Abstract— With the advancement of information technology, we have got an opportunity to change the learning experience of students. E-learning systems such as Learning Management Systems (LMS), Course Management Systems and Students Management Systems have been emerged as a result of this, aiding a vast variety of education institutes around the globe. Yet, the quality assurance processes are still dependent on conventional student feedback mechanisms that have their inherent drawbacks. This has become a major barrier for quality improvement of the learning experience of students. We have introduced a solution for this through a crowdsourcing based perception capturing and analysis platform that can be adapted in many scenarios seamlessly. The design and implementation of backend analysis algorithms have been discussed along with the results gained from this project.

Keywords— quality assurance; perceptions; analysis; crowdsourcing;

I. INTRODUCTION

The earliest web; i.e. web 1.0, restricted content generation and sharing to site authors. With those restrictions, only a specific group of people including authors, journalists and critics were given the chance to express themselves publicly. But the development of technology has facilitated each and every person to express their thoughts using digital media like blogs, social networks and video sharing sites such as YouTube. Ultimately each and every person has the privilege of being an active writer and a viewer with the transition of web 1.0 to web 2.0. In web 2.0 the end users of WWW actively participate in content generation – an area that was dominated by site administrators in web1.0 [1].

With the ability for general public to create web content, people started to create content in different forms such as blog articles, videos, tweets, Facebook status updates, comments, etc. Those content elements reflect various psychological aspects of a human being such as sentiments, opinions and attitudes. Thus, analyzing of the web content makes it possible to evaluate perceptions, which leads to endless opportunities in many ways.

Web content analysis can be used to evaluate the quality of a lecture in a university. We can observe a trend among university students of tweeting and sharing thoughts of their learning experience [2]. This may happen after a particular lecture or while the session is in progress. Those shared perceptions of students have an implicit value for quality assurance of the learning environment.

The conventional method of capturing students’ perception of their learning experience is surveying. The primary stage was using written and verbal surveys to collect feedback, but it has moved to online forms with the development of web. Those survey forms are composed with fixed Likert Scale [3] based questions in which students can convey their overall perception for each question. Usually such feedback is collected on a semester basis in a summative manner and this method is known to have several disadvantages. These disadvantages include the focus being on the actor instead of the script, thereby not measuring teaching effectiveness in terms of improved student learning. Further, such summative evaluations tend to be based on a singular measurement model across all courses in a university, thereby not catering to the diversity of the courses and learning outcomes [4]. It is very rare for a lecturer to conduct surveys per lecture.

Since the conventional feedback provides an overall view of the course, the lecturer or the evaluator of the feedback cannot identify which section/lesson/lecture of a particular course was difficult for the students. Thus providing a more effective quality improvement plan for a particular lecturer is not possible with this traditional method. In other words, the lecturer or the quality assurance cell is not able to identify the areas and aspects of a lecture series that needs to be improved. Design, conduct and analysis of teaching evaluation become very tedious and time consuming if feedback is collected using papers. It does not utilize the E-learning concepts well enough.

We have introduced a crowd sourced perception capturing and analysis platform, which can minimize the drawbacks in conventional approaches. Students are given the opportunity to share their current perception regarding a lecture, while it is in progress. They can use their mobile devices to share their perceptions. This can even be integrated into the university Learning Management System (LMS). The data published in social media channels can also be fed to the platform. The collected data are analyzed in different dimensions such as time and location. Those can be viewed in real-time and as post analytics thereby creating a convenient decision making environment. The methodology and the results of the research have been discussed in the latter parts of the paper.
II. Method

A. Design of platform and user interface

The system comprises three basic perception capturing devices: mobile client, web client and fixed terminals. All these components are connected with the backend servers. The major architectural concern addressed was the extendibility of the platform, because it should allow universities to build their own applications or Portlets to a LMS as per the requirement. High extendibility allows this platform to be used in several ways in an educational institute. For example, the fixed terminal clients can be installed in public places such as canteens and hostels to capture students’ perceptions regarding welfare facilities.

