**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

FACULTY OF TECHNOLOGY AND ENGINEERING

**Devang Patel Institute of Advance Technology & Research**

DEPARTMENT OF COMPUTER ENGINEERING

**CE247 Web Technologies**

**Semester:** IV

**Academic Year :** 2019-20

**PRACTICAL LIST**

# Part–1 ( HTML 4.01 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Study following HTML4.01 tags along with necessary attributes: Doctype and Basic structure tags, Heading tags, Meta tags, Paragraphs,  Text Formatting tags, Link tags, List tag, Image, Table, Form, Iframe, HTML Encode | 9/12/2019 | 06 |  |
| **2.** | Create Personal Home page using all above HTML tags. Create layout of web page using div tag. | 9/12/2019 | 07-08 |  |

# Part–2 ( HTML 5 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Study and Implement new tags which are added into HTML5 along with required attributes. | 30/12/2019 | 09 |  |
| **2.** | Create registration form for social networking website using HTML5 form tag and attributes. Apply necessary validation using  HTML5 pattern attributes which Specifies a regular expression to check th input value against. | 6/1/2020 | 10 |  |
| **3.** | Create different graphics using HTML CANVAS. Apply required properties and methods. | 20/1/2020 | 11-12 |  |
| **4.** | Study HTML5 APIs. | 3/2/2020 | 13-25 |  |

# Part–3 ( CSS2.0 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Implement following properties of CSS 2.0.   1. Background, Font and Text 2. Margin, Padding and Border (Box Model) 3. Table 4. Display 5. Position 6. Floating 7. Pseudo elements | 16/12/2019 | 26 |  |
| **2.** | Create Cascading menu using CSS2.0. | 16/12/2019 | 27 |  |

# Part–4 ( CSS3.0 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Implement Back grounds and Borders properties using CSS 3.0 | 10/2/2020 | 28-29 |  |
| **2.** | Implement Text Effects properties using CSS 3.0 | 10/2/2020 | 28-29 |  |
| **3.** | Implement 2Dand 3D Transform properties using CSS 3.0 | 10/2/2020 | 28-29 |  |
| **4.** | Implement Animations properties using CSS 3.0 | 17/2/2020 | 28-29 |  |
| **5.** | Implement Multiple Column Layout properties using CSS 3.0 | 17/2/2020 | 28-29 |  |
| **6.** | Implement User Interface properties using CSS 3.0 | 17/2/2020 | 28-29 |  |
| **7.** | Refer Practical 2 and apply flexible layout to your page. | 24/2/2020 | 28-29 |  |

# Part–5 ( Java Scripts & DOM )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Create JavaScript function for basic arithmetic operations (Addition, Subtraction, Multiplication and Division). Create proper design layout which asks two numbers to enter from user and display result of respective operation on clicking respective button. | 3/12/2019 | 30 |  |
| **2.** | Create HTML form like any registration form with all necessary controls. Write a JavaScript which retrieve possible value from all controls and display on webpage. Use some style rules to format | 17/12/2019 | 31 |  |
| **3.** | Create JavaScript for form Validation for following tasks   1. Text box must have required value. 2. Textbox must accept only letters. 3. Textbox must accept only numbers. 4. Textbox must contain character in the range of 6-12. 5. Compare entered password and confirmed password. 6. Check whether radio button, check box or dropdown box has selected or not. 7. Email validation. 8. Mobile number validation.   Design proper HTML form and perform all above task. | 31/12/2019 | 32 |  |
| **4.** | Create Dynamic table using DOM which perform following tasks:   * Add new Row and Column * Delete Row and cell. | 7/1/2020 | 33 |  |

# Part–6 ( BASICJQUERY )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Implement following tasks using jQuery:   * Sliding page elements * Hiding and Showing elements * Fading elements * Toggling elements * Stopping Effects | 21/1/2020 | 34 |  |
| **2.** | Create Animated Login form. | 4/2/2020 | 35 |  |

# Part–7 ( BASICJ PHP )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Practical** | **Date** | **Page No.** | **Remark** |
| **1.** | Installation and configuration of WAMP/XAMPP | 11/2/2020 | 36-39 |  |
| **2.** | Study and demonstrate php syntax, data type, variable, function, array, super global variable and form. | 18/2/2020 | 40 |  |
| **3.** | Study and demonstrate MySQL connection and CRUID operations with php. | 3/3/2020 | 41 |  |

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF TECHNOLOGY & ENGINEERING**

**DEVANG PATEL INSTITUTE OF ADVANCED TECHNOLOGY AND RESEARCH**

**DEPARTMENT OF COMPUTER ENGINEERING**

|  |  |
| --- | --- |
| **Subject Name & Code : CE247 Web Technologies** | **Semester: IV** |

**Course Outcomes:**

Upon completion of this course, students will be able to do the following:

* Students will able to develop static and dynamic website and web base applications.
* Students will use knowledge of the subject in higher semester for subjects like Advance Java Technology and .Net Web technology and also for Project development.

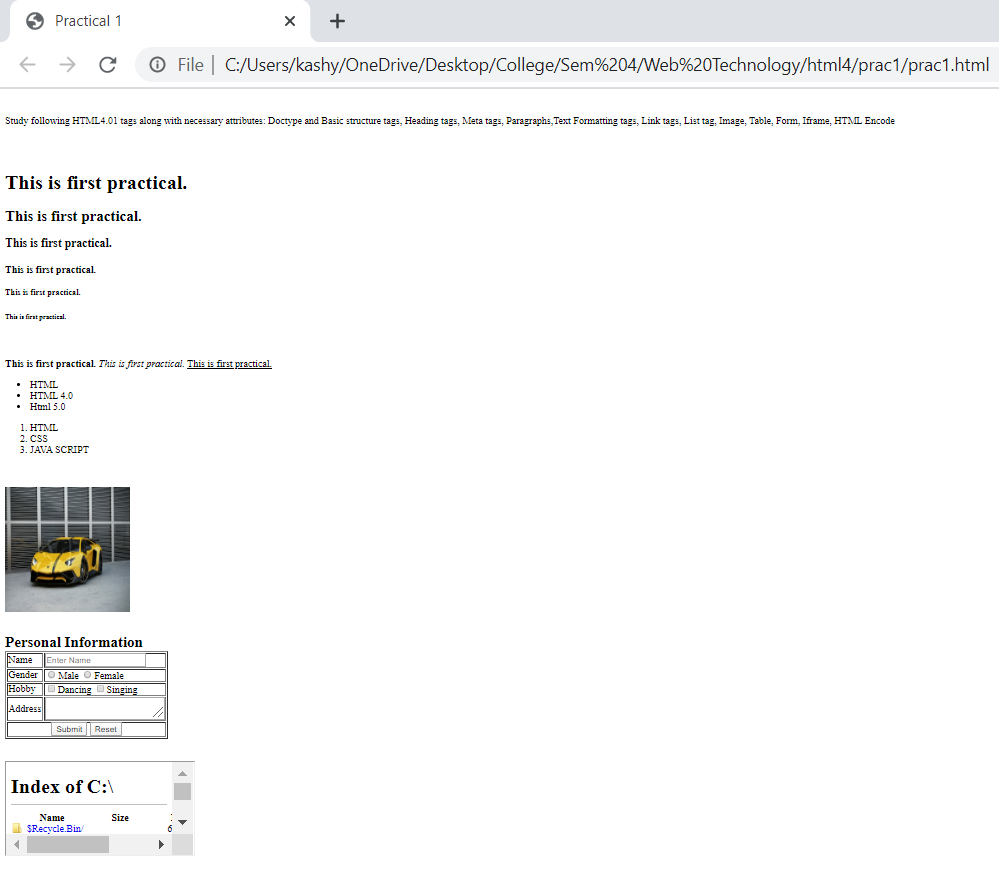
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** | **Practical Name** | **Co1** | **Co2** |
| 1 | Part 1: HTML 4.0 | **√** |  |
| 2 | Part 2: HTML 5.0 | **√** | **√** |
| 3 | Part 3: CSS 2.0 | **√** |  |
| 4 | Part 4: CSS 3.0 | **√** | **√** |
| 5 | Part 5: JavaScript &  DOM | **√** | **√** |
| 6 | Part 6: Basic jQuery | **√** | **√** |
| 7 | Part 7: Basic PHP | **√** | **√** |

**PART – 1 ( HTML 4.01 )**

**PRACTICAL - 1**

**AIM** : Study following HTML4.01 tags along with necessary attributes:Doctype and Basic structure tags, Heading tags, Meta tags, Paragraphs,Text Formatting tags, Link tags, List tag, Image, Table, Form, Iframe,HTML Encode.

