Server/Client Architecture Documentation

Team: BooKeD Blocks

Ben Sottile, Kashod Cagnolatti,
and David Cruz

Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Table of Contents

Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

[Request for Proposal]

- **Introduction**

  Our goal is to make a server/client architecture that is both general purpose and provides a positive user experience. The program will allow for users to sign up to access the server, sign in with their credentials, and retrieve them if anything goes wrong. The program will allow admins to make queries to the program and get information about prespecified commands. The program will be multithreaded so as to allow multiple users to interact with and be logged into the server at any given time.

- **Perspectives**
  - o Customer
    - Customer may use this software for monetary purposes
    - Customer wants the application to be completed in a timely manner and to specification.
    - Using as a general app for future applications that require communication to a database via a simple GUI
    - May want source code and/or documentation for future maintenance.
  - o User
    - Want a completely functional program
      - Ex: login/logout, password recovery, etc should work
    - Easy to navigate / intuitive interface / user-friendly / bug-free
    - The user should be able to assume their data is protected and their emails and passwords are secure.
    - User can use this system to register for an email
  - o Development Team
    - App is general purpose in terms of connecting to a database via GUI, provides a good template for future projects (Intellectual Property)
    - Might provide experience and material for portfolio.
    - May allow for positive word of mouth advertising from customers and users.

- **List of concerns regarding the specification**

  o E.g. Is there something about this proposal that makes you nervous?
  - MySQL - None of us have experience using MySQL, so this is the most worrisome part of the project.
  - Client-Server Networking - We are not familiar with implementing a single-threaded client and a multi-threaded server.
  - JavaFX - We have some experience with JavaFX; however, we would not say we are comfortable with implementing a GUI on our own. We will need to look at

tutorials and consult with our technology consultant for additional help.

- **List of questions/clarifications regarding specification**
    - Are there items in the proposal that you do not understand or for which you would like clarification?
        - Is there anything displayed after the user logs into the system?
        - Should the option of connecting to and disconnecting from the server be given to the user through the GUI or should it be done automatically (ex: connect while logging in or disconnect while logging out)?
        - If the user can manually connect/disconnect, should the option of connecting/disconnecting be given during all instances of using the application or only before login?
            - If you shut down, log out and disconnect?
        - How should passwords be stored? Encrypted, Hashed**, plain text?**
        - How will attempts to enter be limited? Will it be limited via session or by username? If limited by username will it have to verify username first? Would username verification be unlimited attempts?How are attempts to open up a username stored?If the user enters one user name messes up leaves the client and opens it again will attempts be reset? Will attempts refresh with the passing of time?
            - Type in username and password- invalid login. After 3 logins with a specified username locked perpetually counted failed
            - Log in zeros account
            - Server side does all the stuff, Client just gets info
        - How should user data be stored? Encrypted, Hashed, **plain text**?
        - How will query information be displayed?
        - How will queries be activated? Type in commands, **buttons** etc?
        - How to connect?-> Read string
        - Can you log in twice - yes just count

- **Development environment to be used**

    o What development environment will you use?

    Intellij - We all have experience using IntelliJ. We are comfortable using this integrated development environment.

    o What GUI library will you use?

    JavaFX w/ JDK 14 - It provides many options for sleek and modern UI. Our technology consultant said he was relatively experienced in the tech

o What email server will you use?

Gmail SMTP protocol - Making a new gmail is relatively easy and there is a lot of tutorial info out there. Our technology consultant said he was relatively experienced in the technology.

o What database management system will you use?

MySQL - This is the suggested database management system for the project. We also chose MySQL because it is a widespread database management service that has tutorials online. Our technology consultant said he was relatively experienced in this technology.

- **Biographies**

Kashod Cagnolatti

Kashod is currently a student at California Lutheran University studying Computer Science. He has participated in a number of competitive programming contests and game jams that have expanded his resourcefulness and problem-solving skills. Kashod has experience with working in mid-scale projects with groups with the scrum framework. His largest project is a software based 3D Graphics pipeline built entirely in Java, which allows the user to view and manipulate 2D and 3D shapes. Currently he's working on building a website for documenting his past and future projects.

David Cruz

David Cruz is a 3rd year Computer Science student at California Lutheran University. He started learning computer science in high school after a Mobile Application Development class was offered. He was initially interested in Computer Science because he wanted to make games. He has made simple mini games including Tic-Tac-Toe, Click-on-the-balloon-before-it-disappears, and a slot machine. These games were created long ago using JavaFX or JavaSwing. He is currently interested in creating an automated watering system for his plant using Arduino.

Ben Sottile

Ben Sottile is a student of Computer Science at California Lutheran University. He started programming with Scratch in the 7th grade and has programmed all sorts of things during high school including Vex and Mindstorms Robots and a clone of the Tron game from the movie tron. He took AP Computer Science as a junior and then was the

teacher's assistant for the class. In College he has taken numerous computer science classes including Intro, Intermediate and Advanced Programming, Algorithms, and Systems Analysis. The most advanced project he has created was a Tetris clone with new blocks and the ability to add in any block into the program via editing a csv. The program was later edited to save high scores.

Craig Reinhart, Ph.D.

Dr. Reinhart came to CLU from the computer science industry where he directed research in high-level visualization and image processing programs. The author of a number of articles, conference papers, and US patents, Dr. Reinhart worked at Hughes Aircraft, Rockwell International Science Center before joining a team to launch a startup company developing low cost digital cameras. He is currently a technical consultant to Red Digital Cinema, Inc. His interests include image processing/computer vision, computer graphics, and robotics.

[Use Cases]

A. Client Cases

I. Use Case - Connect:

Note: May verify IP address Before Hand, final design not decided
Valid IP address and Server online

- Pre-condition: Client is open and Server is online
- Post-condition: Connection with Server complete, message sent “Connection Successful” (Just a general example for the message, whatever gets the point across)
- Story
    - Input for IP address is “192.158.1.38”
    - Connection with server attempted
        - Successful
    - Message sent
    - End of Story
- Next use case(s) - Login, Registration, Recover Password, Disconnect (if the user wanted to immediately Disconnect for some reason)

Valid address and Server offline

- Pre-condition: Client is open
- Post-condition: Connection with Server incomplete, message sent “Server Offline or Incorrect IP Address”
- Story
    - Input for IP address is “localhost”
    - Connection with server attempted
        - Not successful
    - error message received
    - Check if format of ip address follows (0 - 255).(0 - 255).(0 - 255).(0 - 255) or is “localhost”
        - It does
    - Formulate and send message
    - End of story
- Next use case(s) - Connect (can’t move on without proper connection)

Invalid IP address

- Pre-condition: Client is open

- Post-condition: Connection with Server incomplete, message sent “Server Offline or Invalid IP Address”
- Story
    - Input for IP address is “i like pi”
    - Connection with server attempted
        - Not successful
    - error message received
    - Check if format of ip address follows (0 - 255).(0 - 255).(0 - 255).(0 - 255) or is “localhost”
        - It doesn’t
    - Formulate and send message
    - End of story
- Next use case(s) - Connect

II. Use Case - Disconnect:

Client/Server connection severed (goal) with [X], User isn’t Logged in

- Pre-condition: Client is connected to Server, and User isn’t Logged in
- Post-condition: Client is disconnected from Server, then Client is terminated
- Story
    - The user attempts to terminate the client via the [X] in the top Corner
    - Client is disconnected from server
    - Client is terminated
    - End of story
- Next use case(s) - N/A (Client is Closed)
- Question: Should we handle an accidental disconnected

Client/Server connection severed (goal) with [X], User is Logged in

- Pre-condition: Client is connected to Server, User is Logged in
- Post-condition: User is Logged out, Client is disconnected from Server, then Client is terminated
- Story
    - The user attempts to terminate the client via the [X] in the top Corner
    - The User is logged out of Client
    - Client is disconnected from server
    - Client is terminated
    - End of story
- Next use case(s) - N/A (Client is Closed)

Client/Server connection severed (goal) and User isn't Logged in

- Pre-condition: Client is connected to Server, User isn't Logged in
- Post-condition: Client is disconnected to Server, message sent "Disconnected"
- Story
    - The presses "disconnect" button on current window
    - Client is disconnected from server
    - Client switches back to "Connect" window
    - End of story
- Next use case(s) - Connect

Client/Server connection severed (goal) and User is Logged in

- Pre-condition: Client is connected to Server, User is Logged in
- Post-condition: Client is disconnected to Server, message sent "Disconnected"
- Story
    - The presses "Disconnect" button on current window
    - User is Logged out of Client
    - Client is disconnected from server
    - Client switches back to "Connect" window
    - End of story
- Next use case(s) - Connect

Client/Server connection severed (not the goal/accidental disconnect) and User isn't Logged in

- Pre-condition: Client is connected to Server, User isn't Logged in
- Post-condition: Client is disconnected to Server, message sent "Disconnected"
- Story
    - The user loses internet connection
    - Client is disconnected from server
    - Client switches back to "Connect" window
    - End of story
- Next use case(s) - Connect

Client/Server connection severed (not the goal/accidental disconnect) and User is Logged in

- Pre-condition: Client is connected to Server, User is Logged in
- Post-condition: Client is disconnected to Server, message sent "Disconnected"
- Story
    - The user loses internet connection
    - The user is signed out

    - Client is disconnected from server
    - Message is sent
    - Application is closed
    - End of story
- Next use case(s) - Connect (if they didn't directly close the app)
- **Question: Should we ask for confirmation, when attempting to close the client or disconnect from the server?**
- **Question: Should we close the app when disconnected from the server or internet, or should we have a static option to reconnect to the server where the rest of the client functions are disabled, or something of the like? This is for a sudden disconnect from the server.**

III. Use Case - Registration:

Valid input for username, password, and email.

- Pre-condition: Connection with Server complete
- Post-condition: The user is logged in and the user enters the App and tries to register, message sent "Account Registered"
- Story
    - User inputs username "krillinIsHere"
    - User inputs password "I1ovemyse1fs0mekri11"
    - User inputs email address "HiThere@gmail.com"
    - Check if email already exists
        - It doesn't
    - Check if username already exists
        - It doesn't
    - Write info to User Database
    - Message sent
    - End of story
- Next use case(s) - Login, Disconnect

Username - already registered

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid username message
- Story
    - User inputs username "krillinIsHere"
    - User inputs password "I1ovemyse1fs0mekri11"
    - User inputs email address "HiThere@gmail.com"
    - Check for username and and email address database for repeats

        - There are repeats
        - Return message “Username already registered”
        - End of Story
- Next use case(s) - Registration, login, or disconnect.

Username - username field empty

- Pre-condition: Connection with Server complete
- Post-condition: Return username does not meet the requirements message
- Story
    - User inputs username “”
    - User inputs password “I1ovemyse1fs0mekri11”
    - User inputs email address “HiThere@gmail.com
    - Check if username fits the username requirements
        - It doesn’t
    - Return message “Invalid Registration - Username requirements not satisfied”
    - End of Story
- Next use case(s) - Registration, Login, or Disconnect

Email - invalid email form

- Pre-condition: Connection with Server complete
- Post-condition: Return email address does not the requirements message
- Story
    - User inputs username “krillinIsHere”
    - User inputs password “I1ovemyse1fs0mekri11”
    - User inputs email address “@#2@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It does
    - Check if the email address fits the email address requirements
        - It doesn’t
    - Return message “Invalid registration: email address does not meet the requirements”
    - End of Story

Next use case(s) - Registration, login, or disconnect

Email - already registered

- Pre-condition: Connection with Server complete

- Post-condition: Return invalid email address message
- Story
    - User inputs username “krillinIsHere”
    - User inputs password “I1ovemyse1fs0mekri11”
    - User inputs email address “HiThere@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It does
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered
        - There are duplicates
    - Return message “Email address already registered”
    - End of Story
- Next use case(s) - Registration, login, disconnect
- **Question: Can a single email have multiple accounts/ usernames?**

[Strong Password Enforcement]
Password - Invalid minimum Length, length less than 8.
Note: m equals minimum length

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid password message
- Story
    - User inputs username “IlikePi37”
    - User inputs password “a”
    - User inputs email address “birdsRcool@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It doesn’t, length is less than 8
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered
        - It hasn’t been
    - Return message “Invalid password, too short”
    - End of Story
- Next use case(s) - Registration, login, disconnect

- **Question: What is the min length for password (and is there one for usernames)? Is there a max length for both? Any more requirements for usernames and passwords aside from the ones provided in the Specifications?**

Password - Contains invalid Characters

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid password message
- Story
    - User inputs username “NiceO7”
    - User inputs password “ideemtheeworthyP©134”
    - User inputs email address “rocks@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It doesn’t, the symbol © is invalid
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered
        - It hasn’t been
    - Return message “Invalid password, invalid symbol present”
    - End of Story
- Next use case(s) - Registration, login, disconnect

Password - Invalid minimum Length and Contains invalid Characters

- Pre-condition: Connection with Server complete
- Post-condition: Return password does not meet requirements message
- Story
    - User inputs username name benistherealk1ng
    - User inputs password “©”
    - User inputs email address “rocks@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It doesn’t, length is less than m and the symbol © is invalid
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered
        - It hasn’t been

    - Return message “Invalid password - invalid minimum length and invalid symbol present”
    - End of Story
- Next use case(s) - Registration, login, or disconnect

Password - Doesn’t Include numbers

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid password message
- Story
    - User inputs username “Nice07”
    - User inputs password “password”
    - User inputs email address “birdsRcool@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It doesn’t, password does not contain numbers
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered
        - It hasn’t been
    - Return message “Invalid password: Password must include a combination of numbers and letters”
    - End of Story
- Next use case(s) - Registration, login, or disconnect

Password - Doesn’t include letters

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid password message
- Story
    - User inputs username “Nice07”
    - User inputs password “1234567”
    - User inputs email address “birdsRcool@gmail.com”
    - Check if username fits the username requirements
        - It does
    - Check if password fits the password requirements
        - It doesn’t, password does not contain letters
    - Check if the email address fits email address requirements
        - It does
    - Check if the email address has already been registered

        - It hasn't been
    - Return message "Invalid password - Password must include a combination of numbers and letters"
    - End of Story
- Next use case(s) - Registration, login, or disconnect

IV. Use Case - Login:

Login Successful

- Pre-condition: Connection with Server complete
- Post-condition: Display next screen/message after login
- Story
    - The user types in the username "ben s"
    - The user types the password
    - The user clicks "Log in"
    - End of Story
- The enter username and passwords are checked against the passwords in the server and the login count is reset to 0.
- Next use case(s) - Logout or disconnect

Invalid Username

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid login message
- Story
    - The user types in the username ""
    - The user types the password
    - The user clicks "Log in"
        - The enter username and passwords are checked against the passwords in the server.
        - Username is not in the server database
    - Return message generic "Invalid login" message
    - End of Story
- Next use case(s) - Registration, password recovery, login, or disconnect

Invalid Password (Attempt 0-3)

- Pre-condition: Connection with Server complete
- Post-condition: Return invalid login message and number of attempts left before account gets locked
- Story
    - The user types in the username "ben s"

- The user types an incorrect password
- The user clicks “Log in”
  - Searches for the entered username in User Database, then compares entered password to the password in the User Database
  - Username is in the server database; password is not in the server database
- Increment login count by 1
- Return message generic “Invalid login” message
- End of Story

- Next use case(s) - Registration, password recovery, login, or disconnect

Invalid Password (Attempt 4/ Locked out)

- Pre-condition: Connection with Server complete
- Post-condition: Return locked account message
- Story
  - The user types in the username “ben s”
  - The user types an incorrect password
  - The user clicks “Log in”
    - The enter username and passwords are checked against the passwords in the server.
    - Username is in the server database; password is not in the server database
  - Return message “This account is now locked. ”
  - End of Story
- Next use case(s) - Registration, password recovery, login, or disconnect

V. Use Case - Logout:

Logout Successful

- Pre-condition: Connection with Server complete
- Post-condition: Return successful logout message
- Story
  - User clicks Logout
  - Checks if connection is still active
    - It is
  - Unconfirmed changes for account are cleared and cancelled
  - Client returns to login page, all fields empty
  - End of Story
- Next use case(s) - Register, password recovery, Login, or Disconnect
- **Question: Should we ask for confirmation before logging off?**

Already disconnected *possible safety net

- Pre-condition: The user has been disconnected from
- Post-condition: Return status message
- Story
    - User clicks Logout
    - Checks if connection is still active
        - It isn't
    - Return status message "Already Disconnected"
    - Move to entry page
    - End of Story
- Next use case(s) - Connect

VI. Use Case - Password recovery:

Valid username

- Pre-condition: The user clicks on Forgot your password
- Post-condition: And email containing the password is sent to the user
- Story
    - The user enters their username, "Guardians_of_Galaxy"
    - The user name is checked to see if it is valid
        - It is
    - Is account Locked
        - No
    - The password is sent to the address on record
    - Return a general attempt message "If username is valid, your password has been sent to the email on record"
      End of story
- Next use case(s) - Login, Disconnect

Valid username and Account is Locked

- Pre-condition: The user clicks on Forgot your password
- Post-condition: And email containing the password is sent to the user
- Story
    - The user enters their username
    - The user name is check to see if it is valid
        - It is
    - Is account Locked
        - It is
    - Database locked user with a new valid password - "tempPass12343"

- New valid password is sent to email on record
- Return a general attempt message “If username is valid, your password has been sent to the email on record”
  End of story

- Next use case(s) - Disconnect, Password recovery, Login

Invalid username

- Pre-condition: The user clicks on Forgot your password
- Post-condition: Return confirmation message
- Story
  - User enters a username
  - Server checks for username in User Database
    - Doesn’t exist
  - Return a general attempt message “If username is valid, your password has been sent to the email on record”
  - End of story
- Next use case(s) - Disconnect, Password recovery, Login

VII. Use Case - Change password:

Valid *old password*, *new password* field and *verify new password* field match

- Pre-condition: Connection with server complete
- Post-condition: Password is successfully changed
- Story
  - The user clicks on change password from being logged in
  - The user types in new password “comSciC1A22”
  - The user retypes new password “comSciC1A22”
  - The user clicks submit
  - Check if old password matches address on record
    - Yes
  - Check if new passwords match
    - Yes
  - Check if passwords meet strong password enforcement criteria
    - Yes
  - Write new password to data structure
  - The user is informed of the change
  - End of story
- Next use case(s) - Logout, Disconnect, Change Password (No limit to how many times the user can change their password)

- **Question: Where can you Change password?**
- **What should happen after the user completes the password change. Will they immediately be returned to the main screen or will they be informed via a pop up, will the screen tell them and time out returning them to the main screen/ will they be a button to return to main after they read the message. Are you automatically logged in after changing your password?**

Old password and new password match

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “12345”
    - The user verifies and retypes new password “12345”
    - Check database to see if new password matches old password
        - It does
    - Return message: “Invalid password change: old and new password cannot be the same”
    - End of story
- Next use case(s) - Logout, Disconnect, Change Password

New password field and *verify new password* field don’t match

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “comSciC1A22”
    - The user verifies and retypes new password “cHeese789”
    - Check database to see if new password matches old password
        - It does not
    - Check if new password and verify new password field match
        - It does not
    - Return message: “Invalid password change: new password must match verify new password field”
    - End of story
- Next use case(s) - Logout, disconnect, Change password

[Strong Password Enforcement] - [Similar to Registration]
Invalid minimum Length, less than *8* characters
Note: m is the minimum length

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “a”
    - Check if new password is valid
        - It isn’t
    - Return message “Invalid password change: new password is invalid - minimum length requirement not met”
    - End of story
- Next use case(s) - Logout, Disconnect, Change Password

Contains invalid Characters

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “asd12345©”
    - The user verifies and retypes new password “asd12345©”
    - Check database to see if new password matches old password - it does
    - Return message: “The new password contains invalid characters”
    - End of story
- Next use case(s) - Logout, disconnect, change password

Invalid minimum Length and Contains invalid Characters

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “©”
    - The user verifies and retypes new password “©”
    - Check database to see if new password matches old password - it does

    - Return message: “The new password contains invalid characters” and exceeds minimum length.
    - End of story
- Next use case(s) - Logout, disconnect, change password

Doesn’t Include numbers

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “asdFGHIu”
    - The user verifies and retypes new password “asdFGHIu”
    - Check database to see if new password matches old password - it does
    - Check to see if new password meets the password requirements - it doesn’t
    - Return message: “Invalid password change: Password must contain numbers and letters”
    - End of story
- Next use case(s) - Logout, disconnect, change password

Doesn’t include letters

- Pre-condition: Connection with server complete
- Post-condition: Return invalid password change message
- Story
    - User is logged in
    - User clicks on change password
    - The user types in new password “1234567”
    - The user verifies and retypes new password “1234567”
    - Check database to see if new password matches old password
        - It does
    - Check to see if new password meets the password requirements
        - It doesn’t
    - Return message: “Invalid password change: Password must contain numbers and letters”
    - End of story
- Next use case(s) - Logout, disconnect, change password

B. Server Cases

I. Use Case - Admin Queries

Requests/Queries for user information (Number of registered users, Number of logged in users, Which users are logged in, Which users are locked out, Number of connected users, any requests related to the System Database, etc.)
System Completely functional

- Pre-condition: Server, System Database, and User Database are functioning as expected
- Post-condition: Retrieve requested information
- Story
    - Admin makes a request to see which users have been locked out
    - Is Server online
        - It is
    - Is User Database functional (data not corrupted)
        - It is
    - Server checks User Database for users who are locked out or logged in and collects references to their information (ie. usernames, emails, all info, etc.)
    - Server returns a list of users have been locked out of their accounts or a list of users that are logged in.
    - End of Story
- Next use case(s) - Admin Commands
- **Question: Will Admin have the privilege to delete accounts? If not, how should we handle inactive accounts?**
- **Question: Should there be an option to export query data to an excel or text file?**

System Failure Present

- Pre-condition: Server, System Database, User Database, or a combination are malfunctioning
- Post-condition: Return message of failure, and where failure occurred
- Story
    - Admin makes a request to see the number of registered users
    - Is Server online
        - It isn’t
    - Return Message “Server has not been started”
- Next use case(s) - Admin Commands

C. Database Cases

I. Use Case - Database Functions

Search Attempt Successful

- Pre-condition: Server is online and Database is healthy
- Post-condition: Desired information is sent to client application or admin
- Story
    - User attempts to register information under the username “ReinhartIsGreat”
    - Is Server online?
        - It is
    - Is Database working?
        - It is
    - Search for accounts with username “ReinhartIsGreat” in database
        - Match found
    - Message send “Username already registered”
    - End of Story
- Next use case(s) - Database commands

Search Attempt Failure

- Pre-condition: Server is offline or Database is corrupted or unreachable
- Post-condition: Return message of failure, and where failure occurred
- Story
    - Admin makes a request to see all users currently connected
    - Is Server online?
        - It is
    - Is Database working?
        - It isn’t
    - Search command isn’t applied
    - Return message “Problems with accessing User Database”
    - End of Story
- Next use case(s) - Database commands

Insertion Attempt Successful

- Pre-condition: Server is online and Database is healthy
- Post-condition: Message of success send
- Story

    - User attempts to register information under the username "ReinhartIsTheGreatest"
    - Is Server online?
        - It is
    - Is Database working?
        - It is
    - Search for accounts with username "ReinhartIsTheGreatest" in database
        - Match not found
    - New User data is inserted to Database
    - Message send "Account Registered"
    - End of Story
- Next use case(s) - Database commands

Insertion Attempt Failure

- Pre-condition: Server is offline or Database is corrupted or unreachable
- Post-condition: Return message of failure, and where failure occurred
- Story
    - User attempts to register information under the username "timeToDine"
    - Is Server online?
        - It is
    - Is Database working?
        - It isn't
    - Insertion command isn't applied
    - Return message "Problems with Server, try again later"
    - End of Story
- Next use case(s) - Database commands

Update Attempt Successful

- Pre-condition: Server is online and Database is healthy
- Post-condition: Message of successful update
- Story
    - User confirms all information necessary to change password "L33royJenkins64" to "JenkinsTheGOAT30"
    - Is Server online?
        - It is
    - Is Database working?
        - It is
    - Find location of the current user's data
    - Update the password from to "JenkinsTheGOAT30"

    - End of story
- Next use case(s) - Database commands

Update Attempt Failure

- Pre-condition: Server is offline or Database is corrupted or unreachable
- Post-condition: Return message of failure, and where failure occurred
- Story
    - User enters one last incorrect password
    - Is Server online?
        - It is
    - Is Database working?
        - It isn't
    - Update command isn't applied
    - Send message "Problems with Server, Please try again Later"
    - End of Story
- Next use case(s) - Database commands
- **Question: What should we do if the user disconnects right before a command for the User Database is sent? should we save the command for the next time the client reconnects?**
- **Question: If the Server got the command but the database is corrupted, should the Server save it for later in a Queue?**

Extra possible Database Cases, Stories aren't included since the function isn't explicitly asked for or necessary based on the specifications.

