Week 13

Assignment 1:

This assignment requires us to automate the process of downloading and executing the downloaded file – in this case a .java file. There were several steps which were to be carried out before the final java file could be executed.

Step 1: Download the PostgreSQL driver for connecting to the database:

In this task we had to connect to the database and run a select query to fetch the result which is a link to a website from which the next step is to be executed.

Step 2: Finding the Hidden HTML:

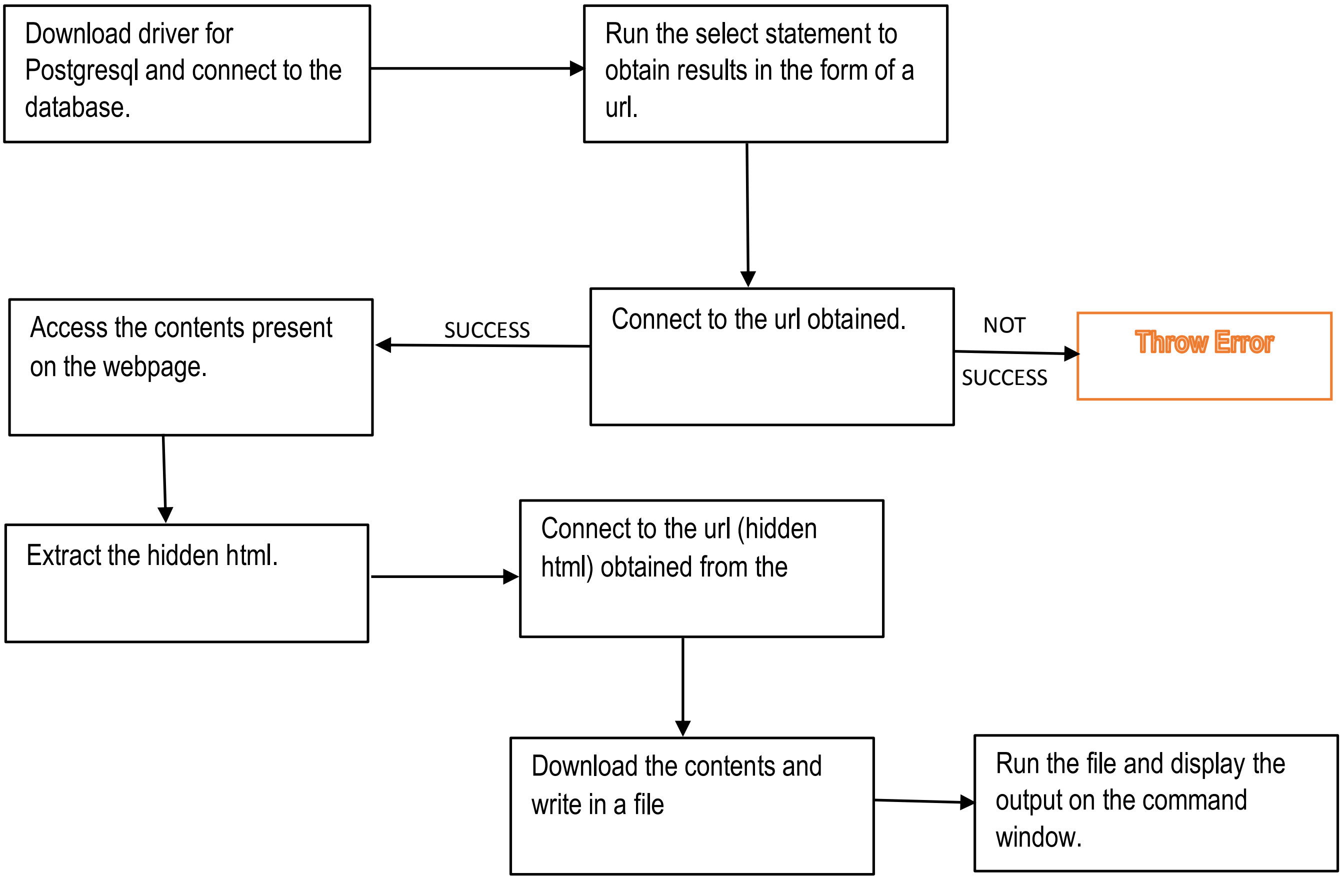
We extracted the hidden html string from the contents we achieved form the previous step by using regex.

Step 3: Download the target file:

From the hidden html, we acquired the link on which the contents of the target file are present. We established a connection to the url and initiated a data stream to download the data off the webpage.

Step 4: Execute the file:

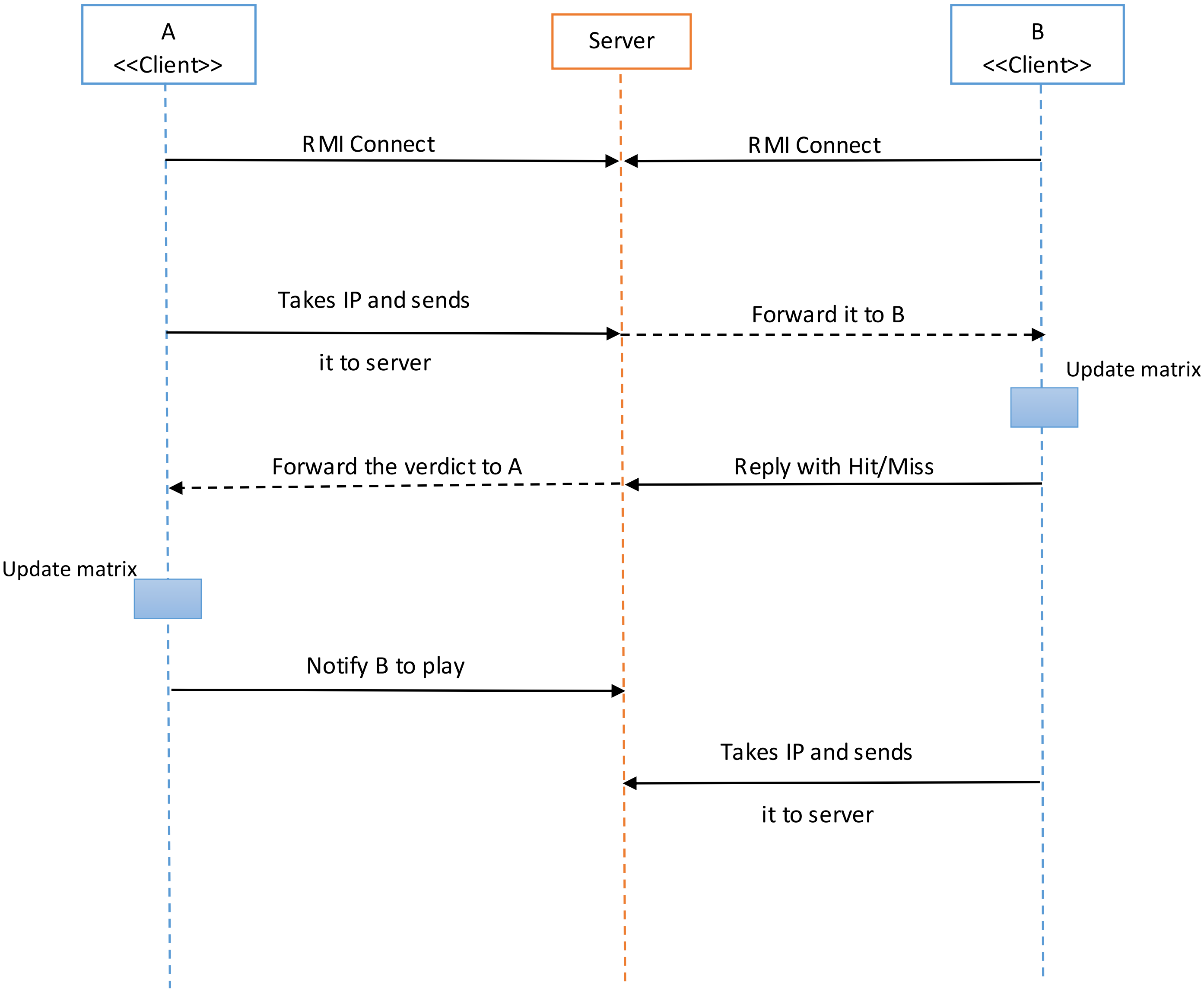
After downloading the contents and copying into a file on the system, we executed the file and displayed its output. This was done autonomously, by using initiating a runtime process, which executed the java file and then streaming the output data to be displayed on the command window.


Download driver for Postgresql and connect to the database.
Run the select statement to obtain results in the form of a url.
Connect to the url obtained.
SUCCESS
NOT SUCCESS
Throw Error
Access the contents present on the webpage.
Extract the hidden html.
Connect to the url (hidden html) obtained from the
Download the contents and write in a file
Run the file and display the output on the command window.

Assignment 2:

This assignment was a modification from the previous weeks' assignment on 'BattleShip'. The prior version of this was played using a TCP protocol, this week we had to redesign the model for it to work with RMI.

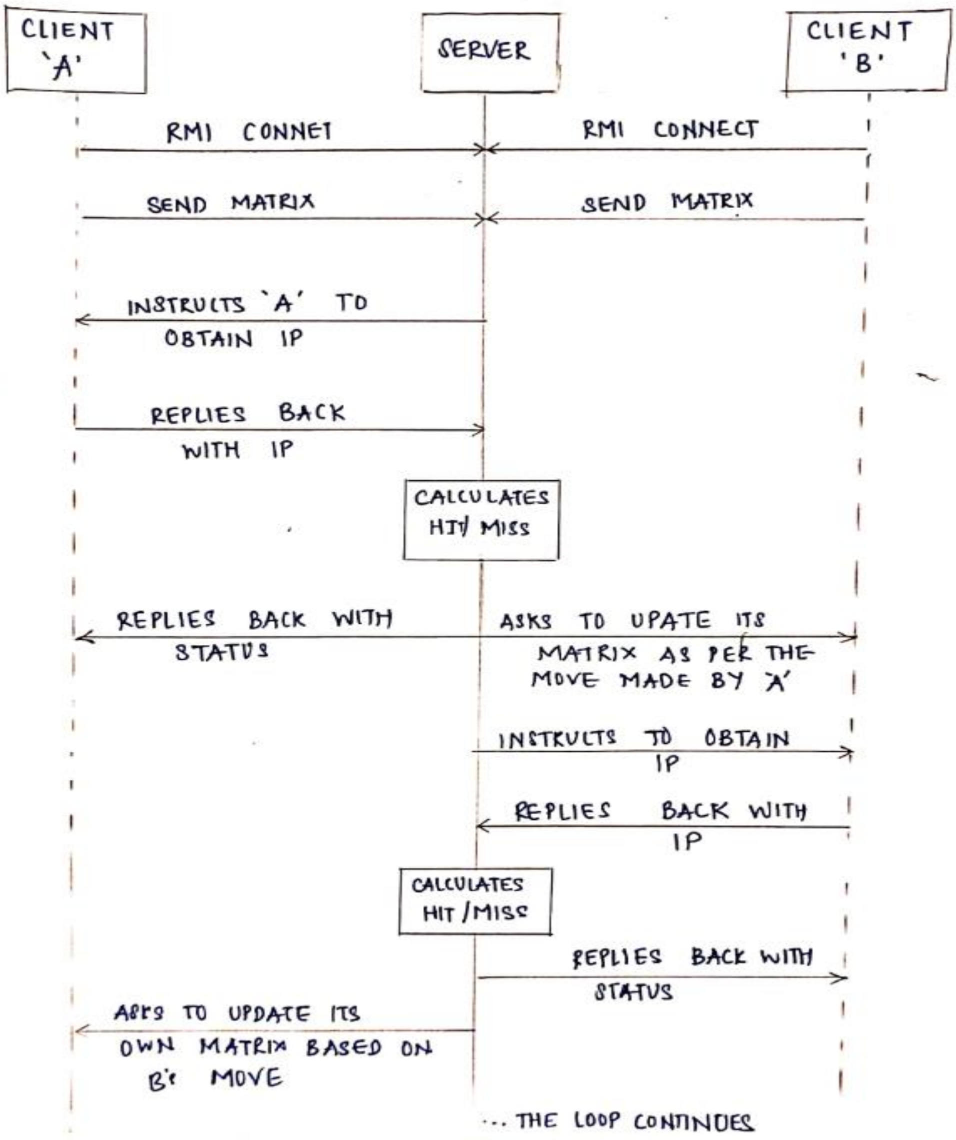
Initial Design:



This process we proposed initially was a bit tedious, for individual player(client) we were updating their matrix, and then were sending the results back and forth the players, and then the said player was updating its own matrix with the result obtained.

We decided to improve upon this design and the server itself calculates the Hit or miss ratio and notifies the players of their wins or losses.

Final design:


CLIENT 'A'
SERVER
CLIENT 'B'
RMI CONNET
RMI CONNECT
SEND MATRIX
SEND MATRIX
INSTRUCTS 'A' TO OBTAIN IP
REPLIES BACK WITH IP
CALCULATES HIT/ MISS
REPLIES BACK WITH STATUS
ASKS TO UPATE ITS MATRIX AS PER THE MOVE MADE BY 'A'
INSTRUCTS TO OBTAIN IP
REPLIES BACK WITH IP
CALCULATES HIT /MISS
REPLIES BACK WITH STATUS
ASKS TO UPDATE ITS OWN MATRIX BASED ON B's MOVE
... THE LOOP CONTINUES