**Tutorial No. 08**

**Title: Introduction to Angular JS and React JS.**

**Batch:B2 Roll No.:1914078 Tutorial No:8**

**Aim**: To implement methods, functions to manipulate DOM element using Angular JS and React JS

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed: Notepad++, Web Browser**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Theory:**

AngularJS is an open source JavaScript framework maintained by Google and community which can help developers to create single page applications. Its purpose is to help developing the web applications with model-view-controller (MVC) capability in an effort to make development, maintaining and testing easier. SPA(Single Page Application) is getting popular nowadays and the technology like AngularJS aids to create such applications. AngularJS is a very powerful JavaScript library mostly used for Single Page Application (SPA) projects. The SPA is a web application that fully loads all of the resources in initial request and then the page components are replaced by other component depending on user interaction.

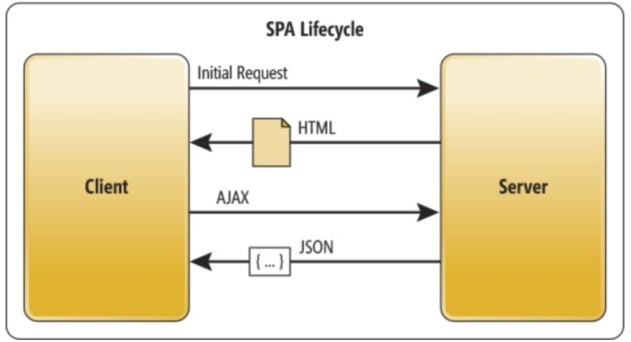
“Single Page Application (SPA) is composed of individual components that can be replaced/updated independently, without refreshing/reloading whole page so that the entire page needs not to be reloaded on every user action.” AngularJS divides the front end into 3 parts.



Code that is easy to read and understand becomes a key to achieve high efficiency and good quality as well as leads to fluid user interface.

In SPA AngularJS,

* Client makes initial request to open a particular website or application.
* Server respond with HTML pages along with images, CSS, script files, and other external resources.
* Now when client does any interaction like button click on input from keyboard with a page, it makes a new request to a server.
* Server now responds with only user’s action result not the entire HTML page again. That is server responds with a JSON (JavaScript Object Notation) which includes only result of user’s action.
* At client side that page is not reloaded rather a particular part of a page is replaced/updated with that response. Hence very less bandwidth as well as time is required for this Client-Server Request-Response Cycle. This leads to very fluid user interface.



**home.html** <!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Single Page Application in AngularJS </title>

<scriptsrc="//ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular .min.js"> <script src="app.js" ></script>

</head>

<body ng-app="myapp" ng-controller="myctrl">

<div ng-show=”showme”>

<label>Enter your Name : <input type=”text” placeholder=”joe” ng-model=”username”/>

</label><br/>

<label><button ng-click=”submit()”>Click Me</button>

</label><br/></div>

<div ng-show=”!showme”>

<p>Welcome to our website {{ user }}.</p>

</div>

</body>

</html>

**app.js--**

var app=angular.module('myapp',['ngRoute']); app.config(function($routeProvider){ $routeProvider .when('/',{ templateUrl: 'home.html' });

});

app.controller('myctrl',function($scope)

{ $scope.showme = true; $scope.submit = function()

{ $scope.showme = false; $scope.user = $scope.username; }

});

In this example we start our application by creating HTML page which is presentation part of the sample program. AngularJS libraries are not available until unless we link them by using following script tags, first and second line will bring AngularJS core libraries into the application and the last line will attach AngularJS module that we have created for our sample. We use ng-app and ng-controller directive to inject our ‘myapp’ module and ‘myctrl’ controller to our page.

ng-model directive is used to bind the data between controller and views, and ng-show directive takes the Boolean value ‘true’ or ‘false’ which allows us to display and hide the content of web page.

<input type=”text” placeholder=”joe” ngmodel=”username”/> and ng-show directive takes the Boolean value ‘true’ or ‘false’ which allows us to display and hide the content of web page.

<div ng-show=”showme”>

Expressions in AngularJS are written inside {{}} curly brasses which are evaluated on page load or some event.

<p>Welcome to our website {{ user }}.</p>

Then create app.js file which have the module and controller for the sample program, to create module and configure it using following line of code.

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

app.config(function($routeProvider){ We also inject the ‘ngRoute’ dependency to our module inside [] square brackets.

Configuring routes in AngularJS is done in following way, $routeProvider .when('/',{ templateUrl:'home.html' });

Controllers are the backbone of AngularJS and they are defined in following code snippet.

app.controller('myctrl',function($scope){ /\* business logic goes here. \*/ });

**React JS**

React is a declarative, efficient, and flexible JavaScript library for building user interfaces.

It lets you compose complex UIs from small and isolated pieces of code called components”. We use components to tell React what we want to see on the screen. When our data changes, React will efficiently update and re-render our components.

React does not manipulate the browser's DOM directly. Instead, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

We will start with React .Component example class ShoppingList extends React.Component { render() {

return (

<div className="shopping-list">

<h1>Shopping List for {this.props.name}</h1>

<ul>

<li>Instagram</li>

<li>WhatsApp</li>

<li>Oculus</li>

</ul>

</div>

);

}

}

Here, ShoppingList is a React component class, or React component type. A component takes in parameters, called props (short for “properties”), and returns a hierarchy of views to display via the render method.

The render method returns a description of what you want to see on the screen. React takes the description and displays the result. In particular, render returns a React element, which is a lightweight description of what to render.

Most React developers use a special syntax called “JSX” which makes these structures easier to write.

The <div /> syntax is transformed at build time to React.createElement('div').

The example above is equivalent to: return React.createElement('div', {className: 'shopping-list'},

React.createElement('h1', /\* ... h1 children ... \*/),

React.createElement('ul', /\* ... ul children ... \*/)

);

The createElement is an inbuilt method. There are lots of methods and functions included in the reactjs API. see the details on reactjs.org API Reference.

JSX comes with the full power of JavaScript. You can put any JavaScript expressions within braces inside JSX. Each React element is a JavaScript object that you can store in a variable or pass around in your program.

**Hello World Example**

<!DOCTYPE html>

<html>

<script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script> <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script> <body>

<div id="mydiv"></div>

<script type="text/babel">

class Hello extends React.Component { render() { return <h1>Hello World!</h1>

}

}

ReactDOM.render(<Hello />, document.getElementById('mydiv')) </script>

</body>

</html>

**Setting up a React Environment**

First of all you need to download NodeJs for your operating system version.

https://nodejs.org/en/download/ npm(node packet manager once you download NodeJs

Then follow the following commands

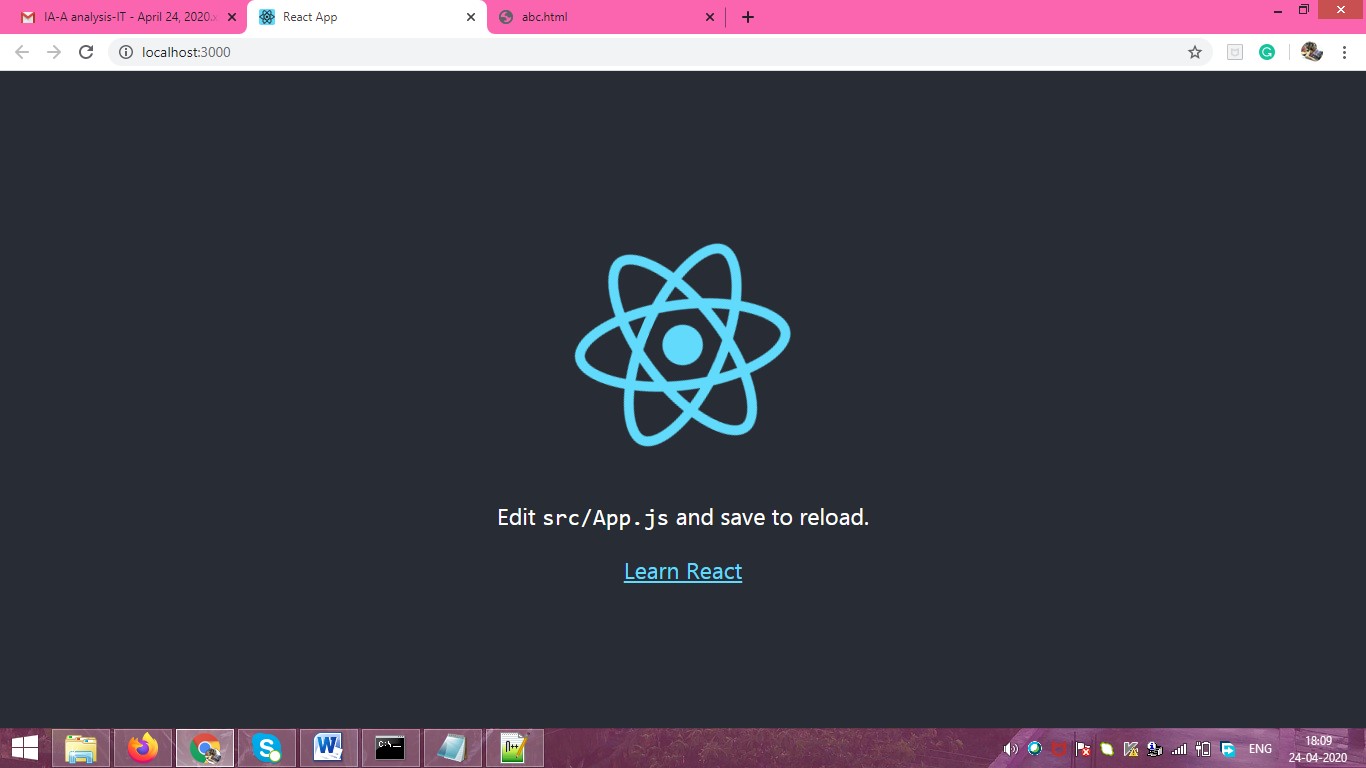
1. **run the command “**C:\Users\Your Name>npm install -g create-react-app”
2. **run the command to create an application name helloworld**

**“**C:\Users\Your Name>npx create-react-app Helloworld”

1. **run this command to get to the current directory** “C:\Users\Your Name>cd helloworld”
2. **run this command to start the react application** “C:\Users\Your

Name\helloworld >npm start”

A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.



**Modify the React Application**

Look in the helloworld directory, and you will find a src folder. Inside the src folder there is a file called App.js, open it and make changes to any HTML part.

You will be able to see the change on the newly opened browser

**App.js File**

import React, { Component } from 'react'; import logo from './logo.svg'; import './App.css';

class App extends Component { render() {

return (

<div className="App">

<header className="App-header">

<img src={logo} className="App-logo" alt="logo" />

<p>

Edit <code>src/App.js</code> and save to reload.

</p>

<p> Hi </p>

<a

className="App-link" href="https://reactjs.org" target="\_blank"

rel="noopener noreferrer"

>

Learn React

</a>

</header>

</div>

);

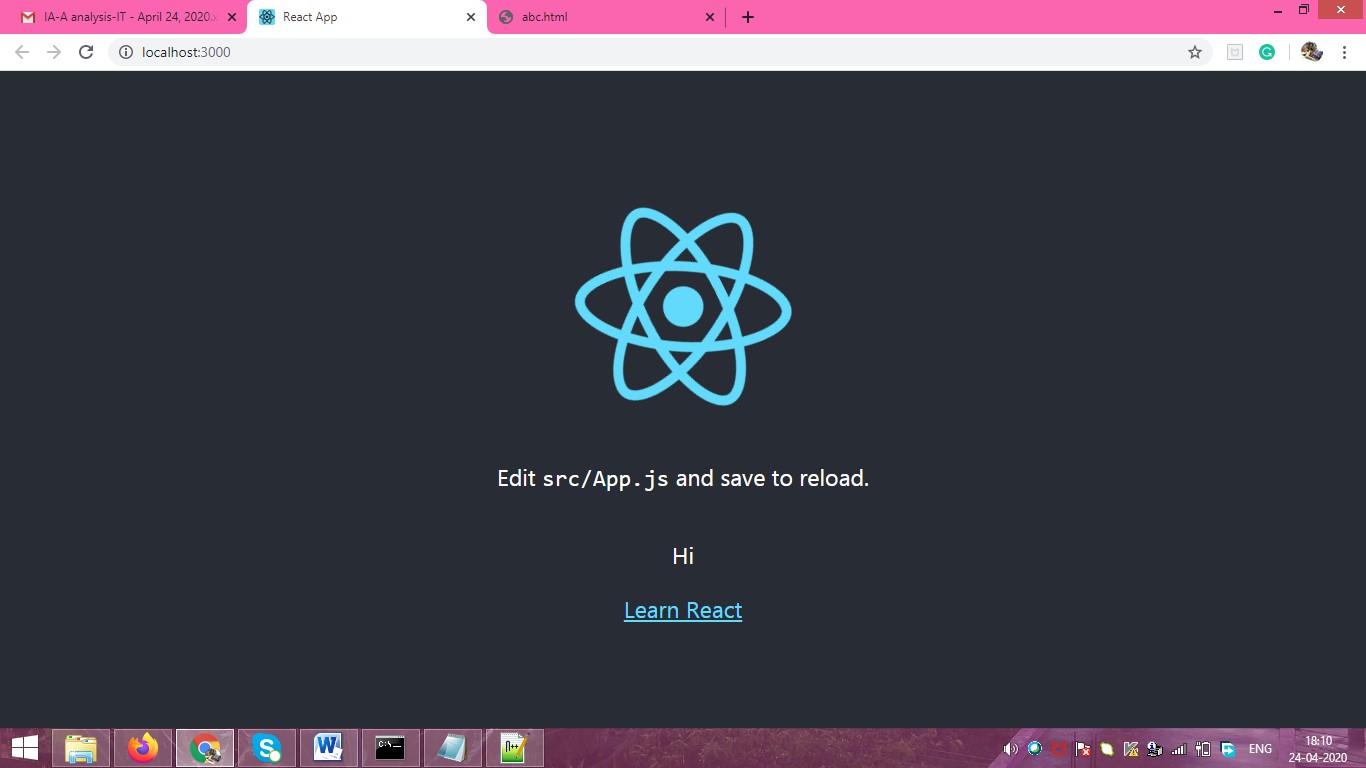
}

}

export default App;

in the file “Hi” is added inside paragraph tag.

The result browser screen is as follows



Activity:

* Create a web page to manipulate any one DOM Element using Angular js • Create a web page to manipulate any one CSS property using Angular JS
* Use React JS to change any of the HTML content.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Results: (Program printout with output)**

1. **Angular JS**

* Create a web page to manipulate any one DOM Element using Angular js and any one CSS property using Angular JS

**Program:**

**Index.html:-**

<!DOCTYPE html>

<html>

<head>

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

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">

<link rel="stylesheet" href="styles.css">

<title>Bill Calculator</title>

<style>

li a:hover {

background-color: lightgrey;

}

</style>

</head>

<body text="white">

<h1 style="text-align: center; margin-top:50px;">Bill Calculator</h1>

<div ng-app ng-controller="OrderFormController">

<form>

<ul>

<div>

<h5 style="margin-left: 10px; display: inline;">ITEM</h5>

<h5 style="margin-left: 410px; display:inline;">PRICE</h5>

<h5 style="margin-left: 10px;display:inline;">QUANTITY</h5>

</div>

<li ng-repeat="service in services" ng-click="toggleActive(service)" ng-class="{active:service.active}">

{{service.name}} <span>Rs.{{service.price}}

<button class="btn btn-light" ng-click="increaseItemAmount(service)" ng-model="myCheck">+</button>

{{service.quantity}}

<button class="btn btn-light" ng-click="decreaseItemAmount(service)">-</button>

</span>

</li>

</ul>

<div class="total">

<h4 style="display: inline; margin-left:80px; font-weight:bold;">TOTAL:</h4> <h4 style="display: inline; margin-left: 145px; font-weight:bold;">Rs.{{total()}}</h4>

