```
import { BrowserModule } from "@angular/platform-browser";
import { BrowserAnimationsModule } from "@angular/platform-browser/animations";
import { NgModule } from "@angular/core";
import { HttpClientModule, HTTP_INTERCEPTORS } from "@angular/common/http";

import { AppComponent } from "./app.component";
import { HeaderComponent } from "./header/header.component";
import { AppRoutingModule } from "./app-routing.module";
import { AuthInterceptor } from "./auth/auth-interceptor";
import { ErrorInterceptor } from "./error-interceptor";
import { ErrorComponent } from "./error/error.component";
import { AngularMaterialModule } from "./angular-material.module";
import { PostsModule } from "./posts/posts.module";

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    ErrorComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    HttpClientModule,
    AngularMaterialModule,
    PostsModule
  ],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: ErrorInterceptor, multi: true }
  ],
  bootstrap: [AppComponent],
  entryComponents: [ErrorComponent]
})
export class AppModule {}
```

```
<mat-card>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <form (submit)="onLogin(loginForm)" #loginForm="ngForm" *ngIf="!isLoading">
    <mat-form-field>
      <input matInput name="email" ngModel type="email" placeholder="E-Mail" #emailInput="ngModel" required email>
      <mat-error *ngIf="emailInput.invalid">Please enter a valid email.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input type="password" name="password" ngModel matInput placeholder="Password" #passwordInput="ngModel" required>
      <mat-error *ngIf="passwordInput.invalid">Please enter a valid password.</mat-error>
    </mat-form-field>
    <button mat-raised-button color="accent" type="submit" *ngIf="!isLoading">Login</button>
  </form>
</mat-card>
```

```
import { Component, OnInit, OnDestroy } from "@angular/core";
import { NgForm } from "@angular/forms";
import { Subscription } from "rxjs";

import { AuthService } from "../auth.service";

@Component({
  templateUrl: "./login.component.html",
  styleUrls: ["./login.component.css"]
})
export class LoginComponent implements OnInit, OnDestroy {
  isLoading = false;
  private authStatusSub: Subscription;

  constructor(public authService: AuthService) {}

  ngOnInit() {
    this.authStatusSub = this.authService.getAuthStatusListener().subscribe(
      authStatus => {
        this.isLoading = false;
      }
    );
  }

  onLogin(form: NgForm) {
```

```
    if (form.invalid) {
      return;
    }
    this.isLoading = true;
    this.authService.login(form.value.email, form.value.password);
  }

  ngOnDestroy() {
    this.authStatusSub.unsubscribe();
  }
}
```

```
mat-form-field {
  width: 100%;
}

mat-spinner {
  margin: auto;
}
```

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Router } from "@angular/router";
import { Subject } from "rxjs";

import { environment } from "../../environments/environment";
import { AuthData } from "./auth-data.model";

const BACKEND_URL = environment.apiUrl + "/user/";

@Injectable({ providedIn: "root" })
export class AuthService {
  private isAuthenticated = false;
  private token: string;
  private tokenTimer: any;
  private userId: string;
  private authStatusListener = new Subject<boolean>();

  constructor(private http: HttpClient, private router: Router) {}

  getToken() {
    return this.token;
  }

  getIsAuth() {
    return this.isAuthenticated;
  }

  getUserId() {
    return this.userId;
  }

  getAuthStatusListener() {
    return this.authStatusListener.asObservable();
  }

  createUser(email: string, password: string) {
    const authData: AuthData = { email: email, password: password };
    this.http.post(BACKEND_URL + "/signup", authData).subscribe(
      () => {
        this.router.navigate(["/"]);
      },
      error => {
        this.authStatusListener.next(false);
      }
    );
  }

  login(email: string, password: string) {
    const authData: AuthData = { email: email, password: password };
    this.http
      .post<{ token: string; expiresIn: number; userId: string }>(
        BACKEND_URL + "/login",
        authData
      )
      .subscribe(
        response => {
          const token = response.token;
```

```
        this.token = token;
        if (token) {
          const expiresInDuration = response.expiresIn;
          this.setAuthTimer(expiresInDuration);
          this.isAuthenticated = true;
          this.userId = response.userId;
          this.authStatusListener.next(true);
          const now = new Date();
          const expirationDate = new Date(
            now.getTime() + expiresInDuration * 1000
          );
          console.log(expirationDate);
          this.saveAuthData(token, expirationDate, this.userId);
          this.router.navigate(["/"]);
        }
      },
      error => {
        this.authStatusListener.next(false);
      }
    );
  }

  autoAuthUser() {
    const authInformation = this.getAuthData();
    if (!authInformation) {
      return;
    }
    const now = new Date();
    const expiresIn = authInformation.expirationDate.getTime() - now.getTime();
    if (expiresIn > 0) {
      this.token = authInformation.token;
      this.isAuthenticated = true;
      this.userId = authInformation.userId;
      this.setAuthTimer(expiresIn / 1000);
      this.authStatusListener.next(true);
    }
  }

  logout() {
    this.token = null;
    this.isAuthenticated = false;
    this.authStatusListener.next(false);
    this.userId = null;
    clearTimeout(this.tokenTimer);
    this.clearAuthData();
    this.router.navigate(["/"]);
  }

  private setAuthTimer(duration: number) {
    console.log("Setting timer: " + duration);
    this.tokenTimer = setTimeout(() => {
      this.logout();
    }, duration * 1000);
  }

  private saveAuthData(token: string, expirationDate: Date, userId: string) {
    localStorage.setItem("token", token);
    localStorage.setItem("expiration", expirationDate.toISOString());
    localStorage.setItem("userId", userId);
  }

  private clearAuthData() {
    localStorage.removeItem("token");
    localStorage.removeItem("expiration");
    localStorage.removeItem("userId");
  }

  private getAuthData() {
    const token = localStorage.getItem("token");
    const expirationDate = localStorage.getItem("expiration");
    const userId = localStorage.getItem("userId");
    if (!token || !expirationDate) {
      return;
    }
    return {
      token: token,
      expirationDate: new Date(expirationDate),
      userId: userId,
    };
  }
}


import {
```

```
  HttpInterceptor,
  HttpRequest,
  HttpHandler,
  HttpErrorResponse
} from "@angular/common/http";
import { catchError } from "rxjs/operators";
import { throwError } from "rxjs";
import { Injectable } from "@angular/core";
import { MatDialog } from "@angular/material";

import { ErrorComponent } from "./error/error.component";
import { ErrorService } from "./error/error.service";

@Injectable()
export class ErrorInterceptor implements HttpInterceptor {

  constructor(private dialog: MatDialog, private errorService: ErrorService) {}

  intercept(req: HttpRequest<any>, next: HttpHandler) {
    return next.handle(req).pipe(
      catchError((error: HttpErrorResponse) => {
        let errorMessage = "An unknown error occurred!";
        if (error.error.message) {
          errorMessage = error.error.message;
        }
        this.dialog.open(ErrorComponent, {data: {message: errorMessage}});
        // this.errorService.throwError(errorMessage);
        return throwError(error);
      })
    );
  }
}
```