ICPC ARCHIVE

# Description

The ICPC (International Collegiate Programming Contest) is one of the most prestigious student programming competitions held annually in Iran. It takes place in two stages: an online qualification round followed by a regional onsite round, which is typically hosted by Sharif University of Technology. During the contest, teams compete under time constraints to solve a series of algorithmic and analytical problems.

Preparing for such competitions requires consistent effort, deep study, and strong teamwork. While many international platforms exist for practice and preparation, there has long been a lack of a localized platform offering a complete archive of past ICPC Tehran problems with an integrated judging system. Solving past regional and online ICPC problems can greatly help participants familiarize themselves with the style and structure of the contests and improve their readiness.

This platform was created to fill this gap by providing a centralized, accessible archive of Tehran ICPC regional problems, along with features such as code submission, automatic judging, detailed results, and problem analysis. This platform aims to support Iranian students and competitive programmers in their journey toward better performance and deeper understanding of algorithmic problem solving.

# Features

**Authentication & Profile**

* Sign up and create a personal profile
* Sign in with existing credentials
* Sign out securely

**Contest & Problem Archive**

* Browse the list of available contests
* View the problems of a selected contest
* Open and read the full content of a problem (description, input/output, examples)

**Submissions**

* Submit code solutions for problems
* View real-time judging results of submissions
* Access a personal submission history with detailed reports

# Component Diagram

# Activity Diagram

## Sign up

* Username: '^[a-zA-Z0-9][a-zA-Z0-9\_]{4,14}$'
  + Min: 5
  + Max: 15
  + Letters, numbers, underscore
  + First character must be a letter
* Email: '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
* Password: '^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[^A-Za-z0-9\s])(?=\S+$).{8,20}$'
  + At least one digit
  + At least one lowercase letter
  + At least one uppercase letter
  + At least one special character
  + No whitespace character
  + Min: 8
  + Max: 20
* Password = Confirm password
* No duplicate entry
* All the fields are required

Username and email are unique. But I don't know where to handle it. Will it be automatically handled in Databse? Because I delcared them UNIQUE. But in real world sites, just after you typed the field, it is shown that the field is already taken or not.

# Database design

## Table - users

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | FK |
| 1 | id | int | 🗶 | ✓ | 🗶 | ✓ | ✓ | 🗶 |
| 2 | username | varchar (15) | 🗶 | 🗶 | 🗶 | ✓ | 🗶 | 🗶 |
| 3 | email | varchar (255) | 🗶 | 🗶 | 🗶 | ✓ | 🗶 | 🗶 |
| 4 | password | varchar (127) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 5 | created\_at | timestamp | 🗶 | 🗶 | current timestamp | 🗶 | 🗶 | 🗶 |

**Notes**

* Future: profiles table

## Table - contests

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | FK |
| 1 | id | int | 🗶 | ✓ | 🗶 | ✓ | ✓ | 🗶 |
| 2 | year | int | 🗶 | 🗶 | 🗶 | ✓ | 🗶 | 🗶 |
| 3 | question\_no | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |

**Notes:**

* Future: type (online or regional)
* Notes aren’t needed. I will put them in html documents directly
* You could set the year as primary key. That would have issues. One issue is that if you later add online contests, one year has more than one contest and so the year doesn’t become unique. Also, the table would have only one column which is kind of odd!
* question\_no is the number of questions inside a contest. This might be a derived attribute and may look redundant, but it won’t change in the future and also by considering the contests page, every time this page loads, you have to write query for every contest and this isn’t optimized. Maybe in the future you don’t write the question\_no value manually, but by querying for every contest in ddl time.

## Table - problems

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | FK |
| 1 | id | int | 🗶 | ✓ | 🗶 | ✓ | ✓ | 🗶 |
| 2 | label | char (1) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 3 | title | varchar (255) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 4 | time\_limit | int | 🗶 | 🗶 | 1000 | 🗶 | 🗶 | 🗶 |
| 5 | memory\_limit | int | 🗶 | 🗶 | 256 | 🗶 | 🗶 | 🗶 |
| 6 | description | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 7 | input | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 8 | output | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 9 | contest\_id | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | ✓ |

**Notes:**

* and? tags, score (or level), difficulty, tutorials and comments
* the text is in html format with tags (like <ul>, <ol>, <p>, latex, <em>, <strong>, <mono>, <quote>, <underline>)
* images are stored locally, only the link of a image is inside the problem text.
* a problem may have multiple images in different places.
* images may position differently. to apply that, you can use conventional class namings so that for each class name, that position gets applied.
* a shape may have caption!

## Table - samples

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | | FK |
| 1 | problem\_id | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | ✓ | ✓ |
| 2 | number | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 3 | input | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | | 🗶 |
| 4 | output | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | | 🗶 |
| 5 | note | text | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | | 🗶 |

**Notes:**

* until, 2017, a problem had only one sample. But then, they became many. To import the samples to view, get the visible samples of that problem
* note is in html format. Because it might contain images and ordered lists.
* The number is the sample numbers in a particular problem. Using this number, samples are sorted
* The combination of problem\_id and number is the primary key. I didn’t use pure id for that

## Table - tests

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | | FK |
| 1 | problem\_id | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | ✓ | ✓ |
| 2 | number | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 3 | input\_path | varchar (255) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | | 🗶 |
| 4 | output\_path | varchar (255) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | | 🗶 |

**Notes:**

* Test inputs and output files are pretty huge. The biggest test file is 2024-E-34-IN with 26 MB size. So, storing them directly inside Database isn’t a good practice. So, instead, I only store the path of the files inside Database.
* The number is the test numbers in a particular problem. Using this number, tests are sorted (test execution order matters)
* The combination of problem\_id and number is the primary key. I didn’t use pure id for that

## Table - submissions

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Name | Datatype | Nullable | AI | Default | Uniqueness | PK | FK |
| 1 | id | int | 🗶 | ✓ | 🗶 | ✓ | ✓ | 🗶 |
| 2 | submit\_time | timestamp | 🗶 | 🗶 | current timestamp | 🗶 | 🗶 | 🗶 |
| 3 | time | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 4 | memory | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 5 | status | enum | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 6 | source\_code | text | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 7 | language | varchar (255) | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| 8 | problem\_id | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | ✓ |
| 9 | user\_id | int | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | ✓ |

**Notes:**

* And? Language table and other tables that may be extracted from API
* Time is in milliseconds
* Memory is in kilobytes
* Status should be compared with judge0 states

## Relationships

|  |  |  |
| --- | --- | --- |
| Name | Entities and Cardinality | Description |
| makes | user (1)  submission (N) | A user can make multiple submissions  A submission belongs to only one user |
| has | contest (1)  problem (N) | A contest has many problems  A problem belongs to only one contest |
| has | problem (1)  sample (N) | A problem has many samples  A sample belongs to only one problem |
| has | problem (1)  test (N) | A problem has many tests  A test belongs to only one problem |
| gets | problem (1)  submission (N) | A problem gets many submissions  A submission is submitted for only one problem |

# Data

## Analysis:

|  |  |
| --- | --- |
| Year | # |
| 1999 | 8 |
| 2000 | 9 |
| 2001 | 8 |
| 2002 | 7 |
| 2003 | 8 |
| 2004 | 8 |
| 2005 | 10 |
| 2006 | 9 |
| 2007 | 10 |
| 2008 | 10 |
| 2009 | 10 |
| 2010 | 11 |
| 2011 | 11 |
| 2012 | 10 |
| 2013 | 11 |
| 2014 | 11 |
| 2015 | 10 |
| 2016 | 12 |
| 2017 | 11 |
| 2018 | 11 |
| 2019 | 12 |
| 2020 | 13 (+ Judging notes) |
| 2022 | 11 (+ Judging notes) |
| 2023 | 13 (+ Judging notes) |
| 2024 | 10 (+ Judging notes) |
| Sum | 254 |
| Number | 25 |
| Average | 10.16 |

## Data Extraction

**How did I extract the problems?**

* + 1. Download PDFs from <https://icpc.ir>
       - Some years were in word format, I converted them to PDF
       - Some years had a separate file for each problem, I combined them into one file
    2. Convert PDFs to markdown using <https://lightpdf.com/pdf-to-markdown>
    3. Format the markdown in particular rules
    4. Convert markdown to HTML using [https://markdowntohtml.com](https://markdowntohtml.com/)
    5. Format HTMLs in particular rules
    6. Convert HTMLs to DML queries manually

**How did I extract samples?**

1. Create samples folder in a particular format
2. Create sample file from pdfs manually (not test case files)

**How did I extract test?**

1. Extract raw test-case files
2. Create folders in a particular format
3. Transfer test files into the folders
4. Change test file names using a python script
5. Change the display hierarchy using a python script

Also, I only included contests since 2017. The past contests don't have a complete test-case