</div>

</form>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>

<script src="script.js"></script>

</div>

<div style="text-align:center; margin-bottom: 50px;">

<input type="submit" class="btn btn-dark" id="submitbtn" value="Confirm Order">

</div>

</body>

<script>

document.getElementById("submitbtn").onclick = function formSubmit(){

alert("Your Bill Amount was Calculated!!");

};

document.getElementById("submitbtn").onmousedown = function formSubmit(e){

e.target.style.background = "rgb(255, 184, 184)";

e.target.style.border="3px solid grey";

};

document.getElementById("submitbtn").onmouse = function formSubmit(e){

e.target.style.removeProperty('border');

e.target.style.removeProperty('background');

};

</script>

</html>

**Script.js:**

function OrderFormController($scope){

$scope.services = [

{

name: 'Pen',

price: 10,

quantity:0,

active:false

},{

name: 'Mobile Cover',

price: 150,

quantity:0,

active:false

},{

name: 'Toys',

price: 250,

quantity:0,

active:false

},{

name: 'Graph Papers',

price: 5,

quantity:0,

active:false

},{

name: 'Decoration Items',

price: 200,

quantity:0,

active:false

},{

name: 'Book',

price: 400,

quantity:0,

active:false

},{

name: 'Notebooks',

price: 60,

quantity:0,

active:false

}

];

// $scope.toggleActive = function(s){

// s.active = !s.active;

// };

$scope.increaseItemAmount = function(item) {

item.quantity++;

item.active= true;

}

$scope.decreaseItemAmount = function(item) {

item.quantity--;

if (item.quantity <= 0) {

item.quantity = 0;

item.active = false;

} else {

item.active = true;

}

}

$scope.total = function(){

var total = 0;

angular.forEach($scope.services, function(s){

if (s.active){

total+= s.price\*s.quantity;

}

});

return total;

};

}

**Styles.css:-**

/\* For mobile phones: \*/

[class\*="col-"] {

width: 100%;

}

@media only screen and (min-width: 600px) {

/\* For tablets: \*/

.col-s-1 {width: 8.33%;}

.col-s-2 {width: 16.66%;}

.col-s-3 {width: 25%;}

.col-s-4 {width: 33.33%;}

.col-s-5 {width: 41.66%;}

.col-s-6 {width: 50%;}

.col-s-7 {width: 58.33%;}

.col-s-8 {width: 66.66%;}

.col-s-9 {width: 75%;}

.col-s-10 {width: 83.33%;}

.col-s-11 {width: 91.66%;}

.col-s-12 {width: 100%;}

}

@media only screen and (min-width: 768px) {

/\* For desktop: \*/

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

}

\*{

margin:0;

padding:0;

}

a, a:visited {

outline:none;

color:#389dc1;

}

a:hover{

text-decoration:none;

}

section, footer, header, aside, nav{

display: block;

}

/\*-------------------------

The order form

--------------------------\*/

form{

background-color:#6497b1;

border-radius: 2px;

box-shadow: 0 1px 1px #ccc;

width: 800px;

padding: 35px 60px;

margin: 50px auto;

}

form h1{

color:#fff;

font-size:64px;

font-family:'Cookie', cursive;

font-weight: normal;

line-height:1;

text-shadow:0 3px 0 rgba(0,0,0,0.1);

}

form ul{

width: 680px;

list-style:none;

color:#fff;

font-size:20px;

font-weight:bold;

text-align: left;

margin:20px 0 15px;

}

form ul li {

padding:15px;

background-color:#03396c;

margin-bottom: 10px;

box-shadow:0 1px 1px rgba(0,0,0,0.1);

cursor:pointer;

}

form ul li span{

float:right;

}

form ul li.active{

background-color:#8ec16d;

}

div.total{

border-top:1px solid rgba(255,255,255,0.5);

padding:15px 30px;

font-size:20px;

font-weight:bold;

text-align: left;

color:#fff;

}

div.total span{

float:right;

}

**Output:**

**The initial Webpage Looks like This:**

Graphical user interface

Description automatically generated

**When we click on the “+” button the quantity gets increased and according to the corresponding prices the final bill is Calculated. The Item row turns Green as the ‘+’ Button is pressed.**

**Graphical user interface, application

Description automatically generated**

**When we click on the “-” button the quantity gets decreased and according to the corresponding prices the final bill is Calculated. The Item row turns Red again as the quantity becomes 0.**

Graphical user interface, application

Description automatically generated

**When we click the “Confirm Order” button we get an alert.**

**Graphical user interface, application, website

Description automatically generated**

1. **React JS:**

* Use React JS to change any of the HTML content.

**Program:**

**Index.js:-**

import React from 'react';

import ReactDOM from 'react-dom';

import './App.css';

import './bootstrap.min.css'

class MyForm extends React.Component {

constructor(props) {

super(props);

this.state = {

fname: '',

lname: '',

age: null,

gender: 'S',

accept: '',

checked: false,

fnameErr: '',

lnameErr: '',

genderErr: '',

ageErr: '',

acceptErr: ''

};

}

myChangeHandler = (event) => {

let nam = event.target.name;

let val = event.target.value;

let checked = event.target.checked;

let err = '';

if (nam === "fname") {

if (val == "") {

err = <strong>\*First Name is required</strong>;

}

this.setState({fnameErr: err});

}

if (nam === "lname") {

if (val == "") {

err = <strong>\*Last Name is required</strong>;

}

this.setState({lnameErr: err});

}

if (nam === "gender") {

if (val == "S") {

err = <strong>\*Please choose appropriate option</strong>;

}

this.setState({genderErr: err});

}

if (nam === "age") {

if (val == "") {

err = <strong>\*Age is required</strong>;

} else if (val !="" && !Number(val)) {

err = <strong>\*Your age must be a number</strong>;

} else if (val > 150) {

err = <strong>\*Enter Valid Age</strong>;

}

this.setState({ageErr: err});

}

if (nam === "accept") {

this.setState({[nam]: checked});

if (checked == false) {

err = <strong>\*Please accept the terms and conditions</strong>;

}

this.setState({acceptErr: err});

}

this.setState({[nam]: val});

}

mySubmitHandler = (event) => {

event.preventDefault();

let fname = this.state.fname;

let lname = this.state.lname;

let age = this.state.age;

let gender = this.state.gender;

let accept = this.state.accept;

let err = '';

if (fname == "") {

err = <strong>\*First Name is required</strong>;

this.setState({fnameErr: err});

}

if (lname == "") {

err = <strong>\*Last Name is required</strong>;

this.setState({lnameErr: err});

}

if (gender == "S") {

err = <strong>\*Please choose appropriate option</strong>;

this.setState({genderErr: err});

}

if (age == null) {

err = <strong>\*Age is required</strong>;

this.setState({ageErr: err});

} else if (!Number(age)) {

err = <strong>\*Your age must be a number</strong>;

this.setState({ageErr: err});

} else if (age > 150) {

err = <strong>\*Enter Valid Age</strong>;

this.setState({ageErr: err});

}

if (accept == false) {

err = <strong>\*Please accept the terms and conditions</strong>;

this.setState({acceptErr: err});

}

if (this.state.fnameErr == '' && this.state.lnameErr == '' && this.state.genderErr == '' && this.state.ageErr == '' && this.state.acceptErr == '') {

alert('Successfully Signed in');

}

}

reset = () => {

let err = '';

this.setState({fnameErr: err});

this.setState({lnameErr: err});

this.setState({genderErr: err});

this.setState({ageErr: err});

this.setState({acceptErr: err});

}

render() {

return (

<div className="con d-flex flex-column justify-content-center align-items-center">

<div className="main py-5">

<center>

<h1 className="pb-3" id="heading">Hello {this.state.fname} {this.state.lname}</h1>

</center>

<div className="d-flex">

<form id="myForm" onSubmit={this.mySubmitHandler}>

<img src="https://1000logos.net/wp-content/uploads/2021/04/Facebook-logo.png" width="500" />

<div className="mb-3 d-flex">

<div className="me-3 width">

<label className="form-label" for="fname">First name:</label>

<input className="form-control" type="text" id="fname" name="fname" onChange={this.myChangeHandler} />

