**Coding overview for QBank**

QBank is written in HTML, CSS, JavaScript and PHP.

1. **QBank Homepage: index.html**

Our homepage consists of straightforward introduction to the interface and two options to “Write a question: and “Find and Edit a question”. Both the options are displayed as links which redirects the user to the following pages to do a particular task.

1. **Stylesheet: new.css**

The style sheet for the interface is written in CSS. This is linked with index.hrml in head tag –

<link href="new.css" rel="stylesheet" type="text/css" />

This document specified the font and color standards that we use across all the various webpages. This provides for a standard that the interface can follow.

**Write a Question:**

The first link on the front page is to “Write a question”. The user is redirected to a form which assists in creating a question. There are 3 parts to this code. Firstly, the Html code that provides for the form display. Secondly, the JavaScript code that dynamically generated content onto the web pages. Finally, the PHP code that writes the user entered data onto the database which is a text file.

1. **HTML part- editQuestion.html**

Using a table structure to format the display of the page, we use a <table> tag with the following elements are <td> and <tr> tags in it.

* **Question as (File Name):** In the interface, every question is stored with a Filename. This allows for easy retrieval for updating/deleting. This will help the question writer to identify and differentiate between the several questions that he’s written.

**For e.g.-**  Question\_1, Q\_Add are examples of file names.

* **Intro text:** A common introduction contains information that is relevant to more than one question and can be shared by related questions. In case of parameterized questions, this serves the purpose of redundant and repetitive writing of an introductory text to a question. This textbox also describes the type of question and operation that will be performed by the question.

**For e.g.-** An exercise to add 2 numbers

* **Questions:** This textbox accepts the actual question with its parameters, operations, etc. For parameterized questions, the variables in the question text should be specified with a “@” sign preceding it.

**For e.g.-** @a + 2, where @a signifies the parameterized part in the question which can take different values based on the type of question selected by the user.

* **Add a parameter name and value**: This is a link that dynamically runs a JavaScript which displays a radio button option to select if the parameter in the question is a list type or a range type.

**Code Snippet-**

<tr><td valign="top"><div id="variable" style="display:inline">

<a href="#" style="cursor:pointer" onClick="createRadioElement('var1', 'variable', 'variable1')" > Add a parameter name and value</a></div>

<div id="variable1" style="display:none">Select variable Type for all variables:</div>

</td>

<td><div id="var1" name="var1"></div></td></tr>

**Code Explanation:**

onClick =”createRadioElement(‘var1’,variable’,’variable1’)” , calls the JavaScript function createRadioElement and passes 3 variables which serve as the placeholders.

style=”display:none” , ensures that the <div> remains hidden until it is set to “display:inline” within the JavaScript function. This supports showing only relevant information to the user depending on the choices made by the user than overloading the user with information.

* **Add an answer:**  Based on the parameter type selected by the user, there are 2 different types of answer formats that will allow the user to specify the correct answer. This link dynamically runs JavaScript and generates the textbox for range and table for list type parameters, where the user can feed the correct answer function/values.
* **Save:** This button saves the entered values in a text file format and a format compatible with Khan Academy framework.

1. **JavaScript functions-**

In this section, we will explain a few important functions we used to dynamically generate different parts of the web interface based on user choices.

1. When the user clicks the option, “Select question type”, a radio button choice of “List” and “Range” appear.

function generateRadio(place) {

var area = document.createElement('input');

area.setAttribute("type", 'radio');

area.setAttribute("name", 'var');

area.setAttribute("value", 'list');

document.getElementById(place).appendChild(area);

i++;

}

1. When the user chooses a radio button, for e.g. “list”, a parameter-value entry form layout appears, which allows the user to specify the values for different parameters.

In the List option, the user has to enter the values for the parameters as comma-separated values.

function addListVariable(){

var tbl = document.createElement("table");

tbl.setAttribute("border",'2');

var row1 = document.createElement("tr");

var r1c1 = document.createElement("td");

var text1 = document.createElement("div");

text1.innerHTML= "Variable Name";

r1c1.appendChild(text1);

row1.appendChild(r1c1);

var r1c2 = document.createElement("td");

var text2 = document.createElement("div");

text2.innerHTML= "Values separated by commas";

r1c2.appendChild(text2);

row1.appendChild(r1c2);

tbl.appendChild(row1);

var var\_name = "list"+list\_var\_num+"\_name";

var var\_val = "list"+list\_var\_num+"\_val";

var div\_name = "div"+list\_var\_num;

var row2 = document.createElement("tr");

var r2c1 = document.createElement("td");

var cellText = document.createElement("input");

cellText.setAttribute("type",'text');

cellText.setAttribute("name",var\_name);

r2c1.appendChild(cellText);

row2.appendChild(r2c1);

tbl.appendChild(row2);

list\_var\_num++; // Counts the number of list variables

document.getElementById('var1').appendChild(tbl);

}

1. For specifying the answer, the interface has to create Table layout to help the user input answers for list type of parameters without making errors. In order to do so, we decided to use an algorithm that would allow us to cover all the possible answer combinations for the inputted parameter values.

function generateAnswer(ans){

var num\_cols = values[list\_var\_num-2].length;

var row\_col = "(" + num\_rows + ", " + num\_cols + ")";

for(var i=0; i<num\_rows; i++){

A[i] = new Array();

}

var combination = new Array();

var comb = new Array();

for(var i=0; i< list\_var\_num-1; i++){

combination[i]=0;

comb[i]=0;

}

combination[list\_var\_num-2] = 1;

comb[0]=1;

for(var i=list\_var\_num-3; i>=0 ; i--){

combination[i]=values[i+1].length\*combination[i+1];

}

for(var i=1; i<list\_var\_num-1; i++){

comb[i] = comb[i-1]\*values[i-1].length;

}

for(var i=1; i<list\_var\_num; i++){

for(var l=1; l<=comb[i-1] ; l++){

for(var j=1; j<=values[i-1].length ;j++){

for(var k=1; k<=combination[i-1]; k++){

var temp3 = (l-1)\*values[i-1].length\*combination[i-1];

var temp = (j-1)\*combination[i-1];

var temp2 = temp3+temp+(k-1);

A[temp2][i-1] = values[i-1][j-1];

}

}

}

}

**3) PHP code: formsave.php**

**A) File Creation:**

Using the File name entered by the user, PHP code helps create a text file in the back end with the name which will help during file retrieval and update.

<?

$fileName = $\_POST['file\_name'];

$myFile = ' /home/algoviz-beta/QBank/OpenDSA/Intermediate\_files/'.$fileName.'.txt';

$fh = fopen($myFile, 'w');

?>

**B) Computing index function for answers of dynamically generated parameters:**

The answers entered by the users using the tabular format provided at the front-end need to be stored in a one dimensional array in the intermediate format. Also, to retrieve the correct answer based on the generated values of the parameter combinations at run-time, we need to compute a function for finding the index of the correct answer stored in the array.

We have made a generic algorithm that generates such a function dynamically.

For e.g. –

@a is a parameter that can take values [2, 3, 4]

@b is a parameter that can take values [8, 9]

There are 6 possible outcomes for the answer which will be stored in a one dimensional array (assuming array index is starting at 0) as follows:

[{ans(2,8)}, {ans(2,9)}, {ans{3,8}}, {ans(3,9)}, {ans(4,8)}, {ans(4,9)} ]

Answer of combination (2,9) has the array index is 1.

Similarly, for combination (3,8), the array index is 2 and so on in the one-dimensional array

The generic formula for finding the answer index is computed as follows:

Number of values of @b \* index of @a + index of @b.

In the above case, since the number of values of @b is 2, and considering index of @a and @b to be i and j respectively, the formula for index computation of answers is 2\*i +j.

The above formula can also be extended for more than 2 variables and their values. Our algorithm takes care of all the computations based on number of dynamic variables and values.

1. **Storing text box values**

**(i)Text Values**

They are stored as a name value pair separated by “=>” and “$”.

<?

$stringData = "Inroduction text => ";

$stringData .= $\_POST['intro\_text'];

?>

**(ii)Comma Separated values**

For the comma separated values inputted into text boxes, it required an explode function to extract the values separated by commas in a text box and place them in an array.

<?

for($i=0; $i<sizeof($list\_name); $i++){

$index = "x".($i+1);

$stringData.= "<var id=\"";

$stringData.=$list\_name[$i];

$stringData.="\">[".$list\_val[$i]."]";

$stringData.= "</var>";

$stringData.= "<var id=\"";

$stringData.=$index;

$stringData.= "\">randRange(0,";

$valArr = array();

$valArr = explode(",", $list\_val[$i]);

$stringData.= sizeof($valArr)-1;

$stringData.= ")</var>";

} ?>

**Searching for a question: searchQ.php**

It displays out all the created question Text files and asks the user which Text file he would like to edit.

<?php

{

$pathQ = ' /home/algoviz-beta/QBank/OpenDSA/Intermediate\_files;

$execCmd = 'ls -rt '.$pathQ.' | tr " " "\t"';

$output = shell\_exec($execCmd);

$filearray = explode(".txt", $output);

for($i=0; $i<sizeof($filearray); $i++){

echo "<p>";

echo $filearray[$i];

echo "<p>";

} ?>

**Editing a question: QEdit.php**

The interface displays all the values entered for the particular question in text boxes which will enable the user to change values and save changes. This rewrites the particular file with new values.

In case of an invalid file name, the user is requested to return to the previous page and re-enter a valid filename.