**Crate and Configure Angular App**

**To create new app with Angular CLI**

ng new cheap-clothes --style=scss

**To compile and run the project in the default browser at** [**http://localhost:4200**](http://localhost:4200)

ng serve –-open

**To configure the project to use Angular Material, Angular Animations and HammerJS**

npm install --save @angular/cdk

npm install --save @angular/animations

npm install --save hammerjs

npm install --save @angular/material

**To use Material Design Icons include into the <head> of index.html the following**

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

**To use Flex Layout**

npm install --save @angular/flex-layout

Import into app.module.ts the following

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

import { MatToolbarModule } from '@angular/material/toolbar';

import { FlexLayoutModule } from '@angular/flex-layout';

import { MatListModule } from '@angular/material/list';

import 'hammerjs';

imports: [

    BrowserModule,

    BrowserAnimationsModule,

    MatToolbarModule,

    FlexLayoutModule,

MatListModule

  ],

To add Material Toolbar replace app.component.html with the following

<mat-toolbar color="primary"> <span>Cheap Clothes</span> </mat-toolbar>

To add built-in Material Theme into styles.scss

/\* You can add global styles to this file, and also import other style files \*/

@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';

// some basic resets

body {

  padding: 0;

  margin: 0;

  font-family: Roboto, sans-serif;

}

.container {

  margin: 20px;

  display:flex;

}

**To create home component**

ng g component home

To add the new component to the template file add selector into app.component.html

<app-home></app-home>

**To use use mat-list**

Import the following in app.module.ts

import { MatListModule } from '@angular/material/list';

Include into home.component.html

<div class="container"

     fxLayout="column"

     fxLayoutGap="10px">

  <mat-list fxFlex>

    <mat-list-item \*ngFor="let c of clothes">

      <img matListAvatar src={{c.image}} alt={{c.name}}>

      <h1 matLine> {{c.name}} </h1>

      <p matLine>

        <span> {{c.description}} </span>

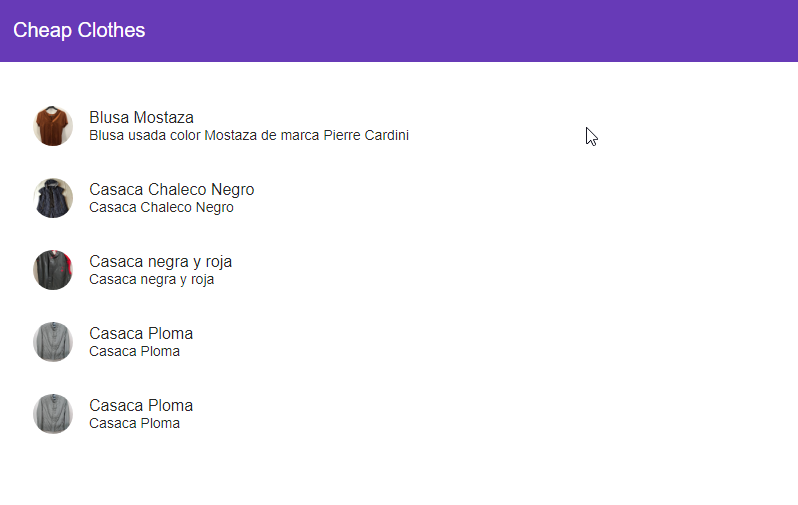
      </p>

    </mat-list-item>

  </mat-list>

</div>

The page shows like next image



**To use mat-grid-list**

Import the following in app.module.ts

import { MatGridListModule } from '@angular/material/grid-list';

import { MatCardModule } from '@angular/material/card';

import { MatButtonModule } from '@angular/material/button';

Include into home.component.html

<div fxFlex>

    <mat-grid-list cols="3" rowHeight="200px">

      <mat-grid-tile \*ngFor="let c of clothes">

        <img height="200px" src={{c.image}} alt={{c.name}}>

        <mat-grid-tile-footer>

          <h1>{{ c.name | uppercase }}</h1>

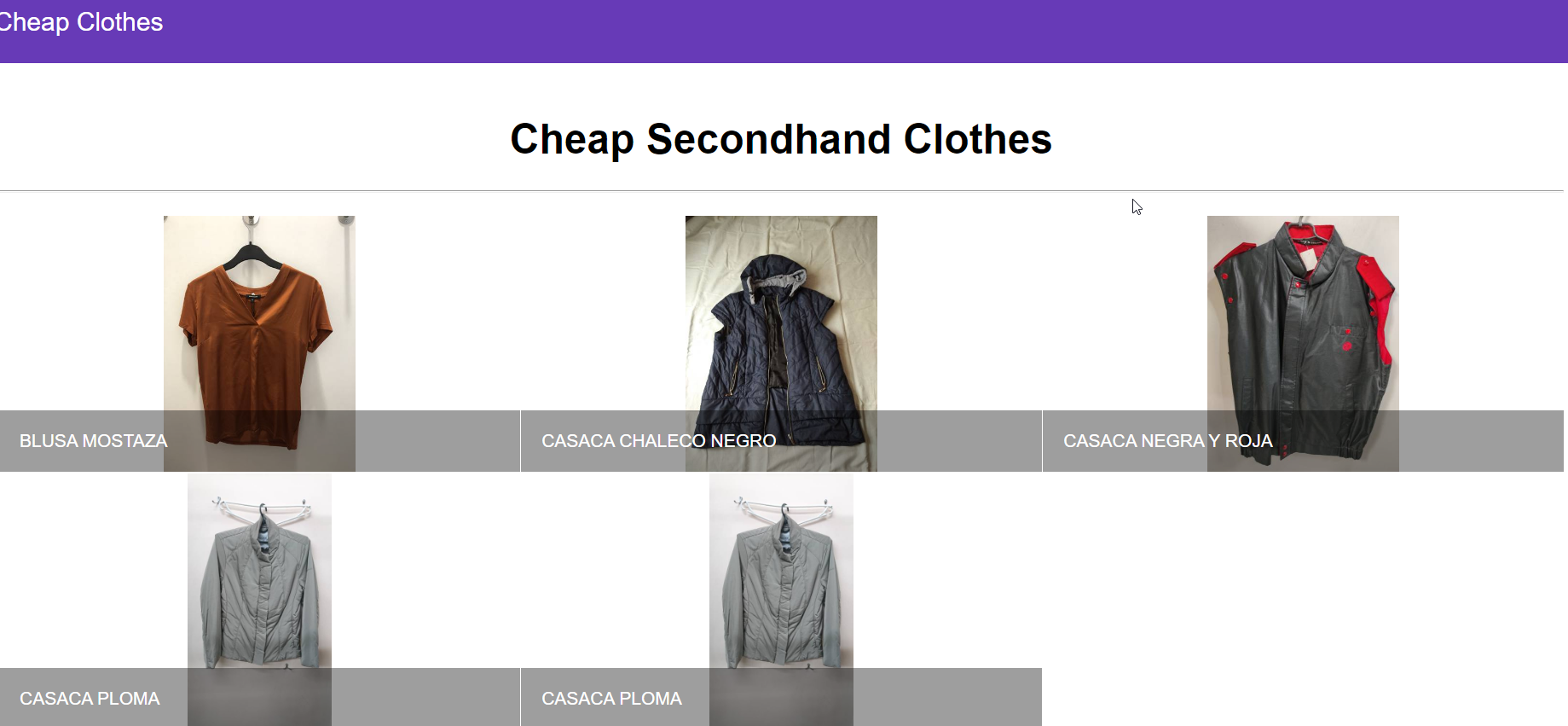
        </mat-grid-tile-footer>

      </mat-grid-tile>

    </mat-grid-list>

  </div>

The page shows like next image



**To add a service**

Create a folder services and type

ng g service services/clothing

Add service to app.module.ts

import { ClothingService } from './services/clothing.service';

providers: [

    ClothingService

  ],

To use service in the [home.component.html](http://home.component.html), inject the service into constructor

constructor(private clothingService: ClothingService) { }

  ngOnInit(): void {

    this.clothes = this.clothingService.getClothes();

  }

**To use Font Awesome Icons**

Install via NPM

npm install –-save font-awesome

Add a new file named \_variables.scss in the src folder and add the following

$fa-font-path : '~node\_modules/font-awesome/fonts';

Update the styles.scss

@import 'variables';

@import '../node\_modules/font-awesome/scss/font-awesome';

**To add and use a Router Module**

Type the following in the src

ng g module app-routing

Create a new file named routes.ts in the app-routing folder and update it as follows

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

import { HomeComponent } from '../home/home.component';

import { AboutComponent } from '../about/about.component';

import { ContactComponent } from '../contact/contact.component';

import { ShopComponent } from '../shop/shop.component';

export const routes: Routes = [

  { path: 'home',  component: HomeComponent },

  { path: 'about',     component: AboutComponent },

  { path: 'shop',     component: ShopComponent },

  { path: 'contact',     component: ContactComponent },

  { path: '', redirectTo: '/home', pathMatch: 'full' }

];

Update the app-routing-module.ts file to maje use of routes

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

import { CommonModule } from '@angular/common';

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

import { routes } from './routes';

@NgModule({

  declarations: [],

  imports: [

    CommonModule,

    RouterModule.forRoot(routes)

  ],

  exports: [ RouterModule ]

})

export class AppRoutingModule { }

Update the app.component.html as follows

<app-header></app-header>

<router-outlet></router-outlet>

<app-footer></app-footer>

Update the app.module.ts as follows

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

imports: [

-----------

AppRoutingModule

-------

Update with routerLink the toolbar in header.component.html

<mat-toolbar color="primary">

  <span><img src="/assets/images/logo.png" height=30 width=41></span>

  <a mat-button routerLink="/home"><span class="fa fa-home fa-lg"></span> Home</a>

  <a mat-button routerLink="/about"><span class="fa fa-info fa-lg"></span> About Us</a>

  <a mat-button routerLink="/shop"><span class="fa fa-list fa-lg"></span> Shop</a>

  <a mat-button routerLink="/contact"><span class="fa fa-address-card fa-lg"></span> Contact Us</a>

</mat-toolbar>

**To highlighting the current component link in Toolbar**

Update each of the links with routerLinkActive and the scss class

<a mat-button routerLink="/home" routerLinkActive="active"><span class="fa fa-home fa-lg"></span> Home</a>

Update the scss class of the header component

$background-moredark: #4527A0;

-------

.active {

    background: $background-moredark;

}

**To configure SPA and pass params another page**

Update routes.ts

import { ClothingdetailComponent } from '../clothingdetail/clothingdetail.component';

{ path: 'clothingdetail/:id', component: ClothingdetailComponent },

Update mat-grid-list in [home.component.html](http://home.component.html)

<mat-grid-tile \*ngFor="let c of clothes" [routerLink]="['/clothingdetail', c.id]">

Update the clothindetail.component.ts

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

import { Location } from '@angular/common';

import { ClothingService } from '../services/clothing.service';

-----

constructor(private clothingService: ClothingService,

    private route: ActivatedRoute,

    private location: Location) { }

ngOnInit(): void {

    const id = this.route.snapshot.params['id'];

    this.clothing = this.clothingService.getClothing(id);

  }

  goBack(): void {

    this.location.back();

  }

**To add a dialog**

Update the toolbar into header.component.html

<span class="flex-spacer"></span>

  <a mat-button (click)="openLoginForm()"><span class="fa fa-sign-in fa-lg"></span> Login</a>

Generate a component to show the dialog

ng g component login

Update the template file login.component.html as follows:

<mat-toolbar color="primary">

    Login

  <span class="flex-spacer"></span>

  <button mat-button mat-dialog-close>&times;</button>

</mat-toolbar>

To make this component be opened from another component, declare this as an EntryComponent in the AppModule, by adding the following to the NgModule decorator. In addition, import the MatDialogModule

import { MatDialogModule } from '@angular/material/dialog';

MatDialogModule

entryComponents: [

    LoginComponent

  ],

To trigger the Dialog view of the Login component, open header.component.ts and update

import { MatDialog } from '@angular/material/dialog';

import { LoginComponent } from '../login/login.component';

constructor(public dialog: MatDialog) { }

openLoginForm() {

    this.dialog.open(LoginComponent, { width: '500px', height: '450px'});

  }

**To use template-driven forms**

Update app.module.ts

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatInputModule } from '@angular/material/input';

import { MatCheckboxModule } from '@angular/material/checkbox';

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

-----

MatFormFieldModule,

    MatInputModule,

    MatCheckboxModule,

    FormsModule

Update login.component.html with Angular Material support

<p>{{ user | json }}</p>

<form novalidate (ngSubmit)="onSubmit()">

  <mat-dialog-content>

    <p>

      <mat-form-field>

        <input matInput placeholder="Username" type="text" [(ngModel)]="user.username" name="username">

      </mat-form-field>

      <mat-form-field>

        <input matInput placeholder="Password" type="password" [(ngModel)]="user.password" name="password">

      </mat-form-field>

      <mat-checkbox [(ngModel)]="user.remember" name="remember">Remember Me</mat-checkbox>

    </p>

  </mat-dialog-content>

  <mat-dialog-actions>

    <span class="flex-spacer"></span>

    <button mat-button mat-dialog-close>Cancel</button>

    <button type="submit" mat-button class="background-primary text-floral-white" >Login</button>

  </mat-dialog-actions>

</form>

Update login.component.html

user = {username: '', password: '', remember: false};

  constructor(public dialogRef: MatDialogRef<LoginComponent>) { }

  ngOnInit(): void {

  }

  onSubmit(): void {

    console.log('User: ', this.user);

    this.dialogRef.close();

  }

**To use simple validation in template-driven forms**

Update login.component.html

<form novalidate #loginForm="ngForm" (ngSubmit)="onSubmit()">

-------

<input matInput placeholder="Username" type="text" [(ngModel)]="user.username" name="username" #username="ngModel" required>

<mat-error \*ngIf="username.errors?.required">Username is required</mat error>

------<input matInput placeholder="Password" type="password" [(ngModel)]="user.password" name="password" #password="ngModel" required>

<mat-error \*ngIf="password.errors?.required">Password is required</mat-error>

------

<button type="submit" mat-button class="background-primary text-floral-white" [disabled]="loginForm.form.invalid">Login</button>

**To add and use Angular Reactive Forms with Validation**

Update app.module.ts

import { MatSelectModule } from '@angular/material/select';

import { MatSlideToggleModule } from '@angular/material/slide-toggle';

import { ReactiveFormsModule } from '@angular/forms';

---------

MatSelectModule,

    MatSlideToggleModule,

    ReactiveFormsModule

Update contact.component.ts

import { FormBuilder, FormGroup } from '@angular/forms';

-------

feedbackForm: FormGroup;

  feedback: Feedback;

--------

@ViewChild('fform') feedbackFormDirective;

constructor(private fb: FormBuilder) {

    this.createForm();

  }

createForm() {

    this.feedbackForm = this.fb.group({

      firstname: ['', Validators.required ],

      lastname: ['', Validators.required ],

      telnum: ['', Validators.required ],

      email: ['', Validators.required ],

      agree: false,

      contacttype: 'None',

      message: ''

    });

  }

onSubmit() {

    this.feedback = this.feedbackForm.value;

    console.log(this.feedback);

    this.feedbackForm.reset({

      firstname: '',

      lastname: '',

      telnum: '',

      email: '',

      agree: false,

      contacttype: 'None',

      message: ''

    });

    this.feedbackFormDirective.resetForm();

  }

Update contact.component.html

<form novalidate [formGroup]="feedbackForm" #fform="ngForm" (ngSubmit)="onSubmit()">

      <p>

        <mat-form-field class="half-width">

          <input matInput formControlName="firstname" placeholder="First Name" type="text" required>

          <mat-error \*ngIf="feedbackForm.get('firstname').hasError('required') && feedbackForm.get('firstname').touched">First name is required</mat-error>

        </mat-form-field>

        <mat-form-field class="half-width">

          <input matInput formControlName="lastname" placeholder="Last Name" type="text" required>

          <mat-error \*ngIf="feedbackForm.get('lastname').hasError('required') && feedbackForm.get('lastname').touched">Last name is required</mat-error>

        </mat-form-field>

      </p>

      <p>

        <mat-form-field class="half-width">

          <input matInput formControlName="telnum" placeholder="Tel. Number" type="tel" required>

          <mat-error \*ngIf="feedbackForm.get('telnum').hasError('required') && feedbackForm.get('telnum').touched">Tel. number is required</mat-error>

        </mat-form-field>

        <mat-form-field class="half-width">

          <input matInput formControlName="email" placeholder="Email" type="email" required>

          <mat-error \*ngIf="feedbackForm.get('email').hasError('required') && feedbackForm.get('email').touched">Email ID is required</mat-error>

        </mat-form-field>

      </p>

      <table class="form-size">

        <td>

          <mat-slide-toggle formControlName="agree">May we contact you?</mat-slide-toggle>

        </td>

        <td>

          <mat-select placeholder="How?" formControlName="contacttype">

            <mat-option \*ngFor="let ctype of contactType" [value]="ctype">

              {{ ctype }}

            </mat-option>

          </mat-select>

        </td>

        </table>

      <p>

        <mat-form-field class="full-width">

          <textarea matInput formControlName="message" placeholder="Your Feedback" rows=12></textarea>

        </mat-form-field>

      </p>

      <button type="submit" mat-button class="background-primary text-floral-white">Submit</button>

    </form>

Update styles of contact component

.full-width {

    width: 95%

}

.half-width {

    width: 45%

}

.form-size {

    width: 75%

}

**To add and use Angular Promise with Delay**

Update clothing.service.ts

getClothes(): Promise<Clothing[]> {

    return new Promise(resolve => {

      setTimeout(() => resolve(CLOTHES), 2000);

    });

  }

  getClothing(id: string): Promise<Clothing> {

    return new Promise(resolve => {

      setTimeout(() => resolve(CLOTHES.filter((c) => (c.id === id))[0]), 2000);

    });

  }

  getFeaturedClothes(): Promise<Clothing[]> {

    return new Promise(resolve => {

      setTimeout(() => resolve(CLOTHES.filter(c => c.featured)), 2000);

    });

  }

Update app.module.ts with progress spinner

import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';

MatProgressSpinnerModule

Update [home.component.html](http://home.component.html) to show progress spinner

<div fxFlex [hidden]="clothes">

    <mat-spinner></mat-spinner> <h4>Loading... Please Wait!</h4>

  </div>

**To add and use Angular RxJS with Observable and Delay**

Update clothing.service.ts

import { Observable, of } from 'rxjs';

import { delay } from 'rxjs/operators';

getClothes(): Observable<Clothing[]> {

    return of(CLOTHES).pipe(delay(2000));

  }

  getClothing(id: string): Observable<Clothing> {

    return of(CLOTHES.filter((c) => (c.id === id))[0]).pipe(delay(2000));

  }

  getFeaturedClothes(): Observable<Clothing[]> {

    return of(CLOTHES.filter(c => c.featured)).pipe(delay(2000));

  }

getClothingIds(): Observable<string[] | any> {

    return of(CLOTHES.map(clothing => clothing.id));

  }

Update the subscribers

this.clothingService.getClothing(id).subscribe(c => this.clothing = c);

-------------

this.clothingService.getFeaturedClothes().subscribe(clothes => this.clothes = clothes); // Using RxJs Observables

Update clothingdetail.component.ts

import { switchMap } from 'rxjs/operators';

------

clothingIds: string[];

  prev: string;

  next: string;

-------

ngOnInit(): void {

    this.clothingService.getClothingIds().subscribe(ids => this.clothingIds = ids);

    this.route.params.pipe(switchMap((params: Params) => this.clothingService.getClothing(params['id'])))

      .subscribe(c => {

        this.clothing = c;

        this.setPrevNext(c.id);

      });

  }

  setPrevNext(clothingId: string) {

    const index = this.clothingIds.indexOf(clothingId);

    this.prev = this.clothingIds[(this.clothingIds.length + index - 1) % this.clothingIds.length];

    this.next = this.clothingIds[(this.clothingIds.length + index + 1) % this.clothingIds.length];

  }

Update clothingdetail.component.html

<button mat-button [routerLink]="['/clothingdetail', prev]"><span class="fa fa-chevron-left fa-lg"></span></button>

--------

<button mat-button [routerLink]="['/clothingdetail', next]"><span class="fa fa-chevron-right fa-lg"></span></button>

**To add and use validation to React Forms**

Update contact.component.ts

formErrors = {

    'firstname': '',

    'lastname': '',

    'telnum': '',

    'email': ''

  };

  validationMessages = {

    'firstname': {

      'required':      'First Name is required.',

      'minlength':     'First Name must be at least 2 characters long.',

      'maxlength':     'FirstName cannot be more than 25 characters long.'

    },

    'lastname': {

      'required':      'Last Name is required.',

      'minlength':     'Last Name must be at least 2 characters long.',

      'maxlength':     'Last Name cannot be more than 25 characters long.'

    },

    'telnum': {

      'required':      'Tel. number is required.',

      'pattern':       'Tel. number must contain only numbers.'

    },

    'email': {

      'required':      'Email is required.',

      'email':         'Email not in valid format.'

    },

  };

createForm() {

    this.feedbackForm = this.fb.group({

      firstname: ['', [Validators.required, Validators.minLength(2), Validators.maxLength(25)] ],

      lastname: ['', [Validators.required, Validators.minLength(2), Validators.maxLength(25)] ],

      telnum: ['', [Validators.required, Validators.pattern] ],

      email: ['', [Validators.required, Validators.email] ],

      agree: false,

      contacttype: 'None',

      message: ''

    });

    this.feedbackForm.valueChanges

      .subscribe(data => this.onValueChanged(data));

    this.onValueChanged(); // (re)set form validation messages

  }

  onValueChanged(data?: any) {

    if(!this.feedbackForm) {

      return;

    }

    const form = this.feedbackForm;

    for (const field in this.formErrors){

      if (this.formErrors.hasOwnProperty(field)) {

        // clear previous error message (if any)

        this.formErrors[field] = '';

        const control = form.get(field);

        if (control && control.dirty && !control.valid) {

          const messages = this.validationMessages[field];

          for (const key in control.errors) {

            if (control.errors.hasOwnProperty(key)) {

              this.formErrors[field] += messages[key] + ' ';

            }

          }

        }

      }

    }

  }

Update contact.component.html

<p>

        <mat-form-field class="half-width">

          <input matInput formControlName="firstname" placeholder="First Name" type="text" required>

          <mat-error \*ngIf="formErrors.firstname">{{formErrors.firstname}}</mat-error>

        </mat-form-field>

        <mat-form-field class="half-width">

          <input matInput formControlName="lastname" placeholder="Last Name" type="text" required>

          <mat-error \*ngIf="formErrors.lastname">{{formErrors.lastname}}</mat-error>

        </mat-form-field>

      </p>

      <p>

        <mat-form-field class="half-width">

          <input matInput formControlName="telnum" placeholder="Tel. Number" type="tel" pattern="[0-9]\*" required>

          <mat-error \*ngIf="formErrors.telnum">{{formErrors.telnum}}</mat-error>

        </mat-form-field>

        <mat-form-field class="half-width">

          <input matInput formControlName="email" placeholder="Email" type="email" email required>

          <mat-error \*ngIf="formErrors.email">{{formErrors.email}}</mat-error>

        </mat-form-field>

      </p>

**To add and use HTTP Client**

Update app.module.ts

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

-------

HttpClientModule

--------

providers: [

{ provide: 'BaseURL', useValue: baseURL }

Create a new file named baseurl.ts in the shared folder with the following

export const baseURL = 'http://localhost:3000/';

Update clothing.service.ts

// USING ANGULAR RxJS OBSERVABLES AND HTTP CLIENT

  getClothes(): Observable<Clothing[]> {

    return this.http.get<Clothing[]>(baseURL + 'clothes');

  }

  getClothing(id: string): Observable<Clothing> {

    return this.http.get<Clothing>(baseURL + 'clothes/' + id);

  }

  getFeaturedClothes(): Observable<Clothing[]> {

    return this.http.get<Clothing[]>(baseURL + 'clothes?featured=true');

  }

  getClothingIds(): Observable<string[] | any> {

    return this.getClothes().pipe(map(clothes => clothes.map(clothing => clothing.id)));

  }

Update constructor in the components

constructor(private clothingService: ClothingService,

    @Inject('BaseURL') public BaseURL) { }

Update html templates of the components

 <img height="200px" src="{{ BaseURL + c.image}}" alt={{c.name}}>

**To add Error Handler in HTTP Client**

Create a new service ProcessHTTPMsg

ng g service services/ProcessHTTPMsg

Update app.module.ts

import { ProcessHTTPMsgService } from './services/process-httpmsg.service';

providers: [

    ProcessHTTPMsgService

Update process-httpmsg.service.ts

public handleError(error: HttpErrorResponse | any) {

    let errMsg: string;

    if (error.error instanceof ErrorEvent) {

      errMsg = error.error.message;

    } else {

      errMsg = `${error.status} - ${error.statusText || ''} ${error.error}`;

    }

    return throwError(errMsg);

  }

Update clothing.service.ts

// USING ANGULAR RxJS OBSERVABLES AND HTTP CLIENT

  getClothes(): Observable<Clothing[]> {

    return this.http.get<Clothing[]>(baseURL + 'clothes')

      .pipe(catchError(this.processHTTPMsgService.handleError));

  }

  getClothing(id: string): Observable<Clothing> {

    return this.http.get<Clothing>(baseURL + 'clothes/' + id)

      .pipe(catchError(this.processHTTPMsgService.handleError));

  }

  getFeaturedClothes(): Observable<Clothing[]> {

    return this.http.get<Clothing[]>(baseURL + 'clothes?featured=true')

      .pipe(catchError(this.processHTTPMsgService.handleError));

  }

  getClothingIds(): Observable<string[] | any> {

    return this.getClothes().pipe(map(clothes => clothes.map(clothing => clothing.id)))

      .pipe(catchError(error => error));

  }

Update home component

errmess: string;

this.clothingService.getClothingIds().subscribe(

      ids => this.clothingIds = ids,

      errmess => this.errmess = errmess

    );

Update [home.component.html](http://home.component.html)

<div fxFlex [hidden]="clothes || errmess">

    <mat-spinner></mat-spinner> <h4>Loading... Please Wait!</h4>

  </div>

  <div fxFlex \*ngIf="errmess">

    <h2>Error</h2>

    <h4>{{errmess}}</h4>

  </div>

**To add new directive**

Create directive

ng g directive directives/highlight

Update highlight.directive.ts

import { Directive, ElementRef, Renderer2, HostListener  } from '@angular/core';

-------

constructor(private el: ElementRef,

    private renderer: Renderer2) { }

  @HostListener('mouseenter')

  onMouseEnter() {

    this.renderer.addClass(this.el.nativeElement, 'highlight');

  }

  @HostListener('mouseleave')

  onMouseLeave() {

    this.renderer.removeClass(this.el.nativeElement, 'highlight');

  }

Update styles.scss

.highlight {

  background-color: $background-pale;

  border: 1px solid $primary-color-dark;

  z-index: 1;

  transform: scale(1.01);

}

Update [home.component.html](http://home.component.html)

<mat-grid-tile \*ngFor="let c of clothes" [routerLink]="['/clothingdetail', c.id]" appHighlight>

**To add and use Animations**

Create a new file app.animation.ts in a new folder animations and update this file

import { trigger, state, style, animate, transition } from '@angular/animations';

export function visibility() {

    return trigger('visibility', [

        state('shown', style({

            transform: 'scale(1.0)',

            opacity: 1

        })),

        state('hidden', style({

            transform: 'scale(0.5)',

            opacity: 0

        })),

        transition('\* => \*', animate('0.5s ease-in-out'))

    ]);

}

export function flyInOut() {

    return trigger('flyInOut', [

        state('\*', style({

            opacity: 1,

            transform: 'translateX(0)'

        })),

        transition(':enter', [

            style({

                transform: 'translateX(-100%)',

                opacity: 0

            }),

            animate('500ms ease-in')

        ]),

        transition(':leave', [

            animate('500ms ease-out', style( {

                transform: 'translateX(100%)',

                opacity: 0

            }))

        ])

    ]);

}

export function expand() {

    return trigger('expand', [

        state('\*', style({

            opacity: 1,

            transform: 'translateX(0)'

        })),

        transition(':enter', [

            style({

                transform: 'translateY(-50%)',

                opacity: 0

            }),

            animate('200ms ease-in', style({

                opacity: 1,

                transform: 'translateX(0)'

            }))

        ])

    ]);

}

Update [home.component.ts](http://home.component.ts)

import { flyInOut, expand } from '../animations/app.animation';

@Component({

-----

  host: {

    '[@flyInOut]': 'true',

    'style': 'display: block;'

  },

  animations: [

    flyInOut(),

    expand()

  ]

})

Update [home.component.html](http://home.component.html)

 <div fxFlex="100" \*ngIf="clothing" [@expand]>