CS 2316

Homework 9b – Chat

Due: Thursday, July 18th, before 11:55 PM
Out of 100 points

Files to submit: **1. HW9b.py**

<u>This is an INDIVIDUAL assignment!</u>
Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:
- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:
- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- ***Do not wait until the last minute*** *to do this assignment in case you run into* problems.
- **Read the entire specifications document before starting this assignment.**

Premise

This homework is part two of the previous GT Login assignment. You will add on to the login functionality that you built to create two more GUI windows that use a database to have chat functionality. As you know, chats are widely used across the internet and maybe this can help you understand simple functionality of a chat application. [It is also a simplified version of a past CS4400 final project, so if you can do this homework you should be ready for the CS4400 final project!]

Although this program does not require knowing much new information, it does test SQL query and insertion abilities as well as coding practices to create two more GUI pages that can be opened, closed, or updated at the click of a button. **This new information may take some understanding, so please do not wait until the last minute to begin this assignment.**

Database Format Information

The database for this part of the assignment only concerns a different table of messages, called GtChat:

```
CREATE TABLE GtChat
(MessageID  INTEGER       PRIMARY KEY AUTO_INCREMENT UNIQUE,
 User      VARCHAR(25)   NOT NULL,
 Message VARCHAR(40)  NOT NULL)
```

The code above was used to create the table. As can be seen, User and Message fields are NOT NULL, meaning you must insert something into each field. Note that the "User" field is the person's USERNAME (not first or last name, which are stored in a different table.)

Further Development

You will be writing on top of your already written part A. **Your old code is probably not perfect so feel free to update it if needed!**

The new code will create a chat page where users can input new messages and see some of the most recent ones. It will also create a statistics page if called upon that displays some of the more avid users and their stats.

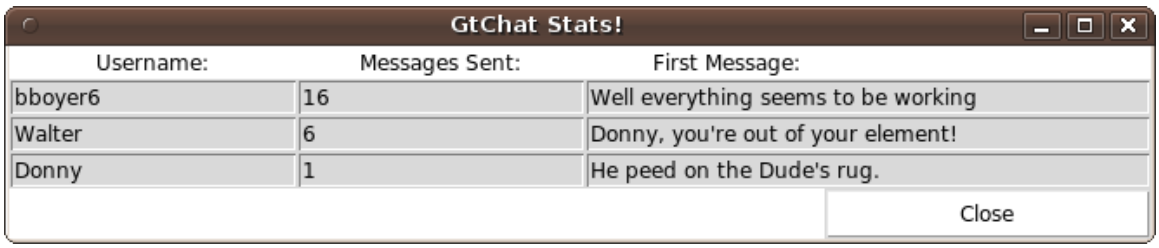
Method Name**: ChatPage**

This method is called to show a new chat window. It should be called when you successfully login from the previous homework's login page. You should create a GUI which looks like the one above (possibly in your __init__ method). You may generate this window however you wish, but it is *highly* suggested that you use the grid layout manager. Things to note about the GUI you see:

- The top image is the same image as in your other GUI windows from part A. You can continue to use the same image from the previous homework, or switch to a different image if you wish, but keep it GT related!
- The bottom label on the left side (circled in red above) is the FIRST NAME that the current user (using your program) entered when registering. If no first name was put in (either blank or NULL in database), the first name that should be displayed is simply the text “User” instead of the blank or NULL. It is NOT a username. In this example the username "Walter" had a blank first name, so we used the string "User" instead of the blank for this label. If the username "Walter" had a firstname of "Jay" this label would instead be a "Jay" instead of "User".
- The readonly entry box to the right of the user’s first name (or “User” if no first name is entered, see red circle) should have the user’s username ("Walter" in this example).
- The other username entries are width 20 and state readonly and display the usernames of the users who submitted the last three messages.
- The message entries are width 40, state readonly for the first three and state normal for the last one to allow a message to be entered. They will probably need to span multiple columns to get the GUI to look right. **The top three should be already filled in with the 3 most recent chat messages when the window opens! (If 3 messages do not exist, they should be blank.)** The bottom of the three should be the latest message, the middle entry should have the 2nd newest message, and the top one should have the 3rd newest message. (So that if a user types a new message, it should first appear in the bottom box, then move up to the middle box, then move to the top box before scrolling off the top...)
- Clicking on the “Chat Statistics” button should call the StatChatPage method.
- Clicking on the “Logout” button should bring back the login page.
- Clicking on the “Update Messages” button should update the GUI with the most recent messages by calling the updateClicked method. **This page does not need to be automatically updated.** It is ok to just show the messages that were the latest three the last time the “Update Messages” button was pressed. (But if you want, feel free to use the .after() method to automatically update the window every second...if you do this, you should STILL have a working "Update Messages” button for users who want to update faster than once per second!)
- Clicking on the “Send Your Message” button should insert your username and message into the database as the next entry in GtChat and re-update the GUI with that new message by calling the sendMessageClicked method (perhaps followed by calling the updateClicked method, if your sendMessageClicked method does not do that).

