Supporting Math Trails on Property Graphs

Sai Sumana P[+]
IIT Guwahati
Guwahati 781039
Assam, India
s.pagidipalli@iitg.ernet.in

Veena S Kambi, Sudharani R Nakati
Sarada Research Labs
56/2 'Anugraha' Seshadri Dr, Benson Town
Bangalore 560046, India
080-23545856
veena.kambi,sudha.nakati@saradaresearchlabs.org

Jagannathan Srinivasan
Oracle Corporation
One Oracle Drive
Nashua, NH 03063, USA
jagannathan.srinivasan@oracle.com

ABSTRACT

pg-MathTrails is an application developed to assess math skills of school students. It includes i) *topic hierarchy* to categorize the questions, ii) *question template repository* using which teachers can add different types of questions, iii) *question play module* to play questions, iv) *student evaluation module* to evaluate student performance, and v) *recommender* that recommends videos residing in a *video repository*. A key feature is that each interactive student session is treated as a *walking trail* in the *property graph* formed by *topic* and *question* nodes, and student evaluation involves operations on the property graph. The application is being used in the after-school program of *Sri Sarada Academy of Excellence, Bangalore*.

1. INTRODUCTION

pg-MathTrails is a student assessment application built for school children. It is inspired by the fitness trails equipped with exercise stations (see Figure 1 [1]) typically found in parks and college campuses. Using the *pg-MathTrails* application, a student undertakes a *math trail* aimed to evaluate or strengthen his *math fitness* with respect to a particular *topic* and as he proceeds he is presented with exercises varying in levels of difficulty. The entire trail along with his attempts at the exercises is recorded in a *property graph* [2], which can be subsequently used by him or his teacher to analyze his performance (hence the name *pg-MathTrails*). The application records the trail, the order in which exercises are attempted, and for each exercise it includes whether correct answer was obtained, number of wrong attempts, and the time taken. Like a fitness trail, our application allows user to quit in middle.

Of course, our application also has some variations when compared to fitness trails. Two students can simultaneously be on the same trail attempting the same exercise at any given point of time. In addition, unlike a fitness trail where exercise stations are fixed, we have a choice of rearranging the exercises within a single level of difficulty so that there is a variation in the order in which questions are presented to different students.

Furthermore, the application monitors the student performance continuously and elevates him to challenging exercises if he is performing well.

Figure 1. A fitness trail with 20-station exercises at Woods Hole, MA, USA; top-right showing station 8.

On the contrary, if he is having difficulty with a particular exercise, our application encourages the student by giving a suitable hint. It also has an option to preempt the student from continuing if we find that his performance is below par. He is then presented with videos on the topic that he can watch to further learn that topic after which he can re-attempt that trail. Also, his complete performance at various exercises is available for him to review, made available by the courtesy of the underlying property graph.

We chose property graph as the basis for this application, because of its versatility. Specifically, property graphs allow associating one or more *properties* as *key-value* pairs with both *nodes* and *edges*. Furthermore, it has the flexibility in that two nodes can have different set of properties. For example, in our case *topic* nodes and *exercise* nodes differ in their properties. In addition, the nodes of single type, for example *exercise* nodes, can differ in their properties. An *exercise* node, which is skipped, would not have *wrongAnswers*, and *timeTaken* properties specified.

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec, 2014 at Hyderabad, India.

[+]This work was done as part of a summer internship at Sarada Research Labs, Bangalore.

We also benefit from the ability to associate *key-values* pairs with *edges*. For math trails, we associate for edges connecting nodes, a key/value pair for key *autoSkip* that indicates if an optional question was skipped as student was performing well (See Section 2.3 for more details).

Challenging aspects in development of the application are:

- Support variation in the trails, including supporting auto skipping of questions for student performing well, and forcefully exiting students who are performing poorly.
- Support multiple students to simultaneously walk the same topic's trail, with each student's information independently maintained.
- Support comparing two trails of a particular topic belonging to the same student or different students.
- Support student to suspend and resume a trail.
- Support adding questions of various types under a topic hierarchy.
- Support evolution of a trail pertaining to a particular topic, for example, addition, update, and/or removal of exercises.
- Support linking exercises to instructional videos.
- Even though developed for Math, the application should be usable for any other subject.
- The application should be driven by a light-weight custom property graph implementation that does not require interfacing with graph databases (such as Neo4j[3], Dex [4]) and yet allows interactive response time.

The complete application is developed as a database-centric web application using Oracle Application Express 4.0.2 (APEX) [5], which is a rapid application development tool, and uses Oracle Database 11gR2 Express Edition [6] as the backend database.

The application is being used in the after school program of Sri Sarada Academy of Excellence, Bangalore for children from grade 1 to grade 10. Also, to illustrate its working we present math trail data based upon the use of this application by children in the after school program.

The key contributions of this work are:

- The use of property graph to represent the question repository and student's math trails.
- A simple and customizable criterion for skipping questions as well as for pre-empting the user if his performance is below par.
- Providing detailed analysis of student's performance on a particular topic's math trail, including comparison of two trails on the same topic, and computing overall student's participation.
- A usability evaluation in an after school program.

The rest of the paper is organized as follows. Section 2 presents the key concepts. Section 3 covers the design and implementation. Section 4 gives a tour of the application. Section 5 reports the usability study. Section 6 covers the related work and Section 7 concludes the paper and outlines the future work.

2. KEY CONCEPTS

This section discusses the key concepts.

2.1 Terminology

A *Property Graph* is key/*value-based, directed, multi-relational, labelled* graph [2]. Specifically, it consists of a set of vertices, and a set of edges like directed labelled graph with the following characteristics:

- Each vertex has a unique identifier, a set of outgoing edges, a set of incoming edges, and a collection of properties defined by a map from key to value.
- Each edge has a unique identifier, a head and tail vertex, a label that denotes the type of relationship between its two vertices, and a collection of properties defined by a map from key to value.

The *multi-relational* aspect refers to the ability to associate multiple edges for same pair of vertices. This along with the ability to associate a set of *key/value properties* with both vertices and edges makes the property graph model suitable for our assessment application.

2.2 Basic Model for Question Repository

The questions are categorized under a topic hierarchy. The questions and the topic nodes together form the default graph. Each set of questions under a topic is referred to as the *topic default trail* and it consists of multiple groups of questions varying in their level of difficulty. Each group itself consists of a *mandatory set of questions* followed by zero or more *optional questions* that gets skipped automatically if the performance of the student is satisfactory.

Figure 2 shows a default topic trail for *Math→Arithmetic→Addition* topic, consisting of 9 questions organized under three levels. The optional questions shown in sand color appears towards the end of each level. The questions are connected to the leaf topic via *hasQuestion* edge. The leaf topic also connects to the first question by the *nextQuestion* edge. In addition, a question is connected to next question through the *nextQuestion* edge. The key/values pairs showed in bold (corresponding to properties *nm*, *lvl*, *opt*, *et*) are the key/values for the default topic trail.

