2024

PS1 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 Meet My Expectations	Meets My Expectations	Exceeds My Expectations
Requirement 1 Requirement 6 **[weight: 9]**	You **do not have to** login before being able to view the Auth view. **[weight: 1]**			
	You **must** login before being able to view the Election, Dashboard, or History views. You can login as the super administrator user. **[weight: 1]**			
	Providing random login credentials **does not** allow you to log into the system successfully. **[weight: 3]**			
	Providing the correct login credentials **does** allow you to log into the system successfully. **[weight: 1]**			
	When you are already logged in, you should not be able to create a new account using the same URL/link used to create accounts when you're not logged in. **[weight: 1]**			

	You can create a brand new account via a registration feature; you can login successfully using the account you just created. **[weight: 2]**			
Requirement 2 Requirement 3 **[weight: 15]**	The Election view (the page that displays information about an election) is pleasant to look at, convincing, and ergonomic. It is pretty simple to figure out and use. It has the appropriate components spelled out in the requirement text and provides the functionality spelled out at each bullet point. **[weight: 5]**			
	After casting a vote, voters have five minutes or until the election closes (whichever is sooner) to change or remove their vote. **[weight: 3]**			
	Closed elections are immutable as described in Requirement 2. **[weight: 2]**			
	A voter cannot access elections to which they have not been assigned. Similarly, moderators cannot moderate elections to which they have not been assigned. **[weight: 2]**			
	The winning option is emphasized in the UI if the election is closed. The current user's vote, if they are a voter, is emphasized in the UI. **[weight: 3]**			
Requirement 4 **[weight: 15]**	The Dashboard view (the page users are redirected to upon login) is pleasant to look at, elegant, and ergonomic. It is pretty simple to figure out and use. It has the appropriate components spelled out in the requirement text and provides the functionality spelled out at each bullet point. **[weight: 5]**			
	Deleted elections do not show up in this view unless the user is an administrator. **[weight: 1]**			

	Users can edit their own personal information from this view. **[weight: 1]**			
	Administrators can assign voters, reporters, and moderators to any election. They can create new account types. They can view and update the information of any other non-administrator user. They can create, update, and delete any election in the system (as described by the problem statement). When creating an election, they can choose the voting method used to determine that election's winner (IRV or CPL). **[weight: 2]**			
	Voters can see the most recent open elections, closed elections they participated in, and upcoming elections they've been assigned to. **[weight: 2]**			
	Moderators can assign voters and reporters to elections to which the moderator has been assigned to oversee. **[weight: 2]**			
	Reporters can see several of the most recent closed elections to which they've been assigned. **[weight: 2]**			
Requirement 5 **[weight: 12]**	The History view (the page that lists all past elections in the system) is pleasant to look at, elegant, and ergonomic. It is pretty simple to figure out and use. It has the appropriate components spelled out in the requirement text and provides the functionality spelled out at each bullet point. **[weight: 5]**			
	Deleted elections are not shown unless the user is an administrator. **[weight: 2]**			
	Elections are sorted in descending order by their closing time. **[weight: 2]**			

	You can sort results by their: title, creation time, opening time, closing time (default). **[weight: 3]**			
Requirement 6 **[weight: 10]**	The Auth view (the page that allows you to login and/or register) is pleasant to look at, elegant, and ergonomic. It is pretty simple to figure out and use. It has the appropriate components spelled out in the requirement text and provides the functionality spelled out at each bullet point. **[weight: 5]**			
	You cannot use the same email address or username to register more than one user. That is: usernames and email addresses must be unique. **[weight: 1]**			
	Registration requires a CAPTCHA. Users are restricted from logging in for 1 hour after 3 failed login attempts. Users can see how many attempts they have remaining. **[weight: 2]**			
	When logging in, you can use “remember me” functionality to stay logged in. That is: you can login using “remember me,” completely exit the browser, reopen it, and still be logged in afterwards. When you disable “remember me” functionality (i.e. log out, then login again without using “remember me”) and you exit and relaunch the browser the same way, you should no longer be logged in. **NOTE: ensure the browser’s session restoration “features” are disabled if you’re having trouble with this one.** **[weight: 2]**			
Requirement 7 **[weight: 2]**	You can use the password recovery feature to regain access to an account without the password. This should be available through a “forgot password” link of some kind when visiting the Auth view. Instructions on how recovery emails are simulated can be found in the team’s submission documents. **You MUST be forced to set a new password and should NEVER be shown your old “forgotten” password which would be a dire security failure.** **[weight: 2]**			
Requirement 8 **[weight: 2]**	The app has a navigation element with the BDPA logo, app title, and the total number of elections, open elections, and closed elections in the system. **[weight: 2]**			

Requirement 9 **[weight: 15]**	**IMPORTANT:** The winner of an election is correctly determined by either the IRV or Copeland algorithm (as described in Requirement 9) depending on which algorithm the administrator chose when creating the election. Please contact NHSCC staff if you need assistance verifying the correctness of these algorithms. **[weight: 15]**			
Requirement 10 **[weight: 10]**	**IMPORTANT:** When looking at the Election view, new/updated votes, status updates (such as when the election opens/closes), the calculated winning option, and any other relevant information must *eventually* (give it 60 seconds) update on the screen automatically when that data changes in the backend. This must happen without you having to press a refresh button or take any other actions. You can test this by creating multiple users across multiple tabs/browsers that are viewing and interacting with the same election; you should eventually see the changes from one tab/browser reflected in the other. **[weight: 10]**			
Requirement 12 **[weight: 1]**	When viewing lists of users, elections, etc., all of these results are *paginated* in an aesthetically pleasing way. See the text of the problem statement for more details. **[weight: 1]**			
Requirement 14 **[weight: 1]**	The app fails gracefully when errors happen. You don't lose all your progress in the registration form when you press submit with invalid data, for instance. When loading a lot of data, the app does not freeze or present a broken UI but instead shows a spinner icon or some other loading indicator while waiting. You are never waiting too long for something to happen in response to user input. The app never "breaks," never displays internal error messages or warnings anywhere on the page, the backend server hosting the solution never crashes, never any HTTP 500 errors, etc. **[weight: 1]**			
Requirement 15 **[weight: 8]**	The app looks good when you view it on a small smartphone-sized screen. **One way to simulate this is by using your mouse to drag the right edge of the browser window to the left until it cannot shrink anymore. This shrunken column represents the size of a phone screen.** Is the app still usable in this state? Is it visually pleasant? **[weight: 2]**			
	The app looks good when you view it on a medium tablet-sized screen. **To simulate this, make your browser window half the size of your screen (try windows key + left arrow). This simulates the size of a tablet screen.** Is the app still usable? Is it visually pleasant? **[weight: 1]**			

	The app looks good when you view it on a normal desktop/laptop screen. **Ensure the browser window is maximized.** Is the app still usable? Is it visually pleasant? **[weight: 2]**			
	Overall, you feel this solution is easy to use. **[weight: 1]**			
	Overall, you feel this solution is beautiful to look at. **[weight: 1]**			
	Overall, you feel this solution functioned well. App functionality that worked fine when the browser window was maximized did not suddenly break when you shrunk the browser to simulate a smartphone or a tablet. **[weight: 1]**			

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.