**MEAN Stack Example**

I am going to create an Angular 11 CRUD App using Bootstrap 4. I will be using Node.js, Express.js, MongoDB for backend and Angular for handling frontend.

For the demo purpose, I will create an employee management system using Angular 11 MEAN stack. I will try to cover the essential topic used in CRUD web application development.

In this MEAN stack tutorial, I will share step by step process to build an Angular 11 CRUD (Create, Read, Update, Delete) app from scratch.

**Let us understand what does MEAN stack means.**

* **Mongo DB** – It’s an open-source NoSQL cross-platform document-oriented database.
* **Express JS** – It’s a web-based application framework work with Node JS, It helps to build web apps and RESTful APIs.
* **Angular** – Its a TypeScript based complete front-end framework developed by Google team.
* **Node JS** – It is a free JavaScript run time environment, It executes JavaScript code outside of a browser. It is available for MacOS, Windows, Linux and Unix.

**I will be using following plugins and tools to create MEAN Stack app.**

* [Node JS](https://nodejs.org/en/)
* [MongoDB](https://www.mongodb.com/)
* [Mongoose JS](https://mongoosejs.com/)
* [Express JS](https://expressjs.com/)
* [Angular CLI 7.2.3](https://cli.angular.io/)
* [Visual Studio Code](https://code.visualstudio.com/)

**Angular 11 MEAN Stack Example**

* Setup Node JS
* Build a Node.JS Backend
* Connect MongoDB Database
* Create Model
* Create Express RESTful APIs
* Creae MEAN Stack Project
* Add MEAN Routing
* Create Angular Service
* Add Data Object
* Show Data List and Delete Object
* Edit Data Data

**#1 Setup Node JS development environment**

Follow this link to [set up Node JS in your system.](https://nodejs.org/en/download/)

**#2 Build a Node.JS Backend**

To write the manageable code, we should keep the MEAN Stack backend folder separate. Create a folder by the name of the **backend** in Angular’s root directory. This folder will handle the backend code of our application, remember it will have the separate **node\_modules** folder from Angular.

**mkdir backend**

Enter the below command to get into the backend folder.

**cd backend**

Now you are inside the **backend** folder, run the below command to create the package.json file. This file will have the meta data of your MEAN Stack app, It is also known as the manifest file of any NodeJS project.

**npm init -y**

**– Install and Configure required NPM packages for MEAN Stack app development**

Use the below command to install the following node modules.

**npm install --save body-parser cors express mongoose**

**body-parser:** The body-parser npm module is a JSON parsing middleware. It helps to parse the JSON data, plain text or a whole object.

**CORS:** This is a Node JS package, also known as the express js middleware. It allows enabling CORS with multiple options. It is available through the npm registry.

**Express.js:** Express js is a free open source Node js web application framework. It helps in creating web applications and RESTful APIs.

**Mongoose:** Mongoose is a MongoDB ODM for Node. It allows you to interact with MongoDB database.

**– Install and Configure nodemon for MEAN Stack app development**

Starting a server every time a change is made is a time-consuming task. To get rid of this problem we use nodemon npm module. This package restarts the server automatically every time we make a change. We’ll be installing it locally by using given below command.

**npm install nodemon --save-dev**

Now, go within the backend folder’s root, create a file by the name of server.js.

**echo > server.js**

or **create the file “server.js”** manually under backend folder

Now within the backend > server.js file add the given below code.

let express = require('express'),

path = require('path'),

mongoose = require('mongoose'),

cors = require('cors'),

bodyParser = require('body-parser'),

dbConfig = require('./database/db');

// Connecting with mongo db

mongoose.Promise = global.Promise;

mongoose.connect(dbConfig.db, {

useNewUrlParser: true

}).then(() => {

console.log('Database sucessfully connected')

},

error => {

console.log('Database could not connected: ' + error)

}

)

// Setting up port with express js

const employeeRoute = require('../backend/routes/employee.route')

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({

extended: false

}));

app.use(cors());

app.use(express.static(path.join(\_\_dirname, 'dist/mean-stack-crud-app')));

app.use('/', express.static(path.join(\_\_dirname, 'dist/mean-stack-crud-app')));

app.use('/api', employeeRoute)

// Create port

const port = process.env.PORT || 4000;

const server = app.listen(port, () => {

console.log('Connected to port ' + port)

})

// Find 404 and hand over to error handler

app.use((req, res, next) => {

next(createError(404));

});

// error handler

app.use(function (err, req, res, next) {

console.error(err.message); // Log error message in our server's console

if (!err.statusCode) err.statusCode = 500; // If err has no specified error code, set error code to 'Internal Server Error (500)'

res.status(err.statusCode).send(err.message); // All HTTP requests must have a response, so let's send back an error with its status code and message

});

## **#3 Connect MongoDB Database with Angular MEAN Stack App**

It’s time to connect the MongoDB database with MEAN app, use the below commands to setup MongoDB.

Create a **database** folder within the **backend** folder.

**mkdir database**

**Go inside the database folder.**

**cd database**

**Then create the backend > database > db.js file inside the database folder.**

**Echo > db.js**

**Include the given below code in backend > database > db.js file.**

module.exports = {

db: 'mongodb://localhost:27017/meandb'

};

**Note:** **meandb** is the database name.

Now we need to make the connection between MongoDB and MEAN Stack app. Go to **backend > server.js** file and include the below code.

let mongoose = require('mongoose');

// Connecting with mongo db

mongoose.Promise = global.Promise;

mongoose.connect(dbConfig.db, {

useNewUrlParser: true

}).then(() => {

console.log('Database sucessfully connected')

},

error => {

console.log('Database could not connected: ' + error)

}

)

**#4 Create Model with Mongoose JS**

Let us create the **models** folder inside the backend folder.

**mkdir models**

Then i will create the **Employee.js** file.

**Echo > Employee.js**

In this file i will define the Schema for employees collection. My data types are **name, email, designation and phoneNumber**. Add the given below code in **backend > models > Employee.js** file.

**#5 Create RESTful APIs using Express JS Routes**

Let us create the routes in Angular app to access the Employee data through RESTful APIs. I will be using Mongoose.js in our MEAN Stack Tutorial to create, read, update & delete data from MongoDB database.

Create **backend > routes > employee.route.js** file inside the routes folder.

Let us create the **routes** folder inside the backend folder.

**mkdir routes**

Then i will create the **Employee.js** file.

**Echo > employee.route.js**

Add the given below code to create RESTful APIs in MEAN Stack app using mongoose.js.

const express = require('express');

const app = express();

const employeeRoute = express.Router();

// Employee model

let Employee = require('../models/Employee');

// Add Employee

employeeRoute.route('/create').post((req, res, next) => {

Employee.create(req.body, (error, data) => {

if (error) {

return next(error)

} else {

res.json(data)

}

})

});

// Get All Employees

employeeRoute.route('/').get((req, res) => {

Employee.find((error, data) => {

if (error) {

return next(error)

} else {

res.json(data)

}

})

})

// Get single employee

employeeRoute.route('/read/:id').get((req, res) => {

Employee.findById(req.params.id, (error, data) => {

if (error) {

return next(error)

} else {

res.json(data)

}

})

})

// Update employee

employeeRoute.route('/update/:id').put((req, res, next) => {

Employee.findByIdAndUpdate(req.params.id, {

$set: req.body

}, (error, data) => {

if (error) {

return next(error);

console.log(error)

} else {

res.json(data)

console.log('Data updated successfully')

}

})

})

// Delete employee

employeeRoute.route('/delete/:id').delete((req, res, next) => {

Employee.findOneAndRemove(req.params.id, (error, data) => {

if (error) {

return next(error);

} else {

res.status(200).json({

msg: data

})

}

})

})

module.exports = employeeRoute;

We have set up our MEAN Stack Angular app’s backend using Node js, Express js, Angular and MongoDB.

We have to start 3 servers in our app to start the development in MEAN Stack Angular app.

**Start Nodemon Server**

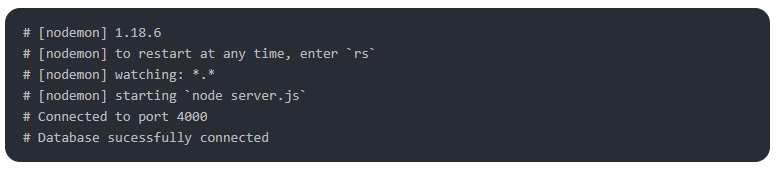
In order to start nodemon server, first enter into the backend folder using given below command.

