



Visvesvaraya Technological University  
“Jnana Sangama”, Belagavi-590 018

\*\*\*\*\*

## Fifth Semester B.E. (CSE)

[As per Choice Based Credit System (CBCS) scheme]

(For Internal Circulation Only)

### “Angular JS Laboratory (21CSL581)” Manual

(For Reference Only)

|            |  |
|------------|--|
| Name       |  |
| USN        |  |
| Section    |  |
| Lab Batch  |  |
| Day / Time |  |

\*\*\*\*\*



Kalpataru Institute of Technology, Tiptur - 572 201

Department of Computer Science and Engineering

AY: 2023-2024

| 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  |             |     |
| <b>Course objectives:</b> <ul style="list-style-type: none"><li>To learn the basics of Angular JS framework.</li><li>To understand the Angular JS Modules, Forms, inputs, expression, data bindings and Filters</li><li>To gain experience of modern tool usage (VS Code, Atom or any other] in developing Web applications</li></ul>  |  |             |     |
| Sl.NO  | Experiments  |             |     |
| 1  | Develop Angular JS program that allows user to input their first name and last name and display their full name. <b>Note:</b> 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. <b>Note:</b> 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. <b>Note:</b> 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. <b>Note:</b> 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. <b>Note:</b> 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. <b>Note:</b> The default values for items may be included in the program. |             |     |
| 11   | Create AngularJS application to convert student details to Uppercase using angular filters. <b>Note:</b> 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   |             |     |
| <b>NOTE:</b> Include necessary HTML elements and CSS for the above Angular applications.   |  |             |     |
| <b>Course outcomes (Course Skill Set):</b><br>At the end of the course the student will be able to: <ol style="list-style-type: none"><li>Develop Angular JS programs using basic features</li><li>Develop dynamic Web applications using AngularJS modules</li><li>Make use of form validations and controls for interactive applications</li><li>Apply the concepts of Expressions, data bindings and filters in developing Angular JS programs</li><li>Make use of modern tools to develop Web applications</li></ol> |  |             |     |

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the **maximum** marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE). The student has to secure a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

**Continuous Internal Evaluation (CIE):**

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

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

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8<sup>th</sup> week of the semester and the second test shall be conducted after the 14<sup>th</sup> week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the **maximum** marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

**Semester End Evaluation (SEE):**

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal/external examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, write up -20%, Conduction procedure and result in - 60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
- The duration of SEE is 02 hours

Rubrics suggested in Annexure-II of Regulation book

**Suggested Learning Resources:**

**Textbooks**

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

**Weblinks 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](https://www.youtube.com/watch?v=ICl7_i2mtZA)
5. [https://www.youtube.com/watch?v=Y2Few\\_nkze0](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)

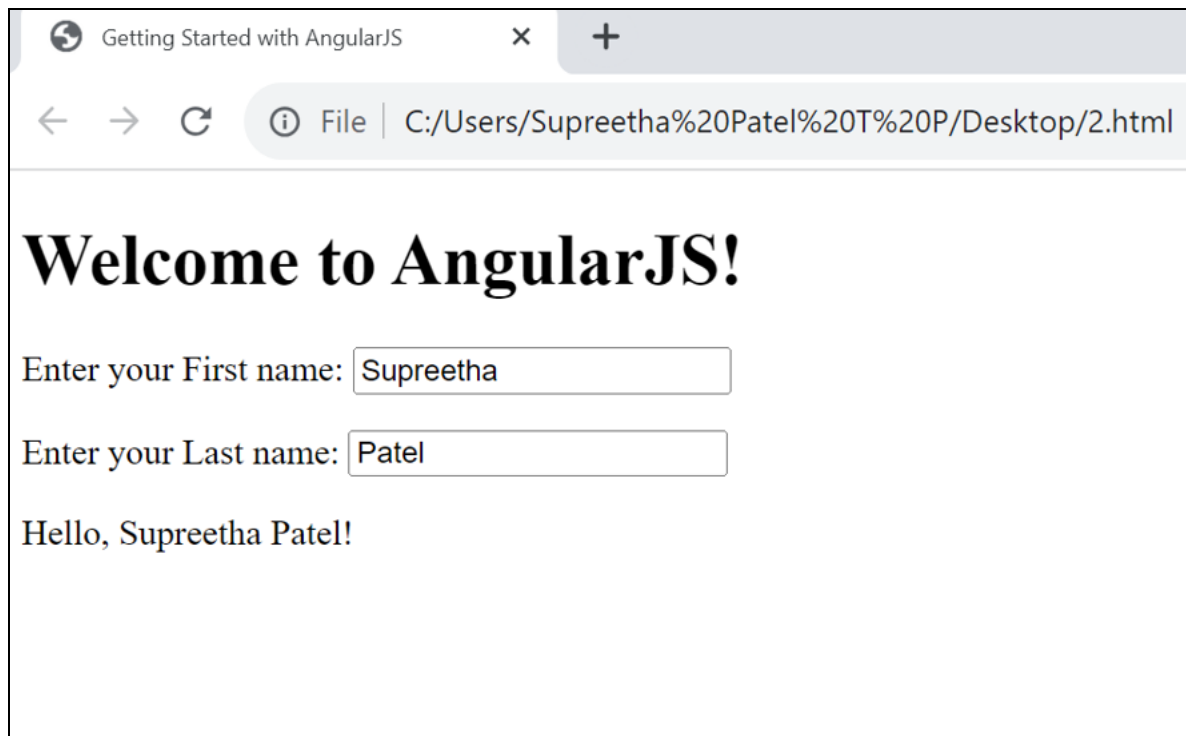
**Lab Program 1: Develop Angular JS program that allows user to input their first name and last name and display their full name.**

**Note: The default values for first name and last name may be included in the program.**

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
<title>Getting Started with AngularJS</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
<div ng-controller="MyCtrl">
<h1>Welcome to AngularJS!</h1>
<p>Enter your First name: <input type="text" ng-model="firstname"></p>
<p>Enter your Last name: <input type="text" ng-model="lastname"></p>
<p>Hello, {{ firstname + ' ' + lastname }}!</p>
</div>

<script> angular.module('myApp', [])
.controller('MyCtrl', function($scope) {
// Initialize scope variables
$scope.firstname = "User";
$scope.lastname = "User";
});
</script>
</body>
</html>
```

## Output



## Explanation:

In this code,

- We have a simple HTML page that uses AngularJS to demonstrate dynamic content binding.
- The **ng-app** attribute defines the AngularJS application module, and the **ng-controller** attribute defines a controller called **MyCtrl**.
- The controller initializes a variable **name** in the scope, and the value is bound to an input field and displayed dynamically on the page.
- When you type your name in the input field, it updates the greeting message below.
- This is a fundamental example to showcase AngularJS's ability to bind data and create dynamic interactions in web applications.
- Students can use this as a starting point to explore AngularJS further.

**Lab 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.**

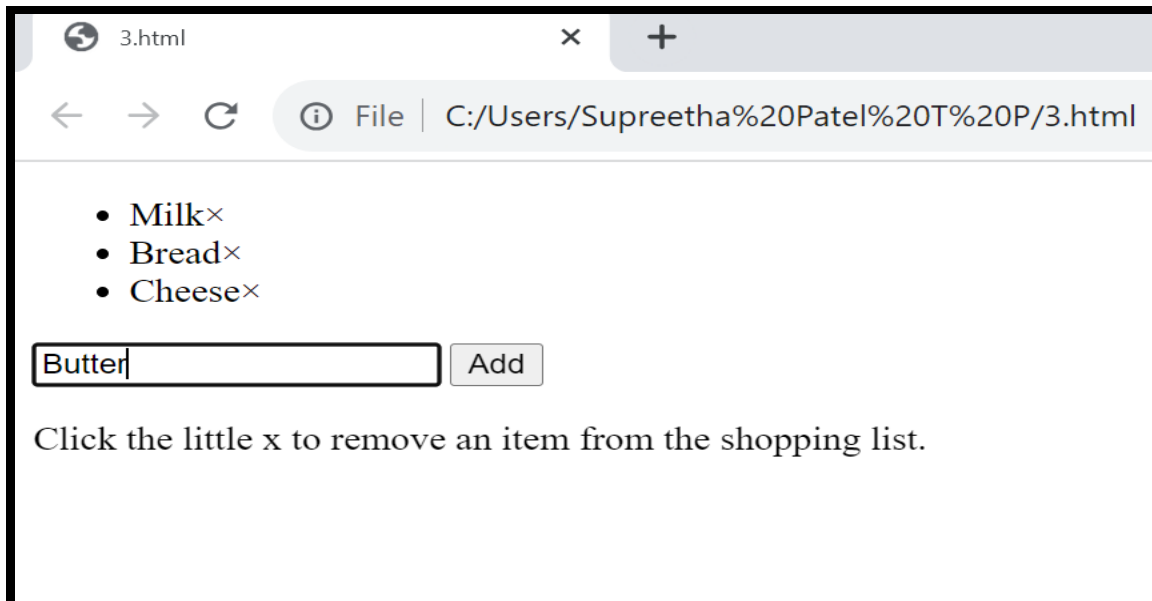
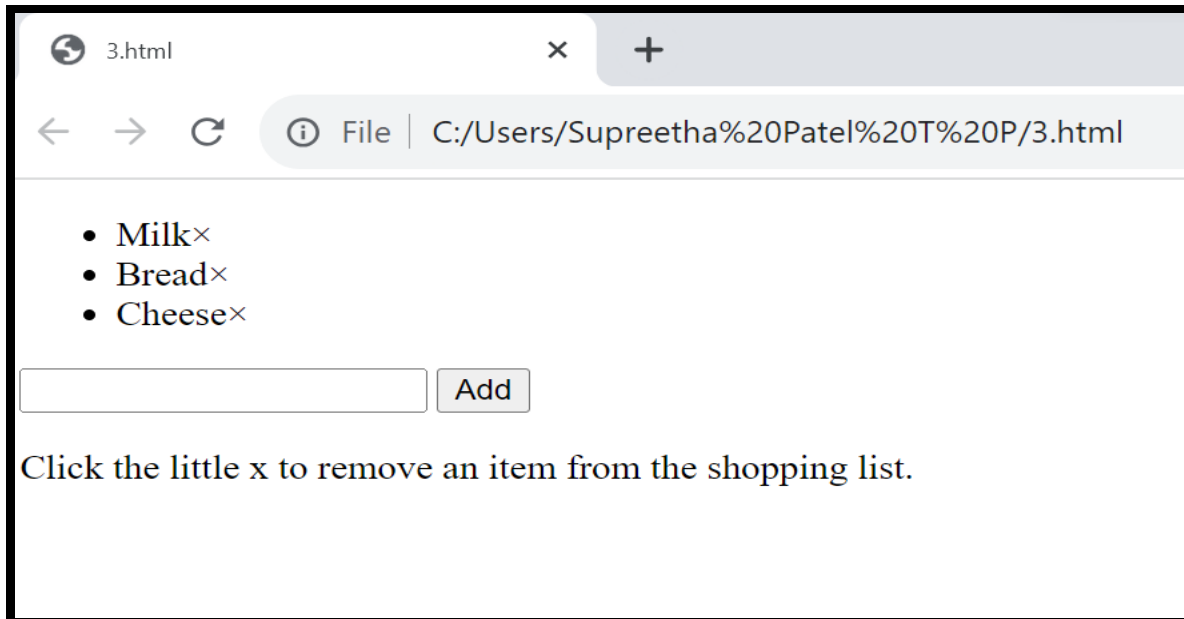
**Note: The default values of items may be included in the program.**

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope) {
    $scope.products = ["Milk", "Bread", "Cheese"];
    $scope.addItem = function () {
        $scope.products.push($scope.addMe);
    }
    $scope.removeItem = function (x) {
        $scope.products.splice(x, 1);
    }
});
</script>

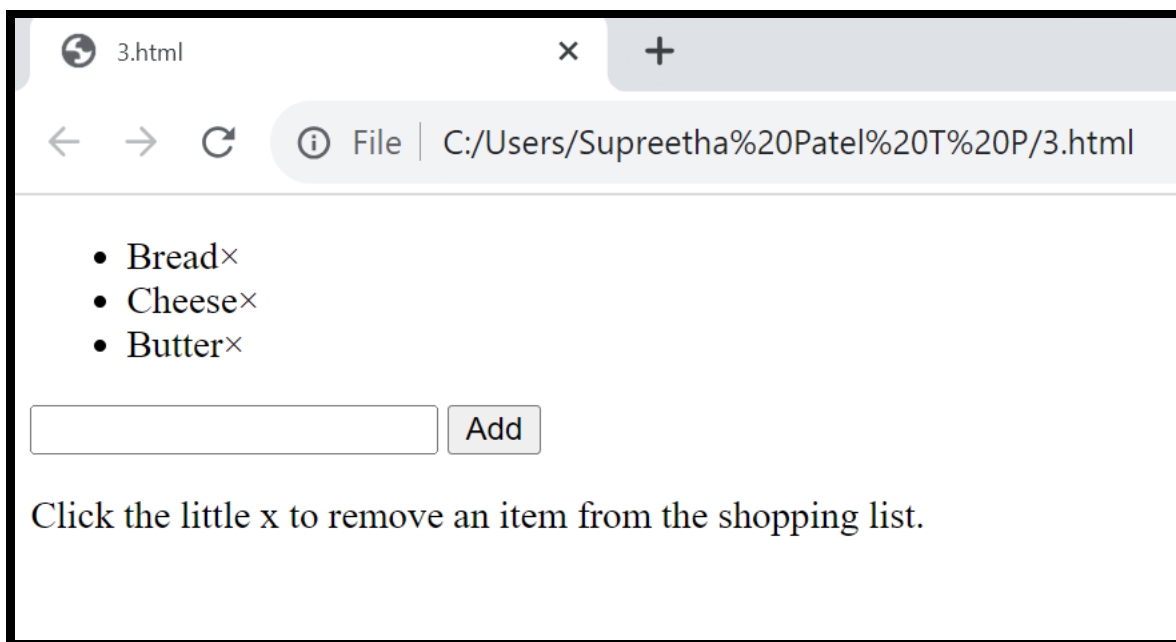
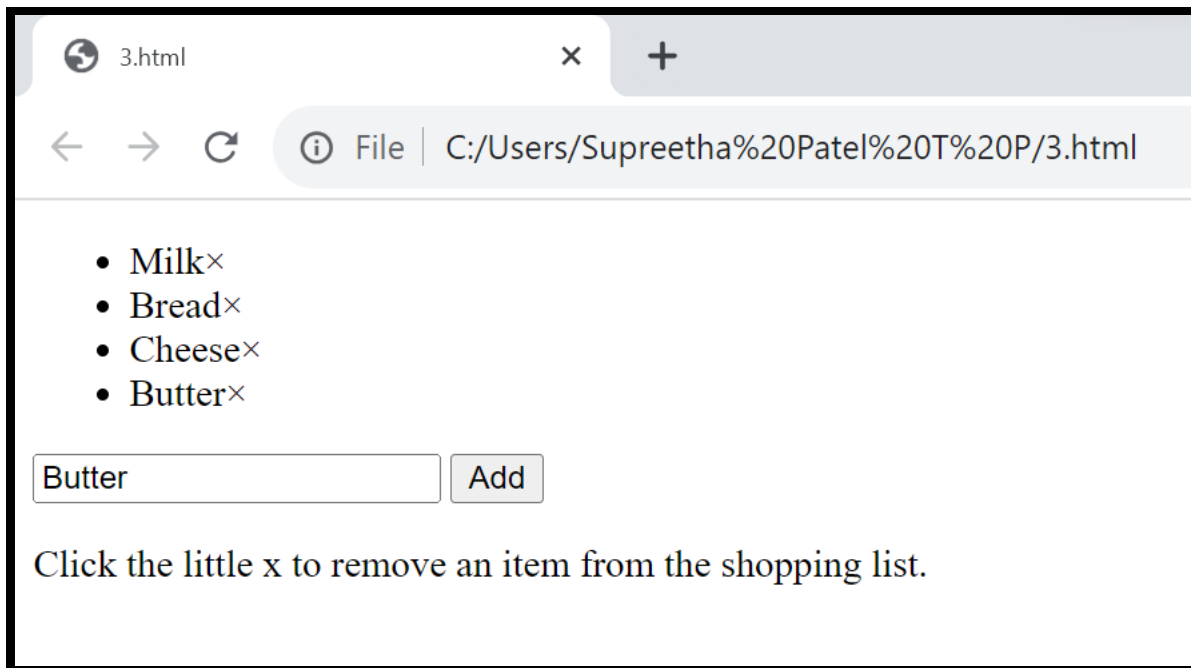
<div ng-app="myShoppingList" ng-controller="myCtrl">
    <ul>
        <li ng-repeat="x in products">{{ x }}<span ng-click="removeItem($index)">×</span></li>
    </ul>
    <input ng-model="addMe">
    <button ng-click="addItem()">Add</button>
</div>

<p>Click the little x to remove an item from the shopping list.</p>
</body>
</html>
```

## Output



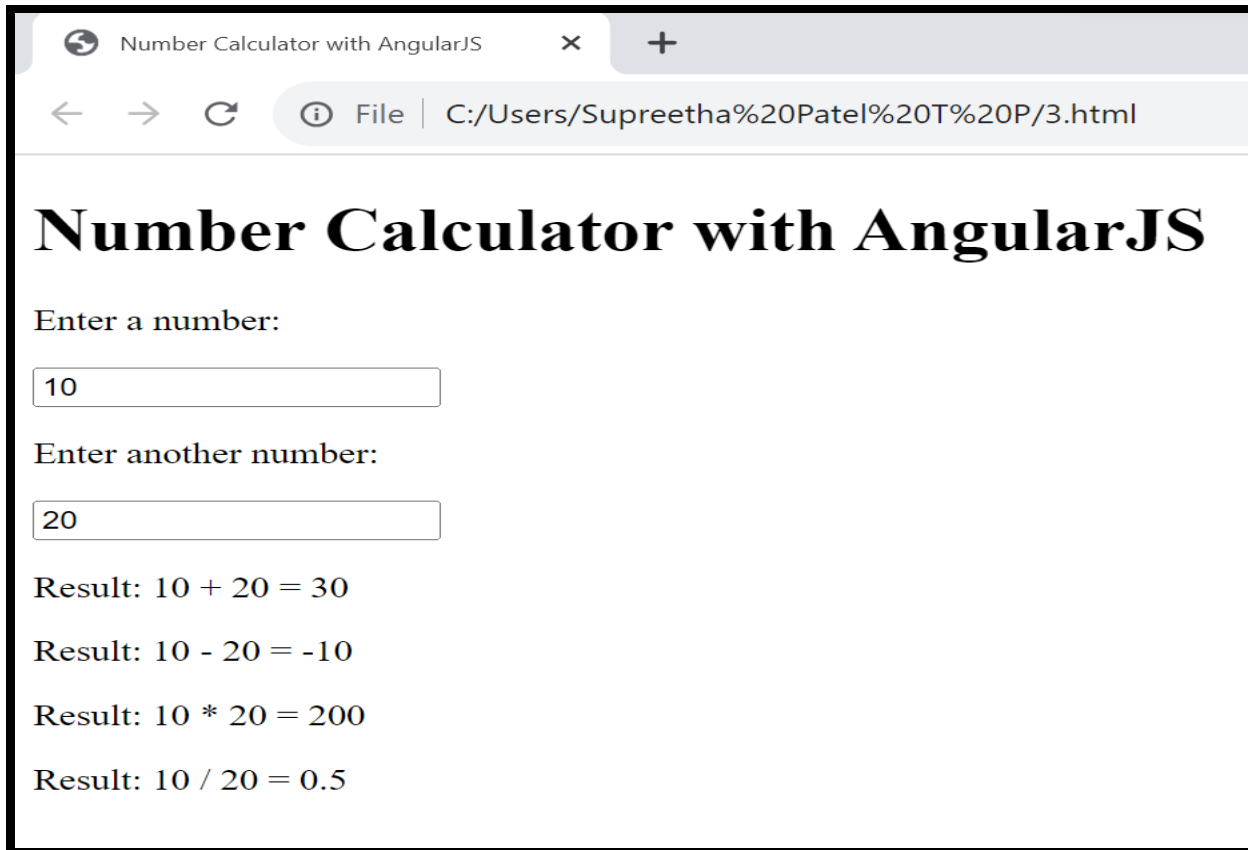




**Lab Program 3: Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input.**

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
<title>Number Calculator with AngularJS</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
<div ng-controller="CalculatorCtrl">
<h1>Number Calculator with AngularJS</h1>
<p>Enter a number:</p>
<input type="number" ng-model="number1">
<p>Enter another number:</p>
<input type="number" ng-model="number2">
<p>Result: {{ number1 }} + {{ number2 }} = {{ number1 + number2 }}</p>
<p>Result: {{ number1 }} - {{ number2 }} = {{ number1 - number2 }}</p>
<p>Result: {{ number1 }} * {{ number2 }} = {{ number1 * number2 }}</p>
<p>Result: {{ number1 }} / {{ number2 }} = {{ number1 / number2 }}</p>
</div>
<script> angular.module('myApp', [])
.controller('CalculatorCtrl', function($scope) {
$scope.number1 = 0;
$scope.number2 = 0;
});
</script>
</body>
</html>
```

## Output



Number Calculator with AngularJS

Enter a number:

10

Enter another number:

20

Result:  $10 + 20 = 30$

Result:  $10 - 20 = -10$

Result:  $10 * 20 = 200$

Result:  $10 / 20 = 0.5$

### Explanation:

In this program, we:

- Create a simple calculator that allows you to enter two numbers using input fields with ng-model.
- Display the results of addition and subtraction using AngularJS expressions: `{{ number1 + number2 }}`, `{{ number1 - number2 }}`, `{{ number1 * number2 }}` and `{{ number1 / number2 }}`
- The program initializes both input fields to 0 when the page loads.

You can enter numbers in the input fields, and the program will instantly display the results of addition and subtraction. This example illustrates how AngularJS can be used for simple calculations and dynamic content updates on a web page.

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

```
<!DOCTYPE html>
<html ng-app="calculatorApp">

<head>
  <title>AngularJS Calculator</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    .calculator {
      max-width: 400px;
      margin: 50px auto;
      padding: 20px;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    input {
      width: 100%;
      margin-bottom: 10px;
      padding: 8px;
      box-sizing: border-box;
    }

    button {
      width: 100%;
      padding: 10px;
      background-color: #4CAF50;
```

```
        color: white;
        border: none;
        border-radius: 3px;
        cursor: pointer;
    }

    button:hover {
        background-color: #45a049;
    }
</style>
</head>

<body ng-controller="calculatorController">

    <div class="calculator">
        <h2>AngularJS Calculator</h2>

        <label for="number">Enter a number:</label>
        <input type="number" id="number" ng-model="inputNumber" placeholder="Enter a number" />

        <button ng-click="calculateFactorial()">Calculate Factorial</button>
        <p>Factorial: {{ factorialResult }}</p>

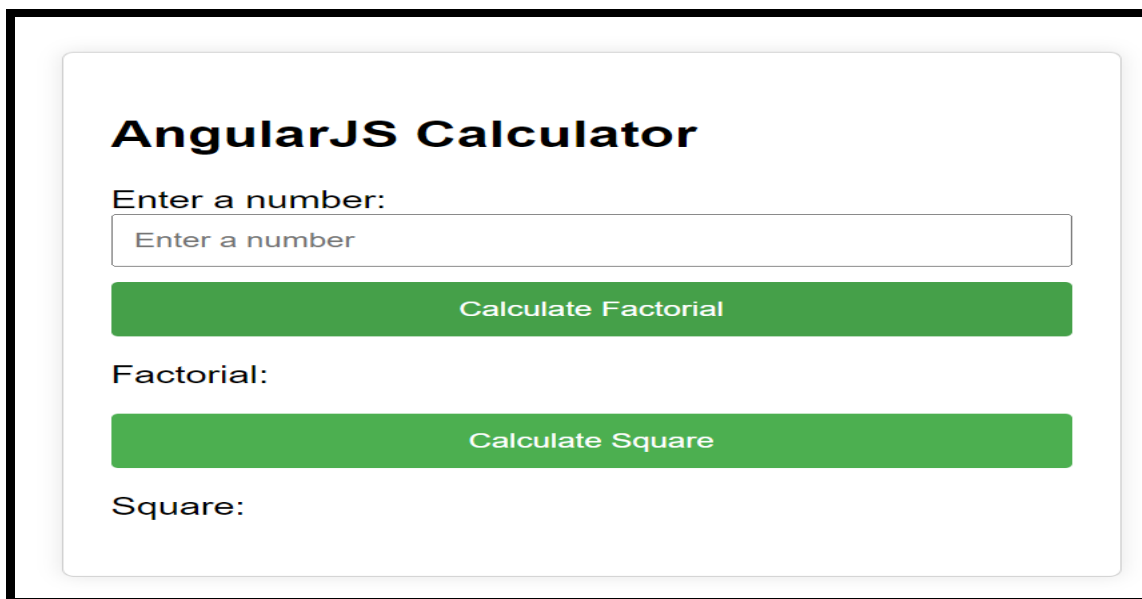
        <button ng-click="calculateSquare()">Calculate Square</button>
        <p>Square: {{ squareResult }}</p>
    </div>

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

        app.controller('calculatorController', function ($scope) {
            $scope.calculateFactorial = function () {
                var result = 1;
```

```
        for (var i = 1; i <= $scope.inputNumber; i++) {  
            result *= i;  
        }  
        $scope.factorialResult = result;  
    };  
  
    $scope.calculateSquare = function () {  
        $scope.squareResult = Math.pow($scope.inputNumber, 2);  
    };  
});  
</script>  
  
</body>  
  
</html>
```

## Output



The screenshot displays a web application titled "AngularJS Calculator". It features a text input field with the placeholder text "Enter a number". Below the input field is a green button labeled "Calculate Factorial". Underneath this button is the label "Factorial:". Below the "Factorial:" label is another green button labeled "Calculate Square". At the bottom of the interface is the label "Square:". The entire application is enclosed in a black border.

## AngularJS Calculator

Enter a number:

Calculate Factorial

Factorial: 24

Calculate Square

Square: 16

**Lab 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.**

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS 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">
  <h2>Student Details</h2>

  <label for="numberOfStudents">Enter the number of students:</label>
  <input type="number" id="numberOfStudents" ng-model="numberOfStudents" />
  <button ng-click="displayStudentDetails()">Display Student Details</button>

  <div ng-show="studentDetails.length > 0">
    <p>Total Students: {{ studentDetails.length }}</p>

    <table border="1">
      <tr>
        <th>Student Name</th>
        <th>CGPA</th>
      </tr>
      <tr ng-repeat="student in studentDetails">
        <td>{{ student.name }}</td>
        <td>{{ student.cgpa }}</td>
      </tr>
    </table>
  </div>
</div>
```



</div>

</div>

<script>

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

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

```
    $scope.studentDetails = [];
```

```
    $scope.displayStudentDetails = function () {
```

```
        $scope.studentDetails = [];
```

```
        for (var i = 0; i < $scope.numberOfStudents; i++) {
```

```
            var student = {
```

```
                name: prompt('Enter the name of student ' + (i + 1)),
```

```
                cgpa: parseFloat(prompt('Enter the CGPA of student ' + (i + 1)))
```

```
            };
```

```
            $scope.studentDetails.push(student);
```

```
        }
```

```
    };
```

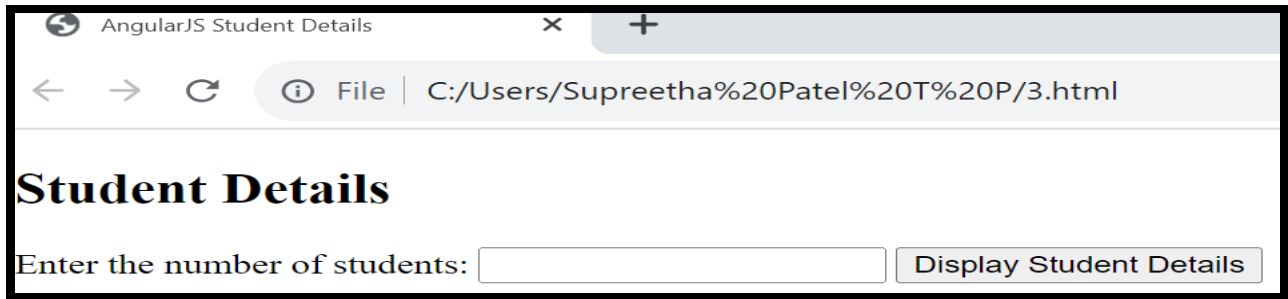
```
});
```

</script>

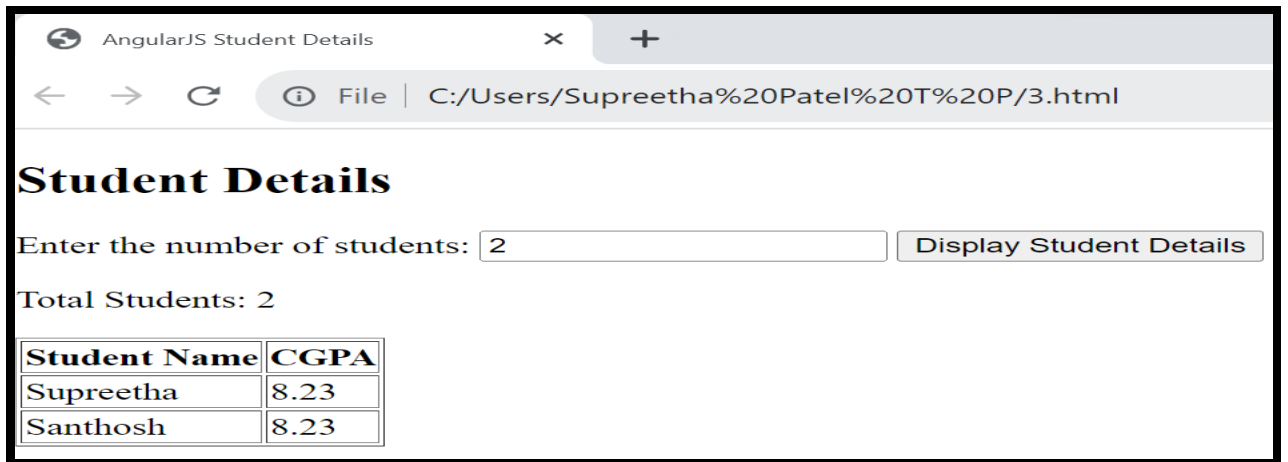
</body>

</html>

## Output



A browser window titled "AngularJS Student Details" shows a web page with the title "Student Details". Below the title, there is a label "Enter the number of students:" followed by an empty text input field and a button labeled "Display Student Details". The browser's address bar shows the file path "C:/Users/Supreetha%20Patel%20T%20P/3.html".



