CS 2316

Homework 9a – GT Thrift Login

Due: Wednesday, April 13th, before 11:55 PM

Out of 100 points

Files to submit: 1. HW9.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 the first of a two-part assignment in which you will do some real world applications with databases and GUIs. For this first part of the assignment, you will be building a two page GUI that offers the user the chance to login, or in the case that the user does not have an account yet, register and then log in. You'll find this quite applicable to building programs with user-accounts, and your future CS4400 project option.

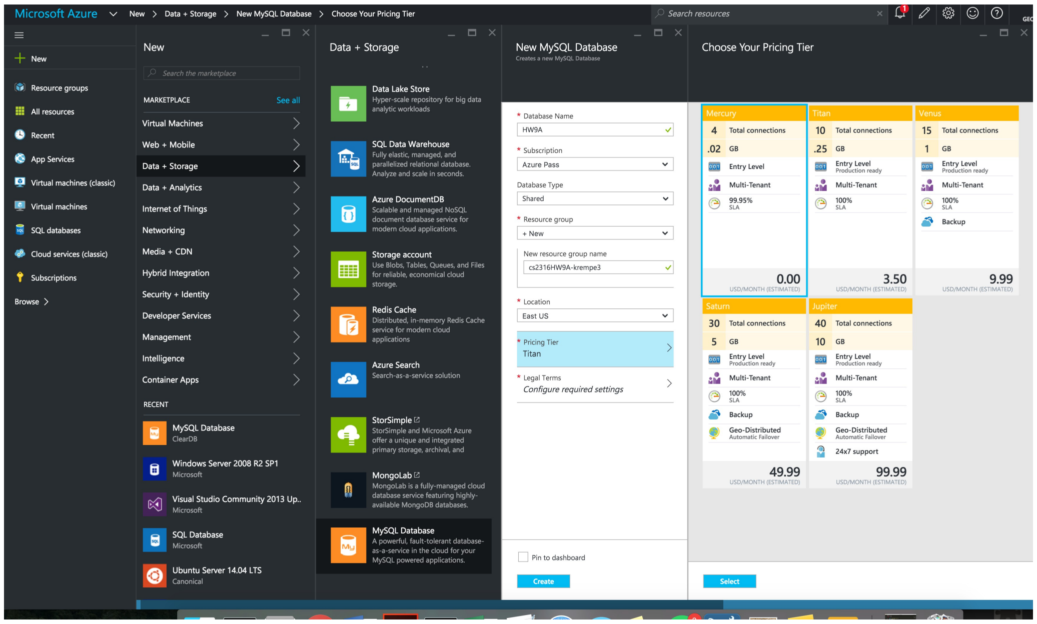
Although this program does not require knowing much new information, it does test simple SQL query and insertion abilities as well as new coding practices to create two GUI windows that can be opened and closed at the click of a button. **This new information may take some understanding, so please do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

Similar to GT Thrift Shop this homework will have different users that can buy and sell items using the GUIs created. The first part of this homework focuses on registering new users and logging into GT Thrift Shop.

Database Format Information

In Azure create a database similar to Lab2 with the following criteria:

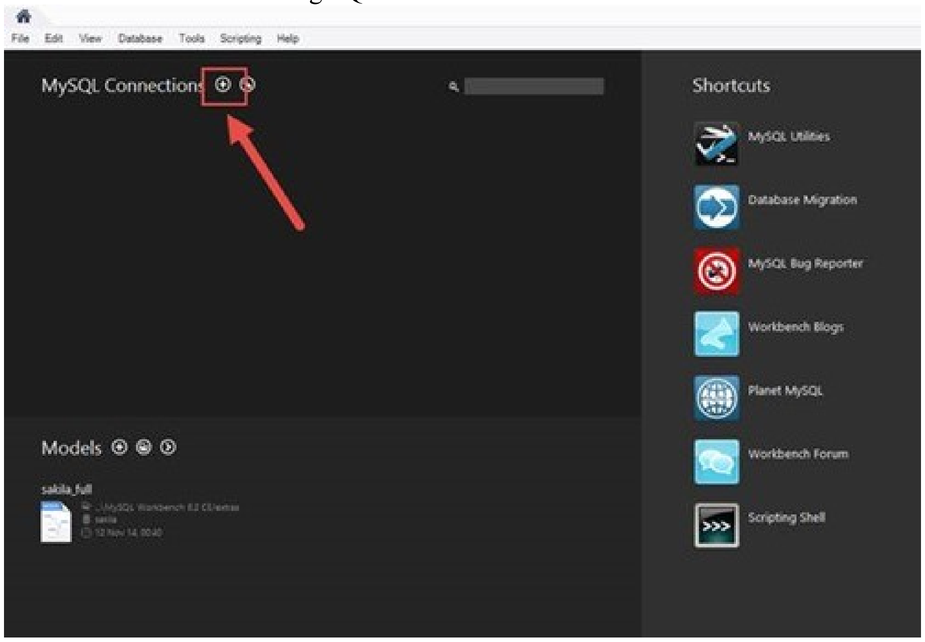
Database name: HW9A
Resource Group: cs2316HW9A-gtusername
Pricing Tier – Free



Database Name: HW9A
Subscription: Azure Pass
Database Type: Shared
Resource group: + New
New resource group name: cs2316HW9A-krempe3
Location: East US
Pricing Tier: Titan
Legal Terms: Configure required settings

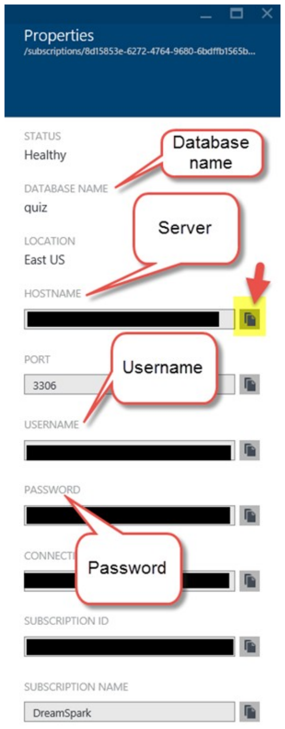
Tier	Total connections	GB	Features	USD/MONTH (ESTIMATED)
Mercury	4	.02	Entry Level; Multi-Tenant; 99.95% SLA	0.00
Titan	10	.25	Entry Level Production ready; Multi-Tenant; 100% SLA	3.50
Venus	15	1	Entry Level Production ready; Multi-Tenant; 100% SLA; Backup	9.99
Saturn	30	5	Entry Level Production ready; Multi-Tenant; 100% SLA; Backup; Geo-Distributed Automatic Failover	49.99
Jupiter	40	10	Entry Level Production ready; Multi-Tenant; 100% SLA; Backup; Geo-Distributed Automatic Failover; 24x7 support	99.99

Connect to the database using SQLWorkbench. Click the + to add a connection to the MySQL database.



Fill in the following From Azure Properties of you MySQL Database HW9

- Connection Name: HW9A
- Hostname copy and paste it from Azure
- Port copy and paste it from Azure
- Username copy and paste it from Azure
- Then **Test the Connection**



Properties
/subscriptions/8d15853e-6272-4764-9680-6bdffb1565b...

STATUS
Healthy

DATABASE NAME
quiz

LOCATION
East US

HOSTNAME
[redacted]

