

**Godavari Foundation's**  
**Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

**Aim: Demonstrate print "Hello Word" with Angular js. It specifies the Model, View, Controller part of an Angular jsp app.**

Model-View-Controller (MVC) is a software design pattern that separates an application into three interconnected parts: the Model (data and logic), the View (user interface), and the Controller (user input and interaction)

- **Model:**  
Represents the data and business logic of the application.  
Handles interactions with databases and other data sources.  
Is responsible for storing, retrieving, and manipulating data.
- **View:**  
Responsible for displaying the data to the user.  
Is the user interface of the application.  
Can be a web page, a form, or any other type of user interface.
- **Controller:**  
Acts as the intermediary between the Model and the View.  
Receives user input, interacts with the Model to process data, and then updates the View to reflect the changes.  
Handles user interactions and application logic.

**Code:**

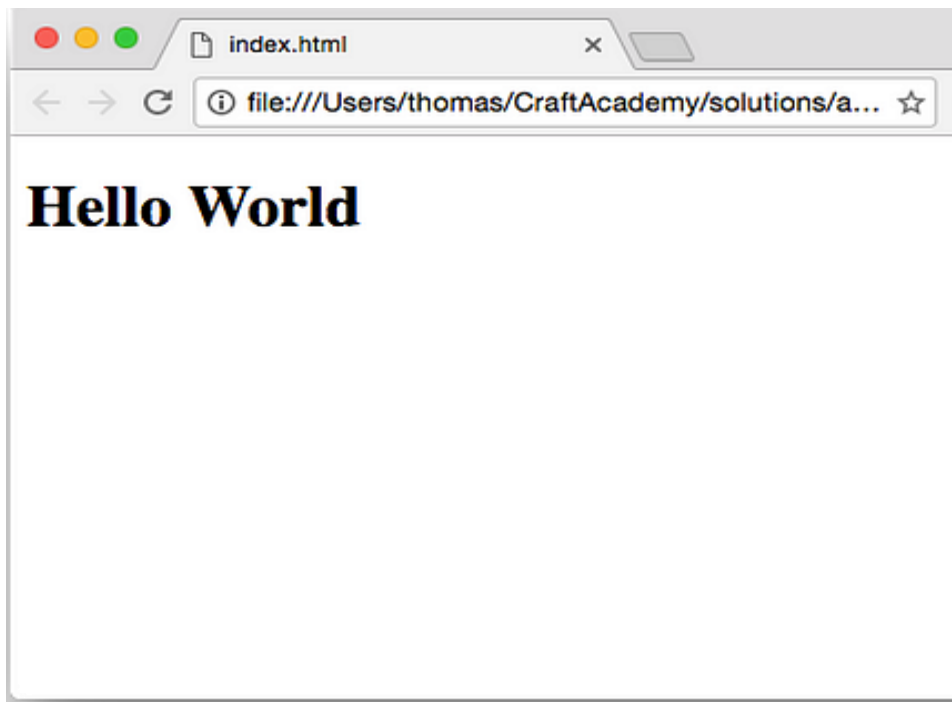
```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AngularJS MVC Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

  <!-- View Part -->
  <div ng-controller="myController">
    <h1>{{ message }}</h1> <!-- Binding Model to View -->
```

```
</div>
<script>
  // Controller Part
  var app = angular.module("myApp", []); // Creating the AngularJS app (Module)

  app.controller("myController", function($scope) {
    $scope.message = "Hello World"; // Defining the Model
  });
</script>
</body>
</html>
```

## Output:



Godavari Foundation's  
**Godavari Institute of Management & Research, Jalgaon**

Name: \_\_\_\_\_ Class: \_\_\_\_\_

Expt.Title: \_\_\_\_\_

Roll No: \_\_\_\_\_ Expt No: \_\_\_\_\_ Date: \_\_\_\_\_

Remarks: \_\_\_\_\_ Sign: \_\_\_\_\_

**Aim: Demonstrate angular js script to implement Built-in Directives.**

AngularJS provides several built-in directives to extend HTML functionality. Some commonly used directives are:

1. **ng-app** → Defines an AngularJS application.
2. **ng-init** → Initializes data in the model.
3. **ng-model** → Binds input fields to variables.
4. **ng-bind** → Binds data dynamically.
5. **ng-repeat** → Loops through arrays.
6. **ng-show/ng-hide** → Show or hide elements conditionally.
7. **ng-if** → Conditionally render elements.
8. **ng-click** → Handles click events.

**Form Directives**

Directive	Description
ng-submit	Handles form submission.
ng-disabled	Disables a form element conditionally.
ng-pattern	Validates input using regular expressions.
ng-change	Executes a function when the value of an input field changes.

**Style and Animation Directives**

Directive	Description
ng-style	Dynamically applies CSS styles.
ng-class	Adds or removes CSS classes dynamically.
ng-animate	Enables animations when elements are added/removed.

## Code:

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Built-in Directives Demo</title>
  <!-- Import AngularJS from CDN -->
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .highlight { background-color: yellow; }
  </style>
</head>
<body ng-controller="MainController">

  <h1>AngularJS Built-in Directives</h1>
  <!-- ng-model and ng-bind -->
  <label>Enter your name: </label>
  <input type="text" ng-model="userName">
  <p>Hello, <span ng-bind="userName"></span>!</p>

  <!-- ng-if -->
  <div ng-if="userName">
    <p class="highlight">Welcome, {{ userName }}! (ng-if is showing this only if userName is not
empty)</p>
  </div>

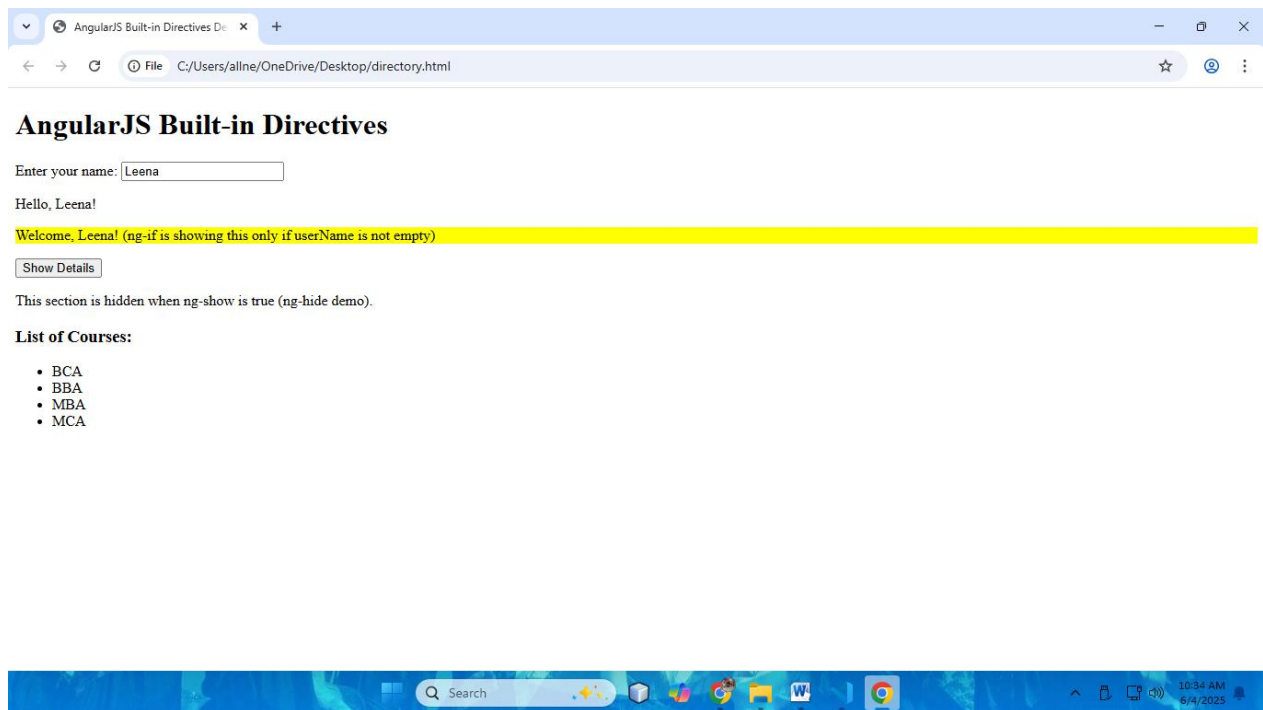
  <!-- ng-show/ng-hide with dynamic button label -->
  <button ng-click="isVisible = !isVisible">
    {{ isVisible ? 'Hide Details' : 'Show Details' }}
  </button>
  <div ng-show="isVisible">
    <p>This is a toggleable section using ng-show.</p>
  </div>
  <div ng-hide="isVisible">
    <p>This section is hidden when ng-show is true (ng-hide demo).</p>
  </div>

  <!-- ng-repeat -->
  <h3>List of Courses:</h3>
  <ul>
    <li ng-repeat="course in courses">
      {{ course }}
    </li>
  </ul>
```

## <!-- AngularJS Application Script -->

```
<script>
var app = angular.module('myApp', []);
app.controller('MainController', function($scope) {
    $scope.userName = "";
    $scope.isVisible = true;
    $scope.courses = ['BCA', 'BBA', 'MBA', 'MCA'];
});
</script>
</body>
</html>
```

## Output:





Godavari Foundation's  
**Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

**Aim: Demonstrate angular js script to add modules and controller.**

**What is Modules and Controllers**

AngularJS modules are used to divide or separate the logic such as controllers, services, application etc. and keep the code clean. AngularJS module helps to link many components, so it is just a group of related components.

AngularJS supports modular approach of programming. AngularJS modules are used to divide or separate the logic such as controllers, services, application etc. and keep the code clean. AngularJS module helps to link many components, so it is just a group of related components.

Generally in most of the applications we have a single entry point (main method) that instantiate and club together different parts of the application. In angularjs applications we don't have that main method instead we have modules that specify how our application will be structured and bootstrapped

**Steps:**

1. Create an **AngularJS Module** using `angular.module()`.
2. Define a **Controller** inside the module using `.controller()`.
3. Bind the controller to the HTML using `ng-controller`.

**Code:**

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Module and Controller Demo</title>
  <!-- Import AngularJS from CDN -->
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f0f4f8;
      margin: 20px;
      padding: 20px;
    }

    .container {
      background-color: #ffffff;
      padding: 20px;
      border: 2px solid #ccc;
      border-radius: 10px;
      max-width: 600px;
      margin: auto;
      box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    }

    h1 {
      color: #1e3a8a;
      background-color: #e0f2fe;
      padding: 10px;
      border-radius: 8px;
      text-align: center;
      box-shadow: 0 2px 5px rgba(0,0,0,0.2);
      border: 1px solid #1e3a8a;
    }

    .section {
      background-color: #f9f9f9;
      padding: 15px;
      margin-top: 20px;
      border: 1px solid #ccc;
      border-radius: 6px;
    }

    h2 {
      color: #155724;
      background-color: #d4edda;
      padding: 8px;
      border-radius: 6px;
      box-shadow: 0 2px 4px rgba(0,0,0,0.1);
      border: 1px solid #155724;
```



```

}

label {
  font-weight: bold;
  color: #333;
}

input[type="text"] {
  padding: 6px;
  margin: 8px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
}

p {
  font-size: 16px;
  color: #444;
  margin: 4px 0;
}

.updated-course {
  color: #d63384;
  font-weight: bold;
}
</style>
</head>
<body ng-controller="MainController">
  <div class="container">
    <h1>Student Information</h1>

    <div class="section">
      <p><strong>Student Name:</strong> {{ student.name }}</p>
      <p><strong>Age:</strong> {{ student.age }}</p>
      <p><strong>Course:</strong> {{ student.course }}</p>
    </div>

    <div class="section">
      <h2>Update Course</h2>
      <label for="courseInput">Enter new course:</label>
      <input type="text" id="courseInput" ng-model="student.course">
      <p><em class="updated-course">Updated Course: {{ student.course }}</em></p>
    </div>
  </div>

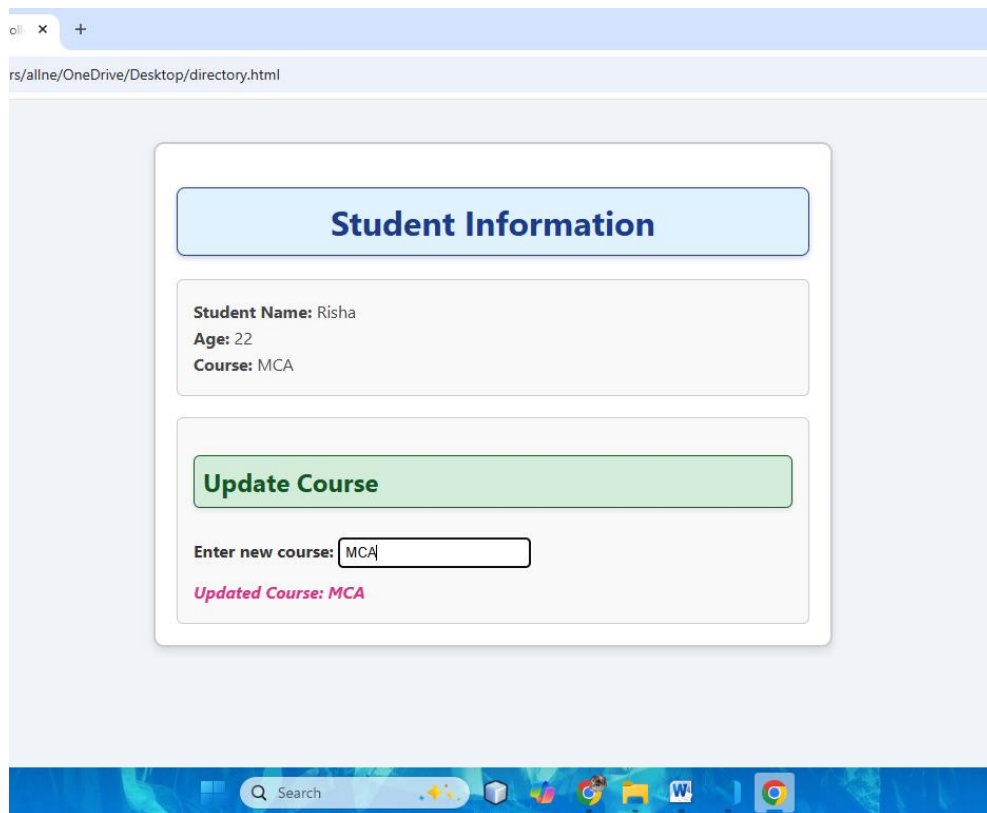
  <script>
    // Create a module named 'myApp'
    var app = angular.module('myApp', []);

    // Create a controller named 'MainController'
    app.controller('MainController', function($scope) {
      // Define student data

```

```
$scope.student = {  
  name: 'Risha',  
  age: 22,  
  course: 'BCA'  
};  
});  
</script>  
</body>  
</html>
```

### Output:



**Godavari Foundation's**  
**Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

**Aim: Demonstrate simple form using angular js script.**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AngularJS Simple Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error { color: red; }
  </style>
</head>
<body ng-app="formApp">

  <div ng-controller="FormController">
    <h2>Simple Form using AngularJS</h2>

    <!-- AngularJS Form -->
    <form name="userForm" ng-submit="submitForm()">
      <label>Name:</label>
      <input type="text" name="name" ng-model="user.name" required>
      <span class="error" ng-show="userForm.name.$touched &&
userForm.name.$invalid">Required</span>
      <br><br>
      <label>Email:</label>
      <input type="email" name="email" ng-model="user.email" required>
      <span class="error" ng-show="userForm.email.$touched && userForm.email.$invalid">Valid
Email Required</span>
      <br><br>
      <label>Gender:</label>
      <select name="gender" ng-model="user.gender" required>
        <option value="">Select</option>
        <option value="Male">Male</option>
        <option value="Female">Female</option>
      </select>
      <span class="error" ng-show="userForm.gender.$touched &&
userForm.gender.$invalid">Required</span>
      <br><br>
```

```

<button type="submit" ng-disabled="userForm.$invalid">Submit</button>
</form>
<!-- Display Submitted Data -->
<div ng-show="submitted">
  <h3>Form Submitted Successfully!</h3>
  <p><strong>Name:</strong> {{ user.name }}</p>
  <p><strong>Email:</strong> {{ user.email }}</p>
  <p><strong>Gender:</strong> {{ user.gender }}</p>
</div>
</div>
<script>
  // Define AngularJS Module
  var app = angular.module("formApp", []);
  // Define Controller
  app.controller("FormController", function($scope) {
    $scope.user = {};
    $scope.submitted = false;
    // Submit Function
    $scope.submitForm = function() {
      if ($scope.userForm.$valid) {
        $scope.submitted = true;
      }
    };
  });
</script>
</body>
</html>

```

### Output:

### Simple Form using AngularJS

Name:

Email:

Gender:

**Form Submitted Successfully!**

**Name:** Leena Patil

**Email:** leena@gmail.com

**Gender:** Female

**Godavari Foundation's**  
**Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

**Aim: Demonstrate the use of JSON in a webpage.**

**What is JSON?**

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent \*

**Why Use JSON**

The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects. Since the format is text only, JSON data can easily be sent between computers, and used by any programming language..

**Steps to Use JSON in a Webpage**

Step 1: Create an HTML file

Step 2: Add Basic HTML Structure

Step 3: Declare a JSON Object in JavaScript

Step 4: Access and Use JSON Data

Step 5: Run the Webpage in a Browser

**Code:**

**data.json file**

```
{
  "users": [
    {"id": 1, "name": "Alice", "email": "alice@example.com"},
    {"id": 2, "name": "Bob", "email": "bob@example.com"},
    {"id": 3, "name": "Charlie", "email": "charlie@example.com"}
  ]
}
```

**Load JSON.html**

```
<!DOCTYPE html>
<html>
<head>
```

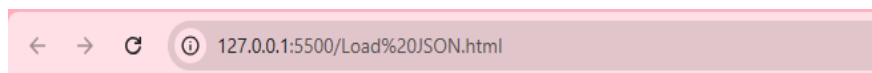
```
<title>Load JSON from File</title>
</head>
<body>
  <h1>Users List Loaded from JSON File</h1>
  <div id="userList">Loading...</div>

  <script>
    // Use fetch API to get JSON file content
    fetch('data.json')
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not OK');
        }
        return response.json(); // Parse JSON data
      })
      .then(data => {
        const userListDiv = document.getElementById('userList');
        userListDiv.innerHTML = ""; // Clear "Loading..."

