**JWT TOKEN**

Backend API

---------------

1. Create a Project and generate package.json



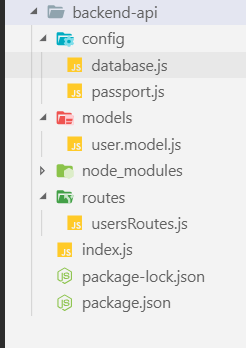
Bcrypt : use to hash a password

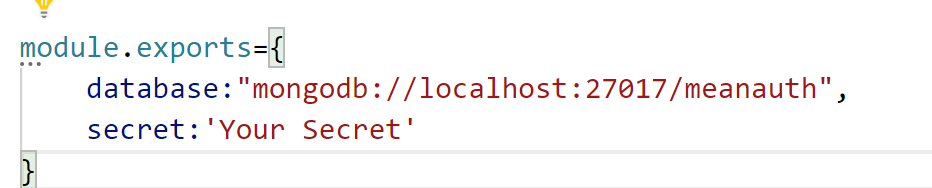
Passport : Express compatible authentication middleware for Node js. Purpose is to authenticate requests which it does through extensible set of plugins known as strategies

Strategies can include verifying username and password from OAuth via Facebook Or Twitter or Ferderated Authentication using OpenID

Passport-jwt : Passport strategy for authenticating with JSON Web token. Its intended to be used to secure RESTful endpoints without sessions.

