UNIT-3

UI Design Angular JS

Differences between AngularJS and Javascript

- Abstraction: AngularJS abstracts many low-level tasks, making development faster and easier, while JavaScript requires more manual coding.
- Structure: AngularJS promotes a structured approach with its MVC architecture, while JavaScript is more flexible but less structured.
- Learning Curve: AngularJS has a steeper learning curve due to its concepts like directives, dependency injection, and the MVC pattern, whereas JavaScript is simpler to start with.
- Performance: JavaScript is faster in terms of raw execution speed since it's closer to the core browser functions, while AngularJS might introduce some overhead due to its abstractions.

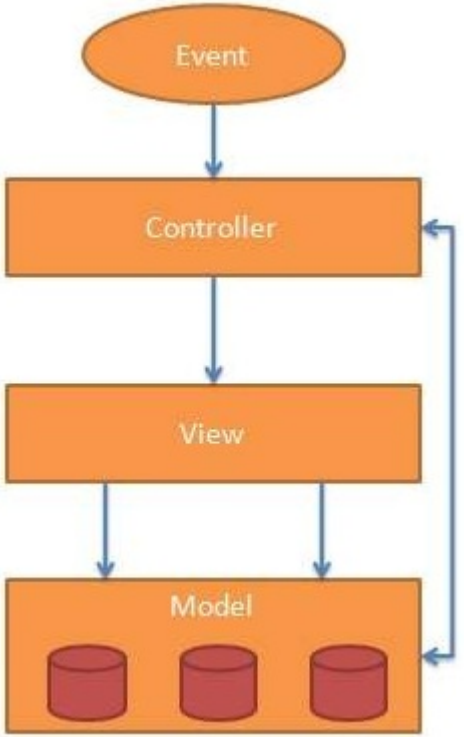
Introduction

- Angular JS is an open source JavaScript framework by Google to build web applications. It can be freely used, changed and shared by anyone.
- Angular Js is developed by Google.
- It is an excellent framework for building business applications.

Advantage of AngularJS

- **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
- **Two way data binding:** AngularJS provides two-way data binding, meaning any changes in the model are reflected in the view, and vice versa, without additional code.
- **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
- **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
- Directives, filters, modules, routes etc.

AngularJS MVC architecture



- AngularJS follows the Model-View-Controller (MVC) architecture, which is a design pattern used to separate concerns in an application, making it easier to manage, scale, and maintain.

1. Model

- **Definition:** The Model in AngularJS represents the data of the application. It is typically in the form of JavaScript objects and handles the business logic and data management.
- **Role:**
  - **Data Management:** The model is responsible for managing and holding the data that the application needs.
  - **Business Logic:** It also handles the business rules of the application, manipulating data and applying logic as needed.
  - **Communication:** In AngularJS, the model communicates with the view via the controller, and data binding helps keep the model and view synchronized.

2. View

- **Definition:** The View is the user interface of the application. It is responsible for displaying the data to the user and providing an interface for interaction.
- **Role:**
  - **Presentation:** The view presents the data from the model to the user. In AngularJS, views are typically written in HTML with embedded AngularJS expressions and directives.
  - **Data Binding:** AngularJS provides two-way data binding, meaning any changes in the model are reflected in the view, and vice versa, without additional code.
  - **Directives:** AngularJS uses directives to extend HTML with additional functionality, allowing for dynamic views.

3. Controller

- **Definition:** The Controller acts as an intermediary between the model and the view.

It processes user input, manipulates the model, and updates the view accordingly.

- **Role:**
  - **Logic Handling:** The controller receives the request to respond to user actions. It processes inputs, interacts with the model, and updates the view.
  - **Scope Management:** In AngularJS, controllers manage the $scope object, which acts as a glue between the view and the model. The $scope object holds the model data and methods that can be accessed and manipulated in the view.
  - **Event Handling:** Controllers handle events triggered by user interactions with the view, such as button clicks or form submissions, and update the model or view as needed.

MVC Works in AngularJS

- **Initialization:**
  - The application initializes the model, setting up data and functions that are needed by the view.
- **User Interaction:**
  - The user interacts with the view, triggering events such as clicks, inputs, or selections.
- **Event Handling:**
  - The controller captures these events, processes the data, and makes necessary changes to the model.
- **Model Update:**
  - Any changes to the model are automatically reflected in the view due to AngularJS's two-way data binding.
- **View Update:**
  - The view updates itself based on the changes in the model, providing feedback to the user.

