# Question 1 - Create Authentication Controller and Configure Security

Code 1: AuthenticationController.java

package com.cognizant.springlearn.controller;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
public class AuthenticationController {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 LOGGER.info("Start");  
 LOGGER.debug("Authorization Header: {}", authHeader);  
 Map<String, String> map = new HashMap<>();  
 map.put("token", "");  
 LOGGER.info("End");  
 return map;  
 }  
}

Code 2: SecurityConfig.java (configuration snippet)

@Override  
protected void configure(HttpSecurity httpSecurity) throws Exception {  
 httpSecurity.csrf().disable().httpBasic().and()  
 .authorizeRequests()  
 .antMatchers("/countries").hasRole("USER")  
 .antMatchers("/authenticate").hasAnyRole("USER", "ADMIN");  
}

Output (Sample - curl):  
curl -s -u user:pwd http://localhost:8090/authenticate  
{"token":""}

# Question 2 - Read Authorization Header and Decode Username

Code 1: AuthenticationController.java (getUser method)

private String getUser(String authHeader) {  
 String encodedCredentials = authHeader.substring(6);  
 byte[] decodedBytes = java.util.Base64.getDecoder().decode(encodedCredentials);  
 String decoded = new String(decodedBytes);  
 return decoded.split(":")[0];  
}

Code 2: Invoke getUser() from authenticate()

@GetMapping("/authenticate")  
public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 LOGGER.info("Start");  
 String user = getUser(authHeader);  
 LOGGER.debug("User: {}", user);  
 Map<String, String> map = new HashMap<>();  
 map.put("token", "");  
 LOGGER.info("End");  
 return map;  
}

Output:  
User value printed in logs.

# Question 3 - Generate JWT Token

Code 1: generateJwt method

private String generateJwt(String user) {  
 io.jsonwebtoken.JwtBuilder builder = io.jsonwebtoken.Jwts.builder();  
 builder.setSubject(user);  
 builder.setIssuedAt(new java.util.Date());  
 builder.setExpiration(new java.util.Date((new java.util.Date()).getTime() + 1200000));  
 builder.signWith(io.jsonwebtoken.SignatureAlgorithm.HS256, "secretkey");  
 return builder.compact();  
}

Code 2: Modify authenticate() to use generateJwt()

@GetMapping("/authenticate")  
public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 LOGGER.info("Start");  
 String user = getUser(authHeader);  
 String token = generateJwt(user);  
 Map<String, String> map = new HashMap<>();  
 map.put("token", token);  
 LOGGER.info("End");  
 return map;  
}

Output (Sample):  
curl -s -u user:pwd http://localhost:8090/authenticate  
{"token":"<JWT\_TOKEN\_STRING>"}

# Question 4 - Authorize Based on JWT

Code 1: JwtAuthorizationFilter.java

package com.cognizant.springlearn.security;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import javax.servlet.FilterChain;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;  
import io.jsonwebtoken.Claims;  
import io.jsonwebtoken.Jws;  
import io.jsonwebtoken.JwtException;  
import io.jsonwebtoken.Jwts;  
  
public class JwtAuthorizationFilter extends BasicAuthenticationFilter {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(JwtAuthorizationFilter.class);  
  
 public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {  
 super(authenticationManager);  
 LOGGER.info("Start");  
 }  
  
 @Override  
 protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,  
 FilterChain chain) throws IOException, ServletException {  
 LOGGER.info("Start");  
 String header = req.getHeader("Authorization");  
 if (header == null || !header.startsWith("Bearer ")) {  
 chain.doFilter(req, res);  
 return;  
 }  
 UsernamePasswordAuthenticationToken authentication = getAuthentication(req);  
 SecurityContextHolder.getContext().setAuthentication(authentication);  
 chain.doFilter(req, res);  
 LOGGER.info("End");  
 }  
  
 private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {  
 String token = request.getHeader("Authorization");  
 if (token != null) {  
 try {  
 Jws<Claims> jws = Jwts.parser()  
 .setSigningKey("secretkey")  
 .parseClaimsJws(token.replace("Bearer ", ""));  
 String user = jws.getBody().getSubject();  
 if (user != null) {  
 return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());  
 }  
 } catch (JwtException ex) {  
 return null;  
 }  
 }  
 return null;  
 }  
}

Code 2: SecurityConfig.java modification

@Override  
protected void configure(HttpSecurity httpSecurity) throws Exception {  
 httpSecurity.csrf().disable().httpBasic().and()  
 .authorizeRequests()  
 .antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")  
 .anyRequest().authenticated()  
 .and()  
 .addFilter(new JwtAuthorizationFilter(authenticationManager()));  
}

Output:  
1. Get token:  
curl -s -u user:pwd http://localhost:8090/authenticate  
  
2. Access countries with token:  
curl -s -H "Authorization: Bearer <JWT\_TOKEN>" http://localhost:8090/countries