University of Derby
The School of Computing and Mathematics

In-course Assignment Specification

Module Code and Title: 5CC515 Networks and Security	
Assignment title: **Designing and building a network application: an online game**	
Assessment tutor: David Evans	Weighting towards module grade: 50%
Date set: November 18, 2013	Deadline: December 12, 2013

Penalty for late submission

Recognising that deadlines are an integral part of professional workplace practice, the University expects students to meet all agreed deadlines for submission of assessments. However, the University acknowledges that there may be circumstances which prevent students from meeting deadlines. There are now 3 distinct processes in place to deal with differing student circumstances:

Assessed Extended Deadline (AED) Students with disabilities or long term health issues are entitled to a Support Plan.

Exceptional Extenuating Circumstances (EEC) The EEC policy applies to situations where serious, unforeseen circumstances prevent the student from completing the assignment on time or to the normal standard. `http://www.derby.ac.uk/files/part_i_exceptional_extenuating_circumstances.pdf`

Late Submission Requests for late submission will be made to the relevant Subject Manager in the School (or Head of Joint Honours for joint honours students) who can authorise an extension of up to a maximum of one week. `http://www.derby.ac.uk/files/part_f_assessment_regulations_ug_programmes.pdf`

Level of collaboration:
This is an individual assignment.
No collaboration with other students or any one else is allowed.

Learning Outcome(s) covered in this assignment:

3. understand the theoretical concepts required for internetworking

Coursework Assignment for the BSc Module of Networks and Security (5CC515)

Assignment:
The development of an network based application complete with report and design notes of 300 words.

Due date/time 2013-12-12T23:59:59Z.

Your assignment MUST be submitted electronically via Course Resources by the due date. Full information on doing this can be found at `http://www.derby.ac.uk/esub`. You should submit a ZIP file named with your student number (e.g., `100999999.zip`) containing four files with the following names:

`pclient.py` containing your player client;

`aclient.py` containing your admin client;

`server.py` containing your server; and

`report.pdf` (or `report.doc` or `report.docx`) containing your report.

and the same report separately.

1 Detailed requirements

This assignment is about building a real network application. You will implement a simple application and appraise the results. Critically, you will make your implementation *interoperable* with others. This is critical for many network applications and is the reason that any web browser can interact with any web server or that you have a choice of email clients.

The network application is the number guessing game that you have been working on in your practicals. You should be familiar with this application and are likely to have built a good portion of it. What follows is a somewhat more formal description of how it works. You should modify your implementation so that it behaves as described here. *Note that the user interaction described here may not be exactly the same as you have seen in your practicals! Wherever there is disagreement, this document takes precedence.*

1.1 The game

You should be familiar with the basic operation of the game. A player uses a client (that you will write) to connect to a single game server (that you will write). For each player the server will pick a random integer between 1 and 20 and the player will attempt to guess this number. The server will notify the client (which will tell the player) if the guess is way off, close, or correct. If the guess is correct then the game ends and the client disconnects.

Any[1] number of users should be able to connect to the server at the same time and the server should multiplex its game logic between them.

The output from a player client might look like this (the numbers are typed by the player):

```
Welcome to the guess the number game!
What is your guess? 5
You are way off.
What is your guess? 8
You are way off.
What is your guess? 3
You are close!
What is your guess? 4
You are close!
What is your guess? 2
You guessed correctly!
```

The server supports administrative access which, for the purpose of this assignment, is limited to finding out the list of current players. Output from running the admin client should look like this (obviously the IP addresses and port numbers will be different for you):

```
The players currently playing are:
192.168.0.2 3065
192.168.4.7 45932
```

The server should produce no output at all. In developing your code you will want to relax this requirement to make debugging easier, but this output should be removed from your final version.

1.2 Network communication

Your server and player clients will communicate using a standard protocol; a skeleton was introduced in lectures. Each message is terminated by `\r\n`, i.e., the bytes `0x0d` and `0x0a` (you are expected to do reading to find out how to do this in Python).

A player client has only two messages that it can send to the server:

`HELLO` A client sends this to the server immediately after it connects. The client may display nothing to the user until it receives the consequent `GREETINGS` message! Once it has, it must print a salutation.

`GUESS␣<int>` The player's most recent guess. `<int>` is a string of ASCII bytes representing the number's digits. Note that there is a space between `GUESS` and the integer.

The server may transmit the following messages to a player client

`GREETINGS` The server sends this after receiving a `HELLO` message from a client.

[1]The resources available will impose a limit but you don't need to worry about this.

`FAR` The most recent guess received by the server from this client is far from the target number.

`CLOSE` The most recent guess received by the server from this client is close to the target number.

`CORRECT` The most recent guess received by the server from this client is correct. After sending this message the server closes the client socket

In addition to player clients there is an *administrative client.* This client connects to the server on a different port from the players. Upon connection such a client sends `HELLO` to the server; the server responds with

```
ADMIN␣GREETINGS
```

(Note the space.) The server *must* not send this to a player client!

After receiving this, the client may issue an administrative command. At the moment only one is defined:

```
WHO
```

Upon receiving this the server should respond with zero or more lines, each corresponding to a current player. Each lime should be of the form

IP-address␣port-number`\r\n`

where each line is made up of two strings (so `196.168.0.4 3000` is an example). After sending the last line the server should close the socket for this admin client. Note that as only one admin command has been defined, the admin client should simply send it and should not get input from the user.

This should be an unambiguous description of the communication, though it hasn't been presented with the same precision as one might find in a standards document or RFC. If you are confused about anything, *please ask.* Making assumptions about protocols is one of the main reasons network software does not interoperate!

The clients and the server should exchange *nothing else* across the network. Your programmes sending any information beyond the protocol messages described above will almost assuredly foul interoperability and result in a loss of marks.

1.3 Security

For the purpose of this assignment we shall worry about secure administrative access to the server. The server should use TLS to ensure that only authorised admin clients connect to the admin socket. A client is considered to be authorised if the certificate that it presents has been signed by the 5CC515 Certificate Authority (CA). You will be provided, via Course Resources, with instructions on how you may obtain this CA's public key as well as a certificate for your own administrative client.

2 What you must do

2.1 Software

You are required to implement a server and two clients (the player client and the admin client) that correspond to the game above. The inputs and outputs of your three programmes should be *exactly* as described above. *This means that from the outside everyone's programmes will look identical.*

As a guide, you may wish to follow these steps.

1. You have been working on the number guessing game throughout the practicals in this module. Get it working using multiplexed I/O so that multiple players may play at once.

2. Refine your client and server so that they communicate according to the protocol described in section 1.2. Remember that unless you apply cleverness that you should use *only* the messages defined!

3. Add support for the administrative client and its one command.

4. Add security to the admin interface using TLS.

5. Add any extra functionality that you think is cool in order to demonstrate your prowess. However, anything you add must not compromise interoperability with other clients and servers!

Your server must listen on the following ports:

Port 4000 player clients

Port 4001 admin clients

2.2 Software testing

Your software will be tested for correctness, meaning that it implements the number guessing game as described and that administrative output for authorised clients is correct. (Unauthorised clients should be treated to no output.)

Furthermore, your software will be tested for interoperability with that written by other students. For example, your player client should work with any other student's server. You are encouraged to test this amongst yourselves prior to submission!

2.3 Report

Alongside this you must write a report about the design of your software. This should be of about 300 words and must be submitted electronically.

Your report will describe the interoperability of your game and, crucially, must present lessons learned for authors of other online games. Its small size means that you should include two sections.

1. **Interoperability**: How does your solution address interoperability with other implementations? How does the game's client/server architecture make this easier? Did you have to take any special care in the construction of your software?

2. **Reflection**: Of all the things that you learned while building this application, what were the *three* most important to you? These things might be technical (such as "the `re` module was really useful") or could be ways of working ("I finally ended up with a comfortable workflow for editing five files at once"). What was the most difficult (and it might not be a terribly important one)? Provide *two* sentences of advice for students taking this module in the future.

In addition, if you implemented any bonus material, you should talk about it in an appendix that is not included in the word limit. Examples of things you might do is a player client in another language (or, for the truly hard core, a single client that can operate in different languages by pointing it at appropriate messages files), or displaying the name of a connecting administrator as extracted from the admin client's certificate. Note that your bonus material *must not* compromise interoperability and you are expected to describe how you did this.

3 Assessment

First, some things that you will *not* be assessed on are:

Invalid input You may assume that input from the user and what comes across the network are well-formed and follow the formats specified in this assignment. In general, robust handling of malformed network data is an important part of building applications but it is a mammoth task to get it right and can involve quite a lot of code.

Code formatting Your choice of variable names, commenting, and indentation style (within the limits of what Python will tolerate) are not important for this assessment.

Your grade will be computed using the following weighting:

Deliverable	Weight
Report: interoperability	20
Report: reflection	15
Software	65

When discussing interoperability, you should be able to come up with at least *three* ways that your software addresses it and at least *one* thing that you had to take special care over. The 20 marks will be divided equally between these four items and each will be assigned a grade according to table 1. Your refection, comprising *three* things, will be graded in a similar way, with 5 marks allocated to each.

The 65% of your grade that comes from your software will be allocated as follows.

20% Can a single client play the number guessing game?

20% Can concurrent clients play the number guessing game?

10% Can authorised admin clients use the `WHO` command to get useful information?

15% Do the above work when the client(s) and server are not written by you?

Half of the marks in each of these four categories will be assigned to predictable output in the small (e.g., does it print something reasonable in response to my guess?) and half for behaviour in the large (such as whether it is possible to win the game).

Centre of grade band	Criteria
95%	Critically and comprehensively answers the questions posed by the assessment specification, including justification for claims that are made. There is a clear and logical structure to the analysis and justification. Shows professional level of understanding of the topic. The answers present a novel view with clear synthesis of a wide range of perspectives.
85%	Critically and comprehensively answers the questions posed by the assessment specification, including justification for claims that are made. There is a clear and logical structure to the analysis and justification. Shows professional level of understanding of the topic. The answers present a novel view.
75%	Critically and comprehensively answers the questions posed by the assessment specification, including justification for claims that are made. There is a clear and logical structure to the analysis and justification. Shows professional level of understanding of the topic.
65%	Critically and comprehensively answers the questions posed by the assessment specification, including justification for claims that are made. There is a clear and logical structure to the analysis and justification.
55%	Comprehensively answers the questions posed by the assessment specification, including justification for claims that are made. There is some structure.
45%	Addresses the questions posed by the assessment specification, including justification for claims that are made.

37%	Identifies the issues needed for answers to the questions posed by the assessment specification but doe not provide answers themselves. The text is descriptive and lacks structure.
17%	Little of merit

Table 1:

A Referencing system

The assignment specification does not require that you use any references but you may want or need to. You are free to use whatever referencing system you want provided that you

1. apply it consistently; and
2. are thorough.

In other words, pick one referencing system and use it for all of your citations and use only that. Furthermore, ensure that each citation includes enough information for the reader to find the reference.

Many assignments at this university require that you use the Harvard referencing system. While it is not required here, using it correctly and consistently will ensure that you satisfy the above two criteria.