**Possible Testing Approaches**

The various approaches that could be taken for testing would be:

1. The most obvious testing strategy that first comes to mind would be something akin to the agile process to test the requirements as stated in the Software requirements document. To accomplish this, we would have to incorporate testing as we move through the development stage. Such testing is also referred to as "proactive." This strategy allows for the creation of "black box" tests, which are created solely based on the requirements. Another variation of this approach could include a “grey box” test where limited implementation information is available.
2. Another approach that could be used is to conduct testing analysis after the system has been fully implemented. Such an approach is monikered “reactive” testing since we start testing at the end stages of the development. Since we are fully aware of all the implementation details while the tests are being developed, the approach uses a technique such as “white box” tests.
3. Another strategy would be one in which we primarily rely on an amalgamation of proactive and reactive testing. Proactive testing would involve predicting potential problems in advance by looking for trends or changes, while reactive testing would consist of responding quickly to address any issues that arise. With this approach, testers can use both reactive testing techniques (in the white-box aspect of testing) to ensure proper functionality early in the development process as well as reactive techniques (in the black-box aspect of testing) to fine tune test cases closer to the release date. This strategy would allow us to make sure that the software performs its desired function while also allowing us to identify any potential problems before they become widespread. This combination of techniques gives the testers an advantage, as it allows them to start testing early while also giving them time to create and refine test cases closer to the release date. By using the combination of reactive and proactive testing techniques in this approach, testers can ensure that all facets of the system are properly tested before it is released.
4. Another testing strategy would be that of heuristic testing; Goldilocks testing is a straightforward, heuristic-based testing approach that encourages testers to find the "just right" value for whatever element they are testing. It uses well-known heuristic testing techniques used by previous software testers.
5. Iterative testing is another testing strategy that may be implemented. Testing would be carried out at the climax of each "iteration" if the development process were divided into smaller "iterations." This would seamlessly combine testing and development, making it possible to find errors before moving on to the next "iteration," and it would also rely on grey-box testing, where only a limited amount of detail about the implementation specifics is available. By combining white-box and black-box testing, we can ensure that the software developed meets the requirements in an efficient manner. This iterative testing approach would provide great benefits, such as a structured approach, better communication between the development and testing teams, and a faster response to changing requirements.
6. The last testing strategy that could be applied is the DevOps continuous testing strategy. In this method, all tests are run simultaneously using an automation tool and pipeline, and if any bugs or errors are found, immediate feedback is generated. This is the least practical option for the chosen software since the project doesn't scale up enough, nevertheless it is a strategy worth mentioning because it essentially has very few drawbacks.

**Chosen Testing approach:**

We must be mindful of aspects like risks, resource costs, and schedule-timeline conflicts when selecting a testing approach. The third testing strategy was chosen because of the following factors: an amalgamation of proactive and reactive testing. Proactive testing would involve predicting potential problems in advance by looking for trends or changes, while reactive testing would consist of responding quickly to address any issues that arise. With this approach, testers can use both reactive testing techniques (in the white-box aspect of testing) to ensure proper functionality early in the development process as well as reactive techniques (in the black-box aspect of testing) to fine tune test cases closer to the release date. This strategy would allow us to make sure that the software performs its desired function while also allowing us to identify any potential problems before they become widespread. This combination of techniques gives the testers an advantage, as it allows them to start testing early while also giving them time to create and refine test cases closer to the release date. By using the combination of reactive and proactive testing techniques in this approach, testers can ensure that all facets of the system are properly tested before it is released. This ultimately allows us to test the system in a comprehensive manner, ensuring that all aspects of the software have been thoroughly tested and evaluated before it is released. This strategy is beneficial to testers, as it allows them to start testing the system early, while also providing them with ample time to refine and create test cases.