PORT
3306

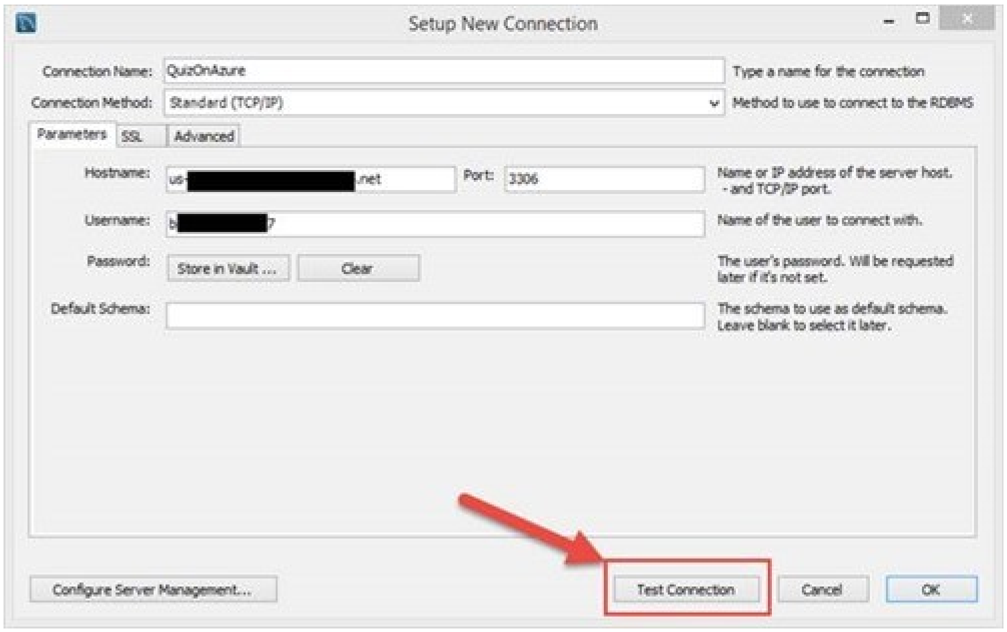
USERNAME
[redacted]

PASSWORD
[redacted]

CONNECTI
[redacted]

SUBSCRIPTION ID
[redacted]

SUBSCRIPTION NAME
DreamSpark



Connection Name: QuizOnAzure — Type a name for the connection
Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS
Hostname: us-[redacted].net Port: 3306 — Name or IP address of the server host. - and TCP/IP port.
Username: b[redacted]7 — Name of the user to connect with.
Password: — The user's password. Will be requested later if it's not set.
Default Schema: — The schema to use as default schema. Leave blank to select it later.

Then **name the schema hw9a** and click Ok if the Test Connection worked successfully.

The database for this part of the assignment will have 1 table of users, conveniently called User.

```
CREATE TABLE schemaname.User
            (Username                   VARCHAR(25)    PRIMARY KEY,
            FirstName                   VARCHAR(20),
            Password                    VARCHAR(10)   NOT NULL);
```

To create the table, copy the code above and **change the name of the schema to reflect the name you gave it**.

Within MySQL Workbench run the SQL. The code will be used to create the table. As can be seen, the Username and Password are non-NULL variables, but FirstName can be left blank and NULL will be assigned instead of a first name. Note that the Username and FirstName are different, a Username is required, but a FirstName is not.

Use MySQL Workbench as a sandbox to practice writing your SQL statements that you will use in your python code. Keep in mind that since the Username is a Primary Key it must be a unique username that does not exist within the table. For example, if you were to use MySQL workbench to put the Username kerenstimpy then I cannot insert that username in the python script because it would already exist in the table.

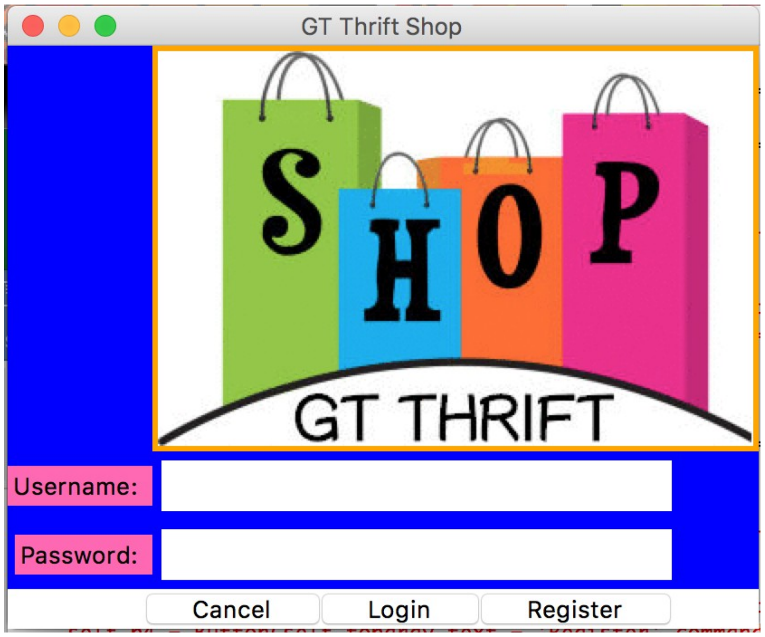
Multiple GUI windows

Multiple GUI windows may not seem difficult, but it takes a certain method to show new windows while hiding old ones. It will involve the creation and button execution of several helper methods to transition between windows. You may need to hide one of the windows you create until it is needed. There will be several helper methods for transitioning. It's up to you as to how many transition methods you will need, but basically one per transition should be enough. In these transition methods, you will .withdraw() a certain TopLevel window instance created for a certain page, while calling a method that deiconifies a different Toplevel instance for a different page. This means that you will be using one main Tk() instance for your main window and different Toplevel window instances for your other window(s).

A note about managing your database connection: You will need to use a connection to the database in several different parts of your code. You can do this in one of two ways: a) You can connect once (for example, in your __init__ method) and save the database connection object as an object variable, and then close the connection after the user successfully logs in. or b) You can connect to the database every time you need to do something with it, and then close your database connection before the method you are in exits. Which you use is up to you, both have advantages and disadvantages, you should think about the pros and cons and be able to explain why you chose to do things the way you did.

Method Name: **__init__**

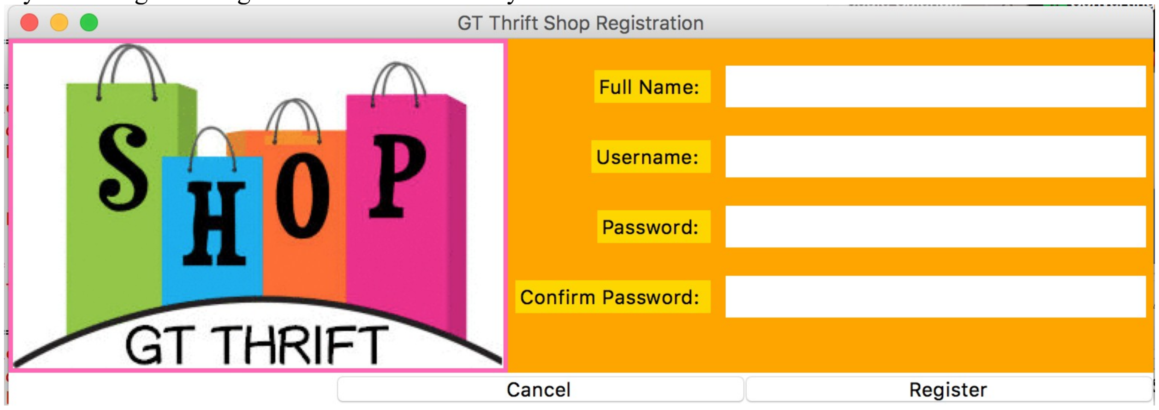
This method may create both your Login Widow GUI and Register window GUIs, but it should not show them both, you'll have to hide the Register window until it is needed. You will need two windows, the Login window and the Register window. One of the two windows will need to be a Toplevel window, while the other window can be created in the main window instance returned by your call to Tk(). You may generate these windows however you wish, but it is *highly* suggested that you use the grid layout manager. Here is a picture of the Login Wndow GUI. Note the following details:



- The top image can be any logo that is GT Related. You can find an image anywhere online that allows webscraping and use the image url for urllib.request.urlretrieve(). Alternatively, you may use the image provided or create your own and upload it from your own computer. Be sure to upload that to t-square when you submit your code. You should use a small .gif image and the method(s) described here: http://effbot.org/tkinterbook/photoimage.htm http://www.summet.com/dmsi/html/guiProgramming.html#adding-images-to-your-gui
- The Title is 'GT Thrift Shop'.
- The labels are sticky to the east and the entry boxes are width 30 and state normal
- Clicking on the "Register" button should call the Register method.
- Clicking on the "Login" should call the LoginCheck method.
- *Feel free to use any color scheme for the background (as long as it's not white).* The different background colors were created by making different frames.

The Register GUI window 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:



- The top image is the same as the one in the login page.
- The title is ‘GT Thrift Shop Registration’
- Each entry is width 30 and state normal.
- The labels are all sticky to the east with a pady and padx of 5.
- Clicking on the “Cancel” button calls a helper function to hide this GUI window and redisplay the Login Window.
- Clicking on the “Register” button calls the RegisterNew method.
- *Feel free to use any color scheme for the background (as long as it’s not white).* The different background colors were created by making different frames.

Method Name**: LoginPage**

This method is called to show your Login window. It is typically called by the RegisterNew method, but depending upon how you write your __init__ method, it may also be called by the __init__ method to make sure the login window is showing at the beginning of the program. You will have already created the login page window and filled it with widgets in your __init__method, so all this method needs to do is to make sure it is visible (deiconify it). (It may also need to hide the Register window, unless one of your other methods takes care of that.)

Method Name: **Register**

Parameters:
    None
Return Value:
    None

This method will simply show the Register window, much as the LoginPage method shows the Login window. (It may also need to hide the Login Window before it shows the register window unless one of your other methods does that.)

Method Name: **RegisterNew**

Parameters:
    None
Return Value:
    None

Description:
This method will take the entries from the register page and put them into the database. You must check that both passwords entered are the same and that the username and password entries are not left blank. Use message boxes to describe any problems with the way the register page was filled out. Also check to see if the username is already used within the database. If so, use a message box to express this. If everything is ok, you can insert into the database. (Hint: The First Name entry can be left blank and in that case would not be inserted into the database.) Be sure to .commit() the database to make sure your changes are saved! Once registered, a message box should tell the user that they are now registered and you should hide the register window and then call the LoginPage method to re-show the login window.

Method Name: **LoginCheck**

Parameters:
    None
Return Value:
    None

Description:

This method will check to see if what was typed into the login page matches any users currently in the database. You will need to create a cursor from the database connection, and check the username and password that was entered in the GUI to see if they match an entry in the database. You may also want to get the first name of the user to use on the 2nd part of his homework later (hint hint). If you find no matches, you should show a message box explaining that the user entered an unrecognizable username/password combination. If you do find a match, you should remember the username for use later in part 2 of the homework, and then display a message box saying that the user has logged in successfully. (At this point you can close your program and all the windows, as the user is finished with part A of the homework.)

Grading:

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

Login GUI Window		**20**
GUI has image at top	6	
GUI has all components with proper characteristics	4	
Login works as described	4	
Register works as described	4	
GUI uses message boxes for errors	2	
Register GUI Window		**20**
GUI has image at top	6	
GUI has all components with proper characteristics	4	
Register works as described	4	
Cancel brings user back to login page	4	
GUI uses message boxes for errors	2	
LoginCheck()		**25**
Uses cursor/db properly	5	
Correctly checks to find matching user	10	
Uses message boxes for errors with user match	5	
Closes GUI with match	5	
RegisterNew()		**25**
Uses cursor/db properly	5	
Correctly puts user in database	10	
Uses message boxes for errors with user entries	4	
Checks if the username is already taken	4	
Commits db when done	2	
Database Code In General		**10**
Correctly closes all cursors and database connections	10	