Bachelorarbeit
Dokumentation

Einkaufszettel Scanner

Semester: Frühlingssemester 2024

Datum: 2024-06-14 12:34:44Z

Projekt Team: Fabian Freitag
Josip Di Benedetto

Projekt Betreuer: Markus Stolze

Departement Informatik
Ostschweizer Fachhochschule OST Rapperswil-Jona

Inhaltsverzeichnis

Teil I

Abstract

Abstract

Das Ziel dieser Bachelorarbeit ist die Entwicklung einer Android-App, die Verbrauchern hilft, Ihre Einkaufserlebnisse durch effiziente Preisvergleiche und die Verfolgung von Preisentwicklungen zu optimieren. Im Kontext der steigenden Inflation und Preisvolatilität steht die Notwendigkeit für einkommensschwache Bevölkerungsschichten, Preise von Produkten aus verschiedenen Geschäften zu vergleichen, im Vordergrund. Dies wird durch die Erstellung einer App ermöglicht, die mittels Optical Character Recognition (OCR) Technologie die Produktnamen und Preise aus Kassenbelegen extrahiert und vergleicht. Wesentliche Aspekte dieser Arbeit umfassen die Evaluation geeigneter Technologien für die automatische und manuelle Erkennung der relevanten Bereiche auf dem Kassenbeleg und die Textzeichenerkennung mittels einer OCR-Bibliothek. Die App soll es den Nutzern ermöglichen, ohne Vorkenntnisse Kassenbelege scannen zu können, damit die App automatisch Preisänderungen ermitteln kann.

Folgende Probleme wurden gelöst: die Erkennung von Textzeichen, relevanten Textbereichen auf Kassenbelegen und die Berechnung von Preisdifferenzen gleichnamiger Produkte aus dem gleichen Laden. Der entwickelte Prototyp ermöglicht es Nutzern, Belege in der App zu erfassen und Preisschwankungen seit dem letzten Einkauf zu sehen. Dabei werden nur selbst erfasste Daten genutzt, ohne Berücksichtigung von Rabattaktionen. Anfangs wurde für die Texterkennung die OCR Library des Google ML Kits verwendet, wobei der Benutzer den relevanten Textbereich zuschneiden musste. Mit einer Regex wurden die Informationen strukturiert ausgelesen. Um die Texterkennung zu verbessern, wurde experimentell festgestellt, dass eine homogene Beleuchtung wichtig ist, um einen effizienten Binarisierungsalgorithmus anzuwenden zu können, der den Text vom Rest des Bildes trennt. Da der Ansatz mit OCR und Regex ein bestimmtes Format der Kassenbelege erfordert, wurde im weiteren Projektverlauf die Google Gemini API verwendet. Gemini führt die Texterkennung und die automatische Bereichserkennung durch, sodass verschiedene Belegstrukturen besser ausgelesen werden können. Der Benutzer muss den relevanten Bereich nicht mehr manuell zuschneiden. Experimentell wurde festgestellt, dass Gemini auch gut mit Bildern umgehen kann, die eine schlechte und ungleichmässige Beleuchtung haben. Es sind somit keine Bildverarbeitungsschritte auf der Benutzerseite erforderlich. Der Nachteil von Gemini ist die erforderliche Internetverbindung und die längere Verarbeitungszeit. In einer Weiterentwicklung könnten die erfassten Daten anonymisiert gesammelt und an alle Benutzer zur Verfügung gestellt werden, um aktuellere Preisänderungen anzuzeigen. Dies eröffnet Möglichkeiten für ein Empfehlungssystem, z.B. ob ein Produkt an einem anderen Standort günstiger ist oder ob ein Rabatt wirklich ein Rabatt ist, oder ob zuvor der Preis erhöht wurde

Die Entwicklung einer App, die Kassenbelege scannen und relevante Informationen daraus extrahieren kann, ist eine grosse Herausforderung, da Kassenbelege sehr unterschiedliche Formate haben und die Daten beliebig strukturiert sein können. Modelle, die ma-

schinelles Lernen verwenden, sind besonders vielversprechend, um mit den verschiedenen Kassenbeleg Formaten umgehen zu können. In der Arbeit wurde eine Grundlage erarbeitet, auf der ein schnelles und robustes System zur Lösung der genannten Probleme entwickelt werden kann.

Teil II

Management Summary

Management Summary

0.1 Ziel der Arbeit

Der Zielgedanke dieser Arbeit ist es, Informationen aus Kassenbelegen auszulesen und mit diesen Daten einen Mehrwert zu generieren. Primär geht es um die Preise der Produkte auf den Kassenbelegen. Wenn diese Daten an einer zentralen Stelle gesammelt werden, kann man sehen, wie sich die Preise von Produkten entwickeln und kann ggf. Anomalien in den Daten entdecken, welche Optionen für interessante Funktionalitäten bieten würden. Der Benutzer soll lediglich mit der Smartphone-Kamera ein Bild des Kassenbelegs erstellen und die App zeigt dem Benutzer an, ob sich die Preise der Produkte aus dem aktuellen Einkauf gegenüber den vorherigen Einkäufen geändert haben. Dabei geht man davon aus, dass es sich um die tatsächlichen Preise handelt und nicht um Rabattaktionen. Es sollen verschiedene Technologien und Lösungsansätze getestet und geprüft werden.

0.2 Ergebnis

Im Verlauf der Arbeit konnte ein Prototyp entwickelt werden, der die grundlegenden Ziele der Arbeit abdeckt. Der Prototyp kann einen Kassenbeleg einscannen, darauf die Produktnamen und die dazugehörigen Preise finden und sie extrahieren. Die gescannten Artikel und Preise werden anschliessend dem Benutzer angezeigt, damit falsch erkannte Textzeichen oder Zahlen manuell korrigiert werden können. Im abschliessenden Schritt wird der Laden ausgewählt, zu dem der gescannte Kassenbeleg gehört. Das ist wichtig, weil die App prüft, ob die eben gescannten Artikel schon einmal in dem gleichen Laden gekauft wurden und ob sich der Preis seit dem letzten Einkauf geändert hat.

Es gibt mehrere Lösungsansätze für die Textzeichenerkennung und die Detektion des relevanten Textbereichs auf dem Kassenbeleg (Produktnamen und Preise). Zu Beginn der Arbeit wurde der Text mittels einer Optical Character Recognition (OCR) Library ausgelesen. Dazu wurde die OCR-Funktion des Google ML-Kits verwendet. Damit kann man aus Bildern den Text auslesen, aber nicht den relevanten Bereich auf dem Beleg detektieren und vom restlichen Text trennen. Der relevante Textbereich wurde dabei vom Benutzer manuell zugeschnitten. Des Weiteren stellte man fest, dass die Beleuchtung des Bildes eine zentrale Rolle bei der Qualität des Scans spielt. Optimal ist eine möglichst gleichmässige Beleuchtung, damit man Text und Hintergrund gut voneinander unterscheiden kann. Dadurch kann man einen Binarisierungsalgorithmus verwenden, der je nach Farbwert die einzelnen Pixel auf dem Bild schwarz oder weiss färbt, sodass der Text vom Bild getrennt wird. Im Verlaufe der Arbeit wurde die Google Gemini API veröffentlicht. Diese kann Bilder entgegennehmen und die Textdaten auf dem Bild auslesen. Man hat sich im späteren Projektverlauf für diesen Lösungsansatz entschieden,

weil man mit Gemini automatisch den relevanten Bereich auf dem Beleg erkennen kann, ohne dass der Benutzer diesen zuschneiden muss. Ausserdem ist auf der Benutzerseite keine Bildverarbeitung wie die Binarisierung notwendig, da Gemini auch mit schlecht beleuchteten Bildern umgehen kann.

0.3 Aufgetretene Probleme

Es sind einige Probleme aufgetreten, die zu Zeitverlust geführt haben. Das grösste Problem war die Bildverarbeitung, welche zu lange dauerte. Nachdem der Benutzer das Bild aufgenommen hat, wurden einige Bildverarbeitungsschritte angewendet, um den Text vom Hintergrund zu lösen. Die Binarisierung eignet sich besonders gut dafür. Die Binarisierung versucht einen Schwellenwert zu finden, der entscheidet, ob ein Pixel schwarz oder weiss eingefärbt sein soll. Dabei gibt es die Möglichkeit, einen Schwellenwert zu ermitteln, der für jedes Pixel auf dem Bild verwendet wird, oder einen Schwellenwert pro Pixel zu berechnen. Falls das Bild ungleichmässig beleuchtet ist, erzielt man mit einem Schwellenwert pro Pixel ein besseres Resultat als mit einem globalen Schwellenwert pro Bild, weil die Beleuchtung von benachbarten Pixeln berücksichtigt und somit ggf. korrigiert wird. Dieser Vorgang dauert aber zu lange und ist nicht geeignet für eine mobile Applikation, weil zu viel Rechenleistung benötigt wird. Der Algorithmus für die Berechnung des globalen Schwellenwertes ist deutlich schneller und besser geeignet für die Verarbeitung auf der Benutzerseite. Allerdings erfordert das eine homogene Beleuchtung des Bildes, was je nach Aufnahme nicht immer der Fall ist. Im späteren Projektverlauf wurde die Google Gemini API veröffentlicht, die auch Bilder entgegennehmen kann. Dabei hat man festgestellt, dass das eine geeignete Lösung ist, um die automatische Erkennung des Scanbereichs und die Textzeichenerkennung durchzuführen. Entscheidend hierbei war, dass Gemini gut mit verschiedenen Beleg-Formaten umgehen kann, was bei der ursprünglichen Implementation mit der Regex nicht der Fall war. Seit Gemini verwendet wird, muss der Benutzer somit nicht mehr den relevanten Bereich manuell auswählen und die Texterkennung funktioniert so auch gut mit ungleichmässig beleuchteten Bildern, wodurch auf der Benutzerseite keine Bildverarbeitung notwendig ist. Daher kann angenommen werden, dass Gemini auch Bildbearbeitungsschritte vornimmt, diese aber auf einem viel stärkeren Computer durchgeführt werden und daher viel schneller durchlaufen.

0.4 Fortsetzung des Projektes

In dieser Arbeit konnte nur ein Prototyp erstellt werden, der die grundlegenden Funktionen implementiert. Entsprechend gibt es viele interessante Weiterentwicklungsmöglichkeiten. Für das Erkennen und Extrahieren der relevanten Textstellen könnte ein eigenes OCR-Modell trainiert werden. Aus zeitlichen Gründen konnte dieser Lösungsansatz nicht umgesetzt werden, aber es wird vermutet, dass durch ein spezifisch trainiertes Modell die Qualität der Scans verbessert werden könnte. Ein eigenes Modell kann auch lokal auf dem Gerät ausgeführt werden und würde somit auch offline funktionieren. Wenn man

ein eigenes ML-Modell einsetzt, um die Textzeichenerkennung und die Erkennung des relevanten Textbereichs zu realisieren, wäre wieder eine Bildverarbeitung auf der Benutzerseite notwendig, um den Text vom Hintergrund des Bildes zu lösen. Da wie bereits erwähnt ein effizienter Algorithmus verwendet werden kann, sollte der Benutzer durch die App angewiesen werden, die Aufnahme zu wiederholen, falls die Beleuchtung zu ungleichmässig ist.
Eine weitere sehr vielversprechende Entwicklungsmöglichkeit wäre es, die gesammelten Daten der Belege anonymisiert anderen Benutzern zur Verfügung zu stellen. So könnte dem Benutzer direkt beim ersten Einkauf der Preisverlauf aller Produkte, die er gekauft hat, angezeigt werden. Ebenfalls könnten so genauere Statistiken über die Preisverläufe bereitgestellt werden.

Teil III

Produkt Dokumentation

Kapitel 1

Einführung

1.1 Aufgabenstellung

Nachfolgend die Aufgabenstellung, wie sie mit dem Betreuer Markus Stolze definiert wurde.

1.1.1 Ausgangslage

In einer Zeit, in der die Wirtschaft durch Inflation und Preisvolatilität gekennzeichnet ist, stehen Verbraucher vor der stetigen Herausforderung, ihre Ausgaben zu überwachen und zu optimieren. Insbesondere bei alltäglichen Einkäufen wie Lebensmitteln oder Haushaltswaren können Preisunterschiede zwischen verschiedenen Geschäften und über die Zeit hinweg signifikant sein. Die Fähigkeit, Preise effektiv zu vergleichen und Preisentwicklungen zu verfolgen, ist daher für viele Verbraucher von grossem Interesse, um Einsparungen zu realisieren und ihr Kaufverhalten anzupassen. Allerdings ist dieser Prozess oft zeitaufwendig und komplex, da er das Sammeln, Vergleichen und Analysieren von Preisen aus verschiedenen Quellen erfordert. Hier bietet die digitale Technologie eine vielversprechende Lösung durch die Entwicklung von Anwendungen, die den Verbrauchern diese Aufgaben abnehmen und den Prozess vereinfachen können.

1.1.2 Ziel der Arbeit

Ziel dieser Bachelorarbeit ist die Entwicklung einer Android-App, die Verbrauchern hilft, ihre Einkäufe effizienter zu gestalten, indem sie Preise für gleiche Produkte auf verschiedenen Kassenbelegen vergleicht und Preisentwicklungen aufzeigt. Hierfür soll zunächst eine umfassende Evaluation geeigneter Technologien durchgeführt werden, einschliesslich der Programmiersprachen, Frameworks und geeigneter Datenbanklösungen, um eine solide Grundlage für die Entwicklung zu schaffen. Ein zentraler Aspekt der Arbeit ist die Gestaltung eines nutzerfreundlichen Interfaces, das es den Nutzern ermöglicht, ohne Vorkenntnisse effektiv mit der App zu interagieren. Besonderes Augenmerk liegt dabei auf der Erarbeitung von Konzepten, wie Nutzer zu einem Erfolgserlebnis geführt werden können, etwa durch intuitive Bedienbarkeit und die unmittelbare Darstellung von Einsparpotentialen durch Preisvergleiche. Die App soll folgende Kernfunktionen umfassen:

- Das Aufnehmen von Bildern von Kassenbelegen
- die Extraktion von Produktnamen und Preisen mittels OCR-Technologie
- den Vergleich von Preisen desselben Produktes auf unterschiedlichen Belegen

- die Anzeige von Preisänderungen bei Verfügbarkeit historischer Daten

Besonderes Augenmerk soll auf die Fähigkeit gelegt werden, Preisvergleiche und Preissteigerungen herauszustellen, um Nutzern einen Mehrwert in der effektiven Überwachung und Anpassung ihres Kaufverhaltens in einem sich ständig verändernden wirtschaftlichen Umfeld zu bieten.

1.1.3 Auftrag

Die folgenden Teilaufgaben sind zu bearbeiten (soweit sinnvoll):

- Analyse: Erhebung und Dokumentation der Anforderungen
  - Beschreibung der Ausgangssituation. Beschreibung des Status Quo aus technischer Sicht und aus Nutzersicht (wer sind Nutzer, was sind deren aktuelle Probleme)
  - Dokumentation der Architecture Constraints
  - Funktionale Anforderungen
  - Nicht-funktionale Anforderungen
- Entwurf: Entwicklung Architektur und Vorschlag UX
  - Analyse sinnvoller Architektur und (soweit sinnvoll) unterstützender Libraries und Frameworks
  - Beschreibung der Nutzer und Rollen
  - Prototyping UX (Paper, Szenario, Durchlauf, ...)
  - Definition eines MVP + Hypothesen zu Ausbauschritten
  - Demonstration des Architekturprototyps und validiertem UX-Vorschlag
- Implementation und Test (mit Referenz auf FA/NFA)
  - Dokumentation des Erreichungsgrads der FA/NFA
  - Beschreibung weiter Limitation
  - Empfehlungen und Beschreibungen zu weiteren Ausbauschritten mit Aufwandschätzung, inklusive von absehbaren Weiterentwicklungen
- Sinnvolle Dokumentation des Projektes und der prototypischen Implementation
  - Notwendige Dokumentation entsprechend «Leitfaden»
  - Developer-Dokumentation
  - Projekt-Dokumentation inkl. Reflexion zur Nutzung von (KI-)Werkzeugen zur Text-Generierung

1.2 Problembeschreibung

Wenn man Lebensmittel einkaufen geht, kauft man häufig die gleichen Artikel. Dabei ändern sich die Preise von Produkten immer wieder, beispielsweise durch Teuerungen oder Rabattaktionen. Bei einigen Artikeln ändert sich der Preis sehr häufig, bei anderen sehr selten. Je nach Einkaufsgewohnheiten kann es eine Herausforderung sein, die Preise von einzelnen Produkten zu verfolgen. Zusätzlich hat der Verkäufer kein Interesse daran, dass Preisänderungen auffallen, wodurch Preisanpassungsstrategien verwendet werden, welche es noch schwieriger machen, Preisänderungen während des Einkaufes wahrzunehmen. Ein weiteres Problem ist, dass man für Garantiefälle den Kassenzettel lange Zeit aufbewahren muss, aber die aufgedruckten Informationen mit der Zeit verblassen. In vielen Fällen hätte man einen Anspruch auf Garantie, aber man kann den Kassenzettel nicht mehr finden oder dieser ist nicht mehr lesbar.

1.3 Vision

Die Idee dieser Arbeit besteht darin, eine mobile Applikation zu entwickeln, welche die Daten eines Kassenbelegs dazu verwendet, um dem Benutzer eine übersichtliche Darstellung seiner Ausgaben anzuzeigen. Der Benutzer sollte primär sehen, welche Produkte teurer oder günstiger geworden sind und welchen Einfluss die Preisänderung einzelner Produkte auf den Gesamtpreis hat. Der Benutzer sollte lediglich mit der Smartphone-Kamera den Kassenbeleg scannen und die App extrahiert daraus die Produktnamen und Preise. Zusätzlich gibt es auf dem Kassenbeleg Barcodes oder QR-Codes, bei denen die Identifikationsnummer des Belegs hinterlegt ist. Diese Identifikationsnummer könnte man ebenfalls digital speichern und den Kassenbeleg archivieren, sodass man beispielsweise in einigen Jahren diese Informationen verwenden kann, um von der Produktgarantie zu profitieren.

1.4 Ziele der Applikation

Das Ziel der Applikation ist es, dem Benutzer einen Überblick über die Veränderung der Preise seiner Produkte zu geben. Der Benutzer soll ein Bild seines Kassenbelegs machen können, aus welchem die Produktnamen und Preise extrahiert werden. Die Produktnamen und die Preise sollen dann in einer tabellarischen Form angezeigt werden. In dieser Ansicht kann der Benutzer die extrahierten Informationen korrigieren, falls sie nicht ganz korrekt erkannt wurden. Die App soll alle gescannten Kassenbelege in Listenform anzeigen. In dieser Liste wird die Preisveränderung einzelner Artikel absolut und / oder relativ angezeigt und farblich markiert (grün = preis gefallen, schwarz = preis unverändert, rot = preis gestiegen).

1.5 Warum sollte jemand eine solche App verwenden?

Die App soll dem Benutzer helfen, Kostenoptimierungen zu identifizieren. Der Benutzer soll auch sehen, wofür Geld ausgegeben wird. Vor allem bei Lebensmitteln besteht grosses Sparpotenzial, weil man häufig nicht gross auf die Preise achtet, sondern Gewohnheitskäufe tätigt. Verkäufer verwenden unter anderem eine sogenannte Stufenpreispolitik. Dabei werden im Laufe der Zeit die Preise in kleinen Schritten erhöht, anstatt dass man grosse, auffällige Änderungen auf einmal durchführt. Wenn man beispielsweise den Preis eines Artikels von 1.50 auf 2.00 erhöht, dann sieht der Verbraucher sofort, dass hier eine Änderung von 0.50 stattgefunden hat. Wenn man den Preis aber zuerst von 1.50 auf 1.75 erhöht und anschliessend von 1.75 auf 2.00, dann sieht der Verbraucher lediglich einen Preisanstieg von 0.25, weil der Preis von 1.75 als neuer Bezugspunkt verwendet wird. In Wahrheit ist der Preis des Artikels um doppelt so viel angestiegen. Ähnlich verhält es sich mit Rabattaktionen, bei denen eine Preisreduktion von beispielsweise 30 % beworben wird, nachdem der Artikel vorher zuerst um beispielsweise 10 % erhöht wurde. Der beworbene Rabatt entspricht somit nicht der Realität. Mit einer App, welche die Preise auf den Kassenbelegen erfasst, kann die realen Preisänderungen im Verlauf der Zeit anzeigen. Der Benutzer sieht somit gleich, wie viel Geld mehr er tatsächlich ausgibt, und kann bessere Entscheidungen treffen bzw. die Gewohnheitskäufe reduzieren.

1.6 Besteht ein Bedarf für eine solche App?

Um herauszufinden, ob es einen allgemeinen Bedarf für eine App gibt, mit der man Kassenbelege scannen kann, wollte man herausfinden, ob und wie stark potenzielle Benutzer auf die Preise beim Einkaufen achten und ob Sie die Kassenbelege aufbewahren und für welchen Zweck. Um der Beantwortung dieser Frage etwas näherzukommen, wurde eine Umfrage durchgeführt, bei der ca. 100 Personen mitgemacht haben. Folgende Fragen wurden gestellt:

- Wie oft haben Sie bei Ihrem letzten Lebensmittel-Einkauf auf die Preise der einzelnen Artikel geachtet?
- Ist Ihnen in letzter Zeit aufgefallen, dass eines der Produkte teurer geworden ist, welches Sie oft kaufen?
- Bewahren Sie die Kassenbelege nach Ihren alltäglichen Einkäufen auf?
- Für welchen Zweck bewahren Sie Kassenbelege auf? Beispielsweise für persönliche Buchhaltung oder ähnliches (Falls letzte Antwort Ja war)?
- Wie alt sind Sie?

Die detaillierten Ergebnisse der Umfrage finden sich im Anhang unter Kapitel 13.6. Die wichtigsten Erkenntnisse sind, dass fast alle Teilnehmer beim Einkaufen immer auf die Preise achten, oder zumindest teilweise bei bestimmten Produkten. Ebenfalls deutlich

ist das Resultat der bei der Frage, ob in letzter Zeit ein bestimmtes Produkt teurer geworden ist, bei dem über 60 % aller Befragten mit Ja geantwortet haben. Das zeigt, dass Leute sich die Preise von Produkten merken, die oft gekauft werden und diese ggf. für alle Produkte sehen möchten, anstatt nur für bestimmte. Sehr interessant war die Frage, ob Kassenbelege aufbewahrt werden. Hier wurde explizit nach den Kassenbelegen von alltäglichen Einkäufen gefragt. Es gaben über 10 % aller Befragten an, diese aufzubewahren. Der Wert wurde vor der Umfrage niedriger geschätzt. Das zeigt, dass es einen Bedarf gibt, die Kassenbelege aufzubewahren. Als häufigste Gründe für die Aufbewahrung der Kassenbelege wurde Folgendes angegeben:

- Persönliche Buchhaltung. Einige geben an, Belege digital zu speichern.
- Als Nachweis, dass man einen Artikel bezahlt hat (Umtausch, Garantie)

Die meisten Antworten beziehen sich auf die persönliche Buchhaltung. Deshalb könnte man die Daten der erfassten Kassenbelege dazu nutzen, um Statistiken zu den eigenen Ausgaben zu erstellen. Eine solche Funktion könnte die Benutzererfahrung steigern, aber diese Arbeit wird sich hauptsächlich auf das Vergleichen der Preise konzentrieren.

1.7 Erwartungen an die App

Aufgrund der Ergebnisse der Umfrage haben sich einige Punkte ergeben, die potenzielle Benutzer von dieser App erwarten könnten.

- Einen Weg, um Kassenbelege zu speichern
- Automatisches Vergleichen der Preise von einzelnen Artikeln
- Überblick über alle Ausgaben
- Unterstützung für Haushaltsabrechnungen
- Speichern von Garantiebelegen für Warenumtausch

Die App soll die ersten beiden Punkte abdecken, um eine solide Basis für eine Weiterentwicklung der App zu schaffen. Die anderen Punkte werden in dieser Arbeit nicht abgedeckt, aber als mögliche Weiterentwicklungsoptionen beschrieben.

1.8 Wer sollte diese App verwenden

Die Umfrage wurde ausschliesslich mit Studenten durchgeführt. Primär wird darauf geachtet, dass die App von Studenten verwendet werden kann. Sie soll aber an alle Personen gerichtet sein, die auf ihre Ausgaben achten wollen und einen einfachen Weg suchen, sich einen Überblick über Preisentwicklungen alltäglicher Produkte zu machen.

1.9 Legitimation für eine Bachelorarbeit

Der Artikel "ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction" [1] beschreibt eine Wettbewerbsveranstaltung im Rahmen der International Conference on Document Analysis and Recognition (ICDAR) 2019. Die Veranstaltung umfasste drei Hauptaufgaben, die die Teilnehmer herausfordern sollten, fortschrittliche Methoden zur Texterkennung und Informationsextraktion von gescannten Kassenbelegen zu entwickeln und zu evaluieren.

Wettbewerbsaufgaben:
Die Aufgaben konzentrieren sich auf die Verarbeitung und Analyse gescannter Belege. In der ersten Aufgabe geht es um eine Lokalisierung von Texten in Belegen. Die zweite Aufgabe forderte die Erkennung von Texten ohne vorherige Lokalisierungsinformationen. Die dritte Aufgabe zielte darauf ab, Schlüsselinformationen wie Datum und Gesamtsumme aus den Belegen zu extrahieren und in JSON-Dateien zu speichern. Zusammengefasst umfassten die Aufgaben die Lokalisierung, Erkennung und Extraktion von Textinformationen aus gescannten Quittungen.

Ergebnisse und Methoden:
Aufgabe 1: Die Teams mit den besten Resultaten nutzten Ansätze wie die Verwendung von Mask-RCNN (Region-Based Convolutional Neural Networks) und FishNet-Backbones zur Textlokalisierung. Die erstplatzierte Methode kombinierte mehrere Modelle und entfernte iterativ redundante Informationen.
Aufgabe 2: Die Top-Teams setzten Methoden wie CRNN (Convolutional Recurrent Neural Network) ein, um die Texterkennung zu optimieren. Die erstplatzierte Methode kombinierte verschiedene Modelle und Bildverarbeitungs-Techniken, um die Erkennungsgenauigkeit zu maximieren.
Aufgabe 3: Die besten Ansätze zur Informationsextraktion nutzten Deep Learning und NLP-Techniken (Natural Language Processing), um die relevanten Informationen präzise zu identifizieren und zu kategorisieren. Die Gewinnerstrategie kombinierte verschiedene Modelle zur Verbesserung der Extraktionsgenauigkeit.

Schlussfolgerung:
Der Artikel verdeutlicht die Herausforderungen, die mit der Extraktion von Informationen aus Kassenbelegen verbunden sind. Die drei definierten Aufgaben – Textlokalisierung, Texterkennung und Extraktion von Schlüsselinformationen – illustrieren die Komplexität und die Probleme, die bei der Verarbeitung solcher Dokumente auftreten können.
Die Lokalisierung von Texten auf gescannten Belegen erfordert präzise Algorithmen, um Texte korrekt zu identifizieren und von Hintergrundrauschen zu unterscheiden. Dies wird durch die Vielzahl unterschiedlicher Layouts und Schriftarten, die auf Kassenbelegen verwendet werden, noch erschwert.
Die dritte Aufgabe, die Extraktion von Schlüsselinformationen, zeigt besonders deutlich

die Komplexität dieser Aufgabe. Hier müssen Algorithmen nicht nur in der Lage sein, Texte zu erkennen, sondern auch deren Bedeutung und Kontext korrekt zu interpretieren, um relevante Informationen wie Datum, Gesamtbetrag und spezifische Artikel korrekt zu extrahieren. Diese Aufgaben erfordern fortschrittliche Techniken aus den Bereichen maschinelles Lernen und natürliche Sprachverarbeitung.
Angesichts dieser Herausforderungen ist das Thema der Informationsextraktion aus Kassenbelegen ein relevantes Thema für eine Bachelorarbeit. Es bietet die Möglichkeit, sich mit aktuellen und praxisrelevanten Problemstellungen zu beschäftigen und innovative Lösungsansätze zu entwickeln, die in der Praxis breite Anwendung finden können. Die Arbeit an einem solchen Thema ermöglicht es, Kenntnisse in verschiedenen Technologiebereichen zu vertiefen und gleichzeitig einen wertvollen Beitrag zur Weiterentwicklung der Texterkennungs- und Informationsextraktionstechnologien zu leisten.

Kapitel 2

Stand der Technik

2.1 Vergleich von ähnlichen Apps

Zu Beginn der Arbeit wurde versucht, andere Apps zu finden, welche die gleichen Ziele verfolgen, wie diese Arbeit. Es wurde keine App gefunden, die die gleichen Funktionen hat. Es wurden aber Apps gefunden, wie zum Beispiel OneSplit [26], die Daten aus einem Kassenbeleg erfassen können. Das Ziel von OneSplit ist allerdings, dass die entstandenen Kosten in einer Gruppe aufgeteilt werden können und nicht, dass eine Preisänderung angezeigt wird. Interessanterweise funktioniert OneSplit sehr gut, was das Auslesen strukturierter Daten von Kassenbelegen angeht. Die Verarbeitung ist sehr schnell und funktioniert auch offline und mit verschiedenen Kassenbeleg-Formaten. Jedoch treten auch hier Fehler auf, wenn bspw. die Qualität des Kassenbelegs schlecht ist oder wenn im Produktnamen ein Zeilenumbruch vorkommt.

2.2 Datenextrahierung via ChatGPT, Gemini und Claude

Im Verlaufe der Arbeit wurde ermittelt, welche der gängigen AI Modelle sich für die Textextrahierung eignen würden. In diesem Kapitel werden die Erkenntnisse dieser Tests beschrieben. Es wurde jeweils das gleiche Bild 2.1 und der gleiche Prompt verwendet.

> The attached image is a picture of a Receipt. Extract the items on this Receipt. For every item, get the productname, the amount and the price per piece. Return only a JSON in the format `[{“Productname”: “string”, “Amount”: “int”, “Price”: “float”}]`

Das Ziel war es, dass das jeweilige AI Modell aus dem Bild die Produktnamen, Mengen und Stückpreise extrahiert und diese Daten als JSON zurückgibt. Von allen getesteten Modellen werden jeweils einzelne, ausgewählte Einträge aus der Antwort verglichen, um eine Aussage über die Qualität der Antwort zu machen. Alle drei getesteten Modelle konnten wie gewünscht ein JSON mit den benötigten Daten erzeugen. Alle getesteten Modelle bieten eine API an, welche eventuell für die Arbeit verwendet werden könnte, allerdings bietet das Free Tier von Gemini am wenigsten Limitationen, wodurch es sich für die Entwicklung eines Prototyps am besten eignet. [27] [14] [2].

Abbildung 2.1: Bild eines Kassenbeleges für den Test mit AI Modellen

2.2.1 ChatGPT 4o

Die Qualität der erkannten Daten ist generell gut. Das Model scheint zu verstehen, was ein Produktname, eine Mengenangabe oder ein Preis ist. Die erkannten Produkte sind nicht fehlerfrei. Beispielsweise wurde beim dritten Produkt “Valflora Milch hoch P.” die Menge fälschlicherweise als “1” statt “2” erkannt. Ebenfalls wurde anstelle des Stückpreises der Gesamtpreis verwendet.

```
[
    {"Productname": "MBud Blutorangensaft", "Amount": 2, "Price": 3.30},
    {"Productname": "MBud Orangensaft", "Amount": 2, "Price": 2.60},
    {"Productname": "Valflora Milch hoch P.", "Amount": 1, "Price":
    2.70},
]
```

Listing 2.1: JSON von ChatGPT 4o generiert

2.2.2 Gemini

Die Qualität der erkannten Daten ist generell gut. Auch Gemini scheint zu verstehen, welcher Text zu welcher Kategorie gehört, allerdings sind auch diesem Modell Fehler

unterlaufen. Für das erste Produkt wurde “KBud” anstelle von “MBud” erfasst. Beim zweiten Produkt ist die Menge falsch und beim dritten Text wurde anstelle eines “P.” ein “F.” erkannt und ein falscher Preis verwendet.

```
[
    {"Productname": "KBud Blutorangensaft", "Amount": "2", "Price": 3.3},
    {"Productname": "MBud Orangensaft", "Amount": "1", "Price": 2.6},
    {"Productname": "Valflora Milch hoch F. 2", "Amount": "2", "Price":
    1.1},
]
```

Listing 2.2: Antwort von Gemini generiert

2.2.3 Claude

Auch bei Claude haben sich vergleichbare Fehler wie bei den anderen beiden Modellen eingeschlichen. Zum Beispiel wurde für das erste Produkt der falsche Preis erkannt.

```
[
    {"Productname": "MBud Blutorangensaft", "Amount": 2, "Price": 1.3},
    {"Productname": "MBud Orangensaft", "Amount": 2, "Price": 2.6},
    {"Productname": "Valflora Milch hoch P.", "Amount": 2, "Price":
    1.35},
]
```

Listing 2.3: JSON von Claude generiert

2.2.4 Schlussfolgerung

Alle drei getesteten Modelle sind in der Lage, aus einem Bild eines Kassenbeleges die relevanten Informationen zu extrahieren und in einer brauchbaren Form zu übergeben. Die Ergebnisse aller Modelle haben eine sehr vergleichbare Qualität und machen in etwa die gleichen Fehler. Da sich keines der Modelle deutlich von den anderen abhebt, aber das Free Tier von Gemini am wenigsten Limitation hat, wird für diese Arbeit Gemini verwendet.

Kapitel 3

Verwendete Tools

In den folgenden Abschnitten wird erläutert, welcher Tools für diese Arbeit eingesetzt wurden und welche Vorteile und Nachteile zur Entscheidung geführt haben.

3.1 Visual Studio Code

Für die Entwicklung der App wird die Visual Studio Code (VSCode) verwendet. Die Flutter Erweiterung [19] und das Dart Plugin [30] bieten eine optimale Unterstützung für die Entwicklung von Flutter Apps. Diese erlauben es, einfach zwischen verschiedenen verbundenen Geräten auszuwählen und die App direkt auf diesen zu testen und debuggen. Zu Beginn der Arbeit wurde mit Android Studio gearbeitet, allerdings hat sich gezeigt, dass VSCode leichtgewichtiger ist. Flutter selbst empfiehlt VSCode.

3.2 Android Studio

Die Android Studio IDE [12] wurde nicht primär zum Programmieren der App verwendet. Sie bietet aber die Möglichkeit, mit dem Device Explorer auf das Dateisystem der angeschlossenen Geräte zuzugreifen, um beispielsweise die Datenbank auszulesen, was sich als sehr hilfreich erwies. Dies erlaubte es, mit einer Datenbank IDE schnell Abfragen zu formulieren und zu testen.

3.3 Programmiersprache/Framework

Zu Beginn der Arbeit wurde evaluiert, welche Programmiersprachen und Frameworks für die Applikation infrage kommen, und was schlussendlich verwendet werden soll.

3.3.1 Flutter

Flutter ist ein von Google entwickeltes UI-Toolkit zur Erstellung native kompilierter Anwendungen für Mobiltelefone, Web und Desktop aus einer einzigen Codebasis [13]. Es verwendet die Programmiersprache Dart und bietet ein Hot-Reload und Hot-Restart Feature, um Änderungen im Code direkt mit dem Emulator oder dem angeschlossenen Android-Gerät zu synchronisieren, ohne die App neu starten zu müssen. Flutter wird für seine schnelle Leistung, das ansprechende Widget-Set und die flexible UI-Gestaltung gelobt.

Vorteile:

- Cross-Plattform-Entwicklung: Ermöglicht die Entwicklung von iOS- und Android-Apps mit einer einzigen Codebase, was Zeit und Ressourcen spart.
- Reichhaltige UI-Komponenten: Bietet eine umfangreiche Bibliothek von vordefinierten Widgets, was die Entwicklung einer ansprechenden und intuitiven Benutzeroberfläche erleichtert.
- Leistung: Nahezu native Performance durch direkte Kompilierung in Maschinencode.

Nachteile:

- Lernkurve: Dart ist weniger verbreitet als andere Programmiersprachen, was die Einstiegshürde erhöhen kann.
- Plattformspezifische Features: Der Zugriff auf einige spezifische Funktionen der Plattform kann zusätzliche Arbeit oder die Verwendung von Drittanbieter Paketen erfordern.

Aufgrund der Interessen des Projektteams und der Hilfestellungen um die App auch auf iOS zu portieren, hat man sich dazu entschieden, Flutter zu verwenden.

3.3.2 Weitere Optionen

MAUI

.NET Multi-platform App UI (MAUI) ist ein Framework von Microsoft zur Erstellung von Cross-Plattform-Anwendungen für Android, iOS, macOS und Windows. Es baut auf dem Erfolg von Xamarin.Forms auf und integriert sich nahtlos in das .NET-Ökosystem, was Entwicklern ermöglicht, die Sprachen C# und XAML zu verwenden.

Vorteile:

- Integration in .NET: Bietet eine starke Unterstützung für .NET-Features und -Ökosystem, was insbesondere für Teams, die bereits mit C# arbeiten, von Vorteil ist.
- Single Project Experience: Vereinfacht den Entwicklungsprozess durch die Verwendung eines einzigen Projekts für alle Plattformen.
- Unterstützung für plattformübergreifende und plattformspezifische APIs: Ermöglicht die Nutzung gemeinsamer Funktionen sowie Zugriff auf plattformspezifische APIs und UIs.

Nachteile:

- Jüngeres Framework: Weniger reife und umfangreiche Widget- oder Komponentenbibliotheken im Vergleich zu etablierten Frameworks.
- Performance: Kann in einigen Fällen hinter rein nativen Apps zurückbleiben, insbesondere bei intensiver Nutzung plattformspezifischer Funktionen.

Kotlin

Kotlin Multiplatform ist ein Teil des Kotlin-Ökosystems, das von JetBrains entwickelt wurde. Es ermöglicht Entwicklern, den plattformübergreifenden Teil ihres Codes in Kotlin zu schreiben, während für die UI und plattformspezifische Funktionen der native Code verwendet wird. Dieser Ansatz zielt darauf ab, die Vorteile der Wiederverwendbarkeit von Code mit der Flexibilität und Leistung nativer Entwicklung zu kombinieren.

Vorteile:

- Code-Wiederverwendung: Teile des Codes, insbesondere Business Logik und Datenzugriff, können über Plattformen hinweg wiederverwendet werden, während das UI nativ bleibt.
- Native Performance und Flexibilität: Die UI-spezifischen und plattformspezifischen Funktionen werden nativ implementiert, was eine hohe Leistung und Zugang zu allen nativen APIs ermöglicht.
- Integration in bestehende Apps: Eignet sich gut zur schrittweisen Integration in bestehende Projekte.

Nachteile:

- Komplexere Architektur: Die Notwendigkeit, UI und einen Teil der Funktionalität plattformspezifisch zu implementieren, kann den Entwicklungsprozess komplizieren.
- Weniger Automatisierung bei der UI-Gestaltung: Im Vergleich zu vollständig integrierten Cross-Platform-Frameworks müssen Entwickler mehr Zeit in die plattformspezifische UI-Gestaltung investieren.

React Native

React Native ist ein von Facebook entwickeltes Framework zur Erstellung von nativen Apps für mehrere Plattformen wie iOS und Android unter Verwendung von JavaScript und React. Es ermöglicht Entwicklern, Apps zu entwickeln, die nahe an der Leistung und der Benutzererfahrung nativer Anwendungen sind, indem sie den grössten Teil ihres Codes über Plattformen hinweg wiederverwenden.

Vorteile:

- **Code-Wiederverwendung:** Ermöglicht es Entwicklern, einen Grossteil des Codes über iOS und Android hinweg zu verwenden, was die Entwicklungszeit und -kosten erheblich reduzieren kann.
- **Native Komponenten:** Verwendet native Komponenten unter der Haube, was zu einer hohen Performance und einem nativen Benutzererlebnis führt.
- **Grosse Entwicklergemeinschaft:** Profitiert von einer grossen und aktiven Entwicklergemeinschaft, reichhaltigen Bibliotheken und Plugins, die die Implementierung von Funktionalitäten erleichtern.

Nachteile:

- **Performance-Einbussen:** Trotz der Verwendung nativer Komponenten kann React Native in Anwendungen mit hohen Leistungsanforderungen immer noch hinter rein nativen Apps zurückbleiben.
- **Abhängigkeit von Drittanbieter-Bibliotheken:** Die Notwendigkeit, auf Drittanbieter-Bibliotheken für viele erweiterte Funktionen angewiesen zu sein, kann zu Inkompatibilitäten und Wartungsproblemen führen.
- **Komplexität bei der Navigation:** Die Handhabung der Navigation kann komplex sein, besonders wenn sie fortgeschrittene und tief integrierte Navigationsmuster erfordert.

3.4 Daten Speicherung

Die Daten, welche aus den Scans der Kassenbelege extrahiert werden, müssen gespeichert werden, damit sie später für die Vergleiche verwendet werden können. Android bietet mehrere Möglichkeiten, wie und wo die Daten gespeichert werden können. Nachfolgend werden diese Möglichkeiten kurz beschrieben [11].

	Art der Daten	**Zugriffsmethode**	**Benötigte Berechtigung**	**Zugriff für andere Apps möglich**	**Bei Deinstallation der App entfernt**
App spezifische Dateien	Solche, die nur von dieser App verwendet werden sollen	getFilesDir() bzw. getExternalFilesDir()	Keine bei API Level 19+	Nein	Ja
Media Storage	Bilder, Audio und Video	MediaStore API	READ WRITE_ EXTERNAL_ STORAGE	Ja	Nein
Dokumente	Shareable content	Storage Access Framework	Kein	Ja	Nein
App preferences	Key-Value pairs	Jetpack Preferences library	Kein	Nein	Ja
Datenbank	Strukturierte Daten	Datenbank spezifisch	Keine	Nein	Ja

Tabelle 3.1: Daten Speichermethoden

Aufgrund der Anforderungen fallen mehrere dieser Methoden direkt weg: Die Daten sollen nicht von anderen Apps verändert werden, demnach fallen die Dokumente weg. Die Daten sollen nicht als Bilder gespeichert werden, was den Media Storage ausschliesst. App preferences erlauben nur Key-Value pairs, was auf die zu speichernden Daten nicht zutrifft. Für den effizienten Vergleich von Preisen würde sich am ehesten eine Datenbank eignen.

3.4.1 Datenbank

Wenn es um Datenbanken geht, gibt es grob zwei Kategorien: Relationale Datenbanken (SQL) und Nicht-Relationale Datenbanken (noSQL). Relationale Datenbanken sind im grossen und ganzen Tabellen, welche miteinander in einer Beziehung stehen können. In Nicht-Relationalen Datenbanken werden die Daten nicht in Tabellen gespeichert, sondern in strukturierten Dokumenten. Aufgrund der Erfahrungen des Projektteams wird eine Relationale Datenbank verwendet.

SQFlite

SQFlite ist eine direkte Implementierung von SQLite für Flutter, die es Entwicklern ermöglicht, mit der SQLite-Datenbank durch direkte SQL-Abfragen zu interagieren [29]. Diese Bibliothek bietet volle Kontrolle über die Datenbankoperationen, was sie besonders für Anwendungen geeignet macht, die komplexe Datenabfragen oder eine optimierte Datenverarbeitung benötigen.

Vorteile:

- Volle Kontrolle über die Datenbank und SQL-Operationen, was eine hohe Flexibilität und Optimierungsmöglichkeiten bietet.
- Leichtgewichtig und ohne grossen Overhead, da es direkt auf SQLite aufbaut.

Nachteile:

- Höherer manueller Aufwand bei der Verwaltung von Datenbankschemas und Abfragen, was zu mehr Code und potenziellen Fehlern führen kann.
- Fehlen höherer Abstraktionen und Hilfsfunktionen, die in anderen Bibliotheken verfügbar sind, was die Entwicklungsgeschwindigkeit reduzieren kann.

Drift

Drift ist eine reaktive Persistenzbibliothek für Flutter und Dart, die auf SQLite basiert und eine breite Palette von Funktionen bietet, einschliesslich automatischer Updates der Benutzeroberfläche basierend auf Datenbankänderungen [4]. Drift nutzt Darts Streams für reaktive Programmiermuster und ermöglicht es Entwicklern, reichhaltige, interaktive Anwendungen zu erstellen. Die Bibliothek unterstützt komplexe SQL-Abfragen, benutzerdefinierte Typkonvertierungen und automatische Codegenerierung.

Vorteile:

- Bietet ein reaktives Programmiermodell, das automatische UI-Updates auf Datenänderungen ermöglicht.
- Unterstützt komplexe SQL-Abfragen und bietet eine hohe Flexibilität bei der Datenmodellierung und -abfrage.
- Einfache Integration und Nutzung durch automatische Codegenerierung.

Nachteile:

- Die reaktive Natur und der Abstraktionsgrad können für einfache Anwendungen überdimensioniert sein.
- Die Abhängigkeit von der Codegenerierung kann den Build-Prozess etwas verlangsamen, besonders in grossen Projekten.

Floor

Floor ist eine Flutter-Bibliothek, die auf der SQLite-Datenbank aufbaut und vom Android Room-Design inspiriert ist [28]. Sie bietet eine höhere Abstraktionsebene durch Annotationen und automatische Codegenerierung, was die Entwicklung beschleunigt und vereinfacht. Die Integration in Flutter-Projekte ist nahtlos, und sie unterstützt typisierte Abfragen, was die Wahrscheinlichkeit von Fehlern reduziert. Floor unterstützt auch Beziehungen zwischen Entitäten, was die Modellierung komplexer Datenstrukturen erleichtert.

Vorteile:

- Einfache Einrichtung und Benutzung durch Annotationen und Codegenerierung.
- Typsichere Abfragen minimieren Fehler bei Datenbankoperationen.
- Inspiriert von Androids Room, was Android-Entwicklern den Einstieg erleichtert.

Nachteile:

- Weniger flexibel bei sehr spezifischen oder komplexen SQL-Operationen im Vergleich zu direkteren SQLite-Zugriffsmethoden.
- Abhängigkeit von der Codegenerierung kann den Build-Prozess verlangsamen.

Da in früheren Projekten bereits Erfahrungen mit der Room-Datenbank gesammelt wurden und die SQL-Operationen nicht komplex werden sollten, wurde entschieden, trotz der möglichen Verlangsamung des Build-Prozesses die Floor-Bibliothek zu verwenden.

3.5 Github

Der Code dieser Arbeit wird in einem GitHub Repository abgelegt. Dieses Repository wird nicht öffentlich zugänglich sein, es handelt sich um ein closed-source Projekt.

3.6 Flutter Test

Flutter Test ist ein umfassendes Testframework, mit dem sichergestellt wird, dass die App auch nach Änderungen immer noch wie erwartet funktioniert.

3.7 KI-Tools

Die folgenden KI-Tools wurden in dieser Arbeit für verschiedene Aufgaben verwendet.

3.7.1 ChatGPT

ChatGPT wurde verwendet, um für verschiedene Punkte einen ersten Anhaltspunkt zu erhalten. Für die Dokumentation hat ChatGPT jeweils einige Punkte generiert, die in einem Kapitel behandelt werden könnten. Anhand dieser Punkte wurden die Kapitel erstellt und erweitert. Ebenfalls wurde ChatGPT für Verbesserungsvorschläge im geschriebenen Text verwendet. Für den Code wurde ChatGPT verwendet, um sich nach Best Practices für die Entwicklung von Flutter Apps zu erkundigen, wie bspw. Namenskonventionen. Auch bei Fehlermeldungen und Exceptions wurde ChatGPT um Hilfe gebeten.

3.7.2 Gemini

Gemini wurde gleich verwendet wie ChatGPT. Ausserdem wurde die API von Gemini im Verlaufe des Projektes für die Bildverarbeitung verwendet.

3.8 Doku - Latex mit Overleaf

Die Dokumentation wurde in Latex geschrieben. Dafür wurde der Overleaf online Latex-Editor verwendet. In der Vergangenheit hat das Projektteam die Dokumentation lokal in Visual Studio Code bearbeitet und in einem GitLab Repository synchronisiert. Dabei mussten mehrfach merge Konflikte gelöst werden, wenn mehrere Mitglieder in einem File schreiben mussten. Overleaf synchronisiert Änderungen fast in Echtzeit und kann so Merge Konflikte verhindern.

3.9 Verwendete Bibliotheken

Folgende Bibliotheken wurden für diese Arbeit verwendet. Die relevanten Lizenztexte sind im Anhang 13.4 als ganzes zu finden.

Library	**Beschreibung**	**Lizenz**
Google ML-Kit	Optical Character Recognition [10]	MIT
Camera Plugin	Verwendung von Kamerafunktion auf diversen Plattformen [8]	BSD-3-Clause
Permission Handler	Plattformübergreifende API für App Berechtigungen [3]	MIT
Floor	SQLite-Abstraktion für Flutter-Anwendungen [28]	Apache-2.0
Bloc	Umsetzung des BLoC (Business Logic Component) Designpatterns [5]	MIT
Get it	Service Locator, um Objekte und Dienste in einer Anwendung zu registrieren und darauf zuzugreifen [6]	MIT
Image	Laden, Manipulieren und Speichern von Bildern [25]	MIT
Google Generative AI	Gemini Implementation [15]	Apache-2.0
Dotenv	Verwendung von .env File für Environment Variablen [9]	MIT

Tabelle 3.2: Verwendete Bibliotheken und ihre Lizenzen

Kapitel 4

Resultate

4.1 Zielerreichung

4.1.1 Requirements

Im folgenden Abschnitt wird gezeigt, welche Funktionale- und Nicht-Funktionalen Anforderungen erreicht wurden und welche nicht erreicht wurden. Eine genauere Beschreibung der Requirements kann im Kapitel Funktionale Anforderungen 5.4 gefunden werden. Aus zeitlichen Gründen wurden die Anforderungen, welche mit mittlerer und tiefer Priorität eingestuft wurden, nicht bearbeitet. Diese könnten in einem weiterführenden Projekt behandelt werden.

Funktionale Muss-Requirements

ID	Name	Status	Kommentar
FA-1	Kamerascan	i.O.	-
FA-2	Scanbereich markieren	N/A	Wegen Gemini nicht mehr benötigt
FA-3	Daten aus Scan werden angezeigt	i.O.	-
FA-4	Manuelle Korrektur	i.O.	-
FA-5	Gescannte Daten speichern	i.O.	-
FA-6	Manuelle Löschung	n.i.O.	Wurde aus Zeitgründen nicht bearbeitet.
FA-7	Alte Belege anschauen	i.O.	-
FA-8	Sortieren	n.i.O.	Wurde aus Zeitgründen nicht bearbeitet.
FA-9	Preisänderungen innerhalb eines Verzeichnisses	i.O.	-
FA-10	Vordefinierte Verzeichnisse	i.O.	-
FA-11	Warnung bei manueller Löschung	n.i.O.	Nicht erreicht, da FA-6 nicht erreicht wurde.

Tabelle 4.1: Erreichung der Funktionalen Anforderungen

Nicht-Funktionale Muss-Requirements

ID	Name	Status	Kommentar
NFA-1	Android Abdeckung	i.O.	App wurde auf allen definierten Geräten getestet. Texterkennung konnte auf HTC Gerät nicht getestet werden, da VPN nicht funktioniert hat.
NFA-2	Deinstallation	N/A	Bei der Deinstallation werden alle Daten der App gelöscht.
NFA-3	Intuitives UI	i.O.	Wurde mittels End-User-Tests überprüft.
NFA-4	Vertraulichkeit der Daten	i.O.	Sichergestellt, indem die Datenbank in den App internen Dateien gespeichert ist.
NFA-5	Code Qualität	i.O.	Wurde mittels Flutter Analyze sichergestellt.

Tabelle 4.2: Erreichung der Nicht Funktionalen Anforderungen

4.2 Prototyp

Im Verlauf der Arbeit konnte ein Prototyp erstellt werden, der die grundlegenden Funktionen der Applikation gemäss dem definierten MVP 5.6 erfüllt. Spezifisch kann der Prototyp die folgenden Funktionen ausführen:

- Kassenbeleg scannen
- Produktnamen und Preise werden extrahiert
- Es wird eine Liste der erkannten Produkte und Preise angezeigt
- Die Listeneinträge können bearbeitet werden
- Der gescannte Kassenbeleg wird in einer lokalen Datenbank gespeichert
- Auf alle gespeicherten Belege kann aus der App her zugegriffen werden

Der Prototyp implementiert alle funktionalen Anforderungen ausser den FA-6, FA-8 und FA-11. Das FA-2 wurde erreicht, allerdings durch die Verwendung von Google Gemini abgelöst. Die FA-6 und FA-8 wurden nicht erreicht, da sie aus zeitlichen Gründen nicht bearbeitet wurden. Das FA-11 wurde nicht erreicht, da es das FA-6 voraussetzt.

Teil IV

Projekt Dokumentation

Kapitel 5

Anforderungen

5.1 Design Constraints

Aus der Aufgabenstellung 1.1 ergeben sich folgende Einschränkungen.

5.1.1 Technologische Einschränkungen

Die Zielplattform ist Android. Für die Entwicklung einer Android-App gibt es mehrere Möglichkeiten, siehe Kapitel 3.3. Unter Berücksichtigung, dass die App nach der Bachelorarbeit für iOS erweitert werden kann, hat man sich für ein plattformübergreifendes Framework entschieden. Ausschlaggebend für die Wahl eines passenden Frameworks sind eine umfassende Dokumentation, einfacher Zugriff auf native Funktionen (Kamera), einfache Integration einer OCR-Library und niedrige Komplexität, da das Projekt zeitlich beschränkt ist. Für diese Eigenschaften kommen mehrere Frameworks infrage. Man hat sich für Flutter entschieden, da Google als Entwickler sehr viel Unterstützung mit zusätzlichen Libraries und einer guten Dokumentation bietet.

5.1.2 Zeitliche Einschränkungen

Die Dauer dieser Arbeit ist auf wenige Monate beschränkt, spezifisch vom 19.02.2024 bis zum 14.06.2024. Es ist wichtig, dass am Ende die Kernfunktionen implementiert sind. Daher ist es nicht möglich, komplexe Funktionen zu implementieren, welche viel Zeit in Anspruch nehmen. Solche Funktionen könnten beispielsweise das automatische Erkennen der wichtigen Informationen auf einem Kassenbeleg durch ein eigenes ML-Model sein, oder das Analysieren von Benutzerdaten, um genauere Daten zu Artikelpreisen anzeigen zu können.

5.1.3 Scannen von Kassenbelegen

Das Scannen von Kassenbelegen erfordert ein Android-Gerät mit einer Kamera, welches mindestens das Android API-Level 21 unterstützt. Diese Einschränkung ergibt sich daraus, dass die Flutter Camera Library auf Funktionen der CameraX Library von Android Jetpack zurückgreift, welches mindestens das Android API-Level 21 voraussetzt [18].

5.2 Erfolgserlebnis

Die folgenden Szenarien führen beim Benutzer zu einem Erfolgserlebnis

5.2.1 Benutzer stellt fest, dass die Einkäufe billiger/teurer werden

Damit der Benutzer sieht, ob seine Einkäufe teurer werden, braucht es Daten. Wenn die App zum ersten Mal gestartet wird, kann der Benutzer noch kein Erfolgserlebnis haben. Die App muss die Preise auf dem Kassenbeleg mit Artikeln auf einem älteren Beleg vergleichen. Somit muss man mindestens zwei Belege einscannen, um ein Ergebnis zu haben. Sollte eine Benutzerbasis vorhanden sein, kann man die erfassten Daten von diesen Benutzern verwenden, um bereits beim ersten Scan eines Belegs sehen zu können, wie sich die Preise in der Vergangenheit entwickelt haben. So hätte man direkt bei der ersten Verwendung der App ein Erfolgserlebnis. Das ist lediglich ein mögliches Feature, welches man in Zukunft implementieren könnte und nicht für diese Bachelorarbeit vorgesehen ist. Damit der Benutzer von Beginn an versteht, wie die App funktioniert, kann man bei der ersten Verwendung Demodaten verwenden, welche angeschaut werden können. Diese Demodaten sollen den Benutzer dazu animieren, die App zu nutzen.

5.2.2 Benutzer sieht, wo grösseres Sparpotenzial vorhanden ist

Es wurde eine Umfrage durchgeführt (Details unter Kapitel 13.6), um ein besseres Gefühl zu bekommen, was einem Benutzer beim Einkaufen wichtig sein könnte. Es stellte sich heraus, dass bei einem signifikanten Anteil der Befragten ein Interesse besteht, nachverfolgen zu können, wofür Geld ausgegeben wurde. Man kann bereits nach dem ersten Einscannen eines Kassenbelegs eine Ausgaben-Statistik anzeigen, beispielsweise auf der Startseite, sodass der Benutzer immer sieht, wo genau wie viel Geld ausgegeben wurde. Je mehr Daten verfügbar sind, desto genauere Statistiken kann man zeigen. Der Benutzer sieht so selbst, welche Einkäufe am teuersten sind und kann an der entsprechenden Stelle optimieren. Die persönliche Buchhaltung, welche von den Befragten oft erwähnt wurde, wäre beim ersten Scan bereits abgeschlossen.

5.2.3 Punktesystem

Die Daten, welche die Benutzer mit der App erfassen, können verwendet werden, um weitere Dienste anzubieten, beispielsweise Analysen von Rabattangeboten (tatsächlicher Rabatt). Diese Daten sind auch deshalb wertvoll, weil man dem Benutzer ganz detaillierte Preisdaten anzeigen kann, die andere Benutzer erfasst haben. Ausserdem kann man ein Machine-Learning-Modell trainieren, welches lernt, automatisch die wichtigsten Informationen aus einem Kassenbeleg zu extrahieren, ohne dass der Benutzer manuell den wichtigen Bereich markieren muss. Dazu sind ebenfalls grosse Datenmengen von Benutzern notwendig. Allgemein besteht ein Interesse daran, anonymisierte Benutzerdaten zu sammeln und diese auszuwerten. Dazu ist es wichtig, dass der Benutzer ein Vorteil hat, Kassenbelege zu scannen. Neben den persönlichen Vorteilen, welche die App bringen soll, kann ein System eingeführt werden, bei dem Benutzer für das Erfassen der Belege Punkte erhalten. Diese könnten in Gutscheine oder etwas Ähnliches umgewandelt werden. Auch das sprengt den Rahmen dieser Bachelorarbeit und ist lediglich ein Hinweis, wie man das Benutzererlebnis und somit das Erfolgserlebnis bei der Verwendung der

App fördern kann.

5.2.4 Big Data

Dem Benutzer werden Preisänderungen anhand seiner eigenen Einkäufe angezeigt. Wenn der Benutzer die App zu ersten Mal installiert und einen Kassenbeleg einscannt, kann ihm daher kein Preisverlauf angezeigt werden. Wenn genügend Daten von anderen Benutzern vorhanden sind, können die Einkäufe mit den erfassten Kassenbelegen anderer Benutzer abgeglichen werden, wodurch der Benutzer direkt beim ersten gescannten Kassenbeleg den Preisverlauf seiner Produkte sieht und ihm direkt ein Erfolgserlebnis geboten werden kann. Wenn genügend Daten vorhanden sind, könnte dem Benutzer auch angezeigt werden, ob er seinen Einkauf in einem anderen Laden günstiger haben kann, und ob es sich lohnt, mit dem ÖV oder dem Auto zu einem weiter entfernten, aber günstigeren Laden zu gehen.

5.3 User Stories

Titel: Kassenbeleg einscannen
User Story: Als Benutzer der App Möchte ich die Möglichkeit haben, einen Kassenbeleg einzuscannen Damit die Daten in der App erfasst sind
Akzeptanzkriterien: Der Benutzer hat auf der Startseite, die Möglichkeit, in die Scanansicht zu kommen. Die Daten werden in der App gespeichert

Tabelle 5.1: User-Story-1

Titel: Übersicht der Kassenbelege
User Story: Als Benutzer der App Möchte ich eine Übersicht über alle gescannten Kassenbelege einer Filiale haben.
Akzeptanzkriterien: Es gibt eine Ansicht, in der alle erfassten Filialen aufgelistet werden. Innerhalb einer ausgewählten Filiale werden alle dazugehörigen Kassenbelege aufgelistet

Tabelle 5.2: User-Story-2

Titel: Vergleich der Preise
User Story: Als Benutzer der App Möchte ich die Möglichkeit haben, die Preise einzelner Produkte über die Zeit zu vergleichen.
Akzeptanzkriterien: Es gibt eine Ansicht, in der der Preisverlauf eines Produktes angezeigt wird.

Tabelle 5.3: User-Story-3

5.4 Funktionale Anforderungen

5.4.1 Hohe Priorität - Sollte erfüllt sein

	FA-1
Name	Kamerascan
Voraussetzung	Smartphone hat eine Kamera.
Beschreibung	Man kann den Kassenbeleg mit der Smartphone-Kamera einscannen, ohne die App zu verlassen.

Tabelle 5.4: FA-1

	FA-2
Name	Scanbereich markieren
Voraussetzung	FA-1 5.4 ist erfüllt.
Beschreibung	Man kann auf dem Scan, den man gemäss FA-1 5.4 gemacht hat, den Bereich des Kassenbelegs markieren, der verarbeitet werden sollte. Dieser Bereich sollte den Produktnamen, den Stückpreis und die Menge des Produktes beinhalten. Der restliche Text auf dem Beleg soll nicht verarbeitet werden.

Tabelle 5.5: FA-2

	FA-3
Name	Daten aus Scan werden angezeigt
Voraussetzung	FA-2 5.5 ist erfüllt.
Beschreibung	Nachdem der Benutzer gemäss FA-2 5.5 den Scanbereich markiert hat, wird der Text aus diesem Bereich ausgelesen und es wird eine Liste mit den gekauften Produkten und dazugehörigen Stückpreisen und Mengen angezeigt.

Tabelle 5.6: FA-3

	FA-4
Name	Manuelle Korrektur
Voraussetzung	FA-3 5.6 ist erfüllt.
Beschreibung	Die angezeigten Produktnamen, Mengen und dazugehörigen Preise gemäss FA-3 5.6, welche dem Benutzer nach dem Scan angezeigt werden, können vom Benutzer einmalig bearbeitet werden.

Tabelle 5.7: FA-4

	FA-5
Name	Gescannte Daten speichern
Voraussetzung	FA-3 5.6 ist erfüllt.
Beschreibung	Nachdem die Daten gemäss FA-3 5.6 erfasst wurden und optional gemäss FA-4 5.7 einmalig korrigiert wurden, können die Daten als digitaler Kassenbeleg in einem Verzeichnis innerhalb der App lokal persistiert werden. Der Benutzer kann optional einen eigenen Dateinamen angeben, ansonsten wird standardmässig das aktuelle Datum als Dateiname verwendet. Beim Persistieren wird der digitale Kassenbeleg mit einem Zeitstempel versehen.

Tabelle 5.8: FA-5

	FA-6
Name	Manuelle Löschung
Voraussetzung	FA-5 5.8 ist erfüllt.
Beschreibung	Man kann den digitalen Kassenbeleg, der gemäss FA-5 5.8 gespeichert wurde, wieder löschen.

Tabelle 5.9: FA-6

	FA-7
Name	Alte Belege anschauen
Voraussetzung	FA-5 5.8 ist erfüllt.
Beschreibung	Bisher gescannte Kassenbelege, welche gemäss FA-5 5.8 gespeichert wurden, können in der App jederzeit angeschaut werden.

Tabelle 5.10: FA-7

	FA-8
Name	Sortierung
Voraussetzung	FA-7 5.10 ist erfüllt.
Beschreibung	Man kann alle in der App gespeicherten Belege alphanumerisch und nach Erfassungsdatum sortieren. Als Erfassungsdatum gilt der Zeitpunkt des Speicherns des digitalen Kassenbelegs gemäss FA-5 5.8.

Tabelle 5.11: FA-8

	FA-9
Name	Preisänderungen innerhalb eines Verzeichnisses (=Laden)
Voraussetzung	FA-5 5.8 ist erfüllt. Damit man eine positive, neutrale oder negative Preisänderung anzeigen kann, muss ein Produkt mit dem exakt gleichen Namen auf mindestens einem anderen digitalen Kassenbeleg im gleichen Verzeichnis vorhanden sein.
Beschreibung	Die Preise der Produkte auf einem digitalen Beleg werden mit den Produkten auf bereits vorhandenen digitalen Belegen im gleichen Verzeichnis miteinander verglichen und die Preisänderungen werden farblich auf dem digitalen Beleg hervorgehoben. Als Referenz für den Preisvergleich wird jeweils immer der letzte erfasste Preis eines Produkts verwendet.

Tabelle 5.12: FA-9

	FA-10
Name	Vordefinierte Verzeichnisse
Voraussetzung	Keine
Beschreibung	Es gibt innerhalb der App vordefinierte Verzeichnisse. Ein Verzeichnis entspricht einem Laden. Der Benutzer kann einen Kassenbeleg von Laden A in Verzeichnis A speichern.

Tabelle 5.13: FA-10

	FA-11
Name	Warnung bei manueller Löschung
Voraussetzung	FA-6 5.9 ist erfüllt.
Beschreibung	Beim Löschen eines vorhandenen digitalen Belegs in der App wird der Benutzer gewarnt und muss bestätigen, dass er den Beleg löschen möchte.

Tabelle 5.14: FA-11

5.4.2 Mittlere Priorität - Falls Zeit vorhanden

	FA-12
Name	Verzeichnis favorisieren
Voraussetzung	FA-10 5.13 ist erfüllt.
Beschreibung	Der Benutzer kann bestimmte Verzeichnisse als Favorit markieren, sodass diese immer oben in der Verzeichnisliste angezeigt werden. Die Markierung kann auch rückgängig gemacht werden.

Tabelle 5.15: FA-12

	FA-13
Name	Verzeichnis erstellen
Voraussetzung	Keine
Beschreibung	Der Benutzer kann selbst ein Verzeichnis erstellen, beispielsweise für einen Laden, welchen es nur einmal in der Nähe des Benutzers gibt.

Tabelle 5.16: FA-13

	FA-14
Name	Filterfunktion für digitale Belege
Voraussetzung	FA-7 5.10 ist erfüllt.
Beschreibung	Vorhandene digitale Kassenbelege können nach selbst einstellbaren Zeitraum und / oder nach Produkt gefiltert werden.

Tabelle 5.17: FA-14

	FA-15
Name	Liste einzelner Produkte
Voraussetzung	FA-5 5.8 ist erfüllt.
Beschreibung	Der Benutzer kann eine Liste aller Produkte anschauen, welche auf den bereits vorhandenen digitalen Kassenbelegen vorhanden sind. Jedes Produkt, welches schon einmal erfasst wurde, egal in welchem Verzeichnis, wird einmalig aufgelistet.

Tabelle 5.18: FA-15

	FA-16
Name	Suchfunktion für Produkte
Voraussetzung	FA-15 5.18 ist erfüllt.
Beschreibung	Man kann einzelne in der Produktliste gemäss FA-15 5.18 einzelne Produkte suchen.

Tabelle 5.19: FA-16

	FA-17
Name	Detaillierte Preisdaten pro Produkt
Voraussetzung	FA-15 5.18 ist erfüllt.
Beschreibung	Man kann in der Produktliste gemäss FA-15 5.18 auf einen Listeneintrag klicken, und sieht dann detaillierte Preisverläufe dieses Produkts. Pro Verkaufsstelle wird ein Liniendiagramm angezeigt, welches den Preisverlauf vom ersten bis zum letzten Kauf anzeigt. Optional kann der Benutzer ein solches Diagramm für alle Läden anzeigen lassen, um die Preise zwischen den einzelnen Läden zu vergleichen.

Tabelle 5.20: FA-17

	FA-18
Name	Statistik zu Kaufverhalten anzeigen
Voraussetzung	FA-5 5.8 ist erfüllt.
Beschreibung	Man kann eine Statistik anzeigen lassen, in der man sieht, wie viel Geld man bei welchem Laden ausgegeben hat. Den Zeitraum dafür kann man selbst einstellen. Zusätzlich kann der Benutzer sehen, welche Artikel am häufigsten gekauft werden.

Tabelle 5.21: FA-18

	FA-19
Name	Backup und Restore
Voraussetzung	FA-5 5.8 ist erfüllt.
Beschreibung	Man kann erfasste Daten in der App als Datei exportieren und wieder in die App, bspw. auf einem anderen Gerät, importieren. Eine Konto-Funktion, bei der man als Benutzer die Daten im Benutzer-Account speichern kann, ist vorerst nicht geplant.

Tabelle 5.22: FA-19

	FA-20
Name	Darkmode
Voraussetzungen	Keine
Beschreibung	Der Benutzer kann in der App zwischen Light- und Darkmode wählen.

Tabelle 5.23: FA-20

	FA-21
Name	Automatisches Erkennen des Scanbereichs
Voraussetzung	FA-1 5.4 ist erfüllt.
Beschreibung	Anstatt den Scanbereich gemäss FA-2 5.5 manuell zu markieren, wird der Bereich mit den Produktnamen, Mengen und Preisen automatisch erkannt.

Tabelle 5.24: FA-21

5.4.3 Niedrige Priorität - Falls Zeit vorhanden

	FA-22
Name	Barcode- / QR-Scan
Beschreibung	Man kann den Barcode oder QR-Code auf dem Kassenbeleg scannen und damit zusätzlich die hinterlegte Referenznummer zum digital eingescannten Beleg hinzufügen. Nützlich für Belege, welche als Garantieschein dienen

Tabelle 5.25: FA-22

	FA-23
Name	Account
Beschreibung	Man kann einen optionalen Account erstellen und die Daten im Account speichern

Tabelle 5.26: FA-23

	FA-24
Name	Günstigste Verkaufsstelle
Beschreibung	Sofern ein Produkt bereits in einem anderen Laden gekauft wurde, oder die entsprechenden Online Funktionalitäten vorhanden sind, kann überprüft werden, in welchem Laden ein gekauftes Produkt am günstigsten wäre

Tabelle 5.27: FA-24

	FA-25
Name	Statistik über Rabatte
Beschreibung	Sofern Daten über mindestens ein Jahr vorhanden sind, kann man Statistiken anzeigen lassen, zu welcher Zeit ein bestimmter Artikel vergünstigt angeboten wird

Tabelle 5.28: FA-25

	FA-26
Name	Bei der Korrektur bereits vorhandene Artikel anzeigen
Beschreibung	Um die Wahrscheinlichkeit zu minimieren, dass der Benutzer beim Korrigieren des Kassenbelegs gemäss FA-4 5.7 einen Tippfehler macht, kann bei der Korrektur eine Dropdown Liste mit bereits vorhandenen Produktnamen angezeigt werden, welche sich während dem Tippen aktualisiert. So kann der Benutzer den Artikel aus der Dropdown Liste wählen, anstatt den Namen ganz eingeben zu müssen.

Tabelle 5.29: FA-26

5.5 Nicht funktionale Anforderungen

	NFA-1
Name	Android Abdeckung
Qualitätsattribut	Adaptability
Beschreibung	Die App soll auf diversen Geräten funktionieren
Zielwert	Die App funktioniert auf allen im Kapitel 9.1.1 aufgelisteten Geräten

Tabelle 5.30: NFA-1

	NFA-2
Name	Deinstallation
Qualitätsattribut	Confidentiality/Integrity
Beschreibung	Beim Deinstallieren der App sollen alle Daten dieser gelöscht werden.
Zielwert	Alle Daten werden gelöscht.

Tabelle 5.31: NFA-2

	NFA-3
Name	Intuitives UI
Qualitätsattribut	Learnability
Beschreibung	Das UI ist für alle Testpersonen selbsterklärend und es braucht keine Anleitung oder Erklärung
Zielwert	Die Szenarien des Papierprototyps 13.5.1 können ohne Hilfe durchgearbeitet werden.

Tabelle 5.32: NFA-3

	NFA-4
Name	Vertraulichkeit der Daten
Qualitätsattribut	Confidentiality/Integrity
Beschreibung	Andere Apps haben keinen Zugriff auf die Daten, welche in der App gespeichert wurden.
Zielwert	Alle Daten werden in den App internen Daten gespeichert

Tabelle 5.33: NFA-4

	NFA-5
Name	Code Qualität
Qualitätsattribut	Analysability
Beschreibung	Die Code Qualitätsmerkmale gemäss 8.2.1 müssen eingehalten werden.
Zielwert	Siehe 8.2.1

Tabelle 5.34: NFA-5

5.6 Minimal Viable Product

- Man kann direkt aus der App einen Kassenbeleg mit der Kamera scannen.
- Die App kann die Produktnamen und dazugehörige Preise auslesen.

- Nach dem Scan wird eine Liste der einzelnen Artikel mit Ihren Preisen angezeigt.
- Die Liste, welche nach dem Scan angezeigt wird, kann bearbeitet werden.
- Die gescannten und extrahierten Daten können als digitaler Beleg lokal gespeichert werden.
- Auf gespeicherte Belege kann man jederzeit zugreifen.

Kapitel 6

Architektur

6.1 Context

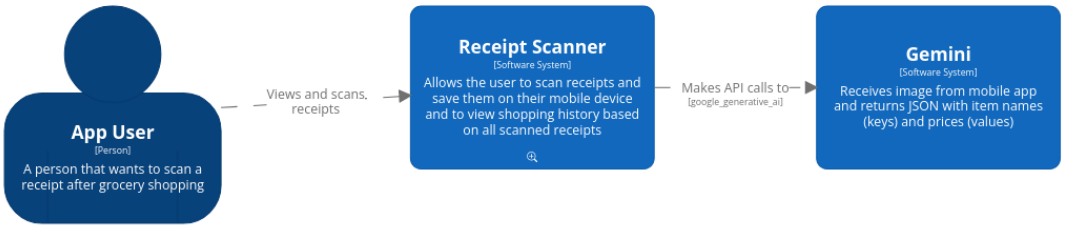


Abbildung 6.1: C4-Kontext

Das Gesamtsystem besteht aus der mobilen App, welche mit Google Gemini kommuniziert. Für die API-Aufrufe wird das google_generative_ai Paket für Flutter verwendet.

6.2 Container

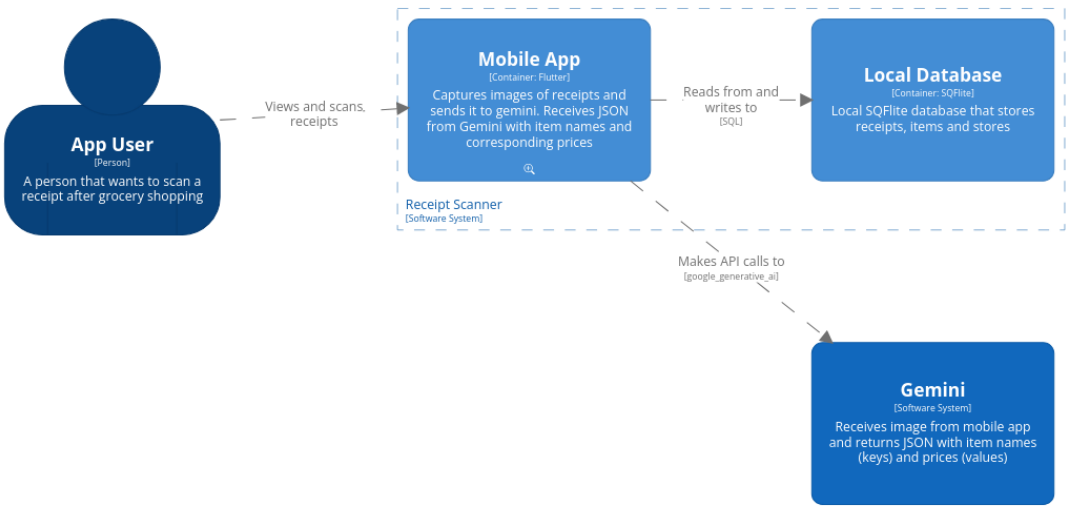


Abbildung 6.2: C4-Container

Die Datenbank befindet sich lokal auf dem Gerät. Dabei handelt es sich um eine SQFlite Datenbank, welches eine Flutter-kompatible Version von SQLite ist. In dieser Datenbank werden die Läden, Kassenbelege und Produkte sowie ihre abhängigkeiten gespeichert.

