**User** implements UserDetails

**Authority** implements GrantedAuthority

**JwtRequest** – username, password

**JwtResponse** – String token

**CustomUserDetailsService** implements UserDetailsService ->

User user = findByUsername -> if not found, throw exception -> else return user

**JwtUtil** – creates token

**JwtAuthenticationEntryPoint** implements AuthenticationEntryPoint {

// called when unauthorized user tries to access -> 401-> commence()}

**JwtAuthenticationFilter** extends OncePerRequestFilter ->

//task is to fetch jwt from header and validate it

**SecurityConfig** extends WebSecurityConfigurerAdapter

**AuthController**

**User** implements UserDetails

getAuthorities() -> get role names assigned to user in a set of type that extends GrantedAuthorty

**Authority** implements GrantedAuthority

String auth -> constructor -> getAuthority() – return auth;

**JwtRequest** – username, password

**JwtResponse** – String token

**CustomUserDetailsService** implements UserDetailsService

UserDetails **loadByUsername**(String username)

User user = findByUsername -> if not found, throw exception -> else return user

**JwtUtil -** String SECRET\_KEY = "secret";

//gets username from the token

**public** String **extractUsername**(String token)

//gets expiration date from token

**public** Date **extractExpiration**(String token)

Boolean **isTokenExpired**(String token)

//creates token

**private** String **createToken**(Map<String, Object> claims, String subject)

//generates token

**public** String **generateToken**(UserDetails userDetails)

Boolean **validateToken**(String token, UserDetails userDetails)

**JwtAuthenticationEntryPoint** implements AuthenticationEntryPoint

commence() // called when unauthorized user tries to access -> 401

**JwtAuthenticationFilter** extends OncePerRequestFilter

**protected** **void** **doFilterInternal**(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

token to be received from key Authorization ->

**final** String requestTokenHeader = request.getHeader("Authorization");

**if**(requestTokenHeader != **null** && requestTokenHeader.startsWith("Bearer "))

jwtToken = extract actual token by ignoring “Bearer “

username = jwtUtil.extractUsername(jwtToken)

//validate token

**if**(userName != **null** && SecurityContextHolder.*getContext*().getAuthentication() == **null**)

UserDetails userDetails = customUserDetailsService.loadUserByUserName(username)

if jwtUtil.validateToken(jwtToken, userDetails) ->

**userDetails** & **userDetails.getAuthorities** passed to

UsernamePasswordAuthenticationToken obj and then

***usernamePasswordAuthenticationToken.setDetails(new***

***WebAuthenticationDetailsSource().buildDetails(request));***

***SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);***

else -> invalid token

filterChain.doFilter(request, response);

**AuthController**

**private** **void** **authenticate**(String userName, String password)

authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(userName, password));

@PostMapping("/generate-token")

**public** ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest)

authenticate(jwtRequest.getUserName(), jwtRequest.getPassword());

UserDetails userDetails = **this**.customUserDetailsService.loadUserByUsername(jwtRequest.getUserName());

String token = **this**.jwtUtil.generateToken(userDetails);

**return** ResponseEntity.*ok*(**new** JwtResponse(token));

**Login.service.ts-**

getCurrentLoggedInUser()

generateToken(loginData: any)

loginUser(token: any)

getToken()

principal(user: any)

getPrincipal()

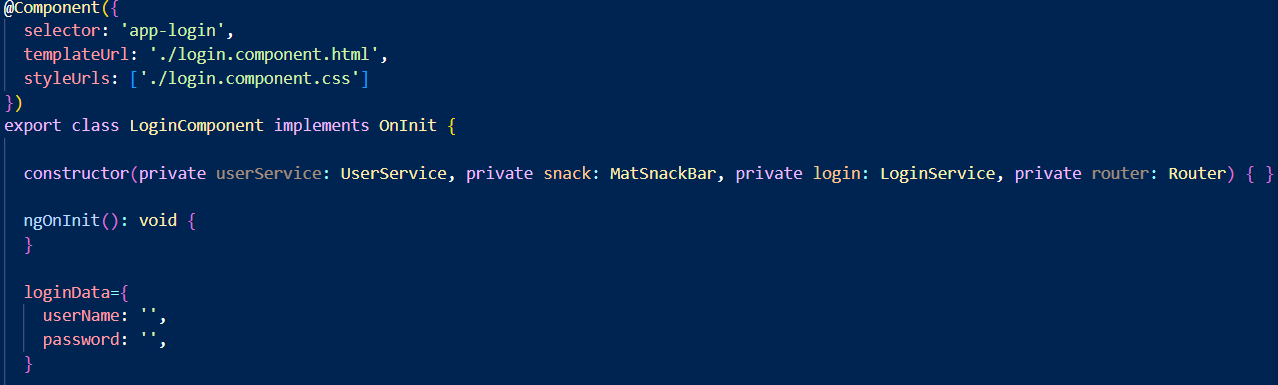
getPrincipalRole()