**OUTPUT :**



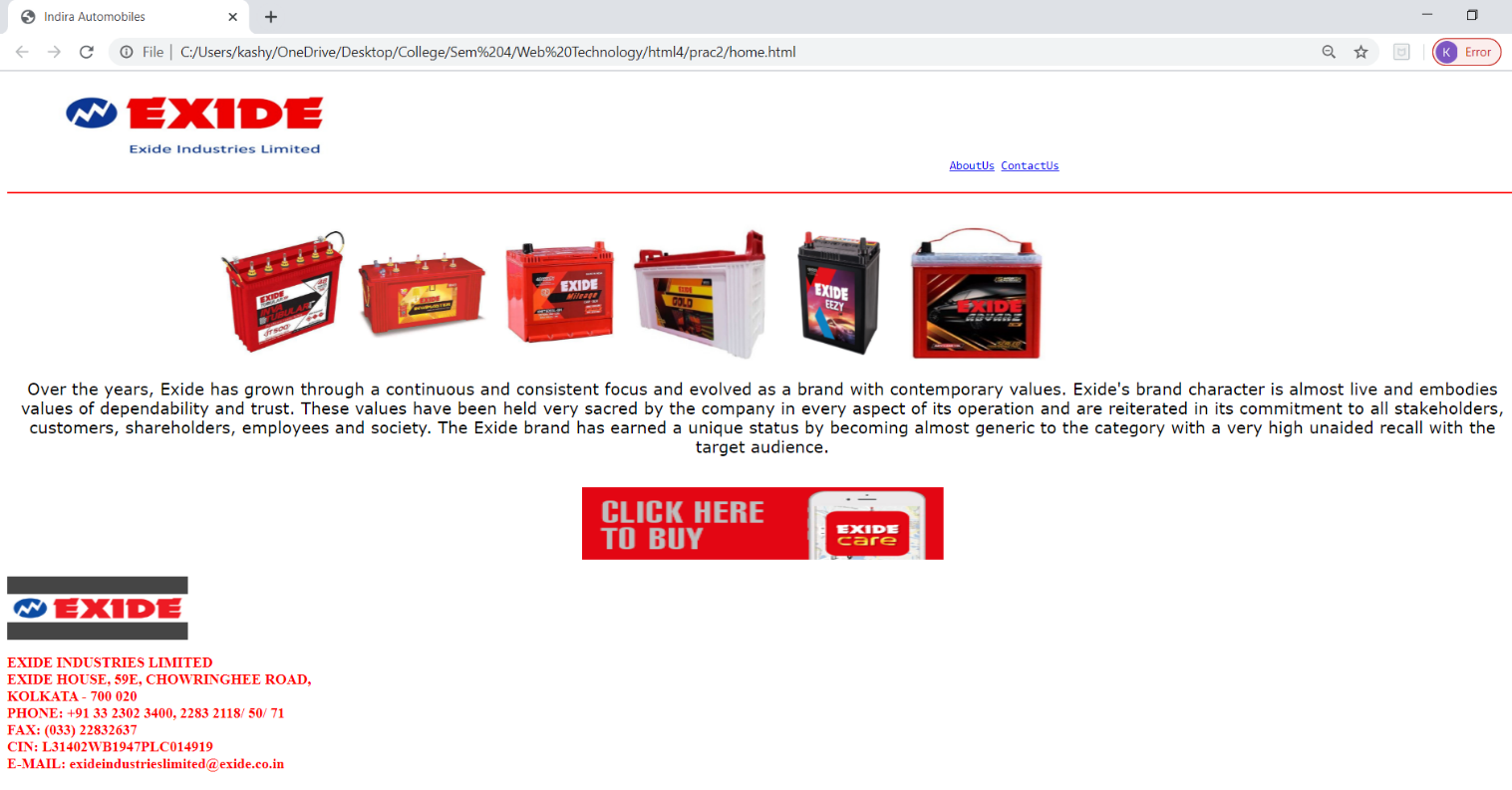
**CONCLUSION:**

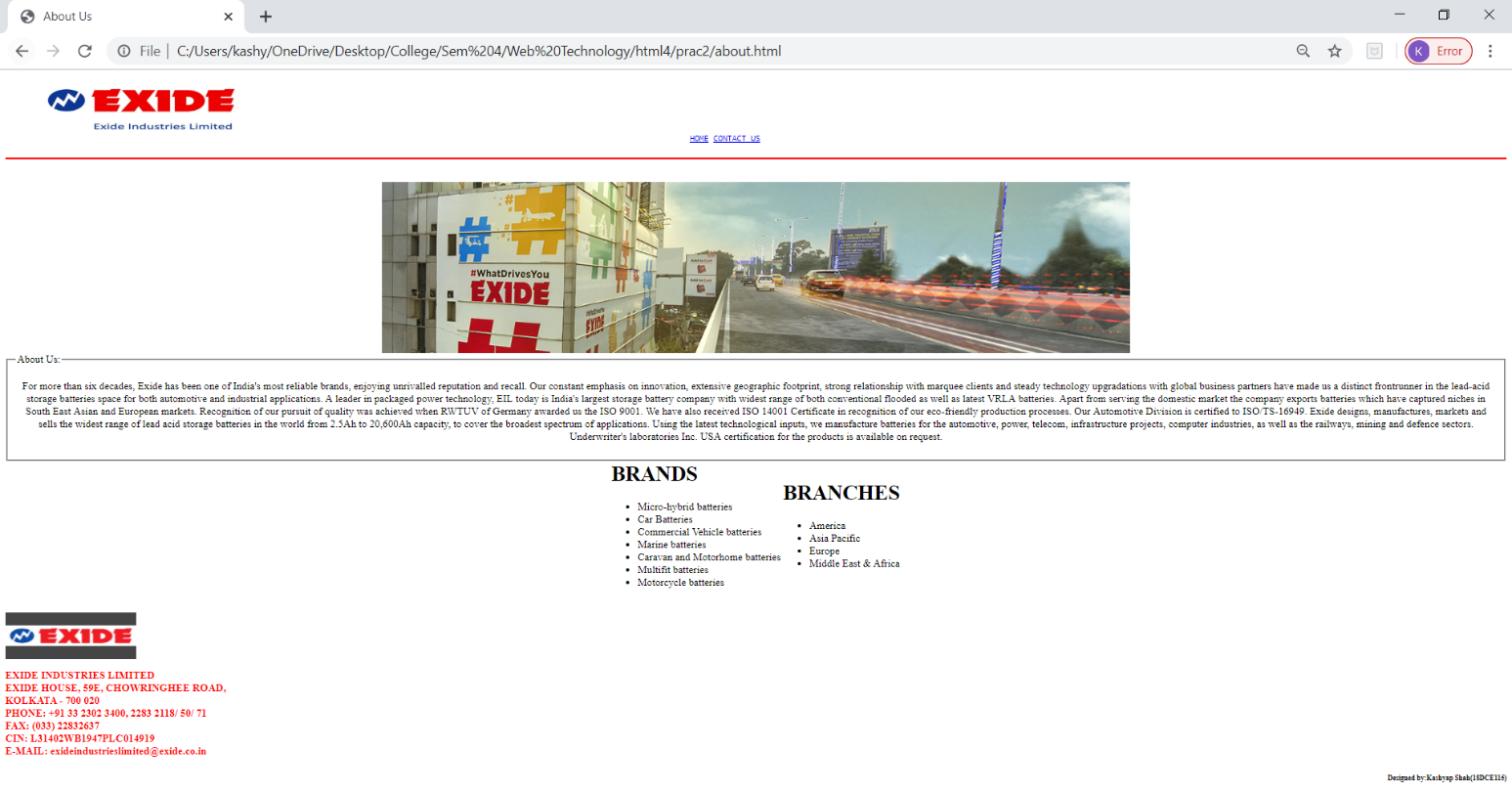
Here in this practical, we learned about the basic tags of HTML.

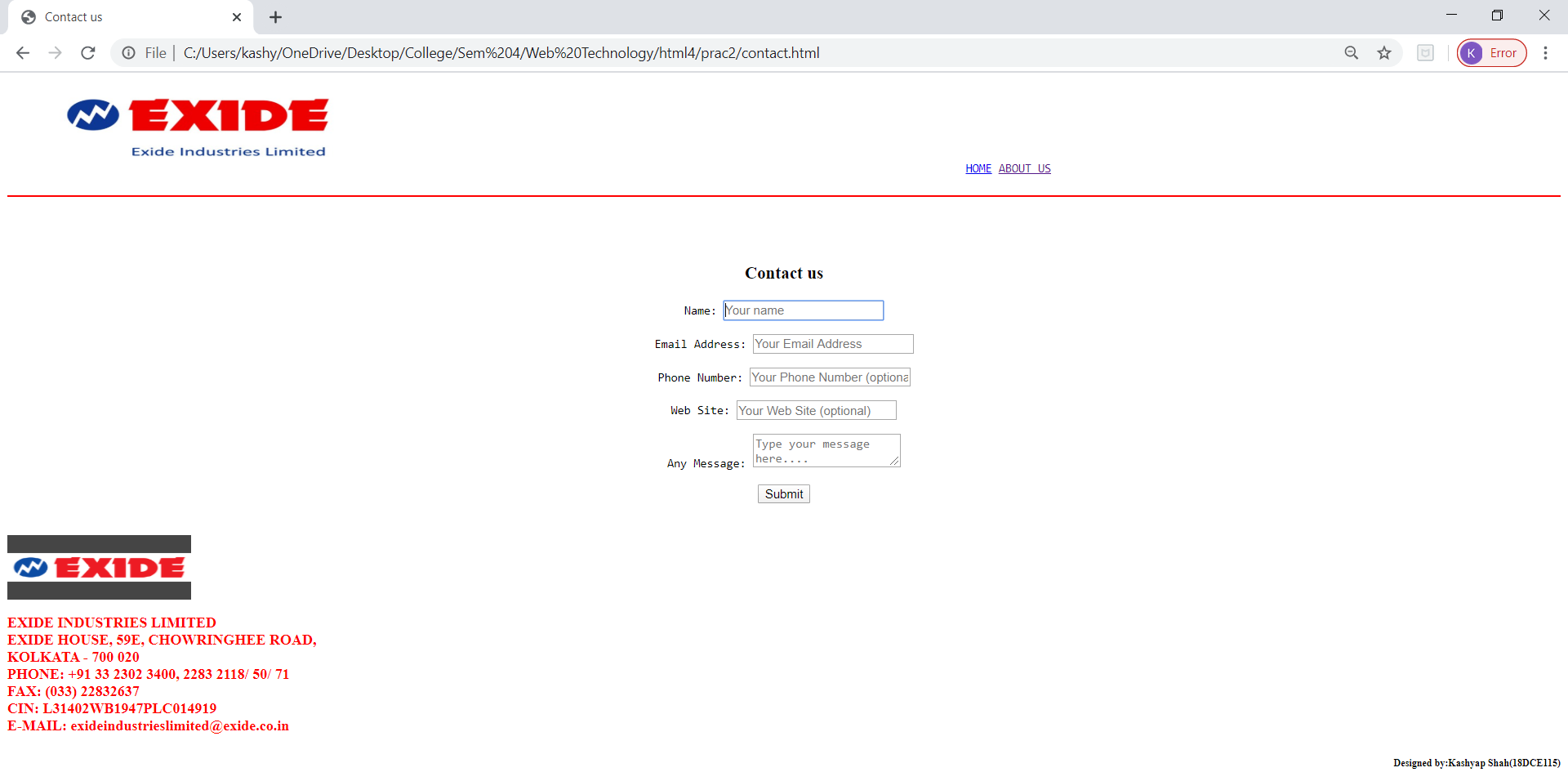
**PRACTICAL - 2**

**AIM:** Create Personal Home page using all above HTML tags. Create layout of web page using div tag.

**OUTPUT:**







**CONCLUSION:**

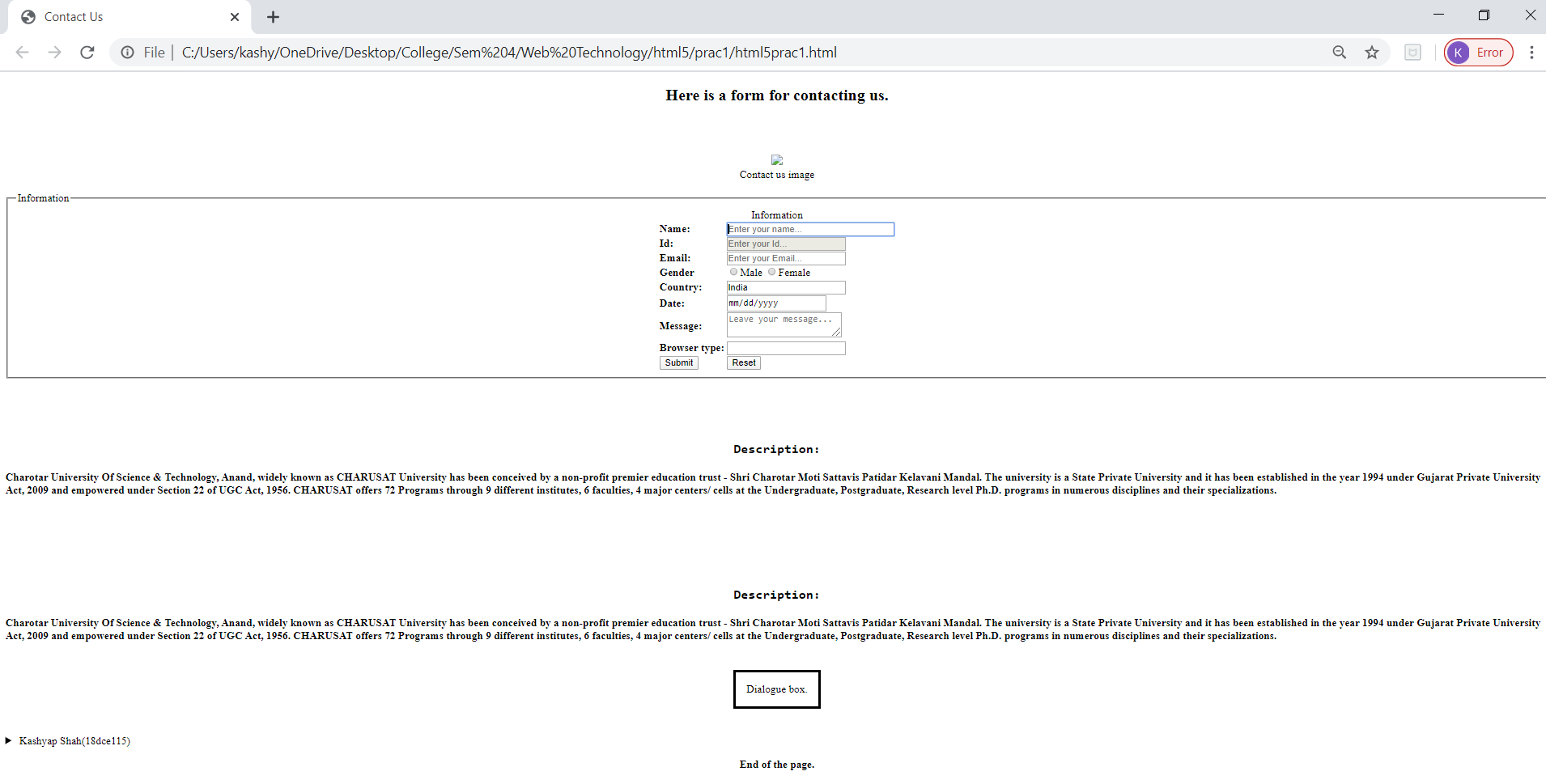
We created our homepage by using the HTML tags.

**PART – 2 ( HTML 5 )**

**PRACTICAL - 1**

**AIM:** Study and Implement new tags which are added into HTML5 along with required attributes.

**OUTPUT:**



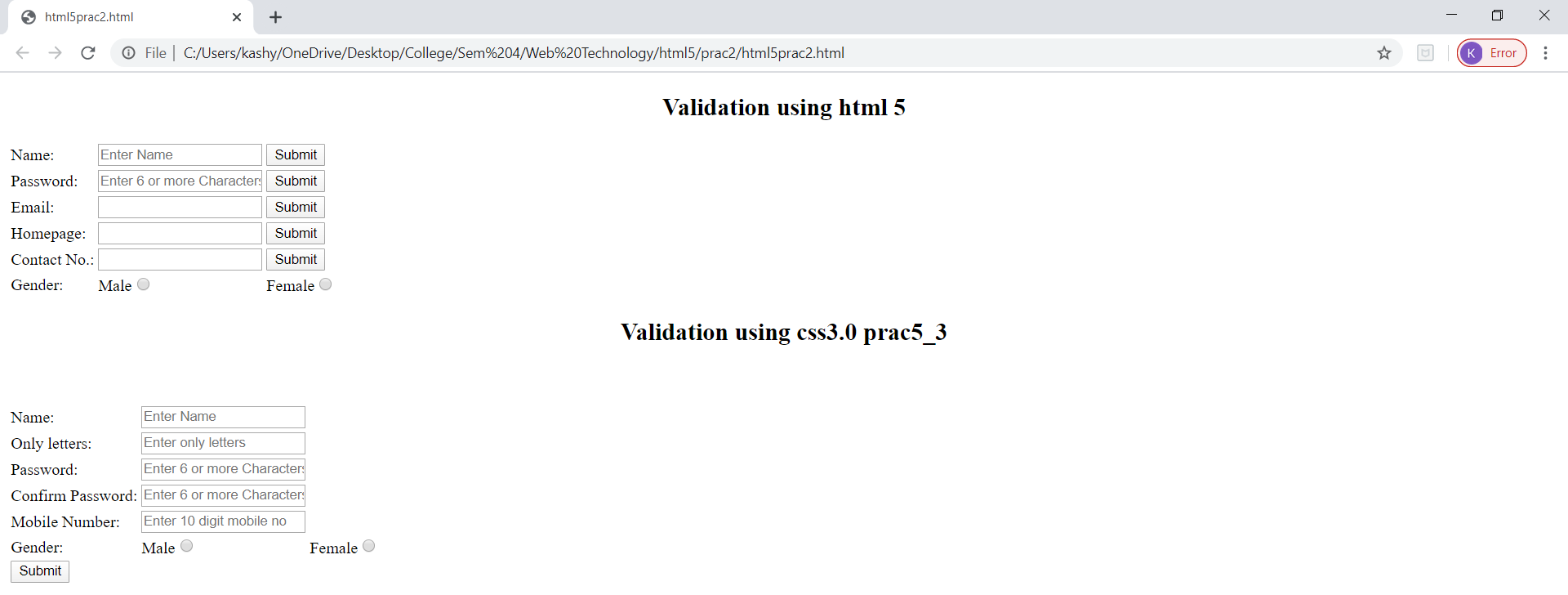
**CONCLUSION:**

In this practical we learn implementation of new tag which are added into HTML5.

**PRACTICAL - 2**

**AIM:** Create registration form for social networking website using HTML5 form tag and attributes. Apply necessary validation using HTML5 pattern attributes which Specifies a regular expression to check the input value against.

**OUTPUT:**



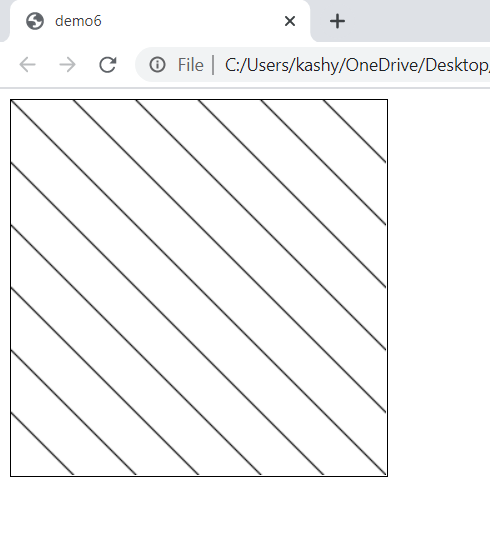
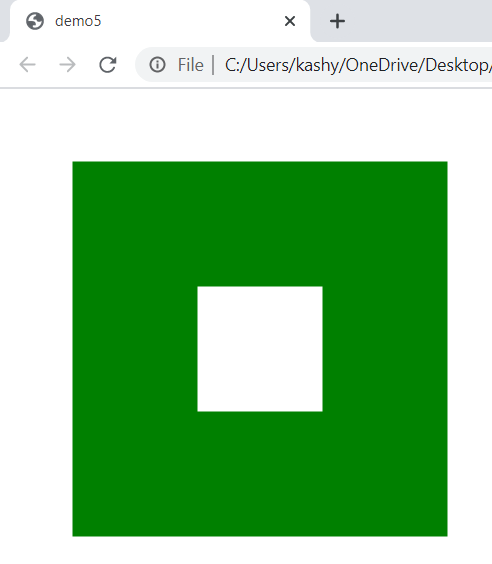
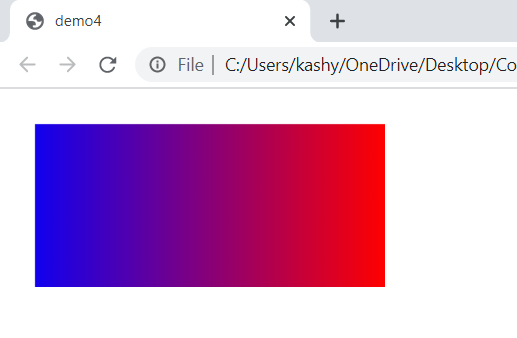
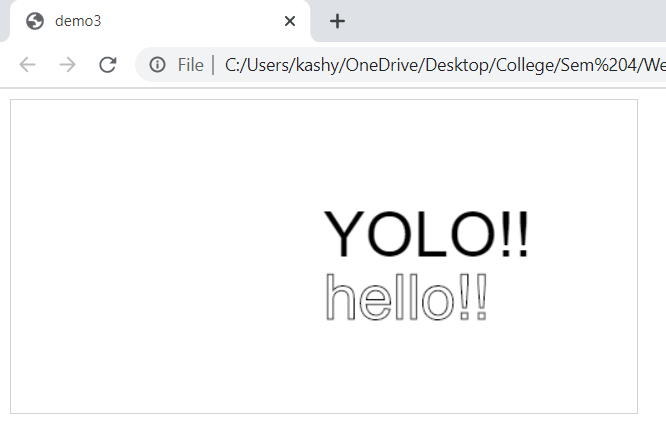
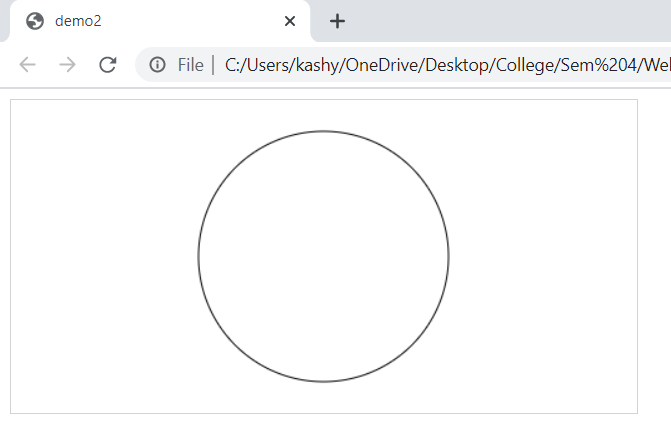
**CONCLUSION:**

In this practical we performvalidation usingHTML5 pattern attributes which Specifies a regular expression to check the input value against.

**PRACTICAL - 3**

**AIM:** Create different graphics using HTML CANVAS. Apply required properties and methods.

**OUTPUT :**



**CONCLUSION:**

In this practical we learn how to use HTML CANVAS and their functionality.

**PRACTICAL - 5**

**Aim:** Study HTML5 APIs.

An API (or application programming interface) is “is a set of subroutine definitions, protocols, and tools for building application software.” In an HTML API, the definitions and protocols are in the HTML itself, and the tools look in HTML for the configuration.

The following five are the most used APIs:-

1. [HTML Drag/Drop](https://www.w3schools.com/html/html5_draganddrop.asp)
2. [HTML Web Storage](https://www.w3schools.com/html/html5_webstorage.asp)
3. [HTML Web Workers](https://www.w3schools.com/html/html5_webworkers.asp)
4. [HTML SSE](https://www.w3schools.com/html/html5_serversentevents.asp)
5. HTML Geolocation

## 1.Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard: Any element can be draggable.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

<img draggable="true">

## What to Drag - ondragstart and setData()

Then, specify what should happen when the element is dragged.

In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data:

function drag(ev) {  
  ev.dataTransfer.setData("text", ev.target.id);  
}

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

## Where to Drop - ondragover

The ondragover event specifies where the dragged data can be dropped.

By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event:

*event*.preventDefault()

## Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs.

In the example above, the ondrop attribute calls a function, drop(event):

function drop(ev) {  
  ev.preventDefault();  
  var data = ev.dataTransfer.getData("text");  
  ev.target.appendChild(document.getElementById(data));  
}

Code explained:

* Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
* Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
* The dragged data is the id of the dragged element ("drag1")
* Append the dragged element into the drop element

# 2.Web Storage

## What is HTML Web Storage?

With web storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML Web Storage Objects

HTML web storage provides two objects for storing data on the client:

* window.localStorage - stores data with no expiration date
* window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

if (typeof(Storage) !== "undefined") {  
  // *Code for localStorage/sessionStorage.*  
} else {  
  // Sorry! No Web Storage support..  
}

## The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

### **Example**

// Store  
localStorage.setItem("lastname", "Smith");  
  
// Retrieve  
document.getElementById("result").innerHTML = localStorage.getItem("lastname");

Example explained:

* Create a localStorage name/value pair with name="lastname" and value="Smith"
* Retrieve the value of "lastname" and insert it into the element with id="result"

## **The sessionStorage Object**

The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

# 3.Web Workers

## What is a Web Worker?

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

## HTML Web Workers Example

The example below creates a simple web worker that count numbers in the background:

### **Example**

Count numbers:

Start Worker Stop Worker

## Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

if (typeof(Worker) !== "undefined") {  
  // Yes! Web worker support!  
  // *Some code.....*  
} else {  
  // Sorry! No Web Worker support..  
}

## Create a Web Worker File

Now, let's create our web worker in an external JavaScript.

Here, we create a script that counts. The script is stored in the "demo\_workers.js" file:

var i = 0;  
  
function timedCount() {  
  i = i + 1;  
  postMessage(i);  
  setTimeout("timedCount()",500);  
}  
  
timedCount();

The important part of the code above is the postMessage() method - which is used to post a message back to the HTML page.

**Note:** Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

## Create a Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page.

The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo\_workers.js":

if (typeof(w) == "undefined") {  
  w = new Worker("demo\_workers.js");  
}

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

w.onmessage = function(event){  
  document.getElementById("result").innerHTML = event.data;  
};

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

## Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

w.terminate();

## Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

w = undefined;

## Full Web Worker Example Code

We have already seen the Worker code in the .js file. Below is the code for the HTML page:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<p>Count numbers: <output id="result"></output></p>  
<button onclick="startWorker()">Start Worker</button>  
<button onclick="stopWorker()">Stop Worker</button>  
  
<script>  
var w;  
  
function startWorker() {  
  if (typeof(Worker) !== "undefined") {  
    if (typeof(w) == "undefined") {  
      w = new Worker("demo\_workers.js");  
    }  
    w.onmessage = function(event) {  
      document.getElementById("result").innerHTML = event.data;  
    };  
  } else {  
    document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";  
  }  
}  
  
function stopWorker() {  
  w.terminate();  
  w = undefined;  
}  
</script>  
  
</body>  
</html>

## Web Workers and the DOM

Since web workers are in external files, they do not have access to the following JavaScript objects:

* The window object
* The document object
* The parent object

# 4.Server-Sent Events

## Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

## Receive Server-Sent Event Notifications

The EventSource object is used to receive server-sent event notifications:

### **Example**

var source = new EventSource("demo\_sse.php");  
source.onmessage = function(event) {  
  document.getElementById("result").innerHTML += event.data + "<br>";  
};

[Try it Yourself »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_sse)

Example explained:

* Create a new EventSource object, and specify the URL of the page sending the updates (in this example "demo\_sse.php")
* Each time an update is received, the onmessage event occurs
* When an onmessage event occurs, put the received data into the element with id="result"

## Check Server-Sent Events Support

In the tryit example above there were some extra lines of code to check browser support for server-sent events:

if(typeof(EventSource) !== "undefined") {  
  // Yes! Server-sent events support!  
  // *Some code.....*  
} else {  
  // Sorry! No server-sent events support..  
}

## Server-Side Code Example

For the example above to work, you need a server capable of sending data updates (like PHP or ASP).

The server-side event stream syntax is simple. Set the "Content-Type" header to "text/event-stream". Now you can start sending event streams.

Code in PHP (demo\_sse.php):

<?php  
header('Content-Type: text/event-stream');  
header('Cache-Control: no-cache');  
  
$time = date('r');  
echo "data: The server time is: {$time}\n\n";  
flush();  
?>

Code in ASP (VB) (demo\_sse.asp):

<%  
Response.ContentType = "text/event-stream"  
Response.Expires = -1  
Response.Write("data: The server time is: " & now())  
Response.Flush()  
%>

Code explained:

* Set the "Content-Type" header to "text/event-stream"
* Specify that the page should not cache
* Output the data to send (**Always** start with "data: ")
* Flush the output data back to the web page

## The EventSource Object

|  |  |
| --- | --- |
| **Events** | **Description** |
| onopen | When a connection to the server is opened |
| onmessage | When a message is received |
| onerror | When an error occurs |

In the examples above we used the onmessage event to get messages. But other events are also available:

# 5.Geolocation

## Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

## Using HTML Geolocation

The getCurrentPosition() method is used to return the user's position.

The example below returns the latitude and longitude of the user's position:

### **Example**

<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

Example explained:

* Check if Geolocation is supported
* If supported, run the getCurrentPosition() method. If not, display a message to the user
* If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
* The showPosition() function outputs the Latitude and Longitude

The example above is a very basic Geolocation script, with no error handling.

## Handling Errors and Rejections

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

### **Example**

function showError(error) {  
  switch(error.code) {  
    case error.PERMISSION\_DENIED:  
      x.innerHTML = "User denied the request for Geolocation."  
      break;  
    case error.POSITION\_UNAVAILABLE:  
      x.innerHTML = "Location information is unavailable."  
      break;  
    case error.TIMEOUT:  
      x.innerHTML = "The request to get user location timed out."  
      break;  
    case error.UNKNOWN\_ERROR:  
      x.innerHTML = "An unknown error occurred."  
      break;  
  }  
}

**Displaying the Result in a Map**

To display the result in a map, you need access to a map service, like Google Maps.

In the example below, the returned latitude and longitude is used to show the location in a Google Map (using a static image):

### **Example**

function showPosition(position) {  
  var latlon = position.coords.latitude + "," + position.coords.longitude;  
  
  var img\_url = "https://maps.googleapis.com/maps/api/staticmap?center=  
  "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR\_KEY";  
  
  document.getElementById("mapholder").innerHTML = "<img src='"+img\_url+"'>";  
}

## Location-specific Information

This page has demonstrated how to show a user's position on a map.

Geolocation is also very useful for location-specific information, like:

* Up-to-date local information
* Showing Points-of-interest near the user
* Turn-by-turn navigation (GPS)

## The getCurrentPosition() Method - Return Data

The getCurrentPosition() method returns an object on success. The latitude, longitude and accuracy properties are always returned. The other properties are returned if available:

|  |  |
| --- | --- |
| **Property** | **Returns** |
| coords.latitude | The latitude as a decimal number (always returned) |
| coords.longitude | The longitude as a decimal number (always returned) |
| coords.accuracy | The accuracy of position (always returned) |
| coords.altitude | The altitude in meters above the mean sea level (returned if available) |
| coords.altitudeAccuracy | The altitude accuracy of position (returned if available) |
| coords.heading | The heading as degrees clockwise from North (returned if available) |
| coords.speed | The speed in meters per second (returned if available) |
| timestamp | The date/time of the response (returned if available) |

## Geolocation Object - Other interesting Methods

The Geolocation object also has other interesting methods:

* watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
* clearWatch() - Stops the watchPosition() method.

The example below shows the watchPosition() method. You need an accurate GPS device to test this (like smartphone):

### **Example**

<script>  
var x = document.getElementById("demo");  
function getLocation() {  
  if (navigator.geolocation) {  
    navigator.geolocation.watchPosition(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>

**PART – 3 ( CSS 2.0 )**

**PRACTICAL - 1**

**AIM:** Implement following properties of CSS 2.0.

1. Background, Font and Text

2. Margin, Padding and Border (Box Model)

3. Table

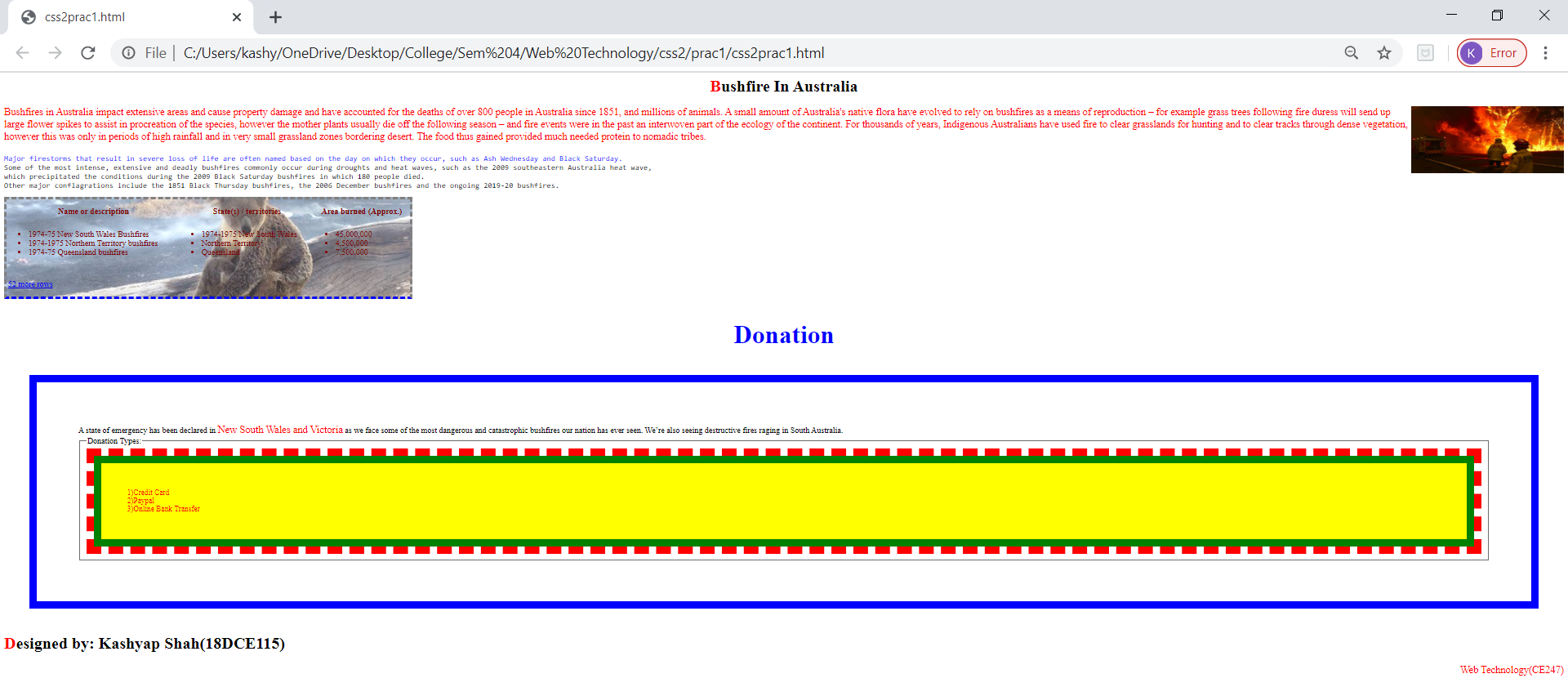
4. Display

5. Position

6. Floating

7. Pseudo elements

**OUTPUT :**

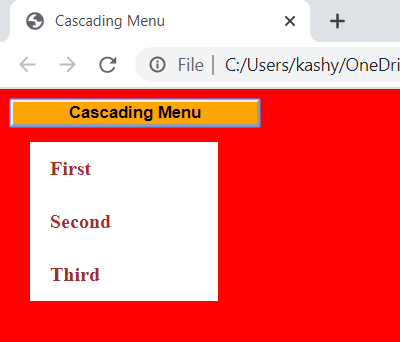


**CONCLUSION:** In this practical we learn how to made Cascading menu using CSS2.0.

**PRACTICAL - 2**

**Aim:** Create Cascading menu using CSS2.0.

**Output:**



**Conclusion:**

Here we learn about cascading Menu in CSS Properties.

**PART-4 ( CSS 3.0 )**

**PRACTICAL-1**

**AIM:** 1 Implement Back grounds and Borders properties using CSS 3.0 .

2 Implement Text Effects properties using CSS 3.0 .

3 Implement 2Dand 3D Transform properties using CSS 3.0 .

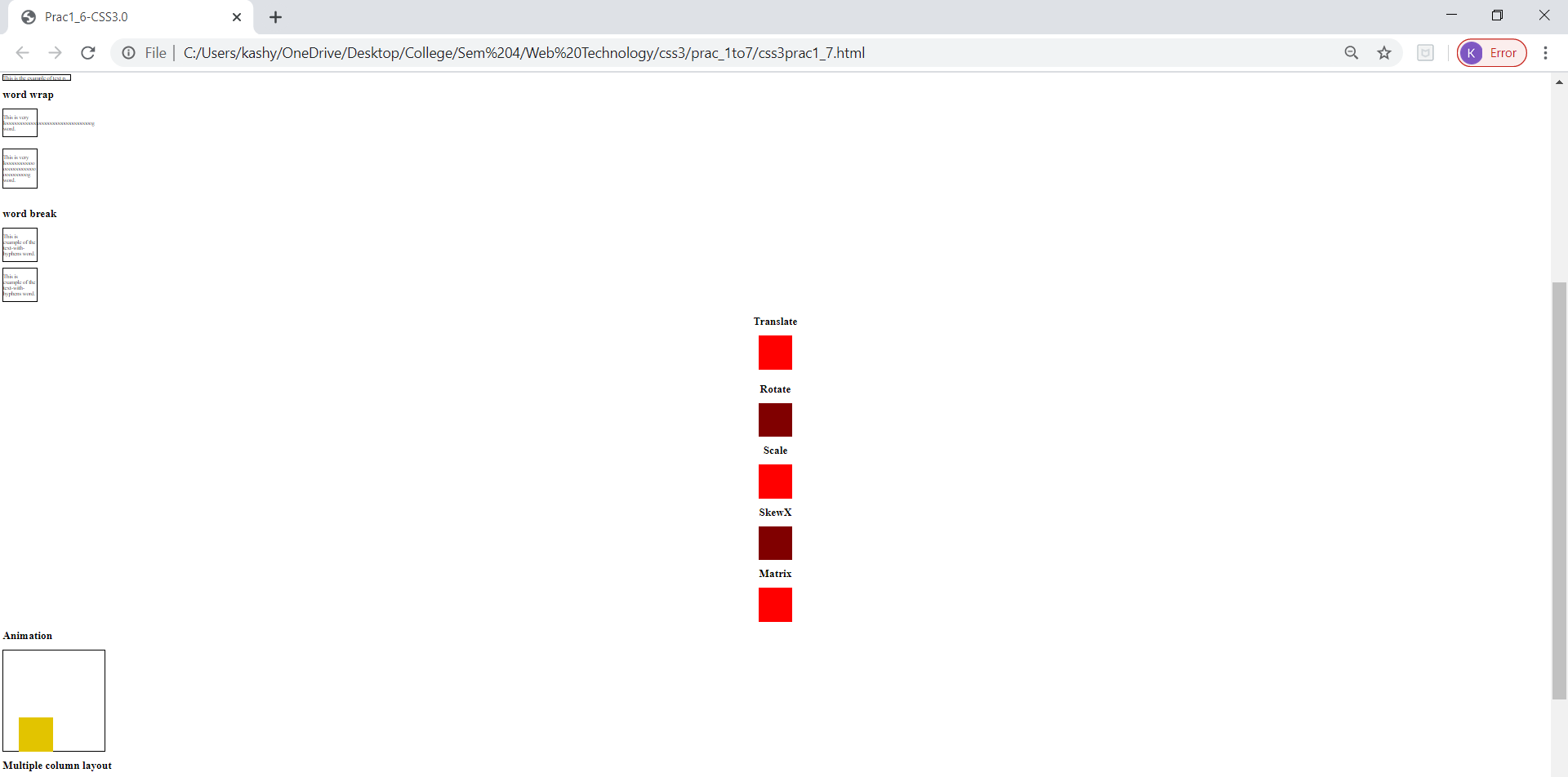
4 Implement Animations properties using CSS 3.0 .

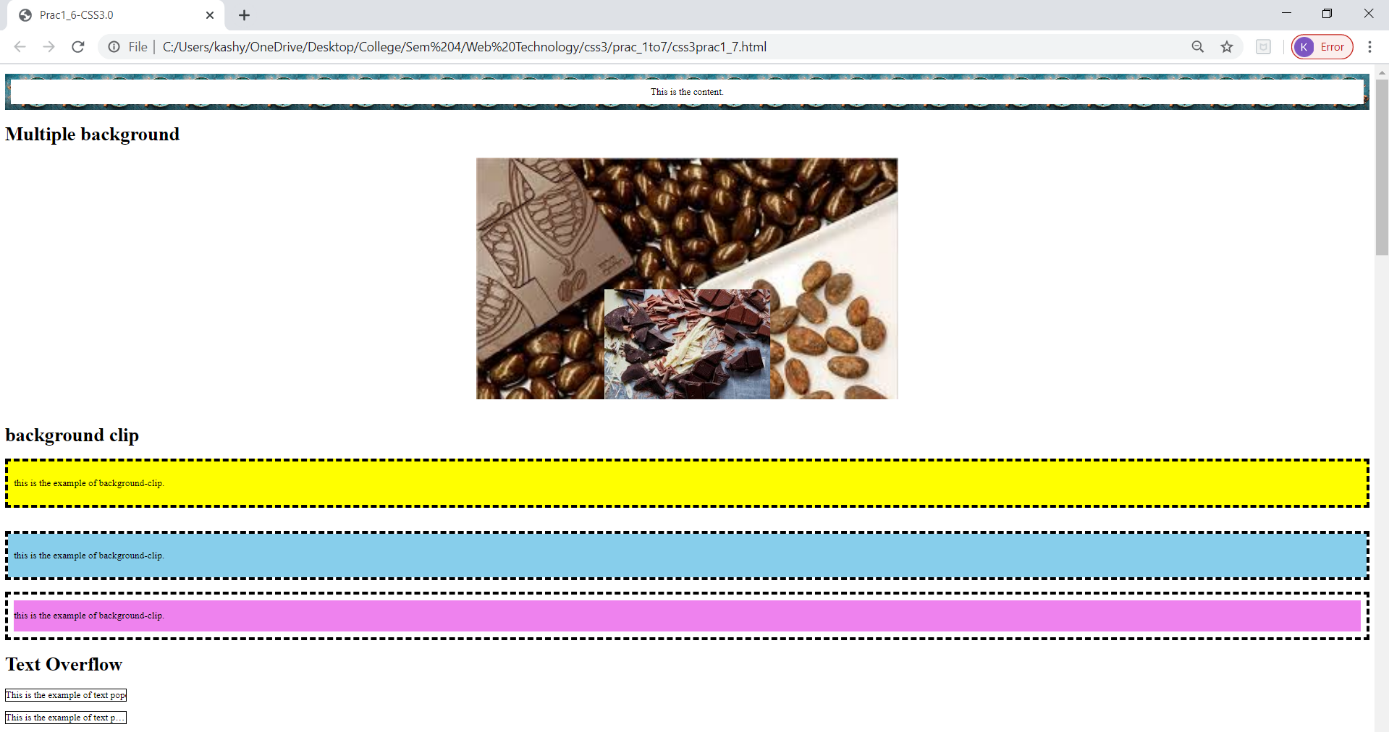
5 Implement Multiple Column Layout properties using CSS 3.0 .

6 Implement User Interface properties using CSS 3.0 .

7 Refer Practical 2 and apply flexible layout to your page.

**OUTPUT:**







**CONCLUSION:**

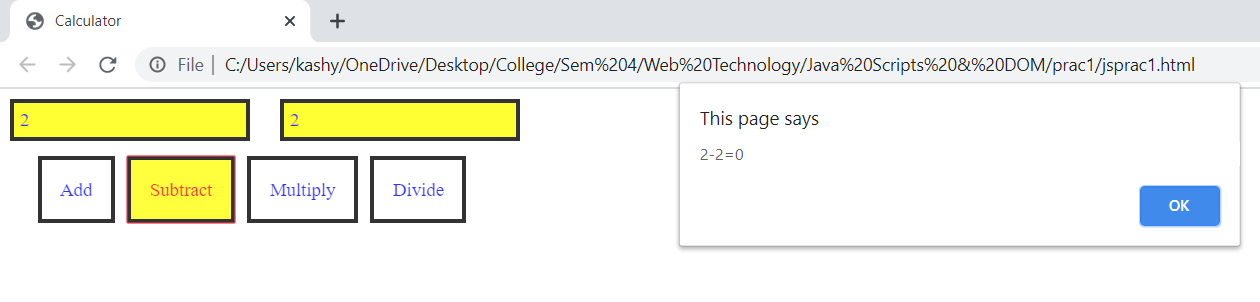
We implemented background and border properties using CSS3.0

**PART–5 ( JAVA SCRIPTS & DOM )**

**PRACTICAL - 1**

**AIM:** Create JavaScript function for basic arithmetic operations (Addition,Subtraction, Multiplication and Division). Create proper design layout which asks two numbers to enter from user and display result of respective operation on clicking respective button.

**OUTPUT:**



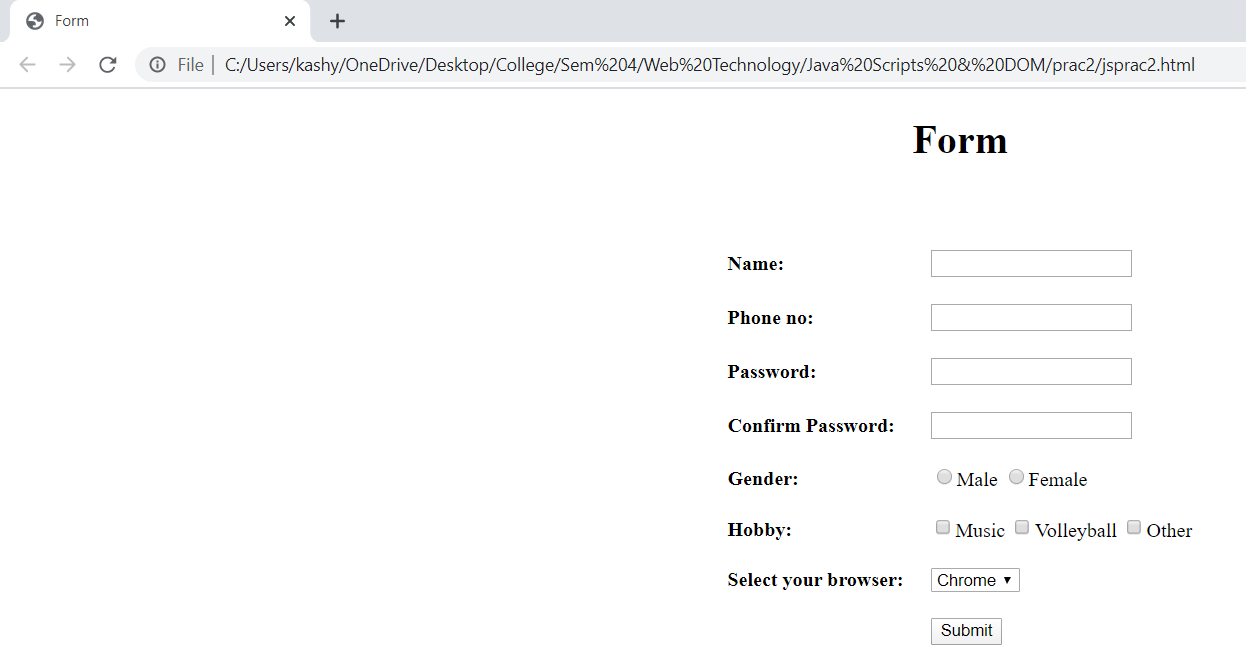
**CONCLUSION:**

In this practical we made basic calculator using JavaScript basic function.

**PRACTICAL - 2**

**AIM:** Create HTML form like any registration form with all necessary controls. Write a JavaScript which retrieve all possible value from controls and display on webpage. Use some style rules to format

**OUTPUT:**



**CONCLUSION:**

In this practical we learn how to retrieve data from form using JavaScript.

**PRACTICAL - 3**

**AIM:** Create JavaScript for form Validation for following tasks

a. Text box must have required value.

b. Textbox must accept only letters.

c. Textbox must accept only numbers.

d. Textbox must contain character in the range of 6-12.

e. Compare entered password and confirmed password.

f. Check whether radio button, check box or dropdown box

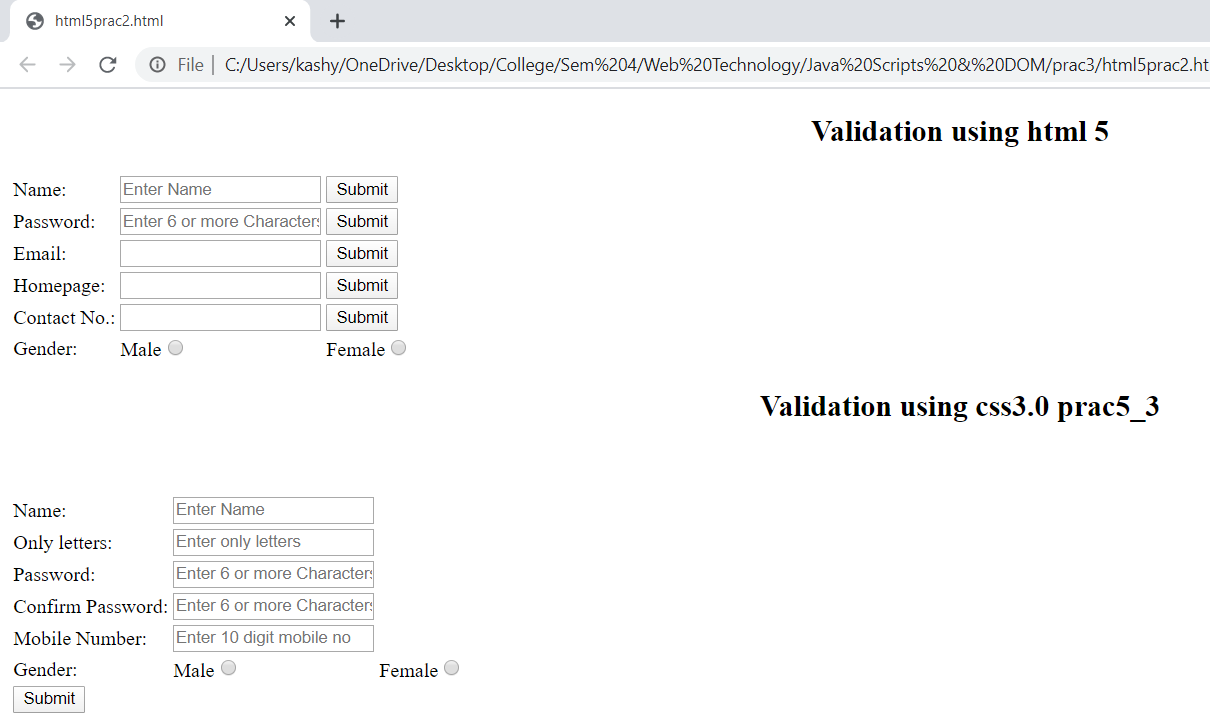
has selected or not.

g. Email validation.

h. Mobile number validation.

Design proper HTML form and perform all above task.

**OUTPUT:**



**CONCLUSION:**

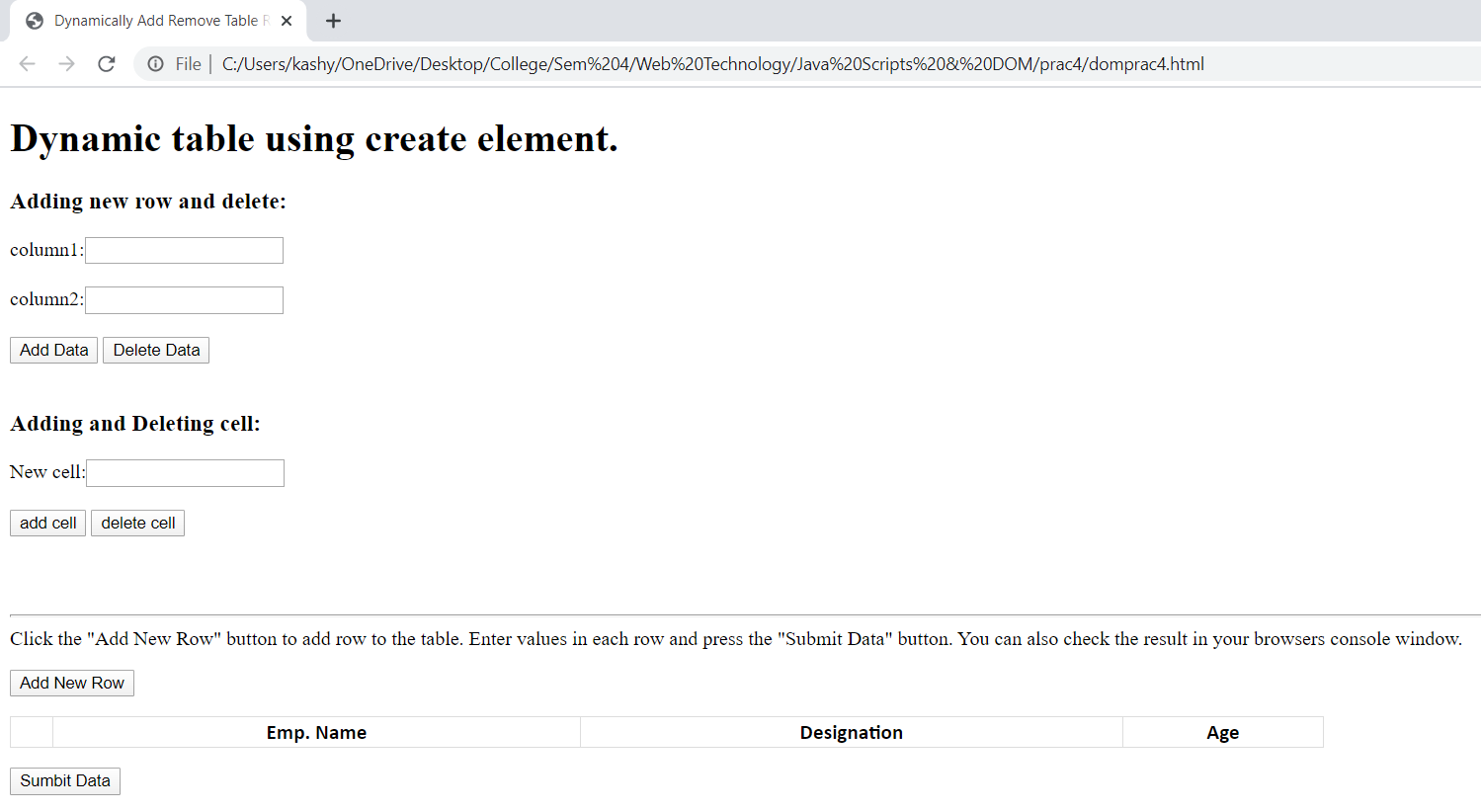
In this practical we learn how to validate form using JavaScript.

**PRACTICAL-4**

**AIM:** Create Dynamic table using DOM which perform following tasks:

* Add new Row and Cell
* Delete Row and Cell.

**OUTPUT:**



**CONCLUSION :**

In this practical we learn how to add and delete row and column using javascript.

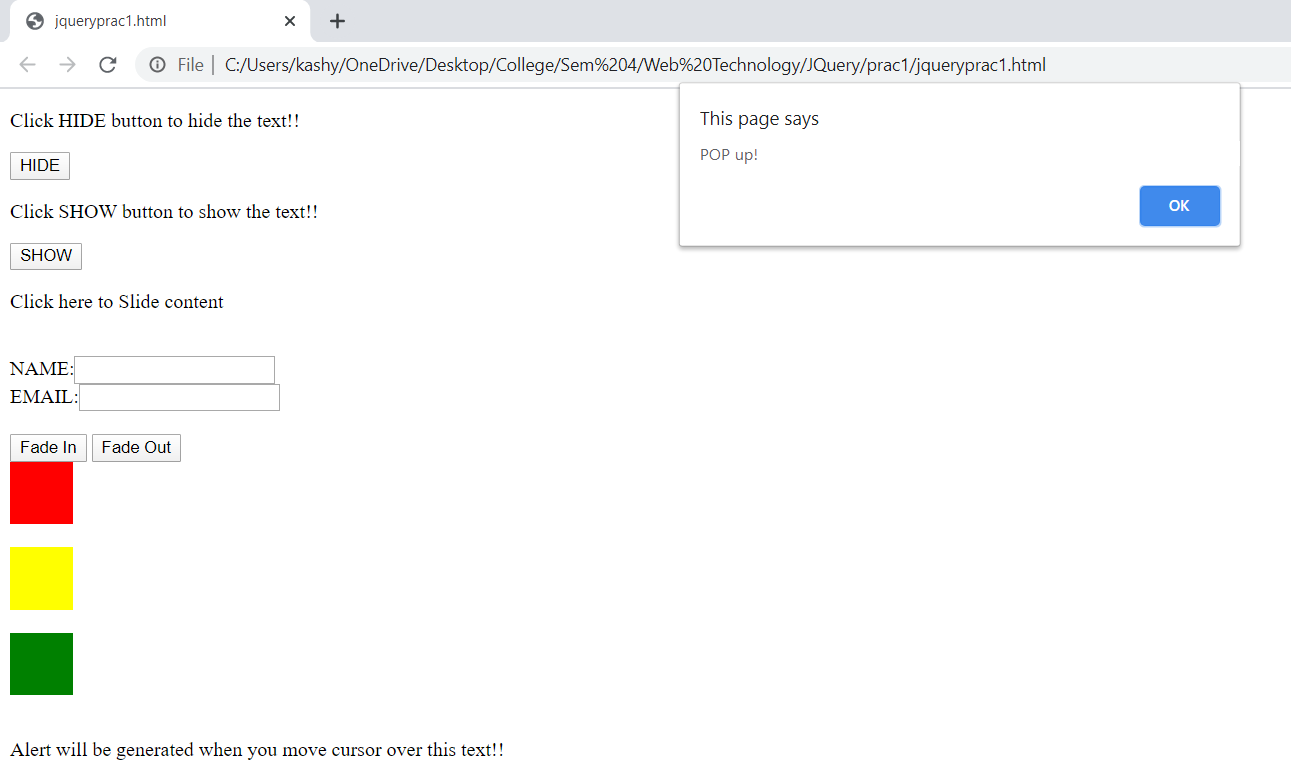
**Part–6 ( BASICJQUERY )**

**PRACTICAL-1**

**AIM:** Implement following tasks using jQuery:

* Sliding page elements
* Hiding and Showing elements
* Fading elements
* Toggling elements
* Stopping effects

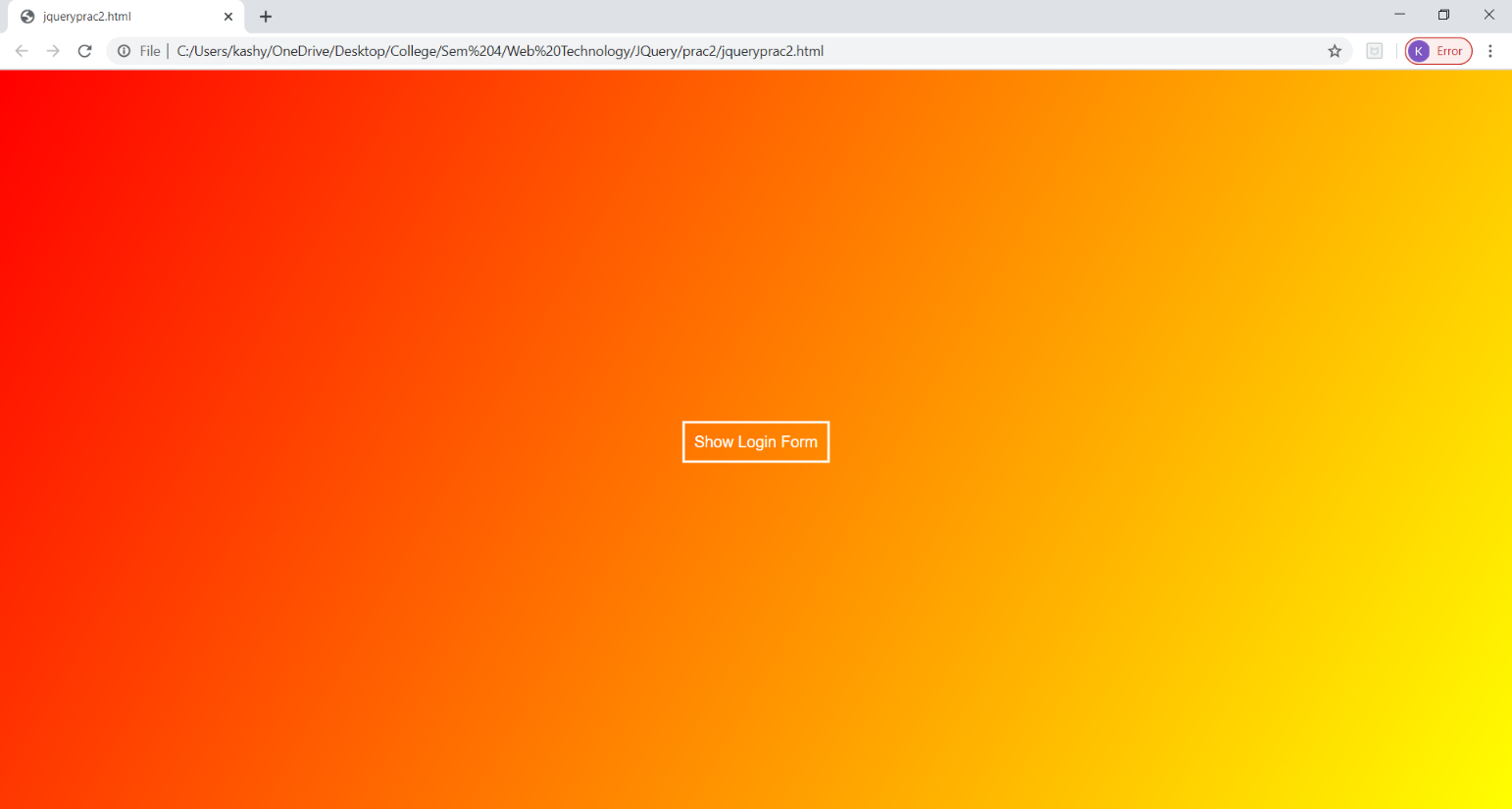
**OUTPUT:**

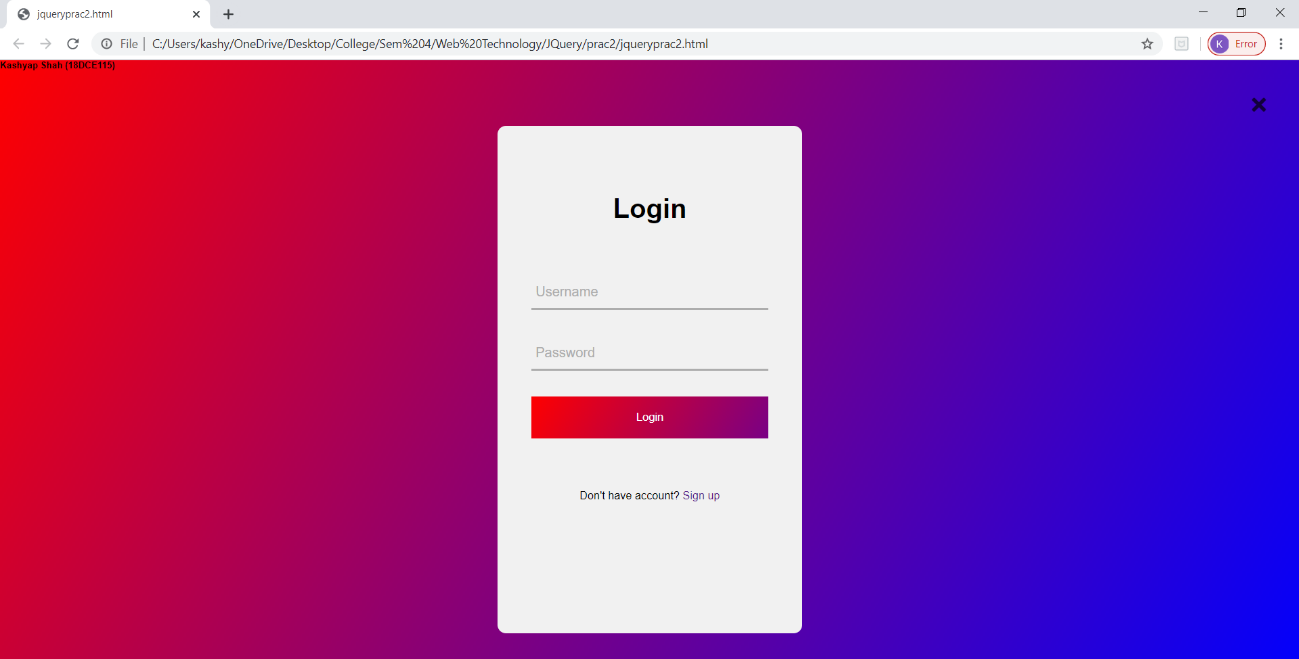


**PRACTICAL-2**

**AIM:** Create Animated Login form.

**OUTPUT:**





**PART–7 (BASIC) PHP**

**PRACTICAL - 1**

**AIM:** Installation and configuration of WAMP/XAMPP.

**STEPS:**

Step 1 : Install XAMPP on your machine.

How to inatall XAMPP on local machine

Step 2: Close the XAMPP.

Closing XAMPP to avoid collision between XAMPP and WAMP.

Step 3 : After installing XAMPP, install WAMP.

Install WAMP with default settings and options. You can install WAMP in any directory.

Step 4 : Now run WAMP for test.

Open your browser and write Localhost in url. You will find below screen.

Step 5 : Following steps for avoiding collision between XAMPP and WAMP.

This is a magic trick to run xampp and wamp on your local computer or server in network. It’s just changing your PORT for the WAMP apache server and mysql.

Step 6 : Changing PORT for WAMP apache server and mysql

By default apache listen on port 80 for the browsers and mysql listen on port 3306. Now, you need to change PORT no for apache and mysql for WAMPP. It’s need to do small change in http.conf file.

Step 7 : Changing PORT for WAMP apache server

Change in apache port to open config file. Basically location “C:\wamp\bin\apache\apache\*.\*.\*\*\conf” where stars are basically the version number.

open config file in notepad OR any editor with administrative permission.

Now, within this file search “Listen 80” and change it with port no “Listen 8081“. We are just changing server listen port address. Now it will listen on port number 8081 instead of 80 port.After changing server port listen address. Now search “ServerName localhost:80” within this file. It need to change to “ServerName localhost:8081“.

Step 8 : Changing PORT for mysql

You must need to change port no of mysql for WAMP because XAMPP using specific mysql and your basic URL “localhost/phpmyadmin/” same for both WAMP and XAMPP. So, you need to differenciate both mysql.

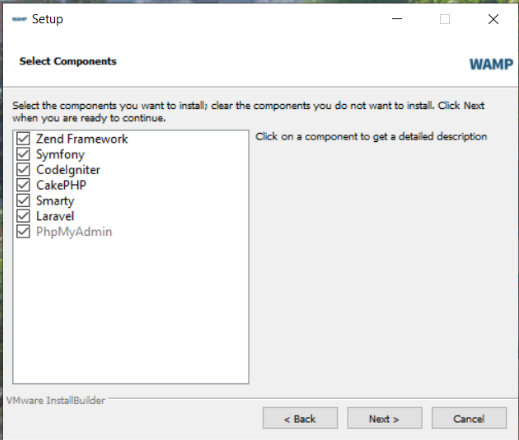
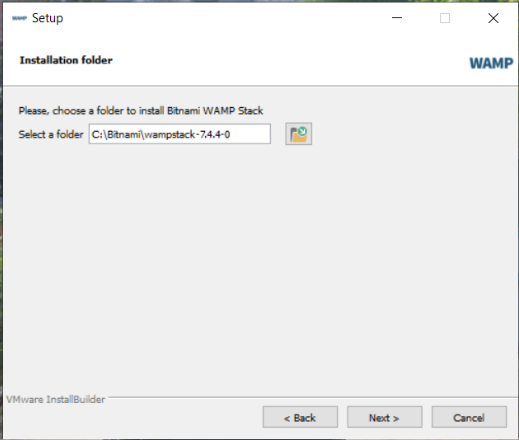
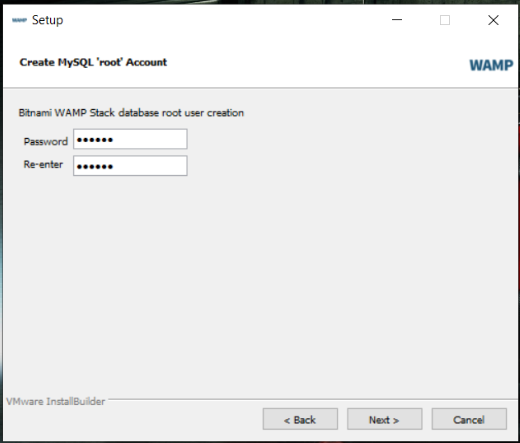
for changing mysql port open the file location which is like this “C:\wamp\bin\mysql\mysql5.5.24“. open the file “my.ini” and find the line port = 3306 and replace it with port = 3307 this will enable your wampmysql to access along with xamppmysql due to different port address.

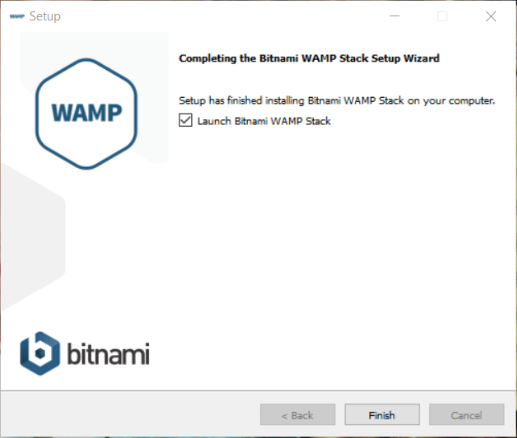
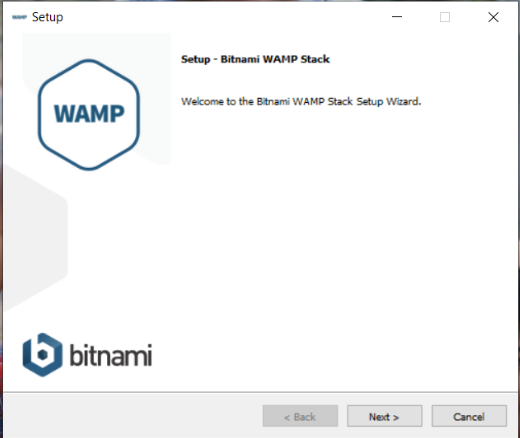
Step 9 : Accessing the wampmysql from wampphpmyadmin interface.

whenever we start phpmyadmin, we need to setup which server start first WAMP OR XAMPP. Overcome to this problem we need to make a small change in phpmyadmin interface code.

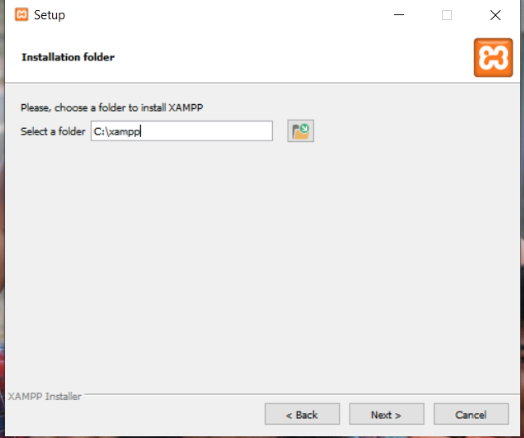
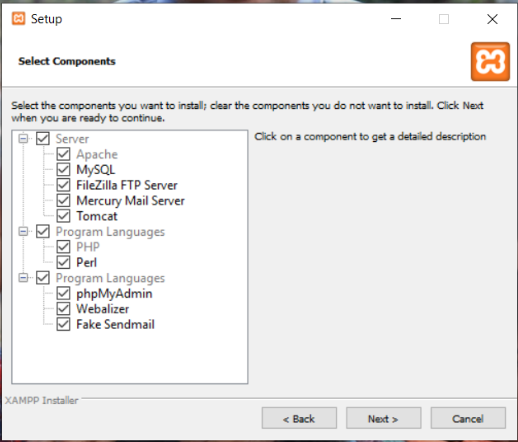
**IMPLEMENTATION:**

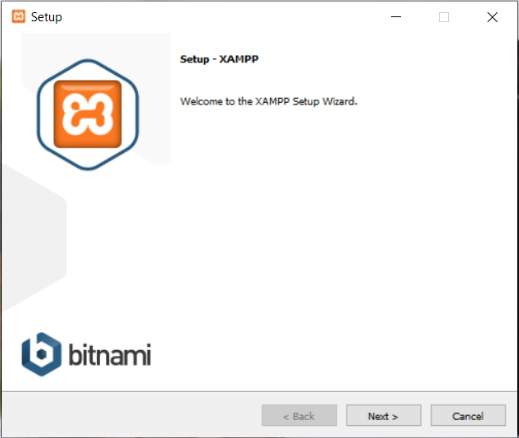
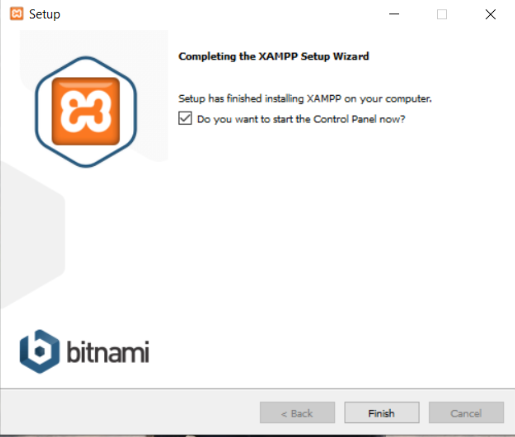
**WAMPP:**

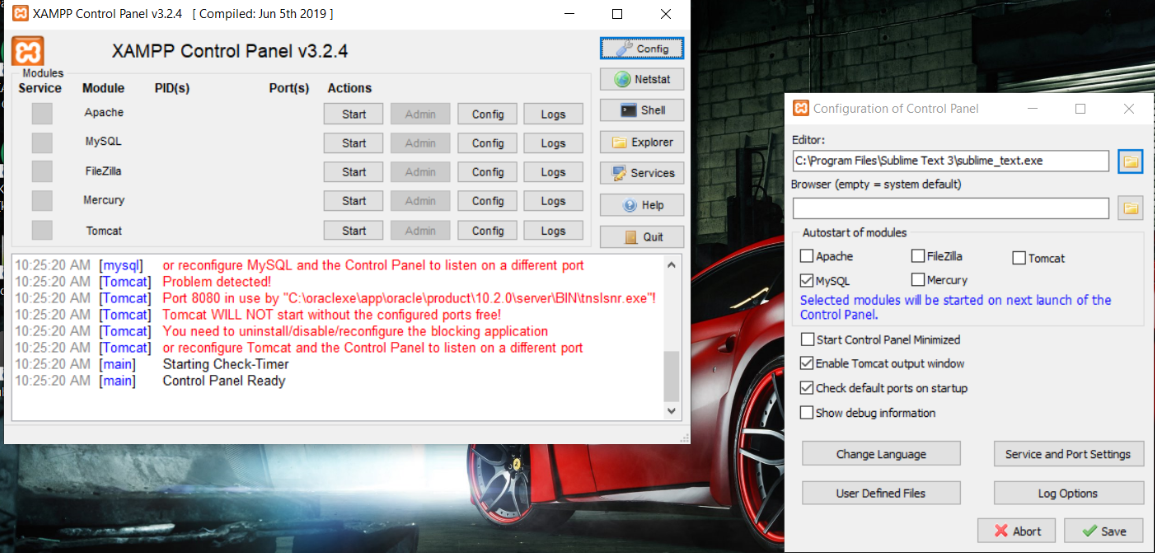




**XAMPP:**



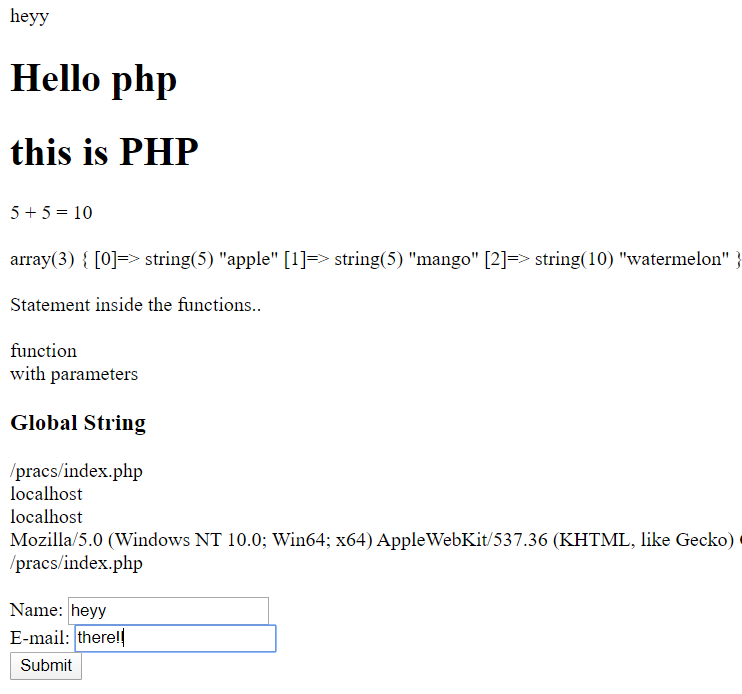
**CONCLUSION:**

In this practical we learn how to install and configure WAMP/XAMPP server.

**PRACTICAL – 2**

**AIM:** Study and demonstrate php syntax, data type, variable, function, array, super global variable and form.

**OUTPUT:**





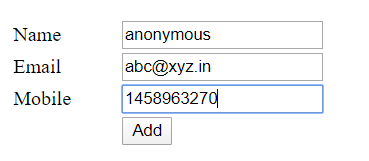
**CONCLUSION:**

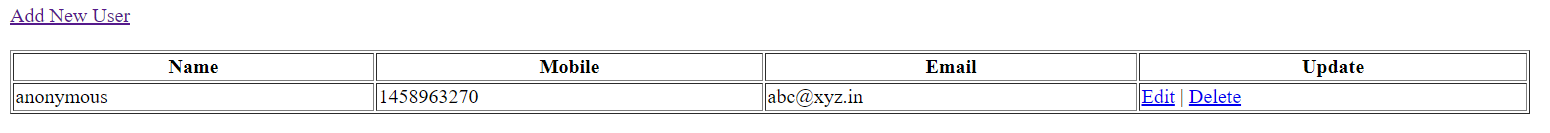
In this practical we learnt about basic syntax, data type, variable, function, array, super global variable and form of PHP.

**PRACTICAL – 3**

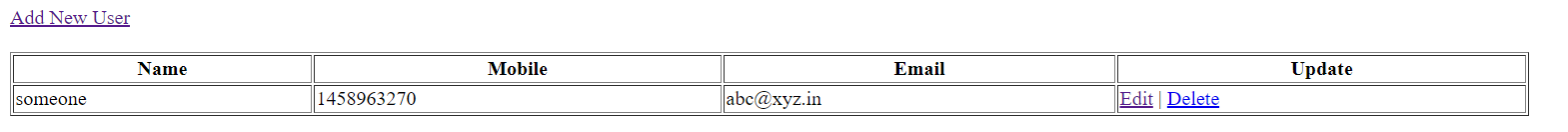
**AIM:** Study and demonstrate mysqli connection and CRUID operations with php.

**OUTPUT:**









**CONCLUSION:**

In this practical we learnt about MySQL connection and CRUID operations with php.