**cd backend**

Then run the following command to start the nodemon server.

**nodemon server**

You’ll get the following output in your terminal



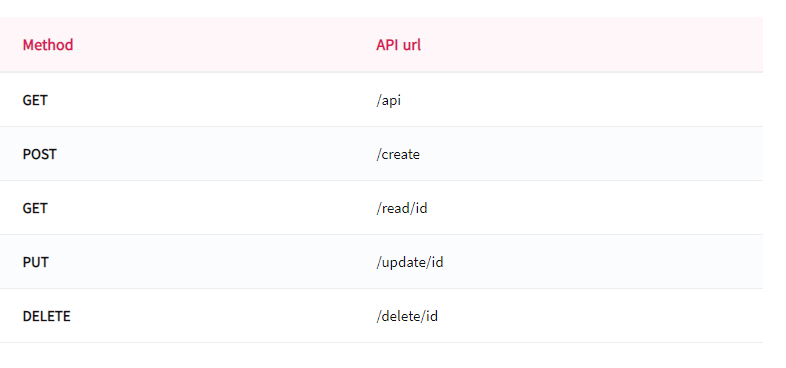
You can access your API route on given below url, here you can check your data.

Check your Angular frontend on – [http://localhost:4200](http://localhost:4200/)

You can check your api url on – <http://localhost:4000/api>

**MEAN Stack App RESTful APIs**

We have successfully created APIs to handle CRUD operations in our MEAN Stack app.



To test the REST API you must use the below command.

**curl -i -H "Accept: application/json" localhost:4000/api**

Given below output indicates that your REST API is ready to go.



**#6 Set up Angular 11 MEAN Stack Project**

**Install Angular CLI**  
Angular project is developed using Angular CLI, so before setting up Angular project. You must have Angular CLI installed in your system. Hit the given below command to install the Angular CLI, ignore if Angular CLI is already installed.

**npm install @angular/cli -g**

Let us install Angular project, run the following command.

**ng new mean-stack-crud-app**

Angular CLI asks for your choices while setting up the project…

**Would you like to add Angular routing?**  
Select y and Hit Enter.

**Which stylesheet format would you like to use?** (Use arrow keys)  
Choose CSS and hit Enter

Your Angular project is installed now get into the project directory.

**cd mean-stack-crud-app**

If using visual studio code editor then use the below cmd to open the project.

**code .**

For this demo **MEAN stack tutorial**, I will use Bootstrap 4 for creating employee management system. Use the following cmd to install Bootstrap 4.

**npm install bootstrap**

Then, Go to **angular.json** file and add the below code in **“styles”: [ ]** array like given below.

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.css"

]

**Generate components in Angular app.**  
In order to manage components i will keep all the components in components folder, use the below cmd to generate components.

**ng g c components/employee-create**

**ng g c components/employee-edit**

**ng g c components/employee-list**

Your Angular app has been set up for MEAN Stack development. enter the below command to run the project.

**Start Angular Server**

**ng serve**

**#7 Activate Routing Service in MEAN Stack Angular App**

In order to navigate between multiple components, we must set up routing service in our app. Now if you remember while setting up an Angular project, CLI asked this question **“Would you like to add Angular routing?”**. We selected yes, it automatically created

app-routing.module.ts and registered in **src > app > app.module.ts** file.

Include the below code to enable routing service in Angular app

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

import { Routes, RouterModule } from '@angular/router';

import { EmployeeCreateComponent } from './components/employee-create/employee-create.component';

import { EmployeeListComponent } from './components/employee-list/employee-list.component';

import { EmployeeEditComponent } from './components/employee-edit/employee-edit.component';

const routes: Routes = [

{ path: '', pathMatch: 'full', redirectTo: 'create-employee' },

{ path: 'create-employee', component: EmployeeCreateComponent },

{ path: 'edit-employee/:id', component: EmployeeEditComponent },

{ path: 'employees-list', component: EmployeeListComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

To enable routing service include the below code in **app.component.html** file.

<nav>

<a routerLinkActive="active" routerLink="/employees-list">View Employees</a>

<a routerLinkActive="active" routerLink="/create-employee">Add Employee</a>

</nav>

<router-outlet></router-outlet>

**#8 Create Angular Service to Consume RESTful APIs**

To consume RESTful API in MEAN Stack Angualr 7 app, we need to create a service file. This service file will handle Create, Read, Update and Delete operations.

Before we create service in MEAN Stack app to consume RESTful APIs, We need to do 2 following things:

**– Configure the HttpClientModule**

We need to import **HttpClientModule**service in app.module.ts file.

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

@NgModule({

imports: [

HttpClientModule

]

})

You’ve successfully placed the **HttpClientModule** in your Angular app.

**– Create Employee Model File**

Create **src > model > employee.ts** file

Add the following code in it.

export class Employee {

name: string;

email: string;

designation: string;

phoneNumber: number;

}

#### Create Angular Service

Use the given below cmd to create Angular Service file to manage CRUD operations in MEAN Stack Angular app.

**ng g s service/api**

Now go to **src > app > service > api.service.ts** file and add the below code.

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

import { Observable, throwError } from 'rxjs';

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

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

@Injectable({

providedIn: 'root'

})

export class ApiService {

baseUri:string = 'http://localhost:4000/api';

headers = new HttpHeaders().set('Content-Type', 'application/json');

constructor(private http: HttpClient) { }

// Create

createEmployee(data): Observable<any> {

let url = `${this.baseUri}/create`;

return this.http.post(url, data)

.pipe(

catchError(this.errorMgmt)

)

}

// Get all employees

getEmployees() {

return this.http.get(`${this.baseUri}`);

}

// Get employee

getEmployee(id): Observable<any> {

let url = `${this.baseUri}/read/${id}`;

return this.http.get(url, {headers: this.headers}).pipe(

map((res: Response) => {

return res || {}

}),

catchError(this.errorMgmt)

)

}

// Update employee

updateEmployee(id, data): Observable<any> {

let url = `${this.baseUri}/update/${id}`;

return this.http.put(url, data, { headers: this.headers }).pipe(

catchError(this.errorMgmt)

)

}

// Delete employee

deleteEmployee(id): Observable<any> {

let url = `${this.baseUri}/delete/${id}`;

return this.http.delete(url, { headers: this.headers }).pipe(

catchError(this.errorMgmt)

)

}

// Error handling

errorMgmt(error: HttpErrorResponse) {

let errorMessage = '';

if (error.error instanceof ErrorEvent) {

// Get client-side error

errorMessage = error.error.message;

} else {

// Get server-side error

errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;

}

console.log(errorMessage);

return throwError(errorMessage);

}

}

We have created Angular service file to handle CRUD operations in our app, now go to **app.module.ts** file and import this service and add into the **providers** array like given below.

import { ApiService } from './service/api.service';

@NgModule({

providers: [ApiService]

})

**#9 Register an Employee by Consuming RESTful API in Angular MEAN Stack App**

To register an employee we will use Angular service and RESTful APIs. I’ve used Reactive Forms to register an employee. We are also covering Reactive forms validations in our MEAN Stack app tutorial.

Go to **components > employee-create > employee-create.component.ts** file and add the following code.

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

import { ApiService } from './../../service/api.service';

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

import { FormGroup, FormBuilder, Validators } from "@angular/forms";

@Component({

selector: 'app-employee-create',

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

styleUrls: ['./employee-create.component.css']

})

export class EmployeeCreateComponent implements OnInit {

submitted = false;

employeeForm: FormGroup;

EmployeeProfile:any = ['Finance', 'BDM', 'HR', 'Sales', 'Admin']

constructor(

public fb: FormBuilder,

private router: Router,

private ngZone: NgZone,

private apiService: ApiService

) {

this.mainForm();

}

ngOnInit() { }

mainForm() {

this.employeeForm = this.fb.group({

name: ['', [Validators.required]],

email: ['', [Validators.required, Validators.pattern('[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$')]],

designation: ['', [Validators.required]],

phoneNumber: ['', [Validators.required, Validators.pattern('^[0-9]+$')]]

})

}

// Choose designation with select dropdown

updateProfile(e){

this.employeeForm.get('designation').setValue(e, {

onlySelf: true

})

}

// Getter to access form control

get myForm(){

return this.employeeForm.controls;

}

onSubmit() {

this.submitted = true;

if (!this.employeeForm.valid) {

return false;

} else {

this.apiService.createEmployee(this.employeeForm.value).subscribe(

(res) => {

console.log('Employee successfully created!')

this.ngZone.run(() => this.router.navigateByUrl('/employees-list'))

}, (error) => {

console.log(error);

});

}

}

}

Go to **employee-create.component.html** add the following code.

