Handout

IU Internationale Hochschule

Studiengang: Informatik

Objektorientierte Programmierung 2

Textalyzer

Patryk Hegenberg, Manuel Keidel, Krzysztof Stankiewicz

Matrikelnummer: 102209025, 102204198, 102208812

Betreuende Person: Frank Krickel

Abgabedatum: 21.01.2024

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Rahmen des fortgeschrittenen Kurses „Objektorientierte Programmierung 2“ wurde die Herausforderung angenommen, eine innovative Lösung als Prüfungsleistung zu entwickeln. In einem sorgfältig strukturierten Entwicklungsprozess entschied sich unsere Gruppe dafür, eine Android-Anwendung in Java zu kreieren, die sich mit einem zunehmenden Problem auseinandersetzt.

Der Fokus unseres Projekts liegt auf der steigenden Komplexität von Texten, die vielen Menschen Schwierigkeiten bereitet, den Inhalt zu verstehen, insbesondere wenn sie nicht mit dem spezifischen Kontext vertraut sind. Diese Herausforderung manifestiert sich in unterschiedlichen Lebensbereichen, die von Schülern über Studenten bis hin zu Bürgern reichen, die mit dem öffentlichen Dienst in Berührung kommen. Neue Themen in der Schule, offizielle Behördenbriefe, juristische Texte oder sprachliche Barrieren stellen oft unüberwindbare Verständnishürden dar.

Das Erkennen dieser Herausforderung führte zu der Entscheidung, eine Android-App zu entwickeln, die auf generativer Künstlicher Intelligenz[1] (AI) basiert und dazu dient, den Zugang zu komplexen Texten zu erleichtern.

Unser Ansatz besteht darin, mithilfe von AI Texte zu analysieren und zu vereinfachen. Diese Technologie ermöglicht nicht nur die Reduktion der Komplexität, sondern auch eine individualisierte und leicht verständliche Präsentation von Informationen. Durch die Berücksichtigung des Kontextes und der spezifischen Herausforderungen der Zielgruppen streben wir nicht nur eine allgemeine Vereinfachung an, sondern auch eine personalisierte Zugänglichkeit.

Die Entscheidung, AI in unsere Lösung zu integrieren, wurde durch ihre bewährte Leistungsfähigkeit in der natürlichen Sprachverarbeitung und Textgenerierung motiviert. Dieser Ansatz adressiert nicht nur die aktuelle Problematik der Komplexitätsreduktion in Texten, sondern schafft auch eine Grundlage für die zukünftige Entwicklung von kognitiven Anpassungsfähigkeiten.

Im weiteren Verlauf dieser Arbeit werden wir detailliert auf die technischen, funktionalen und nichtfunktionalen Aspekte unserer entwickelten Android-Anwendung eingehen. Darüber hinaus werden wir den Entwicklungsprozess, Herausforderungen und Lösungen sowie die Schlussfolgerungen aus unserem Projekt umfassend behandeln.

2 Problemstellung und Ideenfindung

Die zunehmende Komplexität von Texten stellt eine bedeutende Herausforderung für eine breite Palette von Anwendern dar, insbesondere in Bildungs- und Rechtskontexten. Die Unzugänglichkeit von komplexen Texten führt nicht nur zu erheblichen Verständnisschwierigkeiten, sondern hat auch tiefgreifende Auswirkungen auf

[1] Generative künstliche Intelligenz ist eine Art von künstlicher Intelligenz, welche verschiedene Arten von Inhalten generieren kann. („Was Ist Generative Künstliche Intelligenz (KI)?“, n. d.)

die Effizienz und Effektivität des Informationsaustauschs. In Anbetracht dieser Problematik ergab sich die Notwendigkeit, innovative Ansätze zu erkunden, um die Barrieren des Verständnisses zu überwinden und somit die Inklusivität und Erreichbarkeit von Wissen zu fördern.

Die Schlüsselherausforderung besteht in der Tatsache, dass sich Texte in unterschiedlichen Kontexten und Fachgebieten immer spezifischer und komplexer entwickeln. Insbesondere für Schüler, Studenten (Kühn, 1993, S. 28) und Bürger, die mit den Schriftstücken des öffentlichen Dienstes konfrontiert werden (Heutger, 2017), wird der Zugang zu bedeutungsvollen Informationen zu einem überaus anspruchsvollen Unterfangen. Diese Zielgruppen sind häufig mit Fachterminologie, juristischen Formulierungen und neuen Bildungsinhalten konfrontiert, was das Verständnis erheblich erschwert.

In Anbetracht dieses Problems führte unsere Ideenfindung zu einer innovativen Lösung: die Integration von AI in den Prozess der Textvereinfachung. Dies verspricht, nicht nur den Zugang zu komplexen Texten zu erleichtern, sondern auch eine individualisierte und leicht verständliche Präsentation von Informationen zu ermöglichen.

Die Grundidee besteht darin, dass die AI-Engine Texte analysiert und verarbeitet, um anschließend eine vereinfachte Version zu generieren, Inhalte verständlich zusammenzufassen, oder die Stimmung eines Textes zu analysieren.

Die Anwendung von AI in diesem Kontext eröffnet die Möglichkeit, komplexe Ausdrucksformen und Fachterminologie in leicht verständliche Konzepte umzuwandeln (Sharma, 2023). Die algorithmische Analyse ermöglicht es, die Schlüsselinformationen zu identifizieren und hervorzuheben, während unwesentliche Details reduziert werden. Dieser Ansatz verspricht, nicht nur die Lesbarkeit zu verbessern, sondern auch das Verständnis zu fördern, indem er komplexe Zusammenhänge auf verständliche Art und Weise darlegt.

Die Wahl, AI in unsere Lösung zu integrieren, wurde durch ihre bewährte Leistungsfähigkeit in der natürlichen Sprachverarbeitung und Textgenerierung motiviert. Wir sind der Überzeugung, dass dieser Ansatz nicht nur die aktuelle Problematik der Komplexitätsreduktion in Texten angeht, sondern auch eine Grundlage für die zukünftige Entwicklung von kognitiven Anpassungsfähigkeiten schafft.

In der nächsten Entwicklungsphase werden wir uns darauf konzentrieren, die generative AI-Technologie nahtlos in den Entwicklungsprozess der Android-App zu integrieren. Durch eine akribische Umsetzung dieser Idee streben wir danach, nicht nur die unmittelbare Problematik zu adressieren, sondern auch einen wegweisenden Beitrag zur Entwicklung von benutzerzentrierten, adaptiven Textvereinfachungstechnologien zu leisten.

3 Anforderungsanalyse

3.1 Funktionale Anforderungen

Im Rahmen der Entwicklung unserer Android-Anwendung zur Textverarbeitung, wurden spezifische funktionale Anforderungen festgelegt, um sicherzustellen, dass die App den Nutzern eine umfassende und maßgeschneiderte Erfahrung bietet. Die vorliegende Anforderungsanalyse dient der detaillierten Darstellung dieser funktionalen Anforderungen, wobei der Fokus auf der Effektivität der Textzusammenfassung, Inhalts- und Stimmungsanalyse sowie der Vielseitigkeit in Bezug auf Eingabeoptionen liegt.

Die zentrale Funktionalität der App besteht in der Textzusammenfassung, wobei die generative KI es ermöglicht, eingegebene Texte automatisch zu analysieren und prägnante Zusammenfassungen zu generieren. Diese Anforderungen zielen darauf ab, dem Nutzer eine individuelle und präzise Erfahrung bei der Informationsvermittlung zu bieten.

Die Inhaltsanalyse repräsentiert eine weitere entscheidende Funktionalität, wobei die App den Text in seiner Gesamtheit analysieren und Schlüsselinformationen extrahieren kann. Dies schließt die Identifikation und Hervorhebung von Hauptthemen sowie relevanten Details ein. Hierdurch wird nicht nur eine tiefergehende Analyse des Textinhalts ermöglicht, sondern auch eine vereinfachte Interpretation und Priorisierung von Informationen.

Die Integration einer generativen KI zur Stimmungsanalyse hebt die App auf eine höhere Ebene der Textverarbeitung. Die Fähigkeit, zwischen positiven, negativen oder neutralen Tönen im Text zu differenzieren, eröffnet Anwendungsszenarien, in denen die emotionale Intention eines Textes von entscheidender Bedeutung ist. Dies ermöglicht eine präzisere Interpretation und ein tieferes Verständnis des Kontexts.

Um die Nutzerfreundlichkeit und Vielseitigkeit der Anwendung zu maximieren, wurde eine Unterstützung verschiedener Eingabeformate implementiert. Die App ermöglicht die Verarbeitung von vom Nutzer eingegebenem Text, kopiertem Text und PDF-Dateien von der Festplatte. Die automatische Erkennung und Extraktion von Text aus PDF-Dateien erleichtert den Nutzern den Zugriff auf Informationen aus verschiedenen Quellen.

Die Verarbeitungsoptionen für den Nutzer bieten eine entscheidende Schnittstellenoption, um die Art der Verarbeitung auszuwählen. Der Benutzer kann zwischen einer Textzusammenfassung, einer Inhaltsanalyse oder einer Stimmungsanalyse wählen. Diese klare Schnittstelle fördert eine benutzerfreundliche Erfahrung und ermöglicht eine anpassbare Nutzung der App.

Insgesamt gewährleisten diese funktionalen Anforderungen, dass die entwickelte Android-Anwendung nicht nur anspruchsvolle Textverarbeitungsfunktionen bietet, sondern auch den individuellen Anforderungen und Präferenzen der Nutzer gerecht wird. Der Fokus auf Präzision und nahtloser Integration trägt dazu bei, eine hochfunktionale Anwendung zu schaffen.

3.2 Nicht-Funktionale Anforderungen

Die nicht-funktionalen Anforderungen, die die Leistung, Sicherheit, Skalierbarkeit, Wartbarkeit, Zuverlässigkeit, Kompatibilität und Dokumentation betreffen, wurden präzise festgelegt, um eine effiziente und verlässliche Funktionsweise unserer Textverarbeitungs-App zu gewährleisten.

Die Leistungsanforderungen der App sind darauf ausgerichtet, eine nahtlose Benutzererfahrung zu ermöglichen. Eine flüssige Bearbeitung wird angestrebt, wobei Ladezeiten von maximal 5 Sekunden als kritische Schwelle definiert sind. Dies gewährleistet, dass Nutzer die Anwendung effizient und ohne signifikante Verzögerungen verwenden können. Des Weiteren darf der Request an die OpenAI API nicht länger als 60 Sekunden dauern, um sicherzustellen, dass die generative KI-Analyse zeitnah erfolgt und die Nutzererwartungen erfüllt werden.

Im Bereich der Sicherheit wird besonderes Augenmerk darauf gelegt, dass keine Daten von Nutzern gespeichert oder zu Analysezwecken verwendet werden. Diese Gewährleistung beschränkt sich auf die interne Verarbeitung innerhalb unserer App. Dieser Ansatz berücksichtigt Datenschutzprinzipien und garantiert, dass sensible Nutzerinformationen nicht unbefugt gesammelt oder weitergegeben werden.

Hinsichtlich der Skalierbarkeit hebt sich die App durch ihre lokale Ausführung auf dem Smartphone hervor. Diese dezentrale Ausführung ermöglicht eine mühelose Skalierbarkeit, da sie nicht auf externe Server angewiesen ist. Die App kann somit effizient auf unterschiedlichen Geräten und unter verschiedenen Bedingungen agieren.

Die Wartbarkeit des Quellcodes ist von grundlegender Bedeutung für die langfristige Entwicklungsstrategie der App. Der Code wird sorgfältig dokumentiert und wartungsfreundlich gestaltet. Eine klare Dokumentation ermöglicht eine transparente Analyse und erleichtert zukünftige Aktualisierungsprozesse. Die Struktur des Codes ist so konzipiert, dass die Integration neuer Funktionen nahtlos erfolgt und die Skalierbarkeit der App gewährleistet ist.

In Bezug auf Zuverlässigkeit wird eine stabile und zuverlässige Funktionalität der App unter verschiedenen Bedingungen und auf verschiedenen Geräten angestrebt. Automatische Fehlererkennung und -behebung sind implementiert, um eine kontinuierliche Verfügbarkeit zu gewährleisten. Diese Maßnahmen minimieren mögliche Unterbrechungen und tragen zu einer positiven Nutzererfahrung bei.

Die Kompatibilitätsanforderungen betreffen sowohl die Unterstützung verschiedener Android-Versionen als auch die Integration von Schnittstellen für mögliche Erweiterungen oder Kooperationen mit anderen Anwendungen. Dies stellt sicher, dass die App eine breite Nutzerbasis anspricht und flexibel auf zukünftige Entwicklungen reagieren kann.

Abschließend wird die Dokumentation als nicht-funktionale Anforderung betont. Klare und verständliche Entwicklerdokumentation ist von entscheidender Bedeutung für die kontinuierliche Wartung und Weiterentwicklung der App. Sie ermöglicht neuen Entwicklern einen schnellen Einstieg und stellt sicher, dass die App auch in Zukunft effektiv und effizient gepflegt werden kann.

4 Technische Designentscheidungen

Im Zuge der Entwicklung unserer Android-App haben wir entscheidende technische Designentscheidungen getroffen, die den Entwicklungsprozess beeinflusst und eine optimale Umsetzung unserer Ziele ermöglicht haben. Die getroffenen Entscheidungen reichen von der Architektur über die Programmiersprache und Frameworks bis zur Integration der generativen Künstlichen Intelligenz, Netzwerkkommunikation, Benutzeroberfläche, Plattformauswahl, Teststrategie und Dokumentation.

Die Architektur unserer App basiert auf dem Model-View-ViewModel (MVVM)-Muster. Diese Entscheidung ermöglichte es uns, mit Android Studio schnell eine erste funktionierende Version zu erstellen und diese iterativ zu erweitern. Die klare Trennung von Datenhaltung, Benutzeroberfläche und Geschäftslogik, wie sie in der MVVM-Architektur vorgesehen ist, erleichterte nicht nur die Entwicklung, sondern trägt auch zur Skalierbarkeit und Wartbarkeit des Codes bei.

Bezüglich der Programmiersprache und Frameworks entschieden wir uns aufgrund der Kursanforderungen für Java. Die Auswahl von Java ermöglichte eine breite Kompatibilität mit Android-Geräten und erleichterte die Integration von verschiedenen Bibliotheken und Frameworks. Zu den eingesetzten Bibliotheken gehören okhttp3 und retrofit2 für die effiziente Handhabung von HTTP-Anfragen, google.gson und org.json für die Verarbeitung von JSON-Objekten, tom_roush.pdfbox für das Laden und Lesen von PDFs sowie die Extraktion von Text aus PDFs. Die Kommunikation mit der Android-Plattform wurde durch die grundlegende Verwendung des Android-Frameworks sichergestellt.

Die Integration der generativen KI erfolgte über die OpenAI-API unter Verwendung des GPT-4-Modells. Diese Entscheidung basierte auf der schnellen Verfügbarkeit und den erzielten brauchbaren Ergebnissen. Die Integration der KI ermöglichte fortgeschrittene Textanalysen und Zusammenfassungen, die den Anforderungen unserer App entsprechen.

Die Netzwerkkommunikation wurde durch die Nutzung von HTTP-Requests realisiert, insbesondere in der Kommunikation mit einer REST-API. Diese Entscheidung ermöglichte eine effiziente Datenübertragung zwischen unserer App und externen Servern.

In Bezug auf die Benutzeroberfläche (UI/UX) erfolgte eine klare Aufteilung in mehrere Views, darunter der Hauptbildschirm, die Texteingabeansicht, die PDF-Ladeansicht, die Aufgabenauswahlansicht und die Ergebnisansicht. Die Umsetzung erfolgte mithilfe von XML-Layouts in Android Studio, was eine visuell ansprechende und benutzerfreundliche Oberfläche gewährleistete.

Die Sicherstellung der Funktionen unter 83% der Android-Geräte (Belinski, n. d.) erfolgte durch die Auswahl der Android SDK Version 34 als Zielplattform, mit einer mindest SDK Version von 29. Diese Entscheidung berücksichtigte die Verbreitung von Android-Versionen und ermöglichte eine weitreichende Unterstützung.

Die Teststrategie umfasste die Sicherstellung der Funktionalität durch JUnit-Tests für die wichtigsten Komponenten sowie durch Handtests. Integrationstests und End-to-End-Tests wurden ebenfalls manuell

durchgeführt. Obwohl keine Testautomatisierung implementiert wurde, ermöglichten diese Testmethoden eine umfassende Qualitätssicherung und Überprüfung der App-Funktionalitäten.
Die Dokumentation erfolgte durch die Erstellung von JavaDocs für die Entwickler. Diese detaillierte Entwicklerdokumentation dient als unverzichtbare Ressource für die Wartung und Weiterentwicklung der App und ermöglicht eine transparente Analyse des Codes.
In Summe repräsentieren diese technischen Designentscheidungen das Rückgrat unserer Android-App. Sie bieten nicht nur eine effektive Lösung für die formulierten Anforderungen, sondern tragen auch zu einer robusten, skalierbaren und wartbaren Architektur bei, die den aktuellen Standards und Best Practices in der Android-Entwicklung entspricht.

5 Entwicklungsprozess

Die Entwicklung der vorliegenden Android-Anwendung durchlief einen sorgfältig strukturierten Prozess, der durch umfassende Planung, Visualisierung und iterative Umsetzung geprägt war. Beginnend mit einem groben UML-Klassen-Diagramm (siehe 1) wurde eine initiale Übersicht über die erforderlichen Klassen geschaffen. Parallel dazu wurden Programmablaufdiagramme (siehe 2) und Sequenzdiagramme (siehe 3) erstellt, um die Interaktionen zwischen den Klassen und die Dynamik des Systems zu visualisieren. Weiterhin legten wir zu Beginn des Entwicklungsprozesses auch die Aufgabenverteilung fest. So entschieden wir uns, dass Manuel Keidel die Kommunikation mit der OpenAI Api implementiert, Krzysztof Stankiewicz das Design und die Views schreibt und Patryk Hegenberg für die innere Kommunikation und das Backend der App verantwortlich ist. Dieser frühzeitige Planungsansatz legte den Grundstein für eine durchdachte und strukturierte Entwicklung.
Die Einrichtung der Entwicklungsumgebung erfolgte durch die Integration von Android Studio und Git für die Versionskontrolle. Dies ermöglichte eine effiziente Zusammenarbeit im Team und die Nachverfolgung von Änderungen während des gesamten Entwicklungszyklus. Die Einarbeitung in die Android-Entwicklung wurde durch kleinere Tutorials unterstützt, um ein grundlegendes Verständnis für die Plattform und ihre spezifischen Anforderungen zu erlangen.
Ein entscheidender Schritt war die Nutzung generativer Künstlicher Intelligenz (AI) zur Generierung von Boilerplate Code, der dann an die spezifischen Anforderungen der Anwendung angepasst wurde. Diese Herangehensweise ermöglichte eine beschleunigte Implementierung grundlegender Funktionalitäten.
Die Integration der OpenAI-API erwies sich als anspruchsvolle Phase des Entwicklungsprozesses. Nach dem Erstellen einer ersten Grundversion stellte sich heraus, dass Anpassungen erforderlich waren, um die reibungslose Funktionalität sicherzustellen. Die Überwindung dieser Herausforderungen wurde durch umfangreiche Recherchen, Trial-and-Error-Phasen und die Zusammenarbeit im Team erreicht. Schließlich resultierte dies in einer vollständig funktionsfähigen Grundversion, die den Grundstein für die weiteren

Entwicklungsarbeiten legte.
Die nachfolgende Phase konzentrierte sich auf die Implementierung weiterer Features wie die Texteingabe und das Laden von PDFs. Handgesteuerte Tests wurden eingeführt, um die Funktionalitäten sicherzustellen und die Qualität der App zu gewährleisten. Eine Zwischenvorstellung beim Dozenten ermöglichte wertvolles Feedback, das in die Implementierung von zusätzlichen Funktionen wie der Nutzung der App als PDF-Reader, dem automatischen Start der Verarbeitung und der Möglichkeit zum Teilen der generierten Inhalte einfließen konnte.
Parallel dazu erfolgten Anpassungen im UI/UX-Bereich, um sicherzustellen, dass die Benutzeroberfläche den Best Practices entspricht und eine ansprechende Nutzererfahrung gewährleistet. Gleichzeitig wurden JUnit-Tests für die wichtigsten Grundfunktionen implementiert und JavaDoc-Strings erstellt, um eine umfassende Dokumentation zu gewährleisten.
Die finalen Tests konzentrierten sich auf die Sicherstellung der Funktionalität und die Behebung letzter Fehler. Dies führte zur Schaffung einer vollständig lauffähigen App. Der gesamte Entwicklungsprozess zeichnete sich durch eine methodische Vorgehensweise, transparente Zusammenarbeit im Team und die Nutzung innovativer Technologien aus. Dieser Ansatz ermöglichte nicht nur die effiziente Umsetzung der Anforderungen, sondern auch eine kontinuierliche Verbesserung und Anpassung an sich entwickelnde Bedürfnisse während des gesamten Entwicklungszyklus.

6 Herausforderungen und Loesungen

Die Entwicklungsgruppe stand während des Projekts vor diversen Herausforderungen, die sowohl das Unbekannte der Android-Entwicklung als auch die verbundenen Besonderheiten in der Integration von HTTP-Requests und die Verarbeitung von API-Antworten umfassten. Das Team, das zuvor keine Erfahrung in der Android-Entwicklung besaß, musste sich zunächst mit den Grundlagen vertraut machen.
Ein zentrales Problem bestand zu Beginn darin, dass nur ein Teammitglied Zugang zu einem API-Key hatte. Infolgedessen wurde die erste Version der Anwendung entwickelt, ohne tatsächliche Anfragen an die OpenAI-API zu stellen. Dies führte zu intensiven Diskussionen, Recherchen und Trial-and-Error-Phasen, bis eine Lösung gefunden wurde. Die Entscheidung, den API-Key direkt im Code zu verankern, erwies sich als unpraktisch und führte zur Sperrung des Keys. Das Team reagierte darauf, indem es einen neuen Key generierte, diesen auf sicherem Wege austauschte und schließlich durch das Lesen aus einer Konfigurationsdatei in die App einband.
Weitere Schwierigkeiten traten insbesondere bei der Integration des HTTP-Requests an die OpenAI-API auf. Keines der Teammitglieder hatte zuvor Erfahrung mit diesem Aspekt der Programmierung unter Android, und die grundsätzliche Vorgehensweise war nicht bekannt. Die anfänglichen Schwierigkeiten konzentrierten sich auf das Verständnis und die korrekte Implementierung von HTTP-Requests, insbesondere in der korrekten

Einbindung der Rechte unter Android und Verwenden der ausgewählten Bibliotheken. Durch hartnäckiges Trial-and-Error konnte schließlich ein Teammitglied eine funktionierende Lösung entwickeln.
Eine weitere Hürde ergab sich beim Parsen der JSON-Response. Aufgrund fehlender Erfahrung in der Verarbeitung von JSON in Java und Unkenntnis der verfügbaren Bibliotheken erwies sich diese Aufgabe als weitere Hürde. Auch das Design einer ansprechenden Benutzeroberfläche (UI/UX) war für das Team eine neuartige Herausforderung.
Die Ursache des fehlerhaften Requests wurde durch beharrliches Debugging und experimentelles Probieren behoben. Durch Recherche und schrittweises Vorgehen gelang es schließlich, den gesamten Response zu parsen und für die weitere Verarbeitung vorzubereiten. Ein Teammitglied widmete sich intensiv dem Erlernen von UI/UX-Design unter Android, um die Benutzeroberfläche ansprechend zu gestalten. Die Lösung der Herausforderungen wurde durch den Einsatz von generativer Künstlicher Intelligenz an einigen Stellen im Entwicklungsprozess erleichtert.
Insgesamt zeugt die erfolgreiche Bewältigung dieser Herausforderungen von der engagierten Zusammenarbeit im Team sowie von einer kontinuierlichen Lernbereitschaft und Anpassungsfähigkeit an neue Technologien und Konzepte.

7 Die finale App

Die entwickelte App, wie in Abbildung 4 dargestellt, repräsentiert ein ausgereiftes Produkt, das eine sorgfältig konzipierte Architektur und umfassende Funktionalitäten bietet. Die grundlegende Funktionalität umfasst die flexible Eingabe von Texten, sei es durch manuelle Eingabe, Kopieren oder Extraktion aus PDF-Dateien. Die Integration von generativer Künstlicher Intelligenz über die OpenAI-API hebt die App auf ein neues Niveau, indem sie präzise Zusammenfassungen, inhaltliche Analysen und Stimmungsanalysen, unter Einsatz säuberlich vordefinierter Prompts, für die Benutzer generiert.
Die Vielseitigkeit der App manifestiert sich in der Verarbeitung verschiedener Eingabeformate, darunter freier Text und Text aus PDF-Dateien. Die eingebaute PDF-Reader-Funktionalität ermöglicht es den Nutzern, PDFs direkt in die App zu laden, wodurch eine schnelle und verständliche Darstellung der Inhalte ermöglicht wird.
Die Ergebnisweiterleitungsfunktion eröffnet den Benutzern die Möglichkeit, die erlangten Erkenntnisse unmittelbar mit anderen zu teilen, was die kollaborative Nutzung und Diskussion von Textinhalten erleichtert.
Die bewusste Entscheidung für die Android API 34 stellt sicher, dass die App auf einer breiten Palette von Android-Geräten – etwa 83% – nahtlos funktioniert und somit eine weitreichende Nutzerbasis anspricht.
Die Entwicklerdokumentation der App ist akribisch verfasst und bietet eine umfassende Ressource für Entwickler, um die nahtlose Weiterentwicklung der Anwendung zu gewährleisten. Die bewährte native Entwicklung mit Java trägt zu einer effizienten und performanten Anwendung bei. Um die Stabilität

zu garantieren, wurden umfangreiche Tests durchgeführt, darunter JUnit-Tests, Integrationstests und End-to-End-Tests.
In ihrer Gesamtheit präsentiert die App eine robuste Lösung, die Benutzern eine effektive Möglichkeit bietet, Schlüsselinhalte aus verschiedenen Texten zu extrahieren und übersichtlich darzustellen. Die durchdachte Struktur und Technologieintegration zeugen von einem anspruchsvollen Entwicklungsprozess, der sich in einer hochfunktionalen Anwendung niederschlägt.

8 Fazit

Die Entwicklung der Android-Anwendung im Rahmen des Kurses „Objektorientierte Programmierung 2“ war ein intensiver und lehrreicher Prozess. Das Projekt wurde initiiert, um Lösungen für das steigende Problem der Textkomplexität anzubieten und einen Beitrag zur Überwindung von Verständnisbarrieren zu leisten.
Die umfassende Planung, beginnend mit UML-Diagrammen und Programmablaufdiagrammen, legte den Grundstein für einen strukturierten Entwicklungsprozess. Die Integration generativer Künstlicher Intelligenz zur Generierung von Boilerplate-Code ermöglichte eine beschleunigte Implementierung grundlegender Funktionen. Die Einrichtung der Entwicklungsumgebung mit Android Studio und Git erleichterte die Zusammenarbeit im Team und die Versionskontrolle.
Die Integration der OpenAI-API stellte eine Herausforderung dar, die jedoch erfolgreich gemeistert wurde. Schwierigkeiten bei HTTP-Requests, der Verarbeitung von API-Antworten und der Behandlung von JSON wurden durch intensive Recherche, Trial-and-Error-Methoden und Zusammenarbeit im Team überwunden. Die Implementierung weiterer Funktionen wie Texteingabe, PDF-Verarbeitung und die Nutzung der App als PDF-Reader wurden iterativ umgesetzt. Die Anpassung der Benutzeroberfläche an UI/UX-Best Practices sorgte für eine ansprechende Nutzererfahrung.
Die Teststrategie umfasste JUnit-Tests, Handtests und die letzten Tests konzentrierten sich auf die Sicherstellung der Funktionalität und Fehlerbehebung.
Der Entwicklungsprozess zeichnete sich durch eine methodische Vorgehensweise, transparente Zusammenarbeit und den Einsatz innovativer Technologien aus. Im Team bewältigten wir Herausforderungen durch Lernbereitschaft und Anpassungsfähigkeit.
Abschließend lässt sich sagen, dass die Durchführung dieses Projekts eine spannende und lehrreiche Aufgabe war und einen interessanten Abschluss der Vorlesungsreihe „Objektorientierte Programmierung 2“ darstellt sowie einen Vorgeschmack auf die kommenden Module in der Mobile App Entwicklung gewährt hat.

