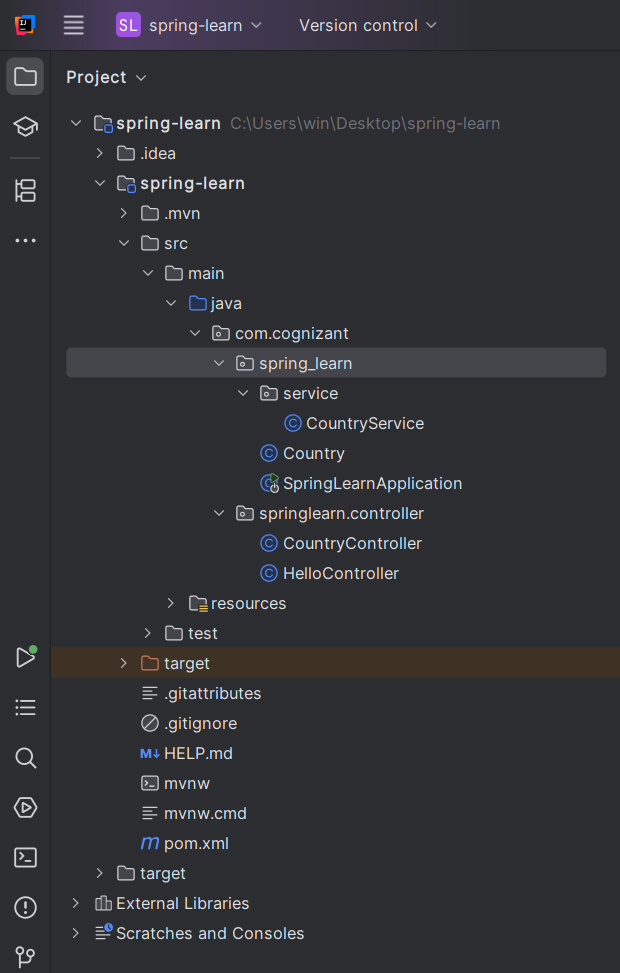
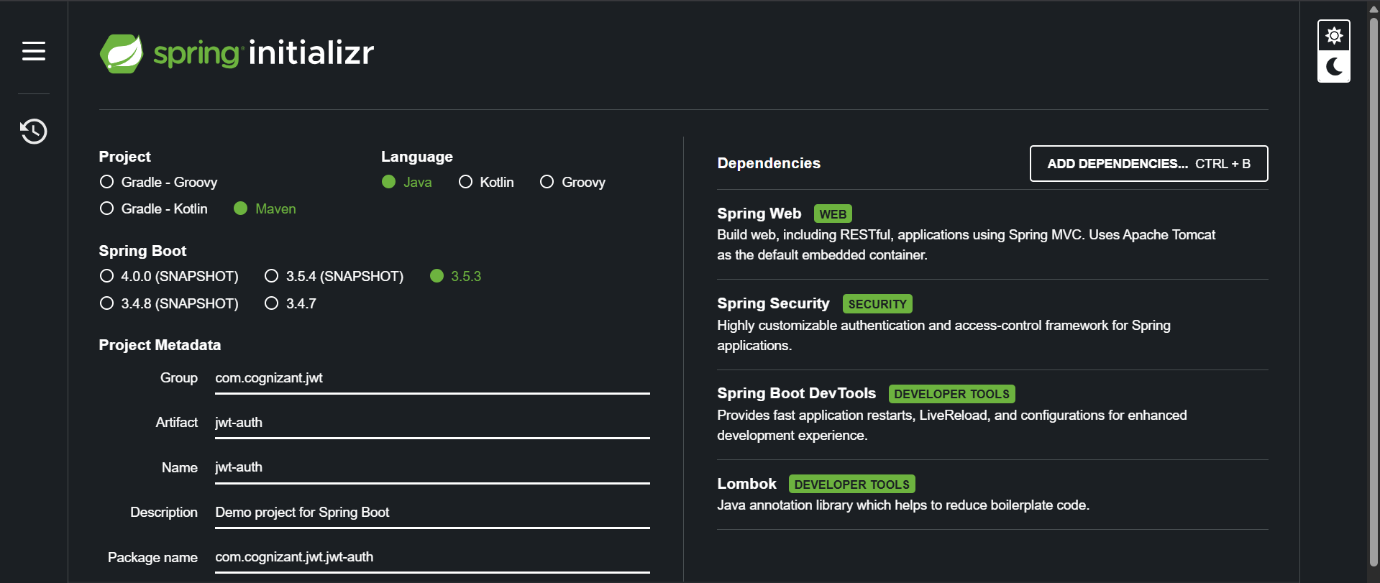
**Create authentication service that returns JWT**

