Heavy Hitters Animation Snippets

Neelmay Desai
Rutgers University
Piscataway, NJ, USA
Email: neelmay.desai@rutgers.edu

Sairama Rachakonda
Rutgers University
Piscataway, NJ, USA
Email: sr1122@scarletmail.rutgers.edu

Nilay Chakraborty
Rutgers University
Piscataway, NJ, USA
Email: nilay.chakraborty@rutgers.edu

***Abstract*— Streaming algorithms are the one of the most enthusiastically studied algorithms of the day because of their practical usage to examine a stream of data such as packets of network. We had the inquisitiveness to know about how can an algorithm process such a large stream of data with less memory size than the data size and with limited processing time. To share the same experience with other users and learners, we came up with a thought to select the streaming algorithms and animate them to a naive user or algorithmist who is interested in learning streaming algorithms and harness the power of these algorithms.**

I. Project Description

There are many algorithmic animations available online for the algorithms which deal with problems where the size of data fits in the RAM or in the local disk. For example algorithm for sorting, comparing, finding median, finding frequent items on a small data set. But in our algorithmic animations, we show algorithms for processing large data streams that can be examined only in few passes generally only once per item. Another limitation for these algorithms is that the memory available for computation is less than their size. Scores of papers providing solutions to such challenges are available. Through animation, we show how such problems have been solved given the environment described above. However, we animate only few algorithms that process streaming data.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

A. Stage1 - The Requirement Gathering Stage.

Deliverables for Stage1 are as follows: As this tool provides the animation of the algorithms, its just a read only tool. Hence we have only two users of this system namely - 1)Learner 2) Admin.
1)Learner - The learner, shall visit the webpage of the tool,shall login the tool and learn the streaming algorithms which are provided by the admin.
2) Admin - The admin shall add new features into the tool. Like the admin shall add few more animations of popular streaming algorithms.

Deliverables for Stage1 are as follows:

- The Learner user shall not have any rights but can view the algorithm through visualization of the algorithm. The admin will have the data as well as update rights.
- The user's interaction modes: The Learner user shall select an algorithm to learn from the provided list of streaming algorithms.
- The real world scenarios:

  - USER TYPE: Learner
  - Scenario1 description: A learner who is a beginner to algorithms shall learn by viewing the animation of the algorithm. He/she can select the algorithm which he/she wants to learn. That person can learn algorithms much faster and easily by seeing the animation
  - System Data Input for Scenario 1: Algorithm name.
  - Input Data Types for Scenario 1: The data type can be a string or drop-down menu query.
  - System Data Output for Scenario 1: An animation of the algorithm.
  - Output Data Types for Scenario1: Animation/Video

  - Scenario2 description: The learner can view the pseudo code of the streaming algorithm.
  - System Data Input for Scenario 2: Algorithm name.
  - Input Data Types for Scenario 2: The data type can be a string or drop-down menu query.
  - System Data Output for Scenario 2: The pseudo code of the algorithm.
  - Output Data Types for Scenario 2: A text file.

  - USER TYPE: ADMIN
  - Scenario1 description: The admin wants to increase the scope of the systems and thinks of adding a few new algorithms to the system.
  - System Data Input for Scenario 1: The new Algorithm
  - Input Data Types for Scenario 1: Code
  - System Data Output for Scenario 1: An animation of the algorithm.
  - Output Data Types for Scenario1: Animation/Video

  - Scenario2 description: The admin wants to decrease the number of algorithms in the system.
  - System Data Input for Scenario 2: Algorithm name by an identifier
  - Input Data Types for Scenario 2: String, integer
  - System Data Output for Scenario 2: System with

the list of algorithms without the algorithm which is currently deleted.

- Timeline : 5-6 weeks

  i) Requirement gathering and analysis - 5-7 days
  ii) Design - 1 week
  iii) Implementation - 2 weeks
  iv) UI and Testing - 2 weeks

- Division of Labor:

  Neelmay: System design, Modeling, back-end implementation, Management, Testing and Documentation

  Santhosh: Requirement gathering & analysis, UI design & implementation and Maintenance

  Nilay: Animation Design and Middleware implementation

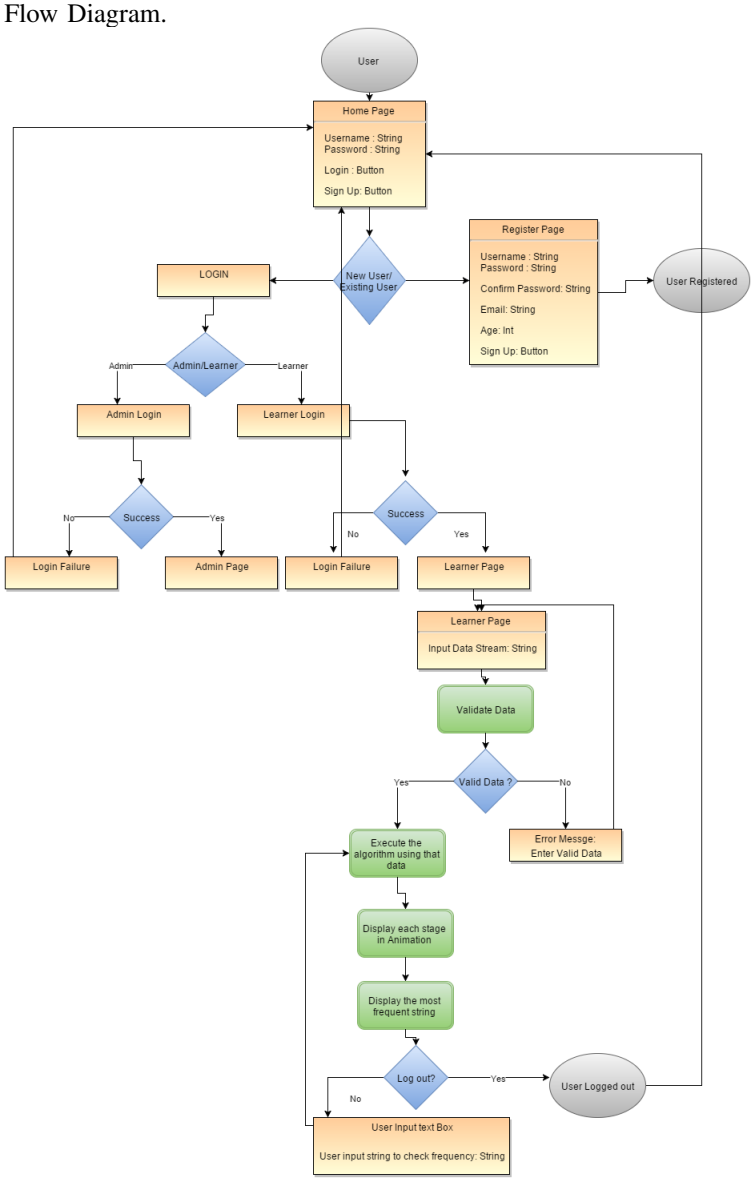
B. Stage2 - The Design Stage.

- The motto of the application is to show 'how to compute the k frequent items in a large data stream in one pass and with relatively small memory than the size of the data and within less time!' These k frequent items are also called as heavy hitters. The heavy hitters of the data stream shall be computed and this procedure shall be animated with good visual cues, so that even a naive user can comprehend it.

  The user (the learner or the admin) will login by entering the login credentials and enter their specific home pages (both learner and admin will have different home pages). If the login fails, they will be redirected again to the previous page. If the login is successfull, the learner will be able to see the animated output of the algorithm and the admin will be able to make any changes needed. Either the user can provide a data stream on which he can visualize the heavy hitter algorithm or choose the default data stream provided by the admin.

  The detailed flow diagram is given as follows. It includes all the important states and conditions required in the system flow.The over all time complexity of the system is O(nlogn) where n is the frequency parameter which the uses chooses. The main and important thing in streaming algorithms is the space complexity. The space complexity of the algorithm is O(log(m) +log(n)). This is because to represent the keys of the alphabet of size n, over which the stream is defined, we require log(n) bits and to represent the counter values in the associative array, we require log(m) bits where m is the size of the data stream.

- Flow Diagram.



- High Level Pseudo Code System Description.
  - Pseudo code for k frequency estimation
- Algorithms and Data Structures. The key problem we need to solve is to pass the stream of data just once, and apply some algorithm during this pass to get an approximate answer for the problem. In our case, we need to pass the stream once and find whether a word is frequent or not. We use the ubiquitous Misra Gries Algorithm to do this.

  We shall have an initialization section, executed before we see the stream, a processing section, that gets executed for every time the algorithm sees a token in the input stream and an output section, where we answer queries about the stream, perhaps in response to a given query.

Algorithm 1: k Frequencey Estimator, Misra Gries Algorithm

Input: A stream $\sigma = \langle a_1, a_2..., a_n \rangle$
of length m where each $a_i \in [n]$
and a frequency parameter k
Output: a data structure A, such that for any j $\in [n]$,
we can return A[j]= $f_j such that f_j > m/k$
Initialize: A $\leftarrow (empty\ associative\ array)$
Process:j
if j $\in A(keys) A[j] \leftarrow A[j] + 1$
else if number of keys of A greater than k-1 then
  A[k] $\leftarrow 1$;
else
for every l $\in keys(A)$
  A[l] $\leftarrow A[l] - 1$;
  if A[l]=0 remove l from A;

The algorithm uses a parameter 'k' that controls the quality of the answers it gives. It maintains an associative array A, whose keys are tokens seen in the stream, and whose values are counters associated with these tokens. We keep at most k-1 counters at any time. Whenever we receive a token from the data stream we look up the associative array with the token. If we find a location, then we increment the value of the counter at that location. If we didnât find any location we perform a check before adding a new location for the token in the associative array. The check is , if there are less than k-1 counters (or locations) in the associative array, then we add a new location for the token and set its counter value to 1. Else if there are greater than or equal to k-1 counters in A, then we decrement the counter values of each of these locations by 1. Now after decrementing the counter values, if for any location the counter value is zero, then we remove all such locations from A. Now add a new location to the token and set the counter value to 1.At any point of time, the user can query the associative array A and get to know the k-frequent items.

Data Structure: The data structure used for the associative array A is a balanced binary search tree. We use this data structure because any operation can be performed with O(log n) time complexity.

- Flow Diagram Major Constraints.
  - Integrity Constraint 1
    Input data size limitation: The application is basically a learning platform. So huge data stream can't be supported. Our project shows the animation for a relatively comparable unit of the huge data sets.
  - Integrity Constraint 2
    Mandatory Value Constraint: The application should be provided with a data stream. Either the user may provide a data stream or he can choose the admin to provide the data stream to visualize the algorithm. The data stream is mandatory.
  - Integrity Constraint 3
    Frozen Value Constraint: Once a data stream is provided to the algorithm, it can't be changed until,the user resets the application to run for a new data stream. While the visualization is running, the user can't change the data stream. He needs to reset the application.

C. Stage3 - The Implementation Stage.

Specification regarding the languages and programming environment used for the implementation are as below:

1) The User Login page is designed using MVC architecture.
   - View- HTML/CSS
   - Controller/Model - Java
   - Backend - Oracle 12c
   - ORM Framework - Hibernate
2) The algorithm animation snippet is implemented using Javascript, HTML and CSS.
3) The server used is Apache Tom Cat
4) IDE used is Eclipse Mars and SQL Developer..

The deliverables for this stage include the following items:

- Sample small data snippet.
  The input data for this animation project would be a stream of text data and the minimum frequency. For example few paragraphs, from any Wikipedia page and any positive integer. The following text data is from wikipedia page

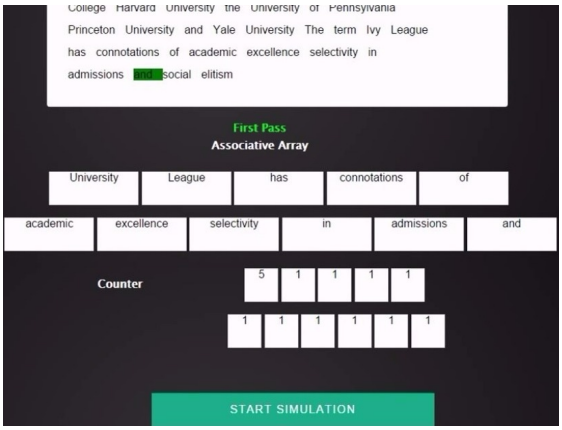
  Input Data Stream:

  The Ivy League is a collegiate athletic conference comprising sports teams from eight private institutions of higher education in the Northeastern United States. The conference name is also commonly used to refer to those eight schools as a group.[2] The eight institutions are Brown University, Columbia University, Cornell University, Dartmouth College, Harvard University, the University of Pennsylvania, Princeton University, and Yale University. The term Ivy League has connotations of academic excellence, selectivity in admissions, and social elitism.

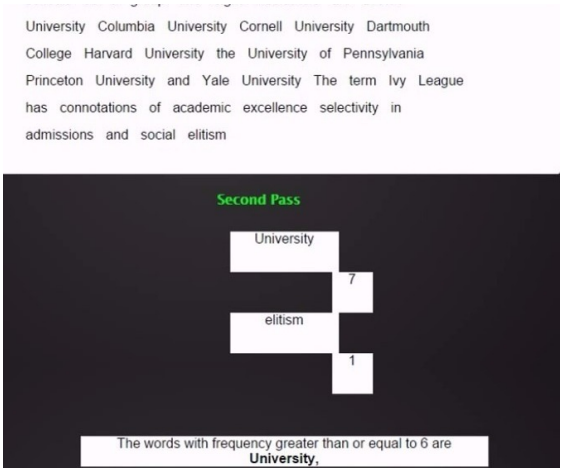
  Input Minimum frequency: 6

- Sample small output
  1) The algorithm outputs an animation of the working of the algorithm on the user provided input.
  2) The list of words, having a frequency greater than the user given minimum frequency input.

     The the screenshot of the output is:



The Second Pass



The workinng code for the animation and the algorithm is as follows:

- Working code

The Working code for the Animation

```
var frequentElements= new Map();

function runAnimation(stream,freq)
{
    //Replacing all the special characters and extra spaces with a single space
    stream=stream.replace(/[^A-Z0-9]/ig, " ");
    stream=stream.replace(/\s\s+/g, ' ');
    stream=stream.trim();

    //Splitting the stream into multiple words
    var elements=stream.split(" ");

    //Calculating the K value
    var k =parseInt(elements.length/freq);


    var list = document.getElementsByClassName("stream-string");
      for (var i = 0; i < list.length; i++)
      {
       list[i].setAttribute("id", "strings" + i);
      }

    var n=0;
    var i=0;

    //Setting up an animator for the algorithm
    var intervalId=setInterval(function()
    {
        document.getElementById("mybutton").disabled = true;
        n=n+1;
        if(i<0 && i<=elements.length)
            {
            var streamDivOld=document.getElementById(("strings"+(i-1).toString()).toString());
            streamDivOld.style.backgroundColor="#fdfdfd";
            }

        if(i<elements.length)
            {
            var streamDiv=document.getElementById(("strings"+i.toString()).toString());
            streamDiv.style.backgroundColor="green";

            //calls the actual algorithm method
            runAlgorithm(elements[i],k);

            var list = document.getElementsByClassName("loader-block");
              for (var m = 0; m < list.length; m++)
              {
               list[m].innerHTML+="";
              }

            var list1 = document.getElementsByClassName("loader-block1");
              for (var m = 0; m < list1.length; m++)
              {
               list1[m].innerHTML+="";
              }

            var p=0;
            frequentElements.forEach(function(value, key)
            {
            var assoDiv=document.getElementById("array"+p.toString());
            assoDiv.innerHTML+key.toString();
            var counterDiv=document.getElementById("counter"+p.toString());
            counterDiv.innerHTML+value.toString();
            p++;
            }, frequentElements);

            }

        if(i==elements.length)
            {

            clearInterval(intervalId);
            document.getElementById("mybutton").disabled = false;
            document.getElementById('state').style.display = "none";
            frequentElements.forEach(function(value, key){

            },frequentElements);

        var count=0;

        var finalDisp=document.getElementById("final-display");

        finalDisp.innerHTML=finalDisp.innerHTML+ "<div class='asso' style='color: #0AF71E;'>Second Pass</div>";

        //Second pass
        frequentElements.forEach(function(value, key)
        {
            count=0;
            finalDisp.innerHTML=finalDisp.innerHTML + "<span><div class='loader-block'>"+key+"</div><br/>";
            for(var j = 0, len3 = elements.length; j < len3; j++)
            {

                if(elements[j]==key)
                {
                    count++;
                }
            }

            finalDisp.innerHTML=finalDisp.innerHTML + "<div class='loader-block1'>"+count.toString()+"</div><br/></span>";

            if(count<freq)
            {
                frequentElements.delete(key);
            }
        },frequentElements);

        finalDisp=document.getElementById("final-output");

        finalDisp.innerHTML="<div class='f-out' id='final-data'>The words with frequency greater than or equal to "+freq+" are </div>";

        finalDisp=document.getElementById("final-data");
        frequentElements.forEach(function(value, key){
        finalDisp.innerHTML=finalDisp.innerHTML+"<span style='font-weight:bold;'>"+key+",</span>";
        },frequentElements);
        frequentElements.clear();
            }

        i++;

    }, 3000);
}
```

The runAlgorithm() function of the animation project

```
//Algorithm
function runAlgorithm(element,k)
{
    //More than once
    if(frequentElements.has(element))
    {

        frequentElements.set(element,parseInt(frequentElements.get(element))+1);
    }
    //First entry and Associative array size in limit
    else if(frequentElements.size <k-1)
    {

        frequentElements.set(element,1);

    }
    //Asoociative array size not in limit and new entry
    else
    {

            frequentElements.forEach(function(value, key) {
                frequentElements.set(key,frequentElements.get(key)-1);


            if(frequentElements.get(key)==0)
            {

                frequentElements.delete(key);
            }
            }, frequentElements);

    }
}
```

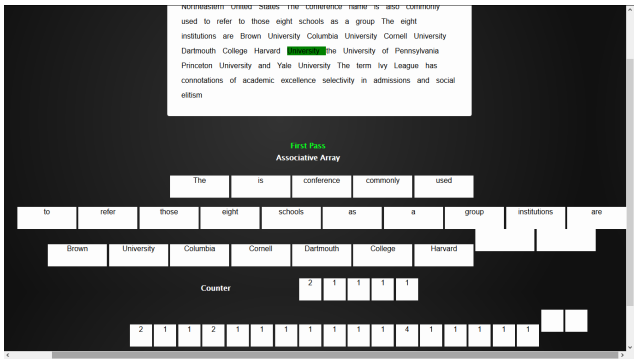
- Demo and sample findings
  - Data size: 12.6 MB RAM size;
    Streaming: The input is a streaming data;
  - Some interesting findings: By giving the minimum frequency as N/2, where N is the number of elements or words in the set, we can find the majority element of the set. This has few practical applications like finding the famous pop singer in the recent tweets, finding the winner of an election analysing the votes. We can design a querying system, where the user can query for a particular word of his interest and the algorithm returns if the word is a frequent word or not.

D. Stage4 - User Interface.

Describe a User Interface (UI) to your application along with the related information that will be shown on each interface view (How users will query or navigate the data and

view the query or navigation results). The emphasis should be placed on the process a user needs to follow in order to meet a particular information need in a user-friendly manner. The deliverables for this stage include the following items :

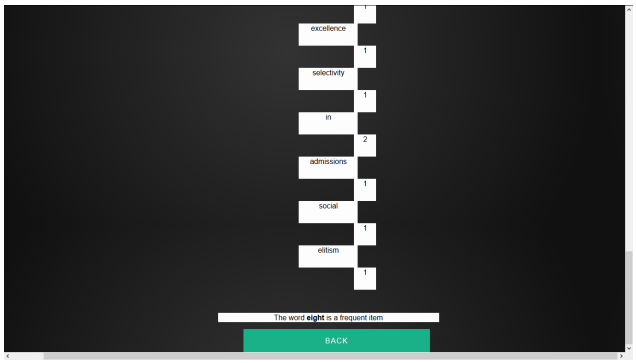
- The user would interact with the system by using the mouse and the keyboard. The input data stream will be given by the user via keyboard. The ”START SIMULATION” button would be clicked using a mouse.
- During the user login, if the user enters the incorrect credentials, the system will pop up an error message and would prompt the user to enter the credentials once again.
- The animation will be displayed according to the inuts provided by the user
- The system would only accept stream of data in form of text. No other data would accepted for animation other than text data. If the user ttepmts to enter the data other than text (i.e. image, audio, video, etc, the system would pop an error message stating that the input is not supported.) The screenshot of the animation is as follows:



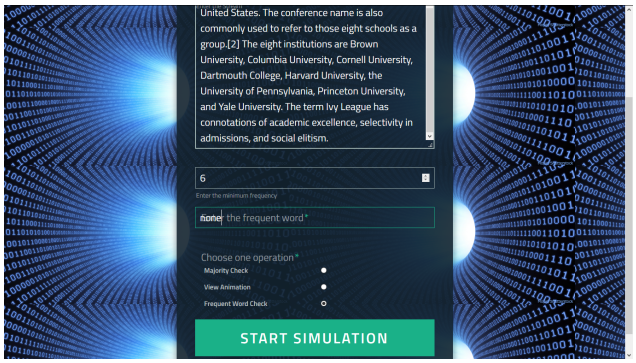
The deliverables for Stage4 are as follows:

- Once the user has entered the user credentials, the user then arrives at the main screen of the application where the user has to enter the data stream for animation. Once the user enters the data stream, the minimum frequency and the operation, the animation of the Misra Gries algorithm will start after the user presses the ”Start Simualation” button. The screenshot of the above step is as follows:

- If the user selects ’Majority check option’, the system will find the words from the given stream of text data which occurs majority of the time. The screenshot for that is given as follows

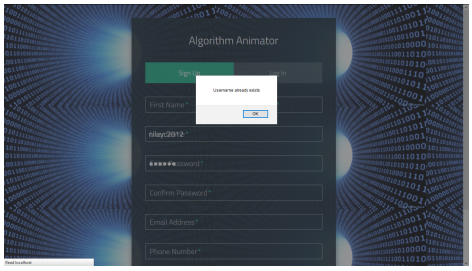


- If the user selects ’Frequent Word Check’ option, the system will find the words from the given stream of text data which occurs frequently. The user need to enter which frequent word he/she might like to see.



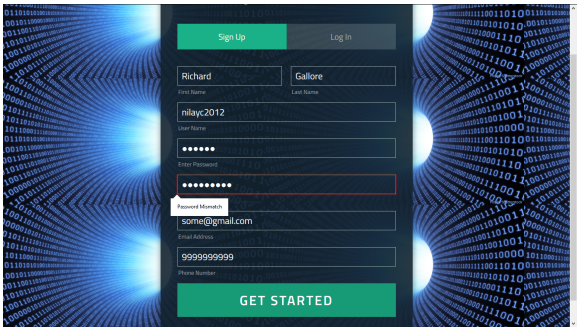
- The error messages popped up by the system when something goes wrong by the user. This messages are basically popped to tell the user that something has gone wrong that needs to be corrected. The following are the error message which gets popped up by the system):
  - The error message: ”Username already exists”
  - The system will show an error if a user tries to sign up with the same username which exists already. In this case the user might have to create another username.

    The screenshot of the above error message is given as follows:



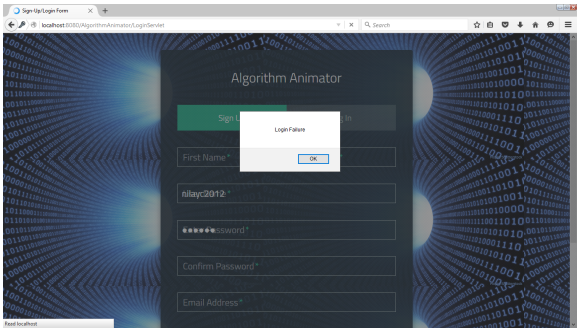
– The error message: ”Password Mismatch”
– The system will show this error message when the user enters an incorrect password.

The screenshot of the above error message is given as follows:



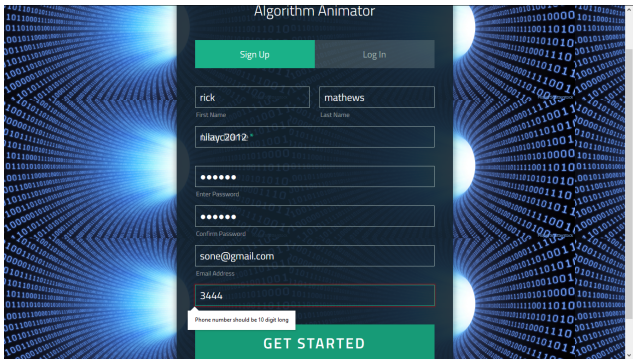
– The error message: ”Login Failure”
– The system will show this error message when the users login fails due to any reason such as invalid credentials, server failure, etc.

The screenshot of the above error message is given as follows:



– The error message: ”Phone number should be 10 digit long”
– The system will show this error message when the user enters a phone number which is shorter than 10 digits.

The screenshot of the above error is given as follows:



II. Project Highlights.

- The main aim of the project is to animate and depict How to find the frequent words which are above certain threshold value, in a streaming data and what value can the user get from such an informationâ.
- We have modified this algorithm in such a way that, the algorithm returns all frequent objects whose frequency is above the threshold value.
- We have algorithms which can find an object, which has occurred the most number of times in a stream of data. One such algorithm is the Misra Gries algorithm which finds the Majority element in a stream of data.

III. References.

- http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides
- http://utexas.influuent.utsystem.edu/en/publications/finding-repeated-elements(6ac4aeb3-8630-4aed-907b-23120087b927).html
- http://www.cc.gatech.edu/ stasko/papers/dag01-aa.pdf
- https://www.cs.utah.edu/jeffp/teaching/cs5955/L12-Heavy-Hitters.pdf