**Assignment 3-B**

Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations on assignment 2.C.

Code –

1.employeeController.js

var express = require('express') ;

var router = express.Router() ;

var ObjectId = require('mongoose').Types.ObjectId;

var {Employee} = require('../models/employee') ;

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

    Employee.find((*err*, *docs*)=>{

*if*(!*err*){*res*.send(*docs*) ;}

*else*{

            console.log(JSON.stringify(*err*, undefined, 2));

        }

    }) ;

}) ;

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

*if*(!ObjectId.isValid(*req*.params.id))

*return* *res*.status(400).send('No record') ;

    Employee.findById(*req*.params.id, (*err*, *docs*)=>{

*if*(!*err*) {*res*.send(*docs*) ;}

*else*{

            console.log(JSON.stringify(*err*, undefined, 2));

        }

    })

}) ;

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

    var emp = new Employee({

        name:*req*.body.name ,

        position: *req*.body.position ,

        office: *req*.body.office,

        salary: *req*.body.salary

    });

    emp.save((*err*, *docs*)=>{

*if*(!*err*){*res*.send(*docs*) ;}

*else*{

            console.log(JSON.stringify(*err*, undefined, 2));

        }

    });

}) ;

router.put('/:id', (*req*,*res*)=>{

*if*(!ObjectId.isValid(*req*.params.id))

*return* *res*.status(400).send('No record') ;

    var emp = new Employee({

            name:*req*.body.name ,

            position: *req*.body.position ,

            office: *req*.body.office,

            salary: *req*.body.salary

    });

    Employee.findByIdAndUpdate(*req*.params.id,{ $set:emp }, {new:true}, (*err*, *docs*)=>{

*if*(!*err*) {*res*.send(*docs*) ;}

*else*{

            console.log(JSON.stringify(*err*, undefined, 2));

        }

    })

}) ;

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

*if*(!ObjectId.isValid(req.params.id))

*return* res.status(400).send('No record') ;

    var emp = new Employee({

            name:req.body.name ,

            position: req.body.position ,

            office: req.body.office,

            salary: req.body.salary

    });

    Employee.findByIdAndRemove(req.params.id, (*err*, *docs*)=>{

*if*(!err) {res.send(docs) ;}

*else*{

            console.log(JSON.stringify(err, undefined, 2));

        }

    })

}) ;

module.exports = router;

2.employee.js

const mongoose = require('mongoose') ;

var Employee = mongoose.model('Employee', {

   name: {type:String},

   position: {type:String},

   office: {type:String},

   salary: {type:Number},

}) ;

module.exports = {Employee} ;

3.database.js

const mongoose = require('mongoose') ;

const url = require("./config");

const CONNECTION\_URL = url.MONGODB;

mongoose.connect(CONNECTION\_URL, (*err*) => {

*if* (!*err*)

        console.log('MongoDB connection succeeded.');

*else*

        console.log('Error in DB connection : ' + JSON.stringify(*err*, undefined, 2));

});

module.exports = mongoose;

4.index.js

const express = require("express");

const bodyParser = require("body-parser");

const cors = require("cors");

const {mongoose} = require("./database.js");

var employeeController = require('./controllers/employeeController') ;

const app = express();

app.use(bodyParser.json());

app.use(cors({ origin: 'http://localhost:4200' }));

app.listen(3200, () => console.log('Server started at port : 3200'));

app.use('/employees', employeeController);

5.employee.component.html

<div class="container userform">

    <h5>Enter employee information</h5>

    <form #employeeForm="ngForm" (ngSubmit)="onSubmit(employeeForm)">

        <input type="hidden" name="\_id" #\_id="ngModel" [(ngModel)]="employeeService.selectedEmployee.\_id">

        <div>

            <label for="">Name:

                <label for="">\*</label>

            </label>

            <input type="text" name="name" class="form-control" required="true" #name="ngModel" [(ngModel)]="employeeService.selectedEmployee.name" placeholder="Enter name">

            <label for="">Position:

                <label for="">\*</label>

            </label>

            <input type="text" name="position" class="form-control" required="true" #name="ngModel" [(ngModel)]="employeeService.selectedEmployee.position" placeholder="Enter position">

            <label for="">Office:

                <label for="">\*</label>

            </label>

            <input type="text" name="office" class="form-control" required="true" #name="ngModel" [(ngModel)]="employeeService.selectedEmployee.office" placeholder="Enter office">

            <label for="">Salary:

                <label for="">\*</label>

            </label>

            <input type="number" name="salary" class="form-control" required="true" #name="ngModel" [(ngModel)]="employeeService.selectedEmployee.salary" placeholder="Enter salary">

            <div class="input-field col s12">

                <button class="btn btn-primary reset" type="button" (click)="resetForm(employeeForm)">Reset</button>

                <button class="btn btn-primary submit" type="submit" [disabled]="!employeeForm.valid">Submit</button>

              </div>

              <table class="table table-striped">

                <thead>

                  <tr>

                    <th scope="col">Name</th>

                    <th scope="col">Position</th>

                    <th scope="col">Office</th>

                    <th scope="col">Salary</th>

                    <th scope="col"></th>

                  </tr>

                </thead>

                <tr scope="row" \*ngFor="let emp of employeeService.employees">

                  <td>{{emp.name}}</td>

                  <td>{{emp.position}}</td>

                  <td>{{emp.office}}</td>

                  <td>{{emp.salary}}</td>

                  <td>

                      <a class="action-btn" (click)="onEdit(emp)">

                        <i class="material-icons">edit</i>

                      </a>

                      <a class="action-btn" (click)="onDelete(emp.\_id,employeeForm)">

                        <i class="material-icons">delete</i>

                      </a>

                    </td>

                </tr>

              </table>

        </div>

    </form>

</div>

6.employee.component.spec.ts

*import* { ComponentFixture, TestBed } *from* '@angular/core/testing';

*import* { EmployeeComponent } *from* './employee.component';

describe('EmployeeComponent', () => {

  let component: EmployeeComponent;

  let fixture: ComponentFixture<EmployeeComponent>;

  beforeEach(async () => {

*await* TestBed.configureTestingModule({

      declarations: [ EmployeeComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(EmployeeComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

7.employee.component.ts

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

*import* { NgForm } *from* '@angular/forms';

*import* { EmployeeService } *from* '../../shared/employee.service';

*import* { Employee } *from* '../../shared/employee.model' ;

*// import { map } from 'rxjs/operators';*

declare var M: any;

@Component({

  selector: 'app-employee',

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

  styleUrls: ['./employee.component.css'],

  providers: [EmployeeService]

})

*export* class EmployeeComponent implements OnInit {

  constructor(public *employeeService*: EmployeeService) { }

  ngOnInit() {

    this.resetForm();

    this.refreshEmployeeList();

  }

  resetForm(*form*?: NgForm) {

*if* (*form*)

*form*.reset();

    this.employeeService.selectedEmployee = {

      \_id: "",

      name: "",

      position: "",

      office: "",

      salary: null

    }

  }

  onSubmit(*form*: NgForm) {

*if* (*form*.value.\_id == "") {

      this.employeeService.postEmployee(*form*.value).subscribe((*res*) => {

        this.resetForm(*form*);

        this.refreshEmployeeList();

        M.toast({ html: 'Saved successfully', classes: 'rounded' });

      });

    }

*else* {

      this.employeeService.putEmployee(*form*.value).subscribe((*res*) => {

        this.resetForm(*form*);

        this.refreshEmployeeList();

        M.toast({ html: 'Updated successfully', classes: 'rounded' });

      });

    }

  }

  refreshEmployeeList() {

    this.employeeService.getEmployeeList().subscribe((*res*) => {

      this.employeeService.employees = *res* *as* Employee[];

    });

  }

  onEdit(*emp*: Employee) {

    this.employeeService.selectedEmployee = emp;

  }

  onDelete(*\_id*: string, *form*: NgForm) {

*if* (confirm('Are you sure to delete this record ?') == true) {

      this.employeeService.deleteEmployee(\_id).subscribe((*res*) => {

        this.refreshEmployeeList();

        this.resetForm(form);

        M.toast({ html: 'Deleted successfully', classes: 'rounded' });

      });

    }

  }

}

**Output –**

