Assignment 2: End-to-End Testing Strategy and Implementation
Test Strategy

1. Test scope

1.1 Overview of solution to be implemented

En simpel terminalbaseret todo applikation. Applikationen har følgende funktionalitet:

- Tilføje, opdatere, slette og markere tasks som færdige
- Kategorisere opgaver
- Sætte deadlines for opgaver
- Interaktionen med applikationen vil ske i terminalen

1.2 In scope

- Alle Java klasser, herunder Main, Task og ToDoList.

1.3 Out of scope

- Brugerfladen (GUI / Terminal)

2. Testing objectives & approach

- Det skal sikres at metoderne som håndterer kernefunktionaliteten af applikationen virker efter hensigten. (Metoderne i Task og ToDoList klasserne.) For at opnå dette benyttes Unit test via JUnit frameworket.
- Der testes for at applikationen ikke crasher på baggrund af illegalt brugerinput. Her udføres integrationstest for at teste interaktionerne mellem UI og applikation.
- Der laves test, som bekræfter, at kravspecifikationerne opfyldes. Denne specifikation baserede test laves på baggrund af begrænsningerne ved et konsolbaseret UI, manuelt og ikke automatiseret.

3. Test environment, infrastructure and tools

3.1 Facilities and infrastructure requirements

- None required.

3.2 Test environment and tools

- JDK: Microsoft OpenJDK 21.0.8
- OS: Windows
- IDE: Intellij
- Test tools: JUnit5

4. Assumptions and constraints

4.1 Assumptions

- Alle udviklere gør brug af det definerede test environment.
- Kravspecifikationerne forbliver konstante.

4.2 Constraints

- Konsolbaseret UI kan være vanskeligt at teste automatisk.

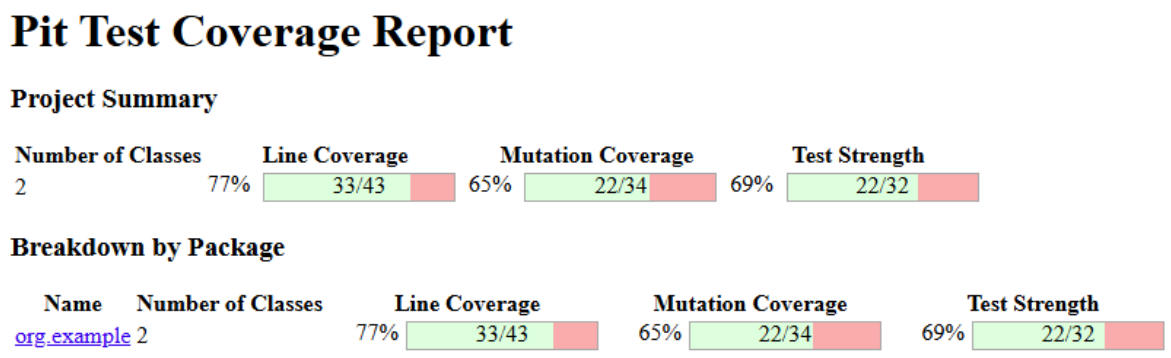
4.3 Dependencies

- Java
- JUnit5

- **Discuss how each type of test contributes to the overall quality of the software.**

Unit tests er med til at sikre stabilitet for brugen af metoderne. Det skal sikre at vi får det korrekte input og output. Alle funktionaliteter testes på et lavt kodeniveau.
Integrationstestene for programmet simulerer applikationen ved at køre Main og ”optage” outputtet fra terminalen med et prædefineret input, som skal sikre at programforløbet foregår korrekt.
Dette er med til at sikre at brugeroplevelsen er som forventet.

- **Provide a detailed analysis of the mutation testing results and how they informed your testing process.**



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	77%	33/43	65%	22/34	69%	22/32

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
org.example	2	77%	33/43	65%	22/34	69%	22/32

Samlet: Line 77% (33/43), Mutation 65% (22/34), Test strength 69% (22/32).

Per klasse: Task 100% mutation coverage (8/8) → stærke asserts; ToDoList 54% (14/26) → tests kører koden men fanger ikke alle fejl.

Læring/handling: Vi tilføjer målrettede tests til tom liste (forventer “No tasks found.” + null) og out-of-range indeks for update/delete/complete, samt eksplicit før/efter-asserts på liste-størrelse og felter. De tester netop de brancher, hvor mutanter overlevede.

- **Clearly explain why and how you used each type of test double.**

Stub (System.in): vi simulerer brugerinput i CLI-integrationstests → deterministiske flows.

Spy (System.out): vi opfanger konsoloutput og asserter brugerens viste linjer.

Ikke brugt bevidst: Mocks/Fakes/Dummies – domænet er simpelt, og vi tester reel integration.

- **Reflect on the distinction between verification and validation as outlined in the Plutora article.**

Verification: Unit- og integrationstests verificerer, at implementeringen opfører sig i henhold til specifikationen (fx at updateTask rent faktisk opdaterer alle felter, og at flows gennem Main udskriver korrekte linjer).

Validation: Specification-baserede scenarier fra brugerperspektiv (CLI-flows) bekræfter, at løsningen leverer den tilsigtede værdi for brugeren (man kan tilføje, se, slette, fuldføre — og konsollen viser det korrekt).

- **Read the article by Peter Morlion and evaluate your test strategy in the context of software quality.**

- **Reflect on how your testing approach could be improved to better ensure quality.**

- **Briefly discuss the test categories as outlined by Martin Fowler and classify the tests you have implemented accordingly.**