Competitive programming website - CodeArea CSP[203]: Software System Laboratory

March 8, 2018

Authors	email id
Girish Kumar	2016csb1040@iitrpr.ac.in
Karan Seghal	2016csb1080@iitrpr.ac.in
Chirag Khurana	2016csb1037@iitrpr.ac.in
Arunaksha Talukdar	2016csb1032@iitrpr.ac.in

1 Overview

The main objective of this site is to provide students a competitive programming platform so students can compete with each other and can improve the competitive programming environment in the college. Through this website teachers will be able to create lab exams on competitive programming e.g. Data Structure. The primary objective of the site is to reduce plagiarism from the internet while using other sites like Hackerrank, Codechef, to access these sites one need access to internet which lead to plagiarism. But the CodeArea site can be hosted on our servers, which can be accessed on the intranet.

1.1 Technical Details

We are using Postgres has our backend with Django, a python based web-framework. For frontend, we are using HTML5, CSS3 and few javascript libraries. The judge is also written in python.

2 Judge

2.1 Introduction

This is the important part of the CodeArea (competitive programming site name). Main work of the judge is to execute user's codes and give them results of the execution e.g output for corresponding input or errors, compile time or run time errors. The judge take care of security and checks if the user program is not going to read or write other files on the server and executing other programs on the server.

2.2 Supported languages

In initial phase of the CodeArea we are supporting following programming languages:

1. C
2. C++14
3. C++11
4. JAVA 8
5. Python 3.5
6. Python 2.6

2.3 Facilities

2.3.1 Custom input-output

To test the code we are providing custom input-output facility so user can test there code for corresponding input and output. Custom input-output will be implemented in the web-page using iframe.

2.3.2 Code Editor

To make writing code easier, we are providing a code editor which can be implemented in the web-page using iframe. The editor provides basic functionalities of syntax highlighting, smart indenting, bracket completion, editor themes, fonts(small, medium, large). All of these functionalities are powered by ACE, an open source library for code editor.

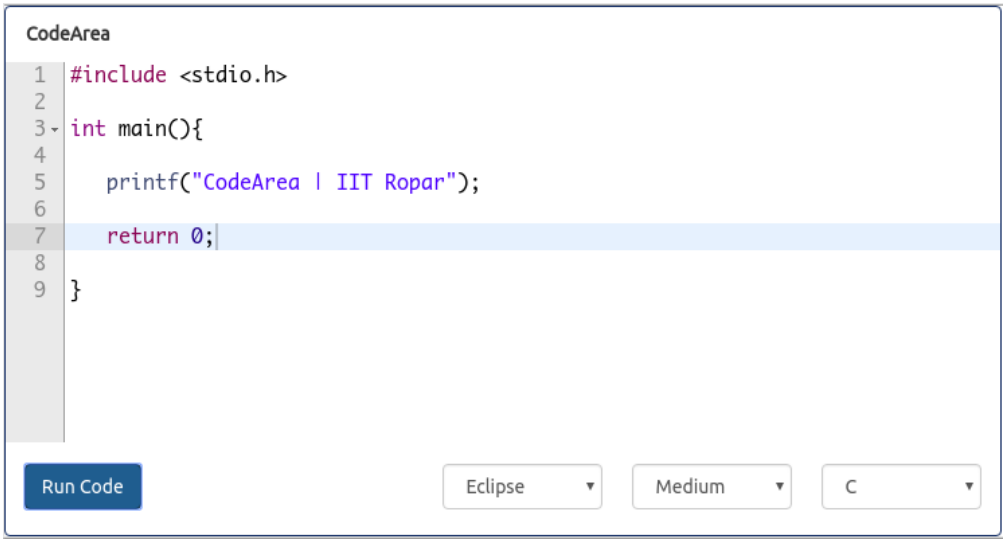

Figure 1: ACE code editor

2.4 Submission

On submission the language and the code is send to a python script which save the code to a temporary file with proper extension and we run the code on the server using os.system(command) and the program execution is killed if it takes more than the specified time. The output of the stdout and stderr is passed to a file and the content of the file is given to the user. Which contain errors in case of compile-time or run-time errors or output of program in case of successful completion of user program. In case of submission against some test cases the content of the file is compared with right output for the problem.

3 Models

3.1 Introduction

A model is the single, definitive source of information about one's data. It contains the essential fields and behaviors of the data one is storing. Generally, each model maps to a single database table. The Models module manages the data, logic and rules of the database.

3.2 Schema

Apart from the User Model, mentioned in the Social Auth Module, the major entities of the application are, Problem, Contest, Participant, Submission, Tag and Post.

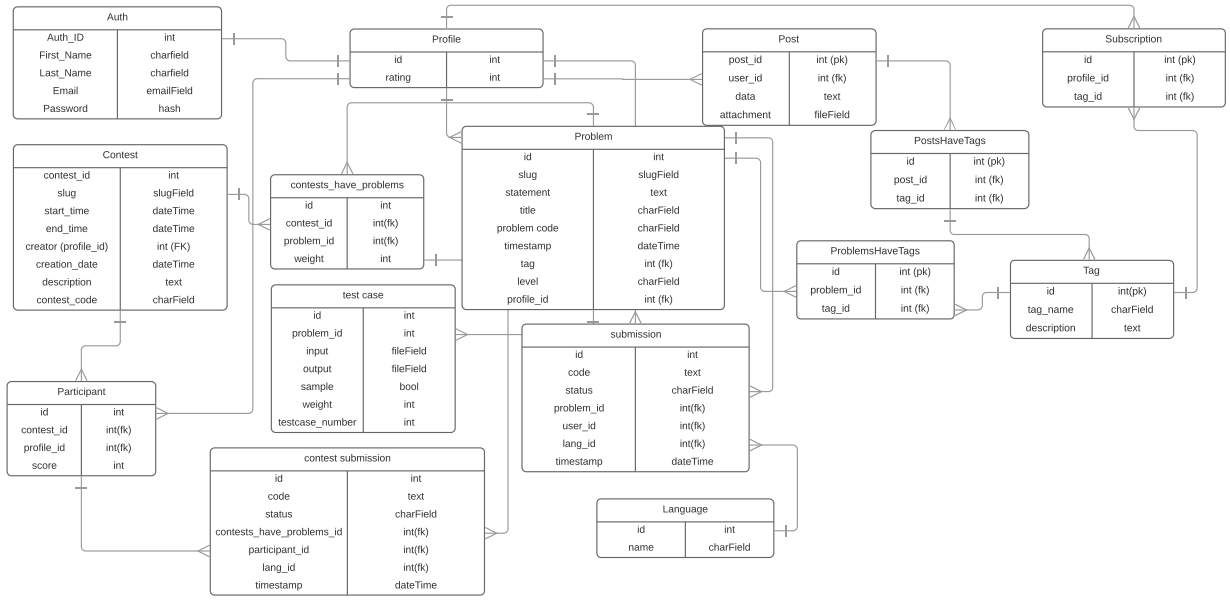

Figure 2: ER Diagram

3.2.1 Problem

Consisting of a title, problem statement, level of difficulty, problem setter and test cases, the problem model represents a problem in the application. The model has a one-many relationship with Test Case, a many-one relationship with User and a many-many relationship with Contest. The model also contains problem code, which is a unique code given to the problem, as well as a slug field which is derived from the problem code. A slug field is used to store and generate valid URLs for dynamically created web pages.

A *pre_save signal*, a signal (set of instructions) sent before the instance of a model is saved in the database, is used to generate a slug from the problem code.

3.2.2 Test Case

Consisting of an input file, an output file, test case number and weight, the Test Case Model represents a test case of a problem. The test case number attribute stores the value of the test case and the weight attribute representing the weight that specific test case has in the problem. There is a boolean attribute sample, which checks whether the test case is a sample test case.

The input and output files are stored in a folder named with the unique problem code, having names <test case number>.in and <test case number>.out, respectively.

3.2.3 Contest

Consisting of a title, description, starting time, ending time, and contest creator, the contest model represents a contest in the application. The model has a many-one relationship with User (creator of the contest), a many-many relationship with Problem and a many-many relationship with User (participants of the contest). The model also contains contest code, which is a unique code given to the problem, as well as a slug field, similar to the one in the Problem Model, which is derived from the problem code.

3.2.4 Contest Problem

The many-many relationship between the Problem and Contest model is through the model Contest Problem, which represents a problem in the contest. Apart from foreign keys from both the models, it has a weight attribute representing the weight that specific problem has in the Contest.

3.2.5 Participant

The many-many relationship between the User and Contest model is through the model Participant, which represents a participant in a contest. Apart from foreign keys from both the models, it has a score attribute representing the total score of the user in the contest.

3.2.6 Submission

Consisting of a submitter, code, language, problem and status, the submission model represents a user submission. A user submission is sent to the Judge Module to get back the status of the submission. The current languages supported and status options are mentioned in the judge module.

3.3 Future Plans

Outside the contest environment, the application aims to extend its functionality by introducing two new models, Post and Tag, if time permits.

3.3.1 Post

Consisting of a title, description, optional attachment and creator, the post model represents a post. A post might be an article about some algorithm or some latest updates in the world of computer science. Various posts can be seen by a user on their homepage.

3.3.2 Tag

A tag is helpful in filtering out various posts and problems for a user. It has a many-many relationship with Post and Problem. Users can subscribe to specific tags to get updates on their feed.

4 Front-end of Code-Area

The aim of the module is to develop the Frontend User Interaction module of Code-Area.

4.1 Schema

The website consist of hierarchical structure of linking of webpages as follow: homepage , login page + about or Website Description , contestlist , contesthost , contest page , problem page , Rankboard.

4.1.1 Homepage

- Login region: Which verifies the user's identification and grant's further functionalities of the system.
- About region: Describes a brief intro of the platform ,functionalities and suppotable code Formats.
- Loginbox: Consist of userID, password entry box and Submit button handled by user authentication Module.

4.1.2 Contestlist

The ongoing contest at the moment or about to happen (With countdown time)

4.1.3 Contesthost

Raising a problem and Describing about it with the following functionalities supportable:

- Problem name
- Problem Description
- Scores Description
- Note box : regarding time limit , input limit
- Save button: To Confirm and proceed.
- Sample input , expected output with description

4.1.4 Contestpage

The main Contest page includes : Countdown timer till completion , Complete Problem Description ,Test cases with expected output.

4.1.5 Problem page

- Short Problem descrition with Sample input ,expected output dialog box
- Code input Submission Frame output managed by JUDGE module. With a management of Code sumbimtted and rank in rankboard by DATABASE module.
- Submit box: Link to Rankboard , and final submission of code.

4.1.6 Rankboard

- Shows rank of Coders (On basis of JUDGE module)
- Resubmit Button: Option to revisit the problem and Modify it and submit without altering the previous SubmitNo limit to Submissions .

5 Forms

The Forms Module manages the process of accepting input from the user, and then processing and responding to the input. Major functions performed by the module are as follows:

- Preparing and restructuring data to make it ready for rendering
- Creating HTML forms for the data
- Receiving and processing submitted forms and data from the client
- Perform validation checks before saving the data to the database

6 Views

A view takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. The view itself contains whatever arbitrary logic is necessary to return that response. The View Module manages all the different views for different pages in the application, all views being their own submodules. All the CRUD operations operate through views.

7 User Permissions

Closely related to the Views Module, the User Permissions module manages all the user permissions of the app. User permissions include what all pages a user can view, what all files on the server can the user have access to, etc.

8 Social Authentication

8.1 Introduction

Social login is a form of single sign-on using existing information from a social networking service such as Facebook, Twitter or Google+, to sign into a third party website instead of creating a new login account specifically for that website. Our mode of authentication is using the Google login.

8.2 Features

- As almost every coder has Google account, he/she can easily sign in with Google without any signup.
- If we have to conduct a competition within the institute, we can put the filter which will allow only sign in with institute's email-id.
- It saves database memory for storing credentials of the users.
- No need to send confirmation mail, and the email would be validated.

8.3 Implementation

We are using Django Social Auth library to implement this, it is an easy way to setup social authentication/authorization mechanism for Django projects. It provides user login using social websites credentials. It populates data from the social websites and creates new users.

We have a one-one relationship between a user and profile, extra application dependent information is stored in the profile model.

9 Rich Text Integration

9.1 Introduction

This module aims to integrate a rich text editor with the text field of a form, especially for problem statements and posts. This module can be integrated with either a markdown editor or a WYSIWYG (what you see is what you get) editor. Websites like hackerrank or stackoverflow use a markdown editor, since it is more customizable for people familiar with markdown.

9.1.1 Markdown Editor

Markdown is a lightweight markup language with plain text formatting syntax. It can be converted to HTML and many other formats. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

9.1.2 WYSIWYG Editor

WYSIWYG implies a user interface that allows the user to view something very similar to the end result while the document is being created. In general, WYSIWYG implies the ability to directly manipulate the layout of a document without having to type or remember names of layout commands.

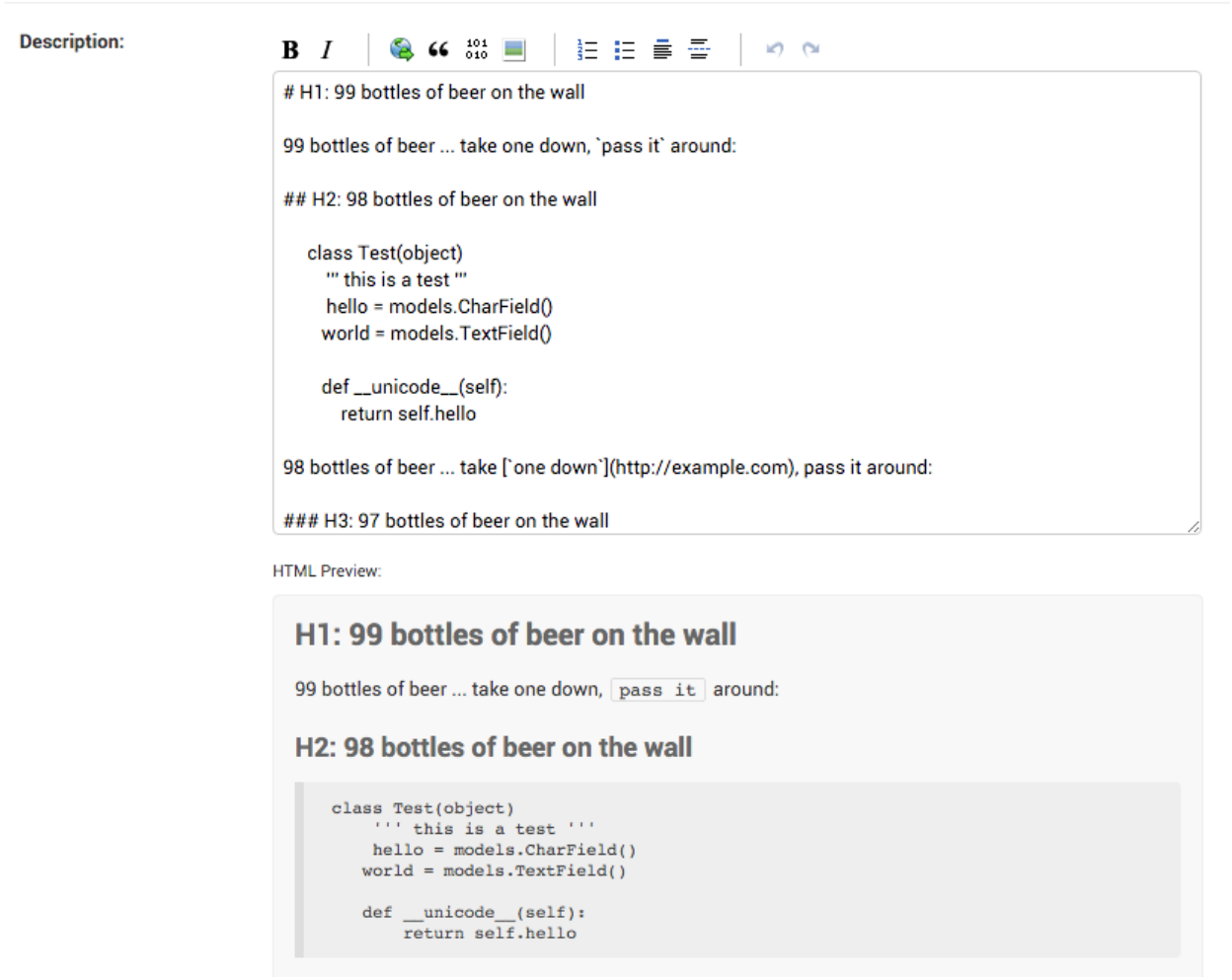

Figure 3: Example of Markdown Editor

9.2 Use

The major use for this module to increase readability and writability in text fields like problem statement, contest description, post content, etc.

9.3 Implementation

There are several libraries compatible with Django for markdown editing (pagedown) as well as WYSIWYG (tinymce). This libraries allow addition of rich text editors in form fields.