SOURCE CODE

1. Application.java

package com.kitchenstory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

import com.kitchenstory.entities.AdminCred;

import com.kitchenstory.repo.AdminCredRepo;

@SpringBootApplication

public class Phase4KitchenStoryApplication implements CommandLineRunner {

@Autowired

private AdminCredRepo credRepo;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

public static void main(String[] args) {

SpringApplication.run(Phase4KitchenStoryApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

AdminCred cred = new AdminCred();

cred.setUsername("admin@kitchenstory.com");

cred.setPassword(this.passwordEncoder.encode("admin123"));

cred.setRole("ROLE\_ADMIN");

this.credRepo.save(cred);

}

}

1. Customer User details.java

package com.kitchenstory.config;

import java.util.Collection;

import java.util.HashSet;

import org.springframework.security.core.GrantedAuthority;

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

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

import com.kitchenstory.entities.AdminCred;

@SuppressWarnings("serial")

public class CustomUserDetails implements UserDetails{

private AdminCred adminCred;

public CustomUserDetails(AdminCred adminCred) {

this.adminCred = adminCred;

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

HashSet<SimpleGrantedAuthority> set = new HashSet<>();

set.add(new SimpleGrantedAuthority(this.adminCred.getRole()));

return set;

}

@Override

public String getPassword() {

// TODO Auto-generated method stub

return this.adminCred.getPassword();

}

@Override

public String getUsername() {

// TODO Auto-generated method stub

return this.adminCred.getUsername();

}

@Override

public boolean isAccountNonExpired() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isAccountNonLocked() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isCredentialsNonExpired() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isEnabled() {

// TODO Auto-generated method stub

return true;

}

}

1. JwtAuth Entry.java

package com.kitchenstory.config;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.web.AuthenticationEntryPoint;

import org.springframework.stereotype.Component;

@Component

public class JwtAuthEntryPoint implements AuthenticationEntryPoint{

@Override

public void commence(HttpServletRequest request, HttpServletResponse response,

AuthenticationException authException) throws IOException, ServletException {

response.sendError(401,"Unauthorized");

}

}

1. Filter.java

package com.kitchenstory.config;

import java.io.IOException;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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 com.kitchenstory.jwthelper.JwtUtil;

import com.kitchenstory.services.CustomUserDetailService;

@Component

public class JwtAuthFilter extends OncePerRequestFilter {

@Autowired

CustomUserDetailService customUserDetailService;

@Autowired

private JwtUtil jwtUtil;

@Override

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

throws ServletException, IOException {

String requestTokenHeader = request.getHeader("Authorization");

String username = null;

String jwtToken = null;

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

jwtToken = requestTokenHeader.substring(7);

try {

username = this.jwtUtil.extractUsername(jwtToken);

}catch(Exception e) {

e.printStackTrace();

}

UserDetails userDetails = this.customUserDetailService.loadUserByUsername(username);

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

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

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

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}else {

System.out.println("Token is invalid!");

}

}

filterChain.doFilter(request, response);

}

}

1. package com.kitchenstory.config;
2. import org.springframework.beans.factory.annotation.Autowired;
3. import org.springframework.context.annotation.Bean;
4. import org.springframework.context.annotation.Configuration;
5. import org.springframework.security.authentication.AuthenticationManager;
6. import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
7. import org.springframework.security.config.annotation.web.builders.HttpSecurity;
8. import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
9. import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
10. import org.springframework.security.config.http.SessionCreationPolicy;
11. import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
12. import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
13. import com.kitchenstory.services.CustomUserDetailService;
14. @SuppressWarnings("deprecation")
15. @Configuration
16. @EnableWebSecurity
17. public class SecurityConfig extends WebSecurityConfigurerAdapter{
19. @Autowired
20. private CustomUserDetailService customUserDetailService;
22. @Autowired
23. private JwtAuthEntryPoint entryPoint;
25. @Autowired
26. JwtAuthFilter authFilter;
27. @Override
28. protected void configure(HttpSecurity http) throws Exception {
29. http
30. .csrf()
31. .disable()
32. .cors().and()
33. .authorizeRequests()
34. .antMatchers("/token").permitAll()
35. .antMatchers("/changePassword").permitAll()
36. .antMatchers("/getProduct/\*\*").permitAll()
37. .antMatchers("/getItem/\*\*").permitAll()
38. .antMatchers("/order").permitAll()
39. .antMatchers("/getOrder/\*\*").permitAll()
40. .anyRequest().authenticated()
41. .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
42. .and().exceptionHandling().authenticationEntryPoint(entryPoint);
43. http.addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class);
45. }

48. @Override
49. protected void configure(AuthenticationManagerBuilder auth) throws Exception {
50. auth.userDetailsService(this.customUserDetailService).passwordEncoder(passwordEncoder());
51. }
52. @Bean
53. public BCryptPasswordEncoder passwordEncoder() {
54. return new BCryptPasswordEncoder(10);
55. }
57. @Bean
58. public AuthenticationManager authenticationManagerBean() throws Exception {
59. return super.authenticationManagerBean();
60. }
61. }

6. Admin Login Control

package com.kitchenstory.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

import com.kitchenstory.entities.Product;

import com.kitchenstory.services.ProductService;

@RestController

@CrossOrigin(origins = "\*")

public class AdminLoginController {

@Autowired

private ProductService prodService;

@PostMapping("/saveItem")

public ResponseEntity<?> saveItem(@RequestBody Product product){

this.prodService.addProduct(product);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

@GetMapping("/getItems")

public ResponseEntity<?> getItems() {

List<Product> allItems = this.prodService.showProducts();

if(allItems.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(allItems);

}

}

@DeleteMapping("/deleteItem/{name}")

public ResponseEntity<?> deleteItem(@PathVariable("name") String name) {

try {

this.prodService.deleteProduct(name);

return ResponseEntity.status(HttpStatus.OK).build();

}catch(Exception e) {

System.out.println("Empty Result Data Access Exception.");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

}

}

@PutMapping("/updateItem/{name}")

public ResponseEntity<?> updateItem(@PathVariable("name") String name,@RequestBody Product product) {

try {

this.prodService.updateProduct(name, product);

return ResponseEntity.status(HttpStatus.OK).build();

}catch(Exception e) {

System.out.println("Empty Result Data Access Exception.");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

}

}

@GetMapping("/getItem/{name}")

public ResponseEntity<?> getItem(@PathVariable("name") String name){

Product product = this.prodService.getProduct(name);

if(product!=null) {

return ResponseEntity.ok(product);

}else {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

}

}

7. Jwt Controller Java

package com.kitchenstory.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.userdetails.UserDetails;

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

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

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;

import com.kitchenstory.entities.JwtRequest;

import com.kitchenstory.entities.JwtResponse;

import com.kitchenstory.jwthelper.JwtUtil;

import com.kitchenstory.services.CustomUserDetailService;

@RestController

@CrossOrigin(origins = "\*")

public class JwtController {

@Autowired

AuthenticationManager authenticationManager;

@Autowired

CustomUserDetailService customUserDetailService;

@Autowired

JwtUtil jwtUtil;

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

public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest){

System.out.println(jwtRequest);

try {

this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(jwtRequest.getUsername(), jwtRequest.getPassword()));

}catch(UsernameNotFoundException e) {

e.printStackTrace();

}

UserDetails userDetails = this.customUserDetailService.loadUserByUsername(jwtRequest.getUsername());

String token = this.jwtUtil.generateToken(userDetails);

System.out.println("Token: "+token);

return ResponseEntity.ok(new JwtResponse(token));

}

}

8. SignIn controller java

package com.kitchenstory.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

import com.kitchenstory.entities.AdminCred;

import com.kitchenstory.services.AdminService;

@RestController

@CrossOrigin(origins = "\*")

public class SignInController {

@Autowired

private AdminService adminService;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

@PutMapping("/changePassword")

public ResponseEntity<?> setPassword(@RequestBody AdminCred admin) {

AdminCred adm = this.adminService.findAdmin(admin.getUsername());

if(adm == null) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

else {

adm.setPassword(this.passwordEncoder.encode(admin.getPassword()));

this.adminService.saveCred(adm);

return ResponseEntity.ok(adm);

}

}

}

9. User Controller java

package com.kitchenstory.controller;

import java.text.DateFormat;

import java.util.Calendar;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

import com.kitchenstory.entities.Order;

import com.kitchenstory.entities.Product;

import com.kitchenstory.services.OrderService;

import com.kitchenstory.services.ProductService;

@RestController

@CrossOrigin(origins = "\*")

public class UserController {

@Autowired

private OrderService service;

@Autowired

private ProductService pservice;

@PostMapping("/order")

public ResponseEntity<?> createOrder(@RequestBody Order order){

DateFormat df = DateFormat.getDateInstance();

Calendar cl = Calendar.getInstance();

String od = df.format(cl.getTime());

order.setOd(od);

this.service.saveOrder(order);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

@GetMapping("/getOrders")

public ResponseEntity<?> getAllOrders(){

List<Order> orders = this.service.getOrders();

if(orders.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

else {

return ResponseEntity.ok(orders);

}

}

@DeleteMapping("/delete/{id}")

public ResponseEntity<?> deleteOrder(@PathVariable("id") String id){

this.service.deleteOrder(id);

return ResponseEntity.status(HttpStatus.OK).build();

}

@GetMapping("/getProduct/{name}")

public ResponseEntity<?> getProducts(@PathVariable("name") String name){

List<Product> products = this.pservice.getAllByName(name);

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

@GetMapping("/getOrder/{id}")

public ResponseEntity<?> getOrder(@PathVariable("id") String id){

Order order = this.service.getOrder(id);

if(order==null) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(order);

}

}

}

10. Admin Cred java

package com.kitchenstory.entities;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "AdminCred")

public class AdminCred {

@Id

private String username;

private String password;

private String role;

public AdminCred() {

super();

// TODO Auto-generated constructor stub

}

public AdminCred(String username, String password, String role) {

super();

this.username = username;

this.password = password;

this.role = role;

}

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;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

11. App component.t

import { TestBed } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

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

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [

        RouterTestingModule

      ],

      declarations: [

        AppComponent

      ],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have as title 'Kitchen-Story'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('Kitchen-Story');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement as HTMLElement;

    expect(compiled.querySelector('.content span')?.textContent).toContain('Kitchen-Story app is running!');

  });

});

12. app

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'Kitchen-Story';

}

13. Index .html

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>KitchenStory</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"

    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

</head>

<body>

  <app-root></app-root>

  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

    integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"

    crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"

    integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"

    crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"

    integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"

    crossorigin="anonymous"></script>

</body>

</html>