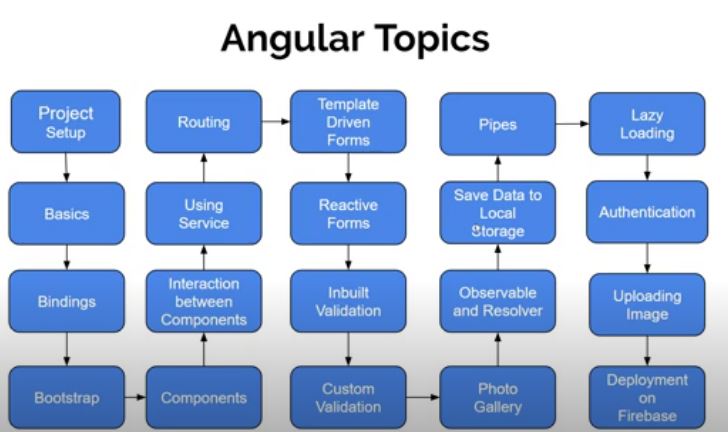
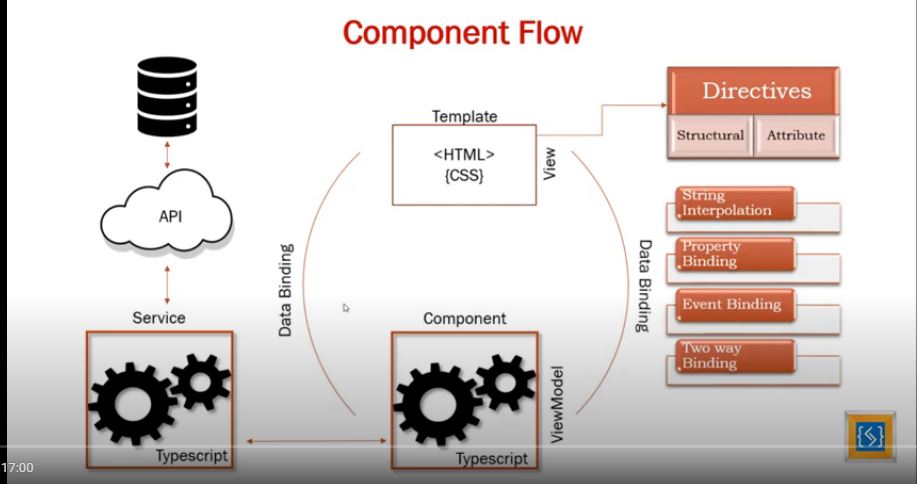
**Real Estate Agency using Angular 11 and Asp.Net core web api and boostrap 4**

These are 2 of the hottest frameworks right now for the ‘back-end’ (Microsoft’s ASP.NET Core) and the ‘front-end’ (Google’s Angular) and are well worth spending the time to learn. This course starts from scratch, you neither need to know Angular 1 nor Angular 2 We will start from nothing and incrementally build this property dealing application front-end using Angular 10. And then we will connect our front-end with the Web-API until we have a fully functional Web Application that we will publish to Firebase and then on IIS. Here are some of the things you will learn about in this course: Creating the Angular App and ASP.NET Core WebAPI using the DotNet CLI and the Angular CLI Adding a Client side login and register function to our Angular application Adding routing to the Angular application and securing routes. Using Automapper in ASP.NET Core Building a great looking UI using Bootstrap Adding Photo Upload functionality as well as a cool looking gallery in Angular Angular Template forms and Reactive forms and validation Paging, Sorting and Filtering Publishing the application to Firebase, IIS and then on Azure.

****

First we install the vs code extension write angular essential (john papa) and install it .



**Components:**

First we create components of our project we make parent and child component relations

Firstly we make a new folder in app name property then we add a component property-card component (child) and then we add a component property-list component(parent) also we create a component navbar they are not include in property folder remember.

**Property-card Component:(Child)**

In property- card component we show the list of houses in column wise where the card show the detail of house name ,number and contact so we created input decorator to tell the parent component about property of house so they use of input using databinding and loop and array to show multiple houses.

