Business & Enterprise Programming

Documentation

The project that we are working on is a language learning app like Duolingo but for English to Welsh and Welsh to English. The program we are using is Eclipse and we use Java code for it. Also, we use HTML and CSS for the Website. This project will help the students to easily learn the languages.

For security on the login page, you need to create a username and a password. The username can be anything it does not have any restrictions but for the password to raise the protection for the account it must contain at least one capital letter, number, and special characters: (@, $, &...).

For the database, we are using MySQL. By using this database, it provides us with a structured and efficient way to store, retrieve and manage data using SQL commands and features.

For the project, we used as many classes as we needed to make it work perfectly. The AdminviewController class. This class is for managing the users, permission, and nouns in an administrative interface.

Autowired Dependencies: passwordEncoder: It autowires an instance of a password encoder for encoding passwords securely. testRepo, permRepo, userRepo, and nounRepo, are those autowired repositories for interacting with the database.

GetMapping:(adminview): This retrieves a list of users and nouns from the database adds them to the model and renders the "adminview" template with the list of users and nouns.

GetMapping:(adduser): Adds a new user to the provided username, password, and role. Encodes the password using the password encoder before saving it and redirects it to the admin view.

GetMapping:(deleteuser): It deletes a user with the provided username and clears the user's permissions before deletion.

GetMapping:(edituserpass): Updates the password of a user with the respective username.

GetMapping:(edituserrole): Update the role of a user with the respective username.

GetMapping:(newnoun, deletenoun. editnoun): The newnoun adds a new noun with the provided English and Welsh words along with the gender, the deletenouns deletes a noun with the provided English word, and the editnoun updates the details of a noun with the provided English word.

Public class LoginController: This controller is a part of the website where the user can access the login page so they can authenticate themselves. This class serves the purpose of rendering the login form and is also an important part of user authentication.

The public class Noun represents the storing information about the nouns including their English and Welsh along with the gender in Welsh. This class can be used to interact with a database to store and retrieve noun-related information.

The public class Permission stores information about the user permissions or roles. The class provides two constructors: a default and a parameterized constructor that initializes the "roleName" field. Those constructors allow the creation of instances of the Permission class with or without specifying the role name. This class can be used to manage user roles and permissions, allowing for access control and authorization based on user roles.

The public class Question represents a model entity for storing information about questions including their content and associated tests. The class provides a default constructor that initializes the Question object.

The public class User implements the UserDetails interface which is part of Spring Security and provides core user information.

The public NounRepository Interface extends the CrudRepository interface and offers CRUD (create, read, update, delete) operation for the Noun entity.

countByEnglishWorld (String englishWord): This method counts the number of occurrences of nouns with the specified English Word. It takes a String parameter englishWord representing the English word of the noun.

findByEnglishWelsh (String englishWord): This method finds a noun entity by its English word. It also takes a String parameter the same as the other one.

DeleteByEnglishWord (String englishWord): This method deletes nouns with the specific English word. It takes a String parameter representing the English word of the noun to be deleted. It also does not return any value.

getEnglishWordById(int id): It retrieves the English word of a noun by its ID.

getWelshWordById(int id): It retrieves the Welsh word of a noun by its ID.

These methods provide convenient ways to interact with the Noun entity in the database.

The public interface PermissionRepository extends the CrudRepository interface.

countByRoleName (String roleName): This method counts the number of occurrences of permissions with the specified role name.

findByRoleName (String roleName): This finds a permission entity by its role name.

These methods provide convenient ways to interact with the Permission entity in the database.

The public TestRepository interface extends the CrudRepository interface.

findById (int id): This method finds a test entity by its ID.

findByUsernameTaken (String username): This method finds all test entities associated with the specific username.

The public UserRepository interface extends the CrudRepository interface.

countByUsernmae (String username): This method counts the number of occurrences of users with the specified username.

deleteByUsername (String username): This method deletes users with the specified username.

findByUsername (String username): This method finds a user entity by its username.

The public Class FirstLoadNoun is a component responsible for loading initial data into the Noun repository (nounRepo).

NounRepository nounRepo: Autowired instance of the NounRepository interface for interacting with the Noun entity.

PostConstruct Method: The method is annotated with @PostConstruct indicating that it should be executed after the bean has been constructed and initialized.

Data Loading: The arrays of English words, Welsh words, and Welsh word genders are defined.

Data Structure: English, Welsh words, and Welsh word genders are provided as arrays of strings with corresponding elements at the same index representing a single noun entity.

The public FirstUserConfigurer class is another component responsible for setting up initial data upon application startup.

The PasswordEncoder passwordEncoder: Autowired repository for managing permissions.

The PermissionRepository permRepo: Autowired repository for managing permissions.

The UserRepository userRepo: Autowired repository for managing users.

Permission Setup: If the count of permissions with the role name ADMIN, USER, or LECT is less than 1, indicating that they don't exist, new Permission objects with these role names are created and saved to the repository using permRepo.save(new Permission(roleName)).

Data Structure: Permissions are predefined with role names: ADMIN, USER, and LECT, ensuring that they exist in the database. The admin user is set up with the username "admin" and password "password".

The public class JPARepositoryUserDetailsServiceImpl class implements the UserDetailsService interface.

UserRepository userRepo: Autowired instance of the UserRepository interface for interacting with user data.

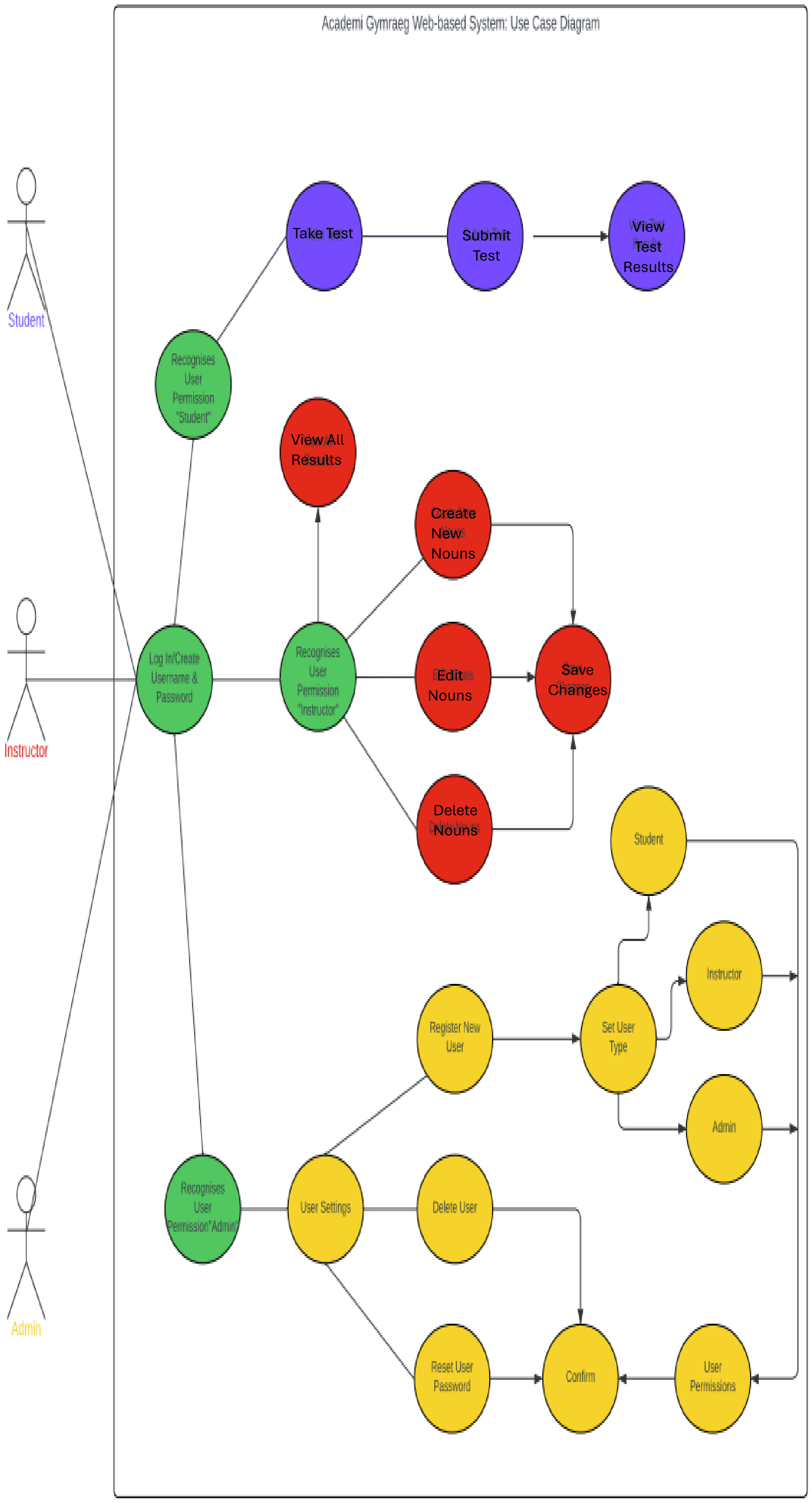
loadUserByUsername (String username): This method is overridden from the UserDetailService interface.

Data Structure: It retrieves a user from the repository based on the provided username and returns it as a UserDetails object.

The security system and its rules: The users are required to authenticate themselves before accessing protected resources. The system validates user credentials against stored user data that ensures the user is who they claim to be. Once the user is authenticated, they are granted access to specific resources based on their assigned roles and permissions. The roles and permissions are typically stored in a database and associated with user accounts. The passwords are securely encrypted using a password encoder before being stored in the database.

Use Case Paths

Student	🟣
Instructor	🔴
Admin	🟡
System Operations	🟢



Academi Gymraeg Web-based System: Use Case Diagram
Student
Instructor
Admin
Log In/Create Username & Password
Recognises User Permission "Student"
Take Test
Submit Test
View Test Results
Recognises User Permission "Instructor"
View All Results
Create New Nouns
Edit Nouns
Delete Nouns
Save Changes
Recognises User Permission"Admin"
User Settings
Register New User
Set User Type
Student
Instructor
Admin
Delete User
Reset User Password
Confirm
User Permissions