|  |
| --- |
| Cute yellow robot |
| AI Written Code Assessment  A1\_T2 |
| |  |  | | --- | --- | | Loai Hataba 20230553  Abdullah Mohammed 2023231  Hossam Abdelaziz 20230121 | CS213-OOP | |  |  | |

**AI-Written Code Assessment**

**Introduction**

* **AI Models Used**: Chat-GPT (4o mini) / Claude (3.5 Sonnet)
* **Initial Prompt**: Both Ai models were prompted with the same prompt

***“You’re a senior programmer with a lot of knowledge in C++, I want you to complete this class … (provide the header file)”***

One of the most important aspects of a good prompt is the persona, where you tell the ai model who it should be so it has a clearer path of where to search and which information to provide; so by telling the models that they’re programmers who are knowledgeable in C++ that would potentially help the results be better than just asking them to complete the class.

* **Modifications/Reprompting**: Both Models have been prompted more than once to get somewhat of an acceptable result that matches the desired outcome.

**1. Correctness**

**Results:**

* **Chat-GPT:** 
  + Overall, the code was acceptable
* Does it solve the problem as expected?

**Key Considerations:**

* **Functionality**: Verify if the code works as intended by running different test cases, including edge cases.
* **Error Handling**: Assess whether the code handles possible errors gracefully (e.g., invalid inputs, exceptions).
* **Edge Cases**: Does the code cover extreme or uncommon situations?

**Example/Observations:**

* (Provide specific examples of tests you ran, and how the code performed.)

**2. Efficiency**

**Overview:**

* How well does the code perform in terms of time and space complexity?

**Key Considerations:**

* **Time Complexity**: Evaluate how the code performs as the input size grows (e.g., O(n), O(log n)).
* **Space Complexity**: Consider how much memory the code uses, especially with large inputs.
* **Optimization**: Are there unnecessary operations or steps that could be improved?

**Example/Observations:**

* (Provide information on the time/space complexity and potential optimizations.)

**3. Elegance**

**Overview:**

* How clear and simple is the logic of the code?

**Key Considerations:**

* **Conciseness**: Is the code unnecessarily verbose, or does it achieve its goal with minimal, clear lines of code?
* **Algorithm Choice**: Has the AI chosen an efficient and appropriate algorithm for the task?
* **Structure**: Does the flow of the program make sense and is it easy to follow?

**Example/Observations:**

* (Highlight areas where the code could be made more elegant, such as refactoring overly complex blocks into simpler ones.)

**4. Cleanliness**

**Overview:**

* Is the code easy to read and maintain?

**Key Considerations:**

* **Readability**: Assess variable names, comments, and structure for clarity.
* **Comments**: Are there sufficient comments to explain non-trivial parts of the code?
* **Formatting**: Is the code properly formatted with consistent indentation, spacing, and naming conventions?
* **Modularity**: Does the code follow principles like DRY (Don't Repeat Yourself) and modularity, separating concerns into functions or classes?

**Example/Observations:**

* (Note specific areas where the code is hard to read, poorly commented, or not modular enough.)

**5. Overall Recommendations**

**Summary:**

* Provide an overall evaluation of the AI-written code, balancing all the above aspects.

**Suggestions for Improvement:**

* Highlight areas for refinement and potential refactoring opportunities.

Links:

-Chat-GPT conversation 1(class implementation): <https://chatgpt.com/share/670fd569-b568-8011-8501-c96948df8a49>

-Chat-GPT conversation 2(menu): <https://chatgpt.com/share/670fd5cd-ee64-8011-98e0-78e1dd90b984>