PA2579 – ASSIGNMENT 3

Adaptive-lean software testing

Testing technology-facing qualities

Student Name: Uday Jain

Let me start with providing a brief overview of my team, its structure and products we own. I work as a part of an internal analytical unit that aims to provide analytical support to different organizations within the telecom company. The analytical support includes not only analytical products such as business focussed dashboards and machine learning models but also includes analytical infrastructure that support analytics such as on-prem and on-cloud data lake, machine learning workbench and data catalog. Since all our products are restricted to internal use, security and privacy are constrained through firewalls in a closed system. This makes the need for security testing very limited in scope. However, performance testing is one area I would be discussing in detail as a part of this assignment.

The unit that I work with is currently in the developing phase of software maturity with many parts done ad-hoc and based on user feedback. Eg. The machine learning (ML) workbench often faces capacity issues learning to code failures when a user(s) initiates a run that requires high computational resources. With projects moving towards analytical maturity, complex ML algorithms that require high computational resources were becoming a frequent task. Code failures and system wide crashes were becoming more common, creating a need for user based resource allocation and capping. Although a well thought performance testing of the workbench could have helped us scale optimally and minimize failures, increasing the overall system resources would have led to exploitation and over utilization of available resources. Provisioning additional resources on demand helped the unit maintain costs under control and scale utilization minimizing system wide failures.

However performance testing, especially capacity testing has been of significant use when deploying ML models or data pipelines where the size of data can vary significantly over time depending on the user behavior and telecom network conditions. As a rule we typically account for 20% variability in resource requirements should the data volume increase, however, with increased focus on data driving decisions, we have now started to use the historic distribution of data volume to understand and accurately predict resource needs.

With additional knowledge on stochastic data generation techniques, I plan to explore its use in stress and capacity testing of these pipelines and ML algorithms. This can also help us with performance testing of end user facing analytical products such as dashboards, some of which are often found to have high time to load given sudden increased volumes of data leading to poor user experience. Historically we have had limited success in implementing any performance testing tools in the space given the complexity.

With increasing analytical maturity and different parts of the company becoming self-reliant, I have found a trend of growing tech stack. This makes it difficult and complex to develop scalable performance testing tools for our internal products. Also, ensuring automated performance testing for multiple smaller products would make it harder given lower returns on investment. Thus making us reliant on manual user based testing or end-user feedback for even smaller products.

Taking learning from the module and inspiration from my past experience, we can aim to explore and develop components of scalable performance testing tools to make it easier for individual teams owning analytical products to develop custom adaptors on top for integration. Eg. a stochastic data generation tool that takes user defined limits on type and quantity of data to provide data stream as an API. This API can be used across different cloud based workbenches or even our on-prem workbench, allowing users to customize stress testing requirements of their pipelines. This can be coupled with a reporting solution that allows users to schedule a predefined report with custom performance metrics such as response time, failure rate & runtime. Although the reporting plugin may need an adaptor customized to tech stack.

Such solutions provide us with flexibility to develop scalable solutions as well as cater to the needs of an audience with a diverse tech stack.