**Top Tips for Test Automation**

### **Design Tests Before Automating Them**

It is always a good practice to create the test cases and scenarios before starting to automate the tests. It is the good test design that can help in identifying defects, automated tests only execute the test design.

The danger in jumping straight to automation is that you’re only interested in making the script to work and usually only automate positive and happy flow scenarios rather than thinking about the other possible scenarios that can be tested.

Also, don’t reduce the scope of testing just to make the test work or pass.

### **Don’t Rely Solely on Automation – Beware of Passing Tests**

Automated regression tests can give a sense of confidence for the team because regression tests should still pass as new functionality is delivered. The team starts relying on the tests and having a good set of regression tests can act as a safety net.

However, note that not all tests are automated or can be automated, therefore always accompany automated tests with exploratory testing.

Sometimes a change in the software should fail a test; however, if all tests are passing that means the defect is missed and because there was no call to action, the defect went unnoticed.

### **Understand the Context**

Tests can be automated at any layer, Unit, API, Service, GUI. Each layer serves a different purpose for testing.  
Unit Tests ensure that the code works at the class level, that it compiles and the logic is as expected. Tests at this layer are more verification than validation.

API Tests or Integration Tests ensure a set of functions and classes can work together and data can be passed from one class to another.

GUI Tests on the other hand test user flows and journeys. Generally, we would not test for functionality from the UI. This should be done at lower layers.

The main purpose of UI tests is to ensure the whole system works as per some common user scenarios and use cases. Testing at this layer is more Validation rather than Verification

At UI level, we automate scenarios rather than stories.

### **Don’t Automate Every Test**

100% Test Coverage is not possible since there can be millions of combinations. We always execute a subset of possible tests. The same principle applies to automated testing.

To create an automated script, it requires time and effort, and aiming for “Automating Every Test”, we require a lot of resource and time, which in many cases is not possible.

Instead, use a Risk-based approach to determine which tests should be automated. To get the most value out of automation, only automate the most important business cases and scenarios.

Also, a high number of automated tests adds maintenance cost and difficult to maintain.

Another note to bear in mind is that not all tests can be automated. Some tests are very complex in nature and require many downstream systems checking and can be inconsistent. In these cases, it is best to leave these checks for manual testing.