Workflow

🌻 Dear Ladies and Gentlemen.

Your task will be graded using this document (you should know its content by heart and be able to cite its rubrics with your eyes closed 😆)

[Perfect Assertions] QC Spec Doc 4 Attempters

🚨 Important Warning Regarding Prefilled Tasks 🚨

You may see filled-out tasks even as an attempter

- **How to identify**: The task is already completed / the golden code response is already provided
- **Workflow**: Leverage the feedback provided by the reviewer and make sure that on the next submission, your task is acceptable.

#	Step	Time Allocated	Step Summary
1	Review the Task Setup & Prompt Validation	5-10m	⚫**Part 1a: Review Task Instructions ->** Carefully read the instructions and workflow outlined at the beginning of the task. Review the file name ({{skeleton_code_fn}}.[file_extension]), programming language, and time constraints. ⚫**Part 1b: Validate Seeded Prompt ->** Determine if the seeded prompt is correct and aligns with your expertise. Verify familiarity with the problem. If unfamiliar, skip the task. ⚫**Part 1c: Create Paraphrased Prompt ->** Read the seeded prompt and paraphrase it without changing the pragmatic meaning. Only change the semantics while preserving the same length. This becomes your target prompt throughout the task.

			For more information, see [to be added]
2	Model Failure Validation (First Workspace)	15-30m	●**Part 2a: Open First Workspace ->** Access the first read-only workspace containing the preloaded seeded code response and preloaded unit test suite. Press the blue [Open IDE] button to open the Sphere Engine. ●**Part 2b: Execute & Analyze Failure ->** Run the seeded code response against the unit tests to validate if it contains a **true model failure**. Analyze the failure output to confirm it fails at least one valid unit test. ●**Part 2c: Task Continuation Decision ->** If no model failure is established, the task is cut short and ends early. If true model failure is confirmed, proceed to the next stage. *Note: For a true model failure, the model must fail at least one valid test. Note that some tests {y, z} may be faulty while {x} is correct - if the model fails the correct test {x}, it still counts as a true failure.*
3	Model Guidance Creation	5-10m	●**Part 3a: Analyze Root Cause ->** Examine the specific error messages and failure points in the seeded code response to understand why it's failing the valid unit tests. ●**Part 3b: Write Model Guidance ->** Create a coherent model guidance consisting of at least two full sentences that addresses the identified failure and guides the model toward a correct solution. Focus on specific technical issues like type errors, logic flaws, or implementation mistakes. **For more information, see [to be added]**
4	[CRUCIAL] Unit Test Evaluation (5 Criteria Assessment)	15m	●**Part 4a:** Evaluate the unit tests against all five criteria linked here: [Five Golden Unit Test Criteria] ●**Part 4b**: Mark Yes/No for each criterion. Note that if you mark "No" for any unit test criteria, you must rewrite the unit tests in 'test.[file_extension]' in the second sphere engine.
5	Golden Unit Test Creation (Second Workspace)	30m	●**Part 5a: Open Second Workspace ->** Access the second editable workspace by pressing the blue [Open IDE] button. This workspace allows you to create your golden code response and golden unit test suite. ●**Part 5b: Modify 'test' File ->** Directly modify the `test.[file_extension]` file preloaded with the same seeded unit tests using your evaluation findings from Stage 4. ●**Part 5c: Test Main Functions ->** Only test the main functions already present in the skeleton code. Your final code response may contain additional helper functions, but it's crucial we only test the functions with the function signatures present in the skeleton code. ●**Part 5d: Sanity check ->** Circle back and truly verify that you have a golden unit test suite that validates all 5 criteria/

6	Golden Code Response Generation	15-30m	⚫**Part 6a: Generate Initial Response Using Fabricator Prompt ->** Generate an LLM response from Deepseek R1 0528 by providing it with the below 5 items and asking for the correct implementation. 1 Paraphrased Prompt 2 Skeleton Code 3 Seeded Response 4 Golden Unit Test Suite 5 Failure Output Seeded Response Against Golden Unit Test Suite 6 Model Guidance Note: For an example fabricator prompt, see this section of the project instructions: [Read Third] Golden Response ⚫**Part 6b: Extract & Evaluate Code ->** Extract only the code block from the Deepseek R1 0528 response and evaluate if it passes your golden unit tests. If you notice any mistakes in your golden unit test suite, fix them immediately, directly in the test file ⚫**Part 6c: Iterate Until Success ->** If the code doesn't pass all golden unit tests, refine the prompt or guidance and regenerate until achieving a passing response. **CRUCIAL TO NOT GET REMOVED FROM THE PROJECT - SIGNIFICANTLY REFACTOR THE LLM CODE AND MAKE IT YOUR OWN** ⚫**Part 6d: Implement Significant Improvements ->** Use the Deepseek R1 0528 response as starter code, but make substantial modifications and improvements. The final code cannot be identical to the generated response. ⚫**Part 6e: Create Golden Code Response ->** Develop the final golden code response that implements all skeleton code functions, includes necessary helper functions, and passes all golden unit tests. **CRUCIAL TO NOT GET REMOVED FROM THE PROJECT - SIGNIFICANTLY REFACTOR THE LLM CODE AND MAKE IT YOUR OWN** Warnings • NEVER modify the file name of both test.[file_extension] and [skeleton_code_fn].[file_extension]. The file name of the skeleton code is predetermined at the start of the task and provided at the top

			• NEVER test functions not present in the skeleton code. If the skeleton code contains two functions, you'll test those two functions in your unit tests using the same function signatures.
7	[🚨CRUCIAL] Final Integration Testing	5-10m	⚫**Part 7a: Final Integration Testing ->** Run the golden code response against the complete golden unit test suite to ensure 100% test passage. ⚫**Part 7b: Validate Feature Completeness ->** Confirm the golden code response fully satisfies the original prompt requirements and implements all requested functionality. ⚫**Part 7c: Performance & Efficiency Check ->** Verify the code is not only correct but also efficient and follows best practices for the given programming language. ⚫**Part 7d: Code Quality Review ->** Ensure the code is clean, well-structured, properly commented, and maintainable. ⚫**Part 7e: Unit Test Quality Review ->** Circle back once more and take a look at your Golden Unit Test Suite. **Leave no stone unturned.**
8	[🚨Crucial] Quality Control & Submission	5-10m	***STOP BEFORE SUBMISSION*** ⚫**Part 8a: Quality Control Evaluation ->** Use the Perfect Assertions QC Spec Doc to evaluate your work, ensuring each rubric scores at least 4/5. Address any areas scoring 3 or lower. [Perfect Assertions] QC Spec Doc 4 Attempters ⚫**Part 8b: Final Code Preparation ->** Copy the golden code response content exactly as it appears in the Sphere Engine files to the final solution field, ensuring no ``` marks are included. **Part 8d: Peer Review & Submission - Conduct final peer review to confirm the task achieves 4/5 quality standards across all dimensions before submission.**
		Min: 1.5 Max: 2.5h	