# Intelligent Online Math Tutorial

## Design Document

**S Vidyasekar (MT2011131)**

**Subhashree S (MT2011155)**


**IIIT – BANGALORE**

Under the guidance of **– Prof. CHANDRASHEKHAR RAMANATHAN**

## 1. Introduction

The purpose of this design document is to describe the system architecture and component design in detail. It gives an in depth overview of how each module of the system is designed and how it is connected to the other modules. This document is intended to serve those who wish to understand the internal technical details of the application implementation and/or continue with the future work.

## 2. System Overview

The functionality implemented in the system can be broadly divided into two, namely learning mode and solving mode. A user first logs into the system using username and password. The system may contain multiple problem types from more than one difficulty level. The user has a choice of problem type and difficulty in the Home page (first page after login). A problem from the selected problem type and difficulty level is displayed to the user. The user has a choice of modes between learning and solving modes.

Learning mode gives the user a set of questions and multiple options for each question. Each question has only one correct option and only on selection of the correct option, the user can proceed to the next question. All options (correct and wrong) have feedback that has to be displayed to the user. On completing all questions in learning mode, the user is automatically redirected to solving mode. Learning mode is aimed at making the user understand the problem. The user gets a basic idea about the problem and how to approach solving it, in learning mode. From the mode selection page, the user can proceed directly to solving mode without going through learning mode.

Solving mode consists of six parts, Given, to Find, Solution, Semantic Knowledge, Schematic Knowledge and Formulae. In solving mode, the problem is displayed to the user and the user has to identify statements about the problem from any of given, to find and solution. User enters these statements following a pre-defined grammar. The statements are validated for correctness of grammar as well as validity of data. When the user has entered all relevant given, to find and solution in the respective parts, the problem is said to be completed and the user is taken to the home page. Solving mode has three options to help the user. They are Semantic Knowledge, Schematic Knowledge and Formulae. Schematic knowledge gives basic knowledge about the problem type selected by the user. Semantic knowledge gives further information about the problem type such as shape used in problem, etc., Formulae option displays all the formulae that are present in the system.

At any point of time, the user can switch between learning and solving modes. And the user is allowed to continue from the point that the user left from without having to redo from the beginning.

The idea behind this system is to build a tutoring system that can be extended to more than one domain in mathematics (geometry, algebra, etc.,). As a prototype, certain mensuration word problems are implemented and tested. The idea is to let the student use the system, provide meaningful feedback and track user actions as necessary. From tracking a student model can be built that can be used to create student learning profiles. Such learning profiles can enable the administrator to improve the system further such that learning experience can be enhanced.

The system is designed as a semantically enabled system, using OWL ontologies as backend. OWL file and Java program communicate with an API known as Jena framework. OWL supports native inferences and rule based inferences. Rule based inferences are written using a language called SWRL. SWRL rules can be executed in Jena framework using external reasoners. For our system, we are using a pellet reasoner integrated with Jena to run SWRL rules. The system is implemented as a web-based system using struts2. Further detailed architecture and components are explained in the following chapters.

# 3. System Architecture

## 3.1 Architecture Design

The system can be broadly divided into four modules. User interface, logic, data access and ontology file. User interface is a collection of jsp files. Logic module is the level below user interface. It receives input from the user interface and translates it into function calls. The function calls access data access modules or functions within the logic module.

The logic module gets result from the data access module after processing and displays it in the user interface. The logic module acts as the only interface between the user interface and the data access module. Some functions of logic module are getFirstQuestion(), validateGiven(), checkAnswer(), loadLearningMode(), etc.,

Data access module has lower level functions that work directly with the OWL file. The logic module calls corresponding functions from data access module. The data access module translates them into data level functions and returns relevant data structures. Some data access functions are runQuery(), loadData(), findTriple(), insertProperty(), etc.,

The data needed by the system is stored in an OWL file. An OWL file is an ontology file stored in RDF format. It consists of Classes, Objects and Properties. Properties are used to connect objects to other objects or objects to data types. The ontology file also consists of rules in SWRL format.

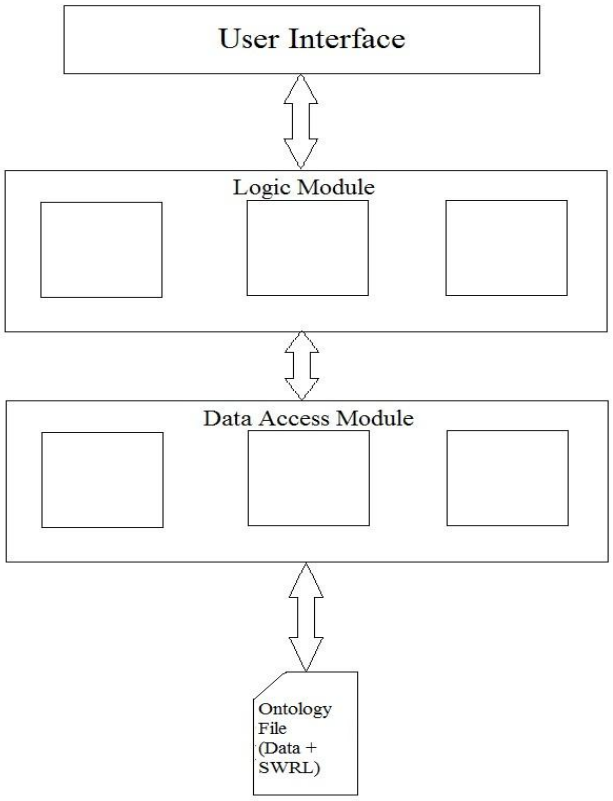


Fig 1: System Architecture

For example, the actions performed when user enters a statement in solving mode are explained. The user enters a statement in say solution. The checkSolution() function is called in Logic module. This function takes the solution statement as input and checks if it follows the pre-defined grammar. Then, the left hand side of the statement is used and queried for the right hand side in the ontology and the returned result is checked with the right hand side of the statement. The query function is implemented in data access module. The query function runs a SPARQL query on a model of the ontology that was loaded at the start of execution of the program.

## 3.2 Decomposition Description

## Logic module and user interface

The logic module is divided into multiple java files. Each java file has an associated jsp file. The interaction between various java files and jsp files is described in the diagram below.

## Data access module

The data access module consists of three java files. GlobalData.java, Tables.java and OwlAccess.java. OwlAccess.java contains all functions that interact directly with the ontology

file. GlobalData contains data needed for learning mode and Tables contains all data needed for solving mode. All three files are implemented as singleton classes.

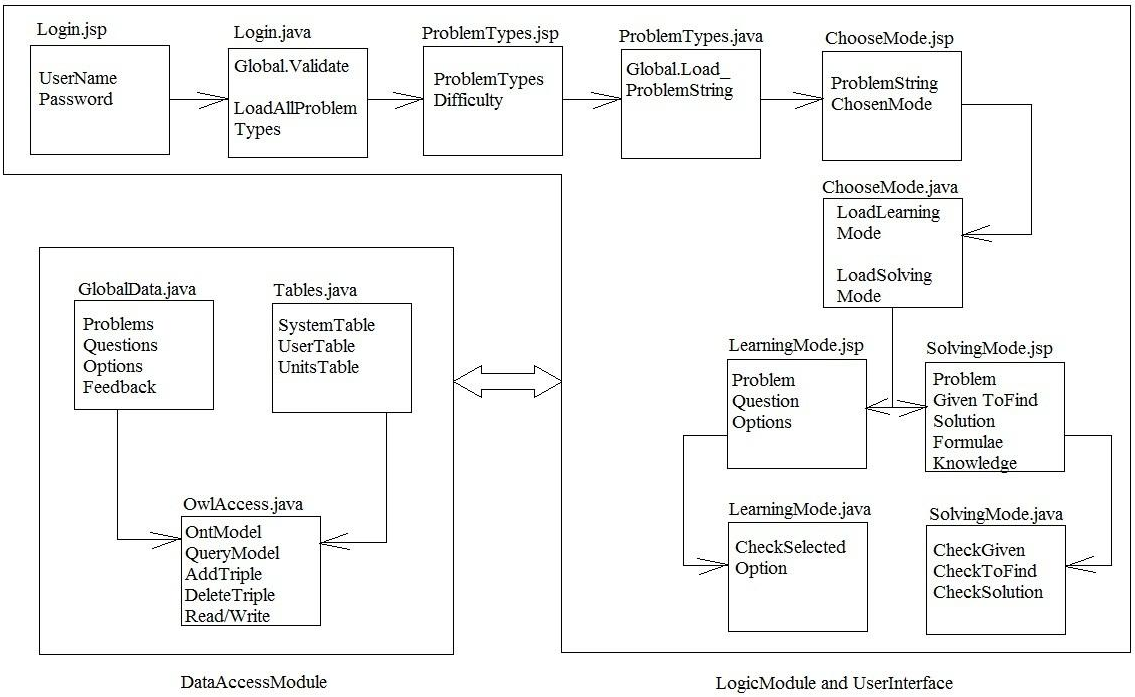


Fig 2: Component Architecture

## 3.3 Design Rationale

The system is implemented as a multi-tier architecture. Such architecture increases the independence of each module. In our system, the ontology is continuously evolving. An evolving ontology means continuous change in data format, data structures, etc., We need a system that is open for change. A loosely tied multi-tier system fits these needs. The level of independence is not absolute due to query result format of SPARQL. SPARQL queries return URI instead of actual data. The actual data or result is returned as a substring of the URI. Extracting this data is dependent on the query. The query is formed in the Logic module. And processing the URI string also happens in the logic module. This creates cohesion in the system. The data access module is created as singleton class. The implication of which is, there is only a single connection from the system to the ontology file. The data from ontology is used to populate data in the data structures. There exists only one instance of the data structures (namely GlobalData and Tables).

# 4. Data Design

## 4.1 Data Description

## Ontology Owl file

Data for the system is contained in the Ontology file with .owl as extension. Data required for learning mode are, Problems, questions, options and correct option. Data associated with user are, Current problem, current question, selected option, current question, completed problems. For solving mode, we need to represent the values in the problem within the ontology, given, to find, tasks, strategies.

## System entity table

System entity table contains three attributes, entity, value and unit. Entity is defined as any measurement given in the system. Example: length of a cuboid. Using our predefined grammar, it is represented as Cuboid.Length. The string is stored as such in the entity field. The corresponding value and unit are extracted from the ontology and stored respectively.

## User entity table

The system entity table stores all values from ontology. Whereas, the user entity table stores only those values the user has explicitly entered in given, to find or solution. The values user enters in to find are stored with a '?' for value.

## Unit table

Unit table is used in unit conversion. It consists of three attributes; Source unit, destination unit and conversion formula. The conversion formula is a function f(source unit) and the left hand side of which is implicitly assumed as the destination unit. Example: cm m cm/1000.

## Problem Type List

It is list of URI of problem types in the ontology file.

## Problem List

For each entry in problem type, a problem list is created. This list stores list of all problems of the chosen problem type.

## Question List

For each entry in problem, a question list is created. This list stores list of all questions in the given problem.

## Option List

For each entry in question list, an option list is created. This list stores list of all options in the given question.

## 5. Limitations

1. Some rigidity has been enforced in the format for filling the Solution part in Solving mode:
   - There should not be any space between the expression for the formula or the substitution of numerical values for the formula.
   - Only parentheses are allowed for sequencing of operations in a formula expression. Symbols like [] are not allowed.
2. There is some delay when the learning mode page gets loaded.

## 6. Future work

1. Learning profile of the student shall be displayed to the student based on the bugs captured by the system as a result of the user's actions
2. When the formula button is clicked in Solving mode, all the formulae stored in the ontology are displayed to the user. Instead, only the formulae pertaining to the current problem can be displayed.
3. Along with schematic and semantic knowledge, diagrammatic representation of the problem can also be included.
4. The user should not be allowed to start the solving mode with solution part directly, when the given and to find parts are empty.
5. The rules for unit conversion have been written into a file and this file is being used for checking units. Instead, this conversion can be stores as a part of the ontology itself.
6. A scratch pad can be included in the learning mode which shows the set of questions and answers completed by the user so far. It shall include the wrong answers chosen by the user too. This will help the user keep a track of the mistakes he has made in each step.