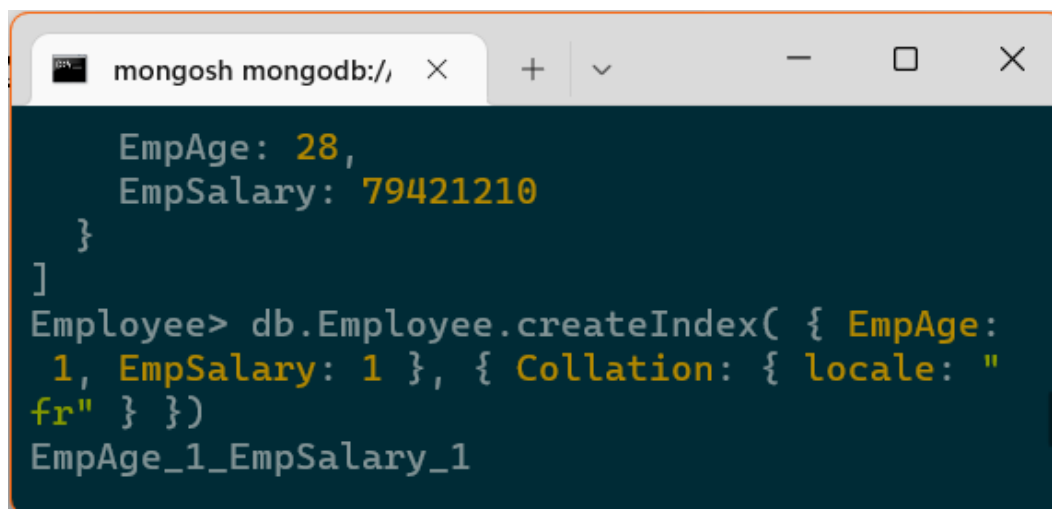
- Use the **Select your language** drop-down menu in the upper-right to set the language of the examples on this page.

To create an index in the Mongo Shell, use `db.collection.createIndex()`.

`db.collection.createIndex(<key and index type specification>, <options>)`

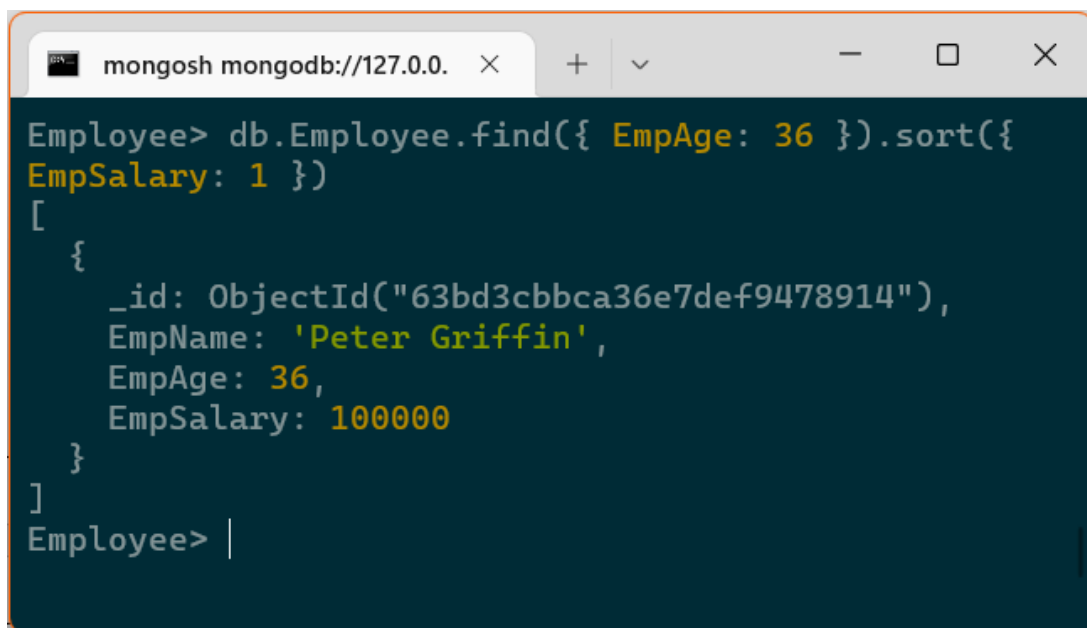
- The following example creates a single key descending index on the `name` field:

`db.collection.createIndex({ name: -1 })`



```
mongosh mongodb:/>
  EmpAge: 28,
  EmpSalary: 79421210
}
]
Employee> db.Employee.createIndex( { EmpAge:
1, EmpSalary: 1 }, { Collation: { locale: "
fr" } })
EmpAge_1_EmpSalary_1
```

- The `db.collection.createIndex()` method only creates an index if an index of the same specification does not already exist.

A screenshot of a MongoDB shell window. The title bar shows 'mongosh mongodb://127.0.0.1'. The command entered is `Employee> db.Employee.find({ EmpAge: 36 }).sort({ EmpSalary: 1 })`. The output is a JSON array containing one document: `[{ _id: ObjectId('63bd3cbbca36e7def9478914'), EmpName: 'Peter Griffin', EmpAge: 36, EmpSalary: 100000 }]`. The prompt `Employee> |` is visible at the bottom.

```
Employee> db.Employee.find({ EmpAge: 36 }).sort({
EmpSalary: 1 })
[
  {
    _id: ObjectId("63bd3cbbca36e7def9478914"),
    EmpName: 'Peter Griffin',
    EmpAge: 36,
    EmpSalary: 100000
  }
]
Employee> |
```

RESULT:

Thus, the creating of index has been completed and executed successfully.

EX.NO: 4.1

Installation of MongoDB Atlas

Date: 03. 01. 2023

AIM:

To install MongoDB Atlas in windows.

DESCRIPTION:

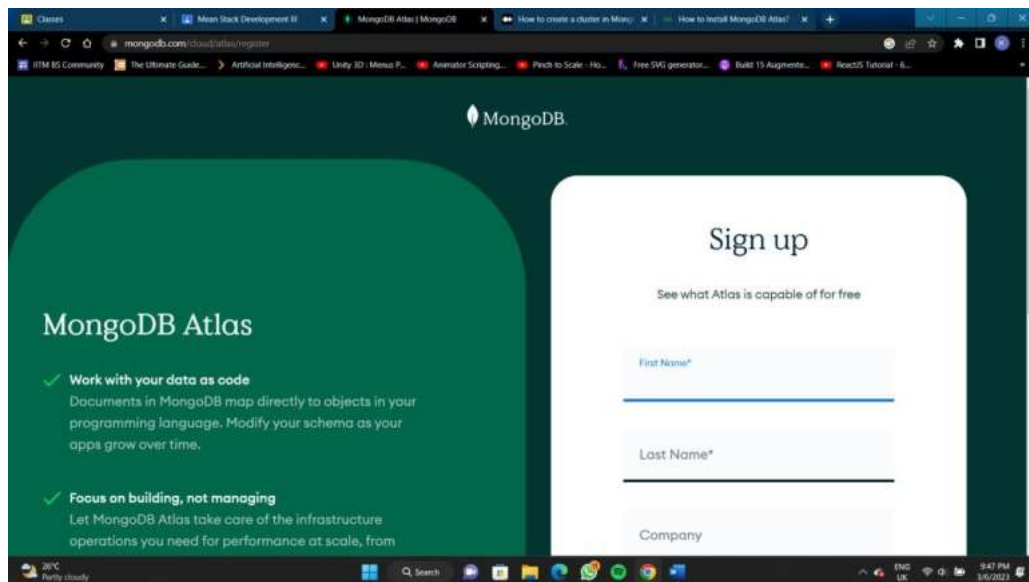
MongoDB has a cloud service also known as MongoDB Atlas. It is a cloud base database that allows us to host up our database and serve us whenever we need it. It removes all our effort to host our database on our local devices and can serve us anytime.

Features of MongoDB Atlas:

- Strong security
- More precise analysis of data
- Easy Scalability
- Technical support

PROCEDURE:

Step 1: Go to the **MongoDB** website and signup. Enter your credentials and click on **create account** and then verify the account or just click **sign up with google** and select the Id from which you want to register.



Step 2: Now select the **Shared option** and click on **Create** for free usage.

The screenshot displays two options for creating a MongoDB cluster:

- Dedicated:** For production applications with sophisticated workload requirements. Advanced configuration controls. Features include network isolation, on-demand performance advice, and multi-region/multi-cloud options. Starting at \$0.08/hr* (estimated cost \$56.94/month).
- Shared:** For learning and exploring MongoDB in a cloud environment. Basic configuration options. Features include no credit card required, sample datasets, and upgrade options. Starting at **FREE**.

Step 3: Now select cloud provider **AWS** or **Google cloud** for free hosting and then select the region in which it should be hosted. And in the Cluster tier select **M0 Sandbox** for the free environment. And If you want you can rename the cluster. It will take 4-5 minutes to set up the environment.

The screenshot shows the 'Cloud Provider & Region' selection interface. The 'aws' provider is selected, and the region 'N. Virginia (us-east-1)' is highlighted. The interface lists recommended and paid tier regions across North America, Europe, Australia, and Asia.

Cloud Provider & Region	Selected
aws	Google Cloud
Azure	
★ Recommended region ⓘ	🟡 Paid tier region ⓘ
NORTH AMERICA	EUROPE
🇺🇸 N. Virginia (us-east-1) ★	🇮🇪 Ireland (eu-west-1) ★
🇺🇸 Oregon (us-west-2) ★	🇸🇪 Stockholm (eu-north-1) ★
🇺🇸 Ohio (us-east-2) ★ 🟡	🇩🇪 Frankfurt (eu-central-1) ★
🇺🇸 N. California (us-west-1) 🟡	🇬🇧 London (eu-west-2) ★ 🟡
🇨🇦 Montreal (ca-central-1) 🟡	🇫🇷 Paris (eu-west-3) ★ 🟡
	AUSTRALIA
	🇦🇺 Sydney (ap-southeast-2) ★
	ASIA
	🇸🇬 Singapore (ap-southeast-1) ★
	🇮🇳 Mumbai (ap-south-1)
	🇯🇵 Tokyo (ap-northeast-1) ★

Step 4: Create a **username** and **password** then select database access to atlasAdmin and network access to allow access to anywhere. That will be the username and password through which the user can access the database given to who do you give access.

✓ How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password. You can manage existing users via the [Database Access Page](#).

Username

Password

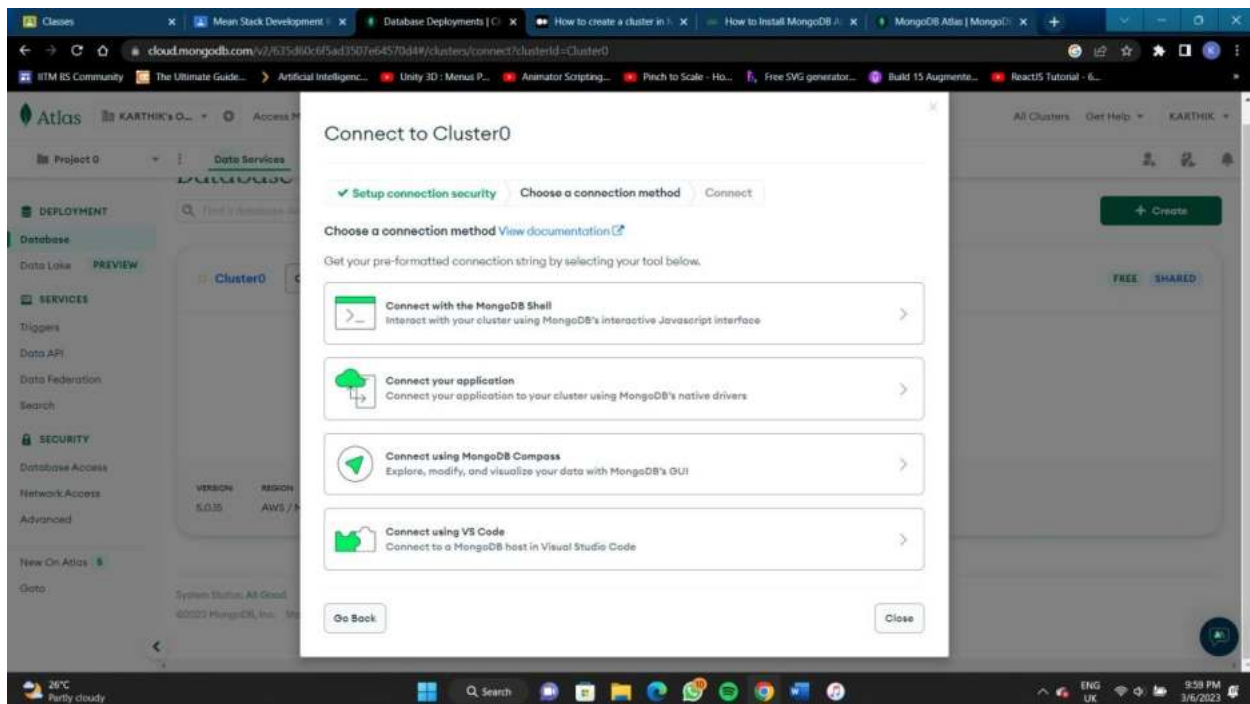
Enter username

Enter password

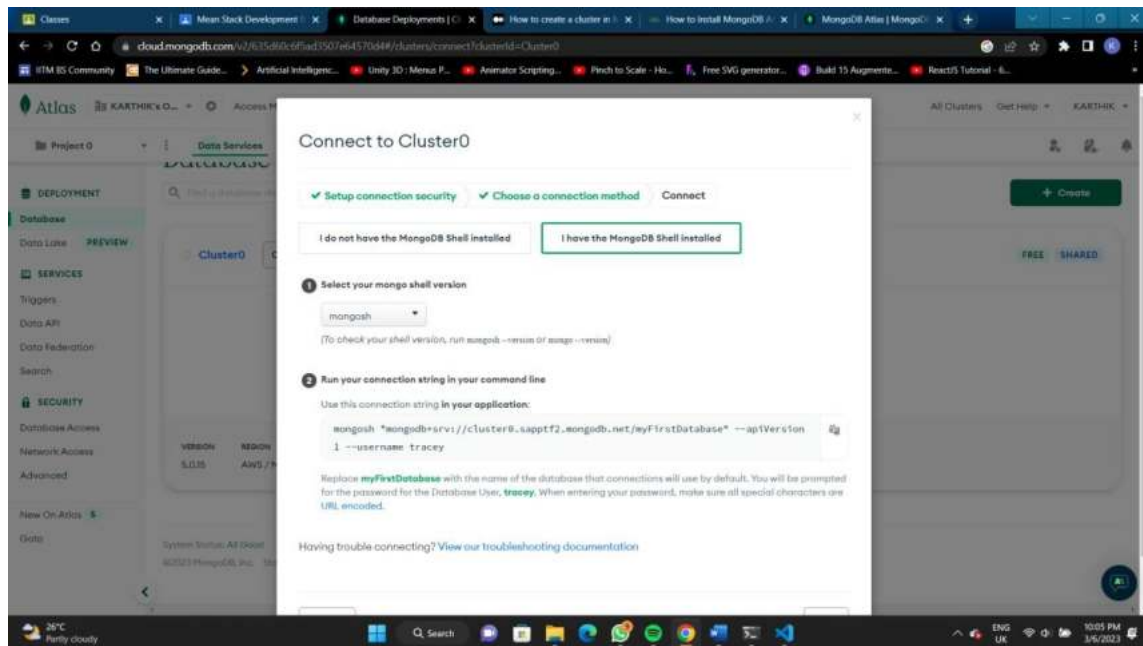
Create User

This password contains special characters which will be URL-encoded.

