User Management App..!

1. Update npm to Latest version

npm i npm@latest -g

1. Check if old version of Angular is already installed on your computer, if exist then remove it

npm uninstall -g @angular/cli

npm cache clean

npm install -g @angular/cli

Note: To install a specific version, you can use npm install -g @angular/cli@1.4.9

1. Create new project

ng new user-management-app

or

skip to install node modules while creating project

ng new user-management-app –-skip-install

npm install [run this command separately to load all dependencies]

1. Run your application to get started

cd angular6-example

ng serve

1. Create Components

ng g component home

ng g component login --spec=false

ng g component add-user --spec=false

ng g component edit-user --spec=false

ng g component list-user --spec=false

Angular CLI Useful Commands

ng g component my-new-component

ng g directive my-new-directive

ng g pipe my-new-pipe

ng g service my-new-service

ng g class my-new-class

ng g guard my-new-guard

ng g interface my-new-interface

ng g enum my-new-enum

ng g module my-module

1. Configure Routing inside app.routing.ts

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

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

import {AddUserComponent} from "./add-user/add-user.component";

import {ListUserComponent} from "./list-user/list-user.component";

import {EditUserComponent} from "./edit-user/edit-user.component";

const routes: Routes = [

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

{ path: 'add-user', component: AddUserComponent },

{ path: 'list-user', component: ListUserComponent },

{ path: 'edit-user', component: EditUserComponent },

{path : '', component : LoginComponent}

];

export const routing = RouterModule.forRoot(routes);

// OR

Inside app-routing.module.ts file

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

// Import RouterModule

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

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

import { AddUserComponent } from './add-user/add-user.component';

import { ListUserComponent } from './list-user/list-user.component';

import { EditUserComponent } from './edit-user/edit-user.component';

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

// Each route maps a URL path to a component

const routes: Routes = [

// This route specifies the route to redirect to if the path

// is empty. In our case we are redirecting to /home

// pathMatch property value can be full or prefix. For now we

// will set it to full as we want to do a full match

{ path: '',component: HomeComponent },

{ path: 'home',component:HomeComponent},

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

{ path: 'add-user', component: AddUserComponent },

{ path: 'list-user', component: ListUserComponent },

// we can use localStorage

  // {path:'edit-user', component:EditUserComponent},

  // we can use routeParams

  { path: 'edit-user/:id', component: EditUserComponent },

//{ path: '',redirectTo:'/home',pathMatch:'full' },

// {path : '', component : LoginComponent}, // or

{ path: '\*\*', component: HomeComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

1. Create a JSON server that serves the data.

npm install -g json-server

1. Run below command on terminal

json-server --watch db.json

1. Add below code inside db.json file

{

"users": [

{

"id": 1,

"firstName": "Anand",

"lastName": "Prabhu",

"age": 25,

"mobileNumber": "9989568950",

"email": "anand-prabhu@gmail.com",

"password": "anand@123"

},

{

"firstName": "Suman",

"lastName": "Singh",

"age": 22,

"mobileNumber": "8956895689",

"email": "suman@gmail.com",

"password": "132456",

"id": 3

}

]

}

1. Run this on browser with the below given url

<http://localhost:3000/users>

1. Create new folder models and add new file user.model.ts

// Creating User Model class

export class User{

    id: number;

    firstName: string;

    lastName: string;

    age: number;

    mobileNumber: string;

    email: string;

    password: string;

}

1. Create new folder services inside app folder and then create user service by typing below command

ng g s user

and Add below code inside user.service.ts

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

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

import { User } from '../model/user.model';

@Injectable()

export class UserService {

constructor(private http:HttpClient) { }

baseUrl:string = 'http://localhost:3000/users';

 // GET - getAllUsers()

  getAllUsers() {

    return this.http.get<User[]>(this.baseUrl);

  }

  // GET - getUserById(id)

  getUserById(id: number) {

    return this.http.get<User>(this.baseUrl + "/" + id);

  }

  // POST - createNewUser(user)

  createNewUser(user: User) {

    return this.http.post(this.baseUrl, user);

  }

  // PUT - updateUserById(user)

  updateUserById(user: User) {

    return this.http.put(this.baseUrl + "/" + user.id, user);

  }

  // DELETE - deleteUserById(id)

  deleteUserById(id: number) {

    return this.http.delete(this.baseUrl + "/" + id);

  }

}

1. Create new folder images inside assets folder and add one image
2. Register your services and routing here, App.module.ts file

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

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

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

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

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

import { AddUserComponent } from './add-user/add-user.component';

import { EditUserComponent } from './edit-user/edit-user.component';

import { ListUserComponent } from './list-user/list-user.component';

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

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

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

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

@NgModule({

declarations: [

AppComponent,

LoginComponent,

AddUserComponent,

EditUserComponent,

ListUserComponent,

HomeComponent

],

imports: [

BrowserModule,

AppRoutingModule,

ReactiveFormsModule,

HttpClientModule

],

providers: [UserService],

bootstrap: [AppComponent]

})

export class AppModule { }

1. Modify App.component.html with below code

<div class="container-fluid">

    <div class="row">

        <div style="padding: 10px" class="col-lg-12 col-md-12 bg-info text-light">

            <marquee direction="right">

                <h2 class="text-center" style="margin:20px">

                    <!-- One Way Data Binding -->

                    <!-- {{ }} -> Interpolation Symbol or Binding Expression   -->

                    <!-- Built-in Pipes - lowercase, uppercase, titlecase [Angular 5]-->

                    <!-- {{title | lowercase}} -->

                    {{ title }}

                    <!-- NOTE: In case if you want to disable angular binding, you can

                        use built-in directive known as ngNonBindable-->

                    <!-- <span ngNonBindable>{{title}}</span> -->

                </h2>

            </marquee>

        </div>

    </div>

    <!-- end of header -->

    <div class="row">

        <div class="offset-lg-1 offset-md-1 col-lg-7 col-md-7">

            <!-- DYNAMIC RENDERING OF COMPONENTS -->

            <!-- It will act like a placeholder, where components will be replaced

            based on routes or navigation clicks -->

            <router-outlet></router-outlet>

            <!-- STATIC RENDERING -->

            <!-- <app-home></app-home>

            <app-home></app-home>

            <app-home></app-home> -->

        </div>

        <div class="col-lg-3 col-md-4 col-sm-12">

            <div style="margin-top:10%">

                <!-- <img src="assets/images/logo.png" style="width:100%" /> -->

                <img src="assets/images/Education.png" style="width:100%" />

                <!-- <img src="assets/images/kids\_education.jpg" style="width:100%" /> -->

            </div>

        </div>

    </div>

    <!-- end of row -->

    <br />

    <footer>

        <div class="row">

            <div class="offset-lg-1 offset-md-1 col-lg-7 col-md-7">

                <span class="text-danger">

                    <strong>This site is copyright &copy; to <mark>Syntel-AtoS</mark>

                        <!-- built-in date pipe -->

                        {{ todaysDate | date:'yyyy'}}.

                        All rights are reserved here.

                    </strong>

                </span>

            </div>

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

                <div class="text-center text-primary">

                    <!-- date: 'medium', 'shortTime', 'dd/MM/yy', 'dd-MM-yy, etc -->

                    <h4> {{todaysDate| date:'medium'}}</h4>

                </div>

            </div>

        </div>

        <!-- end of row -->

    </footer>

</div>

<!-- <h2>Developed By : {{ fullName }} </h2> -->

1. Add below code in app.component.ts file

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'AtoZ Solutions - User Management App';

  fullName = "Narendra G ";

  // creating date object

  todaysDate = new Date();

  // In Typescript, constructors cannot be overloaded

  // Only one constuctor per class

  constructor(){

    // setInterval() asynchronous function

    // Arrow functions ( => )

    setInterval(()=>{

      this.todaysDate = new Date();

    },1000);

  }

}

1. Add below code inside [home.component.html](http://home.component.html) file

<br/> <br/>

<div class="col-lg-12 col-md-12">

    <h3 class="text-primary">

        {{ title | titlecase}}

    </h3>

    <br/>

    <h5 class="text-warning"> In this demo, we will learn about </h5>

    <ul class="text-success font-weight-bold">

        <li>

            Angular {{64 | sqrt}} Architecture

            <ul>

                <li>Modules</li>

                <li>Root Component or App Component</li>

                <li>Services and Dependency Injection</li>

                <li>Directives</li>

                <li>Pipes</li>

                <li>Models</li>

                <li>Routing, etc</li>

            </ul>

        </li>

    </ul>

    <!-- This will reload entire page, but we want only component should be reloaded-->

    <!-- <a class="text-center" href="login"><h3>Login ..</h3></a> -->

    <!-- NOTE: we can achieve through angular directive known as [routerLink]-->

    <a class="text-right" routerLink="/login"><h3>Login ..</h3></a>

</div>

1. Add below code in [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 {

  // Implicitely typed, title will be of string type

  title = "welcome to angular 8";

 // NOTE: if you dont initialise any value to variable, Typescript will take undefined type

  // companyName;

  constructor() { }

  ngOnInit() {

  }

}

1. Modify login.component.html file

<!-- <p>login works!</p> -->

<!-- Built-in Pipes - currency pipe -> default "USD", "INR", "EUR", ETC -->

<!-- <h2>{{ 2-3+5\*6-6-8}}</h2> -->

<!-- <h2> {{ 25000 | currency: 'INR' }} </h2> -->

<!-- Built in Pipes - number -->

<!-- <h2> {{ 25000 | number :'5.2' }} </h2> -->

<!-- Built in Pipes - percent -->

<!-- <h2> {{ 00.54566 | percent  }} </h2> -->

<!-- <h2>{{custDetails.name}}</h2>

<h2>{{custDetails.address.city}}</h2> -->

<!-- Built in Pipes - json pipe -->

<!-- <h2>{{ custDetails | json}}</h2> -->

<!-- Built in Pipes - slice:start:end pipe -->

<!-- <h2>{{ months | slice:2:6}}</h2> -->

<br/>

<br/>

<div class="col-lg-7 col-md-7">

    <h3 class="text-primary">Login</h3>

    <br/>

    <!-- Form Handling in Angular Done in Two ways -->

    <!-- 1) Reactive Forms or Model Driven Forms -->

    <!-- How to identify Reactive Forms? -->

    <!-- [formGoup] directive with form tag, formControlName with input types -->

    <!-- 2) Template Driven Forms -->

    <!-- How to identify Template Driven Forms? -->

    <!-- ngForm directive with form tag, [(ngModel)] with input types -->

    <!-- LET's USE THE FIRST APPROACH -->

    <!-- 1) Reactive Forms or Model Driven Forms -->

    <!-- 4 types of BINGING in Angular -->

    <!-- {{title}} -> ONE WAY DATA BINDING Or INTERPOLATION or BINDING EXPRESSION,

            symbol is {{}} -->

    <!-- [formGroup] -> PROPERTY BINDING or ATTRIBUTE BINDING, symbol is [] -->

    <!-- (ngSubmit) -> EVENT BINDING , symbol is () -->

    <!-- [(ngModel)] -> TWO WAY DATA BINDING, symbol is [()] -->

    <!-- NOTE: Whenever we use REACTIVE FORMS, You have register with

        ReactiveFormsModule inside @NgModule decorator -->

    <form [formGroup]="loginForm" (ngSubmit)="verifyLogin()">

        <div class="form-group">

            <label for="email">Email Address : </label>

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

            placeholder="Email Address" />

            <!-- Predefined Structural Directive -->

            <!-- \*ngIf, \*ngFor, \*ngSwitchCase -->

            <div \*ngIf="submitted && loginForm.controls.email.errors" class="error">

                <div class="text-danger"

                \*ngIf="loginForm.controls.email.errors.required">

                        Email is required ..

                </div>

            </div>

        </div> <!-- end of email -->

        <div class="form-group">

            <label for="password">Password : </label>

            <input type="password" class="form-control" formControlName="password"

            placeholder="Password" />

            <div \*ngIf="submitted && loginForm.controls.password.errors" class="error">

                    <div class="text-danger"

                    \*ngIf="loginForm.controls.password.errors.required">

                            Password is required ..

                    </div>

                </div>

        </div> <!-- end of password -->

        <button class="btn btn-success">Login</button> &nbsp;

        <button class="btn btn-danger" type="reset">Cancel</button>  &nbsp;

        <a class="btn btn-outline-warning" routerLink="home">Go to Home</a> <br/><br/>

        <br/><br/>

        <div \*ngIf="invalidLogin">

            <div class="text-danger">

                Invalid Email or Password ..!

            </div>

        </div>

    </form>

</div>

1. Modify **login.component.ts file**

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

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

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

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  // creatinf FormGroup object

  loginForm: FormGroup;

  submitted: boolean= false;

  // Constructor Dependency Injection

  // Dependency Injection makes your application loosey typed

  // So that application can be easily maintained

  // FormBuilder to build form elements with defaut values and validations

  // Router service to navigate programmatically from component to other

  constructor(private formBuilder: FormBuilder, private router: Router) { }

  // Life Cycle Hook

  ngOnInit() {

    this.loginForm = this.formBuilder.group({

      email:['',Validators.required],

      password: ['', Validators.required]

    });

  }

  verifyLogin(){

    this.submitted = true;

    if(this.loginForm.invalid){

      return;

    }

    let username = this.loginForm.controls.email.value;

    let password = this.loginForm.controls.password.value;

    if(username == "admin@gmail.com" && password=="password@123")

    {

      localStorage.username = username;

      sessionStorage.username = username;

      this.router.navigate(['list-user']);

    }

    else{

      this.invalidLogin = true;

    }

  } // end of verifyLogin() function

  invalidLogin:boolean = false;

}

// Javascript object

  // custDetails = {

  //   name:'Mukund',

  //   age: 25,

  //   address: {

  //     city: 'Bangalore',

  //     state:'Karnataka'

  //   }

  // }

  // Array Data

  // months = ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"];

1. Modify list-user.component.html

<!-- <br/>

<br/>

<h2>Addition is {{sum}}</h2>

<h2>Multiplication is {{product}}</h2> -->

<br />

<br />

<div class="row">

    <div class="col-lg-5">

        <h2 class="text-info">User Details ..</h2>

        <br />

        <!-- Event Binding -->

        <button class="btn btn-outline-success" (click)="addUser()">Add User</button>

        &nbsp;&nbsp;

        <button class="btn btn-outline-danger" (click)="logOutUser()">LogOff</button>

        &nbsp;&nbsp;

        <br />

        <br />

    </div>

    <!-- FOR SEARCH OPERATION -->

    <div class="col-lg-6">

        <!-- Two Way Data Binding -->

        <div class="form-group" style="margin-top:5%">

            <label for="firstName" class="text-primary font-weight-bold">

                Search By First Name :

            </label>

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

            <br />

            <h6>Searching For {{ searchText }}</h6>

        </div>

    </div>

</div>

<!-- LIST USERS -->

<div class="row">

    <div class="col-lg-12 col-md-12">

        <ng-container \*ngIf="(users | search:searchText) as result">

            <table class="table table-striped table-responsive table-hover">

                <thead>

                    <tr>

                        <th class="text-hide">Id</th>

                        <th>First Name</th>

                        <th>Last Name</th>

                        <!-- <th>Age</th> -->

                        <th>Email</th>

                        <th>Mobile Number</th>

                        <th>Action</th>

                    </tr>

                </thead>

                <!-- \*ngFor -> Structural Directive -->

                <tbody>

                    <!-- <tr \*ngFor="let user of users"> -->

                    <!-- <tr \*ngFor="let user of users; let i = index"> -->

                    <!-- <tr \*ngFor="let user of users | search:searchText | slice:0:3">         -->

                    <tr \*ngFor="let user of result">

                        <td class="text-hide">{{user.id}}</td>

                        <td>{{user.firstName}}</td>

                        <td>{{user.lastName}}</td>

                        <!-- <td>{{user.age}}</td> -->

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

                        <td>{{user.mobileNumber}}</td>

                        <td>

                            <button class="btn btn-outline-warning" type="button" (click)="editUser(user)">Edit</button>

                            &nbsp; &nbsp;

                            <button class="btn btn-outline-danger" type="button"

                                (click)="deleteUser(user)">Delete</button>

                            <!-- <button class="btn btn-outline-danger" type="button"

                        (click)="deleteUser(user, i)">Delete</button> -->

                        </td>

                    </tr>

                    <tr>

                        <td \*ngIf="result.length==0" colspan="7">

                            <h5 class="text-center text-danger">

                                Sorry, User does not exist ..!</h5>

                        </td>

                    </tr>

                    <tr \*ngIf="!users || users.length==0">

                        <th colspan="7">

                            <p class="text-right text-danger">

                                Sorry, No Users Available ..!

                            </p>

                        </th>

                    </tr>

                </tbody>

            </table>

        </ng-container>

    </div>

</div>

1. Modify **list-user.component.ts**

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

import { CalcService } from '../../services/calc.service';

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

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

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

@Component({

  selector: 'app-list-user',

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

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

})

export class ListUserComponent implements OnInit {

  // searchText = "Amit";

  searchText:any;

  // to handle list of users

  users: User[];

  // injecting UserService which is using HttpClient service

  constructor(private userService: UserService, private router: Router) { }

  // Life cycle hook

  ngOnInit() {

    if (localStorage.username != null) {

      this.getUsers();

    }

    else {

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

    }

  }

  getUsers() {

    this.userService.getAllUsers().subscribe(data => {

      // on resolve or on success

      this.users = data;

      console.log(this.users);

    },

      err => {

        // on reject or on error

        console.log(err.stack);

      });

  }

  // logOutUser() function

  logOutUser() {

    if (localStorage.username != null) {

      localStorage.removeItem("username");

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

    }

  }

  // addUser() function

  addUser() {

    this.router.navigate(['add-user']);

  }

  // editUser(user) function

  editUser(user){

    this.router.navigate(['edit-user',user.id]);

  }

  // deleteUser(user) function

  deleteUser(user: User) {

    let result = confirm("Do you want to delete user?");

    if (result) {

      this.userService.deleteUserById(user.id)

        .subscribe(data => {

          this.users = this.users.filter(u => u.id !== user.id);

        },

          err => {

            console.log(err.stack);

          });

          // alert(user.firstName+" record is deleted ..!");

          alert(`${user.firstName}  record is deleted ..!`);

    }

  }

  // deleteUser(user: User, index) {

  //   let result = confirm("Do you want to delete user?");

  //   if (result) {

  //     this.userService.deleteUserById(user.id).subscribe(data => {

  //       // this.users.splice(index, 1);

  //       this.getUsers();

  //     },

  //       err => {

  //         console.log(err.stack);

  //       });

  //       // alert(user.firstName+" record is deleted ..!");

  //       alert(`${user.firstName}  record is deleted ..!`);

  //   }

  // }

  // UNDERSTANDING CalcService

  // sum: number;

  // product: number;

  // // Injecting CalcService

  // constructor(private calcService: CalcService) { }

  // ngOnInit() {

  //   this.sum = this.calcService.addition(100,200);

  //   this.product = this.calcService.multiplication(100,200);

  // }

}

1. Modify **add-user.component.html**

<br />

<br />

<div class="col-md-10 col-lg-10">

    <h3 class="text-info">Add User</h3>

    <br />

    <!-- Reactive Forms -->

    <!-- Designing Add User Form -->

    <form [formGroup]="addForm" (ngSubmit)="addUser()">

        <!-- First Name Text Box -->

        <div class="form-group row">

            <label for="firstName" class="col-lg-4 text-right font-weight-bold">

                First Name :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="First Name" formControlName="firstName" />

                <div \*ngIf="submitted && addForm.controls.firstName.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.firstName.errors.required">

                        First Name is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="addForm.controls.firstName.errors.pattern">

                        FirstName must begin with Capital letter and 15 characters long ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  First Name Text Box -->

        <!-- Last Name Text Box -->

        <div class="form-group row">

            <label for="lastName" class="col-lg-4 text-right font-weight-bold">

                Last Name :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="Last Name" formControlName="lastName" />

                <div \*ngIf="submitted && addForm.controls.lastName.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.lastName.errors.required">

                        Last Name is required ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Last Name Text Box -->

        <!-- Age Text Box -->

        <div class="form-group row">

            <label for="age" class="col-lg-4 text-right font-weight-bold">

                Age :

            </label>

            <div class="col-lg-8">

                <input type="number" class="form-control " placeholder="Age" formControlName="age" />

                <div \*ngIf="submitted && addForm.controls.age.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.age.errors.required">

                        Age is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="addForm.controls.age.errors.min || addForm.controls.age.errors.max">

                        Age must be between 20 and 30 ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Age Text Box -->

        <!-- Mobile Number Text Box -->

        <div class="form-group row">

            <label for="mobileNumber" class="col-lg-4 text-right font-weight-bold">

                Mobile Number :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="Mobile Number" formControlName="mobileNumber" />

                <div \*ngIf="submitted && addForm.controls.mobileNumber.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.mobileNumber.errors.required">

                        Mobile Number is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="addForm.controls.mobileNumber.errors.pattern">

                        Mobile Number must be of 10 digits and should be numeric ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Mobile Number Text Box -->

        <!-- Email Text Box -->

        <div class="form-group row">

            <label for="email" class="col-lg-4 text-right font-weight-bold">

                Email :

            </label>

            <div class="col-lg-8">

                <input type="email" class="form-control " placeholder="e.g: abc@example.com" formControlName="email" />

                <div \*ngIf="submitted && addForm.controls.email.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.email.errors.required">

                        Email is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="addForm.controls.email.errors.email">

                        Invalid Email Address ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Mobile Number Text Box -->

        <!-- Password Text Box -->

        <div class="form-group row">

            <label for="password" class="col-lg-4 text-right font-weight-bold">

                Password :

            </label>

            <div class="col-lg-8">

                <input type="password" class="form-control " placeholder="Password" formControlName="password" />

                <div \*ngIf="submitted && addForm.controls.password.errors">

                    <div class="text-danger" \*ngIf="addForm.controls.password.errors.required">

                        Password is required ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Password Text Box -->

        <div class="text-right">

            <button class="btn btn-outline-success">Save</button> &nbsp; &nbsp;

            <button class="btn btn-outline-danger" type="reset">Cancel</button> &nbsp; &nbsp;

            <a class="btn btn-link" routerLink="/list-user">Back to List</a>

        </div>

    </form>

</div>

1. **Modify add-user.component.ts**

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

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

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

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

@Component({

  selector: 'app-add-user',

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

  styleUrls: ['./add-user.component.css']

})

export class AddUserComponent implements OnInit {

  addForm: FormGroup;

  submitted: boolean = false;

  // Injecting all relevant services inside the constructor

  constructor(private formBuilder: FormBuilder,

    private router: Router, private userService: UserService) { }

  // Life Cycle Hook

  ngOnInit() {

    if (localStorage.username != null) {

      this.addForm = this.formBuilder.group({

        firstName: ['', [Validators.required, Validators.pattern("[A-Z][a-z]{2,14}")]],

        lastName: ['', Validators.required],

        age: ['', [Validators.required, Validators.min(20), Validators.max(30)]],

        mobileNumber: ['', [Validators.required, Validators.pattern("[6-9][0-9]{9}")]],

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

        password: ['', Validators.required]

      });

    }

    else {

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

    }

  }

  // addUser() function

  addUser() {

    this.submitted = true;

    if (this.addForm.invalid) {

      return;

    }

    // on successful validations, execute below code snippet

    console.log(this.addForm.value);

    this.userService.createNewUser(this.addForm.value).subscribe(data => {

      alert(`${this.addForm.controls.firstName.value} record is added successfully ..!`);

      this.router.navigate(['list-user']);

    }, err => {

      console.log(err.stack);

    })

  }

}

1. Modify **edit-user.component.html**

<br/>

<br/>

<div class="col-md-10 col-lg-10">

    <h3>Edit User &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

        <button class="btn btn-danger" (click)="logOutUser()">LogOff</button>

    </h3>

    <br/>

    <!-- Reactive Forms -->

    <!-- Designing Edit User Form -->

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

        <!-- First Name Text Box -->

        <div class="form-group row">

            <label for="firstName" class="col-lg-4 text-right font-weight-bold">

                First Name :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="First Name" formControlName="firstName" />

                <div \*ngIf="submitted && editForm.controls.firstName.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.firstName.errors.required">

                        First Name is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="editForm.controls.firstName.errors.pattern">

                        FirstName must begin with Capital letter and 15 characters long ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  First Name Text Box -->

        <!-- Last Name Text Box -->

        <div class="form-group row">

            <label for="lastName" class="col-lg-4 text-right font-weight-bold">

                Last Name :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="Last Name" formControlName="lastName" />

                <div \*ngIf="submitted && editForm.controls.lastName.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.lastName.errors.required">

                        Last Name is required ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Last Name Text Box -->

        <!-- Age Text Box -->

        <div class="form-group row">

            <label for="age" class="col-lg-4 text-right font-weight-bold">

                Age :

            </label>

            <div class="col-lg-8">

                <input type="number" class="form-control " placeholder="Age" formControlName="age" />

                <div \*ngIf="submitted && editForm.controls.age.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.age.errors.required">

                        Age is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="editForm.controls.age.errors.min || editForm.controls.age.errors.max">

                        Age must be between 20 and 30 ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Age Text Box -->

        <!-- Mobile Number Text Box -->

        <div class="form-group row">

            <label for="mobileNumber" class="col-lg-4 text-right font-weight-bold">

                Mobile Number :

            </label>

            <div class="col-lg-8">

                <input type="text" class="form-control " placeholder="Mobile Number" formControlName="mobileNumber" />

                <div \*ngIf="submitted && editForm.controls.mobileNumber.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.mobileNumber.errors.required">

                        Mobile Number is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="editForm.controls.mobileNumber.errors.pattern">

                        Mobile Number must be of 10 digits and should be numeric ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Mobile Number Text Box -->

        <!-- Email Text Box -->

        <div class="form-group row">

            <label for="email" class="col-lg-4 text-right font-weight-bold">

                Email :

            </label>

            <div class="col-lg-8">

                <input type="email" class="form-control " placeholder="e.g: abc@example.com" formControlName="email" />

                <div \*ngIf="submitted && editForm.controls.email.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.email.errors.required">

                        Email is required ..!

                    </div>

                    <div class="text-danger" \*ngIf="editForm.controls.email.errors.email">

                        Invalid Email Address ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Mobile Number Text Box -->

        <!-- Password Text Box -->

        <div class="form-group row">

            <label for="password" class="col-lg-4 text-right font-weight-bold">

                Password :

            </label>

            <div class="col-lg-8">

                <input type="password" class="form-control " placeholder="Password" formControlName="password" />

                <div \*ngIf="submitted && editForm.controls.password.errors">

                    <div class="text-danger" \*ngIf="editForm.controls.password.errors.required">

                        Password is required ..!

                    </div>

                </div>

            </div>

        </div>

        <!-- End of  Password Text Box -->

        <div class="text-right">

            <button class="btn btn-outline-success">Update</button> &nbsp; &nbsp;

            <a class="btn btn-link" routerLink="/list-user">Cancel</a>

        </div>

    </form>

</div>

1. Modify **edit-user.component.ts**

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

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

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

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

@Component({

  selector: 'app-edit-user',

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

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

})

export class EditUserComponent implements OnInit {

  editForm: FormGroup;

  submitted: boolean = false;

  currentId: number;

  // Injecting all relevant services inside the constructor

  constructor(private formBuilder: FormBuilder,

    private router: Router, private userService: UserService,

    private route: ActivatedRoute) {

    this.route.params.subscribe(params => {

      this.currentId = params['id'];

    })

    console.log(this.currentId);

  }

  // Life Cycle Hook

  ngOnInit() {

    if (localStorage.username != null) {

      this.editForm = this.formBuilder.group({

        id: [this.currentId],

        firstName: [{ value: '', disabled: true }, [Validators.required, Validators.pattern("[A-Z][a-z]{2,14}")]],

        lastName: [{ value: '', disabled: true }, Validators.required],

        age: ['', [Validators.required, Validators.min(20), Validators.max(30)]],

        mobileNumber: ['', [Validators.required, Validators.pattern("[6-9][0-9]{9}")]],

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

        password: ['', Validators.required]

      });

      // pulling data from service by id

      this.userService.getUserById(this.currentId).subscribe(data => {

        console.log(data);

        // binding josn response to form object

        this.editForm.setValue(data);

      },

        err => {

          console.log(err.stack);

        })

    }

    else {

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

    }

  }

  // logOutUser() function

  logOutUser() {

    if (localStorage.username != null) {

      localStorage.removeItem("username");

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

    }

  }

  // updateUser() function

  updateUser() {

    this.submitted = true;

    if (this.editForm.invalid) {

      return;

    }

    this.userService.updateUserById

      (this.editForm.getRawValue())

      .subscribe(data => {

        this.router.navigate(['list-user']);

      }, error => {

        console.log(error.stack);

      });

  }

}

1. Adding name custom pipe

* ng g p search --spec=false

Add below code in name.pipe.ts file

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'search'

})

export class SearchPipe implements PipeTransform {

  transform(usersList: any, searchText:any): any {

    let newList: any;

    if(searchText){

      newList = usersList.filter(user=> user.firstName.toLowerCase()

        .startsWith(searchText.toLowerCase()));

    }

    else{

      newList = usersList;

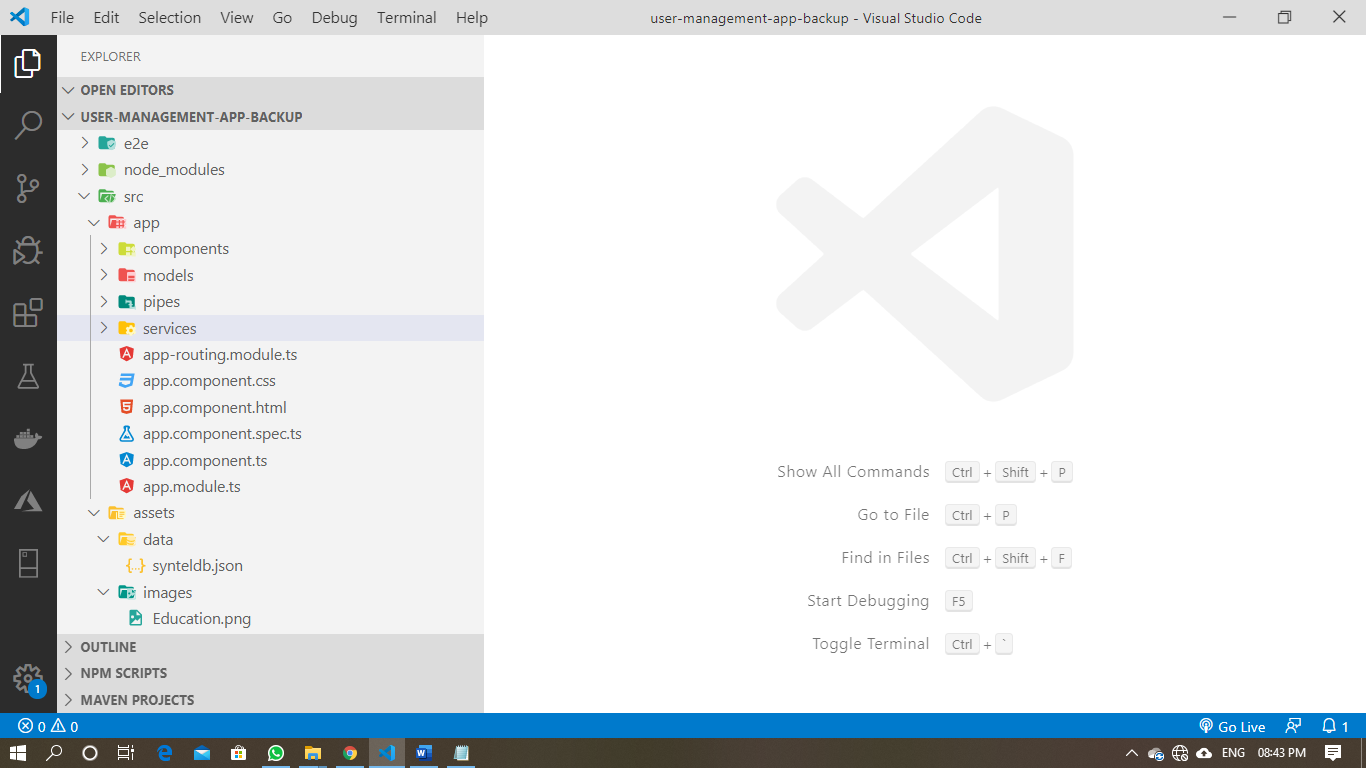
    }

    return newList;