Literaturverzeichnis

Kühn, P. (1993). *Informationen Deutsch als Fremdsprache*, *20*(2-3), 217–218. https://doi.org/doi:10.1515/infodaf-1993-202-333

Heutger, V. (2017). Der Platz der juristischen Fachsprache in der Experten-Laien-Kommunikation. *HERMES - Journal of Language and Communication in Business*, *19*, 55. https://doi.org/10.7146/hjlcb.v19i36.25838

Sharma, C. V. (2023). GENERATIVE AI FOR RESEARCH WRITING.

Belinski, E. (n. d.). Android API Levels. Verfügbar 28. Januar 2024 unter https://apilevels.com/

Was Ist Generative Künstliche Intelligenz (KI)? (n. d.). Verfügbar 28. Januar 2024 unter https://portal.uni-koeln.de/digital-education/ki-in-der-bildung/was-ist-generative-kuenstliche-intelligenz-ki

A Vorläufiges UML-Diagramm

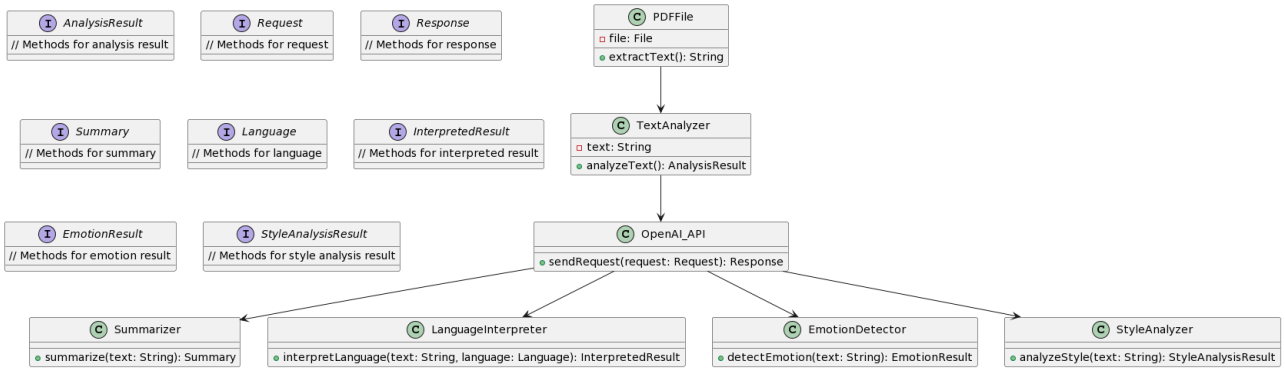

Abbildung 1: vorläufiges UML-Diagramm

B Ablauf-Diagramm

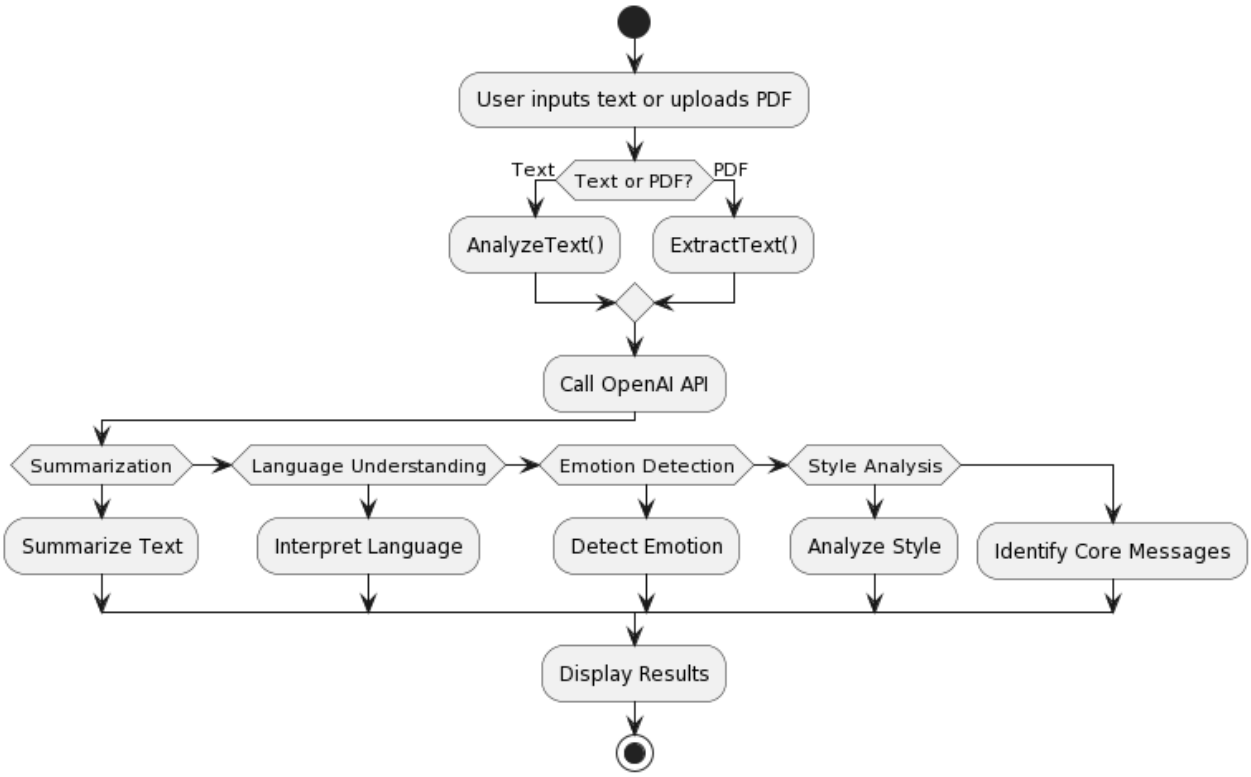

Abbildung 2: Ablauf-Diagramm

C Sequenz-Diagramm

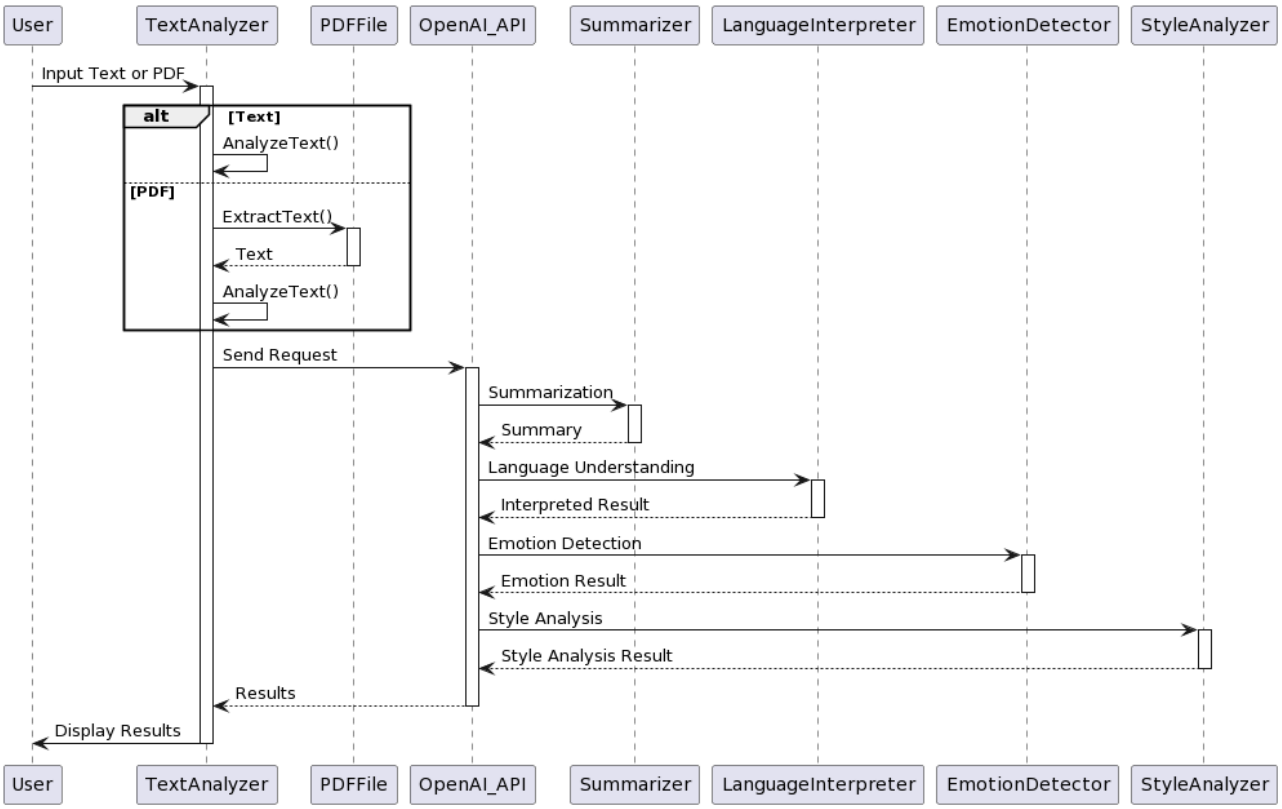

Abbildung 3: Sequenz-Diagramm

D Finales UML-Diagramm

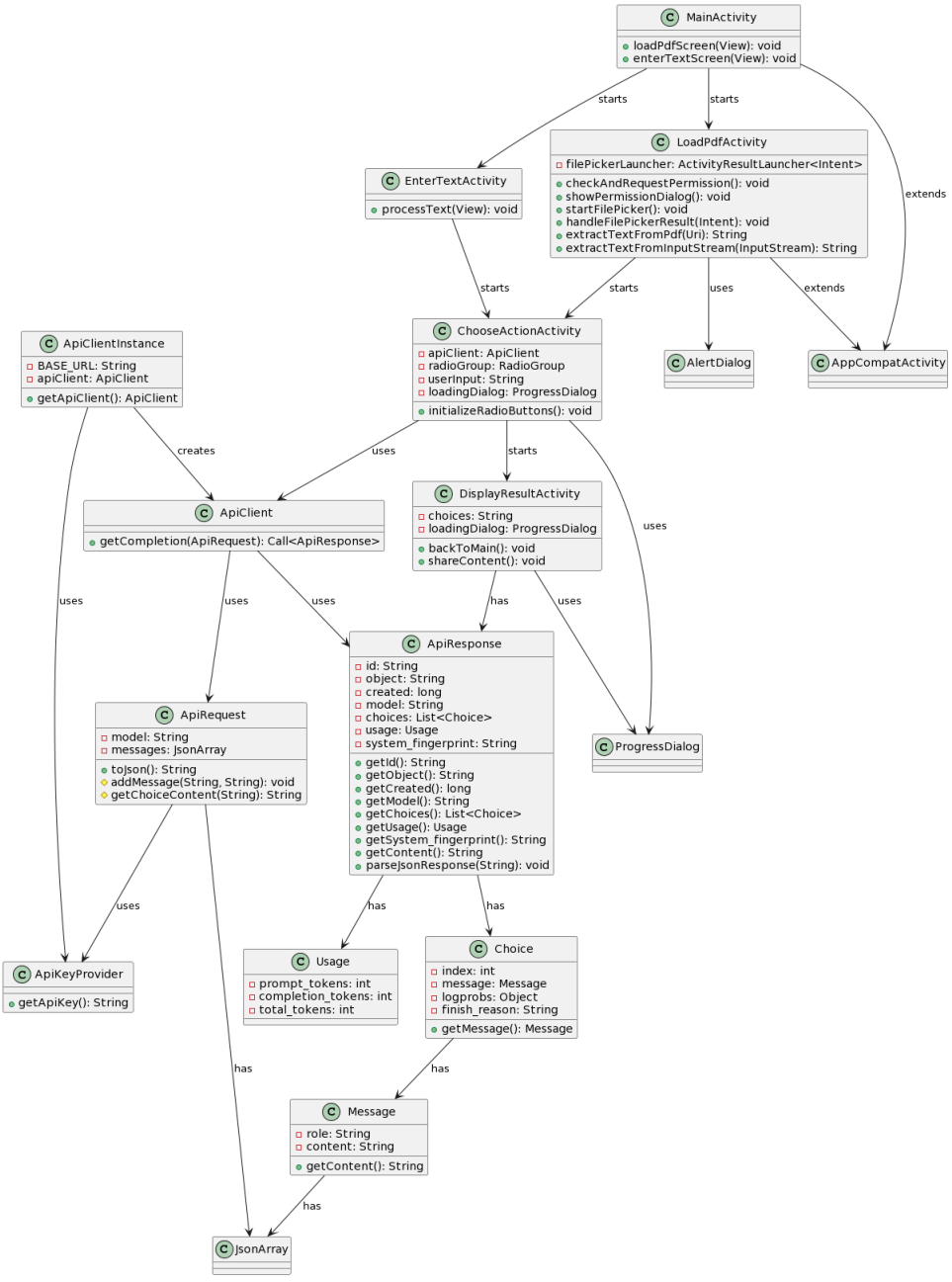

Abbildung 4: finales UML-Diagramm

E Projektidee Pitch

1. Problem:

Texte sind in unserer Gesellschaft allgegenwärtig. Sie finden sich in Nachrichtenartikeln, Büchern, wissenschaftlichen Studien, Gesetzestexten und vielen anderen Quellen. Um diese Texte zu verstehen und zu analysieren, benötigen wir oft spezielle Kenntnisse und Fähigkeiten.

2. Lösung:

Wir entwickeln eine App zur Textanalyse, die es Nutzern ermöglicht, Texte aus verschiedenen Quellen zu analysieren. Die App bietet eine Reihe von Funktionen, darunter:

- **Inhaltliche Zusammenfassung**: Die App erstellt eine kurze Zusammenfassung des Textes, die die wichtigsten Informationen enthält.
- **Bedeutung des Textes in einer Sprache**: Die App versteht die Bedeutung des Textes in einer bestimmten Sprache und identifiziert die Entitäten, Beziehungen und Stimmungen im Text.
- **Emotionen des Textes**: Die App erkennt die Emotionen im Text und identifiziert die Wörter und Phrasen, die mit bestimmten Emotionen verbunden sind.
- **Stil des Textes**: Die App bestimmt den Stil des Textes und identifiziert die Wortwahl, Grammatik und Syntax des Textes.
- **Analyse der Kernaussagen**: Die App identifiziert die Kernaussagen des Textes.

3. Zielgruppe:

Die App richtet sich an Schüler und Studenten. Schüler und Studenten können die App nutzen, um Texte zu verstehen und zu analysieren.

4. Potenzial:

Die App hat das Potenzial, einen großen Nutzen für eine Vielzahl von Menschen zu bieten. Schüler und Studenten können die App nutzen, um ihre Studienleistungen zu verbessern.

5. Zusätzliche Informationen:

5.1. Ihr Fortschritt:

Das Projekt befindet sich aktuell in der Planungsphase.

5.2. Genutzte Technologien:

Die App nutzt folgende Technologien:

- Maschinelles Lernen: Maschinelles Lernen wird verwendet, um die Bedeutung des Textes zu verstehen, die Emotionen im Text zu erkennen und den Stil des Textes zu bestimmen.
- Natural Language Processing: Natural Language Processing wird verwendet, um Texte zu analysieren und zu verstehen.

- Java: Die App wird in Java geschrieben.
- Google Bard API: Die App nutzt die Google Bard API, um die Textanalysen durchzuführen.
- Alternativ OpenAi API oder selbst gehostetes LLM

6. Destination of Done

Bis zum [Datum 6 Wochen ab dem Startdatum] soll die App in der Lage sein, Texte zu analysieren, inhaltlich zusammenzufassen und die Emotionen im Text zu erkennen. Zudem soll ein rudimentäres User Interface (UI) bereitgestellt werden, um die Ergebnisse anzuzeigen. Die Anbindung an die OpenAI API oder eine ähnliche API soll ebenfalls implementiert sein.

7. Fazit

Wir sind davon überzeugt, dass unsere App ein wertvolles Werkzeug für eine Vielzahl von Menschen sein kann.