Web Science

Quiz 1: March 2, 2017

100 points max

Place your name on the top of the document in the header
Enter your answers directly into this document (with the exception of #2 and #3)
All answers should be in be in Your Own Words, and use proper grammar
Make sure your answers use an alternative font and/or color
Save the document as
ITWS4500-S17-Quiz1-*yourname*-quiz1.docx
Place all documents/files including this one in a folder named
ITWS4500-S17-Quiz1-*yourname-yourRCSID*
When finished with the quiz, zip your folder and all related files into a file named
ITWS4500-S17-Quiz1-*yourname-yourRCSID*.zip
And submit it to LMS

1. **Frameworks** (25 points): (Answer in complete sentences, explain your answers)

   a. (5) What is MongoDB? How does it differ from MySQL (aka MariaDB)?
   MongoDB is a DBMS (database management system) built in JavaScript.

   It differs from MySQL in that data is stored in documents (character large object format), as opposed to relations (binary large object format).

   b. (5) What is npm? How is it used? What it used for?
   npm is an abbreviation for Node package manager. It is used to keep track of, as well as install, packages built for NodeJS. These packages can be installed globally to a machine running NodeJS ("`npm install -g foo`"), or in a node_modules directory in a project that uses NodeJS packages ("`npm install –save foo`").

   c. (5) What is nvm? How does it work? Why is it used?
   nvm is an abbreviation for Node version manager. It is a Bash script that is meant to manage multiple active instances of NodeJS. It can also keep NodeJS installations at a specific version, so that new NodeJS releases don't break previously written code.

d. (10) Describe the difference between Front-end and Back-end frameworks. Provide at least 2 examples for each in your answer. (Be clear in your decriptions, ie 'why is it back/front-end?')

Front-end frameworks provide developers a means with which to manipulate client-side webpages, such as creating user interfaces. Back-end frameworks are typically abstractions of server-side application logic, which might allow developers to integrate user interfaces with databases, or open sockets, for example.

The word "front" is associated with client-side programming because the user sees its effects, while "back" is associated with server-side programming because the user may not see its effects.

2. **Node.js** : (40 points) Create a webserver in node.js, using express – (NOT express-generator), which will serve a simple HTML page with an input filed and a button labeled 'Run' when GET request is received on http://localhost:3000. Upon entering a zipcode and clicking the button, the page server should get the current temperature for that zipcode and output a sentence that says whether it is Freezing (<=0C), Cold (btw 0 and 10), Warm (btw 11 and 25) or Hot (>25) – display the corresponding message in a unique color for each category. Include a button that allows the user to refresh the page and enter a new zipcode.

3. (15) Build an npm package.json file for Q2. If we run it, there should be no errors or warning when we try to install & run your code from #2 above. (You may assume your application name is *Quiz1Server*)

4. (20) Explain *in detail* what the following code does; (also add *stylized* comments to the code explaining what each line does)

```
// Use the net module, which contains functions for creating
server and client streams
var net = require('net')

// Maintain a list of open websockets
var sockets=[];

// Create a new server
var s = net.Server(function(socket) {
   // Add some socket to the list of websockets
   sockets.push(socket);

   // Whenever data is transmitted on this socket,
   socket.on('data', function(d) {
      // write that data to all open sockets
      for(var i=0; i<sockets.length;i++) {
         // except itself
          if (sockets[i]==socket) continue;
          sockets[i].write(d);
      }
   });

   // When a socket's connection has ended, remove it from the
list of sockets
   socket.on('end', function() {
      var i=sockets.indexOf(socket);
      sockets.splice(i,1);
   });
});

// Make the server listen for incoming connections on localhost
port 8088
s.listen(8088);
```