2024

PS2 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 given the prompt. 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 <u>not</u> more important than functionality.** If a solution is not pretty but otherwise meets a requirement, that counts as reasonably satisfying that requirement. Keep your expectations high, teams had **a whole month** to work on this. **DO NOT FEEL OBLIGATED TO AWARD HIGH SCORES!** 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, <u>a score of 0 is appropriate.</u> Also, note that you **do not** have to fill this rubric out in order! If you have *any questions at all*, please reach out ASAP!

Solution Functionality Review

[total weight: 100]

Requirement(s) to Reference	Score			
	0	1	2	3
	Missing or Completely Non-functional	Attempted But Does Not Satisfy Requirement	Reasonably Satisfies Requirement	Exceeds Requirement Expectations
Change 1 **[weight: 20]**	**IMPORTANT:** The winner of an election created to use the STAR algorithm as its voting method is correctly determined (as described in Change 1). Please contact NHSCC staff if you need assistance verifying the correctness of this algorithm. **[weight: 20]**			
Change 2 **[weight: 20]**	**IMPORTANT:** The winner of an election created to use the FPTP algorithm as its voting method is correctly determined (as described in Change 1). Please contact NHSCC staff if you need assistance verifying the correctness of this algorithm. **[weight: 20]**			
Change 3 **[weight: 20]**	**IMPORTANT:** In the History view, you have the option to see how the election would have gone had a different algorithm been used to determine the winner. You should be able to select from IRV, CPL, STAR, and FPTP algorithms and the winning option of the election (along with any other relevant data) should update accordingly. **Sometimes, however, it's not possible to determine who would have won under a different voting algorithm.** When this is the case, this should be indicated in the UI. **[weight: 20]**			
Change 4 **[weight: 10]**	When viewing an election in the History or Election view, more detailed information is shown about the election (as described in Change 4). **[weight: 10]**			

Change 5 **[weight: 5]**	Administrators can choose to make elections public. Public elections can be viewed by anyone regardless of if they are logged in or not (as described in Change 5). **[weight: 5]**			
Change 6 **[weight: 5]**	The super administrator user can activate “maintenance mode” on the app, which makes the entire system read-only except to administrators (as described in Change 6). **[weight: 5]**			
Change 7 **[weight: 5]**	Elections that a voter is eligible to vote in that are closing in an hour or less are highlighted in the user dashboard somehow. **[weight: 5]**			
Change 8 **[weight: 5]**	Elections in the Election View show an “audit log” as described in Change 8. **[weight: 5]**			
Change 9 **[weight: 5]**	Administrators can view the real names (or, if not available, the voter IDs) of voters where relevant in the History and Election views. **[weight: 5]**			
Change 10 **[weight: 5]**	Timestamps, where shown, are human friendly and easily read. For example: “20 minutes ago,” “9 days ago,” “2 days from now,” etc. **[weight: 5]**			

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 of their solution's source code from the internet or some illegal source, inform the chief judge immediately.** Also, note that you **do not** have to fill this rubric out in order! If you have *any questions at all*, please reach out ASAP!

Solution Source Code Review

[total weight: 100]

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 and self-documenting with additional commentary where relevant. **[weight: 5]**			
Organization	The code should be well-organized, with a consistent logical file/directory layout, encapsulation, and proper separation of concerns. Similar units of code should be grouped together in a logical way. If object-oriented language features are used, they should be used properly (i.e. SOLID). **[weight: 20]**			
Security	The solution should use modern software engineering practices that protect from common XSS, SQL injection, and other security vulnerabilities. Specifically: form input should be properly sanitized and should not be vulnerable to SQL injection attacks (example). User-generated outputs should not be vulnerable to XSS attacks (example). **[weight: 20]**			
Maintainability and Extensibility	The solution should be implemented such that a future group of programmers would not have to rewrite large portions of the source code to add new functionality. Source code should be DRY where reasonable. Pay special care to inflexible and hardcoded values. **[weight: 20]**			
Readability	How easily readable is the source code? Did you have a hard time going through it?			

	Solution code should be as intuitive and self-documenting as possible. Confusing sections should be documented with comments. There should be few “WTF?!” moments. **[weight: 20]**			
Efficiency	Any implementations 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? Does it make many more network requests than necessary? **[weight: 15]**			

Instructions: please provide any additional comments you have below. Note that your comments are shared directly with the students. Please be thorough, encouraging, and fair.

Positive Comments

Please share any positive comments you have for this team.

Constructive Criticism

Please share any unanswered concerns or comments you have for this team.

Two Questions For This Team

Please share the two most pressing questions you'd like answered about this team's solution.

Shared Thoughts (visible to all judges)

Please add any thoughts or concerns you believe other judges should be made aware of.