Deletion Attempt Successful

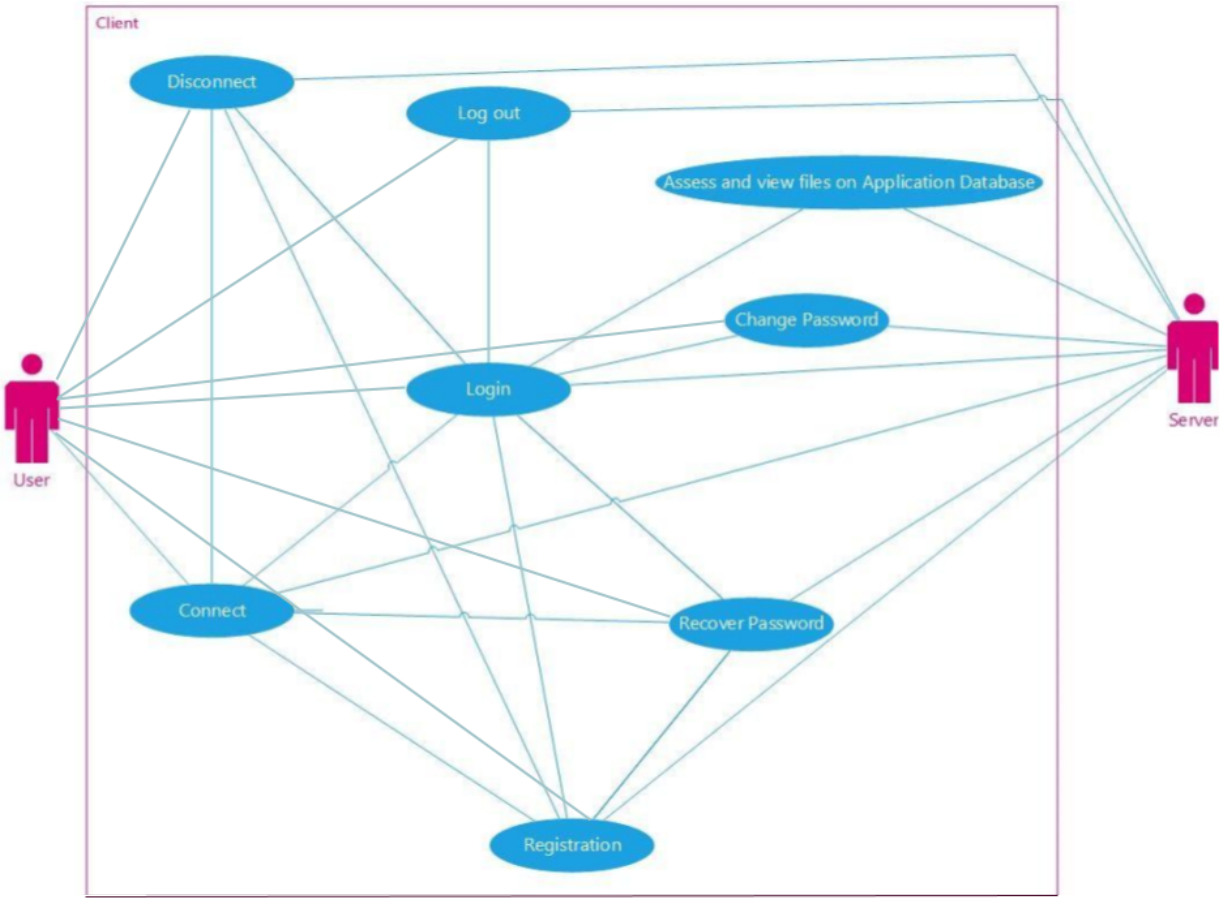
- Pre-condition: Server is online and Database is healthy
- Post-condition: Message of Success
- Story
    - N/A
- Next use case(s) - Database commands

Deletion Attempt Failure

- Pre-condition: Server is offline or Database is corrupted or unreachable
- Post-condition: Return message of failure, and where failure occurred
- Story
    - N/A
- Next use case(s) - Database commands

D. Use Case Diagram

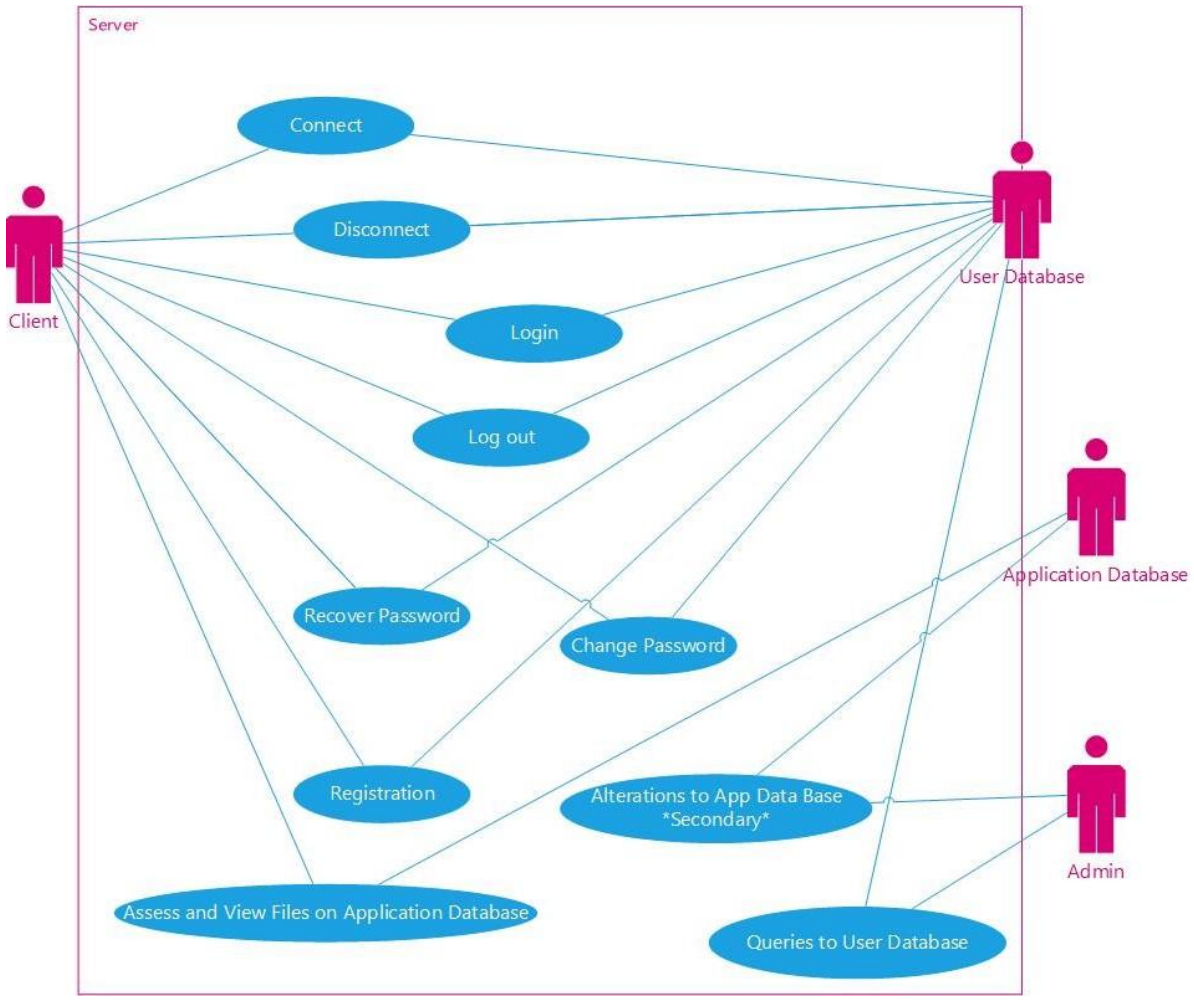
Client Diagram



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

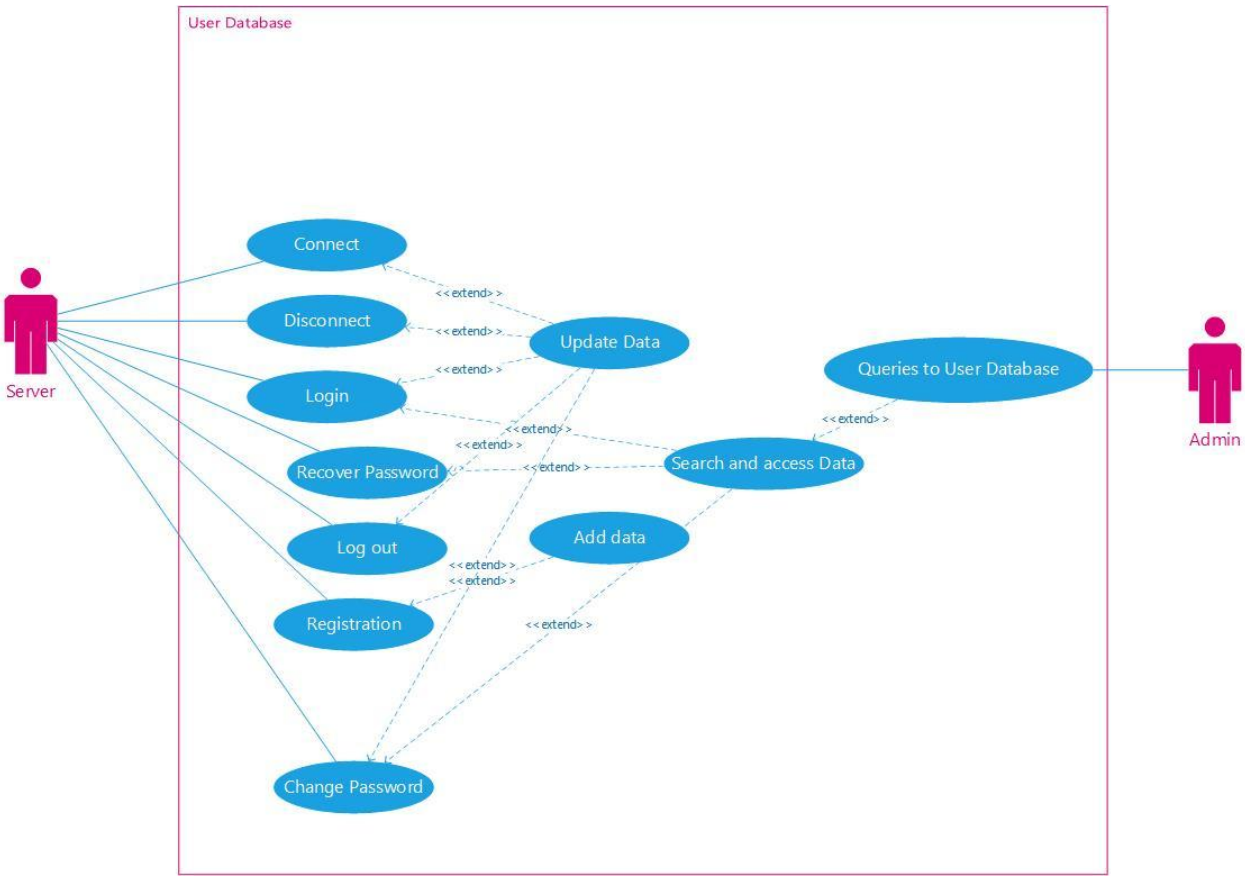
Server Diagram



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

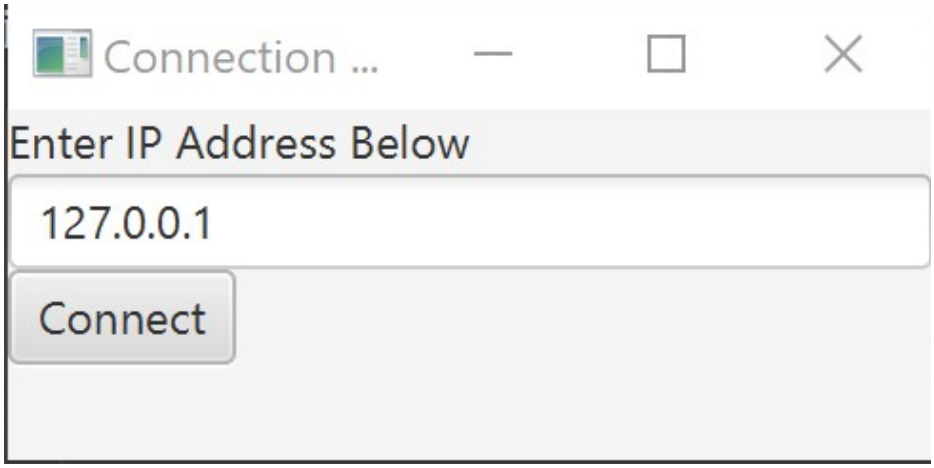
User Database Diagram



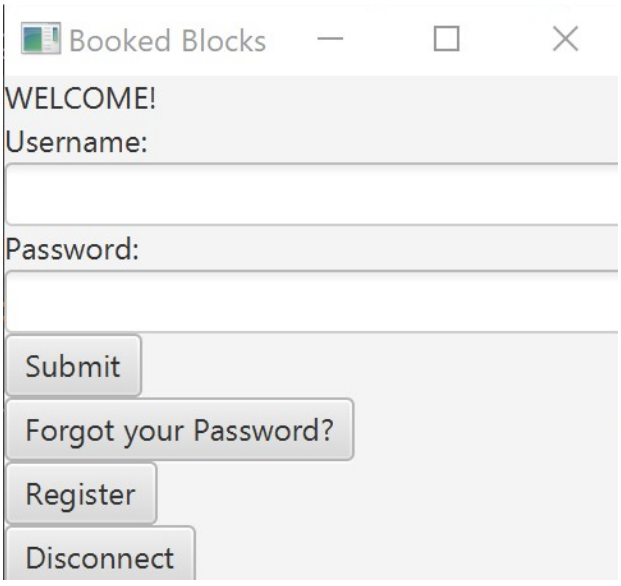
E. Current GUI Design

Client

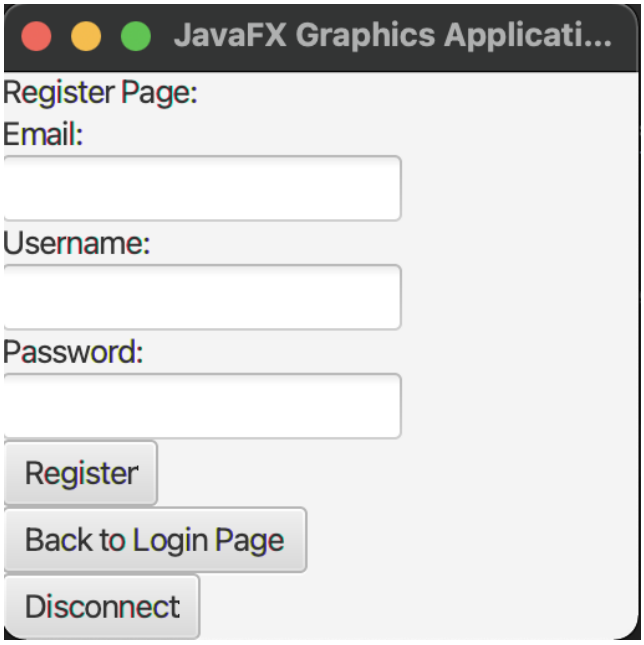
I. Connect Window (Input IP address in field, and Connect Button)



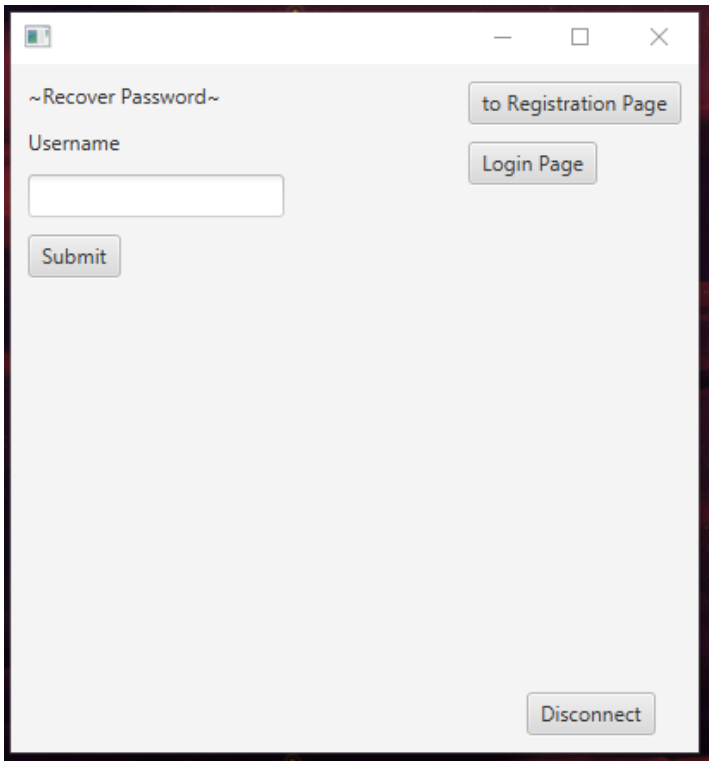
II. Login Page



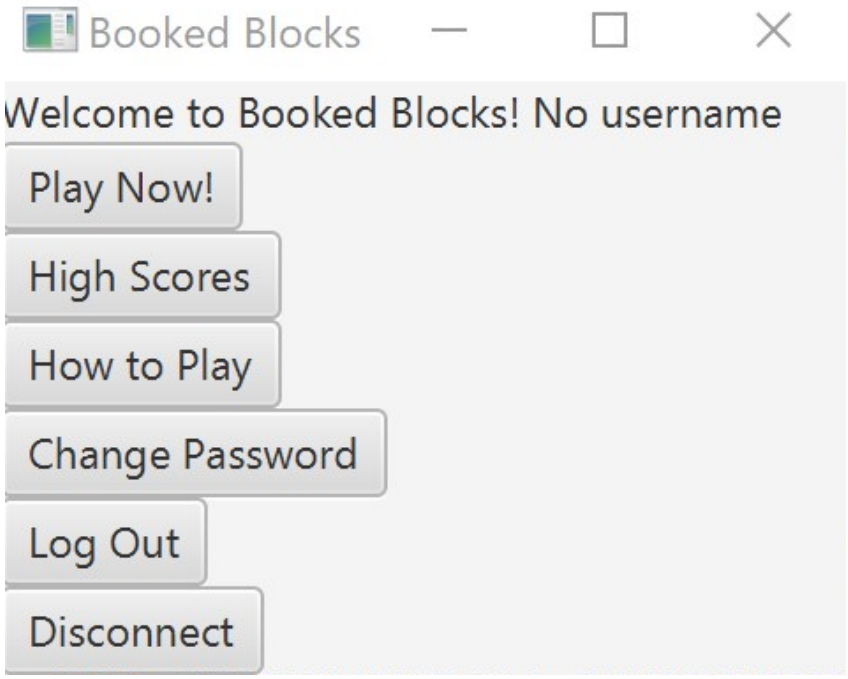
III. Registration Page



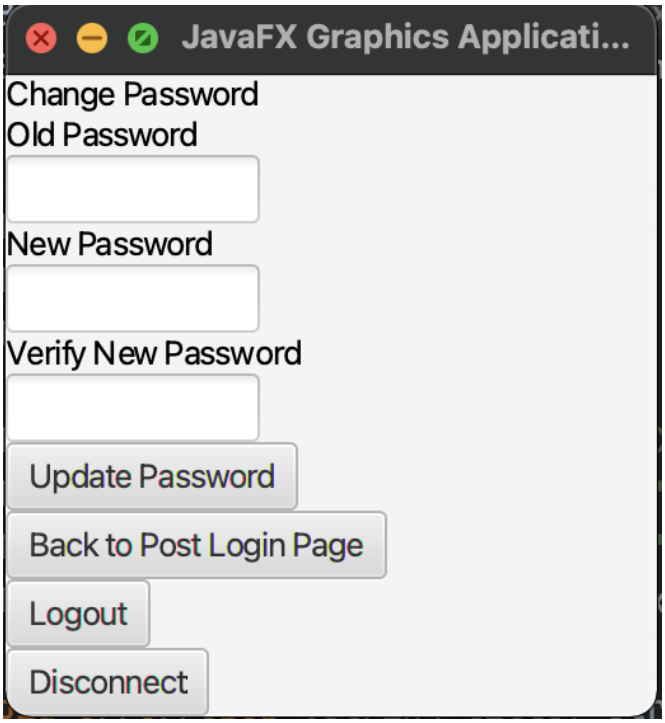
IV. Forgot/Recover Password Page (Fields for username, email address)



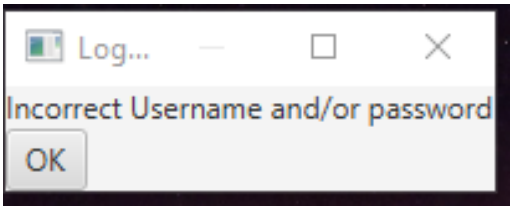
V. User Home Page



VI. Change Password Page

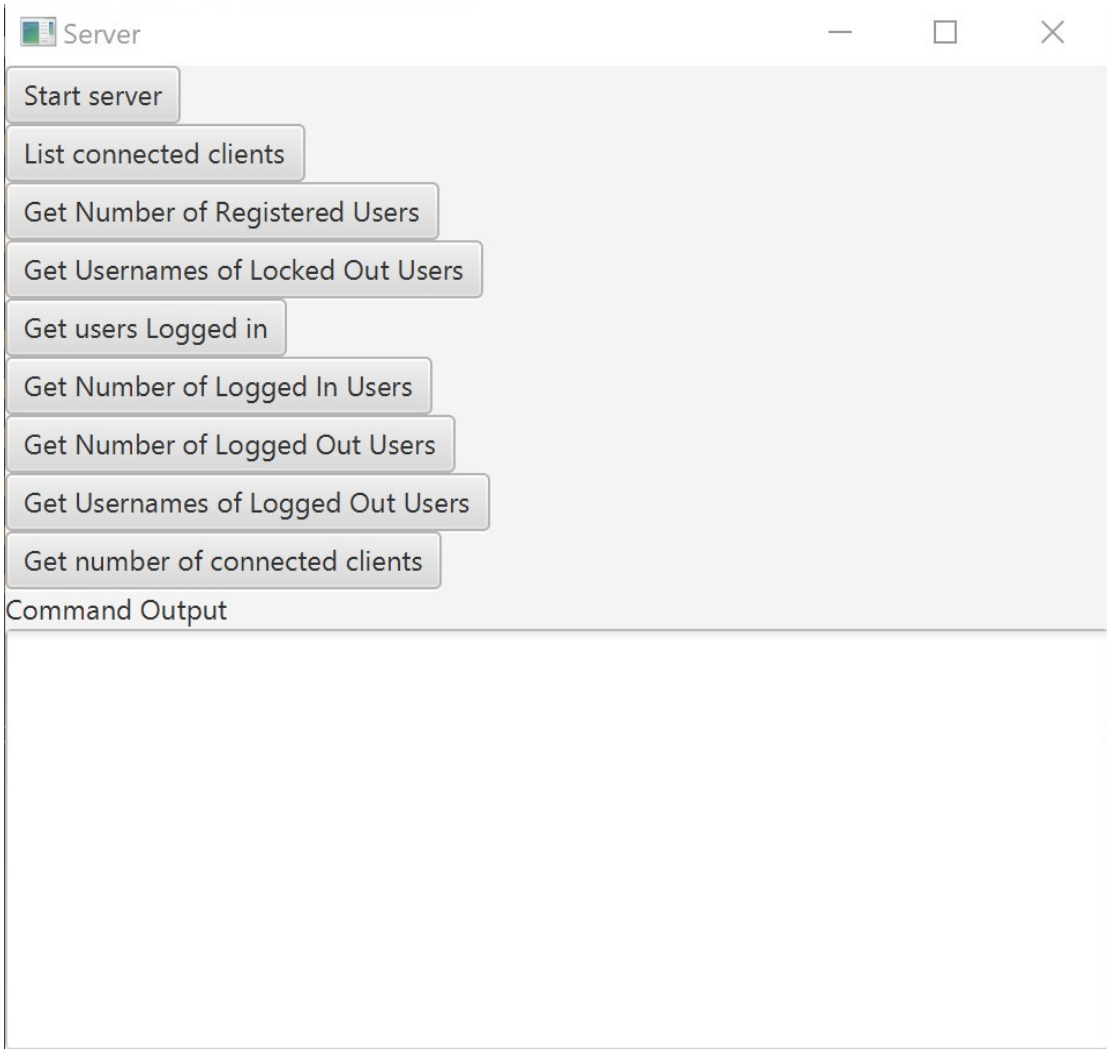


VII. Mini-Miscellaneous Message Window Example



Server

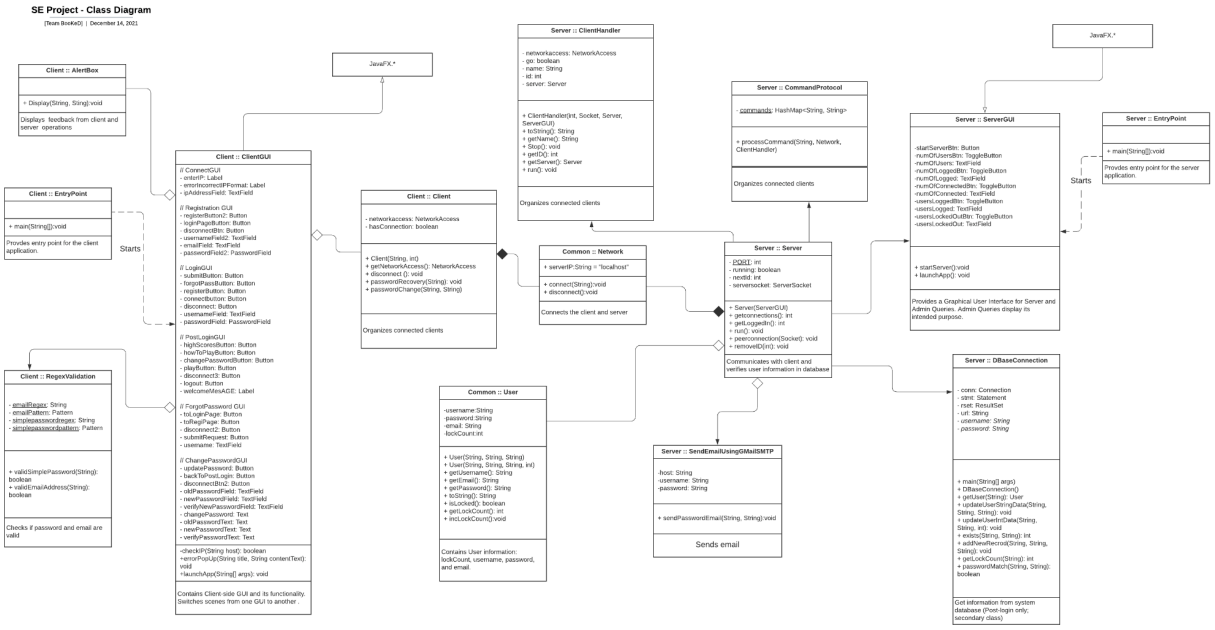
I. Server Start + Admin Query Window



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

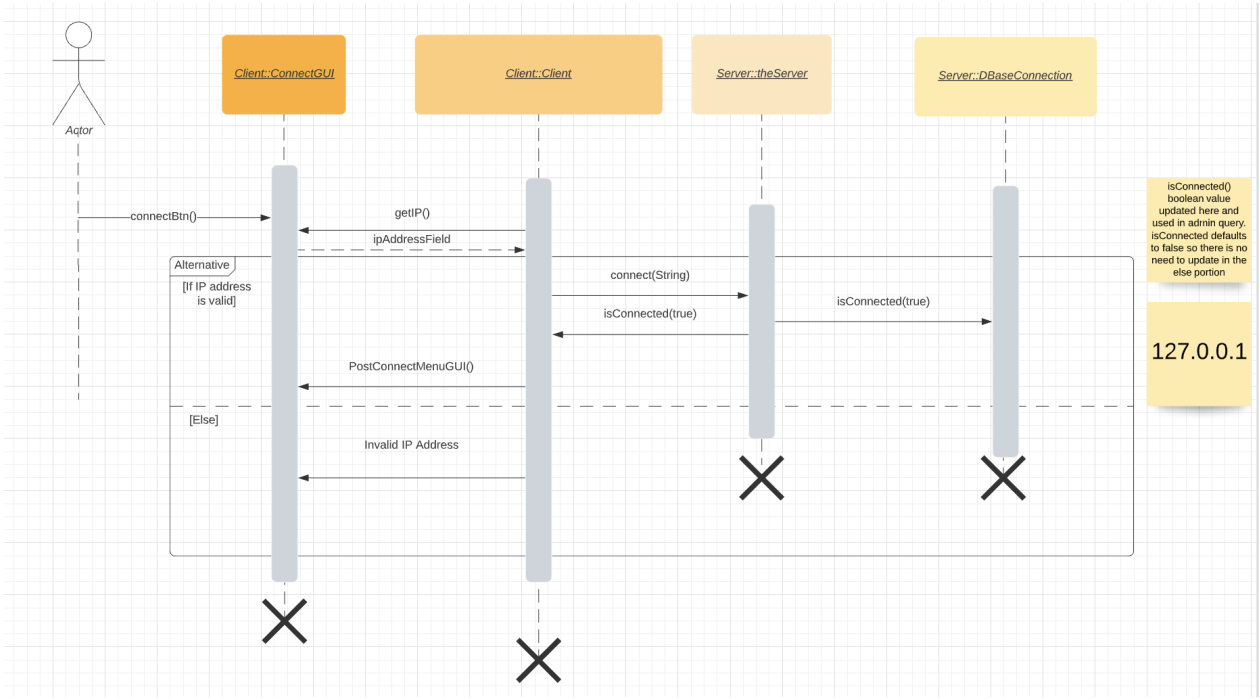
[Class Diagram]



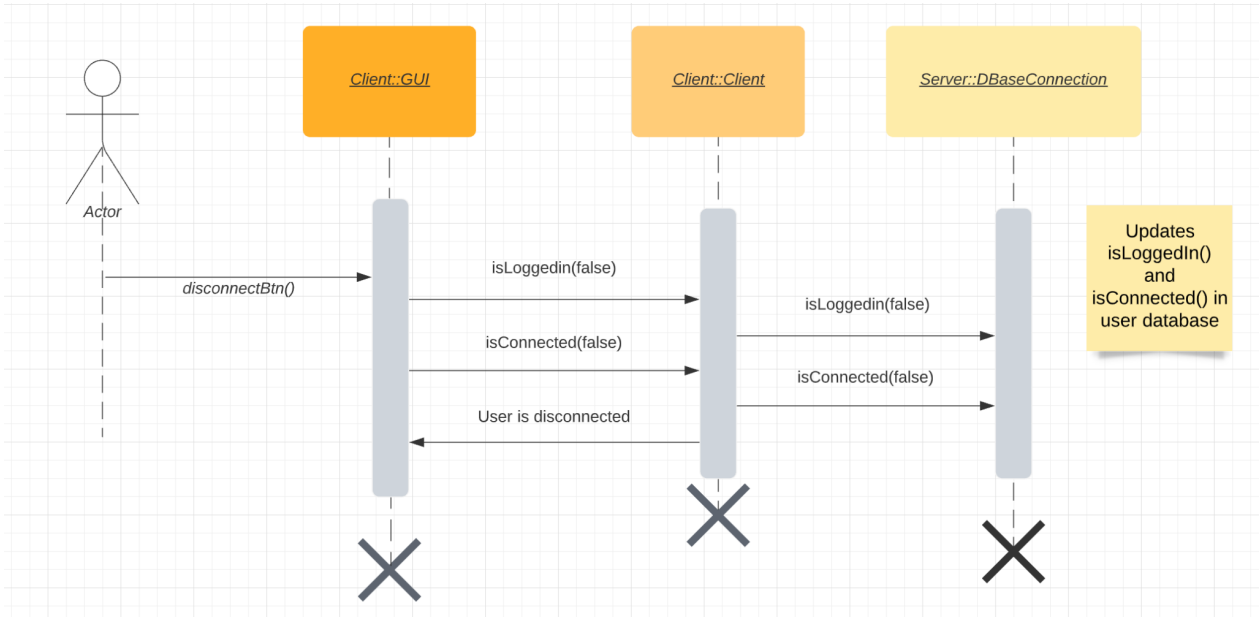
For a better quality image (https://lucid.app/publicSegments/view/06273778-8e47-4a74-9148-77052717705e/image.png)

[Sequence Diagrams]

A. Connect

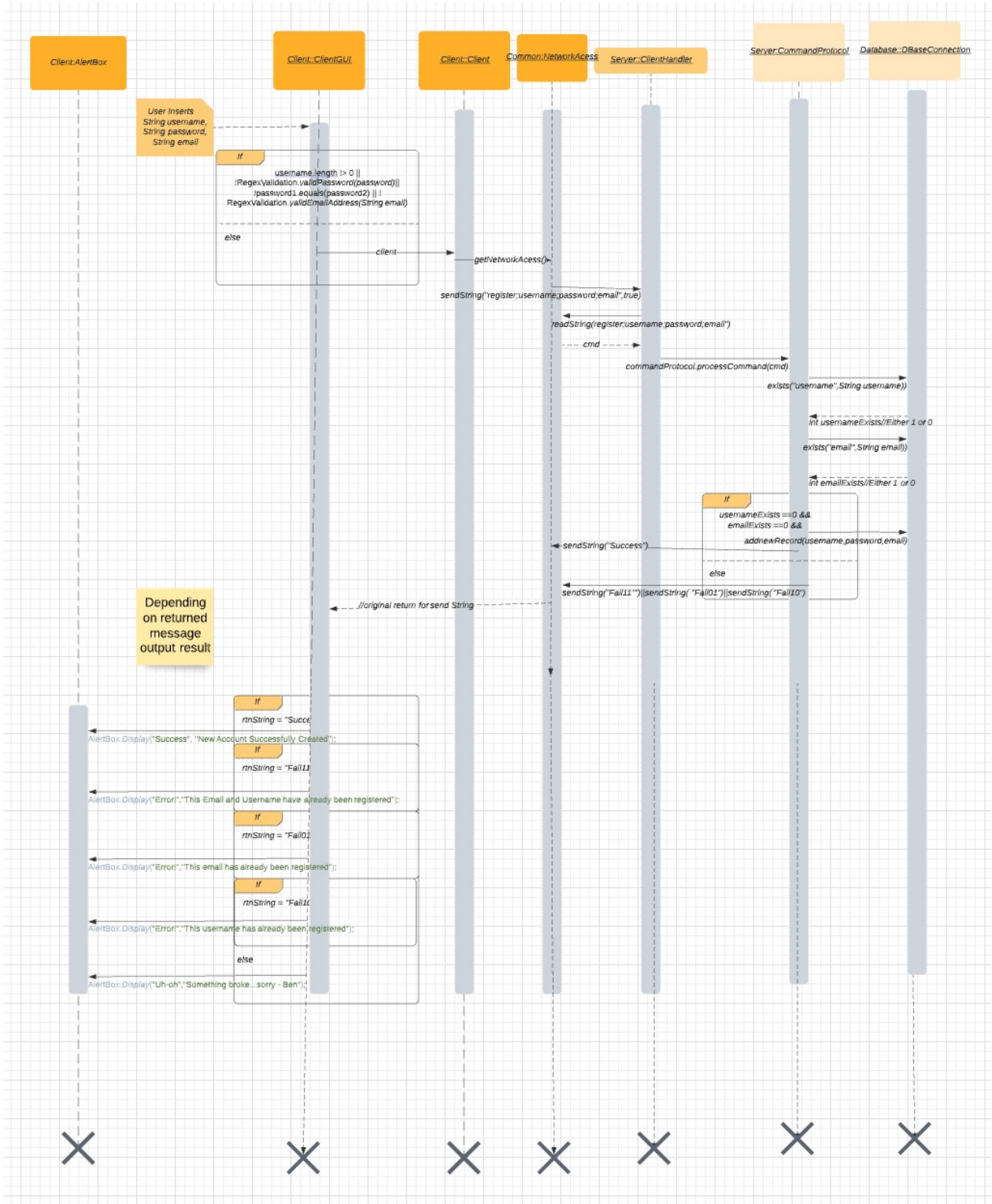


B. Disconnect

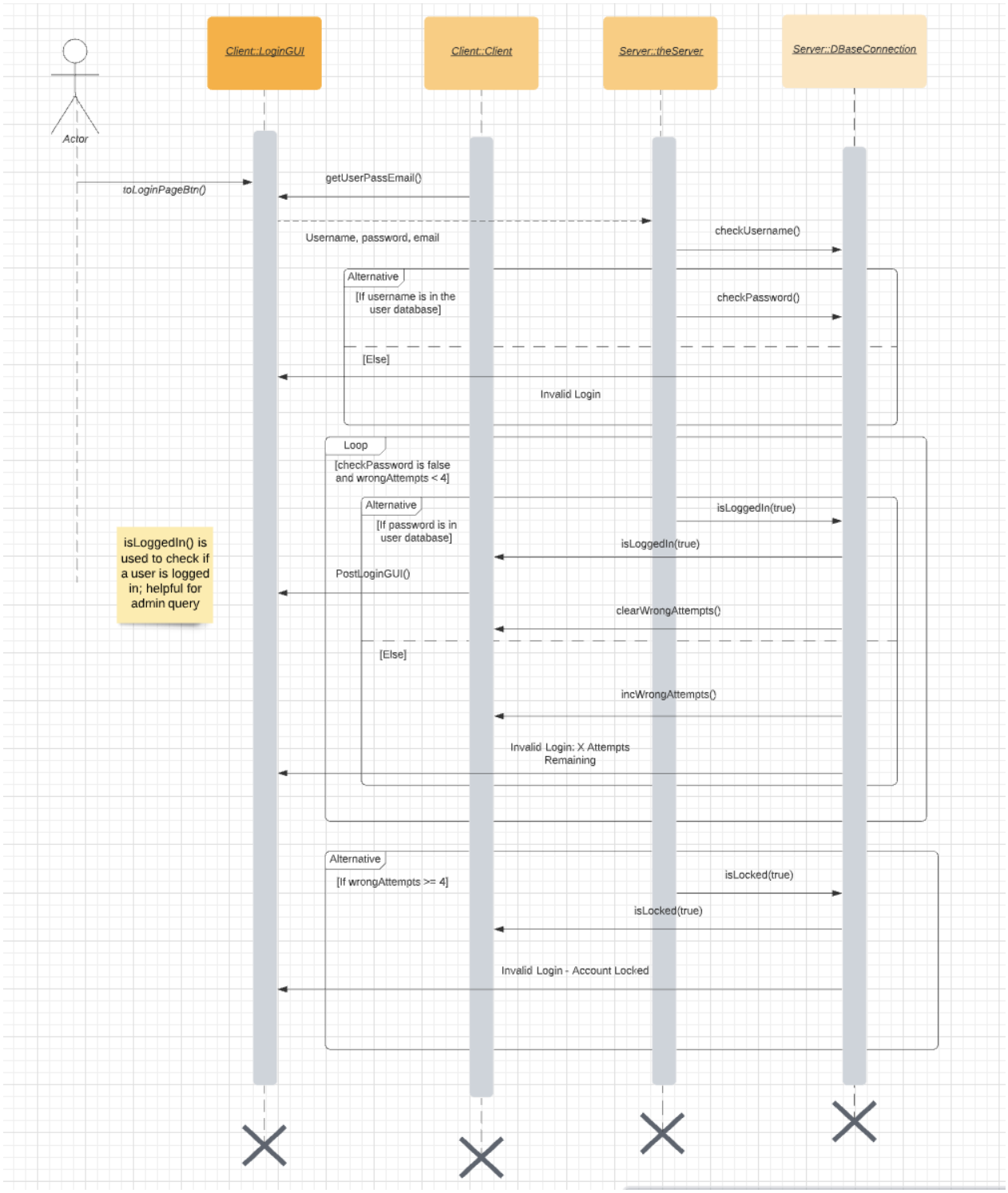


C. Registration-

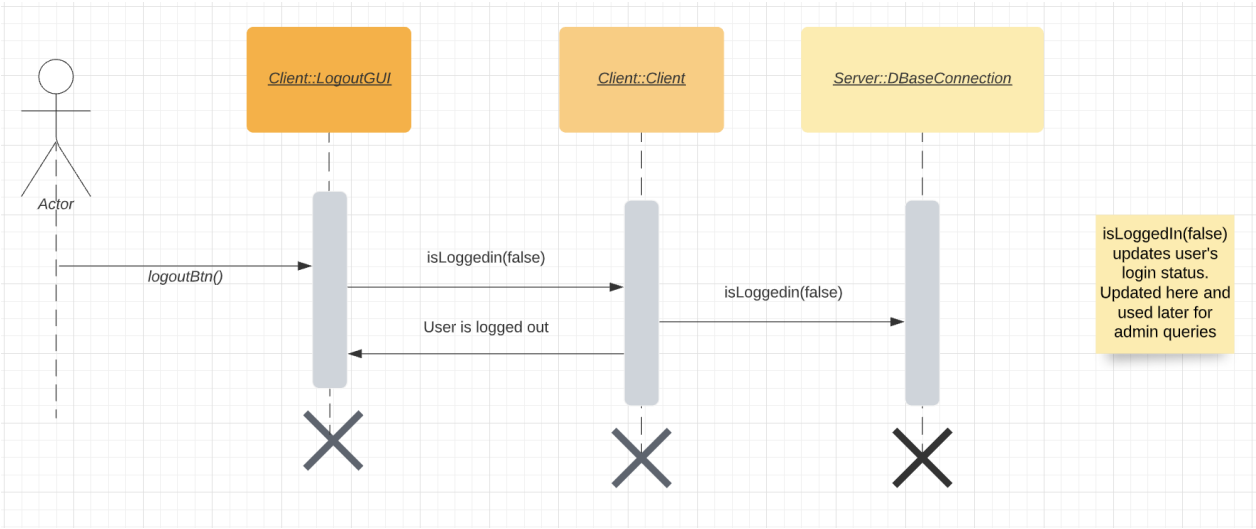
https://drive.google.com/file/d/1rqrXQiLtJg9KjmbB_uIFnedoTzMHxz80/view?usp=sharing



D. Login

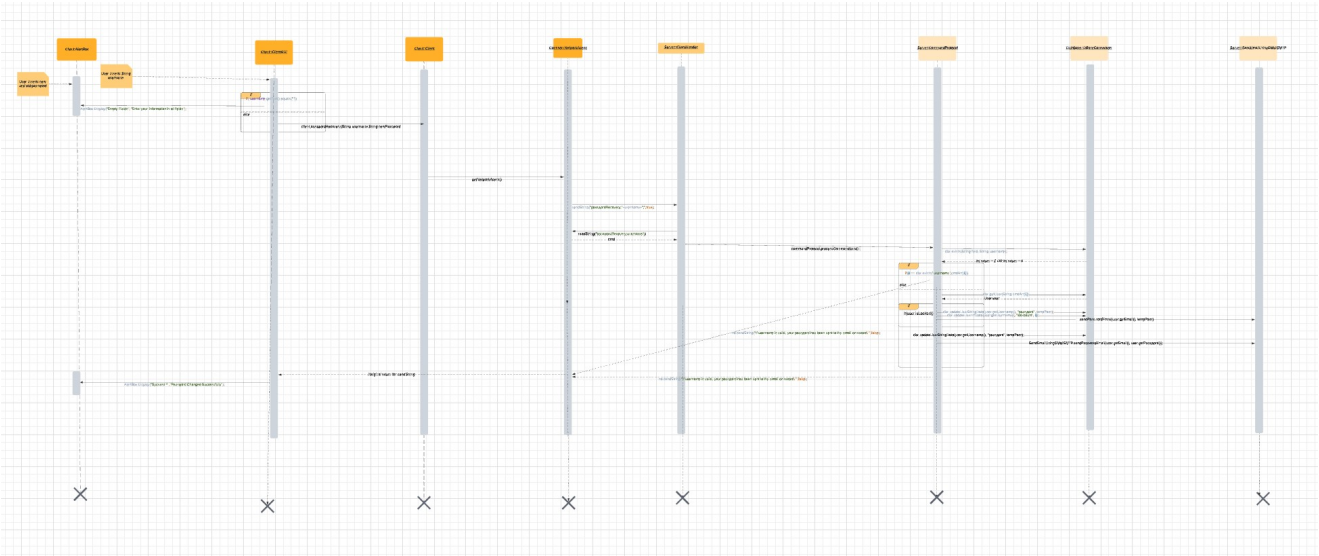

Client::LoginGUI
Client::Client
Server::theServer
Server::DBaseConnection
Actor
toLoginPageBtn()
getUserPassEmail()
Username, password, email
checkUsername()
Alternative
[If username is in the user database]
checkPassword()
[Else]
Invalid Login
Loop
[checkPassword is false and wrongAttempts < 4]
Alternative
[If password is in user database]
isLoggedIn(true)
isLoggedIn(true)
PostLoginGUI()
clearWrongAttempts()
[Else]
incWrongAttempts()
Invalid Login: X Attempts Remaining
isLoggedIn() is used to check if a user is logged in; helpful for admin query
Alternative
[If wrongAttempts >= 4]
isLocked(true)
isLocked(true)
Invalid Login - Account Locked

E. Logout



F. Password Recovery

https://drive.google.com/file/d/1rqrXQiLtJg9KjmbB_uIFnedoTzMHxz80/view?usp=sharing

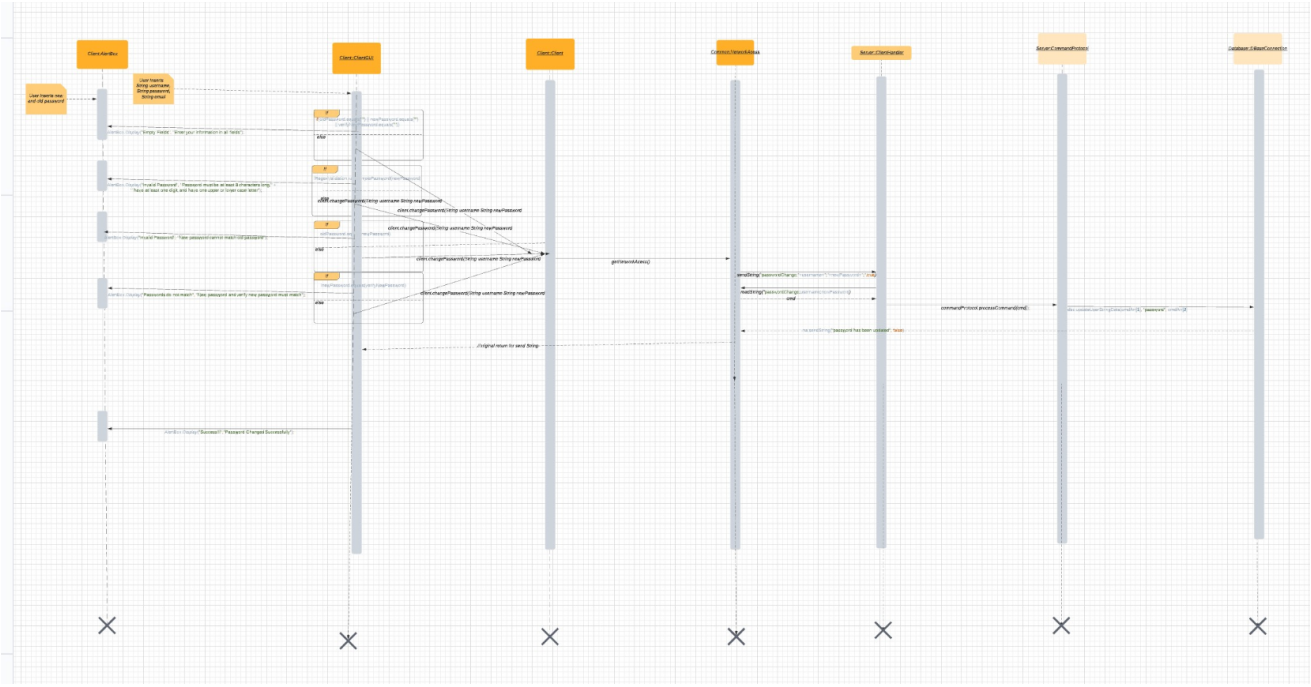


G. Change Password

https://drive.google.com/file/d/1rqrXQiLtJg9KjmbB_uIFnedoTzMHxz80/view?usp=sharing

Server/Client Architecture Documentation

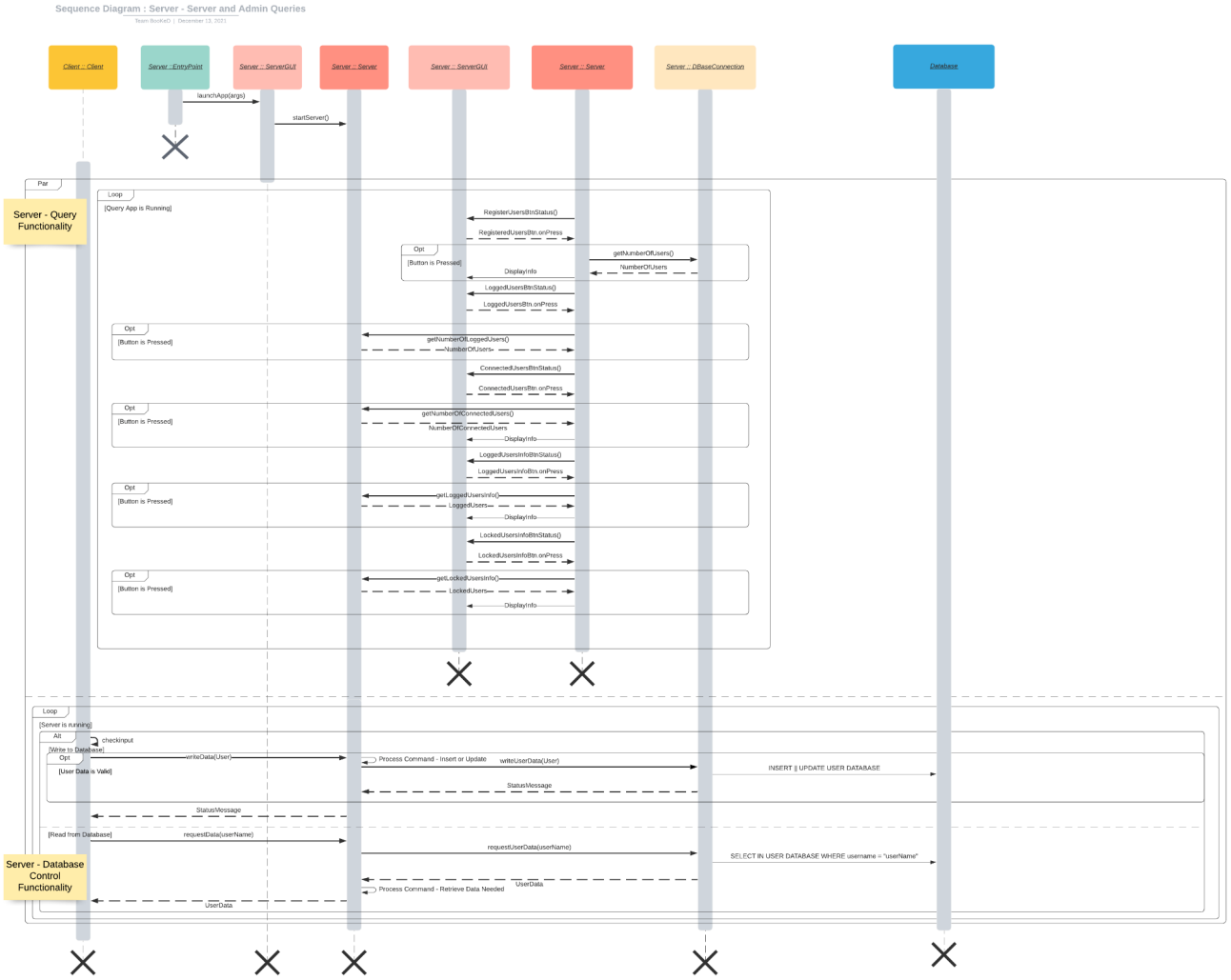
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

H. Server + Admin Queries



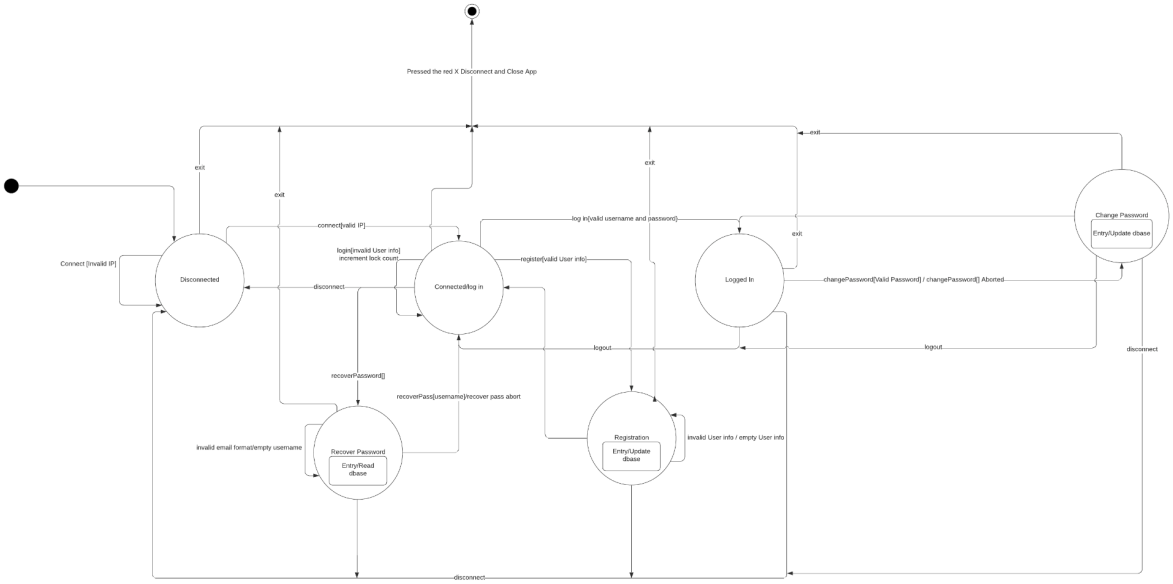
For a better quality image

https://lucid.app/publicSegments/view/845e6d16-4093-4bd2-b6d4-20a66030369d/image.png

Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

[State Machine Diagram]

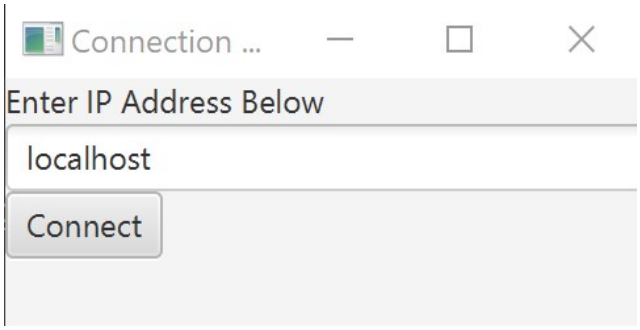


For a better quality image

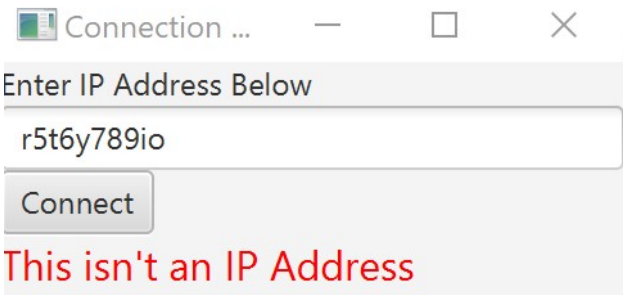
https://lucid.app/publicSegments/view/c8e32222-2518-449e-a2d5-6fa26533216e/image.png

[Program Screenshots]

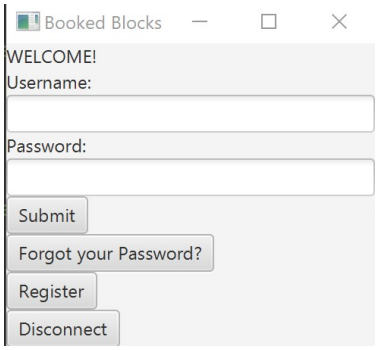
Client Side Connection



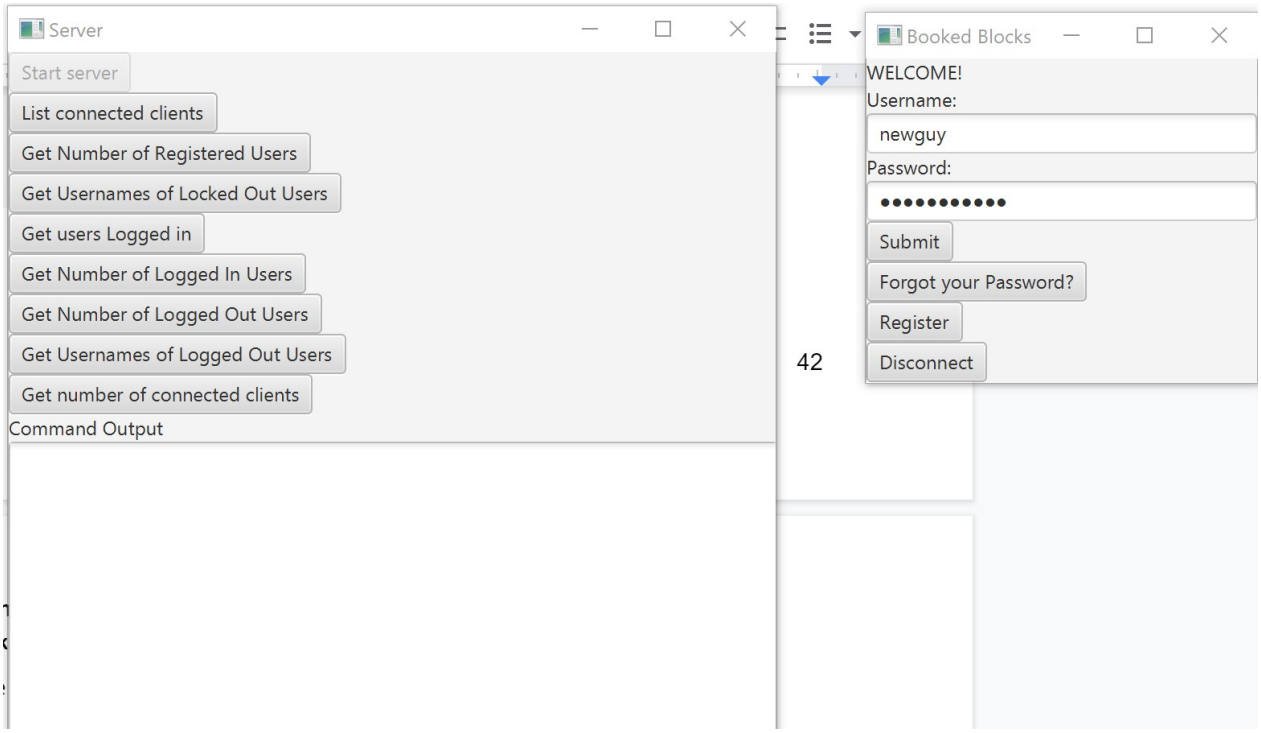
Incorrect IP

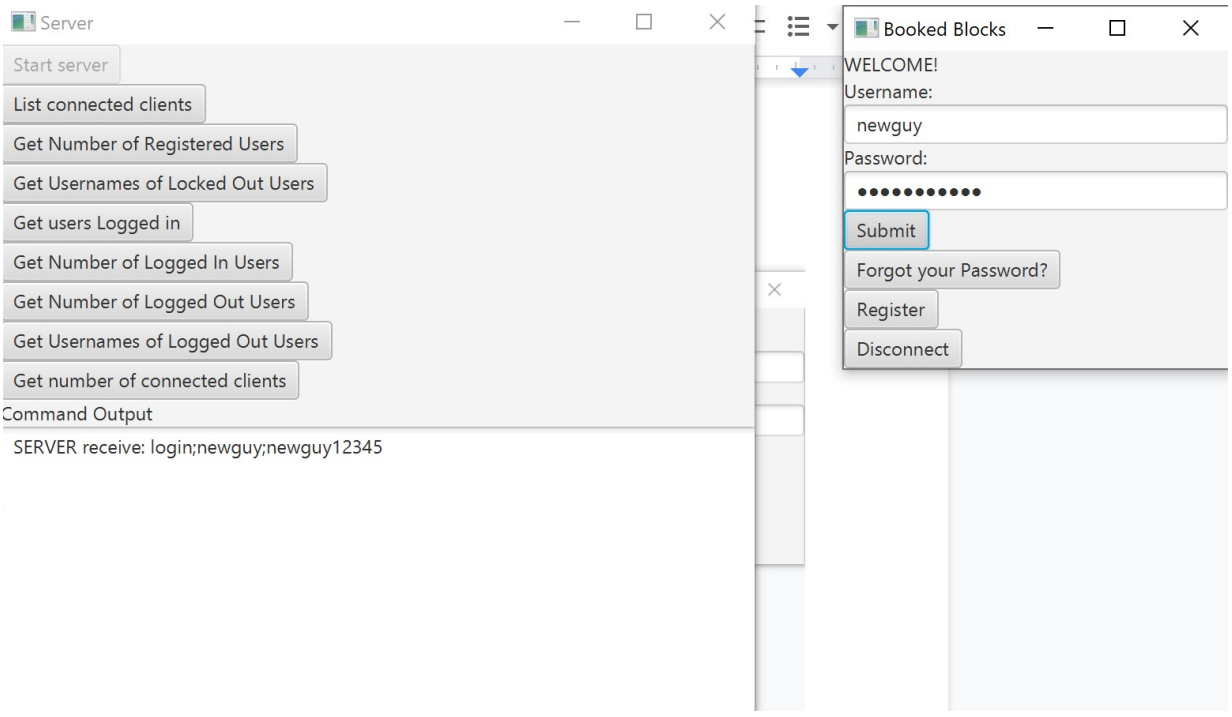


Connected / Login



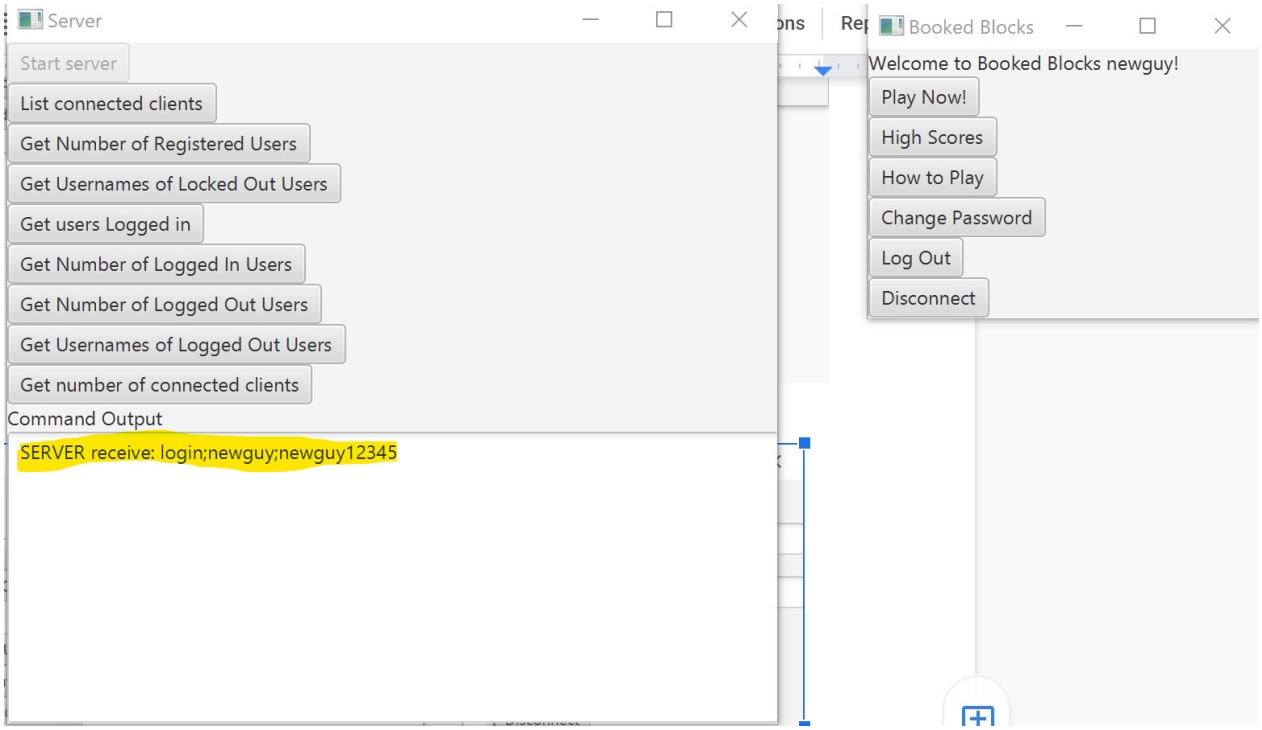
Correct Username and Correct Password

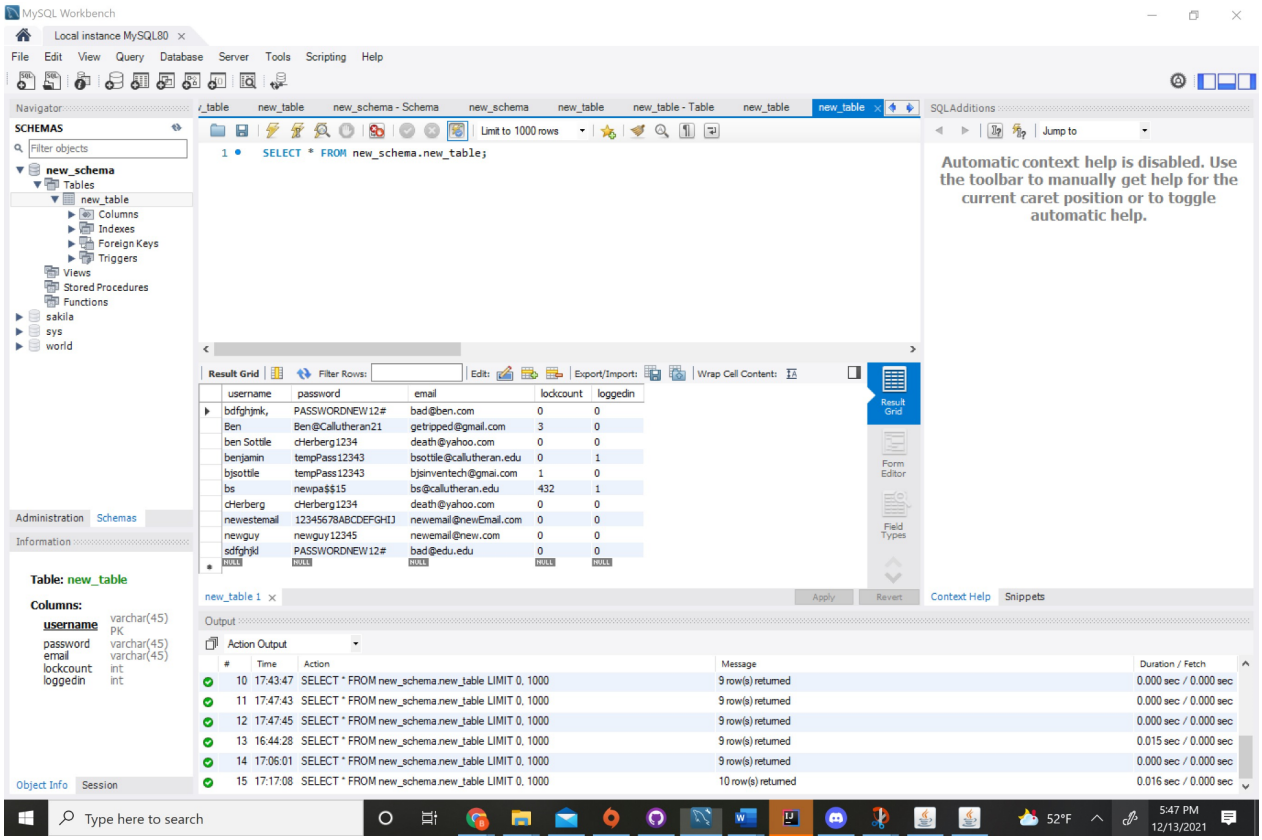




Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

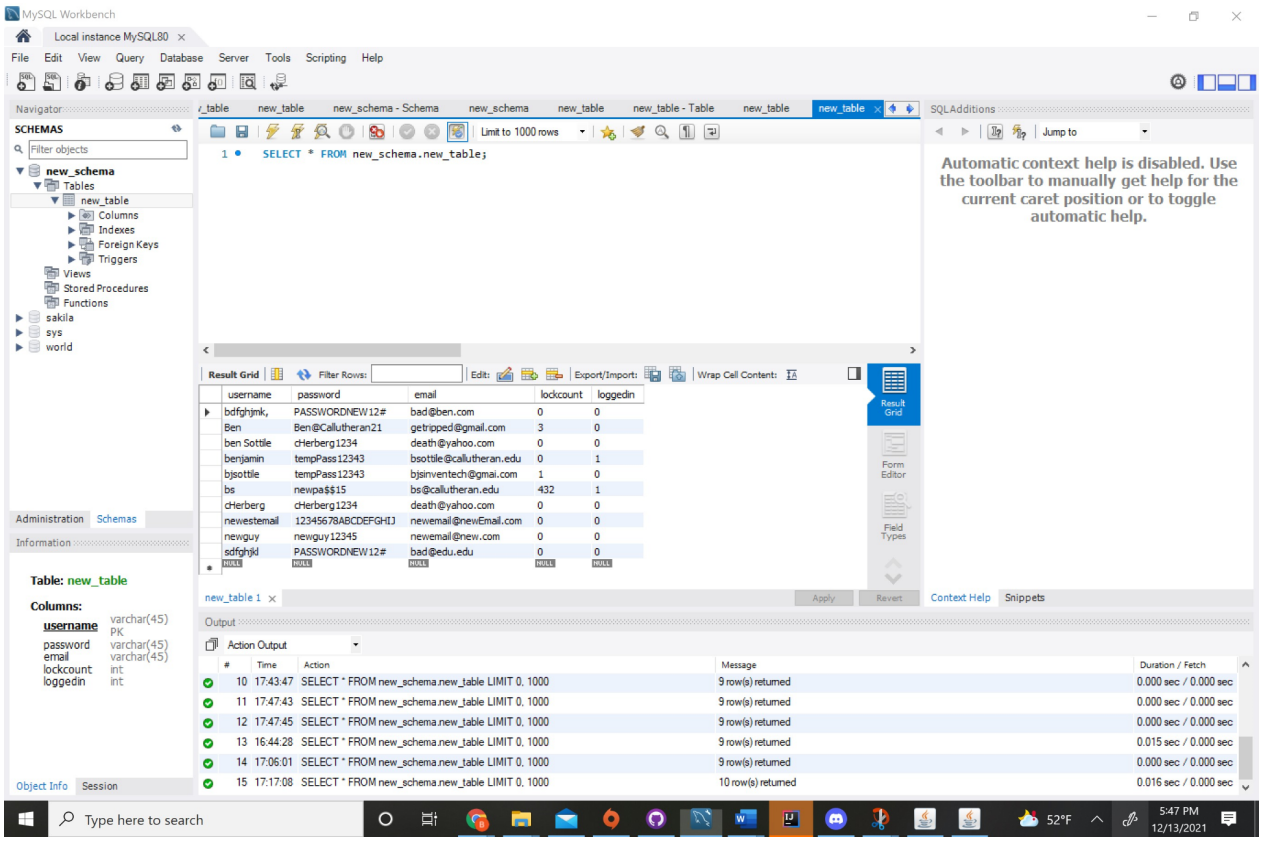




Server/Client Architecture Documentation

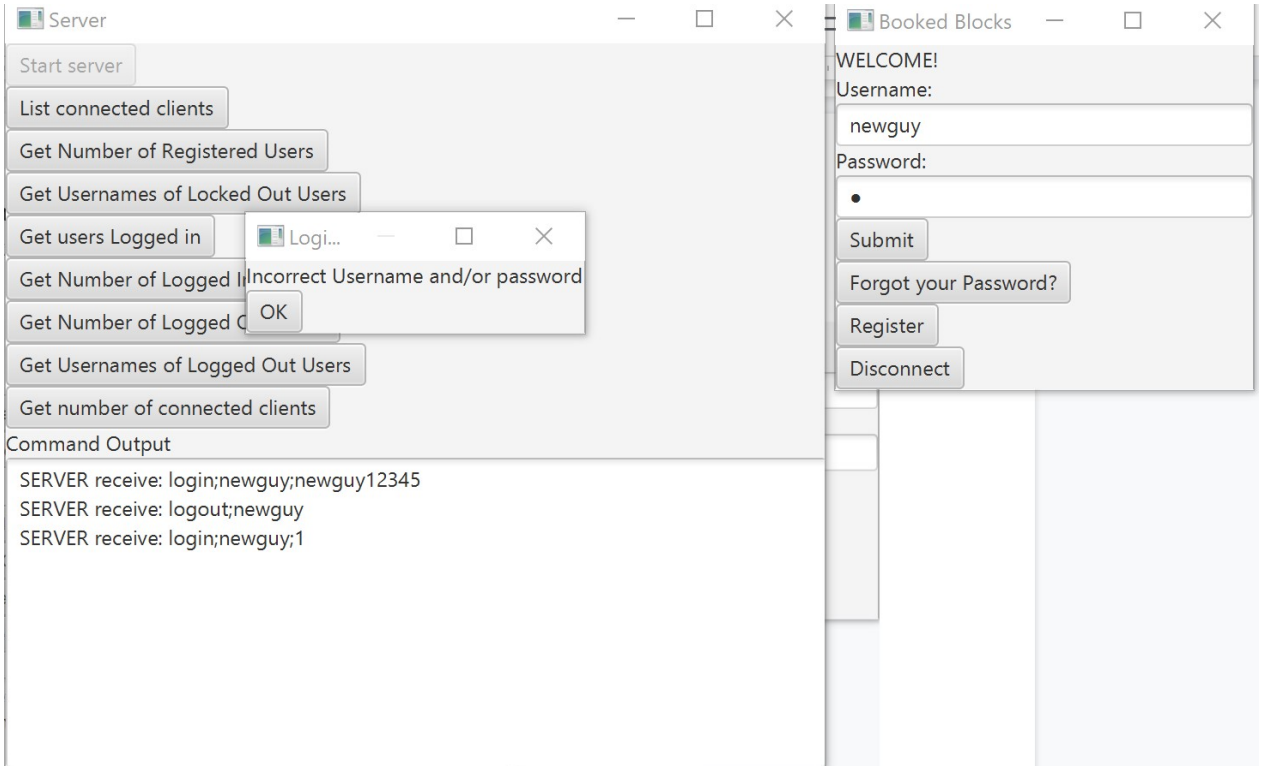
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Correct Username and incorrect Password+increment lock count



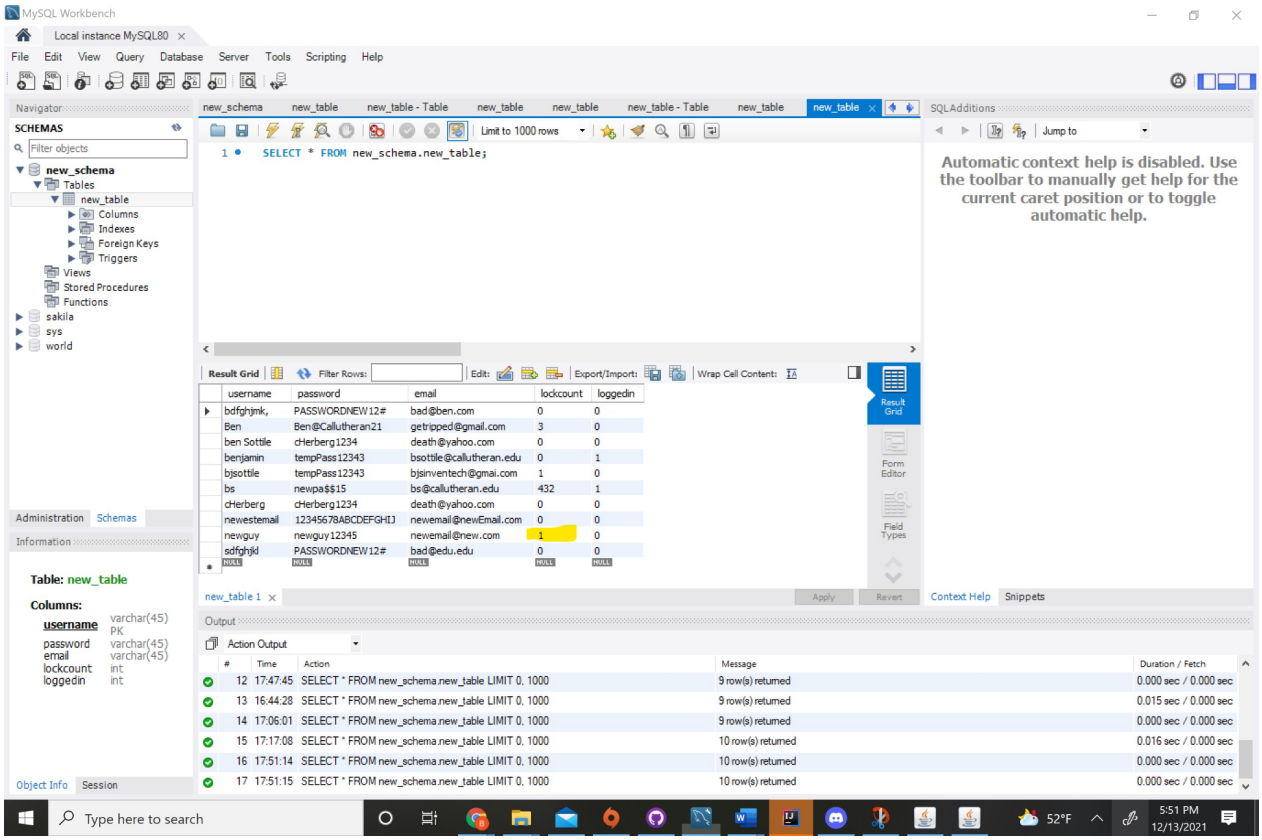
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

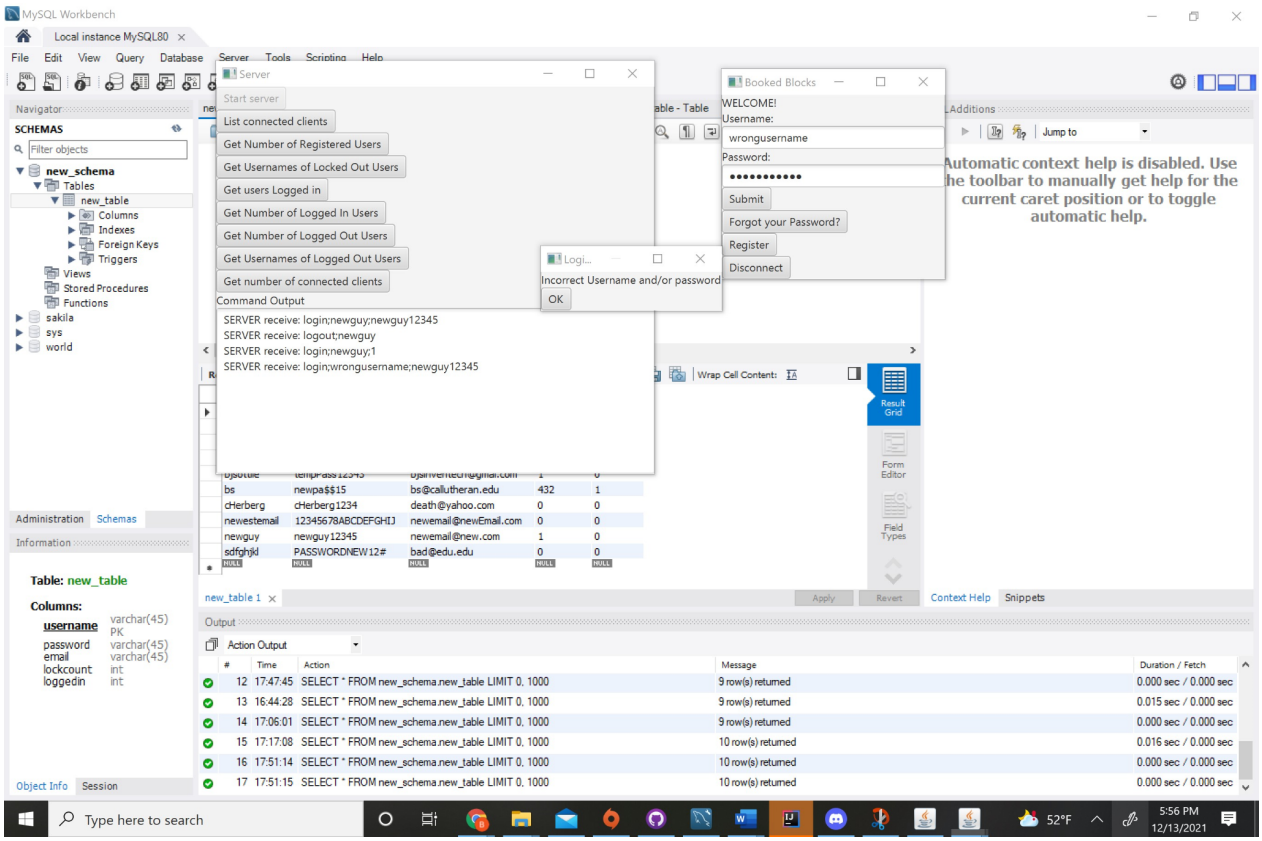


Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



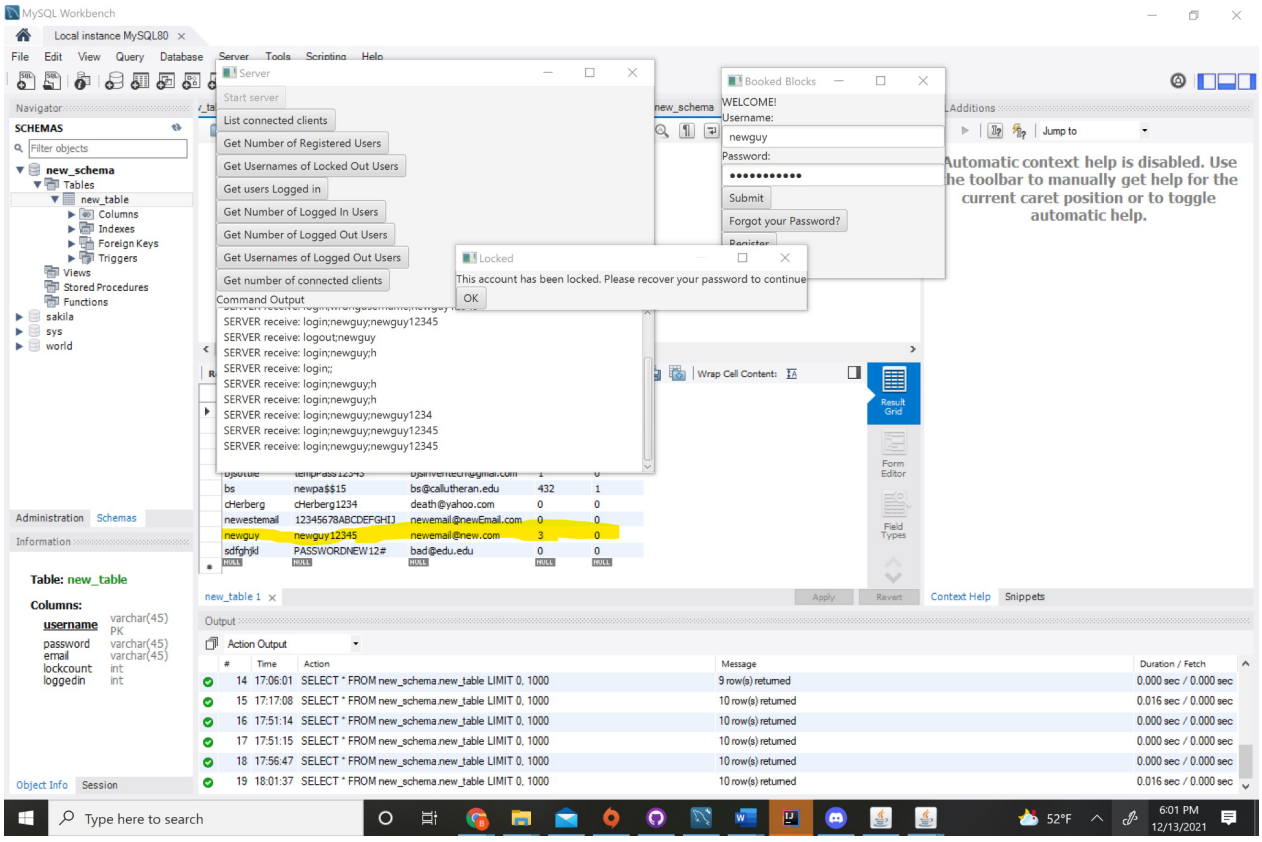
Incorrect Username



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

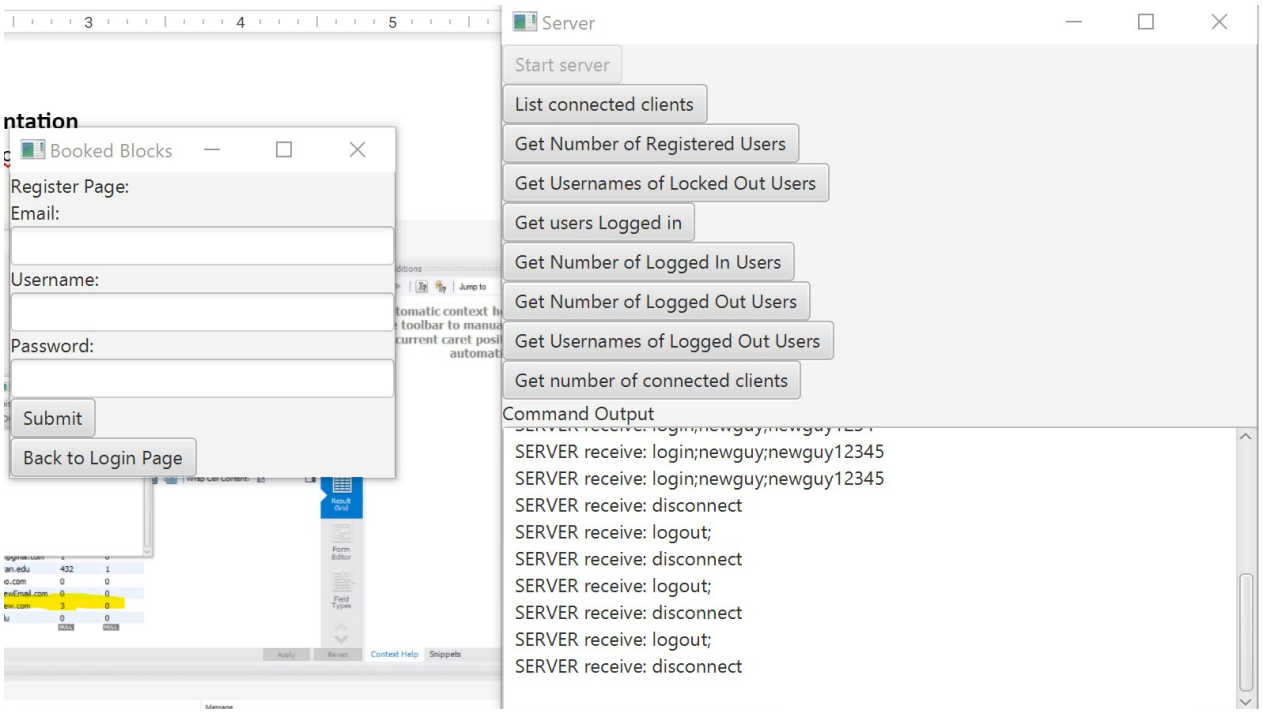
Locked Out



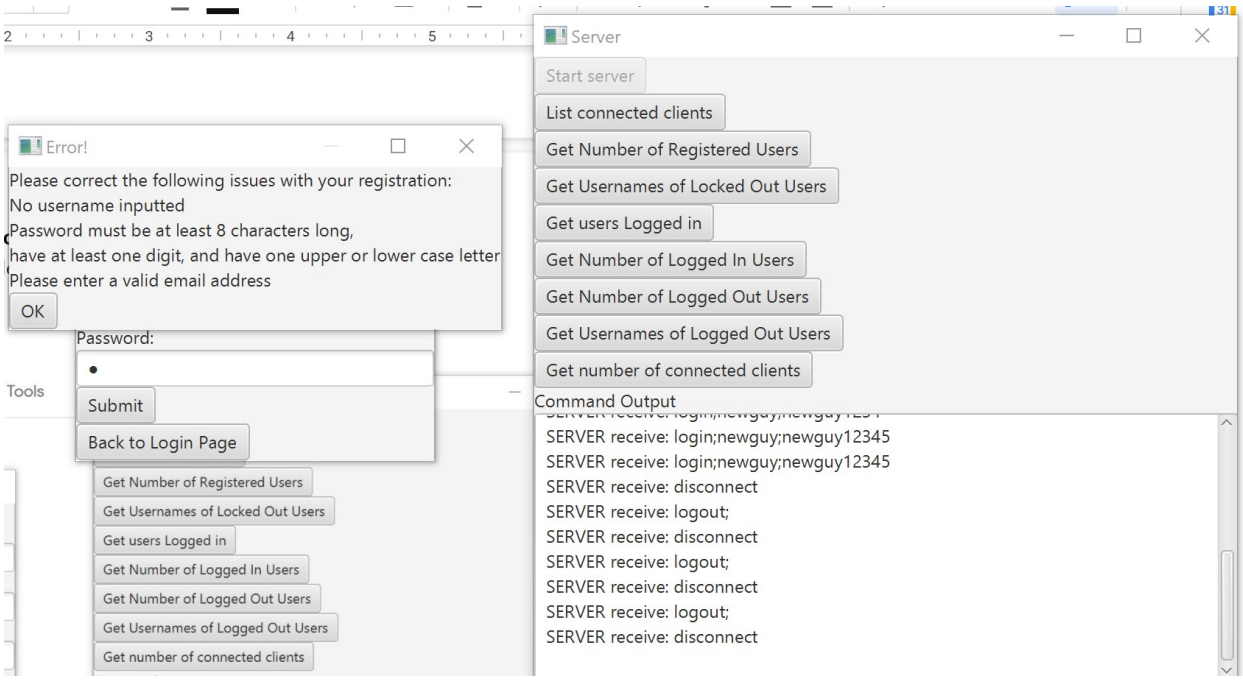
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Registration GUI

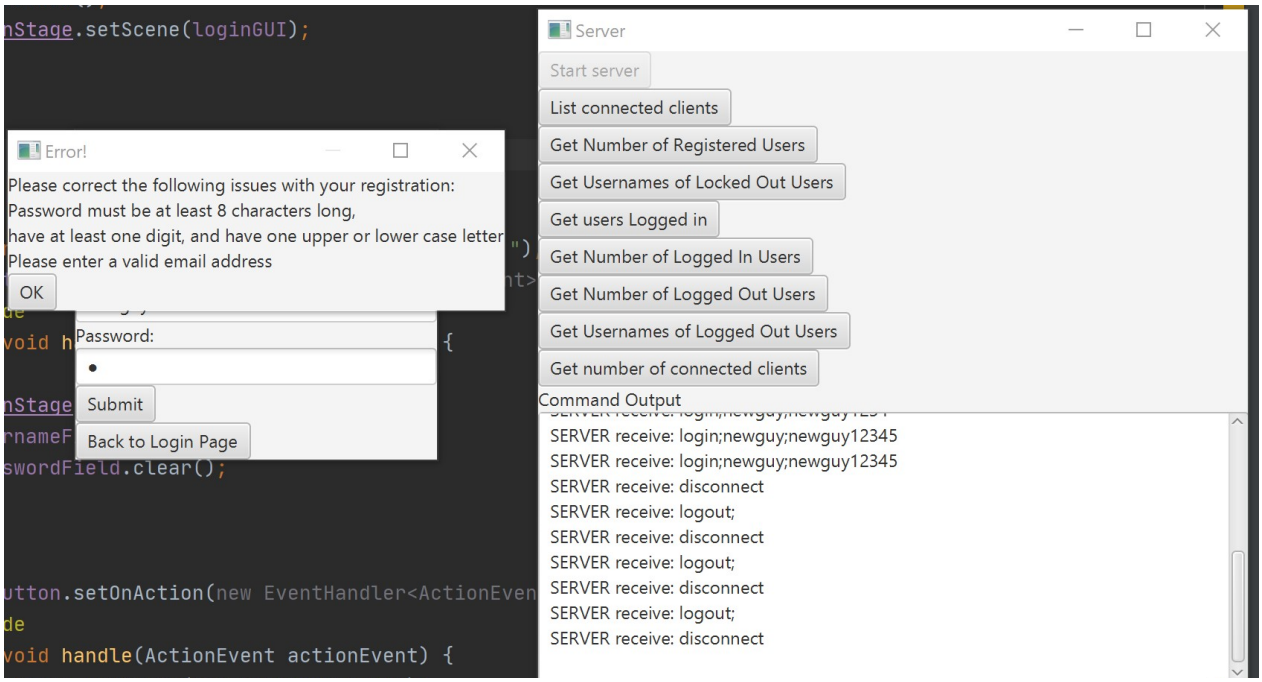


Input Sanitation



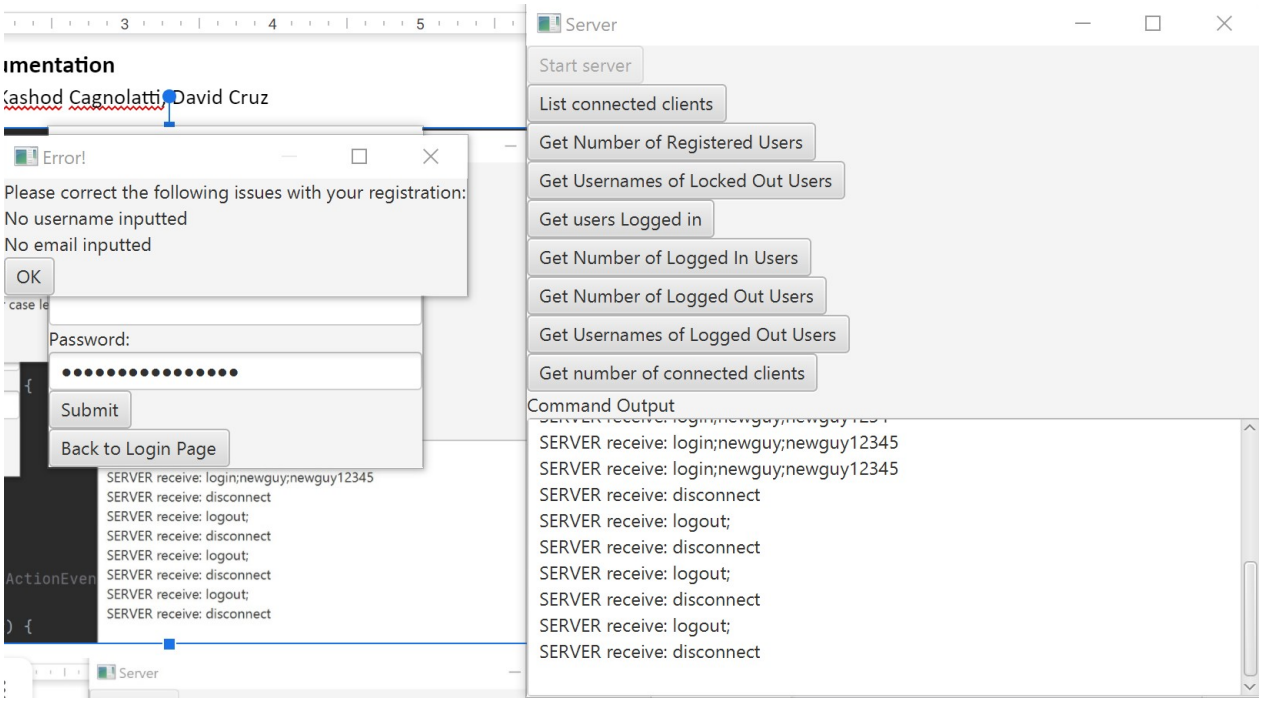
Server/Client Architecture Documentation

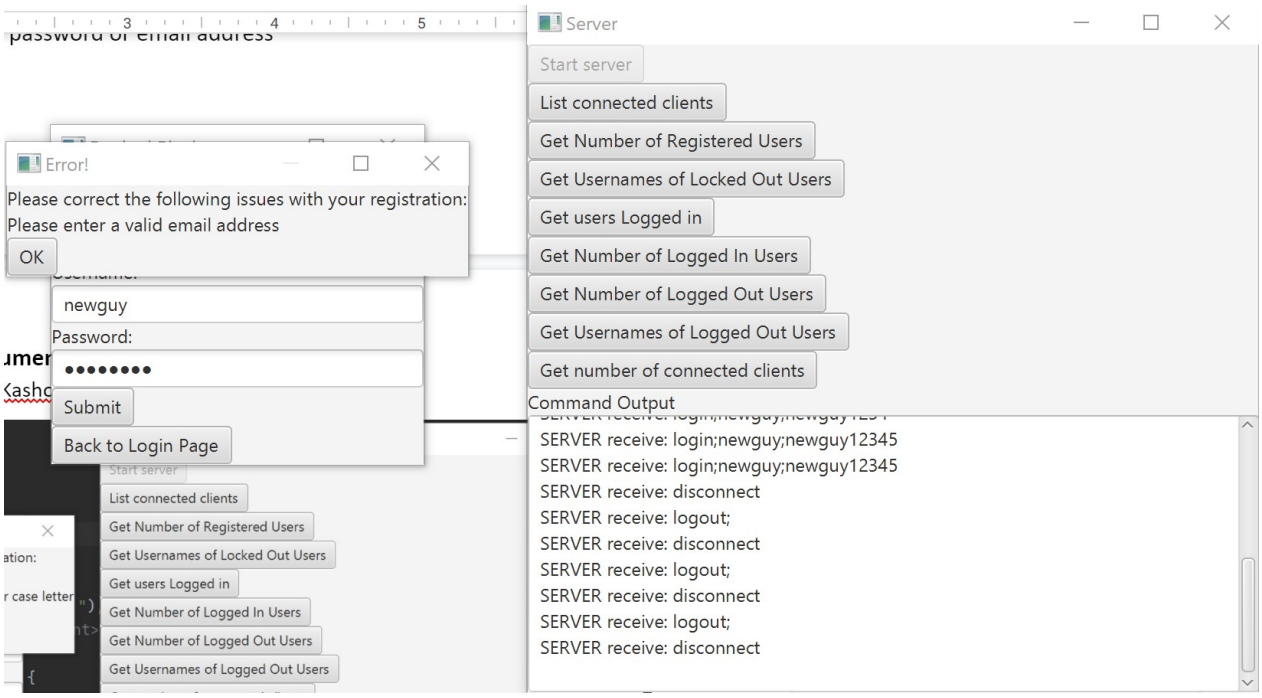
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



Server/Client Architecture Documentation

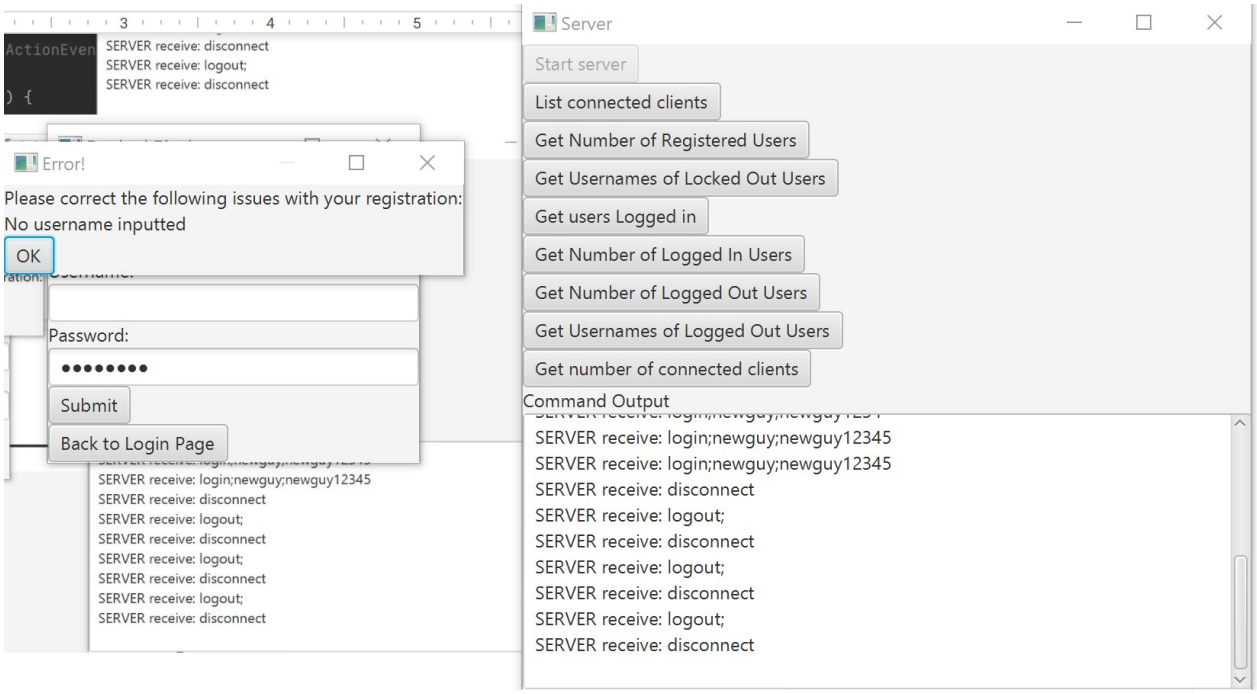
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

