Improve Quality

1. Test early and Test often with Automation

To improve software quality, it is absolutely paramount to Test early and Test often. Early testing will ensure that any defects do not snowball into larger, more complicated issues. The bigger the defect, the more expensive it becomes to iron out any issues.
The earlier you get your testers involved, the better. It is recommended to involve testers early in the software design process to ensure that they remain on top of any problems or bugs as they crop up, and before the issues grow exponentially which generally makes it harder to debug.

Testing often requires a focus on early adoption of the right automated testing discipline. Start by automating non UI tests initially then slowly increasing coverage to UI based tests when the product stabilises. If your application utilises Webservices/APIs then automate these tests to ensure all your business rules and logic are tested.
This is an important time to work with your software developers to ensure automated testing is also introduced to your development teams, increasing testing coverage, accuracy and improving quality of the overall product.

2. Implement quality controls from the beginning

Testers can monitor quality controls and create awareness in partnership with developers to ensure standards are continually being met. Quality control starts from the beginning, which is an ongoing process throughout the delivery.

A good relationship between testers and developers can help the project software strategy develop effectively. A systematic methodology in quality control can ensure that coding errors and bugs are dealt with effectively, following a structured process.

3. Echo the importance of quality assurance

We have identified how important testing is at the beginning of software development; however, the testing does not stop there. Quality assurance should be ever-present throughout the software development process.
Quality Assurance is a governance provided by the project team that instils confidence in

the overall software quality. Assurance testing oversees and validates the processes used in order to deliver outcomes have been tracked and are functioning. Testing should be repeated as each development element is applied. Think of it as layering a cake. After every layer is added, the cake should be tasted and tested again.

4. Encourage innovations

It is important that testing structures and quality measures are in place, however, there should always be room for innovation. A great way to allow for innovation is to automate testing where possible to minimise time spent on controls.

Innovations are so important because they can lead to improvements in software quality that have the capability to transform how projects are delivered. Research and development (R&D) should be encouraged. Empower teams to explore, experiment and investigate continuously. Also, ensure that advancements in innovation are duly rewarded. They have the capacity to transcend your software quality and deliver projects with a competitive advantage over the competition.

5. Communication is key

For any relationship to be successful, whether it's personal or business, communication is key. To improve software quality it is important that all parties to the project have full information through fluid communication channels.

Fluid communication can take many forms. It can be as simple as having clear, consistent KPIs that show how software quality is measured at every step of the development process. It is important that all team members, regardless of seniority, have access to KPIs to keep the entire team on the same page. Another important aspect of fluid communication is that all parties have the opportunity to provide feedback to the team to ensure that all expectations are being met.

It is also important to keep all stakeholders in the loop and not isolate team members from the vendors or end user of the software. Isolation can cause rifts and can often mean that the project is delayed or may not deliver on the goals expected by senior management.

6. Plan for a changeable environment

Software contains so many variables and is in continuous evolution. It relies on several different external factors such as web browsers, hardware, libraries, and operating systems. These constant external factors mean that software development must be consistently monitored using checks and balances to certify that it remains in stride with its immediate environment. It is important to acknowledge that software is interdependent on these external factors. Accepting this interdependence means that you can plan ahead. It allows you to have the software quality tested, at each step of the process, against external variables, to see how it holds up. The end result is that you will prevent software dissonance and maintain software quality.

7. Take the attitude of creating products not projects

This step is a reflection of the attitude of your team. Creating a project indicates to your team that you are producing a finite outcome. However, we are well aware that software is changeable. If you produce a finite outcome, before long the software quality will not stand up against its environment.

Instead, if your team takes the mindset that they are creating a product it is more likely that they will deliver software quality that is adaptable to change and can stand the test of time. Focus on delivering continuous small progressions rather than one final end project and your team will deliver an increase in quality.

8. Have a risk register

A risk register is a fantastic management tool to manage risks. A risk register is more synonymous with financial auditing, however it is still a vital element in software Development.

A risk register will provide everybody aligned on a software project a list of clearly identified risks and then assess them in regards to the importance of delivering the project. A risk register works well for software quality because its creation actively leads to risk mitigation.

A software development risk register must:
● describe the risk
● recognise when the risk was identified
● acknowledge the chance of the risk occurring and its mitigation
● understand the severity of the impact of the risk
● identify who assesses and actions the risk
● relays the response if the risk materialises

● gives the status of each risk
● measures the negative impact of each risk
● prioritises the risks ranked on probability and gravity

9. Producing software quality requires long-term thinking and strategy

Long-term thinking produces software quality because decisions are made to satisfy lasting issues. Here are the advantages of long-term thinking for producing software quality:

● Doing it right first means you don't have to spend time doing it over.
● Placing as much importance on architecture and design as coding ensures the veracity of your project.
● Creating coding standards with long-term vision eliminates unnecessary mistakes.
● Effective peer review ensures errors are minimised even though it may seem time-consuming at that particular moment.
● Testing often allows you to plan further ahead with certainty that errors and bugs have been fixed.

Project leaders need to ensure that short-term gains and immediate gratification do not compromise long-term strategy. Effective planning will make sure all stakeholders prioritise software quality.

10. Outline your deliverables

From the outset of your project it is imperative that your team outline what they are going to deliver. A clear and concise plan of what the project will deliver helps ensure there is an emphasis on quality from the outset.

It also ensures that budgets, resources, and time are aligned correctly to deliver quality. Without clear deliverables it is likely shortcuts will be taken to meet budgets and deadlines. Ultimately, this will compromise the quality of the software delivered at the end of the Project.

11. Review, revise, and remember

Producing software quality is not a coincidence. This is why you must always do the following three things:
● Review – Testing often is a pillar of ensuring software quality. It ensures that standards are continuously met and bugs, errors and distractions can be fixed before

they spiral out of control.

- Revise – Study what has worked throughout the software process. Utilise what is working and see if innovation can transcend your software quality even further.
- Remember – When you deliver quality remember what worked well and did not work well. Keep an updated record of both the positives and negatives of any given project and turn to it frequently when you start the next project from scratch.