# Refreshing Java Script and CSS

1. Write an HTML & CSS code to create Horizontal and Vertical Menu.

<!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>Menu</title>

    <style>

        h1 {

            color:rgb(49, 100, 100);

        }

        ul {

            list-style-type: none;

            margin: 0;

            padding: 0;

            overflow: hidden;

            background-color: rgb(249, 212, 0);

        }

        .nav1{

            float: left;

        }

        li:hover {

            background-color:rgb(157, 156, 87);

        }

        .hnav {

            display: inline-block;

            color: rgb(0, 0, 0);

            padding: 12px 12px;

            text-decoration: none;

        }

        .vnav {

            display: block;

            color: rgb(0, 0, 0);

            padding: 12px 12px;

            text-decoration: none;

        }

    </style>

</head>

<body>

    <div>

        <h1>Horizontal Menu</h1>

        <ul>

            <li class="nav1"><a href="#" class="hnav">Home</a></li>

            <li class="nav1"><a href="#" class="hnav">About</a></li>

            <li class="nav1"><a href="#" class="hnav">Contact</a></li>

            <li class="nav1"><a href="#" class="hnav">Login</a></li>

        </ul>

    </div>

<br>

    <div>

        <h1>Vertical Menu</h1>

        <ul>

            <li><a href="#" class="vnav">Home</a></li>

            <li><a href="#" class="vnav">About</a></li>

            <li><a href="#" class="vnav">Contact</a></li>

            <li><a href="#" class="vnav">Login</a></li>

        </ul>

    </div>

</body>

</html>

Output:-



1. Write JavaScript code for loop that will iterate from 0 to 15 for each iteration, it will check if the current number is odd or even, and display a message to the screen.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Document</title>

    <script type="text/javascript">

        var i;

        for(i=0; i<=15; i++)

        {

            if(i==0)

            {

                document.writeln('<br>'+i+' '+'is Even');

            }

            else if(i%2==0)

            {

                document.writeln('<br>'+i +' '+'is Even');

            }

            else

            {

                document.writeln('<br>'+i+' '+'is Odd');

            }

        }

    </script>

</head>

<body>

</body>

</html>

Output:-



1. Write an HTML and JavaScript program which accepts N as input and display first N Fibonacci numbers as list.

<!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>Fibonacci Series</title>

</head>

<body>

    <script type="text/javascript">

        function fibo() {

            var n1=0, n2=1, nextnum, i;

            var N = document.getElementById('num').value;

            for(i=0; i<N; i++) {

                document.getElementById('result').innerHTML += (" "+n1); //for loop overwrite innerHTML in each iteration so we use +=(compound operator)

                nextnum=n1+n2;

                n1=n2;

                n2=nextnum;

             }

        }

    </script>

    <h1>Fibonacci Series(200210116006)</h1>

    <label for="num">Enter Term:</label>

    <input type="number" id="num">

    <input type="button" value="Submit" onclick="fibo()">

    <br><br>

    <p>Fibonacci Series:<span id="result"></span></p>

</body>

</html>

Output:-



1. Write JavaScript code to know which mouse button was clicked, number of elements in form, and write hello world on the document.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Document</title>

    <script type="text/javascript">

        document.onmousedown = mouse;

     function mouse(event) {

         if(event.button == 0) {

             document.getElementById("result").innerHTML = "Left Mouse Button Was Clicked.";

         }

         if(event.button == 1){

             documnet.getElemntById("result").innerHTML = "Middle Mouse Button Was Clicked";

         }

         if(event.button == 2){

             document.getElementById("result").innerHTML = "Right Mouse Button Was Clicked";

         }

     }

     function num() {

        var x = document.getElementById("myform").elements.length;

        document.getElementById("result2").innerHTML = x +" "+ "element(s) in the form";

     }

    //  document.getElementById("result3").innerHTML = "Hello World!";

 </script>

</head>

<body>

    <h1>Mouse Event</h1>

    <p id="result"></p>

    <h1>Number Of Element In Form</h1>

    <form id="myform" action="">

        <input type="text">

        <input type="number">

    </form>

    <button onclick="num()">Click Me</button>

    <p id="result2"></p>

    <h1>Message</h1>

    <p id="result3"></p>

    <script type="text/javascript">

             document.write("Hello World!");

    </script>

</body>

</html>

Output:-



1. Write JavaScript code to check mobile number (mobile no. should start with 9 or 8).

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Document</title>

    <script type="text/javascript">

        function valid() {

                var x = document.getElementById('mobile').value;

                if (x.charAt(0) == 9 || x.charAt(0) == 8) {

                    document.getElementById('error').innerHTML ="\*Mobile Number should not start with" + " " + x.charAt(0);

                }

                else {

                    document.getElementById('error').innerHTML = "Mobile Number is valid";

                }

            }

    </script>

</head>

<body>

    <label for="mobile">Mobile Number:</label>

    <input id="mobile" type="tel" maxlength="10">

    <span id="error"></span>

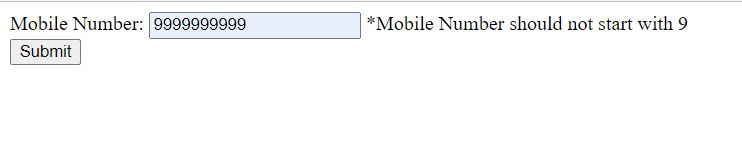
    <br>

    <input type="button" value="Submit" onclick="valid()">

</body>

</html>

Output:-



# Angular JS Program

1. Write an AngularJS code which takes number as an input and display that number is odd or even.

Code:-

<html>

<head>

  <title>OddEven</title>

  <script src="../angular-1.8.2/angular.min.js"></script>

  <script>

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

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

    {

      $scope.oddeven=function(result)      // first we use only num so we got error

      {

        if(result == null)

        {

          return(document.getElementById('output').innerHTML="<br>"+"\*Please Enter a number to check number is Odd or Even");

        }

        else if(result % 2 == 0)

        {

          return(document.getElementById('output').innerHTML="<br><br>"+"Number is even");}

        else {

          return(document.getElementById('output').innerHTML="<br><br>"+"Number is Odd");}}});

  </script>

</head>

<body>

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

      <input type="number" ng-model="num">

      <!-- {{num}} -->

      <span id="output" ng-bind="oddeven(num)"></span>

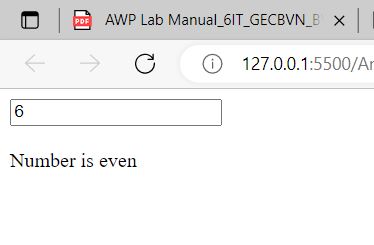
      <!-- {{oddeven()}}   expression -->

    </div>

</body>

</html>

Output:-



1. Design Order Form with a total price updated in real time, which contains name of five products and their prices. Create a bill amount for all the products and calculate GST on the billing amount and display total amount.

Code:-

<!DOCTYPE html>

<html lang="en" ng-app="myapp">

<head>

    <meta charset="UTF-8">

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

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

    <title>Document</title>

    <style>

        table, th, td{

            border: 2px solid black;

        }

    </style>

    <script src="../angular-1.8.2/angular.min.js"></script>

<script>

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

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

        {

            $scope.product = [

                {name: "Apple", price: 10},

                {name: "Banana", price: 20},

                {name: "Mango", price: 30},

                {name: "Graps", price: 40},

                {name: "watermelon", price: 50}

            ]

            $scope.totalamount=0;

            $scope.gst=0;

            $scope.bill=function() {

                var total=0;

                for(var i=0; i< $scope.product.length; i++)

                {

                    var p = $scope.product[i];

                    total += p.price;

                }

                $scope.gst = total \* 0.18;

                $scope.totalamount = total + $scope.gst;;

            }

        });

    </script>

</head>

<body ng-controller="myctrl">

    <table>

        <tr>

            <th>Product Name</th>

            <th>Product Price</th>

        </tr>

        <tr ng-repeat="p in product">

            <td>{{p.name}}</td>

            <td>{{p.price}}</td>

        </tr>

    </table>

    <p>To Calculate TotalAmount & GST</p>

    <button ng-click="bill()">Generate Bill</button>

    <table>

        <tr>

            <th>GST</th>

            <th>TotalAmount</th>

        </tr>

        <tr>

            <td>{{gst}}</td>

            <td>{{totalamount}}</td>

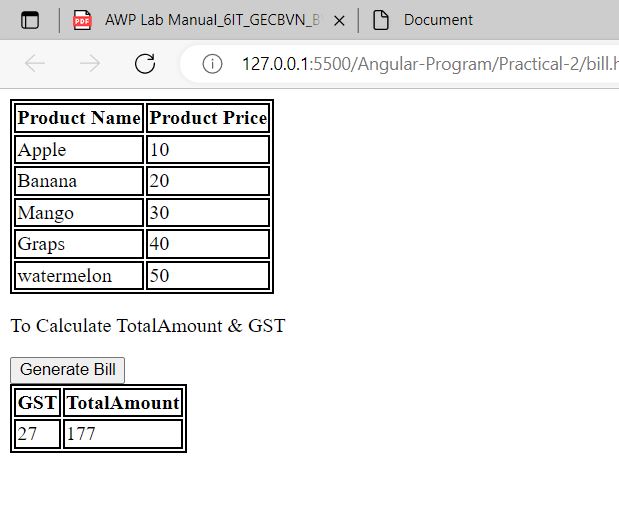
        </tr>

    </table>

</body>

</html>

Output:-



1. Design a webpage which takes one number as an input and generate its factorial number (use module, controller)

Code:-

<!DOCTYPE html>

<html lang="en" ng-app="myApp">

<head>

    <meta charset="UTF-8">

    <title>Factorial Generator</title>

    <script src="../angular-1.8.2/angular.min.js"></script>

    <script>

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

        myApp.controller('myctrl', function($scope) {

            $scope.num = null;

            $scope.result = null;

            $scope.factorial = function() {

                var num = $scope.num;

                var result = 1;

                for (var i = 2; i <= num; i++) {

                    result \*= i;

                }

                $scope.result = result;

            }

            });

    </script>

</head>

<body ng-controller="myctrl">

    <h2>Factorial</h2>

    <p>Enter a number to calculate its factorial:</p>

    <input type="number" ng-model="num">

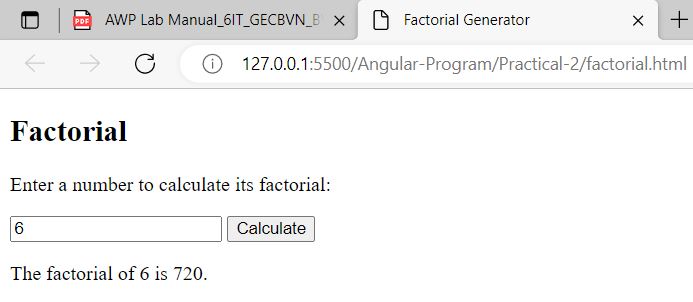
    <button ng-click="factorial()">Calculate</button>

    <p ng-if="result !== null">The factorial of {{num}} is {{result}}.</p>

</body>

</html>

Output:-



1. Design a webpage which takes inputs product name, product quantity and price. Generate table of entered values. When user clicks on table column title, it should sort that column values.(use filter, array)

Code:-

<html ng-app="myapp" >

<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>Product</title>

    <style>

    table,th, td{

            border: 1px solid black;

        }

        /\* button {

            border: none;

        } \*/

    </style>

     <script src="../angular-1.8.2/angular.min.js"></script>

     <script>

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

        app.controller("myctrl",function($scope, $filter){

            $scope.product = [];

            $scope.add = function(){

                   var pname=$scope.pname;

                   var quantity=$scope.quantity;

                   var price=$scope.price;

                   var arr = {

                    productname: pname,

                    productquantity: quantity,

                    productprice: price

                   };

                   $scope.product.push(arr);

                   $scope.pname="";

                   $scope.quantity="";

                   $scope.price="";

            };

            $scope.sortColumn = "productname";

            $scope.reverseSort=false;

            $scope.sortdata = function(column) {

                $scope.reverseSort = ($scope.sortColumn == column) ? !$scope.reverseSort : false;

                $scope.sortColumn = column;

                $scope.product = $filter('orderBy')($scope.product, $scope.sortColumn, $scope.reverseSort);

            };

        });

    </script>

</head>

<body ng-controller="myctrl">

    <form>

        <label for="name1">Product Name:</label>

        <input type="text" id="name1" ng-model="pname">

        <br>

        <label for="name2">Product Quantity:</label>

        <input type="number" id="name2" ng-model="quantity">

        <br>

        <label for="name3">Produc price:</label>

        <input type="number" id="name3" ng-model="price">

        <br>

        <button type="button" ng-click="add()">Add Data</button>

    </form>

        <table>

            <tr>

                <th ng-click="sortdata('name')">Product Name</th>

                <th ng-click="sortdata('quantity')">Product Quantity</th>

                <th ng-click="sortdata('price')">Product Price</th>

            </tr>

            <tr ng-repeat="p in product | orderBy:sortColumn:reverseSort">

                <td>{{ p.productname }}</td>

                <td>{{ p.productquantity }}</td>

                <td>{{ p.productprice }}</td>

            </tr>

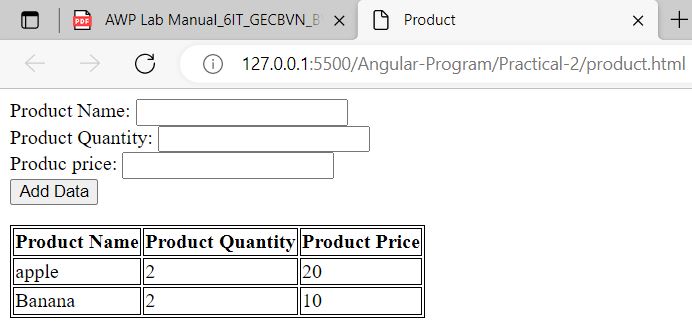
        </table>

    </div>

</body>

</html>

Output:-



1. Design a webpage which display product name and product price using AngularJS $http Service from database. Display the content in tabular format.

Code:-

Database.html

<!DOCTYPE html>

<html>

<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>Database</title>

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

</head>

<body ng-app="myapp"">

    <div ng-controller="myctrl">

        <table>

            <tr>

                <th>Product Name</th>

                <th>Product Price</th>

            </tr>

        <tr ng-repeat="product in productdata track by $index">

            <td>{{product.name}}</td>

            <td>{{product.price}}</td>

            </tr>

    </table>

    </div>

</body>

<script>

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

        app.controller("myctrl",function($scope, $http) {

            $http.get("product.php").then(function(response) {

                $scope.productdata = response.data;

            });

        });

</script>

</html>

product.php

<?php

//create database connection

$host='localhost';

$username = 'root';

$password = '';

$dbname = 'mydb';

$con = mysqli\_connect($host,$username,$password,$dbname);

if (!$con) {

    die('Could not connect: ' . mysqli\_connect\_error());

}

//fetch product data from database

$sql = "SELECT \* FROM products";

$result = mysqli\_query($con,$sql);

$data = array();

//put product data into array

if(mysqli\_num\_rows($result) > 0)

{

    while($row = mysqli\_fetch\_assoc($result))

    {

        $data[] = $row;

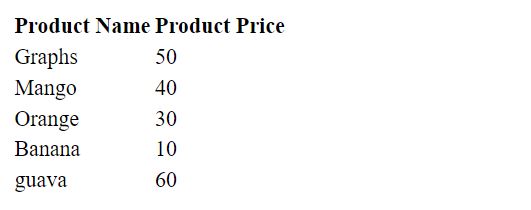
    }

    echo json\_encode($data);

}

?>

Output:-



# Node JS Program

1. Create a Node.js file that will convert the output "Hello World!" into upper-case letters.

Code:-

var http = require('http');

var upper=require('upper-case');

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

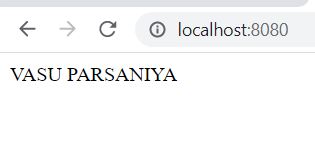
  res.write(upper.upperCase("Vasu Parsaniya"));

  res.end();

}).listen(8080);

// console.log("HelloWorld!");

Output:-



1. Write a Node.js module of calculator which will perform all the basic operations like add(), sub(), mul() and div().Use the module in a program and display the output.

Code:-

calculator.js

module.exports = {

  add: function(a, b) {

    return a + b;

  },

  sub: function(a, b) {

    return a - b;

  },

  mul: function(a, b) {

    return a \* b;

  },

  div: function(a, b) {

    if (b === 0) {

      throw new Error('Division by zero is not allowed.');

    }

    return a / b;

  }

};

main.js

const calculator = require('./calculator');

console.log(calculator.add(2, 3));

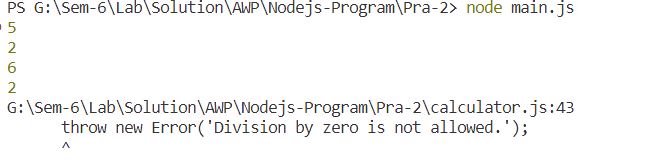
console.log(calculator.sub(5, 3));

console.log(calculator.mul(2, 3));

console.log(calculator.div(6, 3));

console.log(calculator.div(6, 0));

Output:-



1. Create a Node.js Application that uses user defined module circle.js which exports functions area() and circumference() and display details on console.

Code:-

circle.js

module.exports = {

    area: function(r) {

        return (22/7)\*r\*r;

    },

    circumference: function(r) {

        return 2\*(22/7)\*r;

    }

}

main.js

const circle = require('./circle');

console.log(circle.area(2));

console.log(circle.circumference(4));

Output:-



1. Create a Node.js program to perform file operations like create a file, read a file, write to file and delete a file.

Code:-

const fs = require('fs');

// Create a file

fs.writeFile('example.txt', 'Vasu Parsaniya', (err) => {

  if (err) throw err;

  console.log('File created!');

});

// Read a file

fs.readFile('example.txt', 'utf8', (err, data) => {

  if (err) throw err;

  console.log('File contents:', data);

});

// Write to a file

fs.appendFile('example.txt', '\nThis is a new line.', (err) => {

  if (err) throw err;

  console.log('Data written to file!');

});

// Delete a file

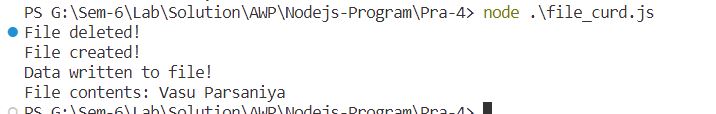
fs.unlink('example.txt', (err) => {

  if (err) throw err;

  console.log('File deleted!');

});

Output:-



1. Write Write Node.js code to perform Insert, update and delete operations on Employee database using Node.js and MongoDB.

Code:-

const { MongoClient} = require('mongodb');

const url = 'mongodb://127.0.0.1:27017';

const client = new MongoClient(url);

async function createNewDatabase(dbName) {

  try {

    await client.connect().then(() => console.log("Connected With Server")).catch(err => console.log("Unable to connect"));

    var db = client.db(dbName);

    // var collection = db.createCollection('employeeCollection');

    console.log(`Created new database '${dbName}'`);

  } catch (err) {

    console.error(err);

  } finally {

    await client.close();

  }

}

//--------------------Insert data-----------------------------------------

async function insertData(dbName, collectionName) {

  //establish the connection with MongoDB database

  await client.connect().then(() => console.log("Again Connected")).catch(err => console.log("Unable to connect to database"));

  //We take database reference

  const myDB = client.db(dbName);

  /\* we cannot use here createCollection() because it create only one time collection if second time same code

  we run then give error So we us collection().

  -->if collection is not created then automatically collection() can create collection\*/

  const myColl = myDB.collection(collectionName);

  try {

    const data = [

      { name: 'Vasu', age: 20 },

      { name: 'Ashneer', age: 30 },

      { name: 'Piyush', age: 40 },

      { name: 'Tata', age: 85 }

    ];

    const insertManyresult = await myColl.insertMany(data);

    let ids = insertManyresult.insertedIds;

    console.log(`${insertManyresult.insertedCount} documents were inserted.`); //count the number of inserted document

    for (let id of Object.values(ids)) {

      console.log(`Inserted a document with id ${id}`);

    }

  } catch (e) {

    console.log(e);

    // console.log(`A MongoBulkWriteException occurred, but there are successfully processed documents.`);

    let ids = e.result.result.insertedIds;

    for (let id of Object.values(ids)) {

      console.log(`Processed a document with id ${id.\_id}`);

    }

    console.log(`Number of documents inserted: ${e.result.result.nInserted}`);

  } finally {

    await client.close();

  }

}

//----------------------Update data-------------------------------------

async function updateData(dbName, collectionName) {

  //establish the connection with MongoDB database

  try {

    await client.connect().then(() => console.log("Again Again Connected")).catch(err => console.log("Unable to connect to database"));

    const myDB = client.db(dbName);

    const myColl = myDB.collection(collectionName);

    const filter = { name: 'Vasu' };

    // update the value of the 'quantity' field to 5

    const updateDocument = {

      $set: { age: 16 },

    };

    const result = await myColl.updateOne(filter, updateDocument);

  } catch (e) {

    console.log(e);

  } finally {

    await client.close();

  }

}

//---------------Delete data-------------------------------------

async function deleteData(dbName, collectionName) {

  //establish the connection with MongoDB database

  try {

    await client.connect().then(() => console.log("Again Again Again Connected")).catch(err => console.log("Unable to connect to database"));

    const myDB = client.db(dbName);

    const myColl = myDB.collection(collectionName);

    const deleteData = { name: 'Vasu', age: 16 };

    const result = await myColl.deleteOne(deleteData);   //deleteOne() take parameter as Object like JSON, javascript object

  } catch (e) {

    console.log(e);

  } finally {

    await client.close();

  }

}

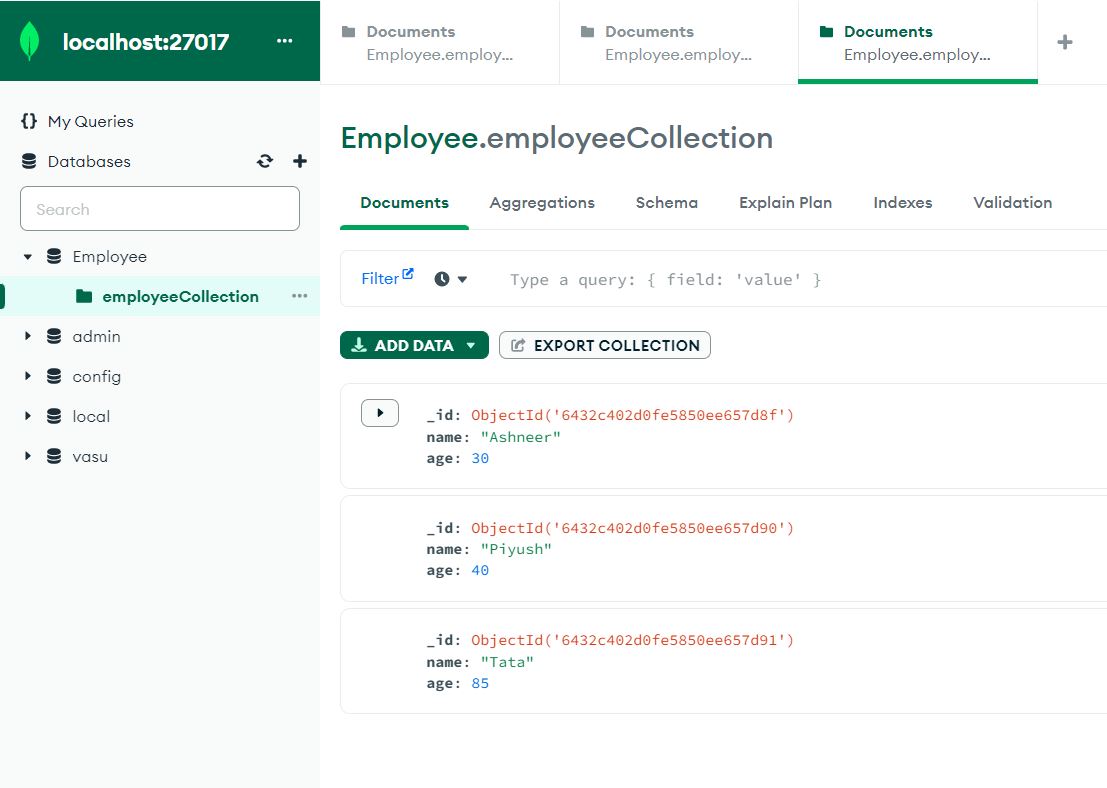
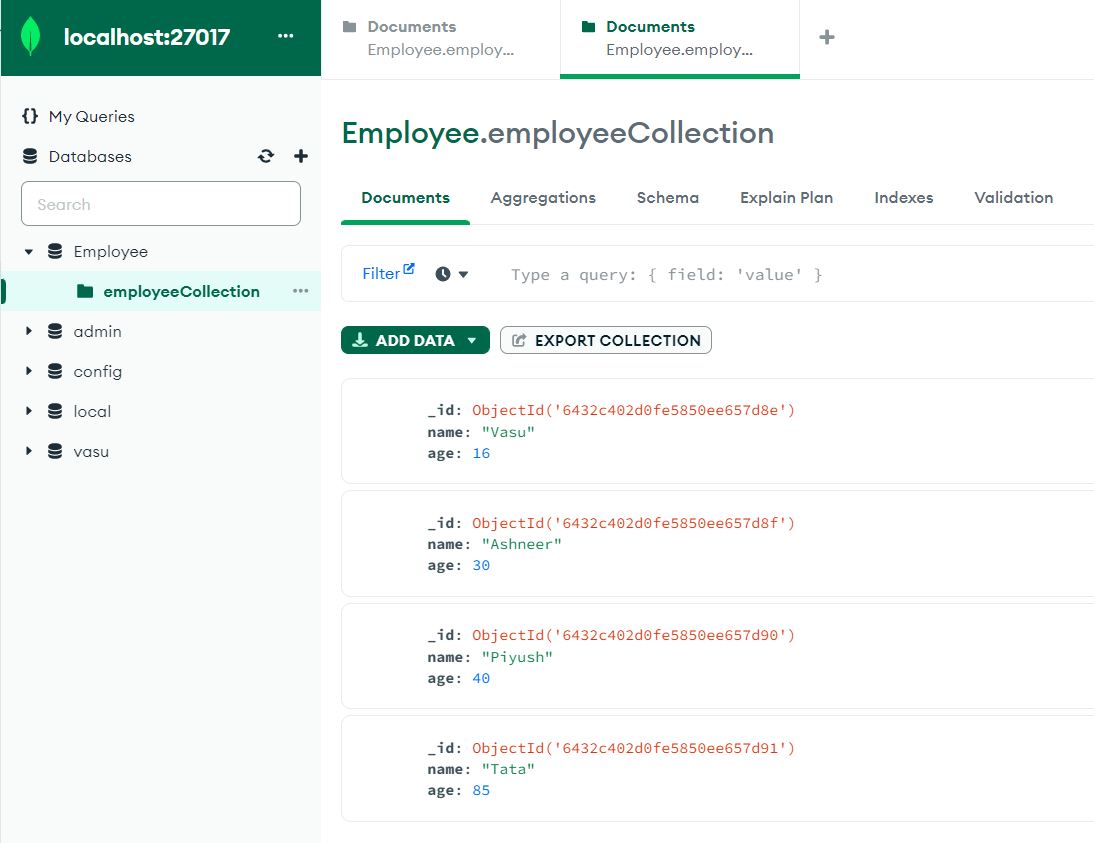
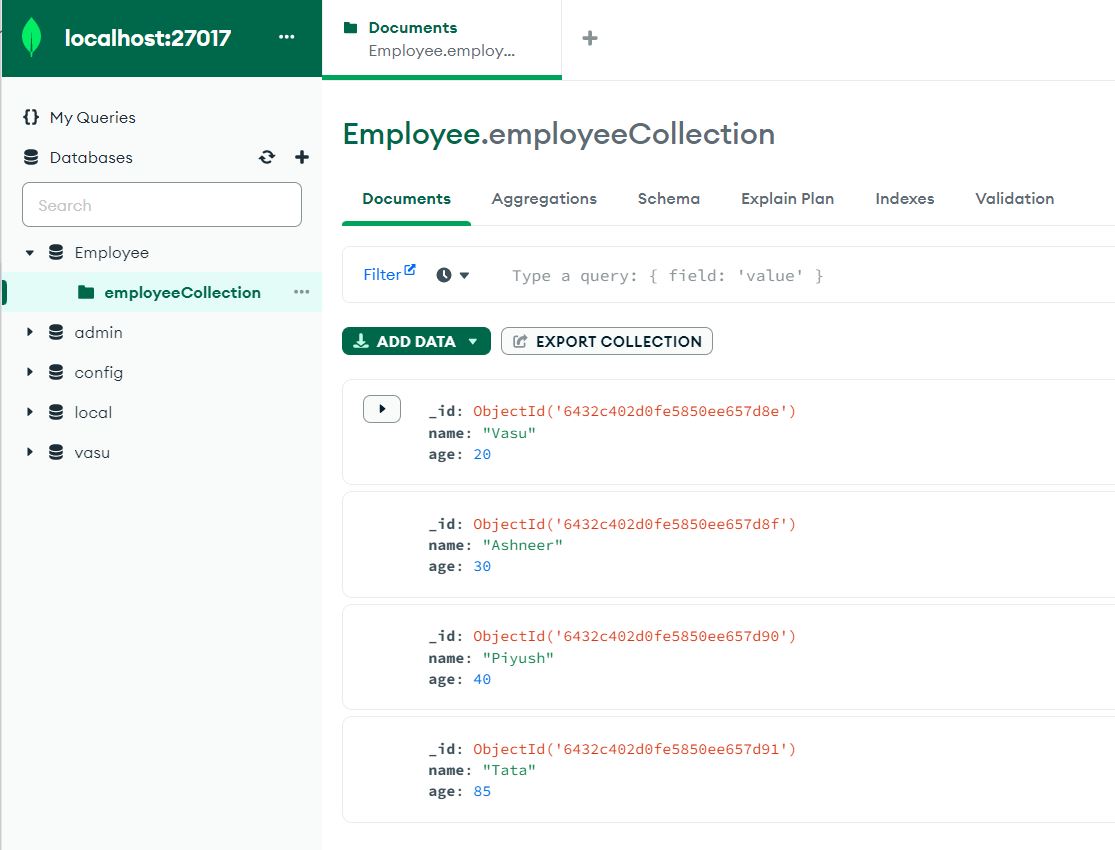
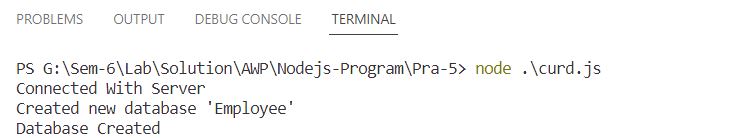
createNewDatabase('Employee').then(() => console.log("Database Created")).catch(err => console.log("Unable to Create Database"));

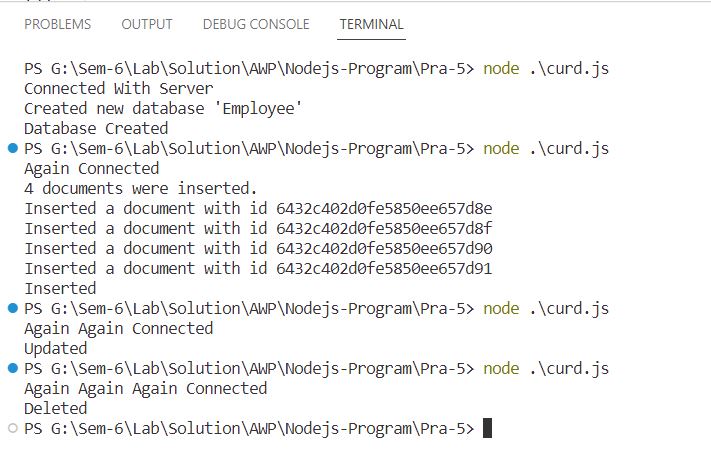
insertData('Employee', 'employeeCollection').then(() => console.log("Inserted")).catch(err => console.log("Unable to Insert"));

updateData('Employee', 'employeeCollection').then(() => console.log("Updated")).catch(err => console.log("Unable to Update"));

deleteData('Employee', 'employeeCollection').then(() => console.log("Deleted")).catch(err => console.log("Unable to Update"));

Output:-





# Database Programming with Angular,Node JS and MySQL

**Web Application:-** Cafe Management System

**Code:-**

**Backend:-**

index.js:

const express = require('express');

var cors = require('cors');     //cross-origin resource sharing

const connection = require('./connection');

const userRoute = require('./routes/user');

const categoryRoute = require('./routes/category');

const productRoute = require('./routes/product');

const billRoute = require('./routes/bill');

const dashboardRoute = require('./routes/dashboard');

const app = express();

app.use(cors());

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

app.use(express.json());

app.use('/user',userRoute);

app.use('/category',categoryRoute);

app.use('/product',productRoute);

app.use('/bill',billRoute);

app.use('/dashboard',dashboardRoute);

module.exports = app;

.env:

#Server

PORT = 8080

#Connection

DB\_PORT = 3306

DB\_HOST = localhost

DB\_USERNAME = root

DB\_PASSWORD = your database password

DB\_NAME = cafenodejs

#secret key

ACCESS\_TOKEN = ba48f495673ff6fd42356a957ce061076a351e8b2a4702ac42aea5cd7bfa536567bb04dd809bff45ef057f5506d50798f7551650e7622a8db16546e0c9a6fcf7

#vwkwjanzdtbdorae

EMAIL = your mailid

PASSWORD = your mailid password

#This is fake mail

#EMAIL =   #ethereal mail id

#PASSWORD =   #ethereal mail password

USER = user

connection.js:

const mysql = require('mysql');

require('dotenv').config();

var connection = mysql.createConnection({

    port: process.env.DB\_PORT,

    host: process.env.DB\_HOST,

    user: process.env.DB\_USERNAME,

    password: process.env.DB\_PASSWORD,

    database: process.env.DB\_NAME

});

connection.connect((err) =>{

    if(!err){

        console.log("Connected");

    }

    else{

        console.log(err);

    }

});

module.exports = connection;

table.sql:

create table user(

    id int primary key AUTO\_INCREMENT,

    name varchar(250),

    contactNumber varchar(20),

    email varchar(50),

    password varchar(20),

    status varchar(20),

    role varchar(20),

    UNIQUE(email)

);

insert into user(name,contactNumber,email,password,status,role) values('Admin','1234567890','admin@gmail.com','admin','true','admin');

--category

create table category(

    id int NOT NULL AUTO\_INCREMENT,

    name varchar(255) NOT NULL,

    primary key(id)

);

--products

create table product(

    id int NOT NULL AUTO\_INCREMENT,

    name varchar(255) NOT NULL,

    categoryId integer NOT NULL,

    description varchar(255),

    price integer,

    status varchar(20),

    primary key(id)

);

--bill

create table bill(

    id int NOT NULL AUTO\_INCREMENT,

    uuid varchar(250) NOT NULL,

    name varchar(255) NOT NULL,

    email varchar(255) NOT NULL,

    contactNumber varchar(20) NOT NULL,

    paymentMethod varchar(20) NOT NULL,

    total int NOT NULL,

    productDetails JSON DEFAULT NULL,

    createdBy varchar(255) NOT NULL,

    primary key(id)

);

user.js:

const express = require('express');

const connection = require('../connection');

const router = express.Router();

const jwt = require('jsonwebtoken');

require('dotenv').config();

const nodemailer = require('nodemailer');

var auth = require('../services/authentication');

var checkRole = require('../services/checkRole');

//----------------------signup API-------------------------------

router.post('/signup', (req, res) => {

    let user = req.body;

    query = "select email,password,role,status from user where email=?"

    connection.query(query, [user.email], (err, results) => {

        if (!err) {

            if (results.length <= 0) {    //user is available with this email or not we check

                /\*if the status is "true" then user is able to login and "false" then user is not able

                ? means user enter those value take\*/

                query = "insert into user(name,contactNumber,email,password,status,role) value(?,?,?,?,'false','user')";

                connection.query(query, [user.name, user.contactNumber, user.email, user.password], (err, results) => {

                    if (!err) {

                        return res.status(200).json({ message: "Successfully registered" });

                    }

                    else {

                        return res.status(500).json(err);

                    }

                })

            } else {

                return res.status(400).json({ message: "Email Already Exist" });

            }

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//------------------------Login API---------------------------------

router.post('/login', (req, res) => {

    const user = req.body;   //enter data by user

    query = "select email,password,role,status from user where email=?";

    connection.query(query, [user.email], (err, results) => {  //results is from data base

        if (!err) {

            if (results.length <= 0 || results[0].password != user.password) {

                return res.status(401).json({ message: "Incorrect Username or Password" });

            }

            else if (results[0].status === "false") {    //we check that user having status is true or not

                return res.status(401).json({ message: "Wait for Admin Approval" });

            }

            else if (results[0].password == user.password) {  //we generate token

                const response = { email: results[0].email, role: results[0].role }

                const accessToken = jwt.sign(response, process.env.ACCESS\_TOKEN, { expiresIn: '8h' })

                res.status(200).json({ token: accessToken });

            }

            else {

                return res.status(400).json({ message: "Something went wrong.Please try again later" });

            }

        } else {

            return res.status(500).json(err);

        }

    })

})

//-----------node mailer-----------------

var transporter = nodemailer.createTransport({

    service: 'gmail',

    host: 'smtp.google.email',

    // host: 'smtp.ethereal.email',

    port: 587,

    secure: false,

    auth: {

        user: process.env.EMAIL,

        pass: process.env.PASSWORD

    }

});

//--------------Forgot Password------------------

router.post('/forgotPassword', (req, res) => {

    const user = req.body;

    query = "select email,password from user where email=?";

    connection.query(query, [user.email], (err, results) => {

        if (!err) {

            if (results.length <= 0) {  //user does not exists in database

                //here we cannot saw error like email id is invalid because any one can check that particular email id exists or not

                return res.status(200).json({ message: "Password sent successfully to your email" });

            }

            else { //we sending mail

                var mailOptions = {

                    from: process.env.EMAIL,

                    to: results[0].email,        //enter by user

                    subject: 'Password by cafe Management System',

                    html: '<p><b>Your Login details for Cafe Management System</b><br><b>Email: </b>' + results[0].email + '<br><b>Password: </b>' + results[0].password + '<br><a href="http://localhost:4200/">Click Here To Login</a></p>'

                };

                transporter.sendMail(mailOptions, function (error, info) {

                    if (error) {

                        console.log(error);

                    }

                    else {

                        console.log('Email Sent:' + info.response);

                    }

                });

                return res.status(200).json({ message: "Password sent successfully to your email" });

            }

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//--------Get All User API---------------

router.get('/get', auth.authenticateToken, checkRole.checkRole, (req, res) => {

    var query = "select id,name,email,contactNumber,status from user where role='user'";

    connection.query(query, (err, results) => {

        if (!err) {

            return res.status(200).json(results);

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//----------Change the status of user API-----------

router.patch('/update', auth.authenticateToken, checkRole.checkRole, (req, res) => {

    let user = req.body;

    var query = "update user set status=? where id=?";

    connection.query(query, [user.status, user.id], (err, results) => {

        if (!err) {

            //we check that row affected or not

            //this is not affected

            if (results.affectedRows == 0) {

                return res.status(404).json({ "message": "User Id does not exist" });

            }

            return res.status(200).json({ "message": "User Updated Successfully" });  //when row change

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//------------------------------check token API-----------------------------

router.get('/checkToken', auth.authenticateToken, (req, res) => {

    return res.status(200).json({ "message": "true" })

})

//--------------------------change Password API-------------------------------

router.post('/changePassword',auth.authenticateToken, (req, res) => {

    const user = req.body;

    const email = res.locals.email;  //this is got from authenticateToken

    var query = "select \* from user where email=? and password=?";

    connection.query(query, [email, user.oldPassword], (err, results) => {

        if (!err) {

            if (results.length <= 0) {  //when old password is not in the database

                return res.status(400).json({ message: "Incorrect Old password" });

            } else if (results[0].password == user.oldPassword) {

                query = "update user set password=? where email=?";

                connection.query(query, [user.newPassword, email], (err, results) => {

                    if (!err) {

                        return res.status(200).json({ "message": "Password Updated Successfully" });

                    } else {

                        return res.status(500).json(err);

                    }

                })

            } else {

                return res.status(400).json({ "message": "Something Went Wrong. please try again later" });

            }

        }

        else {

            return res.status(500).json(err);

        }

    })

})

module.exports = router;

authentication.js:

require('dotenv').config()

const jwt = require('jsonwebtoken');

function authenticateToken(req, res, next) {

    //In this function we check that any API run then token will be exists in header or not

    const authHeader = req.headers['authorization']

    const token = authHeader && authHeader.split(' ')[1]

    // const token = authHeader && authHeader.split(' ')[0] === 'Bearer' && authHeader.split(' ')[1];

    if (token == null) {

        return res.sendStatus(401);   //unauthorized

    }

    jwt.verify(token, process.env.ACCESS\_TOKEN, (err, response) => {

        if (err) {

            return res.sendStatus(403);   //forbiden access

        }

        else {

            res.locals = response; // res.locals is an object that contains response local variables.

            next();

        }

    })

}

module.exports = { authenticateToken: authenticateToken }  //export middleware function  as object

checkRole.js:

require('dotenv').config();

function checkRole(req, res, next) {

    if (res.locals.role == process.env.USER) {

        res.sendStatus(401);  //Unauthorized

    }

    else {

        next();

    }

}

module.exports = { checkRole: checkRole }  //export checkRole function as object

/\*==>Some API is not allow to use by user

  ==>like /get , /update API does not allow to user only allow to admin\*/

dashboard.js:

const express = require('express');

const connection = require('../connection');

const router = express.Router();

var auth = require('../services/authentication');

//-------------------Details API---------------------

router.get('/details',auth.authenticateToken,(req,res,next) =>{

    var categoryCount;

    var productCount;

    var billCount;

    var query = "select count(id) as categoryCount from category";

    connection.query(query,(err,results)=>{

        if(!err){

            categoryCount = results[0].categoryCount;

        }

        else{

            return res.status(500).json(err);

        }

    })

    var query = "select count(id) as productCount from product";

    connection.query(query,(err,results)=>{

        if(!err){

            productCount = results[0].productCount;

        }

        else{

            return res.status(500).json(err);

        }

    })

    var query = "select count(id) as billCount from bill";

    connection.query(query,(err,results)=>{

        if(!err){

            billCount = results[0].billCount;

            var data = {

                category: categoryCount,

                product: productCount,

                bill: billCount

            };

            return res.status(200).json(data);

        }

        else{

            return res.status(500).json(err);

        }

    })

})

module.exports = router;

category.js:

const express = require('express');

const connection = require('../connection')

const router = express.Router();

var auth = require('../services/authentication');

var checkRole = require('../services/checkRole');

//--------------------------Add Category to database API-----------------------

router.post('/add',auth.authenticateToken,checkRole.checkRole,(req,res,next)=>{

    let category = req.body;

    var query = "insert into category(name) values(?)";

    connection.query(query,[category.name],(err,results)=>{

        if(!err){

            return res.status(200).json({ "message":"category Added Successfully" });

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//--------------------------Get All category API---------------------------------

router.get('/get',auth.authenticateToken,(req,res,next)=>{    //in this API we cannot check for Role because anyone can get the category

    var query = "select \* from category order by name";

    connection.query(query,(err,results)=>{

        if(!err){

            return res.status(200).json(results);

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//----------------------Update category API------------------------------------

router.patch('/update',auth.authenticateToken,checkRole.checkRole,(req,res,next)=>{

    let product = req.body;

    var query = "update category set name=? where id=?";

    connection.query(query,[product.name,product.id],(err,results)=>{

        if(!err){

            //any row is not changed

            if(results.affectRows == 0){

                return res.status(404).json({"message":"Category id does not found"});

            }

            return res.status(200).json({"message":"Category Updated Successfully"});

        }

        else{

            return res.status(500).json(err);

        }

    })

})

module.exports = router;

product.js:

const express = require('express');

const connection = require('../connection');

const router = express.Router();

var auth = require('../services/authentication');

var checkRole = require('../services/checkRole');

//-------------------Product Add API-----------------------------

router.post('/add',auth.authenticateToken,checkRole.checkRole,(req,res)=>{

    let product = req.body;

    var query = "insert into product (name,categoryId,description,price,status) values(?,?,?,?,'true')";

    connection.query(query,[product.name,product.categoryId,product.description,product.price],(err,results)=>{

        if(!err){

            return res.status(200).json({"message":"Product Added Successfully"});

        }

        else{

            return res.status(500).json(err);

        }

        })

})

//---------------------Get the Product API------------------------

router.get('/get',auth.authenticateToken,(req,res)=>{

    var query = "select p.id, p.name, p.description, p.price, p.status, c.id as categoryId, c.name as categoryName from product as p INNER JOIN category as c where p.categoryId = c.id";

    connection.query(query,(err,results)=>{

        if(!err){

            return res.status(200).json(results);

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//---------------------Get Product By category ID API--------------------------------

router.get('/getByCategory/:id',auth.authenticateToken,(req,res,next)=>{

    const id = req.params.id;   //categoryId

    var query = "select id,name from product where categoryId=? and status='true'";

    connection.query(query,[id],(err,results)=>{

        if(!err){

            return res.status(200).json(results);

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//----------------Get Product By product Id------------------------------------

router.get('/getById/:id',auth.authenticateToken,(req,res,next)=>{

    const id = req.params.id;  //product id

    var query = "select id,name,description,price from product where id = ?";

    connection.query(query,[id],(err,results)=>{

        if(!err){

            return res.status(200).json(results[0]);   //we need only one record so we use results[0]. ==>Basically it is select only record but it is in the form of array so.

