**Calculations/Logic for Learning Profile**

You will be making one table User (Student) Table and Teacher Role

Teacher must be able to see the analytics of students learning profiles

* **Student table will have following columns:**

completedQuestion, isCompletedProblemForLearningMode, isCurrentQuestion,

isCurrentProblem, appliedStrategy, completedTasks, incompleteTasks, completedProblems,

userId, password, capturedTasksAsBugs

* **For capturing errors for a student, use following information:**

1. The flagged instances of Formula class (whenever it is used wrongly in solving mode),
2. The flagged calculation mistake
3. The task flagged as wrong/incomplete in the first attempt,
4. The list of incomplete tasks,
5. The list of tasks captured by the property hasExpectedBug (from Answer class)
6. The list of tasks captured by the property hasExpectedTasks (from GE and TFE tracked in solving mode),
7. The diagram tool tracking links,

If you do not have code for getting some information elements from above please write the code for that. I have created four data type properties: openFormula(Formula, Any), resolvedFormula(Formula, Any), openTasks (Task, Any), resolvedTasks(Task, Any). We may have to create similar data type properties for diagram tool tracking. Let’s see once the diagram tool be ready. I have not removed the Bug class yet from the ontology. Will do it once, we make sure that this way of capturing the bugs work fine.

**Capturing Errors**

* **Learning Mode:**
* For selected answer take the hasExpectedBug property. Put all the instances in capturedTasksAsBugs of User class and update the openTaskCount by 1 in Task class.
* For every correct answer, get all the Tasks (by using hasExpectedBug property) for all hasOption of the question uri. Check for each task, if it is captured by the above capturedTasksAsBugs then update the resolvedTasks count by 1.
* **Solving Mode:**
* For any wrong input in Given, collect all instances of Task associated with given expression by the property “hasExpectedTasks” and put them in capturedTasksAsBugs in the user class and update the openTasks count by 1.
* When all Given expressions get entered, increment the resolvedTasks count by 1 for all the captured tasks associated with the GivenExpression by the property “hasExpectedTasks”.
* Follow same logic for ToFindExpression.
* For solution part: For every wrong formula statement put the instance in capturedTasksAsBugs in the user class and update the openFormula count by 1. If the formula statement is typed correctly and if it is already included in the capturedTasksAsBugs then upgrade the resolvedFormula count by 1.
* For every wrong statement (i.e. calculation mistake) in the solution part in between the formula statement and the task flag the error as “calculation mistake” create openCalculationMistake count and resolvedCalculationMistake count. Put these “calculation mistakes flag” in capturedTasksAsBugs of User class and update the openCalculationMistake count by 1.
* For each corrected calculation upgrade the resolvedCalculationMistake count by 1.
* For that tasks that mismatch initially must be put into capturedTasksAsBugs of the User class. For each of these tasks the openTasks count must be updateed by 1.
* When the problem is completed two things must be done:

1) Get all completed tasks from strategy and check if any of them listed as capturedTasksAsBugs of the user class and update the resolvedTasks count by 1.

2) Get all incomplete tasks from the strategy by subtracting completed tasks from all the tasks listed under that strategy and update the openTasks count by 1.

* **Calculation in learning profile:**

For each capturedTasksAsBug in the User class,

(Resolved Bug count/(Resolved Bug count + Open Bug count))\*100, rounded off to the nearest integer.

For the above formula as per the nature of capturedTasksAsBug, replace resolvedBug count by resolvedTasks, resolvedFormula, resolvedCalculationMistake count respectively. Similarly openBug count by openTasks, openFormula, openCalculationMistake count respectively.

* **Learned Concepts**

Load all completedTasks for the given problem type from the User model, and get the Supreclass of the Task as Task Name. From the tasks obtained get the type of **inverse\_of\_learnedConcepts** property. Then plot the bar graph for this combination i.e taskName ( inverse\_of\_learnedConcept name). Count must be done for the combination and not the individual task name.

For example: FindingInnerRadiusOfRCCyl (Has\_Dimension\_Of\_the\_Object\_Whose\_CrossSection\_Is\_Given)

* **Learning Profile of Mensuration domain**

For all the completed problems of User class get the Mensuration type using ofMensurationType property. Split the type at “\_” where the left hand side is the problem type and right hand side is the mesuration type (cube,cuboid etc).All combination type shapes are merged into CombinationOfDifferentShapes type.The mensuration types are fixed at following 9 types

Cube, Cuboid, RightCircularCylinder, RightCircularCone, Hemisphere, Sphere, FrustumOfCone, AnyStructure, and CombinationOfDifferentShapes. So any other mensuration type cannot be added dynamically.

* **Problem solving progress**

Get all completed problem and it types and difficulty level. For each of these problems for each difficulty level get the percentage of completed tasks to incomplete tasks from the user model.

* **Calculations**

All percentages must be rounded off to nearest decimal places. For each problem type:

easy\_correct = (sum(completed\_tasks for easy problems)/(sum( completed\_tasks for easy problems) + sum(incomplete\_tasks for easy problem)))\*100

-----------------------------------------------------------------------------------

medium\_correct = (sum(completed\_tasks for medium problems)/(sum( completed\_tasks for medium problems) + sum(incomplete\_tasks for medium problem)))\*100

----------------------------------------------------------------------------------------

hard\_correct = (sum(completed\_tasks for hard problems)/(sum( completed\_tasks for hard problems) + sum(incomplete\_tasks for hard problem)))\*100

-----------------------------------------------------------------------------------------

Overall = (easy\_correct + medium\_correct + hard\_correct)/3

* **Activity (for one student)**

On first opening the learning mode/ solving mode for a problem the learningStartTime/ solvingStartTime must be entered in the ontology for the given problem. Similarly on learning completion and solving completion of the problem, the learningEndTime and solvingEndTime must be updated. The graph must be plotted for the time taken from startTime to endTime for Learning and solving mode in seconds.

* **Activity (for all students) (For Teacher visibility)**

For the total graph of Learning Mode activity of all students and Solving Mode activity of all students Collect individual student’s LearningMode Time (in secs) and Solving Mode Time (in secs) and plot a double bar graph by putting students on x-axis and time in seconds on y-axis

* **For Teacher visibility**

Put selection boxes and run the SQL query for each selection and display the information either as a list or as a graph (as per the requirement).