package security.config;

import java.util.Arrays;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.Customizer;

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.http.SessionCreationPolicy;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

import org.springframework.web.filter.CorsFilter;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import org.springframework.web.cors.CorsConfiguration;

import security.service.CustomerService;

import security.util.JwtFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

    @Autowired

    private CustomerService userDetailsService;

    @Autowired

    private JwtFilter jwtFilter;

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf(csrf -> csrf.disable()).cors().and()

            .authorizeHttpRequests(auth -> auth

                .requestMatchers("/auth/register","/auth/login").permitAll()

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

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

                .anyRequest().authenticated()

            )

          . sessionManagement(sm -> sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS));

        http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();

    }

    @Bean

    public CorsFilter corsFilter() {

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

        CorsConfiguration config = new CorsConfiguration();

        config.setAllowedOrigins(Arrays.asList("http://localhost:4200"));

        config.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));

        config.setAllowedHeaders(Arrays.asList("\*"));

        config.setAllowCredentials(true);

        source.registerCorsConfiguration("/\*\*", config);

        return new CorsFilter(source);

    }

    @Bean

    public AuthenticationManager authManager(HttpSecurity http, PasswordEncoder encoder) throws Exception {

        return http.getSharedObject(AuthenticationManagerBuilder.class)

            .userDetailsService(userDetailsService)

            .passwordEncoder(encoder)

            .and()

            .build();

    }

    @Bean

    public PasswordEncoder passwordEncoder() {

        return new BCryptPasswordEncoder();

    }

}

package security.controller;

import java.util.List;

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

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

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

@RestController

//@CrossOrigin("\*")

public class TestController {

    @GetMapping("/user")

    public List<String> getUser()

    {

        return List.of("VIEW\_PRODUCT","MAKE\_ORDER");

    }

    @GetMapping("/admin")

    public  List<String> getAdmin()

    {

        return List.of("CREATE\_TASK","CHECK\_TASK");

    }

}

package security.controller;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

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

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

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

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 security.dto.LoginDto;

import security.dto.UserDTO;

import security.service.UserService;

import security.util.JwtUtil;

@RestController

@CrossOrigin(origins = "http://localhost:4200")

@RequestMapping("/auth")

public class UserController {

@Autowired

private UserService service;

@Autowired

private AuthenticationManager authManager;

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/register")

public ResponseEntity<String> signUpUser(@RequestBody UserDTO dto)

{

service.registerUser(dto);

return ResponseEntity.ok("successfully registered");

}

@PostMapping("/login")

public ResponseEntity<String> login(@RequestBody LoginDto loginDto) {

Authentication auth = authManager.authenticate(

new UsernamePasswordAuthenticationToken(loginDto.getUsername(), loginDto.getPassword()));

UserDetails userDetails = (UserDetails) auth.getPrincipal();

String token=null;

if(userDetails!=null)

{

token = jwtUtil.generateToken(userDetails);

}

return ResponseEntity.ok(token);

}

}package security.service;

import java.util.List;

import java.util.stream.Collectors;

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

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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 security.entity.User;

import security.repository.UserRepository;

@Service

public class CustomerService implements UserDetailsService {

    @Autowired

    private UserRepository userRepos;

    @Override

    public UserDetails loadUserByUsername(String userName) throws UsernameNotFoundException {

        System.out.println("hi");

    User user=userRepos.findById(userName).orElseThrow(()->new UsernameNotFoundException("User Name not found"));

        //User user=userRepos.getById(userName);

        System.out.println(user);

        List<SimpleGrantedAuthority> authorities=user.getRoles().stream().

                map(role->new SimpleGrantedAuthority("ROLE\_"+role.getRoleName())).collect(Collectors.toList());

    UserDetails d= new org.springframework.security.core.userdetails.User(

                user.getUserName(),

                user.getUserPassword(),

                user.isActivated(), true, true,true,

                authorities

        );

    System.out.println(d.getUsername() + " "+d.getPassword());

    return d;

    }

}

package security.service;

import java.util.Set;

import java.util.stream.Collectors;

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

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

import org.springframework.stereotype.Service;

import security.dto.UserDTO;

import security.entity.Role;

import security.entity.User;

import security.repository.RolesRepository;

import security.repository.UserRepository;

@Service

public class UserService {

    @Autowired

    private UserRepository userRepos;

    @Autowired

    private RolesRepository rolesRepos;

    @Autowired

    private PasswordEncoder encoder;

public User registerUser(UserDTO dto)

{

    User user=new User();

    user.setUserName(dto.getUserName());

    user.setUserPassword(encoder.encode(dto.getUserPassword()));

    user.setActivated(true);

    Set<Role> roles=dto.getRoleNames().stream().

            map(roleNames->  rolesRepos.findById(roleNames).

                    orElseThrow(()->new RuntimeException("role name not found")))

            .collect(Collectors.toSet());

    user.setRoles(roles);

    return userRepos.save(user);

}

}

package security.util;

import java.io.IOException;

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 jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import security.service.CustomerService;

@Component

public class JwtFilter extends OncePerRequestFilter {

    @Autowired

    private JwtUtil jwtUtil;

    @Autowired

    private CustomerService userDetailsService;

    @Override

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

        throws ServletException, IOException {

        //Bearer+\s+token

        final String authHeader = request.getHeader("Authorization");

        String username = null;

        String token = null;

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

            token = authHeader.substring(7);

            try {

                username = jwtUtil.extractUsername(token);

            } catch (Exception e) {

                response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);

                return;

            }

        }

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

            UserDetails userDetails = userDetailsService.loadUserByUsername(username);

            if (jwtUtil.validateToken(token, userDetails)) {

                UsernamePasswordAuthenticationToken authToken =

                    new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

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

                SecurityContextHolder.getContext().setAuthentication(authToken);

            }

        }

        chain.doFilter(request, response);

    }

}

package security.util;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.stream.Collectors;

import org.springframework.security.core.GrantedAuthority;

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

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

@Component

public class JwtUtil {

//private final String secret = "mySuperSecretKey1234567890123456"; // 32 chars = 256 bits

private final String secret = "a-string-secret-at-least-256-bits-long";

    private final long expiration = 1000 \* 60 \* 60; // 1 hour

    public String generateToken(UserDetails userDetails) {

        Map<String, Object> claims = new HashMap();

        claims.put("roles", userDetails.getAuthorities().stream()

            .map(GrantedAuthority::getAuthority).collect(Collectors.toList()));

        return Jwts.builder()

            .setClaims(claims)

            .setSubject(userDetails.getUsername())

            .setIssuedAt(new Date())

            .setExpiration(new Date(System.currentTimeMillis() + expiration))

            .signWith(Keys.hmacShaKeyFor(secret.getBytes()), SignatureAlgorithm.HS256)

            .compact();

    }

    public String extractUsername(String token) {

        return extractClaims(token).getSubject();

    }

    public boolean validateToken(String token, UserDetails userDetails) {

        String username = extractUsername(token);

        return username.equals(userDetails.getUsername()) && !isTokenExpired(token);

    }

    private boolean isTokenExpired(String token) {

        return extractClaims(token).getExpiration().before(new Date());

    }

    private Claims extractClaims(String token) {

        return Jwts.parserBuilder()

            .setSigningKey(Keys.hmacShaKeyFor(secret.getBytes()))

            .build()

            .parseClaimsJws(token)

            .getBody();

    }

}

import { Component, OnInit } from '@angular/core';

import { ApiService } from '../services/api.service';

@Component({

  selector: 'app-admin',

  templateUrl: './admin.component.html'

})

export class AdminComponent implements OnInit {

  adminMessage: string = '';

  constructor(private apiService: ApiService) {}

  ngOnInit() {

    this.apiService.getAdmin().subscribe(

      (data) => {this.adminMessage = data;

        console.log(data);

      },

      (error) => {this.adminMessage = 'Error fetching admin data';

    console.log(error);

      }

    );

  }

}

import { Component } from '@angular/core';

import { AuthService } from '../services/auth.service';

import { Router } from '@angular/router';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html'

})

export class LoginComponent {

  username!: string;

  password!: string;

  constructor(private authService: AuthService, private router: Router) {}

  login() {

    this.authService.login({ username: this.username, password: this.password })

      .subscribe(token => {

        localStorage.setItem('jwtToken', token);

        //console.log(token);

        const role = this.authService.getUserRole();

        console.log(role);

        role == 'ROLE\_ADMIN' ? this.router.navigate(['/admin']) : this.router.navigate(['/user']);

      });

  }

}

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AuthService {

  private apiUrl = 'http://localhost:2040/auth';

  constructor(private http: HttpClient) {}

  login(credentials: { username: string; password: string }): Observable<string> {

    return this.http.post(`${this.apiUrl}/login`, credentials, { responseType: 'text' });

  }

  register(user: any): Observable<string> {

    return this.http.post<string>(`${this.apiUrl}/register`, user);

  }

  getUserRole(): string {

    const token = localStorage.getItem('jwtToken');

    console.log("Hello"+ token);

    if (token) {

      const payload = JSON.parse(atob(token.split('.')[1]));

      console.log(payload);

      return payload.roles;

    }

    return '';

  }

}

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class ApiService {

  private baseUrl = 'http://localhost:2040';

  constructor(private http: HttpClient) {}

  getUser(): Observable<string> {

    const headers = new HttpHeaders({

      Authorization: `Bearer ${localStorage.getItem('jwtToken')}`

    });

    return this.http.get<string>(`${this.baseUrl}/user`, { headers });

  }

  getAdmin(): Observable<string> {

    const headers = new HttpHeaders({

      Authorization: `Bearer ${localStorage.getItem('jwtToken')}`

    });

    console.log(headers);

    return this.http.get<string>(`${this.baseUrl}/admin`, { headers });

  }

}

import { Component, OnInit } from '@angular/core';

import { ApiService } from '../services/api.service';

@Component({

  selector: 'app-user',

  templateUrl: './user.component.html'

})

export class UserComponent implements OnInit {

  userMessage: string = '';

  constructor(private apiService: ApiService) {}

  ngOnInit() {

    this.apiService.getUser().subscribe(

      (data) => this.userMessage = data,

      (error) => this.userMessage = 'Error fetching user data'

    );

  }

}

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { LoginComponent } from './login/login.component';

import { AdminComponent } from './admin/admin.component';

import { UserComponent } from './user/user.component';

@NgModule({

  declarations: [

    AppComponent,

    LoginComponent,

    AdminComponent,

    UserComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }