1. Code Structure
    - Looking at this code snippet it seems like there is no Auth.h file. It is good to split the declaration and implementation of classes and functions into header and source files. The header file should have the class and function declarations, and the source file should have the implementations for the class and functions. Furthermore, since you are using namespace, the signature for implementations of functions should follow the convention: namespace::functionName.

2. std::string declaration
    - In the promptLogin() function (which I think is missing braces around the body) you have declared and initialized a std::string loginMessage. First thing I want to note is that the default constructor for std::string will initial the string to an empty string therefore you don't need to explicitly do that. Furthermore, later on in the body of the function, you are redeclaring loginMessage twice in the control block. You should either declare it once outside the control block and set the string in the control block, or just declare and set the string in the control block.
    - Option 1:

```
std::promptLogin(ChatWindow window){
    std::string loginMessage;
    ...
    if(...){
        loginMessage = "some message";
        return loginMessage;
    }else{
        loginMessage = "some other message";
        return loginMessage;
    }
}
```

    - Option 2:

```
std::promptLogin(ChatWindow window){
    ...
    if(...){
        std::string loginMessage = "some message";
        return loginMessage;
    }else{
        std::string loginMessage = "some other message";
        return loginMessage;
    }
}
```

3. Unnecessary else block
    - The control block in the promptLogin() function has an unnecessary else block. You can pull the code out of the else block. There is no point in having this else block as you are returning from the if block. This means that if your code enters the if block it will not make it to the else block, which is the same as not having the else block. The code in the else block can be seen as a default behaviour that will only occur in the case the if statement is false.
    - Suggested change:

```
std::promptLogin(ChatWindow window){
    std::string loginMessage;
    ...
    if(...){
        loginMessage = "some message";
        return loginMessage;
    }

    loginMessage = "some other message";
    return loginMessage;
}
```

4. Unclear login message
    - From this snippet of code, creating a new user seems a bit unclear. "Enter a username (type NEW to create account):", is the user supposed to enter "NEW username"? It would be better to have a separate command which takes care of creating a new user. So something like “Enter “Login” if you already have an account, or “New” to create an account”. This way is clear to the user what to do and you are doing what the function name implies it will do.

1. Using a namespace for your class will help reduce confusion in the long run. Doing a quick google search suggests that there are quite a few Auth or similarly named classes. However, the name of the namespace might want to be something broader. A user namespace that contains the Auth class may also want to contain the actual user class as well. In this case, it might be best to rename the namespace to something that can encompass both the user and authentication class.

2. It seems odd that the Authentication class is the one handling the creation of users as well. The function 'createUser' may be better in a User Manager interface that creates the User and may use the Auth class. Example: The Auth class hashs the user password and stores it into a database

3. "../include/Auth.h" tells me that the CMakeLists have not been made for the file. This can lead to headaches when cleaning up later on. Imagine if every component/library used relative paths instead of properly implementing CMakeLists. You would need to create each one while remembering how each library exists with one another, as well as remove all of the relative #include inside of the code.

4. Not quite sure how this code works since the functions are not linked to the class. Missing the class declaration for the member functions. Line 35 contains a semicolon instead of a bracket and there is no end brace. Inside of promptLogin, the username is checked to equal the variable NEW. This makes no sense since username is a string and matching a username with another string to determine if it is a new user or not seems odd. The ChatWindow function is also displayText, not DisplayText. However, that could have been changed the implementation.

It seems odd to start with the chat client since it is not directly linked to the actual game. We also do not need to improve/change the client interface as stated by the prof. A recommendation is to begin on the server and work with login on that end. Assuming you want to start with login.

```
//
// Created by Andrew on 2019-01-17.
//

////////////////////////////////////////////////////////////////////////////
// Auth.cpp
//
// This library will hold our implementation for auth
//
//
////////////////////////////////////////////////////////////////////////////
#include "../include/Auth.h"
#include "ChatWindow.h"
#include "Client.h"

using networking::Server
using networking::Connection
using networking::Message


namespace user {

    class Auth {

    public:
        promptLogin(ChatWindow Window);
        authSession(Connection c);
        createUser(Connection c); <-- 1
    };


}

std::string
promptLogin(ChatWindow Window); <-- 2
    std::string loginMessage{""};
    chatWindow.DisplayText{"Enter a username (type NEW to create account):"};
    std::string userName = Window->impl.onTextEntry.text;
    chatWindow.DisplayText{"Enter the password for this username (or for new
account):"};
    std::string pwd = Window->impl.onTextEntry.text;
    //hash pwd before send; <-- 3
    if (username == NEW){
        std::string loginMessage = "!NEW " + userName + " " + pwd;
        return loginMessage;
    }
    else {
        std::string loginMessage = "!LOGIN " + userName + " " + pwd;
        return loginMessage;
    }


void authSession(Connection c){



};

void createUser(Connection c){


};
```

1.

In general, it's hard to tell what the Auth class does in this case. Because of this, I will assume this class was designed to hold the interactions for authorizing a user since there are calls to display text. If that is so, I do not think implementing a method to create a user here is a good idea. It does not fall under the role of authorizing and instead probably be done by something that manages users like a database.

2.

promptLogin is well designed as it is clearly focus's on interacting and getting input from the user. It allows it to be reusable and also resistant to change. If for example, login was switch to a simple PIN, this can still be reuses.

3.

Although a comment, I would advise not doing hashing in login. This changes the expectation of how this method should give back what the user typed in when prompt to login. Instead of string directly from the user, it's gives some string that can be hard to understand when debugging. If the hash is for checking if the username given exists in some storage, I still believe that should be part of the prompt user method. That should belong to the class that contains the information. Implementing it in the class means that it could be potentially reused.

Group Babka — Code Review 1

Code not compiling

Code cannot compile because C++ syntax is invalid and it contains other errors:

1. Declaration type specifiers missing
   ```
   void createUser(Connection c);
   ```
2. Incorrect spelling of names
   ```
   chatWindow.displayText(“text”);
   ```
3. Incorrect member access operator
   ```
   Window.impl.onTextEntry.text
   ```

Invalid scope

Member functions are in `Auth` namespace, `Auth` is in `user` namespace. Function definition `createUser` must specify its namespace or it will be defined in a different scope. Program will crash when calling an undefined function.

```
void user::Auth::createUser(Connection c) { }
```

Parameter value

The parameter passed to function `promptLogin` is passed by value — argument will be a copy. This is unnecessary and inefficient for large objects. Instead, pass parameter by constant reference to an existing instance of a `ChatWindow`:

```
string promptLogin(const ChatWindow &Window);
```

Code responsibility, repetition, and style

- Class name does not describe its functionality: interacting with and displaying text messages in a chat window, hashing passwords, creating new users, and generating text commands are not responsibilities of an authenticator. Consider
  (1) alternative class name to describe it better; or
  (2) alternative class design to septette authentication from chat window.
- Consider more descriptive declaration of `authSession`:
  ```
  bool authConnection(Connection c);
  ```
- Unnecessary duplicate code.

Code Review for Babka

lines 26-28
  - Are these the public methods of the class Auth?
  - What are the return types? Are they purposely not specified?
  - Looking at the rest of the code, they should be:
```
        std::string promptLogin(ChatWindow Window);
        void authSession(Connection c);
        void createUser(Connection c);
```

lines 34-49
  - Is promptLogin() a function that returns a std::string?
  - If it is, the syntax looks a little odd
  - Why is std::string on a line by itself followed by promptLogin(ChatWindow Window);?
    - Why is there a semicolon at the end?
      - The following looks more appropriate:
```
        std::string promptLogin(ChatWindow Window {
                *implementation goes here*
        }
```

line 43
  - Where is the variable NEW initialized?
  - Is this a global variable?
  - If it's not, it may be a good idea to change the variable name to all lower case
  - Since variable new is unavailable in C++, perhaps something like newUser would work?

Vu Nguyen
301284431
Critic to Babka

1) Bad code practices:
   - There is no return type for method declarations in Auth class.
   - promptLogin method implementation missing curly bracket.
   - chatWindow object is undefined in promptLogin method. Should be Window.DisplayTest{}.
   - Misspelled “userName” variable in “if (username == NEW)”.
   - NEW variable is undefined.
2) The design will be easily misuse and security issue:
   - promptLogin method should only responsible for prompting the user to input username and password. However, it handles the logic of hashing password.
   - Hasing shouldn’t be in the method because of security issue. Since the method is public, one can get the user password hashing string from calling the method.
   - Also, the hashing function is weak. The hashing function only indicates if the user wants to create an account or login.
3) The API did not separate from implementation:

```
4) namespace user {
5)
6)     class Auth {
7)
8)     public:
9)         promptLogin(ChatWindow Window);
10)        authSession(Connection c);
11)        createUser(Connection c);
12)    };
13)
14)}
```

   - Should have the header file for Auth called Auth.h so that the API can be separated from implementation.

Is the intent of the code clear?
yes

Does it have a clear, focused responsibility?
N/A

Is it designed to be hard to misuse? Could confusion cause errors?
no

Is the code appropriately idiomatic for the language (C++ in this case)?
no

Does the design violate the principle of least astonishment[1]?
no
It asks user to provide username and password for login/register purpose, in case of registration, it uses the same provided username/password.

Are there issues with coupling? (temporal, odd inheritance, leaky abstractions)
no

Specifically, does the interface hide design details to eash its use?
no

How might the code better handle change?

Are there alternative designs that could provide better trade offs?
Window->impl.onTextEntry.text is not a function and on the running time it would not wait for the user to enter a username or a password. A proper way of implementing this is either use a blocking-function that waits for user to enter a text or using callback design in which upon entering username/password, callbacks are being trigged.

Are there performance or security concerns that should be addressed?
yes
1. the function should not return raw password. We should hash the password before return it.
2. function should check some primary requirements for the text entry, such as the text limit (a range of 3-25 characters) or having sql injection codes.

Do all names make sense? (types, functions, variables, …)
1. loginMessage is an unused variable.
2. chatWindow is not defined, ChatWindow is a type (class) and Window is an instance of ChatWindow. but chatWindow is none of them.

Do comments clarify or just repeat what the code does?
There is only one comment in the code; saying "hash pwd before send", which is what the code is lacking.

Is the API appropriately separated from the implementation?

no, the way we can read the entered text from the ChatWindow is confusing, it should be like a blocking function like Window->readText().

Are invariants clearly expressed?
N/A

Are control structures deeply nested?
yes

Are there broad stylistic issues?
no