**File Hierarchy**

****

STEP 1: Setup Spring Boot Project



STEP 2: Add JWT Dependencies in pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant.jwt</groupId>  
 <artifactId>jwt-auth</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>jwt-auth</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.1</version>  
 </dependency>  
  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <annotationProcessorPaths>  
 <path>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </path>  
 </annotationProcessorPaths>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <exclude>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

STEP 3: Create JwtUtil.java

Path: com.cognizant.jwt.util.JwtUtil

package com.cognizant.jwt.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
  
@Component  
public class JwtUtil {  
  
 private String secret = "my-secret-key"; // Keep this safe  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour  
 .signWith(SignatureAlgorithm.*HS256*, secret)  
 .compact();  
 }  
}

## STEP 4: Create AuthController.java

**Path**: com.cognizant.jwt.controller.AuthController

package com.cognizant.jwt.controller;  
  
import com.cognizant.jwt.util.JwtUtil;  
import jakarta.servlet.http.HttpServletRequest;  
import org.apache.tomcat.util.codec.binary.Base64;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.nio.charset.StandardCharsets;  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @RequestMapping(value = "/authenticate", method = RequestMethod.*GET*)  
 public Map<String, String> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 throw new RuntimeException("Missing or invalid Authorization header");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length());  
 byte[] credDecoded = Base64.*decodeBase64*(base64Credentials);  
 String credentials = new String(credDecoded, StandardCharsets.*UTF\_8*);  
 final String[] values = credentials.split(":", 2);  
  
 String username = values[0];  
 String password = values[1];  
  
 // Hardcoded user for demo  
 if (!username.equals("user") || !password.equals("pwd")) {  
 throw new RuntimeException("Invalid credentials");  
 }  
  
 String token = jwtUtil.generateToken(username);  
  
 Map<String, String> response = new HashMap<>();  
 response.put("token", token);  
 return response;  
 }  
}

STEP 5: Configure Port and Disable CSRF

application.properties:

server.port=8090

## STEP 6: Create SecurityConfig.java

**Path**: com.cognizant.jwt.config.SecurityConfig

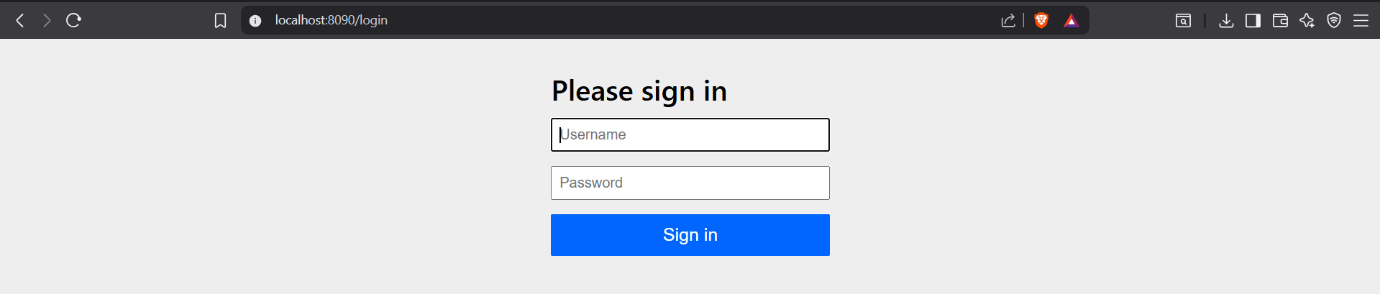
package com.cognizant.jwt.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.configuration.\*;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 );  
 return http.build();  
 }  
}

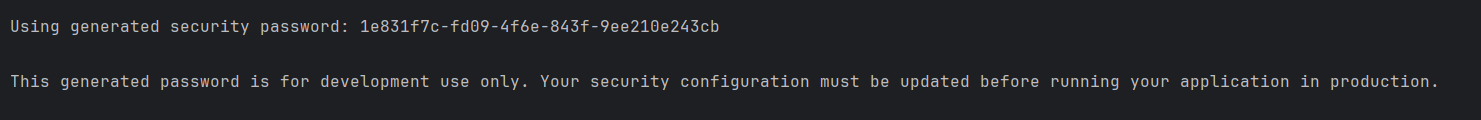
## STEP 7: Run the App

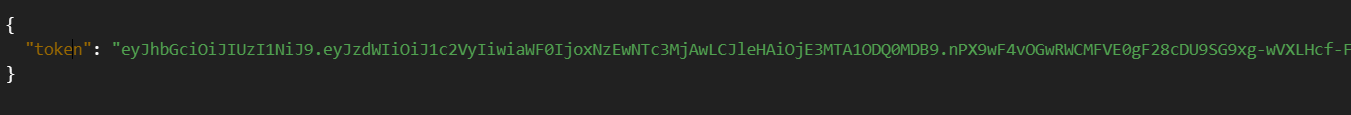
**Run JwtAuthApplication.java or use:**

**Visit: http://localhost:8090/authenticate**

A popup will ask for username/password (enter user / pwd)





****