SPRING SECURITY FUNDAMENTALS ASSIGNMENT SOLUTIONS

Assignment 1 : Create Microservices

```
package com.bharath.springcloud.controller;

import com.bharath.springcloud.dto.Coupon;

import com.bharath.springcloud.model.Product;

import com.bharath.springcloud.repository.ProductRepository;

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

import org.springframework.beans.factory.annotation.Value;

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

import org.springframework.web.client.RestTemplate;

import java.util.List;

@RestController

@RequestMapping("/productApi")

public class ProductController {

  @Autowired

  private ProductRepository productRepository;
```

```
    @Autowired
        private RestTemplate restTemplate;
    @Value("${couponService.url}")
        private String CouponServiceURL;
    @PostMapping("/products")
        public Product createProduct(@RequestBody Product product) {
    Coupon coupon = restTemplate.getForObject(CouponServiceURL +
product.getCouponCode(), Coupon.class);
        product.setPrice(product.getPrice().subtract(coupon.getDiscount()));
    return productRepository.save(product);
}
//  Get product by id
        @GetMapping("/product/{id}")
        public Product getProduct(@PathVariable Long id) {
    return productRepository.findById(id).get();
}
//  Get all products in DB
```

```
    @GetMapping("/products")
    public List<Product> getAllProducts() {
    return productRepository.findAll();
}
}
```

Assignment 2 : Secure REST APIs

```
        @Configuration
        public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {
        @Autowired
        UserDetailsServiceImpl userDetailsService;
    @Override
        protected void configure(HttpSecurity http) throws Exception {
    http.httpBasic();
        http.authorizeRequests().mvcMatchers(HttpMethod.GET,
"/productapi/products")
        .hasAnyRole("USER","ADMIN")
```

```
    .mvcMatchers(HttpMethod.POST, "/productapi/products")
        .hasRole("ADMIN").and().csrf().disable();
}
    @Bean
    public PasswordEncoder passwordEncoder(){
    return new BCryptPasswordEncoder();
}
}
```

Assignment 3 : Secure a WebApp

```
        @Service
        public class SecurityServiceImpl implements SecurityService{
    @Autowired
        UserDetailsService userDetailsService;
    @Autowired
        AuthenticationManager authenticationManager;
    @Override
        public boolean login(String userName, String password) {
```

```
    UserDetails userDetails =
userDetailsService.loadUserByUsername(userName);

    UsernamePasswordAuthenticationToken token = new
UsernamePasswordAuthenticationToken(userDetails, password,
userDetails.getAuthorities());

    authenticationManager.authenticate(token);

    boolean authenticated = token.isAuthenticated();

    if(authenticated){

    SecurityContextHolder.getContext().setAuthentication(token);

}

    return authenticated;

}

}
```

Assignment 4 : OAuth in action

OAuth2SecurityConfig.java

```
package com.bharath.springcloud.productservice.config;

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

```
import org.springframework.context.annotation.Configuration;

import
org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;

import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

public class OAuth2SecurityConfig extends
WebSecurityConfigurerAdapter {

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws
Exception {

  return super.authenticationManagerBean();

}

  @Bean

  public BCryptPasswordEncoder bCryptPasswordEncoder(){

  return new BCryptPasswordEncoder();
```

```
}

}
```

AuthorizationServerConfig.java

```
package com.bharath.springcloud.productservice.config;

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

import org.springframework.context.annotation.Configuration;

import
org.springframework.security.authentication.AuthenticationManager;

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

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

import
org.springframework.security.oauth2.config.annotation.configurers.ClientDe
tailsServiceConfigurer;

import
org.springframework.security.oauth2.config.annotation.web.configuration.A
uthorizationServerConfigurerAdapter;

import
org.springframework.security.oauth2.config.annotation.web.configuration.E
nableAuthorizationServer;
```

```
import
org.springframework.security.oauth2.config.annotation.web.configurers.Aut
horizationServerEndpointsConfigurer;

import
org.springframework.security.oauth2.provider.token.store.JdbcTokenStore;

import javax.sql.DataSource;

