Spring Security Assignment

Q1.

<?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>2.5.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.springsecurity</groupId>

<artifactId>SecurityApp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SecurityApp</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>16</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-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**package** com.springsecurity.SecurityApp;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** HomeResource {

@GetMapping("/")

**public** String home() {

**return** ("<h1>Welcome To Spring Security Hello World Project by Riddhi</h1>");

}

}

**package** com.springsecurity.SecurityApp;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

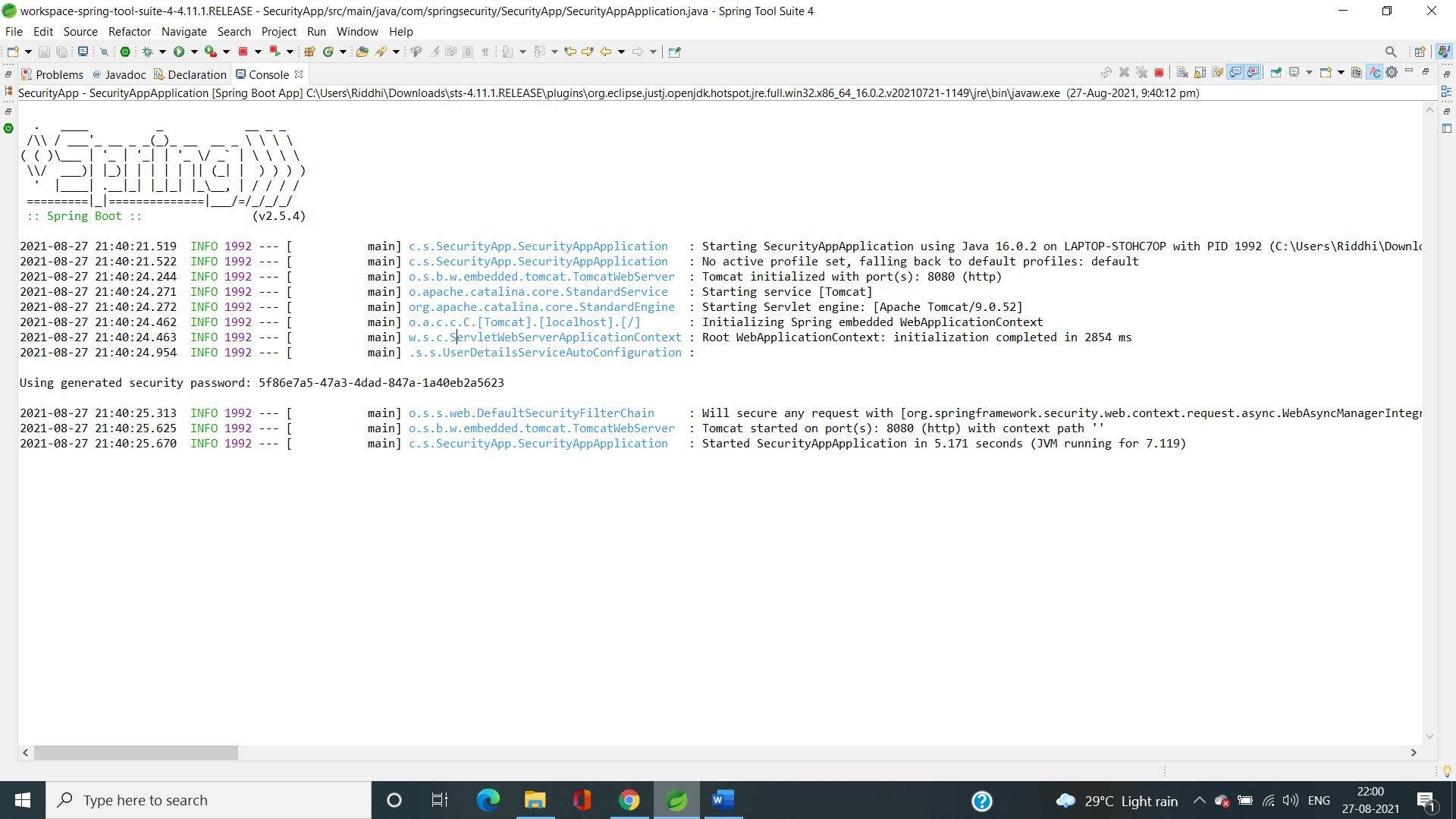
**public** **class** SecurityAppApplication {

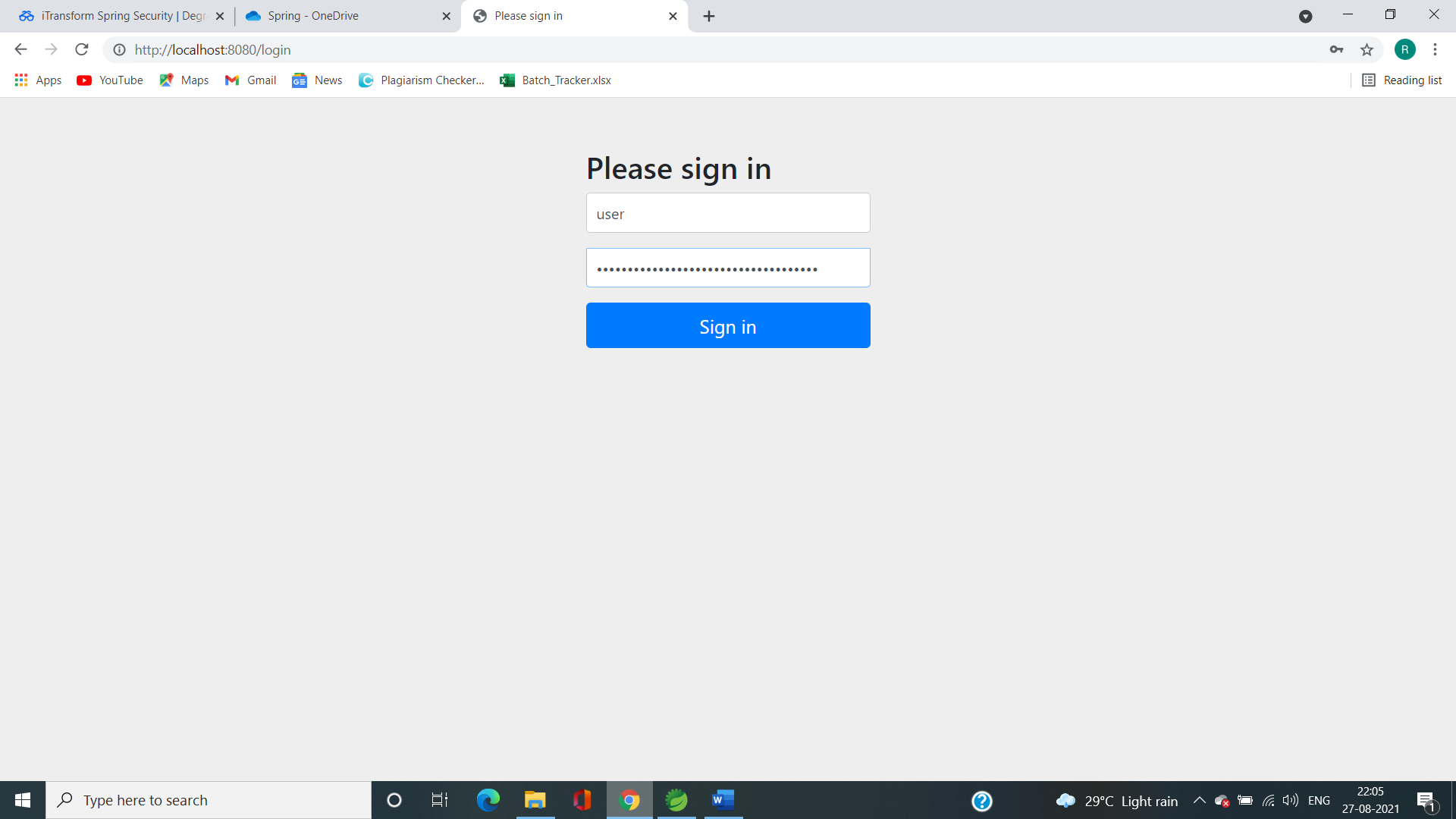
**public** **static** **void** main(String[] args) {

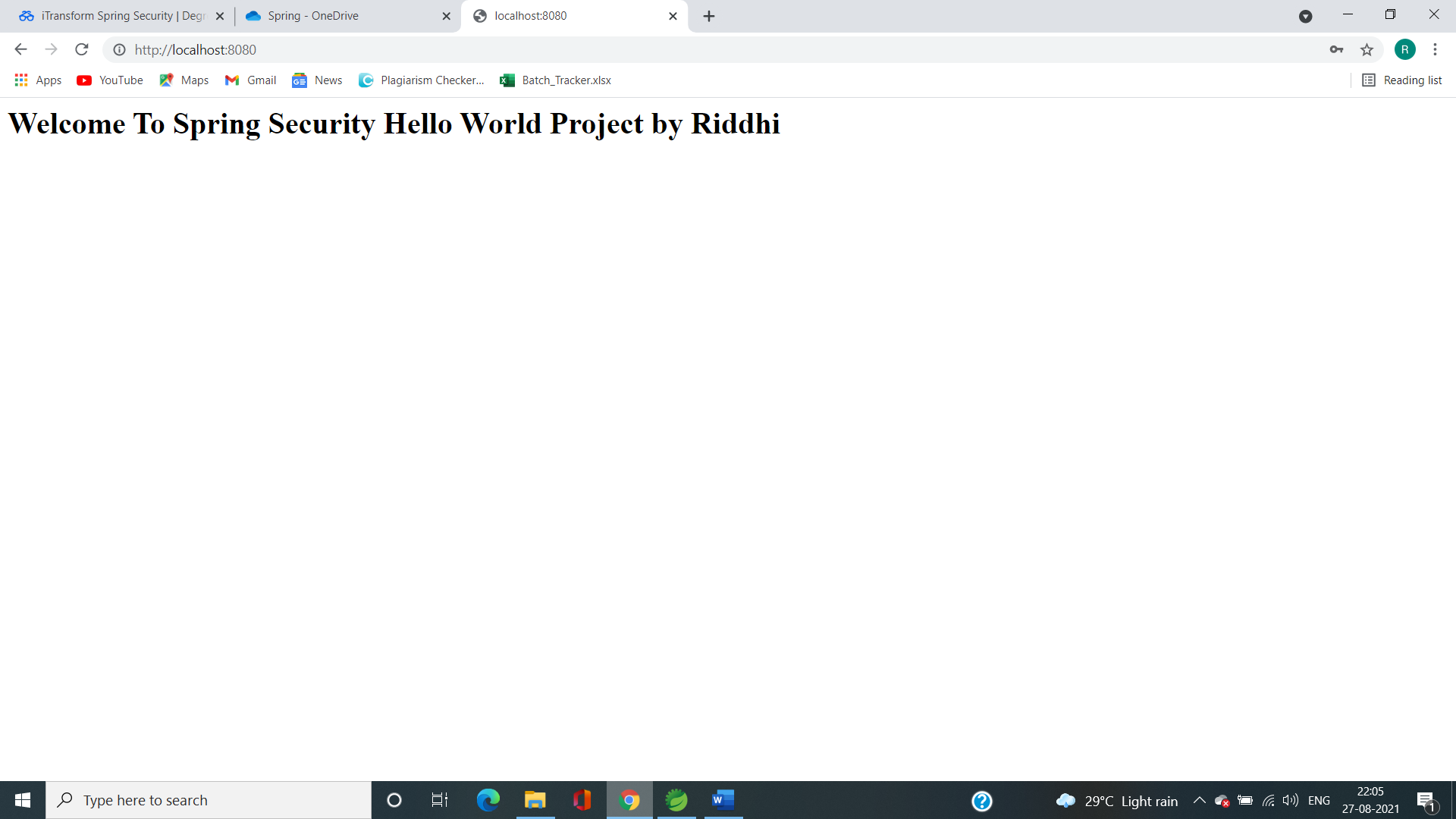
SpringApplication.*run*(SecurityAppApplication.**class**, args);

