Credit Task 7.2C

COS30041 Creating Secure and Scalable Software (Swinburne University of Technology)

Name: Mustafa Mushtaq
Std ID: 101978603
Task 7.2C

Task 1: Research

Enterprise application can authenticate and authorize users based on their roles stored in a database table. This can be done using JDBC realm. The infrastructure to support the following functionality is provided by the glassfish server. When implementing JDBC realm, the database table is mentioned so that the glassfish server knows which table to access. Following important information is required when working with JDBC realm:

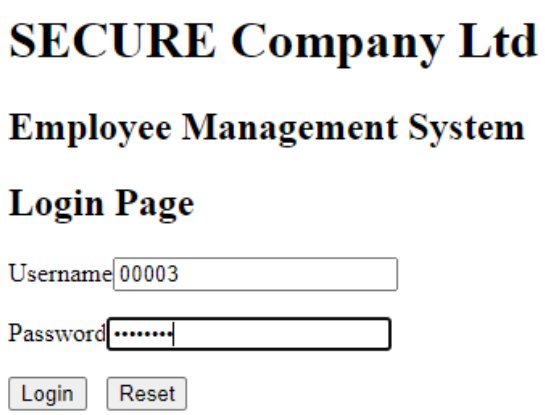
- User Table: This refers to the database table which contains logging credentials.
- Username Column: This column refers to the UserID column in the database table which contains the user Id.
- Password Column: This refers to the User password column in the database table which contains the user password
- Group name column: It refers to the column in the database table that contains the list of groups for this realm.
- Password Encryption Algorithm: Denotes the algorithm for encrypting the password in the database table for security purposes.

When logging in, the id and password entered by the user is checked against the UserID and Password data stored in the database table. It also considers the APPGROUP so users cannot access the admin menu and hence the options are limited. If the attempt is successful, only the data for that UserId is returned which can be modified or edited.
This is the most efficient and reliable way of authenticating users based on their role since most of the functionality is provided by the glassfish server which means there is a minimal requirement for coding.

Task 2: Programming

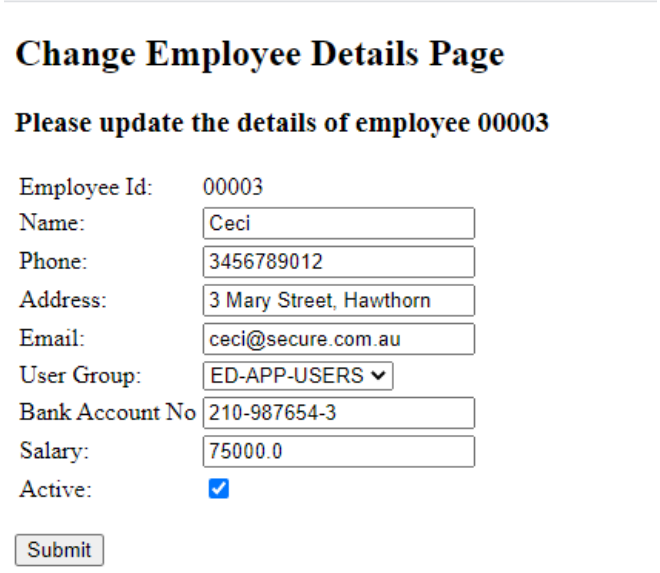
The users are not authorized using the login credentials stored in the database. The EMPID and PASSWORD are used for logging in, and the APPGROUP is used to check if the user-role can have access.

EmpID: 00003
Password: 3333333
App-Group: ED-APP-USERS



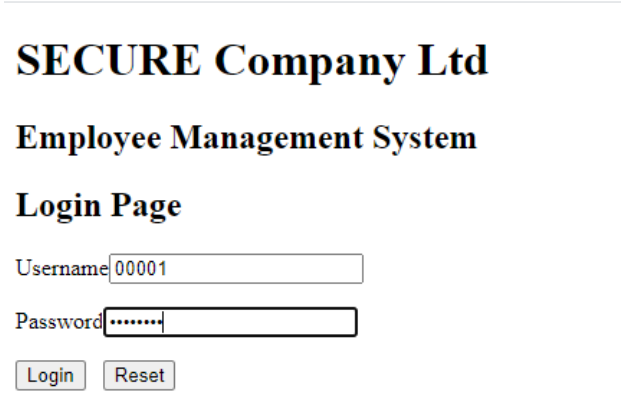
Since we are attempting to log in to the user main menu, only the user can access it and EMPID: 00003 is a user with App-Group: ED-APP-USERS.

On selecting option 1: Change Your Details;



The data for EMPID is returned since EMPID: 00003 logged in.

If the user with another App-Group tries to login in the user main menu, then the error is given. 00001 is an admin trying to login in user.



Hashing Algorithm

In this algorithm, the password stored in the database is converted into a hash. This is basically a series of digits and letters that might not make any sense when looking at it. This is a one-way generator, and once converted to hex, it cannot be converted back. Hence when comparing passwords upon logging, the entered details are first also converted into a hash code and compared with that in the database.

First the string is converted into Bytes:

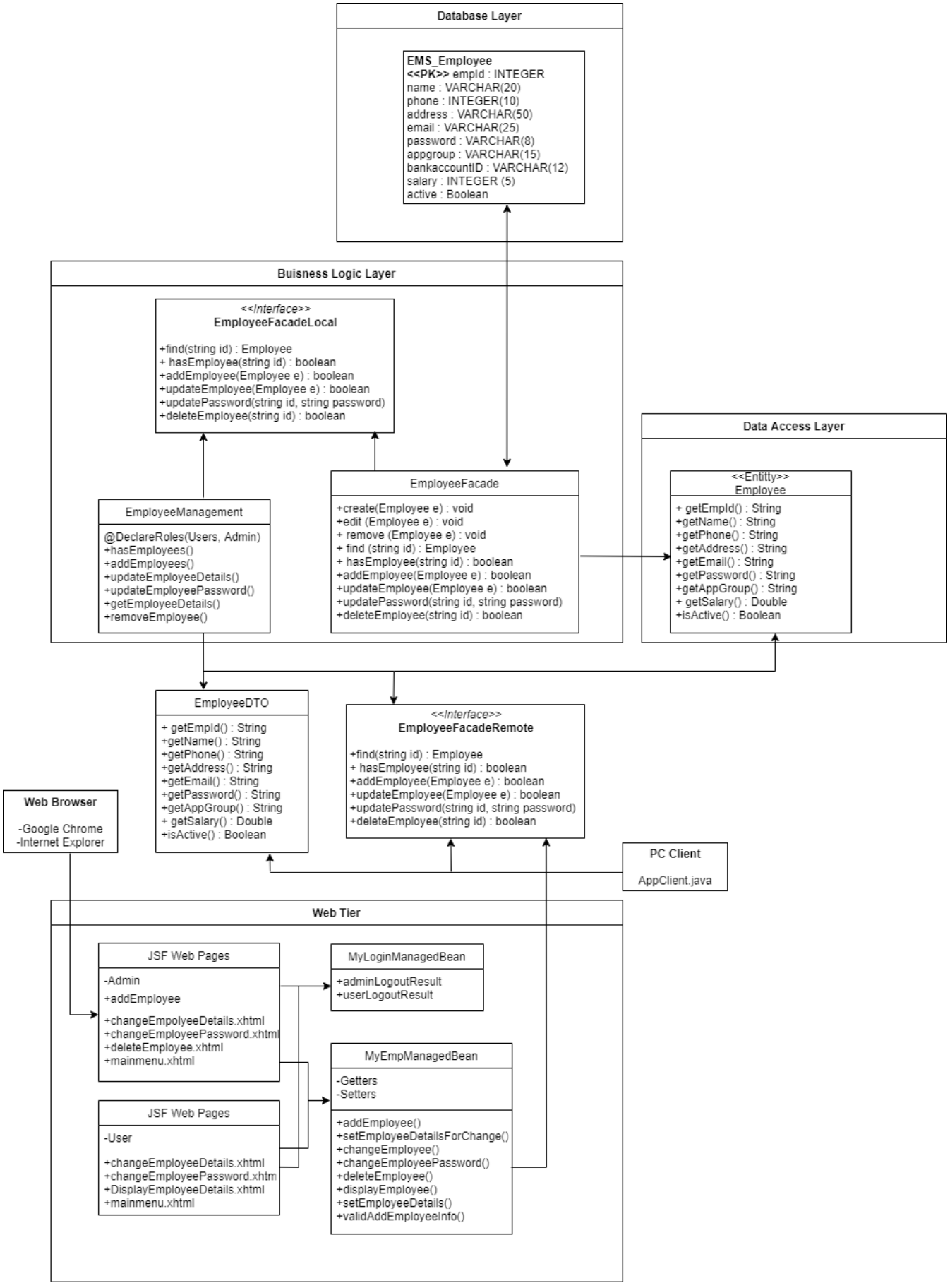
```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] encodedhash = digest.digest(
  originalString.getBytes(StandardCharsets.UTF_8));
```

Then we use a custom byte to hex converter to get the hashed value in hexadecimal

```
private static String bytesToHex(byte[] hash) {
    StringBuffer hexString = new StringBuffer();
    for (int i = 0; i < hash.length; i++) {
    String hex = Integer.toHexString(0xff & hash[i]);
    if(hex.length() == 1) hexString.append('0');
        hexString.append(hex);
    }
    return hexString.toString();
}
```

The value obtained is then stored in the database under Password column.

Task 4: Design



There are roles defined for each type of user. In our case, it's a simple User and an Admin. There are different web pages for different users in this program. This is so that when the user logs in, he will only be able to access the user's main menu and its related pages. While the admin will only be able to view the admin pages. This data is stored in the database, under the APPGROUP data column. JavaEE allows us to use to declare roles and roles allowed for specific users. This can be implemented on methods, so that only the specific users can use that method. An error is given when other user group tries to access that same method.

Another newly component included in this program is the addition of code that takes a string of password, converts it to byte and ultimately into a hex code. This is then stored in the database, under the password column for security reasons. In order to log in, the user must enter the password, which is also converted into a hex code and compared against the password in the database (which is also in the hex code).

Task 5: Learning Journey

Task 7.2C is a relatively different task compared to other tasks where the main purpose was to implement CRUD operations using various techniques while also addressing security issues. Here, the focus was security and authenticating and authorizing users based on their roles stored in the database which in our case could be either an Admin or a simple User. There are several ways to authenticate a user which are simple to understand but very inefficient. Developing an enterprise application is a vast topic and is often very hard to research about it on the internet. This is when the lectures and tutorials came in handy, where most of my queries were satisfied.

Note: I couldn't complete this task. The algorithm for implementing hash code was not successful although I was able to grasp the complete knowledge of it. But due to time limit this wasn't possible.