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//-----------------Update Product API------------------------------------

router.patch('/update',auth.authenticateToken,checkRole.checkRole,(req,res,next)=>{

    let product = req.body;

    var query = "update product set name=?, categoryId=?, description=?, price=? where id=?";

    connection.query(query,[product.name,product.categoryId,product.description,product.price,product.id],(err,results)=>{

        if(!err){

            if(results.affectedRows == 0){

                return res.status(404).json({"message":"Product id does not exist"});

            }

            return res.status(200).json({"message":"Product Updated Successfully"});

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//--------------------Delete product API----------------------------------

router.delete('/delete/:id',auth.authenticateToken,checkRole.checkRole,(req,res,next)=>{

    const id = req.params.id;    //product id

    var query = "delete from product where id=?";

    connection.query(query,[id],(err,results)=>{

        if(!err){

            if(results.affectedRows == 0){

                return res.status(404).json({"message":"Product id does not found"});

            }

            return res.status(200).json({"message":"Product Deleted Successfully"});

        }

        else{

            return res.status(500).json(err);

        }

    })

})

//----------------------Change the Status of product API----------------------

router.patch('/updateStatus',auth.authenticateToken,checkRole.checkRole,(req,res,next)=>{

    let user = req.body;

    var query = "update product set status=? where id=?";

    connection.query(query,[user.status,user.id],(err,results)=>{

        if(!err){

            if(results.affectedRows == 0){  //any row is not changed

                return res.status(404).json({"message":"Product id does not found"});

            }

            return res.status(200).json({"message":"Product Updated Successfully"});

        }

        else{

            return res.status(500).json(err);

        }

    })

})

module.exports = router;

bill.js:

const express = require('express');

const connection = require('../connection');

const router = express.Router();

let ejs = require('ejs');

let pdf = require('html-pdf');

let path = require('path');

var fs = require('fs');

var uuid = require('uuid');

var auth = require('../services/authentication');

//----------------------generate report API-------------------------

router.post('/generateReport', auth.authenticateToken, (req, res) => {

    const generatedUuid = uuid.v1();  //Generate a UUID based on the current time and the MAC address of the Machine

    const orderDetails = req.body;

    var productDetailsReport = JSON.parse(orderDetails.productDetails);  //JSON.parse() it is convert JSON formate to javascript object

    //    "productDetails":"[{\"id\":1, \"name\": \"Black Coffee\", \"price\": 99, \"total\": 99, \"category\": \"Coffee\", \"quantity\": \"1\"}]"

    var query = "insert into bill (name,uuid,email,contactNumber,paymentMethod,total,productDetails,createdBy) values(?,?,?,?,?,?,?,?)";

    connection.query(query, [orderDetails.name, generatedUuid, orderDetails.email, orderDetails.contactNumber, orderDetails.paymentMethod, orderDetails.totalAmount, orderDetails.productDetails, res.locals.email], (err, results) => {

        if (!err) {

            // in the {} we write all the variable value that is used in the report.ejs file

            ejs.renderFile(path.join(\_\_dirname, '', "report.ejs"), { productDetails: productDetailsReport, name: orderDetails.name, email: orderDetails.email, contactNumber: orderDetails.contactNumber, paymentMethod: orderDetails.paymentMethod, totalAmount: orderDetails.totalAmount }, (err, results) => {

                if (err) {

                    return res.status(500).json(err);

                }

                else {

                    pdf.create(results).toFile('./generated\_pdf/' + generatedUuid + ".pdf", function (err, results) {

                        if (err) {

                            console.log(err);

                            return res.status(500).json(err);

                        }

                        else {

                            return res.status(200).json({ uuid: generatedUuid });

                        }

                    })

                }

            })

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//-----Send the uuid and get the pdf if pdf not exist in folder then the create and return API---

router.get('/getPdf', auth.authenticateToken, function (req, res) {

    const orderDetails = req.body;

    const pdfPath = './generated\_pdf/' + orderDetails.uuid + '.pdf';

    if (fs.existsSync(pdfPath)) {

        // return the pdf file

        res.contentType("application/pdf");

        fs.createReadStream(pdfPath).pipe(res);

    }

    else {

        /\*==>In this section we write if given uuid pdf is not exist the first we create pdf and we return \*/

        var productDetailsReport = JSON.parse(orderDetails.productDetails);

        // in the {} we write all the variable value that is used in the report.ejs file

        ejs.renderFile(path.join(\_\_dirname, '', "report.ejs"), { productDetails: productDetailsReport, name: orderDetails.name, email: orderDetails.email, contactNumber: orderDetails.contactNumber, paymentMethod: orderDetails.paymentMethod, totalAmount: orderDetails.totalAmount }, (err, results) => {

            if (err) {

                return res.status(500).json(err);

            }

            else {

                pdf.create(results).toFile('./generated\_pdf/' + orderDetails.uuid + ".pdf", function (err, results) {

                    if (err) {

                        console.log(err);

                        return res.status(500).json(err);

                    }

                    else {

                        // return the pdf file

                        res.contentType("application/pdf");

                        fs.createReadStream(pdfPath).pipe(res);

                    }})}})}})

//--------------------Get Bills API-----------------------------------

router.get('/getBills', auth.authenticateToken, (req, res, next) => {

    var query = "select \* from bill order by id DESC";  //DESC-->decreasing order

    connection.query(query, (err, results) => {

        if (!err) {

            return res.status(200).json(results);

        }

        else {

            return res.status(500).json(err);

        }

    })

})

//--------------------Delete Bill API--------------------------------

router.delete('/delete/:id', auth.authenticateToken, (req, res, next) => {

    const id = req.params.id;

    var query = "delete from bill where id=?";

    connection.query(query, [id], (err, results) => {

        if (!err) {

            if (results.affectedRows == 0) {

                return res.status(404).json({ "message": "Bill id does not found" });

            }

            return res.status(200).json({ "message": "Bill deleted successfully" });

        }

        else {

            return res.status(500).json(err);

        }

    })

})

module.exports = router;

report.ejs:

<html>

    <head>

        <style>

            table{

                font-family: arial, sans-serif;

                border-collapse: collapse;

                width: 100%;

            }

            td,th{

                border: 1px solid #dddddd;

                text-align: left;

                padding: 8px;

            }

        </style>

    </head>

    <body>

        <h3 style="text-align: center;">Cafe Management System</h3>

        <table>

            <tr>

                 <th>Name: <%= name %></th>   <!--name is a javascript variable -->

                 <th>Email: <%= email %></th>

            </tr>

            <tr>

                <th>Contact Number: <%= contactNumber %></th>

                <th>Payment Method: <%= paymentMethod %></th>

            </tr>

        </table>

        <h3>Product Details:</h3>

        <table>

            <tr>

                <th>Name</th>

                <th>Category</th>

                <th>Quantity</th>

                <th>Price</th>

                <th>Sub Total</th>

            </tr>

            <% if(productDetails.length > 0) { %>

                <% productDetails.forEach(product =>{%>

                    <tr>

                        <td><%= product.name %></td>

                        <td><%= product.category %></td>

                        <td><%= product.quantity %></td>

                        <td><%= product.price %></td>

                        <td><%= product.total %></td>

                    </tr>

                    <% }); %>

                    <% } %>

        </table>

        <h3>Total: <%= totalAmount %></h3>

        <h3>Thank You For Visiting. Please Visite Again</h3>

    </body>

</html>

**Frontend:-**

Home:

[home.component.html](http://home.component.html):

<div class="bg-image"></div>

<app-best-seller></app-best-seller>

<div class="wrapper sticky">

  <nav class="navbar-fixed-top">

    <a href="#" class="logo">

      <mat-icon>storefront</mat-icon> Cafe Management System

    </a>

    <ul>

      <li><a (click)="loginAction()">Login</a></li>

      <li><a (click)="signupAction()">Signup</a></li>     <!-- signupAction method -->

      <li><a (click)="forgotPasswordAction()">Forgot Password</a></li>

    </ul>

  </nav>

</div>

<div class="footer" id="signup">

  <h2>All right reserved @Vasu Parsaniya</h2>

</div>

[home.component.ts](http://home.component.ts):

import { Component, OnInit } from '@angular/core';

import { MatDialog, MatDialogConfig } from '@angular/material/dialog';

import { SignupComponent } from '../signup/signup.component';

import { ForgotPasswordComponent } from '../forgot-password/forgot-password.component';

import { LoginComponent } from '../login/login.component';

import { Router } from '@angular/router';

import { UserService } from '../services/user.service';

@Component({

  selector: 'app-home',

  templateUrl: './home.component.html',

  styleUrls: ['./home.component.scss']

})

export class HomeComponent implements OnInit {

  constructor(private dialog:MatDialog,

    private router:Router,

    private userService:UserService) { }

  ngOnInit(): void {

    if(localStorage.getItem('token') != null){

      this.userService.checkToken().subscribe((response:any)=>{

      this.router.navigate(['/cafe/dashboard']);

    },(error:any)=>{

      console.log(error);

    });

    }

  }

  signupAction(){

    // console.log("Test");

    const dialogConfig = new MatDialogConfig();

    //set the size of the dialog

    dialogConfig.width = "550px";

    this.dialog.open(SignupComponent,dialogConfig);

  }

  forgotPasswordAction(){

    const dialogConfig = new MatDialogConfig();

    dialogConfig.width = "550px";

    this.dialog.open(ForgotPasswordComponent,dialogConfig);

  }

  loginAction(){

    const dialogConfig = new MatDialogConfig();

    dialogConfig.width = "550px";

    this.dialog.open(LoginComponent,dialogConfig);

  }

}

[home.component.scss](http://home.component.scss):

\* {

  box-sizing: border-box;

}

/\* Style the body \*/

body {

  font-family: Arial;

  margin: 0;

}

/\* Header/logo Title \*/

.header {

  padding: 60px;

  text-align: center;

  background: #1abc9c;

  color: white;

}

/\* Style the top navigation bar \*/

.navbar {

  display: flex;

  background-color: #333;

}

/\* Style the navigation bar links \*/

.navbar a {

  color: white;

  padding: 14px 20px;

  text-decoration: none;

  text-align: center;

}

/\* Change color on hover \*/

.navbar a:hover {

  background-color: #ddd;

  color: black;

}

/\* Column container \*/

.row {

  display: flex;

  flex-wrap: wrap;

}

/\* Create two unequal columns that sits next to each other \*/

/\* Sidebar/left column \*/

.side {

  flex: 30%;

  background-color: #f1f1f1;

  padding: 20px;

}

/\* Main column \*/

.main {

  flex: 70%;

  background-color: white;

  padding: 20px;

}

/\* Fake image, just for this example \*/

.fakeimg {

  background-color: #aaa;

  width: 100%;

  padding: 20px;

}

/\* Footer \*/

.footer {

  padding: 6px;

  text-align: center;

  color: white;

  background: #e53935;

}

/\* Responsive layout - when the screen is less than 700px wide, make the two columns stack on top of each other instead of next to each other \*/

@media screen and (max-width: 700px) {

  .row, .navbar {

    flex-direction: column;

  }

}

\* {

  box-sizing: border-box;

}

/\* Container for flexboxes \*/

.row {

  display: flex;

  flex-wrap: wrap;

}

/\* Create four equal columns \*/

.column {

  flex: 25%;

  padding: 20px;

}

/\* On screens that are 992px wide or less, go from four columns to two columns \*/

@media screen and (max-width: 992px) {

  .column {

    flex: 50%;

  }

}

body {

  background-color: #384047;

}

.wrapper {

  height: 65px;

  display: flex;

  align-items: center;

  justify-content:center;

  background-color: #e53935;

}

nav {

  display: flex;

  width: 90%;

}

ul li a {

  margin-left: 1.5em;

}

a.logo {

  margin-right: auto;

}

ul {

  display: flex;

  list-style: none;

  margin: 0px;

  padding: 0px;

}

a {

  text-decoration: none;

  color: #ffffff;

  position: relative;

  font-size: 1.25em;

}

a::after {

  content:"";

  top: 80%;

  border-bottom: 2px solid #F8E71C;

  transition: all 0.35s;

  position: absolute;

}

a:hover::after {

  transition: all 0.35s;

}

a::after {

  right: 50%;

  left: 50%;

}

a:hover::after {

  right: 0;

  left: 0;

}

@media (max-width: 1024px) {

  .wrapper {

    height: auto;

  }

  nav {

    flex-direction: column;

    align-items: center;

  }

  a.logo {

    margin-top: 1.5em;

    margin-bottom: 1.5em;

    margin-right: 0px;

  }

  ul {

    width: 100%;

    margin-bottom: 1em;

    justify-content: space-between;

  }

  ul li a {

    margin-left: 0px;

  }

}

@media (max-width: 768px) {

  a.logo {

    margin: 1.5em;

  }

  ul {

    flex-direction: column;

  }

  ul li {

    margin: 0.3em;

    text-align: center;

  }

  ul li a {

    margin-left: 0px;

  }

}

.sticky {

  position: fixed;

  top: 0;

  left: 0;

  width: 100% !important;

}

.mat-icon {

  vertical-align: middle;

}

.bg-image {

  /\* The image used \*/

  background-image: url("../../assets/img/food1.jpg");

  /\* Add the blur effect \*/

  filter: blur(3px);

  -webkit-filter: blur(3px);

  /\* Full height \*/

  height: 100%;

  /\* Center and scale the image \*/

  background-position: center;

  background-repeat: no-repeat;

  background-size: cover;

}

/\* Position text in the middle of the page/image \*/

.bg-text {

  background-color: rgb(0,0,0); /\* Fallback color \*/

  background-color: rgba(0,0,0, 0.8); /\* Black w/opacity/see-through \*/

  color: white;

  font-weight: bold;

  border: 3px solid #f1f1f1;

  position: absolute;

  top: 60%;

  left: 50%;

  transform: translate(-50%, -50%);

  width: 55%;

  padding: 20px;

  text-align: center;

}

Dashboard:

dashboard.component.html:

<body>

  <mat-card>

    <b><span>Dashboard</span></b>

  </mat-card>

  <br>

  <div class="row">

    <div class="column">

      <div class="card">

        <div class="container">

          <h2 class="title">Total Category:</h2>

          <h1 class="title">{{data?.category}}</h1>

          <p><button class="button" [routerLink]="['/cafe/category']">View Category</button></p>

        </div>

      </div>

    </div>

    <div class="column">

      <div class="card">

        <div class="container">

          <h2 class="title">Total Product:</h2>

          <h1 class="title">{{data?.product}}</h1>

          <p><button class="button" [routerLink]="['/cafe/product']">View Product</button></p>

        </div>

      </div>

    </div>

    <div class="column">

      <div class="card">

        <div class="container">

          <h2 class="title">Total Bill:</h2>

          <h1 class="title">{{data?.bill}}</h1>

          <p><button class="button" [routerLink]="['/cafe/bill']">View Bill</button></p>

        </div>

      </div>

    </div>

  </div>

</body>

dashboard.component.ts:

import { Component, AfterViewInit } from '@angular/core';

import { DashboardService } from '../services/dashboard.service';

import { NgxUiLoaderService } from 'ngx-ui-loader';

import { SnackbarService } from '../services/snackbar.service';

import { GlobalConstants } from '../shared/global-constants';

@Component({

  selector: 'app-dashboard',

  templateUrl: './dashboard.component.html',

  styleUrls: ['./dashboard.component.scss']

})

export class DashboardComponent implements AfterViewInit {

  responseMessage:any;

  data:any;

  ngAfterViewInit() { }

  constructor(private dashboardService:DashboardService,

    private ngxService:NgxUiLoaderService,

    private snackbarService:SnackbarService) {

      this.ngxService.start();

      this.dashdoardData();

  }

  dashdoardData(){

    this.dashboardService.getDetails().subscribe((response:any)=>{

      this.ngxService.stop();

      this.data = response;

    },(error:any)=>{

      this.ngxService.stop();

      console.log(error);

      if(error.error?.message){

        this.responseMessage = error.error?.message;

      }

      else{

        this.responseMessage = GlobalConstants.genericError;

      }

      this.snackbarService.openSnackBar(this.responseMessage,GlobalConstants.error);

    })

  }

}

dashboard.component.scss:

.position-relative {

  position: relative;

}

.add-contact {

  position: absolute;

    right: 17px;

    top: 57px;

}

body {

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

    margin: 0;

  }

  html {

    box-sizing: border-box;

  }

  \*, \*:before, \*:after {

    box-sizing: inherit;

  }

  .column {

    float: left;

    width: 33.3%;

    margin-bottom: 16px;

    padding: 0 8px;

  }

  .card {

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    margin: 8px;

  }

  .about-section {

    padding: 50px;

    text-align: center;

    background-color: #474e5d;

    color: white;

  }

  .container {

    padding: 0 16px;

  }

  .container::after, .row::after {

    content: "";

    clear: both;

    display: table;

  }

  .title {

    color: black;

    text-align: center !important;

  }

  .button {

    border: none;

    outline: 0;

    display: inline-block;

    padding: 8px;

    color: white;

    background-color: #e53935;

    text-align: center;

    cursor: pointer;

    width: 100%;

    font-weight: bold;

  }

  .button:hover {

    background-color: #555;

  }

  @media screen and (max-width: 650px) {

    .column {

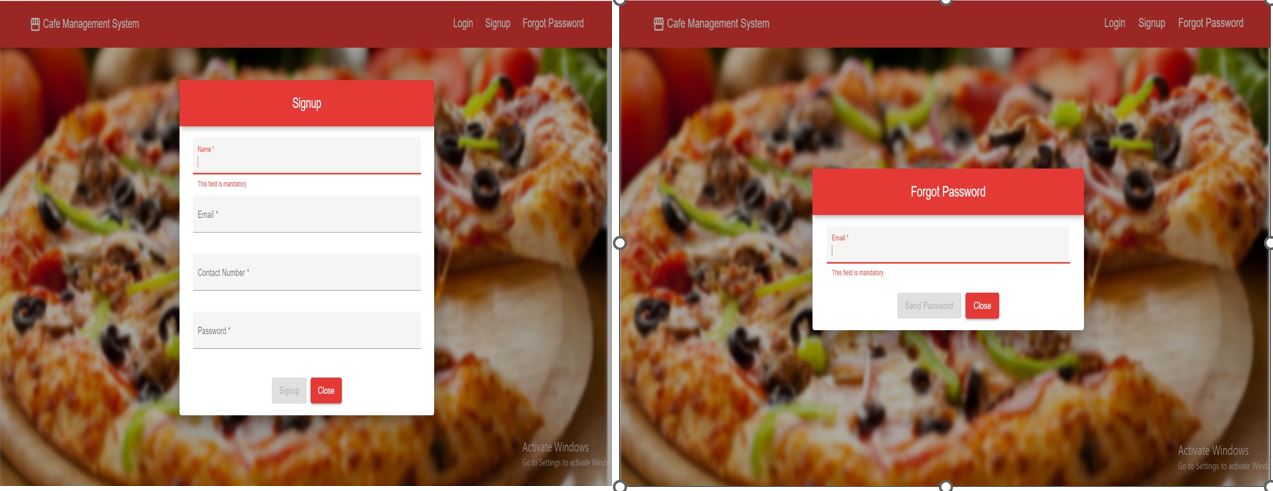
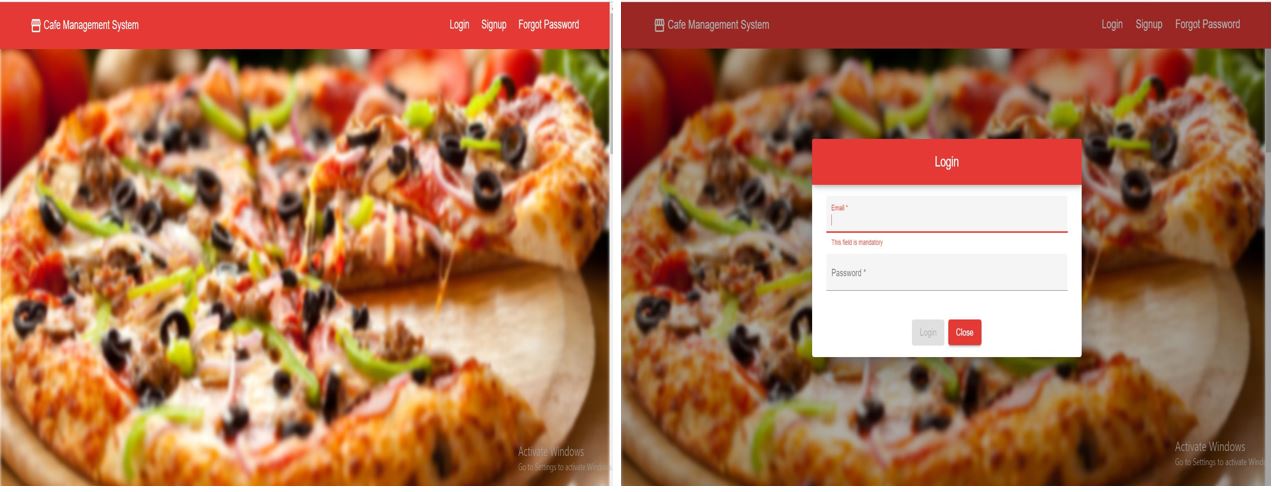
      width: 100%;

      display: block;

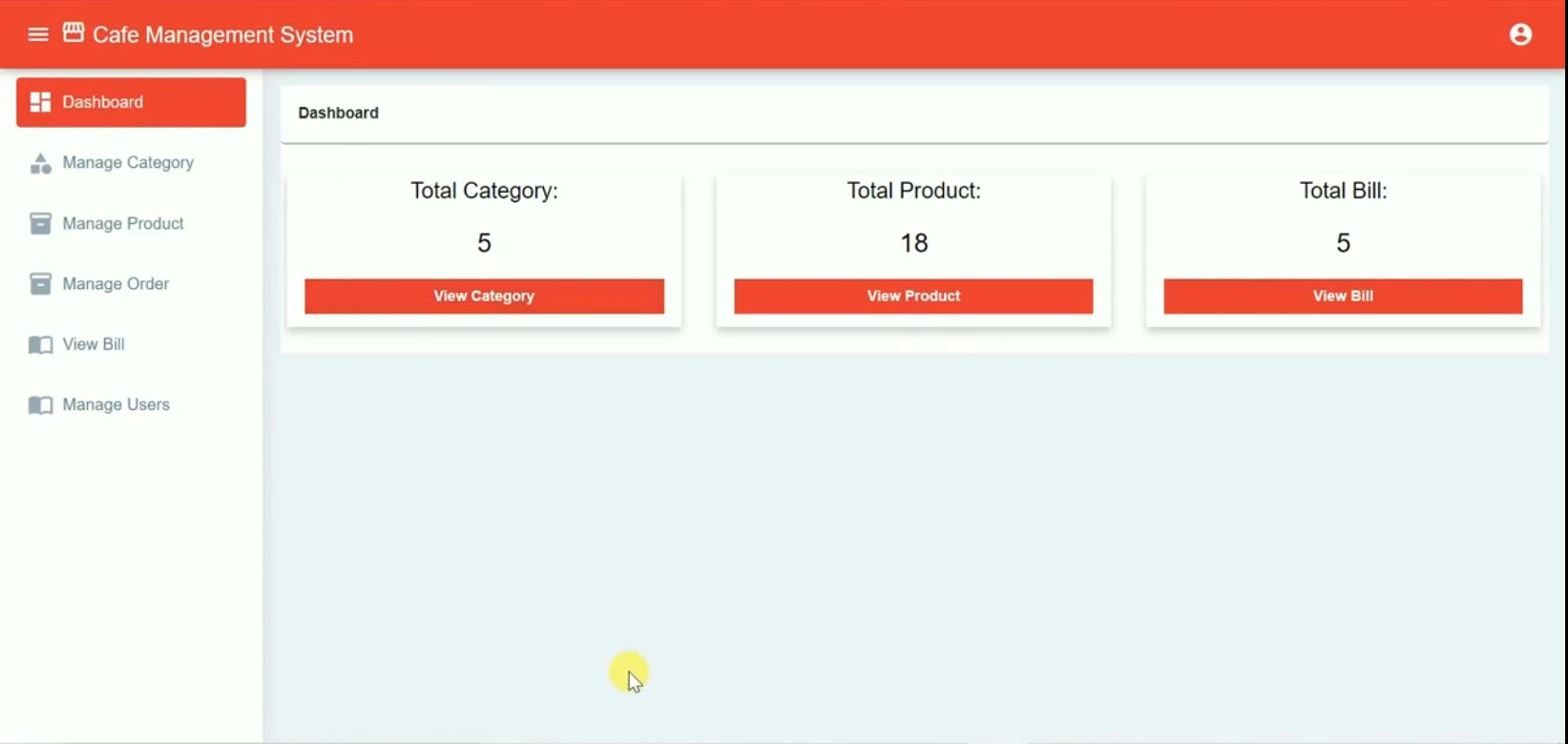
    }

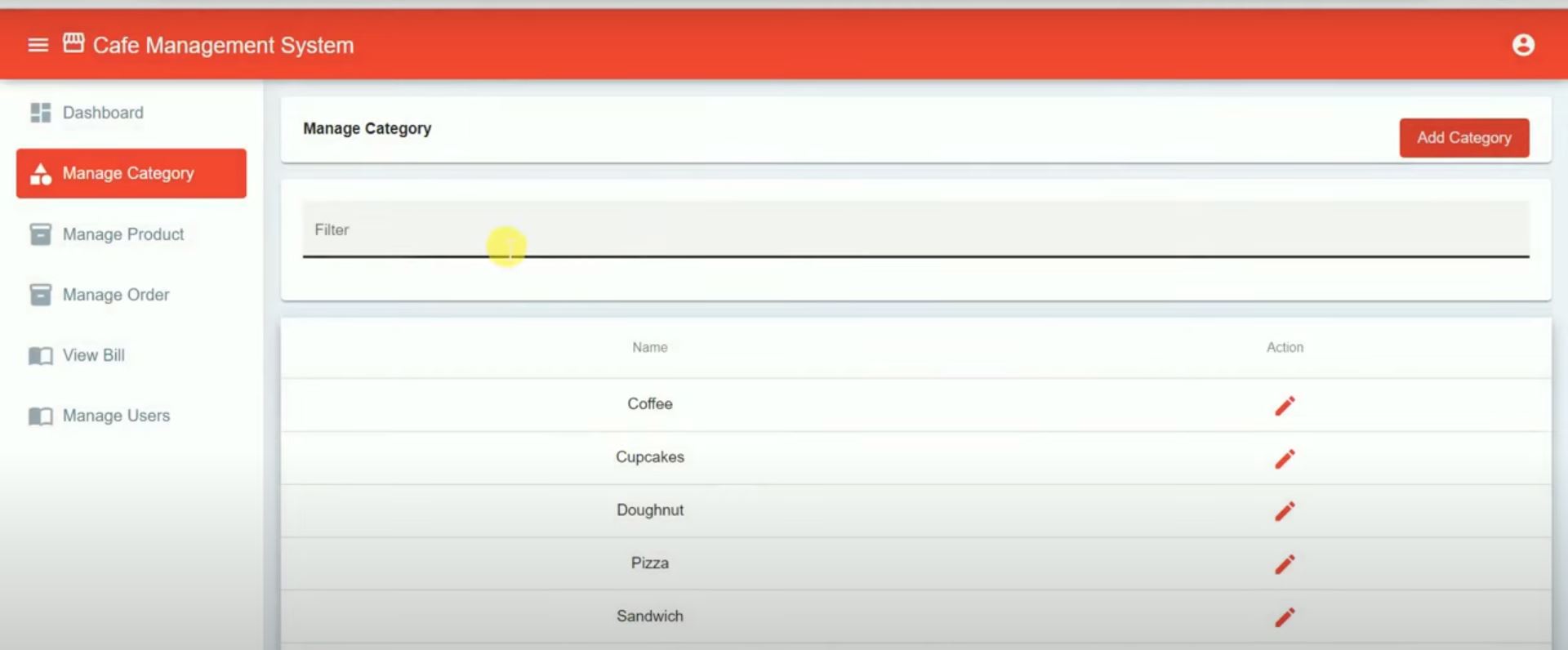
**Output:-**

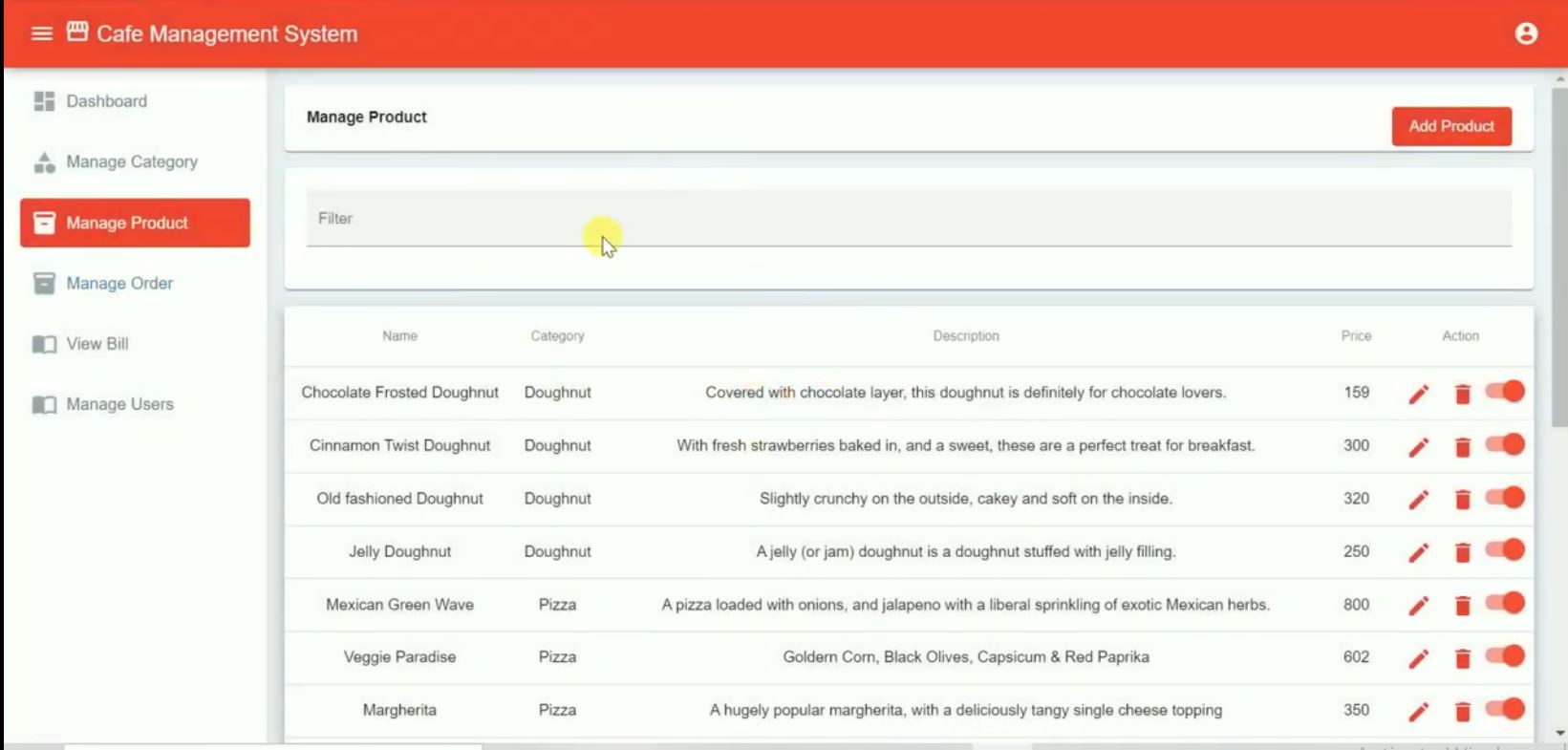
Home Page:

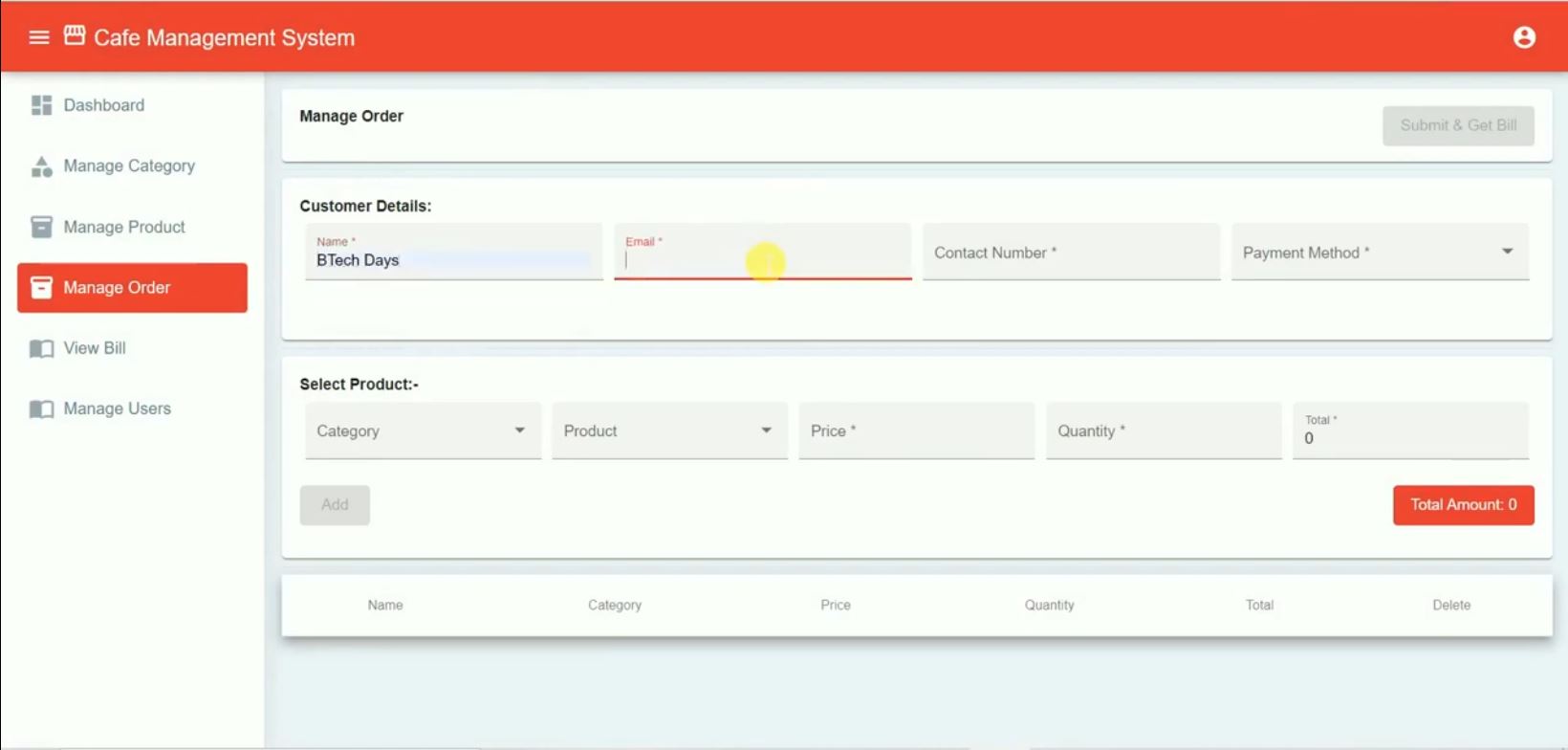


Admin Dashboard:

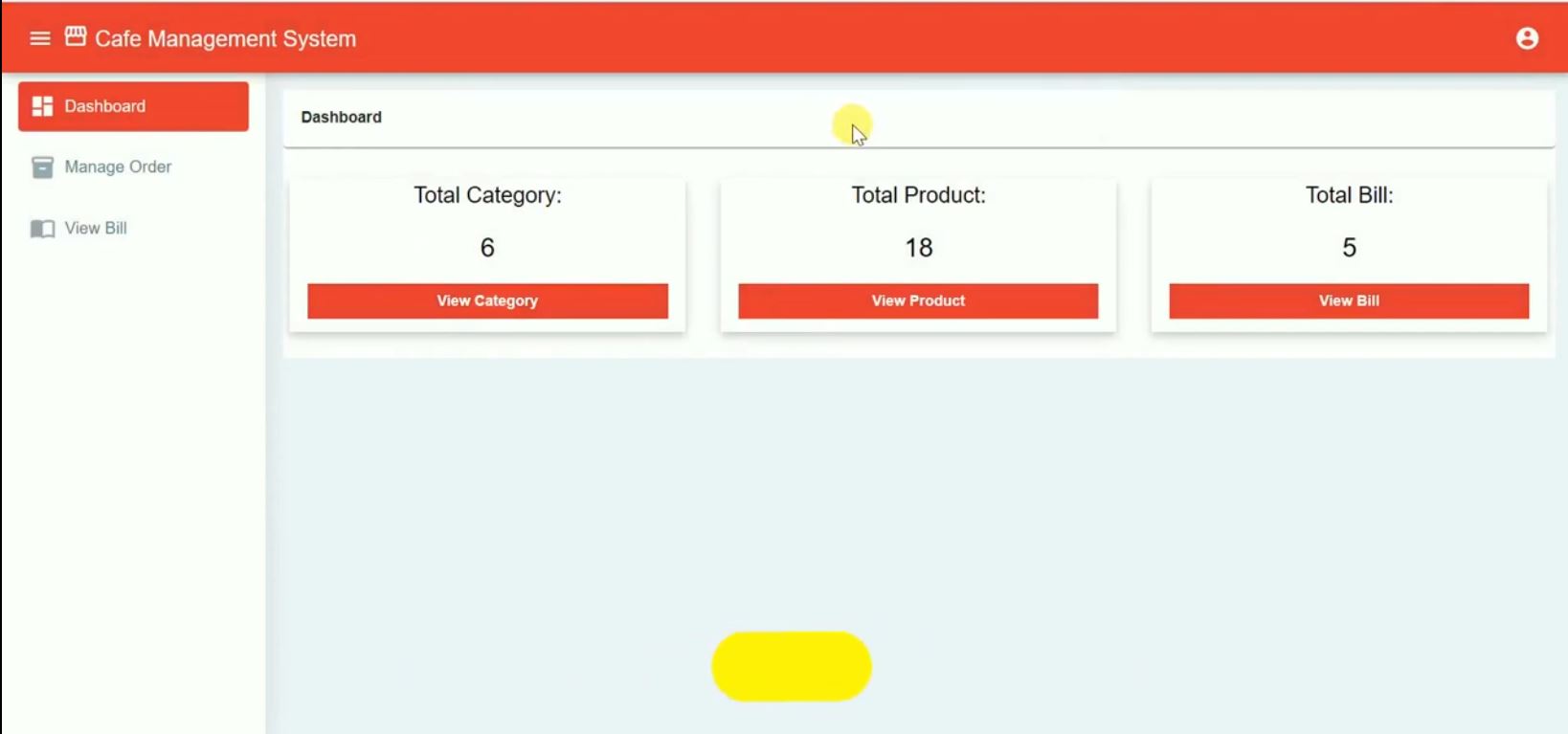




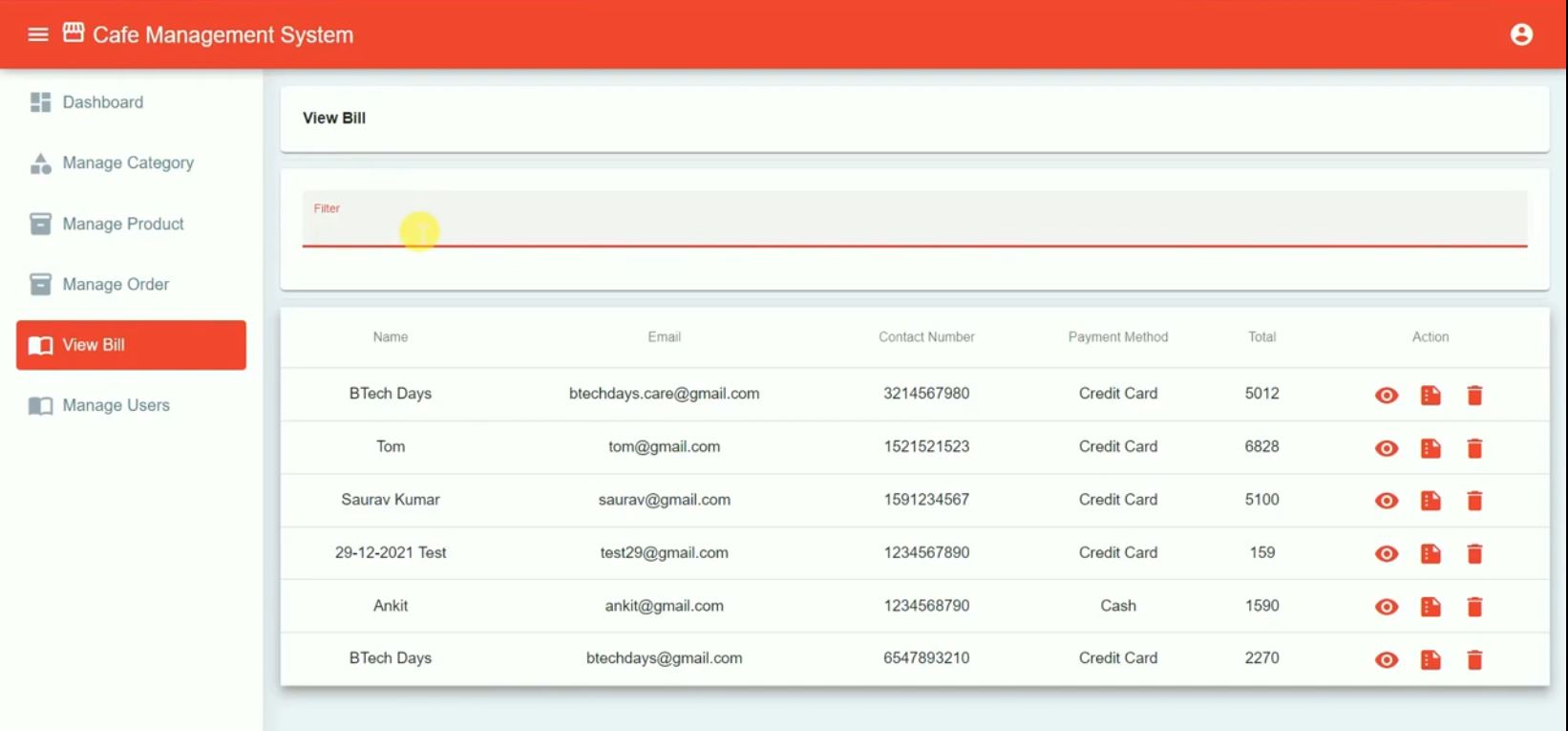




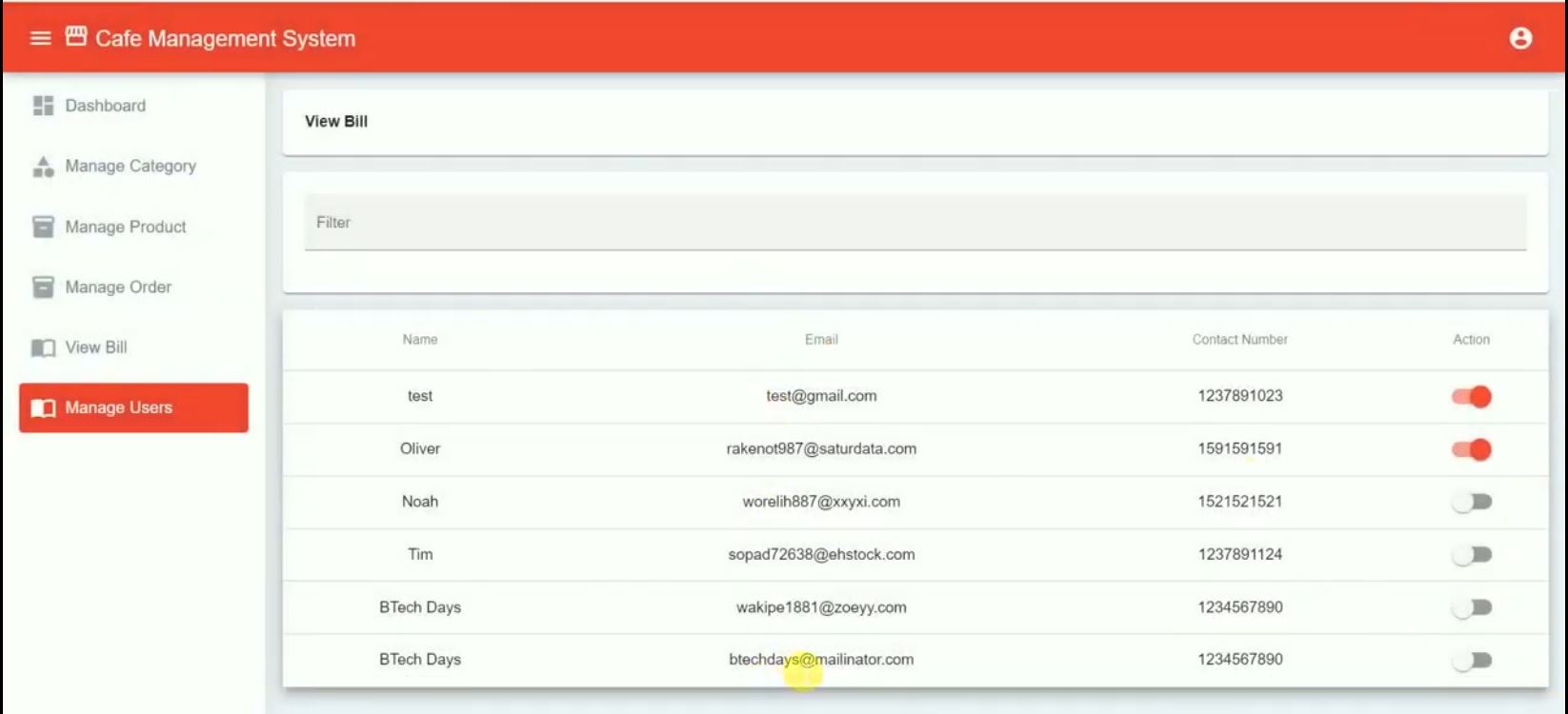
User Dashboard:



View Bill:



Manage Users:



1. Use frames such that page is divided into 3 frames 20% on left to show contents of pages, 60% in center to show body of page, remaining on right to show remarks.
2. Create an HTML form to submit student data for Admission. Upload PDF copy of mark sheet, passport size photo and address proof (Use internal CSS).
3. Create your Resume using HTML tags also experiment with colors, text, link, size and also other tags you studied. (Use external CSS).
4. Write an HTML page that contains a selection box with a list of 5 countries, when the user selects a country, its capital should be printed next to the list; Add CSS to customize the properties of the font of the capital (color, bold and font size).

# JAVA SCRIPT

1. Write a java script program which compute, the average marks of students then this average is used to determine the corresponding grade.
2. Write a Java Script code to display Fibonacci Series of given number. Number should be entered by user through text box.
3. Write a java script for loop that will iterate from 0 to 15 for each iteration, it will check if the current number is odd or even, and display a message to the screen.
4. To design the scientific calculator and make event for each button using java script.
5. Write JavaScript to validate the following fields of the above registration page. Name (Name should contains alphabets and the length should not be less than 6 characters). Password (Password should not be less than 6 characters length).E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com)Phone number (Phone number should contain 10 digits only).

# PHP & MYSQL

1. Write PHP programs
   * Write PHP program to change image automatically using switch case.
   * Write PHP program to calculate current age without using any pre-define function.
   * Write a PHP code to read data from JSON object and display in table form.
   * Create simple login page in PHP. Let user login with predefined username and password. Logout user automatically after 5mins of user login and redirect back to login page.
   * Write PHP code to perform CRUD operation using PHP.

# Advanced Concepts:

1. Develop a web page which contains two list box. First list ask to select State and according to state selection second list box loads name of city. Develop it using AJAX.
2. Create a webpage that hides and show the image when clicked using jQuery.