<div class="row justify-content-center">

<div class="col-md-4 register-employee">

<!-- form card register -->

<div class="card-body">

<form [formGroup]="employeeForm" (ngSubmit)="onSubmit()">

<div class="form-group">

<label for="inputName">Name</label>

<input class="form-control" type="text" formControlName="name">

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.name.errors?.required">

Name is required.

</div>

</div>

<div class="form-group">

<label for="inputEmail3">Email</label>

<input class="form-control" type="text" formControlName="email">

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.email.errors?.required">

Enter your email.

</div>

<div class="invalid-feedback" \*ngIf="submitted && myForm.email.errors?.pattern">

Enter valid email.

</div>

</div>

<div class="form-group">

<label for="inputPassword3">Designation</label>

<select class="custom-select form-control" (change)="updateProfile($event.target.value)"

formControlName="designation">

<option value="">Choose...</option>

<option \*ngFor="let employeeProfile of EmployeeProfile" value="{{employeeProfile}}">{{employeeProfile}}

</option>

</select>

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.designation.errors?.required">

Choose designation.

</div>

</div>

<div class="form-group">

<label for="inputVerify3">Mobile No</label>

<input class="form-control" type="text" formControlName="phoneNumber">

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.phoneNumber.errors?.required">

Enter your phone number.

</div>

<div class="invalid-feedback" \*ngIf="submitted && myForm.phoneNumber.errors?.pattern">

Enter Numbers Only

</div>

</div>

<div class="form-group">

<button class="btn btn-success btn-lg btn-block" type="submit">Register</button>

</div>

</form>

</div>

</div><!-- form card register -->

</div>

To register an employee we will use Angular service and RESTful APIs. I’ve used Reactive Forms to register an employee. We are also covering Reactive forms validations in our MEAN Stack app tutorial.

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule,

**FormsModule,**

**ReactiveFormsModule**

]

## **#10 Show Employees List and Delete Student Object using RESTful API in MEAN Stack App**

I will show the Employees list using RESTful APIs and Angular service. Go to**src/app/components/employee-list/employee-list.component.ts**file and include the below code.

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

import { ApiService } from './../../service/api.service';

@Component({

selector: 'app-employee-list',

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

styleUrls: ['./employee-list.component.css']

})

export class EmployeeListComponent implements OnInit {

Employee:any = [];

constructor(private apiService: ApiService) {

this.readEmployee();

}

ngOnInit() {}

readEmployee(){

this.apiService.getEmployees().subscribe((data) => {

this.Employee = data;

})

}

removeEmployee(employee, index) {

if(window.confirm('Are you sure?')) {

this.apiService.deleteEmployee(employee.\_id).subscribe((data) => {

this.Employee.splice(index, 1);

}

)

}

} }

To display employees list open the**src/app/components/employee-list/employee-list.component.html**file and add the following code in it.

<div class="container">

<!-- No data message -->

<p \*ngIf="Employee.length <= 0" class="no-data text-center">There is no employee added yet!</p>

<!-- Employee list -->

<table class="table table-bordered" \*ngIf="Employee.length > 0">

<thead class="table-success">

<tr>

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

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

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

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

<th scope="col">Phone No</th>

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

</tr>

</thead>

<tbody>

<tr \*ngFor="let employee of Employee; let i = index">

<th scope="row">{{employee.\_id}}</th>

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

<td>{{employee.email}}</td>

<td>{{employee.designation}}</td>

<td>{{employee.phoneNumber}}</td>

<td class="text-center edit-block">

<span class="edit" [routerLink]="['/edit-employee/', employee.\_id]">

<button type="button" class="btn btn-success btn-sm">Edit</button>

</span>

<span class="delete" (click)="removeEmployee(employee, i)">

<button type="button" class="btn btn-danger btn-sm">Delete</button>

</span>

</td>

</tr>

</tbody>

</table>

</div>

## **#11 Edit Employees Data in MEAN Stack Angualr 7 App**

In order to edit employees data we need to add the following code in **src/app/components/employee-edit/employee-edit.component.html**file.