We have decoupled the backend and the frontend user applications via a Representational State Transfer Application Programmer Interface (REST API). The backend servers have a service endpoint which can transfer data in and out using the conventional HTTP protocol. The data is transferred as JavaScript Object Notation (JSON) objects which is parsed in clients and servers. The decoupling allows the application developers to develop their own application on top of the platform. API developers can continuously add new API methods required for the use cases of a particular university. The Service Oriented Architectural (SOA) style has been adapted to the platform, as discussed above, in order to ensure its extendibility. The high-level component diagram is given in Fig. 1.

User interfaces of end user applications play a significant role, because this is a generic platform used by diversified end users. Students share their perception while the lecture is in progress. Therefore it can be done without any interruption to his/her studies. Also the interface should be intuitive and easy to understand. Thus the user interfaces of the mobile and web clients have been designed in such a way that the student can convey his/her perception with a single tap/click. A sample screenshot is shown in Fig. 2.

B. Analytics

Real-Time Perception Analysis:

Real-time analysis gives live updates of the perception state of a particular lecture or location in the university.

A Server Sent Event (SSE) [5] is generated periodically (once per 3 seconds) and pushed to the client side through a pre-built channel. The clients who are subscribed with the channel will receive the update and visualize it as required. In the web application, a JavaScript chart library is used for visualization.

Separate database collections are maintained to keep track of the latest perception of the users registered with a lecture. A single perception record is maintained per user in those tables and that record is updated periodically as the user conveys his/her perception. Thus the resultant data set of this algorithm indicates the latest perception distribution according to the perception types.

Perception categorization for real-time analytics is done using the Map-Reduce programming model [6] because it can be scaled-up easily in a multiprocessor system or a sharded database. This maintains the performance of the algorithm even if a large number of students are registered with a lecture. The following is the pseudo code for the Map and Reduce functions.

map
emit,
this.perceptionValue,
count: 1
reduce(key,values):
reduced = {key:key,count:0};
forEach in values
do
reduced.count←reduced.count+val.count;
end
return reduced;

The key corresponds to the perception type and the value corresponds to the number of perceptions of that type.

After the data set is created, a message has to be composed to be sent to the client side as per HTML5 SSE specification. This message is composed of two data attributes; event and data. Event is used to identify the context of the message when messages have to be processed separately on client side.
A new line character separates each attribute and two consecutive new line characters indicate the end of the message. This message is generated periodically, once per 3 seconds, by default, and can be changed as per the requirements. The clients can receive the message and visualize/represent data, as they desire. Fig. 3 is a screenshot of the real-time visualizations of the web application.

**Real-Time perception pattern recognition:**

Due to the complexity of a high rate of event generation in massive amounts, continuous event capturing in real time, and the need for smooth processing and analyzing; real time pattern detection has become challenging. To overcome those challenges, the concept of Complex Event Processing (CEP) has been successfully adapted in the real time analysis component of the platform. Complex event processing is a concept used in real time processing of complex and massive amounts of event streams which are constantly generated from end user applications like mobile clients, web clients etc.

In the platform discussed in this paper, the complex event processing has been implemented with the use of WSO2 CEP [7]. The main responsibilities of the CEP system are; filtering the incoming event stream of perceptions according to the given conditions, detecting patterns related with perceptions, and triggering new events based on analytics. When CEP brokers receive an event, the rest of event processing and pattern identification is done through a logical execution unit called a ‘bucket’ within the CEP engine. With the use of constructs such as window, patterns, sequences, filters and joins available with SiddhiQL [8], very effective and sensitive real-time queries can be defined to identify patterns.

The real-time perception analytical component is customizable and capable of analyzing events in different contexts. Pattern detection is used to generate notifications as real-time student feedback to the lecturer, based on the analytics done by the CEP engine. If the real-time analytical component gets negative perception patterns, the lecturer can be notified to change the topic, change the teaching style or interact with the students to ascertain the issue, etc.

Following are two samples of Siddhi queries defined on the buckets of the Complex Events Processor.

From studentPerceptionStream
insert into perceptionStream
count(perceptionValue) as perceptionBurst,
perceptionValue,lecture
group by perceptionValue,lecture;
(1)
from perceptionStream#window.time(2 min)
[perceptionBurst>= <threshold>]
insert into perceptionBurstDetect
perceptionBurst, perceptionValue;
(2)

The above two consecutive queries are capable of filtering and detecting a perception burst occurring within two minutes. In the first query (1), the perception values and the count of each perception value are inserted into a separate event stream. The second query (2) triggers an event, if the count of a particular perception type exceeds the defined threshold within 2 minutes. For example, if the incoming ‘exhausted’ perceptions count within 2 minutes exceeds the defined threshold, the lecturer can be notified to give a break. In the second query, the time window slides by milliseconds and triggers new events when the defined condition is satisfied within the defined time length of the window.

**Post Analytics against Time:**

The requirement for this algorithm is to calculate variations of perception with time. There are two variable parameters involved; Time and Perception. The time parameter is sampled to fix time quanta and each sample is analyzed independently. If the value of “NoOfSamples” is too high, the accuracy of the results increases, but data visualization might be problematic if the client side is unable to handle a large data set. Thus its value should always be adapted as per client side capabilities and the length of the time span. The following is the pseudo code for calculating the sample size.

GetSampleSize:

Const NoOfSamples
Array ← Perceptions //Read Perceptions from db and put into an array
MinTime ← Array(0).TimeStamp
MaxTime ← Array(Array.length -1).TimeStamp
SampleSize ← (MaxTime-MinTime) / NoOfSamples

The following is the pseudo code for perception aggregation:

AggregatePerceptions:

Array ← Perceptions
MinTime ← Array(0).TimeStamp
Sample Size ← GetSampleSize
Result ← new object of arrays
Foreach perception in perceptions
   do
      Index ← (perception.TimeStamp- MinTime)/SampleSize)
if(result[perception] is notNull) 
    if(result[perception][index] is not null)) 
        increment result[perception][index]by 1 
    end if 
end if 
result[perception][index] ← 1 
result[perception] ← new Array 
result[perception][index] ← 1 
end 

There is a single iteration through perception array. The corresponding index value is calculated for each perception. That index represents the position of that particular perception in the time axis. Then the array value for the corresponding perception is incremented using the calculated index. Given below is an example of the resultant object.

```plaintext
{ "Perception1" : [null,null,Count;3,null, ..., Count;N],  "Perception2" : [Count;1,Count;2,Count;3, ..., Count;N],  "Perception n" : [null,Count;2,Count;3, ..., Count;N] }
```

\( n \) corresponds to the number of perception types in the defined perception schema. \( N \) corresponds to the number of samples in the time domain. The null elements represent that the perception count for a specific sample of a particular category is 0. This data set is then sent to the client as the response. Further processing and visualization are done at the client side.

### Mathematical model for perception aggregation

The resultant perception data can be represented using an \( n \times m \) matrix where \( n \) corresponds to the number of perception categories and \( m \) corresponds to the number of samples in the time domain. The mathematical model is highly favorable in several scenarios such as; algorithm generation for calculating total perceptions or average perceptions, and identifying patterns from post analytics by using convolution masks for that matrix.

\[
\text{AggregatedMatrix} = \begin{bmatrix}
    C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\
    C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm}
\end{bmatrix}
\]

\( C_{ij} \) is the total count of \( i^{th} \) type perception on \( j^{th} \) sample

\( C_{ij} \) can be calculated using (3).

\[
C_{ij} = \sum_{k=t}^{r} X_{ij}(k) \quad (3)
\]

Where,

\[
X_{ij}(k) = \begin{cases} 
1 & \text{if perception } k \text{ is of } i^{th} \text{ type and occur in } j^{th} \text{ sample} \\
0 & \text{otherwise}
\end{cases}
\]

And \( r = \text{total perception count} \)

The lower limit and the upper limit for \( i^{th} \) time sample can be calculated using (4) and (5).

\[
\text{lower limit} = \text{startTime} + (i - 1) \times \text{sampleSize} \quad (4)
\]
\[
\text{upper limit} = \text{startTime} + i \times \text{sampleSize} \quad (5)
\]

And \( C_{1k}, C_{2k}, C_{3l} \) and \( C_{nn} \) values lie between the above-mentioned time limits.

Several calculations can be done using such a perception-aggregated matrix. Its row summation is the total number of perceptions of particular type represented by that row. Column summation corresponds to the total perception in the time sample represented by that column.

### Spam Avoiding Method

One of the major concerns for a precise and direct perception-sharing platform is bogus/spamming of perceptions. To prevent someone from overflowing the system by continuously sharing perceptions, a threshold time gap between two consecutive perceptions has been implemented. Nevertheless, this is a major concern in the algorithm for perception aggregation against the time, because if there are different perceptions from the same user in a given sample it affects the final aggregated matrix. To avoid that, a function (6) is implemented in perception capturing modules as a constraint.

\[
T_u(i) - T_u(i - 1) \geq \text{SampleSize} \quad (6)
\]

Where, \( T_u(i) \) represents the time in which the user \( u \) conveys his/her \( i^{th} \) perception. The above constraint ensures that the same user cannot share his/her perception more than once in a given sample.

### Location Based Post Analytics

This platform can analyze perceptual data from different locations, which enables the platform to identify how perceptions are scattered through various geographical regions. This approach is very important to identify the possible improvements for university infrastructures such as hostels and canteen facilities. If a university has sub-institutions in different locations, the university administrators can use this to compare the quality of those institutions.

If this analytical information is added as an overlay to an application such as Google street view [9], people can use Google maps to see what places would have the potential to make them feel positive (or negative). Fig. 4 shows the perception variations in 3 different locations in the Sumanadasa Building at the University of Moratuwa.

The main challenge of implementing location-based analytics is the scale of the data set and the performance of the algorithm. Those analytics have to be done periodically to provide up-to-date information.
To fulfill these requirements, the aforementioned map reduce based implementation has been used. The technology used for this is Apache Hadoop [10] which is a popular map reduce implementation. Apache Hive [11] was used to write analytic queries. Apache Hive is a data warehouse software which facilitates querying and managing large datasets residing in distributed storage. This runs on top of Apache Hadoop.

To reduce the complexities of setting up and managing such a configuration we used an open source tool, WSO2 Business Activity Monitor (WSO2 BAM) [12]. The default use case of WSO2 BAM is to monitor key business indicators. But it is designed in such a way that it can be easily extended to do other monitoring and analysis activities.

WSO2 BAM provides an API with low latency for capturing events via various protocols. In our design, we only used the REST API for publishing events. BAM uses Apache Cassandra [13] as its data storage engine to store data. Once the data is stored in the Cassandra data store, it provides facilities to write queries similar to that of SQL on these large, schema-less data sets via Apache Hive. It also provides facilities to schedule these queries.

Because of BAM’s multitenant nature, we could give the capability of writing customized queries to the university administrators to analyze their own data. Our main analytics run with the highest privileges and those have access to all the data in the platform. But analytics are written in such a way that the results do not violate the privacy of individual users.

Our main API implemented using Node JavaScript, clones every event (lectures, workshops, etc.) that occurs in the system to the REST API of BAM. BAM stores those events on its own Apache Cassandra data store and runs all the scheduled queries periodically. The results that are generated using these data are stored in an external MySQL relational database.

These data are retrieved according to the application logic. Fig. 5 shows the overview of BAM integration of the platform.

III. RESULTS

Beta testing for the platform was conducted at the industry workshop series held in Department of Computer Science and Engineering (CSE), of University of Moratuwa [14].

