# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Software Quality Assurance (SQA) Plan & Report

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Projekt:**

Passwortmanager

**Autor: Hugo Hölzle**

# Software Quality Assurance (SQA) Plan & Report

**Datum:**

**10.03.25**

**Version:**

1.0

**Prüfer:**

Sascha Wanninger

**Freigabe durch autorisierte Person:**

Ekaterina Karavaeva

**Changelog:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | **Datum** | **Version** | **Änderung** |
| Hugo Hölzle | **10.03.25** | 1.0 | ------------- |
|  |  |  |  |
|  |  |  |  |

# Software Quality Assurance (SQA) Plan & Report

## 1. Einleitung

### 1.1 Zweck

Dieser Software Quality Assurance (SQA) Plan beschreibt die Maßnahmen zur Sicherstellung der Softwarequalität des entwickelten Passwort-Managers. Ziel ist es, sicherzustellen, dass die Software die funktionalen und nicht-funktionalen Anforderungen erfüllt.

### 1.2 Geltungsbereich

Der Plan gilt für die gesamte Entwicklung des Passwort-Managers und umfasst die Planung, Implementierung, Tests und Wartung der Software.

### 1.3 Referenzen

- IEEE 730-2014 (Standard for Software Quality Assurance Plans)  
- ISO/IEC 25010 (Software Quality Model)  
- Projektanforderungen der DHBW Ravensburg

## 2. Qualitätsziele & Metriken

- Fehlerrate: Weniger als 2 kritische Fehler pro Release  
- Code Coverage: Mindestens 75 % Testabdeckung  
- Pylint Score: Mindestwert von 8/10 pro Datei  
- MyPy: Keine Typfehler  
- Security Audits: Sicherstellen, dass keine bekannten Sicherheitslücken existieren

## 3. Verantwortlichkeiten & Rollen

- Projektleiter: Gesamtverantwortung für Qualitätssicherung  
- Entwickler: Umsetzung der Qualitätsanforderungen  
- Tester: Durchführung von Testverfahren  
- Code Reviewer: Review und Freigabe des Codes

## 4. Entwicklungs- und Testprozesse

- Unit-Tests: Automatisierte Tests mit unittest  
- Integrationstests: Testen der Interaktion zwischen Modulen  
- Systemtests: Gesamtsoftware auf Endbenutzeranforderungen testen  
- Regressionstests: Sicherstellen, dass neue Änderungen keine Fehler einführen

## 5. Verwendete Werkzeuge & Methoden

- Testmanagement: unittest  
- Code-Qualität: pylint, mypy  
- Versionskontrolle: Git (GitHub Repository)  
- Sicherheitsprüfungen: cryptography Bibliothek für sichere Verschlüsselung

## 6. Risikomanagement & Fehlerbehandlung

- Risikominderung: Code-Reviews, Peer-Reviews  
- Fehlerverfolgung: Issue-Tracking über GitHub Issues  
- Fehlerklassifikation:  
 - Kritisch: Beeinträchtigt Kernfunktionen  
 - Hoch: Erfordert sofortige Behebung  
 - Mittel: Beeinträchtigt Usability  
 - Niedrig: Kosmetische oder nicht-kritische Probleme

## 7. Reviews, Audits & Inspektionen

- Code-Review-Prozess: Jeder Code-Commit muss durch mindestens ein Teammitglied freigegeben werden  
- Interne Audits: Monatliche Code- und Sicherheitsprüfung  
- Externe Audits: Falls erforderlich, durch externe Prüfer

## 8. Berichtswesen & Dokumentation

- Testberichte: Automatisch generierte Reports mit coverage  
- Abweichungsberichte: Dokumentation gefundener Fehler und deren Behebung  
- Abschlussbericht: Zusammenfassung der durchgeführten Tests und Ergebnisse

## 9. Abweichungen & Verbesserungen

- Umgang mit Abweichungen: Kritische Fehler müssen innerhalb von 24 Stunden behoben werden  
- Kontinuierliche Verbesserung: Regelmäßige Retrospektiven zur Prozessoptimierung