LITHAN

EDU CLaaS®
DIGITAL ACCELERATOR

Student Name/ID Number:	Francis Roel L. Abarca | bdse-0922-113
Academic Year:	2022 – 2023
Unit Assessor:	Archana Sakpal
Project Title:	Assignment5 – Spring Security, Error Handling, and Logging
Issue Date:	4/18/2023
Submission Date:	4/27/2023
Internal Verifier Name:	Archana Sakpal
Date:	4/26/2023

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.
Student signature: [illegible] Date: 4/26/2023

This assignment is about implementing Authentication & Authorization solution for Know-Your-Neighbourhood application.

Spring Security

1. Create user, role and user_role tables in the database.

 a. [Note: The words user & role are reserved in some databases. Use some other name, if so.]

Figure 1: Know Your Neighborhood database, kyn

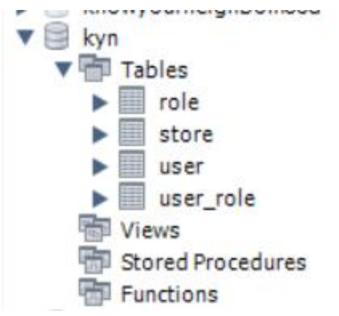


Figure 2: user Table

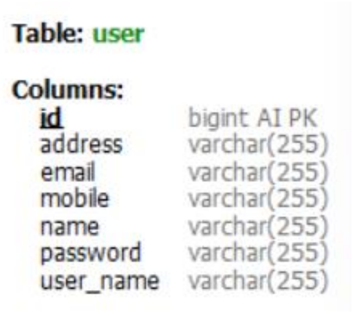


Table: user

Columns:

id	bigint AI PK
address	varchar(255)
email	varchar(255)
mobile	varchar(255)
name	varchar(255)
password	varchar(255)
user_name	varchar(255)

Figure 3: role Table

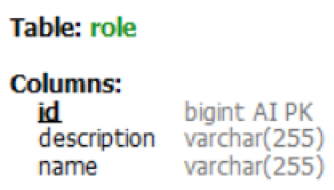
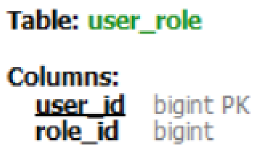


Table: role

Columns:

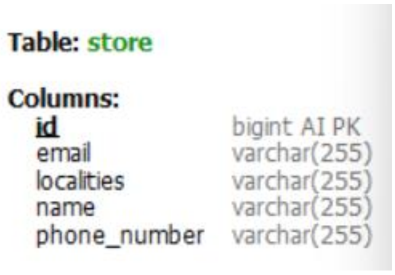
id	bigint AI PK
description	varchar(255)
name	varchar(255)

Figure 4: user_role Table



```
Table: user_role

Columns:
   user_id    bigint PK
   role_id    bigint
```

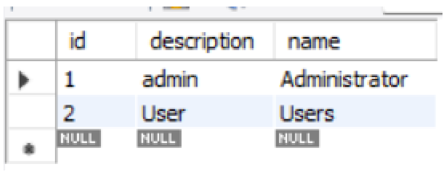
Figure 5: store Table



```
Table: store

Columns:
   id              bigint AI PK
   email           varchar(255)
   localities      varchar(255)
   name            varchar(255)
   phone_number    varchar(255)
```

2. Add entries into user, role(roleid, role name) and user_role tables.
    a. Add two entries into role table. One role is to view the store info (VIEW_STORE) and the other role is to add/modify store info (ADD_STORE).

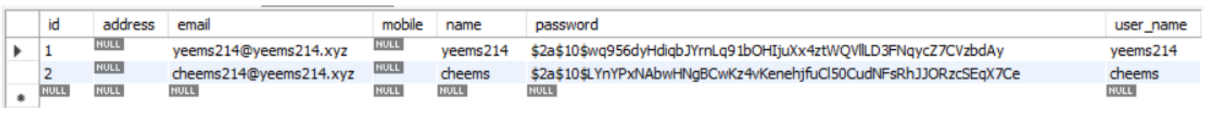
Figure 6: role Table



id	description	name
1	admin	Administrator
2	User	Users
NULL	NULL	NULL

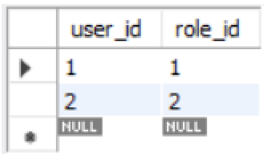
    b. Add two entries into user table. Let one user have play VIEW_STORE role, another user play only both VIEW_STORE & ADD_STORE roles.

Figure 7: user Table



id	address	email	mobile	name	password	user_name
1	NULL	yeems214@yeems214.xyz	NULL	yeems214	$2a$10$wq956dyHdiqbJYrnLq91bOHIjuXx4ztWQVllLD3FNqycZ7CVzbdAy	yeems214
2	NULL	cheems214@yeems214.xyz	NULL	cheems	$2a$10$LYnYPxNAbwHNgBCwKz4vKenehjfuCl50CudNFsRhJJORzcSEqX7Ce	cheems
NULL	NULL	NULL	NULL	NULL	NULL	NULL

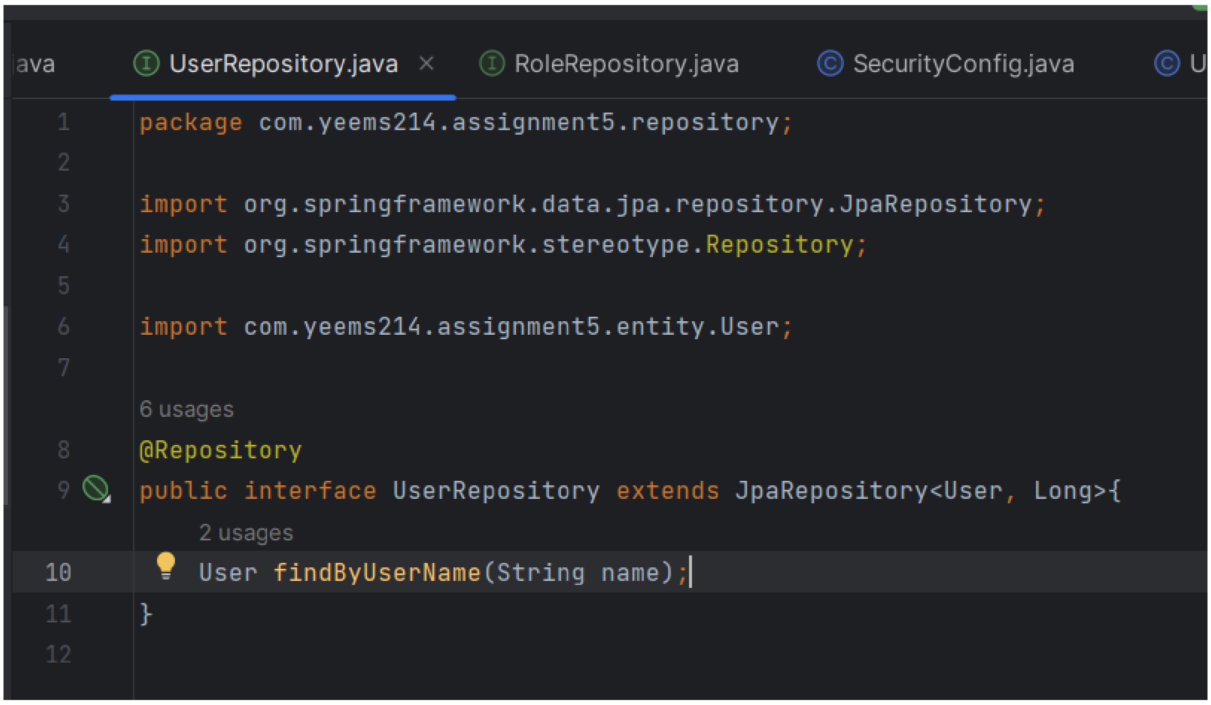
Figure 8: user_role Table



user_id	role_id
1	1
2	2
NULL	NULL

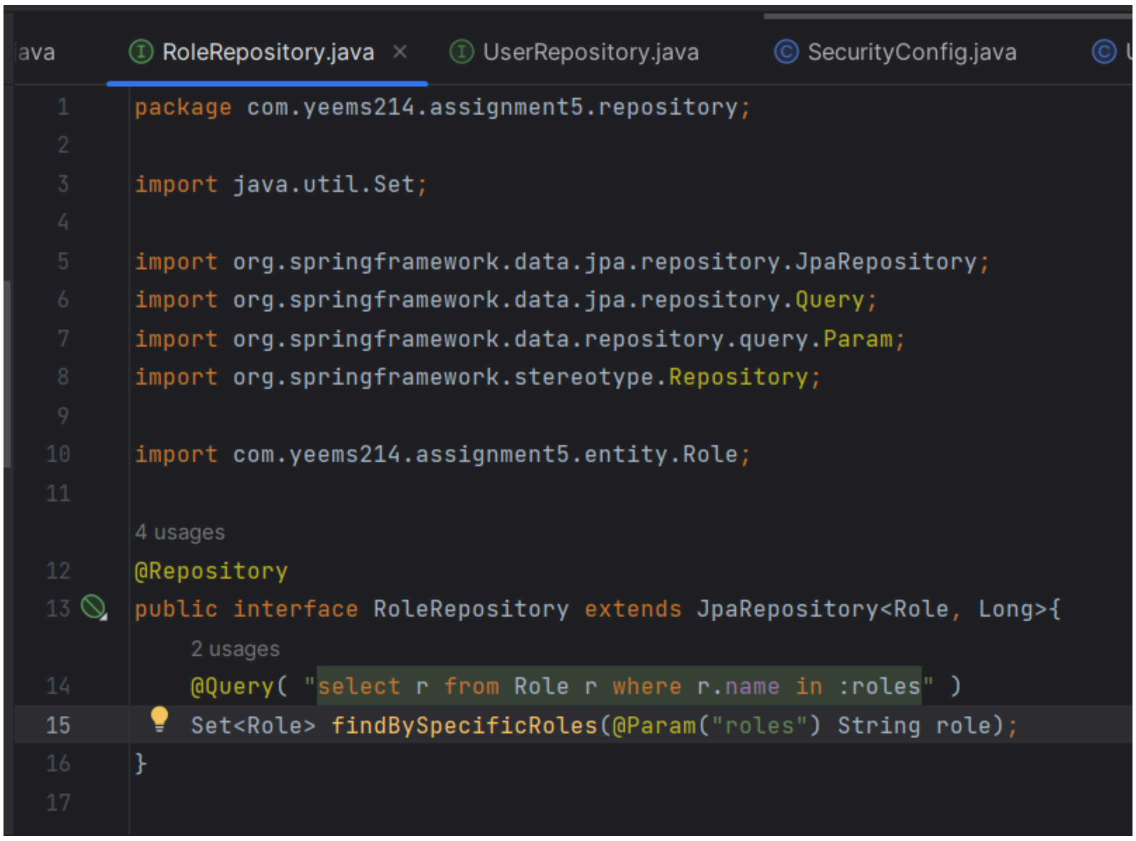
3. Enhance Spring Data JPA enabled version of 'Know-Your-Neighbourhood' application to support authentication and authorization

Figure 9: UserReporitory identifies the user by its user_name



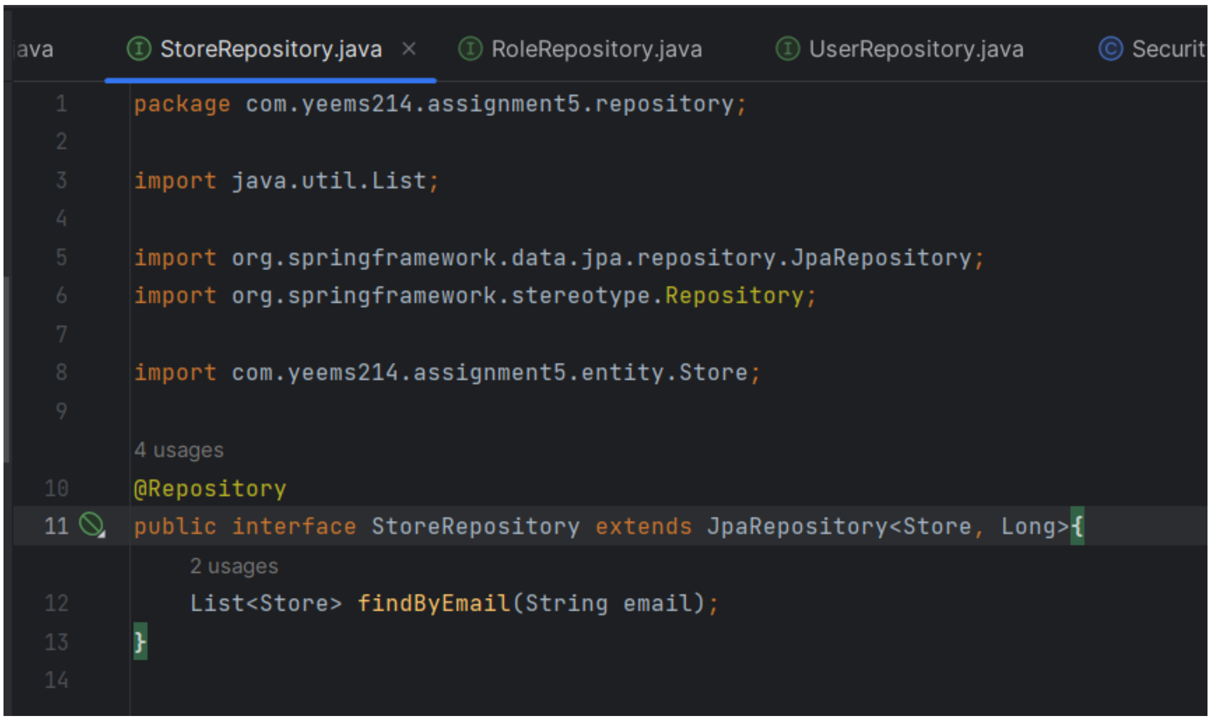
```
package com.yeems214.assignment5.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.yeems214.assignment5.entity.User;

6 usages
@Repository
public interface UserRepository extends JpaRepository<User, Long>{
    2 usages
    User findByUserName(String name);
}
```

Figure 9: RoleRepository



```
package com.yeems214.assignment5.repository;

import java.util.Set;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.yeems214.assignment5.entity.Role;

4 usages
@Repository
public interface RoleRepository extends JpaRepository<Role, Long>{
    2 usages
    @Query( "select r from Role r where r.name in :roles" )
    Set<Role> findBySpecificRoles(@Param("roles") String role);
}
```

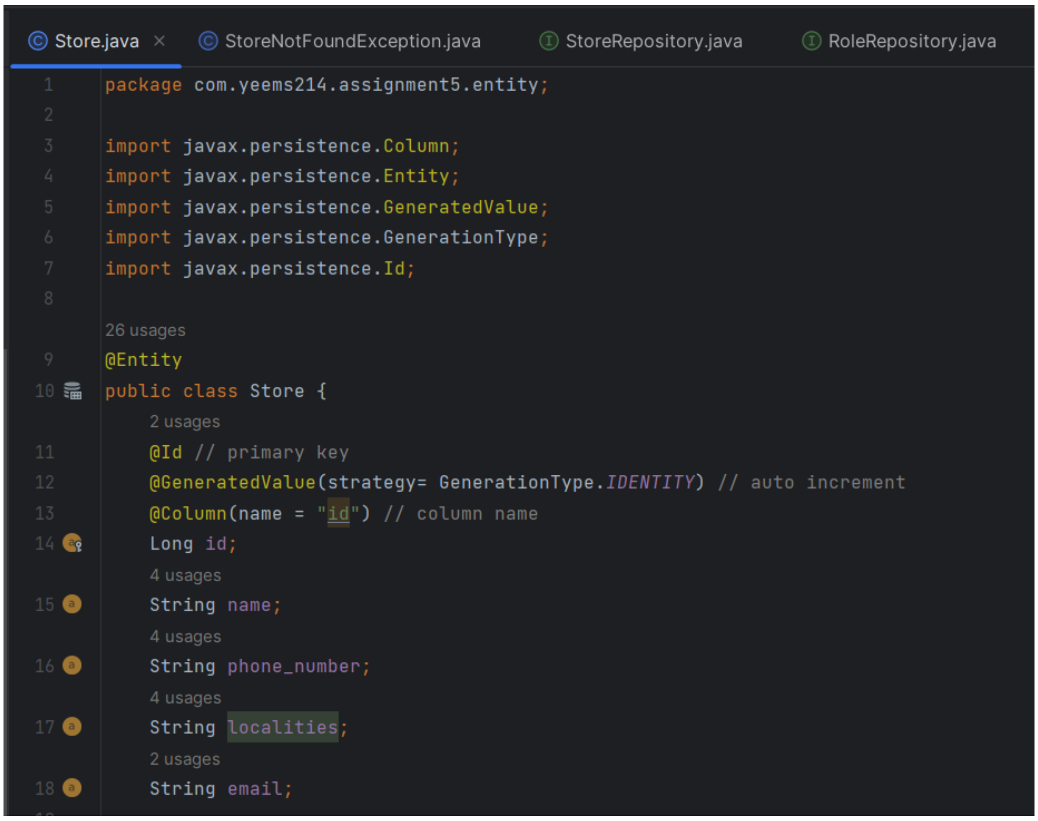
Figure 10: StoreRepository finds the store created by email.



```java
package com.yeems214.assignment5.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.yeems214.assignment5.entity.Store;

4 usages
@Repository
public interface StoreRepository extends JpaRepository<Store, Long>{
    2 usages
    List<Store> findByEmail(String email);
}
```

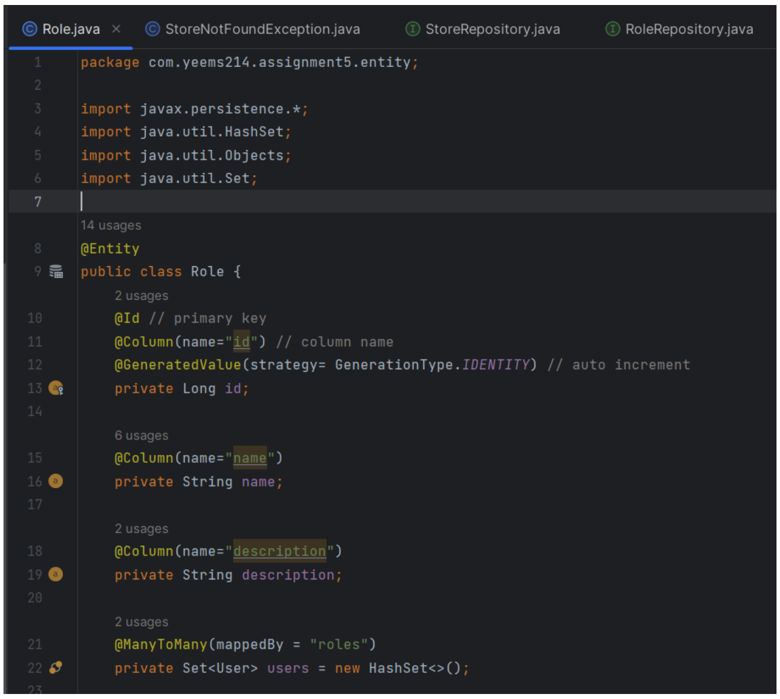
4. **Authentication**:
    a. Create entities for Role and Store.

Figure 11: Store.java



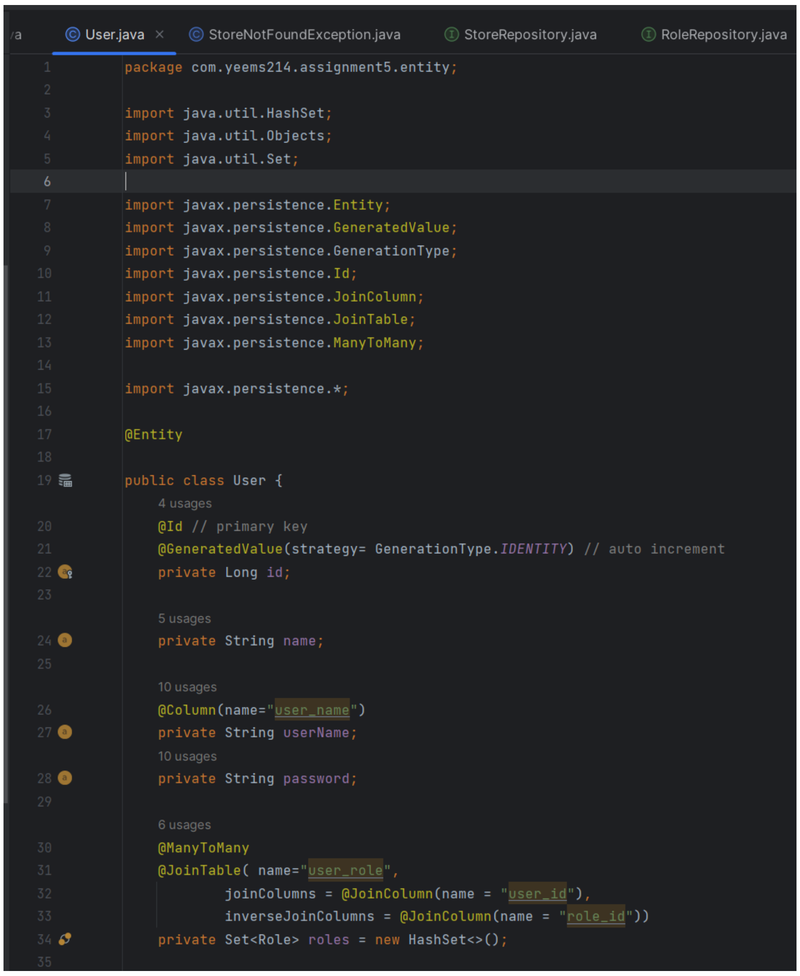
```java
package com.yeems214.assignment5.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

26 usages
@Entity
public class Store {
    2 usages
    @Id // primary key
    @GeneratedValue(strategy= GenerationType.IDENTITY) // auto increment
    @Column(name = "id") // column name
    Long id;
    4 usages
    String name;
    4 usages
    String phone_number;
    4 usages
    String localities;
    2 usages
    String email;
```

Figure 12: Role.java



```
package com.yeems214.assignment5.entity;

import javax.persistence.*;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

14 usages
@Entity
public class Role {
    2 usages
    @Id // primary key
    @Column(name="id") // column name
    @GeneratedValue(strategy= GenerationType.IDENTITY) // auto increment
    private Long id;

    6 usages
    @Column(name="name")
    private String name;

    2 usages
    @Column(name="description")
    private String description;

    2 usages
    @ManyToMany(mappedBy = "roles")
    private Set<User> users = new HashSet<>();
```

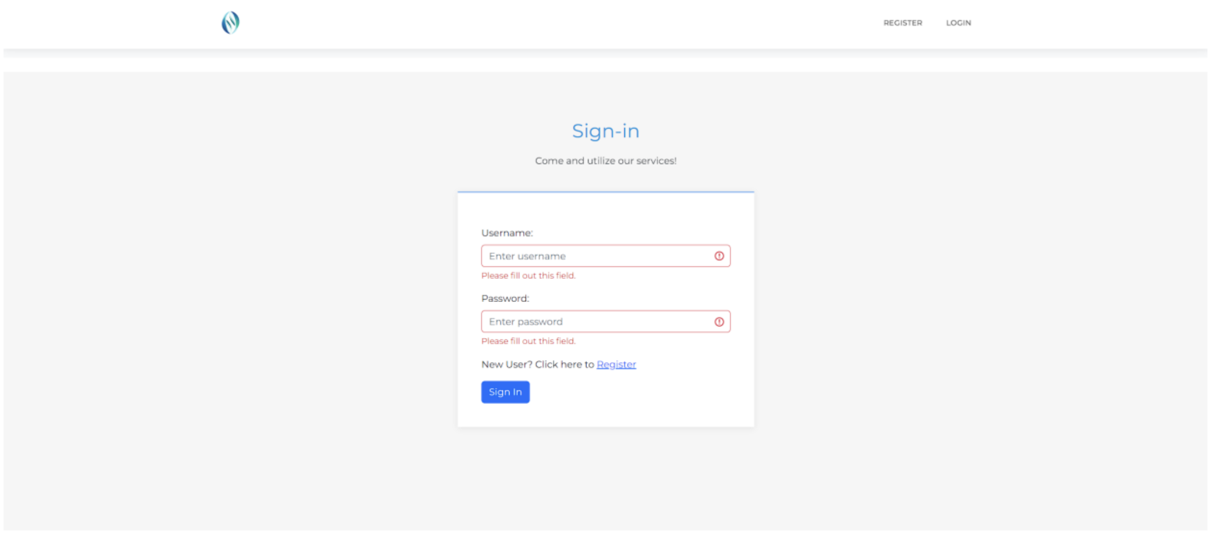
Figure 13: User.java



```
package com.yeems214.assignment5.entity;

import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

import javax.persistence.*;

@Entity

public class User {
    4 usages
    @Id // primary key
    @GeneratedValue(strategy= GenerationType.IDENTITY) // auto increment
    private Long id;

    5 usages
    private String name;

    10 usages
    @Column(name="user_name")
    private String userName;
    10 usages
    private String password;

    6 usages
    @ManyToMany
    @JoinTable( name="user_role",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();
```

b. Create a login page for the application. User should be forced to login before visiting any of the application page.

Figure 14: login.jsp page


REGISTER LOGIN

Sign-in

Come and utilize our services!

Username:

Enter username

Please fill out this field.

Password:

Enter password

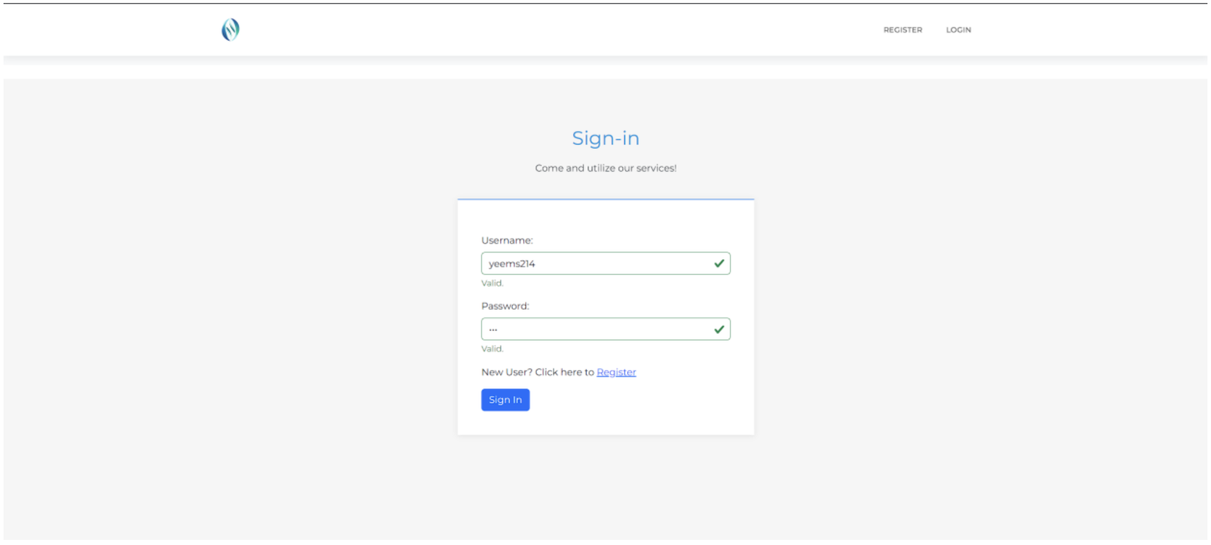
Please fill out this field.

New User? Click here to Register

Sign In

c. Authenticate user with the credentials entered against the database entries added in the previous step.

Figure 15: login.jsp with credentials


REGISTER LOGIN

Sign-in

Come and utilize our services!

Username:

yeems214

Valid.

Password:

...

Valid.

New User? Click here to Register

Sign In

Figure 16: index.jsp for user

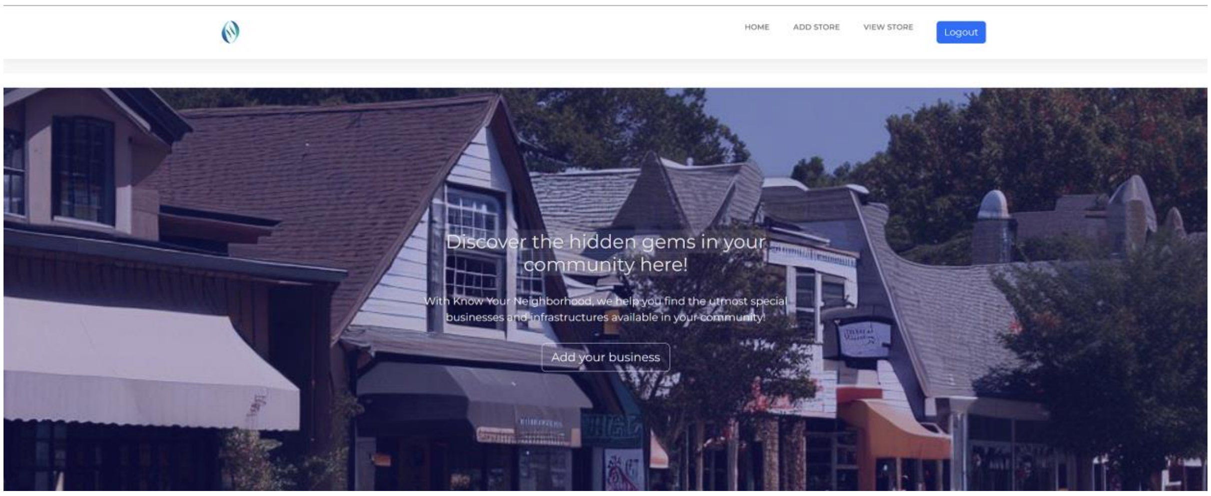

HOME
ADD STORE
VIEW STORE
Logout
Discover the hidden gems in your community here!
With Know Your Neighborhood, we help you find the utmost special businesses and infrastructures available in your community!
Add your business

Figure 17: Standard user can only access View Store thus selecting Add your business redirects to an Access Denied page

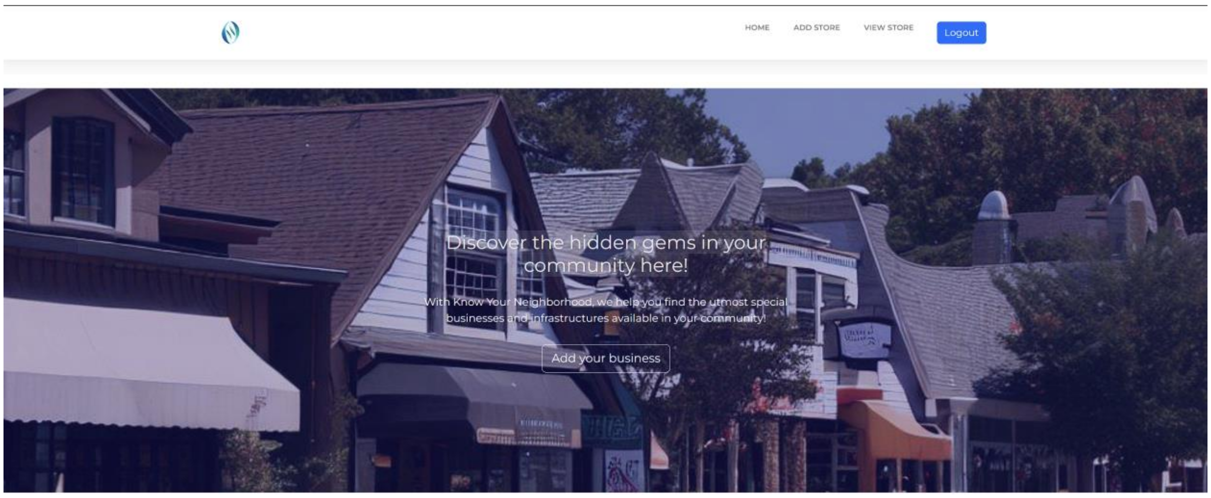
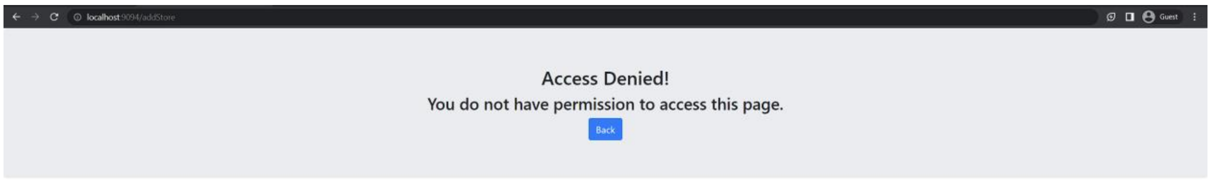

HOME
ADD STORE
VIEW STORE
Logout
Discover the hidden gems in your community here!
With Know Your Neighborhood, we help you find the utmost special businesses and infrastructures available in your community!
Add your business

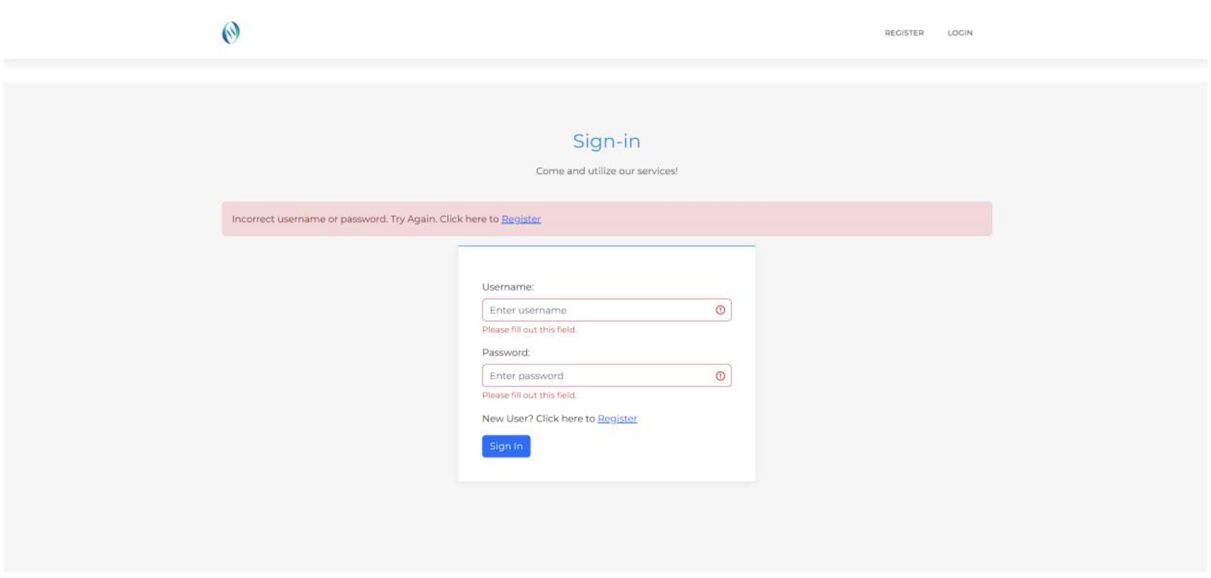
Figure 18: Access Denied page seen by the standard user



Access Denied!
You do not have permission to access this page.
Back

d. Show a message if user enters invalid user and password. And then allow him to re-enter the credentials.

Figure 19: Login error shown after user enters incorrect credentials



REGISTER LOGIN

Sign-in

Come and utilize our services!

Incorrect username or password. Try Again. Click here to Register

Username:

Enter username

Please fill out this field.

Password:

Enter password

Please fill out this field.

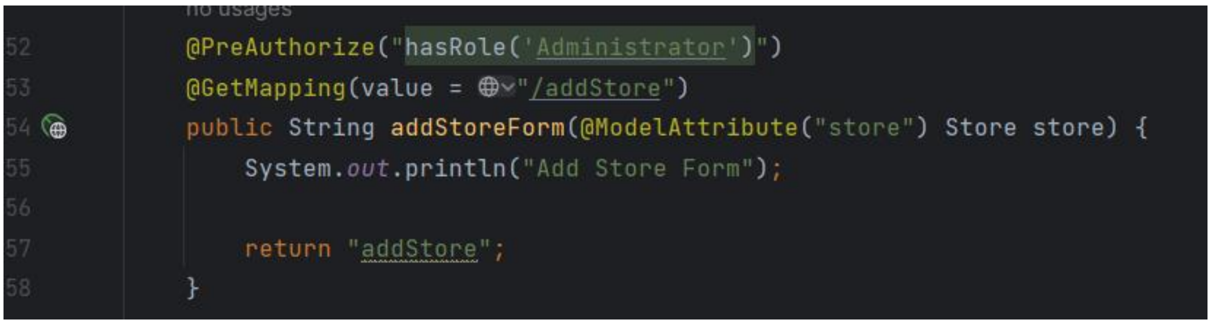
New User? Click here to Register

Sign In

5. **Authorization:**

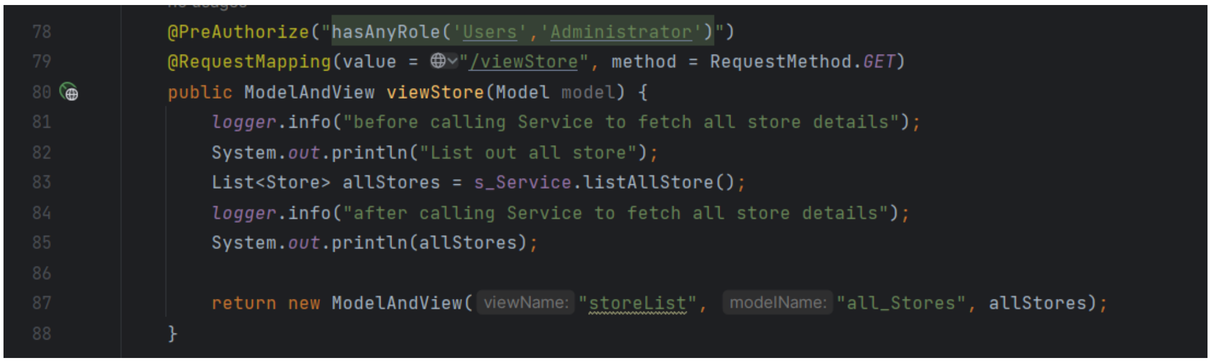
   a. Once the user logged in, make sure that only the user with ADD_STORE role can add/modify store data. The user with VIEW_STORE data can only view the stores entered.

Figure 20: StoreController.java authorizing the Administrator role for Add_Store access



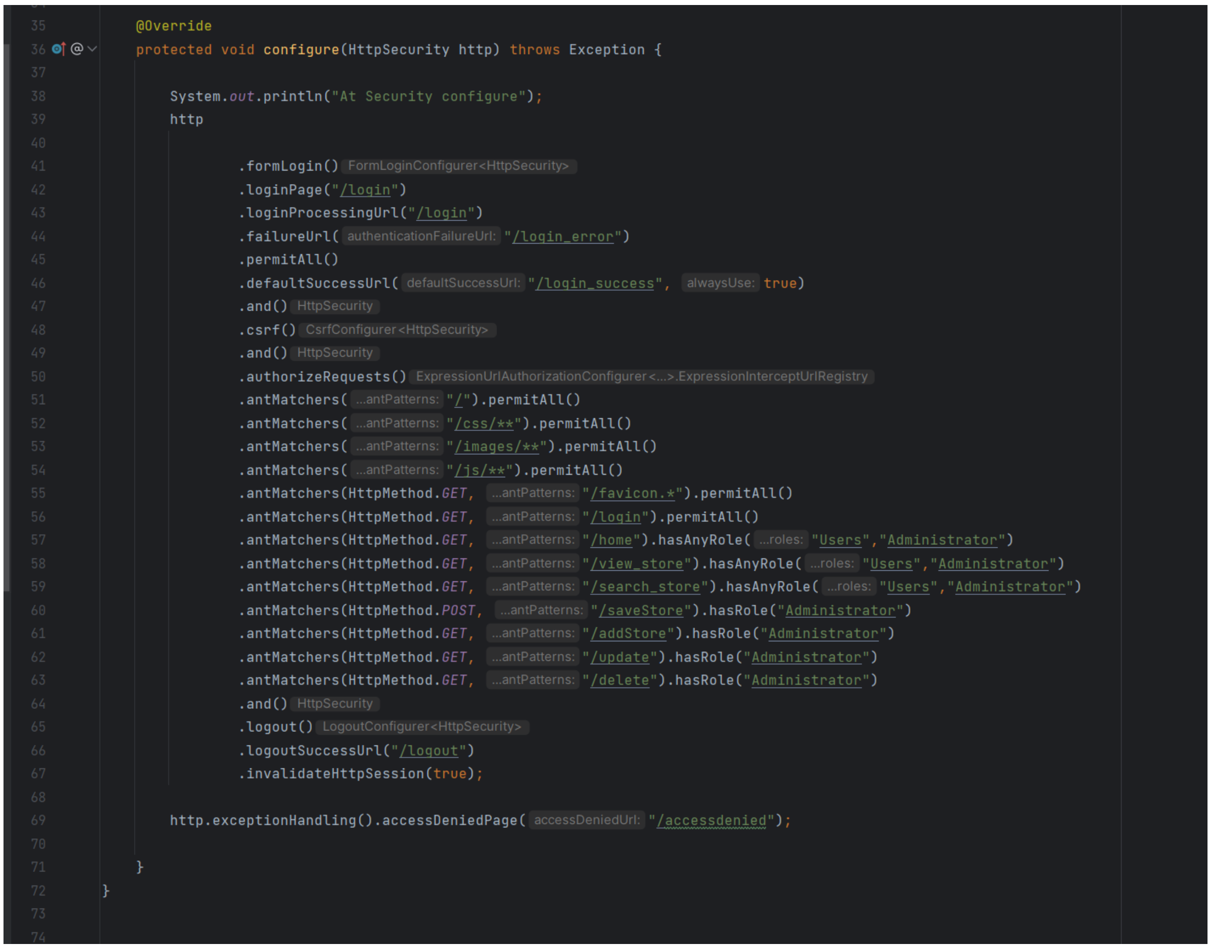
```
no usages
@PreAuthorize("hasRole('Administrator')")
@GetMapping(value = "/addStore")
public String addStoreForm(@ModelAttribute("store") Store store) {
    System.out.println("Add Store Form");

    return "addStore";
}
```

Figure 21: StoreController.java authorizing the User role for View_Store while Administrator role for View_Store and Add_Store



```java
    @PreAuthorize("hasAnyRole('Users','Administrator')")
    @RequestMapping(value = "/viewStore", method = RequestMethod.GET)
    public ModelAndView viewStore(Model model) {
        logger.info("before calling Service to fetch all store details");
        System.out.println("List out all store");
        List<Store> allStores = s_Service.listAllStore();
        logger.info("after calling Service to fetch all store details");
        System.out.println(allStores);

        return new ModelAndView( viewName: "storeList",  modelName: "all_Stores", allStores);
    }
```

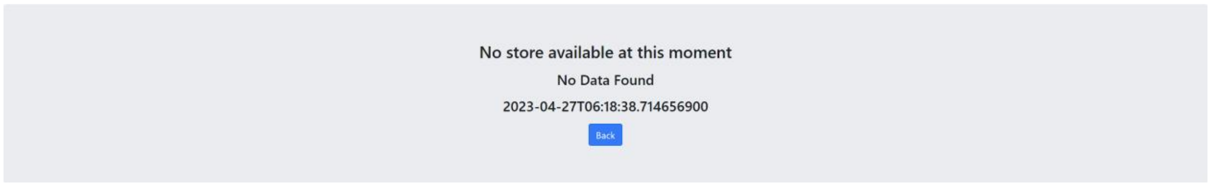
Figure 22: SecurityConfig.java authorizing specific URLs based on the available roles.



```java
    @Override
    protected void configure(HttpSecurity http) throws Exception {

        System.out.println("At Security configure");
        http

                .formLogin() FormLoginConfigurer<HttpSecurity>
                .loginPage("/login")
                .loginProcessingUrl("/login")
                .failureUrl( authenticationFailureUrl: "/login_error")
                .permitAll()
                .defaultSuccessUrl( defaultSuccessUrl: "/login_success",  alwaysUse: true)
                .and() HttpSecurity
                .csrf() CsrfConfigurer<HttpSecurity>
                .and() HttpSecurity
                .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .antMatchers( ...antPatterns: "/").permitAll()
                .antMatchers( ...antPatterns: "/css/**").permitAll()
                .antMatchers( ...antPatterns: "/images/**").permitAll()
                .antMatchers( ...antPatterns: "/js/**").permitAll()
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/favicon.*").permitAll()
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/login").permitAll()
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/home").hasAnyRole( ...roles: "Users","Administrator")
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/view_store").hasAnyRole( ...roles: "Users","Administrator")
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/search_store").hasAnyRole( ...roles: "Users","Administrator")
                .antMatchers(HttpMethod.POST,  ...antPatterns: "/saveStore").hasRole("Administrator")
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/addStore").hasRole("Administrator")
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/update").hasRole("Administrator")
                .antMatchers(HttpMethod.GET,  ...antPatterns: "/delete").hasRole("Administrator")
                .and() HttpSecurity
                .logout() LogoutConfigurer<HttpSecurity>
                .logoutSuccessUrl("/logout")
                .invalidateHttpSession(true);

        http.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/accessdenied");

    }
}
```

b. If there are no stores in the system, show appropriate message to the user.

Figure 23: StoreNotFoundException shown when there are no stores available in the store table

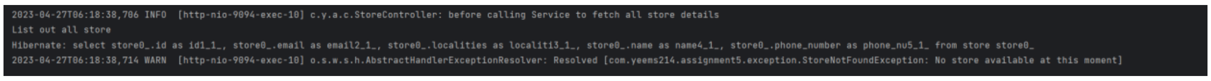


No store available at this moment

No Data Found

2023-04-27T06:18:38.714656900

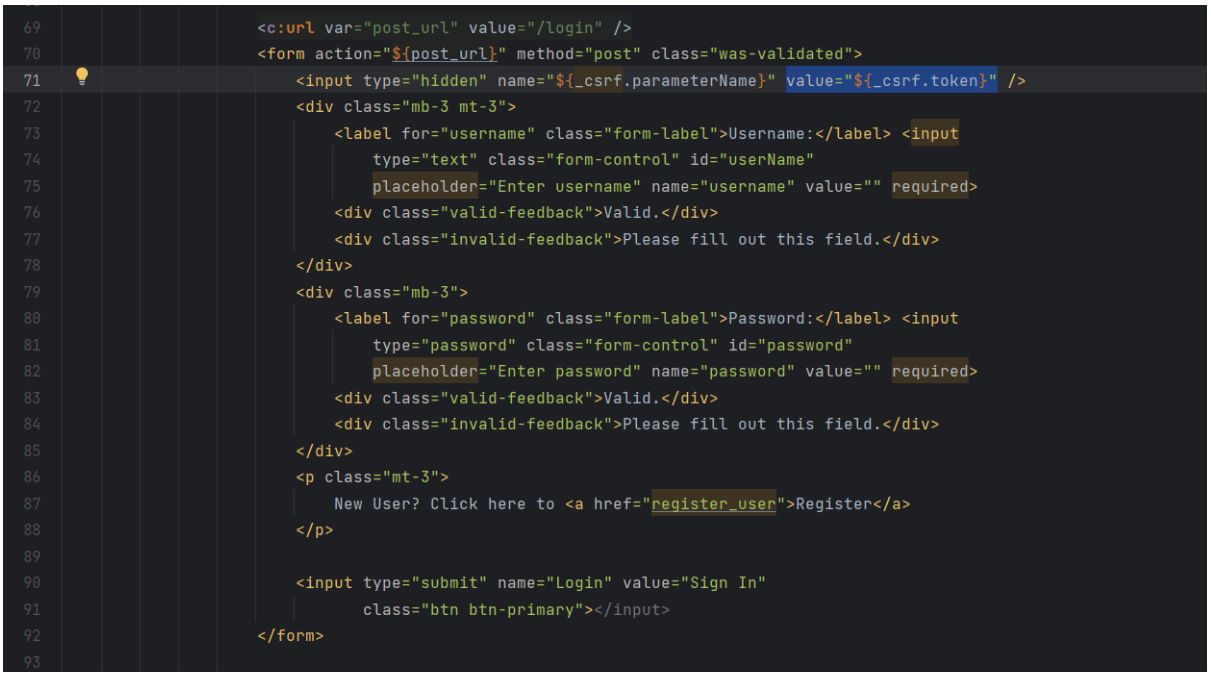
Figure 24: StoreNotFoundException shown in console



```
2023-04-27T06:18:38,706 INFO  [http-nio-9094-exec-10] c.y.a.c.StoreController: before calling Service to fetch all store details
List out all store
Hibernate: select store0_.id as id1_1_, store0_.email as email2_1_, store0_.localities as localiti3_1_, store0_.name as name4_1_, store0_.phone_number as phone_nu5_1_ from store store0_
2023-04-27T06:18:38,714 WARN  [http-nio-9094-exec-10] o.s.w.s.h.AbstractHandlerExceptionResolver: Resolved [com.yeems214.assignment5.exception.StoreNotFoundException: No store available at this moment]
```

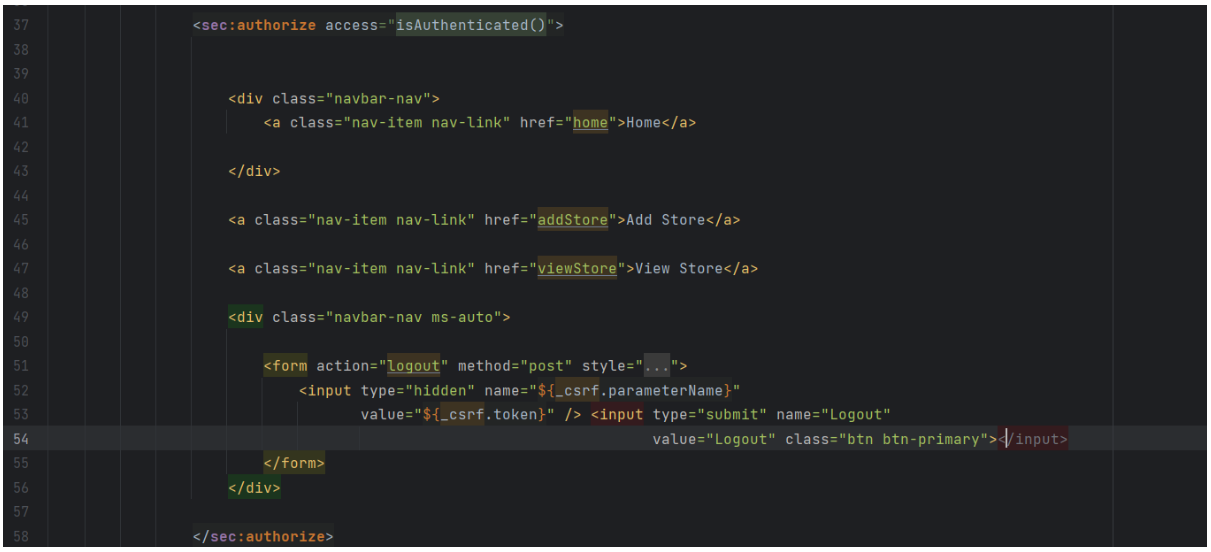
6. Add CSRF security to the application

Figure 25: login.jsp with the CSRF token added



```
<c:url var="post_url" value="/login" />
<form action="${post_url}" method="post" class="was-validated">
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
    <div class="mb-3 mt-3">
        <label for="username" class="form-label">Username:</label> <input
            type="text" class="form-control" id="userName"
            placeholder="Enter username" name="username" value="" required>
        <div class="valid-feedback">Valid.</div>
        <div class="invalid-feedback">Please fill out this field.</div>
    </div>
    <div class="mb-3">
        <label for="password" class="form-label">Password:</label> <input
            type="password" class="form-control" id="password"
            placeholder="Enter password" name="password" value="" required>
        <div class="valid-feedback">Valid.</div>
        <div class="invalid-feedback">Please fill out this field.</div>
    </div>
    <p class="mt-3">
        New User? Click here to <a href="register_user">Register</a>
    </p>

    <input type="submit" name="Login" value="Sign In"
           class="btn btn-primary"></input>
</form>
```

Figure 26: Security taglib used in header for authorization

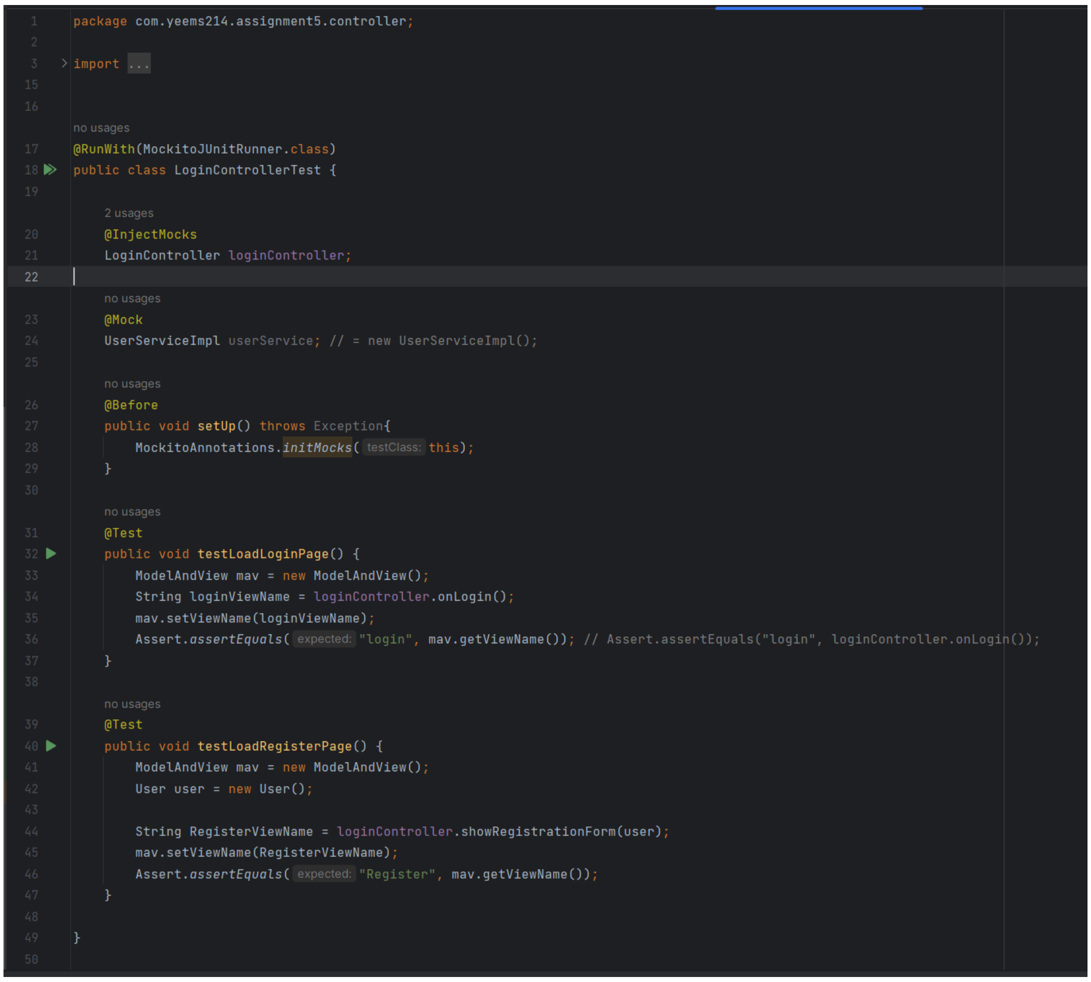


```
<sec:authorize access="isAuthenticated()">


    <div class="navbar-nav">
        <a class="nav-item nav-link" href="home">Home</a>

    </div>

    <a class="nav-item nav-link" href="addStore">Add Store</a>

    <a class="nav-item nav-link" href="viewStore">View Store</a>

    <div class="navbar-nav ms-auto">

        <form action="logout" method="post" style="...">
            <input type="hidden" name="${_csrf.parameterName}"
                   value="${_csrf.token}" /> <input type="submit" name="Logout"
                                                    value="Logout" class="btn btn-primary"></input>
        </form>
    </div>

</sec:authorize>
```

Logging

1. Develop 5 unit test cases to validate the role based authorization

Figure 27: LoginControllerTest



```
package com.yeems214.assignment5.controller;

> import ...


no usages
@RunWith(MockitoJUnitRunner.class)
public class LoginControllerTest {

    2 usages
    @InjectMocks
    LoginController loginController;

    no usages
    @Mock
    UserServiceImpl userService; // = new UserServiceImpl();

    no usages
    @Before
    public void setUp() throws Exception{
        MockitoAnnotations.initMocks( testClass: this);
    }

    no usages
    @Test
    public void testLoadLoginPage() {
        ModelAndView mav = new ModelAndView();
        String loginViewName = loginController.onLogin();
        mav.setViewName(loginViewName);
        Assert.assertEquals( expected: "login", mav.getViewName()); // Assert.assertEquals("login", loginController.onLogin());
    }

    no usages
    @Test
    public void testLoadRegisterPage() {
        ModelAndView mav = new ModelAndView();
        User user = new User();

        String RegisterViewName = loginController.showRegistrationForm(user);
        mav.setViewName(RegisterViewName);
        Assert.assertEquals( expected: "Register", mav.getViewName());
    }

}
```

Figure 28: Output of LoginControllerTest

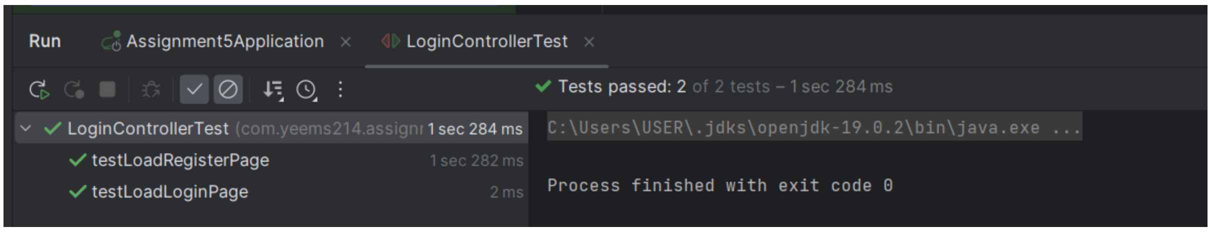

```
Run    Assignment5Application ×    LoginControllerTest ×

                                          ✓ Tests passed: 2 of 2 tests – 1 sec 284 ms
✓ LoginControllerTest (com.yeems214.assigni 1 sec 284 ms   C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe ...
  ✓ testLoadRegisterPage                    1 sec 282 ms
  ✓ testLoadLoginPage                              2 ms   Process finished with exit code 0
```

Figure 29: StoreServiceTest

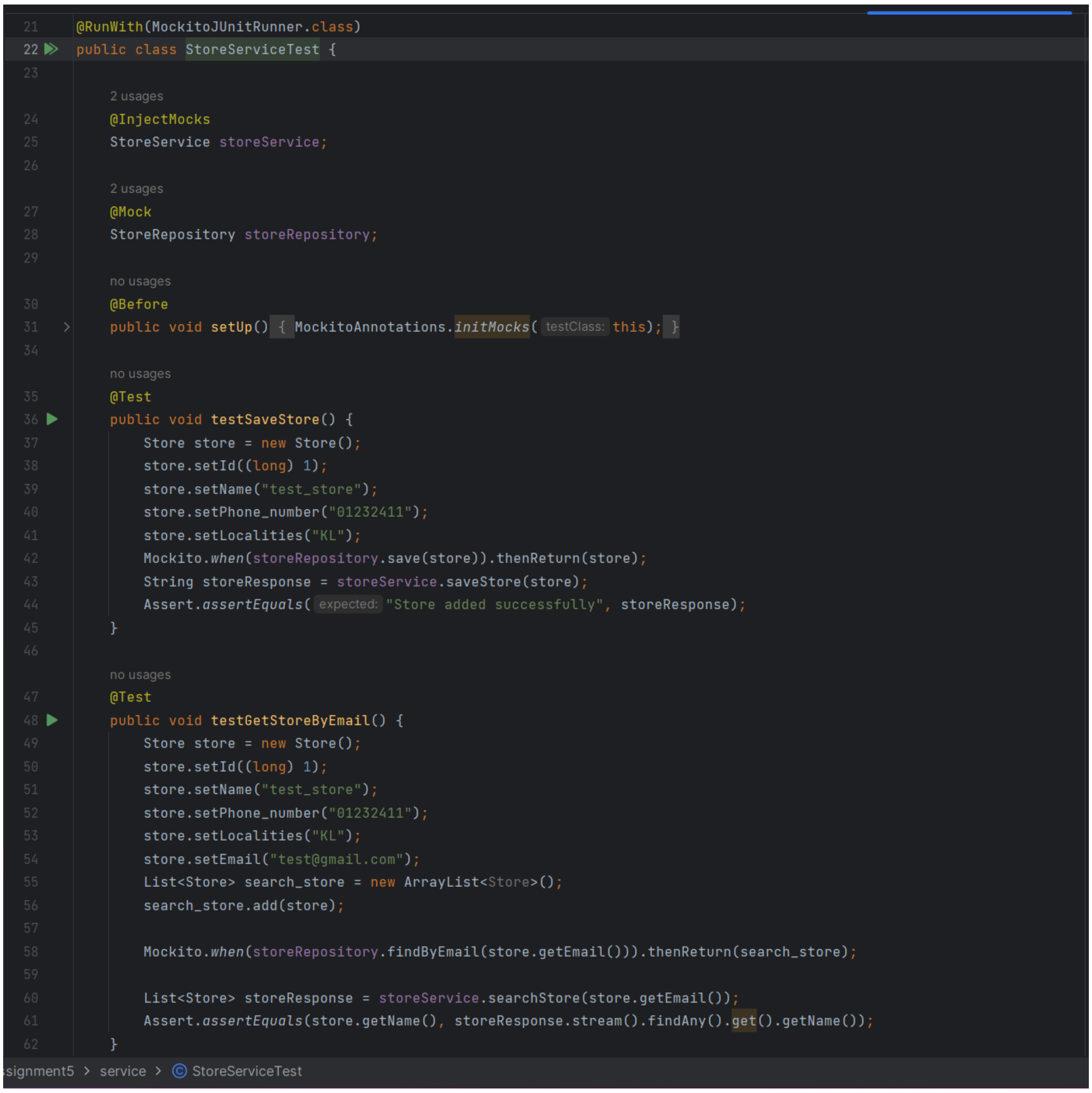
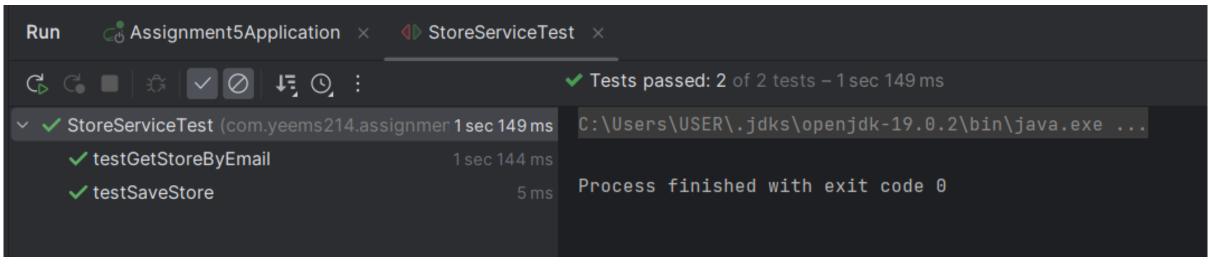
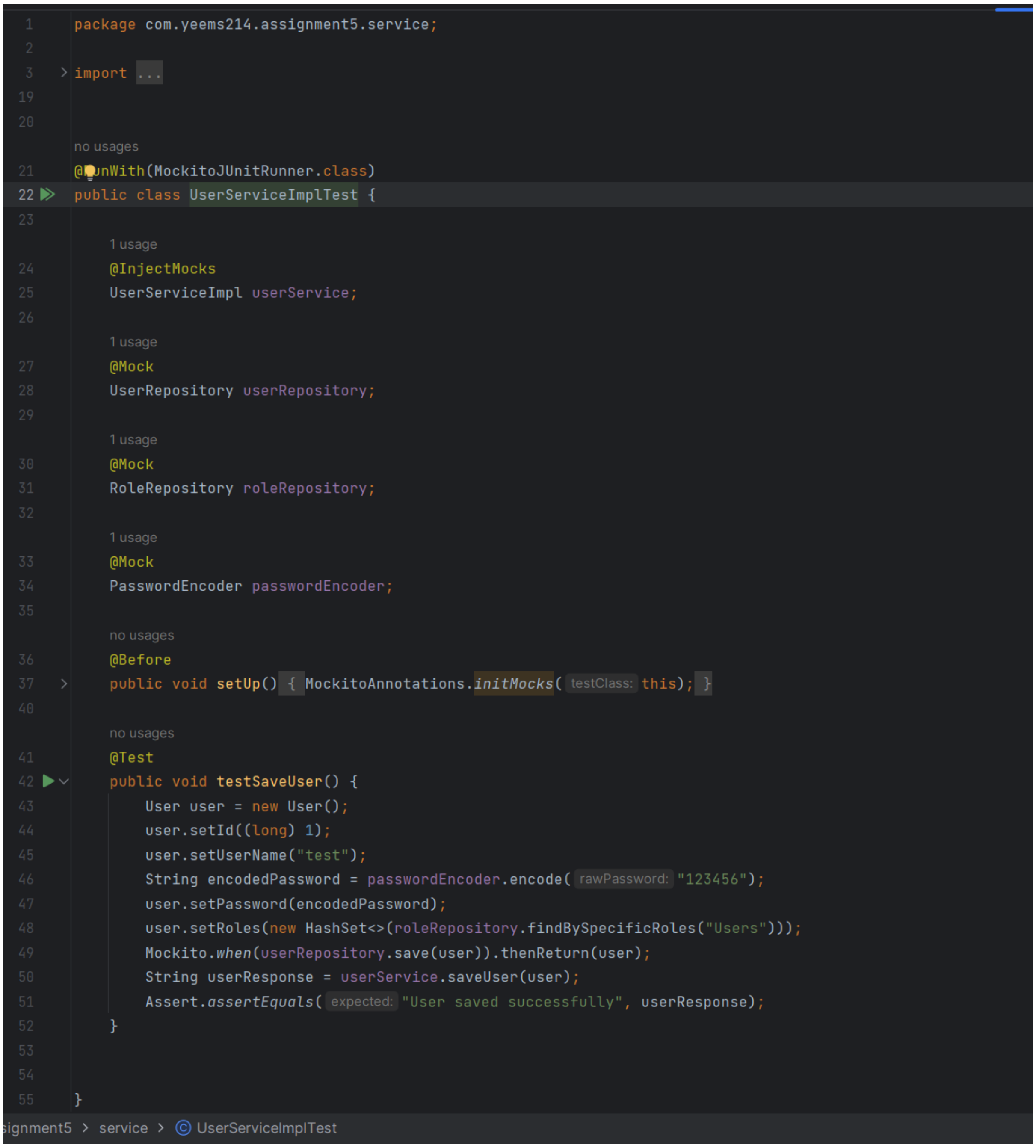

```java
@RunWith(MockitoJUnitRunner.class)
public class StoreServiceTest {

    2 usages
    @InjectMocks
    StoreService storeService;

    2 usages
    @Mock
    StoreRepository storeRepository;

    no usages
    @Before
    public void setUp() { MockitoAnnotations.initMocks( testClass: this); }

    no usages
    @Test
    public void testSaveStore() {
        Store store = new Store();
        store.setId((long) 1);
        store.setName("test_store");
        store.setPhone_number("01232411");
        store.setLocalities("KL");
        Mockito.when(storeRepository.save(store)).thenReturn(store);
        String storeResponse = storeService.saveStore(store);
        Assert.assertEquals( expected: "Store added successfully", storeResponse);
    }

    no usages
    @Test
    public void testGetStoreByEmail() {
        Store store = new Store();
        store.setId((long) 1);
        store.setName("test_store");
        store.setPhone_number("01232411");
        store.setLocalities("KL");
        store.setEmail("test@gmail.com");
        List<Store> search_store = new ArrayList<Store>();
        search_store.add(store);

        Mockito.when(storeRepository.findByEmail(store.getEmail())).thenReturn(search_store);

        List<Store> storeResponse = storeService.searchStore(store.getEmail());
        Assert.assertEquals(store.getName(), storeResponse.stream().findAny().get().getName());
    }
```
signment5 > service > StoreServiceTest

Figure 30: Output of StoreServiceTest



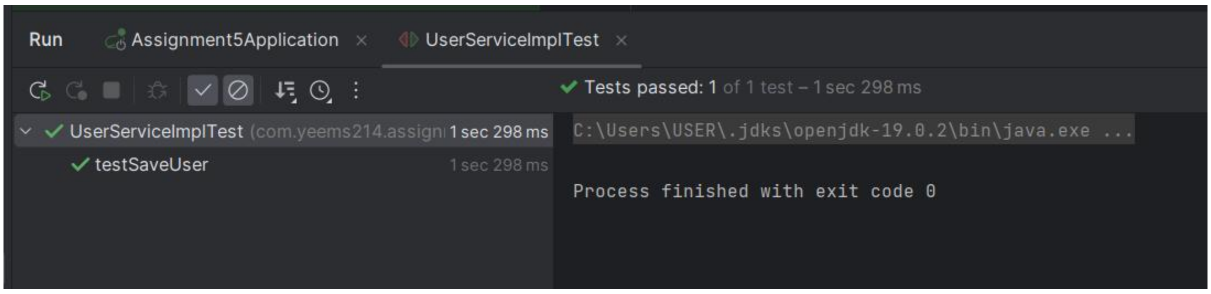
```
Tests passed: 2 of 2 tests – 1 sec 149 ms

StoreServiceTest (com.yeems214.assignmer 1 sec 149 ms
  testGetStoreByEmail                    1 sec 144 ms
  testSaveStore                                  5 ms

C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe ...

Process finished with exit code 0
```

Figure 31: UserServiceImplTest



```java
package com.yeems214.assignment5.service;

import ...


no usages
@RunWith(MockitoJUnitRunner.class)
public class UserServiceImplTest {

    1 usage
    @InjectMocks
    UserServiceImpl userService;

    1 usage
    @Mock
    UserRepository userRepository;

    1 usage
    @Mock
    RoleRepository roleRepository;

    1 usage
    @Mock
    PasswordEncoder passwordEncoder;

    no usages
    @Before
    public void setUp() { MockitoAnnotations.initMocks( testClass: this); }

    no usages
    @Test
    public void testSaveUser() {
        User user = new User();
        user.setId((long) 1);
        user.setUserName("test");
        String encodedPassword = passwordEncoder.encode( rawPassword: "123456");
        user.setPassword(encodedPassword);
        user.setRoles(new HashSet<>(roleRepository.findBySpecificRoles("Users")));
        Mockito.when(userRepository.save(user)).thenReturn(user);
        String userResponse = userService.saveUser(user);
        Assert.assertEquals( expected: "User saved successfully", userResponse);
    }



}
```

Figure 32: Output of UserServiceImplTest

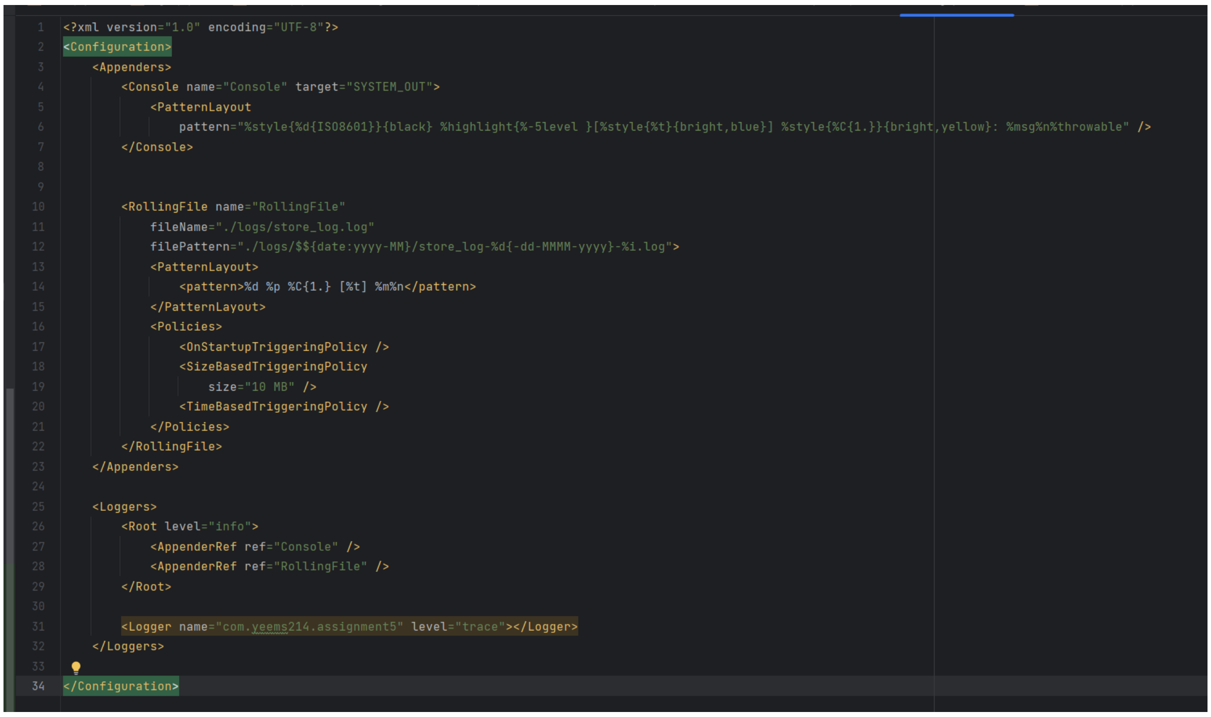

Run
Assignment5Application
UserServiceImplTest
UserServiceImplTest (com.yeems214.assign 1 sec 298 ms
testSaveUser 1 sec 298 ms

```
Tests passed: 1 of 1 test – 1 sec 298 ms
C:\Users\USER\.jdks\openjdk-19.0.2\bin\java.exe ...

Process finished with exit code 0
```

Logging

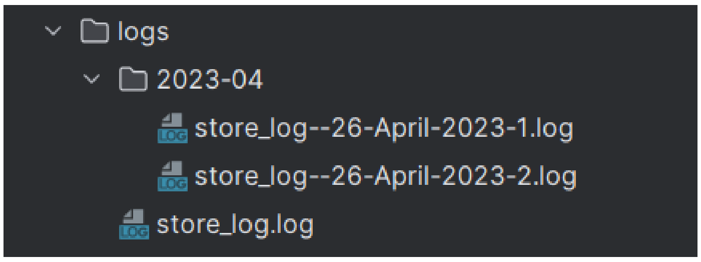
1. Implement the rolling file appended and log at the class level

Figure 33: log4j2.xml Source Code



```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout
                pattern="%style{%d{ISO8601}}{black} %highlight{%-5level }[%style{%t}{bright,blue}] %style{%C{1.}}{bright,yellow}: %msg%n%throwable" />
        </Console>


        <RollingFile name="RollingFile"
            fileName="./logs/store_log.log"
            filePattern="./logs/$${date:yyyy-MM}/store_log-%d{-dd-MMMM-yyyy}-%i.log">
            <PatternLayout>
                <pattern>%d %p %C{1.} [%t] %m%n</pattern>
            </PatternLayout>
            <Policies>
                <OnStartupTriggeringPolicy />
                <SizeBasedTriggeringPolicy
                    size="10 MB" />
                <TimeBasedTriggeringPolicy />
            </Policies>
        </RollingFile>
    </Appenders>

    <Loggers>
        <Root level="info">
            <AppenderRef ref="Console" />
            <AppenderRef ref="RollingFile" />
        </Root>

        <Logger name="com.yeems214.assignment5" level="trace"></Logger>
    </Loggers>

</Configuration>
```

Figure 34: log file directory

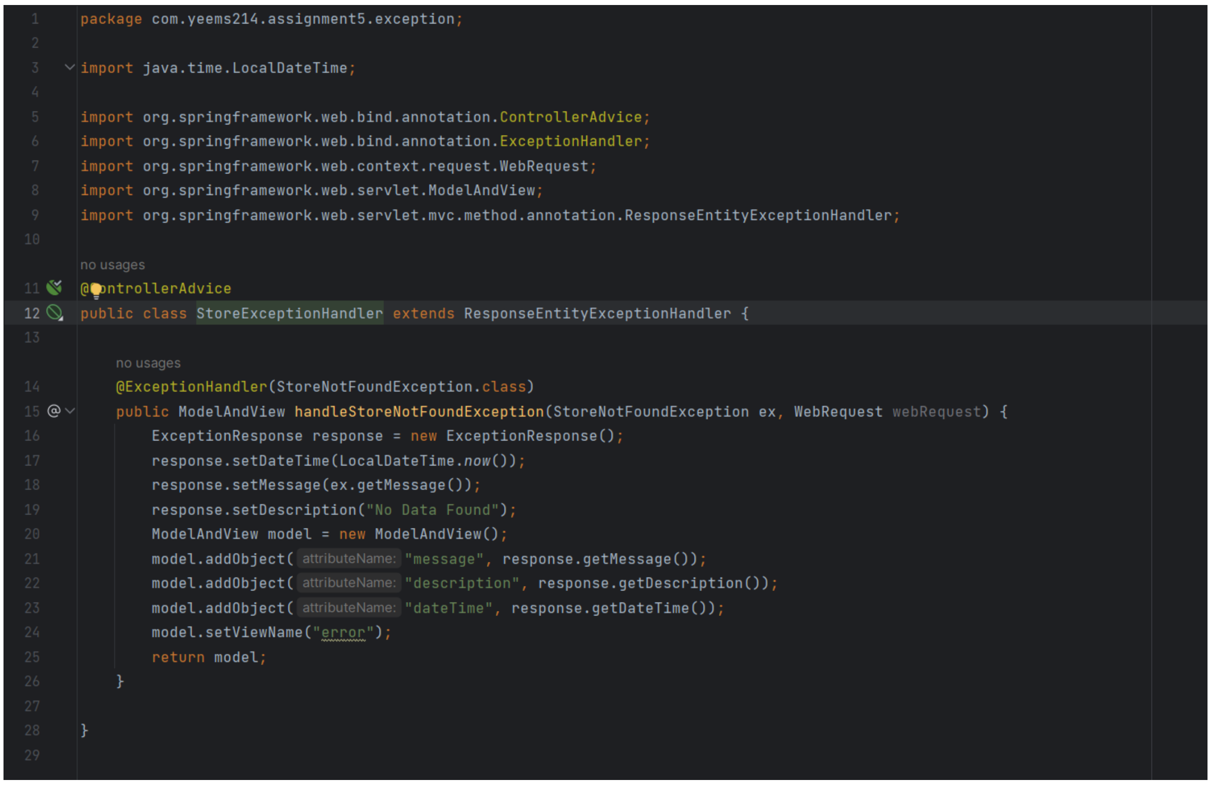


```
logs
    2023-04
        store_log--26-April-2023-1.log
        store_log--26-April-2023-2.log
    store_log.log
```

Error Handling

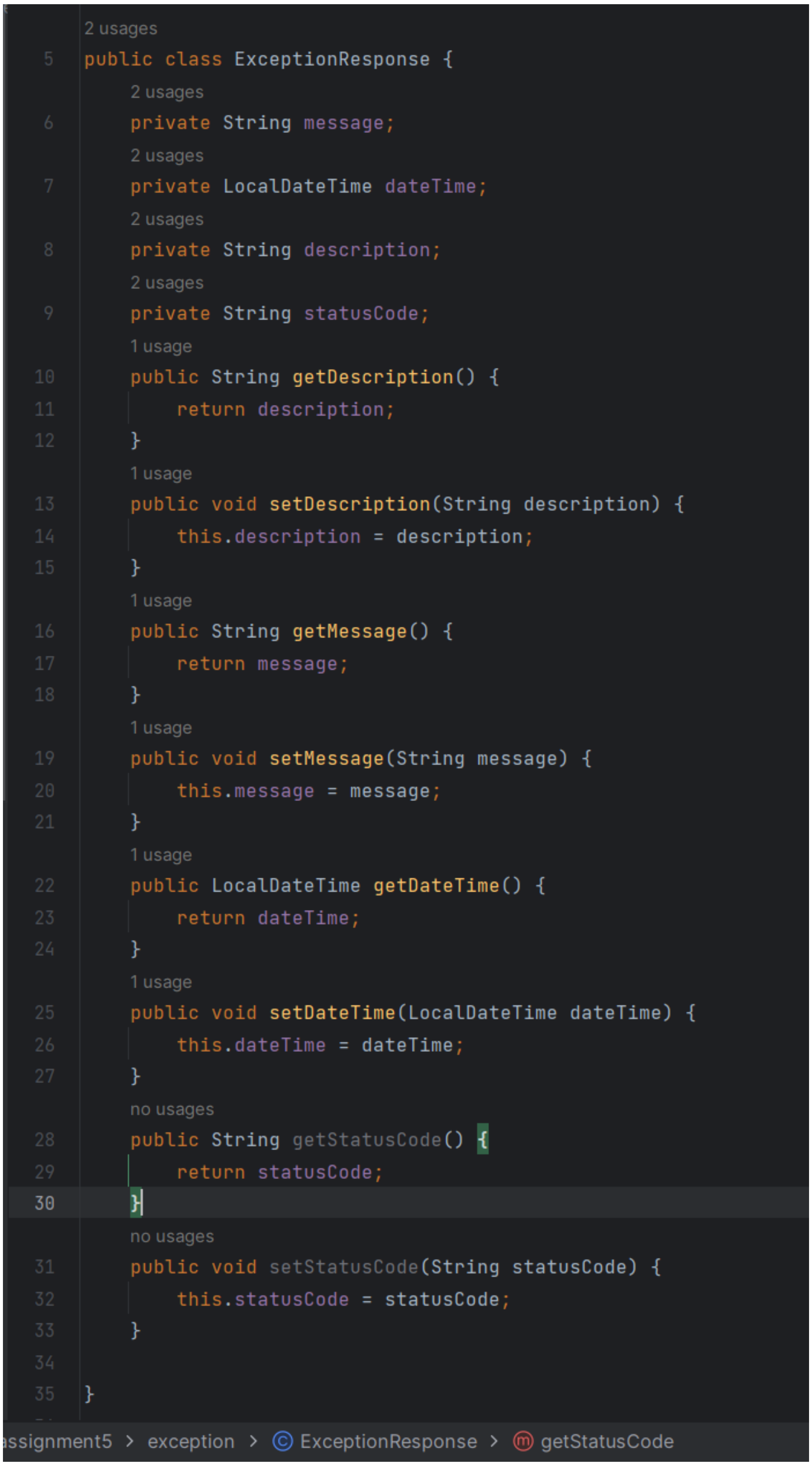
1. Create Global Exception Handler that can handle an exception. Add method to handle run away exception. [Run away exception is a RuntimeException that is not handled by any exception handler].

Figure 35: StoreExceptionHandler.java



```java
package com.yeems214.assignment5.exception;

import java.time.LocalDateTime;

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

no usages
@ControllerAdvice
public class StoreExceptionHandler extends ResponseEntityExceptionHandler {

    no usages
    @ExceptionHandler(StoreNotFoundException.class)
    public ModelAndView handleStoreNotFoundException(StoreNotFoundException ex, WebRequest webRequest) {
        ExceptionResponse response = new ExceptionResponse();
        response.setDateTime(LocalDateTime.now());
        response.setMessage(ex.getMessage());
        response.setDescription("No Data Found");
        ModelAndView model = new ModelAndView();
        model.addObject( attributeName: "message", response.getMessage());
        model.addObject( attributeName: "description", response.getDescription());
        model.addObject( attributeName: "dateTime", response.getDateTime());
        model.setViewName("error");
        return model;
    }

}
```

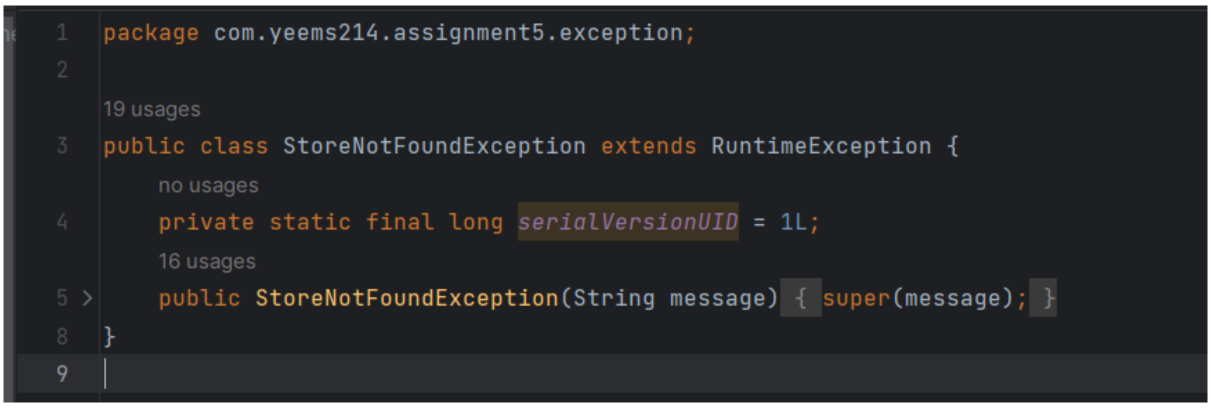
Figure 36: ExceptionResponse.java



```
2 usages
public class ExceptionResponse {
    2 usages
    private String message;
    2 usages
    private LocalDateTime dateTime;
    2 usages
    private String description;
    2 usages
    private String statusCode;
    1 usage
    public String getDescription() {
        return description;
    }
    1 usage
    public void setDescription(String description) {
        this.description = description;
    }
    1 usage
    public String getMessage() {
        return message;
    }
    1 usage
    public void setMessage(String message) {
        this.message = message;
    }
    1 usage
    public LocalDateTime getDateTime() {
        return dateTime;
    }
    1 usage
    public void setDateTime(LocalDateTime dateTime) {
        this.dateTime = dateTime;
    }
    no usages
    public String getStatusCode() {
        return statusCode;
    }
    no usages
    public void setStatusCode(String statusCode) {
        this.statusCode = statusCode;
    }

}
```

2. Create an exception class StoreNotFoundException.

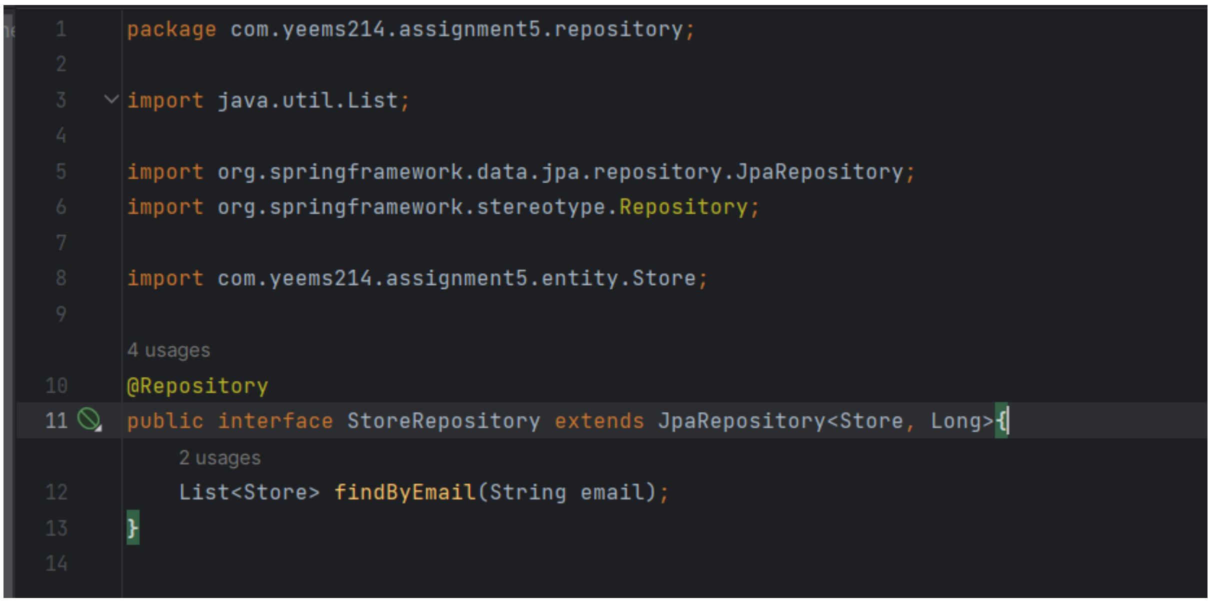
Figure 37: StoreNotFoundException.java



```java
package com.yeems214.assignment5.exception;

19 usages
public class StoreNotFoundException extends RuntimeException {
    no usages
    private static final long serialVersionUID = 1L;
    16 usages
    public StoreNotFoundException(String message) { super(message); }
}

```

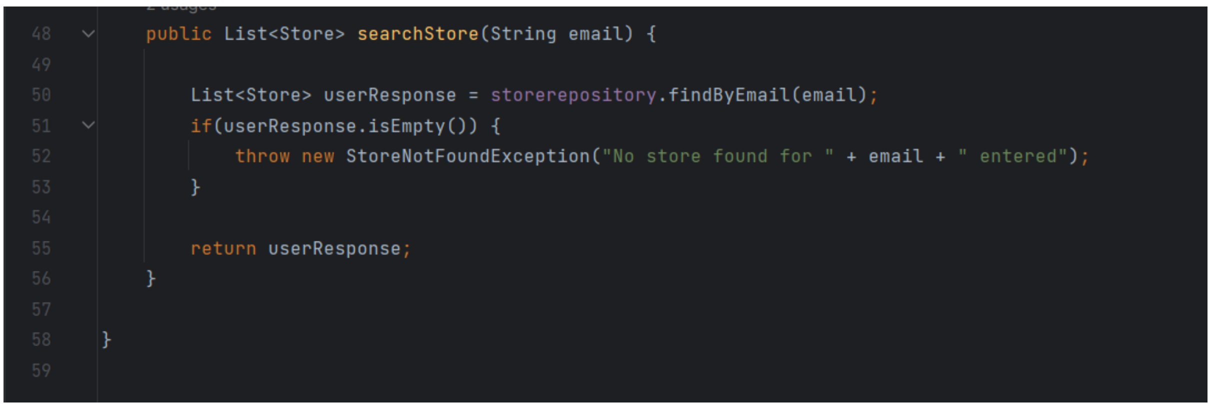
3. Create an end point to search store by email.

Figure 38: StoreRepository.java



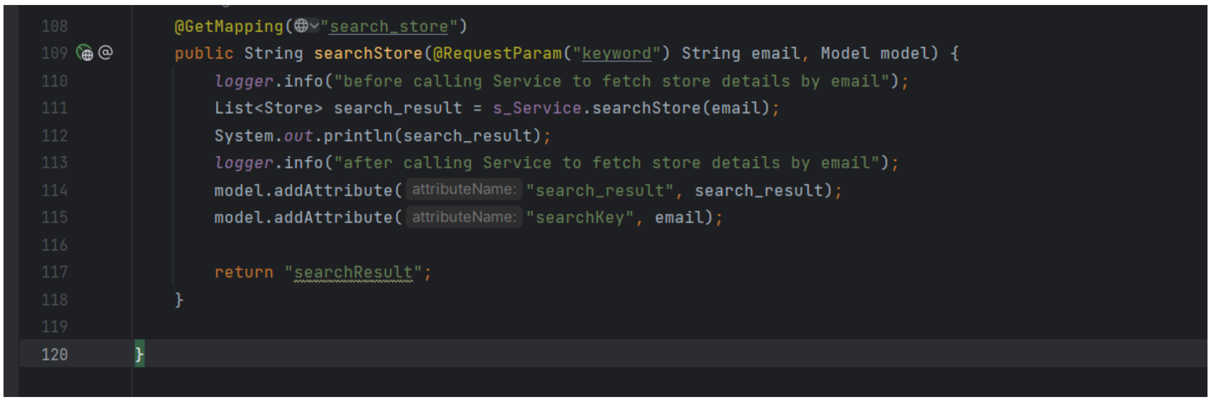
```java
package com.yeems214.assignment5.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.yeems214.assignment5.entity.Store;

4 usages
@Repository
public interface StoreRepository extends JpaRepository<Store, Long>{
    2 usages
    List<Store> findByEmail(String email);
}

```

Figure 39: StoreService.java searchStore method



```java
    public List<Store> searchStore(String email) {

        List<Store> userResponse = storerepository.findByEmail(email);
        if(userResponse.isEmpty()) {
            throw new StoreNotFoundException("No store found for " + email + " entered");
        }

        return userResponse;
    }

}

```

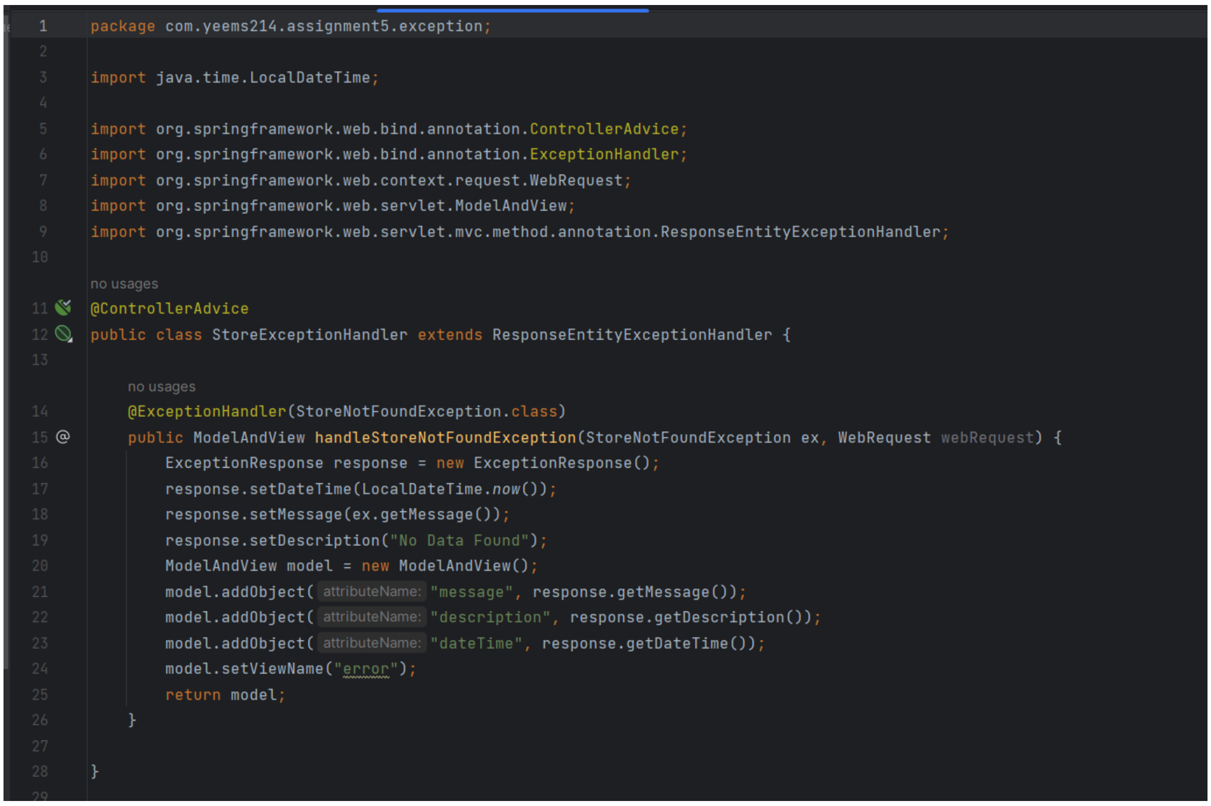
Figure 40: StoreController.java searchStore method



```java
    @GetMapping("search_store")
    public String searchStore(@RequestParam("keyword") String email, Model model) {
        logger.info("before calling Service to fetch store details by email");
        List<Store> search_result = s_Service.searchStore(email);
        System.out.println(search_result);
        logger.info("after calling Service to fetch store details by email");
        model.addAttribute( attributeName: "search_result", search_result);
        model.addAttribute( attributeName: "searchKey", email);

        return "searchResult";
    }

}
```

4. Create exception handler specific to the controller.

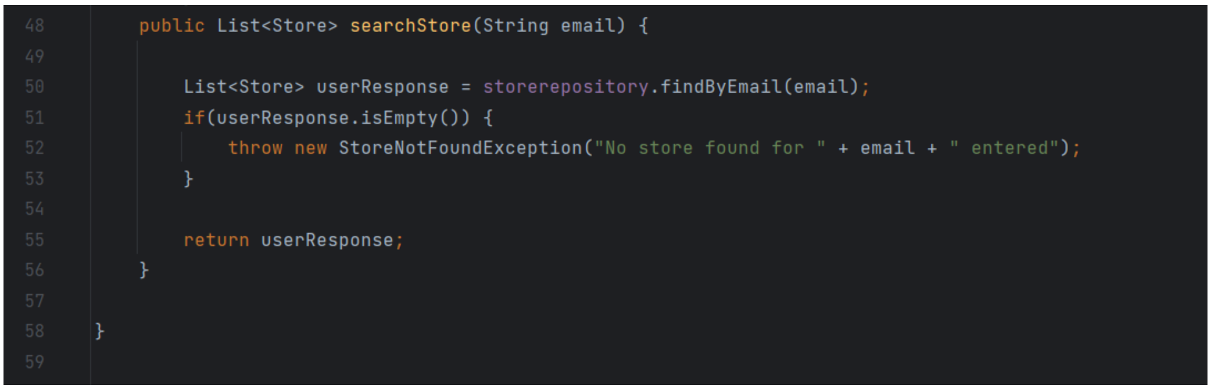
Figure 41: StoreExceptionHandler.java



```java
package com.yeems214.assignment5.exception;

import java.time.LocalDateTime;

import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

no usages
@ControllerAdvice
public class StoreExceptionHandler extends ResponseEntityExceptionHandler {

    no usages
    @ExceptionHandler(StoreNotFoundException.class)
    public ModelAndView handleStoreNotFoundException(StoreNotFoundException ex, WebRequest webRequest) {
        ExceptionResponse response = new ExceptionResponse();
        response.setDateTime(LocalDateTime.now());
        response.setMessage(ex.getMessage());
        response.setDescription("No Data Found");
        ModelAndView model = new ModelAndView();
        model.addObject( attributeName: "message", response.getMessage());
        model.addObject( attributeName: "description", response.getDescription());
        model.addObject( attributeName: "dateTime", response.getDateTime());
        model.setViewName("error");
        return model;
    }

}
```

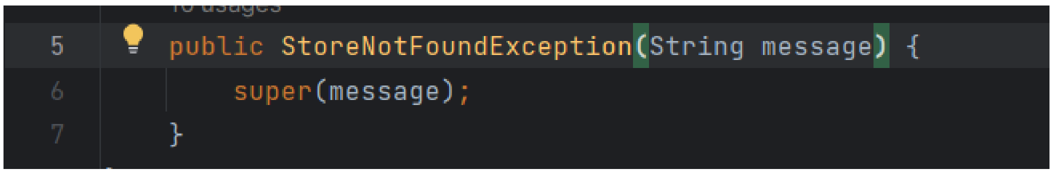
5. Let the service throw StoreNotFoundException if no store is found for given **email.**

Figure 42: StoreService.java serachStore method



```java
    public List<Store> searchStore(String email) {

        List<Store> userResponse = storerepository.findByEmail(email);
        if(userResponse.isEmpty()) {
            throw new StoreNotFoundException("No store found for " + email + " entered");
        }

        return userResponse;
    }

}
```

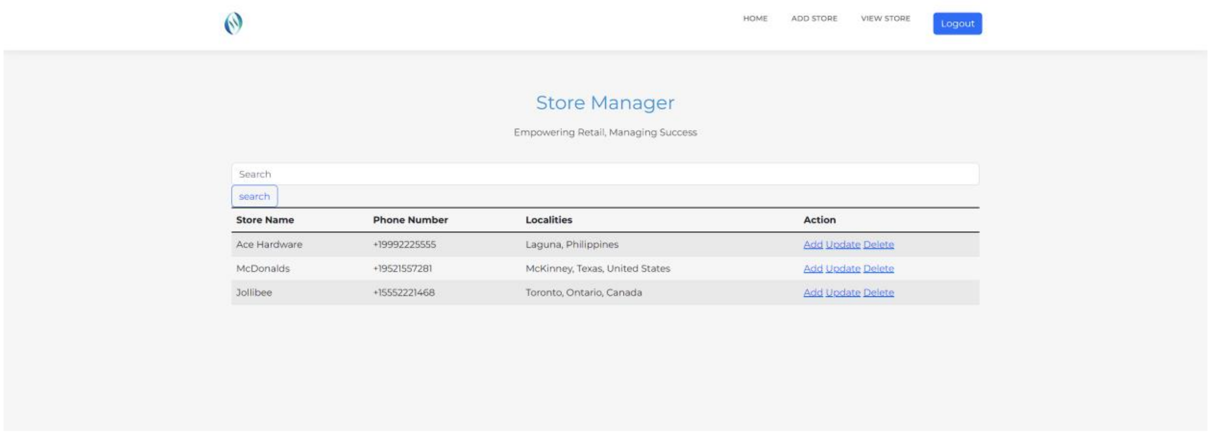
Figure 43: StoreService.java message passes to StoreNotFoundException.java



```java
    public StoreNotFoundException(String message) {
        super(message);
    }
```

7. Let the StoreNotFoundException handler route to view stores page and show the error message in view stores page.

Figure 44: storeList.jsp

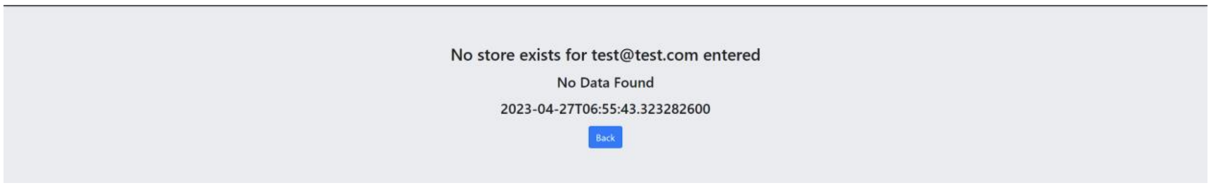


Store Manager

Empowering Retail, Managing Success

Store Name	Phone Number	Localities	Action
Ace Hardware	+19992225555	Laguna, Philippines	Add Update Delete
McDonalds	+19521557281	McKinney, Texas, United States	Add Update Delete
Jollibee	+15552221468	Toronto, Ontario, Canada	Add Update Delete

Figure 45: error.jsp when searching for the email of user with no data

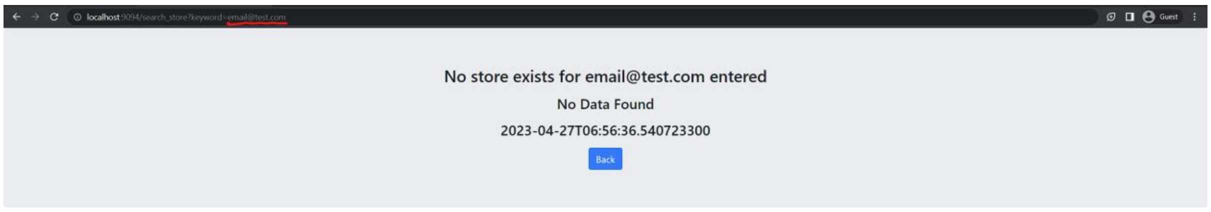

No store exists for test@test.com entered

No Data Found

2023-04-27T06:55:43.323282600

8. Enter the url http://:/stores?email@test.com. This should route to view stores page and show the error message.

Figure 46: error.jsp for keyword=email@test.com


localhost:9094/search_store?keyword=email@test.com

No store exists for email@test.com entered

No Data Found

2023-04-27T06:56:36.540723300