@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeRequests()

.antMatchers("/api/login").permitAll()

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.permitAll();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth

.inMemoryAuthentication()

.withUser("user").password(passwordEncoder().encode("password")).roles("USER");

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

@RestController

public class AuthApiController {

@PostMapping("/api/login")

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

// Authenticate user based on provided JSON credentials

// Return appropriate response, e.g., JWT token for successful login

// For simplicity, just returning a success message

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

}

@GetMapping("/api/logout")

public ResponseEntity<String> logout() {

// Perform logout logic

// Return appropriate response, e.g., success message

// For simplicity, just returning a success message

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

}

}