**Dokumentation Python Projekt Januar:**

**Codefunktionen im Detail:**

**1.Importe und Initialisierung**

* **Importe**:
  + sqlite3: Für die Interaktion mit der SQLite-Datenbank.
  + datetime und timedelta: Für die Handhabung von Datum und Zeit.
  + ui von nicegui: Für die Erstellung der Benutzeroberfläche.
  + Image von PIL: Für die Bildverarbeitung (z.B. Konvertierung von Profilbildern).
* **Datenbankinitialisierung**:
  + Die Funktion init\_db() wird aufgerufen, um die SQLite-Datenbank zu initialisieren. Sie erstellt zwei Tabellen:
    - users: Speichert Benutzerinformationen wie Name, Profilbild, Klasse, Rasse, Erfahrung, Level und Gold.
    - tasks: Speichert Aufgaben mit ID, Name, Schwierigkeitsgrad, Fälligkeitsdatum, Status und dem zugehörigen Benutzer.

**2. Klassen**

* **User-Klasse**:
  + **Initialisierung**: Ein Benutzerobjekt wird mit Name, Profilbild, Klasse, Rasse, Erfahrung, Level und Gold erstellt.
  + **Methoden**:
    - load\_user(name): Lädt einen Benutzer aus der Datenbank anhand des Namens.
    - save\_to\_db(): Speichert den Benutzer in der Datenbank oder aktualisiert ihn, falls er bereits existiert.
* **TaskManager-Klasse**:
  + **Initialisierung**: Lädt alle Aufgaben aus der Datenbank.
  + **Methoden**:
    - load\_tasks(): Lädt alle Aufgaben aus der Datenbank.
    - get\_tasks(user\_name, status): Gibt Aufgaben eines bestimmten Benutzers mit einem bestimmten Status zurück.
    - add\_task(name, difficulty, due\_date, user\_name): Fügt eine neue Aufgabe hinzu.
    - complete\_task(task\_id): Markiert eine Aufgabe als erledigt und gibt die Schwierigkeit zurück.
    - delete\_task(task\_id): Löscht eine Aufgabe.
* **TaskApp-Klasse**:
  + **Initialisierung**: Erstellt eine Instanz von TaskManager und initialisiert einige UI-Elemente.
  + **Methoden**:
    - load\_user(name): Lädt einen Benutzer oder erstellt einen neuen, falls er nicht existiert.
    - update\_user\_class\_race(user\_class, user\_race): Aktualisiert die Klasse und Rasse des Benutzers.
    - update\_user\_info(): Aktualisiert die Benutzerinformationen in der UI.
    - add\_experience(points): Fügt Erfahrungspunkte hinzu und erhöht das Level, falls genug Erfahrung gesammelt wurde.
    - add\_task(name, difficulty, due\_date): Fügt eine neue Aufgabe hinzu.
    - complete\_task(task\_name): Markiert eine Aufgabe als erledigt und fügt Erfahrungspunkte hinzu.
    - delete\_task(task\_name): Löscht eine Aufgabe.
    - clear\_ui(): Löscht die UI-Elemente.
    - display\_open\_tasks(): Zeigt offene Aufgaben an.
    - display\_completed\_tasks(): Zeigt abgeschlossene Aufgaben an.

**3. Hauptprogramm (main-Funktion)**

* **Initialisierung**:
  + Die Datenbank wird initialisiert (init\_db()).
  + Eine Instanz von TaskApp wird erstellt.
* **Benutzeroberfläche**:
  + **Benutzernameneingabe**: Ein Eingabefeld und ein Button, um den Benutzer zu laden oder zu erstellen.
  + **Profilbildauswahl**: Ein Button, um ein Profilbild auszuwählen und zu speichern.
  + **Klasse und Rasse**: Dropdown-Menüs zur Auswahl der Klasse und Rasse des Benutzers.
  + **Aufgabenerstellung**: Eingabefelder und Buttons, um neue Aufgaben hinzuzufügen.
  + **Aufgabenverwaltung**: Eingabefeld und Buttons, um Aufgaben zu erledigen oder zu löschen.
  + **UI-Clear**: Ein Button, um die UI-Elemente zu löschen.
  + **Aufgabenanzeige**: Buttons, um offene und abgeschlossene Aufgaben anzuzeigen.

**4. Programmstart**

* Das Programm wird gestartet, indem die main()-Funktion aufgerufen wird. Die Benutzeroberfläche wird mit ui.run() angezeigt.

**Zusammenfassung**

Das Programm ist eine einfache Aufgabenverwaltungsanwendung, die es Benutzern ermöglicht, sich anzumelden, ihr Profil zu verwalten, Aufgaben hinzuzufügen, zu erledigen und zu löschen. Die Benutzerdaten und Aufgaben werden in einer SQLite-Datenbank gespeichert. Die Benutzeroberfläche wird mit nicegui erstellt und bietet eine intuitive Bedienung.

**Aufteilung der Programmierungsarbeit:**

Sebastian: Erstellung des Konzeptes für den Code und die Umsetzung der grundlegenden Funktionen, Optimierungen am Code, Implementierung von der (app.db) SQL-Datenbank und der Interaktion zwischen dieser und dem Skript

Noah: Erstellung des Konzeptes für das Design der grafische Benutzeroberfläche, Verbesserung des Levelsystems, Belohnung (Gold) hinzugefügt, Optische Optimierungen des Codes Farbe der Schwierigkeit/Aufgabe hinzufügen Knöpfe

**Aufgetretene Probleme:**

-Notify bzw. das Ausgeben der Werte aus der DB zB. Name,

-Profilbildanzeige Bild und

-allgemein das umständliche Layout in Nicegui