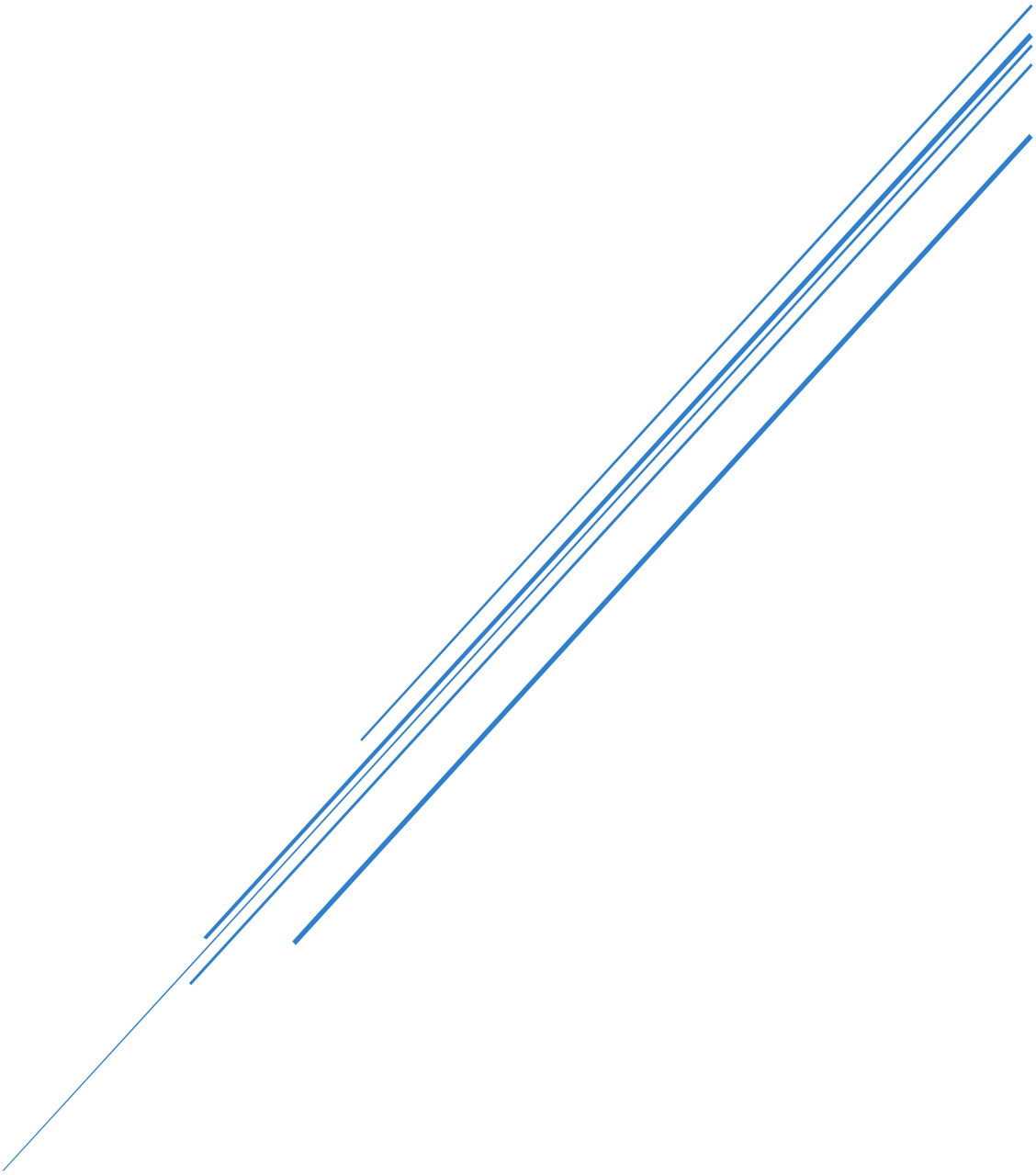
SOFTWARE QUALITY

Assignment 1



Group B

Indholdsfortegnelse

Reflection

Reflect on Code Coverage

Unit Tests: Her fokuserer vi på at teste små, isolerede kodeblokke for at sikre, at vores logik fungerer korrekt uden eksterne afhængigheder, som f.eks. en database.

Integration Tests: Vores integrationstests hjælper med at validere, at forskellige services fungerer sammen, og sikrer korrekt integration på tværs af systemet.

Balance mellem Unit og Integrationstests:

Unit tests er hurtigere og dækker en stor del af logikken, men de kan ikke identificere integrationsproblemer. Integrationstests dækker mere komplekse interaktioner, men er samtidig langsommere og mere ressourcekrævende. Vi sigter efter bred dækning med unit tests og sørger for, at vigtige forbindelser, som f.eks. databaseforbindelser, bliver testet med integrationstests.

Konklusion

For at sikre optimal code coverage har vi dækket al logik med unit tests. Samtidig har vi tilføjet integrationstests for at validere korrekt interaktion mellem komponenter, som API'er og databaser. Denne balance giver os mulighed for både at fange logiske fejl i isolation og reelle problemer, der opstår på grund af interaktioner mellem forskellige systemdele.

Vigtigheden af Code Coverage

Det sikrer, at alle vores kritiske logikker i systemet bliver testet. Det hjælper med at identificere de kodedele, der kan indeholde fejl eller uforudsete handlinger. Ved refaktorering vil code coverage hjælpe med at fange de funktionaliteter, der ikke længere fungerer. Det vil også kunne hjælpe med at opretholde kodekvalitet og fange potentielle fejl tidligt. Men selv ved 100 procent dækning betyder det ikke altid, at alt er testet, eller at testene er meningsfulde. Teknisk set kan man opnå 100 procent code coverage uden at sikre, at alle mulige scenarier eller forretningslogik er testet korrekt, da værktøjet kun måler, om koden er blevet eksekveret. Det er vigtigt at fokusere på testenes dybde og sikre, at de validerer vigtige adfærdsmønstre

Optimering af ydeevne baseret på Load Testing-resultater

Reflektion på Load Testing:

Med vores lille testprojekt har vi ikke særlig meget data, og på baggrund af resultaterne fra load testen klarede systemet sig godt under moderat belastning, med en gennemsnitlig svartid på 12 ms for HTTP-forespørgsler og ingen fejl. Hvis vores projekt var betydeligt større både i forhold til data og kompleksitet i logikken, ville load testresultaterne sandsynligvis se anderledes ud. En af de tiltag, man kan gøre for at optimere ydeevnen, er at implementere caching for at reducere belastningen på databasen.