**Exercise 1**

**1.a. Make use of HTML to implement the following:**

**Horizontal rule, Links, Image Insertion, Ordered and unordered list, Internal hyper linking, Meta, Simple table, Form, Frames, and Hotspot creation.**

**1.b Make use of CSS to implement the following**

**CSS 2: Types of CSS (inline, internal and external), Selectors, Box model, Layout, and Positioning.**

**Aim:**

To learn to implement HTML(HyperText Markup Language) tags and use different features available and to implement different types of CSS styles and methods to style.

**Algorithm:**

1. Start
2. Start html tag
3. Start head tag
4. Declare a link tag for the external CSS file
5. Start style tag for internal CSS
6. Set background color
7. End style tag
8. Declare title tag and set its value to "Box Model and CSS Example"
9. End head tag
10. Start body tag
11. Add inline css using style option in h1 tag
12. Add image using img tag with src=<link of image>
13. Add hr to draw horizontal rule
14. Add ol tag for ordered list within which items are listed through li tag
15. Add ul tag for unordered list within which items are listed through li tag
16. Add internal hyperlinking by defining id for an element in its tag and referencing through it
17. Create table using table tag within which tr tag is for rows and td tag is for columns for which box model CSS is defined
18. Create form using form tag
19. Use the input tag to create form inputs such as text, password, email, and submit
20. Create frames using iframe tag
21. Create hotspot using a and img tags
22. For representing layout in CSS add a div element
23. End body tag
24. End html tag
25. Stop

**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">

<link rel="stylesheet" href="1ab.css">

<style>

\*{

background-color: #FFEBEB;

}

</style>

<title>Exercise 1 a and b</title>

</head>

<body>

<h1 style="text-align: center;">HTML4 and CSS2</h1>

<h2>Image Insertion</h2>

<img src="https://m.media-amazon.com/images/I/81zXtlliBKL.jpg">

<h2>Horizontal rule</h2>

<hr>

<h2>Ordered list</h2>

<ol>

<li>Series</li>

<li>Art</li>

<li>Play</li>

</ol>

<h2>Unordered List</h2>

<ul>

<li>Class</li>

<li>Work</li>

<li>Tour</li>

</ul>

<h2>Internal Hyperlinking</h2>

<a href="#another-section" id="inter">Click me to go to the end of the document</a>

<h2>Table</h2>

<table>

<tr>

<th>Series</th>

<th>Genre</th>

</tr>

<tr>

<td>Bridgerton</td>

<td>Drama</td>

</tr>

<tr>

<td>You</td>

<td>Thriller</td>

</tr>

<tr>

<td>Echoes</td>

<td>Psychotic thriller</td>

</tr>

</table>

<h2>Form</h2>

<form>

<p>Enter your Name:<input name="name" type="text"></p>

<p>Enter your Age:<input name="age" type="number"></p>

<button name="Submit">Submit</button>

</form>

<h2>Frames</h2>

<iframe width="892" height="502" src="https://www.youtube.com/embed/zlJDTxahav0" title="Selena Gomez - Lose You To Love Me (Official Music Video)" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

<h2>Hotspot creation</h2>

<a href="https://youtu.be/bJqdZ20dcTM" target="\_blank"><img src="https://upload.wikimedia.org/wikipedia/en/c/c1/You\_%28TV\_series%29\_intertitle.png"></a>

<!--Layout-->

<div class="Footer">

<h2 id="another-section" style="text-align: center;">THE END</h2>

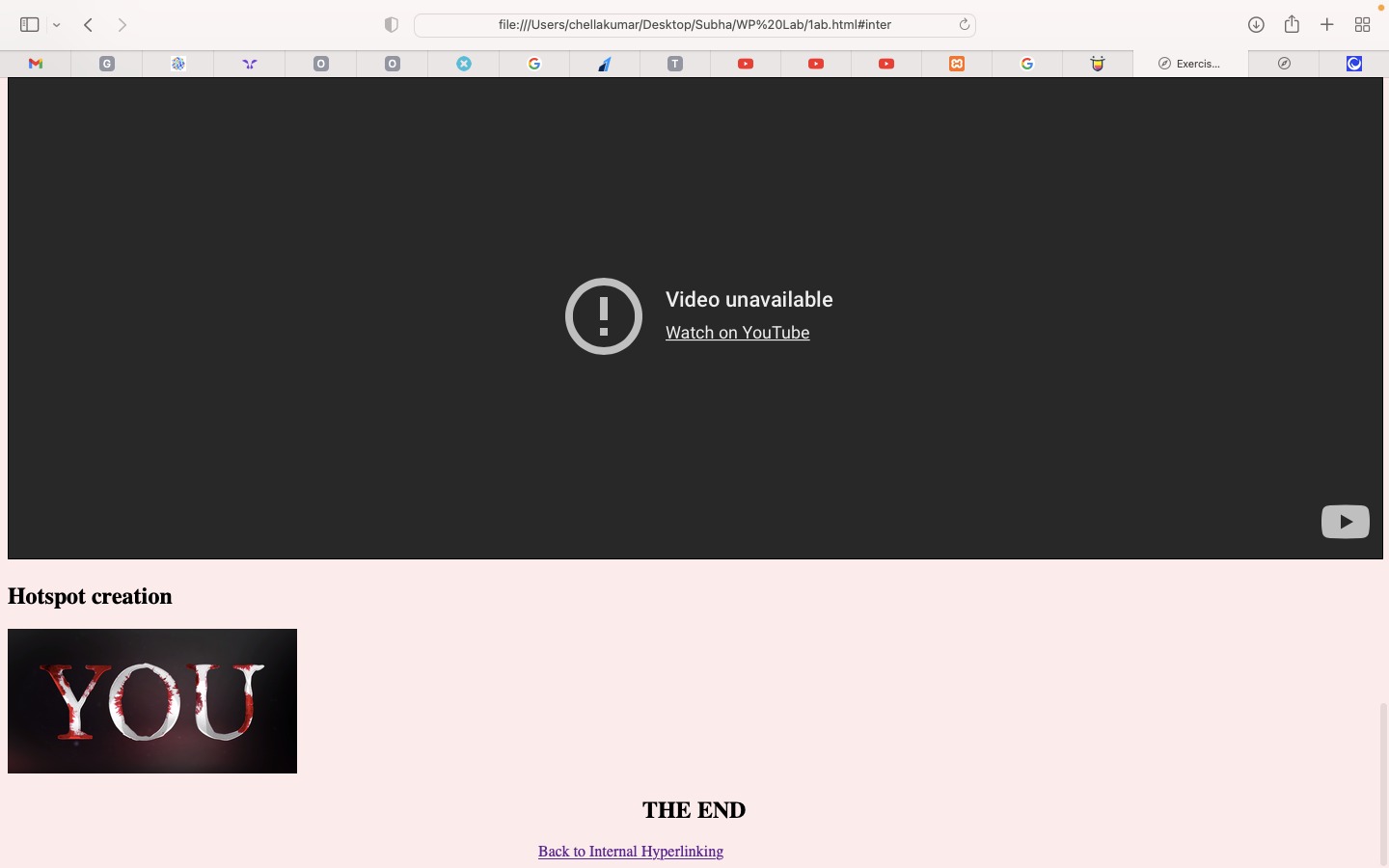
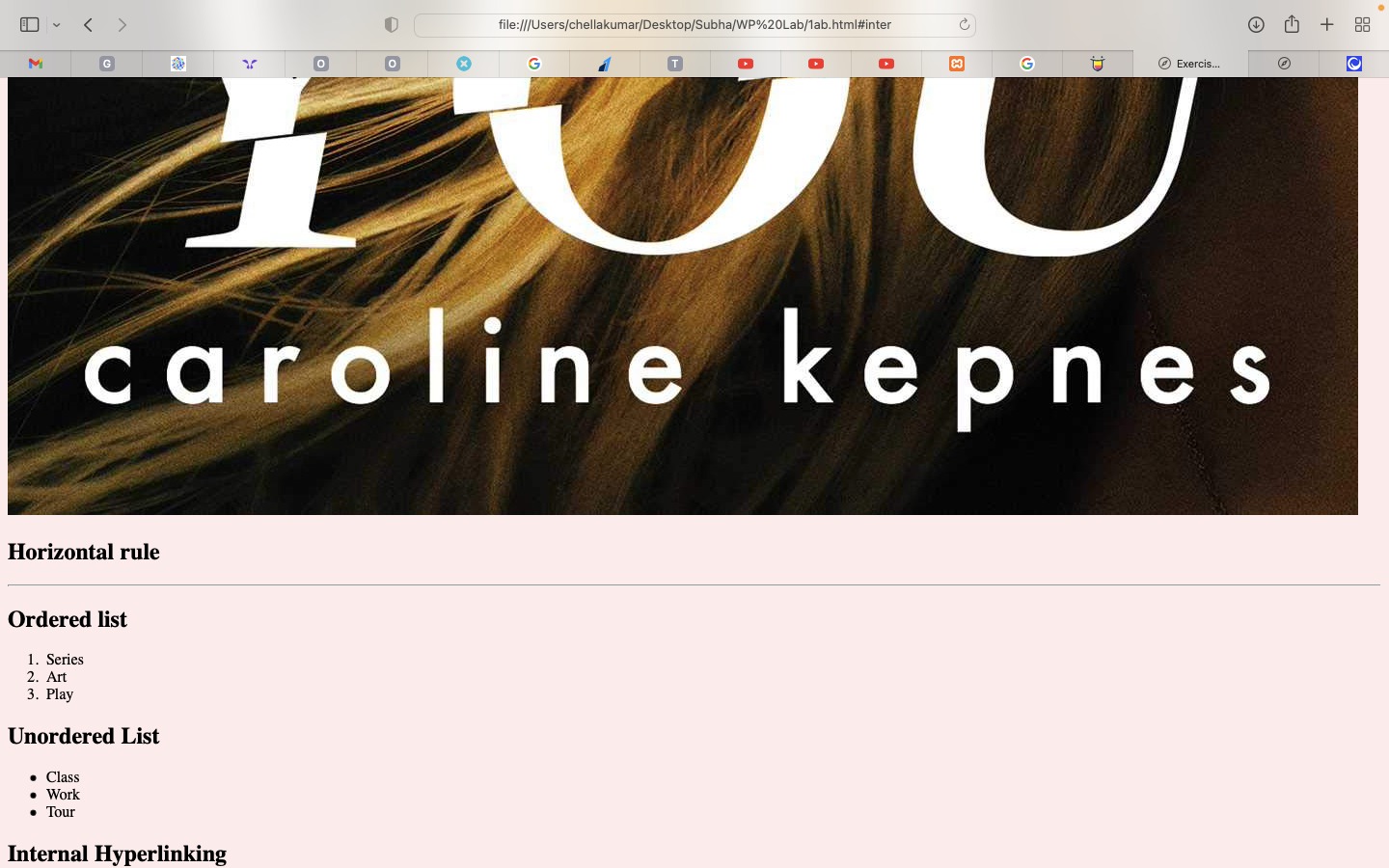
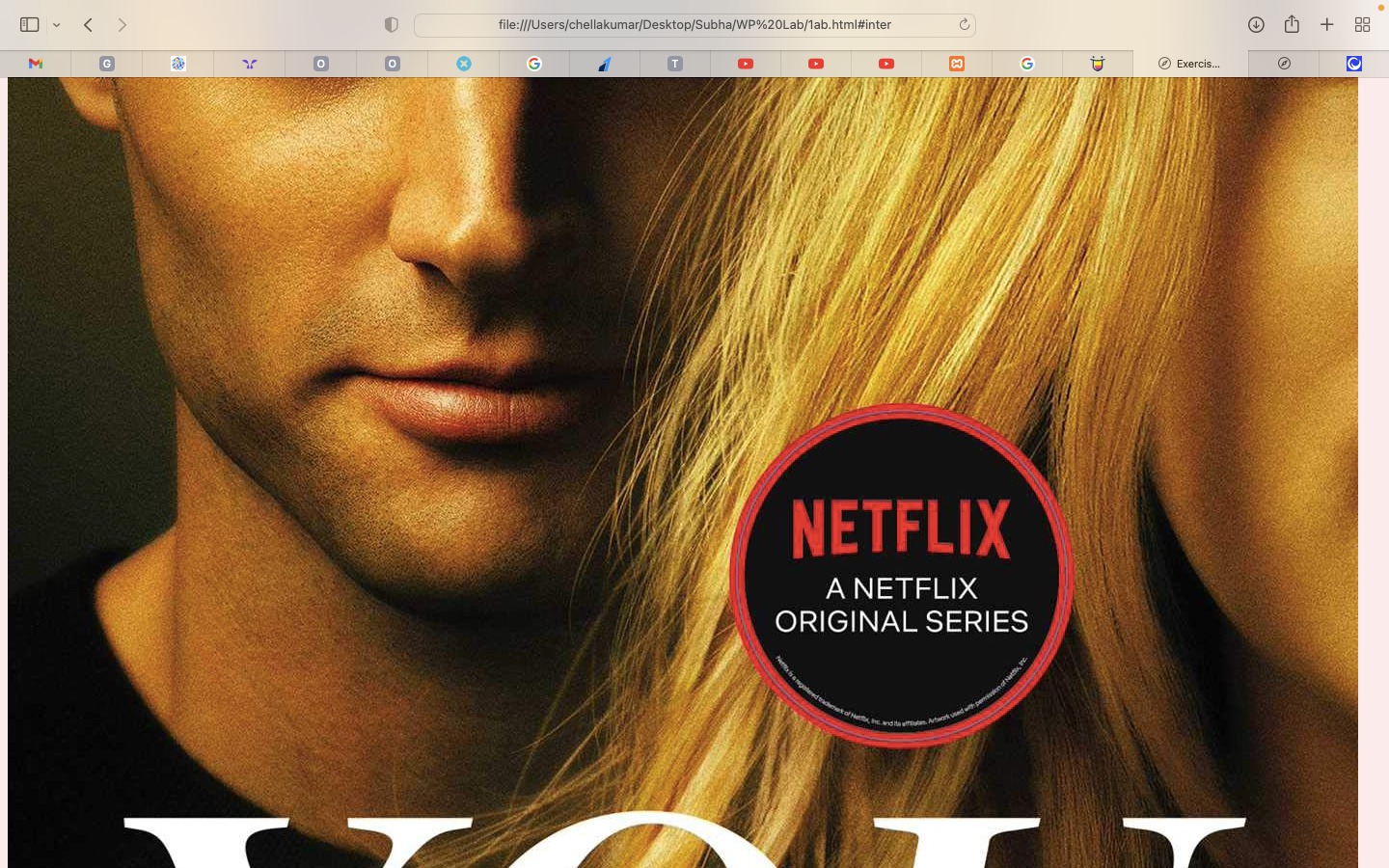
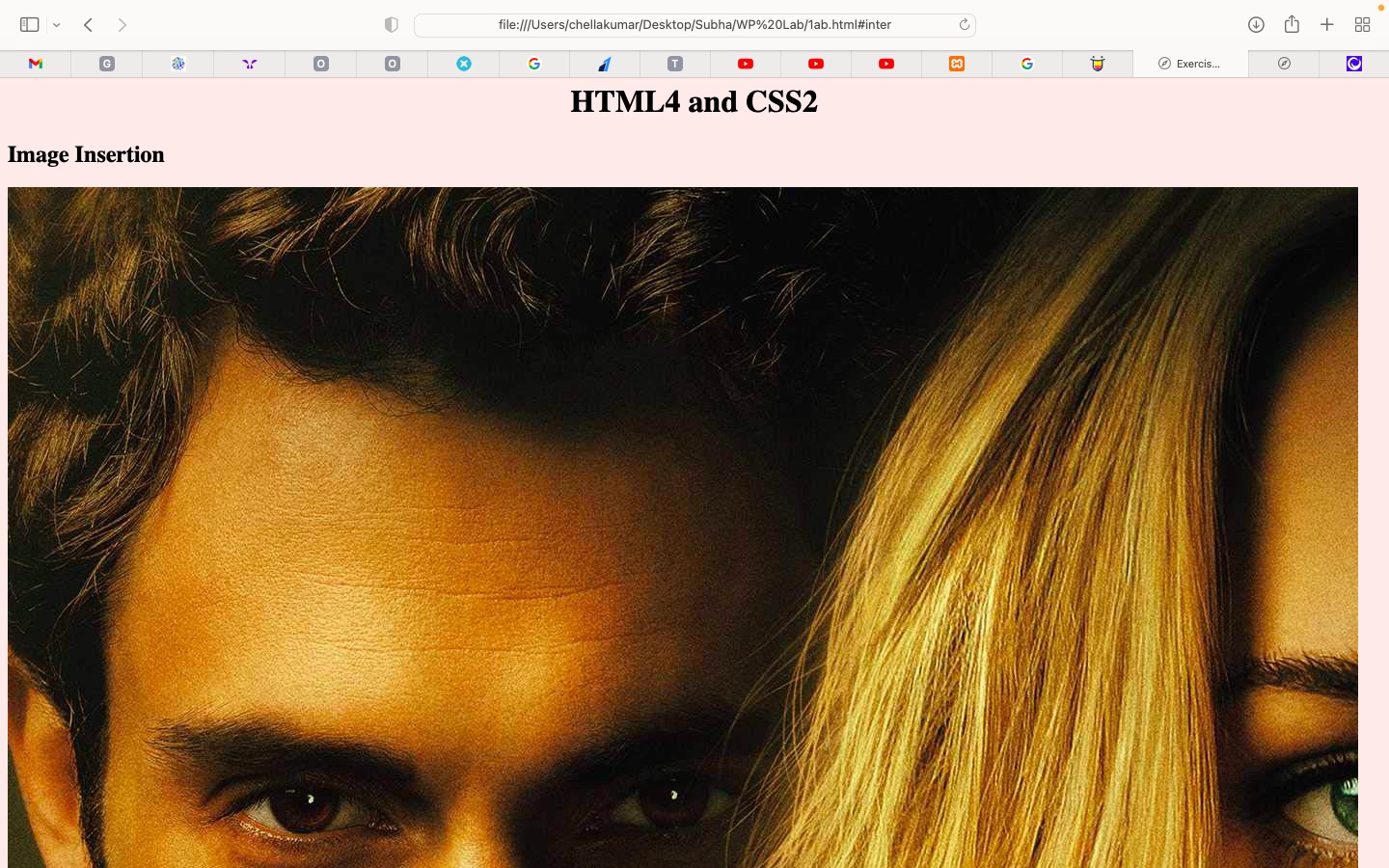
<a href="#inter" style="position: relative; left: 550px;">Back to Internal Hyperlinking</a>

</div>

</body>

</html>

**Output:**



**Result:**

**1.c Make use of HTML5 to implement the following:**

**New Structural Elements, Audio, Video, Form creation, Canvas API, SVG, and Geo location**

**1.d Make use of CSS 3 to implement the following**

**Borders, Text effects, Animation, and Transitions**

**Aim:**

To use HTML5 to implement HTML5 features and to use CSS3 to implement CSS3 features.

**Algorithm:**

1. Start
2. Declare doctype as HTML5
3. Start HTML tag
4. Start body tag on which animations are defined in CSS
5. Use the header tag to define header
6. Use the div tag to show transition
7. Use the audio tag to embed an audio file and set its source and controls
8. Use the video tag to embed a video file and set its source and controls
9. Use the form tag to create a form and set its action and method
10. Use the input tag to create form inputs such as text, password, email, and submit
11. Use the canvas tag to draw graphics using JavaScript
12. Use the svg tag to draw scalable vector graphics
13. To get GeoLocation a function in JavaScript was defined and position.coords.latitude and position.coords.longitude and navigator.geolocation.getCurrentPosition() were used.
14. Use the p tag to define some text on which text effects are applied
15. Use the script tag to include JavaScript code
16. End body tag
17. End html tag

**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">

<style>

#mycanvas{border:1px solid rgb(46, 56, 134);}

#svgelem {

position: relative;

left: 50%;

-webkit-transform: translateX(-20%);

-ms-transform: translateX(-20%);

transform: translateX(-20%);

}

div{

width: 100px;

height: 100px;

background: rgb(39, 74, 212);

}

div:hover {

width: 300px;

height: 300px;

}

h1{

margin-left:600px;

}

p.test1 {

white-space: nowrap;

width: 200px;

border: 1px solid #000000;

overflow: hidden;

text-overflow: ellipsis;

}

body {

width: 100px;

height: 100px;

background-color: rgb(0, 217, 255);

animation-name: example;

animation-duration: 10s;

}

@keyframes example {

from {background-color: rgb(207, 131, 80);}

to {background-color: rgb(15, 53, 123);}

}

</style>

<title>Exercise 1c and d</title>

</head>

<!--Animation-->

<body>

<!--New structural element-->

<header><h1>HTML5</h1></header>

<h2>Transition</h2>

<div></div>

<h2>Audio</h2>

<audio controls>

<source src="piano.mp3" type="audio/mp3">

Your browser does not support the audio tag.

</audio>

<h2>Video</h2>

<video autoplay muted>

<source src="https://cdn.videvo.net/videvo\_files/video/premium/video0261/large\_watermarked/500\_00299\_preview.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

<h2>Form</h2>

<form action = "" method = "get">

Enter email : <input type = "text" name = "newinput" required/>

<p>Try to submit using Submit button</p>

<input type = "submit" value = "submit me" />

</form>

<h2>Canvas</h2>

<canvas id = "mycanvas" width = "100" height = "100"></canvas>

<h2>SVG</h2>

<svg id = "svgelem" height = "200">

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

</svg>

<h2>GeoLocation</h2>

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

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

<p id="demo"></p>

<h2>Text Effect</h2>

<p class="test1">This is some long text that will not fit in the box</p>

<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;

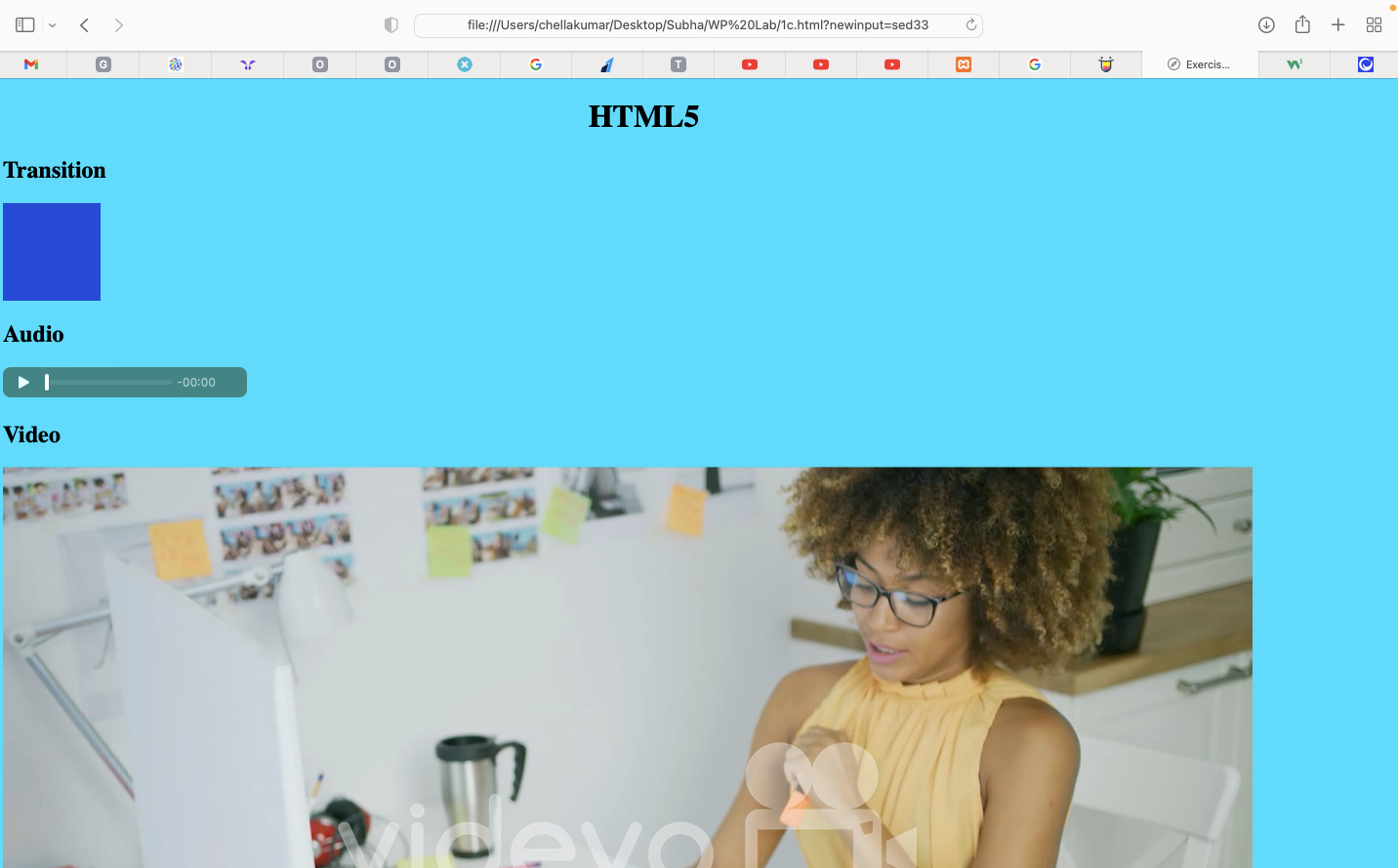
}

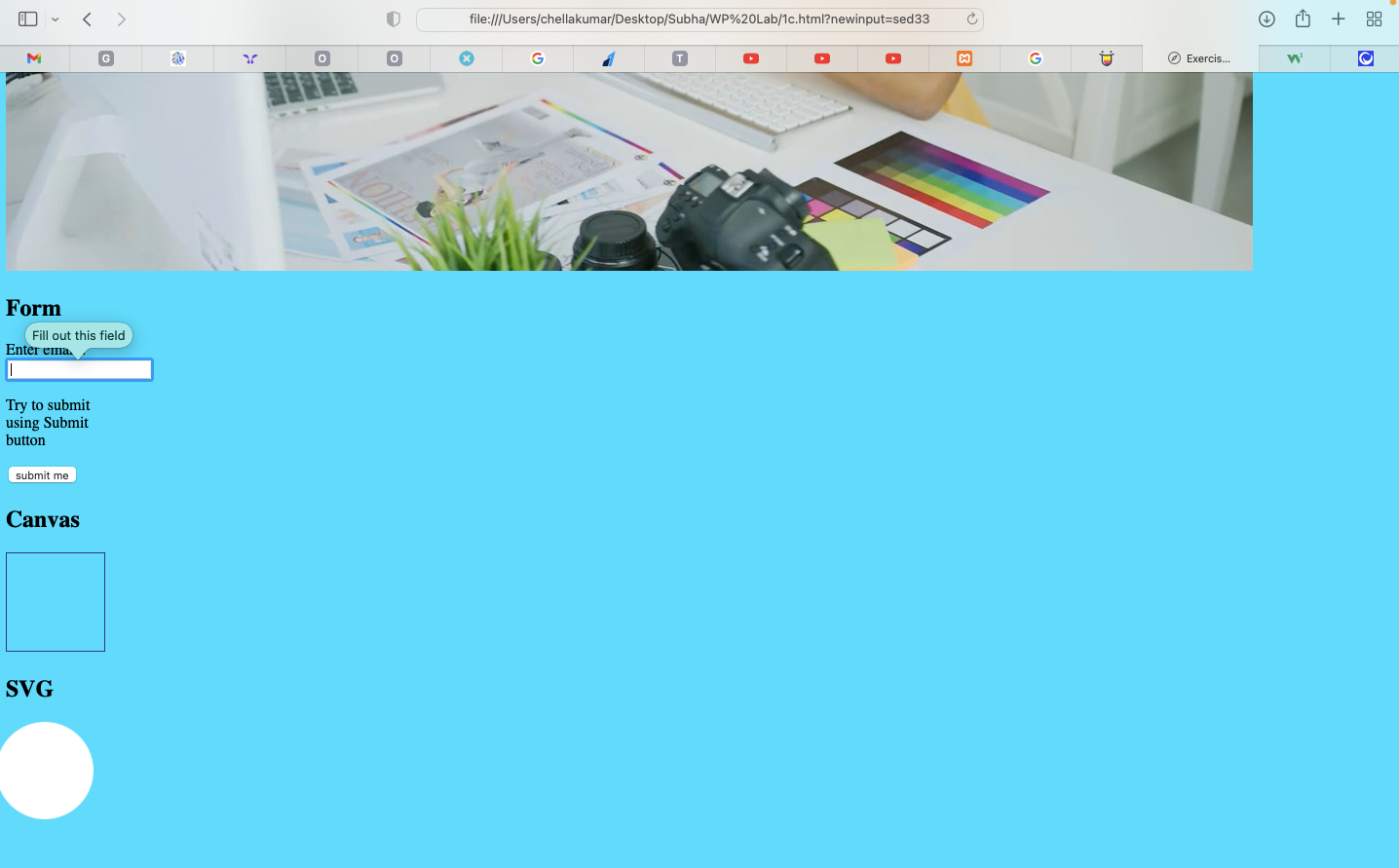
</script>

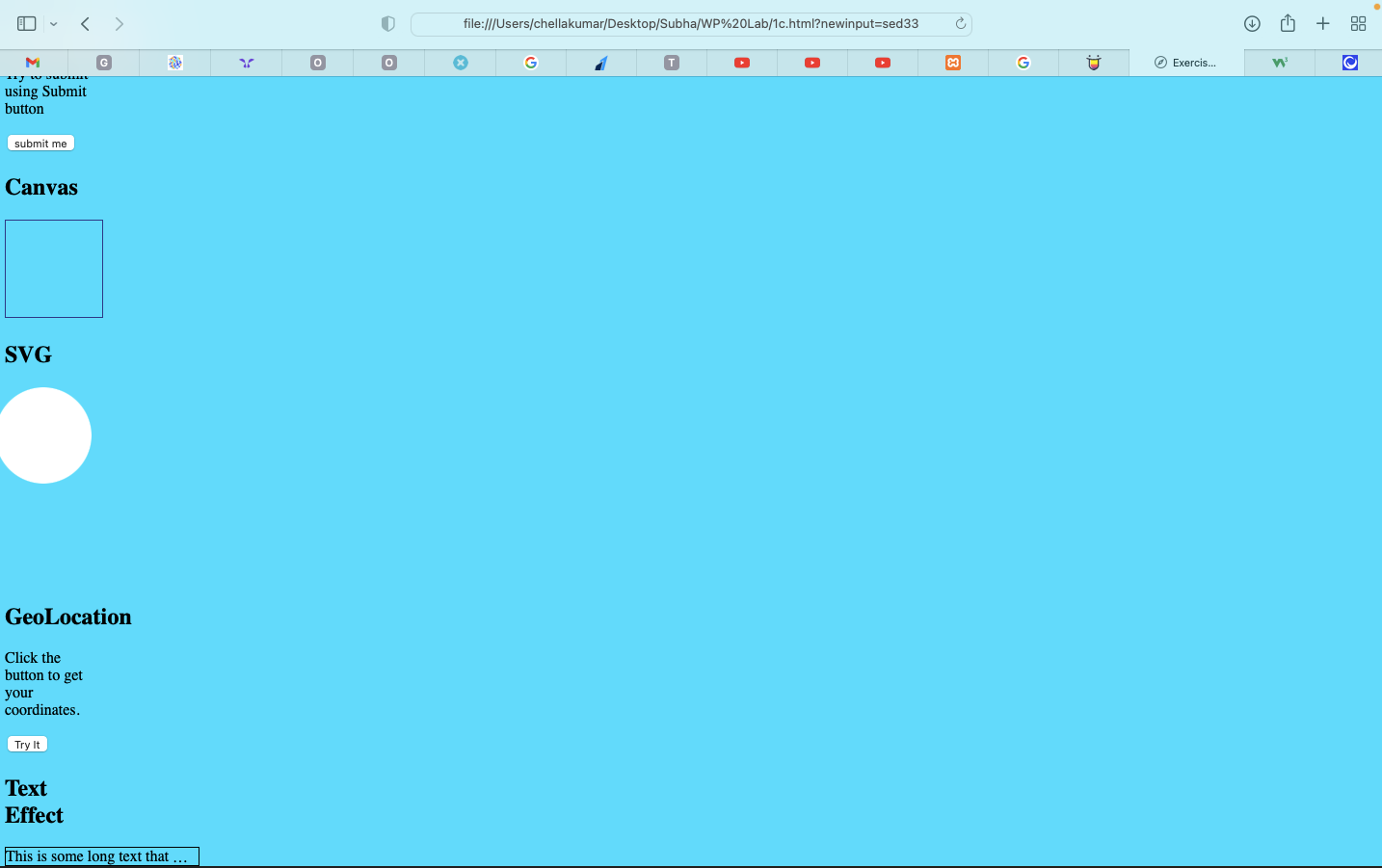
</body>

</html>

**Output:**







**Result:**

**Exercise 2**

**2.a Form validation using Regular Expression**

**Aim:**

To validate forms input using Regular Expressions in JavaScript.

**Algorithm:**

1. Start
2. Start HTML tag
3. Start body tag
4. Create a form with name, Email ID, password, gender etc as inputs using form and input tag.
5. Include the JavaScript required as an external JS file.
6. In the JavaScript file, use query selectors to select the relevant input fields. These will help us validate those inputs with regexes.
7. Define accepted regular expressions for each type of input.
8. Add an event listener on the submit button to check if the typed input pattern is similar to that of the regexes and code alerts to warn otherwise.
9. Stop

**Code:**

*ex2a.html*

<!DOCTYPE html>

<html>

<head>

<title>Form Validation</title>

<style>

legend {

display: block;

padding-left: 2px;

padding-right: 2px;

border: none;

}

</style>

<script type="text/javascript">

function validate() {

var user = document.getElementById("e").value;

var user2 = document.getElementById("e");

var re = /^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

if (re.test(user)) {

alert("done");

return true;

}

else {

user2.style.border = "red solid 3px";

return false;

}

}

</script>

</head>

<body>

<center>

<br/>

<h1>Form Validation using Regular Expression</h1>

<br/>

<form>

<fieldset style="width:500px">

<legend>Registation Form</legend>

<table>

<tr>

Name: <input type="text" placeholder="name" maxlength="10">

</tr>

<br><br>

<tr>

Email: <input type="email"

placeholder="username@gmail.com" id="e">

</tr>

<br><br>

<tr>

Password: <input type="password" placeholder="password">

</tr>

<br><br>

<tr>

Re-enter Password: <input type="password" placeholder="confirm password">

</tr>

<br><br>

<tr>

<label>Gender: </label>

<select id="gender">

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

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

<option value="others">Others</option>

</select>

</tr>

<br><br>

<tr><input type="submit"

onclick="validate()" value="Create">

</tr>

</table>

</fieldset>

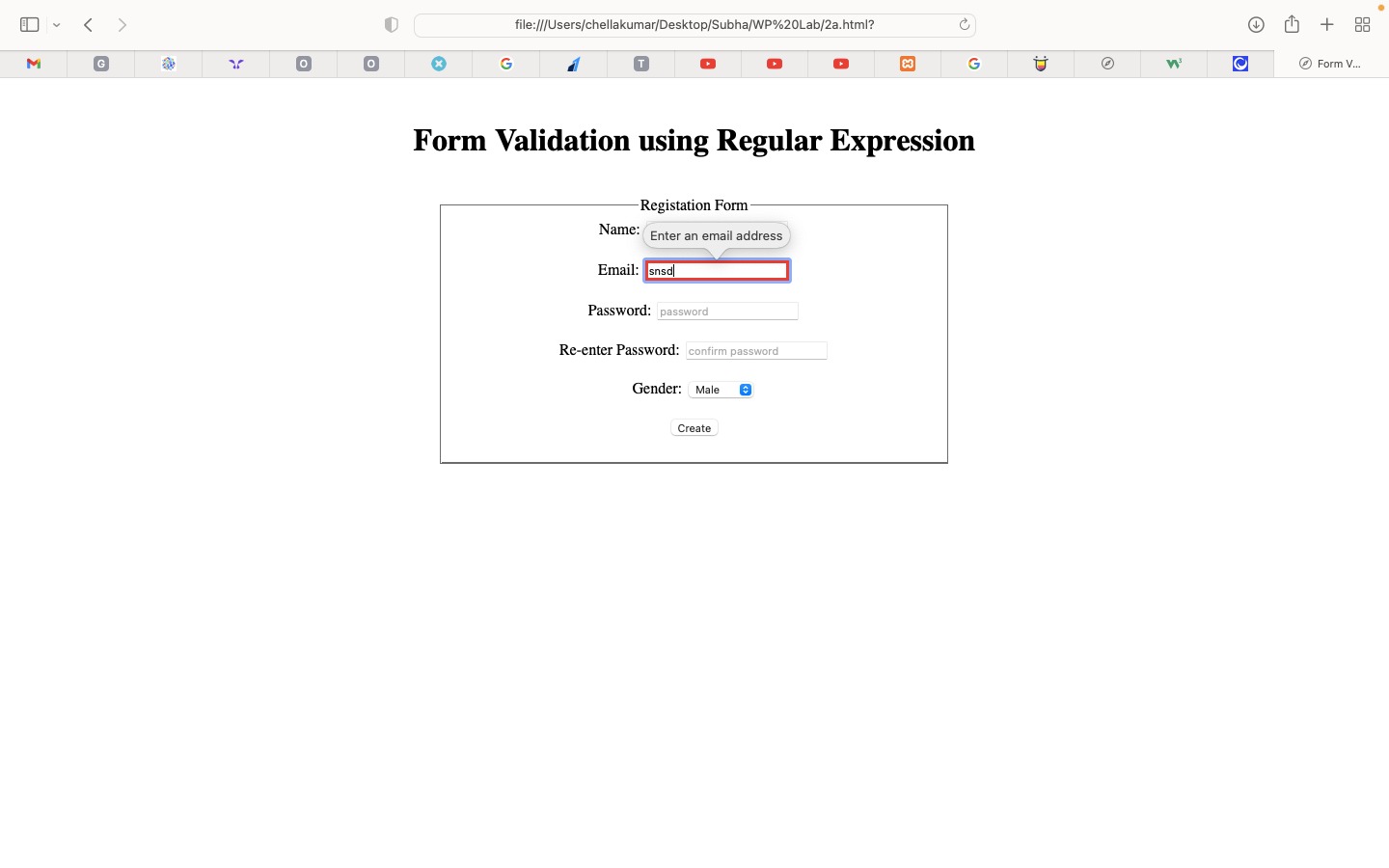
</form>

</center>

</body>

</html>

**Output:**



**Result:**

**2.b HTML DOM Manipulation**

**Aim:**

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

**Algorithm:**

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

**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>DOM Manipulation</title>

</head>

<body>

<p class="master2">I am sad and I eat</p>

<p class="master2">I am fat and I don't eat</p>

<p class="master2">I am sad coz I am fat</p>

<h1 class="master2">I am fat coz I am sad</h1>

<button id="btn">click me</button>

<script>

const btn = document.getElementById('btn')

