# **STARTING AND CONNECTING ANGULAR NODE PROJECT**

**NODEJS SERVER**

* Create a folder for the backend
* Navigate to the folder on VSCode
* Run npm init and initialize project
* Npm i body-parser cors express mongoose --save
* Create a environment.json file

// Setup the dabase environment

{

    "database" : {

        "connection\_url": "mongodb://localhost:27017/FakeBuk"

    },

    "database\_online" : {

        "connection\_url":  "online string connection"

    },

    "secret": {

        "secret\_key" : "secret"

    }

}

* Create a db folder and then create a mongoose.js file to run the db script
* // db connection logic
* const config = require('../environment');
* const mongoose = require('mongoose');
* mongoose.Promise = global.Promise;
* const mongoConnect = config.database.connection\_url;
* mongoose.connect(mongoConnect, {useNewUrlParser: true, useUnifiedTopology: true}, err =>{
* if(err) {
* console.error('Error!' + err)
* } else {
* console.log('Connected to mongodb')
* }
* })
* //To prevent depracation warnings from MongoDB native driver
* mongoose.set('useCreateIndex', true)
* mongoose.set('useFindAndModify', false)
* module.exports = {
* mongoose
* }
* Create a routes folder and start a user.route.js file with a simple login and register route. This will be then tested on postman later

const express = require('express');

const User = require('../models/user.model')

const cors = require('cors');

const router = express.Router()

router.use(cors());

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

    console.log(req.body);

    console.log('we are in');

    res.send('we here')

})

router.post('/login', (req, res) => {

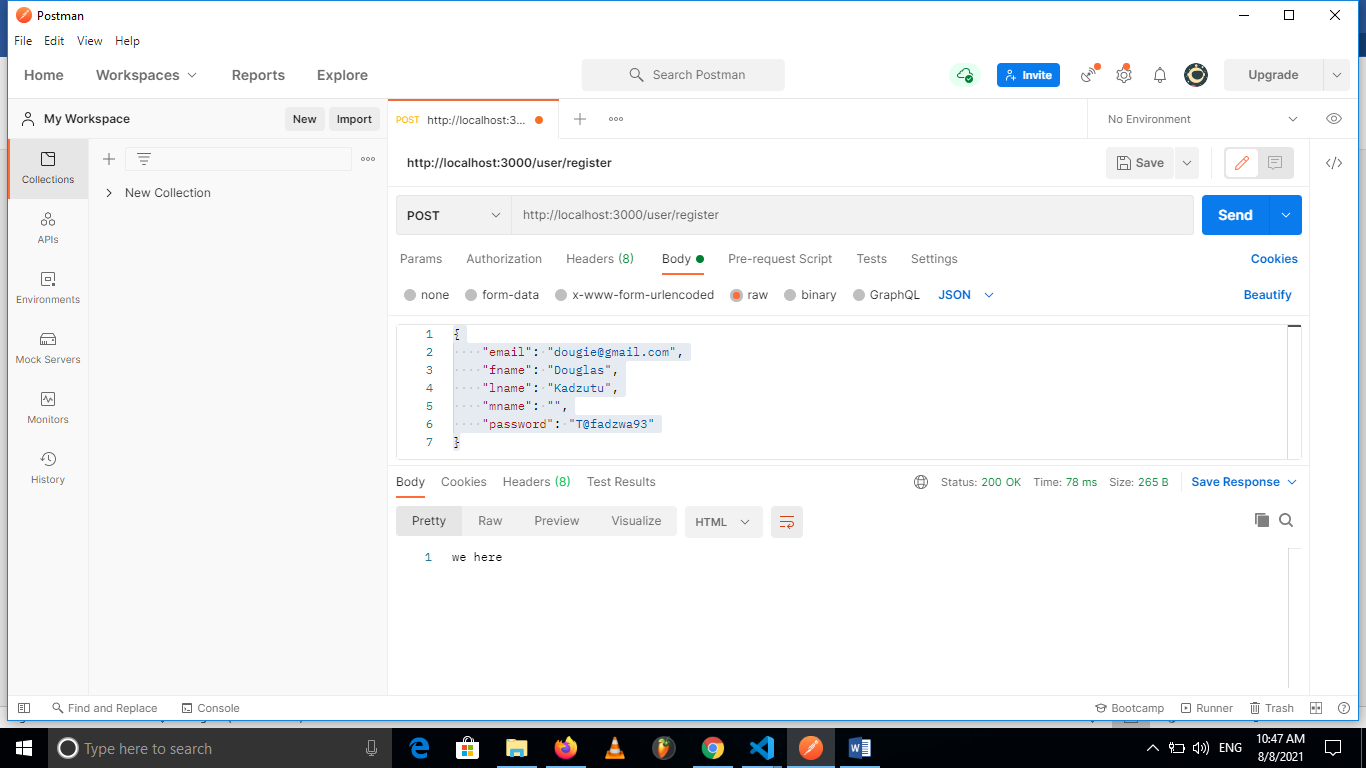
    console.log(req.body);

    console.log('we are in');

})

module.exports = router

* Now create the server.js file
* const express = require('express')
* const {mongoose} = require('./db/mongoose')
* const cors = require('cors')
* const User = require('./db/routes/user.route')
* const app = express();
* app.use(express.urlencoded({ extended: true }));
* app.use(express.json());
* const port = process.env.PORT || 3000
* app.use(cors())
* app.use('/user', User)
* app.get('/', (req, res) => {
* res.send('Hello World')
* })
* app.listen(port , function(){
* console.log('Listening to server from port: ' + port)
* })
* Run node/nodemon server.js
* Test Api route in PostMan the console should work



