CS 2316

# Individual Homework 9b – GTMarketPlace part B

**Due: Thursday April 9th, before 11:55 PM**
**Out of 100 points**

---

**Files to submit:** **1. hw9b.py**

## This is an INDIVIDUAL assignment!

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.

---

# Background:

It's nearing the end of the semester and all of the 1301 students need a way to sell their Scribblers off. You'll be designing a simple marketplace app where users may buy or sell items from each other. All transactions will be made with in-app money which can be cashed out or deposited at any point. As a special promotion, every registered user this year will start with $50 in their account. This will be part B of this assignment, in which you will be asked to implement the market place logic building on top of the code you wrote in part A. You are welcome to change parts of your submission for part A if there were any issues.

## MarketPlaceUsers table:

```
CREATE TABLE MarketPlaceUsers
(
    Fullname        VARCHAR(255)
    ,Username       VARCHAR(255) NOT NULL UNIQUE
    ,Password       VARCHAR(255) NOT NULL
    ,Balance        DECIMAL(6,2) NOT NULL
)
```

## MarketPlaceListings table:

```
CREATE TABLE MarketPlaceListings
(
    listing_id      INT NOT NULL UNIQUE AUTO_INCREMENT
    ,ListingUser    VARCHAR(255) NOT NULL
    ,Itemname       VARCHAR(255) NOT NULL
    ,Price          DECIMAL(6,2) NOT NULL
    ,Sold           BOOLEAN NOT NULL DEFAULT 0
    ,BuyingUser     VARCHAR(255)
)
```

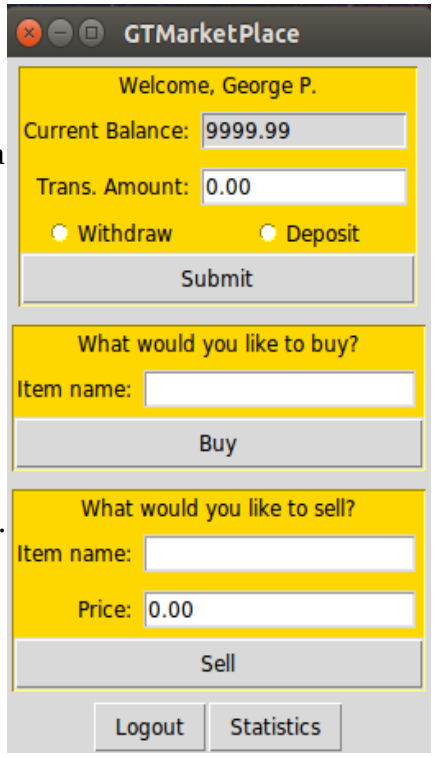
Using pymysql, you will be querying these tables in the *cs2316db* database on host *academic-mysql.cc.gatech.edu* with the username and password given to you. Be sure to use your own username and password.

**Note:** When querying *MarketPlaceListings*, you should not touch the listing_id column when inserting or updating rows. This means in all your queries, you will need to specify which columns you want to get rather than using the * syntax. The listing_id is a field which will be autopopulated by the database whenever a row is inserted into this table. *Sold* is a boolean column which will take on values of either 0 or 1 and is self-explanatory. Rows in this table will be explained in more detail later in the document.

# GUI:

Your GUI up to this point has a login window and a register window. Now we need another “marketplace” window. On successful login, after the “Success!” message box appears, the user should be taken to GTMarketPlace page that you will create which looks as follows:

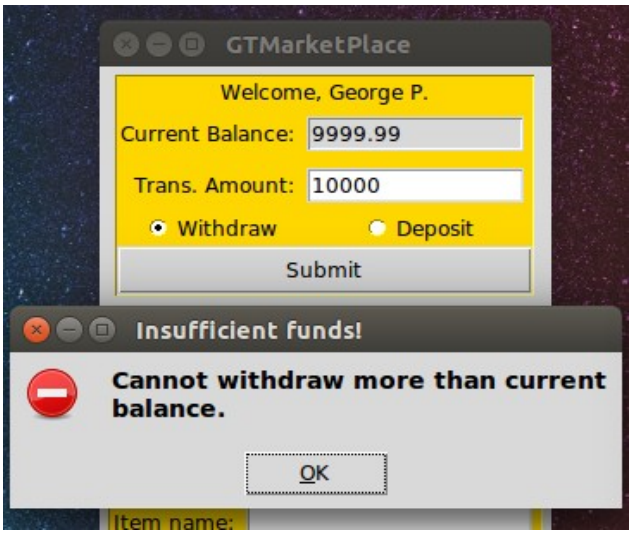
**Traits:**

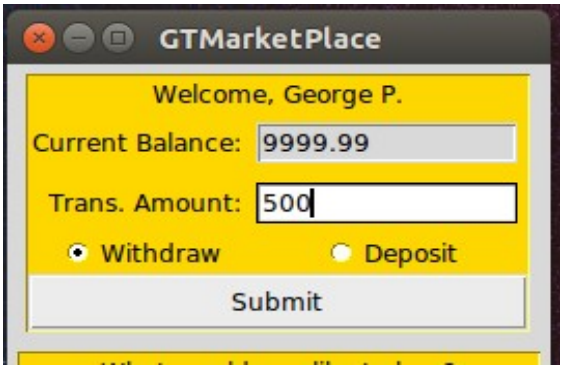


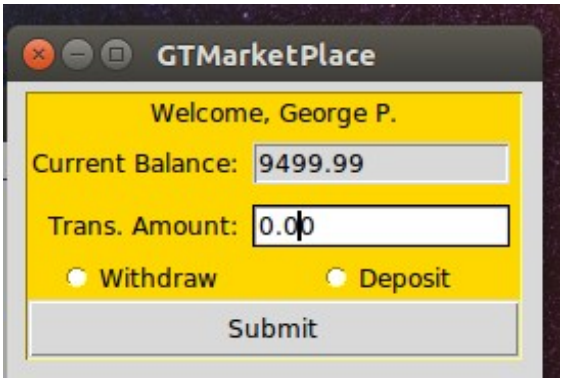

- title: GTMarketPlace
- frames are sunken and have a gold background
- First frame label should be “Welcome, <fullname>” if the user has a non-null full name, else “Welcome!”
- Entries in the first frame have widths of 15
- Current balance Entry should be read-only and contain the user's current balance.
- Trans. Amount and Price Entries should start with “0.00”

**Button behaviors:**

Clicking the “Submit” button will attempt to withdraw the entered Trans. Amount from the user's balance in the MarketPlaceUsers table. If the user is attempting to withdraw more money than he has, an error dialog should be raised instead. The max balance is $9999.99, but you don't have to do any checking for deposit. Just let it hit the max on deposit. After successful withdrawal or deposit, the fields in the frame should be updated/reset.



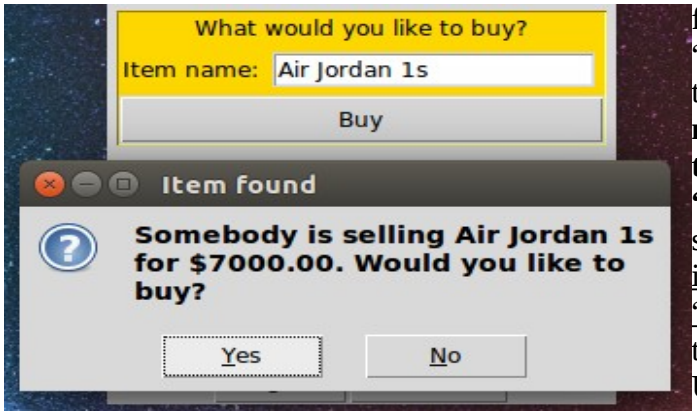

Clicking the “Buy” button will attempt to buy an item in the MarketPlaceListings table satisfying the following conditions:

1. It has not been sold (Sold = 0)
2. It was not listed by the current user
3. The user has enough money to buy the item.

If there are no items in the MarketPlaceListings table with the entered name (listed by somebody other than the current use), display an error dialog informing the user that there is no such item being sold in the market place by other users at this time.

If there is such an item in the marketplace being sold by another user, but the current user doesn't have enough money to buy it, display an error dialog informing the user that he does not have sufficient funds to buy any such items in the market place.

If there are multiple items with the same name, the item with the lowest price should be bought first. If the items are the same price still, the oldest one should be bought (lowest listing_id). **(hint: the ORDER BY statement may prove useful here)** If there is a valid item to be bought, pop up the following confirmation dialog. If the user clicks "yes", you will proceed with buying the item. If the user clicks "no", do not. **(hint: the messagebox.askyesno() dialog returns True if the user clicks "yes" and False if the user clicks "no")**. The message should tell the user who is selling the item and for what price (use Fullname if it is not NULL. If it is NULL, use "Somebody"). Be mindful of the SQL statements that you use in this function. You will be using the UPDATE statement to change the Balance field of the selling user and the buying user as well as the Sold and BuyingUser fields in the MarketPlaceListings table.



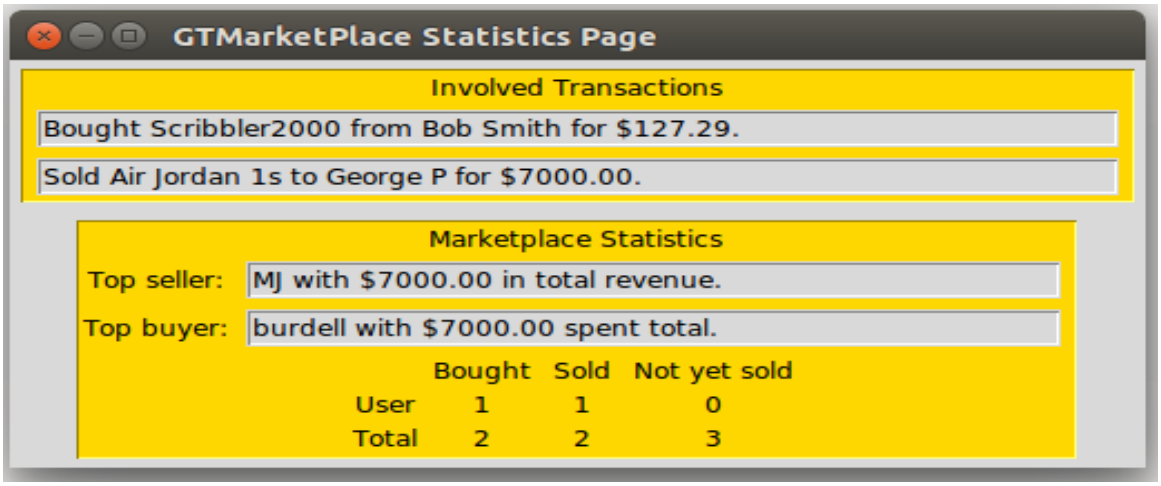

**Note:** As you may have noticed, we can't uniquely identify a row in the MarketPlaceListings table by just the Itemname, or even the Itemname in conjunction with the ListingUser (in the event a user is selling two of the same item). For this reason, you will want to keep track of the listing_id of the item you wish to buy. The listing_id provides a unique identifier that we can use in the UPDATE statement in case the user decides to buy the item (WHERE listing_id = …).

Something else to keep in mind is that ListingUser provides the username of the user who has listed a certain item, but usually we want to use full names in user-facing dialogues like the one below. Luckily, usernames are unique in the MarketPlaceUsers table. You can use a username to get the fullname with a SELECT … WHERE query on the users table.

Clicking the "Sell" button will add a new entry into the MarketPlaceListings table using the current user's username as the ListingUser. Make sure here that you only specify values for ListingUser, Itemname, and Price columns and leave the other columns default. Upon successful insertion, display a success info dialog to the user and reset the Item name and Price fields in the frame.

Clicking the "Logout" button should return to the Login page.

**BONUS:** Clicking the "Statistics" button will display a Toplevel window with statistics about the Marketplace. Implementing this part of this assignment is not required and will act as a bonus. If you choose not to implement statistics, your GUI should only have a "Logout" button. Be as creative as you want with your statistics window. An example statistics page has been given here:




# Suggested Outline:

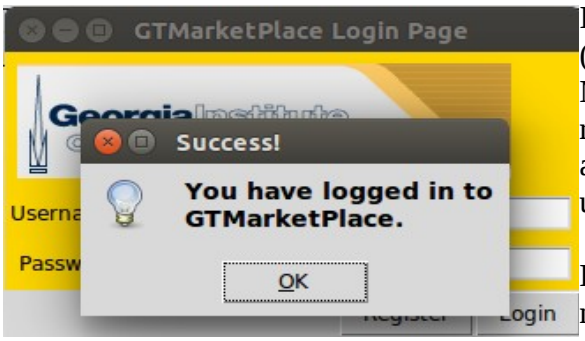
You may use any combination of class/functions/methods you wish to use to do this homework. The following is a *suggested* outline for your program: Unmodified methods from part A not included.

## __init__(slightly modified):

Set up your windows. Use the root window as your login window and create a new Toplevel for your register window and MarketPlace windows. We suggest you make helper methods to initialize your windows rather than creating a super long *__init__* method. You will then withdraw your other windows so that only the login window is shown.

## loginCheck (modified):

Query the database to check if the entered username and password represent a user in the MarketPlaceUsers table. If there is no match, display an error dialog (*messagebox.showerror()*) informing the user that this is an invalid username/password combination.




If there is a match, display an info dialog (*messagebox.showinfo()*) and switch windows to your MarketPlace window. (**hint:** *messagebox.showinfo()* will return after the dialog has been closed by the user). You will also want to store the username in an instance variable for use in later SQL queries.

Be sure to close the connection object here and in any other methods you use one in.

## updateMainPage:

Helper function that will be used to update various fields in your MarketPlace page. You will need to query the database to update the fields. Main two dynamic fields are the “Welcome...” label in the first frame which depends on the full name of the current user and the Current balance field which is self-explanatory. You will want to call this method after first switching to the MarketPlace page (from loginCheck), and after modifying anything in the database (withdrawing, depositing, buying).

## cashier:
This method will be attached to the “Submit” button. It will check which radio buton has been clicked and call the corresponding method. It will then call updateMainPage and reset the value of Trans. Amount to “0.00”.

## withdraw:
Make sure the Trans. Amount is not greater than user's Balance. Then UPDATE MarketPlaceUsers subtracting Trans. Amount from current balance.

## deposit:
UPDATE MarketPlaceUsers adding Trans. Amount to current balance.

## sell:
INSERT INTO MarketPlaceListings. Make sure to leave listing_id, Sold, and BuyingUser fields default (only specify ListingUser, Itemname, and Price fields in your query). Display a success dialog to the user informing him that the item has been added to the marketplace.

## buy:
First, check if there are any listings with the entered name that has not yet been sold and has not been listed by the current user. If there aren't any, display an error dialog informing the user. Next, if there are some listings with the entered name, check to see if any of them can be bought by the user. If they are all too expensive, inform the user that he does not have sufficient funds. If the user does have sufficient funds, buy the cheapest item and oldest (lowest listing_id) if multiple items are the same price. Keep the listing_id of the chosen item. Find the fullname of the ListingUser from the users table. Display the previously specified confirmation dialog to the user. If the user agrees to buy, determine the new balances of the two users involved in the transaction (buying and selling), and UPDATE the users table accordingly. Next, use the listing_id you kept before to UPDATE the listings table and mark this listing as sold (SET Sold=1). Be sure to commit and close connection object and then call updateMainPage.

## Statistics (bonus):
Creates a new top level window with marketplace statistics inside.

# Grading:

You will earn points as follows for each piece of functionality in your code:

| | | |
|---|---|---|
| **GUI:** | | **50** |
| - Part A components work correctly | 15 | |
| - Shows all warning and info dialogs correctly | 15 | |
| - Updates and resets fields in the GUI correctly | 10 | |
| - Logout works correctly | 10 | |
| **SQL:** | | **50** |
| - withdraws and deposits properly (checks balance as well) | 10 | |
| - sells properly | 10 | |
| - buy updates users' balances and listings table correctly | 15 | |
| - buy logic (confirmation dialog) correct | 15 | |
| - Doesn't close/commit connection properly | -10 | |
| **BONUS Statistics:** | | **+10** |
| **Total possible:** | | **100** |
| | | **+10** |