MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG445 Software Development with Scripting Languages

Assignment 2: Clothing Brand Application

This assignment aims to help you practice multithreading and concurrent programming, network programming, and graphical user interface components in Python. On the successful completion of this assignment, you will also practice Python basics. Your main task in this assignment is to develop a client-server application for a clothing brand based on Transmission Control Protocol (TCP).

The client application will be used by both store owners and data analysts, and multiple store panels and analyst panels should be able to be opened at the same time, so the server should be able to communicate with multiple clients at the same time as illustrated in Figure 1.

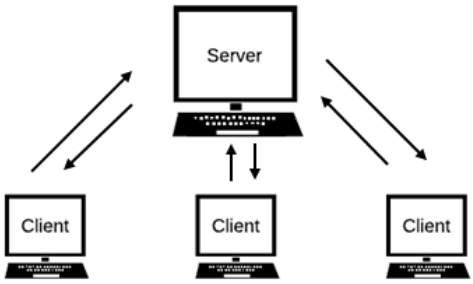

Figure 1: Client-Server Model

Overview:

In this assignment, you will develop a client-server application for a clothing brand. The client application will be used by the various stores of the brand. The customers can visit a store to look at item samples and try them on. If the customer would like to purchase items, the store owner then places an order from the central warehouse to be delivered to the customer. A store owner will be responsible for reporting the purchases and returns of the store, and sending them to the server, and the analysis team at headquarters will be responsible for requesting some statistics related to the operations, which will be generated on the server.

Clothing items on sale in the store are:

1. Basic T-shirt
2. Leather Jacket
3. Robe of the weave
4. Plaid shirt
5. "D4C" Graphic T-shirt
6. Denim jeans
7. Hodd-Toward designer shorts

Customers may also return items according to the store policy. By using the client application, the analysis team can request the following statistics related to the operations. Please note that both store owners and the analysts should login to the client application to perform their operations.

Table 1. Statistics Table

Code	Statistics
1	What is the most bought item overall? If there are multiple, all of them should be reported. You do not need to consider return operations.
2	Which store has the highest number of operations? Both selling and returning operations should be counted. If there are multiple top stores, every one of them should be reported. Please note that an operation can be a purchase operation, or a return operation and it may include multiple items.
3	What is the total generated income by the store? Returned item values should be subtracted from the total purchase value to calculate the income.
4	What is the most returned color for the Basic T-shirt? If the shirt has not been purchased, then "No sales" should be displayed.

The server should manage three text files: **users.txt**, **items.txt** and **operations.txt.** The client should not have access to these files directly.

1. The **users.txt** file will keep track of the details of the stores and analysts as follows (username;password;role):
   ```
   dereboyucd1;12a1;store
   magusa3;1ee7;store
   gregoryhouse;2b9c;analyst
   ```
   Please note that there will be one account for each store, which will be managed by the store owner.

2. The items.txt file will keep track of the clothing items and their details as follows (itemID;itemName;color;price;stockAvailable):
   ```
   1;Basic T-shirt;red;10;7
   1;Basic T-shirt;black;12;2
   2;Leather Jacket;red;20;3
   2;Leather Jacket;black;18;1
   3;Robe of the weave;red;17;3
   3;Robe of the weave;black;18;3
   4;Plaid shirt;red;11;4
   4;Plaid shirt;black;11;4
   5;D4C Graphic T-shirt;red;21;2
   5;D4C Graphic T-shirt;black;22;2
   6;Denim jeans;red;13;8
   6;Denim jeans;black;12;2
   7;Hodd-Toward designer shorts;red;14;0
   7;Hodd-Toward designer shorts;black;14;0
   ```

   To keep operational costs low, the brand has decided to only offer two color options for their customers, red and black. **When you implement the server, you need to consider that the file content can be changed.**

3. The operations.txt file will keep track of purchase and return operations as follows (purchase;store;customerName;quantity-itemID-color,quantity-itemID-color… or return;store;customerName;quantity-itemID-color;quantity-itemID-color…):
   ```
   purchase;dereboyucd1;cem;1-1-red;1-6-black
   purchase;magusa3;balduran;3-2-black
   purchase;magusa3;vyse;1-4-red
   return;magusa3;vyse;1-4-red
   ```

   As you can see from the example above, `cem` bought a red basic t-shirt and a pair of black denim jeans at the `dereboyucd1` store. `balduran` bought 3 black leather jackets at the `magusa3` store, then `vyse` bought and returned a red plaid shirt at the `magusa3` store. Please note that customer names and shipping addresses are managed by an external system, and therefore shipping addresses are not stored in this system. Please also note that customer name is assumed to be unique. A new operation should always be added to the end of the file.

Requirements

- **Authentication:** When the client application is started and the connection is established with the server, the server will send a confirmation message to the client (message: **`connectionsuccess`**). Once this message is received, the client will then show the following login screen to take the username and password from the user (See Figure 2).

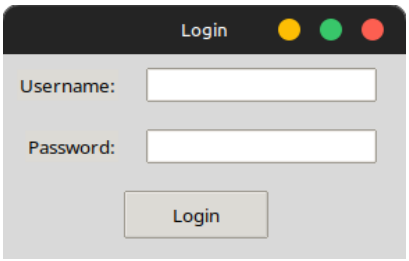

Figure 2: Login Window

Once the client sends the username and password to the server (message format: **`login`**`;username;password`), the server will check the **users.txt** file and send an approval message (message format: **`loginsuccess`**`;username;role`) or rejection message (message format: **`loginfailure`**) back to the client. If the login is not successful, then the error message box will be shown on the client side. However, if the login is successful, then the appropriate user panel will be shown based on the user role.

- **Store Panel:** Figure 3 shows the user panel for a store. **When you implement the client, you can assume that only the following products with two given colors are available.** When the owner wants to add an operation, they need to select the items, enter the quantity and color of the items purchased or returned. When the "**Purchase**" button is clicked, the details of the operation will be sent to the server in the following format: (**`purchase`**`;store;totalquantity;quantity-itemID-color,quantity-itemID-color...;customername)`.

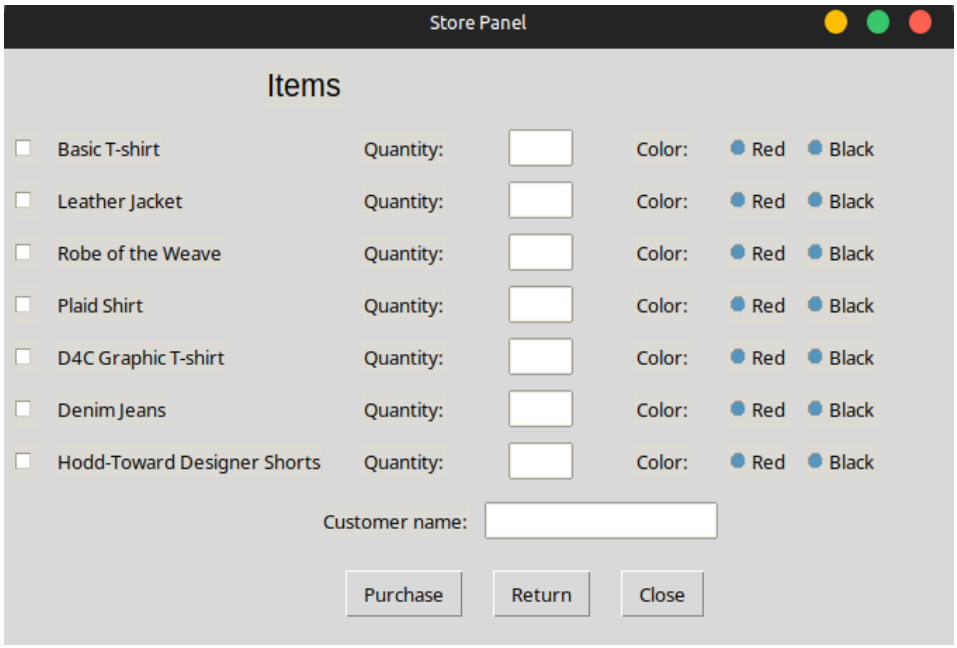

Figure 3: Store Panel

Upon purchase operation, the server should first check the availability of the item(s). In case the owners tries to enter a purchase, but one or more of the items are not available (i.e., stockAvailable in items.txt is 0 (zero) for that color), then the error message (**availabilityerror;**item1;item2;... where item represents the name and colour of the item which is not available with the requested quantity) should be sent back to the client application and the error message should be displayed in the message box, containing the names of the items along with their colors. Therefore, the operation will be cancelled, and no operation will be added to the **operations.txt** file.

If all the above-mentioned criteria are held, then stockAvailable field for each of the purchased items should be updated in the **items.txt** file, and an operation record should be added to the **operations.txt** file and the confirmation message (**purchasesuccess;totalordercost**) should be sent back to the client application and the success message, along with the total cost of the order should be displayed on the client side.

When the "**Return**" button is pressed, the details of the operation will be sent to the client in the following format: (**return;**`store;totalquantity;quantity-itemID-color;quantity-itemID-color...;customername`). Once the return operation message is sent to the server, the server should check if the items have been purchased by the customer before. In the case they have not been purchased by that customer or already returned, the error message (**returnerror**) should be sent to the client side and the appropriate error message should be displayed in the message box (e.g. unsuccessful operation – please re-check the items). If the return operation can be held successfully, then stockAvailable field for each of the purchased items should be updated in the **items.txt** file, and an operation record should be added to the **operations.txt** file and a confirmation message should be sent to client side in the following format (**returnsuccess**) and displayed once again on the client side.

For both successful purchase and return operations, operations should be stored in the operations.txt, for statistics generation and purchase/return validation.

When the "Close" button is clicked, the connection between the client and the server will be terminated, and the window will be destroyed.

- **Analyst Panel:** Figure 4 shows the user panel for an analyst. When the analyst selects one of the available statistics reports, the report code will be sent to the server such as **`report2`**. The server will then generate and send back (message format: **`report2;`***`answer`* *or* **`report2;`***`answer1;answer2;...`* if there are multiple answers). The answer should be shown in a message box appropriately on the analyst side.

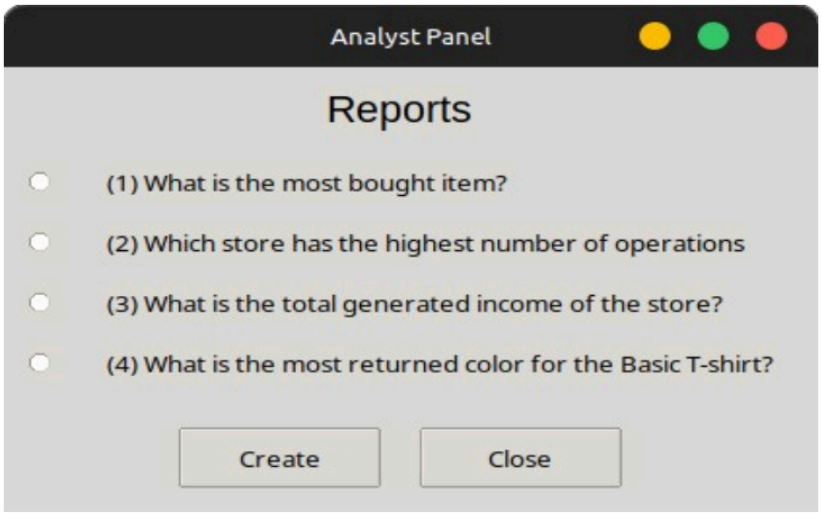

Figure 4: Analyst Panel

When the "Close" button is clicked, the connection between the client and the server will be terminated, and the window will be destroyed.

Thread Synchronization: You should consider the RLock thread synchronization technique to deal with any problems caused by attempting to access the shared data at the same file.

Rules

- You need to write your program by using **Python 3.x**.
- You can **only** use all built-in functions and modules.
- You also need to create a file called ReadMe.txt which contains the following items. Please note that **if you do not submit ReadMe.txt, your submission will not be evaluated.**
    - Team members
    - Which version of Python 3.x you have used
    - Which operating system you have used
    - How you have worked as a team, especially how you have divided the tasks among the team members (who was responsible for what?), how you have communicated, how you have tested the program, etc.
- You should name your server as **server.py** and name your client as **client.py**.
- You should not forget to submit your **users.txt**, **prices.txt**, and **operations.txt** files.
- You need to put all your files into a folder that is named with your student id(s) and submit the compressed version of the folder in the **.zip** format.
- **Only one team member** should submit the assignment.
- **Code quality, modularity, efficiency, maintainability, and appropriate comments** will be part of the grading.

Grading Policy

The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Server Side: Login	5
Server Side: Store Panel - Purchase	20
Server Side: Store Panel - Return	20
Server Side: Analyst Panel - Report 1	5
Server Side: Analyst Panel - Report 2	5
Server Side: Analyst Panel - Report 3	5
Server Side: Analyst Panel - Report 4	8
Server Side: Thread Synchronization	5
Client Side: Login Window and transfer to an appropriate panel	5
Client Side: Store Panel	15
Client Side: Analyst Panel	7