TL Note – Testing Basics

Testing is just one element of Quality Assurance. Testing is so much more than just coding something then seeing if it works, or whether the client or potential users like it. Comprehensive and various testing is important to projects, regardless of what process (e.g., Waterfall, Agile, etc) is being followed. Following is a brief description of testing activities to help with understanding testing appropriate for the TechLauncher Program.

Testing Activities

Testing activities always form part of your overall development process, but there is variation on how the testing related activities map into the chosen process. For example, for a Waterfall process all testing is carried out after implementation. For agile, testing is carried out in every iteration. The activities include:

Test Strategy Development	A test strategy should be defined covering types, including unit testing, system testing, integration testing, and user acceptance testing as appropriate. When devising a test strategy, keep in mind that you need to ensure that things not only work correctly, but also that invalid or unexpected things are also handled correctly. For example, what happens if an input is out of bounds, and are totally random events handled (e.g., as if there was a monkey on the keyboard)? For user acceptance testing, consider how the testers will be identified and whether there are any ethical issues to be resolved. In the absence of access to real potential users, think about whether pseudo users could be used and any associated limitations.
Test Planning	Much of the testing should be done by validating the outcome against predefined tests and expected results. Hence tests should be defined against (and hence trace from) requirements, to validate implementation. Once again, don't forget to think about coverage of invalid or unexpected things. Test plans are used not only as input to test execution, but also as a basis for impact analysis when things change. Seeking feedback from your client or potential users without up-front expectations may be a valid part of your testing. However, this should still be done with a playbook against which coverage can be assured and feedback can be collected.
Executing and documenting	Think carefully about roles and responsibilities for testing activities. Who will write the test scripts (playbooks), deploy and maintain testing infrastructure, compile measurements and monitor user activities. In small teams, members are likely to have to have testing responsibilities in addition to other roles. Tests should be carried out and results recorded and communicated as appropriate. This may be pass/fail against individual tests, summary test reports. These documented outcomes are important evidence that you have produced not only outputs, but accepted and successful outputs.
Handling test outcomes	Depending on your project processes, think about the appropriate spawning of other processes as a result of test outcomes. A test failure or lack of acceptance must first be analysed since it may indicate: - Incorrect implementation - Misunderstood or incorrectly defined requirement(s). - The test itself is not correct or appropriate. The consequence of the failed test should trigger appropriate project processes to rectify the situation. For example, a defect on implementation will trigger your development process, prioritisation of the fix, and relevant tasks.