Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

Remarks: _____ **Sign:** _____

Aim: Demonstrate print "Hello Word" with Angular js. It specifies the Model, View, Controller part of an Angular jsp app.

Model-View-Controller (MVC) is a software design pattern that separates an application into three interconnected parts: the Model (data and logic), the View (user interface), and the Controller (user input and interaction)

- **Model:**

  Represents the data and business logic of the application.

  Handles interactions with databases and other data sources.

  Is responsible for storing, retrieving, and manipulating data.

- **View:**

  Responsible for displaying the data to the user.

  Is the user interface of the application.

  Can be a web page, a form, or any other type of user interface.

- **Controller:**

  Acts as the intermediary between the Model and the View.

  Receives user input, interacts with the Model to process data, and then updates the View to reflect the changes.

  Handles user interactions and application logic.

Code:

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>AngularJS MVC Example</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

   <!-- View Part -->
   <div ng-controller="myController">
      <h1>{{ message }}</h1> <!-- Binding Model to View -->
   </div>

   <script>
      // Controller Part
      var app = angular.module("myApp", []); // Creating the AngularJS app (Module)

      app.controller("myController", function($scope) {
         $scope.message = "Hello World"; // Defining the Model
      });
   </script>
</body>
</html>
```

Output:

Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

Remarks: _____ **Sign:** _____

Aim: Demonstrate angular js script to implement Built-in Directives.

AngularJS provides several built-in directives to extend HTML functionality. Some commonly used directives are:

1. **ng-app** → Defines an AngularJS application.
2. **ng-init** → Initializes data in the model.
3. **ng-model** → Binds input fields to variables.
4. **ng-bind** → Binds data dynamically.
5. **ng-repeat** → Loops through arrays.
6. **ng-show/ng-hide** → Show or hide elements conditionally.
7. **ng-if** → Conditionally render elements.
8. **ng-click** → Handles click events.

Form Directives

Directive	Description
`ng-submit`	Handles form submission.
`ng-disabled`	Disables a form element conditionally.
`ng-pattern`	Validates input using regular expressions.
`ng-change`	Executes a function when the value of an input field changes.

Style and Animation Directives

Directive	Description
`ng-style`	Dynamically applies CSS styles.
`ng-class`	Adds or removes CSS classes dynamically.
`ng-animate`	Enables animations when elements are added/removed.

Code:

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>AngularJS Built-in Directives</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-init="message='Hello, AngularJS!'; showText=true; names=['John', 'Alice', 'Bob'];">

   <!-- ng-bind: Binds data dynamically -->
   <h1 ng-bind="message"></h1>

   <!-- ng-model: Two-way data binding -->
   <input type="text" ng-model="message">
   <p>Updated Message: {{ message }}</p>

   <!-- ng-repeat: Loops through an array -->
   <h3>List of Names:</h3>
   <ul>
      <li ng-repeat="name in names">{{ name }}</li>
   </ul>

   <!-- ng-show / ng-hide -->
   <button ng-click="showText = !showText">Toggle Text</button>
   <p ng-show="showText">This text is visible.</p>

   <!-- ng-if: Conditional rendering -->
   <p ng-if="names.length > 2">There are more than 2 names in the list.</p>

</body>
</html>
```

Output:

Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

Remarks: _____ **Sign:** _____

Aim: Demonstrate angular js script to add modules and controller.

What is Modules and Controllers

AngularJS modules are used to divide or separate the logic such as controllers, services, application etc. and keep the code clean. AngularJS module helps to link many components, so it is just a group of related components.

AngularJS supports modular approach of programming. AngularJS modules are used to divide or separate the logic such as controllers, services, application etc. and keep the code clean. AngularJS module helps to link many components, so it is just a group of related components.

Generally in most of the applications we have a single entry point (main method) that instantiate and club together different parts of the application. In angularjs applications we don't have that main method instead we have modules that specify how our application will be structured and bootstrapped

Steps:

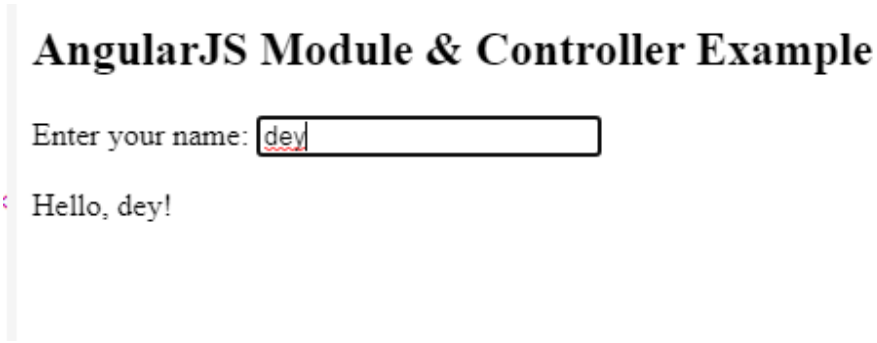
1. Create an **AngularJS Module** using angular.module().
2. Define a **Controller** inside the module using .controller().
3. Bind the controller to the HTML using ng-controller.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AngularJS Module & Controller</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="myApp">
  <!-- Binding Controller to the Div -->
  <div ng-controller="MyController">
    <h2>AngularJS Module & Controller Example</h2>
    <p>Enter your name: <input type="text" ng-model="name"></p>
    <p>Hello, {{ name }}!</p>
  </div>
```

```
    <script>
        // Define the AngularJS Module
        var app = angular.module("myApp", []);

        // Define the Controller inside the module
        app.controller("MyController", function($scope) {
            $scope.name = "Leena"; // Default value
        });
    </script>
</body>
</html>
```

Output:



Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

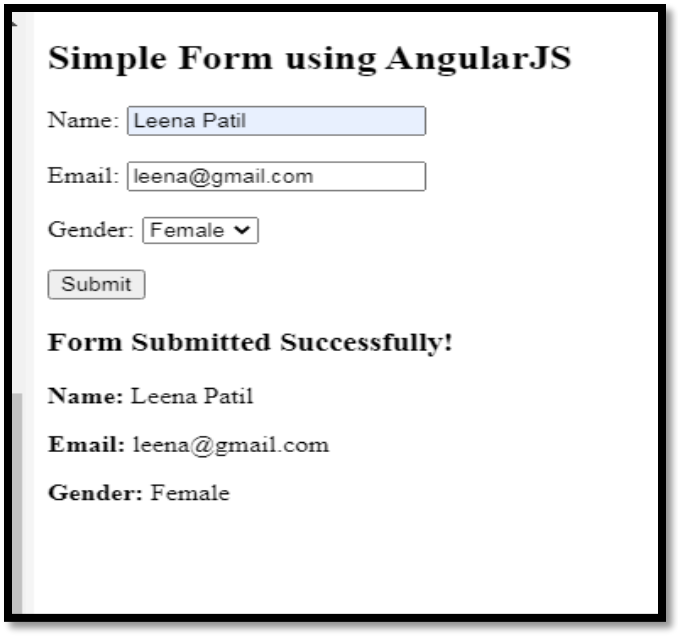
Remarks: _____ **Sign:** _____

Aim: Demonstrate simple form using angular js script.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>AngularJS Simple 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 ng-app="formApp">
   <div ng-controller="FormController">
      <h2>Simple Form using AngularJS</h2>
      <!-- AngularJS Form -->
      <form name="userForm" ng-submit="submitForm()">
         <label>Name:</label>
         <input type="text" name="name" ng-model="user.name" required>
         <span class="error" ng-show="userForm.name.$touched &&
userForm.name.$invalid">Required</span>
         <br><br>
         <label>Email:</label>
         <input type="email" name="email" ng-model="user.email" required>
         <span class="error" ng-show="userForm.email.$touched && userForm.email.$invalid">Valid Email
Required</span>
         <br><br>
         <label>Gender:</label>
         <select name="gender" ng-model="user.gender" required>
            <option value="">Select</option>
            <option value="Male">Male</option>
            <option value="Female">Female</option>
         </select>
```

```
        <span class="error" ng-show="userForm.gender.$touched &&
userForm.gender.$invalid">Required</span>
        <br><br>
        <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
      </form>
      <!-- Display Submitted Data -->
      <div ng-show="submitted">
        <h3>Form Submitted Successfully!</h3>
        <p><strong>Name:</strong> {{ user.name }}</p>
        <p><strong>Email:</strong> {{ user.email }}</p>
        <p><strong>Gender:</strong> {{ user.gender }}</p>
      </div>
    </div>
    <script>
      // Define AngularJS Module
      var app = angular.module("formApp", []);
      // Define Controller
      app.controller("FormController", function($scope) {
        $scope.user = {};
        $scope.submitted = false;
        // Submit Function
        $scope.submitForm = function() {
          if ($scope.userForm.$valid) {
            $scope.submitted = true;
          }
        };
      });
    </script>
</body>
</html>
```

Output:



Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

Remarks: _____ **Sign:** _____

Aim: Demonstrate the use of JSON in a webpage.

