Login,component.ts ➔

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from
'@angular/forms';
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';
import { MatFormFieldModule } from '@angular/material/form-field'
import { MatInputModule } from '@angular/material/input';
import { UserService } from '../../services/user.service';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { Router, RouterLink, RouterOutlet } from '@angular/router';
import { CommonModule } from '@angular/common';
import { Observable } from 'rxjs';
import { LoadingService } from '../../services/loading.service';
@Component({
  selector: 'app-login',
  standalone: true,
  imports: [MatFormFieldModule,
    MatInputModule,
  MatCardModule,
MatButtonModule,
ReactiveFormsModule,
MatProgressSpinnerModule,
RouterOutlet,RouterLink ,
CommonModule],
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'
})
export class LoginComponent implements OnInit{
  loginForm:FormGroup;
  userDtaFromAPI!:any;
  showSpinner!: Observable<boolean>;

  constructor(private fb:FormBuilder,
    private loaderService:LoadingService,
    private userService: UserService,
    private router: Router
  ){
    this.loginForm = this.fb.group({
      email : ['',[Validators.required,Validators.pattern('[a-z0-9]+@[a-
z]+\.[a-z]{2,3}')]],
      password: ['',[Validators.required,Validators.minLength(4)]]
    }
    )
  }
  ngOnInit(): void {
      this.showSpinner = this.loaderService.loading$;
  }

  verifyUser(){
    alert(`data: ${this.loginForm.value.email}
${this.loginForm.value.password}`)
    const userEmail = this.loginForm.value.email;
    const userPass = this.loginForm.value.password
    this.userService.getUser().subscribe(res => {
      this.userDtaFromAPI = res;
      this.userDtaFromAPI.forEach((each: any) => {
        if(each.email == userEmail && each.password == userPass){
          this.router.navigate(['/dash'])
        }
      })
    })
    this.loginForm.reset()
  }


}
```

Login.component.html ➔

```
<h1>Login Form</h1>



<mat-card class="login-card ">
    <mat-card-header>
        <mat-card-subtitle>
            <p>email: alan&#64;gmail.com</p>
            <p>ps: 123456</p>
        </mat-card-subtitle>
    </mat-card-header>
    <mat-card-content>
        <form [formGroup]="loginForm" (ngSubmit)="verifyUser()">
            <mat-form-field class="full-width">
                <mat-label>Email</mat-label>
                <input formControlName="email" type="text" matInput>
                @if(loginForm.get('email')?.hasError('required')){
                    <mat-error>required error</mat-error>
                }
                @if(loginForm.get('email')?.hasError('pattern')){
                    <mat-error>pattern error</mat-error>
                }
            </mat-form-field>
            <mat-form-field class="full-width">
                <mat-label>Password</mat-label>
                <input formControlName="password" type="password" matInput>
                @if(loginForm.get('password')?.hasError('required')){
                    <mat-error>required error</mat-error>
                }
                @if(loginForm.get('password')?.hasError('minlength')){
                    <mat-error>length 4 error</mat-error>
                }
            </mat-form-field>
            <button mat-raised-button type="submit">Submit</button>
        </form>
    </mat-card-content>
</mat-card>
@if(showSpinner | async){
    <div class="spinner">
        <mat-spinner></mat-spinner>
    </div>
}
```

Login.component.css ➔

```
<h1>Login Form</h1>



<mat-card class="login-card ">
    <mat-card-header>
        <mat-card-subtitle>
            <p>email: alan&#64;gmail.com</p>
            <p>ps: 123456</p>
        </mat-card-subtitle>
    </mat-card-header>
    <mat-card-content>
        <form [formGroup]="loginForm" (ngSubmit)="verifyUser()">
            <mat-form-field class="full-width">
                <mat-label>Email</mat-label>
                <input formControlName="email" type="text" matInput>
                @if(loginForm.get('email')?.hasError('required')){
                    <mat-error>required error</mat-error>
                }
                @if(loginForm.get('email')?.hasError('pattern')){
                    <mat-error>pattern error</mat-error>
                }
            </mat-form-field>
            <mat-form-field class="full-width">
                <mat-label>Password</mat-label>
                <input formControlName="password" type="password" matInput>
                @if(loginForm.get('password')?.hasError('required')){
                    <mat-error>required error</mat-error>
                }
                @if(loginForm.get('password')?.hasError('minlength')){
                    <mat-error>length 4 error</mat-error>
                }
            </mat-form-field>
            <button mat-raised-button type="submit">Submit</button>
        </form>
    </mat-card-content>
</mat-card>
@if(showSpinner | async){
    <div class="spinner">
        <mat-spinner></mat-spinner>
    </div>
}
```

loginInterceptor.interceptor.ts ➔

```
import {  HttpInterceptorFn} from '@angular/common/http';
import { catchError, finalize, throwError } from 'rxjs';
import { LoadingService } from '../services/loading.service';
import { inject } from '@angular/core';
import { Router } from '@angular/router';


export const loginInterceptor: HttpInterceptorFn = (req, next) => {
  const router = inject(Router);
  const loaderService = inject(LoadingService);
  const startTime = Date.now();

  loaderService.setLoader(true);

  const modifiedReq = req.clone({
    setHeaders: {
      'secretMessage' : 'I am an Interceptor'
    }
  });

  console.log('intercepting login api')



  return next(modifiedReq).pipe(
    catchError(error => {
      if(error.status === 401){
        console.log('unauthorized request. redirecting to login.')
        router.navigate(['ddd'])
      }
      return throwError(error);
    }),
    finalize(()=>{
      setTimeout(() => {
        loaderService.setLoader(false);
        const endTime = Date.now();
        const responseTime = endTime - startTime ;
        console.log('interceptor ended')
        console.log(`Request ccompletedd in ${responseTime} ms`)
      }, 500);

    })
  )
};
```

User.service.ts ➔

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class UserService {

  private getUserURL =
"https://interceptortesting.free.beeceptor.com/my/api/path"
  constructor(private http: HttpClient) { }

  getUser(){
    return this.http.get(this.getUserURL);
  }

}
```

Loadingspinner.service.ts ➔

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
```

```
@Injectable({
  providedIn: 'root'
})
export class LoadingService {
  private loadingObj:BehaviorSubject<boolean> = new
BehaviorSubject<boolean>(false);
  loading$ = this.loadingObj.asObservable();

  constructor() { }

  setLoader(loaderFlag:boolean){
    this.loadingObj.next(loaderFlag);
  }

}
```