- In AngularJS, ng-app, ng-controller, module, and scope are essential concepts and directives that form the foundation of an AngularJS application.

ng-app

- DirectiveDefinition: The ng-app directive is used to define the root element of an AngularJS application. It tells AngularJS that the element and its children are part of an AngularJS application.
- Role:
  - Bootstrap Application: When the browser loads the page, AngularJS looks for the ng-app directive to initialize the application. It bootstraps the AngularJS application and initializes the module associated with it.
  - Module Association: The ng-app directive can be used to specify the name of the AngularJS module that should be associated with the application.

ng-controller

- Definition: The ng-controller directive attaches a controller class to the view. The controller is responsible for setting up the model and managing the business logic for the view.
- Role:
  - Scope Binding: The ng-controller directive creates a new scope ($scope) object and associates it with the controller. The $scope object is then available for the view to bind data and functions.
  - Controller Logic: The controller logic defined in JavaScript is executed when the view is initialized, setting up the data and behavior that the view will use.

Necessary Steps for creating simple AngularJS Application

- Create the angularJS application with angular.module(‘AppName', []);
- Define the controller with

```
appName.controller(‘Controllername', function($scope) {
  $scope.variable = '';
});
```

In html,

- Include the angularJS Library with the link https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js using script tag or download the angular.min.js and use it path
- Include the Angular application and controllers using the directives ng-app and ng-controller. Then access the scope objects variable.

First Example “app1.html”

```
<html ng-app="myApp1">
<head>
   <title>My AngularJS App</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
   <script src="app1.js"></script>
</head>
<body>
   <div ng-controller="myController">
      <input type="text" ng-model="name" placeholder="Enter your name">
      <h1>Hello, {{ name }}!</h1>
   </div>
</body>
</html>
```

First Example “app1.js”

```
var app = angular.module('myApp1', []);
app.controller('myController', function($scope) {
  $scope.name = '';
});
```

Scope Object

- In AngularJS, the concept of scope ($scope) is a fundamental part of the framework's design, acting as a crucial link between the controller (business logic) and the view (HTML).
- In AngularJS, the scope is a JavaScript object that contains the application's model data. It serves as a context for expressions, and it's where you define the properties and methods that will be available to both the view and the controller.

Role of Scope

- Data Binding: The scope is primarily used for binding data between the view (HTML) and the controller (JavaScript). AngularJS's two-way data binding allows the view and the controller to stay in sync. Any changes in the scope's data are immediately reflected in the view, and vice versa.
- Event Handling: Scope is also responsible for handling events. You can define methods on the scope that are triggered by user interactions, like clicking a button or submitting a form.
- Scope Hierarchy: Scopes can have a hierarchical structure. A child scope inherits properties from its parent scope, but it can also have its own properties. This allows for modular and encapsulated components within an application.

Example2: Event Handling app2.html

- <html>
- <html ng-app="myApp2">
- <head>
- <title>AngularJS Event Handling</title>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
- <script src="app2.js"></script>
- </head>
- <body>
-
- <button ng-click="greet()">Click Me!</button>
- <h1>{{ message }}</h1>
-
- </body>
- </html>

Example2: Event Handling app2.js

```
var app = angular.module('myApp2', []);
    app.controller('myController', function($scope) {
        $scope.message = "Welcome to AngularJS!";
        $scope.greet = function() {
            $scope.message = "Click Event";
        };
    });
```

Example3: Scope Hierarchy app3.html

```
<html>
<html ng-app="myApp3">
<head>
  <title>AngularJS Event Handling</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script src="app3.js"></script>
</head>
<body>
  <div ng-controller="ParentController">
    <h1>Parent Scope: {{ parentMessage }}</h1>
    <div ng-controller="ChildController">
      <h2>Child Scope: {{ childMessage }}</h2>
      <h3>Inherited Message: {{ parentMessage }}</h3>
    </div>
  </div>

 </body>
</html>
```

Example3: Scope Hierarchy app3.js

- var app = angular.module('myApp3', []);
- app.controller('ParentController', function($scope) {
- $scope.parentMessage = "Hello from Parent!";
- });
- app.controller('ChildController', function($scope) {
- $scope.childMessage = "Hello from Child!";
- });
-

AngularJS Expressions

- In AngularJS, expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written.
- Expressions are written inside double braces {{expression}}.
- They can also be written inside a directive:

  ng-bind="expression".

