**Web Technologies**

**Subject Code: MCAL14**

A Practical Journal Submitted in Fulfilment of Degree of

**Master in Computer Application**

**Year 2023-2024**

By

**Mr. Sagar Govind Kamtekar**

**(Application ID: 81690)**

Semester – 1

Institute of Distance and Open Learning Vidya Nagari, Kalina, Santacruz East – 400098,

University of Mumbai

**PCP Center**

[Vidyavardhini’s College of Engineering & Technology, Vasai Road]

Sagar Govind Kamtekar (81690)

Institute of Distance and Open Learning Vidya Nagari, Kalina, Santacruz East – 400098,

University of Mumbai

**CERTIFICATE**

This is to certify that**, Sagar Govind Kamtekar** appearing **Master’s in Computer Application (Semester 1) 81690**, has satisfactorily completed the prescribed practical of **MCA14 – Web Technologies** as laid down by the University of Mumbai for the academic year 2023-24.

Teacher in charge Examination Coordinator

IDOL, MCA

University of Mumbai

Date: Place:

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Practical | Signature |
| 1. | Implementation of Node Function like Arithmetic Operations in AngularJS. |  |
| 2. | Implementation of Create, Insert, Delete Operations in AngularJS. |  |
| 3. | Create an Application to establish a connection with MySQL Database and perform basic database operation on it. |  |
| 4. | Create an Application using Filters in AngularJS. |  |
| 5. | Create an Application to demonstrate Directives in AngularJS. |  |
| 6. | Create an Application to demonstrate Controllers in AngularJS. |  |
| 7. | Create an Angular Form in AngularJS. |  |
| 8. | Create a Single Page Application in AngularJS. |  |
| 9. | Create an Application to demonstrate Events in AngularJS. |  |
| 10. | Create an Application to demonstrate REPL in AngularJS. |  |

Practical 1

AIM:

Implementation of Node Function like Arithmetic Operations in AngularJS.

OBJECTIVE:

The objective of implementing arithmetic operations in AngularJS is to create a web application that allows users to perform basic arithmetic operations (addition, subtraction, multiplication, and division) on two input numbers.

THEORY:

AngularJS is a JavaScript-based open-source front-end web application framework developed and maintained by Google. It simplifies the process of building dynamic web applications by providing a modular and structured approach to development.

CODE:

<!DOCTYPE html>

<html>

<head>

<title>AngularJS Arithmetic Application </title>

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

</head>

<body ng-app>

<h1> Addition AngularJS Application </h1>

Enter Numbers to Add: <input type="number" ng-model="Num1"/> + <input type="number" ng-model="Num2"/> = <span>{{Num1 + Num2}}</span>

<h1> Subtraction AngularJS Application </h1>

Enter Numbers to Subtract: <input type="number" ng-model="Num3"/> - <input type="number" ng-model="Num4"/> = <span>{{Num3 - Num4}}</span>

<h1> Multiplication AngularJS Application </h1>

Enter Numbers to Multiply: <input type="number" ng-model="Num5"/> x <input type="number" ng-model="Num6"/> = <span>{{Num5 \* Num6}}</span>

<h1> Division AngularJS Application </h1>

Enter Numbers to Divide: <input type="number" ng-model="Num7"/> / <input type="number" ng-model="Num8"/> = <span>{{Num7 / Num8}}</span>

</body>

</html>

OUTPUT:

Practical 2

AIM:

Implementation of Create, Insert, Delete Operations in AngularJS.

OBJECTIVE:

Use some of the AngularJS features to make a shopping list, where user can add or remove items.

THEORY:

A shopping cart is similar to original grocery shopping cart; it means that on a website that sells products or services online, the shopping cart is a common metaphor that acts as online store’s catalog and ordering process. It is a graphical representation of a supermarket on a vendor’s website that keeps a list of items a customer has picked up from the online store.

CODE:

<!DOCTYPE html>

<html>

<title> AngularJS File Handling Application</title>

<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.errortext = "";

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>

<div ng-app="myShoppingList" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in products">

{{x}} <span ng-click="removeItem($index)">&times;</span>

</li>

</ul>

<input ng-model="addMe">

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

<p>{{errortext}}</p>

</div>

</body>

</html>

OUTPUT:

Practical 3

AIM:

Create an Application to establish a connection with MySQL Database and perform basic database operation on it.

OBJECTIVE:

THEORY:

CODE:

OUTPUT:

Practical 4

AIM:

Create an Application using Filters in AngularJS.

OBJECTIVE:

A Filter in AngularJS helps to format the value of an expression to display to the user without changing the original format.

THEORY:

AngualrJS filter is a tool, which we can use to format the data. With this filter, the user can see and modify accordingly to the requirement. It is added in angular to format the data that is being displayed on the view part.