Step 5: Select **Connect** and then select **connect with the MongoDB Shell** then go to **I have MongoDB shell installed**. Select the node version from there. And after that run the command in the command line which is given. Then enter the password which the user has set earlier with the username and password.



Step 6: Go to **Connect** then **Connect your application** select your node version. Now copy the link given and paste it into your application where you connect MongoDB. And replace <password> with your password.



```
mongosh mongodb+srv://<c>
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karth>mongosh "mongodb+srv://cluster0.sapptf2.mongodb.net/myFirstDatabase" --apiVersion 1 --username tracey
Enter password: *****
Current Mongosh Log ID: 640617fe2b1553faa9143911
Connecting to:      mongodb+srv://<credentials>@cluster0.sapptf2.mongodb.net/myFirstDatabase?appName=mongosh+1.7.1
```

RESULT:

Thus, the connection between MongoDB Shell and MongoDB atlas has been made successfully.

EX.NO: 4.2	<u>Running a MongoDB cluster in MongoDB Atlas</u>
Date: 10. 01. 2023	

AIM:

To run MongoDB cluster in MongoDB Atlas.

DESCRIPTION:**Setting Up a MongoDB Cluster on Atlas**

Every application is different, and MongoDB Atlas provides you with numerous ways to set up your cluster to suit your specific needs. Some particular configurations need to be thought of ahead of time, while you can change others on the fly. Using these settings, you will put in place all the best practices for Atlas in production. In this section, you will learn more about the various configurations that you can adjust on your initial cluster creation.

Deployment Type

The deployment type is the first option you will need to pick. Based on what you decide for the type of instance, the other configuration options will vary.

- Serverless: This type of cluster is the most flexible one from a pricing point of view. It's meant for applications that have infrequent or variable traffic. The possible configurations are kept at the bare minimum.
- Dedicated: A dedicated cluster is meant for production loads. It can support a wide range of server sizes as well as advanced configurations. You should choose this for your production environment.
- Shared: These clusters are meant to be a way to explore MongoDB. They can provide you with a sandbox where you can try out MongoDB for free. The server configurations available are somewhat limited.

You can find more information about the different database deployment types in the documentation.

Global Cluster Configuration

If you need multiple sharded clusters with read and write operations in specific locations, you will need to enable Global Cluster Configuration. From here, you can choose exactly where you want each of your clusters and configure the mappings between the user country and the server they will use to access the data.

Cloud Provider and Region

No matter which deployment type you picked, you will need to choose the cloud provider, along with the specific region in which you want to deploy your cluster. You can instantiate MongoDB clusters on any of the three major cloud providers. If you want to ensure even better availability, you can deploy each node of your cluster on different regions or even different clouds. To do so, you will need to enable the Multi-Cloud, Multi-Region & Workload Isolation option. From here, you will be able to configure the number and types of nodes (electable, read-only, or analytical) that will be part of your replica set.

Cluster Tier

Now that you've picked a region and cloud provider, you will need to choose which tier you want to use for the nodes in your cluster. This configuration will have the most significant impact on the pricing of your cluster. There is a wide range of options available, and you can further tweak each of them. Take into consideration the amount of CPU and RAM you will need. Your resource needs will help you find the right tier for your cluster.

You can then further tweak the cluster configuration by adjusting the storage size, toggling the auto-scaling options and the IOPS that you will need. On AWS higher tiers (M40+), you will also be able to choose the class of servers (low-CPU, general, or local NVMe SSD), which will also impact the number of CPUs, RAM, and storage capacity.

PROCEDURE:

Step-1: Create a cluster in MongoDB Atlas by signing in to it.

Step-2: Connect the MongoDB Atlas with MongoDB shell using the provided connection string.

Step-3: After making the connection in the shell create a database in the cluster.

Step-4: Create collections in the database.

Step-5: Insert and retrieve document from the collections.


```
mongosh mongodb+srv://<c> X + -
(c) Microsoft Corporation. All rights reserved.

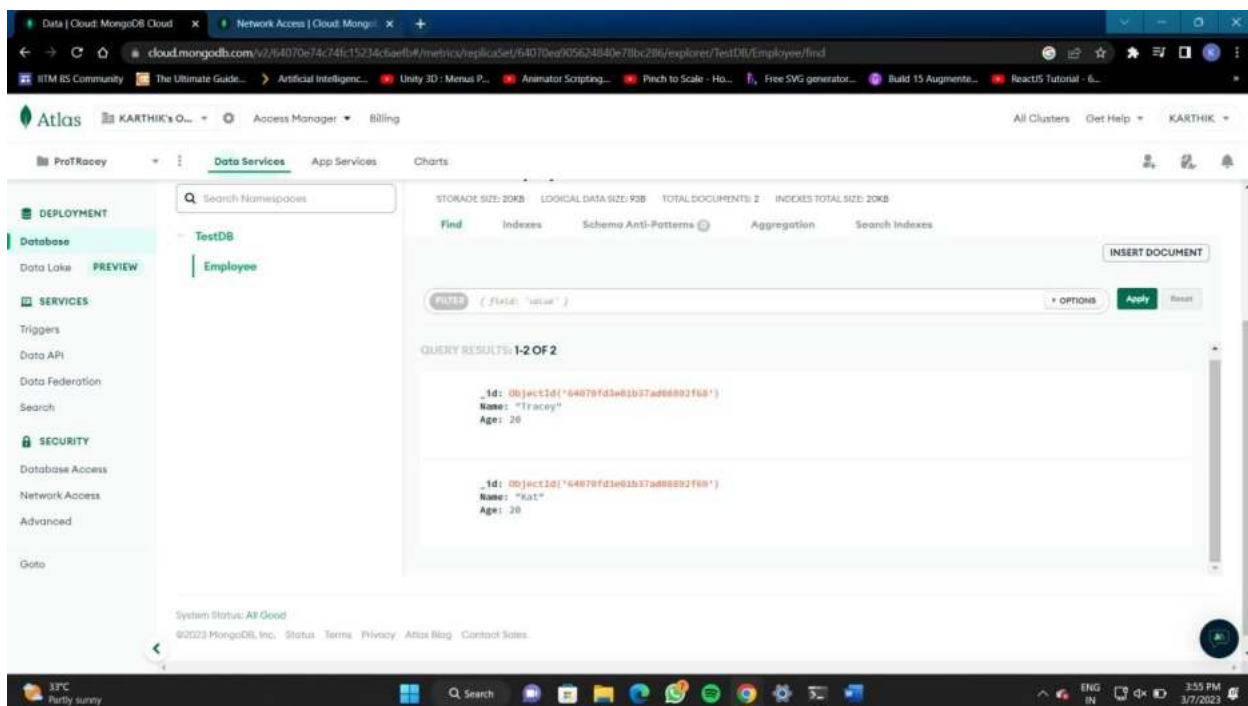
C:\Users\karth>mongosh "mongodb+srv://tracey.cmsrtxx.mongodb.net/myFirstDatabase" --apiVersion 1 --username karthik
Enter password: *****
Current Mongosh Log ID: 64070f4f8cd74cf14ce0d1d0
Connecting to:      mongodb+srv://<credentials>@tracey.cmsrtxx.mongodb.net/myFirstDatabase?appName=mongosh+1.7.1
Using MongoDB:      5.0.15 (API Version 1)
Using Mongosh:      1.7.1

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

Atlas atlas-86qu2o-shard-0 [primary] myFirstDatabase> use TestDB
switched to db TestDB
Atlas atlas-86qu2o-shard-0 [primary] TestDB> db.Employee.insertMany([{Name:'Tracey',Age:20},{Name:'Kat',Age:20}]]
Uncaught:
SyntaxError: Missing semicolon. (1:22)

> db.Employee.insertMany([{Name:'Tracey',Age:20},{Name:'Kat',Age:20}]]
      ^
SyntaxError: Missing semicolon. (1:22)

Atlas atlas-86qu2o-shard-0 [primary] TestDB> db.Employee.insertMany([{Name:'Tracey',Age:20},{Name:'Kat',Age:20}]]
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64070fd3e01b37ad08892f68"),
    '1': ObjectId("64070fd3e01b37ad08892f69")
  }
}
Atlas atlas-86qu2o-shard-0 [primary] TestDB> |
```



RESULT:

Thus, the MongoDB cluster is created in the MongoDB Atlas and successfully connected with the MongoDB Shell.

EX.NO: 5	<u>Communicating Components Through Binding in AngularJS</u>
Date: 24. 01. 2023	

AIM:

To create and communicate between components in AngularJS.

PROCEDURE:

Step-1: Define a model name using ng-model directive

Step-2: Create textboxes to add name in it.

Step-3: Add a controller to display the names entered names.

PROGRAM:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

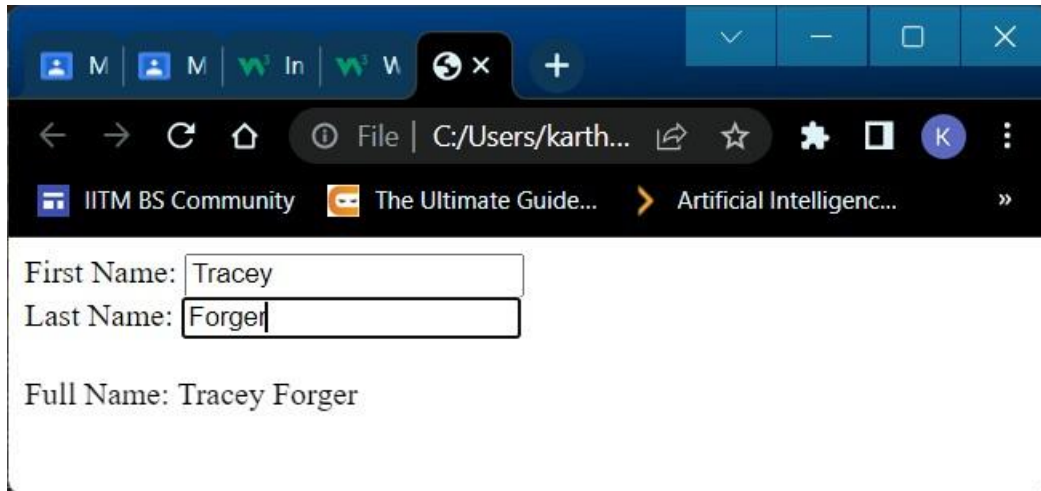
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
</script>

</body>
</html>
```

OUTPUT:

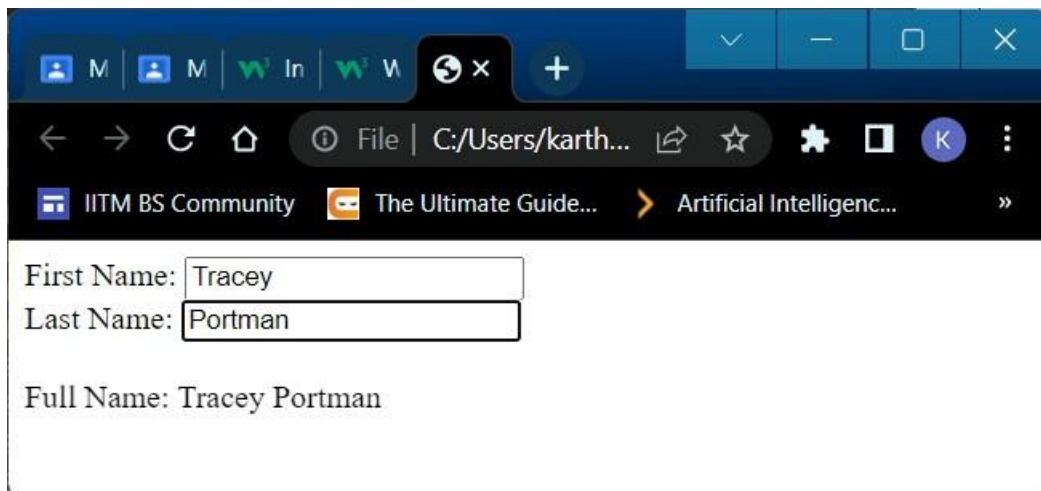


A screenshot of a web browser window. The address bar shows the file path C:/Users/karth... The browser tabs include 'IITM BS Community', 'The Ultimate Guide...', and 'Artificial Intelligenc...'. The form contains the following text:

First Name:

Last Name:

Full Name: Tracey Forger



A screenshot of a web browser window, similar to the one above. The address bar shows the file path C:/Users/karth... The browser tabs include 'IITM BS Community', 'The Ultimate Guide...', and 'Artificial Intelligenc...'. The form contains the following text:

First Name:

Last Name:

Full Name: Tracey Portman

RESULT:

Thus, the creation of communication between components in AngularJs has been completed and executed successfully.

EX.NO: 6	<u>HTTP Services in AngularJS</u>
Date: 31. 01. 2023	

AIM:

To use HTTP services in AngularJS.

PROCEDURE:

Step-1: Create a html page

Step-2: Include the angular js cdn

Step-3: create a app variable using module().

Step-4: Use \$http.get() to receive the data from the server. On successful retrieval display the data. Else display the error.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myApp">
<h1>AngularJS $http Demo: </h1>
  <div>
    <div ng-controller="myController">
      Response Data: {{data}} <br />
      Error: {{error}}
    </div>
  </div>
<script>
  var myApp = angular.module('myApp', []);

  myApp.controller("myController", function ($scope, $http) {

    var onSuccess = function (data, status, headers, config) {
      $scope.data = data;
    };

    var onError = function (data, status, headers, config) {
```

```
        $scope.error = status;
    }

    var promise = $http.get("/demo/getdata");

    promise.success(onSuccess);
    promise.error(onError);

    });
</script>
</body>
</html>
```

OUTPUT:



RESULT:

Thus, the http service has been successfully used in the AngularJS and executed successfully.

