Sri Shridevi Charitable Trust (R.)
SHRIDEVI INSTITUTE OF ENGINEERING AND TECHNOLOGY
Sira Road, Tumkur - 572 106, Karnataka, India.
Phone: 0816 - 2212629 | Principal: 0816 - 2212627, 9686114899 | Telefax: 0816 - 2212628
Email: info@shrideviengineering.org, principal@shrideviengineering.org | Website: www.shrideviengineering.org
(Approved by AICTE, New Delhi, Recognised by Govt. of Karnataka and Affiliated to Visvesvaraya Technological University, Belagavi)
ESTD:2002

LAB MANUAL

(As per CBCS Scheme 2021)

Angular Js

Sub Code: **21CSL581**

Department of CSE

Prepared By:

Mr. Chethan M S
Assistant Professor
Dept of CSE,
SIET

ANGULAR JS			
Course Code	**21CSL581/ 21CBL583**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Total marks	100
Examination type (SEE)	PRACTICAL		

Course objectives:

- To learn the basics of Angular JS framework.
- To understand the Angular JS Modules, Forms, inputs, expression, data bindings and Filters
- To gain experience of modern tool usage (VS Code, Atom or any other] in developing Web applications

SL NO.	**Experiments**
1	Develop Angular JS program that allows user to input their first name and last name and display their full name. **Note**: The default values for first name and last name may be included in the program.
2	Develop an Angular JS application that displays a list of shopping items. Allow users to add and remove items from the list using directives and controllers. **Note**: The default values of items may be included in the program.
3	Develop a simple Angular JS calculator application that can perform basic mathematical operations (addition, subtraction, multiplication, division) based on user input.
4	Write an Angular JS application that can calculate factorial and compute square based on given user input.
5	Develop Angular JS application that displays 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 Angular JS 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 Angular JS program to create a simple CRUD application (Create, Read, Update, and Delete) for managing users.
8	Develop Angular JS program to create a login form, with validation for the username and password fields.
9	Create an Angular JS 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 Angular JS 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 Angular JS 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 Angular JS application that displays the date by using date filter parameters

NOTE: Include necessary HTML elements and CSS for the above Angular applications.

Course outcomes (Course Skill Set):

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

1. Develop Angular JS programs using basic features
2. Develop dynamic Web applications using Angular JS modules
3. Make use of form validations and controls for interactive applications
4. Appy the concepts of Expressions, data bindings and filters in developing Angular JS programs
5. Make use of modern tools to develop Web applications

Assessment Details (both CIE and SEE)

The weight age 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 10marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weight age 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 8t^ week of the semester and the second test shall be conducted after the 14'h week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weight age 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):

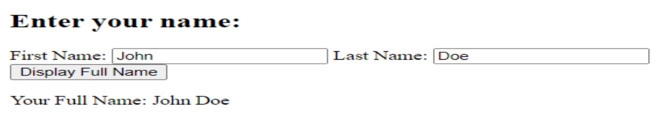
- SEE marks for the practical course is 50Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed 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 bythe examiners)
- The duration of SEE is 02hours

Rubrics suggested in Annexure-lI of Regulation book

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="fullNameApp">

<head>
  <title>AngularJS Full Name App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>

<body ng-controller="fullNameCtrl">

  <h2>Enter your name:</h2>

  <label for="firstName">First Name:</label>
  <input type="text" ng-model="firstName" placeholder="Enter your first name">

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

  <br>

  <button ng-click="displayFullName()">Display Full Name</button>

  <br>

  <p ng-show="fullName">Your Full Name: {{ fullName }}</p>

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

    app.controller('fullNameCtrl', function ($scope) {
      // Default values
      $scope.firstName = 'John';
      $scope.lastName = 'Doe';

      $scope.displayFullName = function () {
        // Concatenate first name and last name
        $scope.fullName = $scope.firstName + ' ' + $scope.lastName;
      };
    });
  </script>

</body>

</html>
```

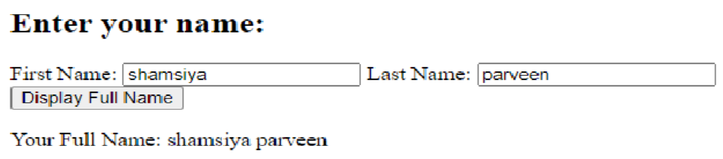
OUTPUT:



Enter your name:

First Name: John Last Name: Doe
Display Full Name

Your Full Name: John Doe



Enter your name:

First Name: shamsiya Last Name: parveen
Display Full Name

Your Full Name: shamsiya parveen

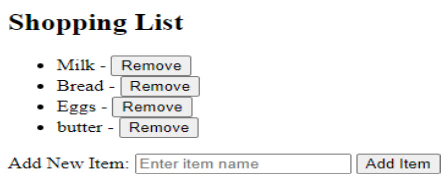
In this Application Program, the AngularJS code creates an app named fullNameApp with a controller named fullNameCtrl. The controller has default values for first name and last name, and a function displayFullName is defined to concatenate the first name and last name and assign it to the $scope.fullName variable. The HTML contains input fields for the first name and last name, a button to trigger the display function, and a paragraph to show the full name.

2. Develop an Angular JS application that displays a list of shopping items. Allow users to add and remove items from the list using directives and controllers. Note: The default values of items may be included in the program.

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

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

<body ng-controller="shoppingCtrl">

  <h2>Shopping List</h2>

  <ul>
    <li ng-repeat="item in shoppingItems">{{ item.name }} - <button ng-
click="removeItem($index)">Remove</button></li>
  </ul>

  <label for="newItem">Add New Item:</label>
  <input type="text" ng-model="newItem" placeholder="Enter item name">
  <button ng-click="addItem()">Add Item</button>

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

    app.controller('shoppingCtrl', function ($scope) {
      // Default shopping items
      $scope.shoppingItems = [
        { name: 'Milk' },
        { name: 'Bread' },
        { name: 'Eggs' }
      ];

      // Function to add a new item
      $scope.addItem = function () {
        if ($scope.newItem) {
          $scope.shoppingItems.push({ name: $scope.newItem });
          $scope.newItem = ''; // Clear input after adding item
        }
      };

      // Function to remove an item
      $scope.removeItem = function (index) {
        $scope.shoppingItems.splice(index, 1);
      };
    });
  </script>
```

```
</body>

</html>
```

OUTPUT:


Shopping List

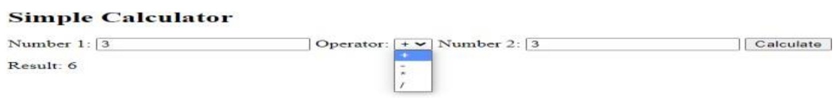
```
• Milk - Remove
• Bread - Remove
• Eggs - Remove
• butter - Remove
Add New Item: Enter item name Add Item
```

In this Application Program, the AngularJS code creates an app named shoppingApp with a controller named shoppingCtrl. The controller has a default list of shopping items and functions to add and remove items. The HTML contains an unordered list (<ul>) that uses ng-repeat to loop through the shopping items and display them. There's also an input field and a button to add new items, and each item has a "Remove" button to remove it from the list.

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 lang="en" ng-app="calculatorApp">
<head>
   <meta charset="UTF-8">
   <title>AngularJS Calculator</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-controller="CalculatorController">
   <h2>Simple Calculator</h2>

   <label for="num1">Number 1:</label>
   <input type="number" ng-model="num1" id="num1" />

   <label for="operator">Operator:</label>
   <select ng-model="operator" id="operator">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="*">*</option>
      <option value="/">/</option>
   </select>

   <label for="num2">Number 2:</label>
   <input type="number" ng-model="num2" id="num2" />

   <button ng-click="calculate()">Calculate</button>

   <p ng-if="result !== undefined">Result: {{ result }}</p>
</div>

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

   app.controller('CalculatorController', function ($scope) {
      $scope.calculate = function () {
         var num1 = parseFloat($scope.num1);
         var num2 = parseFloat($scope.num2);

         if (isNaN(num1) || isNaN(num2)) {
            $scope.result = 'Invalid input';
            return;
         }

         switch ($scope.operator) {
            case '+':
                $scope.result = num1 + num2;
                break;
            case '-':
                $scope.result = num1 - num2;
                break;
            case '*':
                $scope.result = num1 * num2;
                break;
            case '/':
                if (num2 !== 0) {
                    $scope.result = num1 / num2;
                } else {
                    $scope.result = 'Cannot divide by zero';
                }
                break;
            default:
                $scope.result = 'Invalid operator';
        }
    };
});
</script>

</body>
</html>
```

OUTPUT:



Simple Calculator

Number 1: 3 Operator: + Number 2: 3 Calculate

Result: 6

In this Application Program,Two input fields (num1 and num2) are used to take numeric input from the user.

A dropdown (operator) is used to select the mathematical operation (addition, subtraction, multiplication, division).

The calculate function is invoked when the user clicks the "Calculate" button, which performs the selected operation based on user input and displays the result.

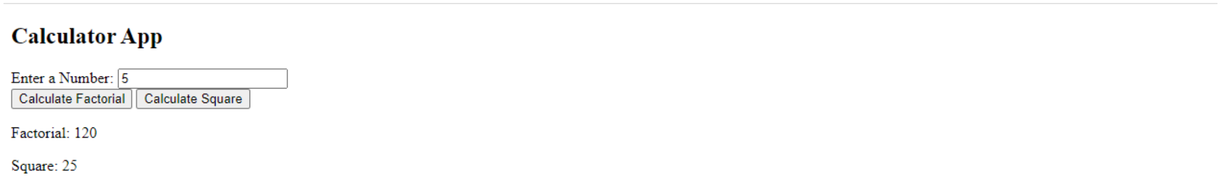
Basic input validation is performed to ensure that the user inputs valid numeric values and to avoid division by zero.

This is a basic example, and you can enhance it further based on your specific requirements.

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

<body ng-controller="calculatorCtrl">

  <h2>Calculator App</h2>

  <label for="numberInput">Enter a Number:</label>
  <input type="number" ng-model="userInput" placeholder="Enter a number">

  <br>

  <button ng-click="calculateFactorial()">Calculate Factorial</button>
  <button ng-click="calculateSquare()">Calculate Square</button>

  <br>

  <p ng-show="factorialResult">Factorial: {{ factorialResult }}</p>
  <p ng-show="squareResult">Square: {{ squareResult }}</p>

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

    app.controller('calculatorCtrl', function ($scope) {
      $scope.userInput = 0;
      $scope.factorialResult = null;
      $scope.squareResult = null;

      $scope.calculateFactorial = function () {
        $scope.factorialResult = getFactorial($scope.userInput);
      };

      $scope.calculateSquare = function () {
        $scope.squareResult = $scope.userInput * $scope.userInput;
      };

      function getFactorial(num) {
        if (num === 0 || num === 1) {
          return 1;
        } else {
          return num * getFactorial(num - 1);
        }
      }
    });
  </script>

</body>

</html>
```

OUTPUT:



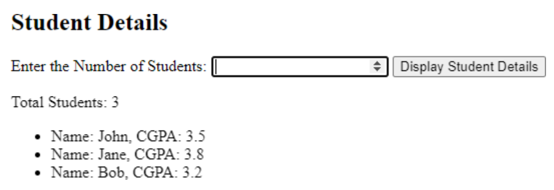
```
Calculator App

Enter a Number: 5
[Calculate Factorial] [Calculate Square]

Factorial: 120

Square: 25
```

In this Application Program, the AngularJS code creates an app named calculatorApp with a controller named calculatorCtrl. The controller has a default value for user input, and two functions, calculateFactorial and calculateSquare, that update the results based on the user input. The HTML contains input fields for the number, buttons to calculate the factorial and square, and paragraphs to display the results. The getFactorial function is a helper function to calculate the factorial recursilvely .

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 ng-app="studentApp">

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

<body ng-controller="studentCtrl">

  <h2>Student Details</h2>

  <label for="numberOfStudents">Enter the Number of Students:</label>
  <input type="number" ng-model="numberOfStudents">

  <button ng-click="displayStudentDetails()">Display Student Details</button>

  <br>

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

  <ul>
    <li ng-repeat="student in students">Name: {{ student.name }}, CGPA: {{ student.cgpa
}}</li>
  </ul>

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

    app.controller('studentCtrl', function ($scope) {
      $scope.students = [
        { name: 'John', cgpa: 3.5 },
        { name: 'Jane', cgpa: 3.8 },
        { name: 'Bob', cgpa: 3.2 }
      ];

      $scope.displayStudentDetails = function () {
        $scope.students = $scope.students.slice(0, $scope.numberOfStudents);
      };
    });
  </script>

</body>

</html>
```

OUTPUT:

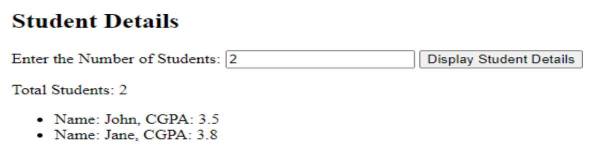


Student Details

Enter the Number of Students:

Total Students: 3

- Name: John, CGPA: 3.5
- Name: Jane, CGPA: 3.8
- Name: Bob, CGPA: 3.2



Student Details

Enter the Number of Students: 2

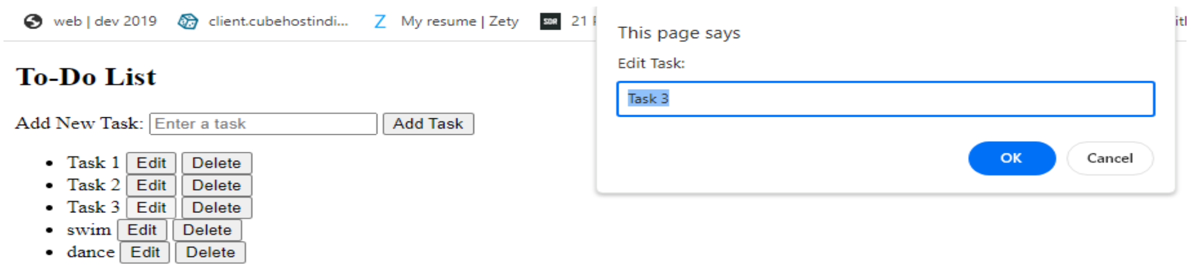
Total Students: 2

- Name: John, CGPA: 3.5
- Name: Jane, CGPA: 3.8

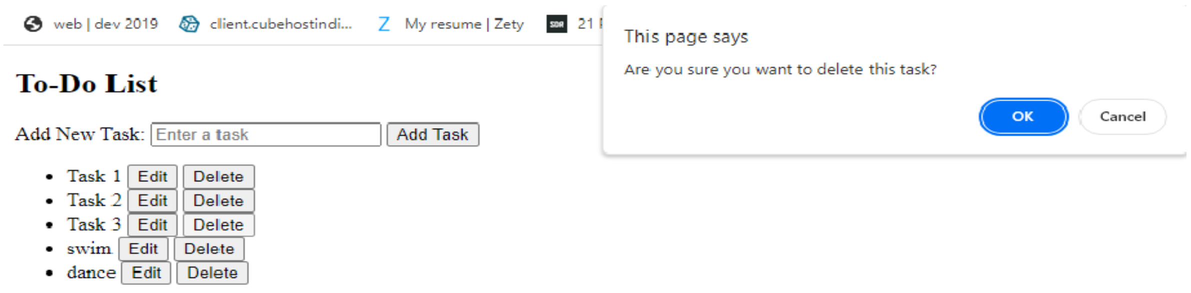
6. Develop an AngularJS program to create a simple to-do list application. Allow users to add, edit, and delete tasks.Note: The default values for tasks may be included in the program.

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

<head>
  <title>AngularJS To-Do App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>

<body ng-controller="todoCtrl">

  <h2>To-Do List</h2>

  <label for="newTask">Add New Task:</label>
  <input type="text" ng-model="newTask" placeholder="Enter a task">
  <button ng-click="addTask()">Add Task</button>

  <br>

  <ul>
    <li ng-repeat="task in tasks">
      {{ task }}
      <button ng-click="editTask($index)">Edit</button>
      <button ng-click="deleteTask($index)">Delete</button>
    </li>
  </ul>

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

    app.controller('todoCtrl', function ($scope) {
      $scope.tasks = ['Task 1', 'Task 2', 'Task 3'];

      $scope.addTask = function () {
        if ($scope.newTask) {
          $scope.tasks.push($scope.newTask);
          $scope.newTask = ''; // Clear input after adding task
        }
      };

      $scope.editTask = function (index) {
        var editedTask = prompt('Edit Task:', $scope.tasks[index]);
        if (editedTask !== null) {
          $scope.tasks[index] = editedTask;
        }
      };
      $scope.deleteTask = function (index) {
        var confirmDelete = confirm('Are you sure you want to delete this task?');
        if (confirmDelete) {
          $scope.tasks.splice(index, 1);
        }
      };
    });
  </script>

</body>

</html>
```

