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 =
ba48f495673ff6fd42356a957ce061076a351e8b2a4702ac42aea5cd7bfa536567bb04dd809bff45ef057f55
06d50798f7551650e7622a8db16546e0c9a6fcf7

#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:

```
<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:

```
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:

```
* {
  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);
    })
  }
}
```

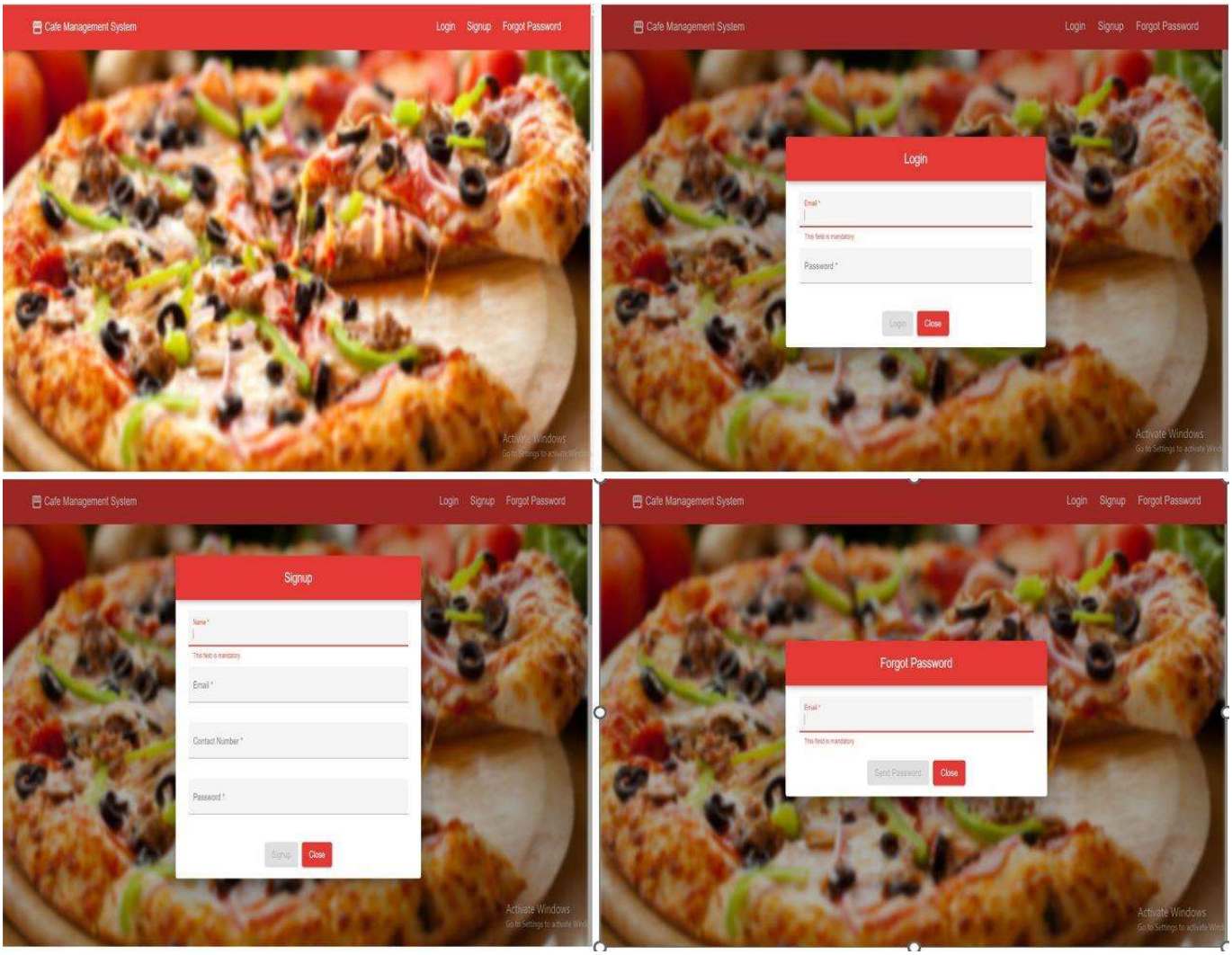
dashboard.component.scss:

```
.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;
  }
```

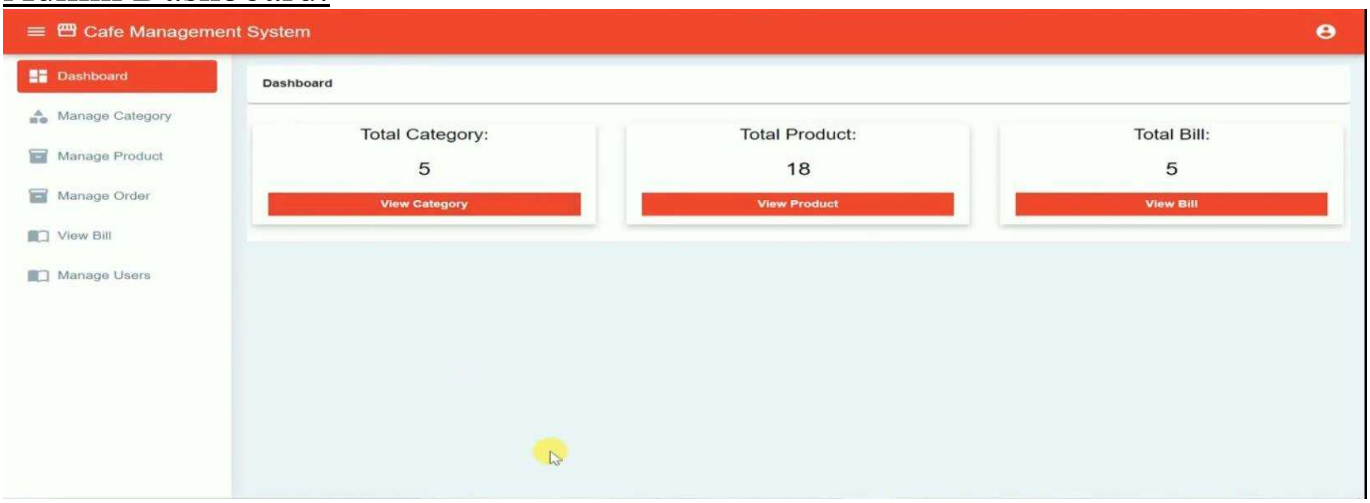
```
.container {
  padding: 0 16px;
}
.container::after, .row::after {
  content: "";
  clear: both;
  display: table;
}
.title {
  color: black;
  text-align: center !important;
}
.button {
  border: none;
  outline: 0;
  display: inline-block;
  padding: 8px;
  color: white;
  background-color: #e53935;
  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;
  }
```

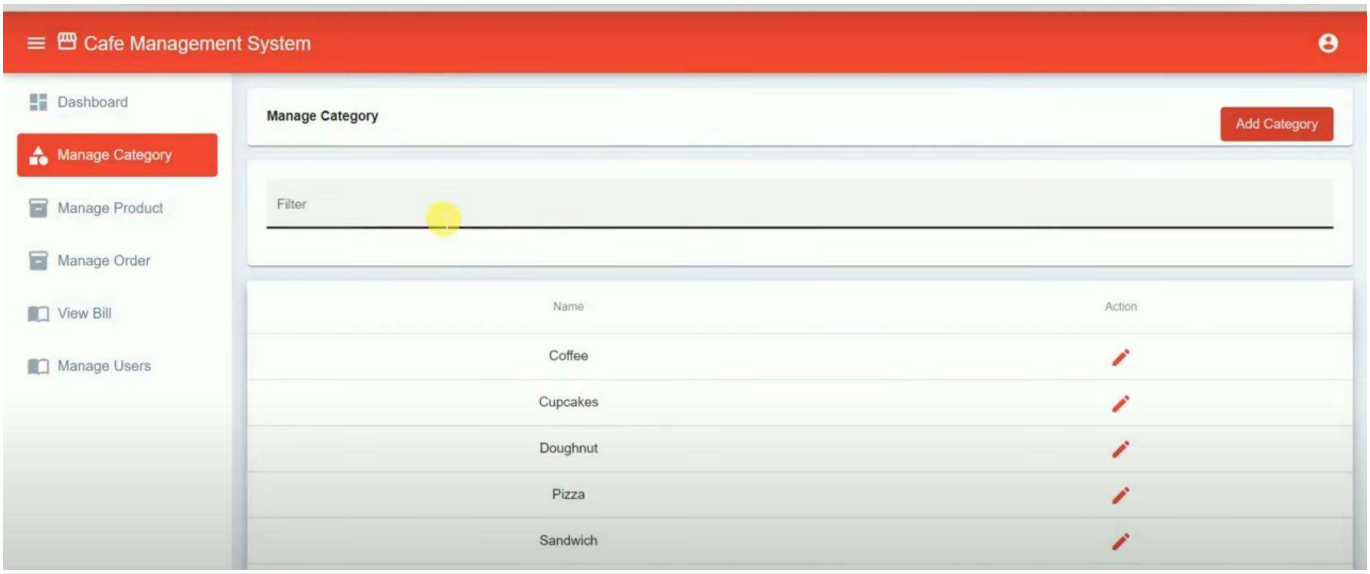
Output:-

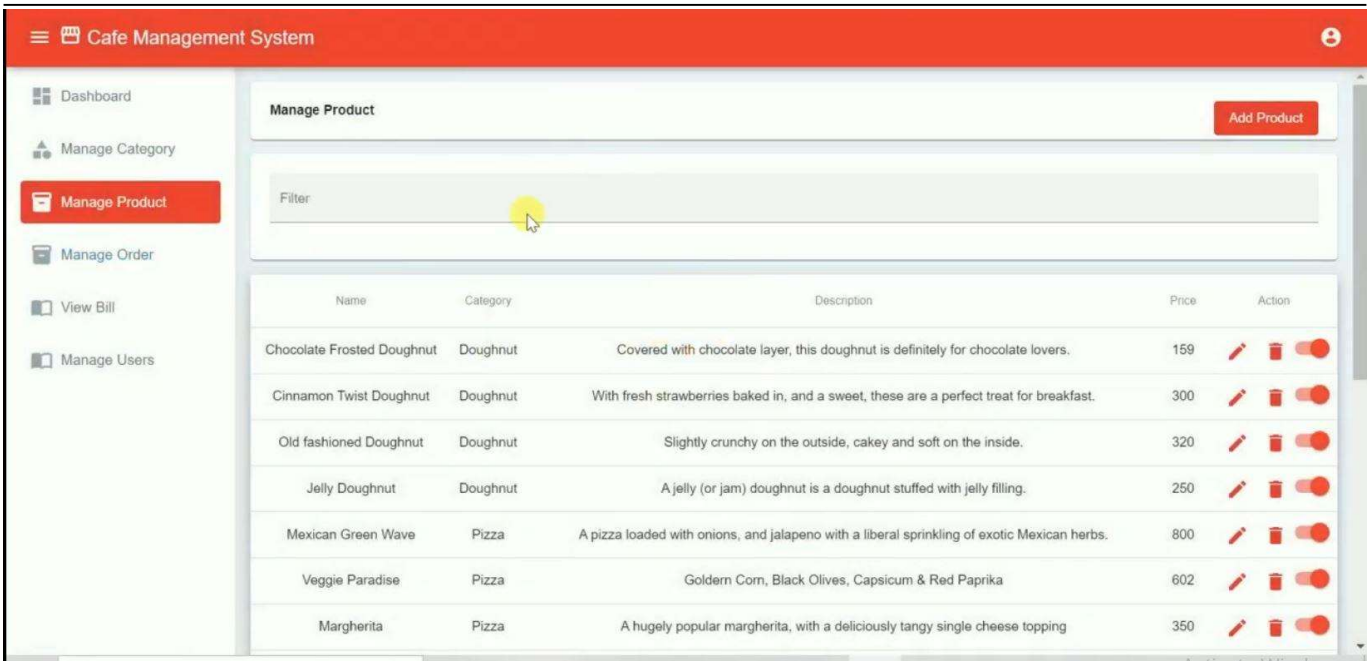
Home Page:

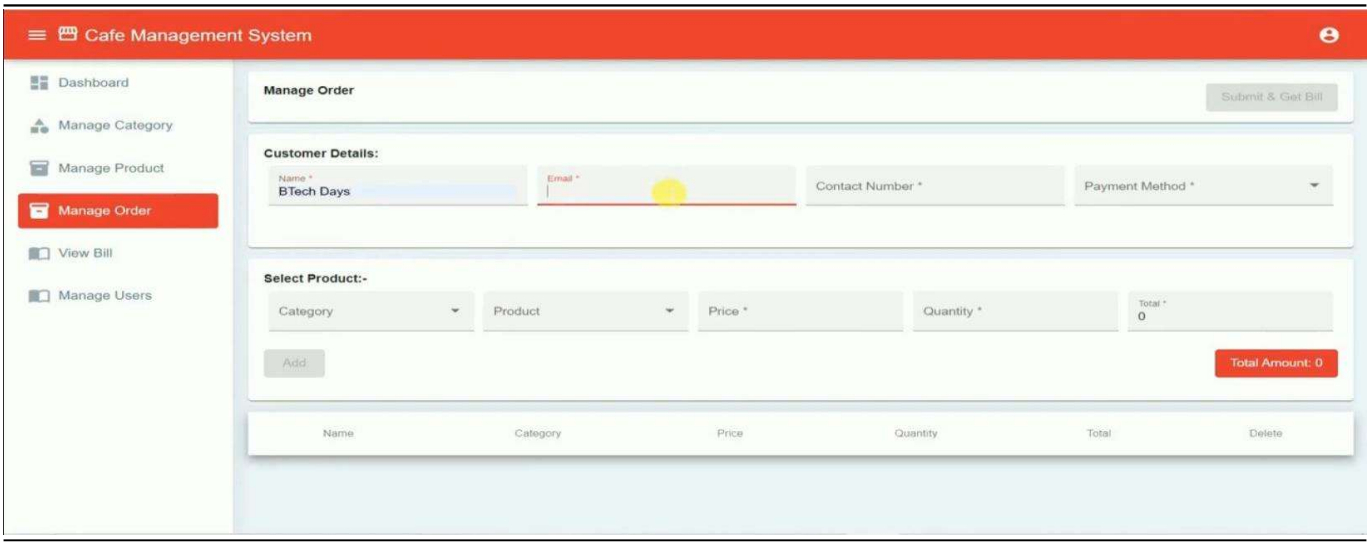


Admin Dashboard:

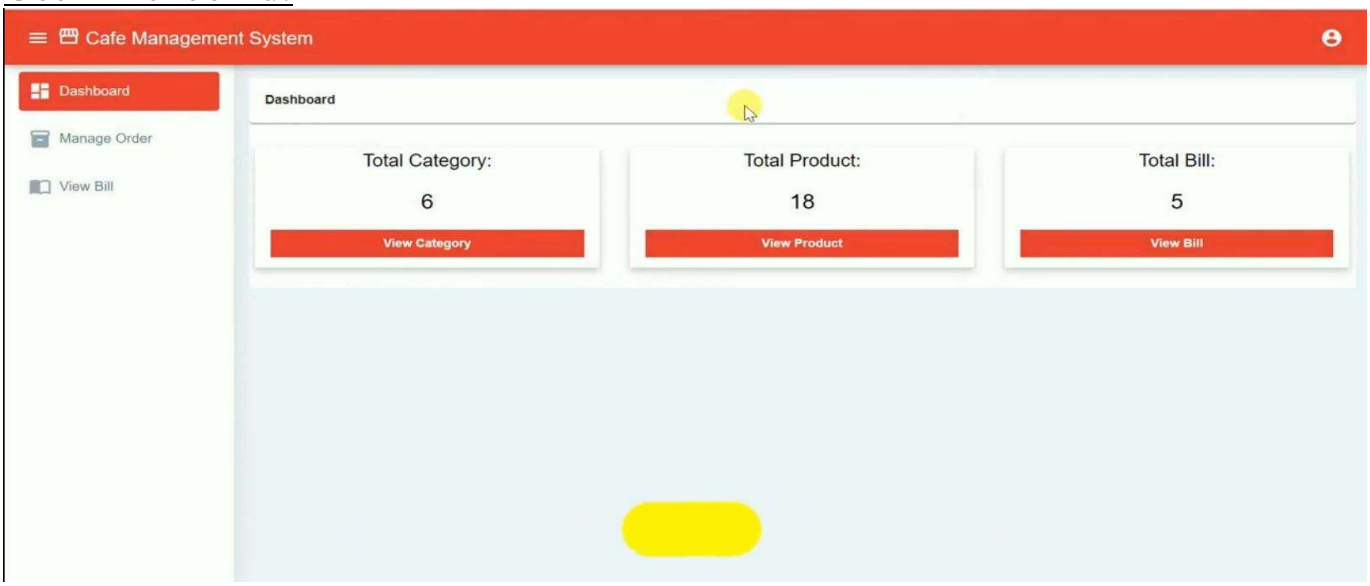



Cafe Management System
Dashboard
Manage Category
Manage Product
Manage Order
View Bill
Manage Users
Manage Category
Add Category
Filter
Name
Action
Coffee
Cupcakes
Doughnut
Pizza
Sandwich

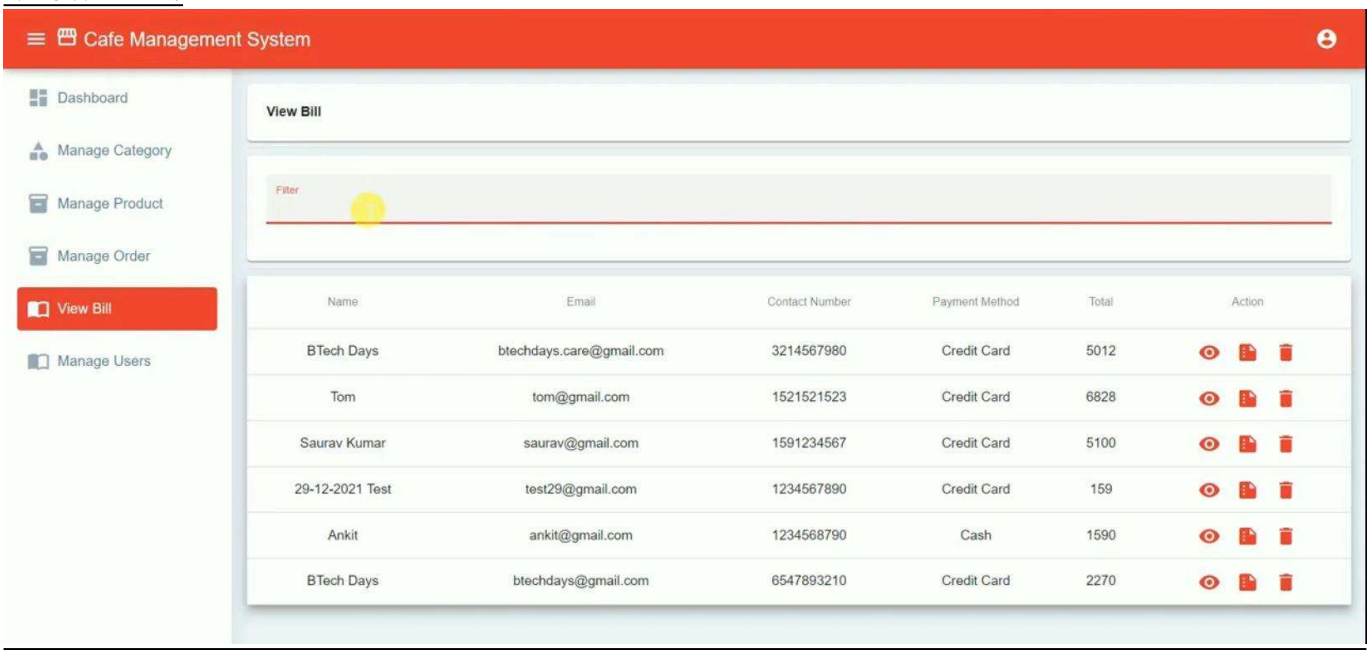

Cafe Management System
Dashboard
Manage Category
Manage Product
Manage Order
View Bill
Manage Users
Manage Product
Add Product
Filter
Name
Category
Description
Price
Action
Chocolate Frosted Doughnut
Doughnut
Covered with chocolate layer, this doughnut is definitely for chocolate lovers.
159
Cinnamon Twist Doughnut
Doughnut
With fresh strawberries baked in, and a sweet, these are a perfect treat for breakfast.
300
Old fashioned Doughnut
Doughnut
Slightly crunchy on the outside, cakey and soft on the inside.
320
Jelly Doughnut
Doughnut
A jelly (or jam) doughnut is a doughnut stuffed with jelly filling.
250
Mexican Green Wave
Pizza
A pizza loaded with onions, and jalapeno with a liberal sprinkling of exotic Mexican herbs.
800
Veggie Paradise
Pizza
Goldern Corn, Black Olives, Capsicum & Red Paprika
602
Margherita
Pizza
A hugely popular margherita, with a deliciously tangy single cheese topping
350


Cafe Management System
Dashboard
Manage Category
Manage Product
Manage Order
View Bill
Manage Users
Manage Order
Submit & Get Bill
Customer Details:
Name *
BTech Days
Email *
Contact Number *
Payment Method *
Select Product:-
Category
Product
Price *
Quantity *
Total *
0
Add
Total Amount: 0
Name
Category
Price
Quantity
Total
Delete

User Dashboard:



View Bill:



Manage Users: