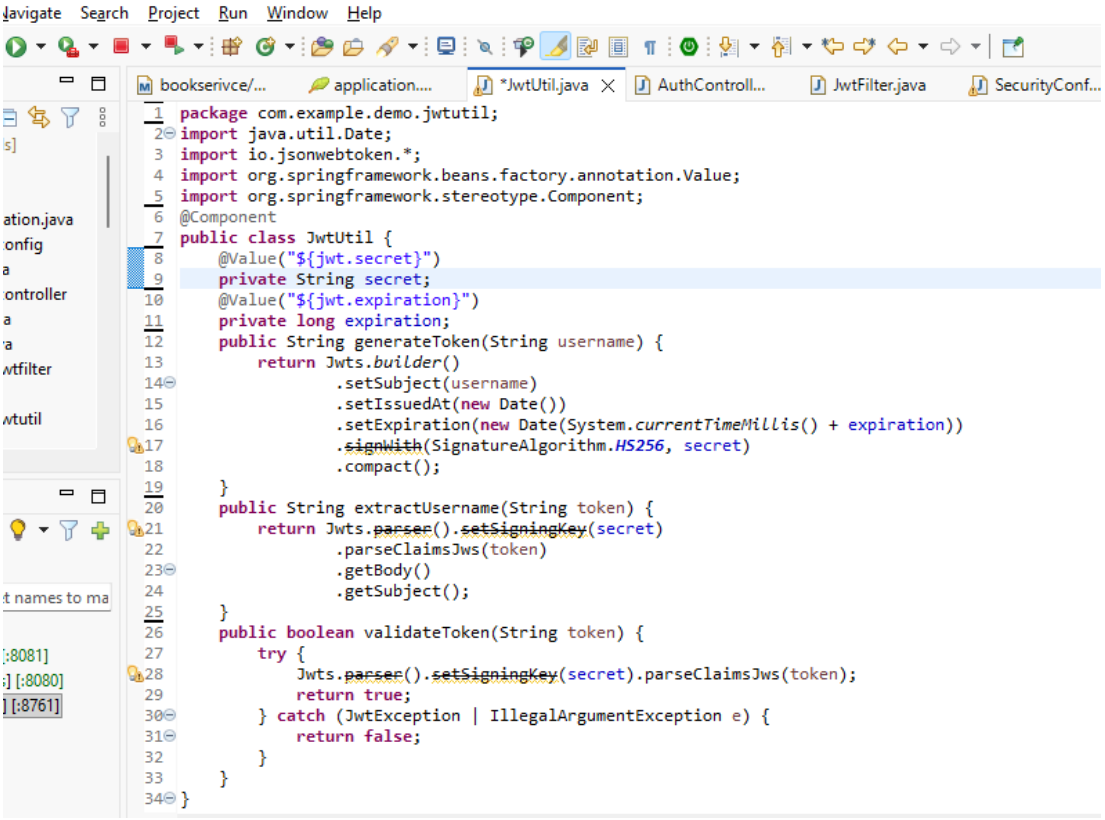
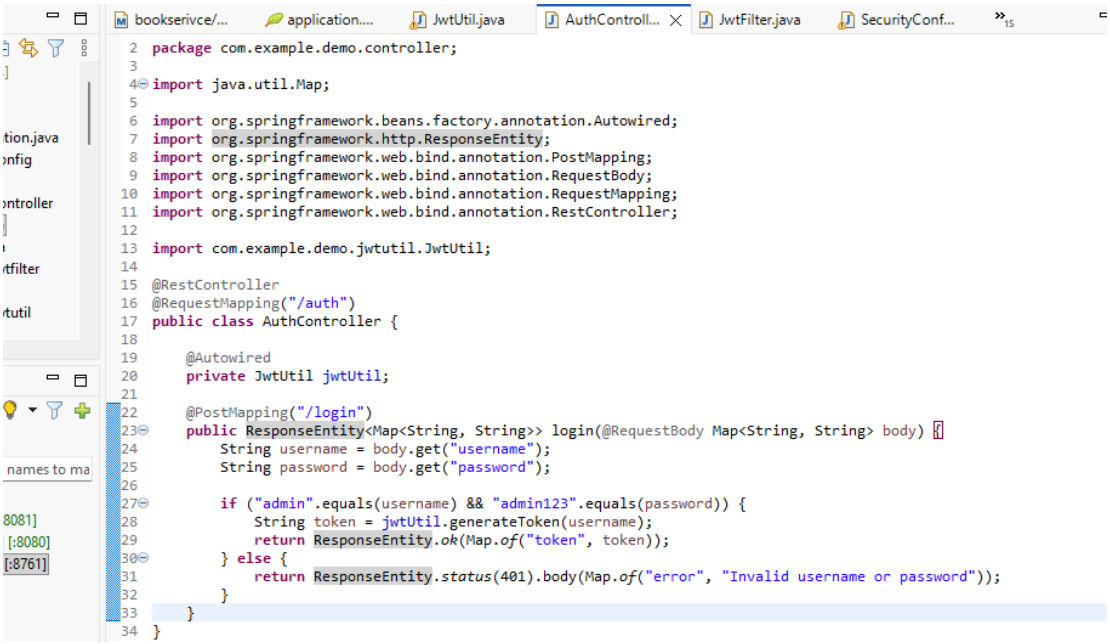
Daily task : 10/10/2025

Jwtutil.java



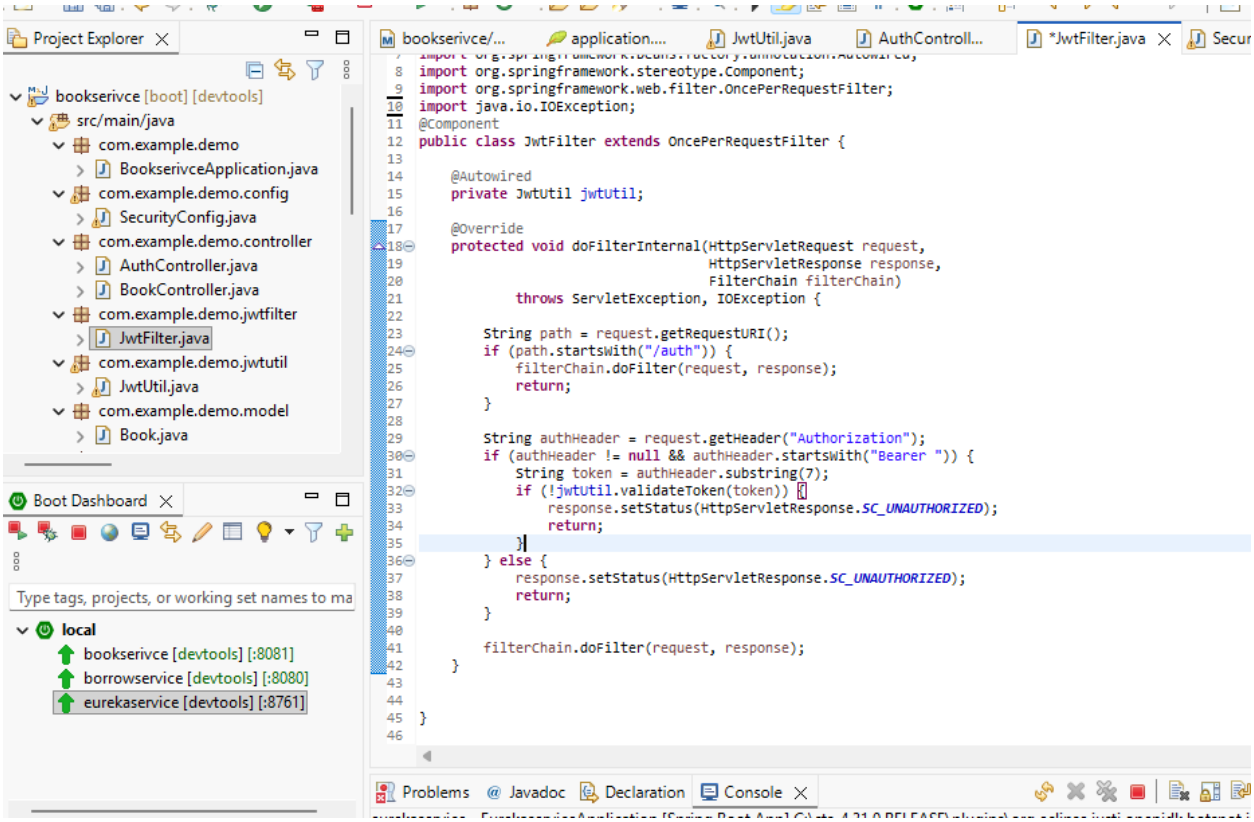
```java
package com.example.demo.jwtutil;
import java.util.Date;
import io.jsonwebtoken.*;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
@Component
public class JwtUtil {
    @Value("${jwt.secret}")
    private String secret;
    @Value("${jwt.expiration}")
    private long expiration;
    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + expiration))
                .signWith(SignatureAlgorithm.HS256, secret)
                .compact();
    }
    public String extractUsername(String token) {
        return Jwts.parser().setSigningKey(secret)
                .parseClaimsJws(token)
                .getBody()
                .getSubject();
    }
    public boolean validateToken(String token) {
        try {
            Jwts.parser().setSigningKey(secret).parseClaimsJws(token);
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            return false;
        }
    }
}
```

AuthController.java

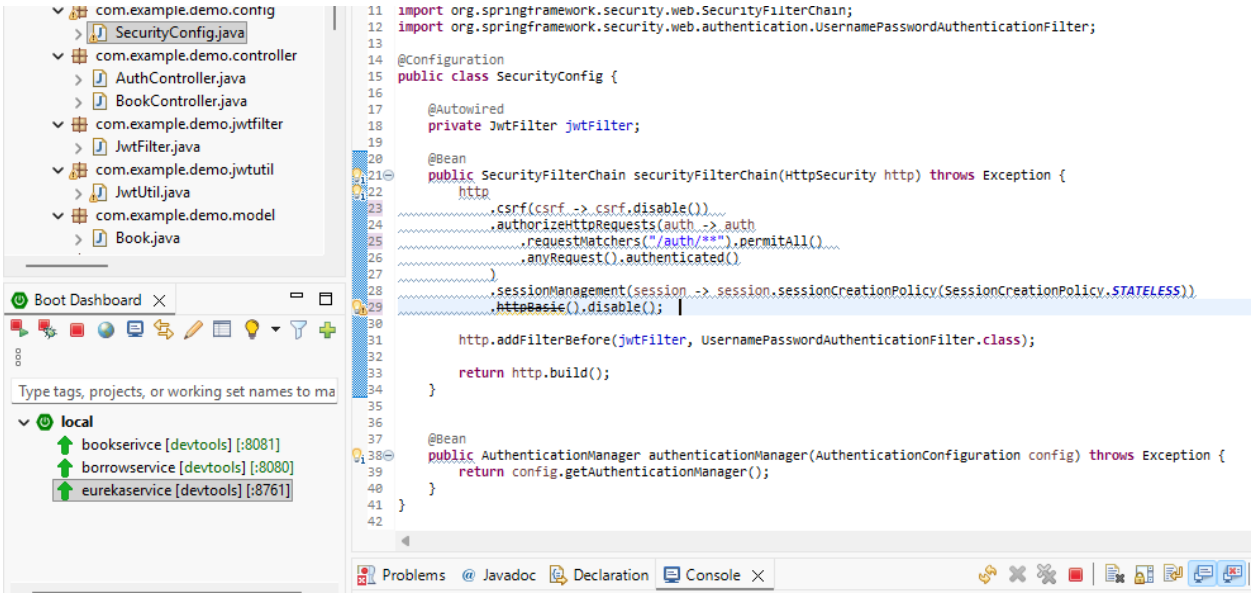


```java
package com.example.demo.controller;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.jwtutil.JwtUtil;

@RestController
@RequestMapping("/auth")
public class AuthController {

    @Autowired
    private JwtUtil jwtUtil;

    @PostMapping("/login")
    public ResponseEntity<Map<String, String>> login(@RequestBody Map<String, String> body) {
        String username = body.get("username");
        String password = body.get("password");

        if ("admin".equals(username) && "admin123".equals(password)) {
            String token = jwtUtil.generateToken(username);
            return ResponseEntity.ok(Map.of("token", token));
        } else {
            return ResponseEntity.status(401).body(Map.of("error", "Invalid username or password"));
        }
    }
}
```

