DHBW-Lörrach Python Projekt Projektdokumentation

Dozent: Hr. Laage-Witt

Projekt-Beschreibung:

Nach persönlicher Absprache beim Feedback der mündlichen Präsentation wurde auf Nachfrage keine Nachholung der Projekt-Beschreibung verlangt.

Zielplattform:

Windows, Mac und alle gängigen Linux-Distributionen.

Externe Komponenten:

Keine externen Komponenten für die Ausführung/Anwendung benötigt.

Modulauflistung und -beschreibung:

tkinter	„tkinter“ steht für „tk interface“ und ist das erste GUI-Toolkit für Python. Mit tkinter lassen sich verschiedene widgets in die erstellten Fenster einbinden. Des weiteren verfügt tkinter über 3 Layout-Manager: „Pack()“, „Grid()“ und „Place()“ mit denen die Anordnung der widgets bestimmt werden kann.
localStorage	„localStorage“ erlaubt das persistente Speichern von Daten auf einem lokalen Datenträger. Dabei kann das entsprechende Storage-Backend auf „text“, „sqlite“ oder „json“ basieren. Die Informationen werden in Form von key-value Paaren gespeichert. https://pypi.org/project/localStoragePy/
Pillow	„Pillow“ ist ein „Fork“ von „PIL“ Pythons Imaging Library. Es unterstützt viele Dateiformate und ermöglicht mit schnellen Bildverarbeitungsfähigkeiten Zugriff auf gespeicherte Bildinhalte.

Klassenauflistung und -beschreibung:

Application()	Die Klasse „Application“ enthält das „root“-Fenster, sowie alle Funktionen die darauf Zugreifen oder neue Fenster erzeugen wollen. Das Objekt hält ebenso globale Schaltervariablen, die aktiv in Funktionsprozessen benutzten Klassenobjekte „Box()“ und „Card()“. Hinzu kommen neben den „tkinter“ widgets des GUI auch ihre Zugriffsvariablen. Damit enthält die „Application“-Klasse alle Funktionen, die für die Prozessroutine verantwortlich sind, ausgenommen von den auf den lokalen Speicher zugreifenden Funktionen.
Box()	Die Klasse „Box“ repräsentiert eine Einheit einer Merkbox und listet all ihre Eigenschaften auf. Dazu gehört ein Name, ein Kartenspeicher („card_dict“) in „Dictionary“-Form und weitere noch ungenutzte Eigenschaften, wie die Anzahl der eigenen Karten und der dazugehörigen Methode.
Card()	Die Klasse „Card“ repräsentiert eine Einheit einer Karteikarte und listet alle ihre Eigenschaften auf. Dazu gehören ihr Name, ihre Frage, Lösung, sowie noch unbenutzte Informationen, wie das Tier-Level das den Fächerfortschritt abbildet, sowie die Anzahl der Versuche und ihre Erfolge.

Auflistung der Abhängigkeiten:

Die Abhängigkeiten werden zusätzlich in der „requirements.txt“-Datei gespeichert.
Weiteres: Siehe Externe Komponenten u. Module. (s. oben)

Zusammenfassung des Resultats:

Einen Fehler den ich recht spät bemerkt habe, war die Unterstützung von json im Modul „localStorage“. Den lokalen Speicher also von sqlite auf json umzustellen hätte mir womöglich die Umwandlung der zu speichernden oder lesenden Inhalte sparen können.
Ein Anzeigefehler im Startbildschirm lässt den Rahmen nicht gänzlich erscheinen. Dieser wird womöglich durch einen darüber liegenden Frame verdeckt. Das Problem wurde kaschiert.
Eine weitere Herausforderung war das Arbeiten mit „Dictionaries“ in „Dictionaries“, da nur ein Key-Value Abruf, jedoch keine Iteration möglich war. Gelöst habe ich das Problem, indem ich das „Dictionary“ in eine Liste gecastet und darüber iteriert und mit dem entsprechenden Schlüssel den Value im tieferliegenden „Dict“ abgerufen habe.
Ein weiterer Fehler war die zu groß werdende Application-Klasse. Ich hätte sie lieber in eine Application-Klasse mit Prozessfunktionen und in eine Window-Klasse mit den GUI Inhalten aufteilen sollte, doch das viel mir zu Beginn schwer aufgrund der ständigen Abhängigkeiten von Variablen und Widgets.