Kitchen Story
(Project Source Code)

Version History:

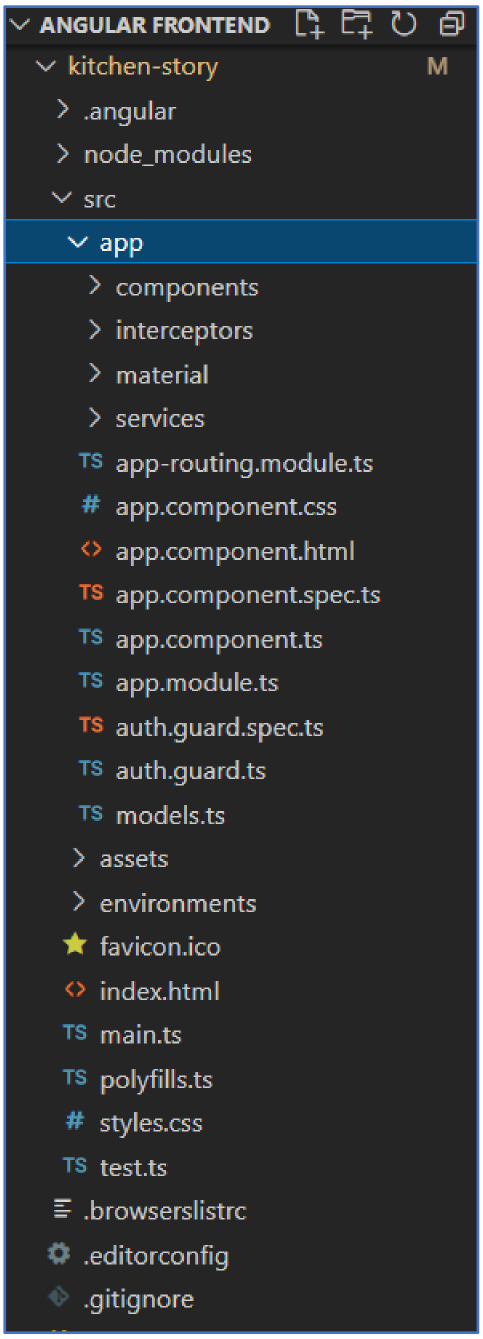
Author	Nikhil Jain
Purpose	Source Code of the application
Date	10th December 2021
Version	1.0