2.3 Representing a Student Trail

The actual student assessment of a topic forms a *student trail*, which also is stored in the property graph. The trail is formed by questions visited by the student during the session. The questions are connected to the topic and to the next question node by the label *nextQuestion*. For each question node, the key/value properties include the ones for default topic trail as well as values for additional properties (*wa, ca, tt, tid*). The *tid* (refers to trail id) is not shown in Figure 2 to save space. A unique *tid* is generated for each student trail and associated with each of its question nodes. The student trail shown in Figure 2 in red indicates that the student only visited questions *q1*, *q2*, *q4*, *q5*, *q6*, and *q7*. The trail illustrates the concept of *auto_skip* that allowed him to skip optional questions, *q3*, *q8*, and *q9* as he was performing very well.

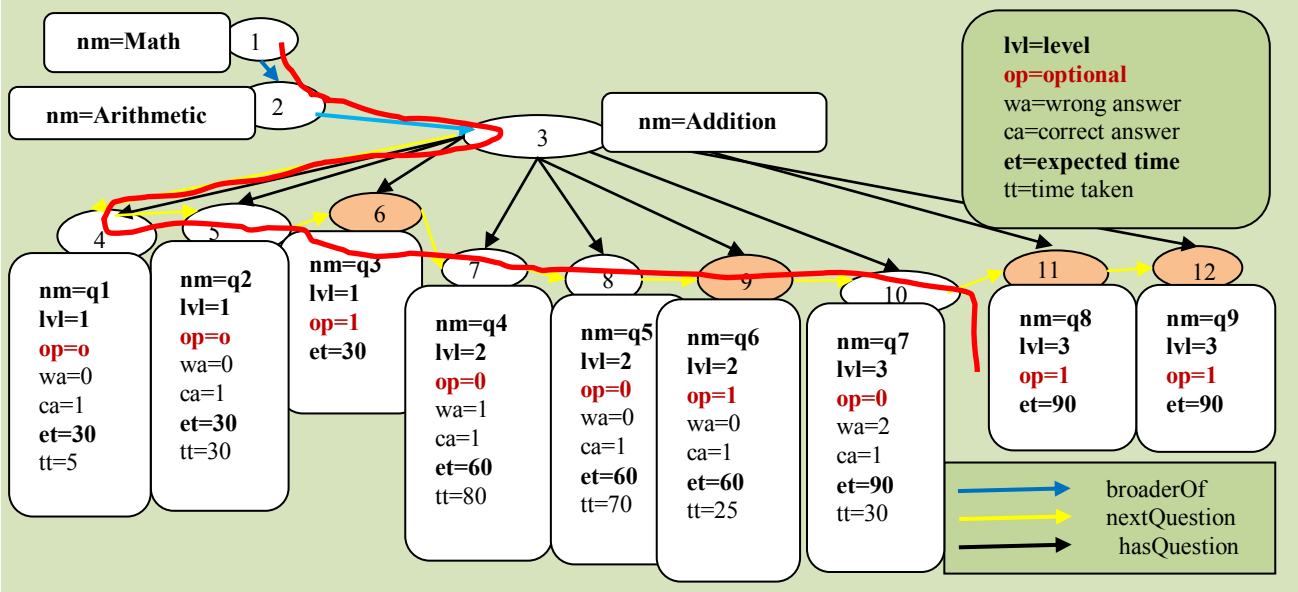


Figure 2. A Math topic trail with 9-station exercises; the actual trail undertaken by student shown in red.

The *auto_skip* criterion is defined as ability for student to skip optional questions if he has so far answered all questions visited correctly within the cumulative expected time.

Let a student topic trail be made of a set of nodes $Q=\{Q_1, Q_2, ...\}$ and each pair of consecutive *nodes* (Q_i, Q_{i+1}) are connected by *nextQuestion* edge. Each question Q_i itself has a set of properties (attributes), which can be referenced using '.' operator. For example, $Q_1.ca$ represents the value of *ca* attribute for question node Q_1. The *auto_skip* criterion can be defined by the following formula:

Skip to next level

WHEN $\forall i\, Q_i.ca=1$ for $i=1$ to cq and

$$\sum_{i=1}^{cq} Q_i.tt <= \sum_{i=1}^{cq} Q_i.et \text{ and } Q_{cq+1}.op=1$$

where *cq* is the current question.

For example of trail Figure 2, the student was able to skip question *q3*, because it was optional, and prior to that he had answered all questions correctly and the time taken (5s+30s=35s) was less than cumulative expected time (30s+30s=60s).

Note that for the edge with label *nextQuestion*, we associate the key *autoSkip (as)* to indicate optional questions were skipped. For example of trail in Figure 2, the edge *q2-nextQuestion*→*q4*, the property *as=1* is associated. In addition, student gets credit for auto skipped questions.

In addition to *auto_skip* feature, we also support *forced_exit* feature. This feature is used to terminate the student session if he is performing poorly. Specifically, if the total time taken for questions so far exceeds twice the cumulative expected time, we terminate the student trail.

This *forced_exit* criterion can be defined by the following formula:

Stop the lesson

WHEN

$$\sum_{i=1}^{cq} Q_i.tt >= 2 \text{ x } \sum_{i=1}^{cq} Q_i.et$$

where *cq* is the current question.

Also, we support *self_exit* feature, where the student can himself terminate the trail. Note that the *auto_skip* and *forced_exit* criteria can be further refined if needed. The key point is that our choice of storing the trail as an annotated graph, allows us to enforce these behavior easily when student is taking the trail.

2.4 Student Performance on a Math Trail

To keep things simple, raw score for mandatory and optional questions is reported separately. The raw score for mandatory questions can be specified as

$$Score = 100 \text{ x } \frac{\sum_{i=1}^{nq} (Q_i.ca \text{ x } Q_i.lvl)}{\sum_{i=1}^{nq} Q_i.lvl} \text{ and } \forall i\, Q_i.op=0$$

where *nq* is number of questions in the trail. For the trail shown in Figure 2,

$$Score = 100 \text{ x } \frac{((1 \text{ x } 1+1 \text{ x } 1)+(1 \text{ x } 2+1 \text{ x } 2)+(1 \text{ x } 3))}{(1+1+2+2+3)}$$

$$=100\%$$

Similar formula is used to report score for optional questions based upon questions attempted or credited because of *auto_skip* feature. Also, total points contributed by auto skipped questions will be maintained separately. At present the wrong attempts are not used in computing raw score. Our goal is to encourage

students to try and get answers right. The wrong attempts, and time take for each question, are provided separately.

2.5 Comparing Math Trails

We support comparing performances on same trail. The comparison becomes interesting because of occurrences of *auto_skip*, *forced_exit*, or *self_exit* due to which the two performances may differ in the questions visited.

Raw scores for each trail is compared using formula presented in Section 2.4. For the actual trail comparison, we show wrong attempts, and time taken for *all* the questions (the union of questions visited by the two trails). Our approach allows for comparing two trails even when the questions were visited in different order. However, we assume the order of questions can only be changed within questions of same level.

2.6 Computing Student Participation

Since student performance for each trail is stored as a separate graph, the overall student participation is readily available. The student or teacher can examine his performance by topic. Specifically, the *topic tree annotated with trails* undertaken by student capture the student participation. For a topic trail, a single trail performance can be examined, or two trails can be compared to see his progress over time.

3. DESIGN

This section discusses the design and implementation of applications.

3.1 Overview of Application

The application is made up of three sub-applications, which operate on a single consolidated backend database (Figure 3):

- *Video Repository*: This application manages the instructional videos, which are downloaded from Internet (for example, Khan Academy [7]) organized under a *topic hierarchy*. It also links to assessment module that can be used as a follow-up after watching a video on a particular topic.
- *Assessment Module*: This application handles all aspects of assessment including maintaining a topic hierarchy, adding different types of questions based on a question template repository, playing questions, and recording and evaluating student's performance (math trails). In addition, it also includes a recommender to recommend relevant videos.
- *Supplementary School Program Info*: This application handles the supplementary school program information, including student enrolment, and class composition in terms of videos screened and the assessments undertaken by students.

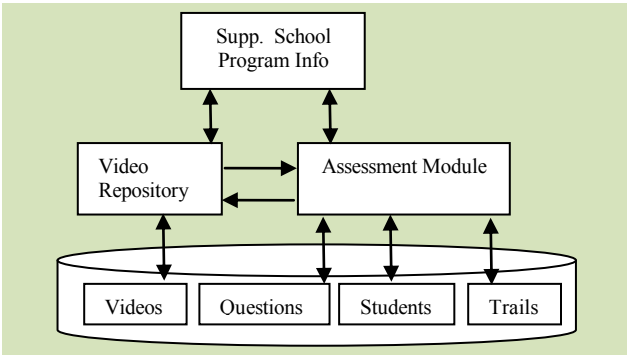


Figure 3. An overview of applications.

3.2 Database Schema

The key tables are shown in Figure 4. A student can have many trails and each trail itself is represented as graph of nodes and edges. A node in the trail refers to a topic or a question. Trail is for a particular leaf topic. Each topic has many questions, and videos. In present application, we currently have one-to-one relation from `M_QDB_TOPICS` to `T_QDB_TRAILS`. However, in future, we plan to extend this to one-to-many relationship that will allow a single leaf topic to have one or more trails. Thus, the student can select one of the trails available for the particular topic.

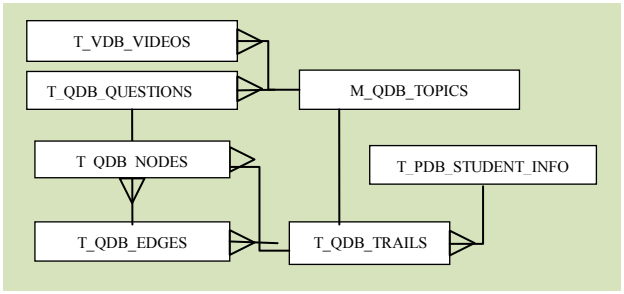


Figure 4. Database Schema (Key tables).

We have used the approach of creating custom property graph tables. That is, instead of generic nodes and edge tables, we use custom tables (`T_QDB_NODES` and `T_QDB_EDGES`), where the node/edge properties are represented as additional columns in node/edge tables. Specifically, `T_QDB_NODES` includes columns for representing *nm*, *lvl*, *opt*, *et, wa, ca, tt,* and *tid* properties. Similarly, `T_QDB_EDGES` includes a column for representing *autoSkip* property. This approach leads to efficient implementation.

3.3 Topic Hierarchy

Topics data is collected from Khan Academy [7] and are ordered under main topics Arithmetic, Geometry, Algebra (Figure 5). There are 32, 36, and 62 leaf subtopics respectively under these main topics. These are arranged using APEX Tree object in the order of their level. Users can take assessment in those topics which are leaf nodes.

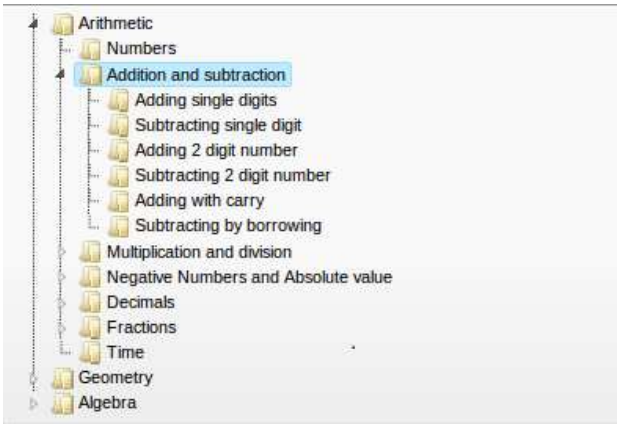


Figure 5. A portion of topic hierarchy

3.4 Topic Question Template and Question Repository

Specifically there are four types of question templates:

1. *Single Answer*: Solution to the question is a number or a fraction.
2. *Multiple Choice single correct answer*: A problem statement and four options will be given to the user out of which only one option is correct.
3. *Multiple Choice multiple correct answers*: A problem statement and four options will be given to the user out of which there may be two or more correct options
4. *True or False:* A statement would be given and the user has to identify whether it is true or false.

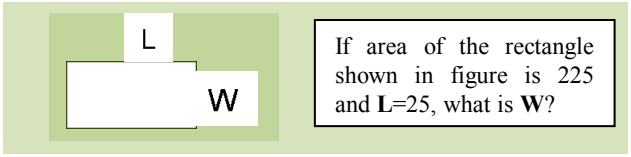


Figure 6. Supporting questions with figure.

3.5 Student Session and Trails

Each student signs-up prior to starting the trail, browses the topic hierarchy and makes a selection, which determines the start of the trail.

The trail itself is represented as a graph involving topic and question nodes. The question nodes are organized in increasing level of difficulty. Also, for each level, optional questions are maintained towards the end, which may be automatically skipped, if the student is performing well. Figure 2 represents one sample trail.

3.6 Playing Questions and Recording Trails

The algorithm for playing questions and recording trail is presented below. In the algorithm, step 1 initializes Cumulative Correct Answer (*cca*), Cumulative Wrong Answer (*cwa*), Cumulative Time Taken (*ctt*) and Cumulative Expected Time (*cet*) to `0`.

Algorithm : Play Questions in Trail

Input: *Leaf* Topic -- Chosen Topic Trail

1. Initialize a New Trail and cumulative values
   cca:=0; cwa:=0; ctt:=0; cet:=0;
2. Insert nodes of type `'TOPICS'` from *Root* to *Leaf* Topic;
3. Insert edges for the nodes in topic path;

-- Start assessment

4. Insert node of type `'QUESTION'` Q_1; *cq:=1;*
5. Update meta-data (*nm, op, et*) of question Q_{cq};
6. Insert edge from *Leaf* Topic node to question node;

-- Attempting question

7. WHILE *cq>0* LOOP – more questions –
   autoSkip:=False;

7.1 IF (*correct_answer=True*) THEN
   Q_{cq}.*ca:=1,* Q_{cq}.*wa:=0,* Q_{cq}.*tt:=time taken*;
   cca:=cca+1;ctt:=ctt+Q_{cq}.*tt;cet:=cet+*Q_{cq}.*et;*
   IF (cwa=0 AND ctt < cet AND Q_{cq+1}.*op*=1) THEN
      autoSkip:=True;
      cq=nq -- nq is firstQ in next level or 0 (if no more)
   ELSE – go to nextQ in sequence or 0 (if no more)
      cq:=cq+1;
   END IF;
   nextQ:=True;
  ELSE -- wrong answer
   Q_{cq}.*wa:=*Q_{cq}.*wa+1; cwa:=cwa+1; nextQ:=False;*
   IF (skip_question_by_user) THEN
      Q_{cq}.*tt:=time taken*;
      ctt:=ctt+Q_{cq}.*tt;cet:=cet+*Q_{cq}.*et;*
      – go to nextQ in sequence or 0 (if no more)
      cq:=cq+1; nextQ:=True;
   END IF;
  END IF;
  – too slow (*forced_exit*) or done assessment

