ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY
(Unit of Alva's Education Foundation (R), Moodbidri)
Shobhavana Campus, MIJAR-574225, Moodbidri, D.K., Karnataka,
Affiliated to VTU, Belagavi & Approved by AICTE New Delhi.
Recognized by Govt. of Karnataka.
Accredited with 'A+' grade by NAAC & NBA (ECE & CSE)

DEPRTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

B.E 5th semester AngularJS Laboratory

Course Code: 21CSL581

Lab Manual

Academic Year 2023-24

Prepared By:
Mr. Mahesh Kini M
Sr. Assistant Professor,
Dept. of CSE
AIET, Miyar, Moodbidire.

Approved By:
Dr. Manjunath Kotari
Professor & Head,
Dept. of CSE
AIET, Miyar, Moodbidire.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", BELGAM-590 018

COURS E PRE-REQUISITES:

COURSE NAME	DESCRIPTION	COURSE CODE	SEM
Web Programming Laboratory	**Fundamentals of HTML, Javascript**	**21CSL481**	**IV**

COURSE OBJECTIVES:

This course will enable the students to:	
1	To learn the basics of Angular JS framework.
2	To understand the Angular JS Modules, Forms, inputs, expression, data bindings and Filters.
3	To gain experience of modem tool usage (VS Code, Atom or any other) in developing Web applications.

COURSE OUTCOMES (COs):

CO	DESCRIPTION OF COURSE OUTCOME:
	At the end of the course the student will be able to D:
CO1	Develop Angular JS programs using basic features.
CO2	Develop Web applications using AngularJS modules.
CO3	Make use of form validations and controls for interactive applications.
CO4	Apply the concepts of Expressions, data bindings and filters in developing Angular JS programs.
CO5	Make use of modern tools to develop Web applications.

COURSE INFORMATION SHEET

PROGRAM	CSE
DEGREE	Bachelor of Engineering(BE)
COURSE TITLE	Angular JS
SEMESTER	IV
COURSE CODE	21CSL81
COURSE TYPE (CONTENT)	Practical Based
REGULATION/SCHEME	VTU BE 2021
CREDITS	01
L-T-P-S	0-0-2-0
CONTACT HOURS/WEEK	2
TOTAL CONTACT HOURS	12T+24T
COURSE CATEGORY (ASSESSMENT)	Medium
SEE & CIE Marks	50, 50

COURSE CONTENT:

SlNo.	CONTENTS	HOURS
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
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.	2
3	Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input.	2
4	Write an Angular JS application that can calculate factorial and compute square based on given user input.	2
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.	2
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.	2
7	Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.	2
8	Develop AngularJS program to create a login form, with validation for the username and password fields.	2
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.	2
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.	2
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.	2
12	Create an AngularJS application that displays the date by using date filter parameters.	2
TOTAL HOURS		24

TEXT, REFERENCE BOOKS & E-RESOURCES

Textbooks:

1. ShyamSeshadri, Brad Green – "AngularJS: Up and Running: Enhanced Productivity with Structured Web Apps", Apress, 0'Reilly Media,Inc.
2. AgusKurniawan – "AngularJS Programming by Example", First Edition, PE Press, 2014

Web links and Video Lectures (e-Resources):

1. Introduction to Angular JS : https://www.youtube.com/watch?v=HEbphzK-0xE

2. Angular JS Modules : https://www.youtube.com/watch?v=gWm0KmgnQkU
3. https://www.youtube.com/watch?v=zKkUN-mJtPQ
4. https://www.youtube.com/watch?v=ICl7_i2mtZA
5. https://www.youtube.com/watch?v=Y2Few_nkze0
6. https://www.youtube.com/watch?v=QoptnVCQHsU

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

- Demonstration of simple projects/applications (course project)

COURSE PO/PSO CORRELATION MATRIX:

Course Code	Course Title				POs Mapped							PSOs Mapped		
21CSL581	ANGULAR JS				1,3,5,12							2		
Course Outcomes (COs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
CO1	3	2	2		2							3		2
CO2	3	2	2		2							3		2
CO3	3	2	2		2							3		2
CO4	3	2	2		2							3		2
CO5	3	2	2		2							3		2
Average score of COs mapping to POs / PSOs	1	0.66	0.66		0.66							1		0.66
Course Articulation level	3	2	2		2							3		2

Course Outcome Addresses Program Outcomes: 1-slight,2-Moderat,3 – Substantial.

Rubrics for course articulation level 66% and above - Articulation Level 3.

From 34%o up to 65% - Articulation level 2.

33% and below - Articulation level 1.

Detailed explanation of articulation matrix:

CO	Articulation / Reason for correlation (mapping)
1	The Course Outcome 1 has been mapped substantially to PO1, as students apply the knowledge of engineering fundamentals and principles of Web programming. Mapped moderately to PO2, as students identify and review complex engineering problems using principles of Engineering Sciences. Mapped moderately to PO3, as students design web applications that meet the specified needs with appropriate consideration for the public, business. The course outcome is mapped moderately to PO5, as students can select the modern tools like Visual Studio Code, Atom in building the web applications. Mapped substantially to PO12, as students engage in lifelong learning in the context of building web applications using upgraded web technology. Mapped moderately to PSO 2 as students apply

	the programming, designing in building the applications and guiding them innovative career path.
2	The course outcome 2 has been mapped substantially to PO1, as Students apply the knowledge of engineering fundamentals and principles of Web programming. Mapped moderately to PO2, as students identify and review complex engineering problems using principles of Engineering Sciences. Mapped moderately to PO3, as students design web applications that meet the specified needs with appropriate consideration for the public, business. The course outcome is mapped moderately to PO5, as Students can select the modem tools like Visual Studio Code, Atom in building the dynamic web applications. Mapped substantially to PO12, as students engage in lifelong learing in the context of building web applications using upgraded web technology. Mapped moderately to PSO 2 as students apply the programming, designing in building the applications and guiding them innovative career path.
3	The course outcome 3 has been mapped substantially to PO1, as Students apply the knowledge of engineering fundamentals and principles of Web programming. Mapped moderately to PO2, as students identify and review complex engineering problems using principles of Engineering Sciences. Mapped moderately to PO3, as students design web applications that meet the specified needs with appropriate consideration for the public, business. The course outcome is mapped moderately to PO5, as Students can select the modern tools like Visual Studio Code, Atom in building the interactive web applications. Mapped substantially to PO12, as students engage in lifelong learning in the context of building web applications using upgraded web technology. Mapped moderately to PSO 2 as students apply the programming, designing in building the applications and guiding them innovative career path.
4	The course outcome 4 has been mapped substantially to PO1, as Students apply the knowledge of engineering fundamentals and principles of Web programming. Mapped moderately to POs, as students identify and review complex engineering problems using principles of Engineering Sciences. Mapped moderately to PO3, as students design web applications that meet the specified needs with appropriate consideration for the public, business. The course outcome is mapped moderately to PO5, as Students can select the modern tools like Visual Studio Code, Atom for implementing Angular filters in web applications. Mapped substantially to PO12, as students engage in lifelong learning in the context of building web applications using upgraded web technology. Mapped moderately to PSO 2 as students apply the programming, designing in building the applications and guiding them innovative career path.
5	The course outcome 5 has been mapped substantially to PO1, as Students apply the knowledge of engineering fundamentals and principles of web programming. Mapped moderately to PO2, as students identify and review complex engineering problems using principles of Engineering Sciences. Mapped moderately to PO3, as students design web applications that meet the specified needs with appropriate consideration for the public business. The course outcome is mapped substantially to PO5, as Students select the modern tools like visual Studio code, Atom for developing better web applications. Mapped substantially to PO 12, as students engage in lifelong learning in the context of building web applications using upgraded web technology. Mapped moderately to PSO 2 as students apply the programming, designing in building the applications and guiding them innovative career path.

Lesson Plan for experiments:

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)	**PRACTICLE**		

SlNo.	CONTENTS	HOURS	Page No
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	01
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.	2	03
3	Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input.	2	06
4	Write an Angular JS application that can calculate factorial and compute square based on given user input.	2	09
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.	2	11
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.	2	13
7	Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.	2	15
8	Develop AngularJS program to create a login form, with validation for the username and password fields.	2	19
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.	2	21
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.	2	23
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.	2	25
12	Create an AngularJS application that displays the date by using date filter parameters.	2	27
TOTAL HOURS		24	

Program1: 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.

Description: Users can input their first and last names into an AngularJS program, which then displays their entire name. 'Srinivasa ' and ' Ramanujan ' are the default values assigned to the first and last names. These values are modifiable in the controller (myCtrl) as needed.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
      <h1>Full Name Program</h1>
      First Name: <input type="text" ng-model="firstName"><br>
      Last Name:  <input type="text" ng-model="lastName"><br>
      <br>
      Full Name: {{firstName + " " + lastName}}
</div>

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

OUTPUT:

Full Name Program

First Name: Srinivasa Last Name: Ramanujan

Full Name: Srinivasa Ramanujan

Full Name Program

First Name: AIET Last Name: MOODBIDIRE

Full Name: AIET MOODBIDIRE

Full Name Program

First Name: COMPUTER Last Name: SCIENCE

Full Name: COMPUTER SCIENCE

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.

Description: AngularJS application that shows a shopping list and lets users edit the list by adding and removing items with controllers and directives. The shopping list is managed by this AngularJS application using a controller called myCtrl. The item list is displayed using ng-repeat, and there is an input field for adding new items. When an item is clicked on the "X" text next to each one, it is removed from the list. Clicking the "Add Item" button adds the entered item to the list. The items in the default list are "Eggs," "Bread," and "Milk.".

