Angular 18 application that persists student records in localStorage using an interface

**Step 1: Install Angular CLI (if not already installed)**

Make sure you have Angular CLI installed. If not, install it globally using npm:

npm install -g @angular/cli

**Step 2: Create a New Angular Application**

Generate a new Angular application:

ng new student-app

cd student-app

**Step 3: Create a Student Interface**

Create an interface to define the structure of a student record. Run the following command to generate the interface:

ng generate interface models/student

This will create a file src/app/models/student.ts. Update it as follows:

export interface Student {

id: number;

name: string;

age: number;

grade: string;

}

**Step 4: Create a Service to Manage Student Records**

Generate a service to handle CRUD operations with localStorage:

ng generate service services/student

Update the generated service (src/app/services/student.service.ts) as follows:

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

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

@Injectable({

providedIn: 'root',

})

export class StudentService {

private readonly storageKey = 'students';

constructor() {}

// Get all students from localStorage

getStudents(): Student[] {

const studentsJson = localStorage.getItem(this.storageKey);

return studentsJson ? JSON.parse(studentsJson) : [];

}

// Add a new student

addStudent(student: Student): void {

const students = this.getStudents();

students.push(student);

localStorage.setItem(this.storageKey, JSON.stringify(students));

}

// Update a student

updateStudent(updatedStudent: Student): void {

const students = this.getStudents();

const index = students.findIndex((s) => s.id === updatedStudent.id);

if (index !== -1) {

students[index] = updatedStudent;

localStorage.setItem(this.storageKey, JSON.stringify(students));

}

}

// Delete a student by ID

deleteStudent(id: number): void {

const students = this.getStudents().filter((s) => s.id !== id);

localStorage.setItem(this.storageKey, JSON.stringify(students));

}

// Generate a unique ID for a new student

generateId(): number {

const students = this.getStudents();

return students.length > 0 ? Math.max(...students.map((s) => s.id)) + 1 : 1;

}

}

**Step 5: Create a Component to Display and Manage Students**

Generate a component to display and manage student records:

ng generate component components/student-list

Update the generated component (src/app/components/student-list/student-list.component.ts) as follows:

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

import { StudentService } from '../../services/student.service';

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

@Component({

selector: 'app-student-list',

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

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

})

export class StudentListComponent implements OnInit {

students: Student[] = [];

newStudent: Student = { id: 0, name: '', age: 0, grade: '' };

isEditing = false;

constructor(private studentService: StudentService) {}

ngOnInit(): void {

this.loadStudents();

}

loadStudents(): void {

this.students = this.studentService.getStudents();

}

addStudent(): void {

this.newStudent.id = this.studentService.generateId();

this.studentService.addStudent(this.newStudent);

this.loadStudents();

this.newStudent = { id: 0, name: '', age: 0, grade: '' }; // Reset form

}

editStudent(student: Student): void {

this.newStudent = { ...student };

this.isEditing = true;

}

updateStudent(): void {

this.studentService.updateStudent(this.newStudent);

this.loadStudents();

this.isEditing = false;

this.newStudent = { id: 0, name: '', age: 0, grade: '' }; // Reset form

}

deleteStudent(id: number): void {

this.studentService.deleteStudent(id);

this.loadStudents();

}

}

Update the template (src/app/components/student-list/student-list.component.html) as follows:

<h2>Student List</h2>

<form (ngSubmit)="isEditing ? updateStudent() : addStudent()">

<input [(ngModel)]="newStudent.name" name="name" placeholder="Name" required />

<input [(ngModel)]="newStudent.age" name="age" type="number" placeholder="Age" required />

<input [(ngModel)]="newStudent.grade" name="grade" placeholder="Grade" required />

<button type="submit">{{ isEditing ? 'Update' : 'Add' }} Student</button>

</form>

<ul>

<li \*ngFor="let student of students">

{{ student.name }} (Age: {{ student.age }}, Grade: {{ student.grade }})

<button (click)="editStudent(student)">Edit</button>

<button (click)="deleteStudent(student.id)">Delete</button>

</li>

</ul>

Run HTML

**Step 6: Update the App Module**

Ensure the FormsModule is imported in src/app/app.module.ts to enable two-way data binding:

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

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

import { FormsModule } from '@angular/forms'; // Import FormsModule

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

import { StudentListComponent } from './components/student-list/student-list.component';

@NgModule({

declarations: [AppComponent, StudentListComponent],

imports: [BrowserModule, FormsModule], // Add FormsModule here

providers: [],

bootstrap: [AppComponent],

})

export class AppModule {}

**Step 7: Update the App Component**

Replace the content of src/app/app.component.html with:

<app-student-list></app-student-list>

**Step 8: Run the Application**

Start the application:

ng serve

Open your browser and navigate to http://localhost:4200. You should see the student management interface where you can add, edit, and delete student records. All data will be persisted in localStorage.