

NÂNG CAO HƠN TIN HỌC

Đánh dấu vào ngón chân với ổ cắm Python

Khóa học SQA Công việc

Năm - 2020, SCN -

Moustapha Isaac Diaby

NỘI DUNG

Phân tích	3
Mục tiêu của dự án (SDD) này	3
Phạm vi	3
Hạn chế	4
Hạn chế	4
Ranh giới	5
Người dùng cuối / Yêu cầu chức năng	5
Kế hoạch dự án	6
Sơ đồ UML	số 8
Biểu đồ Gantt	9
Thiết kế	10
Thiết kế cơ sở dữ liệu	10
SQL: Khởi tạo bảng và trường cơ sở dữ liệu	10
SQL: kiểm tra xem tên người dùng có trong cơ sở dữ liệu	10
SQL: tạo tài khoản người dùng mới trong cơ sở dữ liệu	10
SQL: cập nhật dữ liệu người dùng sau khi chơi trò chơi	11
Thuật toán sắp xếp	11
Khung dây của giao diện người dùng (UI) của khách hàng	12
Trang Đăng nhập / Đăng ký	12
Trang chủ	13
Tham gia trang tải trò chơi	13
Trang trò chơi	13
Trang Ban lãnh đạo	13
Các lớp UML	16
Lớp Socket Server	16
Lớp Máy chủ Socket Máy khách	16
Thực hiện / thiết kế - "Hành động" sẽ là gì?	19
Mã giả: Hành động	20
Các loại hành động	20
Ví dụ sử dụng: Mã giả để xác thực người dùng (OOP)	26
Thực hiện	29
Nghiên cứu	29

Ô cấm Python & các đối tượng của Python	29
lập trình đồng thời trong python	33
Các hành động đóng gói và đóng gói	35
Xử lý các ứng dụng khách chưa được xác thực	36
[ĐĂNG NHẬP NGƯỜI DÙNG]	36
[ĐĂNG KÝ NGƯỜI DÙNG]	38
Nhật ký sự cố:	39
Thử nghiệm các tính huống khác nhau: Giao diện người dùng	40
Trang xác thực	40
Kiểm tra đầu vào không chính xác	41
Đăng ký nút Kiểm tra	43
Nút đăng nhập	46
Cung cấp thành công mật khẩu phù hợp với tên người dùng tồn tại trong cơ sở dữ liệu	47
Trang Ban lãnh đạo	48
Tham gia trang tài trợ chơi	51
Nhận thành công 2 người chơi trong một trò chơi	52
Nhiều kết nối máy chủ 6	55
Sự đánh giá	56

PHÂN TÍCH

MỤC TIÊU CỦA DỰ ÁN NÀY (SDD)

Để phát triển trò chơi tick tack toe sử dụng Sockets để cho phép người chơi (máy khách) đấu với máy khách khác trên các máy tính khác nhau (hoặc giống nhau) trên cùng một mạng. Ứng dụng này sẽ là một trò chơi thú vị dành cho các học sinh để câu nhau trong trò chơi tic tac toe.

Tôi sẽ cần phát triển một máy chủ xử lý xác thực, các lệnh SQL với cơ sở dữ liệu và kết nối máy khách với các máy khách khác trong một phiên trò chơi. Điều này có thể đạt được bằng cách sử dụng các ỗ cắm trong python.

Tôi sẽ cần xác định cấu trúc bảng cơ sở dữ liệu phù hợp cho ứng dụng sẽ lưu trữ dữ liệu người chơi như số trận thắng, trận thua, trò chơi đã chơi, tên người dùng và mật khẩu.

Người chơi cũng sẽ có thể xem bảng xếp hạng sẽ được sắp xếp bằng thuật toán sắp xếp chèn (nghịch đảo).

Điều này sẽ làm cho trò chơi cảm thấy cạnh tranh hơn vì học sinh sẽ muốn leo lên bảng dẫn đầu để trở thành người giỏi nhất. Người chơi cũng sẽ có thể tìm kiếm thứ hạng của họ và bất kỳ người dùng nào khác thông qua tên người dùng của họ, điều này cung cấp một cách nhanh chóng và dễ dàng để tìm thứ hạng của người chơi.

Người chơi sẽ có thể tạo người dùng mới bằng cách gửi tên người dùng và mật khẩu duy nhất đến máy chủ - sẽ tạo và lưu thông tin đăng nhập người dùng mới vào cơ sở dữ liệu.

Người chơi cần được xác thực để sử dụng các chức năng chính của ứng dụng. Điều này sẽ được thực hiện bằng cách chuyển thông tin đăng nhập hợp lệ của người dùng (tên người dùng và mật khẩu) đến máy chủ - sẽ so sánh mật khẩu mà máy khách chưa được xác thực đã gửi với mật khẩu chính xác cho tên người dùng mong muốn trong cơ sở dữ liệu.

Sau khi được máy chủ xác thực, máy chủ sẽ gửi lại máy khách dữ liệu người dùng đã lưu từ cơ sở dữ liệu và nhận ra bất kỳ yêu cầu hành động nào trong tương lai từ máy khách đó là trình phát đã được xác thực.

PHẠM VI

Để hoàn thành dự án này, tôi cần thực hiện những việc sau:

- Tạo thiết kế giao diện người dùng khung dây trực quan cho các chế độ xem khác nhau của khách hàng. Điều này nên bao gồm trang xác thực, trang chủ, trang trong trò chơi và trang bảng lãnh đạo. Các khung nhìn khác cũng sẽ được thiết kế nếu tôi cần.
- Tôi cũng sẽ cần thiết kế một biểu đồ UML ca sử dụng sẽ cho thấy cách người dùng sẽ tương tác với máy khách và sau đó với máy chủ.
- Tạo một biểu diễn mã giả của một "Hành động", đây sẽ là định dạng đối tượng sẽ được gửi và nhận từ máy khách. Khái niệm "Hành động" sẽ giúp gửi dữ liệu đến máy chủ hoặc máy khách - có cấu trúc và dễ dàng hơn nhiều để xử lý dữ liệu được gửi vì nó có thể dự đoán được.
- Lập trình giao diện người dùng được thiết kế với python Tkinter và làm cho mỗi nút hoạt động chính xác hàm số.
- Nghiên cứu về cách sử dụng các ỗ cắm với các đối tượng Python và gấp các đối tượng Python. ỗ cắm và tẩy đi vượt quá trình cao hơn trước; tuy nhiên, nó sẽ được yêu cầu để hoàn thành ứng dụng này. Nó được sử dụng bởi cả máy khách và máy chủ để gửi các đối tượng python đã ngâm cho nhau theo byte. (sau đó bỏ chọn các byte trả lại đối tượng python khi dữ liệu được nhận).

- Lập trình máy chủ ở cảm biến bằng python. Máy chủ phải có khả năng xử lý mọi "Hành động" đã xác định chính xác và chuyển lại dữ liệu chính xác (bao gồm cả lỗi) cho khách hàng đã gửi "Hành động".
- Lập trình hệ thống đăng ký / xác thực đang hoạt động giao tiếp với máy chủ socket sẽ được sử dụng bởi ô cảm biến khách.
- Duy trì dữ liệu người dùng thông qua SQLite làm cơ sở dữ liệu.
- Lập trình một trò chơi tic tac toe hoạt động hoàn toàn, biết khi nào trò chơi kết thúc (nếu một người chơi thắng hoặc trò chơi là một trận hòa, đây là trạng thái kết thúc của bảng trò chơi).
- Tạo biểu diễn mã giả đang hoạt động của một loại chèn (nghịch đảo) và triển khai mã trong python để sắp xếp ban lãnh đạo dựa trên số trò chơi đã thắng.
- Tôi cũng muốn phát triển một quy trình kiểm tra mạnh mẽ, quy trình này sẽ cố gắng bao hàm một loạt các giá trị được gửi đến máy chủ bởi một máy khách thử nghiệm và kiểm tra xem tất cả các yêu cầu chức năng có đang hoạt động hay không.
- Cuối cùng viết đánh giá về dự án (độ chắc chắn của mã, khả năng bảo trì và mức độ phù hợp với mục đích là ứng dụng)

GIỚI HẠN

Đây là những điều mà ứng dụng này sẽ không xử lý hoặc mong đợi người chơi làm:

- o Máy chủ sẽ không hoàn toàn mong đợi việc tiêm SQL (tuy nhiên nó sẽ không giúp bạn dễ dàng thực hiện việc này. Sẽ có một số mức độ bảo vệ trong mỗi lệnh SQL)
- o Giải pháp sẽ không đảm bảo rằng chỉ có một phiên cho mỗi tên người dùng đã đăng nhập vào bất kỳ thời điểm nào. Điều này sẽ ngăn người chơi gian lận và có thể chơi với chính họ.
- o Máy chủ sẽ không tự động đóng phiên trò chơi khi một người chơi không hoạt động hoặc bị ngắt kết nối.

HẠN CHẾ

Có một số hạn chế về kỹ thuật, pháp lý và thời gian cho dự án này:

- Máy khách và máy chủ sẽ được lập trình bằng python 3.7. Tôi chọn điều này là vì tôi có nhiều kinh nghiệm sử dụng python trong trường học và trong thời gian rảnh rỗi của riêng tôi.
- Tôi sẽ sử dụng cơ sở dữ liệu SQLite và SQL để tương tác với cơ sở dữ liệu từ máy chủ. SQLite giống như một phiên bản MySQL nhỏ hơn được lưu vào một tệp, tôi chọn cơ sở dữ liệu này vì nó sẽ giúp thiết lập máy chủ dễ dàng hơn nhiều cho người dùng cuối.
- Các phiên trò chơi tick-tack-toe đồng thời mà máy chủ có thể xử lý nhiều hơn 3, do đó máy chủ phải có thể xử lý đồng thời ít nhất 6 kết nối của máy khách với máy chủ ở cảm biến.
- Không có chi phí phát triển cho dự án này vì tôi đang sử dụng thư viện tích hợp sẵn của python để phát triển toàn bộ ứng dụng. Do đó, không có chi phí phụ thuộc của bên thứ ba.
- Do GDPR, tôi sẽ phải cho phép khách hàng xem tất cả dữ liệu mà bộ điều khiển dữ liệu có trên họ. Đó là tại sao tôi không lưu trữ bất kỳ dữ liệu cá nhân nào về người dùng như tên và ngày sinh. Tất cả dữ liệu sẽ được lưu sẽ chỉ là một phần của trò chơi và có thể xem được trong phần bảng xếp hạng của ứng dụng. (Vì tôi đang làm cho ứng dụng dễ sử dụng, nên bất kỳ ai cũng có thể thiết lập máy chủ của họ và trở thành người kiểm soát dữ liệu.)
- Dự án phải được hoàn thành trước tháng 4 - đây là lúc tôi phải bàn giao dự án của mình cho SQA.
- Ứng dụng phải hoạt động trên máy tính của trường.

BOUNDARIES

Dự án này sẽ được thực hiện khi:

- Người chơi có thể đăng ký thành công tài khoản mới và xác thực vào ứng dụng.
- Người chơi có thể chơi toàn bộ trò chơi đánh dấu vào nhau và đạt đến trạng thái kết thúc (thắng hoặc hòa) trên hội đồng quản trị.
 - o Khi dữ liệu của mỗi người chơi trong cơ sở dữ liệu được cập nhật sau khi trận đấu kết thúc theo kết quả trò chơi.
- Người chơi có thể xem 20 người chơi hàng đầu trên bảng xếp hạng và có thể tìm kiếm người chơi của họ hoặc của người chơi khác thứ hạng.

YÊU CẦU CỦA NGƯỜI DÙNG / CHỨC NĂNG END

Có rất nhiều người có thể sử dụng ứng dụng này. Học sinh trung học có thể sử dụng chương trình này để chơi với bạn bè và gia đình, tuy nhiên, họ sẽ cần một máy tính / máy chủ duy nhất để lưu trữ máy chủ sẽ lắng nghe các yêu cầu hành động của máy khách. Người dùng cuối sẽ mong đợi những điều sau:

Hệ thống đăng ký và xác thực đang hoạt động có giao diện rõ ràng và dễ sử dụng để khách hàng nhập tên người dùng và mật khẩu của họ (cùng với thông tin để kết nối với máy chủ socket, ví dụ: Ip và cổng của máy chủ socket).

Người dùng sẽ xác thực ứng dụng khách của họ bằng cách nhập tên người dùng và mật khẩu của họ, mật khẩu này sẽ được gửi đến máy chủ socket nơi nó sẽ được xác minh nếu không tìm thấy tên người dùng hoặc mật khẩu trong cơ sở dữ liệu không chính xác, socket sẽ từ chối ứng dụng khách.

- Phải có kết nối với máy chủ socket sẽ nhận được hành động đăng nhập - thực hiện sau đây:
 - o Kiểm tra tên người dùng có tồn tại trong cơ sở dữ liệu hay không.
 - o Kiểm tra xem mật khẩu được gửi (được mã hóa bằng thuật toán MD5) có khớp với mật khẩu trong cơ sở dữ liệu hay không. Nếu vậy, nó sẽ xác thực socket của khách hàng và gửi lại dữ liệu của người dùng.

Khách hàng sẽ có thể đăng ký một tài khoản người dùng mới bằng cách cung cấp cho giao diện khách hàng một tên người dùng và một mật khẩu duy nhất.

- Phải có một kết nối với máy chủ socket sẽ nhận được một hành động đăng ký - thực hiện sau đây:
 - o Xác thực rằng tên người dùng là duy nhất.
 - o Mã hóa mật khẩu bằng thuật toán MD5 và đăng ký người dùng trong cơ sở dữ liệu.

Hệ thống xác thực / đăng ký của khách hàng phải kiểm tra xem tên người dùng và mật khẩu đã được cung cấp và không được để trống và mật khẩu có độ dài từ 6 trở lên, máy chủ sẽ xử lý việc xác thực và đăng ký của khách hàng.

Ứng dụng khách được xác thực sẽ được phép thực hiện những việc sau:



Thiết kế cơ sở dữ liệu (1,5 giờ):

- Tên bảng và các trường của chúng. Điều này sẽ giúp tôi hiểu rõ hơn về cách cấu trúc đầu sau.
- Tôi cũng có thể tạo một số mã SQL để lấy tất cả các trình phát và nhận một trình phát bằng tên người dùng. Bằng cách liệt kê các lệnh này, sẽ rất hữu ích khi triển khai mã SQL trên người phục vụ.

Thiết kế máy chủ (2,5 giờ):

- Lập kế hoạch các chức năng lớp sẽ được yêu cầu cho dự án để đáp ứng đặc điểm kỹ thuật. Điều này sẽ bao gồm đầu vào và đầu ra.
- Đưa ra giải pháp để kiểm soát cách Khách hàng sẽ tương tác với máy chủ và ngược lại.
- Vẽ sơ đồ dòng sẽ minh họa sự tương tác của khách hàng với máy chủ và với cơ sở dữ liệu.

Thiết kế khách hàng (2h):

- Tạo một thiết kế giao diện người dùng khung dây sẽ giới thiệu những gì khách hàng sẽ thấy trên mỗi chế độ xem.
- Khung dây cũng sẽ bao gồm các hình ảnh minh họa về các hành động. Ví dụ: nhấp vào nút, chuyển trang, v.v.

2. Thiết lập cơ sở dữ liệu (0,5 giờ):

- Điều này sẽ bao gồm việc tạo mã SQL để tạo cơ sở dữ liệu, xác định từng bảng và thêm người dùng đăng nhập thử nghiệm. (Mã này sẽ được máy chủ thực thi khi khởi động ban đầu. Điều này là do cơ sở dữ liệu SQLite sẽ cần được định cấu hình cục bộ mỗi khi máy chủ đang chạy trên cơ sở dữ liệu SQLite mới.)

3. Nghiên cứu (3h)

- Cách sử dụng ô cảm trong python.
- Ruột trăn.

4. Thiết lập máy chủ (8h):

- Cho phép người dùng kết nối với máy chủ, xác thực và thực hiện các yêu cầu trong sự chỉ rõ.
- Tôi cũng sẽ phải liên tục thực hiện một số thử nghiệm trong quá trình phát triển mỗi lần tạo một hàm. Điều này cũng có thể làm giảm thời gian cần thiết để kiểm tra.
- Lập trình cho hệ thống phiên để người chơi chơi trò chơi tick tack toe với các điều kiện kết thúc.
- Để thử nghiệm, máy chủ nên in ra tất cả các "Hành động" mà nó nhận được, sau đó là ứng dụng khách nào đã gửi nó (tên người dùng).

5. Đặt máy khách (8h):

- Tao giao diện người dùng của khách hàng từ thiết kế giao diện người dùng khung dây.
- Lập trình các chức năng khách cần thiết được chỉ định trong biểu đồ Ca sử dụng UML.
- Liên tục thực hiện các bài kiểm tra trong quá trình phát triển các chức năng của khách hàng. Tôi cũng có thể cần quay lại chương trình máy chủ và tinh chỉnh một số chức năng gây ra sự cố hoặc cần được lập trình lại.

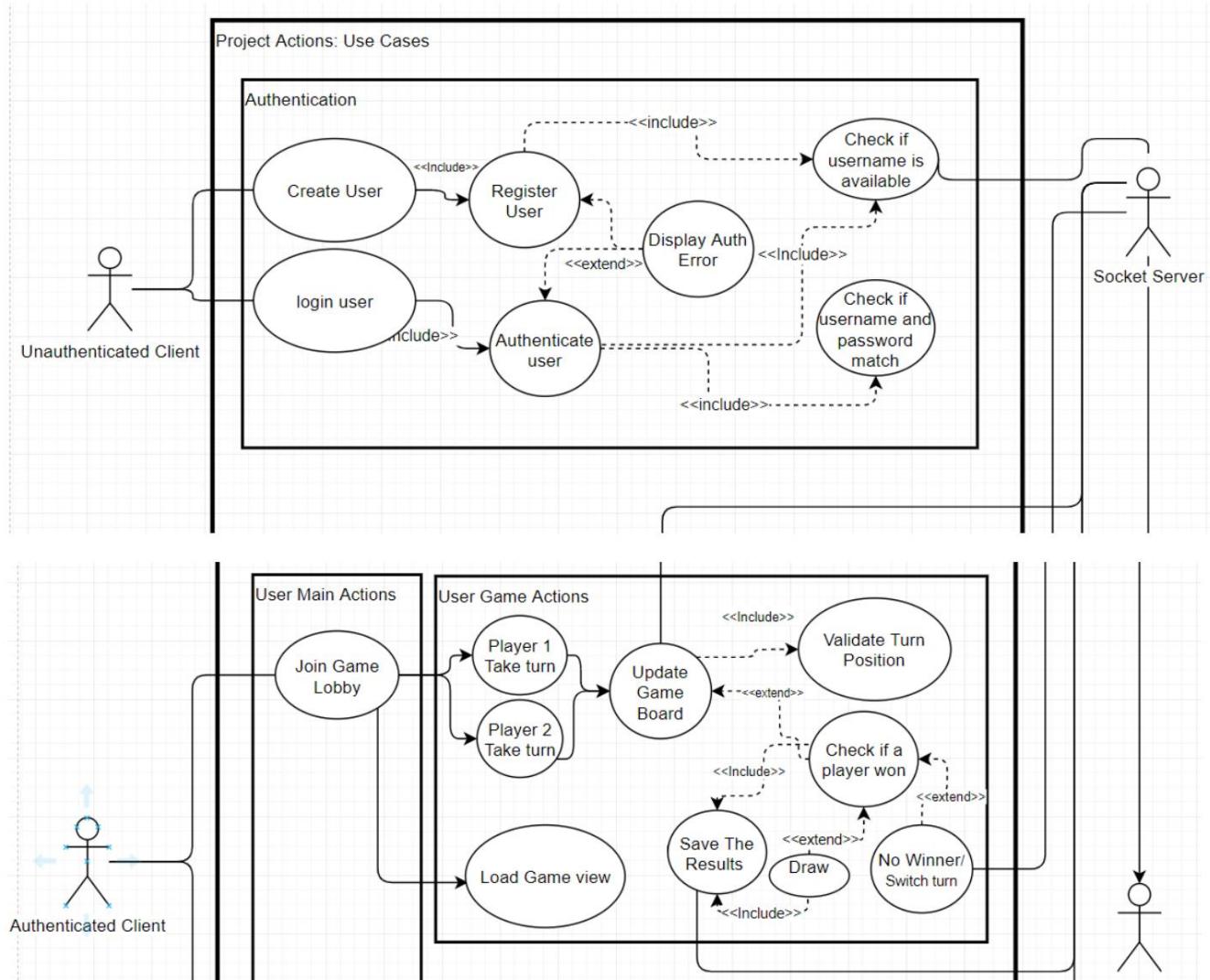
6. Kiểm tra (3h):

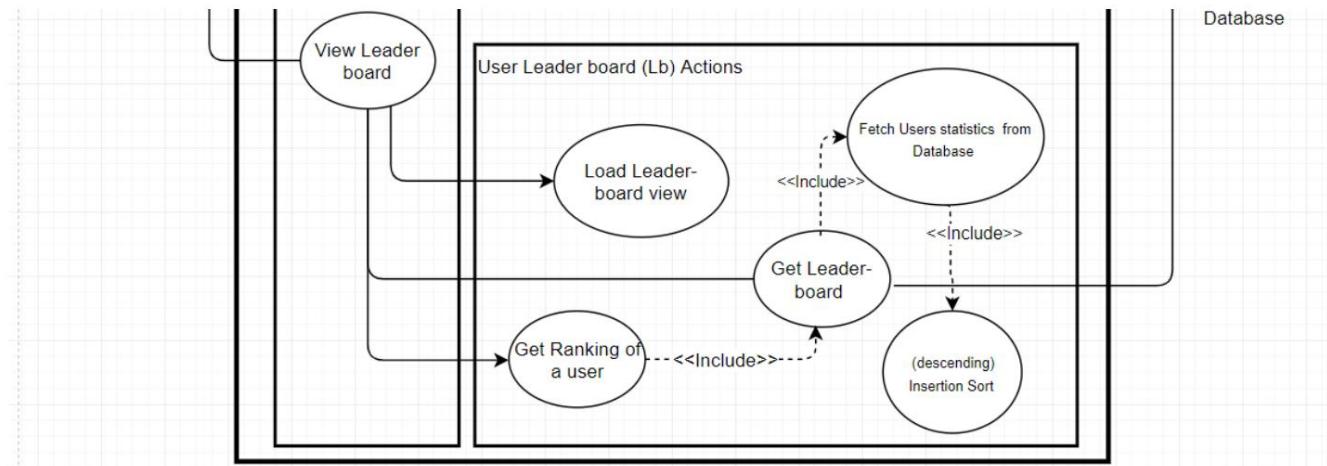
- Tôi sẽ cần kiểm tra toàn bộ ứng dụng và ghi lại kết quả của bài kiểm tra. Điều này sẽ đảm bảo rằng chương trình nằm trong phạm vi / thực hiện chính xác đến ranh giới và bao gồm các ràng buộc của đặc tả.

- Trong giai đoạn này, tôi cũng sẽ làm cho chương trình xử lý các lỗi. Ghi nhận chúng xuống như tôi làm.
7. Đánh giá chương trình. Tính chính chương trình để đáp ứng bất kỳ thông số kỹ thuật còn thiếu nào / làm cho chương trình có thể bảo trì được và đưa ra các lỗi hữu ích. (3 giờ)

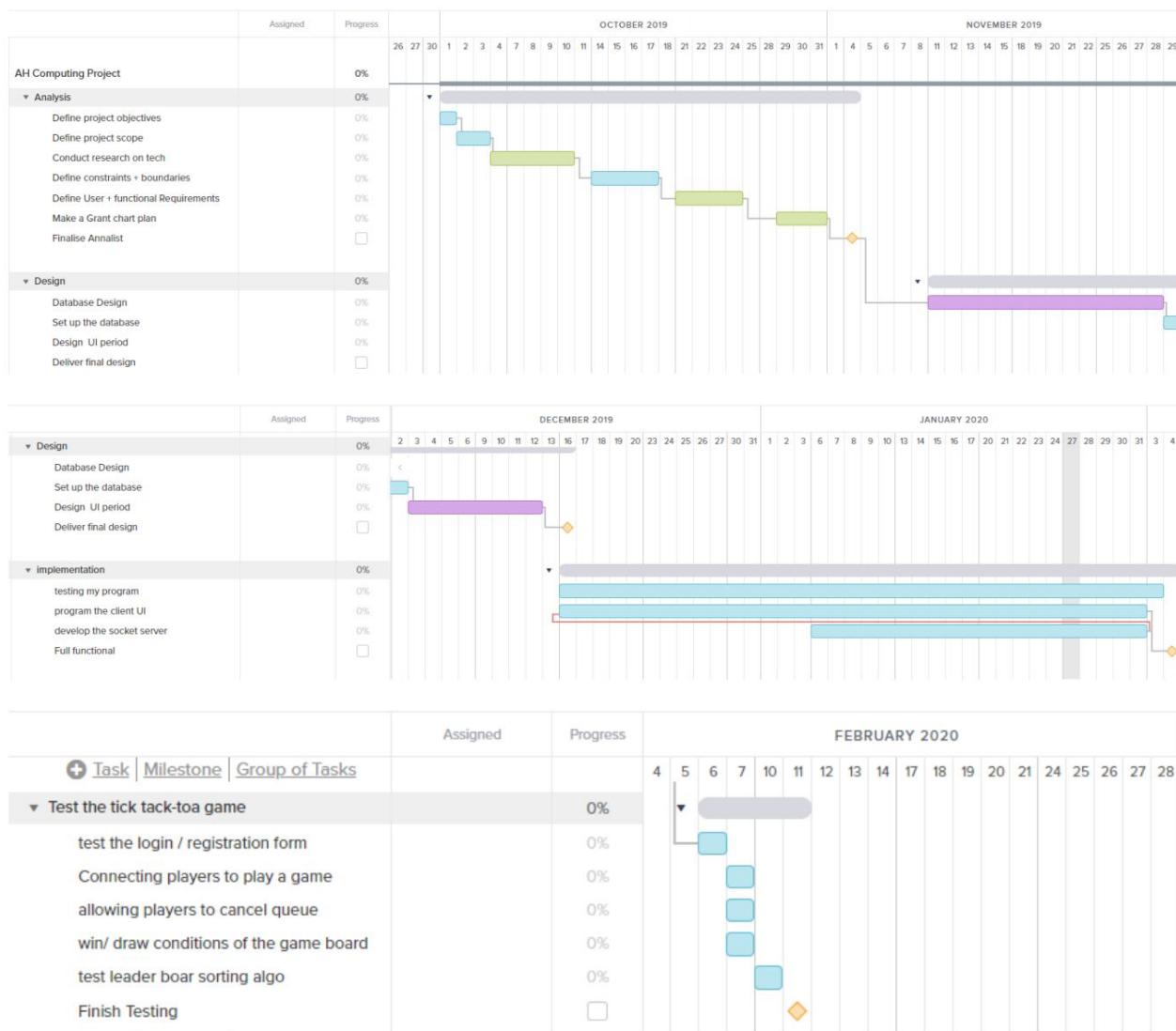
Tổng thời gian ước tính để hoàn thành ứng dụng này là 31,5 giờ. Tuy nhiên, điều này có thể thay đổi vì tôi sẽ làm việc trên nhiều phần cùng một lúc. Ví dụ, khi lập trình máy chủ, tôi cũng sẽ lập trình máy khách vì tôi sẽ cần thử nghiệm các giải pháp của mình.

Sơ đồ UML





Biểu đồ Gantt



THIẾT KẾ

Trong phần này, tôi sẽ tạo một kế hoạch về những gì tôi sẽ lập trình như wireframe, cấu trúc dữ liệu và một số khái niệm như hành động và ô cắm.

THIẾT KẾ CƠ SỞ DỮ LIỆU

Cơ sở dữ liệu sẽ là SQLite, dễ khởi động (nó chỉ là một tệp) và yêu cầu các lệnh SQL để tương tác với nó. Tất cả các truy vấn SQL sẽ được thực thi trên ô cắm máy chủ, nó bổ sung thêm một lớp bảo mật.

SQL: BẢN ĐẦU BẮNG CƠ SỞ DỮ LIỆU VÀ CÁC LĨNH VỰC

Khi khởi động máy chủ lần đầu tiên, tôi sẽ cần Khởi tạo cơ sở dữ liệu.

TẠO BẢNG NẾU KHÔNG CÓ người dùng (

```
tên người dùng VARCHAR (25) KHÔNG ĐỦ DUY NHẤT,  
mật khẩu VARCHAR (500) KHÔNG ĐỦ,  
thắng INT UNSIGNED DEFAULT 0,  
mất INT UNSIGNED DEFAULT 0,  
games_played int UNSIGNED DEFAULT 0  
);
```

- Tùy chọn “NẾU KHÔNG TỒN TẠI” sẽ ngăn máy chủ tạo thành công bảng người dùng thứ hai khi chương trình khởi động lại.
- Các trường như “thắng”, “thua” và “game_played” là các số nguyên không dấu vì bạn không thể có các trò chơi phủ định thắng, thua và chọi. Nó làm cho cơ sở dữ liệu mạnh mẽ hơn.

SQL: KIỂM TRA NẾU TÊN NGƯỜI DÙNG CÓ TRONG CƠ SỞ DỮ LIỆU

”: Tên người dùng” là một biến cho tên người dùng

CHỌN tên người dùng TỪ người dùng WHERE tên người dùng =: tên người dùng;

- Tôi không nhận được cột mật khẩu bây giờ vì tôi muốn có một lớp bảo mật. • Tôi sẽ chỉ trả lại tất cả các cột nếu mật khẩu (được mã hóa) khớp với mật khẩu đã lưu trong cơ sở dữ liệu cho tên người dùng đó.
- Nếu truy vấn này trả về null điều đó có nghĩa là không có người dùng nào trong cơ sở dữ liệu của người dùng có tên người dùng đó. Nếu khách hàng muốn đăng ký một tài khoản mới, bây giờ họ sẽ có thể làm như vậy.

SQL: TẠO TÀI KHOẢN NGƯỜI DÙNG MỚI TRONG CƠ SỞ DỮ LIỆU

CHÈN VÀO người dùng (tên người dùng, mật khẩu) GIÁ TRỊ (?,?);

- Đầu chấm hỏi (?) Được thay thế bằng giá trị tên người dùng và mật khẩu, với SQL injection sự bảo vệ. Sử dụng chúng làm cho chương trình mạnh mẽ hơn và sử dụng an toàn hơn.
- Chúng tôi không cần cung cấp các cột "thắng", "thua" và "trò chơi đã chơi" vì chúng tôi đã thiết lập giá trị mặc định của 0 khi chúng tôi tạo bảng.

SQL: ĐÃ CẬP NHẬT DỮ LIỆU NGƯỜI DÙNG SAU KHI CHƠI TRÒ CHƠI

"Tên người dùng" là một biến cho tên người dùng

Nếu người chơi thắng:

CẬP NHẬT người dùng SET thắng = thắng + 1, games_played = games_played + 1 WHERE tên người dùng =: tên người dùng

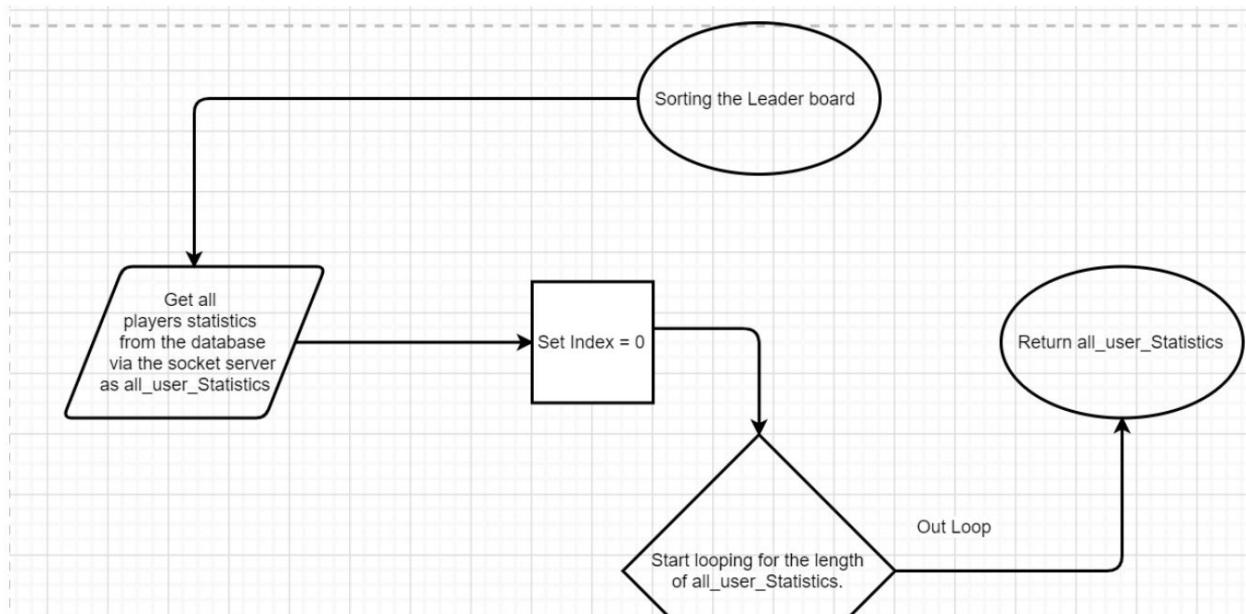
Khác:

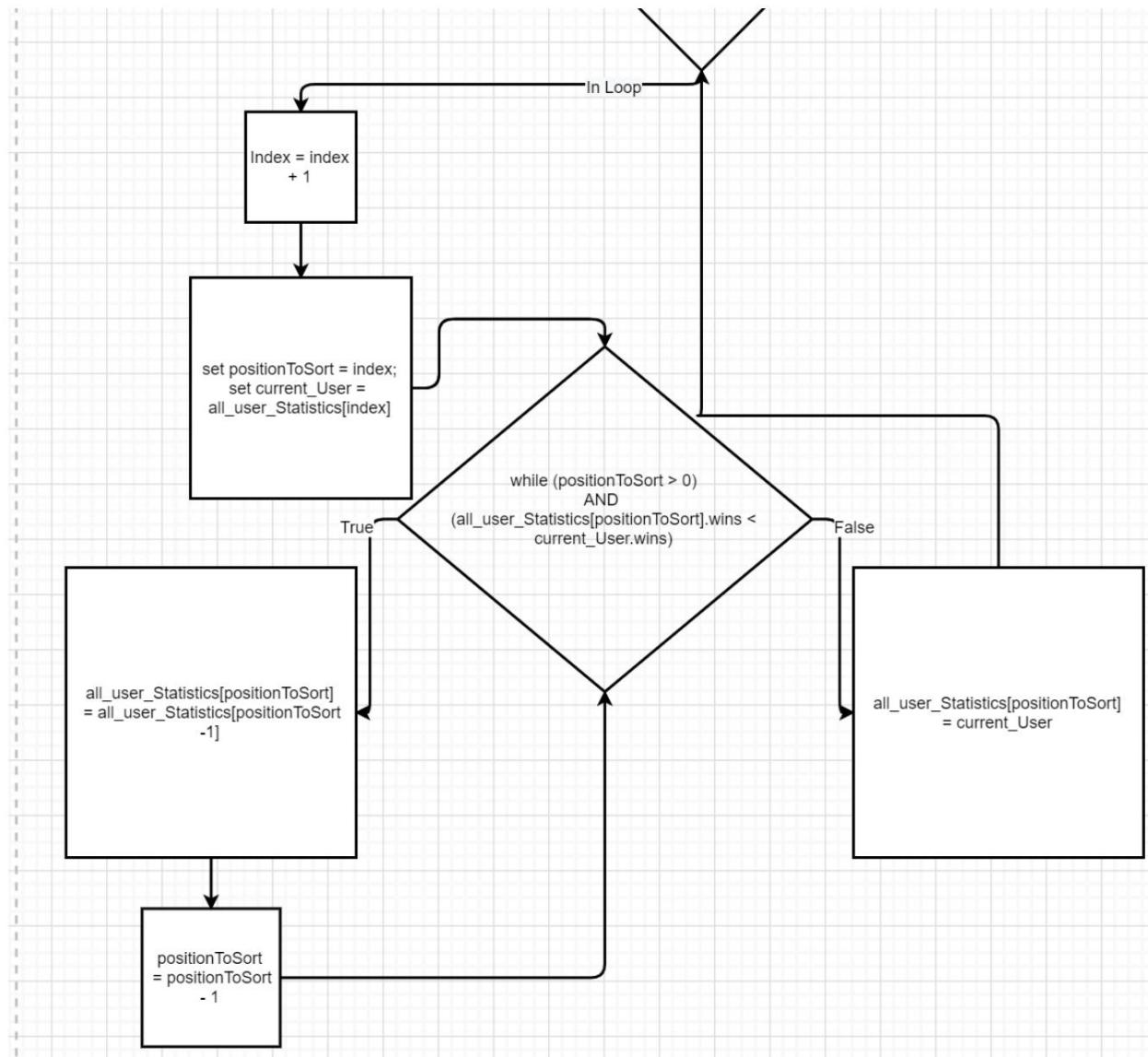
CẬP NHẬT người dùng SET thua = thua + 1, games_played = games_played + 1 WHERE tên người dùng =: tên người dùng

- Cả hai lệnh sẽ tăng số lượt chơi lên 1
- Tùy thuộc vào việc người chơi thắng hoặc thua / hòa, nó sẽ tăng số tiền thắng hoặc thua tương ứng.

THUẬT TOÁN SẮP XẾP

Vì tôi đang sử dụng sáp xếp chèn (nghịch đảo) là một khái niệm cao cấp hơn, nên tôi sẽ viết một biểu đồ chi tiết về cách tôi sẽ sử dụng nó trong ứng dụng này. Nghịch đảo của sáp xếp chèn sẽ chỉ đơn giản là có số chiến thắng lớn nhất trước (giảm dần).





DÂY GIAO DIỆN NGƯỜI DÙNG CỦA KHÁCH HÀNG (UI)

Các giao diện người dùng máy khách sau sẽ được tạo bằng python Tkinter. Mỗi thiết kế nhằm mục đích:

- Làm cho mỗi nhãn rõ ràng, dễ đoán và dễ theo dõi.
- Xem xét trải nghiệm của người dùng, bằng cách bao gồm một phần dành riêng để thông báo cho khách hàng về những gì đang xảy ra (lỗi và thông báo hành động.)
- Các thiết kế minh họa các nút điều hướng hoặc các sự kiện một cách rõ ràng.

ĐĂNG NHẬP / TRANG ĐĂNG KÝ

Phản xá thực - khách hàng sẽ có thẻ cung cấp tên người dùng và mật khẩu của họ (cả hai loại Chuỗi) mà họ muốn xác thực.

Phần thông tin máy chủ - người dùng phải có thẻ cung cấp cho máy khách vị trí của máy chủ ở cắm (loại Chuỗi) và cổng của nó (loại Số nguyên). Nếu máy khách không được hướng dẫn đúng địa chỉ của máy chủ socket và cổng đã mở của nó, máy khách sẽ không thể kết nối với máy chủ khi có găng xác thực / đăng ký. Sẽ có một thông báo lỗi hiển thị cho biết "Không thể kết nối với máy chủ ở cắm".

Nếu người dùng có găng đăng ký một tài khoản mới, sẽ có một hành động '[ĐĂNG KÝ NGƯỜI DÙNG]' được gửi đến máy chủ. Nếu tên người dùng đã được đăng ký, khách hàng sẽ được thông báo rằng tên người dùng được cung cấp đã được sử dụng. Khác Nếu tên người dùng được cung cấp chưa được đăng ký trong cơ sở dữ liệu, tài khoản sẽ được tạo và khách hàng sẽ nhận được thông báo xác nhận rằng tài khoản của họ đã được tạo và họ có thể có găng đăng nhập bằng thông tin đăng nhập của mình.

Nếu người dùng có găng đăng nhập vào tài khoản của họ, sẽ có một hành động '' [USER LOGIN] "được gửi đến máy chủ. Nếu tên người dùng không được tìm thấy trong cơ sở dữ liệu hoặc mật khẩu không chính xác, máy khách sẽ được thông báo rằng không có người dùng nào có tên người dùng đó hoặc mật khẩu không chính xác tương ứng. Nếu không, nếu cả hai thông tin xác thực được thông qua đều chính xác, máy khách sẽ được xác thực bởi máy chủ và máy khách sẽ được xác thực và gửi đến Trang chủ.

TRANG CHỦ

Trang này chỉ có thể truy cập được đối với khách hàng đã xác thực. Sẽ có một thông báo chào mừng ở đầu trang, theo sau là tên người dùng người chơi hiện đang đăng nhập. Nút "tham gia trò chơi" và "xem bảng xếp hạng" theo sau là một phần ở cuối trang cho phép tôi hiển thị bất kỳ thông tin / thông báo lỗi nào.

THAM GIA TRANG TÀI TRÒ CHƠI

Khi một người chơi muốn tham gia một trò chơi, họ sẽ được hiển thị với một trang có chứa thông báo về những gì đang xảy ra ("Đang chờ một người chơi khác tham gia") và nút hủy mà khách hàng có thể sử dụng để thoát khỏi trạng thái "đang chờ một hàng đợi trò chơi" trên máy chủ.

Nếu có người chơi thứ hai cũng đang đợi chơi trò chơi, máy chủ sẽ bắt đầu trò chơi với hai máy khách, sau đó chế độ xem sẽ chuyển sang Trang trò chơi

TRANG TRÒ CHƠI

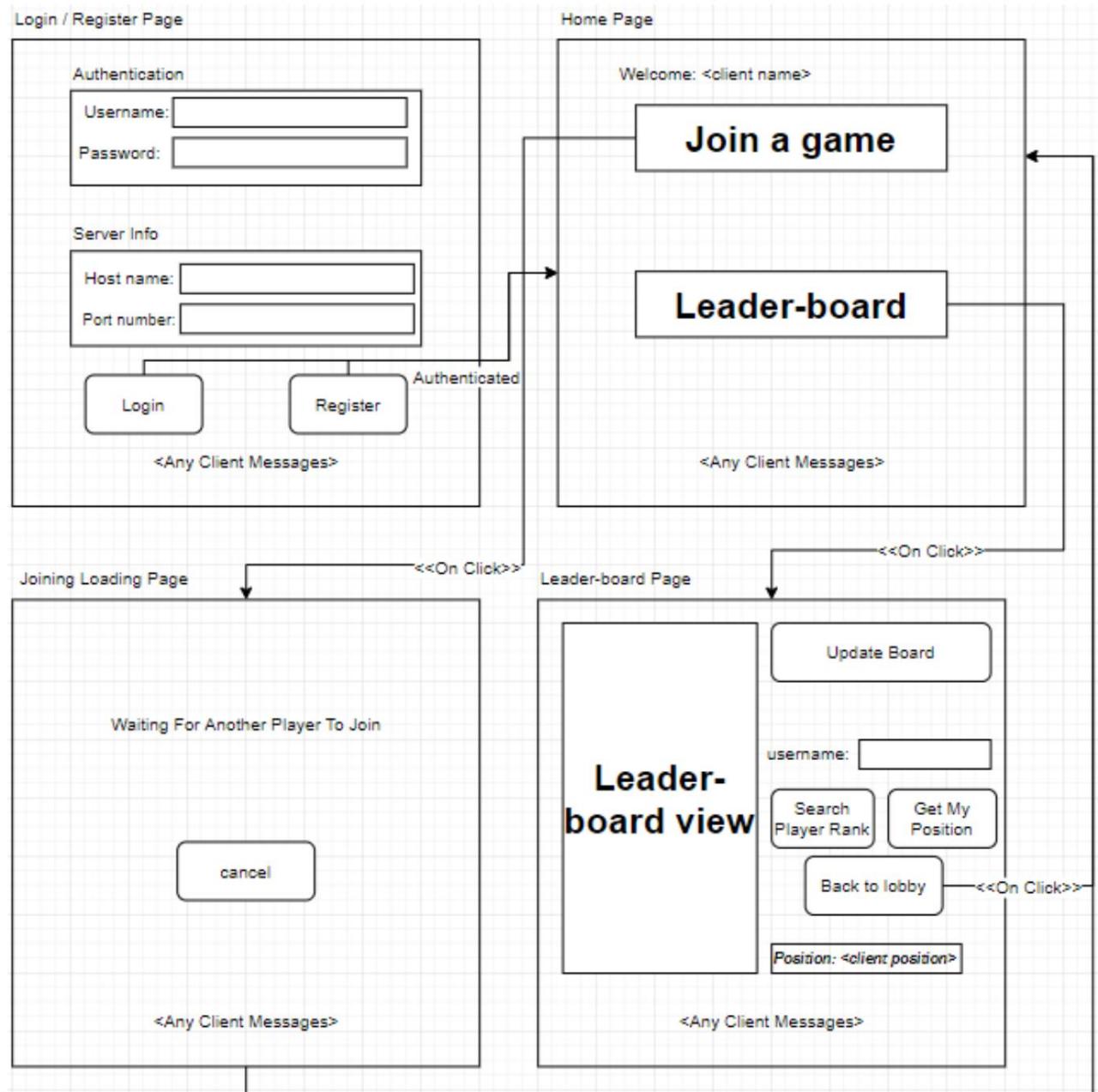
Sẽ có một bảng trò chơi 3 x 3 cho phép người chơi hiện tại thực hiện một lượt bằng cách chọn một ô vuông mà họ muốn đặt màu của mình. Nếu không đến lượt khách hàng, việc chọn các ô vuông sẽ không có tác dụng gì. Nếu không, ô vuông đã chọn sẽ được xác nhận và gửi đến máy chủ để cập nhật bảng.

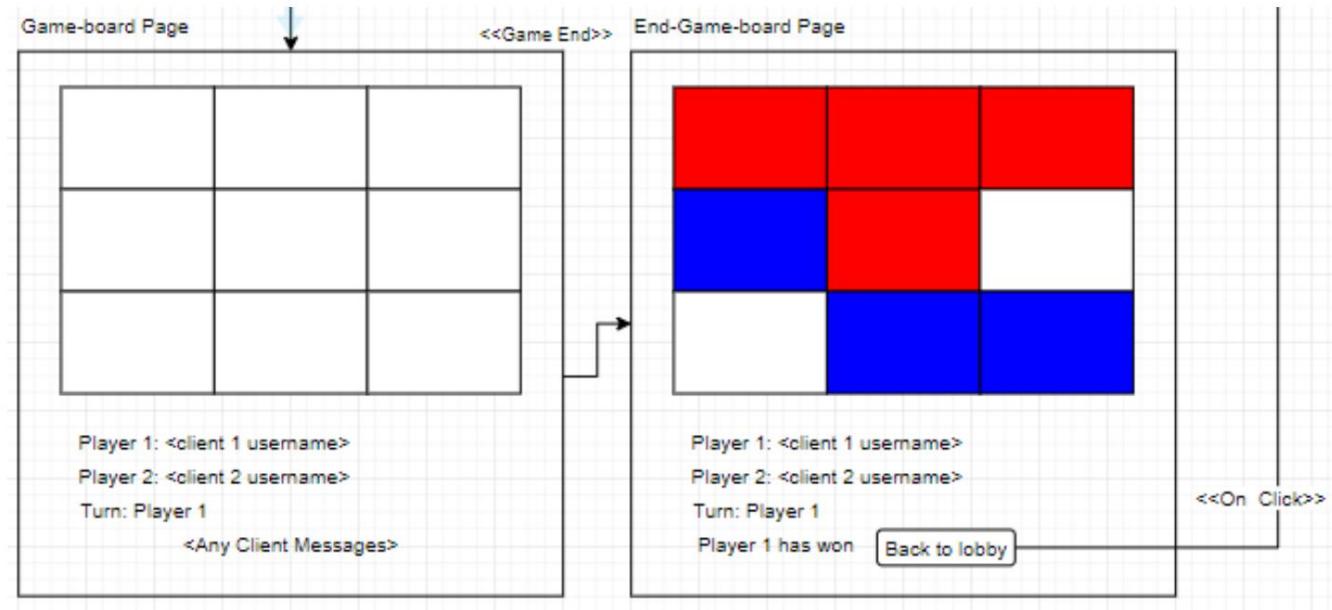
Có găng chọn các ô vuông đã được chọn cũng sẽ không có tác dụng gì. Ở cuối trang, sẽ có một phần cho thông tin của người chơi tiếp theo là lượt người chơi hiện tại và bất kỳ lỗi hoặc thông tin nào.

Khi trò chơi kết thúc, trang của họ sẽ hiển thị điều kiện kết thúc của trò chơi và một nút có nội dung "quay lại Trang chủ"

TRANG LÃNH ĐẠO-BẢN LÃNH ĐẠO

Ở phía bên tay trái, sẽ có một danh sách được sắp xếp của tất cả các tên tài khoản với số liệu thống kê của chúng theo thứ tự giảm dần về số lần thắng. Ở phía bên tay phải, sẽ có một nút để cập nhật dữ liệu ban lãnh đạo, tìm kiếm thứ hạng của tên người dùng cụ thể thông qua trường văn bản và nút và một cách nhanh chóng để tìm thứ hạng của bạn. Cũng sẽ có một nút cho phép người chơi quay lại Trang chủ. Ở cuối trang, sẽ có phần để người dùng nhận bất kỳ thông tin / lỗi nào.





CÁC LỚP UML

Tôi đã tạo một biểu đồ UML lớp cho ô cắm máy khách-máy chủ và ô cắm-máy chủ. Cả client-server-socket và socket-server sẽ nhận / gửi các đối tượng có dạng của lớp SocketActions . Một ví dụ về cách các Hành động sẽ được sử dụng sẽ có trong phần tiếp theo.

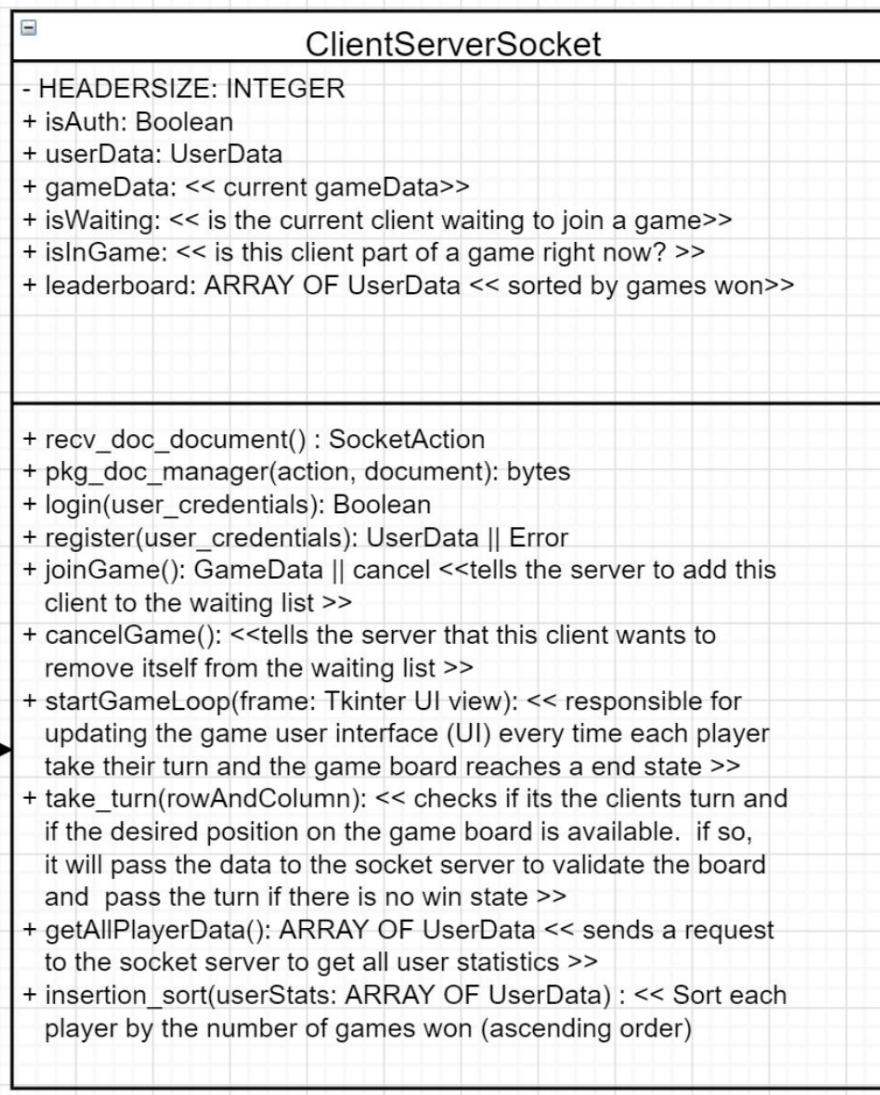
LỚP SOCKET SERVER

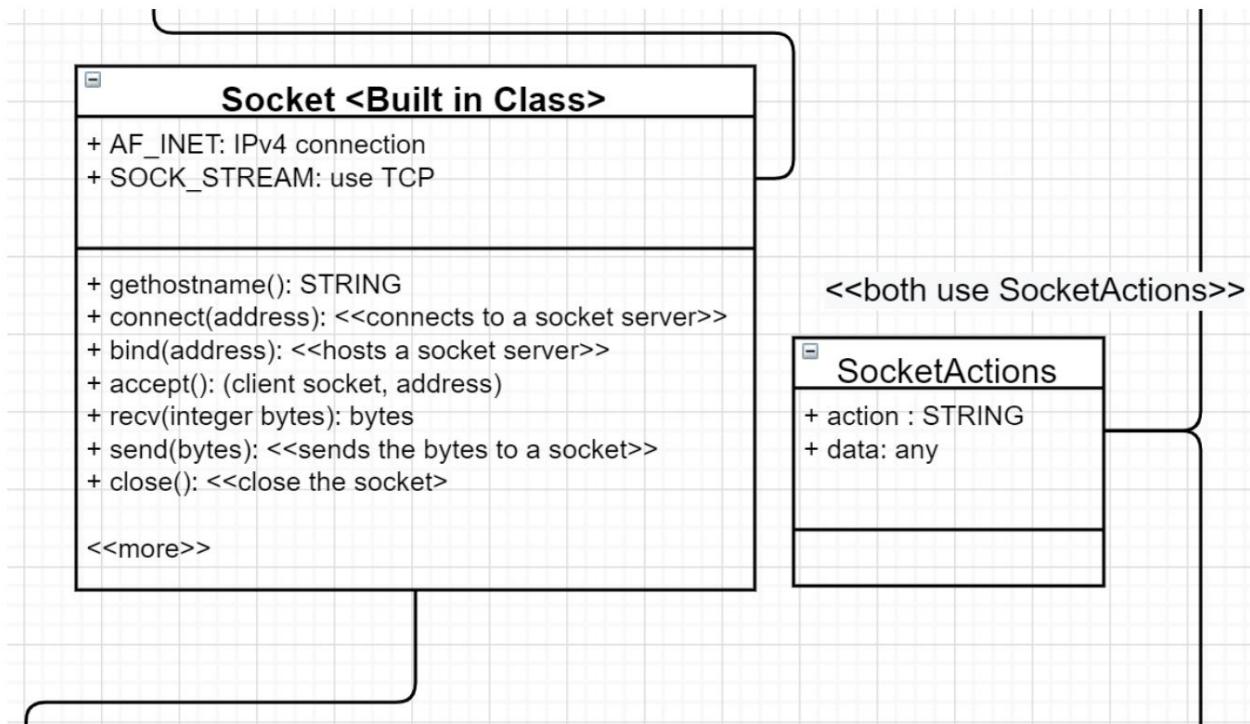
Máy chủ không được truy cập theo cách thủ công vì mục đích của nó là xác thực máy khách, thực hiện các hoạt động CRUD (Tạo, đọc, cập nhật và xóa) trên cơ sở dữ liệu SQLite và xử lý logic trò chơi, đó là lý do tại sao tất cả các thuộc tính và chức năng của nó là riêng tư và nó kế thừa Lớp socket trong python. Nó cũng sử dụng lớp SQL-Server Connection sẽ chỉ kết nối với cơ sở dữ liệu với thông tin xác thực hợp lệ, thiết lập bảng của cơ sở dữ liệu và cho phép lớp Socket-Server thực hiện các hoạt động CRUD trên cơ sở dữ liệu.

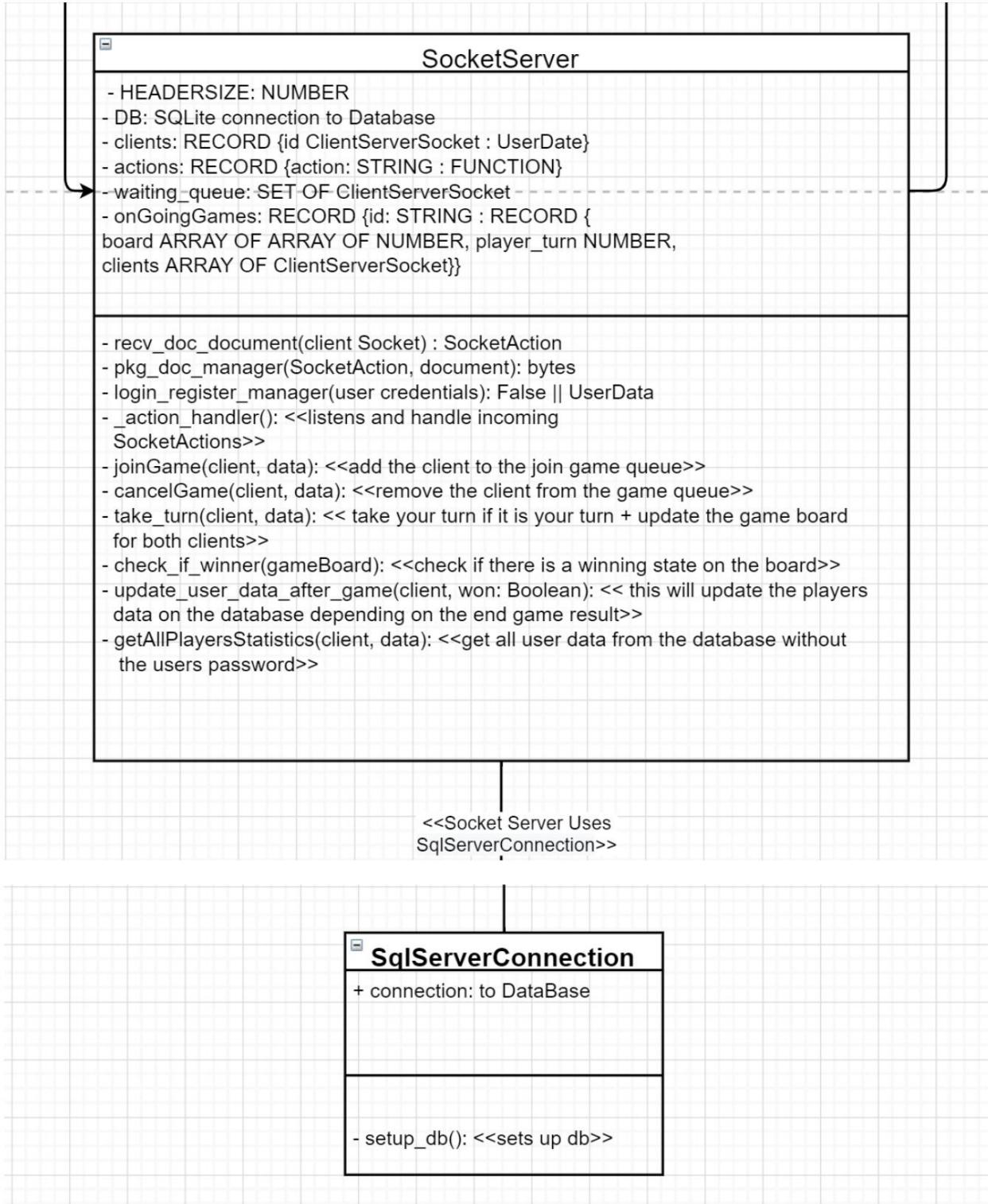
CLIENT SOCKET SERVER CLASS

Ô cắm máy khách sẽ chịu trách nhiệm cung cấp dữ liệu ứng dụng Tkinter, xác minh / xác thực, kết nối với các máy khách khác (để chơi trò chơi tick tack toe với bất kỳ ai khác được kết nối với máy chủ).

Vì lớp máy chủ Client Socket sẽ được ứng dụng front-end sử dụng nhiều, nên tôi đã thực hiện tất cả (ngoại trừ HEADERSIZE vì nó cần giống với máy chủ) các thuộc tính và chức năng của lớp máy chủ Client Socket thành công khai.







THỰC HIỆN / THIẾT KẾ - "HÀNH ĐỘNG" SẼ LÀ GÌ?

