**Unicom TIC - Lab worksheet.**

**Week 09/Nov/2024 - 10/Nov/2024**

1. Modify your Task Management application to use ngx-bootstrap **modals** for displaying the *Add Task*, *Edit Task*, *Add User*, and *Edit User* forms as pop-up dialogs instead of separate pages.
2. Create two new angular components for layouts: one for the login page and another for admin pages.
3. Implement an authentication guard to control access to the admin pages(task management, user management and dashboard), based on a token stored in localStorage. Follow these requirements:
4. ***Login Page****: This should be accessible without authentication. When a user successfully logs in, store a token in localStorage and redirect them to the admin dashboard.*
5. ***Admin Pages****: These should only be accessible if the token exists in localStorage. Use an Angular Auth Guard to protect the authorized routes.*
6. ***Auth Guard Implementation****:*
   * *Create an AuthGuard that checks if a token exists in localStorage.*
   * *If the token exists, allow access to the requested route.*
   * *If the token is missing or invalid, redirect the user to the login page.*
7. ***Layout Handling****:*
   * *Implement layout components for the login page and admin pages.*
   * *The login page should not display the admin navigation bar, while the admin pages should include the navigation bar and other admin-specific UI elements.*

**Implementation approach**

Refer below link to design the login page.

<https://bootstrapbrain.com/demo/components/logins/login-6/#how-to-use>

1. Use the command below to create the AuthGuard.

***ng generate guard auth***

1. **Implementing canActivate() Method**

In the `AuthGuard` class, implement the `canActivate()` method using the `AuthService` to check if a user is authenticated. If not authenticated, redirect them to `/login` using Angular’s `Router`.

1. **Protecting Routes with Auth Guard**

Configure the `AppRoutingModule` to protect a route `/dashboard` with the `AuthGuard`. Write the routing configuration for this.

1. **Passing Intended URL as Query Parameter**

Modify the `canActivate()` method in `AuthGuard` to pass the intended URL as a query parameter when redirecting to `/login`. Use Angular’s `ActivatedRouteSnapshot` to capture the intended URL.

1. **Role-Based Access Control**

Extend the `AuthService` to include a method `hasRole()` that checks if a user has a specific role stored in `localStorage`. Implement the `hasRole()` method to return `true` if the user role matches the given role.

Update the `AuthGuard` to allow only users with the role `admin` to access the `/admin` route. Assume the `AuthService`’s `hasRole()` method is already implemented.

1. **Token Expiration Handling**

Modify `AuthService` to add a method `isTokenExpired()` that checks if the token is expired based on a stored `expiryTime`. Assume the expiry time is stored in `localStorage` when the user logs in.

1. **Error Handling in Auth Guard**

Add error handling in the `canActivate()` method of `AuthGuard` to catch unexpected errors (e.g., network issues) and log them to the console. Redirect users to an error page `/error` if an error occurs.