Testing – Soft Skills

Requirements Gathering

1. **Purpose:** Clearly define the purpose of the tests. For example, are they being used to validate the functionality of a new feature, or to ensure the stability of an existing feature?
2. **Scope:** Define the scope of the tests. This should include a list of the specific modules, functions, and features that will be tested.
3. **Inputs:** Identify any input data or parameters that will be required for the tests. This could include test cases, data files, or environment variables.
4. **Outputs:** Specify the expected outputs or results of the tests. For example, do the tests need to return a specific value or generate a report?
5. **Assumptions:** Make a list of any assumptions that have been made about the environment or dependencies of the tests.
6. **Constraints:** Identify any constraints or limitations that may impact the execution of the tests. This could include time limits, resource limitations, or compatibility issues.
7. **Risks: Outline any risks or potential issues that could arise during the tests.** This could include the possibility of test failures or unintended side effects

Example User Stories

1. As a developer, I want to be able to run unit tests on my code so that I can quickly identify and fix any issues before deploying to production.
2. As a QA engineer, I want to be able to create and maintain a comprehensive suite of integration tests so that I can ensure the stability of the entire system.
3. As a product manager, I want to be able to define and prioritize acceptance criteria for new features so that I can confirm that the implemented solution meets the needs of the users.
4. As a technical lead, I want to be able to monitor test coverage and identify areas of the codebase that are not tested so that I can improve the overall quality of the product.
5. As a developer, I want to be able to write tests that are expressive and easy to understand so that I can more effectively collaborate with my team and maintain the test suite over time.

Example Test Cases

1. Test the functionality of a new feature by providing a variety of input values and verifying that the correct output is returned.
2. Verify that an existing function continues to operate correctly by comparing its output to a known reference value.
3. Test the error handling of a function by intentionally triggering an exception and verifying that the function handles it correctly.
4. Perform integration testing by testing the interaction between multiple functions or modules to ensure they work correctly together.
5. Test the performance of a function by measuring the execution time and comparing it to a performance threshold.

Example Test Report Summary

Test Summary

- Tested feature: new user registration
- Test environment: Development
- Test duration: 2 hours
- Test coverage: 90%
- Test results: Pass

Test Cases Executed

Test case: Verify that a new user can successfully register with a unique username and password

  - Test result: Pass
  - Notes: Tested with a variety of valid input values and verified that a new user account was created for each test.

Test case: Verify that the system correctly handles invalid input values during registration

  - Test result: Pass
  - Notes: Tested with a variety of invalid input values and verified that the system correctly rejected them and displayed appropriate error messages.

Test case: Verify that the system correctly handles an existing username during registration

  - Test result: Pass
  - Notes: Tested with a variety of existing usernames and verified that the system correctly rejected them and displayed an appropriate error message.

Test Issues

- None

Conclusion Based on the results of the test cases executed, it can be concluded that the new user registration feature is functioning as expected. No issues were encountered during testing