INDEX

11		CREATING A NODEJS SERVER USING EXPRESS		
12		CREATING MONGODB NODEJS CONNECTION USING EXSPRESS		
13		CALCULATOR EXCERSICE IN NODEJS USING EXPRESS		
14		COUNTER USING REACTJS		
15		TODO APPLICATION USING REACTJS		
16		HANDLING COOKIES WITH NODEJS AND EXPRESS FRAMEWORK		

1.TEXT FORMATTING

Aim:

To write a HTML document using Text Formatting Tags.

Procedure:

1. Start the program.
2. Open Notepad application
3. Type the source code in notepad and save with the extension .html
4. Open the html document in the browser and verify

Algorithm:

Step1:Start

Step2: <p> tag used for paragraph.

Step 3: <i> used for italic.

Step4:<b>used for bold.

Step 5: <u> used for under line

Step 6: used for division.

Step7: pre>used for pre formatted text.

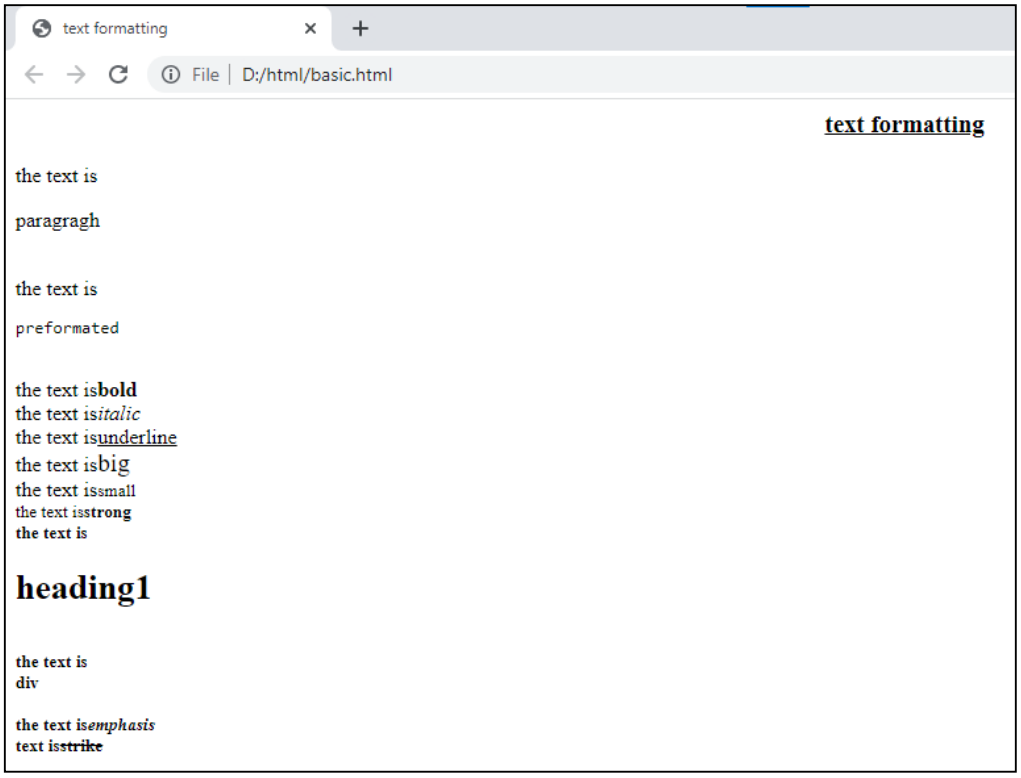
Step8: Stop.

SOURCECODE:-

```
<html>
<head>
<title>TextFormatting</title>
</head>
<body>
<h3align="center"><b><u>Text Formatting</u></b></h3>
The Text is<p>paragraph</p><br>
The Text is<pre>pre formatted</pre><br>The
Text is<b>bold</b><br>
The Text is<i>italic</i><br>
The Text is
<u>underline</u><br>The Text
is<big>big</big><br>

The Text is  <small>small</small><br>TheTextis<strong>strong</strong><br>TheTextis
<h1>heading1</h1><br>
TheTextis<div>div</div><br>
```

```
The Text is
<em>emphasis</em><br>TheTextis<strike>
strike</strike><br>
</body>
</html>
```

OUTPUT:-



RESULT:

A HTML document using text formatting tags was created and output was verified

2.CREATING LISTS

Aim:

To write a html documents which illustrates different types of list.

Algorithm:

Step 1: Start.

Step 2: Create a document about list.html.

Step 3: By using <ol> tags makes an ordered list. Step

4:Byusing<ul>tags makes a nun ordered list Step

5:Byusing<dl>tags makes an definition list.

Step 6: <li> tag used to declare the list of item option such as number (or) alphabet (or) some special symbol like disk, circle and square by using the “type” attribute.

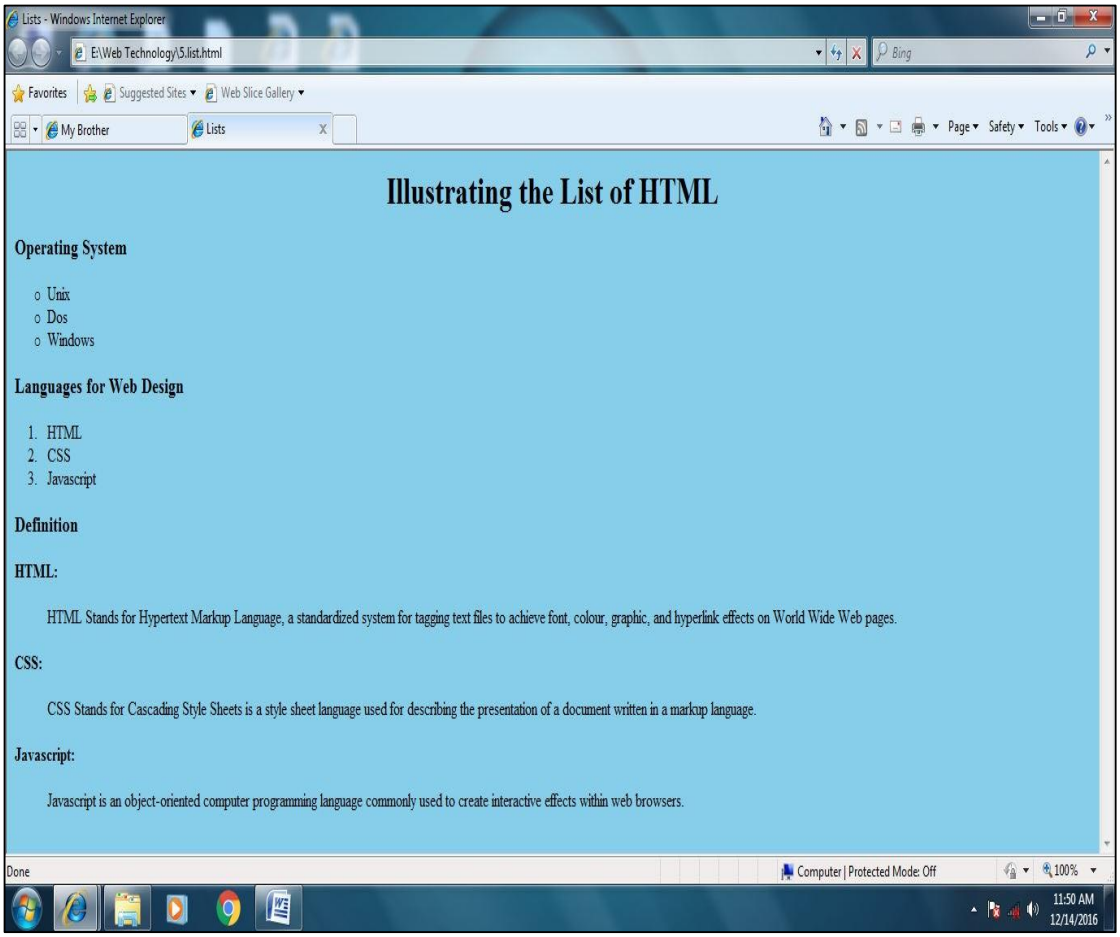
Step 7:<dt>tag is for define term and <dd>tagisfordefinitiondata.Step8:Stop.

SOURCECODE:

```
<html>
<head>
<title>Lists</title>
</head>
<bodybgcolor="skyblue">
<h1align="center">Illustratingthe ListofHTML</h1>
<h3>OperatingSystem</h3>
<ultype="circle">
<li>Unix</li>
<li>Dos</li>
<li>Windows</li>
</ul>
<h3>LanguagesforWebDesign</h3>
<oltype="1">


<li>HTML</li>
<li>CSS</li>
<li>Javascript</li>
</ol>
<h3>Definition</h3>
<dl>
<dt><h4>HTML:</h4></dt>
<dd>HTML Stands for Hypertext Markup Language, a standardized system for tagging text files to
```

```
achieve font, colour, graphic, and hyperlink effects on World Wide Webpages.</dd>
<dt><h4>CSS:</h4></dt>
<dd>CSS Stands for Cascading Style Sheets is a style sheet language used for describing the
presentation of a document written in a markup language.</dd>
<dt><h4>Javascript:</h4></dt>
<dd>Javascript is an object-oriented computer programming language commonly used  create
interactive effects within web browsers.</dd>
</dl>
</body>
</html>
```

OUTPUT:-



RESULT:

A HTML Document showing different types of lists was created and output was verified.

3.USING IMAGES IN HTML DOCUMENT

Aim:

To create HTML document for image alignment

Algorithm

Step1: Start

Step2: Createflower.html. In this file , we are using<p>paragraph tag and<img>tag.

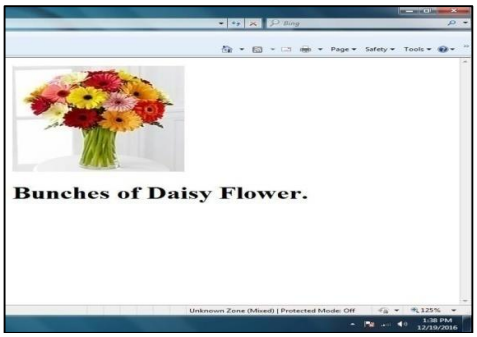
Step3: After creating sub files, create the main document. It has hyperlinks as the sub files.

Step4:Stop.

SOURCECODE:-

```
<html>
<title>Flower</title>
<body>
<imgsrc="C:\Users\Public\Pictures\Sample Pictures\daisyflower.jpg"
height="200"width="200">
<h1>BunchesofDaisyFlower.</h1>
</body>
</html>
```

OUTPUT:-



RESULT:

A HTML Document was created to display image and output was verified.

4.CREATING TABLES IN HTML DOCUMENT

Aim:

To writea HTML program for student details using table.

Procedure:

- ✓ In this program, we are going to develop the student details.
- ✓ We use the techniques for this program to create table like< table order=”value”>
- ✓ Here we can use the logic for total is equal to m1+m2+m3 (i.e. total =m1+m2+m3).

Algorithm:

Step1:Start.
Step2:To create table.

Step3:To declare the heading for table like sno ,sname, course,marks(m1,m2,m3)andtotal.

Step

4:Enterthevaluesintothetable.Step5:Heremar

km1, m2andm3.

Step6:Heretotal=m1+m2+m3.Step

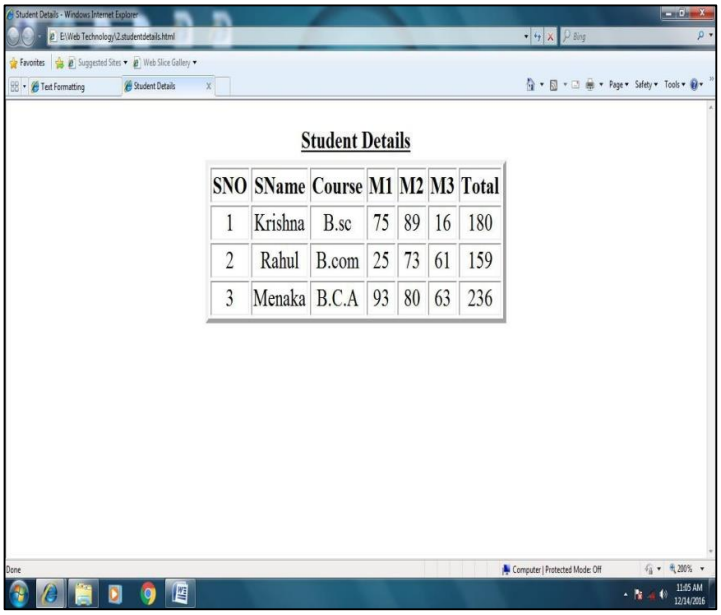
7:Stop

SOURCECODE:-

```
<html>
<head>
<title>StudentDetails</title>
</head>
<body>
<tableborder="3"align="center">
<caption><b><u>StudentDetails</u></b></caption>
<tralign="center">
<th>SNO</th>
<th>SName</th>
<th>Course</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
```

```
<th>Total</th>
</tr>
<tralign="center">
<td>1</td>
<td>Krishna</td>
<td>B.sc</td>
<td>75</td>
<td>89</td>
<td>16</td>
<td>180</td>
</tr>
<tralign="center">
<td>2</td>
<td>Rahul</td>
<td>B.com</td>
<td>25</td>
<td>73</td>

<td>61</td>
<td>159</td>
</tr>
<tralign="center">
<td>3</td>

<td>Menaka</td>
<td>B.C.A</td>
<td>93</td>
<td>80</td>
<td>63</td>
<td>236</td>
</tr>
</table>
</body>
</html>
```

OUTPUT:



Student Details

SNO	SName	Course	M1	M2	M3	Total
1	Krishna	B.sc	75	89	16	180
2	Rahul	B.com	25	73	61	159
3	Menaka	B.C.A	93	80	63	236

RESULT:

A HTML Document was created to display a table and output was verified.

5.DESIGNING A FORMS IN HTML DOCUMENT

Aim:

To write a HTML program for HTML forms.

procedure:

Algorithm:

Step1:Start.

Step2: Create a document to demonstrate HTML form.

Step 3: Use <input type=”text”> tag make a name textbox.
Step4: Use <input type=”email”>tag make an email field.

Step 5: Use <input type=”number”> tag get a contact number

Step6: Use <input type=”radio”>tag get a gender field.

Step7: Use<select>tag to create a drop down list and <option>tag to list the items of

the list

Step8: Use<textarea>tag to create a comment box.

Step 9: Use <script> tag to display an alert when the submit button is clicked

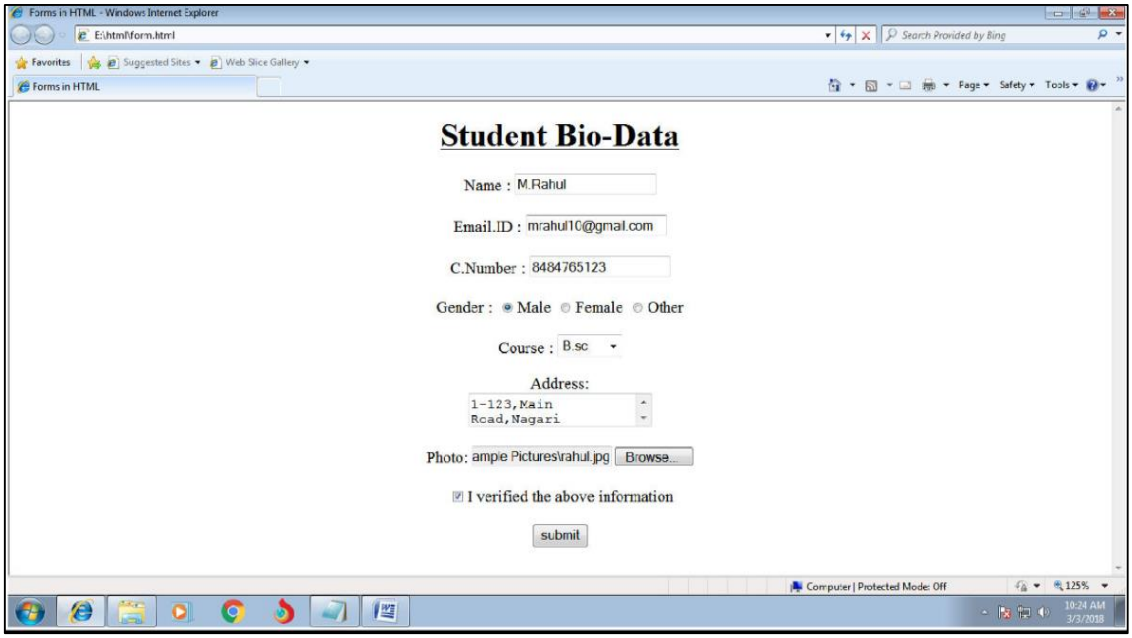
Step10: Stop.

SOURCECODE:-

```
<html>
<head>
<title>FormsinHTML</title>
</head>
<body>
<h1><u><center>StudentBio-Data</center></u></h1>
<center>
<form>
Name : <input type ="text" name="myname"><br><br>Email.ID :
<input type="email"
name="emailid"><br><br>C.Number:<inputtype="number"name="n
umber"><br><br>Gender:  <input type="radio"name="gender">Male
            <input type="radio" name="gender">Female
            <input type="radio"
name="gender">Other<br><br>Course:<select>
            <option>B.sc</option>
```

```
                <option>B.com</option>
                <option>B.C.A</option>
                </select><br><br>Ad
dress:<br>
<textarea>Enter
Here...</textarea><br><br>Photo:<input type="file"
name="file"><br><br>
<inputtype="checkbox"name="chkb1">Iverifiedtheaboveinformation<br><br><inputtype="submit"
name="button"onclick="submitted()"value="submit">

<script
type="text/javascript">function
submitted()
{
 alert("Your application is submitted");
}
</script>

</form>
</center>
</body>
</html>
```

OUTPUT:-



RESULT:
A HTML Document with form was created and output was verified.

6 .INTERNAL STYLE SHEET

Aim:

To write a HTML program with CSS using Internal Style Sheet.

Algorithm:

Step 1: Start.

Step 2: Declare style in <head> tag.

Step 3: Create ISS class with font-family: verdana and font-color: Green

Step4:Setcolor:red for the text which use<h2>tag.

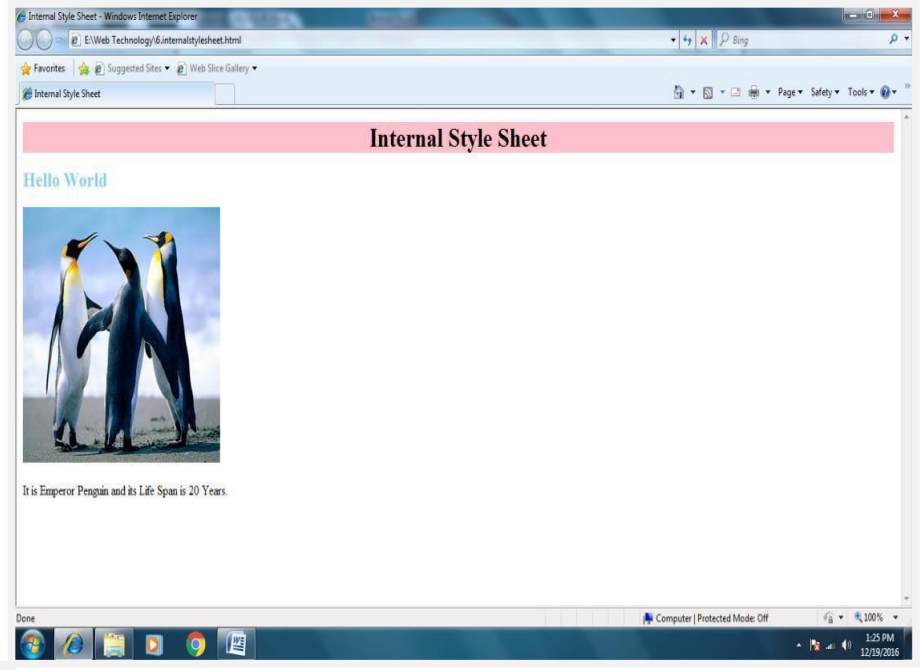
Step5:Applystyleinside

the<body>tag.Step6:Stop.

SOURCECODE:-

```
<html>
<head>
<title>InternalStyleSheet</title>
<styletype="text/css">
h1.ISS
{
 text-align:center;
 font-family:Footlight
 MT;background-
 color:pink;
}

h2
{
 color:skyblue;
}
</style>


</head>
<body>
<h1class="ISS">InternalStyleSheet</h1>
<h2>HelloWorld</h2>
<imgsrc="C:\Users\Public\Pictures\Sample Pictures\Penguins.jpg" height="300"width="300">
<p>ItisEmperorPenguinandits LifeSpanis 20Years.</p>
</body>
</html>
```

OUTPUT:-



RESULT:

A HTML Document using Internal CSS was created and the output was verified

7.FORM VALIDATION USING JAVASCRIPT

AIM

Create a form and validate the contents of the form using JavaScript.

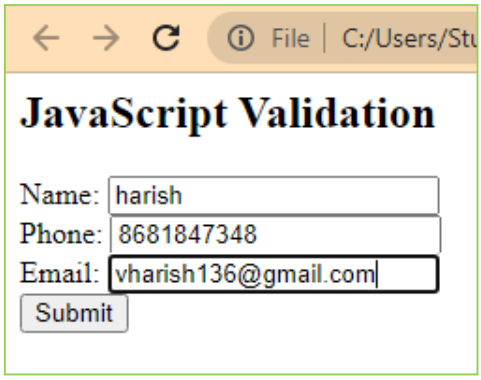
ALGORITHM

1. Start
2. Create a JavaScript function to validate input field
3. Pass values of all input fields into individual variables
4. Check if values of variables empty
5. Invoke an alert box if empty variables occur
6. Stop

CODE

```
<html>
<head>
<script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  let y = document.forms["myForm"]["phone"].value;
  let z = document.forms["myForm"]["email"].value;
  if (x == "" || y=="" || z==="") {
    alert("fill all fields");
  }
}
</script>
</head>
<body>
<h2>JavaScript Validation</h2>
<form name="myForm" action="#" onsubmit="return validateForm()" method="post">
 Name: <input type="text" name="fname"><br>
 Phone: <input type="text" name="phone"><br>
 Email: <input type="email" name="email"><br>
 <input type="submit" value="Submit">
</form>
</body>
</html>
```

OUTPUT



RESULT:
A HTML Document with form validation was created and the output verified

8.DISPLAY CONTENT IN FORM OF CARD

Aim

Get Data And Display The Contents In The Form Of A Card.

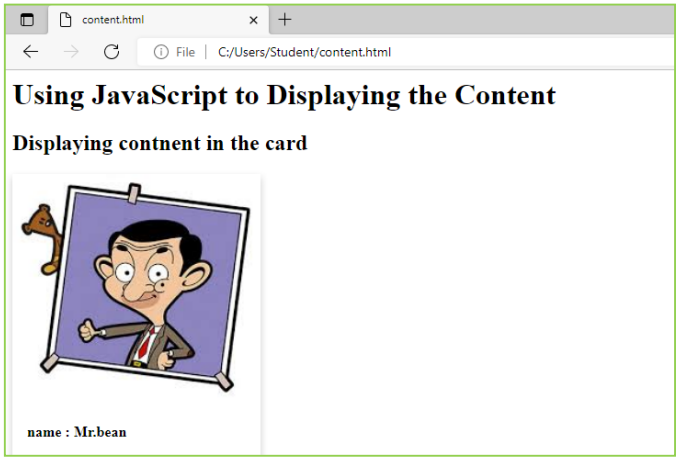
Algorithm

1. Start
2. Create a html document to display a card
3. Use CSS To Display Card.
4. Stop

Source Code

```
<html>
<head>
<style>
.card {
  box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
  transition: 0.3s;
  width: 20%;
}
.card:hover {
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
}
.container {
  padding: 2px 16px;
}
</style>
</head>
<body>
<h1> Using JavaScript to Displaying the Content</h1>
<h2>Displaying contnent in  the card</h2>
<div class="card">
  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT_257oB3YKc781W7RfRva-
1ESZc9BgR2WO-A&usqp=CAU" alt="Avatar" style="width:100%">
  <div class="container">
<b><p id =id:10 >name : Mr.bean </p></b>
  </div>
</div>
</body>
</html>
```

Output



Result

A HTML Document was created to display a card and output was verified.

9.USING FETCH API TO DISPLAY THE CONTENT IN THE FORM OF A CARD

AIM

Get data using Fetch API from an open-source endpoint and display the contents in the form of a card.

ALGORITHM

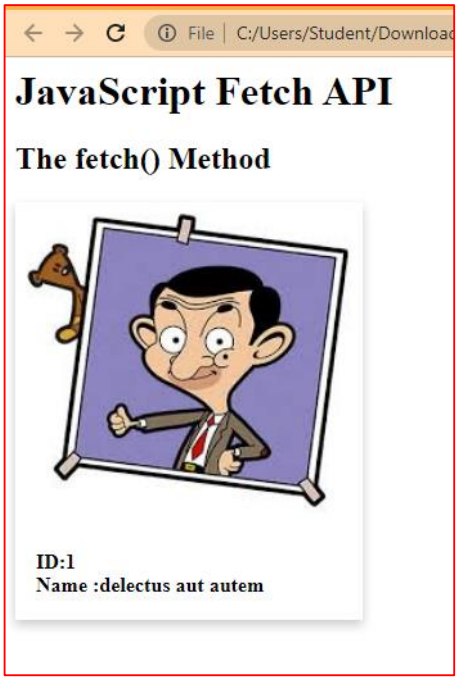
5. Start
6. Declare variable with API url
7. Use Fetch() function to fetch data from API by passing url variable
8. Convert JSON data using JSON. parse() function
9. Pass data to HTML using inner HTML() function
10. Use CSS to display card.
11. Stop

CODE

```
 <html>
<head>
<style>
.card {
  box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
  transition: 0.3s;
  width: 20%;
}
.card:hover {
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
}
.container {
  padding: 2px 16px;
}
</style>
</head>
<body>
<h1>JavaScript Fetch API</h1>
<h2>The fetch() Method</h2>
<div class="card">
  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT_257oB3YKc781W7RfRva-
1ESZc9BgR2WO-A&usqp=CAU" alt="Avatar" style="width:100%">
  <div class="container">
<b><p id="demo">Fetch a file to change this text.</p></b>
  </div>
</div>
<script>
let file = "https://jsonplaceholder.typicode.com/todos/1"
fetch (file)
.then(x => x.text())
.then(data => {
var id = JSON.parse(data).id;
```

```
var title = JSON.parse(data).title;
document.getElementById('demo').innerHTML ='ID:'+ id +'<br>' + 'Name :' + title;
});
</script>
</body>
</html>
```

Output



RESULT:

A HTML Document was created to display data using fetch API from an open source in a card and the output as verified.

10.CREATING A NODEJS SERVER WITHOUT USING EXPRESS

Aim

To create a Nodejs Server that servers Static Html And CSS Files to the user without using Express

Procedure:

Use VSCode to develop the appplication

Prerequisite : NodeJS Installled

Algorithm

Step1:Create A Html File

Step2:Save The File As Demofile1.Html

Step3:Open Notepad Editor To Write A Js Code

Step4:Save It As Node Filename.Js

Step5:Open The Browser For The Output

Program

demofile1.html

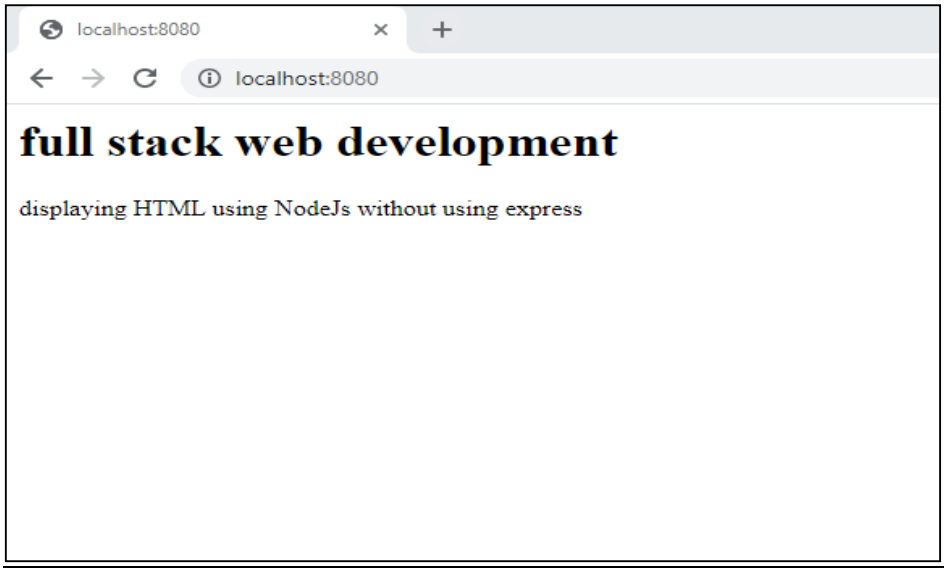
```
<html>
<body>
<h1>full stack development lab excerise</h1>
<p>displaying html file using nodejs without using express</p>
</body>
</html>
```

filename.js

```
var http = require('http')
var fs = require('fs')
http.createServer(function(req,res){
fs.readFile('demofile1.html',function(err,data){
res.writeHead(200,{'Content-Type':'text/html'});
```

```
res.write(data);

return res.end();

});

}).listen(8080);
```

Output



RESULT:
Therefore the program executed successfully

11.CREATING A NODEJS SERVER USING EXPRESS

Aim

Create Nodejs Application With Express

Algorithm

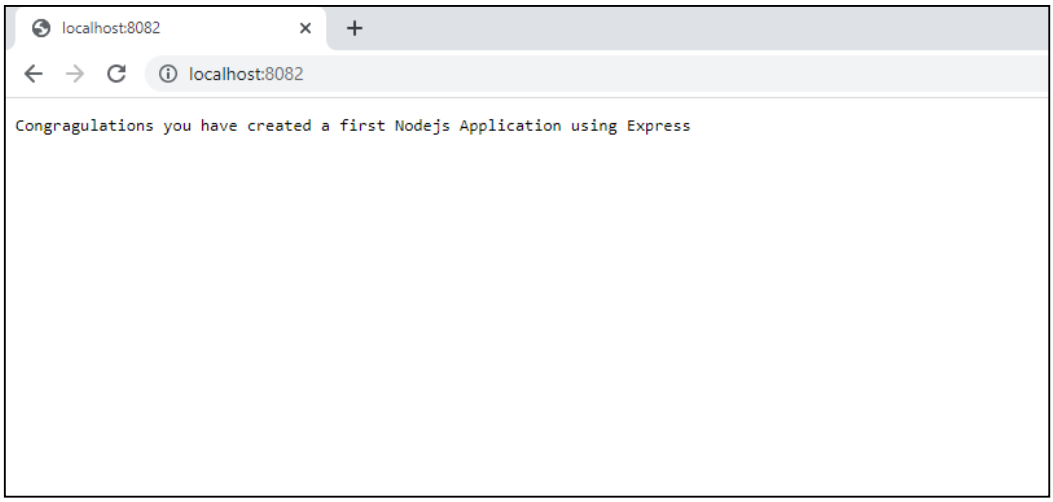
Step1:Open Notepad Editor To Write A Js Code
Step2: Save It As Node Name.Js
Step3:Open A Command Prompt And Change The Directory To Where You File Saved And Type Node
Filename.Js

Step4:Print The Result

Program

```
var http=require("http");
http.createServer(function(req,res){
res.writeHead(200,{'Content-Type':'text/plain'});
res.end("Congragulations you have created a first Nodejs Application using Express");
}).listen(8082);
```

Output



RESULT:

A nodejs server using express was created and the output verified

12.CREATING MONGODB NODEJS CONNECTION USING EXSPRESS

Aim

To Create A Mongodb Nodejs Connection Using Express

Algorithm

Step 1: Create Your Own Folder(Name It As Mongo)

Step 2: Open VS Code And Select Your Folder(Mongo) ThatYou Have Created, From **File->Open Folder**.

Step 3: Then Select The **Terminal->New Terminal** And Type TheFollowing Command >**Npx Create-React-App Frontend**.

Step 4: Add The Extension **ES7+** In VS Code.

Step 5: Then Go To Terminal Again And Type The FollowingCommands:

>Mkdir Backend

>Cd Backend

>Npm Init

=>Then Click On **Ente**r **Button** Repeatedly To Add **Package.Json** Automatically.

Step 6: After That Type The Command >**Npm I Express** ToInstall Express.

Step 7: Then Type >**Npm I Cors** To Determine What Type OfError Has Occurred.

Step 8: Then Right Click On **Backend** (Folder), Which Is At The Left Side, Click Open **Newfile** And Name It As **App.Js**

In app.js copy the following program

```
const express = require('express');const cors =

require('cors');

const app = express();
```

```
var bodyParser = require('body-parser');

var MongoClient = require('mongodb').MongoClient;

var url="mongodb://localhost:27017/?appName=demo&read

Preference=primary";


//in the above line you will change your mongodb url

app.use(cors());

app.use(bodyParser.json());

app.post('/address', (req, res) => {

    MongoClient.connect(url, function(err, db) {

        if (err) throw err;

        var dbo = db.db("user1");

        console.log("re",req.body)

        var myobj = { name: req.body.name, address:
req.body.address };// db value name.

        dbo.collection("login").insertOne(myobj, function(err, result) {

          if (err) throw err;

          console.log("1 document inserted");

          db.close();

          if (err) {

            //      throw err;

            return res.status(400).json({

                status: false,
```

```
                description: "400 bad request"
            })
        }
        else {
            return res.status(200).json({status: true,
                description: result
            })
        }
        });
      })
  })
  app.listen(8090, () => console.log('Assessment1 api runs onhttp://localhost:8090/'));
```

Step 9: Now write the following code in the **backend** and wehave to change the **URL** of **MongoDB** in **backend**.

Step 10: In your explorer(i.e, left side menus) select **frontend**folder and select **src** folder. then you can see **App.js**,select it. Copy the following program in App.js

```
import React from 'react'
import { useState } from 'react'
  function App() {
    const [name,setname]=useState("")
    const [password,setpassword]=useState("")
    function Handle(){
      console.log("data",name,password)  ;
      var myHeaders = new Headers();
      myHeaders.append("Content-Type", "application/json");
      var raw = JSON.stringify({
            name:name,
```

```
                address:password

        });

        var requestOptions = {

            method: 'POST',

            headers: myHeaders,

            body: raw,

            redirect: 'follow'

        };

                                fetch("http://localhost:8090/address", requestOptions)

                .then(response => response.json())

                .then(result =>
                {
console.log("res", result);

                    if (result.status === true) {

                        alert("register succefully!")

                      }

                    else {

                        alert("your are not register sucess")

                    }

                })

                .catch(function(error){

                    console.log("siginerror",error)

                })

      }

      return (

        <div className='App'>

          <input placeholder='email' onChange={(e)=>setname(e.target.value)}  />

          <input placeholder='password' onChange={(e)=>setpassword(e.target.value)}/>
```

```
      <button onClick={Handle}>submit</button>

    </div>

  )
export default App
```

Step 11: To change your **MongoDB URL**, go to **MongoDB compass**(if it is not available , then install it fromOnline).

Step 12: In MongoDB , click on **New Connection** -> **Advancedconnection** option.

Step 13: Then select **Advanced tab** and then click **Readpreference->Primary**.

Step 14: On **URI** options, set **Key**=appname and **Value**=DEMO
=>After that go up and **Copy the URL,**which is to bereplaced in the **backend MongoDB**.

Step 15: Click Save and Continue.Then enter the name **DEMO**

Step 16: On the left bottom , there is **+ Create databasebutton**. Click it and enter the following ,

Database Name: user
Login Name: login

Step 17: Now go back to **VS Code** and change the **URL** in **backend**.

Step 18: Type the following command in the existing **Terminal**,

>**cd backend**
>**node app.js**

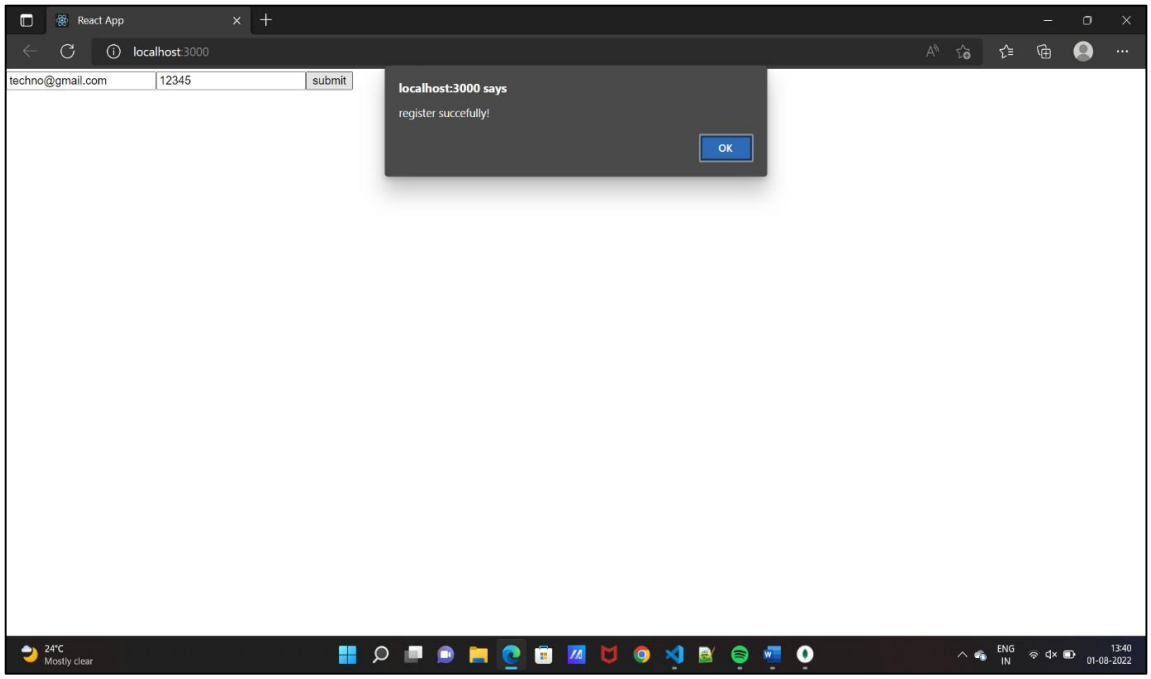
Step 19: Open another **terminal** by clicking **+** on the **bottomright** of the **existing terminal**.

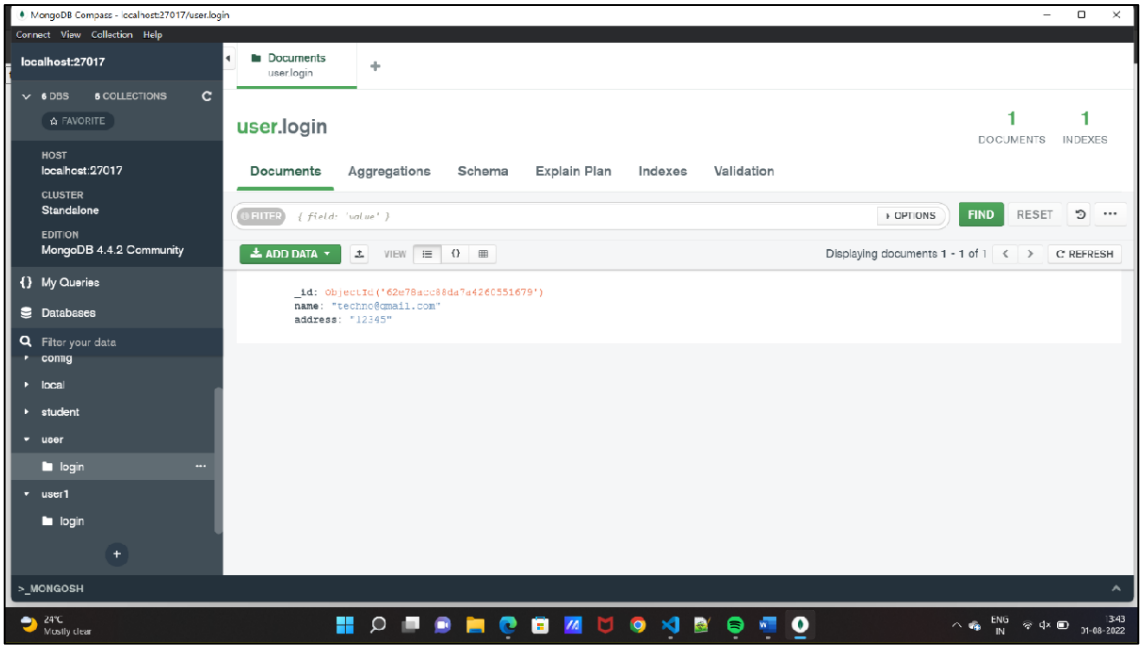
Step 20: In the **new Terminal** , type the following commands

cd frontend

npm start

OUTPUT:





RESULT:

An application showing Mongodb nodejs connection using exspress was developed and the output verified

13.CALCULATOR EXCERSICE IN NODEJS USING EXPRESS

AIM:

Create A Calculator In Node JS.

Algorithm:

Step 1: Create A Folder (Ex: Expression.App) And Create A File Inside That Folder (Ex: Calculator.Js) And Write The Code For Addition, Subtraction And Multiplication.

Step 2: Next Create Another File (Ex: App.Js) Which Uses This Calculator Module And Calls All The Functions Of The Calculator Module.

Step 3: Run The App.Js At The VS Code Terminal And Observe The Result Node App.Js.

Source Code:

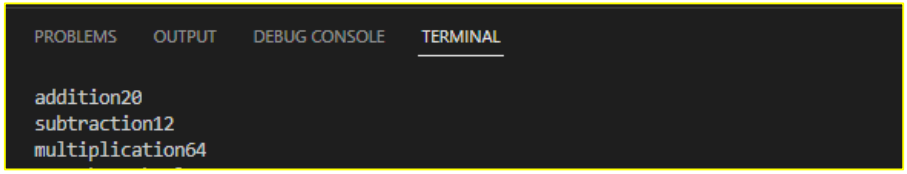
File Name : Calculator.js

```
exports.add = function(a,b){
   return a+b
}
exports.sub = function(a,b){
   return a-b
}
exports.mul = function(a,b){
   return a*b
}
```

File Name : App.js

```
var calculator = require('./calculator')
var a = 16
var b = 4
console.log("addition" + calculator.add(a,b))
console.log("subtraction" + calculator.sub(a,b))
console.log("multiplication" + calculator.mul(a,b))
```

Output



RESULT

Nodejs program to create a calculator has been successfully verified and was executed.

14.COUNTER USING REACTJS

Aim:

To Create a counter program using ReactJS

Algorithm:

Step 1: Install VS code
Step 2: Create new ReactJS program inside the folder ReactProjects in terminal using
npm create-react-app-counter
Step 3: Replace the following program inside into the App.js file

```
import './App.css';
import React from 'react';
import { useState } from 'react';
 function App() {

  const[counter,setcounter] = useState(0);
  const increase = ()=>{
    setcounter(count => count+1);
  };
  const decrease = () =>{
    setcounter(count => count-1);
  }
  const reset = () =>{
    setcounter(0)
  }

  return (

    <div className='counter'>
      <h1>Counter Using REACT</h1>
      <span className="output"><h1>{counter}</h1></span><br></br>
      <button className='ctrl__btn' onClick={increase}>+</button>
      <button className='ctrl__btn' onClick={decrease}>-</button>
      <button className='ctrl__btn' onClick={reset}>reset</button>
    </div>
  )
 }

 export default App;
```

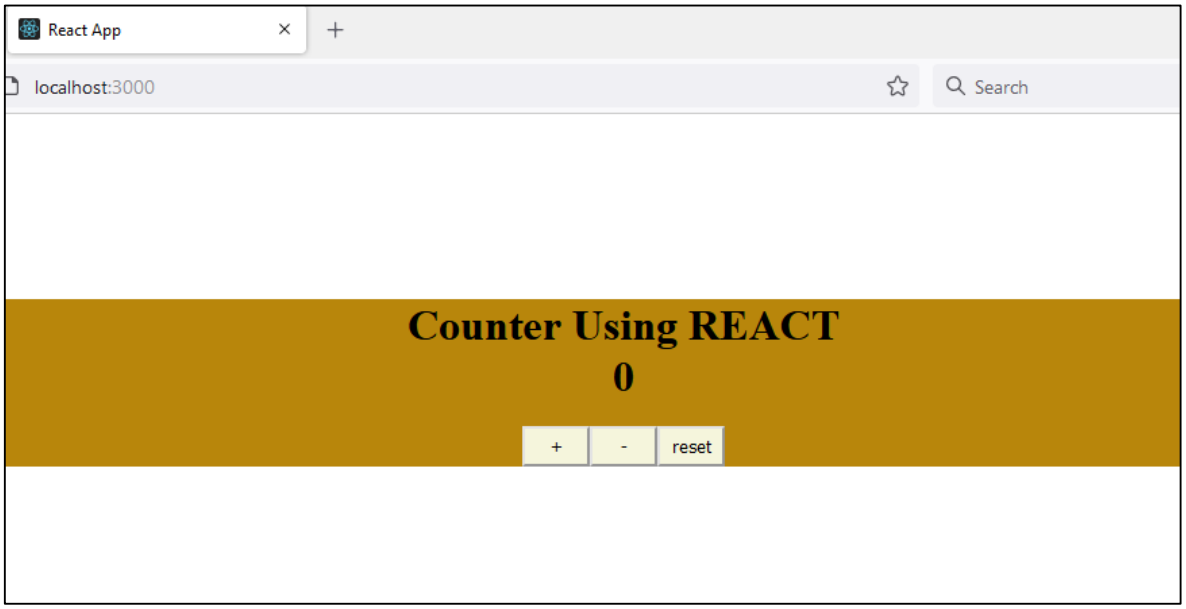
Step 4: Replace the following program inside into the App.css file

```
.counter{
    text-align: center;
    background-color: darkgoldenrod;
    margin-top: 18%;
}
.ctrl__btn{
    width: 50px;
    height: 30px;
    background-color: beige;
}
```

Step 5: Use the following command in terminal to run the program

>npm start

Output :



Result

Creating Counter Using React Has Been Executed Successfully And The Output Is Verified.

15.TODO APPLICATION USING REACTJS

Aim

To Create A Program Of Todo Application Using The Reactjs.

Algorithm

Step 1- Start The Program
Step 2 – Create An Folder With Anyname
Step 3 – Open Vs Code To Write The Program
Step 4 – Create An Todo Application Using Reactjs
Step 5 – Create Functions With The Todo List
Step 6 – Now Create An File With The Todo.Js
Step 7 – Run The Code
Step 8 – Getting The Output In The Browser
Step 9 – Stop The Program.

Source code

APP.JS

```
import TodoList from './TodoList';
import {useState,React,useRef,useEffect} from 'react';
import { v4 as uuidv4 } from 'uuid';
function App() {
  const [todos,setTodos] = useState([]);
  consttodoNameRef = useRef();
  function addTodo(event){

    constnametodo = todoNameRef.current.value;
    setTodos(prevTodos=>{
      return [...prevTodos,{id:uuidv4(),name:nametodo,completed:false}]});

  }
  const LOCAL_STORAGE_KEY = 'todoApp.todos';

useEffect(()=>{
  conststoredTodos = JSON.parse(localStorage.getItem(LOCAL_STORAGE_KEY))
  if(storedTodos)
  setTodos(storedTodos)
},[])
  useEffect(()=>{
    localStorage.setItem(LOCAL_STORAGE_KEY,JSON.stringify(todos))
  },[todos])
function toggleTodo(id){
  constnewTodos = [...todos]
```

```
  consttodo = newTodos.find(todo=>todo.id === id)
  todo.completed = !todo.completed
  setTodos(newTodos)
}
function clearTodo(){
  constnewTodo = todos.filter(todo =>!todo.completed)
  setTodos(newTodo)
}
  return (
    <>
    <input type="text" ref={todoNameRef} />
    <button onClick={addTodo}>Add</button>
    <button onClick={clearTodo}>Clear</button><br></br>
    <TodoListtodos={todos} toggleTodo={toggleTodo}/>
    </>
  );
}

export default App;
```

