**Quiplash – Protocol Planning**

About the protocol:

This protocol is used for the Quiplash game in this project.

The protocol services the game and handles the communication between the game and the browser.

This protocol is very simplistic. It has only four headers that allow the game to run smoothly.

Protocol overview:

* The protocol is text-based.
* The protocol is very minimal but allows much needed flexibility for the game.
* The protocol is header based.
* The length of the header content does not matter.
* The order of the headers also does not matter.
* All the headers and header-contents will be in string format.
* The protocol will always be sent in a http message body.  
  The protocol does not need to send it’s length because it is already included within the http message (Content-Length).

Message format:

* The protocol uses the JSON file format.
* Every key in the JSON is a header.
* Every value in the JSON is the header-content.

**EXAMPLE:**

{

“header”: “header-content”,

“other header”: “other header-content”

}

Headers:

1. **time-left** – tells the client the time left until the game is done.
2. **txt** – this header is a “dynamic” header. It is the body of the protocol. It can contain any string to send to the client or vice versa.
3. **start-game** – holds a Boolean that says whether the game started or not. This can be used not only for the start of the game but also for moving scenes and relocating the player (example: the vote page uses this header as well to know when to move on from the vote and to the next vote.  
   the header-content must be in lower case formatting – “true” or “false”.  
   **The http message that has this header in the body must also have a location header to the next page that the player needs to go when the header is set to true.**
4. **location** – when using sse the base Js lib does not allow to read the http headers. The location header in the protocol allows to use it when needed with sse.

Examples:

1. {  
    “start-game”: “true”  
    “time-left”: “192.343453452”  
   }
2. {  
    “txt”: “B”  
   }

HTTP FULL EXAMPLE:

POST /username HTTP/1.1\r\nContent-Type: text/plain\r\nContent-Length: 12\r\n\r\n{"txt": "B"}