Authentication and security

Authentication Flow

1. **Login** (`/login`)
   - User submits credentials
   - API returns JWT token
   - Token stored in `localStorage` as `jwt`
   - User details and roles stored in `localStorage`
2. **Token Management**
   - JWT token sent with every API request via `Authorization` header
   - Token validated on each route navigation via `AuthGuard`
   - Invalid tokens redirect to login page
3. **Logout**
   - Clear `localStorage`
   - Redirect to login page

Guards

AuthGuard (`auth.guard.ts`)

- Protects all authenticated routes
- Validates JWT token on navigation
- Calls `guardianTokens()` API to verify token validity
- Redirects to `/login` if authentication fails

```
// Usage in routing
{
  path: 'home',
  loadChildren: () => import('./home/home.module').then(m => m.HomeModule),
  canActivate: [AuthGuard]
}
```

RoleGuard (`role.guard.ts`)

- Provides role-based access control
- Checks user roles from `localStorage`

- Can be used for feature-specific authorization

Security Best Practices

1. **Token Storage**: JWT tokens stored in `localStorage` (consider `httpOnly` cookies for enhanced security)
2. **HTTPS**: Always use HTTPS in production
3. **Token Expiration**: Implement token refresh mechanism
4. **XSS Protection**: Angular's built-in sanitization helps prevent XSS
5. **CSRF Protection**: Use Angular's CSRF token support for POST requests

---

Revision #1
Created 24 February 2026 07:11:13 by ondeliveloper
Updated 24 February 2026 07:11:44 by ondeliveloper