

CITY ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING

Affiliated to

Visveswaraya Technological University

“Jnana Sangama”, Belgaum-560018



LABORATRY MANUAL

V SEMESTER, CSE

“ANGULAR JS LABORATORY”

Subject Code: 21CSL581

Name: _____

Branch: _____

Semester: _____



CITY ENGINEERING COLLEGE

Department of Computer Science and Engineering

Email: info@cityengineeringcollege.ac.in

www.cityengineeringcollege.ac.in

Angular JS

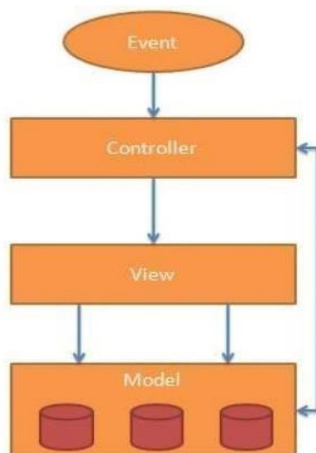
Angular JS is an open-source JavaScript framework by Google to build web applications. It can be freely used, changed and shared by anyone. Angular JS It is an excellent framework for building single phase applications and line of business applications.

Advantage of AngularJS

- **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
- **Two-way data binding:** AngularJS creates a two way data-binding between the select element and the orderBy model. orderBy is then used as the input for the orderBy filter.
- **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
- **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.

AngularJS MVC Architecture

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.



The MVC pattern is made up of the following three parts:

Model: It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.

View: It is responsible for displaying all data or only a portion of data to the users. It also specifies the 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.

Controller: It is responsible to control the relation between models and views. It 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.

AngularJS First Example

AngularJS applications are a mix of HTML and JavaScript. The first thing you need is an HTML page.

Second, you need to include the AngularJS JavaScript file in the HTML page so we can use AngularJS.

Following is a simple "Hello Word" example made with AngularJS. It specifies the Model, View, Controller part of an AngularJS app.

```
<!DOCTYPE html>

<html lang="en">

<head>

  <script
    src="https://ajax.googleapis.com/ajax/libs/a
    ngularjs/1.2.5/angular.min.js"></script>

</head>

<body ng-app="myapp">

  <div ng-controller="HelloController" >

<h2>Hello {{helloTo.title}} !</h2>
```

Output:

Hello World, AngularJS !

View Part

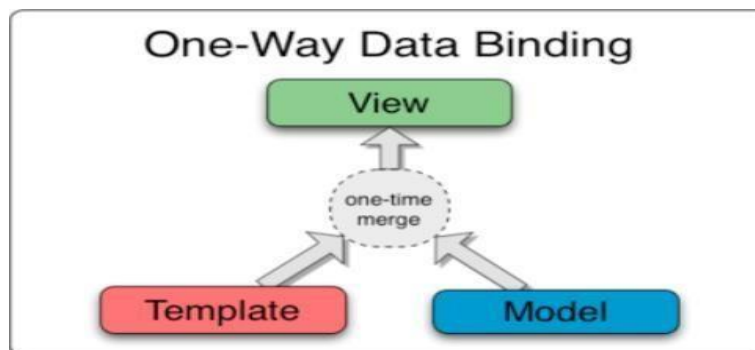
```
<div ng-controller="HelloController" >
<h2>Hello {{ helloTo.title }} !</h2>
<
/
d
```

AngularJS Data Binding

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application. AngularJS follows Two-Way data binding model:

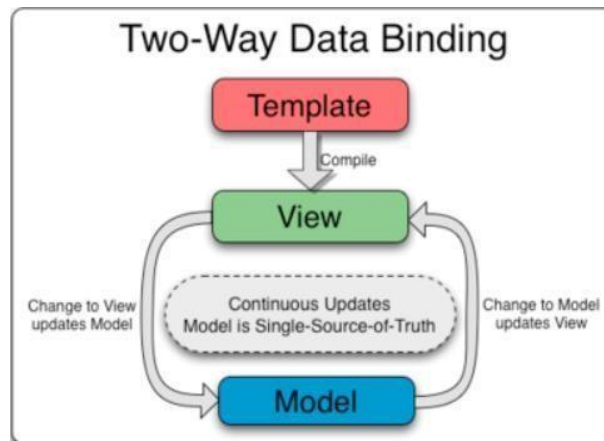
One-Way Data Binding

The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.



Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components. Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.



```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
```

Output:

Input something in the input box:

Name:

You wrote: Ajeet

In the above example, the `{{ firstName }}` expression is an AngularJS data binding expression. Data binding in AngularJS binds AngularJS expressions with AngularJS data.

`{{ firstName }}` is bound with `ng-model="firstName"`.

Let's take another example where two text fields are bound together with two ng-model directives:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div data-ng-app="" data-ng-
```

Output:

```
C
o
```

AngularJS Expressions

In AngularJS, expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written.

Expressions are written inside double braces {{ expression }}. They can also be written inside a directive:

ng-bind="expression"

AngularJS expressions are very similar to JavaScript expressions. They can contain literals, operators, and variables. For example,

{{ 5 + 5 }} or {{ firstName + " " + lastName }}

AngularJS Expressions Example

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

A simple expression example: 10

Note: If you remove the directive "ng-app", HTML will display the expression without solving it.

See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>
```

Output:

If you remove the directive "ng-app", HTML will display the expression without solving it.

A simple expression example: 10

You can also write expressions wherever you want, AngularJS will resolve the expression and return the result.

Let's take an example to change the color of input box by changing its value. See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<p>Change the value of the input field.</p>
```

Output:

Change the value of the input field:

pink

AngularJS resolves the expression and returns the result.

The background color of the input box will be whatever you write in the input field.

AngularJS Numbers

AngularJS numbers are similar to JavaScript numbers.


```
<!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="" ng-init="quantity=5;cost=5">

<p>Total in dollar: {{quantity * cost }}</p>

</div>

</body>

</html>
```

Output:

Total in dollar: 25

We can use the same example by using ng-bind: See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

Total in dollar: 25

AngularJS Strings

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

My full name is: Sonoo Jaiswal

Same example with ng-bind:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

My full name is: Sonoo Jaiswal

AngularJS Objects

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

My name is Sonoo

Same example with ng-bind:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

The name is Sonoo

AngularJS Arrays

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

The first result is 1

Same example with ng-bind:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Output:

The first result is 1

Difference between AngularJS Expressions and JavaScript expressions:

- AngularJS expressions can be written inside HTML, while JavaScript expressions cannot.

- AngularJS expressions support filters, while JavaScript expressions do not.
- AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
- Similarity between AngularJS Expressions and JavaScript expressions:
- AngularJS expressions and JavaScript expressions both can contain literals, operators and variables.

AngularJS Directives

AngularJS facilitates you to extend HTML with new attributes. These attributes are called directives.

There is a set of built-in directives in AngularJS which offers functionality to your applications. You can also define your own directives:

Directives are special attributes starting with ng- prefix. Following are the most common directives.

ng-app: This directive starts an AngularJS Application.

ng-init: This directive initializes application data.

ng-model: This directive defines the model that is variable to be used in AngularJS.

ng-repeat: This directive repeats html elements for each item in a collection.

ng-app directive

ng-app directive defines the root element. It starts an AngularJS Application and automatically initializes or bootstraps the application when web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application.

See this example:

In following example, we've defined a default AngularJS application using ng-app attribute of a div element.

```
<div ng-app = "">
```

ng-init directive

ng-init directive initializes an AngularJS Application data. It defines the initial values for an

AngularJS application.

In following example, we'll initialize an array of countries. We're using JSON syntax to define array of countries.

```
<div ng-app = "" ng-init = "countries =  
[{locale:'en-IND',name:'India'}, {locale:'en-  
PAK',name:'Pakistan'}, {locale:'en-
```

ng-model directive:

ng-model directive defines the model/variable to be used in AngularJS

Application. In following example, we've defined a model named "name".

```
<div ng-app = "">  
...  
<p>Enter your Name: <input type = "text" ng-model = "name"></p>  
</div>
```

ng-repeat directive

ng-repeat directive repeats html elements for each item in a collection. In following example, we've iterated over array of countries.

```
<div ng-app = "">  
...  
<p>List of Countries with locale:</p>
```

AngularJS directives Example

Let's take an example to use all the above discussed directives:

```
<!DOCTYPE html>

<html>

<head>

  <title>AngularJS Directives</title>

</head>

<body>

  <h1>Sample Application</h1>

  <div ng-app = "" ng-init =
"countries = [{locale:'en-
IND',name:'India'}, {locale:'en-
PAK',name:'Pakistan'}, {locale:'en-
```

Output:

Sample Application

Enter your Name:

Hello !

List of Countries with locale:

1. Country: India, Locale: en-IND
2. Country: Pakistan, Locale: en-PAK
3. Country: Australia, Locale: en-AUS

AngularJS Directives List

AngularJS directives are used to add functionality to your application. You can also add your own directives for your applications.

Following is a list of AngularJS directives:

Directives and Description

- **ng-app** It defines the root element of an application.
- **ng-bind** It binds the content of an html element to application data.
- **ng-bind-html** It binds the inner HTML of an HTML element to application data, and also removes dangerous code from the html string.
- **ng-bind-template** It specifies that the text content should be replaced with a template.
- **ng-blur** It specifies a behavior on blur events.
- **ng-change** It specifies an expression to evaluate when content is being changed by the user.
- **ng-checked** It specifies if an element is checked or not.
- **ng-class** It specifies css classes on html elements.
- **ng-class-even** It is same as ng-class, but will only take effect on even rows.
- **ng-class-odd** It is same as ng-class, but will only take effect on odd rows.
- **ng-click** It specifies an expression to evaluate when an element is being clicked.
- **ng-cloak** It prevents flickering when your application is being loaded.
- **ng-controller** It defines the controller object for an application.
- **ng-copy** It specifies a behavior on copy events.
- **ng-csp** It changes the content security policy.
- **ng-cut** It specifies a behavior on cut events.
- **ng-dblclick** It specifies a behavior on double-click events.
- **ng-focus** It specifies a behavior on focus events.
- **ng-hide** It hides or shows html elements.
- **ng-href** It specifies a URL for the <a> element.
- **ng-if** It removes the html element if a condition is false.
- **ng-include** It includes html in an application.

- **ng-init** It defines initial values for an application.
- **ng-jq** It specifies that the application must use a library, like jQuery.
- **ng-keydown** It specifies a behavior on keydown events.
- **ng-keypress** It specifies a behavior on keypress events.
- **ng-keyup** It specifies a behavior on keyup events.
- **ng-list** It converts text into a list (array).
- **ng-open** It specifies the open attribute of an element.
- **ng-options** It specifies <options> in a <select> list.
- **ng-paste** It specifies a behavior on paste events.
- **ng-pluralize** It specifies a message to display according to en-us localization rules.
- **ng-readonly** It specifies the readonly attribute of an element.
- **ng-required** It specifies the required attribute of an element.
- **ng-selected** It specifies the selected attribute of an element.
- **ng-show** It shows or hides html elements.
- **ng-src** It specifies the src attribute for the element.
- **ng-srcset** It specifies the srcset attribute for the element.
- **ng-style** It specifies the style attribute for an element.
- **ng-submit** It specifies expressions to run on onsubmit events.
- **ng-switch** It specifies a condition that will be used to show/hide child elements.
- **ng-transclude** It specifies a point to insert transcluded elements.
- **ng-value** It specifies the value of an input element.
- **ng-disabled** It specifies if an element is disabled or not.
- **ng-form** It specifies an html form to inherit controls from.
- **ng-model** It binds the value of html controls to application data.
- **ng-model-options** It specifies how updates in the model are done.
- **ng-mousedown** It specifies a behavior on mousedown events.
- **ng-mouseenter** It specifies a behavior on mouseenter events.
- **ng-mouseleave** It specifies a behavior on mouseleave events.
- **ng-mousemove** It specifies a behavior on mousemove events.

- **ng-mouseover** It specifies a behavior on mouseover events.
- **ng-mouseup** It specifies a behavior on mouseup events.
- **ng-non-bindable** It specifies that no data binding can happen in this element, or it's children.
- **ng-repeat** It defines a template for each data in a collection.

How to add directives

See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="myApp" w3-test-directive></div>
```

Output:

This is a directive constructor.

AngularJS Controllers

AngularJS controllers are used to control the flow of data of AngularJS application. A controller is defined using ng-controller directive. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts \$scope as a parameter which refers to the application/module that controller is to control.

AngularJS Controller Example

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

First

Name:
```

Output:

First Name:
Last Name:

Full Name: Aryan Khanna

Note:

Here, the AngularJS application runs inside the <div> is defined by ng-app="myApp". The AngularJS directive is ng-controller="myCtrl" attribute.

The myCtrl function is a JavaScript function.

AngularJS will invoke the controller with a \$scope object.

In AngularJS, \$scope is the application object (the owner of application variables and functions).

The controller creates two properties (variables) in the scope (firstName and lastName).

The ng-model directives bind the input fields to the controller properties (firstName and lastName).

AngularJS controller example with methods (variables as functions)

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/
angularjs/1.4.8/angular.min.js"></script>

<body>

<div

ng-

app=

"my

App"
```

Output:

First Name:
Last Name:

Full Name: Aryan Khanna

AngularJS Controller in external files

In larger applications, generally the controllers are stored in external files.

Create an external file named "personController.js" to store controller.

Here, "personController.js" is:

```
angular.module('myApp',  
[]).controller('personCtrl',  
function($scope) {  
  
    $scope.firstName = "Aryan",
```

See this example:

```
<!DOCTYPE html>  
  
<html>  
  
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>  
  
<body>  
  
<div ng-app="myApp" ng-controller="personCtrl">  
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 src="personController.js"></script>  
  
</body></html>
```

Output:

First Name:
Last Name:

Full Name: Aryan Khanna

AngularJS Module

In AngularJS, a module defines an application. It is a container for the different parts of your application like controller, services, filters, directives etc.

A module is used as a Main() method. Controller always belongs to a module.

How to create a module

The angular object's module() method is used to create a module. It is also called AngularJS function angular.module

```
<div ng-app="myApp">...</div>  
<script>
```

Here, "myApp" specifies an HTML element in which the application will run. Now we can add controllers, directives, filters, and more, to AngularJS application. How to add controller to a module

If you want to add a controller to your application refer to the controller with the ng-controller directive.

See this example:

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

```
{{ firstName + " " + lastName }}  
  
</div>  
  
<script>  
var app = angular.module("myApp", []);  
app.controller("myCtrl", function($scope) {  
    $scope.firstName = "Ajeet";  
    $scope.lastName = "Maurya";  
});  
</script>  
</body>  
</html>
```

Output:

Ajeet Maurya

How to add directive to a module

```
<!DOCTYPE html>  
  
<html>  
  
<script  
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>  
  
<body>  
  
  
<div ng-app="myApp" w3-test-
```

```
</html>
```

Output:

This is a directive constructor.

Modules and controllers in file

In AngularJS applications, you can put the module and the controllers in JavaScript files.

In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller:

See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

Here "myApp.js" contains:

```
app.controller("myCtrl", function($scope) {

$scope.firstName = "Ajeet";
```

Here "myCtrl.js" contains:

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


This example can also be written as:

```
<!DOCTYPE html>

<html>

<body>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<div ng-app="myApp" ng-
controller="myCtrl">
{{ firstName + " " + lastName }}
```

Output:

Ajeet Maurya

AngularJS Scopes

The Scope is an object that is specified as a binding part between the HTML (view) and the JavaScript (controller). It plays a role of joining controller with the views. It is available for both the view and the controller.

How to use Scope

To make a controller in AngularJS, you have to pass the \$scope object as an argument. See this example:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<h1>{{ carname }}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
<p>The property "carname" was made in the controller, and can be referred to in the view by
using the {{ }} brackets.</p>
</body>
</html>
```

Output:

Volvo

The property "carname" was made in the controller, and can be referred to in the view by using the {{ }} brackets.

AngularJS Filters

In AngularJS, filters are used to format data. Following is a list of filters used for transforming data.

Filters and Description

- **Currency** It formats a number to a currency format.
- **Date** It formats a date to a specified format.
- **Filter** It select a subset of items from an array.
- **Json** It formats an object to a Json string.
- **Limit** It is used to limit an array/string, into a specified number of elements/characters.
- **Lowercase** It formats a string to lower case.
- **Number** It formats a number to a string.
- **Orderby** It orders an array by an expression.
- **Uppercase** It formats a string to upper case.

How to add filters to expressions

You can add filters to expressions by using the pipe character |, followed by a filter. In this example, the uppercase filter format strings to upper case:

See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
```

Output:

The name is SONOO

Let's apply the lowercase filter into the same example: See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
```

Output:

The name is sonoo

How to add filters to directives

Filters can be added to directives, like ng-repeat, by using the pipe character |, followed by a filter.

Let's take an example: In this example, orderBy filter is used to sort an array.

See this example:

```
<!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<p>Looping with objects:</p>

<ul>

<li ng-repeat="x in names | orderBy:'country'">

  {{ x.name + ', ' + x.country }}

</li>

</ul>

</div>

<script>

angular.module('myApp', []).controller('namesCtrl', function($scope) {

$scope.names = [

  { name:'Ramesh',country:'India'},

  { name:'Alex',country:'USA'},

  { name:'Pooja',country:'India'},

  { name:'Mahesh',country:'India'},

  { name:'Iqbal',country:'Pakistan'},

  { name:'Ramanujam',country:'India'},

  { name:'Osama',country:'Iraq'},

  { name:'Johnson',country:'UK'},

  { name:'Karl',country:'Russia'}

];

});

</script>

</body>

</html>
```

Output :

Looping with objects:

- Ramesh, India
- Pooja, India
- Mahesh, India
- Ramanujam, India
- Osama, Iraq
- Iqbal, Pakistan
- Karl, Russia
- Johnson, UK
- Alex, USA

The filter Filter

The filter Filter can only be used on arrays because it selects a subset of an array. It returns an array containing only the matching items.

Let's take an example:

This example will return the names that contain the letter

"o". See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="myApp" ng-
controller="namesCtrl">
```

```
'Ramesh',  
'Pooja',  
'Mahesh',  
'Ramanujam',  
'Osama',  
'Iqbal',  
'Karl',  
'Johnson',  
'Alex'  
  ];  
});  
</script>  
<p>This example displays only the names containing the letter "o".</p>  
</body>  
</html>
```

Output:

- Pooja
- Osama
- Johnson

This example displays only the names containing the letter "o".

Filter an array based on user input

You can use the value of the input field as an expression in a filter by setting the ng-model directive on an input field.

See this example:

```
<!DOCTYPE html>
```

```
<body>

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

<p>Type a letter in the input field:</p>

<p><input type="text" ng-model="test"></p>

<ul>

<li ng-repeat="x in names | filter:test">

  {{ x }}

</li>

</ul>

</div>

<script>

angular.module('myApp', []).controller('namesCtrl', function($scope) {

$scope.names = [

'Ramesh',

'Pooja',

'Mahesh',

'Ramanujam',

'Osama',

'Iqbal',

'Karl',

'Johnson',

'Alex'

];

});

</script>

<p>The list will only contain the names matching the filter.</p>

</body>

</html>
```


Output:

Type a letter in the input field:

- Ramesh
- Pooja
- Mahesh
- Ramanujam
- Osama
- Iqbal
- Karl
- Johnson
- Alex

The list will only contain the names matching the filter.

Sort an array based on user input

You can sort an array according to the table columns. See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<p>Click the table headers to change the
sorting order:</p>
```

```
</table>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        { name:'Ramesh',country:'India'},
        { name:'Alex',country:'USA'},
        { name:'Pooja',country:'India'},
        { name:'Mahesh',country:'India'},
        { name:'Iqbal',country:'Pakistan'},
        { name:'Ramanujam',country:'India'},
        { name:'Osama',country:'Iraq'},
        { name:'Johnson',country:'UK'},
        { name:'Karl',country:'Russia'}
    ];

    $scope.orderByMe = function(x) {
        $scope.myOrderBy = x;
    }
});
</script>
</body>
</html>
```

Output:

Click the table headers to change the sorting order:

Name	Country
Ramesh	India
Alex	USA
Pooja	India
Mahesh	India
Iqbal	Pakistan
Ramanujam	India
Osama	Iraq
Johnson	UK
Karl	Russia

AngularJS Custom Filters

You can create your own filters by register a new filter factory function with your module. See this example:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<p>Click the table headers to change the sorting order:</p>
<div ng-app="myApp" ng-controller="namesCtrl">
<table border="1" width="100%">
<tr>
<th ng-click="orderByMe('name')">Name</th>
<th ng-click="orderByMe('country')">Country</th>
</tr>
<tr ng-repeat="x in names | orderBy:myOrderBy">
<td>{{ x.name }}</td>
<td>{{ x.country }}</td>
</tr>
```

```
</table>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        { name:'Ramesh',country:'India'},
        { name:'Alex',country:'USA'},
        { name:'Pooja',country:'India'},
        { name:'Mahesh',country:'India'},
        { name:'Iqbal',country:'Pakistan'},
        { name:'Ramanujam',country:'India'},
        { name:'Osama',country:'Iraq'},
        { name:'Johnson',country:'UK'},
        { name:'Karl',country:'Russia'}
    ];
    $scope.orderByMe = function(x) {
        $scope.myOrderBy = x;
    }
});
</script>
</body>
</html>
```

Output:

Click the table headers to change the sorting order:

Name	Country
Ramesh	India
Alex	USA
Pooja	India
Mahesh	India
Iqbal	Pakistan
Ramanujam	India
Osama	Iraq
Johnson	UK
Karl	Russia

AngularJS Tables

The ng-repeat directive is used to draw tables in AngularJS. Displaying tables with AngularJS is very easy and simple.

Let's take an example. This example use ng-repeat directive to draw a table. See this example:

```
<
t
a
b
l
e
>
```

Displaying with CSS style

You can also style the tables by using CSS. See this example:

```
<style>
table, th , td {
    b
    o
```

```
background-color: #f2f2f2;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>
```

AngularJS Table example with CSS

```
<!DOCTYPE
html>

<html>
"http://ajax.googleapis.com/ajax/libs/a
ngularjs/1.3.14/angular.min.js"></scri
pt>

<style>

table, th , td {

b

o

r

d
```

```
<div ng-app = "mainApp" ng-controller = "studentController">
  <table border = "0">
    <tr>
      <td>Enter first name:</td>
      <td><input type = "text" ng-model = "student.firstName"></td>
    </tr>
    <tr>
      <td>Enter last name: </td>
      <td>
        <input type = "text" ng-model = "student.lastName">
      </td>
    </tr>
    <tr>
      <td>Name: </td>
      <td>{{ student.fullName() }}</td>
    </tr>
    <tr>
      <td>Subject:</td>
      <td>
        <table>
          <tr>
            <th>Name</th>
            <th>Marks</th>
          </tr>
          <tr ng-repeat = "subject in student.subjects">
            <td>{{ subject.name }}</td>
            <td>{{ subject.marks }}</td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
</div>
```

```
        </tr>
      </table>

    </div>
    <script>
      var mainApp = angular.module("mainApp", []);
      mainApp.controller('studentController', function($scope) {
        $scope.student = { firstName:
          "Rahul", lastName: "Rai",
          fees:500,
          subjects:[
            { name:'Physics',marks:850},
            { name:'Chemistry',marks:80},
            { name:'Math',marks:90},
            { name:'English',marks:80},
            { name:'Hindi',marks:70}
          ],
          fullName: function() { var
            studentObject;
            studentObject = $scope.student;
            return studentObject.firstName + " " + studentObject.lastName;
          }
        };
      });
    </script>
  </body>
</html>
```


Output:**AngularJS Sample Application**

Enter first name:	<input type="text" value="Rahul"/>												
Enter last name:	<input type="text" value="Rai"/>												
Name:	Rahul Rai												
Subject:	<table><thead><tr><th>Name</th><th>Marks</th></tr></thead><tbody><tr><td>Physics</td><td>850</td></tr><tr><td>Chemistry</td><td>80</td></tr><tr><td>Math</td><td>90</td></tr><tr><td>English</td><td>80</td></tr><tr><td>Hindi</td><td>70</td></tr></tbody></table>	Name	Marks	Physics	850	Chemistry	80	Math	90	English	80	Hindi	70
Name	Marks												
Physics	850												
Chemistry	80												
Math	90												
English	80												
Hindi	70												

AngularJS HTML DOM

In AngularJS, some directives can be used to bind application data to attributes of HTML DOM elements.

These directives are:

Directive Description

- **ng-disabled** It disables a given control.
- **ng-show** It shows a given control.
- **ng-hide** It hides a given control.
- **ng-click** It represents an AngularJS click event.

ng-disabled directive: The ng-disabled directive binds AngularJS application data to the disabled attribute of HTML elements. In the below code, it binds a model to a checkbox.

```
<input type = "checkbox" ng-model =
```

ng-show directive: The ng-show directive shows or hides an HTML element. In the below code, it binds a model to a checkbox.

```
<input type = "checkbox" ng-model =
```

ng-hide directive: The ng-hide directive hides or shows an HTML element. In the below code, it binds a model to a checkbox.

```
<input type = "checkbox" ng-model =
```

ng-click directive: The ng-click directive counts the total clicks an HTML element. In the below code, it binds a model to a checkbox.

```
<p>Total click: {{ clickCounter }}</p>
```

Let's take an example to deploy the all above directives and test the variations:

See this example:

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS HTML DOM</title>
</head>
<body>
  <h2>AngularJS Sample
  Application</h2>
```

```
<tr>
  <td><input type = "checkbox" ng-model = "showHide2">Hide Button</td>
  <td><button ng-hide = "showHide2">Click Me!</button></td>
</tr>
<tr>
  <td><p>Total click: {{ clickCounter }}</p></td>
  <td><button ng-click = "clickCounter = clickCounter + 1">Click Me!</button></td>
</tr>
</table>
</div>
<script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</body>
</html>
```

Output:**AngularJS Sample Application**

☐ Disable Button

☐ Show Button

☐ Hide Button

Total click: 3

AngularJS Forms

AngularJS facilitates you to create a form enriches with data binding and validation of input controls. Input controls are ways for a user to enter data. A form is a collection of controls for the purpose of grouping related controls together.

Following are the input controls used in AngularJS forms:

- input elements
- select elements
- button elements
- textarea elements

AngularJS provides multiple events that can be associated with the HTML controls. These events are associated with the different HTML input elements.

Following is a list of events supported in AngularJS:

- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseup
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

Data Binding

ng-model directive is used to provide data binding.

Let's take an example where ng-model directive binds the input controller to the rest of your application

See this example:

```
<!DOCTYPE html>
<html lang="en">

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

  <form>

    First Name: <input type="text" ng-model="firstname">

  </form>

</div>

<script>

var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {

  $scope.firstname = "Ajeet";

});

</script>

</body>
```

Output:

First Name:

You can also change the example in the following way: See this example:

```
<!DOCTYPE html>

<html>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

```
</div>

<p>Change the name inside the input field, and you will see the name in the header changes accordingly.</p>

</body>

</html>
```

Output:

First Name:

You entered:

Change the name inside the input field, and you will see the name in the header changes accordingly.

AngularJS Checkbox

A checkbox has a value true or false. The ng-model directive is used for a checkbox. See this example:

```
<!DOCTYPE html>

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="">

  <form>

    Check to show this:

    <input type="checkbox" ng-model="myVar">

  </form>

  <h1 ng-show="myVar">Checked</h1>

</div>

<p>The ng-show attribute is set to true when the checkbox is checked.</p>

</body>

</html>
```

Output:

Check to show this: ☐

The ng-show attribute is set to true when the checkbox is checked.

AngularJS Radio Buttons

ng-model directive is used to bind radio buttons in your applications.

Let's take an example to display some text, based on the value of the selected radio buttons. In this example, we are also using ng-switch directive to hide and show HTML sections depending on the value of the radio buttons.

See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<form>
  Pick a topic:
  <input type="radio" ng-model="myVar"
value="tuts">Tutorials
```

```
<div ng-switch-when="news">
  <h1>News</h1>
  <p>Welcome to the news portal.</p>
</div>
</div>
<p>The ng-switch directive hides and shows HTML sections depending on the value of the
radio buttons.</p>
</body>
</html>
```

Output:

Pick a topic: ☐ Tutorials ☐ Festivals ☐ News

Tutorials

Welcome to the best tutorials over the net

Festivals

Most famous festivals

News

Welcome to the news portal.

The ng-switch directive hides and shows HTML sections depending on the value of the radio buttons.

AngularJS Selectbox

ng-model directive is used to bind select boxes to your application. See this example:

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



```
<form>
Select a topic:
<select ng-model="myVar">
  <option value="">
  <option value="tuts">Tutorials
  <option value="fest">Festivals
  <option value="news">News
</select>
</form>
<div ng-switch="myVar">
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Welcome to the best tutorials over the net.</p>
  </div>
  <div ng-switch-when="fest">
    <h1>Festivals</h1>
    <p>Most famous festivals.</p>
  </div>
  <div ng-switch-when="news">
    <h1>News</h1>
    <p>Welcome to the news portal.</p>
  </div>
</div>
<p>The ng-switch directive hides and shows HTML sections depending on the value of
the radio buttons.</p>
</body>
</html>
```

Output:

Select a topic:

Tutorials

Welcome to the best tutorials over the net.

Festivals

Most famous festivals.

News

Welcome to the news portal.

The ng-switch directive hides and shows HTML sections depending on the value of the radio buttons.

AngularJS form example

```
<!DOCTYPE
html>

<html>
  <script src="http://ajax.googleapis.com/ajax/libs/a
    ngularjs/1.3.14/angular.min.js"></scri
    pt>

  <style>
    table, th , td {
```

```
}
table tr:nth-child(even) {
  background-color: lightyellow;
}
</style>
</head>
<body>
<h2>AngularJS Sample Application</h2>
<div ng-app = "mainApp" ng-controller = "studentController">
<form name = "studentForm" novalidate>
<table border = "0">
<tr>
<td>Enter first name:</td>
<td><input name = "firstname" type = "text" ng-model = "firstName" required>
      <span style = "color:red" ng-show = "studentForm.firstname.$dirty
      && studentForm.firstname.$invalid">
        <span ng-show = "studentForm.firstname.$error.required">First Name is
        required.</span>
      </span>
    </td>
</tr>
<tr>
<td>Enter last name: </td>
<td><input name = "lastname" type = "text" ng-model = "lastName" required>
      <span style = "color:red" ng-show = "studentForm.lastname.$dirty
      && studentForm.lastname.$invalid">
        <span ng-show = "studentForm.lastname.$error.required">Last Name is
        required.</span>
      </span>
    </td>
</tr>
```

```
<tr>
    <td>Email: </td><td><input name = "email" type = "email" ng-model =
"email" length = "100" required>
        <span style    ="color:red" ng-show    =    "studentForm.email.$dirty    &&
studentForm.email.$invalid">
            <span ng-show    ="studentForm.email.$error.required">Email is
required.</span>
            <span ng-show ="studentForm.email.$error.email">Invalid email
address.</span>
        </span>
    </td>
</tr>
<tr>
<td>
    <button ng-click = "reset()">Reset</button>
</td>
<td>
    <button ng-disabled = "studentForm.firstname.$dirty && studentForm.firstname.$invalid ||
studentForm.lastname.$dirty && studentForm.lastname.$invalid || studentForm.email.$dirty
&& studentForm.email.$invalid" ng-click="submit()">Submit</button>
</td>
</tr>
</table>
</form>
</div>
<script>
var mainApp = angular.module("mainApp", []);
mainApp.controller('studentController', function($scope) {
    $scope.reset = function(){
        $scope.firstName = "John";
        $scope.lastName = "Doe";
    }
});
```

```
$scope.email = "Johndoe@gmail.com";

}

$scope.reset();

});

</script>
```

AngularJS Form Validation

AngularJS provides client-side form validation. It checks the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

It also holds the information about whether the input fields have been touched, or modified, or not.

Following directives are generally used to track errors in an AngularJS form:

- \$dirty - states that value has been changed.
- \$invalid - states that value entered is invalid.
- \$error - states the exact error.

AngularJS Form Validation Example

```
<!DOCTYPE html>

<html>

<head>

  <title>Angular JS Forms</title>

  <script
src="http://ajax.googleapis.com/
ajax/libs/angularjs/1.3.14/angula
```

```
background-color: lightpink;
}
table tr:nth-child(even) {
background-color: lightyellow;
}
</style>
</head>
<body>
<h2>AngularJS Sample Application</h2>
<div ng-app = "mainApp" ng-controller = "studentController">
<form name = "studentForm" novalidate>
<table border = "0">
<tr>
<td>Enter first name:</td>
<td><input name = "firstname" type = "text" ng-model = "firstName" required>
<span style = "color:red" ng-show = "studentForm.firstname.$dirty &&
studentForm.firstname.$invalid">
<span ng-show = "studentForm.firstname.$error.required">First Name is
required.</span>
</span>
</td>
</tr>
<tr>
<td>Enter last name: </td>
<td><input name = "lastname" type = "text" ng-model = "lastName" required>
<span style = "color:red" ng-show = "studentForm.lastname.$dirty &&
studentForm.lastname.$invalid">
<span ng-show = "studentForm.lastname.$error.required">Last Name is
required.</span>
</span>
</td>
</tr>
</table>
</div>
```

```
<tr>
  <td>Email: </td><td><input name = "email" type = "email" ng-model =
"email" length = "100" required>
  <span style = "color:red" ng-show="studentForm.email.$dirty &&
studentForm.email.$invalid">
  <span ng-show="studentForm.email.$error.required">Email is
required.</span>
  <span ng-show= "studentForm.email.$error.email">Invalid email
address.</span>
</span>
</td>
</tr>
<tr>
<td>
<button ng-click = "reset()">Reset</button>
</td>
<td>
<button ng-disabled = "studentForm.firstname.$dirty && studentForm.firstname.$invalid ||
studentForm.lastname.$dirty && studentForm.lastname.$invalid || studentForm.email.$dirty
&& studentForm.email.$invalid" ng-click="submit()">Submit</button>
</td>
</tr>
</table>
</form>
</div>
<script>
var mainApp = angular.module("mainApp", []);
mainApp.controller('studentController', function($scope) {
$scope.reset = function(){
$scope.firstName = "CEC";
$scope.lastName = "Bangalore";
```

```
$scope.email = "CEC@gmail.com";  
  
}  
  
$scope.reset();  
  
});  
  
</script>  
  
</body>  
  
</html>
```

Output:

AngularJS Sample Application

Enter first name:	<input type="text"/>
-------------------	----------------------

ANGULAR JS			
Course Code	21CSL581/ 21CBL583	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Total marks	100
Examination type (SEE)	PRACTICAL		
Course objectives: <ul style="list-style-type: none">• To learn the basics of Angular JS framework.• To understand the Angular JS Modules, Forms, inputs, expression, data bindings and Filters• To gain experience of modern tool usage (VS Code, Atom or any other) in developing Web applications			
SLNO	Experiments		
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 user input.		
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.		
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.		
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 search for 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		
NOTE: Include necessary HTML elements and CSS for the above Angular applications.			
Course outcomes (Course Skill Set): At the end of the course the student will be able to: <ol style="list-style-type: none">1. Develop Angular JS programs using basic features2. Develop dynamic Web applications using AngularJS modules3. Make use of form validations and controls for interactive applications4. Apply the concepts of Expressions, data bindings and filters in developing Angular JS programs5. Make use of modern tools to develop Web applications			

Program 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.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
<title>Full Name Input with Default Values</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></
script>
<style>
  body {
    font-family: sans-serif;
  }
  .container {
    margin: 20px;
    padding:
20px;
    border: 1px solid #ccc;
  }
  label {
    display: block;
    margin-bottom:
5px;
  }
  input {
    width: 100%;
    padding: 5px;
    border: 1px solid #ccc;
  }
  .fullName {
    font-weight: bold;
  }
</style>
</head>
<body>
```

```
<div class="container" ng-controller="myCtrl">
  <h2>Full Name Input with Default Values</h2>
  <label for="firstName">First Name:</label>
    <input type="text" ng-model="firstName" id="firstName"
placeholder="Enter your first name" value="John">

  <label for="lastName">Last Name:</label>
    <input type="text" ng-model="lastName" id="lastName"
placeholder="Enter your last name" value="Doe">

  <br>

  <span class="fullName">Full Name: {{ firstName + ' ' + lastName }}</span>
</div>

<script>

var app = angular.module('myApp', []);

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

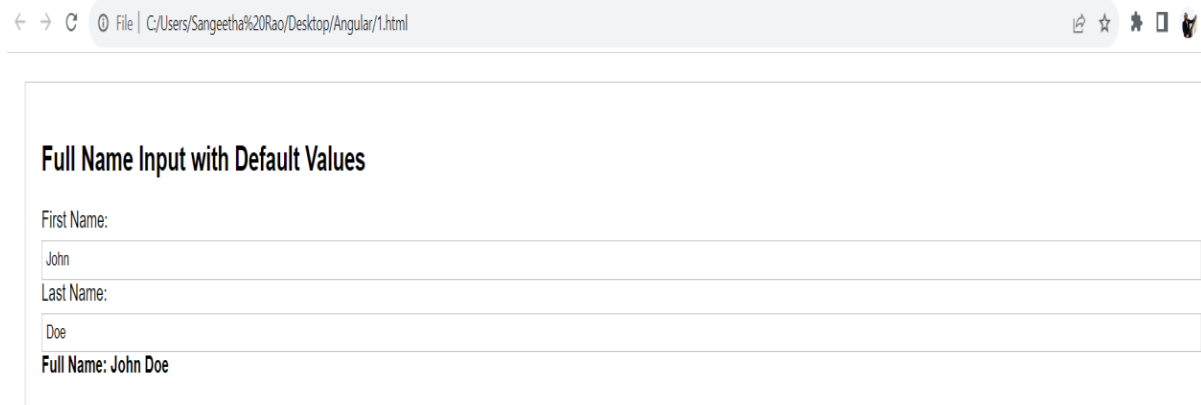
</body>

</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension
Step 2: Open the file using any browser



← → ↻ File | C:/Users/Sangeetha%20Rao/Desktop/Angular/1.html

Full Name Input with Default Values

First Name:
John

Last Name:
Doe

Full Name: John Doe

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.

```
<!DOCTYPE html>
<html ng-app="shoppingListApp">
<head>
<title>Shopping List</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></
scr ipt>
<style>
  body {
    font-family: sans-serif;
  }
.container {
  margin: 20px;
  padding:
20px;
  border: 1px solid #ccc;
  }
.shopping-list {
  list-style:
none;
  padding: 0;
  margin: 0;
  }
.shopping-list li {
  margin-bottom:
10px; padding: 5px;
  border: 1px solid #ccc;
  }
.shopping-list input {
  width: 80%;
  padding: 5px;
  border: 1px solid #ccc;
```

```
}
.shopping-list button {
padding: 5px 10px;
border: 1px solid #ccc;
cursor: pointer;
}
</style>
</head>
<body>
<div class="container" ng-controller="shoppingListCtrl">
<h1>Shopping List</h1>
<ul class="shopping-list">
<li ng-repeat="item in shoppingItems">
<input type="text" ng-model="item" placeholder="Enter item">
<button ng-click="removeItem(item)">Remove</button>
</li>
</ul>
<input type="text" ng-model="newItem" placeholder="Add new item">
<button ng-click="addItem()">Add</button>
</div>

<script>
var app = angular.module('shoppingListApp', []);

app.controller('shoppingListCtrl', function($scope) {
$scope.shoppingItems = ["Milk", "Bread", "Eggs"];

$scope.addItem = function() {
  if ($scope.newItem) {
    $scope.shoppingItems.push($scope.newItem);
    $scope.newItem = "";
  }
};


$scope.removeItem = function(item) {
  var index = $scope.shoppingItems.indexOf(item);
  if (index >= 0) {
    $scope.shoppingItems.splice(index, 1);
  }
}
```

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

Output:

Execution Steps:

Step 1: Save the program with .html extension
Step 2: Open the file using any browser



← → ↻ 127.0.0.1:5500/2.html

Shopping List

Milk	Remove
Bread	Remove
Eggs	Remove

Add new item

Program 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 ng-app="calculatorApp">
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></
scr ipt>
<style>
body{
font-family: sans-serif;
}
.container {
margin: 110px;
padding:20px;
border: 20px;
}

.operator-button {
padding: 5px 10px;
border:3px solid green;
background-color:skyblue;
cursor:pointer;
}
</style>
</head>
<div class="container" ng-controller="calculatorController">
<div class="operator-button"
<h2>Simple Calculator</h2>
<input type="number" ng-model="num1" placeholder="Enter number 1" >
<input type="number" ng-model="num2" placeholder="Enter number 2">
<select class="operator-button" ng-model="operator">
<option value="+">Addition</option>
<option value="-">Subtraction</option>
<option value="*">Multiplication</option>
<option value="/">Division</option>
```



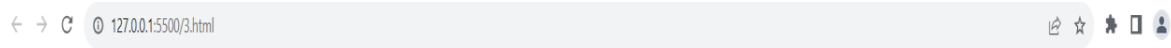
```
</select>
<button class="operator-button" ng-click="calculate()">Calculate</button>
<span class="result">Result: {{ result }}</span>
</div>
</div>
</html>
```

```
<script>
var app = angular.module('calculatorApp', []);
app.controller('calculatorController', function($scope) {
$scope.num1 = 0;
$scope.num2 = 0;
$scope.operator = '+';
$scope.result = 0;
$scope.calculate =
function() { if
($scope.operator === '+') {
$scope.result = $scope.num1 + $scope.num2;
} else if ($scope.operator === '-') {
$scope.result = $scope.num1 - $scope.num2;
} else if ($scope.operator === '*') {
$scope.result = $scope.num1 * $scope.num2;
} else if ($scope.operator ===
'/') { if ($scope.num2 === 0) {
$scope.result = 'Cannot divide by zero';
} else {
$scope.result = $scope.num1 / $scope.num2;
}
}
};
});
</script>
</body>
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension
Step 2: Open the file using any browser

A screenshot of a simple calculator web application. It has a light blue background. On the left, it says "Simple Calculator". There are two input fields, both containing the number "0". Between them is a dropdown menu currently showing "Addition". To the right of the dropdown is a button labeled "Calculate". Further right, it says "Result: 0".

Program 4

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

```
<!DOCTYPE html>
<html ng-app="mathApp">
<head>
<title>Math Operations</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></
scr ipt>
<style>
  body {
    font-family: sans-serif;
  }

.container {
  margin: 20px;
  padding:
20px;
  border: 1px solid #ccc;

}

.input-field {
  width: 100%;
  padding: 5px;
  border: 1px solid #ccc;

}

.button {
  padding: 5px 10px;
  border: 1px solid #ccc;
  cursor: pointer;

}

.result {
```

```
margin-top:
10px; font-
weight: bold;
```

```
}
</style>
</head>
<body>
<div class="container" ng-controller="mathCtrl">
<h2>Math Operations</h2>

<h3>Factorial</h3>
  <input type="number" ng-model="number" class="input-field"
placeholder="Enter number">
  <button class="button" ng-click="calculateFactorial()">Calculate
Factorial</button>
<span class="result">Factorial: {{ factorialResult }}</span>

<h3>Square</h3>
  <input type="number" ng-model="number2" class="input-field"
placeholder="Enter number">
  <button class="button" ng-click="calculateSquare()">Calculate
Square</button>
<span class="result">Square: {{ squareResult }}</span>
</div>

<script>
var app = angular.module('mathApp', []);

app.controller('mathCtrl', function($scope) {
$scope.number = 0;
$scope.factorialResult = 0;
$scope.number2 = 0;
$scope.squareResult = 0;

$scope.calculateFactorial = function() {
if ($scope.number < 0) {
alert("Invalid input. Please enter a non-negative number.");
return;
}

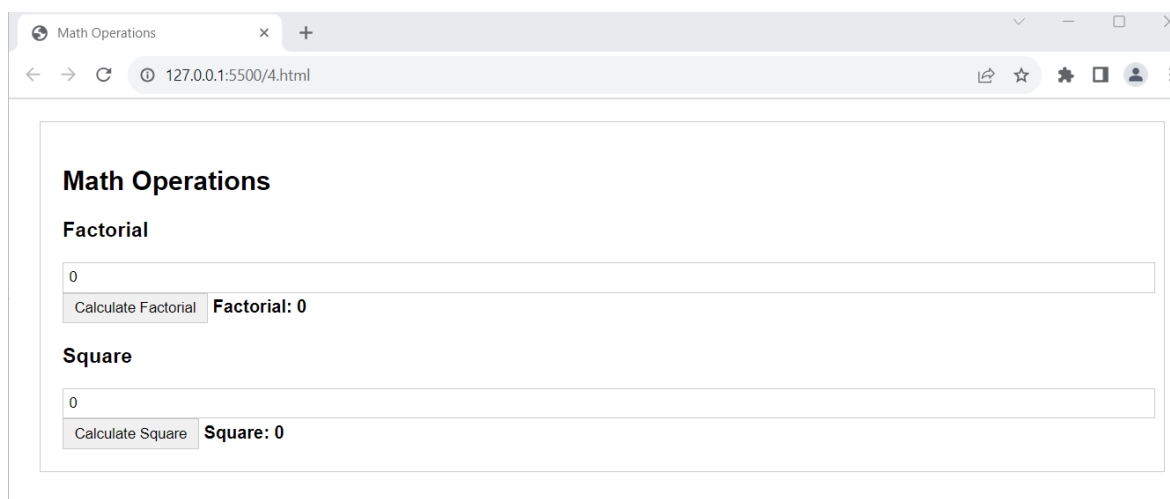
$scope.factorialResult = 1;
```

```
for (var i = $scope.number; i > 1; i--) {  
  $scope.factorialResult *= i;  
}  
};  
  
$scope.calculateSquare = function() {  
  $scope.squareResult = $scope.number2 * $scope.number2;  
};  
});  
</script>  
</body>  
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension
Step 2: Open the file using any browser



Program 5

Develop AngularJS application that displays a detail 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 ng-app="studentApp">
<head>
<title>Student Details</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></
scr ipt>
<style>
  body {
    font-family: sans-serif;
  }

.container {
  margin: 20px;
  padding:
20px;
  border: 1px solid #ccc;
}

table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  border: 1px solid #ddd;
  padding: 5px;
  text-align: center;
}

th {
  background-color: #f2f2f2;
}
</style>
</head>
```

```
<body>
<div class="container" ng-controller="studentCtrl">
<h2>Student Details</h2>

<table>
<thead>
<tr>
<th>Name</th>
<th>CGPA</th>
</tr>
</thead>
<tbody>
<tr ng-repeat="student in students">
<td>{{ student.name }}</td>
<td>{{ student.cgpa }}</td>
</tr>
</tbody>
</table>

<p>Number of Students: {{ students.length }}</p>
</div>

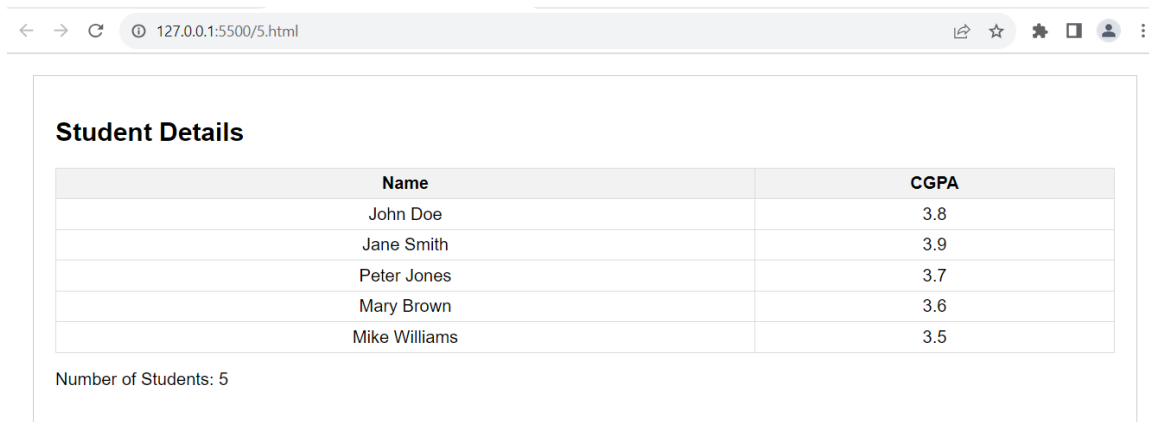
<script>
var app = angular.module('studentApp', []);

app.controller('studentCtrl', function($scope) {
$scope.students = [
{ name: "John Doe", cgpa: 3.8 },
{ name: "Jane Smith", cgpa: 3.9 },
{ name: "Peter Jones", cgpa: 3.7 },
{ name: "Mary Brown", cgpa: 3.6 },
{ name: "Mike Williams", cgpa: 3.5 },
];
});
</script>
</body>
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension
Step 2: Open the file using any browser



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/5.html". The page content is titled "Student Details" and features a table with two columns: "Name" and "CGPA". The table lists five students: John Doe (3.8), Jane Smith (3.9), Peter Jones (3.7), Mary Brown (3.6), and Mike Williams (3.5). Below the table, it states "Number of Students: 5".

Name	CGPA
John Doe	3.8
Jane Smith	3.9
Peter Jones	3.7
Mary Brown	3.6
Mike Williams	3.5

Number of Students: 5

Program 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 ng-app="todoListApp">
<head>
<title>To-Do List</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<style> body {
  font-family: sans-serif;
}

.container { margin: 20px;
padding: 20px;
border: 1px solid #ccc;
}

.task-list {
list-style: none; padding:
0;
margin: 0;
}

.task-list li {
margin-bottom: 10px; padding: 5px;
border: 1px solid #ccc;
}

.task-list input { width:
100%; padding: 5px;
border: 1px solid #ccc;
}

.task-list .edit-button { padding:
5px 10px; border: 1px solid #ccc;
cursor: pointer; margin-right:
10px;
}
```

```
.task-list .delete-button { padding:
5px 10px; border: 1px solid #ccc;
cursor: pointer;
}
</style>
</head>
<body>
<div class="container" ng-controller="todoListCtrl">
<h2>To-Do List</h2>

<input type="text" ng-model="newTask" placeholder="Enter new task">
<button ng-click="addTask()">Add Task</button>
<br>

<ul class="task-list">
<li ng-repeat="task in tasks">
<input type="checkbox" ng-model="task.done">
{{ task.text }}
<button class="edit-button" ng-click="editTask(task)">Edit</button>
<button class="delete-button"
ng-click="deleteTask(task)">Delete</button>
</li>
</ul>

<div ng-show="editingTask">
<input type="text" ng-model="editingTask.text" placeholder="Edit task">
<button ng-click="saveTask(editingTask)">Save</button>
<button ng-click="cancelEdit()">Cancel</button>
</div>
</div>

<script>
var app = angular.module('todoListApp', []); app.controller('todoListCtrl',

function($scope) {

$scope.tasks = [
{ text: "Buy groceries", done: false },
{ text: "Finish homework", done: false },
{ text: "Call doctor", done: false },
];
```

```
$scope.editingTask = null;

$scope.addTask = function() { if
($scope.newTask) {
  $scope.tasks.push({ text: $scope.newTask, done: false });
  $scope.newTask = "";
}
};

$scope.editTask = function(task) {
  $scope.editingTask = angular.copy(task);
};

$scope.saveTask = function(task) {
for (var i = 0; i < $scope.tasks.length; i++) { if
($scope.tasks[i].id === task.id) {
  $scope.tasks[i].text = task.text; break;
}
}
  $scope.editingTask = null;
};

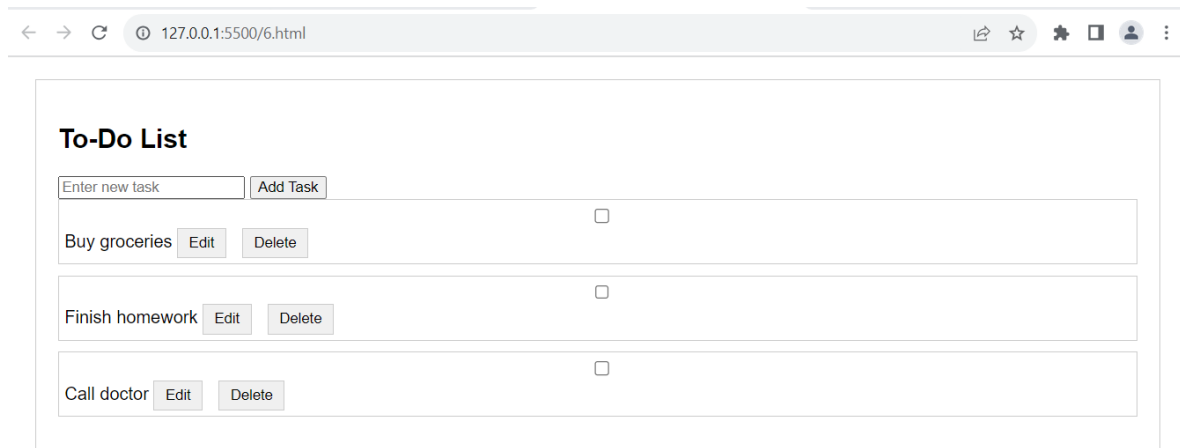
$scope.cancelEdit = function() {
  $scope.editingTask = null;
};

$scope.deleteTask = function(task) {
  var index = $scope.tasks.indexOf(task); if (index >=
0) {
    $scope.tasks.splice(index, 1);
  }
};
});
</script>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension Step 2:
Open the file using any browser



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/6.html". The page content is a "To-Do List" application. At the top, there is a text input field labeled "Enter new task" and an "Add Task" button. Below this, there are three task entries, each in a light gray box. The first task is "Buy groceries" with "Edit" and "Delete" buttons and an unchecked checkbox. The second task is "Finish homework" with "Edit" and "Delete" buttons and an unchecked checkbox. The third task is "Call doctor" with "Edit" and "Delete" buttons and an unchecked checkbox.

To-Do List	
Enter new task <input type="text"/> <input type="button" value="Add Task"/>	
Buy groceries <input type="button" value="Edit"/> <input type="button" value="Delete"/>	<input type="checkbox"/>
Finish homework <input type="button" value="Edit"/> <input type="button" value="Delete"/>	<input type="checkbox"/>
Call doctor <input type="button" value="Edit"/> <input type="button" value="Delete"/>	<input type="checkbox"/>

Program 7

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

```
<!DOCTYPE html>
<html ng-app="crudApp">
<head>
<title>AngularJS CRUD Application</title>

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

<style>
#userList {
width: 600px;
margin: auto;
border: 1px solid #ccc;
border-radius: 5px;
padding: 20px;
margin-top: 20px;
}

table {
width: 100%;
border-collapse: collapse;
margin-top: 10px;
}
th, td {
border: 1px solid #ddd;
padding: 8px;
text-align: left;
}

button {
background-color: #dc3545;
color: #fff;
border: none;
padding: 5px 10px;
```

```

border-radius: 3px;
cursor: pointer;
}
</style>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>

</head>
<body>

<div ng-controller="crudCtrl" id="userList">
<h2>Users List</h2>

<form ng-submit="addUser()">

<label for="userName">Name:</label>
<input type="text" id="userName" ng-model="newUser.name" required>

<label for="userEmail">Email:</label>
<input type="email" id="userEmail" ng-model="newUser.email" required>

<button type="submit">Add User</button>
</form>

<table ng-show="users.length > 0">
<tr>
<th>Name</th>
<th>Email</th>
<th>Actions</th>
</tr>
<tr ng-repeat="user in users">
<td>{{ user.name }}</td>
<td>{{ user.email }}</td>
<td>
<button ng-click="editUser(user)">Edit</button>
<button ng-click="deleteUser(user)">Delete</button>
</td>
</tr>
</table>
</div>

```

```
<script>
var app = angular.module('crudApp', []);
app.controller('crudCtrl', function ($scope) {
$scope.users = [
{ name: 'John', email: 'John@gmail.com' },
{ name: 'Smith', email: 'Smith@gmail.com' }
];
$scope.newUser = { };

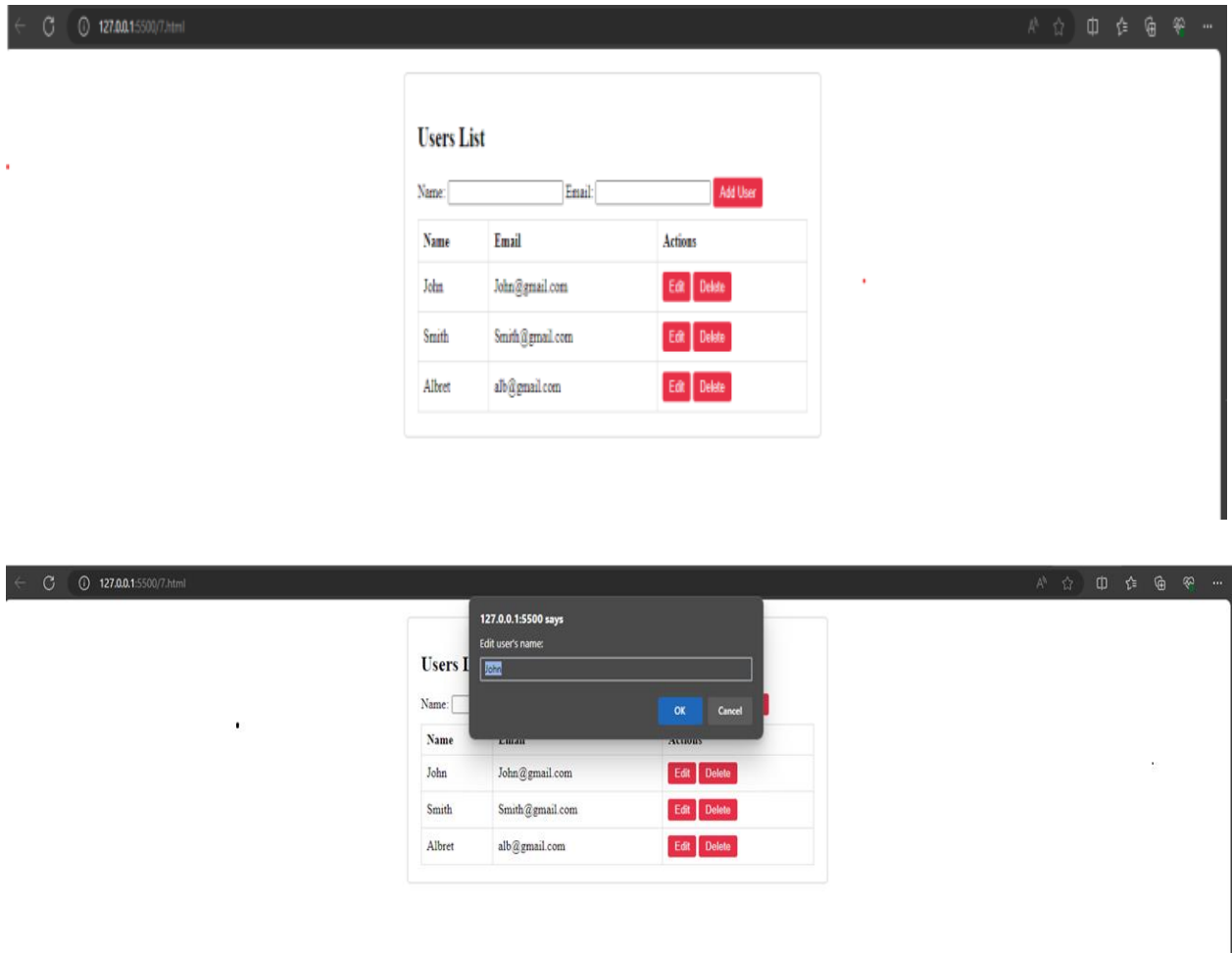
$scope.addUser = function () {
if ($scope.newUser.name && $scope.newUser.email) {
$scope.users.push(angular.copy($scope.newUser));
$scope.newUser = { };
}
};
$scope.editUser = function (user) {
var editedName = prompt("Edit user's name:", user.name);
var editedEmail = prompt("Edit user's email:", user.email);
if (editedName !== null && editedEmail !== null) {
user.name = editedName;
user.email = editedEmail;
}
};
$scope.deleteUser = function (user){
var index = $scope.users.indexOf(user);
$scope.users.splice(index, 1);
};
});
</script>
</body>
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension

Step 2: Open the file using any browser



Program 8

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

```
<!DOCTYPE html>
<html>
<head>
<title>Login Form with Validation</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
</head>
<body ng-app="myApp">

    <div ng-controller="LoginController">

        <h2>Login Form with Validation</h2>

        <form name="loginForm" novalidate>

            <label>Username:</label>

            <input type="text" ng-model="user.username" name="username" required>

            <span ng-show="loginForm.username.$error.required && loginForm.username.$dirty"> Username is required.</span>

            <br>
            <br>

            <label>Password :</label>

            <input type="password" ng-model="user.password" name="password"
ng-pattern="/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/" required>

            <span ng-show = "loginForm.password.$error.required &&
loginForm.password.$dirty"> Password is required.</span>

            <span ng-show="loginForm.password.$error.pattern &&
loginForm.password.$dirty">
```

Password must be alphanumeric and at least 8 characters long.

```
</span>
<br>
<br>
<button ng-click="login()" ng-disabled="loginForm.$invalid">Login</button>
<br>

<br>
WELCOME TO CEC!!!
</form>
<div ng-show="isLoggedIn">
<p>Login successful! Welcome, {{ user.username }}!</p>
</div>
</div>
<script>
var app = angular.module('myApp', []);
  app.controller('LoginController', function ($scope) {
    $scope.user = { username: "", password: "" };
    $scope.isLoggedIn = false;
    $scope.login = function () {
      $scope.isLoggedIn = true;
    };
  });
</script>
</body>
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension

Step 2: Open the file using any browser

Login Form with Validation

Username:

Password :

Login

WELCOME TO CEC!!!

Program 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 ng-app="employeeApp">

<head>
  <title>Employee Management</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>

<body ng-controller="employeeController">

  <h2>Employee Management</h2>

  <label for="searchName">Search by Name:</label>
  <input type="text" id="searchName" ng-model="searchName">

  <label for="searchSalary">Search by Salary:</label>
  <input type="number" id="searchSalary" ng-model="searchSalary">

  <table>
    <tr>
      <th>Name</th>
      <th>Salary</th>
    </tr>
    <tr ng-repeat="employee in employees | filter: { name: searchName, salary:
searchSalary }">
      <td>{{ employee.name }}</td>
      <td>{{ employee.salary | currency }}</td>
    </tr>
  </table>
```

```
<script>
  var app = angular.module('employeeApp', []);
  app.controller('employeeController', function ($scope) {
    $scope.employees = [
      { name: 'John Doe', salary: 50000 },
      { name: 'Jane Smith', salary: 60000 },
      { name: 'Bob Johnson', salary: 55000 },
      { name: 'Alice Williams', salary: 70000 },
      // Add more employees as needed
    ];
  });
</script>

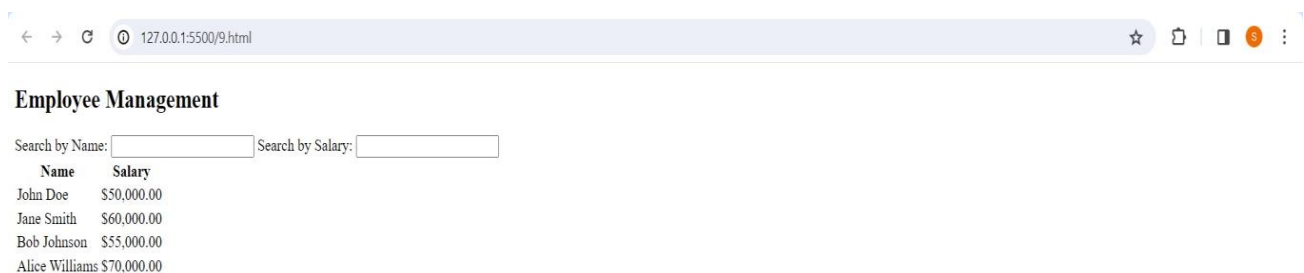
</body>

</html>
```

Output:

Execution Steps:

- Step 1: Save the program with .html extension
- Step 2: Open the file using any browser



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

```
<!DOCTYPE html>
<html>
<head>
<title>Item Collection Management</title>
<script      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
</head>
<body ng-app="myApp">
<div ng-controller="ItemController">
<h2>Item Collection</h2>
<p>Total number of items: {{ items.length }}</p>
<ul>
<li ng-repeat="item in items">
  {{ item.name }}
  <button ng-click="removeItem(item)">Remove</button>
</li>
</ul>
<div>
<label>New Item: </label>
<input type="text" ng-model="newItemName">
<button ng-click="addItem()">Add Item</button>
</div>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('ItemController', function ($scope) {
  // Default items
  $scope.items = [
    { name: 'Apple'},
    { name: 'Banana'},
    { name: 'Orange'}];
  $scope.newItemName = "";
  $scope.addItem = function () {
```

```
if ($scope.newItemName) {  
  $scope.items.push({ name: $scope.newItemName });  
  $scope.newItemName = "";  
}  
};  
$scope.removeItem = function (item) {  
  var index = $scope.items.indexOf(item);  
  if (index !== -1) {  
    $scope.items.splice(index, 1);  
  }  
};  
});  
</script>  
</body>  
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension

Step 2: Open the file using any browser



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

```
<!DOCTYPE html>
<html ng-app="studentApp">
<head>
  <title>AngularJS Student Details</title>
  <style>
    #studentDetails {
      width: 400px;
      margin: auto;
      border: 1px solid #ccc;
      border-radius: 5px;
      padding: 20px;
      margin-top: 20px;
      text-align: center;
    }

    input {
      width: 100%;
      margin-bottom: 10px;
      padding: 5px;
    }
  </style>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-controller="studentCtrl" id="studentDetails">
  <h2>Student Details</h2>

  <label for="studentName">Name:</label>
  <input type="text" id="studentName" ng-model="student.name" required>

  <label for="studentBranch">Branch:</label>
  <input type="text" id="studentBranch" ng-model="student.branch" required>

  <p>Uppercase Name: {{ student.name | uppercase }}</p>
  <p>Uppercase Branch: {{ student.branch | uppercase }}</p>
```

```
</div>
```

```
<script>
```

```
var app = angular.module('studentApp', []);
```

```
app.controller('studentCtrl', function ($scope) {
```

```
  $scope.student = {
```

```
    name: 'John Doe',
```

```
    branch: 'Computer Science'
```

```
  };
```

```
});
```

```
</script>
```

```
</body>
```

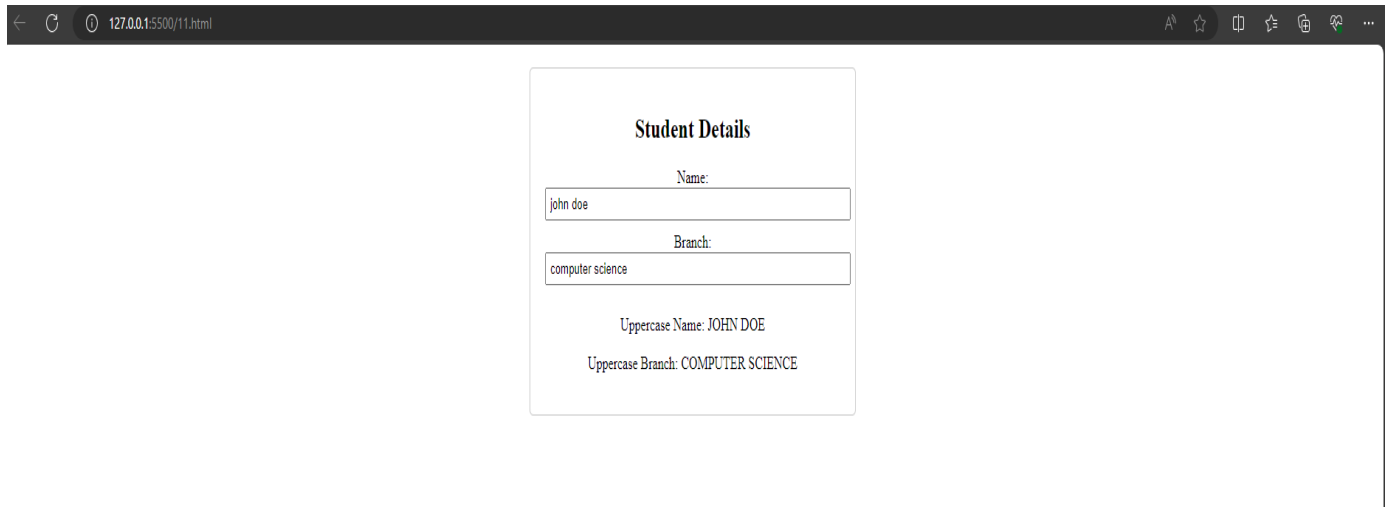
```
</html>
```

Output:

Execution Steps:

Step 1: Save the program with .html extension

Step 2: Open the file using any browser



Program 12

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

```
<!DOCTYPE html>
<html ng-app="dateApp">
<head>
  <title>AngularJS Date Display</title>

  <style>
    #dateDisplay {
      width: 400px;
      margin: auto;
      border: 1px solid #ccc;
      border-radius: 5px;
      padding: 20px;
      margin-top: 20px;
      text-align: center;
    }
  </style>

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

</head>

<body>

  <div ng-controller="dateCtrl" id="dateDisplay">
    <h2>Date Display</h2>

    <p>Default Format: {{ currentDate | date }}</p>
    <p>Custom Format: {{ currentDate | date:'fullDate' }}</p>
    <p>Short Format: {{ currentDate | date:'short' }}</p>
    <p>Custom Format (MM/dd/yyyy): {{ currentDate | date:'MM/dd/yyyy' }}</p>
  </div>

  <script>
    var app = angular.module('dateApp', []);

    app.controller('dateCtrl', function ($scope) {
      $scope.currentDate = new Date();
    });
  </script>
</body>
</html>
```

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

Output:

Execution

Steps:

Step 1: Save the program with .html extension

Step 2: Open the file using any browser

