```
<mat-card>
  <mat-card-header>
    <mat-card-title> Login User </mat-card-title>
  </mat-card-header>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <form (submit)="onLogin(loginForm)" #loginForm="ngForm" *ngIf="!isLoading">
    <mat-form-field>
      <input
        matInput
        name="username"
        ngModel
        type="text"
        placeholder="username"
        #username="ngModel"
        required
      />
      <mat-error *ngIf="username.invalid"
      >Please enter a valid username.</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>

<mat-card>
  <mat-card-header>
    <mat-card-title> Create User </mat-card-title>
  </mat-card-header>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <form (submit)="onCreateUser(userForm)" #userForm="ngForm" *ngIf="!isLoading">
    <mat-form-field>
      <input
        matInput
        name="user"
        ngModel
        type="text"
        placeholder="user"
        #user="ngModel"
        required
      />
      <mat-error *ngIf="user.invalid">Please enter a valid username.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        type="password"
        name="p"
        ngModel
        matInput
        placeholder="Password"
        #p="ngModel"
        required
      />
      <mat-error *ngIf="p.invalid">Please enter a valid password.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        matInput
        name="cageId"
        ngModel
        type="number"
        placeholder="cage ID"
        #cageId="ngModel"
        required
      />
      <mat-error *ngIf="cageId.invalid">Please enter a valid cage.</mat-error>
    </mat-form-field>
```

```
    <mat-form-field>
      <input
        matInput
        name="boxId"
        ngModel
        type="number"
        placeholder="Box ID"
        #boxId="ngModel"
        required
      />
      <mat-error *ngIf="boxId.invalid">Please enter a valid box.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        matInput
        name="status"
        ngModel
        type="text"
        placeholder="status"
        #status="ngModel"
        required
      />
      <mat-error *ngIf="status.invalid">Please enter a valid status.</mat-error>
    </mat-form-field>
    <mat-checkbox [(ngModel)]="isAdmin" [ngModelOptions]="{standalone: true}"
    >Checked</mat-checkbox
    >
    <button mat-raised-button color="accent" type="submit" *ngIf="!isLoading">
      create
    </button>
  </form>
</mat-card>

<mat-card>
  <mat-card-header>
    <mat-card-title> Modify User </mat-card-title>
  </mat-card-header>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <form
    (submit)="onModifyUser(modifyForm)"
    #modifyForm="ngForm"
    *ngIf="!isLoading"
  >
    <mat-form-field>
      <input
        matInput
        name="a"
        ngModel
        type="text"
        placeholder="user"
        #a="ngModel"
        required
      />
      <mat-error *ngIf="a.invalid">Please enter a valid username.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        type="password"
        name="b"
        ngModel
        matInput
        placeholder="Password"
        #b="ngModel"
      />
      <mat-error *ngIf="b.invalid">Please enter a valid password.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        matInput
        name="c"
        ngModel
        type="number"
        placeholder="cage ID"
        #c="ngModel"
        required
      />
      <mat-error *ngIf="c.invalid">Please enter a valid cage.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        matInput
        name="d"
        ngModel
        type="number"
```

```
        placeholder="Box ID"
        #d="ngModel"
        required
      />
      <mat-error *ngIf="d.invalid">Please enter a valid box.</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input
        matInput
        name="e"
        ngModel
        type="text"
        placeholder="status"
        #e="ngModel"
        required
      />
      <mat-error *ngIf="e.invalid">Please enter a valid status.</mat-error>
    </mat-form-field>
    <button mat-raised-button color="accent" type="submit" *ngIf="!isLoading">
      modify
    </button>
  </form>
</mat-card>

<mat-card>
  <mat-card-header>
    <mat-card-title> Delete User </mat-card-title>
  </mat-card-header>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <form
    (submit)="onDeleteUser(loginForm)"
    #loginForm="ngForm"
    *ngIf="!isLoading"
  >
    <mat-form-field>
      <input
        matInput
        name="deleteUsername"
        ngModel
        type="text"
        placeholder="username"
        #deleteUsername="ngModel"
        required
      />
      <mat-error *ngIf="deleteUsername.invalid"
      >Please enter a valid username.</mat-error
      >
    </mat-form-field>
    <button mat-raised-button color="accent" type="submit" *ngIf="!isLoading">
      delete
    </button>
  </form>
</mat-card>

<mat-card>
  <mat-card-header>
    <mat-card-title> Get Users </mat-card-title>
  </mat-card-header>
  <mat-spinner *ngIf="isLoading"></mat-spinner>
  <button
    mat-raised-button
    color="accent"
    type="button"
    *ngIf="!isLoading"
    (click)="onGetUsers()"
  >
    get
  </button>
</mat-card>

<-- css -->


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

mat-spinner {
  margin: auto;
}

<-- ts -->

import { Component } from "@angular/core";
```

```
import { NgForm } from "@angular/forms";

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

@Component({
  selector: "app-login",
  templateUrl: "./login.component.html",
  styleUrls: ["./login.component.css"],
})
export class LoginComponent {
  isLoading = false;
  isAdmin = false;

  constructor(public authService: AuthService) {}

  onLogin(form: NgForm) {
    if (form.invalid) {
      return;
    }
