

# Take-home Assessment - Rubric

Category / Description	Needs improvement (below 40%)	Satisfactory (40-60%)	Good (60-75%)	Very Good (Top 25%)	Excellent (Top 10%)	Outstanding (Top 5%)
	What are we looking for?					
Correctness	<ul style="list-style-type: none"> <li>• Passes all of our tests, and shows great attention to detail</li> <li>• Goes above and beyond the requirements, such as handling validation of improper inputs, or completing bonus requirements and stretch goals (like deploying an application).</li> </ul>					
Code Organization, Readability & Maintainability	<ul style="list-style-type: none"> <li>• Code utilizes a consistent design pattern. The code is well organized and would be easy to extend. Design made with consideration to potential future features.</li> <li>• Code is easy to read - proper indentation, vertical spacing, named constants (no magic numbers), big picture comments, and meaningful variable names. You can investigate using a linter to help make your code properly formatted.</li> <li>• There are small amounts of repeated code, and is well factored into functions. Related items are grouped into classes (or something equivalent like structs or objects).</li> </ul>					
Code Performance - Efficiency, Data Structures & Algorithms	<ul style="list-style-type: none"> <li>• Code that is performant (has a good time and space complexity for the problem given). Avoids unnecessary nested for loops, and uses the appropriate data structures and algorithms.</li> <li>• Efficient code that reduces unnecessary code.</li> <li>• Implements additional features to improve code performance such as (but not limited to) caching, parallel API requests, reducing re-rendering on the front-end, memoization, and other code performance techniques.</li> </ul>					
Best Practices & Proficiency in Language/ Framework	<ul style="list-style-type: none"> <li>• Utilizes great programming practices such proper unit testing, reduced warnings, and defensive coding strategies (validation, asserts).</li> <li>• Demonstrate a good grasp of the language and/or framework. Avoids major anti-patterns in the project (using let vs. const in Javascript, modifying the DOM in React, etc.)</li> <li>• Utilizes the most advanced features in the framework (for example using functional components, Hooks in React).</li> <li>• Properly replicates the styling spec provided, for a front end assessment.</li> </ul>					

*\* A note about plagiarism: any submission that copies another person's assessment posted online will be automatically disqualified.*