# Software Test Rapport – Unit & Integration Testing

## 1. Generel Information

**Projekt / Systemnavn:** PDF-downloader  
**Testansvarlig: Viktor From**  
**Versionsnummer / Build ID: 1**

## 2. Testformål

**Formål med testen:**

Formålet med denne test er at verificere at PDF Downloaderen korrekt kan håndtere download af PDF’er.  
(Beskriv kort hvorfor testen udføres – f.eks. for at validere korrektheden af en metode, sikre korrekt dataudveksling mellem moduler, eller bekræfte at nye ændringer ikke bryder eksisterende funktionalitet.)

## 3. Testresumé

**Overordnet opsummering af testforløbet:**

(Beskriv hvor mange tests der blev udført, hvor mange der bestod, og om der blev fundet fejl. Angiv eventuelle bemærkninger om stabilitet, performance eller funktionalitet.)

| **Testtype** | **Antal tests** | **Bestået** | **Fejlet** | **Kommentar** |
| --- | --- | --- | --- | --- |
| Unit tests | 8 | 2 | 4 | Mange af dem fejler grundet at  De ikke giver Exceptions |
| Integration tests |  |  |  |  |
| **Samlet resultat** |  |  |  |  |

## 4. Testet Funktionalitet

**Beskrivelse af hvilke dele af koden der blev testet:**

| **Klasse / Modul** | **Metoder / Funktioner** | **Formål med testen** | **Resultat** |
| --- | --- | --- | --- |
| **Prepare.py** | Prepare\_folders\_and\_find\_pdf\_duplicates()  load\_and\_filter\_excel\_data |  |  |
|  |  |  |  |
|  |  |  |  |

## 5. Testtilgang og Miljø

**Tilgang til test:**  
(Beskriv hvordan testene blev udført – fx TDD, manuelt med testdata, automatisk via testframeworks som xUnit, JUnit, NUnit, Jest etc. Angiv også om mocking/stubs blev brugt.)

*Eksempel:*  
Testene blev udført automatisk via NUnit. Mocking af databaseadgang blev implementeret med Moq for at isolere logiklag fra dataadgangslag.

**Testmiljø:**  
(Beskriv systemets opsætning – OS, udviklingsmiljø, frameworks, versioner, database, API’er mv.)

| **Komponent** | **Version / Setup** |
| --- | --- |
| Operativsystem |  |
| IDE / Framework |  |
| Database |  |
| API’er / eksterne services |  |
| Build-værktøj |  |

## 6. Testresultater

**Detaljeret resultatoversigt:**  
(For hver test, beskriv input, forventet output og faktisk output. Tilføj evt. skærmbilleder, testkoder eller loguddrag i bilag.)

| **Test ID** | **Testnavn / Enhed** | **Input** | **Forventet Output** | **Faktisk Output** | **Resultat** | **Kommentar** |
| --- | --- | --- | --- | --- | --- | --- |
| UT-001 | AddUser() | gyldig bruger | Bruger oprettes | Bruger oprettes | ✅ Bestået |  |
| UT-002 | AddUser() | tomt navn | Fejlmeddelelse | Fejlmeddelelse | ✅ Bestået |  |
| IT-003 | OrderService + PaymentGateway | ordredata | Ordre betalt | Timeout | ❌ Fejlet | Fejl ved API kald |

## 7. Fejl og Observationer

**Beskrivelse af fundne fejl eller uoverensstemmelser:**  
(Angiv kort beskrivelse, sværhedsgrad og evt. forslag til løsning.)

| **Fejl-ID** | **Beskrivelse** | **Sværhedsgrad** | **Forslag til løsning** | **Status** |
| --- | --- | --- | --- | --- |
| BUG-01 | Betaling fejler ved manglende token | Høj | Tilføj token-validering før kald | Åben |

## 8. Opsummering og Konklusion

**Overordnet vurdering af testforløbet:**  
(Hvordan vurderes systemets stabilitet og kvalitet? Er integrationerne pålidelige? Hvilke dele kræver yderligere test?)

*Eksempel:*  
Unit testene viser generelt høj stabilitet i forretningslogikken. Dog kræver integration med betalingstjenesten yderligere fejlhåndtering. Testdækningen estimeres til ca. 85 %.

## 9. Refleksion og Videre Arbejde

**Refleksion:**  
(Hvad fungerede godt under testarbejdet? Hvilke udfordringer opstod? Hvordan kan testprocessen forbedres fremadrettet?)

*Eksempel:*  
Mocking af API’er fungerede godt, men testdatahåndteringen var tidskrævende. I fremtidige iterationer bør der anvendes et fast testdatasæt og CI-integration med automatiske testkørsler.

**Plan for videre arbejde:**

* Udvide testdækning for edge cases
* Implementere integrationstest for nye moduler
* Automatisere testkørsel i CI/CD pipeline
* Dokumentere testresultater i versioneringssystem