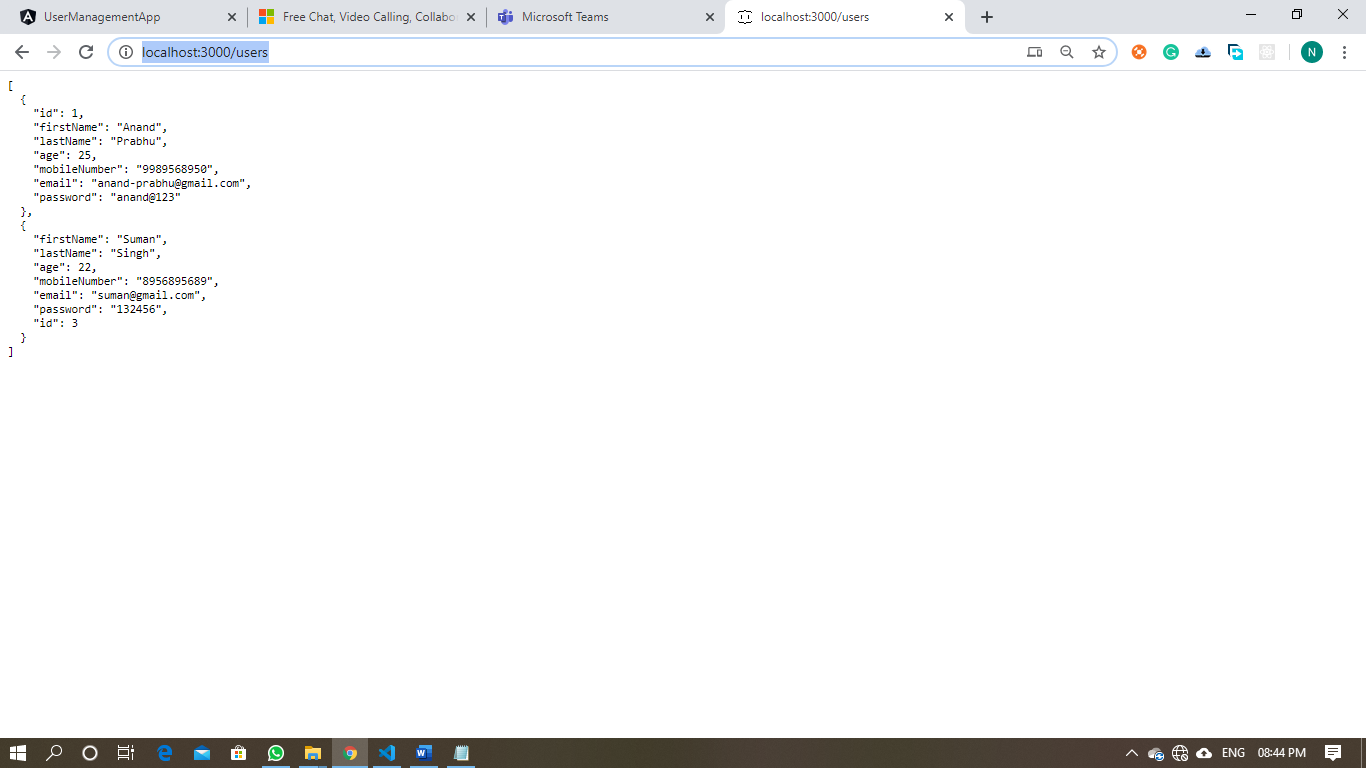
  }

}

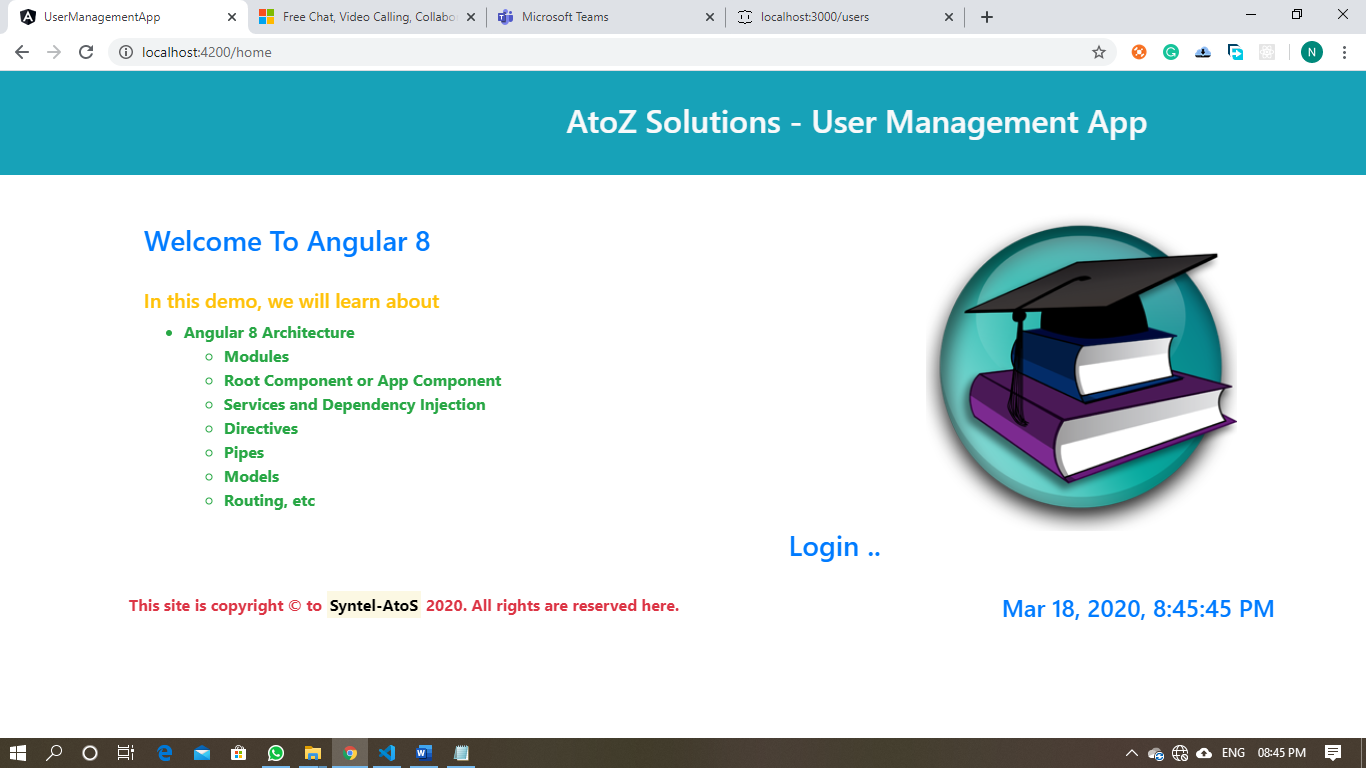
1. Testing Angular Application

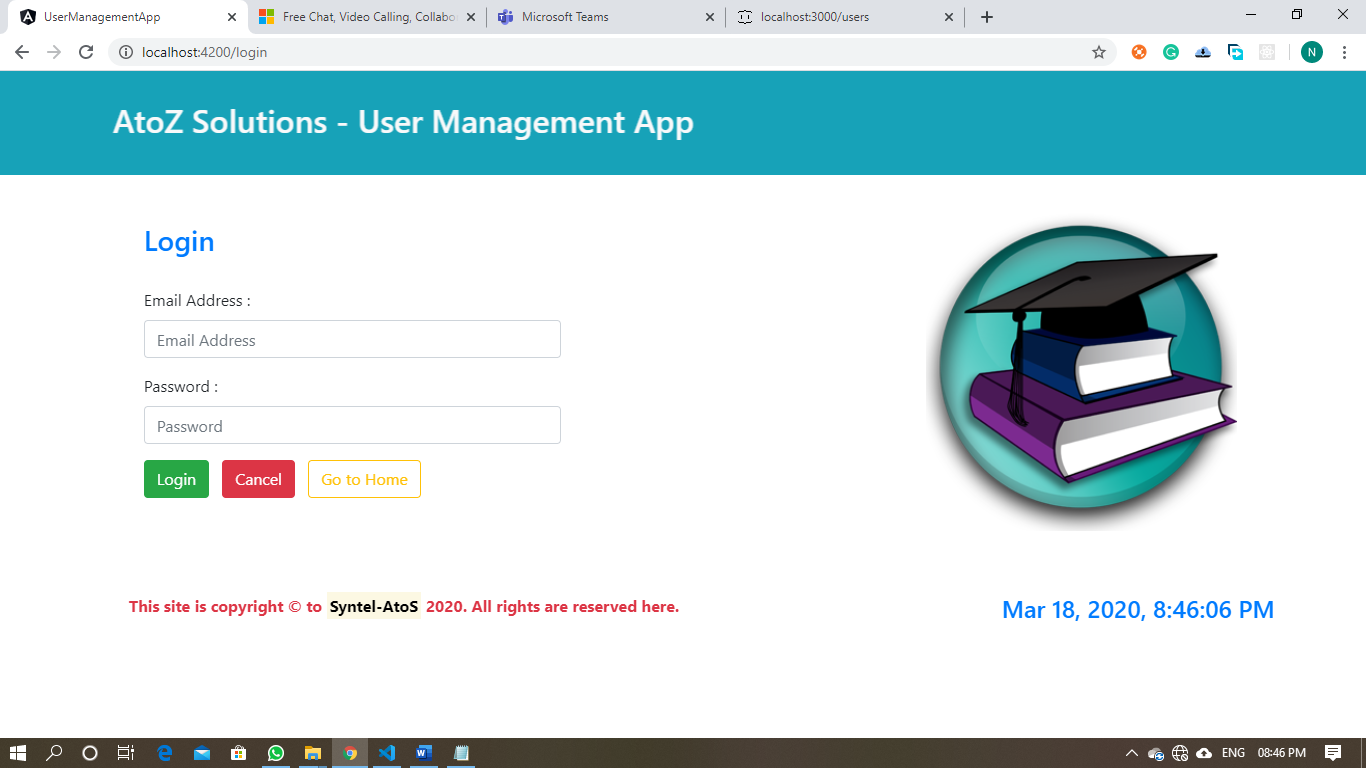


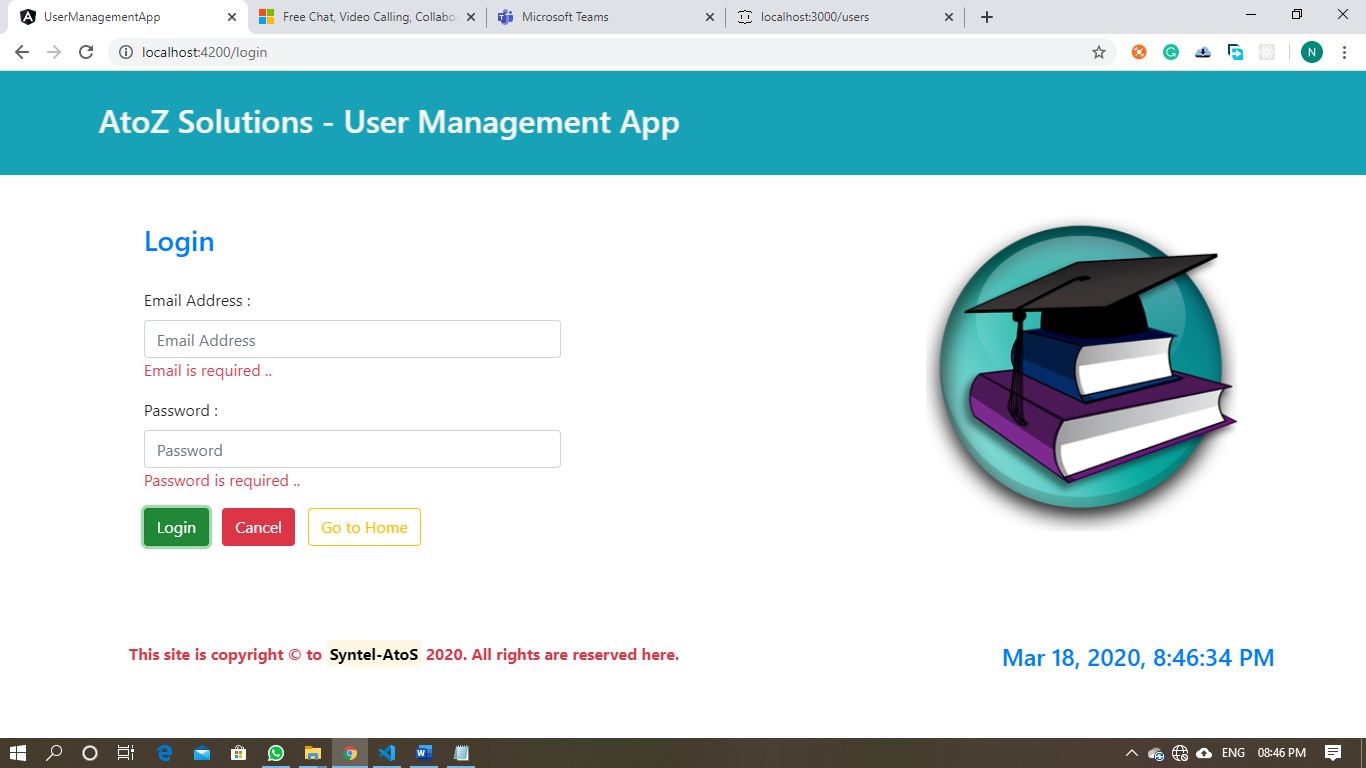
json-server –watch db.json

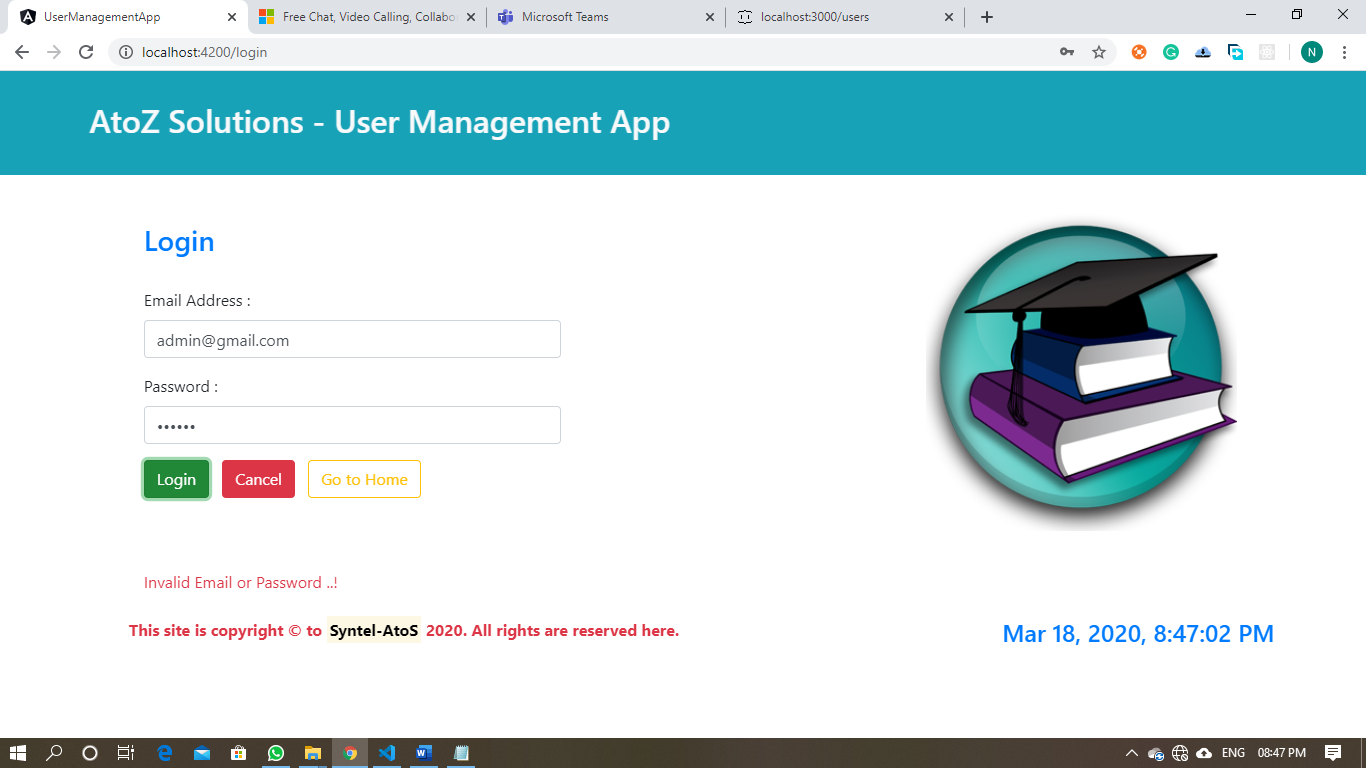
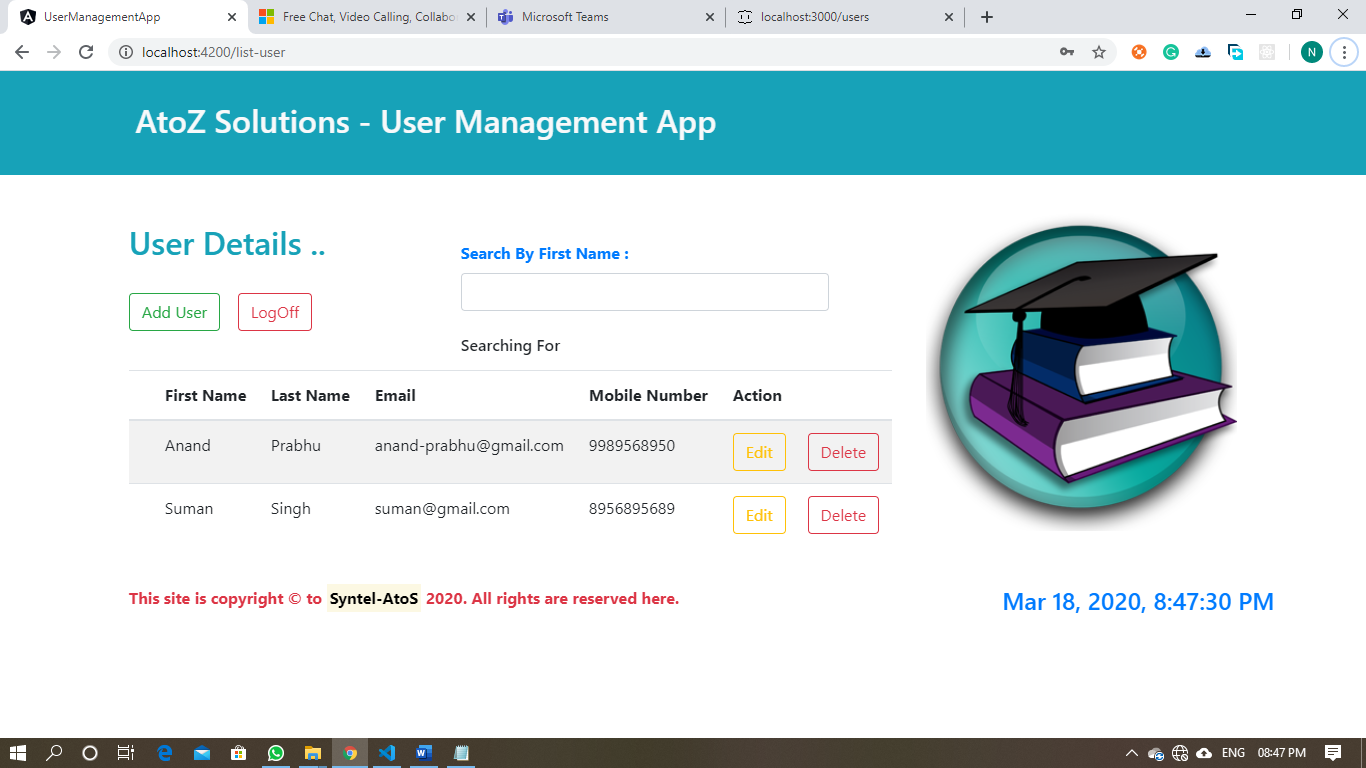
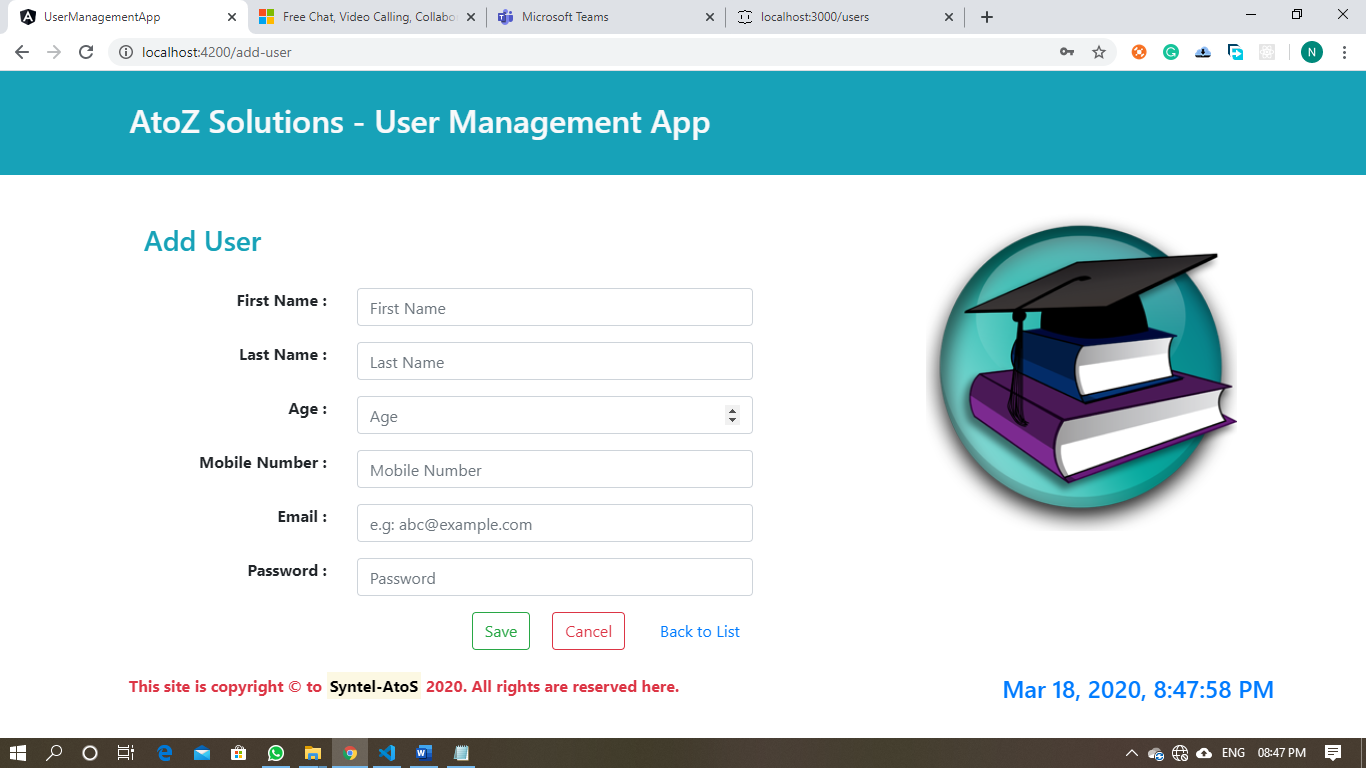