<div class="row justify-content-center">

<div class="col-md-4 register-employee">

<!-- form card register -->

<div class="card card-outline-secondary">

<div class="card-header">

<h3 class="mb-0">Edit Employee</h3>

</div>

<div class="card-body">

<form [formGroup]="editForm" (ngSubmit)="onSubmit()">

<div class="form-group">

<label for="inputName">Name</label>

<input class="form-control" type="text" formControlName="name">

<div class="invalid-feedback" \*ngIf="submitted && myForm.name.errors?.required">

Name is required.

</div>

</div>

<div class="form-group">

<label for="inputEmail3">Email</label>

<input class="form-control" type="text" formControlName="email">

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.email.errors?.required">

Enter your email.

</div>

<div class="invalid-feedback" \*ngIf="submitted && myForm.email.errors?.pattern">

Enter valid email.

</div>

</div>

<div class="form-group">

<label for="inputPassword3">Designation</label>

<select class="custom-select form-control" (change)="updateProfile($event.target.value)"

formControlName="designation">

<option value="">Choose...</option>

<option \*ngFor="let employeeProfile of EmployeeProfile" value="{{employeeProfile}}">{{employeeProfile}}

</option>

</select>

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.designation.errors?.required">

Choose designation.

</div>

</div>

<div class="form-group">

<label for="inputVerify3">Mobile No</label>

<input class="form-control" type="text" formControlName="phoneNumber">

<!-- error -->

<div class="invalid-feedback" \*ngIf="submitted && myForm.phoneNumber.errors?.required">

Enter your phone number.

</div>

<div class="invalid-feedback" \*ngIf="submitted && myForm.phoneNumber.errors?.pattern">

Enter Numbers Only

</div>

</div>

<div class="form-group">

<button class="btn btn-success btn-lg btn-block" type="submit">Update</button>

</div>

</form>

</div>

</div><!-- form -->

</div>

</div>

To edit employees data we need to add the following code in **src/app/components/employee-edit/employee-edit.component.ts** file.

import { Employee } from './../../model/Employee';

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

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

import { ApiService } from './../../service/api.service';

import { FormGroup, FormBuilder, Validators } from "@angular/forms";

@Component({

selector: 'app-employee-edit',

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

styleUrls: ['./employee-edit.component.css']

})

export class EmployeeEditComponent implements OnInit {

submitted = false;

editForm: FormGroup;

employeeData: Employee[];

EmployeeProfile: any = ['Finance', 'BDM', 'HR', 'Sales', 'Admin']

constructor(

public fb: FormBuilder,

private actRoute: ActivatedRoute,

private apiService: ApiService,

private router: Router

) {}

ngOnInit() {

this.updateEmployee();

let id = this.actRoute.snapshot.paramMap.get('id');

this.getEmployee(id);

this.editForm = this.fb.group({

name: ['', [Validators.required]],

email: ['', [Validators.required, Validators.pattern('[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$')]],

designation: ['', [Validators.required]],

phoneNumber: ['', [Validators.required, Validators.pattern('^[0-9]+$')]]

})

}

// Choose options with select-dropdown

updateProfile(e) {

this.editForm.get('designation').setValue(e, {

onlySelf: true

})

}

// Getter to access form control

get myForm() {

return this.editForm.controls;

}

getEmployee(id) {

this.apiService.getEmployee(id).subscribe(data => {

this.editForm.setValue({

name: data['name'],

email: data['email'],

designation: data['designation'],

phoneNumber: data['phoneNumber'],

});

});

}

updateEmployee() {

this.editForm = this.fb.group({

name: ['', [Validators.required]],

email: ['', [Validators.required, Validators.pattern('[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$')]],

designation: ['', [Validators.required]],

phoneNumber: ['', [Validators.required, Validators.pattern('^[0-9]+$')]]

})

}

onSubmit() {

this.submitted = true;

if (!this.editForm.valid) {

return false;

} else {

if (window.confirm('Are you sure?')) {

let id = this.actRoute.snapshot.paramMap.get('id');

this.apiService.updateEmployee(id, this.editForm.value)

.subscribe(res => {

this.router.navigateByUrl('/employees-list');

console.log('Content updated successfully!')

}, (error) => {

console.log(error)

})

}

}

}

}

We have created basic MEAN Stack Angular 11 CRUD app, now enter the below command to start your project on the browser.

**ng serve**