**Property-card.component.ts**

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

@Component({

  selector: 'app-property-card',

  templateUrl: './property-card.component.html',

  styleUrls: ['./property-card.component.css']

})

export class PropertyCardComponent implements OnInit {

  @Input() propertynames:any

  constructor() { }

  ngOnInit(): void {

  }

}

**Property-card.component.html**

  <div class="card">

    <div class="card-img-wrapper">

<img class="card-img-top" src="assets/images/house.jpg">

<ul class="list-inline text-center member-icons animate">

  <li class="list-inline-item">

    <button class="btn btn-primary">

      <i class="far fa-edit"></i> <!--font awesome icon used-->

    </button>

  </li>

    <li class="list-inline-item">

<button class="btn btn-primary">

  <i class="far fa-address-card"></i><!--font awesome icon used-->

    </button>

  </li>

</ul>

</div>

<div class="card-body p-2">

<h3>{{propertynames.Name}}</h3>

<h3 >{{propertynames.Type}}</h3>

<h3>{{propertynames.Price}}</h3>

</div>

</div>

**Property-card.component.css**

.h{

  color:green;

}

.propImg{

  height: 250px;

}

.card:hover img{

  transform: scale(1.2,1.2);

  transition-duration: 500ms;

  transition-timing-function: ease-out;

}

.card img{

  transform: scale(1,1);

  transition-duration: 500ms;

  transition-timing-function: ease-out;

}

.card-img-wrapper{

  overflow: hidden;

  position: relative;

}

 .member-icons{

  position: absolute;

  bottom: -30%;

  right: 0;

  left: 0;

  margin-right: auto;

  margin-left: auto;

}

.card-img-wrapper:hover .member-icons{

  bottom: 0;

}

.animate{

  transition: all 0.3s ease-in-out;

}

**Property-list.component.ts(Parent)**

Where we fetch the data of our child component using @input decorator first we make an array of object wo make 6 houses data then in html file we using structural directives ngFor to loop through the array data and using databinding to communicate our child component.

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

@Component({

  selector: 'app-property-list',

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

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

})

export class PropertyListComponent implements OnInit {

  Properties: Array<any> = [

    {

    "Id":1,

    "Name": "Zaryab House",

    "Type":"House",

    "Price": 1200000

  },

  {

    "Id":2,

    "Name": "Shahroz House",

    "Type":"Plot",

    "Price": 2000000

  },

  {

    "Id":3,

    "Name": "Ahsan House",

    "Type":"Flat",

    "Price": 1300000

  },

  {

    "Id":4,

    "Name": "Minaj House",

    "Type":"bugalow",

    "Price": 2400000

  },

  {

    "Id":5,

    "Name": "Minaj House",

    "Type":"bugalow",

    "Price": 2400000

  },

  {

    "Id":6,

    "Name": "Minaj House",

    "Type":"bugalow",

    "Price": 2400000

  },

]

  constructor() { }

  ngOnInit(): void {

  }

}

**Property-list.component.html**

<div class="row">

  <div \*ngFor='let prop of Properties ' class="shadow-lg col-sm-4 p-3">

<app-property-card  [propertynames]= "prop" ></app-property-card>

</div>

</div>

**App.component.html**

<app-nav-bar></app-nav-bar>

<div class="container" style="margin-top: 70px;">

<app-property-list></app-property-list>

</div>

**Navbar.component.html**

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

  <a class="navbar-brand" href="#">Real Estate</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

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

  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item active">

        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="#">Link</a>

      </li>

      <li class="nav-item dropdown">

        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

          Dropdown

        </a>

        <div class="dropdown-menu" aria-labelledby="navbarDropdown">

          <a class="dropdown-item" href="#">Action</a>

          <a class="dropdown-item" href="#">Another action</a>

          <div class="dropdown-divider"></div>

          <a class="dropdown-item" href="#">Something else here</a>

        </div>

      </li>

      <li class="nav-item">

        <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>

      </li>

    </ul>

    <form class="form-inline my-2 my-lg-0">

      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

    </form>

  </div>

</nav>

Chapter 2

Using Service and HTTP calls in Angular 9

We use HTTP Calls to fetch the data for our properties. So first we add new folder in src named data then add a new files properties.json and write our house data here

[

  {

  "Id":1,

  "Name": "Zaryab House",

  "Type":"House",

  "Price": 1200000

},

{

  "Id":2,

  "Name": "Shahroz House",

  "Type":"Plot",

  "Price": 2000000

},

{

  "Id":3,

  "Name": "Ahsan House",

  "Type":"Flat",

  "Price": 1300000

},

{

  "Id":4,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000

},

{

  "Id":5,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000

},

{

  "Id":6,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000

}

]

Now we add a new folder services in app folder and using cli add a new service ng g service housing

And write a following the code here

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

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

@Injectable({

  providedIn: 'root'

})

export class HousingService {

  constructor(private http: HttpClient) { }

  getAllProperties(){

return this.http.get('data/properties.json')

  }

}

And some changes in property.list.component.ts

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

import { HousingService } from 'src/app/services/housing.service';

@Component({

  selector: 'app-property-list',

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

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

})

export class PropertyListComponent implements OnInit {

  Properties: any ;

  constructor(private housingService:HousingService ) { }

  ngOnInit(): void {

    this.housingService.getAllProperties().subscribe(

      data =>{

             this.Properties=data;

               console.log(data);

             }, error =>{

               console.log(error);

               //This error condition is used in when api server is down and sometimes database is down so thats why use the conditions

             }

    );

}

}

And all same as previous no change in other component then run it will perfectly fine.

# Modify HTTP Data using Pipe

We are using Pipe in service to catch the error In early stage of development and also replace the any type in component and service because this is not good so that’s why we use pipe

To replace this any type add a new file in Property folder

IProperty.interface.ts

export interface IProperty{

  Id: number;

  Name: string;

  Type: string;

  Price: number;

}

The benefit is using interface instead of any when you chech view and press . they give you intellisence of property to save the typo mistake like name and names so save this

Now we use the pipe also in services here

**housing service.ts**

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

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

import {map} from 'rxjs/operators';

import { IProperty } from '../property/IProperty.interface';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class HousingService {

  constructor(private http: HttpClient) { }

  getAllProperties():Observable<IProperty[]>{

return this.http.get('data/properties.json').pipe(

map(data=>{

  const propertiesarray: Array<IProperty> = [];

  for(const id in data) {

    if(data.hasOwnProperty(id)){

      propertiesarray.push(data[id]);

    }

  }

  return propertiesarray

})

);

  }

}

Now we replace the any in **propertylist.component.ts**

export class PropertyListComponent implements OnInit {

 // Properties: Array<any> ;

  Properties: Array<IProperty> ;

Same do in the **propertycard.ts**

And then run

# Understand Routing - Part-1

# Now we add new component in property folder name add-property because when user click the property-list so it will go to add- property

# Now changes in app.module.ts

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

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

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

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

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

import { PropertyListComponent } from './property/property-list/property-list.component';

import { PropertyCardComponent } from './property/property-card/property-card.component';

import { NavBarComponent } from './nav-bar/nav-bar.component';

import { HousingService } from './services/housing.service';

import { AddPropertyComponent } from './property/add-property/add-property.component';

const appRoutes : Routes = [

  {path:'', component:PropertyListComponent},

{path: 'add-property', component:AddPropertyComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    PropertyListComponent,

    PropertyCardComponent,

    NavBarComponent,

    AddPropertyComponent,

  ],

  imports: [

    BrowserModule,

    HttpClientModule,

    RouterModule.forRoot(appRoutes)

  ],

  providers: [HousingService],

  bootstrap: [AppComponent]

})

export class AppModule { }

Now change in navbar remove dropdown and rename the menus in navbars

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

  <a class="navbar-brand" href="/">Real Estate</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

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

  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item active">

        <a class="nav-link" href="/">Buy <span class="sr-only">(current)</span></a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="/">Sell</a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="/add-property">List Property (free)</a>

      </li>

    </ul>

    <form class="form-inline my-2 my-lg-0">

      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

    </form>

  </div>

</nav>

# Understand Routing - Part 2

So the problem is when we click the nav link the page is complete reloading so it is dangerous when the data is passed through component it make complete reset in order to prevent this problem we remove href in nav bar and used routerlink.

**Navbar.component.html**

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

  <a class="navbar-brand" routerLink="/">Real Estate</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

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

  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item">

        <a class="nav-link" [routerLinkActiveOptions]="{exact:true}" routerLinkActive="active" routerLink="/">Buy <span class="sr-only">(current)</span></a>

      </li>

      <li class="nav-item">

        <a class="nav-link" routerLinkActive="active" routerLink="/rent-property">Rent</a>

      </li>

      <li class="nav-item">

        <a class="nav-link" routerLinkActive="active" routerLink="/add-property">List Property (free)</a>

      </li>

    </ul>

    <form class="form-inline my-2 my-lg-0">

      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

    </form>

  </div>

</nav>

Now add next component so they should display the detail of property when we click the property list in detail button

So add in property folder

**Property-detail.component**

Now when we click on one property card detail button it passed through the property-detail component to do that change in **property card component.html**:

  <div class="card">

    <div class="card-img-wrapper">

<img class="card-img-top" src="assets/images/house.jpg">

<ul class="list-inline text-center member-icons animate">

  <li class="list-inline-item">

    <button class="btn btn-primary" routerLink="/property-detail/{{propertynames.Id}}">

      <i class="far fa-edit"></i> <!--font awesome icon used-->

    </button>

  </li>

    <li class="list-inline-item">

<button class="btn btn-primary">

  <i class="far fa-address-card"></i><!--font awesome icon used-->

    </button>

  </li>

</ul>

</div>

<div class="card-body p-2">

<h3>{{propertynames.Name}}</h3>

<h3 >{{propertynames.Type}}</h3>

<h3>{{propertynames.Price}}</h3>

</div>

</div>

So every property has there own unique property and we want that property id should be passed in the property detail url

<http://localhost:4200/property-detail/2> like this we need a way to pass the id so go to app.module.ts provide id url in property-detail routes

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

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

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

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

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

import { PropertyListComponent } from './property/property-list/property-list.component';

import { PropertyCardComponent } from './property/property-card/property-card.component';

import { NavBarComponent } from './nav-bar/nav-bar.component';

import { HousingService } from './services/housing.service';

import { AddPropertyComponent } from './property/add-property/add-property.component';

import { PropertyDetailComponent } from './property/property-detail/property-detail.component';

const appRoutes : Routes = [

  {path:'', component:PropertyListComponent},

{path: 'add-property', component:AddPropertyComponent},

{path: 'rent-property', component:PropertyListComponent},

{path: 'property-detail/:id', component:PropertyDetailComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    PropertyListComponent,

    PropertyCardComponent,

    NavBarComponent,

    AddPropertyComponent,

    PropertyDetailComponent,

  ],

  imports: [

    BrowserModule,

    HttpClientModule,

    RouterModule.forRoot(appRoutes)

  ],

  providers: [HousingService],

  bootstrap: [AppComponent]

})

export class AppModule { }

also change in property-card.component.html using string interpolation

<li class="list-inline-item">

    <button class="btn btn-primary" routerLink="/property-detail/{{propertynames.Id}}">

      <i class="far fa-edit"></i> <!--font awesome icon used-->

    </button>

  </li>

To fetch the data in property-detail component we injected activated routes

**Property-detail.component.ts**

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

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

@Component({

  selector: 'app-property-detail',

  templateUrl: './property-detail.component.html',

  styleUrls: ['./property-detail.component.css']

})

export class PropertyDetailComponent implements OnInit {

  public propertyId: number;

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void {

    this.propertyId = this.route.snapshot.params['id']//id is same as when you

give in app.module.ts diff will give error

  }

}

**Property-detail.component.html**

<p>property-detail works!</p>

<p>Selected Property id is {{propertyId}}</p>

Next we programmatically navigate to set the clicking of the button

**Add-property.component.html**

<p>add-property works!</p>

<button (click)="onBack()" class="btn btn-primary">Back</button>

**Add-property.component.ts**

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

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

@Component({

  selector: 'app-add-property',

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

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

})

export class AddPropertyComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit(): void {

  }

  onBack(){

    this.router.navigate(['/'])

  }

}

# Understand Routing - Part 3

# There is a scenario when we need to navigate on the same component and we just want to change the parameter on the same route suppose we need a button in property-detail component to support the next record that button should increment property id parameter.

# Update some component here:

# Property-detail.component.html

<p>property-detail works!</p>

<p>Selected Property id is {{propertyId}}

  <br/> <br/>

<button (click)="onSelectNext()" class="btn btn-primary">Select next page</button>

<br/> <br/>

<button routerLink="/property-detail/1" class="btn btn-primary">1</button>

<button routerLink="/property-detail/2" class="btn btn-primary">2</button>

<button routerLink="/property-detail/3" class="btn btn-primary">3</button>

</p>

# 

# Property-detail.component.ts

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

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

@Component({

  selector: 'app-property-detail',

  templateUrl: './property-detail.component.html',

  styleUrls: ['./property-detail.component.css']

})

export class PropertyDetailComponent implements OnInit {

  public propertyId: number;

  constructor(private route: ActivatedRoute, private router: Router) { }

  ngOnInit(): void {

    this.propertyId = +this.route.snapshot.params['id'];

    this.route.params.subscribe(

      (params)=>{

        this.propertyId = +params['id'];

      }

    );

//this logic is used to update parameter value with in same component coz router not update value In same component that’s why we used this

  }

  onSelectNext(){

      this.propertyId +=1;

      this.router.navigate(['property-detail', this.propertyId]);

  }

}

Next scenario when user enter wrong url path it show page not found to do this we add a new component page not found and update app.module.ts

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

Next scenario we categoriezed these properties into two different segment one those are available for buying and other are those are available for rent.

Lets add a new flag into a data file remember “1” will represent a buying and “2” will represent a rent.

Go to data folder update

**Properties.json**

[

  {

  "Id":1,

  "SellRent":1,

  "Name": "Zaryab House",

  "Type":"House",

  "Price": 1200000,

  "Image":"pic"

},

{

  "Id":2,

  "SellRent":1,

  "Name": "Shahroz House",

  "Type":"Plot",

  "Price": 2000000,

  "Image": "house"

},

{

  "Id":3,

  "SellRent":1,

  "Name": "Ahsan House",

  "Type":"Flat",

  "Price": 1300000,

  "Image": "pic4"

},

{

  "Id":4,

  "SellRent":1,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000,

  "Image": "pic4"

},

{

  "Id":5,

  "SellRent":2,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000,

  "Image": "pic2"

},

{

  "Id":6,

  "SellRent":2,

  "Name": "Minaj House",

  "Type":"bugalow",

  "Price": 2400000,

  "Image": "pic6"

}

]

Also add some image in asset folder naming pic1 pic2 .. etc and add in property.json

Also update interface of IProperty

export interface IProperty{

  Id: number;

  SellRent: number;

  Name: string;

  Type: string;

  Price: number;

  Image?:string

}

Now update **property-card.component.html**

  <div class="card">

    <div class="card-img-wrapper">

<img class="card-img-top" src="assets/images/{{propertynames.Image}}.jpg">

<ul class="list-inline text-center member-icons animate">

  <li class="list-inline-item">

    <button class="btn btn-primary" routerLink="/property-detail/{{propertynames.Id}}">

      <i class="far fa-edit"></i> <!--font awesome icon used-->

    </button>

  </li>

    <li class="list-inline-item">

<button class="btn btn-primary">

  <i class="far fa-address-card"></i><!--font awesome icon used-->

    </button>

  </li>

</ul>

</div>

<div class="card-body p-2">

<h3>{{propertynames.Name}}</h3>

<h3 >{{propertynames.Type}}</h3>

<h3>{{propertynames.Price}}</h3>

</div>

</div>

Update **property-list.component.ts**

For categories rent and buy category in property list

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

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

import { HousingService } from 'src/app/services/housing.service';

import { IProperty } from '../IProperty.interface';

@Component({

  selector: 'app-property-list',

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

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

})

export class PropertyListComponent implements OnInit {

 // Properties: Array<any> ;

  Properties: Array<IProperty> ;

  constructor(private route: ActivatedRoute,  private housingService:HousingService ) { }

  SellRent = 1;

  ngOnInit(): void {

if(this.route.snapshot.url.toString()){

  this.SellRent = 2;

  //Means we are on rent-property URL else we are on base url

}

    this.housingService.getAllProperties(this.SellRent).subscribe(

      data =>{

             this.Properties=data;

               console.log(data);

             }, error =>{

               console.log(error);

               //This error condition is used in when api server is down and sometimes database is down so thats why use the conditions

             }

    );

}

}

Update **housing.service.ts**

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

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

import {map} from 'rxjs/operators';

import { IProperty } from '../property/IProperty.interface';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class HousingService {

  constructor(private http: HttpClient) { }

  getAllProperties(SellRent: number):Observable<IProperty[]>{

return this.http.get('data/properties.json').pipe(

map(data=>{

  const propertiesarray: Array<IProperty> = [];

  for(const id in data) {

    if(data.hasOwnProperty(id) && data[id].SellRent === SellRent){

      propertiesarray.push(data[id]);

    }

  }

  return propertiesarray

})

);

  }

}