</div>

<div className="width">

<label className="form-label" for="lname">Last name:</label>

<input className="form-control" type="text" id="lname" name="lname" onChange={this.myChangeHandler} />

</div>

</div>

<div className="mb-3 d-flex">

<div className="me-3 width">

{this.state.fnameErr}

</div>

<div className="width">

{this.state.lnameErr}

</div>

</div>

<div className="mb-3">

<label className="form-label">Gender:</label>

<select className="form-select" name="gender" value={this.state.gender} onChange={this.myChangeHandler}>

<option value="S">Select</option>

<option value="M">Male</option>

<option value="F">Female</option>

<option value="O">Other</option>

</select>

{this.state.genderErr}

</div>

<div className="d-flex mb-3">

<div className="me-3 width">

<label className="form-label" for="start">Date Of Birth:</label>

<input className="form-control" type="date" id="start" name="trip-start" value="2020-02-18" min="2020-02-18" max="2020-06-31" />

</div>

<div className="width">

<label className="form-label">Time of Birth:</label>

<input className="form-control" type="time" id="time" name="time" value="12:00" autocomplete="off" />

</div>

</div>

<div className="mb-3">

<label className="form-label" for="age">Age:</label>

<input className="form-control" type="text" id="age" name="age" onChange={this.myChangeHandler} />

{this.state.ageErr}

</div>

<div className="mb-3">

<input className="form-check-input me-1" type="checkbox" name="accept" onChange={this.myChangeHandler} checked={this.state.check} />

<label className="form-check-label" for="accept">I accept the terms and conditions.</label>

<br />{this.state.acceptErr}

</div>

<input className="btn btn-primary me-3" type="submit" id="submitbtn" value="Sign Up" />

<input className="btn btn-danger" type="reset" id="resetbtn" value="Reset" onClick={this.reset}/>

</form>

</div>

</div>

</div>

);

}

}

ReactDOM.render(<MyForm />, document.getElementById('root'));

**App.css:-**

.main {

border: 2px solid black;

padding-left: 80px;

padding-right: 80px;

border-radius: 30px;

}

.con {

height: 150vh;

}

h2{

color: rgb(133, 12, 82);

}

.width {

width: 50%;

}

**Index.html:-**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>React App</title>

</head>

<body>

<div id="root"></div>

</body>

<script>

document.getElementById("submitbtn").onclick = function formSubmit(){

alert("You are successfully Signed In!");

};

document.getElementById("submitbtn").onmousedown = function formSubmit(e){

e.target.style.background = "rgb(255, 184, 184)";

e.target.style.border="3px solid grey";

};

document.getElementById("submitbtn").onmouse = function formSubmit(e){

e.target.style.removeProperty('border');

e.target.style.removeProperty('background');

};

</script>

</html>

**Output:**

* **The Initial Webpage Looks like this**:Graphical user interface, application

  Description automatically generated
* **As, we enter the First Name and Last Name the Header changes like this:-**

**Graphical user interface, application

Description automatically generated**

* **If some field is left out we get an error:**

Graphical user interface, application

Description automatically generated

* **When we enter an invalid age we get the following error:**

Graphical user interface, text, application

Description automatically generated

* **When we enter age in wrong format we get the following error:**

Graphical user interface, text, application

Description automatically generated with medium confidence

* **After filling out the form and clicking on the Sign Up Button ,we get an alert:**

Graphical user interface, application

Description automatically generated

**Graphical user interface, application

Description automatically generated**

* **On clicking The “Reset” button all the entered information gets erased:-**

**Graphical user interface, application

Description automatically generated**

**Outcomes:**

**CO4:** Implement web application using React JS, Angular JS, Json and CBOR

**Conclusion: (Conclusion to be based on the outcomes achieved)**

In this experiment, we learned Angular Js and created a webpage to manipulate DOM element and CSS property using the same.

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **References:**

**Books/ Journals/ Websites:**

* http://www.w3schools.com
* <https://www.tutorialspoint.com/angularjs/angularjs_tutorial.pdf>
* [https://angularjs.org](https://angularjs.org/)