Angular HTTP CURD Operations Project Steps

we will learn how to quickly integrate HttpClient service in Angular 8 application to consume RESTful API JSON data from the server.

HttpClient enables the communication between client and server through many available methods like get(), post(), put(), delete() etc. We will also add Error handling techniques using RxJS operators at one place in our service.

For demonstrating real-world scenarios, here we will create a Student management application with the following operations:

- Create a new Student record using the post() method.
- List/Get Students data to show in a table using get() method.
- Update any Student data by clicking on Edit put() method.
- Delete any Student by clicking Delete on table row using delete() method.
- HttpClient was introduced after Angular v4.3 so if you have later version you can simply continue

Step 1: First, we will create a new Angular project using the latest version of Angular CLI. If you don't have it just install by running following NPM command☹if already angular installed ignore this steps

> npm install @angular/cli –g

Step 2: Create a Angular Project Run following CLI command to create a new Angular project

>> ng new angular-httpclient-application

? Would you like to add Angular routing? Yes

? Which stylesheet format would you like to use? CSS

Now enter the project

> cd angular-httpclient-application

Step 3: To show controls to create new records, list of Students and update existing Student information, we will create new components in our project.

Run the following commands to generate new components to create, update and list students data.

Angular HTTP CURD Operations Project Steps

> ng generate component student-create --skipTests=true

> ng generate component student-edit --skipTests=true

> ng generate component student-list --skipTests=true

Above ng generate command will automatically inject the components in App's main module. We also added FormsModule to use [(ngModel)] in our components.

Step 4: (**Implement HttpClient in Project):**To make remote server calls and consume JSON responses from RESTful API, we will setup HttpClient which is responsible for making a communication channel between application at client and server-side API.

Step 5: **(Inject HttpClient Module**)Now open app's main module app.module.ts file to import HttpClientModule from '@angular/common/http' then add in imports array of @NgModule decorator as shown below:

```
//app.module.ts

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

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

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

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

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

@NgModule({
 declarations: [
  AppComponent
 ],
 imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule
 ],
```

Angular HTTP CURD Operations Project Steps

```
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 6: (Update Routing Module) Also, update app-routing.module.ts file to add the route paths for above new added components.

```
//app-routing.module.ts
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StudentCreateComponent } from './student-create/student-create.component';
import { StudentEditComponent } from './student-edit/student-edit.component';
import { StudentListComponent } from './student-list/student-list.component';
const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: 'create' },
  { path: 'create', component: StudentCreateComponent },
  { path: 'edit/:id', component: StudentEditComponent},
  { path: 'list', component: StudentListComponent }
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular HTTP CURD Operations Project Steps

Step 7: (Create Dummy Database using JSON Server)To test our HTTP calls, we will create a mock server on over project root folder with a data.json file acting as an API response from the server.

JSON-server NPM package creates a local server very easily for development purposes.

Run following NPM command to install JSON-server package:

> npm install -g json-serverCopy

Now place data.json file in folder API/data.json with following data:

```
{
 "students": [
 {
  "id": 1,
  "name": "Chandra",
  "age": "36",
  "address": "West Venkatapuram,Secundarabad - 50010"
 },{
  "id": 2,
  "name": "Charan",
  "age": "8",
  "address": "West Venkatapuram,Secundarabad - 50010"
 },{
  "id": 3,
  "name": "Madhavi",
  "age": "28",
  "address": "West Venkatapuram,Secundarabad - 50010"
```

Angular HTTP CURD Operations Project Steps


```
 },{
   "id": 4,
   "name": "Anjaneyulu",
   "age": "70",
   "address": "Chirala,Prakasam District,AP-523135"
 },{
   "id": 5,
   "name": "Suguna Kumari",
   "age": "65",
   "address": "Chirala,Prakasam District,AP-523135"   }
 ] }
```


You can run server response by running following NPM command:

> json-server --watch API/data.jsonCopy

It will return a smiley face \{^_^}/ hi! with the server data path: http://localhost:3000/students

Step 8: Create an Interface class for Students data by running following command defining the type of values for student item.

> ng generate class models/StudentCopy

then replace the following content in the newly created file "~/models/student.ts"

```
export class Student {
  id: number;
  name: string;
  age: string;
  address: string;
}
```

Angular HTTP CURD Operations Project Steps

Step 9 : Create Angular Service to Communicate Server using HttpClient methods

For making HTTP calls we will now create a new service which will keep server communication and error handling login separate from rest of the application. This service will import HttpClient services and method using which we will do CRUD operation on data.

