Role Based Authorization + Add Role inside JWT Token

This document explains the exact file-wise changes required to implement role-based authorization in JWT.

File / Module	What to Implement	Why We Do It
JwtService.java	Update generateToken(UserDetails userDetails) to include claims map and put role inside token.	So the JWT contains user role and we can apply role-based authorization.
AuthServiceImpl.java	Update login(): load UserDetails and call jwtService.generateToken(userDetails) instead of passing email.	Because token generation now needs role data from UserDetails.
SecurityConfig.java	Update authorizeHttpRequests(): permit /api/auth/**, restrict /api/admin/** to ADMIN, allow /api/users/** to ADMIN + CUSTOMER.	Because Spring Security must enforce access rules based on roles.
AdminController.java	Create controller /api/admin/test endpoint for testing ADMIN access.	To confirm that ADMIN role can access admin-only APIs.
UserTestController.java	Create controller /api/users/test endpoint for testing CUSTOMER + ADMIN access.	To confirm both CUSTOMER and ADMIN roles can access user APIs.
Postman Testing	Login → copy token → call /api/admin/test and /api/users/test using Authorization: Bearer .	To verify role-based restriction works (ADMIN allowed, CUSTOMER gets 403 for admin API).

Expected Output:

• If role = ADMIN → /api/admin/test works
• If role = CUSTOMER → /api/admin/test gives 403
• /api/users/test works for both ADMIN and CUSTOMER