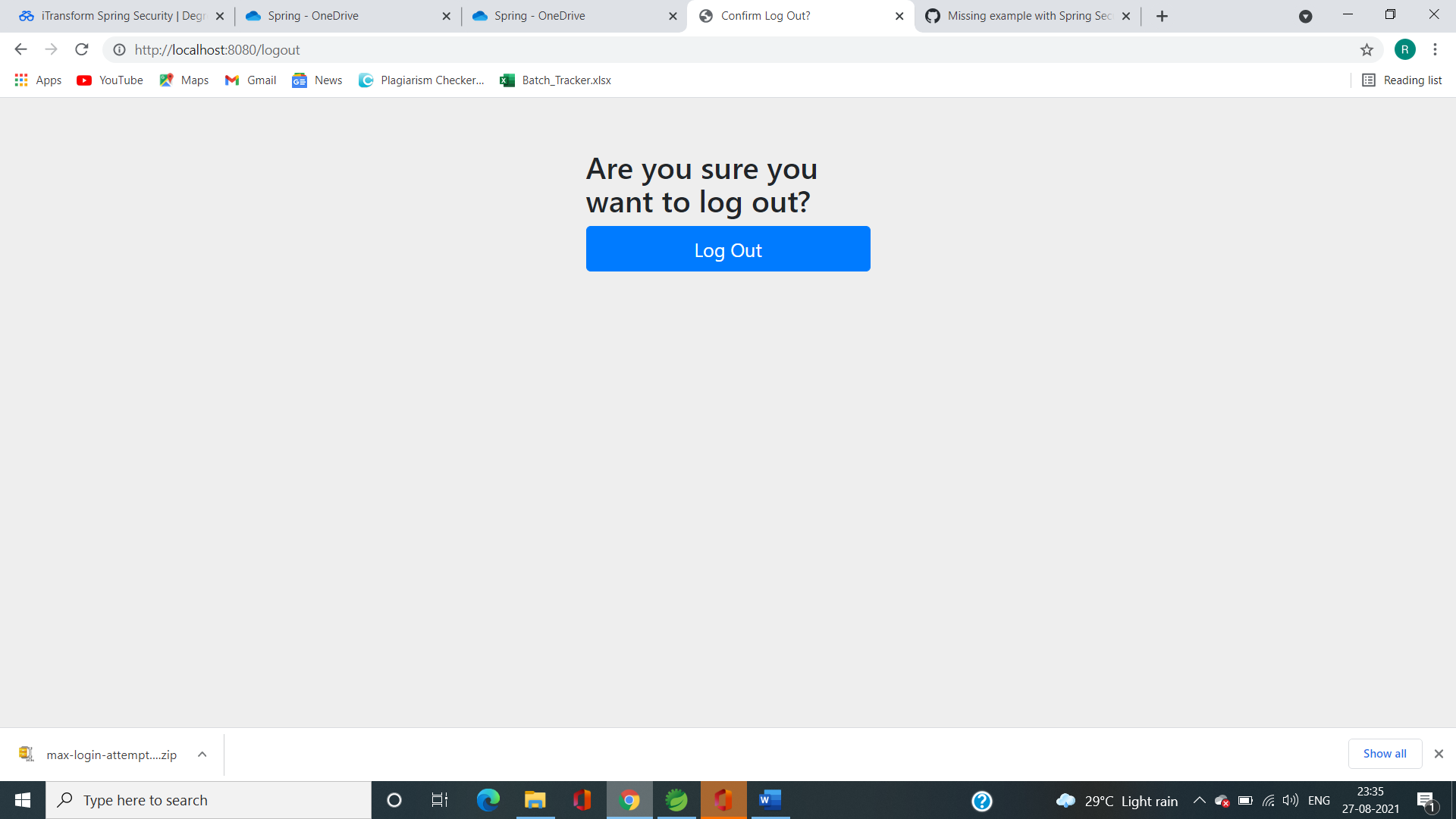
}

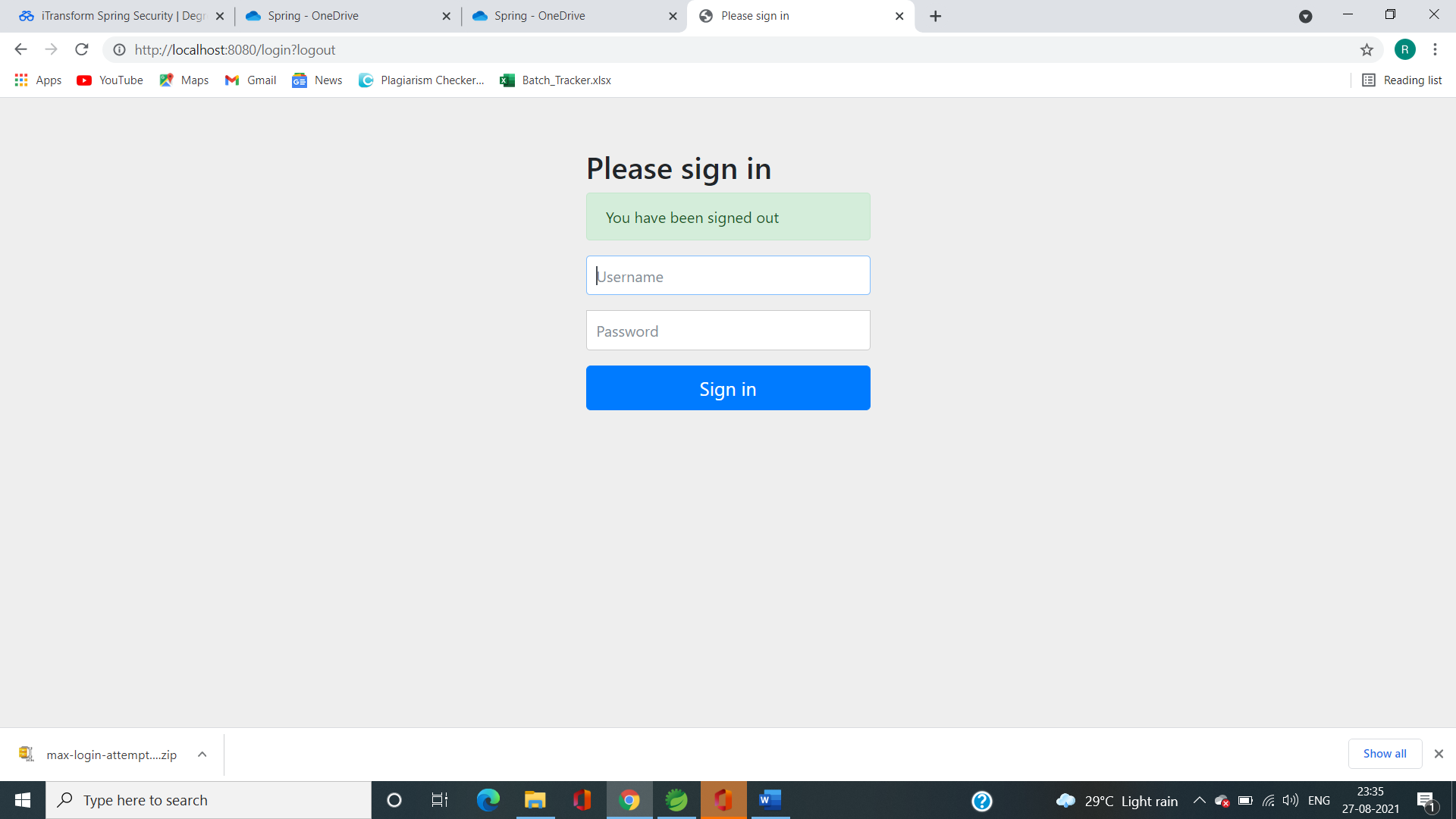
}











Q2. Hello World Program

**package** com.springsecurity.SecurityApp;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** HomeResource {

@GetMapping("/")

**public** String home() {

**return** ("<h1>Welcome To Spring Security Hello World Project by Riddhi</h1>");

}

@GetMapping("/users")

**public** String user() {

**return** ("<h1>Welcome User To User Manager </h1>");

}

@GetMapping("/admin")

**public** String admin() {

**return** ("<h1> Welcome Admin Riddhi To Hello World </h1>");

}

}

**package** com.springsecurity.SecurityApp;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SecurityAppApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SecurityAppApplication.**class**, args);

}

}

**package** com.springsecurity.SecurityApp;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfiguration;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.crypto.password.~~NoOpPasswordEncoder~~;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter{

//this method is for authentication

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

//set your configuration to auth object of AMB

auth.inMemoryAuthentication()

.withUser("Riddhi")

.password("12345")

.roles("USER")

.and()

.withUser("admin")

.password("3030303")

.roles("ADMIN");

}

@Bean

**public** PasswordEncoder getPasswordEncoder() {

**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

//This override method is for Authorization of certain roles

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.authorizeRequests()

.antMatchers("/admin").hasRole("ADMIN") //Allows only admin

.antMatchers("/users").hasAnyRole("USER", "ADMIN") //Allows Both user and admin

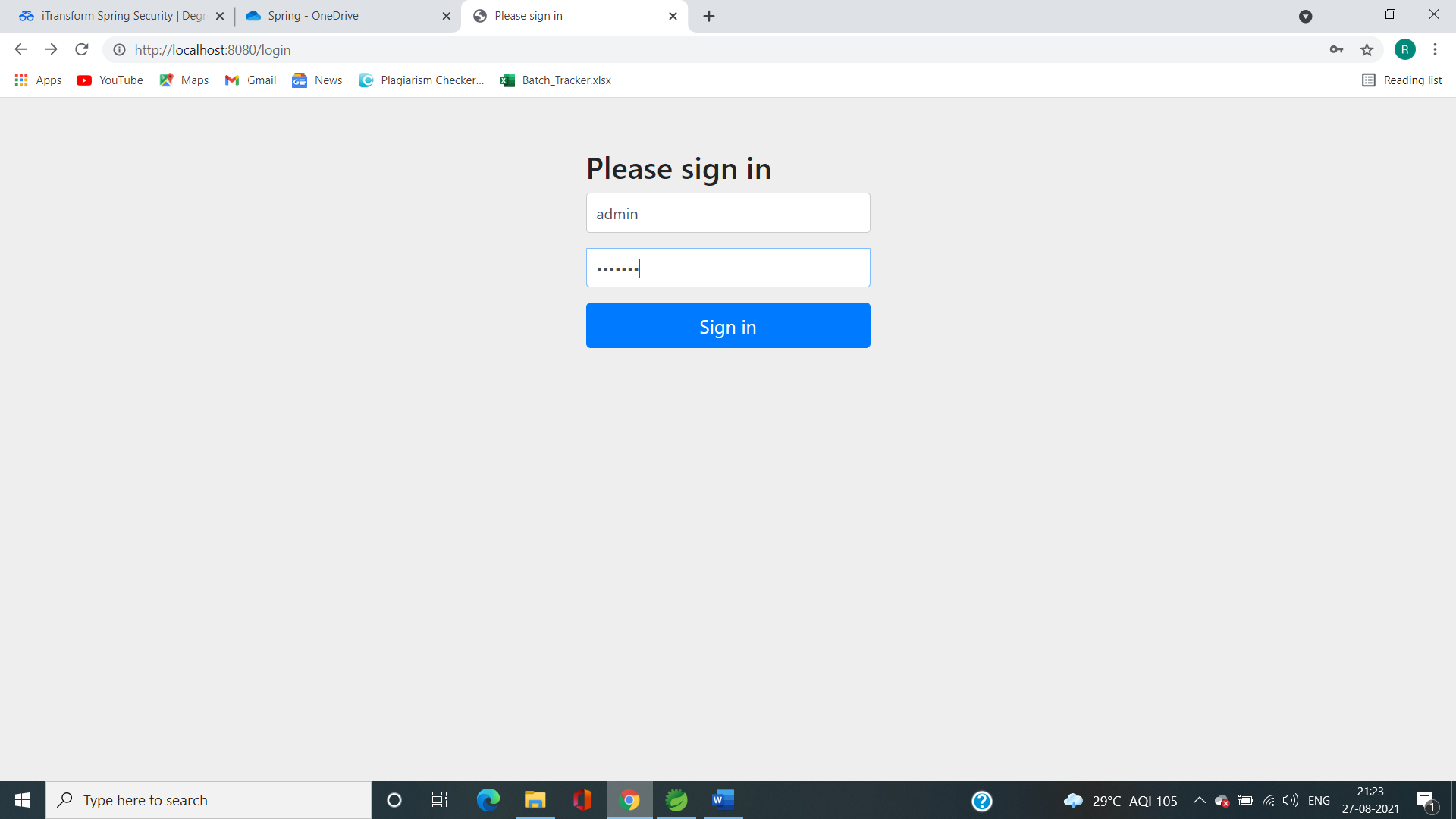
.antMatchers("/").permitAll() //Allows Everyone without login also

