Konzeptdokument

„Entwicklung eines Dashboards für dein Studium“

Kursbeschreibung:
DLBDSOOFPP01_D – „Objektorientierte und funktionale Programmierung mit Python“

Studiengang:
Angewandte Künstliche Intelligenz, Bachelor of Science (B.Sc.)

Verfasser:
Phillip Riemer
Solmsstraße 25
60486 Frankfurt am Main
phillip.riemer@iu-study.org

Matrikelnummer:
IU14128175

1. Festlegen der Ziele

Das Dashboard dient der kontinuierlichen Überwachung meiner Studienziele, um frühzeitig Abweichungen zu erkennen und gegensteuern zu können. Dabei stehen folgende Ziele im Fokus:

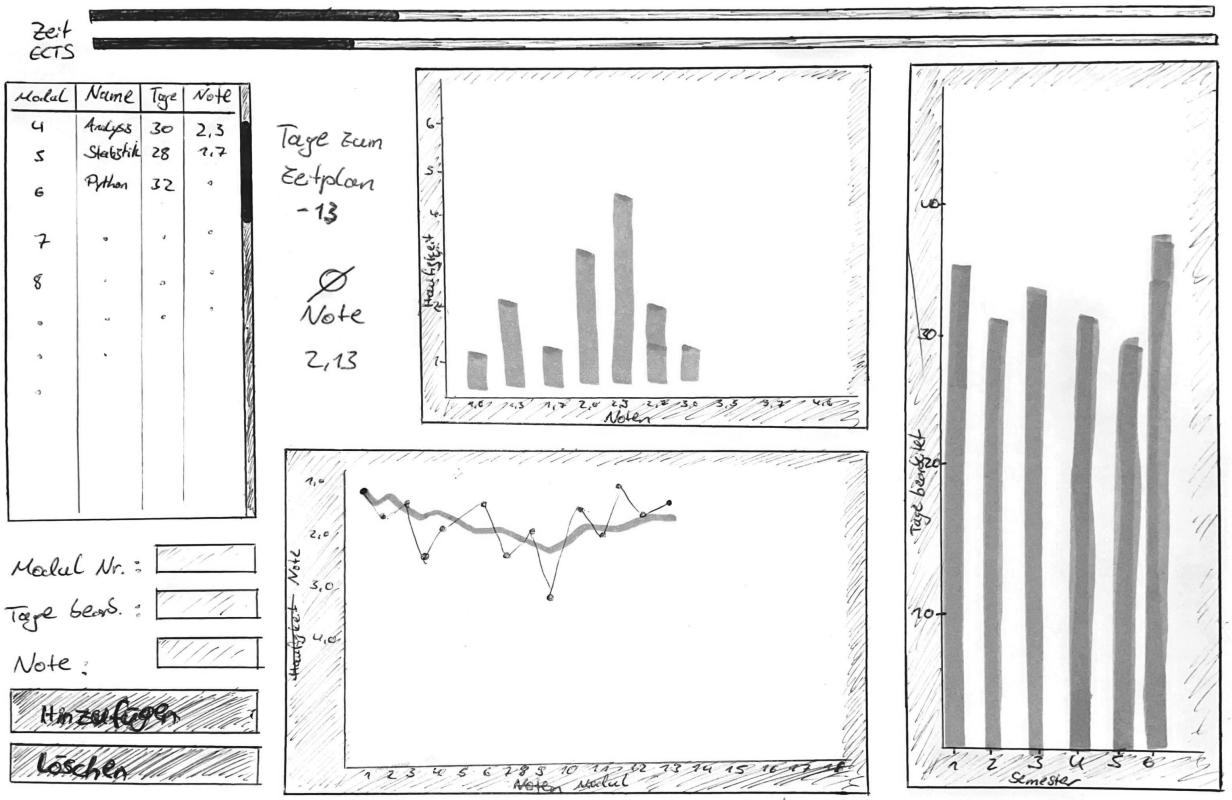
Ein Ziel ist der Studienabschluss innerhalb von drei Jahren, also bei mir konkret bis spätestens zum 11.01.2028. Das Dashboard unterstützt dieses Ziel, indem es den Studienfortschritt in ECTS-Punkten der vergangenen Zeit gegenüberstellt.

Mein zweites Ziel besteht darin, die durchschnittliche Bearbeitungszeit pro Modul bei Modulen mit 5 ECTS-Punkten auf eine durchschnittliche Bearbeitungsdauer von einem Monat zu begrenzen. Die Module werden von mir nacheinander bearbeitet und der tatsächliche Bearbeitungszeitraum durch den Modulbeginn bis zum Prüfungsdatum dokumentiert. Das Dashboard unterstützt mich beim Monitoring dieses Ziels, indem es diese Daten aufnimmt und die durchschnittliche Bearbeitungszeit für ein Modul je nach Semester berechnet und im Balkendiagramm ausgibt. So kann ich möglicherweise einen Trend erkennen.

Mein drittes Ziel ist ein Notendurchschnitt von mindestens 1,7 zum Studienabschluss. Das Dashboard zeigt hierzu den aktuellen Durchschnitt aller bisher erzielten Noten und bereitet gleichzeitig alle erreichten Noten und den daraus abgeleiteten gleitenden Durchschnitt chronologisch visuell auf. So ist auf einen Blick erkennbar, ob das gesetzte Ziel bisher eingehalten wurde.

2. Festlegen der angezeigten Inhalte auf dem Dashboard

Folgende Skizze zeigt meinen Entwurf und die Inhalte des Dashboards.



Am oberen Rand befindet sich ein horizontales Balkendiagramm, das den Fortschritt anhand erreichter ECTS-Punkte und der bereits vergangenen Kalendertage ab Start des Studiums innerhalb der geplanten Studiendauer von drei Jahren prozentual gegenüberstellt. So erhalte ich schnell einen Überblick, ob ich hinter meinem Zeitplan bin oder sogar schneller vorankomme. Darunter wird der aktuelle Zeitversatz in Tagen angegeben, gemessen als Differenz zwischen dem tatsächlichen Studienfortschritt und dem Soll-Fortschritt auf Basis des Startdatums 12.01.2025.

Rechts zeigt ein vertikales Balkendiagramm die durchschnittliche Bearbeitungsdauer eines Moduls mit 5 ECTS-Punkten je Semester. Die Darstellung ermöglicht direkt zu prüfen, ob das Ziel von maximal einem Monat Bearbeitungszeit pro Modul mit 5 ECTS-Punkten eingehalten wird. Als Eingabedaten dienen das Datum des Bearbeitungsstarts eines Moduls und das Datum der jeweils bestandenen Prüfungsleistung. Die Differenz wird in Kalendertagen berechnet.

Zentral im Dashboard befindet sich ein Balkendiagramm zur Notenverteilung. Es stellt dar, wie häufig bestimmte Noten erreicht wurden. Direkt links daneben wird der aktuelle Notendurchschnitt als Einzelwert angezeigt.

Am unteren Rand zeigt mittig ein Liniendiagramm den Notenverlauf, chronologisch nach Prüfungsdatum sortiert. Die dünne Linie markiert die Einzelleistungen, während die fette Linie den gleitenden Durchschnitt darstellt. Dadurch sollten Trends gut erkennbar sein.

Links ist eine Datentabelle integriert, in der zu jedem bereits erfassten und absolvierten Modul nur die wichtigsten Daten angezeigt werden, der Modulname, die Bearbeitungsdauer in Tagen und die Note. Unterhalb der Tabelle befinden sich Eingabefelder für die Moduldaten. Mit den Schaltflächen „Hinzufügen“ und „Löschen“ können Datensätze flexibel verwaltet werden. Dadurch bleibt das Dashboard kontinuierlich aktuell und individuell anpassbar.

3. Erstellen eines UML-Klassendiagramm

Das UML-Klassendiagramm stellt die wesentlichen Entity-Klassen für mein Dashboard dar und berücksichtigt objektorientierte Konzepte wie Aggregation, Komposition und Assoziation.

Die Klasse Student ist über eine Aggregation mit dem Studiengang verbunden. Diese Beziehung erlaubt es, Studierende einem Studiengang zuzuordnen, ohne dabei deren Eigenständigkeit aufzugeben. Die Methode setze_student() ermöglicht die Initialisierung oder Anpassung dieser Verbindung.

Die Klasse Studiengang steht in einer Kompositionsbeziehung zur Klasse Semester, da ein Semester ohne zugehörigen Studiengang nicht existiert. Innerhalb eines Studiengangs besteht eine feste Zuordnung der Semester, wodurch eine starke Beziehung bestehen muss.

Jedes Semester wiederum steht nur in einer Aggregationsbeziehung zur Klasse Modul, da ein Großteil der Module im Rahmen meines Studienplans nicht in einem konkreten Semester verankert sind,

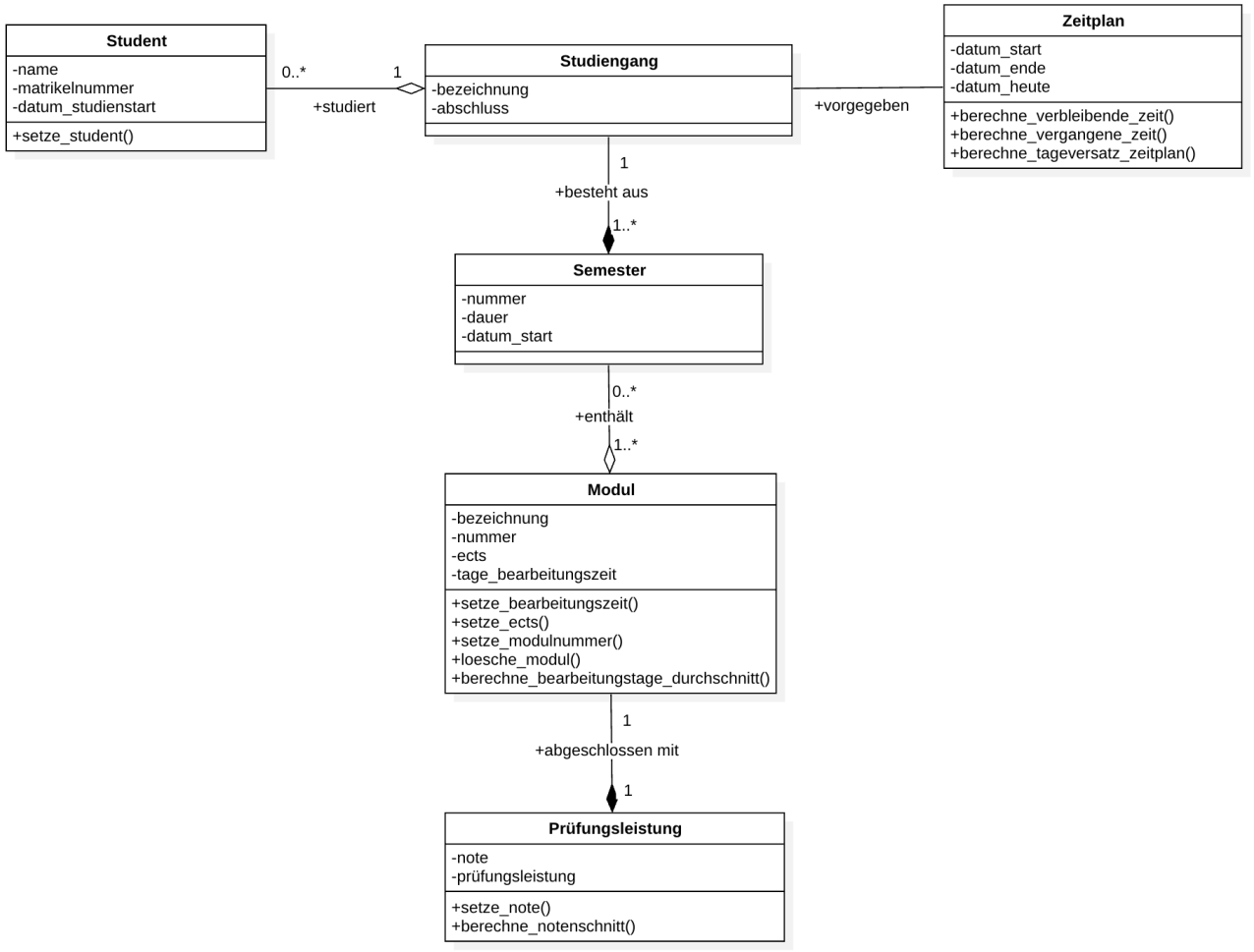
aber trotzdem ohne ein Semester nicht existieren können. Die Module enthalten Attribute wie Bezeichnung, ECTS-Punkte und Bearbeitungsdauer und sind fester Bestandteil der Studienstruktur.

Ein Modul ist über Komposition mit der Klasse Prüfungsleistung verbunden. Diese Beziehung spiegelt wider, dass Prüfungsleistungen nur im Kontext von jeweils einem spezifischen Modul existieren. Die Klasse Prüfungsleistung verwaltet Note und Prüfungsform und enthält Methoden wie berechne_notenschnitt(), die eine wichtige Rolle im Monitoring übernehmen.

Die unterstützende Klasse Zeitplan ist über eine Assoziation mit dem Studiengang verbunden und enthält Zeitparameter. Mit Methoden wie berechne_verbleibende_zeit() oder berechne_tageversatz_zeitplan() lassen sich Abweichungen vom geplanten Studienverlauf berechnen. Der Zeitplan bezieht sich auf die gesamte Studiendauer.

Die Methodennamen sind konsistent und eindeutig gewählt, um die spätere Implementierung zu erleichtern. Setter-Methoden wie setze_ects() oder setze_bearbeitungszeit() machen die Struktur klar und trennen Zustand von Verhalten. Insgesamt bietet das Klassendiagramm eine übersichtliche, logisch konsistente und objektorientiert saubere Grundlage für die spätere Umsetzung des Dashboards mit Python.

Folgende Grafik zeigt das UML-Diagramm der Entity-Klassen.



4. Machbarkeitsprüfung und Erproben von Umsetzungsmöglichkeiten mit Python

Als Entwicklungsumgebung wurde Visual Studio Code gewählt. Sie bietet zahlreiche hilfreiche und kostenlose Erweiterungen, darunter automatische Syntaxprüfung, intelligente Codevervollständigung und Funktionsbeschreibungen per Mouseover. Darüber hinaus ermöglicht der integrierte Datei-Explorer eine übersichtliche und intuitive Verwaltung aller Projektdateien.

Zum Einsatz kommen verschiedene Bibliotheken. Pandas wird verwendet, um tabellarische Daten effizient einzulesen, zu filtern und weiterzuverarbeiten. Für statistische Auswertungen und Verteilungen dienen scipy.stats und numpy. Die grafische Darstellung von Daten erfolgt mit matplotlib. Für den Aufbau des grafischen Benutzerinterfaces wird tkinter genutzt, damit lassen sich Benutzeroberflächen mit Buttons, Abständen, Scrollleisten und Tabellen strukturiert und flexibel gestalten.

Für den prototypischen Zweck genügen Text-Dateien und eine CSV-Datei als Datenspeicher. Durch die Eingabe neuer abgeschlossener Module werden jeweils neue Datenzeilen in der CSV-Datei erzeugt. Dieses Dateiformat ermöglicht es auch gezielt spalten anzusprechen, zu berechnen und später zu visualisieren sowie in Diagrammen darzustellen. Listen in Textdateien können für die Speicherung von Stammdaten eines Studenten zum Einsatz kommen.

In dieser frühen Konzeptionsphase verwalte ich den Quellcode auf dem lokalen PC-Speicher mit automatischer OneDrive-Speicherung als BackUp. Das ermöglich flexibles Experimentieren mit Umsetzungsmöglichkeiten in separaten Dateien, die in Ordnern schnell erstellt und gelöscht werden können. Im weiteren Verlauf wird der Quellcode mit GitHub und der entsprechenden Integration für Visual Studio Code verwaltet werden. So können frühere Versionen des Codes wiederhergestellt und verwaltet werden. Außerdem kann hier, wie gefordert, ein Link zum gesamten Portfolio versendet werden.

Die Interaktion mit dem Dashboard erfolgt über eine grafische Oberfläche. Über einfache Eingabefelder und Buttons können neue Daten hinzugefügt, bestehende Einträge gelöscht und die Darstellung des Dashboards aktualisiert werden, wie in Aufgabe 2 dargestellt.

Die technische Machbarkeit wurde bereits in mehreren kleinen Testprogrammen überprüft. Dazu gehörten das Einlesen und Beschreiben von CSV-Dateien, die Erstellung von Diagrammen, die Darstellung von Tabellen sowie der Aufbau eines grafischen Userinterfaces mit Buttons, Eingabefeldern und dynamischer Anordnung der Elemente.