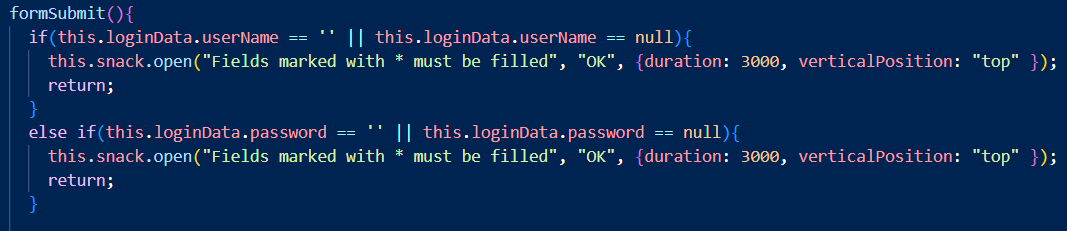
isLoggedIn()

logout()

setPrincipal(user: any)

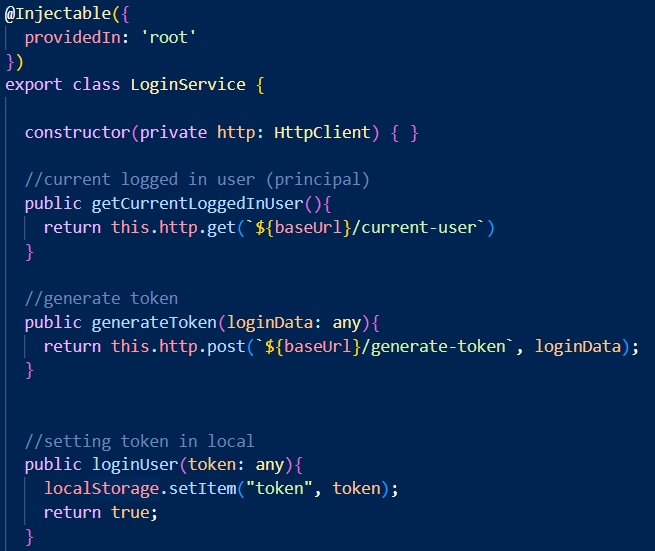
**Login.component.ts**



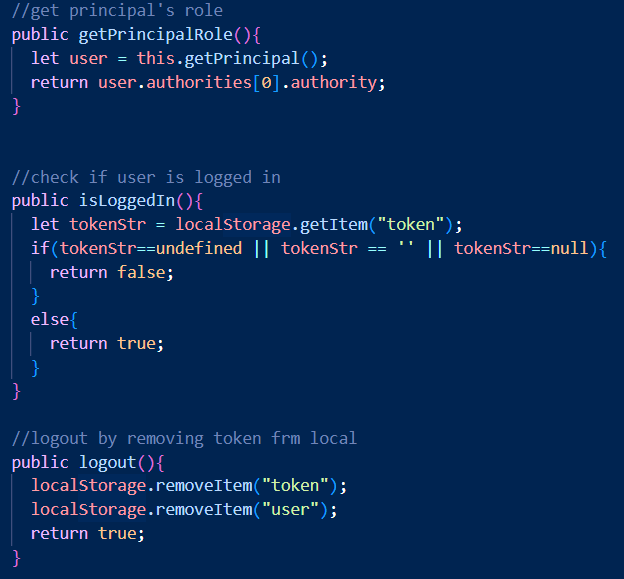


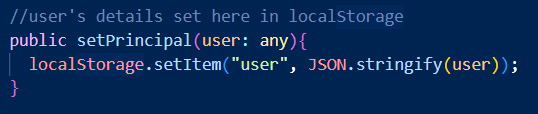


**Login.service.ts**

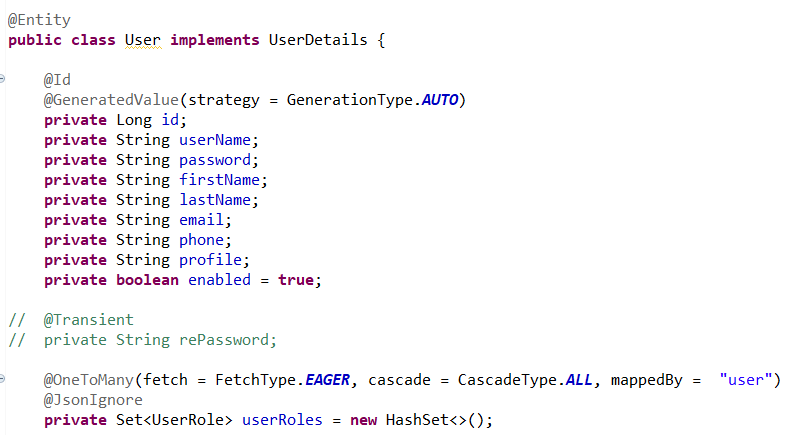
****

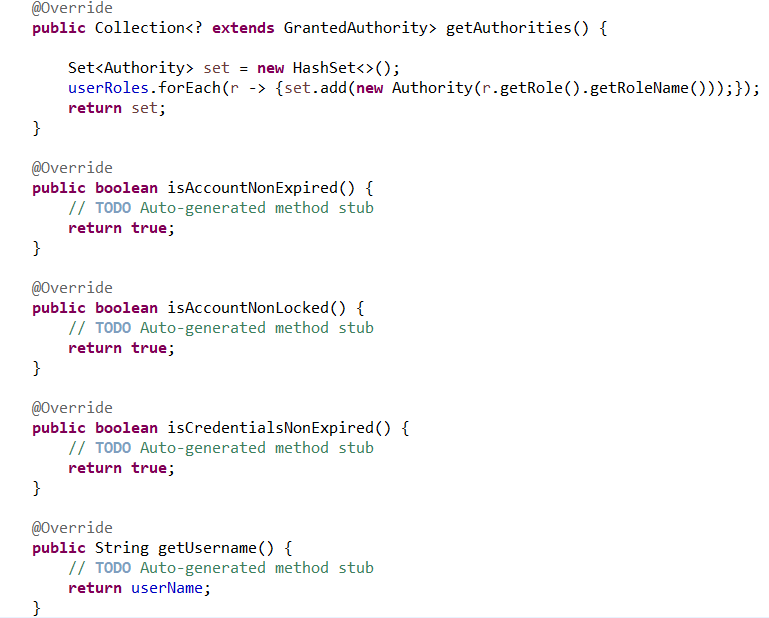
****

****

****

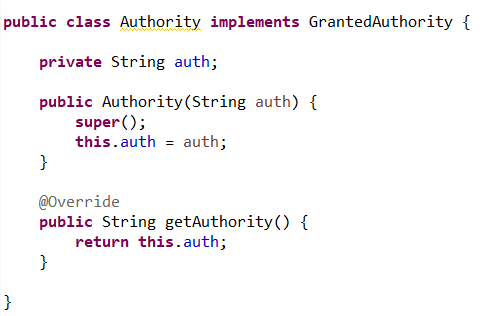
**User** implements UserDetails



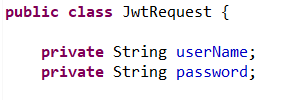


**Authority** implements GrantedAuthority

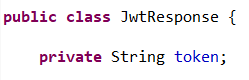
String auth -> constructor -> getAuthority()



**JwtRequest** – username, password



**JwtResponse** – String token

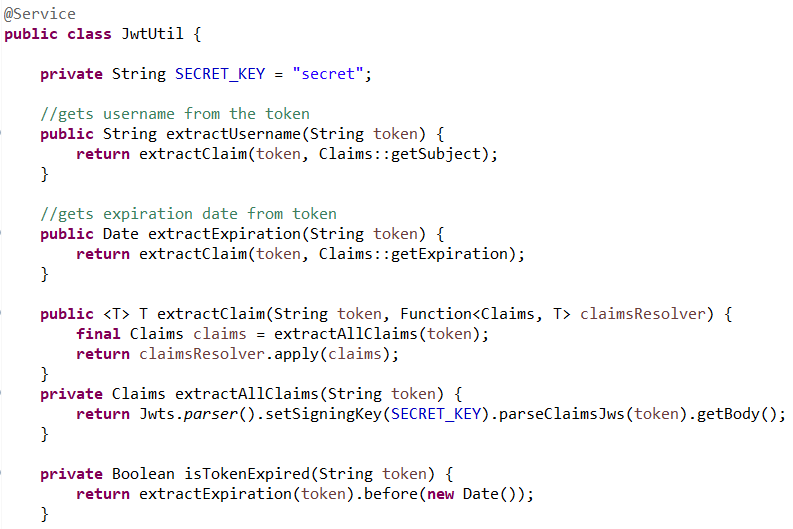


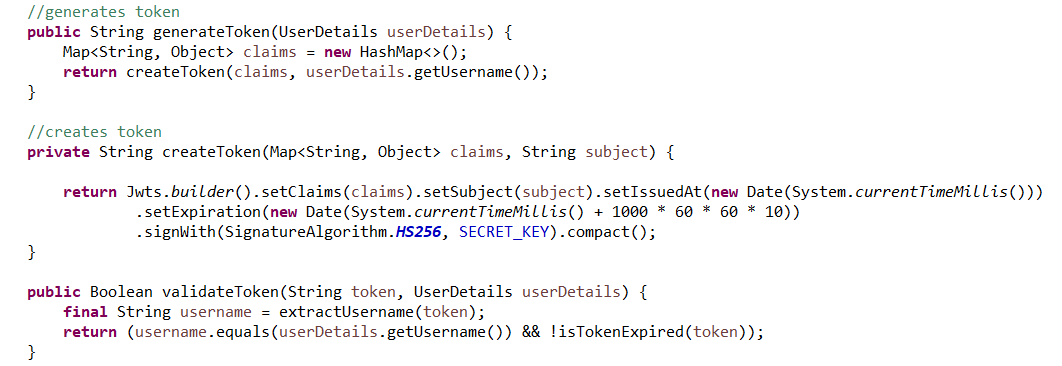
**CustomUserDetailsService** implements UserDetailsService

User user = findByUsername -> if not found, throw exception -> else return user

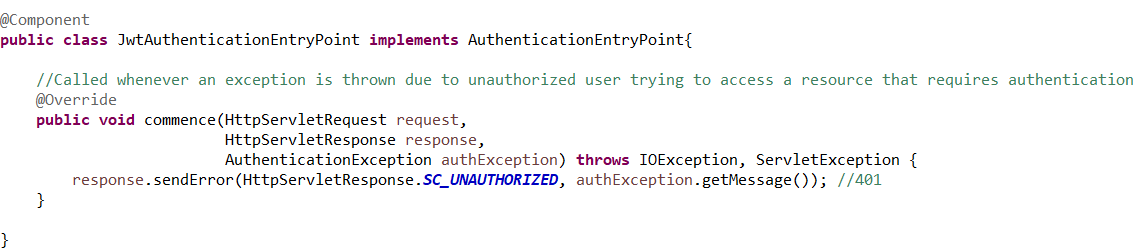


**JwtUtil** – creates token

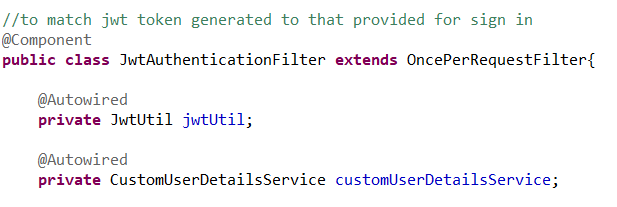




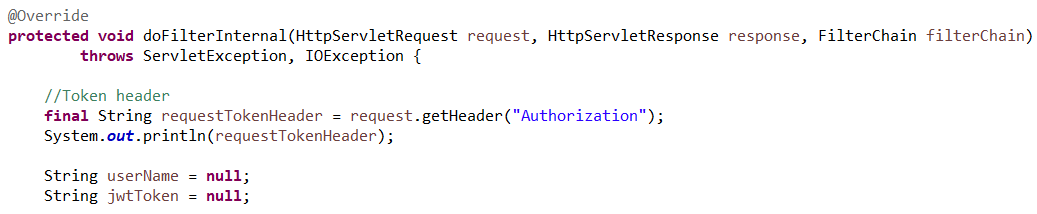
**JwtAuthenticationEntryPoint** implements AuthenticationEntryPoint { // called when unauthorized user tries to access -> 401-> commence()}



**JwtAuthenticationFilter** extends OncePerRequestFilter -> //task is to fetch jwt from header and validate it



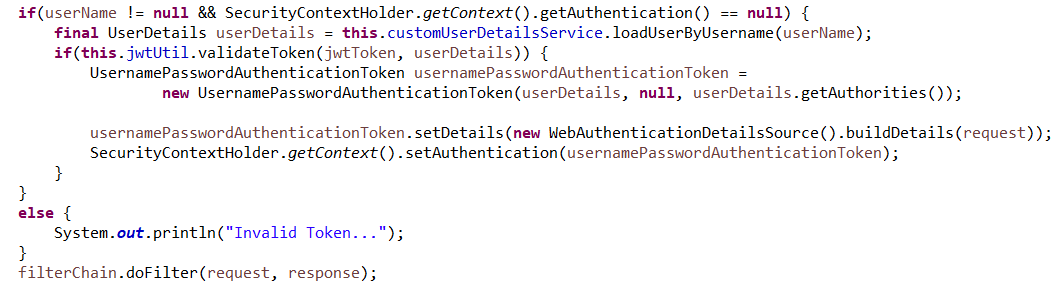
Remove header Authorization from token



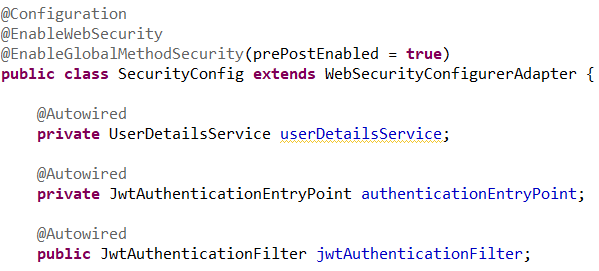


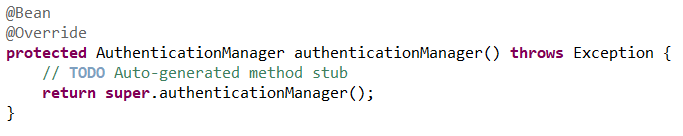


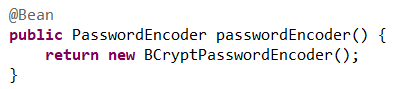
Validate token

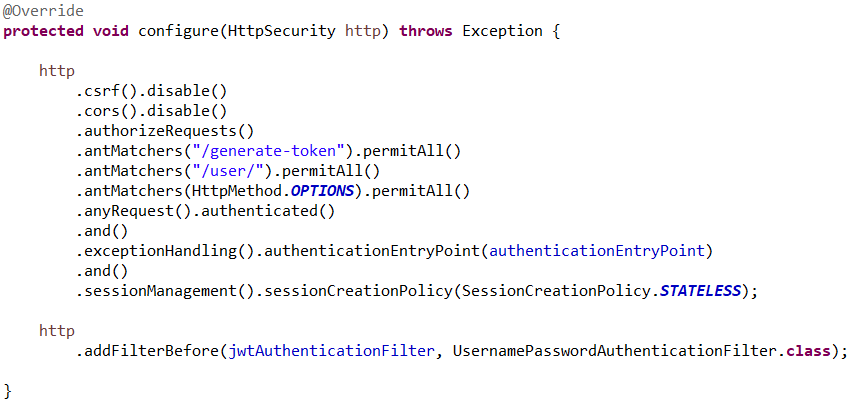


**SecurityConfig** extends WebSecurityConfigurerAdapter









**AuthController**