@Configuration

@EnableAuthorizationServer

public class AuthorizationServerConfig extends
AuthorizationServerConfigurerAdapter {

private static final String RESOURCE_ID = "productservice";

  @Autowired

private UserDetailsService userDetailsService;

  @Autowired

private AuthenticationManager authenticationManager;

  @Autowired

private PasswordEncoder passwordEncoder;

  @Autowired
```

```
        private DataSource dataSource;

    @Override

        public void configure(AuthorizationServerEndpointsConfigurer
endpoints) throws Exception {

    endpoints.tokenStore(new JdbcTokenStore(dataSource))

    .authenticationManager(authenticationManager)

    .userDetailsService(userDetailsService);

    }

    @Override

    public void configure(ClientDetailsServiceConfigurer clients) throws
Exception {

    clients.inMemory()

    .withClient("productclientapp")

    .secret(passwordEncoder.encode("9999"))

    .authorizedGrantTypes("password", "refresh_token")

    .scopes("read", "write")

    .resourceIds(RESOURCE_ID);

}
```

}

ResourceServerConfig.java

```
package com.bharath.springcloud.productservice.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

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

import
org.springframework.security.oauth2.config.annotation.web.configuration.E
nableResourceServer;

import
org.springframework.security.oauth2.config.annotation.web.configuration.R
esourceServerConfigurerAdapter;

import
org.springframework.security.oauth2.config.annotation.web.configurers.Res
ourceServerSecurityConfigurer;

@Configuration

@EnableResourceServer

public class ResourceServerConfig extends
ResourceServerConfigurerAdapter {
```

```
        private static final String RESOURCE_ID = "productservice";

    @Override

        public void configure(ResourceServerSecurityConfigurer resources)
throws Exception {

    resources.resourceId(RESOURCE_ID);

}

    @Override

    public void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests().mvcMatchers(HttpMethod.GET,

    "/productapi/products/**"

        ).hasAnyRole("USER", "ADMIN")

        .mvcMatchers(HttpMethod.POST,

    "/productapi/products"

    ).hasRole("ADMIN")

    .anyRequest().denyAll().and().csrf().disable();

}

}
```

Assignment 5 : JWT

```
// productservice/application.properties


security.oauth2.resource.jwt.key-uri=http://localhost:9092/oauth/token_key

@Configuration

public class OAuth2SecurityConfig extends
WebSecurityConfigurerAdapter {

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws
Exception {

return super.authenticationManagerBean();

}

@Bean

public BCryptPasswordEncoder bCryptPasswordEncoder() {

return new BCryptPasswordEncoder();

}

}
```

```
	@Configuration

	@EnableResourceServer

	public class ResouceServerConfig extends
ResourceServerConfigurerAdapter {

	private static final String RESOURCE_ID = "productservice";

	@Override

	public void configure(ResourceServerSecurityConfigurer resources)
throws Exception {

	resources.resourceId(RESOURCE_ID);

}

	@Override

	public void configure(HttpSecurity http) throws Exception {

	http.authorizeRequests().mvcMatchers(HttpMethod.GET,
"/productapi/products/{id:^[0-9]*$}")

	.hasAnyRole("USER", "ADMIN").mvcMatchers(HttpMethod.POST,
"/productapi/products").hasRole("ADMIN")

	.anyRequest().denyAll().and().csrf().disable();

}

}
```

```
@Configuration

@EnableAuthorizationServer

public class AuthorizationServerConfig extends
AuthorizationServerConfigurerAdapter {

private static final String RESOURCE_ID = "productservice";

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private UserDetailsService userDetailsService;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

private DataSource dataSource;

@Value("${keyFile}")

private String keyFile;

@Value("${password}")

private String password;
```

```
	@Value("${alias}")

	private String alias;

	@Override

	public void configure(AuthorizationServerEndpointsConfigurer
endpoints) throws Exception {


endpoints.tokenStore(tokenStore()).accessTokenConverter(jwtAccessToke
nConverter())


.authenticationManager(authenticationManager).userDetailsService(userD
etailsService);

}

	@Override

	public void configure(ClientDetailsServiceConfigurer clients) throws
Exception {


clients.inMemory().withClient("couponclientapp").secret(passwordEncoder.
encode("9999"))

	.authorizedGrantTypes("password", "refresh_token").scopes("read",
"write").resourceIds(RESOURCE_ID);

	;
```

```
}

    @Override

    public void configure(AuthorizationServerSecurityConfigurer security)
throws Exception {

    security.tokenKeyAccess("permitAll()");

}

    @Bean

    public TokenStore tokenStore() {

    return new JwtTokenStore(jwtAccessTokenConverter());

}

    @Bean

    public JwtAccessTokenConverter jwtAccessTokenConverter() {

    JwtAccessTokenConverter jwtAccessTokenConverter = new
JwtAccessTokenConverter();

    KeyStoreKeyFactory keyStoreKeyFactory = new
KeyStoreKeyFactory(new ClassPathResource(keyFile),

    password.toCharArray());

    KeyPair keyPair = keyStoreKeyFactory.getKeyPair(alias);
```

```
        jwtAccessTokenConverter.setKeyPair(keyPair);

        return jwtAccessTokenConverter;

}

}
```

2.

The instructions are in the video

```
// productservice/application.properties


security.oauth2.resource.jwt.key-uri=http://localhost:9092/oauth/token_key

        @Configuration

        public class OAuth2SecurityConfig extends
WebSecurityConfigurerAdapter {

        @Override

        @Bean

        public AuthenticationManager authenticationManagerBean() throws
Exception {

        return super.authenticationManagerBean();

}
```

```
	@Bean
	public BCryptPasswordEncoder bCryptPasswordEncoder() {
	return new BCryptPasswordEncoder();
}
}
	@Configuration
	@EnableResourceServer
	public class ResouceServerConfig extends
ResourceServerConfigurerAdapter {
	private static final String RESOURCE_ID = "productservice";
	@Override
	public void configure(ResourceServerSecurityConfigurer resources)
throws Exception {
	resources.resourceId(RESOURCE_ID);
}
	@Override
	public void configure(HttpSecurity http) throws Exception {
```

```
        http.authorizeRequests().mvcMatchers(HttpMethod.GET,
"/productapi/products/{id:^[0-9]*$}")

        .hasAnyRole("USER", "ADMIN").mvcMatchers(HttpMethod.POST,
"/productapi/products").hasRole("ADMIN")

        .anyRequest().denyAll().and().csrf().disable();

}

}

        @Configuration

        @EnableAuthorizationServer

        public class AuthorizationServerConfig extends
AuthorizationServerConfigurerAdapter {

        private static final String RESOURCE_ID = "productservice";

        @Autowired

        private AuthenticationManager authenticationManager;

        @Autowired

        private UserDetailsService userDetailsService;

        @Autowired

        private PasswordEncoder passwordEncoder;
```

```
	@Autowired

	private DataSource dataSource;

	@Value("${keyFile}")

	private String keyFile;

	@Value("${password}")

	private String password;

	@Value("${alias}")

	private String alias;

	@Override

	public void configure(AuthorizationServerEndpointsConfigurer
endpoints) throws Exception {


endpoints.tokenStore(tokenStore()).accessTokenConverter(jwtAccessToke
nConverter())


.authenticationManager(authenticationManager).userDetailsService(userD
etailsService);

}

	@Override
```

```
	public void configure(ClientDetailsServiceConfigurer clients) throws
Exception {


clients.inMemory().withClient("couponclientapp").secret(passwordEncoder.
encode("9999"))

	.authorizedGrantTypes("password", "refresh_token").scopes("read",
"write").resourceIds(RESOURCE_ID);

;

}

	@Override

	public void configure(AuthorizationServerSecurityConfigurer security)
throws Exception {

	security.tokenKeyAccess("permitAll()");

}

	@Bean

	public TokenStore tokenStore() {

	return new JwtTokenStore(jwtAccessTokenConverter());

}

	@Bean
```

```
	public JwtAccessTokenConverter jwtAccessTokenConverter() {

	JwtAccessTokenConverter jwtAccessTokenConverter = new
JwtAccessTokenConverter();

	KeyStoreKeyFactory keyStoreKeyFactory = new
KeyStoreKeyFactory(new ClassPathResource(keyFile),

	password.toCharArray());

	KeyPair keyPair = keyStoreKeyFactory.getKeyPair(alias);

	jwtAccessTokenConverter.setKeyPair(keyPair);

	return jwtAccessTokenConverter;

}

}
```

Assignment 6 : CSRF

```
	@Configuration

	public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

   @Autowired

	UserDetailsServiceImpl userDetailsService;

   @Override
```

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

    auth.userDetailsService(userDetailsService);

}

    @Override

        protected void configure(HttpSecurity http) throws Exception {

    //http.formLogin(); //replaced by our own login form

    http.authorizeRequests()

        .mvcMatchers(HttpMethod.GET,

//  "/productapi/products/{id:^[0-9]*$}",

    "/index",

    "/showGetProduct",

        "/getProduct",

        "/productDetails").hasAnyRole("USER", "ADMIN")

    .mvcMatchers(HttpMethod.GET,

    "/showCreateProduct",

    "/createProduct",
```

```
    "/createResponse").hasRole("ADMIN")

    .mvcMatchers(HttpMethod.POST,

    "/getProduct").hasAnyRole("USER", "ADMIN")

    .mvcMatchers(HttpMethod.POST,

//  "/productapi/products",

        "/saveProduct",

        "/getProduct").hasRole("ADMIN")

    .mvcMatchers("/", "/login", "/showRegistrationPage",
"/registerUser").permitAll()

        .anyRequest().denyAll()

    .and()

    .logout().logoutSuccessUrl("/");


    http.csrf(csrfCustomizer -> {

//  //methode1

//  csrfCustomizer.ignoringAntMatchers("/couponapi/coupons/**");

    //methode2

    RequestMatcher requestMatcher = new
```

```
RegexRequestMatcher("/productapi/products/*", "POST");

    csrfCustomizer.ignoringRequestMatchers(requestMatcher);

//  //methode3

//  MvcRequestMatcher mvcRequestMatcher = new
MvcRequestMatcher(new HandlerMappingIntrospector(),
"/couponapi/coupons/**");

//  csrfCustomizer.ignoringRequestMatchers(mvcRequestMatcher);

    });

    }

    @Bean

    public PasswordEncoder passwordEncoder(){

    return new BCryptPasswordEncoder();

}

    @Override

    @Bean //exposes THIS authenticationmanager as a bean to
SecurityServiceImpl

    public AuthenticationManager authenticationManagerBean() throws
Exception {

    return super.authenticationManagerBean();
```

```
}

}
```

a seperate config for the api

```
        @Configuration

        @Order(1)

        public class ApiSecurityConfig extends
WebSecurityConfigurerAdapter {

    @Autowired

        UserDetailsServiceImpl userDetailsService;

    @Override

        protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

    auth.userDetailsService(userDetailsService);

}

    @Override

    protected void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests()
```

```
        .mvcMatchers(HttpMethod.GET,
    "/productapi/products/{id:^[0-9]*$}").hasAnyRole("USER", "ADMIN")
    .mvcMatchers(HttpMethod.POST,
        "/productapi/products").hasRole("ADMIN")
        .and()
        .httpBasic().authenticationEntryPoint(authenticationEntryPoint());
    http.csrf(csrfCustomizer -> {
        MvcRequestMatcher mvcRequestMatcher = new
MvcRequestMatcher(new HandlerMappingIntrospector(),
    "/productapi/products/**");
        csrfCustomizer.ignoringRequestMatchers(mvcRequestMatcher);
});
}
    @Bean
    public AuthenticationEntryPoint authenticationEntryPoint(){
    BasicAuthenticationEntryPoint entryPoint =
    new BasicAuthenticationEntryPoint();
```

```
    entryPoint.setRealmName("api realm");

    return entryPoint;

}

}
```