1. Create config folder and database.js inside it

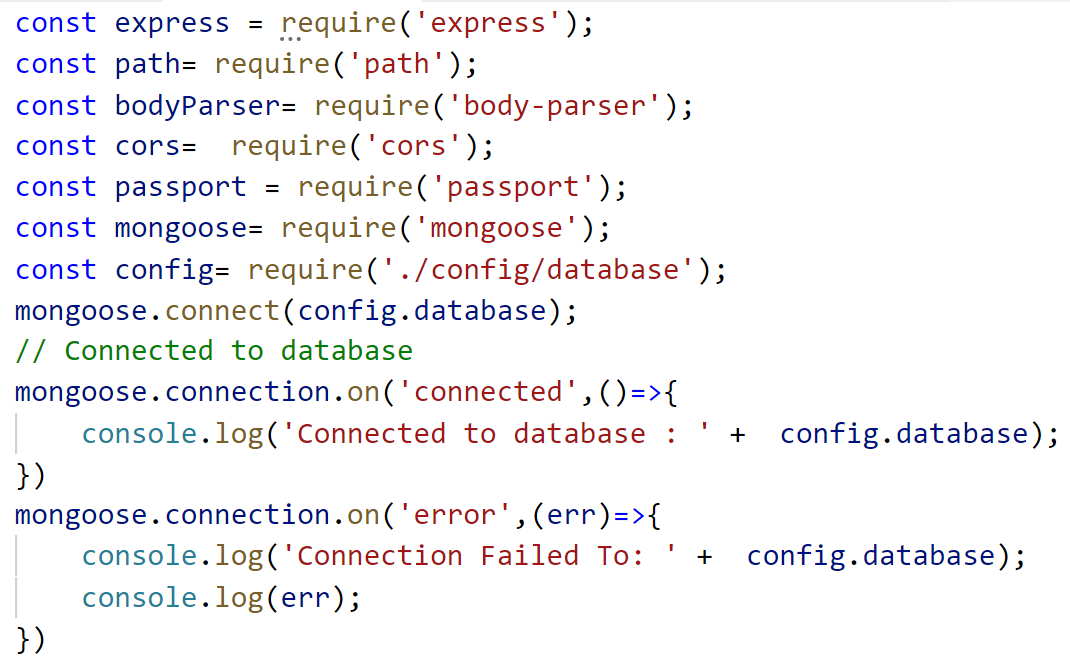




1. Create route folder and user.model.jsin it



1. Create index.js and write code inside it.

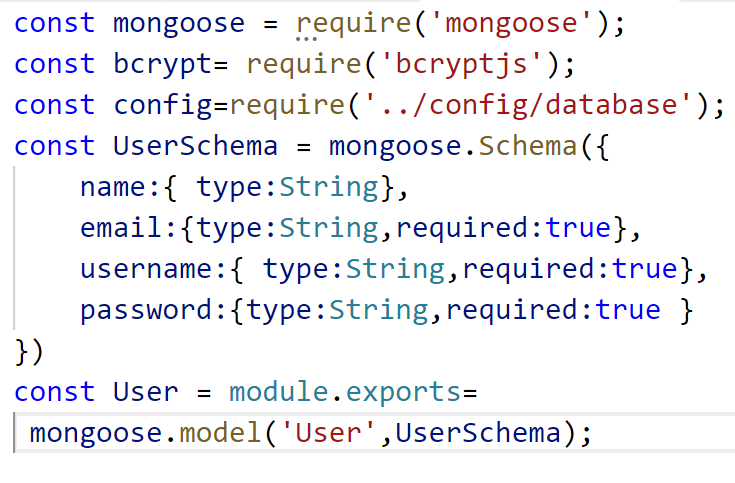




1. Check the Output

Creating a model for Registration

1. Create user.model.js model in the models folder



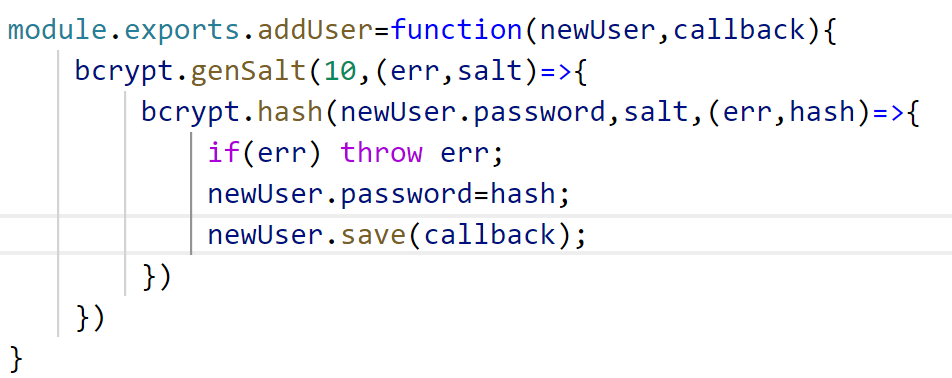
1. Add couple of methods into user.model.js which we are going to use later



1. Go to usersRoutes.js and write the Registration code



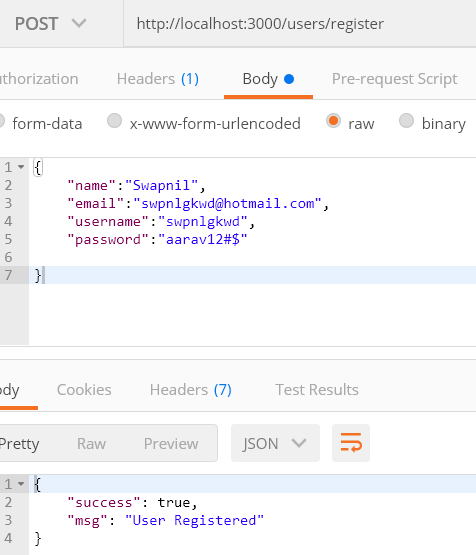
1. Add addUser in user.js



What is genSalt and Salt

* GenSalt , generates the salt , default value is 10
* Salt hashing is a technique in which we take the user entered password and a random string of characters called as salt. Hash the combined string with a suitable hashing and algorithm and store in database.
* Adding salt to a password and then hashing the result reduces the possibility of having duplicate hashes and if your salt length is long enough chances are minimal.

1. Check whether data can be inserted or not using POST MAN



**Authentication and Authorization using Passport**

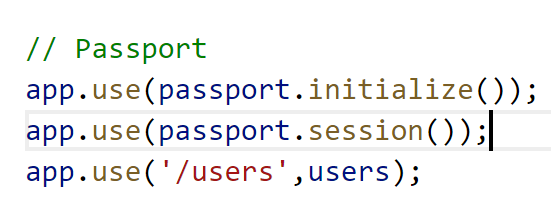
* Passport is authentication middleware for Node JS
* Supports authentication using a username and password, facebook and Twitter.
* After Successful authentication , passport will establish a persistent login session. This is useful for the common scenarios of users accessing a web application via browser

However in some cases session support is not necessary for example API servers typically require credentials to be supplied with each request when this is the you can safely disabled setting the session option to false.

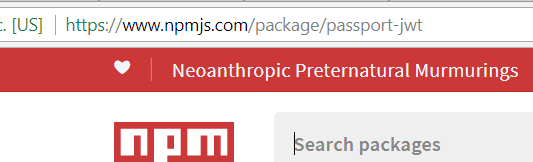
passport.initialize() middleware is required to initialize Passport. If your application uses persistent login sessions, passport.session() middleware must also be used.

Authentication Mechanism in Express

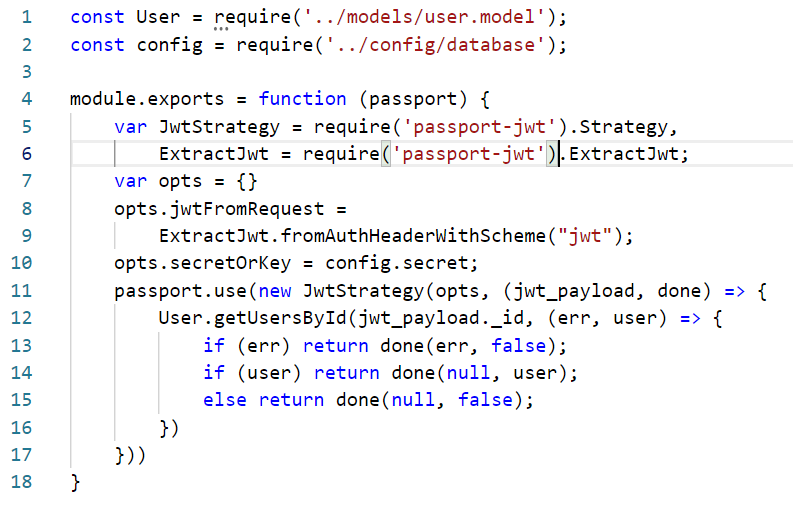
1. Enable passport and session . go to index.js and write a following code.



