INTERNATIONALE HOCHSCHULE

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

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 eine Senior-Entwicklerin bewertet, welche die 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 zunächst die Grundlagen der KI und der generativen KI erläutert. Im Anschluss werden die wichtigsten Grundlagen der später aufgeführten Softwareentwicklung dargelegt. Insbesondere die verwendeten Frameworks. Des Weiteren wird noch grundlegende Konzepte der Software-Entwicklung und des in der Fallstudie verwendeten Frameworks diskutiert. Zum Schluss werden aktuelle Chancen und Limitationen diskutiert.

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). In dieser Arbeit wird die KI nach der ersten Perspektive definiert.

Generative KI

Pyinaya et al. (2023, S. 2) definieren generative Künstliche Intelligenz als eine Ansammlung von Techniken und Modellen, die Zusammenhänge 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 bereitzustellen 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.
Dushel (2024, S. 139) spricht davon, dass KI die Softwareentwicklung positiv beeinflusst. Insbesondere, da dadurch Softwarefehler vermindert werden und die Entwicklungsleistung erhöht wird. Er fügt an, dass die Entwicklungs-Performance vor allem durch in IDE integrierte Coding Assistenten wesentlich erhöht.

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. Im Anschluss werden die allgemeinen Softwareanforderungen definiert. Zum Schluss wird die Implementation dargestellt. Zunächst geht es um die Implementierung des Entwicklers, im Anschluss um die generierte Version.

Fallstudie

Um die Effektivität und Qualität der durch die KI produzierten Software zu überprüfen, wird eine Fallstudie durchgeführt. Diese besteht darin, eine 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 eine Senior-Entwicklerin bewertet. Hier sind die Hauptpunkte: Optisch ansprechend, Code Style, ist der Code verständlich und eine erste Einschätzung, welche Implementation weniger störanfällig ist. Das Ziel dieser Fallstudie ist es, herauszufinden, ob eine generative Künstliche Intelligenz in der Lage ist, mittels konkreter Vorgaben eine React-Native-UI-Komponente zu erstellen. Die Komponente soll nicht nur implementiert werden, sondern auch eine ausreichende Qualität aufweisen, sodass diese in der Produktion eingesetzt werden kann.

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. Der Zweck der Mitgliederverwaltung ist es, alle Bauern im Verband zu erfassen und ihnen darüber passende Angebote bereitzustellen. Die wesentlichen Merkmale, die mittels der Mitgliederverwaltung erfasst werden, sind die Kontaktdaten, die Flächen, die angebauten Früchte und die Mitgliedsbeiträge. Anhand der Mitgliedsbeiträge

ist der Bauer berechtigt, Dienstleistungen in Anspruch zu nehmen. Auf diesem Aspekt liegt das Hauptaugenmerk der Mitgliederverwaltung.Die wichtigste Dienstleistung in dem umfassenden Angebot ist die Buchung einer Beratung. Sobald der Bauer die Mitgliedschaft angemeldet hat und die entsprechenden Flächen angegeben hat, kann er eine Beratung zu seinen Flächen oder Kulturen bekommen. Dabei fährt ein Berater zu dem Bauern, besichtigt die Flächen und berät den Bauern. An diesem Punkt wurde beschlossen, die Arbeit der Berater zu erleichtern. Sie sollen eine eigene Anwendung bekommen, um die Beratungsstunden zu verwalten. Da diese Funktionalität stark von der Mitgliederveraltung abweicht und Offline-funktionieren muss, wurde beschlossen, dafür eine eigene Handy-App zu entwickeln.

Die Handy-App basiert auf dem Java-Script-Framework React Native. Die Entscheidung ist auf dieses gefallen. Ausschlaggebend war die bereits vorhandene Expertise mit anderen React-Frameworks. Dadurch entfällt die Einarbeitungszeit in andere Tech-Stacks. Ein weiterer Vorteil, der sich daraus ergibt, ist die Generierung einer IOS und Android-Anwendung. Mit einer Codebasis können also native Apps für beide Betriebssysteme generiert werden. Das reduziert den Programmieraufwand und die Gefahr, dass die Codestände auseinanderlaufen, existiert nicht.
Für die Systemarchitektur wurde eine Server-Client-Architektur gewählt. Die Endgeräte kommunizieren mittels HTTPS-Requests mit dem Backend. Da die Mitgliederverwaltung bereits implementiert war und die Daten bereits in der Datenbank eingefügt sind, wurde beschlossen, die Mitgliederverwaltung um eine REST-API zu erweitern. Diese API stellt alle nötigen Endpunkte für die App bereit. Damit die App auch offline funktioniert, werden die Daten erst einmal lokal zwischengespeichert. Wenn eine Internetverbindung besteht, werden die Daten dann dem Backen übermittelt. Insbesondere wird auf die Datenintegration geachtet.

Insgesamt setzt sich die App aus folgenden Screens zusammen. Zunächst interagiert der Berater mit dem Log-in-Screen. Ein weiterer Screen ist das Profil, hier kann der Berater sein E-Mail und Password ändern. Der Dritte sind die Beratungsstunden. Das ist der Hauptscreen der App. Hier sieht er alle Stunden, die er geleistet hat. Diese sind absteigend nach dem Datum des Eintrages sortiert. Hier kann der Berater die neuen Stunden eintragen. Es gibt noch die Funktionalität, alle Elemente anders zu sortieren und zu filtern. Nach Fläche, Name, ID, Kultur oder Tätigkeit. Diese Ansicht hat den intern den Namen ConsultingSessionList. Eine weitere Funktionalität ist die Historie. Dabei werden alle Einträge zu einem Bauern aufgelistet. In dieser Ansicht soll nun eine neue Kachel eingefügt werden. In dieser Kachel sind alle Informationen zu dem Bauern, dessen Historie angezeigt wird. In der ersten Zeile sollen der Name und die ID des Bauern ausgeben werden. Darunter die Kontaktdaten. Im Anschluss eine Übersicht mit allen Flächen, die der Bauer bei dem Verband angemeldet hat. Zum Schluss sollen zwei Buttons sein, mit denen der Berater Direktkontakt mit dem Bauern aufnehmen kann. Einmal ein Button, der die Telefon-Anwendung auf dem Handy öffnet. Der andere Button öffnet das E-Mail-Programm. Für die Umsetzung sollen bereits vorgefertigte Komponenten der Material-UI

genutzt werden. Die Implementation dieser Komponente ist die Fallstudie. Einmal wird diese von einem Entwickler-umgesetzt und einmal durch eine generative Künstliche Intelligenz.

Die Kachel soll mittels der Komponente ConsultingSessionMemberContactInfo umgesetzt werden. Die Komponente dient rein alleine der Ausgabe der Daten. Dementsprechend wird keine weitere Logik in der Komponente verwendet. Die Komponente wird in der ConsultingSessionList inkludiert und in den HTML-Baum eingefügt. Die konditionelle Anzeige der Komponente wird über die ConsultingSessionList gesteuert. In dieser findet auch der Request an die Mitglieder-Verwaltung statt. Das Ergebnis wird in einem React-State gespeichert. Nach dem erfolgreichen Abruf der Daten werden diese aufbereitet und der ConsultingSessionMemberContactInfo übergeben. Dies geschieht in Form eines Parameters, im React-Native auch Prop genannt.

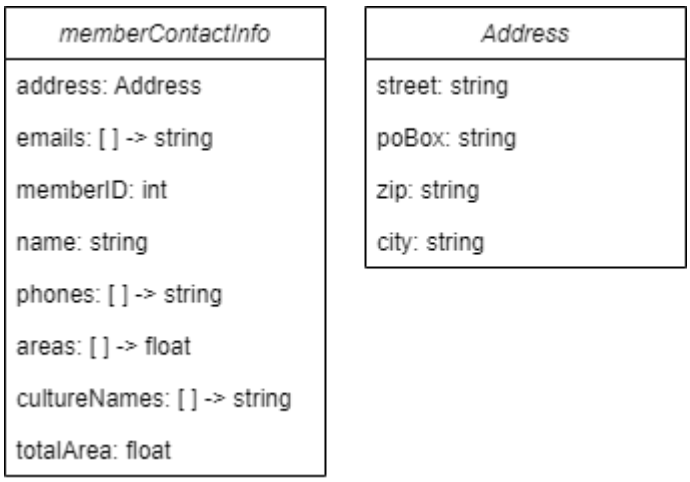

Abbildung 1: Klassendiagramme für memberContactInfo Ursprung: Eigene Darstellung

Softwareanforderungen

Basierend auf der gewählten Software-Architektur, der Einsatzumgebung und Ziel ergeben sich folgende formale Anforderungen an die zu implementierende Software:

- 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 und Material-UI eingesetzt
- Die Kommunikation findet über das Internet mittels HTTPS Request statt.
- Die Beratungsstunden-App interagiert mit dem Restservice der Mitgliederverwaltung.

Neben 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.

Folgende Anforderungen existieren in Bezug auf die fertig implementierten Komponenten. Die Komponenten sollen optisch nahe an der Vorgabe im Screen-Design sein. Es ist nicht nötig, dass es eins zu eins gleich aussieht. Es ist möglich, Abweichungen einzugehen, wenn diese der UI/UX weiterhelfen. Durch die Komponente darf kein Fehler entstehen, der das Programm zum Beenden zwingt. Zu guter Letzt muss der Code qualitätstechnisch auf dem höchsten Stand sein. Dies ist erreicht, wenn der Codestil gut ist und deklarierter Code geschrieben wird. Sobald die Anforderungen erfolgreich umgesetzt sind, ist die Implementation als abgeschlossen zu betrachten.

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 Planung der HTML-Strukturierung. Das UI-Element wird in vier Bereiche unterteilt. In der ersten sind Name und Mitglieds-ID in einer Reihe. Der zweite Bereich enthält die Adressdaten. Zunächst kommt das Icon, dann die Anschrift. Dabei steht die Postleitzahl und Stadt unter der

Straße. Alle Daten bezüglich der Flächen stehen im dritten Abschnitt. Zunächst werden alle einzelnen Flächen mit den entsprechenden Früchten aufgelistet. Danach wird die Gesamtanzahl berechnet. Im vierten sind die Buttons mit den Kontaktmöglichkeiten. Anhand dieser Aufteilung wird zunächst das MUI-Element Box verwendet. Die Komponente dient als Container. Für jeden Abschnitt wird ein View-Element eingefügt. Damit ist die erste Strukturierung der Elemente abgeschlossen und das Skelett der Komponente ConsultingSessionMemberContactInfo besteht.

Der erste Bereich, der programmiert wird, ist der erste Container mit dem Namen und der ID. Für die Ausgabe des Namens und der ID wird ein Java-Template String verwendet. Dieser ist so aufgebaut: ` ${memberContactInfo.name} [${memberContactInfo.memberID}`. Dies ist der Inhalt einer View-Komponente.Danach geht es an den zweiten Bereich, die Kontakt-Daten. Um einen kritischen Fehler zu vermeiden, der unweigerlich zum Absturz des Programmes führt, muss überprüft werden, ob alle Kontaktdaten vorhanden sind. Dieser Umstand kann passieren, wenn in der Mitgliederverwaltung die Daten unzureichend gepflegt wurden. Um diesen Fehler abzufangen, wird der Bereich nur dann ausgeben, wenn die Abfrage, !***Object***.values(address).every(value => value === '' || value === null), den Booleschen Wert wahr annimmt. Damit die Daten nebeneinander angezeigt werden, werden zwei weitere View-Elemente hinzugefügt. Im Ersten kommt das Location-Icon von MUI, <PlaceIcon />. Im anderen View-Element die Adressdaten. Hier ist es zu beachten, dass die Straße nicht hinterlegt ist, sondern eine PO-Box. Damit die beiden View-Elemente nebeneinander stehen, wird auf das Elternelement Display Flex und Flexdirection row angewandt. Im Anschluss geht es mit der Implementierung des dritten Abschnitts weiter. 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. Für die Ausrichtung der Daten wird justify: space-between verwendet. Darunter befindet sich die Anzeige der Gesamtfläche, gerundet auf zwei Nachkommastellen. Zum Schluss sind hier die interaktiven Buttons. Diese sind eigene View-Elemente. Sie bekommen abgerundete Ecken. Die Beschreibung des Buttons ist einmal ein aussagekräftiges Icon, respektiv Telefon oder E-Mail, gefolgt von den Daten. Für die Funktionalität wird die React-Native-Funktion Linking.openURL verwendet. 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. Im letzten Schritt werden die Stadtart-Stylings der MUI-Elemente angepasst. Insbesondere die Font-Weight der Schrift und die Abstände. Das Ergebnis ist eine funktionierende Komponente, die optisch exakt wie die Vorgabe im Screendesign aussieht.

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 Screendesign; 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 E-Mail 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.

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

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

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

Abbildung 4) Ergebnis der Iteration; Ursprung: Eigene Darstellung

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.

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.

Bei dieser Implementation ist zu erwähnen, dass die HTML-Grundstruktur bei der ersten Iteration bereits zufriedenstellend generiert wurde. Bei der Optik sah es etwas anders aus. Selbst mit der Anweisung, dass sich die KI exakt an das Screendesign halten soll, wich es immer weiter ab. Dadurch sind die meisten Iterationsschritte zustande gekommen. Dies weist bereits auf die ersten Chancen und Limitationen hin. Die größte Chance ist das schnelle Erstellen der HTML-Struktur. Die größte Limitation liegt in dem Einhalten der genauen Vorgaben.

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.

In der theoretischen Fundierung wurde die Studie von Peng et al. (2023) verwiesen, die zeigte, dass Künstliche Intelligenz die Entwicklungszeit um 58,5 % Prozent reduzieren kann. Dieses Ergebnis spiegelt sich in der Fallstudie wider. Indem die Künstliche Intelligenz für dieselbe Aufgabe ca. 50 % schneller war. Diese Übereinstimmung zwischen Theorie und Praxis verdeutlicht, dass die theoretischen Erwartungen bezüglich der Effizienz von KI-gestützter Entwicklung weitgehend erfüllt wurden.

Anhand der aufgeführten Gründe für die Verzögerungen würde sich ein hybrider Ansatz eignen. Die KI wird benutzt, um eine Boilerplate-HTML-Struktur zu generieren, und der Entwickler übernimmt die kreativeren Aufgaben. Insbesondere das Styling.

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.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass die Entwicklungsgeschwindigkeit durch die Künstliche Intelligenz stark erhöht wird. Beide Implementationen verfügen über hohe Standards in Bezug auf die Codequalität, die Codelesbarkeit und die Fehleranfälligkeit. Beide Versionen könnten nach Aussage der Seniorentwicklerin bedenkenlos in den Live-Stand übernommen werden. Die Senior-Entwicklerin war erstaunt, wie weit die Technologie der Generativen künstlichen Intelligenzen entwickelt wurde. Insbesondere das Verarbeite n von Screenshots. Des Weiteren stimmen die theoretischen Erwartungen mit den praktischen Erkenntnissen überein. Die Ergebnisse machen deutlich, dass künstliche Intelligenzen ein wertvolles Werkzeug für Entwickler sind, schnell einen funktionierenden Prototyp zu erstellen. Dieser Prototyp kann dann von dem Entwickler verbessert und optimiert werden.

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 Werkzeug im Entwicklungsprozess sein kann. Besonders dvielversprechend ist der Ansatz, die KI zur Generierung der Grundstruktur des Quellcodes einzusetzen, wodurch eine Einhaltung der Coding-Standards sichergestellt wird. Der Entwickler kann diese Basis nutzen, um den Code gezielt weiterzuentwickeln und auf projektspezifische Anforderungen, insbesondere grafische Aspekte, einzugehen. Dieser hybride Ansatz hat das Potenzial, die Entwicklungsgeschwindigkeit und Effizienz zu maximieren und gleichzeitig eine hohe Codequalität zu gewährleisten.

Anhang

A)Literaturverzeichnis

1. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March 3). On the dangers of stochastic parrots: Can language models be too big? 🦜 *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610-623. https://doi.org/10.1145/3442188.3445922
2. Cassieri, P., Romano, S., & Scanniello, G. (2024, March 12). Generative Artificial Intelligence for test-driven development: GAI4-TDD. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 902-906). IEEE. https://doi.org/10.1109/SANER60148.2024.00098
3. Duschl, D. (2024). *Nutzung der KI zur Fehlervermeidung*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-44337-5_7
4. Lämmel, U., & Cleve, J. (2020). *Künstliche Intelligenz [electronic resource]: Wissensverarbeitung - neuronale Netze* (5th ed.). Hanser.
5. Occhino, T. (2015, March 26). React Native: Bringing modern web techniques to mobile. *Engineering at Meta*. https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/
6. Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot.
7. Pinaya, W. H. L., Graham, M. S., Kerfoot, E., Tudosiu, P.-D., Dafflon, J., Fernandez, V., Sanchez, P., Wolleb, J., da Costa, P. F., Patel, A., Chung, H., Zhao, C., Peng, W., Liu, Z., Mei, X., Lucena, O., Ye, J. C., Tsaftaris, S. A., Dogra, P., Feng, A., Modat, M., Nachev, P., Ourselin, S., & Cardoso, M. J. (2023). Generative AI for medical imaging: Extending the MONAI framework. *arXiv*. https://arxiv.org/abs/2307.15208
8. Vision - Material UI. (2024, September 12). *Material UI*. https://mui.com/material-ui/discover-more/vision/

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
```