adapted productrestcontroller:

```
    @PostMapping("/products")

    public Product create(@RequestBody Product product){

    System.out.println(couponServiceUrl);

    System.out.println(couponServiceUrl+product.getCouponCode());

    restTemplate.getInterceptors().add(

    new BasicAuthenticationInterceptor(couponServiceUser,
couponServicePassword));

    Coupon coupon =
restTemplate.getForObject(couponServiceUrl+product.getCouponCode(),
Coupon.class);

//  ResponseEntity<Coupon> coupon = restTemplate.exchange(

//  couponServiceUrl+product.getCouponCode(),

//  HttpMethod.GET, null, Coupon.class);
```

```
    product.setPrice(product.getPrice().subtract(coupon.getDiscount()));

    return productRepo.save(product);

}
```

And a added class, for the productrestcontroller:

```
        public class HttpComponentsClientHttpRequestFactoryBasicAuth

    extends HttpComponentsClientHttpRequestFactory {

    HttpHost host;

    public HttpComponentsClientHttpRequestFactoryBasicAuth(HttpHost
host) {

    super();

    this.host = host;

}

    protected HttpContext createHttpContext(HttpMethod httpMethod, URI
uri) {

    return createHttpContext();

}

    private HttpContext createHttpContext() {

    AuthCache authCache = new BasicAuthCache();
```

```
    BasicScheme basicAuth = new BasicScheme();
    authCache.put(host, basicAuth);
    BasicHttpContext localcontext = new BasicHttpContext();
    localcontext.setAttribute(HttpClientContext.AUTH_CACHE, authCache);
    return localcontext;
}
}
```

Assignment 7 : CORS

React code -

```
import React from 'react';
import axios from "axios";
import {useEffect} from "react";
function App() {
useEffect(()=>{
axios.post('http://localhost:9091/couponapi/coupon', {
code: 'SUPERSALE_CORS',
```

```
        discount: 50,

        expDate: 11/11/2222

        }).then((response) => {

        document.write("Coupon Code: "+response.data.code+"<br/>")

        document.write("Coupon Discount:
"+response.data.discount+"<br/>")

        document.write("Coupon Expiry Date:
"+response.data.expDate+"<br/>")

});

},[])

        return (

    <div className="App">

    </div>

);

}

        export default App;

        Java Code -

        WebSecurityConfig  -
```

```
package com.csrf.spCsrfCoupon.config;

import java.util.List;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import
org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.config.annotation.authentication.builders.Auth
enticationManagerBuilder;

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

import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;

import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

```
import
org.springframework.security.web.servlet.util.matcher.MvcRequestMatcher;

import
org.springframework.security.web.util.matcher.RegexRequestMatcher;

import
org.springframework.security.web.util.matcher.RequestMatcher;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.CorsConfigurationSource;

import
org.springframework.web.servlet.handler.HandlerMappingIntrospector;

import com.csrf.spCsrfCoupon.security.UserDetailServiceImpl;

@Configuration

public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

@Autowired

UserDetailServiceImpl userDetailServiceImpl;

@Override

protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

auth.userDetailsService(userDetailServiceImpl);
```

```
}

        @Override

        protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()

        .mvcMatchers(HttpMethod.GET,
"/couponapi/coupon/**","/index","/showGetCoupon","/getCoupon","/coupon
Details")

        .hasAnyRole("USER", "ADMIN")

        .mvcMatchers(HttpMethod.GET,
"/showCreateCoupon","/createCoupon","/createResponse").hasRole("ADMI
N")

        .mvcMatchers(HttpMethod.POST,
"/getCoupon").hasAnyRole("USER", "ADMIN")

        .mvcMatchers(HttpMethod.POST,
"/couponapi/coupon","/saveCoupon","/getCoupon").hasRole("ADMIN")

        .mvcMatchers("/",
"/login","logout","/showReg","registerUser").permitAll()

        .and().logout().logoutSuccessUrl("/");

        http.cors(corsCustomizer -> {

        CorsConfigurationSource configurationSource = request->{
```

```
        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.setAllowedOrigins(List.of("localhost:3000"));
        corsConfiguration.setAllowedMethods(List.of("POST"));
        return corsConfiguration;
};
        corsCustomizer.configurationSource(configurationSource);
});
        @Bean
        public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
}
        @Override
        @Bean
        public AuthenticationManager authenticationManagerBean() throws
Exception {
        return super.authenticationManagerBean();
}
```

}

2.The instructions are in the video

React code -

```
import React from 'react';

import axios from "axios";

import {useEffect} from "react";

function App() {

useEffect(()=>{

axios.post('http://localhost:9091/couponapi/coupon', {

code: 'SUPERSALE_CORS',

discount: 50,

expDate: 11/11/2222

}).then((response) => {

document.write("Coupon Code: "+response.data.code+"<br/>")

document.write("Coupon Discount:
"+response.data.discount+"<br/>")

document.write("Coupon Expiry Date:
```

```
"+response.data.expDate+"<br/>")

});

},[])

        return (

    <div className="App">

    </div>

);

}

        export default App;

        Java Code -

        WebSecurityConfig  -

        package com.csrf.spCsrfCoupon.config;

        import java.util.List;

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

        import org.springframework.context.annotation.Bean;

        import org.springframework.context.annotation.Configuration;

        import org.springframework.http.HttpMethod;
```

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

import
org.springframework.security.config.annotation.authentication.builders.Auth
enticationManagerBuilder;

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

import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;

import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

import
org.springframework.security.web.servlet.util.matcher.MvcRequestMatcher;

import
org.springframework.security.web.util.matcher.RegexRequestMatcher;

import
org.springframework.security.web.util.matcher.RequestMatcher;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.CorsConfigurationSource;
```

```
	import
org.springframework.web.servlet.handler.HandlerMappingIntrospector;

	import com.csrf.spCsrfCoupon.security.UserDetailServiceImpl;

	@Configuration

	public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

	@Autowired

	UserDetailServiceImpl userDetailServiceImpl;

	@Override

	protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

	auth.userDetailsService(userDetailServiceImpl);

}

	@Override

	protected void configure(HttpSecurity http) throws Exception {

	http.authorizeRequests()

	.mvcMatchers(HttpMethod.GET,
"/couponapi/coupon/**","/index","/showGetCoupon","/getCoupon","/coupon
Details")
```

```
        .hasAnyRole("USER", "ADMIN")

        .mvcMatchers(HttpMethod.GET,
"/showCreateCoupon","/createCoupon","/createResponse").hasRole("ADMI
N")

        .mvcMatchers(HttpMethod.POST,
"/getCoupon").hasAnyRole("USER", "ADMIN")

        .mvcMatchers(HttpMethod.POST,
"/couponapi/coupon","/saveCoupon","/getCoupon").hasRole("ADMIN")

        .mvcMatchers("/",
"/login","logout","/showReg","registerUser").permitAll()

        .and().logout().logoutSuccessUrl("/");

        http.cors(corsCustomizer -> {

        CorsConfigurationSource configurationSource = request->{

        CorsConfiguration corsConfiguration = new CorsConfiguration();

        corsConfiguration.setAllowedOrigins(List.of("localhost:3000"));

        corsConfiguration.setAllowedMethods(List.of("POST"));

        return corsConfiguration;

};

        corsCustomizer.configurationSource(configurationSource);
```

```
});

    @Bean

    public PasswordEncoder passwordEncoder() {

    return new BCryptPasswordEncoder();

}

    @Override

    @Bean

    public AuthenticationManager authenticationManagerBean() throws
Exception {

    return super.authenticationManagerBean();

}

}
```