**Angular project with unit testing –**

**Test.ts**

import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

BrowserDynamicTestingModule,

platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

context(path: string, deep?: boolean, filter?: RegExp): {

keys(): string[];

<T>(id: string): T;

};

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

// Then we find all the tests.

/\*const context = require.context('./', true, /\.spec\.ts$/);\*/

const context = require.context('./', true, /user.component\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

==================================================================

**Authentication.service.ts**

import { Injectable } from '@angular/core';

import { Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';

import { catchError, tap, map } from 'rxjs/operators';

import { LoginModel } from '../login/models/login.model';

import { SessionStorageService } from '../service/session-storage.service';

import { TokenModel } from '../models/token.model';

import { BehaviorSubject } from 'rxjs';

const apiServer = "https://localhost:44371/api/authentication/";

@Injectable({

providedIn: 'root'

})

export class AuthenticationService {

public serviceTimer: Observable<number>;

public userName: BehaviorSubject<string> = new BehaviorSubject<string>("");

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

})

}

constructor(private httpService: HttpClient,

private sessionStorageService: SessionStorageService) {

this.serviceTimer = null;

}

setUserName(userEmail: string) {

this.userName.next(userEmail);

}

getUserName() {

return this.userName;

}

public loginUser(loginModel: LoginModel): Observable<any> {

//const url = apiServer + "login";

//return this.http.post<any>(url, JSON.stringify(loginModel)).pipe(

// map(response => {

// return response;

// }),

// catchError(error => {

// return error;

// })

//

const url = apiServer + "login";

return this.httpService.post<any>(url, JSON.stringify(loginModel));

}

refreshToken(refreshToken: string): Observable<any> {

const url = apiServer + "refreshtoken";

return this.httpService.post<any>(url, JSON.stringify(refreshToken));

}

validateUserAuthentication() {

const session = this.sessionStorageService.getServiceTokenModel();

var accessToken = session.accessToken;

var refreshToken = session.refreshToken;

if (accessToken != null && refreshToken != null) {

return true;

}

else {

return false;

}

}

setTokenFromResponse(data: any) {

const expireDate = new Date(data.accessTokenExpiry);

const utcDate = new Date();

const startTime = Date.UTC(utcDate.getUTCFullYear(), utcDate.getUTCMonth(),

utcDate.getUTCDate(), utcDate.getUTCHours(), utcDate.getUTCMinutes(),

utcDate.getUTCSeconds(), utcDate.getUTCMilliseconds());

const expireTime = Date.UTC(expireDate.getUTCFullYear(), expireDate.getUTCMonth(),

expireDate.getUTCDate(), expireDate.getUTCHours(), expireDate.getUTCMinutes(),

expireDate.getUTCSeconds(), expireDate.getUTCMilliseconds());

const model = new TokenModel();

model.accessTokenExpiryTime = (expireTime - startTime);

model.accessToken = data.accessToken;

model.refreshToken = data.refreshToken;

this.sessionStorageService.setServiceToken(model);

}

}

==================================================================

**Authentication.service.spec.ts**

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

import { RouterTestingModule } from '@angular/router/testing';

import { AuthenticationService } from '../../app/service/authentication.service';

import { HttpClientModule, HttpClient } from '@angular/common/http'

import { SessionStorageService } from '../../app/service/session-storage.service';

import { BehaviorSubject, of } from 'rxjs';

import { LoginModel } from '../login/models/login.model';

import { TokenModel } from '../models/token.model';

describe('AuthenticationService', () => {

let authenticationService: AuthenticationService;

beforeEach(() => {

TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [],

providers: [AuthenticationService, SessionStorageService, HttpClient]

});

authenticationService = TestBed.get(AuthenticationService);

});

it('should be created', () => {

expect(authenticationService).toBeTruthy();

});

it('should setUserName', () => {

const userName: BehaviorSubject<string> = new BehaviorSubject<string>("abc");

authenticationService.setUserName("abc");

expect(authenticationService.userName).toEqual(userName);

});

it('should getUserName', () => {

const userName: BehaviorSubject<string> = new BehaviorSubject<string>("abc");

authenticationService.setUserName("abc");

var result = authenticationService.getUserName();

expect(result).toEqual(userName);

});

it('Should return response from loginUser', inject([AuthenticationService, SessionStorageService, HttpClient],

(authenticationService, sessionStorage, httpClient) => {

var loginModel: LoginModel = {

userName: 'userName',

password: 'password'

};

var response = {

"status": {

"message": "",

"code": "200"

},

"data": {

"accessToken": "access\_token",

"refreshToken": "refresh\_token",

"accessTokenExpiryTime": "expiry\_time",

}

};

spyOn(httpClient, 'post').and.returnValue(of(response));

authenticationService.loginUser(loginModel).subscribe((response) => {

expect(response.data.accessToken).toEqual('access\_token');

expect(response.data.refreshToken).toEqual('refresh\_token');

expect(response.data.accessTokenExpiryTime).toEqual('expiry\_time');

expect(response.status.code).toEqual("200");

});

}));

it('Should return response from refreshToken', inject([AuthenticationService, SessionStorageService, HttpClient],

(authenticationService, sessionStorageService, httpClient) => {

var response = {

"status": {

"message": "",

"code": "200"

},

"data": {

"accessToken": "access\_token",

"refreshToken": "refresh\_token",

"accessTokenExpiryTime": "expiry\_time",

}

};

spyOn(httpClient, 'post').and.returnValue(of(response));

authenticationService.refreshToken(response.data.refreshToken).subscribe((response) => {

expect(response.data.accessToken).toEqual('access\_token');

expect(response.data.refreshToken).toEqual('refresh\_token');

expect(response.data.accessTokenExpiryTime).toEqual('expiry\_time');

expect(response.status.code).toEqual("200");

});

}));

it('Should validate authentication token', inject([AuthenticationService, SessionStorageService], (authenticationService, sessionStorageService) => {

var token: TokenModel = { accessToken: "accessToken", accessTokenExpiryTime: 12, refreshToken: "refreshToken" };

spyOn(sessionStorageService, 'getServiceTokenModel').and.returnValue(token);

var result = authenticationService.validateUserAuthentication();

expect(result).toEqual(true);

}));

it('Should setToken from response', inject([AuthenticationService, SessionStorageService], (authenticationService, sessionStorageService) => {

var token: TokenModel = { accessToken: "accessToken", accessTokenExpiryTime: 12, refreshToken: "refreshToken" };

authenticationService.setTokenFromResponse(token);

expect(sessionStorage.getItem("accessToken")).toEqual(token.accessToken);

expect(sessionStorage.getItem("refreshToken")).toEqual(token.refreshToken);

expect(sessionStorage.length).toEqual(3);

}));

});

=================================================================

**Session-storage.service.ts**

import { Injectable } from '@angular/core';

import { Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';

import { catchError, tap, map } from 'rxjs/operators';

import { LoginModel } from '../login/models/login.model';

import { TokenModel } from '../models/token.model';

const apiServer = "https://localhost:44371/api/authentication/";

@Injectable({

providedIn: 'root'

})

export class SessionStorageService {

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

})

}

constructor(private http: HttpClient) { }

setServiceToken(data: any) {

sessionStorage.setItem("accessToken", data.accessToken);

sessionStorage.setItem("refreshToken", data.refreshToken);

sessionStorage.setItem("accessTokenExpiryTime", data.accessTokenExpiryTime.toString());

}

getAccessToken() {

return sessionStorage.getItem("accessToken");

}

getUserFromSession() {

debugger;

var user = sessionStorage.getItem("currentUser");

if (user != null) {

return user;

}

return null;

}

getServiceTokenModel() {

const tokenModel = new TokenModel();

tokenModel.accessToken = sessionStorage.getItem("accessToken");

tokenModel.refreshToken = sessionStorage.getItem("refreshToken");

const expireTime = sessionStorage.getItem("accessTokenExpiryTime");

if (expireTime) {

tokenModel.accessTokenExpiryTime = Number(expireTime);

}

else {

tokenModel.accessTokenExpiryTime = 300000;

}

return tokenModel;

}

}

===================================================================

**Sesson-storage.service.spec.ts**

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

import { RouterTestingModule } from '@angular/router/testing';

import { HttpClientModule, HttpClient } from '@angular/common/http'

import { SessionStorageService } from '../../app/service/session-storage.service';

import { TokenModel } from '../models/token.model';

import { UserInfo } from '../../app/models/user.model';

describe('SessionStorageService', () => {

let sessionStorageService: SessionStorageService;

beforeEach(() => {

TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [],

providers: [SessionStorageService]

});

sessionStorageService = TestBed.get(SessionStorageService);

