Health Care Domain Mini Project

Project Name: HealthCare Management System

Objective: Create a simple HealthCare Management System using Angular that allows users to manage patient information. This mini-project will focus on utilizing Angular Directives, Debugging and Error Handling, Life Cycle Hooks, and Pipes.

1. Project Setup

- **Angular CLI Installation:**
  ```sh
  npm install -g @angular/cli@15
  ng new healthcare-management-system
  cd healthcare-management-system
  ng serve
  ```

2. Directory Structure

~~~
```
src/
├── app/
│   ├── components/
│   │   ├── patient-list/
│   │   │   ├── patient-list.component.html
│   │   │   ├── patient-list.component.ts
│   │   │   ├── patient-list.component.css
│   │   ├── patient-detail/
│   │   │   ├── patient-detail.component.html
│   │   │   ├── patient-detail.component.ts
│   │   │   ├── patient-detail.component.css
│   ├── directives/
│   │   ├── highlight.directive.ts
│   ├── services/
│   │   ├── patient.service.ts
│   ├── app.component.html
│   ├── app.component.ts
│   ├── app.module.ts
├── assets/
├── environments/
~~~

~~~

3. Creating Components

- **Patient List Component:**

  ```typescript
  // patient-list.component.ts
  import { Component, OnInit } from '@angular/core';
  import { PatientService } from '../../services/patient.service';

  @Component({
    selector: 'app-patient-list',
    templateUrl: './patient-list.component.html',
    styleUrls: ['./patient-list.component.css']
  })
  export class PatientListComponent implements OnInit {
    patients: any[];
~~~

```
  constructor(private patientService: PatientService) {}

  ngOnInit(): void {
    this.patientService.getPatients().subscribe((data) => {
      this.patients = data;
    });
  }
}
```

```html
<!-- patient-list.component.html -->
<div>
  <h2>Patient List</h2>
  <ul>
    <li *ngFor="let patient of patients" [appHighlight]>{{ patient.name }}</li>
  </ul>
```

~~~~
  </div>
  ```
~~~~

- **Patient Detail Component:**

~~~~
  ```typescript
  // patient-detail.component.ts
  import { Component, Input, OnInit, OnChanges, SimpleChanges } from '@angular/core';

  @Component({
    selector: 'app-patient-detail',
    templateUrl: './patient-detail.component.html',
    styleUrls: ['./patient-detail.component.css']
  })
  export class PatientDetailComponent implements OnInit, OnChanges {
    @Input() patient: any;

    ngOnInit(): void {
~~~~

~~~
    console.log('PatientDetailComponent initialized');
  }

  ngOnChanges(changes: SimpleChanges): void {
    console.log('PatientDetailComponent changes detected', changes);
  }
}
```

```html
<!-- patient-detail.component.html -->
<div *ngIf="patient">
  <h2>{{ patient.name }} Details</h2>
  <div><span>Age: </span>{{ patient.age }}</div>
  <div><span>Condition: </span>{{ patient.condition }}</div>
</div>
```
~~~

4. Creating Directives

- **Highlight Directive:**

  ```typescript
  // highlight.directive.ts
  import { Directive, ElementRef, Renderer2, HostListener } from '@angular/core';

  @Directive({
    selector: '[appHighlight]'
  })
  export class HighlightDirective {
    constructor(private el: ElementRef, private renderer: Renderer2) {}

    @HostListener('mouseenter') onMouseEnter() {
      this.renderer.setStyle(this.el.nativeElement, 'backgroundColor', 'yellow');
    }
  ```

    @HostListener('mouseleave') onMouseLeave() {
      this.renderer.setStyle(this.el.nativeElement, 'backgroundColor', 'white');
    }
  }
  ~~~

5. Debugging and Error Handling

- **Service with Error Handling:**

  ```typescript
  // patient.service.ts
  import { Injectable } from '@angular/core';
  import { HttpClient, HttpErrorResponse } from '@angular/common/http';
  import { catchError } from 'rxjs/operators';
  import { throwError } from 'rxjs';

  @Injectable({
    providedIn: 'root'
  ~~~

```
})
export class PatientService {
  private apiUrl = 'api/patients'; // URL to web API

  constructor(private http: HttpClient) {}

  getPatients() {
    return this.http.get<any[]>(this.apiUrl).pipe(
      catchError(this.handleError)
    );
  }

  private handleError(error: HttpErrorResponse) {
    let errorMessage = 'Unknown error!';
    if (error.error instanceof ErrorEvent) {
      // Client-side errors
      errorMessage = `Error: ${error.error.message}`;
    } else {
      // Server-side errors
      errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
```

```
      }
      return throwError(errorMessage);
    }
  }
  ```

6. Life Cycle Hooks

- **Implementing Life Cycle Hooks in Components:**

  - Already implemented in `PatientDetailComponent` with `ngOnInit` and `ngOnChanges`.

7. Creating Pipes

- **Custom Pipe for Formatting:**

  ```typescript
  // pipes/age.pipe.ts
  import { Pipe, PipeTransform } from '@angular/core';
  ```

~~~
@Pipe({
  name: 'age'
})
export class AgePipe implements PipeTransform {
  transform(value: number, ...args: unknown[]): string {
    return `${value} years old`;
  }
}
```

```html
<!-- patient-detail.component.html -->
<div *ngIf="patient">
  <h2>{{ patient.name }} Details</h2>
  <div><span>Age: </span>{{ patient.age | age }}</div>
  <div><span>Condition: </span>{{ patient.condition }}</div>
</div>
~~~

~~~

8. App Module

- **Register Components, Directives, and Pipes:**

  ```typescript
  // app.module.ts
  import { BrowserModule } from '@angular/platform-browser';
  import { NgModule } from '@angular/core';
  import { HttpClientModule } from '@angular/common/http';

  import { AppComponent } from './app.component';
  import { PatientListComponent } from './components/patient-list/patient-list.component';
  import { PatientDetailComponent } from './components/patient-detail/patient-detail.component';
  import { HighlightDirective } from './directives/highlight.directive';
~~~

```
import { AgePipe } from './pipes/age.pipe';
import { PatientService } from './services/patient.service';

@NgModule({
  declarations: [
    AppComponent,
    PatientListComponent,
    PatientDetailComponent,
    HighlightDirective,
    AgePipe
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [PatientService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Solution Overview

1. **Directives:** The `HighlightDirective` demonstrates how to create and use custom directives in Angular.

2. **Debugging and Error Handling:** The `PatientService` includes error handling to manage HTTP request errors.

3. **Life Cycle Hooks:** The `PatientDetailComponent` uses `ngOnInit` and `ngOnChanges` to demonstrate Angular life cycle hooks.

4. **Pipes:** The `AgePipe` formats patient age data in a readable format.