**Angular App**

* Create a folder for the frontend
* Navigate to that folder in vsCode then run ng new ‘name of frontend’
* Then ng serve
* Clear app.component.html and leave router-outlet
* Open environment.ts folder and setup the connection url
* / This file can be replaced during build by using the `fileReplacements` array.
* // `ng build --prod` replaces `environment.ts` with `environment.prod.ts`.
* // The list of file replacements can be found in `angular.json`.
* export const environment = {
* production: false,
* apiUrl: 'http://localhost:3000'
* };
* /\*
* \* For easier debugging in development mode, you can import the following file
* \* to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.
* \*
* \* This import should be commented out in production mode because it will have a negative impact
* \* on performance if an error is thrown.
* \*/
* // import 'zone.js/dist/zone-error';  // Included with Angular CLI.
* Create and auth folder then create the components for signup and login
* App routing folder should look something like this

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

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

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

import { SignUpComponent } from './auth/sign-up/sign-up.component';

const routes: Routes = [

  { path: '', redirectTo: 'sign-up', pathMatch: 'full'},

  { path: 'sign-up', component: SignUpComponent},

  { path: 'login', component: LoginComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

* Then example of the signup component should look like this
* import { Component, OnInit, ViewChild } from '@angular/core';
* import { NgForm } from '@angular/forms';
* import { RegisterUser } from 'src/app/core/interface';
* import { AuthService } from '../auth.service';
* @Component({
* selector: 'app-sign-up',
* templateUrl: './sign-up.component.html',
* styleUrls: ['./sign-up.component.scss']
* })
* export class SignUpComponent implements OnInit {
* @ViewChild('f', {static: true}) registerForm: NgForm;
* user: RegisterUser
* constructor (private authService: AuthService) { }
* ngOnInit() {
* }
* onSignUp() {
* console.log(this.registerForm.value);
* const regex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[#$@!%&\*?])[A-Za-z\d#$@!%&\*?]{6,16}$/;
* // check password
* let pass = this.registerForm.value.password;
* if (!this.validateEmail(this.registerForm.value.email)) {
* console.log('email format is invalid');
* return false;
* }
* if (!regex.test(pass)) {
* console.log('password pattern incorrect')
* return false
* }
* if (pass !== this.registerForm.value.confPassword) {
* console.log('password misatch');
* return false;
* } else {
* if(!regex.test(pass)) {
* console.log('password pattern incorrect')
* return false;
* }
* console.log('everything is ok');
* }
* // prepare to send to Backend
* this.user = {
* firstname: this.registerForm.value.fname,
* middlename: this.registerForm.value.mname,
* lastname: this.registerForm.value.lname,
* email: this.registerForm.value.email,
* password: this.registerForm.value.password
* }
* if (this.registerForm.valid) {
* this.authService.signUp(this.user).subscribe((res)=> {
* console.log(res);
* });
* } else {
* console.log('Form is not valid')
* }
* }
* validateEmail(email: string) {
* const re = /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
* return re.test(String(email).toLowerCase());
* }
* }
* Then create Auth Service in the auth folder which will look like this
* import { Injectable } from '@angular/core';
* import { LoginUser, RegisterUser } from '../core/interface';
* import { ApiService } from '../core/services/api.service';
* import { shareReplay, tap } from 'rxjs/operators';
* @Injectable(
* {providedIn: 'root'}
* )
* export class AuthService{
* constructor(private apiService: ApiService) {
* }
* login(credentials: LoginUser) {
* return this.apiService.login(credentials);
* }
* signUp(credentials: RegisterUser) {
* return this.apiService.signUp(credentials).pipe(
* shareReplay(),
* tap((res) => {
* console.log(res);
* })
* );
* }
* }
* Now create a cores folder and in the folder create a Servive folder with api.service.ts and global.service.ts
* Api Service should look like this
* import { Injectable } from '@angular/core';
* import { HttpClient } from '@angular/common/http'
* import { LoginUser, RegisterUser } from '../interface';
* import { Globals } from './globals.service';
* @Injectable({
* providedIn: 'root'
* })
* export class ApiService {
* constructor ( private http: HttpClient, private globals: Globals ) {
* }
* login(credentials: LoginUser) {
* // send to backend url
* console.log('ready to send to backend: ' + credentials.email + ' ' + credentials.password);
* }
* signUp(credentials: RegisterUser) {
* // send to backend url
* console.log('ready to send to backend: ' + credentials);
* return this.http.post(`${this.globals.api}/user/register`,  credentials , { observe: 'response' });
* }
* }
* Global service.ts should look like this

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

import { environment } from '../../../environments/environment';

@Injectable(

    {providedIn: 'root'}

)

export class Globals {

    constructor () {}

    private \_api = environment.apiUrl;

    get api(): string {

        return this.\_api;

    }

}

* In the core folder also add the interface.ts file which will look something like this
* import {Injectable} from '@angular/core';
* @Injectable({providedIn: 'root'})
* export class Interface {
* constructor() {
* }
* }
* export interface LoginUser {
* email: string;
* password: string;
* }
* export interface RegisterUser {
* firstname: string;
* middlename?: string;
* lastname: string;
* email: string;
* password: string;
* created?: Date;
* }
* At this point the app.module.ts should look like this

import { BrowserModule } from '@angular/platform-browser';

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

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

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

import { CommonModule } from '@angular/common';

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

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { SignUpComponent } from './auth/sign-up/sign-up.component';

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

@NgModule({

  declarations: [

    AppComponent,

    SignUpComponent,

    LoginComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    ReactiveFormsModule,

    CommonModule,

    HttpClientModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

* **This should connect you to the backend and the coding can begin**

**GOODLUCK!!!**