Walkthrough 7: Client/Server Communication

**Requirements:**

**Walkthrough 5 completed.**

**Part 1: Client Modifications**

1. Open up your final walkthrough 5 project unity rogue like game.
2. Replace the calls in your gamemanager.cs file for playerDied and Login to instead call to a local server with <http://localhost:8777>.
3. Download the server from blackboard.
4. Open the solution and run.
5. Run the game and note any errors.
   * Note that we likely have some communication of data errors.
   * Ensure your calls to PlayerDied and Login are calling:

*Debug*.*Log*(req.*downloadHandler*.*text*);

This will be essential for debugging our issues.   
TIP: you can change your Debug.Log’s to Debug.Warning’s to help identify them easier if you like.

**Part 2: Client Server Debugging**

1. Attempt to run the game connecting to your new server.
2. Examine the Login Requests response.
   * You should see output like the following:



This is a response sent back from the server.

We should now evaluate the server code for what it expects for a header that may be missing.

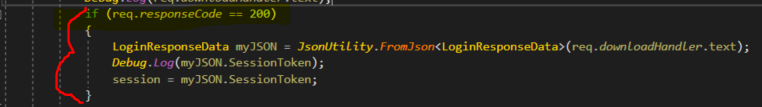
1. Notice that we also get an error immediately after our log:



We assume that because we received a response we can parse it.

This is a bad assumption to make in the event the server returns an error.

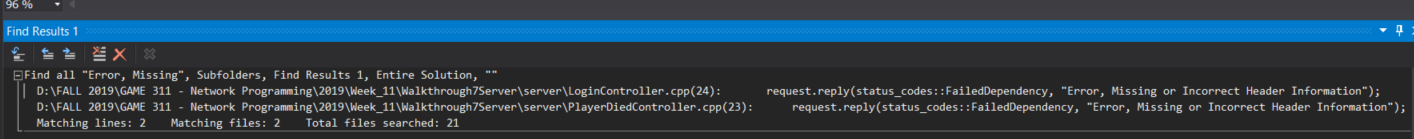
1. We can safe guard our code against this by wrapping our processing of the response inside of a check for status code 200 (SUCCESS)



Run again and ensure no LOG errors are present.

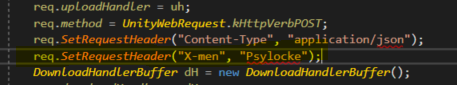
We now need to deal with the missing Header error from the server

1. Open up your Solution for the Server and hit CTRL+SHIFT+F to perform an entire solution search.
2. Type in a portion of the error.
   * This should give us a hint on where to begin looking on the server to solve this problem.
   * You may see a window of results like the following in the bottom:



* + Determine which file is more appropriate using your analytical skills…
  + Note that the Controller is the file printing out the error.
  + Examine the Code and determine why the error is printed out.
    1. Notice the ValidateHeaders Function call. The Controller creates a request object to compartmentalize data. This object is used strictly for parsing the request objects data for Headers and the Body.
    2. If you investigate the contents of that function you should notice that this call is requiring a X-men header be added to the request.

1. Return to your C# Unity projects solution and add a header to the Login Request:



Feel free to enter the name of whatever X-men character you like.

Run Unity again and examine the console outputs.

Has the error been removed?

What is displayed now?



1. Perform another solution search on the server to determine where the error occurred.

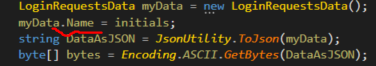
CTRL+SHIFT+F then “Error, Unable to”

Notice that the error is now occurring processing the request inside ProcessRequest

1. Review the code of the function for a moment and try to determine when a false is returned.



Let’s cross compare this with the data we are sending from the request:



1. Fix the client to match with the Server and attempt to run again.

Was it the Login Call Successful now?  
**SIGN OFF WITH INSTRUCTOR FOR WALKTHROUGH 7**

**Complete the Lab 5 available on blackboard.**