# Global Token Function Design

Yes, absolutely! You can create global/reusable functions for token handling that can be used throughout your application. Here's how this would work:

## Global Token Functions Approach

After the login page creates the initial JWT token and sets it as an HTTP-only cookie, you would have:

### 1. A Global Middleware Function (server-side) that automatically:

- Extracts the token from the HTTP-only cookie on every request
- Verifies the JWT signature and expiration
- Verifies the CSRF token if you're using one
- Attaches the decoded user information to the request object
- Allows the request to proceed if valid, or redirects to login if invalid

### 2. A Global Send/Refresh Function (server-side) that:

- Can refresh an expiring token
- Re-sets the cookie with updated expiration
- Maintains the same security settings (httpOnly, secure, sameSite)

## The Flow

- Login page creates the initial token and sets the cookie (one-time setup)
- Every subsequent page request automatically includes the cookie (browser behavior)
- The global middleware runs on every protected route to verify the token
- Pages are served or requests are processed only if the token is valid
- No page-specific token handling code needed - it's all centralized

## Benefits of This Approach

- **Write once, use everywhere** - no code duplication
- **Consistent security** across all pages
- **Easy to maintain** and update security logic in one place
- **Individual pages don't need to know** about authentication details
- **Can easily add features** like token refresh, logout, or session management globally

This is actually the standard pattern in most web frameworks - they provide authentication middleware that handles all token verification globally, so individual routes/pages just need to specify whether they require authentication or not.