**Coding Challenge:**

This section covers coding challenges related to quality assurance:

**1. Automation Testing**

This section covers coding challenges related to automation testing.

This challenge focuses on building a robust test automation framework for a web application.

**Functionality**:

* Develop a test script using a testing framework of your choice to automate the following scenario:
  + Navigate to an e-commerce website.
  + Search for a specific product.
  + Add the product to the cart.
  + Proceed to checkout.
* Ensure the script includes appropriate assertions to validate each step.
* The test script should handle dynamic elements and varying data inputs.

**Bonus Features**:

* Implement a retry mechanism for flaky tests.
* Add logging and reporting capabilities to the test framework.
* Integrate the test script with a CI/CD pipeline (e.g., Jenkins, GitHub Actions).

**Technical Requirements**:

* Use a widely accepted programming language for test automation.
* Follow best practices for code structure, modularity, and readability.
* Utilize appropriate libraries and tools for handling web elements, waits, and assertions.
* Provide clear documentation on how to set up and run the tests.

**Evaluation Criteria**:

* Functionality and completeness of the implemented test script.
* Code quality, readability, and adherence to best practices.
* Efficiency and reliability of the test automation framework.
* Documentation clarity and ease of integration for other developers.

**Additional Notes**:

* Focus on demonstrating your understanding of test automation, web element handling, and building maintainable test scripts.
* Feel free to use existing libraries and tools for logging, reporting, and CI/CD integration.

**2. API Tests:**

* **Objective:** Automate the testing of a sample API.
* **Tools:** Use any API testing framework or tool (e.g., Postman, SoapUI, JMeter, RestAssured, etc.).
* **API to Test:** Use <https://github.com/public-apis/public-apis>.
* **Tests Coverage:**
  + Smoke Test:
    - Create a set of smoke tests to quickly verify the most crucial functionalities of the API.
    - Ensure these tests can be run frequently and provide quick feedback on the health of the API.
  + Performance Test:
    - Conduct performance testing to assess the response times and throughput of the API under various load conditions.
    - Include stress testing to determine the breaking point of the API.
  + Integration and Functional Test:
    - Write integration tests to validate the interaction between the API endpoints.
    - Write functional tests to check effectiveness and efficiency of the endpoints.
  + Regression Test:
    - Create a suite of regression tests to ensure existing functionality remains unaffected by new changes.
    - Simulate a code change in the API and run your regression tests to verify no existing functionality is broken.
    - Document the process and results of your regression testing.
  + Generate a test report that summarizes the results of your automated tests.
  + Bonus on creativity and additional tests.

**3. Case Study**:

* Your fintech application is facing quality issues. The product is frequently crashing, and users report inconsistent behavior across different devices.
* Prepare a presentation that covers:
  + Your approach to diagnosing the issues.
  + A test strategy to address these issues.
  + Recommendations for improving the overall quality of the product.
* Include details on test automation, manual testing, and any tools or processes you would implement.