    this.isLoading = true;
    this.authService.login(form.value.username, form.value.password).subscribe(
      () => {
        console.log("Success");
        form.reset();
        this.isLoading = false;
      },
      error => {
        console.log(error);
      }
    );

    this.isLoading = false;
  }

  onCreateUser(form: NgForm) {
    this.isLoading = true;
    this.authService
      .createUser(
        form.value.user,
        form.value.p,
        this.isAdmin,
        form.value.cageId,
        form.value.boxId,
        form.value.status
      )
      .subscribe(
        () => {
          console.log("Success creating user");
        },
        error => {
          console.log(`${error}`);
        }
      );
    this.isLoading = false;
  }
  onModifyUser(form: NgForm) {
    this.isLoading = true;
    let admin = true;
    this.authService
      .modifyUser(
        form.value.a,
        form.value.b,
        admin,
        form.value.c,
        form.value.d,
        form.value.e
      )
      .subscribe(
        () => {
          console.log("Success");
        },
        error => {
          console.log(`${error}`);
        }
      );
    this.isLoading = false;
  }

  onDeleteUser(form: NgForm) {
    this.isLoading = true;

    this.authService.deleteUser(form.value.deleteUsername).subscribe(
      () => {
        console.log("Success");
```

```
        this.isLoading = false;
      },
      error => {
        console.log(`${error}`);
        this.isLoading = false;
      }
    );
  }

  onGetUsers() {
    this.isLoading = true;
    this.authService.getUsers().subscribe(
      users => {
        console.log(users);
      },
      error => {
        console.log(`${error}`);
      }
    );
    this.isLoading = false;
  }
}

<!-- Auth Service -->

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

import { environment } from "../../environments/environment";

const BACKEND_URL = environment.apiUrl;

export enum STATUS {
  ACTIVE = "active",
  INACTIVE = "inactive",
  ERROR = "error",
}
export interface AuthData {
  username: number;
  password?: string;
}
export interface UserData {
  username: number;
  password?: string;
  admin: boolean;
  cageId: number;
  boxId: number;
  status: STATUS;
}

@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) {}

  createUser(
    username: number,
    password: string,
    admin: boolean,
    cageId: number,
    boxId: number,
    status: STATUS
  ) {
    const userData: UserData = {
      username: username,
      password: password,
      admin: admin,
      cageId: cageId,
      boxId: boxId,
      status: status,
    };
    return this.http.post(BACKEND_URL + "/admin/users/register", userData);
  }

  login(username: number, password: string) {
    const authData: AuthData = { username: username, password: password };
    return this.http.post<AuthData>(`${BACKEND_URL}/login`, authData);
  }
```

```
  modifyUser(
    username: number,
    password: string,
    admin: boolean,
    cageId: number,
    boxId: number,
    status: STATUS
  ) {
    const userData: UserData = {
      username: username,
      password: password,
      admin: admin,
      cageId: cageId,
      boxId: boxId,
      status: status,
    };
    return this.http.put(
      `${BACKEND_URL}/admin/user/${username}/modify`,
      userData
    );
  }

  deleteUser(username: number) {
    return this.http.delete(`${BACKEND_URL}/admin/user/${username}/delete`);
  }

  getUsers() {
    return this.http.get(`${BACKEND_URL}/admin/users`);
  }

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

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

<!-- Auth interceptor -->

import {
  HttpInterceptor,
  HttpRequest,
  HttpHandler
} from "@angular/common/http";
import { Injectable } from "@angular/core";

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

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) {}

  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const authToken = this.authService.getToken();
    const authRequest = req.clone({
      headers: req.headers.set("Authorization", "Bearer " + authToken)
    });
    return next.handle(authRequest);
  }
}

<!-- Auth guard -->

import {
  CanActivate,
  ActivatedRouteSnapshot,
  RouterStateSnapshot,
  Router
} from "@angular/router";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";

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

@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private authService: AuthService, private router: Router) {}
```

```
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): boolean | Observable<boolean> | Promise<boolean> {
    const isAuth = this.authService.getIsAuth();
    if (!isAuth) {
      this.router.navigate(['/login']);
    }
    return isAuth;
  }
}

<!-- Error interceptor -->

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);
      })
    );
  }
}

<!-- Error.comp.html -->

<h1 mat-dialog-title>An Error Occurred!</h1>
<div mat-dialog-content>
  <p class="mat-body-1">{{ data?.message }}</p>
</div>
<div mat-dialog-actions>
  <button mat-button mat-dialog-close>Okay</button>
</div>

<!-- Error.comp.css -->

:host {
  position: fixed;
  top: 150px;
  left: 40%;
  width: 20%;
  background: white;
  padding: 0.5rem;
  box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.5);
}

<!-- Error.Service.ts -->

import { Subject } from "rxjs";
import { Injectable } from "@angular/core";

@Injectable({ providedIn: "root" })
export class ErrorService {
  private errorListener = new Subject<string>();

  getErrorListener() {
    return this.errorListener.asObservable();
  }
```

```
  throwError(message: string) {
    this.errorListener.next(message);
  }

  handleError() {
    this.errorListener.next(null);
  }
}
```