## Questions

1. In an ideal world with no schedule or budget constraints, what types of functional and non-functional testing (include at least three of each) do you think should be done on the Radiant Shutters project? For each type of testing that you list, describe the potential kinds of bugs in this project might it uncover in this project?

**Functional:**

Unit Testing. Definitely I would use unit testing because of the deep math calculations the application needs which involves a lot of individual functions that should be roughly tested to get early potentially bad calculations that could affect the final product dramatically.

Integration Testing: This testing will be helpful because of the complexity of the application, and each shutter has so many components and also the process of building, shipping and showing it. So, with this testing we will cover bigger modules of code and evaluate if the modules work together as expected or discover possible correlated bugs.

System testing: Because of the risk and sensitive that the product is to math calculations, It is important to test the application in real-user scenarios and cover or uncover enough edge cases.

**Non-functional:**

Performance testing: We could evaluate the stress and load testing because this application has a lot of variables (like different frame styles, regional preferences, precise measurements including the number of permutation) that could make the calculations more intense. Also, we should add and evaluate the time of the query responses and check if the final numbers are right.

Usability testing: Make a prototype definitely will evaluate the level of satisfaction of the franchise owners. Comparing the old system (DOS and excel) to the new one, we will get to know new errors and efficiency and cover more test cases.

Security testing: Include possible intruders attack and protect sensible information of the system is a critical point to evaluate. An error could turn in financial loss.

2. If you could choose to have Jessie complete 100% code coverage through unit and integration tests, or to have Brice complete 100% coverage of detailed system requirements through user acceptance testing, which would you choose and why? (An excellent response will not only provide rationale for the choice, but also demonstrate an understanding of its limitations as well.)

**I would choose Jessie plan because:**

Responsibility: First of all, Jessie is the Software Engineer in her team and the responsible of the quality assurance of the product. Even the 100 % of code coverage do not guarantee bug-free code but would make the team more confident during the presentation.

Quality assurance: the complexity involved in the high number of variables along with its permutations and the process time response required more intense testing including integration testing. Besides, Brice Team would never understand code as Jessie´s team would.

High financial risk: The product was highly sensitive to math calculation and needed the most rigorous testing possible. If the final product is wrong, it could be a huge financial loss.

Limitations: Too much testing could fall into too much time-consuming and resources. The mindset of developer´s team could be limited because is so focused on internal parts of the code and problem solving and it is not so involved in the user experience.

**I would choose Brice plan because:**

Time constraints: They never talked about the time but, Jessie was feeling pressure because of the amount of work she had to perform, and she had the responsibility to show the demo.

Budget constraints: The solution would reduce 20% of resources and would make Mark happy.

Jessie´s new mindset: Her team started to work making code cleaner, more efficient and analyzing possible points of failure. They made a great work adapting to the new solution and having more empathy from the user perspective.

Limitations: Jessie would reduce code coverage at 60% percent. She was uncertain of the load and stress testing and the chances to fin critical bugs.

3. When Brice agreed to take the bulk of the testing responsibility, Jessie felt that being a good enough coder and being precise enough would mitigate the need for robust testing. What are the fallacies in that conclusion? (An excellent response will recognize the value of quality processes in development, but also demonstrate why they are insufficient.) [*Explains the value of quality processes in development but demonstrates why these development practices cannot replace the need for testing.]*

Jessie needs to remember the importance of testing. Testing comes after development. It is through testing that we ensure the software functionality and non-functionality. Even making the code cleaner and effective, does not guarantee a free-bug code.

Testing is about managing risks. Jessie and Brice decided to put a limit to test according to the problem Jessie was passing through and then decided to put some components of the products as non-critical. Maybe that is the trap, viewing something as non-critical, reducing testing and limiting quality. Through this decision, Jessie increased the tolerance for possible problems, and depositing her trust in UAT.

4. At the end of the case, Jessie is faced with a difficult situation, being told to stop development, but also knowing that the code will be used despite known areas of vulnerability or technical debt. [*Articulates at least 4 possible actions that Jessie can take and explains concrete risks associated with each action. A specific decision is made and supported by reasonable rationale].*

A) What are 4 possible actions Jessie could take? For each one, list the risks associated with that action (technical, reputational, or other risks).

B) Which of these actions would you take and why?