**JWT Implementation**

pom.xml

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.6.0 </version>

</dependency>

**Step1: JWT token generation.**

**public** **static** String generateToken(String username, String password) {

String jwtToken = **null**;

String *SECRET\_KEY = “my-jwt-secrect-token”;*

JwtBuilder builder = Jwts.*builder*().setSubject(String.*valueOf*(username)).setIssuer(*SECRET\_KEY*)

.claim("password", password)

.signWith(SignatureAlgorithm.***HS256***, *SECRET\_KEY*);

jwtToken = builder.compact();

//setIssuedAt(new Date())

**return** jwtToken;

}

***Step2: Once JWT Token generated include this jwt token either in headers or cahe storage or hazzle cast.***

***Step3: In the target Micro Service validate the jwt token as mentioned below.***

**public** **static** **boolean** verifyToken(String jwtToken){

**boolean** verificationFlag = **false**;

Jws<Claims> claims;

**try** {

claims = Jwts.*parser*().setSigningKey(*SECRET\_KEY*).parseClaimsJws(jwtToken);

String subject = claims.getBody().getSubject();

System.***out***.println("subject>>>>"+subject);

**if** (subject.matches("testUser")) {

System.***out***.println("User Authenticated");

verificationFlag = **true**;

} **else** {

System.***out***.println("User not Authenticated.");

}

} **catch** (ExpiredJwtException | UnsupportedJwtException

| MalformedJwtException | SignatureException

| IllegalArgumentException e) {

verificationFlag = **false**;

e.printStackTrace();

}

**return** verificationFlag;

}

***Step4: successful JWT Validation the above method returns true value.***

***So that we can provide the access to target resource.***