@RestController

public class AuthApiController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtTokenProvider jwtTokenProvider;

@PostMapping("/api/login")

public ResponseEntity<?> login(@RequestBody Map<String, String> credentials) {

String username = credentials.get("username");

String password = credentials.get("password");

try {

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

SecurityContextHolder.getContext().setAuthentication(authentication);

String token = jwtTokenProvider.generateToken(authentication);

return ResponseEntity.ok(Map.of("token", token));

} catch (AuthenticationException e) {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid username or password");

}

}

@GetMapping("/api/logout")

public ResponseEntity<String> logout(HttpServletRequest request) {

try {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

if (auth != null) {

new SecurityContextLogoutHandler().logout(request, null, auth);

}

return ResponseEntity.ok("Logout successful!");

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("Logout failed");

}

}

}

/////

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtTokenProvider {

@Value("${app.jwtSecret}")

private String jwtSecret;

@Value("${app.jwtExpirationInMs}")

private int jwtExpirationInMs;

public String generateToken(Authentication authentication) {

UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();

Date now = new Date();

Date expiryDate = new Date(now.getTime() + jwtExpirationInMs);

return Jwts.builder()

.setSubject(Long.toString(userDetails.getId()))

.setIssuedAt(new Date())

.setExpiration(expiryDate)

.signWith(SignatureAlgorithm.HS512, jwtSecret)

.compact();

}

}