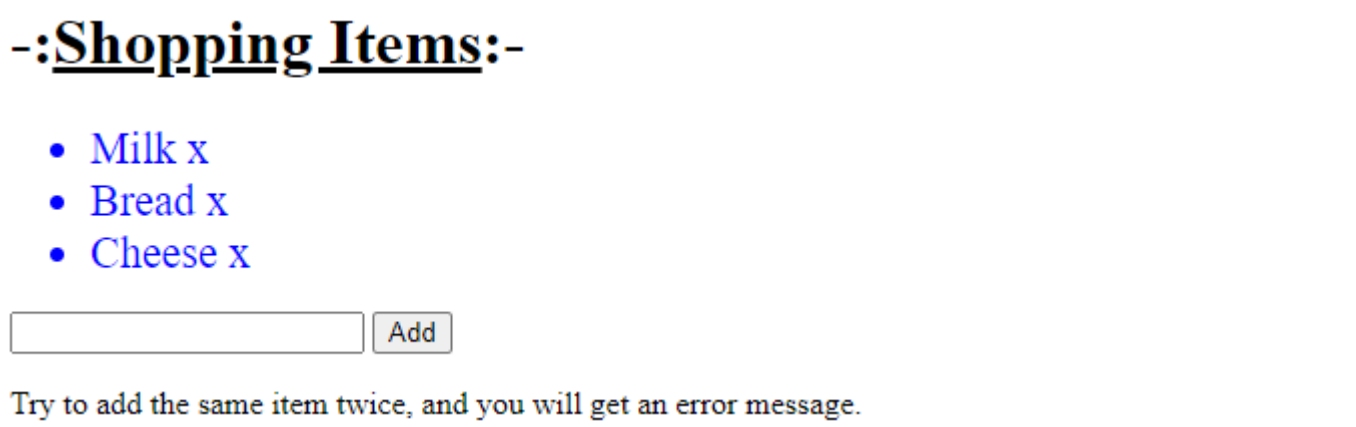
```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>

<body>
<div ng-app="myShoppingList" ng-controller="myCtrl">
 <h1> -:<u>Shopping Items</u>:- </h1>
 <ul>
  <li style="color:blue;font-size:22px" ng-repeat="x in products">{{x}}
      <span ng-click="removeItem($index)">x</span></li>
 </ul>
 <input ng-model="addMe">
 <button ng-click="addItem()">Add</button>
 <p>{{errortext}}</p>
</div>
<p>Try to add the same item twice, and you will get an error message.</p>

<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope) {
   $scope.products = ["Milk", "Bread", "Cheese"];
   $scope.addItem = function ()
   {
      $scope.errortext = "";
      if (!$scope.addMe) {return;}
```

```
      if ($scope.products.indexOf($scope.addMe) == -1)
       {
         $scope.products.push($scope.addMe);
      } else
       {
         $scope.errortext = "The item is already in your shopping list.";
      }
   }
   $scope.removeItem = function (x) {
      $scope.errortext = "";
      $scope.products.splice(x, 1);
   }
});
</script>
</body>
</html>
```

OUTPUT:



-:Shopping Items:-

- Milk x
- Bread x
- Cheese x
- Butter x

Butter Add

Try to add the same item twice, and you will get an error message.

-:Shopping Items:-

- Milk x
- Butter x

Butter Add

The item is already in your shopping list.

Try to add the same item twice, and you will get an error message.

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

Description: This AngularJS calculator app has two input fields where users can enter numbers, a dropdown menu where users can choose the operation to perform, and a 'Calculate' button where users can actually perform the calculation. The myCtrl handles the calculation based on the selected operation (addition, subtraction, multiplication, division) and displays the result.

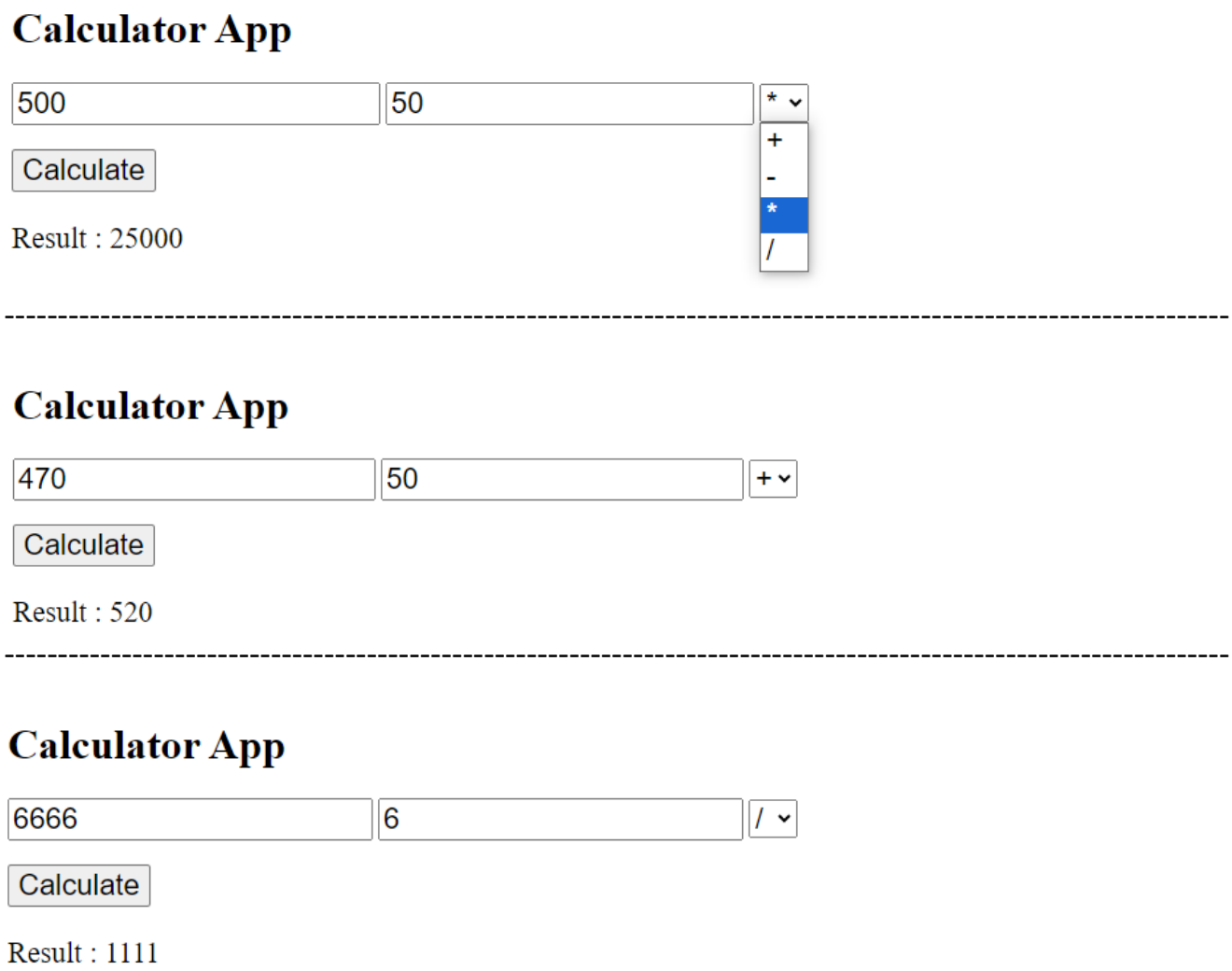
```
<!DOCTYPE html>
<html>
<head>
	<style>
		#f1 { font-size:22px }
	</style>
	<title>Calculator WebApp</title>
	<script
		 src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
	</script>
</head>

<body>

<div  ng-app="calApp" ng-controller="calCtrl">
  <h1>Calculator App</h1>
  <input id="f1" type="number"  ng-model="num1" placeholder="Enter First Number">
  <input id="f1" type="number"  ng-model="num2" placeholder="Enter Second Number">
  <select id="f1" ng-model="operator">
    <option value="+">+</option>
    <option value="-">-</option>
    <option value="*">*</option>
    <option value="/">/</option>
  </select>
  <br><br>
  <button  id="f1" ng-click="calculate()">Calculate</button>
```

```
    <p id="f1"> Result : {{result}}</p>
</div>
<script>
  var app = angular.module("calApp", []);
  app.controller("calCtrl", function($scope) {
   $scope.num1 = 0;
   $scope.num2 = 0;
   $scope.operator = "+";
   $scope.result = 0;
   $scope.calculate = function() {
    switch ($scope.operator) {
      case "+":
       $scope.result = parseFloat($scope.num1) + parseFloat($scope.num2);
       break;
      case "-":
       $scope.result = $scope.num1 - $scope.num2;
       break;
      case "*":
       $scope.result = $scope.num1 * $scope.num2;
       break;
      case "/":
       $scope.result = $scope.num1 / $scope.num2;
       break;
    }
   };
  });
</script>
</body>
</html>
```

OUTPUT:


Calculator App
500
50
*
+
-
*
/
Calculate
Result : 25000
Calculator App
470
50
+
Calculate
Result : 520
Calculator App
6666
6
/
Calculate
Result : 1111

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

Description: The factorial of a number is the product of all the numbers from 1 to that number. For example, factorial of 3 is equal to 1 * 2 * 3 = 6. The factorial of negative numbers does not exist and the factorial of 0 is 1. Two buttons are included in this AngularJS application: one calculates the factorial, and the other computes the square of a number entered by the user. Based on user input, the Factorial and Square can be calculated using functions in the myCtrl.

```
<!DOCTYPE html>
<html ng-app="Fact_SquareApp">
<head>
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
</head>
<body style="font-size:22px" ng-controller="FactController">
 <h2>Factorial and Square of a Number</h2>
 <input type="number" style="font-size:22px" ng-model="num" placeholder="Enter
number">
  <button style="font-size:22px" ng-click="factcal()">Calculate</button>
 <p>Factorial: {{ fresult }}</p>
 <p>Square: {{ sresult }}</p>
 <script>
  var app = angular.module('Fact_SquareApp', []);
  app.controller('FactController', function($scope) {
   $scope.num = 0;
   $scope.fresult = 1;
   $scope.sresult = 0;
    $scope.factcal = function()
   {
    if ($scope.num === 0)
     {
      $scope.fresult = 1;
      $scope.sresult = 0
     }
     else
     {
```

```
        $scope.fresult = 1;
        for (var i = 2; i <= $scope.num; i++)
                $scope.fresult = $scope.fresult * i;
         $scope.sresult = $scope.num * $scope.num;
      }
    };
  });
 </script>
</body>
</html>
```

OUTPUT:

Factorial and Square of a Number

8 Calculate

Factorial: 40320

Square: 64

Factorial and Square of a Number

15 Calculate

Factorial: 1307674368000

Square: 225

Program 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.

Description: The AngularJS program is designed to provide a user-friendly interface for displaying information about students, including their Cumulative Grade Point Average (CGPA). The total number of students is displayed using {{ students.length }}. The student details, including name and CGPA, are displayed in a table using ng-repeat.

Program5.html
Program.js

```
<!DOCTYPE html>
<html >
<head>
  <title>Student Details with CGPA</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="myController">
  <h2>Student Details</h2>
  <p>Total number of students: {{ students.length }}</p>
  <table border="1">
    <tr>
      <th>Name</th>
      <th>CGPA</th>
    </tr>
    <tr ng-repeat="student in students">
      <td>{{ student.name }}</td>
      <td>{{ student.cgpa }}</td>
    </tr>
  </table>
</div>
<script src="Program5.js"> </script>
</body>
</html>
```

```
var app = angular.module('myApp', []);
app.controller('myController', function ($scope) {
      // Default student details
      $scope.students = [
         { name: 'Athri',  cgpa: 9.8 },
         { name: 'Nachiketh',  cgpa: 9.2 },
         { name: 'Mary Disoza',  cgpa: 9.5 },
          { name: 'Md. Bilal',  cgpa: 9.4 }
         // Add more students if needed
      ];
  });
```

OUTPUT:

Student Details

Total number of students: 4

Name	CGPA
Athri	9.8
Nachiketh	9.2
Mary Disoza	9.5
Md. Bilal	9.4

Student Details

Total number of students: 8

Name	CGPA
Athri	9.8
Nachiketh	9.2
Mary Disoza	9.5
Md. Bilal	9.4
Babu Sequera	9.9
Rizwan Shrik	10
Ramani	9.7
Prakash Sharma	9.1

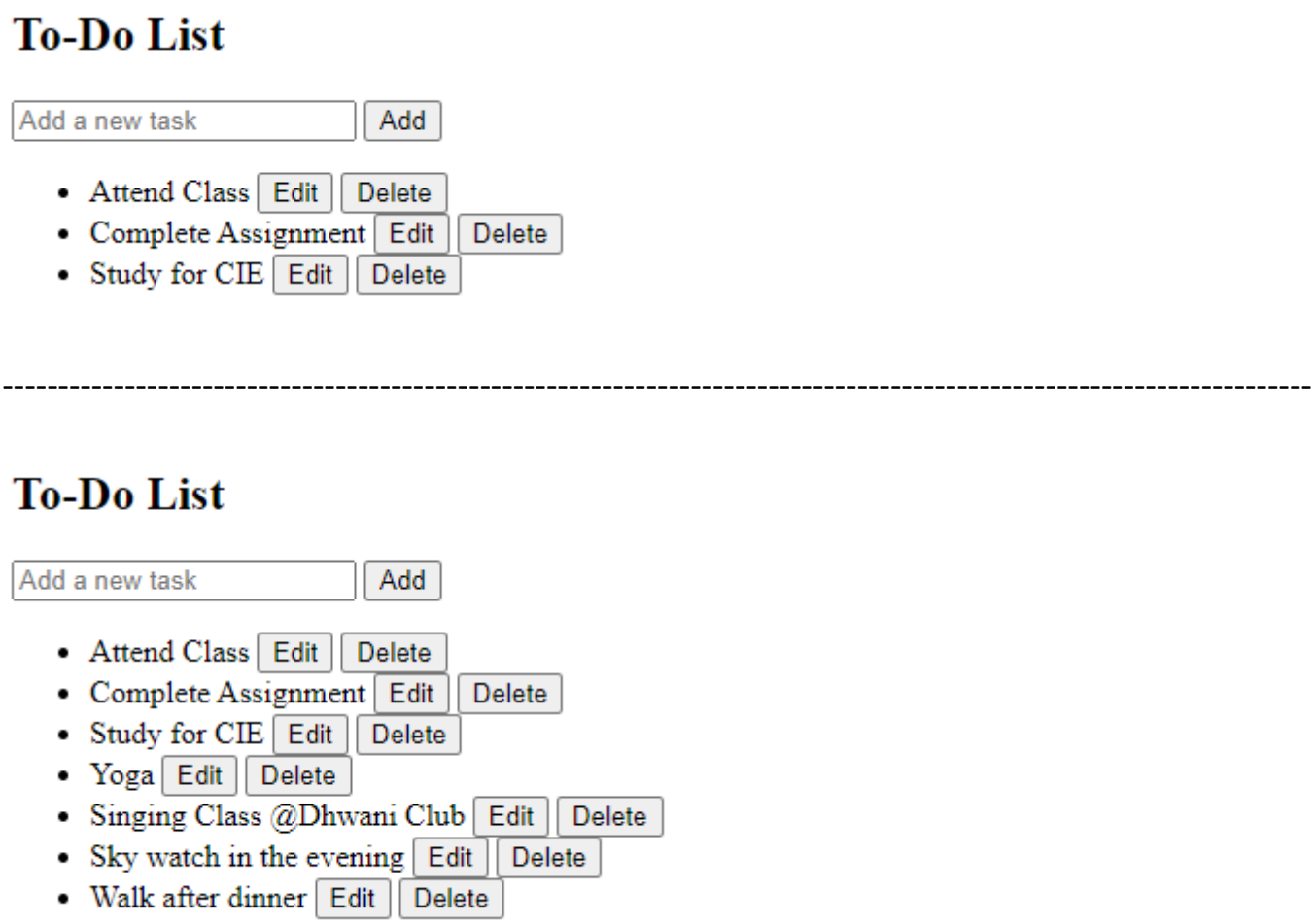
Program 6: Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks. Note: The default values for tasks may be included in the program.

Description: Users can add, edit, and remove tasks from a to-do list application using the AngularJS program. Tasks are displayed in an unordered list (<ul>). Users can add a new task by entering the task name and clicking the "Add Task" button. Each task has "Edit" and "Delete" buttons. Clicking the "Edit" button allows users to edit the task name, and they can save or cancel the edit. Clicking the "Delete" button removes the task from the list

```
<!DOCTYPE html>
<html ng-app="ToDoApp">
<head>
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
</head>
<body ng-controller="ToDoController">
 <h2>To-Do List</h2>
 <div>
  <input type="text" ng-model="newTask" placeholder="Add a new task">
  <button ng-click="addTask()">Add</button>
 </div>
 <ul>
  <li ng-repeat="task in tasks">
      {{ task.name }}
  <span ng-show="task.editing">
     <input type="text" ng-model="task.name" ng-blur="saveTask(task)">
  </span>
  <span ng-show="!task.editing">
     <button ng-click="editTask(task)">Edit</button>
     <button ng-click="removeTask(task)">Delete</button>
  </span>
  </li>
 </ul>
<script>
  var app = angular.module('ToDoApp', []);
  app.controller('ToDoController', function($scope) {
   //Default Tasks
      $scope.tasks = [{ name: 'Attend Class'},
```

```
                { name: 'Complete Assignment'},
                { name: 'Study for CIE'}];
    $scope.addTask = function()
    {
     if ($scope.newTask)
      {
       $scope.tasks.push({ name: $scope.newTask, editing: false });
       $scope.newTask = '';
      }
    };
    $scope.editTask = function(task)
    {
     task.editing = true;
    };
    $scope.saveTask = function(task)
    {
     task.editing = false;
    };
    $scope.removeTask = function(task)
    {
     const index = $scope.tasks.indexOf(task);
     if (index !== -1) {
       $scope.tasks.splice(index, 1);
     }
    };
  });
 </script>
</body>
</html>
```

OUTPUT:


To-Do List
Add a new task
Add
Attend Class Edit Delete
Complete Assignment Edit Delete
Study for CIE Edit Delete

To-Do List
Add a new task
Add
Attend Class Edit Delete
Complete Assignment Edit Delete
Study for CIE Edit Delete
Yoga Edit Delete
Singing Class @Dhwani Club Edit Delete
Sky watch in the evening Edit Delete
Walk after dinner Edit Delete

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

Description: The CRUD (Create, Read, Update, Delete) application for managing users written in AngularJS. A table shows the user's details. The "create" button can be used to add new users. There are "Edit" and "Delete" buttons for each user. Clicking "Edit" allows users to edit the user's name and email. Clicking "Delete" removes the user from the list.

```
<!DOCTYPE html>
<html ng-app="userApp">
<head>
<title>AngularJS CRUD Application for Users</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
</script>
</head>
<body ng-controller="UserController">
    <h1>User Management </h1>
      <table border="1">
      <thead>
       <tr>
          <th>Name</th>
          <th>Email</th>
          <th>Actions</th>
       </tr>
      </thead>
           <tr ng-repeat="user1 in users">
             <td>{{user1.name}}</td>
             <td>{{user1.email}}</td>
             <td>
                <button ng-click="editUser(user1)">Edit</button>
                <button ng-click="deleteUser(user1)">Delete</button>
             </td>
          </tr>
    </table>
  <hr>
  <h2>Create User</h2>

  <input type="text" ng-model="newUser.name" placeholder="Name"/>
```

```
    <input type="text" ng-model="newUser.email" placeholder="Email"/>
    <button ng-click="createUser()">Create</button>
    <hr>
     <h2>Edit User</h2>
        <input type="text" ng-model="editedUser.name" placeholder="Name"/>
        <input type="text" ng-model="editedUser.email" placeholder="Email"/>
        <button ng-click="updateUser()">Update</button>
  <script src="Program7.js"> </script>
</body>
</html>
var app = angular.module('userApp', []);
app.controller('UserController', function ($scope) {

 $scope.users = [
{name: 'Suma K', email: 'sumak@example.com' },
{ name: 'Ratan Kumar', email: 'ratankumar@example.com' }
   ];

   $scope.newUser = {};

   $scope.createUser = function() {
     $scope.users.push($scope.newUser);
     $scope.newUser = {};
   };

   $scope.editUser = function(user) {
     $scope.editedUser = user;
   };

   $scope.updateUser = function() {
     $scope.editedUser = {};
   };

   $scope.deleteUser = function(user) {
     $scope.users.splice($scope.users.indexOf(user), 1);
   };
  });
```

User Management

Name	Email	Actions
Suma K	sumak@example.com	Edit Delete
Ratan Kumar	ratankumar@example.com	Edit Delete

Create User

Name Email Create

Edit User

Name Email Update

User Management

Name	Email	Actions
Suma K M	sumakm@example.com	Edit Delete
Ratan Kumar	ratankumar@example.com	Edit Delete
Mohan	mohan@gmail.com	Edit Delete

Create User

Name Email Create

Edit User

Name Email Update

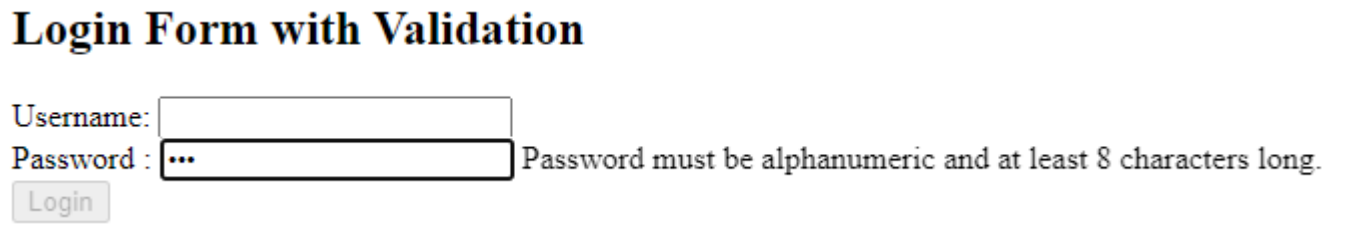
Program 8: Develop AngularJS program to create a login form, with validation for the username and password fields.

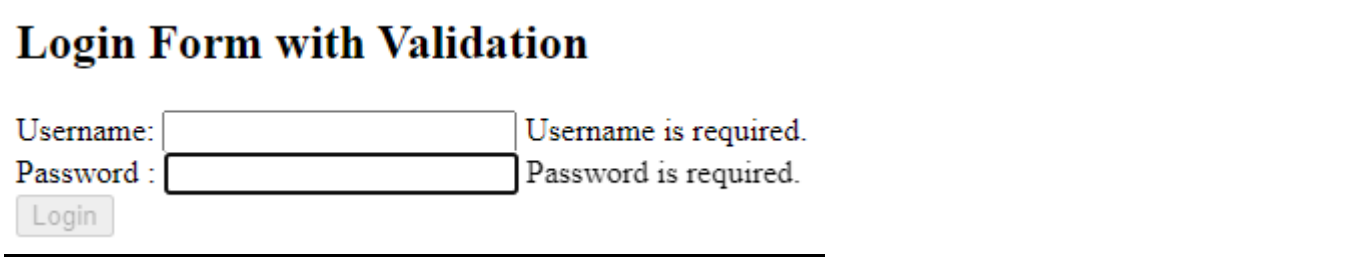
Description: For a login form with validation for the username and password. The password is required to be alphanumeric and 8 characters long. The form uses AngularJS validation with the required attribute for both the username and password fields. The password field also uses the ng-pattern attribute to enforce the alphanumeric requirement and a minimum length of 8 characters. Error messages are displayed based on the validation status. The "Login" button is disabled until the form is valid. The ng-click directive triggers the login() function, which can contain the actual login logic. For simplicity, it just sets isLoggedIn to true in this example.

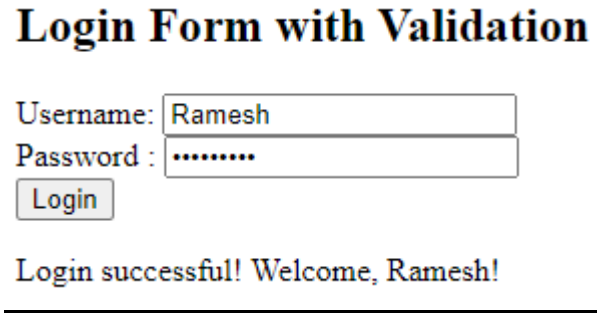
```
<!-- Test regex in https://www.regextester.com/105521 -->
<!DOCTYPE html>
<html>
<head>
  <title>Login Form with Validation</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
  </script>
</head>
<body ng-app="myApp">
<div ng-controller="LoginController">
  <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.
$dirty"> Username is required.</span>
    <br>
    <label>Password :</label>
    <input type="password" ng-model="user.password" name="password" ng-
pattern="/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/" 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>
    <button ng-click="login()" ng-disabled="loginForm.$invalid">Login</button>
```

```
  </form>
  <div ng-show="isLoggedIn">
    <p>Login successful! Welcome, {{ user.username }}!</p>
  </div>
</div>
<script src="Program8.js"> </script>
</body>
</html>
var app = angular.module('myApp', []);
app.controller('LoginController', function ($scope) {
  $scope.user = { username: '', password: '' };
  $scope.isLoggedIn = false;
  $scope.login = function () {
    $scope.isLoggedIn = true;
  };
});
```

OUTPUT:







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.

Description: The AngularJS application that displays a list of employees and their salaries. Employees are displayed in an unordered list (<ul>). Users can search for employees by name and salary using the ng-model directive. The filter pipe is used to filter employees based on the search criteria (name and salary). Employee salaries are formatted using the number filter to display two decimal places.

```
<!DOCTYPE html>
<html ng-app="employeeApp">
<head>
  <title>Employee List</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="EmployeeController">
  <h2>Employee List</h2>
  <label>Search by Name:</label>
  <input type="text" ng-model="searchName" />
  <label>Search by Salary:</label>
  <input type="number" ng-model="searchSalary" />
  <button ng-click="searchEmployees()">Search</button>
  <ul>
    <li ng-repeat="employee in filteredEmployees">
      {{ employee.name }} - {{ employee.salary}}
    </li>
  </ul>
  <script>
angular.module('employeeApp', [])
    .controller('EmployeeController', function ($scope) {
      $scope.employees = [
        { name: 'Keerthana SS', salary: 50000 },
        { name: 'Rakshith B', salary: 60000 },
        { name: 'Santhosh', salary: 70000 },
        { name: 'Radhika Pai', salary: 55000 },
        { name: 'Skandha', salary: 80000 }
      ];
```

```
    $scope.filteredEmployees = $scope.employees;
    $scope.searchEmployees = function () {
      $scope.filteredEmployees = $scope.employees.filter(function (employee) {
        return (
          (employee.name.toLowerCase().includes($scope.searchName.toLowerCase())    ||
!$scope.searchName) &&
          (employee.salary == $scope.searchSalary || !$scope.searchSalary)
        );
      });
    };
  });
 </script>
</body>
</html>
```

Employee List

Search by Name: Search by Salary: Search

- Keerthana SS - 50000
- Rakshith B - 60000
- Santhosh - 70000
- Radhika Pai - 55000
- Skandha - 80000
- Rakshith C - 125000

Employee List

Search by Name: Rakshith Search by Salary: Search

- Rakshith B - 60000
- Rakshith C - 125000

Employee List

Search by Name: Search by Salary: 70000 Search

- Santhosh - 70000

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.

Description: The AngularJS application that allows users to maintain a collection of items. The application displays the current total number of items, and this count updates automatically as items are added or removed. The total number of items is displayed using {{ items.length }}. Items are displayed in an unordered list (<ul>). Users can add a new item by entering the item name and clicking the "Add Item" button. Each item has a "Remove" button that allows users to remove the item from the collection.

```
<!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>
```

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.

Description: The uppercase Filter in AngularJS is used to change a string to an uppercase string or letters.

Syntax: {{ string | uppercase}} The uppercase filter is applied to each property of the student details to convert them to uppercase. The default student details are defined in the controller using the $scope.students array. The details are displayed in a table using ng-repeat to iterate over the students

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
 <title>Student Details</title>
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
<div ng-controller="StudentController as ctrl">
 <h2>Student Details</h2>
 <table border="1">
  <tr>
   <th>Name</th>
   <th>CGPA</th>
  </tr>
```

```
    <tr ng-repeat="student in ctrl.students">
      <td>{{ student.name | uppercase }}</td>
      <td>{{ student.cgpa | number:2 }}</td>
    </tr>
  </table>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('StudentController', function () {
  this.students = [
    { name: "Nagesh Rao", cgpa: 3.8 },
    { name: "Mohan N", cgpa: 3.6 },
    { name: "Rajesh Patak", cgpa: 3.9 },
    { name: "Shanthi N", cgpa: 3.7 }
    // Add more default students if needed
  ];
});
</script>
</body>
</html>
```

Student Details

Name	CGPA
NAGESH RAO	3.80
MOHAN N	3.60
RAJESH PATAK	3.90
SHANTHI N	3.70

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

Description: AngularJS date filter is used to convert a date into a specified format. When the date format is not specified, the default date format is 'MMM d, yyyy'.

Syntax: {{ date | date : format : timezone }} Parameter Values: The date filter contains format and timezone parameters which is optional.

In this program the user can select a date format from a dropdown list. The ng-change directive is used to trigger the updateDate function whenever the selected format changes. The updateDate function uses the AngularJS $filter service to format the current date based on the selected format. Formatted date is then displayed in the HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Date Display</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="myController">
  <h2>Date Display</h2>

  <label>Date Format:
    <select ng-model="selectedFormat" ng-change="updateDate()">
      <option value="fullDate">Full Date</option>
      <option value="shortDate">Short Date</option>
      <option value="mediumTime">Medium Time</option>
      <option value="shortTime">short Time</option>
      <option value="yyyy-MM-dd HH:mm:ss">Custom Format (yyyy-MM-dd
HH:mm:ss)</option>
    </select>
  </label>
  <p>Selected Date Format: {{ selectedFormat }}</p>
  <p>Formatted Date: {{ formattedDate }}</p>
</div>
```

```
<script>
var app = angular.module('myApp', []);
app.controller('myController', function ($scope, $filter) {
   $scope.selectedFormat = 'fullDate'; // Default date format
   $scope.updateDate = function () {
      var currentDate = new Date();
      $scope.formattedDate = $filter('date')(currentDate, $scope.selectedFormat);
   };
   // Initial date update
   $scope.updateDate();
});
</script>
</body>
</html>
```

Date Display

Date Format: Full Date

Selected Date Format: fullDate

Formatted Date: Thursday, November 30, 2023

Date Display

Date Format: Custom Format (yyyy-MM-dd HH:mm:ss)

Selected Date Format: yyyy-MM-dd HH:mm:ss

Formatted Date: 2023-11-30 20:33:13