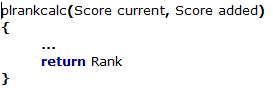
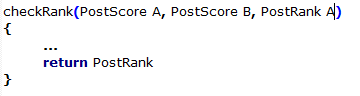
**Outline of core logic architecture**

* **plrankcalc**



This will take the current users score, and the new score and return the rank of the player. Rank thresholds will be defined in the method

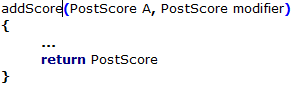
* **Checkrank**



The idea is that a posts rank will determine where it is on the answer page. Every so many minutes, posts will check their score compared to those 1 rank higher than them, or when they have been voted on. If the post A, has a higher score than post B, it will return the new rank. Otherwise it will return the old rank.

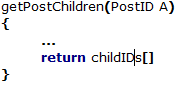
(Need to discuss how will the ranks change, what kind of logic should be used)

* **addscore**



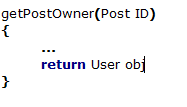
This will be used to modify a posts score, by either adding 1 for if it’s a vote, or for adding or subtracting a certain amount to the score. Will do checks to make sure that the score is not going out of bounds.

* **Getpostchildren**



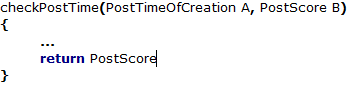
This will retrieve all the IDs of answers to a question post and return that information as an array. (Not sure how to deal with time of answer sorting, maybe happens later on? Will it not matter due to the database?)

* **Getpostowner**



This will get information on the user who posted the specific post.

* **checkPostTime**



This will compare the posts time of creation to the current time and then determine how long the post has been there. Depending on its age the posts score will go down by a certain amount. The exact formula for this is yet to be determined.

**Check rank**

Notes 3/24/16

* Function that gets an array of the 10 newest posts and then returns them as an array of json objects
* Have the users rank send back color information to the front end as a json object
  + Have that be part of the user object that is being sent back
* Modify function
  + This will be called when th user makes a modification, and will pass a new question json object to the function and it will then get the relevant information and send that information to the database
  + Will need one for if it is a question or if it an answer
* Create a new user function
  + If a user already exists then send back an empty json object
* Make a question
  + Question json object
* Make an answer
  + Answer json object
* Sign in information
  + User json object
  + If it failed send an empty json object to the front end
* Search function
  + Gets a sentence then return a list of 10 questions

Other notes:

Have a set list of tags be strings for the frontend

Search would take a sentences, see if there are tags, then check the titles of questions and see if they are similar

4/7/16 notes

For persitant logic

* Function to sign in
* Function to get information of a user
  + Sends name of user to the database and gets back a json object
* Function to get all the answers of a question
  + Returns a list of the users in a Json object
* Make a new user
* Search function
  + Needs to do several queries
  + Return top ten
  + Needs to query the tags
  + Send the whole string
  + Empty, then return first 10 questions
* Post question
  + Grab all the information from the webpage, create it into json object and send to database
* Post answer
  + Grab all information and send that to database
* Modify
  + Send a question object
  + Persistent logic does an update query
* Rank increase
  + It will send a user name to the database
  + Keep it simple for now, and just go with questions and answers, but do keep in mind that there is a problem of gaming the system and that we should be checking the rank of each answer post as well.
* Vote
  + Give the id of the answer
  + Vote is given by 1
* Display answers for a question
  + Gives a question id
  + Returns a list of all the answers

Notes:

URL for a question is the webstie name/ question id

Google has its own search function going to look into that

Extra security if we have time.

Core logic only be communicating with the front end by using one class