Table of Contents

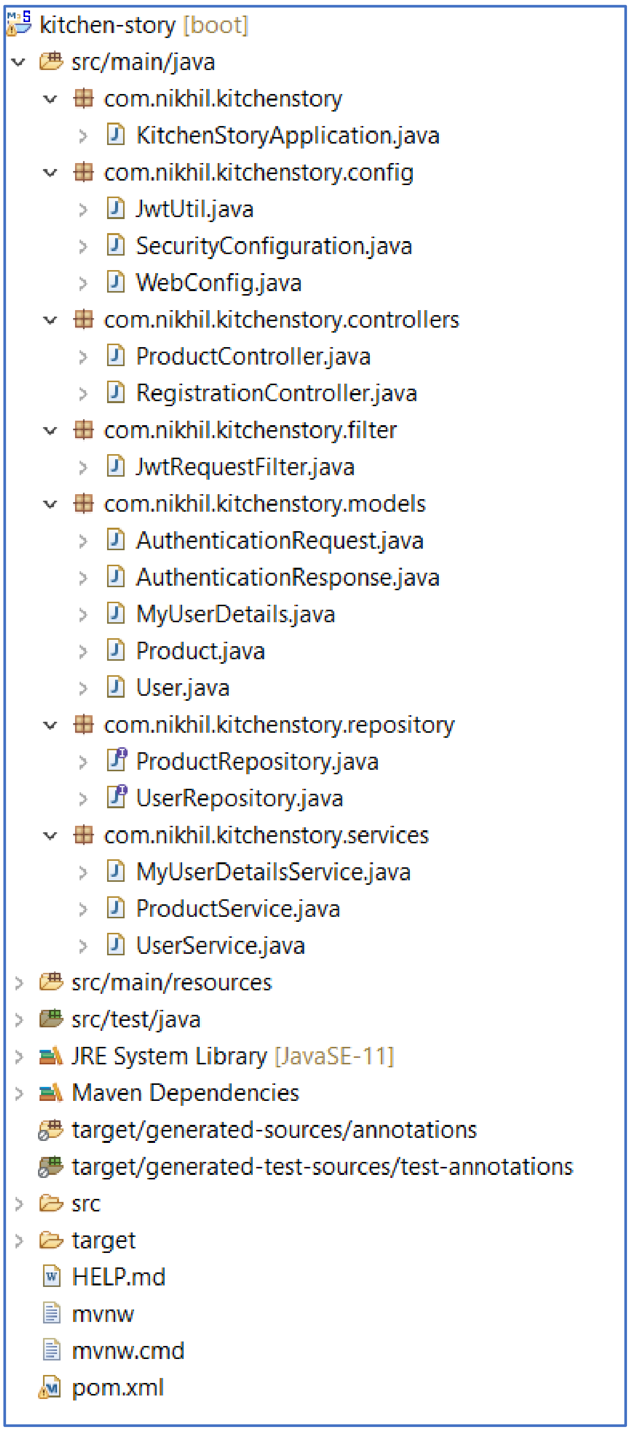
- ## Project Link

Repository Name	**Kitchen Story**
GitHub Link	https://github.com/Niks4u2/KitchenStory
Deployed On	http://kitchen-story-angular.herokuapp.com/home

- Folder Structure



```
ANGULAR FRONTEND
  kitchen-story                M
    .angular
    node_modules
    src
      app
        components
        interceptors
        material
        services
        app-routing.module.ts
        app.component.css
        app.component.html
        app.component.spec.ts
        app.component.ts
        app.module.ts
        auth.guard.spec.ts
        auth.guard.ts
        models.ts
      assets
      environments
      favicon.ico
      index.html
      main.ts
      polyfills.ts
      styles.css
      test.ts
    .browserslistrc
    .editorconfig
    .gitignore
```



```
kitchen-story [boot]
  src/main/java
    com.nikhil.kitchenstory
      KitchenStoryApplication.java
    com.nikhil.kitchenstory.config
      JwtUtil.java
      SecurityConfiguration.java
      WebConfig.java
    com.nikhil.kitchenstory.controllers
      ProductController.java
      RegistrationController.java
    com.nikhil.kitchenstory.filter
      JwtRequestFilter.java
    com.nikhil.kitchenstory.models
      AuthenticationRequest.java
      AuthenticationResponse.java
      MyUserDetails.java
      Product.java
      User.java
    com.nikhil.kitchenstory.repository
      ProductRepository.java
      UserRepository.java
    com.nikhil.kitchenstory.services
      MyUserDetailsService.java
      ProductService.java
      UserService.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

- ## Source Code

- ## Configuration

1. SecurityConfiguration.java

```
package com.nikhil.kitchenstory.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.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
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.UsernamePasswordAuthenticationFi
lter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import com.nikhil.kitchenstory.filter.JwtRequestFilter;

@SuppressWarnings("deprecation")
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
	@Autowired
	UserDetailsService userDetailsService;

	@Autowired
	private JwtRequestFilter jwtRequestFilter;

	@Bean
	public PasswordEncoder passwordEncoder() {
	    return ~~NoOpPasswordEncoder~~.~~getInstance~~();
	}
	@Bean
	public DaoAuthenticationProvider authenticationProvider()
	{
		DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
		auth.setUserDetailsService(userDetailsService);
		auth.setPasswordEncoder(passwordEncoder());
		return auth;
	}

	@Override
	@Bean
	public AuthenticationManager authenticationManagerBean() throws Exception
{
		return super.authenticationManagerBean();
	}

	// authentication
	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
		auth.authenticationProvider(authenticationProvider());
	}

	// authorization
	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http.cors().and().csrf().disable()
		.authorizeRequests().antMatchers("/authenticate", "/registration",
"/products").permitAll().
					anyRequest().authenticated()

	.and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.ST
ATELESS);
		http.addFilterBefore(jwtRequestFilter,
UsernamePasswordAuthenticationFilter.class);
	}

    @Bean
    CorsConfigurationSource corsConfigurationSource()
    {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOriginPatterns(Arrays.asList("*"));
        configuration.setAllowedMethods(Arrays.asList("HEAD", "GET", "POST",
"PUT", "DELETE", "PATCH"));
        configuration.setAllowCredentials(true);
        configuration.setAllowedHeaders(Arrays.asList("Authorization", "Cache-
Control", "Content-Type"));
        final UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }

}
```

2. JwtUtil.java

```java
package com.nikhil.kitchenstory.config;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtUtil {
    private String SECRET_KEY = "secret";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }
    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }
    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver)
{
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }
    private Claims extractAllClaims(String token) {
        return
Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }
    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }
    private String createToken(Map<String, Object> claims, String subject) {

        return
Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 *
60 * 48))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }
    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) &&
!isTokenExpired(token));
    }
}
```

- ## Controllers

1. ProductController.java

```
package com.nikhil.kitchenstory.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.nikhil.kitchenstory.models.Product;
import com.nikhil.kitchenstory.services.ProductService;

@RestController
@CrossOrigin(origins = "http://localhost:4200")
@RequestMapping("/products")
public class ProductController
{
	@Autowired
	private ProductService service;

	@PostMapping
	public String addProduct(@RequestBody Product product)
	{
		service.addProduct(product);
		return "Product add success.";
	}

	@GetMapping
	public List<Product> getProducts()
	{
		List<Product> productlist = service.getAllProducts();
		return productlist;
	}

	@DeleteMapping("/delete/{prodId}")
	public String deleteProduct(@PathVariable("prodId") String prodId) {
		service.deleteProductById(prodId);
		return "Product delete success.";
	}
}
```

2. RegistrationController.java

```java
package com.nikhil.kitchenstory.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import 
org.springframework.security.authentication.BadCredentialsException;
import 
org.springframework.security.authentication.UsernamePasswordAuthenticatio
nToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.CrossOrigin;
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.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.nikhil.kitchenstory.config.JwtUtil;
import com.nikhil.kitchenstory.models.AuthenticationRequest;
import com.nikhil.kitchenstory.models.AuthenticationResponse;
import com.nikhil.kitchenstory.models.User;
import com.nikhil.kitchenstory.services.MyUserDetailsService;
import com.nikhil.kitchenstory.services.UserService;

@RestController
@CrossOrigin(origins = "http://localhost:4200")
public class RegistrationController {

	@Autowired
	private UserService service;

	@Autowired
	private AuthenticationManager authenticationManager;

	@Autowired
	private JwtUtil jwtTokenUtil;

	@Autowired
	private MyUserDetailsService userDetailsService;

	@PostMapping("/registration")
	public void registerUserAccount(@RequestBody User user)
	{
		user.setRoles("USER");
		service.registerUser(user);
	}

	@RequestMapping(value = "/authenticate", method = 
RequestMethod.POST)
	public ResponseEntity<?> createAuthenticationToken(@RequestBody 
AuthenticationRequest authenticationRequest) throws Exception {
		try {
			authenticationManager.authenticate(
					new 
UsernamePasswordAuthenticationToken(authenticationRequest.getEmail(), 
authenticationRequest.getPassword())
			);
		}
		catch (BadCredentialsException e) {
			throw new Exception("Incorrect username or password", 
e);
		}


		final UserDetails userDetails = userDetailsService

	.loadUserByUsername(authenticationRequest.getEmail());

		final String jwt = jwtTokenUtil.generateToken(userDetails);

		return ResponseEntity.ok(new AuthenticationResponse(jwt));
	}
}
```

- ## Filters

1. JwtRequestFilter.java

```
package com.nikhil.kitchenstory.filter;

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.nikhil.kitchenstory.config.JwtUtil;
import com.nikhil.kitchenstory.services.MyUserDetailsService;

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(usernamePasswordAuthenticat
ionToken);
            }
        }
        chain.doFilter(request, response);
    }
}
```

- ## Models

1. Product.java

```
package com.nikhil.kitchenstory.models;

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

@Document(collection = "products")
public class Product {

	@Id
	private String id;
	private String productName;
	private double price;
	private String description;
	private String image;

	public Product() {}

	public Product(String productName, double price, String description,
String image) {
		this.productName = productName;
		this.price = price;
		this.description = description;
		this.image = image;
	}

	public String getImage() {
		return image;
	}

	public void setImage(String image) {
		this.image = image;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getProductName() {
		return productName;
	}

	public void setProductName(String productName) {
		this.productName = productName;
	}

	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}
}
```

2. User.java

```
package com.nikhil.kitchenstory.models;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.index.Indexed;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "users")
public class User {
	@Id
	private String id;
	private String firstName;
	private String lastName;
	@Indexed(unique = true)
	private String email;
	private String password;
	private String roles;

	public User() {};

	public User(String firstName, String lastName, String email, String 
password) {
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
		this.password = password;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getRoles() {
		return roles;
	}

	public void setRoles(String roles) {
		this.roles = roles;
	}
}
```