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *****

Uses of JSON
It is used while writing JavaScript based applications that includes browser extensions and websites.
JSON format is used for serializing and transmitting structured data over network connection.
It is primarily used to transmit data between a server and web applications.
Web services and APIs use JSON format to provide public data.
It can be used with modern programming languages.

Code:

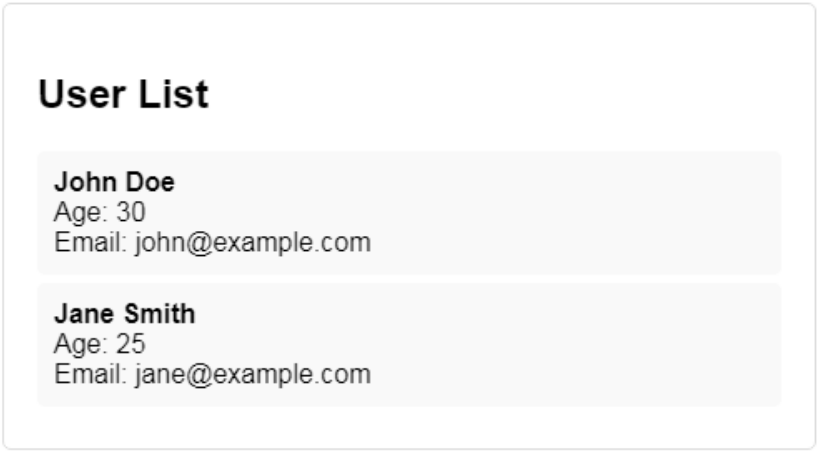
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JSON Data in Webpage</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    .container { max-width: 600px; margin: auto; padding: 20px; border: 1px solid #ddd; border-radius:
5px; }
    .user { background: #f9f9f9; padding: 10px; margin: 5px 0; border-radius: 5px; }
  </style>
</head>
<body>
  <div class="container">
    <h2>User List</h2>
    <div id="userList"></div>
  </div>
```

```
    <script>
        // Sample JSON Data
        const jsonData = [
            { "name": "John Doe", "age": 30, "email": "john@example.com" },
            { "name": "Jane Smith", "age": 25, "email": "jane@example.com" }
        ];

        // Function to display JSON data in the webpage
        function displayUsers(data) {
            const userList = document.getElementById("userList");
            userList.innerHTML = "";
             data.forEach(user => {
                let userDiv = document.createElement("div");
                userDiv.classList.add("user");
                userDiv.innerHTML = `<strong>${user.name}</strong><br>Age: ${user.age}<br>Email:
${user.email}`;
                userList.appendChild(userDiv);
            });
        }

        // Load JSON data on page load
        window.onload = () => displayUsers(jsonData);
    </script>
</body>
</html>
```

Output:



Name: _____ **Class:** _____

Expt.Title:_____

Roll No: _____ **Expt No:** _____ **Date:** _____

Remarks: _____ **Sign:** _____

Aim: Demonstrate Installation steps of MongoDB

Step 1: Download MongoDB

1. Open your web browser and go to the **MongoDB Official Download Page**.
2. Select your **Operating System** (Windows).
3. Choose the latest **MongoDB Community Server** version.
4. Click **Download** to get the .msi installer.

Step 2: Install MongoDB

1. Locate the downloaded .msi file and double-click to start the installer.
2. In the **MongoDB Setup Wizard**, click **Next**.
3. Accept the **License Agreement** and click **Next**.
4. Choose **"Complete"** setup type (recommended).
5. Check the box **"Run MongoDB as a Service"** to start MongoDB automatically after installation.
6. Click **Install** and wait for the process to complete.
7. Click **Finish** when done.

Step 3: Add MongoDB to System Path (Optional)

To use mongod and mongo commands from the terminal:

1. Open **System Properties → Advanced → Environment Variables**.
2. Under **System Variables**, find and edit the Path variable.
3. Click **New** and add:
   C:\Program Files\MongoDB\Server\<version>\bin
4. Click **OK** and restart your computer.

Step 4: Start MongoDB Server

1. Open **Command Prompt** (cmd) and run:
   mongod
   -----If MongoDB is installed correctly, it will start running on **port 27017**.-----

Step 5: Connect to MongoDB

1. Open a new **Command Prompt** window.
2. Run the following command to start the MongoDB shell:
   mongo
3. If connected successfully, you'll see the MongoDB shell ready for commands.

Step 6: Verify Installation

Run the command inside the MongoDB shell:

db.version()

It should return the installed version of MongoDB.