Authentication & Security

JWT Authentication

The application uses JWT (JSON Web Token) for authentication:

1. **Login Flow**:

```
// User logs in
this.apiService.getUser(username, password).subscribe(
  response => {
    // Store JWT token
    localStorage.setItem('jwt', response.token);
    // Navigate to home
    this.router.navigate(['/']);
  }
);
```

2. **Token Storage**: JWT tokens are stored in localStorage with key `'jwt'`
3. **Token Injection**: The `AuthInterceptor` automatically adds the token to all requests:

```
const token = localStorage.getItem("jwt")
  ? `Bearer ${localStorage.getItem("jwt")}`
  : "";

const authReq = req.clone({
  headers: req.headers.set("authorization", token)
});
```

4. **Token Verification**: AuthGuard verifies tokens before allowing access to protected routes

HTTP Interceptor (`auth.interceptor.ts`)

Handles authentication and error responses:

Features:

- Automatically adds JWT token to request headers
- Handles 401/403 unauthorized responses
- Redirects to login on session expiration
- Shows user-friendly error messages
- Clears localStorage on auth failure

Error Handling:

```
private handleAuthError(err: HttpErrorResponse): Observable<any> {
  if (err.status === 401 || err.status === 403) {
    // Show error notification
    this.Toast.fire({
      title: "Sesi berakhir, mohon login kembali",
      icon: "error"
    });

    // Redirect to login
    this.router.navigate(["/login"]);

    // Clear token
    localStorage.removeItem("jwt");
  }
  return throwError(err);
}
```

Security Best Practices

1. **Route Guards**: Protect sensitive routes with AuthGuard
2. **Token Expiration**: Tokens are verified on each protected route access
3. **Secure Communication**: All API calls use HTTPS
4. **XSS Protection**: Angular's built-in sanitization
5. **CSRF Protection**: Handled by backend API

Revision #1
Created 24 February 2026 07:24:31 by ondeliveloper
Updated 24 February 2026 07:24:54 by ondeliveloper