The same browser window now displays the results. The input field contains the number "2". Below the input field, the text "Total Students: 2" is shown. A table with two columns, "Student Name" and "CGPA", lists the details of two students: Supreetha with a CGPA of 8.23, and Santhosh with a CGPA of 8.23. The "Display Student Details" button remains visible.

| Student Name | CGPA |
|--------------|------|
| Supreetha    | 8.23 |
| Santhosh     | 8.23 |

**Lab Program 6: Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks.**

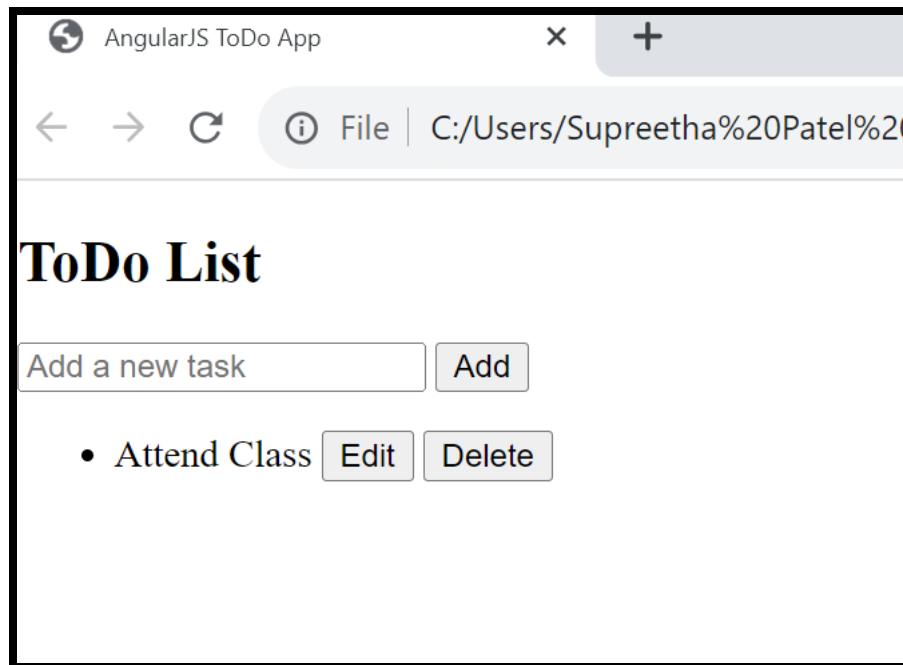
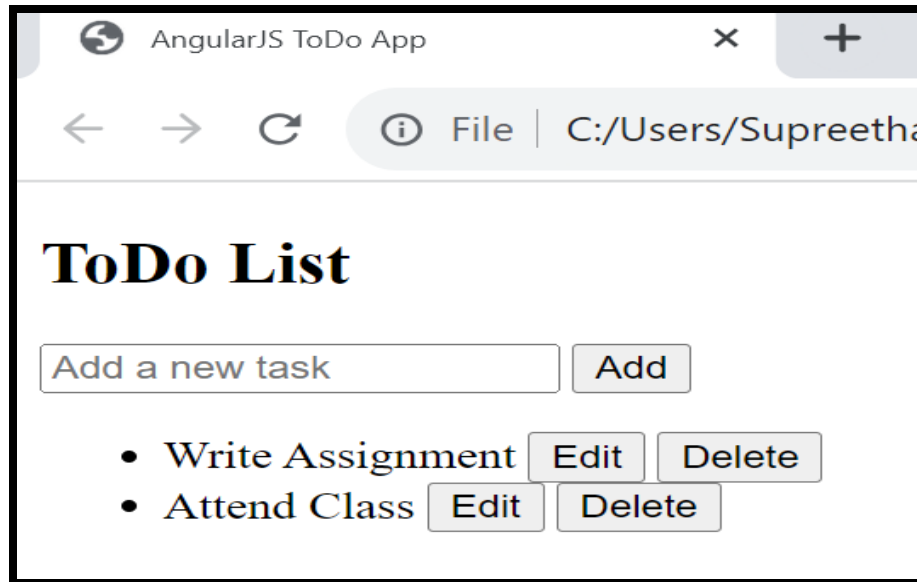
**Note: The default values for tasks may be included in the program.**

```
<!DOCTYPE html>
<html ng-app="todoApp">

<head>
  <title>AngularJS ToDo App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="TodoController">
  <h2>ToDo List</h2>
  <form ng-submit="addTask()">
    <input type="text" ng-model="newTask" placeholder="Add a new task" required />
    <button type="submit">Add</button>
  </form>
  <ul>
    <li ng-repeat="task in tasks">
      <span>{{ task.name }}</span>
      <button ng-click="editTask(task)">Edit</button>
      <button ng-click="deleteTask(task)">Delete</button>
    </li>
  </ul>
  <div ng-show="editingTask">
    <h3>Edit Task</h3>
    <form ng-submit="updateTask()">
      <input type="text" ng-model="editingTask.name" required />
      <button type="submit">Update</button>
      <button type="button" ng-click="cancelEdit()">Cancel</button>
    </form>
  </div>
</script>
```

```
var app = angular.module('todoApp', []);
app.controller('TodoController', function ($scope) {
    $scope.tasks = [];
    $scope.newTask = "";
    $scope.editingTask = null;
    $scope.addTask = function () {
        if ($scope.newTask.trim() !== "") {
            $scope.tasks.push({ name: $scope.newTask });
            $scope.newTask = "";
        }
    };
    $scope.editTask = function (task) {
        $scope.editingTask = angular.copy(task);
    };
    $scope.updateTask = function () {
        var index = $scope.tasks.indexOf($scope.editingTask);
        $scope.tasks[index] = $scope.editingTask;
        $scope.editingTask = null;
    };
    $scope.cancelEdit = function () {
        $scope.editingTask = null;
    };
    $scope.deleteTask = function (task) {
        var index = $scope.tasks.indexOf(task);
        $scope.tasks.splice(index, 1);
    };
});
</script>
</body>
</html>
```

