

CS349 - Database and Information Systems Lab

CodeForces Revamped

Ananya Rao (22B0980)
Deeksha Dhiwakar (22B0988)
Sharvanee Sonawane (22B0943)
Sanchana Arun (22B1034)

April 2024

1 Project Idea

The idea for our project is to build a clone of the popular website [CodeForces](#). It is a competitive programming platform that hosts regular contests and provides a vast archive of algorithmic problems for users to practice and enhance their coding skills. It features a rating system that categorizes participants from Newbie to Legendary Grandmaster, fostering a competitive learning environment. Users can engage in virtual contests, solve problems with editorial support, and participate in community discussions through blogs. It also offers a "Gym" for training with custom problem sets, making it an essential resource for aspiring competitive programmers.

We intend to replicate the essential functionalities of CodeForces, and then extend it to support several novel features such as an inbuilt plagiarism checker, an option to generate model solutions using AI when creating a problemset, a version control system that enables users to access previous iterations of their code (similar to the timeline feature of VS Code), isolated sandbox environments for a better coding experience, an inbuilt coding IDE featuring intellisense and autocomplete for multiple languages and many more.

We also plan to revamp the user interface by improving the design and layout and making it more intuitive and user friendly.

2 Progress Made

ER Diagram

We have formulated the below ER diagram that our database schema will follow (implemented using PostgreSQL).

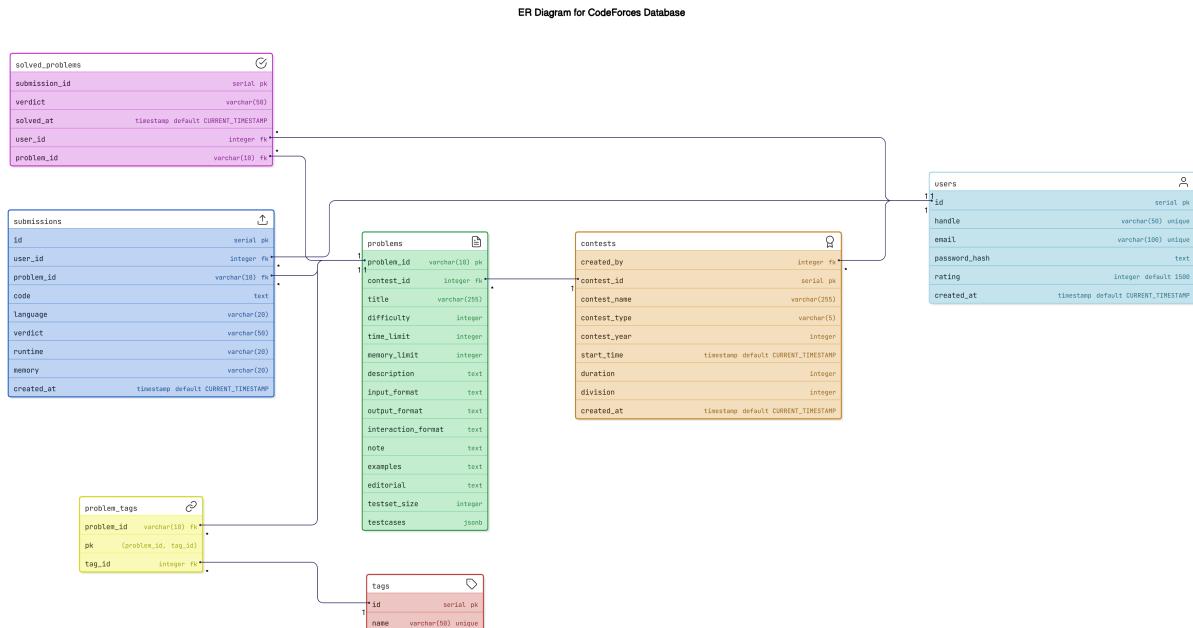


Figure 1: ER Diagram

Dataset

In this project, for problems and contests we use the publicly available dataset from [this link](#). It contains 10024 rows of data, where each row has the details of a CodeForces problem such as the problem ID, contest ID, difficulty, tags, description, testcases etc.

To populate the `users` table, we generate fake user data (including email ID, username, location, joining date etc) using a Python script.

For submission data, we use the publicly available dataset from [this link](#). It has been generated based on actual user submissions till the end of 2024, using aliases for all the users to protect their anonymity. It contains several details of the submissions such as the code, verdict, runtime, programming language etc.

Signup/Login/Logout

The pages in focus have been implemented. We aim to add *Google authenticator* as well to facilitate the signup/login process for users who already have an email id registered with *Google*

The image contains two side-by-side screenshots of web forms. The left screenshot, labeled (a) Sign-up, shows a 'Create your account' form with fields for a handle ('sharvanee8') which is noted as 'Handle is available!', an email ('s@s'), and two password fields. A blue 'Sign Up' button is at the bottom. Below it is a link 'Already have an account? Login here'. The right screenshot, labeled (b) Login, shows a 'Sign In' form with fields for an email ('s@s') and a password. It includes a 'Remember me for a month' checkbox, a 'Forgot Password?' link, and a large blue 'Login' button. Below the button is a link 'Don't have an account? Sign up here'.

Once the user has registered (or logged in), they are redirected to the dashboard page

Dashboard

Following is an image of the dashboard. It has a Navigation bar at the top which has buttons for important pages like **Contests**, **Problem-Set**, **Profile** etc.

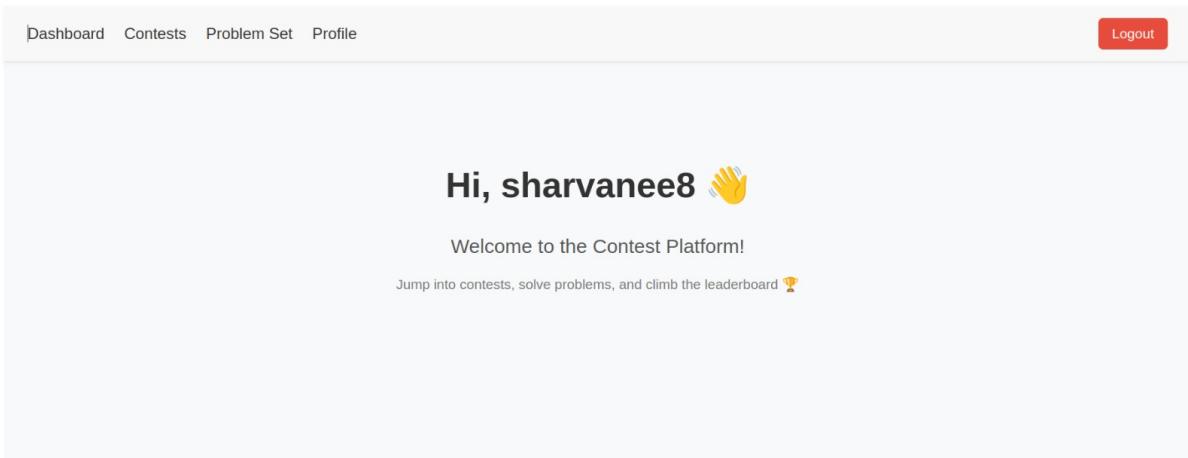


Figure 3: Dashboard

Contests

Here we show the **Contests** page which has a list of all active and inactive contests. The name of the contests are hyperlinks to the contest page. The contest list shows important details like start-time, and duration of individual contests

Contests

Title	Start Time
Bubble Cup X - Finals [Online Mirror]	03/09/2017, 15:30:00
Codeforces Round 609 (Div. 1)	21/12/2019, 16:35:00
Codeforces Round 421 (Div. 1)	27/06/2017, 20:05:00
Codeforces Round 265 (Div. 2)	07/09/2014, 21:00:00
Kotlin Heroes: Practice 4	22/05/2020, 19:05:00
Technocup 2020 - Elimination Round 1	06/10/2019, 20:35:00
Codeforces Round 205 (Div. 2)	12/01/2015,

Figure 4: Contest List

Following is the page for a contest. It shows the list of all the problems which are a part of the contest and has hyperlinks to all of them.

Future Modifications: We aim to make different interfaces for active and inactive contests with active contest page being more interactive, providing live score-tables, rankings with other relevant features. Similarly, in inactive contests, we aim to provide details about the contest, problems' difficulties and hints to solve them.

Start Time: 03/09/2017, 15:30:00

Problems in this Contest

Digits	Difficulty: 2500
Neural Network country	Difficulty: 2000
Property	Difficulty: 2100
Exploration plan	Difficulty: 2100
Casinos and travel	Difficulty: 2100
Product transformation	Difficulty: 2200
Bathroom terminal	Difficulty: 1700
Rob and stages	Difficulty: 2000

Figure 5: Contest Page

Problems

The Problem-Set page displays a list of all problems ever inserted in the database. It especially displays the problem title, difficulty and tags for coders to choose a question of their choice and need.

ID	Title	Difficulty	Tags
1000A	Codehorses T-shirts	1200	implementation greedy
1000B	Light It Up	1500	greedy
1000C	Covered Points Count	1700	implementation sortings data structures
1000D	Yet Another Problem On a Subsequence	1900	dp combinatorics
1000E	We Need More Bosses	2100	dfs and similar graphs trees
1000F	One Occurrence	2400	data structures divide and conquer
1000G	Two-Paths	2700	dp trees data structures
1003A	Dolarm's Pockets	800	implementation

Figure 6: Problem-Set

The hyperlink to any Problem takes us to the Problem page, which display details like problem description, input/output format, tags, extra notes and any relevant link or information required to help users.

Future Modifications: We aim to make two different pages to show inactive problems and active problems (those which are included in active contests); each with slightly different but relevant characteristics.

Digits
Difficulty: 2500
Tags: brute force, implementation, math

John gave Jack a very hard problem. He wrote a very big positive integer A0 on a piece of paper. The number is less than 10200000. In each step, Jack is allowed to put '+' signs in between some of the digits (maybe none) of the current number and calculate the sum of the expression. He can perform the same procedure on that sum and so on. The resulting sums can be labeled respectively by A1, A2 etc. His task is to get to a single digit number.

The problem is that there is not much blank space on the paper. There are only three lines of space, so he can't perform more than three steps. Since he wants to fill up the paper completely, he will perform exactly three steps.

Jack must not add leading zeros to intermediate results, but he can put '+' signs in front of digit 0. For example, if the current number is 1000100, 10 + 001 + 00 is a valid step, resulting in number 11.

[Submit Solution](#)

Submissions

Submission ID	Status	Time	Memory	Language
---------------	--------	------	--------	----------

Figure 7: Problem Page

We have also added a submission feature in each problems page, for users to submit their solution

through either a valid file or directly through an editor, and a list of submissions for easy tracking of submissions.

(a) Submit code

(b) Submit file

```

Submission Details

Problem:
Language: python
Verdict:
Submitted: 09/04/2025, 23:25:21
Code:

from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, lit, least, min as spark_min
import sys

def parse_line(line):
    parts = line.strip().split(":")
    if len(parts) < 2:
        return []
    vi = int(parts[0].strip())
    neighbors = sorted(set(map(int, parts[1].strip().split())))
    edges = []
    candidates = []
    for i in range(len(neighbors)):
        for j in range(i + 1, len(neighbors)):
            vj, vk = neighbors[i], neighbors[j]
            candidates.append(((vj, vk), ('C', vi)))
    edges.append(((vi, neighbors[i]), ('E')))
    return edges + candidates

def process_triangles(pair, values):
    ...

```

Figure 9: Submission details

Future Modifications: We plan to implement Sandboxing to enable a safer and more convenient environment for users to run their programs. We also aim to add built-in compilers in the Sandbox container for direct compilation and testing.

Profile

The profile page shows personal information (handle, rating, problems solved etc.) of a user along with a hyper link for adding contests.

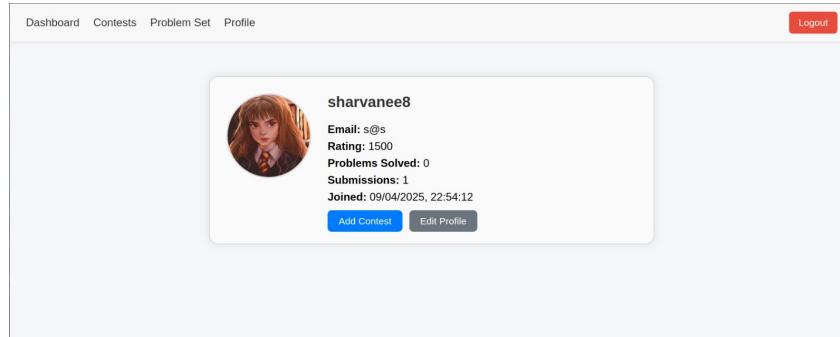


Figure 10: Profile

Future Modifications: We aim to add activity calendar and detailed performance report for users to track their progress and activities on our platform. We also plan to make the rating system more

dynamic and use it to grant various privileges to users (like adding contests, making problems and organizing hackathons)

2.1 Add-Contest

This page allows a user to organize a contest on our platform. The number and description of problems depend on the creator and there is a dynamic support for adding as many problems as required.

Figure 11: Add-Contest

3 Plan for Final Submission

The remaining tasks to be completed by our final submission are listed out below:

- Inbuilt plagiarism and AI code detector to automatically flag unethical submissions and restrict users with multiple offences.

Username	Submission ID	Problem	Suspicion Type	Confidence	Actions
alice123	5271	Add Two Numbers	Plagiarism	95%	<button>Warn</button> <button>Ban</button>
coder_pro	5270	Binary Search	AI-generated	87%	<button>Warn</button> <button>Ban</button>
alice123	5267	Add Two Numbers	Plagiarism	85%	<button>Warn</button> <button>View</button>
dev_john	5265	Palindrome Check	AI-generated	92%	<button>Warn</button> <button>Ban</button>

Figure 12: Plagiarism Detector

- Improved inbuilt coding IDE that supports features inspired from VS Code such as autocomplete and intellisense.

A screenshot of a code editor interface. At the top, there are tabs for 'SOLUTION', 'Description', 'Submissions', 'Tests', and a blue 'Run' button. Below the tabs, a file named 'solution.py' is open. In the code editor area, the following Python code is visible:

```
1 import sys
2 import solve
3
4 def two_sum(nums, target):
5     for i in range(0, lenums)):
```

The cursor is positioned at the end of the line '5 for i in range(0, lenums)):', and a dropdown menu is displayed, listing several Python functions: 'range', 'range(start, stop[t, step])', 'rapartition', 'ray', 'radians', and 'raise'. The word 'range' is highlighted in the dropdown.

Figure 13: Autocomplete feature

- Version control feature that allows users to access previous iterations of their code (inspired from VS Code timeline feature).

A screenshot of a code editor interface showing a 'TIMELINE' panel. The timeline shows the following history of changes:

- Saved saved
- 2 minutes ago (highlighted in blue)
- 7 minutes ago
- 12 minutes ago
- 20 minutes ago
- 34 minutes ago
- 45 minutes ago

To the right of the timeline, a code editor window is open with the file 'TS delimiter.ts'. The code is as follows:

```
1 function isDelimiter(char: string): boolean {
2     return
3         char === ' ' || char === '\n' || char =
4             't' || char === ';' || char === ';;'
5     }
6 }
```

Figure 14: Version control

- Option to use an AI model to generate model solutions and testcases when creating problems.
- Google authenticator feature for two-factor-authentication, added support for passkeys.

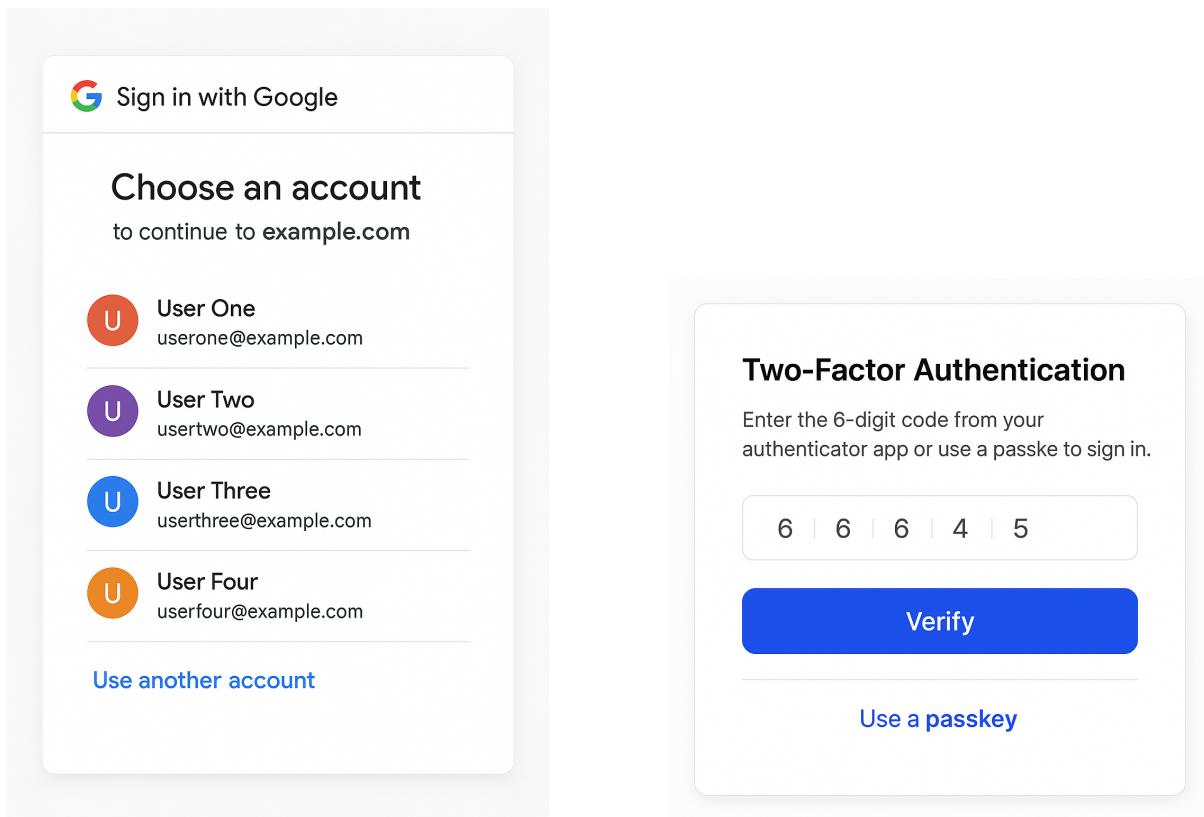


Figure 15: Google Authentication

- Separate interfaces for an interactive page for active contests with live scoreboards and rankings, and a detailed page for inactive contests with problem insights, difficulty levels, and hints.

ID	Title	Solved
A	Problem A	73 >
B	Problem B	56
C	Problem C	41
D	Problem D	18
E	Problem E	3

Figure 16: Interactive contest page

- Activity tracking, performance reports (SWOT analysis for example), and a dynamic rating system to unlock user privileges.

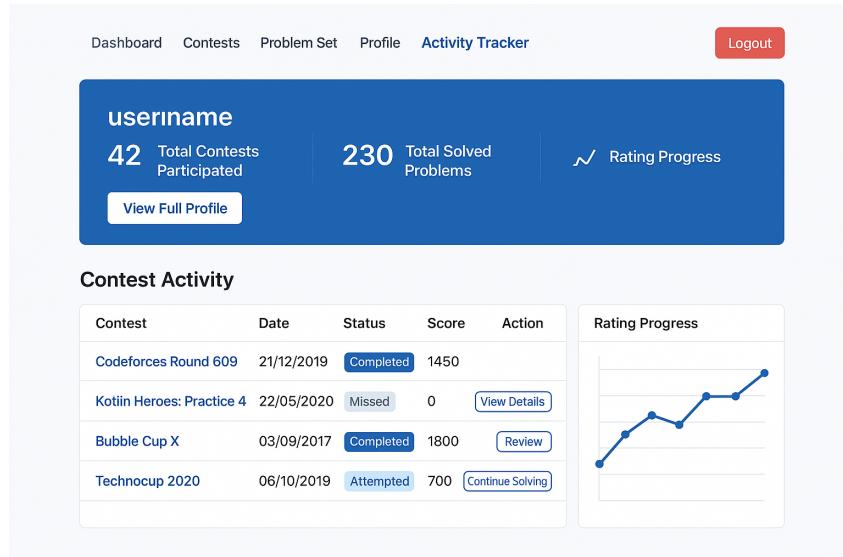


Figure 17: Activity tracking and performance reports

- Sandboxing for safe code execution, with built-in compilers for direct testing and compilation.

The figure shows a code submission form with the following fields:

- Title:** Submit Your C++ Code
- Text Area:** Paste your C++ code here (containing `int main(){return 0;}`)
- Buttons:** SUBMIT, View Question PDF

Figure 18: Sandboxing

- Active and inactive problems will be shown on separate pages, each tailored with relevant features.

The figure shows two separate pages for problems:

- (a) Active problems:** Shows a table of active problems with columns: ID, Title, Difficulty, and Tags.
- (b) Inactive problems:** Shows a table of inactive problems with columns: ID, Title, Difficulty, Hint, and Solution.

ID	Title	Difficulty	Tags
1000A	Codehorses T-shirts	1200	implementation greedy
1000B	Light It Up	1500	greedy
1000C	Covered Points Count	1700	implementation sortings
1000D	Yet Another Problem On a Sequence	1900	data structures dp combinatorics

ID	Title	Difficulty	Hint	Solution
1000A	Codehorses T-shirts	1200	Hint	Solution
1000B	Light It Up	1500	Hint	Solution
1000C	Covered Points Count	1700	Hint	Solution
1000D	Yet Another Problem On a Sequence	1900	Hint	Solution

(a) Active problems

(b) Inactive problems