Chadi ElHakim 202000289 cme16

Zein Zebib 202002334 zhz07

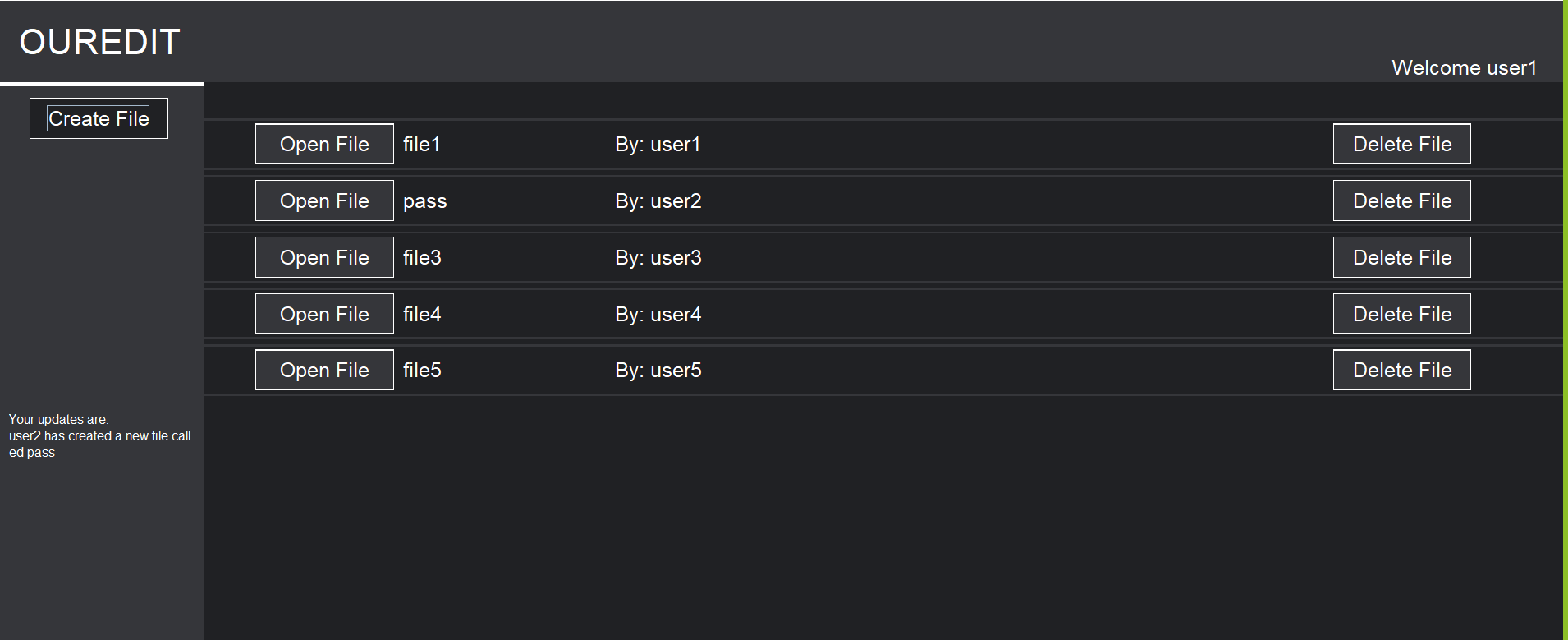
Mohammed A. Asaad 202000933 maa331

**Progress Report**

**Implemented Functions:**

**Create:**

First, creating a file will create a new entry in the sql specifying the file creator and the file name. Also a file will be created on the specified directory in the code.Finally, a message will be broadcasted to other users informing them that a file is created along with the file name and creator.

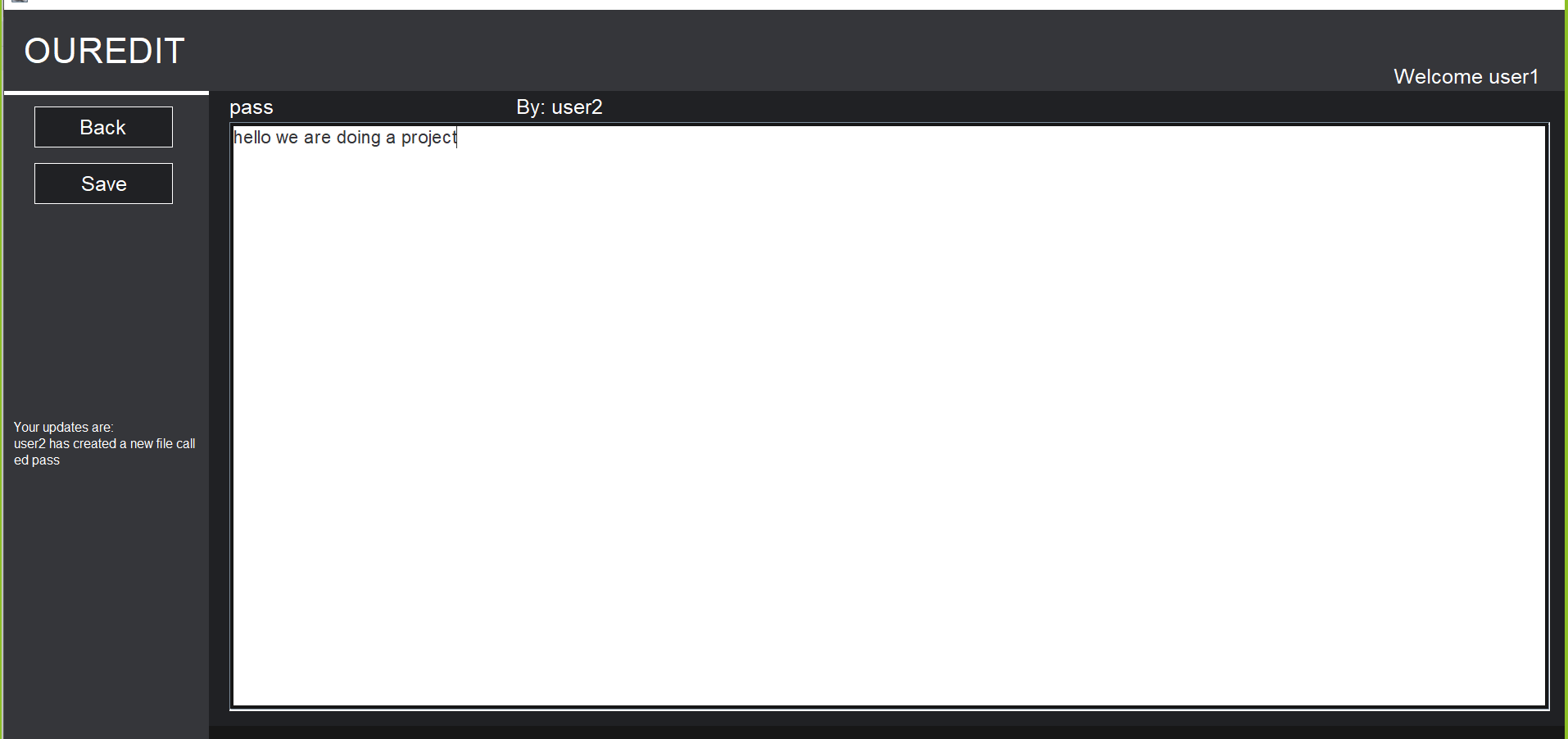
****

**Delete:**

Only the file creator can delete his files,and when someone other than the creator wants to delete the file, the creator will be notified of this. When a file gets deleted by the creator a message will be broadcasted to all users saying this file was deleted.

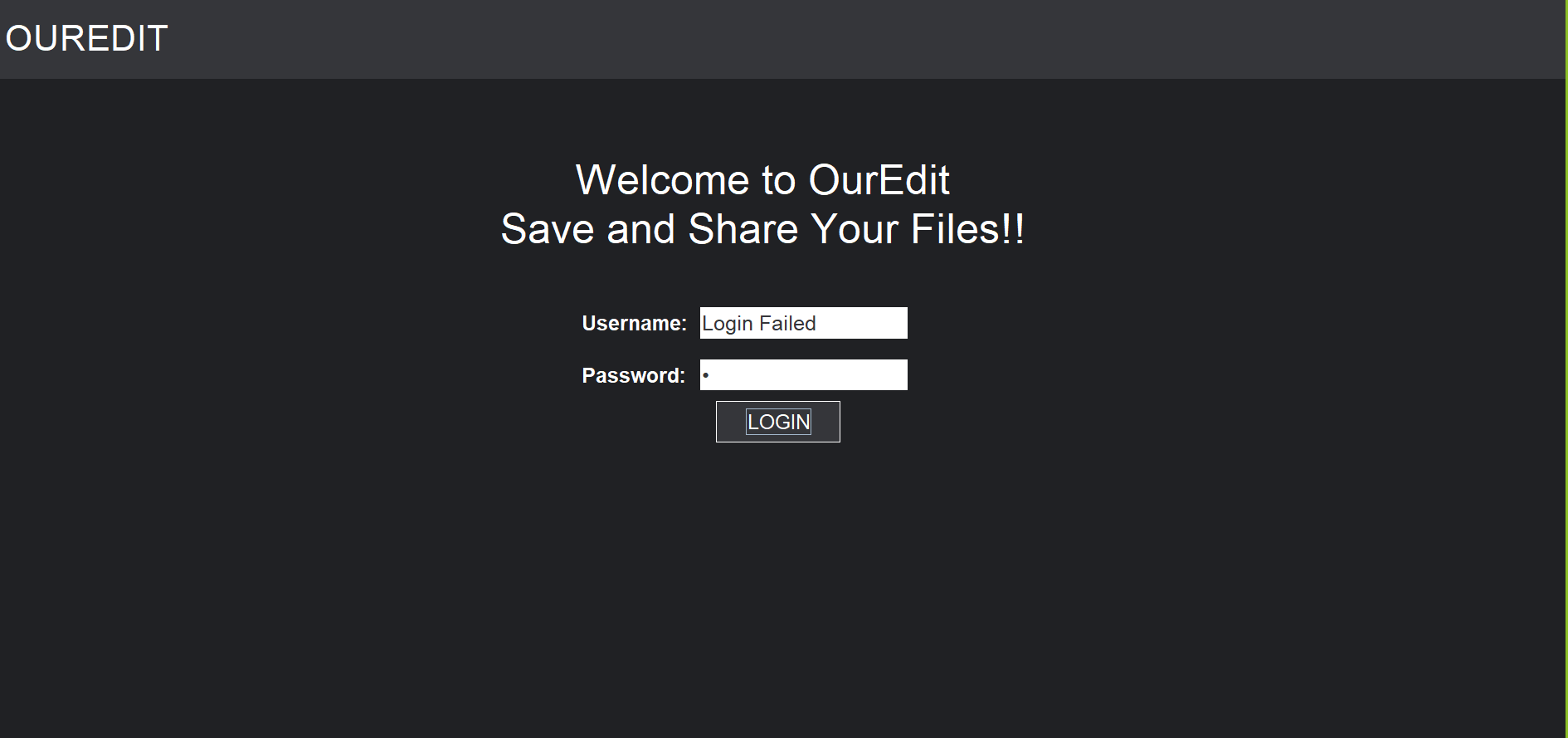
**Edit:**

One person at a time can edit the file and you have to press the save function in order for your edits to be saved.



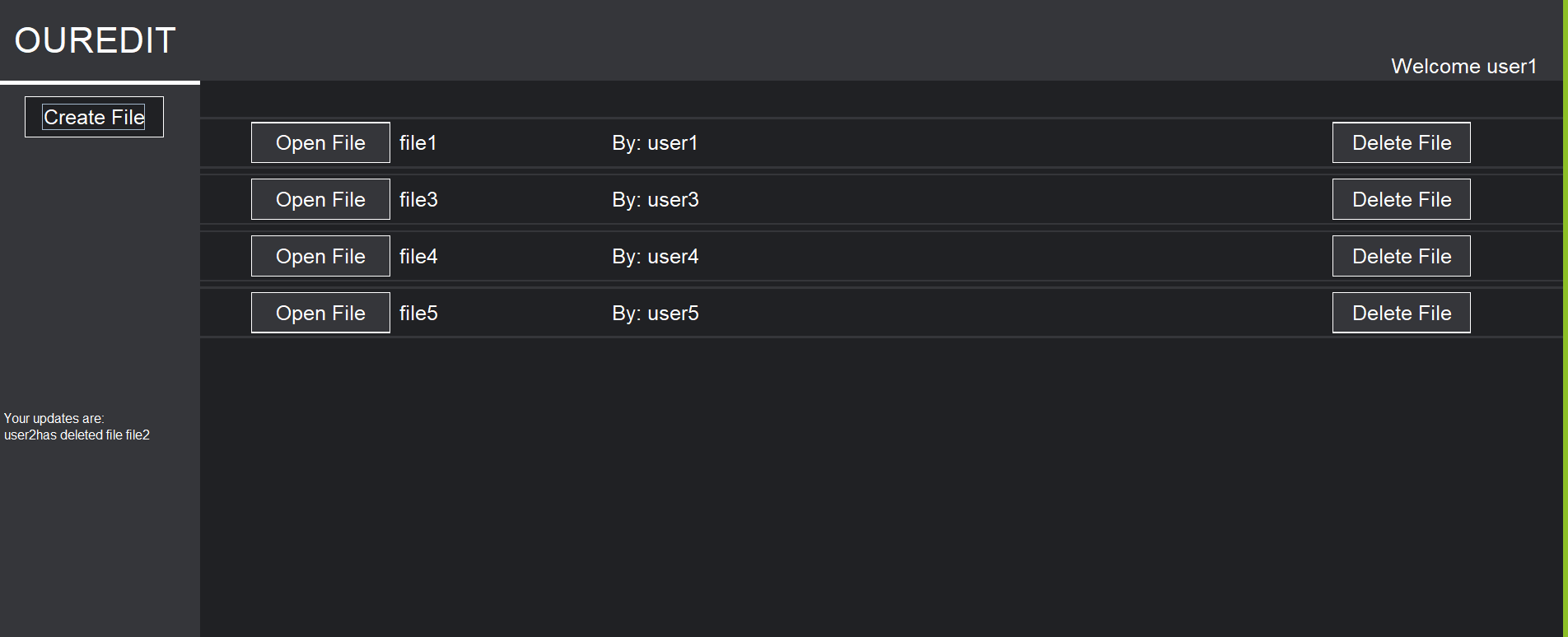
**Login Function:**

You enter the username and password, and if the username or password is invalid the login will fail.

**Update:**

Every time a file is created or deleted, and upon refreshing, the entries on the GUI will be modified accordingly and display only the remaining files. For example if user1 deleted a file, and user2 was online using the app and he refreshed, he would be notified.

Moreover, whenever an update is called an array of the file names and creators will be extracted from the SQL database and be sent to the client.

****

**Client Code Structure:**

The code begins with private static arrays, strings, buttons, and labels. This step is done in order to call these objects in different functions.

The main parts of the GUI come from two functions; frontpage() and fileLocation() each of them have their own lines to define objects for the design.

The frontpage is just where the user has to login and fileLocation is where the user can open, edit, create, and delete files.

Every Jbutton in the code has an actionlistener added to it, hence having the class GUI implement an action listener. This way whenever a button is pressed the new GUI() object is initiated and goes to actionPreformed() function where we will search for the source for the actionevent and run the desired operation.

**For example:** When the create file button is pressed we go to actionPreformed() and after going into the correct if statement we call createFilePressed() this will make the JTextField and two JButtons appear. When the create button is pressed a function createToServer() will be called.

Server communication:

Each time the client was to communicate with the server a function will be called. updateFromServer, editToServer… Each one of these functions opens a connection and sends a socket with the needed information to the server. These functions also have lines of code to read the replies from the server

**Server Code Structure:**

The server code follows the same multithreaded implementation as phase one. For each action we want to do to our file, there is a function named for it. Before that function is called we read the lines of the socket and depending on the function we want to perform we read a certain sequence and call the function we want. The functions editFile,openFile,.. All return a string that will be sent to the client. The first line of this reply will include a simple message to tell if the action was performed successfully or not. This has also helped with the design process as we were able to tell that everything was going fine.

The server code handles all the communication that is needed with the SQL databases, it updates the databases in case a file is deleted or added and it retrieves this information and sends it to the clients.

Another feature on the server is that it has three ArrayLists that store files that have been created, deleted, and want to be deleted. This information is stored on the server and it gets broadcasted to the needed users.

**Communication Protocol:**

Four main headers were sent with every socket; Function, File-Creator,FIle-Name, and Username.

Sometimes these headers are not utilised in the action we are trying to do and they are left with nothing.

Username is the user that is sending this socket which can differentiate from the FIle-Creator.

Types of functions: update, update2, login, create, edit, open, delete, deleteReq

Depending on each function that is called the server will handle the information differently and sometimes will have to read additional headers.

Line-Count and File-Content are used when the user wants to edit the file. Line count is used to determine the amount of times we have to use .readline() with the file content.

Password is used when we are logging in.

After each socket is sent the server will reply with a certain statement and the client will be able to interpret it and see what is going on.

**Group Roles:**

Zein Zebib worked on the GUI and the communication protocol.

Chadi worked on linking the client and the server together and he helped with working on the SQL.

Mohammed did the SQLdatabase and linked it to the server using java.

We all agreed on not inserting any percentage for any group member since it was a cumulative process where each one of us was learning from the other.

**References:**

SQL:

* The tutorial video done by the instructor
* The pdf provided by Jad Matta
* <http://www.w3schools.com/sql>
* <https://stackoverflow.com/>

Any question relating to editing files or simple java questions were found from here.

GUI:

* <https://www.youtube.com/watch?v=iE8tZ0hn2Ws>

This video provided the basis of how to implement action listeners and how to structure the GUI.

* <https://stackoverflow.com/>

Any question needed to find how to implement a design feature was found from here.