**2. what are some common causes of flakiness in automation tests and what can you do to help mitigate these causes?**

* Good amount of time and resources are spent on writing good and stable XPath/CSS locators and they will need frequent updates whenever the underlying HTML mark-up changes.
  1. This can be mitigated if developers own/communicate to the testers about the changing locator ids or name properties for elements that could potentially change.
* Not having a suitable framework is the most common mistake in automation testing. Framework defines set of rules or guidelines used for creating and designing testcases. These guidelines defines coding standards, test data handling, object repositories and process of storing the test results.
  1. Automation tool is one of the strongest deciding factors in choosing the correct suitable framework and automation tool is dependent on the technology and requirement set of the project being tested. It is very important to do a steady analysis and complete the mapping of requirements vs. Automation tools vs. Framework. I would say this is more of a precautionary step than a mitigation plan.
* Automation code version control and Test data hard coding are some more reasons that I have noticed to be the most common issues that cause delays in the testing timelines.
  1. Mitigation is to have version control and strong code reviews (also unit testing on the code) so that everyone is aware of the correct code to be plugged in to perform testing). Often times, test data is hardcoded into the code which make the code unsuitable in different scenarios or in some cases different domains, so it is always recommended to have the test data in a repository such as Json or .xls files.
* Synchronisation of code execution in tandem with application – There could be lot of factors impacting the application such as the elements to load into the application, internet speed and the amount of test data being used. So, the automation code should be scripted to accommodate the time, the application takes to load completely for the automation action to occur.
* Automation is not suitable for all testing scenarios. Multiple reasons contribute it to be a not-suitable path:
  1. Developing automation code for some functionalities is expensive and resource consuming and manual testing could be easy and quick. As a test strategy, it is important to understand what should testing functionalities/phases should be automated or manually tested.
* There is no single automation testing tool that supports all applications across the board. Eg: Selenium will not support windows-based applications.
  1. Test Strategy and Planning is key here. It is during this phase that decides what technology should be used for testing. If the product/project being tested has high testing volume in windows-based applications, QTP should be used and If its web-based application, Selenium should be used. Sometimes, it could end up being manual testing.

**3. Consider this scenario:**

**You have estimated that a project needs 100 test cases automated, and it will take you 2 months to complete.**

**However, you have now been told that you only have 1 month to complete the project. What options can you consider to solve this problem?**

I am assuming here that the project has not started or at least in early stages of a testing phase.

1. My first immediate step would be to sit with the BA team and developers’ team to understand the business-critical and technically critical functionalities that must be tested and align the test cases to execute only high priority tests.
2. Reusable code – Complete an assessment to see what are the code snippets that can be reused that covers most of the scenarios.
3. Separate out test cases that can executed manually. If resourcing is not a constraint, I will allocate some non-critical test cases to UAT team which is mutually beneficial. It gives UAT teams a heads-up of what they will be testing in future at the same time gives the testing team flexibility to cover as much testing as possible.
4. **Choose 3 software quality characteristics that relate to automation code and explain why you think they are important or not important to observe.**
5. Reusability is the first thing that comes to my mind if I have to explain about any software code. It becomes more evident in software testing where there are multiple phases of testing and these iterations should on every time a defect is found. Regression testing can be executed multiple times using the same code reducing the time, cost and effort. It also increases the test coverage.
6. Automation is always reliable. It does exactly what it is being coded for – There is no chance of human errors in execution as long as it is tested and the same variables (software, hardware etc…) are used.
7. Exception handling is one more important feature where the automatic execution does not step when there is a failure. Every time there is a failure/mismatch, code will catch these exceptions and record them appropriately with evidences for the testers to investigate and developers to fix.