CSCE608: Project #1

Project name: Assessment System for Any Tournament contest(ASAT)
Student Name: Sungkeun Kim (UIN: 325003839)

Web site link:
https://db-project1.herokuapp.com/

Summary
For this project we will be designing a site to assist in managing and judging contestants for any types of a contest that you want to hold. This will be done by allowing an administrator to create questions, add judges, and formulate contestants' performances based on a tournament-like hierarchal structure. The key component to our project is that the admin has all authority. There is no signup page since the power of making accounts is under the scope of the admin.

This site will help any type of contests make the process of judging contestants easier by removing most of the physical work of summing/averaging scores, assigning contestants to each judge, and prevent the judges from backtracking on their previous score decision. Each judge will rate contestants based on metrics determined by the administrator. The admin can monitor all the results and inputs of judges for every contestant and the admin gets to decide which contestant will make it to the next round based on judge's score input.

User Stories

User Story 1	User Story 2
Feature: Create a Contest ● As an admin (user) ● I can setup contests, divisions and rounds ● I want to see all contests, divisions, and rounds already created as a table	Feature: Create a Judge ● As an admin (user) ● I can setup judge accounts ● I can put in name, email address, password and which contest to participate to the judge ● I want to see all judges already created as a table
User Story 3	User Story 4
Feature: Create an contestant ● I can assign contestants to contests ● I can put in name, email address and which division to participate in ● I want to see all contestants as a table	Feature: Create Questions ● As an admin(user) Admin ● I can create questions ● I can assign assign questions to different rounds ● I want to see all questions assigned to each round
User Story 5	User Story 6
Feature: Manage an ongoing contest ● As an admin (user)	Feature: Judge a contest ● As a judge(user)

● I can see all active contests ● I can go to each rounds to see the performance of each contestants ● I can decide which contestant will make it to the next round	● I can see all active divisions that I am assigned to ● I can go to each rounds to see the contestants and give scores

For User story 1, we were able to implement the basic Contest adding feature of the software. Here we allow the admin to include the division and rounds for each contest. We ended up having to make multiple divisions when divisions had many rounds. We also included a table that allows the admin to see which contests have been made.

For User story 2, we were able to allow the admin to add judges. The judge page takes in the name, email address, password and contest affiliation. We also included a table that allows the admin to see which judges have been made.

For User story 3, we were able to implement the contestant-adding feature. Since contestants would be notified about their evaluation separately we only included the contestant's name and email address. We also included a table that allows the admin to see which contestants have been made.

For User story 4, we were able to implement the question setup feature.The admin will create custom questions for every contest. We created this page to allow the admin to create questions that are either Integer based or string based reply.

For User story 5, we were able to implement the manage contest feature. This feature allows the admin to monitor all evaluations of contestants from each judge. This allows the admin all the information in an organized fashion. This is crucial because admin gets to decide which contestants move on to the next round. However, this feature is not implemented yet.

For User story 6, we were able to implement the judge evaluation feature. This feature is only applicable for the judge accounts. When logged in, the judge can see the affiliation of contest and all the contestants that are available in each round. After making the evaluation, the judge can push all the results to the database.

Data Collection

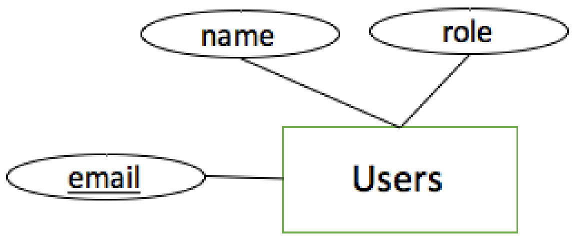
Default data are generated by programming in Ruby on Rails. In order for me to test database, I made every name of contests, judges, contestants having same patterns. For the contest, for instance, its name starts with "contest" and number (contest1, contest2, …).

I created 4 contests that includes 4 divisions. Each division has 4 round and each round has 7 questions. Also I added both 8 judges and contestants and each contest has 2 contestants and judges respectively. Therefore, Assess relation has 11,200 tuples (4*4*4*7*2*2 = 1,792). Related source code is located in db/seeds.rb

Entity-Relational Diagram

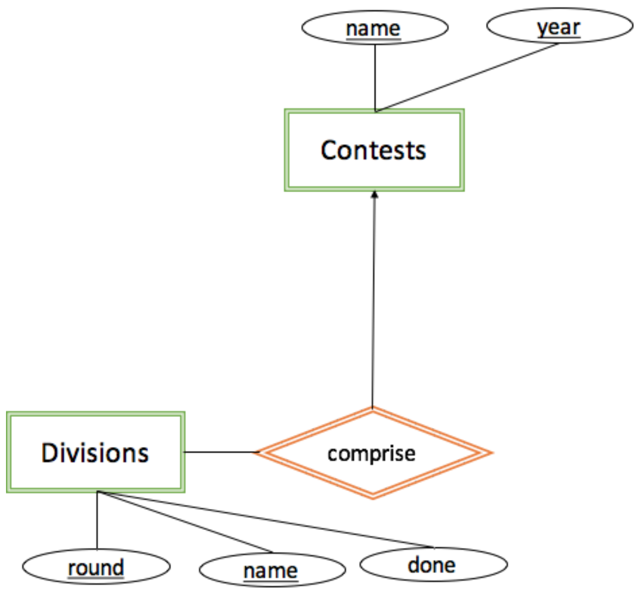
1. User(<u>email</u>, name, role)

Our web site has to deal with users and they have three different types - admin, judge, contestant. User entity has three attributes which are name, email, and role. By using role attribute, our system can distinguish user's type. There can be alternative db design for the user. For example, we can use subclass design pattern. That is, User can be super class and Admin, Judge, Contestant can be subclasses. Each subclass has an attribute called "role". However, I didn't chose subclasses because every user should have one role.



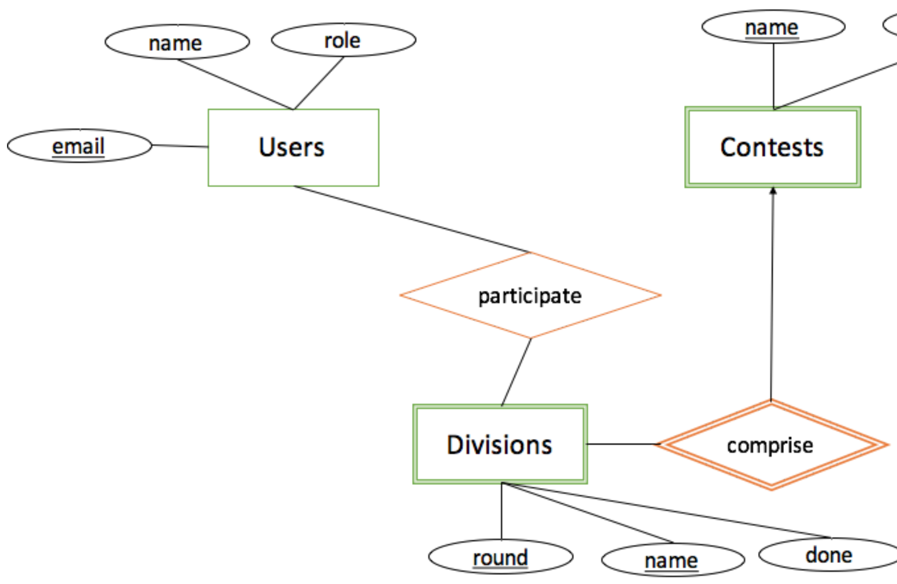
2. Contest(<u>name</u>, <u>year</u>), Division(<u>name</u>, <u>round</u>, done) and Comprise E/R

Each contest includes more than one divisions and each division includes more than one rounds. Also if every contestant is assessed, the system should remember it by done attribute. Division is weak entity set because it cannot be distinguished by itself because there can be a division with same name and round. For example, there can be "Rookie" division with 4 rounds in the different contest. Also, this relationship has many to one relationship.

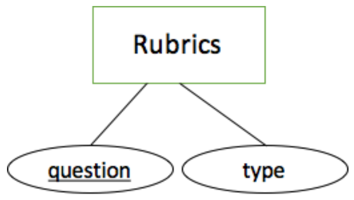


3. Participate Relationship

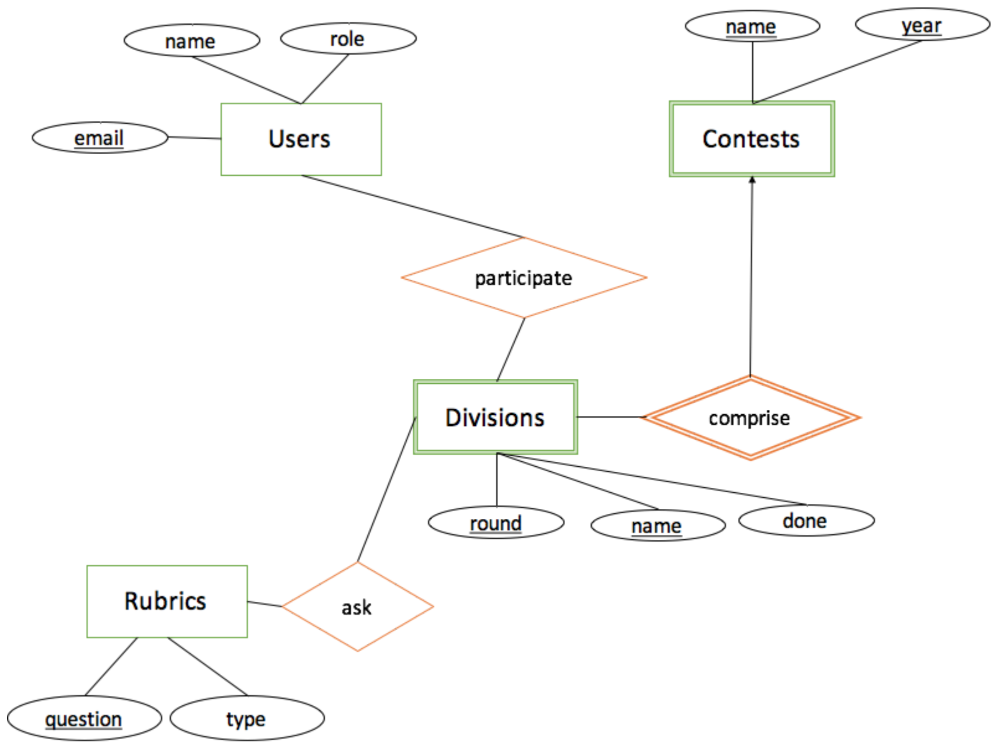
Either judge or contestant participates a contest. Currently, we assume that every judge or contestant participates all divisions and rounds even if database can manage it separately. This is because it make program complex and our goal is focused on database design. For the future work, program will provide an interface that admin assign a contestant to a specific division of a contest.



4. Rubrics(<u>question</u>, type)
Each round has it’s own rubrics for assessing contestants. Rubrics have question and its type.

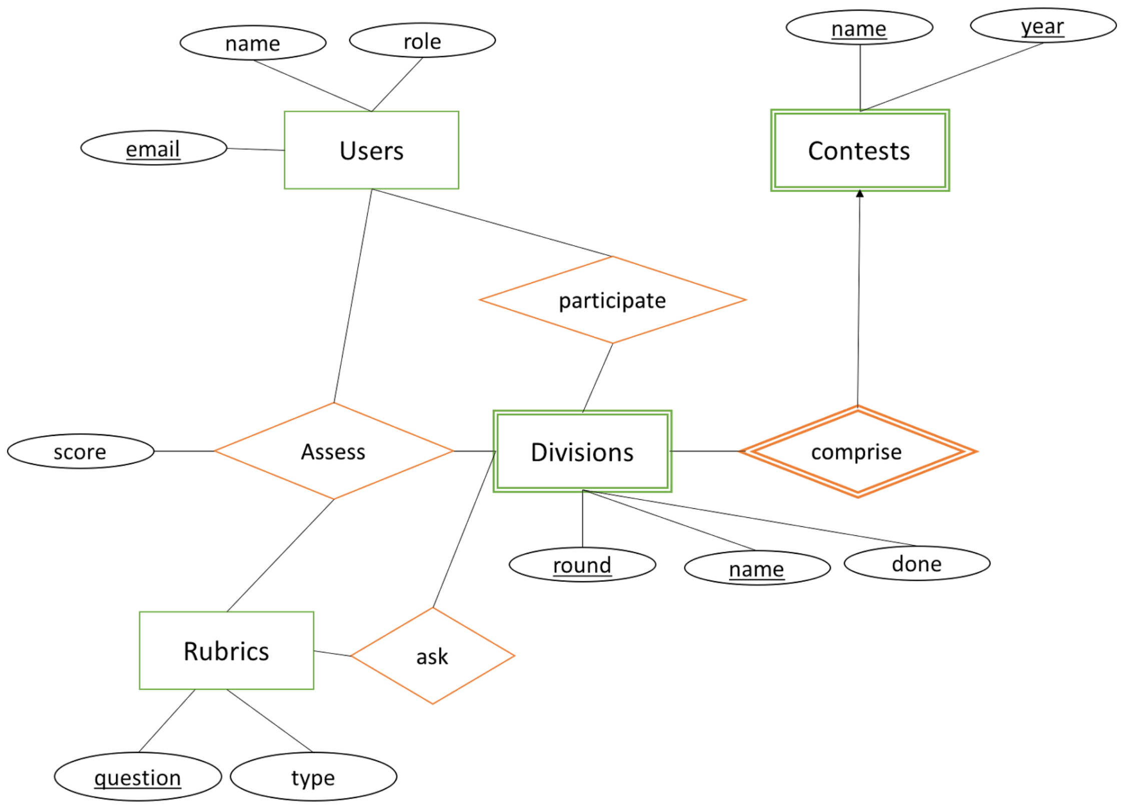


5. Ask relationship
A judge will ask some questions(rubrics) to a contestant who participates a division of a contest. In other words, ask relationship holds a list of rubrics that will be asked in a division.



7. Assess relationship

Judge who participates a contest will assess a contestant using rubrics that a division has. Assess relationship has an attributes called "score" that contains a score that judge assessed.

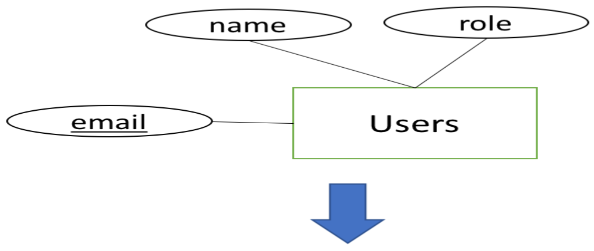


Final version of E/R Diagram

Table normalization

1. Convert User

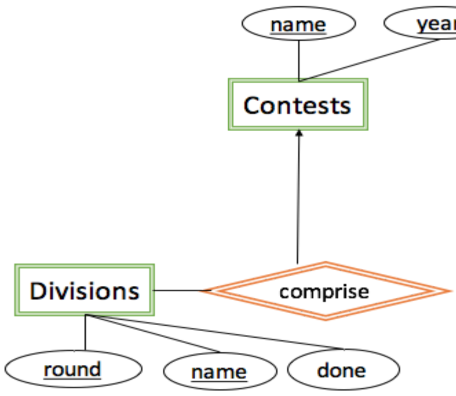
When user entity is converted to a relation with BCNF, it does not change anything because all FDs are {email -> name role} and its left side of every nontrivial FD is a super key.



Email	Name	Role
admin@gmail.com	Lance	admin
judge1@gmail.com	Sungkeun Kim	judge
participant1@gmail.com	Arron	contestant

2. Convert Contest, Division and Comprise

When the Contest, Division, and Comprise E/R Diagram are converted to relations, Comprise relationship is removed because it has no attributes and a weak entity set Division includes *name, year* of Contests relation for its complete key. There are no changes during the normalization because every left side of non-trivial FDs is super key.



Divisions

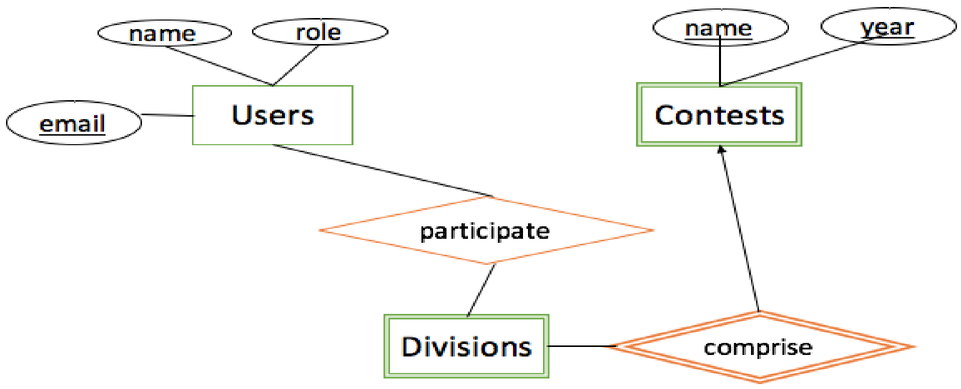
Contest	Year	Division	Round	Done
Texas Auctioneers	2017	Rookie	1	true
Texas Auctioneers	2017	Rookie	2	false
Texas Auctioneers	2017	Champion	1	false

Contests

Name	Year
Texas Auctioneers	2017
Dog Photo Contest	2016

3. Convert Participate relationship

When Participate relationship is converted to a relation, Participate relation includes *contest, year, division, round* attributes for its complete key as well as its own attribute *user*. There are no changes during the normalization because every left side of non-trivial FDs is super key.

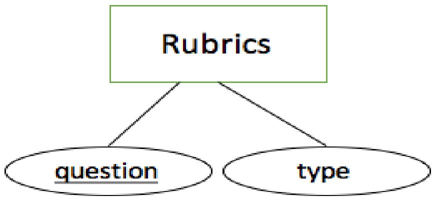


Participate

user	contest	year	division	round
judge1@gmail.com	Texas Auctioneers	2017	Rookie	1
paricipant1@gmail.com	Texas Auctioneers	2017	Rookie	1

4. Convert Rubric Entity Set

When user entity is converted to a relation with BCNF, it does not change anything because all FDs are {question -> type} and its left side of every nontrivial FD is a super key.

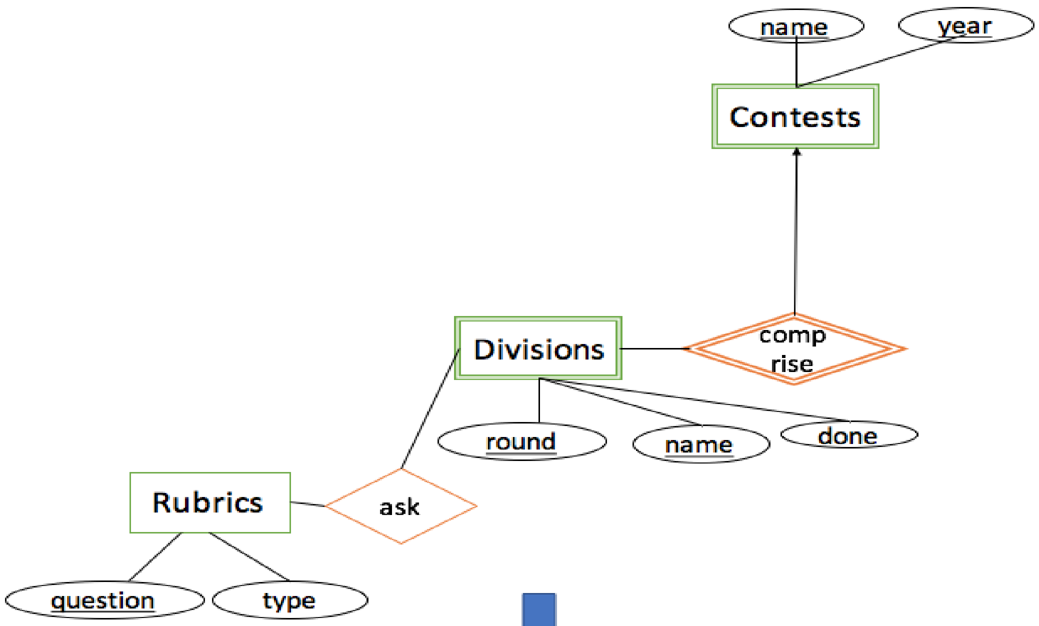


Rubrics

Question	Type (String or integer)
[Professional Image] Appearance, Manner and Attitude.	integer
[Professionalism] Overall impression: Do you believe the Auctioneer fairly represented the merchandise, the auction profession, and would you hire this Auctioneer to sell your sale.	string

5. Convert Ask relationship

When Ask relationship is converted to a relation, Ask relation includes contest, year, division, round, and questions attributes from Division and Rubric relation for its complete key. There are no changes during the normalization because every left side of non-trivial FDs is super key.

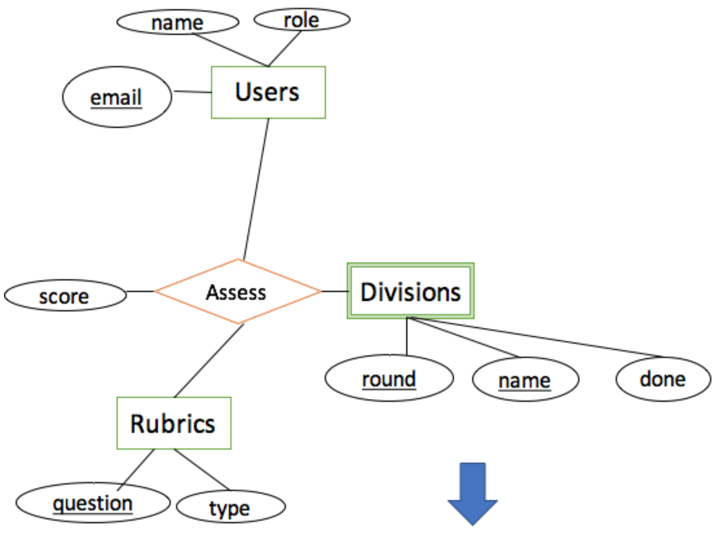


Ask

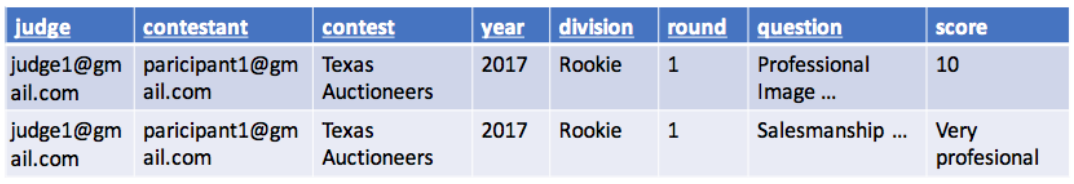
contest	year	division	round	question
Texas Auctioneers	2017	Rookie	1	Professional Image …
Texas Auctioneers	2017	Rookie	1	Salesmanship …

5. Convert Assess relationship

When the Assess relationship is converted to a relation, the Assess relation includes attributes judge(email), contestant(email) from User relation, contest, year, division, round from Divisions relation, and question from Rubrics relation as well as its own attribute score. There is no changes during the normalization because of the above reason.



Assess

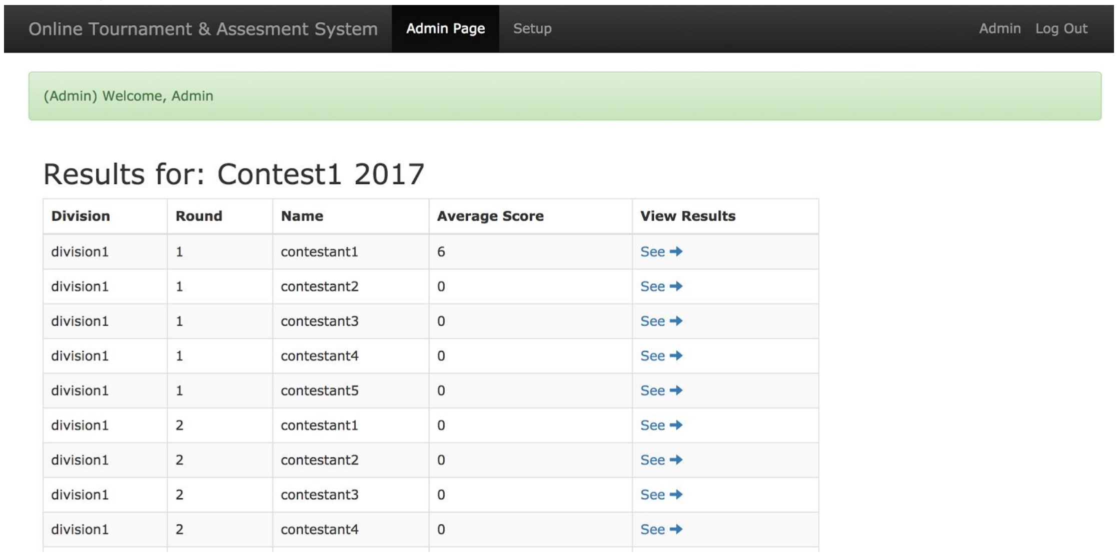


judge	contestant	contest	year	division	round	question	score
judge1@gmail.com	paricipant1@gmail.com	Texas Auctioneers	2017	Rookie	1	Professional Image ...	10
judge1@gmail.com	paricipant1@gmail.com	Texas Auctioneers	2017	Rookie	1	Salesmanship ...	Very profesional

User Interface

1. Summary page for all Contests

Once you login the site as a admin user, you can see summary page of all the contests have registered.

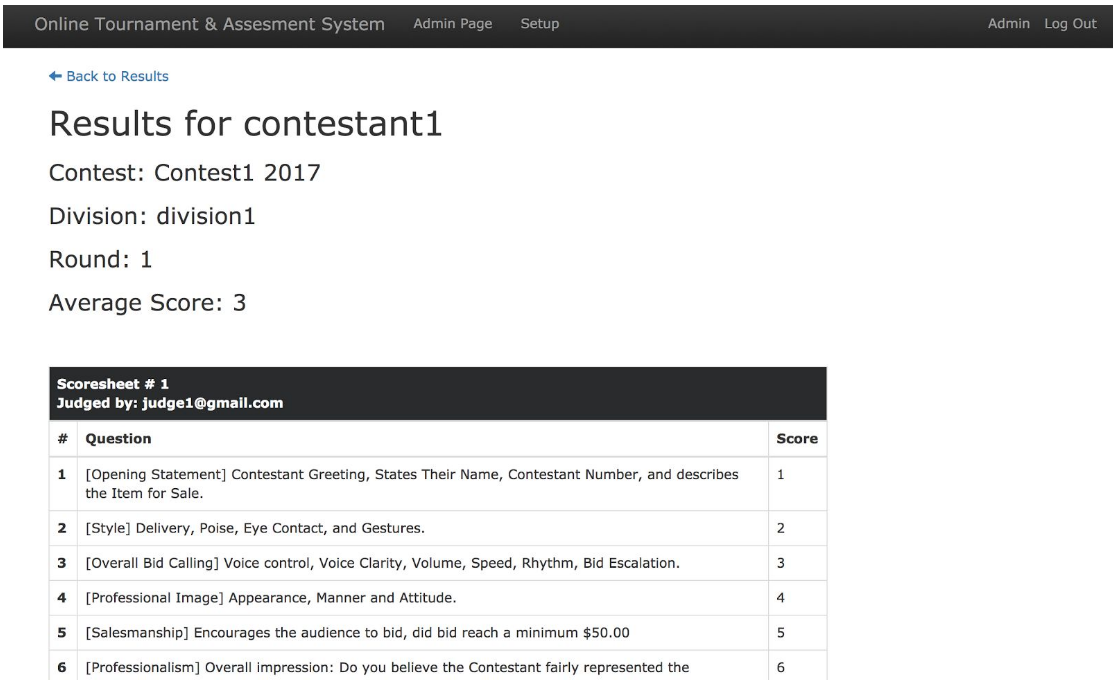


(Admin) Welcome, Admin

Results for: Contest1 2017

Division	Round	Name	Average Score	View Results
division1	1	contestant1	6	See ➔
division1	1	contestant2	0	See ➔
division1	1	contestant3	0	See ➔
division1	1	contestant4	0	See ➔
division1	1	contestant5	0	See ➔
division1	2	contestant1	0	See ➔
division1	2	contestant2	0	See ➔
division1	2	contestant3	0	See ➔
division1	2	contestant4	0	See ➔

Image: Admin result page


Online Tournament & Assesment System Admin Page Setup Admin Log Out

Back to Results

Results for contestant1

Contest: Contest1 2017

Division: division1

Round: 1

Average Score: 3

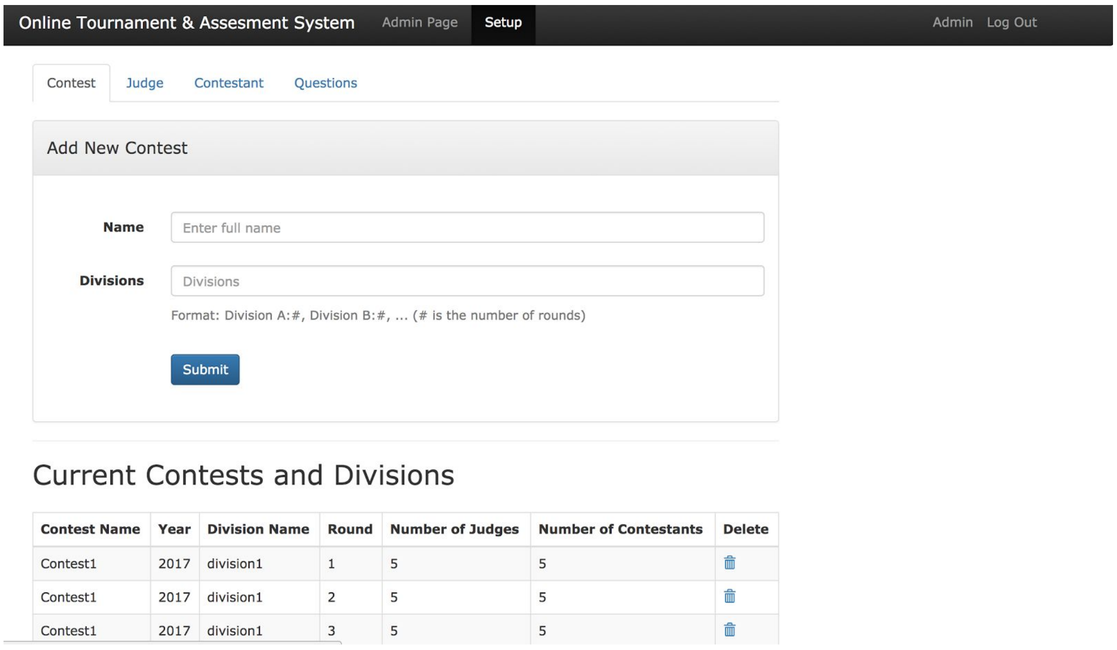
Scoresheet # 1
Judged by: judge1@gmail.com

#	Question	Score
1	[Opening Statement] Contestant Greeting, States Their Name, Contestant Number, and describes the Item for Sale.	1
2	[Style] Delivery, Poise, Eye Contact, and Gestures.	2
3	[Overall Bid Calling] Voice control, Voice Clarity, Volume, Speed, Rhythm, Bid Escalation.	3
4	[Professional Image] Appearance, Manner and Attitude.	4
5	[Salesmanship] Encourages the audience to bid, did bid reach a minimum $50.00	5
6	[Professionalism] Overall impression: Do you believe the Contestant fairly represented the	6

Image: Admin views result of contestant 1

2. Creating a Contest

When setting up a new tournament the first step is to create the Contest. Above there is a tab labeled 'Contest', this links to the contest setup page. This page allows for a new contest to be created, or an old contest to be deleted. Enter in the contest name then the division, **i.e Contest Name: LoneStar 2017 Divisions: Veteran:3, Rookie:2.** will create a Contest of name LoneStar 2017 and 3 Veteran divisions (1,2,3) and 2 Rookie divisions (1,2).


Online Tournament & Assesment System Admin Page Setup Admin Log Out

Contest Judge Contestant Questions

Add New Contest

Name: Enter full name

Divisions: Divisions

Format: Division A:#, Division B:#, ... (# is the number of rounds)

Submit

Current Contests and Divisions

Contest Name	Year	Division Name	Round	Number of Judges	Number of Contestants	Delete
Contest1	2017	division1	1	5	5	🗑
Contest1	2017	division1	2	5	5	🗑
Contest1	2017	division1	3	5	5	🗑

Image: Contest setup page

3. Creating a Judge

Now that the contest is created you can add the judges. Start by clicking the Judge tab above. By selecting the contest in the dropdown menu it is possible to change which contest the judge will be added to. It is assumed all judges will score all contestants.If a judge will be scoring another contest in the future it is possible to reassign the current judges to different contests by using the dropdown bar located within the table at the bottom of this page.

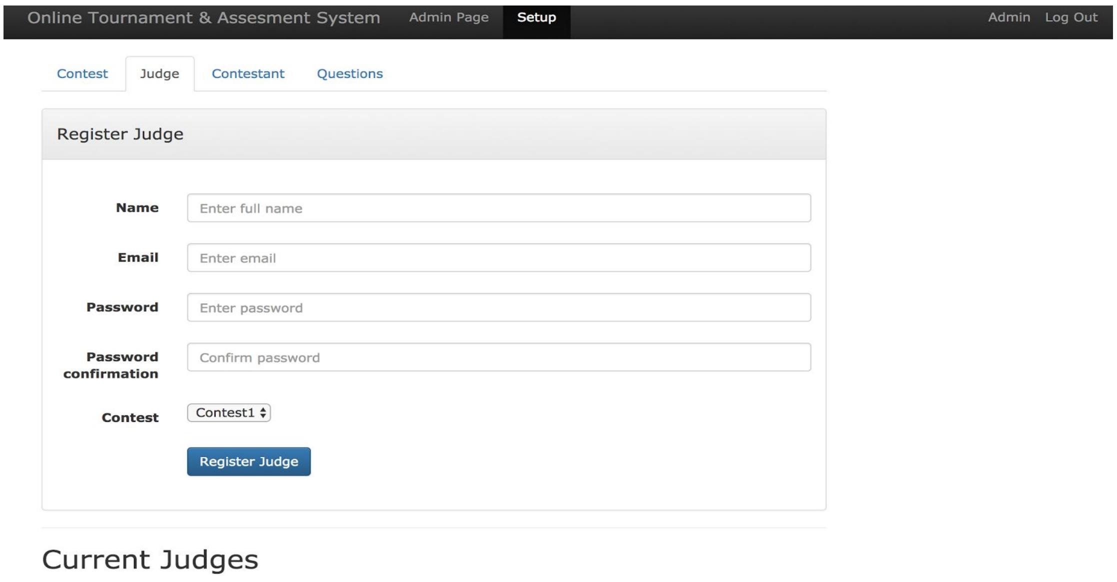
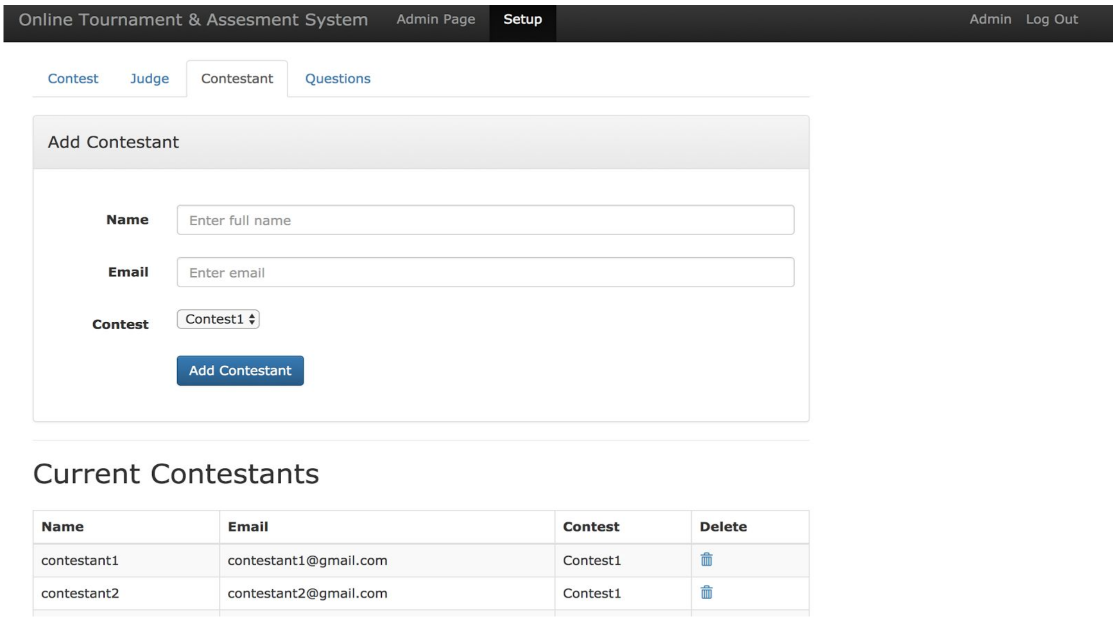


Image: Judge setup page

4. Creating a Contestant

Now that the judges are created you can start adding Contestants. Start by clicking the Contestant tab above. Fill out the name / email then select the correct division and round. Contestants previously created can also be deleted from the table below.

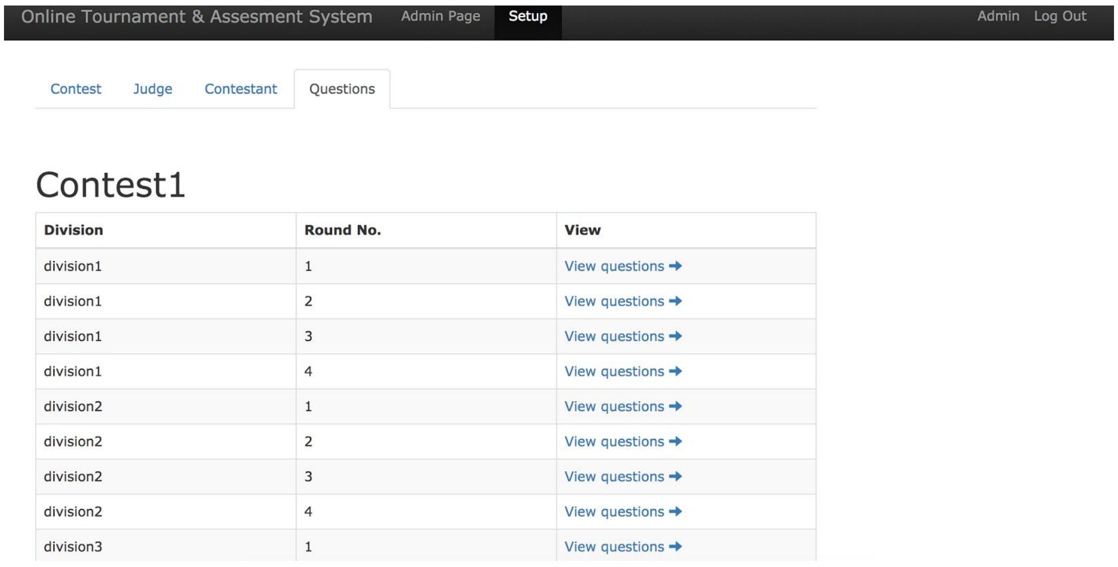


Name	Email	Contest	Delete
contestant1	contestant1@gmail.com	Contest1	🗑
contestant2	contestant2@gmail.com	Contest1	🗑

Image: Contestant setup page

5. Creating a Questions

To create a new question sheet first select the Questions tab above. The current contests split by their divisions will be listed below. By clicking 'Show' the question sheet for the particular division / round is displayed. Here there is the option to delete old questions or create new ones. When creating a question there is field for selecting the type of the question. If the question is meant hold a score from 0-10.0 then the correct type would be 'int'. If the question area is meant to hold comments or categorical data select the 'string' type.



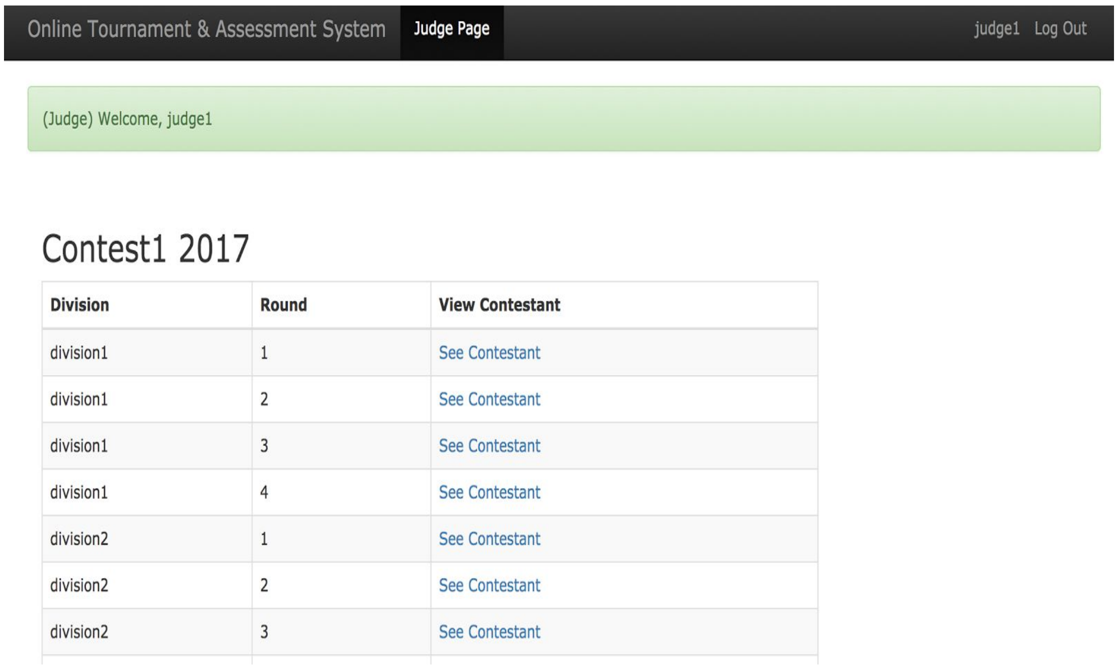
Contest1

Division	Round No.	View
division1	1	View questions ➔
division1	2	View questions ➔
division1	3	View questions ➔
division1	4	View questions ➔
division2	1	View questions ➔
division2	2	View questions ➔
division2	3	View questions ➔
division2	4	View questions ➔
division3	1	View questions ➔

Image: Question setup page

5. Assessing a contestant

Once you login the site as a Judge, you can see the list of contest you have registered and assess each contestants.

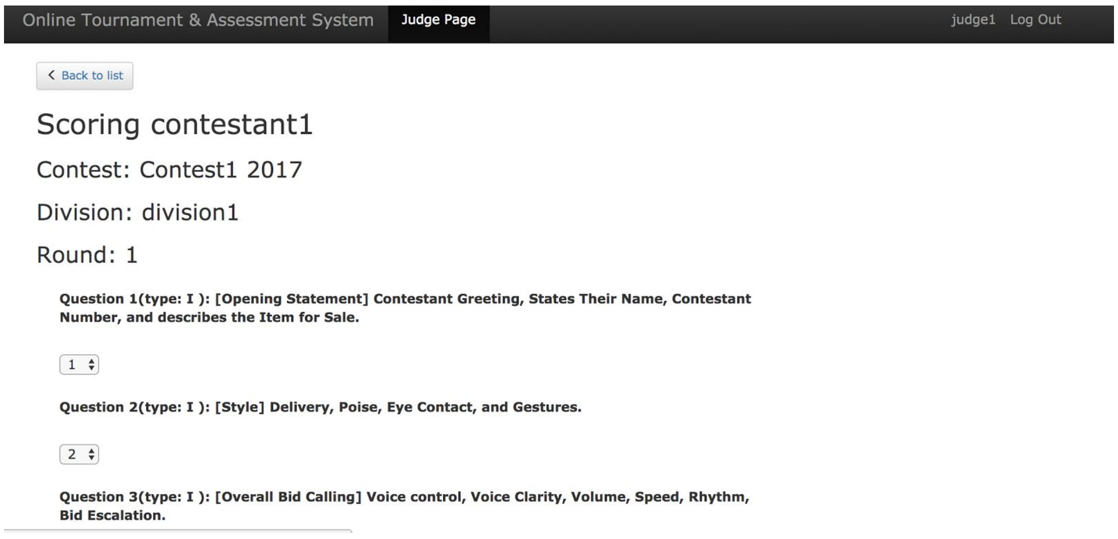


(Judge) Welcome, judge1

Contest1 2017

Division	Round	View Contestant
division1	1	See Contestant
division1	2	See Contestant
division1	3	See Contestant
division1	4	See Contestant
division2	1	See Contestant
division2	2	See Contestant
division2	3	See Contestant

Image: Judge main page


Online Tournament & Assessment System Judge Page judge1 Log Out

Back to list

Scoring contestant1

Contest: Contest1 2017

Division: division1

Round: 1

Question 1(type: I): [Opening Statement] Contestant Greeting, States Their Name, Contestant Number, and describes the Item for Sale.

1

Question 2(type: I): [Style] Delivery, Poise, Eye Contact, and Gestures.

2

Question 3(type: I): [Overall Bid Calling] Voice control, Voice Clarity, Volume, Speed, Rhythm, Bid Escalation.

Image: Judge assessment page

Discussion

For the designing database, there were no changes during the normalization process. I think this is because most attributes are member of super keys which mean there are few non-trivial FDs. Also our system is so clear to describe that there is no chance to have bad FDs.
For the using the database, it's difficult to get familiar with the Ruby on Rails framework. It has Model-View-Controller model, but it makes me confused. Therefore, I have developed most features on the view and controller side. It means I didn't follow the coding convention of Ruby on Rails, and I remain this for the future works.
For the deploying the website, I used Heroku which is a cloud platform as a service that is used as a web application deployment model. It uses Postgres DBMS but I used SQLite3 in the development phase. During the deployment process, I encountered many problems because there are many differences between the two DBMS.

Source Code

There are many source files that automatically generated by Ruby on Rails framework. When you review the source files you can just focus on the folders listed below.

- app/views : html files for our web site
- app/controllers : main ruby controller files that manipulating database
- app/models : active record classes that related to tables (it just exists to follow the Ruby on Rails policies)
- db/seeds.rb : ruby script files to generate default data collection
- db/migrate/*.rb : database scheme files for our database system
- config/routes.rb: routing system files that maps html to controller files

For making user login system, I used sorcery library that provides useful login facilities. Because of it, some database scheme files such as db/migrate/[date]sorcery*.rb are automatically generated by the *sorcery library*. Also 20150213094736_sorcery_core.rb is the *User* relation scheme file and there are more attributes that I didn’t mention in E/R Diagram section because of using sorcery library.