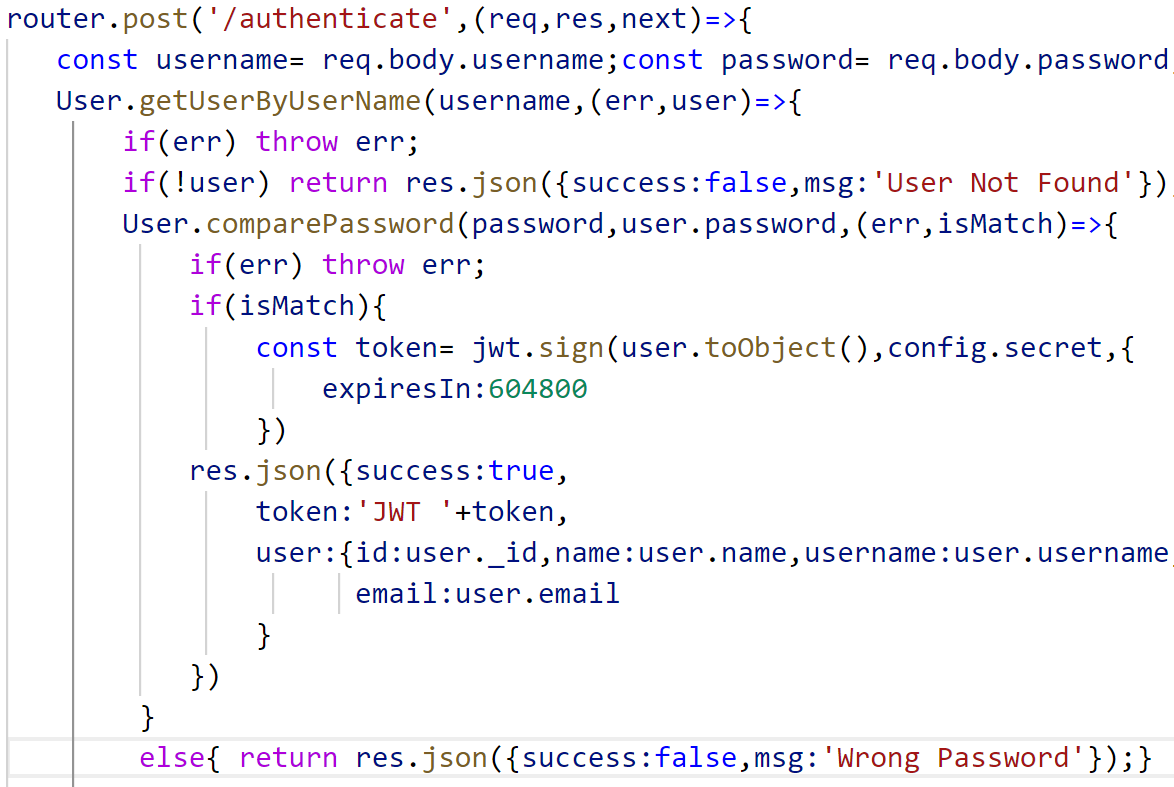
1. Create a strategy : We are going to create passport with jwt strategy



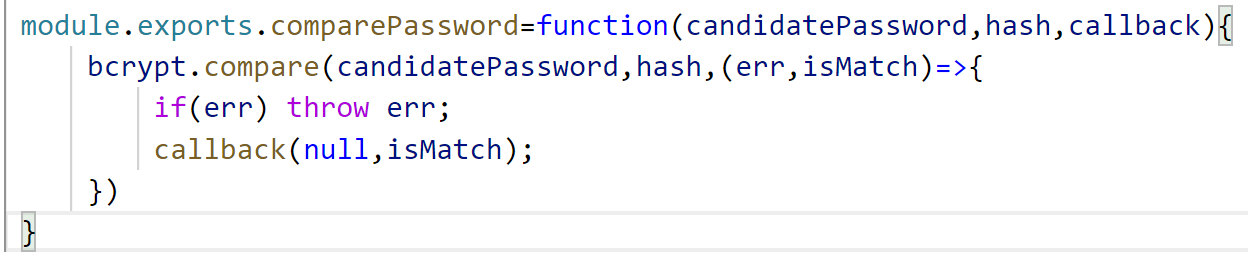
1. Create config/passport.js



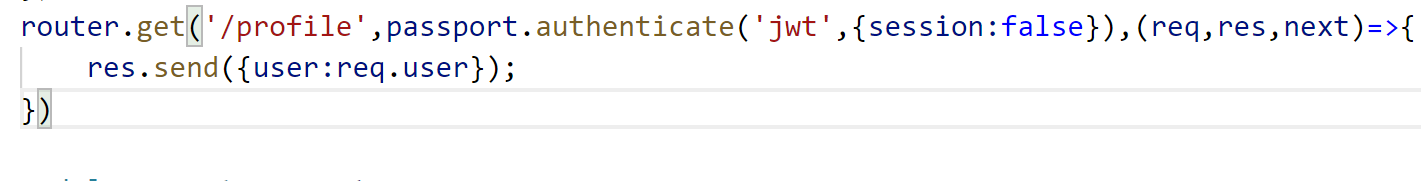
1. Add router.post inside usersRoutes.js



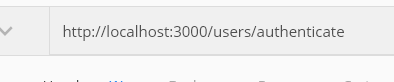
1. Go to user.model.js and add following method



1. Protect Routes



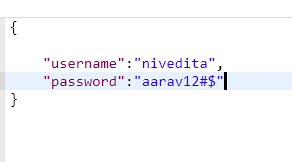
1. Go to Postman and type and use POST



Add header

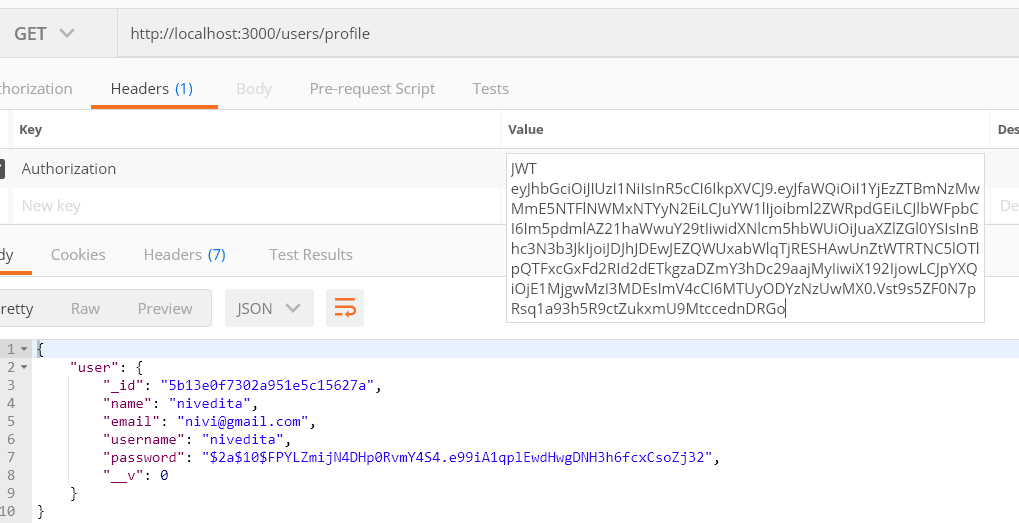


Add body



Copy the JWT token

Go to another tab and try to access profile with or without passing key



CHANGES FROM HERE

Index.js file

const express = require('express');

const path = require('path');

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

const cors = require('cors');

var passport = require('passport');// create a passport instance

var myPassportService = require('./config/passport')(passport);

const mongoose = require('mongoose');

const config = require('./config/database');

const users = require('./routes/usersRoutes');

const app = express();

mongoose.connect(config.database, { useNewUrlParser: true });

// connecting database

mongoose.connection.on('connected', () => {

console.log('Connected to database: ' + config.database);

})

mongoose.connection.on('error', (err) => {

console.log('Connection Failed To : ' + config.database);

console.log(err);

})

const port = 3000;

app.use(cors());

// app.use(express.static(\_\_dirname, 'public'));

app.use(bodyParser.json());

// enabling passport authentication for routes

app.use(passport.initialize());

app.use(passport.session());

app.use('/users', users);

// Index Route

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

res.send('Invalid Endpoint');

})

// starting server

app.listen(port, () => {

console.log('Server started on port : ' + port);

})

routes/userRoutes.js

const express = require('express');

const router = express.Router();

const jwt = require('jsonwebtoken');

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

const passport = require('passport');

const config = require('../config/database');

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

let newUser = new User({

name: req.body.name,

email: req.body.email,

username: req.body.username,

password: req.body.password

});

User.addUser(newUser, (err, user) => {

if (err) {

res.json({

success: false,

msg: 'Failed to Register'

});

}

else {

res.json({

success: true,

msg: 'User is Registered'

});

}

})

})

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

const email = req.body.email;

const password = req.body.password;

User.getUsersByUserName(email, (err, user) => {

if (err) throw err;

if (!user)

return res.json({

success: false,

msg: 'User not found'

})

User.comparePassword(password, user.password,

(err, isMatch) => {

if (err) throw err;

if (isMatch) {

const token = jwt.sign(user.toObject(),

config.secret, { expiresIn: 604800 })

res.json({

success: true,

token: 'JWT ' + token,

user: {

id: user.\_id,

name: user.name,

username: user.username,

email: user.email

}

})

}

else {

return res.json({

success: false,

msg: 'Wrong Password'

})

}

})

})

})

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

// res.send('Register');

// })

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

// res.send('authenticate');

// })

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

// res.send('validate');

// })

router.get('/profile', passport.authenticate('jwt', { session: false }), (req, res, next) => {

res.json({ user: req.user });

})

module.exports = router;

models/user.model.js

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const config = require('../config/database');

const UserSchema = mongoose.Schema({

name: { type: String },

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

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

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

});

const User = module.exports =

mongoose.model('User', UserSchema);

module.exports.getUsersById = function (id, callback) {

User.findById(id, callback);

}

module.exports.getUsersByUserName =

function (email, callback) {

const query = { email: email };

User.findOne(query, callback);

}

module.exports.addUser = function (newUser, callback) {

bcrypt.genSalt(10, (err, salt) => {

bcrypt.hash(newUser.password, salt,

(err, hash) => {

if (err) throw err;

newUser.password = hash;

newUser.save(callback);

})

})

}

module.exports.comparePassword = function (candidatePassword, hash, callback) {

bcrypt.compare(candidatePassword, hash, (err, isMatch) => {

if (err) throw err;

callback(null, isMatch);

})

}

Config/database.js

// configuring database connection

module.exports = {

database: "mongodb://localhost:27017/usersdb",

secret: "My Secret Key"

}

Config/passport.js

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

const config = require('../config/database');

module.exports = function (passport) {

var JwtStrategy = require('passport-jwt').Strategy,

ExtractJwt = require('passport-jwt').ExtractJwt;

var opts = {}

opts.jwtFromRequest =

ExtractJwt.fromAuthHeaderWithScheme("jwt");

opts.secretOrKey = config.secret;

passport.use(new JwtStrategy(opts, (jwt\_payload, done) => {

User.getUsersById(jwt\_payload.\_id, (err, user) => {

if (err) return done(err, false);

if (user) return done(null, user);

else return done(null, false);

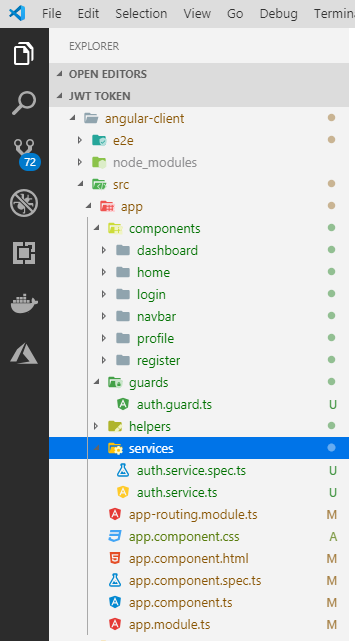
})

}))

}

Angular Client

Create application using ng new command



App.module.ts

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

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

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

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

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

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

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

import { HomeComponent } from './components/home/home.component';

import { RegisterComponent } from './components/register/register.component';

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

import { DashboardComponent } from './components/dashboard/dashboard.component';

import { ProfileComponent } from './components/profile/profile.component';

import { NavbarComponent } from './components/navbar/navbar.component';

@NgModule({

declarations: [

AppComponent,

HomeComponent,

RegisterComponent,

LoginComponent,

DashboardComponent,

ProfileComponent,

NavbarComponent

],

imports: [

BrowserModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

AppRoutingModule

],

providers: [

],

bootstrap: [AppComponent]

})

export class AppModule { }

app-routing.module.ts

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

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

import { HomeComponent } from './components/home/home.component';

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

import { RegisterComponent } from './components/register/register.component';

import { DashboardComponent } from './components/dashboard/dashboard.component';

import { ProfileComponent } from './components/profile/profile.component';

import { AuthGuard } from './guards/auth.guard';

const routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'login', component: LoginComponent },

{ path: 'register', component: RegisterComponent },

// Routes which will be accessible through token must pass thru AuthGuard

{ path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard] },

{ path: 'profile', component: ProfileComponent, canActivate: [AuthGuard] },

{ path: '\*\*', redirectTo: '' }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

App.component.ts

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'jwt-auth';

}

App.component.html

<app-navbar></app-navbar>

<br /><br /><br /><br />

<div class="container">

<div class="row">

<div class="col-md-4 offset-md-4">

<router-outlet></router-outlet>

</div>

</div>

</div>

Services/auth.service.ts

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

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

import { map } from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class AuthService {

authToken: any;

user: any;

constructor(private http: HttpClient) { }

// for logOut to disappear incase, if you didnt login yet

isUserLoggedIn = false;

loggedIn() {

return this.isUserLoggedIn;

}

// Register User

registerUser(user) {

console.log('Received: ' + user);

let headers = new HttpHeaders();

headers.append('Content-Type', 'application/json');

return this.http.post('http://localhost:3000/users/register', user, { headers: headers });

}

// Authenticate User

login(user) {

console.log('Received: ' + user);

let headers = new HttpHeaders();

headers.append('Content-Type', 'application/json');

return this.http.post('http://localhost:3000/users/login', user, { headers: headers })

.pipe(map(user => {

// login successful if there's a jwt token in the response

// store user details and jwt token in local storage to keep user logged in between page refreshes

console.log(user);

this.isUserLoggedIn = true;

if (user && user.token) {

this.storeUserDate(user.token, user);

}

return user;

}));

}

// Store token and user details inside browser under localStorage

storeUserDate(token, user) {

localStorage.setItem('id\_token', token);

localStorage.setItem('user', JSON.stringify(user));

this.authToken = token;

this.user = user;

}

// logOut() function

logOut() {

this.isUserLoggedIn = false;

this.authToken = null;

this.user = null;

localStorage.clear();

}

// getProfile()

getProfile() {

console.log('getProfile from Service called');

let headers = new HttpHeaders({

"Authorization": this.authToken,

'Content-Type': 'application/json'

});

this.loadToken();

console.log('Token Set: ' + this.authToken);

return this.http.get('http://localhost:3000/users/profile', { headers: headers });

}

// loadToken()

loadToken() {

const token = localStorage.getItem('id\_token');

console.log('Token Received: ' + token);

this.authToken = token;

}

}

Guards/auth.guard.ts

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

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

import { AuthService } from '../services/auth.service';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate {

constructor(private authService: AuthService, private router: Router) { }

canActivate() {

if (this.authService.loggedIn() || localStorage.getItem('user')) {

console.log('IF canActivate');

return true;

}

console.log('canActivate');

this.router.navigate['/login'];

return false;

}

}

CREATE NEW FOLDER components INSIDE app folder

Create below components

Ng g c dashboard

Ng g c home

Ng g c login

Ng g c navbar

Ng g c profile

Ng g c register

[Home.component.ts](http://Home.component.ts) file

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

@Component({

selector: 'app-home',

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

styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

constructor() { }

ngOnInit() {

}

}

[Home.component.html](http://Home.component.html)

<div class="jumbotron text-center">

<h1>MEAN Auth</h1>

<p class="lead">Welcome to custom MEAN App</p>

<div>

<a class="btn btn-primary" [routerLink]="['/register']">Register</a> &nbsp;|&nbsp;

<a class="btn btn-primary" [routerLink]="['/login']">Login</a>

</div>

</div>

<div class="row">

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

<h3>Express BackEnd</h3>

<p>Node JS/Express JS Server using mongoose</p>

</div>

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

<h3>Angular CLI</h3>

<p>Node JS/Express JS Server using mongoose</p>

</div>

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

<h3>JWT Tokens</h3>

<p>Node JS/Express JS Server using mongoose</p>

</div>

</div>

Login.component.ts file

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

import { AuthService } from 'src/app/services/auth.service';

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

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

email: string;

password: string;

constructor(private authService: AuthService, private router: Router) { }

ngOnInit() {

}

onLoginSubmit() {

let user = { email: this.email, password: this.password }

this.authService.login(user)

.subscribe(data => {

console.log(data);

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

},

err => {

console.log(err.stack);

alert('User not found');

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

})

}

}

Login.component.html file

<div class="col-md-8 offset-md-2">

<h2 class="page-header">Login</h2>

<form>

<div class="form-group">

<label>Email</label>

<input type="email" [(ngModel)]="email" name="email" class="form-control" />

</div>

<div class="form-group">

<label>Password</label>

<input type="password" [(ngModel)]="password" name="password" class="form-control" />

</div>

<button class="btn btn-primary" (click)="onLoginSubmit()">Login</button>

</form>

</div>

Navbar.component.html file

<div class="container">

<nav class="navbar navbar-expand-lg navbar-dark bg-info">

<div class="navbar-header">

<button class="navbar-toggler collapsed" type="button" data-toggle="collapse" data-target="#collapsibleNavbar">

<span class="navbar-toggler-icon"></span>

</button>

<a class="navbar-brand" href="#">MEAN Auth App</a>

</div>

<div class="collpase navbar-collapse " id="collapsibleNavbar">

<ul class="nav navbar-nav mr-auto ">

<li class="nav-item active ">

<a [routerLink]="['/']" class="text-white nav-link">Home</a>

</li>

</ul>

<ul class="nav navbar-nav ml-auto">

<li class="nav-item" \*ngIf="authService.loggedIn()">

<a [routerLinkActive]="['active']" [routerLink]="['/dashboard']" class="text-white nav-link">Dashboard</a>

</li>

<li class="nav-item" \*ngIf="authService.loggedIn()">

<a [routerLinkActive]="['active']" [routerLink]="['/profile']" class="text-white nav-link">Profile</a>

</li>

<li class="nav-item" \*ngIf="!authService.loggedIn()">

<a [routerLinkActive]="['active']" [routerLink]="['/login']" class="text-white nav-link">Login</a>

</li>

<li class="nav-item" \*ngIf="!authService.loggedIn()">

<a [routerLinkActive]="['active']" [routerLink]="['/register']" class="text-white nav-link">Register</a>

</li>

<li class="nav-item" \*ngIf="authService.loggedIn()">

<a (click)="onLogoutClick()" href="#" class="text-white nav-link">Logout</a>

</li>

</ul>

</div>

</nav>

</div>

Navbar.component.ts file

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

import { AuthService } from 'src/app/services/auth.service';

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

@Component({

selector: 'app-navbar',

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

styleUrls: ['./navbar.component.css']

})

export class NavbarComponent implements OnInit {

constructor(private authService: AuthService, private router: Router) { }

ngOnInit() {

}

// Logout function

onLogoutClick() {

this.authService.logOut();

alert('You are logged out');

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

return false;

}

}

Profile.component.html file

<div \*ngIf="user">

<h2 class="page-header">{{user.name}}</h2>

<ul class="list-group">

<li class="list-group-item">User Name: {{user.username}}</li>

<li class="list-group-item">Email: {{user.email}}</li>

</ul>

</div>

Profile.component.ts file

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

import { AuthService } from 'src/app/services/auth.service';

@Component({

selector: 'app-profile',

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

styleUrls: ['./profile.component.css']

})

export class ProfileComponent implements OnInit {

user: Object;

constructor(private authService: AuthService) { }

ngOnInit() {

console.log('ngOnInit Called');

console.log(this.authService.loggedIn());

this.authService.getProfile().subscribe(data => {

console.log('Get Profile Called: ' + data);

this.user = data.user;

}, err => {

console.log('Error occured: ' + err.stack);

return false;

})

}

}

Register.component.html file

<div class="col-md-8 offset-md-2">

<h2 class="page-header">Register</h2>

<form>

<div class="form-group">

<label>Name</label>

<input type="text" [(ngModel)]="name" name="name" class="form-control" />

</div>

<div class="form-group">

<label>User Name</label>

<input type="text" [(ngModel)]="username" name="username" class="form-control" />

</div>

<div class="form-group">

<label>Email</label>

<input type="email" [(ngModel)]="email" name="email" class="form-control" />

</div>

<div class="form-group">

<label>Password</label>

<input type="password" [(ngModel)]="password" name="password" class="form-control" />

</div>

<button class="btn btn-primary" (click)="onRegisterSubmit()">Submit</button>

</form>

</div>

Register.component.ts file

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

import { AuthService } from 'src/app/services/auth.service';

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

@Component({

selector: 'app-register',

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

styleUrls: ['./register.component.css']

})

export class RegisterComponent implements OnInit {

name: string;

username: string;

email: string;

password: string;

submitted: boolean = false;

constructor(private authService: AuthService, private router: Router) { }

ngOnInit() {

}

onRegisterSubmit() {

this.submitted = true;

console.log(this.name);

let user = {

name: this.name,

username: this.username,

email: this.email,

password: this.password

}

console.log(user.name);

this.authService.registerUser(user)

.subscribe(data => {

alert('User registered');

console.log(data);

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

},

err => {

alert('Some Error Occuered');

console.log(err.stack);

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

}

)

}

}

Index.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JwtAuth</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<!-- <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.6.2/css/bulma.min.css">

<script defer src="https://use.fontawesome.com/releases/v5.0.0/js/all.js"></script> -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

</head>

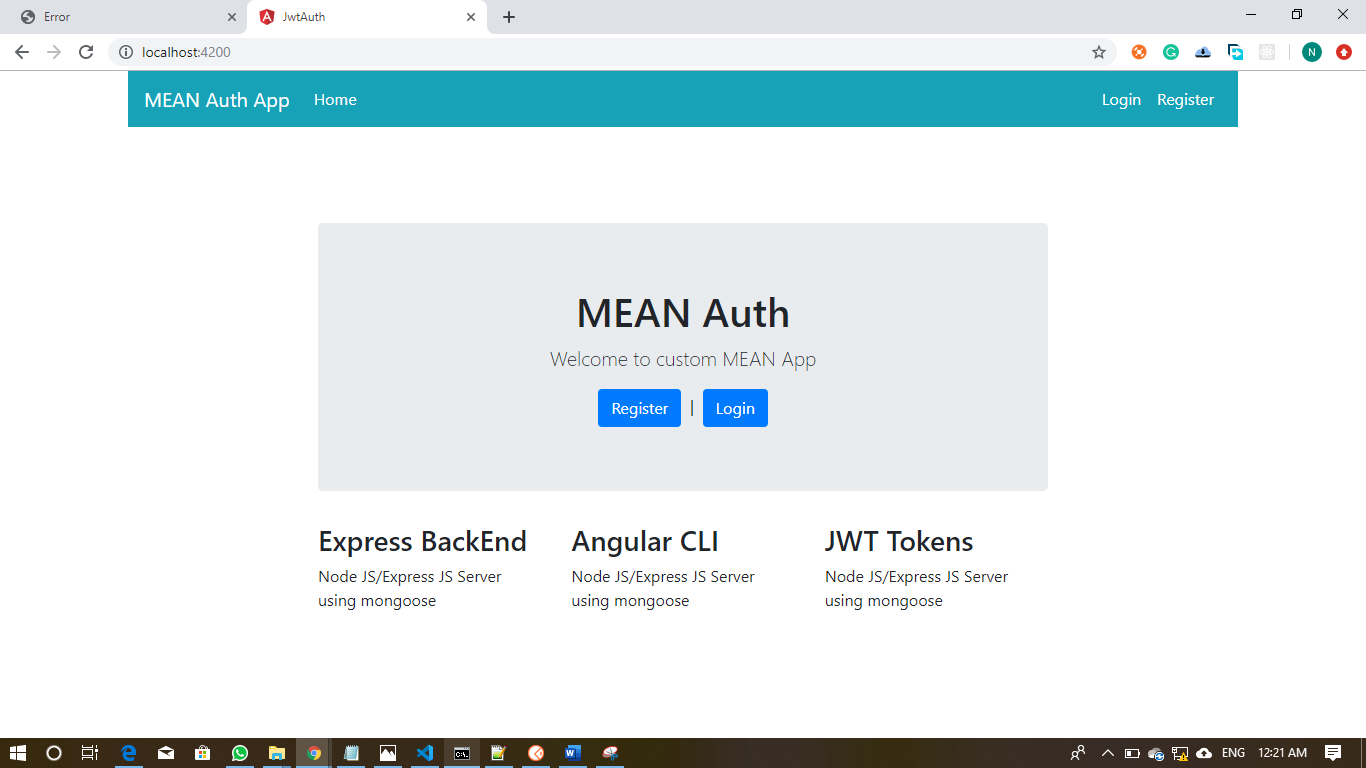
<body>

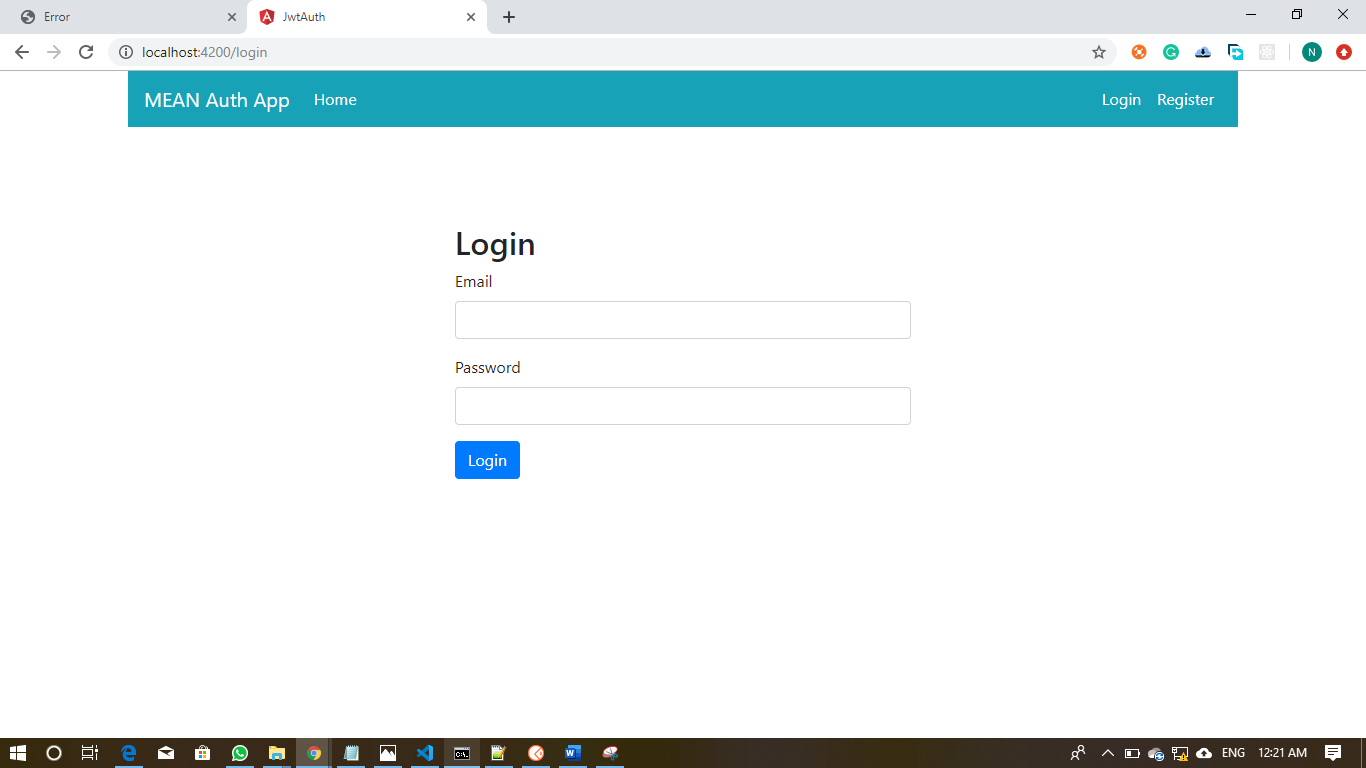
<app-root></app-root>

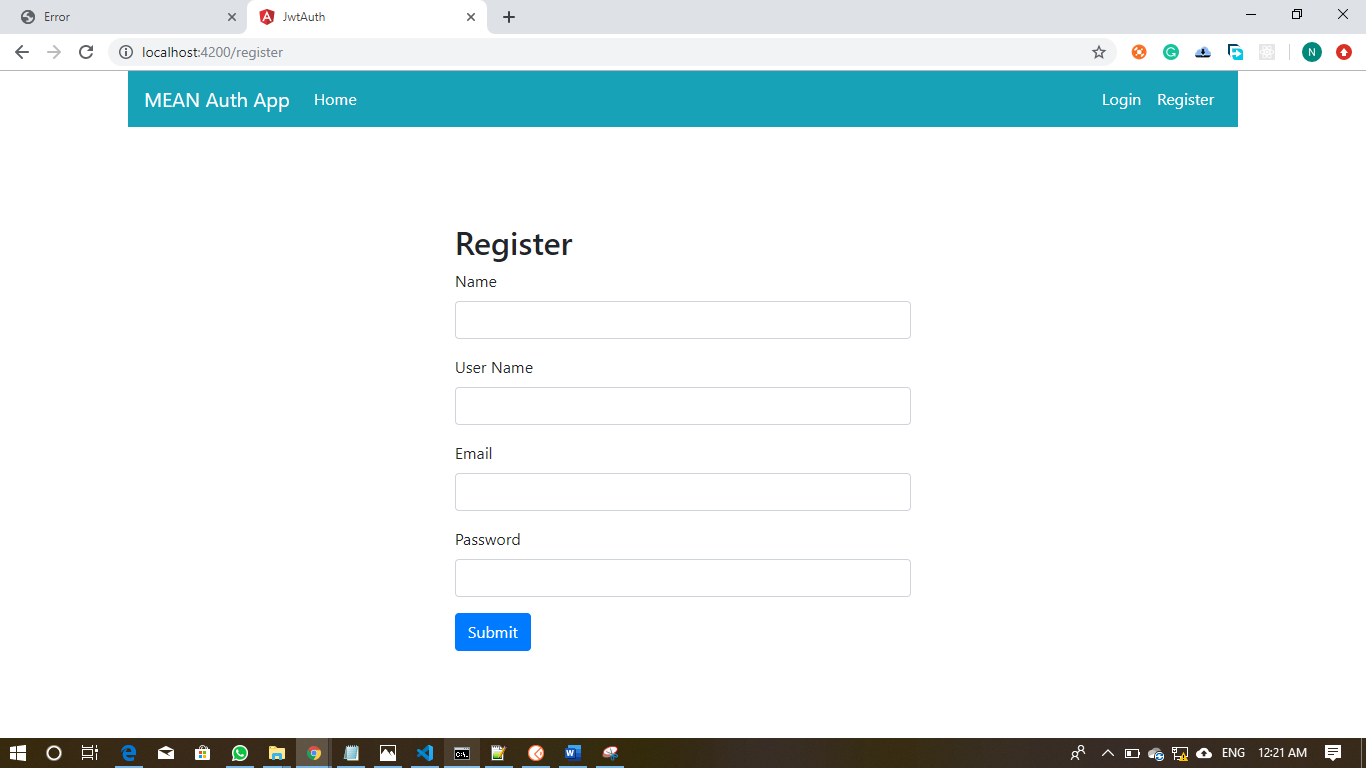
</body>

</html>

Output







Likewise you can see output of Profile ..