EX.NO: 7	<u>Installation of Node.JS</u>
Date: 07. 02. 2023	

AIM:

To install node.js in windows

PROCEDURE:**What is Node js?**

Node.js is an open source, cross-platform runtime environment and library that is used for running web applications outside the client's browser.

It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based architectures in mind. Every browser has its own version of a JS engine, and node.js is built on Google

Chrome's V8 JavaScript engine. Sounds a bit complicated, right?

In simple terms, what this means is that entire sites can be run using a unified 'stack', which makes development and maintenance quick and easy, allowing you to focus on meeting the business goals of the project.

The fact that Node.js is open source means that it is free to use and constantly being tweaked and improved by a global community of developers.

An important thing to understand about Node.js is that it is actually neither a framework or a library - as with traditional application software -, but a runtime environment.

A runtime environment (sometimes shortened to RTE) contains Web API's that a developer can access to build a code, and a JavaScript engine that parses that code. This makes it lightweight, flexible and easy to deploy, all features that will help to optimize and speed up your application project.

What is NPM?

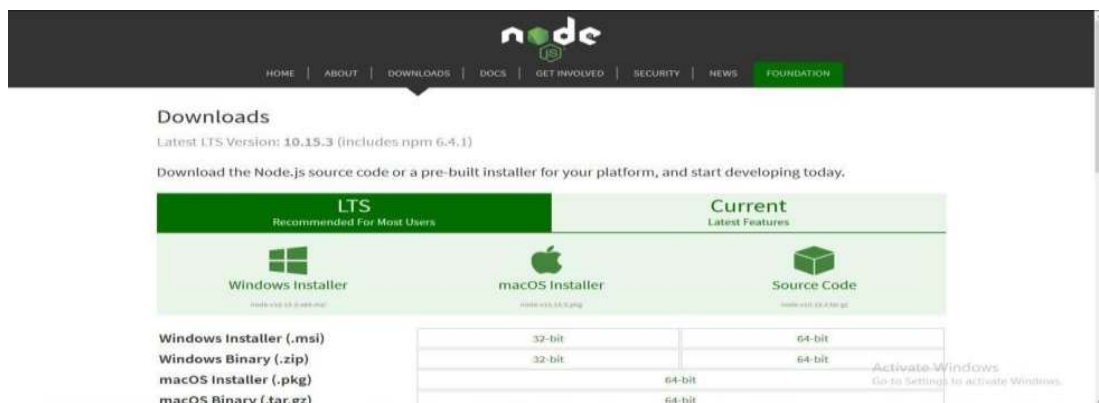
NPM is a package manager for Node. js packages, or modules if you like. www.npmjs.com hosts thousands of free packages to download and use. The NPM program is installed on your computer when you install Node.js. npm install downloads a package and it's dependencies. npm install can be run with or without arguments. When run without arguments, npm install downloads dependencies defined in a package. json file and generates a node_modules folder with the installed modules.

STEPS TO INSTALL

NODE:

Step-1: Download the installer

The first step to install Node.js on windows is to download the installer. Visit the official Node.js website i.e) <https://nodejs.org/en/download/> and download the .msi file according to your system environment (32-bit & 64-bit). An MSI installer will be downloaded on your system.



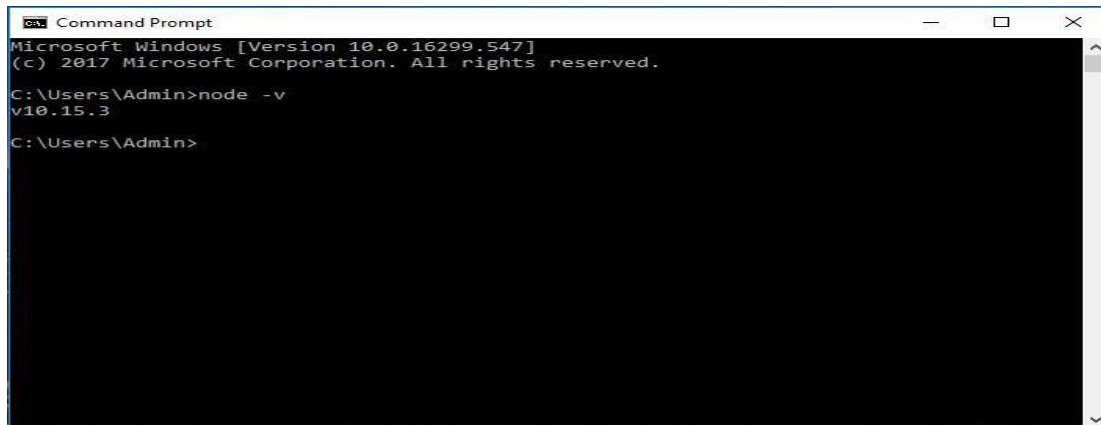
Step-2: Running the Node.js installer.



Step 3: Verify that Node.js was properly installed or not.

To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows Powershell and test it:

```
C:\Users\Admin> node -v
```



```
Command Prompt
Microsoft Windows [Version 10.0.16299.547]
(c) 2017 Microsoft Corporation. All rights reserved.

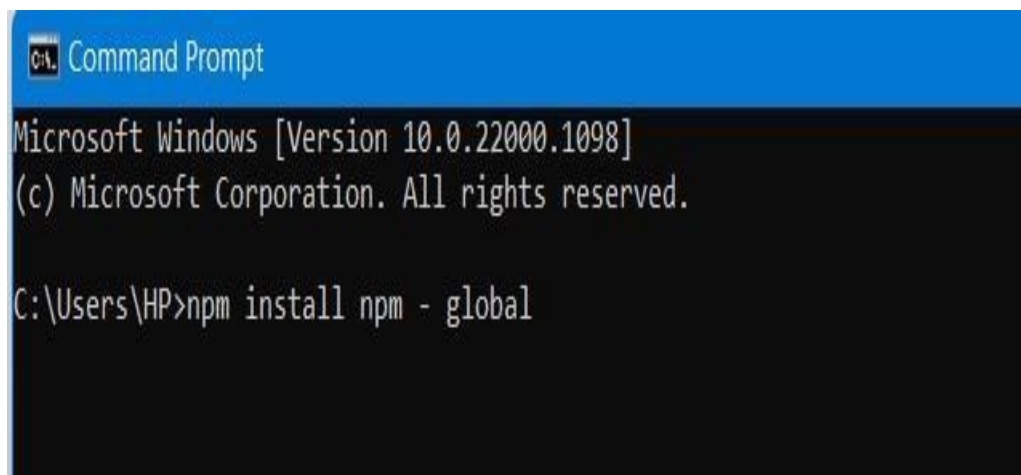
C:\Users\Admin>node -v
v10.15.3

C:\Users\Admin>
```

Step 4: Updating the Local npm version.

The final step in node.js installed is the updation of your local npm version(if required) – the package manager that comes bundled with Node.js. You can run the following command, to quickly update the npm

npm install npm –global // Updates the ‘CLI’ client



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>npm install npm - global
```

Result:

Thus Node.JS has been installed successfully.

EX.NO: 8.1	<u>Module Creation in Node.js</u>
Date: 14. 02. 2023	

Aim:

To learn and create modules in Node.js

What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

Built-in Modules

Node.js has a set of built-in modules which you can use without any further installation.

Look at our [Built-in Modules Reference](#) for a complete list of modules.

PROCEDURE

To include a module, use the **require()** function with the name of the module:

```
var http = require('http');
```

Now your application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello World!');  
}).listen(8080);
```

Create Your Own Modules

You can create your own modules, and easily include them in your applications.

The following example creates a module that returns a date and time object:

Create a module that returns the current date and time:

```
exports.myDateTime = function () {  
  return Date();  
};
```

Use the **exports** keyword to make properties and methods available outside the module file.

Save the code above in a file called "myfirstmodule.js"

Include Your Own Module

Now you can include and use the module in any of your Node.js files.
Use the module "myfirstmodule" in a Node.js file:

PROGRAM:

myModule.js

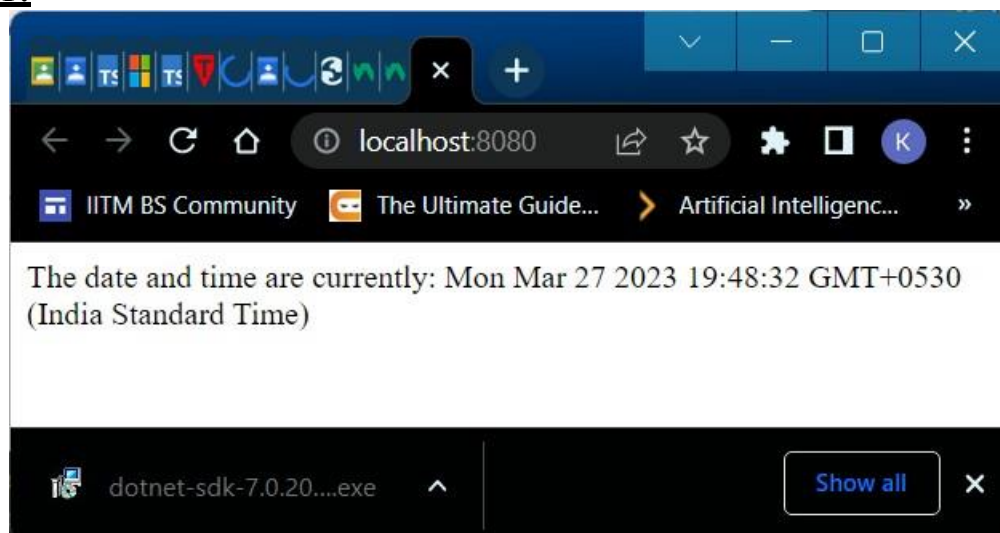
```
exports.myDateTime = function () {  
    return Date();  
};
```

myFirstModule.js

```
var http = require('http');  
var dt = require('./myModule');
```

```
http.createServer(function (req, res) {  
    res.writeHead(200, {'Content-Type': 'text/html'});  
    res.write("The date and time are currently: " + dt.myDateTime());  
    res.end();  
}).listen(8080);
```

OUTPUT:



RESULT:

Thus the module has been created and used successfully.

EX.NO: 8.2	<u>Interaction with file system in Node.js</u>
Date: 21. 02. 2023	

AIM:

To create a node js application to interact with the file system and serve a webpage from a file.

ALGORITHM:

- 1) START
- 2)First install all the necessary packages such as express mongoose using “npm – I”.
- 3) Create a main file to be starting point of the projecct and other sub files are connected to the main file.
- 4) Using express() function create a route in all files.
- 5) Using app.listen() we can handle the port of which it is running.
- 6) Create a model for every document to be stored in the database using mongoose.schema.
- 7)Inside the model give the required fields and type of data.
- 8) Now create function to get and post request to the database or to the frontend application.
- 9)END

PROCEDURE:

File Paths

Managing the file system means we have to deal with relative paths, absolute paths, resolving file paths by providing file names, joining file paths, etc. We can get these using the path module. The path module takes care of all this irrespective of the operating system such as windows, Linux, Mac, etc. Here is a simple example of using the path module.

Joining Paths

You can pass as many arguments as you want to the join method and this will create the full path depending on the platform. In the above file, we have used the join method to create paths at line numbers 4 and 6.

Relative Paths

You can find the relative path between two absolute paths using the relative method. The first argument is the from the path and the second argument is the to path and the result would be a relative path from the first argument to the second argument.

Resolving Paths

This accepts multiple string arguments and resolves the path and prints the whole path. This is like going to the location and printing the current working directory. For example, in the above example, we have given three string arguments folder1, folder2, folder3 and it prints the complete path.

Parsing Paths

If you want to deconstruct the path and return the object that contains a directory, file name, and extension, etc. You should use the parse function from the path module.