OUTPUT:



```
To-Do List
Add New Task: [Enter a task] [Add Task]
• Task 1 [Edit] [Delete]
• Task 2 [Edit] [Delete]
• Task 3 [Edit] [Delete]
• swim [Edit] [Delete]
• dance [Edit] [Delete]

This page says
Edit Task:
[Task 3]
[OK] [Cancel]
```



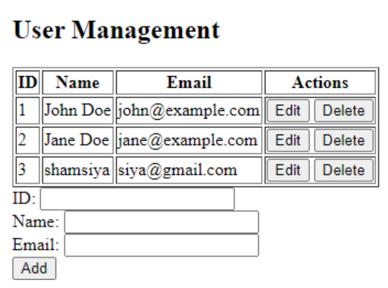
```
To-Do List
Add New Task: [Enter a task] [Add Task]
• Task 1 [Edit] [Delete]
• Task 2 [Edit] [Delete]
• Task 3 [Edit] [Delete]
• swim [Edit] [Delete]
• dance [Edit] [Delete]

This page says
Are you sure you want to delete this task?
[OK] [Cancel]
```

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

```
<!DOCTYPE html>
<html lang="en" ng-app="crudApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS CRUD App</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body>

<div ng-controller="UserController as userCtrl">
  <h2>User Management</h2>

  <!-- Display Users -->
  <table border="1">
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Email</th>
      <th>Actions</th>
    </tr>
    <tr ng-repeat="user in userCtrl.users">
      <td>{{ user.id }}</td>
      <td>{{ user.name }}</td>
      <td>{{ user.email }}</td>
      <td>
        <button ng-click="userCtrl.editUser(user)">Edit</button>
        <button ng-click="userCtrl.deleteUser(user.id)">Delete</button>
      </td>
    </tr>
  </table>

  <!-- Add or Edit User Form -->
  <form ng-submit="userCtrl.saveUser()">
    <label>ID:</label>
    <input type="text" ng-model="userCtrl.currentUser.id" required><br>

    <label>Name:</label>
    <input type="text" ng-model="userCtrl.currentUser.name" required><br>

    <label>Email:</label>
    <input type="email" ng-model="userCtrl.currentUser.email" required><br>

    <button type="submit">{{ userCtrl.isEditing ? 'Update' : 'Add' }}</button>
  </form>

</div>
```

```
<script>
  angular.module('crudApp', [])
    .controller('UserController', function () {
      var vm = this;
      vm.users = [
        { id: 1, name: 'John Doe', email: 'john@example.com' },
        { id: 2, name: 'Jane Doe', email: 'jane@example.com' },
      ];

      vm.currentUser = {};
      vm.isEditing = false;

      vm.editUser = function (user) {
        vm.currentUser = angular.copy(user);
        vm.isEditing = true;
      };

      vm.saveUser = function () {
        if (vm.isEditing) {
          // Update existing user
          var index = vm.users.findIndex(u => u.id === vm.currentUser.id);
          vm.users[index] = angular.copy(vm.currentUser);
        } else {
          // Add new user
          vm.users.push(angular.copy(vm.currentUser));
        }

        // Clear the form and reset flags
        vm.currentUser = {};
        vm.isEditing = false;
      };

      vm.deleteUser = function (userId) {
        // Delete user
        var index = vm.users.findIndex(u => u.id === userId);
        vm.users.splice(index, 1);
      };
    });
</script>

</body>
</html>
```

OUTPUT:



User Management

ID	Name	Email	Actions
1	John Doe	john@example.com	Edit Delete
2	Jane Doe	jane@example.com	Edit Delete
3	shamsiya	siya@gmail.com	Edit Delete

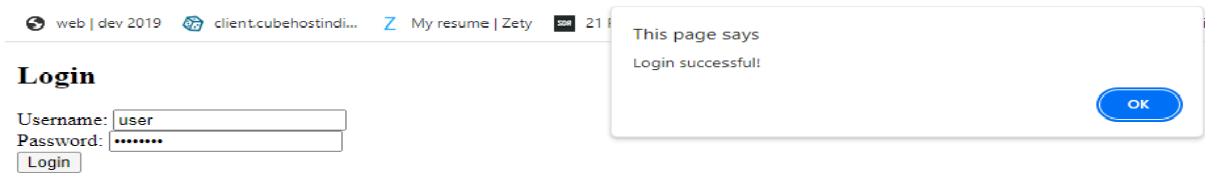
ID:
Name:
Email:
Add

In this Application Program, we have a simple AngularJS application with a controller (UserController) that manages a list of users. The application provides a table to display users, a form for adding or editing users, and buttons to trigger CRUD operations (Create, Read, Update, Delete). Please note that this is a basic example, and in a real-world application, you would likely want to use a server and RESTful API for data persistence.

8. Develop Angular JS program to create a login form, with validation for the usernameand password fields.

```
<!DOCTYPE html>
<html lang="en" ng-app="loginApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Login Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
  <style>
    .error {
      color: red;
    }
  </style>
</head>
<body>

<div ng-controller="LoginController as loginCtrl">
  <h2>Login</h2>

  <form ng-submit="loginCtrl.login()">
    <label>Username:</label>
    <input type="text" ng-model="loginCtrl.username" ng-pattern="/^[a-zA-Z0-9]+$/"
           ng-required="true" name="username">
    <span class="error" ng-show="loginForm.username.$error.required">Username is
required.</span>
    <span class="error" ng-show="loginForm.username.$error.pattern">Invalid characters in
username.</span>
    <br>

    <label>Password:</label>
    <input type="password" ng-model="loginCtrl.password" ng-required="true"
name="password">
    <span class="error" ng-show="loginForm.password.$error.required">Password is
required.</span>
    <br>

    <button type="submit">Login</button>
  </form>

</div>

<script>
  angular.module('loginApp', [])
    .controller('LoginController', function () {
      var vm = this;
      vm.username = '';
      vm.password = '';

      vm.login = function () {
        // Perform login logic here
        if (vm.username === 'user' && vm.password === 'password') {
          alert('Login successful!');
        } else {
          alert('Invalid username or password');
        }
      };
    });
</script>

</body>
</html>
```

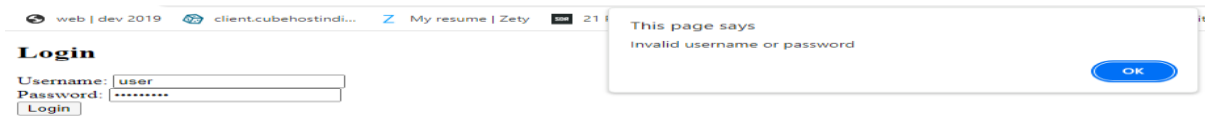
OUTPUT:



Login

Username: user
Password: ••••••••

This page says
Login successful!



Login

Username: user
Password: •••••••••

This page says
Invalid username or password

In this Application Program, we have a login form with a username and password field. The form uses AngularJS directives such as ng-model for two-way data binding, ng-pattern for input pattern validation, and ng-required for making fields required.

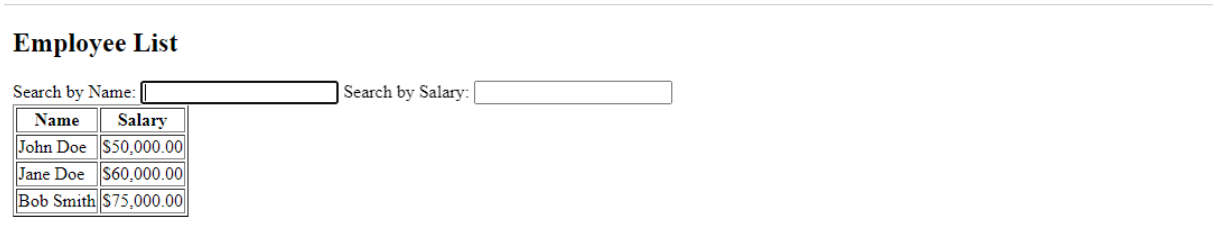
The login function is triggered when the form is submitted, and it performs a simple check for a predefined username ('user') and password ('password'). If the entered credentials match, it displays an alert indicating a successful login; otherwise, it shows an alert for invalid credentials. In a real-world scenario, you would typically perform authentication against a server.

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" ng-app="employeeApp">
<head>
  <meta charset="UTF-8">
  <title>Employee Search</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body>

<div ng-controller="EmployeeController as employeeCtrl">
  <h2>Employee List</h2>

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

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

  <table border="1">
    <tr>
      <th>Name</th>
      <th>Salary</th>
    </tr>
    <tr ng-repeat="employee in employeeCtrl.employees | filter:{name:
employeeCtrl.searchName, salary: employeeCtrl.searchSalary}">
      <td>{{ employee.name }}</td>
      <td>{{ employee.salary | currency }}</td>
    </tr>
  </table>

</div>

<script>
  angular.module('employeeApp', [])
    .controller('EmployeeController', function () {
      var vm = this;
      vm.employees = [
        { name: 'John Doe', salary: 50000 },
        { name: 'Jane Doe', salary: 60000 },
        { name: 'Bob Smith', salary: 75000 },
        // Add more employee details as needed
      ];

      vm.searchName = '';
      vm.searchSalary = '';
```

```
   });
</script>

</body>
</html>
```

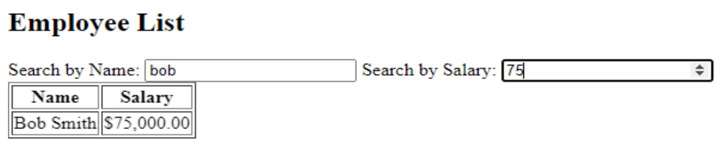
OUTPUT:



Employee List

Search by Name: Search by Salary:

Name	Salary
John Doe	$50,000.00
Jane Doe	$60,000.00
Bob Smith	$75,000.00



Employee List

Search by Name: bob Search by Salary: 75

Name	Salary
Bob Smith	$75,000.00

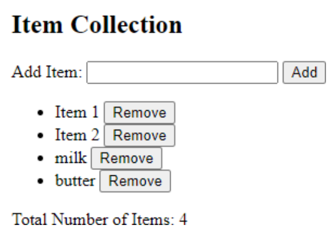
In this Application Program, we have an Angular JS application with a controller (Employee Controller) that manages a list of employees. The application provides input fields for searching employees by name and salary. The ng-repeat directive is used to display the filtered list of employees in a table.

You can extend the vm.employees array with more employee details as needed. The search functionality is implemented using the filter filter, which filters the employees based on the entered name and salary values. The currency filter is used to format the salary as currency.

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 lang="en" ng-app="itemApp">
<head>
  <meta charset="UTF-8">
  <title>Item Collection</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body>

<div ng-controller="ItemController as itemCtrl">
  <h2>Item Collection</h2>

  <label>Add Item:</label>
  <input type="text" ng-model="itemCtrl.newItemName">

  <button ng-click="itemCtrl.addItem()">Add</button>

  <ul>
    <li ng-repeat="item in itemCtrl.items">
      {{ item.name }}
      <button ng-click="itemCtrl.removeItem(item)">Remove</button>
    </li>
  </ul>

  <p>Total Number of Items: {{ itemCtrl.items.length }}</p>

</div>

<script>
  angular.module('itemApp', [])
    .controller('ItemController', function () {
      var vm = this;
      vm.items = [
        { name: 'Item 1' },
        { name: 'Item 2' },
        { name: 'Item 3' },
        // Add more default items as needed
      ];

      vm.newItemName = '';

      vm.addItem = function () {
        if (vm.newItemName.trim() !== '') {
          vm.items.push({ name: vm.newItemName });
          vm.newItemName = ''; // Clear the input field after adding an item
        }
      };

      vm.removeItem = function (item) {
        var index = vm.items.indexOf(item);
        if (index !== -1) {
        vm.items.splice(index, 1);
        }
      };
    });
</script>

</body>
</html>
```

OUTPUT:



```
Item Collection

Add Item: [          ] Add

  • Item 1 Remove
  • Item 2 Remove
  • milk Remove
  • butter Remove

Total Number of Items: 4
```

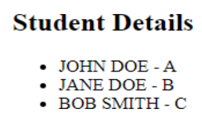
In this Application Program, we have an Angular JS application with a controller (ItemController) that manages a collection of items. Users can add items to the collection by entering a name in the input field and clicking the "Add" button. Items can be removed by clicking the "Remove" button next to each item.

The total number of items is dynamically displayed below the list, and it automatically updates as items are added or removed. You can extend the vm.items array with more default items as needed.

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 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>

   <ul>
      <li ng-repeat="student in students">
         {{ student.name | uppercase }} - {{ student.grade | uppercase }}
      </li>
   </ul>
</div>

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

   app.controller('StudentController', function ($scope) {
      // Default student details
      $scope.students = [
         { name: 'John Doe', grade: 'A' },
         { name: 'Jane Doe', grade: 'B' },
         { name: 'Bob Smith', grade: 'C' },
         // Add more students as needed
      ];
   });
</script>

</body>
</html>
```

OUTPUT:



```
Student Details

 • JOHN DOE - A
 • JANE DOE - B
 • BOB SMITH - C
```

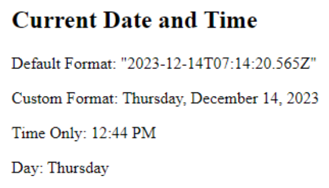
In this Application Program, the uppercase filter is used to convert the name and grade properties of each student to uppercase. The ng-repeat directive is used to iterate over the list of students and display their details in uppercase.

Note: Make sure to include AngularJS before your custom script to ensure that the AngularJS functionalities are available when your script runs. Additionally, consider upgrading to Angular for a more modern and actively maintained framework.

12. Create an AngularJS application that displays the date by using date filter parameters NOTE: Include necessary HTML elementsand CSS for the above Angular applications.

```
<!DOCTYPE html>
<html lang="en" ng-app="dateApp">
<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>

<div ng-controller="DateController">
   <h2>Current Date and Time</h2>

   <p>Default Format: {{ currentDate }}</p>
   <p>Custom Format: {{ currentDate | date:'fullDate' }}</p>
   <p>Time Only: {{ currentDate | date:'shortTime' }}</p>
   <p>Day: {{ currentDate | date:'EEEE' }}</p>
</div>

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

   app.controller('DateController', function ($scope, $interval) {
      // Update the date every second
      $interval(function () {
         $scope.currentDate = new Date();
      }, 1000);
   });
</script>

</body>
</html>
```

OUTPUT:



```
Current Date and Time

Default Format: "2023-12-14T07:14:20.565Z"

Custom Format: Thursday, December 14, 2023

Time Only: 12:44 PM

Day: Thursday
```

In this Application Program:

The date filter is used to format the current date in different ways.
The default format is displayed as it is ({{ currentDate }}).
The fullDate parameter is used to display the date in a full format ({{ currentDate | date:'fullDate' }}).
The shortTime parameter is used to display the time only ({{ currentDate | date:'shortTime' }}).
The EEEE parameter is used to display the day of the week ({{ currentDate | date:'EEEE' }}).
The date is updated every second using the $interval service,