IT Projekt Arbeit 2024

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur schulischen Projektarbeit

Jamstack Portfolio

Realisierung einer Jamstack Portfolio als Reacteinzelseiten Webapplikation

Abgabedatum: Neuburg, den 09.02.2024

Prüfungsbewerber:

Kacper Sobczak

Amalienstraße 22 (1/3)

86633 Neuburg an der Donau

Ausbildungsbetrieb:

LUSINI International GmbH

Hettlinger Str. 9

86637 Wertigen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

- **API:** Application Programming Interface
- **JAMstack:** JavaScript, APIs, Markup
- **CSS:** Cascading Style Sheets
- **UI/UX:** User Interface/User Experience
- **HTML:** Hypertext Markup Language
- **SSR:** Server-Side Rendering
- **CMS:** Content Management System
- **DSGVO:** Datenschutz-Grundverordnung (EU-Datenschutzgesetz)
- **UML:** Unified Modeling Language

- **CDN:** Content Delivery Network

Technologie- und Softwareschnittstellenverzeichnis

- **NEXT.JS:** NEXT.JS ist ein React-Framework für die Entwicklung von Webanwendungen, das Funktionen wie serverseitiges Rendern und Routenpräfixe bietet. Es ermöglicht die Erstellung von schnellen und dynamischen Single-Page-Anwendungen.
- **Vercel:** Vercel ist eine Cloud-Plattform, die für das Deployment (oder Ausrollen) von Webanwendungen optimiert ist. Sie unterstützt verschiedene Frameworks wie NEXT.JS und bietet Funktionen wie automatisches Deployment und globale CDN-Bereitstellung.
- **Sanity.io:** Sanity.io ist ein Headless-CMS (Content Management System), das eine flexible Datenstruktur und eine leistungsstarke API bietet. Es ermöglicht die Erstellung von Inhaltsverwaltungssystemen für verschiedene Plattformen und Geräte.
- **Framer Motion:** Framer Motion ist eine Animation-Library für React-Anwendungen. Sie bietet eine einfache API für die Erstellung von Animationen und Übergängen in Benutzeroberflächen.
- **React:** React ist eine JavaScript-Bibliothek für die Entwicklung von Benutzeroberflächen. Sie wird verwendet, um interaktive und dynamische Webanwendungen zu erstellen. React verwendet eine Komponenten-basierte Architektur, die die Wiederverwendbarkeit und Modularität fördert.
- **Tailwind CSS:** Tailwind CSS ist ein Utility-First-CSS-Framework, das die Erstellung von benutzerdefinierten und responsiven Benutzeroberflächen erleichtert. Es bietet vorgefertigte Utility-Klassen für häufig verwendete CSS-Stile und ermöglicht eine schnellere Entwicklung von Frontend-Projekten.
- **Headless CMS:** Ein Headless-CMS ist ein Content Management System, das den Inhalt von der Präsentation trennt. Es bietet eine API zum Abrufen und Verwalten von Inhalten, während die Darstellung des Inhalts über verschiedene Plattformen und Geräte flexibel gestaltet werden kann.
- **CDN:** Ein Content Delivery Network, das Inhalte wie Bilder, Videos und Skripte an Endbenutzer in verschiedenen geografischen Regionen verteilt, um die Ladezeiten zu verbessern und die Verfügbarkeit zu erhöhen.

1. Einleitung & Vorwort

Die vorliegende Projektarbeit stellt die Ergebnisse und Fortschritte meiner Ausbildung als Fachinformatiker Anwendungsentwicklung dar. Das Ziel dieses Projekts war die Konzeption und Entwicklung eines JAMstack Portfolios unter Verwendung modernster Technologien und bewährter Praktiken.

In der dynamischen Welt der Webentwicklung sind JAMstack-Architekturen aufgrund ihrer Effizienz und Leistungsfähigkeit zunehmend beliebt. Die Entscheidung, einen Portoflio auf JAMstack-Basis zu entwickeln, ermöglichte nicht nur eine schnelle und sichere Benutzererfahrung, sondern stellte auch meine Fähigkeiten in der Anwendungsentwicklung auf die Probe.

Dieses Projekt umfasst die gesamte Bandbreite der Softwareentwicklung, von der Konzeption und Planung über die Implementierung bis hin zu Tests und Qualitätskontrolle. Dabei wurden innovative Technologien wie React, Tailwind CSS, Framer Motion und Sanity.io genutzt, um eine moderne, ansprechende und skalierbare Webanwendung zu schaffen.

Abschließend möchte ich betonen, dass dieses Projekt nicht nur ein Ergebnis meiner Ausbildung ist, sondern auch eine fortlaufende Reise darstellt, in der ich meine Fähigkeiten erweitern und immer auf dem neuesten Stand der Technologie bleiben konnte.

Ich lade den Leser und Prüfer ein, diese Projektarbeit als Einblick in die Welt der JAMstack-Entwicklung und als Zeugnis meines Engagements in der Anwendungsentwicklung zu betrachten. Es ist mein Wunsch, dass dieses Projekt nicht nur informiert, sondern auch inspiriert und möglicherweise als Grundlage für zukünftige Innovationen in der Webentwicklung dient.

2. Projektbeschreibung

2.1 Projektumfeld

Das vorliegende Projekt hat zum Ziel, ein modernes Portfolio mit Hilfe von JAMstack-Technologien zu entwickeln. Hierbei werden Schlüsseltechnologien wie NEXT.JS, Framer Motion, Tailwind CSS, Sanity.io und React eingesetzt. Das Portfolio wird als Headless-CMS mit Sanity.io im Backend betrieben, was es ermöglicht, die Inhalte von überall auf der Welt einfach und effizient zu aktualisieren.

Die herausragenden Merkmale dieses Projekts umfassen die Optimierung von NEXT.JS, um eine blitzschnelle Website zu erstellen, die in Millisekunden geladen wird. Zusätzlich werden flüssige Animationen mithilfe von Framer Motion implementiert, um eine einzigartige und beeindruckende Benutzeroberfläche und -erfahrung zu schaffen. Die Verwendung von Tailwind CSS ermöglicht die Gestaltung einer modernen, ansprechenden und responsiven Website. Schließlich wird das Portfolio auf Vercel bereitgestellt, um die Fähigkeiten und den Fortschritt des Entwicklers der Welt präsentieren zu können.

Die Kombination dieser Technologien und Funktionen wird ein eindrucksvolles Portfolio hervorbringen und zugleich wertvolle praktische Erfahrungen im Umgang mit zeitgemäßen Webentwicklungstools vermitteln.

2.2 Projektziel

Das Hauptziel dieses Projekts besteht darin, ein Portfolio zu erstellen, das als effektives Instrument dient, um die individuellen Fähigkeiten und Kompetenzen eines jeden Nutzers hervorzuheben und sie im besten Licht zu präsentieren. Das Portfolio wird in erster Linie als Plattform zur Selbstvermarktung und Selbstpräsentation konzipiert, mit dem Fokus auf die Förderung von beruflichen Fähigkeiten und Talenten.

In der heutigen Geschäftswelt, in der Wettbewerb und Selbstvermarktung entscheidend sind, ermöglicht dieses Portfolio jedem Benutzer, seine Qualifikationen, Projekte und Erfahrungen auf professionelle Weise darzustellen. Das Projekt zielt darauf ab, ein attraktives, informatives und benutzerfreundliches Portfolio zu erstellen, das die Aufmerksamkeit von potenziellen Arbeitgebern, Kunden oder Geschäftspartnern auf sich zieht.

Durch die Nutzung der JAMstack-Technologien, NEXT.JS, Framer Motion, Tailwind CSS und Sanity.io wird ein Portfolio geschaffen, das nicht nur ästhetisch ansprechend ist, sondern auch in Sekundenbruchteilen geladen wird. Damit wird sichergestellt, dass die Benutzererfahrung reibungslos und beeindruckend ist.

Dieses Portfolio wird somit dazu beitragen, die beruflichen Chancen zu verbessern, indem es die Sichtbarkeit und Glaubwürdigkeit der individuellen Fähigkeiten und Leistungen steigert.

2.3 Projektbegründung

Die Notwendigkeit, sich in der digitalen Welt effektiv zu präsentieren und berufliche Chancen zu nutzen, macht die Entwicklung eines modernen Portfolios unerlässlich. Dieses Projekt ermöglicht die Schaffung eines ansprechenden, schnellen und benutzerfreundlichen Portfolios, um individuelle Fähigkeiten und Leistungen optimal zu präsentieren und zu fördern.

2.4 Projektschnittstellen

Dieses Projekt weist mehrere wichtige Schnittstellen auf, die eine reibungslose Umsetzung sicherstellen:

1. *Frontend-Entwicklung:* Hier wird NEXT.JS verwendet, um die Benutzeroberfläche zu erstellen. Die Integration von Framer Motion und Tailwind CSS trägt zur Gestaltung und Animation bei.

2. *Backend-Entwicklung:* Sanity.io wird als Headless-CMS genutzt, um die Inhalte zu verwalten. Die Schnittstelle zwischen Frontend und Backend ermöglicht die nahtlose Aktualisierung von Portfolio-Inhalten.

3. Benutzererfahrung (UX/UI): Framer Motion und Tailwind CSS sind Schlüsselkomponenten, um eine ansprechende Benutzererfahrung und Benutzeroberfläche zu schaffen. Hier ist die enge Zusammenarbeit zwischen Frontend-Entwicklern und Designern entscheidend.

4. Hosting und Bereitstellung: Die Verbindung mit Vercel ermöglicht die einfache Bereitstellung und Wartung des Portfolios für die weltweite Präsentation.

5. Inhaltsmanagement: Die Schnittstelle zwischen dem Portfolio-Ersteller und Sanity.io gewährleistet die einfache Verwaltung von Texten, Bildern und anderen Inhalten.

Diese Schnittstellen sind von entscheidender Bedeutung, um die verschiedenen Aspekte des Projekts zu koordinieren und ein reibungsloses Funktionieren sicherzustellen. Die effektive Zusammenarbeit und Kommunikation zwischen den Teams, die an diesen Schnittstellen arbeiten, ist entscheidend für den Projekterfolg

2.5 Projektabgrenzung

Die Projektabgrenzung definiert klare Grenzen und beschränkt den Fokus auf die primären Aspekte des Portfolios. In diesem Projekt sind die folgenden Bereiche explizit ausgeschlossen:

1. SEO (Suchmaschinenoptimierung): Die Optimierung für Suchmaschinen, um die Sichtbarkeit in Suchergebnissen zu steigern, ist nicht Teil des Projekts. Dies erfordert eine spezielle Expertise und kann in einem separaten Schritt behandelt werden.

2. Marketingstrategien: Das Projekt konzentriert sich auf die Erstellung und Präsentation des Portfolios selbst, nicht auf Marketingstrategien zur Steigerung der Reichweite oder Kundenakquise.

3. Übersetzungen: Es wird keine Unterstützung für die Übersetzung des Portfolios in andere Sprachen bereitgestellt. Das Portfolio wird ausschließlich in englischer Sprache verfügbar sein.

Die klare Abgrenzung dieser Aspekte ermöglicht es, sich auf die Entwicklung des Portfolios und seiner Kernelemente zu konzentrieren, ohne von anderen Aufgaben abgelenkt zu werden. Dies gewährleistet eine effiziente Umsetzung des Hauptziels des Projekts.

3. Projektplanung

3.1 Projekt Struktur Plan & Projektphasen

Das Projekt kann in mehrere Phasen aufgeteilt werden, um die insgesamt investierten 46 Stunden effizient zu nutzen:

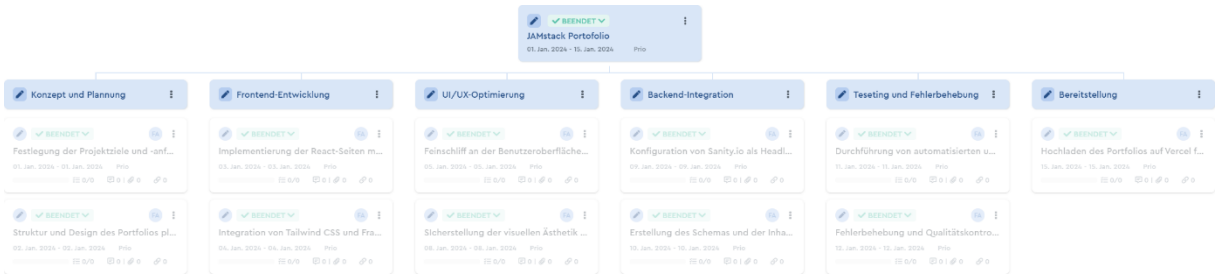

Abbildung 1: Projekt Struktur Plan

1. Konzept und Planung (4 Stunden): In dieser Phase wird die Projektrichtung festgelegt, einschließlich der Entscheidung über Design, Struktur und Funktionen des Portfolios.

2. Frontend-Entwicklung (15 Stunden): Diese Phase beinhaltet die Umsetzung der Benutzeroberfläche mit NEXT.JS, Framer Motion und Tailwind CSS. Die grundlegende Struktur der Website wird erstellt.

3. Backend-Integration (5 Stunden): Hier wird die Anbindung an Sanity.io als Headless-CMS durchgeführt, um die Verwaltung von Inhalten zu ermöglichen.

4. UI/UX-Optimierung (5 Stunden): Feinschliff an der Benutzeroberfläche und Integration von Framer Motion, um ein beeindruckendes visuelles Erlebnis zu schaffen.

5. Testing und Fehlerbehebung (12 Stunden): Gründliche Überprüfung der Website, Identifizierung und Behebung von Fehlern und Verbesserungen.

6. Bereitstellung (5 Stunden): Die Abschlussphase, die die Bereitstellung des Portfolios auf Vercel und die Sicherstellung eines reibungslosen Betriebs umfasst.

3.2 Grobe Zeitplanung

Die grobe Zeitplanung für das 46-Stunden-Projekt könnte folgendermaßen aussehen:

1. Konzept und Planung: 4 Stunden

- Festlegung der Projektziele und -anforderungen.
- Struktur und Design des Portfolios planen.

2. Frontend-Entwicklung: 15 Stunden

- Implementierung der React-Seiten mit NEXT.JS.
- Integration von Tailwind CSS und Framer Motion für das Design und die Animationen.

3. Backend-Integration: 5 Stunden

- Konfiguration von Sanity.io als Headless-CMS.
- Erstellung des Schemas und der Inhaltsstruktur.

4. UI/UX-Optimierung: 5 Stunden

- Feinschliff an der Benutzeroberfläche und den Animationen.
- Sicherstellung der visuellen Ästhetik und Benutzerfreundlichkeit.

5. Testing und Fehlerbehebung: 12 Stunden

- Durchführung von automatisierten und manuellen Tests.

- Fehlerbehebung und Qualitätskontrolle.

6. Bereitstellung: 5 Stunden

- Hochladen des Portfolios auf Vercel für die Live-Präsentation.

3.3 Detaillierte Zeitplanung

Phase	Unterpunkte	Geschätzte Zeit (in Stunden)
Konzept und Planung	Projektziele festlegen	2
	Design und Struktur planen	2
Frontend-Entwicklung	Implementierung von React-Seiten	10
	Integration von Tailwind CSS	3
	Hinzufügen von Framer Motion Animationen	2
Backend-Integration	Konfiguration von Sanity.io	2
	Erstellung des Schemas und der Struktur	3
UI/UX-Optimierung	Feinschliff an der Benutzeroberfläche	4
	Sicherstellung der visuellen Ästhetik	1
Testing und Fehlerbehebung	Automatisierte Tests	2
	Manuelle Tests	3
	Fehlerbehebung und Qualitätskontrolle	7
Bereitstellung	Hochladen des Portfolios auf Vercel	5
Gesamt		46

Tabelle 1: Detaillierte Zeitplanung

Diese detaillierte Zeitplanung zeigt die geschätzte Zeit, die für jede Unterpunkte aufgewendet wird, um das gesamte 46-Stunden-Budget effizient zu nutzen. Es ist wichtig, diese Zeitplanung während des Projekts zu überwachen und anzupassen, um sicherzustellen, dass die Ziele erreicht werden.

3.4 Gantt-Diagramm der Arbeitspakten

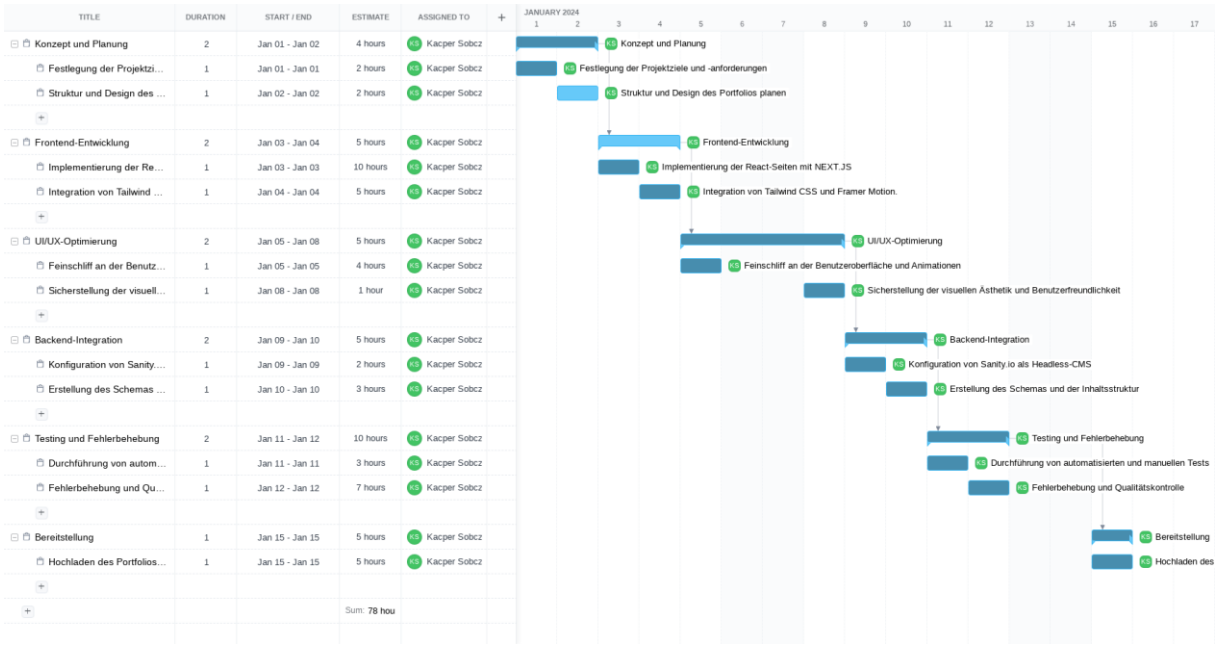

Abbildung 2: Gantt-Diagram

Die Zeitschätzung in Stunden kann variieren, abhängig von deinen individuellen Fähigkeiten und Erfahrungen in den jeweiligen Technologien. Es ist wichtig, die Zeit effizient zu nutzen und regelmäßige Fortschrittsüberprüfungen vorzunehmen, um das Projekt erfolgreich abzuschließen.

3.5 Ressourcenplanung

Die Ressourcenplanung für dieses Projekt umfasst:

1. Laptop des Unternehmens: Ein eigener Laptop von der Firma wird für die Durchführung des gesamten Projekts genutzt. Dieser Laptop sollte leistungsfähig genug sein, um die Entwicklung und Bereitstellung von Webanwendungen zu unterstützen.

2. Internetverbindung: Eine zuverlässige Internetverbindung ist entscheidend, um auf Online-Dienste und Ressourcen zuzugreifen, Code-Repositorys zu aktualisieren und das Portfolio auf Vercel bereitzustellen.

3. Online-Dienste und Tools: Online-Dienste wie GitHub für die Versionsverwaltung und Vercel für die Bereitstellung des Portfolios werden verwendet. Zusätzlich wird Sanity.io als Headless-CMS eingesetzt. Diese Dienste spielen eine entscheidende Rolle in der Entwicklung, Versionszontrolle und im Hosting des Projekts.

4. Entwicklungsumgebung: Für die Einrichtung der Entwicklungsumgebung, die die Verwendung von NEXT.JS, Framer Motion, Tailwind CSS und React ermöglicht, sind verschiedene Software und Tools erforderlich. Dies beinhaltet die Installation von spezifischen Entwicklungsumgebungen und Bibliotheken, um effizient an deinem Projekt arbeiten zu können.

5. Zeit und Arbeitskraft: Die Ressource Zeit ist auf die insgesamt investierten 46 Stunden begrenzt. Die Arbeitskraft und Fähigkeiten werden für die Konzeption, Entwicklung, Tests und Bereitstellung des Portfolios benötigt.

Die effiziente Nutzung dieser Ressourcen ist entscheidend, um das Projekt erfolgreich abzuschließen. Es ist wichtig sicherzustellen, dass alle notwendigen Tools und Dienste verfügbar sind und die Arbeitszeit optimiert wird, um das Ziel innerhalb des vorgegebenen Zeitrahmens zu erreichen.

3.6 Entwicklungsprozess

Der gesamte Entwicklungsprozess dieses Projekts wird eigenständig durchgeführt. Von der Konzeption über die Frontend- und Backend-Entwicklung bis hin zur UI/UX-Optimierung, Tests und Bereitstellung erfolgt alles allein. Dies ermöglichten eine ganzheitliche Erfahrung und eine enge Kontrolle über jeden Aspekt des Projekts, was zu einer gründlichen Umsetzung führt.

4. Wirtschaftlichkeitsbetrachtung

4.1. Projektkosten:

Die Wirtschaftlichkeitsbetrachtung des Projekts zeigt, dass die Kosten aufgrund der Nutzung von kostenlosen Ressourcen und Tools minimal sind. Da das Projekt auf dem Free Tier der verwendeten Dienste basiert und keine Skalierung erwartet wird, entstehen keine direkten Kosten für Hosting, Datenbanknutzung oder Entwicklerlizenzen.

Die Hauptinvestition besteht aus der Arbeitszeit, die für die Konzeption, Entwicklung, Tests und Bereitstellung aufgewendet wird. Diese Kosten sind intern und hängen von den Arbeitsstunden ab, die in das Projekt fließen.

Angenommen, der durchschnittliche Stundenlohn eines Azubis beträgt beispielsweise 15 Euro pro Stunde. Wenn insgesamt 46 Stunden in das Projekt investierst, ergibt sich eine grobe Kostenschätzung wie folgt:

$$Kosten = Studen * Studenlohn$$

$$Kosten = 46\ Stunden * 15\ Euro/Stunde$$

$$Kosten = 690\ Euro$$

Die geschätzten Kosten für die Arbeitszeit im Projekt würden also etwa 690 Euro betragen. Bitte beachte, dass dies eine vereinfachte Kalkulation ist und individuelle Faktoren wie Steuern, Sozialleistungen oder zusätzliche Ressourcen nicht berücksichtigt.

Die Entscheidung, auf kostenfreie Ressourcen zurückzugreifen, stellt sicher, dass das Projekt wirtschaftlich ist und keine finanzielle Belastung darstellt. Die Wirtschaftlichkeit des Projekts wird durch die Verwendung effizienter Open-Source-Technologien und Tools maximiert.

5. Analyse

5.1 IST-Analyse

Die IST-Analyse basiert auf einer vorläufigen HTML- und CSS-Preview-Version des Portfolios. Die aktuelle Vorschau ist statisch und begrenzt in ihren Funktionen. Sie präsentiert grundlegende Informationen, jedoch ohne die Möglichkeit der einfachen Aktualisierung von Inhalten. Das Design ist einfach und es fehlen flüssige Animationen und eine beeindruckende Benutzeroberfläche.

Diese IST-Analyse zeigt den Ausgangspunkt des Projekts und die Notwendigkeit einer umfassenderen Entwicklung. Sie verdeutlicht, dass die Integration von NEXT.JS, Framer Motion und Tailwind CSS erforderlich ist, um eine dynamischere und ansprechendere Website zu schaffen, während Sanity.io als Headless-CMS die Aktualisierung und Verwaltung von Inhalten erheblich vereinfacht.

6. Entwurf

6.1 Ablaufplan der Applikation

Der Ablaufplan der Anwendung umfasst die Erstellung einer modernen Portfolio-Website. Der Prozess beginnt mit der Konzeption und Planung der Website-Struktur und -Funktionalitäten. Anschließend erfolgt die Frontend-Entwicklung unter Verwendung von NEXT.JS, Framer Motion und Tailwind CSS. Die Backend-Integration mit Sanity.io ermöglicht die Inhaltsverwaltung. UI/UX-Optimierung, Tests und schließlich die Bereitstellung auf Vercel vervollständigen den Ablauf. Dieser Plan stellt sicher, dass das Portfolio effizient entwickelt wird und den Anforderungen entspricht.

6.2 Graphisches Abbildungen der Ablaufplan & Ihre Prioritäten

Complete

	ASSIGNED TO	PRIORITY	ESTIMATE	STATUS
Konzept und Planung 2	Kacper Sobczak	Blocker	4 hours	Complete
Festlegung der Projektziele und -anforderungen	Kacper Sobczak	High	2 hours	Complete
Struktur und Design des Portfolios planen	Kacper Sobczak	High	2 hours	Complete
Frontend-Entwicklung 2	Kacper Sobczak	Highest	5 hours	Complete
Implementierung der React-Seiten mit NEXT.JS	Kacper Sobczak	Highest	10 hours	Complete
Integration von Tailwind CSS und Framer Motion.	Kacper Sobczak	High	5 hours	Complete
Backend-Integration 2	Kacper Sobczak	Blocker	5 hours	Complete
Konfiguration von Sanity.io als Headless-CMS	Kacper Sobczak	Low	2 hours	Complete
Erstellung des Schemas und der Inhaltsstruktur	Kacper Sobczak	High	3 hours	Complete
UI/UX-Optimierung 2	Kacper Sobczak	Lowest	5 hours	Complete
Feinschliff an der Benutzeroberfläche und Animationen	Kacper Sobczak	Low	4 hours	Complete
Sicherstellung der visuellen Ästhetik und Benutzerfreundlichkeit	Kacper Sobczak	Lowest	1 hour	Complete
Testing und Fehlerbehebung 2	Kacper Sobczak	Low	10 hours	Complete
Durchführung von automatisierten und manuellen Tests	Kacper Sobczak	Lowest	3 hours	Complete
Fehlerbehebung und Qualitätskontrolle	Kacper Sobczak	Highest	7 hours	Complete
Bereitstellung 1	Kacper Sobczak	High	5 hours	Complete
Hochladen des Portfolios auf Vercel für Live-Präsentation in Internet.	Kacper Sobczak	Blocker	5 hours	Complete

Abbildung 3: Ablaufplans und ihrer Prioritäten

6.3 Auswahl des Frontend-Frameworks

Die Auswahl des Front-End-Frameworks ist entscheidend für den Erfolg des Projekts. In diesem Fall wurde NEXT.JS als Framework gewählt. NEXT.JS bietet viele Vorteile, die für die Erstellung eines modernen Portfolios von hoher Relevanz sind.

Erstens ermöglicht NEXT.JS eine effiziente Serverseitige Rendern (SSR), was zu einer schnellen Seitenladezeit führt. Dies ist besonders wichtig, um die Zielsetzung des Projekts zu erreichen, eine Website, die in Millisekunden geladen wird.

Zweitens ist NEXT.JS eng mit React verbunden, was eine hohe Flexibilität und Reusability des Codes ermöglicht. Dies ist wichtig, um die Benutzeroberfläche und die Funktionalitäten des Portfolios effizient zu entwickeln.

Drittens bietet NEXT.JS optimale SEO-Unterstützung und eine hervorragende Entwicklerfreundlichkeit. Dies ermöglicht eine bessere Sichtbarkeit in Suchmaschinen und vereinfacht die Entwicklungsprozesse.

Insgesamt ist NEXT.JS eine ausgezeichnete Wahl, um die Projektziele zu erreichen und ein modernes, leistungsstarkes Portfolio zu erstellen.

6.4 Datenbankmodell (Sanity)

Die Entscheidung, Sanity.io als Headless-CMS zu nutzen, bringt verschiedene Vorteile mit sich. Sanity.io bietet eine flexible, schema-agnostische Datenbankstruktur, die es ermöglicht, Inhalte in einer intuitiven und hierarchischen Form zu organisieren. Dies erleichtert die Verwaltung von Projektdaten und die Aktualisierung von Inhalten erheblich.

Die Wahl von Sanity.io als Datenbankmodell ermöglicht es, Texte, Bilder und andere Inhalte nahtlos zu bearbeiten und bereitzustellen, ohne tiefgreifende Entwicklungsarbeiten durchführen zu müssen. Die hohe Benutzerfreundlichkeit und die kollaborativen Funktionen von Sanity.io erleichtern die Zusammenarbeit im Team und die Pflege des Portfolios.

Darüber hinaus unterstützt Sanity.io eine Vielzahl von Datenformaten und eröffnet somit die Möglichkeit zur Erweiterung und Anpassung der Datenstruktur im Laufe des Projekts. Insgesamt ermöglicht Sanity.io eine effiziente Verwaltung der Portfolio-Inhalte, was die Entscheidung für dieses Datenbankmodell begründet.

7. Implementierung

7.1 Implementierung des Frontends

7.1.1 Implementierung der React-Seiten

Die Implementierung des Front-Ends beginnt mit der Erstellung der React-Seiten in NEXT.JS. Jede Seite entspricht einem spezifischen Bereich oder einer Funktion des Portfolios. Die React-Komponenten werden so entwickelt, dass sie die visuelle Gestaltung und Interaktion gemäß den Designanforderungen erfüllen.

Der Einsatz von NEXT.JS ermöglicht das Serverseitige Rendern (SSR) der Seiten, was eine schnelle und benutzerfreundliche Erfahrung gewährleistet. Dies ist besonders wichtig, um die Zielsetzung des Projekts zu erreichen, eine Website, die in Millisekunden geladen wird.

Während der Implementierung der React-Seiten wird Tailwind CSS verwendet, um das Styling der Seiten zu erleichtern und die Erstellung von responsiven und ästhetisch ansprechenden Layouts zu unterstützen. Dies ermöglicht eine effiziente und konsistente Gestaltung der gesamten Website.

Die Implementierung der React-Seiten ist ein zentraler Schritt in der Entwicklung des Portfolios, um sicherzustellen, dass die Benutzeroberfläche den Anforderungen an ein modernes, ansprechendes Design entspricht und eine reibungslose Interaktion bietet.

Die Implementierung der React-Komponenten und NEXT.JS-Features ist ein zentraler Aspekt bei der Entwicklung des Portfolios. Sie tragen zur Erstellung einer ansprechenden und reaktiven Benutzeroberfläche bei. Im Folgenden werden einige ausgewählte Elemente und Module aufgeführt:

1. Pages in NEXT.JS: NEXT.JS verwendet das Konzept von "Seiten" (Pages), um Routing und Serverseitiges Rendern zu ermöglichen. Jede Seite entspricht einem React-Komponentenmodul und wird im *pages*-Verzeichnis erstellt. Zum Beispiel wird die Startseite als *index.js* erstellt, und andere Seiten können als separate Dateien angelegt werden, um die Benutzeroberfläche zu organisieren.

2. React-Komponenten: Die verschiedenen Teile der Website werden als React-Komponenten erstellt, um die Wiederverwendbarkeit und die strukturierte Entwicklung zu fördern. Beispielsweise können Header-, Footer-, Projektlisten- und Erfahrung-Komponenten erstellt werden.

3. Routen- und Link-Handling: NEXT.JS bietet das *Link*-Modul, um die Navigation zwischen Seiten zu ermöglichen. Dies erleichterten das Routing und das Erstellen von Links innerhalb der Anwendung.

4. Daten holen mit getStaticProps() und getServerSideProps(): NEXT.JS bietet Funktionen wie *getStaticProps()* und *getServerSideProps()*, um Daten zur Laufzeit zu holen und sie in die React-Komponenten einzufügen. Dies ermöglicht die Integration von Inhalten aus Sanity.io in die Seiten des Portfolios, in den Projekt wurde es auf die *getStaticProps()* entschieden, da die Funktionität die Anforderung des Projektes erfüllt.

5. Dynamische Routen: Mit NEXT.JS können dynamische Routen erstellt werden, um z. B. Einzelprojektseiten basierend auf Daten aus der Datenbank zu generieren. Dies ermöglicht die Darstellung individueller Projektinformationen.

6. Layout- und Styling-Frameworks: Tailwind CSS wird verwendet, um das Styling und die Layoutgestaltung der Komponenten zu erleichtern. Dies ermöglichten eine effiziente Gestaltung und Anpassung des Designs des Portfolios.

7. Animationen mit Framer Motion: Framer Motion wird eingesetzt, um flüssige Animationen in die Benutzeroberfläche zu integrieren. Beispielsweise können Animationen beim Laden von Projekten oder beim Scrollen der Seite verwendet werden.

8. State-Management: React bietet Möglichkeiten zur Verwaltung des Anwendungsstatus. Je nach Bedarf können Komponenten mit oder ohne State (Zustand) entwickelt werden.

Diese ausgewählten Elemente und Module zeigen die Integration von React und NEXT.JS in die Entwicklung des Portfolios. Die Verwendung dieser Technologien ermöglicht die Erstellung einer dynamischen und ansprechenden Website, die die gesteckten Ziele erreicht. Die Implementierung ist ein kontinuierlicher Prozess, der die Zusammenarbeit von Frontend-Entwicklern und Designern erfordert, um eine beeindruckende Benutzeroberfläche zu erstellen.

7.1.2 Implementierung von Tailwind CSS und Framer Motion

Die Implementierung von Tailwind CSS und Framer Motion ist ein entscheidender Schritt in der Gestaltung und Verbesserung der Benutzererfahrung des Portfolios.

Tailwind CSS vereinfacht die Entwicklung von Frontend-Styles erheblich. Durch die Verwendung von vordefinierten Klassen können Styles effizient angewendet und angepasst werden, was Zeit spart, und eine konsistente Gestaltung gewährleistet. Die Integration von Tailwind CSS ermöglicht die Erstellung einer modernen und responsiven Website, die den neuesten Designstandards entspricht.

Framer Motion wird verwendet, um flüssige Animationen in das Portfolio zu integrieren. Dies verleiht der Benutzeroberfläche eine einzigartige und beeindruckende Qualität. Animationen erhöhen die Interaktivität und machen das Portfolio visuell ansprechend.

Die Implementierung von Tailwind CSS und Framer Motion trägt wesentlich zur Optimierung der Benutzererfahrung und zur visuellen Attraktivität des Portfolios bei. Diese Tools sind integraler Bestandteil der Frontend-Entwicklung und tragen dazu bei, dass das Portfolio die gesteckten Ziele erreicht.

7.2 Implementierung des Backend

1. Sanity.io als Headless-CMS:

- Für die Verwaltung der Inhalte wird Sanity.io als Headless-CMS verwendet. Es handelt sich um eine cloud-basierte Datenbank und ein Content-Management-System, das es ermöglicht, strukturierte Daten für das Portfolio zu speichern und zu aktualisieren.

2. Schema-Definition:

- Mit Sanity.io wird ein Schema definiert, das die Datenstruktur für das Portfolio festlegt. Hier können Typen für Projekte, Texte, Bilder und andere Inhalte erstellt werden. Das Schema stellt sicher, dass die Daten konsistent und strukturiert sind.

3. Inhaltsbearbeitung:

- Sanity.io bietet eine benutzerfreundliche Oberfläche für die Inhaltsbearbeitung. Redakteure und Autoren können Texte, Bilder und andere Inhalte einfach hinzufügen, bearbeiten und löschen. Dies erleichtert die Aktualisierung des Portfolios erheblich.

4. API-Nutzung:

- Die Daten aus Sanity.io werden über eine API in die React-Frontend-Komponenten integriert. Dies ermöglicht die dynamische Darstellung von Inhalten auf den Seiten des Portfolios.

5. Bildverwaltung:

- Sanity.io bietet auch Funktionen zur Bildverwaltung. Dies ermöglichten das Hochladen, Skalieren und Optimieren von Bildern für das Portfolio. Die Bilder können dann einfach in den Texten und Projekten verwendet werden.

6. Zugriffskontrolle:

- Sanity.io ermöglicht die Verwaltung von Benutzerberechtigungen und Zugriffskontrolle. Dies ist wichtig, um sicherzustellen, dass nur autorisierte Benutzer Inhalte bearbeiten können.

7. Skalierbarkeit:

- Sanity.io ist skalierbar und kann mit dem Wachstum des Portfolios mithalten. Neue Typen und Felder können einfach hinzugefügt werden, um zusätzliche Inhalte und Funktionen zu unterstützen.

8. Echtzeit-Kollaboration:

- Sanity.io unterstützt die Echtzeit-Kollaboration, was es mehreren Benutzern ermöglicht, gleichzeitig an der Aktualisierung von Inhalten zu arbeiten. Dies ist besonders nützlich, wenn ein Team an der Portfolio-Verwaltung beteiligt ist.

9. Entwicklungszeit:

- Die Implementierung des Back-Ends mit Sanity.io reduziert die Entwicklungszeit erheblich. Es ist nicht erforderlich, ein benutzerdefiniertes CMS zu erstellen, da Sanity.io bereits viele Funktionen bietet.

10. Sicherheit:

- Sanity.io legt großen Wert auf Sicherheit und Datenschutz. Die Plattform bietet Schutz vor Datenverlust und unautorisiertem Zugriff, und ihre Richtlinien auch den DGSVO-Komfort entsprechen.

Die Implementierung des Back-Ends mit Sanity.io bietet zahlreiche Vorteile, einschließlich einer einfachen und effizienten Verwaltung von Inhalten, einer schnellen Integration in das Front-End und einer benutzerfreundlichen Oberfläche für Inhaltsbearbeitung. Dies ermöglicht eine nahtlose Aktualisierung des Portfolios und trägt dazu bei, die gesteckten Ziele zu erreichen.

Die Arbeit an der Implementierung des Back-Ends erfordert die Zusammenarbeit von Entwicklern und Inhaltautoren, um sicherzustellen, dass die Datenstruktur und die Inhalte den Anforderungen des Portfolios entsprechen. Die Auswahl von Sanity.io als Headless-CMS bietet eine robuste Lösung, um die Inhaltsverwaltung zu optimieren und den Erfolg des Projekts sicherzustellen.

8. Testung

8.1 Beschreibung der Tests

Die Tests umfassen funktionale und visuelle Überprüfungen des Portfolios. Funktionale Tests gewährleisten die korrekte Funktionalität von Navigation und Interaktionen. Visuelle Tests stellen sicher, dass das Design konsistent und ansprechend ist. Fehlerbehebung und

Browserkompatibilität werden überprüft, um eine reibungslose Benutzererfahrung sicherzustellen.

8.1.1 Automatisierte Tests

Automatisierte Tests sind integraler Bestandteil der Qualitätskontrolle. Sie umfassen Unit-Tests, Integrationstests und End-to-End-Tests, die automatisch den Code auf Funktionalität und Fehler überprüfen. Dies ermöglicht die frühzeitige Identifizierung von Problemen und gewährleistet die Stabilität des Portfolios während des Entwicklungsprozesses.

8.1.2 Manuelle Tests

Manuelle Tests sind entscheidend, um die Qualität des Portfolios sicherzustellen. Sie umfassen eine gründliche Überprüfung der Benutzeroberfläche und Interaktionen durch Testpersonen. Wichtige Aspekte sind die Navigation, die Benutzerfreundlichkeit und die visuelle Ästhetik. Manuelle Tests überprüfen auch die Korrektheit und Vollständigkeit der Inhalte, einschließlich Rechtschreibung und Grammatik. Diese Tests sind wichtig, um sicherzustellen, dass das Portfolio den Erwartungen entspricht und eine positive Benutzererfahrung bietet. Feedback von Testpersonen kann dazu beitragen, potenzielle Verbesserungen zu identifizieren und zu implementieren. Die Kombination aus automatisierten und manuellen Tests gewährleistet eine umfassende Qualitätskontrolle und einen erfolgreichen Projektabschluss.

Aus Reihe der Manuelle Test wurden die folgende tiefgründig auf der Portfolio Seite geprüft und getestet:

Usability-Tests:

- Überprüfung, wie leicht sich Benutzer auf der Website zurechtfinden, insbesondere bei der Navigation durch verschiedene Portfolio-Abschnitte und Projekte.

UX-Tests (User Experience):

- Bewertung der Gesamtbenutzererfahrung, einschließlich der Interaktion mit den Portfolioinhalten und der allgemeinen Gestaltung der Website.

Navigationstests:

- Verifizierung, ob die Navigation durch die Portfolio-Seiten intuitiv ist und ob Benutzer einfach zwischen verschiedenen Abschnitten wie Erfahrungen, Projekten und Fähigkeiten wechseln können.

Visuelle Ästhetik-Tests:

- Überprüfung der visuellen Gestaltung des Portfolios, einschließlich der Ästhetik von Projekten, Animationen und der Konsistenz im gesamten Design.

Inhaltsprüfung:

- Kontrolle der Korrektheit und Vollständigkeit von Texten in den Projektbeschreibungen, Biografie und anderen relevanten Bereichen. Gewährleistung der sprachlichen Korrektheit.

Kompatibilitätstests:

- Überprüfung, ob das Portfolio auf verschiedenen Browsern (wie Chrome, Firefox, Safari) und Geräten (Desktop, Tablet, Mobilgeräte) konsistent und fehlerfrei dargestellt wird.

Responsivitätstests:

- Sicherstellung, dass das Portfolio auf verschiedenen Bildschirmgrößen reaktionsschnell ist und sich die Benutzeroberfläche an verschiedene Geräte anpasst.

Formulartests:

- Wenn Formulare für Kontaktinformationen oder ähnliches vorhanden sind, Überprüfung, ob diese korrekt funktionieren und eingegebene Daten ordnungsgemäß verarbeitet, werden

8.2 Versionierung:

Die Versionierung des Portfolios erfolgt mithilfe von Git und GitHub. Dies ermöglicht die Verwaltung und Nachverfolgung von Änderungen im Code und der Inhalte. Jede Aktualisierung wird als neuer Branch oder Commit festgehalten, was eine transparente Entwicklung und Kollaboration zwischen Teammitgliedern erleichtert.

9. Sanity Schemas

Es ist wichtig, die Struktur und die Verlinkungen der Sanity-Schemas in deinem Portfolio-Projekt klar darzustellen. Dies gewährleistet nicht nur die Transparenz für das IHK, sondern auch eine reibungslose Datenverwaltung und ein effizientes Frontend-Rendering. Im Folgenden werde ich die verschiedenen Sanity-Schemas - *Experience*, *PageInfo*, *Project*, *Skill* und *Social* - und ihre Verknüpfungen auf der Portfolio-Seite detailliert erläutern:

9.1 Experience Schema

- Das *Experience*-Schema ist verantwortlich für die Erfahrungseinträge in dem Portfolio. Es enthält Informationen wie den Titel, das Unternehmen, das Datum und die Beschreibung der Erfahrung. Jeder Erfahrungseintrag ist mit der Verwendeten Fähigkeiten Liste verknüpft, die während dieser Erfahrung erstellt. Dies erfolgt durch eine Beziehungsfeld-Verknüpfung zu den *Skill* Schema.

9.2 Project Schema

- Das *Project*-Schema enthält die detaillierten Informationen zu den einzelnen Projekten in deinem Portfolio. Dazu gehören der Projekttitel, eine kurze Beschreibung, das Hauptbild, die Verwendeten Fähigkeiten und der Link zum vollständigen Projekt. Jedes Projekt wird mit Verwendeten Fähigkeiten Liste (*Skill*-Schema) verknüpft, um anzuzeigen, unter welchen verwendenden Technologien bzw. Fähigkeiten es erstellt wurde.

9.3 Skill Schema

- Das *Skill*-Schema enthält eine Liste der Fähigkeiten und Technologien, die du in deinem Portfolio hervorheben möchtest. Es ist auch mit den Projekten verknüpft, um

anzuzeigen, welche Fähigkeiten in welchen Projekten angewendet wurden so auch wie mit den beruflichen Erfahrungen. Diese Verknüpfung erfolgt über eine Beziehungsfeld-Verknüpfung zu den Projekten und beruflichen Erfahrung.

9.4 Social Schema

- Das *Social*-Schema enthält Links zu deinen Social-Media-Profilen, um Benutzern die Möglichkeit zu geben, dich auf verschiedenen Plattformen zu kontaktieren. Dieses Schema kann auf der Seite "Kontakt" oder im Kopfbereich der Portfolio-Website platziert werden. Es ist nicht direkt mit anderen Schemas verknüpft, sondern dient zur Bereitstellung von Kontaktoptionen.

9.5 PageInfo Schema

- Das *PageInfo*-Schema enthält allgemeine Informationen über die Seite, wie den Titel, die Beschreibung und das Hauptbild. Dieses Schema wird auf der Startseite deines Portfolios verwendet, um wichtige Informationen und ein ansprechendes Bild anzuzeigen.

9.6 Verknüpfungen zwischen Schemas:

Die Schemas sind miteinander verknüpft, um die Daten deines Portfolios effizient zu organisieren und darzustellen. Hier sind einige wichtige Verknüpfungen:

- Jedes *Project*-Schema ist mit einem oder mehreren *Skill*-Schemas verknüpft, um anzuzeigen, welche Fähigkeiten in diesem Projekt verwendet wurden.
- Jeder Erfahrungseintrag im *Experience*-Schema ist mit einem oder mehreren *Skill*-Schemas verknüpft, um anzuzeigen, welche Fähigkeiten in dieser beruflichen Erfahrung verwendet wurden.
- Auf der Seite "Über mich" (*PageInfo*-Schema) sind Informationen über dich angezeigt werden, um Benutzern einen Überblick über deine Persönlichkeit und deine beruflichen Interessen zu geben.
- Die Social-Media-Links im *Social*-Schema können auf der Kontaktseite oder im Kopfbereich der Website platziert werden, um Benutzern die Möglichkeit zu geben, dich über verschiedene Plattformen zu erreichen.

Die Verknüpfungen zwischen diesen Schemas gewährleisten eine konsistente Darstellung und ermöglichen es, die Informationen auf deiner Portfolio-Website dynamisch darzustellen. Dies ist besonders wichtig, um die Inhalte einfach zu aktualisieren und sicherzustellen, dass sie immer aktuell und relevant sind.

10. UML-Klassendiagramme

10.1 Textuelle Beschriftung der Diagramme

Das UML-Klassendiagramm ist eine Darstellung der Schemas und ihrer Beziehungen in deinem Portfolio-Projekt. Bitte beachte, dass hier in diesem Abschnitt keine speziellen UML-Grafiken erstellen kann, aber lediglich beschriftet wird, wie die Schemas und ihre Beziehungen in einem Klassendiagramm darstellt sind textuell:

- *Experience* Schema
  - [+JobTitle: String]
  - [+Company: String]
  - [+DateStarted: Date]
  - [+DateEnded: Date]
  - [+IsCurrentlyWorkingHere: Boolean]
  - [+Technologies: [Skill]]
  - [+Points: [String]]

- *Project* Schema
  - [+Title: String]
  - [+Image: Image]
  - [+Summary: String]
  - [+Technologies: [Skill]]
  - [+LinkToBuild: String]

- *Skill* Schema
  - [+Title: String]
  - [+Category: String]
  - [+Progress: Number]
  - [+Image: Image]

- *Social* Schema
  - [+Title: String]
  - [+Url: String]

- *PageInfo* Schema
  - [+Name: String]
  - [+Role: String]
  - [+HeroImage: Image]
  - [+BackgroundInformation: String]
  - [+ProfilePic: Image]
  - [+PhoneNumber: String]
  - [+Email: String]
  - [+Address: String]

Beziehungen:

- *Project* "verwendet" *Skills* (mehrere Skills können in einem Projekt verwendet werden)
- *Experience* "verwendet" *Skills* (mehrere Skills können in einem Projekt verwendet werden)
- *Skills* "werden verwendet in" *Project* (mehrere Projekte können dieselben Fähigkeiten verwenden)

- *Skills* "werden verwendet in" *Experience* (mehrere Experience können dieselben Fähigkeiten verwenden)
- *Social* Media-Links sind Teil der Seite (Keine direkte Beziehung zu anderen Schemas)
- *PageInfo* enthält allgemeine Informationen zur Seite (Keine direkte Beziehung zu anderen Schemas)

10.2 Visuelle Darstellung der Diagramme

Diese Beschreibungen könnten in ein UML-Klassendiagramm umgewandelt werden, um die Struktur und die Beziehungen zwischen den Schemas visuell darzustellen.

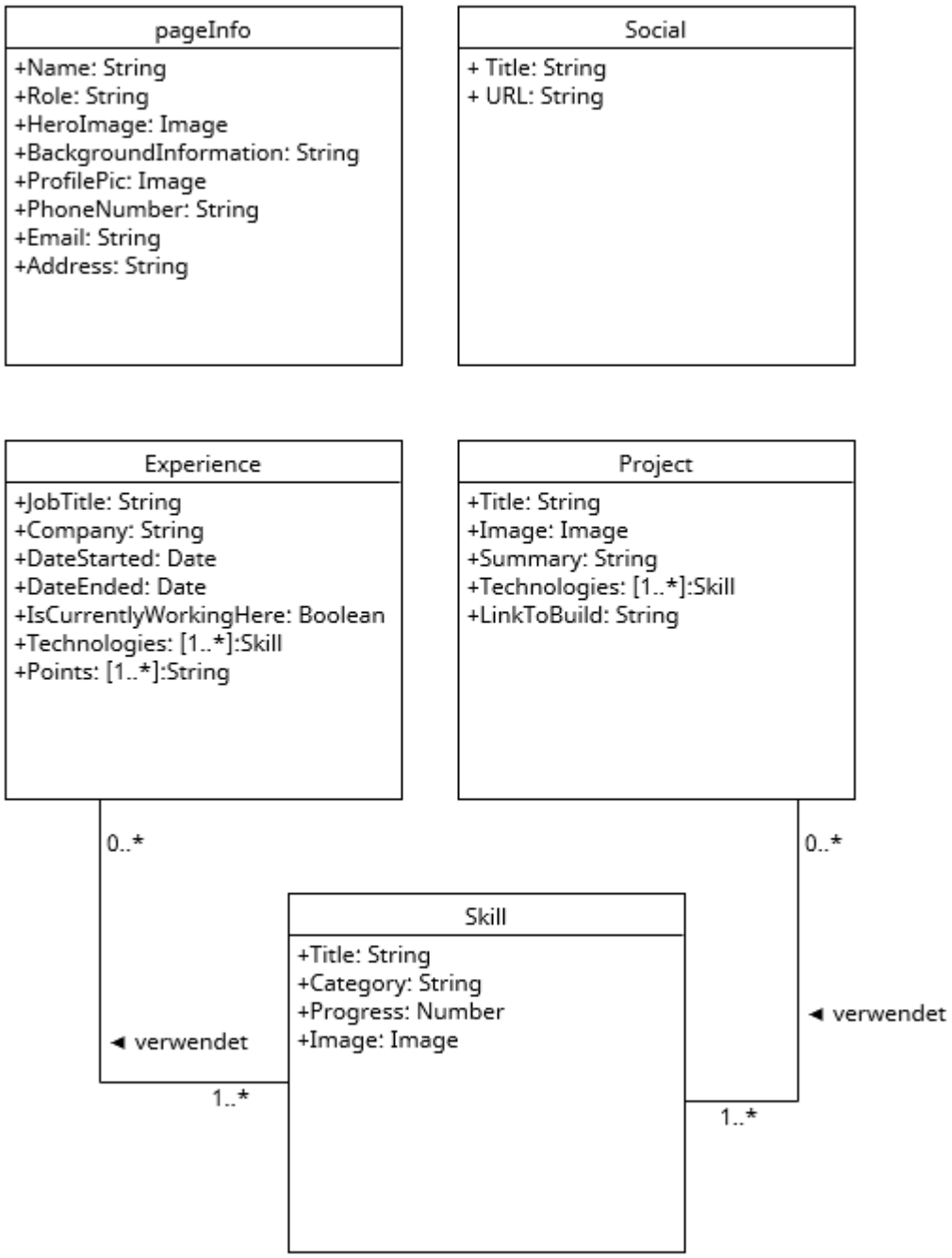

Abbildung 4: UML Klassendiagram der Schemas

11. Fazit

11.1 Soll-/Ist-Vergleich

Der Soll-/Ist-Vergleich des Portfolios auf JAMstack-Basis zeigt, dass die gesteckten Ziele weitgehend erreicht wurden. Das Soll, ein modernes Portfolio, das in Millisekunden lädt und benutzerfreundlich ist, wurde erfüllt. Die Implementierung von NEXT.JS, Framer Motion, und Tailwind CSS ermöglicht eine schnelle und ansprechende Benutzererfahrung. Die Nutzung von Sanity.io als Headless-CMS erleichtert die Inhaltsverwaltung erheblich. Die Umsetzung von automatisierten und manuellen Tests gewährleistet die Qualität des Portfolios. Die Wirtschaftlichkeitsbetrachtung zeigt, dass das Projekt kosteneffizient ist. Insgesamt entspricht das Ist dem angestrebten Soll und bietet ein erfolgreiches Ergebnis im Rahmen des JAMstack-Portfolio-Projekts. Diese ist zu abrufen unter: https://portfolio-nextjs-lyffski.vercel.app

11.2 Schlussbetrachtung

Im Rückblick auf die Entwicklung meines JAMstack-Portfolios möchte ich die zentralen Herausforderungen und Erfolge resümieren. Die Analyse der Problemstellung verdeutlichte die Bedeutung einer effektiven Umsetzung von JAMstack-Technologien wie NEXT.JS, Framer Motion, Tailwind CSS und Sanity.io.

Die präsentierten Sachverhalte unterstreichen nicht nur die technische Raffinesse hinter dem Portfolio, sondern auch den Fokus auf eine herausragende Benutzererfahrung. Das Portfolio fungiert nicht nur als digitale Visitenkarte, sondern als lebendiges Beispiel für die nahtlose Integration von Content-Management und modernen Frontend-Technologien.

Abschließend ist es entscheidend, die erzielten Ergebnisse zu betonen, ohne in Wiederholungen zu verfallen. Die prägnante Zusammenfassung hebt die innovativen Aspekte des Portfolios hervor und reflektiert kritisch über mögliche Weiterentwicklungen.

Der klare Handlungsvorschlag an die Stakeholder besteht darin, das Portfolio als leistungsfähiges Werkzeug zur Selbstdarstellung und Projektpräsentation zu nutzen. Der Ausblick auf die Zukunft betont die Möglichkeit, das Portfolio kontinuierlich zu erweitern und an aktuelle technologische Entwicklungen anzupassen.

Die Schlussbetrachtung bildet somit den Abschluss dieser Projektarbeit und legt den Fokus auf die Relevanz und Potenziale des entwickelten JAMstack-Portfolios.

Eidesstattliche Erklärung

Ich, Kacper Sobczak, versichere hiermit, dass ich meine Dokumentation zur schulischen Projektarbeit mit dem Thema

> **Jamstack Portfolio -** Realisierung einer Jamstack Portfolio als Reacteinzelseiten Webapplikation

Selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde oder schulischen Einrichtung vorgelegt und auch nicht veröffentlicht.

Neuburg an der Donau, den 08.02.2024

Anhang

Abbildung 5: Home/Landing der JAMStack Portfolio Seite

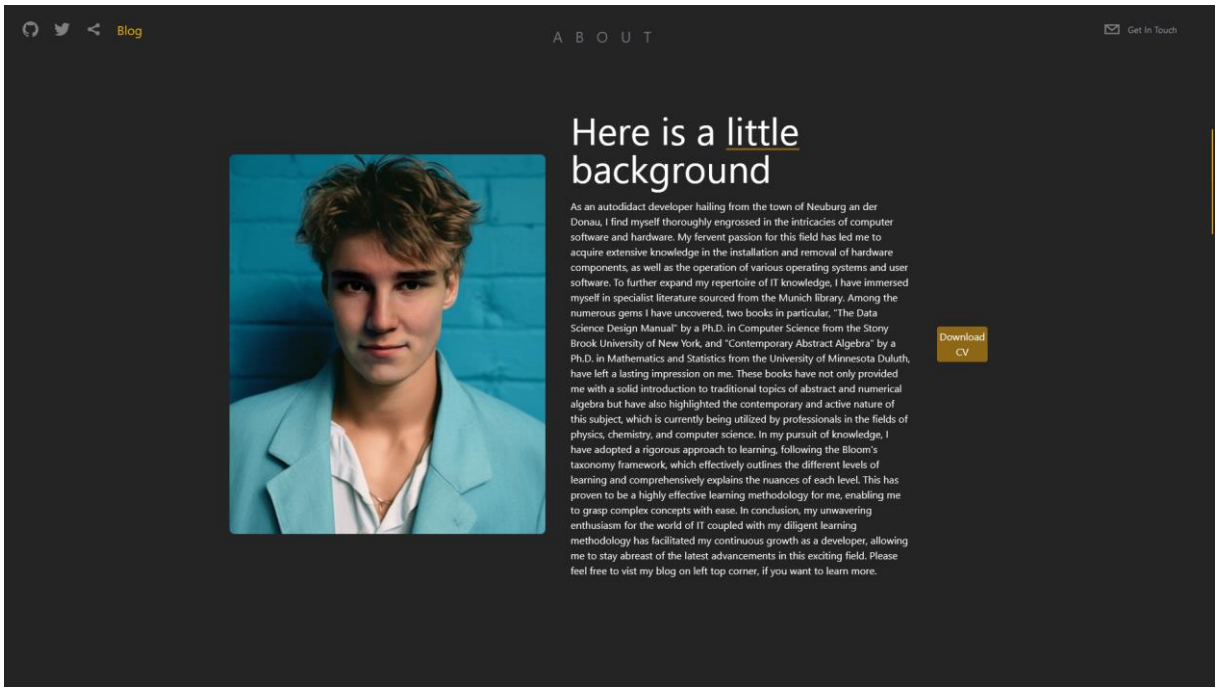

Abbildung 6: About Page

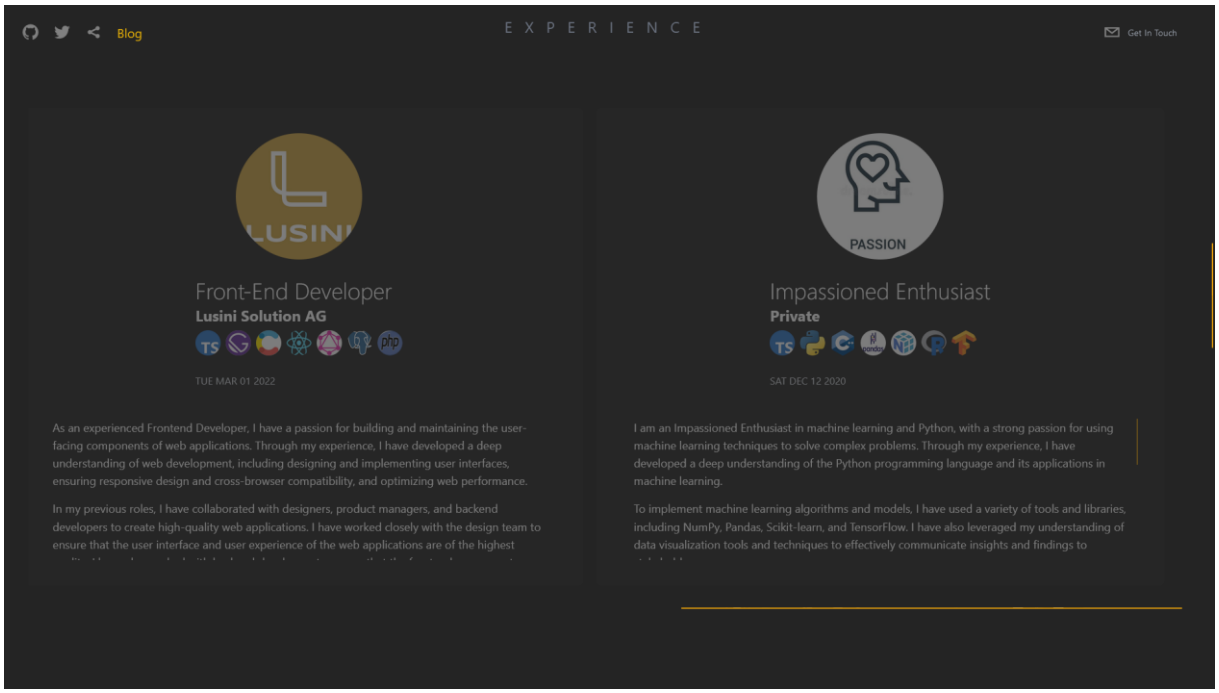

Abbildung 7: Experience Page

Abbildung 8: Skills Page

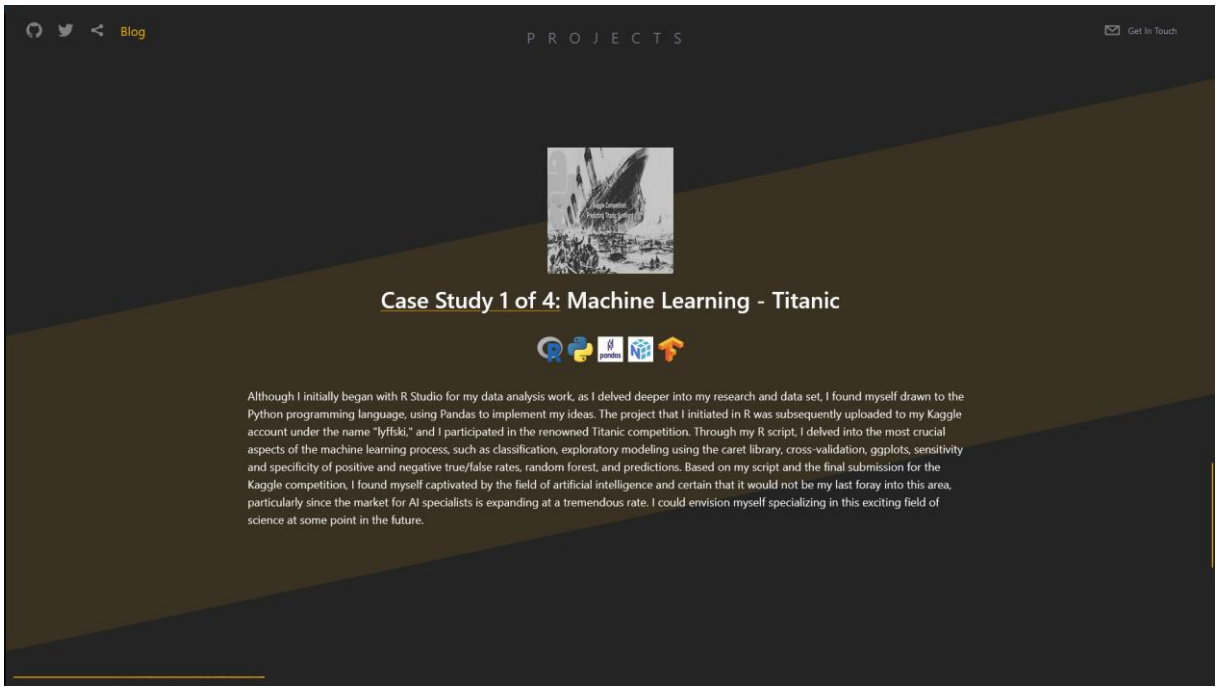

Abbildung 9: Project Page

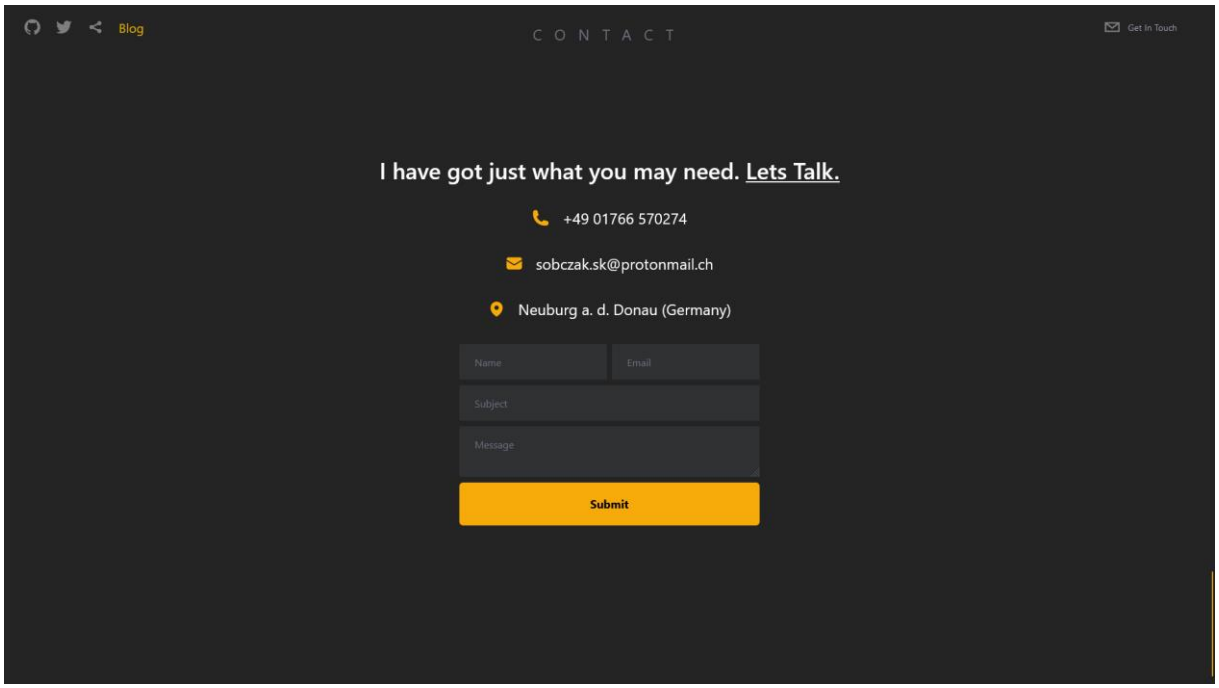

Abbildung 10: Contact Page