Indledning

Serveren er hostet på azure, udviklet i .NET Core og koden kan findes her
https://github.com/cph-ol24/jmeter

Spørgsmål

De 4 agile quadrants

De 4 agile quadrants handler om måden hvorpå man planlægger tests, og hvilken slags tests man skriver.

I “Quadrant One” fokusere man hovedsageligt på

- Unit Tests
- Component Tests

Ud fra hvilken tests vi kører, så det tests som vi kan køre automatisk. Målet med “Quadrant One” er at skabe kvalitets kode, mod og sikkerhed i forhold til designet.

I “Quadrant Two” fokusere man på prototyper, funktionelle tests, eksempler mm. Kendetegnende for disse tests er at de kan køre halv automatiske (lidt manuelt arbejde er nødvendigt). Målet med “Quadrant Two” er at skabe forretning tests, som skaber værdi for kunden.

“Quadrant Three” handler om at teste de kritiske komponenter og forretnings dele. Man kan gøre det med Exploratory Testing, og Usability Testing. Målet er at man tester om brugerne rent faktisk kan finde ud af at bruge den udviklede software. Man gør det ved at lave demoer sammen med brugerne.

“Quadrant Four” handler om at teste om systemet kan klare presset. Dvs. hvis der kommer et stort load på databasen, vil den så gå ned? Eller hvis det er en hjemmeside, kan serveren klare en pludselig stigning i antal af requests? Derudover er det også her at man tester om systemet er sikkert at benytte. Man vil også f.eks. teste sit backup system, og om man migere databasen.

Exploratory testing

Exploratory testing fik sit navn i år 1983 af Cem Kaner, og handler om hvordan man “uforberedt” kan teste et system. Exploratory testing er brugbart til at finde ting i softwaren, som man

nødvendigvis ikke har haft tænkt over i test planlægningen. Exploratory testing er ikke et testing teknik, men en måde at tænke omkring tests.

Exploratory testing foregår ved at man finder en tester, som så får en opgave som han eller hun skal prøve at løse. Undervejs prøver testeren forskellige inputs eller opsætninger, og ser hvad der sker. Hvis der sker en fejl bliver det noteret, så det kan blive evt. kan blive rettet inden softwaren er færdig.

System testing

System Testing går ud på at man tester systemet i sin helhed. Man tester her alle komponenter om de fungere som man forventer. Dette er en test som man først kan lave når systemet er færdig og er klar til levering til kunden.

Når man laver system tests, så er det vigtigt at kigge på det store billede, for at sikre at man ikke har udviklet et system som ikke virker efter hensigten.

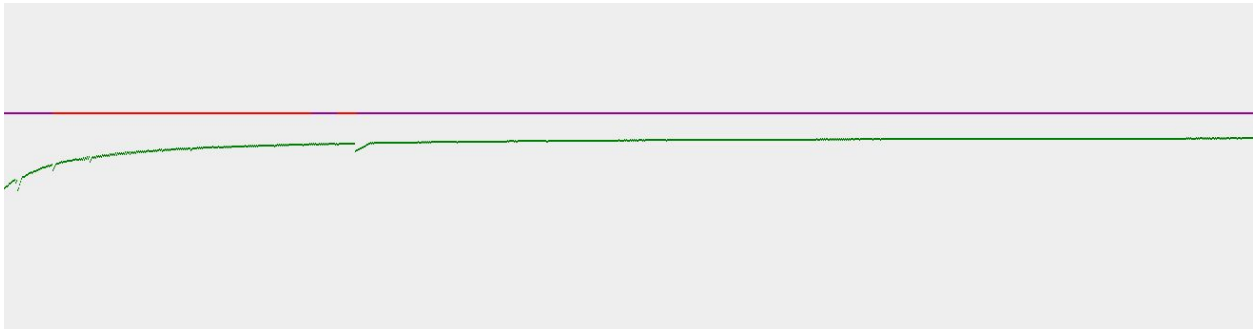
Man kan på en måde sige at system testing er integration testing, dog er der flere dele man tester ad gangen, og man skal ikke kun fokusere på en enkelt ting, men på helheden.

JMeter

Ud fra det data jeg har fået igennem JMeter kan jeg se at serveren lokalt godt kan klare det store load, og når den får det rigtige input, så sker der ingen fejl og returnere det forventede resultat.

Den kan klare det store load rigtig hurtigt fordi det er en kestrel server, som er optimeret til store load, og fordi den ikke har brug for eksterne komponenter, som f.eks. en database eller læsning af filer som kan skabe deadlocks.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
scount	3000	1	0	73	2.45	0.00%	8.9/sec	1.41	1.58	161.6
count	3000	1	0	82	2.80	0.00%	8.9/sec	1.40	1.23	161.0
TAL	6000	1	0	82	2.63	0.00%	17.8/sec	2.81	2.81	161.3



På grafen kan man se at serveren (eller JMeter) lige skal varme op, men ellers så kan den klare loadet problemfrit.