Customer Care Application Mini Project

Project Overview

We'll develop a simple Customer Care Application where users can view, add, and manage customer complaints. This project will cover:

1. **Angular Directives**: Custom directives for handling form validation and dynamic styling.
2. **Debugging and Error Handling in Angular**: Implementing error handling in services and components.
3. **Life Cycle Hooks in Angular**: Utilizing different life cycle hooks in components.
4. **Pipes in Angular**: Creating custom pipes for formatting data.

Project Structure

1. **Components**:
   - `CustomerListComponent`: Displays a list of customer complaints.

- `CustomerFormComponent`: A form to add a new complaint.

- `CustomerDetailComponent`: Displays detailed information of a selected complaint.

2. **Services**:

- `CustomerService`: Handles CRUD operations for customer complaints.

3. **Directives**:

- `HighlightDirective`: Highlights complaints based on priority.

- `ValidateDirective`: Custom validation for form fields.

4. **Pipes**:

- `PriorityPipe`: Formats the priority level of complaints.

- `DateFormatPipe`: Formats the date of the complaint.

5. **Error Handling**:

- Global error handling service.
- Component-specific error handling.

6. **Life Cycle Hooks**:
- Demonstrating `ngOnInit`, `ngOnChanges`, `ngOnDestroy`, etc.

Step-by-Step Implementation

1. Setup Angular Project

```bash
ng new customer-care-app
cd customer-care-app
ng generate component customer-list
ng generate component customer-form
ng generate component customer-detail
ng generate service customer
ng generate directive highlight
ng generate directive validate
ng generate pipe priority
```

ng generate pipe dateFormat
```

#### 2. Customer Service (`customer.service.ts`)

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

export interface CustomerComplaint {
  id: number;
  name: string;
  email: string;
  complaint: string;
  date: string;
  priority: string;
}

```
@Injectable({
  providedIn: 'root'
})
export class CustomerService {
  private apiUrl = 'api/complaints';

  constructor(private http: HttpClient) {}

  getComplaints():
Observable<CustomerComplaint[]> {
    return
this.http.get<CustomerComplaint[]>(this.apiUrl).pip
e(
      catchError(this.handleError)
    );
  }

  addComplaint(complaint: CustomerComplaint):
Observable<CustomerComplaint> {
    return
this.http.post<CustomerComplaint>(this.apiUrl,
complaint).pipe(
```

```
      catchError(this.handleError)
    );
  }

  private handleError(error: HttpErrorResponse) {
    console.error('An error occurred:', error.error.message);
    return throwError('Something went wrong; please try again later.');
  }
}
```

3. Customer List Component (`customer-list.component.ts`)

```typescript
import { Component, OnInit } from '@angular/core';
import { CustomerService, CustomerComplaint } from '../customer.service';

@Component({
```

```
  selector: 'app-customer-list',
  templateUrl: './customer-list.component.html',
  styleUrls: ['./customer-list.component.css']
})
export class CustomerListComponent implements OnInit {
  complaints: CustomerComplaint[] = [];

  constructor(private customerService: CustomerService) {}

  ngOnInit(): void {
    this.customerService.getComplaints().subscribe(
      (data) => (this.complaints = data),
      (error) => console.error('Error fetching complaints', error)
    );
  }
}
```

4. Customer Form Component (`customer-form.component.ts`)

```typescript
import { Component } from '@angular/core';
import { CustomerService, CustomerComplaint } from '../customer.service';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';

@Component({
  selector: 'app-customer-form',
  templateUrl: './customer-form.component.html',
  styleUrls: ['./customer-form.component.css']
})
export class CustomerFormComponent {
  complaintForm: FormGroup;

  constructor(private fb: FormBuilder, private customerService: CustomerService) {
    this.complaintForm = this.fb.group({
      name: ['', Validators.required],
```

```
    email: ['', [Validators.required, Validators.email]],
    complaint: ['', Validators.required],
    date: ['', Validators.required],
    priority: ['', Validators.required]
  });
 }

 onSubmit() {
  if (this.complaintForm.valid) {

this.customerService.addComplaint(this.complaintForm.value as CustomerComplaint).subscribe(
      (data) => console.log('Complaint added', data),
      (error) => console.error('Error adding complaint', error)
    );
  }
 }
}
```

5. Customer Detail Component (`customer-detail.component.ts`)

```typescript
import { Component, Input } from '@angular/core';
import { CustomerComplaint } from '../customer.service';

@Component({
  selector: 'app-customer-detail',
  templateUrl: './customer-detail.component.html',
  styleUrls: ['./customer-detail.component.css']
})
export class CustomerDetailComponent {
  @Input() complaint!: CustomerComplaint;
}
```

6. Highlight Directive (`highlight.directive.ts`)

```typescript
```

```
import { Directive, ElementRef, Input, OnChanges } from '@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective implements OnChanges {
  @Input() appHighlight!: string;

  constructor(private el: ElementRef) {}

  ngOnChanges() {
    this.highlight(this.appHighlight);
  }

  private highlight(priority: string) {
    switch (priority) {
      case 'High':
        this.el.nativeElement.style.backgroundColor = 'red';
```

```
      break;
    case 'Medium':
      this.el.nativeElement.style.backgroundColor =
'yellow';
      break;
    case 'Low':
      this.el.nativeElement.style.backgroundColor =
'green';
      break;
    default:
      this.el.nativeElement.style.backgroundColor =
'white';
   }
  }
}
```

7. Validate Directive (`validate.directive.ts`)

```typescript
import { Directive, Input } from '@angular/core';
```

```
import { NG_VALIDATORS, Validator,
AbstractControl, ValidationErrors } from
'@angular/forms';


@Directive({
  selector: '[appValidate]',
  providers: [{ provide: NG_VALIDATORS,
useExisting: ValidateDirective, multi: true }]
})
export class ValidateDirective implements Validator
{
  @Input('appValidate') validationType!: string;


  validate(control: AbstractControl): ValidationErrors
| null {
    if (this.validationType === 'email' &&
control.value) {
      const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-
Z0-9.-]+\.[a-zA-Z]{2,4}$/;
      return emailRegex.test(control.value) ? null : {
invalidEmail: true };
    }
    return null;
```

```
  }
}
```

8. Priority Pipe (`priority.pipe.ts`)

```typescript
import { Pipe, PipeTransform } from
'@angular/core';

@Pipe({
  name: 'priority'
})
export class PriorityPipe implements PipeTransform
{
  transform(value: string): string {
    switch (value) {
      case 'High':
        return '🔴 High';
      case 'Medium':
        return '🟡 Medium';
```

```
      case 'Low':
        return '🟢 Low';
      default:
        return value;
    }
  }
}
```

9. Date Format Pipe (`date-format.pipe.ts`)

```typescript
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'dateFormat'
})
export class DateFormatPipe implements PipeTransform {
  transform(value: string): string {
```

```
    const date = new Date(value);
    return date.toLocaleDateString();
  }
}
```

10. Error Handling Service (`error-handler.service.ts`)

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

@Injectable({
  providedIn: 'root'
})
export class GlobalErrorHandlerService implements ErrorHandler {
  handleError(error: any): void {
    console.error('An unexpected error occurred:', error);
```

```
    alert('An unexpected error occurred. Please try again later.');
  }
}
```

11. App Module (`app.module.ts`)

```typescript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, ErrorHandler } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { CustomerListComponent } from './customer-list/customer-list.component';
```

```
import { CustomerFormComponent } from
'./customer-form/customer-form.component';
import { CustomerDetailComponent } from
'./customer-detail/customer-detail.component';
import { HighlightDirective } from
'./highlight.directive';
import { ValidateDirective } from
'./validate.directive';
import { PriorityPipe } from './priority.pipe';
import { DateFormatPipe } from './date-format.pipe';
import { GlobalErrorHandlerService } from './error-
handler.service';

@NgModule({
  declarations: [
    AppComponent,
    CustomerListComponent,
    CustomerFormComponent,
    CustomerDetailComponent,
    HighlightDirective,
    ValidateDirective,
    PriorityPipe,
```

```
    DateFormatPipe
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [{ provide: ErrorHandler, useClass:
GlobalErrorHandlerService }],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Running the Application

1. Start the Angular application using `ng serve`.
2. Navigate to `http://localhost:4200` to view the application.

3. You can add, view, and manage customer complaints using the different components and see the directives, pipes, and error handling in action.

This mini-project provides a comprehensive overview of how to use Angular Directives, Debugging and Error Handling, Life Cycle Hooks, and Pipes in a real-world application.