Technical Manual
Group 2
Online Programming Portal

NOTE: Make sure that you are well versed with the User Manual before going through this Technical Document.

Index

Introduction

Requirements:

all the compilers of the following languages should be installed ::
c,c++,awk,bash,brain,c#,java,javascript,pascal,perl,php,python3,ruby,text

The software has been developed in net beans 8.0 and other editors that support php,html,css,javascript,jquery and ajax.

Users ::

This portal has five kind of users:

1. **Admin:** Admin can access all portions of the portal. Admin can create contests, add and edit problems , broadcast messages, give clarifications , modify team and group settings , re-judge and disqualify solutions.
2. **Problem Setters:** These users can add problems and edit the problems they have added. They can also re-judge and disqualify the solutions of the problems they have added.

3. **Team**
4. **Individual Users**

Team and Individual users can take part in contests and solve practice problems.

5. **Non-Registered Users**

non-Register users can access all the problems, tutorials, forums and scheduled contests. but these users cannot participate in contest, cannot submit their solutions and cannot post question or answer on the forum.

Features ::

1. **Lightweight:** This portal is lightweight yet has the essential basic features.
2. Supports **13 Languages**:
    (a) C
    (b) C++
    (c) Java
    (d) Python 2.7
    (e) Python 3
    (f) Javascript
    (g) PHP
    (h) Bash
    (i) Perl
    (j) Pascal
    (k) C#
    (l) Ruby
    (m) BF

3. **Forums:** A place where participants can discuss about various problems. Users can also search the forums by tags.
4. **Search:** Search a problem by its tags or name.
5. **Tutorials:** Various tutorials for languages like C,C++,Java,Python.
6. **Ranking:** Maintains a list of user sorted by the scores.
7. **User profile:** A place where user can view his/her recent submissions and stats.
8. Link to search for compilation errors.
9. Complete log of activities is viewable by admin.
10. Admin can broadcast certain messages to all the users at once.
11. **Compile online :** Facility to just execute general programs and see their outputs(just like compiling the program online).
12. Coloring of problems according to the status of the problems(solved,unsolved).
13. Easy to host and operate on the server side.

Database

1. **Introduction**

One goal of the initiative is to standardize data management and to facilitate the sharing of comparable and accurate specimen-level paleontological data within a global scientific consortium. A history and rationale for these efforts is provided in this manual.

The database also holds a small amount of invertebrate and fish data. It contains the technical information about the system architecture, data tables, security etc. It also contains details of changes made to the data during migration from the old system. This manual is accompanied by the User Guide.

2. System requirements

Processor Pentium P400 or better
Xampp/PHPMyAdmin for database architects, developers etc.
Operating system: Linux
Screen resolution Min of 1024 x 768 Optimized for 1280 x 1024
MySQL components
The full version of Xampp is recommended.
Browsers : firefox,google chrome,internet explorer

3. Data tables

These are in the SQL data files provided with the software/website.
There are 39 data tables in the Database.
The data tables are as follows:

Tables in Database:-

1. Admin: (variable, value)
2. Anon_run: (rid, language, code, input, output, error, submittime, timetaken, tid, verdict)
3. Broadcast: (id, title, msg, createdOn, updatedOn, deleted)
4. clar: (Time, tid, pid, query, reply, access, createtime)
5. Contest: (id, code, name, starttime, endtime, announcement)
6. Groups: (gid, groupname, status)
7. Logs: (time, ip, tid, request)
8. Problems: (pid, code, name, type, contest, status, pgroup, statement, image, imgext, timelimit, score, languages, options, dishplayio, maxfilesize, input, tid)
9. Qa_blobs:()
10. Qa_cache:()
11. Qa_categories:()
12. qa_categorymetas:()
13. qa_contentwords: (postid, worded, count, type, questioned)

14. qa_cookies:()
15. qa_iplimits: (ip, action, period, count)
16. qa_options: (title, content)
17. qa_pages:()
18. qa_postmetas:()
19. qa_posts: (postid, type, parented, categoryid, catidpath1, catidpath2, catidpath3, account, amaxvote, selchildid, closedbyid, userid, cookieid, createip, lastuserid, lastip, upvotes, downvotes, netvotes, lastviewip, views, hotness, flagcount, format, created, updated, updatetype, title, content, tags, name, notify)
20. qa_posttags: (postid, worded, postcreated)
21. qa_sharedevents: (entitytype, entityid, questioned, updatetype, lastuserid, updated)
22. qa_tagmetas:()
23. qa_tagwords: (postid, wordid)
24. qa_titlewords: (postid, wordid)
25. qa_userevents: (userid, entitytype, entityid, questioned, lastpostid, updatetype, lastuserid, updated)
26. qa_userfavorites:()
27. qa_userlevels:()
28. qa_userlimits: (userid, action, period, count)
29. qa_usermetas:()
30. qa_usernotices:()
31. qa_userpoints: (userid, points, qposts, aposts, cposts, aselects, aselecteds, qupvotes, qdownvotes, aupvotes, adownvotes, qvoteds, avoteds, upvoteds, downvoteds bonus)
32. qa_uservotes: (postid, userid, vote, flag)
33. qa_widgets:()
34. qa_words: (wordid, word, titlecount, contentcount, tagwordcount, tagcount)
35. runs: (rid, pid, tid, language, time, result, access, submittime)
36. subs_code: (rid, name, code, error, output, tcpassed, failed_tc, correct_output,)
37. teams: (tid, teamname, teamname2, pass, status, score, penalty, name1, roll1, branch1, email1, phone1, name2, roll2, branch2, email2, phone2, name3, roll3, branch3, email3, phone3, platform, ip, session, gid, vercode)
38. testcase: (id probed, input, output)
39. tutorials: (id, title, content)

judge.py -

Requirements:

The script works only in Linux.

bf ,bc ,g++,gcc ,fpc, mono-gmcs ,oracle-java/openjdk-6-jdk ,perl ,php5 ,python ,python3 ,python-mysqldb ,rhino ,ruby modules to be installed in *NIX system.

platform,re, os, shutil, sys, thread, time, urllib, SocketServer, subprocess, MySQLdb libraries to be installed for python.

Overview:

This script is the judge of the site. Whenever this script is executed it creates a server at the requested ip using the requested port. The script can be easily configured as required. A sample configuration is given below.

```
sql_hostname = '127.0.0.1'
sql_hostport = 3306
sql_username = 'root'
sql_password = 'root'
sql_database = 'matrix_exp'
HOST, PORT = "127.0.0.1", 8725
```

We can set the appropriate constants as required. HOST and PORT is the HOST and PORT where the judge server will be instantiated. You should have the permission to do so. The 'sql' constants specify the address of the database.

The judge creates a TCP server. Whenever data is sent to this server the server parses the data and acts accordingly. A request with prefix 'com' specifies the request for compile and run for the online IDE. A request rejudge evalutes all the problems where the result has been set to 'NULL' , the remaining requests are evaluated as compile and run for general submissions.

Basic Working of Judge:

1.Judge fetches the submission from the database. Then according to the language some prepocessing is done on the code to check for malicious code.(Presently it does so only for C,C++,Python,Python3) .

2.The code is compiled if required . In the compile step the IO is redirected.

3.The input and output as present in the io_cache folder is copied into env folder and named as input.txt and output.txt

4.Then the code is executed using timeout command of UNIX. This sets the timelimit. The time for execution is calculated by calculating the time which the executable runs in the subprocess module.

5..The errors are all redirected to env/error.txt. The error is detected by getting the return code for the process executed.

6.To check if the output is correct, the output as present in env/output is compared against the correct output which is stored in io_cache folder.

7. If there are multiple test cases then the above steps after compilation is executed multiple times.The time is enforced is per test case but the total time is stored in database and that is displayed. In case of wrong anser or run time error the loop breaks and the result is updated on database.

functions.php

Functions:

- **processDirPath:**
    - **Parameters:**
        - **$dir (string):** Path string of directory to be converted.
    - **Return Value:** Returns a string of path converted to Windows style.
    - **Action:** Converts UNIX style path to Windows style path string.

- **GetAllFilesfromDirectory:**
    - **Parameters:**
        - **$dir (string):** Path of the directory to fetch the files from.
        - **$sort (boolean):** Flag to determine whether to sort the results or not. It is set if sorting is required. Default value is set to true.
        - **$sortby (string):** Attribute by which the result needs to be sorted. Default is set to “name” which gives output sorted by name only if $sort is set.
        - **$sortorder (flags):** Sort either in ascending or descending order. 'SORT_ASC' is for ascending order and 'SORT_DESC' for descending order. Default value is set to 'SORT_ASC'.
        - **$ignore (array of strings):** List of files to be ignored. Default is set to empty array.
    - **Return Value:** Returns array of files if successful else returns false.
    - **Action:** Fetches all files present in the specified directory if the directory with path specified exists.

Class DB:

Methods:

1. **initialize:**
    1. **Parameters:** None.
    2. **Return Value:** Return boolean if connection is built else returns false.
    3. **Action:** Establishes connection with database.
2. **closeConnection:**
    1. **Parameters:** None.
    2. **Return Value:** Returns true if connection is closed successfully.
    3. **Action:** Closes connection with database.
3. **handleError:**
    1. **Parameters:**

    a) **$e (error object):** Error encountered during execution is returned. Default value is set to null.

    b) **$data (string) :** Error string to be written in provided.

    2. **Return Value:** None.
    3. **Action:** Displays error in case of any.
4. **query:**
    1. **Parameters:**
        a) **$query (string):** The query to be made in database is provided as a string.
        b) **$values (array):** Required when performing extended query. Default is set to null.
    2. **Return Value:** Return object after performing query. In case of an error false is returned and error is handled by handleError.
    3. **Action:** As the name suggests, it is used to perform query from the database with some extended features.
5. **findAllWithCount:**
    1. **Parameters:**
        a) **$select (string):** String aiding query is provided usually a prefix of a query string.
        b) **$body (string):** A part of query string similar to '$select'..
        c) **$page (int):** As whole output is not displayed at once.
        d) **$limit (int):** Page limit.
    2. **Return Value:** Returns object retrieved after querying the database.
    3. **Action:** Performs query to retrieve a particular range of results.
6. **insert:**
    1. **Parameters:**
        a) **$table (string):** The table in which the objects needs to be inserted is provided.
        b) **$data (dictionary):** The data which is to inserted is provided.
    2. **Return Value:** Returns true if successful else returns false is returned.
    3. **Action:** As the name suggests, it inserts data into the desired table in the database.

7. **update:**
    1. **Parameters:**
        a) **$table (string):** The table to which the data has to be updated.
        b) **$data (dictionary):** The data which has to be inserted into the desired table.
        c) **$where (string):** Auxiliary string for aiding query. These strings are concatenateded to form the final query string.
        d) **$values (string):** Auxiliary string for performing query. Default value is set to null.
    2. **Return Value:** Returns true in case the query is performed else false.
    3. **Action:** As the names suggests, it updates the database, inserts values into desired tables.
8. **delete:**
    1. **Parameters:**
        a) **$table (string):** Name of the table from where the data has to deleted.
        b) **$where (string):** Auxiliary string to perform deletion.
    2. **Return Value:** Returns true if query is executed successfully else false is returned.
    3. **Action:** As the name suggests, it deletes the required entry from the table. It uses 'update' as its aid.
9. **findAllFromQuery:**
    1. **Parameters:**
        a) **$query (string):** Query string to execute. Fetches results matching the constraints.
        b) **$value (array):** For extended results '$values' is provided. Default value is set to null.
    2. **Return Value:** Returns object of results if query is executed successfully, else returns false.
    3. **Action:** Performs query and returns the final object after performing query.
10. **findOneFromQuery:**
    1. **Parameters:**
        a) **$query (string):** Query string to execute. Fetches results matching the constraints.
        b) **$value (array):** For extended results '$values' is provided. Default value is set to null.
    2. **Return Value:** Returns object of result if query is executed successfully, else returns false.
    3. **Action:** Performs query and returns the final object after performing query.
11. **logActivity:**
    1. **Parameters:**
        a) **$activity (string):** The activity that is affecting database is logged into database table 'activity_log'. Attributes of activity, including errors.
        b) **$message (string):** Messages including errors that are to be written in database.
        c) **$result (string):** The result of that activity is written along with '$activity' and '$message'.

2. **Return Value:** Return true if log in entered into database else returns false.
3. **Action:** Logs website activities and enters them in 'activity_log' table.

problems.php

This php shows the user the problems from the database and then any user can select the problem he wants to solve and then submit the solution to that problem.
functions of this php::
1. $result stores the rows from the problems table of the database of the selected problem.
2. $check stores the rows from the contest datatable if the problem is of a contest.
3. preg_replace() used to perform a regular expression search and then replace.
4.stripslashes() used to remove slashes from the statement of the problem which is stored in the database.

contact.php

This php gives the user the privilege to contact the admin in order to ask any queries regarding the problems.
Functions performed by this php::
1. $tid stores the team id which is logged in.
2. $result stores the rows from the clar table in the datatable which are meant for feedback.
3. Then this file gives the user a text box to post his query which is then inserted into the database so that adminclar.php can retrieve this row and then show this query to the admin.

rank.php

functions used are ::

```
function cmp($a, $b){
        if($a['score'] > $b['score'])
        return -1;
        else if ($a['score'] < $b['score'])
        return 1;
        else {
        if($a['time'] + $a['penalty']*20*60 < $b['time'] + $b['penalty']*20*60)
        return -1;
        else if($a['time'] + $a['penalty']*20*60 > $b['time'] + $b['penalty']*20*60)
```

```
        return 1;
        else
        return 0;
        }
}
```
above function is for comparing date and time..

```
function timeformating($a){
        $sec = $a%60;
        $a -= $sec;
        $a = $a/60;
        $min = $a%60;
        $a -= $min;
        $hr = $a/60;
        $str = "";
        if($hr > 0)
        $str .= $hr.":";
        if ($min > 0)
        $str .= $min.":";
        $str .= $sec;
        return $str;
}
```

in this php, we use a query $query = "select teams.tid, teamname,
group_concat(distinct(problems.pid) SEPARATOR ',') as pids
from problems, teams, runs , contest
where
runs.result = 'AC' and
problems.pid = runs.pid and
problems.contest = 'contest' and
runs.access != 'deleted' and
teams.tid = runs.tid and
problems.pgroup = '$_GET[code]' and
(problems.status = 'Active' or problems.status = 'Inactive' or problems.status ='deleted')
and
teams.status != 'Admin' and teams.status != 'Problem Setter' and
submittime<=endtime
group by teams.tid";
Above query is to find rank of users.

rankings.php

this php is used to calculate ranks of the users.

adminclar.php

any user will ask his doubts with the admin and admin will reply from clarifications.
If a user asks his query regarding a problem, his request goes to the setter of the problem, the admin/problem setter will get a clarification under that problem.
Else if the user sends a request from contact us tab his request will go to the admin and the admin will get the request as a feedback request.
Functions performed by this php:

1. this php checks whether the logged in candidate is admin or problem setter
2. then it runs a query to select rows from the datatable clar that have status as pending or deleted or replied.
3. it accordingly makes some tabs corresponding to each status and then for each row it checks whether the team id matches with the logged in team id and then accordingly displays the clarifications of that particular team(admin or the problem setter).
4. it then gives user a textbox to reply to the user and it automatically changes the status of the query to “replied”.

AdminGroup.php:

Through this the admin can add groups.There are two parts in the code. In which one part the user can add groups and in other part the user can see groups that are there.

```
echo "<h1>Groups Settings</h1>
    <form method='post' action='".SITE_URL."/process.php'>
    <div class='col-lg-3'><input class='form-control' type='text' name='groupname' /></div>
    <input class='btn btn-primary' type='submit' name='addgroup' value='Add Group' />
    </form>";
```

The request is sent to process.php where the group is added.Then the below code shows or lists the groups that are present.

```
$query = "select * from groups";
$result = DB::findAllFromQuery($query);
echo "<h3>List of Groups</h3>
    <table class='table table-hover'><tr><th>Name</th><th>Option</th></tr>";
```

Then the user can update or delete the group using the below code. The request is sent to process.php.

```
foreach ($result as $row){
    echo "<tr>
        <td>
        <form role='form' method='post' action='".SITE_URL."/process.php'>
        <input type='hidden' name='gid' value='$row[gid]' />
        <div class='col-lg-6'><input class='form-control' type='text' name='groupname' value='$row[groupname]' /></div>
        <input class='btn btn-primary' type='submit' name='updategroup' value='Update Group' />
        </form>
        </td>
        <td>
        <form method='post' action='".SITE_URL."/process.php'>
        <input type='hidden' name='gid' value='$row[gid]' />
        <input class='btn btn-danger' type='submit' name='deletegroup' value='Delete Group' />
```

AdminContest.php:

Using this admin can add or update the contest. This file contains the required UI and request is sent to process.php where it is handled. This code also contains the javascript to check whether the “end date> start date”.

```
function validateTimes()
{
    var start=new Date(document.getElementById("starttime").value);
    var end=new Date(document.getElementById("endtime").value);

    if(start>end)
    {
        alert("End Time should be after Start Time");
        return false;
    }
    return true;
}
```

The below code checks if the contest is existing one or not.

```
if (isset($_SESSION['loggedin']) && $_SESSION['team']['status'] == 'Admin') {
    if (isset($_GET['code'])) {
        $_GET['code'] = addslashes($_GET['code']);
        $query = "select * from contest where code = '$_GET[code]'";
        $res = DB::findOneFromQuery($query);
```

if the code is existing then it gives the option to edit. If it is a new one then it displays fiels to add a new contest.

Home.php

At the beginning of file, we are checking for either user is logged in or not.
If user is not logged in, then show the corresponding right side bar, i.e. login option etc. otherwise, show the ranking etc. in the right bar.
There is a query for the notices and stored the notices as a string in $data.
Then splitting that string in an array and showing on the home page all the notices as a list.

Register.php

At the beginning of file, I am checking either admin has given the permission of registering or not and also checking either user is logged in or not. If permission is given: there is a form for registering whose all the data will be transferred to process.php file where all the information will be saved in the database and user will get registered.
User can register himself as an individual or as a team of maximum 3 people.
If he is registering himself as individual then he must fill data only in member1 area, which is compulsory to fill.
And if he is registering as a team, he needs to fill the data corresponding to each member.
Member 1 will be considered as team leader, so all the information for member 1 is must to fill.

Account Update

At the beginning of file, we are checking for either user is logged in or not as this option is only available when user is logged in.
In this page, there is a form which will be already filled with previous given data corresponding to each member. User can change their name, roll no., branch, phone no., email id and update their information in the database. All the data will be transferred to process.php file where all the information will be saved in the database and user will get updated.

Tutorials Tab:

xyz.php:- This file retrieve all the tutorials from database “matrix_exp”
And list their titles. If login as admin(judge) then add tutorials features
Will be shown for adding new tutorials.

This part of code displays the list of tutorials.

```php
<?php
//-------------------------------------Display list of tutorials available-----------------------------------------------------------------------------
                $query = 'select * from tutorials';
                $result = DB::findAllFromQuery($query);
                $id = $row['id'];

               echo '<h1> Tutorials! </h1><hr/>';
               //echo "<form method = 'post' action = '" . SITE_URL . "/tut_show'>";
                foreach($result as $row){
                //echo "<li>" . "<option value='$row[gid]'".(($team['gid'] == $row['gid'])?("selected='selected'"):("")).">$row[title]</option>" . "</li

                //$strlink = "<a id = " . $row['id'] . " href = '" . SITE_URL . "/tut_show '>" . $row['title'] . "</a>";

                $strlink = "<a  class = 'add' id = " . $row['id'] .">" . $row['title'] . "</a>";

                //$strlink = "<li><a href = '" . SITE_URL . "/tut_show'>". $row['title'] ."</a></li>";
                echo "<li>" . $strlink . "</li>";
//-------------------------------------------------------------------------------------------------------------------------------------------------->
```

This code checks for Admin login for displaying the ADD tutorial feature!!..

```php
<?php
//-----------------------------------------------Check for Admoin login for displaying ADD tutorial features-----------------------------------------
if (isset($_SESSION['loggedin']) && $_SESSION['team']['status'] == 'Admin') { ?>
<hr/><h3>Add Tutorial</h3>
<form class="form-horizontal" role='form' method="post" action="add_tut">
        <div class="form-group">
            <label class='control-label col-lg-2' for='title'>Enter Title</label>
            <div class='col-lg-4'><input class='form-control' type="text" name="title" id="title" /></div>
        </div>
        <div class="form-group">
            <label class='control-label col-lg-2' for='content'>Content</label>
            <div class='col-lg-8'><textarea class='form-control' style="width: 550px; height: 400px;" name="content" id="content"></textarea></div>
        </div>
        <div class="form-group">
            <label class='control-label col-lg-2'></label>
            <div class='col-lg-4'><input type="submit" name="add" value="Add" class="btn btn-primary"/></div>
          <!----  <div class='col-lg-4'><input type="file" name="content1" id="content1" class="btn btn-primary"/></div>------>
        </div>
    </form>
            <?php

    }
?>
```

There is form in above code which is displayed, if loggedin as “Admin”.
On clicking Add button , submit event is performed which Post the title and the content of tutorial, to specified file.
Given below are the javascripts used for performing onclick event and to pass the variable “id” via Url.
In onclick() function we call post_to_url() function which pass the variable “id” to the defined path(ie. tut_show.php).
This id holds tutorial_id, we have just clicked.

```
<script type="text/javascript">
                    $('.add').on("click",function(event){
                        event.preventDefault();
                        var id = event.target.id;
                        //alert(id);
                        post_to_url('<?php echo SITE_URL."/tut_show"?>',{"id":id});
                    });
</script>
<script type="text/javascript">
       function post_to_url(path, params, method) {
       method = method || "post"; // Set method to post by default if not specified.
       // The rest of this code assumes you are not using a library.
       // It can be made less wordy if you use one.
       var form = document.createElement("form");
       form.setAttribute("method", method);
       form.setAttribute("action", path);
       for(var key in params) {
           if(params.hasOwnProperty(key)) {
               var hiddenField = document.createElement("input");
               hiddenField.setAttribute("type", "hidden");
               hiddenField.setAttribute("name", key);
               hiddenField.setAttribute("value", params[key]);
                  form.appendChild(hiddenField);
                                       }
                               }
                document.body.appendChild(form);
       form.submit();
       }
</script>
```

add_tut.php: File that receive the variables posted by form in xyz.php for adding new tutorial to the list.

Below code checks for title and content of tutorial ,is empty pr not.
If not then it is added to the tutorials list and message id displayed for conformation which redirects you to tutorial page containing list.

```
<?php
if($_POST['title']!=NULL && $_POST['content']!=NULL){
            $title = addslashes($_POST['title']);
            $content = addslashes($_POST['content']);

            $query = "insert into tutorials values ('','" . $title . "','" . $content . "')";
            DB::query($query);
?>
            <a href= <?php echo SITE_URL . "/xyz"; ?> > Tutorial added!...  </a>
<?php


            }
            else{
?>
            <a href= <?php echo SITE_URL . "/xyz"; ?> > Tutorial title or content is empty!... </a>
<?php } ?>
```

tut_show.php: this file display the content of the selected tutorial.
If logged in as admin then there is facilities to remove that

tutorial.
The id variable send by the tutorial list page is used for retrieving the contents of the tutorials.

· Included contact1.php: this file is used for posting user query and comments on tutorials contents. Which is further replied by the admin.

adminlog.php: This file enables the admin to check the logs of different team .Admin can watch the history of logs on the website.
Here is defined javascript for onclick event enable admin to watch the history of selected team . Tables are used to display retreived information from database tabel "logs".

Editor Module of Discussion Forums (qa-editor-basic.php)

function calc_quality($content, $format) --
Returns a numerical value indicating editor's compatibility with the supplied content ($content), as retrieved from matrix forum's database.If $format is ' ', then $content contains plain text in UTF-8 encoding. If $format is 'html', then $content contains HTML with UTF-8 encoding.Our basic text editor returns 1.0 for plain text and 0.2 for HTML.

function get_field(&$qa_content, $content, $format, $fieldname, $rows) --
Returns an HTML-based field for editor.The $fieldname parameter contains the HTML element name that we should use.The $rows parameter indicates a suggested height for your editor, in lines of text.

function read_post($fieldname) --
Should retrieve the content from the editor, as POSTed from the user's web browser, and convert it for storage in the forum's database. The $fieldname parameter matches the value that was previously passed to get_field(). To store plain text, the function returns array('format' => '', 'content' => '[text in UTF-8]') from this function.

View Submitted Solutions:-

viewsolution.php makes the team to see the submitted solutions. A team can't see the other's submissions during contest. For the practice problems a team can browse through the submissions, only if the access is made Public by the admin.
What different users can do here:-
I.Admin:- a. He/she can rejudge the solution.
b.He/She can disqualify the team. By doing so, the account of the team will get deleted.
c.He.She can update the access granted for that submission.
II.Normal User:-For a normal user, only the code submitted along with the expected output and the output generated is shown. But the admin can change these settings making sure what user is allowed to see.
III.Problem setter:-Problem setter can request the judge to rejudge the solution.
Code Explanation:-

- In lockdown condition, only the admin can log in.
- addslashes() puts the backslashes in front of the special characters like single quote, double quote, backslash ,NULL in the ID sent.
- Then it is checked that who is logged in.
- 'code' is the variable that stores the run id of the code to be viewed.
- SQL query ($query) is to check whether any submitted code is present with the given id 'code' or not.
- For the normal only those submissions can be viewed which are made public by the admin. He can browse his private submissions also.
- Admin don't have restriction.
- DB::findOneFromQuery($query) gives us the result in $res variable.
- If the $res is null, this means that no such solution with 'code' run id exists.
- The java scripts then are for the syntax highlighting.
- $query is for extracting the information about the problem for which the submission was made.
- $prob will store the results.
- Table will show the generated results.
- For normal users, we have to show him the code only.
- · Htmlspecialchars() — Convert special characters to HTML entities. The code submitted is in $res['code'].
- · The code will be shown.
- · $res['error'] is having the errors produced during the judgement of the code.
- ·preg_split() — Split string by a regular expression.
- · Then the generated and the correct output are shown(conditions are applied).
- ·Now, if the user is admin, we were also to show an extra-privileged table from where he can disqualify, rejudge or change the access of the solution.
- ·Table headings are Rejudge, Disqualify and Access.

- ·Press of any of the button will be handled by process.php.

Search_button() in qa-theme-base:-

Three radio buttons and one submit button is present in forums section for searching information.

· Both the radio buttons run a java script function i.e myfunc(sent_id) .

· On clicking any of the radio buttons, the set_search_for_what (SESSION variable) will be sent a value accordingly using ajax.

· The file opened during ajax is set_search_for_what.php

· SESSION variable search_for_what is set accordingly on the click of the radio button.

· On the click of the search button, when db-selects.php is run, the search_for_what variable is ready with us. Now we do the search accordingly.

testcase.php:-Only for the problem setters or the admin. He/she can edit the test cases. New test cases can be added.

Explanation:-

· $_POST['pid'] is having the problem id for which we have to open the testcase.php

· ($owner_query="select tid from problems where pid=".$pid) is the query which will verify the problem setter for that problem.

· ($owner_query_result=DB::findOneFromQuery($owner_query);) will tell us the results.

· Then the check will be made.

· ($query="select name from problems where pid=".$pid;) query for problem name.

· ($problem=DB::findOneFromQuery($query);) will give us the information about the problem.

· ($query_test="select id,input,output from testcase where probid=".$pid;) query for extracting the input and output information about the problem.

· ($test_cases=DB::findAllFromQuery($query_test);) will give us the result.

· Results will be shown in the table.

· Using delete button, the test case can be deleted as well.

· Then the test cases can also be added.

· This dynamic addition is handled by j queries the run onclick functions.

adminteam.php:-

This file contains the information of all the teams which are registered. This file can only be accessed by admin. First it will display all the registered team name. Admin can update settings of any user.

Array $res is used for storing all the team whose data is to be updated.

Array $result is used for storing information of each row of the group.

Functions used:-

1. findOneFromQuery($query)

   This function return the first row that matched with query.
2. findAllFromQuery($query)

   This function return all the rows that matched with query.

adminproblem.php:-

This file can be accessed by admin as well as problem setter. Admin or Problem setter can add or update practice problems or contest problems.

Array $res is used to store the information of questions.

Functions used:-

1. findOneFromQuery($query)

   This function return the first row that matched with query.
2. findAllFromQuery($query)

   This function return all the rows that matched with query.

stats.php:-

My stats is feature for any user which allows him/her to see his/her total progress so far. Also user can see number of problems solved in particular language.

Array $res store the information of problem solved/tried by user.

process.php:-

This is one of the most important files as it many useful functions implemented.

Functions:-

1. customhash($str):-

   This function takes password as input and convert to hash for security purposes.
2. Login:-

   This function checks if the user is registered or not.

   $_POST is array which stores all the login data user has entered. Then this data is validated against the data in database. If some data matched with database then user is directed according to its status (if user status is admin then page will directed to admin

page etc.). If user status is ‘pending’ or ‘suspended’ then user will be prevented from login.

3. Logout:-
   If user chooses to logout the session is destroyed.
4. Display solutions:-
   For admin this function will show all the solution from database.
   For normal user first his/her team id will be checked against database and only those solutions submitted by him/her will be printed.
5. Submit solution:-
   User can submit solutions for practice problems and contest problems.
   First it will check that user is login.
   Practice problem: When user submit solution its solution with team id and result will be stored in database. Result is based on judge evaluation.
   Contest problem: First end time of contest is checked. If it is greater than current time then solution will be evaluated as practice problem. If end time of contest is less than current time then solution will not be evaluated.
6. Register:-
   There are two options for user to register. User can register individually or in team of maximum 3. If all the data entered by user is according to rules then data will be stored in database and user can login with credentials entered. Array $_SESSION keep track the activities by user.
   7. Account Update:-
   This function checks whether someone is logged in or not and then it decides whether the password entered and re-entered by the candidate are same or not.

   8.Clarifications:-
   this function checks whether $_post[‘clar’] is set or not.
   then this function inserts the values into the clar datatable where the values are stored in $_post.

   9.Add problem:-
   this function ensures that the logged in candidate is an admin or a problem setter and then inserts a row in the datatable by taking the values from the $_post. The functions used are addslashes() which adds slashes for some special characters.

   10.Update problem:-
   this function helps to insert a row in the database by taking values from $_post which comes from the adminproblem.php. This function is used to modify a problem.

Edit.php:

This file creates the account settings page. In this page a user can update his password. It contains a form with 3 text boxes namely, Old Password, New Password, Re-Password and a drop-down menu for group name and an update button.

In the code below fetches the group names from the database and shows them in the drop-down menu to be selected by the user.

```
<select name="group" class='form-control'>
    <?php
        $query = 'select * from groups';
        $result = DB::findAllFromQuery($query);
        foreach($result as $row){
            echo "<option value='$row[gid]'".(($team['gid'] == $row['gid'])?("selected='selected'"):("")).">$row[groupname]</option>";
        }
    ?>
</select>
</div>
```

The action of the form is defined in “process.php”. If the user is not logged in it redirects the user to the home page and shows the message “You are not logged in”.

submit.php

If not in lockdown mode and not in guest mode then include the mentioned .js files through script.
If not logged in then on clicking the submit button "You are not logged in!" is shown.
$_GET['code'] is providing the problem whose code we are submitting.
Through the following line we are using POST method for the submitted code and the submit button action is directed to the process.php page. **<form id='form' action='<?php echo SITE_URL; ?>/process.php' method='post' enctype='multipart/form-data'> .**
$prob = DB::findOneFromQuery($query) is set when the query is properly executed. $lang contains the languages for the problem.The items in the dropdown are according to the languages in $lang. Once the language is selected, set the editor(codemirror) mode as selected from the dropdown. Rest of the code is to set up the various elements of the page accordingly.

submissions.php

This php page is for the submissions tab of the matrix defining all the actions specifically for admin, guest and logged in user.

If it is Admin-
When the admin clicks the button "CODE" aside to a particular run-id then with the help of the following form:
<form method='post' action='" . SITE_URL . "/process.php'>
<input type='hidden' name='tid' value='$tid' />
<input type='submit' name='rejudge' class='btn btn-danger' value='Rejudge All Selected Submisssions'/> </form> the admin can rejudge the submitted code.

$result = DB::findAllWithCount($select, $query, $page, 25) sets up the table that appears in the submissions page. Here -
$query = fetches everything from runs table which is not "deleted".
25 = is the limit on the number of rows in a page.

The same is also done for guest users except that he/she does'nt have the privileges of admin like rejudge, etc. In case of guest user the submission page does not have "CODE" button under the options column.

Single-sign on facility for Judge and forums:-

This facility makes sure that the there is a single log-in and register point for both the Judge and the forums,
This feature has been implemented by taking the union of the mysql databases of both the judge and the forums and using Session Services.
When a user logs into the matrix judge, The session value for loggedin is set to true i.e
$_SESSION['loggedin'] = "true";
The Session Service performs the following actions:-
1. Generates session identifiers.
2. Maintains a master copy of session state information.
3. Implements session life cycle events such as logout and session destruction.
4. Implements session failover.

//**qa-external-users.php**
This file is the key to creating a single sign on facility for the judge and the forum.

The functions involved in this page are:-

(i). **qa_get_mysql_user_column_type()** :- This function returns the type of the teamname column in teams table of our
database, which is set to VARCHAR(16) in our case.

(ii). **qa_get_login_links($relative_url_prefix, $redirect_back_to_url)** :- This function returns an array of links, containing the links to login, logout and register page of the matrix judge.

(iii). **qa_get_logged_in_user()** :- This the most important function of the entire page which decides which page will be served on the discussion forum, It checks if the SESSION variable for the 'logged_in' value of the judge is set to true or false, If its false then simply the index page of forum is served with no user logged in, Else if its true then we first check for the status of the logged in user(i.e we check if the logged in user is having admin privilegs or not) and then the session is started in the forum corresponding to the specified user.

(iv). **qa_get_user_email($userid)**:- This is a trivial function which retuns the email-id of the logged in user or null if no one is logged in, It requires only one query in the teams table of the database.

(v). **qa_get_userids_from_public($publicusernames)**:- Given the username this funciton returns the corresponding userid. The required query is
$query = 'SELECT teamname, tid FROM teams WHERE teamname IN ('.implode(',', $escapedusernames).')';

(vi). **qa_get_public_from_userids($userids)**:- Given the userid this function returns the username, the query required is:- $query = 'SELECT teamname, tid FROM teams WHERE tid IN ('.implode(',', $escapeduserids).')';

(vii). **qa_user_report_action($userid, $action)**:- Informs you about an action by user $userid that modified the database, such as posting, voting, etc... If you wish, you may use this to log user activity or monitor for abuse.

---

contestranking.php:-

Contestranking can be accessed by all the users when judge is not in lockdown mode else admin only can access it. This file allows user to see current ranking status of active contest. Ranking is based on number of problem solved and time taken to solve the problem.
Array $_GET used to store information of active contest.

Contest.php:-

Contest can be accessed by all the users when judge is not in lockdown mode else admin only can access it. New contest is added by admin while problems can be added by both admin and problem setter.
Two main functions used in this file:

1. **zeroPad(num, places)** = It returns the time in H-M-S(hour-minute-second) format
2. **Timer()** = If contest is not started yet then displays the remaining time for the contest to start.
   If contest has just started then display alarming message the contest has been started and reload page to see problems.
   If the contest has been started then it will show the remaining time for contest to end.

adminjudge.php :

CSS is renedered using bootstrap library in this form.

This file contains the interface for interacting with the judge settings. The penalty value.

Only admin can acess the data of this file.

General settings:

This data is taken by a form 'genform' . The value of penalty,endtime is validated using javascript.

Socket settings:

The second form is 'socform'. This takes the value of IP and port of judge.

Notice

The third form is for notices . The data inputed in this form is not validated.

Android App:

Made App using ‘webview’ toolkit provided by android libraries. I chose ‘webview’ since it is easy to use and implement. There was another way i.e connecting to website and using JSON to parse the data. In both ways we were connecting to the website for data so I felt webview was good for our project. Our team was also very good in implementing the ‘css’ part very

diligently and made a mobile compatible website which made the webview easier to implement. I had to make very few changes in the ‘html’ and ‘php’ part to check compatibility for android devices.

The app is basically a browser(customized for our project) in which we are loading the url of the site.

```
class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
    }
    public void onReceivedError(WebView view, int errorCode, String description, String failingUrl) {
        view.loadUrl("file:///android_asset/error.html");

    }
```

Above code is the custom webclient for our android device.The error.htm loads if the setting are incorrect.The below code is about the customized settings for this app. Javascript is disabled by default. So had to re-enable it.

```
    }
    view.setWebViewClient(new MyWebViewClient());
    String url = "http://10.10.2.32/matrix/";
    view.getSettings().setJavaScriptEnabled(true);
    view.getSettings().setSupportMultipleWindows(true);
    view.getSettings().setAppCacheEnabled(true);
    view.getSettings().setPluginsEnabled(true);
    view.getSettings().setAllowFileAccess(true);
    view.getSettings().setPluginsEnabled(true);
    view.loadUrl(url);
```

Back button by default closes the app in webview so had to change the function of backbutton to go to the previous screen.

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    // Check if the key event was the Back button and if there's history
    if ((keyCode == KeyEvent.KEYCODE_BACK) && view.canGoBack() ) {
        view.goBack();
        return true;
    }
    // If it wasn't the Back key or there's no web page history, bubble up to the default
    // system behavior (probably exit the activity)
    return super.onKeyDown(keyCode, event);
}
```

Some changes in index.html have been done to make the website android compatible.

```
<!--Checking for android -->
<?php
        function mobile_user_agent_switch(){

            if( stristr(strtolower($_SERVER['HTTP_USER_AGENT']),'android') ) {
                $device = "android";
            }
            if( $device ) {
                return true;
            }
            return false;

        }
        $xyz=mobile_user_agent_switch();
?>

<!--check complete -->
```

Forgot password:

For implementing forgot password, an external server was needed to send the verification code to email id. For implementing forgot password new_password.php, ver_code.php, change_password.php have been to website and mail.php was added to external server.

When user clicks on 'Forgot Password' link he will be redirected to a page where he has to give his email. then a verification code will be sent to his mail. Then user has to enter the verification code and change his password.

The external code in the server is as below:

```
$to = $_REQUEST['webmail'];
$subject = 'Verification Code';
$from = 'Programming';
$message =$_REQUEST['post'];
$newURL = $_SESSION['url']."/change_password";
# Attempt to send email
if(mail($to, $subject, $message, "From: $from"))
echo "Mail sent";
header('Location: '.$newURL);
```

The below code contains two forms for getting a verification code, and then changing password with the verification code mailed to our email.

```
<div class="col-md-4" style ="padding:20px">
    <div class="input-group" style="margin-bottom: -1px;">
    <form class="diff_forms" action="ver_code" method="POST">
    <p>GET VERIFICATION CODE NOW</p><br>
    <input class="form-control" style="border-bottom-right-radius: 0;" type="text" placeholder="email" name="get_webmail" /><br>
    <input id="signup" type="submit" value="GET IT NOW" class="btn btn-primary btn-block"/>
    </form>
    </div>
</div>
<div class="col-md-2">
</div>
<div class="col-md-4" style ="padding:20px">


    <div class="input-group" style="margin-bottom: -1px;">
    <form class="diff_forms" action="new_password" method="POST">
        <p>GOT YOUR VERIFICATION CODE?</p><br>
                <input class="form-control" style="border-bottom-right-radius: 0;"  placeholder="email" name="webmail_id" type="text" /><br>
    <input class="form-control" style="border-bottom-right-radius: 0;"  placeholder="Verification Code" name="ver_code" type="text" /><br>
    <input class="form-control" style="border-bottom-right-radius: 0;"   placeholder="New Password" name="new_pass" type="password"/><br>
    <input class="form-control" style="border-bottom-right-radius: 0;"  placeholder="Re-Enter New Password"name="new_pass_again" type="password" /><br><br>
    <input id="signup" type="submit" value="CHANGE PASSWORD" class="btn btn-primary btn-block"/>

    </form>

    </div>
</div>
```