## Output



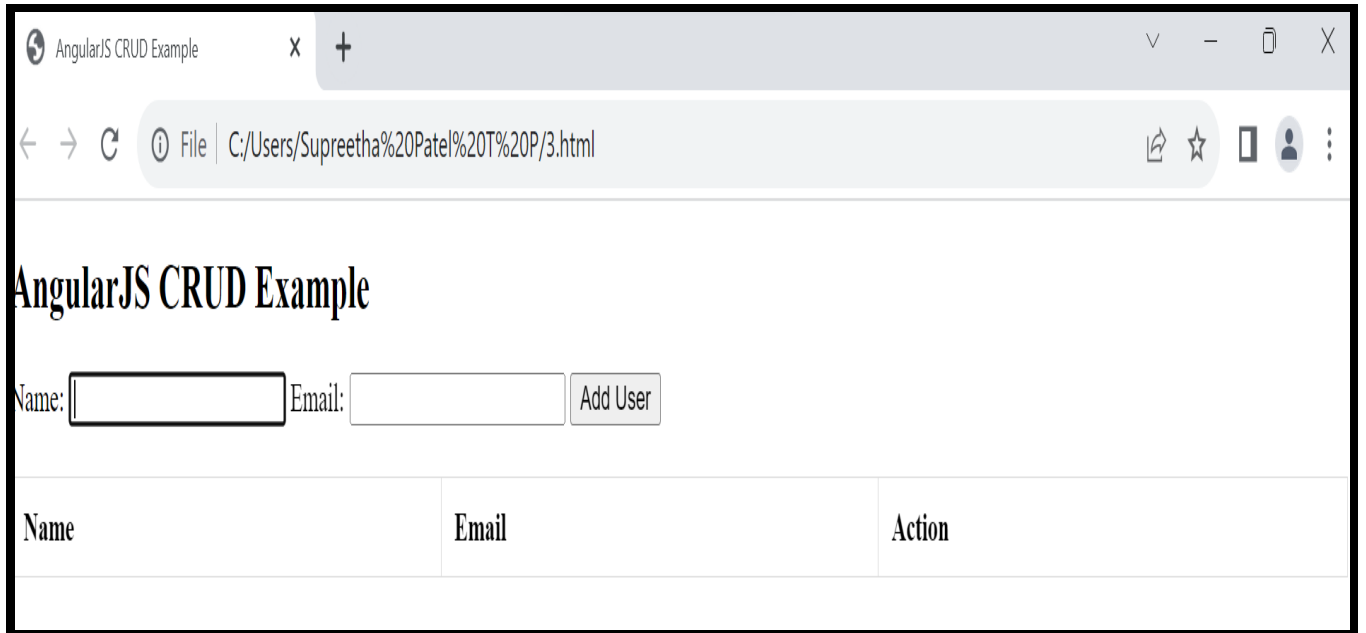
**Lab Program 7: Write an AngularJS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.**

```
<!DOCTYPE html>
<html ng-app="crudApp">
<head>
  <title>AngularJS CRUD Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
  <style>
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    table, th, td {
      border: 1px solid #ddd;
    }
    th, td {
      padding: 10px;
      text-align: left;
    }
  </style>
</head>
<body ng-controller="crudController">
  <h2>AngularJS CRUD Example</h2>
  <form>
    <label>Name:</label>
    <input type="text" ng-model="newUser.name" required>
    <label>Email:</label>
    <input type="email" ng-model="newUser.email" required>
    <button ng-click="addUser()">Add User</button>
  </form>
```

```
<table>
  <tr>
    <th>Name</th>
    <th>Email</th>
    <th>Action</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>
<script>
var app = angular.module('crudApp', []);
app.controller('crudController', function ($scope) {
  $scope.users = [];
  $scope.newUser = { };
  $scope.addUser = function () {
    $scope.users.push(angular.copy($scope.newUser));
    $scope.newUser = { };
  };
  $scope.editUser = function (user) {
    $scope.newUser = angular.copy(user);
    $scope.deleteUser(user);
  };
  $scope.deleteUser = function (user) {
    var index = $scope.users.indexOf(user);
    $scope.users.splice(index, 1);
  };
});
```

```
</script>
</body>
</html>
```

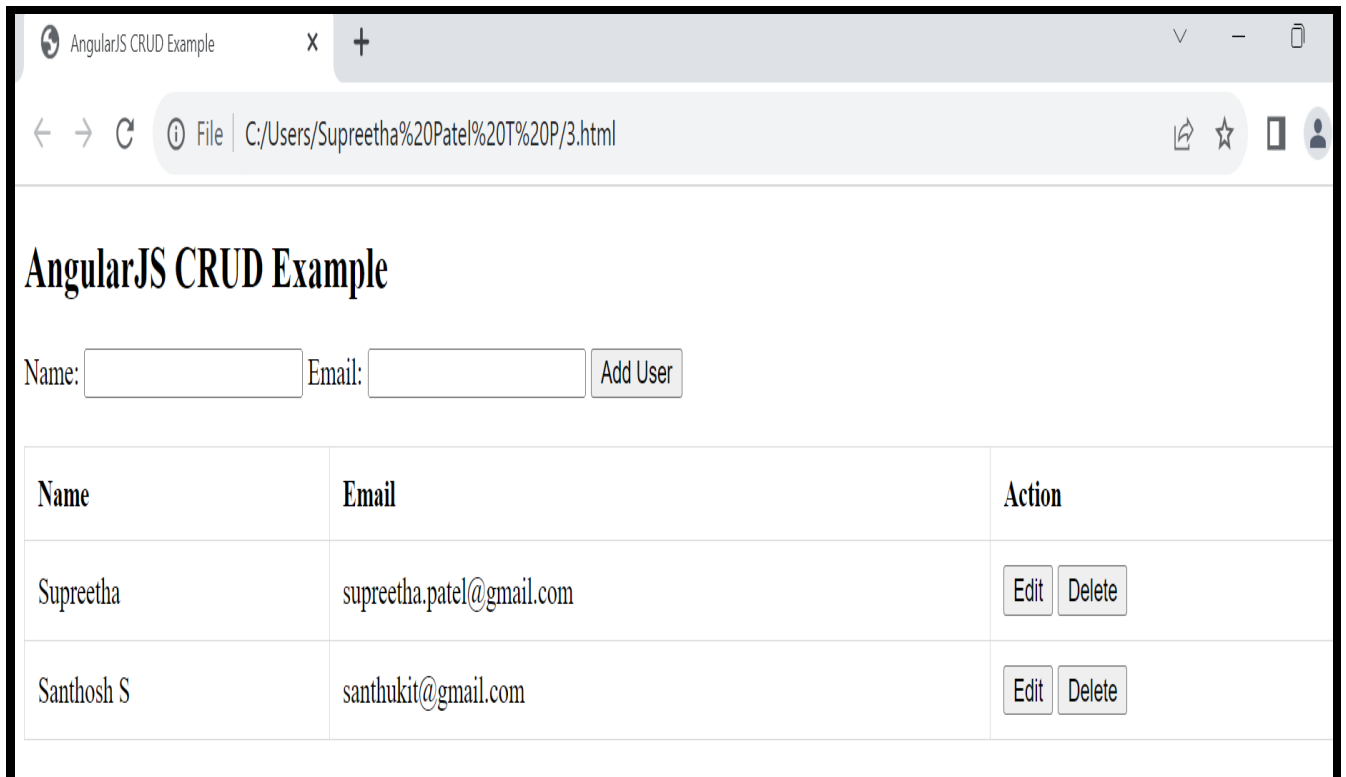
## Output



AngularJS CRUD Example

Name:  Email:

| Name | Email | Action |
|------|-------|--------|
|------|-------|--------|



AngularJS CRUD Example

Name:  Email:

| Name       | Email                     | Action  |
|------------|---------------------------|---|
| Supreetha  | supreetha.patel@gmail.com | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |
| Santhosh S | santhukit@gmail.com       | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |



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

```
<!DOCTYPE html>
<html lang="en" ng-app="loginApp">
<head>
  <meta charset="UTF-8">
  <title>Login Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error {
      color: red;
    }
  </style>
</head>
<body>
<div ng-controller="loginController as loginCtrl">
  <h2>Login Form</h2>
  <form name="loginForm" ng-submit="loginCtrl.submitForm()" novalidate>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" ng-model="loginCtrl.user.username"
required>
    <span class="error" ng-show="loginForm.username.$dirty && loginForm.username.$invalid">
      <span ng-show="loginForm.username.$error.required">Username is required.</span>
    </span>
    <br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" ng-model="loginCtrl.user.password"
required>
    <span class="error" ng-show="loginForm.password.$dirty && loginForm.password.$invalid">
      <span ng-show="loginForm.password.$error.required">Password is required.</span>
    </span>
    <br>
    <button type="submit" ng-disabled="loginForm.$invalid">Login</button>
```

```
</form>
</div>
<script>
  angular.module('loginApp', [])
    .controller('loginController', function () {
      var vm = this;
      vm.user = {};
      vm.submitForm = function () {
        // Perform login logic here
        console.log('Logging in:', vm.user);
      };
    });
</script>
</body>
</html>
```