Example 4

```
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app>
<p>A simple expression example: {{ 5 + 15 }}</p>
</div>
</body>
</html>
```

Example5

```
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app ng-init="quantity=5;cost=5">
<p>Total in dollar: {{ quantity * cost }}</p>
</div>
</body>
</html>
```

Example6

```
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app>
<p>A simple expression example: <span ng-bind=5+25></span></p>
</div>
</body>
</html>
```

Example7

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app ng-init="quantity=5;cost=6">
<p>Total in dollar: <span ng-bind=quantity*cost></span></p>
</div>
</body>
</html>
```

AngularJS Control Statements

- In AngularJS, control statements are typically implemented using directives that allow you to control the flow and behavior of your application's logic directly within the HTML. These control statements can be used for conditional rendering, looping, and handling different states in your application.

ng-if

- The ng-If directive is used to include or exclude an element in the DOM based on a condition.

- ngRepeat - Looping/Iteration
- The ngRepeat directive is used to repeat a set of HTML elements for each item in a collection (like an array). It’s analogous to a for loop in traditional programming.

Example

```
<html ng-app="myApp">
<head>
  <title>Odd or Even Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="myController">

  <ul>
    <li ng-repeat="number in numbers">
      <span ng-if="number % 2 === 0">{{ number }} is Even</span>
      <span ng-if="number % 2 !== 0">{{ number }} is Odd</span>
    </li>
  </ul>

</body>
</html>
```

```
var app = angular.module('myApp', []);
    app.controller('myController', function($scope) {
      $scope.numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    });
```

ngSwitch - Conditional Case Statements

- The ngSwitch directive allows you to conditionally display elements based on a single expression. It's similar to the switch-case statement in traditional programming languages.

Example

```
<html ng-app="myApp">
<head>
  <title>ngSwitch Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-controller="myController">
  <ul>
    <li ng-repeat="number in numbers" ng-switch="numberCategory(number)">
       <span ng-switch-when="positive">{{ number }} is Positive</span>
       <span ng-switch-when="negative">{{ number }} is Negative</span>
       <span ng-switch-when="zero">{{ number }} is Zero</span>
     </li>
  </ul>
</body>
</html>
```

```
var app = angular.module('myApp', []);
  app.controller('myController', function($scope) {
    $scope.numbers = [-2, -1, 0, 1, 2];
    // Function to categorize numbers
    $scope.numberCategory = function(number) {
      if (number > 0) {
        return 'positive';
      } else if (number < 0) {
        return 'negative';
      } else {
        return 'zero';
      }
    };
  });
```

ng-show and ng-hide

- Displays or hides elements based on the evaluation of an expression.

Example

```
<html>
   <head>
      <title>ngShow and hide Example</title>
      <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
      <script src="app.js"></script>
     </head>
<div ng-app="myApp">
   <div ng-controller="MyController">
     <button ng-click="showDiv = !showDiv">Toggle Visibility</button>
     <div ng-show="showDiv">
       This div will be hidden when the button is clicked.
     </div>
   </div>
 </div>
 </html>
```

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

app.controller('MyController', function($scope) {
  $scope.showDiv = false;
});
```

Controller (ctrl) or this in Controller

- ctrl is a common shorthand for controller, and it's often used to refer to this when using the controller-as syntax. In AngularJS, the controller-as syntax allows us to assign the controller instance to a variable (commonly ctrl for "view model") and bind data and methods directly to this instance instead of using $scope
- Encapsulation: Using ctrl with the controller-as syntax encourages better encapsulation and is more aligned with object-oriented principles. It also simplifies the mental model by reducing the need to think about $scope.

Controller as ctrl Ex1

```
<div ng-controller="MyController as ctrl">
  <p>{{ ctrl.message }}</p>
  <button ng-click="ctrl.updateMessage()">Change Message</button>
</div>
```

```
angular.module('app', [])
  .controller('MyController', function() {
    var ctrl = this;
    ctrl.message = "Hello, World!";
    ctrl.updateMessage = function() {
      ctrl.message = "Hello, AngularJS!";
    };
  });
```

Controller as ctrl Ex2

```
<html ng-app="counterApp">
<head>
  <title>Simple Counter with ctrl in AngularJS</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
</head>
<body>
  <div ng-controller="CounterController as ctrl">
    <h1>Counter Example</h1>
    <p>Current Count: {{ ctrl.count }}</p>
    <button ng-click="ctrl.increment()">Increment</button>
    <button ng-click="ctrl.decrement()">Decrement</button>
  </div>

  <script src="app.js"></script>
</body>
</html>
```

```
angular.module('counterApp', [])
  .controller('CounterController', function() {
    var ctrl = this;

    // Initialize the count variable
    ctrl.count = 0;

    // Function to increment the count
    ctrl.increment = function() {
      ctrl.count += 1;
    };

    // Function to decrement the count
    ctrl.decrement = function() {
      ctrl.count -= 1;
    };
  });
```

“App.js”

Scope vs ctrl

Aspect	Scope	Ctrl
Syntax	This approach is used in angular JS only.	Controller-as syntax, which is considered more modern and preferable**.**
Usage in Components	When working with legacy AngularJS codebases or if you need to work with inherited scopes or advanced directive features where $scope is necessary. It is not used with components	When writing new code or working with components, as it encourages better practices, cleaner code, and aligns more closely with modern JavaScript patterns and Angular (2+) and react practices.

Components

- In AngularJS later versions, components provide a streamlined way to create self-contained, reusable elements with their own view and logic. They encourage the use of a **component-based architecture**

Key Features of Components :

- **Simplified Syntax:** Components have a straightforward API compared to directives.
- **Encapsulation:** Each component manages its own scope, template, and behavior.
- **Reusability:** Components can be easily reused across different parts of the application.

Differences B/W Directives and Components

- Directives are a core feature of AngularJS, but as the web development landscape evolved, the need for a more standardized approach became clear.
- Components align more closely with the architecture of modern frameworks like Angular (2+), and React. This makes it easier for developers to transition from AngularJS to these newer frameworks when needed.

Creating a Simple Component Ex1

```
<html ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Component Example</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.
min.js"></script>
<script src=“App.js”></scritp>
</head>
<body>
  <!-- Component Usage -->
  <greeting></greeting>
</body>
</html>
```

”Ex.html”

```
angular.module('myApp', [])
  .component('greeting', {
    template: '<h1>Hello, Good Morning!</h1>‘
    controller:operation
  });
function operation(){ }
```

“App.js”

- Component to perform the arithmetic operations using two numbers

• Component Ex2

```
angular.module('myApp', [])
.component('arithop', {
 template: '<div>' +
'<input type="number" ng-model="$ctrl.num1">' +
'<input type="number" ng-model="$ctrl.num2">' +
'<button ng-click="$ctrl.add()">Add</button>' +
'<button ng-click="$ctrl.subtract()">Subtract</button>' +
'<button ng-click="$ctrl.multiply()">Multiply</button>' +
'<button ng-click="$ctrl.divide()">Divide</button>' +
'<p>Result: {{ $ctrl.result }}</p>' +
'</div>‘ ,
controller: operation
 });

 function operation(){
var ctrl = this;

ctrl.add = function () {
  ctrl.result = ctrl.num1 + ctrl.num2;
};

ctrl.subtract = function () {
  ctrl.result = ctrl.num1 - ctrl.num2;
};

ctrl.multiply = function () {
  ctrl.result = ctrl.num1 * ctrl.num2;
};

ctrl.divide = function () {
  if (ctrl.num2 !== 0) {
    ctrl.result = ctrl.num1 / ctrl.num2;
  } else {
    ctrl.result = 'Cannot divide by zero!';
  }
};
 }
```

Ex2

- <html ng-app="myApp">
- <head>
- <meta charset="UTF-8">
- <title>AngularJS Component Example</title>
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
- <script src="app1.js"></script>
- </head>
- <body>
- <!-- Component Usage -->
- <arithop></arithop>
- </body>
- </html>
-

AngularJS Service

- In AngularJS, services are used to share data and logic across multiple components, controllers, or other services. A service is essentially a singleton object that can be injected into different parts of the application.