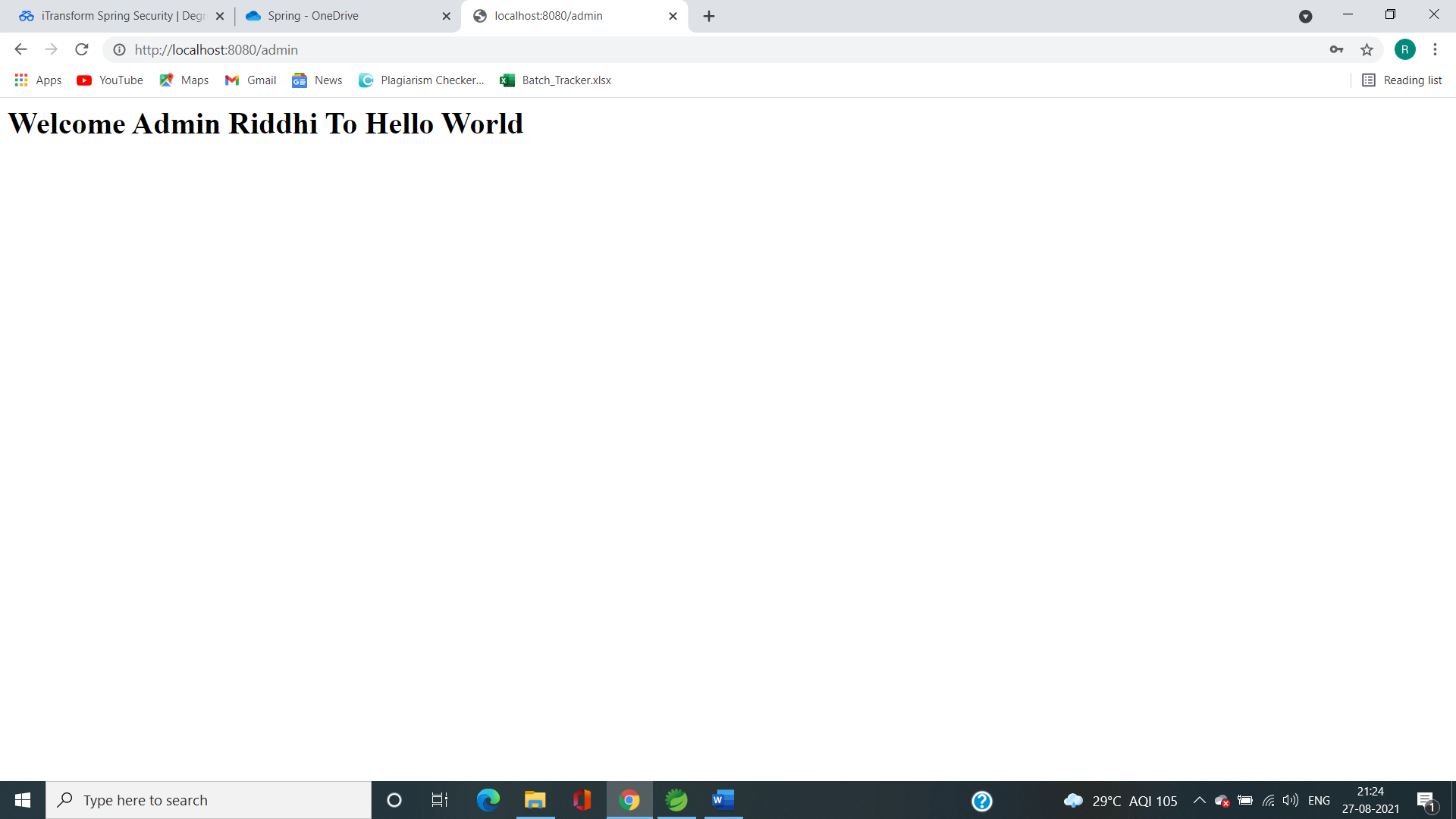
.and().formLogin();

}

}

Output:

****



Q3.

**package** io.javabrains.springsecurityjdbc;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** HomeResource {

@GetMapping("/")

**public** String home() {

**return** ("<h1>Welcome</h1>");

}

@GetMapping("/user")

**public** String user() {

**return** ("<h1>Welcome User</h1>");

}

@GetMapping("/admin")

**public** String admin() {

**return** ("<h1>Welcome Admin</h1>");

}

}

**package** io.javabrains.springsecurityjdbc;

**import** javax.sql.DataSource;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.crypto.password.~~NoOpPasswordEncoder~~;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter{

@Autowired

DataSource dataSource;

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception{

auth.jdbcAuthentication()

.dataSource(dataSource);

/\*.withDefaultSchema() // this is for default schema

.withUser(

User.withUsername("user")

.password("pass")

.roles("USER")

)

.withUser(

User.withUsername("admin")

.password("pass")

.roles("ADMIN")

);

\*/

}

@Bean

**public** PasswordEncoder getPasswordEncoder() {

**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

**protected** **void** configure(HttpSecurity http) **throws** Exception{

http.authorizeRequests()

.antMatchers("/admin").hasRole("ADMIN")

.antMatchers("/user").hasAnyRole("USER", "ADMIN")

.antMatchers("/").permitAll()

.and()

//.rememberMe().userDetailsService(this.userDetailsService())

.formLogin();

}

}

**package** io.javabrains.springsecurityjdbc;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringSecurityJdbcApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringSecurityJdbcApplication.**class**, args);

}

}

Sql files

Schema.sql

create table users(

username varchar\_ignorecase(50) not null primary key,

password varchar\_ignorecase(50) not null,

enabled boolean not null

);

create table authorities (

username varchar\_ignorecase(50) not null,

authority varchar\_ignorecase(50) not null,

constraint fk\_authorities\_users foreign key(username) references users(username)

);

create unique index ix\_auth\_username on authorities (username,authority);

data.sql file

INSERT INTO users(username, password, enabled)

values('user', 'pass', 'true');

INSERT INTO users(username, password, enabled)

values('admin', 'pass', 'true');

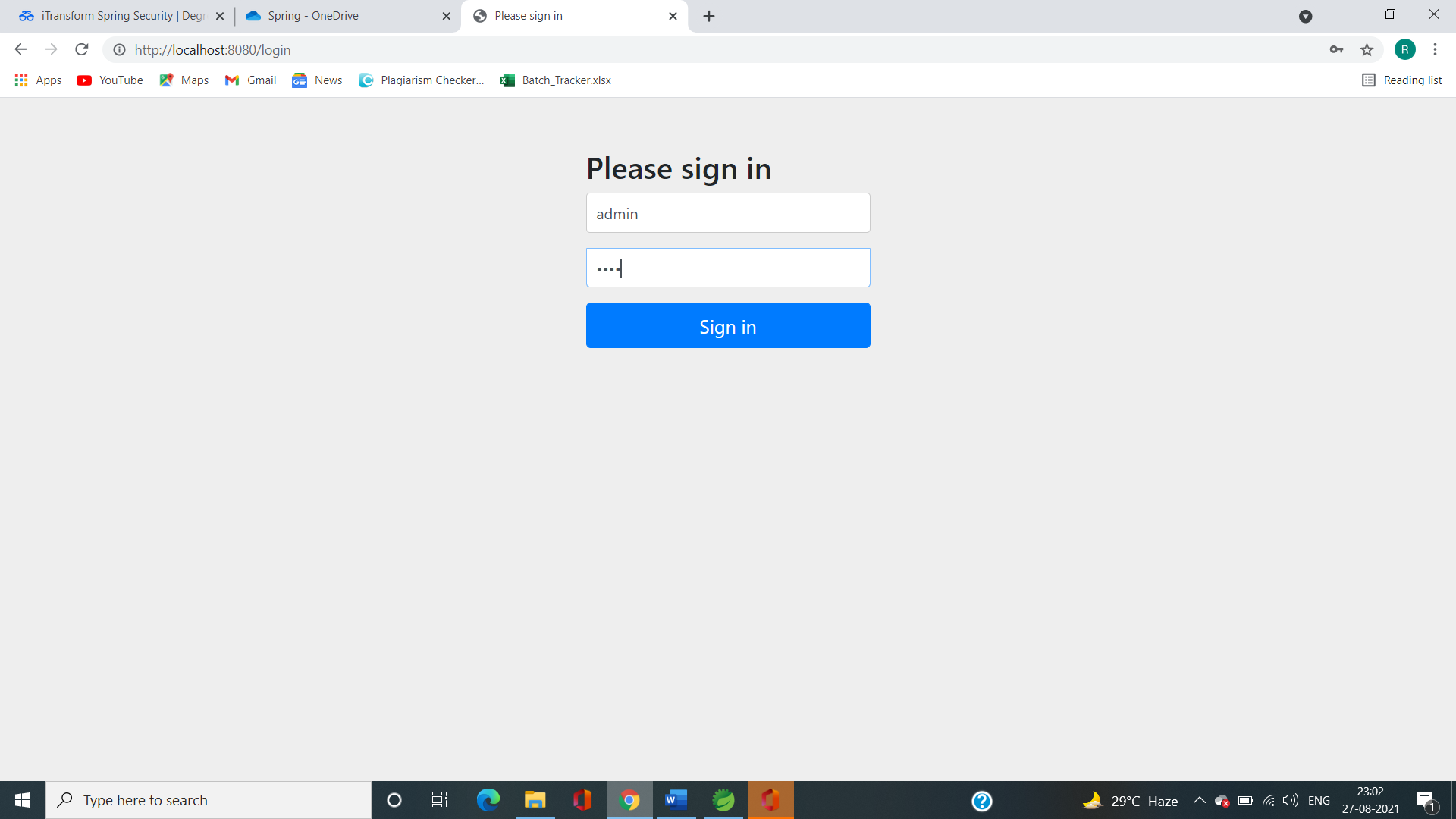
INSERT INTO authorities(username, authority)

