



|  |  |
| --- | --- |
| **Student Name/ID Number:** | Agung Yuda Pratama / 0922-076 |
| **Academic Year:** | 2022 |
| **Unit Assessor:** | Archana Sakpal |
| **Project Title:** | Assignment5 – Spring Security, Error Handling, and Logging |
| **Issue Date:** |  |
| **Submission Date:** |  |
| **Internal Verifier Name:** |  |
| **Date:** | 19-04-2023 |

|  |
| --- |
| **Learner declaration** |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.    19/04/2023  Student signature: Date: |

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

Figure 1 know your neighbourhood database, kyn

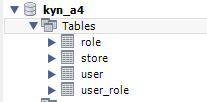


Figure 2 user Table

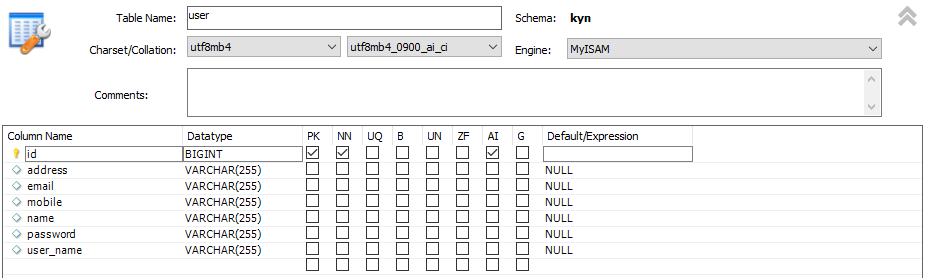


Figure 3 role Table

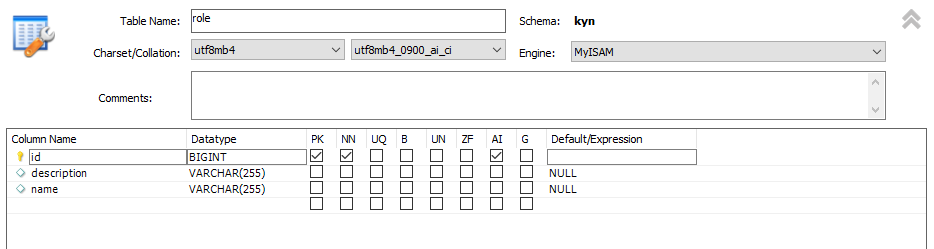


Figure 4 user\_role table

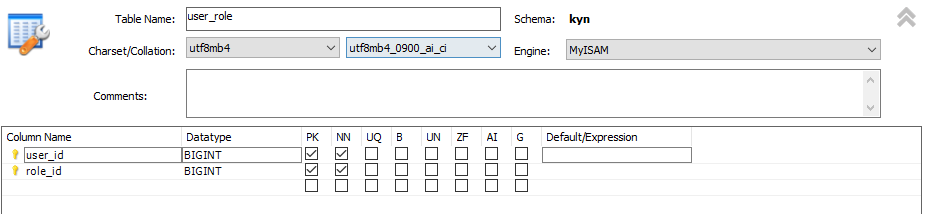
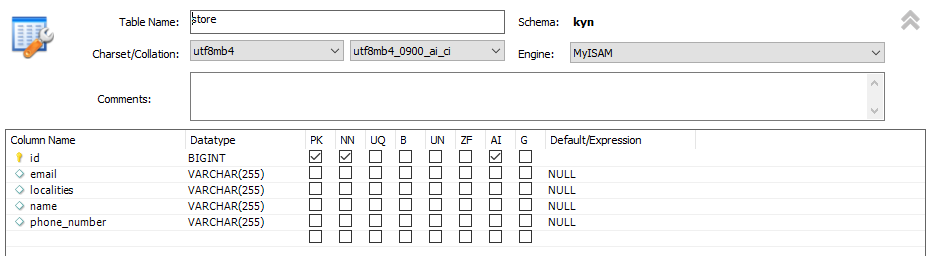


Figure 5 store table



1. Add entries into user, role (roleid, role name) and user\_role tables.
   1. 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).

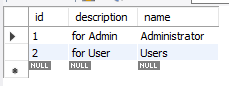
Figure 6 insert statements for Administrator with role to add/modify store info (ADD\_STORE)



Figure 7 insert statements for Users with role to view store info (VIEW\_STORE)



Figure 8 role Table



* 1. Add two entries into user table. Let one user have play VIEW\_STORE role, another user plays only both VIEW\_STORE & ADD\_STORE roles.

Figure 9 insert statements for user with Administrator role that can VIEW\_STORE & ADD\_STORE

Note: To add the user after we first launch the app, we register for the admin first, If success go to the database and make the update on the user\_role table



Figure 10 insert statements for user with Users role that can VIEW\_STORE



Figure 11 user Table

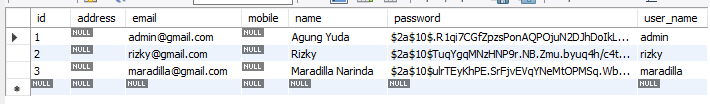
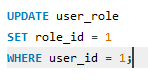
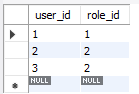


Figure 12 user-role Table

After we finish the registration the default role id is 2. We must make the update so the admin can work properly





1. Enhance Spring Data JPA enabled version of ‘Know-Your-Neighbourhood’ application to support authentication and authorization

Figure 13 UserRepository to identify user by username

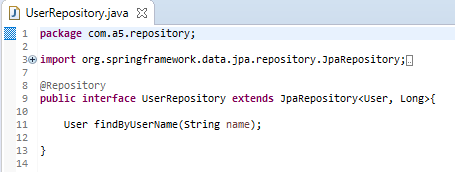


Figure 14 RoleRepository to assign user role by finding either Users or Administrator role from the role table. On registration, user will be assigned, Users role as default

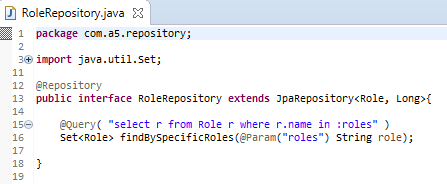
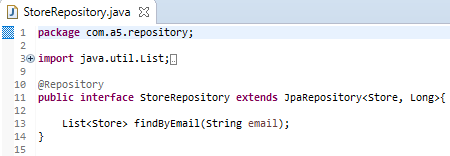


Figure 15 StoreRepository to find store by email of administrator that ADD\_STORE to the system



1. **Authenticatio**n:
   1. Create entities for Role and Store.

Figure 16 Store.java

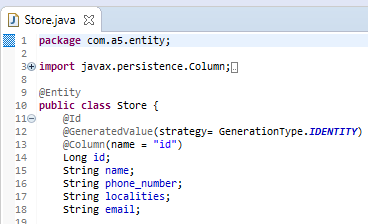


Figure 17 Role.java

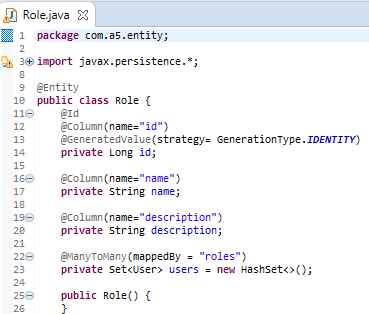
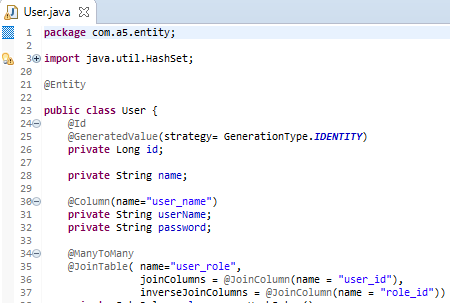
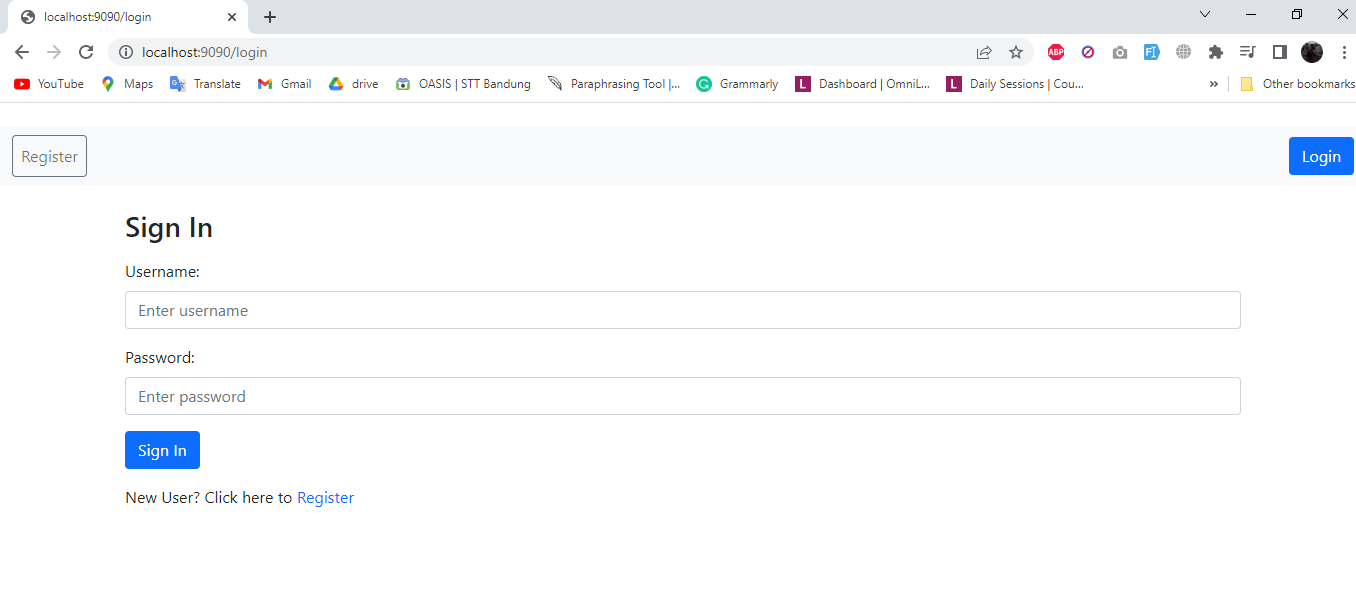


Figure 18 User.java



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

Figure 19 login.jsp page



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

Figure 20 login.jsp page with credentials added

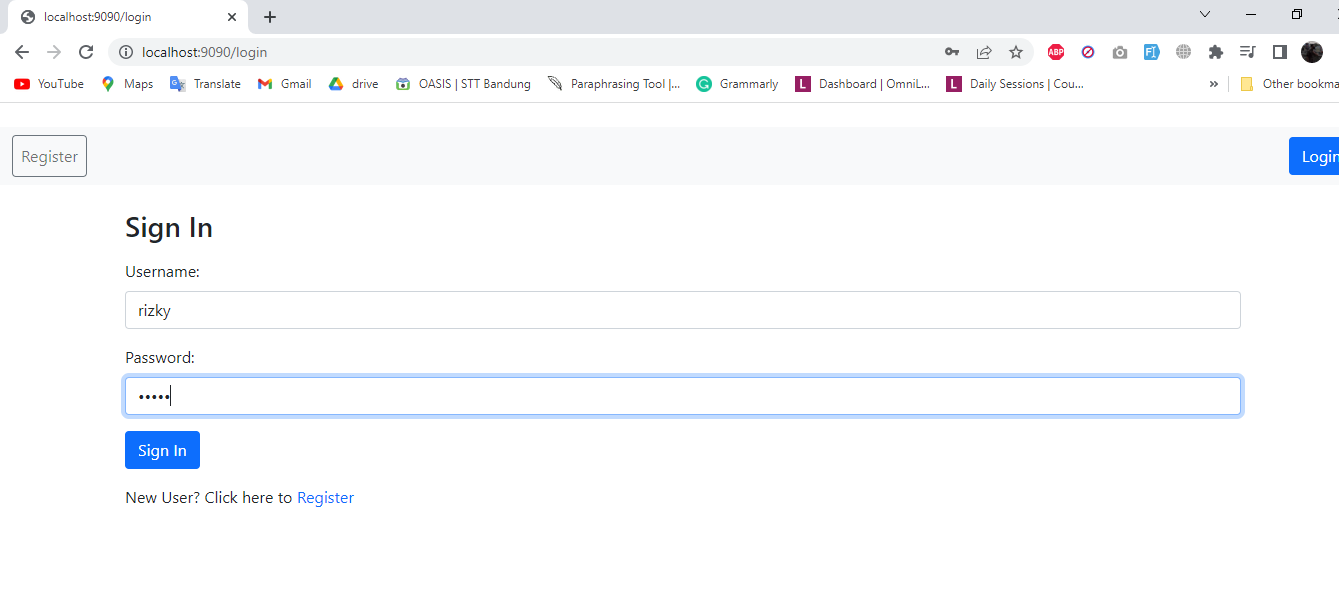


Figure 21 login success after entering correct credentials

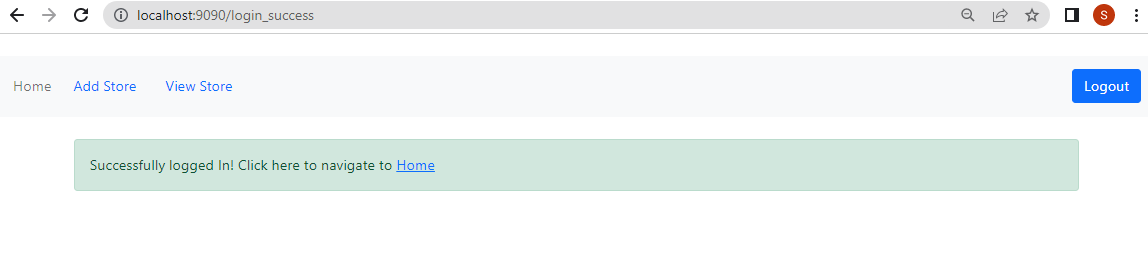


Figure 22 index.jsp page for user Role that can VIEW\_STORE only

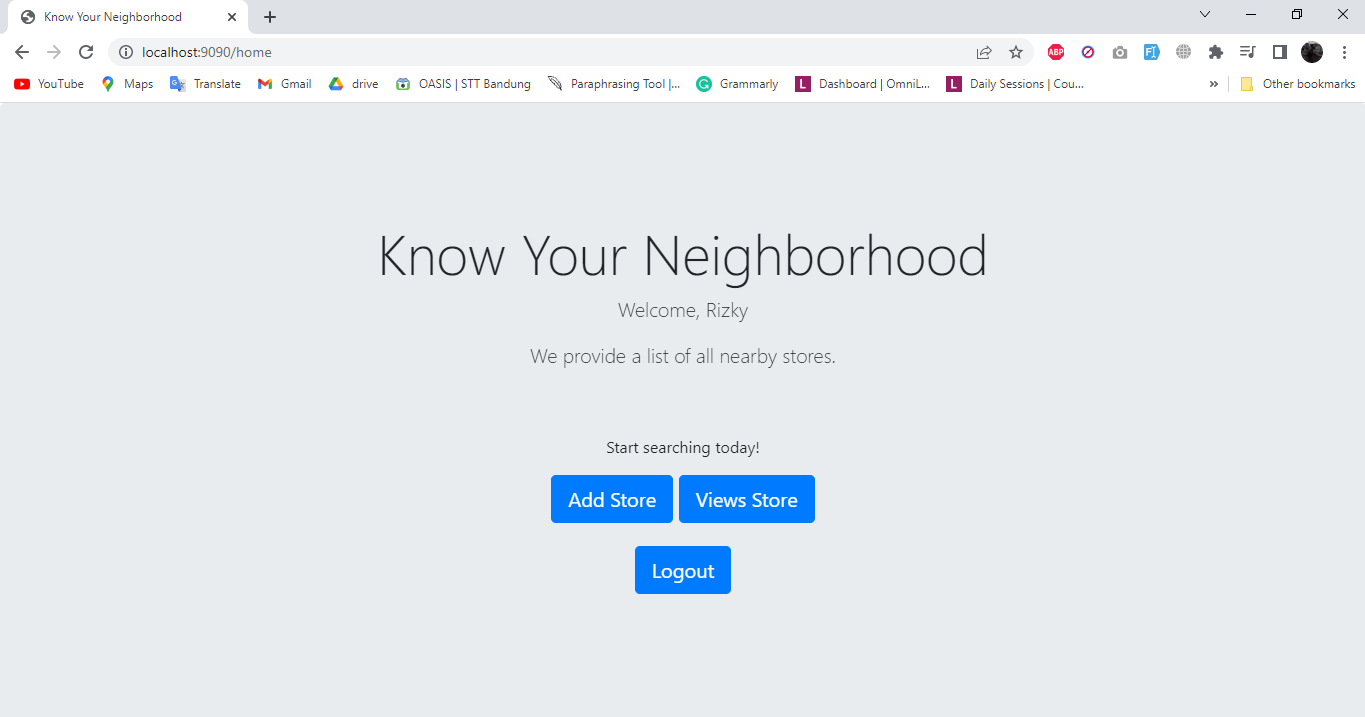


Figure 23 user with role 'User' can only View Store only and if Add Store was selected, user will be directed to an access denied page

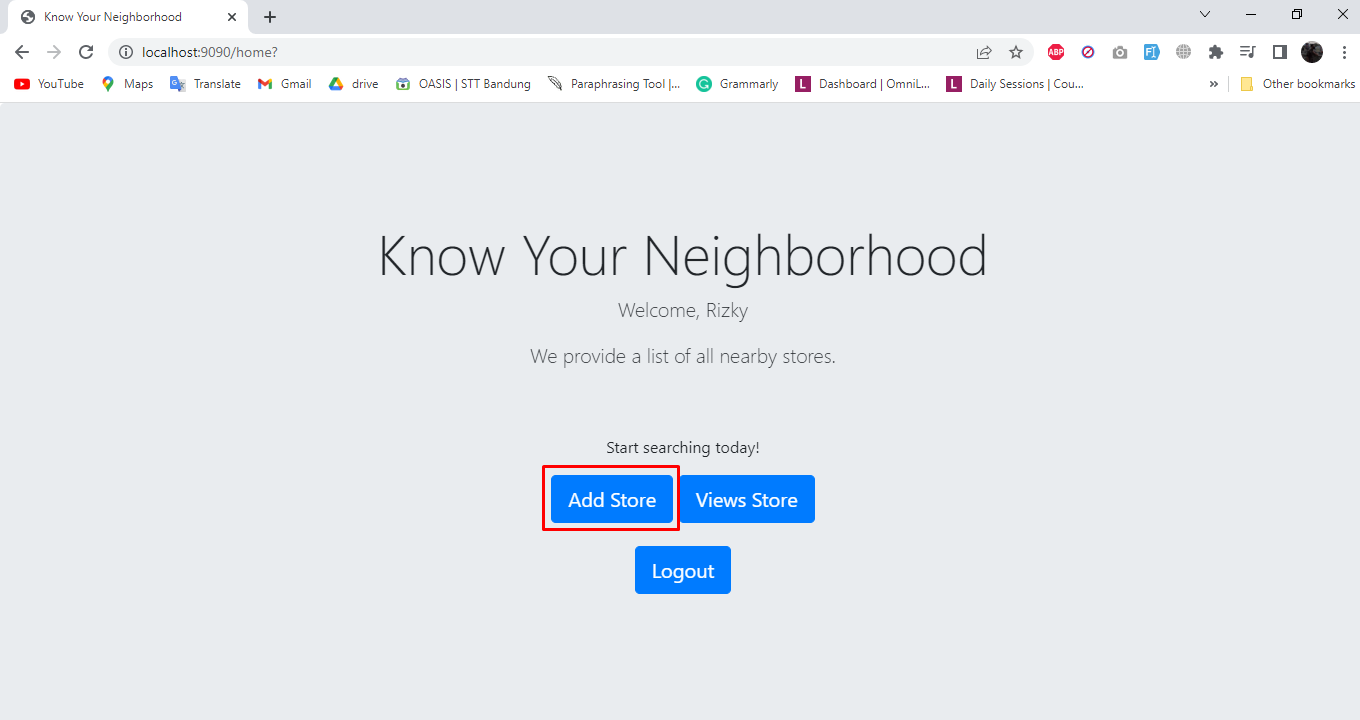
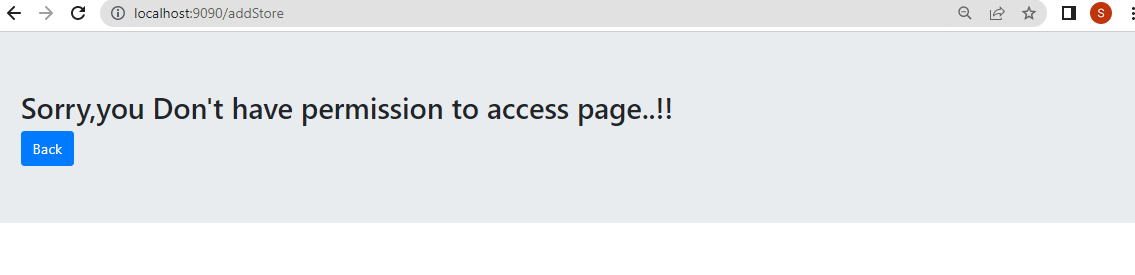
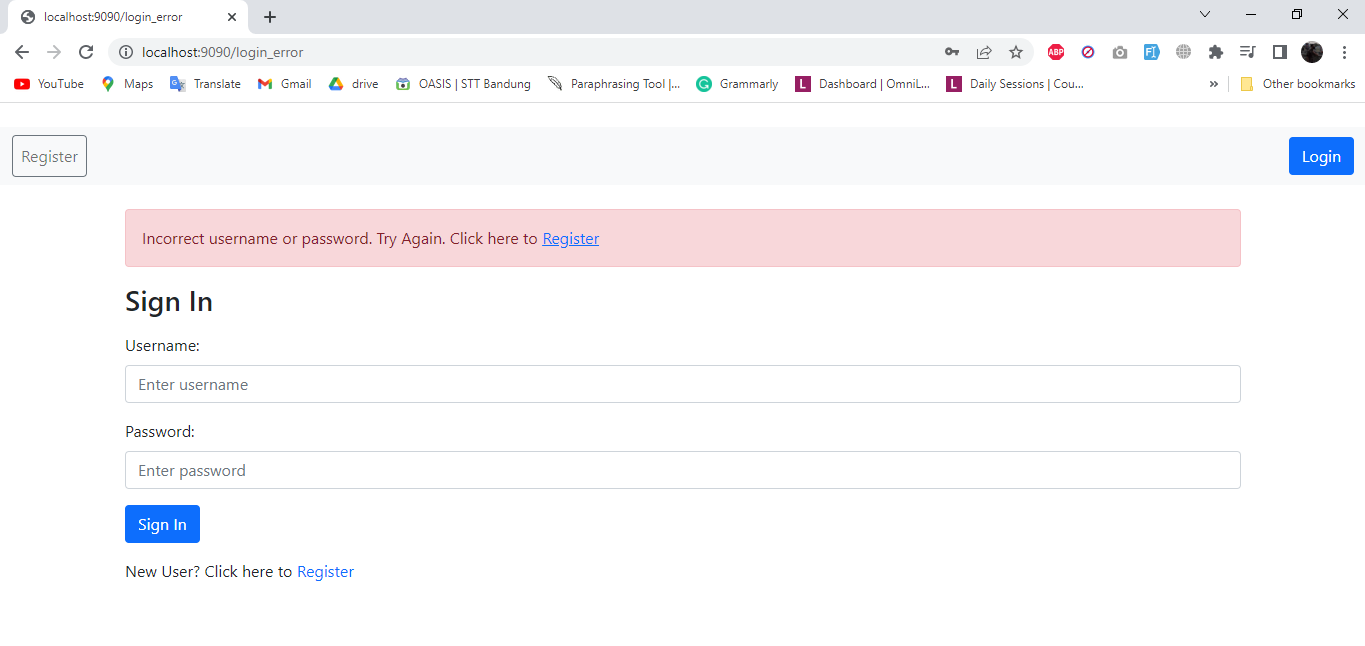


Figure 24 Access denied page seen by user with Role 'User'



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

Figure 25 login error after entering incorrect credentials



1. **Authorization:** 
   1. 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 25 StoreController.java for Administrator role that can ADD\_STORE

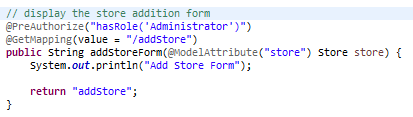


Figure 26 StoreController.java for Administrator role that can VIEW\_STORE and ADD\_STORE and User role that can only VIEW\_STORE

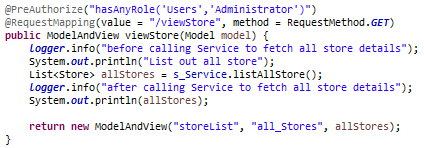
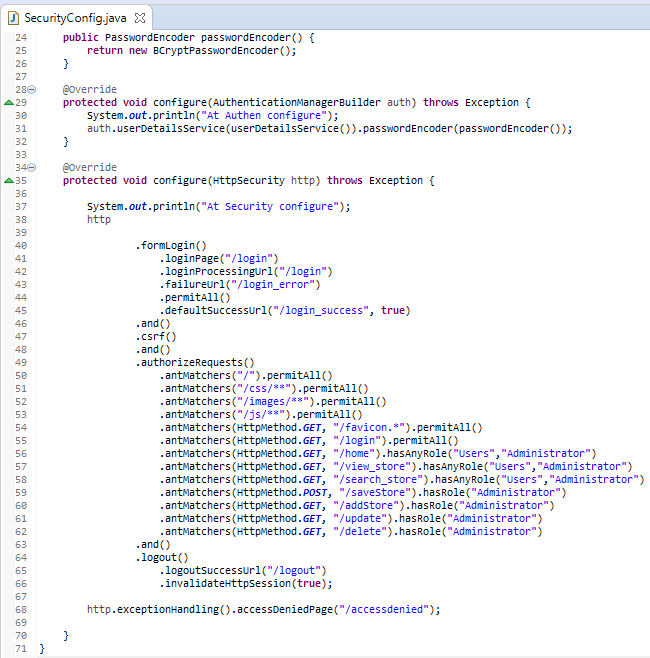


Figure 27 security configuration, authorization for each URL mapping in website based on role User that can VIEW\_STORE only and role Administrator that can VIEW\_STORE & ADD\_STORE



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

Figure 28 StoreNotFoundException when there is no store added in the system displayed after user click on view store

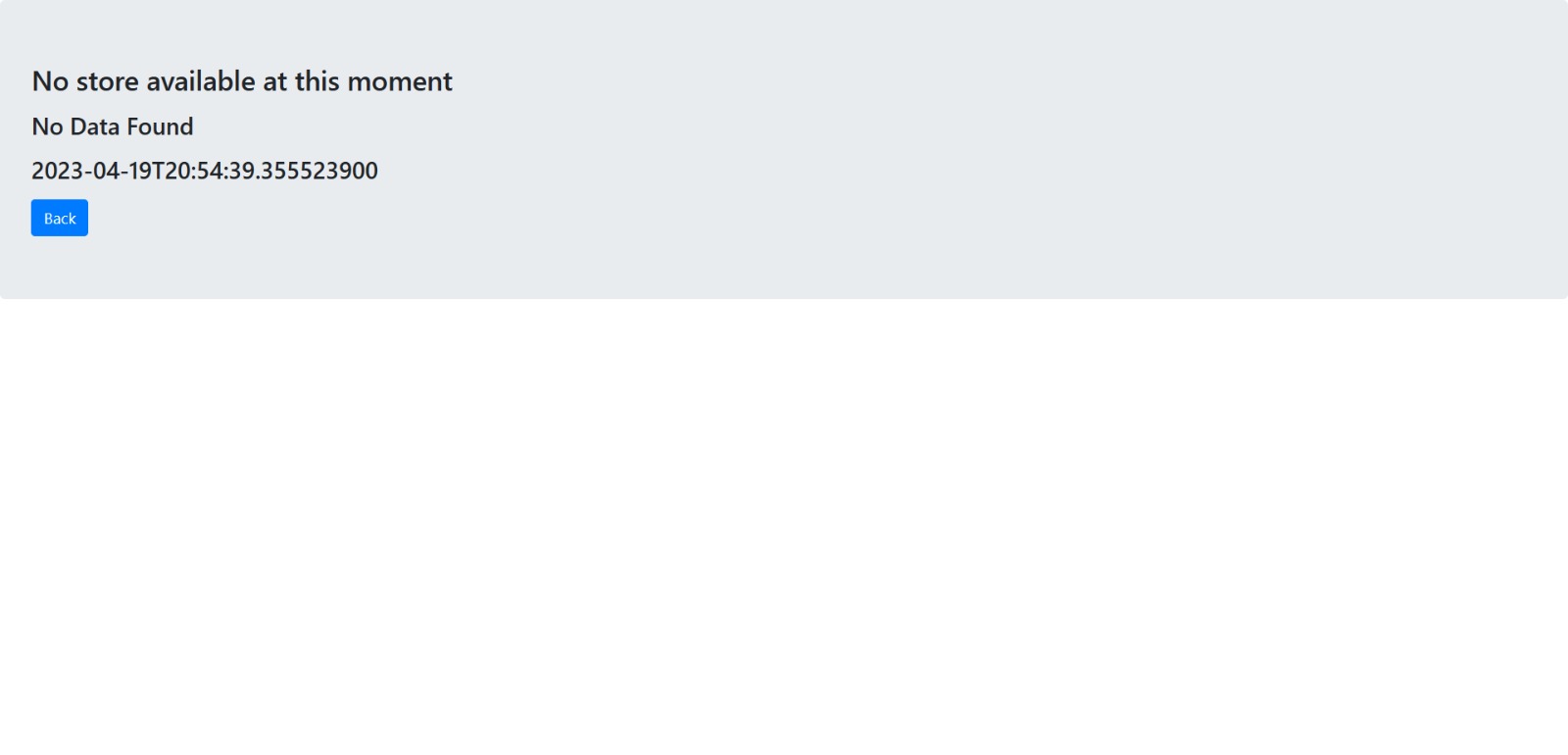


Figure 29 StoreNotFoundException in console



1. Add CSRF security to the application

Figure 30 login.jsp with csrf token to avoid CSRF attacks

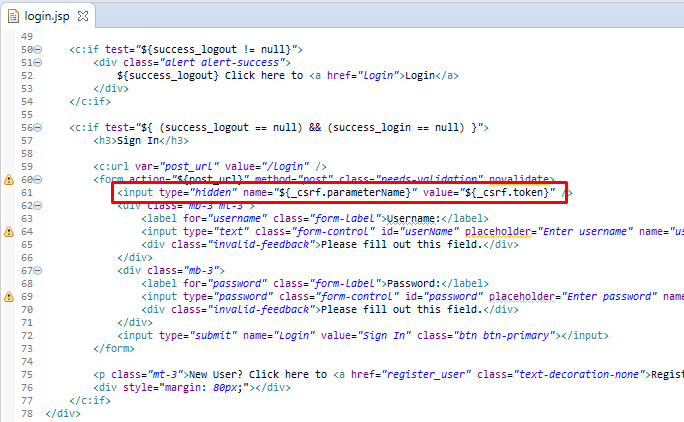


Figure 31 Security taglib used for header included in each webpage, authorization of register and login links before user login successfully

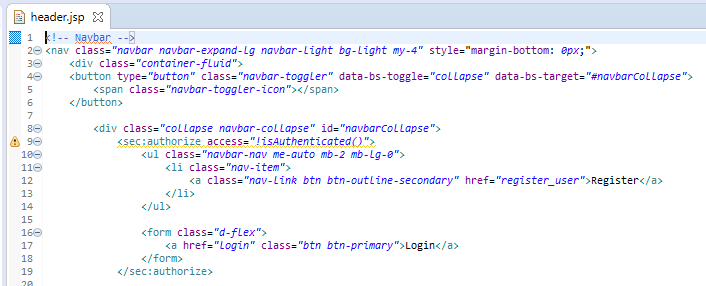


Figure 32 Security taglib used for header included in each webpage, authorization of register and login links after user login successfully



**Logging**

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

Figure 33 LoginControllerTest to test loading of login and registration page by testing onLogin() and showRegistrationForm() method from LoginController

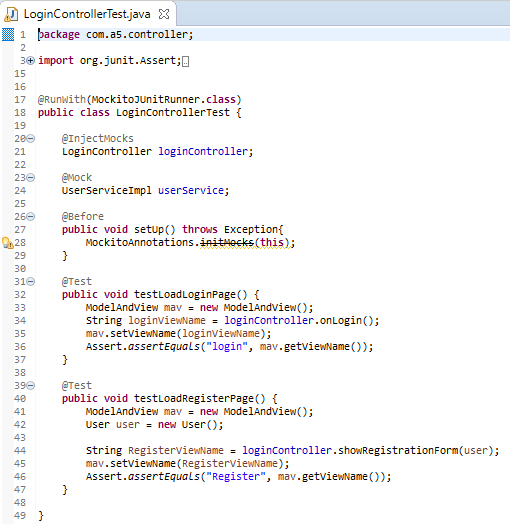


Figure 34 Output of LoginControllerTest methods

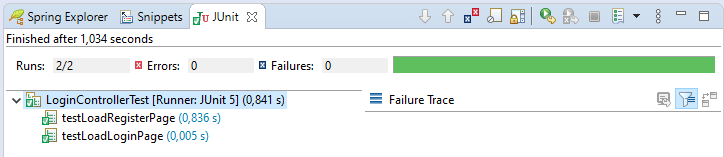


Figure 35 StoreServiceTest to test adding/save a store and searching for store by email by testing methods saveStore() and searchStore() from StoreService



Figure 36 Output of StoreServiceTest methods

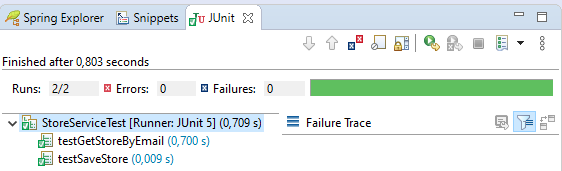


Figure 37 UserServiceImplTest to test adding/save user by testing method saveUser() from UserServiceImpl

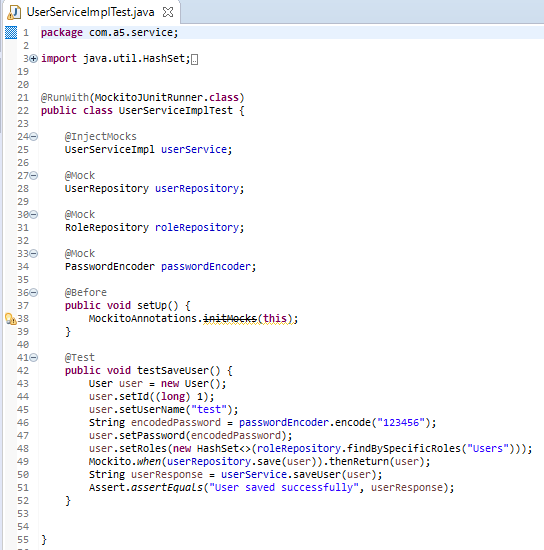
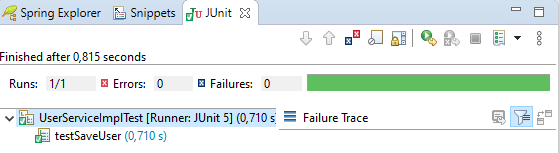


Figure 38 Output of UserServiceImplTest method



**Logging**

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

Figure 39 log4j2.xml source code

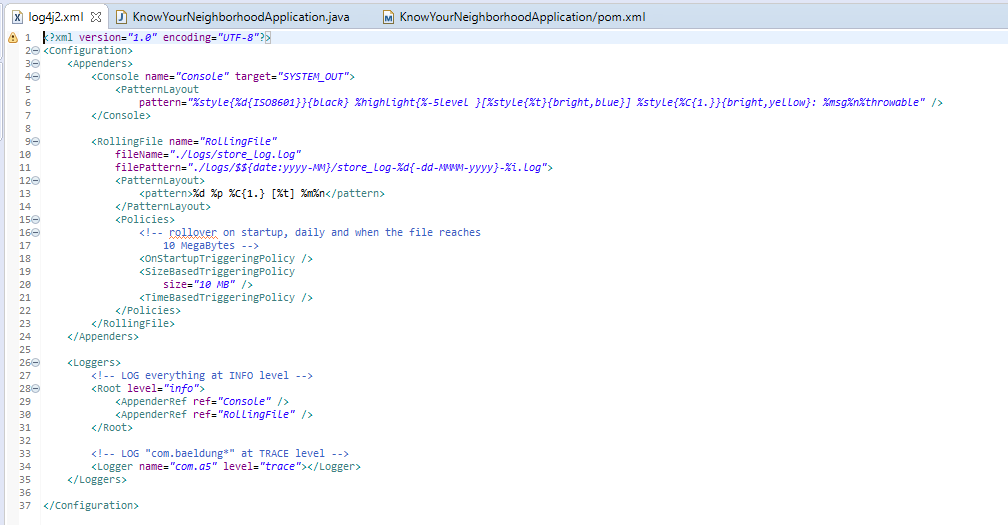
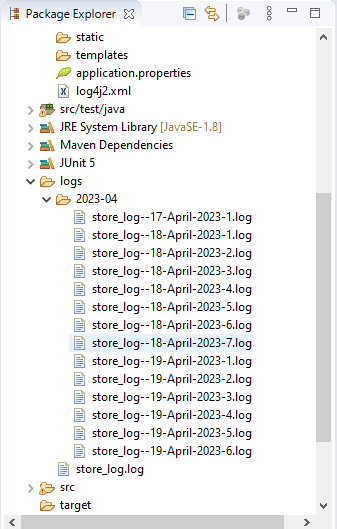


Figure 40 xml and log file directory



**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 41 StoreExceptionHandler.java class

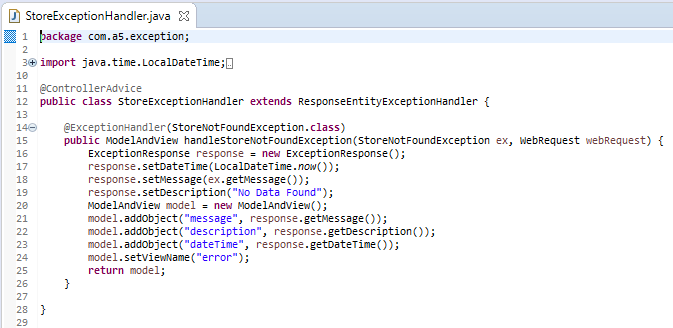
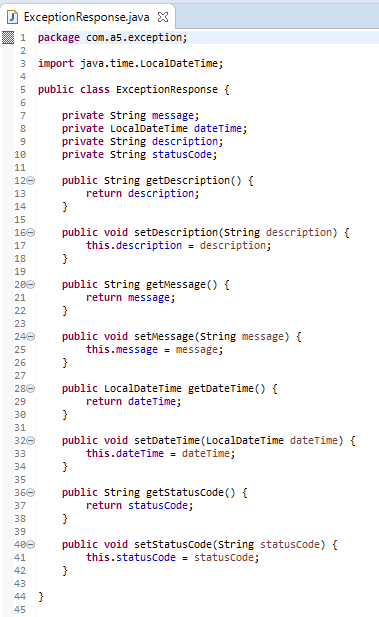
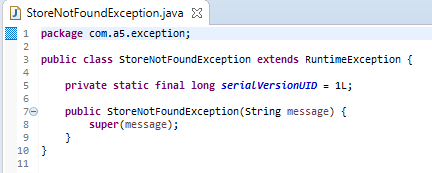


Figure 42 ExceptionResponse.java class



1. Create an exception class StoreNotFoundException.

Figure 43 StoreNotFoundException.java



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

Figure 44 StoreRepository.java

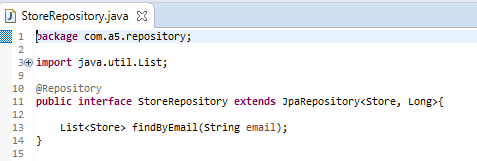


Figure 45 StoreService class searchStore method

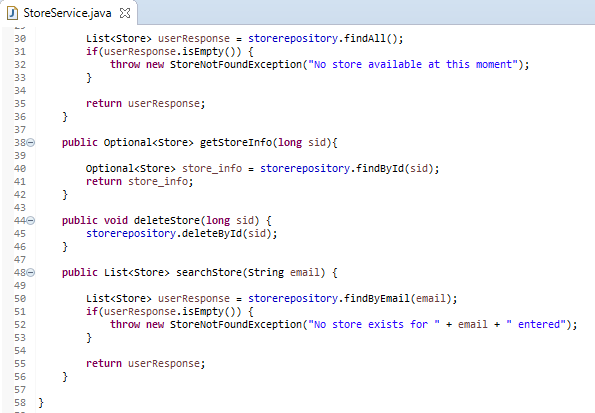
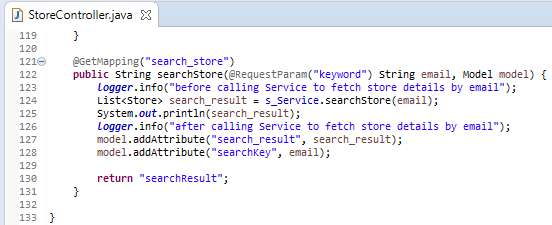
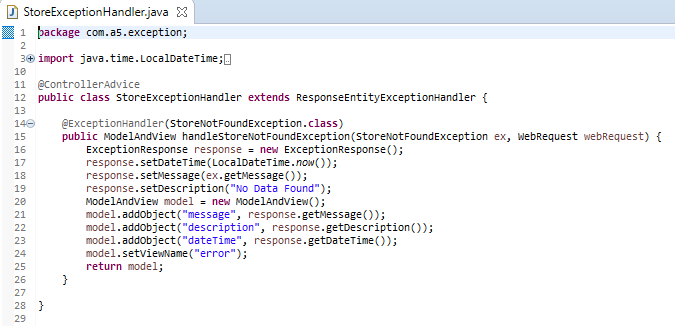


Figure 46 StoreController class searchStore method



1. Create exception handler specific to the controller.

Figure 47 StoreExceptionHandler.java class



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

Figure 48 StoreService class searchStore method with StoreNotFoundException if no store found for given email

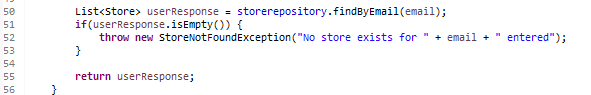
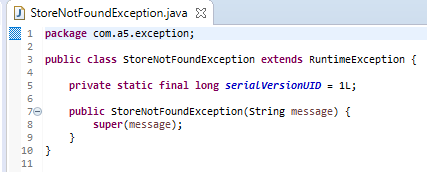


Figure 49 the message from StoreServiceClass will pass to StoreNotFoundException.java class



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

Figure 50 storeList.jsp VIEW\_STORE page with search bar to search store by email

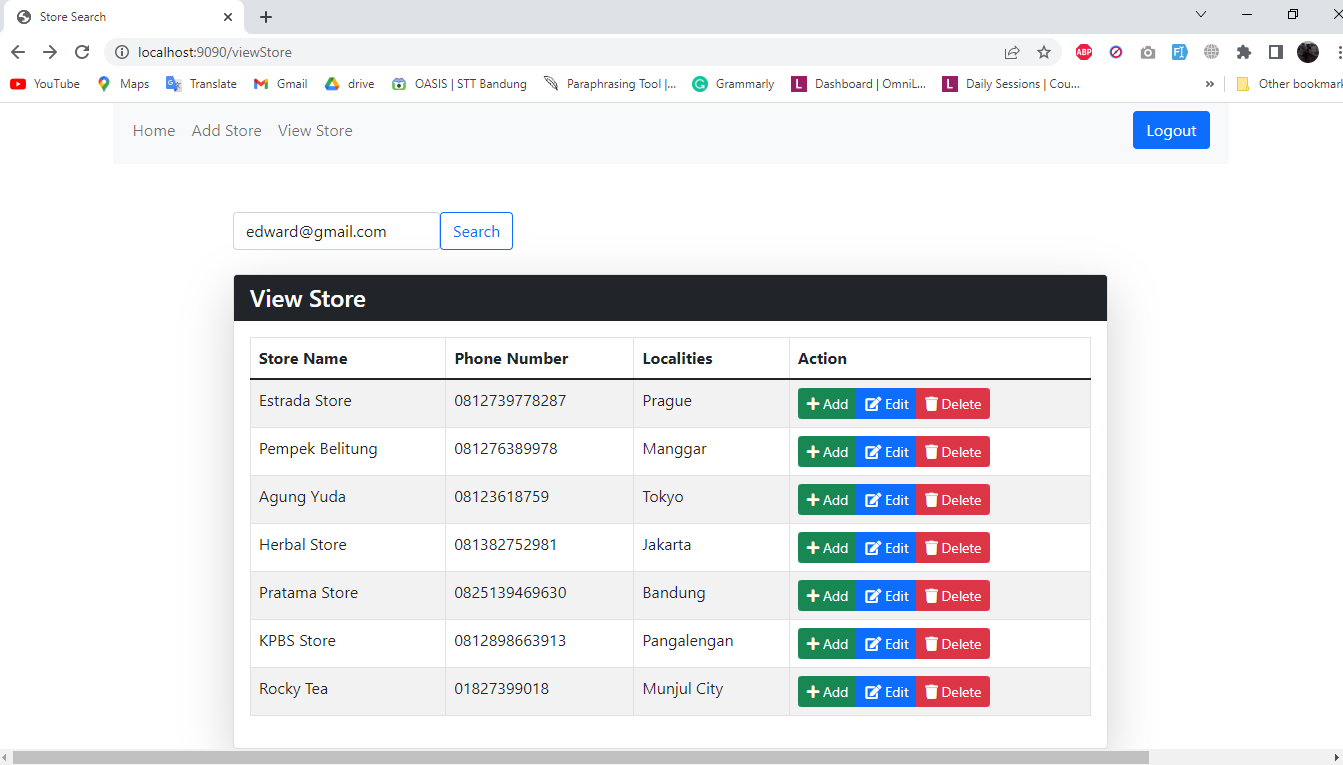
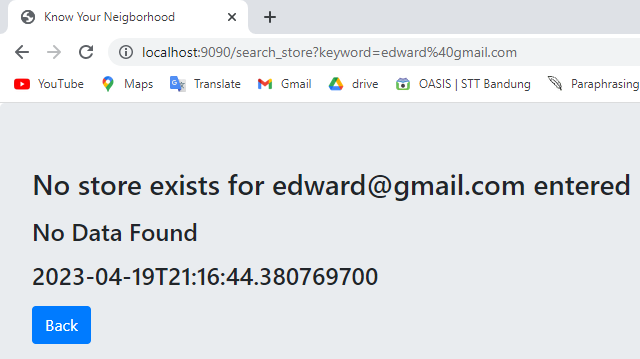


Figure 51 screenshot of error.jsp displayed if store not found for given email



1. Enter the URL http://:/stores?edward@test.com. This should route to view stores page and show the error message.

Figure 52 screen capture of error.jsp page for above action