});

it('should be created', () => {

expect(sessionStorageService).toBeTruthy();

});

it('should setServiceToken to session storage', () => {

var token: TokenModel = { accessToken: "accessToken", accessTokenExpiryTime: 12, refreshToken: "refreshToken" };

sessionStorageService.setServiceToken(token);

expect(sessionStorage.getItem("accessToken")).toEqual(token.accessToken);

expect(sessionStorage.getItem("refreshToken")).toEqual(token.refreshToken);

expect(sessionStorage.getItem("accessTokenExpiryTime")).toEqual(token.accessTokenExpiryTime.toString());

});

it('should getAccessToken from session storage', () => {

var token: TokenModel = { accessToken: "accessToken", accessTokenExpiryTime: 12, refreshToken: "refreshToken" };

sessionStorageService.setServiceToken(token);

var result = sessionStorageService.getAccessToken();

expect(result).toEqual(token.accessToken);

});

it('should getUser from session storage', () => {

var userInfo: UserInfo = { userId: 1, userName: 'chaitali', firstName: 'test', lastName: 'lastName' };

sessionStorage.setItem("currentUser", JSON.stringify(userInfo));

var result = sessionStorageService.getUserFromSession();

const sessionUser = JSON.parse(result);

expect(sessionUser.userName).toEqual(userInfo.userName);

expect(sessionUser.userId).toEqual(userInfo.userId);

expect(sessionUser.firstName).toEqual(userInfo.firstName);

expect(sessionUser.lastName).toEqual(userInfo.lastName);

});

it('should getServiceTokenModel from session storage', () => {

var token: TokenModel = { accessToken: "accessToken", accessTokenExpiryTime: 1200, refreshToken: "refreshToken" };

sessionStorageService.setServiceToken(token);

var result = sessionStorageService.getServiceTokenModel();

expect(result.accessToken).toEqual(token.accessToken);

expect(result.accessTokenExpiryTime).toEqual(token.accessTokenExpiryTime);

expect(result.refreshToken).toEqual(token.refreshToken);

});

});

===================================================================

**Employee.service.ts**

import { Injectable } from '@angular/core';

import { Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';

import { catchError, tap, map } from 'rxjs/operators';

import { EmployeeModel } from '../models/employee.model';

const apiServer = "https://localhost:44371/api/employee/";

@Injectable({

providedIn: 'root'

})

export class EmployeeService {

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

})

}

constructor(private http: HttpClient) { }

//getEmployeelist(): Observable<any> {

// /\*const url = apiServer + "getEmployeeList";\*/

// //return this.http.get(url).pipe(

// // map(response => {

// // return response;

// // }),

// // catchError(error => {

// // return error;

// // })

// //);

//}

//saveEmployee(employee: EmployeeModel): Observable<any> {

// const url = apiServer + "saveEmployee";

// //return this.http.post<any>(url, JSON.stringify(employee)).pipe(

// // map(response => {

// // return response;

// // }),

// // catchError(error => {

// // return error;

// // })

// //);

//}

getEmployeelist(): Observable<any> {

const url = apiServer + "getEmployeeList";

return this.http.get(url);

}

getEmployeeById(id: number): Observable<any> {

const url = apiServer + "getEmployeeById/" + id;

return this.http.get(url);

}

saveEmployee(employee: EmployeeModel): Observable<any> {

const url = apiServer + "saveEmployee";

return this.http.post(url, JSON.stringify(employee));

}

updateEmployee(employee: EmployeeModel): Observable<any> {

const url = apiServer + "updateEmployee";

return this.http.put(url, JSON.stringify(employee));

}

deleteEmployee(id: number): Observable<any> {

const url = apiServer + "deleteEmployee/" + id;

return this.http.delete(url);

}

}

================================================================

**Employee.service.spec.ts**

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

import { EmployeeService } from '../../app/service/employee.service';

import { HttpClientModule, HttpClient } from '@angular/common/http'

import { of } from 'rxjs';

import { EmployeeModel } from '../models/employee.model';

fdescribe('AuthenticationService', () => {

let employeeService: EmployeeService;

beforeEach(() => {

TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [],

providers: [EmployeeService, HttpClient]

});

employeeService = TestBed.get(EmployeeService);

});

fit('should be created', () => {

expect(employeeService).toBeTruthy();

});

fit('Should return response from getEmployeelist', inject([EmployeeService, HttpClient],

(employeeService, httpClient) => {

var responseTemp = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

]

;

spyOn(httpClient, 'get').and.returnValue(of(responseTemp));

employeeService.getEmployeelist().subscribe((response) => {

expect(responseTemp.length).toEqual(response.length);

});

}));

fit('Should return response from getEmployeeById', inject([EmployeeService, HttpClient],

(employeeService, httpClient) => {

var responseTemp = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

]

;

spyOn(httpClient, 'get').and.returnValue(of(responseTemp));

employeeService.getEmployeeById(1).subscribe((response) => {

expect(responseTemp.length).toEqual(response.length);

});

}));

fit('Should return response from saveEmployee', inject([EmployeeService, HttpClient],

(employeeService, httpClient) => {

var responseTemp = 1;

var employeeModel: EmployeeModel = {

empId: 1, empName: 'abc', empaddress:'address',empEmailId:'email'

}

spyOn(httpClient, 'post').and.returnValue(of(responseTemp));

employeeService.saveEmployee(employeeModel).subscribe((response) => {

expect(responseTemp).toEqual(response);

});

}));

fit('Should return response from updateEmployee', inject([EmployeeService, HttpClient],

(employeeService, httpClient) => {

var responseTemp = 1;

var employeeModel: EmployeeModel = {

empId: 1, empName: 'abc', empaddress: 'address', empEmailId: 'email'

}

spyOn(httpClient, 'put').and.returnValue(of(responseTemp));

employeeService.updateEmployee(employeeModel).subscribe((response) => {

expect(responseTemp).toEqual(response);

});

}));

fit('Should return response from deleteEmployee', inject([EmployeeService, HttpClient],

(employeeService, httpClient) => {

var responseTemp = 1;

var id = 1;

spyOn(httpClient, 'delete').and.returnValue(of(responseTemp));

employeeService.deleteEmployee(id).subscribe((response) => {

expect(responseTemp).toEqual(response);

});

}));

});

==================================================================

**User.service.ts**

import { Injectable } from '@angular/core';

import { Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';

import { catchError, tap, map } from 'rxjs/operators';

const apiServer = "https://localhost:44371/api/user/";

@Injectable({

providedIn: 'root'

})

export class UserService {

httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

})

}

constructor(private http: HttpClient) {

}

getUserDetails(userName: string): Observable<any> {

//const url = apiServer + "getUserDetails/" + userName;

//return this.http.get(url).pipe(

// map(response => {

// return response;

// }),

// catchError(error => {

// return error;

// })

//);

const url = apiServer + "getUserDetails/" + userName;

return this.http.get(url);

}

getUserList(): Observable<any> {

const url = apiServer + "getUserList";

return this.http.get(url);

}

}

=================================================================

**User.service.spec.ts**

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

import { EmployeeService } from '../../app/service/employee.service';

import { HttpClientModule, HttpClient } from '@angular/common/http'

import { of } from 'rxjs';

import { EmployeeModel } from '../models/employee.model';

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

import { UserInfo } from '../models/user.model';

fdescribe('UserService', () => {

let userService: UserService;

beforeEach(() => {

TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [],

providers: [UserService, HttpClient]

});

userService = TestBed.get(UserService);

});

fit('should be created', () => {

expect(userService).toBeTruthy();

});

fit('Should return response from getUserDetails', inject([UserService, HttpClient],

(userService, httpClient) => {

var responseTemp: any =

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

};

spyOn(httpClient, 'get').and.returnValue(of(responseTemp));

userService.getUserDetails(responseTemp.userName).subscribe((response) => {

expect(responseTemp.length).toEqual(response.length);

});

}));

fit('Should return response from getUserList', inject([UserService, HttpClient],

(userService, httpClient) => {

var responseTemp: any[] = [

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

},

{

"userId": 2,

"userName": "abc@gmail.com",

"firstName": "abc",

"lastName": "mno"

}

];

spyOn(httpClient, 'get').and.returnValue(of(responseTemp));

userService.getUserList().subscribe((response) => {

expect(responseTemp.length).toEqual(response.length);

});

}));

});

================================================================

**Login.component.ts**

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

import { LoginModel } from '../login/models/login.model';

import { AuthenticationService } from '../service/authentication.service';

import { SessionStorageService } from '../service/session-storage.service';

import { Router, ActivatedRoute, ParamMap } from '@angular/router';

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

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

loginModel: LoginModel = new LoginModel();

constructor(private authenticationService: AuthenticationService,

private sessionStorageService: SessionStorageService,

private router: Router,

private userService: UserService) { }

ngOnInit(): void {

}

login() {

this.authenticationService.setUserName(this.loginModel.userName);

this.authenticationService.loginUser(this.loginModel).subscribe((response) => {

if (response != null) {

this.authenticationService.setTokenFromResponse(response);

this.getUserDetails(this.loginModel.userName);

}

}, (error) => {

console.log(error);

})

}

getUserDetails(emailId: string) {

this.userService.getUserDetails(emailId).subscribe((result: any) => {

sessionStorage.setItem("currentUser", JSON.stringify(result));

this.router.navigate(['']);

}, (error) => {

alert(error);

console.log(error);

});

}

}

==================================================================

**Login.component.spec.ts**

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

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

import { AuthenticationService } from '../service/authentication.service';

import { SessionStorageService } from '../service/session-storage.service';

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

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

import { of } from 'rxjs';

import { LoginModel } from './models/login.model';

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { RouterTestingModule } from '@angular/router/testing';

fdescribe('LoginComponent', () => {

let component: LoginComponent;

let fixture: ComponentFixture<LoginComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [HttpClientModule, RouterTestingModule],

declarations: [LoginComponent],

providers: [LoginComponent,

AuthenticationService,

SessionStorageService,

UserService,

HttpClient

]

})

.compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(LoginComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

it('Should return response from login', inject([LoginComponent, AuthenticationService, SessionStorageService, HttpClient, UserService],

(loginComponent, authenticationService, sessionStorageService, httpClient, userService) => {

const loginModelTemp: LoginModel = {

userName: 'chaitali.narkhede1991@gmail.com',

password: 'abc'

}

var responseTemp = {

"accessToken": "access\_token",

"refreshToken": "refresh\_token",

"accessTokenExpiryTime": "expiry\_time",

};

var responseUserTemp: any =

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

};

spyOn(authenticationService, 'setUserName').and.callThrough();

spyOn(authenticationService, 'loginUser').and.returnValue(of(responseTemp));

spyOn(authenticationService, 'setTokenFromResponse').and.callThrough();

spyOn(loginComponent, 'getUserDetails').withArgs(loginModelTemp.userName).and.callThrough();

spyOn(userService, 'getUserDetails').and.returnValue(of(responseUserTemp));

component.loginModel = loginModelTemp;

component.login();

expect(authenticationService.setUserName).toHaveBeenCalledWith(loginModelTemp.userName);

expect(authenticationService.setTokenFromResponse).toHaveBeenCalledWith(responseTemp);

expect(userService.getUserDetails).toHaveBeenCalledWith(loginModelTemp.userName);

}));

it('Should return response from getUserDetails', inject([LoginComponent, AuthenticationService, SessionStorageService, HttpClient, UserService],

(loginComponent, authenticationService, sessionStorageService, httpClient, userService) => {

var responseTemp: any =

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

};

const emailId = 'chaitali.narkhede1991@gmail.com';

spyOn(userService, 'getUserDetails').withArgs(emailId).and.returnValue(of(responseTemp));

component.getUserDetails(emailId);

expect(sessionStorage.getItem("currentUser").length).toBeGreaterThan(0);

sessionStorage.clear();

}));

it('Should return exception from login', inject([LoginComponent, AuthenticationService, SessionStorageService, HttpClient],

(loginComponent, authenticationService, sessionStorageService, httpClient) => {

const loginModelTemp: LoginModel = {

userName: 'chaitali.narkhede1991@gmail.com',

password: 'abc'

}

var responseTemp = {

"Message": "Server Error",

};

spyOn(authenticationService, 'setUserName');

spyOn(authenticationService, 'loginUser').and.throwError("Error: Server Error");

//expect(sessionStorage.length).toEqual(0);

}));

it('Should return exception from getUserDetails', inject([LoginComponent, AuthenticationService, SessionStorageService, HttpClient, UserService],

(loginComponent, authenticationService, sessionStorageService, httpClient, userService) => {

var responseTemp: any =

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

};

spyOn(userService, 'getUserDetails').withArgs(responseTemp.userName).and.throwError("Error: Server Error");

expect(sessionStorage.getItem("currentUser")).toEqual(null);

}));

});

==================================================================

**Employee.component.ts**

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

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

import { EmployeeModel } from '../../models/employee.model';

@Component({

selector: 'app-employee',

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

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

})

export class EmployeeComponent implements OnInit {

public empList: EmployeeModel[] = [];

public employee: EmployeeModel = new EmployeeModel();

public isCreateView: boolean = false;

constructor(private employeeService: EmployeeService) { }

ngOnInit(): void {

this.getEmployeeList();

}

getEmployeeList() {

this.isCreateView = false;

this.employeeService.getEmployeelist().subscribe((res: any) => {

this.empList = res;

}, (error) => {

console.log(error);

});

}

createEmployee() {

this.isCreateView = true;

this.employee = new EmployeeModel();

}

editData(emp: any) {

this.isCreateView = true;

this.employee = new EmployeeModel();

this.employee = emp;

}

saveEmployee() {

if (this.employee.empId == undefined || this.employee.empId <= 0) {

this.employee.empId = 0;

this.addEmployee(this.employee);

}

else {

this.updateEmployee(this.employee);

}

}

addEmployee(employee: EmployeeModel) {

this.employeeService.saveEmployee(employee).subscribe((res: any) => {

if (res > 0) {

alert("Record Saved");

}

this.getEmployeeList();

this.isCreateView = false;

}, (error) => {

console.log(error);

});

}

updateEmployee(employee: EmployeeModel) {

this.employeeService.updateEmployee(employee).subscribe((res: any) => {

if (res > 0) {

alert("Record Updated");

}

this.getEmployeeList();

this.isCreateView = false;

}, (error) => {

console.log(error);

});

}

deleteEmployee(emp: EmployeeModel) {

this.isCreateView = false;

this.employeeService.deleteEmployee(emp.empId).subscribe((res: any) => {

if (res > 0) {

alert("Record Deleted");

}

this.getEmployeeList();

}, (error) => {

console.log(error);

});

}

cancel() {

this.isCreateView = false;

}

}

==================================================================

**Employee.component.spec.ts**

import { HttpClient, HttpClientModule } from '@angular/common/http';

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

import { of } from 'rxjs/internal/observable/of';

import { EmployeeModel } from '../../models/employee.model';

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

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

fdescribe('EmployeeComponent', () => {

let component: EmployeeComponent;

let fixture: ComponentFixture<EmployeeComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [EmployeeComponent],

providers: [EmployeeComponent,

EmployeeService,

HttpClient

]

})

.compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(EmployeeComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

fit('should create', () => {

expect(component).toBeTruthy();

});

fit('should call ngOnInIt', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

var responseTemp = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

];

spyOn(component, 'getEmployeeList').and.callThrough();

spyOn(employeeService, 'getEmployeelist').and.returnValue(of(responseTemp));

component.ngOnInit();

expect(component.empList).toEqual(responseTemp);

}));

fit('should call getEmployeeList', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

var responseTemp = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

];

spyOn(employeeService, 'getEmployeelist').and.returnValue(of(responseTemp));

component.getEmployeeList();

expect(component.empList).toEqual(responseTemp);

expect(component.empList.length).toBeGreaterThan(0);

}));

fit('should call createEmployee', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

component.createEmployee();

expect(component.isCreateView).toEqual(true);

}));

fit('should call editData', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

var emp: EmployeeModel = {

empId: 1,

empName: 'test',

empaddress: 'addr',

empEmailId:'email'

}

component.editData(emp);

expect(component.isCreateView).toEqual(true);

expect(component.employee).toEqual(emp);

}));

fit('should call saveEmployee ', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

const flag = 1;

var emp: EmployeeModel = {

empId: 0,

empName: 'test',

empaddress: 'addr',

empEmailId: 'email'

}

component.employee = emp;

spyOn(component, 'addEmployee');

spyOn(employeeService, 'saveEmployee').withArgs(emp).and.returnValue(of(flag));

component.saveEmployee();

expect(component.addEmployee).toHaveBeenCalled();

}));

fit('should call saveEmployee -> addEmployee', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

const flag = 1;

var emp: EmployeeModel = {

empId: 0,

empName: 'test',

empaddress: 'addr',

empEmailId: 'email'

}

var responseEmployeeList = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

];

component.employee = emp;

spyOn(component, 'addEmployee').withArgs(emp).and.callThrough();

spyOn(employeeService, 'saveEmployee').withArgs(emp).and.returnValue(of(flag));

spyOn(component, 'getEmployeeList').and.callThrough();

spyOn(employeeService, 'getEmployeelist').and.returnValue(of(responseEmployeeList));

component.saveEmployee();

expect(employeeService.saveEmployee).toHaveBeenCalledWith(emp);

expect(component.getEmployeeList).toHaveBeenCalled();

expect(component.empList).toEqual(responseEmployeeList);

expect(component.empList.length).toBeGreaterThan(0);

}));

fit('should call saveEmployee -> updateEmployee', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

const flag = 1;

var emp: EmployeeModel = {

empId: 1,

empName: 'test',

empaddress: 'addr',

empEmailId: 'email'

}

var responseEmployeeList = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

];

component.employee = emp;

spyOn(component, 'updateEmployee').withArgs(emp).and.callThrough();

spyOn(employeeService, 'updateEmployee').withArgs(emp).and.returnValue(of(flag));

spyOn(component, 'getEmployeeList').and.callThrough();

spyOn(employeeService, 'getEmployeelist').and.returnValue(of(responseEmployeeList));

component.saveEmployee();

expect(employeeService.updateEmployee).toHaveBeenCalledWith(emp);

expect(component.getEmployeeList).toHaveBeenCalled();

expect(component.empList).toEqual(responseEmployeeList);

expect(component.empList.length).toBeGreaterThan(0);

}));

fit('should call deleteEmployee', inject([EmployeeComponent, EmployeeService, HttpClient],

(employeeComponent, employeeService, httpClient) => {

const flag = 1;

var emp: EmployeeModel = {

empId: 1,

empName: 'test',

empaddress: 'addr',

empEmailId: 'email'

}

var responseEmployeeList = [

{

"empId": 1,

"empName": "Chaitali narkhede",

"empEmailId": "chaitali.narkhede@gmail.com",

"empaddress": "Pune"

}

];

spyOn(employeeService, 'deleteEmployee').withArgs(emp.empId).and.returnValue(of(flag));

spyOn(component, 'getEmployeeList').and.callThrough();

spyOn(employeeService, 'getEmployeelist').and.returnValue(of(responseEmployeeList));

component.deleteEmployee(emp);

expect(employeeService.deleteEmployee).toHaveBeenCalledWith(emp.empId);

expect(component.getEmployeeList).toHaveBeenCalled();

expect(component.empList).toEqual(responseEmployeeList);

expect(component.empList.length).toBeGreaterThan(0);

}));

});

==================================================================

**User.component.ts**

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

import { UserInfo } from '../../models/user.model';

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

@Component({

selector: 'app-user',

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

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

})

export class UserComponent implements OnInit {

public userList: UserInfo[] = [];

constructor(private userService: UserService) { }

ngOnInit(): void {

this.getUserList();

}

getUserList() {

this.userService.getUserList().subscribe((res: any) => {

this.userList = res;

}, (error) => {

console.log(error);

});

}

}

==================================================================

**User.component.spec.ts**

import { HttpClient, HttpClientModule } from '@angular/common/http';

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

import { of } from 'rxjs';

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

import { UserComponent } from './user.component';

fdescribe('UserComponent', () => {

let component: UserComponent;

let fixture: ComponentFixture<UserComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [HttpClientModule],

declarations: [UserComponent],

providers: [UserComponent,

UserService,

HttpClient

]

})

.compileComponents();

});

beforeEach(() => {

fixture = TestBed.createComponent(UserComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

fit('should create', () => {

expect(component).toBeTruthy();

});

fit('should call ngOnInIt', inject([UserComponent, UserService, HttpClient],

(userComponent, userService, httpClient) => {

spyOn(component, 'getUserList');

component.ngOnInit();

expect(component.getUserList).toHaveBeenCalled();

}));

fit('should call getUserList', inject([UserComponent, UserService, HttpClient],

(userComponent, userService, httpClient) => {

var responseTemp: any[] = [

{

"userId": 1,

"userName": "chaitali.narkhede@gmail.com",

"firstName": "Chaitali",

"lastName": "Narkhede"

},

{

"userId": 2,

"userName": "abc@gmail.com",

"firstName": "abc",

"lastName": "mno"

}

];

spyOn(userService, 'getUserList').and.returnValue(of(responseTemp));

component.getUserList();

expect(userService.getUserList).toHaveBeenCalled();

expect(component.userList).toEqual(responseTemp);

}));

});