

C BYREGOWDA INSTITUTE OF TECHNOLOGY

Affiliated to Visvesvaraya Technological University

Approved by AICTE New Delhi & Govt. of Karnataka



Laboratory Manual

Angular JS AND Node JS

V Semester (CBCS Scheme)

Prepared by

Prof. STELLA .J

Asst. Prof, Dept. of CSE

Scrutinized by:

Prof.VASUDEVA .R

Asst.Prof, Dept.of CSE

Department of Computer Science and Engineering

Department of Artificial Intelligence and Machine Learning

C BYREGOWDA INSTITUTE OF TECHNOLOGY

An ISO 9001:2015 certified Institute

Kolar-Srinivasapur Road, Kolar- 563101

2023-24

Course objectives:

- ✓ CLO 1. To learn the basics of Angular JS.
- ✓ CLO 2. To understand the Angular JS Modules.
- ✓ CLO 3. To implement Forms, inputs and Services
- ✓ CLO 4. To implement Directives and Databases
- ✓ CLO 5. To understand basics of Node JS

Course outcomes:

At the end of the course the student will be able to:

- ✓ CO 1. Describe the features of Angular JS.
- ✓ CO 2. Recognize the form validations and controls.
- ✓ CO 3. Implement Directives and Controllers.
- ✓ CO 4. Evaluate and create database for simple application
- ✓ CO 5. Plan and build webservers with node using Node .JS.

Conduction of Practical Examination:

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.

Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).

The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

SEE marks for the practical course is **50 Marks**.

List of Programs

1. Develop Angular JS program that allows user to input their first name and last name and display their full name. **Note:** The default values for first name and last name may be included in the program.
2. Develop an Angular JS application that displays a list of shopping items. Allow users to add and remove items from the list using directives and controllers. **Note:** The default values of items may be included in the program.
3. Develop a simple Angular JS calculator application that can perform basic mathematical operations(addition, subtraction, multiplication, division) based on user input.
4. Write an Angular JS application that can calculate factorial and compute square based on given userinput.
5. Develop AngularJS application that displays a details of students and their CGPA. Allow users to readthe number of students and display the count. **Note:** Student details may be included in the program.
6. Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, anddelete tasks. **Note:** The default values for tasks may be included in the program.
7. Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete)for managing users.
8. Develop AngularJS program to create a login form, with validation for the username and password fields.
9. Create an AngularJS application that displays a list of employees and their salaries. Allow users to searchfor employees by name and salary. **Note:** Employee details may be included in the program.

10. Create AngularJS application that allows users to maintain a collection of items. The application should display the current total number of items, and this count should automatically update as items are added or removed. Users should be able to add items to the collection and remove them as needed.

Note: The default values for items may be included in the program.

11. Create AngularJS application to convert student details to Uppercase using angular filters.

Note: The default details of students may be included in the program.

12. Create an AngularJS application that displays the date by using date filter parameters

Introduction to Angular JS

AngularJS is an open-source web application framework..

AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly. Its data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

General Features

The general features of AngularJS are as follows: □

- ✓ AngularJS is an efficient framework that can create Rich Internet Applications (RIA). □
- ✓ AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way. □
- ✓ Applications written in AngularJS are cross-browser compliant.
- ✓ AngularJS automatically handles JavaScript code suitable for each browser.
- ✓ AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0. Overall,
- ✓ AngularJS is a framework to build large scale, high-performance, and easy-to-maintain web applications.

Core Features

The core features of AngularJS are as follows: □

Data-binding: It is the automatic synchronization of data between model and view components.

Scope: These are objects that refer to the model. They act as a glue between controller and view. □

Controller: These are JavaScript functions bound to a particular scope. □

Services: AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests.

These are singleton objects which are instantiated only once in app. □

Filters: These select a subset of items from an array and returns a new array. □

Directives: Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc. □

Templates: These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials. □

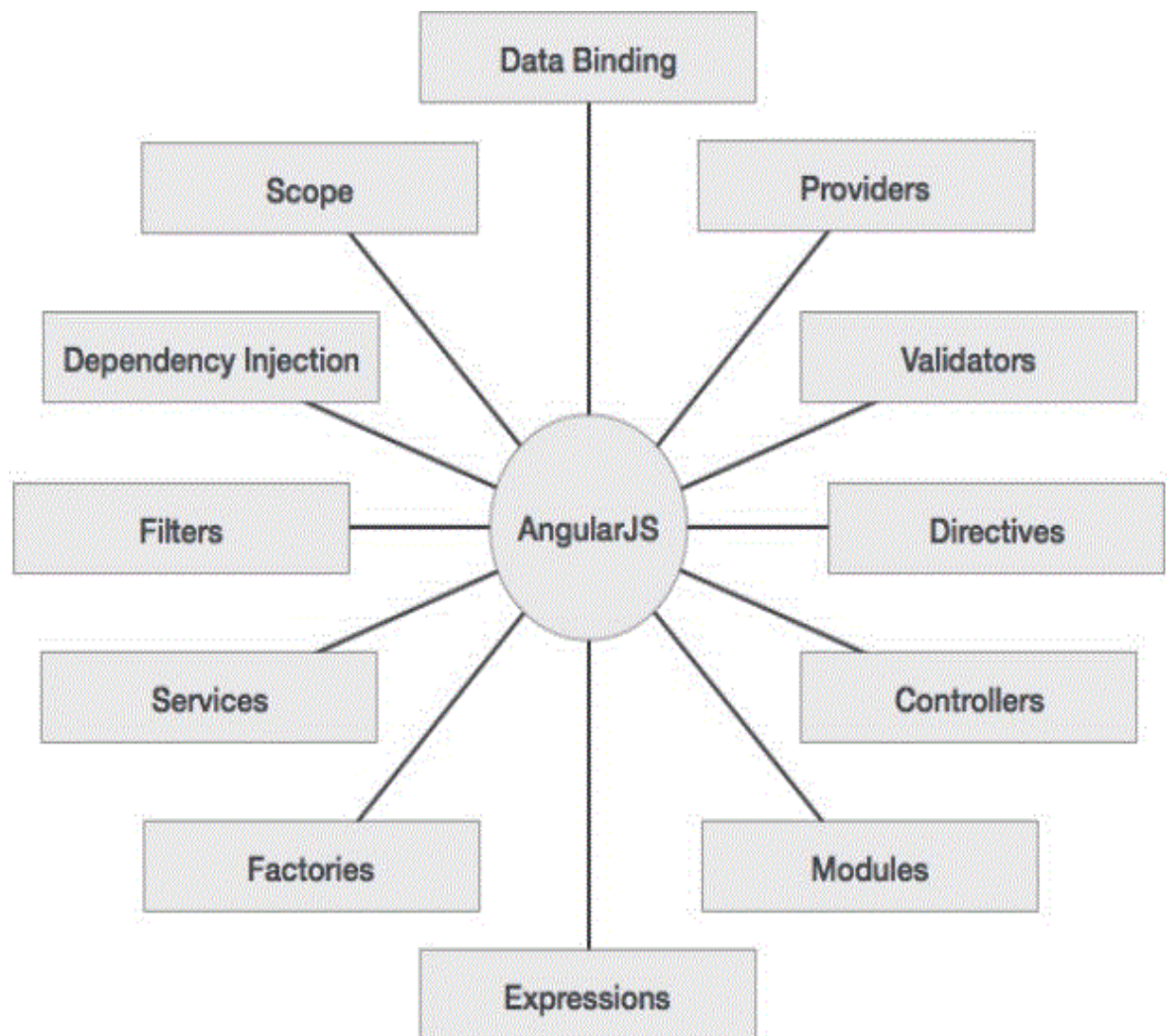
Routing: It is concept of switching views. □

Model View Whatever: MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever. □

Deep Linking: Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state. □

Dependency Injection: AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

The following diagram depicts some important parts of AngularJS



Advantages of AngularJS

The advantages of AngularJS are:

- It provides the capability to create Single Page Application in a very clean and maintainable way. □
- It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience. □
- AngularJS code is unit testable. □
- AngularJS uses dependency injection and make use of separation of concerns. □
- AngularJS provides reusable components. □
- With AngularJS, the developers can achieve more functionality with short code. In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
- On the top of everything, AngularJS applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

AngularJS Directives

The AngularJS framework can be divided into three major parts:

□

ng-app : This directive defines and links an AngularJS application to HTML. □

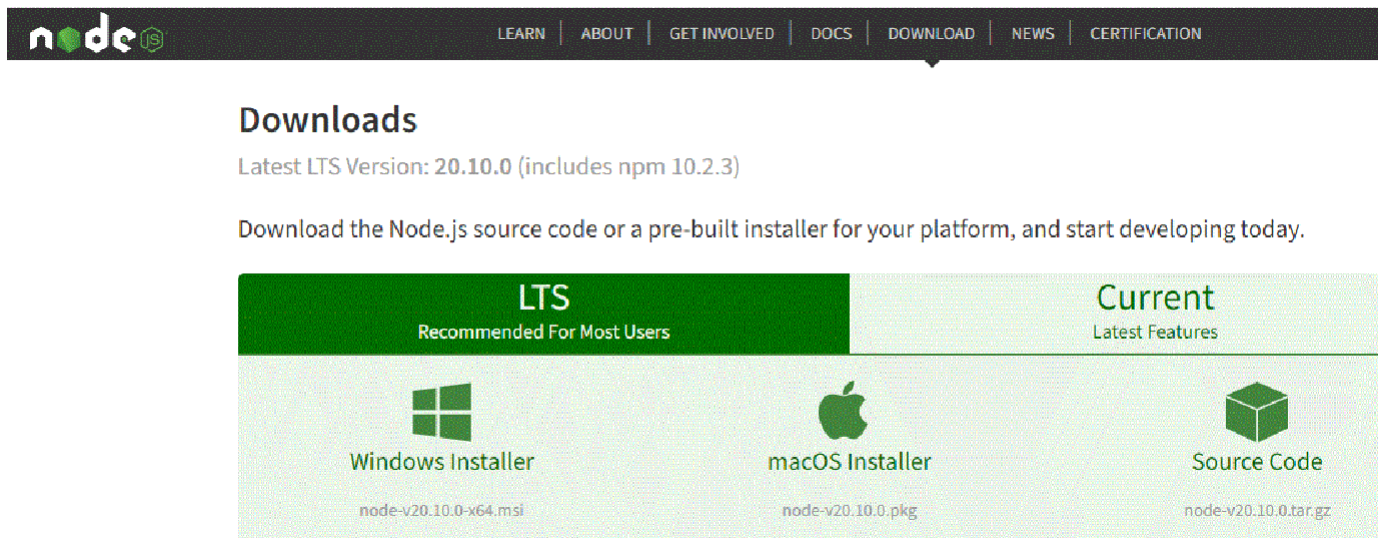
ng-model : This directive binds the values of AngularJS application data to HTML input controls.

ng-bind : This directive binds the AngularJS application data to HTML tags.

ENVIRONMENTAL SETUP

1. Install Node js and npm

<https://nodejs.org/en/>



2. Install Angular-CLI

<https://cli.angular.io/>

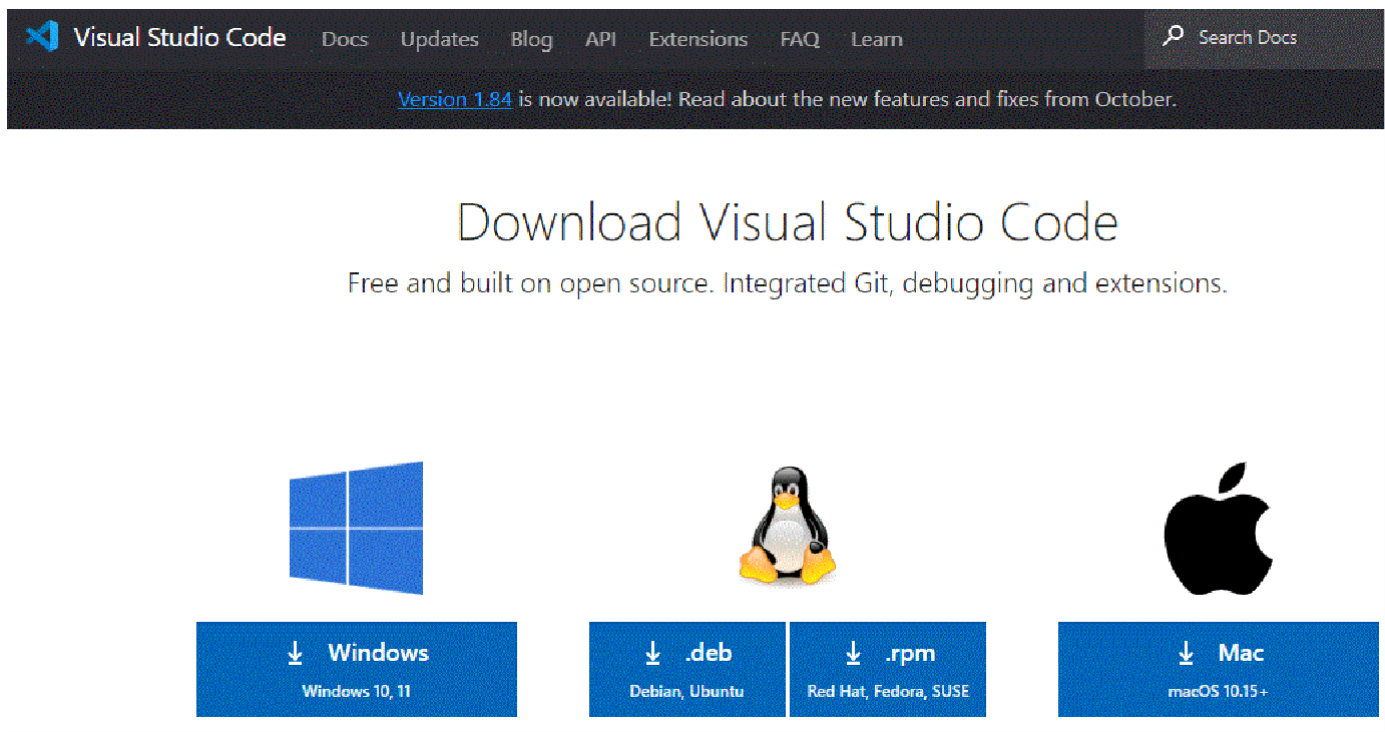
Install the CLI using the npm package manager:

```
npm install -g @angular/cli
```

3. Install visual studio code

4. Command prompt :

```
npm install -g browser-sync
```



Include AngularJS

We include the AngularJS JavaScript file in the HTML page so that we can use it:

```
<head>
  <script
    src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.
    min.js"></script>
</head>
```

Point to AngularJS app

Next, it is required to tell which part of HTML contains the AngularJS app. You can do this by adding the ng-app attribute to the root HTML element of the AngularJS app. You can either add it to the html element or the body element as shown below:

```
<body ng-app="myapp">
</body>
```

View

The view is this part:

```
<div ng-controller="HelloController" >
  <h2>Welcome {{helloTo.title}} to the world of Tutorialspoint!</h2>
</div>
```

Controller

The controller part is:

```
<script>
  angular.module("myapp", [])
  .controller("HelloController", function($scope) {
    $scope.helloTo = {};
    $scope.helloTo.title = "AngularJS";
  });
</script>
```

MVC ARCHITECTURE

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications.

A Model View Controller pattern is made up of the following three parts:

□

Model - It is the lowest level of the pattern responsible for maintaining data.

View - It is responsible for displaying all or a portion of the data to the user.

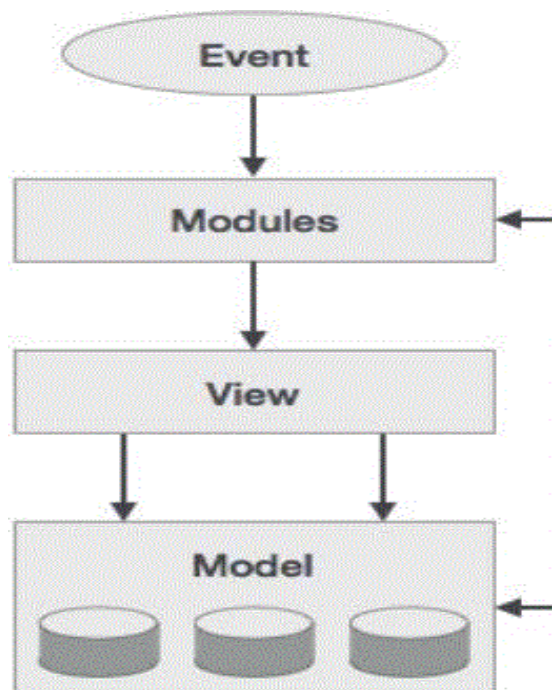
Controller - It is a software Code that controls the interactions between the Model and View.

MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns.

The controller receives all requests for the application and then works with the model to prepare any data needed by the view.

The view then uses the data prepared by the controller to generate a final presentable response.

The MVC abstraction can be graphically represented as follows.



The Model

The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.

The View

A presentation of data in a particular format, triggered by the controller's decision to present the data. they are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.

The Controller

The controller responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

Syntax for ng-app, ng-model, ng-bind

- ng-app
 <html ng-app>
- ng-model
 <input ng-model="name"/>
- ng-bind

Execution by localhost

Save the above code as myfirstexample.html.

Select terminal, new terminal go to command prompt

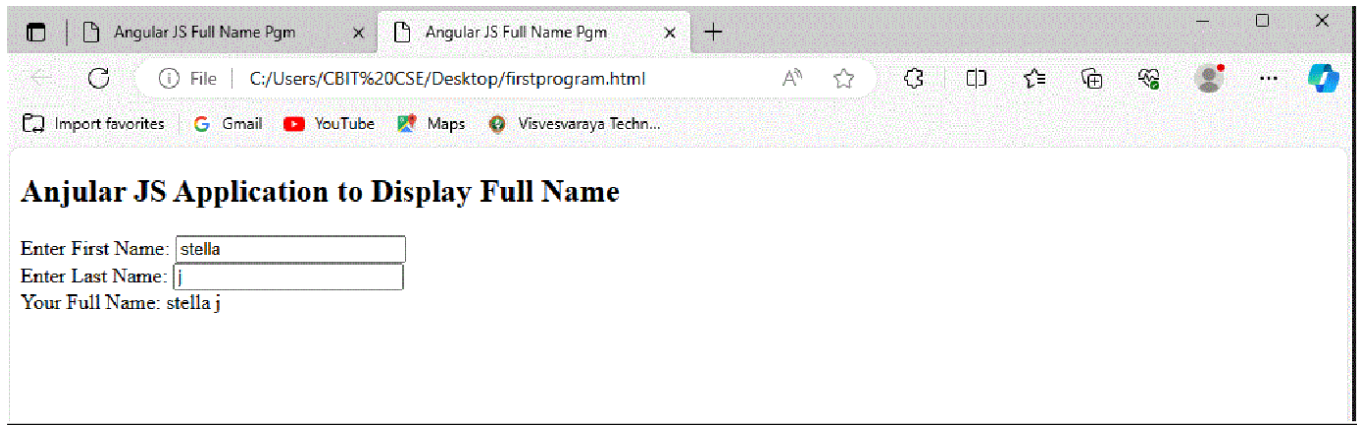
npx browser-sync -w start .

Output is given by local host web browser

1. Develop Angular JS program that allows user to input their first name and last name and display their full name.

```
<!DOCTYPE html>
<html>
<title>
  Angular JS Full Name Pgm
</title>
<head>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
<script>
  var app=angular.module("myApp",[]);
  app.controller("myCntrl",function($scope){
    $scope.firstName="stella"
    $scope.lastName="j"
  });
</script>
</head>
<body ng-app="myApp">
<h2>Angular JS Application to Display Full Name</h2>
<div ng-controller="myCntrl">
  Enter First Name: <input type="text" ng-model="firstName"><br/>
  Enter Last Name: <input type="text" ng-model="lastName"><br/>
  Your Full Name: {{ firstName + " " + lastName }}
</div>
</body>
</html>
```

Output:



2. Develop an Angular JS application that displays a list of shopping items. Allow users to add and remove items from the list using directives and controllers.

Note: The default values of items may be included in the program.

```
<!DOCTYPE html>
<html>
<title>
    Shopping Items Application
</title>
<head>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
<script>
    var app=angular.module("myApp",[]);
    app.controller("myCntrl",function($scope){
        $scope.shoppingItems=['Apple','Mango','Banana','Grapes']
        $scope.addItem=function(){
if($scope.newItem && $scope.shoppingItems.indexOf($scope.newItem)==-1)
{
    $scope.shoppingItems.push($scope.newItem)
    $scope.newItem=""
}
else
{
if($scope.newItem)
    alert("This item is already there in the shopping list")
else
    alert("Please enter an item to add")
}
}

    $scope.removeItem=function(){
        //console.log("function called")
if($scope.shoppingItems.indexOf($scope.selectItem)==-1)
```

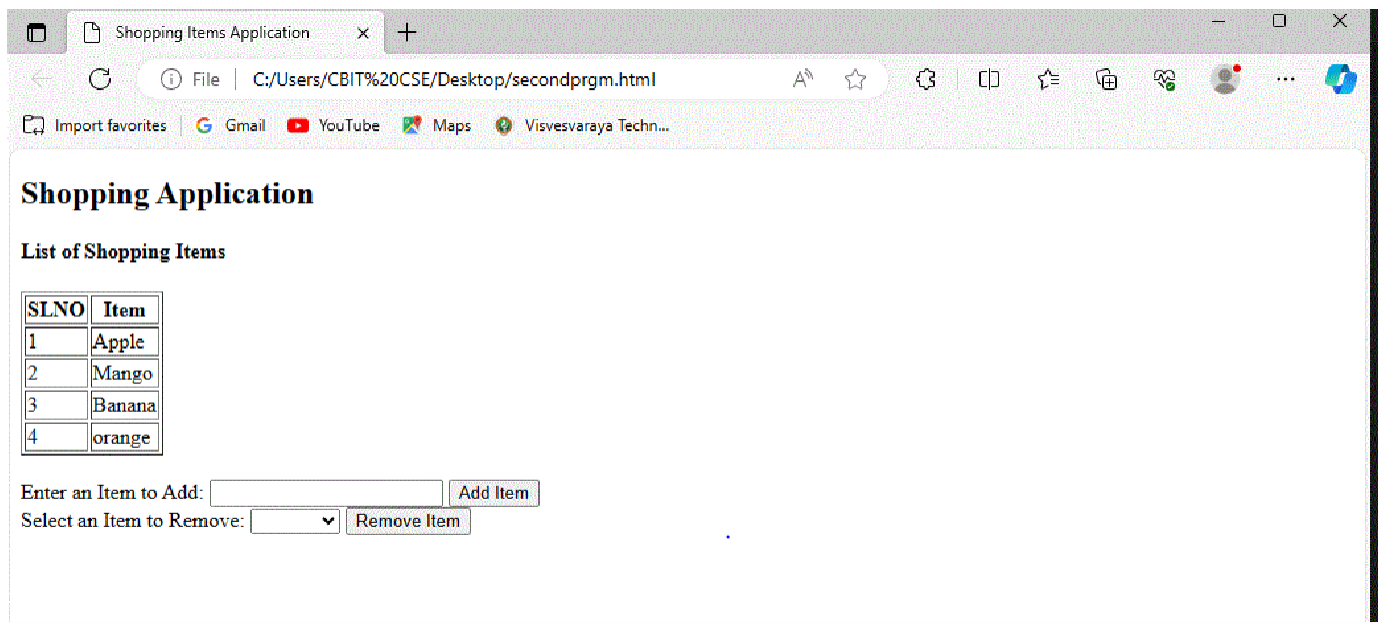


```
{
    alert("Please select an item to remove")
}
else {
    var index=$scope.shoppingItems.indexOf($scope.selectItem)
    $scope.shoppingItems.splice(index,1)
    $scope.selectItem=""
}
}
});
</script>
</head>
<body ng-app="myApp">
<div ng-controller="myCntrl">
    <h2>Shopping Application</h2>
    <h4>List of Shopping Items</h4>
    <table border="1">
<tr>
    <th>SLNO</th>
    <th>Item</th>
</tr>
<tr ng-repeat="items in shoppingItems">
    <td>{{ $index+1 }}</td>
    <td>{{ items }}</td>
</tr>
</table>
<br/>
<div>
Enter an Item to Add: <input type="text" ng-model="newItem">
<button ng-click="addItem()">Add Item</button>
</div>
</div>
```

Select an Item to Remove:

```
<select ng-model="selectItem" ng-options="item for item in shoppingItems"></select>
  <button ng-click="removeItem()">Remove Item</button>
</div>
</div>
</body>
</html>
```

Output:



3. Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input.

```
<!DOCTYPE html>

<html>

<title>
    AJS Simple Calculator
</title>

<head>

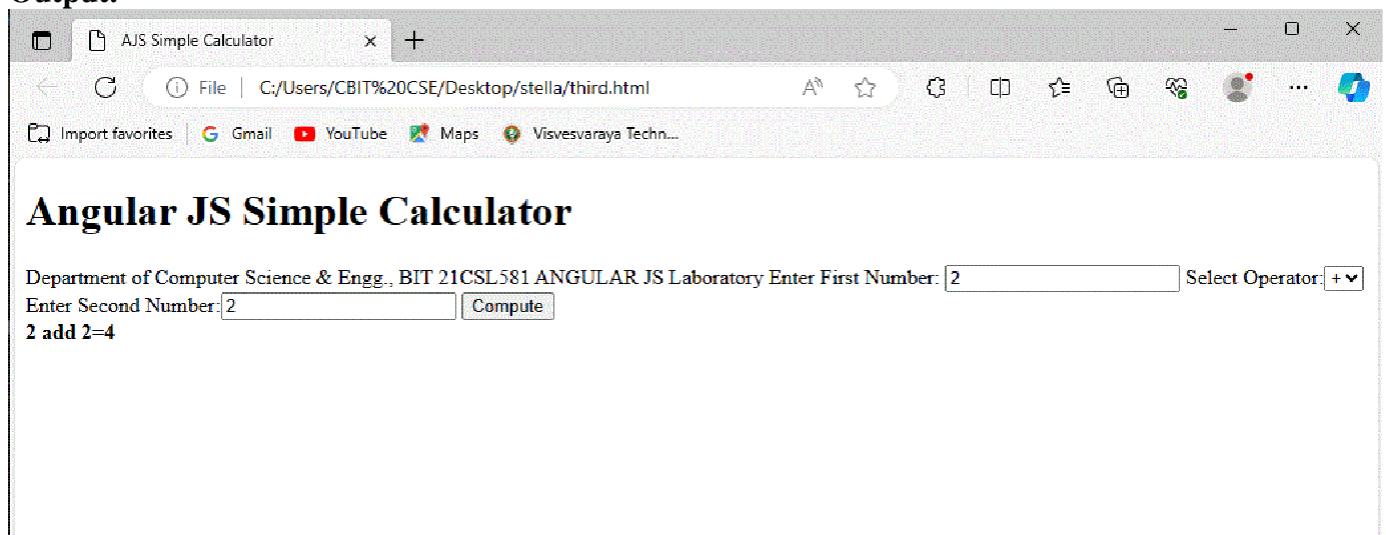
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>

<script>
    var app=angular.module("calcApp",[]);
    app.controller("calcCntrl",function($scope)
    {
        $scope.num1=0
        $scope.num2=0
        $scope.result=0
        $scope.operator="add"
        $scope.compute=function(){
            switch($scope.operator){
                case 'add': $scope.result=$scope.num1 + $scope.num2
                    break
                case 'sub': $scope.result=$scope.num1 - $scope.num2
                    break
                case 'mul': $scope.result=$scope.num1 * $scope.num2
                    break
                case 'div': if($scope.num2==0){
                    alert("Divide by zero error")
                }
            }
            else{
                $scope.result=$scope.num1/$scope.num2
            }
        }
    })
}
```

```

}}}
});
</script>
</head>
<body ng-app="calcApp">
    <h1>Angular JS Simple Calculator</h1>
    <div ng-controller="calcCntrl">
        Enter First Number: <input type="number" ng-model="num1">
        Select Operator:<select ng-model="operator">
            <option value="add">+</option>
            <option value="sub">-</option>
            <option value="mul">*</option>
            <option value="div">/</option>
        </select>
        Enter Second Number:<input type="number" ng-model="num2">
        <button ng-click="compute()">Compute</button>
        <br/>
        <b>{{ num1 + " "+operator+ " "+ num2+ "="+result }}</b>
    </div>
</body>
</html>

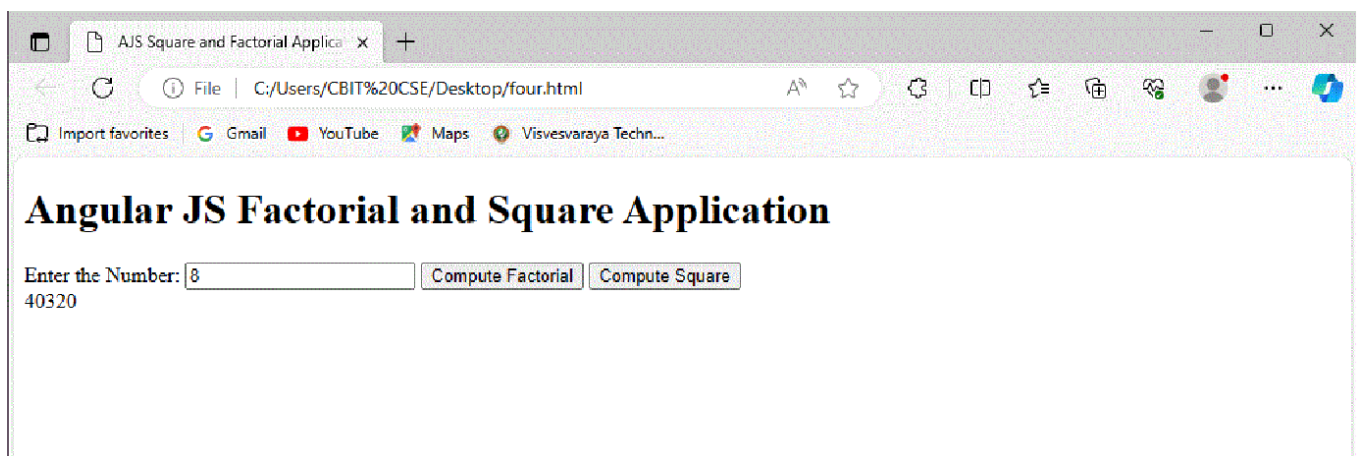
```

Output:

4. Write an Angular JS application that can calculate factorial and compute square based on given user input.

```
<!DOCTYPE html>
<html>
<title>
    AJS Square and Factorial Application
</title>
<head>
<script type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
<script>
    var app=angular.module("mySqFct", []);
    app.controller("mySqFctCntrl", function($scope){
        $scope.num=0
        $scope.result
        $scope.factorial=function()
        {
            if($scope.num==0)
            {
                $scope.result=1
            }
            else{
                $scope.fact=1
                for(var i=$scope.num; i>=1; i--)
                {
                    $scope.fact=$scope.fact*i
                }
                $scope.result=$scope.fact
            }
        }
        $scope.square=function(){
```

```
$scope.result=$scope.num*$scope.num
}
});
</script>
</head>
<body ng-app="mySqFct">
<h1> Angular JS Factorial and Square Application</h1>
    <div ng-controller="mySqFctCntrl">
        Enter the Number: <input type="number" ng-model="num">
        <button ng-click="factorial()">Compute Factorial</button>
        <button ng-click="square()">Compute Square</button>
        <br/>
        {{ result }}
    </div>
</body>
</html>
```

Output:

5. Develop AngularJS application that displays a details of students and their CGPA. Allow users to read the number of students and display the count. Note: Student details may be included in the program.

```
<!DOCTYPE html>
<html>
  <title>Student Details Application</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>
    <script>
      var app=angular.module("studDetailsApp",[]);
      app.controller("studDetailsAppCntrl",function($scope){
        $scope.studData=[]

        $scope.generateData=function()
        {
          $scope.studData=[]
          for(var i=1;i<=$scope.num;i++)
          {
            var stud={
              "SLNO":i,
              "NAME":'Student-'+i,
              "CGPA":(Math.random()*10+1).toFixed(2)
            }
            $scope.studData.push(stud)
          }
        }
      });
    </script>
  </head>
```

```
<body ng-app="studDetailsApp">
  <h1>Student Details Application</h1>
  <div ng-controller="studDetailsAppCntrl">
    Enter the Number of Students to Generate the Data:
    <input type="number" ng-model="num">
    <button ng-click="generateData()">Generate</button>
    <br/>
    <table border="1" ng-show="studData.length>0">
      <tr>
        <th>SLNO</th>
        <th>NAME</th>
        <th>CGPA</th>
      </tr>
      <tr ng-repeat="student in studData">
        <td>{{ student.SLNO }}</td>
        <td>{{ student.NAME }}</td>
        <td>{{ student.CGPA }}</td>
      </tr>
    </table>
    <br/>
    Number of Students={{ studData.length }}
  </div>
</body>
</html>
```


Output:

Student Details Application

Enter the Number of Students to Generate the Data:

SLNO	NAME	CGPA
1	Student-1	2.79
2	Student-2	1.18
3	Student-3	1.44
4	Student-4	4.56
5	Student-5	9.99
6	Student-6	8.56

Number of Students=6

6. Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks. Note: The default values for tasks may be included in the program.

```
<!DOCTYPE html>
<html>
  <title>TO DO Application</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>

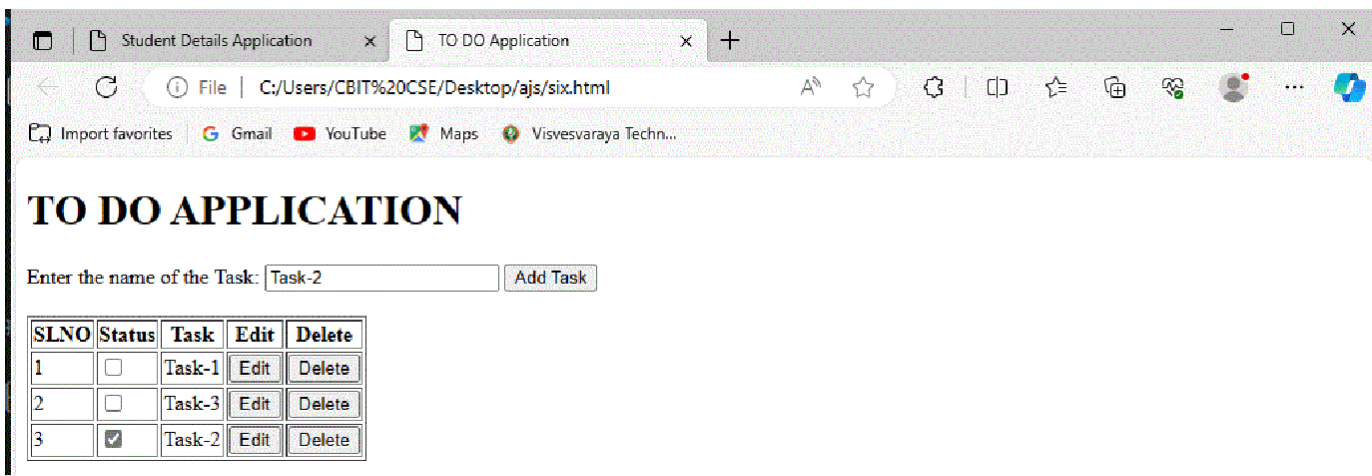
    <script>
      var app=angular.module("toDoApp",[]);
      app.controller("toDoAppCntrl",function($scope){
$scope.tasks=[
  {'TITLE':'Task-1','COMPLETED':true,'EDITING':false},
  {'TITLE':'Task-2','COMPLETED':false,'EDITING':false},
  {'TITLE':'Task-3','COMPLETED':false,'EDITING':false}
  ]
  $scope.addTask=function(){
    if($scope.newTask)
    {
      var t={
        'TITLE':$scope.newTask,
        'COMPLETED':false,
        'EDITING':false
      }
      $scope.tasks.push(t)
    }
    else{
      alert("Please enter the task to add")
    }
  }
}
```

```
$scope.editTask=function(task)
{
    task.EDITING=true
}
$scope.turnOffEditing=function(task){
    task.EDITING=false
}
$scope.deleteTask=function(task)
{
    var index=$scope.tasks.indexOf(task)
    $scope.tasks.splice(index,1)
}
});
</script>
</head>
<body ng-app="ToDoApp">
    <h1>TO DO APPLICATION</h1>
    <div ng-controller="ToDoAppCntrl">
        Enter the name of the Task:
        <input type="text" ng-model="newTask">
        <button ng-click="addTask()">Add Task</button>
        <br/>
        <br/>
        <table border="1">
            <tr>
                <th>SLNO</th>
                <th>Status</th>
                <th>Task</th>
                <th>Edit</th>
                <th>Delete</th>
            </tr>
            <tr ng-repeat="task in tasks">
```

```

        <td>{{ $index+1 }}</td>
        <td>
            <input type="checkbox" ng-model="task.COMPLETED">
        </td>
        <td>
            <span ng-show="!task.EDITING">{{ task.TITLE }}</span>
            <input type="text" ng-show="task.EDITING"
            ng-model="task.TITLE" ng-blur="turnOffEditing(task)">
        </td>
        <td>
            <button ng-click="editTask(task)">Edit</button>
        </td>
        <td>
            <button ng-click="deleteTask(task)">Delete</button>
        </td>
    </tr>
</table>
</div>
</body>
</html>

```

Output:

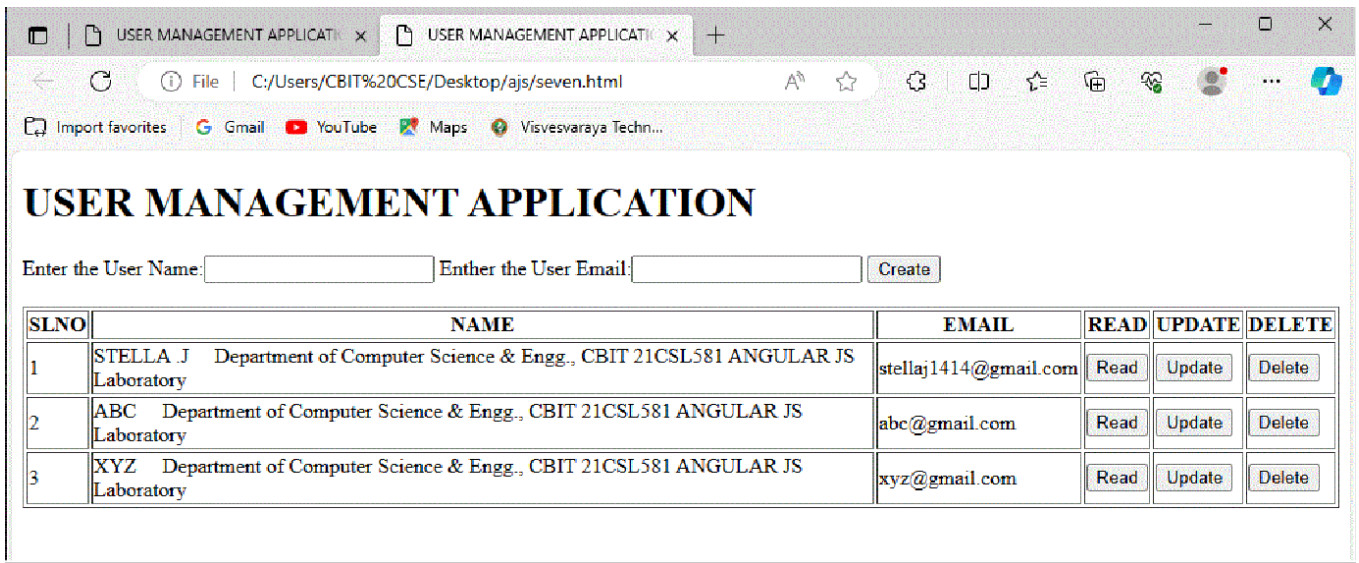
7. Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.

```
<!DOCTYPE html>
<html>
  <title>USER MANAGEMENT APPLICATION</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>
    <script>
      var app=angular.module("userMgmtApp",[]);
      app.controller("userMgmtAppCntrl",function($scope){
        $scope.users=[
          { 'name':"STELLA.J",
            'email':'stellaj1414@gmail.com','editing':false},
          { 'name':'ABC','email':'abc@gmail.com','editing':false},
          { 'name':'XYZ','email':'xyz@gmail.com','editing':false}
        ]
        $scope.createUser=function()
        {
          if($scope.newUserName && $scope.newUserEmail)
          {
            var u={
              'name':$scope.newUserName,
              'email':$scope.newUserEmail,
              'editing':false
            }
            $scope.users.push(u)
            $scope.newUserName=""
            $scope.newUserEmail=""
          }
          else{
            alert("Please provide the user name and email id")
          }
        }
      });
    </script>
  </head>
</html>
```

```
    }  
  }  
  
  $scope.readUser=function(user)  
  {  
    user.editing=true  
  }  
  
  $scope.updateUser=function(user){  
    user.editing=false  
  }  
  
  $scope.deleteUser=function(user)  
  {  
    var yes=confirm("Are you sure you want to delete")  
    if(yes==true)  
    {  
      var index=$scope.users.indexOf(user)  
      $scope.users.splice(index,1)  
    }  
  }  
  
  });  
</script>  
</head>  
<body ng-app="userMgmtApp">  
  
  <h1>USER MANAGEMENT APPLICATION</h1>  
  <div ng-controller="userMgmtAppCntrl">  
  
    Enter the User Name:<input type="text" ng-model="newUserName">  
    Enther the User Email:<input type="text" ng-model="newUserEmail">  
    <button ng-click="createUser()">Create</button>  
  
    <br/>  
    <br/>  
    <table border="1">
```

```
<tr>
  <th>SLNO</th>
  <th>NAME</th>
  <th>EMAIL</th>
  <th>READ</th>
  <th>UPDATE</th>
  <th>DELETE</th>
</tr>
<tr ng-repeat="user in users">
  <td>{{ $index+1 }}</td>
  <td>
    <span
      ng-show="!user.editing">{{ user.name }}</span>
    <input type="text" ng-show="user.editing" ng-model="user.name">
    Department of Computer Science & Engg., CBIT
    21CSL581 ANGULAR JS Laboratory
  </td>
  <td>
    <span ng-show="!user.editing">{{ user.email }}</span>
    <input type="text" ng-show="user.editing" ng-model="user.email">
  </td>
  <td>
    <button ng-click="readUser(user)">Read</button>
  </td>
  <td>
    <button ng-click="updateUser(user)">Update</button>
  </td>
  <td>
    <button ng-click="deleteUser(user)">Delete</button>
  </td>
</tr>
```

```
</table>
</div>
</body>
</html>
```

Output:

USER MANAGEMENT APPLICATION

Enter the User Name: Enter the User Email:

SLNO	NAME	EMAIL	READ	UPDATE	DELETE
1	STELLA J Department of Computer Science & Engg., CBIT 21CSL581 ANGULAR JS Laboratory	stellaj1414@gmail.com	<input type="button" value="Read"/>	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
2	ABC Department of Computer Science & Engg., CBIT 21CSL581 ANGULAR JS Laboratory	abc@gmail.com	<input type="button" value="Read"/>	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
3	XYZ Department of Computer Science & Engg., CBIT 21CSL581 ANGULAR JS Laboratory	xyz@gmail.com	<input type="button" value="Read"/>	<input type="button" value="Update"/>	<input type="button" value="Delete"/>

8. Develop AngularJS program to create a login form, with validation for the username and password fields.

```
<!DOCTYPE html>
<html>
  <title>Angular JS Login Form</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>
    <script>
      var app=angular.module("loginApp",[]);
      app.controller('loginAppCntrl',function($scope){
        $scope.userName=""
        $scope.password=""

        $scope.noAttempts=0
        $scope.login=function(){
          // console.log("Inside login function")
          if($scope.userName=="harish" &&
$scope.password=="12345678")
          {
            alert("Login Successfull")
          }
          else{
            $scope.noAttempts++
            if($scope.noAttempts<=3)
            {
              alert("Incorrect user name/password! Attempt No.
"+$scope.noAttempts)
            }
            else{
              document.getElementById("loginButton").disabled=true
```

```
        }
    }
}

));
</script>
<style>
    .error-message{
        color:green;
        font-size: 20px;
    }
</style>
</head>

<body ng-app="loginApp" ng-controller="loginAppCntrl">

    <h1>Angular JS Login Form</h1>
    <form name="loginForm" ng-submit="submitForm()">

        Enter the User Name:<input type="text" name="userName"
ng-model="userName" ng-minlength="5" ng-maxlength="8" required placeholder="Enter
User Name">
        <span class="error-message"
ng-show="loginForm.userName.$error.required && loginForm.userName.$dirty">User
Name is Required</span>
        <span class="error-message"
ng-show="loginForm.userName.$error.minlength">Minimum Length Must be 5</span>
        <span class="error-message"
ng-show="loginForm.userName.$error.maxlength">Maximum user name length is limited
to 8</span>

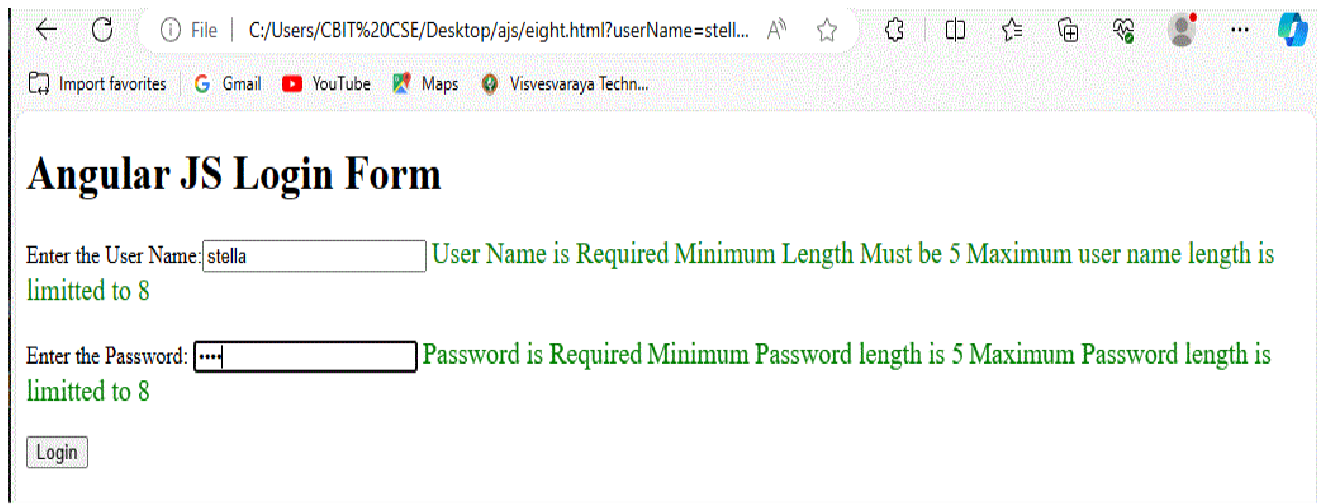
        <br/>
        <br/>

        Enter the Password: <input type="password" name="password"
```

```

ng-model="password" ng-minlength="5" ng-maxlength="8" required placeholder="Enter your
password">
    <span class="error-message" ng-show="loginForm.password.$error.required
    && loginForm.password.$dirty">Password is required</span>
    <span class="error-message"
    ng-show="loginForm.password.$error.minlength">Minimum Password length is 5</span>
    <span class="error-message"
    ng-show="loginForm.password.$error.maxlength">Maximum password length is limited
    to 8</span>
    <br/>
    <br/>
    <button type="submit" ng-disabled="loginForm.$invalid"
    ng-click="login()" id="loginButton">Login</button>
</form>
</body>
</html>

```

Output:

9. Create an AngularJS application that displays a list of employees and their salaries. Allow users to search for employees by name and salary. Note: Employee details may be included in the program.

```
<!DOCTYPE html>

<html>

  <title>Angular JS Filter Employee Search Application</title>

  <head>

    <script type="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">

    </script>

    <script>

var app=angular.module("empSearchApp",[]);

app.controller("empSearchAppCntrl",function($scope){

  $scope.empList=[

    { 'name':'Stella','salary':500000},

    { 'name':'sam','salary':400000},

    { 'name':'Jayden','salary':300000},

    { 'name':'Anjela','salary':400000},

    { 'name':'Priya','salary':500000},

    { 'name':'Manasa','salary':600000}

  ]

  $scope.clearFilters=function()

  {
```

```
$scope.searchName=""
    $scope.searchSalary=""
    }

});

</script>

</head>

<body ng-app="empSearchApp">
    <h1>Employee Search Application</h1>
    <div ng-controller="empSearchAppCntrl">

        Search by Employee Name:<input type="text" ng-
        model="searchName">

        Search by Employee salary:<input type="number"
        ng-model="searchSalary">

        <button ng-click="clearFilters()">Clear Filters</button>

        <br/>

        <h3>List of Employees</h3>

        <table border="1">
```

```
<tr>

    <th>SLNO</th>

    <th>EMP NAME</th>

    <th>SALARY</th>

</tr>

<tr ng-repeat="emp in empList |
filter: { name:searchName,salary:searchSalary}">

    <td>{{ $index+1 }}</td>

    <td>{{ emp.name }}</td>

    <td>{{ emp.salary }}</td>

</tr>

</table>

</div>

</body>

</html>
```

Output:

Employee Search Application

Search by Employee Name: Search by Employee salary:

List of Employees

SLNO	EMP NAME	SALARY
1	Stella	500000
2	sam	400000
3	Jayden	300000
4	Anjela	400000
5	Priya	500000
6	Manasa	600000

10. Create AngularJS application that allows users to maintain a collection of items. The application should display the current total number of items, and this count should automatically update as items are added or removed. Users should be able to add items to the collection and remove them as needed. Note: The default values for items may be included in the program.

```
<!DOCTYPE html>
<html>
  <title>Item Management Application</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>

    <script>
      var app=angular.module("itemMgmtApp",[]);
      app.controller("itemMgmtAppCntrl",function($scope){
        $scope.itemList=['Pen','Pencil','Eraser','Book']

        $scope.addItem=function()
        {
          if($scope.newItem)
          {
            if($scope.itemList.indexOf($scope.newItem)==-1)
            {
              $scope.itemList.push($scope.newItem)
            }
            else{
              alert('This item is already there in the item collection')
            }
          }
          else{
            alert('Please Enter the item to add')
          }
        }
      }
    </script>
  </head>
</html>
```

```
}

$scope.removeItem=function(item)
{
    var yes=confirm("Are you sure you want to delete "+item)
    if(yes==true)
    {
        var index=$scope.itemList.indexOf(item)
        $scope.itemList.splice(index,1)
    }
}

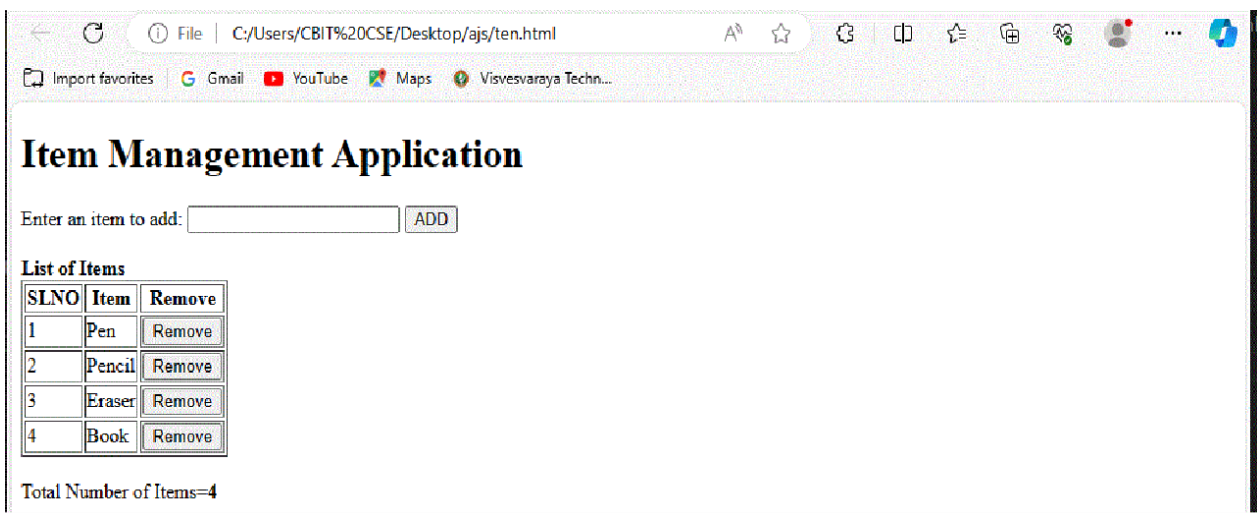
});
</script>
</head>
<body ng-app="itemMgmtApp">
<h1>Item Management Application</h1>

<div ng-controller="itemMgmtAppCntrl">
    Enter an item to add: <input type="text" ng-model="newItem">
    <button ng-click="addItem()">ADD</button>
    <br/><br/>

    <b>List of Items</b>
    <table border="1">
        <tr>
            <th>SLNO</th>
            <th>Item</th>
            <th>Remove</th>
        </tr>
        <tr ng-repeat="item in itemList">
            <td>{{ $index+1 }}</td>
            <td>{{ item }}</td>
            <td><button ng-click=removeItem(item)>Remove</button></td>
        </tr>
    </table>
</div>
```



```
</table>
<br/>
    Total Number of Items=<b>{{ itemList.length }}</b>
</div>
</body>
</html>
```

Output:

11. Create AngularJS application to convert student details to Uppercase using angular filters.

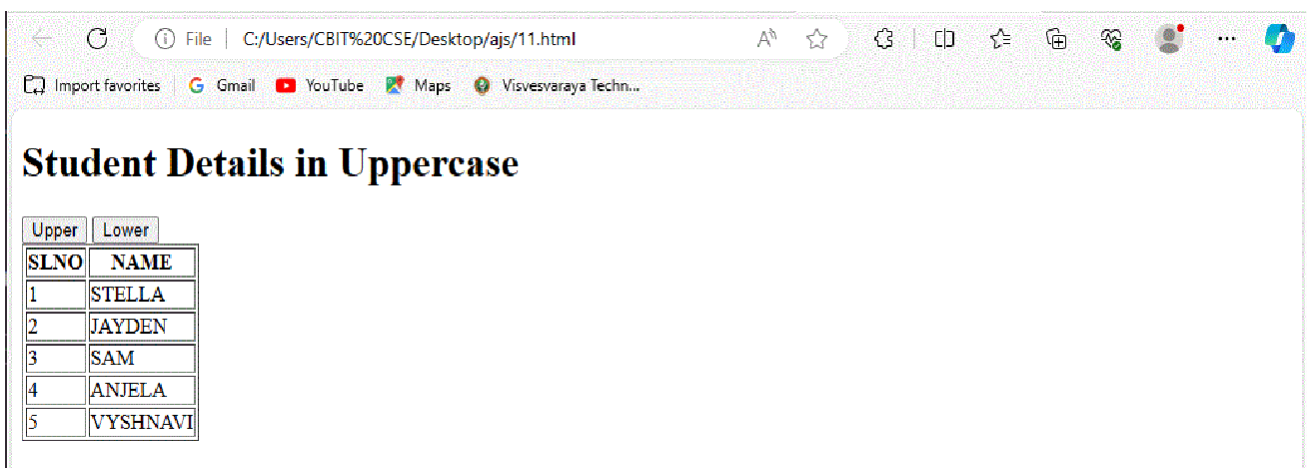
Note: The default details of students may be included in the program.

```
<!DOCTYPE html>
<html>
  <title>Student Details in Uppercase</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>

    <script>
      var app=angular.module("studDetailsUpperApp",[]);
      app.controller("studDetailsUpperAppCntrl",function($scope){
        $scope.studDetails=['stella','jayden','sam','anjela','vyshnavi']
        $scope.upper=true
        $scope.lower=false

        $scope.Lower=function()
        {
          //console.log('called')
          $scope.upper=false
          $scope.lower=true
        }
        $scope.Upper=function()
        {
          $scope.upper=true
          $scope.lower=false
        }
      });
    </script>
  </head>
```

```
<body ng-app="studDetailsUpperApp">
  <h1>Student Details in Uppercase</h1>
  <div ng-controller="studDetailsUpperAppCntrl">
    <button ng-click="Upper()">Upper</button>
    <button ng-click="Lower()">Lower</button>
    <table border="1">
      <tr>
        <th>SLNO</th>
        <th>NAME</th>
      </tr>
      <tr ng-repeat="student in studDetails">
        <td>{{ $index+1 }}</td>
        <td ng-show="upper">{{ student|uppercase }}</td>
        <td ng-show="lower">{{ student|lowercase }}</td>
      </tr>
    </table>
  </div>
</body>
</html>
```

Output:

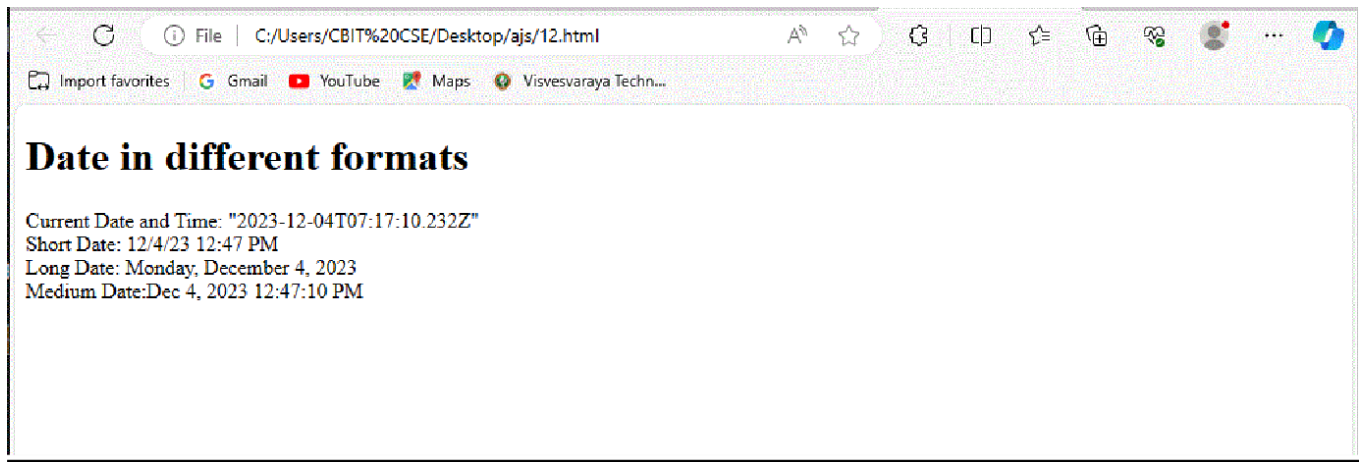
12. Create an AngularJS application that displays the date by using date filter parameters.

```
<!DOCTYPE html>
<html>
  <title>Date Application</title>
  <head>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
<script>
var app=angular.module("dateApp",[]);
app.controller("dateAppCntrl",function($scope){
$scope.currentDate=new Date();
});
</script>
</head>
<body ng-app="dateApp">
  <h1>Date in different formats</h1>
  <div ng-controller="dateAppCntrl">

    Current Date and Time: {{currentDate}}<br/>
    Short Date: {{currentDate|date:'short'}}<br/>
    Long Date: {{currentDate|date:'fullDate'}}<br/>
    Medium Date:{{currentDate|date:'medium'}}

  </div>
</body>
</html>
```

Output:



VIVA QUESTIONS

1. What is Angular JS?
2. What are SPA's?
3. Explain pre requisites of Angular JS?
4. What is environmental setup?
5. Explain general features of Angular JS?
6. Explain core features of Angular JS?
7. Explain MVC Architecture?
8. What is html and the syntax for html?
9. Explain CSS and syntax for CSS?
10. Explain Javascript and its syntax?
11. Explain some basic tags in html?
12. What are Angular JS Directives?
13. Explain ng-app syntax?
14. Explain ng-model syntax?
15. Explain syntax for ng-bind?
16. What is module?
17. How to create a module?
18. What is controller?
19. How to create a controller?
20. How to register controller with module?
21. What is Node.JS?
22. Explain installation of Node.JS?
23. Explain the command to install Angular cli?
24. Explain features of Node JS?
25. Explain the parts of Node JS?
26. What is error handling in Node JS?
27. Explain 4 types of error handling in Node JS?
28. Explain \$scope function ?
29. Explain require () function in Node JS?
30. Explain diagram of MVC and its 3 parts?