Coders Candidates Assessment Information

Contact email for escalations acquisition-latam@outlier.ai
Latam Coders Process Slides: Click Here
Support Webinars: Click Here

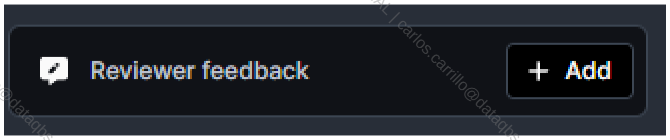
Assessment Task (Average duration: 2-3 hours)

The Assessment task is designed to **simulate the reviewer experience, allowing candidates to demonstrate their ability to evaluate an attempter's work and apply any necessary modifications for this task to have the expected quality**.

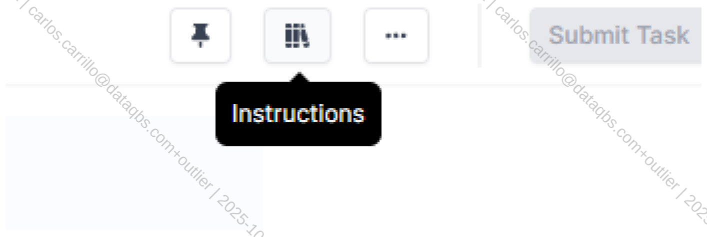
The attempter is responsible for doing the task from scratch, strictly following the project documentation. **The reviewer** is in charge of reviewing the completed work and **making any necessary adjustments to ensure it aligns with the project documentation.**

The other documents contain information from both the attempter and the reviewer's perspectives, which will help you understand both points of view and complete the assessment efficiently.

IMPORTANT CONSIDERATIONS

- **The task time is shown as 24 hr, however it is unlimited, even if the time runs out, the timer will reset and your progress will be saved, this means that you can finish it at different times.**
- **The SBQ button or anything related to it is not used on this Assessment Task**
- **The time it takes you to complete your assessment does not affect your score.**
- **Do not write in the "Reviewer Feedback" sections, as this is not taken into account for the score.**
    - 

- **To check the assessment instructions, you can click on the book icon in the upper right corner to open them in a new tab**



-

Pre-filled Attempter's Submission:

The Assessment task would **provide a pre-filled submission that mimics what an attempter would submit**. This would include:

- A prompt with its tags (**category**, **sub-category**, **difficulty**).
- Two pre-filled model responses to the prompt. **(Click on one response to expand it and see what the attempter has done with the task).**
- The attempter's ratings for each response across all dimensions (**Instruction Following**, **Accuracy**, **Optimality**, **Presentation**, **Up-to-Date**).
- The attempter's ranking of the responses with justification.
- The attempter's rewritten version of the preferred response.
- Any relevant code execution documentation (if applicable).

Reviewer's Role:

Candidates would act as reviewers and **assess the attempter's submission.** Their task would be to:

- Evaluate the prompt for clarity and quality.
- Verify the attempter's ratings for each model response against the provided guidelines and rubrics **and make changes if needed**.
- Assess the attempter's ranking and justification for accuracy and reasoning.
- Review the rewritten response for correctness, completeness, improvement **and make changes if needed**.

Using Provided Documentation:

- Candidates would need to refer to the other tabs or documentation that outline the attempter and reviewer processes.
- These documents would provide guidelines, rubrics, and examples to help candidates make informed judgments.
- Candidates must demonstrate an understanding of these documents by applying them correctly in their review.

Assessment Criteria:

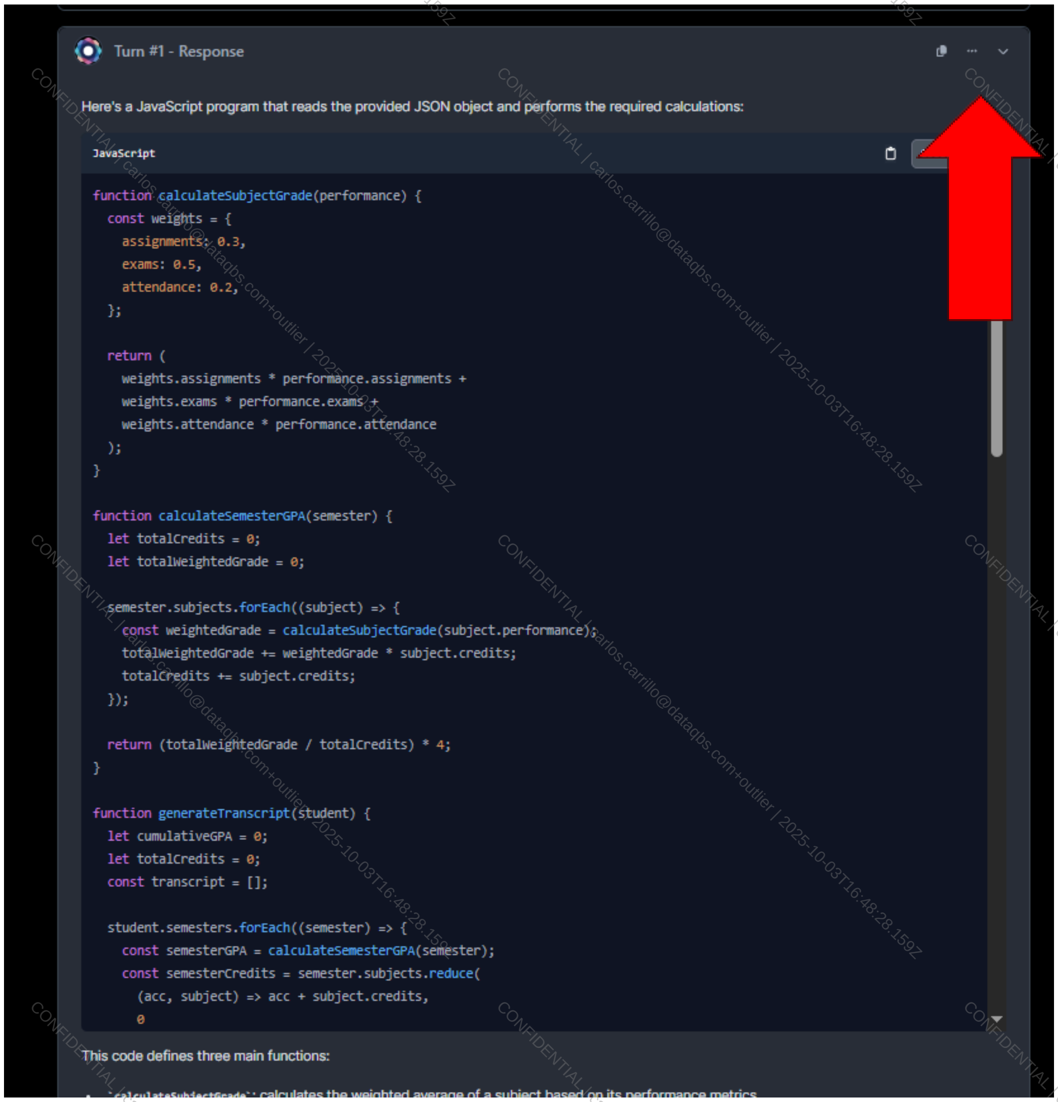
The Assessment task would be evaluated based on:

- **Accuracy of the reviewer's evaluation of the applicant's work, changes made and descriptions.**
- **Correct application of the provided guidelines and rubrics.**

Expected Deliverable

The reviewed submission with the correct adjustments (if needed) to make it good quality. To evaluate, we need to click on the first response (or first turn) to view it. Then, we must assess it according to the rubrics provided in the instructions.

Click on "Turn #1- Response" to get access to the attempter's submission.



Here's a JavaScript program that reads the provided JSON object and performs the required calculations:

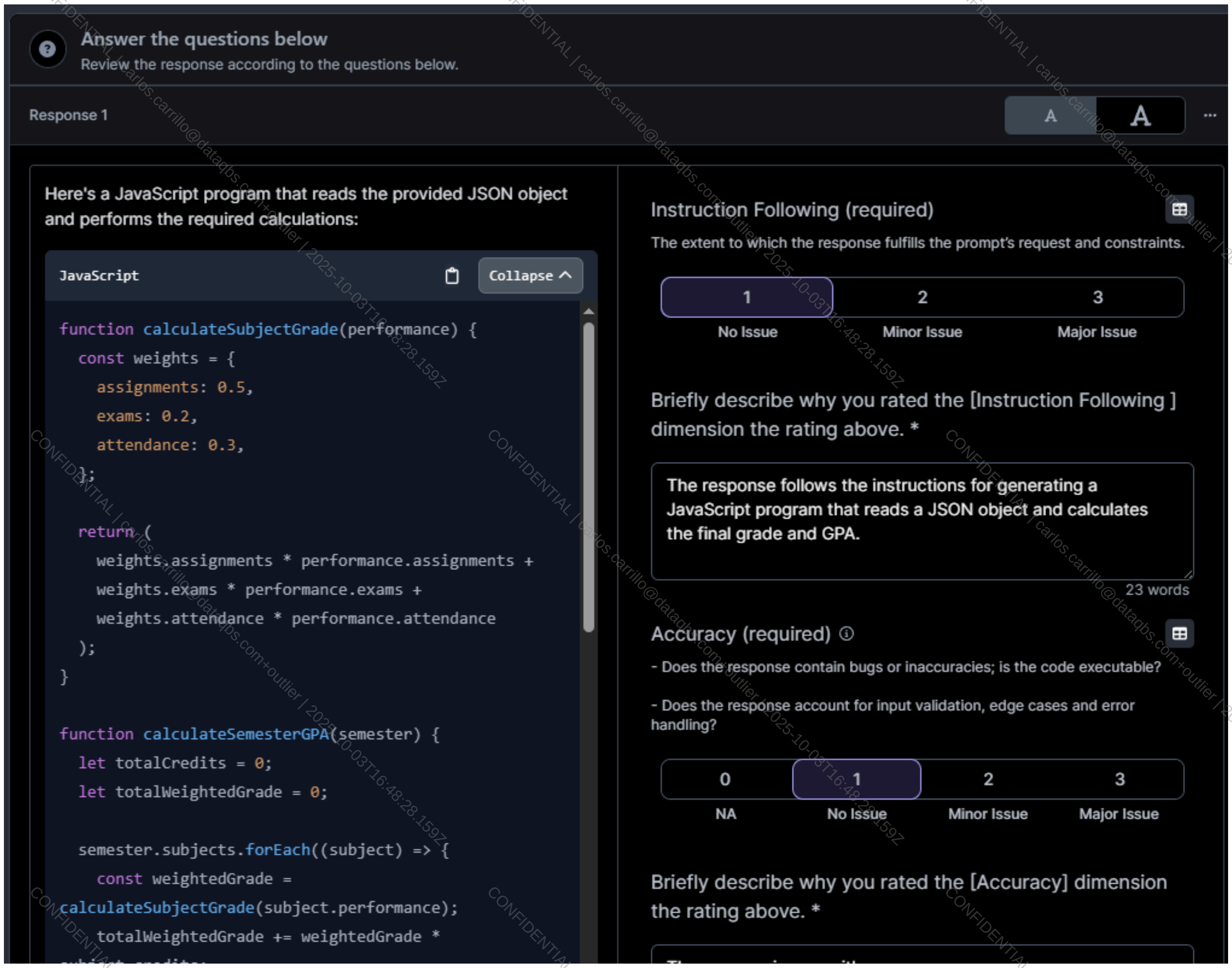
```javascript
function calculateSubjectGrade(performance) {
  const weights = {
    assignments: 0.3,
    exams: 0.5,
    attendance: 0.2,
  };

  return (
    weights.assignments * performance.assignments +
    weights.exams * performance.exams +
    weights.attendance * performance.attendance
  );
}

function calculateSemesterGPA(semester) {
  let totalCredits = 0;
  let totalWeightedGrade = 0;

  semester.subjects.forEach((subject) => {
    const weightedGrade = calculateSubjectGrade(subject.performance);
    totalWeightedGrade += weightedGrade * subject.credits;
    totalCredits += subject.credits;
  });

  return (totalWeightedGrade / totalCredits) * 4;
}

function generateTranscript(student) {
  let cumulativeGPA = 0;
  let totalCredits = 0;
  const transcript = [];

  student.semesters.forEach((semester) => {
    const semesterGPA = calculateSemesterGPA(semester);
    const semesterCredits = semester.subjects.reduce(
      (acc, subject) => acc + subject.credits,
      0
```

This code defines three main functions:

- `calculateSubjectGrade`: calculates the weighted average of a subject based on its performance metrics

Once opened, refer to the "Attempter Instructions" document, to learn how to do the task from the attempter's perspective, and to change ratings and descriptions if they do not match the guidelines.

[Reference video]()



Answer the questions below

Review the response according to the questions below.

Response 1

Here's a JavaScript program that reads the provided JSON object and performs the required calculations:

```javascript
function calculateSubjectGrade(performance) {
  const weights = {
    assignments: 0.5,
    exams: 0.2,
    attendance: 0.3,
  };

  return (
    weights.assignments * performance.assignments +
    weights.exams * performance.exams +
    weights.attendance * performance.attendance
  );
}

function calculateSemesterGPA(semester) {
  let totalCredits = 0;
  let totalWeightedGrade = 0;

  semester.subjects.forEach((subject) => {
    const weightedGrade =
calculateSubjectGrade(subject.performance);
    totalWeightedGrade += weightedGrade *
```

Instruction Following (required)

The extent to which the response fulfills the prompt's request and constraints.

[✓ 1]	2	3
No Issue	Minor Issue	Major Issue

Briefly describe why you rated the [Instruction Following] dimension the rating above. *

The response follows the instructions for generating a JavaScript program that reads a JSON object and calculates the final grade and GPA.

23 words

Accuracy (required)

- Does the response contain bugs or inaccuracies; is the code executable?

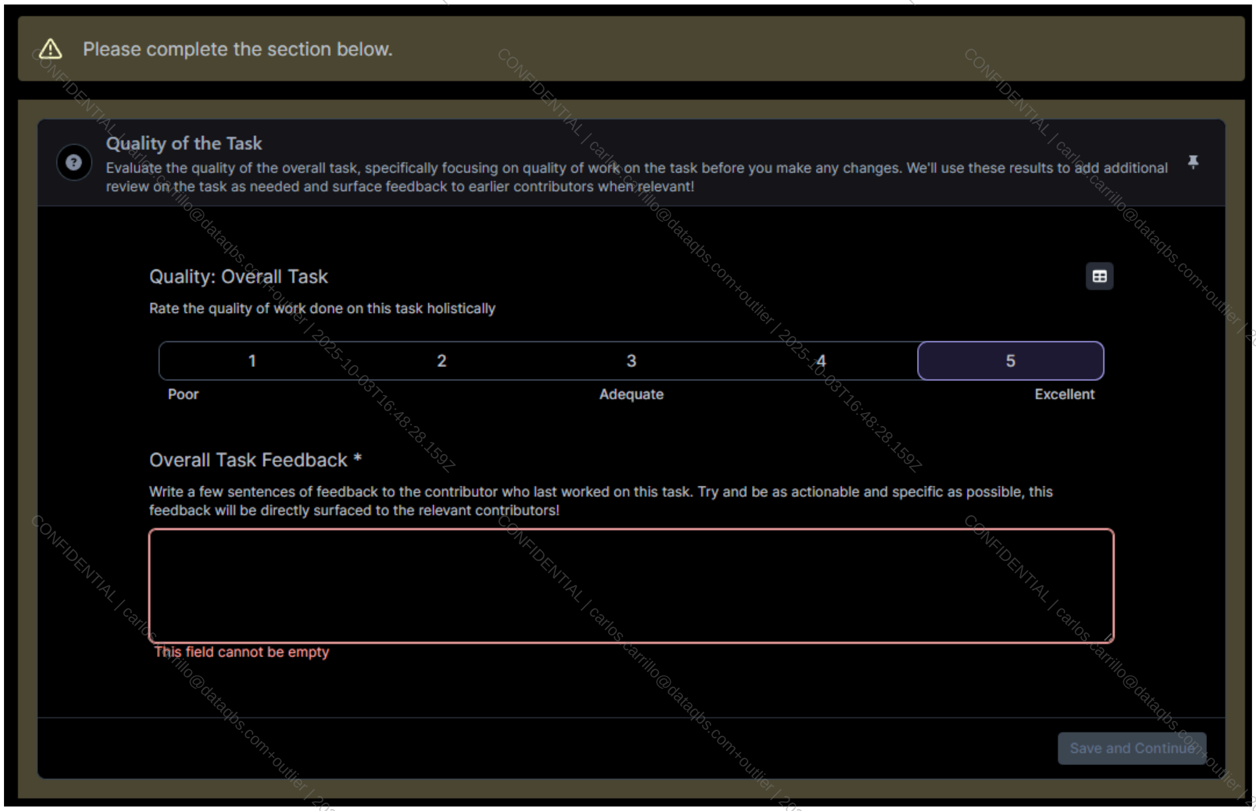
- Does the response account for input validation, edge cases and error handling?

0	[✓ 1]	2	3
NA	No Issue	Minor Issue	Major Issue

Briefly describe why you rated the [Accuracy] dimension the rating above. *

After reviewing all of the attempter's submissions according to the guidelines, we will need to evaluate the submission.

Answer the last question at the end of the task with its corresponding evaluation **and submit the task**. **(check the document "Reviewer Checklist - How to score a Task")**


Please complete the section below.

Quality of the Task

Evaluate the quality of the overall task, specifically focusing on quality of work on the task before you make any changes. We'll use these results to add additional review on the task as needed and surface feedback to earlier contributors when relevant!

Quality: Overall Task

Rate the quality of work done on this task holistically

1 2 3 4 [✓ 5]

Poor · Adequate · Excellent

Overall Task Feedback *

Write a few sentences of feedback to the contributor who last worked on this task. Try and be as actionable and specific as possible, this feedback will be directly surfaced to the relevant contributors!

This field cannot be empty