DEPARTMENT OF CSE
COURSE CODE: 23SDCS12A / 23SDCS12R
FULL STACK APPLICATION DEVELOPMENT

Date of the Session: __/__/_____ **Time of The Session:_____to_____**

#LAB – 11 ➔ Implementing JWT Tokens with encryption and decryption

Prerequisites:

Basic Idea on Spring Security

Basic Idea on JWT Tokens & RBAC

Exercise:

Develop a Spring Boot application with JWT-based security for role-based authentication and authorization.

Call the JWT token generation function and validation function from browser to generate and validate the tokens.

Encrypt the provided data to the token and then where required decrypt the token to get the original data to display in the browser.

❖ Watch The Video And Do In Eclipse Workspace

11 https://youtu.be/khAaaKzWqjU?si=VnsDkjtBl_TLQJtS

JWTManager.java

```
package com.klu;

import java.security.MessageDigest;
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

import io.jsonwebtoken.*;
import io.jsonwebtoken.security.Keys;
import org.springframework.stereotype.Service;
```

```
import java.util.Base64;

@Service
public class JWTManager {

   private final SecretKey key =
Keys.hmacShaKeyFor("awdsiuchuidcidvijsuidjuiwehdcdhuichecefuerhfui".getBytes());

   public String generateToken(String username) {
      return Jwts.builder()
            .setClaims(Map.of("username", username))
            .setSubject(username)
            .setIssuedAt(new Date())
            .setExpiration(new Date(System.currentTimeMillis() + 86400000))
            .signWith(key)
            .compact();
   }

   public Map<String, String> validateToken(String token) {
      try {
         Claims claims =
Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(token).getBody();
         if (claims.getExpiration().before(new Date()))
            return Map.of("code", "404", "message", "Invalid Token");
         return Map.of("code", "200", "message", claims.get("username", String.class));
      } catch (Exception e) {
         return Map.of("code", "404", "message", "Invalid Token");
      }
   }

   public String decryptData(String encryptedData) {
      try {
         byte[] keyBytes = MessageDigest.getInstance("SHA-
256").digest("THANOS".getBytes());
         SecretKey key1 = new SecretKeySpec(keyBytes, 0, 16, "AES");
         Cipher cipher = Cipher.getInstance("AES");
```

```
 cipher.init(Cipher.DECRYPT_MODE, key1);
          return new
String(cipher.doFinal(Base64.getDecoder().decode(encryptedData)));
       } catch (Exception e) {
          return e.getMessage();
       }
    }
}
```

AppController.java

```
package com.klu;

import org.springframework.web.bind.annotation.*;
import java.util.Map;

@RestController
public class AppController {

   private final JWTManager jwt;

   public AppController(JWTManager jwt) {
      this.jwt = jwt;
   }

   @GetMapping("/login")
   public String login(@RequestParam String username) {
      return jwt.generateToken(username);
   }

   @GetMapping("/validate")
   public Map<String, String> validate(@RequestParam String token) {
      return jwt.validateToken(token);
   }
}
```

VIVA QUESTIONS:

1. What is JWT (JSON Web Token)?

JWT is a compact, URL-safe token used for **secure authentication**. It has 3 parts:

- **Header:** Algorithm & token type.
- **Payload:** User info & roles.
- **Signature:** Verifies token integrity.

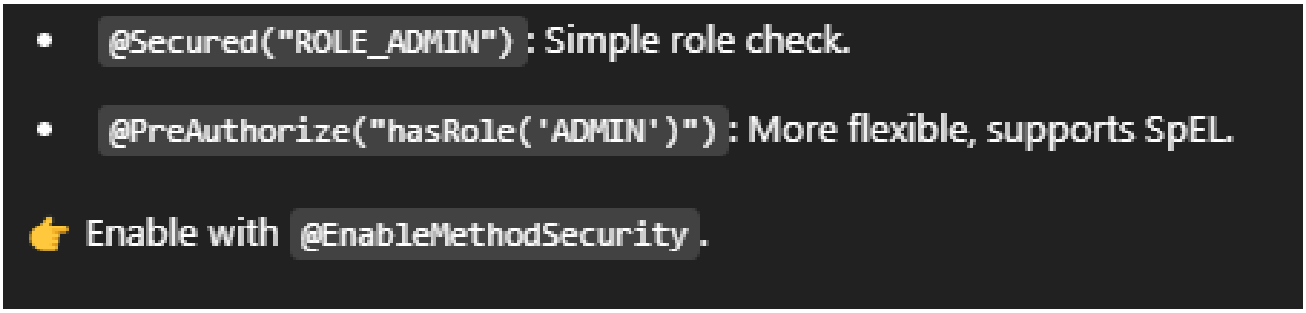
2. How does JWT facilitate secure authentication between a React frontend and a Spring Boot backend?

1. User logs in → Spring Boot generates JWT.
2. React stores JWT (localStorage/sessionStorage).
3. JWT is sent in `Authorization: Bearer <token>`.
4. Backend validates the token for protected routes.

3. How do you create and configure security filters for handling JWT tokens in Spring Boot?

- Create a `JwtAuthenticationFilter` to extract & validate the token.
- Register it in the `SecurityFilterChain` with `.addFilterBefore(...)`.
- Set session to **stateless** and secure the endpoints.

4. Discuss the use of @PreAuthorize and @Secured annotations in role-based access control.



(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured _____out of 50 Name of the Evaluator: Signature of the Evaluator Date of Evaluation: