Konzept Vorschlag

Projektname:
AronAuthent

Programmierwerkzeug:

- **Programmiersprache:** Java
- **Entwicklungsumgebung:** Visual Studio Code
- **Datenbank:** SQLite

Verschlüsselungsalgorithmus für das Passwort:

- **SHA-256:** Für das Passwort-Hashing, in Kombination mit Salt.
    - **Salt:** Zufällig generierter Wert pro Benutzer.

Weitere Sicherheit:

- **2 Faktor Authentifizierung**
    - **6-stelliger Code** zugesendet per **SMS** (evtl. E-Mail)

DB-Tabellen:

1. **Tabelle 1: Benutzer**
    - **Tabellenname:** users

user_id (INTEGER, Primärschlüssel, Auto-Inkrement), username (TEXT, Eindeutig), email (TEXT, Eindeutig)

password_hash (TEXT), salt_id (INTEGER, Fremdschlüssel, verweist auf die Salt-Tabelle), phone_number (TEXT)

2fa_enabled (BOOLEAN)

2. **Tabelle 2: Salt**
    - **Tabellenname:** salts

salt_id (INTEGER, Primärschlüssel, Auto-Inkrement), salt_value (TEXT)

3. **Tabelle 3: OTPs**
    - **Tabellenname:** otps

otp_id (INTEGER, Primärschlüssel, Auto-Inkrement), user_id (INTEGER, Fremdschlüssel, verweist auf die users-Tabelle)

otp_value (TEXT), expires_at (DATETIME)

Passwortstärke-Validierung:
Die Passwortstärke wird bei der Registrierung überprüft.

Kriterien:

Mindestlänge von 8 Zeichen, Mindestens ein Großbuchstabe, Mindestens ein Kleinbuchstabe,

Mindestens eine Ziffer, Mindestens ein Sonderzeichen

Testszenarien:

1. **Test der Passwortvalidierung:**
    - **Ziel:** Überprüfe, ob schwache Passwörter korrekt abgelehnt und starke akzeptiert werden.
    - **Methode:** Führe mehrere Registrierungsversuche mit schwachen und starken Passwörtern durch.
2. **Test der 2FA-Verifizierung:**
    - **Ziel:** Teste, ob der SMS-Versand korrekt funktioniert und der Benutzer den OTP richtig eingeben muss.
    - **Methode:** Simuliere einen erfolgreichen Login und prüfe, ob der OTP korrekt überprüft wird.