1. *How does testing help in assessing software quality? Does simply "running enough tests" ensure quality? Why or why not?*

I believe that testing help in assessing software quality by discovering software's defects/bugs and fixing them before the delivery to the client, leading to a more reliable and easier to use software. Furthermore, I believe "running enough tests" ensures quality only if all testing components (unit, integration, system, and validation testing) are involved in those tests because of how unit testing determines if there is any issue by the code itself, integration testing validates the interaction between the software units or modules, system testing determines if a complete build aligns with functional and nonfunctional requirements made for it, and validation testing checks if the final product after development meets customer requirements. That is, I believe going through all those testing components and running enough tests for each of these components ensures quality because of how they validate software quality in all aspects.

2. *How much testing is "enough"?*

I believe the answer to this question varies from one project to another depending on multiple factors like the level of risk involved (technical and business risks), deadlines, budget, and what passing/coverage percentage a project has a minimum. For example, for small projects that have low-level risks, flexible deadlines, and small budget like a notes app, having test passing/coverage percentage as 50-70% should be enough that testing the basic functionalities with one or two testing components, like unit and integration testing, will be enough. In contrast, for large projects that have high-level risks, tight deadlines, and huge budget like an air traffic control software, the test passing/coverage should be 99% if not 100%, because of how this system might affect people lives if it fails, so testing every scenario and functionality using all 4 testing components will be the least to do to say testing is "enough" for such system especially when you do not have budget constraints. That is, the bigger and higher risk a project is, the more testing it requires, and vice versa.