Entwicklung und Evaluation einer Applikation zur automatischen Erstellung informativer Social Media Stories

Bachelorarbeit

Zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von
Jan Hillesheim

Erstgutachter:	Jun.-Prof. Dr. Dennis Riehle (Institut für Wirtschaftsinformatik)
Zweitgutachter:	Arnold Arz von Straussenburg, M.Sc. (Institut für Wirtschaftsinformatik)

Kaltenengers, im November 2024

Inhaltsverzeichnis

Abbildungsverzeichnis ... III
Tabellenverzeichnis ... IV
Abkürzungsverzeichnis ... V
1 Motivation und Forschungsansatz ... 1
2 Hintergrund ... 3
2.1 Large Language Models (LLM) ... 3
2.2 Generative Text-zu-Bild-Modelle ... 4
2.3 Text-zu-Sprache-Modelle ... 5
2.4 Weitere verwandte Arbeiten ... 5
3 Forschungsmethodologie ... 7
3.1 Design Science Research ... 7
3.2 Prozess der Design Science Research Methodology ... 8
4 Design des Softwareartefakts ... 10
4.1 Anforderungen ... 10
4.2 Designentscheidungen ... 11
5 Entwicklung ... 19
5.1 Architektur-Überblick ... 19
5.2 Entwicklung des Backends ... 20
5.3 Entwicklung der Worker ... 24
5.4 Entwicklung des Frontends ... 30
6 Demonstration ... 40
6.1 Demonstration der Videogenerierung aus einer Stellenanzeige ... 40
6.2 Demonstration der Videogenerierung aus einem informativen Text ... 42
6.3 Demonstration der Videogenerierung aus einem reinen Prompt ... 44
7 Evaluation ... 46
7.1 Evaluation der resultierenden Videoartefakte ... 46
7.2 Evaluation des Softwareartefakts ... 48
7.3 Technische Evaluation ... 51
8 Diskussion ... 56
8.1 Interpretation der Ergebnisse ... 56
8.2 Abgrenzung zu verwandten Arbeiten ... 58
8.3 Einschränkungen und Vorschläge zur zukünftigen Forschung ... 59
9 Fazit ... 62

Abbildungsverzeichnis

Abbildung 1 DSRM-Prozessmodell 8
Abbildung 2 Komponentendiagramm mit Technologieübersicht 17
Abbildung 3 Komponentendiagramm 20
Abbildung 4 Datenmodell als ER-Diagramm 21
Abbildung 5 Flussdiagramm eines generalisierten Workers 24
Abbildung 6 Pydantic Datenmodell des Storyboards 25
Abbildung 7 Standbild aus einem generierten Kurzvideo 28
Abbildung 8 Screenshot der Login-Komponente 30
Abbildung 9 Screenshot der Hauptmenü-Komponente 31
Abbildung 10 Screenshot der Projekterstellung aus einem Text 32
Abbildung 11 Screenshot der Projekterstellung aus einer Stellenanzeige 33
Abbildung 12 Screenshot der Detailansicht einer Stellenanzeige 33
Abbildung 13 Screenshot der Storyboard-Komponente 34
Abbildung 14 Screenshot der Komponente zur Bearbeitung eines Kapitels 35
Abbildung 15 Screenshot der Komponente, die das finale Video anzeigt 36
Abbildung 16 Screenshot der Komponente für Einstellungen 37
Abbildung 17 Screenshot der eingefügten URL der Stellenanzeige 40
Abbildung 18 Screenshot der zur Stellenanzeige ermittelten Details 41
Abbildung 19 Screenshot des Storyboards zur Stellenanzeige 41
Abbildung 20 Screenshot des fertigen Kurzvideos zur Stellenanzeige 42
Abbildung 21 Screenshot des eingefügten Textes über das Grundgesetz 43
Abbildung 22 Screenshot des Storyboards über das Grundgesetz 43
Abbildung 23 Screenshot des fertigen Grundgesetz-Kurzvideos 44
Abbildung 24 Screenshot des Prompts für ein Kurzvideo zur Geschichte Englands . . 44
Abbildung 25 Screenshot des Storyboards über die Geschichte Englands 45
Abbildung 26 Screenshot des fertigen Kurzvideos zur Geschichte Englands 45
Abbildung 27 Evaluationsergebnisse zu den Aussagen *A1* bis *A9* 47
Abbildung 28 Evaluationsergebnisse zu den Aussagen *A10*, *A11* und *A12* 48
Abbildung 29 Generierung einer Story aus der Stellenanzeige von *AceGaming* 49

Tabellenverzeichnis

Tabelle 1 Liste von funktionalen und nicht-funktionalen Anforderungen 11
Tabelle 2 Liste von Designentscheidungen 14
Tabelle 3 LLM-Benchmark Resultate 15
Tabelle 4 *GenEval*-Benchmark Resultate 15
Tabelle 5 Relevante Parameter der *Jobsuche-API* 23
Tabelle 6 Umsetzung der Anforderungen 39
Tabelle 7 Evaluation der Designentscheidungen 54

Abkürzungsverzeichnis

AI	Artificial Intelligence
API	Application Programming Interface
CD	Corporate Design
DSR	Design Science Research
DSRM	Design Science Research Methodology
GAN	Generative Adversarial Network
JSON	JavaScript Object Notation
JWT	JSON Web Token
KI	Künstliche Intelligenz
KMU	Kleines und mittleres Unternehmen
LLM	Large Language Model
MGSM	Multilingual Grade School Math Benchmark
NLP	Natural Language Processing
RLS	Row Level Security
TTS	Text to Speech
UI	User Interface
WER	Word Error Rate

Zusammenfassung

Generative Künstliche Intelligenz (KI) findet in immer mehr Einsatzgebieten Anwendung. Auch die Erstellung von Social Media Inhalten wird von der neuen Technologie geprägt. Das Ziel dieser Bachelorarbeit ist es, ein prototypisches Softwareartefakt zu entwickeln und zu evaluieren, das die Erstellung informativer Social Media Stories mit generativer KI automatisiert. Dazu werden Sprach-, Text-zu-Bild- und Text-zu-Sprache-Modelle eingesetzt, um Kurzvideos in Story-Form aus beliebigen textuellen Eingaben zu erzeugen. Für die Gestaltung, Entwicklung und Evaluation des Softwareartefakts wird *Design Science Research* nach Hevner et al. als Forschungsmethodologie angewendet. Dazu wird zuerst eine umfangreiche Wissensgrundlage über die Problemstellung und die genutzten Technologien gebildet, aus der dann Anforderungen und Designentscheidungen für den Softwareprototypen abgeleitet werden. Die modulare Softwarearchitektur ermöglicht die verteilte Ausführung mehrerer KI-Modelle auf verschiedenen Systemen und stellt deren Ergebnisse in Echtzeit auf einer webbasierten Benutzeroberfläche dar. So wird demonstriert, wie für das konkrete Anwendungsgebiet des Ausbildungsmarketings informative Stories generiert werden, die die wichtigsten Fakten aus existierenden Stellenanzeigen in Textform zielgruppengerecht als Kurzvideo zusammenfassen. Darüber hinaus wird auch der allgemeine Einsatz der Software für jegliche Art von informativen Inhalten gezeigt und der Automatisierungsgrad bewertet. Einschränkungen des Prototyps und Möglichkeiten zur Verbesserung durch zukünftige Forschung werden durch ein Experteninterview festgestellt und diskutiert. Die Ergebnisse der Evaluation durch eine qualitative Befragung der Kernzielgruppe zeigen, dass mithilfe der Software verständliche und informative Stories generiert werden konnten.

Abstract

Generative Artificial Intelligence (AI) is increasingly being applied across a wide range of fields, including the creation of social media content. The aim of this bachelor's thesis is to develop and evaluate a prototypical software artifact that automates the generation of informative social media stories using generative AI. To achieve this, large language models, text-to-image models, and text-to-speech models are employed to generate short videos in a story format from arbitrary textual inputs. The design, development, and evaluation of the software artifact are conducted using Design Science Research according to Hevner et al. An extensive knowledge base including details of the problem and used technologies is first established, from which requirements and design decisions for the software prototype are derived. The modular software architecture enables the distributed execution of multiple AI models across various systems, presenting their results in real-time through a web-based user interface. The prototype demonstrates how informative stories can be generated specifically for human resources marketing, summarizing the key points from existing job postings into short videos for the corresponding audience. Additionally, the broader application of the software for various types of informative content is showcased, and the level of automation is assessed. Limitations of the prototype and possibilities for improvement through future research are identified and discussed through an expert interview. The results of a qualitative survey with the core target audience show that the software was able to generate comprehensible and informative stories.

1 Motivation und Forschungsansatz

Organisationen stehen durch sinkende Bewerberzahlen in den letzten 10 Jahren vor der Herausforderung, für ihre Ausbildungsplätze und dualen Studienplätze die Aufmerksamkeit potenzieller Bewerber zu gewinnen und diese Stellen effektiv zu besetzen (Bundesagentur für Arbeit, n. d.). Neben der Darstellung eines Ausbildungsplatzes als Fließtext-Stellenanzeige gewinnt die Erstellung von visuellen Inhalten, z. B. Kurzvideos, immer mehr an Bedeutung (Statista, 2018). Viele größere Organisationen produzieren daher schon solche Werbeinhalte, um sich als Ausbildungsbetrieb modern und zielgruppenorientiert zu präsentieren. Inhalte, die gezielt eine der vielen offenen Stellen vermarkten sollen, sind jedoch noch selten, da der Produktionsaufwand für große Organisationen mit teilweise hunderten verschiedenen Stellen nicht wirtschaftlich sinnvoll ist und Kleine und mittlere Unternehmen (KMUen) die benötigten Ressourcen und das Knowhow häufig erst gar nicht besitzen. Zeitgleich werden generative KI-Modelle, insbesondere Text-zu-Bild-Modelle, immer relevanter, da die Qualität der erzeugten Bilder mittlerweile auf einem für Marketing-Zwecke gut nutzbaren Niveau angekommen ist (Hartmann et al., 2023). Als Forschungsziel sollen u. a. diese Modelle genutzt werden, um ein prototypisches Softwareartefakt zur automatisierten Erstellung von informativen Stories aus Ausbildungs-Stellenanzeigen zu entwickeln und zu evaluieren.

Im Rahmen der Bachelorarbeit soll ein prototypisches Softwareartefakt entwickelt und evaluiert werden, das aus strukturierten oder unstrukturierten Daten u. a. mittels eines Text-zu-Bild-Modells Kurzvideos in Story-Form generiert. Dieses Artefakt soll das Verfahren an dem konkreten Anwendungsfall von Stories für Ausbildungs-Stellenanzeigen demonstrieren. Zu der Nutzergruppe des Artefakts gehören beispielsweise Content-Ersteller aus dem Personalmarketing von Ausbildungsbetrieben und Betreiber von Plattformen zur Vermittlung von Ausbildungsstellen. Bei der Zielgruppe der generierten Stories handelt es sich vor allem um Schulabsolventen auf der Suche nach einem Ausbildungsplatz. Die Applikation soll webbasiert sein und durch eine interaktive Benutzeroberfläche bedient werden. Der Fokus liegt auf der technischen Machbarkeit und den Möglichkeiten, das Verfahren durch den Einsatz kommerzieller Modelle in der Cloud oder quelloffener Modelle auf eigenem Hosting zu optimieren. Es wird untersucht, wie verschiedene generative Modelle für die Erstellung visueller Inhalte genutzt werden können und diskutiert, inwieweit diese Ansätze auf unterschiedliche Anwendungsbereiche übertragbar sind. Die Entwicklung einer angemessenen Evaluationsmethode, um die Effektivität und Effizienz der erzeugten Story-Videos zu messen, ist ein wesentlicher Bestandteil der Arbeit. Diese Evaluationsmethode soll den Grad der Automatisierung bewerten und feststellen, wann menschliches Eingreifen erforderlich oder vorteilhaft ist. Die Qualität der Ergebnisse soll durch eine Nutzerbefragung bewertet werden.

Die beschriebenen Ziele sollen im Rahmen des *Design Science Research* (DSR) Frameworks nach Hevner et al. (2004) erreicht werden. Dazu wird in Kapitel 2 „Hintergrund“ existierendes Wissen in der Domäne des Problems in Form von Grundlagen und verwandten Arbeiten gesammelt. Nachdem in Kapitel 3 „Forschungsmethodologie“ genauer auf das DSR Framework und dessen konkrete Anwendung in dieser Arbeit eingegangen wird, kommt in Kapitel 4 „Design des Softwareartefakts“ das gesammelte Grundlagen-Wissen zur Anwendung, um Anforderungen und Designentscheidungen für das zu entwickelnde Softwareartefakt abzuleiten. Nach der „Entwicklung“ (Kapitel 5) und der „Demonstration“ (Kapitel 6) des Softwareartefakts, wird im 7. Kapitel dessen Evaluation beschrieben. Die Arbeit schließt mit einer Diskussion (Kapitel 8) und einem Fazit (Kapitel 9) ab.

In dieser Bachelorarbeit wird ausschließlich das generische Maskulinum verwendet. Dies dient der Vereinfachung und besseren Lesbarkeit des Textes. Es wird betont, dass sämtliche Bezeichnungen von Personen unabhängig vom grammatikalischen Geschlecht alle Geschlechter gleichermaßen einschließen.

2 Hintergrund

In diesem Kapitel werden zentrale Begriffe und Konzepte beschrieben, auf denen diese Abschlussarbeit aufbaut. Dazu gehören insbesondere Large Language Models (Kapitel 2.1), generative Text-zu-Bild-Modelle (Kapitel 2.2) und generative Text-zu-Sprache-Modelle (Kapitel 2.3). Außerdem werden in Kapitel 2.4 weitere verwandte Arbeiten identifiziert, um diese Bachelorarbeit in aktuelle Forschungsthemen einzuordnen.

2.1 Large Language Models (LLM)

Große Sprachmodelle (Large Language Models, LLMs) sind KI-Modelle, die entwickelt wurden, um menschliche Sprache zu verstehen (Natural Language Processing, NLP) und Texte auf der Grundlage umfangreicher Datenmengen zu generieren (Vaswani et al., 2017). Diese Modelle werden auf vielen existierenden Texten trainiert und basieren häufig auf Deep-Learning-Technologien, insbesondere der 2017 von Vaswani et al. vorgestellten Transformer-Architektur (Vaswani et al., 2017). Das ermöglicht ihnen, komplexe sprachliche Muster, Kontexte und Semantik zu erfassen. LLMs, wie zum Beispiel *Metas Llama 3*, können eine Vielzahl von Aufgaben der natürlichen Sprachverarbeitung ausführen, einschließlich Textgenerierung, Übersetzung, Zusammenfassung und Beantwortung von Fragen (Llama Team, 2024). Dadurch wurden sie schnell zu wichtigen Werkzeugen in verschiedenen Bereichen wie Bildung, Gesundheitswesen und Wirtschaft.

Für die Anwendung von LLMs in einer bestimmten Anwendungsdomäne lässt sich deren Performance für diese eine bestimmte Aufgabe durch Nachtrainieren (fine-tuning) signifikant steigern (Brown et al., 2020). Brown et al. zeigen in ihrer 2020 veröffentlichten Arbeit „Language Models are Few-Shot Learners“ jedoch, dass die Vergrößerung der Parameter-Anzahl von LLMs einen solchen Performance-Anstieg bewirkt, dass diese großen Modelle bei der Nutzung von *Few-Shot-Prompting* mit kleineren gezielt nachtrainierten Modellen mithalten können. In vielen Anwendungsdomänen genügt es also, innerhalb des Prompts einige Beispiele zur Problemlösung zu demonstrieren, um so die Qualität der Ausgabe deutlich zu verbessern (Brown et al., 2020).

Ihre Komplexität und die Abhängigkeit von großen Datensätzen werfen jedoch auch Bedenken hinsichtlich der Generierung von falschen oder sogar schädlichen Inhalten auf (Llama Team, 2024). Für die Anwendungsdomäne der Generierung von Inhalten in deutscher Sprache ist besonders zu beachten, dass beim Training der meisten LLMs überwiegend englische Texte verwendet werden. Somit ist die Fehleranfälligkeit bei der Textgenerierung, sowohl syntaktisch als auch semantisch, signifikant höher als bei englischen

Konversationen (Llama Team, 2024). Bei der Auswahl eines Modells ist die Menge der für das Training verwendeten fremdsprachigen Texte also ein wichtiger Faktor.

Um zuverlässige Applikation zu entwickeln, die die Ausgabe von LLMs programmatisch weiterverarbeiten können, ist es essenziell, diese Ausgabe in einem strukturierten Format zu erhalten. Während verschiedene LLMs schon länger die Möglichkeit bieten, eine Ausgabe in validem JSON-Format zu erzwingen, garantiert dies nicht das Einhalten eines bestimmten vorgegebenen Schemas (OpenAI, 2024). Diese Einschränkung hat *OpenAI* für seine *GPT-4o*-Modelle im August 2024 mit der Vorstellung des „Structured Outputs“ gelöst (OpenAI, 2024). Hier kann für die Generierung einer Antwort eine konkrete Struktur-Vorgabe als *JSON Schema*[1], einer deklarativen Sprache zur Definition der Struktur von JSON-Daten, festgelegt werden. Durch *Constrained decoding*, also das systematische Einschränken des Modells in der Wahl der Antwort-Token, kann eine 100-prozentige Einhaltung des Schemas garantiert werden (OpenAI, 2024). Dies ist essenziell für die Weiterverarbeitung der LLM-Ausgabe, auf der diese Abschlussarbeit aufbaut.

2.2 Generative Text-zu-Bild-Modelle

Generative Text-zu-Bild-Modelle wandeln Textbeschreibungen in synthetische Bilder um, wobei tiefe neuronale Netze, insbesondere *Generative Adversarial Networks (GANs)* und Diffusionsmodelle, eine zentrale Rolle spielen (Rombach et al., 2022). GANs bestehen aus einem Generator, der Bilder erstellt, und einem Diskriminator, der zwischen echten und synthetischen Bildern unterscheidet. Diese beiden Netze konkurrieren miteinander, was zu immer realistischeren Bildern führt (Brock et al., 2018). Diffusionsmodelle hingegen arbeiten durch eine schrittweise Reduzierung von Rauschen in den Bildern, um hochaufgelöste und realistische Grafiken zu generieren. Sie bieten oft stabilere Trainingsprozesse und qualitativ hochwertigere Ergebnisse als GANs (Rombach et al., 2022). Daher sind in den letzten Jahren Diffusionsmodelle zum De-facto-Ansatz für die Generierung hochauflösender Bilder und Videos aus natürlichen Spracheingaben geworden (Esser et al., 2024).

Rectified Flow ist ein neuer Ansatz für den Aufbau von generativen Diffusionsmodellen. Hierbei werden Daten und Rauschen auf einer geraden Linie verbunden. Trotz besserer theoretischer Eigenschaften und konzeptioneller Einfachheit hat sich dieser Ansatz bisher noch nicht flächendeckend als Standard etabliert. *Rectified Flow* wird beispielsweise von den neuen generativen Text-zu-Bild-Modellen der *Stable Diffusion 3*-Familie von *Stability AI* verwendet (Esser et al., 2024).

[1] *JSON Schema* Dokumentation: json-schema.org

Durch die hohe Qualität der generierten Bilder haben sich Diffusionsmodelle bereits in vielen Marketingbereichen als wichtiges Werkzeug etabliert.

2.3 Text-zu-Sprache-Modelle

Generative Text-zu-Sprache-Modelle (Text to Speech, TTS) wandeln Eingaben in Textform in Audiodateien um, in denen der eingegebene Text ausgesprochen wird. Mit dem Erfolg von Transformer-basierten Modellen in verschiedensten Domänen, existieren auch einige generative Text-zu-Sprache-Modelle, die auf der Transformer-Architektur aufbauen (L. Chen & Rudnicky, 2022). Diese Transformer-basierten TTS-Modelle stellen eine gute Alternative zu sogenannten *seq2seq*-TTS-Modellen dar, die zwar eine qualitativ ähnliche Ausgabe liefern, aber mehr Zeit zur Generierung dieser Ausgaben benötigen (L. Chen & Rudnicky, 2022). Beispiele für aktuell beliebte Transformer-basierte TTS-Modelle sind *XTTS-v2* von *Coqui.ai* und *Bark* von *Suno* (Hugging Face Inc., n. d. a).

Menschliche Sprache besteht neben dem linguistischen Inhalt auch aus para-linguistischen Informationen. Dazu gehören u. a. die Sprachcharakteristika eines bestimmten Sprechers. Viele TTS-Modelle können im Trainingsprozess die Sprachcharakteristika aus dem Input extrahieren und so später die Sprachinterferenz mit verschiedenen Sprachcharakteristika ermöglichen (L. Chen & Rudnicky, 2022).

Neben objektiven Evaluationsmethoden, wie dem Ermitteln der *Word Error Rate (WER)*, ist das automatisierte Bestimmen der Qualität von TTS-Modellen schwierig (L. Chen & Rudnicky, 2022). Da es für Menschen einfach ist, die Natürlichkeit und den Tonfall einer Stimme zu beurteilen, gibt es Projekte wie die *TTS-Arena* von *Huggingface*, die TTS-Leaderboards, basierend auf subjektiven Bewertungen von Nutzern, veröffentlichen (Hugging Face Inc., n. d. b). Dies ist ein wichtiges Werkzeug, um passende TTS-Modelle auswählen zu können.

2.4 Weitere verwandte Arbeiten

In den vorangegangenen Abschnitten wurden bereits mehrere verwandte Arbeiten genannt, die zum Grundverständnis der beschriebenen Technologien beitragen, auf die diese Abschlussarbeit aufbaut. Neben den grundliegenden Forschungsergebnissen existieren noch weitere verwandte Arbeiten, in denen ähnliche Ziele und/oder ähnliche Vorgehensweisen wie in dieser Bachelorarbeit beschrieben werden.

In der 2020 erschienenen Arbeit „Automatic Video Creation From a Web Page“ von Chi et al. beschreiben die Google-Mitarbeiter, wie sich die Inhalte einer Website als Kurz-

video automatisiert darstellen lassen. Dabei wurden die Resultate durch direkten Vergleich mit manuell erstellten Inhalten evaluiert, und es konnte gezeigt werden, dass der Videoerstellungs-Prozess mit dem vorgestellten Tool effektiv unterstützt wird (Chi et al., 2020).

In „Automatic Generation of Multimedia Teaching Materials Based on Generative AI“ haben Xu Chen und Di Wu im Jahr 2024 über den Einsatz von KI im Bildungssektor geschrieben. Konkret wurde ein technisches Framework entwickelt, um mit generativen Modellen multimediale Ressourcen für die Lehre zu generieren. Dazu wurden Techniken zur Text-, Bild- und Videogenerierung angewandt, um passende Videos zu Poesie aus der chinesischen Tang-Dynastie zu generieren. So konnten die Konnotationen und Hauptbestandteile der Gedichte von Probanden besser verstanden werden (X. Chen & Wu, 2024).

Eine ähnliche Vorgehensweise beschreiben Liu et al. im 2023 erschienenen Paper „Generative Disco: Text-to-Video Generation for Music Visualization“. Hier werden LLMs und generative Text-zu-Video-Modelle eingesetzt, um Visualisierungen für existierende Musik zu generieren. Es werden verschiedene Entwurfsmuster für die Videogenerierung eingeführt und gezeigt, dass Experten durch den Einsatz der Software in der Videoerstellung unterstützt werden konnten (Liu et al., 2023).

Einen Beitrag dazu, wie Autoren von generativer KI bei der Erstellung von Handlungssträngen unterstützt werden können, wurde 2022 von Chung et al. in „TaleBrush: Sketching Stories with Generative Pretrained Language Models“ veröffentlicht. Hier geht es zwar nicht um die tatsächliche Generierung von fertigen Videos, die Arbeit liefert aber relevantes Wissen, das zur Erstellung der ‘Drehbücher‘ in dieser Arbeit genutzt werden kann (Chung et al., 2022).

Im Jahr 2021 haben Chi et al. ihre Arbeit „Automatic Instructional Video Creation from a Markdown-Formatted Tutorial“ publiziert, in der mit *HowToCut* ein automatisierter Ansatz zur Erstellung informativer Tutorial-Videos vorgestellt wird. Dazu wird aus einem Tutorial im Markdown-Format ein interaktives Video mit synthetisierter Stimme und visuellen Instruktionen generiert. Die Videobearbeitung erfolgt automatisch und enthält unter anderem auch Texteinblendungen. Die Evaluation der Videoqualität durch eine Online-Umfrage zeigte, dass die vorgestellte Methode in der Lage ist, informative und nützliche Videos zu generieren (Chi et al., 2021).

Das in diesem Kapitel gesammelte Hintergrundwissen in Verbindung mit der Problembeschreibung und Motivation aus Kapitel 1 bildet im Folgenden die *Knowledge Base*, sodass nach der genaueren Betrachtung der Forschungsmethodologie in Kapitel 3 mit dem Design des Softwareartefakts (Kapitel 4) begonnen werden kann.

3 Forschungsmethodologie

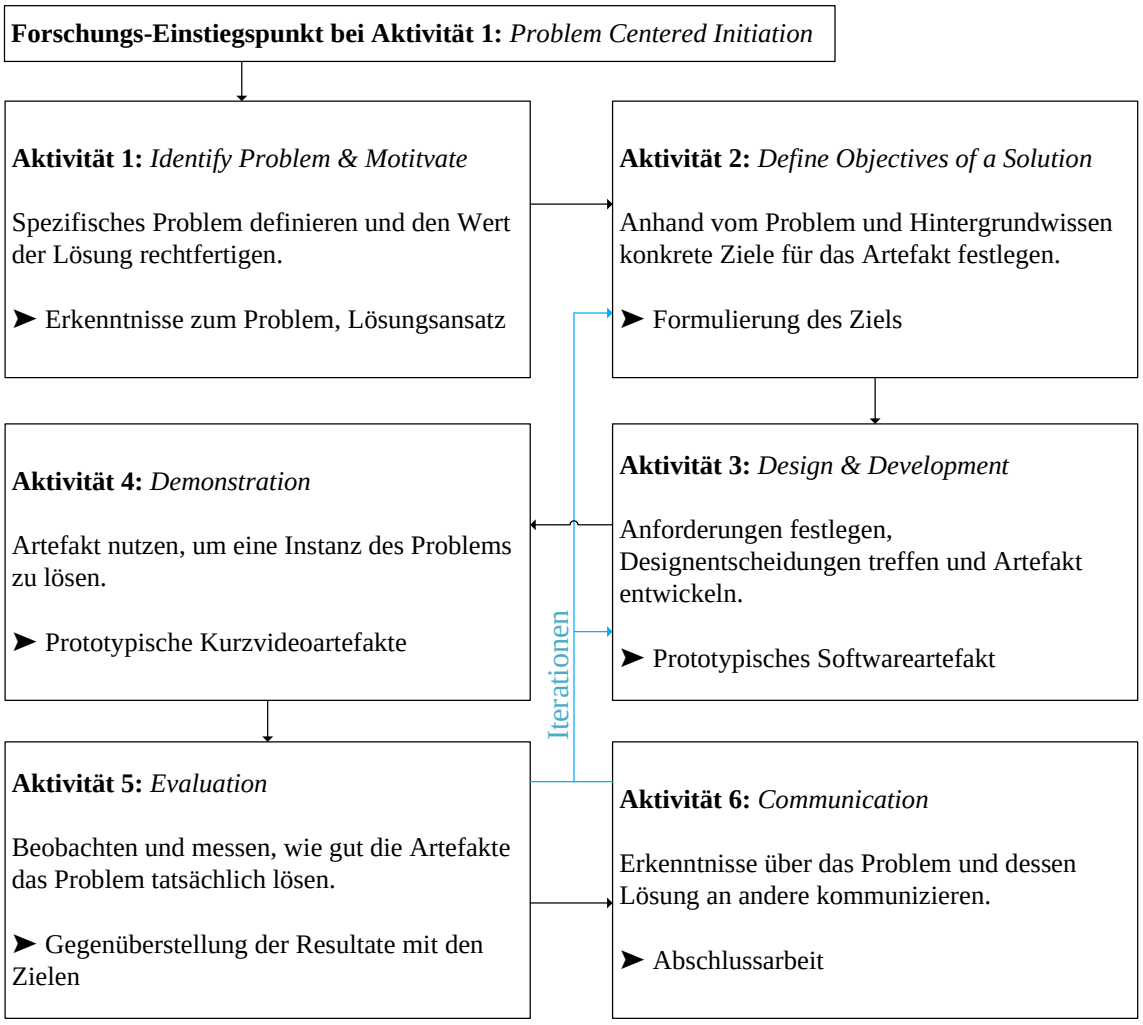
Im Folgenden wird zuerst allgemein das *Design Science Research (DSR)* Framework nach Hevner et al. (2004) erläutert, um anschließend konkret darauf einzugehen, wie die Schritte der *Design Science Research Methodology (DSRM)* von Peffers et al. (2007) in dieser Arbeit umgesetzt werden.

3.1 Design Science Research

Als Forschungsmethodologie zur Problemlösung und Entwicklung der Artefakte wird ein gestaltungsorientierter Ansatz genutzt. Durch den Einsatz von *Design Science Research* nach Hevner et al. (2004) wird Wissen darüber gewonnen, wie Informationssysteme entworfen oder entwickelt werden können. Konkret leistet die Abschlussarbeit vor allem einen Beitrag zu *Design Entities* und gehört somit zur ersten DSR-Projekt-Kategorie nach Brocke und Maedche (2019). Als DSR-Prozessmodell werden die sechs Schritte der *Design Science Research Methodology* von Peffers et al. (2007) verwendet, die in Abbildung 1 dargestellt werden.

Um das Problem und *Environment* für das DSR-Framework zu definieren, werden Erkenntnisse aus der Literatur verwendet. Für die spezielle Domäne des Ausbildungsmarketings werden außerdem Resultate aus einem Workshop zu den Erfahrungen bei der Ausbildungssuche (mit Auszubildenden, deren Suche nach einem Ausbildungsplatz weniger als drei Jahre zurückliegt) genutzt. Als Grundlage für die *Knowledge Base* dient eine Literaturrecherche.

Nach der Design- und Entwicklungsphase, in der u. a. ein prototypisches Softwareartefakt und erste KI-generierte Videos entstehen, wird die Ex-Post-Evaluation durch Interviews mit Personen aus der Story-Zielgruppe (Schülerinnen und Schüler ab der 10. Klassenstufe, die durch Ausbildungsmarketing angesprochen werden sollen) durchgeführt. Dazu werden die Kurzvideos und die Quell-Daten (i.e. Stellenanzeigen in Textform) gezeigt und von den Probanden anhand verschiedener subjektiver Merkmale verglichen.



Vgl. (Peffers et al., 2007)

Abbildung 1 DSRM-Prozessmodell

3.2 Prozess der Design Science Research Methodology

Die sechs Schritte bzw. *Activities* der *Design Science Research Methodology* von Peffers et al. (2007) werden häufig für gestaltungsorientierte Forschung verwendet und gelten im Bereich der Informationssysteme als allgemein akzeptiert. Im Folgenden werden diese etablierten Schritte kurz allgemein erläutert und dann konkret beschrieben, wie sie in dieser Abschlussarbeit Anwendung finden. Dazu wird zu jedem Schritt angegeben, mit welchen Abschnitten dieser Arbeit er korrespondiert.

Activity 1: Problem identification and motivation. Hier soll lt. Peffers et al. das spezifische Problem definiert und der Wert einer Lösung gerechtfertigt werden (Peffers et al., 2007). Die aus diesem Schritt gewonnenen Erkenntnisse zu Problem und Lösungsansatz werden in der Motivation in Kapitel 1 beschrieben und bilden die Grundlage zur Definition des *Environments* für das DSR-Framework.

Activity 2: Define the objectives for a solution. Anhand der Problemdefinition und dem Wissen darüber, was möglich und machbar ist, sollen hier konkrete Ziele für das zu entwickelnde Artefakt festgelegt werden (Peffers et al., 2007). Diese werden in der Zielformulierung in Kapitel 1 genannt und erklärt.

Activity 3: Design and development. In diesem Schritt werden Anforderungen festgelegt, Designentscheidungen getroffen und das tatsächliche Artefakt entwickelt (Peffers et al., 2007). Diese Aktivitäten korrespondieren in der Arbeit mit den Kapiteln „Anforderungen“ (Kapitel 4.1), „Designentscheidungen“ (Kapitel 4.2) und „Entwicklung“ (Kapitel 5). Als Ressource wird für diesen Schritt außerdem Theoriewissen für die *Knowledge Base* benötigt (Peffers et al., 2007), das im Kapitel 2 „Hintergrund“ aus existierenden Forschungsergebnissen gewonnen wird.

Activity 4: Demonstration. Das im vorherigen Schritt entwickelte Artefakt soll hier genutzt werden, um eine Instanz des Problems zu lösen (Peffers et al., 2007). Die Generierung von ersten Stories als Artefakte zur späteren Evaluation wird in der „Demonstration“ (Kapitel 6) beschrieben.

Activity 5: Evaluation. Hier soll beobachtet und gemessen werden, wie gut die bisher entstandenen Artefakte das Problem tatsächlich lösen (Peffers et al., 2007). Neben einem Experteninterview zur Evaluation des Softwareartefakts (Kapitel 7.2), werden die generierten Stories durch Interviews mit der Zielgruppe in „Evaluation der resultierenden Videoartefakte“ (Kapitel 7.1) evaluiert. Im Kapitel 7.3 „Technische Evaluation“ werden schlussendlich noch die real erzielten Resultate mit den vorher gesetzten Zielen und Anforderungen gegenübergestellt.

Activity 6. Communication. Im letzten Schritt sollen die gewonnenen Erkenntnisse über das Problem und dessen Lösung an andere Forschende kommuniziert werden (Peffers et al., 2007). Dies geschieht über das Verfassen und Einreichen dieser Abschlussarbeit und kann zukünftig durch den praktischen Einsatz des Softwareartefakts ergänzt werden.

Mit diesem Plan zur Anwendung der Forschungsmethodologie und dem in den vorangegangenen Kapiteln aufgebautem Wissen zur Umgebung des Softwareartefakts und der *Knowledge Base*, wird im Folgenden das Design und die Entwicklung der prototypischen Software beschrieben.

4 Design des Softwareartefakts

In diesem Kapitel werden Anforderungen an das zu entwickelnde Softwareartefakt aufgestellt und auf deren Grundlage Designentscheidungen getroffen. Zusätzliche Designentscheidungen, für die weiteres Hintergrundwissen aus der Literatur benötigt wird, werden im Anschluss in eigenen Unterabschnitten behandelt.

4.1 Anforderungen

In diesem Abschnitt werden Anforderungen (Requirements) für die Entwicklung des Softwareartefakts aufgestellt. Diese werden aus der Motivation, der Problembeschreibung und dem gesammelten (Hintergrund-) Wissen abgeleitet. Die Anforderungen erhalten jeweils einen Bezeichner (*REQ1* bis *REQ18*) und werden in funktionale bzw. nicht-funktionale Anforderungen eingeteilt (Shehadeh et al., 2021). Als resultierendes Artefakt entsteht Tabelle 1, eine Liste aller Anforderungen, auf deren Grundlage die Designentscheidungen im nächsten Abschnitt getroffen werden.

Bez.	**Kategorie**	**Anforderung**
REQ1	funktional	Die Software muss Stellenanzeigen sowohl in strukturierter Form (im JSON-Format) als auch in unstrukturierter Form akzeptieren.
REQ2	funktional	Die Software soll die relevantesten Informationen aus der Eingabe identifizieren und extrahieren können.
REQ3	funktional	Die Software muss aus den extrahierten Informationen ein Storyboard für ein informatives Kurzvideo in Story-Form generieren.
REQ4	funktional	Das Storyboard soll aus mehreren Abschnitten bestehen, die jeweils einen Titel, den textuellen Inhalt und ein Prompt für eine passende Hintergrundgrafik enthalten.
REQ5	funktional	Die Software muss dem Nutzer die Möglichkeit geben, das Storyboard vor der Generierung des Videos zu bearbeiten.
REQ6	funktional	Die Software muss aus den Prompts für die Hintergrundgrafiken die entsprechenden Bilddateien generieren.
REQ7	funktional	Die Software muss aus den generierten Texten die entsprechenden Audiodateien generieren.
REQ8	funktional	Die Software muss aus den Informationen des Storyboards und den generierten Bild- und Audiodateien ein Kurzvideo im MP4-Format rendern.

Bez.	Kategorie	Anforderung
REQ9	funktional	Der Nutzer muss die Möglichkeit haben, das generierte Video anzusehen und es bei Bedarf nach Änderungen am Storyboard erneut zu generieren.
REQ10	funktional	Die Software muss dem Nutzer ermöglichen, das fertige Video im MP4-Format herunterzuladen.
REQ11	funktional	Der Nutzer muss die Möglichkeit haben, die Parameter der Generierung (z. B. Prompts für die generativen KI-Modelle) anzusehen und zu verändern.
REQ12	nicht-funktional	Die Software soll auf allen gängigen Endgeräten (Desktops und Mobilgeräten) ohne Installation zugänglich sein.
REQ13	nicht-funktional	Die Software darf nur autorisierten Nutzern zugänglich sein.
REQ14	nicht-funktional	Die rechenintensiven Generierungsprozesse sollen auf mehrere verschiedene Systeme verteilt werden können.
REQ15	nicht-funktional	Der Nutzer soll permanent einen Überblick über den Fortschritt der Generierung haben und diese in Echtzeit mitverfolgen können.
REQ16	nicht-funktional	Die Software muss von Nutzern ohne technischen Hintergrund bedienbar sein.
REQ17	nicht-funktional	Die generierten Kurzvideos sollen möglichst ähnlich zu gängigen Social Media Stories aussehen.
REQ18	nicht-funktional	Die generierten Kurzvideos in Story-Form sollen eine Kennzeichnung enthalten, dass sie mit Hilfe generativer KI erstellt wurden.

Tabelle 1 Liste von funktionalen und nicht-funktionalen Anforderungen

4.2 Designentscheidungen

Aus den im letzten Abschnitt definierten Anforderungen an das Softwareartefakt und den im Hintergrund gesammelten Informationen und Einschränkungen werden nun entsprechende Designentscheidungen abgeleitet.

Designentscheidungen, die sich unmittelbar aus den Anforderungen *REQ1* bis *REQ18* ergeben, werden in Tabelle 2 aufgeführt und als *DE1* bis *DE20* bezeichnet. Weitere Designentscheidungen, wie die Auswahl der verschiedenen KI-Modelle, benötigen umfangreichere Entscheidungsprozesse. Diese werden inklusive Argumentation in den nachfolgenden Abschnitten beschrieben.

Bez.	Anf.	Designentscheidung
DE1	REQ1, REQ2	Die vom Nutzer bereitgestellten Informationen (sowohl in strukturierter als auch in nicht-strukturierter Form) sollen als Eingabe für ein LLM genutzt werden. Dazu werden sie mit einem entsprechenden Prompt zusammengeführt, der das LLM anweist, wie mit den Informationen verfahren werden soll.
DE2	REQ1	Die Daten der Jobsuche-API[2] der Bundesagentur für Arbeit soll verwendet werden, um dem Nutzer eine einfache Import-Möglichkeit für Informationen in strukturierter Form anzubieten.
DE3	REQ3	Um konsistente Resultate von dem LLM zu erhalten, die in den Folgeschritten verarbeitet werden können, soll das LLM angewiesen werden, die Ausgabe in strukturierter Form (als JSON-Objekt) bereitzustellen.
DE4	REQ4	Die strukturierte Ausgabe des LLMs soll aus einer Liste von Abschnitten mit variabler Länge bestehen. Jedes Listenelement soll den Titel, den Inhalt und ein Prompt für eine passende Grafik des jeweiligen Abschnitts beinhalten.
DE5	REQ5	Die Datengrundlage für das Storyboard soll in einer Datenbank gespeichert werden und über eine grafische Benutzeroberfläche durch den Nutzer bearbeitet werden können.
DE6	REQ6	Ein generatives Text-zu-Bild-Modell soll genutzt werden, um aus den vom LLM erzeugten Prompts die Hintergrundgrafiken zu generieren. Diese sollen als Bilddateien in der Datenbank gespeichert werden und im Storyboard angezeigt werden, sobald sie verfügbar sind.
DE7	REQ7	Ein generatives Text-zu-Sprache-Modell soll genutzt werden, um aus den vom LLM erzeugten Texten Audiodateien zu generieren. Diese sollen in der Datenbank gespeichert werden und im Storyboard abgespielt werden können, sobald sie verfügbar sind.
DE8	REQ8	Die strukturierten Informationen aus dem Storyboard und die generierten Bild- und Audiodateien sollen verwendet werden, um durch deren Komposition, Einblendungen von Texten und weiteren visuellen Effekten, regelbasiert ein Video zu definieren und dieses automatisiert zu rendern.
DE9	REQ9	Nachdem das fertige Kurzvideo generiert wurde, soll es in der Datenbank gespeichert und in der Benutzeroberfläche abgespielt werden. Nach Änderungen am Storyboard hat der Nutzer die Möglichkeit, die automatisierte Videogenerierung mit den veränderten Eingaben erneut zu starten.

[2] Dokumentation der Jobsuche-API: github.com/bundesAPI/jobsuche-api

Bez.	Anf.	Designentscheidung
DE10	REQ10	Die Benutzeroberfläche stellt das generierte Video nach einem Klick auf eine Schaltfläche als Download im MP4-Format zur Verfügung.
DE11	REQ11	Die Parameter und Einstellungen für die verschiedenen generativen Modelle und die Videogenerierung sollen in der Datenbank gespeichert werden. Die Benutzeroberfläche soll diese Informationen anzeigen und Änderungen durch den Nutzer in der Datenbank abspeichern.
DE12	REQ12	Die Benutzeroberfläche ist von der Logik zur Generierung von Bildern, Audio und Video getrennt und wird als Web-Applikation entwickelt und bereitgestellt.
DE13	REQ13	Beim Zugriff auf die Web-Applikation wird ein Benutzername und Passwort abgefragt und mit einer Liste von Benutzern in der Datenbank abgeglichen.
DE14	REQ14	Die Programme zur eigentlichen Erstellung von Inhalten durch generative KI-Modelle (Worker) sollen komplett unabhängig von der Benutzeroberfläche auf beliebigen Systemen ausgeführt werden können. Diese greifen mit der Benutzeroberfläche auf eine gemeinsame Datenbank zu, um so Aufträge zu erhalten und die fertigen Resultate der Benutzeroberfläche zur Verfügung zu stellen.
DE15	REQ15	Damit generierte Inhalte in der Benutzeroberfläche erscheinen, sobald die Generierung abgeschlossen ist, soll die Datenbank die Benutzeroberfläche über Websockets in Echtzeit über Änderungen und Einfüge-Operationen durch Worker benachrichtigen.
DE16	REQ16	Die Web-Applikation soll aus gängigen Bedienelementen aufgebaut sein und den Nutzer intuitiv durch den Generierungsprozess leiten, indem in jedem Schritt kurze Erklärungen angezeigt werden.
DE17	REQ17	Im resultierenden Kurzvideo soll sich das Hintergrundbild jedes Kapitels auf einem Animationspfad bewegen, um einen ‘Kamera-Schwenk-Effekt‘ zu erzielen.
DE18	REQ17	Im resultierenden Kurzvideo soll der Titel jedes Kapitels als ‘Sticker‘ farblich hervorgehoben angezeigt werden.
DE19	REQ17	Im resultierenden Kurzvideo soll der vorgelesene Text in jedem Kapitel auch in kleinerer Schriftart am unteren Bildschirmrand angezeigt werden, sodass dieser mitgelesen werden kann.

Bez.	Anf.	Designentscheidung
DE20	REQ18	In der finalen Videodatei wird vor dem eigentlichen Kurzvideo eine Sekunde lang ein Hinweis angezeigt, dass das folgende Video mit Hilfe von generativer KI erstellt wurde.

Tabelle 2 Liste von Designentscheidungen

Für das geplante Softwareartefakt, das automatisiert Kurzvideos im Story-Format erstellen soll, werden außerdem verschiedene Arten von generativen KI-Modellen benötigt. Um die Eingaben zu verarbeiten und ein erstes Storyboard zu generieren, soll ein *Large Language Model (LLM)* genutzt werden. Jeder Abschnitt in diesem Storyboard besteht neben dem Titel aus einer Hintergrundgrafik und einem kurzen Text. Während das LLM einen Prompt für ein passendes Hintergrundbild vorschlägt, soll für die tatsächliche Generierung der Grafik ein generatives Text-zu-Bild-Modell verwendet werden. Um die textuellen Inhalte eines Abschnitts als Audiospur im Kurzvideo verwenden zu können, sollen die Texte durch ein generatives Text-zu-Sprache-Modell vorgelesen werden.

Im Folgenden werden für jede der drei KI-Kategorien mehrere geeignete quelloffene Modelle aus der Literatur identifiziert und anhand von objektiven Benchmarks und subjektiven Tests im Kontext der Anwendungsdomäne verglichen. Es wird jeweils ein Modell ausgewählt, das für die Implementierung des Softwareartefakts genutzt wird.

Auswahl eines geeigneten Large Language Models. Die Textgenerierung bei Nutzung der Software soll in Echtzeit bzw. möglichst nah an der durchschnittlichen menschlichen Lesegeschwindigkeit erfolgen. Damit dies im Rahmen der für diese Abschlussarbeit nutzbaren Ressourcen möglich ist, soll ein Modell mit einer entsprechend geringen Parameteranzahl verwendet werden. Bei der Interferenz auf einer Grafikkarte mit 10 GiB VRAM bieten sich dazu vor allem Modelle mit weniger als 10 Milliarden Parameter an, wie zum Beispiel *Llama 3.1 8B* von *Meta* (Llama Team, 2024). Zu den beliebtesten in dieser Größenordnung konkurrierenden offenen Modellen gehören außerdem *Gemma 2 9B* von *Google* und *Mistral 7B* von *Mistral AI* (Llama Team, 2024). Zum Vergleich der Modelle ist es üblich, standardisierte Benchmarks zu verwenden. Die jeweils erzielten Resultate zu den relevanten Benchmarks für die relevanten Modelle aus dem *Llama*-Paper werden in Tabelle 3 dargestellt. Gerade in der Anwendungsdomäne der Ausbildungsstellenanzeigen spielt die Fähigkeit, deutsche Inhalte zu generieren, eine wichtige Rolle. Eine Metrik, wie gut die Modelle mit Sprachen abseits von Englisch umgehen können, liefert die *Multilingual Grade School Math Benchmark (MGSM)* nach Shi et al. (2022). Hier schneidet das LLM von *Meta* im Vergleich besser ab als die beiden anderen Modelle.

Modell	MMLU (5-shot)	MMLU-Pro (5-shot, CoT)	IFEval	MGSM (0-shot, CoT)
Llama 3 8B	69,4	48,3	80,4	68,9
Gemma 2 9B	72,3	k. A.	73,6	53,2
Mistral 7B	61,1	36,9	57,6	29,9

Vgl. (Llama Team, 2024)

Tabelle 3 LLM-Benchmark Resultate

Dies deckt sich mit der subjektiven Wahrnehmung bei der Benutzung aller drei Modelle auf Deutsch. Durch diese Erkenntnisse aus der Literatur und durch eigenes Testen der Modelle für die konkrete Anwendungsdomäne werden für die weitere praktische Ausarbeitung in dieser Abschlussarbeit die Modelle der *Llama 3* Familie (*Llama 3 8B* und *Llama 3.1 8B*) verwendet (Designentscheidung *DE21*).

Auswahl eines geeigneten Text-zu-Bild-Modells. Auch bei der Auswahl eines geeigneten Text-zu-Bild-Modells ist eine Abwägung zwischen der Ausführungsgeschwindigkeit auf einer Grafikkarte mit 10 GiB VRAM und der Qualität der generierten Bilder nötig. Zu den führenden quelloffenen Modellen gehören die verschiedenen *Stable Diffusion* Modelle von *Stability AI*, *minDALL-E*, *PixArt-alpha* und *DeepFloyd-IF-XL* (Esser et al., 2024). Eine gängige Metrik, um generative Text-zu-Bild-Modelle zu vergleichen, liefert das Evaluierungsframework *GenEval* nach Ghosh et al. (2023). Die jeweils erzielten Resultate für diese Benchmark werden in Tabelle 4 aufgeführt.

Modell	**GenEval-Score (in der Kategorie „Objects Overall“)**
minDALL-E	0,23
SD v1.5	0,43
PixArt-alpha	0,48
SD v2.1	0,50
SDXL	0,55
SDXL Turbo	0,55
IF-XL	0,61
SD 3 Medium (depth=24)	0,62

Vgl. (Esser et al., 2024)

Tabelle 4 *GenEval*-Benchmark Resultate

Durch diese Ergebnisse aus der Literatur und der subjektiv ausreichenden Generierungsgeschwindigkeit mit den verfügbaren Ressourcen wird für die weitere praktische Ausarbeitung das *Stable Diffusion 3 Medium* Modell (*SD 3 Medium*) von *Stability AI* verwendet (Designentscheidung *DE22*).

Auswahl eines geeigneten Text-zu-Sprache-Modells. Als drittes generatives KI-Modell soll ein geeignetes Text-zu-Sprache-Modell für die Erzeugung von Audiodateien aus einer Texteingabe ausgewählt werden. Hierzu werden über die Suchfunktion von *Hugging Face* Modelle der Kategorie „Text to Speech“ selektiert und die Resultate nach Beliebtheit (Anzahl der Likes) sortiert (Hugging Face Inc., n. d. a). Dabei heben sich zwei aktuelle Modelle hervor, da sie die einzigen sind, die mehr als 1.000 Likes erhalten haben (Stand Oktober 2024): *XTTS-v2* von *Coqui.ai* und *Bark* von *Suno*. Da zu letzterem Modell keine offizielle Forschungsarbeit (research paper) mit Benchmarks im Vergleich zu anderen Modellen existiert, wurden beide Modelle in der Anwendungsdomäne der Ausbildungsstellenanzeigen getestet und deren Ergebnisse subjektiv verglichen. Hierbei fiel auf, dass *XTTS-v2* häufig Probleme mit sprachlichen Besonderheiten wie Abkürzungen (z. B. „bzw. “ wird als „B-Z-W-Punkt“ vorgelesen) und Aufzählungen (z. B. „1. August“ wird als „Eins August“ vorgelesen) aufweist. Die mit *Bark* durchgeführten Tests lieferten auch keine perfekten Ergebnisse für die deutsche Sprache, der Redefluss wurde aber deutlich weniger durch Fehler in der Aussprache gestört. Aufgrund dieser Ergebnisse und weiterer durchgeführter Versuche wird für die praktische Ausarbeitung das Modell *Bark* von *Suno* verwendet (Designentscheidung *DE23*).

Auswahl geeigneter weiterer Technologien. Neben dem Entscheidungsprozess zur Auswahl der zu nutzenden KI-Modelle werden in diesem Abschnitt weitere Technologien ausgewählt und die Auswahl begründet. Eine Übersicht, über die schlussendlich verwendeten Technologien befindet sich in Abbildung 2.

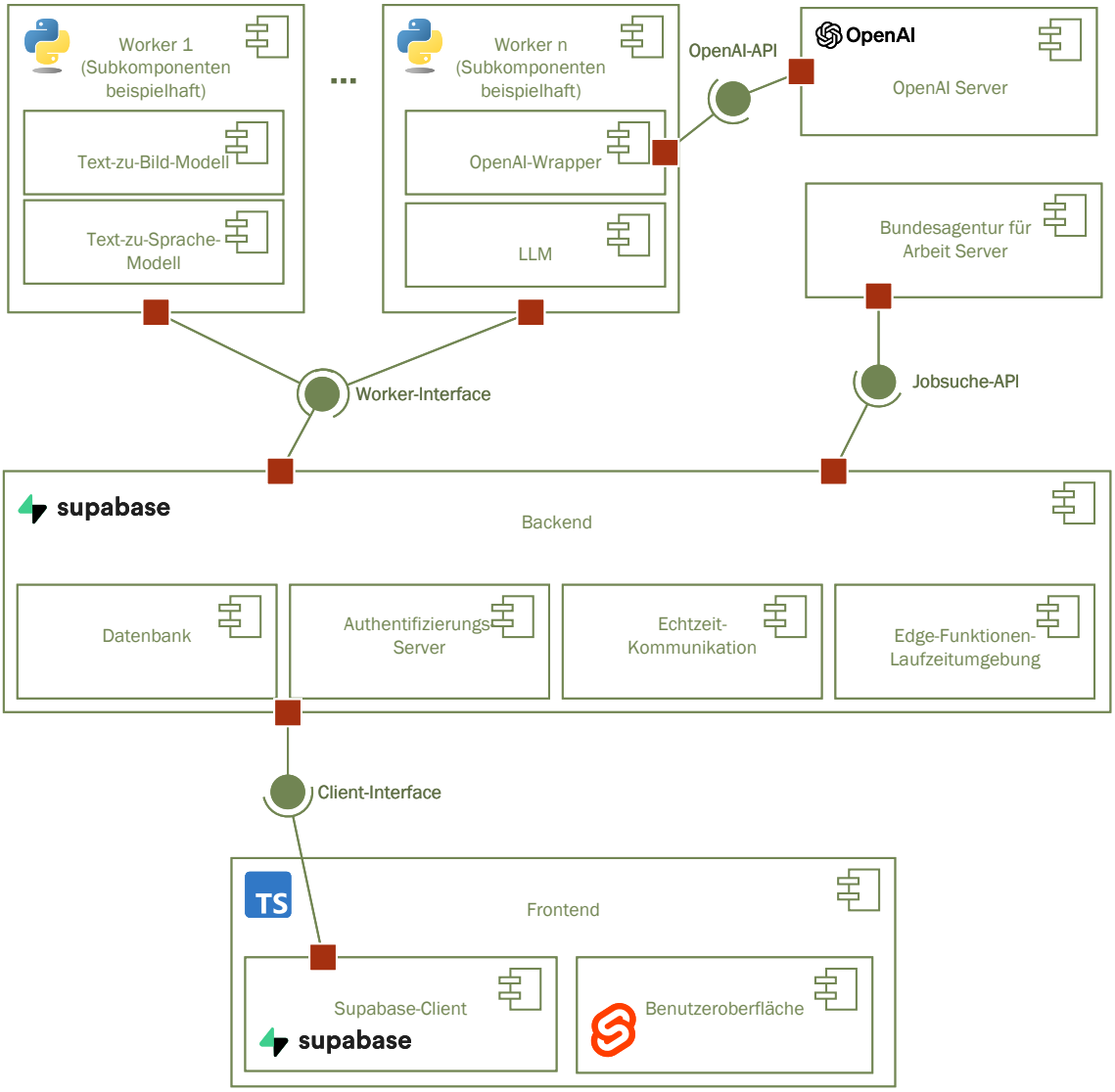


Abbildung 2 Komponentendiagramm mit Technologieübersicht

Für die Ausführung bzw. die Interferenz der generativen KI-Modelle soll Python[3] als Programmiersprache verwendet werden, da fast alle Dokumentationen und Ressourcen der ausgewählten Modelle Python verwenden (Designentscheidung *DE24*). Für die Entwicklung des Frontends mit Webtechnologien ist es gängige Praxis, ein JavaScript-Framework zu nutzen (Stack Exchange Inc., 2024). Durch Erfahrungen mit dem Metaframework *SvelteKit*[4] und der laut Stack Exchange Inc. (2024) generell hohen Beliebtheit dieser Technologie soll *SvelteKit* mit TypeScript[5] zur Entwicklung der Benutzeroberfläche verwendet werden (Designentscheidung *DE25*). Innerhalb des Backends soll es eine relationale Datenbank für persistenten Speicher, eine Authentifizierung der Nutzer und Kommuni-

[3] Python Homepage: python.org
[4] SvelteKit Homepage: kit.svelte.dev
[5] TypeScript Homepage: typescriptlang.org

kationsmöglichkeiten in Echtzeit per Websockets geben. Um den Entwicklungsaufwand des prototypischen Softwareartefakts möglichst gering zu halten, soll ein existierendes Backend-Framework verwendet werden, das alle benötigten Dienste integriert hat. Eine bekannte quelloffene Softwarelösung, die alle genannten Anforderungen erfüllt und Clientbibliotheken für JavaScript und Python bereitstellt, ist *Supabase*[6]. Für die Entwicklung des Backends soll *Supabase* auf eigenem Hosting verwendet werden (Designentscheidung *DE26*). Für die Bereitstellung und Orchestrierung der Dienste für Frontend und Backend soll *Docker*[7] genutzt werden. Die Dienste sollen entsprechend containerisiert auf eigenem Hosting ausgeführt werden (Designentscheidung *DE27*). Als Reverseproxy für den Zugriff auf die verschiedenen Dienste soll *Nginx*[8] verwendet werden (Designentscheidung *DE28*).

Um neben den quelloffenen KI-Modellen, auf denen der Fokus dieser Arbeit liegen soll, im Softwareartefakt auch die Möglichkeit zu haben, kommerzielle Modelle zu verwenden, werden die aktuellen Produkte von *OpenAI*[9] als kommerzielle Alternativen genutzt. Dazu gehören *gpt-4o-mini* als LLM und *tts-1-hd* als Text-zu-Sprache-Modell (*DE29*).

Die in diesem Kapitel getroffenen Designentscheidungen werden im nächsten Kapitel als Grundlage für die Entwicklung des prototypischen Softwareartefakts genutzt.

[6] Supabase Homepage: supabase.com
[7] Docker Homepage: docker.com
[8] Nginx Homepage: nginx.org
[9] OpenAI Homepage: openai.com

5 Entwicklung

In diesem Kapitel werden die relevantesten Konzepte aus dem Entwicklungsprozess des prototypischen Softwareartefakts vorgestellt und der Bezug zu den vorher aufgestellten Designentscheidungen hergestellt. Dazu wird zuerst ein allgemeiner Überblick über die Architektur der Software gegeben, um später die Entwicklung und Funktionsweisen der einzelnen Komponenten im Detail zu beschreiben. Der komplette Quellcode des Softwareartefaks und weitere zugehörige Dateien befinden sich im Gitlab-Repository dieser Abschlussarbeit[10].

5.1 Architektur-Überblick

Das prototypische Softwareartefakt besteht aus verschiedenen Komponenten, die über Schnittstellen miteinander kommunizieren. Durch diese lose Kopplung untereinander können Komponenten einfacher ausgetauscht oder abgeändert werden, was den Prototyping-Prozess beschleunigt und in Zukunft mehr Flexibilität gewährt. Außerdem können verschiedene Komponenten auf diese Art und Weise auch auf verschiedenen Systemen ausgeführt werden und so eine Verteilung der rechenintensiven Aufgaben ermöglicht werden (siehe *DE12*, *DE14*). Eine allgemeine Übersicht über die verschiedenen Komponenten und deren Interaktion wird in Abbildung 3 gezeigt. Die Darstellung der Worker-Komponenten wird dabei nur beispielhaft mit den Workern *Worker 1* bis *Worker n* angegeben, da sowohl deren Anzahl als auch die Aufgabenverteilung (im Diagramm als Sub-Komponenten gezeigt) zur Laufzeit frei gewählt und verändert werden können.

Die Komponenten *OpenAI Server* und *Bundesagentur für Arbeit Server*, mit den jeweiligen Schnittstellen *OpenAI-API* und *Jobsuche-API*, gehören nicht zu dem entwickelten Softwareartefakt und werden von *OpenAI* bzw. von der *Bundesagentur für Arbeit* bereitgestellt. Sie wurden in das Komponentendiagramm aufgenommen, da deren Existenz wichtig für das Gesamtverständnis der Architektur ist.

In den folgenden Abschnitten wird detaillierter auf die Entwicklung der Backend-, Frontend- und Worker-Komponenten eingegangen.

[10] Gitlab Repository zu dieser Abschlussarbeit: gitlab.uni-koblenz.de/hillesheim/bachelorarbeit

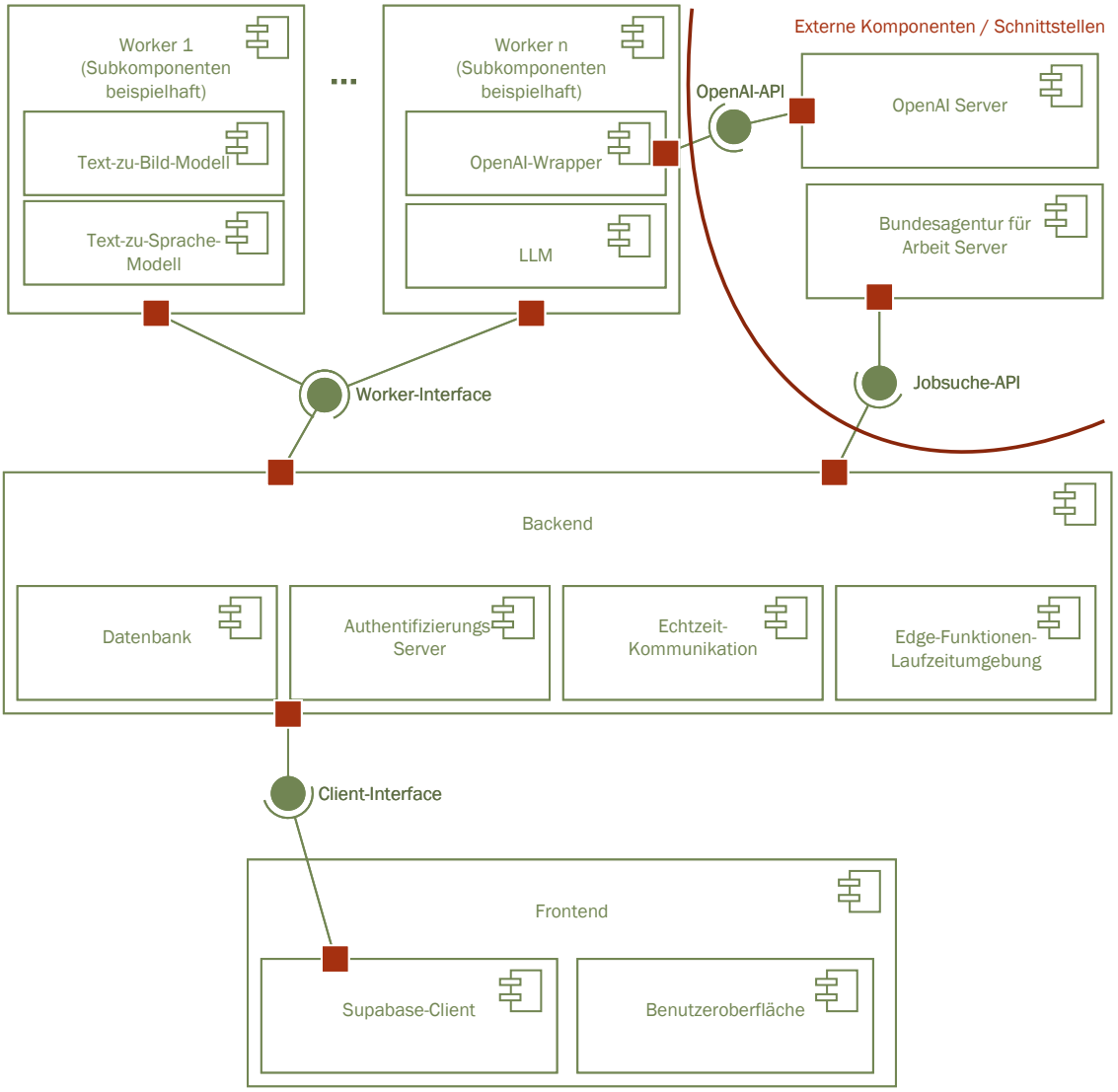


Abbildung 3 Komponentendiagramm

5.2 Entwicklung des Backends

Nachdem im Design-Prozess festgestellt wurde, welche Voraussetzungen das Backend erfüllen soll und welche Aufgaben es hat, fiel die Entscheidung auf die Nutzung einer existierenden Sammlung von Softwarewerkzeugen namens *Supabase* (*DE26*). Hauptaufgabe hierbei ist das Bereitstellen einer relationalen Datenbank per *PostgreSQL*[11]. In den Tabellen dieser Datenbank werden alle Daten und Artefakte gespeichert, die während der Generierung eines Videoprojekts entstehen (*DE5*, *DE6*, *DE7*, *DE9*). Diese Daten sind in einer hierarchischen Ordnung organisiert. In Abbildung 4 werden diese Entitäten mit ihren relevanten Attributen und Beziehungen zueinander in einem ER-Diagramm dargestellt.

[11] PostgreSQL Homepage: postgresql.org

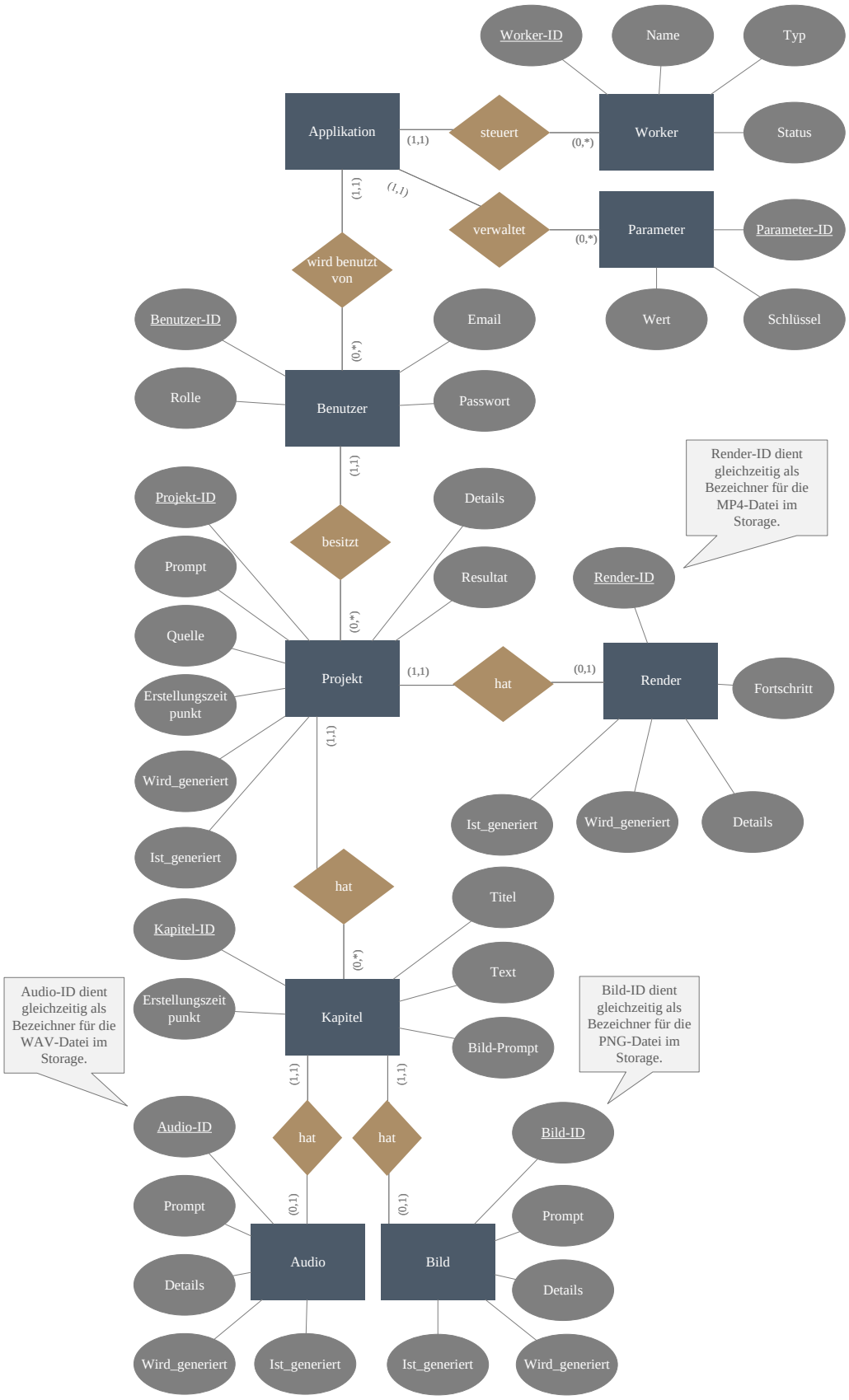


Abbildung 4 Datenmodell als ER-Diagramm

Wird vom Benutzer (Eintrag in der Tabelle *Benutzer*) ein neues Videoprojekt erstellt, korrespondiert dies zu einem neuen Projekt (Eintrag in der Tabelle *Projekt*). Bei der Generierung des Storyboards für dieses Projekt werden mehrere Kapitel (Einträge in der Tabelle *Kapitel*) erstellt (*DE4*). Zu jedem dieser Kapitel gehört dann genau ein Hintergrundbild (Eintrag in der Tabelle *Bild*) und eine vorgelesene Audiospur (Eintrag in der Tabelle *Audio*). Nachdem alle Kapitel mit den zugehörigen Medien generiert wurden, kann es zu jedem Projekt ein fertig gerendertes Kurzvideo in Story-Form (Eintrag in der Tabelle *Render*) geben (*DE8*).

Um dem Nutzer möglichst viel Kontrolle über die erstellten Videos zu geben, wurden alle Schritte im Generierungsprozess möglichst stark parametrisiert (*DE11*). Alle diese veränderbaren Parameter (z. B. die Meta-Parameter der verschiedenen KI-Modelle) werden in der Tabelle *Parameter* in der Datenbank als Schlüssel-Wert-Paar gespeichert.

Außerdem soll der Nutzer zu jedem Zeitpunkt eine Übersicht über die verschiedenen Worker und deren Status haben. Dazu hat jeder Worker einen Eintrag in der Tabelle *Worker*, in der neben dem Namen und Typ auch dynamisch der Status (z. B. `offline`, `online` oder `working`) gesetzt werden kann (*DE14*).

Neben dem Speichern von Daten und Dateien, gehört auch die Authentifizierung der Nutzer zu einer der Hauptaufgaben des Backends (*DE13*). Dazu wird der *Supabase-Authentifizierungs-Service* (basierend auf *GoTrue*[12]) selbst gehostet. Diese Software basiert auf dem *OAuth2*-Standard und stellt eine JSON Web Token (JWT) basierte API bereit, durch die Nutzer und Berechtigungen gemanagt werden können. In dem Softwareartefakt wird die Authentifizierung des Nutzers benötigt, um ihm seine erstellten Projekte und deren Inhalte zuordnen zu können. Ein Nutzer muss sich mit E-Mail-Adresse und Passwort authentifizieren, damit er ein neues Projekt anlegen kann und Zugriff auf seine existierenden Projekte erhält (*DE13*). Dazu werden in der Datenbank *Row Level Security (RLS)* Regeln angewendet, durch die granular für jeden Nutzer und jede Anfrage entschieden werden kann, ob der jeweilige Lese- oder Schreibzugriff erlaubt ist.

Mit dem Dienst *Supabase Realtime*[13] stellt das Backend eine Funktionalität bereit, durch die das Frontend und alle Worker in Echtzeit Informationen austauschen können (*DE15*). Mit bestimmten *Listenern* in der Postgres-Datenbank können Änderungen direkt und ohne *Polling* über Websocket-Technologien an die verschiedenen Abonnenten gesendet werden. In dem Softwareartefakt wird dies genutzt, um dem Nutzer in Echtzeit eine Übersicht über den Generierungsfortschritt des Videos zu ermöglichen. Nach jedem Schritt, wie der Generierung eines Bildes, einer Audiospur oder eines neuen Kapitels, wird das neue Ar-

[12] GoTrue Repository: github.com/netlify/gotrue
[13] Supabase Realtime Repository: github.com/supabase/realtime

tefakt innerhalb weniger Sekunden im Frontend angezeigt. So kann der Nutzer noch während der Generierung der restlichen Mediendateien damit beginnen, die ersten generierten Inhalte zu prüfen und ggf. anzupassen (*DE5*).

Zuletzt muss das Backend noch eine Möglichkeit bereitstellen, eigenen JavaScript-Code seitens des Servers auszuführen. Dies wird beispielsweise zur Abfrage von Daten von der *Jobsuche-API* der *Bundesagentur für Arbeit* benötigt (*DE2*). Zur Umsetzung wird ein weiterer Dienst, die *Self-Hosted Functions* von *Supabase*, verwendet, der eine *Deno*[14] Laufzeitumgebung bereitstellt, auf der beliebige JavaScript- und TypeScript-Funktionen ausgeführt werden können. Um alle diese Microservices möglichst einfach und wartbar auf dem Server bereitzustellen, werden die containerisierten Varianten genutzt und diese mit *Docker* ausgeführt (*DE27*). Dazu werden alle Dienste und deren Parameter in einer *Docker-Compose-Datei*[15] definiert und gemeinsam gestartet.

Um die *Jobsuche-API* der *Bundesagentur für Arbeit* nutzen zu können (*DE2*), wird eine selbstgehostete serverseitige JavaScript-Funktion genutzt. Nachdem überprüft wurde, dass der Aufruf von einem authentifizierten Nutzer stammt, wird die *Job-ID*, der eindeutige Identifikator für eine Stelle in der *Jobsuche-API*, aus den Anfrageparametern ausgelesen. Über eine GET-Anfrage an den *jobdetails* Endpunkt der API werden zu der *Job-ID* alle Details zur Stelle abgefragt und als JSON-Objekt geparst. Aus der großen Menge an Schlüssel-Wert-Paaren werden die in Tabelle 5 dargestellten Informationen für den Prompt zur Generierung des Kurzvideos genutzt.

Attributname	**Beschreibung**	**Beispiel**
titel	Der Titel der Stellenanzeige	„Ausbildung als Orthopädieschuhmacher (m/w/d)“
arbeitgeber	Die Bezeichnung des Arbeitgebers	„Thönnissen GmbH“
arbeitsorte	Eine Liste von möglichen Arbeitsorten, aus der nur der primäre Ort (erster Eintrag) genutzt wird.	„Koblenz am Rhein“
beruf	Die genaue Berufsbezeichnung	„Orthopädieschuhmacher/in“
stellenbeschreibung	Die Stellenbeschreibung in Textform	„Wir suchen für die Ausbildung [...]“

Tabelle 5 Relevante Parameter der *Jobsuche-API*

[14] Deno Homepage: deno.com
[15] Docker Compose Dokumentation: docs.docker.com/compose

5.3 Entwicklung der Worker

Eine der wichtigsten Anforderungen an das Softwareartefakt besteht darin, die einzelnen Schritte zur Videogenerierung getrennt voneinander und auf verschiedenen Systemen verteilt ausführen zu können (*REQ14*). Um diese Anforderung umzusetzen, wurden verschiede Designentscheidungen getroffen, die zu einer Architektur mit Workern geführt haben (*DE12*, *DE14*). Diese Worker übernehmen verschiedene Aufgaben, können in beliebiger Anzahl auf beliebigen Systemen ausgeführt werden und stehen in ständigem Kontakt mit dem Backend (*DE15*). Die generalisierte Funktionsweise, die die Grundlage für die verschiedenen Worker-Arten bildet, wird im Flussdiagramm in Abbildung 5 visualisiert.

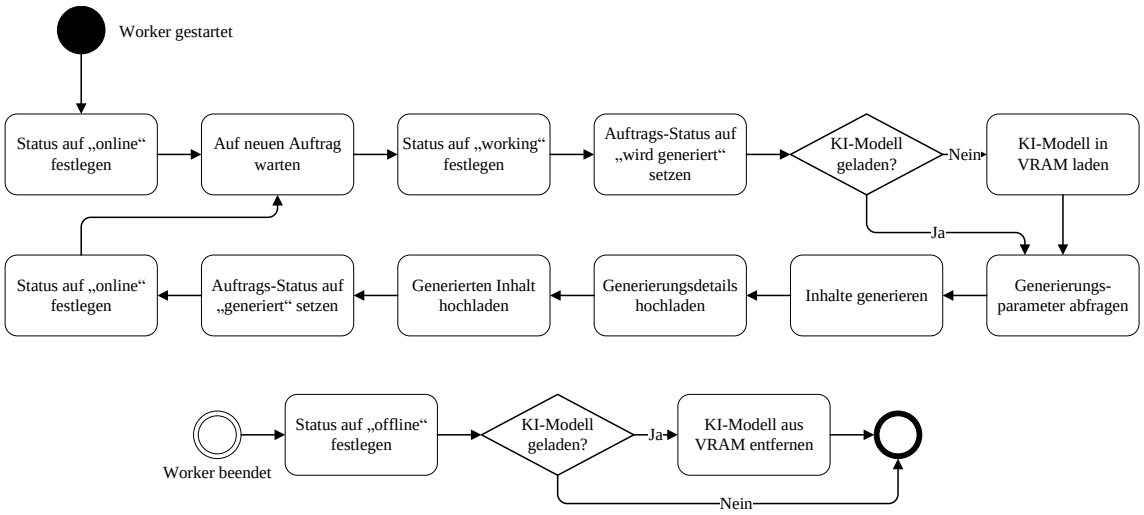


Abbildung 5 Flussdiagramm eines generalisierten Workers

Sobald ein Worker gestartet wird, sendet er eine Nachricht an das Backend, sodass sein Status auf `online` gesetzt wird. Ab dann wartet der Worker auf neue Aufträge. Sobald ein neuer Auftrag (i.e. ein Eintrag in der entsprechenden Tabelle) zur Datenbank hinzugefügt wird, fragt der Worker die nötigen Daten ab und startet die Generierung (*DE14*). Während der Generierung wird ein entsprechendes Attribut im Eintrag des Auftrags gesetzt, sodass die anderen Worker wissen, dass dieser Auftrag bereits bearbeitet wird. Nach der Generierung werden der erstellte Inhalt und Details zur Generierung (z. B. der exakte genutzte Prompt) zurück an das Backend gesendet und in die Datenbank eingetragen (*DE5*). Diese Änderung in der Datenbank triggert auch automatisch das entsprechende Update im Frontend, sodass der Nutzer die generierten Inhalte sofort angezeigt bekommt. Nachdem während des Generierungsprozess der Worker-Status von `online` auf `working` gesetzt wurde, wird dieser auf `online` zurückgesetzt und der Worker wartet auf den nächsten Auftrag. Handelt es sich um einen Worker, der ein KI-Modell in den Grafikspeicher laden muss, um Inhalte zu generieren, gibt es im Ablauf entsprechende Stellen, an denen das

Modell bei Bedarf geladen oder aus dem Grafikspeicher entfernt wird. Sobald der Prozess ein Signal erhält, das ihn beendet, wird als letzte Aktion der Worker-Status auf `offline` gesetzt.

Entwicklung des LLM-Workers. Der LLM-Worker wird genutzt, um im ersten Schritt aus den Eingaben des Nutzers das Storyboard zu generieren. Als Eingabe erhält er einen Auftrag, der alle Informationen zu dem zu generierenden Video enthält (*DE1*). Als Ausgabe wird eine Liste im JSON-Format erwartet, die die Beschreibung für mehrere (vier bis fünf) Kapitel enthält und beschreibt, wie das Kurzvideo in Story-Form aufgebaut sein soll (*DE3*). Jedes dieser Kapitel besteht aus einem Titel, einem Text und einem Prompt für eine zum Inhalt passende Hintergrundgrafik (*DE4*). Dieses Schema für ein Storyboard wird in Form eines *Pydantic*[16] Modells, wie in Abbildung 6 gekürzt zu sehen, angegeben.

```
1 class Storyboard(BaseModel):
2   chapters: List[Chapter] = Field(..., title="List of 4 to 5 chapters
        of the video")
3
4 class Chapter(BaseModel):
5   title: str = Field(..., title="A German title for the chapter")
6   text: str = Field(..., title="A German text for the chapter")
7   image_prompt: str = Field(..., title="An English prompt for a fitting
        background image")
```

Abbildung 6 Pydantic Datenmodell des Storyboards

Neben den Informationen für das zu generierende Kurzvideo werden außerdem weitere Parameter aus der Datenbank abgefragt (*DE11*). Dazu gehören unter anderem die Parameter `prompt_prefix` und `prompt_suffix`, also feste Zeichenketten, die an die variable Nutzereingabe angehangen werden, um das LLM möglichst nah an die gewünschten Ausgaben heranzuführen. Diese Parameter können zur Laufzeit vom Nutzer im Menüpunkt *Einstellungen* im Frontend angepasst werden (*DE11*). Da das eigentliche Prompt-Engineering von möglichst optimalen Prompts nicht im Scope dieser Abschlussarbeit liegt, da jeder Anwender eigene Anforderungen an diesen Prompt mitbringt, wird ausschließlich ein beispielhafter Prompt genannt, der sich während der Entwicklung des prototypischen Softwareartefakts als geeignet herausgestellt hat. Daher ist dieser Text als Standardwert für den Parameter `prompt_prefix` vorausgewählt:

> *Generate 4 to 5 chapters for a short video that summarizes the most important information of the following job ad. Start by presenting the company itself and the job position. The video should be short and interesting for younger*

[16] Pydantic Dokumentation: docs.pydantic.dev

audiences, talk directly to the viewer and use the informal 'you' ('Du') instead of 'Sie'.

Mit diesen Angaben, den weiteren Modell-Parametern und dem Hinzufügen des Pydantic-Modells zum Kontext ist es möglich, konsistente Ergebnisse für die Struktur des Kurzvideos, dem Storyboard, zu erzielen.

Zur Text-Generierung erwartet der Worker, dass eine *Ollama*[17] Instanz mit dem als Parameter `llm_parameters.model` festgelegten Sprachmodell auf dem System läuft. Dann wird die Python-Bibliothek *ollama* genutzt, um mit dieser Instanz zu kommunizieren und die Ausgabe des Sprachmodells als Stream weiterzuverwenden.

In der ersten Iteration des Softwareartefakts wurde an dieser Stelle gewartet, bis die Ausgabe des LLMs vollständig empfangen wurde. Diese wurde dann mit der Python-Standardbibliothek *json* geparst und die resultierenden Kapitel an das Backend übermittelt. Dies hatte jedoch zur Folge, dass der Nutzer nach der Erstellung eines Projekts zuerst länger auf die ersten Resultate warten musste und dann alle Kapitel direkt auf einmal angezeigt wurden. Auch die restlichen Generierungsprozesse konnten erst dann gestartet werden, sobald das komplette Storyboard fertiggestellt wurde. Um die Gesamtzeit der Videogenerierung zu verringern und durch schrittweises Einblenden der generierten Kapitel die Nutzererfahrung zu verbessern, wird in der zweiten Iteration des LLM-Workers jeder Ausgabetoken direkt verarbeitet und so jedes Kapitel aus dem Ausgabe-Stream des LLMs sofort geparst, nachdem es vollständig ist. So können die Kapitel schon an das Backend übermittelt werden, während die restliche Ausgabe des LLMs noch ermittelt wird (*DE7*). Das hat zur Folge, dass der Nutzer bereits nach wenigen Sekunden Wartezeit das erste Kapitel angezeigt bekommt und auch die weiteren Generierungsprozesse für dieses Kapitel schon gestartet werden können, während die anderen Kapitel noch generiert werden.

Entwicklung des Text-to-Image-Workers. Der Text-to-Image-Worker wird genutzt, um die Hintergrundbilder zu den einzelnen Kapiteln des Storyboards zu generieren. Als Eingabe erhält der Worker dazu den jeweiligen Prompt, den der LLM-Worker bei der Generierung des Storyboards für das Kapitel festgelegt hat (*DE6*). In der ersten Iteration wurde dieser Prompt, wie auch der Titel und der Text des Kapitels, in Deutsch generiert. Da das verwendete generative Text-zu-Bild-Modell *Stable Diffusion 3 Medium* jedoch deutlich bessere Ergebnisse für englische Prompts erzielte, wird der LLM-Worker in der zweiten Iteration angewiesen, den `image_prompt` in Englisch zu generieren. Auch hier werden vor der Generierung noch weitere Parameter aus der Datenbank abgefragt, zu denen u. a. die Modell-Parameter für das Diffusionsmodell gehören, aber auch wieder ein

[17] Ollama Homepage: ollama.com

Präfix und Suffix, mit denen der Prompt ergänzt wird. Auf diese Weise kann ein einheitlicher Stil festgelegt werden, sodass alle generierten Bilder eines Kurzvideos in Story-Form beispielsweise fotorealistisch generiert werden und zueinander passen (*DE11*). Für die eigentliche Generierung der Bilddatei nutzt der Worker die `generate` Methode der `StableDiffusion`-Klasse. Diese Klasse gehört zu einem selbst entwickelten Python-Modul, das von der eigentlichen Ausführung der Diffusions-Pipeline abstrahiert. Durch diese Form von *Information Hiding* kann das eigentliche KI-Modell zukünftig mit wenig Aufwand ausgetauscht werden und die Wartbarkeit des Softwareartefakts wird erhöht (*DE14*). Nach der Generierung des Bildes werden die Generierungsdetails und die Bilddatei ans Backend übermittelt. Diese Änderung wird anschließend direkt an das Frontend weiter propagiert, sodass hier innerhalb weniger Sekunden das generierte Bild im zugehörigen Kapitel angezeigt wird (*DE12*).

Entwicklung des Text-to-Speech-Workers. Der Text-to-Speech-Worker (TTS-Worker) wird genutzt, um aus dem Inhalt der Kapitel in Textform eine vorgelesene Audiodatei zu generieren. Als Eingabe erhält er dazu den jeweiligen Text eines Kapitels aus dem Storyboard und lädt weitere Generierungsparameter aus der entsprechenden Tabelle der Datenbank (*DE7*). Für die eigentliche Generierung der Audiodatei nutzt der TTS-Worker die `generate` Methode der `Bark`-Klasse. Diese Klasse gehört zu einem selbst entwickelten Python-Modul, das von der eigentlichen Ausführung des Transformer-Modells *Bark* abstrahiert. Durch das *Information Hiding* ist es auch hier möglich, das Modell einfach auszutauschen, solange das festgelegte Interface verwendet wird (*DE14*). In der ersten Iteration dieses Workers war die Sprechgeschwindigkeit der generierten Audiospuren eher langsam. Dadurch haben die resultierenden Kurzvideos in Story-Form ihren kurzweiligen Charakter verloren. Die Anpassung verschiedener Parameter des *Bark*-Modells konnte zwar eine Besserung erzielen, die aber weiterhin nicht als optimal empfunden wurde. Als pragmatische Lösung in der zweiten Iteration des Workers wird nun die generierte Audiodatei in das Kommandozeilentool *FFmpeg*[18] geladen und hier der Filter `atempo` angewendet, der die Geschwindigkeit der Audiospur anpasst. Der Faktor, um den das Sprechtempo erhöht werden soll, ist parametrisiert und über das Frontend anpassbar (*DE11*). Als Standardwert wird der Faktor 1,3 (Tempoerhöhung um 30%) verwendet. Mit diesem Wert werden Kurzvideos generiert, deren Tempo mit den meisten Kurzvideos in Story-Form auf Social Media vergleichbar ist.

Nach der Generierung und Modifikation der Audiodatei wird sie gemeinsam mit den Generierungsdetails in der Datenbank gespeichert und so auch im Frontend abspielbar gemacht (*DE7*).

[18] FFmpeg Homepage: ffmpeg.org

Entwicklung des Rendering-Workers. Der Rendering-Worker hat die Aufgabe, aus den in den vorherigen Schritten generierten Texten, Bildern und Audiospuren ein fertiges Kurzvideo in Story-Form im MP4-Format zu erstellen (*DE8*). Im Gegensatz zu den anderen Workern werden dazu keine generativen KI-Modelle genutzt, sondern die Python-Bibliothek *MoviePy*[19], die als Python-Wrapper für die verschiedenen Funktionalitäten des Tools *FFmpeg* fungiert. Als Eingabe erhält dieser Worker die `ID` des zu rendernden Projekts, mit der alle zugehörigen Mediendateien vom Backend heruntergeladen werden. Dann wird über alle Kapitel des Storyboards iteriert und das jeweilige Hintergrundbild mit der Audiospur des Kapitels unterlegt. Der Titel des Kapitels wird als Text in der oberen Hälfte des Bildes eingeblendet. Um mit dem einzelnen Hintergrundbild trotzdem einen Bewegbild-Effekt zu erzielen, wurden vier verschiedene Animationspfade hinterlegt, die für jedes Kapitel zufällig ausgewählt werden (*DE17*). So werden laufend neue Ausschnitte des Bildes sichtbar, mit dem Ziel, die Aufmerksamkeit des Betrachters länger zu halten. Auch die Einblendung des Titels wird farblich hervorgehoben und in einem festgelegten Bereich in einem zufälligen Winkel rotiert, sodass die Gesamtästhetik noch mehr an die aus Social Media bekannten Stories erinnert (*DE18*). Zuletzt wird noch der vorgelesene Text in einer kleineren Schriftart am unteren Bildschirmrand eingeblendet, sodass der Betrachter den gesprochenen Text mitlesen kann (*DE19*). Ein beispielhafter Ausschnitt, der die grafischen Elemente verdeutlicht, wird in Abbildung 7 gezeigt.

Abbildung 7 Standbild aus einem generierten Kurzvideo

[19] MoviePy Dokumentation: zulko.github.io/moviepy

Nach einem Hinweis, dass die Inhalte des Kurzvideos durch KI generiert wurden (*DE20*), werden auf diese Art und Weise alle Kapitel nacheinander abgespielt und als MP4-Datei gerendert. Um dem Nutzer im Frontend den prozentualen Fortschritt des Rendering-Prozesses anzuzeigen, wurde die `ProgressBarLogger`-Klasse der *MoviePy*-Bibliothek umimplementiert, sodass jede Änderung im Fortschritt in die gemeinsame Datenbank geschrieben wird und somit vom Frontend abonniert werden kann. Zum Schluss wird die fertige Videodatei mit den Generierungsdetails ans Backend übermittelt und dem Nutzer angezeigt (*DE9*).

Entwicklung der alternativen OpenAI-Worker. Neben dem Einsatz der quelloffenen generativen KI-Modelle auf eigenem Hosting, soll das Softwareartefakt auch kommerzielle Modelle zur Videogenerierung nutzen können. Dazu wird die vorhandene Worker-Architektur genutzt, und es werden weitere Worker entwickelt, die identisch zu den anderen Workern ausgeführt werden, aber im Hintergrund die verschiedenen APIs von *OpenAI* ansprechen (*DE29*). Zur Kommunikation mit der *OpenAI*-API wird das Python-Modul *openai* genutzt, das als Python-Wrapper den Zugriff auf die verschiedenen Modelle mit wenigen Zeilen Code ermöglicht. Bei der Nutzung des *GPT-4o*-Modells kann beim Aufruf der `create`-Funktion neben dem eigentlichen Prompt auch das *Pydantic*-Schema für das Storyboard als Zielschema angegeben werden. Dadurch müssen weniger Einschränkungen in natürlichsprachlicher Form im Prompt angegeben werden. Somit werden gesonderte Prompt-Parameter für die Nutzung der OpenAI-Worker verwendet, die analog zu den anderen Parametern im Frontend angepasst werden können (*DE11*).

Entwicklung eines Schedulers zur pseudoparallelen Ausführung der Worker. Während der Entwicklung des Softwareartefakts war es häufig ein Problem, dass aufgrund ihrer Größe nicht alle KI-Modelle der verschiedenen Worker zur gleichen Zeit in den Grafikspeicher geladen werden konnten. In der ersten Iteration wurden daher z. B. erst alle Bilder mit dem Text-to-Image-Worker generiert und erst im Anschluss der TTS-Worker gestartet, um alle Audiospuren zu generieren. In der zweiten Iteration wird nun ein Python-Skript als Workaround verwendet, das die beiden Modelle mit möglichst wenigen Modellwechseln pseudoparallel ausführen kann. Dazu behält der Scheduler einen ständigen Überblick über die offenen Auftragsarten und entscheidet, wann ein Modell in den Grafikspeicher geladen werden soll und wann es wieder entfernt wird.

Nach den Ausführungen über die verschiedenen Worker-Komponenten wird im folgenden Abschnitt die Entwicklung des Frontends beschrieben.

5.4 Entwicklung des Frontends

Für die Entwicklung des Frontends wird *Sveltekit* als Framework und TypeScript als Sprache verwendet (*DE25*). Als Metaframework von *Svelte* wird bei *Sveltekit* zwischen serverseitig gerenderten und clientseitig gerenderten Komponenten unterschieden. Da lediglich das clientseitige *User Interface (UI)* implementiert werden soll und die komplette serverseitige Logik bereits durch das in Kapitel 5.2 beschriebene Backend bereitsteht, wird auf serverseitige Komponenten verzichtet (*DE12*). Durch das Setzen der Option `ssr = false` in der obersten Layout-Datei wird die komplette Applikation nur clientseitig gerendert. Texte in der Benutzeroberfläche werden in Englisch angezeigt, damit die Software auch von nicht deutschsprachigen Nutzern eingesetzt werden kann. Diese Sprachauswahl ist komplett unabhängig von der Sprache der generierten Kurzvideos, da diese allein durch die konfigurierbaren Prompts bestimmt wird. Im Rahmen dieser Abschlussarbeit werden deutsche Videos generiert. Unabhängig davon wird das LLM angewiesen, Prompts zur Bildgenerierung auf Englisch zu formulieren, da beim Testen des Prototyps so bessere Ergebnisse mit den Diffusionsmodellen erzielt werden konnten. Innerhalb des Frontends können über Routen die verschiedenen Komponenten erreicht werden, aus deren Gesamtheit die clientseitige Applikation besteht. Im Folgenden wird die Entwicklung der relevantesten Komponenten beschrieben.

Entwicklung der Login-Komponente. Weil die Applikation nur für authentifizierte Nutzer zugänglich sein soll (*REQ13*), wird ein neuer Nutzer als Erstes auf die `/login` Route umgeleitet, damit er sich mit E-Mail-Adresse und Passwort anmelden kann (*DE13*). Dazu wird eine Komponente mit zwei Eingabefeldern und einem Button zum Anmelden genutzt, wie in Abbildung 8 gezeigt.

Abbildung 8 Screenshot der Login-Komponente

Bei der Anmeldung werden die Angaben an die *Supabase*-Authentifizierungsfunktion `signInWithPassword` übergeben und der Nutzer wird bei erfolgreicher Authentifizierung zum Hauptmenü unter der Route `/app` weitergeleitet.

Da es sich beim Frontend um eine reine clientseitige Web-Applikation handelt, ist es die Aufgabe des Backends, bei jedem künftigen API-Aufruf die Authentifizierung und Autorisierung des Nutzers zu überprüfen. Für eine bessere Nutzererfahrung wird außerdem durch eine Layout-Datei für die `/app`-Route bei jedem Seitenzugriff überprüft, ob der Nutzer einen gültigen JWT hat. Sollte dies nicht der Fall sein, wird er sofort auf die Login-Route umgeleitet. Wenn ein bereits authentifizierter Nutzer die `/login`-Route aufruft, wird er automatisch zum Hauptmenü für authentifizierte Nutzer weitergeleitet.

Entwicklung des Hauptmenüs. Im Hauptmenü kann der Nutzer entweder ein neues Projekt erstellen oder ein existierendes Projekt auswählen, wie in Abbildung 9 gezeigt. Dazu wird bei jedem Aufruf eine Liste der letzten 20 Projekte des Nutzers aus der Datenbank abgefragt und in einer Liste angezeigt.

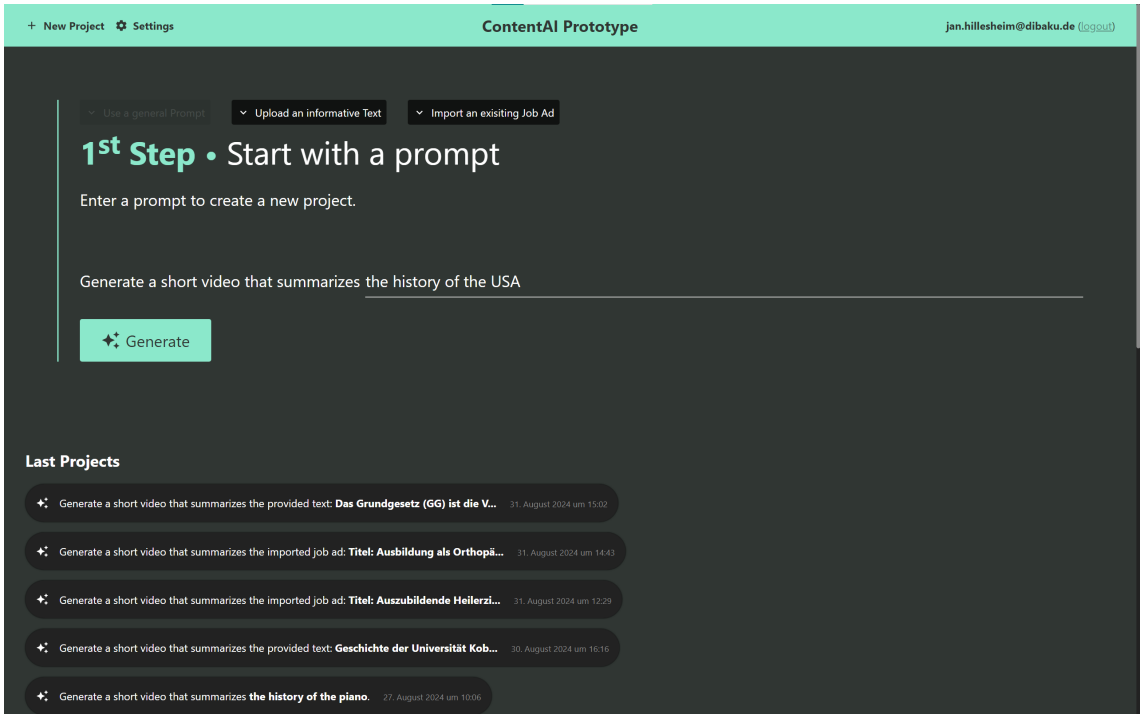


Abbildung 9 Screenshot der Hauptmenü-Komponente

Über verschiedene Schaltflächen hat der Nutzer drei Möglichkeiten, ein neues Projekt zu erstellen (*DE1*).

Videogenerierung aus einem Prompt. Hier kann ein neues Projekt gestartet werden, indem der Nutzer einen einzigen Satz als Prompt zur Videogenerierung eingibt. Dazu kann

der Nutzer über ein Textfeld den Satz „Generate a short video that summarizes . . . “ vervollständigen. Hierbei ist zu beachten, dass das resultierende Video ausschließlich Informationen enthält, die das genutzte LLM während des Trainingsprozesses ‘gelernt‘ hat. Die hierfür zu nutzenden Bedienelemente werden in Abbildung 9 dargestellt.

Videogenerierung aus einem informativen Text. Hier hat der Nutzer die Möglichkeit, einen beliebigen Text (im Rahmen der Kontext-Größe des genutzten LLMs) in das mehrzeilige Eingabefeld einzufügen, wie in Abbildung 10 zu sehen. Aus diesem Text werden die relevanten Informationen extrahiert und so ein informatives Kurzvideo in Story-Form erstellt, dessen Inhalte auch über das ‘Wissen‘ des LLMs hinausgehen können.

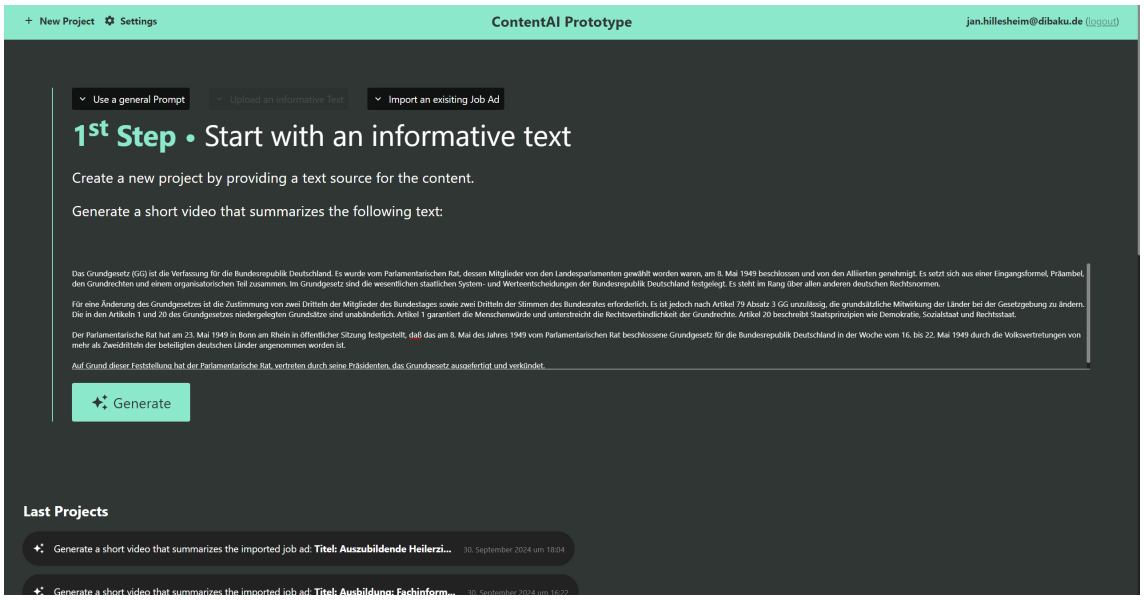


Abbildung 10 Screenshot der Projekterstellung aus einem Text

Videogenerierung aus einer Stellenanzeige. Während die vorherigen Erstellungsmethoden für Kurzvideos aller Art verwendet werden können, wird hier ein Kurzvideo in Story-Form aus einer Ausbildungs-Stellenanzeige in strukturierter Form generiert. Dazu kann der Nutzer den Link oder die ID zur gewünschten Stellenanzeige auf der Jobsuche-Website der *Bundesagentur für Arbeit* angeben (siehe Abbildung 11). Die benötigten Informationen werden über die *Jobsuche-API* von den Servern der *Bundesagentur für Arbeit* abgefragt (*DE2*) und zur Überprüfung angezeigt (siehe Abbildung 12).

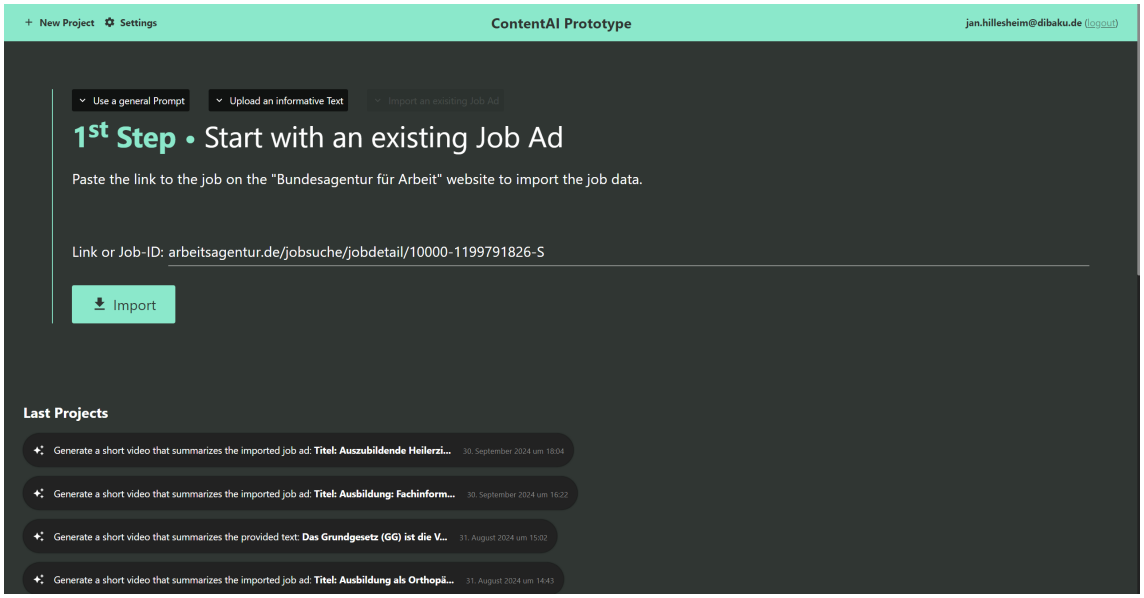


Abbildung 11 Screenshot der Projekterstellung aus einer Stellenanzeige

Abbildung 12 Screenshot der Detailansicht einer Stellenanzeige

Unabhängig von der gewählten Projektart wird bei der Erstellung ein neuer Datenbankeintrag in der Projekt-Tabelle angelegt und dem Projekt wird eine eindeutige ID zugeordnet (*DE5*). Während die ID zurück ans Frontend übermittelt wird, damit der Nutzer automatisch auf die Projekt-Seite weitergeleitet werden kann, triggert die Erstellung des Datenbankeintrages den LLM-Worker, der im Hintergrund mit der Erstellung des Storyboards beginnt (*DE1*).

Entwicklung der Storyboard-Komponente. Jedes Projekt erhält bei der Erstellung eine eindeutige ID in Form einer *UUIDv4*, sodass über die Route `/app/p/{ID}` auf das Storyboard des Videos zugegriffen werden kann. Diese Projektseite enthält, wie in Abbildung 13 zu sehen, neben den relevanten Informationen zum Projekt und einer Statusanzeige für die Worker, auch das eigentliche Storyboard, über das die Videoerstellung geplant wird.

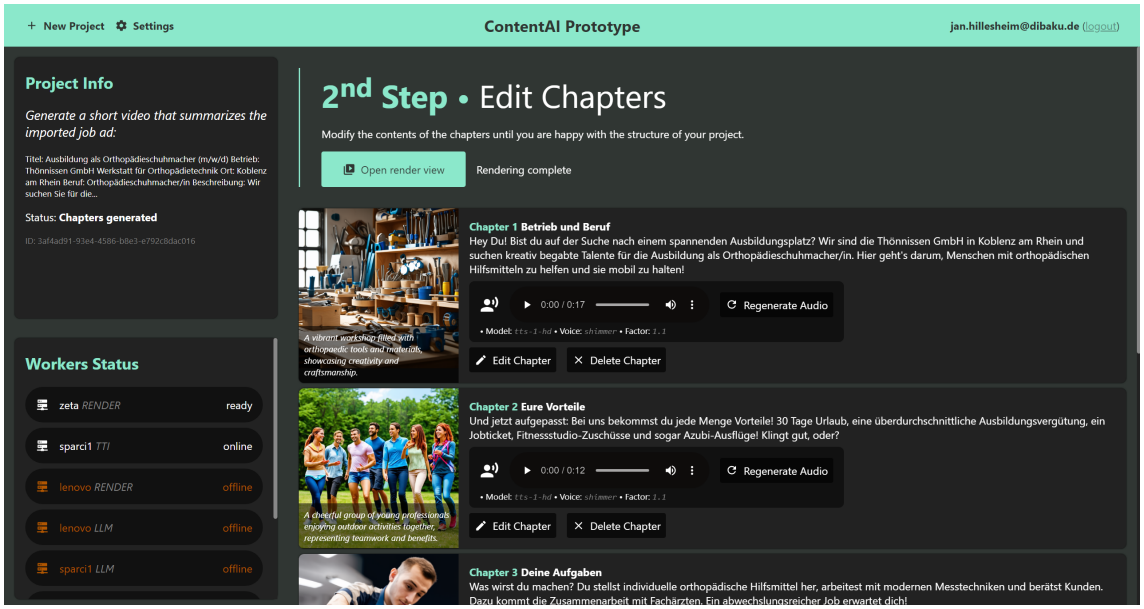


Abbildung 13 Screenshot der Storyboard-Komponente

Das Storyboard ist eine Liste von Kapiteln des finalen Kurzvideos, die jeweils als Container farblich vom Hintergrund abgehoben sind. Neben der vorgegebenen Nummerierung der Kapitel (*Chapter 1* bis *Chapter n*) wird der vom LLM generierte Titel und der vorzulesende Text angezeigt (*DE4*). Für jedes Kapitel wird außerdem das quadratische Hintergrundbild und der zugehörige Prompt, aus dem es generiert wurde, dargestellt (*DE6*). Bewegt man den Mauszeiger über das Bild, werden dessen Generierungsdetails (z. B. die verwendeten Parameter des Diffusionsmodells, wie `steps` und `guidance`) sichtbar und eine Schaltfläche *Regenerate* erscheint, um das Bild erneut generieren zu lassen. Unter dem vorzulesenden Text befindet sich das Bedienelement der generierten Audiospur (*DE7*). Hier kann der gesprochene Text abgespielt werden und dessen Generierungsde-

tails wie Modell, Stimme und Geschwindigkeitsfaktor werden angezeigt. Außerdem kann auch hier über die entsprechende Schaltfläche die Neugenerierung gestartet werden.

Während die Kapitel vom LLM-Worker generiert werden, erscheint ca. alle fünf Sekunden (abhängig von LLM-Auswahl und genutzter Hardware) ein weiteres Kapitel auf dem Storyboard, bis alle 4-5 Kapitel (kann im entsprechenden Prompt verändert werden) generiert wurden. Diese Darstellung der Kapitel in Echtzeit, direkt nach der Fertigstellung ihrer Generierung, funktioniert durch das Abonnieren von neuen Einträgen in der *Kapitel*-Tabelle der Datenbank, die dem geöffneten Projekt zugeordnet sind (*DE15*). Außerdem werden zu jedem neuen Kapitel auch die zugehörigen Änderungen in den *Audio*- und *Bild*-Tabellen abonniert. So können in der Benutzeroberfläche Platzhalter (z. B. der Bild-Prompt anstatt des tatsächlichen Bildes) angezeigt werden, die automatisch durch das generierte Asset ersetzt werden, sobald es von dem entsprechenden Worker erstellt wurde (*DE15*). Durch die feinere Einteilung von wartenden Aufträgen in `queued` (in der Warteschlange) und `generating` (wird gerade generiert), werden außerdem Icons, Farben und Animation der Platzhalter so angepasst, dass der Nutzer genau sieht, woran das System gerade arbeitet und welche Generierungsprozesse als nächstes anstehen (*DE16*).

Jedes Kapitel kann vom Nutzer über die Schaltflächen unter dem Audiobedienelement gelöscht und bearbeitet werden. Zur Bearbeitung werden die Daten des Kapitels in Eingabefeldern angezeigt und können vom Nutzer editiert werden, wie in Abbildung 14 gezeigt (*DE5*).

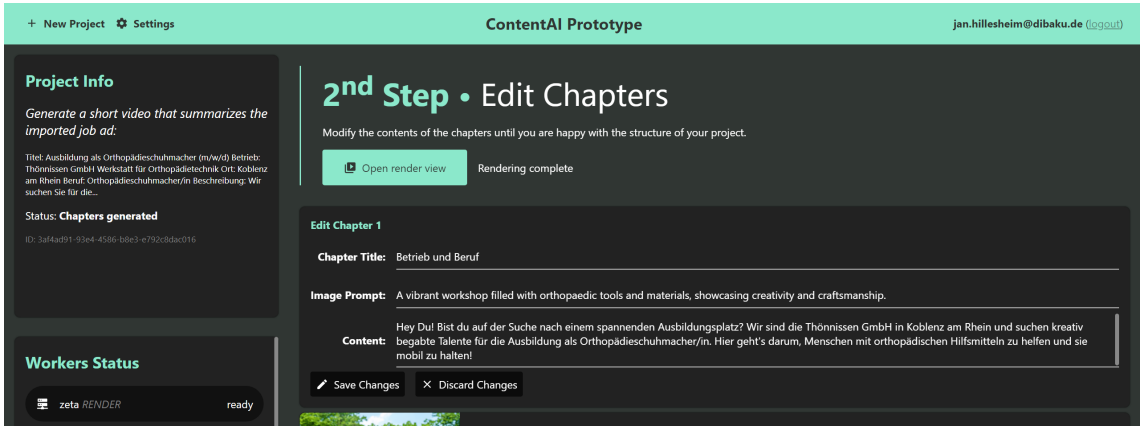


Abbildung 14 Screenshot der Komponente zur Bearbeitung eines Kapitels

Sobald der Nutzer die Änderungen speichert, wird geprüft, ob Änderungen am Hintergrundbild-Prompt oder dem Inhalt des Kapitels getätigt wurden. Ist dies der Fall, wird automatisch der Bildgenerierungsprozess bzw. der Audiogenerierungsprozess für das gewählte Kapitel mit den veränderten Eingaben gestartet (*DE9*).

Das Element zur Statusanzeige der Worker, links unten in der Projektansicht, ist eine eigene eingebundene Komponente. Diese iteriert zum Zeitpunkt der Erstellung über die Liste der Worker in der Datenbank und abonniert deren Status-Felder. In der Auflistung werden so zu jedem Worker der in der Worker-Konfiguration festgelegte Name, der Worker-Typ und der Live-Status angezeigt und durch verschiedene CSS-Klassen, basierend auf dem Status, intuitiv hervorgehoben. Beispielsweise wird ein Worker, der gerade einen Auftrag bearbeitet (Status: `working`), türkis pulsierend angezeigt, um auf den laufenden Vorgang hinzuweisen (*DE14*).

Sobald alle Kapitel, Grafiken und Audiodateien generiert wurden und der Nutzer alle gewünschten Änderungen am Storyboard durchgeführt hat, kann der Videogenerierungsprozess per Klick auf *Render chapters as video* gestartet werden (*DE8*). Es erfolgt eine Weiterleitung zur Resultat-Komponente.

Entwicklung der Komponente zum Anzeigen des Resultats. Sobald das Rendering des Videos vom Nutzer gestartet wurde, wird er zur entsprechenden Komponente weitergeleitet, die das gerenderte Video im für Social Media Inhalten gängigen 9:16 Format abspielt (siehe Abbildung 15).

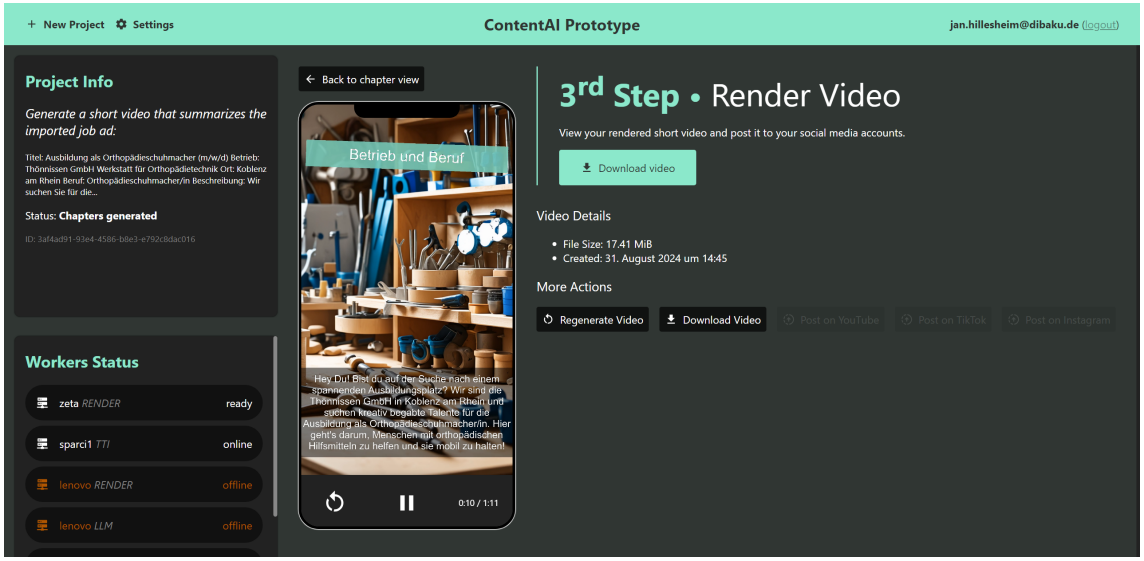


Abbildung 15 Screenshot der Komponente, die das finale Video anzeigt

Solange das Video noch nicht existiert und der Rendering-Prozess läuft, wird in dem schematisch dargestellten Smartphone ein Fortschrittsbalken angezeigt. Dieser kann den exakten Stand des Rendering-Prozesses angeben, da der Rendering-Worker bei jeder Änderung des prozentualen Fortschritts das entsprechende Datenbankfeld verändert. Dieser Wert wird beim Erstellen der Komponente abonniert, sodass der Fortschritt live angezeigt wird (*DE15*).

Beim Abspielen des Videos werden Details zu der erstellten Videodatei angezeigt, und der Nutzer kann das Kurzvideo in Story-Form über Bedienelemente steuern. Schaltflächen ermöglichen das erneute Generieren des Videos und das Herunterladen als MP4-Datei.

Entwicklung der Komponente für Einstellungen. Eine wichtige Designentscheidung für das prototypische Softwareartefakt liegt in der Parametrisierung aller möglicher Generierungsoptionen (*DE11*). Dazu gehören die Parameter der generativen KI-Modelle, die verwendeten Prompt-Templates, Optionen, wie das Kurzvideo gerendert werden soll, uvm. Diese Einstellungen können vom Nutzer in der Web-Applikation über den Navigationspunkt *Einstellungen* aufgerufen werden. In den Einstellungen werden, wie in Abbildung 16 gezeigt, alle anpassbaren Parameter aus der Tabelle *Parameter* in Kategorien eingeteilt und können durch den Nutzer bearbeitet werden.

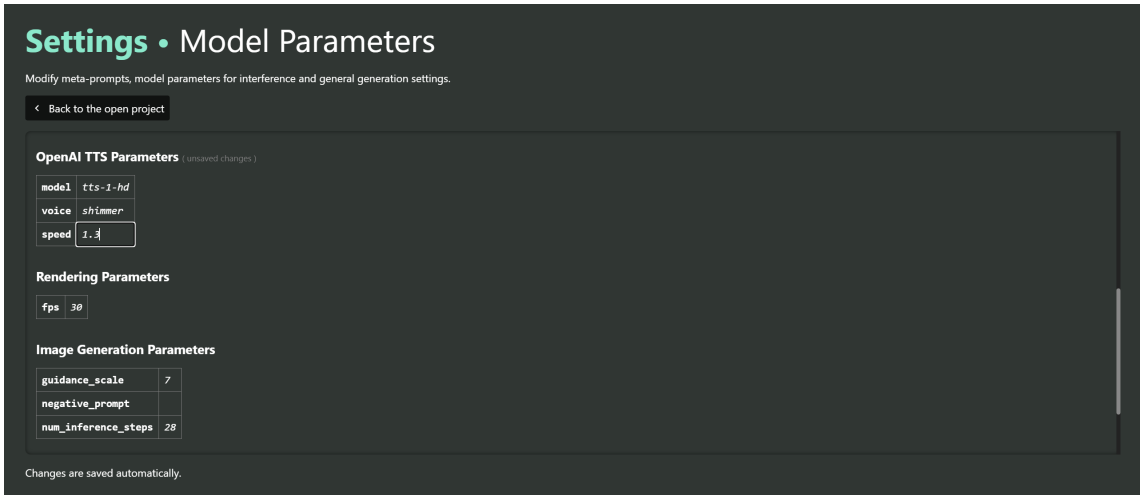


Abbildung 16 Screenshot der Komponente für Einstellungen

Änderungen des Nutzers an dieser Stelle werden automatisch in der Datenbank gespeichert. Durch die asynchrone Architektur der Worker werden diese neuen Optionen auch ohne Neustart direkt mit dem nächsten bearbeiteten Auftrag aus der Tabelle *Parameter* übernommen. So wird es dem Nutzer einfach gemacht, während der Arbeit an einem Projekt die Generierungs-Parameter zu verändern, bis das gewünschte Resultat erreicht ist.

In der Beschreibung der Entwicklung der verschiedenen Komponenten wurden die entsprechenden vorangegangenen Designentscheidungen im Text angegeben. Abschließend wird in Tabelle 6 eine Übersicht dargestellt, wie die aufgestellten Anforderungen jeweils mit den verschiedenen Entwicklungen umgesetzt wurden.

Anf.	**Umsetzung**
REQ1	Die Software kann Stellenanzeigen in strukturierter Form verarbeiten, indem die Informationen im JSON-Format über die Schnittstelle der Bundesagentur für Arbeit importiert werden. Stellenanzeigen in unstrukturierter Form werden verarbeitet, indem die Informationen beim Erstellen eines neuen Projekts in das Freitextfeld eingetragen werden.
REQ2	Die Software kann die relevantesten Informationen aus der Eingabe identifizieren und extrahieren, indem diese als Input für ein LLM genutzt werden. Das wird mit dem entsprechenden Prompt angewiesen, die wichtigsten Informationen zurückzugeben.
REQ3	Die Software generiert aus den extrahierten Informationen ein Storyboard, indem das LLM neben der Nutzereingabe auch ein genaues Schema für dessen Aufbau erhält. So ist die LLM-Ausgabe in ihrer Form deterministisch und kann in diesem strukturierten Format als Storyboard weiterverarbeitet werden. In der Storyboard-UI-Komponente wird dem Nutzer das Storyboard angezeigt.
REQ4	Das Storyboard besteht aus mehreren Kapiteln, die jeweils einen Titel, den textuellen Inhalt und ein Prompt für eine passende Hintergrundgrafik enthalten, da durch das definierte Storyboard-Schema diese Vorgaben an das LLM übergeben werden.
REQ5	Die Software gibt dem Nutzer die Möglichkeit, das Storyboard zu bearbeiten, indem er für jedes Kapitel die Bearbeitungsansicht einschalten und den Titel, Text und Bild-Prompt frei verändern kann. Außerdem kann ein Kapitel mit einem Klick auf den entsprechenden Button auch komplett aus dem Storyboard entfernt werden.
REQ6	Durch den Einsatz eines generativen Diffusions-Modells innerhalb des Text-zu-Bild-Workers kann die Software aus den Prompts für die Hintergrundgrafiken die entsprechenden Bilddateien generieren.
REQ7	Durch den Einsatz eines generativen Audio-Modells innerhalb des Text-zu-Sprache-Workers kann die Software aus den generierten Texten die entsprechenden Audiodateien generieren.
REQ8	Die Software kann aus den Informationen des Storyboards und den generierten Bild- und Audiodateien ein Kurzvideo im MP4-Format rendern, indem der Render-Worker alle Daten für ein Projekt aus der Datenbank mit *FFmpeg* kombiniert und als Videodatei ausgibt.
REQ9	Der Nutzer hat die Möglichkeit, das generierte Video in der Resultat-UI-Komponente anzusehen. Nach Änderungen am Storyboard kann die Videogenerierung mit dem entsprechenden Button neu gestartet werden.
REQ10	Die Software ermöglicht das Herunterladen des fertigen Videos im MP4-Format über den entsprechenden Button in der Resultat-UI-Komponente.

Anf.	Umsetzung
REQ11	Durch die parametrisierte Architektur der Worker-Komponenten und der Speicherung der Parameter in der Datenbank, kann der Nutzer diese über die Einstellungs-UI-Komponente ansehen und verändern.
REQ12	Durch die getrennte Entwicklung des Frontends als Web-Applikation und das Nutzen von Web-Technologien, ist die Software auf allen gängigen Endgeräten ohne Installation nutzbar.
REQ13	Die Software ist nur autorisierten Benutzern zugänglich, da vor der Nutzung die Anmeldeinformationen über die Login-UI-Komponente abgefragt werden. Diese Daten werden genutzt, um über den Authentifizierungsservice und die Anwendung von RLS in der Datenbank sicherzustellen, dass nur genehmigte Aktionen ausgeführt werden können.
REQ14	Durch die modulare Worker-Architektur können die rechenintensiven Generierungsprozesse auf mehrere Systeme verteilt werden.
REQ15	Der Nutzer hat einen permanenten Überblick über den Generierungsfortschritt, da sowohl die Worker als auch das Frontend in Echtzeit mit der Datenbank im Backend kommunizieren. Die Benutzeroberfläche zeigt Änderungen an relevanten Werten nahezu sofort, da diese vorher frontendseitig abonniert wurden.
REQ16	Durch das Einteilen der Benutzeroberfläche in drei leicht verständliche Schritte und dem Verstecken der unterliegenden Komplexität der generativen KI-Modelle kann die Software auch von Nutzern ohne technischen Hintergrund bedient werden.
REQ17	Die generierten Kurzvideos werden möglichst ähnlich zu gängigen Social Media Stories gestaltet, indem innerhalb des Render-Workers Einblendungen, Untertitel und Animationspfade für die Bilder genutzt werden.
REQ18	Durch den vom Render-Worker eingeblendeten Hinweis am Anfang des Kurzvideos, wird dessen Inhalt als KI-generiert gekennzeichnet.

Tabelle 6 Umsetzung der Anforderungen

Nachdem in diesem Kapitel die Entwicklung des Softwareartefakts aufgeteilt nach dessen Komponenten beschrieben wurde, wird im nächsten Kapitel die Anwendung des Prototyps anhand mehrerer Beispielprojekte demonstriert.

6 Demonstration

Wie im vorherigen Kapitel beschrieben, bietet das prototypische Softwareartefakt drei verschiedene Möglichkeiten, ein Kurzvideo in Story-Form zu generieren. In diesem Kapitel wird die Videogenerierung aus einer Stellenanzeige, aus einem allgemeinen informativen Text und aus einem reinen Prompt jeweils an einem beispielhaften Projekt durchgeführt. So wird demonstriert, wie der Prototyp den Prozess der Kurzvideoerstellung automatisieren kann.

6.1 Demonstration der Videogenerierung aus einer Stellenanzeige

Diese Demonstration wird anhand der *Ausbildung als Orthopädieschuhmacher (m/w/d)* der *Thönnissen GmbH* in Koblenz durchgeführt (Referenznummer *10000-1194821802-S* bei der *Bundesagentur für Arbeit*). Dazu wird im ersten Schritt die URL dieser Stelle auf der Jobsuche-Website der *Bundesagentur für Arbeit* kopiert und, wie in Abbildung 17 zu sehen, auf der Startseite des Softwareprototypen unter dem Punkt *Import an existing Job Ad* eingefügt.

Abbildung 17 Screenshot der eingefügten URL der Stellenanzeige

Nach dem Klick auf *Import* werden die Details der Stellenanzeige über die API der *Bundesagentur für Arbeit* abgefragt und die für die Videoerstellung relevanten Informationen angezeigt, wie in Abbildung 18 zu sehen.

Abbildung 18 Screenshot der zur Stellenanzeige ermittelten Details

Wenn der Nutzer festgestellt hat, dass er das Kurzvideo aus diesen Informationen erstellen möchte, klickt er auf *Generate*, wird zur Projektansicht weitergeleitet und die Generierung des Storyboards durch den LLM-Worker startet. Ungefähr im Fünf-Sekunden-Takt erscheinen nun die Kapitel auf dem Storyboard, bis alle fünf Kapitel generiert wurden. Nach einer Wartezeit von insgesamt 63 Sekunden auf die Text-zu-Sprache- und Text-zu-Bild-Worker ist das Storyboard komplett. Die letzten drei Kapitel des Storyboards werden in Abbildung 19 gezeigt.

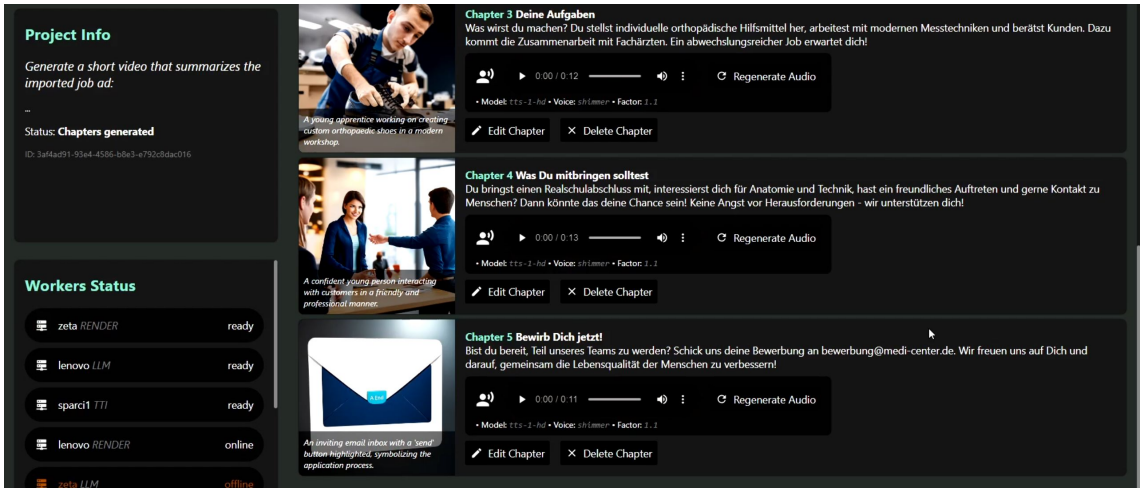


Abbildung 19 Screenshot des Storyboards zur Stellenanzeige

Nachdem beliebige Änderungen an den Texten und Bildern vorgenommen wurden, kann über den Button *Render chapters as video* der dritte Schritt, das Rendern des finalen Videos, angestoßen werden. Nach einer Rendering-Dauer von 3 Minuten und 20 Sekunden,

in der der prozentuale Fortschritt des Rendering-Workers angezeigt wird, startet die Wiedergabe des Kurzvideos (siehe Abbildung 20).

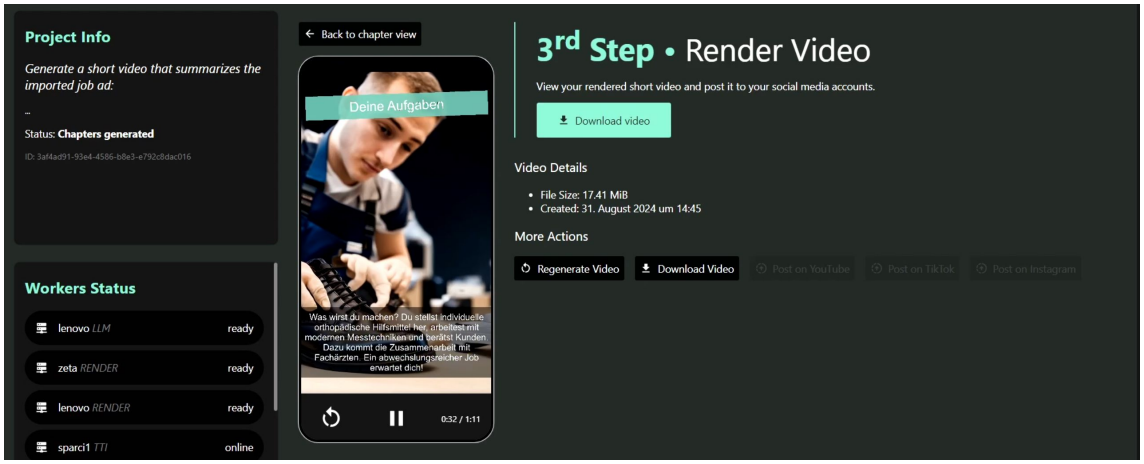


Abbildung 20 Screenshot des fertigen Kurzvideos zur Stellenanzeige

Die Videodatei wird nun mit einem weiteren Klick auf *Download Video* als MP4-Datei heruntergeladen. Der gesamte Videoerstellungsprozess hat 4 Minuten und 45 Sekunden gedauert. Das resultierende Video kann im Repository zu dieser Abschlussarbeit unter dem Dateinamen `Orthopaedieschuhmacher.mp4` angesehen werden.

6.2 Demonstration der Videogenerierung aus einem informativen Text

Neben der Erstellung von Kurzvideos für Ausbildungsstellenanzeigen, soll in dieser Abschlussarbeit auch untersucht werden, wie der gezeigte Prozess auf andere Anwendungsbereiche übertragbar ist. Dazu wird in diesem Abschnitt demonstriert, wie aus einem Sachtext mit den wichtigsten Fakten zum deutschen Grundgesetz ein informatives Kurzvideo in Story-Form erstellt wird. Die Textquelle[20] wird von der Website des *Deutschen Bundestags* kopiert und in das Textfeld unter *Upload an informative Text* eingefügt, wie in Abbildung 21 zu sehen.

[20] Link zum Text: bundestag.de/parlament/aufgaben/rechtsgrundlagen/grundgesetz/grundgesetz-197094

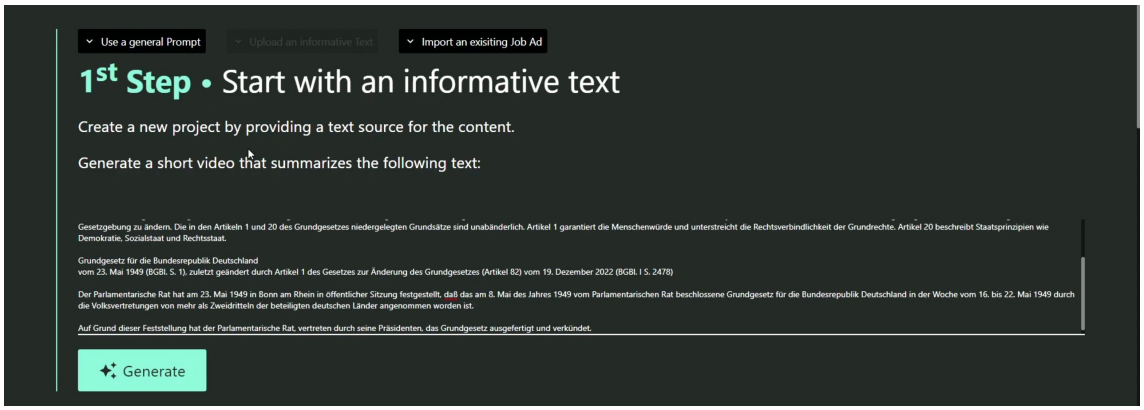


Abbildung 21 Screenshot des eingefügten Textes über das Grundgesetz

Durch einen Klick auf *Generate* wird das neue Projekt erstellt und der Generierungsprozess beginnt. Eine Minute und 15 Sekunden später ist das Storyboard mit allen Audio- und Bilddateien fertig generiert (siehe Abbildung 22) und der Rendering-Prozess wird gestartet.

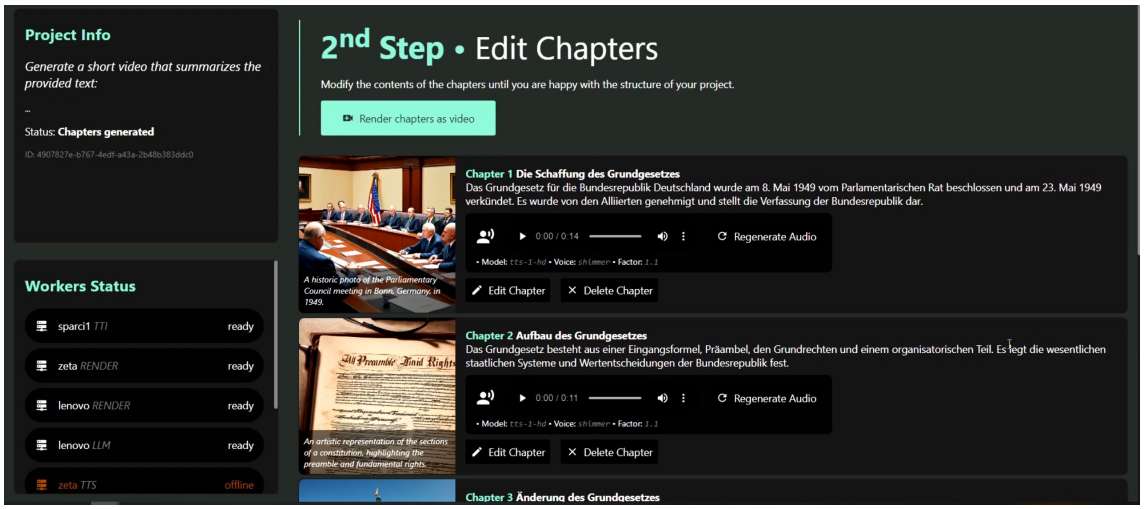


Abbildung 22 Screenshot des Storyboards über das Grundgesetz

Nach 3 Minuten und 30 Sekunden, der Dauer des Renderns der fünf Kapitel, wird das fertige Kurzvideo abgespielt und über den entsprechenden Button heruntergeladen, wie in Abbildung 23 gezeigt.

Auch dieses Kurzvideo in Story-Form befindet sich im angeschlossenen Repository unter dem Dateinamen `Grundgesetz.mp4`.

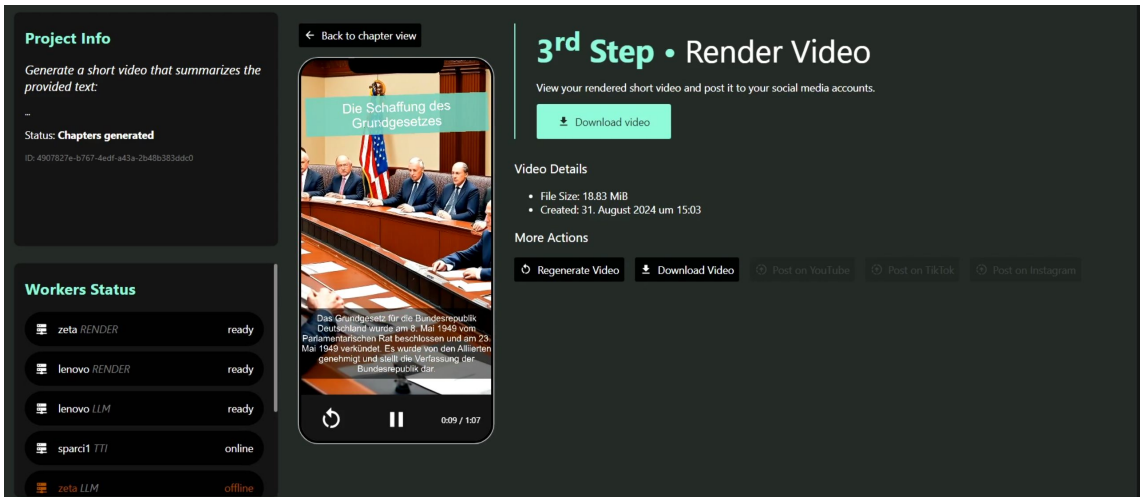


Abbildung 23 Screenshot des fertigen Grundgesetz-Kurzvideos

6.3 Demonstration der Videogenerierung aus einem reinen Prompt

Weil LLMs auf großen Mengen von Text trainiert werden, haben sie ein inhärentes Wissen über viele Thematiken (siehe Kapitel 5.4). Mit dem Softwareprototypen kann daher auch ohne das Hinzufügen von Informationsquellen aus einem einzelnen Prompt ein Kurzvideo generiert werden. Dies wird demonstriert, indem mit dem Prompt „Generate a short video that summarizes ... the history of England.“ ein Video generiert wird, das die Geschichte Englands zusammenfassen soll (siehe Abbildung 24).

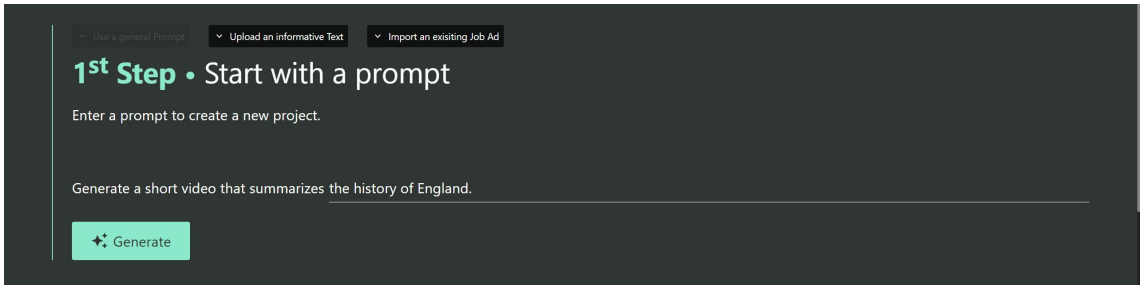


Abbildung 24 Screenshot des Prompts für ein Kurzvideo zur Geschichte Englands

Für das Storyboard werden dieses Mal per *Llama 3* insgesamt acht Kapitel generiert, die jedoch, wie in Abbildung 25 gezeigt, deutlich weniger Text beinhalten als die in den Abschnitten 6.1 und 6.2 erstellten Kapitel.

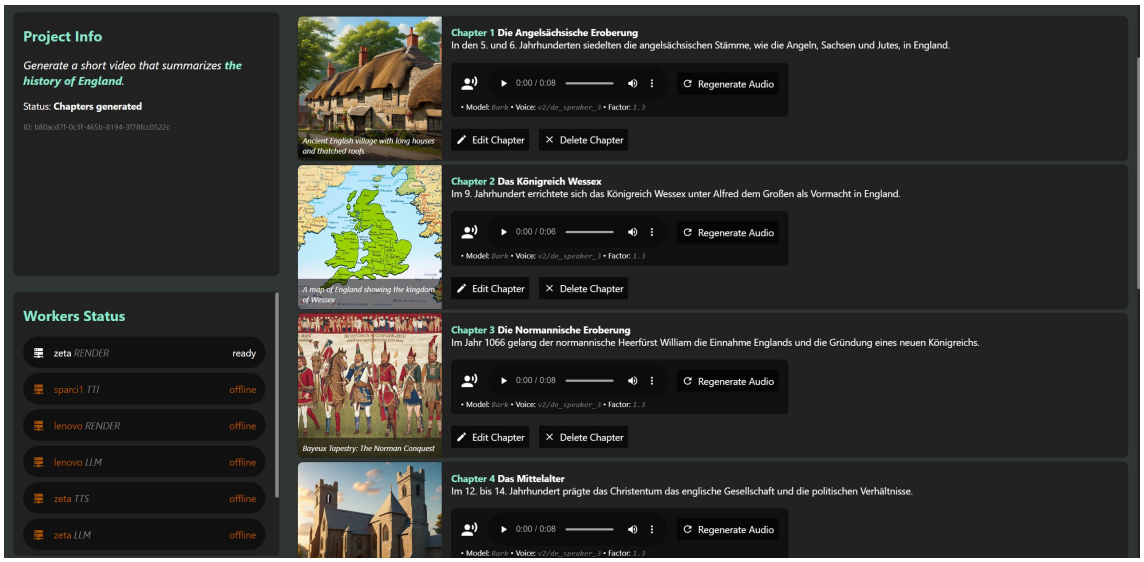


Abbildung 25 Screenshot des Storyboards über die Geschichte Englands

Nachdem der Text-zu-Bild-Worker mit *Stable Diffusion* alle Grafiken und der Text-zu-Sprache-Worker mit *Bark* alle Audiospuren generiert hat, wird das Rendering gestartet. Auch das in Abbildung 26 zu sehende fertige Kurzvideo in Story-Form kann unter dem Dateinamen `England.mp4` im Repository dieser Abschlussarbeit angesehen werden.

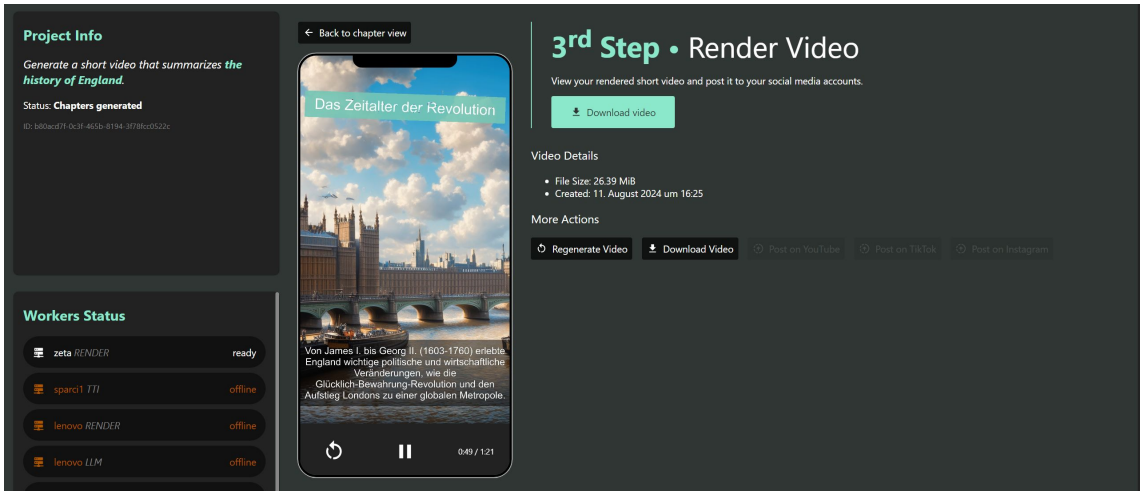


Abbildung 26 Screenshot des fertigen Kurzvideos zur Geschichte Englands

Nachdem in diesem Kapitel drei beispielhafte Kurzvideoerstellungen über verschiedene Informationsquellen demonstriert wurden, sollen im nächsten Kapitel das Softwareartefakt und dessen resultierende Videoartefakte evaluiert werden.

7 Evaluation

In diesem Kapitel wird das prototypische Softwareartefakt evaluiert, indem zukünftige Ausbildungssuchende in Form einer quantitativen Befragung die resultierenden Kurzvideos in Story-Form bewerten (Kapitel 7.1) und ein Experteninterview zur Nutzung der Software durchgeführt wird (Kapitel 7.2). Die hieraus resultierenden Erkenntnisse werden gemeinsam mit einer eigenen technischen Evaluation (Kapitel 7.3) genutzt, um die in Kapitel 4 getroffenen Designentscheidungen zu validieren und zu kritisieren.

7.1 Evaluation der resultierenden Videoartefakte

In diesem Abschnitt wird zuerst auf die verwendete Evaluationsmethode für die Videoartefakte eingegangen und danach die Ergebnisse der Evaluation vorgestellt.

Zu der Zielgruppe für die aus den Ausbildungsstellenanzeigen generierten Kurzvideos in Story-Form zählen vor allem Schülerinnen und Schüler ab der 10. Klasse. Im Rahmen einer quantitativen Befragung haben Jugendliche aus dieser Zielgruppe die Videos bewertet. Um Erkenntnisse über die generierten Videoartefakte und die vorangegangenen Designentscheidungen zu erhalten, wurden die Videos von den Probanden mit der jeweiligen Stellenanzeige in Textform verglichen, die als Informationsquelle genutzt wurde. Zuerst haben sie sich die originale Stellenanzeige angesehen, wie sie seitens der *Bundesagentur für Arbeit* veröffentlicht wurde. Daraufhin wurde das durch das Softwareartefakt generierte Kurzvideo in Story-Form vorgeführt und die Probanden wurden gebeten, verschiedene geschlossene Fragen zu beantworten. Zu den Aussagen haben sie ihre jeweilige Zustimmung auf einer Skala von 1 (stimme überhaupt nicht zu) bis 5 (stimme voll zu) angegeben. Folgende Aussagen *A1* bis *A9* haben die Probanden zu jeder Stellenanzeige bewertet.

A1 Die Inhalte des Kurzvideos sind gut verständlich.
A2 Das Kurzvideo ist informativ.
A3 Das Kurzvideo sieht qualitativ hochwertig aus.
A4 Die Hintergrundbilder passen zum Inhalt des jeweiligen Abschnitts.
A5 Das Kurzvideo ähnelt anderen Inhalten auf Social-Media.
A6 Es ist erkennbar, dass das Kurzvideo mithilfe von KI generiert wurde.
A7 Das Kurzvideo beinhaltet alle wichtigen Informationen zur vorgestellten Stelle.
A8 Durch die Darstellung als Kurzvideo bleibt die Stellenanzeige länger in Erinnerung.
A9 Ich bevorzuge das Kurzvideo gegenüber der Stellenanzeige in Textform.

Zusätzlich zu den Aussagen *A1* bis *A9*, die spezifisch zu einer Stelle und deren Video gehören, haben die Probanden drei weitere Aussagen *A10* bis *A12* bewertet, die generell die Erstellung von KI-generierten Kurzvideos aus Ausbildungsstellenanzeigen behandeln.

A10 Ich finde es gut, wenn ein Unternehmen seine Stellen auch in Form von Kurzvideos präsentiert.

A11 Ich bin eher bereit, mir mehrere Stellenanzeigen im Kurzvideo-Format anzusehen, als diese in Textform durchzulesen.

A12 Mir ist wichtig, dass KI-generierte Inhalte als solche kenntlich gemacht werden.

Die Befragung wurde persönlich an einer weiterführenden Schule in Koblenz durchgeführt und es wurden jeweils 28 Datenpunkte zu den Aussagen *A1* bis *A9* gesammelt und 14 Datenpunkte zu den Aussagen *A10* bis *A12*. Sowohl der originale Fragebogen als auch eine Tabelle mit allen gesammelten Datenpunkten befinden sich im Repository dieser Abschlussarbeit. Die vollständige Auswertung zu den Aussagen *A1* bis *A9* als Diagramm befindet sich in Abbildung 27.

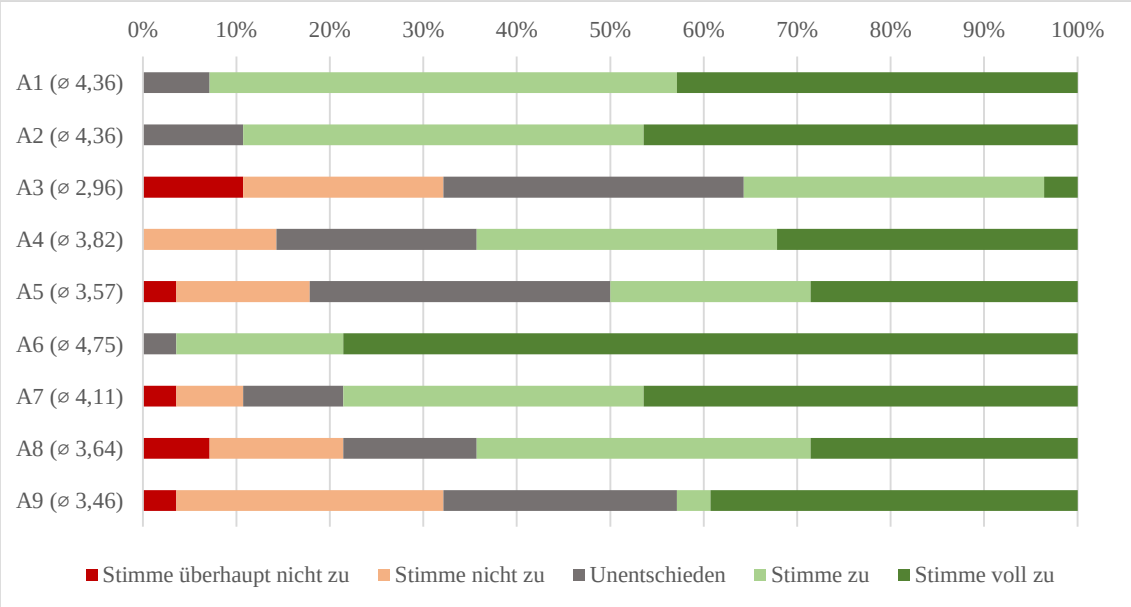


Abbildung 27 Evaluationsergebnisse zu den Aussagen *A1* bis *A9*

Während die Probanden das gezeigte Kurzvideo in Story-Form überwiegend verständlich (*A1*, Ø 4,36) und informativ (*A2*, Ø 4,36) fanden, wurde dessen Qualität mittelmäßig bewertet (*A3*, Ø 2,96). Den Aussagen, dass die generierten Bilder gut zum Inhalt des jeweiligen Abschnitts passen (*A4*, Ø 3,82) und dass das Kurzvideo anderen Inhalten auf Social Media ähnelt (*A5*, Ø 3,57), haben die Probanden teilweise zugestimmt. Über 90%

der Probanden stimmen zu, dass erkennbar ist, dass das Video mithilfe von KI generiert wurde (*A6*, Ø 4,75). Der Aussage, dass das Video alle wichtigen Informationen zu der vorgestellten Stelle enthält, stimmten die meisten Probanden zu (*A7*, Ø 4,11). Bei der Frage, ob die Stelle durch die Darstellung als Kurzvideo länger in Erinnerung bleibt, herrschte Uneinigkeit. Knapp zwei Drittel der Probanden stimmten der Aussage zu, während ca. 20% antworteten, dass sie ihr nicht zustimmen (*A8*, Ø 3,64). Über 40% der Befragten bevorzugten das Kurzvideo gegenüber der Stellenanzeige in Textform, während ca. 30% den Fließtext präferierten. Ca. ein Viertel der Probanden waren hierzu unentschieden (*A9*, Ø 3,46).

Ein Diagramm mit den Ergebnissen zu den generellen Aussagen *A10* bis *A12* befindet sich in Abbildung 28.

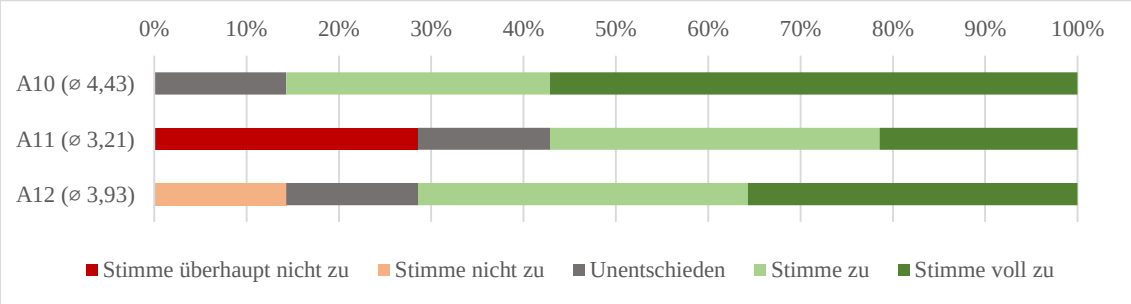


Abbildung 28 Evaluationsergebnisse zu den Aussagen *A10*, *A11* und *A12*

Es wird von über 85% der Probanden als gut empfunden, wenn ein Unternehmen seine Stellen auch in Form von Kurzvideos in Story-Form präsentiert. Die restlichen Befragten waren hierzu unentschieden (*A10*, Ø 4,43). Über die Hälfte der Schülerinnen und Schüler waren außerdem eher bereit, sich mehrere Stellenanzeigen im Kurzvideo-Format anzusehen, als diese in Textform durchzulesen. Knapp 30% stimmten dem jedoch überhaupt nicht zu (*A11*, Ø 3.21). Zuletzt stimmten über 70% der Probanden zu, dass es wichtig ist, KI-generierte Inhalte als solche kenntlich zu machen (*A12*, Ø 3,93).

Die Erkenntnisse aus dieser Befragung und des folgenden Experteninterviews werden in Kapitel 7.3 auf die Designentscheidungen bezogen, um diese zu validieren und zu kritisieren.

7.2 Evaluation des Softwareartefakts

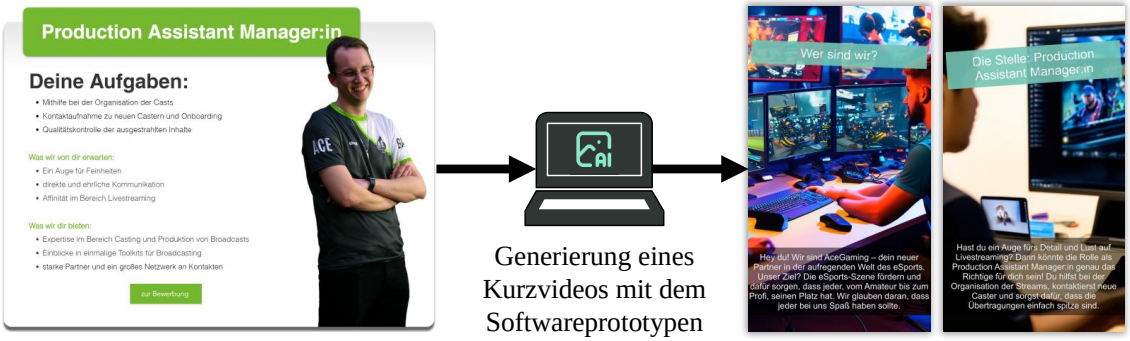
Zur Evaluation des Softwareartefakts wurde ein Experteninterview mit dem Social-Media-Manager von *AceGaming*[21] geführt, der unter anderem verantwortlich für die Erstellung

[21] AceGaming Homepage: acegaming.de

von Content für die verschiedenen Social-Media-Kanäle des E-Sports-Vereins ist. Das Interview wurde online geführt und genutzt, um Feedback zu dem entwickelten prototypischen Softwareartefakt zu erhalten. Dazu wurden relevante Informationen zu dem aktuellen Prozess der Story-/Kurzvideo-Erstellung innerhalb der Organisation erfragt und diese mit der Bedienung des prototypischen Softwareartefakts verglichen, indem gemeinsam ein Kurzvideo generiert wurde.

Die Organisation erstellt neben Posts als Bilder auch viel Content in Videoform. Gerade Posts im Kurzvideoformat spielen dabei für Plattformen wie *X* und *TikTok* eine große Rolle. Teilweise werden hierzu Ausschnitte aus längeren Videos verwendet und diese als Kurzvideo gepostet. Häufig werden von der Organisation aber auch informative Kurvideos erstellt, für die professionelles Equipment und eine geeignete Location gemietet werden. Dabei beläuft sich allein der anschließende Editierungsaufwand auf ungefähr 8 bis 12 Stunden. Generative KI wird von der Organisation schon teilweise eingesetzt, vor allem um Hintergrundgrafiken für ihre geposteten Bilder zu generieren. Diese generierten Bilder werden dann als Basis zur weiteren Editierung in Bildbearbeitungsprogrammen genutzt. Gerade auch ausgeschriebene Stellen wurden von der Organisation bisher nur als Bild über die Social-Media-Kanäle beworben und noch nicht in Kurzvideoform.

Um den Softwareprototypen zu evaluieren, wurde während des Interviews gemeinsam ein Kurzvideo im Story-Format generiert. Als Informationsquelle für das Video wurde die von *AceGaming* zum Zeitpunkt des Interviews ausgeschriebene Stelle als *Production Assistant Manager:in* gewählt, die in Abbildung 29 zu sehen ist. Das finale Kurzvideo kann unter dem Dateinamen `AceGaming.mp4` im Repository gefunden werden.



Originale Stellenanzeige auf der AceGaming Homepage

Generierung eines Kurzvideos mit dem Softwareprototypen

Screenshots aus dem generierten Kurzvideo

Abbildung 29 Generierung einer Story aus der Stellenanzeige von *AceGaming*

Bei der Erstellung des neuen Projektes wurde vom Experten als erstes die Schnelligkeit der Storyboard-Erstellung positiv hervorgehoben. Auch die generierten Titel der Kapitel und deren Inhalte wurden als qualitativ hochwertig bezeichnet. Der Prototyp hat es

laut dem Experten geschafft, die wichtigsten Informationen aus dem eingegebenen Text zu extrahieren. Zwar wurde bei der Bildgenerierung auch die schnelle Generierungsgeschwindigkeit gelobt, die Qualität der einzelnen erstellten Grafiken jedoch bemängelt. Hier waren offensichtliche Fehler des Diffusionsmodells erkennbar. Die automatische Hintergrundbildgenerierung als generelles Feature, also unabhängig von den Schwächen des gewählten Text-zu-Bild-Modells, wurde als sehr hilfreich bewertet. Ähnliche Aussagen wurden auch im Bezug auf die Audiogenerierung getroffen. Der Experte hat die Automatisierung der Stimmengenerierung und deren Einbindung in die Benutzeroberfläche positiv bewertet. Beim Anhören der Audiospuren kritisierte er jedoch die aufgetretenen Aussprachefehler und generell die Unnatürlichkeit der Stimme. Um ein Kurzvideo zu erstellen, das zur Zielgruppe der Organisation passt, wurde in den Einstellungen des Softwareprototyps auch während des Interviews der Meta-Prompt zur Erstellung des Storyboards angepasst. Hierbei wurde gelobt, dass die verschiedenen Generierungsparameter und Prompts vom Nutzer einfach angepasst werden können. Gleichzeitig wurden aber auch Bedenken geäußert, ob die komplett freie Veränderung des Meta-Prompts in einem produktiven Einsatz nicht dazu führen könnte, dass die Software zur Generierung von schädlichen Inhalten missbraucht werden kann. Nach weiterer Nutzung der Software hat der Experte die Benutzeroberfläche gelobt und als intuitiv bedienbar beschrieben. Auch Kolleginnen und Kollegen ohne technischen Hintergrund könnten diese ohne Probleme bedienen. Ausschließlich die Komponente mit der Statusanzeige der verschiedenen Worker würde er in einem Produktivsystem verbergen, da Nutzer durch das Anzeigen von Workern mit dem Status `offline` irritiert werden könnten.

Nachdem das manuelle Nacharbeiten der generierten Inhalte im Storyboard nach ca. fünf Minuten abgeschlossen war, wurde das Rendering des Kurzvideos gestartet. Die Dauer des Renderings von unter zwei Minuten und auch die Benutzeroberfläche der Videoansicht wurden vom Experten positiv bewertet. Beim gemeinsamen Ansehen des generierten Kurzvideos gab es mehrere Kritikpunkte. Zum einen sei das Kurzvideo sehr offensichtlich KI-generiert. Dies erkenne man sowohl an den Hintergrundbildern als auch an der verwendeten TTS-Stimme. Der Experte gab zu bedenken, dass dies auf den verschiedenen Social-Media-Plattformen zu weniger *User Engagement* (Bereitschaft der Nutzer, sich mit dem Inhalt auseinanderzusetzen) führen könnte als manuell erstellte Kurzvideos. Nichtsdestotrotz wären auch offensichtlich KI-generierte Kurzvideos eine Chance für Organisationen mit geringerem Budget, die ansonsten gar keine Inhalte für Social Media erstellen würden. Darüber hinaus wurde kritisiert, dass die durch den Softwareprototypen generierten Kurzvideos alle einen sehr ähnlichen Stil bzw. sehr ähnliches Aussehen haben. Es wäre also vorteilhaft, mehr Anpassungsmöglichkeiten für die Ästhetik des finalen Videos zu haben. Hierin lag auch der größte Kritikpunkt, weswegen das generierte Video so nicht ohne weitere Anpassungen gepostet werden könnte. Alle erstellten Inhal-

te müssen dem *Corporate Design (CD)* der Organisation entsprechen. Da die Software keine Möglichkeit liefert, Designmerkmale aus anderen Posts zu übernehmen und auch keine Änderung an Farben, Logos und Stilen möglich ist, müsste das Video erst noch in einem externen Videobearbeitungsprogramm angepasst werden, um so den Vorgaben des CD gerecht zu werden. Den Hinweis zu Beginn des Videos, dass es mithilfe generativer KI generiert wurde, befand der Experte als sinnvoll und wichtig.

Als konkreter Verbesserungsvorschlag wurde genannt, dass die Bild- und Videodateien nicht nur durch die Veränderung der Prompts ersetzt werden können, sondern auch eigene Bilddateien hochgeladen und Audiospuren aufgenommen werden können. So könnten Nutzer selbst entscheiden, ob sie mit mehr Bearbeitungsaufwand eine bessere Videoqualität erzielen möchten. In diesem Fall würde die Software eher als Werkzeug fungieren, um per KI ein Basis-Storyboard zu generieren, das dann mit eigenen Inhalten ausgefüllt wird, so der Experte. Außerdem müssen Content Creator zu jedem erstellten Post auch immer eine Post-Beschreibung verfassen. Ein wichtiges Feature läge also auch darin, basierend auf den Inhalten des Kurzvideos, eine passende Beschreibung für den Post in Textform zu generieren.

Zum Ende des Interviews hat der Experte zusammengefasst, dass das Softwareartefakt in seiner aktuellen Form am ehesten als Werkzeug zur Storyboard-Generierung eingesetzt werden könnte, das Content Creator als Basis für eine Story verwenden. Durch das Entwickeln von weiteren Features zur besseren Individualisierung der Kurzvideos könne die Software zu einem sehr hilfreichen Werkzeug für den kompletten Prozess der Content-Erstellung werden und diesen enorm vereinfachen.

7.3 Technische Evaluation

In der folgenden technischen Evaluation werden die Erkenntnisse aus der vorangegangenen Befragung zu den Videoartefakten und dem Experteninterview genutzt, um die getroffenen Designentscheidungen zu validieren und zu kritisieren. Außerdem wird die Erfüllung der Anforderungen bewertet.

Eine systematische Prüfung aller Designentscheidungen mit den neuen Erkenntnissen aus der Evaluation befindet sich in Tabelle 7. Hier wird angegeben, wie Designentscheidungen durch die Evaluierung validiert werden konnten, aber auch, welche Änderungen sinnvoll gewesen wären. Diese Bewertungen können als Grundlage für die Weiterentwicklung des Softwareartefakts dienen.

DE	Bewertung
DE1	Sowohl strukturierte als auch unstrukturierte Eingaben konnten während der Entwicklung, Demonstration und Evaluation problemlos vom Softwareprototypen verarbeitet werden. Gerade diese Flexibilität hat es im Experteninterview ermöglicht, die Stellenanzeige in unstrukturierter Form als Eingabe zu nutzen, was vom Experten positiv hervorgehoben wurde. Für die bei der Befragung genutzten Stellen wurde die strukturierte Eingabemöglichkeit verwendet und knapp 80% der Probanden gab an, dass alle wichtigen Informationen von der Stellenanzeige ins Video übernommen wurden (*A7*).
DE2	Durch die Importmöglichkeit von Stellenanzeigen im strukturierten Format über die Jobsuche-API der *Bundesagentur für Arbeit* kann laut Mehrheit der Probanden ein informatives Kurzvideo generiert werden (*A2*), das die wichtigsten Informationen der Stelle enthält (*A7*). Die Nutzung dieser Schnittstelle ist also sinnvoll.
DE3	Die Verwendung der strukturierten Ausgabe des LLMs im JSON-Format für die Darstellung des Storyboards hat während der Entwicklung, Demonstration und Evaluation durchgängig einwandfrei funktioniert. Daher wird diese Designentscheidung als validiert angenommen.
DE4	Im Experteninterview wurde der Aufbau der Kapitel als sinnvoll erachtet. Außerdem gaben nur 14% der Befragten an, dass die Hintergrundbilder nicht zum Inhalt des jeweiligen Abschnitts passen (*A4*). Die entwickelte Struktur des Storyboards ist also positiv zu bewerten.
DE5	Während der Evaluation konnte festgestellt werden, dass die grundliegende Bearbeitung des Storyboards durch die grafische Benutzeroberfläche zwar grundsätzlich möglich ist, aber einige sinnvolle Anpassungsmöglichkeiten fehlen. Dazu gehört laut Experten unter anderem die Möglichkeit, das Video an das CD der jeweiligen Organisation anzupassen, was beispielsweise durch das Einfügen von Logos und das Ändern des Farbschemas möglich wäre. Diese Designentscheidung hätte also umfangreicher definiert werden sollen.
DE6	Die Möglichkeit, passende Hintergrundbilder mit einem generativen Text-zu-Bild-Modell zu erstellen, wurde im Experteninterview positiv bewertet. Es sollte jedoch zusätzlich noch die Möglichkeit geben, eigene Grafiken einzufügen.
DE7	Die Möglichkeit, automatisch Audiospuren mit einem generativen Text-zu-Sprache-Modell zu erstellen, wurde im Experteninterview auch positiv bewertet. Es sollte jedoch zusätzlich noch die Möglichkeit geben, die Texte mit der eigenen Stimme vorzulesen und diese Aufnahme für das Kurzvideo zu nutzen.

DE	Bewertung
DE8, DE9, DE10	Die strukturierten Informationen aus dem Storyboard und die generierten Bild- und Audiodateien werden verwendet, um automatisiert ein Video zu rendern, anzuzeigen und herunterzuladen. In der Demonstration wurde gezeigt, dass dieser Prozess funktioniert, und in der Evaluation wurde die Automatisierung dieses Prozesses vom Experten positiv hervorgehoben. Dadurch konnten alle drei Designentscheidungen validiert werden.
DE11	Die einfache Änderbarkeit der verschiedenen Generierungsparameter und Prompts wurden vom Experten gelobt. Es ist aber sicherzustellen, dass die komplett freie Veränderung des Meta-Prompts nicht für die Generierung schädlicher Inhalte genutzt werden kann. Diese Designentscheidung hätte gegebenenfalls um diese Maßnahme erweitert werden sollen.
DE12	Die Trennung der Benutzeroberfläche als Web-Applikation von der Ausführung der verschiedenen KI-Modelle hat sich als sinnvoll herausgestellt, da die Software so auf mehreren Systemen (z. B. während des Experteninterviews) ohne Installation genutzt werden konnte.
DE13	Die Zugriffsbeschränkung durch Abfragen von Benutzername und Passwort hat sich während der Entwicklung, Demonstration und Evaluation als sinnvoll herausgestellt, damit keine fremden Nutzer Zugang zur Software haben.
DE14	Die Entwicklung einer Architektur mit mehreren Workern, die unabhängig voneinander betrieben werden können und über die Datenbank mit dem Frontend kommunizieren, war sinnvoll. Unter anderem durch die Parallelisierung, die hierdurch möglich wird und das unkomplizierte Hinzufügen von Rechenressourcen, konnten geringe Generierungszeiten während der Demonstration und des Experteninterviews erreicht werden, die vom Experten gelobt wurden.
DE15	Die in Echtzeit im Frontend erscheinenden Resultate wurden vom Experten positiv hervorgehoben, da so bereits Änderungen an den ersten generierten Kapiteln vorgenommen werden können, während an anderen Stellen noch Texte, Grafiken oder Audiospuren parallel generiert werden. Außerdem ist dies vorteilhaft für die Nutzererfahrung, da der Fortschritt real angezeigt wird, anstatt diesen beispielsweise durch einen Fortschrittsbalken zu abstrahieren, so der Experte.
DE16	Der Experte hat bestätigt, dass es wichtig ist, dass auch Nutzer ohne technischen Hintergrund die Software bedienen können. Außerdem hat er zugestimmt, dass dies beim Softwareprototypen der Fall ist. Lediglich die detaillierte Anzeige aller Worker-Systeme ist laut Experten unwichtig für einen Anwender und könnte sogar die Nutzererfahrung verschlechtern.

DE	Bewertung
DE17, DE18, DE19	Die Hälfte der Befragten stimmten der Aussage zu, dass das Kurzvideo Ähnlichkeit zu anderen Inhalten auf Social Media hat (*A5*). Jedoch wurde im Experteninterview bemängelt, dass alle generierten Videos ähnlich aussehen, da immer der gleiche Generierungsstil verwendet wird. Ggf. hätten bei diesen Designentscheidungen also für mehr Varianz in den generierten Kurzvideos gesorgt werden sollen.
DE20	Das Einblenden des Hinweises, dass das Kurzvideo KI-generiert ist, hat sich als sinnvoll herausgestellt. Sowohl der Experte als auch über 70% der befragten Schülerinnen und Schüler finden diese Kennzeichnung wichtig (*A12*).
DE21	Von der Auswahl des LLMs hängt unmittelbar die Verständlichkeit und die Informativität des generierten Videos ab. Beide Attribute wurden in der Befragung von ca. 90% der Befragten bestätigt (*A1*, *A2*). Auch der Experte hat die generierten Texte als qualitativ hochwertig beschrieben. Das validiert die LLM-Auswahl.
DE22	Nur ca. ein Viertel der Befragten empfanden das Aussehen des Kurzvideos als qualitativ hochwertig (*A3*). Da das Aussehen des Videos maßgeblich vom verwendeten Text-zu-Bild-Modell beeinflusst wird, sollte dessen Auswahl überdacht werden.
DE23	Wie auch bei der Auswahl des LLMs kann die Auswahl des Text-zu-Sprache-Modells durch die Verständlichkeit des Videos validiert werden. Über 90% der Befragten empfanden das Kurzvideo als gut verständlich (*A1*). Der Experte kritisierte jedoch die Qualität der vorlesenden Stimme und hat diese als „deutlich erkennbar KI-generiert“ beschrieben.
DE24, DE25, DE26, DE27, DE28	Die Auswahl der konkreten Technologien zur Entwicklung des Back- und Frontends wurden nicht explizit in der Evaluierung bewertet. Da der Softwareprototyp während der Entwicklung, den verschiedenen Demonstrationen und Evaluationen problemlos und performant ausgeführt wurde, hat sich die Technologieauswahl als sinnvoll herausgestellt.
DE29	Die Möglichkeit, neben den quelloffenen KI-Modellen auch auf kommerzielle Modelle zurückgreifen zu können, wurde vom Experten als sinnvoll befunden. Ob hierfür konkret die Modelle von *OpenAI* besonders geeignet sind, lag außerhalb des Evaluierungsumfangs.

Tabelle 7 Evaluation der Designentscheidungen

Neben den Erkenntnissen zu den verschiedenen Designentscheidungen wird zur Vollständigkeit nochmals auf die Erfüllung der Anforderungen eingegangen. Diese ist in den meisten Fällen einfach belegbar, da die Entwicklung der entsprechenden Features in Kapitel 5 ausführlich beschrieben wurde. Tabelle 6 beinhaltet eine Auflistung aller Anforderungen und beschreibt, wie diese vom Softwareprototypen umgesetzt werden.

Für die folgenden Anforderungen wurden während der Evaluation neue Erkenntnisse gewonnen: *REQ2* beschreibt, dass die Software die relevantesten Informationen aus der Eingabe identifizieren und extrahieren können soll. Da es schwierig ist, den Grad der Relevanz vollständig objektiv zu bestimmen, können hier die subjektiven Eindrücke aus der Evaluation genutzt werden. Von den befragten Schülerinnen und Schülern stimmten über 78% der Aussage zu, dass das Kurzvideo alle wichtigen Informationen zur vorgestellten Stelle enthält (*A7*). Auch der Experte bestätigte im Rahmen des Experteninterviews, dass die relevantesten Informationen aus der Eingabe ins generierte Kurzvideo übernommen werden. *REQ5* beschreibt, dass der Nutzer die Möglichkeit haben soll, das Storyboard vor der Generierung des Videos zu bearbeiten. Während Titel, Text und Hintergrundbild-Prompt bearbeitet werden können, wurde im Experteninterview kritisiert, dass beispielsweise die Bilder und Audiospuren nicht durch eigene Inhalte ersetzt werden können. Diese Anforderung wurde also rückblickend nur teilweise erfüllt. Die Anforderung *REQ16* verlangt, dass auch Nutzer ohne technischen Hintergrund die Software bedienen können. Dies wurde während des Experteninterviews überprüft und vom Experten bestätigt. Neben der Bewertung der existierenden Anforderungen ist durch die Evaluierung außerdem aufgefallen, dass es sinnvoll gewesen wäre, weitere Anforderung an das Softwareartefakt zu stellen. Denkbar wäre beispielsweise eine Anforderung, die die Möglichkeit zur Einhaltung eines vorgegebenen CD gewährleistet. Auch eine Anforderung, dass neben dem eigentlichen Video zusätzlich eine Post-Beschreibung in Textform generiert werden soll, wäre sinnvoll gewesen.

Mit diesen Erkenntnissen aus der Evaluation der Software und der Videoartefakte werden im nächsten Kapitel die Ergebnisse dieser Arbeit diskutiert.

8 Diskussion

In diesem Kapitel werden die Forschungsergebnisse zusammengefasst und interpretiert. Neben der Abgrenzung von verwandten Arbeiten werden Einschränkungen des entwickelten Artefakts aufgezeigt und Anknüpfungspunkte für zukünftige Forschung diskutiert.

8.1 Interpretation der Ergebnisse

In den vorherigen Kapiteln konnte der größte Teil der Designentscheidungen positiv validiert werden, während für die anderen neben der Kritik schon konkrete Verbesserungsvorschläge formuliert wurden. Im Folgenden werden diese Ergebnisse genutzt, um die Effektivität und Effizienz der Kurzvideogenerierung einzuschätzen, über den Grad der Automatisierung zu diskutieren und die Allgemeingültigkeit des Softwareartefakts zu betrachten.

Die Effektivität der Stories zur Präsentation von Ausbildungsplätzen wird vor allem dadurch bestimmt, wie gut die Videos bei der Zielgruppe von Ausbildungssuchenden ankommen. Hierzu wurden verschiedene Merkmale, die die Effektivität des Kurzvideos ausmachen, wie der Informationsgehalt, die Qualität des Videos und das ggf. geweckte Interesse an der Stelle, ermittelt. Mit der quantitativen Befragung von Schülerinnen und Schülern aus der Zielgruppe der Videos wurde eine Methode entwickelt, um diese Merkmale zu quantifizieren und auswerten zu können. Aus der Gesamtheit der ermittelten Meinungen lässt sich die Effektivität der Stories schon überwiegend positiv bewerten, mit dem Ausblick auf eine noch deutlich höhere Effektivität durch die Weiterentwicklung des Softwareartefakts. Neben der Bestimmung der Effektivität der Stories, ist es ein Hauptbestandteil dieser Arbeit, den Zeit- und Ressourcenaufwand des Erstellungsprozesses durch generative KI möglichst stark zu reduzieren. Die Messung dieser Aufwandsreduzierung bestimmt die Effizienz der Videogenerierung und somit des Softwareartefakts an sich. Hierfür ist besonders der Vergleich der Zeitmessungen der verschiedenen Generierungsschritte zusammen mit den Erkenntnissen zum traditionellen Videoerstellungsprozess aus dem Experteninterview relevant. Dieser zeigt, dass vergleichbar lange Videos, mit den oben genannten Nachteilen gegenüber traditionell erstellten Kurzvideos, in nur einem Bruchteil der Zeit erstellt werden können. Gegenüber den 8 bis 12 Stunden, die im traditionellen Prozess laut Experteninterview allein der Schnitt und die Nachbearbeitung des Videomaterials dauert, können mit dem Softwareprototypen Kurzvideos innerhalb weniger Minuten generiert werden (durchschnittlich 6 Minuten inklusive menschlicher Nachbearbeitung des Storyboards). Zusätzlich zum Zeitaufwand spielt auch der finanzielle Aufwand eine Rolle. Neben den Kosten zum Betreiben quelloffener Modelle auf

eigenem Hosting werden auch teilweise kommerzielle Modelle von *OpenAI* eingesetzt. Konkret kostet die Generierung eines Storyboards bei Verwendung von *gpt-4o* als LLM durchschnittlich unter einem Cent (Ø 1.085 Tokens). Die Verwendung von *tts-1-hd* als Text-zu-Sprache-Modell verursacht bei einem durchschnittlichen Projekt Kosten von ca. fünf Cent (Ø 1.802 Zeichen). Sowohl der Zeit- als auch der Kostenaufwand für die Generierung informativer Stories mit dem entwickelten Softwareprototypen ist also ausgesprochen gering.

Der Grad der Automatisierung, der mit dem KI-gestützten Videogenerierungsprozess erreicht werden kann, wird maßgeblich davon bestimmt, wann menschliches Eingreifen erforderlich bzw. vorteilhaft ist. Rein technisch betrachtet könnte der Softwareprototyp durch eine Schnittstelle programmatisch bedient werden und vollständig automatisiert Kurzvideos generieren. In der Praxis ist jedoch menschliches Eingreifen erforderlich, um sicherzustellen, dass die Stories keine Fehlinformationen enthalten, die beispielsweise durch Halluzinationen des genutzten LLMs entstehen können. Außerdem muss je nach genutzten generativen KI-Modellen kontrolliert werden, dass die generierten Texte und Bilder keine schädlichen Inhalte enthalten. Während kommerzielle Modelle häufig integrierte Schutzmechanismen beinhalten, gibt es quelloffene Modelle, die keine inhärenten Beschränkungen implementieren. Neben diesen erforderlichen Maßnahmen gibt es weitere Bereiche, in denen menschliches Eingreifen vorteilhaft ist. Hierzu gehört beispielsweise das Umformulieren von ungewöhnlichen Textpassagen in den generierten Kapiteln und das Abändern von Hintergrundgrafik-Prompts, die kein gutes Ergebnis bei der Bildgenerierung liefern. So kann, mit vergleichsweise wenig manuellem Aufwand, ein qualitativ deutlich hochwertigeres Endergebnis erzielt werden als ohne Veränderungen.

Neben dem Einsatz des Softwareartefakts zur Generierung von Kurzvideos in der Anwendungsdomäne der Ausbildungsstellenanzeigen, konnte gezeigt werden, dass der Softwareprototyp auch informative Kurzvideos zu beliebigen anderen Inhalten generieren kann. Für jedes Anwendungsgebiet gibt es zwei Möglichkeiten die Software einzusetzen. Zum einen kann der voreingestellte generische Prompt zur Generierung eines informativen Kurzvideos ohne weitere Angaben genutzt werden. Dann werden die Informationen aus dem Text in einem Kurzvideo zusammengefasst, dessen Aufbau und Form aber sehr unterschiedlich ausfallen kann und keinen domänenspezifischen Anforderungen folgt. Alternativ kann für eine bestimmte Anwendungsdomäne auch einmalig ein expliziter Prompt formuliert werden, durch den verschiedene Vorgaben für das Video festgelegt werden. Im Prompt zur Videogenerierung von Ausbildungsstellenanzeigen wurde beispielsweise festgelegt, dass am Anfang des Videos erst der Ausbildungsbetrieb kurz vorgestellt werden soll, der Zuhörer geduzt wird und das Video vor allem interessant für eine jüngere Zielgruppe sein soll. Nachdem ein Prompt für eine bestimmte Anwendungskategorie formu-

liert wurde, kann dieser zur Generierung beliebig vieler Videos in dieser Domäne genutzt werden. Dies bewirkt, dass die Resultate deutlich konsistenter sind. Diese Anpassungsmöglichkeit bietet viel Potenzial, die Software zukünftig auch in anderen spezifischen Anwendungsgebieten einsetzen zu können.

Abschließend werden noch die Ergebnisse der quantitativen Befragung der Story-Zielgruppe verwendet, um die Motivation dieser Abschlussarbeit zu diskutieren. In der Anwendungsdomäne der Ausbildungsstellenanzeigen soll mit den generierten Kurzvideos in Story-Form ein Mehrwert sowohl für Ausbildungssuchende als auch für Ausbildungsbetriebe erzeugt werden, indem die Stellen in einem moderneren Format präsentiert werden. Die Befragung hat jedoch gezeigt, dass die Stellenanzeigen im Kurzvideo-Format in ihrer aktuellen Form nicht für jeden Ausbildungssuchenden als Mehrwert wahrgenommen werden. Obwohl über 60% der Probanden der Meinung sind, dass die Stellenanzeige als Kurzvideo länger in Erinnerung bleibt als die Stellenanzeige in Textform, bevorzugen nur ca. 40% der Befragten das Kurzvideo gegenüber der Stellenanzeige in Textform. Ca. 30% der Befragten gaben an, dass sie die Stellenanzeige in Textform bevorzugen. Während es über 85% der Probanden gut finden, wenn ein Unternehmen seine Stellen auch in Form von Kurzvideos präsentiert, sind nur ca. 55% bereit, sich mehr Stellenanzeigen im Kurzvideo-Format anzusehen, als diese in Textform durchzulesen. Diese Resultate zeigen, dass mit dem aktuellen Stand des Softwareartefakts zumindest für einen Teil der Zielgruppe ein Mehrwert geschaffen werden kann. Durch die Weiterentwicklung der Software und Implementierung der gesammelten Verbesserungsvorschläge kann eine Verbesserung der Videoqualität erreicht und somit auch die Akzeptanz der Stories gesteigert werden.

8.2 Abgrenzung zu verwandten Arbeiten

In Kapitel 2 wurden mehrere verwandte Arbeiten identifiziert und beschrieben. In diesem Abschnitt sollen die Ergebnisse dieser Arbeit von den verwandten Arbeiten abgegrenzt werden.

Während in „Automatic Video Creation From a Web Page“ von Chi et al. auch automatisiert ein Video gerendert wird, dient immer eine Website als Datenquelle. Es werden keine neuen Inhalte mithilfe von generativer KI erstellt, sondern lediglich existierende Grafiken und Texte von der Website extrahiert und in das Video eingefügt. Der Vorteil an dieser Vorgehensweise ist, dass so im Gegensatz zur selbst entwickelten Software, die automatische Anpassung des Videos an das originale Design der Website und somit an das CD der Organisation möglich ist (Chi et al., 2020). In „Automatic Generation of Multimedia Teaching Materials Based on Generative AI“ von Xu Chen und Di Wu wird auch generative KI eingesetzt und die resultierenden Grafiken mithilfe von *MoviePy* als Video gerendert.

Das vorgestellte Softwareartefakt wurde jedoch explizit für die Anwendungsdomäne der chinesischen Poesie entwickelt und bietet keine Möglichkeit, allgemeine informative Videos zu generieren (X. Chen & Wu, 2024). Den Einsatz von generativer KI zur Visualisierung von Musik wird von Liu et al. in „Generative Disco: Text-to-Video Generation for Music Visualization“ beschrieben. Hier werden zwar verschiedene Entwurfsmuster für die Videogenerierung vorgeschlagen, es können aber keine informativen Videos generiert werden. Außerdem ist das entwickelte Softwareartefakt eher als Werkzeug für die manuelle Erstellung eines Videos einzuordnen, da der Nutzer selbst die passenden Prompts und Bilder auswählt und deren Position im Video bestimmt (Liu et al., 2023). In „TaleBrush: Sketching Stories with Generative Pretrained Language Models“ von Chung et al. wird gezeigt, wie die Ausgabe von LLMs zur Erstellung von Drehbüchern für Geschichten genutzt werden kann. Das ist ein direkter Zusammenhang zu dieser Abschlussarbeit, da bei der Generierung des Storyboards auch eine Art Drehbuch erstellt wird, das dann mit Grafiken und Audiospuren zum vollständigen Storyboard wird. Die Generierung von weiteren audiovisuellen Inhalten oder gar eines Videos ist jedoch nicht Teil der verwandten Arbeit (Chung et al., 2022). Die letztgenannte verwandte Arbeit „Automatic Instructional Video Creation from a Markdown-Formatted Tutorial“ von Chi et al. beschäftigt sich mit der Erstellung informativer Tutorial-Videos. Sie grenzt sich von dieser Abschlussarbeit ab, da die Eingabe für die Software von Chi et al. schon eine *Step-by-Step*-Anleitung im Markdown-Format sein muss. Durch diese Einschränkung können so auch nur Videos generiert werden, die die *Schritt-für-Schritt*-Vorgehensweise des Tutorials darstellen (Chi et al., 2021).

Während in dem Bereich der automatisierten Erstellung von Inhalten per generativer KI aktiv geforscht wird und einige verwandte Arbeiten aus den letzten Jahren ähnliche Ansätze oder Vorgehensweisen gewählt haben, grenzt sich diese Abschlussarbeit durch die Kombination der genutzten Technologien deutlich ab.

8.3 Einschränkungen und Vorschläge zur zukünftigen Forschung

Wie in der Interpretation der Ergebnisse bereits beschrieben, gibt es verschiedene Einschränkungen des entwickelten Softwareartefakts. Diese Einschränkungen, eine Beurteilung der genutzten Methodik und Vorschläge zur zukünftigen Forschung werden in diesem Abschnitt diskutiert.

Verwendung eines Text-zu-Video-Modells. Zur Generierung einer informativen Story werden von dem Softwareartefakt mehrere passende Hintergrundbilder generiert, mit verschiedenen Kameraschwenkanimationen versehen und zu einem Kurzvideo aneinandergereiht. Es handelt sich also technisch um ein Video, der visuelle Effekt gleicht aber eher

einer Diashow. Zukünftig wäre es möglich, das Text-zu-Bild-Modell durch ein Text-zu-Video-Modell zu ersetzen, das für jedes Kapitel ein kurzes Hintergrundvideo generiert. Dies wäre beispielsweise mit *Sora* von *OpenAI* möglich, einem auf der Transformer-Architektur basierenden Diffusionsmodell. Mit diesem Modell können Bilder, aber auch qualitativ hochwertige Videos in unterschiedlichen Längen, Größen und Auflösungen generiert werden, deren Laufzeit bis zu einer Minute betragen kann (OpenAI et al., 2024). Durch den Einsatz von Hintergrundvideos anstatt Hintergrundbildern könnten die resultierenden Stories visuell deutlich interessanter werden.

Nachtrainieren der generativen KI-Modelle. Eine weitere Einschränkung des Softwareartefakts besteht darin, dass die generierten Hintergrundgrafiken häufig stilistisch nicht zueinander passen. Der standardmäßige Meta-Prompt, der für jede Bildgenerierung genutzt wird, gibt nur vor, dass das Bild fotorealistisch sein soll. Weil er zu der Liste der veränderbaren Parameter des Softwareartefakts gehört, kann der Prompt über die Benutzeroberfläche in den Einstellungen verändert werden. Aber selbst nach dem Erweitern des Prompts um verschiedene stilistische Merkmale, werden diese vom genutzten Diffusionsmodell häufig nicht umgesetzt. Eine Möglichkeit, zukünftig bessere und konsistentere Ergebnisse mit einem Diffusionsmodell zu erzielen, liegt im Nachtrainieren (finetuning) eines existierenden Modells wie *Stable Diffusion*. So kann das öffentliche Modell als Basis genutzt werden und eine vergleichsweise kleine Menge an eigenen Trainingsdaten dient zum Nachtrainieren der KI. Auf diese Weise kann das Text-zu-Bild-Modell für eine bestimmte Domäne getunt werden, was zu ästhetisch ansprechenderen Bildern in dem nachtrainierten Anwendungsfall führen kann (Esser et al., 2024). Dieses Finetuning ist auch für LLMs etabliert (Llama Team, 2024) und könnte im Rahmen des Softwareartefakts die Qualität des generierten Storyboards verbessern. Da die Prompts für das Text-zu-Bild-Modell auch von dem Sprachmodell generiert werden, könnte das Nachtrainieren des LLMs mit ausgewählten Bildgenerierungsprompts auch einen positiven Einfluss auf die Qualität und Konsistenz der Hintergrundbilder haben.

Verwendung alternativer KI-Modelle. Im aktuellen Stand des Softwareprototypen steht nur eine kleine Anzahl an generativen KI-Modellen zur Verfügung. Außerdem gibt es die Einschränkung, dass der Nutzer beim Erstellen eines neuen Projekts keine Auswahlmöglichkeit für die zu nutzenden Modelle hat. Welche Modelle für die verschiedenen Generierungsschritte genutzt werden, hängt von den aktiven Workern und deren hinterlegten Modellen ab. Durch die erweiterbare Worker-Architektur ließen sich mit geringem Aufwand weitere KI-Modelle hinzufügen. Dem Nutzer könnte dann eine Auswahl aller aktiven Worker und deren Modelle angezeigt werden, sodass er selbst über deren Verwendung bestimmen kann.

Verbesserte Untertitel. Für jedes Kapitel wird aus dem inhaltlichen Text eine Audiospur generiert, in der der Text vorgelesen wird. Bei beiden im Prototyp nutzbaren Modellen, *Bark* und *tts-1-hd* von *OpenAI*, wird dazu für jedes Kapitel eine eigene Audiodatei generiert und diese Dateien beim Rendering zusammengefügt. Keines der Modelle unterstützt jedoch die zusätzliche Ausgabe von Zeitstempel, wann die einzelnen Wörter genau in der Audiodatei ausgesprochen werden. Diese Informationen werden jedoch für die Darstellung von Untertiteln im gängigen Umfang von wenigen Worten pro Einblendung benötigt. So entstand bei der Entwicklung des Softwareartefakts die Einschränkung, dass für jedes Kapitel der komplette Text auf einmal als Untertitel angezeigt wird. Zukünftig wäre es möglich, ein TTS-Modell zu verwenden, das diese Meta-Informationen bereitstellt. Alternativ besteht die Möglichkeit, weiterhin beliebige TTS-Modelle zu verwenden und nach der Sprachgenerierung ein weiteres KI-Modell zur Transkription einzusetzen. Ein solches Modell, wie z. B. *Whisper* von *OpenAI*, erhält als Eingabe die Audiodatei und liefert eine detaillierte Abfolge der gesprochenen Wörter inklusive der genauen Zeitstempel für jedes Wort (Radford et al., 2022). So könnte die Darstellung der Untertitel deutlich verbessert werden.

Beurteilung der Methodik. Neben der Wahl von *Design Science Research* als übergeordnete Forschungsmethodologie wurden für die Bewertung des Softwareartefakts außerdem zwei verschiedene Evaluationsmethoden selektiert. Konkret wurde ein Experteninterview als qualitatives Verfahren zur Evaluation des Softwareprototypens genutzt und eine quantitative Befragung zur Evaluation der resultierenden Videoartefakte durchgeführt. Durch das Interview mit dem Experten konnten wichtige Informationen zum ‘normalen‘ Prozess der Content-Erstellung gewonnen werden und der entwickelte Softwareprototyp konnte durch die gemeinsame Nutzung ausführlich getestet werden. So wurde wertvolles Feedback für die getroffenen Designentscheidungen gesammelt und zu den vom Experten kritisierten Designentscheidungen konnten direkt gemeinsam Verbesserungsvorschläge formuliert werden. Außerdem konnten Effektivität, Effizienz und der Grad der Automatisierung des neuen Prozesses direkt dem normalen Prozess innerhalb der Organisation des Experten gegenübergestellt werden. Auch die Befragung der Schülerinnen und Schüler aus der Kernzielgruppe des Ausbildungsmarketings hat wichtige Erkenntnisse zur Evaluation geliefert. Dafür, dass eine quantitative Methode angewandt wurde, ist die Anzahl der gesammelten Datenpunkte mit 28 Werten pro Aussagen *A1* bis *A9* jedoch eher gering. Darüber hinaus handelt es sich bei allen Befragten um Schülerinnen und Schüler derselben Schule, was die Repräsentativität für die gesamte Zielgruppe der Ausbildungsinteressierten einschränken könnte.

9 Fazit

Das Forschungsziel dieser Abschlussarbeit ist es, ein prototypisches Softwareartefakt zu entwickeln und zu evaluieren, das aus strukturierten oder unstrukturierten Daten u. a. mittels eines Text-zu-Bild-Modells Kurzvideos in Story-Form generiert. Dazu wurde ein gestaltungsorientierter Ansatz als Forschungsmethodologie eingesetzt und die Abschlussarbeit nach dem Prozessmodell der *Design Science Research Methodology* aufgebaut.

Vor dem Design des Softwareartefakts wurden zu allen relevanten Themen und Technologien umfangreiche Hintergrundinformationen gesammelt und zusammengefasst. Besonders hervorzuheben sind hierbei die Grundlagenarbeiten zu LLMs, generativen Text-zu-Bild-Modellen und generativen Text-zu-Sprache-Modellen. Verbunden mit der Identifizierung verwandter Arbeiten wurde so eine *Knowledge Base* aufgebaut, die die Grundlage für den weiteren DSRM-Prozess lieferte. Aus dem Wissen zur Umgebung, in der das Softwareartefakt eingesetzt werden soll, wurden Anforderungen abgeleitet, aus denen Designentscheidungen getroffen wurden. Innerhalb dieser Designentscheidungen wurde eine modulare Architektur mit Workern entwickelt, die die Orchestrierung von KI-Modellen auf verschiedenen Systemen ermöglicht und diese in Echtzeit mit der unabhängig betriebenen Benutzeroberfläche kommunizieren lässt. So kann durch das automatisierte Zusammenspiel von generativen Sprachmodellen, Text-zu-Bild-Modellen und TTS-Modellen in wenigen Schritten ein Kurzvideo im Story-Format gerendert werden. Nach der detaillierten Beschreibung des Entwicklungsprozesses und wichtiger Iterationen während der Implementierung, wurde die Verwendung des entstandenen Softwareprototyp demonstriert. So konnte die technische Machbarkeit des Vorhabens unabhängig von der konkreten Wahl der generativen KI-Modelle gezeigt werden, indem sowohl quelloffene Modelle auf eigenem Hosting als auch kommerzielle Modelle in der Cloud verwendet und durch die iterative Anpassung der Prompts optimiert wurden. Die Allgemeingültigkeit des Softwareartefakts wurde untersucht, indem neben den Kurzvideos zu Ausbildungsstellenanzeigen auch Stories außerhalb der Anwendungsdomäne generiert wurden. Alle diese generierten Videoartefakte können im verlinkten Repository zu der Abschlussarbeit eingesehen werden. Im Rahmen der Evaluation wurden qualitative und quantitative Methoden eingesetzt. Durch eine Befragung von Schülerinnen und Schülern aus der Kernzielgruppe des Ausbildungsmarketings wurden die vom Softwareprototypen generierten Videoartefakte evaluiert. Ein anschließendes Experteninterview hat wichtige Erkenntnisse über die Vor- und Nachteile der Nutzung des Softwareartefakts geliefert, die auf die getroffenen Designentscheidungen rückgeführt wurden, um das *Design Knowledge* ausweiten zu können. Kritisiert wurde während der Evaluation vor allem die Qualität der Kurzvideos, während aber gleichzeitig durch das Experteninterview schon festgestellt werden konnte, wie diese im Rahmen einer Weiterentwicklung des Softwareartefakts verbessert

werden kann. Allgemein konnte durch beide Evaluationsarten festgestellt werden, dass die Software, auch in ihrem jetzigen prototypischen Stand, bereits einen Mehrwert bietet. In der Diskussion wurden die Ergebnisse interpretiert, indem unter anderem die Effektivität und Effizienz des neuen Videogenerierungsprozesses bestimmt wurde. Auch der erreichte Grad der Automatisierung und die Einsatzmöglichkeiten in Anwendungsdomänen außerhalb des Ausbildungsmarketings wurden diskutiert. Die Vorschläge zur zukünftigen Forschung beinhalten mehrere konkrete Möglichkeiten, das Softwareartefakt weiterzuentwickeln und zu verbessern. Dazu gehört beispielsweise das Nachtrainieren der genutzten generativen KI-Modelle für den konkreten Anwendungsbereich und das Verwenden eines Text-zu-Video-Modells zur Generierung von Hintergrundvideos anstelle von Hintergrundbildern. Der Einsatz eines Sprache-zu-Text-Transkriptionsmodells kann auch die Qualität der Untertitel verbessern.

Diese Arbeit beschreibt einen vielversprechenden ersten Schritt in der automatischen Generierung informativer Social Media Stories im Kurzvideoformat. Die prototypische Umsetzung zeigt, wie generative KI-Modelle den Prozess der Content-Erstellung vereinfachen können und skizziert, was mit dem anhaltenden Fortschritt im Bereich generativer KI zukünftig möglich werden könnte.

Literatur

Brock, A., Donahue, J., & Simonyan, K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ICLR 2019 Conference*. https://doi.org/10.48550/arxiv.1809.11096

Brocke, J. V., & Maedche, A. (2019). The DSR grid: six core dimensions for effectively planning and communicating design science research projects. *Electronic Markets*, *29*(3), 379–385. https://doi.org/10.1007/s12525-019-00358-7

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. *Neural Information Processing Systems*, *33*, 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Bundesagentur für Arbeit. (n. d.). *Ausbildungsmarkt - Statistik der Bundesagentur für Arbeit*. Verfügbar 6. April 2024 unter https://statistik.arbeitsagentur.de/DE/Navigation/Statistiken/Interaktive-Statistiken/Ausbildungsmarkt/Ausbildungsmarkt-Nav.html

Chen, L., & Rudnicky, A. (2022). Fine-Grained style control in Transformer-Based Text-To-Speech synthesis. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. https://doi.org/10.1109/icassp43922.2022.9747747

Chen, X., & Wu, D. (2024). Automatic Generation of Multimedia Teaching Materials Based on Generative AI: Taking Tang Poetry as an Example. *IEEE Transactions on Learning Technologies*, *17*, 1353–1366. https://doi.org/10.1109/tlt.2024.3378279

Chi, P., Frey, N., Panovich, K., & Essa, I. (2021). Automatic Instructional Video Creation from a Markdown-Formatted Tutorial. *34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*, 677–690. https://doi.org/10.1145/3472749.3474778

Chi, P., Sun, Z., Panovich, K., & Essa, I. (2020). Automatic Video Creation From a Web Page. *In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*, 279–292. https://doi.org/10.1145/3379337.3415814

Chung, J. J. Y., Kim, W., Yoo, K. M., Lee, H., Adar, E., & Chang, M. (2022). TaleBrush: Sketching Stories with Generative Pretrained Language Models. *CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3491102.3501819

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., & Rombach, R. (2024). Scaling Rectified Flow Transformers for

High-Resolution Image Synthesis. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2403.03206

Ghosh, D., Hajishirzi, H., & Schmidt, L. (2023). GenEval: An Object-Focused Framework for Evaluating Text-to-Image Alignment. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2310.11513

Hartmann, J., Exner, Y., & Domdey, S. (2023). The power of generative marketing: Can generative AI reach human-level visual marketing content? *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4597899

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.5555/2017212.2017217

Hugging Face Inc. (n. d. a). *Hugging Face - Text-to-Speech-Models*. Verfügbar 20. August 2024 unter https://huggingface.co/models?pipeline_tag=text-to-speech&sort=likes

Hugging Face Inc. (n. d. b). *TTS Arena: Benchmarking Text-to-Speech Models in the Wild*. Verfügbar 6. September 2024 unter https://huggingface.co/blog/arena-tts

Liu, V., Long, T., Raw, N., & Chilton, L. (2023). Generative Disco: Text-to-Video Generation for Music Visualization. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2304.08551

Llama Team, A. @. M. (2024). The Llama 3 Herd of Models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2407.21783

OpenAI. (2024, 6. August). *Introducing Structured Outputs in the API*. Verfügbar 3. September 2024 unter https://openai.com/index/introducing-structured-outputs-in-the-api

OpenAI, Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., & Ramesh, A. (2024, 15. Februar). *Video generation models as world simulators*. Verfügbar 5. November 2024 unter https://openai.com/research/video-generation-models-as-world-simulators

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/mis0742-1222240302

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2212.04356

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10674–10685. https://doi.org/10.1109/cvpr52688.2022.01042

Shehadeh, K., Arman, N., & Khamayseh, F. (2021). Semi-Automated Classification of Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools. *International Conference on Information Technology (ICIT)*, 527–532. https://doi.org/10.1109/icit52682.2021.9491698

Shi, F., Suzgun, M., Freitag, M., Wang, X., Srivats, S., Vosoughi, S., Chung, H. W., Tay, Y., Ruder, S., Zhou, D., Das, D., & Wei, J. (2022). Language Models are Multilingual Chain-of-Thought Reasoners. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2210.03057

Stack Exchange Inc. (2024, Mai). *2024 Stack Overflow Developer Survey*. Verfügbar 29. August 2024 unter https://survey.stackoverflow.co/2024/

Statista. (2018, Januar). *Umfrage zu verwendeten Recruiting-Maßnahmen in deutschen Unternehmen bis 2017*. Verfügbar 6. April 2024 unter https://de.statista.com/statistik/daten/studie/822717/umfrage/umfrage-zu-verwendeten-recruiting-massnahmen-in-deutschen-unternehmen/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *31st Conference on Neural Information Processing Systems (NIPS 2017)*, *31*, 5998–6008. https://arxiv.org/pdf/1706.03762v5

Abschließende Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit „ Entwicklung und Evaluation einer Applikation zur automatischen Erstellung informativer Social Media Stories“ selbständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Kaltenengers, den 13. November 2024

Jan Hillesheim

Einverständniserklärung

zur Prüfung meiner Arbeit mit einer Software zur Erkennung von Plagiaten

Name: Hillesheim **Vorname**: Jan
Matrikelnummer: 219201598 **Studiengang**: Informatik
Adresse: Hauptstraße 13, 56220 Kaltenengers
Titel der Arbeit: „ Entwicklung und Evaluation einer Applikation zur automatischen Erstellung informativer Social Media Stories“

Was ist ein Plagiat? Als ein Plagiat wird eine Übernahme fremden Gedankengutes in die eigene Arbeit angesehen, bei der die Quelle, aus der die Übernahme erfolgt, nicht kenntlich gemacht wird. Es ist dabei unerheblich, ob z.B. fremde Texte wörtlich übernommen werden, nur Strukturen (z.B. argumentative Figuren oder Gliederungen) aus fremden Quellen entlehnt oder Texte aus einer Fremdsprache übersetzt werden.

Softwarebasierte Überprüfung Alle Bachelor- und Masterarbeiten werden vom Prüfungsamt mit Hilfe einer entsprechenden Software auf Plagiate geprüft. Die Arbeit wird zum Zweck der Plagiatsüberprüfung an einen Software-Dienstleister übermittelt und dort auf Übereinstimmung mit anderen Quellen geprüft. Zum Zweck eines zukünftigen Abgleichs mit anderen Arbeiten wird die Arbeit dauerhaft in einer Datenbank gespeichert. Der Studierende erklärt sich damit einverstanden, dass allein zum beschriebenen Zweck der Plagiatsprüfung die Arbeit dauerhaft gespeichert und vervielfältigt werden darf. Das Ergebnis der elektronischen Plagiatsprüfung wird den Gutachtern mitgeteilt.

Sanktionen Liegt ein Plagiat vor, ist dies ein Täuschungsversuch i.S. der Prüfungsordnung, durch den die Prüfungsleistung als „nicht bestanden“ gewertet wird. Es erfolgt eine Mitteilung an das Prüfungsamt und die dortige Dokumentation. In schwerwiegenden Täuschungsfällen kann der Prüfling von der Prüfung insgesamt ausgeschlossen werden. Dies kann unter Umständen die Exmatrikulation bedeuten. Plagiate können auch nach Abschluss des Prüfungsverfahrens und Verleihung des Hochschulgrades zum Entzug des erworbenen Grades führen.

Hiermit erkläre ich, dass ich die obigen Ausführungen gelesen habe und mit dem Verfahren zur Aufdeckung und Sanktionierung von Plagiaten einverstanden bin.

Kaltenengers, den 13. November 2024

Jan Hillesheim