Để trả lời câu hỏi, tôi phải giải thích việc chọn một đối tượng trong Python nghĩa là gì. Pickling chỉ đơn giản là một cách để biến các đối tượng python thành byte. Vì tôi chỉ có thể gửi các byte giữa các ổ cắm, nên lấy dữ liệu là một cách dễ dàng để gửi các đối tượng hành động. Khi dữ liệu được nhận bởi một ổ cắm, tôi sẽ cần phải bỏ chọn dữ liệu để nhận đối tượng python.

Một bản ghi "hành động" đã chọn sẽ là thứ duy nhất được gửi đến và nhận từ máy chủ socket. Bằng cách xác định cấu trúc của bản ghi, sẽ làm cho dữ liệu được gửi và nhận từ máy chủ socket, có thể dự đoán được và dễ bảo trì hơn khi sử dụng dữ liệu chưa được chọn. Tôi lấy cảm hứng từ mẫu "Redux / NgRx", mà bản thân tôi thích sử dụng làm trình quản lý dữ liệu cho các ứng dụng web của mình. Khách hàng sẽ là "Người điều phối hành động" (có nghĩa là, nó sẽ là nguồn của tất cả các hành động). Máy chủ sẽ hoạt động giống như bộ giảm tốc sẽ chịu trách nhiệm kiểm soát trạng thái và phản ứng với các hành động được gửi đến nó và phản hồi bằng hành động của nó phản ánh kết quả của hành động đã nhận.

PSEUDOCODE: HÀNH ĐỘNG

Ghi lại SocketActions Là {STRING hành động, BẤT KỲ dữ liệu}

Giá trị hành động sẽ là một chuỗi mô tả ngắn gọn loại hành động đang được gửi. Mỗi hành động phải là duy nhất để không nên sử dụng hai loại hành động và dự kiến sẽ làm những việc khác nhau. Ví dụ: "[LOGIN USER]" nên được sử dụng để xử lý xác thực của một người dùng đã có và "[USER REGISTER]" chỉ xử lý các ứng dụng khách đang cố gắng tạo một tài khoản người dùng mới.

Giá trị dữ liệu có thể thay đổi tùy thuộc vào hành động đang được gửi, đó là lý do tại sao nó có một loại bất kỳ. Tôi sẽ phải làm cho máy khách gửi dữ liệu chính xác cho từng loại hành động nếu không máy chủ sẽ gửi lại một hành động lỗi với một chuỗi thông báo là dữ liệu.

CÁC LOẠI HÀNH ĐỘNG

Dưới đây là tất cả các hành động khả thi có thể được gửi giữa máy khách và máy chủ. Các tiêu đề là các loại hành động mà một ổ cắm máy khách (trình phát) có thể gửi đến máy chủ vì chúng là "Người điều phối hành động", mỗi loại hành động cũng có "Dữ liệu" dự kiến cần được chuyển cho nó. Máy chủ sẽ phản hồi với loại hành động theo sau là "THÀNH CÔNG" hoặc "THẤT BẠI" phản ánh kết quả của yêu cầu.

[ĐĂNG NHẬP NGƯỜI DÙNG]

Có gắng đăng nhập vào một tài khoản. Nếu thành công, máy chủ sẽ cho phép ổ cắm máy khách kết nối với máy chủ và nhận ra các hành động trong tương lai từ ổ cắm người gửi là người dùng đó cho đến khi kết nối bị đóng. Nếu không, máy chủ sẽ không chấp nhận kết nối của máy khách và phản hồi bằng một thông báo lỗi. Nếu có lỗi khi kết nối với cơ sở dữ liệu để tra cứu tên người dùng và so sánh mật khẩu đã băm hoặc xảy ra lỗi không mong muốn, máy chủ sẽ phản hồi bằng một hành động không thành công với thông báo lỗi là dữ liệu của nó.

Dữ liệu: (tên người dùng: STRING, mật khẩu: STRING)

Phản hồi:

- [NGƯỜI DÙNG ĐĂNG NHẬP - THÀNH CÔNG] - Dữ liệu: RECORD {kết quả: Đúng, dữ liệu: UserData} •
- [NGƯỜI DÙNG ĐĂNG NHẬP - FAIL] - Dữ liệu: RECORD {kết quả: Sai, dữ liệu "" Mật khẩu không chính xác "} HOẶC

RECORD {kết quả: Sai, dữ liệu: "Lỗi khi xác thực tài khoản của khách hàng."} HOẶC

RECORD {kết quả: Sai, dữ liệu: "Không tìm thấy người dùng nào có tên người dùng: (tên người dùng)"}

UserData = RECORD {tên người dùng: STRING, thắng: INTEGER, thua: INTEGER, games_played: INTEGER}

[ĐĂNG KÝ NGƯỜI DÙNG]

Có gắng đăng ký một tài khoản người dùng mới. Nếu thành công, máy chủ sẽ tạo một thực thể người dùng mới trong bảng của người dùng với tên người dùng và mật khẩu được băm mong muốn trong cơ sở dữ liệu SQLite, máy khách có thể sử dụng cùng thông tin đăng nhập để xác thực khi nó gửi hành động đăng nhập đến máy chủ. Nếu tên người dùng đã được đăng ký hoặc máy chủ không thể kết nối với cơ sở dữ liệu, máy chủ sẽ phản hồi bằng một hành động không thành công với thông báo lỗi là dữ liệu của nó.

Dữ liệu: (tên người dùng: STRING, mật khẩu: STRING)

Phản hồi:

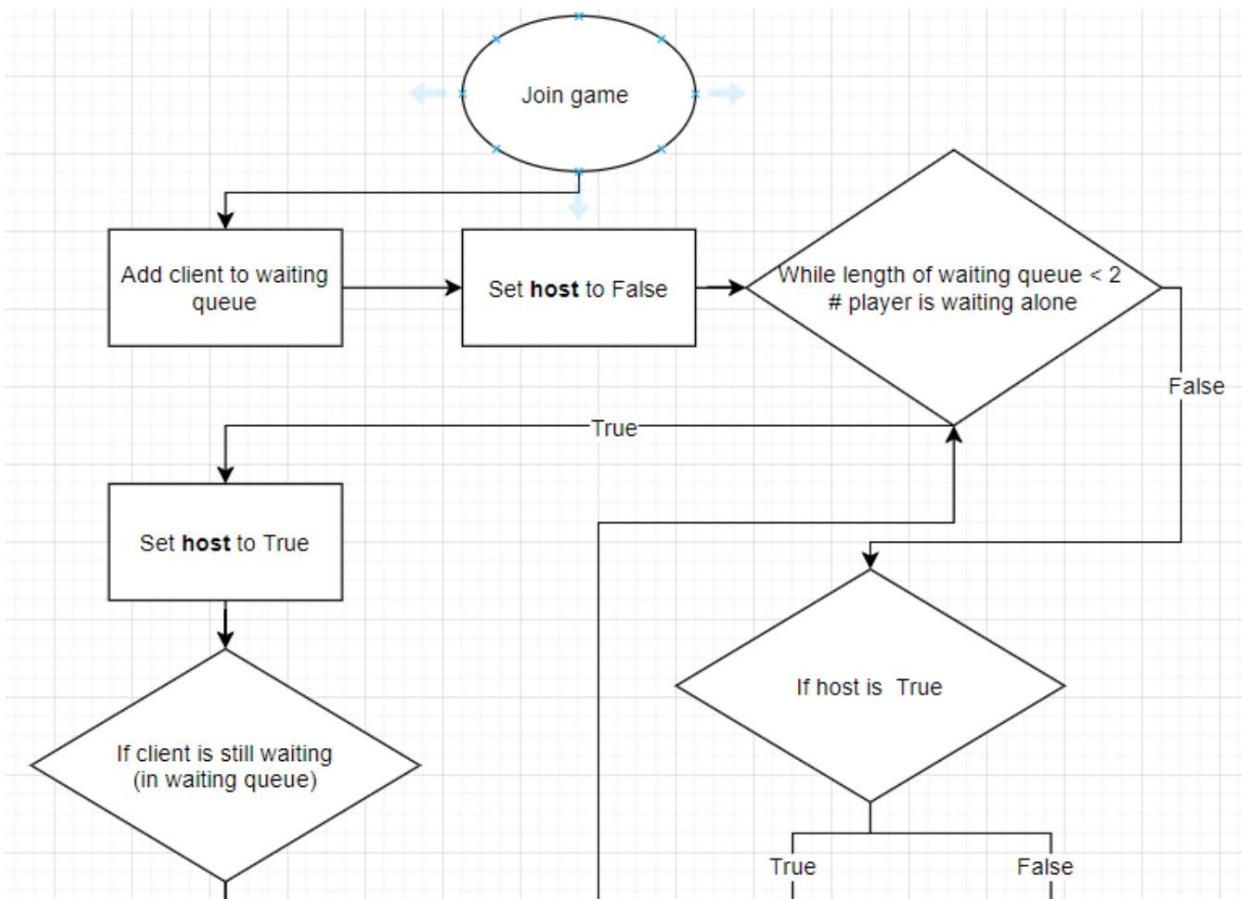
- [ĐĂNG KÝ NGƯỜI DÙNG - THÀNH CÔNG] - Dữ liệu: RECORD {kết quả: Đúng, thông báo: "Đã tạo tài khoản thành công. "}
- [ĐĂNG KÝ NGƯỜI DÙNG - THẤT BẠI] - Dữ liệu: RECORD {kết quả: Sai, thông báo: "Tên người dùng đã tồn tại."} HOẶC

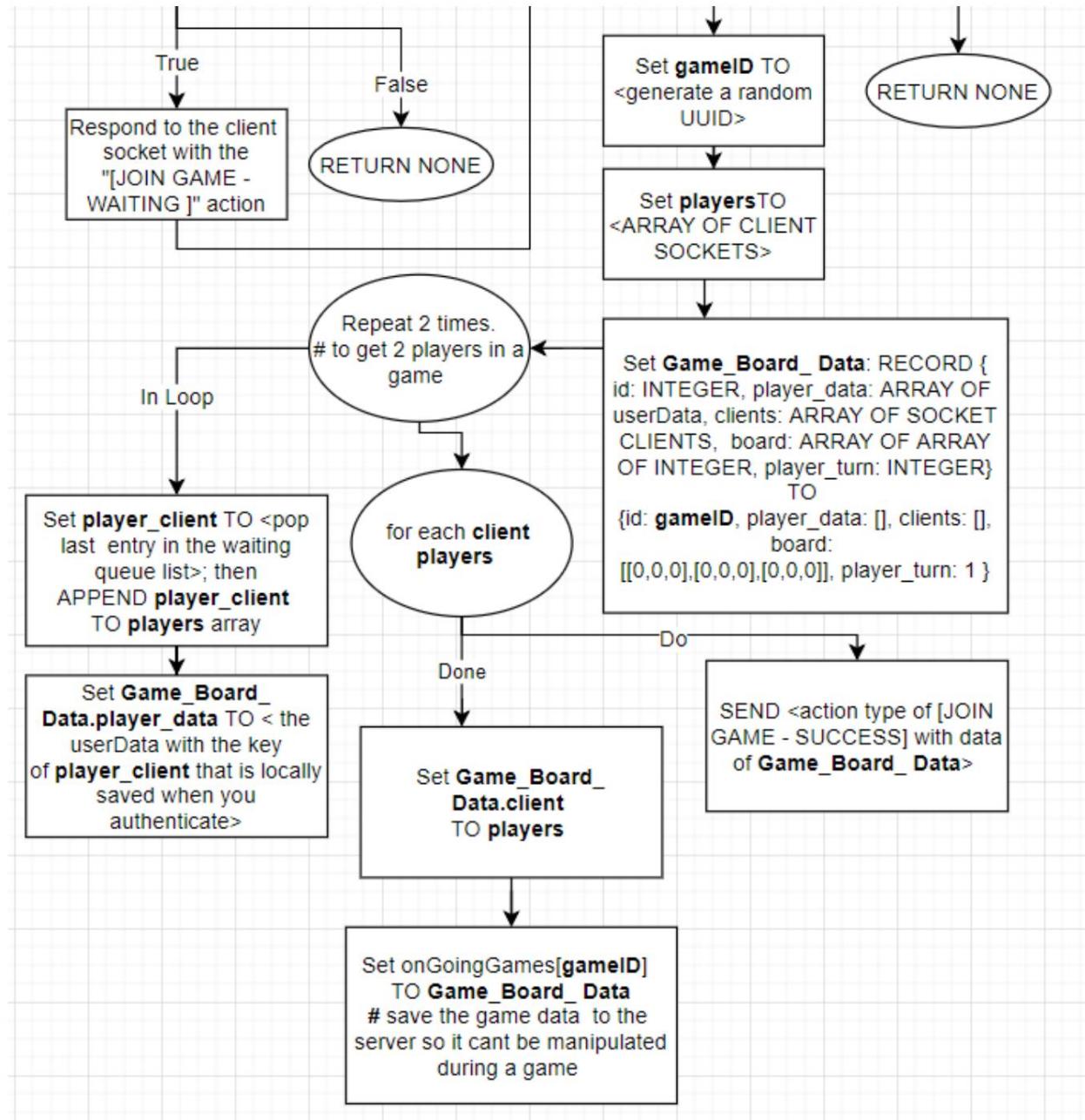
RECORD {kết quả: Sai, thông báo: "Lỗi khi xác thực tài khoản của khách hàng."}

[THAM GIA TRÒ CHƠI]

Khách hàng đang cố gắng tham gia một trò chơi. Nếu thành công, máy chủ sẽ thêm dữ liệu trò chơi mới vào bản ghi onGoingGames với một gameID được tạo ngẫu nhiên làm khóa và Game_Board_Data làm giá trị của nó.

Cả hai người chơi sẽ nhận được một hành động thành công khi cả hai đều ở trong một trò chơi với Game_Board_Data làm dữ liệu của nó. Nếu người chơi đang đợi một mình trong hàng đợi, yêu cầu của họ sẽ trở thành máy chủ (điều này để tôi không bị trùng lặp các trò chơi đang bắt đầu bởi cả hai người chơi trong hàng khi họ đang bắt cặp), chỉ có yêu cầu máy chủ mới có thể tạo một phiên trò chơi mới và thông báo cho cả hai người chơi rằng họ đang tham gia một trò chơi. Trong khi chờ đợi, máy chủ sẽ kiểm tra xem yêu cầu máy chủ có còn đang chờ trong hàng đợi hay không và sẽ đợi người chơi thứ hai tham gia hàng đợi trên một chuỗi mới để chương trình không bị chặn. (cho phép hành động của người chơi khác được xử lý trên chuỗi chính mà không cần mã trước đó để kết thúc). Có một sơ đồ bên dưới minh họa cách máy chủ sẽ xử lý hành động "[tham gia Trò chơi]".





```

Game_Board_Data = RECORD {id: INTEGER, player_data: ARRAY OF userData, client: ARRAY OF SOCKET
CLIENTS, board: ARRAY OF ARRAY OF INTEGER, player_turn: INTEGER}

```

Dữ liệu: (tên người dùng: STRING) HOẶC BẤT KỲ

Phản hồi:

- [THAM GIA TRÒ CHƠI - THÀNH CÔNG] - Dữ liệu: Game_Board_Data <không có thuộc tính của khách hàng>
- [THAM GIA TRÒ CHƠI - ĐANG CHƠI] - Dữ liệu: (tên người dùng: STRING) HOẶC BẤT KỲ

[HỦY TRÒ CHƠI]

Điều này sẽ cho máy chủ biết rằng người chơi không còn muốn tham gia trò chơi nữa. Nếu máy khách nằm trong Hàng đợi, máy chủ sẽ xóa máy khách khỏi danh sách và phản hồi bằng một hành động thành công. Nếu không, máy chủ sẽ phản hồi với một hành động không thành công.

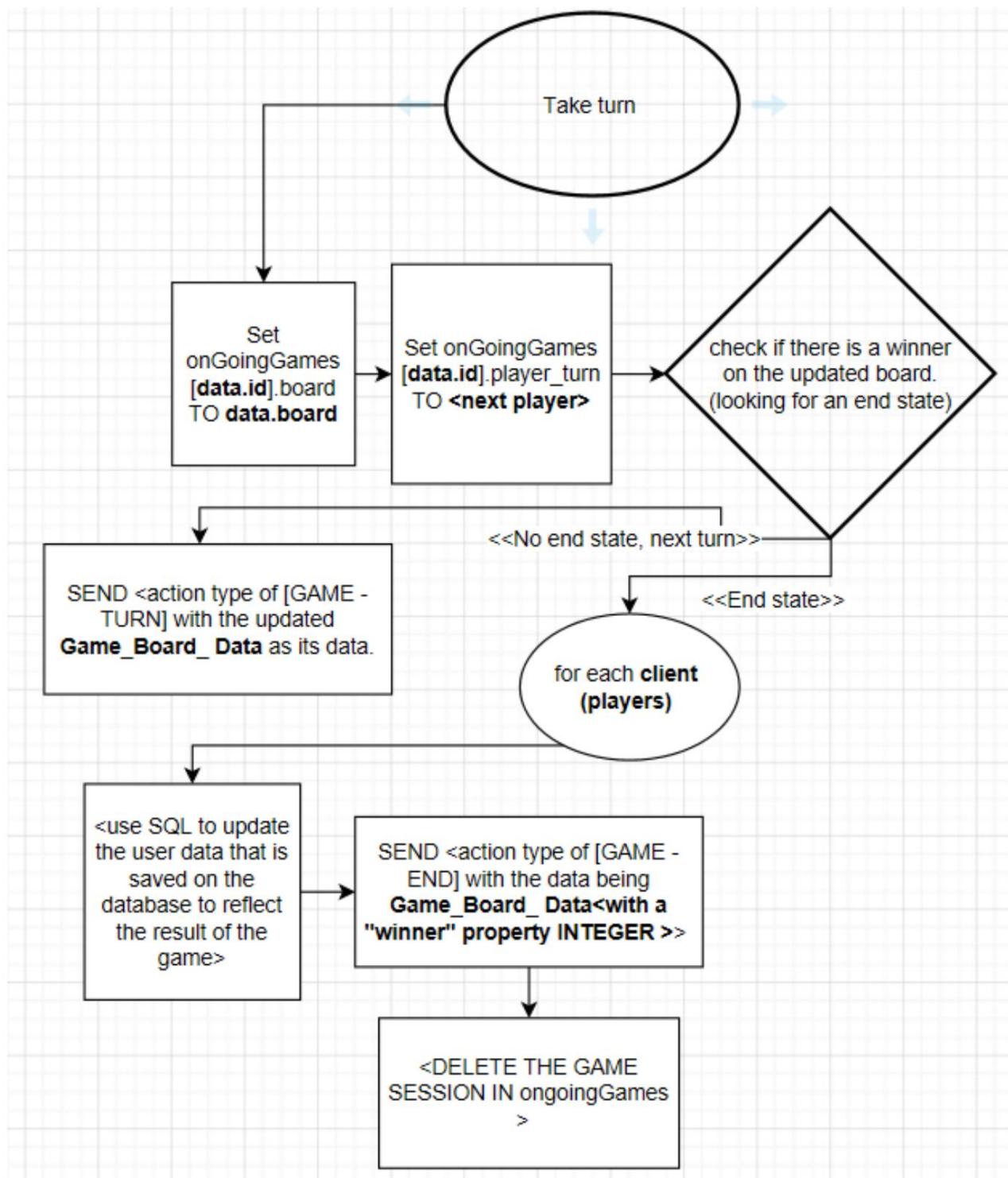
Dữ liệu: (tên người dùng: STRING) HOẶC BẤT KỲ

Phản hồi:

- [HỦY TRÒ CHƠI - THÀNH CÔNG] - Dữ liệu: "Đã hủy"
- [HỦY TRÒ CHƠI - THẤT BẠI] - Dữ liệu: "Không đợi trò chơi"

[THAY PHIÊN NHAU]

Hành động này sẽ được gửi đến máy chủ mỗi khi người chơi chọn một ô vuông hợp lệ trên bảng trò chơi. Sau đó, máy chủ sẽ kiểm tra xem có trạng thái thắng trên bàn cờ (3 màu giống nhau liền kề nhau hoặc trên một đường chéo) hay trạng thái hòa (không có người chiến thắng và không thể chơi hành động nào). Nếu bất kỳ trạng thái nào trong số này là đúng thì máy chủ sẽ cập nhật từng dữ liệu người dùng tương ứng và kết thúc phiên trò chơi với kiểu hành động cuối trò chơi. Tuy nhiên, nếu trò chơi chưa kết thúc và vẫn có thể chơi được, máy chủ sẽ phản hồi bằng kiểu hành động theo lượt với bảng trò chơi được cập nhật và lượt người chơi tiếp theo (ví dụ: nếu người chơi một vừa chơi, bây giờ người chơi hai có thể tham gia một lượt) cho cả hai. Người chơi để họ có thể cập nhật bảng của họ. Một số dòng được cung cấp bên dưới minh họa cách điều này sẽ hoạt động trong mã.



