**Projektskizze:**

**Changelog:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | **Datum** | **Version** | **Änderung** |
| David Pelcz | 02.10.2024 | 1 |  |
| David Pelcz | 10.10.2024 | 2 | Trennung der Projektskizze von dem 1. Meeting und Überarbeitung |
| David Pelcz | 14.11.2024 | 3 | Randbedingung update |

|  |  |
| --- | --- |
| **Name** | **Rolle** |
| David Pelcz | Scrum Master |
| Katya Karavaeva | Product Owner |
| Alexander Bernecker | Entwickler |
| Finn Ludwig | Entwickler |
| Luca Jäger | Entwickler |
| Hugo Hölzle | Entwickler |

**Scope of work:**

Aufgabe der Arbeit ist es, einen Konsolen basierten Passwort Manager in Python zu implementieren.

Technologie Es muss ein Passwort Manager implementiert werden. Das Programm muss in der Programmiersprache Python entwickelt werden. Das Programm muss in Python Version 3.11.5 mit dem Betriebssystem Linux (Ubuntu 20.04 LTS) lauffähig sein. Das Programm muss eine grafische Oberfläche über die Kommandozeile bieten.

Funktionale Anforderungen

Passwortspeicherung: Sichere und Verschlüsselte Speicherung von Benutzernamen, Passwörtern und zugehörigen Website-URLs. Verschlüsselung der gespeicherten Daten mit einem starken Algorithmus (z.B. AES-256). Möglichkeit, zusätzliche Informationen zu speichern (z.B. Notizen, Kategorien). Verschlüsselte Speicherung von Metadaten wie z.B. Erstelldatum und Uhrzeit. Passwortverlauf: Speicherung von älteren Passwortversionen, um die Wiederverwendung von Passwörtern zu verhindern und bei Bedarf auf ältere Passwörter zurückgreifen zu können.

Passwortgenerierung: Generierung sicherer, zufälliger Passwörter mit konfigurierbarer Länge und Zeichensatz (Groß-/Kleinbuchstaben, Zahlen, Sonderzeichen). Anpassung der Passwortstärke (z.B. Ausschluss bestimmter Zeichen, Erzwingen bestimmter Muster). Passwortabruf: Suche und Anzeige gespeicherter Passwörter anhand von Website-URLs oder Suchbegriffen. Anzeige von Benutzernamen und zusätzlichen Informationen. Passwortbearbeitung: Änderung von Benutzernamen, Passwörtern und zusätzlichen Informationen. Löschen von gespeicherten Passwörtern. Master-Passwort: Schutz der gespeicherten Daten durch ein Master-Passwort. Sichere Speicherung des Master-Passworts (z.B. Hashing).

Benutzeroberfläche: Intuitive Menüführung über die Konsole. Klare Anweisungen und Fehlermeldungen. Steuerung über Pfeiltasten oder alternative Eingabemethoden. Nutzermanagement Erstellung von mehreren Accounts mit jeweils eigenem Master-Passwort

Import/Export: Export und Import der Passwörter in ein Text Format (z.B. CSV oder JSON). Passwortprüfung:

Überprüfung der Stärke von Passwörtern anhand gängiger Kriterien (z.B. Länge, Zeichensatz, Wiederverwendung). Warnung vor schwachen Passwörtern. Warnung bei gleichen Passwörtern.

Nicht-funktionale Anforderungen Sicherheit: Schutz vor unbefugtem Zugriff auf die gespeicherten Daten. Sichere Implementierung der Verschlüsselung und des Hashing. Zuverlässigkeit: Stabile Funktionalität ohne Abstürze oder Datenverlust. Fehlertolerante Handhabung von ungültigen Eingaben oder unerwarteten Ereignissen. Benutzerfreundlichkeit: Einfache Bedienung auch für Benutzer ohne technische Vorkenntnisse.

Dynamische Code Analyse Es müssen automatisierte Unittests entwickelt werden. Es muss die Bibliothek Unittest genutzt werden. Die Tests müssen logisch sinnvoll entwickelt sein. Statische Code Analyse - Pylint Es muss Pylint auf jede Python Datei ausgeführt werden. Alle Pylint Warnungen müssen behoben werden. Meldungen können durch eine gute Begründung unterdrückt werden. Pylint muss über Konsole ausgeführt werden. Type Checker - MyPy Alle Python Dateien müssen von mypy überprüft werden. Es müssen alle mypy Warnungen behoben werden. Es muss folgende mypy.ini Datei genutzt werden

**Risiko:**

Sicherheitslücken 🡪 Passwörter nicht richtig verschlüsselt 🡪 Verschlüsselung

Passwort vergessen 🡪 kein Zugriff mehr auf Passwörter 🡪 Option Passwort vergessen

**Projektziele:**

1. Sichere Passwörter: Passwort-Manager generieren und speichern starke, einzigartige Passwörter für jede Website, was die Sicherheit erhöht

2. Zeitersparnis: Man muss sich keine Passwörter mehr merken oder manuell eingeben, was den Anmeldeprozess beschleunigt

3. Zentrale Verwaltung: Alle Passwörter werden an einem sicheren Ort gespeichert und sind leicht zugänglich, was die Verwaltung vereinfacht.

**Stakeholder:**

Entwickler, Nutzer, Unternehmen und Gesetzgeber

**Randbedingungen:**

Rechtliche Anforderungen, Datenschutzanforderungen, kleines Team, begrenzte Zeit, kein Budget