Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Coding Cat

CSE 3224

Information System Design

&

Software Engineering Lab

Submitted By:

Ahmad Subaktagin Jabir	16.02.04.061
Arunima Ayshee	16.02.04.066
Rahat Bin Osman	16.02.04.083
Aniqua Tabassum	16.02.04.085

Date of Submission: **24 September, 2019**

Introduction:

Our project, Coding Cat, is an online judge, where it is mandatory to store several crucial data of the users and the programming problems in the database. We have tried to design our database in such a manner that it will not create any redundancy and will not cause any unwanted disaster while manipulating the data within. Additionally, we have tried to make the data retrieval process as efficient as possible.

Name of the Entities with Primary, Foreign, and Composite keys:

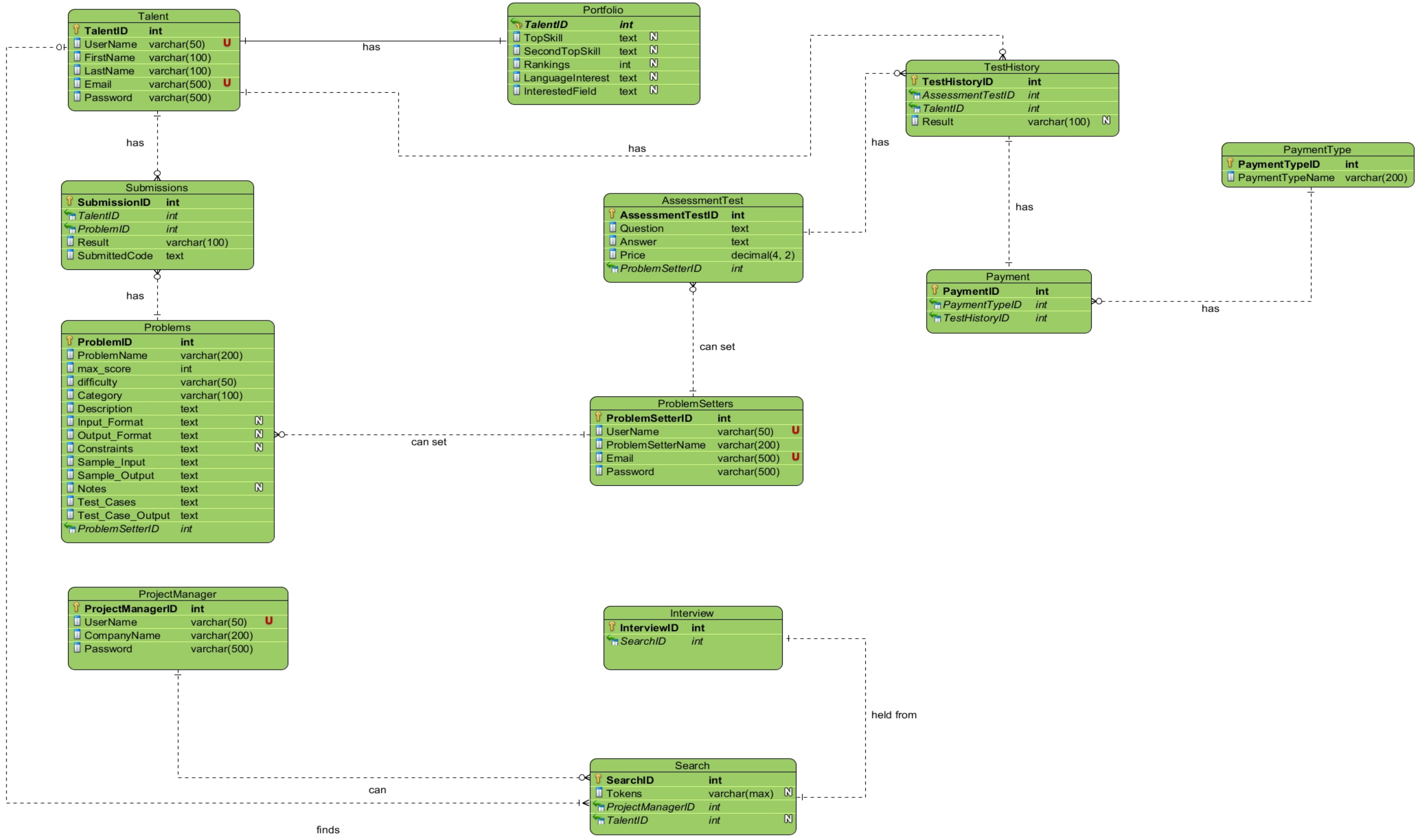
An entity is any object in the system that we want to model and store information about. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database. Our Project has 12 Entities. They are given below:

1. Talents
2. Portfolio
3. Problems
4. Submission
5. Problem Setter
6. Assessment Test
7. Test History
8. Payment
9. Payment Type
10. Project Manager
11. Search
12. Interview

Primary and Foreign Key

Entity	**Primary Key**	**Foreign Key**
Talents	TalentID	
Portfolio	TalentID	TalentID
Problems	ProblemID	ProblemSetterID
Submission	SubmissionID	TalentID ProblemID
Problem Setter	ProblemSetterID	
Assessment Test	AssessmentTestID	ProblemSetterID
Test History	TestHistoryID	AssessmentTestID TalentID
Payment	PaymentID	TestHistoryID PaymentTypeID
Payment Type	PaymentTypeID	
Project Manager	ProjectManagerID	
Search	SearchID	ProjectManagerID TalentID
Interview	InterviewID	SearchID

Entity Relationship (ER) Diagram with multiplicity:



Talents:

- Each Talent can have one portfolio. It is a One to One Relationship.
- Each Talent can have many submissions. It is a One to Many Relationship.
- Each Talent can have many Test History. It is a One to Many Relationship.

Portfolio:

- One Portfolio belongs to One Talent. It is a One to One Relationship.

Problems:

- Many problems can be made by One Problem Setter. It is a Many to One Relationship.
- One problem can have many submissions. It is a One to Many Relationship.

Submission:

- Many submissions can be made by a Talent. It is a Many to One Relationship.
- Many submissions can come for a Problem. It is a Many to One Relationship.

Problem Setter:

- Each Problem Setter can set many assessment tests. It is a One to Many Relationship.
- Each Problem Setter can set many problems. It is a One to Many Relationship.

Assessment Test:

- Each assessment test can have many test histories. It is a One to Many Relationship.
- Many assessment tests can be made by One Problem Setter. It is a Many to One Relationship.

Test History:

- Each Test History has a payment. It is a One to One Relationship.

Payment Type:

- Each Payment Type can be used many times during Payment. It is a One to Many Relationship.

Payment:

- Each payment will be done on the basis of each test History. It is a One to One Relationship.
- A payment type can be used Multiple times. It is a Many to One Relationship.

Project Manager:

- Each Project Manager can search many times. It is a One to Many Relationship.

Search:

- From every search, there can be one interview. It is a One to One Relationship.

Interview:

- Many interviews can be done by a search. It is a Many to One Relationship.

Relational Model:

a) SQL Commands with appropriate relationships and Highlighting Primary & Foreign Key:

Talents

```
CREATE TABLE Talents (
  TalentID  int IDENTITY(1,1) NOT NULL,
  UserName  varchar(50) NOT NULL UNIQUE,
  FirstName varchar(100) NOT NULL,
  LastName  varchar(100) NOT NULL,
  Email     varchar(500) NOT NULL UNIQUE,
  Password  varchar(500) NOT NULL,
  PRIMARY KEY (TalentID));
```

Portfolio

```
CREATE TABLE Portfolio (
 TalentID     int NOT NULL FOREIGN KEY REFERENCES Talents(TalentID),
 TopSkill       text,
 SecondTopSkill  text,
 Rankings       int,
 LanguageChoice  text,
 InterestedField text,
 PRIMARY KEY (TalentID));
```

Problems

```
CREATE TABLE Problems (
 ProblemID       int IDENTITY NOT NULL,
 ProblemName      varchar(200) NOT NULL,
 max_score       int NOT NULL,
 difficulty      varchar(50) NOT NULL,
 Category        varchar(100) NOT NULL,
 Description     text NOT NULL,
 Input_Format     text NULL,
 Output_Format    text NULL,
 Constraints      text NULL,
 Sample_Input     text NOT NULL,
 Sample_Output    text NOT NULL,
 Notes           text NULL,
 Test_Cases       text NOT NULL,
 Test_Case_Output text NOT NULL,
 ProblemSetterID  int NOT NULL FOREIGN KEY REFERENCES
ProblemSetters(ProblemSetterID),
 PRIMARY KEY (ProblemID));
```

Submissions

```
CREATE TABLE Submissions (
 SubmissionID  int IDENTITY (1,1) NOT NULL,
 TalentID     int NOT NULL FOREIGN KEY REFERENCES Talents(TalentID),
 ProblemID    int NOT NULL FOREIGN KEY REFERENCES Problems(ProblemID),
 Result       varchar(100) NOT NULL,
 SubmittedCode text NOT NULL,
 PRIMARY KEY (SubmissionID));
```

Problem Setters

```
CREATE TABLE ProblemSetters (
 ProblemSetterID   int IDENTITY(1,1) NOT NULL,
 UserName         varchar(50) NOT NULL UNIQUE,
 ProblemSetterName varchar(400) NOT NULL,
 Email           varchar(500) NOT NULL UNIQUE,
 Password        varchar(500) NOT NULL,
 PRIMARY KEY (ProblemSetterID));
```

Assessment Test

```
CREATE TABLE AssessmentTest (
 AssessmentTestID int IDENTITY(1,1) NOT NULL,
 Question        text NOT NULL,
 Answer          text NOT NULL,
 Price           decimal(4, 2) NOT NULL,
 ProblemSetterID  int NOT NULL FOREIGN KEY REFERENCES
ProblemSetters(ProblemSetterID),
 PRIMARY KEY (AssessmentTestID));
```

Test History Table

```
CREATE TABLE TestHistory (
  TestHistoryID int IDENTITY(1,1) NOT NULL,
  AssessmentTestID int NOT NULL FOREIGN KEY REFERENCES
AssessmentTest(AssessmentTestID),
  TalentID        int NOT NULL FOREIGN KEY REFERENCES Talents(TalentID),
  Result          varchar(100),
  PRIMARY KEY (TestHistoryID));
```

Payment Type Table

```
CREATE TABLE PaymentType (
  PaymentTypeID   int IDENTITY(1,1) NOT NULL,
  PaymentTypeName varchar(200) NOT NULL,
  PRIMARY KEY (PaymentTypeID));
```

Payment Table

```
CREATE TABLE Payment (
  PaymentID       int IDENTITY(1,1) NOT NULL,
  PaymentTypeID    int NOT NULL FOREIGN KEY REFERENCES
PaymentType(PaymentTypeID),
  TestHistoryID int NOT NULL FOREIGN KEY REFERENCES
TestHistory(TestHistoryID),
  PRIMARY KEY (PaymentID));
```

Project Manager Table

```
CREATE TABLE ProjectManager (
  ProjectManagerID int IDENTITY(1,1) NOT NULL,
  Company         varchar(300) NOT NULL,
  UserName         varchar(50) NOT NULL UNIQUE,
  Password        varchar(500) NOT NULL,
 PRIMARY KEY (ProjectManagerID));
```

Search Table

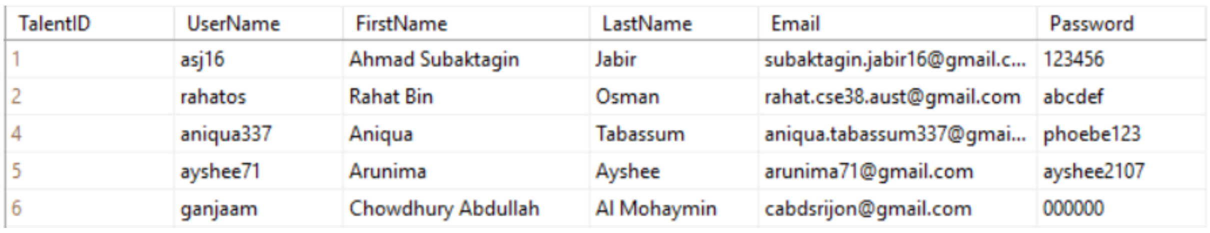
```
CREATE TABLE Search (
 SearchID        int IDENTITY(1,1) NOT NULL,
 Tokens          varchar(max),
 ProjectManagerID int NOT NULL FOREIGN KEY REFERENCES
ProjectManager(ProjectManagerID),
 TalentID      int NOT NULL FOREIGN KEY REFERENCES Talents(TalentID),
 PRIMARY KEY (SearchID));
```

Interview Table

```
CREATE TABLE interview (
 InterviewID      int IDENTITY(1,1) NOT NULL,
 SearchID int NOT NULL FOREIGN KEY REFERENCES Search(SearchID),
 PRIMARY KEY (InterviewID));
```

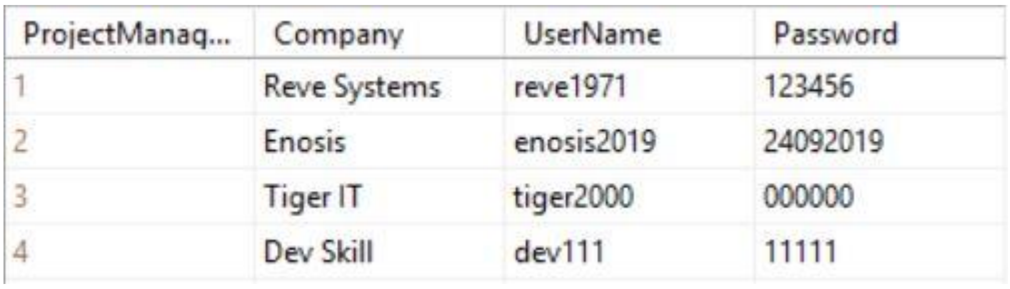
b) Entities with Dummy Data:

1. Talent:



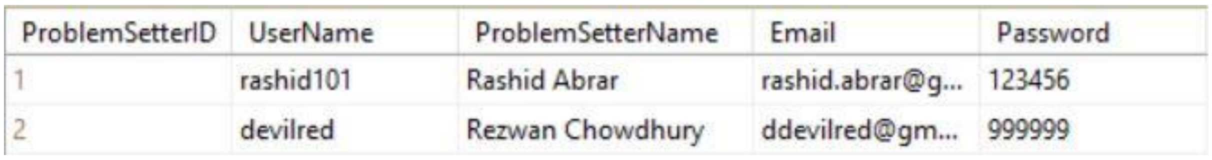
TalentID	UserName	FirstName	LastName	Email	Password
1	asj16	Ahmad Subaktagin	Jabir	subaktagin.jabir16@gmail.c...	123456
2	rahatos	Rahat Bin	Osman	rahat.cse38.aust@gmail.com	abcdef
4	aniqua337	Aniqua	Tabassum	aniqua.tabassum337@gmai...	phoebe123
5	ayshee71	Arunima	Ayshee	arunima71@gmail.com	ayshee2107
6	ganjaam	Chowdhury Abdullah	Al Mohaymin	cabdsrijon@gmail.com	000000

2. Project Manager:



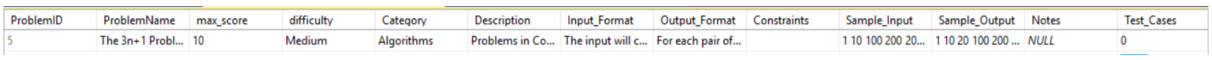
ProjectManag...	Company	UserName	Password
1	Reve Systems	reve1971	123456
2	Enosis	enosis2019	24092019
3	Tiger IT	tiger2000	000000
4	Dev Skill	dev111	11111

3. Problem Setters:



ProblemSetterID	UserName	ProblemSetterName	Email	Password
1	rashid101	Rashid Abrar	rashid.abrar@g...	123456
2	devilred	Rezwan Chowdhury	ddevilred@gm...	999999

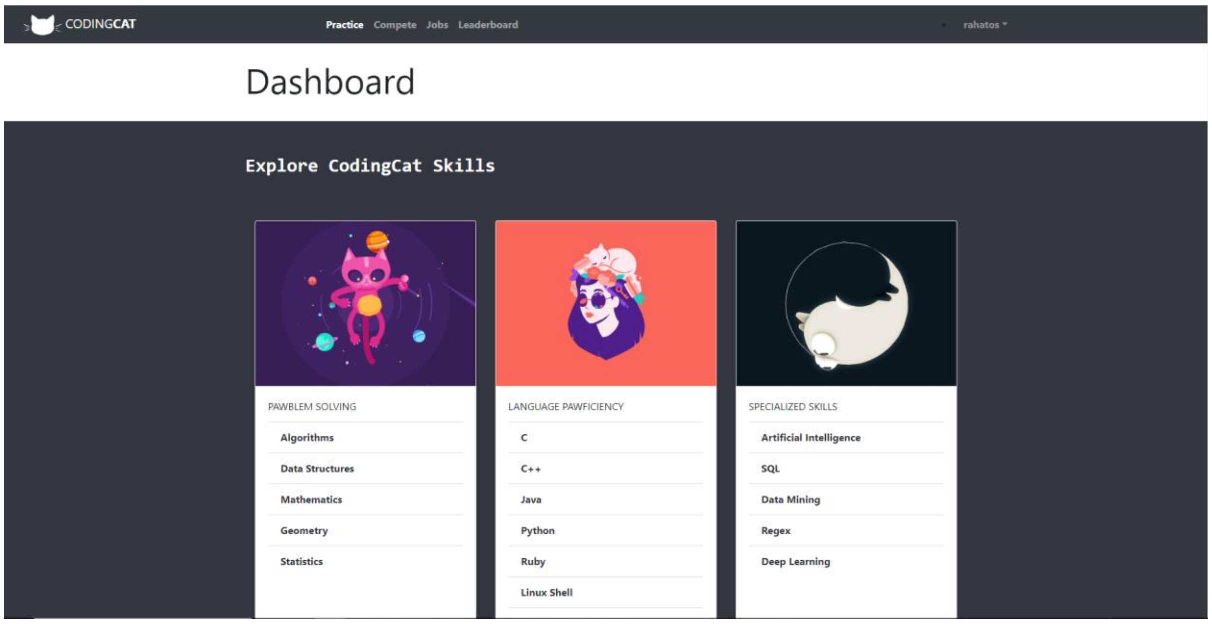
4. Problems:



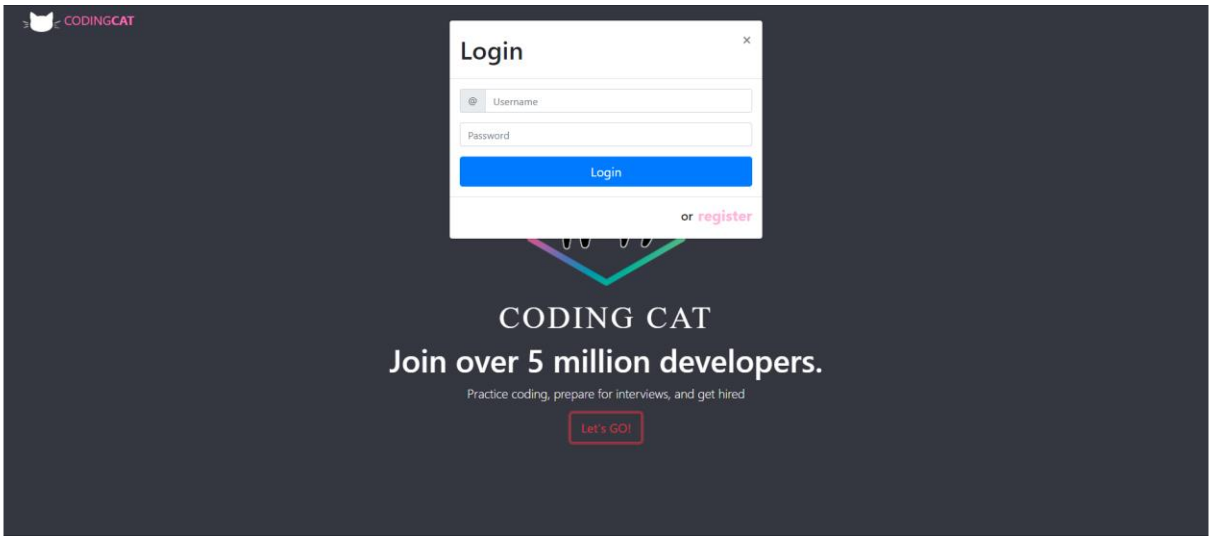
ProblemID	ProblemName	max_score	difficulty	Category	Description	Input_Format	Output_Format	Constraints	Sample_Input	Sample_Output	Notes	Test_Cases
5	The 3n+1 Probl...	10	Medium	Algorithms	Problems in Co...	The input will c...	For each pair of...		1 10 100 200 20...	1 10 20 100 200 ...	*NULL*	0

<u>Designs of Our Project:</u>

a) Home Page: In the homepage, the user will be able to navigate among our various features such as practicing problems or increasing language efficiency, browsing through jobs, participating in coding competitions etc.



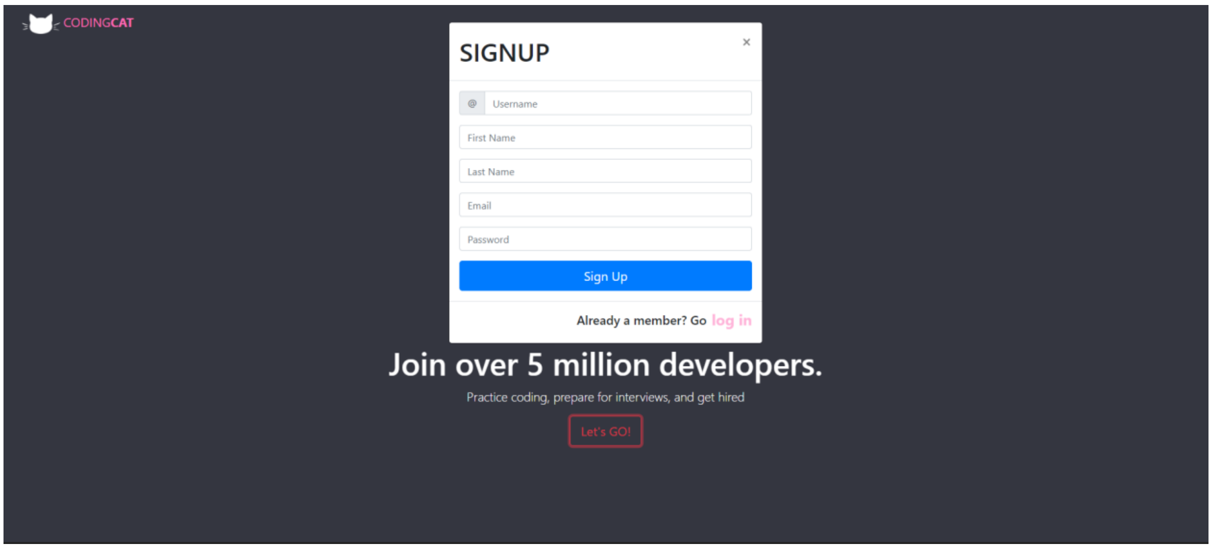
b) Login Page: To be able to get access to our features, the registered users must be logged in to our system, and they will be able to log in throughout this login page. The login process will require users to enter the unique username with which they registered into the system and the corresponding password. If the user is not a registered into our system, then s/he can go to the "register" link placed in the bottom right of the box and get themselves registered.


CODINGCAT
Login
×
Username
Password
Login
or register
CODING CAT
Join over 5 million developers.
Practice coding, prepare for interviews, and get hired
Let's GO!

c) **Registration Page:** Unregistered users must register first to be a part of the system. For that purpose, we have designed this page. The registration process will be completed by taking a few information from the user, such as the username by which they will be recognized in our system, email ID, first name, last name and password. This password will be converted into a hashcode before being saved into the database.

If the user is already registered in our system, then s/he can navigate to the login page using the link provided in the bottom right of the box.

In our report on Functional and Non-functional report on Coding Cat, we had mentioned that it will not take more than 120 seconds for a new user to register. We have designed the signup page accordingly.


CODINGCAT

SIGNUP

×

@ Username

First Name

Last Name

Email

Password

Sign Up

Already a member? Go log in

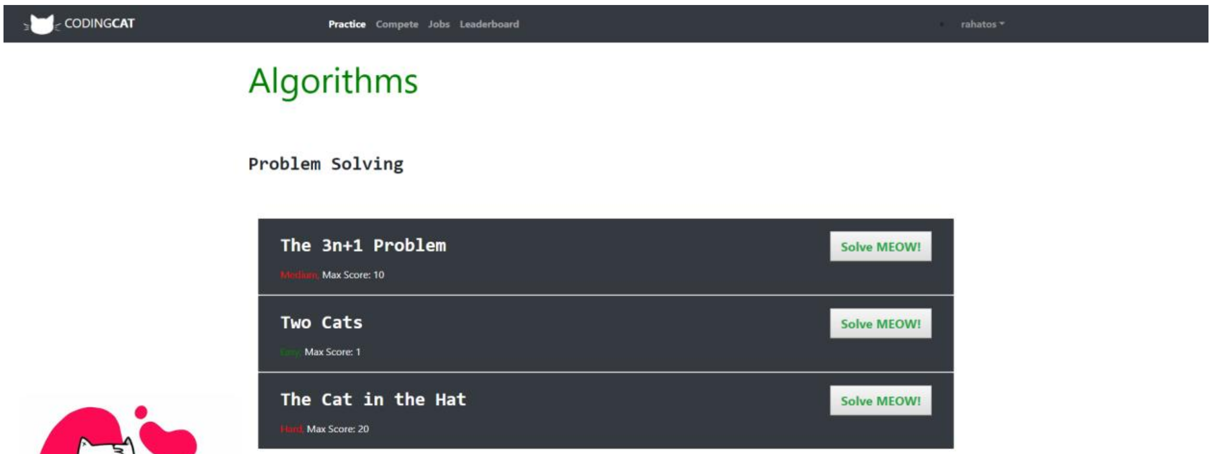
Join over 5 million developers.

Practice coding, prepare for interviews, and get hired

Let's GO!

d) Practice Problems Page:

In the Practice Problem page, users will be able to browse through any branch of computer science and solve any problem they like. The name of the problem will be on the left of every row, as mentioned below, and a button will be placed on the right that will navigate them to the problem page, where they will see the complete problem statement and will be able to submit their codes. The difficulty level and maximum score of the problems will also be mentioned under the name of the problem.


CODINGCAT Practice Compete Jobs Leaderboard rahatos

Algorithms

Problem Solving

The 3n+1 Problem — Solve MEOW!
Medium: Max Score: 10

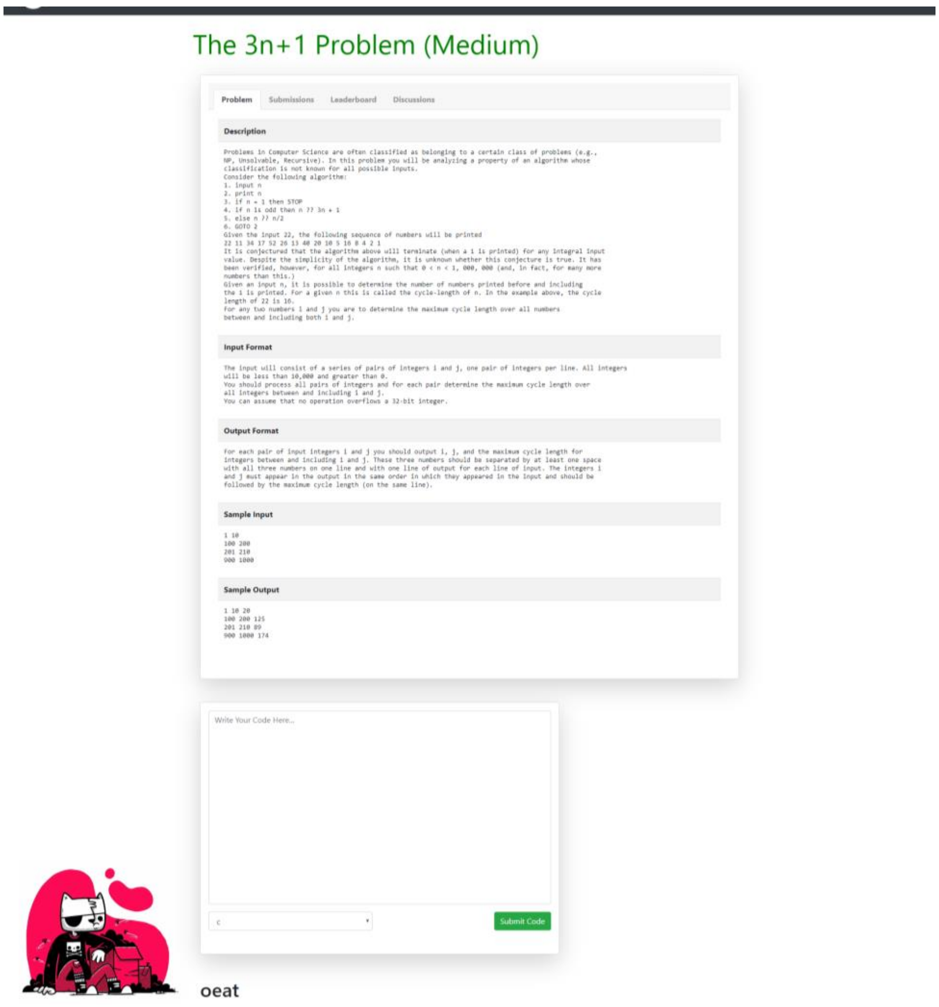
Two Cats — Solve MEOW!
Easy: Max Score: 1

The Cat in the Hat — Solve MEOW!
Hard: Max Score: 20

e) Problem Page:

This is the page where users will see the full problem statement of the problem they decided to give a try. The name of the problem will be mentioned on the top of this page, along with its difficulty level. Right below that, users will be given options to navigate to their submission history, dashboard and leader board of this problem. Next will come the complete problem statement, where the programming problem will be discussed at length. After that, input and output format, sample input and output will be displayed in respective orders. If the problem setter mentions some constraints for a problem, that will be mentioned in this page too.

In this page, users will be able to submit their code as well, and get the verdict. This feature has been placed below the sample output, after the complete description of a problem ends.



Description

```
Problems in Computer Science are often classified as belonging to a certain class of problems (e.g.,
NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose
classification is not known for all possible inputs.
Consider the following algorithm:
1. input n
2. print n
3. if n = 1 then STOP
4. if n is odd then n ?? 3n + 1
5. else n ?? n/2
6. GOTO 2
Given the input 22, the following sequence of numbers will be printed
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input
value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has
been verified, however, for all integers n such that 0 < n < 1, 000, 000 (and, in fact, for many more
numbers than this.)
Given an input n, it is possible to determine the number of numbers printed before and including
the 1 is printed. For a given n this is called the cycle-length of n. In the example above, the cycle
length of 22 is 16.
For any two numbers i and j you are to determine the maximum cycle length over all numbers
between and including both i and j.
```

Input Format

```
The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers
will be less than 10,000 and greater than 0.
You should process all pairs of integers and for each pair determine the maximum cycle length over
all integers between and including i and j.
You can assume that no operation overflows a 32-bit integer.
```

Output Format

```
For each pair of input integers i and j you should output i, j, and the maximum cycle length for
integers between and including i and j. These three numbers should be separated by at least one space
with all three numbers on one line and with one line of output for each line of input. The integers i
and j must appear in the output in the same order in which they appeared in the input and should be
followed by the maximum cycle length (on the same line).
```

Sample Input

```
1 10
100 200
201 210
900 1000
```

Sample Output

```
1 10 20
100 200 125
201 210 89
900 1000 174
```

Conclusion:

The necessity of an attractive and user-friendly user interface, along with an efficient and rich database cannot be denied while making a system nowadays. We have tried our best to design our website in such a way that the user gets the best experience and their useful data gets stored in our database in the most efficient manner, which will help us improve the user experience even more in the advanced development phases. We hope that our work so far is sufficient to fulfill our goals so far, although we are still trying relentlessly to improve and renovate our system to provide our users with the top-class experience. It is our humble request to see any mistakes of ours in the eyes of forgiveness.