

**BAHRIA UNIVERSITY (KARACHI CAMPUS**)

OPEN ENDED LAB 2 – Fall 2022

(CLO-5)

**Course Title:**  Web Engineering  **Course Code**: SEL 310

**Course Instructor: Engr. Adnan ur Rehman Class**: BSE-5(A/B)

**Lab Engineer:** **Engr. Rehan Baig Name: Adeel**

**Max. Marks:** 6 Marks

**Time: 120** Min

**Note:**

* **The probability of similarity is 0% hence Copied OEL will be marked as ZERO**
* Save Document (Containing Screenshot of Code and Result) of your solution with your name+enrollment like xyz015
* **Must Upload your code folder to Github Public Repository and share your repo URL in doc file.**

**[Marks 6]**

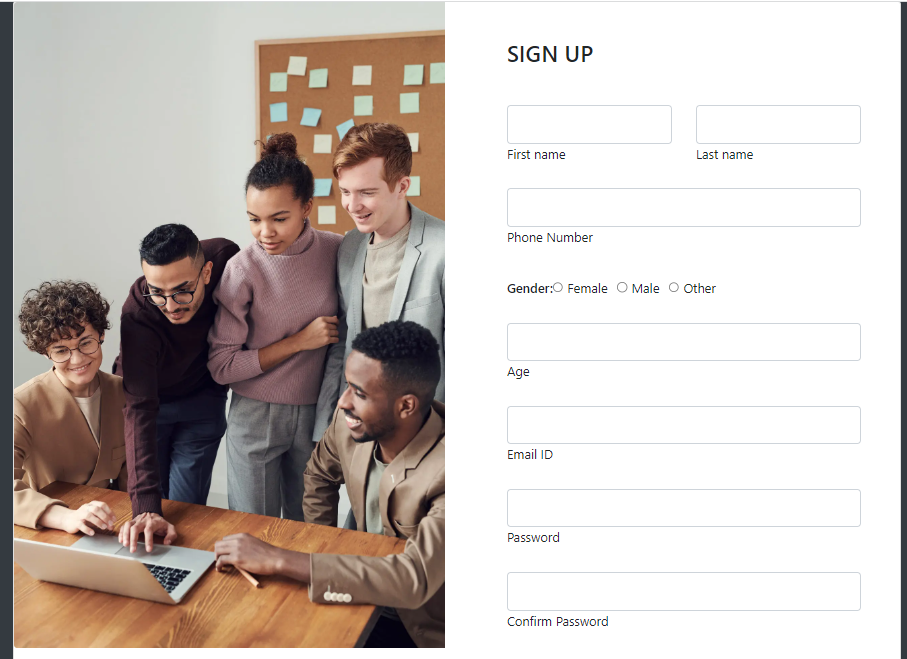
**Scenario:** The **US** based company **Rently** in its second phase of development now wanted to expand its Web SaaS Application to Mobile Devices and also have requirements for low end mobile devices for liteApp. Following are the Requirements

* **Fully Responsive UI (100% mobile optimized) For the Following Pages**
  1. Login
  2. Sign-up
  3. Dashboard
  4. Pay Rent
  5. Collect Rent
  6. Delete Account (With Confirmation – (hint: Use Soft Delete from database just set **isDeleted** check true))
* **Database (you are flexible to implement any database for this application)**
* **REST API’s (Well Commented code for understanding)  
  Following are the requirements for the API’s** 
  1. Use Rest API verbs
  2. APIs contains two types of URL’s one is **Public** **(Any User Can Use)** and other one is **Private** **(Authenticated Users can Access only)**  
     Login/Sign-up Api’s will use public URL’s  
     Dashboard/PayRent/CollectRent/DeleteAccount API’s will use Private URL’s
  3. All Delete Operations for the above API’s are Soft Delete Operations Means Every Delete Operation IsDeleted column will be true against the deleted record but data will not remove.  
       
     **Important Note**  
       
     You as an Engineer responsible for this application design, Storage and implementation phases from scratch with best practices of software engineering.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GOOD LUCK \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Web Engineering OEL Rubrics**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CATEGORY** | **Excellent (1)** | **Good (0.5)** | **Poor (0)** | **Marks** |
| Design | Mobile Pages are properly designed in Html | Mobile Pages are somehow properly designed in Html | Mobile Pages are not properly designed in Html |  |
| Responsiveness | Student uses a style sheet to define attributes (along with some inline style), which makes page attractive looking, | Student uses a style sheet, however there is some inconsistencies in relation to the styles on some elements on the webpage. | Student uses a style sheet; however, it is not consistent on page, which disrupts consistency of page. |  |
| Database Design & Implementation | Tables, Datatype and Naming of Columns are properly defined with best practices | Tables, Datatype and Naming of Columns are averagely defined | Tables, Datatype and Naming of Columns are not defined. Or missing |  |
| Security of Application | Only Authorized Users Can Access Private + Public Routes and All Unauthorized Users Can Access Public Routes only | All Users can Access Private routes and Public Routes | URL’s not reachable |  |
| Standards & Best Practices | Best Practices, Conventions and Standards are properly implemented | Best Practices, Conventions and Standards are somehow implemented | Best Practices, Conventions and Standards are not implemented |  |
| Restful API | Restful APIs are implemented as per standards and verbs | Restful APIs are somehow implemented as per standards and verbs | Restful APIs are not implemented as per standards and verbs |  |
| **Total Marks** = **6** | | | | |



signUpModel

import mongoose from "mongoose";

import validator from "validator";

let signUpModel=mongoose.Schema({

    firstname:{type:String, required:true},

    lastname:{type:String, required:true},

    email:{type:String, required:true,unique:true,lowercase:true ,validate(value){

        if (!validator.isEmail(value)) {

            console.log("Wrong Email input")

            throw new Error("Wrong Email input")

        }

    }},

    gender:{type:String, required:true},

    phoneNumber:{type:Number, required:true},

    age:{type:Number, required:true},

    password:{type:String, required:true},

    confirmpassword:{type:String, required:true},

})

const Register=mongoose.model('SignUp',signUpModel)

export default Register

LoginApis

import express from 'express'

import mongoose from 'mongoose';

import Register from "../Models/signUpModel.mjs";

const router=new express.Router();

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

    try {

        console.log(req.body);

        const register = Register(req.body);

        const createRegister = await register.save();

        res.status(201).send("Send Successfuly")

    } catch (e) {

        res.status(400).send(e)

    }

})

router.get('/registers',async(req,res)=>{

    try {

        const createRegister = await Register.find();

        res.status(200).send(createRegister)

    } catch (e) {

        res.status(400).send(e)

    }

})

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

    try {

        let email=req.body.email

        let password=req.body.password

        const userDetail = await Register.findOne({email:email});

        if (userDetail.password===password) {

        res.status(200).send(userDetail)

        }else{

            res.send("Invalid Auth")

        }

    } catch (e) {

        res.status(400).send("Bad Error")

    }

})

export default router

For Checking Security

  useEffect(() => {

    if (userInfo) {

      history.push(redirect);

    }

  }, [userInfo, history, redirect]);

  const submitHandler = (e) => {

    e.preventDefault();

    dispatch(login(email, password));

  };

For Rent

import express from 'express'

import mongoose from 'mongoose';

const router=new express.Router();

import productModel from "../Models/productModels.mjs";

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

    try {

        console.log(req.body);

        const product = productModel(req.body);

        const createProduct = await product.save();

        res.status(201).send("Send Successfuly")

    } catch (e) {

        res.status(400).send(e)

    }

})

router.get('/products', async (req, res) => {

    try {

        const productData = await productModel.find()

        res.send(productData)

    } catch (error) {res.send(error)}

})

router.get('/product/:id', async (req, res) => {

    try {

        const \_id=req.params.id

        const productData = await productModel.findById(\_id)

        if (!productData) {

            res.send(400).send("require Parameter are  missing")

        } else {

            res.send(productData)

        }

    } catch (error) {res.send(error)}

})

router.delete('/product/:id', async (req, res) => {

    try {

        const \_id=req.params.id

        const productData = await productModel.findByIdAndDelete(\_id)

        if (!\_id) {

            res.send(400).send("require Parameter are  missing")

        } else {

            res.send("Deleted Suceesful")

        }

    } catch (error) {res.status(500).send(error)}

})

router.patch('/product/:id', async (req, res) => {

    try {

        const \_id=req.params.id

        let updateProduct = await productModel.findByIdAndUpdate(req.params.id,req.body,{new:true}).exec();

        console.log(req.body);

        res.send({

            message: "Product is updated successfully",

            data: updateProduct

        })

    } catch (e) {

        res.status(500).send("Internal Error!")

    }

})

export default router;