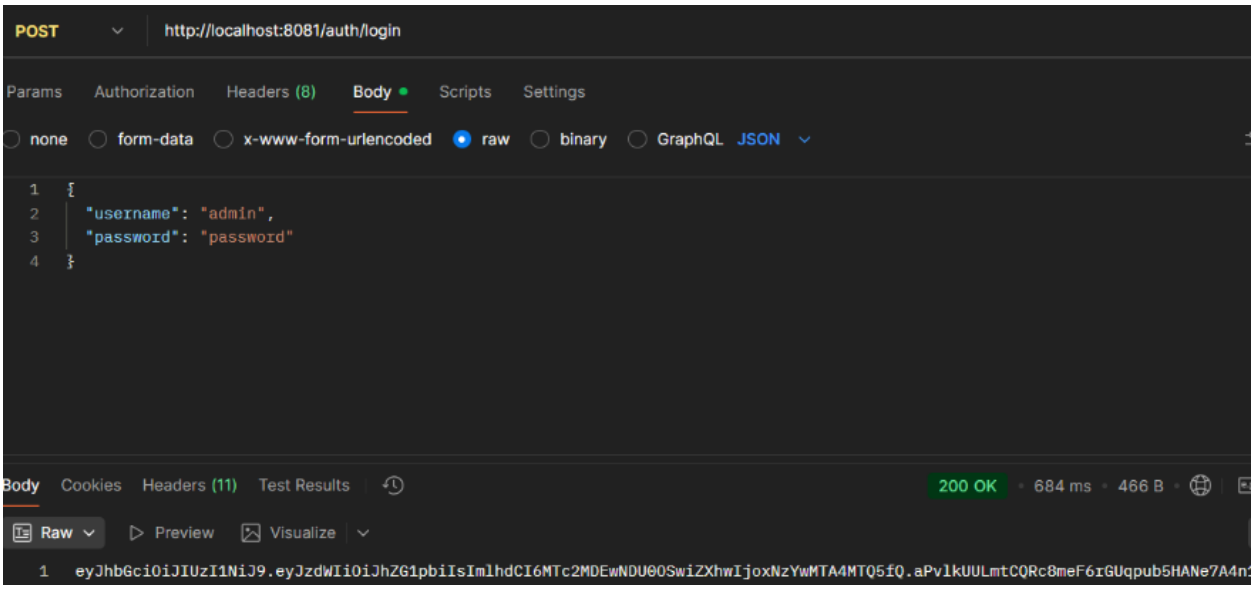
JwtFilter.java

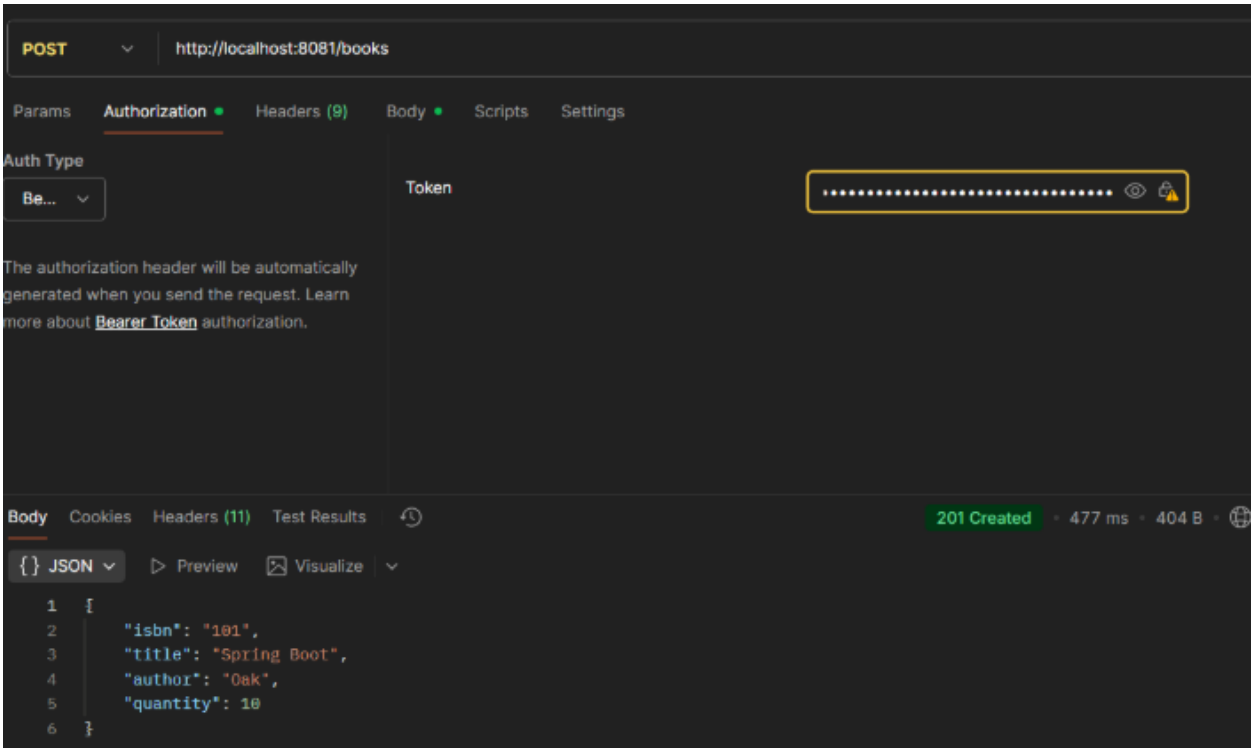


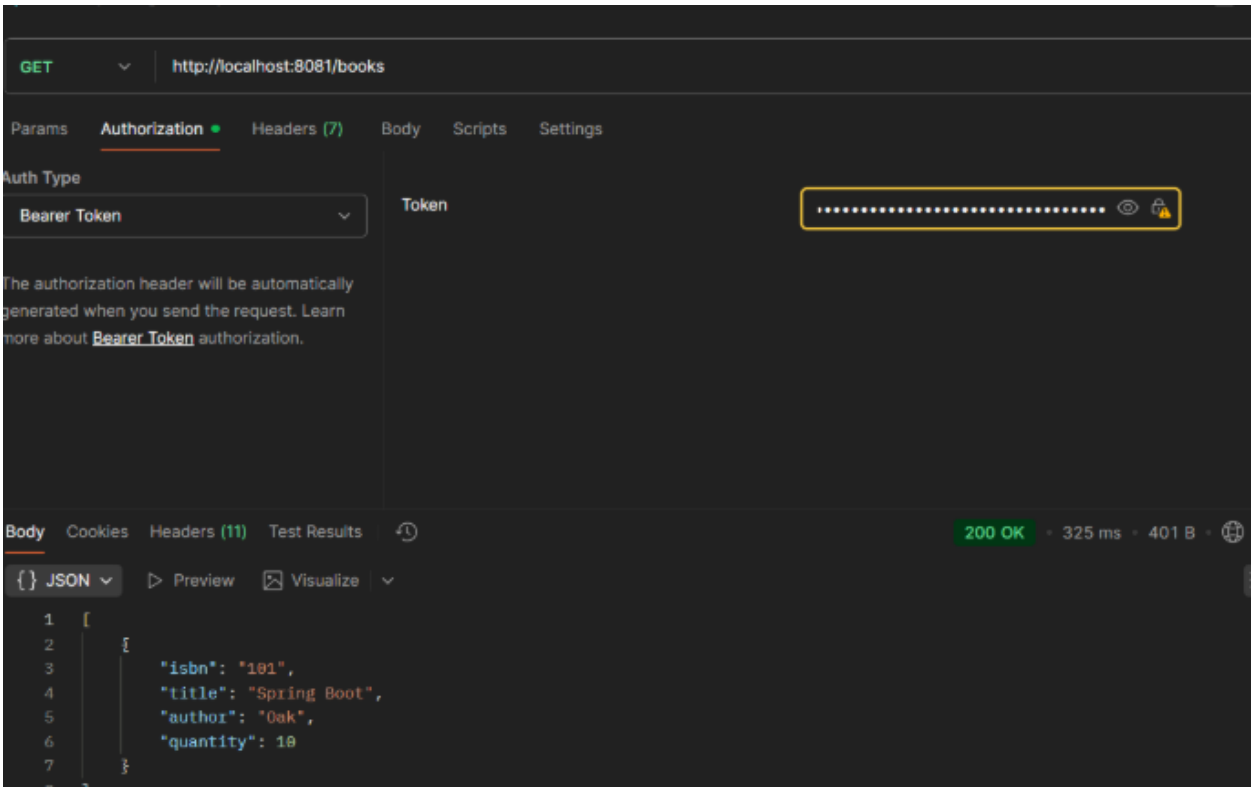
SecurityConfig.java



Postman


POST
http://localhost:8081/auth/login
Params
Authorization
Headers (8)
Body
Scripts
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"username": "admin",
"password": "password"
}
Body
Cookies
Headers (11)
Test Results
200 OK
684 ms
466 B
Raw
Preview
Visualize
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc2MDEwNDU0OSwiZXhwIjoxNzYwMTA4MTQ5fQ.aPvlkUULmtCQRc8meF6rGUqpub5HANe7A4n


POST
http://localhost:8081/books
Params
Authorization
Headers (9)
Body
Scripts
Settings
Auth Type
Be...
Token
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Body
Cookies
Headers (11)
Test Results
201 Created
477 ms
404 B
{} JSON
Preview
Visualize
{
"isbn": "101",
"title": "Spring Boot",
"author": "Oak",
"quantity": 10
}


GET
http://localhost:8081/books
Params
Authorization
Headers (7)
Body
Scripts
Settings
Auth Type
Bearer Token
Token
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Body
Cookies
Headers (11)
Test Results
200 OK
325 ms
401 B
{} JSON
Preview
Visualize
[
{
"isbn": "101",
"title": "Spring Boot",
"author": "Oak",
"quantity": 10
}