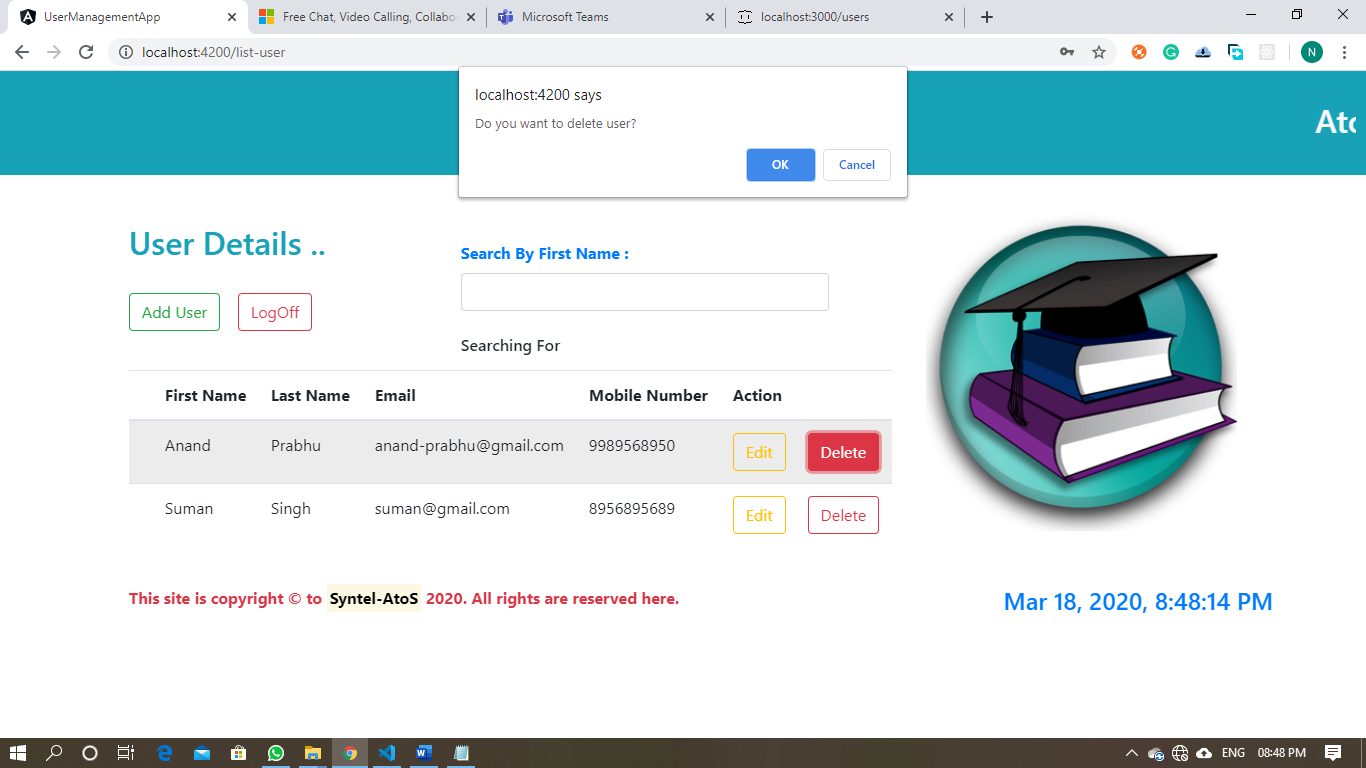
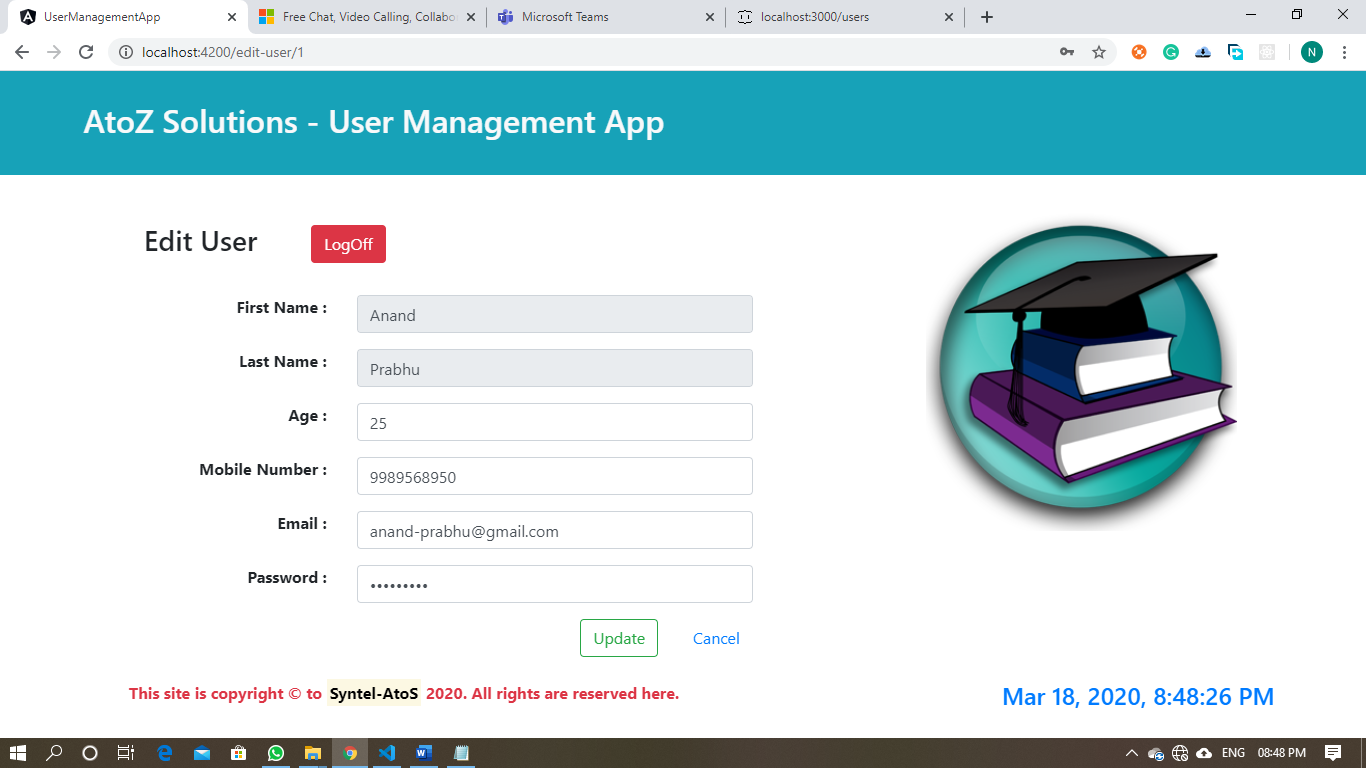
ng serve







Checking search pipe filter

