<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "your-secret-key"; // Change this to a secure key

public String generateToken(UserDetails userDetails) {

return Jwts.builder()

.setSubject(userDetails.getUsername())

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10)) // 10 hours

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

public boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

public String extractUsername(String token) {

return extractClaims(token).getSubject();

}

private Claims extractClaims(String token) {

return Jwts.parser()

.setSigningKey(secretKey)

.parseClaimsJws(token)

.getBody();

}

private Boolean isTokenExpired(String token) {

return extractClaims(token).getExpiration().before(new Date());

}

}

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private JwtUtil jwtUtil;

@Autowired

private CustomUserDetailsService userDetailsService;

@Bean

public JwtRequestFilter jwtRequestFilter() {

return new JwtRequestFilter(jwtUtil, userDetailsService);

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService);

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().

.anyRequest().authenticated()

.and()

.sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

// Add the JWT filter to the security filter chain

http.addFilterBefore(jwtRequestFilter(), UsernamePasswordAuthenticationFilter.class);

}

}

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private JwtUtil jwtUtil;

@Autowired

private CustomUserDetailsService userDetailsService;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws ServletException, IOException {

final String authorizationHeader = request.getHeader("Authorization");

String username = null;

String jwtToken = null;

if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {

jwtToken = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwtToken);

}

if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

if (jwtUtil.validateToken(jwtToken, userDetails)) {

UsernamePasswordAuthenticationToken usernamePasswordAuthToken = new UsernamePasswordAuthenticationToken(

userDetails, null, userDetails.getAuthorities());

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthToken);

}

}

chain.doFilter(request, response);

}

}

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/authenticate")

public String authenticate(@RequestBody AuthenticationRequest authenticationRequest) throws Exception {

try {

Authentication authenticate = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(authenticationRequest.getUsername(), authenticationRequest.getPassword()));

if (authenticate.isAuthenticated()) {

return jwtUtil.generateToken(authenticate.getPrincipal());

} else {

throw new Exception("Invalid credentials");

}

} catch (AuthenticationException e) {

throw new Exception("Invalid credentials", e);

}

}

}

public class AuthenticationRequest {

private String username;

private String password;

// Getters and setters

}

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig implements WebMvcConfigurer {

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*")

.allowedOrigins("http://localhost:3000") // Change to your frontend URL

.allowedMethods("GET", "POST", "PUT", "DELETE")

.allowedHeaders("\*");

}

}