HTML – 4

Ex. No. : 1. a.

Aim:

To implement HTML 4: Horizontal rule, Links, Image Insertion, Ordered and unordered list, Internal hyper linking, Meta, Simple table, Form, Frames, and Hotspot creation.

Procedure:

1. Define the HTML page using HTML tag.
2. Internal linking is created using <a href="#"></a> tag.
3. Image inserted using <img src=""> tag.
4. The Ordered and unordered list are used to show user the services offered by the site.
5. The contact information is displayed to the user using the table tag insert the data into the table using TR to add data to table use TD tag.
6. The uses information is received by the Form tag.

Source Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Home Page</title>

<link rel="stylesheet" href="./style.css">

<style>

header {

position: relative;

}

img {

width: 100%;

height: 75%;

}

h1 {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<header>

<img src="aptimg.jpeg" alt="Banner image" usemap="#apartment-map">

        <!-- </a> -->

        <map name="apartment-map">

            <area shape="rect" coords="0, 0, 1280,330" href="https://youtube.com">

        </map>

<h1 style="color: aliceblue; font-size: 90px;" align="center">Welcome to online code editor</h1>

</header>

<nav ng-app="menuApp" ng-controller="menuController">

<ul>

      <li ng-repeat="menu in menus"><a href="{{ menu.link }}">{{ menu.title }}</a></li>

<li><button onclick = darkmode()>Dark mode</button></li>

    </ul>

</nav>

<main id = "main">

<section id = "text" class = "sec">

<a href="./text.html">

<h2 >START CODING</h2>

</a>

</section>

<section id = "secser" class = "sec">

<h2 id="services">Services</h2>

<ul>

<li>Code editing</li>

<li>Compiling</li>

<li>Code saving</li>

</ul>

</section>

<section id = "secabt" class = "sec">

<h2 id = "Abtus">About Us</h2>

<p>This is a online code editor that supports all major programming languages such as C, C++, python, Java etc...</p>

</section>

<section id = "seccon" class = "sec">

<h2 id="contact">Contact Us</h2>

<table border="1">

<tr>

<td>Email:</td>

<td>info@onlinecodeeditor.com</td>

</tr>

<tr>

<td>Phone:</td>

<td>555-555-5555</td>

</tr>

<tr>

<td>Address:</td>

<td>123 Private St, siValley, USA</td>

</tr>

</table>

</section>

</main>

<footer>

<p>&copy; 2023 Apartment Management Portal</p>

</footer>

</body>

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

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

<script>

    // AngularJS code

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

    app.controller("menuController", function($scope) {

      $scope.menus = [

        {

          title: "Home",

          link: "#home"

        },

        {

          title: "About",

          link: "./item1.html"

        },

        {

          title: "Rules",

          link: "item2.html"

        },

        {

          title: "Contact",

          link: "#contact"

        },

{

          title: "Register",

          link: "./regist.html"

        },

{

          title: "Angular calculator",

          link: "./angcalc.html"

        },

{

          title: "Search User",

          link: "./service.html"

        },

{

          title: "Languages offered",

          link: "./layout.html"

        },

{

          title: "View poster",

          link: "./anim.html"

        }

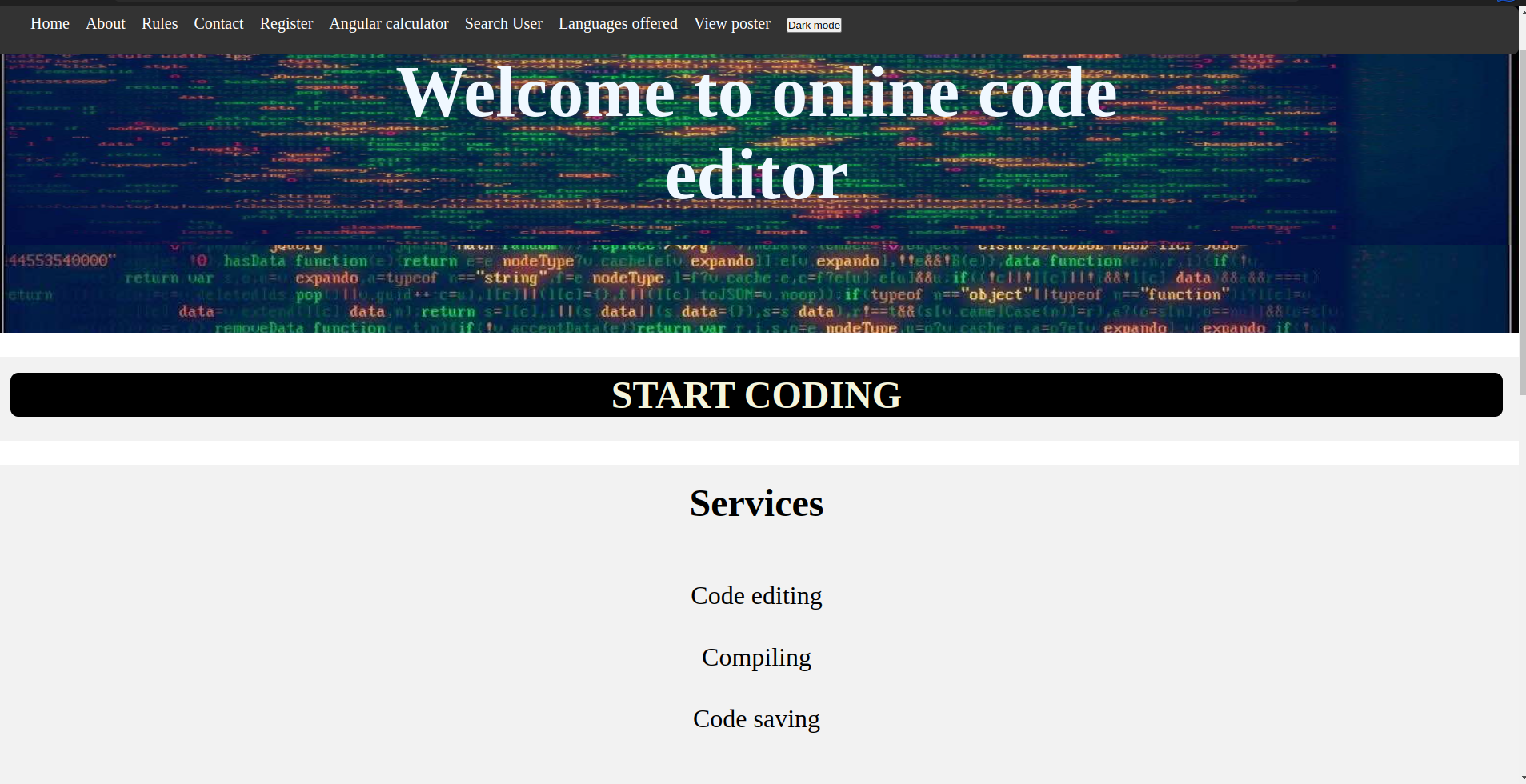
      ];

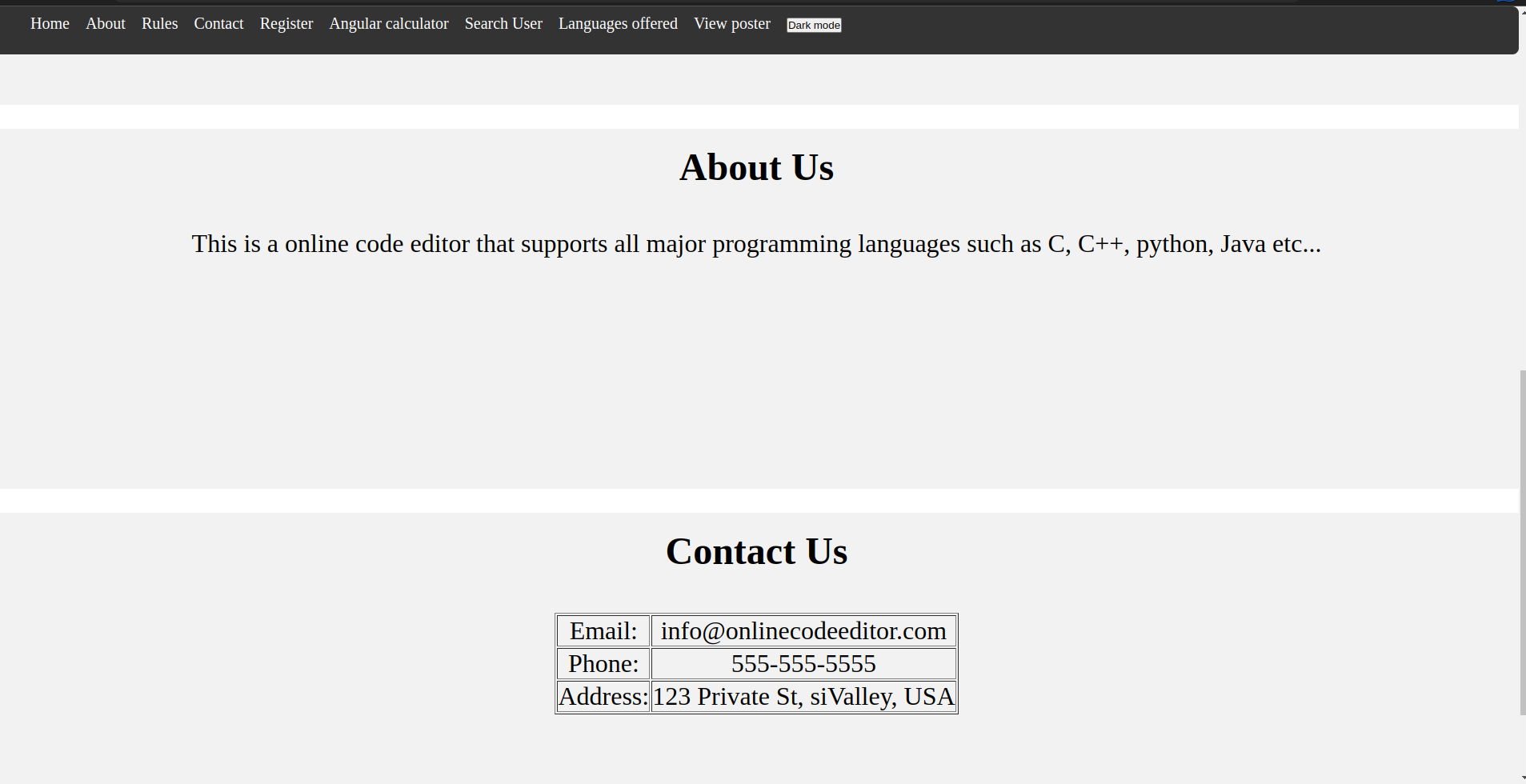
    });

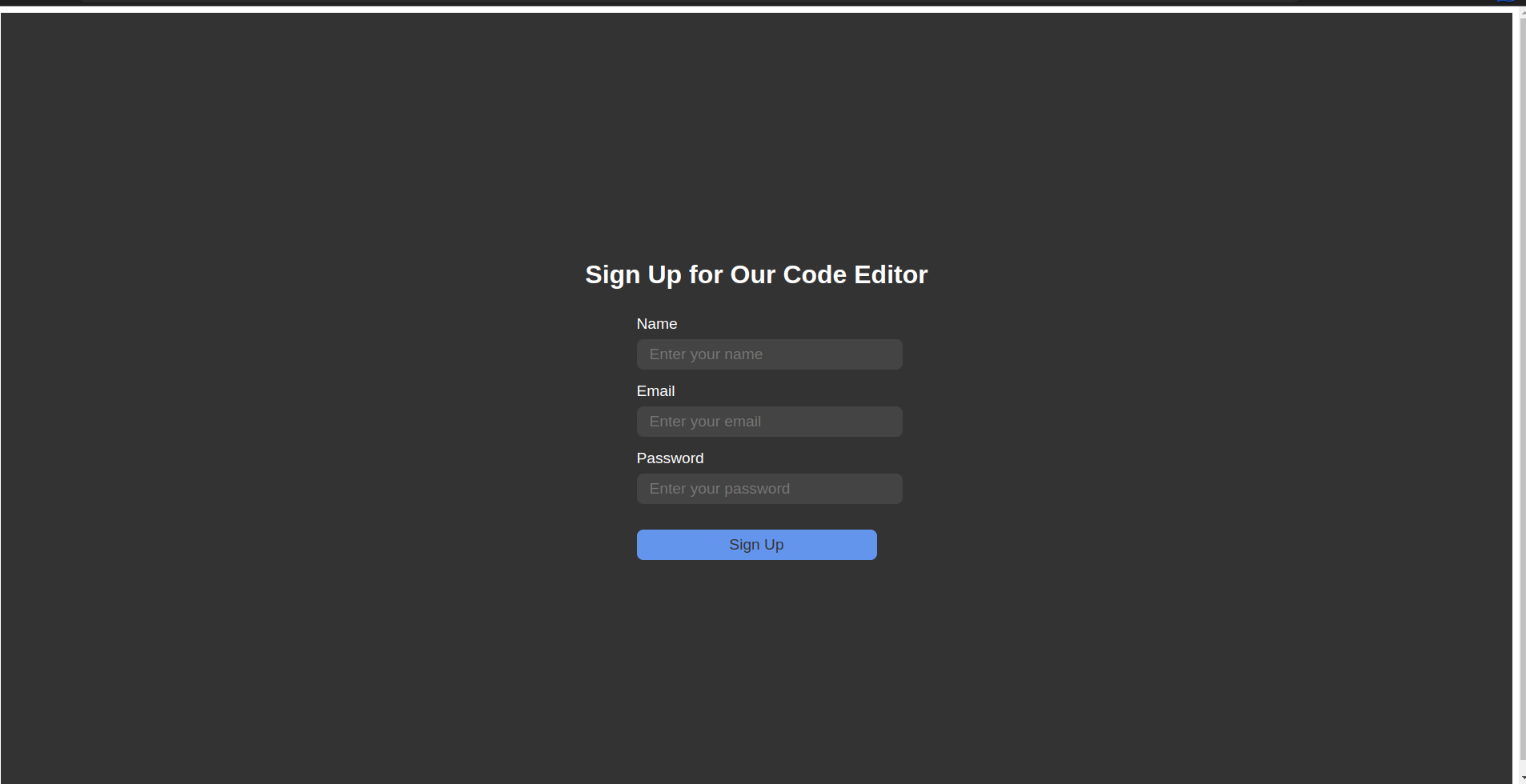
  </script>

</html>

Output:







Result:

The program to implement HTML4 elements was executed successfully.

Ex. No. : 1.b

Aim:

CSS 2: types of CSS, selectors, box model, positioning, layout.

Procedure:

1. CSS is written using style tag in the HTML pages to style the internal hyperlink.
2. inline CSS is written inside the HTML tag using the style attribute.
3. external CSS is written in another file with extension.css then it is included in the HTML file using link tag.
4. the dot selector denotes the class attribute of a tag.
5. # selector is used for the tag which has id as attribute.
6. various properties position background colour width height a used.
7. use margin, border, padding to align elements.
8. use position: fixed to the number.
9. use justify-content: space-between Twilight elements inside nearby

Source Code:

/\* Set default styles for all elements \*/

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

/\* Set styles for header \*/

header {

background-image: url("aptimg.jpeg");

background-repeat: no-repeat;

background-size: cover;

padding-top: 60px;

/\* position: fixed; \*/

/\* top: 0; \*/

width: 100%;

height: 500px;

}

#secser

{

margin-top:30px;

background-color:#f2f2f2;

padding: 20px;

width: 100%;

height: 450px;

font-size: 200%;

text-align: center;

}

#text

{

margin-top:30px;

background-color:#f2f2f2;

padding: 20px;

width: 100%;

font-size: 200%;

text-align: center;

}

#text h2

{

background-color:black;

border-radius: 10px;

color:beige

}

a

{

text-decoration: none;

}

#secabt

{

margin-top:30px;

background-color:#f2f2f2;

padding: 20px;

width: 100%;

height: 450px;

font-size: 200%;

text-align: center;

}

#seccon

{

margin-top:30px;

background-color:#f2f2f2;

padding: 20px;

width: 100%;

height: 450px;

font-size: 200%;

text-align: center;

}

#seccon table

{

margin: auto;

margin-top: 50px;

}

#secser ul

{

list-style-type: none;

}

#secser h2

{

margin-bottom: 50px;

}

#secabt h2

{

margin-bottom: 50px;

}

#secser li

{

padding: 20px;

}

/\* Set styles for navigation menu \*/

nav {

background-color: #333;

color: #fff;

/\* display: flex; \*/

justify-content: space-between;

padding: 10px;

position: fixed;

top: 0px;

width: 100%;

height:60px;

font-size: 20px;

border-radius: 8px;

}

nav ul {

display: flex;

list-style-type: none;

margin: 0;

}

nav li {

margin: 0 10px;

}

nav a {

color: #fff;

text-decoration: none;

}

/\* Set styles for main content \*/

h2 {

margin-bottom: 10px;

}

/\* Set styles for footer \*/

footer {

background-color: #f2f2f2;

bottom: 0;

padding: 10px;

/\* position: fixed; \*/

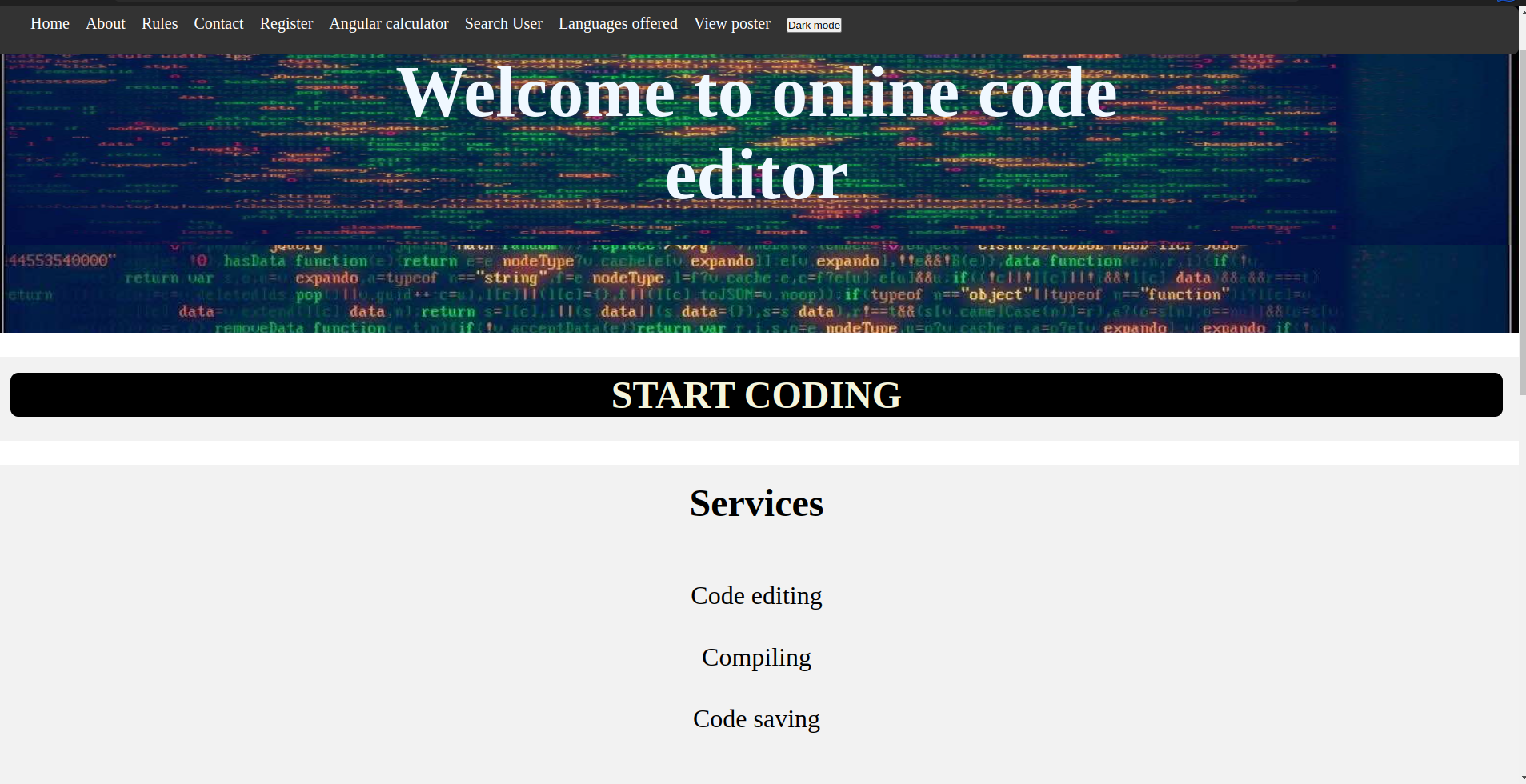
width: 100%;

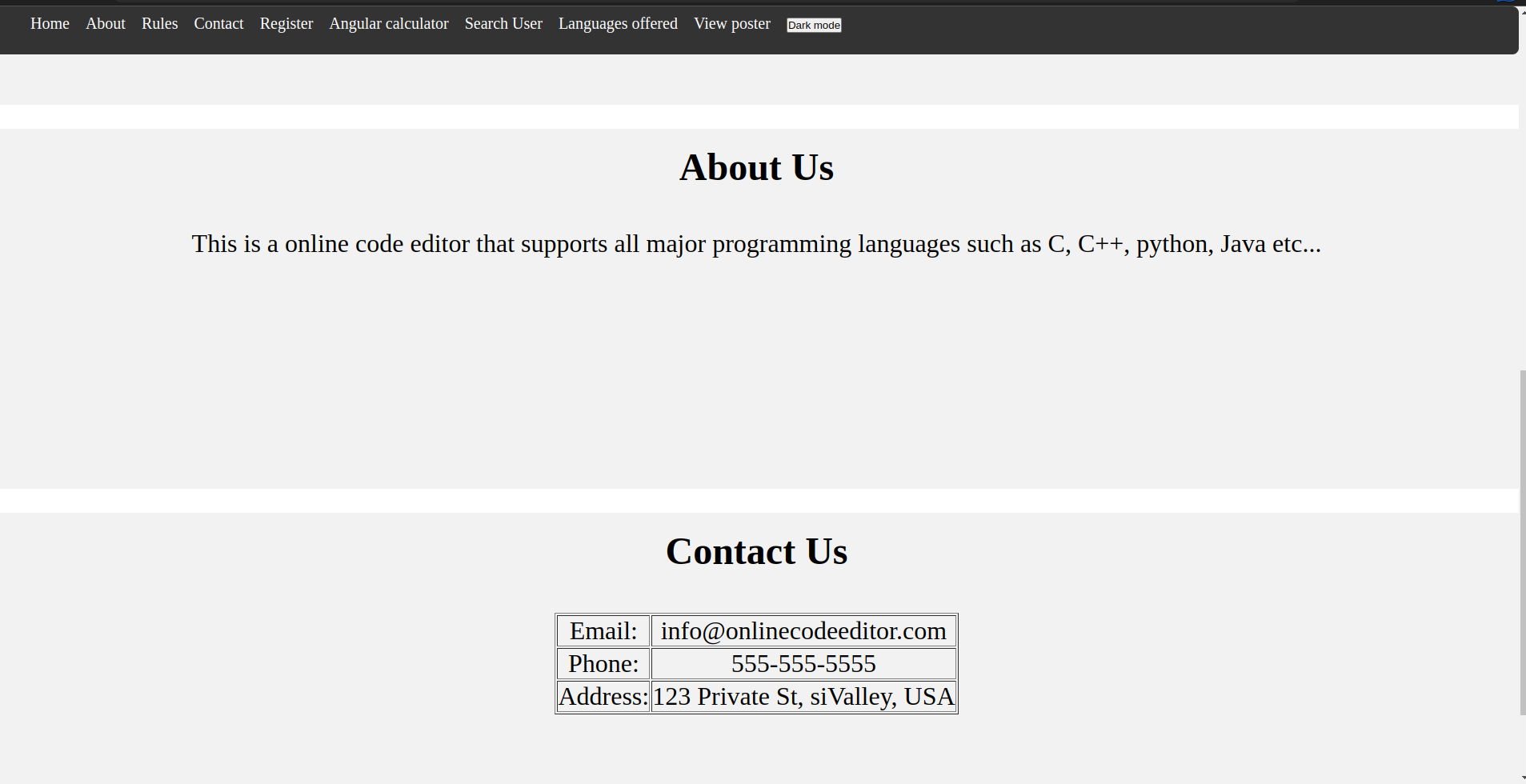
font-size: 150%;

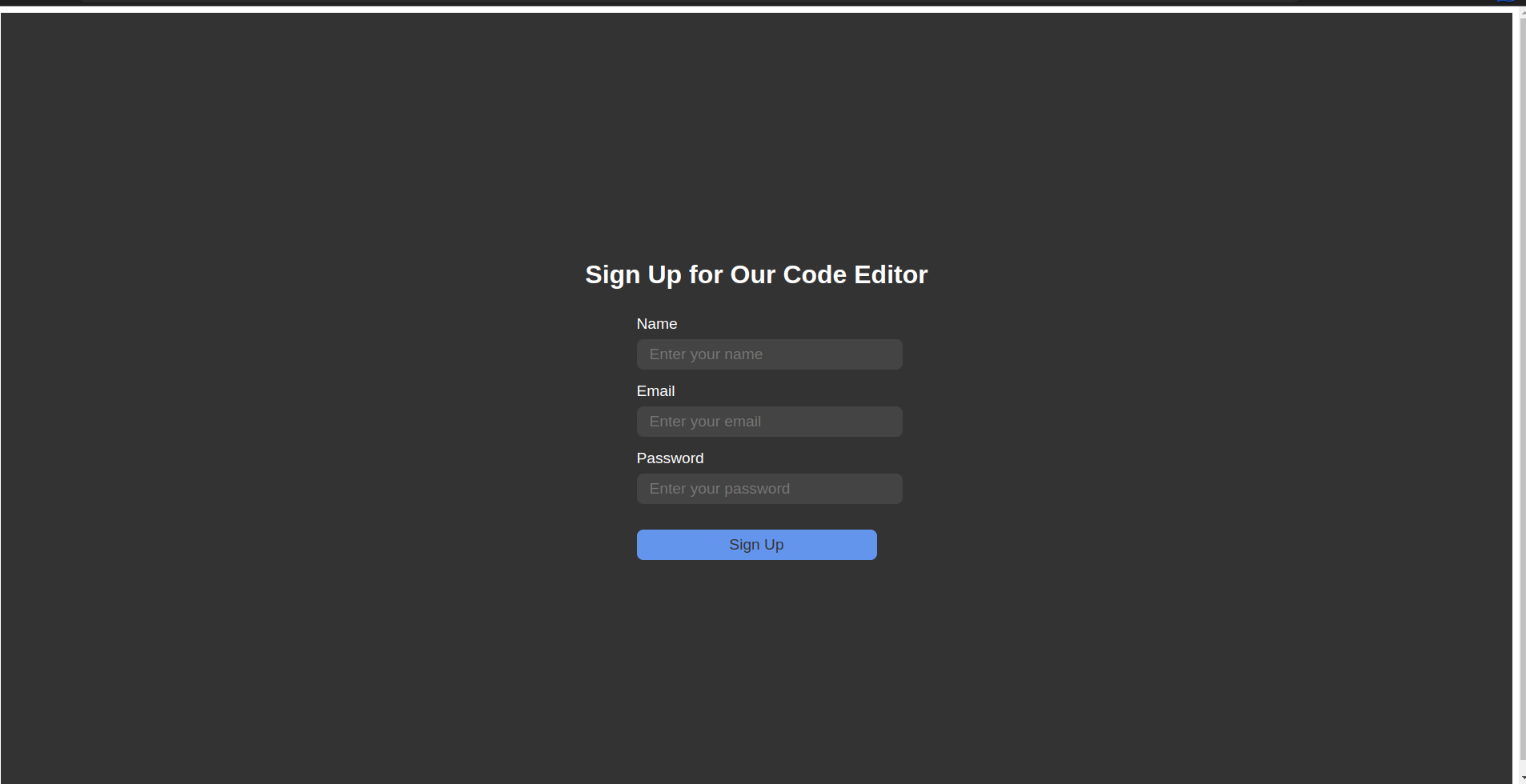
text-align: center;

}

Output:







Result:

The code to implement CSS2 features was executed successfully.

Ex. No. : 1.c.

Aim:

HTML5: new structural elements, video, audio, form creation, Canvas API, SVG, geo location.

Procedure:

1. use new structural elements such as header footer Nav section form.
2. use the audio tag to insert audio file.
3. use video tag to insert video file.
4. use <button onclick = getlocation()> </button> to get the geolocation.
5. The geo location function used the inbuilt function get current position method.
6. then using inner HTML the geo location is shown to the user in the body of the web page.

Source Code:

<!DOCTYPE html>

<html>

<head>

    <title>Canvas Example</title>

    <link rel="stylesheet" href="./geo.css">

</style>

</head>

<body>

<h1>Cnavas API example</h1>

    <canvas id="myCanvas" width="500" height="500"></canvas>

    <script>

        var canvas = document.getElementById("myCanvas");

        var ctx = canvas.getContext("2d");

        // Create a linear gradient from red to yellow

        var gradient = ctx.createLinearGradient(0, 0, canvas.width, canvas.height);

        gradient.addColorStop(0, "blue");

        gradient.addColorStop(1, "green");

        // Fill a rectangle with the gradient

        ctx.fillStyle = gradient;

        ctx.fillRect(50, 50, 400, 400);

    </script>

    <h1>Click the button to get your coordinates.</h1>

    <button onclick="getLocation()">Try It</button>

    <h2 id="demo"></h2>

    <script>

    var x = document.getElementById("demo");

    function getLocation() {

    if (navigator.geolocation) {

        navigator.geolocation.getCurrentPosition(showPosition);

    } else {

        x.innerHTML = "Geolocation is not supported by this browser.";

    }

    }

    function showPosition(position) {

    x.innerHTML = "Latitude: " + position.coords.latitude +

    "<br>Longitude: " + position.coords.longitude;

    }

    </script>

    <h1>SVG Example</h1>

<br>

    <svg viewBox="0 0 100 100">

        <circle cx="50" cy="50" r="58" fill = "white" />

<rect x="10" y="10" width="80" height="80" fill = "black" />

    </svg>

</body>

</html>

Output:



Result:

The program HTML5 elements was executed successfully.

Ex.No. : 1. d.

Aim:

To use CSS3 to implement CSS3 features.

Procedure:

1. Create a div element and assign it the class "box"
2. Set the width and height of the div to 200px
3. Set the background color of the div to light blue
4. Set the border of the div to 2px solid black
5. Set the text color of the div to white
6. Set the text alignment of the div to center
7. Set the font size of the text in the div to 24px
8. Set the text shadow of the text in the div to a 2px blur and a 1px offset in black
9. Use the hover selector to apply a transition to the div on hover
10. Set the transform of the div on hover to rotate 360 degrees

Source Code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <link rel="stylesheet" href="./style.css">

</head>

<body>

    <div class="container">

        <h1 class="text-effect">Hello, World!</h1>

    </div>

    <div class="container">

        <button class="btn">Click me</button>

    </div>

</body>

</html>

.container {

    border: 2px solid black;

    width: 300px;

    height: 200px;

    display: flex;

    justify-content: center;

    align-items: center;

  }

  .text-effect {

    font-size: 3em;

    text-align: center;

    color: white;

    background-color: black;

    padding: 20px;

    border-radius: 10px;

    animation: text-animation 2s infinite;

  }

  @keyframes text-animation {

    0% {

      transform: scale(1);

    }

    50% {

      transform: scale(1.1);

    }

    100% {

      transform: scale(1);

    }

  }

  .btn {

    background-color: black;

    color: white;

    padding: 10px 20px;

    border: none;

    border-radius: 5px;

    transition: all 0.5s ease-in-out;

  }

  .btn:hover {

    background-color: white;

    color: black;

    transform: scale(1.2);

  }

Output:





Result:

The program to implement css3 features was executed successfully.

Ex. No. : 2

Aim: HTML form validation using regular expressions and DOM manipulation.

Procedure:

1. create a simple HTML form using <form></form> tag and use different input methods.
2. give name for each input element.
3. in the JS file access all input elements using document.queryselector('input[name=""]');
4. define a regular expression for each field.
5. used form.adEventlistener to call preventDefault function when the form the submitted.
6. inside the call back function use test() function to validate each input field using respective regular expression.
7. if invalid, alert the user with appropriate message.
8. if all feels are valid submit form using form.submit().

Source Code:

Index.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Registration Form</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f2f2f2;

        }

        form {

            width: 500px;

            margin: auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 10px;

            box-shadow: 0px 0px 10px #ccc;

        }

        h1 {

            text-align: center;

            margin-bottom: 20px;

        }

        label {

            display: inline-block;

            width: 100px;

            margin-bottom: 10px;

            font-weight: bold;

        }

        input[type="text"] {

            padding: 5px;

            width: 100%;

            border-radius: 5px;

            border: 1px solid #ccc;

        }

        input[type="submit"], input[type="reset"] {

            display: block;

            margin-top: 20px;

            padding: 10px 20px;

            background-color: #4CAF50;

            color: #fff;

            border: none;

            border-radius: 5px;

            cursor: pointer;

        }

        input[type="submit"]:hover, input[type="reset"]:hover {

            background-color: #3e8e41;

        }

    </style>

</head>

<body>

    <h1>Registration Form</h1>

    <form action="" method="post">

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

        <input type="text" name="name" placeholder="Enter your name">

        <br>

        <label for="age">Age:</label>

        <input type="text" name="age" placeholder="Enter your age">

        <br>

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

        <input type="text" name="email" placeholder="Enter your email">

        <br>

        <label for="phone">Phone:</label>

        <input type="text" name="phone" placeholder="Enter your phone">

        <br>

        <label for="address">Address:</label>

        <input type="text" name="address" placeholder="Enter your address">

        <br>

        <label for="city">City:</label>

        <input type="text" name="city" placeholder="Enter your city">

        <br>

        <label for="state">State:</label>

        <input type="text" name="state" placeholder="Enter your state">

        <br>

        <label for="aadhar">Aadhar:</label>

        <input type="text" name="Aadhar" placeholder="Enter your Aadhar No.">

        <br>

        <input type="submit" name="submit" value="Submit">

        <input type="reset" value="Reset">

    </form>

    <script src="./app.js"></script>

</body>

</html>

App.js:

const form = document.querySelector('form');

const nameInput = document.querySelector('input[name="name"]');

const ageInput = document.querySelector('input[name="age"]');

const emailInput = document.querySelector('input[name="email"]');

const phoneInput = document.querySelector('input[name="phone"]');

const addressInput = document.querySelector('input[name="address"]');

const cityInput = document.querySelector('input[name="city"]');

const stateInput = document.querySelector('input[name="state"]');

const aadharInput = document.querySelector('input[name="Aadhar"]');

const nameRegex = /^[a-zA-Z ]{2,30}$/;

const ageRegex = /^(1[89]|[2-9]\d)$/;

const emailRegex = /^([a-zA-Z0-9.\_-]+)@([a-zA-Z0-9.-]+)\.([a-zA-Z]{2,5})$/;

const phoneRegex = /^[0-9]{10}$/;

const addressRegex = /^[a-zA-Z0-9\s,'-]\*$/;

const cityRegex = /^[a-zA-Z\s]+$/;

const stateRegex = /^[a-zA-Z\s]+$/;

const aadharRegex = /^[0-9]{12}$/;

form.addEventListener('submit', (e) => {

    e.preventDefault();

    if (!nameRegex.test(nameInput.value)) {

        alert('Please enter a valid name (2-30 characters, only letters and spaces)');

        return;

    }

    if (!ageRegex.test(ageInput.value)) {

        alert('Please enter a valid age (18-99 years)');

        return;

    }

    if (!emailRegex.test(emailInput.value)) {

        alert('Please enter a valid email address');

        return;

    }

    if (!phoneRegex.test(phoneInput.value)) {

        alert('Please enter a valid 10-digit phone number');

        return;

    }

    if (!addressRegex.test(addressInput.value)) {

        alert('Please enter a valid address (letters, numbers, spaces, commas, apostrophes, and hyphens only)');

        return;

    }

    if (!cityRegex.test(cityInput.value)) {

        alert('Please enter a valid city name (letters and spaces only)');

        return;

    }

    if (!stateRegex.test(stateInput.value)) {

        alert('Please enter a valid state name (letters and spaces only)');

        return;

    }

    if (!aadharRegex.test(aadharInput.value)) {

        alert('Please enter a valid Aadhar number (12 digits)');

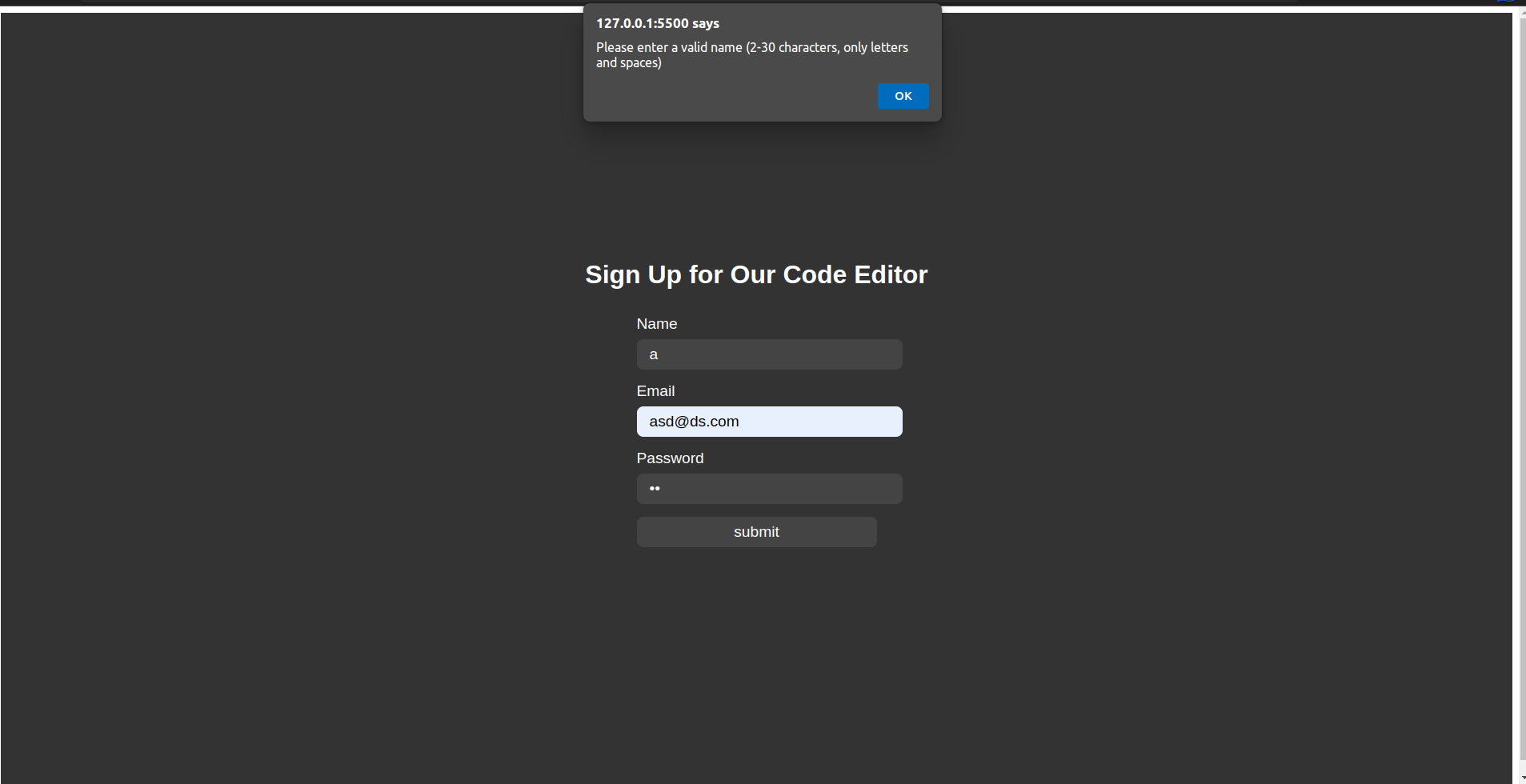
        return;

    }

    form.submit();

});

Output:



Result:

The program to implement HTML form validation using regular expressions was executed successfully.

Ex. No. : 2. b.

Aim:

To manipulate HTML’s Document Object Model(DOM) to change document markup on the happening of an event.

Procedure:

1. Start the HTML tag.
2. Start the body tag.
3. Include some element tags like p or h1 tag with some example text.
4. Add a button tag, which on clicking, calls a JavaScript function to change the appearance of the elements included in step 4.
5. Use internal JavaScript to define the function

Source Code:

function darkmode() {

const main = document.getElementById("main")

const sec = document.getElementsByTagName("section")

main.style.backgroundColor = "black"

main.style.color = "white"

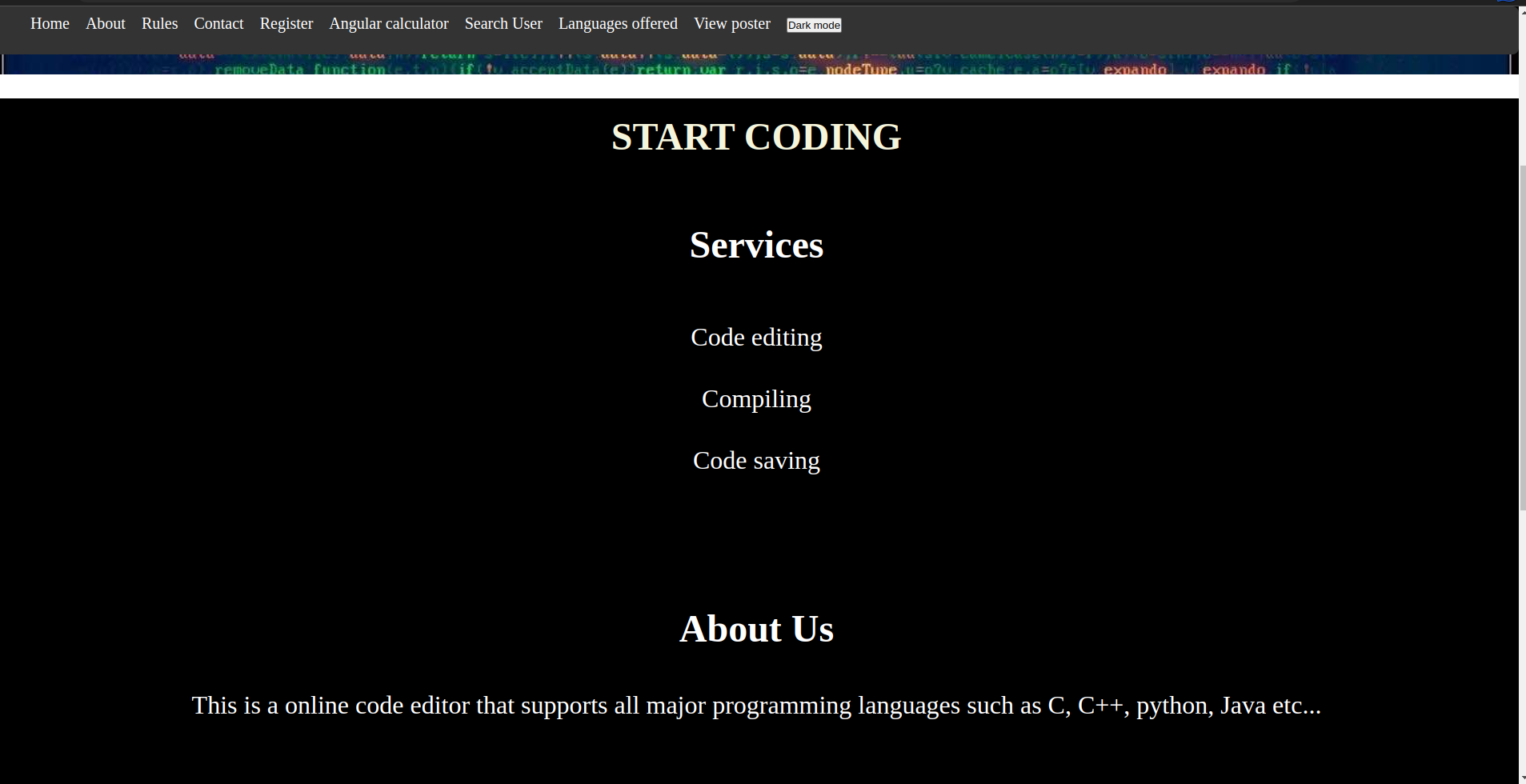
for(let i = 0; i<sec.length; i++) {

sec[i].style.backgroundColor = "black"

}

}

Output:



Result:

The program to implement dom manipulation was executed successfully.

Ex. No. : 2.c.

Aim:

To implement a simple messaging service using HTML, CSS and JavaScript.

Procedure:

1. Start the HTML tag.
2. Include some internal CSS to style the view.
3. Start the body tag.
4. Make use of div tags to create the chat window, the input tag and the send button.
5. Add a send button tag, which on clicking, calls a JavaScript function to display whatever was typed as input in the chat window.
6. Use internal JavaScript to define the function.

Source Code:

const { Vonage } = require('@vonage/server-sdk')

// Importing the http module

const http = require("http")

const express = require('express');

const path = require("path");

const app = express();

const bp = require("body-parser");

app.use(bp.urlencoded({extended: true}));

app.use(express.json())

app.listen(3000, function(){console.log("running...")});

app.get("/form", function(req, res)

{

res.sendFile(path.join(\_\_dirname, "./reg.html"));

}

)

app.post("/form", function(req, res)

{

console.log(req.body)

var ph = req.body.phone;

const vonage = new Vonage({

apiKey: "5c817c9e",

apiSecret: "U4hzyT1iJX7D0G1F"

})

const from = "Vonage APIs"

const to = ph

const text = 'oombu da saravana Kumar'

async function sendSMS() {

await vonage.sms.send({to, from, text})

.then(resp => { console.log('Message sent successfully'); console.log(resp); })

.catch(err => { console.log('There was an error sending the messages.'); console.error(err); });

}

sendSMS();

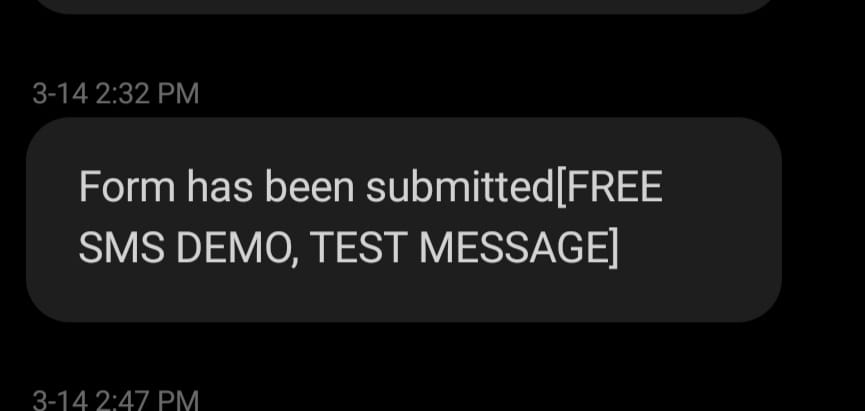
res.send("Message sent successfully");

}

)

Output:





Rsult:

The program to implement a messaging service is implemented successfully.

Ex. No. : 3. a.

Aim:

Angular program to create a simple calculator.

Procedure:

1. using ng-app the angular application is defined in the HTML page.
2. using <script src=""></script> the JS file is imported.
3. the ng-controller = calculateCntrl defined for getting Input and storing it in a model.
4. use the ng-model in the HTML to get input values.
5. use ng-click to trigger calculate() then the control will be transferred to js file.
6. by using switch case the operator is identified and the appropriate operation is performed.
7. the result is shown to the user using angular expression {{result}}.

Source Code:

Calc.html:

Calc.js:

angular.module('CalculatorApp', [])

    .controller('CalculatorController', function($scope) {

        $scope.result = function() {

        var a = $scope.a;

        var b = $scope.b;

        var operator = $scope.operator;

        var result;

        switch(operator) {

            case '+':

            result = a + b;

            break;

            case '-':

            result = a - b;

            break;

            case '\*':

            result = a \* b;

            break;

            case '/':

            result = a / b;

            break;

            case '%':

            result = a % b;

            break;

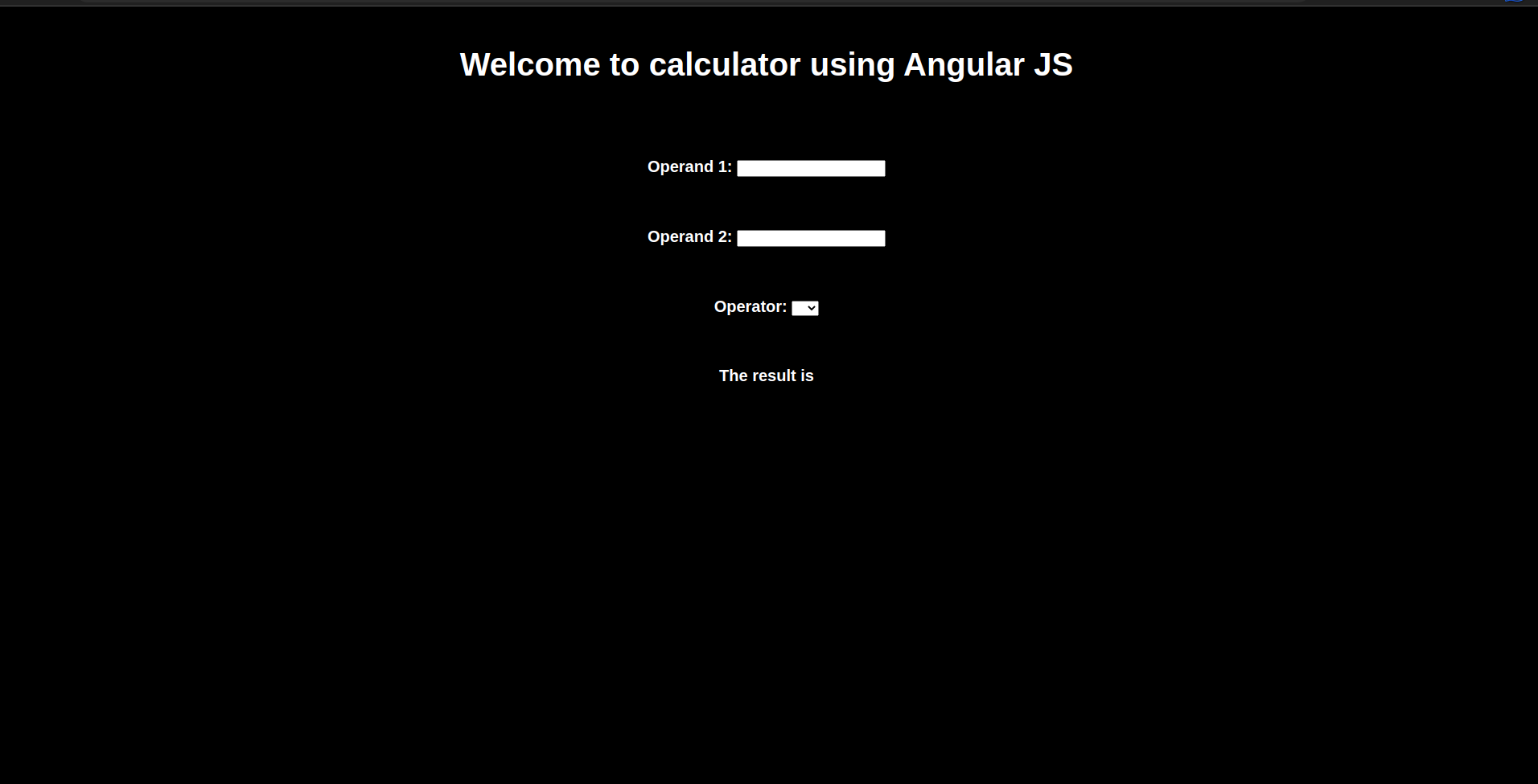
        }

        return result;

        };

    });

Output:



Result:

The program to create a simple calculator using Angular.js was executed successfully.

Ex.No. : 3.b.

Aim:

To write an AngularJS program to implement Components, Services and Controllers.

Procedure:

1. It defines a new AngularJS module called "myApp" using the **angular.module()** method.
2. It defines a service called "myService" using the **service()** method. The service has a single method called **getData()** that returns an array of names.
3. It defines a component called "myComponent" using the **component()** method. The component has a template that displays a list of names, and a controller that uses the **myService** service to get the list of names.
4. It defines a controller called "myController" using the **controller()** method. The controller has a single property called **title** that is used to display a heading in the HTML.
5. In the HTML, the **ng-app** directive initializes the AngularJS application, the **ng-controller** directive assigns the "myController" controller to the body element, and the **my-component** directive displays the "myComponent" component in the HTML.

Code:

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

app.service('exampleService', function($http) {

this.getData = function() {

return $http.get('./user.json');

}

this.sendData = function(data) {

// send data to server

}

});

app.controller('exampleController', function($scope, exampleService) {

$scope.data = {};

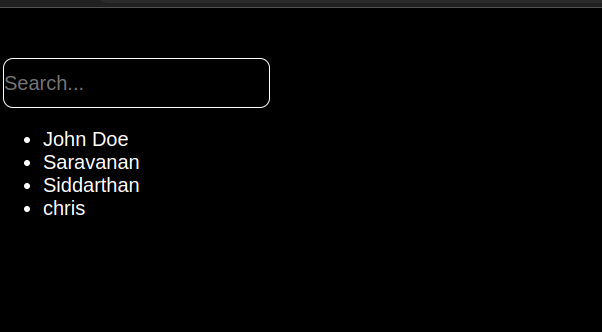
exampleService.getData().then(function(response) {

$scope.data = response.data;

});

});

Output:



Result:

The program was executed successfully.

Ex. No. : 3. c.

Aim:

Angular program to implement search to filter items.

Procedure:

1. inside the controller create an Array as an property of dollars scope object.
2. These array values are rendered in html page using unordered list.
3. use ng-repeat in the <li> tag to iterate over the array of names.
4. To search the name in the array use the filter in the <li> tag.
5. using inptut tag get the input from the user using ng-model.
6. pass that variable to the filter in the li tag.

Code:

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

app.service('exampleService', function($http) {

this.getData = function() {

return $http.get('./user.json');

}

this.sendData = function(data) {

// send data to server

}

});

app.controller('exampleController', function($scope, exampleService) {

$scope.data = {};

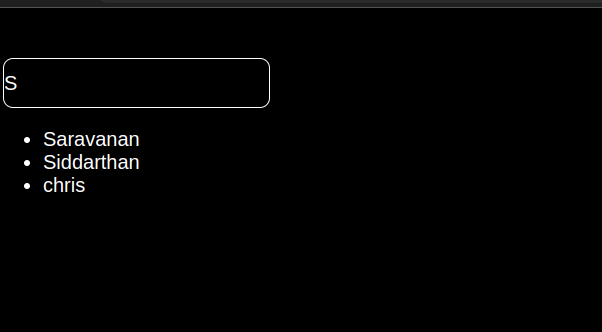
exampleService.getData().then(function(response) {

$scope.data = response.data;

});

});

Output:



Result:

The program to implement search to filter items using angular.js was executed successfully.

Ex. No. : 4.

Aim:

To write an Angular program for navigation menu

Procedure:

1. Create a Controller to define the menu options.
2. Use the ng-repeat directive to capture each menu item at a time and display them on the screen.
3. Update the URLs to display different details as menu items are clicked to distinguish between them.
4. Style the elements with internal CSS.

Source Code:

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

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

<script>

    // AngularJS code

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

    app.controller("menuController", function($scope) {

      $scope.menus = [

        {

          title: "Home",

          link: "#home"

        },

        {

          title: "About",

          link: "./item1.html"

        },

        {

          title: "Rules",

          link: "item2.html"

        },

        {

          title: "Contact",

          link: "#contact"

        },

{

          title: "Register",

          link: "./regist.html"

        },

{

          title: "Angular calculator",

          link: "./angcalc.html"

        },

{

          title: "Search User",

          link: "./service.html"

        },

{

          title: "Languages offered",

          link: "./layout.html"

        },

{

          title: "View poster",

          link: "./anim.html"

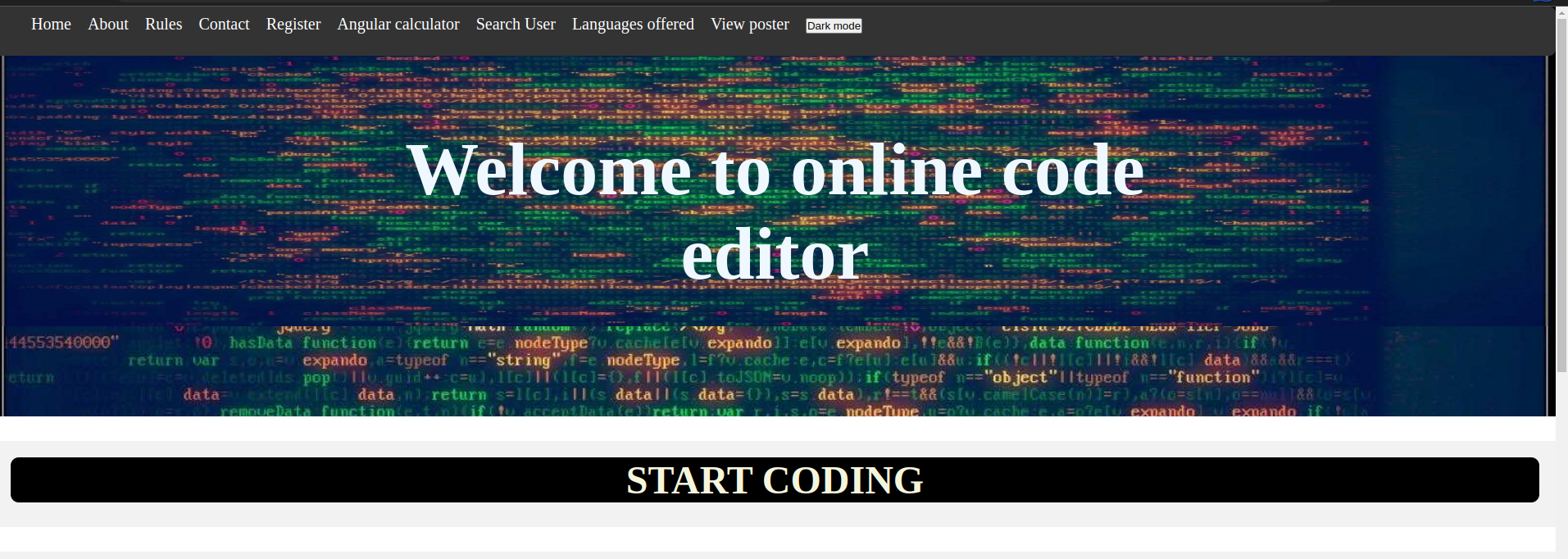
        }

      ];

    });

  </script>

Output:



Result:

The program was executed successfully.

Ex. No. : 5.a.

Aim:

Angular program to switch between layouts.

Procedure:

1. Create an Angular application and name it myApp.
2. Create the links to different layouts using the a tag.
3. Inside the controller, change the template URLs to display the link that was clicked to open the layout desired.
4. Add internal CSS to the file.

Source Code:

<!DOCTYPE html>

<html ng-app="myApp">

<head>

  <title>Switching Layouts in AngularJS</title>

<link rel="stylesheet" href="./layout.css">

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

  <script>

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

    app.controller('LayoutController', function() {

      this.layout = 'grid';

      this.toggleLayout = function() {

        this.layout = (this.layout === 'grid') ? 'list' : 'grid';

      };

    });

  </script>

</head>

<body ng-controller="LayoutController as layout">

  <button ng-click="layout.toggleLayout()">Toggle Layout</button>

  <div class="{{layout.layout}}">

    <div class="item">C++</div>

    <div class="item">Java</div>

    <div class="item">Python</div>

    <div class="item">C</div>

    <div class="item">Rust</div>

  </div>

</body>

</html>

CSS:  
body

{

background-color: #f0f0f0;

font-family: Arial, Helvetica, sans-serif;

font-size: 30px;

color: #333;

margin: 0;

padding: 0;

}

.grid .item {

text-align: center;

line-height: 200px;

display: inline-block;

width: 200px;

height: 200px;

background-color: #ccc;

margin: 10px;

border-radius: 10px;

}

.list .item {

display: block;

width: 30%;

height: 50px;

background-color: #ccc;

margin-bottom: 10px;

border-radius: 10px;

margin-left: 10px;

padding-left: 10px;

}

button{

/\* centre the button \*/

display: block;

margin: 0 auto;

background-color: #333;

color: #fff;

border: none;

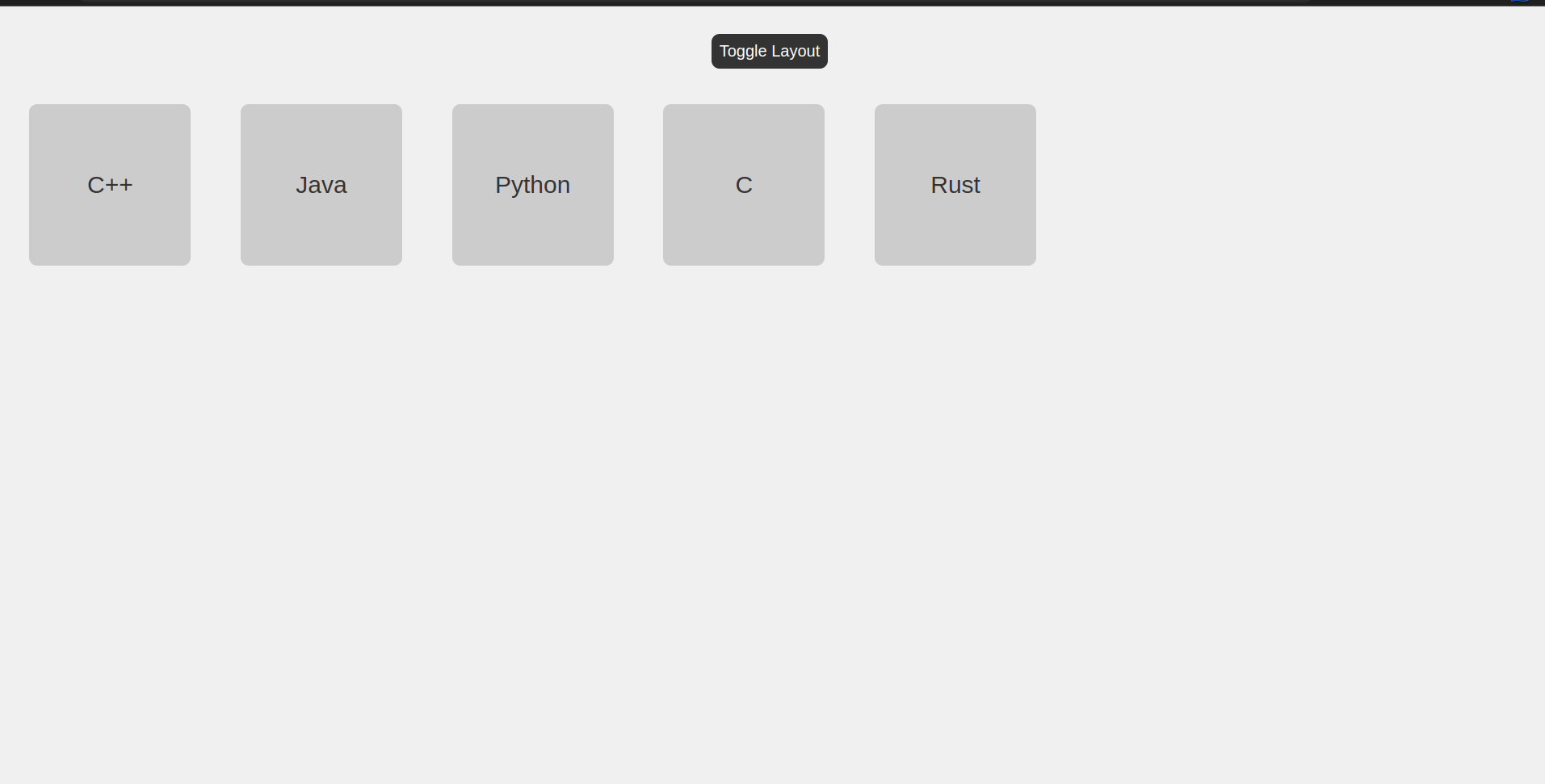
padding: 10px;

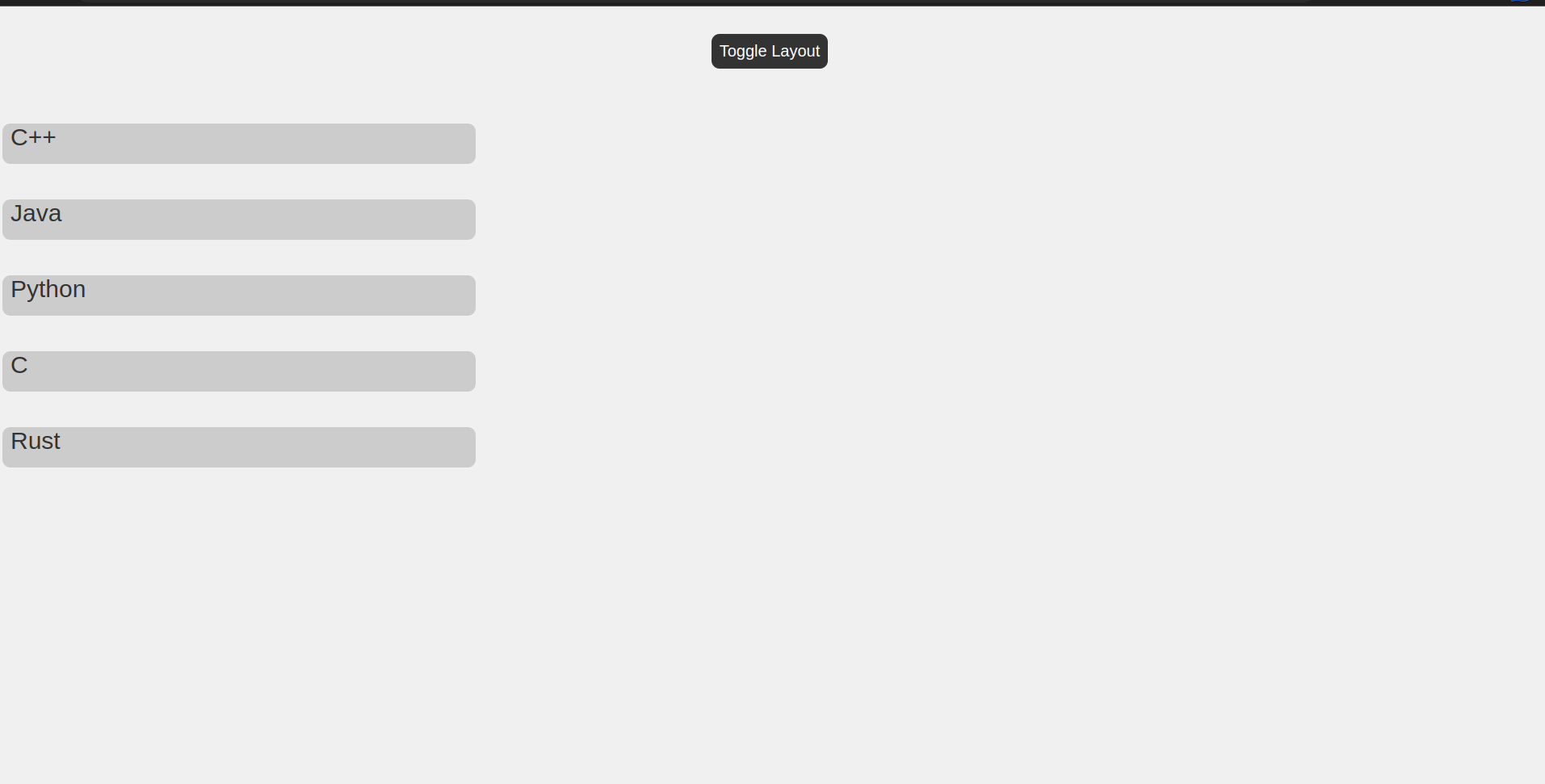
border-radius: 10px;

font-size: 20px;

}

Output:





Result:

The program was executed successfully.

Ex. No. : 5. b.

Aim:

To implement animation features in AngularJS.

Procedure:

1. Start html tag
2. Create an Angular module and controller.
3. Make use of checkboxes and use “ng-show”, “ng-hide” and “ng-click” directives to show and hide certain HTML elements in the view on clicking the checkboxes.
4. Add CSS animations to make the divs “fade in” or “roll down” to get displayed.

Source Code:

<!DOCTYPE html>

<html>

    <head>

        <title>Angular JS Animations</title>

        <style>

            div {

              transition: all linear 0.5s;

              background-color: rgb(199, 37, 37);

              height: 100px;

              width: 100%;

              position: relative;

              top: 0;

              left: 0;

              padding-top: 25px;

              font-size: 50px;

              text-align: center;

            }

            .ng-hide {

              height: 0;

            }

            .css-class-add, .css-class-remove {

            transition: all 0.5s cubic-bezier(0.250, 0.460, 0.450, 0.940);

            }

            .css-class,

            .css-class-add.css-class-add-active {

            color: rgb(216, 38, 76);

            font-size: 5em;

            }

            .css-class-remove.css-class-remove-active {

            font-size: 2em;

            color: black;

            }

        </style>

        <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-animate.js"></script>

    </head>

    <body ng-app="myApp">

        <h1>Hide the DIV: <input type="checkbox" ng-model="myCheck"></h1>

        <div ng-hide="myCheck">WELCOME TO THE RED CARPET</div>

        <h1>Show the DIV: <input type="checkbox" ng-model="myCheck2"></h1>

        <div ng-show="myCheck2">Hope you enjoy your stay here</div>

        <p>

            <button ng-click="myCssVar='css-class'">Set</button>

            <button ng-click="myCssVar=''">Clear</button>

            <br>

            <span ng-class="myCssVar">Stay Happy</span>

        </p>

        <script>

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

        </script>

    </body>

</html>

Output:



Result:

The program was executed successfully.

Ex. No. : 6

Aim:

To write an Angular Program to create form with real time updations.

Procedure:

1. In the HTML code, add a form with input fields. Give an action to the form to a file “connect.php” with a “POST” method, which indicates that data entered in the form is to be stored in the database.
2. Create a database “test1” and a table “formdeets” in the phpMyAdmin page.
3. Create a “connect.php” file, and include functions to connect to the database “test1”.
4. Assign the data in the forms to different variables and write an SQL query to store the data in the “formdeets” table in the database.
5. Add error handling routines to display messages in case database connection isn’t successful.

Source Code:

*test.php*

<html>

    <head>

        <style>

            .error {

  color: red;

}

        </style>

    </head>

    <body>

    <h1>Real Time Form Data Updation</h1>

    <form action='connect.php' method="POST">

    <label for="name">Name:</label>

    <input type='text' id="name" name='name' required /> <br> <br>

    <label for="email">Email:</label>

    <input type='email' id="email" name='email' required /> <br> <br>

    <input name='submit' type='submit' ID="Submit"/>

    </form>

    </body>

</html>

*connect.php*

<?php

// The argument inside $\_POST is the name of the submit button

    if($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($\_POST['submit'])) {

        // 1.name of the server, 2. type of user 3. password, 4.name of DB

        $conn= mysqli\_connect('localhost', 'root', '', 'test1') or die("Connection Failed:" .mysqli\_connect\_error());

        if(isset($\_POST['name']) && isset($\_POST['email']))

        {

            // Inside $\_POST, give the name of the input field in test.php

            $name= $\_POST['name'];

            $email= $\_POST['email'];

            // Enter details into DB

            $sql= "INSERT INTO `formdeets` (`name`, `email`) VALUES ('$name', '$email')";

            // Create a query to connect it

            $query = mysqli\_query($conn, $sql);

            if($query) {

                echo 'Entry Successful';

            }

            else {

                echo 'Error';

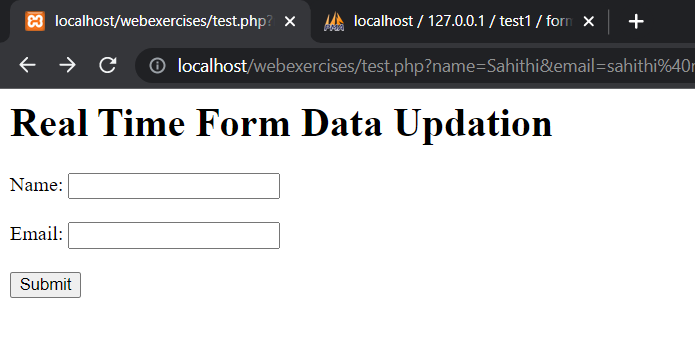
            }

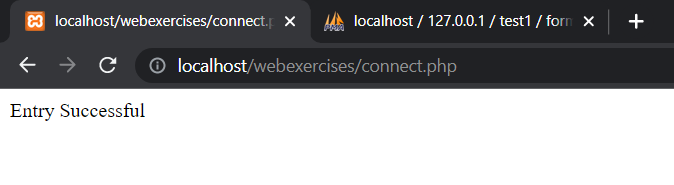
        }

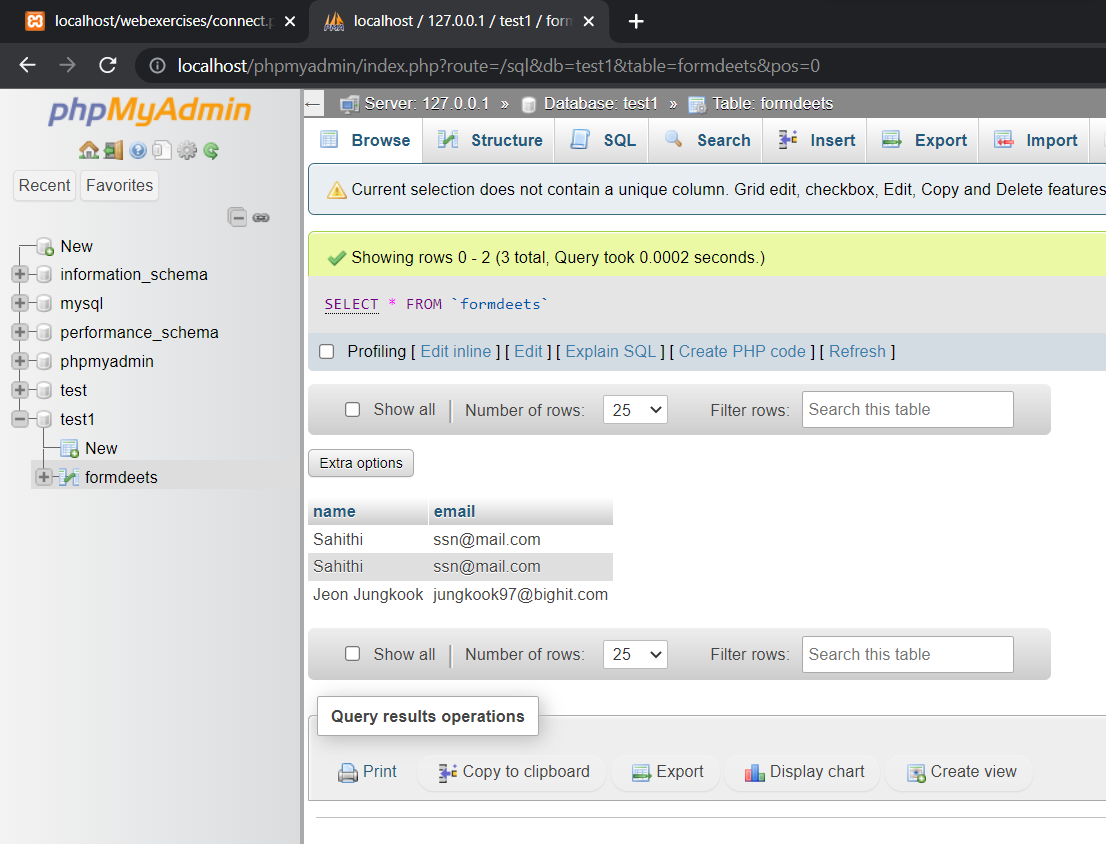
    }

?>

Output:







Result:

The program was executed successfully.

Ex. No. : 7

Aim:

To implement a video player using an AngularJS program

Procedure:

1. Start HTML tag
2. Create an AngularJS module and controller for the video player.
3. Define the video source by setting the "src" attribute of the video tag in the HTML. This can be a URL pointing to a video file or a media stream.
4. Use AngularJS directives such as "ng-src" to bind the "src" attribute of the video tag to a variable in the controller that holds the URL of the video file.
5. Use AngularJS directives such as "ng-show" or "ng-if" to control the visibility of various elements of the video player, such as the video controls or the "play" button, based on the current state of the player.
6. Use AngularJS event handlers such as "ng-click" to bind functions in the controller to various user actions such as clicking on the "play" button or seeking to a specific point in the video.
7. Use the built-in methods and properties of the video tag in the controller to manipulate the playback of the video, such as starting and stopping the video, adjusting the volume, or seeking to a specific point in the video.
8. Add error handling and error messages to handle cases where the video file or stream cannot be loaded or played properly.

Source Code:

<!DOCTYPE html>

<html ng-app="videoPlayer">

<head>

    <title>AngularJS Video Player</title>

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

</head>

<body ng-controller="videoPlayerCtrl">

    <h1>AngularJS Video Player</h1>

    <video ng-src="{{videoUrl}}" controls></video>

    <button ng-click="play()">Play</button>

    <button ng-click="pause()">Pause</button>

    <button ng-click="stop()">Stop</button>

    <script>

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

        app.controller("videoPlayerCtrl", function($scope) {

            $scope.videoUrl = "https://sample-videos.com/video123/mp4/240/big\_buck\_bunny\_240p\_1mb.mp4";

            var video = document.getElementsByTagName("video")[0];

            $scope.play = function() {

                video.play();

            };

            $scope.pause = function() {

                video.pause();

            };

            $scope.stop = function() {

                video.currentTime = 0;

                video.pause();

            };

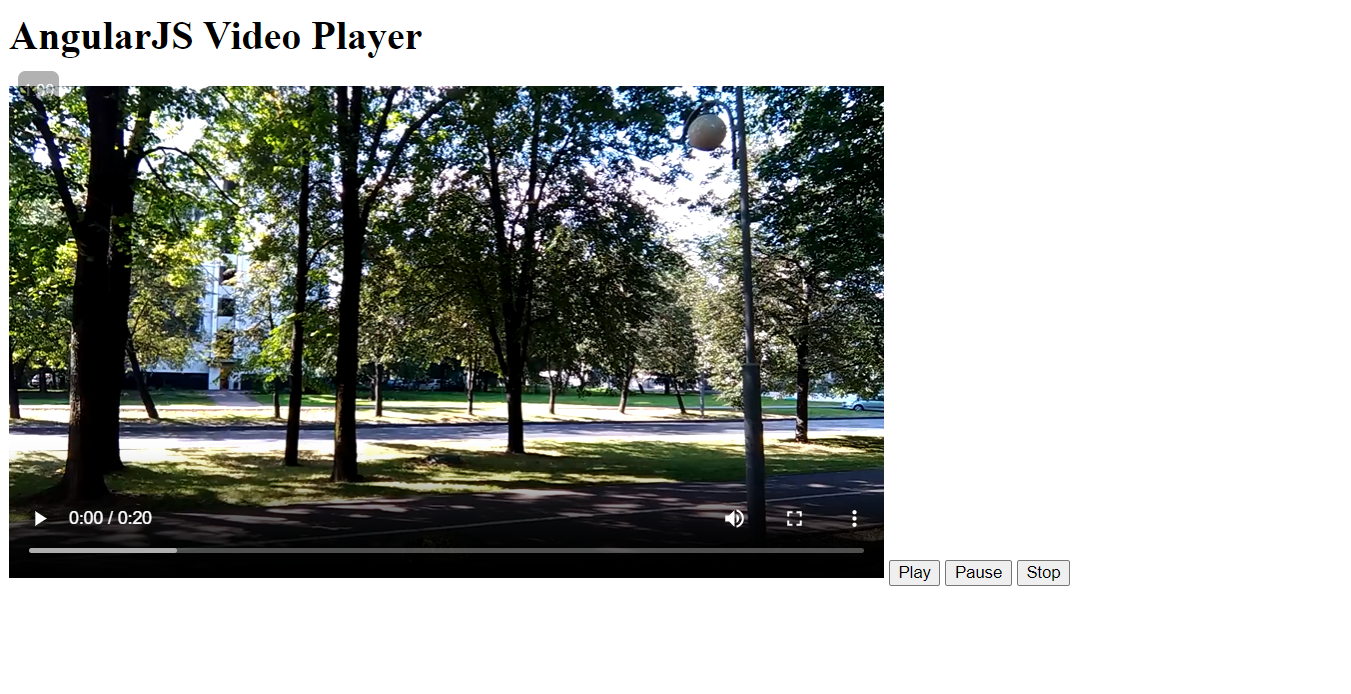
        });

    </script>

</body>

</html>

Output:



Result:

The program was executed successfully.

Ex. No. : 8

Aim:

To implement a text editor using AngularJS

Procedure:

1. Define the AngularJS app module and controller.
2. Create a textarea element in the HTML with ng-model directive to bind it to the $scope.text variable in the controller.
3. Create two buttons in the HTML with ng-click directive to bind them to their respective functions in the controller.
4. In the controller, initialize the $scope.text and $scope.savedText variables to empty strings.
5. Define a clear function that sets the $scope.text variable to an empty string.
6. Define a save function that sets the $scope.savedText variable to the value of the $scope.text variable.
7. Display the saved text below the buttons using the ng-if directive to check if the $scope.savedText is not an empty string.

Source Code:

<!DOCTYPE html>

<html ng-app="textEditor">

<head>

    <title>AngularJS Text Editor</title>

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

    <style>

        textarea {

            width: 100%;

            height: 200px;

        }

    </style>

</head>

<body ng-controller="textEditorCtrl">

    <h1>AngularJS Text Editor</h1>

    <textarea ng-model="text"></textarea>

    <button ng-click="clear()">Clear</button>

    <button ng-click="save()">Save</button>

    <div ng-if="savedText.length > 0">

        <h2>Saved Text:</h2>

        <p>{{ savedText }}</p>

    </div>

    <script>

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

        app.controller("textEditorCtrl", function($scope) {

            $scope.text = "";

            $scope.savedText = "";

            $scope.clear = function() {

                $scope.text = "";

            };

            $scope.save = function() {

                $scope.savedText = $scope.text;

            };

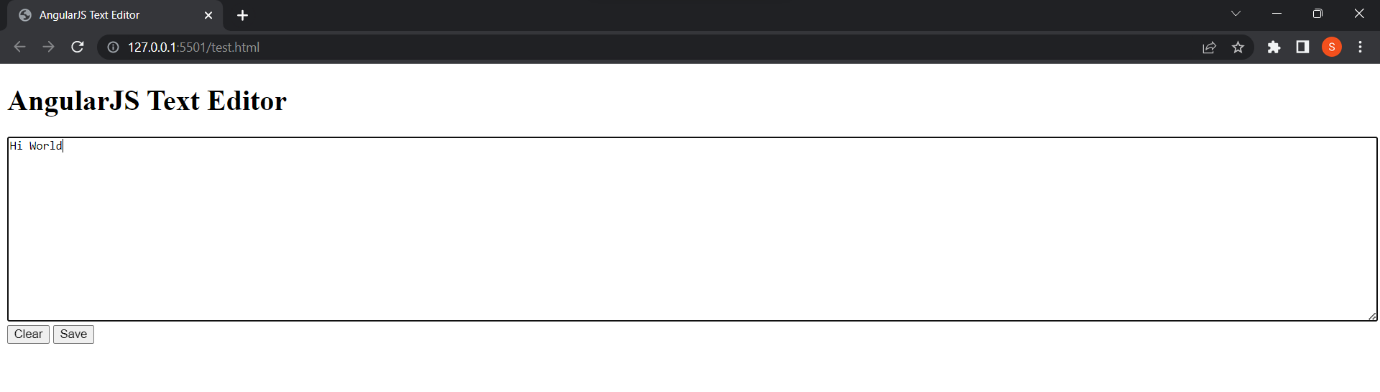
        });

    </script>

</body>

</html>

Output:



Result:

The Program was executed successfully.