2019

Code Judgement Rubric

ONLY CODE JUDGES SHOULD BE USING THIS VERSION OF THE RUBRIC!

Judge (not shared with students):	________________________________
Judged Team (name or number):	________________________________
Date:	________________________________

Instructions: for each requirement, mark the box that most closely aligns to your impression of the solution. Your judgments here should reflect how well the solution satisfies the directions in the problem statement and the team's presentation as well as your own review of the solution's source code. **Aesthetic appeal of a solution is not more important than functionality.** If a solution is not pretty but otherwise meets a requirement, that counts as reasonably satisfying that requirement. **DO NOT FEEL OBLIGATED TO AWARD POINTS! In the case that a solution attempts to "satisfy" a requirement by making it *look* like it works via the UI/demo code but the required feature is actually non-functional, 0 points should be awarded.**

Solution Functionality Review

Criterion	Score			
	0	1	2	3
	Missing or Completely Non-functional	Attempted But Does Not Satisfy Requirement	Reasonably Satisfies Requirement	Exceeds Requirement Expectations
Requirement 1	A web client must login first before being able to access any page of the site other than the login page.			
	A random first name, last name, and ssn and/or password combination fails to successfully log into the system.			
	The first name, last name, and ssn and/or password combination of an existing user can successfully log into the system and can access other pages.			
Requirement 2 Requirement 6 Requirement 8	The system supports the four types of users: voters, moderators, administrators, and reporters.			
	Voters can vote in elections that they're eligible to vote in (via moderator action). Voters cannot vote in elections that they're not eligible to vote in.			
	Voters can view a complete list of past election results			
	Moderators can add and remove voters from elections.			

	Administrators have moderator privileges for all elections.			
	Administrators can create, modify, and delete elections. For deleting elections, the election should only be able to be deleted if it is “upcoming”.			
	Only Administrators can create new users. When creating a new user, administrators need to provide **at least**: a username, a full name (first and last), an email, phone number, city, state, zip, address, a secure password.			
	Administrators can restrict existing users. A restricted user cannot login. An unrestricted user can login. Administrators cannot be restricted.			
	Administrators can change the type of a user. Administrators should not be able to change the type of other administrators or the root account. Administrators should not be able to make other users into Administrators.			
	Reporters can view the history of past elections.			
	The root user exists and can make other users into Administrators. There is not more than one root user.			
	Elections show their titles, descriptions, available choices, when they open(ed), and when they close(d).			
Requirement 3	When viewing a closed election (e.g. by clicking it), the winning choice is emphasized. Eliminated choices are clearly marked.			
	After voting in an election and then viewing it, the voter’s choices are shown somehow in the UI.			

Requirement 4	Every user type has access to a personal dashboard. At a minimum, the dashboard displays the user’s name, email, last login IP, and a timestamp of their last login time.			
Requirement 5	Users can view a complete history of past elections. There is a method provided to sort the list in some way.			
Requirement 9	Administrators **cannot** edit or delete closed elections. Root **can** edit and delete closed elections, however.			
Requirement 10	Instead of super long lists that require a lot of scrolling, any element where a lot of data is presented should be paginated.			
Requirement 12	Somewhere in the frontend UI, the total number of elections in the system is displayed. This should be true even on the login page.			
Requirement 13	When printing an arbitrary page from the website, it is printer-friendly in that it does not use a lot of ink and the text on the page is legible.			
Requirement 11	When participating in an election, voters can rank their choices from 1 to N.			
	Test election 1 and test election 2 units pass (ask chief judge for test election 1 and 2).			

Instructions: for each of the four criteria below, mark the box that most closely aligns to your impression of the students and their teamwork during the programming portion of the competition. Did their teamwork meet your expectations? **DO NOT FEEL OBLIGATED TO AWARD POINTS!**

Team Performance Review

Criterion	Score			
	0	1	2	3
	Does Not Meet Expectations	Partially Meets Expectations	Meets Expectations	Exceeds Expectations
Engagement	Are all members engaged in solving the problem? One or two students should not be dominating the computers or doing all the work. No students should be idle/sleeping.			
Communication	Do team members communicate well with one another? Members should not be openly arguing, yelling, harassing, belittling, or fighting with one another.			
Planning	Is there evidence that teams planned their solution approach before programming? Are there notes? Did presenting team members speak at all about any strategy or planning phase?			

Instructions: for each of the five criteria below, mark the box that most closely aligns to your impression of the solution's source code after your review. Does their source code meet your expectations of quality? **If you notice source code that seems strangely out of place, overly complex, extremely nonsensical, suspiciously unformatted or very badly organized syntactically, or you otherwise infer that students copy-pasted/plagiarized part or all of their solution's source code from the internet or some illegal source, please inform the chief judge immediately.**

Solution Source Code Review

Criterion	Score			
	0	1	2	3
	Does Not Meet Expectations	Partially Meets	Meets Expectations	Exceeds Expectations
Naming Conventions	Variables should be well-named (e.g. **accountSum** is good while **acsu** is bad). There should be few "magic numbers" floating around. The chosen variable naming scheme should be consistent.			
Organization	The code should be well-organized, with encapsulation and separation of concerns evident where necessary. Similar units of code should be grouped together. If object-oriented language features are used, they should be used properly (SOLID).			
Maintainability and Extensibility	The solution should be implemented such that a future group of programmers would not have to rewrite large portions of the application source code to add new functionality. Source code should be DRY where possible.			
Readability	How easily readable is the source code? Solution code should be as intuitive and self-documenting as possible. Confusing sections should be documented with comments; however, there should not be too many generic/filler/unnecessary comments.			
Efficiency	The implementation should not be too inefficient. Does some piece of code waste cycles looping or otherwise doing something unnecessary? Is the application wasteful with system resources?			

Positive Comments

Constructive Criticism

Two Questions For This Team