

Frankfurt University of Applied Sciences
Fachbereich 2: Informatik und Ingenieurwissenschaften
Studiengang: Informatik (B.Sc.)



Lernassistent

Design und Implementierung einer KI-basierten Anwendung

Dozent: Prof. Dr.-Ing. Markus Miettinen

Modul 25: Informatik Projekt

Wintersemester: 2024/2025

Team:

Annika Zoe Schäffler - Matrikelnummer: 1453180,

Finn Schäffler - Matrikelnummer: 1350212,

Mohammed Taha El Youssefi - Matrikelnummer: 1454943,

Mohamed Hamzaoui - Matrikelnummer: 1462582,

Nisrine El Youncha - Matrikelnummer: 1464975,

Tim Bornemann - Matrikelnummer: 1431393

1. Inhaltsverzeichnis

1. Inhaltsverzeichnis.....	2
2. Aufgabenstellung.....	4
3. Zielsetzung.....	4
3.1. Dokumentenverarbeitung und Texterkennung.....	4
3.2. KI-gestützte Zusammenfassung und Wissensaufbereitung.....	5
3.3. Lernkarten.....	5
3.4. Lernstrategien.....	5
3.5. Simulierte Prüfungen.....	5
3.6. Sharepoint für Dokumente und Lernkarten.....	5
3.7. Gamification zur Lernmotivation.....	6
4. Umsetzung.....	6
4.1. Frontend.....	6
4.1.1. Planung: Webapplikation.....	6
4.1.2. Technische Komponenten: html, css, javaScript, FontAwesome.....	7
4.1.3. Ansprüche.....	8
4.1.4. Realisierung der Ansprüche.....	8
4.1.5. Design.....	10
4.1.6. Nächste Schritte.....	18
4.1.7. Backendverknüpfung.....	18
4.1.8. Problemanalyse.....	19
4.2. Backend.....	19
4.2.1. Überblick.....	19
4.2.2. Architektur.....	20
4.2.3. Project Structure.....	22
4.2.4. Verwendete Technologien.....	22
4.2.5. Rollen der verschiedenen Klassen.....	24
4.2.6. Anki Service.....	27
4.2.7. Fehler und Erkenntnisse aus dem Projekt.....	30
4.3. Künstliche Intelligenz.....	31
4.3.1. Lokaler Dienst.....	32
4.3.2. Externer Dienst.....	33
4.4. CI/CD-Pipeline für die Multi-Service-App.....	33
4.4.1. Ablauf der CI/CD-Pipeline:.....	34
4.4.2. Technologien & Tools:.....	34
5. User-Handbook.....	35
5.1. How to get started.....	35
5.2. Registrieren.....	36
5.3. Dashboard.....	37
5.4. Navigationsleiste.....	39

5.5. File Manager.....	40
5.6. Anki Cards.....	45
5.7. Ausloggen.....	46
5.8. Split View.....	47
6. Ergebnis/Fazit.....	48
7. Ausblick.....	48
8. Arbeitsaufteilung.....	50
8.1. Alle.....	50
8.2. Frontend-Team: (Annika Zoe und Finn Schäffler).....	50
8.3. Annika Zoe Schäffler - Matrikelnummer: 1453180.....	51
8.4. Finn Schäffler - Matrikelnummer: 1350212.....	51
8.5. Backend-Team: (Taha, Nisrine, Hamzaoui).....	51
8.6. Mohammed Taha El Youssefi - Matrikelnummer: 1454943.....	52
8.7. Mohamed Hamzaoui - Matrikelnummer: 1462582.....	53
8.8. Nisrine El Youncha - Matrikelnummer: 1464975.....	54
8.9. Tim Bornemann - Matrikelnummer: 1431393.....	55

2. Aufgabenstellung

Künstliche Intelligenz wurde in den letzten Jahren zu einer immer prominenter werdenden Technologie und findet in immer mehr Bereichen ihren Einsatz. Sie ermöglicht es, große Datenmengen zu analysieren und daraus automatisiert Muster zu erkennen oder Vorhersagen zu treffen.

Im Rahmen des “Informatik Projekt” Moduls der Frankfurt University of Applied Sciences soll eine KI-gestützte Anwendung entwickelt werden, die eine spezifische Aufgabe automatisiert durchführt.

Die gewählte Anwendungsidee ist ein Lernassistent, der Studierenden dabei helfen soll, sich besser zu organisieren und strukturiert mit Hilfe von Karteikarten zu lernen. Um den Lernprozess angenehmer zu gestalten und die Hürde des Aufraffens zu verkleinern, wird der Lernassistent Künstliche Intelligenz verwenden, um automatisiert Zusammenfassungen von Vorlesungen zu schreiben. Zusätzlich werden automatisch Karteikarten durch die KI erstellt.

Ziel ist es, das Lernen zu optimieren und den Zeitaufwand für das Erfassen und Wiederholen von Vorlesungsinhalten zu reduzieren und das mit Hilfe von Künstlicher Intelligenz.

3. Zielsetzung

Die Zielgruppe des Lernassistenten richtet sich vorrangig an Studierende, wobei auch andere Lernende dieses Tool nutzen können, um ihr Lernen zu vereinfachen, besser zu strukturieren und effizienter zu gestalten.

Mit Studenten als Hauptzielgruppe haben wir verschiedene Funktionen ausgearbeitet, die unstrukturierte Notizen, ineffizientes Lernen oder zeitaufwendige Vorbereitungen der Vergangenheit angehören lassen.

Die Anwendung sollte aus mehreren Modulen bestehen, die miteinander interagieren. Dabei soll Künstliche Intelligenz den Studenten einen Teil ihrer Arbeit abnehmen, große Sprachmodelle (LLMs) werden verschiedene Aufgaben übernehmen.

Um diese Ziele zu erreichen, haben wir großzügig geplant, die folgenden Funktionalitäten zu integrieren:

3.1. Dokumentenverarbeitung und Texterkennung

- Studierende sollen in der Lage sein, Vorlesungsfolien, Skripte und handschriftliche Notizen hochzuladen.
- Mittels OCR werden handschriftliche Dokumente digitalisiert.
- Automatische Fehlerkorrektur und Formatierung der erkannten Inhalte.

3.2. KI-gestützte Zusammenfassung und Wissensaufbereitung

- Die Anwendung nutzt LLMs, um lange Texte automatisch zu analysieren und prägnante Zusammenfassungen zu erstellen.
- Die Nutzer sollen zwischen verschiedenen Detaillierungsstufen wählen können, um entweder eine einfache oder eine ausführliche Erklärung eines Themas zu erhalten.

3.3. Lernkarten

- Auf Grundlage der hochgeladenen und verarbeiteten Dokumente werden automatisch Karteikarten erstellt.
- Studierende können die Karten nach Themenbereichen organisieren und individuell anpassen.
- Ein Algorithmus erkennt, welche Inhalte bereits gut verstanden wurden und welche noch weiter geübt werden müssen.
- Die Lernkarten basieren auf der Selbstreflexion

3.4. Lernstrategien

- Ein Spaced Repetition System, um Wissen in optimalen Zeitabständen zu wiederholen.
- Integration der Pomodoro-Technik, um effektive Lernphasen mit geplanten Pausen zu kombinieren.
- Durch Analyse von Lernverhalten und Leistung soll das System Schwachstellen identifizieren und gezielt Verbesserungsvorschläge geben.

3.5. Simulierte Prüfungen

- Erstellung und Durchführung von Probeklausuren und Quizen
- Dynamische Anpassung des Schwierigkeitsgrads der Fragen an den Lernfortschritt.
- Detaillierte Analyse und Statistiken der Testergebnisse.

3.6. Sharepoint für Dokumente und Lernkarten

- Alle Lernmaterialien, einschließlich hochgeladener Dokumente, Zusammenfassungen und Karteikarten, sollen über eine zentrale SharePoint verwaltet werden.
- Die Daten sollen geräteübergreifend synchronisiert und von überall aus abgerufen werden können.
- Studierende sollen die Möglichkeit haben, ihre Lernressourcen mit Kommilitonen zu teilen.

3.7. Gamification zur Lernmotivation

- Belohnungssystem mit Abzeichen und Erfolgen.
- Ein Leaderboard für den spielerischen Wettbewerb zwischen den Lernenden.
- Fortschritt wird in Diagrammen und Statistiken visualisiert, für eine klare Übersicht über die persönlichen Entwicklungen.
- Regelmäßige Erinnerungen.

4. Umsetzung

4.1. Frontend

Aufgrund der erst kürzlich erfolgten Verknüpfung des Front- und Backends ergaben sich kurzfristig Veränderungen des Frontends. Eine Überarbeitung des Designs, um das ursprüngliche Aussehen wiederherzustellen, kann aufgrund von Zeitmangel nicht mehr stattfinden, findet aber im Abschnitt 4.1.5. Next Steps Erwähnung. Die folgende Dokumentation des zuvor erreichten und wieder angestrebten Frontend-Designs ist daher mithilfe des separaten Github Repositorys: <https://github.com/A-Z-Schaeffler/Lernassistent-Frontend> gut nachzuvollziehen. Es enthält einen älteren Projektstand, in dem die Veränderungen durch die Verknüpfung noch nicht stattgefunden haben. Die in der Dokumentation genannten HTML- und CSS-Dateien sind hier unter src → main → View zu finden.

4.1.1. Planung: Webapplikation

Das Frontend unseres Projekts wurde als Webanwendung geplant und realisiert, da der Benutzer für dessen Nutzung keine großartigen Voraussetzungen erfüllen muss. So erfolgt der Zugang über einen Webbrowser, bei dem lediglich ein Benutzerkonto angelegt werden muss. Es entfällt dadurch die lokale Installation einer Desktopanwendung auf dem Rechner des Benutzers und gleichzeitig wird der Zugang von verschiedenen Rechnern auf die in der Webapplikation hinterlegten Nutzerdaten gewährt. Gerade aus der Perspektive eines Studenten ist eine solche Flexibilität eine wichtige Anforderung an einen nützlichen Lernassistenten.¹

¹ <https://de.wikipedia.org/wiki/Webanwendung>, Stand: 23.01.25

4.1.2. Technische Komponenten: html, css, javaScript, FontAwesome

Der Frontend Teil des Projekts besteht aus einem Zusammenspiel aus verschiedenen technischen Komponenten. Hypertext Markup Language² oder kurz HTML ist der standardisierte Weg, eine webbasierte Anwendung stilistisch darzustellen. Die Sprache kann mit jedem beliebigen Texteditor geschrieben werden und ist einfach zu handhaben. Ein HTML Dokument liegt dann vor, wenn es anfangs als solches deklariert wird. Ansonsten besteht ein Dokument aus `<head>` und `<body>` Containern, die verschiedene Informationen beinhalten. Im `<head>` Container befinden sich generelle Informationen wie zum Beispiel der Titel der Seite oder sogenannte Metadaten. HTML kann keine Sonderzeichen darstellen, es sei denn, sie werden durch `“charset=“utf-8“` deklariert.

HTML Dokumente bringen bereits eine große Auswahl an vorprogrammierten Funktionen mit sich, wie das Erstellen von Buttons und Überschriften. Um das Design dieser Elemente zu bearbeiten, nutzen wir CSS³ Dateien. Diese Stylesheet Sprache ermöglicht es uns zum Beispiel, Hover-Animationen hinzuzufügen, Abstände der Elemente besser zu steuern und das vereinfachte Aussehen je nach Belieben zu erweitern.

Um den Buttons oder anklickbaren Links auch eine Funktion zu übergeben, benutzen wir JavaScript als Frontend Sprache. Dieser Code Abschnitt benötigt keine eigene Datei in unserem Projekt, da HTML einen eigenen Container für JavaScript bereitstellt. Man kann diesen Abschnitt auch in eine eigene Datei packen. Alle externen Dateien, die von unserer HTML benötigt werden (in unserem Fall nur die CSS Dateien), müssen im `<head>` Bereich deklariert werden. Wichtig dabei ist aber zu verstehen, dass Funktionen nur in der eigenen HTML angesprochen werden können, es sei denn man deklariert eine Verbindung beider Komponenten.

Unsere Webanwendung benutzt verschiedene Icons, welche von FontAwesome zur Verfügung gestellt werden. Diese Website erlaubt jedem Nutzer eine Auswahl an kostenlosen Icons, die man nach Belieben in sein Projekt einbauen kann. Man erhält einen Zugangslink, welcher im `<head>` Bereich der HTML Datei deklariert werden muss. Da die Inhalte aus dem Internet geladen werden, benötigt man beim Ausführen des Projekts eine bestehende Internetverbindung. FontAwesome ermöglicht es einem, die verwendeten Icons durch CSS Code auch stilistisch anzupassen (Farbe, Breite, Abstände, etc.).

² <https://en.wikipedia.org/wiki/HTML>, Stand: 03.02.2025

³ https://de.wikipedia.org/wiki/Cascading_Style_Sheets, Stand: 03.02.2025

4.1.3. Ansprüche

4.1.3.1. Dynamische Anpassung der Seitengröße an unterschiedliche Bildschirmgrößen

Um dem Benutzer unabhängig vom verwendeten Bildschirm und dessen Größe ein ansprechendes und einheitliches Design des Lernassistenten zu gewährleisten, muss sich dessen Webseiten-Layout dynamisch an unterschiedliche Bildschirmgrößen anpassen können. Hierfür wurde das Konzept des Responsive Layouts verfolgt, welches die Eigenschaften und Vorteile eines fluiden Layouts mit denen eines adaptiven Layouts verbindet.⁴

4.1.3.2. Splitview

Auch das Arbeiten im Split View soll für Benutzer möglich sein, um parallel im Lernassistenten und anderen Quellen weiterarbeiten zu können. Dies verschafft einen besseren Überblick und gestaltet die Lernroutine komfortabler. Hilfreich ist dies beispielsweise, wenn per Drag and Drop Dateien von einem Split View in den anderen gezogen werden.

4.1.3.3. Dynamisches Laden der Navigationsleiste

Um redundanten Code zu vermeiden, haben wir die Navigationsleiste von den jeweiligen HTML-Dateien separiert und einmal in eine eigene Komponente geschrieben. Die Leiste wird durch eine Fetch-Anfrage im <script> Bereich der einzelnen Seiten angesprochen. Hierbei wird der HTML-Teil in die jeweilige Seite eingefügt und dynamisch an die geöffnete Seite angepasst. Die Navigationsleiste wird im Moment von drei Dateien angesprochen: Dashboard, Filemanager und Anki Cards. Die Komponente wird niemals alleine aufgerufen.

4.1.4. Realisierung der Ansprüche

4.1.4.1. Responsive Web Design

Die dynamische Anpassung des Webseiten-Layouts wird, wie bereits weiter oben beschrieben, mithilfe des Responsiven Layouts realisiert. Dabei werden Konzepte des Fluiden Layouts umgesetzt, um die Größe des Anzeigefenster dynamisch zu gestalten. So wird diese in Prozentwerten an die Sichtöffnung (englisch „Viewport“) angepasst. Für eine sinnvolle Anzeige der wichtigen Arbeitsbereiche des Lernassistenten im Split View wird wiederum ein Konzept des Adaptiven Layouts verfolgt. Hierbei wird die Breite der Navigationsleiste dynamisch angepasst, indem sie ab einem definierten Umbruch Punkt von der breiteren Version auf eine schmalere Version springt und umgekehrt.⁵

⁴ https://de.wikipedia.org/wiki/Webdesign-Layouttyp#Adaptives_Layout, Stand: 24.01.25

⁵ https://de.wikipedia.org/wiki/Webdesign-Layouttyp#Adaptives_Layout, Stand: 24.01.25

4.1.4.2. FlexBox Prinzip

Um das Konzept des Fluiden Layouts umzusetzen und damit dynamische Layouts zu erstellen, die automatisch reagieren und sich bei Bedarf neu anordnen, ist der CSS Layout-Modus Flexbox hervorragend für unsere Webapplikation geeignet. Dessen Algorithmus verwendet fließende Prinzipien und hat viele hilfreiche Eigenschaften. Dabei geht es in erster Linie um das Arrangieren einer Gruppe von Elementen in einer Reihe oder Spalte und das exakte Steuern ihrer Verteilung mithilfe einer Haupt- und dazu senkrecht verlaufenden Querachse. Da alle Flexbox-Regeln auf die Achsen definiert wurden, kann das Layout problemlos von Reihen- zu Spaltenausrichtung und umgekehrt wechseln.

Darüber hinaus ist eine weitere essentielle Eigenschaft von Flexbox, dass Höhe und Breite keine harten Einschränkungen wie beispielsweise im klassischen Flow-Layout darstellen. Stattdessen kann über die Definition einer hypothetischen Größe angegeben werden, wie viel Platz sich ein Kind-Element nehmen dürfte, sofern das Eltern-Element diesen Platz zur Verfügung stellen kann. Ist dies nicht der Fall, reduziert sich die Größe des Kind-Elementes so weit, dass es in den zur Verfügung stehenden Platz passt.

Auch das Wachsen und Schrumpfen von Elementen innerhalb eines Flex-Kontextes kann gesteuert werden. Und falls diese Prinzipien nicht ausreichen, kann man sich der flex-wrap Eigenschaft bedienen. Dadurch werden Elemente in die nächste Zeile oder Spalte umgebrochen.

Eine letzte wichtige Eigenschaft von Flexbox ist auto margin, womit frei zur Verfügung stehender Platz neben einem Element dem Rand desselben hinzugefügt wird. Dies ermöglicht eine genaue Kontrolle darüber, wie zusätzlicher Platz verteilt wird und zudem besteht damit die Möglichkeit, Elemente zu zentrieren.⁶

4.1.4.3. Dynamische Anpassung der Breite der Navigationsleiste

Die Navigationsleiste passt sich durch unser Responsive Layout an die verschiedenen Fenstergrößen an. Dabei arbeiten wir mit sogenannten Breakpoints, welche in der zu der Navigationsleiste dazugehörigen CSS Datei initialisiert werden. Ab einer bestimmten Größe des Fensters verkleinert sich die Leiste und zeigt nur noch die Icons der Listen Items an. Diese Anpassung war notwendig, um bei allen Fenstergrößen eine übersichtliche Navigationsleiste zu gewährleisten.

⁶ <https://www.joshwcomeau.com/css/interactive-guide-to-flexbox/>, Stand: 24.01.25

4.1.5.Design

4.1.5.1. CSS Stylesheets

Der HTML Style der Webapplikation wird, wie bereits erwähnt, mithilfe von Cascading Style Sheets, kurz CSS umgesetzt.⁷ Hierfür besitzt fast jede HTML-Seite unseres Programms ein eigenes CSS Stylesheet. Einige grundsätzliche Styles werden darüber hinaus im allgemein gehaltenen Styles.css Stylesheet definiert. Dabei gilt für jedes Stylesheet, dass CSS Selektoren wie Klasse und Id in der Reihenfolge auftauchen, in der sie im HTML-Skript erstmals verwendet wurden.

4.1.5.2. Styles.css: Body

Der Style des HTML <body> Elements wird im Styles.css Stylesheet definiert. Hier werden bereits die Konzepte des Fluiden Layouts angewendet, indem Höhe und Breite mithilfe von prozentualen Angaben und in Relation zum Viewport bestimmt werden. Somit passt er sich automatisch an die individuellen Bildschirm Parameter des Endgeräts des Nutzers an und füllt sie vollständig aus. Es bleibt dadurch weder ungenutzter Platz noch wird die Seite scrollbar. Zudem wird mithilfe von “display: flex” ein sogenannter Flex Kontext erstellt, um die Vorzüge des Flexbox-Algorithmus nutzen zu können. Mithilfe von “justify-content: center” und “align-content: center” werden Kind-Elemente entlang der Primär- und Querachse zentriert.⁸

4.1.5.3. Login

Das Programm wird gestartet und als erstes öffnet sich die Login Seite. Diese Seite kümmert sich um die Nutzerdaten und lässt sich wie jedes Standard Login bedienen. Das Design ist in zwei verschiedene Felder aufgeteilt. Zwei Drittel der Content Box sind mit Inputfeldern für die Daten des Nutzers gefüllt. Das andere Drittel beinhaltet ein mehrfarbiges Bild. Momentan hat dieses Bild noch keine Bedeutung. Es dient lediglich als Platzhalter für ein Logo der Applikation.

Über den Feldern befindet sich eine Überschrift, die zur Orientierung dient, da das Design der Login- und Register-Seite sehr ähnlich ist. Die Inputfelder überprüfen eine E-Mail Adresse, den dazugehörigen Benutzernamen und ein Passwort. Das Passwort kann mit Hilfe eines Icons⁹ angezeigt werden. Das gleiche Icon versteckt es beim wiederholten Klicken.

Unter den Input Feldern befinden sich ein Button und ein Link mit jeweiligen Click Event. Der Login-Button ist deaktiviert, solange die Felder noch nicht befüllt worden sind. Durch das Klicken des Buttons werden mehrere Sachen überprüft. Die Inputfelder weisen den eingesetzten Variablen eine ID zu (zum Beispiel ist der eingegebene

⁷ https://www.w3schools.com/html/html_css.asp, Stand: 25.01.25

⁸ <https://www.joshwcomeau.com/css/interactive-guide-to-flexbox/>, Stand: 24.01.25

⁹ https://www.w3schools.com/howto/howto_js_toggle_password.asp, Stand: 03.02.2025

Benutzername mit der ID "username" verbunden) und werden an den API Request gesendet. Wenn alle Daten stimmen, wird der Benutzer erfolgreich zum Hauptprogramm weitergeleitet. Wenn die Daten abweichen, gibt es eine Fehlermeldung.

Unter dem Button befindet sich ein Link, welcher den Benutzer zur Register-Seite bringt. Diese Funktion wurde durch HTML Code implementiert. Wenn man mit Hilfe eines Links auf eine andere Seite verweist, muss dieser nicht im <head> Bereich des Dokuments deklariert sein.

Die Content Box passt sich dank unseres Responsive Designs an die Größe des Fensters an. Der linke Teil nimmt sich bei der kleinsten Größe nur noch den Teil, den es braucht. Das Bild bleibt gleich.

4.1.5.4. Register

Der Benutzer ist noch nicht registriert und möchte ein neues Konto einrichten. Die Register Seite ist im Vergleich zur Login-Seite nur in wenigen Aspekten verändert worden. Diese Seite kümmert sich um die Nutzerdaten und lässt sich wie jedes Standard Register bedienen. Das Design ist in zwei verschiedene Felder aufgeteilt. Ein Drittel beinhaltet ein mehrfarbiges Bild. Momentan hat dieses Bild noch keine Bedeutung. Zwei Drittel der Content Box sind mit Inputfeldern für die Daten des Nutzers gefüllt.

Die Register Seite ermöglicht dem Benutzer, vier verschiedene Werte festzulegen. Das Programm erfragt eine E-Mail Adresse, einen Benutzernamen und ein Passwort, welches zwei Mal eingesetzt werden muss. Wieder haben wir für beide vertraulichen Inputfelder ein Icon hinzugefügt, welches das Passwort sichtbar macht und wieder versteckt.

Unter den Inputfeldern befindet sich ein deaktivierter Button, der erst nach dem Füllen der Felder aktiviert wird. Das Frontend überprüft vor dem Übermitteln der Daten, ob das eingegebene Passwort mit dem wiederholten Passwort übereinstimmt. Das Backend geht durch alle bisher erstellten Konten und schaut, ob die eingegebene E-Mail Adresse bereits einen Account hinterlegt hat. Nach der Überprüfung werden die Daten mithilfe einer "POST" Funktion an das Backend gesendet und in der Datenbank hinterlegt. Der Benutzer wird daraufhin, wie beim Login-Button, zum Hauptprogramm weitergeleitet.

Unter dem Register-Button befindet sich wieder ein Link. Dieser ermöglicht es dem Benutzer, zurück zur Login-Seite zu gelangen, falls bereits doch ein Account besteht.

Die Content Box passt sich dank unseres Responsive Designs an die Größe des Fensters an. Das Bild bleibt gleich. Der rechte Teil nimmt sich bei der kleinsten Größe nur noch den Teil, den es braucht.

4.1.5.5. Navigationsleiste

Die Navigationsleiste ist ein wichtiger Bestandteil unseres Programms und wird bereits in zwei Abschnitten beschrieben (4.1.2.3 Dynamisches Laden der Navigationsleiste, 4.1.3.3 Dynamische Anpassung der Breite der Navigationsleiste). Dieses HTML Dokument wird dynamisch an die aufgerufene Seite hinzugefügt. Wie bereits erwähnt, wird die Seite nie alleine aufgerufen, sie wird nur beim Aufrufen einer der Hauptseiten sichtbar.

Die Leiste ist in Reihen aufgeteilt. Ganz oben befindet sich der Programmname. Darunter werden `` (Listen Items) angezeigt. Jedes Item ist klickbar und leitet den Benutzer auf eine andere Seite des Programms weiter (zum Beispiel befindet man sich auf der Dashboard Seite und wechselt zum FileManager nach dem Klicken des Listen Items). Zur Verdeutlichung der einzelnen Seitennamen befinden sich links von den Titeln Icons. Das Dashboard wird mit einem Haus dargestellt. Dieser Teil kann später nach Belieben erweitert werden, wenn neue Seiten zu unserem Programm implementiert werden.

Unter den drei Hauptseiten befindet sich noch ein letztes Item mit dem Titel "Log out". Dieser Button leitet den Benutzer wieder zurück zur Login-Seite des Programms und meldet ihn ab. Der Abstand zu den anderen Listen Items ist mit Hilfe der "last-child" Klasse möglich. Das Programm erkennt, welches Item das letzte ist und wird mit einem "margin-top" versehen, welcher den Abstand zu den vorherigen Items ermöglicht. Beim Verkleinern des Fensters zeigt die Navigationsleiste nur noch die Icons der Verknüpfungen, um Platz zu sparen.

4.1.5.6. Styles.css: Mainbox und Contentbox

Neben dem HTML `<body>` Element werden im Styles.css Stylesheet zwei wichtige Klassen definiert, die von den Hauptseiten des Lernassistenten für ein gleichbleibendes Layout als Grundgerüst genutzt werden. Dabei handelt es sich um die Klassen "main-" und "content-box".

Dabei bildet die Mainbox eine Art optischen Rahmen um die wechselnden Hauptinhalte der Applikation und sorgt somit für ein kontinuierliches Erscheinungsbild des Lernassistenten. Gleichzeitig wird hier, wie bereits beschrieben, die Navigationsleiste durch dynamisches Laden am linken Rand eingefügt. Die Contentbox stellt wiederum den Bereich dar, in den die wechselnden Hauptinhalte der Applikation eingefügt werden.

Beide Klassen nutzen den Flex Kontext und es werden stilistische Einstellungen wie abgerundete Ecken, Box-Schatten und die Schriftart Quicksand festgelegt.

4.1.5.7. Dashboard

Nachdem der Benutzer sich angemeldet hat oder registriert hat, wird das Hauptprogramm geöffnet. Die Dashboard-Seite dient als zentraler Hub mit verschiedenen Funktionen. Das Dokument lädt zuerst die dynamische Navigationsleiste, bevor es die Contentbox lädt. Beide Spalten passen sich dank unseres Designs an die Größe des Fensters an. Um bei beiden eine bestimmte Größe für die Leserlichkeit des Programms zu gewährleisten, haben wir den Spalten jeweils eine flex-basis zugewiesen. Mit dieser Variable ist es uns möglich, eine Mindestgröße bei kleinen Fenstern zu erreichen. Außerdem kann man dadurch festhalten, welche Spalte mehr Platz bei großen Fenstern einnehmen soll.

Das Dashboard ist in zwei Abschnitte unterteilt. Auf der linken Seite befindet sich eine Willkommensnachricht und eine Beschreibung der einzelnen Seiten, welche durch die Navigationsleiste erreichbar werden. Dieser Teil kann erweitert werden, wenn eine neue Seite zu unserem Programm hinzugefügt wird.

Auf der rechten Seite befindet sich ein "Drag und Drop" Feld. Dieses Feld ermöglicht dem Benutzer, die ersten Funktionen unseres Lehrassistenten auszuprobieren. Ein Vorlesungsfolie kann hochgeladen werden, indem das Dokument in das Feld gezogen wird.

Unter dem Feld befindet sich noch ein Button mit einem Klick-Event, welches den Explorer öffnet und eine manuelle Suche nach den gewünschten Dokumenten ermöglicht. Wenn das PDF-Dokument akzeptiert wird, bekommen wir einen positiven Alert. Es öffnet sich ein dazugehöriges Modal (mehr dazu im nächsten Punkt 4.1.4.7.1).

4.1.5.7.1. Dashboard - Modal

Beim Upload eines PDF-Dokuments erscheint dem Benutzer ein Modal, welches die Dashboard-Seite solange deaktiviert, bis er mit diesem interagiert hat. Er wird aufgefordert, Modulname und Thema anzugeben, um das hochgeladene PDF im FileManager ablegen zu können.

Das Erscheinungsbild des Modals wird durch die Klasse „modal“ im DashboardStyles.css Stylesheet definiert. Per default ist es durch „display: none“ ausgeblendet. Der Upload eines PDF-Dokuments stellt einen mit JavaScript implementierten Trigger für die Aktivierung des Modals dar, dadurch wird dessen Style auf „display: block“ gesetzt. Des Weiteren wird mithilfe des „z-index: 1“ die Position des Modals in Z-Achse definiert. Es wird so über dem restlichen Inhalt des Dashboards angezeigt und dieser wird wiederum durch eine opake Hintergrundfarbe leicht ausgegraut.

Diese Eigenschaften sollen den Fokus des Benutzers auf den Inhalt des Modals lenken, der durch die Klasse „modal-content“ definiert wird. Dieser Bereich wird durch „margin: 15% auto“ 15 Prozent von oben und horizontal zentriert positioniert. Höhe und Breite des Modals sind harte Einschränkungen. Neben einem Titel und zwei Buttons beinhaltet das Modal noch zwei Inputfelder, die eine Dropdown-Liste mit Vorschlägen einblenden können. Der Benutzer kann so entweder bereits existierende Module und Themen auswählen, in denen das PDF-Dokument abgelegt werden soll, oder durch Eingabe neue erzeugen.

Da das Erscheinungsbild der bestehenden Implementation der Kombination von Input-Element und Dropdown-Liste von jedem Browser unterschiedlich gestaltet wird, wurde die Funktionalität mithilfe von JavaScript eigens implementiert und mit CSS der Gestaltung des Lernassistenten angepasst. Somit bleibt die Kontinuität des Designs erhalten und Irritationen durch ein ungewöhntes Erscheinungsbild der Benutzeroberfläche beim Wechsel des Browsers werden vermieden.

4.1.5.8. FileManager

Der FileManager ist eine weitere Seite des Lernassistenten. Er übernimmt die Organisation von hochgeladenen Dokumenten sowie erzeugten Zusammenfassungen und AnkiCards und wird in der HTML-Datei FileManager.html implementiert.

Der Inhalt der Contentbox enthält neben der Überschrift „FileManager“ drei Bereiche: einen größeren Bereich, der ungefähr die linke Hälfte der Contentbox ausfüllt, und zwei kleinere, die jeweils untereinander angeordnet die rechte Hälfte ausfüllen. Sie tragen die englischsprachigen Überschriften für Modul, Ordner und Dateien.

Die Bereiche für Module und Dateien enthalten ungeordnete HTML Listen, während der Bereich für die Ordner durch eine andere Struktur optisch hervorgehoben wird. Hier werden mithilfe von Ordner-Icons die Ablagebereiche für Dateien gekennzeichnet.

Die Styles werden im CSS Stylesheet FileManager.css definiert. So bekommen die Listen-Elemente, sowie die Bereiche mit den Ordner-Icons nicht nur ihr generelles Erscheinungsbild, bestehend aus ihrer Größe, Farbe und abgerundete Ecken, sondern es wird auch mithilfe der „:hover“ und „:active“ Pseudo-Klassen ein optisches Feedback für den Benutzer definiert, wenn dieser mit der Maus über ihnen schwebt beziehungsweise wenn sie angeklickt werden.¹⁰¹¹

Da auch etwas im Programm passieren soll, wenn diese Elemente angeklickt werden, ist ihre Antwort auf ein Klick-Event im JavaScript-Teil definiert. Hierfür wird ihnen in einer forEach-Schleife jeweils ein EventListener für das Klick-Event zugewiesen. So werden teilweise Inhalte des FileManagers ein- und andere ausgeblendet.

¹⁰ https://www.w3schools.com/cssref/selector_hover.php, Stand: 25.01.25

¹¹ https://www.w3schools.com/cssref/selector_active.php, Stand: 25.01.25

Im Detail geschieht Folgendes: Wenn die Dashboard-Seite aufgerufen wird, werden zunächst nur die Module und die Überschrift des Ordner-Bereichs eingeblendet. Weitere Bereiche sind mit „display: none“ ausgeblendet. Wählt der Benutzer nun ein bestimmtes Element aus der Liste der Module aus, werden die darin enthaltenen Ordner-Elemente durch „display: flex“ eingeblendet und gleichzeitig erscheint die Überschrift des Bereichs für die Dateien. Das Einblenden der besagten Elemente für das angeklickte Modul wird durch die Funktion „makeFoldersVisible“ im JavaScript-Teil der FileManager.html Datei implementiert. Ein ähnliches Verhalten wird auch beim Anklicken eines Ordner-Elements über die Funktion „makeFilesVisible()“ erreicht. Hierdurch werden die im Ordner enthaltenen Dateien im Dateien-Bereich sichtbar gemacht.

Für eine flüssige Navigation durch den FileManager wird das Ausblenden bestimmter Bereiche über die Methoden „makeFoldersInvisible()“ und „makeFilesInvisible()“ ebenfalls benötigt. Um die Verwaltung des FileManagers zu ermöglichen, ist per Rechtsklick das Löschen von einzelnen Modulen, Ordnern oder Dateien in JavaScript umgesetzt worden. Hierfür werden die jeweiligen „delete“ Funktionen ausgeführt, die zudem die „makeInvisible“ Funktionen ausführen, um nach dem Löschvorgang die erforderlichen Bereiche auszublenden.

Um die Verwaltung des FileManagers zu vervollständigen, enthalten die drei Bereiche jeweils ein ellipsenförmiges Icon, welches das Hinzufügen von Modulen, Ordnern und Dateien ermöglicht. Durch Anklicken öffnet sich ein kleines Dialogfeld, in dem der Benutzer den Namen für das hinzuzufügende Element eingeben kann. Im JavaScript-Teil wird das hinzuzufügende Element der jeweiligen Liste hinzugefügt, anklick- und löschbar gemacht.

Die verwendeten Icons stammen von Font Awesome, ihre Größe und Farbe wird in der FileManagerStyles.css Datei festgelegt.

4.1.5.8.1. FileManager - Modal

Wenn der Benutzer im FileManager auf eine Zusammenfassung klickt, öffnet sich ein Modal zur Ansicht und deaktiviert die FileManager-Seite solange, bis es durch den Benutzer geschlossen wird. Per Default wird die Zusammenfassung in der schreibgeschützten Ansicht geöffnet. Hier kann sich der Benutzer in Ruhe die KI-generierte Zusammenfassung durchlesen. Klickt er auf den „Edit“-Button rechts oben im Modal, so kann er in den Bearbeitungsmodus wechseln. Dadurch ermöglicht ihm das Programm, die Zusammenfassung für ein noch individuelleres Ergebnis umzuformulieren und mithilfe des „Safe“-Buttons abzuspeichern. Nach dem Abspeichern wechselt das Modal automatisch in die schreibgeschützte Ansicht zurück. Das Schließen des Modals ist durch Klick auf den „Quit“-Button in der oberen linken Ecke des Modals möglich.

Das Erscheinungsbild des Modals wird durch die Klasse „modal“ im FileManagerStyles.css Stylesheet definiert. Per default ist es durch „display: none“ ausgeblendet. Das Anklicken einer Zusammenfassung stellt einen mit JavaScript implementierten Trigger für die Aktivierung des Modals dar, und dadurch wird dessen Style auf „display: block“ gesetzt. Des Weiteren wird mithilfe des „z-index: 1“ die Position des Modals in Z-Achse definiert. Es wird so über dem

restlichen Inhalt des FileManagers angezeigt und dieser wird wiederum durch eine opake Hintergrundfarbe leicht ausgegraut.

Diese Eigenschaften sollen den Fokus des Benutzers auf den Inhalt des Modals lenken, dessen Erscheinungsbild durch die Klasse „modal-content-box“ definiert wird. So wird die Modal Contentbox durch „margin: auto“ horizontal zentriert und durch „margin-top“ und „margin-bottom“ in der Vertikalen jeweils 20px bis zum Rand positioniert. Die Höhe ist mit 90% des Viewports und die Breite flexibel je nach vorhandenem Platz definiert, maximal jedoch mit 816px. So hat die Modal Contentbox nahezu die Breite eines DIN A4 Blattes, was an einen Lernzettel erinnern soll.

Neben einem Titel und zwei Buttons, die auch beim nach unten Scrollen der Seite am oberen Rand haften bleiben dank der „sticky“-Klasse, beinhaltet das Modalfenster noch den Bereich, in dem der Inhalt der Zusammenfassung geladen werden kann. In der schreibgeschützten Ansicht handelt es sich dabei um einen Paragraphen, der mithilfe der Klasse „modal-read-only“ gestaltet wird. Im Bearbeitungsmodus wird der Paragraph ausgeblendet und stattdessen ein textarea-Element eingeblendet, welches ein mehrzeiliges Eingabefeld ist.¹² [1] Dieses wird über die entsprechende Klasse „modal-textarea“ gestaltet.

In der schreibgeschützten Ansicht passt sich die Höhe der Modal Contentbox flexibel der Menge des Inhalts an, sodass der Benutzer eine unterbrechungsfreie Ansicht der Zusammenfassung erhält. Im Bearbeitungsmodus passt sich dagegen die Höhe der Modal Contentbox nicht an. Hier lässt sich stattdessen mithilfe von „overflow: auto“ im textarea-Element scrollen, wenn die Menge des Inhalts eine komplette Ansicht der Zusammenfassung auf dem Bildschirm nicht möglich macht.

Durch Klicken auf den „Save“-Button gerät der Benutzer wieder in die schreibgeschützte Ansicht mit aktualisiertem Inhalt zurück, wobei das textarea-Element wieder ausgeblendet wird. In beiden Modi kann das Modal durch Klicken des „Quit“-Buttons geschlossen werden.

Das Laden des Titels und Inhalts einer Zusammenfassung aus dem Backend ist zur Zeit noch nicht möglich. Um das dynamische Laden simulieren zu können, wird ein JSON Datensatz, wie er vom Backend erwartet wird, im JavaScript-Teil zur Verfügung gestellt.

4.1.5.9. Anki Cards

Die Anki Cards Seite ist in mehrere Schichten unterteilt. Dieser Abschnitt bezieht sich auf das ursprüngliche Design der Seite, bevor durch die Anbindung des Backends Container verändert wurden. Dieses Dokument lädt wie das Dashboard und der FileManager die Navigationsleiste dynamisch zu der Content Box.

Das Dokument ist in zwei Ansichten unterteilt: die Auswahlseite und die Lernsession. Auf der Auswahlseite kann der Nutzer zwischen den Decks in den hinterlegten Modulen

¹² https://www.w3schools.com/tags/tag_textarea.asp, Stand: 01.02.25

entscheiden. Module sind hier wie Container aufgebaut, die je nach Anzahl in die Seite geladen werden. Wenn die Container in der Weite keinen Platz mehr haben, dargestellt zu werden, fangen sie an, in der Reihe darunter weiter abgebildet zu werden. Die CSS Variable flex-wrap ermöglicht diese Darstellung.

Die Modul Container beinhalten Listenelemente, welche jeweils mit einem Click Event ausgestattet sind. Diese Elemente stellen die Anki Card Decks dar. Ziel war es, durch Klicken des Decks von der Auswahlseite zur Lernsession zu gelangen. Beim Klicken werden die CSS Klassen entweder sichtbar gemacht ("display: none" wird zu "display: flex") oder versteckt. Die Höhe der Container ist in der CSS Datei hinterlegt. Um alle Listenelemente zu sehen, kann man durch den Container scrollen ("overflow: hidden" sorgt für einen scrollbaren Container ohne die Seite im Allgemeinen zu verändern)

Die Lernsession besteht aus drei Komponenten. Ein Button oben links ermöglicht es dem Benutzer, die Lernsession zu beenden und zurück zur Auswahlseite zu gelangen. Der Hauptteil der Lernsession stellt die Anki Card dar. Diese zeigt eine Frage des Decks an, ohne die Antwort preiszugeben. Man beantwortet die Frage für sich selbst und klickt daraufhin auf die Karte. Eine Funktion im Frontend zeigt die zuvor versteckte Antwort innerhalb der Karte. Ein wiederholtes Klicken versteckt die Antwort erneut.

Durch das Klicken der Karte werden auch vier Buttons sichtbar gemacht. Diese dienen als eigene Entscheidungsmöglichkeit für den Schwierigkeitsgrad der Frage und sollen Daten zurück ans Backend senden. Nach der Auswahl des Benutzers soll die nächste Frage geladen werden. Antwort und Schwierigkeitsgrad Button werden wieder versteckt.

In diesem Abschnitt gehe ich noch kurz auf das Design der Seite nach Anbindung der Endpoints ein. Da die Anki Card Schnittstellen Teil eines Open Source Programms sind, weichen sie von unseren Absprachen ab und mussten demnach anders im Design dargestellt werden.

Der Benutzer hat die Möglichkeit, neue Decks anzulegen. Dabei öffnet sich ein Modal, das fragt, zu welchem Modul und zu welchem Topic dieses neu erstellte Deck gehören soll. Dieses Modal und alle anderen auf der Anki Cards Seite befinden sich in einem separaten HTML Dokument. Erst nach dem Erstellen eines Decks wird ein Container im Auswahlbereich eingebaut.

Statt Modul Containern beinhalten die Container nur ein Deck. Die Listenelemente stellen die einzelnen Fragen dar. Am Boden des Containers befinden sich zwei Buttons, diese öffnen ein weiteres Modal. Fragen und Antworten der Decks können erstellt

werden, dabei wählt man ein Dokument und lässt die Künstliche Intelligenz diesbezüglich Fragen erstellen. Eine manuelle Methode Fragen zu erstellen, gibt es auch. Innerhalb des Containers befindet sich noch ein weiterer Button, welcher mit dem ausgewählten Deck in die Lernsession wechselt.

4.1.6. Nächste Schritte

4.1.6.1. Wiederherstellung eines kontinuierlichen Designs bei FileManager und AnkiCards

Um die Verknüpfung mit dem Backend zu realisieren und schnellstmöglich zu testen, wurde das ursprüngliche Design des Frontends zum Teil abgewandelt. Die Wiederherstellung eines kontinuierlichen Designs ist der nächste wichtige Schritt bei der Entwicklung des Frontends. Das Hauptaugenmerk liegt hierbei auf dem FileManager und den AnkiCards.

4.1.6.2. Erweiterung des Frontends

Des weiteren soll das Frontend um weitere Seiten ergänzt werden, die der Realisierung der ausstehenden Requirements dienen. Beispielsweise ist im Projekt bereits ein Grundgerüst und ein darauf verweisender Link in der Navigationsleiste für das Profil des Benutzers angelegt. Der Link ist zur Zeit noch auskommentiert und deswegen inaktiv und nicht sichtbar. In der Navigationsleiste können nach und nach weitere Links zu neuen Seiten des Lernassistenten hinzugefügt werden und somit das Frontend iterativ weiterentwickelt werden.

4.1.6.3. Erweiterung des Dashboards

Neben den Erklärungen der einzelnen Seiten und dem "Drag und Drop" Feld zum Hochladen von Dokumenten soll das Dashboard in Zukunft auch Benutzerdaten anzeigen. Aus Zeitgründen war es zum jetzigen Zeitpunkt noch nicht möglich, den Aspekt Gamification in unser Projekt einzubinden. Dementsprechend soll das Dashboard Statistiken der Nutzung des Programms und Shortcuts zu den zuletzt verwendeten Dateien und Anki Cards anzeigen können.

4.1.6.4. Erweiterung durch verschiedene Sprachen und Farbvarianten

Unser Projekt hat zum Zeitpunkt der Abgabe keine deutsche sondern nur eine englische Version. Um das Projekt für alle zugänglich zu machen, wäre eine Übersetzung des Programms denkbar, da es heutzutage der Standard ist.

Zudem wäre es wichtig, weitere Farbvarianten zur Verfügung zu stellen, um jedem Benutzer ein individuelles Erlebnis zu gewährleisten. Verschiedene Farben sind bereits in den CSS Dateien hinterlegt und können nach Belieben auskommentiert und eingefügt werden.

4.1.7. Backendverknüpfung

Die Backendverknüpfung erfolgt mithilfe der fetch()-Methode in JavaScript.

4.1.8. Problemanalyse

4.1.8.1. Aufgetretenes Problem

Verzögerung des Entwickelns mit Backend Verbindung

4.1.8.2. Analyse des Problems

Aufgrund der erst später im Projektverlauf erfolgten Bereitstellung der Endpoints im Backend und Konfigurationen am Server, die Anfragen der Webapplikation bisher abgeblockt haben, musste das Entwickeln des Frontends den Großteil der Projektzeit/über die gesamte Projektzeit ohne Verbindung mit dem Backend erfolgen.

4.1.8.3. Lösung des Problems

Damit die Entwicklung des Frontends nicht stagnierte, wurden die folgenden Maßnahmen ergriffen:

1. Maßnahmen im Team:

Der jeweils aktuelle Stand von Design und Funktion des Frontends wurde über den gesamten Projektverlauf in regelmäßigen Meetings dem Team vorgestellt und besprochen. So konnten wir in engmaschigen Intervallen Feedback einholen und die Weiterentwicklung der Webapplikation planen. Die Planung der grafischen Benutzeroberfläche geschah unter anderem mithilfe von gemeinsam angefertigten Skizzen. Über erforderliche Endpoints, die im Backend implementiert werden mussten, wurde zusammen gesprochen. Sie wurden anschließend in unserem gemeinsamen Kanban-Board gesammelt, sodass jedes Teammitglied jederzeit Zugriff darauf hatte.

2. Maßnahmen im Frontend:

Die Entwicklung des Frontends geschah statisch, dabei wurde die grafische Benutzeroberfläche entsprechend der gezeichneten Designs gestaltet. Daten, die eigentlich aus dem Backend geladen werden sollten, wurden simuliert. Dies geschah größtenteils mit rein fiktiven Variablen, die ähnlich denen aus dem Backend zu erwartenden sein sollten. In einem Fall konnte mithilfe eines Mock-Objekts gearbeitet werden, da uns zu diesem Zeitpunkt die Struktur der Daten, wie sie aus dem Backend zur Verfügung gestellt werden sollten, bekannt war. Dank dieser Maßnahmen konnte das Aussehen des Lernassistenten schon weitgehend inklusive dynamischer Effekte gestaltet werden. Sobald die Endpoints im Backend fertig gestellt waren, konnten diese im Frontend im JavaScript-Teil integriert werden. Allerdings ohne sie testen zu können, da der Server nach wie vor Anfragen der Webapplikation blockiert hat.

4.2. Backend

4.2.1. Überblick

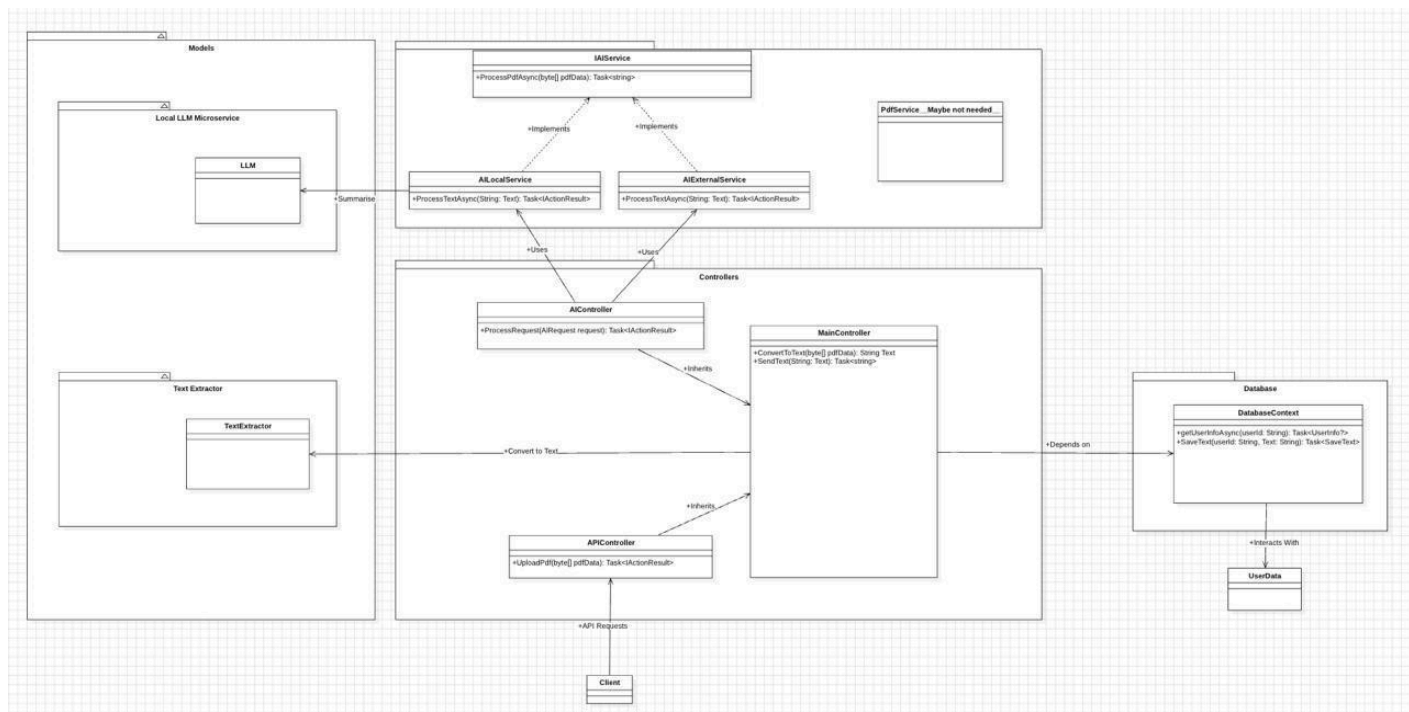
Zu Beginn des Projekts haben wir eine gründliche Organisation durchgeführt. Mithilfe eines Trello-Boards wurden Aufgaben verwaltet und der Fortschritt dokumentiert. Die Architektur

basiert auf einem Microservices-Ansatz, der mit Diagrammen festgehalten wurde. Ursprünglich begannen wir die Implementierung in C# (.NET), entschieden uns jedoch aufgrund der Komplexität für einen Wechsel zu Python (Flask). Anschließend richteten wir einen Server ein, öffneten die erforderlichen Ports, um die Kommunikation zwischen den Microservices und ihren APIs zu ermöglichen, und etablierten eine CI/CD-Pipeline. Sobald Änderungen in den „main“-Branch gepusht werden, erfolgt eine automatische Aktualisierung des Servers. Während das Backend-Team die API entwickelte, arbeitete das Frontend-Team an der Benutzeroberfläche.

Das Backend wurde mit Flask, SQLAlchemy und Docker realisiert und stellt eine RESTful-API bereit. Diese ermöglicht die Verwaltung von „modules“, „topics“, „files“ und „Anki-cards“. Zudem verarbeitet sie PDF-Dateien und generiert mithilfe von KI-Modellen Zusammenfassungen.

4.2.2. Architektur

Die folgende Abbildung zeigt die anfängliche Backend-Architektur dieses Projekts, einschließlich der Microservices, der Datenbank und externer Komponenten. Aufgrund der zunehmenden Komplexität und starken Vernetzung des Projekts haben wir jedoch aufgehört, UML-Diagramme zu verwenden.



Die Backend-Architektur ist modular, skalierbar und wartungsfreundlich gestaltet. Sie nutzt mehrere Schlüsseltechnologien und folgt einem Microservices-Ansatz, bei dem jeder Dienst in einem eigenen Docker-Container läuft.

- **Flask Application:** Der Kern des Backends, der HTTP-Anfragen und -Antworten verarbeitet.

- Database: PostgreSQL-Datenbank zur Speicherung von Modulen, Themen, Dateien und Zusammenfassungen.
- AI Services: Externe und lokale KI-Dienste zur Textverarbeitung und Generierung von Zusammenfassungen.
- Nginx: Ein Reverse-Proxy-Server, der Anfragen an die entsprechenden Dienste weiterleitet.
- Postman: Ein leistungsstarkes API-Entwicklungstool, das für das Testen, Dokumentieren und Automatisieren von API-Anfragen verwendet wird. In diesem Projekt wird Postman genutzt, um HTTP-Anfragen an die Flask-API zu senden, Endpunkte zu testen und deren Funktionalität sicherzustellen. Mit Postman können verschiedene Anfragetypen (GET, POST, PUT, DELETE) simuliert, Authentifizierungsmechanismen überprüft und API-Tests automatisiert werden. Zudem ermöglicht Postman die Erstellung und Verwaltung von API-Sammlungen, wodurch wiederholbare Testfälle effizient organisiert werden können.

4.2.3. Project Structure

The project is structured into several directories and files, each serving a specific purpose:

- src/main/App: Enthält die Haupt-Flask-Anwendung.
- src/main/Database: Enthält den Datenbankkontext.
- src/main/Services: Enthält verschiedene Services für Textextraktion, KI-Verarbeitung und Usermanagement.
- src/main/Controllers: Enthält Controller zur Verarbeitung von HTTP-Anfragen.
- docker-compose.yml: Definiert die Services und deren Konfigurationen für Docker.
- Dockerfile.app: Dockerfile für die Flask-Anwendung.
- Dockerfile.nginx: Dockerfile für den Nginx reverse proxy.
- nginx.conf: Konfigurationsdatei file for Nginx.
- .env: Datei für die Environment variables.

4.2.4. Verwendete Technologien

4.2.4.1. Python

Die Programmiersprache, die für die Backend-Entwicklung verwendet wird.

4.2.4.2. Flask

Flask ist ein leichtgewichtiges und flexibles Mikro-Web-Framework für Python. Es wurde entwickelt, um einfach und dennoch leistungsstark zu sein, und bietet Entwicklern die Werkzeuge, die sie für die Erstellung von Webanwendungen und APIs mit minimalem Aufwand benötigen.

4.2.4.3. SQLAlchemy

Eine ORM-Bibliothek (Object-Relational Mapping) für Python, die für die Interaktion mit der PostgreSQL-Datenbank verwendet wird. Sie bietet eine hochgradige Abstraktion für Datenbankoperationen.

4.2.4.4. PostgreSQL

Ein leistungsstarkes, quelloffenes relationales Datenbanksystem.

4.2.4.5. Nginx

A web server used as a reverse proxy for the services.

4.2.4.6. Psycopg2

A PostgreSQL adapter for Python used by SQLAlchemy to connect to the database.

4.2.4.7. Docker

Docker ist eine Plattform, die es Entwicklern ermöglicht, Anwendungen mithilfe von Containern zu erstellen, bereitzustellen und auszuführen. Container sind leichtgewichtige, portable und isolierte Umgebungen, die alles enthalten, was eine Anwendung zum Laufen benötigt, einschließlich des Codes, der Laufzeit, der Systemtools, der Bibliotheken und der Einstellungen.

4.2.4.8. Hauptmerkmale von Docker:

4.2.4.8.1. Containerisierung:

Anwendungen und deren Abhängigkeiten werden in Containern verpackt, was sicherstellt, dass sie in jeder Umgebung gleich funktionieren, sei es auf dem lokalen Rechner, einem Server oder in der Cloud.

4.2.4.8.2. Portabilität:

Docker-Container können auf jeder Plattform ausgeführt werden, die Docker unterstützt, wie z. B. Windows, macOS, Linux und Cloud-Dienste.

4.2.4.8.3. Isolierung:

Jeder Container läuft in einer isolierten Umgebung, was Konflikte zwischen Anwendungen und deren Abhängigkeiten verhindert.

4.2.4.8.4. Effizienz:

Im Gegensatz zu virtuellen Maschinen (VMs) teilen sich Container den Host-Kernel, wodurch sie weniger Ressourcen verbrauchen und schneller starten.

4.2.4.8.5. Skalierbarkeit:

Docker erleichtert das Hochskalieren von Anwendungen, indem mehrere Container für dieselbe Anwendung gestartet werden können.

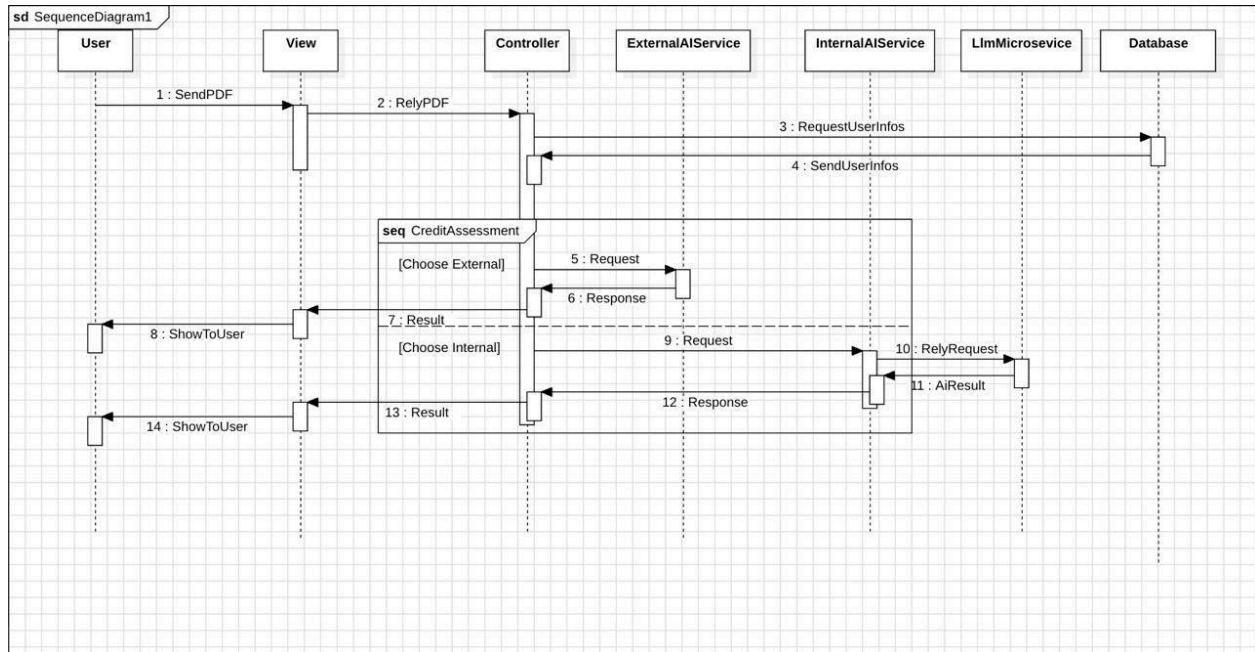
4.2.4.8.6. Wiederholbare Entwicklungs- und Produktionsumgebungen:

Mit Dockerfiles können Entwickler die Infrastruktur als Code definieren und so sicherstellen, dass alle Umgebungen konsistent sind.

4.2.5. Rollen der verschiedenen Klassen

4.2.5.1. Controllers.

Das nachfolgende Diagramm zeigt die Interaktion der Controller mit den zugehörigen Services und beschreibt deren Verantwortlichkeiten innerhalb des Systems.



4.2.5.1.1. TextController

Verwaltet die Textextraktion und -verarbeitung. Er nutzt TextExtractor, um PDF-Daten in Text umzuwandeln, und AIProcessor, um fortgeschrittene Textverarbeitung durchzuführen. Die Methode `convert_to_text` extrahiert den Textinhalt aus einer gegebenen PDF-Datei. Die Methode `process_text` führt eine KI-gestützte Textanalyse durch und kann dabei optional einen lokalen oder externen Verarbeitungsmodell nutzen.

4.2.5.1.2. AIController

Bietet Funktionen zur Textverarbeitung mithilfe eines externen KI-Dienstes oder eines lokalen KI-Modells. Die Methode `process_text` entscheidet basierend auf den Benutzereinstellungen, ob der lokale oder externe Dienst verwendet wird. Sie verarbeitet den Text asynchron und gibt die von dem ausgewählten KI-Modell generierte Ausgabe zurück.

4.2.5.1.3. UserController

Verwaltet benutzerbezogene Vorgänge wie Registrierung, Anmeldung und Abruf von Benutzerdaten. Er interagiert mit dem UserService, um Benutzerdaten und Authentifizierung zu verwalten. Zum Beispiel überprüft die Methode `register`, ob ein Konto bereits existiert, und erstellt andernfalls einen neuen Benutzer. Die Methode `login` überprüft die Benutzeranmeldeinformationen und gewährt bei erfolgreicher Authentifizierung Zugriff.

4.2.5.1.4. FileManagerController

Verwaltet "files" Speicher- und Abrufvorgänge in einem datenbankgestützten System. Er ermöglicht es Benutzern, „Modules“ und „Topics“ zu erstellen und zu verwalten, PDF-Dateien hochzuladen und abzurufen sowie gespeicherte Dateien bei Bedarf zu löschen.

4.2.5.2. Services

4.2.5.2.1. TextExtractor

Verantwortlich für die Extraktion von Text aus PDF-Dokumenten. Es sendet die PDF-Daten an einen externen Mikroservice zur Textextraktion. Der extrahierte Text wird als String zurückgegeben oder eine Fehlermeldung, falls die Extraktion fehlschlägt.

4.2.5.2.2. AITextProcessor

Ermöglicht die KI-gestützte Textverarbeitung. Es arbeitet mit dem AIController zusammen, um extrahierten Text asynchron zu verarbeiten. Die Klasse formatiert den Eingabetext, indem sie Benutzeranweisungen voranstellt, bevor sie ihn zur Verarbeitung sendet.

4.2.5.2.3. AIEternalService

Kommuniziert mit dem externen KI-Dienst GroqCloud, um Texteingaben zu verarbeiten. Es sendet eine API-Anfrage mit dem bereitgestellten Text und den Authentifizierungsdaten und gibt die verarbeitete Antwort zurück.

4.2.5.2.4. AILocalService

Interagiert mit einem lokalen KI-Modell zur Textverarbeitung. Es sendet eine Anfrage an eine lokale LLM API und gibt die generierte Ausgabe zurück.

4.2.5.2.5. UserService

Verwaltet die Benutzerverwaltung und Authentifizierung. Es interagiert mit der Datenbank, um zu prüfen, ob ein Konto bereits existiert, neue Benutzerkonten zu erstellen und Benutzer anhand ihrer Zugangsdaten zu authentifizieren. Passwörter werden vor der Speicherung sicher mit `generate_password_hash` gehasht, und die Authentifizierung erfolgt mit `check_password_hash`, um Anmeldeversuche zu verifizieren.

4.2.5.2.6. DatabaseContext

erwartet die Datenbankverbindung und stellt Methoden zur Interaktion mit der Datenbank bereit. Es umfasst Funktionen zum Erstellen, Abfragen und Bearbeiten von Datenbanktabellen, darunter für Users, Modules, Topics, Files und Zusammenfassungen.

4.2.5.2.7. SummaryProcessor

Die Zusammenfassungsfunktionalität ermöglicht es Benutzern, Texte aus hochgeladenen PDF-Dokumenten zu extrahieren und mithilfe von KI-gestützten Diensten zusammenzufassen. Diese Funktion ist in mehreren Modulen implementiert, die zusammenarbeiten, um eine nahtlose Benutzererfahrung zu gewährleisten. Die wichtigsten Module sind:

- SummaryProcessor.py
- app.py
- DatabaseContext.py

4.2.5.2.7.1. SummaryProcessor.py

Die SummaryProcessor-Klasse verwaltet die Interaktion mit KI-Diensten und die Generierung von Zusammenfassungen.

Kernfunktionen:

- Initialisierung: Konfiguriert externe und lokale KI-Dienste.
- get_summary-Methode: Erstellt eine Anweisung zur Zusammenfassung und wählt den entsprechenden KI-Dienst.
- Asynchrone Verarbeitung: Ruft den KI-Dienst auf und gibt die generierte Zusammenfassung zurück.

4.2.5.2.7.2. app.py

Dieses Modul definiert die Flask-API, die die Zusammenfassungsfunktionalität bereitstellt.

Kernfunktionen:

- /get_summary-Endpunkt: Verarbeitet POST-Anfragen zur Zusammenfassung von PDF-Texten.
- Datenbankprüfung: Prüft, ob eine Zusammenfassung bereits existiert.
- KI-Integration: Sendet extrahierten Text an SummaryProcessor zur Verarbeitung.

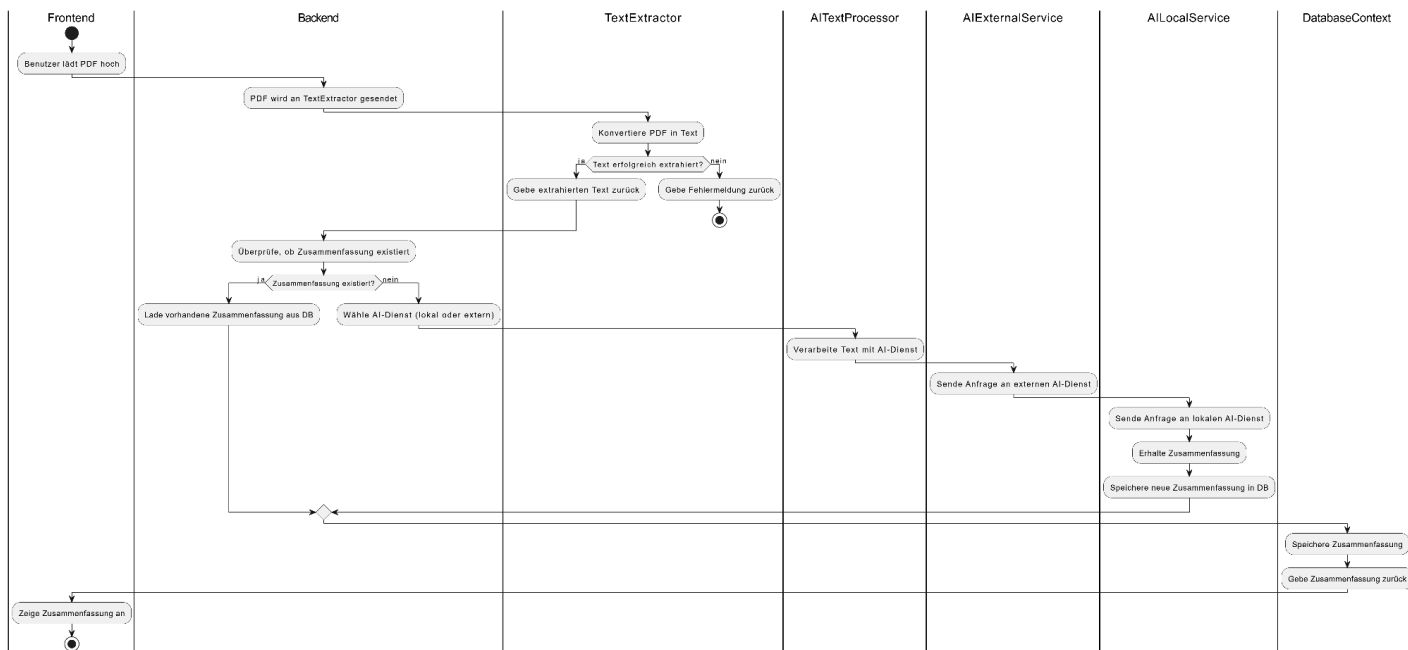
4.2.5.2.7.3. DatabaseContext.py

Dieses Modul verwaltet die Speicherung und den Abruf von Zusammenfassungen aus der Datenbank.

Kernfunktionen:

- save_summary: Speichert eine neue Zusammenfassung.
- find_summary_by_text: Sucht nach einer bestehenden Zusammenfassung.

4.2.5.2.7.4. Aktivitätsdiagramm für die Zusammenfassungsfunktionalität



4.2.6. Anki Service

4.2.6.1. Übersicht

Der Anki-Service ist eine Flask-basierte Webanwendung, die Benutzern das Erstellen, Verwalten und Überprüfen von digitalen Lernkarten (Flashcards) ermöglicht. Die Anwendung verwendet SQLAlchemy zur Datenbankinteraktion und unterstützt CORS für die Kommunikation mit Frontend-Anwendungen.

4.2.6.2. Hauptkomponenten

4.2.6.2.1. Datenbankmodelle (database.py)

- Card: Repräsentiert eine Lernkarte mit folgenden Feldern:
 - front: Vorderseite der Karte
 - back: Rückseite der Karte
 - next_review: Nächste Überprüfung
 - deck_id: Zugehöriges Deck
 - interval: Wiederholungsintervall
 - ef: Effizienzfaktor
 - repetitions: Anzahl der Wiederholungen
 - lapses: Anzahl der Fehler
- Deck: Eine Sammlung von Karten mit:
 - name: Name des Decks

- description: Beschreibung
- AnkiDatabaseContext: Verwaltung der Datenbankverbindung mit Methoden zum Erstellen, Abrufen, Aktualisieren und Löschen von Karten und Decks.

4.2.6.2.2. API-Endpunkte (app.py)

- Gesundheitscheck:
 - GET /health: Prüft den Serverstatus.
- Kartenverwaltung:
 - POST /cards: Erstellt eine neue Karte (benötigt front, back, deck_id im JSON-Format).
 - GET /cards/<int:card_id>: Ruft eine spezifische Karte ab.
 - PUT /cards/<int:card_id>: Aktualisiert eine Karte basierend auf Benutzerinteraktionen.
 - DELETE /cards/<int:card_id>: Löscht eine Karte.
- Deckverwaltung:
 - POST /decks: Erstellt ein neues Deck.
 - PUT /decks/<int:deck_id>: Aktualisiert ein bestehendes Deck.
 - GET /decks: Listet alle Decks auf.
 - GET /decks/<int:deck_id>: Ruft ein spezifisches Deck und dessen Karten ab.
- Karten-Generierung:
 - POST /generate-cards: Generiert Karten basierend auf externem Input.

4.2.6.2.3. Dockerfile (Dockerfile.anki)

- Definiert die Umgebung und enthält Installationsbefehle sowie den Startbefehl für den Flask-Server.

4.2.6.2.4. Abhängigkeiten (requirements.txt)

- Enthält alle benötigten Python-Pakete:
 - Flask: Web-Framework
 - SQLAlchemy: ORM für Datenbankinteraktionen
 - psycopg2-binary: PostgreSQL-Treiber
 - Requests: HTTP-Anfragen

4.2.6.2.5. Tests (test.py)

- Beinhaltet grundlegende Tests zur Überprüfung der API-Endpunkte.

4.2.6.3. Funktionsweise

4.2.6.3.1. Datenbankinteraktion

Der AnkiDatabaseContext verwaltet die Verbindung zur PostgreSQL-Datenbank und führt CRUD-Operationen für Karten und Decks aus.

4.2.6.3.2. API-Logik

Die Flask-Anwendung stellt verschiedene REST-API-Routen zur Verfügung, die Benutzeranfragen verarbeiten und Daten in JSON-Format zurückgeben.

4.2.6.3.3. Fehlerbehandlung

Das System behandelt gängige Fehlerfälle wie ungültige Eingaben, nicht gefundene Ressourcen und Datenbankfehler, um eine robuste Nutzererfahrung zu gewährleisten.

4.2.6.4. Update-Logik für Karten

Die `update_card_logic`-Funktion ist der Kern des spaced repetition Algorithmus. Sie aktualisiert Intervalle und Effizienzfaktoren (EF) basierend auf der Benutzerantwort.

4.2.6.4.1. Implementierung

```
def update_card_logic(button, interval, ef, repetitions, lapses)
    if button == "Again":
        interval = 10 / 60 # 10 Minuten in Tagen
        repetitions = 0
        lapses += 1
    elif button == "Hard":
        interval = max(1, interval * 1.2)
        ef = max(1.3, ef - 0.15)
    elif button == "Good":
        interval = interval * ef
        repetitions += 1
    elif button == "Easy":
        interval = interval * ef * 1.3
        ef = min(3.0, ef + 0.15)
        repetitions += 1

    return {
        "next_interval": interval,
        "new_ef": ef,
        "repetitions": repetitions,
        "lapses": lapses,
    }
```

4.2.6.4.2. Parameter

- `button`: Benutzerbewertung der Antwort (Again, Hard, Good, Easy).
- `interval`: Zeit bis zur nächsten Wiederholung.
- `ef`: Effizienzfaktor, der das Lernverhalten widerspiegelt.
- `repetitions`: Anzahl erfolgreicher Wiederholungen.
- `lapses`: Anzahl falscher Antworten.

4.2.6.4.3. Logik

- Again: Setzt das Intervall auf 10 Minuten und erhöht lapses.
- Hard: Erhöht das Intervall um 20% und reduziert ef minimal.
- Good: Multipliziert das Intervall mit ef und erhöht repetitions.
- Easy: Multipliziert das Intervall mit $ef * 1.3$, erhöht ef und repetitions.

4.2.6.5. Kartenparameter

Beim Abrufen von Karten werden die Parameter in ein Array eingefügt:

```
for card in cards:
    result.append({
        "id": card.id,
        "front": card.front,
        "back": card.back,
        "next_review": card.next_review,
        "deck_id": card.deck_id,
        "interval": card.interval,
        "ef": card.ef,
        "repetitions": card.repetitions,
        "lapses": card.lapses
    })
```

4.2.6.6. Fazit

Der Anki Service optimiert das Lernen durch spaced repetition. Die update_card_logic-Funktion passt Intervalle und Effizienzfaktoren basierend auf Benutzerantworten an.

Dank einer durchdachten Datenbankstruktur und einer flexiblen API können Benutzer ihre Lernkarten effizient verwalten und ihren Fortschritt systematisch nachverfolgen.

4.2.7. Fehler und Erkenntnisse aus dem Projekt

4.2.7.1. Fehler

- Frontend und Backend werden separat implementiert, und die Verknüpfung zwischen die beiden haben wir zu lange verzögert
- Die Verknüpfung zwischen Frontend und Backend wurde unterschätzt, was zu Verzögerungen führte.
- Für das Hosting des Frontends über unsere Domain haben wir Nginx als Reverse Proxy verwendet. Dabei traten jedoch unerwartete Komplikationen auf, möglicherweise aufgrund von fehlerhaften Konfigurationen oder Kommunikationsproblemen zwischen Nginx und den Backend-Services

4.2.7.2. Was wir daraus gelernt haben

- Eine frühzeitige Abstimmung zwischen Backend- und Frontend-Teams hätte Probleme mit der Integration minimiert.
- Eine bessere Planung bei der Nutzung von Nginx und anderen Server-Technologien hätte helfen können, Probleme frühzeitig zu vermeiden.

4.3. Künstliche Intelligenz

Um den Lernassistenten mit Künstlicher Intelligenz auszustatten, basiert unsere Architektur auf einer intelligenten Kombination aus zwei unterschiedlichen Komponenten: einem lokalen Dienst sowie einem externen Dienst.

Diese beiden Module arbeiten Hand in Hand, um eine optimale Leistung und Zuverlässigkeit zu gewährleisten. Dabei kombinieren wir ein leistungsfähiges Sprachmodell, das direkt lokal auf dem Server arbeitet, mit einer externen modernen KI-Plattform.

Durch diese doppelte Struktur wird sichergestellt, dass der Lernassistent auch dann weiterhin funktioniert, wenn die externe Plattform temporär nicht verfügbar ist oder ihre Nutzungsgrenzen erreicht wurden. Die Integration dieser beiden Dienste bietet eine wichtige Sicherheitsschicht, dies stellt sicher, dass die Lernenden stets Zugriff auf KI-gestützte Funktionen haben, unabhängig von äußeren Einflüssen.

Das System selbst ist so konzipiert, dass es tokenbasiert operiert. Das bedeutet, dass jeder einzelne Benutzer über eine bestimmte Anzahl an Tokens verfügt, die er für Anfragen an das externe Modell verwenden kann. Sobald diese vorgegebene Anzahl an Tokens aufgebraucht ist, schaltet das System automatisch um. In diesem Fall wird das lokale Modell aktiviert, dies arbeitet aufgrund der begrenzten Ressourcen langsamer, ist aber dennoch in der Lage, qualitativ hochwertige Inhalte zu generieren.

Da das System mit sensiblen Nutzerdaten arbeitet, wurden verschiedene Sicherheitsmaßnahmen implementiert. Die Kommunikation mit Groq erfolgt verschlüsselt über HTTPS, sodass keine ungesicherten Datenübertragungen stattfinden. Das lokale Modell arbeitet vollständig offline und stellt damit eine sichere Alternative dar.

4.3.1. Lokaler Dienst

Die lokale Architektur besteht aus mehreren Komponenten: Einem API-Server, der Anfragen entgegennimmt, einer Modellverwaltungsschicht und einer Inferenz-Engine, die das Modell ausführt. Ollama übernimmt dabei die Optimierung des Modells für den lokalen Einsatz, indem es Speichermanagement und Model-Loading automatisiert. Ollama ist ein Programm, welches die Verwendung von Sprachmodellen erheblich vereinfacht. Ollama stellt einen lokalen Server bereit. Diesen Server spricht der Lernassistent mit der Rest API des AI Dienstes an. Diese API ist ein autonomes Programm, welches in einem Docker Container läuft, sie wird im Folgenden vereinfacht nur als "API" referenziert.

Als lokales Sprachmodell setzt der Lernassistent auf das von Meta entwickelte Llama 3.2 mit 3 Milliarden Parametern, es wurde am 25. September 2024 veröffentlicht, bietet eine gute Leistung und unterstützt viele Sprachen.

Die API stellt verschiedene Funktionen zur Verfügung wie das Generieren von Antworten oder das Laden unterschiedlicher Sprachmodelle. Alle weiteren Funktionen sind in der API Dokumentation unter https://github.com/Magn4/Lernassistent/blob/main/Documentation%20%26%20Artifacts/Artifacts/LLM/LLM_Rest_API_Documentation.pdf zu finden.

Die API bereitet alles automatisch für die einfache Benutzung vor, sie überprüft die verfügbaren LLMs und lädt das eingestellte Modell, während es alle, die nicht benutzt werden, entlädt. Zusätzlich besitzt die API die Möglichkeit, auf das externe Sprachmodell zuzugreifen, sie besitzt ihren eigenen Zugang.

Im Gegensatz zum Backend, was Tokenbasiert entscheidet, ob der externe oder interne Dienst benutzt werden soll, schickt die API vorrangig alle Anfragen an den externen Dienst, bis die Anfragen aufgrund von Nutzungslimits abgelehnt werden. Erst danach greift die API auf das lokale Sprachmodell via Ollama zu.

Dieser doppelte Mechanismus mit getrennten Zugängen zum externen Dienst garantiert schnelle und zuverlässige Antworten für den Großteil der Aufgaben, die dem Sprachmodell gestellt werden.

4.3.2. Externer Dienst

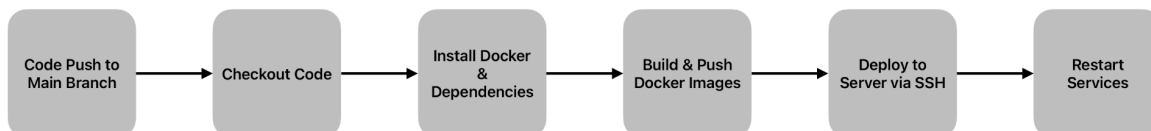
Um den Hardwarevoraussetzungen des Servers, auf dem der Lernassistent bereitgestellt wird, entgegenzutreten und schnellere Inferenzen zu ermöglichen, wird mithilfe einer API die Plattform Groq genutzt. Groq ist eine innovative Plattform, die leistungsstarke KI-Modelle bereitstellt. Sie wurde als externer KI-Dienst gewählt, da sie sich durch extrem schnelle Inferenzen mit minimalen Latenzen auszeichnet. Durch den Einsatz spezialisierter KI-Beschleuniger kann Groq Anfragen in wenigen Millisekunden verarbeiten, was für unsere Anwendung wichtig ist. Unser System nutzt die API von Groq für Textverarbeitungsaufgaben wie das Erstellen von Zusammenfassungen oder die automatische Generierung von Lernkarten. Darüber hinaus ermöglicht die Skalierbarkeit von Groq, dass unser System zukünftige Nutzeranstiege problemlos standhalten kann.

Ein weiterer Vorteil von Groq ist, dass die Plattform bis zu bestimmten Nutzungsgrenzen kostenlos genutzt werden kann. Für unseren externen Dienst haben wir uns dabei für das Sprachmodell "llama3-8b-8192" entschieden, das speziell für komplexe Textverarbeitungsaufgaben optimiert ist. Mit diesem Modell stehen uns bis zu 30 Anfragen pro Minute zur Verfügung, wobei wir maximal 6000 Tokens pro Minute generieren lassen können. Dies entspricht ungefähr 4500 Wörtern.

Durch die Kombination dieser Funktionen, schnelle Inferenzen, hohe Skalierbarkeit und kostenfreie Nutzung innerhalb der definierten Grenzen, stellt Groq einen wesentlichen Bestandteil unserer Architektur dar. Sie trägt dazu bei, dass unser Lernassistent performant arbeitet.

4.4. CI/CD-Pipeline für die Multi-Service-App

Die CI/CD-Pipeline (Continuous Integration / Continuous Deployment) automatisiert die Prozesse der Softwareentwicklung, des Testens und der Bereitstellung. Für dieses Projekt wird die Pipeline mit GitHub Actions implementiert. Sie automatisiert das Erstellen, Testen, Hochladen und Bereitstellen der Docker-Container.



4.4.1. Ablauf der CI/CD-Pipeline:

1. Auslösen der Pipeline
 - Die Pipeline startet automatisch bei einem Push auf den Main-Branch.
2. Code-Checkout
 - Der aktuelle Code wird aus dem GitHub-Repository abgerufen.
3. Docker Compose Installation
 - Docker Compose wird installiert, um das Multi-Service-Setup zu verwalten.
4. Docker Buildx einrichten
 - Buildx wird eingerichtet, um effizientes Multi-Architektur-Building zu ermöglichen.
5. Caching von Docker-Layern
 - Durch die Zwischenspeicherung von Docker-Layern wird die Build-Geschwindigkeit optimiert.
6. Login bei Docker Hub
 - Authentifizierung mit Docker Hub für den Upload der erstellten Images.
7. Build und Push der Docker-Images
 - Die Docker-Container werden mithilfe von Docker Compose gebaut und in Docker Hub hochgeladen.
8. Deployment auf dem Server via SSH
 - Verbindung zum Server über SSH.
 - Git-Repository wird aktualisiert (git pull).
 - Die neuesten Docker-Images werden gezogen (docker-compose pull).
 - Alte Container werden gestoppt und entfernt (docker-compose down).
 - Neue Container werden gestartet (docker-compose up -d).

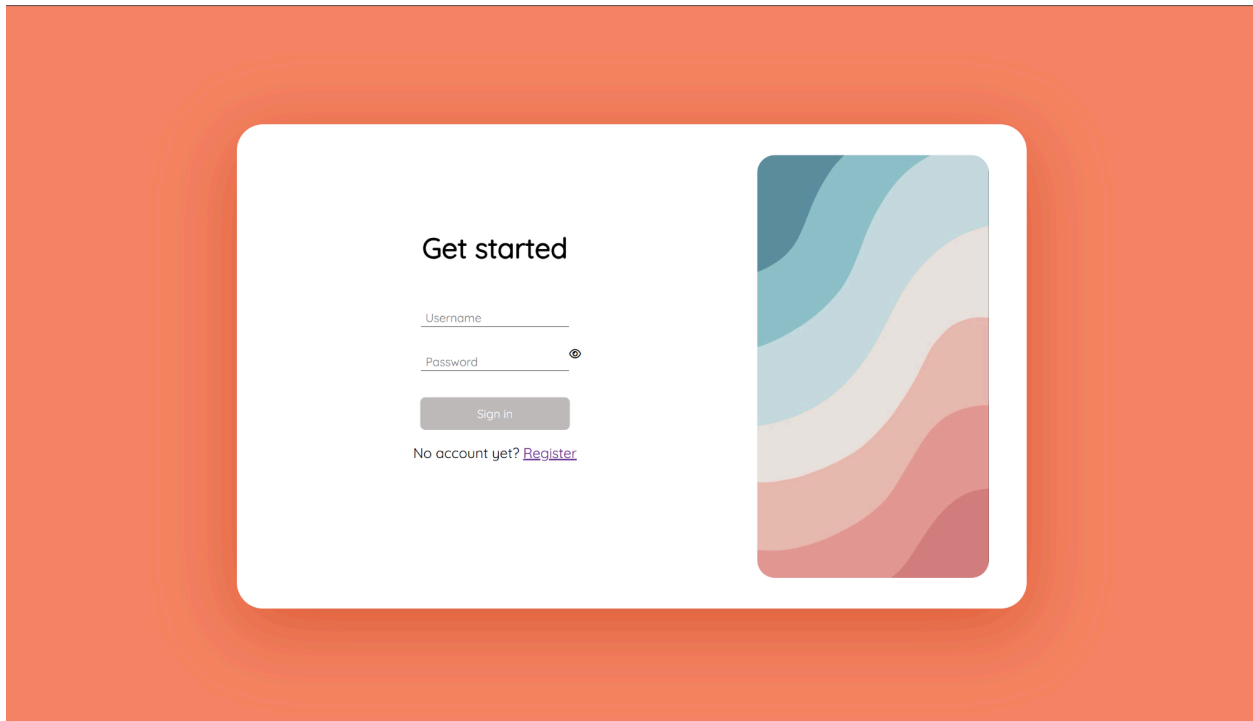
4.4.2. Technologien & Tools:

- GitHub Actions – Automatisierung der Pipeline
- Docker & Docker Compose – Containerisierung der Microservices
- Docker Hub – Container Registry für Images
- Buildx – Multi-Architektur-Builds und Performance-Optimierung
- SSH – Remote-Deployment auf den Server

5. User-Handbook

5.1. How to get started

Willkommen beim Lernassistenten, deinem KI-gestützten Helfer, wenn es um Zusammenfassungen und Karteikarten geht.



Get started

Max

.....

Sign in

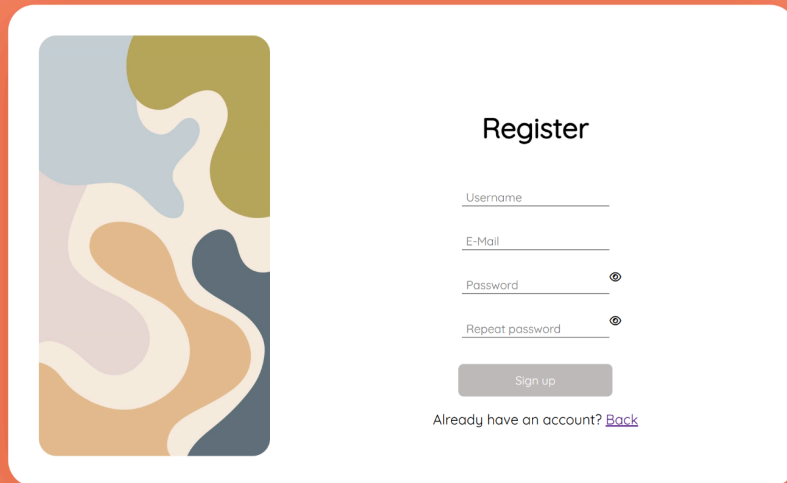
No account yet? [Register](#)

Um diesen Webdienst nutzen zu können, musst du dich mit deinem Benutzerkonto anmelden. Gib einfach deinen Benutzernamen und dein Passwort ein. Solltest du noch kein Konto haben, klicke einfach auf den Link und registriere dich. Es geht ganz schnell.

5.2. Registrieren

Hier kannst du dich registrieren, um den Lernassistenten nutzen zu können.


Zur Registrierung werden dein Benutzername, deine E-Mail und ein Passwort benötigt.




Register

Username

E-Mail

Password 

Repeat password 

Already have an account? [Back](#)

.....



Dein Passwort sollte aus mindestens 8 Zeichen und vier verschiedenen Zeichenarten (Groß-, Kleinbuchstaben, Zahlen und Sonderzeichen) bestehen. Du kannst mithilfe des Icons rechts das Passwort sichtbar machen, um deine Eingabe überprüfen zu können.

Strong-Pass-123!



Passwords do not match

Ok

Sollten deine Eingaben für das Passwort nicht übereinstimmen, wirst du darüber benachrichtigt. Bitte überprüfe in diesem Fall nochmal dein Passwort.

Register

Max

max@gmail.com

.....

.....

Sign up

Already have an account? [Back](#)

Das Erstellen eines Benutzerkontos ist erfolgreich, sobald alle vier Felder korrekt ausgefüllt wurden und sofern deine E-Mail nicht bereits bei uns mit einem anderen Konto verknüpft ist. Du wirst dann auf die Dashboard Seite des Lernassistenten weitergeleitet und kannst nun mit dem Lernen beginnen! Solltest du doch schon ein Konto bei uns besitzen, kannst du mithilfe des unten stehenden Links zurück zum Login gelangen.

5.3. Dashboard

Dies ist die Startseite des Lernassistenten, das Dashboard. Du erhältst hier eine kleine Einweisung, was das Programm für dich zu bieten hat. Verschaffe dir gerne einen Überblick.

Learning Assistant

- Dashboard
- File Manager
- Anki Cards

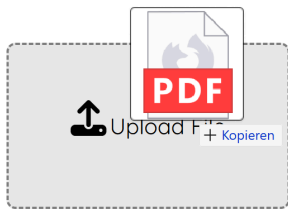
Log out

Dashboard

Generate summaries and AnkiCards in one click with the help of the Learning Assistant. Simply upload a PDF and be surprised how easy learning can be!

Upload File

- **Dashboard:** Try it yourself. Upload your first PDF!
- **File Manager:** Find all your already uploaded slides / summaries.
- **Anki Cards:** No master has yet fallen from the sky! Learn with Anki Cards to be well prepared for your exams.



Sobald du bereit bist, kannst du deine erste PDF mit den Vorlesungsfolien, die du lernen möchtest, in das Feld „Upload File“ per Drag and Drop ziehen.

You dropped 1 file(s).

Ok

Dir erscheint daraufhin diese Meldung.

Nachdem du die Nachricht mit okay bestätigt hast, öffnet sich dieses Modal. Hier kannst du den Namen des Moduls angeben, zu welchem die Folien gehören, die du gerade hochgeladen hast und einen Namen für das Thema der Vorlesung angeben. Klicke auf Save, um das Hochladen abzuschließen. Durch einen Klick auf Quit kannst du den Vorgang jederzeit abbrechen.

Learning Assistant

- Dashboard
- File Manager
- Anki Cards

Log out

Dashboard

Generate summaries and AnkiCards in one click with the help of the Learning Assistant. Simply upload a PDF and be surprised how easy learning can be!

- **Dashboard:** Try it yourself! Upload a PDF!
- **File Manager:** Find all your uploaded slides / summaries.
- **Anki Cards:** No master list! Learn with Anki Cards for your exams.

Store File

Quit

Save

Upload File

Store File

Analysis

IT-Security

CompNet

Quit

Save

Store File

Directoryname

Lecture 01

Lecture 02

Lecture 03

Wenn du den Lernassistenten nicht zum ersten Mal benutzt, werden dir die Eingabefelder Module und Ordner, die du bereits angelegt hast, in einer Dropdown Liste vorschlagen. Du kannst neue Module und Ordner erstellen, oder bereits vorhandene auswählen.

Modulename: Analysis, Directoryname: Lecture 02

Ok

Nachdem du auf Save gedrückt hast, erscheint eine kleine Meldung, um die Eingabe zu bestätigen. Ohne dass du es merkst, erstellt dir der Lernassistent im Anschluss deine Zusammenfassung und Karteikarten.

5.4. Navigationsleiste

Learning Assistant

Dashboard

File Manager

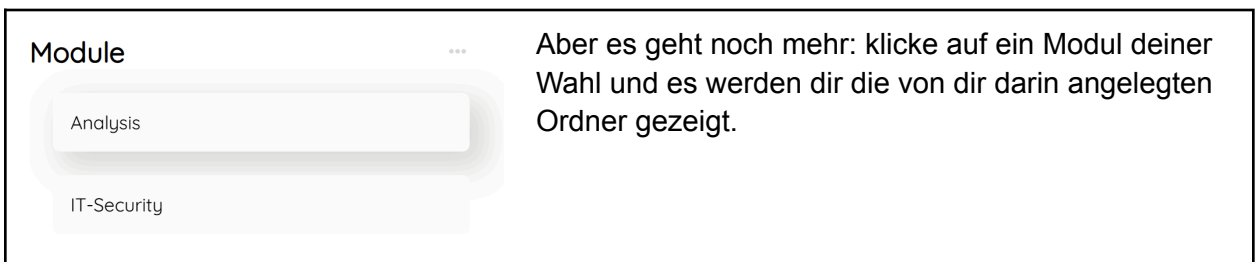
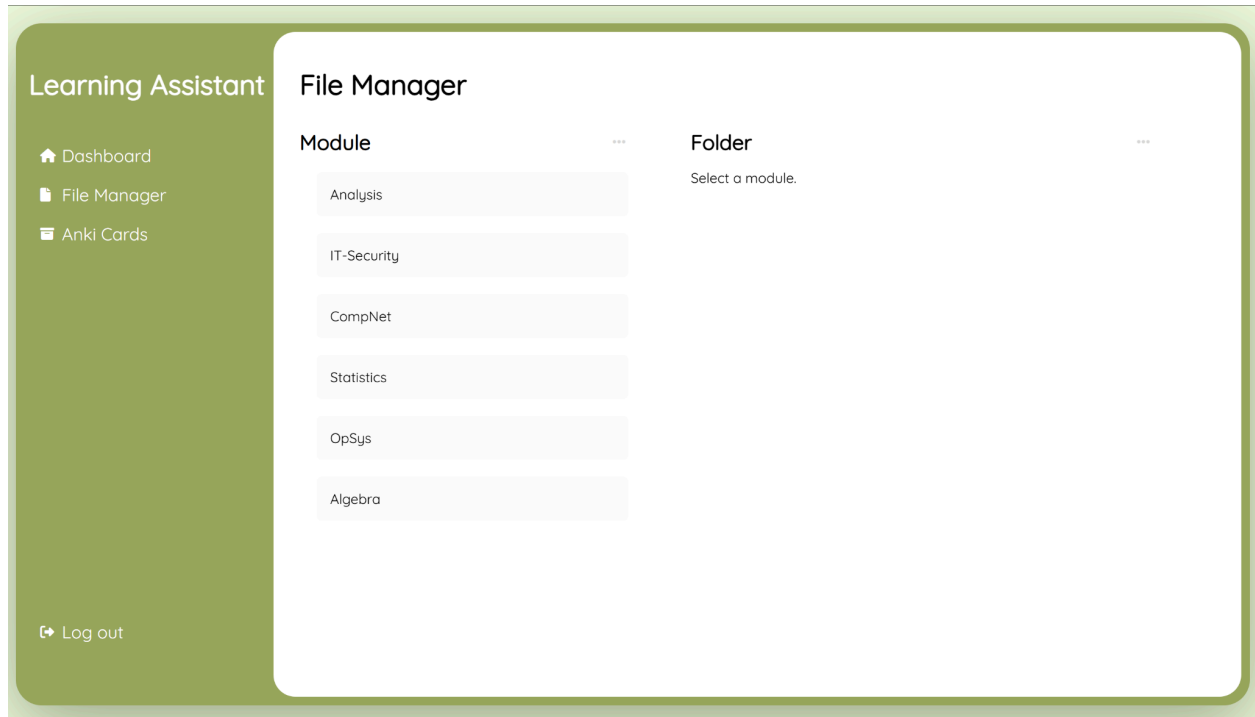
Anki Cards

Log out

Wenn du nun deine Dokumente einsehen möchtest, solltest du einen Blick auf die Navigationsleiste werfen, die sich auf der linken Seite befindet. Hier kannst du durch die Seiten des Lernassistenten navigieren. Klicke nun auf File Manager, um zu deiner zentralen Ablage für Dokumente zu gelangen.

5.5. File Manager

Im File Manager findest du eine Übersicht aller von dir angelegten Module. Wenn du bereits für viele Module gelernt hast, kannst du durch die Liste scrollen.



Aber es geht noch mehr: klicke auf ein Modul deiner Wahl und es werden dir die von dir darin angelegten Ordner gezeigt.

Das sieht dann wie folgt aus:

File Manager

Module ...

Analysis

IT-Security


CompNet


Statistics


OpSys


Algebra


Folder ...


Lecture 01


Lecture 02


Lecture 03



Lecture 04



Lecture 05


Files ...

Select a folder.

Folder


Lecture 01


Lecture 02


Lecture 03

Klicke nun auf einen Ordner, und du gelangst zu den darin abgelegten Dateien.

Module ...

Analysis

IT-Security


CompNet


Statistics


OpSys


Algebra


Folder ...


Lecture 01


Lecture 02


Lecture 03


Lecture 04


Lecture 05

Files ...

Lecture slides

Summary

AnkiCards

Unter den Dateien befindet sich nicht nur deine hochgeladene PDF, sondern auch die von der KI generierte Zusammenfassung und Karteikarten.

Files

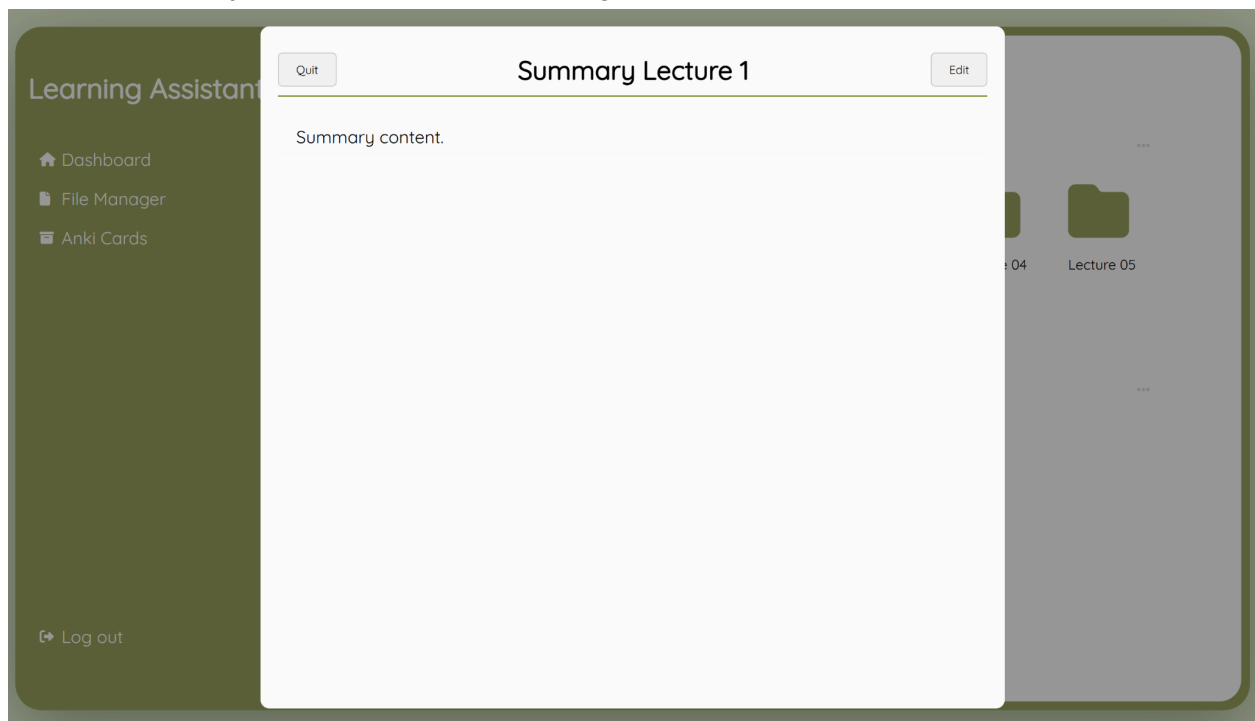
Lecture slides

Summary

AnkiCards

Möchtest du dir nun die Zusammenfassung durchlesen, kannst du darauf klicken und es öffnet sich ein Fenster für die schreibgeschützte Ansicht.

Natürlich ist eine Zusammenfassung länger als hier im Beispiel. Falls sie so lang ist, dass sie nicht auf deinen Bildschirm passt, kannst du einfach nach unten scrollen und weiterlesen. Und solltest du hier und da etwas umformulieren wollen, klicke auf den Edit Button oben rechts und du gelangst in den Bearbeitungsmodus. Dieser Button behält seine Position auch wenn du scrollst, damit du jederzeit einen schnellen Zugriff darauf hast.



Quit

Summary Lecture 1

Save

Summary content edit.

Im Bearbeitungsmodus lässt sich das Fenster an sich nicht scrollen, aber innerhalb des Textfeldes ist es möglich. Deine Änderungen kannst du, wenn du fertig bist, mit dem Save Button speichern. Möchtest du die Zusammenfassung schließen, kannst du jederzeit auf Quit drücken. Achte aber darauf, vorher zu speichern, wenn deine Veränderungen nicht verloren gehen sollen!

Module

...

Analysis

Den File Manager kannst du auch verwalten. Um Module oder Ordner hinzuzufügen, kannst du auf das ellipsenförmige Icon jeweils oben rechts klicken.

Enter the name for the new module:

RA|

Ok

Abbrechen

Du wirst nun aufgefordert, den Namen des Moduls beziehungsweise des Ordners anzugeben. Bestätige mit okay oder breche den Vorgang wieder ab.

Module

OpSys

Algebra

Programmieren C

TI

SWEA

RA

Dein neu angelegtes Element erscheint nun am Ende der bereits bestehenden.

File Manager

Module

OpSys

Algebra

Programmieren C

TI

SWEA

RA

Folder



Lecture 01



Lecture 02

Files

Select a folder.

Would you like to delete this element?

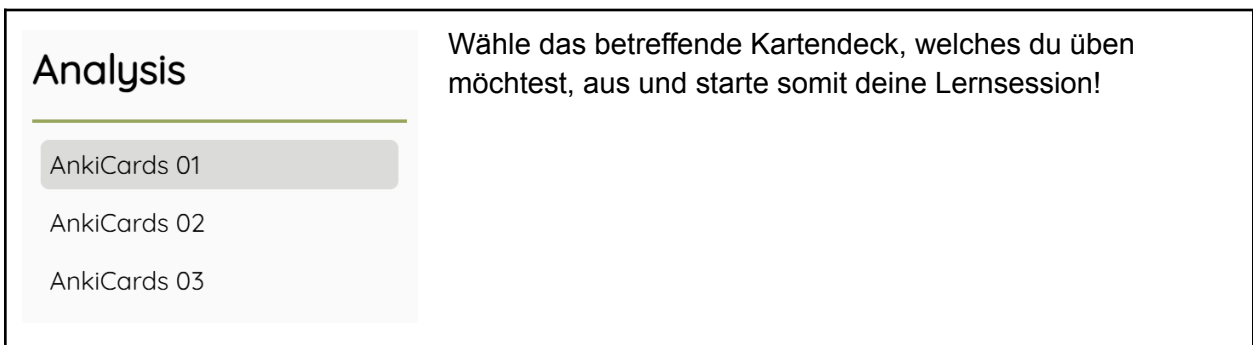
Ok

Abbrechen

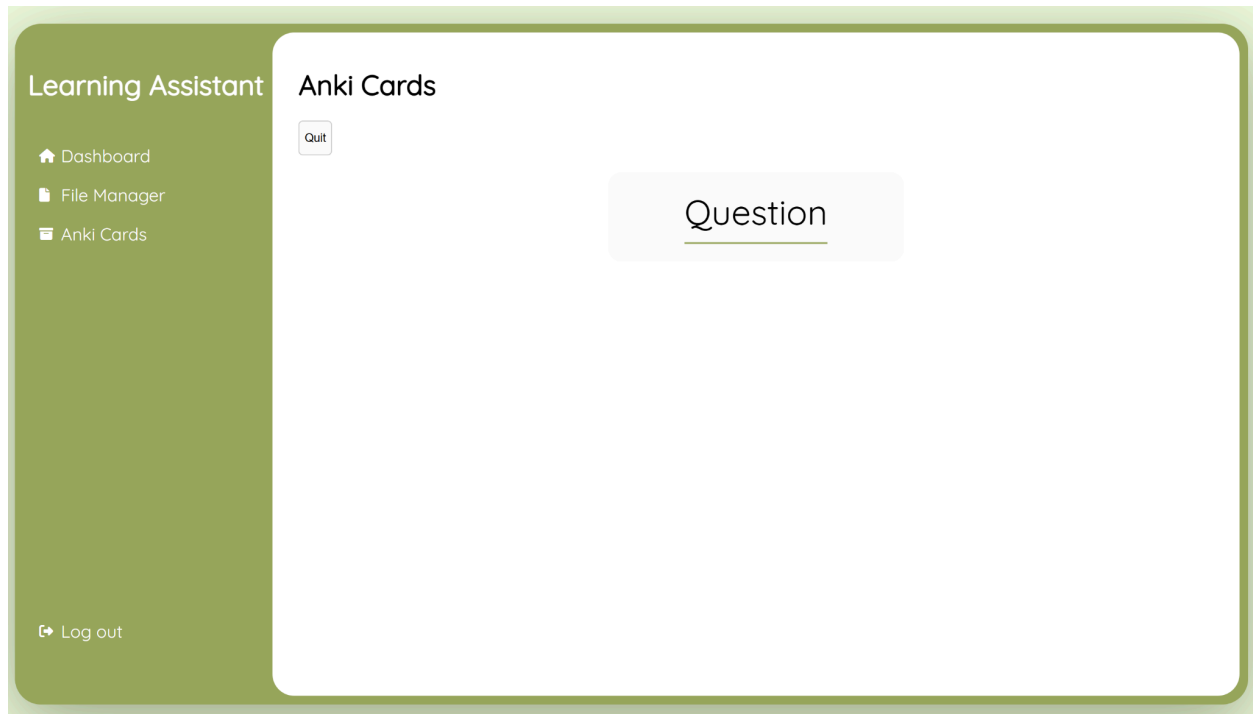
Wenn du ein Modul entfernen möchtest, wähle es per Rechtsklick aus und bestätige die Meldung mit okay oder breche den Vorgang ab. Du kannst auch Ordner so entfernen. Löscht du ein Modul, so werden, sofern es nicht leer ist, alle darin enthaltenen Ordner und Dateien mitgelöscht. Gleiches gilt beim Entfernen eines Ordners für die darin enthaltenen Dateien. Sei bitte vorsichtig, da dieser Vorgang nicht rückgängig gemacht werden kann.

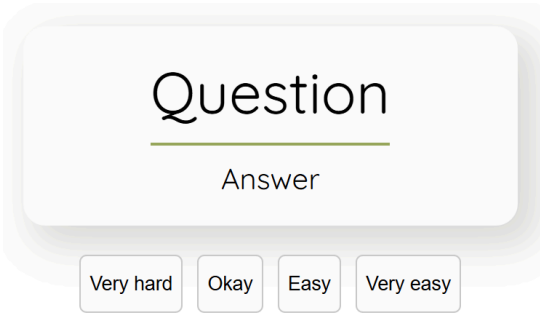
5.6. Anki Cards

Wenn du bereit bist für eine Lernsession, navigiere dich zu den Anki Cards. Hier findest du deine Module und die darin erstellten Karteikarten.




Eine Lernsession läuft dabei wie folgt ab: bei jeder neuen Karte wird zu Beginn nur die Frage angezeigt. Ob du die richtige Antwort dazu weißt, kannst du mit einem Klick auf die Karte überprüfen.



The image shows a card interface for a question. It has a light gray rounded rectangle with 'Question' and a horizontal line, followed by 'Answer'. Below the card are four buttons: 'Very hard', 'Okay', 'Easy', and 'Very easy'.

Es wird dir nun die richtige Antwort und einige Buttons angezeigt. Klicke wieder auf die Karte selbst, und du siehst erneut nur die Frage. Teste, ob du die Antwort aus dem Kopf formulieren kannst. Sobald du bereit für die nächste Frage bist, klicke auf einen der Buttons und gib dem Lernassistenten ein Feedback, wie einfach dir das Beantworten der Frage gefallen ist. Deine Antwort nimmt darauf Einfluss, in welchem Intervall dir die Karte während deiner Lernsession angezeigt wird. Zögere also nicht, zuzugeben, dass es dir noch schwer fällt. Das wird sich bald ändern. Deine Lernsession kannst du jederzeit mithilfe des Quit Buttons beenden.

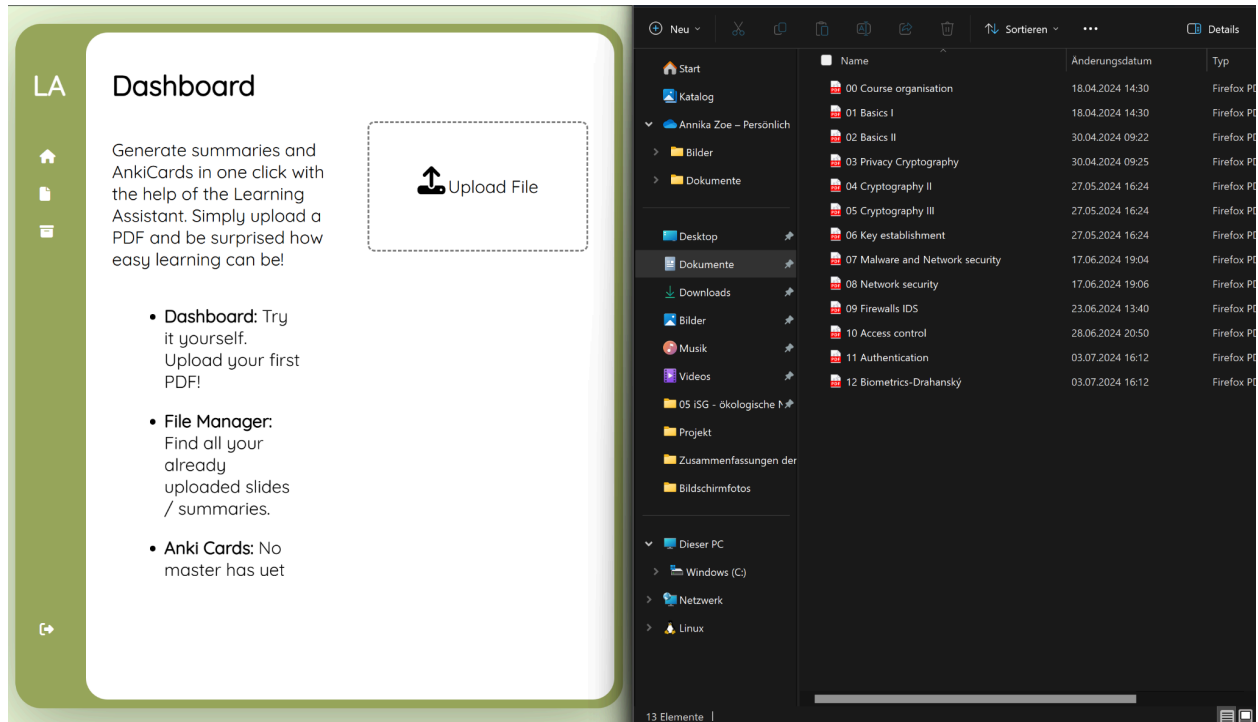
5.7. Ausloggen

The image shows a green button with a door icon and the text 'Log out'.

Wenn du den Lernassistenten verlassen möchtest, kannst du dich mithilfe des Log out Buttons in der Navigationsleiste abmelden.

5.8. Split View

Einen letzten Tipp, bevor du loslegen kannst, gibt es noch: der Lernassistent lässt sich jederzeit auch im Split View nutzen! Probier es einfach aus.



Und nun: viel Erfolg beim Lernen mit deinem Lernassistenten!

6. Ergebnis/Fazit

Im Verlauf unseres Projekts haben wir erfolgreich eine Vielzahl der geplanten Funktionen umgesetzt und ein leistungsfähiges Lernassistenten-Tool entwickelt. Dazu gehören insbesondere die PDF-Textverarbeitung, die automatische Erstellung von Zusammenfassungen sowie die Generierung und Verwaltung von Lernkarten in Form von AnkiCards. Zusätzlich haben wir eine Login- und Registrierungsfunktion implementiert, einen File-Manager zur Organisation und Verwaltung von hochgeladenen Dateien integriert und die Möglichkeit geschaffen, AnkiCards aktiv zu lernen. Ein wesentlicher Bestandteil war zudem die Nutzung von Sprachmodellen, sowohl lokal als auch über externe Dienste, um das Erstellen von Zusammenfassungen und AnkiCards zu automatisieren.

Unser ursprünglicher Plan, das System als reine Java-Anwendung zu realisieren, hat sich im Laufe der Entwicklung mehrfach geändert. Über eine Kombination aus HTML-Frontend und C#-Backend sind wir letztlich bei einer Architektur mit HTML, CSS, Python und Flask gelandet.

Das Projekt hat uns vor zahlreiche Herausforderungen gestellt, sei es in der technischen Umsetzung, der Anpassung der Architektur, der Integration von KI-gestützten Funktionen oder auch der Verknüpfung zwischen Frontend und Backend. Trotz dieser Herausforderungen war es eine sehr lehrreiche Erfahrung für alle Teammitglieder. Jeder von uns konnte viel über Softwarearchitektur, KI-gestützte Anwendungen, CI/CD Pipelines und die praktische Umsetzung von einem Lernsystem mitnehmen. Besonders spannend war es zu sehen, wie sich unsere Idee von der ersten Planung bis zum finalen Produkt entwickelt hat. Trotz ständig neuer kleiner oder großer Hürden hat das Projekt viel Spaß gemacht und war ein wertvoller Beitrag zu unserem Verständnis moderner Softwareentwicklung.

7. Ausblick

Das Hauptziel unseres Programms wurde erfüllt. Unsere Künstliche Intelligenz interagiert mit zur Verfügung gestellten Dokumenten und fasst diese zusammen. Außerdem werden auf Anfrage automatisch Fragen und Antworten für unsere Anki Card Decks erstellt.

7.0.0.1. Handschriftliche Dokumente erkennen

Da unsere KI bereits in der Lage ist, PDF-Dokumente zu scannen und diese zusammenzufassen, sollen als nächstes handschriftliche Notizen erkannt werden. Dafür wird ein OCR-Scanner benötigt. Das Backend stellt Endpoints zur Verfügung und das Frontend muss Modale so überarbeiten, dass man zwischen normalen PDF-Dokumenten und Notizen entscheiden kann.

7.0.0.2. Zusammenfassungen verändern

Der Benutzer soll in Zukunft die Möglichkeit haben, die Zusammenfassungen des Programms in verschiedenen Versionen zu erhalten. Zum Beispiel bei komplexen Themen entscheidet man sich für eine Zusammenfassung, die einfacher geschrieben ist.

7.0.0.3. Neue Lernmethoden

Zusätzlich zu den Anki Cards sollen weitere Features wie Spaced Repetition Systems und simulierte Probeklausuren hinzugefügt werden. In Zukunft soll die künstliche Intelligenz persönliche Schwächen erkennen und diese mit gezielten Verbesserungsvorschlägen ansprechen.

7.0.0.4. Sharepoint für Dokumente und Lernkarten

In Zukunft sollen alle Benutzer die Möglichkeit haben, sich untereinander auszutauschen und ihre Dokumente in einem zentralen Sharepoint miteinander zu teilen.

7.0.0.5. Gamification

Das Programm kann eine Art Belohnungssystem erhalten, um Benutzer auf lange Sicht zu halten. Sprich man erhält Erfolge und Abzeichen bei anhaltenden Lern-Streaks oder Punkte beim Abschließen eines Decks. Zudem soll der Benutzer eine Möglichkeit haben, alle Daten einzusehen. Diese Statistiken können auf einer eigenen Seite hinterlegt sein (Ausschnitte können auf dem Dashboard hinterlegt sein).

7.0.0.6. Kontinuierliches Design

Durch das Anbinden des Backends sind inkonsistente Designs entstanden. Diese müssen abgeändert werden, um das Aussehen und die Struktur des Programms in allen Aspekten harmonisch zu gestalten.

7.0.0.7. Erweiterung durch verschiedene Sprachen und Farbvarianten

Um jedem Benutzer ein individuelles Erlebnis zu gewährleisten, wird in Zukunft daran gearbeitet, mehrere Ausgaben unseres Programms zur Verfügung zu stellen (mehrere Sprachen zur Auswahl, mehrere Farben zur Auswahl).

8. Arbeitsaufteilung

8.1. Alle

- Planung der Programmstruktur
 - Grundlegendes Architekturkonzept entworfen.
 - Verteilung der Microservices definiert.
 - Technologiestack festgelegt.

8.2. Frontend-Team: (Annika Zoe und Finn Schäffler)

- Für genauere Einsicht in die Entwicklung des Frontends kann in das separate Github Repository: <https://github.com/A-Z-Schaeffler/Lernassistent-Frontend> geschaut werden. Die HTML- und CSS-Dateien sind unter src → main → View zu finden.
- Implementierung von übergreifenden Frontend-Styles in Styles.css.
- Login GUI:
 - Statisches Design, Implementierung eines dynamischen Verhaltens sowie Vorbereitung für Backendverknüpfung in Login.html mit integriertem JavaScript-Teil und LoginStyles.css.
- Register GUI:
 - Statisches Design, Implementierung eines dynamischen Verhaltens sowie Vorbereitung für Backendverknüpfung in Register.html mit integriertem JavaScript-Teil und RegisterStyles.css.
- Navigation GUI:
 - Statisches Design sowie Implementierung eines dynamischen Verhaltens in Navigation.html mit integriertem JavaScript-Teil und NavigationStyles.css.
- Dashboard GUI:
 - Vorbereitung für Backendverknüpfung in Dashboard.html mit integriertem JavaScript-Teil.
- FileManager GUI:
 - Statisches Design sowie Implementierung eines dynamischen Verhaltens des Modals zum Anzeigen und Bearbeiten von Zusammenfassungen in FileManager.html mit integriertem JavaScript-Teil und FileManagerStyles.css.

8.3. Annika Zoe Schäffler - Matrikelnummer: 1453180

- Dashboard GUI:
 - Statisches Design sowie Implementierung eines dynamischen Verhaltens des Modals zur Abfrage von Modul- und Ordnernamen in Dashboard.html mit integriertem JavaScript-Teil und DashboardStyles.css.
- FileManager GUI:
 - Statisches Design, Implementierung eines dynamischen Verhaltens sowie Vorbereitung für Backendverknüpfung in FileManager.html mit integriertem JavaScript-Teil und FileManagerStyles.css.
- Sonstige Aufgaben
 - Dokumentation

8.4. Finn Schäffler - Matrikelnummer: 1350212

- Navigation GUI:
 - Dynamisches Laden der Navigation.html Datei in Dashboard.html, FileManager.html und AnkiCards.html sowie Implementierung einer dynamischen Anpassung an die halbe Bildschirmbreite in Navigation.html mit integriertem JavaScript-Teil und NavigationStyles.css.
- Dashboard GUI:
 - Statisches Design sowie Implementierung eines dynamischen Verhaltens in Dashboard.html mit integriertem JavaScript-Teil und DashboardStyles.css.
- AnkiCards GUI:
 - Statisches Design, Implementierung eines dynamischen Verhaltens sowie Vorbereitung für Backendverknüpfung in AnkiCards.html mit integriertem JavaScript-Teil und AnkiCardsStyles.css.
- Sonstige Aufgaben
 - Dokumentation

8.5. Backend-Team: (Taha, Nisrine, Hamzaoui)

- Die Architektur des Backends
- Die Entscheidung für den Wechsel der Programmiersprache und die Implementierung in Python (Flask)
- Testing und Fehlerbehebung
- Entwicklung und Implementierung der Kern-Flask-Anwendung zur Verarbeitung von HTTP-Anfragen und -Antworten.
- Integration und Verwaltung externer und lokaler KI-Dienste zur Textverarbeitung
- Entwicklung von Controllern, die HTTP-Anfragen entgegennehmen und entsprechende Antworten generieren

8.6. Mohammed Taha El Youssefi - Matrikelnummer: 1454943

- Infrastruktur & Deployment
 - Implementierung der Docker-Compose- und Docker-Infrastruktur.
 - Migration der gesamten Bereitstellung auf den Server und Verwaltung aller Ports, um extern auf die API zugreifen zu können.
 - Wechsel zu Caddy anstelle von NGINX aufgrund seiner Einfachheit und Benutzerfreundlichkeit für HTTPS.
 - Hinzufügen einer Token-Authentifizierung für die API im Reverse Proxy.
 - Zuordnung des Domainnamens zur Server-IP.
- Forschung & Problemlösung
 - Umfangreiche Recherche zu:
 - Reverse Proxy & CORS
 - Microservices-Architektur
 - Anki-Karten Logik
- API- und Backend-Entwicklung
 - Implementierung des /health-Endpunkts zur Überprüfung des Serverstatus.
 - Entwicklung des Card-Modells mit Feldern für Lernkarten (Vorderseite, Rückseite, Wiederholungsintervall).
 - Entwicklung des Deck-Modells mit Feldern für das Lern Kartendeck (Name, Beschreibung).
 - Entwicklung des Anki DatabaseContext zur Verwaltung der PostgreSQL-Datenbank Verbindung und CRUD-Operationen für Karten und Decks.
- Karten- und Deck-Verwaltung (API-Endpunkte)
 - Lernkarten:
 - POST /cards – Erstellen neuer Lernkarten.
 - GET /cards/int:card_id – Abrufen einer spezifischen Karte.
 - PUT /cards/int:card_id – Aktualisieren von Karten nach Benutzerinteraktionen.
 - DELETE /cards/int:card_id – Löschen einer Karte.
 - Decks:
 - POST /decks – Erstellung eines neuen Decks.
 - PUT /decks/int:deck_id – Aktualisierung eines bestehenden Decks.
 - GET /decks – Auflistung aller Decks.
 - GET /decks/int:deck_id – Abrufen eines spezifischen Decks und seiner Karten.
 - Zusätzliche Funktionen:
 - POST /generate-cards – Generierung von Karten basierend auf externem Input.
- Lernlogik & Algorithmus:

- Implementierung der update_card_logic-Funktion zur Berechnung und Aktualisierung von Intervallen und Effizienzfaktoren (EF) basierend auf den Benutzerantworten.
- Implementierung der Logik für die unterschiedlichen Antwortarten:
 - Again
 - Hard
 - Good
 - Easy
- Frontend-Integration:
 - Implementierung der API-Funktionen im Frontend durch Änderung des Codes.
- Sonstige Aufgaben
 - Dokumentation

8.7. Mohamed Hamzaoui - Matrikelnummer: 1462582

- Recherche über Lernstrategie: wie Lernpläne erstellt und durch Technologie angepasst werden können. Mit NLP und Priorisierung werden Inhalte strukturiert, während adaptive Algorithmen den Plan je nach Nutzerfeedback anpassen. mehr Details [hier](#).
- Implementierung von folgenden Endpoints mit folgenden Features:
 - Register:
 - Hashing der Passwörter mit einem sicheren Hash-Algorithmus.
 - Validierung der Eingabedaten.
 - Erstellung eines neuen Benutzers in der Datenbank.
 - Login:
 - Überprüfung der Benutzerdaten.
 - Vergleichen der Hash-Werten von Passwörter
 - Authentifizierung des Benutzers
 - open File:
 - Abrufen des Dateipfads aus der Datenbank über den.
 - Senden der Datei aus dem Dateisystem des Servers mit.
 - upload File:
 - Lädt eine Datei auf den Server und speichert sie im angegebenen Modul und Thema.
 - Speichern der Datei im Dateisystem des Servers.
 - Speichern der Datei-Metadaten in der Datenbank
 - download File:
 - Abrufen des Dateipfads aus der Datenbank über den.
 - Senden der Datei als Anhang aus dem Dateisystem des Servers
 - create Module :
 - Erstellt ein neues Modul.
 - create Topic:
 - Erstellt ein neues Thema innerhalb eines Moduls.
 - create File:

- Lädt eine neue Datei innerhalb eines Themas hoch.
 - List Modules:
 - Abruf der Module aus der Datenbank.
 - List Topics:
 - Abruf der Themen aus der Datenbank basierend auf dem Modulnamen.
 - List Files:
 - Abruf der Dateien aus der Datenbank basierend auf den Modul- und Themennamen.
 - delete Module:
 - Löschen des Moduls aus der Datenbank.
 - Kaskadierte Löschung der zugehörigen Themen und Dateien.
 - delete Topic:
 - Löschen des Themas aus der Datenbank.
 - Kaskadierte Löschung der zugehörigen Dateien.
 - delete File:
 - Löschen der Datei aus der Datenbank.
 - Entfernen der Datei aus dem Dateisystem des Servers.
- Unterstützung bei der Behebung von Backend-Problemen, die während des Projekts aufgetreten sind.
- Einführung von Postman zur effizienten Verwaltung und zum Testen der API-Endpunkte. Erstellung einer detaillierten Übersicht der implementierten Endpoints in Postman, einschließlich der spezifischen Anfrage-Bodys und Header. Dies dient dazu, die Endpoints durch definierte Tests zu vereinfachen, schnelle Tests zu ermöglichen und die Automatisierung des Testprozesses zu unterstützen.
- Sonstige Aufgaben
 - Dokumentation

8.8. Nisrine El Youncha - Matrikelnummer: 1464975

- Textextraktion und Verarbeitung
 - Entwicklung und Implementierung von Services für die Textextraktion
 - Integration von TextExtractor zur Textextraktion aus PDF-Dokumenten
 - Sicherstellen, dass extrahierter Text als String oder als Fehlermeldung zurückgegeben wird
 - Integration von AIProcessor zur fortgeschrittenen Textverarbeitung
 - Zusammenarbeit mit AIController, um extrahierten Text asynchron zu verarbeiten
 - Empfang und Verarbeitung der Antwort des externen KI-Dienstes
 - Entwicklung der Kommunikation mit einem lokalen KI-Modell zur Textverarbeitung
 - Senden von Anfragen an eine lokale LLM-API und Rückgabe der generierten Textausgabe
- Server- und Microservice-Management

- Konfiguration der offenen Ports zur Kommunikation zwischen den Microservices und ihren APIs
- Migration der gesamten Bereitstellung auf den Server und Verwaltung aller Ports für den externen Zugriff auf die API
- Zusammenfassungsfunktionalität
 - Entwicklung der get_summary-Methode zur Zusammenfassung von Texten
 - Implementierung der KI-Dienst-Integration (extern oder lokal) für die Zusammenfassungsfunktionalität
 - Integration der asynchronen Verarbeitung zur Generierung von Zusammenfassungen
 - Implementierung der Datenbankprüfung in app.py, um zu überprüfen, ob eine Zusammenfassung bereits existiert
 - Entwicklung des /get_summary-Endpunkts in app.py, um POST-Anfragen zur Zusammenfassung von PDF-Texten zu verarbeiten
 - Entwicklung von Funktionen zum Erstellen, Abfragen und Bearbeiten von Zusammenfassungen
 - Implementierung von save_summary und find_summary_by_text zur Speicherung und Suche von Zusammenfassungen in DatabaseContext.py
- Sonstige Aufgaben
 - Dokumentation

8.9. Tim Bornemann - Matrikelnummer: 1431393

- Recherche über Mögliche LLMs zur verwendung im Lernassistenten
- Organisation des Servers über DigitalOcean
 - Server einrichten
 - Grundinstallation von benötigten Diensten vorgenommen.
 - Zwischenzeitliches neu aufsetzen des Servers
 - Neuinstallation und Konfiguration aller Abhängigkeiten.
- Installation von Ollama auf dem Server
 - Ollama LLM installiert und konfiguriert.
 - Testläufe zur Funktionsüberprüfung durchgeführt.
- [Entwicklung der LLM-REST-API](#)
 - Erstellung und Konfiguration des REST-API-Servers
 - API-Endpunkte entwickelt & getestet
 - Implementierung der API-Logik
 - Optimierung & Vereinfachung der API-Verwendung
 - Groq LLM API Integration
 - Ollama Integration & Verwaltung
 - [API Dokumentation](#)

- Für genauere Einsicht in die Entwicklung der LLM Rest API kann in das separate Github repository: <https://github.com/timbornemann/LLM-Request-Rest-API-Modul> geschaut werden.
- Sonstige Aufgaben
 - Dokumentation