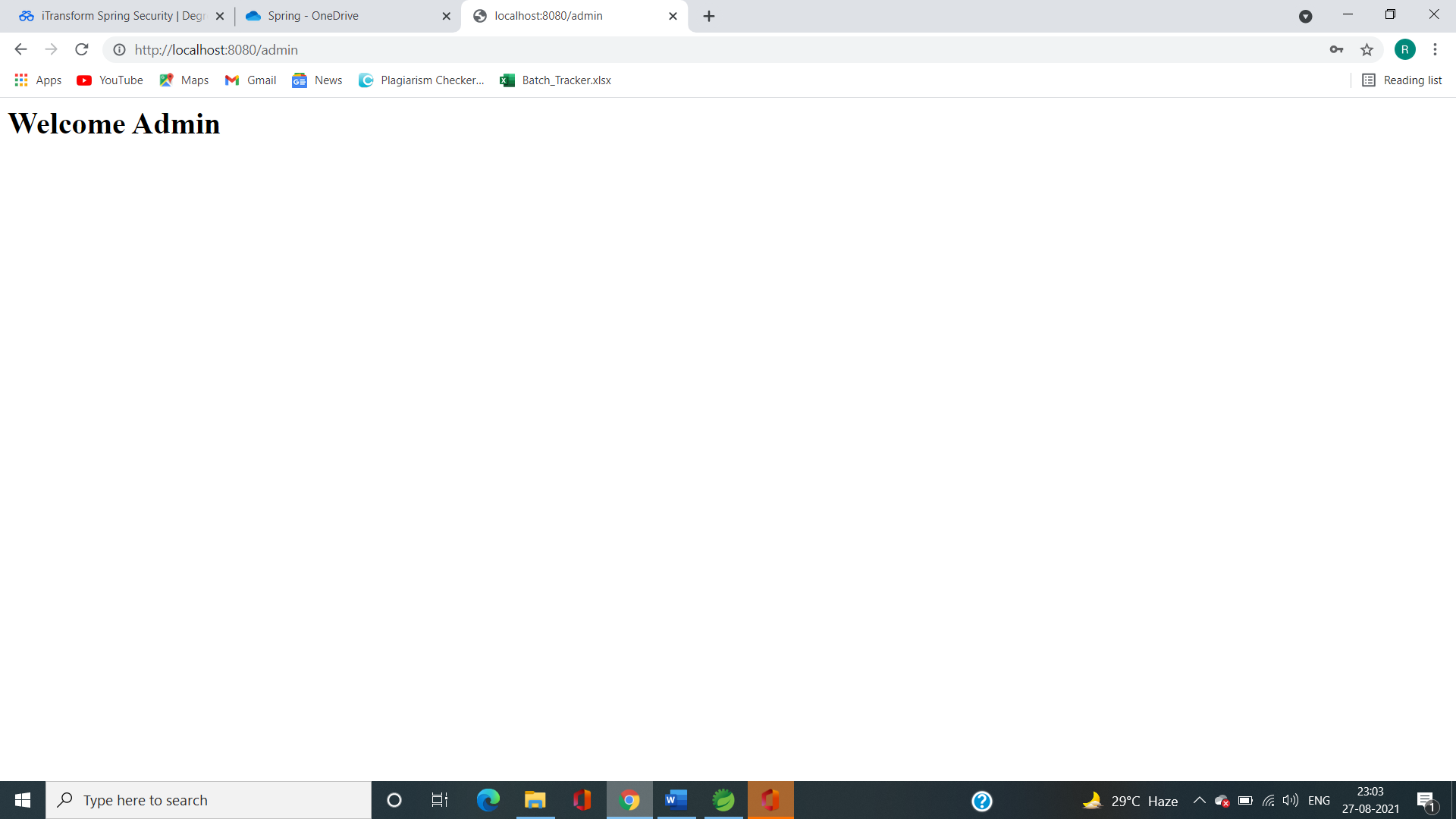
values('user', 'ROLE\_USER');

INSERT INTO authorities(username, authority)

values('admin', 'ROLE\_ADMIN');

output by matching values from data.sql





Assignment 2

**package** com.springsecurity.SecurityApp;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfiguration;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.crypto.password.~~NoOpPasswordEncoder~~;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter{

//this method is for authentication

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

//set your configuration to auth object of AMB

auth.inMemoryAuthentication()

.withUser("Riddhi")

.password("12345")

.roles("USER")

.and()

.withUser("admin")

.password("3030303")

.roles("ADMIN");

}

@Bean

**public** PasswordEncoder getPasswordEncoder() {

**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

//This override method is for Authorization of certain roles

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.authorizeRequests()

.antMatchers("/admin").hasRole("ADMIN") //Allows only admin

.antMatchers("/users").hasAnyRole("USER", "ADMIN") //Allows Both user and admin

.antMatchers("/").permitAll() //Allows Everyone without login also

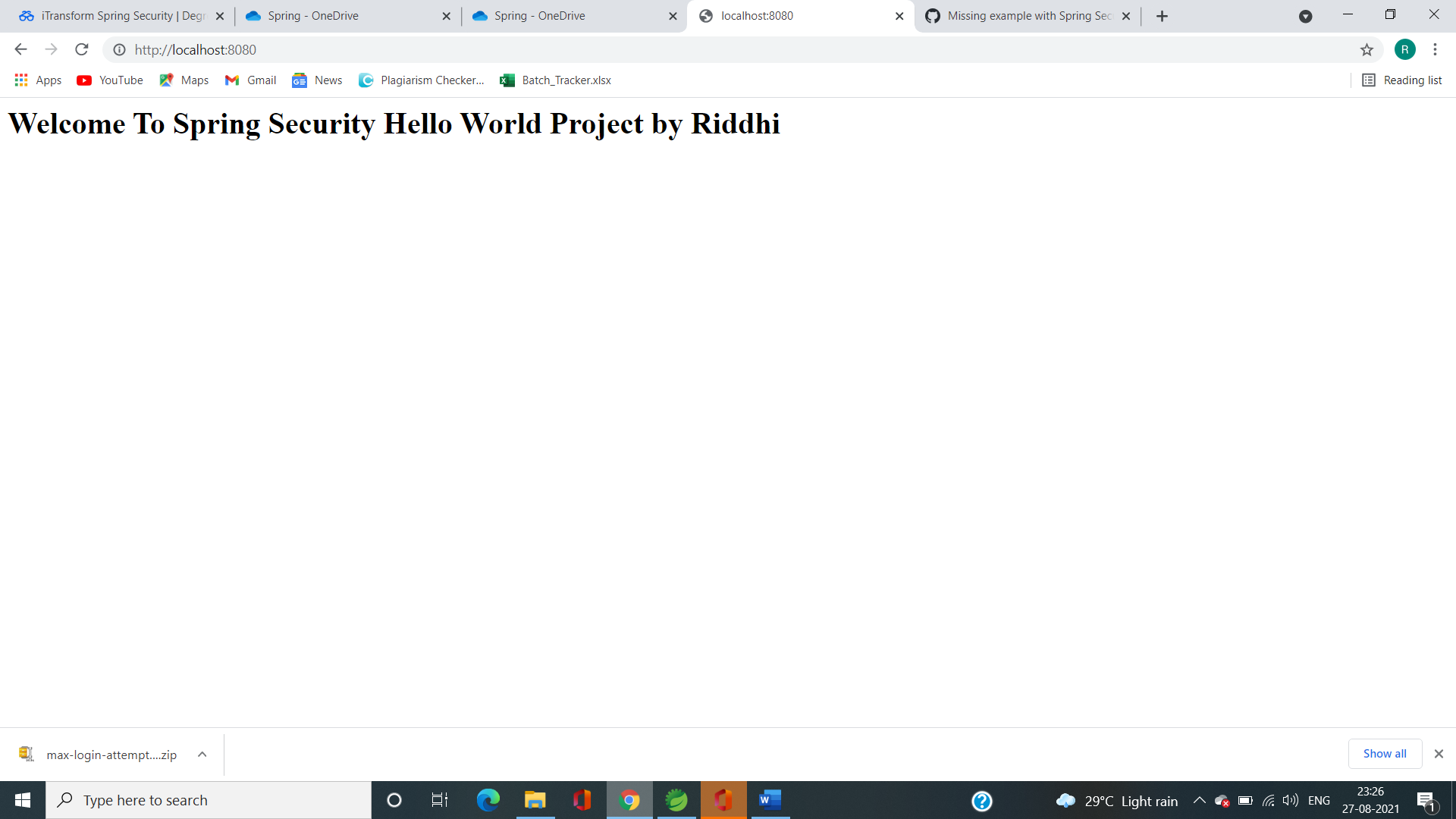
.and().formLogin();