        const ul = document.createElement('ul');

        data.users.forEach(user => {
          const li = document.createElement('li');
          li.textContent = `ID: ${user.id}, Name: ${user.name}, Email: ${user.email}`;
          ul.appendChild(li);
        });
        userListDiv.appendChild(ul);
      })
      .catch(error => {
        document.getElementById('userList').textContent = 'Error loading JSON data: ' + error;
      });
  </script>
</body>
</html>
```

## Output:



## Users List Loaded from JSON File

- ID: 1, Name: Alice, Email: alice@example.com
- ID: 2, Name: Bob, Email: bob@example.com
- ID: 3, Name: Charlie, Email: charlie@example.com

Godavari Foundation's  
**Godavari Institute of Management & Research, Jalgaon**

Name: \_\_\_\_\_ Class: \_\_\_\_\_

Expt.Title: \_\_\_\_\_

Roll No: \_\_\_\_\_ Expt No: \_\_\_\_\_ Date: \_\_\_\_\_

Remarks: \_\_\_\_\_ Sign: \_\_\_\_\_

Aim: Demonstrate Installation steps of MongoDB and Connect to the database

### 1: Installation of MongoDB

#### Step 1: Download MongoDB

1. Go to: <https://www.mongodb.com/try/download/community>
2. Choose:
  - **Edition:** MongoDB Community Server
  - **Platform:** Your OS (Windows, macOS, Linux)
  - **Package:** MSI (Windows) or TGZ/ZIP (macOS/Linux)
3. Click **Download**.

#### Step 2: Run the Installer:

- On Windows, double-click the .msi file and follow the wizard.
- Accept the license agreement.
- Choose “Complete” setup.

#### Step 3: Install MongoDB as a Service:

- The installer offers to install MongoDB as a Windows service (recommended).
- This allows MongoDB to run automatically in the background.

#### Step 4: Verify MongoDB Is Running

- On Windows, MongoDB runs as a background service.
- Open Command Prompt and check version:

mongod –version

*For Connection of Database we are using **MongoDB compass***

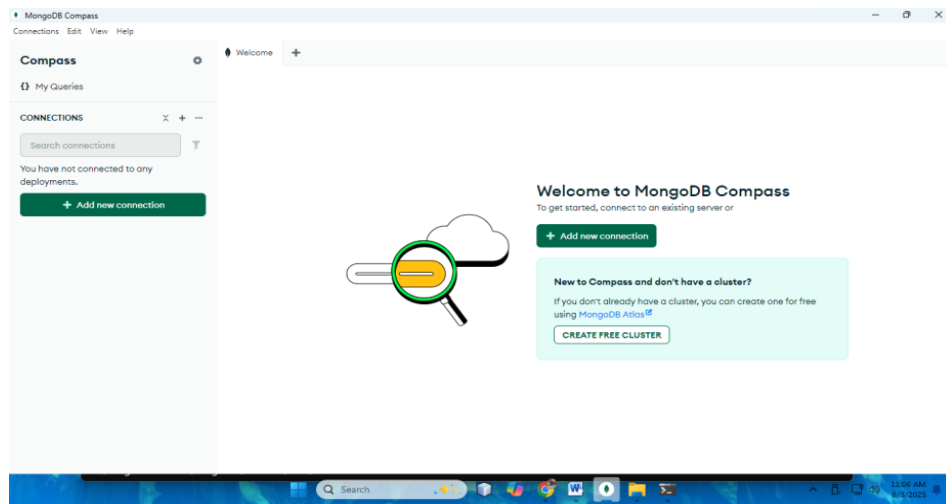
#### Step 1: Download and Install MongoDB Compass

1. Go to the MongoDB Compass download page:  
<https://www.mongodb.com/try/download/compass>
2. Choose your operating system:
  - Windows (64-bit MSI installer)
3. Download the installer file.
4. Install MongoDB Compass:

- **Windows:** Run the .msi installer and follow the prompts.

## Step 2: Open MongoDB Compass

After installation, open the **MongoDB Compass** application.



## Step 3: Connect to Your MongoDB Database

You need to provide connection details to connect: Click on **Add New Connection**

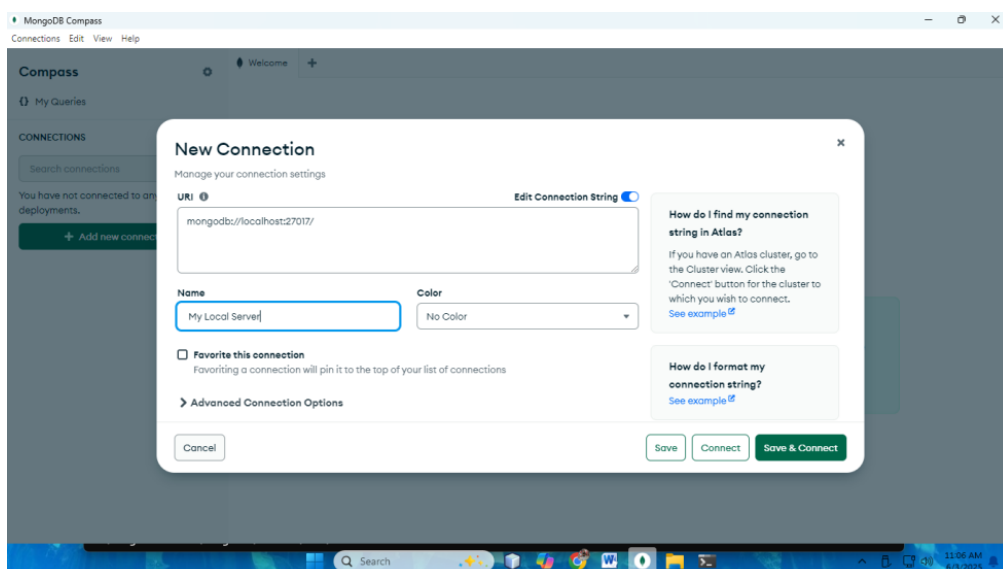
### ➤ For a local MongoDB server (default setup)

Enter your connection string or fill in the connection form:

- Hostname (e.g., My Local Server)
- Port (default is 27017)
- Authentication details (username, password, if required)
- Database name (optional for initial connection)

## Step 4: Click the save & Connect button.

You will see your databases and collections listed **on the left sidebar**.





**Godavari Foundation's  
Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

**Aim: Demonstrate Create a Table in MongoDB**

**Step 1: Launch MongoDB Compass**

- Open MongoDB Compass from your desktop or start menu.
- Connect to your MongoDB server by filling in the **hostname** and **port** (default: localhost:27017).

**Step 2: Create a New Database**

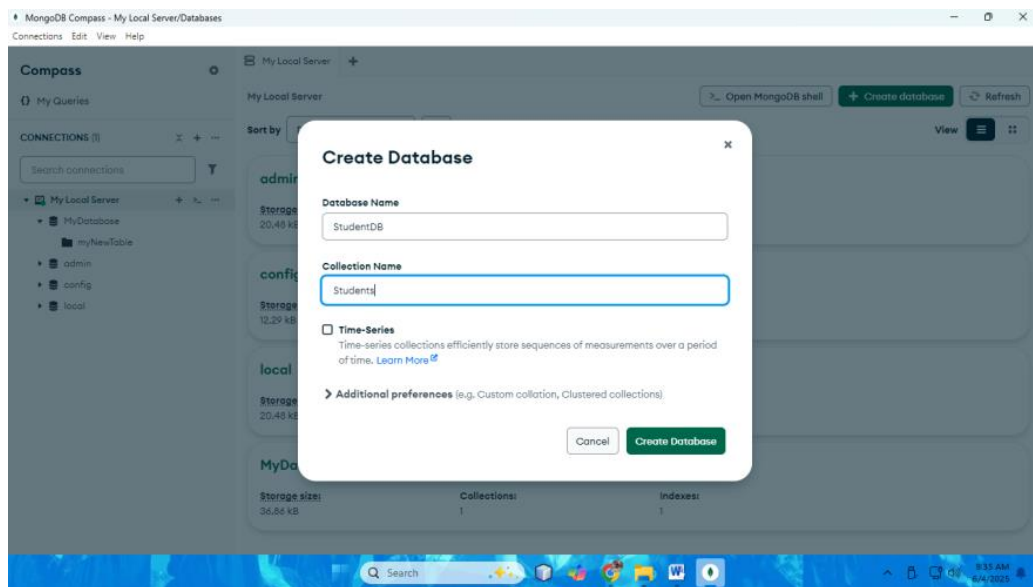
1. Click the “+ Create Database” button (top-left).
2. In the dialog:

**Database Name:** (e.g., studentDB)

**Collection Name:** (e.g., students)

Click “Create Database”.

This will create: Database: **studentDB** and Collection: **students**



**Step 3: View Your Database and Collection**

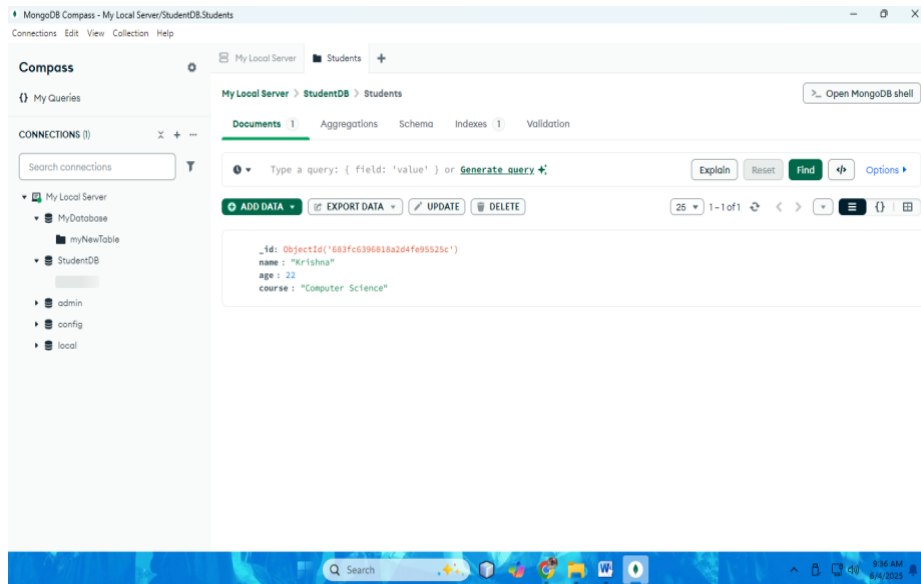
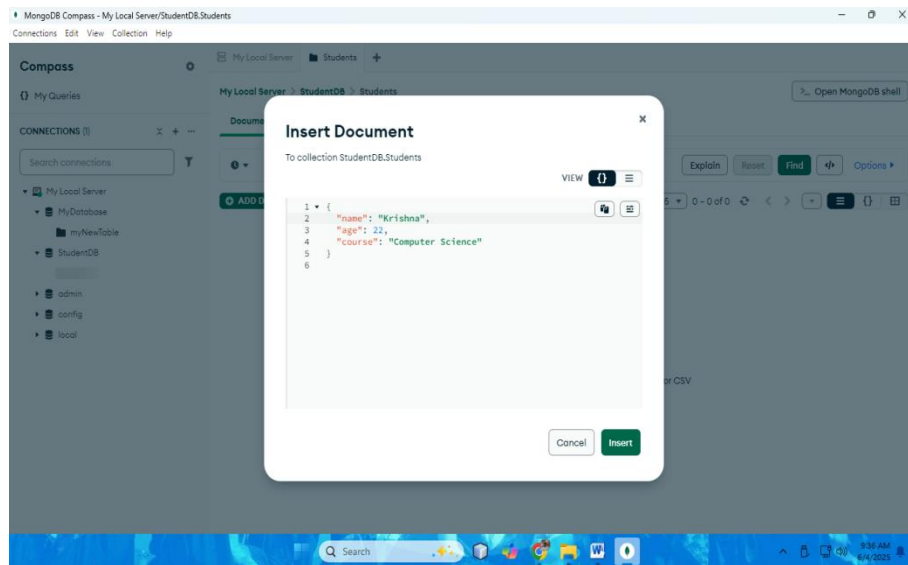
- In the left sidebar, expand the **studentDB** database.
- You should see the **students** collection under it.

## Step 4: Add Documents (Rows) to the Collection

1. Click on the **students** collection.
2. Click the “**Insert Document**” button. You’ll see a JSON editor (or visual editor).
3. Enter a document (for example):

```
{  
  "name": "Krishna",  
  "age": 22,  
  "course": "Computer Science"  
}
```

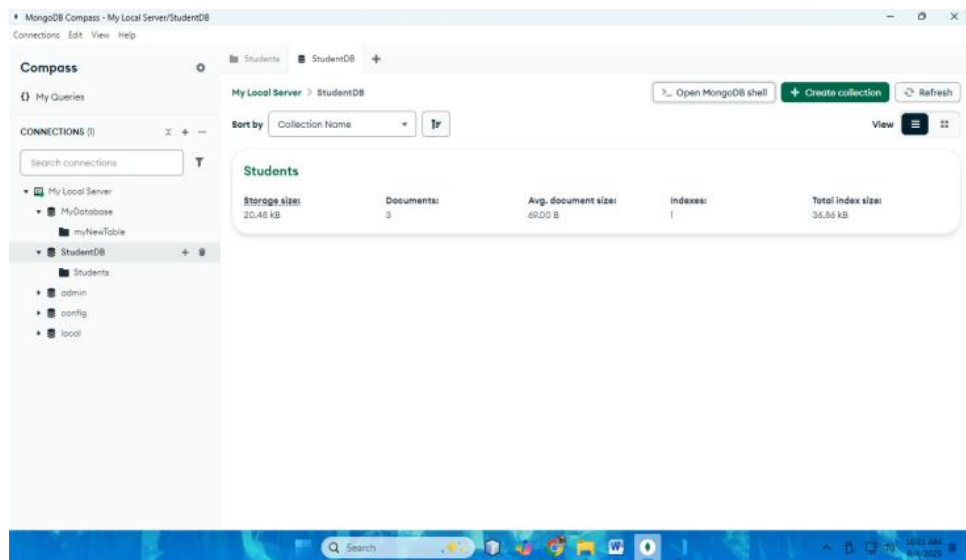
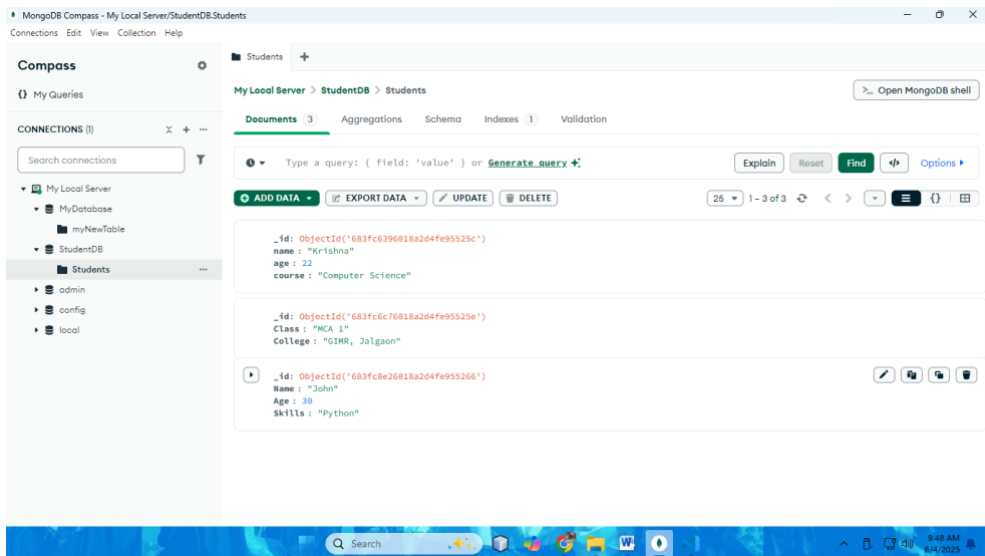
Click “**Insert**”.



You have now added a **document** (like a row) to your **students** collection

## Step 5: View the Documents

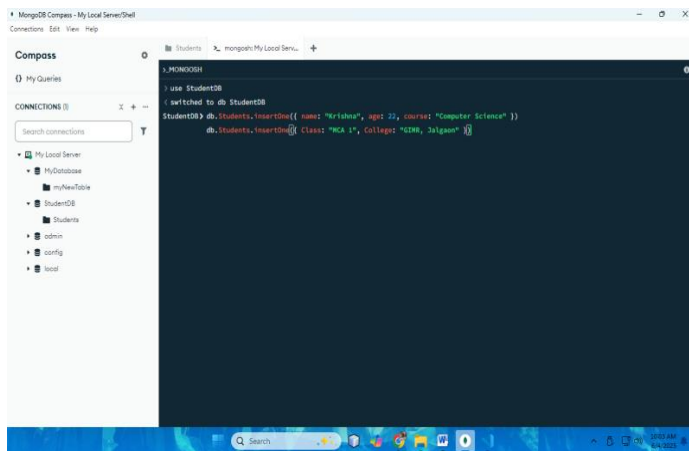
- You'll see your inserted documents listed in the table view.
- You can click on each to edit or delete them.



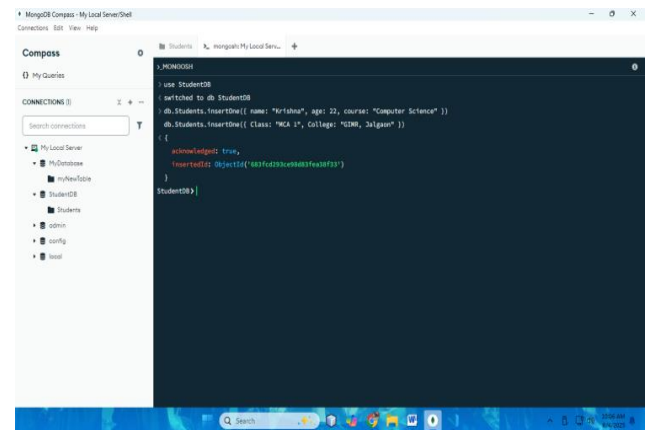
## Creating Table by using MongoDB Shell

**Step 1:** Click on MongoDB Shell at right corner

**Step 2:** Insert data



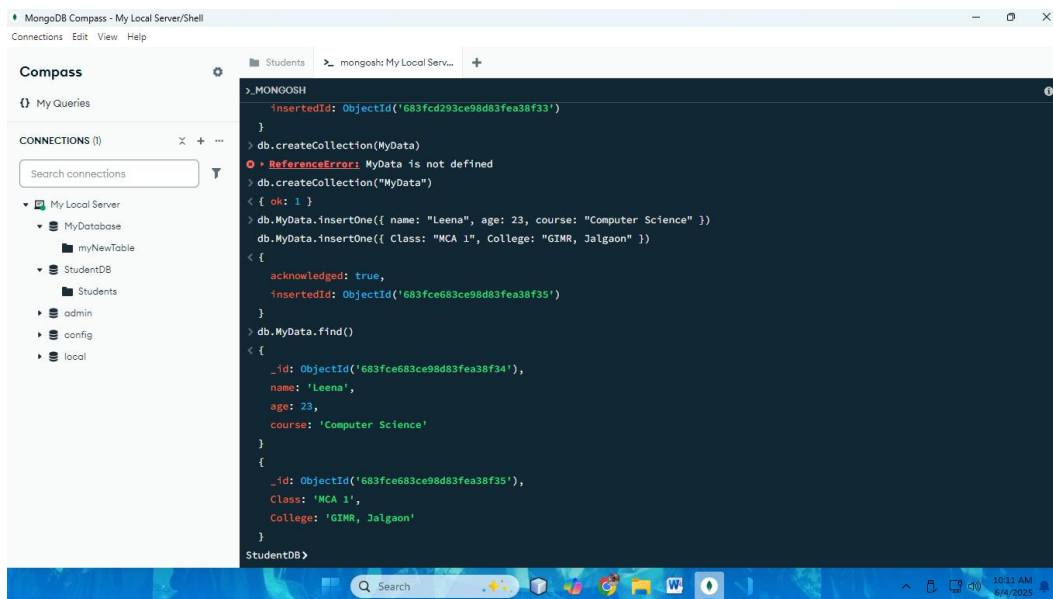
```
.MONGODB
> use StudentDB
switched to db StudentDB
StudentDB> db.Students.insertOne({ name: "Krishna", age: 22, course: "Computer Science" })
StudentDB> db.Students.insertOne({ Class: "MCA 1", College: "GIMR, Jalgaon" })
```



```
.MONGODB
> use StudentDB
switched to db StudentDB
StudentDB> db.Students.insertOne({ name: "Krishna", age: 22, course: "Computer Science" })
StudentDB> db.Students.insertOne({ Class: "MCA 1", College: "GIMR, Jalgaon" })
{
  acknowledged: true,
  insertedIds: {
    '683fcd293ce98d83fea38f33': 1
  }
}
StudentDB>
```

**Step 3:** Create a Collection Explicitly like as above “Students” collection

Here we have created new Database collection(table) name as. “**MyData**” and inserted newly record to it. And after that we displayed all the information which is stored in **MyData** collection by using find() method.



```
.MONGODB
> db.createCollection(MyData)
ReferenceError: MyData is not defined
> db.createCollection("MyData")
{ ok: 1 }
> db.MyData.insertOne({ name: "Leena", age: 23, course: "Computer Science" })
db.MyData.insertOne({ Class: "MCA 1", College: "GIMR, Jalgaon" })
{
  acknowledged: true,
  insertedId: ObjectId('683fce683ce98d83fea38f35')
}
> db.MyData.find()
{
  _id: ObjectId('683fce683ce98d83fea38f34'),
  name: 'Leena',
  age: 23,
  course: 'Computer Science'
}
{
  _id: ObjectId('683fce683ce98d83fea38f35'),
  Class: 'MCA 1',
  College: 'GIMR, Jalgaon'
}
StudentDB>
```

**Godavari Foundation's**  
**Godavari Institute of Management & Research, Jalgaon**

**Name:** \_\_\_\_\_ **Class:** \_\_\_\_\_

**Expt.Title:** \_\_\_\_\_

**Roll No:** \_\_\_\_\_ **Expt No:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Remarks:** \_\_\_\_\_ **Sign:** \_\_\_\_\_

## **Aim: Demonstrate CRUD Operations on MongoDB tables**

**CRUD** stands for the **four basic operations** that almost every application performs on its data:

Letter	Operation	Description
<b>C</b>	<b>Create</b>	Add new data to the database.
<b>R</b>	<b>Read</b>	Retrieve (or query) existing data.
<b>U</b>	<b>Update</b>	Modify existing data.
<b>D</b>	<b>Delete</b>	Remove data from the database.

### **Where do you use CRUD operations?**

CRUD operations are everywhere in software development, especially when working with:

- **Databases** — managing data tables (users, products, orders, etc.)
- **Web apps** — user profiles, posts, comments, etc.
- **Mobile apps** — notes, contacts, messages, etc.
- **APIs (Backend services)** — expose CRUD endpoints for clients to interact with data
- **Content Management Systems** — adding, editing, deleting articles or pages

### **Step 1: Setup**

Open the terminal and start the MongoDB shell:

*mongosh*

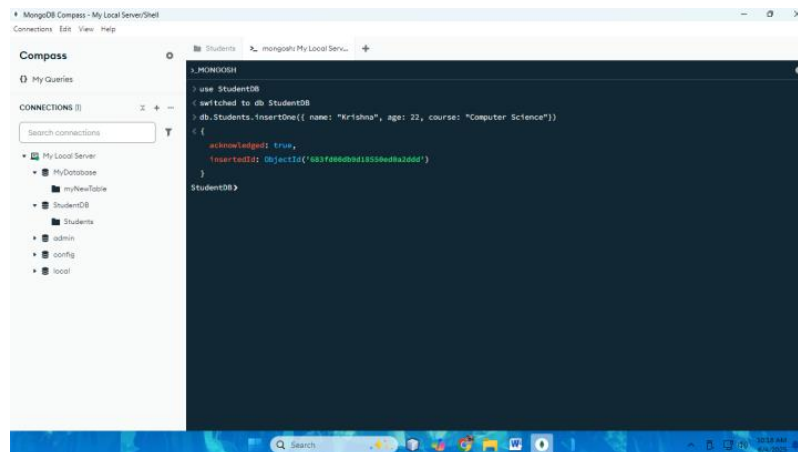
Then switch (or create) a database:

*use StudentDB*

### **1 CREATE**

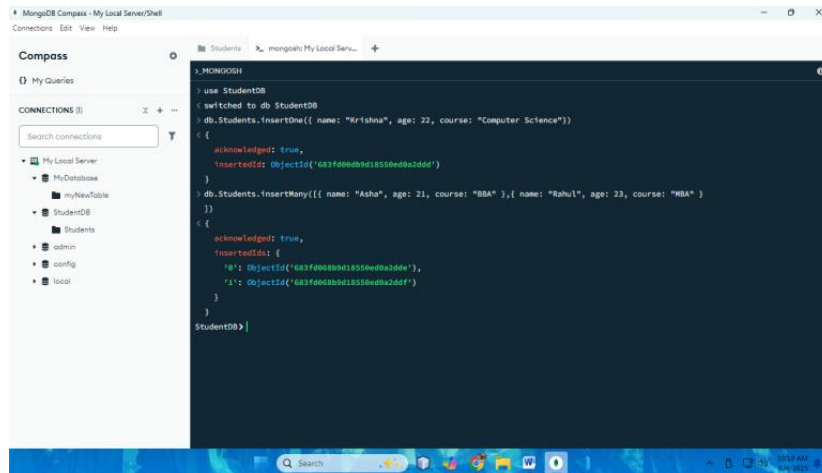
You create a collection by inserting a document:

*db.Students.insertOne({name: "Krishna", age: 22, course: "Computer Science"})*



Or insert multiple documents:

```
db.Students.insertMany([
  { name: "Asha", age: 21, course: "BBA" },
  { name: "Rahul", age: 23, course: "MBA" }
])
```



## 2 READ

Read (query) all documents:

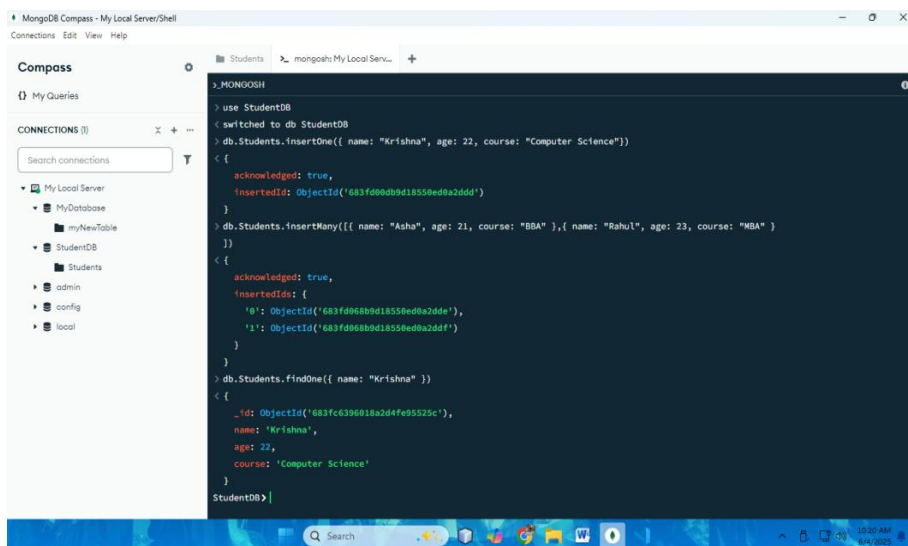
```
db.Students.find()
```

Find documents with a filter:

```
db.Students.find({ age: 22 })
```

Find one document:

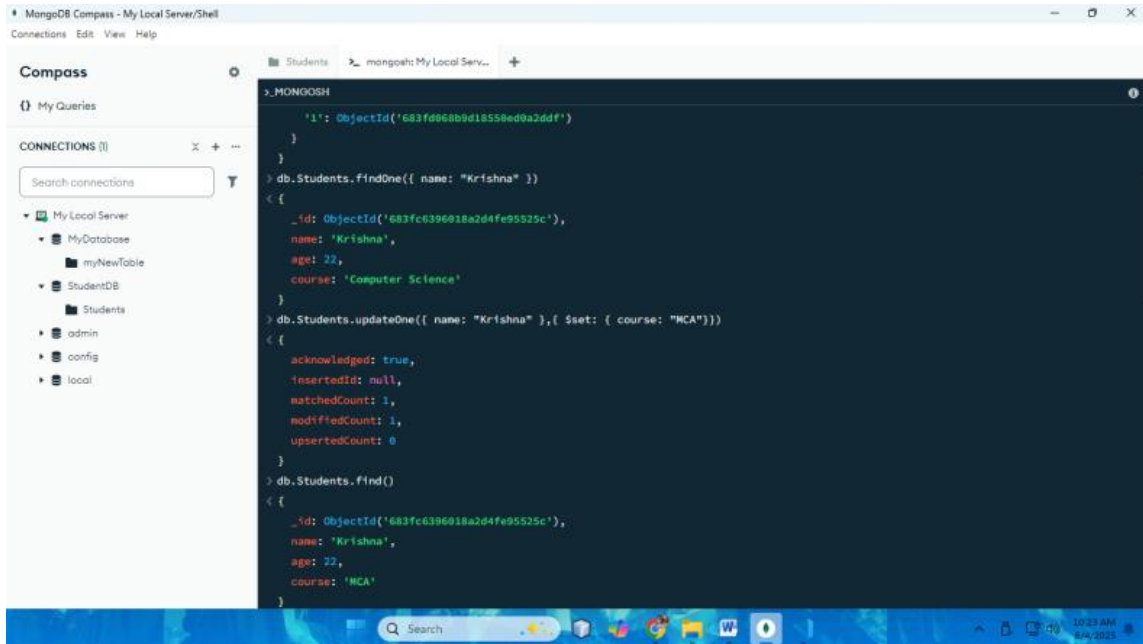
```
db.Students.findOne({ name: "Krishna" })
```



### 3 UPDATE

Update one document:

```
db.Students.updateOne(  
  { name: "Krishna" }, // filter  
  { $set: { course: "MCA" } } // update  
)
```



Here Course of Krishna Which was “**Computer Science**” is updated as “**MCA**”

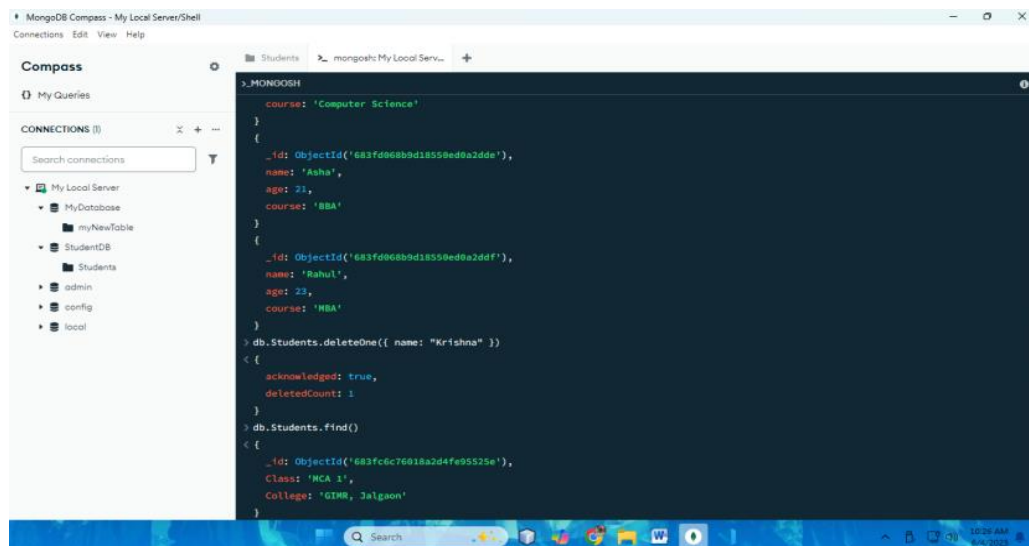
### 4 DELETE

Delete one document:

```
db.Students.deleteOne({ name: "Krishna" })
```

Delete multiple documents:

```
db.Students.deleteMany({ age: { $lt: 23 } })
```



We can see here the row of name “**Krishna**” is deleted.