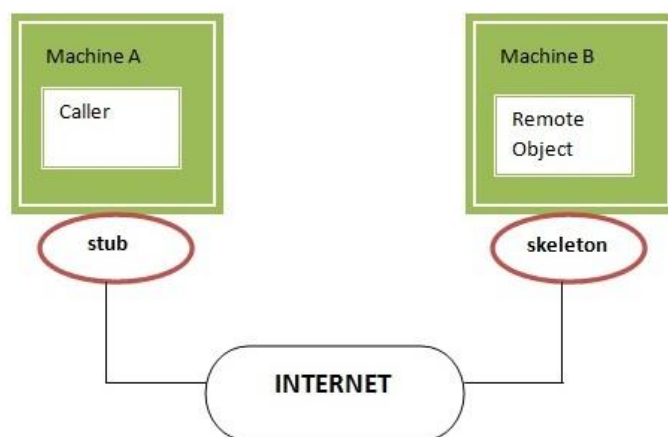


Exercise #1: RMI Implementation in python

Distributed Systems, winter 2021 (available on github.com/Mohammadmahdi-Mohammadi)

درواقع RMI مخفف عبارت **Remote Methode Invocation** به معنی «فراخوانی متد از راه دور» است. همانطور که از نام این اصطلاح مشخص می شود، RMI مکانیسمی در اختیار برنامه نویسان قرار می دهد که می توانند از طریق آن متد اشیای گوناگون را بر روی یک ماشین مجازی راه دور اجرا کنند. مکانیسم های فراخوانی راه دور متفاوتی در دنیای نرم افزار ایجاد شده اند، اما RMI برخلاف بسیاری از آنها محدود به انواع داده ای اولیه یا ساختار هایی متشکل از داده های ساده نیست و به کمک آن می توان اشیای نرم افزاری را به تمامی همانند یک پارامتر عبور داد یا باز گرداند. این ویژگی از RMI یک مکانیسم منحصر به فرد می سازد. چنین خاصیتی به این معنی است که یک برنامه نویس می تواند به کمک RMI، کدی را در شبکه انتقال دهد و در ماشین های مجازی راه دور به صورت دینامیک آن ها را اجرا نماید. بدین ترتیب برنامه نویسان در زمان برنامه نویسی سیستم های گسترده، آزادی عمل زیادی به دست خواهند آورد.¹



این پروژه با فرض اجرای آن در یک محیط محلی تحت عنوان مدیریت کاربران و کتاب های کتابخانه صورت پذیرفته است. به دلیل عدم اهمیت رجیستر کردن کاربران و همچنین عدم تمرکز داکيومنت پروژه روی این مساله، صرفا برای بحث احراز هویت، تعدادی کاربر به صورت استاتیک درون قسمت سرور در نظر گرفته شدن و صرفا با درخواست ارتباط هر کلاینت به سرور، نام کاربری و پسورد با اطلاعات سرور تطبیق داده شده و در صورت تطابق، اجازه دسترسی و استفاده به کلاینت تحت عنوان همان کاربر داده میشود و بدین ترتیب به هیچ عنوان امکان تغییر در لاگ سرور به صورت ناشناس امکان پذیر نیست. در کنار این شرایط، یک نام کاربری تحت عنوان ادمین نیز در نظر گرفته شده است که صرفا وی در صورت لاگین از یک کلاینت به عنوان ادمین شبکه، امکان اجرای مستقیم

¹ <http://www.thisblog.blogfa.com>

دستورات و کدهای سمت سرور را در سمت کاربر داشته که این کدها سمت سرور اجرا و نتایج آنها تاثیر مستقیم در ادامه عملکرد سرور و تغییرات احتمالی در دسترسی کاربران متصل با کلاینت های مختلف دارد.

در سیستم پیاده سازی شده تا حدودی سعی شده که سرور در صورت بروز هرگونه مشکل احتمالی که ممکن است که در سمت کلاینت اتفاق بیفتد، تعادل خود در شبکه را حفظ کند و بتواند در صورت لزوم ارتباط را به گونه ای مدیریت کند که این مشکل در یک کلاینت، توسط کلاینت های دیگر احساس نشود.

امکاناتی که برای کاربران عادی در نظر گرفته شده است شامل مشاهده کتاب های قابل دریافت از کتاب خانه در لحظه فعلی، امکان امانت گرفتن کتاب و برگشت دادن کتاب امانت گرفته شده میباشد. در صورتی که ادمین از سمت کلاینت به سیستم لاگین کند، میتواند با ارسال مستقیم کامند، تصمیمات مدیریتی برای سرور اخذ کند که آن دستورات به همان شکل در سمت سرور اجرا میشوند و همچنین ادمین میتواند شرایط فعلی سیستم در لحظه فعلی، شامل کاربران و کتاب های موجود و امانت گرفته شده را همگی به صورت آبجکت دریافت میکند و بستر لازم برای ساخت آبجکت های مناسب با استفاده از داده های دریافتی از سوی سرور، فراهم گردیده است.

شرح اجرای برنامه:

*در ابتدا لازم به ذکرست که در پیاده سازی درموارد محدودی، از توابع مشخصی استفاده شد که همگی برای تصویرسازی مناسبتر صورت گرفته اند و در کنار این توابع، منبع و نام نویسندۀ ذکر گردیده است.

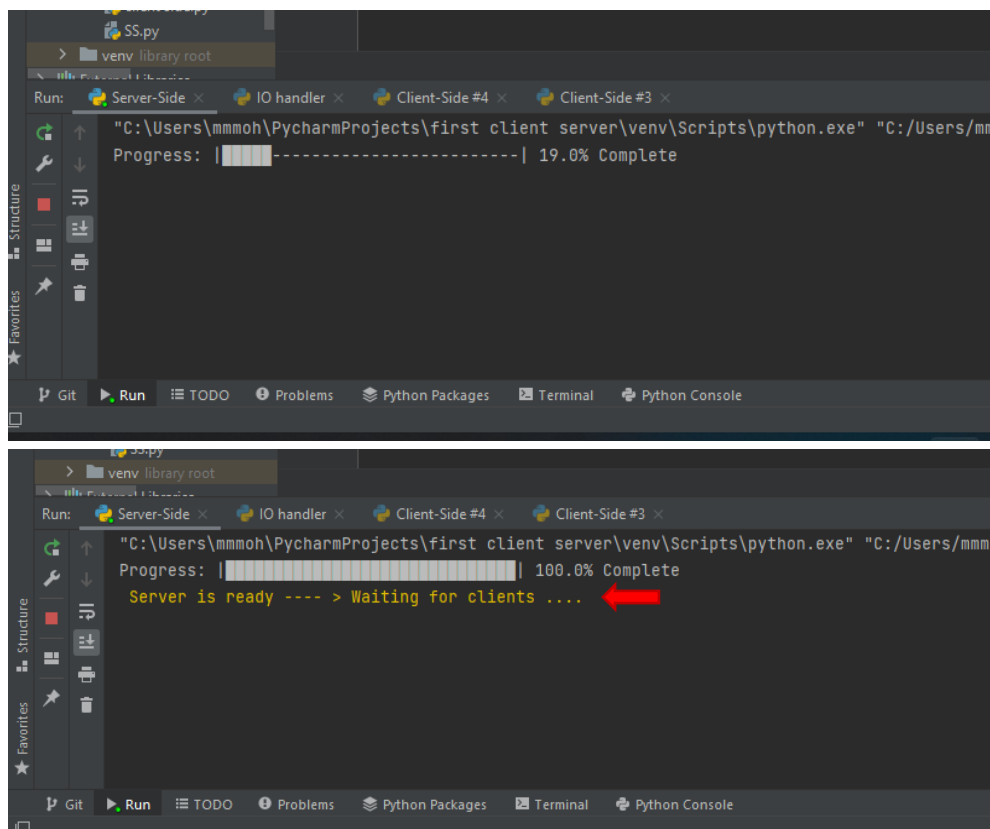
```
# -----
# function for login interface
# copyright:https://stackoverflow.com/questions/3173320/text-progress-bar-in-terminal-with-block-characters
# Print iterations progress
def printProgressBar(iteration, total, prefix = '', suffix = '', decimals = 1, length = 100, fill = '█', printEnd = "\r"):
    """
    Call in a loop to create terminal progress bar
    @params:
        iteration - Required : current iteration (Int)
        total      - Required : total iterations (Int)
        prefix     - Optional : prefix string (Str)
        suffix     - Optional : suffix string (Str)
        decimals   - Optional : positive number of decimals in percent complete (Int)
        length     - Optional : character length of bar (Int)
        fill       - Optional : bar fill character (Str)
        printEnd   - Optional : end character (e.g. "\r", "\r\n") (Str)
    """
    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filledLength = int(length * iteration // total)
    bar = fill * filledLength + '-' * (length - filledLength)
    print(f'\r{prefix} |{bar}| {percent}% {suffix}', end = printEnd)
    # Print New Line on Complete
    if iteration == total:
        print()
# -----
```

```

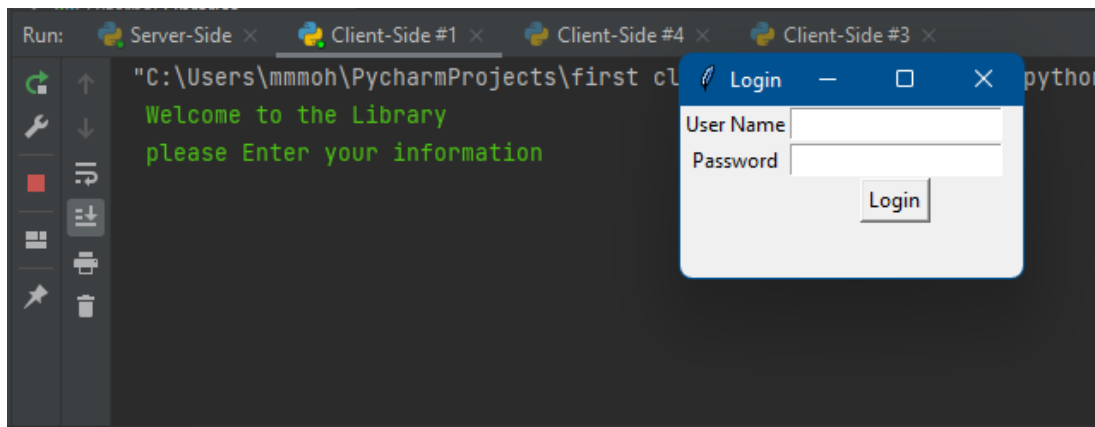
56
57 # copyright: https://stackoverflow.com/questions/3906232/python-get-the-print-output-in-an-exec-statement
58 # Since Python 3.4 there is a solution is the stdlib:
59
60 # -----
61
62 # uses in code migrations
63 import time
64 import sys
65 from io import StringIO
66 import contextlib
67
68 @contextlib.contextmanager
69 def stdoutIO(stdout=None):
70     old = sys.stdout
71     if stdout is None:
72         stdout = StringIO()
73     sys.stdout = stdout
74     yield stdout
75     sys.stdout = old

```

در مرحله اول قسمت آماده سازی سرور است که در فایل ارسالی، فایل پایتون با نام Server_side انجام این وظیفه را بر عهده دارد.



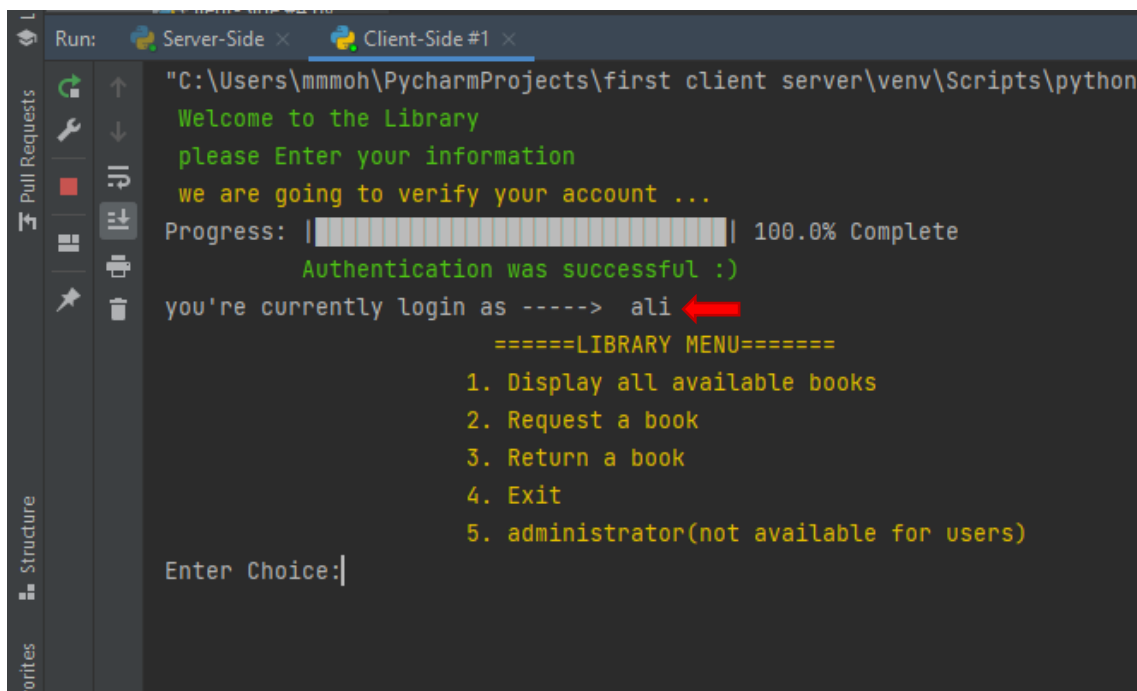
پس از آماده سازی سرور، یکی از نسخه کد های کلاینت که در فایل های ارسالی موجود است که باید اجرا شود تا فرایند اتصال به سرور توسط کلاینت صورت گیرد. نکته قابل توجه در فایل های ارسالی ، وجود یک کد کلاینت یکسان است که صرفا برای راحتی کار به هنگام اجرا، در ۵ فایل کلاینت مختلف قرار گرفته است.



در این مرحله و پس از اجرای کد کلاینت، یک احراز هویت صورت میگیرد که به صورت استاتیک در سمت سرور، ۴ کاربر با مشخصات زیر قرار گرفته اند. کاربر ادمین است که دارای دسترسی به اجرای مستقیم کد در سمت سرور و دریافت اطلاعات به صورت آبجکت در سمت سرور است.

Username	ali	amir	hamid	admin
password	1985	1998	2000	admin

**استفاده از حروف کوچک انگلیسی

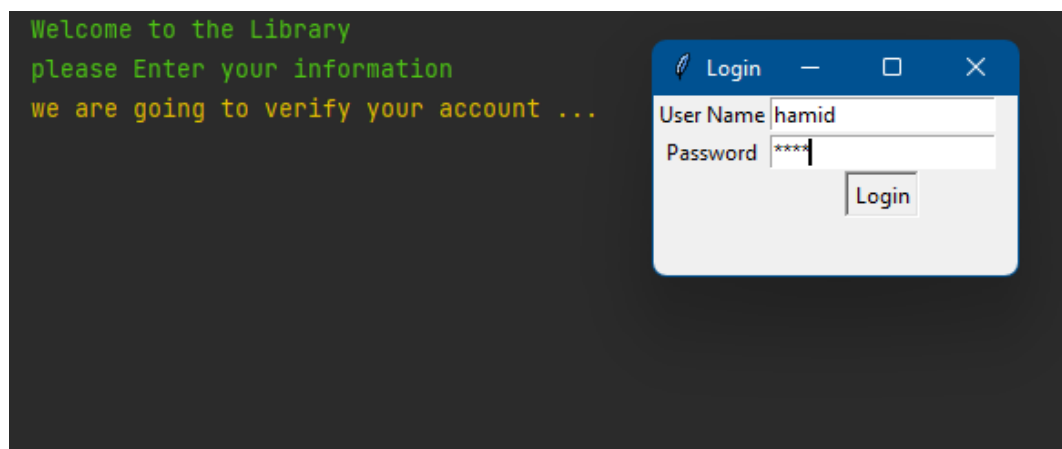


پس از احراز هویت، گزارش لحظه ای در سمت سرور قابل دسترس است و شرایط سرور و وضعیت کاربر لاگین شده و اطلاعاتش به نمایش گذاشته میشود:

```
"C:\Users\mmmoh\PycharmProjects\first client server\env\Scripts\python.exe" "C:/Users/mmmoh/Pycha
Progress: |████████████████████████████████████████████████████████████████████████████████| 100.0% Complete
Server is ready ---- > Waiting for clients ....

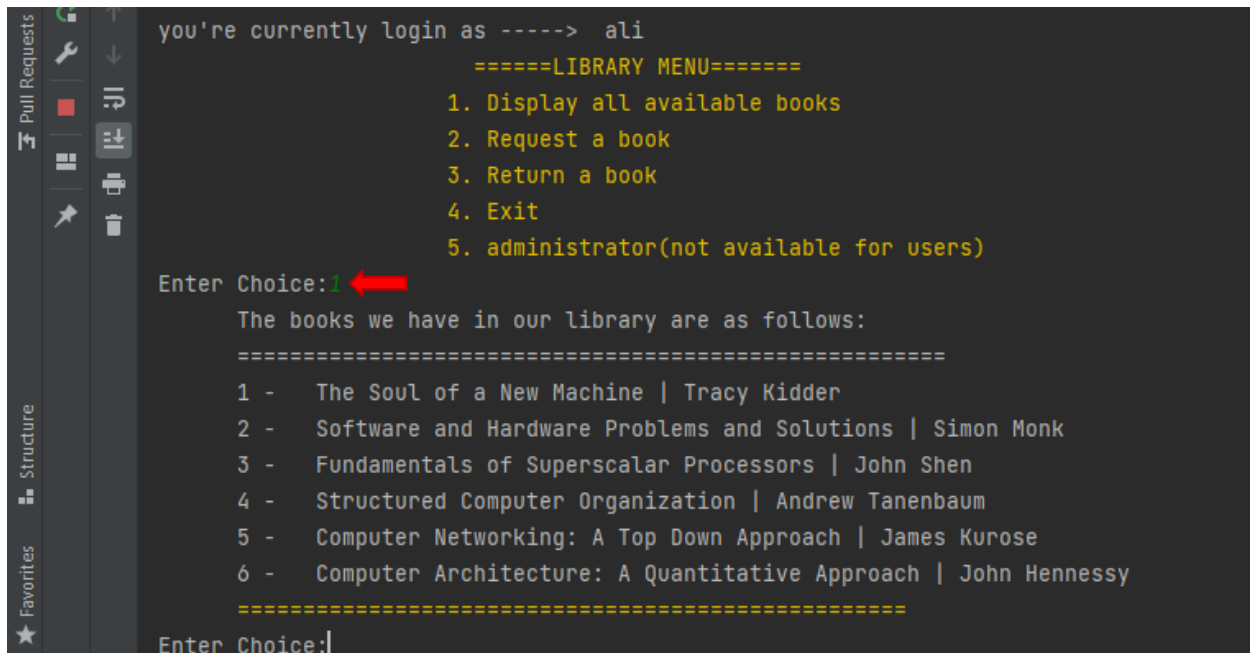
=====
Server status report:
A new client connected with the following information:
here is client: <socket.socket fd=260, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
Communication information: ('192.168.1.106', 3741)
connected to 192.168.1.106 3741
Number of accepted clients: 1
=====
authentication was successful :)
Current user info is: ('ali', '1985')
```

در هر مرحله اگر کلاینت دیگری به سرور لاگین کند، سرور همزمان با ارایه خدمت به کاربر قبلی، به کاربر جدید حتی اگر همان کاربر قبلی باشد و همزمان با کلاینت دیگری به سیستم لاگین کرده باشد خدمت رسانی میکند.



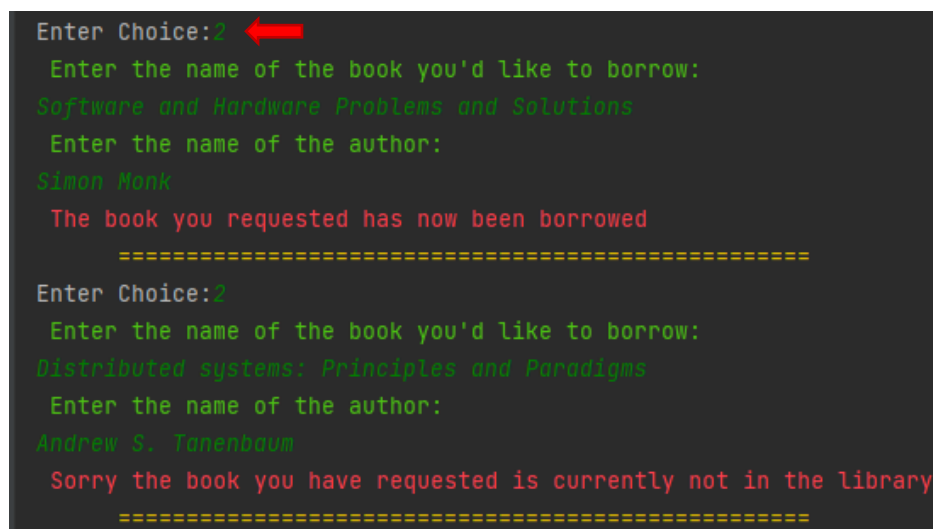
```
=====
Server status report:
A new client connected with the following information:
here is client: <socket.socket fd=388, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM,
Communication information: ('192.168.1.106', 6829)
connected to 192.168.1.106 6829
Number of accepted clients: 2
=====
authentication was successful :)
Current user info is: ('ali', '1985')
authentication was successful :)
Current user info is: ('hamid', '2000')
```

پس از فرایند احراز هویت، منوی کتابخانه شامل ۵ انتخاب برای کاربر نمایش داده شده و متناسب با پاسخ کاربر خدمات موردنظر وی را به صورت شفاف ارایه میکند. با انتخاب گزینه اول، تمام کتاب های موجود و قابل دریافت توسط کاربر فعلی برای وی به نمایش گذاشته میشود. در ابتدا برای تست اجرا از سیستم، ۶ کتاب در سمت سرور به صورت استاتیک قرارگرفته اند و کاربران مختلف برای استفاده از این کتب به سیستم مراجعه میکنند.



```
you're currently login as ----> ali
=====LIBRARY MENU=====
1. Display all available books
2. Request a book
3. Return a book
4. Exit
5. administrator(not available for users)
Enter Choice:1
The books we have in our library are as follows:
=====
1 - The Soul of a New Machine | Tracy Kidder
2 - Software and Hardware Problems and Solutions | Simon Monk
3 - Fundamentals of Superscalar Processors | John Shen
4 - Structured Computer Organization | Andrew Tanenbaum
5 - Computer Networking: A Top Down Approach | James Kurose
6 - Computer Architecture: A Quantitative Approach | John Hennessy
=====
Enter Choice:|
```

هرکاربر با انتخاب دومین خدمت، میتواند درخواست امانت گرفتن کتاب موردنظر خود را برای سرور ارسال کند که درصورت در دسترس بودن این کتاب، این درخواست با موفقیت ثبت خواهد شد.



```
Enter Choice:2
Enter the name of the book you'd like to borrow:
Software and Hardware Problems and Solutions
Enter the name of the author:
Simon Monk
The book you requested has now been borrowed
=====
Enter Choice:2
Enter the name of the book you'd like to borrow:
Distributed systems: Principles and Paradigms
Enter the name of the author:
Andrew S. Tanenbaum
Sorry the book you have requested is currently not in the library
=====
```

فرایند خدمت رسانی سرور به کلاینت ها به صورت پایدار و عدم سرایت مشکل در یکی از کلاینت ها به کل سیستم، با این ایده در سمت سرور پیاده سازی شده است که پس از درخواست هرکلاینت، سرور یک حالت چندنخی را از قبل پیاده سازی کرده باشد و با درخواست کلاینت، یک نخ وظیفه انجام خواسته های کلاینت را برعهده داشته باشد و بدین ترتیب خواسته مولتی کلاینت بودن پروژه فراهم گردیده است.

```
client_address = serversocket.accept()
prRed("\n=====")
prGreen("Server status report: ")
print("A new client connected with the following information:")
print("here is client: ", client)
print("Communication information: ", address)
print("connected to " + address[0] + " " + str(address[1]))

# -----
# _thread.start_new_thread(function, args[, kwargs])
# -----> Start a new thread and return its identifier. The thread executes the function
#         function with the argument list args (which must be a tuple). The optional kwargs
#         argument specifies a dictionary of keyword arguments.
#         When the function returns, the thread silently exits.

start_new_thread(client_thread, (client,)) ←
ThreadCount += 1
# -----
```

در واقع سرور زمانی که از طریق سوکت درخواست اتصالی را دریافت میکند، قبل از بررسی فرایند احراز هویت، یک نخ را ساخته و اطلاعات کلاینت و ارتباطش را در اختیار او قرار میدهد تا پس از احراز هویت به خواسته ای این کلاینت رسیدگی کند. فرایند اتصال و باندینگ کلاینت و سرور از طریق سوکت و به شکل زیر صورت گرفته اند:

```
# -----
import socket
from _thread import *
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

...

host = socket.gethostname()
port = 10000
ThreadCount = 0
# -----
```

سمت سرور

```
# -----
#Network config

clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

host = socket.gethostname()
port = 10000

# -----
```

سمت کلاینت

پروتکل ارتباطی کلاینت سرور ما از طریق ipv4 و براساس پروتکل TCP می باشد و به دلیل عدم مقدارهی مستقیم برای آی پی در قسمت هاست، به نظر نمی رسد که کد بر روی ماشین های مختلف مشکلی برای اجرا داشته باشد.

آبجکت های سمت سرور:

```
# -----
class Library:
    def __init__(self, listofbooks):...
    def displayAvailablebooks(self):...
    ...
    def currentbooks(self):...
    def addBook(self, student):...
    def Student_and_library_array_handler(self, index, student):...
    def lendBook(self, requestedBook, requestedBook_author, student_array):...

class Student:
    def __init__(self, name, password):...
    def get_value(self):...
    def get_value_to_migrate(self):...
    def get_array(self):...
    def get_array_value(self, index):...
    def del_b_book(self, index):...
    def requestBook(self):...
    def returnBook(self):...
# -----
```


آبجکت های سمت کلاینت برای امکان بازسازی آبجکت:

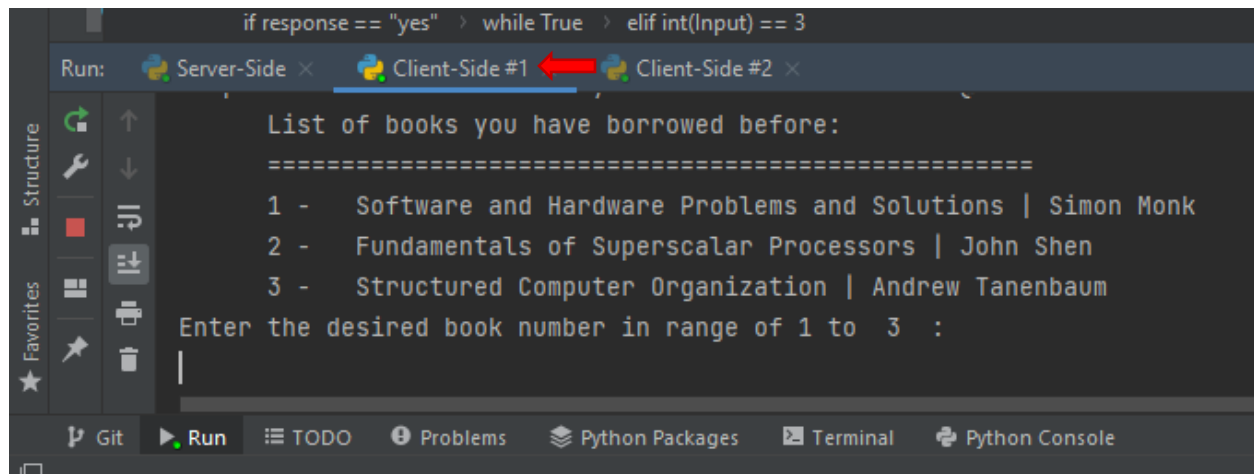
```
class Student:
    def __init__(self, name, password):...
    def get_value(self):...
```

** در صورت استفاده از سینتکس وراثت پایتون، باید فایل سرور در بالای فایل کلاینت import مییابد که در این صورت بحث thin client بودن تحت شعاع این دستور قرار میگیرد.

ادامه توضیحات اجرای کد:

قبل از این دیدیم که دوکلاینت با اطلاعات دو کاربر متفاوت به سیستم لاگین کردند و یکی از کاربران کتابی را برای امانت گرفتن به سیستم معرفی کرد و امکان ارایه کتاب وی تایید شد و از آن لحظه به بعد، امکان دریافت آن کتاب وجود ندارد و هرکلاینت متصل یا کلاینتی که در آینده به سیستم متصل شود، این کتاب را در لیست کتب قابل دریافت مشاهده نخواهد کرد.

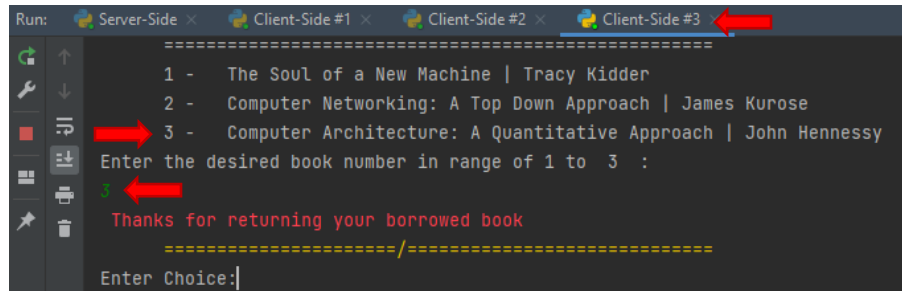
برای قسمت سوم منوی کتابخانه، حالتی را فرض کنید که کاربری از طریق #1 client-side به سرور متصل شده و تعداد سه کتاب را از کتاب خانه دریافت کرده است، حال در صورت انتخاب گزینه سوم از منوی کتابخانه تمام کتاب های دریافت شده از قبل تاکنون توسط این کاربر نمایش داده میشود:



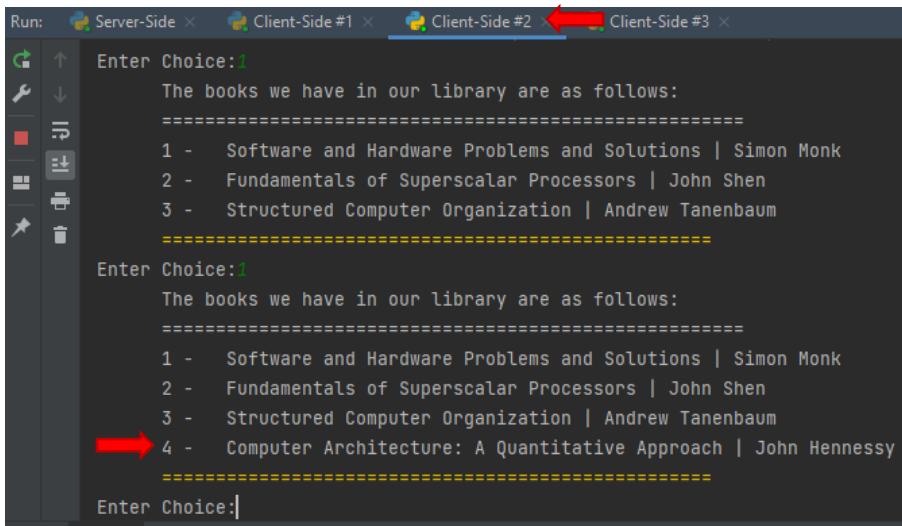
```
if response == "yes" > while True > elif int(Input) == 3
Run: Server-Side x Client-Side #1 Client-Side #2 x
List of books you have borrowed before:
=====
1 - Software and Hardware Problems and Solutions | Simon Monk
2 - Fundamentals of Superscalar Processors | John Shen
3 - Structured Computer Organization | Andrew Tanenbaum
Enter the desired book number in range of 1 to 3 :
|
```

حال زمانی که کاربر دیگر از طریق #2-client با سرور در تماس است در صورتی که درخواست کتب قابل دریافت را ارسال میکند، هیچ یک از کتب در اختیار کاربر دیگر را در لیست کتاب ها مشاهده نمیکند.

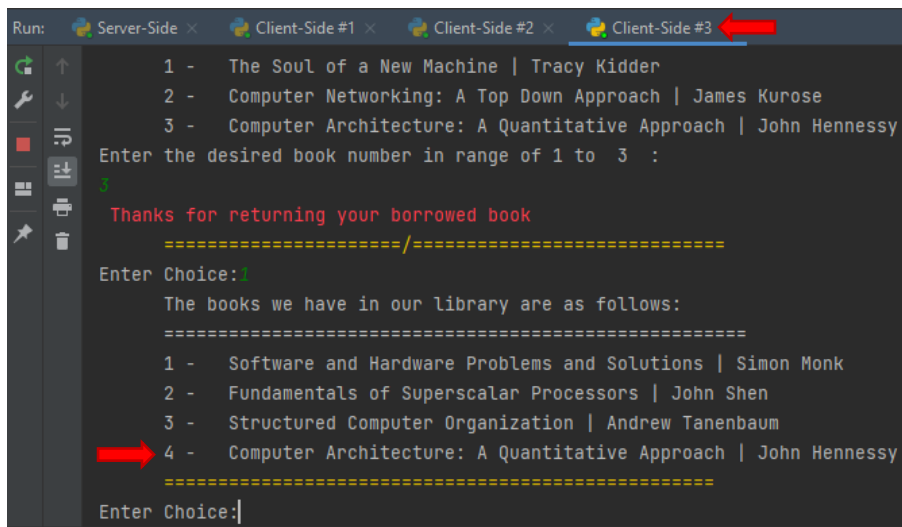
طبیعتاً به محض اینکه هرکاربری کتابی را به کتابخانه برگرداند، تمام کاربران متصل و کاربران آینده سیستم آن کتاب را در لیست کتب قابل دریافت مشاهده میکنند. هرکاربر طی ارتباط با سرور پس از اجرای هر درخواست همچنان به سرور متصل است و میتواند درخواست جدیدی به سرور ارایه کند، مگراینکه که خود گزینه Exit را انتخاب کند و یا اینکه، اگر کاربر لاگین شده ادمین نباشد و درخواست اختیارات انحصاری که برای ادمین درنظر گرفته شده است را از سرور داشته باشد.



```
Run: Server-Side x Client-Side #1 x Client-Side #2 x Client-Side #3 x
=====
1 - The Soul of a New Machine | Tracy Kidder
2 - Computer Networking: A Top Down Approach | James Kurose
3 - Computer Architecture: A Quantitative Approach | John Hennessy
Enter the desired book number in range of 1 to 3 :
3
Thanks for returning your borrowed book
=====
Enter Choice:|
```

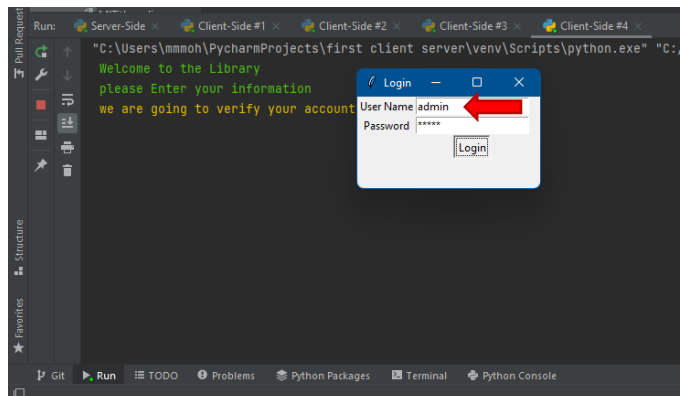


```
Run: Server-Side x Client-Side #1 x Client-Side #2 x Client-Side #3 x
Enter Choice:|
The books we have in our library are as follows:
=====
1 - Software and Hardware Problems and Solutions | Simon Monk
2 - Fundamentals of Superscalar Processors | John Shen
3 - Structured Computer Organization | Andrew Tanenbaum
=====
Enter Choice:|
The books we have in our library are as follows:
=====
1 - Software and Hardware Problems and Solutions | Simon Monk
2 - Fundamentals of Superscalar Processors | John Shen
3 - Structured Computer Organization | Andrew Tanenbaum
4 - Computer Architecture: A Quantitative Approach | John Hennessy
=====
Enter Choice:|
```



```
Run: Server-Side x Client-Side #1 x Client-Side #2 x Client-Side #3 x
1 - The Soul of a New Machine | Tracy Kidder
2 - Computer Networking: A Top Down Approach | James Kurose
3 - Computer Architecture: A Quantitative Approach | John Hennessy
Enter the desired book number in range of 1 to 3 :
3
Thanks for returning your borrowed book
=====
Enter Choice:|
The books we have in our library are as follows:
=====
1 - Software and Hardware Problems and Solutions | Simon Monk
2 - Fundamentals of Superscalar Processors | John Shen
3 - Structured Computer Organization | Andrew Tanenbaum
4 - Computer Architecture: A Quantitative Approach | John Hennessy
=====
Enter Choice:|
```

وضعیت فعلی سیستم به این شکل است که تعدادی کتاب توسط کاربران دریافت و بعضی از آنها به کتابخانه برگشت داده شد. حال با یک کلاینت دیگر به عنوان ادمین شبکه لاگین میکنیم و انتخاب آخر منو کتابخانه را بررسی میکنیم.

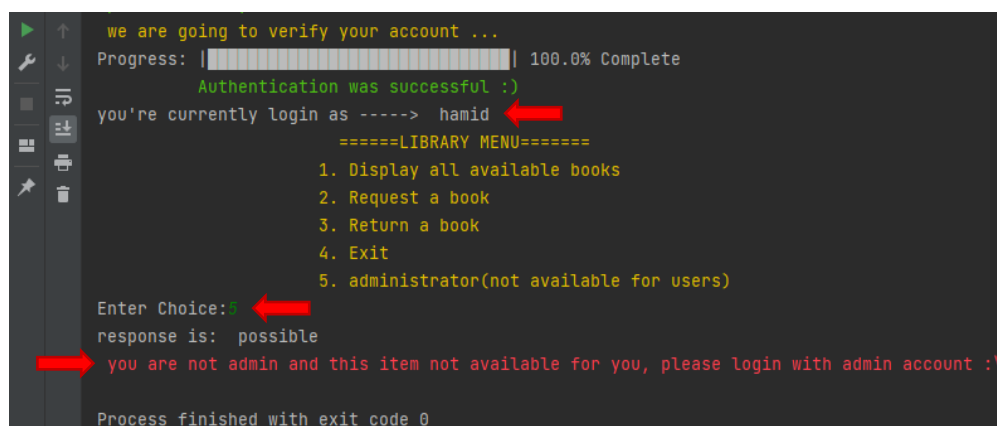


وضعیت کنونی سرور:

```
=====
Server status report:
A new client connected with the following information:
here is client: <socket.socket fd=400, family=AddressFamily.AF_INET,
Communication information: ('192.168.1.106', 2823)
connected to 192.168.1.106 2823
Number of accepted clients: 5

=====
authentication was successful :)
Current user info is: ('admin', 'admin')
```

اگر کاربر غیر ادمین برای دسترسی به امکانات ادمین شبکه اقدام کند، اجازه دسترسی به او داده نمیشود و ارتباط وی با سرور نیز قطع میشود و باید دوباره برای اتصال اقدام کند.

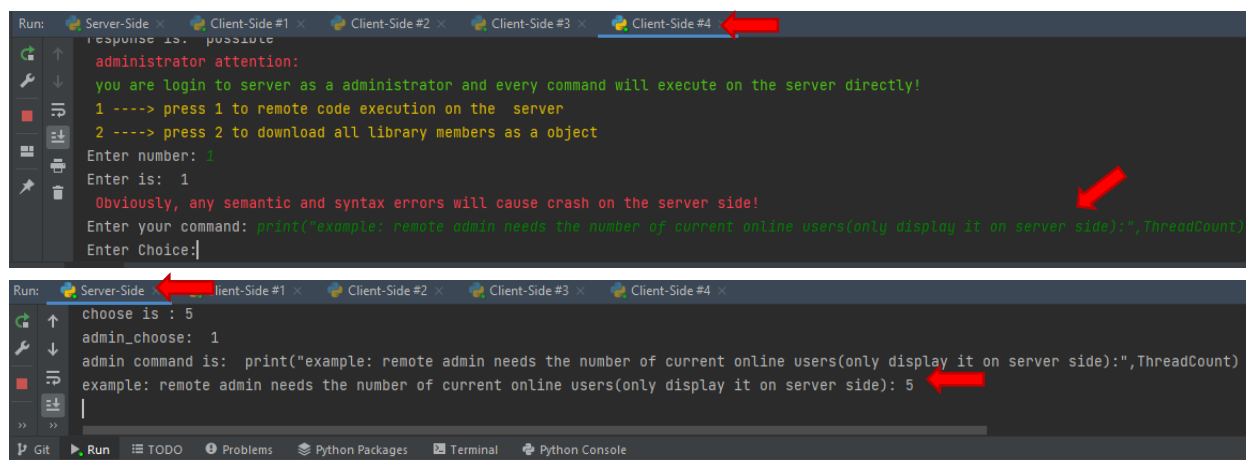


با انتخاب گزینه آخر منو توسط کلاینت ادمین، منویی شامل دو گزینه برای وی نمایش داده میشود که گزینه اول این امکان را به وی میدهد با رعایت قوانین مربوط به سینتکس و سمنتیک در سمت سرور، دستور موردنظر خود را وارد کند که این دستور برای سرور درحال اجرا دقیقاً به همان صورت اجرا خواهد شد.

```
you're currently login as ----> admin
=====LIBRARY MENU=====
1. Display all available books
2. Request a book
3. Return a book
4. Exit
5. administrator(not available for users)

Enter Choice:5
response is: possible
administrator attention:
you are login to server as a administrator and every command will execute on the server directly!
1 ----> press 1 to remote code execution on the server
2 ----> press 2 to download all library members as a object
Enter number: |
```

با انتخاب این گزینه و وارد کردن دستور مورد نظر داریم:



```
Run: Server-Side x Client-Side #1 x Client-Side #2 x Client-Side #3 x Client-Side #4
response is: possible
administrator attention:
you are login to server as a administrator and every command will execute on the server directly!
1 ----> press 1 to remote code execution on the server
2 ----> press 2 to download all library members as a object
Enter number: |
Enter is: 1
Obviously, any semantic and syntax errors will cause crash on the server side!
Enter your command: print("example: remote admin needs the number of current online users(only display it on server side):",ThreadCount)
Enter Choice:|

Run: Server-Side x Client-Side #1 x Client-Side #2 x Client-Side #3 x Client-Side #4
choose is : 5
admin_choose: 1
admin command is: print("example: remote admin needs the number of current online users(only display it on server side):",ThreadCount)
example: remote admin needs the number of current online users(only display it on server side): 5
```

طرح مشکل: این مساله در ابتدا، زمانی که تمام درخواست ها توسط سرور به تنهایی پاسخ داده میشد، میتوانست یک مشکل امنیتی بزرگ برای سیستم باشد اما به دلیل ارجاع دادن هر کلاینت به یک نخ توسط سرور، این دستور نهایتاً بتواند در ارتباط کلاینت فعلی با سرور ایجاد مشکل کند و مشکل توقف را در سیستم ما ایجاد نمیکند ولی با اینکه عدم امکان دسترسی به سایر کاربران توسط هر کاربر لاگین شده در سمت سرور درنظر گرفته شده است، این کاربر درصورت ارایه دستور نامعتبر نهایتاً کانکشن خود با سرور را از دست میدهد ولی درصورت آگاهی از سیستم پیاده سازی سمت سرور میتواند با دسترسی به کتب و سایر بخش ها، مشکلاتی را در کیفیت خدمات ارایه شده به سایر کلاینت ها ایجاد کند. به همین دلیل باید برای فرایند احراز هویت اهمیت بالاتری درنظر بگیریم و ادمین شبکه نیز از خطرات احتمالی آگاه باشد و سوی دیگر میتوان در سمت سرور یک فیلتر قرارداد که وجود کلیدواژه های خطرآفرین و غیرمجاز در دستورات دریافتی تا حدودی کنترل شوند (مانند فراخوان دسترسی به آبجکت ها و یا ساختمان ذخیره سازی اطلاعات کاربران)

در قسمت دوم انتخاب های منو برای ادمین، قسمتی در نظر گرفته شده است که تمام کتب و کاربران از سرور به کلاینت منتقل میشوند و کانستراکتور متناسب این داده ها برای دوباره ساختن به شکل فرزند از آبجکت های سمت سرور در کلاینت قرار گرفته اند.

```
you're currently login as ----> admin
=====LIBRARY MENU=====
1. Display all available books
2. Request a book
3. Return a book
4. Exit
5. administrator(not available for users)

Enter Choice: 5
response is: possible
administrator attention:
you are login to server as a administrator and every command will execute on the server directly!
1 ----> press 1 to remote code execution on the server
2 ----> press 2 to download all library members as a object
Enter number: 2
```

به دلیل اینکه پس از دریافت این آبجکت ها برای نمایش صحت انجام این دستور راهکاری وجود نداشت، صرفا به نمایش کتب قابل دسترس و امانت گرفته شده و کاربران سیستم، اکتفا شد و البته در صورت هرگونه خواسته ای که باید توسط برنامه نویس پیاده سازی میشد، در زمان آرایه حضوری امکان برآورده کردن خواسته جنابعالی با افزودن دستور مناسب برای اطمینان از انجام انتقال آبجکت از سرور به کلاینت، امکان پذیر میباشد ولی در زمان تهیه این گزارش، روش دیگری برای اطمینان یافتن از این انتقال یافت نشد. در ادامه پس از نمایش کاربران یک دستور پرینت هم بر روی آرایه کاربران قرار گرفت تا با نمایش تایپ تاحدودی اطمینان لازم حاصل شود.

```
Enter number: 2
Enter is: 2
Download is going to start ...
0 index is: ('ali', '1985')
1 index is: ('amin', '1998')
2 index is: ('hamid', '2000')
3 index is: ('admin', 'admin')
[<__main__.Student object at 0x0000024E3E726B80>, <__main__.Student object at 0x0000024E3E72CF40>, <__main__.Student object at 0x0000024E3E72D400>, <__main__.Student object at 0x0000024E3E72D400>]
Library status report:
available books:
The Soul of a New Machine | Tracy Kidder
Structured Computer Organization | Andrew Tanenbaum
Computer Networking: A Top Down Approach | James Kurose
not available books:
[' Fundamentals of Superscalar Processors | John Shen', ' Computer Architecture: A Quantitative Approach | John Hennessy', ' Software and Hardware Problems | John Hennessy']
Enter Choice: 1
```

برداشت بنده از فرمایشات جنابعالی برای امکان انتقال آبجکت، به دلیل تاکید شما مبنی بر استفاده از وراثت در سرور و کلاینت بدین صورت بود که امکان دوباره ساختن اشیا در سمت کلاینت از طریق وجود کانستراکتوری که در کلاینت وجود دارد صورت گیرد ولی اگر definition آبجکت نباید در سمت کلاینت وجود داشته باشد، راهکار این است که میتوان با ذخیره سازی خط به خط تعریف آن آبجکت در یک آرایه به شکل رشته و ارسال آن به کلاینت و اجرای تک تک و به ترتیب این داده های دریافتی، این امکان را فراهم کرد که برای اجرای این داده به عنوان دستور نیز میتوان از همان امکانی که در قسمت اول منوی ادمین برای اجرای دستور در سمت کلاینت بر روی سرور استفاده کردیم، بهره ببریم که در صورت صلاحدید جنابعالی در زمان تحویل پروژه امکان پیاده سازی این مورد وجود دارد.

قسمتی از کد کلاینت که وظیفه ساختن آجکت ها بر اساس داده های دریافتی را برعهده دارد:

```
temp_name = temp[0]
temp_pass = temp[1]
student = Student(temp_name,temp_pass)
final_memebers.append(student)
final_memebers.pop(0)
final_memebers.pop(0)

for i in range(0, len(final_memebers)):
    print(i,"index is: ", final_memebers[i].get_value())
print(final_memebers)

final_list = ["none"]
currentbooks = response2.split("@")
for i in range(0, len(currentbooks)):
    if i != 0 and i != 1:
        final_list.append(currentbooks[i])

prGreen("Library status report:")
prGreen("available books:")
for i in range(1,len(final_list)):
    prGreen_(final_list[i])
prRed("not available books: ")
prRed(Diff(final_list, all_books_in_library))
```

- در این پیاده سازی برای ارتباط کلاینت و سرور از کتابخانه سوکت پایتون استفاده شد.
- فرایند اینکود و دیکود کردن داده ها از سوکت به وسیله `decode("UTF-8")` و `str.encode()` انجام شد.
- تابع `stdoutIO` در سمت سرور برای اجرای دستوراتی دریافتی از کلاینت به طور مستقیم قرار گرفته است.
- بخش وسیعی از کدهای سمت کلاینت مربوط به کتابخانه `tkinter` برای نمایش قسمت واردکردن یوزر پسورد و نمایش قسمت بررسی اطلاعات کاربر برای احراز هویت است و تناقضی با بحث سبک بودن کلاینت در مقایسه با سرور ندارد.
- مواقعی که حجم بالای از اطلاعات باید منتقل میشد، به دلیل بالا بردن کارایی و جلوگیری از انتقال مستقل داده ها، در سمت سرور داده ها با نماد گذاری خاصی مانند `#` به یکدیگر الحاق شدند و در سمت کلاینت به کمک تابع `split` دوباره از یکدیگر تفکیک شدند.
- از **ایراداتی** که میتوان به این پیاده سازی وارد دانست، بحث گزارش لحظه ای سرور از تعداد کلاینت های متصل است که در صورتی که کلاینت خود گزینه `Exit` را انتخاب نکند و اتصال دچار مشکل شود و کلا کد سمت کلاینت متوقف شود، از تعداد کلاینت های متصل سمت سرور کاسته نمیشود و انتظار میرفت که برنامه نویس با استفاده از `try-except` مانند هنگام اتصال سرور و کلاینت رویکردی برای کنترل خطاهای احتمالی برای نخ های اختصاصی هر کلاینت داشته باشد چراکه در بحث مقیاس بندی احتمال دارد تعداد زیادی نخ مشغول باشند و امکان ورود کاربر جدید نباشند درحالیکه آن نخ دچار مشکل شده است و هرگز به چرخه استفاده دوباره بازخواهد گشت.

بررسی اهداف سیستم های توزیع شده در این پیاده سازی:

۱ – امکان استفاده مشترک از منابع پردازشی به طور شفاف در این پیاده سازی وجود دارد و در صورتی که سختگیری های امنیتی برای پروژه در فضای آکادمیک را کم رنگتر در نظر بگیریم، میتوان با امکاناتی که به ادمین ارائه شده است امکان اشتراک داده ها و منابع را نیز تاحدودی فراهم دانست.

۲ – شفافیت توزیع در این پیاده سازی کاملاً رعایت شده است اما باید در نظر داشت که این پروژه صرفاً جهت آشنایی بیشتر دانشجو با این سامانه ها و بر روی یک ماشین صورت گرفته است و طبیعتاً بحث مدیوم ارتباطی و شرایط شبکه برای ارتباط کاملاً توسط پردازنده سیستم برطرف میشود و درواقع آنچنان معنایی ندارد ولی زمانیکه سرور و کلاینت به صورت حقیقی در مکان های متفاوت حضور دارند، قطعاً ملاحظات بیشتری برای پیاده سازی وجود خواهد داشت.

۳ – بحث Openness را هم به دلیل وجود واسط کاربری کاربری برای ارتباط کلاینت با سرور و همچنین سادگی ارتباط داشتن سرور با سایر سامانه با همان سازوکار ارتباط با نودها دیگر به وسیله تعیین وظیفه برای نخ های سیستم، میتوان ارتباط با سایر سامانه هارا نیز به همین شکل برای سرور فراهم کرد و همچنین قابلیت گسترش نیز دارد. از طرفی به دلیل عدم تنظیم دقیق IP برای سرور و کلاینت، این پروژه بر روی هرماشین در همین حالت اجرای مجازی بر روی یک ماشین یکسان امکان پذیرست و در صورت انتقال به ماشین های مجزا در صورت صرف نظر از ملاحظات بحث شبکه و ارتباط، برای ارتباط فقط باید قسمت IP در سرور و کلاینت با اطلاعات مناسب تنظیم شود به پورت های مشغول این دوماشین نیز توجه شود.

```
# -----  
#Network config  
-----  
clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
host = socket.gethostname() ←  
port = 10000  
# -----
```

۴ – بحث مقیاس پذیری به دلیل یک ایراد مطرح شده در صفحه قبل مقداری با مشکل روبروست ولی با بررسی های انجام شده ، خود پایتون در صورت بروز مشکل در هر نخ، آن نخ را متوقف کرده و امکانات این نخ به سیستم باز میگردد و با این اوصاف سیستم از این بابت دچار مشکل مقیاس پذیری نیست و صرفاً مشکل ارائه گزارش نادرست به ادمین شبکه است. از سوی دیگر به دلیل پیاده سازی سرور به عنوان پاسخ دهنده اصلی درخواست ها، قطعاً نمیتوان سیستم را با Vertical scaling مدیریت کرد و از سوی دیگر سرور نقطه گلوگاه سیستم نیز میباشد و باید نسخه های متعددی از سرور در دسترس باشند که مدیریت این نسخه و چگونگی دسترسی هرکلاینت به کدام نسخه نیز باید بررسی شود. نتیجه تحقیقات به صورت انحصاری برای این پیاده سازی و ماکسیموم حد start_new_thread عدد مشخصی نیست و کاملاً بستگی به امکانات سخت افزاری ماشین مورد استفاده دارد.

ایفای نقش server stub & client stub :

```
256
257 while True:
258
259     Input = input("Enter Choice:")
260     while (not Input.isnumeric()):
261         Input = input("Please enter a number! : ")
262
263     if int(Input) == 1:...
264
265     elif int(Input) == 2:...
266
267     elif int(Input) == 3:...
268
269     elif int(Input) == 5:...
270
271     elif int(Input) == 4:...
272
273     # clientsocket.send(str.encode(Input))
274     # response = clientsocket.recv(1024)
275
276 else:
277     prRed("Authentication was unsuccessful :(")
```



































































حلقه روبرو با کمک یک window که به وسیله کتابخانه tkinter نمایش داده میشود، با ارایه یک واسط در سمت کلاینت وظیفه client stub و ارایه یک اینترفیس مشابه با متود طراحی شده در سمت سرور را برای کاربر در سمت کلاینت را برعهده دارد.

```
277 connection.sendall(str.encode("yes"))
278 while True:
279
280     data = connection.recv(2048)
281
282     data = data.decode("UTF-8")
283     choice = int(data)
284     print("choose is : ", choice)
285
286
287     if choice == 1:...
288
289     elif choice == 2:...
290
291     elif choice == 3:...
292
293     elif choice == 5:...
294
295     elif choice == 4:...
296
297     ...
298     if not data:...
299     ..
300
301     # connection.close()
302
303 else:
```

حلقه روبرو نیز در سمت سرور با کمک کتابخانه سوکت و داده هایی که از سمت کلاینت و به صورت خاص، همان client stub دریافت میکند، این مقادیر را به عنوان ورودی به متود های لوکال خود ارایه کرد و پس از دریافت نتیجه، آن را اینکود کرده و برای client stub ارسال میکند.

قسمت برقراری اتصال و استفاده از try-except برای مدیریت و کنترل خطا در سرور و کلاینت :

```
"""
# A try block allows you to handle an expected error. The except block should only
# catch exceptions you are prepared to handle. If you handle an unexpected error, your code
# may do the wrong thing and hide bugs.
# print("waiting for connections")
try:
    clientsocket.connect((host,port))
except:
    print("Sth Went wrong during binding! ")
# except socket.error as e:
#     print(str(e))
#
```

admin_choose  Mohammadmahdi Mohammadi • 1d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
download all library members as a obje...  Mohammadmahdi Mohammadi • 1d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
semantic and syntax errors  Mohammadmahdi Mohammadi • 1d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
admin command  Mohammadmahdi Mohammadi • 2d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
direct execute  Mohammadmahdi Mohammadi • 2d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
Update c1.py  Mohammadmahdi Mohammadi • 2d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
expected erro  Mohammadmahdi Mohammadi • 2d	Update client side.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
return book handling  Mohammadmahdi Mohammadi • 2d	Update SS.py  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
Update SS.py  Mohammadmahdi Mohammadi • 2d	admin  Mohammadmahdi Mohammadi • 1d	Update Client-Side #1.py  Mohammadmahdi Mohammadi • 8h
login_chec  Mohammadmahdi Mohammadi • 2d	Availablebooks  Mohammadmahdi Mohammadi • 1d	comment  Mohammadmahdi Mohammadi • 21h
Update SS.py  Mohammadmahdi Mohammadi • 2d	Update c1.py  Mohammadmahdi Mohammadi • 1d	Update IO handler.py  Mohammadmahdi Mohammadi • 23h
Update client side.py  Mohammadmahdi Mohammadi • 4d	client_thread  Mohammadmahdi Mohammadi • 3d	enumerate  Mohammadmahdi Mohammadi • 2d
Update client side.py  Mohammadmahdi Mohammadi • 4d	client response  Mohammadmahdi Mohammadi • 3d	Update c1.py  Mohammadmahdi Mohammadi • 2d
Update client side.py  Mohammadmahdi Mohammadi • 4d	method config  Mohammadmahdi Mohammadi • 3d	progress  Mohammadmahdi Mohammadi • 2d
Object definition  Mohammadmahdi Mohammadi • 4d	remote methods  Mohammadmahdi Mohammadi • 3d	terminating connection  Mohammadmahdi Mohammadi • 2d
security check  Mohammadmahdi Mohammadi • 4d	Update SS.py  Mohammadmahdi Mohammadi • 3d	garbage collection  Mohammadmahdi Mohammadi • 2d
Tkinter  Mohammadmahdi Mohammadi • 4d	Update client side.py  Mohammadmahdi Mohammadi • 4d	final check :]  Mohammadmahdi Mohammadi • 2d
Contex_manager  Mohammadmahdi Mohammadi • 5d	Update client side.py  Mohammadmahdi Mohammadi • 4d	authentication  Mohammadmahdi Mohammadi • 2d
EXEC()  Mohammadmahdi Mohammadi • 5d	Update client side.py  Mohammadmahdi Mohammadi • 4d	Update SS.py  Mohammadmahdi Mohammadi • 2d
socket binding  Mohammadmahdi Mohammadi • 5d	Library management  Mohammadmahdi Mohammadi • 4d	Update SS.py  Mohammadmahdi Mohammadi • 2d
single thread to multi  Mohammadmahdi Mohammadi • 6d	Update client side.py  Mohammadmahdi Mohammadi • 4d	second checkpoint  Mohammadmahdi Mohammadi • 2d
Initial commit  Mohammadmahdi Mohammadi • 6d	Update client side.py  Mohammadmahdi Mohammadi • 4d	check point  Mohammadmahdi Mohammadi • 2d