}

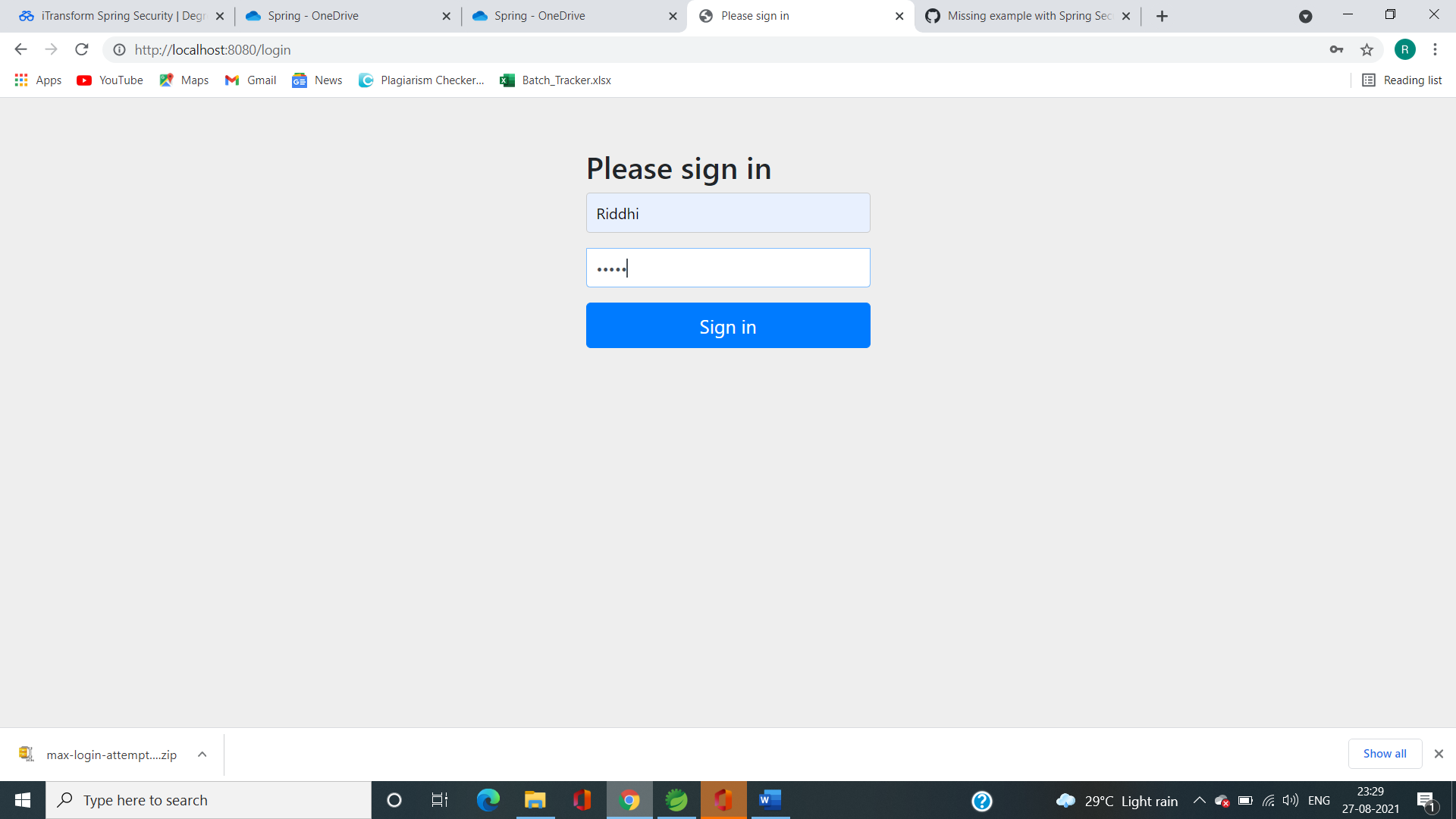
Note: Other code files are same as the code files of 2 question of assignment 1

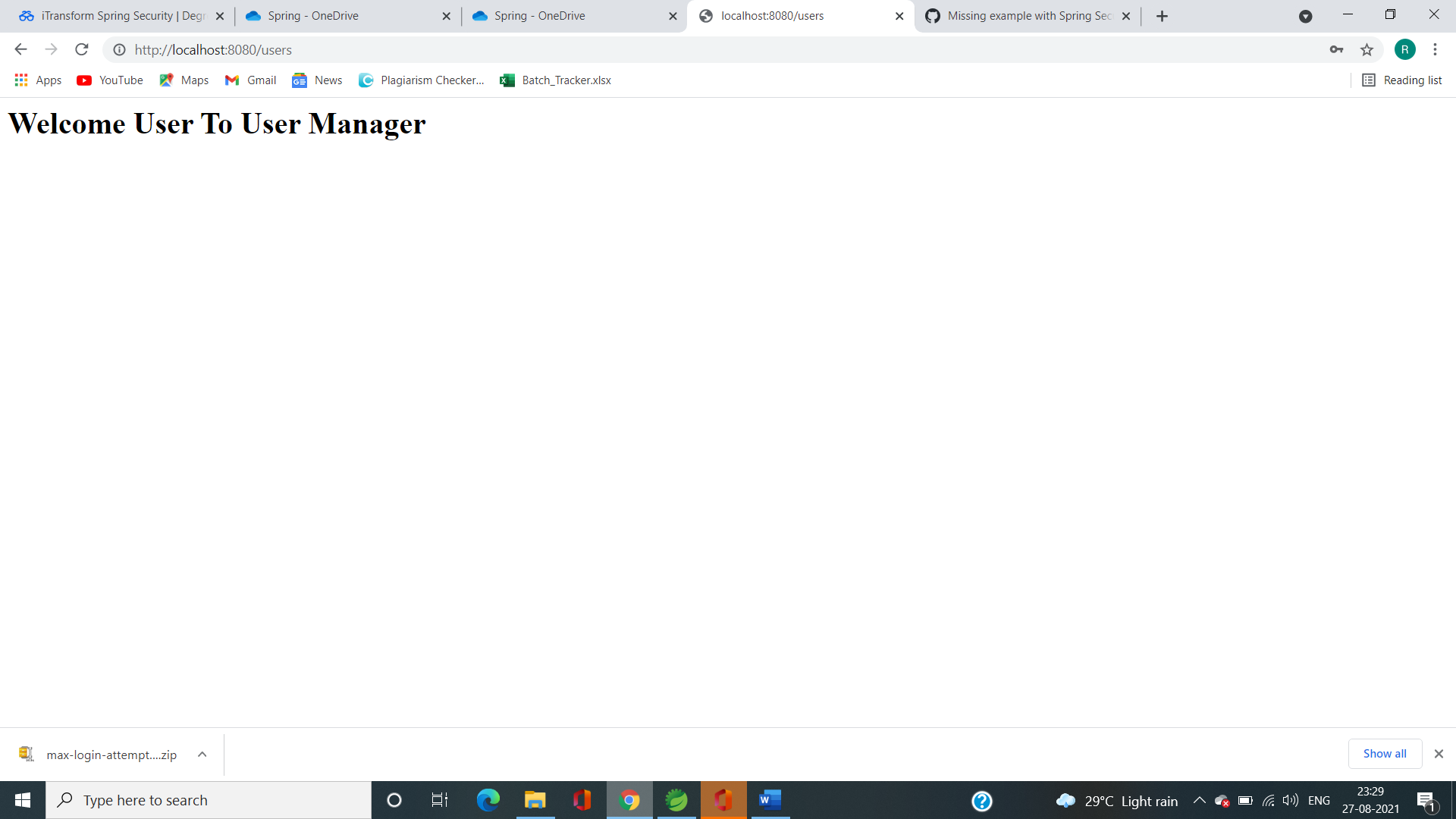
Output:

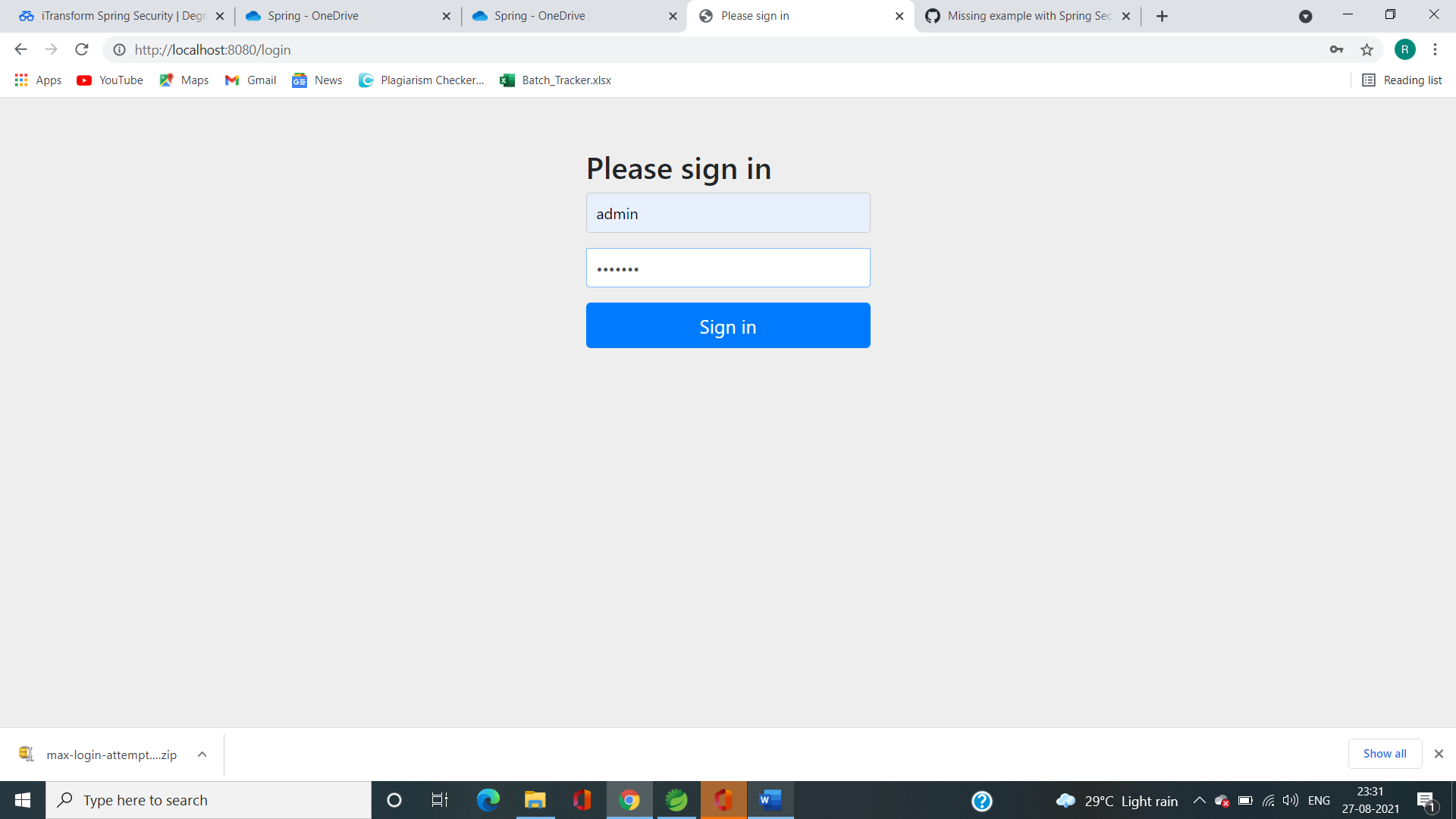
1. Allowed to all users without login

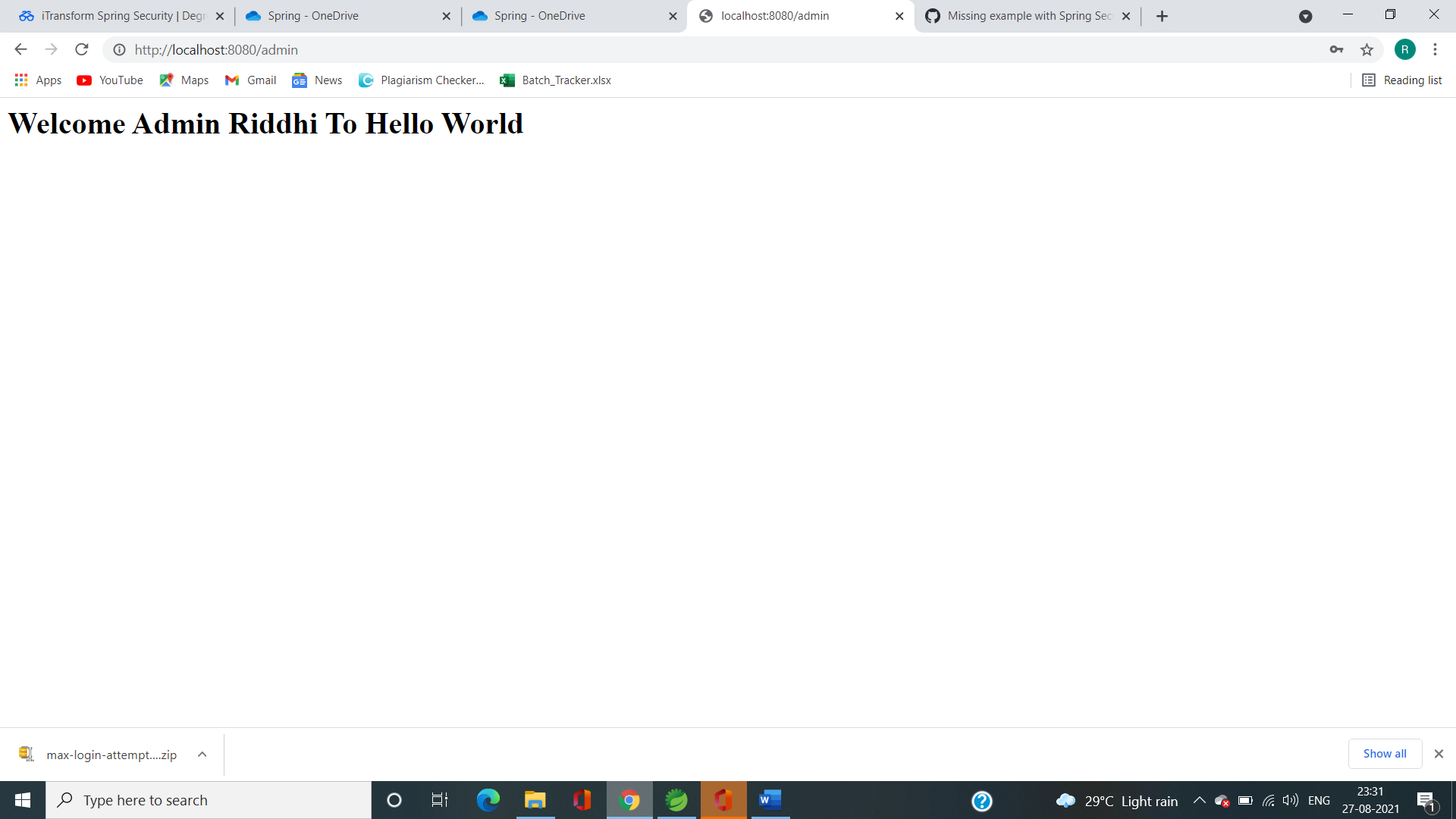


2.Login allowed to users and admin

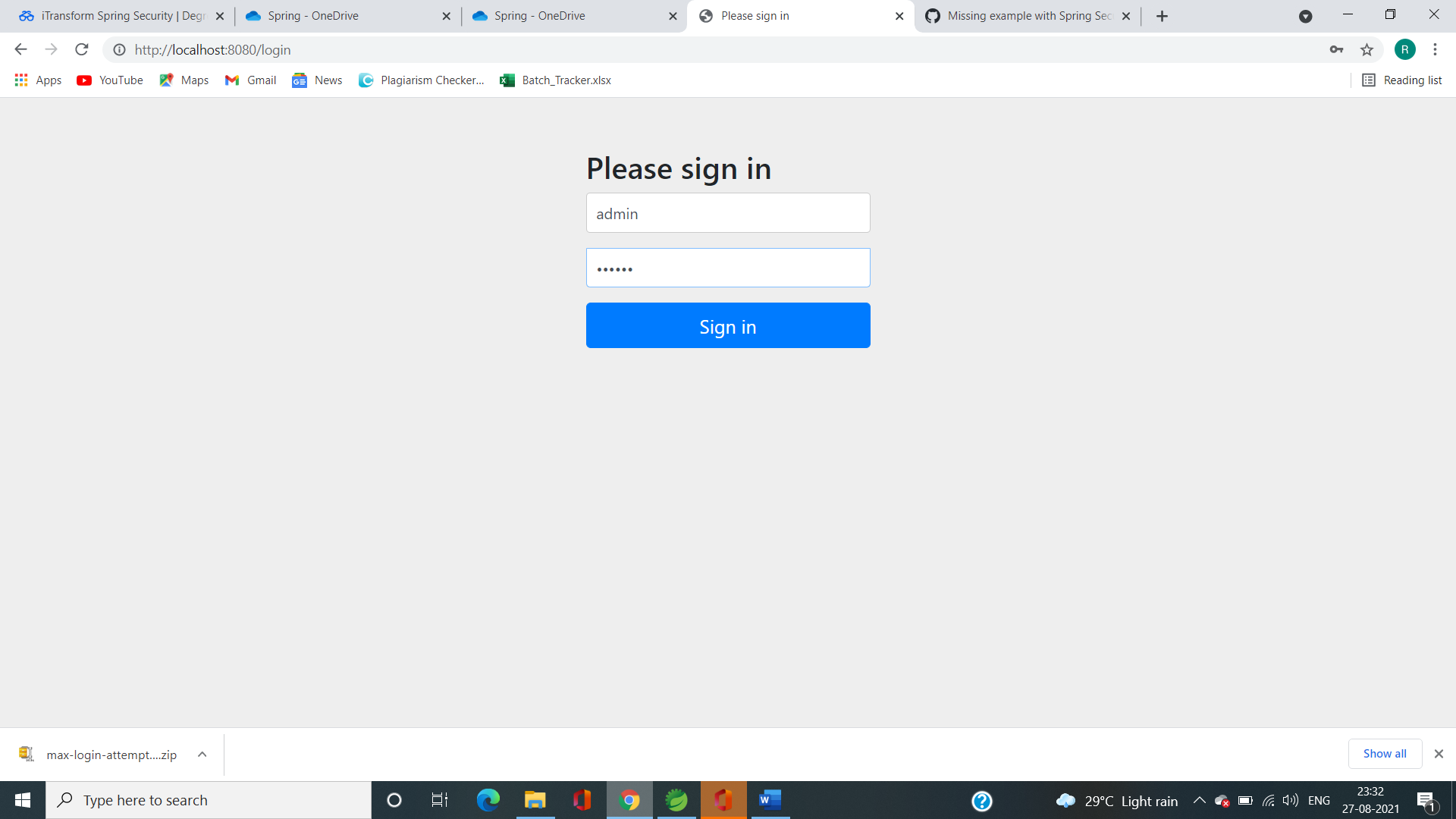


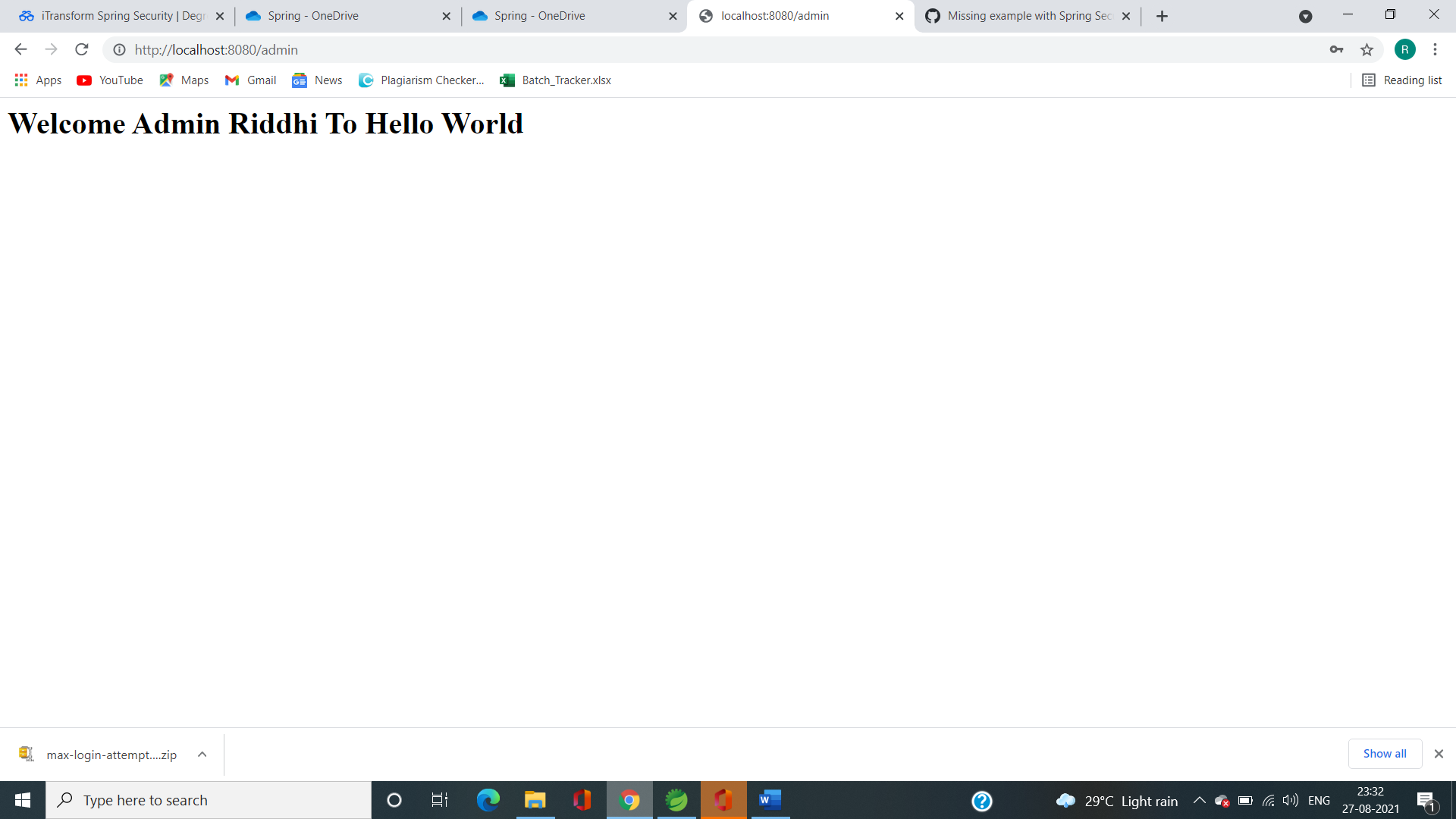






3.Only admin access





Q3. Session Token

**package** io.javabrains.springsecurityjwt;

**import** io.javabrains.springsecurityjwt.filters.JwtRequestFilter;

**import** io.javabrains.springsecurityjwt.models.AuthenticationRequest;

**import** io.javabrains.springsecurityjwt.models.AuthenticationResponse;

**import** io.javabrains.springsecurityjwt.util.JwtUtil;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.BadCredentialsException;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.config.http.SessionCreationPolicy;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.crypto.password.~~NoOpPasswordEncoder~~;

**import** org.springframework.security.crypto.password.PasswordEncoder;

**import** org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

**public** **class** SpringSecurityJwtApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringSecurityJwtApplication.**class**, args);

}

}

@RestController

**class** HelloWorldController {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** JwtUtil jwtTokenUtil;

@Autowired

**private** MyUserDetailsService userDetailsService;

@RequestMapping({ "/hello" })

**public** String firstPage() {

**return** "Hello World";

}

@RequestMapping(value = "/authenticate", method = RequestMethod.***POST***)

**public** ResponseEntity<?> createAuthenticationToken(@RequestBody AuthenticationRequest authenticationRequest) **throws** Exception {

**try** {

authenticationManager.authenticate(

**new** UsernamePasswordAuthenticationToken(authenticationRequest.getUsername(), authenticationRequest.getPassword())

);

}

**catch** (BadCredentialsException e) {

**throw** **new** Exception("Incorrect username or password", e);

}

**final** UserDetails userDetails = userDetailsService

.loadUserByUsername(authenticationRequest.getUsername());

**final** String jwt = jwtTokenUtil.generateToken(userDetails);

**return** ResponseEntity.*ok*(**new** AuthenticationResponse(jwt));

}

}

@EnableWebSecurity

**class** WebSecurityConfig **extends** WebSecurityConfigurerAdapter {

@Autowired

**private** UserDetailsService myUserDetailsService;

@Autowired

**private** JwtRequestFilter jwtRequestFilter;

@Autowired

**public** **void** configureGlobal(AuthenticationManagerBuilder auth) **throws** Exception {

auth.userDetailsService(myUserDetailsService);

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

@Override

@Bean

**public** AuthenticationManager authenticationManagerBean() **throws** Exception {

**return** **super**.authenticationManagerBean();

}

@Override

**protected** **void** configure(HttpSecurity httpSecurity) **throws** Exception {

httpSecurity.csrf().disable()

.authorizeRequests().antMatchers("/authenticate").permitAll().

anyRequest().authenticated().and().