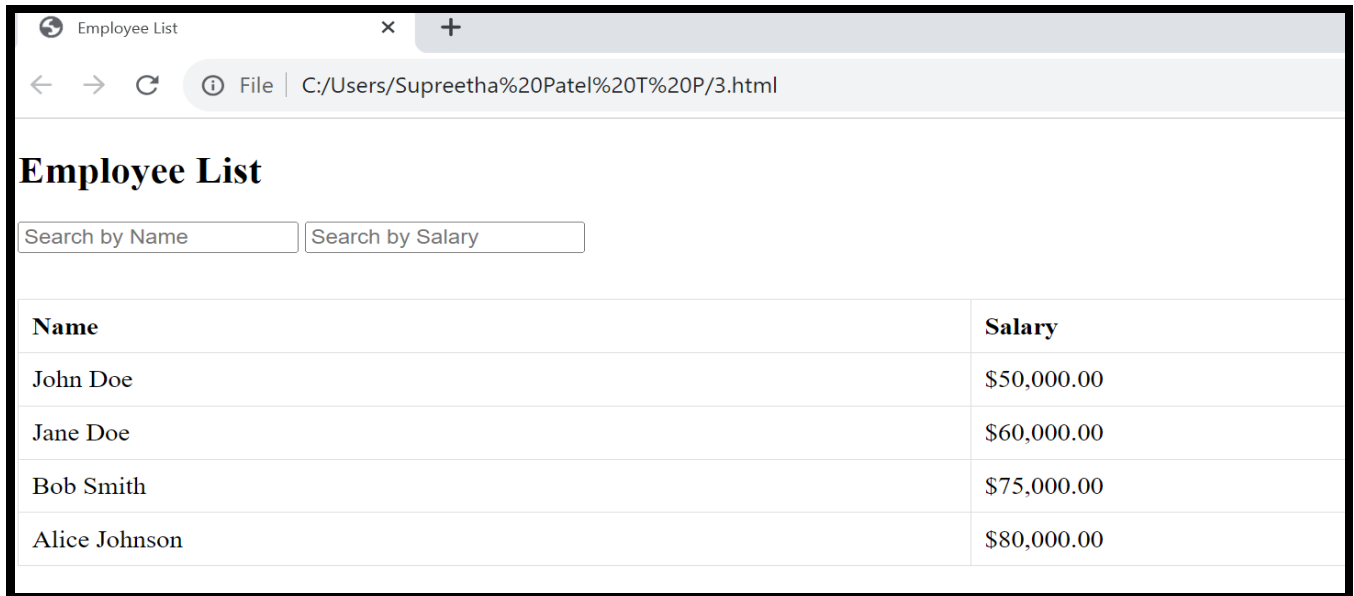
**Lab Program 9: Create an AngularJS application that displays a list of employees and their salaries. Allow users to search for employees by name and salary.**

**Note: Employee details may be included in the program.**

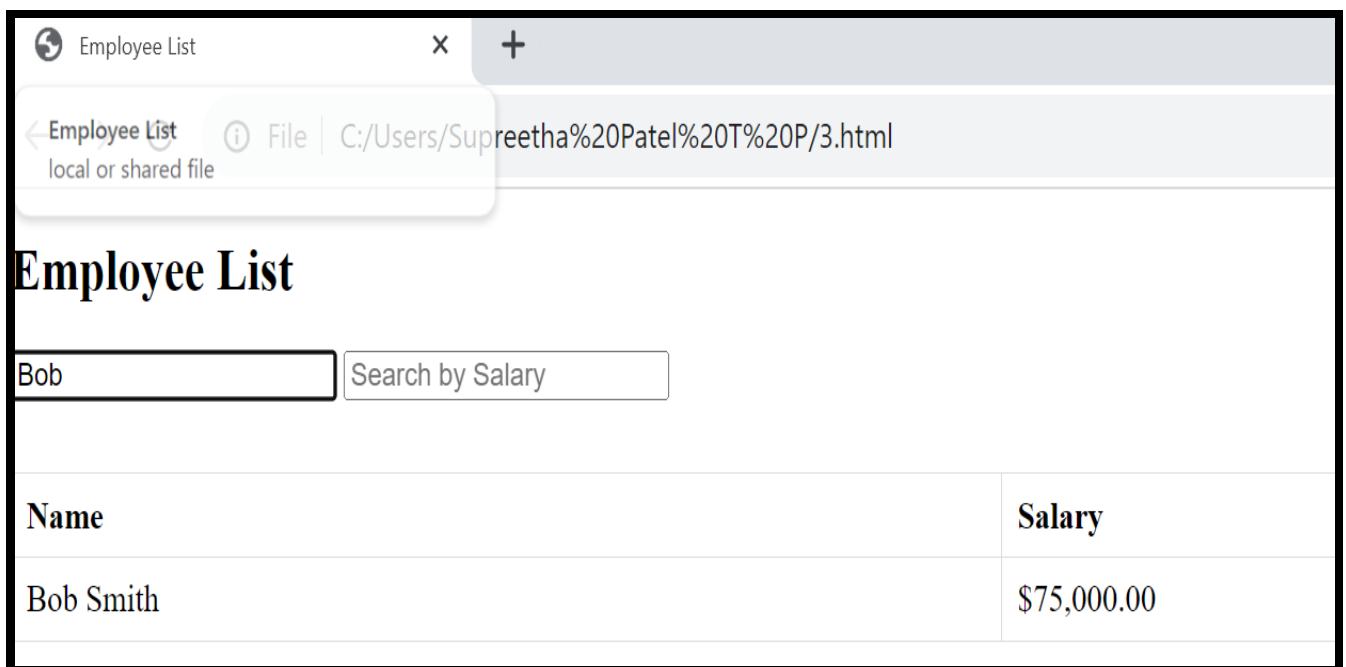
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Employee List</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    th, td {
      border: 1px solid #ddd;
      padding: 8px;
      text-align: left;
    }
    input {
      margin-bottom: 10px;
    }
  </style>
</head>
<body>
  <div ng-app="employeeApp" ng-controller="employeeCtrl">
    <h2>Employee List</h2>
    <input type="text" ng-model="searchName" placeholder="Search by Name">
    <input type="number" ng-model="searchSalary" placeholder="Search by Salary">
    <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>
</div>
<script>
var app = angular.module('employeeApp', []);
app.controller('employeeCtrl', function ($scope) {
  $scope.employees = [
    { name: 'John Doe', salary: 50000 },
    { name: 'Jane Doe', salary: 60000 },
    { name: 'Bob Smith', salary: 75000 },
    { name: 'Alice Johnson', salary: 80000 },
    // Add more employees as needed
  ];
});
</script>
</body>
</html>
```

## Output



| Name          | Salary      |
|---------------|-------------|
| John Doe      | \$50,000.00 |
| Jane Doe      | \$60,000.00 |
| Bob Smith     | \$75,000.00 |
| Alice Johnson | \$80,000.00 |



| Name      | Salary      |
|-----------|-------------|
| Bob Smith | \$75,000.00 |

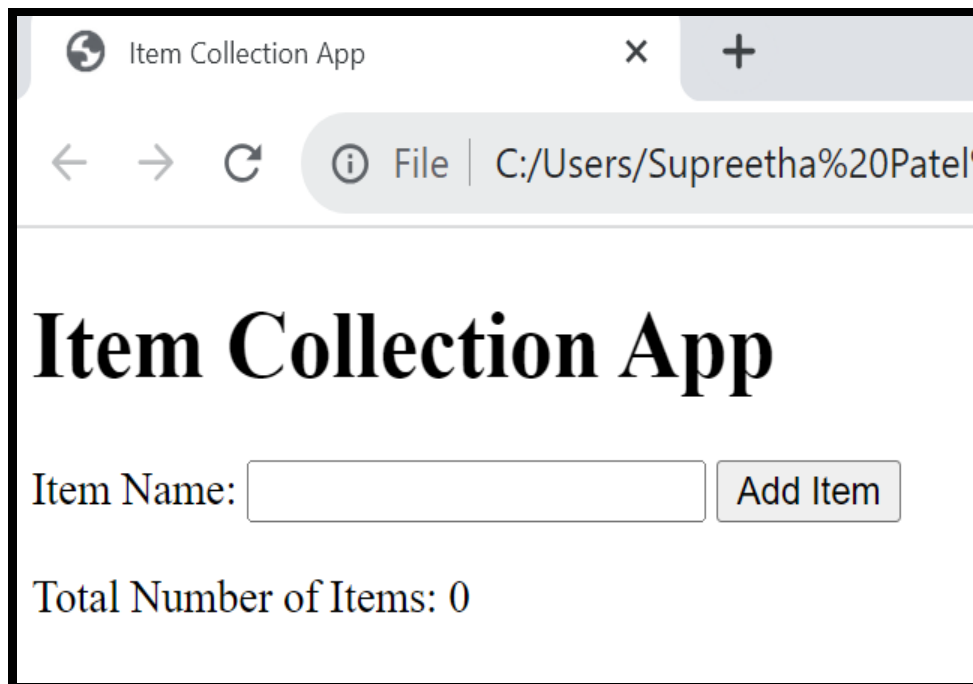
**Lab Program 10: Create AngularJS application that allows users to maintain a collection of items. The application should display the current total number of items, and this count should automatically update as items are added or removed. Users should be able to add items to the collection and remove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

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

```
<!DOCTYPE html>
<html ng-app="itemApp">
<head>
  <title>Item Collection App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body ng-controller="ItemController">
  <h1>Item Collection App</h1>
  <form ng-submit="addItem()">
    <label for="itemName">Item Name:</label>
    <input type="text" ng-model="newItem" required>
    <button type="submit">Add Item</button>
  </form>
  <ul>
    <li ng-repeat="item in items">
      {{ item }}
      <button ng-click="removeItem($index)">Remove</button>
    </li>
  </ul>
  <p>Total Number of Items: {{ items.length }}</p>
  <script>
    var app = angular.module('itemApp', []);
    app.controller('ItemController', function ($scope) {
      $scope.items = [];
      $scope.newItem = "";
      $scope.addItem = function () {
        if ($scope.newItem) {
```

```
$scope.items.push($scope.newItem);  
$scope.newItem = "";  
}  
};  
$scope.removeItem = function (index) {  
    $scope.items.splice(index, 1);  
};  
});  
</script>  
</body>  
</html>
```

## Output



Item Collection App

×

+

← → ↻ ⓘ File | C:/Users/Supreetha%20Patel%20T

# Item Collection App

Item Name:

- Item 1
- Item 2
- Item 3

Total Number of Items: 3

Item Collection App

×

+

← → ↻ ⓘ File | C:/Users/Supreetha%20Patel%20T

# Item Collection App

Item Name:

- Item 2
- Item 3

Total Number of Items: 2

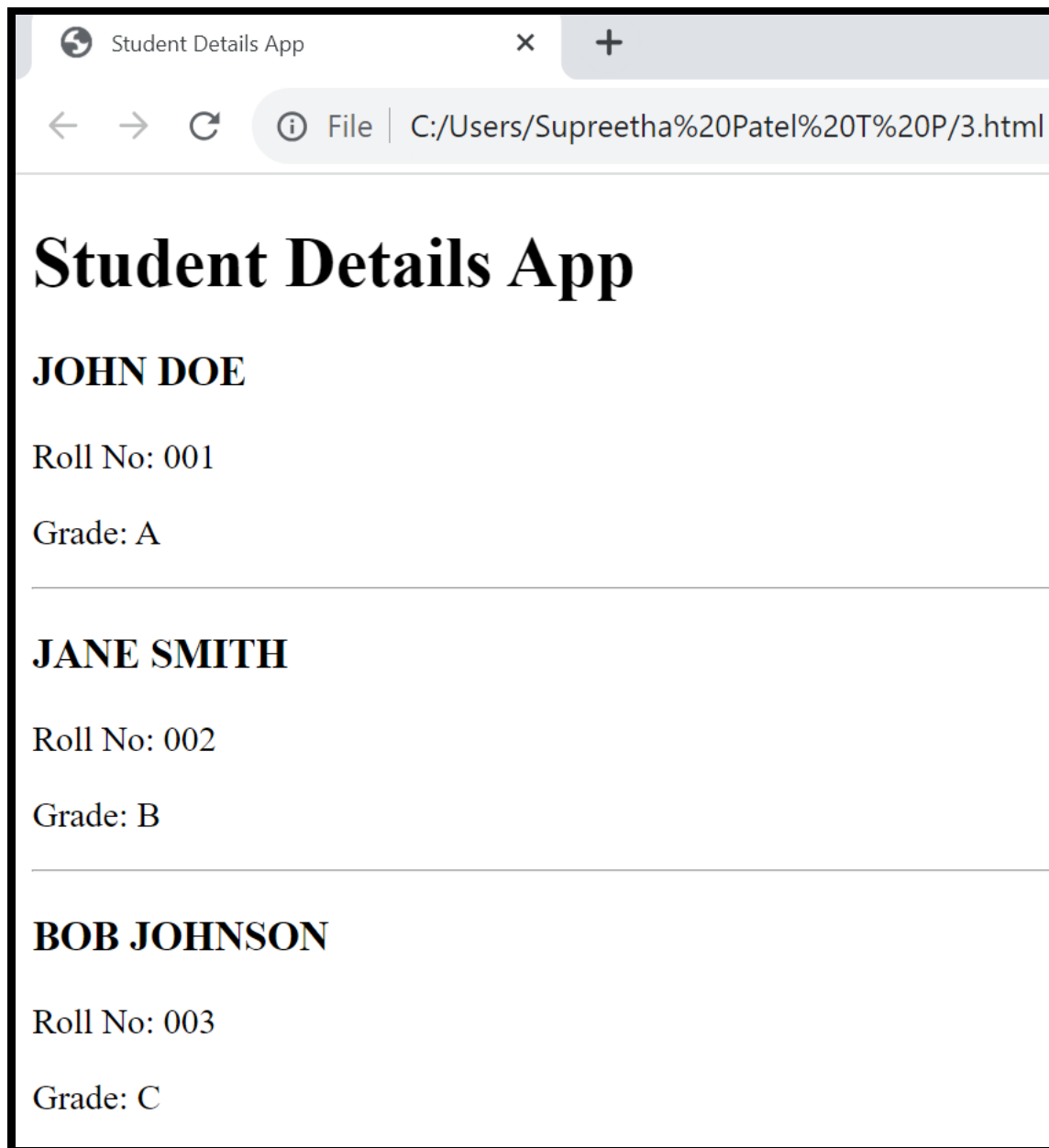


**Lab Program 11: Create AngularJS application to convert student details to Uppercase using angular filters.**

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

```
<!DOCTYPE html>
<html ng-app="studentApp">
<head>
  <title>Student Details App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body ng-controller="StudentController">
  <h1>Student Details App</h1>
  <div ng-repeat="student in students">
    <h3>{{ student.name | uppercase }}</h3>
    <p>Roll No: {{ student.rollNo }}</p>
    <p>Grade: {{ student.grade }}</p>
    <hr>
  </div>
  <script>
    var app = angular.module('studentApp', []);
    app.controller('StudentController', function ($scope) {
      $scope.students = [
        { name: 'John Doe', rollNo: '001', grade: 'A' },
        { name: 'Jane Smith', rollNo: '002', grade: 'B' },
        { name: 'Bob Johnson', rollNo: '003', grade: 'C' }
      ];
    });
  </script>
</body>
</html>
```

## Output



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

```
<!DOCTYPE html>
<html ng-app="dateApp">
<head>
  <title>Date Display App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body ng-controller="DateController">
  <h1>Date Display App</h1>
  <p>Current Date: {{ currentDate }}</p>
  <p>Formatted Date (short): {{ currentDate | date:'short' }}</p>
  <p>Formatted Date (medium): {{ currentDate | date:'medium' }}</p>
  <p>Formatted Date (long): {{ currentDate | date:'long' }}</p>
  <p>Formatted Date (yyyy-MM-dd): {{ currentDate | date:'yyyy-MM-dd' }}</p>
  <p>Formatted Date (EEEE, MMMM d, y): {{ currentDate | date:'EEEE, MMMM d, y' }}</p>
  <script>
    var app = angular.module('dateApp', []);
    app.controller('DateController', function ($scope) {
      $scope.currentDate = new Date();
    });
  </script>
</body>
</html>
```

## Output

