Projektdokumentation

[Cheetah / Nàray, Casadei, Becirovic, Delvecchio]

Datum	Version	Änderung	Autor
3.11.2021	0.0.1	Planung und Anfang der Implementation	Gruppe Cheetah
10.11.2021	0.1.0	Erste Version (Wort kann geladen und abgefragt werden)	Gruppe Cheetah
10.11.2021	0.2.0	Korrigierte Version	Gruppe Cheetah
17.11.2021	1.0.0	Finale Version (nochmal korrigiert und erweitert)	Gruppe Cheetah

1. Informieren

1.1 Ihr Projekt

Wir entwickeln eine Konsolenapplikation zum Vokabeln lernen. Diese besteht aus mehreren Elementen und kann mehrere Vokabellisten abfragen.

1.2 Quellen

Wie überschreibt / liest man ein .txt Dokument in einer Konsolenapplikation
https://docs.microsoft.com/en-us/troubleshoot/dotnet/csharp/read-write-text-file

Verwendung von System.IO
https://docs.microsoft.com/de-ch/dotnet/api/system.io?view=net-5.0

1.3 Anforderungen

Nummer	Muss / Kann?	Funktional? Qualität? Rand?	Beschreibung
1	M	F	Vokabular muss in einer .txt-Datei abgespeichert werden.
2	M	F	Vokabular muss aus einer .txt-Datei abgelesen werden.
3	M	Q	Mehrere Sprachen werden unterstützt, indem Benutzer selber die Übersetzung eingibt.
4	K	Q	Falsch beantwortete Wörter werden noch einmal abgefragt.
5	M	F	Nach der Eingabe einer Übersetzung gibt das Programm ein Feedback (nur falls falsch, sonst nichts)
6	K	R	Lernblocks je 10 Wörter machen
7	K	Q	Statistiken pro Lernblock werden gezeigt
8	K	Q	Das Programm speichert automatisch bei welchem Lernblock man aufgehört hat, so kann man dort weitermachen, wo man aufgehört hat.
9	K	R	Es wird einem eine Note am Schluss ausgerechnet (lineare Skala, 4er bei 60%)
10	K	Q	Zufälliger Motivationsspruch nach jedem Block
11	M	Q	Anfällige Fehler Auffangen (try catch)
12	K	Q	Ausgaben vom Programm sind dick geschrieben (Bold)
13	K	Q	Benutzer kann mehrere Listen erstellen, diese werden gespeichert.
14	M	F	Benutzer kann wählen ob von DE zu Fremd. oder von Fremd. zu DE
15	K	R	Highscore System
16	K	R	Nach jedem Block ab und zu ein Lerntipp
17	K	R	Manchmal Witze, wenn man etwas falsch schreibt

* Verwenden Sie für Ihre Anforderungen ganze Zahlen. Die Beschreibung entspricht der folgenden Formel:

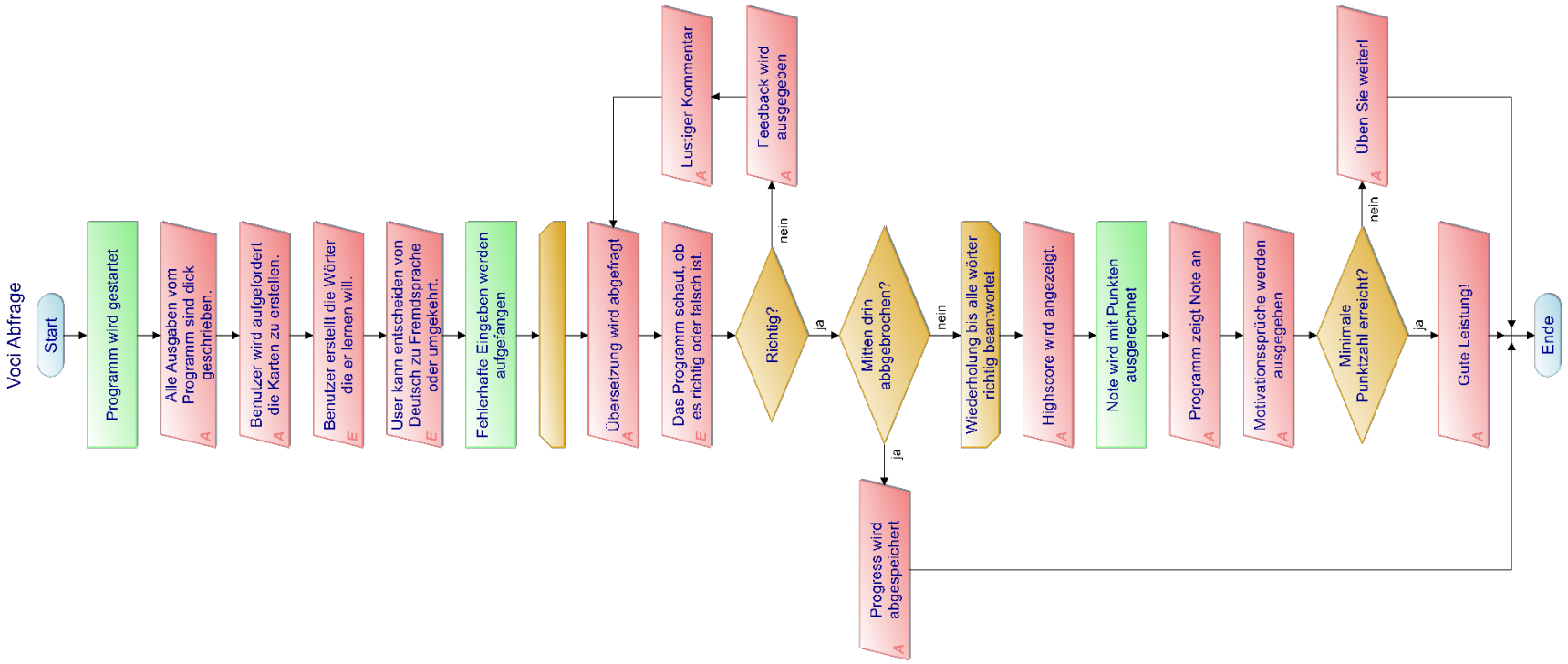
Zielsystem + Priorität + Systemaktivität + Funktionalität + Bedingungen

- **Zielsystem** ist das zu entwickelnde System
- **Priorität** wird durch *muss* bei hoher und durch *soll* bei niedriger Priorität angegeben
- **Systemaktivität** beschreibt, ob das System selbständig handeln soll, einem Benutzer eine Funktion anbieten soll oder einer Schnittstelle bedarf. Wählen Sie die Formulierung *dem Administrator o. ä. die Möglichkeit bieten* für Benutzerfunktionen, und *fähig sein* für Schnittstellenanforderungen.
- Zeitliche **Bedingungen** werden mit *wenn*, und logische Bedingungen mit *falls* beschrieben

Beispiel: Reflect Media Player (Zielsystem) muss (= höhe Priorität) dem Benutzer die Möglichkeit bieten (= Benutzerinteraktion), J3D Szenengraphen aus einer wrml-Datei über das Netzwerk zu laden (Funktionalität), falls der Benutzer eingeloggt ist.

⚠ Falls sich Ihre Anforderungen nicht aus dem Auftrag ergeben, halten Sie unter **3. Entscheiden** fest, welche Anforderungen Sie warum selbst sich gestellt haben.

1.4 Diagramme



1.5 Testfälle

Nummer	Vorbereitung	Eingabe	Erwartete Ausgabe
1.1	Programm gestartet	[ENTER]	Keine, aber die Lernblöcke sollen gespeichert werden.
2.1	Programm gestartet	Wort X	Keine, aber die Wörter sollen pro Lernblock ausgelesen werden.
3.1	Programm gestartet und Wörter werden eingeben.	Wort X	Wort Richtig oder falsch
4.1	Programm gestartet	Falsches Wort eingeben	Wort Falsch
5.1	Programm gestartet und 10 Wörter eingeben	Keine	Nach 10 Wörter gibts ein Feedback
6.1	Programm gestartet und einen Lernblock absolviert.	Keine	Lernblock gibt Statistiken zum Lernblock aus.
7.1	Programm gestartet und mitten im Lernblock abgebrochen. Danach das Programm wieder starten.	Keine	Programm startet wieder bei Lernblock X
8.1	Programm gestartet und alle Lernblöcke abgeschlossen.	Alles	Note X sollte erscheinen und ausgerechnet werden.
9.1	Programm gestartet und einen Lernblock abgeschlossen	Alle Wörter vom Lernblock eingeben.	Zufälliger Motivationsspruch wird ausgeben.

2. Planen

Nummer	Frist	Beschreibung	Zeit in min. (geplant)
alle	28.10.	Alles wird grob in ein PAP eingetragen	45
1.0, 2.0, 3.0	28.10.	Planen der Eingabe, Abspeicherung und Ablesung in PAP	45
1.1, 2.1, 3.1	28.10	Testfälle für Eingabe, Abspeicherung und Ablesung	
1.2, 2.2, 3.2	3.11.	Programmierung	45
1.2, 2.2, 3.2	3.11.	Debuggen des vorhandenen Codes, um Weiterarbeit zu erleichtern und Programmierung weiterer Anforderungen	15
4.1, 5.1, 11.1, 14.1	3.11.	Realisieren aller restlichen Anforderungen, welche ein Muss sind.	45
"	"	"	"
"	"	"	"
4.2, 5.2, 11.2, 14.2	3.11.	Debuggen des vorhandenen Codes	45
-	10.11.	Programmierung aller Anforderungen, welche ein Kann sind.	45
"	"	"	"
"	"	"	"
alle	17.11.	Debuggen und Testen von allem.	135 (3 Lektionen)
"	"	"	"
"	"	"	"
		TOTAL:	Sitzungen × Lektionen × Gruppenmitglieder

* Die Nummer hat das Format N.m, wobei N die Nummer der Anforderung ist, zu der das Arbeitspaket gehört, und m von 1 an fortlaufend durchnummeriert wird.

** Teilen Sie diesmal Ihre Anforderungen in 45-Minuten-Arbeitspakete ein.

3. Entscheiden

Unsere Gruppe hat sich entschieden, das Vokabular-Abfrage-Programm zu machen.

Für welchen Ablauf wir und entschieden haben, sieht man bei **1.4**.

Wir haben uns auch dazu entschieden das .csv-Format für die Speicherdatei zu verwenden. So ist es leichter Arrays zu erstellen und diese abzulesen.
Jedoch werden wir sehr wahrscheinlich etwas anderes für die Abspeicherung des Fortschrittes brauchen.

4. Realisieren

Nummer	Datum	Beschreibung	Zeit (geplant)	Zeit (effektiv)
1.2, 2.2, 3.2	3.11.	Programmierung	45	90
1.2, 2.2, 3.2	3.11.	Debuggen des vorhandenen Codes, um Weiterarbeit zu erleichtern und Programmierung weiterer Anforderungen	15	30
1.0, 2.0, 3.0	3.11.	PAP korrigiert	0	15
1.2, 2.2, 3.2	10.11	Programmierung	45	90
1.2, 2.2, 3.2	10.11.	Debuggen des vorhandenen Codes, um Weiterarbeit zu erleichtern und Programmierung weiterer Anforderungen	15	45
4.1, 5.1, 11.1, 14.1	10.11.	Realisieren aller restlichen Anforderungen, welche ein Muss sind.	45	90
4.2, 5.2, 11.2, 14.2	10.11.	Debuggen des vorhandenen Codes	45	45
-	10.11.	Programmierung aller Anforderungen, welche ein Kann sind.	45	übersprungen
alle	10.11.	Debuggen und Testen von allem.	135 (3 Lektionen)	90
alle	17.11.	Debuggen und Testen von allem.	135 (3 Lektionen)	45

[Übernehmen Sie Ihre Planung aus 2., und tragen Sie nach, wie lang Sie effektiv zur Bearbeitung der jeweiligen Arbeitspakete benötigt haben.]

5. Kontrollieren

5.1 Testprotokoll

Nummer	Datum	Resultat	Durchgeführt
1.1	17.11.2021	Funktioniert	Becirovic
2.1	17.11.2021	Funktioniert	Becirovic

3.1	17.11.2021	Funktioniert	Becirovic
4.1.	17.11.2021	Funktioniert	Becirovic
51.	17.11.2021	Funktioniert	Becirovic
61.	17.11.2021	Funktioniert	Becirovic
7.1	17.11.2021	Funktioniert	Becirovic
8.1	17.11.2021	Funktioniert	Becirovic
9.1	17.11.2021	Funktioniert	Becirovic

Fazit:

Das Testen lief generell sehr gut. Wir hatten nur ein Problem mit den voreingestellten Wörtern. Es blieb immer beim siebten Wort hängen, sobald man selbst die Wörter geschrieben hat, hat alles funktioniert.

6. Auswerten

- Die gemeinsame Programmierung der Funktionen ist gut gelaufen. Wir konnten sie sauber im Hauptprogramm dann einbauen
- Wir konnten anfangs leider nicht kooperativ zusammenarbeiten. Später ging das dann besser.