Dữ liệu: Game_Board_Data <bảng trò chơi được cập nhật, không có thuộc tính ứng dụng khách>

Phản hồi:

- [GAME - END] - Dữ liệu: RECORD {người chiến thắng: INTEGER, updated(userData: UserData)}
 - [GAME - TURN] - Dữ liệu: Game_Board_Data <với bảng trò chơi được cập nhật và lượt người chơi.
- Không có tài sản của khách hàng>
-

[NHẬN TẮT CẢ THÔNG KÊ NGƯỜI CHƠI]

Khách hàng sẽ muốn tất cả số liệu thống kê của người chơi được đưa vào bảng xếp hạng. Máy chủ sẽ truy vấn cơ sở dữ liệu cho bảng Người dùng để lấy tất cả thống kê người chơi (loại bỏ mật khẩu đã băm). Nếu thành công, máy chủ sẽ phản hồi với một loại hành động thành công với dữ liệu là một ARRAY của UserData. Nếu có lỗi, máy chủ sẽ phản hồi bằng loại hành động không thành công với thông báo lỗi là dữ liệu của nó.

Dữ liệu: (tên người dùng: STRING) HOẶC BẤT KỲ

Phản hồi:

- [NHẬN TẮT CẢ THÔNG KÊ NGƯỜI CHƠI - THÀNH CÔNG] - Dữ liệu: ARRAY of UserData
 - [NHẬN TẮT CẢ THÔNG KÊ NGƯỜI CHƠI - THẤT BẠI] - Dữ liệu: "Lỗi khi nhận tất cả số liệu thống kê về người chơi"
 -
-

VÍ DỤ SỬ DỤNG: PSEUDOCODE CHO NGƯỜI DÙNG XÁC THỰC (OOP)

Đây sẽ là cách máy khách cố gắng đăng nhập (xác thực) bằng cách sử dụng ổ cắm máy khách và gửi biểu diễn byte của hành động “[LOGIN USER]” với tên người dùng và mật khẩu của máy khách tới máy chủ ổ cắm. Máy chủ sẽ phản hồi với hành động của nó theo byte cần được bỏ chọn, điều này sẽ cho phép khách hàng biết liệu yêu cầu của họ đã thành công hay đã gặp lỗi.

MÃ SỐ PSEUDOCODE ĐĂNG NHẬP BẰNG CỦA KHÁCH HÀNG

```
FUNCTION login_client (ĐỀN TRONG STRINGS user_authentication_data) QUAY LẠI BOOLEAN

    # định dạng cho user_Authentication_Data sẽ là một bộ mã tràn

    # được truy cập giống như một mảng, do đó lý do tôi truyền kiểu dữ liệu là ARRAY

    # TRONG STRINGS gây ra sự khác biệt duy nhất là giá trị tuple là

    # Immutable (không thể thay đổi). ví dụ ("tên người dùng", "mật khẩu")

    # BIẾN SỐ KHAI THÁC

    DECLARE userAuthData AS SocketActions BAN ĐẦU KHÔNG ĐỦ

    DECLARE packagedUserAuthData AS STRING BAN ĐẦU ''

    DECLARE serverResponseData AS SocketActions BAN ĐẦU KHÔNG ĐỦ

    NẾU user_authentication_data KHÔNG HỢP LỆ THÌ
```

GỬI "INVALID" & user_authentication_data ĐỂ HIỂN THỊ

QUAY LẠI Sai

ĐẶT userAuthData VÀO SocketActions ("[LOGIN USER]", user_authentication_data)

ĐẶT packagedUserAuthData TÓM TẮT <parse (pickle) userAuthData TÓM TẮT BYTES để nó có thể được gửi đến máy chủ>

GỬI packagedUserAuthData ĐẾN <socket server>

RECEIVE serverResponseData TÓM TẮT <đợi máy chủ phản hồi, sau đó bỏ phân tích (bỏ chọn) phản hồi BYTES TÓM TẮT một SocketActions (dữ liệu)> <socket server>

IF serverResponseData.action == "[ĐĂNG NHẬP NGƯỜI DÙNG THÀNH CÔNG]" THÌ

ĐẶT NÀY.userData CHO serverResponseData.data

TRẢ LẠI Đúng

KHÁC

QUAY LẠI Sai

KẾT THÚC NẾU

CHỨC NĂNG KẾT THÚC

MÃ PSEUDOCODE ĐĂNG NHẬP BÊN MÁY CHỦ

SET client_action_document TÓM TẮT <Đã nhận hành động "[LOGIN USER]" được chọn từ một ô cấm ứng dụng chưa được xác thực và bỏ chọn thành công các byte thành đối tượng "Hành động" trong python>

NẾU client_action_document.action == "[ĐĂNG NHẬP NGƯỜI DÙNG]" THÌ

ĐẶT người dùng ĐỂ login_manager (client_action_document.data)

IF user.result == False thì

đã xảy ra lỗi hoặc tên người dùng / mật khẩu không chính xác. Gửi thông báo lỗi cho khách hàng

<phản hồi lại ô cấm ứng dụng khách với loại hành động "[NGƯỜI DÙNG ĐĂNG NHẬP - THẤT BẠI]" và dữ liệu là giá trị của user.data>

Khác

thông tin đăng nhập được xác thực thành công

<Phản hồi với ô cấm ứng dụng khách với loại hành động "[NGƯỜI DÙNG ĐĂNG NHẬP - THÀNH CÔNG]" và dữ liệu là giá trị của user.data>

<thêm ô cảm ứng khách vào danh sách các máy khách đã được xác thực dưới dạng GHI với dữ liệu người dùng từ cơ sở dữ liệu là giá trị của user.data>

hiển thị thông báo nói rằng một người dùng mới được xác thực

GỬI "Kết nối mới được chấp nhận từ:" & user.data.username ĐẾ HIỂN THỊ

KẾT THÚC NẾU

KẾT THÚC NẾU

FUNCTION login_manager (ARRAY OF STRINGS userCredentials

) QUAY LẠI BẢN GHI {kết quả: BOOLEAN, dữ liệu: STRING hoặc userData}

THỦ:

ĐẶT userCredentials_fromDB T0 <sử dụng lớp SqlServerConnection để truy vấn cơ sở dữ liệu cho tên người dùng và mật khẩu được băm của nơi username = userCredentials [0] (thuộc là tên người dùng đang cố gắng xác thực)>

NẾU userCredentials_fromDB LÀ KHÔNG CÓ

không có người dùng nào có tên người dùng đó trong cơ sở dữ liệu

RETURN RECORD {result: FALSE, data: "Không tìm thấy người dùng nào có tên người dùng:" & (userCredentials [0])}

KẾT THÚC NẾU

SET hashPassword T0 <MD5 hash userCredentials [1] (đây sẽ là mật khẩu được cung cấp cho đăng nhập)>

NẾU hashPassword == userCredentials_fromDB.password THEN

ô cảm ứng khách có thể được chấp nhận và xác thực

SET userData T0 <sử dụng lớp SqlServerConnection để truy vấn cơ sở dữ liệu về tên người dùng, thắng, thua và số trò chơi đã chơi trong đó username = userCredentials [0]>

QUAY LẠI BẢN GHI {kết quả: TRUE, data: userData}

KẾT THÚC NẾU

CHỤP LẤY:

CÓ LỖI KHI HIỆN MÃ TRÊN vì chương trình không thể truy vấn cơ sở dữ liệu hoặc có điều gì đó không mong muốn đã xảy ra.

RETURN RECORD {result: FALSE, data: "Lỗi khi xác thực tài khoản của khách hàng."}

HẾT THỨ

CHỨC NĂNG KẾT THÚC

THỰC HIỆN

Trong phần này của tài liệu, tôi sẽ xem xét các phần khác nhau của chương trình mà tôi đã phát triển, các khái niệm tôi phải học, tích hợp và một số lý do tại sao tôi lập trình các phần theo cách tôi đã làm. Ứng dụng này là tất cả các đối tượng được định hướng, vì vậy tôi sẽ cố gắng giữ tất cả các thực lề để giữ cho mọi thứ dễ theo dõi.

NGHIÊN CỨU

Ô CẮM PYTHON & ĐỐI TƯỢNG PICKLING PYTHON

Các ô cắm cho phép tôi kết nối 2 nút trên một mạng. Tôi cần sử dụng ô cắm để tạo kết nối giữa người chơi và máy chủ. Sau đó, điều này sẽ cho phép tôi cập nhật cả hai người chơi trong cùng một phiên trò chơi và cho phép họ đấu với nhau trên các máy khách (và máy tính) khác nhau.

Các trang web tôi đã sử dụng để tìm hiểu mọi thứ mà tôi cần biết là từ (<https://pythonprogramming.net/sockets-tutorial-python-3/>) đã cung cấp video và hướng dẫn bằng văn bản về cách tạo ứng dụng đầu tiên của bạn (ứng dụng trò chuyện) với ô cắm python. Tôi đã sử dụng thông tin và các ví dụ từ trang web này để tạo ra trò chơi tick tack toe trực tuyến này.

TẠO MÁY CHỦ Ô CẮM VÀ KHÁCH HÀNG

Hướng dẫn này không tận dụng lợi thế của lập trình hướng đối tượng như tôi đã làm, tuy nhiên, khái niệm vẫn là tương tự.

THIẾT LẬP Ô CẮM MÁY CHỦ

```
import socket # used to run the socket server
import select # used to manage concurrent connections to the server socket

class SocketServer(socket.socket):
    def __init__(self):
        super().__init__(socket.AF_INET, socket.SOCK_STREAM)
        # socket.AF_INET is saying our socket host's IP is going to be a IPv4 (Internet Protocol version 4)
        # socket.SOCK_STREAM is saying that the port that the socket will be using is a TCP (Transmission Control Protocol)
        # gets the current host ip https://www.whatismyip.com/
        socketHostData = (socket.gethostname(), 4201)
        # binds the socket server to the current host name and listens to active connections
        self.bind(socketHostData)
        print(
            f"server started on Host: {socketHostData[0]}, Port: {socketHostData[1]}")

        # set max connection to 6
        self.listen(6)
        self.sockets_list = [self]
```

Ps, khi lưu trữ trên một máy chủ, bạn cần đặt socketHostData TO ("", PORT). Điều này cho máy chủ kết nối với bất kỳ IP nào có sẵn (IP máy chủ). Điều này cho phép kết nối từ các máy tính khác nhau trên các mạng khác nhau để kết nối với máy chủ.

self._action_handler()

Máy chủ tự liên kết với địa chỉ máy chủ và lắng nghe mọi dữ liệu đến từ luồng của nó. Khi máy chủ nhận được bất kỳ dữ liệu hoặc yêu cầu kết nối socket nào chúng đều đi qua hàm _action_handler .

```
def _action_handler(self):
    while True:
        # defines our read, write and errored listed sockets
        read_sockets, _write, error_sockets = select.select(
            self.sockets_list, [], self.sockets_list)

        for user_socket in read_sockets:
            if user_socket == self:
                """
                These will be the client sockets trying to connect to the server socket.
                So in this block i will be have to
                """

                # accept connections from server, this is so i can read the package
                client_socket, client_address = self.accept()
                print( client_socket, client_address)

                # handle the incoming document action
                client_action_document = self.recv_doc_manager(
                    client_socket)

                if client_action_document is False:
                    # disconnection, left the socket
                    continue
```

Ví dụ sử dụng thư viện lựa chọn chỉ là một cách hiệu quả để lặp lại và giám sát nhiều kết nối và yêu cầu tới máy chủ trên quy mô lớn. Tôi cũng đã sử dụng thư viện chọn vì nó cho tôi cấu trúc và luồng rõ ràng để xử lý các yêu cầu gửi đến máy chủ. Tôi tạo một vòng lặp không xác định sẽ xử lý bất kỳ dữ liệu nào đến ở cắm máy chủ. Dữ liệu đến sẽ được gửi bởi máy khách đã xác thực hoặc máy khách chưa được xác thực (tôi nói về quy trình này trong phần "xử lý máy khách chưa được xác thực"). Đó là lý do tại sao tôi kiểm tra xem user_socket có bằng với ổ cắm máy chủ hay không nếu máy khách chưa được xác thực và sẽ bị giới hạn để cố gắng đăng nhập hoặc tạo mới tài khoản người dùng.

```

else:
    """
    At this point the connected client socket has already been authenticated
    """
    client_action_document = self.recv_doc_manager(
        user_socket)
    if client_action_document is False:
        # disconnection, left the socket
        print(f'Closed connection from User:{self.clients[user_socket][0]}')
        self.sockets_list.remove(user_socket)
        del self.clients[user_socket]
        continue

    # handle any other action being passed to the server
    print(f'{client_action_document["action"]}: UserName: {self.clients[user_socket][0]}')
    action = self.actions[client_action_document["action"]]
    if action is False:
        # this happens when the action type sent to the server isn't known
        user_socket.send(self.pkg_doc_manager(
            "[ERROR - ACTION]", f"Unregistered action type {client_action_document['action']}"))
        continue

    # start a new thread so that the action that is being sent doesn't halt reading other client messages
    _thread.start_new_thread(
        action, (user_socket, client_action_document["data"]))
    continue

```

Nếu user_socket nằm trong self.sockets_list , điều đó có nghĩa là họ là một ứng dụng khách đã được xác thực và được phép sử dụng các chức năng khác. Khi một hành động được gửi bởi một ô cắm máy khách đã xác thực, máy chủ sẽ thực hiện những việc sau:

- Kiểm tra xem đó có phải là một yêu cầu kết nối đóng không (vì vậy tôi có thể xóa kết nối của người chơi đó khỏi máy chủ). Điều này sẽ giải phóng dung lượng và làm cho ứng dụng trở nên mạnh mẽ và có thể mở rộng.
- Chuyển dữ liệu đến vào một đối tượng hành động (tôi nói về điều này trong phần “Đóng gói và rút gọn hành động”)
- Kiểm tra xem kiểu hành động đã được gửi có được xác định trong trình tạo hay không. Điều này chỉ làm cho mã nhiều hơn mạnh mẽ vì máy chủ sẽ không xử lý bất kỳ loại hành động không xác định nào.

```

# Actions that authenticated users can call
self.actions = {
    "[JOIN GAME]": self.joinGame,
    "[CANCEL GAME)": self.cancelGame,
    "[TAKE TURN]": self.takeTurn,
    "[GET ALL PLAYER STATS]": self.getAllPlayerStats
}

```

- Ghi nhận ký hành động đã được gửi và người dùng đã gửi hành động. Điều này rất hữu ích khi gỡ lỗi những gì được gửi bởi máy khách.
- Nếu loại hành động được gửi được xác định thì máy chủ sẽ xử lý hành động trên một chuỗi mới (tôi nói về “_Thread” ở phần sau của phần nghiên cứu)

THIẾT LẬP Ô CẨM KHÁCH HÀNG

```
import socket # used to create the client socket connection with the server socket.
```

```
class ClientServerSocket(socket.socket):
    # default the host will be "socket.gethostname()" which in the current computer.
    # socketHostData will have to be passed if the server isn't running on the current computer.
    def __init__(self, socketHostData=(socket.gethostname(), 4201)):
        super().__init__(socket.AF_INET, socket.SOCK_STREAM)
        # socket.AF_INET is saying our socket host's IP is going to be a IPv4 (Internet Protocal version 4)
        # socket.SOCK_STREAM is saying that the port that the socket will be using is a TCP (Transmission Control Protocol)
    try:
        self.connect(socketHostData)
    except BaseException as e:
        print(e)
        raise ConnectionError(f"Could not connect to the HostName: {socketHostData[0]}, Port: {socketHostData[1]} - It may not exist or be down.")
```

Sau đó máy khách cần kết nối với máy chủ ổ cắm. Ổ cắm máy khách không được chấp nhận ngay lập tức khi chúng kết nối, chúng sẽ cần gửi hành động “[ĐĂNG NHẬP NGƯỜI DÙNG]” như tôi đã nói trong phần thiết kế.

GỬI DỮ LIỆU GIỮA CÁC Ô CẮM

Tôi đã sử dụng phương pháp đệm được hiển thị trong các ví dụ <https://pythonprogramming.net/pickle-objects-sockets-tutorial-python-3/?Complete=/buffering-streaming-data-sockets-tutorial-python-3/>. đã cho phép tôi gửi độ dài luồng động. Thông thường, bạn sẽ có kích thước bộ đệm cố định là số byte sẽ được đọc, vì tôi sẽ gửi các đối tượng hành động động, đây là một giải pháp hoàn hảo để làm điều đó.

```
>>> import pickle
>>> action = {"action": "[USER LOGIN]", "data": ("test1","test1Password")}
>>> # now i will pickle the object (turn it into bytes)
>>> pickledAction = pickle.dumps(action)
>>> # i will take a look of the pickleAction value
>>> print(pickledAction)
b'\x80\x03}q\x00(X\x06\x00\x00\x00actionq\x01X\x0c\x00\x00\x00[USER LOGIN]q\x02X\x04\x
00\x00\x00dataq\x03X\x05\x00\x00\x00test1q\x04X\r\x00\x00\x00test1Passwordq\x05\x86q\x
06u.'
>>> # as you can see the object is now in bytes
>>> # now i will add the header so i can tell the receiving node
>>> # the length of the action data. this will allow them to read
>>> # the next buffer from the stream
>>> HEADERSIZE = 10
>>> packaged_action = bytes(f"{len(pickledAction):<(HEADERSIZE)}", 'utf-8')+pickledAct
ion
>>> # i will take a look of the formating of the header (first 10 bytes)
>>> print(packaged_action)
b'86      \x80\x03}q\x00(X\x06\x00\x00\x00actionq\x01X\x0c\x00\x00\x00[USER LOGIN]q\x
02X\x04\x00\x00\x00dataq\x03X\x05\x00\x00\x00test1q\x04X\r\x00\x00\x00test1Passwordq\x
05\x86q\x06u.'
```

Sau đó, tôi gửi biến `packaged_action` qua socket. Sau khi truy xuất các byte hành động từ luồng. Tôi sẽ tải dữ liệu bằng cách làm như sau

```
>>> depackage_data = pickle.loads(pickledAction)
>>> print(depackage_data)
{'action': '[USER LOGIN]', 'data': ('test1', 'test1Password')}
>>> print(depackage_data["action"])
[USER LOGIN]
```

Như bạn có thể thấy, đối tượng action hoàn toàn có thể được sử dụng bởi nút nhận. Tôi sẽ sử dụng phương pháp này để gửi dữ liệu giữa các ô cấm.

LẬP TRÌNH HIỆN NAY TRONG PYTHON

Lập trình đồng thời về bản chất là khả năng thực hiện nhiều hơn một lệnh cùng một lúc. Tôi đã có một số kinh nghiệm trong việc viết đồng thời vì vậy tôi sẽ chỉ chuyển sang các ví dụ về trường hợp sử dụng.

ASYNCIO

Khi người chơi gửi một hành động đến máy chủ, có thể mất một khoảng thời gian để máy chủ phản hồi. Máy khách có thể cần đợi cho đến khi nhận được phản hồi trước khi chuyển sang dòng mã tiếp theo. Đây là lúc chúng ta có thể sử dụng asyncio để yêu cầu chương trình đợi (chờ đợi) để một chức năng hoàn thành trước khi chuyển sang dòng mã tiếp theo. Điều này cho phép chúng ta kiểm soát luồng thực thi lệnh. Bất cứ khi nào một ô cấm máy khách gửi một hành động đến máy chủ, nó sẽ cần phải chờ phản hồi của máy chủ.

```
async def getAllPlayerData(self):
    # make sure the client is authenticated and not in a game
    try:
        if self.isAuth is True and self.isInGame is False:
            packaged_get_player_all_data_action_document = self.pkg_doc_manager(
                "[GET ALL PLAYER STATS]", self.userData[0])
            self.send(packaged_get_player_all_data_action_document)
            results = await self.recv_doc_manager()

        # nothing was sent back, something broke on the server (disconnected)
        if results is None:
            return "Error: no connection to the socket"

        if results["action"] == "[GET ALL PLAYER STATS - FAIL]":
            # failed to get user statistics
            self.leaderboard = None
            return results["data"]

        # successfully get user statistics
        self.leaderboard = self.insertion_sort(results["data"]) # sort the player data
        return True
    except:
        raise
```

Chúng tôi chỉ có thể sử dụng từ khóa await bên trong một hàm Không đồng bộ (Async) , mà chúng tôi có thể thực hiện bằng cách thêm không đồng bộ trước từ khóa def khi định nghĩa một hàm. Như bạn có thể thấy, tôi đang chờ hàm rec_doc_manager vì

máy chủ cần thời gian để phản hồi. Làm như vậy sẽ luôn có một số dữ liệu trong biến kết quả vì nó được dựa vào các dòng tiếp theo.

```
"""
lets the program run async-ly (controls whether or not to wait
for a process to finish before moving on to the next or leave it
to run in the background)
"""

import asyncio
```

```
def render(self):
    if self.controller.SocketConnection != None:
        # username will be used to lookup a single user's ranking
        username = tk.StringVar()
        username.set(self.controller.SocketConnection.userData[0])

    asyncio.run(self.get_all_userData_sorted())
```

Khi kết xuất bảng xếp hạng, tôi cần lấy tất cả dữ liệu thống kê người dùng trước khi đặt tiện ích ra màn hình. Đó là lý do tại sao tôi gọi một hàm không đồng bộ có tên get_all_userData_sorted sẽ thực hiện điều này trước khi hiển thị các widget. Bạn chỉ có thể gọi các hàm không đồng bộ trong một chương trình đăng quang có thể được tạo bằng hàm `async.run` như tôi đã làm ở trên.

```
async def get_all_userData_sorted(self):
    """
    get all user datas from the database via the server connection
    """

    # get all user stats data from the server (username, wins, loses, games_played)
    # fetch_all_user_stats = [("isaac", 12, 0, 12), ("test1", 0, 10, 10), ("test2", 4, 10, 14), ("test3", 10, 4, 14)]
    try:
        self.MSG.set("")
        fetch_all_user_stats = await self.controller.SocketConnection.getAllPlayerData()
        if fetch_all_user_stats != True:
            self.MSG.set(fetch_all_user_stats)
            self.userData = self.controller.SocketConnection.leaderboard
    except:
        raise
```

Bên trong hàm `get_all_userData_sorted`, chúng ta đang chờ kết quả của hàm `getAllPlayerData` mà chúng ta đã xem ở trên. Chúng tôi chờ nó kết thúc vì dòng tiếp theo sau đó sẽ kiểm tra xem chúng tôi đã truy xuất thành công dữ liệu từ máy chủ hay chưa. Nếu chúng tôi không chờ đợi hàm `getAllPlayerData`, nó sẽ hoàn thành ở chế độ nền, nhưng chương trình đã thực hiện kiểm tra và kết luận sai vào thời điểm máy chủ phản hồi và dữ liệu đã được sắp xếp.

CHÚ Ý

Phản luồng cho phép tôi tạo một quy trình khác để thực hiện một tác vụ đồng thời. Tôi đã sử dụng `_thread` trên máy chủ khi xử lý một hành động hợp lệ. Điều này là do tôi không muốn một hành động nào chặn các yêu cầu khác được thực hiện. Ví dụ: không xử lý yêu cầu từ chuỗi chính - nếu một người chơi đang đợi để tham gia trò chơi, chuỗi chính của máy chủ sẽ bị chặn vì nó sẽ bị mắc kẹt trong vòng chờ. Hành động của những người chơi khác sẽ không được xử lý, đây là điều tôi gọi là "chặn" và là một vấn đề mà tôi gặp phải. Do đó, tại sao tôi tạo một chuỗi mới bắt đầu khi nào một hành động được thực hiện

được xử lý trên máy chủ. Phương thức này không chặn luồng chính.

```
# start a new thread so that the action that is being sent doesn't halt reading other client messages
_thread.start_new_thread(
    action, (user_socket, client_action_document["data"]))
continue
```

HÀNH ĐỘNG ĐÓNG GÓI VÀ ĐÓNG GÓI

Một phần khó khăn là tạo một hàm có thể chọn và bỏ chọn đối tượng Action (thành từng byte và từ từng byte). Vì bạn chỉ có thể gửi byte qua kết nối socket, điều này sẽ cho phép tôi gửi và nhận đối tượng hành động qua socket tới cả máy khách và máy chủ.

```
import pickle # parser that is used when accept and send any python class

def pkg_doc_manager(self, action, document):
    """
    Format the document that the server wants to send to the client socket to bytes.
    This will be done by pickling the doc object and adding the heading (length of
    the object in bytes).

    action = The action type that is being packaged up.
    document = The data attached to the action.
    """

    # check if the action and its data are not left empty.
    if not action:
        raise('You can not send an empty action type')
    if not document:
        raise('You can not send an empty document')
    doc = {"action": action, "data": document}

    # This turns the python class into bytes that can be sent to the server.
    pkged_doc = pickle.dumps(doc)

    # The header will contain the length of the pkged_doc in bytes.
    pkged_doc_with_header = bytes(
        f'{len(pkged_doc):<{self.HEADERSIZE}}', 'utf-8')+pkged_doc

    return pkged_doc_with_header # this can be sent over the socket.
```

Tôi đã sử dụng thư viện pickle để phân tích cú pháp của đối tượng python thành từng byte. Sau đó, tôi lấy độ dài của byte dưới dạng số nguyên và thêm vào trước độ dài với kích thước của KÍCH THƯỚC ĐẦU (ví dụ: "512 *****", ví dụ: 2. "12345 *****". Kích thước của tiêu đề sẽ luôn được cố định) đối với đối tượng hành động (tính bằng byte.).

```
# define the size of the length of the header
self.HEADERSIZE = 10
```

Tôi đã bao gồm KÍCH THƯỚC ĐẦU (10 byte) vì điều đó sẽ chứa độ dài của hành động. Nút nhận sẽ có thể đọc 10 byte đầu tiên để có được độ dài đầy đủ của hành động, Điều này sẽ cho nút nhận biết thêm bao nhiêu byte nữa là một phần của hành động đang được gửi,

sau đó đọc theo byte tiếp theo để nhận tài liệu hành động đầy đủ theo byte.

```
def recv_doc_manager(self, client_socket):
    try:
        # Get the header of the package.
        message_header = client_socket.recv(self.HEADERSIZE)

        # This occurs if the user disconnects or sends back no data.
        if not len(message_header):
            return False
        # Remove the extra spaces that we added in the HEADERSIZE and cast the string into a integer.
        document_length = int(message_header.decode('utf-8').strip())
        # Get and store the actual action document that was sent to the server.
        doc = client_socket.recv(document_length)
        return pickle.loads(doc) # Turn bytes into a python object.

    except:
        return False
```

XỬ LÝ KHÁCH HÀNG KHÔNG ĐƯỢC XÁC NHẬN

Khi một ứng dụng khách chưa được xác thực gửi yêu cầu đến máy chủ, nó sẽ được chuyển thành một đối tượng hành động (biến `client_action_document`) và được lọc một trong 2 loại hành động: “[NGƯỜI DÙNG ĐĂNG NHẬP]” hoặc “[ĐĂNG KÝ NGƯỜI DÙNG]”.

[ĐĂNG NHẬP NGƯỜI DÙNG]

```

# HANDLE UNAUTHENTICATED USERS
if client_action_document["action"] == '[USER LOGIN]': # Login attempt
    """
    authenticate the user, this should return the user data in the database
    if the username and password is correct.
    """
    user = self.login_manager(
        client_action_document["data"]) # get user data from database

    if user["result"] is False:
        # client failed to authenticate
        # user[data] is the error message
        client_socket.send(self.pkg_doc_manager(
            "[USER LOGIN - FAIL]", user["data"]))
        continue

    # login user successful
    # user[data] is the users account data
    client_socket.send(self.pkg_doc_manager(
        "[USER LOGIN - SUCCESS]", user["data"]))
    self.sockets_list.append(client_socket) # Accept future request from this socket.
    self.clients[client_socket] = user["data"] # keep track of who is logged in.
    print(
        f'Accepted new connection from {client_address[0]}:{client_address[1]}, \
        action_type: {client_action_document["action"]}, Username: {user["data"][0]}')
    continue

```

Hàm login_manager chịu trách nhiệm truy vấn cơ sở dữ liệu và kiểm tra xem mật khẩu được gửi có khớp với mật khẩu đã băm được lưu trong cơ sở dữ liệu cho tên người dùng đó hay không.

```

def login_manager(self, userCredentials: ("username", "password") ):
    """
    Handle the authentication of the client.
    This function will query the database for the desired username and compare hashed passwords
    if they match, this will return the desired userdata <minus the hashed password>
    """
    try:
        c = self.DB.connection.cursor()
        c.execute("SELECT username, password FROM users WHERE username = :username", {
            "username": userCredentials[0]})
        userCredentials_fromDB = c.fetchone()

        if userCredentials_fromDB == None:
            # there is no accounts with the passed in username, return error
            return {"result": False, "data": f"No user found with the username: {userCredentials[0]}"}

        hashedPassword = hashlib.md5() # set hashing algorithm to MD5 <not suitable for production>
        hashedPassword.update(bytes(userCredentials[1], 'utf-8')) # hash the provided password
        # check if hashed passwords match
        if userCredentials_fromDB == (userCredentials[0], hashedPassword.hexdigest()):
            c.execute("SELECT username, wins, loses, games_played FROM users WHERE username = ?",
                      (userCredentials[0],))
            userData = c.fetchone()

            return {"result": True, "data": userData} # return user data
        else:
            return {"result": False, "data": "Incorrect password"} # return error

    except BaseException as e:
        print(e)
        return {"result": False, "data": "Error when authenticating client's account."}

```

[ĐĂNG KÝ NGƯỜI DÙNG]

Trình tự này hoạt động như tôi đã đề cập trong phần “thiết kế - loại hành động - đăng ký người dùng” của tài liệu này.

```

if client_action_document["action"] == '[USER REGISTER]': # Register attempt
    # register the user in the db
    create_account_status = self.registration_manager(
        client_action_document["data"])

    if create_account_status["result"] is False:
        # something went wrong whilst creating the user account on the database
        client_socket.send(self.pkg_doc_manager(
            "[USER REGISTER - FAIL]", create_account_status["msg"]))
        continue

    # tell the client that they have successfully created an account in the database
    client_socket.send(self.pkg_doc_manager(
        "[USER REGISTER - SUCCESS]", create_account_status["msg"]))
    print('Created new account from {}:{}'.format(
        *client_address, client_action_document["action"]))
    continue

```

Hàm register_manager chịu trách nhiệm tạo dữ liệu người dùng mới trong cơ sở dữ liệu SQLite.

```

def registration_manager(self, userCredentials):
    """
    Register the username on the database only if the username isn't taken

    userCredentials = ("username", "password")
    """

    try:
        c = self.DB.connection.cursor()
        c.execute("SELECT username FROM users WHERE username = :username",
                  {"username": userCredentials[0]})
        userCredentials_fromDB = c.fetchone()

        if userCredentials_fromDB == None:
            # create new user WITH USERNAME AND PASSWORD because there is no user with the desired username
            hashedPassword = hashlib.md5()
            hashedPassword.update(bytes(userCredentials[1], 'utf-8'))

            c.execute(
                "INSERT INTO users (username, password) VALUES (?, ?)", (userCredentials[0], hashedPassword.hexdigest()))
            self.DB.connection.commit()
            return {"result": True, "msg": "Account was created successfully."}
        else:
            return {"result": False, "msg": "Username already exists."}
    except BaseException as e:
        print(e)
        return {"result": False, "msg": "Error when creating client's account."}

```

VĂN ĐỀ ĐĂNG NHẬP:

- Tôi quyết định giải quyết phần giao tiếp máy khách và máy chủ trước, vì vậy, tôi bắt đầu nghiên cứu cách thiết lập một máy chủ ổ cắm sẽ lắng nghe các yêu cầu. Tuy nhiên, tôi đã gặp phải sự cố trong đó máy chủ không biết cách xử lý dữ liệu đến. Đó là khi tôi triển khai đối tượng "Hành động" và tìm hiểu về việc loại bỏ các đối tượng Python. Giờ đây, việc gửi dữ liệu giữa ổ cắm máy khách và ổ cắm máy chủ đã có thể dự đoán được và giúp việc xử lý dữ liệu nhận / gửi dễ dàng hơn nhiều.

- Khi tạo giao diện người dùng, tôi gặp khó khăn khi chuyển đổi màn hình, đó là khi tôi đã triển khai hàm `switch_to_frame` sẽ đưa khung mà bạn đang chuyển sang lên phía trước, sau đó loại bỏ khung trước đó. Bằng cách làm này, tôi có thể dễ dàng chuyển đổi chế độ xem nếu một nút được nhấp hoặc trạng thái điều kiện (ví dụ: đăng nhập chế độ xem vào chế độ xem chính).
- Một vấn đề khác đã được trình bày khi tôi làm việc với các chế độ xem động (ví dụ: cập nhật người lãnh đạo bảng sau khi lấy dữ liệu từ máy chủ). Tôi cần một cách để cập nhật toàn bộ khung hình. Giải pháp của tôi là thêm chức năng kết xuất cho mỗi khung xem. Điều này cho phép tôi kết xuất lại chế độ xem mỗi khi tôi muốn xây dựng lại chế độ xem. Tôi cũng thực hiện hiển thị lại từng chế độ xem mỗi khi chế độ xem được chuyển sang (`switch_to_frame`).

```
def switch_frame_to(self, frame: tk.Frame):
    """
    This function allows navigation between frame (views).
    """
    # tries to find the frame instance in self.frames
    frame = self.frames[frame]
    if frame != None: # check if there is an initialize frame that has been created.
        self.PreviousFrame.grid_remove() # remove the last frame so the page will resize to best fit height and width
        frame.grid(row=0, sticky="news")
        self.PreviousFrame = frame
        # then re-build the frame but with the new frame.
        frame.render() # each frame view should have this method, it allows me to re-render views with updated Data
        frame.tkraise() # brings the frame to the front of the window.
    return
```

<https://pythonprogramming.net/change-show-new-frame-tkinter/>: Tôi đã sử dụng bài viết này để đi đến giải pháp của mình.

KIỂM TRA KỊCH BẢN KHÁC BIỆT: GIAO DIỆN NGƯỜI DÙNG

Giao diện người dùng của tôi gần giống với thiết kế của tôi.

Các lệnh được sử dụng để khởi động máy khách và máy chủ. Các chương trình cần python 3.7 hoặc cao hơn để chạy. Nếu bạn sử dụng python 2.7, các chương trình sẽ gặp lỗi khi sử dụng các thao tác chuỗi thông qua f string. Đây là một tính năng của python 3.7.

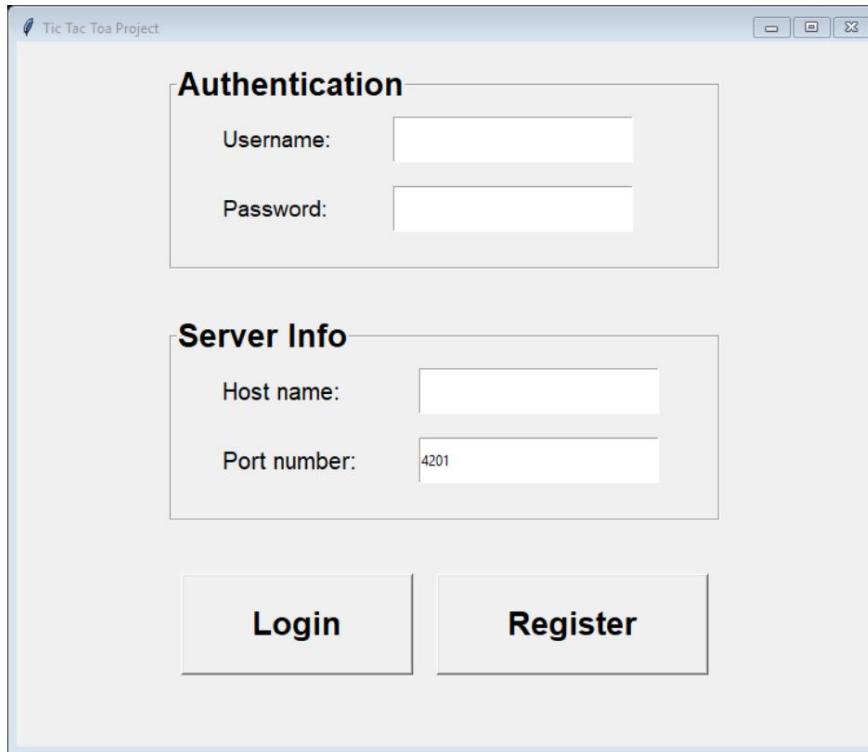
```
PS C:\Users\090492276\OneDrive - The City of Edinburgh Council\school\Computing\AH Project\Project Code> PY .\client.py
```

```
PS C:\Users\090492276\OneDrive - The City of Edinburgh Council\school\Computing\AH Project\Project Code> PY .\server.py
server started on Host: EDU-D004872 , Port: 4201
```

Tôi đã chạy ứng dụng này trên máy tính của trường học, tên máy chủ của máy chủ là EDU-D004872 và nó đang chạy trên cổng 4201

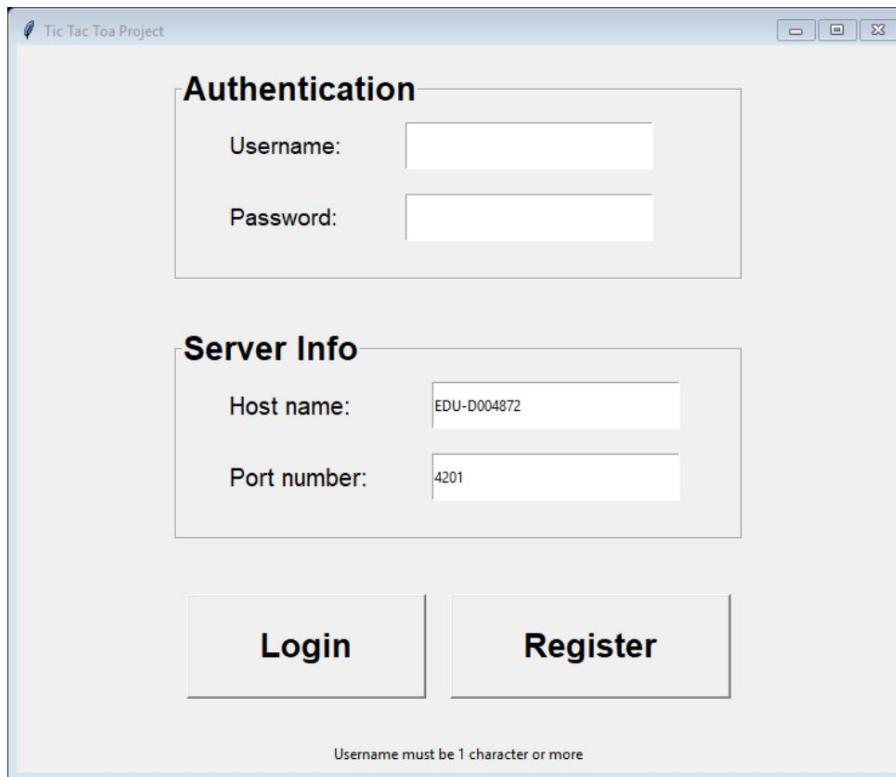
Trang xác thực

Đây là trang đầu tiên mà mọi người chơi sẽ thấy. Họ được yêu cầu cung cấp tên người dùng, mật khẩu tài khoản, địa chỉ của máy chủ và cổng mà máy chủ đang chạy để đăng nhập hoặc đăng ký tài khoản mới. Điều này gần giống với thiết kế giao diện người dùng của tôi.



KIỂM TRA ĐẦU VÀO ĐÚNG

Tôi đã thử đăng ký và đăng nhập một tài khoản mà không có tên người dùng nhập, điều này sẽ dẫn đến lỗi.



Chương trình khách hàng chọn thành công điều này và cho người dùng biết rằng phải có ít nhất 1 ký tự trong trường nhập tên người dùng.

Tiếp theo, tôi đã thử nghiệm đăng ký và đăng nhập một tài khoản với mật khẩu của bạn ít hơn 6 ký tự, điều này cũng sẽ dẫn đến lỗi.

The screenshot shows a Windows application window titled "Tic Tac Toe Project". Inside, there are two main sections: "Authentication" and "Server Info".

Authentication:

- Username: testuser
- Password: ****

Server Info:

- Host name: EDU-D004872
- Port number: 4201

At the bottom, there are two large buttons: "Login" and "Register". A message "Password must be 6 characters or more" is displayed below the "Register" button.

Ứng dụng khách cũng bắt lỗi này và nói với người dùng rằng mật khẩu của họ phải dài từ 6 ký tự trở lên.

ĐĂNG KÝ KIỂM TRA NÚT**LƯU Ý ĐĂNG KÝ TÊN NGƯỜI DÙNG ĐÃ TỒN TẠI TRONG CƠ SỞ DỮ LIỆU**

Điều này sẽ dẫn đến lỗi vì tên người dùng là duy nhất cho mỗi người chơi.

Tic Tac Toe Project

Authentication

Username:

Password:

Server Info

Host name:

Port number:

Login **Register**

Username already exists.

Máy chủ phản hồi với lỗi "tên người dùng đã tồn tại". Nếu chúng ta nhìn vào tệp SQLite, chúng ta có thể thấy rằng đã có tên người dùng là "test1" trong cơ sở dữ liệu. (Tôi đã sử dụng <https://sqliteonline.com/> để xem và gỡ lỗi tệp SQLite của tôi)

```
SQLite
1 | SELECT * FROM users;
```

Username	Password	Wins	Loses	Games_played
test1	60474c9c10d7142b7508ce7a50acf414	3	6	9
test2	60474c9c10d7142b7508ce7a50acf414	4	5	9

ĐĂNG KÝ THÀNH CÔNG NGƯỜI DÙNG VỚI TÊN NGƯỜI DÙNG DUY NHẤT.

Điều này sẽ dẫn đến một thực thể mới được tạo trong bảng người dùng với tên người dùng được cung cấp của tôi, mật khẩu băm và thông kê trình phát mặc định.

The screenshot shows a Windows application window titled "Tic Tac Toe Project". It contains two main sections: "Authentication" and "Server Info".

Authentication:

- Username: testuser
- Password: *****

Server Info:

- Host name: EDU-D004872
- Port number: 4201

At the bottom of the window, there are two buttons: "Login" and "Register". A message "Account was created successfully." is displayed at the bottom center.

Trong thử nghiệm này, tôi đã tạo tài khoản người dùng mới với tên người dùng là "testuser" và mật khẩu là "test12". Như bạn có thể thấy, người dùng mới được lưu trong cơ sở dữ liệu SQLite với số liệu thống kê mặc định và mật khẩu được băm.

Isaac Diaby

Nâng cao tính toán cao hơn
Khoa học

SCN

The screenshot shows a SQLite database interface with a toolbar at the top. A sidebar on the left lists tables: 'users' (selected), 'temp' (disabled), and 'temp1' (disabled). The main area displays the 'users' table with the following data:

	Username	Password	Wins	Loses	Games_played
1	test21	60474c9c10d7142b7508ce7a50acf414	1	1	2
2	testuser	60474c9c10d7142b7508ce7a50acf414	0	0	0

NÚT ĐĂNG NHẬP

ĐĂNG NHẬP VỚI TÊN NGƯỜI DÙNG KHÔNG TỒN TẠI

Điều này sẽ dẫn đến lỗi vì không có người dùng với tên người dùng được cung cấp trong cơ sở dữ liệu.

The screenshot shows a Java Swing application window titled "Tic Tac Toe Project". The window has two main panels: "Authentication" and "Server Info".

Authentication Panel:

- Label: Username:
- Label: Password:

Server Info Panel:

- Label: Host name:
- Label: Port number:

Action Buttons:

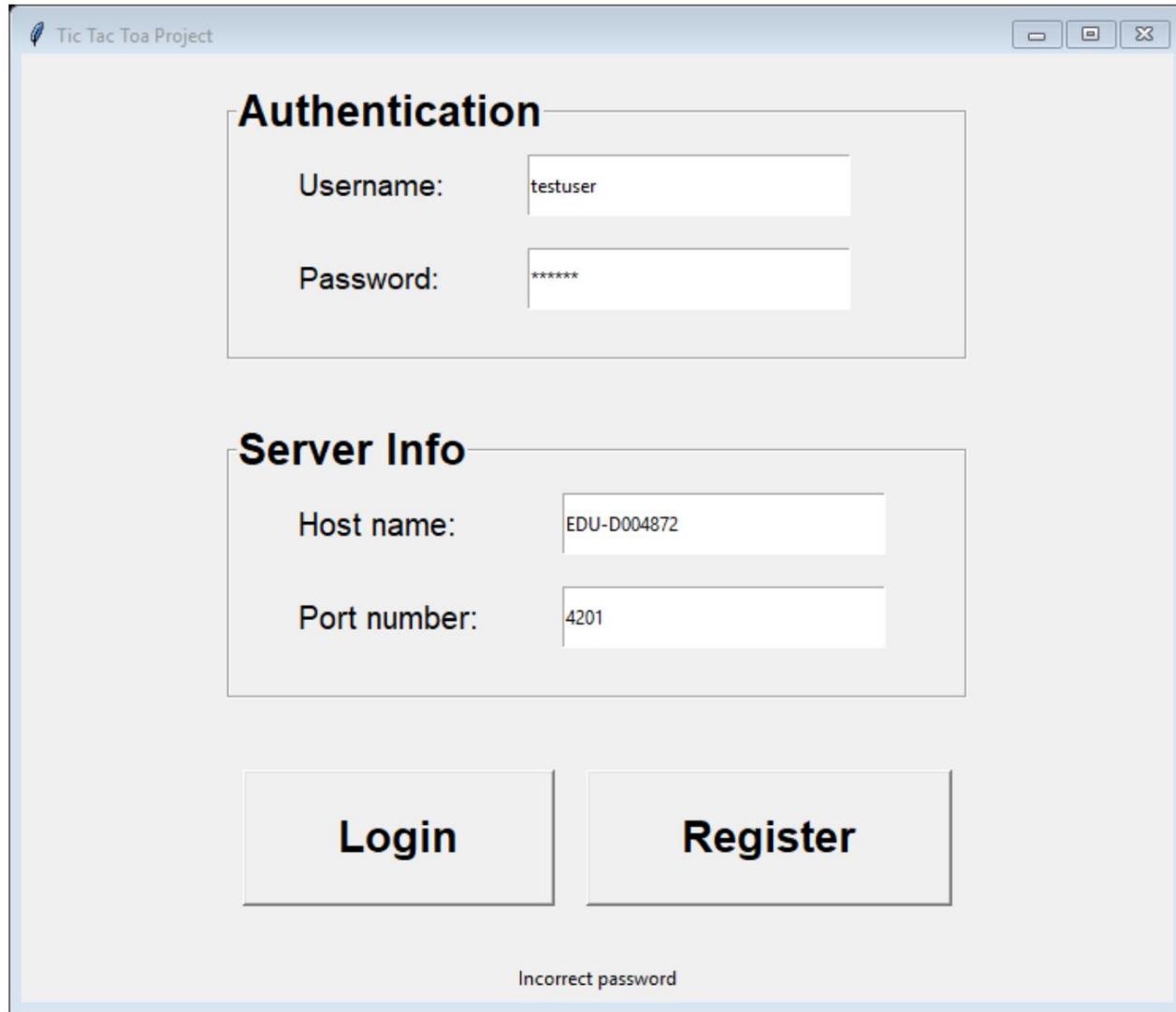
- Login** (Left button)
- Register** (Right button)

Message: No user found with the username: testuser

Trước khi tạo tài khoản, tôi đã cố gắng đăng nhập vào tài khoản "testuser" không tồn tại. Máy chủ phản hồi với lỗi "Không tìm thấy người dùng với tên người dùng: testuser" được hiển thị cho người dùng.

LƯU Ý ĐĂNG NHẬP BẰNG MẬT KHẨU SAI CHO TÊN NGƯỜI DÙNG ĐÃ TỒN TẠI.

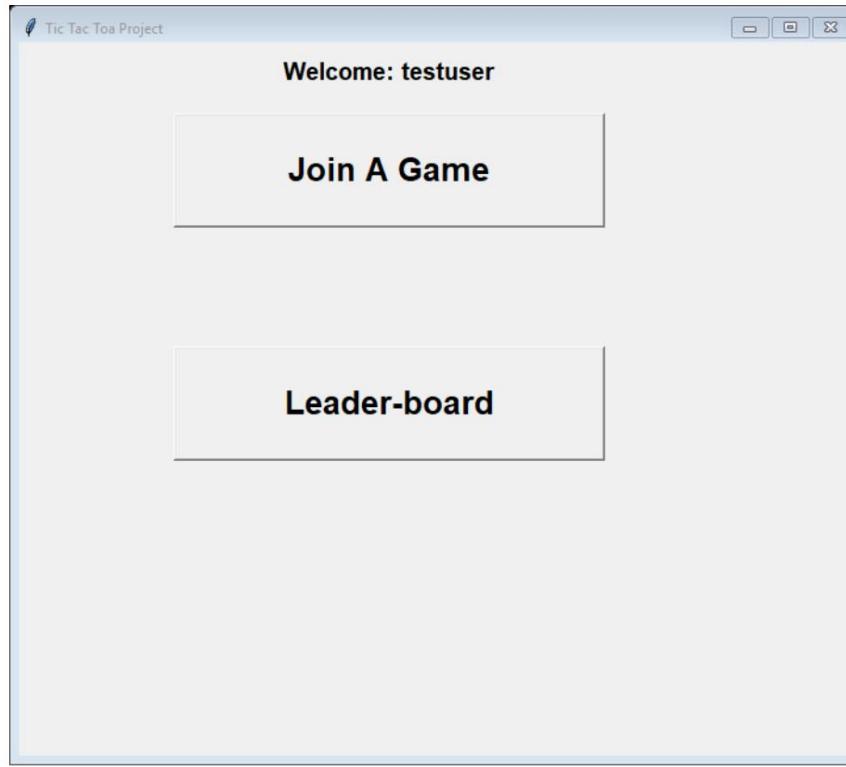
Điều này sẽ dẫn đến lỗi vì để được xác thực, mật khẩu bạn gửi đến máy chủ phải khớp với mật khẩu được băm trong cơ sở dữ liệu được liên kết với tên người dùng khi mật khẩu bạn cung cấp được băm.



Máy chủ xác định thành công rằng mật khẩu được băm được cung cấp không khớp với mật khẩu được băm được liên kết với tên người dùng. Máy chủ phản hồi bằng thông báo "Mật khẩu không chính xác" được hiển thị cho người dùng.

CUNG CẤP THÀNH CÔNG MẬT KHẨU ĐÚNG VỚI TÊN NGƯỜI DÙNG TỒN TẠI
CƠ SỞ DỮ LIỆU

Thao tác này sẽ xác thực thành công trình phát và chuyển màn hình sang trang chủ.



Ở cắm máy khách của người chơi hiện đã được ỏ cắm máy chủ xác thực (chấp nhận) thành công. Bất kỳ yêu cầu nào khác từ ỏ cắm máy khách đó sẽ được đăng ký là “người dùng thử nghiệm”

TRANG BẢN LÃNH ĐẠO

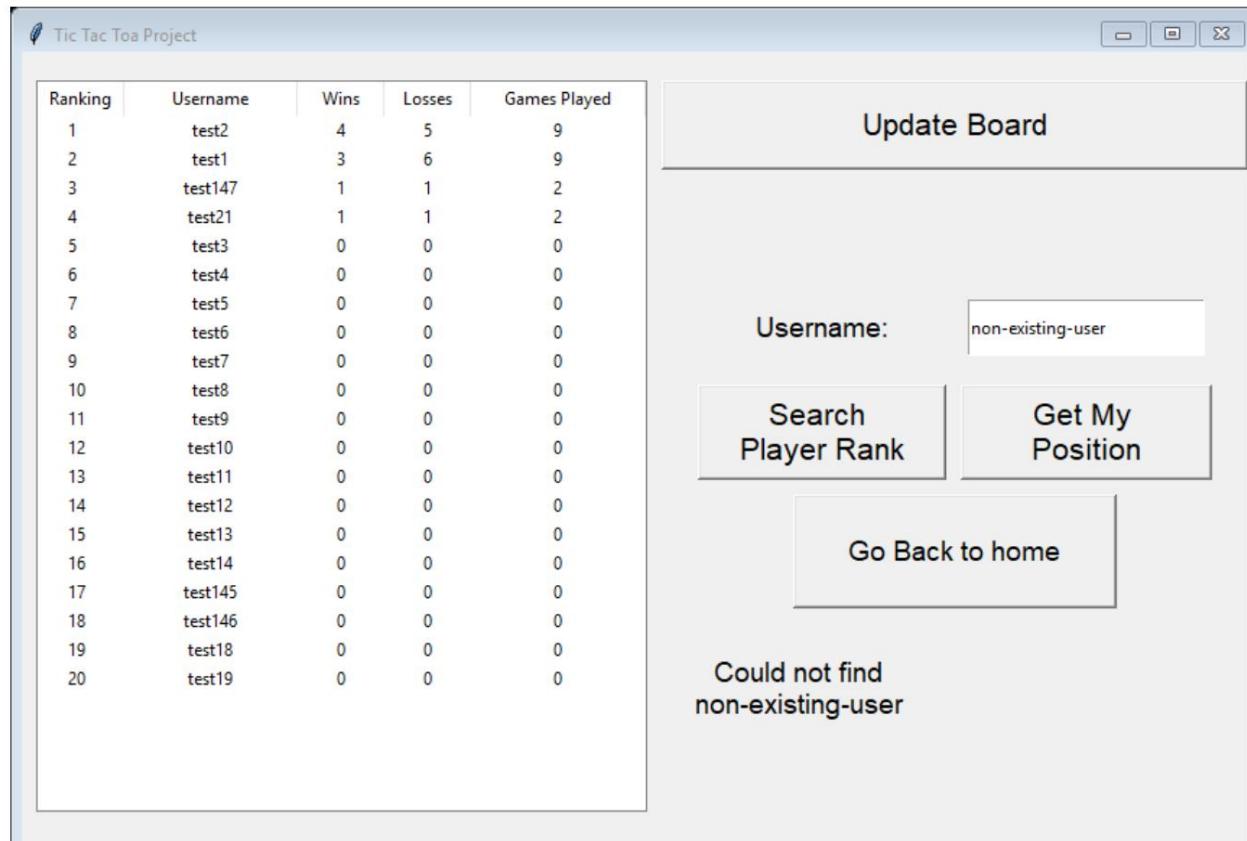
Khi nháp vào nút bảng xếp hạng, người chơi sẽ thấy trang bảng xếp hạng. Điều này cũng phù hợp với thiết kế của tôi. Theo yêu cầu, chỉ 20 người chơi hàng đầu (có số chiến thắng cao nhất) mới được hiển thị trên bảng xếp hạng.

The screenshot shows a "Leader Board" page. On the left is a table with columns: Ranking, Username, Wins, Losses, and Games Played. The table lists 20 entries from 1 to 20. On the right are several buttons: "Update Board", "Username: testuser" (with an input field), "Search Player Rank", "Get My Position", and "Go Back to home".

Ranking	Username	Wins	Losses	Games Played
1	test2	4	5	9
2	test1	3	6	9
3	test147	1	1	2
4	test21	1	1	2
5	test3	0	0	0
6	test4	0	0	0
7	test5	0	0	0
8	test6	0	0	0
9	test7	0	0	0
10	test8	0	0	0
11	test9	0	0	0
12	test10	0	0	0
13	test11	0	0	0
14	test12	0	0	0
15	test13	0	0	0
16	test14	0	0	0
17	test145	0	0	0
18	test146	0	0	0
19	test18	0	0	0
20	test19	0	0	0

TÌM KIẾM RANK CỦA NGƯỜI CHƠI KHÔNG TỒN TẠI

Điều này sẽ dẫn đến lỗi vì chương trình sẽ không thể tìm thấy tên người dùng mong muốn trong mảng được sắp xếp của tất cả thống kê người chơi.



Chương trình hiển thị thông báo "Không thể tìm thấy người dùng không tồn tại" cho người chơi. Điều này có nghĩa là nó không thể tìm thấy người chơi và thông báo cho người chơi.

TÌM KIẾM RANK CỦA TÔI

Điều này sẽ dẫn đến việc người chơi nhận được thứ hạng của họ trong số tất cả những người chơi hiện có.

Tic Tac Toa Project

Ranking	Username	Wins	Losses	Games Played
1	test2	4	5	9
2	test1	3	6	9
3	test147	1	1	2
4	test21	1	1	2
5	test3	0	0	0
6	test4	0	0	0
7	test5	0	0	0
8	test6	0	0	0
9	test7	0	0	0
10	test8	0	0	0
11	test9	0	0	0
12	test10	0	0	0
13	test11	0	0	0
14	test12	0	0	0
15	test13	0	0	0
16	test14	0	0	0
17	test145	0	0	0
18	test146	0	0	0
19	test18	0	0	0
20	test19	0	0	0

Update Board

Username:

[Search Player Rank](#)
[Get My Position](#)

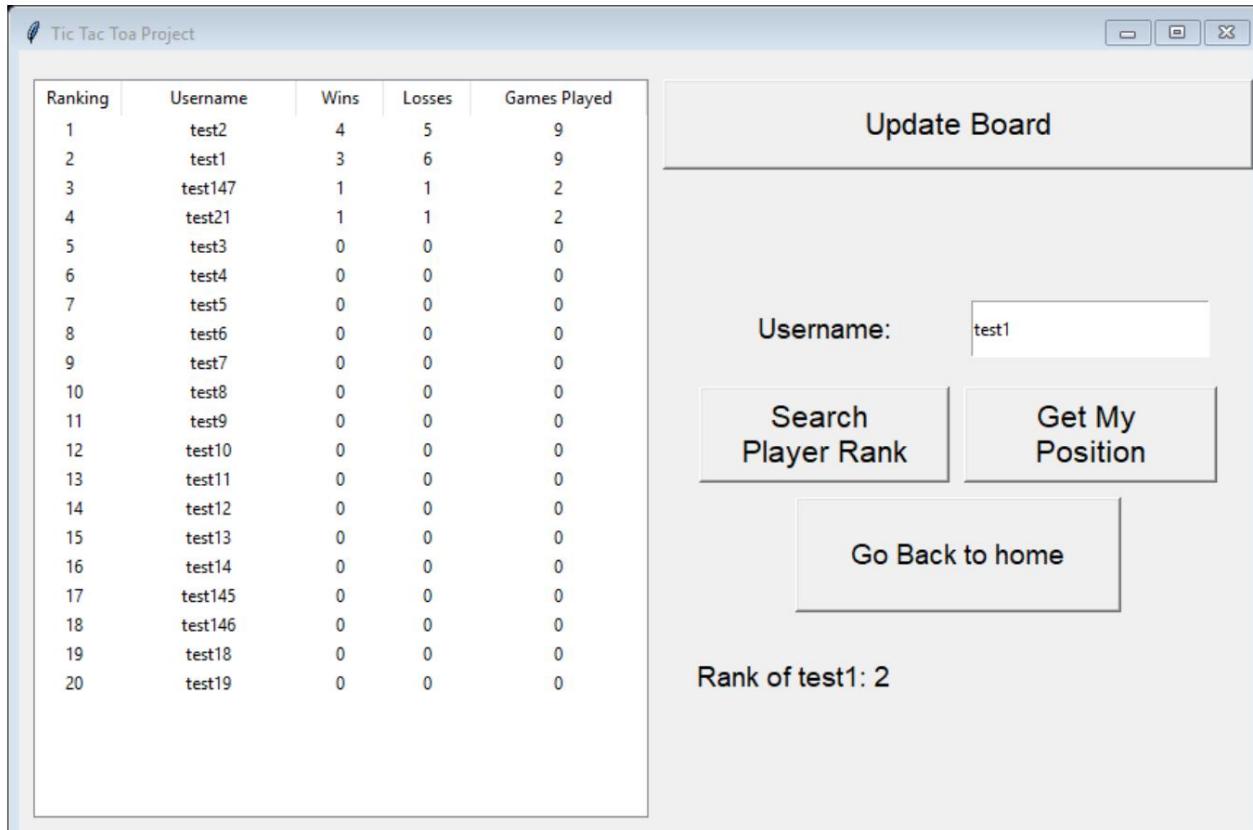
[Go Back to home](#)

Rank of testuser: 22

Chương trình đã hiển thị thành công thứ hạng người chơi hiện tại.

TÌM KIẾM NGƯỜI CHƠI KHÁC RANK

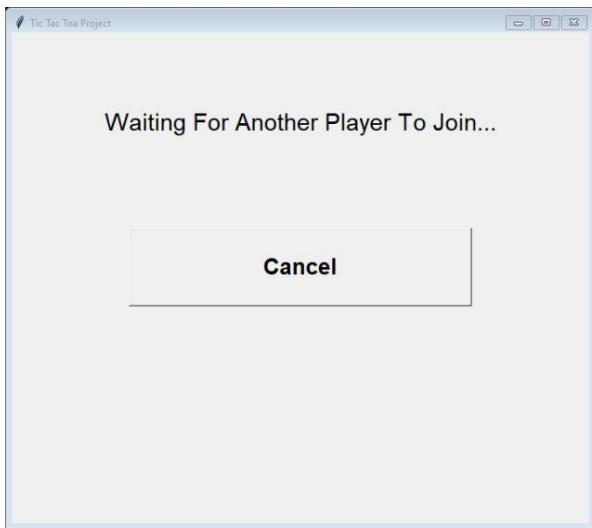
Điều này sẽ dẫn đến việc người chơi nhận được thứ hạng của tên người dùng mong muốn trong số tất cả những người chơi hiện có nếu tên người dùng đó tồn tại.



Trong bài kiểm tra này, tôi muốn có được thứ hạng của tên người dùng "test1", người được xếp hạng 2 với tổng số 3 chiến thắng. Chương trình trả về thành công "Xếp hạng của test1: 2"

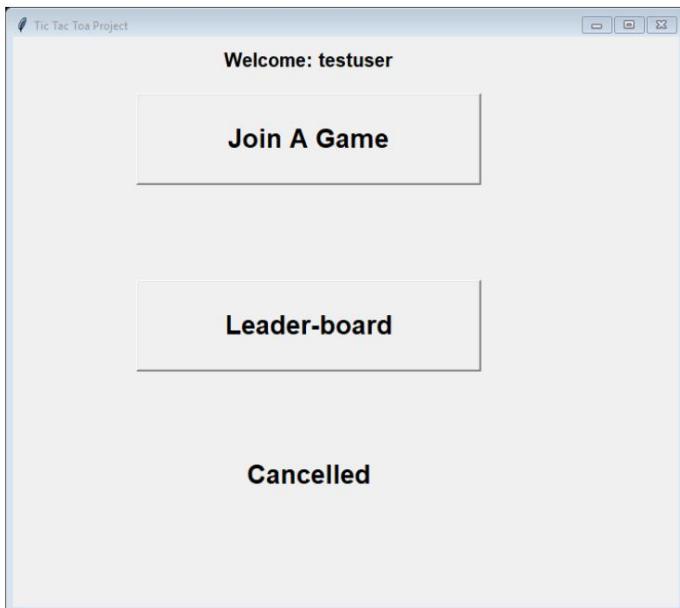
THAM GIA TRANG TẢI TRÒ CHƠI

Khi người chơi nhấp vào nút "chơi trò chơi", họ sẽ thấy trang "tham gia trò chơi". Người chơi hiện đang trong hàng đợi trên máy chủ. Logic là giống nhau trong thiết kế của tôi. Chờ đợi máy chủ phản hồi với đối thủ để đấu được xử lý trên một chuỗi khác, điều này cho phép người chơi vẫn hủy trong khi chờ trong hàng đợi.



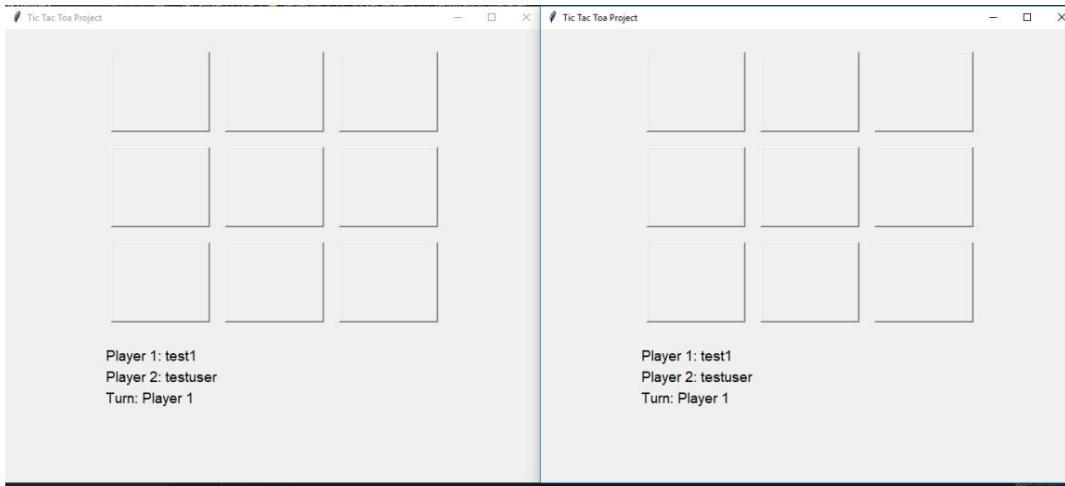
NHẤP VÀO NÚT HỦY.

Thao tác này sẽ xóa người chơi khỏi hàng đợi trên máy chủ. Máy chủ xử lý vấn đề này bằng cách luôn kiểm tra xem người chơi có còn chờ để tham gia trò chơi hay không khi họ đang đợi.

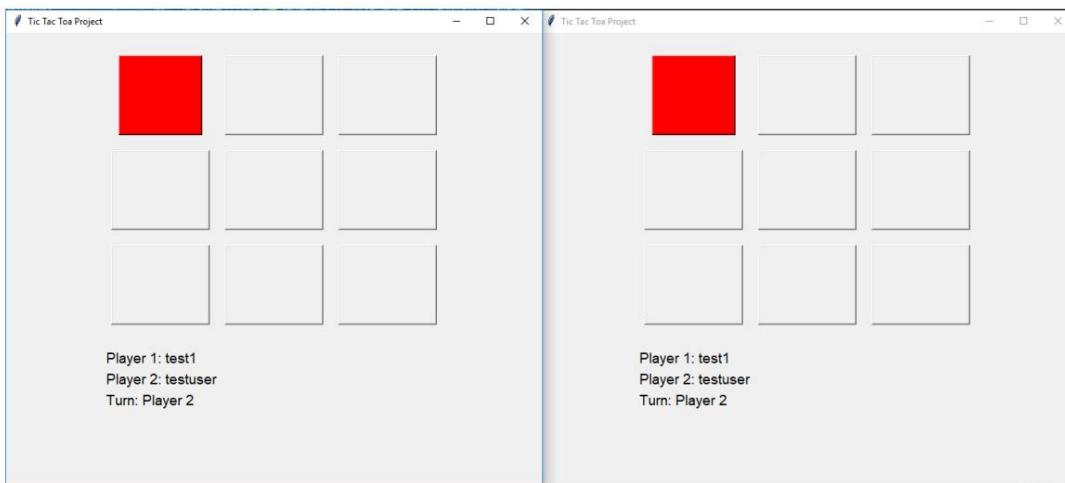


THÀNH CÔNG NHẬN ĐƯỢC 2 NGƯỜI CHƠI TRONG MỘT TRÒ CHƠI

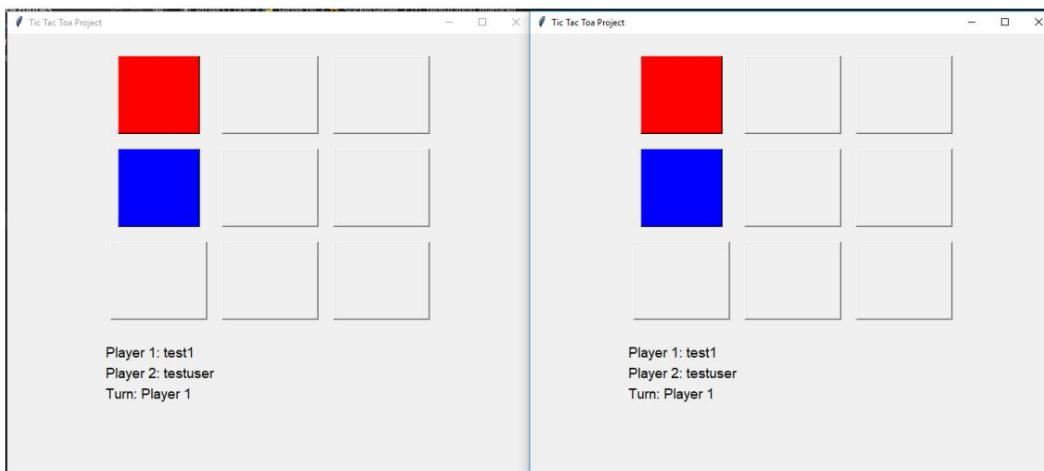
Khi 2 người chơi đang đợi trong hàng đợi, yêu cầu máy chủ sẽ tạo dữ liệu trò chơi và thông báo cho cả hai người chơi rằng trò chơi đã bắt đầu. với dữ liệu này, chương trình khách hàng sẽ tạo giao diện người dùng dưới phù hợp với thiết kế của tôi

**NGƯỜI CHƠI MANG ĐI**

Người chơi 1 đi một lượt.



Người chơi 2 sẽ đến lượt.

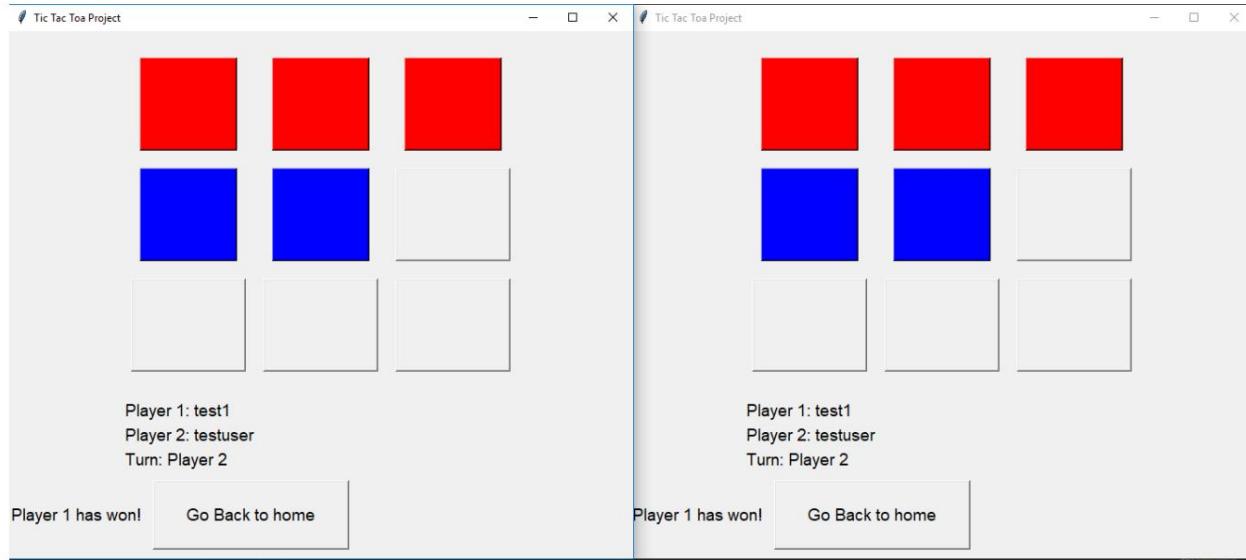


Isaac Diaby

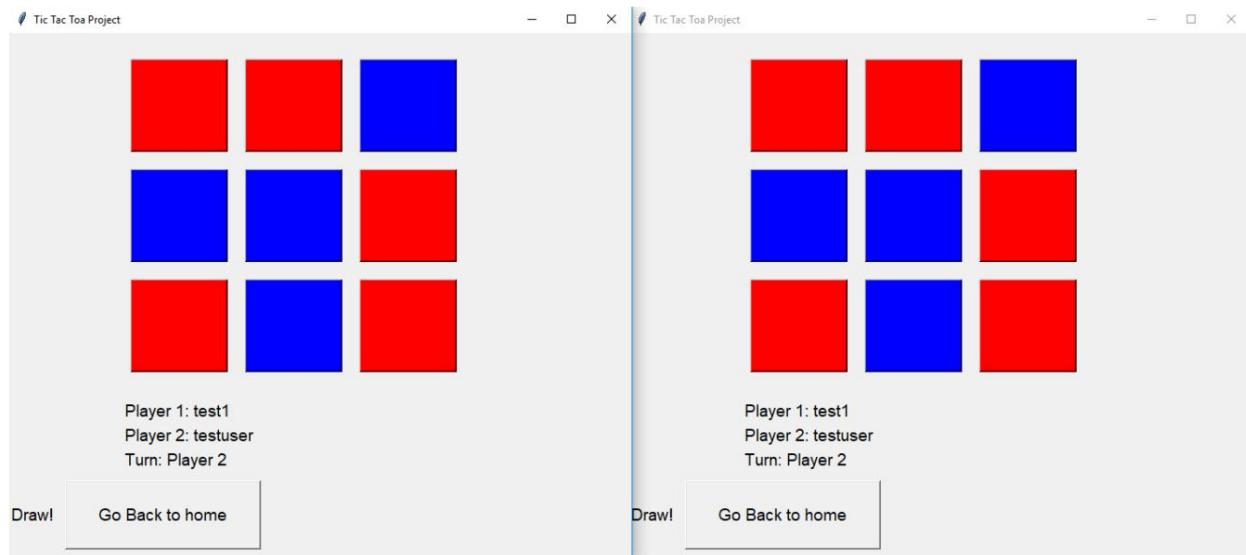
Nâng cao tính toán cao hơn
Khoa học

SCN

Người chơi 1 thắng.



Trò chơi có kết quả hòa.



Mỗi thông kê tài khoản người chơi được cập nhật sau trận đấu phản ánh kết quả của trò chơi. Ví dụ: vì người chơi "testuser" đã thua một trò chơi và rút ra trò chơi tiếp theo, thống kê của nó trong cơ sở dữ liệu bây giờ là: 0 thắng, 2 thua 2 trận đã chơi.

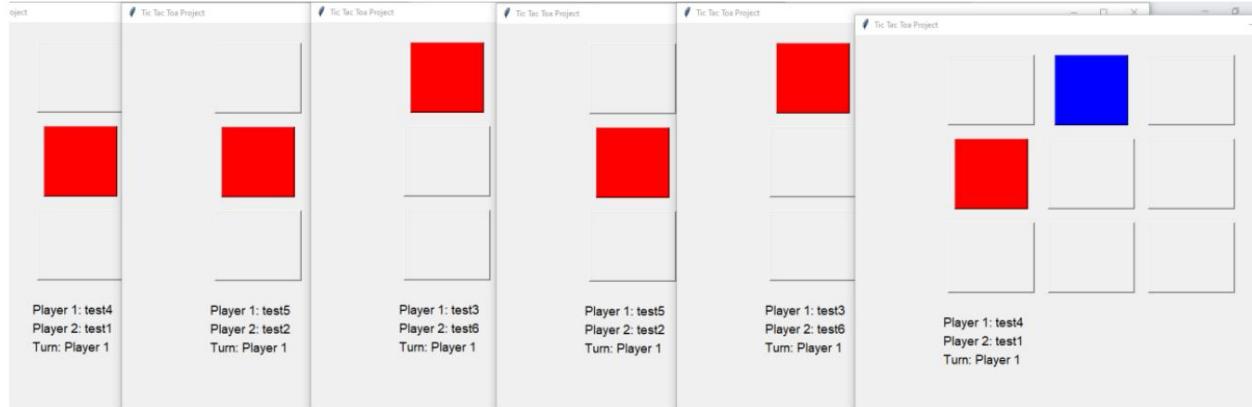
SQLite

```
1 SELECT * FROM users;
```

Username	Password	Wins	Loses	Games_played
testuser	60474c9c10d7142b7508ce7a50acf414	0	2	2

KẾT NỐI NHIỀU MÁY CHỦ 6

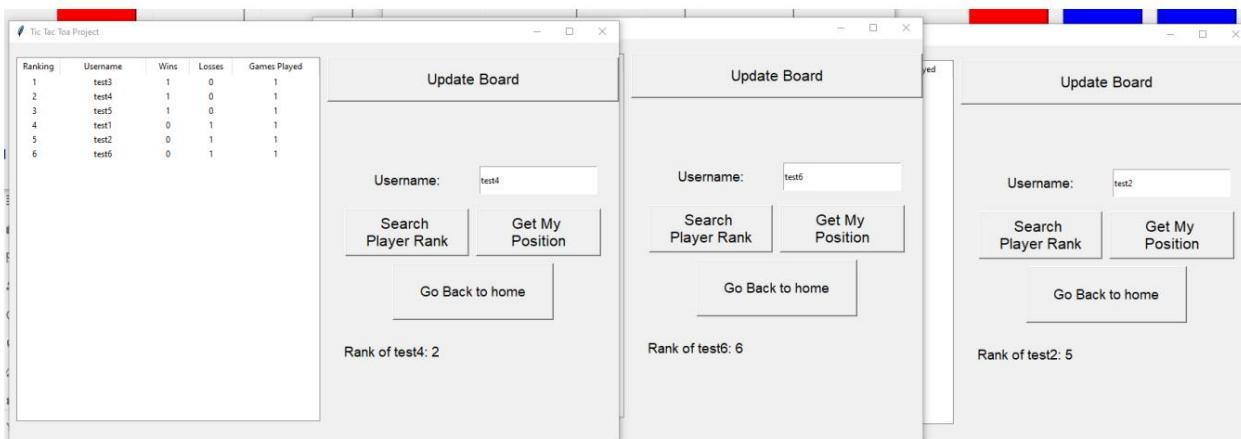
Tôi đã kiểm tra xem máy chủ có thể xử lý 6 máy khách được kết nối chơi 3 trò chơi khác nhau hay không.



Máy chủ đã ghép thành công từng người chơi vào một trò chơi và có thể theo dõi cả 3 trò chơi cùng một lúc.



Máy chủ cũng đã có thể xác định trạng thái kết thúc cho cả 3 ván đấu thành công.



Cuối cùng, máy chủ đã có thể cập nhật chính xác dữ liệu từng người chơi trong cơ sở dữ liệu thành công.

SỰ ĐÁNH GIÁ

Vì tất cả các thử nghiệm của tôi đều thành công, tôi tin rằng ứng dụng mà tôi đã viết đáp ứng đầy đủ các yêu cầu về đặc điểm kỹ thuật:

- Người chơi tương tác với giao diện người dùng rõ ràng và dễ sử dụng với xác thực đầu vào.
- Người chơi có thể đăng ký tài khoản và xác thực khách hàng của mình. Điều này cung cấp một mức độ bảo mật cho người chơi và cơ sở dữ liệu.
- Mật khẩu người chơi được băm trong cơ sở dữ liệu. • Máy chủ và máy khách sử dụng ổ cứng python để giao tiếp.
- Ngoài ra, máy chủ sử dụng lập trình đồng thời và phân luồng để xử lý mức tối thiểu yêu cầu máy khách là 6, điều này cũng cho phép máy chủ xử lý nhiều người chơi hơn.
- Người chơi có thể chơi trò chơi đánh dấu và cập nhật số liệu thống kê vào cuối trò chơi. Đây làm cho việc chơi trò chơi trở nên thú vị và có tiền đặt cọc cho mỗi trò chơi.
- Người chơi có thể xem bảng xếp hạng, bảng này hiển thị 20 người chơi hàng đầu trong danh sách. Điều này sử dụng sắp xếp chèn để sắp xếp người chơi theo số trò chơi, thứ tự giảm dần đã thắng.
- Các lớp cũng có thể tìm kiếm xếp hạng của họ và xếp hạng người chơi khác.
- Khởi động máy chủ dễ dàng và dễ dàng di chuyển.

Tôi đã ghi lại đầy đủ mã của mình và thêm các mô tả nhỏ cho từng chức năng. Mã mà tôi đã viết là hướng đối tượng và mô-đun. Điều này giúp cho việc cập nhật hoặc thay đổi các phần trong mã của tôi trở nên rõ ràng và dễ dàng. Tôi cũng cung cấp tệp README.md và một số tập lệnh bash để người dùng cuối dễ dàng bắt đầu. Bằng cách làm tất cả những điều này, mã của tôi dễ bảo trì hơn.

Khi viết các truy vấn SQL của mình, tôi cũng đã suy nghĩ về khả năng chèn SQL. Tôi đã sử dụng lối thoát nhân vật dụng sẵn làm cho các câu lệnh SQL của tôi an toàn hơn. Ngoài ra, tôi đã thiết lập xác thực đầu vào mạnh mẽ ở phía máy khách và các chương trình giải thích và xử lý lỗi khi chúng xảy ra. Những điều này làm cho mã của tôi mạnh mẽ hơn.