At the 4 workshops that were conducted in the CSE Seminar room, the students used their own laptops and mobile devices and at the 2 workshops that were conducted at the Advanced Computing Lab, students used the desktop PCs in the lab and their own devices to share perceptions. All these workshops were on different technologies and industry practices such as SQL, .NET, QA practices, Agile methodologies etc.

Students participated in perception sharing while the workshops were in progress. Real-time analytics were displayed to the presenter and to the workshop organizers using separate laptops. Environment setup of all the workshops were very diversified, unbiased and can be mapped to a real operation environment. The number of participants for each workshop is in Table I.

The feedback obtained from the organizers of the workshops and participants were positive. They accepted the usability of the user interfaces and the comprehension of the analytics and visualizations.

The evaluation report (Fig. 6) generated from the platform helped them to identify weaknesses and improve the quality of the workshops gradually. With the lesson learnt from previous workshops, there were observable quality improvements in successive workshops.

<table>
<thead>
<tr>
<th>Workshop</th>
<th>Number of participants</th>
<th>Number of users registered with the event</th>
<th>Percentage of participation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workshop 1</td>
<td>89</td>
<td>73</td>
<td>82</td>
</tr>
<tr>
<td>Workshop 2</td>
<td>78</td>
<td>74</td>
<td>94.9</td>
</tr>
<tr>
<td>Workshop 3</td>
<td>65</td>
<td>60</td>
<td>92.3</td>
</tr>
<tr>
<td>Workshop 4</td>
<td>67</td>
<td>60</td>
<td>92.3</td>
</tr>
<tr>
<td>Workshop 5</td>
<td>60</td>
<td>38</td>
<td>63.3</td>
</tr>
<tr>
<td>Workshop 6</td>
<td>57</td>
<td>42</td>
<td>73.7</td>
</tr>
</tbody>
</table>

The evaluation report (Fig. 6) generated from the platform helped them to identify weaknesses and improve the quality of the workshops gradually. With the lesson learnt from previous workshops, there were observable quality improvements in successive workshops.
We obtained promising results for quality improvement of the workshops. Table II and Fig. 7 contains the analysis of positive and negative perceptions of each workshop.

Workshops 1, 2, 4, 5, 6 were conducted by the same group of IT industry professionals and workshop 3 was done by another group. A significant reduction of negative perception percentage can be observed across workshops, 1, 2, 4, 5, and 6. All post analytic data were sent to the people who conducted the workshops.

There is an increment for negative perceptions from workshop 2 to 3. We can deduce the reason as not having prior knowledge on the behavior of the audience to be the reason for this.

The perception analytic results obtained from the system discussed in this paper had a huge impact on quality improvement of the workshop series.

IV. CONCLUSION

The conventional feedback collection methodology used in universities is not effective and does not carry enough information. Therefore planning and execution of effective quality assurance has become difficult. To overcome these weaknesses, we have introduced a crowd sourced perception capturing and analysis platform. It considers the perceptions of a student crowd in order to come to a judgment of their learning experience and the infrastructure facilities of the whole university.

This helps to identify the specific parts of a particular lecture that needs to be improved or to compare various locations of a university. We have got promising results by using this platform for several university workshops.

The next steps for this research project is to enhance the platform to be used outside the university, because we believe that this also has the potential to be used in mercantile and entertainment sectors to analyze the trends in the society and measure the performance.

REFERENCES


| Table II: Figures of Positive and Negative Perceptions for Each Workshop |
| Workshops | Positive Perceptions | Negative Perceptions |
| Number | Percentage | Number | Percentage |
| 1 | 304 | 33.0 | 270 | 47.0 |
| 2 | 356 | 56.2 | 277 | 43.8 |
| 3 | 348 | 54.5 | 200 | 45.5 |
| 4 | 450 | 59.8 | 302 | 40.2 |
| 5 | 490 | 61.3 | 310 | 38.8 |
| 6 | 386 | 65.9 | 200 | 34.1 |

Figure 6. Post analytic results generated by the platform

Figure 7. Variations of positive and negative perception among workshops.