btn.addEventListener('click', function master(){

var master = document.getElementsByClassName("master2");

master[2].innerHTML = 'I am sad and fat';

master[3].innerHTML = 'I am fat and sad';

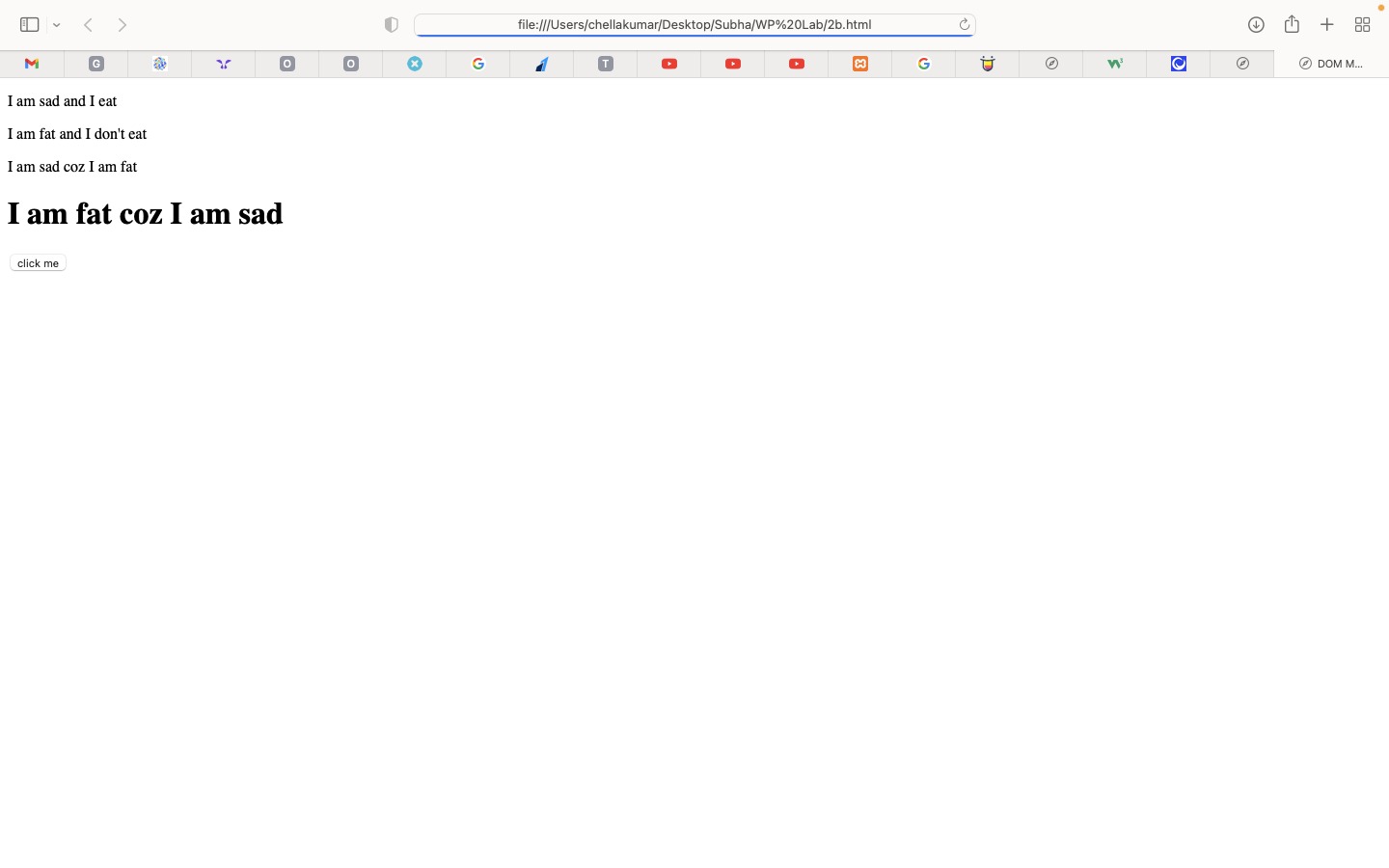
})

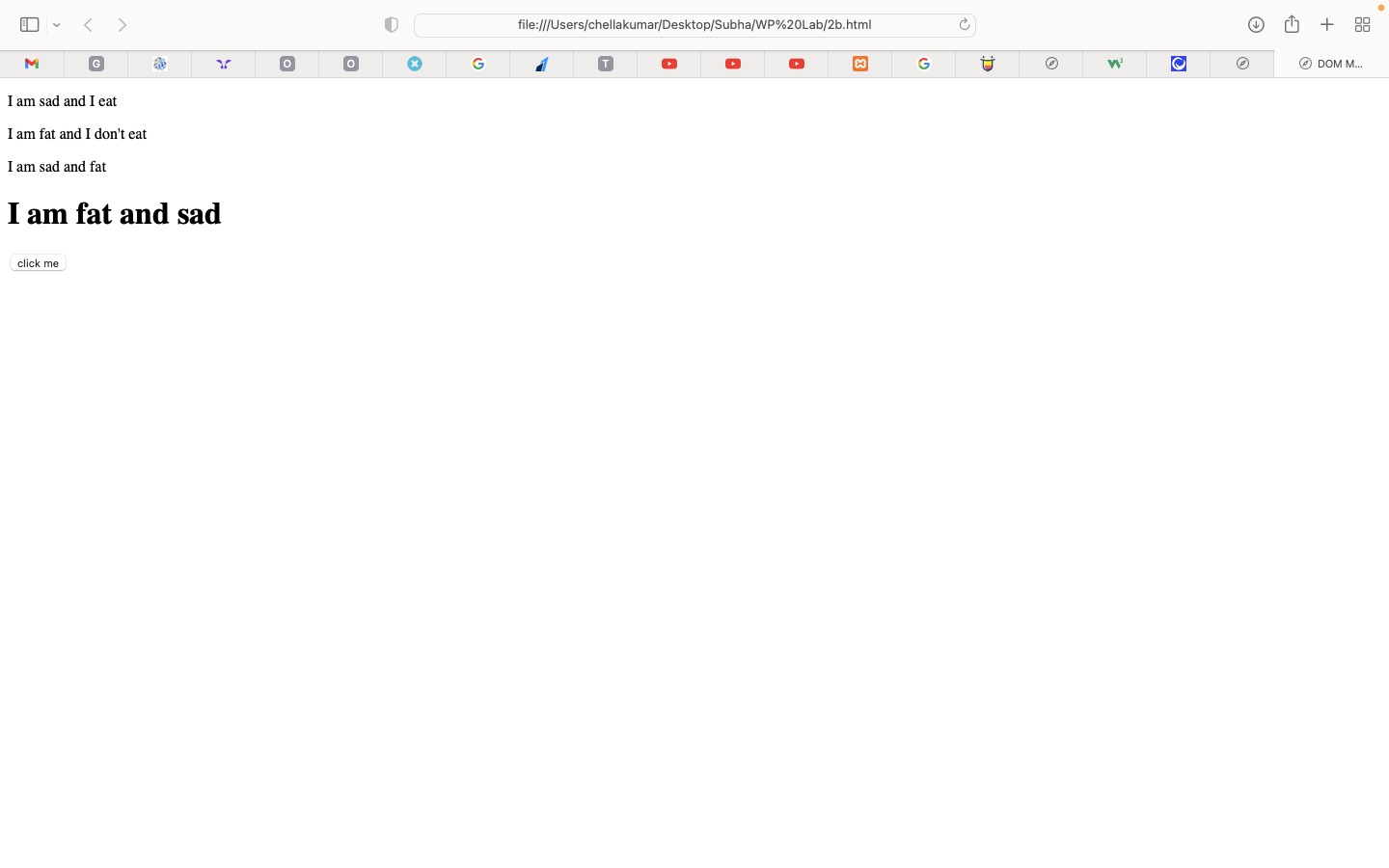
</script>

</body>

</html>

**Output:**





**Result:**

**2.c Simple Messaging Service**

**Aim:**

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

**Algorithm:**

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

**Code:**

*ex2c.html*

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Simple Messaging Service</title>

<style>

/\* CSS styles \*/

body {

font-family: Arial, sans-serif;

}

#chat {

height: 300px;

overflow-y: scroll;

border: 1px solid #ccc;

padding: 10px;

}

#message {

width: 100%;

box-sizing: border-box;

padding: 10px;

border: 1px solid #ccc;

}

#send-button {

background-color: #9b55c9;

color: white;

padding: 10px;

border: none;

cursor: pointer;

}

#send-button:hover {

background-color: #a663d2;

}

</style>

</head>

<body>

<h1>Simple Messaging Service</h1>

<div id="chat"></div>

<div>

<input type="text" id="message" placeholder="Type your message...">

<button id="send-button">Send</button>

</div>

<script>

var chat = document.getElementById("chat");

var messageInput = document.getElementById("message");

var sendButton = document.getElementById("send-button");

sendButton.addEventListener("click", function() {

var message = document.createElement("div");

message.innerText = messageInput.value;

chat.appendChild(message);

messageInput.value = "";

chat.scrollTop = chat.scrollHeight;

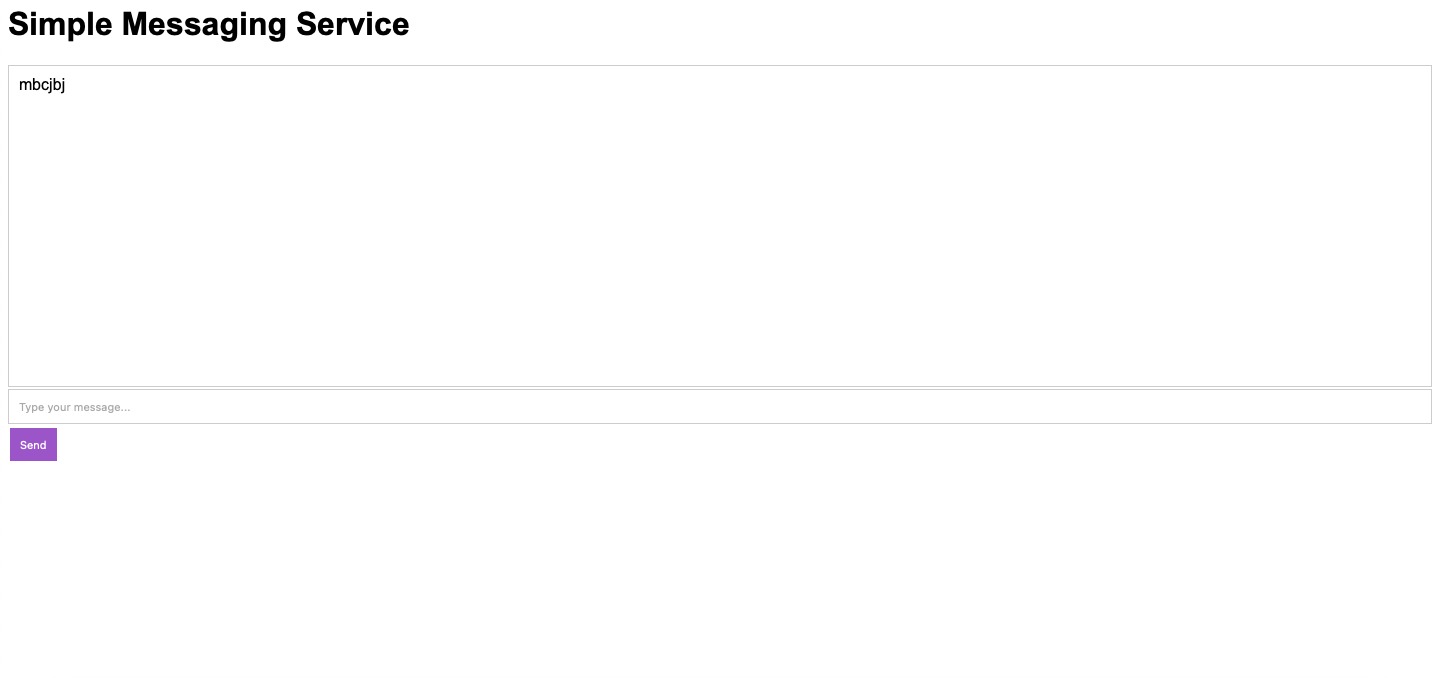
});

</script>

</body>

</html>

**Output:**



**Result:**

**Exercise 3**

**3.a Simple Calculator using AngularJS**

**Aim:**

To build a simple calculator using AngularJS.

**Algorithm:**

1. Start
2. Start HTML tag
3. Start head tag
4. Create an AngularJS app and a controller for the calculator application.
5. Use input tags to get operator and operand input from the user.
6. Apply ng-model directive to get the value of the inputs given and calculate the output in the controller in a JavaScript function.
7. Include the JavaScript function in an external JavaScript file.
8. Stop.

**Code:**

*ex3a.html*

<!DOCTYPE HTML>

<html>

    <head>

        <title>Calculator</title>

        <script data-require="angular.js@1.0.7" data-semver="1.0.7" src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.js"></script>

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

    </head>

    <body>

        <h1>Calculator using Angular JS</h1>

        <div ng-app="CalculatorApp" ng-controller="CalculatorController">

        <p style="font-weight: bold;"> Operand 1: <input type="number" ng-model="a"></p>

        <p style="font-weight: bold;"> Operand 2: <input type="number" ng-model="b"></p>

        <p style="font-weight: bold;"> Operator: <select ng-model="operator">

                <option>+</option>

                <option>\*</option>

                <option>-</option>

                <option>/</option>

                <option>%</option>

            </select></p>

        <p style="font-weight: bold;">The result is {{result()}}</p>

        </div>

    </body>

</html>

*ex3a.js*

angular.module('CalculatorApp', [])

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

        $scope.result = function() {

            if ($scope.operator == '+')

            {

                return $scope.a + $scope.b;

            }

            if ($scope.operator == '-')

            {

                return $scope.a - $scope.b;

            }

            if ($scope.operator == '\*')

            {

                return $scope.a \* $scope.b;

            }

            if ($scope.operator == '/')

            {

                return $scope.a / $scope.b;

            }

            if ($scope.operator == '%')

            {

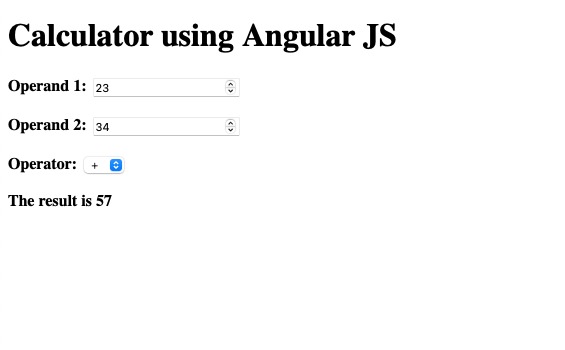
                return $scope.a % $scope.b;

            }

        };

    });

**Output:**



**Result:**

**3.b Angular program to implement Components, Service, and Controllers**

**Aim:**

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

**Algorithm:**

1. Start.
2. Write a HTML code declaring an Angular application.
3. Write a JavaScript file to create a Component that displays data from the JavaScript file to the HTML view.
4. Declare a user-defined tag in the HTML file to indicate where this data should go in the view.
5. Create an Angular service that displays data after a certain timeout interval.
6. Include the JS file as an external JavaScript file.
7. Stop.

**Code:**

*ex3b\_com.html*

<!DOCTYPE html>

<html>

<head>

<title>Component</title>

<script data-require="angular.js@1.5.8" data-semver="1.5.8" src="https://code.angularjs.org/1.5.8/angular.js"></script>

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

</head>

<body>

<div ng-app="app">

<hello-world name="Subhiksha"> </hello-world>

</div>

</body>

</html>

*test.js*

angular.module('app', [])

      .component('helloWorld', {

        template: '<h1 style="text-align:center">Hello World! This is {{$ctrl.name}}, and I am from {{$ctrl.class}}</h1>',

        bindings: { name : '@' },

        controller: function()

        {

            this.class = "IT-B";

        }

      });

**Output:**



**Code:**

*ex3b.html*

<!DOCTYPE HTML>

<html>

<head>

<title>Services</title>

<script data-require="angular.js@1.0.7" data-semver="1.0.7" src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.js"></script>

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

</head>

<body>

<h1>Services:</h1>

<div ng-app="Services" ng-controller="ServicesController" style="text-align:center">

<h3>URL of this website is:</h3>

<p>{{loc}}</p>

<h3>{{head}}</h3>

</div>

</body>

</html>

*ex3b.js*

angular.module('Services', [])

    .controller('ServicesController', function($scope, $location, $timeout) {

        $scope.loc=$location.absUrl();

        $scope.head="I'd rather be lost in the lights, lost in the lights, I'm out of my mind";

        $timeout(function(){

            $scope.head="Can you help me numb the pain? Each night, you spin me up high, emotions on ice~";

        },5000)

    });

**Output:**





**Result:**

**3.c Angular program to implement search to filter items.**

**Aim:**

To write an AngularJS program to implement a search feature to filter items.

**Algorithm:**

1. Start.
2. Write a HTML code declaring an Angular application.
3. Create a form with name, subject, fees inputs.
4. Create a search box and add an Angular Filter to it to display only those subjects that match the search box.
5. Write the Angular code in an external JavaScript file and include the file in the HTML document.
6. Stop.

**Code:**

<html>

<head>

<title>Angular JS Filters</title>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>

<body>

<h2>AngularJS Sample Application</h2>

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

<table border = "0">

<tr>

<td>Enter first name:</td>

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

</tr>

<tr>

<td>Enter last name: </td>

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

</tr>

<tr>

<td>Enter fees: </td>

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

</tr>

<tr>

<td>Enter subject: </td>

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

</tr>

</table>

<br/>

<table border = "0">

<tr>

<td>Name in Upper Case: </td><td>{{student.fullName() | uppercase}}</td>

</tr>

<tr>

<td>Name in Lower Case: </td><td>{{student.fullName() | lowercase}}</td>

</tr>

<tr>

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

</tr>

<tr>

<td>Subject:</td>

<td>

<ul>

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

{{ subject.name + ', marks:' + subject.marks }}</li>

</ul>

</td>

</tr>

</table>

</div>

<script>

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

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

$scope.student = {

firstName: "Subhiksha",

lastName: "Sakthivel",

fees:500,

subjects:[

{name:'English',marks:99},

{name:'Physics',marks:95},

{name:'Chemistry',marks:95},

{name:'Math',marks:95},

{name:'Computer Science',marks:97},

],

fullName: function() {

var studentObject;

studentObject = $scope.student;

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

}

};

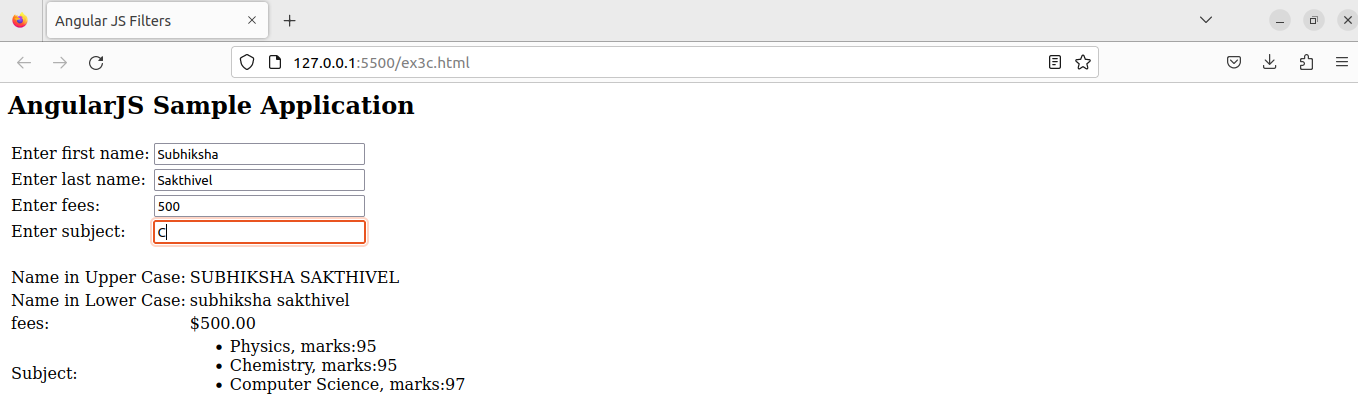
});

</script>

</body>

</html>

**Output:**



**Result:**

**Exercise 4**

**Angular Program to implement a navigation bar.**

**Aim:**

To write an Angular program for navigation menu

**Algorithm:**

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

**Code:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Navigation Menu</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>

<style>

/\* CSS styles \*/

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

</style>

</head>

<body ng-app="menuApp">

<h1>Navigation Menu</h1>

<div ng-controller="menuController">

<ul>

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

</ul>

</div>

<script>

// AngularJS code

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

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

$scope.menus = [

{

title: "Home",

link: "#home"

},

{

title: "About",

link: "#about"

},

{

title: "Services",

link: "#services"

},

{

title: "Contact",

link: "#contact"

}

];

});

</script>

</body>

</html>

**Output:**





**Result:**

**Exercise 5**

**5.a Angular program to switch between layouts.**

**Aim:**

To write an Angular program to switch between layouts.

**Algorithm:**

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

**Code:**

<!DOCTYPE html>

<html>

<head>

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

    </script>

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

    </script>

    <style>

        a{

            display:inline;

            margin-left:150px;

            margin-right:150px;

            font-weight: bold;

        }

    </style>

</head>

<body ng-app="myApp">

    <p> Switching between Layouts:

    <a href="#!/">Main</a>

    <a href="#!item1">Item 1</a>

    <a href="#!item2">Item 2</a>

    </p>

    <div ng-view></div>

    <script>

       const app = angular.module("myApp", ["ngRoute"]);

       app.config(function ($routeProvider) {

            $routeProvider

                .when("/item1", {

                    templateUrl:"item1.html"

                })

                .when("/item2", {

                    templateUrl: "item2.html"

                });

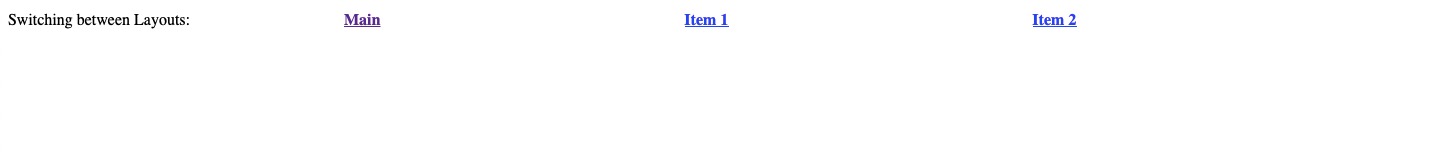
        });

    </script>

</body>

</html>

**Output:**







**Result:**

**5.b Angular JS Animations**

**Aim:**

To implement animation features in AngularJS.

**Algorithm:**

1. Start
2. Start html tag
3. Create an Angular module and controller.
4. 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.
5. Add CSS animations to make the divs “fade in” or “roll down” to get displayed.
6. Stop

**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>

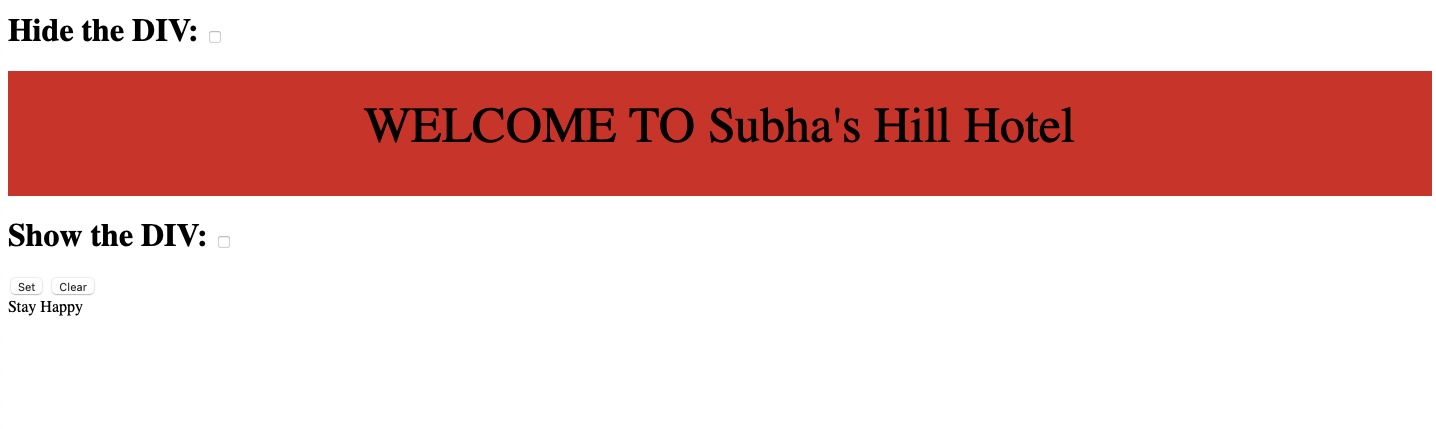
            var app = angular.module('myApp', ['ngAnimate']);

        </script>

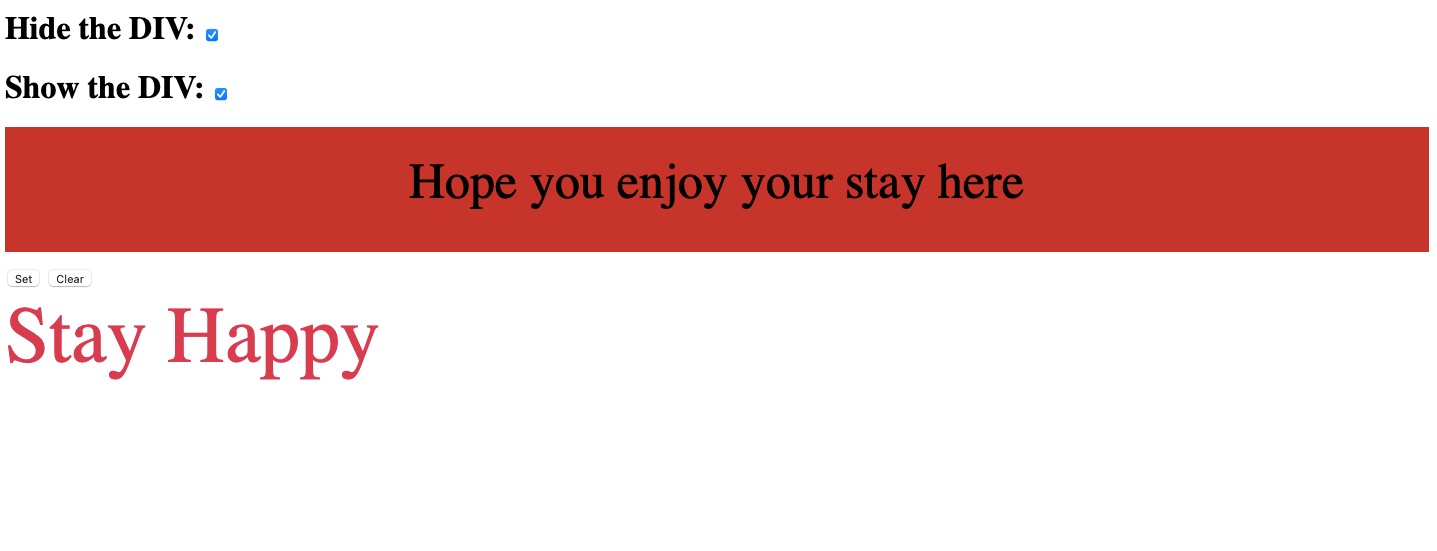
    </body>

</html>

Output:







**Result:**

**Exercise 6**

**Angular Program to create form with real time updations**

**Aim:**

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

**Algorithm:**

1. Start
2. Start html tag
3. Create an Angular module and controller and initialize using ng-init.
4. 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.
5. Create a database “test1” and a table “formdeets” in the phpMyAdmin page.
6. Create a “connect.php” file, and include functions to connect to the database “test1”.
7. 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.
8. Add error handling routines to display messages in case database connection isn’t successful.
9. Stop

**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';

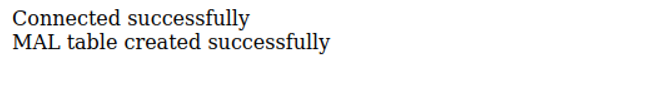
            }

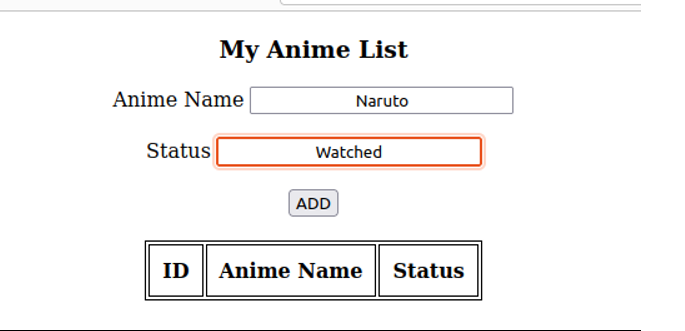
        }

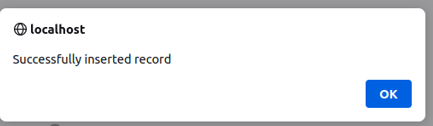
    }

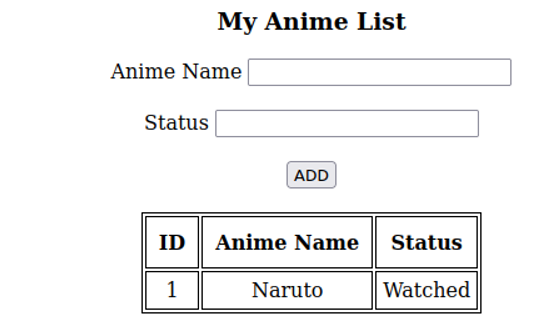
?>

**Output:**









**Result:**

**Exercise 7**

**Video Player using AngularJS**

**Aim:**

To implement a video player using an AngularJS program

**Algorithm:**

1. Start
2. Start HTML tag
3. Create an AngularJS module and controller for the video player.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. Add error handling and error messages to handle cases where the video file or stream cannot be loaded or played properly.

**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();

            };

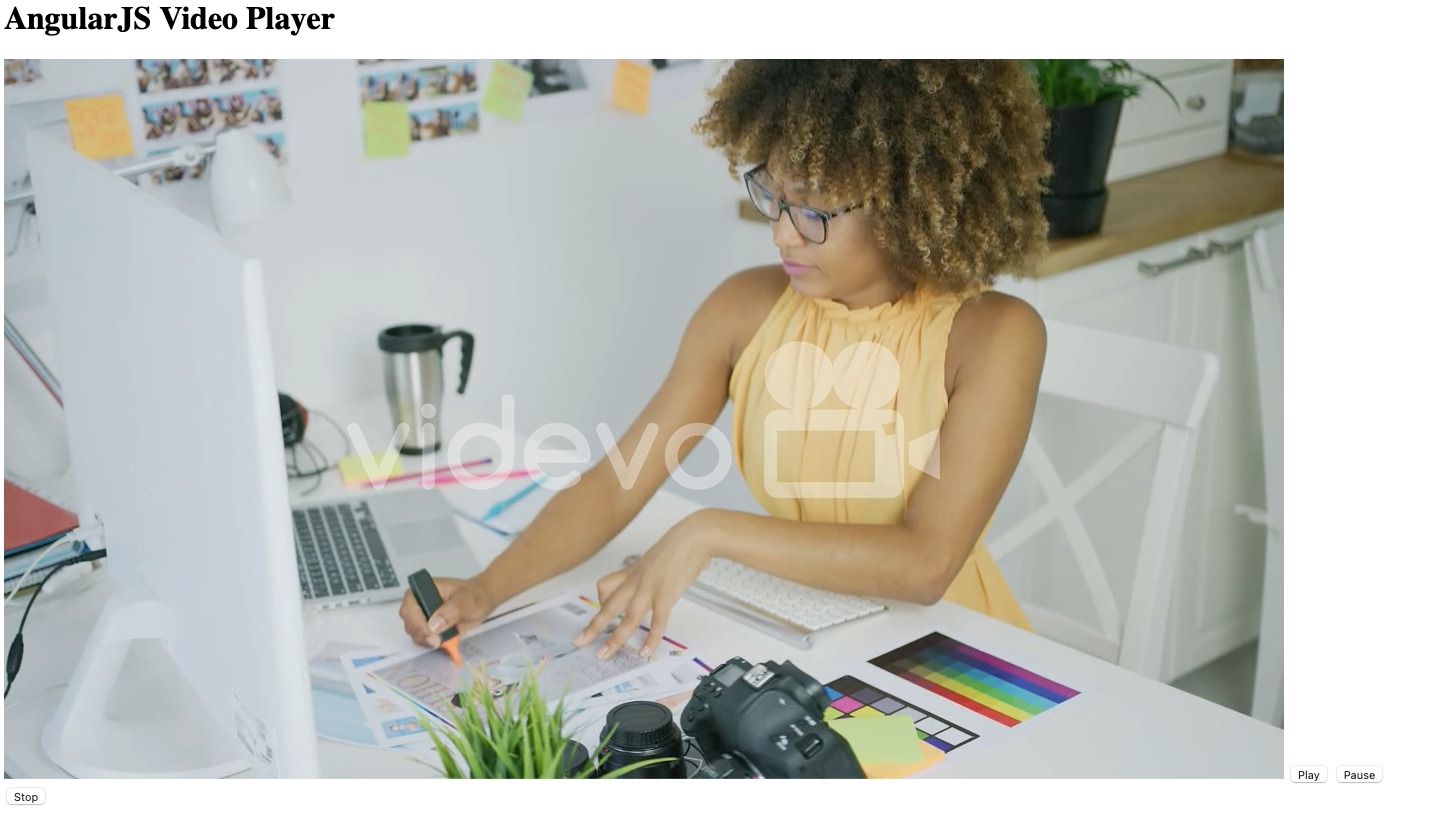
        });

    </script>

</body>

</html>

**Output:**



**Result:**

\

**Exercise 8**

**Text Editor using AngularJS**

**Aim:**

To implement a text editor using AngularJS

**Algorithm:**

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

**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;

            };

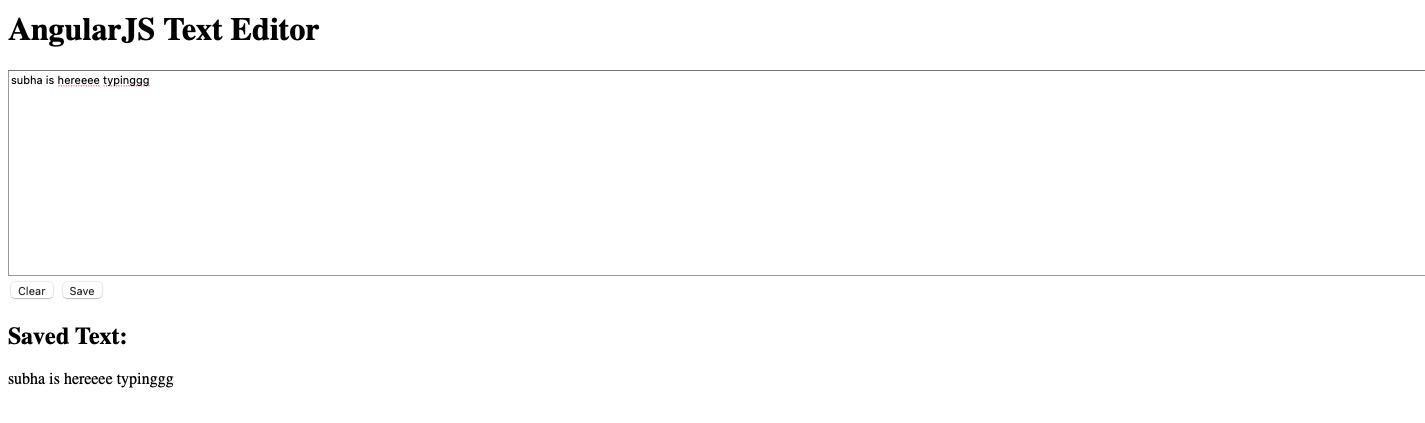
        });

    </script>

</body>

</html>

**Output:**



**Result:**