**Code Review Checklist**

Description: Use this document as a guide to evaluate the quality, readability, and structure of the code being reviewed. Enter a score from 1-5 (worst to best) next to each point and leave any sections blank if they do not apply. Be sure to sign your initials at the end of the document once the review is complete.

Name of Reviewer: Caleb Lopez

Name of Coder: Ruben Martinez

File(s) under review: App.py

Date of Code Review: 10/26/24

**Code Readability**

|  |  |
| --- | --- |
| **5** | Code is easy to understand and follow. |
| **5** | Variables, functions, and classes are named meaningfully and consistently. |
| **5** | Code is formatted consistently (indentation, line spacing, etc.). |
| **4** | Appropriate use of comments without being excessive. |
| **5** | Code is self-documenting where possible (e.g., clear variable names). |

**Code Structure and Design**

|  |  |
| --- | --- |
| **4** | Code is modular, with a single responsibility for each function or class. |
| **5** | Functions and methods are short and focused (ideally under 40 lines). |
| **5** | Similar pieces of code are refactored into reusable functions or modules. |
| **4** | Appropriate separation of concerns (e.g., logic and data handling). |

**Error Handling and Edge Cases**

|  |  |
| --- | --- |
| **4** | Potential errors and exceptions are handled gracefully. |
| **5** | Edge cases are considered and tested. |
| **5** | Input validation is implemented where necessary. |
| **5** | Code is easy to understand and follow. |

**Security**

|  |  |
| --- | --- |
| **5** | Sensitive data (e.g., passwords, tokens) is handled securely. |
| **4** | External inputs are sanitized and validated to prevent injection attacks. |

**Consistency with Standards**

|  |  |
| --- | --- |
| **5** | Code follows language-specific best practices and guidelines. |
| **4** | Style guides (e.g., PEP 8 for Python, Google Java Style Guide) are adhered to. |

**Testing**

|  |  |
| --- | --- |
| **3** | Sufficient unit tests and/or integration tests for the functionality. |
| **3** | Edge cases, error conditions, and main scenarios are tested. |
| **5** | Test cases are isolated and independent from one another. |

**Documentation**

|  |  |
| --- | --- |
| **5** | Purpose and behavior of complex logic are documented clearly. |
| **3** | Configuration options and dependencies are documented. |
| **5** | External libraries and APIs used in the project are properly referenced. |

**Removal of Unnecessary Elements**

|  |  |
| --- | --- |
| **5** | No dead code, commented-out sections, or redundant variables. |
| **5** | Debugging statements (e.g., print statements) are removed from production code. |

**Code Optimization**

|  |  |
| --- | --- |
| **5** | Code can be optimized without sacrificing readability. |
| **5** | Any quick wins for reducing code complexity are considered. |

Reviewer Initials: CL