CODE:

<!DOCTYPE html>

<html>

<head>

<title> AngularJS Filter Application </title>

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

</head>

<body>

<h1> AngularJS Filter Application </h1>

<div ng-app="mainApp" ng-controller="studentController">

<table border="0">

<tr>

<td> Enter First Name: </td>

<td><input type="text" ng-model="student.firstName"></td>

</tr>

<tr>

<td> Enter Last Name: </td>

<td><input type="text" ng-model="student.lastName"></td>

</tr>

<tr>

<td> Enter Fees: </td>

<td><input type="text" ng-model="student.fees"></td>

</tr>

<tr>

<td> Enter Subject: </td>

<td><input type="text" ng-model="subjectName"></td>

</tr>

</table>

<table border="0">

<tr>

<td> Name in Upper Case: </td>

<td> {{student.fullName()|uppercase}}</td>

</tr>

<tr>

<td> Name in Lower Case: </td>

<td> {{student.fullName()|lowercase}}</td>

</tr>

<tr>

<td> Fees: </td>

<td> {{student.fees|currency}}</td>

</tr>

<tr>

<td> Subject: </td>

<td>

<ul>

<li ng-repeat="subject in student.subjects | filter: subjectName | orderBy:'marks' as filteredSubjects">

{{subject.name + ', marks: ' + subject.marks}}

</li>

</ul>

</td>

</tr>

</table>

</div>

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller('studentController', function($scope) {

$scope.student = {

firstName: "Mahesh",

lastName: "Gharat",

fees: 500,

subjects:[

{name: 'Physics', marks: 70},

{name: 'Chemistry', marks: 80},

{name: 'Maths', marks: 65}

],

fullName: function() {

studentObject = $scope.student;

return studentObject.firstName + " " + studentObject.lastName;

}

};

});

</script>

</body>

</html>

OUTPUT:

Practical 5

AIM:

Create an Application to demonstrate Directives in AngularJS.

OBJECTIVE:

Angular provides a number of built-in directives, which are attributes, added to the HTML elements that give us dynamic behavior.

THEORY:

Directives are markers on a DOM element that tell AngularJS to attach a specified behavior to that DOM element or even transform the DOM element and its children. Most of the directives in AngularJS are starting with “ng-“ where “ng” stands for Angular.

CODE:

<!DOCTYPE html>

<html>

<head>

<title>AngularJS Directives Application</title>

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

<style>

div {

width: 100%;

height: 50px;

display: block;

margin: 15px 0 0 10px;

}

</style>

</head>

<body ng-app ng-init="checked=true">

<input type="text" ng-model="name1"/>

<div>

Hello {{name1}}

</div><br>

Enter Name: <input type="text" ng-model="name"/><br>

data-ng-bind: <span data-ng-bind="name"></span><br>

data-ng-bind: <span data-ng-bind="name"></span><br>

data-ng-bind: <span data-ng-bind="name"></span><br>

x-ng-bind: <span x-ng-bind="name"></span><br>

x\_ng\_bind: <span x\_ng\_bind="name"></span><br>

ng-bind: <span ng-bind="name"></span><br>

Click Me: <input type="checkbox" ng-model="checked"/>

<div>

New: <input ng-if="checked" type="text"/>

</div>

<div>

Read-only: <input ng-readonly="checked" type="text" value="This is read-only"/>

</div>

<div>

Disabled: <input ng-disabled="checked" type="text" value="This is disabled."/>

</div>

</body>

</html>

OUTPUT:

Practical 6

AIM:

Create an Application to demonstrate Controllers in AngularJS.

OBJECTIVE:

Create an application that needs to set up the initial state for the AngularJS “$scope.” set up the initial state of a scope by attaching properties to the $scope object.

THEORY:

The Controller in AngularJS is a JavaScript function that maintains the application data and behavior using $scope object. This can attach properties and methods to the $scope object inside a controller function, which in turn will add or update the data and attach behavior to HTML elements. The $scope object is a glue between the controller and HTML.

CODE:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title> AngularJS Controller Application </title>

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

</head>

<body ng-app="spicyApp2">

<div ng-controller="SpicyController">

<input ng-model="customSpice">

<button ng-click="spicy('chili')"> Chili </button>

<button ng-click="spicy(customSpice)"> Custom Spice </button>

<p> The food is {{spice}} spicy! </p>

</div>

<script>

var app = angular.module('spicyApp2', []);

app.controller('SpicyController', function($scope) {

$scope.spice = 'very';

$scope.spicy = function(spice) {

$scope.spice = spice;

};

});

</script>

</body>

</html>

OUTPUT:

Practical 7

AIM:

Create an Angular Form in AngularJS.

OBJECTIVE:

Create Simple Angular Form using different input controls & events.

THEORY:

AngularJS facilitates you to create a form enriched with data binding and validation of input controls. Input controls are ways for a user to enter data. A form is a collection of controls for the purpose of grouping related controls together.

CODE:

<!DOCTYPE html>

<html>

<head>

<title>AngularJS Form Validation</title>

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

<style>

.error {

color: red;

}

</style>

</head>

<body>

<h2>AngularJS Form Validation</h2>

<div ng-app="validationApp" ng-controller="mainController">

<form name="myForm" novalidate>

<label for="name">Name:</label>

<input type="text" name="name" ng-model="user.name" ng-minlength="3" ng-maxlength="10" required>

<span class="error" ng-show="myForm.name.$error.required">Name is required.</span>

<span class="error" ng-show="myForm.name.$error.minlength">Name is too short.</span>

<span class="error" ng-show="myForm.name.$error.maxlength">Name is too long.</span>

<br>

<label for="email">Email:</label>

<input type="email" name="email" ng-model="user.email" required>

<span class="error" ng-show="myForm.email.$error.required">Email is required.</span>

<span class="error" ng-show="myForm.email.$error.email">Invalid email address.</span>

<br>

<button ng-disabled="myForm.$invalid" ng-click="submitForm()">Submit</button>

</form>

</div>

<script>

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

app.controller('mainController', function($scope) {

$scope.user = {};

$scope.submitForm = function() {

// Form submission logic can be added here

alert('Form submitted successfully!');

};

});

</script>

</body>

</html>

OUTPUT:

Practical 8

AIM:

Create a Single Page Application in AngularJS.

OBJECTIVE:

Create a single page application that loads a single HTML page and only a part of the page instead of the entire page gets updated with every click of the mouse.

THEORY:

Single page applications or SPA’s are web applications that load a single HTML page and dynamically update the page based on the user interaction with the web application. Single page application is a web application that fits on a single page. All your code (JS, HTML, CSS) is retrieved with a single page load and navigation between pages performed without refreshing the whole page.

CODE:

<!DOCTYPE html>

<html ng-app="myApp">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Form Application</title>

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.min.js"></script>

</head>

<body>

<script type="text/ng-template" id="pages/first.html">

<h1>First</h1>

<h3>{{message}}</h3>

</script>

<script type="text/ng-template" id="pages/second.html">

<h1>Second</h1>

<h3>{{message}}</h3>

</script>

<script type="text/ng-template" id="pages/third.html">

<h1>Third</h1>

<h3>{{message}}</h3>

</script>

<a href="#/first">First</a>

<a href="#/second">Second</a>

<a href="#/third">Third</a>

<div ng-view></div>

<script>

var app = angular.module('myApp', ['ngRoute']);

app.config(function ($routeProvider) {

$routeProvider

.when('/first', {

templateUrl: 'pages/first.html',

controller: 'FirstController'

})

.when('/second', {

templateUrl: 'pages/second.html',

controller: 'SecondController'

})

.when('/third', {

templateUrl: 'pages/third.html',

controller: 'ThirdController'

})

.otherwise({

redirectTo: '/first'

});

});

app.controller('FirstController', function ($scope) {

$scope.message = 'Hello from First Controller';

});

app.controller('SecondController', function ($scope) {

$scope.message = 'Hello from Second Controller';

});

app.controller('ThirdController', function ($scope) {

$scope.message = 'Hello from Third Controller';

});

</script>

</body>

</html>

OUTPUT:

Practical 9

AIM:

Create an Application to demonstrate Events in AngularJS.

OBJECTIVE:

The objective of implementing a Circular Queue in the C language is to create a data structure that follows the First In, First Out (FIFO) principle with a circular arrangement of elements.

THEORY:

A Circular Queue is a variation of a regular queue where the last element is connected to the first element, forming a circle. This implementation allows for better space utilization and avoids the waste of memory space.

CODE:

<!DOCTYPE html>

<html ng-app="eventExample">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AngularJS Event Example</title>

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

</head>

<body ng-controller="EventController">

<button ng-click="toggleMessage()">Toggle Message</button>

<p>{{ message }}</p>

<script>

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

app.controller('EventController', function ($scope) {

$scope.message = 'Hello';

$scope.toggleMessage = function () {

$scope.message = ($scope.message === 'Hello') ? 'Goodbye' : 'Hello';

};

});

</script>

</body>

</html>

OUTPUT:

Practical 10

AIM:

Create an Application to demonstrate REPL in AngularJS.

OBJECTIVE:

THEORY:

CODE:

OUTPUT: