**To post information to the JSON file and to fetch and display that information using HTTP.**

**STEPS:**

### 1) Create an Ionic Application

Run the following command in terminal to create a new Ionic application of type angular with a blank template:

### 2) Add New Pages in Application

> ionic generate page student-create

>ionic generate page student-edit

>ionic generate page student-list

>ionic generate page student-detail

### 3) Update Routing Module

### //app-routing.module.ts

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

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

const routes: Routes = [

  { path: '', redirectTo: 'student-list', pathMatch: 'full' },

  {

    path: 'student-create',

    loadChildren: () => import('./student-create/student-create.module').then( m => m.StudentCreatePageModule)

  },

  {

    path: 'student-edit',

    loadChildren: () => import('./student-edit/student-edit.module').then( m => m.StudentEditPageModule)

  },

  {

    path: 'student-list',

    loadChildren: () => import('./student-list/student-list.module').then( m => m.StudentListPageModule)

  },

  {

    path: 'student-detail',

    loadChildren: () => import('./student-detail/student-detail.module').then( m => m.StudentDetailPageModule)

  },

];

@NgModule({

  imports: [

    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })

  ],

  exports: [RouterModule]

})

export class AppRoutingModule { }

### 4) Add HttpClientModule in Application

//**app.module.ts**

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

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

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

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

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

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

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

@NgModule({

  declarations: [AppComponent],

  entryComponents: [],

  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,HttpClientModule],

  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],

  bootstrap: [AppComponent],

})

export class AppModule {}

### 5) HttpConfig Service with Methods for CRUD Operations

### > ng generate service services/api

### //Services->api.service.ts

//api.service.ts

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

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

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

  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.');

  };

  // 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(

        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)

      )

  }

}

### >ng generate class models/Student

### Models->student.ts

export class Student {

    id: number;

    name: string;

    age: string;

    address: string;

 }

### 6) Mock Server for Application

### > npm install -g json-server

### //**API->data.json**

{

    "students": [

    {

      "id": 1,

      "name": "Enola Rowe",

      "class": "tony@mcu.com",

      "address": "131 Oswaldo Street"

    },{

      "id": 2,

      "name": "Timmothy Lueilwitz",

      "age": "15",

      "address": "37137 Abbigail Lock"

    },{

      "id": 3,

      "name": "Madilyn Pacocha",

      "age": "14",

      "address": "094 Morris Plains"

    },{

      "id": 4,

      "name": "Harley Cremin",

      "age": "17",

      "address": "14855 Cathy Square"

    },{

      "id": 5,

      "name": "Juana Ziemann",

      "age": "16",

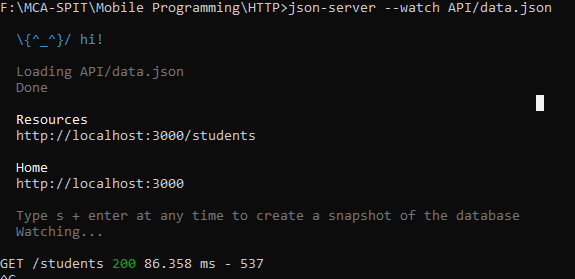
      "address": "612 Dayana Stream"

    }

    ]

  }

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



### 

### 7) Create New Students

### //****student-create.page.html****

<!-- student-create.html -->

<ion-header>

  <ion-toolbar color="tertiary">

    <ion-title>Create Student</ion-title>

  </ion-toolbar>

</ion-header>

<ion-content class="ion-padding">

  <ion-item>

    <ion-label>Name</ion-label>

    <ion-input [(ngModel)]="data.name" placeholder="Enter Name"></ion-input>

  </ion-item>

  <ion-item>

    <ion-label>Age</ion-label>

    <ion-input [(ngModel)]="data.age" placeholder="Enter Age"></ion-input>

  </ion-item>

  <ion-item>

    <ion-button (click)="submitForm()">Add

    </ion-button>

  </ion-item>

</ion-content>

### //****student-create.page.ts****

//student-create.page.ts

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.page.html',

  styleUrls: ['./student-create.page.scss'],

})

export class StudentCreatePage 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(['student-list']);

    });

  }

}

### 8) List Students Component

### //****student-list.page.html****

<!-- student-list.page.html -->

<ion-header>

  <ion-toolbar color="tertiary">

    <ion-title>List All Students</ion-title>

  </ion-toolbar>

</ion-header>

<ion-content class="ion-padding">

  <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>

        <td>{{ item.address }}</td>

        <td style="display: flex">

          <ion-button color="warning" size="small" routerLink='/student-edit/{{item.id}}'>

            <ion-icon name="create"></ion-icon>

          </ion-button>

          <ion-button color="danger" size="small" (click)="delete(item)">

            <ion-icon name="trash"></ion-icon>

          </ion-button>

        </td>

      </tr>

    </tbody>

  </table>

  <ion-button [routerLink]="['/student-create']">

    Add Student

  </ion-button>

</ion-content>

### //  ****student-list.page.ts****

//student-list.page.ts

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

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

@Component({

  selector: 'app-student-list',

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

  styleUrls: ['./student-list.page.scss'],

})

export class StudentListPage implements OnInit {

  studentsData: any;

  constructor(

    public apiService: ApiService

  ) {

    this.studentsData = [];

  }

  ngOnInit() {

    // this.getAllStudents();

  }

  ionViewWillEnter() {

    // Used ionViewWillEnter as ngOnInit is not

    // called due to view persistence in Ionic

    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();

    });

  }

}

### 9) Update Student Item

### //****student-edit.page.html****

<!-- student-edit.page.html -->

<ion-header>

  <ion-toolbar color="tertiary">

    <ion-title>Edit Student Record</ion-title>

  </ion-toolbar>

</ion-header>

<ion-content class="ion-padding">

  <ion-item>

    <ion-label>Name</ion-label>

    <ion-input [(ngModel)]="data.name" placeholder="Enter Name"></ion-input>

  </ion-item>

  <ion-item>

    <ion-label>Age</ion-label>

    <ion-input [(ngModel)]="data.age" placeholder="Enter Age"></ion-input>

  </ion-item>

  <ion-item>

    <ion-label>Address</ion-label>

    <ion-input [(ngModel)]="data.address" placeholder="Enter Address"></ion-input>

  </ion-item>

  <ion-button (click)="update()" color="success" size="small">

    Update

  </ion-button>

  <ion-button [routerLink]="[ '/student-list']" color="danger" size="small">

    Cancel

  </ion-button>

</ion-content>

### //****student-edit.page.ts****

//student-edit.page

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.page.html',

  styleUrls: ['./student-edit.page.scss'],

})

export class StudentEditPage implements OnInit {

  id: number;

  data: Student;

  constructor(

    public activatedRoute: ActivatedRoute,

    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(['student-list']);

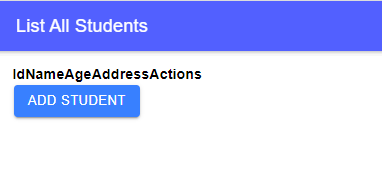
    })

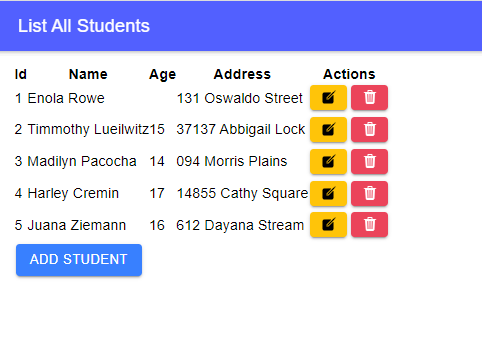
  }

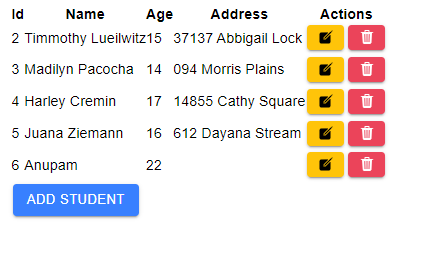
}

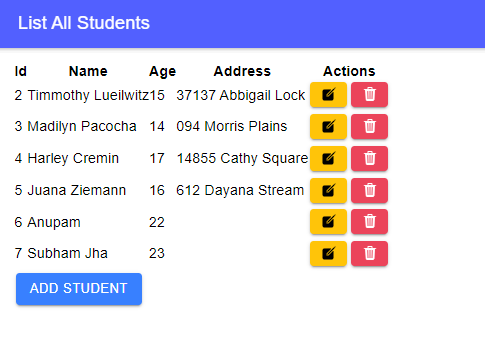
### 

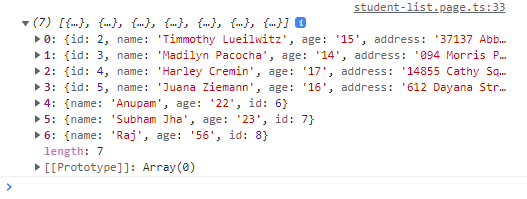
**Output:**











**Q.1 What is the use of HTTP?**

->

HTTP was invented alongside HTML to create the first interactive, text-based web browser: the original World Wide Web. Today, the protocol remains one of the primary means of using the Internet.

**Q.2 What is the difference between HTTPClient and HTTPClientModule**?

->

They both support HTTP calls but HTTP is the older API and will eventually be deprecated. The new HttpClient service is included in the HttpClientModule that used to initiate HTTP request and responses in angular apps. The HttpClientModule is a replacement of HttpModule.

**Q.3 What is GET and POST method used for? what is the structure for the same?**

->

* GET method supports only string data types while POST method supports different data types, such as string, numeric, binary, etc.
* GET request is often cacheable while POST request is hardly cacheable.
* GET performs are better compared to POST.

**Q.4 List all the methods used with HTTP.**

->

The primary or most commonly-used HTTP methods are **POST, GET, PUT, PATCH, and DELETE**.

**Q.5 What is the use of subscribe and map in HTTP?**

->

subscribe() returns a Subscription that can not be subscribed to, but it can be used **to cancel the subscription**. map() returns an Observable which can be subscribed to. Think map as a middleware which transforms the response.