

Universität Potsdam
Humanwissenschaftliche Fakultät
Department Linguistik



**Design und Implementierung einer webbasierten
Schnittstelle zur Auswertung von
Freitext-Schülerantworten**

Wissenschaftliche Arbeit zur Erlangung des akademischen Grades
Bachelor of Science in Computerlinguistik

Erstgutachter Prof. Dr. Manfred Stede
Zweitgutachter Robin Schäfer

Vorgelegt am 1. Februar 2024
von **Nhật Trường Thomas Phạm**
Matrikelnummer 797536

Zusammenfassung

Lern-Apps bieten für Schüler*innen eine Plattform zum selbstständigen Lernen und Arbeiten. Nach wie vor sind in diesen Anwendungen Multiple-Choice-Fragen und die Abfrage kurzer Texteingaben weit verbreitet, welche keine komplexe Auswertung erfordern. Das Projekt AKI-LAS verfolgt hingegen das Ziel ein KI-basiertes Assistenzsystem für Lehrkräfte zu entwickeln, das längere Freitextantworten auswerten und individuelles Feedback bereitstellen kann. Diese Arbeit beschäftigt sich mit der Entwicklung einer GUI in Form einer Web-App für eben diesen Zweck. Dafür wird die Geschichte der GUI umrissen sowie relevante Forschungsdisziplinen vorgestellt. Es wird deutlich, dass die Entwicklung einer guten GUI eine Kollaboration von Expert*innen verschiedenster Fachrichtungen erfordert. Die Realisierung der Web-App wird mit besonderem Blick auf Designprinzipien durchgeführt. Damit schließt zwar diese Arbeit ab, die (Weiter-)Entwicklung hat jedoch erst begonnen.

Inhaltsverzeichnis

Abkürzungsverzeichnis	1
1 Einleitung	1
2 Theoretische Grundlagen	2
2.1 Die Geschichte der GUI	2
2.2 Forschungsgebiete	7
2.3 Designprinzipien	9
3 Motivation: AKILAS	11
4 Die Web-App	12
4.1 Funktionsweise	12
4.2 Technischer Aufbau	14
4.2.1 Programmiersprachen, Bibliotheken & Frameworks	14
4.2.2 Infrastruktur	17
4.2.3 Ordnerstruktur	24
4.3 Design	25
5 Ausblick	29
6 Fazit	31
Literaturverzeichnis	32
Eidesstattliche Erklärung	32

Abkürzungsverzeichnis

AKILAS	Adaptiver KI-Lern-Assistent für die Schule
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
CERN	Conseil Européen pour la Recherche Nucléaire
CRF	Conditional Random Field
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HCD	Human-Centered Design
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
MCI	Mensch-Computer-Interaktion
NLP	Natural Language Processing
RST	Rhetorical Structure Theory
SVM	Support Vector Machine
UI	User Interface
UP	Universität Potsdam
URL	Uniform Resource Locator
UX	User Experience
VPN	Virtual Private Network
WSGI	Web Server Gateway Interface
WWW	World Wide Web

1 Einleitung

Unsere Welt ist im ständigen Wandel. Mit jedem technologischen Fortschritt eröffnen sich neue Chancen und Möglichkeiten. Diese müssen jedoch zuerst erforscht werden. Während der Entwicklung neuer Technologien entstehen zahlreiche Prototypen, die sich kontinuierlich an die jeweiligen Bedürfnisse anpassen. Dabei spielt die künstliche Intelligenz (KI) eine zunehmend wichtigere Rolle und hat das Potenzial, unser Bildungssystem durch den Einsatz von KI-Lernassistenten grundlegend zu verändern. Sie gewinnt besonders an Bedeutung, wenn man den Lehrkräftemangel in Deutschland betrachtet. “Die Kultusministerkonferenz prognostiziert, dass bis 2025 rund 25.000 Lehrkräfte fehlen” (Ständige Wissenschaftliche Kommission der Kultusministerkonferenz (SWK), 2023, S. 6), diese Zahlen sollen danach sogar weiter steigen. Dadurch sind nicht nur Lehrkräfte überlastet, auch den Schüler*innen bleiben dringend benötigte individuelle Zuwendung und Förderung verwehrt. Am Ende des Tages sind dann schließlich die Eltern damit überfordert, das Lerndefizit zu Hause zu kompensieren.

Darin liegt das Potenzial von KI-Lernassistenten. Sie können Kinder bei der Weiterentwicklung ihrer Kompetenzen unterstützen und dabei Lehrkräfte und Eltern entlasten. Diesen Ansatz verfolgt auch AKILAS¹ - *Adaptiver KI-Assistent für die Schule*. AKILAS ist ein Forschungsprojekt mit dem Ziel Technologien für einen Lernassistenten für deutschsprachige Schüler*innen zu entwickeln. AKILAS soll Lehrkräfte nicht ersetzen, sondern vielmehr unterstützen, unter anderem bei der Korrektur von Aufgaben, beim Erstellen von personalisiertem Feedback oder beim Eingehen auf individuelle Lernbedürfnisse. Bisher wurden automatisiert meistens Multiple-Choice-Fragen oder kurze Texteingaben verarbeitet, die leicht von einem regelbasierten Programm ausgewertet werden können. AKILAS hingegen verfolgt das Ziel, auch längere Freitext-Antworten mittels KI inhaltlich auszuwerten und hilfreiches Feedback zu geben.

Bevor diese Software jedoch verwendet werden kann, muss die Benutzung einfach und zugänglich gestaltet werden, besonders für Schüler*innen und Lehrkräfte. Dazu eignet sich eine *Graphical User Interface* (GUI), welche eine Schnittstelle zwischen dem User und der eigentlichen Software bietet. Die GUI wird in Form einer Webanwendung, hier auch Web-App genannt, realisiert. Das Design und die Implementierung dieser Web-App ist Gegenstand dieser Arbeit.

Zunächst werden in Kapitel 2 theoretische Grundlagen erläutert, darunter

¹ <https://www.interaktive-technologien.de/projekte/akilas>

die Geschichte der GUI, wobei die Entwicklung in groben Zügen nachverfolgt wird. Es folgen die Beschreibung einiger Forschungsdisziplinen, die bei der Entwicklung einer GUI relevant sind, und schließlich Designprinzipien, die Ergebnis jahrelanger Forschung sind. In Kapitel 3 wird das Projekt AKILAS näher beschrieben und eine Brücke geschaffen zwischen der theoretischen Überlegung und der praktischen Umsetzung. In Kapitel 4 wird die technische Umsetzung der Web-App erklärt. Dabei wird auf die Funktionsweise eingegangen, zudem welche Werkzeuge benutzt wurden, die Infrastruktur erklärt, die benutzten Evaluationsmodelle beschrieben und schließlich die Designentscheidungen begründet. Darauf folgt ein Ausblick in Kapitel 5 und ein Fazit in Kapitel 6.

In dieser Arbeit werden folgende Begriffe als Synonyme voneinander verwendet: User und Benutzer*innen, Schulkinder und Schüler*innen, Evaluationsmodell 1 und Evaluationsmodell für Aufgabe 1 (entsprechend für Aufgabe 2), Web-App und Webanwendung.

Außerdem werden Sternchen verwendet wie zum Beispiel *Expert*innen*, um eine inklusive Sprache zu erzeugen, die allen Geschlechtern gerecht wird. Obwohl dies bei Lehnwörtern des Englischen wie *User* oder *Designer* unterlassen wurde, ist stets die geschlechtsneutrale Bedeutung gemeint.

2 Theoretische Grundlagen

In diesem Kapitel werden theoretische Hintergründe erläutert, die bei dem Design und der Implementierung einer Benutzungsschnittstelle hilfreich sind. In Abschnitt 2.1 wird zuerst die Geschichte der GUIs in groben Zügen nachgezeichnet, wobei der Fokus auf Innovationen in der GUI-Entwicklung liegt. Beschriebene Ereignisse sind unter anderem die Anfänge in den 1930er Jahren, die Entwicklung und Weiterentwicklung der ersten GUIs durch Xerox, Apple und Microsoft, die Entstehung des *World Wide Web* sowie abschließend heutige Herausforderungen. In Abschnitt 2.2 werden verschiedene Forschungsdisziplinen vorgestellt, die sich mit der Entwicklung von GUIs befassen. Danach folgt Abschnitt 2.3, das einige grundlegende Prinzipien erklärt, die relevant sind für das Designen einer GUI.

2.1 Die Geschichte der GUI

Eine *Graphical User Interface* (GUI), zu Deutsch "Grafische Benutzungsoberfläche", repräsentiert eine spezielle Form des *User Interfaces* (UI), die durch

visuelle Elemente wie Icons und Fenster gekennzeichnet ist (*What is a graphical user interface (GUI)?*, 2020). Diese visuellen Elemente dienen dazu, die Interaktion zwischen dem User und dem Computer zu erleichtern. Die Verwendung von GUIs hat die Art und Weise, wie Menschen mit Software und Anwendungen interagieren, grundlegend verändert. Anstatt auf textbasierte Eingaben beschränkt zu sein wie bei der Kommandozeile, ermöglicht die grafische Benutzeroberfläche eine intuitivere und visuell ansprechendere Kommunikation.

Ein Blick in die Geschichte der GUI liefert Einblicke in die Entwicklungen, die dazu geführt haben, wie GUIs heute funktionieren und aussehen. Jedes Merkmal und jede Designentscheidung, die wir in modernen GUIs sehen, spiegeln oft bewusste Überlegungen wider. Sogar die Abwesenheit bestimmter Elemente ist selten dem Zufall geschuldet, sondern eine bewusste Entscheidung.

Vorgeschichte Eine der ersten Inspirationen für GUIs entstammt aus den Ideen von Vannevar Bush, der in den frühen 1930er Jahren anfang, über die hypothetische Maschine *Memex* zu schreiben, noch bevor die ersten digitalen Computer gebaut wurden (Reimer, 2005). *Memex* (*memory extender*) sollte eine Erweiterung zum eigenen Gedächtnis bilden (Bush, Vannevar, 1945) und konnte Wissen organisieren und abrufen. Konkret hatte es Funktionen wie das Speichern von Texten, das gezielte Suchen nach Informationen und das Verbinden von Dokumenten. Durch Letzteres konnten Verweise geschaffen werden, in der Art, wie wir heute Hyperlinks benutzen. Diese Funktionen ließen sich mit Touchscreens, einer Tastatur und weiteren Schaltern bedienen. Obwohl eine solche Maschine nie gebaut wurde, gelten Bushs Visionen als wegweisend (Simpson et al., 1996) und beeinflussten die nachfolgende Forschung in den 1960er Jahren, welche die Grundlagen legte für Mensch-Computer-Interaktion.

Vater der GUI Inspiriert von Bushs Visionen beschäftigte sich Douglas Engelbart am Stanford Research Institute damit, wie ein Computer den menschlichen Intellekt verbessern könnte (Engelbart, 1962). 1968 führte er die *Mother of All Demos* durch, eine Präsentation, in der er das NLS-System (oNLine-System) vorstellte, das eine grafische Benutzungsoberfläche, eine Tastatur und zum ersten Mal eine Maus enthielt (Reimer, 2005). Die Navigation erfolgte über eine Maus mit drei Tasten und einem Mauszeiger auf einem grafischen Bildschirm, mit dem beliebige Pixel ausgewählt werden konnten, im Gegensatz zu bisherigen textorientierten Systemen, bei denen eine Zeile selektiert und in der Kommandozeile manipuliert werden konnte (Preim & Dachsel, 2010, S. 173). Im

Vergleich zu Stiften und Touchscreens hat sich die Benutzung einer Maus in Usertests als am natürlichsten erwiesen, um einen Bildschirmzeiger zu manipulieren (Reimer, 2005). Zudem wurden einzelne Anwendungen in verschiedenen Bildschirmfenstern angeordnet, jedoch noch ohne offensichtliche Fensterrahmen und Titelleisten (Reimer, 2005). Weitere Funktionen waren Hyperlinks, wie sie Bush bereits für die hypothetische Maschine *Memex* beschrieb, sowie E-Mails, Videokonferenzen und eine ausgereifte Textverarbeitung. Alles in allem wurden viele Funktionen und Konzepte eingeführt, die Grundlage für die Entwicklung der GUI wurden. Ein Beispiel ist die Idee, dass Computer über eine visuelle, symbolbasierte Interaktion gesteuert werden können sollten. Douglas Engelbart wird daher von Reimer (2005) als Vater der GUI bezeichnet.

Xerox Alto Engelbarts Demonstration im Jahre 1968 überwältigte viele Menschen und motivierte Unternehmen wie Xerox Corporation die Arbeit an GUIs fortzusetzen. 1970 wurde das Palo Alto Research Center, oder auch PARC, gegründet und 1973 der Xerox Alto, einer der ersten Computer mit einer funktionsfähigen GUI, vorgestellt (Reimer, 2005). Elemente, die in *Mother of All Demos* vorgestellt wurden, wurden ebenfalls integriert wie beispielsweise Fenster, Maus, Tastatur, und zusätzlich dazu Icons. Icons sind Piktogramme oder Ideografien, können also physische Objekte, Aktionen oder Konzepte repräsentieren und ermöglichen eine direkte Interaktion mit einer Anwendung (Harrison et al., 2011), was eine Alternative zu Texteingaben in der Kommandozeile bietet. Außerdem nahm der Mauszeiger zum ersten Mal die Form an, die wir noch heute kennen (Reimer, 2005). 1974 entwickelten PARC und ARPA (Advanced Research Projects Agency) *Smalltalk*, das gleichzeitig der Name der Programmiersprache und ihrer Entwicklungsumgebung war (Kay, 1993). *Smalltalk* führte Titelleisten und Abgrenzungen für Fenster ein, die sich nun überlappen konnten. Außerdem gab es noch Icons, Popup-Menüs, Scrollleisten und Radiobuttons.

Apple & Microsoft Im Jahr 1983 veröffentlichte Apple den Lisa Computer, der viele bisherige Konzepte aufgriff und verfeinerte (Reimer, 2005). Dies beinhaltete ein Icon-basiertes Interface, das Anwendungen und Dokumente als Icons darstellte und die erste Pull-Down-Menüleiste, bei der alle Optionen oben angezeigt wurden. Innovativ waren vor allem Keyboard Shortcuts, Häkchen, die neben ausgewählten Elementen angezeigt wurden, das ausgegraute Erscheinungsbild von nicht auswählbaren Elementen und das Konzept eines Papierkorbs für zu löschende Elemente. Ebenso war es nun möglich per Drag-and-Drop

Dateien zu manipulieren, das heißt, per Mausklick mehrere Dokumente gleichzeitig auszuwählen und gemeinsam zu verschieben. Zudem wurde Engelbarts dreitastige Maus zu einer eintastigen Maus modernisiert. Da für Icons mindestens zwei Aktionen notwendig sind, nämlich auswählen und ausführen, wurde außerdem zum Ausführen von Anwendungen das Prinzip des Doppelklicks eingeführt. Dies ist bis heute Standard. Trotz der fortgeschrittenen Technologie war der Lisa Computer aber nicht kommerziell erfolgreich aufgrund des hohen Preises und der hohen Hardwareanforderungen.

Aus diesem Grund entwickelte Apple den preislich erschwinglicheren Macintosh, der 1984 für ein breiteres Publikum zugänglich war (Preim & Dachzelt, 2010, S. 184). Dieser integrierte viele Interaktionskonzepte von Apple's Lisa Computer und präsentierte ebenso ein Desktop-Paradigma mit Icons, Fenstern und Mauszeiger.

Microsoft reagierte 1985 mit Windows 1.0 (Reimer, 2005), konnte aber erst 1990 mit Windows 3.0 seinen bis dahin größten kommerziellen Erfolg verbuchen (Edwards, 2020). Es beinhaltete zwar nicht alle Funktionen von Apples Macintosh, bot aber dafür visuell ansprechendere Icons (Reimer, 2005). Mit Windows 95 konnte Microsoft seine Spitzenposition im Verkauf von GUIs weiter stärken und führte damit zum ersten Mal eine Taskleiste sowie ein Startmenü ein.

WWW Es darf nicht vergessen werden, dass es zu dieser Zeit weitaus mehr Unternehmen gab, die ihre Computer mit immer fortschrittlicheren GUIs auf den Markt brachten wie zum Beispiel NeXT, Sun Microsystems oder IBM (Reimer, 2005). Diese beeinflussten sich gegenseitig und trieben die Entwicklung der GUI maßgeblich voran. Ein wichtiger Meilenstein in der Geschichte der Benutzungsschnittstelle ist die Entwicklung des *World Wide Web* (WWW), das den Fokus auf webbasierte GUIs verlagerte. Das WWW wurde 1989 unter der Leitung von Tim Berners-Lee am *Conseil Européen pour la Recherche Nucléaire* (CERN) in Genf entwickelt und benutzt die Infrastruktur und Dienste des Internets (Preim & Dachzelt, 2010, S. 191). Die Umsetzung der Idee von Hypertext, bei der Dokumente über Hyperlinks miteinander verbunden sind, war ein integraler Bestandteil dieser Entwicklung. Die *Hypertext Markup Language* (HTML) wurde eingeführt, um die Struktur von Webdokumenten zu beschreiben und die Integration von Hyperlinks zu ermöglichen. Gleichzeitig sind das HTTP-Protokoll, der erste Webbrowser, die erste Suchmaschine und ein Webserver entstanden. Beachtenswert ist, dass keines dieser Elemente patentiert wurde, was sich als entscheidende Voraussetzung für die weite Verbreitung des

WWW erwies (Preim & Dachsel, 2010, S. 193). 1992 wurde der NSCA Mosaic entwickelt, der als erster Webbrowser mit einer GUI gilt, die Bilder und Texte anzeigen konnte. Trotz der Fortschritte waren Webseiten immer noch primär textbasiert und statisch. Es fehlte HTML an Interaktivität und visueller Gestaltung. Erst mit der Entwicklung von JavaScript in 1995 erhielten Webseiten die Fähigkeit zur dynamischen Aktualisierung, konnten also direkt auf Eingaben des Users reagieren, ohne sich neu zu laden. Außerdem konnten Webseiten nun auch visuell ansprechende Animationen anzeigen und interaktiver gestaltet werden (Keith & Sambells, 2010, S. 1). 1996 wurde die erste Version von CSS (*Cascading Style Sheets*) vorgestellt, welches die Trennung von Inhalt und Präsentation ermöglicht. Nun konnte die Grundstruktur von Webseiten durch HTML definiert und der Stil durch CSS festgelegt werden, während JavaScript die dynamische Funktionalität bot.

Heutige Entwicklungen Mit dem Aufkommen von Smartphones und Tablets in den 2000er Jahren veränderte sich die Art der Interaktion mit Computern. Apple veröffentlichte 2007 sein erstes iPhone und popularisierte damit Touchscreens, die ein Umdenken der bisherigen Interaktionen erforderten (Nuhel, 2021). Die fortschreitende Entwicklung der Hardware ermöglichte auch die Erweiterung auf andere Modalitäten der Interaktion. Die Einbindung der Kamera an den Computer ermöglichte beispielsweise Gestensteuerung, während Mikrofone Sprachbefehle erlaubten. Dies erweitert dementsprechend auch Bedürfnisse in der Verwendung von GUIs, sodass sie nicht mehr nur mit grafischen Elementen arbeiten, sondern GUIs auch in Verbindung mit anderen UIs bedacht werden müssen wie zum Beispiel VUIs (Voice User Interfaces). Eine weitere wichtige Entwicklung ist die Integration von Künstlicher Intelligenz (KI) in UIs. Die Forschung an KI existiert zwar schon seit den 1950er Jahren, deren Nutzung stieg aber erst mit der Verbesserung der Performance in den frühen 2000er (Roser, 2022). Ende 2022 veröffentlichte OpenAI das Konversationsmodell ChatGPT, das so erfolgreich war, dass es nach zwei Monaten bereits über 100 Millionen aktive Benutzer*innen gab (Wu et al., 2023). Auch die Integration von KI in verschiedensten Anwendungen veränderte die Interaktion von Mensch und Computer (Xu et al., 2021).

Mit dem Anstieg der Nutzung mobiler Geräte wurde auch das Konzept des responsiven Designs wichtiger. Dieses fordert eine Anpassung und Optimierung der GUI an verschiedene Bildschirmgrößen, also auch für kleinere Bildschirme. Aktuelle Standards erfordern, dass GUIs plattformübergreifend und intuitiv

funktionieren.

In der heutigen Zeit stehen GUI-Designer vor der Herausforderung, nicht nur verschiedene Bildschirmgrößen zu berücksichtigen, sondern auch die vielfältigen Arten der Interaktion, sei es Touch, Gesten, Sprache oder sogar Augenbewegungen abzudecken. Die GUI-Entwicklung ist zu einem multidisziplinären Bereich geworden, der Kenntnisse in Psychologie, Technologie und Design erfordert.

Zusammenfassend lässt sich sagen, dass die Entwicklung der GUI von verschiedensten Menschen, Unternehmen und Forschungseinrichtungen geprägt war. Die Erfindung der modernen GUI lässt sich also nicht eindeutig einem Individuum zuschreiben, sondern ist vielmehr das Ergebnis einer kollektiven Anstrengung, während die Bedürfnisse und Anforderungen stets an die technologischen Fortschritte angepasst werden mussten.

2.2 Forschungsgebiete

Die Entwicklung einer benutzungsfreundlichen und visuell ansprechenden GUI ist ein schwierigeres Unterfangen, als man es zuerst vermuten mag. (Antcheva et al., 2005, S. 613). Es ist ein komplexer Prozess, der Kenntnisse aus vielen verschiedenen Fachbereichen erfordert. Einerseits sind Humanwissenschaften wie Verhaltens-, Kognitions-, Neuro-, Sozialwissenschaften und Philosophie von Bedeutung (Norman, 2013, S. 45), da es Menschen sind, die mit diesen Benutzungsoberflächen interagieren. Weil in Verbindung mit der Erforschung von menschlichen Bedürfnissen auch Experimente und Tests mit Usern durchgeführt werden, kann es hilfreich sein, für die Auswertung Statistiker*innen hinzuzuziehen (Preim & Dachsel, 2010, S. 3). Andererseits muss die GUI technisch umgesetzt werden, was die Kenntnis entsprechender Frameworks und Fähigkeiten im Programmieren erfordert. Hinzu kommt, dass die Benutzungsoberfläche durch die korrekte Wahl von Layout, Schriftart und Farbpalette für alle zugänglich und ansprechend gestaltet werden muss. Dies macht das Design einer GUI zu einer multidisziplinären Aufgabe, die eine Kollaboration von Expert*innen erfordert.

Designansätze Das Forschungsfeld, das Prinzipien aus diesen verschiedenen Fachbereichen vereint, wird Mensch-Computer-Interaktion (MCI) genannt (Valverde, 2011, S. 7). MCI beinhaltet die Erforschung, die Planung und das Design der Interaktion zwischen Mensch und Computer, sodass menschliche Bedürfnisse optimal erfüllt werden (Galitz, 2002, S. 4). Die Entwicklung einer erfolgreichen

Interaktion zwischen Mensch und Computer ist also ein komplexer Prozess.

User Interface Design (UID) ist hierbei ein Teilgebiet der MCI konzentriert sich auf das “visuelle Design, z.B. Layoutaspekte und Konzepte für die Nutzung von Farben, Formen und Fonts” (Preim & Dachzelt, 2010, S. 3). Prinzipien des UID sind besonders hilfreich bei der Entwicklung einer GUI, die die Bedürfnisse des Users erfüllen.

Dass menschliche Anforderungen jedoch so berücksichtigt wurden, war nicht immer der Fall. Als in den 1980er Jahren PCs entwickelt wurden, wurde ein technologiezentrierter Ansatz verfolgt (Xu et al., 2021). Im Gegensatz dazu steht eine userzentrierte Vorgehensweise, bei der “unter anderem die gesamte Terminologie an die ‘Sprache des Benutzers’ angepasst” (Preim & Dachzelt, 2010, S. 15) wird, sodass nur exakt die dem User bekannten und vertrauten Begriffe verwendet und Fachwörter konsequent vermieden werden. Verwandt mit der userzentrierten Gestaltung ist das weiterentwickelte Konzept des *human-centered design* (HCD) beziehungsweise der menschenzentrierten Gestaltung laut Norman & Draper (1986) (zitiert nach Preim & Dachzelt, 2010, S. 15). Das menschenzentrierte Design “bezieht neben den tatsächlichen Benutzern auch andere Menschen ein, die von der Einführung dieser Software mittelbar betroffen sind. Wenn Lernsoftware für Kinder entwickelt wird, werden neben den Kindern der entsprechenden Altersklasse auch Eltern und Pädagogen einbezogen. Diese Personengruppen können einerseits zur Entwicklung beitragen, andererseits sind sie für die Akzeptanz entscheidend.” (Preim & Dachzelt, 2010, S. 15). Es ist eine Philosophie, die ein umfassendes Verständnis voraussetzt, einerseits über Benutzer*innen und ihr Umfeld und andererseits über die Anforderungen, die es zu erfüllen gilt (Norman, 2013, S. 8). Dieses Verständnis kann durch Observation erzeugt werden, da betroffenen Menschen oft selbst nicht bewusst ist, was ihre eigentlichen Bedürfnisse sind. Norman weist darauf hin, dass es Menschen sogar oft nicht bewusst ist, wenn sie bei einer Interaktion auf Schwierigkeiten stoßen.

Eine wichtiger Aspekt des HCD ist die *User Experience* (UX). Sie bezeichnet die Erfahrung beziehungsweise “inwiefern interaktive Produkte ein bestimmtes Erlebnis vermitteln, geprägt durch Emotionen und Gefühle” (Preim & Dachzelt, 2010, S. 228). Von einer Anwendung wird also “eine attraktive Gestaltung und eine angenehme Benutzung erwartet” (Preim & Dachzelt, 2010, S. 228). Man könnte einwenden, dass Menschen verschiedene Präferenzen haben und positive Erfahrungen subjektiv sind. “Dies ist jedoch nicht der Fall, wie viele Untersuchungen auch bei interaktiven Produkten zeigen [Thielsch und Hassenzahl,

2008]. Menschen haben eine sehr ähnliche (visuelle) Wahrnehmung und auch die ablaufenden kognitiven und emotionalen Prozesse sind ähnlich.” (Preim & Dachzelt, 2010, S. 229). Dies macht Erkenntnisse aus diesem Forschungsfeld sogar noch praktischer bei der Entwicklung einer GUI.

Eine andere Perspektive bietet *Usability Engineering*. “Der Begriff Usability Engineering charakterisiert ein systematisches ingenieurmäßiges Vorgehen zur Entwicklung gut benutzbarer Computersysteme” (Preim & Dachzelt, 2010, S. 19). Dabei sind die Prinzipien der anderen Disziplinen nicht unwichtig. Der Fokus ist lediglich auf ein anderes Ziel gerichtet, nämlich die Entwicklung einer Anwendung, die eine fehlerfreie Benutzung ermöglicht (Preim & Dachzelt, 2010, S. 228).

Zusammenfassend lässt sich sagen, dass es viele Forschungsfelder gibt, die sich mit der Entwicklung einer (guten) GUI beschäftigen. Sie sind alle miteinander verwandt und teilen sich Konzepte und Prinzipien, nach denen GUIs gestaltet werden. Die verschiedenen Forschungsfelder, darunter MCI, UID, HCD, UX, Usability Engineering und noch weitere, verfolgen zwar teilweise verschiedene Ziele und haben einen anderen Fokus, jedoch ergänzen sie sich in ihrem Bestreben, Benutzungsoberflächen benutzungsfreundlich, effektiv und ansprechend zu gestalten. Einige Konzepte und Prinzipien dieser Disziplinen, die für die Entwicklung einer GUI relevant sind, werden im folgenden Abschnitt vorgestellt.

2.3 Designprinzipien

Es gibt verschiedene Prinzipien, die bei dem Design einer benutzungsfreundlichen Anwendung in Betracht gezogen werden. Die folgende Auflistung erklärt einige dieser relevanten Prinzipien. Sie ist jedoch nicht vollständig, da eine umfassende Darstellung aller Aspekte den Rahmen dieser Arbeit überschreiten würde.

Menschliche Fehler vs. Designfehler Irren ist menschlich. Die Bedienung von GUIs ist nicht davon ausgenommen. Menschen machen oft Fehler, zum Beispiel aus Unwissenheit oder aus Unaufmerksamkeit. Damit umzugehen ist eine Aufgabe des Designers (Norman, 2013, S. 67). Laut Norman ist eine misslungene Interaktion, also beispielsweise eine Interaktion, die den User nicht an sein Ziel bringt, nicht die Schuld oder der Fehler des Users, sondern vielmehr des Designs. Er schlägt vor, dass bei der Entwicklung von Systemen diese Fehler bereits mitgedacht werden sollen, sodass darauf angemessen reagiert werden kann. Im

Kontext von GUIs kann das bedeuten, dass mit allen möglichen Eingaben des Users gerechnet werden muss.

Feedback Feedbacks sind ein wichtiges Konzept bei Interaktionen, ob mit Menschen oder mit Maschinen. Ob eine Aktion gelungen ist, kann anhand der Reaktion festgestellt werden. Das ist bei der Kommunikation von Menschen nicht anders. Gegeben sei zum Beispiel ein Gespräch zwischen zwei Personen. Beide möchten sich Neuigkeiten mitteilen. Eine erfolgreiche Interaktion wäre also, wenn eine Neuigkeit erzählt wird und die andere Person die Informationen auch empfängt, also zuhört. Anhand des Feedbacks, also beispielsweise passenden Nachfragen, kann die erzählende Person vermuten, dass die Informationen richtig übermittelt wurden. GUIs sollten ebenso mit Usern kommunizieren, so dass das Verhalten des Systems verständlich wird (Norman, 2013, S. 67), beispielsweise mit Fehlermeldungen oder Bestätigungen, dass eine Aktion erfolgreich war.

Feedforward Die Kommunikation mit Anwendungen beschränkt sich nicht nur auf Reaktionen. Systeme können bereits vor der eigentlichen Aktion darauf hinweisen, welche Elemente auf welche Art benutzbar sind (Norman, 2013, S. 19). Das ist die Rolle von *Feedforward* (Norman, 2013, S. 72). Dies kann zum Beispiel explizit geschehen mit einem Hinweistext oder auch implizit mit einem Einkaufswagensymbol auf einem Button, welcher auf einen Kaufvorgang hindeutet. Letzteres sollte so “gestaltet sein, dass es naheliegt, ihn zu betätigen” (Preim & Dachzelt, 2010, S. 138). Ein gutes Design macht sich also gut verständlich. Diese Verständigung ist allerdings nicht universell und muss je nach Kultur und Sprache angepasst werden. Beispielsweise unterscheidet sich die Bedeutung von Farben und Symbolen in verschiedenen Kulturen (Preim & Dachzelt, 2010, S. 16).

Konvention Eng verknüpft mit dem Konzept des *Feedforward* sind Konventionen. Konventionen helfen Usern vorherzusehen, welche Elemente welche Wirkung haben (Nielsen, 2004). Diese Transparenz ist wichtig, da sonst eine mühsame Einarbeitung in jede einzelne Anwendung notwendig wäre. Dadurch, dass sich aber gewisse Designelemente in vielen Anwendungen wiederfinden, erwarten User das gleiche Verhalten. Beispielsweise deutet ein stilisiertes Fragezeichen oben rechts auf Webseiten oft eine Hilfemöglichkeit für User an (Preim & Dachzelt, 2010, S. 314).

Dazu passend ist Jakobs Gesetz der Internet-UX. Es besagt, dass User die meiste Zeit auf anderen Webseiten verbringen, was bedeutet, dass sie es präferieren, dass diese Webseite genau so funktioniert wie die, die sie bereits kennen. Das Design sollte sich also anpassen an die gewohnten Muster (Nielsen, 2017). Konventionen sind also wichtig, um eine effiziente Bedienung zu ermöglichen (Preim & Dachzelt, 2010, S. 147). Man sollte dabei allerdings beachten, dass Konventionen stets relativ sind und oft kulturell geprägt.

Konsistenz Ähnlich wie eine Konvention hilft auch Konsistenz dabei, eine Interaktion für den User transparent zu machen (Valverde, 2011, S. 58). Bei der Interaktion mit einer GUI lernen User, wie sie funktioniert. Diese Erfahrung hilft ihnen sich auf der Benutzungsoberfläche weiterzubewegen. Es ist davon abzuraten, Ausnahmen hinzuzufügen, da sonst weiteres Lernen erforderlich ist, da diese Ausnahmen gemerkt werden müssen (Galitz, 2002, S. 114). Dies lenkt den Fokus von der eigentlichen Aufgabe ab. Die gleichen Dinge sollten gleich funktionieren (Antcheva et al., 2005, S. 613). Das bedeutet, dass innerhalb einer Web-Anwendung der Mausklick auf einen Link immer das gleiche bewirken sollte, also entweder die Erstellung von Pop-ups oder die direkte Umleitung auf die Webseite, jedoch nicht beides.

3 Motivation: AKILAS

Die gegebene Web-App wird im Rahmen des Forschungsprojekts *AKILAS – Adaptiver KI-Lern-Assistent für die Schule* erstellt. Das Projekt AKILAS verfolgt das Ziel einen KI-basierten Lernassistenten für deutschsprachige Schulkinder zu entwickeln². Dieser soll individuell auf die Bedürfnisse aller Schulkinder eingehen. Gleichzeitig sollen aber auch Lehrkräfte dabei entlastet werden, einerseits bei der Auswertung von Übungsaufgaben und andererseits bei der Auswahl von individuellem Lernmaterial. AKILAS soll also Lehrkräfte nicht ersetzen sondern unterstützen.

Es sind insgesamt drei Forschungsgruppen beteiligt. Eine Gruppe von der Universität Magdeburg arbeitet an der Generierung von Lerntypen durch datenschutzsensibles Machine Learning³. Dann gibt es noch zwei Forschungsgruppen von der Universität Potsdam, wovon sich eine auf die empirische Evaluation von Intelligenten Tutoring-Systemen fokussiert. Die andere Gruppe hat ihren

² <https://www.interaktive-technologien.de/projekte/akilas>

³ <http://angcl.ling.uni-potsdam.de/projects/akilas.html>

Schwerpunkt in der Evaluation von Freitextantworten von Schulkindern und in der Generierung von individualisiertem Feedback. Die gegebene Web-App beziehungsweise Lernplattform wird im Zusammenhang mit der zuletzt genannten Forschungsgruppe erstellt.

Inhaltszonen & Attributionen

Die Web-App implementiert zwei Aufgaben für Schüler*innen. Für Aufgabe 2 soll eine Erörterung geschrieben werden. Dabei werden die Konzepte von Inhaltszonen und Attributionen relevant, da bei der Evaluation auf eben diese Bezug genommen wird. Inhaltszonen geben die Funktion des Satzes beziehungsweise Satzsegments für die Argumentation an. Zum Beispiel gibt es die Einleitung, die in ein Thema einführt, die eigene Meinung, oder die Pro- und Contra-Argumente aus einem gegebenen Text. Im Gegensatz zu Inhaltszonen werden Attributionen von Meinungen innerhalb der Sätze ermittelt. Es wird die Meinung selbst bestimmt, ihre Quelle und zusätzlich noch die Formulierungen, die diese Attribution anzeigen.

4 Die Web-App

Da die GUI in Zukunft potenziell von vielen Schüler*innen benutzt wird, eignet sich die Umsetzung als Webanwendung beziehungsweise Web-App, da sie dann unkompliziert über das Internet im Webbrowser verfügbar ist und somit weder eine Installation noch ein spezifisches Betriebssystem notwendig ist (*Was ist eine Web-App? Definition und Web-App-Beispiele*, 2021). Da die Web-App wie im Kapitel 3 beschrieben für deutschsprachige Schulkinder konzipiert wird, ist die Sprache der Anwendung dementsprechend deutsch.

4.1 Funktionsweise

In diesem Abschnitt wird erläutert, wie ein User die Web-App benutzen kann. Dafür sind zwei Aufgaben implementiert. Die Bearbeitung dieser Aufgaben wird durch die Web-App ermöglicht und im Folgenden erläutert. Es wird gegebenenfalls zusätzlich technischer Hintergrund geliefert.

Themaauswahl Die Auswahl des Themas erfolgt über Radiobuttons. Alternativ kann der Link daneben angeklickt werden, der ein Popup-Fenster mit dem

Artikel öffnet und gleichzeitig den entsprechenden Radiobutton auswählt. Der Artikel sollte für die nächsten Aufgaben gelesen werden.

Aufgabe 1: Textverständnis Die erste Aufgabe bezieht sich auf das Textverständnis. Es gibt insgesamt 4 Fragen, wovon die zweite Frage nicht beantwortet werden kann für das Thema *Schulbeginn*, da die Antwort nicht im Text gegeben ist (siehe Abbildung 8). Für jede Frage existiert ein eigenes Texteingabefeld, in das der User Antworten schreiben kann. Eine Antwort zu einer Frage kann dabei aus mehreren stichpunktartigen Sätzen bestehen. Diese müssen dann in verschiedene Zeilen geschrieben werden. Beispieleingaben werden in grauer Farbe in den verschiedenen Texteingabefeldern angezeigt, die jedoch verschwinden, sobald im jeweiligen Feld eine Eingabe getätigt wurde. Dies wird ermöglicht durch im HTML-Dokument eingebetteten JavaScript-Code.

Bei der Evaluation werden die eingegebenen Antworten mit Referenzantworten verglichen und anhand dessen Feedback generiert. Der Feedbacktext wird unter den jeweiligen Antworten angezeigt.

Aufgabe 2: Argumentation Für die zweite Aufgabe wird eine Erörterung geschrieben zum ausgewählten Thema. Ähnlich wie in Aufgabe 1 gibt es hier ein Texteingabefeld, dessen Größe ebenfalls vertikal verändert werden kann. Hier wird der Text eingetippt oder einkopiert. Alternativ kann sich der User auch dazu entscheiden, eine txt-Datei hochzuladen, die den Essay enthält. Darüber hinaus gibt es die Option die Analyse zusätzlich mit einer Diskurssegmentierung durchzuführen. Die Auswahl erfolgt über Radiobuttons. Standardmäßig ist die Analyse ohne Textsegmentierung ausgewählt, da die Analyse mit Textsegmentierung einige Minuten beanspruchen kann je nach Länge des Textes. In diesem Fall wird der in Sätzen segmentierte⁴ Text an das Evaluationsmodell übergeben wird. Der Diskurssegmentierer würde den Text in eventuell kleinere Einheiten zerteilen und dann erst an das Evaluationsmodell übergeben, was eine sensitivere Auswertung ermöglicht.

Nach der Evaluation wird die Erörterung angezeigt, wobei verschiedene Inhaltszonen und Attributionen unterschiedlich visualisiert werden. Inhaltszonen können farblich voneinander unterschieden werden und Attributionen werden mithilfe von unterschiedlichen Textformatierungen (z. B. fett, unterstrichen oder kursiv) markiert. Eine Legende wird vor der Erörterung angezeigt, um diese

⁴ Die Segmentierung in Sätzen wird von NLTK übernommen:
<https://www.nltk.org/api/nltk.tokenize.html>

Zuordnung zu erklären. Da die Darstellung der Erörterung abhängig ist vom Userinput, muss der Inhalt dynamisch generiert werden. Das wird ermöglicht durch Jinja2, eine in Flask eingebundene Template-Engine. Damit lässt sich das Styling dynamisch erzeugen, während es in der CSS-Datei definiert wird.

Außerdem wird ein Satz pro Zeile angezeigt, wobei die jeweilige Zeilennummer links vom Satz angegeben wird. Dies ist praktisch, da unter der Erörterung das Feedback steht, welches sich auf Satznummern bezieht.

1 Heutzutage benutzen wir ständig soziale Netzwerke .
2 Für die Jugendliche spielen sie eine große Rolle und LehrerInnen können das benutzt .
3 Das Thema meiner Erörterung lautet : Twitter-Unterricht ist es hilfreich ?
4 Zuerst wollte ich Vor- und Nachteile nennen und dann meine Meinung ausdrücken .

5 Befürworter sagen , dass die Kinder ist sehr gute Möglichkeit für die Schüler .
6 Sie können Kommentare schreiben und nicht die eigene Meinung laut sagen .
7 Viele Schüler haben Angst davor , weil sie schämen sich .
8 Sie können das nur tippen .
9 Ein weiteres Argument ist , dass die Schüler mehr in Erinnerung behalten können .
10 Das ist etwas Interessanter .
11 Es ist schwer , 8 Stunden pro Tag auf einen Lehrer zu hören .
12 Später gibt es auch eine Möglichkeit , diese Kommentare zu Hause noch einmal lesen .
Feedback: Vielleicht kannst du noch etwas auf die Gegenargumente eingehen, die im Artikel stehen? Bitte mache in deinem Text immer klar, ob du ein Argument aus dem Artikel wiedergibst oder ob du deine eigene Meinung äußerst. Die Passage bei Satz 1 scheint deine eigene Meinung wiederzugeben. Das kannst du stärker kenntlich machen, z.B. durch Phrasen wie 'Meiner Ansicht nach ...' Bei Satz 2 sehe ich Sachverhalte, die aus dem Artikel stammen. Diese sollten besser als solche gekennzeichnet werden, z.B. durch indirekte Rede. Von Satz 5 bis Satz 7 sehe ich Sachverhalte, die aus dem Artikel stammen. Diese sollten besser als solche gekennzeichnet werden, z.B. durch indirekte Rede. Die Passage bei Satz 8 scheint deine eigene Meinung wiederzugeben. Das kannst du stärker kenntlich machen, z.B. durch Phrasen wie 'Meiner Ansicht nach ...' Bei Satz 9 sehe ich Sachverhalte, die aus dem Artikel stammen. Diese sollten besser als solche gekennzeichnet werden, z.B. durch indirekte Rede. Die Passage zwischen Satz 10 und Satz 12 scheint deine eigene Meinung wiederzugeben. Das kannst du stärker kenntlich machen, z.B. durch Phrasen wie 'Meiner Ansicht nach ...'

Abbildung 1: Anzeige des Feedbacks für Aufgabe 2

Evaluation Die Evaluation kann für beide Aufgaben gleichzeitig erfolgen, es kann aber auch nur die erste Aufgabe bearbeitet und ausgewertet werden. Das Feedback wird generiert, indem der Button *FEEDBACK GENERIEREN* angeklickt wird. Das Feedback wird dann auf einer neuen HTML-Seite mit dem relativen Pfad */feedback* angezeigt. Dabei bleiben die Auswahl der Radiobuttons und die eingegebenen Texte erhalten, sodass eine Veränderung beziehungsweise Verbesserung der Antworten und erneute Evaluation möglich ist. Der technische Hintergrund zur Evaluation wird in Abschnitt 4.2.2 beschrieben.

4.2 Technischer Aufbau

4.2.1 Programmiersprachen, Bibliotheken & Frameworks

Python 3 Python 3 wurde aus verschiedenen Gründen als Programmiersprache ausgewählt um die Web-App zu entwickeln. Einerseits bietet sie umfassende

Bibliotheken und Frameworks, die für die Webentwicklung hilfreich sind, darunter auch Flask. Andererseits ist diese Programmiersprache Teil des Lehrplans an der humanwissenschaftlichen Fakultät der Universität Potsdam. Angehörige eben dieser Fakultät werden in Zukunft voraussichtlich mit der Wartung und Weiterentwicklung der Web-App beauftragt sein. Dabei profitieren sie von ihrer Familiarität mit Python 3.

HTML, CSS & JavaScript HTML, CSS und JavaScript sind clientseitige Technologien, die Grundlage für viele Webentwicklungen sind. Sie ergänzen sich, um ansprechende, funktionale und interaktive Webseiten zu schaffen (Dean, 2018, S. xi).

HTML (*HyperText Markup Language*) wird genutzt, um die Grundstruktur des Inhalts auf der Webseite zu definieren. Damit wird festgelegt, welche Elemente, zum Beispiel Texte und Bilder, existieren und in welchem Verhältnis sie zu einander stehen. Webbrowser interpretieren HTML-Dateien um Webseiten anzuzeigen.

CSS (*Cascading Style Sheets*) hingegen definiert das Erscheinungsbild von HTML-Elementen. Dadurch lassen sich unter anderem Eigenschaften wie Farbe, Schriftart, Größe, Layout, Abstand oder Position festlegen. Die Definition von Stilvorlagen führt zudem auch zu einem einheitlichen und konsistenten Design. Die Kombination von HTML und CSS ermöglicht eine Trennung von Inhalt (HTML) und Design (CSS) (Saputra & Azizah, 2013, S. 906) und erleichtert somit die Wartung und Wiederverwendbarkeit von Code.

JavaScript ist eine Skriptsprache, also eine Programmiersprache, die über einen Interpreter ausgeführt wird. Sie wird in der Webentwicklung verwendet, um Webseiten interaktiver zu gestalten, indem HTML-Dateien mit dynamischer Funktionalität ausgestattet werden (Dean, 2018, S. 313). Das bedeutet, dass mehr als statische Informationen angezeigt werden können. Zum Beispiel kann JavaScript auf Ereignisse wie Mausklicks der User reagieren und dynamisch den Inhalt einer Webseite verändern, ohne die gesamte Seite neu zu laden.

Flask Flask ist ein *lightweight* WSGI-Framework (*Web Server Gateway Interface*) für Web-Anwendungen und zeichnet sich durch seine schnelle Einarbeitung, Flexibilität und einfache Erweiterbarkeit aus⁵. Ein *lightweight* Framework ist hierbei ein Programmskelett mit minimal ausgestatteter Funktionalität, die

⁵ <https://pypi.org/project/Flask/>

leicht auf den Zweck der spezifischen Anwendung angepasst und erweitert werden kann (Vojislav et al., 2011). Für eine schnelle und effiziente Entwicklung der Web-App wurde deshalb Flask ausgewählt.

WSGI gilt als Kommunikationsstandard zwischen Webservern und Webanwendungen beziehungsweise Webframeworks für die Programmiersprache Python⁶. Ein WSGI-Framework implementiert diese Schnittstelle und erleichtert damit die Interaktion zwischen dem Webserver und der Webanwendung unter Einhaltung des WSGI-Standards. Es definiert beispielsweise, wie Anfragen vom Webserver an die Webanwendung weitergeleitet werden und wie darauf reagiert wird. Durch den WSGI-Standard kann Portabilität und Interoperabilität zwischen verschiedenen Komponenten gewährleistet werden, was bedeutet, dass Webanwendungen, die ein WSGI-Framework benutzen, auf vielen Webservern und Plattformen laufen können.

Obwohl Flask sich durch seine Einfachheit und einen schnellen Einstieg auszeichnet, ist es leistungsfähig und nützlich durch die folgenden Funktionen. Zum einen sind Template-Engines integriert, also Software, die in der Webentwicklung verwendet werden, um dynamische Webseiten zu erstellen. Flask verwendet standardmäßig Jinja2, das Kontrollstrukturen wie Schleifen oder Bedingungen in HTML-Dateien zulässt. Dies macht die Generierung von dynamischen HTML-Inhalten möglich. Das ist besonders hilfreich, da die Web-App eine interaktive Benutzeroberfläche bieten und User Eingaben je nach Bedingung unterschiedlich visualisiert werden soll. Zum anderen erleichtert Flask die Verwaltung von HTTP-Anfragen. Dies geschieht beispielsweise durch das Festlegen von URL-Routen, um verschiedene Endpunkte der Webanwendung zu definieren. Zusätzlich wird auch das Erstellen und Weiterverarbeiten von Formularen durch Flask erleichtert. Flask unterstützt somit die Entwicklung von interaktiven Benutzeroberflächen. Zu guter Letzt, verfügt Flask über eine umfassende und gut strukturierte Dokumentation⁷, die notwendig ist für die Nutzung und Entwicklung.

requests Requests ist eine Python-Bibliothek und bietet Funktionen rundum die Verwaltung von HTTP-Anfragen und HTTP-Antworten⁸. Sie unterstützt verschiedene HTTP-Anfragemethoden wie beispielsweise GET oder POST, die unterschiedliche Aktionen auf Webservern ermöglichen. Für die Entwicklung

⁶ <https://peps.python.org/pep-3333/>

⁷ <https://flask.palletsprojects.com/en/3.0.x/>

⁸ <https://requests.readthedocs.io/en/latest/>

dieser Web-App wurden Usereingaben im JSON-Format als Anfrage an einen Server an die Evaluationsmodelle geschickt und die Auswertungen im JSON-Format wiederum empfangen und weiterverarbeitet.

4.2.2 Infrastruktur

Das gegebene System kombiniert zwei klar voneinander getrennte Teilprogramme. Während ein Programm die rechenintensive Aufgabe übernimmt, für einen eingegebenen Text angemessenes Feedback zu erstellen, ist das andere Programm mit der Visualisierung diesen Feedbacks beauftragt. Die Kommunikation zwischen beiden Programmen erfolgt über eine vordefinierte Programmierschnittstelle (API). Diese Herangehensweise hat viele Vorteile gegenüber einer monolithischen Architektur.

Zwei getrennte Programme, die über eine API kommunizieren, können auf unterschiedlichen Servern bereitgestellt werden. Wenn der Rechenaufwand des Evaluationsmodells zunimmt, könnte es bei einem gemeinsam genutzten Server zu Verzögerungen beim Laden der Website kommen. Um dies zu vermeiden, kann das Evaluationsmodell auf seinem eigenen Server bereitgestellt werden. Dieser kann dann auch mit den entsprechenden Ressourcen ausgestattet und beliebig skaliert werden. Da beide Programme über eine eigene *code base* verfügen, können separate Entwicklungsteams gleichzeitig an der Weiterentwicklung des Evaluationsmodells beziehungsweise der Web-App arbeiten, ohne sich dabei zum Beispiel durch aufkommende Versionskonflikte zu behindern. Durch die klar definierte API werden jegliche Änderungen maskiert und Komponenten können beliebig ersetzt oder umstrukturiert werden. Im gegebenen System werden Usereingaben mithilfe eines HTML-Formulars von der Web-App entgegen genommen. Diese Eingaben werden vorverarbeitet und in ein angemessenes Format überführt. In Form einer HTTP-Anfrage werden sie dann an das Evaluationsmodell übergeben, welches den Text annotiert und das Ergebnis als Antwort auf die Anfrage zurückgibt. Für jedes Evaluationsmodell wird eine gesonderte POST-Anfrage akzeptiert.

POST-Anfragen erhalten ihre Eingabe im JSON-Format. Anfragen für den Aufgabentyp 1 erwarten als Eingabe neben einem Eingabe-String auch einen Themenindex. Ein Eingabestring ist wie folgend formatiert: Antworten zu verschiedenen Fragen müssen durch leere Zeilen getrennt sein, jede stichpunktartige Antwort auf einer eigenen Zeile stehen und wenn ein*e Schüler*in keine Antwort auf eine bestimmte Frage gibt, sollte der String 'DUMMY' als Platzhalter

Eingabe

```
{
  "text": "Sollen soziale Netzwerke wie Twitter als
           Werkzeug im Schulunterricht benutzt werden\n\nAutor
           sieht in Twitter Chancen für innovative konstruktive
           Unterrichtsmethoden\nNutzung von Twitter wird als
           positiv dargestellt\nTwitter sei ein Teil des
           medialen Wandels\n\nLaut Psychologen fördert das
           Internet die Gesprächskultur\nUnterricht ist
           abwechslungsreicher\nDurch Nutzung von Twitter und
           Social Media kann die Diskussion lebhafter verlaufen\n
           Schüchterne Schüler können sich dadurch besser am
           Unterricht beteiligen\nDer Unterricht bleibt länger
           im Gedächtnis der Schüler\nEs ist einfach ein Teil
           des Lebens!\n\nAblenkungen können leichter passieren\n
           Lehrer sind skeptisch gegenüber neuen Methoden",
  "topic": 1
}
```

Abbildung 2: Beispielergabe der API des Evaluationsmodells 1

Ausgabe

```
{
  "0": "In Bezug auf Frage 1:\\n Hier hast du alle
wesentlichen Argumente aus dem Artikel wiedergegeben!
Sehr schön! Zu Frage 2:\\n Im Artikel wird folgendes
Argument benannt: 'autor befürwortet die nutzung von
twitter im unterricht'. Deine Antwort 'autor sieht
in twitter chancen für innovative konstruktive
unterrichtsmethoden' scheint in die gleiche Richtung
zu gehen. Der Artikel sagt auch etwas zu diesem
Argument: 'es liegt an den lehrer die über twitter
geäußerten inhalte auf nützlicher art in den
unterricht zu integrieren'. Vielleicht hast du es
übersehen? Bzgl. Frage 3:\\n Folgendes Argument kommt
auch im Artikel vor und sollte mitberücksichtigt
werden: 'angst vor ablenkung ist nicht begündet da
auch bücher ablenken können'. Dieser Punkt aus dem
Artikel 'nutzung von twitter senkt die hemmung vor
aktiver beteiligung am unterricht' scheint mit deiner
Eingabe 'durch nutzung von twitter und social media
kann die diskussion lebhafter verlaufen' in dieselbe
Richtung zu gehen. Folgendes Argument kommt auch im
Artikel vor und sollte mitberücksichtigt werden: '
pädagogen raten zur auseinandersetzung mit dem
medialen wandel'. Dieser Punkt aus dem Artikel 'es
gibt erfolgsbeispiele für die nutzung von twitter im
unterricht zb in den usa' scheint mit deiner Eingabe
'durch nutzung von twitter und social media kann die
diskussion lebhafter verlaufen' in dieselbe Richtung
zu gehen. Dieser Punkt aus dem Artikel 'auch
schüchterne schüler werden ermutigt ihre meinungen zu
äußern' scheint mit deiner Eingabe 'schüchterne
schüler können sich dadurch besser am unterricht
beteiligen' in dieselbe Richtung zu gehen. Steht
dieses Argument 'es ist einfach ein teil des lebens'
wirklich im Text? Zu Frage 4:\\n Hier hast du alle
wesentlichen Argumente aus dem Artikel wiedergegeben!
Sehr schön!"
}
```

Abbildung 3: Beispielausgabe der API des Evaluationsmodells 1

Eingabe

```
{  
  "text": "Ist Twitter nützlich? Ja"  
}
```

Abbildung 4: Beispielergabe der API des Evaluationsmodells 2

verwendet werden. POST-Anfragen für den Aufgabentyp 2 erwarten nur einen Eingabe-String und keinen Themenindex. Das Format des Strings ist identisch. Da die Web-App die Anpassung der Formatierung übernimmt, müssen sich Benutzer dementsprechend keine Gedanken machen. Sollten dennoch Probleme auftauchen, resultiert eine ungültige POST-Anfrage in einem 422 Error Code.

Setup Die Entwicklung der gegebenen Web-App erfolgte lokal, während die Evaluationsmodelle auf einem Server der Universität Potsdam (UP) lagen. Der Server ist aus Sicherheitsgründen von außen nicht erreichbar. Für den Zugriff auf den Server sind spezielle Rechte notwendig. Um trotzdem Anfragen an die Evaluationsmodelle schicken zu können, muss man den entsprechenden Port tunnelt. Dafür ist eine Verbindung mit dem Intranet der UP notwendig. Das geht beispielsweise mithilfe von *Cisco AnyConnect Secure Mobility Client*, ein VPN (*Virtual Private Network*) Client, der eine sichere und verschlüsselte Verbindung gewährleistet. Das Einrichten eines Tunnels auf dem Betriebssystem Windows ist ein eher aufwändiges Unterfangen im Vergleich zum Einrichten auf Unix-Betriebssystemen wie Linux oder macOS. Für letzteres genügt eine kurze Recherche im Internet.

Deshalb werden hier im Folgenden nur die notwendigen Schritte zum Tunneln für Windows erklärt:

1. Um verschiedene Serverkonfigurationen unter einfach wiederverwendbaren Schlüsselwörtern zu hinterlegen, empfiehlt es sich eine SSH-Konfigurationsdatei zu erstellen. Dafür erstellt man eine Datei **config** (ohne Dateiendung) im Ordner `C:\Users\<username>\.ssh`. Nachfolgend ist der Inhalt der SSH-Konfigurationsdatei angegeben, die für die Entwicklung dieser Web-App erstellt wurde.

Ausgabe

```
{
  "0": {
    "essay_analysis": {
      "0": {
        "text": "Ist Twitter nützlich? Ja",
        "tokens": {
          "0": "Ist",
          "1": "Twitter",
          "2": "nützlich",
          "3": "?",
          "4": "Ja"
        },
        "annotation": {
          "content_zone": [
            {
              "start_token_id": 0,
              "end_token_id": 4,
              "label": "own"
            }
          ],
          "attribution": []
        }
      }
    },
    "feedback_text": "Am Anfang wäre eine Einleitung mit Eckdaten zum Lesetext (Titel, Autor/in, Erscheinungsdatum etc.) angebracht. Vielleicht kannst du noch etwas auf die Gegenargumente eingehen, die im Artikel stehen? Deine Erörterung besteht zu großen Teilen aus deiner eigenen Meinung. Bitte schaue nochmal, ob du nicht mehr Bezug auf den Lesetext nehmen kannst.\nDie Passage bei Satz 1 scheint deine eigene Meinung wiederzugeben. Das kannst du stärker kenntlich machen, z.B. durch Phrasen wie 'Meiner Ansicht nach ...'"
  }
}
```

Abbildung 5: Beispielausgabe der API des Evaluationsmodells 2

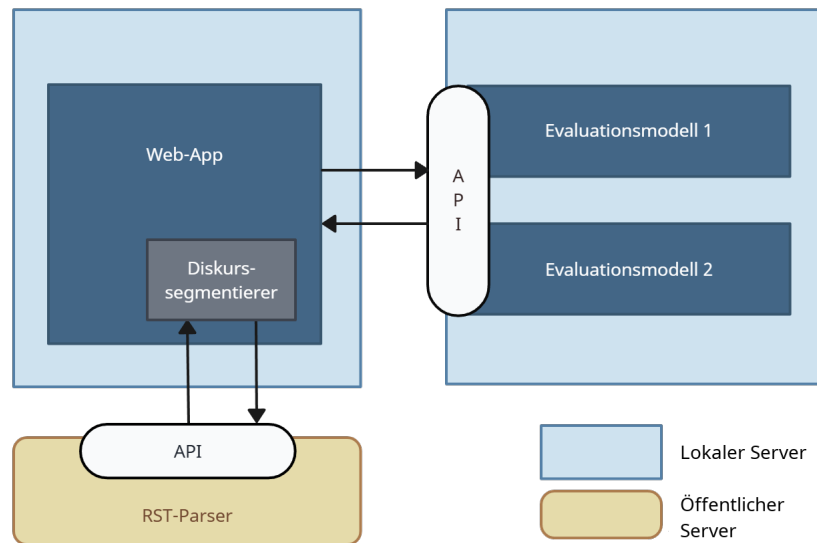


Abbildung 6: Infrastruktur der Web-App

```

1 Host jarvis
2     HostName jarvis.ling.uni-potsdam.de
3     User tnpham
4     Port 22200
5
6 Host stede-hackathon
7     ProxyCommand ssh jarvis nc stede-hackathon 22
8     User tnpham

```

2. Als nächstes erstellt man ein öffentlich/privates Schlüsselpaar um später die Authentifizierung des eigenen PCs durch den Server zu ermöglichen:

```
$ ssh-keygen
```

In diesem Beispiel geben wir dem Schlüsselpaar den Namen **akilas**. Wenn das System nach einer *passphrase* für das Schlüsselpaar fragt, sollte man mit **ENTER** reagieren, um eine Authentifizierung ohne Passwort zu ermöglichen.

3. Nun öffnet man die Windows PowerShell im Admin Modus.

4. Das bereits erstellte Schlüsselpaar muss nun noch offiziell im SSH-Verwaltungssystem registriert werden.

```
$ Get-Service -Name ssh-agent | Set-Service -StartupType  
Manual  
$ Start-Service ssh-agent  
$ ssh-add "C:\Users\\.ssh\akilas"
```

Wichtig: Für den letzten Befehl `akilas` und nicht `akilas.pub` verwenden.

5. Schließlich lässt man den Server wissen, dass er unserem Computer Zugriff gewähren soll. Dafür registriert man den für die Authentifizierung benötigten öffentlichen Teil des Schlüsselpaars auf dem Server.

```
$ type "C:\Users\\.ssh\akilas.pub" | ssh jarvis "  
cat >> .ssh/authorized_keys"  
$ type "C:\Users\\.ssh\akilas.pub" | ssh stede-  
hackathon "cat >> .ssh/authorized_keys"
```

6. Im letzten Schritt tunneln wir den Port des Servers zu unserem lokalen System.

```
$ ssh -L 5678:localhost:5678 stede-hackathon
```

7. War der oben beschriebene Prozess erfolgreich und laufen auf dem Server gerade die Evaluationsmodelle als Service, dann sollte man sich jetzt folgende Adresse der API anzeigen lassen können: `http://localhost:5678/docs`. Sollte noch keine Instanz der Evaluationsmodelle auf dem Server laufen, so kann man sie mit folgendem Befehl erzeugen:

```
~/project/akilas-backend/ArgEssays/Experiments$ python3 app  
.py --port 5678 --debug
```

8. Um die Web-App lokal zu starten, kann man über die Kommandozeile zum Ordner navigieren, in dem die App liegt und den folgenden Befehl ausführen:

```
$ flask run
```

9. Man kann sich nun die Web-App im Browser unter folgendem Link anzeigen lassen: `http://127.0.0.1:5000`

Evaluationsmodelle Es gibt zwei externe Programme, die für die Auswertung der Eingaben verantwortlich sind. Diese werden hier als Evaluationsmodelle beschrieben und liegen auf einem privaten Server der UP. Das Evaluationsmodell für Aufgabe 1 ist ein BERT-basiertes Modell von Bai & Stede (2022) mit zusätzlichen Kontextfeatures. Es vergleicht die Antworten des Users mit Referenzantworten, wobei die Ähnlichkeit einen von drei diskreten Werten annimmt und anhand dessen Feedback generiert wird. Die Eingabe an das Evaluationsmodell 1 beinhaltet einen Themenindex zwischen 1 und 3 und die Antwortstrings. Die Antworten können dabei aus stichpunktartigen Sätzen bestehen, die jeweils in einer eigenen Zeile stehen, während die verschiedenen Antworten zu den Fragen mit einer Leerzeile voneinander getrennt sind. Als Ausgabe wird ein zusammenhängender Feedbacktext erwartet.

Für Aufgabe 2 wird die Klassifikation von Inhaltszonen und Attributionen jeweils von zwei unterschiedlichen Modellen übernommen. Ersteres verwendet ein Ensemble-Modell aus Variationen von SVMs (*Support Vector Machines*) und *Random Forests*, während letzteres ein BiLSTM-CRF Modell (*Bidirectional Long Short-Term Memory with Conditional Random Field*) benutzt (X. Bai, persönliche Kommunikation, 30. Januar 2024). Beide machen von *BERT Embeddings* Gebrauch, die im NLP-Bereich oft gute Resultate erzielen (Devlin et al., 2019). Als Eingabe für das Evaluationsmodell 2 wird ein einzelner Textstring benötigt. Als Ausgabe werden die Annotationen für die einzelnen Tokens erwartet. Die genaue Struktur ist in Abbildung 5 anzusehen.

Diskurssegmentierer Der Diskurssegmentierer nutzt einen externen RST-Parser (*Rhetorical Structure Theory*), der auf einem öffentlichen Server liegt⁹, um Text in Diskurssegmente zu unterteilen. Dazu wird der Output des RST-Parsers verwendet, der ein Parsbaum im XML-Format beinhaltet. Mithilfe von regulären Ausdrücken wird dann auf die Blätter beziehungsweise die Diskurssegmente zugegriffen.

4.2.3 Ordnerstruktur

In diesem Abschnitt wird die Ordnerstruktur erklärt. Abbildung 4.2.3 zeigt hierbei die Hierarchien auf. Diese Dateien befinden sich unter anderem auf GitHub¹⁰.

⁹ Die GUI des RST-Parsers ist über folgende Adresse erreichbar:
<http://85.214.128.188/parser>

¹⁰ <https://github.com/tnt-pham/AKILAS-Web-App>

Listing 1: Projektordnerstruktur

```
project /
|-- README.md Informationen zur Benutzung
|-- requirements.txt Liste der benötigten Python-Pakete mit Versionen
|-- app.py Hauptanwendungsdatei
|-- discoursesegmenter.py Klasse für die Diskurssegmentierung
|-- task2_input_example.txt Beispielinputdatei für Aufgabe 2
|-- templates /
|   |-- index.html Webseite der Anwendung
|
|-- static /
|   |-- css /
|       |-- main.css Style Sheets
|
|   |-- img /
|       |-- up_logo.svg Logo der Universität Potsdam
|
|   |-- js /
|       |-- utilities.js Funktionen für User-Interaktion mit Web-App
```

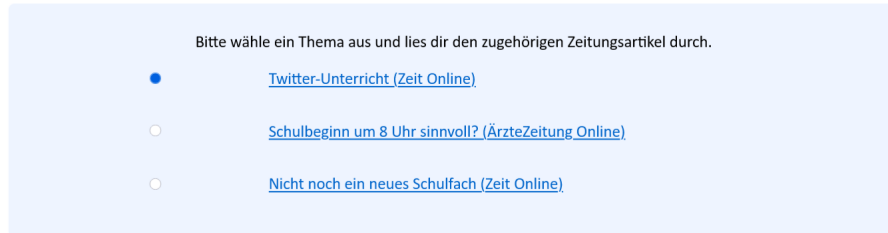
4.3 Design

In diesem Abschnitt wird beschrieben, welche Designentscheidungen getroffen wurden für die Web-App. Es wird dabei nicht nur Bezug genommen auf die in Abschnitt 2.3 erwähnten Designprinzipien, sondern auch auf weitere spezifische für die jeweiligen Elemente relevanten Designempfehlungen.

Auswahl des Artikels Radiobuttons sind Bedienelemente, “mit denen genau eine Option aus mehreren ausgewählt werden kann” (Preim & Dachselt, 2010, S. 381). Eben diese Eigenschaft ist in der Web-App erforderlich bei der Auswahl des Artikels (siehe Abbildung 7). Es muss genau ein Artikel ausgewählt werden, da sonst die Bearbeitung der nächsten Aufgaben nicht stattfinden kann. Es wurde Gebrauch gemacht von Konventionen, da Radiobuttons ein grundlegendes Element in der Webentwicklung sind und das Bedienen dadurch gewohnt erscheint. Dadurch ist keine zusätzliche Erklärung notwendig.

Außerdem empfiehlt Preim & Dachselt die Zusammengehörigkeit der Radiobuttons “durch eine einheitliche Anordnung” (2010, S. 381), am besten in einer

vertikalen Spalte, und zusätzlich durch eine Umrandung zu kennzeichnen. Diese Gruppierung wurde mit einer hellblauen Box umgesetzt.



Bitte wähle ein Thema aus und lies dir den zugehörigen Zeitungsartikel durch.

- ☒ [Twitter-Unterricht \(Zeit Online\)](#)
- ☐ [Schulbeginn um 8 Uhr sinnvoll? \(Ärztezeitung Online\)](#)
- ☐ [Nicht noch ein neues Schulfach \(Zeit Online\)](#)

Abbildung 7: Radiobuttons für Artikelauswahl

Artikel lesen Die Auswahl des Artikels kann durch einen Mausklick auf die Radiobuttons erfolgen, wodurch sich ein Popup-Fenster öffnet. Es ist jedoch auch möglich lediglich auf den zugehörigen Artikellink zu klicken. Die unterstrichene, hellblaue Schrift deutet hierbei auf einen Link hin. “Dies entspricht gängigen Quasistandards, die mit der Entwicklung des WWW entstanden sind” (Preim & Dachzelt, 2010, S. 138). Auch hier wurde also eine Konvention umgesetzt. Dazu gehört ebenso die Änderung des Mauscursors beim Bewegen über den anklickbaren Elementen. Das Benutzen dieser vertrauten Strukturen und Elementen vereinfacht den Einstieg in die Web-App für den User.

Texteingaben Texte können über Eingabefelder übergeben werden. Diese Textfelder müssen geeignet dimensioniert werden (Preim & Dachzelt, 2010, S. 391). Da nicht mit Sicherheit vorhergesehen werden kann, wieviel der User eintippt, existiert die Option, das Feld flexibel in vertikaler Richtung anzupassen. Diese Möglichkeit wird angedeutet durch die geriffelte untere rechte Ecke. “Wenn der eingegebene Text einer gewissen Struktur folgen muss, z.B. bei . . . Daten, ist es besonders wichtig, dass das Format beschrieben wird oder initial ein Standardwert als Beispiel für eine korrekte Eingabe vorhanden ist” (Preim & Dachzelt, 2010, S. 392).

Für Aufgabe 1 werden deshalb ausgegraute Beispielantworten in den Texteingabefeldern gezeigt, um das Format darzustellen. Die weniger gesättigte Farbe deutet darauf hin, dass die derzeitige Eingabe nicht anwendbar ist (Galitz, 2002, S. 148). Die Beispielantwort eines Textfeldes verschwindet jeweils direkt, sobald das erste Zeichen eingegeben wurde. Dies lässt sich rückgängig machen durch das Löschen der Texteingabe.

Aufgabe 1: Textverständnis

Bitte beantworte folgende Verständnisfragen.

1. Was ist das Thema/Streitthema?

- Die Nutzung von Twitter und Social Media im Unterricht

2. Was ist der im Artikel vertretene Standpunkt?

(Falls das Thema "Schulbeginn um 8 Uhr sinnvoll?" ausgewählt wurde, kann diese Frage ignoriert werden, da der Artikel nicht darauf eingeht.)

- Nutzung von Twitter wird als positiv dargestellt
- Twitter sei ein Teil des medialen Wandels

3. Was spricht FÜR den im Artikel vertretenen Standpunkt? Welche Pro-Argumente werden genannt?

- Stichpunktsatz 1
- Stichpunktsatz 2
- ...

4. Was spricht GEGEN den im Artikel vertretenen Standpunkt? Welche Contra-Argumente werden genannt?

- Stichpunktsatz 1
- Stichpunktsatz 2
- ...

Abbildung 8: Aufgabe 1

Die Eingabe für Aufgabe 2 wurde hingegen anders umgesetzt, da hier ein längerer Text eingegeben werden muss. Dabei ist es gut möglich, dass im Laufe des Prozesses das Format der Eingabe vergessen wird. Aus diesem Grund existiert ein separater Kasten mit einer Beschreibung des Formats, sodass eine Orientierung daran während des Schreibens möglich ist (siehe Abbildung 9).

Tipp: Du kannst mehrere Erörterungen schreiben und gleichzeitig evaluieren lassen. Lasse dafür zwischen den Texten genau eine Leerzeile frei, damit sie als einzelne Texte evaluiert werden.
Für Absatzgrenzen kann **<PB>** benutzt werden.

Beispiel:

Dies ist der erste Satz der ersten Erörterung. Ein weiterer Satz folgt.
<PB> Dies ist der nächste Absatz der ersten Erörterung.

Nun fängt die zweite Erörterung an. Dies ist ein weiterer Satz.
Dies ist immer noch Teil des ersten Absatzes der zweiten Erörterung.

Abbildung 9: Eingabehinweis für Aufgabe 2

Die Web-App wurde zudem robust gegenüber Eingaben des Users implementiert. Das bedeutet, dass auch nicht eingehaltene Formate nicht zum Absturz der Anwendung führen. Das gilt konkret für übermäßige Zeilenumbrüche, die von der Web-App tolerant verarbeitet werden.

Eine weitere Entscheidung musste getroffen werden, wie damit umgegangen wird, wenn der User Text ins Eingabefeld und gleichzeitig eine txt-Datei übergibt. Da die txt-Datei des Users bereits in gespeicherter Form vorliegt, ist es kein großes Problem, wenn die Datei ignoriert wird und stattdessen lediglich der Text aus dem Eingabefeld weiterverarbeitet wird. Andersherum würde potentiell mehr Schaden entstehen, wenn der eingetippte Text verloren ginge.

Auswahl der Diskurssegmentierung Laut Preim & Dachzelt sollte es bei Radiobuttons “immer eine aktuelle Auswahl geben” (2010, S. 381), auch beim Start der Anwendung. Ein Standardwert wurde aus diesem Grund stets eingestellt. Im Falle der Diskurssegmentierung wurde sie standardmäßig weggelassen, da die Wartezeit beim Laden mehrere Minuten beträgt und ein die Web-App ohne sie auch funktioniert. Gleichzeitig werden dem User dadurch keine Optionen genommen und alle Freiheiten bleiben.

Evaluation Das Feedback kann generiert werden, indem ein Button von der Maus angeklickt wird. Preim & Dachzelt empfiehlt, dass Buttons “möglichst mit Verben beschriftet werden” sollten (2010, S. 398). Dies wurde umgesetzt mit *FEEDBACK GENERIEREN* (siehe Abbildung 10). Beim Überfahren des Mauscursors über diesen Feedbackbutton ändert sich zudem die Form des Mauscursors und zusätzlich die Farbe des Buttons von blau in orange. Dies suggeriert eine Interaktionsmöglichkeit. Da der Button von zentraler Bedeutung ist, ist er sichtbar am unteren Bildschirmrand angebracht und bleibt fixiert, auch beim Scrollen.



Abbildung 10: Button zur Feedbackgenerierung

5 Ausblick

In diesem Kapitel werden mögliche Erweiterungen und Verbesserungen für die Web-App beschrieben und Ideen für zukünftige Forschung vorgeschlagen. Zum einen würde die Entwicklung der Web-App von einem Team aus Expert*innen verschiedenster Fachrichtungen profitieren. Diese Fachgebiete wurden bereits in Abschnitt 2.2 beschrieben. Die Kollaboration und Kooperation von Entwickler*innen unterschiedlicher Perspektiven und Schwerpunkte kann zur Verbesserung der Web-App beitragen. Beispielsweise könnten pädagogische Einblicke dabei helfen, die Aufgaben so zu gestalten, dass Lernziele effektiver erreicht werden. Man könnte also die Aufgabenauswahl modifizieren oder erweitern. Konkret bedeutet das, dass zum Beispiel mehr Artikel zum Lesen zur Option freigegeben, für Aufgabe 1 die Fragen umformuliert oder ganz neue Fragen hinzugefügt werden könnten. Weitere Aufgabentypen könnten ergänzt werden, sodass abwechslungsreiches Lernen ermöglicht würde. Zudem soll die Web-App zukünftig von Schüler*innen im jugendlichen Alter benutzt werden. Deswegen sollte die Anwendung auch mit einem Fokus auf diese Benutzer*innengruppe gestaltet werden. Dabei sollten die Bedürfnisse des Umfelds jedoch mitberücksichtigt werden, also von Lehrkräften und Eltern.

Zum anderen würde die Web-App auch von der Verbesserung der Evaluationsmodelle profitieren. Einerseits könnte ein detaillierteres Feedback dabei hel-

fen, dass Schüler*innen ihre Stärken und Schwächen besser verstehen. Dafür müssten die Evaluationsmodelle zuerst bessere Resultate erzielen. Andererseits würde aus Programmierungsperspektive eine Modifizierung des Ausgabeformats der Evaluationsmodelle die Programmstruktur der Web-App verschönern. Programmlogik, die eindeutig den Evaluationsmodellen zuzuordnen wäre, wird bisher nämlich von der Web-App übernommen. Beispielsweise wird bei Aufgabe 1 das Feedback zu allen vier Antworten als ein einziger String vom Evaluationsmodell zurückgegeben. Die Web-App muss deshalb das Feedback zu den einzelnen Antworten mühsam extrahieren. Weiterhin kann das Evaluationsmodell 1 verbessert werden, indem eine beliebige Anzahl an Fragen beantwortet werden kann, bevor eine Auswertung möglich ist. Bisher ist es so, dass eine zu kleine Anzahl von beantworteten Fragen zu einem Programmfehler im Evaluationsmodell führt, obwohl sogenannte *DUMMY*-Antworten generiert werden. Um Schüler*innen vor einem dadurch verursachten potentiellen 500er Fehlercode zu schützen, verhindert daher die GUI eine Evaluation, solange nicht genug Antworten geliefert wurden.

Zu guter Letzt profitiert die Web-App vor allem durch Tests mit Schüler*innen. Sie sollte vor allem möglichst in einem Kontext angewendet werden, in dem sie auch später eingesetzt wird. Einblicke aus dieser Evaluation können in der Entwicklung benutzt werden, um die Web-App zu optimieren, bevor sie letztendlich auf einem öffentlichen Server für alle zugänglich wird.

6 Fazit

Diese Arbeit beschäftigte sich mit der Entwicklung einer GUI als Lernplattform für Schüler*innen. Es wurde dafür gezeigt, dass das Design einer solchen Benutzungsschnittstelle das Ergebnis einer kollaborativen Arbeit ist und Kenntnisse aus vielen Fachgebieten erfordert. Zum einen wurden Prinzipien aus dem Interaktionsdesign genutzt, um die Web-App benutzungsfreundlicher zu gestalten und zum anderen das Python-Framework Flask in Kombination mit Jinja2 und JavaScript verwendet, um interaktive Funktionalität zu integrieren. Die Realisierung der GUI als Web-App ist also gelungen. Damit hat die Entwicklung der Web-App jedoch lediglich begonnen. Die Weiterentwicklung und Optimierung sind daher Gegenstand der zukünftigen Forschung.

Literatur

- Antcheva, I., Brun, R. & Rademakers, F. (2005). Guidelines for Developing a Good GUI. *Computing in High Energy Physics 2004*, 613-616. doi: 10.5170/CERN-2005-002.613
- Bai, X. & Stede, M. (2022). Argument Similarity Assessment in German for Intelligent Tutoring: Crowdsourced Dataset and First Experiments. *European Language Resources Association*, 2177-2187.
- Bush, Vannevar. (1945). As We May Think. *The Atlantic Monthly*.
- Dean, J. (2018). *Web Programming with HTML5, CSS, and JavaScript*. Jones & Bartlett Learning.
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. doi: 10.18653/v1/N19-1423
- Edwards, B. (2020, 23. Mai). *Windows 3.0 Is 30 Years Old: Here's What Made It Special*. How To Geek. <https://www.howtogeek.com/674574/windows-3.0-is-30-years-old-heres-what-made-it-special/>.
- Engelbart, D. C. (1962, Oktober). *Augmenting Human Intellect: A Conceptual Framework*. Doug Engelbart Institute. <https://dougengelbart.org/content/view/138/>.
- Galitz, W. O. (2002). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques* (2. Aufl.). Wiley Computer Publishing.
- Harrison, C., Hsieh, G., Willis, K. D. D., Forlizzi, J. & Hudson, S. E. (2011). *Kineticons: Using Iconographic Motion in Graphical User Interface Design*. Association for Computing Machinery. doi: 10.1145/1978942.1979232
- Kay, A. (1993). The Early History of Smalltalk. *Sigplan Notices - SIGPLAN*, 28, 69-95. doi: 10.1145/155360.155364
- Keith, J. & Sambells, J. (2010). *DOM Scripting: Web Design with JavaScript and the Document Object Model*. Apress. doi: 10.1007/978-1-4302-3390-9_1

LITERATUR

- Nielsen, J. (2004, 26. September). *Checkboxes vs. Radio Buttons*. Nielsen Norman Group. <https://www.nngroup.com/articles/checkboxes-vs-radio-buttons/>.
- Nielsen, J. (2017, 18. August). *Jakob's Law of Internet User Experience [Video]*. YouTube. <https://www.youtube.com/watch?v=wzb4mK9DiHM>.
- Norman, D. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. MIT Press.
- Norman, D. & Draper, S. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Taylor & Francis.
- Nuhel, A. (2021). Evolution of Smartphone. *American International University-Bangladesh*.
- Preim, B. & Dachsel, R. (2010). *Interaktive Systeme - Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung*. Springer. doi: 10.1007/978-3-642-05402-0
- Reimer, J. (2005, 5. Mai). *A History of the GUI*. Ars Technica. <https://arstechnica.com/features/2005/05/gui/>.
- Roser, M. (2022, 6. Dezember). *The brief history of artificial intelligence: The world has changed fast – what might be next?* Our World in Data. <https://ourworldindata.org/brief-history-of-ai>.
- Saputra, D. G. & Azizah, F. N. (2013). A Metadata Approach for Building Web Application User Interface. *Procedia Technology*, 11, 903-911. doi: <https://doi.org/10.1016/j.protcy.2013.12.274>
- Simpson, R. M., Renear, A., Mylonas, E. & van Dam, A. (1996). 50 Years After “As We May Think”: The Brown/MIT Vannevar Bush Symposium. *Interactions*, 3 (2), 47-67. doi: 10.1145/227181.227187
- Ständige Wissenschaftliche Kommission der Kultusministerkonferenz (SWK). (2023). *Empfehlungen zum Umgang mit dem akuten Lehrkräftemangel: Stellungnahme der Ständigen Wissenschaftlichen Kommission der Kultusministerkonferenz (SWK)* (Bericht). doi: 10.25656/01:26372
- Valverde, R. (2011). *Principles of Human Computer Interaction Design: HCI Design*. Lambert Academic Publishing.

LITERATUR

- Vojislav, S., Vlajić, S., Milic, M. & Ognjanovic, M. (2011). Guidelines for Framework Development Process. *7th Central and Eastern European Software Engineering Conference*, 1-9. doi: 10.1109/CEE-SECR.2011.6188465
- Was ist eine Web-App? Definition und Web-App-Beispiele.* (2021, 11. März). Digital Guide IONOS. <https://www.ionos.de/digitalguide/websites/web-entwicklung/verschiedene-app-formate-was-ist-eine-web-app/>.
- What is a graphical user interface (GUI)?* (2020, 14. September). Digital Guide IONOS. <https://www.ionos.com/digitalguide/websites/web-development/what-is-a-gui/>.
- Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L. & Tang, Y. (2023). A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of Automatica Sinica*, 10, 1122-1136. doi: 10.1109/JAS.2023.123618
- Xu, W., Dainoff, M., Ge, L. & Gao, Z. (2021). From Human-Computer Interaction to Human-AI Interaction: New Challenges and Opportunities for Enabling Human-Centered AI. *Computing Research Repository*.

Selbstständigkeitserklärung

Hiermit versichere ich, Nhật Trường Thomas Phạm, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Zuhilfenahme anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht.

Berlin, den 1. Februar 2024

Nhật Trường Thomas Phạm