7.2 IF (*ctt > 2* x *cet*) OR *done_assessment=True* THEN
   EXIT;
  END IF;

7.3 IF (*cq>0*) and *nextQ=True* THEN
  – next question exists
   Insert node of type `'QUESTION'` Q_{cq};
   Update properties (*nm, op, et*) of question Q_{cq};
   Insert edge from previous question node to Q_{cq};
   IF (*autoSkip=True*) THEN
    Set current edge property *as* to *1*;
   END IF;
  END IF;
 END LOOP;

8. Set Record trail end time;

In step 2, student trail gets initiated with root topic to leaf topic path, and in step 3, corresponding edges are created. As assessment starts, node type and current question (*cq*) is set to `'QUESTION'` and `1` respectively and Q_{cq} node properties (*nm, op, et*) are updated to respective values from the question node of default trail (steps 4-5). Step 6 connects the leaf topic node to the question node.

Steps 7 iterates over each question, which is presented to the student in the trail. In step 7.1, the question is attempted and based on student's performance (correct or wrong answer) the

relevant question node keys are updated, along with the cumulative measures (*ctt, cet, cwa, cca*). For correct answer the *auto_skip* condition is also evaluated and if met the next question is set to the first question in the next level.

In step 7.2, the *forced_exit* condition is met or it is the case of *self_exit*, the trail is terminated. In step 7.3, as student moves to next question, a new question node is created with default values for properties, and an edge is created to connect to prior question node. If *auto_skip* feature is exercised then the edge property *as* is set to 1. Step 8 records the end of the trail.

3.7 A Student Trail Summary and Details

Presenting student performance translates to traversing the recorded property graph for the trail and printing its properties. Students can review their performance by selecting the topic name in Topic list (Figure 7).

Trail Summary. For the selected trail, it shows student id, student name, topic name, trail id (*Trail)*, trail date (*T Date*), total time taken (*T Time*) by student, total expected time of that trail ((*E Time*), raw score of mandatory *(Raw score M Q)* and optional questions (*Raw score O Q*), and percentage.

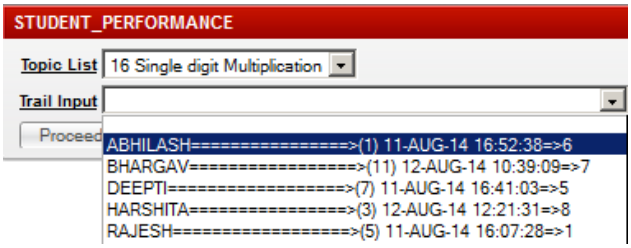


Figure 7. Trail selection.

The summary (Figure 8) is shown using a dynamically created HTML region followed by a SQL report.

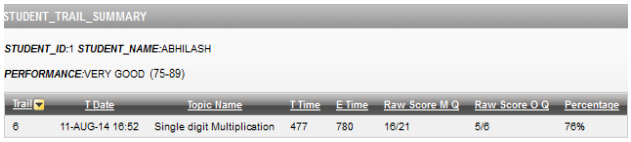
```
STUDENT_TRAIL_SUMMARY:
htp.p ('<p>'||'<em><b>STUDENT_ID:</b></em>'
            || V_STUDENT_ID||' '
            || '<em><b>STUDENT_NAME:</b></em>'
            || V_STUDENT_NAME||'</p>');
PRC:=nvl(PRC, (N_R1/N_TQ)*100);
IF (PRC<25) THEN
htp.p('<p>'||'<em><b>TOPIC_NAME:</b></em>'
           ||V_TOPIC_NAME||' '
           ||'<em><b>PERFORMANCE:</b></em>'
           ||'BAD'||'</p>');

END IF;
...
```

The `STUDENT_TRAIL_SUMMARY` contains the SQL statement which generates the report shown in Figure 8:

```
SQL_STMT:= 'SELECT '||P_TRAIL_ID||' TRAIL, '''
 || S_DATE||''' T_DATE, '
 || V_CUMULATIVE_TT ||' T_TIME, '
 || V_CUMULATIVE_ET|| ' E_TIME, '''
 || N_R1||'/'||N_TQ||''' RAW_SCORE_M_Q, '''
 || N_O||'/'||N_TQ_O||''' RAW_SCORE_O_Q, '''
 || ROUND(PRC)||'%'||'''PERCENTAGE '
 ||' FROM DUAL ';
```

This function will be called in report as shown below:

```
 DECLARE
  SQL_STMT VARCHAR2(1000);
  BEGIN
  SQL_STMT:=STUDENT_TRAIL_SUMMARY(:P8_TRAIL);
  RETURN(SQL_STMT);
END;
```


STUDENT_TRAIL_SUMMARY

STUDENT_ID:1 STUDENT_NAME:ABHILASH

PERFORMANCE:VERY GOOD (75-89)

Trail	T Date	Topic Name	T Time	E Time	Raw Score M Q	Raw Score O Q	Percentage
6	11-AUG-14 16:52	Single digit Multiplication	477	780	16/21	5/6	76%

Figure 8. Trail summary.

Trail Details. This report (Figure 9) shows the details of the selected trail by querying `T_QDB_TRAILS` and `T_QDB_NODES`. It shows level of question (*LVL*), question id (*Q ID*), optional question (*OPT Q*), correct answer count (*CORR ANS*), wrong answer count (*WRNG ANS*), expected time for question (*EXP TM*), and time taken by student (*TM TKN*).

TRAIL_DETAILS

LVL	Q_ID	OPT_Q	CORR_ANS	WRNG_ANS	EXP_TM	TM_TKN
1	Q01	-	1	0	30	22
1	Q02	-	1	0	30	22
1	Q03	-	1	0	30	20
1	Q04	-	1	0	30	29
2	Q06	-	1	0	60	42
2	Q08	-	0	1	60	50
2	Q11	-	1	0	60	38
2	Q13	-	1	0	60	35
2	Q15	1	1	0	60	39
3	Q17	-	1	0	90	37
3	Q19	-	0	1	90	60
3	Q22	-	1	0	90	41
3	Q23	1	1	0	90	42

Figure 9. Trail details.

Correct and Wrong Answer Comparison Chart. An APEX Flash Chart region is created to compare correct and wrong answer counts, shown in blue and maroon respectively (Figure 10). For the trail, student got 11 correct and 2 wrong. The chart data was generated using a SQL query shown below. For computing cumulative counts, Oracle's built-in Analytic function SUM is used.

```
SELECT null link, rownum label,
  SUM(nd_q_ca) OVER(ORDER BY nd_q_id) value1
FROM(
 SELECT loj.nd_q_id, t.nd_q_ca FROM(
 (SELECT  nd_q_id
  FROM  "VAP"."T_QDB_NODES"
  WHERE nd_name='QUESTION' AND
        nd_trail_id=:P8_TRAIL) loj
 LEFT OUTER JOIN
 (SELECT  nd_q_id, nd_q_ca
  FROM  "VAP"."T_QDB_NODES"
  WHERE nd_name='QUESTION' AND
        nd_trail_id=:P8_TRAIL) t
  ON loj.nd_q_id=t.nd_q_id)
  UNION ALL
  SELECT  0,0 FROM dual
ORDER BY nd_q_id)
```

Post-Trail Report. This report is shown after completion of a topic trail. Student can end the trail normally, or leave in the middle, or have a *forced_exit*. This report is similar to the one shown in Figure 8. We have used the same function `STUDENT_TRAIL_SUMMARY` to generate this report.

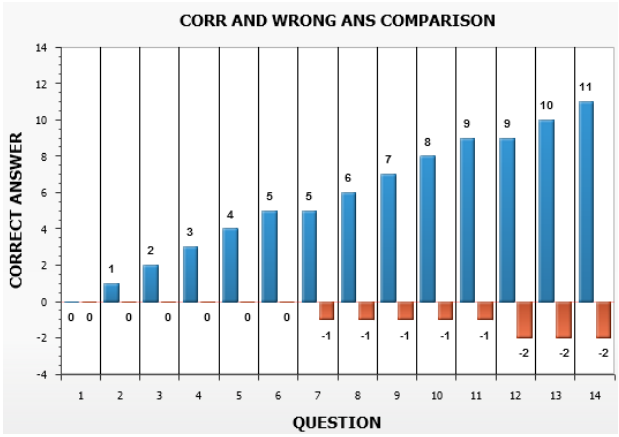


Figure 10. Correct and wrong answer comparison.

3.8 Comparing Trails

Compare trail translates to comparing the respective property graphs for the trails. Two trails on the same topic belonging either to a single student taken at different times or to two students can be compared. Input for the trails comparison is *trail_id*s of two different trails and output will show up as summary (Figure 11) and detail region (Figure 12).

Student Name	T Date	Topic Name	T Time	E Time	Raw Score M Q	Raw Score O Q
NIDA	05-AUG-14 11:58	Adding single digits	151	420	18/21	6/6
VIVEK	06-AUG-14 16:54	Adding single digits	541	420	14/21	4/6

Figure 11. Trail comparison (Summary).

The summary report compares the two trail details. Cumulative expected time may not be same as one of the students can exit the trail in middle or he is able to skip questions due to *auto_skip* feature.

T1_trail_id	T2_trail_id	T1_Q_ID	T2_Q_ID	EXP_TIME	T1_TTL_T	T2_TTL_T
5	28	Q01	Q01	20	16	21
5	28	Q02	Q02	20	15	16
5	28	Q03	Q03	40	12	62
5	28	Q04	Q04	40	17	48
5	28	Q21	Q21	60	10	84
5	28	Q32	Q32	20	21	40
5	28	Q33	Q33	60	13	60
5	28	Q34	Q34	60	13	65
5	28	Q35	Q35	40	22	55
5	28	Q36	Q36	60	12	90

Figure 12. Trail comparison (Detail).

The report region is formed by dynamically generating SQL statement using function mentioned below. This function is similar to the function `STUDENT_TRAIL_SUMMARY` described in Section 3.7. Instead of creating single row of summary it is creating 2 rows of summary report taking 2 *trail_id*s as inputs.

```
COMPARE_TRAIL_SUMMARY (p_trail_id NUMBER,
                       q_trail_id NUMBER)
RETURN VARCHAR2
```

The detailed trails comparison Report (Figure 12) presents the individual nodes of the trails, representing number of questions attempted, whether question is answered correct or wrong, and the time taken. This helps us understand the performance of the student in comparison with another student or with a prior trail taken by him.

This report is generated by using left outer join to connect node values for the two trails, union operator to have total questions attempted in both the trails.

The graph (Figure 13) presents cumulative expected time (shown in Green) for the two trails along with the individual cumulative time taken to complete trail by each student for his attempted questions. In this graph, Nida's trail is shown in Blue and Vivek's trail is shown in Red.

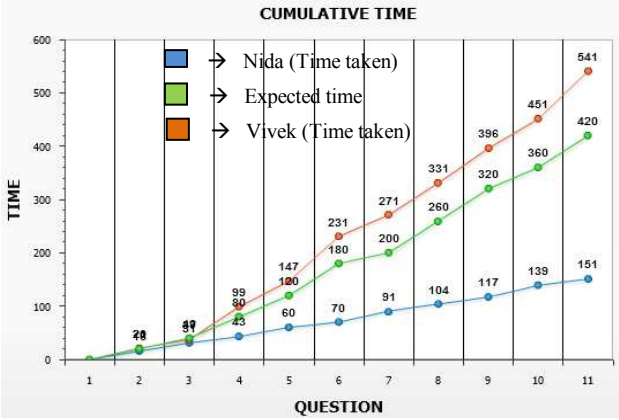


Figure 13. Cumulative time graph.

The cumulative expected time is computed by SQL query shown below.

```
SELECT null link, ROWNUM label,
    sum(ND_Q_ET) over (ORDER BY ND_Q_ID) value1
   FROM
  ((
   (SELECT ND_Q_ET , ND_Q_ID
    FROM "VAP"."T_QDB_NODES"
   WHERE ND_NAME='QUESTION'AND
         ND_TRAIL_ID=:P5_TRAIL_INPUT_1
   UNION
   SELECT ND_Q_ET , ND_Q_ID
   FROM "VAP"."T_QDB_NODES"
   WHERE ND_NAME ='QUESTION' AND
         ND_TRAIL_ID =:P5_TRAIL_INPUT_2)
   UNION ALL
         SELECT  0,0 FROM dual)
   ORDER BY ND_Q_ID)
```

The cumulative expected time is formed by adding expected time of the each node using Oracle's analytic function `SUM`, which are ordered by `nd_q_id`. Similarly, we get trails lines by `'LEFT OUTER JOIN'` join of cumulative expected time query with cumulative total time query of individual trails, which is combined with `'SELECT 0,0 FROM dual'` to start the trail from zero.

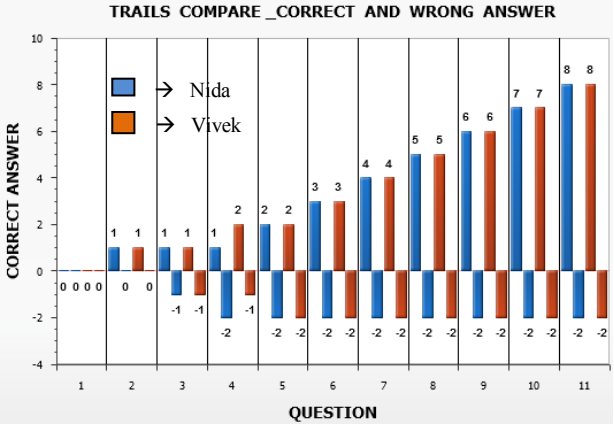


Figure 14. Cumulative correct and wrong answer counts.

Similarly, we have a bar graph representing the difference in two trails with respect to number of correct and wrong attempts (Figure 14).

3.9 Recommending Videos

Video recommender in the *Assessment Module* acts like a helper for the student taking assessment. On clicking `'Recommended Videos'` button, a report of videos relevant to the current topic is shown (Figure15).

Figure 15. Recommended videos for a topic.

This report is produced using the function:

```
recommend_video(f_topic VARCHAR2)
RETURN VARCHAR2
IS
sql_stmt VARCHAR2(512);
WORDS_ARRAY APEX_APPLICATION_GLOBAL.VC_ARR2;
MYWORDS VARCHAR2(256);
BEGIN
sql_stmt:='SELECT Store_id S_id,
       Subject Sub,
       Main_topic M_topic,
       Item_type I_type,
       Grade,
       Sub_topic S_topic,
       Item,
       Link RELATIVE_PATH
       FROM T_VDB_VIDEO_STORE WHERE ';
WORDS_ARRAY := APEX_UTIL.STRING_TO_TABLE(f_topic,'
');
FOR z IN 1..WORDS_ARRAY.count LOOP
  MYWORDS:='';
  FOR x IN 1..z LOOP
    MYWORDS:= MYWORDS||WORDS_ARRAY(x)||' ';
END LOOP;
IF z< WORDS_ARRAY.count THEN
  sql_stmt:=sql_stmt||
'UTL_MATCH.JARO_WINKLER_SIMILARITY(Upper(main_topi
c), upper('''||MYWORDS||'''))> 75 OR  ';
ELSE
  sql_stmt:=sql_stmt||
'UTL_MATCH.JARO_WINKLER_SIMILARITY(Upper(main_topi
c), upper('''||MYWORDS||'''))> 75';
END IF;
END LOOP;
RETURN(sql_stmt);
END;
```

where `f_topic` is the particular topic name.

In the report region of recommended videos the 'Link' column is not shown which represents the HTML hyper link for the individual ITEM of the respective topic. The link is formed by concatenating 'Relative path' to the Application Item `'F106_VROOT'` from `Video Repository` which is the root path of the video store.

When the button `RECOMMENDED VIDEOS` is clicked a confirm message `'Do you want to end the active trail? '` is presented. On confirmation, the application will terminate the trail and present recommended videos. The topic name of the assessment is compared to the similar topic present in the *Video Repository*, which is similar to searching for a relevant video for a topic. Since topics may not completely be similar, we consider the topic as a phrase (`'WORDS_ARRAY'`) and expand it into multiple phrases (`'MYWORDS'`) formed by variation of keywords.

These generated phrases `'MYWORDS'` are used in approximate string comparison with `'Main topic'` of `Video Repository` table. The approximate string comparison via *Jero-Winkler* metrics helps reduce the *false negatives*.

4. A TOUR OF pg-MathTrails

This section describes the features and functioning of *pg-MathTrails* assessment module.

4.1 Take Assessment

This module allows user to login, select a topic, and take an assessment (Figure 16-b and c). On entering the answer, the answer is checked, and the score is updated, and user can move to the next question.

On the next question, if the user makes a mistake the system displays 'wrong answer' as well as an option to click the 'hint' (Figure 16-d). The scoreboard is updated to include the wrong attempt.

4.2 Student Performance and Compare Trails

The student performance and compare trails have already been covered in Section 3, where individual student performance can be examined as well as two performances of same or different students on a topic trail can be compared.

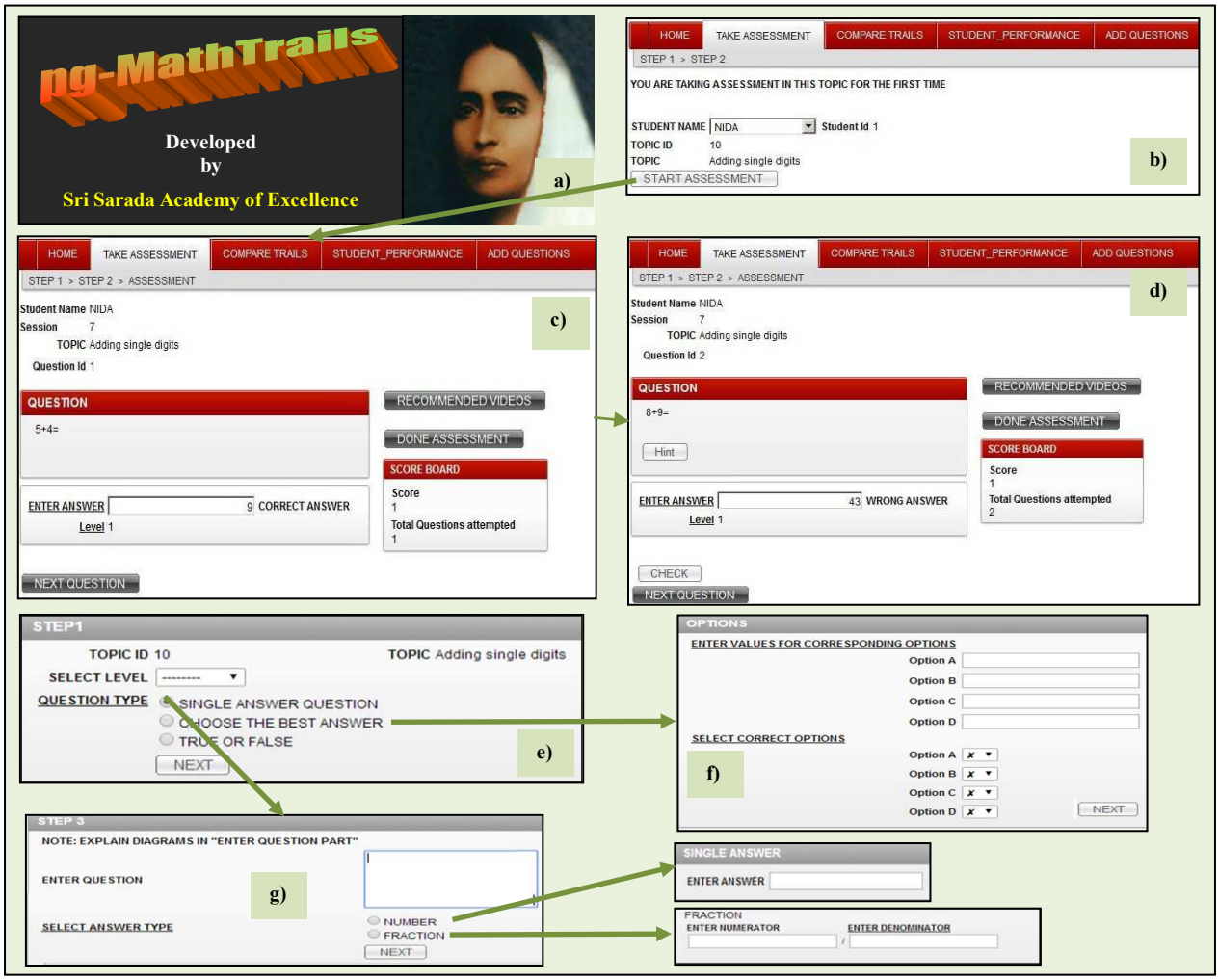


Figure 16. *pg-MathTrails* a) Home Page b) Start Assessment, c,d) Play Question, e to g) Add Questions.

4.3 Add Questions

This functionality is meant for teachers or instructors to add questions. (Figure 16-e) shows adding a question with single answer and multiple answer choices. For single answer (Figure 16-g), we provide the option of entering a single number or a fraction specified through two number fields.

For multiple choice questions (Figure 16-f), we allow user to specify the choices as answer. Note for the multiple choice question, we support both single choice and multiple choices as correct answer. The true or false type question is similar to a multiple choice question, except that user has to only specify either the true or false.

5. EXPERIENCE

This section describes our experience of using this application.

5.1 Usability Study

Objective. The objective of the Usability Study was twofold: i) To report the students experience of using the application, and ii) To study the concurrent use of the application by multiple clients.

Setup. Three computers were setup with a wireless LAN, one as the server as well as a client, and two other laptops as clients. We created two topic trails: i) for '`multiplication by single digit`', and ii) for '`adding negative numbers`'. Both trails had questions categorized into levels 1, 2, 3, and each trail had a total of 12 questions, of which 9 were mandatory and 3 were optional. The trails were taken by children in the after school program of Sri Sarada Academy of Excellence, Bangalore (SSAE). The first and second topic trail online test was administered to 3rd to 6th and 7th to 9th grade students respectively. Individual students from each group took the assessment.

Observations. User interface looked attractive to students and the performance was satisfactory in the wireless LAN setup. Students were very enthusiastic as they never had taken online test like this before. In the first session, students were struggling with mouse and keyboard as they came from underprivileged families having limited exposure to computers.

The concurrent use of application worked correctly (no interference). The response time was reasonably well, meeting the interactive response time limit of 2000 ms.

Challenges faced by students.

- Cursor was not set on the answer field.
- The answers entered by prior students were appearing as the browser form history was getting cached.
- Hint button was reappearing in the next question after correcting the first question. This was a bug.
- 7th and 8th students were struggling in using `('-', '+')` signs as they were taking test on `'adding negative numbers'`.

Overall performance. In the first session (1st August 2014, time: 3.30 to 4.45pm), students from 3rd and 4th grade performed very well as they had taken video based lessons on the `'multiplication'` topic in prior classes. These students attempted all the questions and took less time for questions in the beginning of the trail, and they were rewarded with *auto_skip*, which allowed them to skip optional questions. Students from 5th and 6th grade were little uncomfortable and slow while taking the same trail. This could be attributed to the fact that they were rusty, that is, they had covered this topic a while back. So as those students took more time to attempt initial questions, they were forced to end the trail (via the *forced_exit* feature).

Solution to problems faced.

- We made sure the students became familiar with the system and comfortable with mouse and keyboard.
- We fixed the application to have the cursor focus on the answer field automatically after each question.
- We configured the browser to not to cache form history.
- We fixed the bug, to clear the hint when next question button is pressed.

In the second session (7th August 2014, time: 3.30 to 5.00 pm), the students were more confident and comfortable with user interface due to changes made as well as they were more familiar. Similar to first session, two new trails were created for the same two topics. Students performed better when compared to the prior session and as one student got all the mandatory questions correct with no wrong attempts plus was able to skip optional questions.

Challenges faced. We noticed that using *Backspace* button on the keyboard multiple times was moving the user to the previous page and was repeatedly recording multiple similar nodes.

We resolved this problem by configuring the browser to not map *Backspace* key event to browser *Back* event.

The student trail performance was recorded and the application automatically generated `'STATISTICS OF TRAILS'` reports. For inputs, the `TOPIC`, `START TIME`, and `END TIME` of the session, we got the trails statistics (shown in Figure 17 top) representing total trails have been taken by all students, and the count of minimum, maximum, and average correct attempts, wrong attempts, and total time taken.

In addition, the application automatically generates the trail statistics by Question (Figure 17 bottom), with the drill-down ability. By clicking any question listed under `ND_Q_ID`, the detailed summary for the selected question is shown (Question 9 shown in Figure 18). The 9th question was attempted only by 5 students. The 6th student did not attempt this question as she exited from the trail after 1st question because of the time constraints.

TOPIC_ID	NUM_TRLS	MAX_WA	MIN_WA	AVG_WA	MIN_CA	MAX_CA	AVG_CA	MIN_TT	MAX_TT	AVG_TT
22	6	18	0	8	1	8	5	80	622	421

ND_Q_ID	SUM_ET	SUM_TT	MAX_TT	MIN_TT	AVG_TT	SUM_CA	SUM_WA	MAX_WA	MIN_WA
7	180	286	80	19	48	3	3	1	0
9	150	158	56	14	32	1	4	1	0
10	150	149	46	14	30	3	3	2	0
12	150	141	50	17	28	5	0	0	0
14	300	173	55	19	35	1	4	1	0
16	300	160	45	19	32	2	4	2	0
18	300	132	44	17	26	3	2	1	0
20	300	161	40	23	32	0	6	2	1
21	300	162	41	16	32	4	2	2	0
24	500	275	81	34	55	1	4	1	0
25	500	251	80	29	50	0	6	2	1
26	500	260	76	40	52	2	4	2	0
27	500	216	68	31	43	3	4	2	0

Figure 17. Overall and by question trail statistics.

TRAIL STATISTICS FOR QUESTION 9

ND_ID	ND_TRAIL_ID	STUDENT	ND_Q_ET	ND_Q_WA	ND_Q_CA	ND_Q_TT
27	5	NANDINI	30	1	0	14
89	8	BHARGAV	30	1	0	56
134	11	GNANESHWARI	30	0	1	27
179	14	DHANALAKSHMI	30	1	0	26
205	15	MOHAN	30	1	0	35

Figure 18. Trails statistics for question 9.

5.2 Additional Comments

Currently, we allow creating a single trail for each leaf topic, which will be extended to support multiple trails. Also, we will present a bird's eye view of the trail using the recorded property graph as the student makes progress, which he can navigate to review and make changes at the end of the trail completion.

Adding question in this application has simple and easy steps. We can update and make changes in questions, and time of the particular question by directly updating underlying tables through APEX table browser. For lower grades, for topics such as 'Numbers', which has very basic questions, it might be difficult to give any hints. For those questions, we may hide the hint option.

Based on our experience, we may provide an option where *forced_exit* and *auto_skip* feature can be turned on or off. Students taking assessment have time constraints. For example, first level question has 30sec time to answer, 2nd level has 60sec, and 3rd level has 90sec. If student spends more time on any question then the *forced_exit* condition can be trigger termination of the trail. In both the class sessions, we found students were unaware of the time constraint and spent longer while answering initial questions and their trail was terminated. So we are thinking of introducing 'starting buffer time' to provide extra time for initial questions.

An important reason for developing *pg-MathTrails* was that it runs in a wireless LAN without requiring Internet. The benefits are that it has interactive response time and there is no instability due to fluctuating speed of Internet or the site hosting the online

assessment application. Furthermore, students also stay focused as they are not tempted to browse other Internet sites.

The approach of storing each individual student performance on a trail as a separate graph opens interesting possibilities. For example, this can be used to conduct a Math Challenge contest, where the progress of various participants can be observed live.

Our vision is to expand *pg-MathTrails* to *e-Learning Trails*, which will allow creating learning trails for all subjects. The present application is well suited for Math as answers to Math questions tend to be precise and numeric. However, the current application is not suitable for question such as proving theorem. Despite this we feel Math is a good fit since the concepts and logic of the math theorems gets exercised in solving numeric problems.

Currently, the following question templates are supported 1) `'Single answer'`, 2) `'Choose the best answer'`, and 3) `'True or False'`. These question templates can be used to form questions for all subjects, including Science (Physics, Chemistry, and Biology), Social Science and Languages (English). However, the current set of question templates do not provide direct way to form questions like `'Fill in the blanks'`, `'Match the following'`, `'Odd man out'` and `'Paragraph based group of questions'`. We can handle these using current templates in the following way:

- By using `'Single answer'` template, we can form `'Fill in the blanks'`, example, `1, 2, _, 4` and `'Odd man out'` question, example, `1, 3, 8, 5`.
- Similarly by using `'Choose the best answer'` we can form `'Match the following'` question:

```
1. 2 + 3       a) 10
2. 2 x 5       b) 5
3. 15/5        c) 14
4. 16 - 2      d) 3
Ans: A. (1-a), (2-b), (3-c), (4-d)
     B. (1-b), (2-a), (3-d), (4-c)
     C. (1-b), (3-a), (4-d), (2-c)
     D. (3-b), (4-a), (1-c), (2-d)
```

For questions with numeric answers, checking the answer is easy. However, for question requiring answer in the form of phrase or sentences is difficult to match. These kinds of questions are common in subjects like Science and English. For example,

`'Sunday is Raju's favourite day of the week. He likes it because on Sunday, he watches football. On other days, he also gets to watch football but not all day.'`

The question `'Which is Raju's favourite day?'` can be answered either as `'Raju's favourite day is Sunday'`, or, `'Sunday'`, or `'Sunday is Raju's favourite day'` making it difficult to match the answer.

6. RELATED WORK

There are quite a few online assessment modules available (Table 1). Khan Academy [7] is one of the popular free educational sites, having good collection of videos and assessments on various topic categorized by grades. We utilize the videos provided by them for our students. LearnNext [13] is somewhat similar to Khan Academy but meant for primary and high school students with Maths and Science videos and online tests. IXL [16] is another good site that covers the complete math syllabus for K-12th grades with good assessment pattern, but charges membership fee. emathematics.net [11] is another good site for students to practice.

Math-Drills [10] is an interesting site, which provides free Math Worksheets on a variety of math topics for classes 1 to 12. However, the sheet has to be downloaded and printed and does not have online testing capability.

Math Trail [12] tests geography and math skill together, where student has to find the location on the map and then proceed to answer the question. The idea of the trail is similar to our application. However, in our application, the topics and the questions form the trail as opposed to real geographic locations.

Table 1: Comparison of Math Assessment Software

SOFTWARE	*Q. in the form of Games*	*Q. in increasing level of difficulty*	*Links to videos*	*Hint for Q.*	*100% Syllabus Coverage*	*Detailed student report*	*Multiple Subjects*	*Free*	*Off line*	*Alexa+ Website rank*
Khan Academy	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	No	3,308
IXL	**Yes**	**Probably**	No	No	**Yes**	**Probably**	No	No	No	8,169
Math-Drills	No	No	No	No	**Yes**	No	No	No	**Yes**	93,774
LearnNext	**Yes**	No	**Yes**	No	**Yes**	**Yes**	No	**Yes**	**Yes**	92,118
Math Trail	**Yes**	No	No	No	No	No	**Yes**	No	No	1,006,101
emathematics.net	No	No	No	No	**Probably**	No	No	**Yes**	No	1,892,116
pg-MathTrails*	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**-**

**auto_skip*, *forced_exit*, and comparing two trails are supported in *pg-mathTrails* only.
+Website rank tends to change. The current ranks reported are based on 6th October 2014.

Also, our application differs in that it is built on property graphs, which enable in-depth analysis of student's performance on the trail, allows comparing two trails, and provides overall student participation. Plus, our application supports features such as *auto_skip*, or *forced_exit*, which we have not seen in these sites mentioned above.

Property graphs [2] have recently become popular for handling large scale connected graph like data. Property graphs are supported by graph databases such as Neo4j [3], Sparksee [4] and Infinite Graph [14]. Graph databases are used to build social networking and scientific applications [9]. Companies such as Facebook, and Twitter have developed their own graph databases social graph and interest graph respectively.

We are using a custom property graph to store the default trail as well as student trails in our Math assessment module. Our approach is similar to defining typed graph classes to enable a statically type query language for property graphs [8]. That is, instead of defining a generic structure such as Entity-Attribute-Values (EAV) [15] to hold properties of nodes and edges, we define custom node and edge tables where properties are implicitly stored in additional columns.

7. CONCLUSIONS AND FUTURE WORK

The paper presented *pg-MathTrails*, an Math student assessment module built using property graphs. The distinguishing aspect of this application is to store topic trail as well as student performance on the trail as individual graphs that are readily available for performing in-depth analysis of student performance on individual trails, as well as for comparing two trails of the same student or two students. Instructors can create topic trails by making use of pre-existing question templates. It also features a video recommender that allows linking with relevant videos in a video repository.

The application has been tried out in an after school program with positive feedback. The application also features *auto_skip* and *forced_exit* capabilities to make the math trail interesting and challenging for students.

In future, we plan to handle evolution of a trail pertaining to a particular topic, and support 'Suspend and Resume' option for a trail, which will be useful for long trails. We also plan to extend the tool to support e-learning trails on other subjects as well.

Acknowledgments: We thank Khan Academy [7] for having such a wonderful site with a huge collection of instructional videos and making them available free to download. We thank the children in the after school program of Sri Sarada Academy of Excellence, Bangalore for participating in the usability study. We thank Vani S Kambi for proofreading the paper, and Priya Das of Sarada Research Labs, Bangalore for testing the application. We thank all reviewers for their useful suggestions.

8. REFERENCES

[1] WHOI Fitness Trail, http://www.whoi.edu/generalinfo/fittrail

[2] Property Graph Model · tinkerpop/blueprints Wiki · GitHub, https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model

[3] Neo4j Graph Database, www.neo4j.com

[4] Sparsity Technologies -- Powering Extreme Data, http://www.sparsity-technologies.com

[5] Oracle Application Express, www.oracle.com/technetwork/developer-tools/apex

[6] Oracle Database Express Edition 11g Release 2, www.oracle.com/technetwork/database/express-edition

[7] Khan Academy, https://www.khanacademy.org

[8] Tausch, N., Philippsen, M.,Adersberger, J.: A statically typed query language for property graphs. IDEAS 2011: 219-225

[9] Buerli , M., The current state of graph databases, 2012 http://www.cs.utexas.edu/~cannata/dbms/Class%20Notes/09%20Graph_Databases_Survey.pdf

[10] Math-Drills. www.math-drills.com

[11] emathematics.net, www.emathematics.net

[12] Math Trail. mathtrail.heymath.com

[13] LearnNext, www.learnnext.com

[14] Objective, www.objectivity.com/infinitegraph

[15] Nadkarni, P. M., et al., Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. JAMIA 6(6): 478-493 (1999)

[16] IXL, http://www.ixl.com