- Angular JS service example to use the square service across different component

Service Example: app2.js

```
angular.module('myApp', [])
  // Define the service
 .service('squareservice', function() {
  this.square = function(a) {
    return a * a;
  };
 })
```

```
.component('arithop', {
 template:
  `<div><input type="number" ng-model="$ctrl.num1"><button ng-click="$ctrl.square()">Square</button><p>Result: {{ $c
 controller: operation
});

// Define the operation function (controller)
function operation(squareservice) {
 var ctrl = this;
 ctrl.num1 = 0;
 ctrl.result = 0;

 ctrl.square = function() {
  ctrl.result = squareservice.square(ctrl.num1);
 };
}
```

Service_ex.html

```
<html ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Service Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script src="app2.js"></script>
</head>
<body>
  <arithop></arithop>
</body>
</html>
```

USECASE1

- Create a component to print the square and cube of the given number
  - Component should have text box to read the number
  - Square button to perform square operation and display square result
  - Cube button to perform square operation and display cube result

USECASE2

- Create a service to perform the arithmetic operations
  - Create a Component should have text boxes to read the two numbers
  - Square button to perform arithemetic operation and display the result

Filters

- In AngularJS, filters are used to format the value of an expression for display to the user. We can use filters in view templates, controllers, components or services. Filters can be used to transform data such as formatting dates, filtering arrays, or converting strings to uppercase.

- AngularJS comes with several built-in filters, such as
  - currency,
  - date,
  - filter,
  - json,
  - lowercase,
  - uppercase, and more.

- Currency Filter: Formats a number as a currency string (e.g., $99.99).
- Date Filter: Formats a date object to a human-readable string based on a specified format (e.g., fullDate, shortDate).
- Filter Filter: Selects a subset of items from an array based on a search criteria.
- JSON Filter: Converts an object into a JSON string, useful for debugging or displaying raw data.
- Lowercase Filter: Converts a string to all lowercase letters.
- Uppercase Filter: Converts a string to all uppercase letters.

Currency filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <p>Price: {{ ctrl.price | currency }}</p>
</div>
```

```
 angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.price = 99.99;
});
```

Date filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <p>Today's Date: {{ ctrl.today | date:'fullDate' }}</p>
</div>
```

```
 angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.today = new Date();
});
```

The date filter can format dates in various ways using different format options. The fullDate is just one of the predefined formats. Here are some other predefined format options you can use:

1.short:

- Outputs: M/d/yy, h:mm a
- Example: 9/3/24, 12:30 PM

3.medium:

- Outputs: MMM d, y, h:mm:ss a
- Example: Sep 3, 2024, 12:30:00 PM

5.long:

- Outputs: MMMM d, y, h:mm:ss a z
- Example: September 3, 2024, 12:30:00 PM GMT+0

7.full:

- Outputs: EEEE, MMMM d, y, h:mm:ss a zzzz
- Example: Tuesday, September 3, 2024, 12:30:00 PM GMT

9.shortDate:

- Outputs: M/d/yy
- Example: 9/3/24

11.mediumDate:

- Outputs: MMM d, y
- Example: Sep 3, 2024

13.longDate:

- Outputs: MMMM d, y
- Example: September 3, 2024

15.shortTime:

- Outputs: h:mm a
- Example: 12:30 PM

17.mediumTime:

- Outputs: h:mm:ss a
- Example: 12:30:00 PM

19.longTime:

- Outputs: h:mm:ss a z
- Example: 12:30:00 PM GMT+0

21.fullTime:

filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <p>Search: <input ng-model="ctrl.searchText"></p>
  <ul>
    <li ng-repeat="item in ctrl.items | filter:ctrl.searchText">
      {{ item }}
    </li>
  </ul>
</div>
```

```
angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.items = {“one”,”two”,three”,”four”};
});
```

Custom filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">

  <ul>
    <li ng-repeat="item in ctrl.numbers | filter:ctrl.greaterThan20">
      {{ item }}
    </li>
  </ul>
</div>
```

```
angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  //ctrl.items = ["one","two","three","four"]
  ctrl.numbers = [5, 12, 25, 30, 7, 21, 18];

  ctrl.greaterThan20 = function(number) {
    return number>20;
  };
});
```

json filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <pre>{{ ctrl.data | json }}</pre>
</div>
```

```
 angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.data = {name: "John", age: 30, city: "New York"};
});
```

lowercase filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <p>{{ ctrl.message | lowercase }}</p>
</div>
```

```
angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.message = "HELLO WORLD";
});
```

uppercase filter

```
<div ng-app="myApp" ng-controller="myCtrl as ctrl">
  <p>{{ ctrl.message | uppercase }}</p>
</div>
```

```
angular.module('myApp', [])
.controller('myCtrl', function() {
  var ctrl = this;
  ctrl.message = "hello world";
});
```

AngularJS Form validation

- AngularJS provides a built-in form validation mechanism that can be used to validate user input in forms. Here are some of the key features and directives used for form validation in AngularJS:

  1.ng-required: Specifies that a field is required.

  2. ng-minlength and ng-maxlength: Specify the minimum and maximum length of a field.

  3. ng-pattern: Specifies a regular expression pattern that the field value must match.

  4. ng-email: Specifies that a field must contain a valid email address.

  5. $error: An object that contains error messages for each field.

Example for AngularJS Form validation

```
<html>
<head>
   <title>Form Validation</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.
js"></script>
<script src=“app.js”></script>
</head>
<body ng-app="validationApp" ng-controller="formController as ctrl">
```

Textbox validation

```
<form name="myForm" ng-submit="ctrl.submitForm(myForm)" novalidate>

    <label for="name">Name:</label>
 <input type="text" name="name" ng-model="ctrl.user.name"
       required maxlength="50"
       pattern="^[A-Za-z ]+$" /> <br>
 <div ng-show="myForm.name.$touched && myForm.name.$invalid">
  <small ng-show="myForm.name.$error.required">Name is required.</small>
  <small ng-show="myForm.name.$error.maxlength">Name must be less than 50 characters.</small>
  <small ng-show="myForm.name.$error.pattern">Name can only contain letters and spaces.</small>
 </div>
```

Radio button validation

```
<label for="gender">Gender:</label><br>
    <input type="radio" name="gender" value="Male" ng-model="ctrl.user.gender" required> Male
    <input type="radio" name="gender" value="Female" ng-model="ctrl.user.gender" required> Female <br>

    <div ng-show="myForm.gender.$touched && myForm.gender.$invalid">
     <small ng-show="myForm.gender.$error.required">Gender is required.</small>
    </div>


    <button type="submit" ng-disabled="myForm.$invalid">Submit</button>
  </form> </body>
</html>
```

E-mail Validation

```
<label for="email">Email:</label>
  <input type="email" name="email" ng-model="ctrl.user.email" required maxlength="100" />
 <div ng-show="myForm.email.$touched && myForm.email.$invalid">
      <small ng-show="myForm.email.$error.required">Email is required.</small>
     <small ng-show="myForm.email.$error.email">Invalid email format.</small>
      <small ng-show="myForm.email.$error.maxlength">Email must be less than 100 characters.</small>
 </div>
```

App.js

```
angular.module('validationApp', [])
  .controller('formController', function() {
    var ctrl = this;  // Assign 'this' to 'ctrl'

    ctrl.user = {};   // Initialize the user object

    ctrl.submitForm = function(form) {
      if (form.$valid) {
        alert('Form is valid!');
      } else {
        alert('Form is invalid!');
      }
    };
  });
```

Implement the form validation using AngularJs for Students Registration forms with

Name

Regno

Mailid

Gender

Age

Department

Hobbies

Mini Projects

Use Case: Shopping Cart Management

- Description: Implement a shopping cart system that allows users to browse a list of products, add products to the cart, and view the cart contents with total price. The system should also update the product quantity in real-time when added or removed from the cart.
- Requirements:

1. Display a list of products with images, names, prices, and available quantities.
2. Allow users to add products to the shopping cart.
3. Update the product quantity in real-time when added to the cart.
4. Display the cart contents with product names, prices, and quantities.
5. Calculate and display the total price of the cart contents.
6. Allow users to remove products from the cart.
7. Update the product quantity in real-time when removed from the cart.
8. Use the CSS to display the web page with your creative style

Use Case: Online Food Ordering System

- Description: A restaurant wants to implement an online food ordering system that allows customers to browse their menu, add items to their order, and place the order online.
- Requirements:

1. The system should display the restaurant's menu, including food images, names, and prices.

2. Customers should be able to add menu items to their order and view their order summary.

3. The system should calculate the total price of the order based on the items added.

4. Customers should be able to remove items from their order.

5. The system should allow customers to place their order online.

6. Use the CSS to display the web page with your creative style

Usecase: Event Registration System

- Description: A conference organizer wants to implement an event registration system that allows attendees to register for events, select sessions, and pay registration fees.
- Requirements:

1. Display event details, including date, time, location, and description.
2. Allow attendees to register for the event.
3. Display session details, including speaker names, session times, and descriptions.
4. Allow attendees to select sessions.
5. Calculate registration fees based on selected sessions.
6. Implement payment gateway integration.
7. Use the CSS to display the web page with your creative style

Application1: Develop an AngularJS application that displays a list of products retrieved from a JSON file. [**Note:Example to retrieve data from the JSON file]**

“ product.json” file


```
{
   "products": [
      { "prod_id": 101, "name": "Laptop", "price": 1200, "availability": true },
      { "prod_id": 102, "name": "Smartphone", "price": 800, "availability": true },
      { "prod_id": 103, "name": "Headphones", "price": 150, "availability": false },
      { "prod_id": 104, "name": "Keyboard", "price": 50, "availability": true },
      { "prod_id": 105, "name": "Monitor", "price": 300, "availability": false }
   ]
}
```

Index.html

```
<html ng-app="productApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Product List</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-controller="ProductController as ctrl">
  <h1>Product List</h1>
  <!-- Display products in a table -->
  <table border="1" cellpadding="5" cellspacing="0">
    <thead>
      <tr>
        <th>Product ID</th>
        <th>Product Name</th>
        <th>Price</th>
        <th>Status</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="product in ctrl.products">
        <td>{{ product.prod_id }}</td>
        <td>{{ product.name }}</td>
        <td>{{ product.price | currency }}</td>
        <td>
          <span ng-if="product.availability">In Stock</span>
          <span ng-if="!product.availability" style="color:red;">Out of Stock</sp
        </td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

App.js

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

// Define the controller
app.controller('ProductController', ['$http', function($http) {
    var ctrl = this;
    ctrl.products = [];

    // Fetch the product data from the JSON file
    $http.get('products.json').then(function(response) {
        ctrl.products = response.data.products;
    });
}]);
```

OUTPUT:

Application 2: AngularJS application that displays a list of products, with each product having a name, price, and availability status. It will display a message if no products are available.

Note : Example to retrieve the data from the list

Index.html

```
<html  ng-app="productApp">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product List</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
   </script>
    <script src="app.js"></script>
</head>
<body ng-controller="ProductController as ctrl">

    <h1>Product List</h1>

    <!-- Conditionally display a message if no products are available -->
    <p ng-if="!ctrl.products.length">No products available.</p>

    <!-- Display products in a table if available -->
```

```
<!-- Display products in a table if available -->
 <table ng-if="ctrl.products.length" border="1" cellpadding="5" cellspacing="0">
    <thead>
      <tr>
        <th>Product Name</th>
        <th>Price</th>
        <th>Availability Status</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="product in ctrl.products">
        <td>{{ product.name }}</td>
        <td>{{ product.price | currency }}</td>
        <td>
          <span ng-if="product.availability">In Stock</span>
          <span ng-if="!product.availability" style="color:red;">Out of Stock</span>
        </td>
      </tr>
    </tbody>
  </table>

</body>
</html>
```

App.js

```
// Define the AngularJS application module
var app = angular.module('productApp', []);

// Define the controller for managing the product data
app.controller('ProductController', function() {
    var ctrl = this;

    // Array of product objects with name, price, and availability
    ctrl.products = [
        { name: 'Laptop', price: 1200, availability: true },
        { name: 'Smartphone', price: 800, availability: true },
        { name: 'Headphones', price: 150, availability: false }
    ];
});
```

Output