

Projektarbeit

Internationale Hochschule Duales Studium

Studiengang: Informatik

**Generative Programmierung mittels künstlicher Intelligenz**

Müller Korbinian

Matrikelnummer: 102302316

Adresse

Lohwald Straße 59

86356 Neusäß

Abgabedatum: 30.09.2024

Inhalt

[Einleitung 3](#_Toc177054083)

[Theoretische Fundierung 3](#_Toc177054084)

[Begriffserklärung 3](#_Toc177054085)

[Künstliche Intelligenz 3](#_Toc177054086)

[Generative KI 4](#_Toc177054087)

[Grundlagen der Softwareentwicklung 4](#_Toc177054088)

[React-Native 4](#_Toc177054089)

[Material-UI 4](#_Toc177054090)

[Möglichkeiten und Grenzen der generativen KI in der Programmierung 4](#_Toc177054091)

[Methodik 5](#_Toc177054092)

[Fallstudie 5](#_Toc177054093)

[Voraussetzung 5](#_Toc177054094)

[Softwareanforderungen 6](#_Toc177054095)

[Implementation 7](#_Toc177054096)

[Durch Entwickler umgesetzt 7](#_Toc177054097)

[Generative KI 8](#_Toc177054098)

[Analyse 11](#_Toc177054099)

[Entwicklungszeit 11](#_Toc177054100)

[Code Style 11](#_Toc177054101)

[Lesbarkeit des Codes 11](#_Toc177054102)

[Resilienz gegenüber potenziellen Fehlern 12](#_Toc177054103)

[Implikationen 12](#_Toc177054104)

[Fazit 12](#_Toc177054105)

[Anhang 13](#_Toc177054106)

[A) Literaturverzeichnis 13](#_Toc177054107)

Einleitung

Wir befinden uns im Zeitalter der Künstlichen Intelligenz. Es gibt kaum eine Branche, die noch nicht mit dieser Technologie in Berührung gekommen ist. Auch im Bereich der Softwareentwicklung kommt die Technologie an. Sei es durch GitHub-Copilot, Gemini oder ChatGPT. Hier liegt der Fokus auf der automatischen Code-Generierung durch die generativen KI-Modelle. Allem voran basierend auf dem Large Language Model. Dadurch besteht die Möglichkeit, die Entwicklungsgeschwindigkeit zu erhöhen, indem einfache und repetitive Aufgaben automatisiert werden.

Die vorliegende Arbeit untersucht die Chancen und Grenzen bei der Erzeugung von Programmcode mittels Künstlicher Intelligenz. Dabei werden die Chancen und Grenzen bei der Entwicklung einer UI-Komponente in React-Native untersucht. Das Ziel ist es, die Qualität und Effizienz der KI-basierten Codegenerierung im Vergleich zu einer herkömmlichen Implementierung durch einen Entwickler zu bewerten. Dazu wird eine Fallstudie durchgeführt, bei der dieselbe UI-Komponente durch einen Entwickler und durch eine generative Künstliche Intelligenz implementiert wird. Im Anschluss wird durch ein Senior-Entwickler bewertet, welche d Implementationen in Bezug auf Leserlichkeit, Code-Qualität und Korrektheit besser abschneidet.

Durch diese Arbeit soll ein tieferes Verständnis gewonnen werden, welcher Nutzen und welche Grenzen es bei dem Einsatz von generativer Künstlicher Intelligenz bei der Softwareentwicklung gibt. Des Weiteren soll beleuchtet werden, in welchen Bereichen es bereits jetzt einen Mehrwert durch die KI gibt und bei welchen es noch Herausforderungen gibt, die es zu überwinden gilt.

Theoretische Fundierung

Die Grundlage dieser Arbeit bildet die theoretische Fundierung. Im Folgenden werden wichtige Begriffe definiert. Es werden auch zentrale Ideen und Ansätze der generativen künstlichen Intelligenzen erläutert. Des Weiteren wird noch grundlegende Konzepte der Software-Entwicklung und des in der Fallstudie verwendeten Frameworks diskutiert. Zum Schluss wird die Relevanz zur Fallstudie aufgezeigt.

Begriffserklärung

Künstliche Intelligenz

Insgesamt gibt es vier Perspektiven; um den Begriff Künstliche Intelligenz zu definieren. Die erste besagt, dass Künstliche Intelligenz ein Teilgebiet der Informatik ist. Es werden Hard- und Software-Systeme entwickelt, um Probleme zu lösen, für die Intelligenz erforderlich ist. Die Zweite besagt, dass KI in Hard- und Software eingesetzt wird um, eine Effizienzsteigerung zu erreichen. Eine andere Perspektive ist die, dass Künstliche Intelligenz ein System ist, welches intelligentes Problemlösungsverhalten zeigt. Zuletzt ein künstliches Wesen, das Intelligenz besitzt (Lämmel, U & Cleve, J ; 2020, S. 12)

Generative KI

Pyinaya et al. (2023, S. 2) definieren generative Künstliche Intelligenz als eine Ansammlung von Techniken und Modellen, die Zusammenhängen in den Trainingsdaten identifizieren. Basierend auf den Zusammenhängen sind diese nun in der Lage, neue Daten zu generieren. Sowohl solche, die zu dem ursprünglichen Datensatz gehören, als auch neue in anderen Bereichen.

Dass generative KI einen großen Teil der Entwicklungsarbeit leisten kann, wiesen Cassieri et al.(2024, S. 902-904) in ihrer Studie nach. Das Ergebnis zeigt, dass die generative Künstliche Intelligenz GPT-4 nach maximal einem Iterationsschritt in den meisten Fällen Code produziert, der für den Einsatz geeignet ist.

Grundlagen der Softwareentwicklung

React-Native

React-Native ist ein Framework, basierend auf React. Es wurde durch Meta entwickelt und ist Open Source. Das Ziel hinter React-Native ist es, dem Nutzer eine native Anwendung bereit zu stellen und dem Entwickler den maximalen Komfort bei der Implementierung bereitzustellen. Durch die erhöhte Abstraktion ist es möglich, Reakt-Komponenten zu Nativen-Elementen zu transformieren (Occhino T. 2015)

Material-UI

Material-UI ist eine populäre UI-Bibliothek, die auf den Richtlinien des Material-Designs von Google basiert. Sie bietet eine Vielzahl von vorgefertigten, responsiven UI-Komponenten und ist besonders in der React-Entwicklung weit verbreitet (2021).

Möglichkeiten und Grenzen der generativen KI in der Programmierung

In ihrer Studie stellten Peng et al. (2023) fest, dass Künstliche Intelligenz sehr großen Einfluss auf die Entwicklungsgeschwindigkeit von Software hat. In der Studie wird von einer Zeitreduktion von 55,8 % gesprochen.  
Daraus resultieren sehr weitreichende Implikationen für die Wirtschaft.

Auf der anderen Seite gibt es auch klare Grenzen. KI kann Schwierigkeiten haben, den Kontext komplexer Anforderungen zu verstehen oder kreative, maßgeschneiderte Lösungen zu entwickeln. Fehlende domainspezifische Kenntnisse führen oft zu Ergebnissen, die nicht den Erwartungen entsprechen und weitere manuelle Anpassungen erfordern (Bender et al., 2021).

Methodik

Dieser Abschnitt beschäftigt sich eingehend mit dem praktischen Ansatz dieser Arbeit. Zunächst wird die Fallstudie beschrieben. Hier wird insbesondere der Kontext zu dem Kundenprojekt gesetzt. Die formalen Voraussetzungen für die Software werden hier ebenfalls dargestellt.

Fallstudie

Um die Effektivität und Qualität der durch die KI produzierten Software zu überprüfen, wird eine Fallstudie durchgeführt. Diese besteht darin, ein React-Native-UI-Komponente zu realisieren. Diese Komponente wird einmal durch einen Entwickler und einmal durch eine generative Künstliche Intelligenz umgesetzt. Um die Effizienz der Implementation rudimentär zu prüfen, wird die Zeit festgehalten, die benötigt wurde, um diese Komponente zu Implementieren. Die Qualität wird durch einen Senior-Entwickler bewertet. Hier sind die Hauptpunkte: Optisch ansprechend, Code Style, ist der Code verständlich und eine erste Einschätzung welche Implementation weniger störanfällig ist.

Voraussetzung

Die Grundlage für die Fallstudie ist ein Kundenprojekt bei dem Praxispartner. Der Kunde ist ein großer Obstbauernverband in Deutschland. Dieser beauftragte eine digitale Mitgliederverwaltung. Die Kunden, Bauern, werden hiermit erfasst. Neben der Anschrift, Firmenname und Kontaktdaten werden auch die Flächen registriert. Zu den Flächen wird dokumentiert, wo sich diese befinden und welche Obstsorten angebaut werden. Anhand dieser Information kann der Bauer Dienstleistungen buchen. Der Beitritt zu dem Verband ist für die Bauern kostenpflichtig. Dazu kommt noch eine monatliche Gebühr. Eine der Dienstleistungen ist die Beratung. Dabei wird ein Termin ausgemacht und ein Beraten kommt vorbei. Diese Beratungen können individuell, basierend auf den Früchten oder Flächen geschehen. Um den Beratern die Arbeit zu erleichtern, wurde beschlossen, eine Handy-App für diese zu erstellen. Diese Handy-App ist die Grundlage für die Fallstudie.

Die Handy-App basiert auf dem Java-Script-Framework React Native. Die Entscheidung ist auf dieses gefallen, da es ein sehr modernes Framework ist. Ein weiterer Grund ist die Entwicklungsgeschwindigkeit. Das Projekt wird zu nativen Apps für Android und IOS kompiliert und ist als solche für die Berater zum Download bereitgestellt.   
In der App werden die relevanten Daten ausschließlich zwischengespeichert, damit wird eine Offline-Funktion realisiert. Die Hauptdaten-Verwaltung passiert zentral in der Mitgliederverwaltung. Dazu wurde die Mitgliederverwaltung um einen Rest-Service erweitert. Die Kommunikation zwischen der App und dem Backend geschieht über verschlüsselte HTTPS-Anfragen mit dem Paket Axios.

Die App ist einfach gehalten. Der Berater loggt sich ein und kommt zum Dashboard. Hier findet die Hauptnavigation statt. Von hier aus gelangt er zu seinem Profil, einer Übersicht über alle verfügbaren Beratungstätigkeiten und der Maske zum Erfassen und Dokumentieren seiner Tätigkeit. Hier wird erfasst, bei welchem Bauer, zu welcher Uhrzeit er war. Für die Tätigkeit selber wird dokumentiert, was er gemacht hat und wie lange es gedauert hat.

Einer der wichtigsten Bestandteile der App ist die Stundenübersicht. Der Name der Seite ist ConsultingSessionList. Hier sieht der Berater, welche Tätigkeiten er geleistet hat. Diese sind nach dem Datum absteigend sortiert. Es gibt hier die Möglichkeit, die einzelnen Elemente zu sortieren nach Beratungstyp, Fruchttyp oder nach Bauer. Diese nach Bauer sortierte Ansicht ist die Grundlage für die Fallstudie. Hier soll am oberen Bildschirmrand eine Kachel hinzugefügt werden. Die Kachel wird benutzt, um die Daten des Bauern darzustellen. Dazu zählen: Name, Anschrift, alle Flächen und alle Früchte, die angebaut werden. Ebenso sollen zwei Butten hinzugefügt werden, die die Telefonnummer und die E-Mail-Adresse anzeigen. Durch einen Klick auf die Buttons öffnet sich das entsprechende Programm, die Daten werden automatisch übernommen. Für die Umsetzung sollen Elemente der Material UI genommen werden.

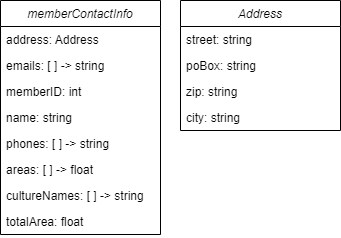
Der Name dieser Komponente ist ConsultingSessionMemberContactInfo. Diese Komponente wird in der ConsultingSessionList genutzt. Die ConsultingSessionList kümmert sich um das konditionelle Anzeigen der Komponente. Die entsprechenden Daten werden in der Komponente ConsultingSessionList von Server abgerufen und für den Einsatz aufbereitet. Die Daten werden in der Form eines Prop mit dem Namen memberContactInfo übergeben.  


Abbildung 1: Klassendiagramme für memberContactInfo Ursprung: Eigene Darstellung

Softwareanforderungen

Anhand dieser Projekt-Konstellation gehen folgende Anforderungen an die App hervor.

* Die App soll Native auf Mobilgeräten laufen, hier kommen die Betriebssysteme Android und IOS vor
* Als Programmiersprache wird Javascript in Verbindung mit dem Framework React Native eingesetzt
* Die Kommunikation findet über das Internet mittels HTTPS Request statt.
* Die Beratungsstunden-App interagiert mit dem Restservice der MitgliederverwaltungNeben den Allgemeinen gibt es noch weitere Anforderungen in Bezug auf die verwendeten Softwarebibliotheken. In diesem Projekt werden folgende Node Module eingesetzt:

|  |  |
| --- | --- |
| Node | V20.16.0 |
| Dependencies | react:^18.2.0, react-native: ^0.72.4, expo: ~49.0.8 |
| Material UI | mui/material:^5.12.7, mui/icons-material: ^5.14 |

Einen genaueren Überblick über alle in diesem Projekt verwendeten Dependencies finden sie im Anhang.

Implementation

Der Inhalt und der Aufbau des Projekts sind nun bekannt und die Anforderungen sind geklärt. Mit diesen Parametern beginnt nun die Implementierung der UI-Komponente ConsultingSessionMemberContactInfo. Die Komponente wird zweimal implementiert. Das erste Mal geschieht es rein durch den Entwickler. Das zweite Mal wird ein Prompt an die KI gestellt und iteriert, bis ein funktionierendes Ergebnis erreicht wird.   
Bei beiden Entwicklungsansätzen ist das Ziel gleich: eine funktionierende Komponente. Die Komponente gilt als funktionierend, wenn diese keine kritischen Fehler bei der Benutzung auftreten und diese optisch genau so aussieht, wie es im Screendesign vorgesehen ist.   
Der Ausgangspunkt ist gleich bei beiden Implementationen. Die Datei mit dem Namen CosultingSessionMemberContactInfo.jsx ist angelegt. Diese befindet sich im Projekt im gleichen Ordner wie die Datei ConsultingSessionList.jsx. In der ConsultingSessionList werden die Daten abgerufen und bereitgestellt. Die Komponente ConsultingSessionMemberContactInfo wird inkludiert und in der HTML-Struktur hinzugefügt. Die Daten werden als Prop übergeben. Beide Implementationen werden auf separaten Git-Branches umgesetzt. Die Branches basieren auf der aktuellsten Version des Branch Main.

Durch Entwickler umgesetzt

Die Umsetzung durch den Entwickler beginnt in der IDE PHP-Storm. Die Datei ist hier im Editor geöffnet. Zunächst wird das übergebene Prop dekonstruiert. Dadurch sind alle Informationen separat in Variablen zugänglich. Anschließend folgt die HTML-Strukturierung. Das UI-Element wird in vier Bereiche unterteilt. In der ersten sind Name und Mitglieds-ID. Der zweite Bereich enthält die Adressdaten. Alle Daten bezüglich der Flächen stehen im dritten Abschnitt. Im vierten sind die Buttons mit den Kontaktmöglichkeiten. Anhand dieser Aufteilung wird zunächst das MUI-Element Box verwendet; dieses dient als Container. Für jeden Abschnitt wird ein View-Element eingefügt. Damit ist die erste Strukturierung abgeschlossen.   
Der erste Bereich, der programmiert wird, ist der erste Container. Um das Styling zu vereinfachen, wird das Output-Text-Element in einem weiteren View-Element verschachtelt. Für die Ausgabe des Namens und der ID wird ein Java-Template String verwendet. Dieser ist so aufgebaut: ` ${memberContactInfo.name} [${memberContactInfo.memberID}`. Danach geht es an den zweiten Bereich, die Kontakt-Daten. Hier können die Daten null sein; dies passiert, wenn diese nicht in der Mitgliederverwaltung eingepflegt sind. Um einen kritischen Fehler zu vermeiden, der zu einem Absturz führt, muss überprüft werden, ob alle Daten vorhanden sind. Dies geschieht mit der Abfrage !***Object***.values(address).every(value => value === '' || value === null). Falls diese den Boolesche Wert wahr annimmt, wird der Inhalt ausgegeben. Zunächst einmal wird das Location-Icon von MUI, <PlaceIcon />, ausgegeben. Im Anschluss die Adressdaten. Hier ist es zu beachten, dass die Straße nicht hinterlegt ist, sondern eine PO-Box. Die Flächen werden in Form eines Arrays bereitgestellt. Um diese anzuzeigen, wird durch das Array iteriert und die einzelnen Elemente in Form von „Array Index Fläche“ dargestellt. Darunter befindet sich die Anzeige der Gesamtfläche, gerundet auf zwei Nachkommastellen. Zum Schluss sind hier die interaktiven Buttons. Da das Öffnen der Telefon-Anwendung auf IOS anders ist, als bei Android, wird hier die eine eigene Funktion ausgelöst, sobald ein Press-Event stattfindet. Für den E-Mail-Button wird die React-Native-Funktion Linking verwendet. Im letzten Schritt wird das Styling noch angepasst, sodass es gleich des Screendesigns ist.

Insgesamt hat die Implementation eine Zeit von zwei Stunden und 15 Minuten. Einen großen Teil der Zeit wurde damit verbracht, die richtigen MUI-Elemente auszuwählen. Auch das Überschreiben von Standard MUI-Styling hat einiges an Zeit in Arbeit genommen.

Generative KI

Für die Umsetzung wurde die KI ChatGPT von OpenAI verwendet. Hier wurde der Online-Prompt von OpenAI verwendet. Die Version ist GPT-4. Der generierte Code wird kopiert und so in die Datei eingefügt.   
**Prompt**Damit die KI ein möglichst gutes Resultat bekommt, wird der Inhalt und die Schnittstellen im Detail beschrieben. Dazu werden noch ein Screenshot aus dem Screendesign und ein Bild der Klassendiagramme hochgeladen.



Abbildung 2 Screenshot aus dem Screendesig; Ursprung: Screendesign OVR-PWA

Insgesamt Resultiert daraus folgender Prompt an die KI:

Das Projekt setting ist React-Native App. Die App wird auf Android und IOS Geräten verwendet. Hier verwenden wir Node in der Version 20.16.0. Für das Frontend nutzen wir Material UI. Das Ziel ist es eine UI-Komponente mittels MUI Elementen zu erzeugen, die genau so aussieht wie das erste Bild. Die Komponente heißt ConsultingSessionMemberContactInfo. Das Zweite Bild zeigt die Struktur der Daten. Diese wird mit dem Prop memberContactInfo bereitgestellt. Beachte alle Elemente könnten Null sein. Die Gesamtfläche soll auf eine Ganze Zahl gerundet werden. Die Telefonnummer und email sind in einem Array, wenn diese nicht Null sind nimm die Ersten Elemente daraus. Die Buttons sollen funktionsfähig sein. Wenn auf die Telefonnummer geklickt wird, soll sich die Telefon-Anwendung öffnen. Bei der E-Mail das E-Mail-Programm. Beachte den Schatten unter dem Element. Das nötige styling soll mit in die Datei als const styles = StyleSheet.create. Das Ergebnis soll auch alle Includes enthalten

**Iterationen**

Das Ergebnis des Promptes wird in die Datei eingefügt, das Projekt wird neu gebildet und im Browser aufgerufen. Beim Aufruf der Beratungs-Historie tritt ein kritischer Fehler auf. Der Fehler wird beschrieben und erneut in den Prompt geschrieben. Der Input:

Der gerade generierte Code führt zu dem Fehler: Uncaught TypeError: areas[0].to Fixed is not a function. Es sollen alle Elemente in dem array areas ausgegeben werden.

Mit dieser Addition funktioniert es und die Komponente wird angezeigt.

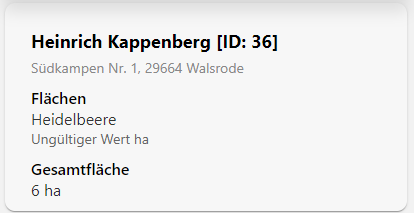
 Es fällt auf, dass es noch nicht ganz wie im Screendesign aussieht. Der Container hat abgerundete Ecken. In den eckigen Klammern steht ID. Die Anordnung der Adressdaten ist anders. Kulturnahme und die Fläche sollten in einer Linie sein. Jedoch ist das Wichtigste, dass die Buttons nicht angezeigt werden. Abbildung

Abbildung 3) Ergebnis nach der ersten Iteration; Ursprung: Eigene Darstellung

Mit diesen Informationen wird ein weiterer Iterationsschritt gemacht. Der Prompt dazu:  
Halte dich näher an dem Screendesign. Die Komponente hat keine abgerundeten Ecken. Es hat auch eine leichte Hintergrundfarbe von #e5e5e5. In der Überschrift steht keine ID vor der Nummer. Die Adresse ist anders angeordnet. Location Icon in Grau, daneben die Straße, unter der Straße ist die PLZ gefolgt von dem Ortsnamen. Die Flächen sollten in einer Zeile stehen. Hier ist es in Ordnung, die Abfrage bei Areas wegzulassen. Auch die Gesamtfläche sollte in einer Reihe stehen. Die Fläche auch fett geschrieben. Die Buttons sind nicht zu sehen. Muss der Link mit der Telefonnummer nicht unterschiedlich sein, für iOS? Passe bitte alles an und halte die exakt an das Screendesign.

Abbildung 4) Ergebnis der Iteration; Ursprung: Eigene Darstellung

Das Resultat ist nach dieser Iteration deutlich besser und näher an dem Ziel dran. Dennoch gibt es auch jetzt ein paar Stellen, die noch verbessert werden müssen. Insbesondere hat die generative KI ein neues Layout für die Komponenten gewählt. Die Abstände wurden verändert. Erneut wurden alle Elemente beschrieben, die nicht ganz den Anforderungen entsprechen. Ferner wurde darauf verwiesen, dass das Ergebnis exakt wie das Screendesign aussehen sollte. Für die dritte Iteration entsteht folgender Prompt:

Die ID sollte neben dem Namen stehen, in Form von [memberID]. Die Auflistung der einzelnen Flächen sollte unter der Überschrift Flächen geschehen. Die Buttons sind nicht zu sehen, der Container ist zu klein. Es fehlt der Dropshadow beim Element. Die Buttons sind nicht zu sehen. Sind die auch wie im Screendesign zu sehen übereinander? und haben diese in grauen Buttons? halte dich exakt an das Screendesign, das ich dir am Anfang gegeben habe. Lass die Abstände zwischen den einzelnen Abschnitten weg. Genauso die unterschiedlichen Schriftgrößen, der Name und die ID passen, der Rest soll einheitlich sein. Der Abstand vom Containerrand zum Inhalt sollte 1 rem sein.

Das Ergebnis nach diesem Prompt ist sehr ähnlich zu dem Screendesign. Damit ist die Komponente umgesetzt.



Abbildung 5) Finales Ergebnis der KI; Ursprung: Eigene Darstellung

Insgesamt wurde für diese Implementation eine Stunde und 8 Minuten benötigt.

Analyse

Nachdem nun beide Implementationen vollständig sind, geht es an die Analyse der Ergebnisse. Die Senior-Entwicklerin, deren Meinungen und Einschätzungen hier aufgegriffen werden, ist seit mehreren Jahren bei mpunkt tätig. Diese hat umfangreiches Wissen im Bereich der Fullstack-Webentwicklung und ist die Projektleitung bei mehreren Projekten. Hierunter sind auch Projekte, die auf dem React-Ökosystem basieren. Dadurch sind die Grundlegenden geeignet, diese Überprüfung durchzuführen. Sie wurde explizit dafür ausgewählt, da sie nicht in diesem Projekt involviert ist. Dadurch ist sie nicht voreingenommen und kann objektiv die Implementationen analysieren.

Entwicklungszeit

Die Implementation durch den Entwickler dauerte insgesamt zwei Stunden und 15 Minuten. Ein großer Teil der Entwicklungszeit lag in der Planung und der Recherche, welche Material-UI-Elemente für die Implementation genommen werden.

Im Gegensatz dazu lag die Entwicklungszeit für die Künstliche Intelligenz bei einer Stunde und acht Minuten. Es wurden einige Iterationsschritte rein allein für das Styling verwendet, da die Künstliche Intelligenz von dem vorgegebenen Screendesign abgewichen ist.

Im Punkt Entwicklungszeit liegt die Künstliche Intelligenz vor dem Menschen. In dieser Fallstudie ist der Unterschied signifikant. Da es sich um eine Reduktion bei fast 50 % liegt.

Code Style

Die Analyse des Codestyle sorgte für die Erkenntnis, dass beide Implementationen auf einem sehr hohen und ähnlichen Stand sind. Die Senior-Entwicklerin bevorzugte die generierte UI-Komponente. Ein Grund war der Aufbau innerhalb der Datei, bei der generierten Implementation wurde das Styling nach der Komponente deklariert. Sie findet, dass es dadurch besser getrennt ist. Ein weiterer Punkt ist die Verwendung von externen Funktionen für die Buttons. In der manuell erstellten Komponente wurde nur die Anruf-Funktionalität in eine eigene Funktion ausgelagert, da diese eine höhere Komplexität aufweist.

Lesbarkeit des Codes

Auch die Lesbarkeit ist auf einem ähnlich guten Niveau. Auch hier wurde die Variante von der generativen KI bevorzugt. Diese hat stärkere Einrückungen für die Codeblöcke generiert. Die Künstliche Intelligenz hat außerdem noch aussagekräftigere Kommentare verwendet. Unter dem Aspekt der Lesbarkeit entscheidet sich die Seniorentwicklerin für die generierte CosultingSessionMemberContactInfo.

Resilienz gegenüber potenziellen Fehlern

Die Senior-Entwicklerin hat sich zunächst einmal mit beiden Varianten auseinandergesetzt und die Funktionalität überprüft. Beide erfüllen die Anforderung, dass sie die relevanten Informationen ausgeben und die beiden Buttons die entsprechende Funktion ausführen. Bei der Analyse des Quelltextes entdeckt sie keine potenziellen Fehlerquellen. Beide Implementierungen fangen Edge-Cases zufriedenstellend ab. Einzig bei der Handhabung der E-Mail-Adressen und Telefon-Nummer gibt es einen kleinen Unterschied. Bei der Entwicklerversion wird der Button mit dem Text „Kein Eintrag vorhanden“ ausgegeben. Der Nutzer kann immer noch auf den Button klicken und die Aktionen ausführen. Hier findet sie die Lösung der Künstlichen Intelligenz besser, denn in diesem Fall werden die Buttons nicht angezeigt.

Implikationen

Die Fallstudie zeigt, dass die generative Künstliche Intelligenz einen erheblichen Beitrag zur Beschleunigung des Entwicklungsprozesses leisten kann. Mit einer Reduktion der Entwicklungszeit von fast 50 % ist es eine eindrucksvolle Effizienzsteigerung. Insbesondere bei zeitkritischen Projekten ist dies von großer Bedeutung.

Ebenso wurde gezeigt, dass die Codequalität auf einem hohen Niveau liegt. Insbesondere die Strukturierung und Kommentierung des Quelltextes. Die Kommentierung ist besonders hervorzuheben, da diese bei Entwicklern meist vernachlässigt wird. Dies hat weitreichende Implikationen für die Automatisierung von Aufgaben, die bisher ausschließlich Menschen vorbehalten waren, wie z. B. die UI-Entwicklung.

Die Ergebnisse zeigen jedoch, dass die Flexibilität und Präzession der generativen künstlichen Intelligenz noch nicht auf dem nötigen Stand sind. Dies beruht darauf, dass etliche Korrektur- und Iterationsschleifen bedarf, um ein Resultat zu erlangen, das nah genug an den Vorgaben liegt. Dies unterstreicht, dass menschliches Eingreifen noch erforderlich ist. Es ist also angebracht, generative künstliche Intelligenzen als ein unterstützendes Werkzeug anzusehen, anstelle von einem Tool, das alles automatisiert erledigen kann.

Fazit

Die Fallstudie belegt, dass generative Künstliche Intelligenz ein vielversprechendes Werkzeug in der Softwareentwicklung ist. In den Bereichen Entwicklungszeit und Codequalität sind diese eindeutig konkurrenzfähig mit Softwareentwicklern. Ebenso sind sie in der Lage, qualitativen und funktionsfähigen Code zu produzieren. Dies wurde durch die Beurteilung des Senior-Entwicklers bestätigt, da diese den generierten Quelltext bevorzugt.

Gleichzeitig muss festgehalten werden, dass die menschliche Programmierung ihre Stärken im Bereich der Kreativität und Präzession hat. Insbesondere, wenn nach strickten Designvorgaben entwickelt wird. Trotz möglicher Vorteile der Künstlichen Intelligenz kommt diese nicht ohne einen Entwickler mit Auge zum Detail vorbei.

Insgesamt zeigt diese Arbeit, dass KI ein wertvolles Tool im Entwicklungsprozess sein kann. Die Symbiose aus menschlicher Expertise und der Effizienz der KI scheint aktuell der beste Ansatz, um die Effizienz und Qualität des Projekts zu maximieren.

Anhang

A) Literaturverzeichnis

(Bender et al., 2021)

**B) Dependencies**

"@emotion/styled": "^11.11.0",  
"@expo/vector-icons": "^13.0.0",  
"@expo/webpack-config": "^19.0.0",  
"@mui/icons-material": "^5.14.7",  
"@mui/material": "^5.14.7",  
"@mui/x-data-grid": "^6.12.1",  
"@mui/x-date-pickers": "^6.12.1",  
"@react-native-async-storage/async-storage": "1.18.2",  
"@react-native-community/netinfo": "^11.2.1",  
"@react-navigation/bottom-tabs": "^6.5.8",  
"@react-navigation/native-stack": "^6.9.13",  
"axios": "^1.5.0",  
"date-fns": "^2.30.0",  
"dayjs": "^1.11.9",  
"expo": "~49.0.8",  
"expo-status-bar": "~1.6.0",  
"moment": "^2.29.4",  
"react": "^18.2.0",  
"react-datepicker": "^4.16.0",  
"react-dom": "^18.2.0",  
"react-native": "^0.72.4",  
"react-native-gesture-handler": "~2.12.0",  
"react-native-safe-area-context": "4.6.3",  
"react-native-screens": "^3.27.0",  
"react-native-swipe-gestures": "^1.0.5",  
"react-native-swipe-list-view": "^3.2.9",  
"react-native-vector-icons": "^10.0.0",  
"react-native-web": "^0.19.8",  
"react-password-checklist": "^1.4.3",  
"react-swipe-to-reveal-actions": "^1.1.1",  
"react-swipeable": "^7.0.1",  
"react-swipeable-list": "^1.9.1",  
"react-table": "^7.8.0",  
"react-use": "^17.4.0",  
"zustand": "^4.4.1