“Will you be able to build a web platform that can deliver 60,000 requests per second, with a sub-second response time?“

This was the requirement given to us in a customer meeting.

This was our opportunity to build a challenging project from scratch and I think I can say very confidently that, very few companies and people get such opportunities. We were jumping with joy, but then we realized the scale of the challenge.

I want to highlight the practices that helped us deliver the solution for the above challenges.

These practices, I believe can be termed as ‘Best Practices‘ that have come forth from the entire journey of Designing, Building, Testing, and Deploying the solution.

1) Theme of your Project

The Theme of a project is difficult to decide, it needs to be carefully derived from the project definition and requirements.

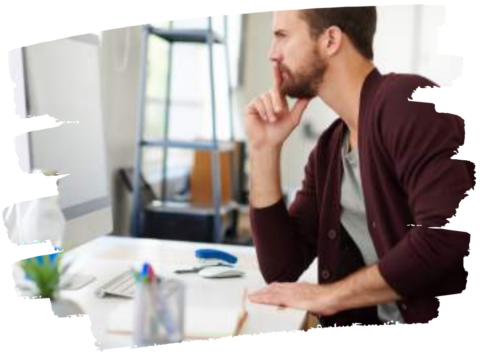
The Theme of the project is a summary of what is expected to be reached by the project.

2) Know your Performance targets

You won’t get a performant system, by asking someone to build a “really fast system”.

Demands of the performance vary for different problems and domains; and accordingly, it needs very different solutions and technologies to deliver these performances.

Software Performance is absolute; you get what you demand.



3) Have an Eye for Details

Details are very important for all kinds of projects. Especially at a large scale, seemingly small problems can spiral out of control due to negligence of minute details.

4) Measuring your System Health

After having done the best possible development and testing for the proposed application the system is now in a production / BETA environment. It is subjected to real Variety-Velocity-Veracity of live data.

It starts creaking & showing a slew of performance issues, slow responses, dropped requests, DB performance drops, Queues choked, etc.

And, guess what??

These symptoms were never visible in the test labs.

The system is live and being used by customers and/or end-users.
tails.

5) Shift-Left Testing

Shift-Left Testing is a transformation from Defect detection to defect prevention”

Traditionally Testing has been at the very right extreme of Requirements, Design, Development, and Testing of the software development cycle. Shift-Left concept says move your testing to the left to involve testing and testers in every phase.

This approach helps detect and fix defects very early in the game, rather than detecting them in your traditional test cycles at the extreme right. It gives you time to understand and design the test scenarios, test cases and identify automation hotspots.

6) Extreme Automation

Automation plays a very important role in product and project development. Delivering platforms and systems at this scale is not possible without extreme automation.

7) Controlled Roll-Outs

How do you seamlessly roll out your platform for a 25+ Million user base (hopefully) without causing service disruption?

8) Everything can go wrong – Build Contingencies and Switches

“Anything that can go wrong will go wrong”
-Our good old friend Murphy

Please believe him, especially when you are building systems of such scales.

You should have a hot switch to toggle any settings while the system is in progress. Changing the settings, in the settings file and restarting systems is never an option since systems at this scale take a huge amount of time to restart and again warm-up; forget about reproducing the same state of components again.

Get more insights on our website

www.coreviewsystems.com