Now create a new service file named api.service.ts under services folder by running following ng generate command:

> ng generate service services/api --skipTests=true

--skipTests=true option will not generate spec test files.

Now we will add methods in our api.service.ts file for communicating API server through different HttpClient methods. Here we also added RxJS Observable and Operators to handle errors. handleError() method is handling any error reported by HTTP calls.

For getting JSON response from the server we need to set the 'Content-Type' of every with 'application/json'

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

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

import { Student } from '../model/student';

import { Observable, throwError } from 'rxjs';

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

@Injectable({

  providedIn: 'root'

})

export class ApiService {

  // API path

  base_path = 'http://localhost:3000/students';
```

Angular HTTP CURD Operations Project Steps

```
constructor(private http: HttpClient) { }


// Http Options
httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
}


// Handle API errors
handleError(error: HttpErrorResponse) {
  if (error.error instanceof ErrorEvent) {
    // A client-side or network error occurred. Handle it accordingly.
    console.error('An error occurred:', error.error.message);
  } else {
    // The backend returned an unsuccessful response code.
    // The response body may contain clues as to what went wrong,
    console.error(
      `Backend returned code ${error.status}, ` +
      `body was: ${error.error}`);
  }
  // return an observable with a user-facing error message
  return throwError(
    'Something bad happened; please try again later.');
```

Angular HTTP CURD Operations Project Steps

```
};

// Create a new item

createItem(item): Observable<Student> {

  return this.http

    .post<Student>(this.base_path, JSON.stringify(item), this.httpOptions)

    .pipe(

      retry(2),

      catchError(this.handleError)

    )

}

// Get single student data by ID

getItem(id): Observable<Student> {

  return this.http

    .get<Student>(this.base_path + '/' + id)

    .pipe(

      retry(2),

      catchError(this.handleError)

    )

}

// Get students data

getList(): Observable<Student> {

  return this.http

    .get<Student>(this.base_path)

    .pipe(
```

Angular HTTP CURD Operations Project Steps

```
      retry(2),
      catchError(this.handleError)
    )
  }
  // Update item by id
  updateItem(id, item): Observable<Student> {
   return this.http
    .put<Student>(this.base_path + '/' + id, JSON.stringify(item), this.httpOptions)
    .pipe(
      retry(2),
      catchError(this.handleError)
    )
  }
  // Delete item by id
  deleteItem(id) {
   return this.http
    .delete<Student>(this.base_path + '/' + id, this.httpOptions)
    .pipe(
      retry(2),
      catchError(this.handleError)
    )
  }

}
```

Angular HTTP CURD Operations Project Steps

Step 10: (Update Component Classes and Templates) To style our components, we will add bootstrap.css file in the <head> section of our index.html file

```
<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>AngularHttpclientDemo</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://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

</head>

<body>

  <app-root></app-root>

</body>

</html>
```

Step 11: **(Creat Student Component)** In Create component template we will add a form to take Name, Age, and Address values from the user which will get submitted using submitForm() method. Update student-create.component.html file with below code:

```
<div class="container">

   <h3>Create Student Record</h3>

      <div class="form-group">

         <label for="name">Name</label>
```

Angular HTTP CURD Operations Project Steps

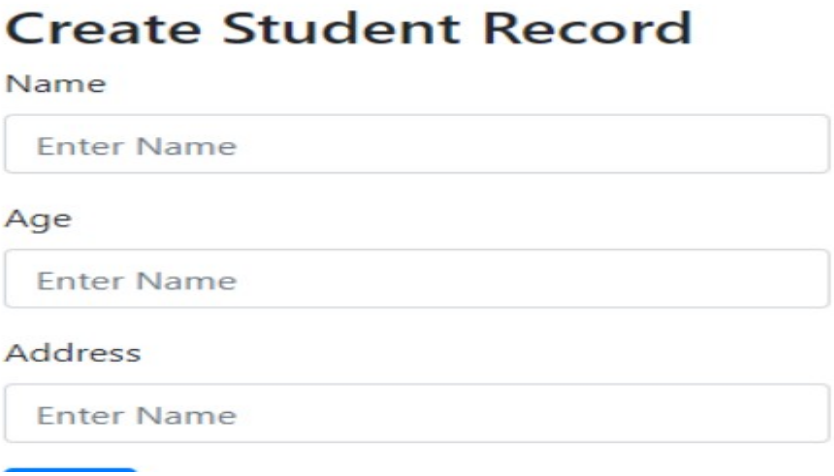
```
        <input type="text" class="form-control" id="name" [(ngModel)]="data.name"
placeholder="Enter Name">

      </div>

      <div class="form-group">

        <label for="age">Age</label>

        <input type="text" class="form-control" id="age" [(ngModel)]="data.age"
placeholder="Enter Name">

      </div>

      <div class="form-group">

        <label for="address">Address</label>

        <input type="text" class="form-control" id="address" [(ngModel)]="data.address"
placeholder="Enter Name">

      </div>

      <button type="submit" class="btn btn-primary" (click)="submitForm()">Add</button>

</div>
```

Step 12: (In the student-create.component.ts file), we will have the submitForm() method to call API service method createItem() to return an Observable. After successfully submitting value we will navigate to list page using Router service.

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

import { Student } from '../models/student';

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

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

@Component({

  selector: 'app-student-create',

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

  styleUrls: ['./student-create.component.css']
```

Angular HTTP CURD Operations Project Steps

```
})
export class StudentCreateComponent implements OnInit {
  data: Student
  constructor(
    public apiService: ApiService,
    public router: Router
  ) {
    this.data = new Student();
  }
  ngOnInit() {
  }
  submitForm() {
    this.apiService.createItem(this.data).subscribe((response) => {
      this.router.navigate(['list']);
    });
  }
}
```

Angular HTTP CURD Operations Project Steps

Step 13: (List Students Component) In list component, we will list all Students in our data.json file in a bootstrap table with rows iterating using *ngFor directive.

The table will also have an Action colum to show Edit and Delete buttons. In Edit action we are simply redirecting to Edit component with item id which we will update in later. The delete button will call delete method in our API service.

Replace the student-list.column.html file with the following code:

```
<div class="container">
  <h3>List All Students</h3>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">Id</th>
        <th scope="col">Name</th>
        <th scope="col">Age</th>
        <th scope="col">Address</th>
        <th scope="col">Actions</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let item of studentsData">
        <td>{{item.id}}</td>
        <td>{{item.name}}</td>
        <td>{{item.age}}</td>
```

Angular HTTP CURD Operations Project Steps

```
      <td>{{item.address}}</td>

      <td style="display: flex">

        <button type="button" class="btn btn-warning btn-sm" [routerLink]="[
'/edit/${item.id}']">Edit</button>

        <button type="button" class="btn btn-danger btn-sm ml-1"
(click)="delete(item)">Delete</button>

      </td>

    </tr>

  </tbody>

 </table>

 <button type="button" class="btn btn-success" [routerLink]="[ '/create']">Add
Student</button>

</div>
```

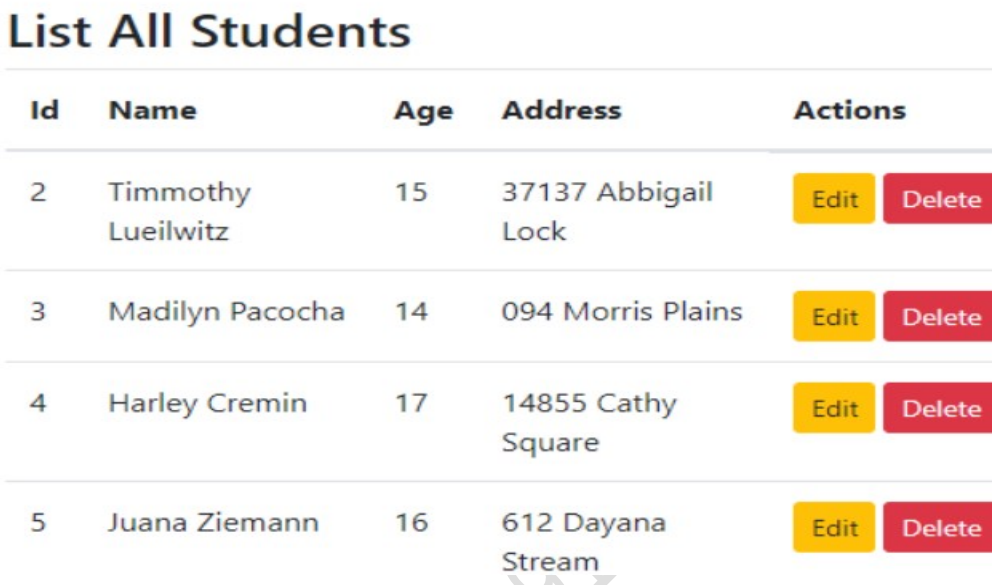
Step 14: In the student-list.component.ts file, we will get the list of all students by calling getList() and also add delete() to call deleteItem() method in API service.

```
//student-list.component.ts

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

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


@Component({

 selector: 'app-student-list',

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

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

})

export class StudentListComponent implements OnInit {
```

Angular HTTP CURD Operations Project Steps

```
studentsData: any;

constructor(

  public apiService: ApiService

) {

  this.studentsData = [];

}

ngOnInit() {

  this.getAllStudents();

}

getAllStudents() {

  //Get saved list of students

  this.apiService.getList().subscribe(response => {

    console.log(response);

    this.studentsData = response;

  })

}

delete(item) {

  //Delete item in Student data

  this.apiService.deleteItem(item.id).subscribe(Response => {

    //Update list after delete is successful

    this.getAllStudents();

  });

 }

}
```

Angular HTTP CURD Operations Project Steps



Step 15: (Update/ Edit Student Item) In Edit component we will get the id of item using ActivatedRoute service then get its details. After that, we will show Form field controls to edit them, after that user can update the value to call the updateItem method in API service.

In the student-edit.component.html file replace following HTML content:

```
<div class="container">

    <h3>Edit Student Record</h3>

        <div class="form-group">

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

         <input type="text" class="form-control" id="name" [(ngModel)]="data.name"
placeholder="Enter Name">

    </div>

    <div class="form-group">

         <label for="age">Age</label>

         <input type="text" class="form-control" id="age" [(ngModel)]="data.age"
placeholder="Enter Name">

    </div>
```

Angular HTTP CURD Operations Project Steps

```
        <div class="form-group">
          <label for="address">Address</label>
          <input type="text" class="form-control" id="address" [(ngModel)]="data.address"
placeholder="Enter Name">
        </div>
        <button type="submit" class="btn btn-success" (click)="update()">Update</button>
        <button type="submit" class="btn btn-warning ml-1" [routerLink]="[
'/list']">Cancel</button>
    </div>
```

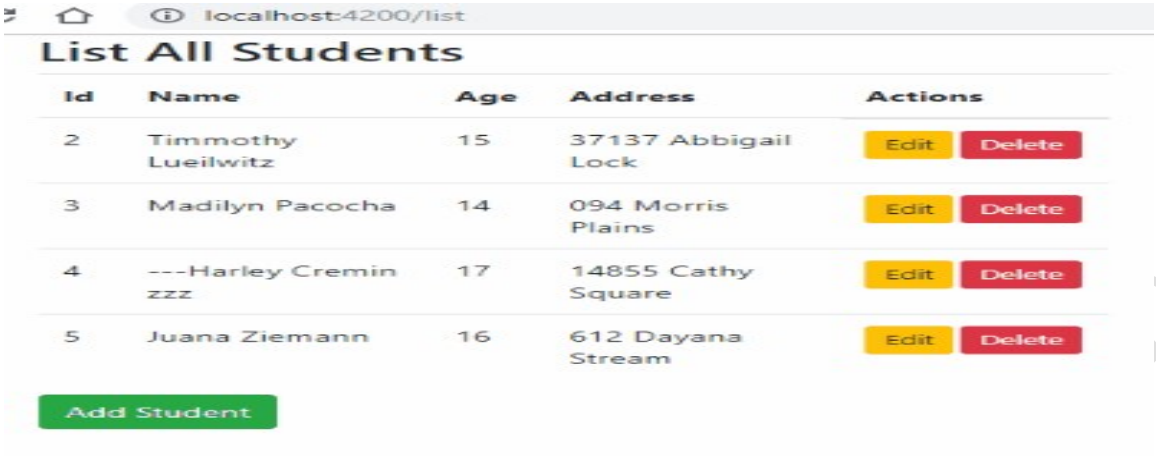
Step 16: Now in student-edit.component.ts file replace following class component code:

```
//student-edit.component.ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Student } from '../models/student';
import { ApiService } from '../services/api.service';
@Component({
  selector: 'app-student-edit',
  templateUrl: './student-edit.component.html',
  styleUrls: ['./student-edit.component.css']
})
export class StudentEditComponent implements OnInit {
  id: number;
  data: Student;
  constructor(
    public activatedRoute: ActivatedRoute,
```

Angular HTTP CURD Operations Project Steps

```
  public router: Router,
  public apiService: ApiService
 ) {
  this.data = new Student();
 }
 ngOnInit() {
  this.id = this.activatedRoute.snapshot.params["id"];
  //get item details using id
  this.apiService.getItem(this.id).subscribe(response => {
    console.log(response);
    this.data = response;
  })
 }
 update() {
  //Update item by taking id and updated data object
  this.apiService.updateItem(this.id, this.data).subscribe(response => {
    this.router.navigate(['list']);
  })
 }
}
```

Angular HTTP CURD Operations Project Steps



Step 17: That's it now you are ready to run your app by hitting following command

> ng serve –open

Step 18: Don't forget to run the json-server to up API server by running following command in a separate console.

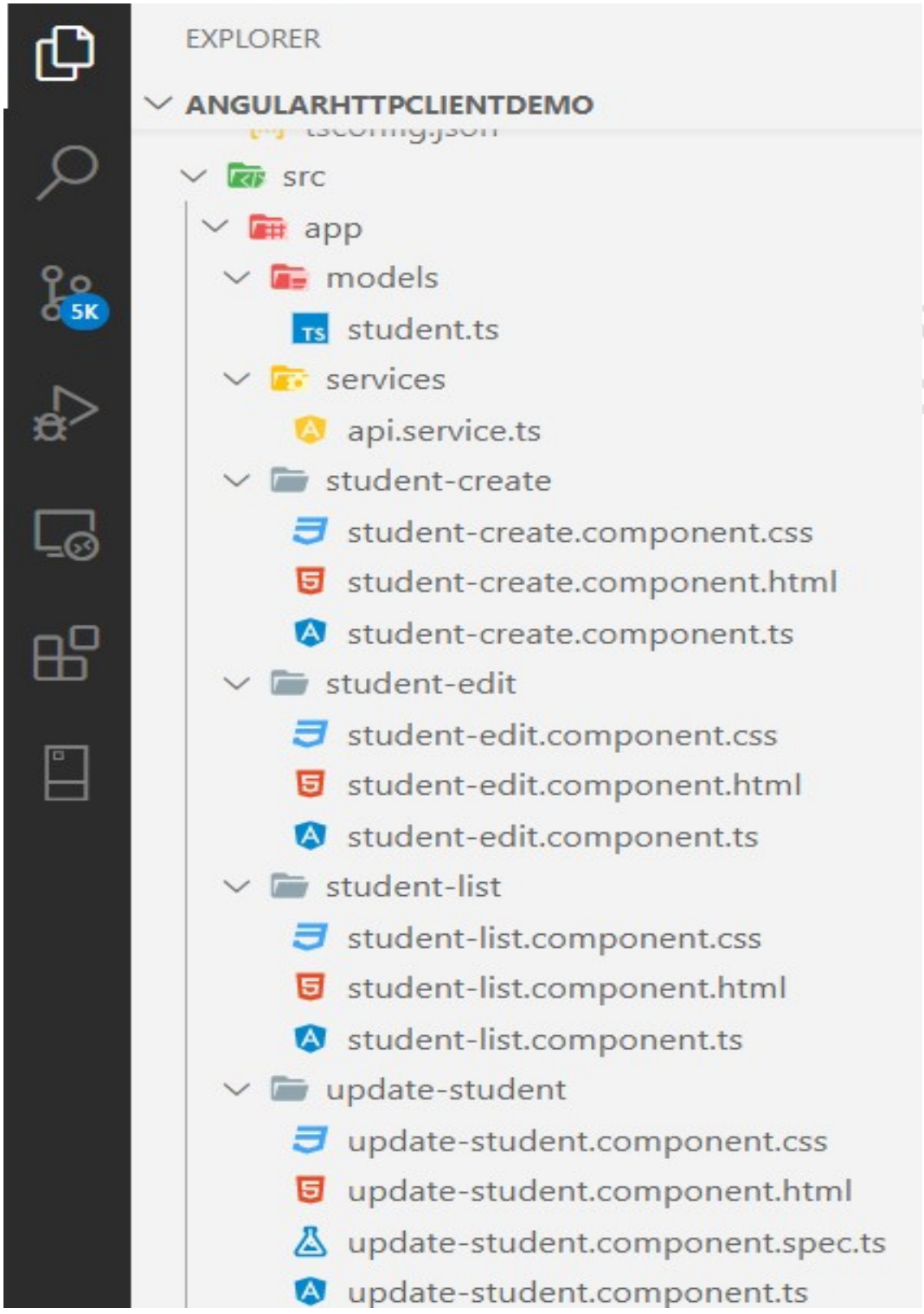
> json-server --watch API/data.json

Note: Angular Application to run use the following command before start the server

npm install --save-dev @angular-devkit/build-angular

check Project Structure

Angular HTTP CURD Operations Project Steps



Thank you. Refer any One Interested for Angular Training