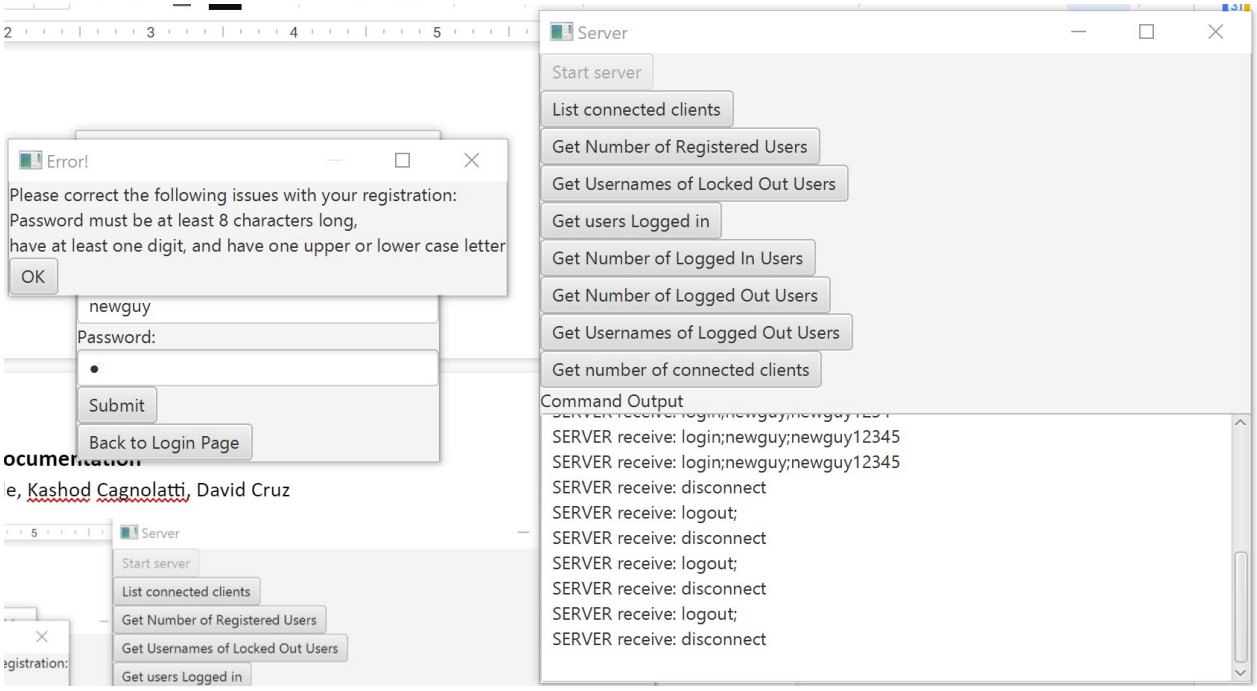




Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



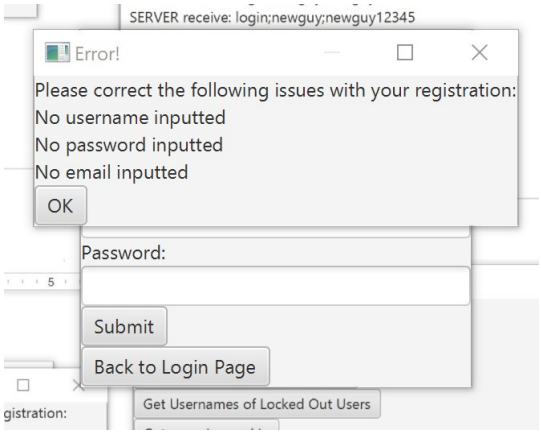


ocumentation

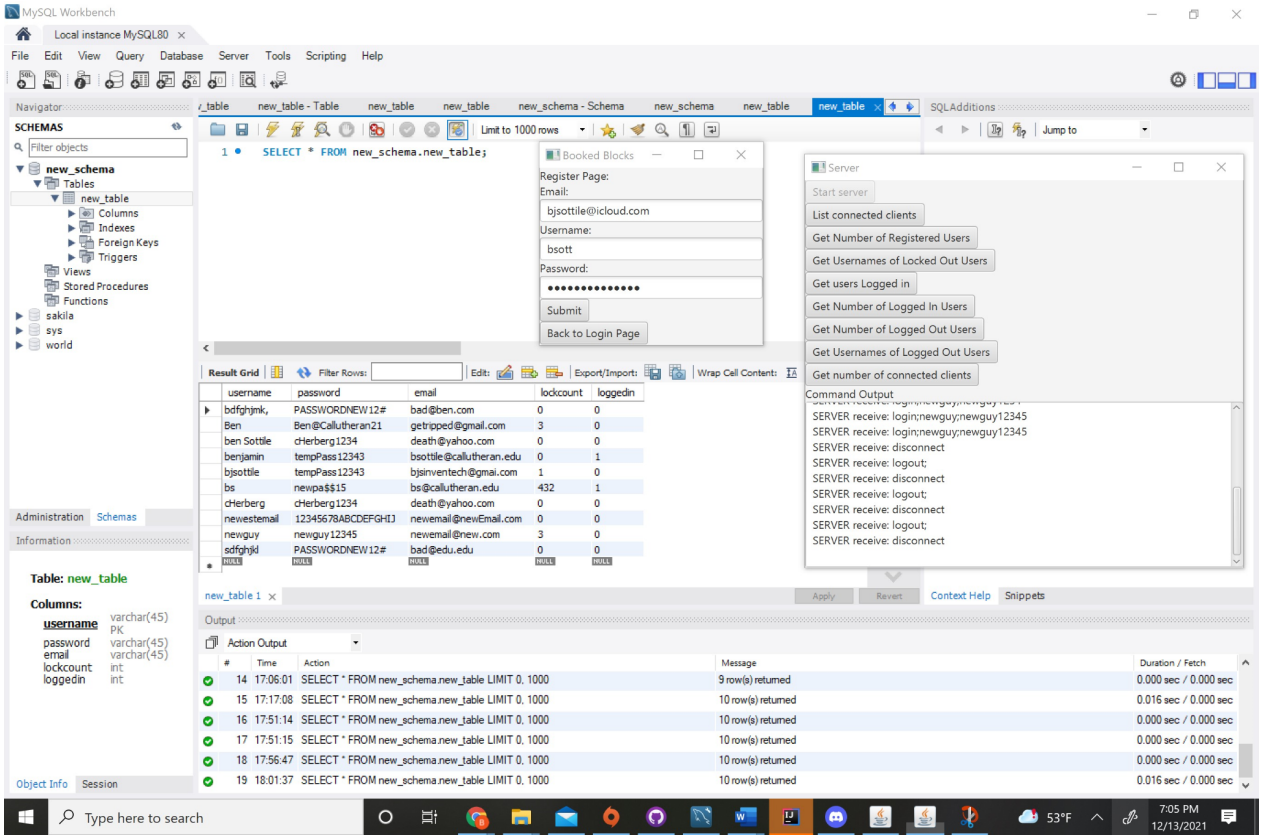
e, Kashod Cagnolatti, David Cruz

Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

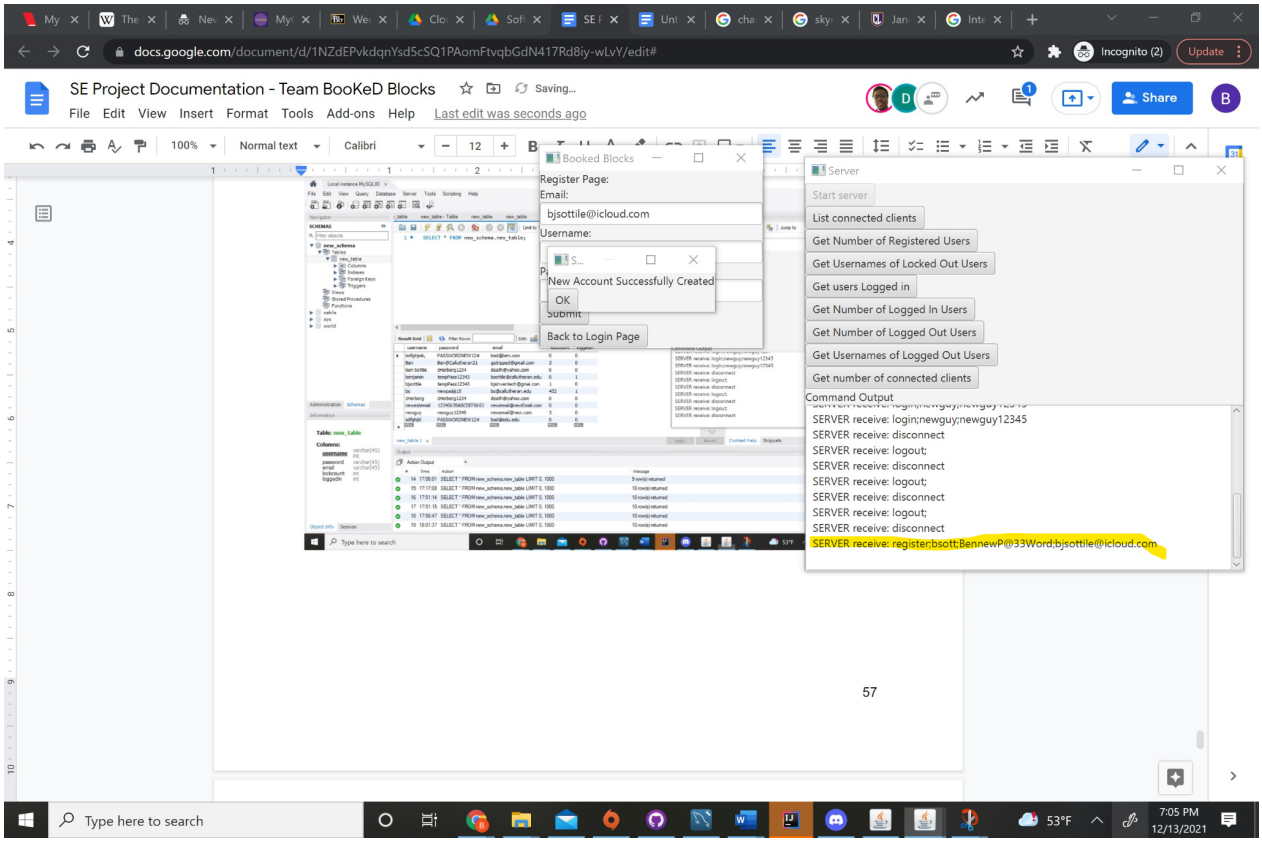


Correct Registration



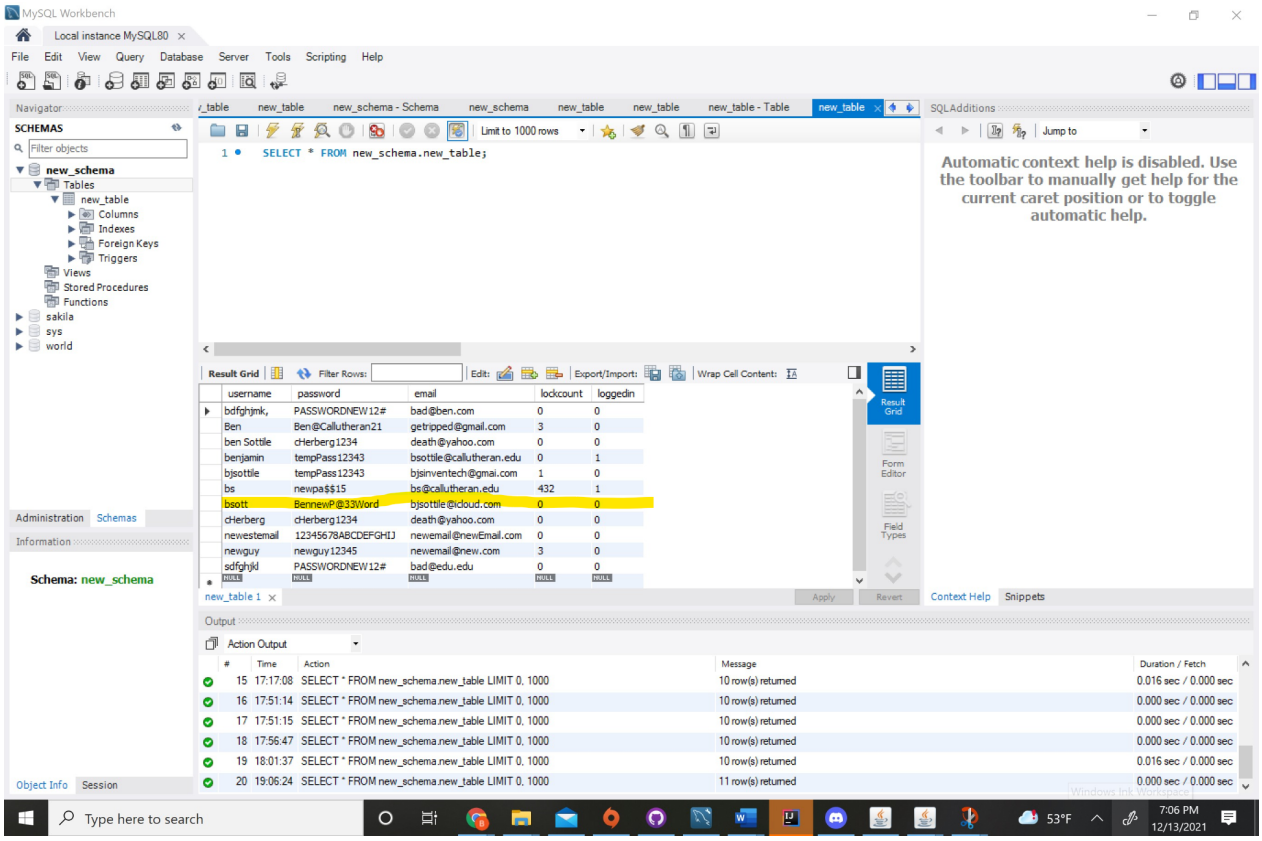
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



Server/Client Architecture Documentation

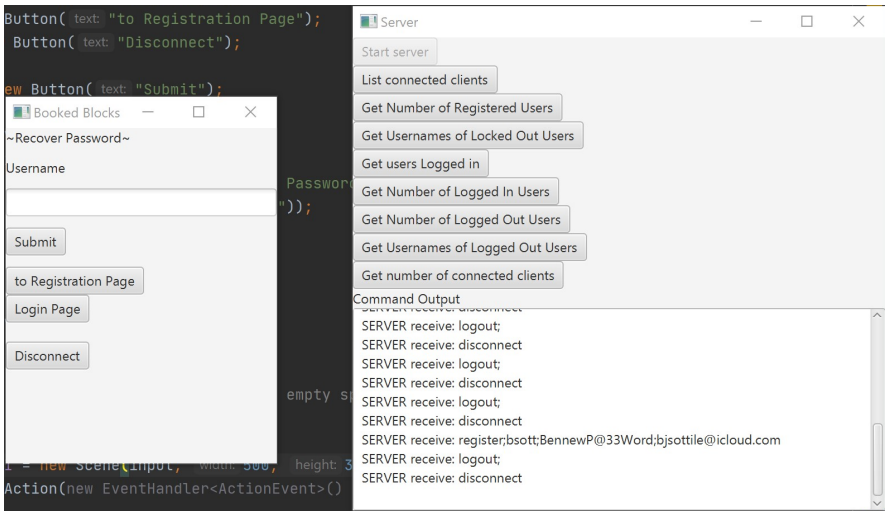
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

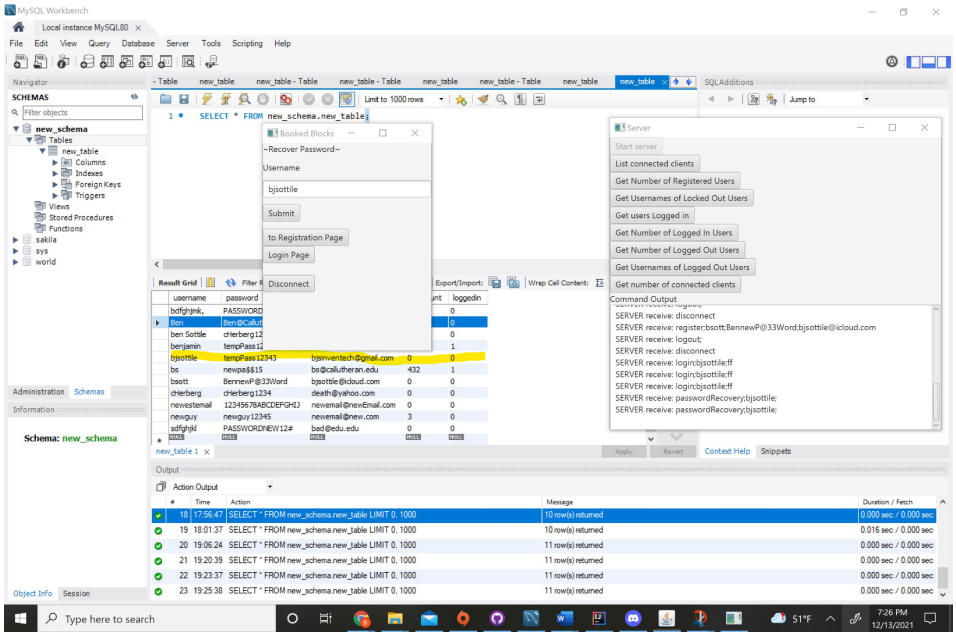


Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

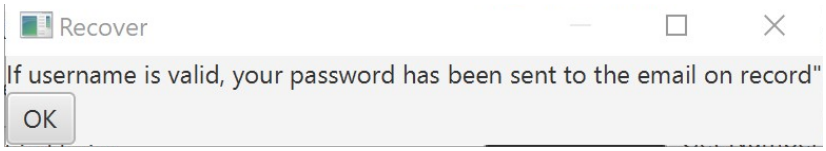
Recover Password GUI

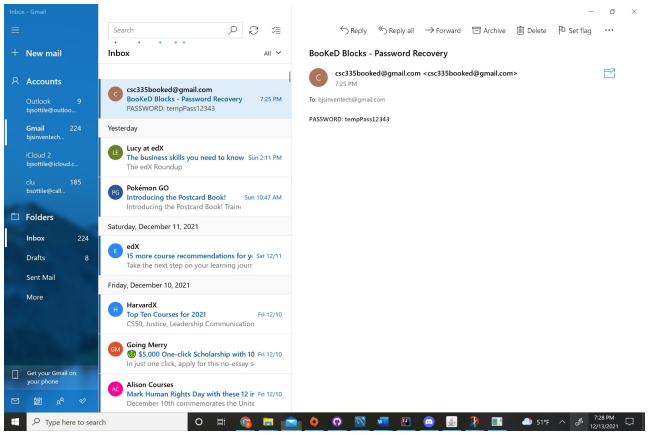




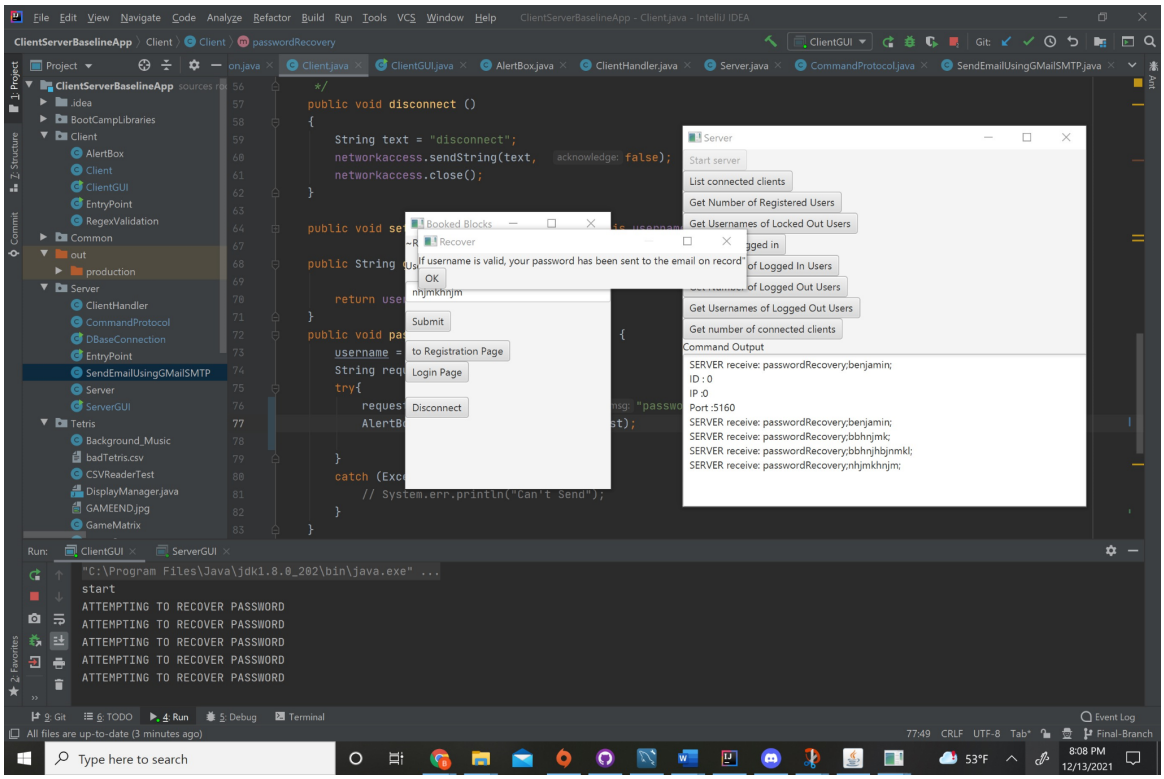
Server/Client Architecture Documentation

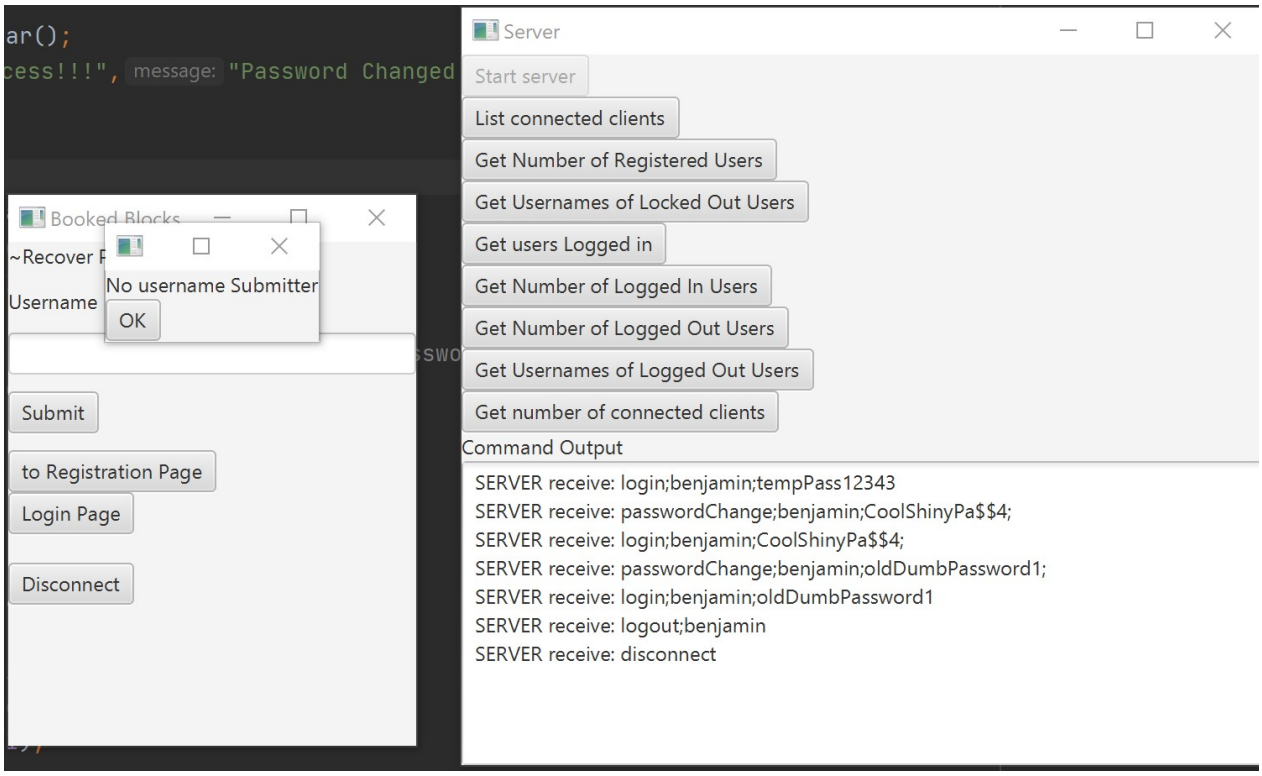
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



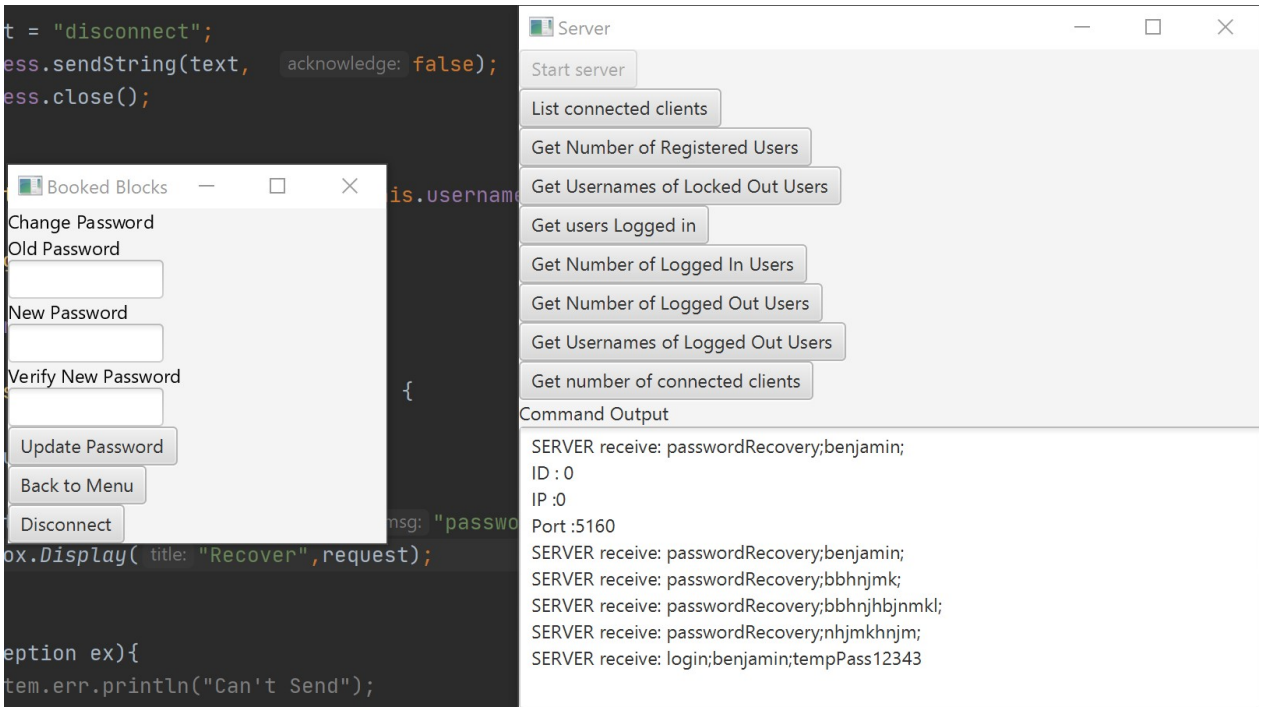


Bad User Name Recovery




ar();
cess!!!", message: "Password Changed
Booked Blocks
~Recover P
Username
No username Submitter
OK
Submit
to Registration Page
Login Page
Disconnect
Server
Start server
List connected clients
Get Number of Registered Users
Get Usernames of Locked Out Users
Get users Logged in
Get Number of Logged In Users
Get Number of Logged Out Users
Get Usernames of Logged Out Users
Get number of connected clients
Command Output
SERVER receive: login;benjamin;tempPass12343
SERVER receive: passwordChange;benjamin;CoolShinyPa$$4;
SERVER receive: login;benjamin;CoolShinyPa$$4;
SERVER receive: passwordChange;benjamin;oldDumbPassword1;
SERVER receive: login;benjamin;oldDumbPassword1
SERVER receive: logout;benjamin
SERVER receive: disconnect

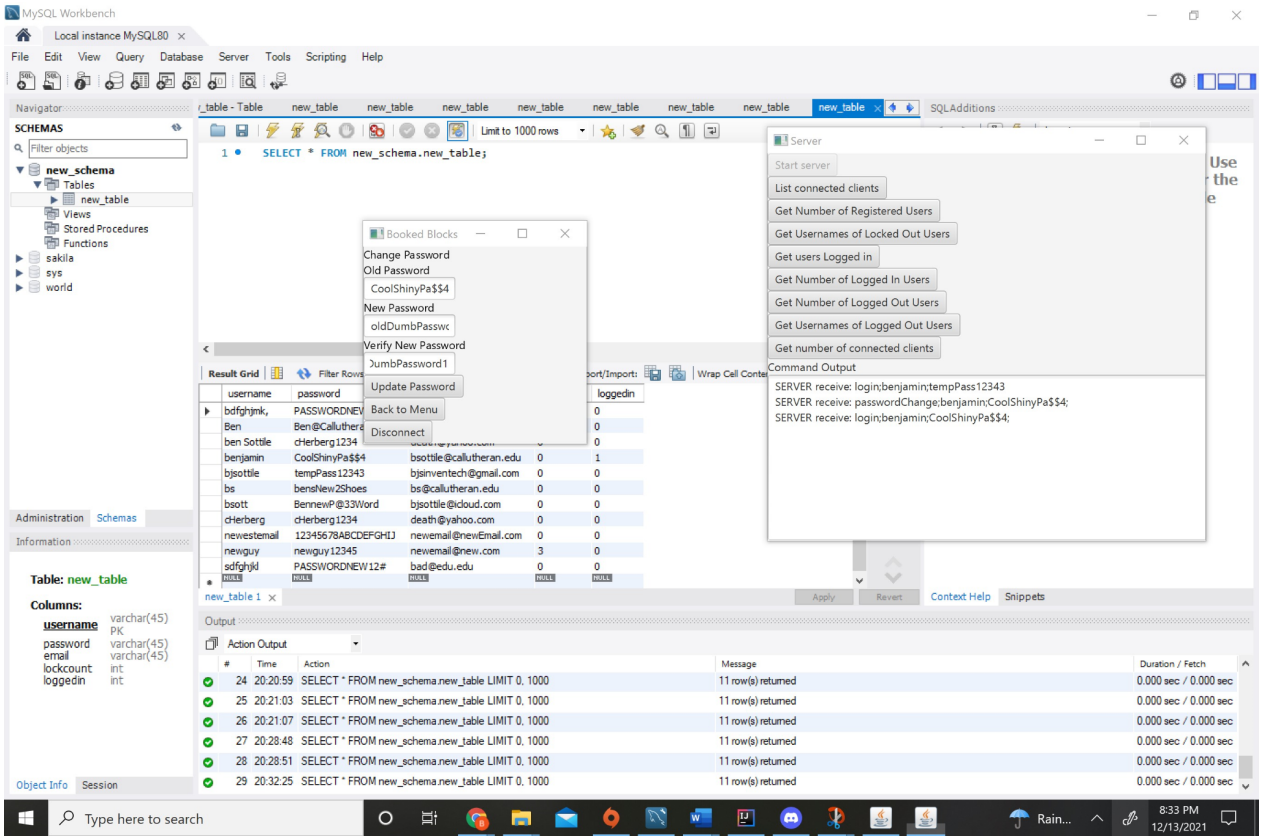
Change Password GUI



Valid Password Change

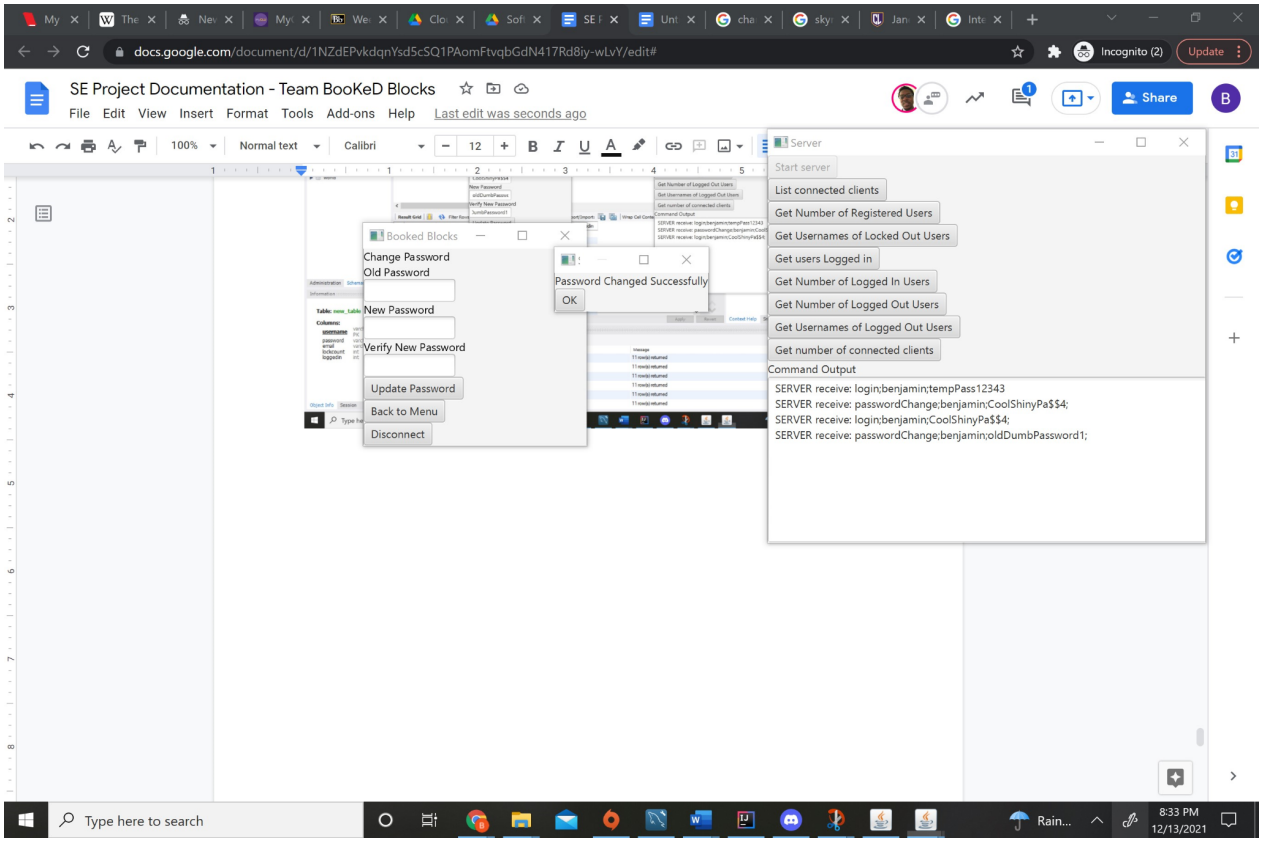
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



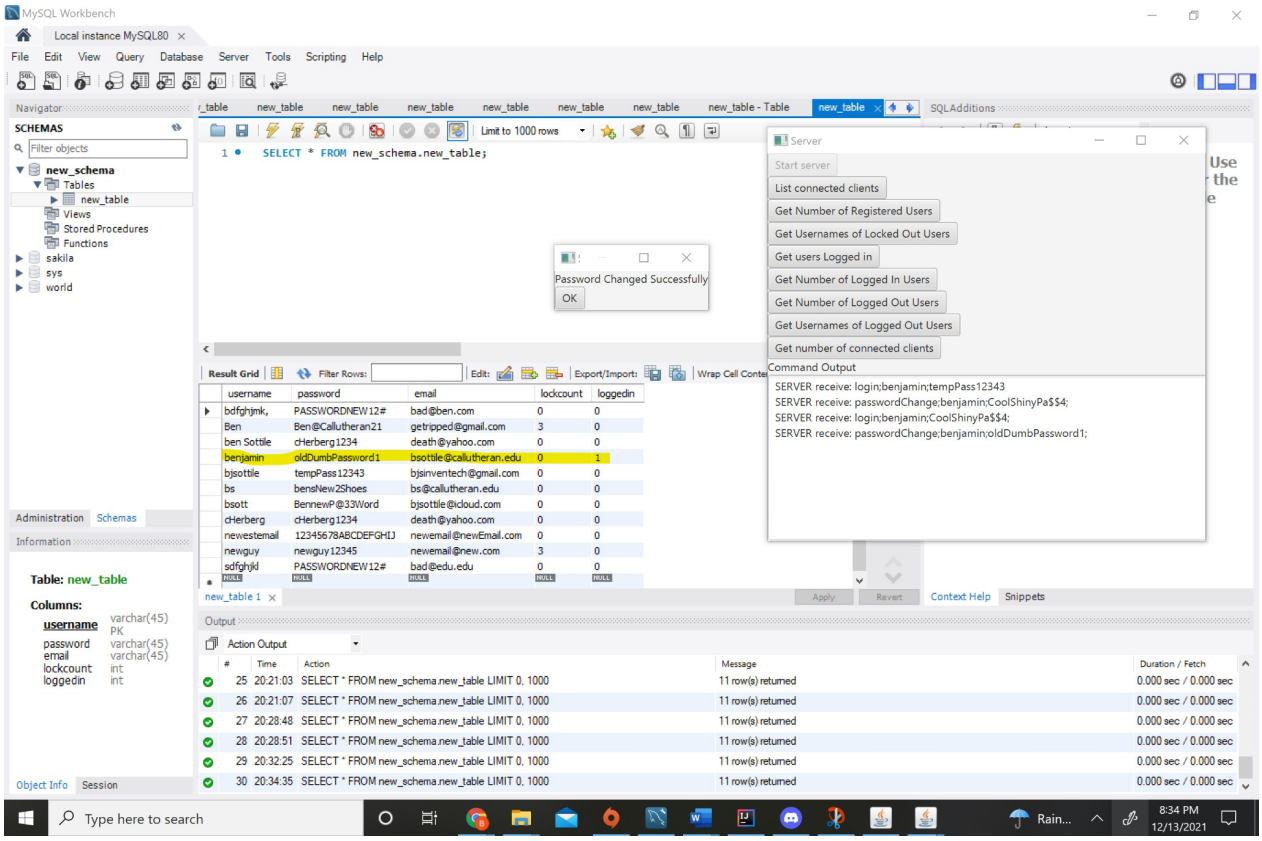
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

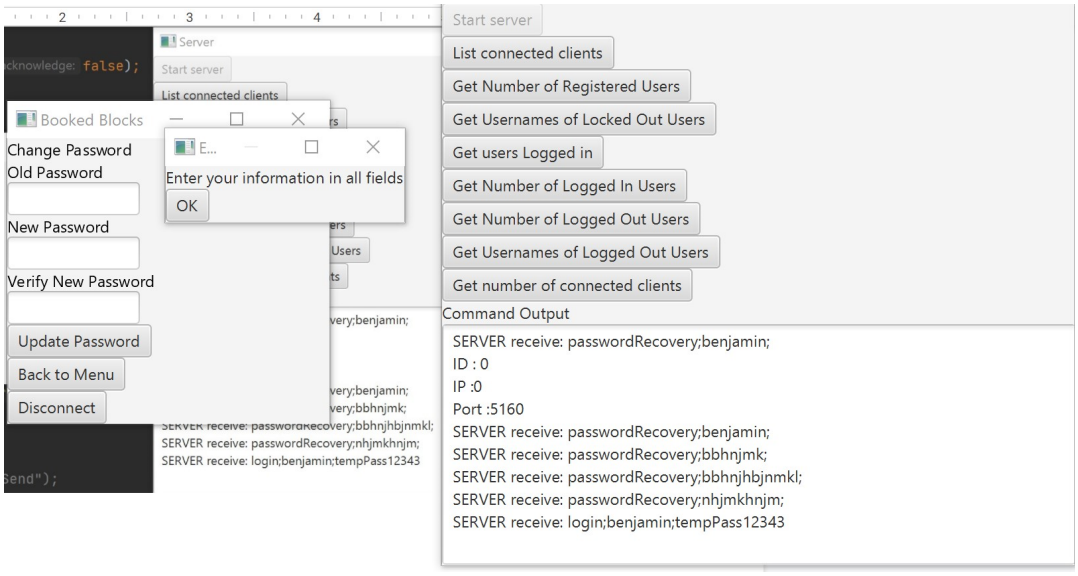


Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

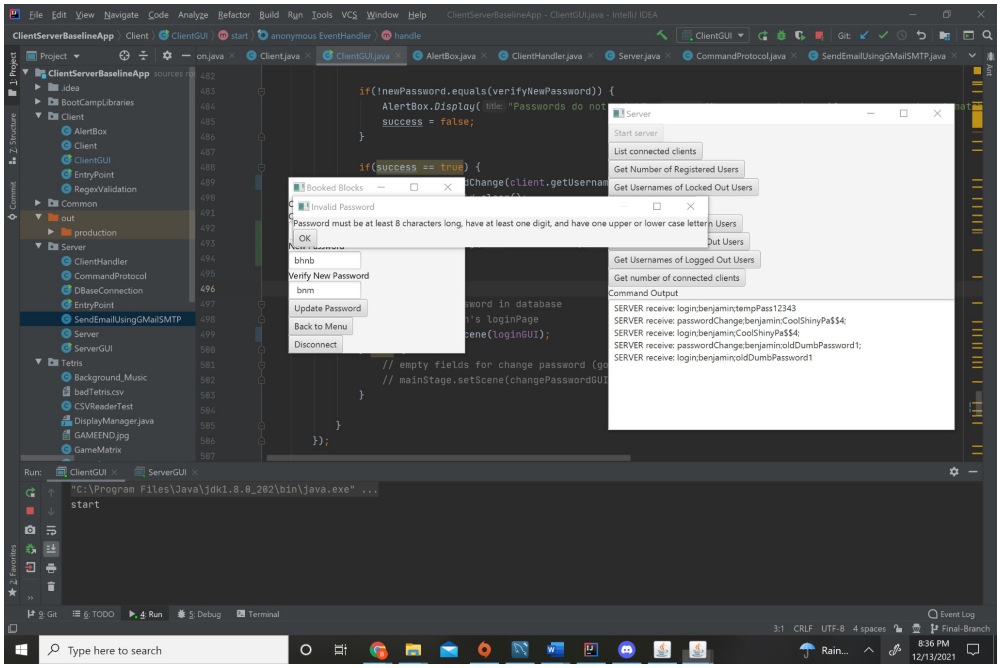


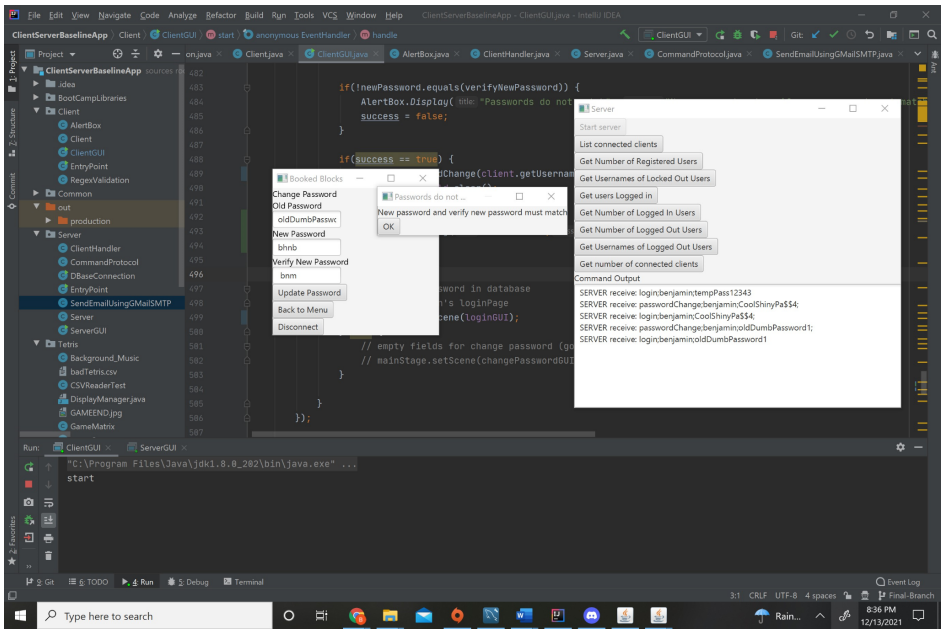
Invalid Inputs For Password Change



Server/Client Architecture Documentation

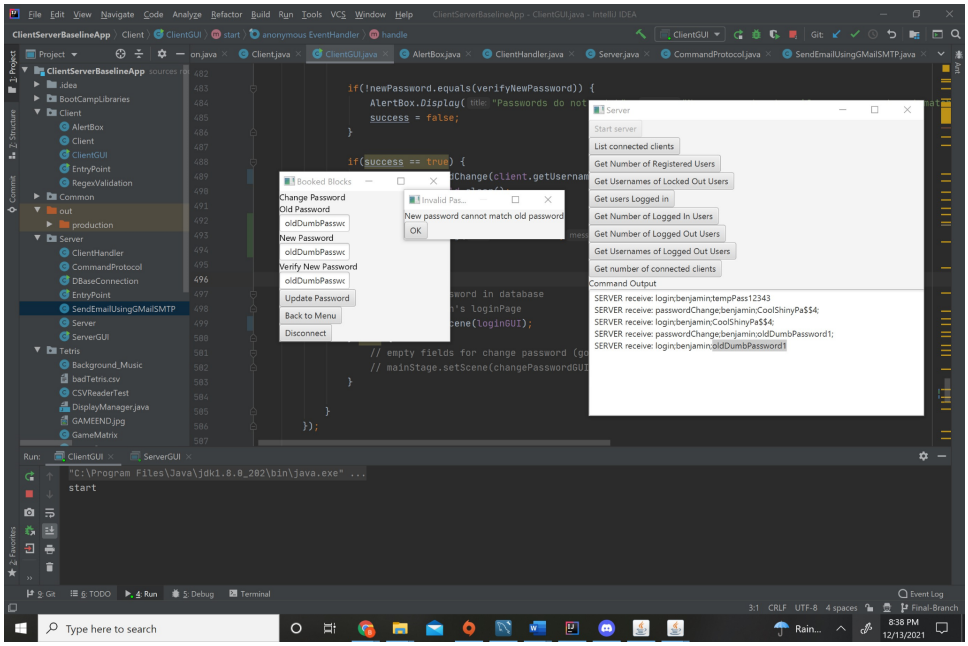
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz





Server/Client Architecture Documentation

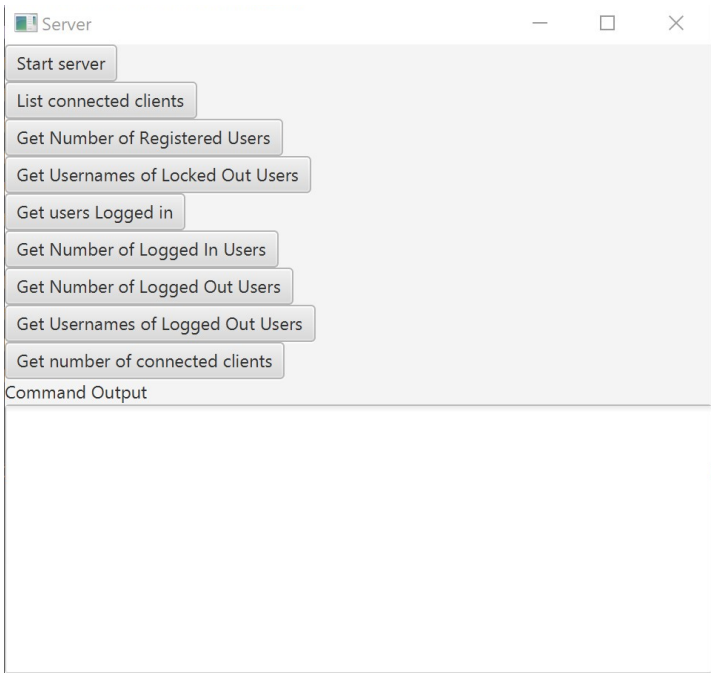
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Server/Query GUI Screenshot

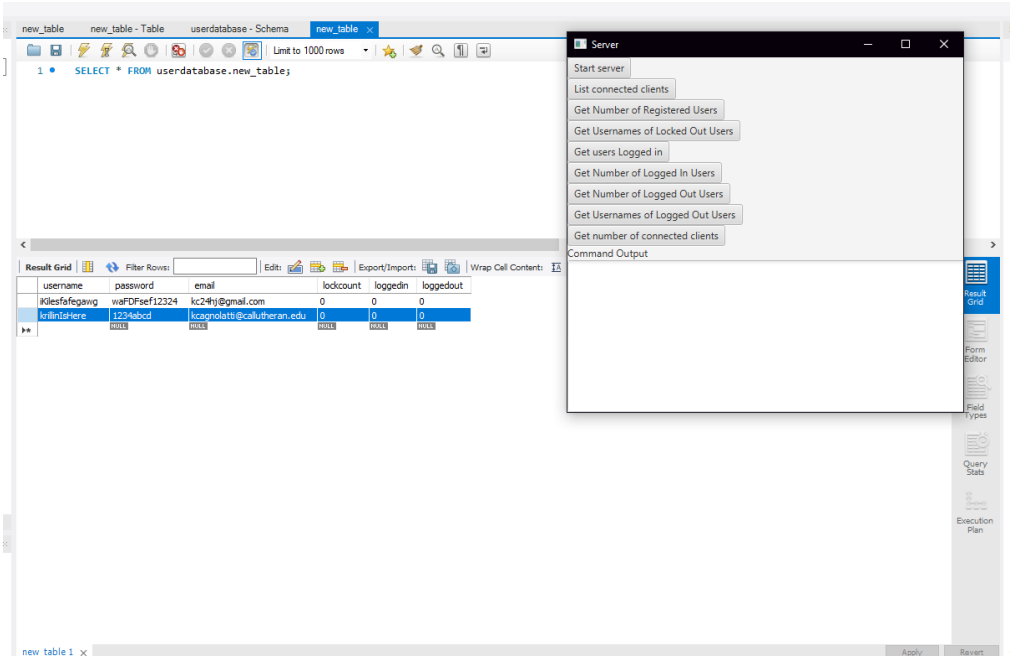


Server/Client Architecture Documentation

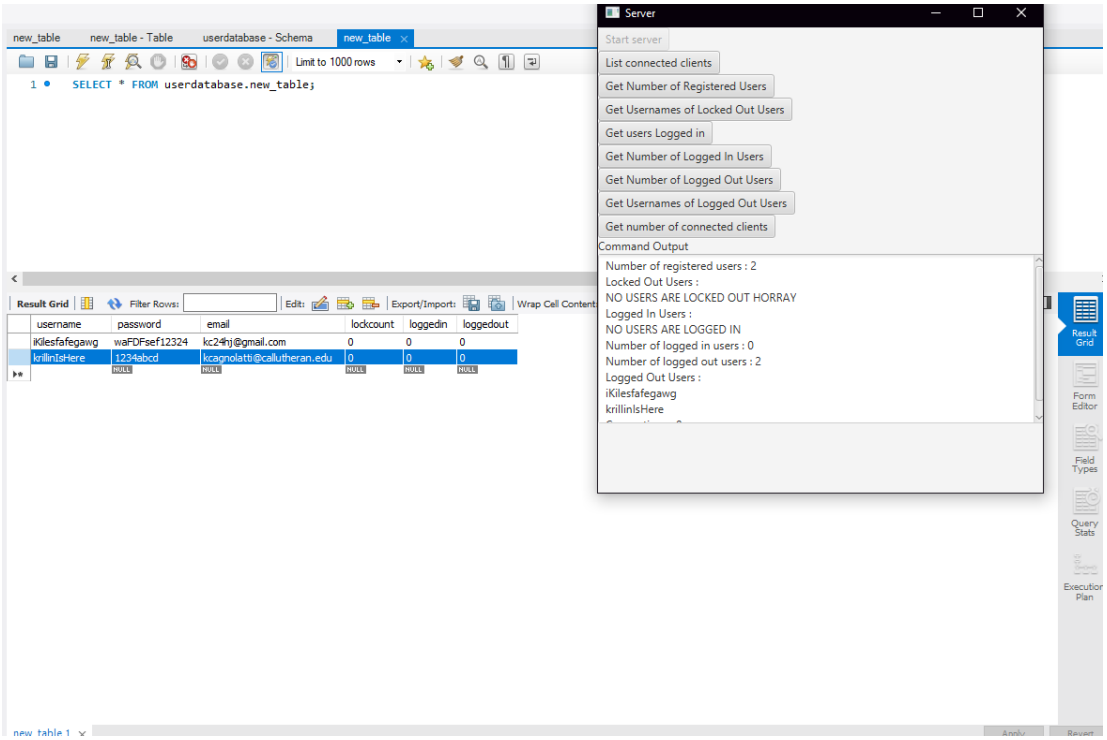
Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Admin Queries

Server Offline - Messages Query Messages Displayed to Command Output - No Connected Clients



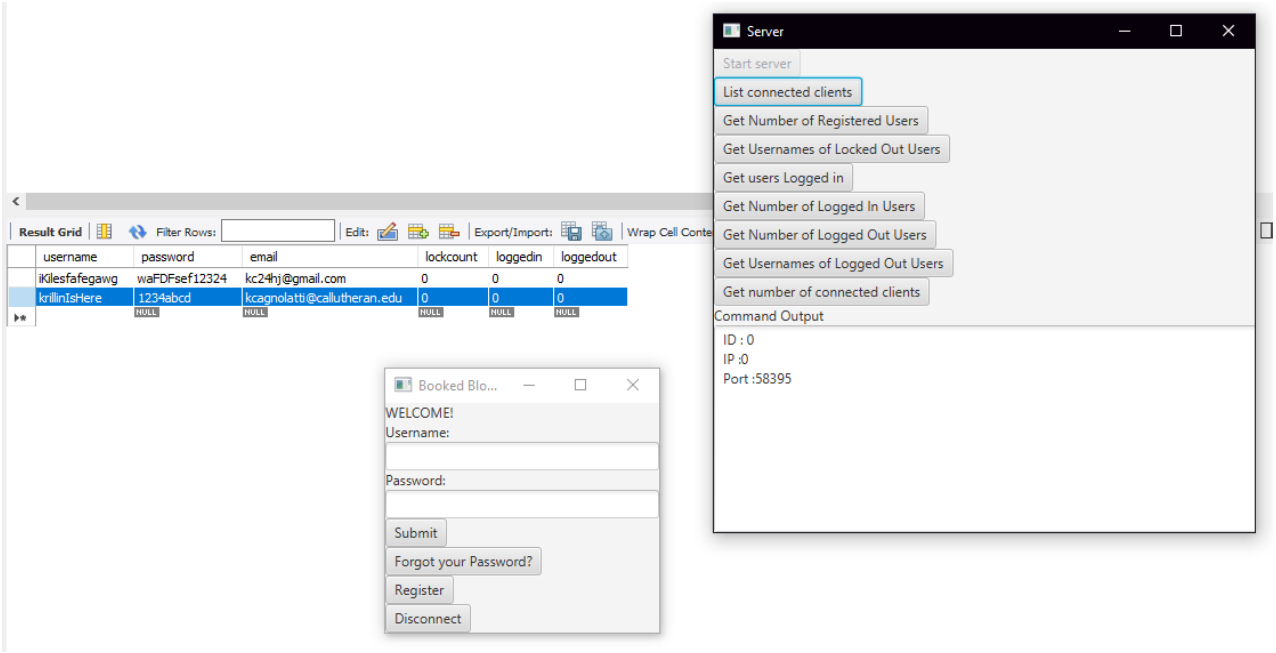
Server Online - Messages Query Messages Displayed to Command Output - No Connected Clients



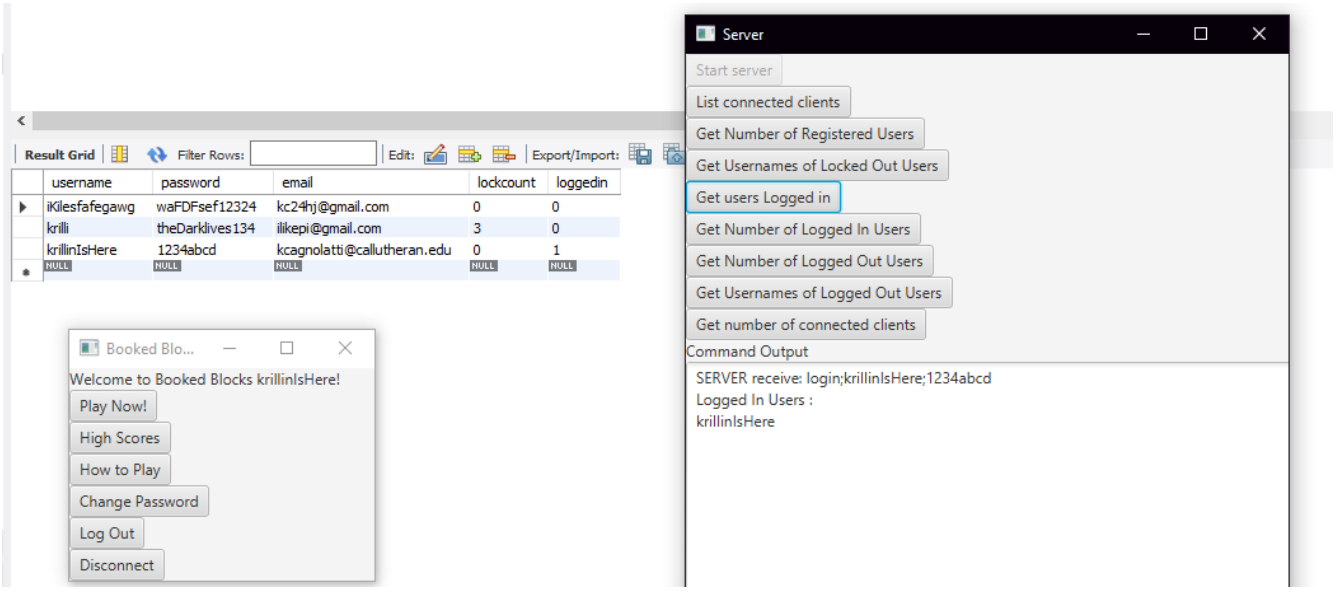
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Function Check - List Connected Clients



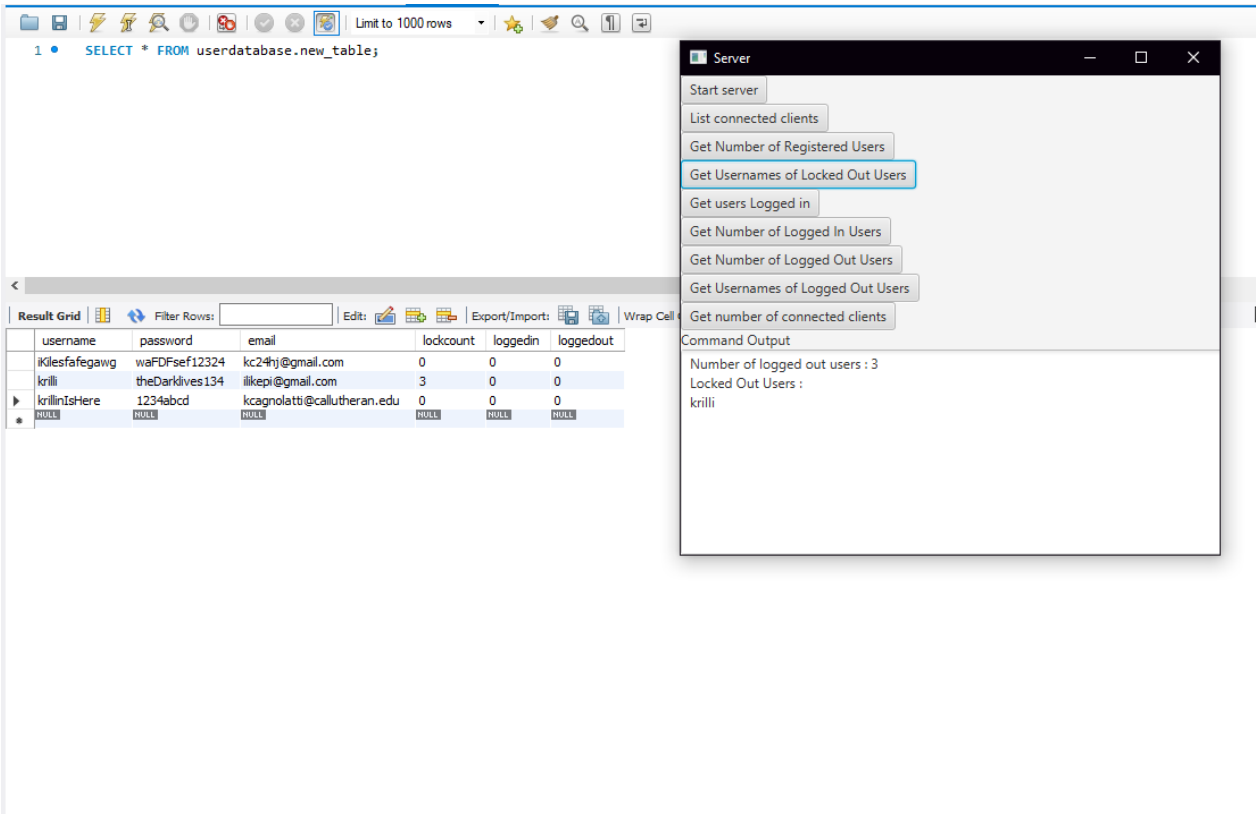
Function Check - Get users Logged in



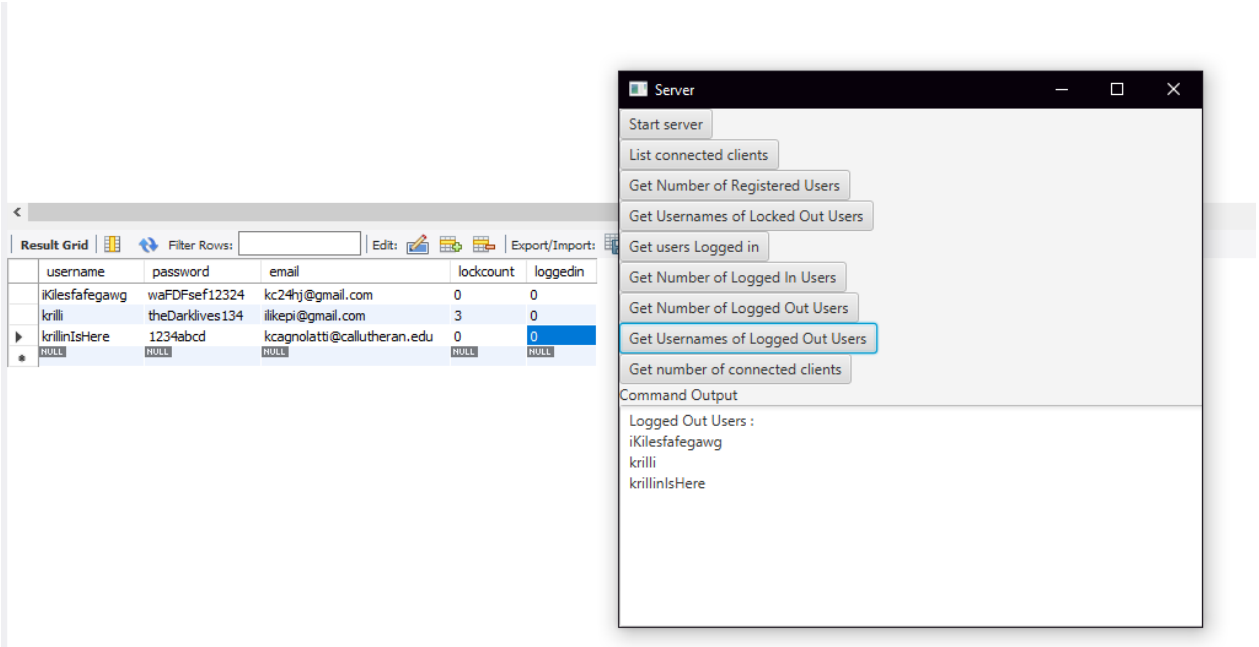
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Function Check - Get Usernames of Locked Out Users



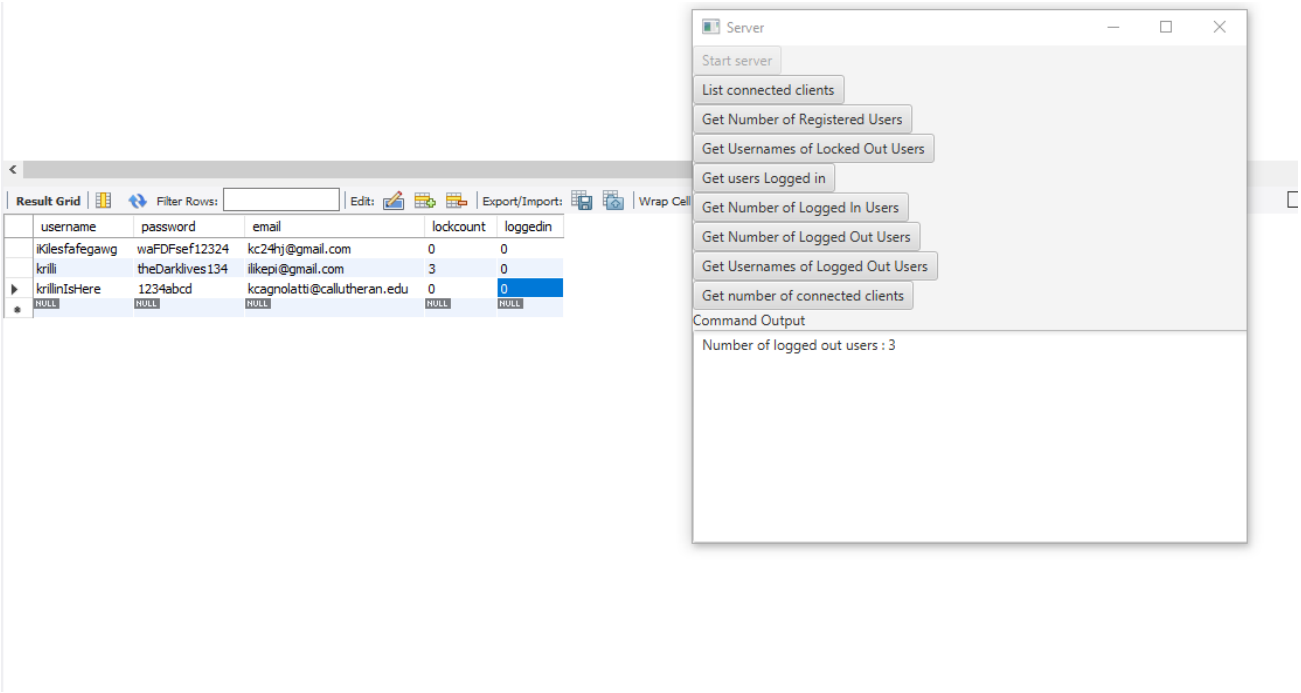
Function Check - Get Usernames of Logged Out Users



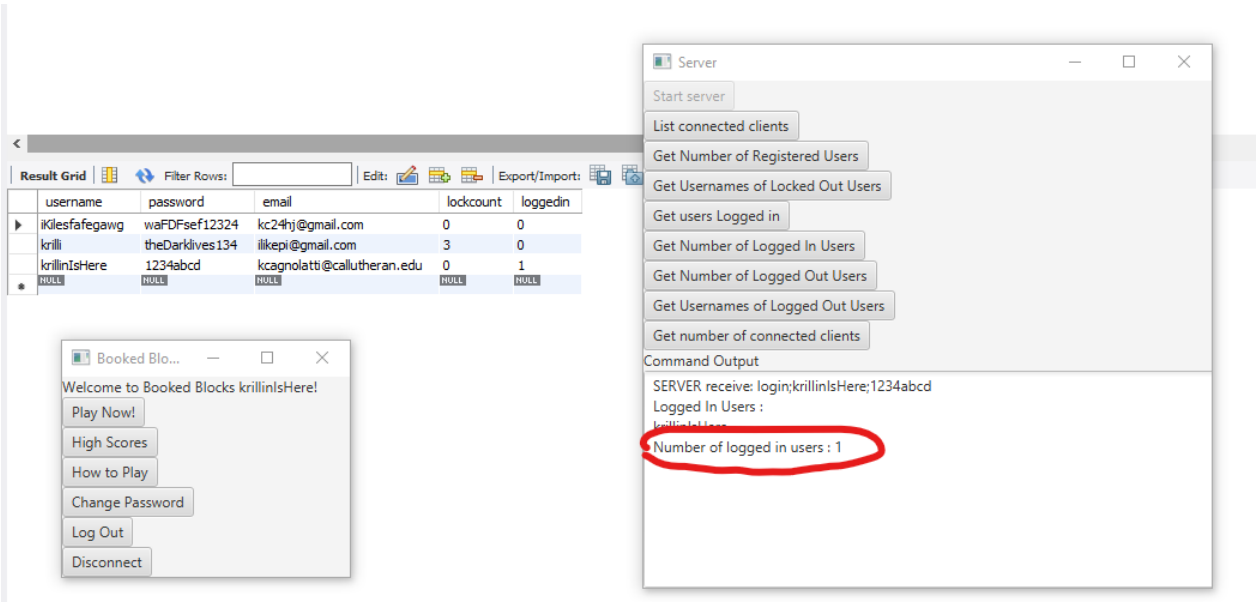
Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Function Check - Get Number of Logged Out Users



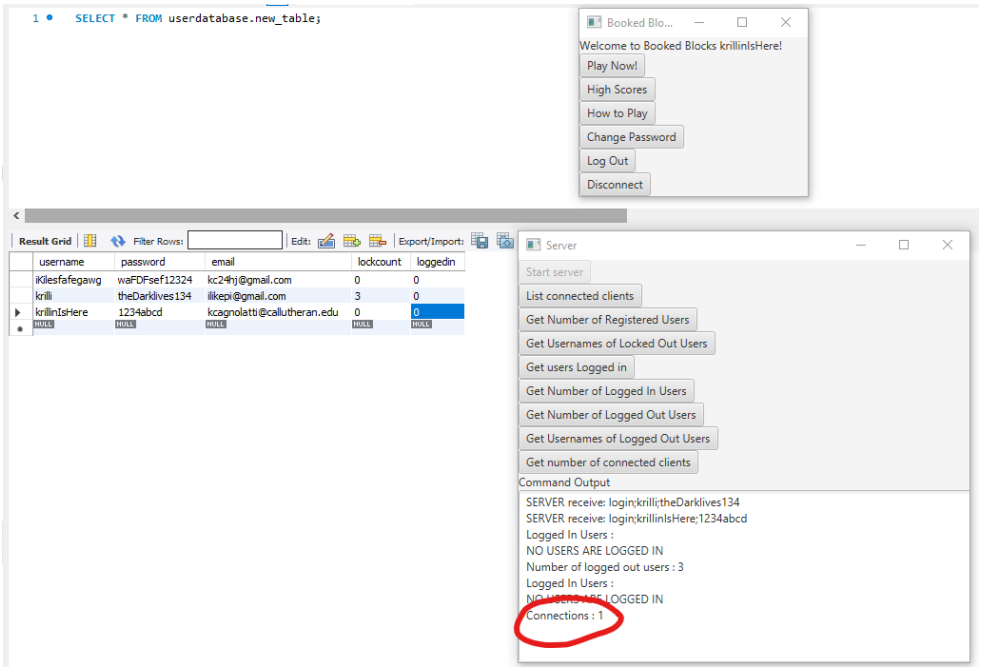
Function Check - Get Number of Logged In Users



Server/Client Architecture Documentation

Team **BooKeD Blocks**: Ben Sottile, Kashod Cagnolatti, David Cruz

Function Check - Get number of connected clients



[Report Conclusion]

List of fully completed (and working in code use cases)

a. Connect
    i. Via localhost as textbox input and working server
    ii. Via “127.0.0.1” Input Server IP Directly
    iii. Incorrect IP Format
b. Disconnect
    i. Client/Server connection severed (goal) with [X], User isn’t Logged in
    ii. Client/Server connection severed (goal) with [X], User is Logged in
    iii. Client/Server connection severed (goal) and User isn’t Logged in
    iv. Client/Server connection severed (goal) and User is Logged in
c. Registration
    i. Valid input for username, password, and email
    ii. Username - already registered
    iii. Username - username field empty
    iv. Email - invalid email form
    v. Email - already registered
    vi. Password - Invalid minimum Length, length less than 8.
    vii. Password - Contains invalid Characters
    viii. Password - Invalid minimum Length and Contains invalid Characters
    ix. Password - Doesn’t Include numbers
    x. Password - Doesn’t include letters
d. Login
    i. Login successful
    ii. Invalid username
    iii. Invalid password (attempts 0-3)
    iv. Invalid password (locked out)
e. Logout
    i. Logout Successful
f. Password recovery
    i. Valid Username
    ii. Valid username and account is locked
    iii. Invalid Username
g. Change password
    i. Valid old password, new password field and verify new password field match,
    ii. Old password and new password match

iii. New password field and verify new password field don't match
iv. Invalid minimum Length, less than 8 characters
v. Contains invalid Characters
vi. Invalid minimum Length and Contains invalid Characters
vii. Doesn't Include numbers
viii. Doesn't include letters

h. Admin queries
   i. System Completely Functional
   ii. System Failure Present
i. Database functions
   i. Search Attempt Successful
   ii. Search Attempt Failure
   iii. Insertion Attempt Successful
   iv. Insertion Attempt Failure
   v. Update Attempt Successful
   vi. Update Attempt Failure

List of remaining, incomplete use cases

a. Connect
   i. Can't handle valid unsupported IP addresses, currently prints "Unable to create I/O streams." then closes application
b. Logout
   i. Already disconnected message use case*possible safety net*
c. Admin queries
   i. Client/Server connection severed (not the goal/accidental disconnect) and User isn't Logged in
   ii. Client/Server connection severed (not the goal/accidental disconnect) and User is Logged in
   iii. System Failure Present - only displays errors in red right now

Description of difficulties encountered and how you addressed them

a. Version Control Errors - GitHub
   i. We attempted to use Github to make the implementation phase more flexible, due to our inexperience this idea was somewhat unsuccessful. We decided to send individual changes to Ben, as his PC was being used for our in class demonstration
b. JavaFX would not operate on Ben and David's IntelliJ

    i. MySQL was not working for David

c. MySQL was not working for David
    i. He worked on sanitizing code / other functionality unrelated to the database instead
d. Ben's Design (All GUI's in one class, retrieve information directly from the server) vs. David's Design (GUI's have their own class, pull information directly from the database)
    i. Our final design included ideas from both designs. We ended up having all GUI's in one class and decided it was best to pull information directly from the database. We used one GUI because it was easier for us to switch scenes to the login page, registration page, etc. We added columns for logged in, lockedout containing boolean values (0 or 1), and this way we were able to pull information directly from the database and see who is logged in and the amount of people logged in for our server queries.