PROGRAM:

File Paths1:

```
const { join, relative, resolve, parse } = require('path');
console.log('File ', join(__dirname, 'files', 'sample.txt'));
console.log('File ', join(__dirname, 'files', 'files2', 'files3', 'files4', 'sample.txt'));
// Finding the relative paths between Absolute Paths
console.log('Relative Path ', relative('/Users/bhargavbachina/Projects/nodejs/nodejs-
filesystem/file1.js', '/Users/bhargavbachina/Projects/nodejs/nodejs-file-system/files/sample1.txt'))
// Resolving the Path
console.log('Resolving the path', resolve('folder1', 'folder2', 'folder3'));
// Parsing the path
console.log('Parsing the path', parse('/Users/bhargavbachina/Projects/nodejs/nodejs-file-
system/file1.js'));
```

File Paths2:

```
const { join } =
require('path'); const
fs = require('fs');
fs.readFile(join(__dirname, './files/sample1.txt'), {encoding: 'utf8'}, (err,
data) => { if(err) {
    console.log('Error ', err)
  }
})
```

```
    console.log(data);
  })
```

PROMISED BASED:

```
const { join } =
require('path'); const
fs =
require('fs').promise
s;
const fileContent = fs.readFile(join(__dirname, '../files/sample1.txt'), {encoding:
'utf8'}); fileContent
    .then(data => console.log(data))
    .catch(err => console.log('Error ', err))
```

WRITING FILES:

```
const { join } = require('path');
const { readFile, writeFile } = require('fs');
readFile(join(__dirname, '../files/sample1.txt'), {encoding: 'utf8'}, (err,
data) => {    if(err) {

console.log('Er
ror ', err)    }
writeFile(join(
__dirname,
'../out/out1.txt')
,
data.toUpperCase
se(), (err) => {
if(err) {
    console.log('Error:::', err);
    }
});
})
```

STREAM BASED:

```
const { join } =
require('path'); const
fs = require('fs');
const readableStream = fs.createReadStream(join(__dirname, '../files/sample1.txt'),
{encoding: 'utf8'}); readableStream.on('data', (data) => { console.log('Data Received
', data);
}) readableStream.on('end', () =>
console.log("No more data!!!"));
readableStream.on('error', (err) =>
console.log('Error ', err));
```

OUTPUT:

```
Bhargavs-MacBook-Pro:reading bhargavbachina$ node example-sync-based.js
Contents This is sample text1 this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file This is sample text1
this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file This is sample text1
this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file This is sample text1
this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file This is sample text1
this is sample data from the sample text file This is sample text1 this is sample data from the sample text file
This is sample text1 this is sample data from the sample text file This is sample text1
this is sample data from the sample text file
```

```
1 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
2 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
3 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
4 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
5 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
6 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
7 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
8 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
9 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
10 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
11 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
12 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
13 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
14 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
15 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE
16 THIS IS SAMPLE TEXT1 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE THIS IS SAMPLE TEXT1
17 THIS IS SAMPLE DATA FROM THE SAMPLE TEXT FILE
```

RESULT:

Interaction with the file system and serving a webpage from a file has been successfully executed.

EX.NO: 9	<u>Installation of Express JS</u>
Date: 28. 02. 2023	

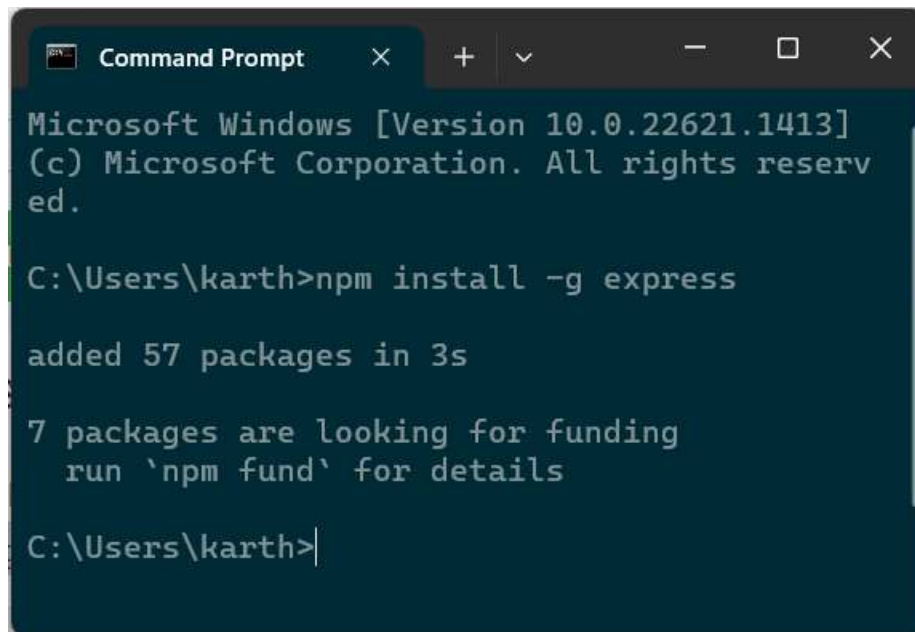
AIM:

To install Express JS in windows.

PROCEDURE:

Step-1: Install the express framework globally to create web application using Node terminal. Use the following command to install express framework globally.

`npm install -g express`



```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karth>npm install -g express

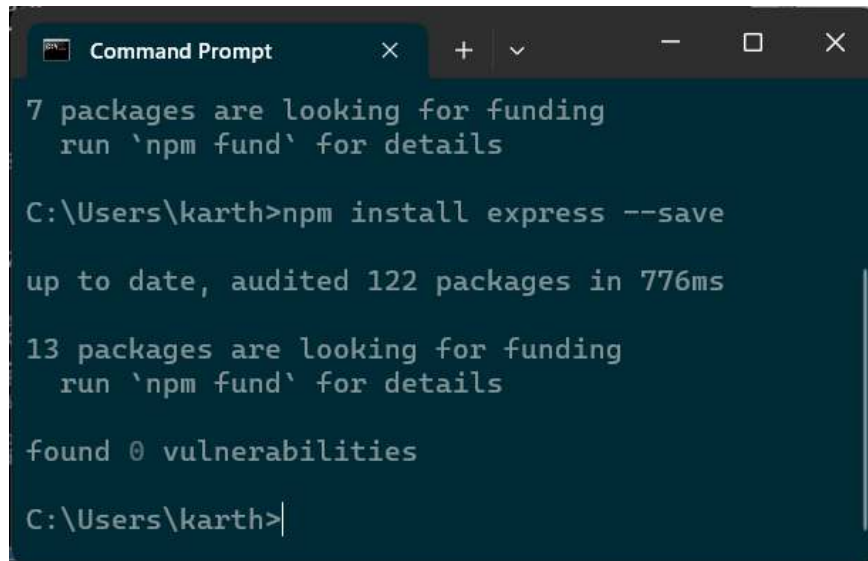
added 57 packages in 3s

7 packages are looking for funding
  run `npm fund` for details

C:\Users\karth>
```

Step-2: Use the following command to install express

`npm install express --save`



```
7 packages are looking for funding
  run 'npm fund' for details

C:\Users\karth>npm install express --save

up to date, audited 122 packages in 776ms

13 packages are looking for funding
  run 'npm fund' for details

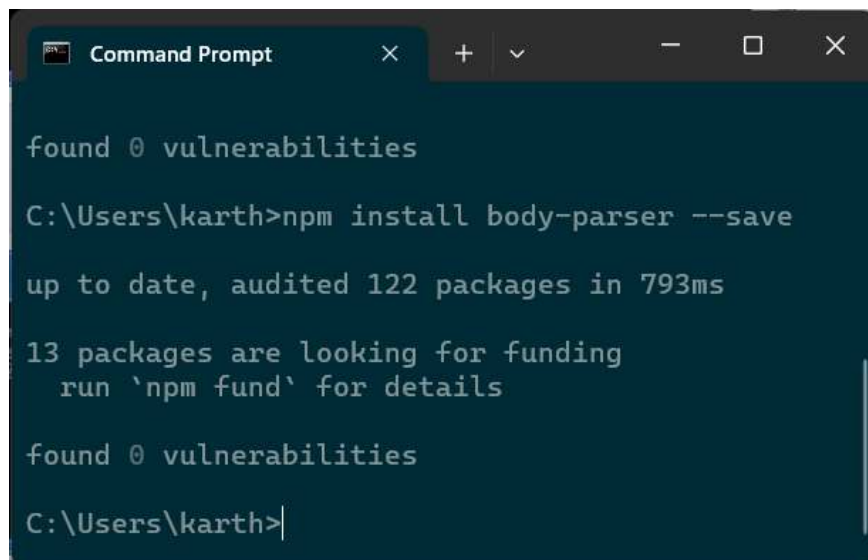
found 0 vulnerabilities

C:\Users\karth>|
```

The above command install express in node_module directory and create a directory named express inside the node_module. You should install some other important modules along with express. Following is the list:

- **body-parser:** This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
- **cookie-parser:** It is used to parse Cookie header and populate req.cookies with an object keyed by the cookie names.
- **multer:** This is a node.js middleware for handling multipart/form-data.

npm install body-parser --save



```
found 0 vulnerabilities

C:\Users\karth>npm install body-parser --save

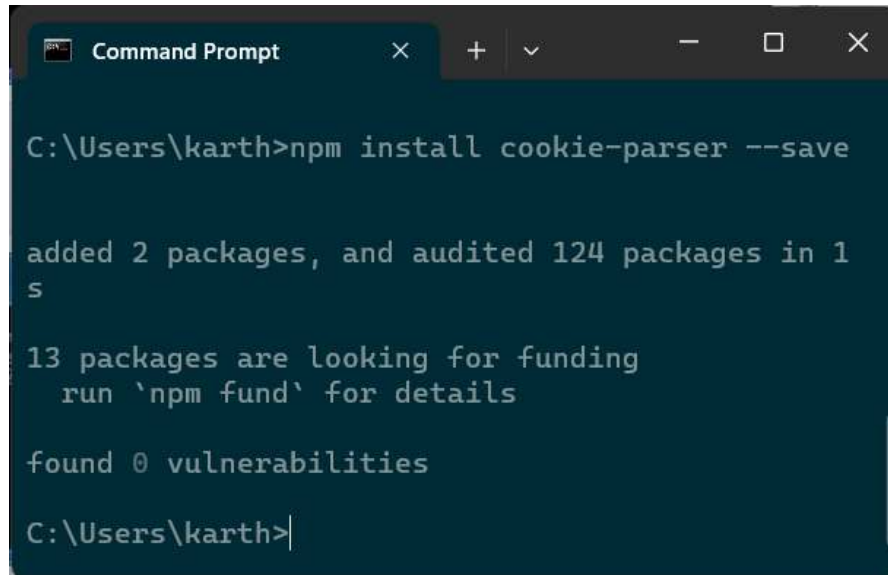
up to date, audited 122 packages in 793ms

13 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities

C:\Users\karth>|
```

npm install cookie-parser --save



```
Command Prompt
C:\Users\karth>npm install cookie-parser --save

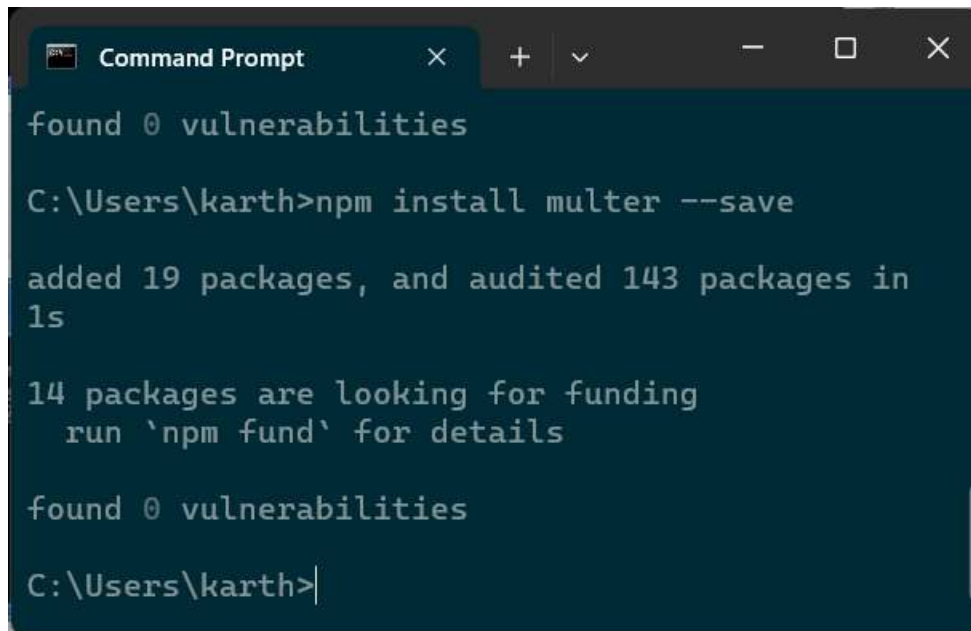
added 2 packages, and audited 124 packages in 1
s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\karth>
```

npm install multer --save



```
Command Prompt
found 0 vulnerabilities
C:\Users\karth>npm install multer --save

added 19 packages, and audited 143 packages in
1s

14 packages are looking for funding
  run `npm fund` for details

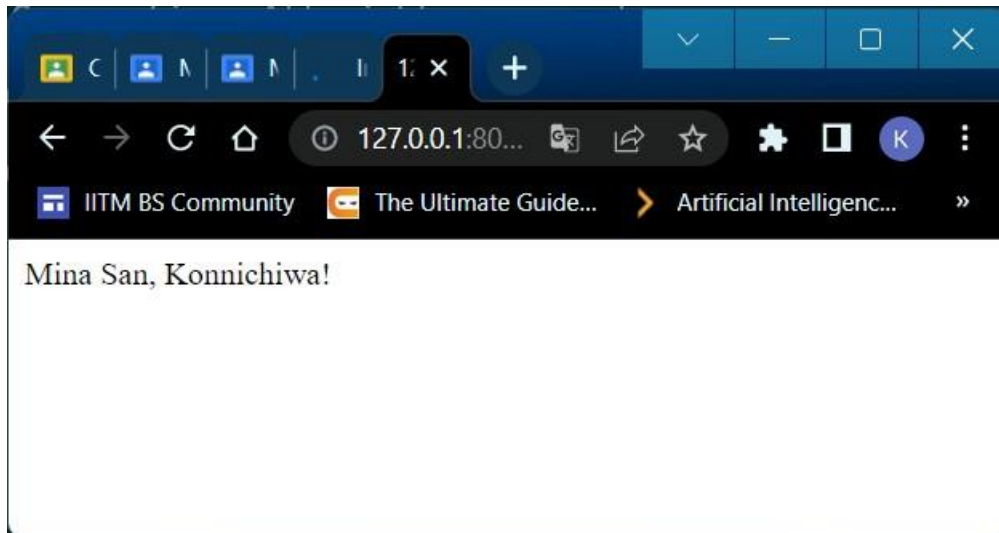
found 0 vulnerabilities

C:\Users\karth>
```

Step-3: Create a simple express which starts a server and listen on a local port. It only responds to homepage. For every other path, it will respond with a 404 Not Found error.

PROGRAM:

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
  res.send('Mina San, Konnichiwa!');
})
var server = app.listen(8000, function () {
var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

OUTPUT:**RESULT:**

Thus, the installation of Express JS has been completed successfully.

EX.NO: 10. 1	<u>ExpressJS Middleware functions</u>
Date: 07. 03. 2023	

AIM:

To understand the concepts of ExpressJS middleware and to implement them.

DESCRIPTION:**Express.js Middleware**

Express.js Middleware are different types of functions that are invoked by the Express.js routing layer before the final request handler. As the name specified, Middleware appears in the middle between an initial request and final intended route. In stack, middleware functions are always invoked in the order in which they are added.

Middleware is commonly used to perform tasks like body parsing for URL-encoded or JSON requests, cookie parsing for basic cookie handling, or even building JavaScript modules on the fly.

What is a Middleware function

Middleware functions are the functions that access to the request and response object (req, res) in request-response cycle.

A middleware function can perform the following tasks:

- It can execute any code.
- It can make changes to the request and the response objects.
- It can end the request-response cycle.
- It can call the next middleware function in the stack.

Express.js Middleware

Following is a list of possibly used middleware in Express.js app:

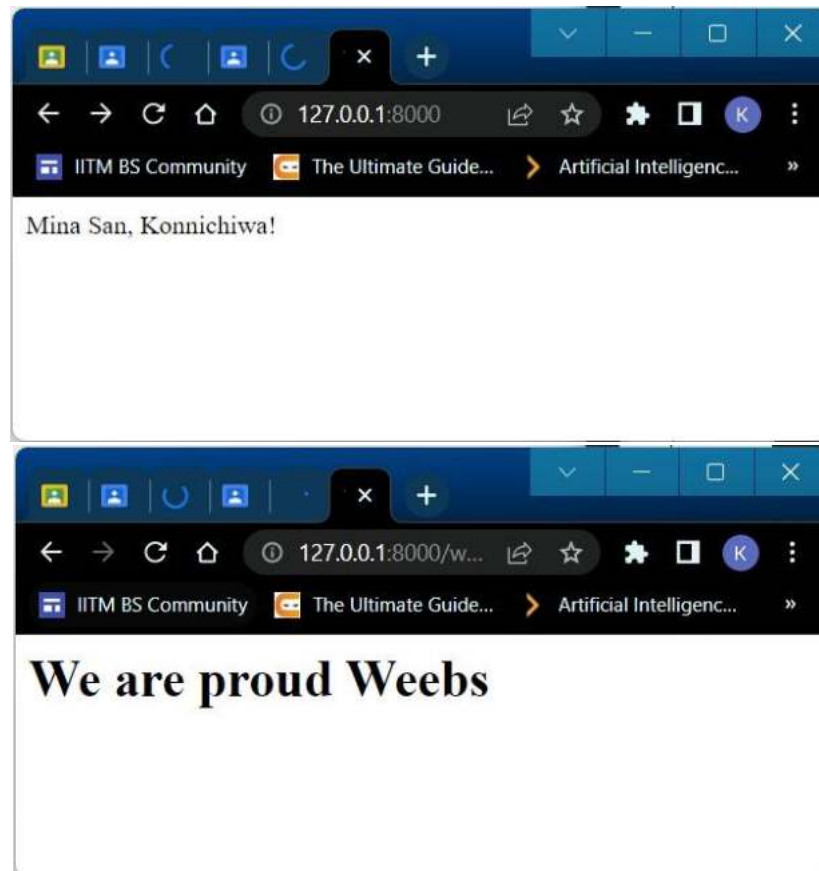
- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware

PROGRAM:

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('</h1>Mina San, Konnichiwa!</h1>');
});
app.get('/help', function(req, res) {
  res.send('<h1>We are proud Weebs</h1>');
});
var server = app.listen(8000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

OUTPUT:



RESULT:

Thus, the ExpressJS middleware functions have been implemented successfully.

EX.NO: 10. 2	<u>ExpressJS Middleware functions</u>
Date: 14. 03. 2023	

AIM:

To understand the concepts of ExpressJS Scaffold and to implement them.

DESCRIPTION:**What is scaffolding**

Scaffolding is a technique that is supported by some MVC frameworks.

It is mainly supported by the following frameworks:

Ruby on Rails, OutSystems Platform, Express Framework, Play framework, Django, MonoRail, Brail, Symfony, Laravel, CodeIgniter, Yii, CakePHP, Phalcon PHP, Model-Glue, PRADO, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET etc.

Scaffolding facilitates the programmers to specify how the application data may be used. This specification is used by the frameworks with predefined code templates, to generate the final code that the application can use for CRUD operations (create, read, update and delete database entries).

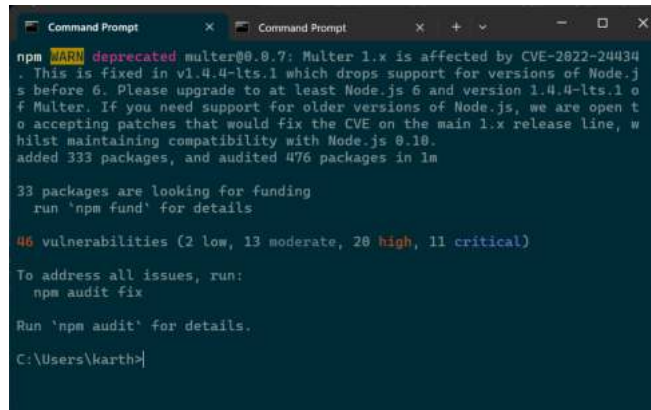
Express.js Scaffold

An Express.js scaffold supports candy and more web projects based on Node.js.

Install scaffold

Execute the following command to install scaffold.

npm install express-scaffold



```
Command Prompt
npm WARN deprecated multer@0.8.7: Multer 1.x is affected by CVE-2022-24434. This is fixed in v1.4.4-lts.1 which drops support for versions of Node.js before 6. Please upgrade to at least Node.js 6 and version 1.4.4-lts.1 of Multer. If you need support for older versions of Node.js, we are open to accepting patches that would fix the CVE on the main 1.x release line, whilst maintaining compatibility with Node.js 0.10.
added 333 packages, and audited 476 packages in 1m

33 packages are looking for funding
  run 'npm fund' for details

46 vulnerabilities (2 low, 13 moderate, 28 high, 11 critical)

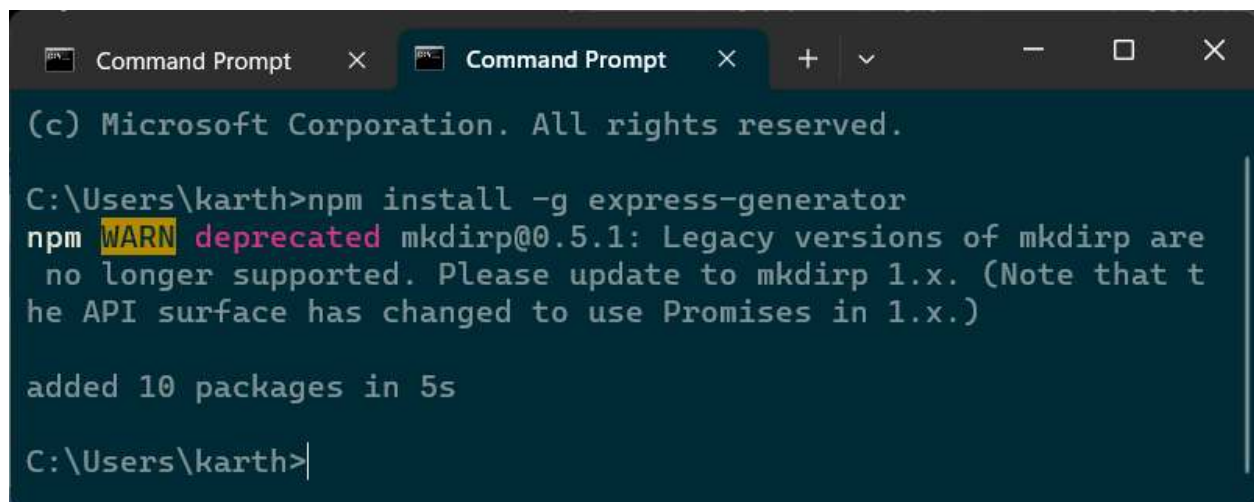
To address all issues, run:
  npm audit fix

Run 'npm audit' for details.
C:\Users\karth>
```

It will take a few seconds and the screen will look like this:

After this step, execute the following command to install express generator:

npm install -g express-generator



```
Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Users\karth>npm install -g express-generator
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has changed to use Promises in 1.x.)

added 10 packages in 5s

C:\Users\karth>
```

PROCEDURE:

- Step-1: Create a directory name myApp. Create a file name app.js in the myApp directory having the below mentioned program.
- Step-2: Open Node.js command prompt, go to myApp and run init command
- Step-3: Fill the and entries and press enter
- Step-4: It will create a package.json file in myApp folder and the data is show in JSON format.

PROGRAM:

```
var express = require('express');

var app = express();

app.get('/', function (req, res) {

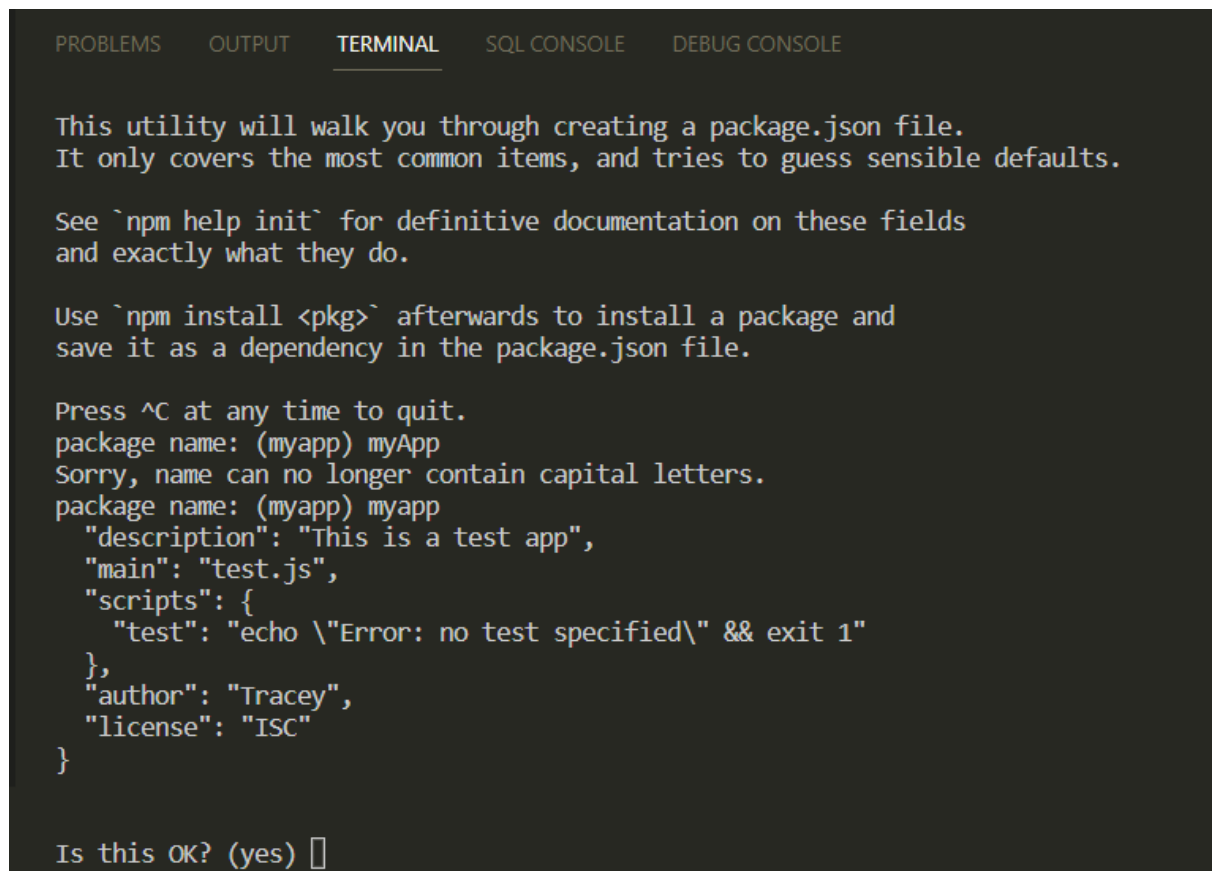
  res.send('Welcome to JavaTpoint!');

});

app.listen(8000, function () {

  console.log('Example app listening on port 8000!');

});
```

OUTPUT:

```
PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE

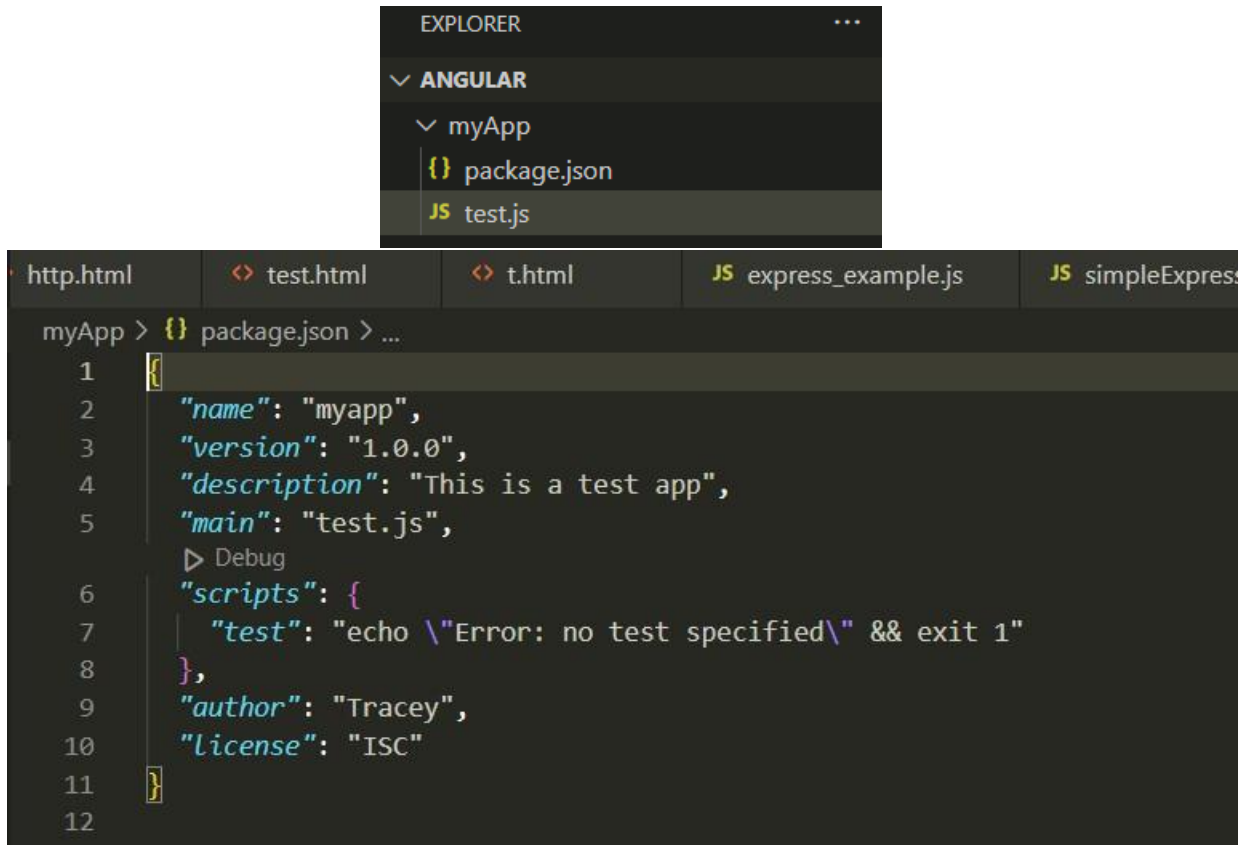
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (myapp) myApp
Sorry, name can no longer contain capital letters.
package name: (myapp) myapp
  "description": "This is a test app",
  "main": "test.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Tracey",
  "license": "ISC"
}

Is this OK? (yes) [ ]
```

RESULT:

Thus, the ExpressJS Scaffold has been successfully implemented.

	<u>Web Application using TypeScript and ASP.Net</u>
Date: 21. 03. 2023	

Aim:

To create a web application using typescript and ASP.Net

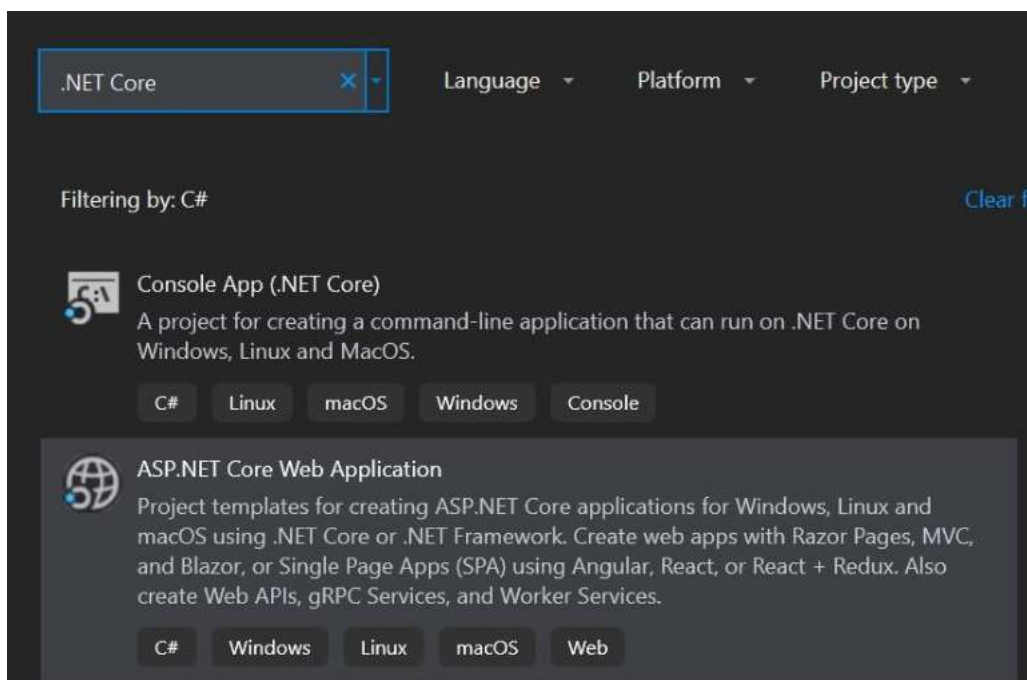
Install ASP.NET Core and TypeScript

First, install ASP.NET Core if you need it. This quick-start guide requires Visual Studio 2015 or 2017.

Next, if your version of Visual Studio does not already have the latest TypeScript, you can install it.

Create a new project

1. Choose **File**
2. Choose **New Project** (Ctrl + Shift + N)
3. Search for **.NET Core** in the project search bar
4. Select **ASP.NET Core Web Application** and press the *Next* button



5. Name your project and solution. After select the *Create* button

Configure your new project

ASP.NET Core Web Application C# Windows Linux macOS Web

Project name

Netcore+TS

Location

C:\Users\gabriellecrevecoeur\source\repos

Solution

Create new solution

Solution name ⓘ


Netcore+TS

☐ Place solution and project in the same directory


6. In the last window, select the **Empty** template and press the *Create* button

Create a new ASP.NET Core Web Application


.NET Core ASP.NET Core 2.2

 **Empty**


An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

 **API**


A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

 **Web Application**


A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.

 **Web Application (Model-View-Controller)**

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

 **Razor Class Library**

A project template for creating a Razor class library.

 **Angular**

[Get additional project templates](#)

Authentication

No Authentication

[Change](#)

Advanced

☒ Configure for HTTPS

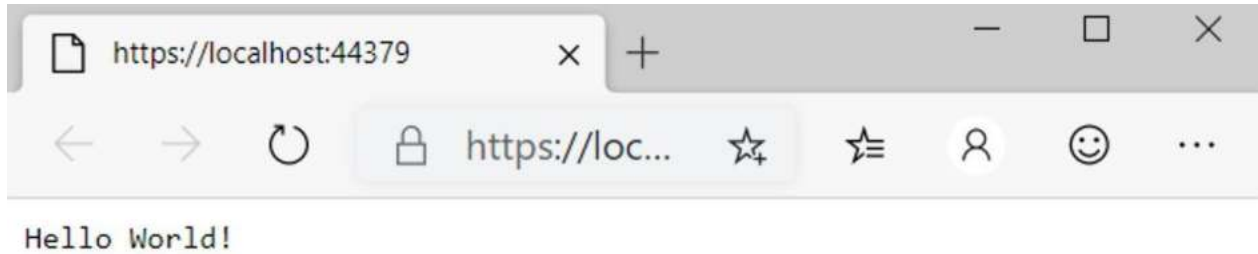
☐ Enable Docker Support
(Requires [Docker Desktop](#))

Linux

Author: Microsoft
Source: SDK 2.2.402

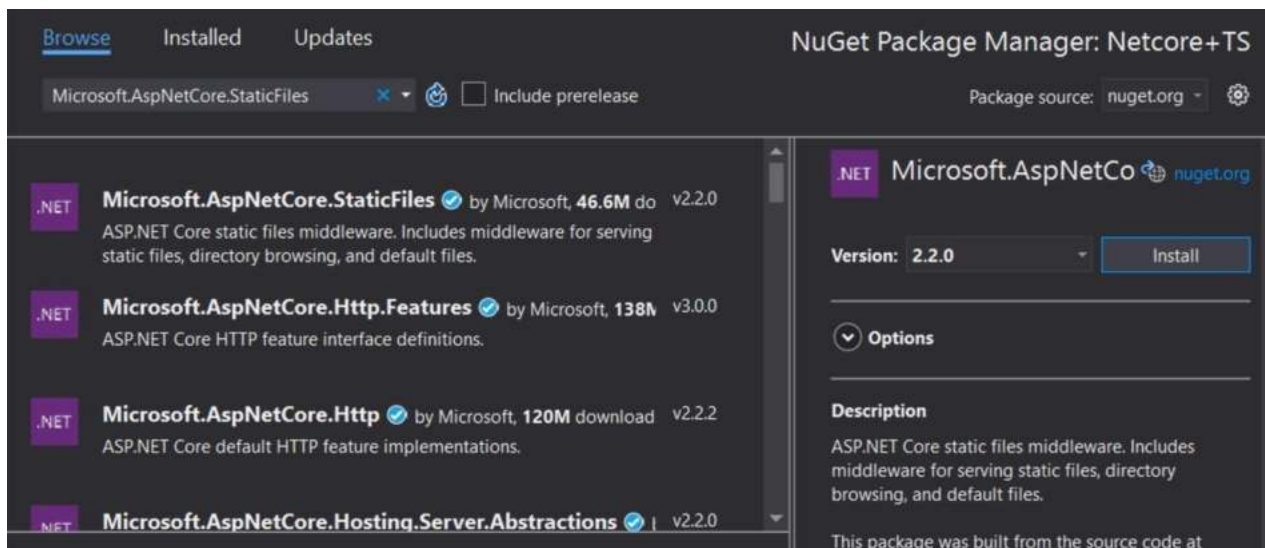
[Back](#) [Create](#)

Run the application and make sure that it works.



Set up the server

Open **Dependencies > Manage NuGet Packages > Browse**. Search and install Microsoft.AspNetCore.StaticFiles and Microsoft.TypeScript.MSBuild:



Open up your Startup.cs file and edit your Configure function to look like this:

```
public void Configure(IApplicationBuilder app, IHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

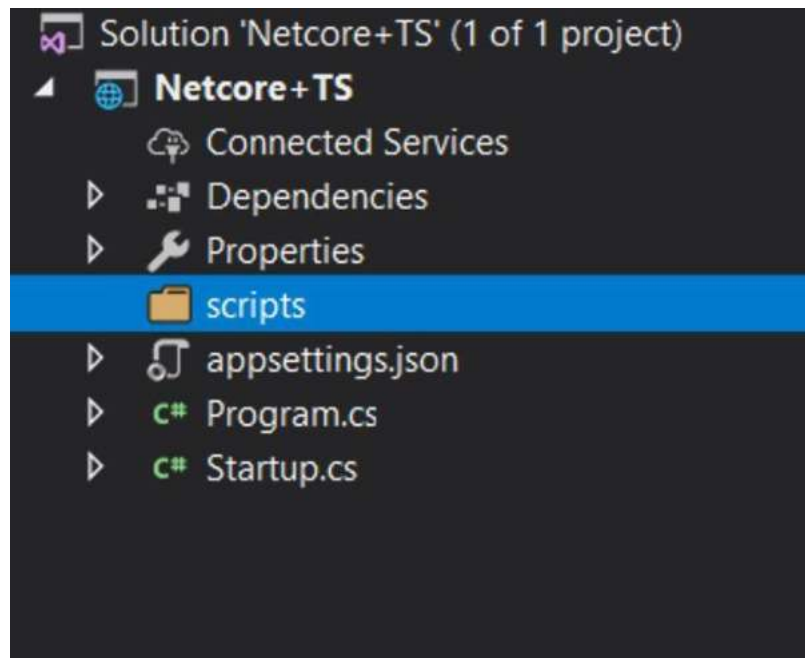
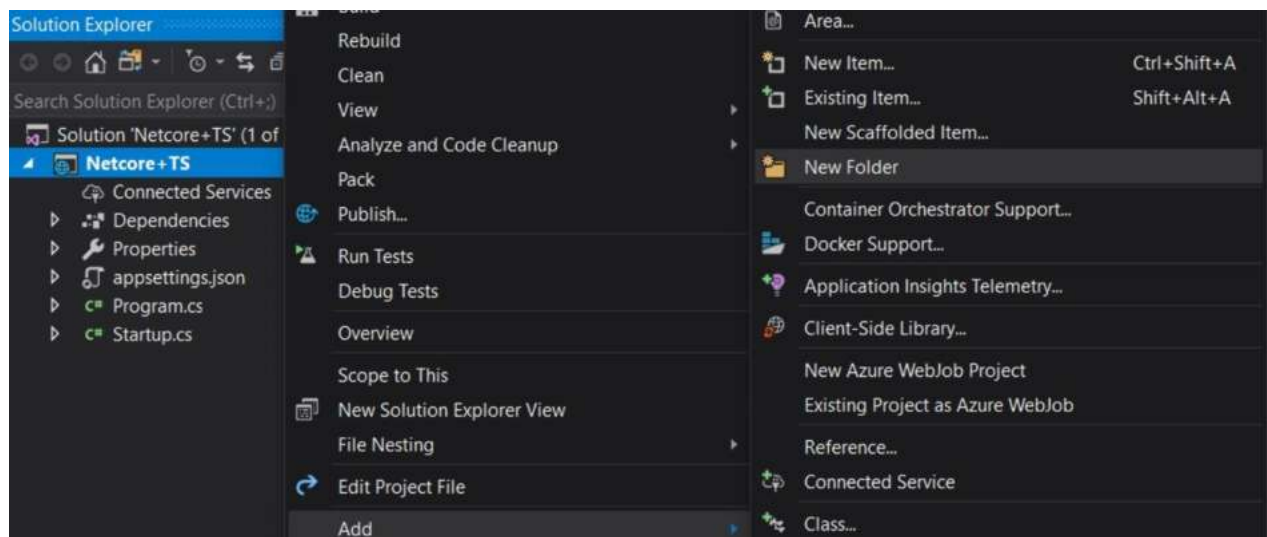
    app.UseDefaultFiles();
}
```

```
    app.UseStaticFiles();  
}
```

You may need to restart VS for the red squiggly lines below `UseDefaultFiles` and `UseStaticFiles` to disappear.

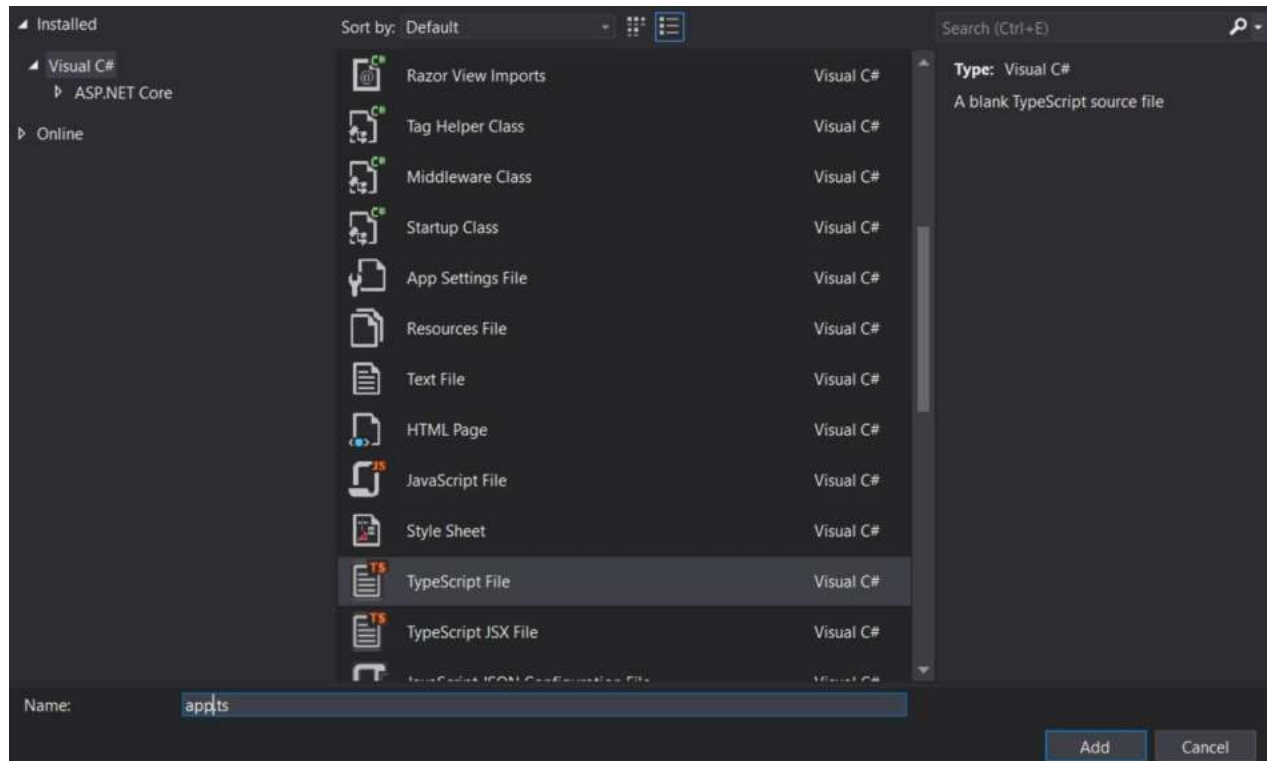
Add TypeScript

Next we will add a new folder and call it `scripts`.



Add TypeScript code

Right click on scripts and click **New Item**. Then choose **TypeScript File** and name the file app.ts



Add example code

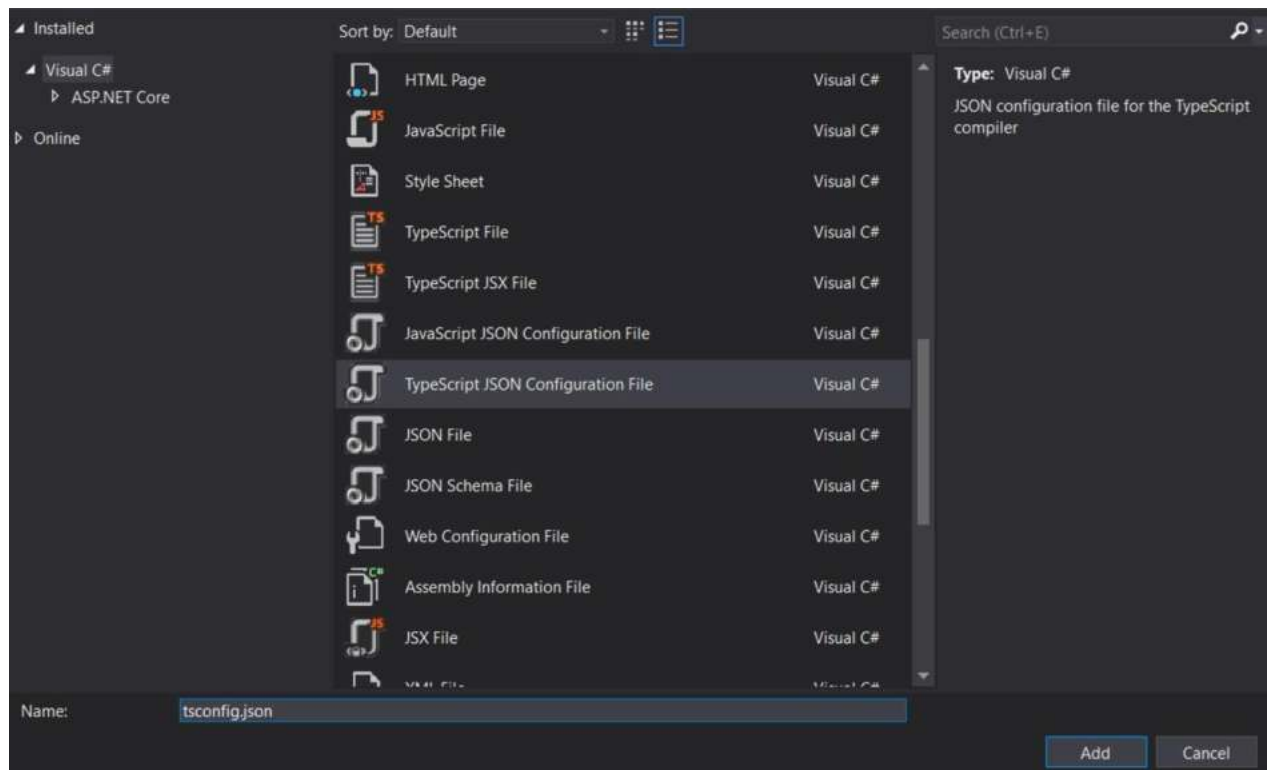
Add the following code to the app.ts file.

```
function sayHello() {  
    const compiler = (document.getElementById("compiler") as HTMLInputElement)  
        .value;  
    const framework = (document.getElementById("framework") as HTMLInputElement)  
        .value;  
    return `Hello from ${compiler} and ${framework}!`;  
}
```

Set up the build

Configure the TypeScript compiler

First we need to tell TypeScript how to build. Right click on scripts and click **New Item**. Then choose **TypeScript Configuration File** and use the default name of tsconfig.json



Replace the contents of the tsconfig.json file with:

```
{
  "compilerOptions": {
    "noEmitOnError": true,
    "noImplicitAny": true,
    "sourceMap": true,
    "target": "es6"
  },
  "files": ["/app.ts"],
  "compileOnSave": true
}
```

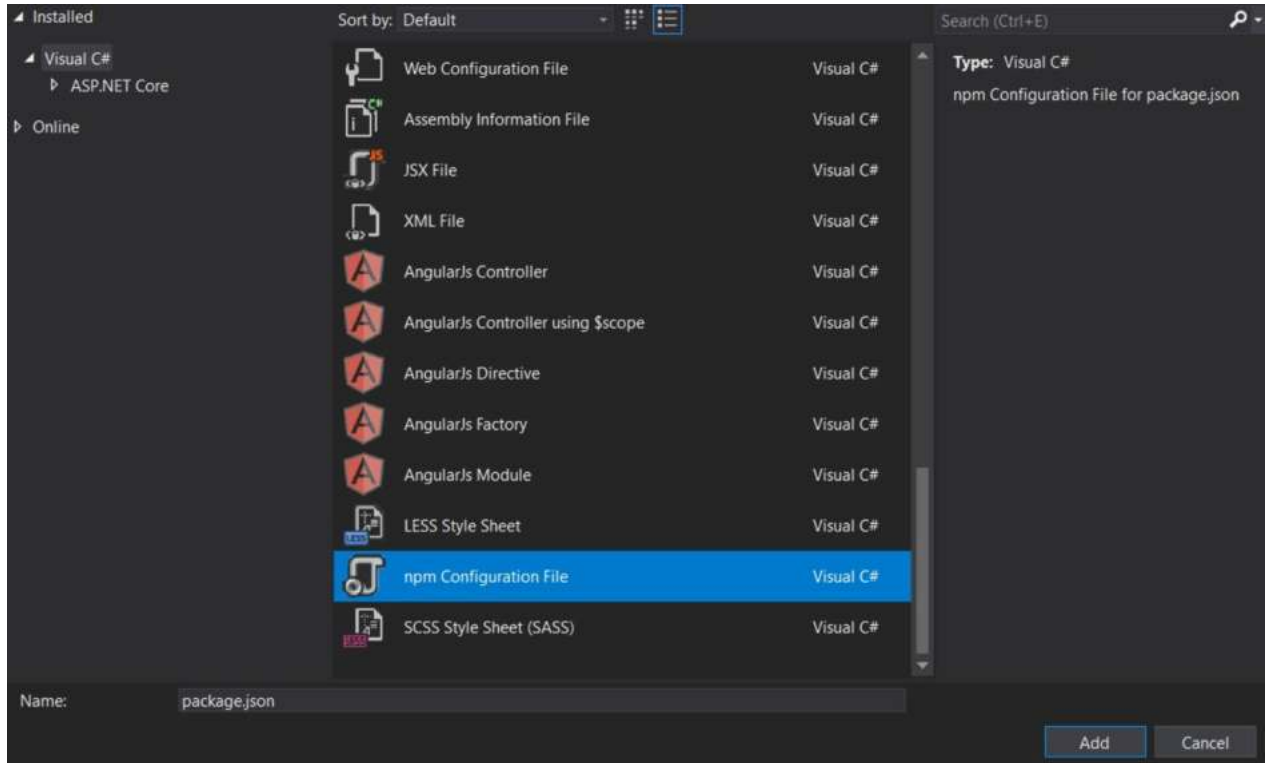
- [noEmitOnError](#) : Do not emit outputs if any errors were reported.
- [noImplicitAny](#) : Raise error on expressions and declarations with an implied any type.
- [sourceMap](#) : Generates corresponding .map file.
- [target](#) : Specify ECMAScript target version.

Note: "ESNext" targets latest supported

[noImplicitAny](#) is good idea whenever you're writing new code — you can make sure that you don't write any untyped code by mistake. "compileOnSave" makes it easy to update your code in a running web app.

Set up NPM

We need to setup NPM so that JavaScript packages can be downloaded. Right click on the project and select **New Item**. Then choose **NPM Configuration File** and use the default name of package.json.

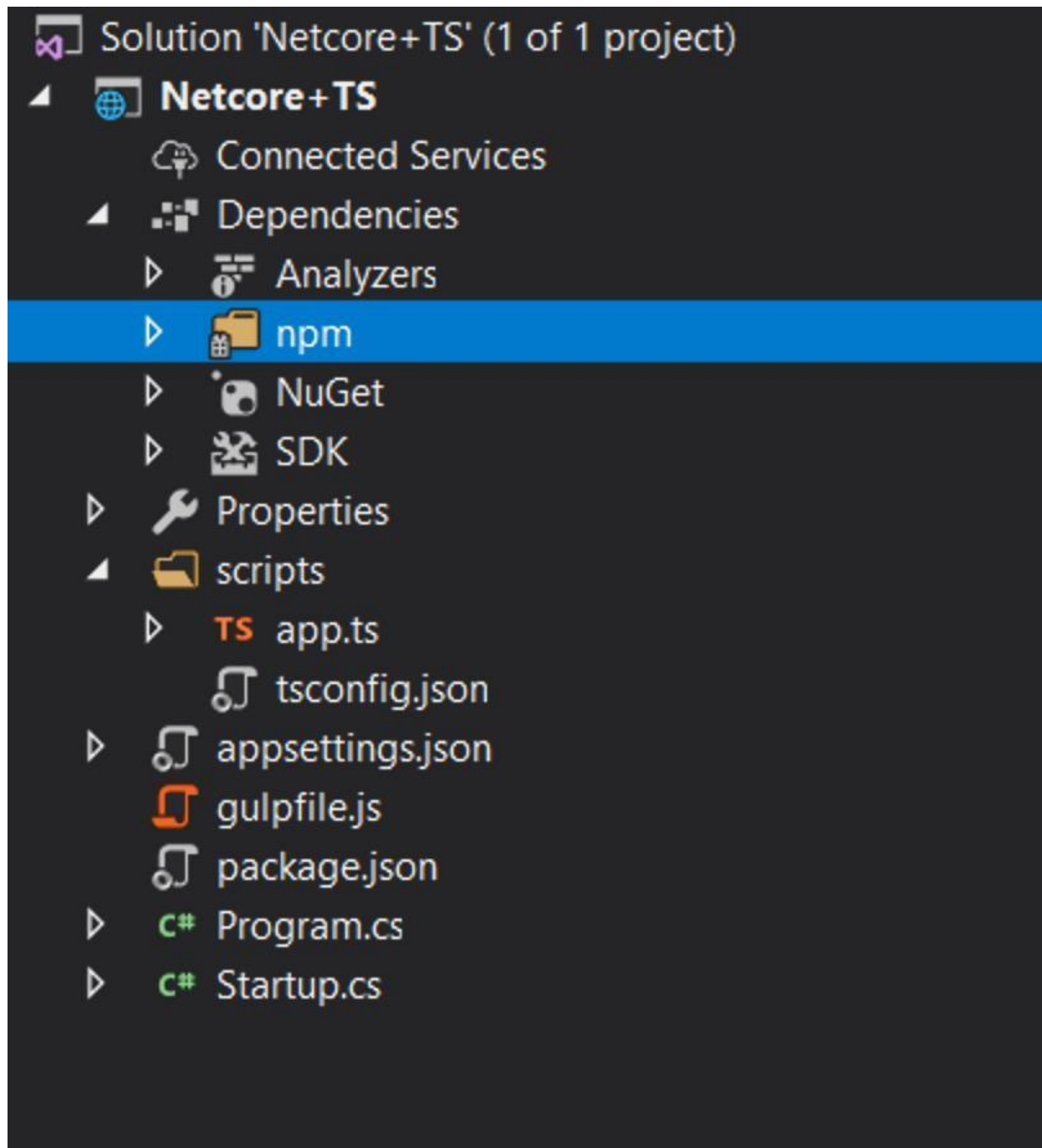


Inside the "devDependencies" section of the package.json file, add *gulp* and *del*

```
"devDependencies": {  
  "gulp": "4.0.2",  
  "del": "5.1.0"  
}
```

Visual Studio should start installing gulp and del as soon as you save the file. If not, right-click package.json and then Restore Packages.

After you should see an npm folder in your solution explorer



Set up gulp

Right click on the project and click **New Item**. Then choose **JavaScript File** and use the name of gulpfile.js

```
/// <binding AfterBuild='default' Clean='clean' />  
/*
```

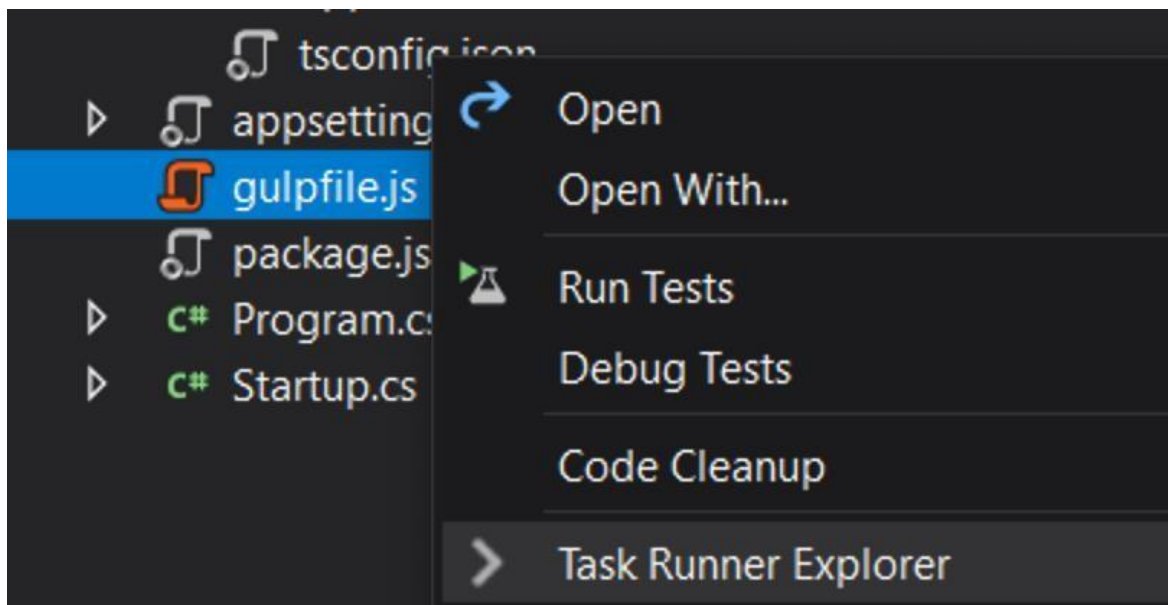
This file is the main entry point for defining Gulp tasks and using Gulp plugins.

Click here to learn more. <http://go.microsoft.com/fwlink/?LinkId=518007>

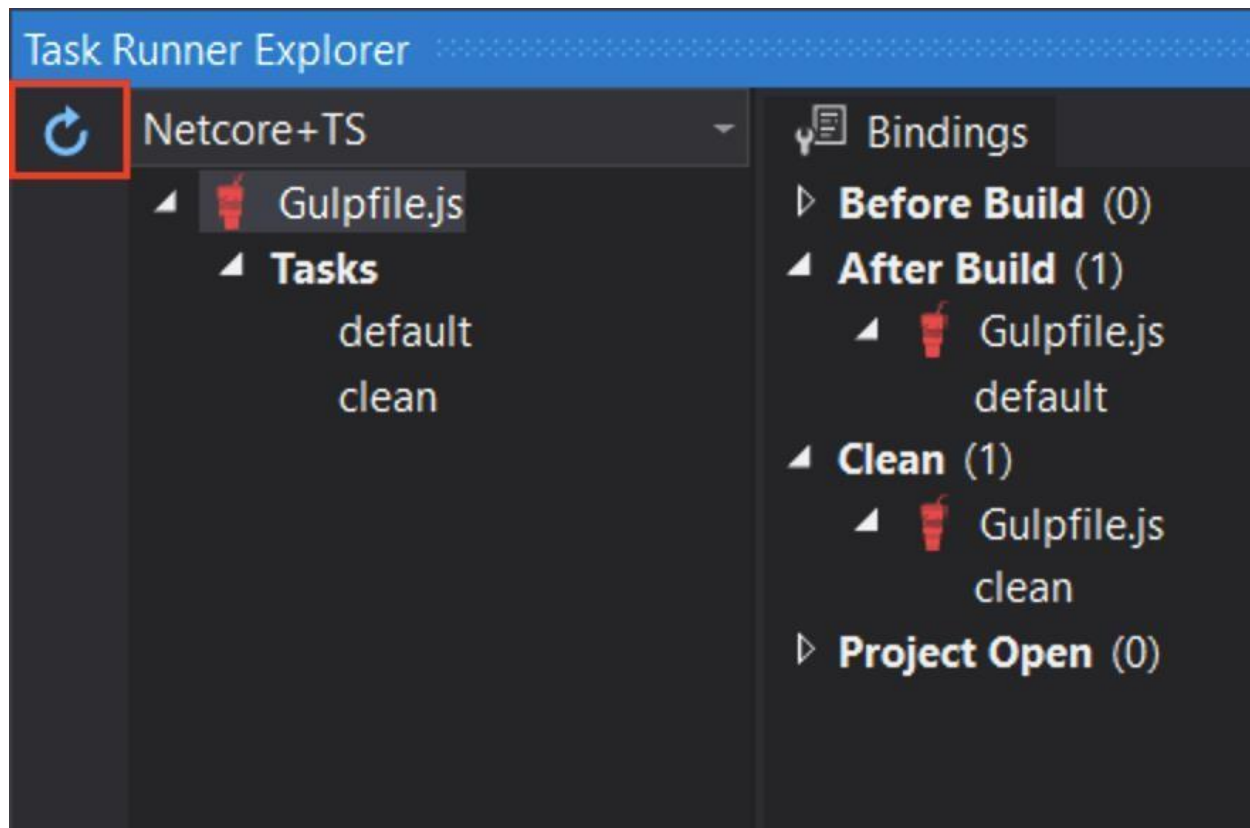
```
*/  
var gulp = require("gulp");  
var del = require("del");  
var paths = {  
  scripts: ["scripts/**/*.js", "scripts/**/*.ts", "scripts/**/*.map"],  
};  
gulp.task("clean", function () {  
  return del(["wwwroot/scripts/**/*.*"]);  
});  
gulp.task("default", function (done) {  
  gulp.src(paths.scripts).pipe(gulp.dest("wwwroot/scripts"));  
  done();  
});
```

The first line tells Visual Studio to run the task 'default' after the build finishes. It will also run the 'clean' task when you ask Visual Studio to clean the build.

Now right-click on gulpfile.js and click Task Runner Explorer.



If 'default' and 'clean' tasks don't show up, refresh the explorer:



Write a HTML page

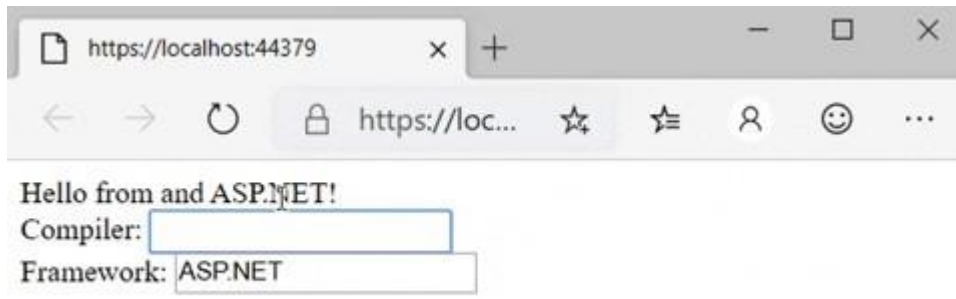
Right click on the wwwroot folder (if you don't see the folder try building the project) and add a New Item named index.html inside. Use the following code for index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <script src="scripts/app.js"></script>
  <title></title>
</head>
<body>
  <div id="message"></div>
  <div>
    Compiler: <input id="compiler" value="TypeScript"
onkeyup="document.getElementById('message').innerText = sayHello()" /><br />
    Framework: <input id="framework" value="ASP.NET"
onkeyup="document.getElementById('message').innerText = sayHello()" />
  </div>
</body>
```

</html>

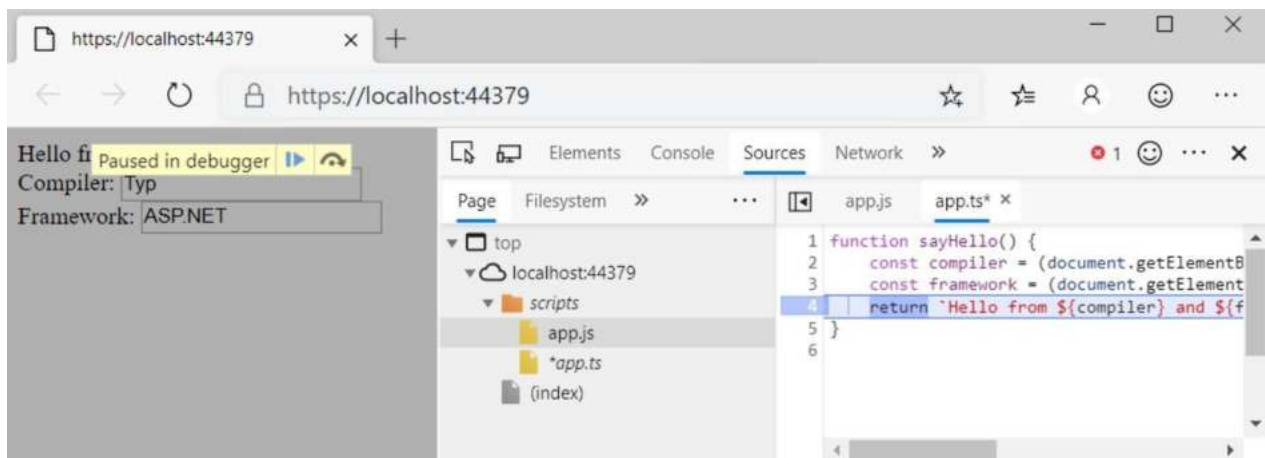
Test

1. Run the project
2. As you type on the boxes you should see the message appear/change!



Debug

1. In Edge, press F12 and click the Debugger tab.
2. Look in the first localhost folder, then scripts/app.ts
3. Put a breakpoint on the line with return.
4. Type in the boxes and confirm that the breakpoint hits in TypeScript code and that inspection works correctly.



Congrats you've built your own .NET Core project with a TypeScript frontend.

Result:

Thus, the webpage has been created successfully using TypeScript and ASP.Net.