TodoList.js

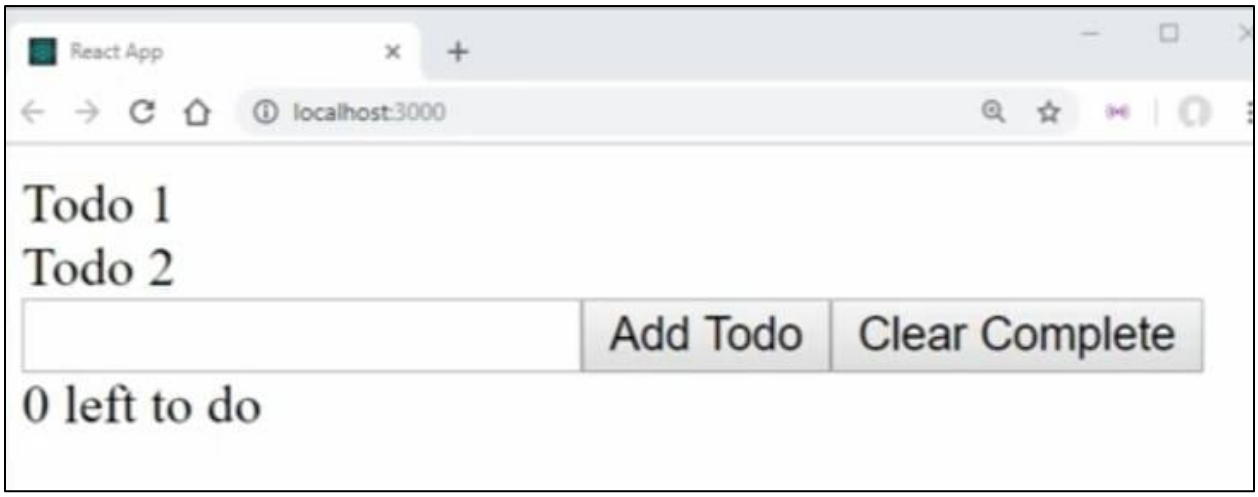
```
import React from 'react';
import Todo from './Todo'
export default function TodoList( {todos,toggleTodo} ){
  return (

          todos.map((todo) => {
              return <Todo key={todo.id} todo={todo} toggleTodo={toggleTodo}/>
          })

  )

}
```

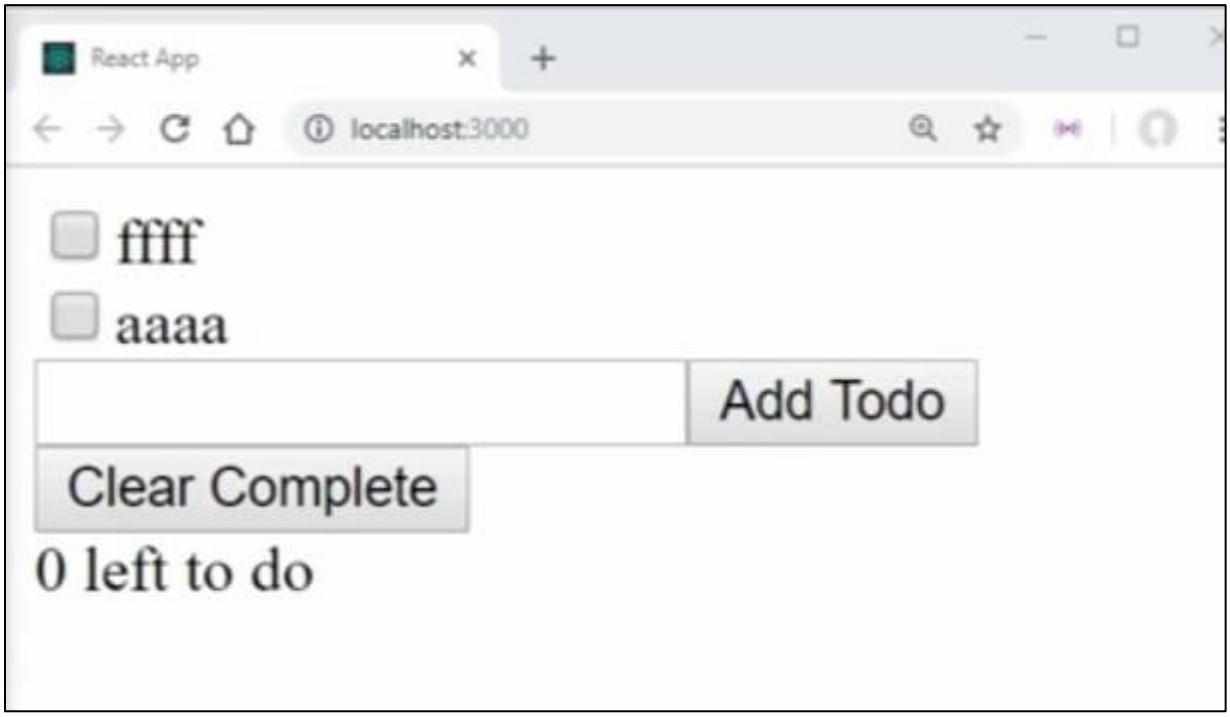
Todo.js

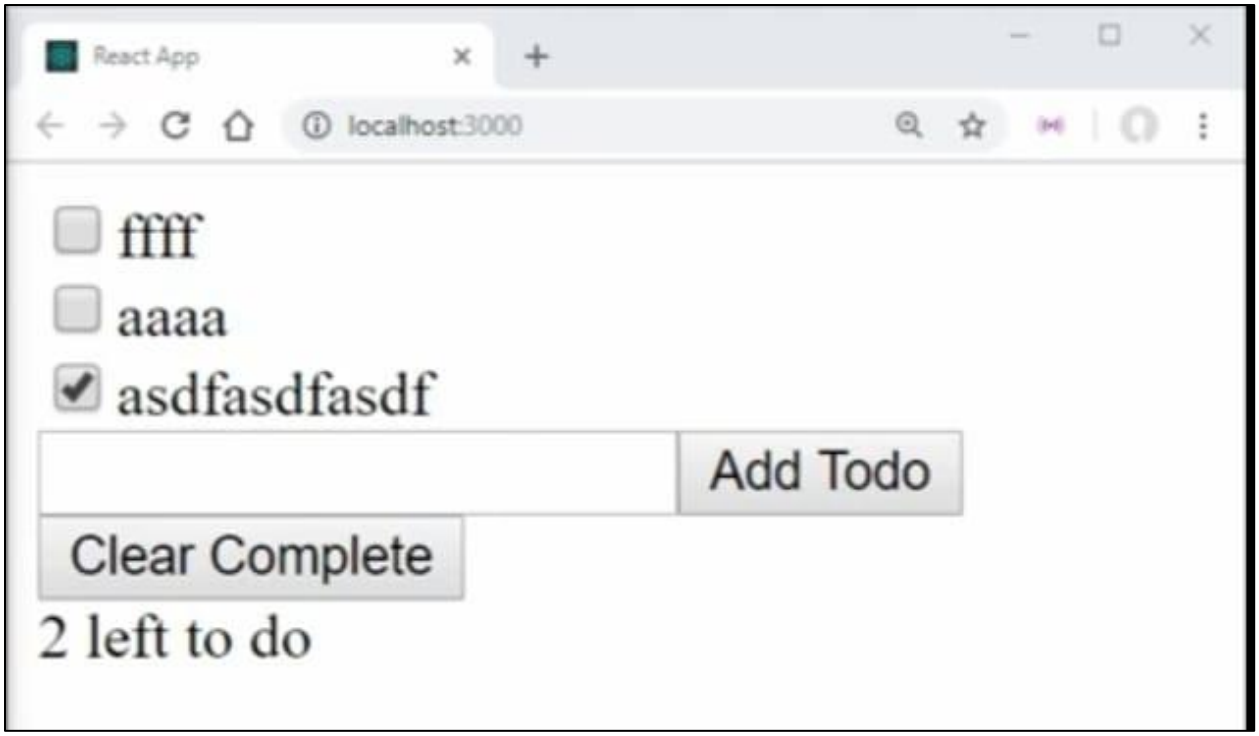
```
import React from 'react'

export default function Todo({todo,toggleTodo}) {
  function handleTodoClick(){
    toggleTodo(todo.id)
  }
  return (
      <div>
          <input type="checkbox" checked={todo.completed} onChange={handleTodoClick}/>
          <label >{todo.name}</label>

    </div>
```

```
  )
}
```

Output


React App
localhost:3000
Todo 1
Todo 2
Add Todo
Clear Complete
0 left to do


React App
localhost:3000
ffff
aaaa
Add Todo
Clear Complete
0 left to do



Result

The Given TODO Application Has Been Successfully Verified And The Output Was Executed.

16.HANDLING COOKIES WITH NODEJS AND EXPRESS FRAMEWORK

Aim

To Create An Handling Cookies With Nodejs And Express Framework.

ALGORITHM

Step 1 : Open The Vs Code To Write The Program
Step 2 : Create An Folder For The Cookies Program
Step 3 : Code The Codings In The App.Js
Step 4 : Create An Style And The Frame For The Website
Step 5 : Crate An Username And The Password
Step 6 : Create An Separate Style Sheet For The Program
Step 7 : Create Background Styles And Padding
Step 8 : Create An Folder For The Cookies And
Step 9 : Write The Code
Step 10 : Stop The Program

SOURCE CODE

Login.html

```
<html>
   <head>
      <title></title>
   </head>
   <link ref="stylesheet" href="style.css">
   <body>
      <div class="form">
         <h2>Login Form</h2>
         <div>
            <form action="/loginResult" method="POST">

         <input type="text" id="text" name="username" placeholder="Enter your name">
         <input type="text" id="text" name="pass" placeholder="Enter your password">
         <input id="login" type="submit">
      </form>
      </div>
      </div>
   </body>
</html>
```

Register.html

```
<html>
   <head>
      <title></title>
```

```
    </head>
    <link rel="stylesheet" ref="style.css">
    </style>
    <body>
        <div class="form">
            <h2>Register Form</h2>
            <div>
                <form action="/login" method="POST">
            <input type="text" id="text" name="username" placeholder="Enter your name">
            <input type="text" id="text" name="pass" placeholder="Enter your password">
            <input id="register" type="submit">
        </form>
        </div>
        </div>
    </body>
</html>
```

Style.css

```
.form{
    background:rgb(0,0,0,0.5);
    width:450px;
    height:400px;
    border-radius:20px;
    box-shadow:0 0 50px teal;
    margin:auto;
    margin-top:100px;
}
input[type="text"]{
    background:transparent;
    padding:20px 20px 20px20px;
    width:400px;
    border:none;
    outline:none;
    border-bottom:3px solid teal;
    border-radius:6px;
    margin-top:40px;
    margin-left:26px;
    color:white;
    column-rule-color:yellow;
}
input[type="text"]::placeholder{
    background:transparent;
    text-align:center;
    font-family: Arial, sans-serif;
    color:black;
    cursor: pointer;

}
input[type="button"]{
```

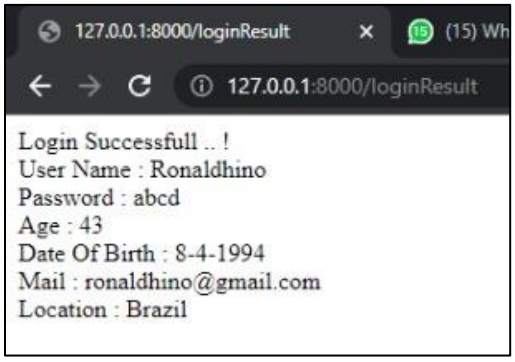
```
    font-size: large;
    font-style:normal;
    font-family: Arial,sans-serif;
    padding:20px;
    border:none;
    align-items: centers;
    margin:auto;
}
#login1{
    background:teal;
    width:140px;
    border:none;
    outline:none;
    margin-left:150px;
    border-radius:20px;
    margin-top:20px;
}
.form_login{
    margin-left:60px;
}
h2{

    color:white;
    text-align: center;
}
#login{
    width:150px;
    padding:10px 30px;
    cursor:pointer;
    display:block;
    margin:auto;
    margin-top:15px;
    font-size: 30px;;
    border-radius:10px;
    background:linear-gradient(to right,#ff105f,#ffad06)
}
```

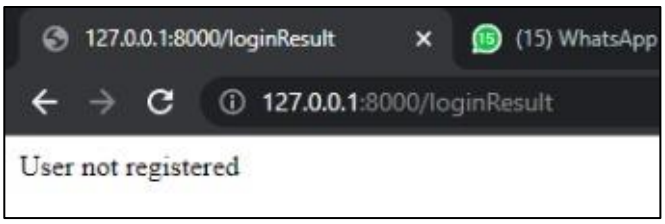
Cookie.js

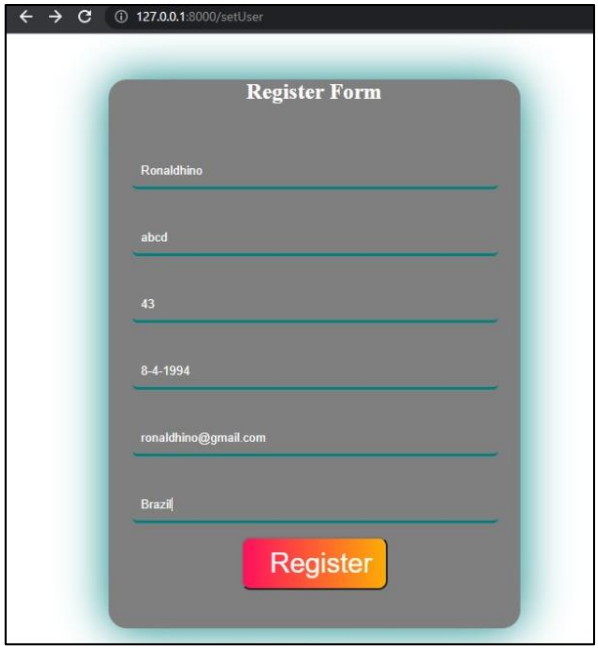
```
const express = require('express')
const path = require('path')
const cookie = require('cookie-parser')
constbodyParser = require('body-parser');
const app = express()
app.use(cookie())
app.use(bodyParser.urlencoded({extended:false}))
app.get('/',(req,res)=>{
res.redirect('setUser')
res.send("Hi register first")
```

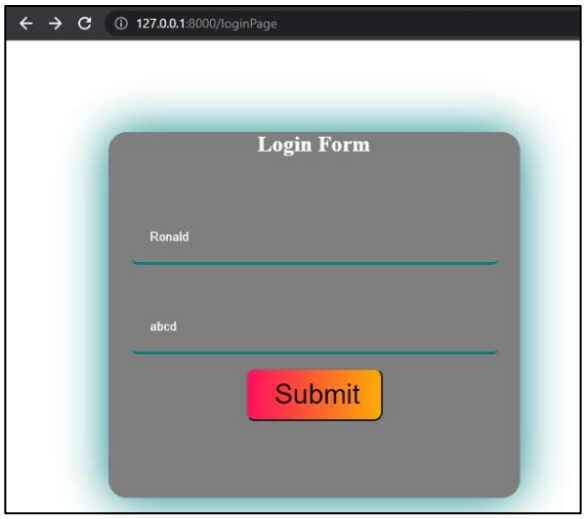
```
})
app.get('/setUser',(req,res)=>{
   res.sendFile(path.join(__dirname+'/register.html'))
})
app.get('/loginPage',(req,res)=>{
   res.sendFile(path.join(__dirname + '/login.html'))
})
app.post('/login',(req,res)=>{
   const name = req.cookies.username
   const pass = req.cookies.pass
   res.cookie('username',req.body.username)
   res.cookie( 'password',req.body.pass)
   res.redirect('/loginPage')

})
app.post('/loginResult',(req,res)=>{
   const name = req.cookies.username;
   const pass = req.cookies.password;
   const username = req.body.username;
   const password = req.body.pass;
   console.log(name + ' ' + username)
   console.log(pass + ' ' + password)
   if(name == username && pass == password){
      const co = res.cookies;
      res.send("Login Successfull .. !<br>User Name : " + name +'<br>Password : '+pass)
   }
   else{

      res.send("User not registered")
   }
})

app.listen(8000,()=>{
   console.log("Running ... !")
});
```

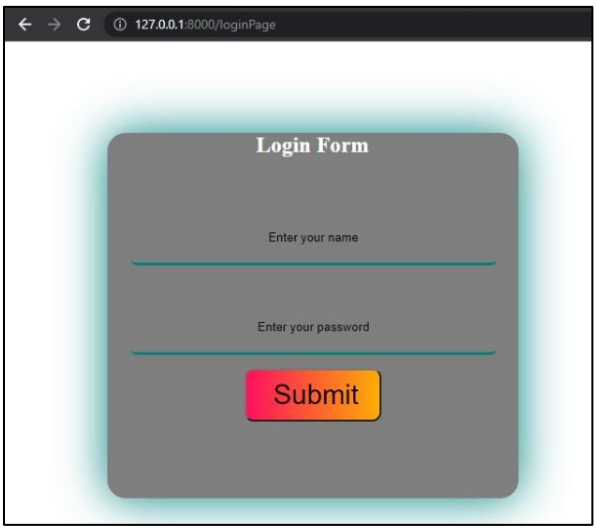
Output






127.0.0.1:8000/setUser
Register Form
Ronaldhino
abcd
43
8-4-1994
ronaldhino@gmail.com
Brazil
Register


127.0.0.1:8000/loginPage
Login Form
Ronald
abcd
Submit


127.0.0.1:8000/loginPage
Login Form
Enter your name
Enter your password
Submit

Results

The Given Program To Create Handle The Cookies Has Been Verified And The Output Was Succesfully Executed.