Method Name: **StatChatPage**

This method will simply show a new GUI window for viewing chat statistics. **This will not close the chat application window! This window opens in addition to the already open chat page.** The GUI should look something like this:



You may generate this window however you wish, but it is *highly* suggested that you use the grid layout manager. Things to note about the GUI you see:

- Everything is width 20, except the First Message entries which are width 40.
- All entries are state readonly
- The last column is for the FIRST message the user ever posted. This is easier to obtain than you might believe using only SQL, just play with it.
- Clicking on the "Close" button closes (only) the chat statistics page.

This method also contains the code to fill out the statistical information. The statistics obtained are from the three users that have sent the most messages. The user with the most sent messages appears at the top with the others following underneath. If two users have the same amount of messages sent, the users should be sorted alphabetically by username. The GROUP BY and ORDER BY mysql functions may be of use here and remember that you can order by more than one field by separating the fields with a comma. In addition, to streamline query speed it may be useful to use the LIMIT mysql function to only gather the top 3 users instead of every user who has ever entered a message in the system. **This statistics page does not need to be dynamically updated.** It is ok to just show the statistics that were correct when you pressed the Chat Statistics button to show the window. (But if you want, feel free to use the .after() method to automatically update the statistics page every second...)

Method Name: **updateClicked**

Parameters:

None

Return Value:

None

Description:

This method will take the last three messages from the database and fill them into the GUI. It may be useful to use the LIMIT mysql function so as to streamline the query process so you only grab as many messages as is necessary. The entries from the chat

page are updated so that the most recent message is at the bottom and least recent (of the three messages) at the top. This allows for a new message to appear closest to the message entry box from which it was taken. If the database has been cleared recently, and you don't find 3 messages, leave one or more entries blank!

Method Name: **sendMessageClicked**

Parameters:

None

Return Value:

None

Description:

This method will take the message entered by the user, insert it into the database, and update the GUI with the new message(s). The entered message should first be checked to make sure that it is not blank and that it is not greater than 40 characters. If it fails either of these tests, a message box should pop up to inform the user of the error. If it passes both tests you should insert the message into the database, clear the user message entry to make room for a new entry from the user, and update the GUI with the most recent messages again. (The easy way to update the GUI is to call the updateClicked method...) Be sure to .commit() the database when you’re done inserting data.

Grading:

You will earn points as follows for each piece of functionality that works correctly according to the specifications.

Login GUI		**5**
Login GUI works (as specified in previous HW)	5	
Register GUI		**5**
Register Page works (as specified in previous HW)	5	
ChatPage GUI		**25**
Logo on top	4	
Label at bottom left is user's first name or "User:"	5	
Username entries proper and filled	5	
Message entries proper and filled	5	
Buttons all linked correctly	6	
StatChatPage GUI		**30**
Does not close Chat/Opens separately	5	
Top users selected correctly	5	
Top users ordered correctly	5	
GUI elements proper and filled	10	
Close button works correctly	4	
updateClicked()		**10**
Updates gui correctly	5	
SQL functionality is correct	5	
sendMessageClicked()		**25**
Checks for blank entry (Messagebox)	5	
Checks for long entry (Messagebox)	5	
Enters in new message	5	
Updates GUI after insertion	5	
Deletes entry box for re-use	3	
Commits db when done	2	

Extra Credit: +5 points if the program automatically updates both the chat window and the statistics window every second (without causing an exception when the windows are closed or program is exited...)