JSS MAHAVIDYAPEETHA
JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU-60
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
JSSATE Campus, Dr. Vishnuvardhana Main Road, Bengaluru – 560060

V SEMESTER

AngularJS Lab Manual

[21CSL581]

Prepared By:

Dr. Abhijith H V
Associate Professor
Department of ISE
JSSATE-Bengaluru

Mrs. Sukrutha C Basappa
Assistant Professor
Department of ISE
JSSATE-Bengaluru

JSS MAHAVIDYAPEETHA
JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU-60
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
JSSATE Campus, Dr. Vishnuvardhana Main Road, Bengaluru – 560060

V SEMESTER

AngularJS Lab Manual

[21CSL581]

Signature of Faculty **Signature of HOD**

JSS MAHAVIDYAPEETHA
JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU-60
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

VISION

To emerge as a centre for achieving academic excellence, by producing competent professionals to meet the global challenges in the field of Information science and Technology.

MISSION

M1:To prepare the students as competent professionals to meet the advancements in the industry and academia by imparting quality technical education.

M2:To enrich the technical ability of students to face the world with confidence, commitment and teamwork

.

M3: To inculcate and practice strong techno-ethical values to serve the society.

Program Educational Objectives (PEOs):

PEO1: To demonstrate analytical and technical problem solving abilities.

PEO2:To be conversant in the developments of Information Science and Engineering, leading towards the employability and higher studies.

PEO3: To engage in research and development leading to new innovations and products.

Program Specific Outcomes (PSOs):

PSO1:Apply the mathematical concepts for solving engineering problems by using appropriate programming constructs

.

PSO2: Adaptability to software development methodologies.

PSO3:Demonstrate the knowledge towards the domain specific initiatives of Information Science and Engineering.

Program Outcomes (POs):

Information Science and Engineering Graduates will be able to:

PO1	Apply the knowledge of mathematics, science, engineering fundamentals, and an Engineering specialization to the solution of complex engineering problems.
PO2	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

AngularJS 21CSL581

Course Code	**21CSL581**	Sem	**V**
CIE Marks	**50**	No of contact hours/week	**0:0:2**
Total Number of Lab Contact hrs	**24**	SEE Marks	**50**
Exam Duration	**3 hrs**	Credits	**1**

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.

Lesson plan

AngularJS 21CSL581

Pre-requisites:

HTML, CSS and JavaScript

Course Outcomes:

CO#		CO Statement	BLL
CO1	C308.1	Appy the concepts of Directives, Expressions, data bindings in developing Angular JS programs	L3
CO2	C308.2	Apply the concepts of form validations and controls for interactive applications	L3
CO3	C308.3	Apply the concepts of filters for developing AngularJS Single page applications	L3

Class No.	Programs to be executed	CO	Material
1	Course Overview and Basics		
2	Develop Angular JS program that allows user to input their first name and last name and display their full name. 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.	CO1	TB1,TB2
3	Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input. Write an Angular JS application that can calculate factorial and compute square based on given user input.	CO1	TB1,TB2
4	Develop AngularJS application that displays a details of students and their CGPA. Allow users to read the number of students and display the count.	CO1	TB1,TB2
5	Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks.	CO1	TB1,TB2
6	Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.	CO1	TB1,TB2
7	Lab Internals - 1		
8	Develop AngularJS program to create a login form, with validation for the username and password fields.	CO2	TB1,TB2
9	Create an AngularJS application that displays a list of employees and their salaries. Allow users to search for employees by name and salary.	CO2	TB1,TB2
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.	CO3	TB1,TB2
11	Create AngularJS application to convert student details to Uppercase using angular filters.	CO3	TB1,TB2
12	Create an AngularJS application that displays the date by using date filter parameters	CO3	TB1,TB2
13	Lab Internals - 2		

TEXT BOOKS (TB):

Book Type	Code	Author/Title/Publishers
Text Book	TB1	ShyamSeshadri, Brad Green —"AngularJS: Up and Running: Enhanced Productivity with Structured Web Apps", Apress, 0'Reilly Media,Inc.
Text Book	TB2	AgusKurniawan–"AngularJS Programming by Example", First Edition, PE Press, 2014

CONTINUOUS EVALUATION PROCEDURE

Each program/Experiment will be evaluated for 30 Marks (CIE) and 2 IA for 50 marks each scaled down to 10.

Exp. No	Title of Experiment	Continuous Internal Evaluation				
		Execution (10)	Observation (5)	Record (10)	Viva (5)	Total (15)

Internal Assessment Number	Write up (10)	Execution (30)	Viva (10)	Total (50)	Scale Down to 10
IA1					
IA2					

CONTENTS

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: Information
Last Name: Science

Full Name: Information 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.".

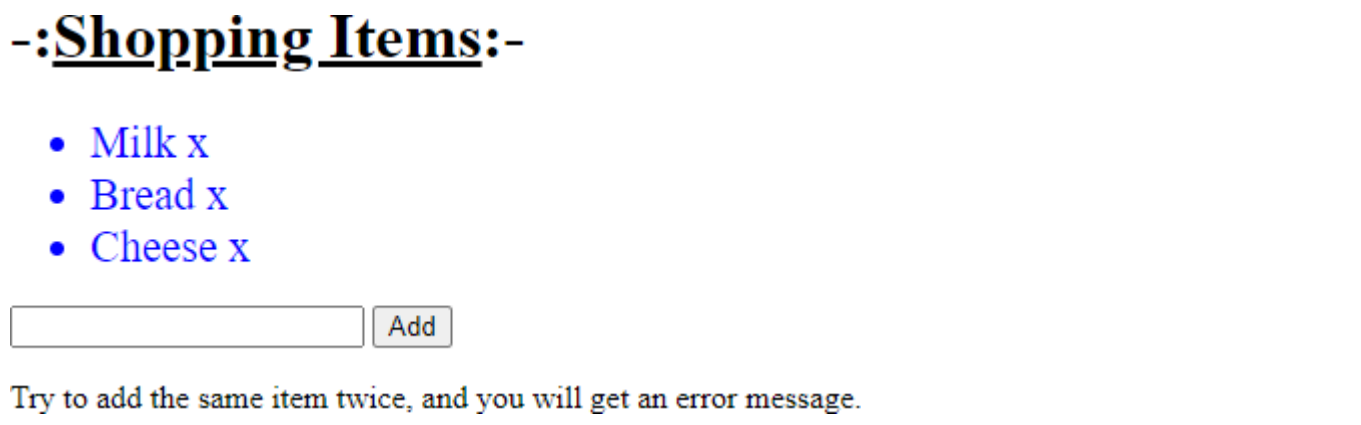
```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>

<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.

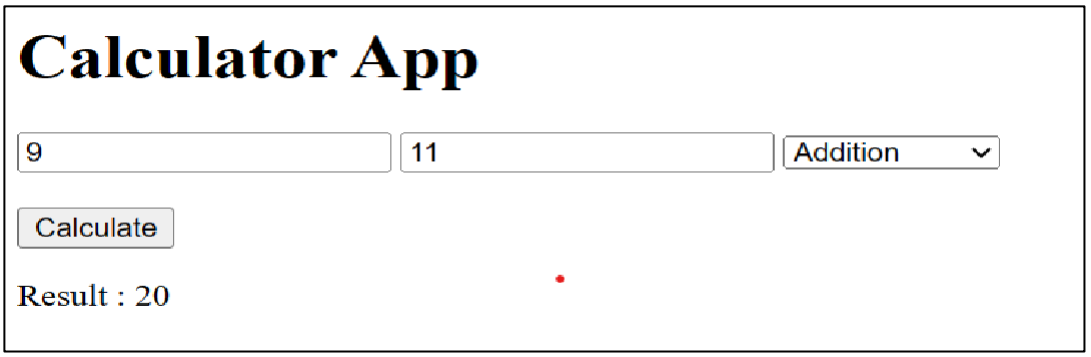
```
<!DOCTYPE html>
<html>
<head>
	<title>Calculator WebApp</title>
	<script
		src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
	</script>
</head>

<body>

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

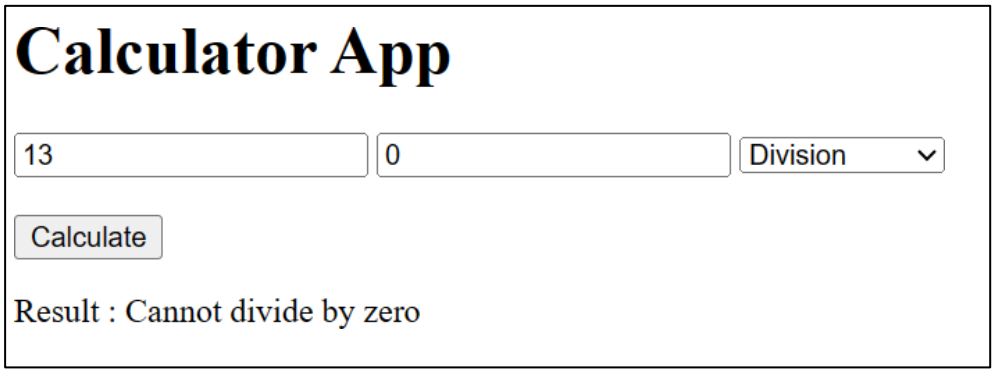
```
    <p>Result : {{result}}</p>
</div>
<script>
  var app = angular.module("calApp", []);
  app.controller("calCtrl", function($scope) {
   $scope.num1 = 0;
   $scope.num2 = 0;
   $scope.operator = "+";
   $scope.result = 0;
   $scope.calculate = function() {
   if($scope.operator==='+')
        $scope.result=$scope.num1 + $scope.num2;
   else if($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:




Calculator App
9
11
Subtraction
Calculate
Result : -2


Calculator App
9
11
Multiplication
Calculate
Result : 99


Calculator App
13
0
Division
Calculate
Result : Cannot divide by zero


Calculator App
216
4
Division
Calculate
Result : 54

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

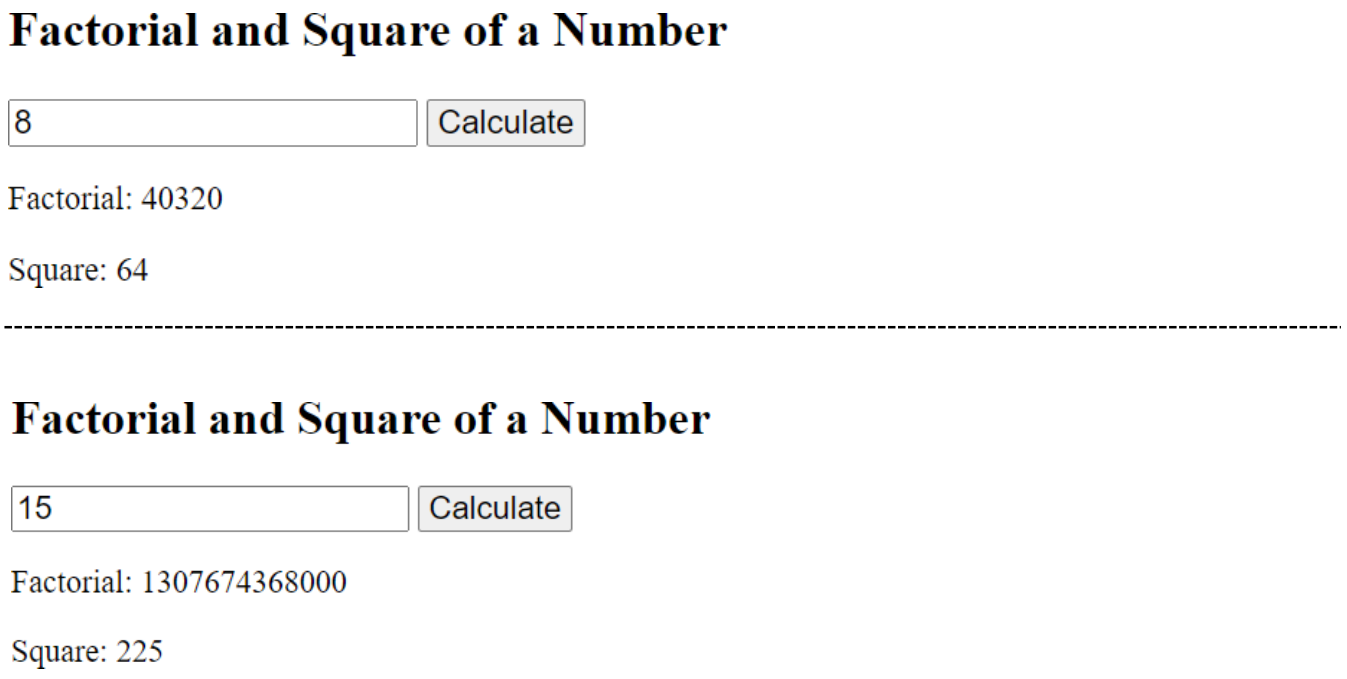
Description: The factorial of a number is the product of all the numbers from 1 to that number. For example, factorial of 3 is equal to 1 * 2 * 3 = 6. The factorial of negative numbers does not exist and the factorial of 0 is 1. Two buttons are included in this AngularJS application: one calculates the 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:



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

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

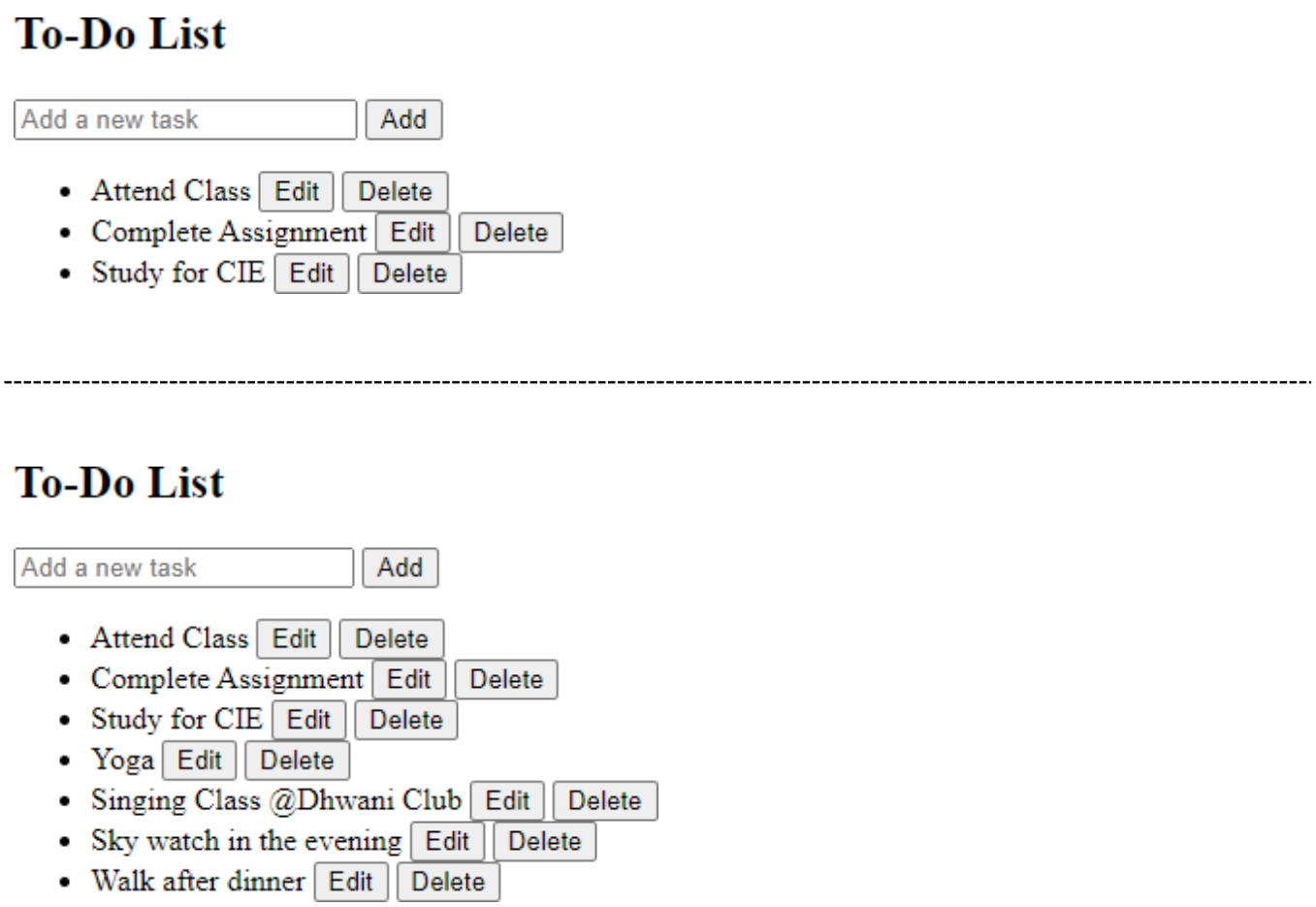
Program 6: Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks. Note: The default values for tasks may be included in the program.

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

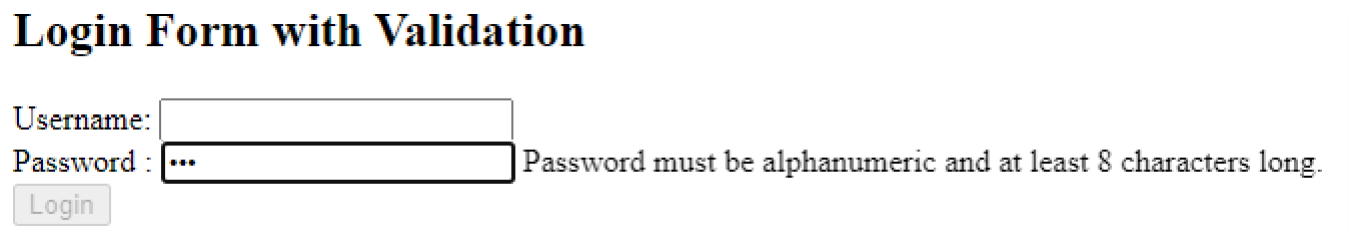
Program 8: Develop AngularJS program to create a login form, with validation for the username and password fields.

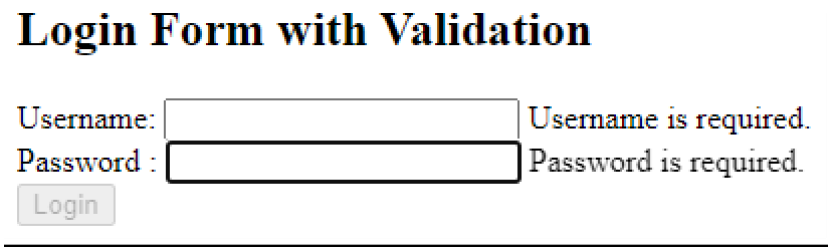
Description: For a login form with validation for the username and password. The password is required to be alphanumeric and 8 characters long. The form uses AngularJS validation with the required attribute for both the username and password fields. The password field also uses the ng-pattern attribute 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.

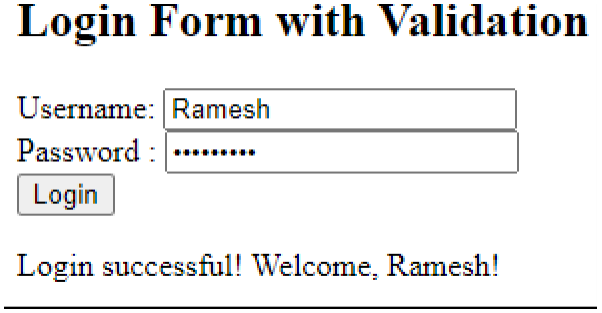
```
<!-- Test regex in https://www.regextester.com/105521 -->
<!DOCTYPE html>
<html>
<head>
  <title>Login Form with Validation</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
  </script>
</head>
<body ng-app="myApp">
<div ng-controller="LoginController">
  <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>
```

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>
```