# Full Sequential Process of Spring Security Authentication Flow

## 1️⃣ User Submits Login Request

The user enters email and password in the login form.  
The request is sent to Spring Security’s authentication process.  
  
Example Request:  
POST /login  
email: user@example.com  
password: secret

## 2️⃣ UsernamePasswordAuthenticationFilter Intercepts the Request

This filter extracts the email and password from the request.  
It creates a UsernamePasswordAuthenticationToken containing these credentials.

## 3️⃣ AuthenticationManager Calls DaoAuthenticationProvider

The AuthenticationManager delegates authentication to DaoAuthenticationProvider.  
  
Spring Security Code:  
Authentication authResult = authenticationManager.authenticate(authenticationToken);

## 4️⃣ BlogUserDetailsService Fetches User

Calls loadUserByUsername(email) to fetch user details from the database.  
  
Your Code in BlogUserDetailsService:  
@Override  
public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {  
 return userRepo.findByEmail(email)  
 .map(BlogUserDetails::new)  
 .orElseThrow(() -> new UsernameNotFoundException("User not found: " + email));  
}

## 5️⃣ Password is Verified

Compares the entered password with the hashed password from the database using PasswordEncoder.  
  
Spring Security Code:  
if (!passwordEncoder.matches(enteredPassword, user.getPassword())) {  
 throw new BadCredentialsException("Incorrect password");  
}

## 6️⃣ Account Status is Checked

Spring Security calls the following methods in BlogUserDetails:  
- isAccountNonExpired() → Checks if the account is expired  
- isAccountNonLocked() → Checks if the account is locked  
- isCredentialsNonExpired() → Checks if the password is expired  
- isEnabled() → Checks if the user is active

## 7️⃣ Roles & Authorities are Loaded

Calls getAuthorities() to load user roles and permissions.  
  
Your Code in BlogUserDetails:  
@Override  
public Collection<? extends GrantedAuthority> getAuthorities() {  
 return List.of(new SimpleGrantedAuthority("ROLE\_USER"));  
}

## 8️⃣ SecurityContextHolder Stores Authentication

Spring Security stores the authenticated user in the SecurityContextHolder.  
  
Spring Security Code:  
SecurityContextHolder.getContext().setAuthentication(authentication);

## 9️⃣ User is Redirected or Receives Token

If authentication is successful:  
- For Web Apps: Redirects to /home.  
- For REST APIs: Returns a JWT token.  
  
Example Response:  
{  
 "token": "eyJhbGciOiJIUzI1NiIsIn..."  
}

## 🔄 10️⃣ Future Requests Are Authenticated

On every request, Spring Security checks if the user is already authenticated.  
If authenticated, the request is processed normally; otherwise, the user is redirected to login.  
  
Spring Security Code:  
Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
if (auth == null || !auth.isAuthenticated()) {  
 response.sendRedirect("/login");  
}

## ✅ Final Summary (Full Step-by-Step Process)

| Step | Action |  
|------|--------------|  
| \*\*1️⃣ User submits login form\*\* | Sends email & password to `/login` |  
| \*\*2️⃣ UsernamePasswordAuthenticationFilter intercepts\*\* | Extracts email & password, creates auth token |  
| \*\*3️⃣ AuthenticationManager processes authentication\*\* | Calls `DaoAuthenticationProvider` |  
| \*\*4️⃣ BlogUserDetailsService fetches user\*\* | Calls `loadUserByUsername(email)` |  
| \*\*5️⃣ Password is verified\*\* | Compares entered password with stored hash |  
| \*\*6️⃣ Account status is checked\*\* | Calls `isEnabled()`, `isAccountNonLocked()`, etc. |  
| \*\*7️⃣ Roles & authorities are loaded\*\* | Calls `getAuthorities()` (ROLE\_USER) |  
| \*\*8️⃣ Authentication is stored\*\* | `SecurityContextHolder` stores authentication |  
| \*\*9️⃣ User is redirected or receives JWT\*\* | Web apps → Redirect, APIs → JWT Token |  
| \*\*🔄 1️⃣0️⃣ Future requests are authenticated\*\* | Checks session or JWT for authentication before processing requests |

## 🚀 Next Steps

Right now, every user has only "ROLE\_USER".  
👉 Would you like to load dynamic roles from the database (e.g., `ROLE\_ADMIN`)? I can show you how! 😊