exceptionHandling().and().sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.***STATELESS***);

httpSecurity.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.**class**);

}

}

**package** io.javabrains.springsecurityjwt;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.core.userdetails.UsernameNotFoundException;

**import** org.springframework.stereotype.Service;

**import** java.util.ArrayList;

@Service

**public** **class** MyUserDetailsService **implements** UserDetailsService {

@Override

**public** UserDetails loadUserByUsername(String s) **throws** UsernameNotFoundException {

**return** **new** User("foo", "foo",

**new** ArrayList<>());

}

}

**package** io.javabrains.springsecurityjwt.filters;

**import** io.javabrains.springsecurityjwt.MyUserDetailsService;

**import** io.javabrains.springsecurityjwt.util.JwtUtil;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

**import** org.springframework.stereotype.Component;

**import** org.springframework.web.filter.OncePerRequestFilter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** java.io.IOException;

@Component

**public** **class** JwtRequestFilter **extends** OncePerRequestFilter {

@Autowired

**private** MyUserDetailsService userDetailsService;

@Autowired

**private** JwtUtil jwtUtil;

@Override

**protected** **void** doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

**throws** ServletException, IOException {

**final** String authorizationHeader = request.getHeader("Authorization");

String username = **null**;

String jwt = **null**;

**if** (authorizationHeader != **null** && authorizationHeader.startsWith("Bearer ")) {

jwt = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwt);

}

**if** (username != **null** && SecurityContextHolder.getContext().getAuthentication() == **null**) {

UserDetails userDetails = **this**.userDetailsService.loadUserByUsername(username);

**if** (jwtUtil.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = **new** UsernamePasswordAuthenticationToken(

userDetails, **null**, userDetails.getAuthorities());

usernamePasswordAuthenticationToken

.setDetails(**new** WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}

}

chain.doFilter(request, response);

}

}

**package** io.javabrains.springsecurityjwt.models;

**import** java.io.Serializable;

**public** **class** AuthenticationRequest **implements** Serializable {

**private** String username;

**private** String password;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

//need default constructor for JSON Parsing

**public** AuthenticationRequest()

{

}

**public** AuthenticationRequest(String username, String password) {

**this**.setUsername(username);

**this**.setPassword(password);

}

}

**package** io.javabrains.springsecurityjwt.models;

**import** java.io.Serializable;

**public** **class** AuthenticationResponse **implements** Serializable {

**private** **final** String jwt;

**public** AuthenticationResponse(String jwt) {

**this**.jwt = jwt;

}

**public** String getJwt() {

**return** jwt;

}

}

Output:

