PASSWORT-MANAGER

Nils, Joel, Matthis, Nico



INHALT

- 1. Projektziele
- 2. Rollenaufteilunc
- 3. Technische Aufgabenverteilung
- 4. Projektmanagement
- 5. Aufbau Datenbank
- 6. Vorstellung

- 7. GUI & Navigatior
- 8. Verschlüsselung & Speicherung
- 9. Passwort-Generierung
- 10. Passwort-Sicherheitscheck
- 11. Zukunftsideen
- 12. Unser Fazit



Datenbank

Speicherung der Daten in einer lokalen .DB-Datei

Sicherheit

Sichere Speicherung und Verwaltung der Daten

Benutzeroberfläche

Speichern, Bearbeiten und Löschen von Einträgen

Sicherheitsbewertung

Bewertung des eingegebenen Passworts

Generator

Generierung eines Sicheren Passworts

Wartbarkeit

Gute Wartbarkeit durch Kommentare und docstrings



Rollenaufteilung





Zeitwächter

Joel

Überwachung der Zeitplanung



Moderator

Matthis

Koordination der Zusammenarbeit



Protokollant

Nico

Dokumentation der Sitzungen



Statusmanager

Nils

Fortschrittsüberwachung

Technisce Aufgabenverteilung



Projektmanagement

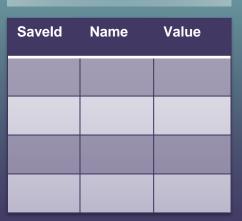
Meilenstein	Geplantes Datum	Erreicht am	Notizen
Projektstart	08.05.2025	08.05.2025	Kickoff-"Meeting"
Datenbankmodell fertig	15.05.2025	15.05.2025	Entwurf & SQL- Datei
GUI-Prototyp	22.05.2025	23.05.2025	Screenshot HERE
Funktionale Integration	22.05.2025	03.05.2025	Erste Testversion
Testphase	29.05.2025	03.06.2025	Protokoll Testfälle
Präsentation / Abgabe	11.06.2025	11.06.2025	PowerPoint-Datei, Lastenheft & Pflichtenheft

AUFBAU - DATENBANK

VALUES

SAVES

Valueld	Name	Key







GUI & Navigation





Verschlüsselung & Speicherung

```
# Class to save all local settings for the program
class ProgramSettings:
    DEFAULT_CRYPT_KEY = "?14![e_-_~:[C8oZO#Y3K,z53Mb$#2x6"
    DATABASE_PATH = "app_database.db"
    CRYPT_KEY = DEFAULT_CRYPT_KEY
```

Passwort-Generierung

```
@staticmethod
def Generate(length):
   Generate a secure random password containing letters, digits, and punctuation.
   Parameters:
    length (int): The desired length of the generated password.
   Returns:
   str: A randomly generated password string of the specified length.
   # Define the character pool: uppercase, lowercase letters, digits and punctuation
   characters = string.ascii letters + string.digits + string.punctuation
   # Use secrets.choice for cryptographically secure random selection
   password = ''.join(secrets.choice(characters) for in range(length))
   return password
```





Passwort-Sicherheitscheck

```
# Check for minimum length
if len(password) < 8:
   messagebox.showwarning(self.DEFAULT INFO HEAD, f"{self.DEFAULT INFO} Passwort zu kurz (mind. 8 Zeichen).")
    return False
# Check for at least one uppercase letter
if not re.search(r"[A-Z]", password):
   messagebox.showwarning(self.DEFAULT INFO HEAD, f"{self.DEFAULT INFO} Mindestens ein Großbuchstabe fehlt.")
    return False
if not re.search(r"[a-z]", password):
    messagebox.showwarning(self.DEFAULT INFO HEAD, f"{self.DEFAULT INFO} Mindestens ein Kleinbuchstabe fehlt.")
    return False
if not re.search(r"[0-9]", password):
   messagebox.showwarning(self.DEFAULT INFO HEAD, f"{self.DEFAULT INFO} Mindestens eine Zahl fehlt.")
    return False
if not re.search(r"[!@\#$%^&*()_+\-=\[\]{};:'\"\],.<>/?]", password):
   messagebox.showwarning(self.DEFAULT INFO HEAD, f"{self.DEFAULT INFO} Mindestens ein Sonderzeichen fehlt.")
    return False
return True
```

Zukunftsideen



Backups

Backups der Datenbank erstellen und separat speichern



Zuletzt Benutze / Favoriten

Direkt die letzten Passwörter / Favorieten erkennen können



Standortfilterung

Passwörter nach Abrufstandort abfragen können







DANKE FÜR DIE AUFMERKSAMKEIT