6.3 Components

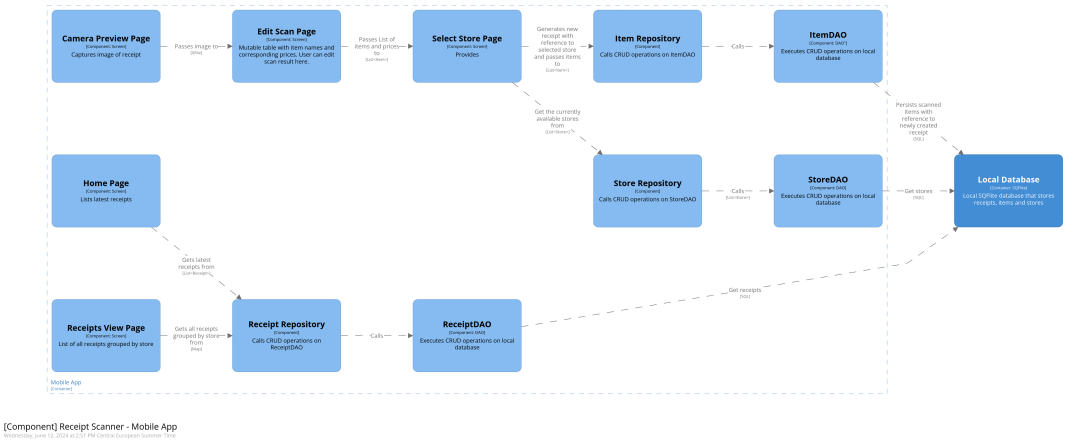


Abbildung 6.3: C4-Components

Im Komponentendiagramm in Abbildung 6.3 sieht man das Zusammenspiel der wichtigsten Komponenten. Der Ablauf, um einen Kassenbeleg aus Nutzersicht einzuscannen, startet oben links bei der Camera Preview Page (Alle Komponenten, welche mit Page enden, gehören zur Präsentationsschicht, mit der der Benutzer direkt interagiert). Das aufgenommene Bild des Kassenbelegs wird an die Edit Scan Page weitergegeben, auf welcher man die Möglichkeit hat, die einzelnen Artikelnamen und Preise anzupassen. Sobald alles angepasst und bestätigt wurde, gelangt man auf die Select Store Page, welche verlangt, dass ein Laden ausgewählt wird, der mit dem Kassenbeleg referenziert werden soll. Nachdem der Laden ausgewählt und bestätigt wurde, wird ein neues Receipt-Objekt (Kassenbeleg) erstellt und persistiert (nicht im Komponentendiagramm abgebildet). Die automatisch generierte ID dieses persistierten Kassenbelegs wird verwendet, um alle Produkte mit diesem Kassenbeleg zu referenzieren, welche von der Edit Scan Page an die Select Store Page übergeben wurden. Für die Persistierung von Entitäten wie Receipts oder Items (Artikel), wird das Repository Entwurfsmuster verwendet. Diese rufen jeweils das DAO-Objekt auf, welche die entsprechende CRUD-Operation auf der Datenbank ausführen.

Ausgehend von den Komponenten Home Page und Receipts View Page in der Abbildung 6.3 links, sieht man das Zusammenspiel der Komponenten beim Auslesen der Belege aus der Datenbank.

6.4 Schichten

Die mobile Applikation ist in die Schichten Daten, Domain und Präsentation aufgeteilt. Der Benutzer interagiert mit der Präsentationsschicht. Diese konzentriert sich auf die Benutzeroberfläche (UI) und die Art und Weise, wie Benutzer mit der Anwendung interagieren. Die Präsentationsschicht enthält Ansichten für die Anzeige der Kassenbelege und verwendet die Kamera-Schnittstelle und weitere Widgets für die Bearbeitung und Anzeige von Kassenbelegen.

Die Domain-Schicht enthält die zentrale Geschäftslogik der Anwendung. Sie definiert Entitäten (Datenstrukturen, die Eingänge darstellen), Repository-Schnittstellen, Dienste und Anwendungsfälle. Sie verwaltet im Wesentlichen die vom Datencontainer abgerufenen Daten und stellt sie dem Präsentationscontainer zur Verfügung.

6.5 Verwendete Entwurfsmuster

6.5.1 DAO Pattern

Für jede Entität gibt es einen DAO (Data Access Object). Die DAOs haben direkten Zugriff auf die Datenbank und definieren die SQL-Abfragen, um die einzelnen Entitäten zu schreiben oder zu lesen. Die DAOs sind Teil der Datenschicht.

6.5.2 Repository Pattern

Für jeden DAO gibt es ein Repository. Der Zugriff auf die Domänen-Objekte erfolgen immer über das Repository, welches die entsprechende Abfrage auf dem DAO aufruft. Das Repository konvertiert beim Lesen die Entität in das Model, indem eine entsprechende Mapper funktion aufgerufen wird. Umgekehrt wird beim Schreiben das Model in eine Entität umgewandelt und vom DAO persistiert. Die Repositories dienen als einheitliche Schnittstelle auf die Objekte und beinhaltet auch Fehlerbehandlung wie bspw. Null-Checks beim Lesen.

6.5.3 DI Pattern

Für die Umsetzung des DI (Dependency Injection) Patterns wird das Flutter Paket GetIt verwendet. Dieses beinhaltet einen Service Locator, mit dem man überall im Projekt auf die registrierten Abhängigkeiten zugreifen kann. Folgende Ressourcen sind als Singleton definiert:

- Store Dao
- Receipt Dao
- Item Dao

- Store Repository
- Receipt Repository
- Item Repository
- App Database (Datenbank)

6.6 Datenbank Schema

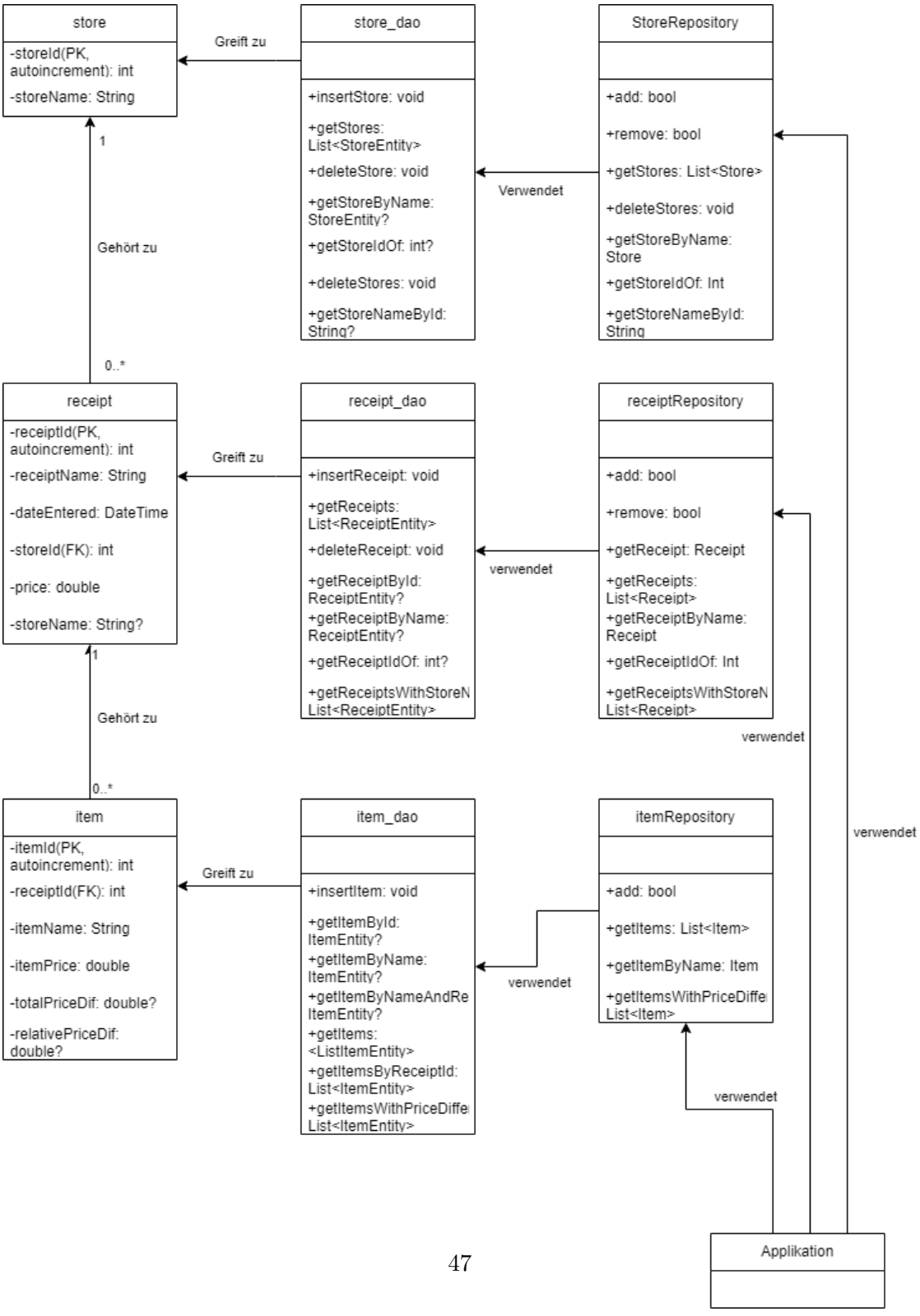


Abbildung 6.4: UML Darstellung der Datenbank

6.7 Verzeichnisstruktur des Flutter Projekts

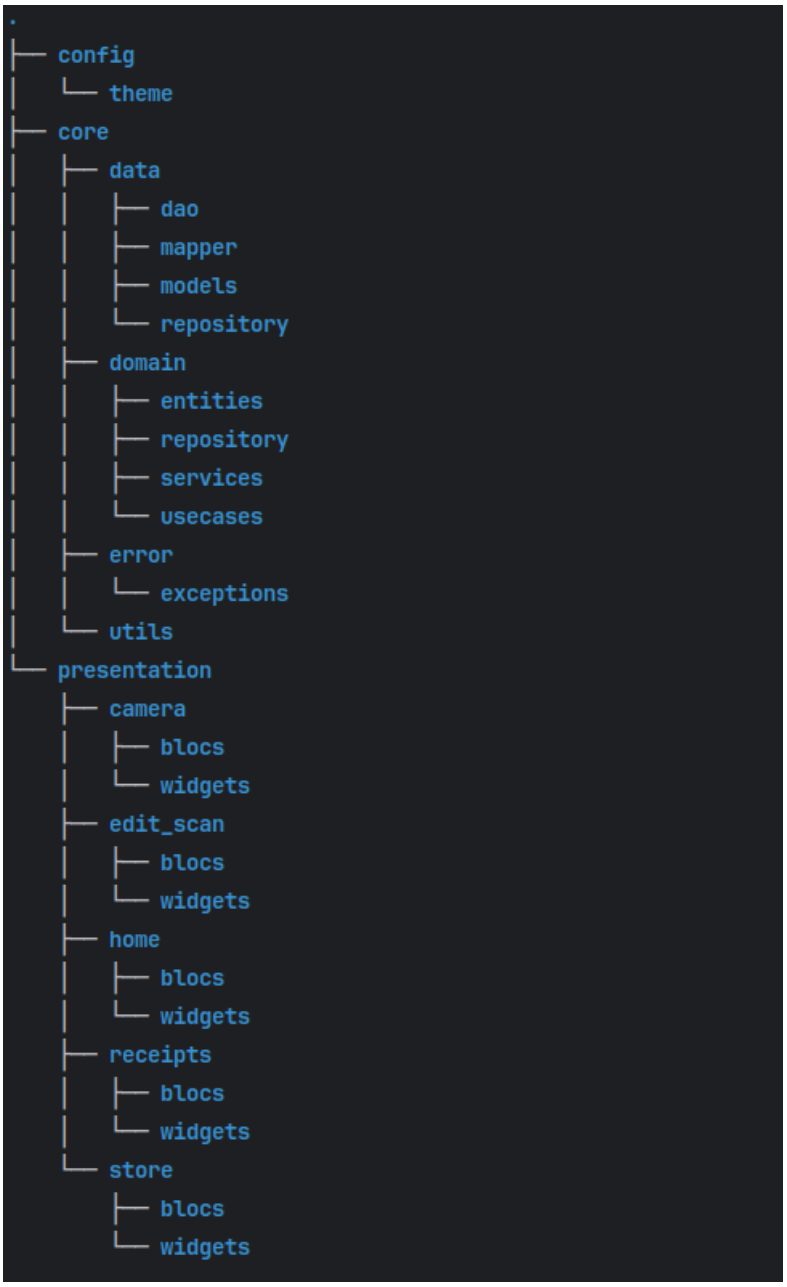


Abbildung 6.5: Verzeichnisstruktur

Kapitel 7

Implementation

7.1 Texterkennung

In diesem Kapitel werden verschiedene Optionen evaluiert, welche verwendet werden können, um den Text zu erkennen und diesen den Kategorien Produktnamen, Menge und Stückpreis zuzuweisen.

7.1.1 Google ML Kit + RegEx

Das Google ML Kit bietet eine Vielzahl von Machine Learning Tools, welche unter einer Apache-2 Lizenz verwendet werden können [16]. Für diese Applikation besonders von Interesse ist dabei die Texterkennung. Der erkannte Text wird automatisch in Blöcke, Zeilen, Elemente und Symbole unterteilt [17]. In dieser Aufteilung ist ein Block ein zusammenhängender Textteil, z.B. ein Absatz. Eine Zeile ist innerhalb eines Blockes eine zusammengehörige Zeile von Text. Ein Element ist eine zusammengehörige Gruppe von alphanumerischen Zeichen, also ein Wort. Ein Symbol ist ein einzelnes alphanumerisches Zeichen. Für ein optimales Ergebnis werden oft einige Preprocessing Schritte benötigt, welche viel Zeit in Anspruch nehmen.

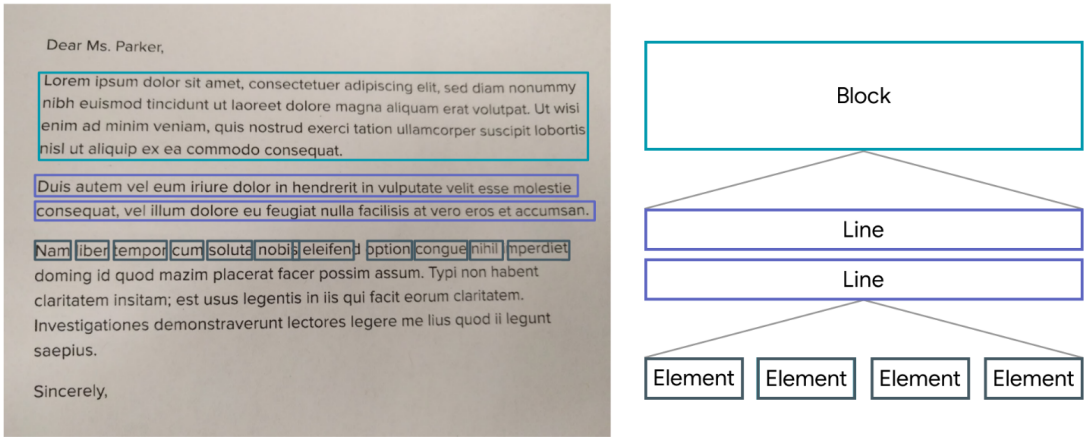


Abbildung 7.1: Beispiel der Texterkennung mit Aufteilung [17]

Textkategorisierung

Die ursprüngliche Lösung basierte auf der Annahme, dass ein Produktname immer eine Kombination von Buchstaben und Zahlen ist, eine Menge immer ein Integer und ein

Preis immer eine Fliesskommazahl. Anhand dieser Annahme wurde der erkannte Text dann von der strengsten bis zu der lockersten Definition kategorisiert. Wenn sich ein Textteil fehlerfrei auf einen Integer parsen lässt, handelt es sich um eine Menge. Lässt er sich nicht auf einen Integer parsen, wird versucht ihn auf einen Float zu parsen. Ist dies erfolgreich, handelt es sich um einen Preis. Ist dies auch nicht der Fall, wird zuerst mithilfe einer Regular Expression überprüft, ob die letzte Stelle des Textes ein Integer ist, um zu erkennen, ob ein Produktname und eine Menge kombiniert wurden, um das Problem der Nähe von Produktnamen und Mengen 10.2.2 zu beheben. Wenn dieser Fall eintrifft, wird der Text aufgetrennt und die Teiltexte der jeweiligen Kategorie zugewiesen. Wenn keiner der vorherigen Fälle zutrifft, wird angenommen, dass es sich um einen Produktnamen handelt.

Preprocessing

Wenn das Bild des Kassenbeleges, aus welchem die Daten extrahiert werden sollen, den perfekten Kontrast und die perfekte Beleuchtung hätte, könnte direkt daraus der Text mit einem guten Resultat extrahiert werden. Da dies aber nicht immer der Fall ist, muss das Bild noch verarbeitet werden. Dafür werden mehrere Schritte durchgeführt.

Image Package Das Image Package wird für die meisten dieser Preprocessing Schritte verwendet, da es bereits fertige Funktionen anbietet [25].

Graustufen

Für OCR Anwendungen werden Bilder normalerweise in Graustufen umgewandelt. Dies reduziert die Komplexität des Bildes und damit die Datenmenge. Die kleinere Datenmenge führt einer Erhöhung der Verarbeitungsgeschwindigkeit der Texterkennung und der weiteren Preprocessing Schritte. Ausserdem können Farbinformationen für die Texterkennung störend wirken.

Kontrast

Durch die Erhöhung des Kontrastes kann der Unterschied zwischen den Pixeln des Textes und den Pixeln des Hintergrundes erhöht werden. Ebenfalls kann damit schlechte Beleuchtung in einem Bild ein bisschen kompensiert werden. Diese beiden Punkte führen zu einer besseren Texterkennung.

Blurring und Smoothing

Blurring kann dabei helfen, Bildrauschen zu reduzieren und Hintergrundtexturen zu unterdrücken. Es führt auch dazu, dass scharfe Kanten von Text geglättet werden, was die Formen der Buchstaben vereinfacht. Diese Punkte verbessern die Texterkennung.

Binarisierung

Bei der Binarisierung wird ein Schwellenwert für die Helligkeit eines Pixels festgelegt. Ist der Pixel heller als der Schwellenwert, wird der Pixel komplett weiss gefärbt. Ist er dunkler, dann wird er komplett schwarz gefärbt. Durch die Binarisierung wird der Text

vom Hintergrund getrennt. Das Finden eines geeigneten Schwellenwerts stellt sich als erhebliche Herausforderung heraus und wird im Kapitel 10.3.1 weiter erläutert.

Endergebnis Preprocessing

Mithilfe der oben beschriebenen Schritte können aus fast allen Bildern von Kassenbelegen die relevanten Daten extrahiert werden. In der Abbildung 7.2 ist ersichtlich, dass das Endergebnis des Preprocessings nicht mehr stark von der Qualität des ursprünglichen Bildes abhängt.

(a) Gutes Input Bild nach Preprocessing

(b) Schlechtes Input Bild nach Preprocessing

Abbildung 7.2: Gegenüberstellung der Ergebnisse des Preprocessings

Vorteile

- Die Implementation dieser Variante ist relativ einfach.
- Die gesamte Verarbeitung findet offline statt.
- Die reine Texterkennung und Zuweisung ist sehr schnell.

Nachteile

- Muss auf jedes Belegformat angepasst werden.
- Scanbereich muss manuell gewählt werden.
- Image Preprocessing muss ausgeführt werden, um das Scan-Resultat zu verbessern.

7.1.2 Bekannte AI APIs

Die bekannten Large Language Models wie ChatGPT, Gemini und Claude erlauben die Verwendung ihrer API, um ihre Funktionalität in beliebigen Applikationen zu verwenden. Die Nützlichkeit der LLMs wurde bereits im Kapitel 2.2 evaluiert und es wurde erkannt, dass sich dieser Ansatz sehr gut für diese Applikation eignet.

Vorteile

- Flexibel mit dem Format der Belege.
- Automatisches Erkennen des relevanten Bereiches.
- Kein lokales Preprocessing nötig.
- Prompt kann angepasst werden, um Ergebnis zu verbessern.

Nachteile

- Nur Online möglich.
- Ev. werden Daten für das Training des Models verwendet.
- Geschwindigkeit hängt von der Netzwerkqualität ab.
- Durch Weiterentwicklung der LLMs kann sich das Ergebnis verändern.

7.1.3 Eigenes ML Model

Eine weitere Option wäre es, ein eigenes Modell zu trainieren. Dies könnte zum Beispiel mit EasyOCR [21] gemacht werden. Um ein eigenes Modell zu trainieren, sind mehrere Schritte notwendig, die mit erheblichem Aufwand verbunden sein können.

Datensammlung und Aufbereitung

Um ein eigenes Modell zu trainieren, wird ein Trainings- und ein Testdatenset benötigt. Diese Datensets sollten im besten Fall aus echten Kassenbelegen bestehen. Generell führt ein grösseres Datenset zu einem besseren Ergebnis, allerdings müssen alle Bilder dieser Datensets annotiert werden, das heisst, die Produktnamen, Mengen und Stückpreise müssen markiert und zugeordnet werden, was natürlich für ein grösseres Datenset länger dauert. Die annotierten Daten müssen in einem Format gespeichert werden, welches für EasyOCR verwendbar ist. Eine weitere Option wäre es, ein synthetisches Datenset zu erstellen. Dies könnte zum Beispiel so gemacht werden, wie es im Artikel "How to Fine-Tune EasyOCR with a Synthetic Dataset" [23] beschrieben wurde. In diesem Artikel wird beschrieben, wie aus einem Kassenbeleg relativ schnell viele Belege erzeugt werden können. Der Vorteil darin wäre, dass auch die Annotation sofort generiert wird und dies nicht von Hand gemacht werden muss. Allerdings ist es relativ schwierig, so auch die Fehler, welche manchmal auf Kassenbelegen vorhanden sind, zu generieren. Ausserdem müsste dieser Ansatz für jedes Format von Belegen, die unterstützt werden sollen, wiederholt werden.

Training des Modells

Nachdem die Daten für das Modell aufbereitet wurden, muss das Modell selbst trainiert werden. Die Dauer dieses Schrittes ist abhängig von der Grösse des Datensets und der verfügbaren Hardware. Auf dem Github Repository von EasyOCR wird ein Beispiel evaluiert, für welches acht Nvidia RTX 3090 Ti, also sehr leistungsstarke Grafikkarten, verwendet wurden. Von den acht Grafikkarten wurden vier für das Training und vier für die Überwachung und das Anpassen der Parameter verwendet. Das Training dieses Beispielmodells hat etwa 14 Stunden gedauert. [22]

Testen und Validieren des Modells

Nachdem das Modell trainiert wurde, sollte es intensiv getestet werden, um sicherzustellen, dass es die erwartete Funktion mit sinnvoller Zuverlässigkeit erfüllt.

Vorteile

- Verarbeitung der Daten wäre schnell.
- Kann offline verwendet werden.
- Genaue Anpassung auf den Anwendungsfall möglich.

Nachteile

- Sehr aufwändige Datenaufbereitung.
- Training kann extrem lange dauern.

7.1.4 Schlussentscheidung

Aufgrund des grossen Aufwandes, wurde die Option ein eigenes Modell zu trainieren bereits früh verworfen. Diese Option sollte allerdings in Betracht gezogen werde, falls diese Applikation ausserhalb des Kontexts einer Bachelorarbeit weiterentwickelt werden sollte, da sie das Potenzial hat, ein optimales Ergebnis zu erbringen.
Aufgrund des geringen initialen Aufwandes wurde zu Beginn das Google ML Kit und eine Regex verwendet. Allerdings hat sich herausgestellt, dass diese Option limitiert ist. Im Verlaufe der Arbeit wurde die Verwendung der Gemini API genauer evaluiert und es wurde entschlossen, diese Option zu verwenden. Diese Entscheidung wurde getroffen, da mit diesem Ansatz das Preprocessing, welches extrem lange dauert, nicht notwendig ist und ausserdem das Format des Beleges weniger relevant ist.

Kapitel 8

Qualität

8.1 Qualitätsanforderungen

Diese Qualitätsanforderungen entsprechen dem ISO / IEC 25010 Standard [20].

8.1.1 Reliability

Eigenschaft	Beschreibung
Maturity	Die Muss-Anforderungen gemäss Kapitel 5.4.1 sollten erfüllt sein.
Availability	App muss auf dem Android-Gerät des Benutzers verfügbar sein, und die Muss-Anforderungen gemäss 5.4.1 sollen offline nutzbar sein. Die App-Daten werden lokal auf dem Gerät des Benutzers gespeichert und können über die App abgerufen werden.
Fault Tolerance	Benutzer kann Texte, welche fehlerhaft erfasst wurden, einmalig bearbeiten, bevor diese gespeichert werden (FA-4 5.7). Bei einem App-Absturz sollen keine Daten verloren gehen.
Recoverability	Erfasste Daten werden lokal auf dem Gerät des Benutzers gespeichert. Der Benutzer kann entscheiden, was mit bereits gespeicherten Daten geschehen soll, wenn die App deinstalliert wird (NFA-2 5.31).

Tabelle 8.1: Reliability

8.1.2 Performance / Efficiency

Attribut	Beschreibung
Time behaviour	Der Scanvorgang sollte möglichst schnell und effizient sein. Da das Google ML Kit kleinere Bilder schneller verarbeiten kann, kann der Benutzer den relevanten Scan-Bereich gemäss FA-2 5.5 markieren.
Resource Utilization	Der Benutzer sollte möglichst wenig Schritte durchführen müssen, um einen Kassenbeleg zu erfassen. Best Case: Der Benutzer fotografiert lediglich den Kassenbeleg und die App extrahiert und verarbeitet die Daten, ohne dass der Benutzer den zu scannenden Bereich markieren soll. Der Best Case ist für diese Arbeit nicht relevant, aber relevant als mögliche Option für die Weiterentwicklung.
Capacity	Da die Daten lokal auf dem Gerät des Benutzers gespeichert werden, muss sichergestellt werden, dass genug Speicher vorhanden ist. Sollte kein Speicher mehr vorhanden sein, wird der Benutzer von der App darauf hingewiesen (NFA-3 5.32).

Tabelle 8.2: Performance / Efficiency

8.1.3 Compatibility

Attribut	Beschreibung
Co-Existence	Für die Nutzung der App benötigt das Android-Gerät eine Kamera. Für die Interaktion der Kamera wird die Camera Bibliothek von Flutter verwendet. Die App muss mit einer Datenbank auf dem Gerät des Benutzers und mit dem Google ML Kit interagieren, welches ebenfalls auf dem Gerät des Benutzers ausgeführt wird.
Interoperability	In dieser Bachelorarbeit ist die Interoperabilität mit anderen Apps (bspw. E-Mail-Client für Austausch von digitalen Belegen) nicht vorgesehen, sondern eher ein Punkt für die mögliche Weiterentwicklung. Gemäss FA-19 5.22 muss man sich auf ein einheitliches Dateiformat für die Backupfunktion einigen. Die Backup- und Restore-Funktionalität gehört nicht zu den Kernfunktionen dieser Bachelorarbeit und wurde mit einer mittleren Priorität eingestuft.

Tabelle 8.3: Compatibility

8.1.4 Usability

Attribut	Beschreibung
Appropriateness / Recognizability	Es wird eine Umfrage durchgeführt, um zu schauen, wie die potenziellen Benutzer, welche befragt werden, auf die Kosten beim Einkaufen achten und ob die Kassenbelege für Budget-Zwecke aufbewahrt werden 13.6. Vor der Realisierung des Software-Prototyps werden einige UI-Tests auf Papier durchgeführt 13.5.1, um die wahrscheinlichsten Szenarien zu testen und die Testperson zu beobachten.
Learnability	Die Benutzung der App sollte für den Benutzer gemäss NFA-4 5.33 selbsterklärend sein. Diese Eigenschaft soll vom Software-Prototypen 4.2 erfüllt werden.
Operability	Die Kernfunktionalitäten, welche im Kapitel 5.4.1 beschrieben sind, müssen am Ende der Bachelorarbeit erfüllt sein.
User Error Protection	Selbst wenn der Benutzer nach dem Erfassen des Belegs eine Korrektur gemäss FA-4 5.7 vornehmen kann, können Tippfehler passieren. Eine Massnahme, um dieses Risiko zu minimieren, wird in FA-24 5.27 beschrieben. Es handelt sich um eine niedrig priorisierte Anforderung, weil es nicht zur Kernfunktionalität der App gehört.
User Interface Aesthetics / Accessibility	Bei der Gestaltung des UI werden die Best Practices gemäss Google angewendet.

Tabelle 8.4: Usability

8.1.5 Security

Attribut	Beschreibung
Confidentiality / Integrity	Daten werden lokal auf dem Gerät des Benutzers gespeichert und können gemäss NFA-5 5.34 nur von der App selbst verwendet werden. Andere Apps sollen keinen Zugriff auf die Daten haben, da während dieser Bachelorarbeit keine Interaktion mit anderen Apps implementiert werden.
Non-Repudiation / Authenticity / Accountability	Es gibt keine Anwendungsfälle in dieser Bachelorarbeit, bei der eine bestimmte Aktion auf einen Benutzer zurückgeführt werden muss, da keine Online-Funktionen implementiert werden. Der Benutzer hat nur sein eigenes Datenset und somit ein Interesse daran, dass dieses korrekt ist. Für die mögliche Weiterentwicklung mit Big Data gemäss 11.1.8 können Daten von Benutzern gesammelt und anaylisert werden, um die Funktionalität der App zu verbessern und zu erweitern. Da wäre es wichtig zu verhindern, dass Benutzer absichtlich falsche Daten eintragen, um die Preisstatistik zu verfälschen.

Tabelle 8.5: Security

8.1.6 Maintainability

Attribut	Beschreibung
Modularity / Reuseability / Modifiability	Die Hauptfunktionen der App werden in separate Module mit klar definierten Schnittstellen ausgelagert. Dabei soll jedes Modul eine klar definierte Funktion haben. Das fördert gleichzeitig die Wiederverwendbarkeit und lose Kopplung von Komponenten. Während der Entwicklung soll darauf geachtet werden, dass man die funktionalen Anforderungen unter 5.4.2 und 5.4.3 ohne grossen Aufwand implementieren kann. Detaillierte Informationen zur Architektur finden sich im Kapitel 6.
Analysability	Die Flutter best practices müssen eingehalten werden
Testability	Elementare Funktionen werden mit Unit-Tests und Integration Tests sichergestellt. Szenarien werden mittels End-User-Tests und einer UI-Test-Library getestet. Detaillierte Informationen finden sich im Kapitel 9.

Tabelle 8.6: Maintainability

8.1.7 Portability

Attribut	Beschreibung
Adaptability	Es gibt verschiedene Android Versionen und Distributionen und unterschiedliche Geräte mit verschiedenen Auflösungen, Kameras, Prozessoren, etc. Mit mehreren Versionen und Geräten testen, welche im Kapitel 9.1.1 beschrieben sind. Für eine weiterführende Arbeit sollte iOS berücksichtigt werden.
Installability	Die App wird während dieser Arbeit nicht im Google Play Store oder im Apple Store veröffentlicht. Wo nötig, wird die App direkt als APK verteilt.

Tabelle 8.7: Portability

8.2 Code Qualität

Um die Qualität des Codes zu überprüfen, werden die folgenden Code Metriken untersucht:

- Cyclomatic comlexity
- Source lines of code
- Maintainability Index
- Number of Arguments
- Maximum nesting Level
- Technical Debt

Diese Metriken werden für alle Files des Projektes erstellt. Sie geben einen Hinweis auf die Wartbarkeit des Codes. Die Metriken wurden mithilfe des Dart Code Metrics Packages evaluiert. Die gesamte Auswertung findet sich im Anhang 13.7.

8.2.1 Cyclomatic complexity

Die Cyclomatic complexity ist ein Mass für die strukturelle Komplexität des Codes. Ein Wert von 1-10 weist darauf hin, dass der Code sehr simpel ist und kaum ein Risiko darstellt, ein Wert von 11-20 weist auf leicht komplexen Code hin, der ein moderates Risiko darstellt, ein Wert von 21-50 weist auf komplexen Code hin, der nach Möglichkeit überarbeitet werden sollte und ein Wert über 50 weist auf extrem komplexen, quasi untestbaren Code hin [7]. Es wurde entschiden, dass der maximale akzeptierte Wert 20 ist. Die komplexeste Klasse wurde mit einem Wert von 18 gemessen und ist somit noch im akzeptierten Bereich. Die genauen Werte können im Anhang 13.7 gefunden werden.

8.3 Codereviews

Im Verlaufe der Arbeit wurden zwei Codereviews durchgeführt. Das folgende Feedback wurde gegeben.

8.3.1 Codereview 1

- Readme anpassen
- Überflüssige Standarddateien löschen
- Falls Mehrsprachigkeit ein Thema ist; Direkt jetzt vorbereiten
- Login als möglicher Ausbauschritt betrachten und evtl. beschreiben
- Feature-basierte Struktur: Eventuell nicht nach Features gliedern
- Immer gleiche Ordnerstruktur verwenden

Aufgrund dieses Feedbacks wurde das Readme angepasst, damit es keine Verwirrung mehr gibt. Einige Standarddateien, die nicht verwendet werden, wurden gelöscht. Die Mehrsprachigkeit wird nicht beachtet, da die Zielgruppe lokal ist. Die Ordnerstruktur wurde angepasst, sodass diese jetzt konsistent ist.

8.3.2 Codereview 2

- Punkte aus Review 1 wurden umgesetzt
- Wichtige Architekturentscheidungen in Doku erwähnen
- Kapitel Ausbaumöglichkeiten in Doku beschreiben
- Unit-Tests sind vorhanden
- Wenn Regex verwendet wird, diese beschreiben
- Generische Werte ersetzen (Platzhalter Text “Das ist die Homepage“
- Was passiert, wenn Kamera Berechtigung nicht vorhanden ist
- “Edit“ kein günstiger Name. Edit wovon?

Aufgrund des Reviews wurde die Regex im Code mit einem beschreibenden Kommentar versehen. Einige Platzhalter Texte wurden ersetzt. “Edit“ Page wurde zu “Edit Scan“ Page umbenannt.

Kapitel 9

Testing

9.1 Geräte

9.1.1 Physische Geräte

Hersteller	Modell	Auflösung in Pixel	Bildschirm Grösse in Zoll	Kamera-Auflösung in MP	Android Version
Samsung	Galaxy S10e	2280x1080	5.8	16	12
Samsung	Galaxy A14	2408x1080	6.6	50	13
Samsung	Galaxy A8	2220x1080	5.6	16	9
HTC	One M7	1920x1080	4.7	4.1	5

Tabelle 9.1: Liste der getesteten Geräte

9.2 Unit Tests

Die Funktionen der Applikation werden mittels Unit-Tests überprüft. Damit soll erreicht werden, dass sich keine Fehler einschleichen, die unerwarteterweise die Funktionalität der Applikation beeinträchtigen. Dass alle Unit-Tests erfolgreich ausgeführt werden, soll jeweils direkt während der Entwicklung überprüft werden. Ausserdem werden alle Unit-Tests ausgeführt, wenn ein Branch im GitHub Projects in den Master Branch gemerged wird.

9.3 Papier UI Tests

Zu Beginn des Projekts gibt es eine Vorstellung, wie das UI der App aussehen sollte. Das Ziel ist es herauszufinden, ob potenzielle Benutzer mit dieser ersten Vorstellung des UI etwas anfangen können und wo es wichtige Verbesserungsoptionen gibt. Man möchte herausfinden, wie der Benutzer mit dem UI interagiert, während er die User Storys, welche in Kapitel 5.3 beschrieben sind, durchläuft. Es sollte auch herausgefunden werden, ob bestimmte zusätzliche Funktionen verwendet werden sollen oder bestehende Funktionen entfernt werden sollen. Die Testprotokolle zu diesen Tests sind im Anhang 13.5.1 abgelegt.

9.3.1 Testszenarien

Für die Tests wurden die folgenden Szenarien getestet:

- Scanne einen einzelnen Einkaufszettel ein und speichere diesen in einem Verzeichnis.
- Scanne einen einzelnen Einkaufszettel und speichere diesen mit dem Namen "lorem_ipsum" in einem Verzeichnis ab.
- Lösche den gescannten Einkaufszettel mit dem Namen "lorem_ipsum".
- Zeige die Preisänderungen der Migros-Quittung mit dem Namen "2024-03-01" an. Zeige, wie oft sich die Preise von "Weggli" in der Migros geändert haben.
- Zeige den Preisverlauf des Produkts "Cola 100ml" an. Es sollen die Preisverläufe aller Läden angezeigt werden, wo das Produkt gekauft wurde.
- Benenne einen erfassten Einkaufszettel um.
- Zeige alle gescannten Einkaufszettel von Migros an.

9.3.2 Resultat der ersten Papier-UI-Tests

Es wurden die Szenarien in Kapitel 9.3.1 getestet. Die vollständigen Testprotokolle sind im Anhang im Kapitel 13.5.1 zu finden. Dabei ist Folgendes aufgefallen:

- Benutzer scannt Beleg ein und speichert diesen im Verzeichnis
  - Alle Testpersonen erkannten nicht, dass man für lange Belege ein zweites Foto machen kann.
  - Zwei Testpersonen erkannten nicht, dass man die Listeneinträge nach dem Scannen bearbeiten kann.
  - Eine Testperson war der Meinung, dass man die gescannten Belege beliebig oft korrigieren können sollte, anstatt nur einmal.
  - Drei Testpersonen erkannten nicht, dass der Standardname geändert werden kann.
- Benutzer löscht den gescannten Kassenbeleg mit dem Namen lorem ipsum
  - Zwei Testpersonen waren verwirrt, dass man den Beleg nur auf der Startseite löschen kann.
  - Eine Testperson erkannte nicht, dass drei Punkte ein Menü darstellen.
  - Eine Testperson war der Meinung, dass es keine Löschfunktion braucht.
- Benutzer zeigt die Preisänderungen des Migros Belegs mit dem Namen 2024-03-01

  - Zwei Testpersonen fragte, seit wann die angezeigte Preisänderung ist, also seit dem ersten Kauf oder seit dem letzten Kauf.
  - Eine Testperson erkannte nicht sofort, dass man den Schieber auf Belege stellen soll.

- Benutzer zeigt alle Preisänderungen vom Artikel Cola 500ml an
  - Eine Testperson war der Meinung, dass es nicht offensichtlich ist, dass die Produktspalte alle Produkte beinhaltet, welche in jedem Laden gekauft wurde.

Aufgrund dieser Tests kann man sagen, dass zuerst die Funktion für das Einscannen von langen Belegen überdacht werden sollte, da niemand verstanden hat, dass man einen langen Beleg durch mehrere Aufnahmen einscannen kann. Man sollte auch dem Benutzer nach dem Scan explizit zeigen, dass man auf einen Listeneintrag tippen kann, um diesen manuell zu bearbeiten, falls etwas falsch gescannt wurde. Die Rückmeldung eines Benutzers, dass man Belege beliebig oft bearbeiten kann anstatt nur einmal nach dem Scan, wird nicht weiter berücksichtigt, weil der Beleg ganz gelöscht und neu aufgenommen werden kann. Man hat generell kein Interesse daran, bereits erfasste Daten zu einem späteren Zeitpunkt zu bearbeiten. Des Weiteren haben drei Benutzer nicht erkannt, dass man den Namen des gescannten Belegs ändern kann. Hier sollte man den Benutzer darauf hinweisen, dass das möglich ist. Beispielsweise mit einem Icon neben dem Textfeld oder indem der Text im Textfeld standardmässig markiert ist. Zwei Benutzer waren verwirrt, dass die Belege nur auf der Startseite umbenannt werden können, das sollte auch auf der Seite, auf der alle Belege angezeigt werden, gemacht werden können. Da mehrere Benutzer nachgefragt haben, auf welchen Preis sich die Preisänderung bezieht, sollte dies klar markiert werden.
Folgende Punkte werden aufgrund der Papier UI-Tests beachtet:

- Einscannen langer Belege wird nochmals überarbeitet.
- Die Möglichkeit, einen Listeneintrag manuell zu bearbeiten, soll klar markiert werden.
- Die Möglichkeit, den Namen eines Beleges zu ändern, soll klar markiert werden.
- Es soll auch auf der “Alle Scans“ Seite möglich sein, den Namen von Belegen zu ändern.
- Es muss klar ersichtlich sein, auf welchen Preis sich die markierte Preisänderung bezieht.

9.4 End-User-Tests

Der Prototyp der App wurde mit mehreren Benutzern getestet. Dabei wurden alle erreichten Funktionen in die Tests eingebunden.

9.4.1 Testszenarien

Die folgenden Szenarien wurden getestet:

- Scanne den Kassenbeleg und speichere ihn im relevanten Verzeichnis.
- Zeige, wie sich die Preise der Produkte, die du eben gescannt hast, verändert haben.
- Scanne einen Beleg, der nicht unterstützt wird und erkläre die Fehlermeldung.

9.4.2 Resultate

Die Testprotokolle sind als ganzes im Anhang 13.5.2 aufzufinden. Es wurde festgestellt, dass die Kernfunktion, also das Einscannen und das Vergleichen der Preise, gut funktioniert. Etwas verwirrend war die Sortierung der Belege, aber auch das haben die Benutzer nach einem Moment verstanden. Es hat sich gezeigt, dass es zwar gut ist, dass dem Benutzer angezeigt wird, wenn ein Fehler beim Scannen passiert ist, dass aber die Fehlermeldungen noch zu ungenau sind und nicht ganz verstanden wird, warum der Scan nicht funktioniert hat. Wenn der Benutzer einen Beleg gescannt hat, der einen Float als Menge hatte, wurde der JSON parsing error angezeigt. Dies hat zu Verwirrung geführt, da die Testpersonen nicht wussten, was das bedeutet. Da die Menge in der aktuellen Version der App nicht mehr verwendet wird, wurde der Code so umgeschrieben, dass diese Spalte ignoriert wird, dadurch kann dieser Fehler nicht mehr auftreten. Die folgenden erkannten Punkte scheinen besonders wichtig:

- Fehlermeldungen müssen auch für Laien Sinn machen.
- Die Sortierung der gescannten Belege sollte nochmals überdacht werden.

Kapitel 10

Aufgetretene Probleme

In den folgenden Abschnitten werden Probleme beschrieben, welche bei der Implementation der App aufgetreten sind.

10.1 Kameravorschau

Der Benutzer soll für das Aufnehmen mit der Kamera die App nicht verlassen müssen. In folgenden Abschnitten werden Probleme im Zusammenhang mit der Kameravorschau beschrieben.

10.1.1 Seitenverhältnis von Kamera und Bildschirm unterschiedlich

Problembeschreibung

Verschiedene Geräte haben unterschiedliche Seitenverhältnisse. Wenn man die Kameravorschau im Vollbildmodus anzeigen möchte, wird dieser verzerrt, weil der Bildschirm ein anderes Seitenverhältnis hat. Das Ziel ist es, dass die Kameravorschau nicht verzerrt oder vergrössert werden muss, um die Qualität der Aufnahme nicht zu beeinträchtigen.

Lösung

Damit die Kameravorschau auf allen Geräten gleich aussehen soll, muss man jeweils das ideale Seitenverhältnis für die Kameravorschau berechnen. Das kann man in Flutter mit der AspectRatio-Klasse:

```
1       return AspectRatio(
2         aspectRatio: 1 / cameraController.value.aspectRatio,
3         child: Stack(
4           children: [
5             ...
6           ],
7         ),
8       );
```

Listing 10.1: Flutter AspectRatio-Klasse

Aktueller Stand

Die Lösung gemäss Beschreibung wurde implementiert. Die Kameravorschau ist nicht mehr verzerrt oder unscharf.

10.2 Lokale Texterkennung mittels OCR Library

Eine Kernfunktion der App ist das Auslesen von Texten aus Bildern von Kassenbelegen. Folgende Daten möchte man auslesen:

- Name des Artikels
- Stückpreis
- Ev. Quantität

Ein möglicher Lösungssatz ist die lokale Texterkennung auf dem Smartphone mittels geeigneter OCR-Bibliothek. Für diesen Zweck wurde die OCR-Funktionalität des Google ML-Kits eingesetzt. Dieses eignet sich besonders gut, weil die Integration mit Flutter sehr einfach ist und weil die Library für die Ausführung auf Mobilgeräten optimiert ist. Probleme, welche beim Einsatz des Google ML-Kits aufgetreten sind, werden hier detailliert beschrieben.

10.2.1 Ähnlichkeit zwischen Buchstaben und Zahlen

Problembeschreibung

Einige Buchstaben sehen aus wie Zahlen. Beispielsweise wird aus einem grossen B eine 8 und umgekehrt. Ein weiteres Beispiel, welches häufig beobachtet wurde, ist, dass ein W als V erkannt wird, wenn bei einem W die Buchstabeninnenfläche zu klein ist. Das liegt daran, dass die Druckqualität der Quittungen nicht immer gut sind.

Lösung

- Manuelle Korrektur: Man bietet dem Benutzer die Möglichkeit, einzelne falsch erkannte Zeichen manuell zu korrigieren.

  + Einfach zu implementieren.

  – Bei vielen Fehlern kann die manuelle Korrektur sehr lange dauern.

- Font-Training: es gibt Buchstaben, welche besser und schlechter erkannt werden, wie bspw. das B oder das W. OCR Modelle lassen sich mittels Bildannotationen hervorragend für neue Schriftarten trainieren, um ein besseres Ergebnis erhalten.

  + Die Anzahl der Texterkennungsfehler kann mit viel Trainingsdaten stark reduziert werden, wodurch die manuellen Korrekturen weniger werden.

  – Für ein gutes Ergebnis ist ein grosses Datenset erforderlich. Zusätzlich müssen die Texte in allen Bildern des Datensets manuell annotiert werden, was sehr zeitintensiv sein kann.

- Man kann nach dem Aufnehmen des Bildes einige Bearbeitungsschritte anwenden, wie bspw. das Erhöhen des Kontrasts. Es gibt verschiedene Techniken, die man anwenden kann, um ein besseres Bildverarbeitungsergebnis zu erhalten.

  + Minimiert die Fehlerrate bei schlechten Lichtverhältnissen und reduziert somit die Anzahl manueller Korrekturen.
  + On-Device Ausführung: Funktioniert auch offline.
  + Externe Ausführung: Schneller als On-Device und schont den Akku des Geräts.

  – On-Device Ausführung: Mehrere Bildverarbeitungsschritte auf dem Gerät auszuführen ist sehr rechenintensiv, was sich negativ auf die Laufzeit und den Stromverbrauch des Geräts auswirkt.
  – Externe Ausführung: Erfordert eine aktive Internetverbindung.

Aktueller Stand

Es wurde eine Möglichkeit implementiert, mit der die Fehler manuell korrigiert werden können. Das funktioniert sehr gut, wenn die Menge an Artikeln überschaubar ist und wenn die Schrift auf dem Beleg gut lesbar ist (bspw. keine weissen Stellen, wo eigentlich Tinte sein sollte). Allerdings braucht man für das Korrigieren sehr lange, wenn es sich um viele Artikel handelt. Aus Zeitgründen hat man sich dazu entschieden, keine weiteren Lösungen zu implementieren. Für eine Weiterentwicklung der App sollte man auf jeden Fall das Font-Training und verschiedene Bearbeitungsschritte in Betracht ziehen. Bei der Implementation der Bearbeitungsschritte On-Device wurde festgestellt, dass dieses sehr lange dauert. Deshalb wäre es besser, die Verarbeitung extern durchzuführen oder die Bearbeitungsschritte zu reduzieren.

10.2.2 Mengenangabe wird zu Produktname genommen

Problembeschreibung

Das Google ML Kit verwendet die Distanz zwischen den erkannten Zeichen, um zu bestimmen, welche Zeichen zu ein Wort bilden. Lange Produktnamen haben je nach Format des Belegs das Problem, dass sie auf dem Beleg sehr nahe an den Mengenangaben sind. Das führt dazu, dass die Menge zum Produktnamen genommen wird.

Lösung

- Manuelle Korrektur: Siehe Kapitel 10.2.1.
- Regex: Bei den beobachteten Kassenbelegen ist es so, dass der Produktname links steht und die Menge steht rechts vom Produktnamen. Man kann somit einen regulären Ausdruck definieren, welcher den Produktnamen vom Preis trennt. Beispielsweise kann man prüfen, ob nach dem Produktnamen eine Ganzzahl kommt oder nicht. Anschliessend prüft man, ob der Produktname mit einer Ganzzahl

endet. Ist das der Fall, trennt man die Zahl vom Namen und nimmt sie als Mengenangabe.

  + Funktioniert sehr gut, wenn man weiss, in welchem Format der Beleg ist. Beispielsweise können Quittungen, welche immer vom gleichen Laden stammen, immer sauber geparst werden.

  – Keine Flexibilität für verschiedene Beleg Formate.
  – Funktioniert nicht, falls ein Produktname tatsächlich mit einem Integer endet.

- Text Detection Modell: Man kann ein eigenes Text-Detection Modell implementieren und dieses mit vielen verschiedenen Belegen trainieren. Ein gutes Beispiel ist EasyOCR, dessen Pipeline zwei Modelle beinhaltet; eines für die Erkennung von Textzeichen, welches man ebenfalls trainieren kann und eines für die Erkennung der relevanten Textbereiche (Artikelname, Stückpreis und Menge).

  + Funktioniert gut für verschiedene Arten von Belegen.
  + Modell lässt sich einfach auf neue Belege trainieren, sofern man die Daten dazu hat.

  – Für ein gutes Ergebnis ist ein grosses Datenset erforderlich. Zusätzlich müssen die Texte in allen Bildern des Datensets manuell annotiert werden, was sehr zeitintensiv sein kann.

Aktueller Stand

Da man verschiedene Beleg Formate scannen möchte, ist der Lösungsansatz mit einer Regex nicht geeignet. Es war die erste Methode, welche man implementiert und getestet hat. Schnell hat sich herausgestellt, dass diese Methode nur sehr schwer erweiterbar ist, wenn bspw. weitere Beleg Formate ebenfalls erkannt werden sollen. Es wurde überprüft, ob man mittels EasyOCR ein eigenes Modell trainieren kann. Es ist zwar der vielversprechendste Lösungsansatz, aber das Trainieren von eigenen KI-Modellen erfordert ggf. sehr viel Einarbeitung in das Thema und viel Zeit für die Erstellung eines Datensets. Aus zeitlichen Gründen ist das nicht realisierbar. Es ist jedoch ein wichtiger Punkt für eine mögliche Weiterentwicklung und sollte auf jeden Fall in Betracht gezogen werden.

10.3 Bildverarbeitung

Bei OCR-Applikationen ist es üblich, dass das Bild bearbeitet wird, damit die OCR-Funktion den Text besser erkennen kann. Für den Anwendungsfall dieser App kann folgende Annahme getroffen werden: Die Bilder haben mehr oder weniger stark beieinander liegende Farbwerte. Der Hintergrund eines Kassenbelegs ist i.d.R. einfarbig. Der Text ist in der Regel schwarz und unterscheidet sich in der Farbintensität deutlich vom Hintergrund. Der ideale Fall für OCR-Anwendungen sind Bilder, welche genau zwei

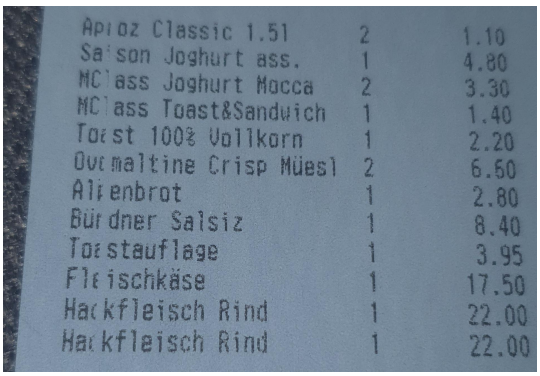
Farbwerte haben: einen für den Text und einen für den Hintergrund. Ziel ist es, das Rauschen aus den Bildern zu entfernen. Wie bereits erwähnt, liegen die Farbwerte aufgrund von äusseren Einflüssen wie Papier, Glättung, Belichtung sehr nahe beieinander, aber sie sind nicht genau gleich. Ein Verfahren, welches hilft, den Text auf Bildern zu verbessern, nennt sich Binarisierung. Dabei wird die Farbintensität jedes Pixels angeschaut

10.3.1 Fixe Schwellenwerte für Binarisierung

Problembeschreibung

In einem ersten Versuch wurde die Binarisierung mit einem fixen Schwellenwert getestet. Man schaute jedes Pixel an und färbte es schwarz oder weiss, wenn die Farbintensität über oder unter diesem fixen Schwellenwert lag. Da der Hintergrund bei Kassenbelegen i.d.R. weiss ist und die Schrift schwarz, wurde angenommen, dass ein fixer Schwellenwert verwendet werden kann. Man wählte einen Wert, der etwa in der Mitte zwischen den Farbwerten weiss und schwarz liegt. Es wurde der Wert 0.55 gewählt. In Abbildung 10.1 sieht man die Fotos von zwei Kassenbelegen, welche für die Binarisierung verwendet wurden.

(a) Gutes Bild eines Beleges



(b) Schlechtes Bild eines Beleges

Abbildung 10.1: Gegenüberstellung zweier Bilder von Belegen

Die Abbildung 10.1a zeigt ein gutes Bild eines Belegs. Die Hintergrundfarbe sowie die Beleuchtung sind gleichmässig. Hat man ein solches Bild, funktioniert der mittlere Schwellenwert von 0.55 sehr gut, um zu entscheiden, ob ein Pixel zur Schrift oder zum Hintergrund gehört. Abbildung 10.1b zeigt ein schlechtes Bild, weil die Beleuchtung im oberen und rechten Teil tiefer ist als an anderen Stellen. Ausserdem ist der Kontrast zwischen Schrift und Hintergrund deutlich geringer. Welche Auswirkungen das auf die Bildbearbeitung hat, sieht man in Abbildung 10.2:

(a) Ergebnis des Thresholding für das gute Bild

(b) Ergebnis des Thresholding für das schlechte Bild

Abbildung 10.2: Gegenüberstellung der Ergebnisse des Thresholdings

Man sieht deutlich, dass der Schwellenwert von 0.55 für Abbildung 10.2a hervorragend funktioniert, aber in Abbildung 10.2b kann man kaum noch Text erkennen. Hier sieht man auch sehr gut, dass bestimmte Stellen in Abbildung 10.2b stärker beleuchtet werden als andere.

Lösung

- Benutzerführung: Wenn der Benutzer ein Foto aufnimmt, kann man die durchschnittliche Helligkeit und die maximalen Abweichungen davon berechnen und dem Benutzer mitteilen, falls die Beleuchtung zu schlecht oder ungleichmässig ist.
- Adaptive Schwellenwerte mittels Pixelwerten ermitteln: Man kann jedes Pixel des Bilds betrachten und dadurch dynamisch den Schwellenwert für die Binarisierung berechnen.
  - \+ Wenn korrekt angewendet, wird der Text auf dem Bild deutlich besser erkannt und es werden weniger Fehler auftreten.
  - – Kann je nach Algorithmus teuer sein und damit nicht geeignet für On-Device Ausführung auf dem Mobilgerät.
  - – Kann je nach Algorithmus lange dauern, was das Benutzererlebnis verschlechtert.
- Machine Learning Mechanismen: Man kann über eine API Machine Learning Dienste verwenden, welche sich um die Bildverarbeitung kümmern und automatisch die idealen Werte ermittelt.
  - \+ Einfach zu implementieren.
  - \+ Schnellere Bildverarbeitung, weil diese extern auf einem leistungsstarken Server stattfindet, anstatt lokal auf dem Gerät.
  - – Erfordert Netzwerkverbindung.
  - – Abhängigkeit von Drittanbieter.
  - – Weniger Kontroller über die einzelnen Bildverarbeitungsschritte.

Aktueller Stand

Aus zeitlichen Gründen wurden die Möglichkeiten für die Binarisierung erkundet und getestet, aber die Implementation in der App wurde niedrig priorisiert, um sich auf die Kernfunktionalität der App zu fokussieren. Es wurde der Lösungsansatz mit der Berechnung des adaptiven Schwellenwertes getestet. Hierbei handelt es sich um das Local Adaptive Thresholding. Das Ergebnis ist in Abbildung 10.3 zu sehen.

(a) Ergebnis des Local Adaptive Thresholding für das gute Bild

(b) Ergebnis des Local Adaptive Thresholding für das schlechte Bild

Abbildung 10.3: Gegenüberstellung der Ergebnisse des Local Adaptive Thresholdings

Man sieht, dass die Binarisierung für das schlecht beleuchtete Bild viel besser funktioniert. Das Local Adaptive Thresholding eignet sich besonders gut für ungleichmässig beleuchtete Bilder, weil für jedes Pixel (unter Berücksichtigung der Nachbarpixel) ein eigener Schwellenwert ermittelt wird. So werden Ungleichheiten korrigiert. Diese Methode ist für die On-Device-Ausführung aber nicht geeignet, da sie sehr teuer ist. Für eine Weiterentwicklung der App sollte man von daher sicherstellen, dass die Beleuchtung gut ist und ggf. eine entsprechende Benutzerführung gemäss Lösungsansatz implementieren.

Daraufhin sollte es nicht notwendig sein, für jedes Pixel einen eignen Schwellenwert zu berechnen, sondern evtl. nur einen einzigen für das ganze Bild. Details diesbezüglich werden in Abschnitt 10.3.2 erläutert. Aufgrund der Einfachheit und Effektivität der Lösung mittels AI, wurde entschieden, diesen Ansatz weiterzuverfolgen.

10.3.2 Binarisierung sehr langsam

Problembeschreibung

Ein Verfahren, welches für die Trennung von Text und Hintergrund verwendet wurde, nennt sich Local Adaptive Thresholding. Dieses Verfahren erstellt einen individuellen Schwellenwert pro Pixel. Wen der Farbwert des Pixels (für den der Schwellenwert berechnet wurde) diesen Schwellenwert unter- bzw. überschreitet, dann wird es weiss resp. schwarz eingefärbt. Das Problem ist, dass die Berechnung eines Schwellenwertes für jeden einzelnen Pixel sehr teuer und langsam ist.

Lösung

- Farbwerte des Bildes minimieren: Man kann die Laufzeit senken, indem man schwarz-weiss Bilder als Input für das Local Adaptive Thresholding verwendet.
  - + Schwarz-weiss Bilder haben Farbwerte von 0-255. Die Berechnung der Schwellenwerte ist damit deutlich weniger ressourcenintensiv als bei Farbbildern.
  - + Schont den Akku des Geräts.
  - – Je nach Beleuchtung kann es sein, dass nach der schwarz-weiss Konvertierung der Unterschied zwischen Text und Bild nicht mehr gut ist und einzelne Pixel falsch eingefärbt werden.
  - – Selbst mit schwarz-weissen Bildern muss der Schwellenwert jedes einzelnen Pixels berechnet werden, was je nach Gerät trotzdem noch lange dauern und viel Strom verbrauchen kann.
- Auslagerung der Bildverarbeitung an externen Server.
  - + Sehr schnell.
  - + Schont Akku des Geräts im Vergleich zu On-Device Ausführung.
  - – Erfordert Internetverbindung.
  - – Man benötigt einen Server, was die Komplexität der Architektur erhöht.
- Verwendung von globalem Thresholding: Aufgrund der Annahme in der Problembeschreibung ist es ggf. nicht notwendig, einen Schwellenwert für jedes Pixel zu berechnen. Man kann einen Algorithmus, wie bspw. Otsu's Methode verwenden. Diese berechnet einen Schwellenwert pro Bild, nicht pro Pixel. Auch für diese Methode sollten schwarz-weiss Bilder verwendet werden.

+ Sehr gut geeignet für On-Device Ausführung.
+ Schont Akku des Geräts im Vergleich zu Local Adaptive Thresholding.

– Wenn es auf dem Bild Stellen gibt, bei denen die Beleuchtungen stark variieren, funktioniert Otsu's Methode nicht gut. In diesem Fall wäre das Local Adaptive Thresholding die bessere Methode.

Vergleich verschiedener Binarisierungsmethoden

Da festgestellt wurde, dass ein einziger globaler Schwellwert für die Binarisierung nicht funktioniert und Local Adaptive thresholding zu langsam ist, wurde wie im Paper "Robust Pre-processing Techniques for OCR Applications on Mobile Devices" [24] die Otsu's Methode ausprobiert. Alle drei Methoden wurden mit drei Bildern verglichen. Die Ergebnisse des Vergleiches sind in der Abbildung 10.4 zu sehen. Das erste Bild quasi der Optimalfall mit gleichmässiger, guter Beleuchtung. Im zweiten Bild ist die Beleuchtung ungleichmässig, in diesem Bild durch den Kamerablitz ausgelöst. Im dritten Bild ist ein Teil des Bildes im Schatten, hier ausgelöst durch eine starke Lichtquelle, die sich hinter dem Smartphone befindet. Interessant ist, dass das Ergebnis des Local Adaptive thresholdings scheinbar schlecht ist. Es sollte aber beachtet werden, dass die Methoden nicht optimiert wurden und dieser Algorithmus in Kombination mit anderen Preprocessing Schritten wie im Kapitel 7.1.1 beschrieben ein sehr gutes Ergebnis liefert. Hier ging es hauptsächlich um den Vergleich der Zeit. Dieser Test wurde nicht auf dem Smartphone durchgeführt, somit ist die absolute gemessene Zeit nicht repräsentativ, dennoch ist deutlich sichtbar, dass die einfache Binarisierungsmethode und die Otsu's Methode in etwa gleich schnell sind und die Local Adaptive thresholding Methode deutlich langsamer ist. Es ist aber auch ersichtlich, dass nur die Local Adaptive thresholding Methode mit allen Lichtverhältnissen arbeiten kann. Durch diese Tests wurde erkannt, dass es sinnvoller ist, den Benutzer so zu führen, dass das Input Bild bereits gut ist, als zu versuchen, eine Binarisierungsmethode zu optimieren, da die einzige getestete Methode, die immer funktioniert, sehr langsam ist.

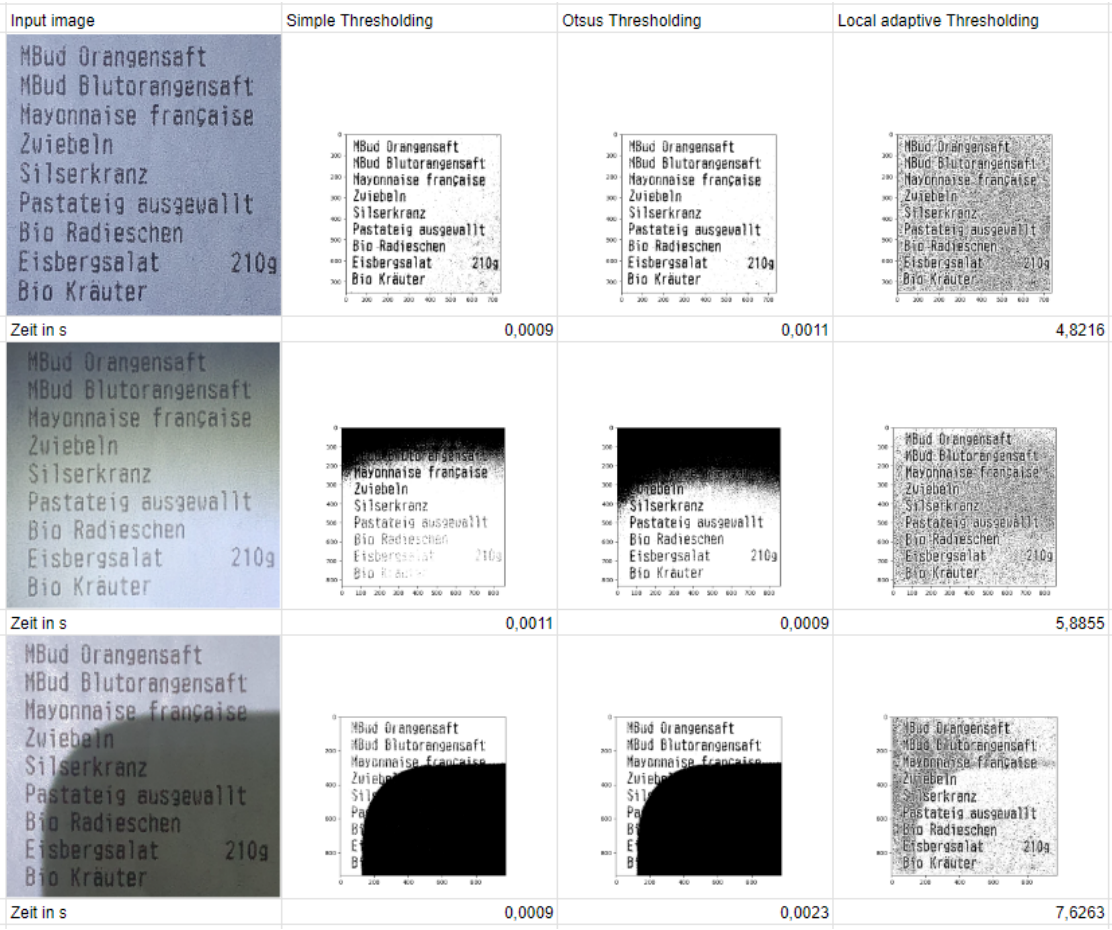


Abbildung 10.4: Vergleich von Thresholding Methoden

Aktueller Stand

Die einfachere Lösung ist die Verwendung von Schwarz-Weiss Bildern, unabhängig davon, ob man mit lokalen oder globalen Schwellenwerten arbeitet. Aus zeitlichen Gründen und da die Texterkennung mittels AI API mit ungleichmässiger Beleuchtung zurechtkommt, hat man sich aber dazu entschieden, die Implementation für die Binarisierung des Bildes niedrig zu priorisieren, um sich auf die Kernfunktionalität der App zu fokussieren. Bei einer Weiterentwicklung der App sollte aufgrund der Annahme in der Problembeschreibung kein lokales Thresholding verwendet werden.

Kapitel 11

Ausblick

11.1 Mögliche Weiterentwicklungen

11.1.1 Erfassen von sehr langen Belegen

In den meisten Fällen lassen sich die Preise und Produktnamen eines Kassenbeleges, auch wenn dieser sehr lange ist, in einem einzigen Bild einscannen. Für die wenigen Kassenbelege, für die dies nicht der Fall ist, gibt es verschiedene Möglichkeiten, dem Benutzer doch eine angemessene Erfahrung zu ermöglichen.

Scrollende Aufnahme

Man könnte dem Benutzer die Möglichkeit geben, den Scanbereich während dem Scannen zu verschieben, ähnlich wie die beim Aufnehmen eines Panoramabildes gemacht wird.

Mehrere Bilder, die zusammengefügt werden

Man könnte den Benutzer bitten, den Beleg in mehreren Fotos einzuscannen, welche dann intern zusammengefügt werden. Dabei muss aber beachtet werden, dass man den Benutzer darauf aufmerksam macht, welchen Bereich er bereits eingescannt hat und wo er mit dem nächsten Foto ansetzen muss. Dies könnte zum Beispiel damit erreicht werden, dass der letzte Teil des bereits gescannten Bereiches am oberen Bildschirmrand bzw. über dem Kamerabereich angezeigt wird.

11.1.2 Duplicate detection

Wenn der Benutzer einen Kassenbeleg einscannt, den er bereits eingescannt hat, werden die gleichen Daten ein zweites Mal erfasst. Auf dem “zweiten“ Beleg werden dann die Preise mit denen, vom ersten Beleg verglichen. Da die gleichen Produkte dann den gleichen Preis haben, wird der Benutzer keine Veränderung im Preis sehen. Es könnte erkannt werden, dass bereits ein Beleg mit den gleichen Produkten und den gleichen Preisen vorhanden ist. Dann könnte der Benutzer gefragt werden, ob er den neuen Beleg wirklich speichern möchte beziehungsweise, ob er sich sicher ist, dass es sich um einen neuen Beleg handelt.

11.1.3 Named Entity Recognition und Relation Extraction

Die Forschungsarbeit „Named Entity Recognition and Relation Extraction: State-of-the-Art“ [31] bietet eine umfassende Übersicht über Techniken und Fortschritte bei der

Erkennung benannter Entitäten (Named Entity Recognition, NER) und der Extraktion von Beziehungen (Relation Extraction, RE). Die Autoren konzentrieren sich insbesondere auf Entwicklungen im Bereich der maschinellen Lernmodelle und adressieren die wachsende Notwendigkeit, die riesigen Mengen an unstrukturierten Textdaten, die online erzeugt werden, effektiv zu verarbeiten und zu strukturieren. Die wichtigsten Punkte des Papers umfassen:

- Erkennung benannter Entitäten (NER): Dieser Abschnitt behandelt die Identifikation und Klassifizierung von Entitäten in Texten, wie Personen, Organisationen und Orte. Die Autoren diskutieren verschiedene Ansätze, von frühen Heuristiken bis zu modernen Deep-Learning-Modellen, die aktuell den Stand der Technik darstellen.

- Extraktion von Beziehungen (RE): Hier wird die Herausforderung beschrieben, wie Beziehungen zwischen identifizierten Entitäten aus Texten extrahiert werden können. Die Autoren erörtern Methoden, die von regelbasierten Ansätzen bis zu fortschrittlichen maschinellen Lernverfahren reichen, die auf neuralen Netzen basieren.

- Fortschritte und Herausforderungen: Es wird festgestellt, dass trotz bedeutender Fortschritte bei der Verwendung von Deep-Learning-Techniken zur Verbesserung der Genauigkeit der Informationsgewinnung die Verfügbarkeit von annotierten Daten nach wie vor eine Herausforderung darstellt. Dies beschränkt den Fortschritt in weniger erforschten Anwendungsbereichen.

- Anwendungsbereiche und zukünftige Richtungen: Die Autoren betonen die Anwendung von NER und RE in verschiedenen Bereichen, darunter soziale Medien, Gesundheitswesen und juristische Texte, und diskutieren zukünftige Forschungsrichtungen, die auf die Integration von mehr Kontextverständnis und die Verarbeitung von Sprachnuancen abzielen.

Insgesamt liefert das Paper eine Analyse der Methoden und Technologien, die zur Strukturierung von Textdaten verwendet werden, und zeigt sowohl die aktuellen Möglichkeiten als auch die Grenzen dieser Technologien auf. Mithilfe der im Paper beschriebenen Technologien könnte es dem Benutzer ermöglicht werden, nur ein Bild des gesamten Kassenbeleges zu verwenden, ohne dass ein bestimmter Bereich markiert werden muss. Mithilfe der Gemini API wird dies in der aktuellen Version der App erreicht, um dies mit einem eigenen ML Modell zu erreichen, müssten die im Paper beschriebenen Schritte aber erarbeitet werden.

11.1.4 Eigenes Modell trainieren

Eine Möglichkeit, die App deutlich zu verbessern, wäre die Verwendung eines ML Modells, welches explizit auf diese Aufgabe trainiert wurde. Diese Option wurde bereits im Kapitel 7.1.3 evaluiert. Es wurde allerdings entschieden, dass der Aufwand für diese Arbeit zu gross wäre. Um eine marktreife Applikation zu bauen, wäre es aber durchaus

denkbar, die nötige Zeit zu investieren, um diesen Lösungsansatz zu implementieren. Dazu müssten grob drei Schritte erarbeitet werden: Ein Datenset erstellen, entweder durch Sammeln echter Daten oder durch das Erstellen eines synthetischen Datensets. Wenn die Trainings- und Testdatensets erstellt wurden, kann das Modell trainiert werden. Dieser Schritt erfordert sehr viel Zeit. Wenn das Modell trainiert wurde, kann es mit dem Testdatenset getestet werden. Diese beiden Schritte werden so lange wiederholt, bis das Modell ein Testergebnis liefert, das als gut genug erachtet wird.

11.1.5 Benutzerführung

Im Kapitel 10.3.2 wurde erkannt, dass es nicht nötig ist, komplexe und zeitaufwendige Preprocessing Schritte durchzuführen, wenn das ursprüngliche Bild bereits gleichmässig beleuchtet ist. Es wäre also eine wertvolle Erweiterung, wenn der Benutzer direkt beim Aufnehmen eines Bildes darauf hingewiesen wird, wie gleichmässig beleuchtet das Bild ist. Es könnte versucht werden, die durchschnittliche Helligkeit des Bildes zu berechnen und zu überprüfen, dass keine Bildbereiche zu stark von diesem Wert abweichen. Falls das Bild zu stark von diesem Wert abweicht, könnte der Benutzer darauf hingewiesen werden. Der Benutzer könnte dann aufgefordert werden, das Bild nochmals in besseren Lichtverhältnissen aufzunehmen. Auch hier muss allerdings darauf geachtet werden, wie der Benutzer darauf aufmerksam gemacht wird und wie er geführt wird, damit er nicht frustriert wird und den Vorgang abbricht.

11.1.6 Inflationsrechner

Es könnte eine Art Inflationsrechner erarbeitet werden, der evaluiert, ob ein Produkt nur wegen der Inflation teurer geworden ist, oder ob das Produkt mehr, als es durch die Inflation zu erwarten ist, teurer geworden ist. Somit könnte der Benutzer darauf achten, dass er Produkte kauft, die nicht zusätzlich verteuert wurden. Dazu müssten Daten über die aktuelle Inflation am Standort des Benutzers gesammelt werden, die erwartete Preisänderung anhand dieser Daten berechnet werden und die tatsächliche Preisänderung mit dieser vergleichen.

11.1.7 Einfaches Einfügen einer Zeile

In einigen Fällen, besonders wenn der Kassenbeleg etwas schief eingescannt wird, kann es vorkommen, dass die Mengen oder Preise um eine Zeile nach oben oder unten rutschen. Wenn dies geschieht, muss der Benutzer in der aktuellen Version alle folgenden Mengen oder Preise anpassen. Wenn das bei einem der letzten Einträge passiert, ist der Aufwand dafür relativ gering, wenn es aber bei einer der ersten Zeilen der Fall ist und der Beleg lang ist, ist dieser Fehler extrem unangenehm zu korrigieren. Um es dem Benutzer einfacher zu machen, diesen Fehler zu beheben, könnte eine Möglichkeit eingebaut werden, die es erlaubt, eine Menge oder einen Preis an einer Stelle in der Liste einzufügen oder zu löschen, was den Rest der Liste wieder in die richtige Position verschieben würde.

11.1.8 Big Data

Wenn eine genügend grosse Nutzerbasis vorhanden ist, können die Daten aller Benutzer anonymisiert gesammelt und allen Benutzern zur Verfügung gestellt werden. Mit diesen Daten könnten dann diverse Weiterentwicklungen erarbeitet werden. Einige Beispiele wären die Anzeige von mehr historischen Daten, um zu sehen, wann ein Produkt, das letzte Mal so teuer war, wie es jetzt ist. Eine weitere Option wäre es, den Benutzern zu zeigen, in welchem Laden ihre Einkäufe wie viel billiger wären und zu evaluieren, ob es sich für sie lohnt, mit dem Auto oder dem ÖV zu einem weiter entfernten Laden zu reisen, um diese Einkäufe zu tätigen. Dazu müssten zusätzliche Daten gesammelt und gespeichert werden, die Informationen über den Standort des Benutzers beim Erfassen eines Beleges geben. Falls dieser Ansatz verfolgt wird, muss der Datenschutz unbedingt berücksichtigt werden, da dann auch Daten in Bezug auf den Benutzer gesammelt werden.

11.1.9 Umgang mit Rabattaktionen

In der aktuellen Version der App werden Rabattaktionen nicht berücksichtigt. Je nachdem, wie ein Laden Rabattaktionen auf dem Beleg vermerkt, kann dies dazu führen, dass fälschlicherweise eine Preisänderung erkannt wird, die eigentlich gar keine Preisänderung ist. Dies ist der Fall, wenn auf einem Beleg die Rabattaktion direkt den Stückpreis verändert. Es könnte nach einer Option gesucht werden, um auf den Belegen zu erkennen, dass eine Rabattaktion einen Preis beeinflusst hat, und wo diese Änderung angezeigt wird. Zum Beispiel könnte zu jedem Produkt eine Checkbox “Rabatt“ angezeigt werden. Wählt der Benutzer diese Checkbox aus, wird die Preisänderung anders markiert, zum Beispiel durch blauen Text, um dem Benutzer zu zeigen, dass es sich um einen Rabatt handelt. In Kombination mit einer Online Funktion, könnte dann zum Beispiel den Benutzern gezeigt werden, dass Produkte, die sie oft kaufen, gerade heruntergesetzt sind und sie so Geld sparen könnten. Wenn Rabattaktionen korrekt erkannt werden, könnte auch verhindert werden, dass Preisänderungen falsch erkannt werden.

Kapitel 12

Projektplanung

12.1 Projektplanungsmethode

Für die Langzeitplanung wird das Rational Unified Process (RUP) Modell verwendet und für die Kurzzeitplanung werden Sprints erstellt, ähnlich wie bei SCRUM. Das RUP Modell definiert die Meilensteine. Die Meilensteine werden in Iterationen (Sprints) abgearbeitet, wobei man sich für eine Iteration auf zwei Wochen festgelegt hat. Diese Projektplanungsmethode wurde gewählt, da die Projektmitarbeiter bereits positive Erfahrungen damit sammeln konnten und weil diese Arbeitsmethoden für den Umfang dieser Bachelorarbeit als angemessen betrachtet werden.

Man hat sich bewusst für eine agile Arbeitsmethode entschieden, da die Entwicklung einer OCR-Applikation anspruchsvoll ist und weil es verschiedene Lösungsansätze dafür gibt. Deshalb ist es wahrscheinlich, dass sich die Anforderungen im Projektverlauf ändern werden.

12.2 Involvierte Personen und Verantwortlichkeiten

Name	Rolle - Verantwortlichkeit
Josip Di Benedetto	Entwickler - Implementation der Funktionalität und Dokumentation.
Fabian Freitag	Entwickler - Implementation der Funktionalität und Dokumentation.
Prof. Dr. Markus Stolze	Betreuer - Unterstützung für das Projekt und Bewertung des Projekts.
Prof. Dr. Mitra Purandare	Korreferent
Markus Flückiger	Korreferent

Tabelle 12.1: Rollen und Verantwortlichkeiten

12.3 Prozesse

Jeden Sprint werden folgende wiederkehrende Aufgaben ausgeführt:

Aufgabe	Beschreibung	Wann
Wöchentliches Meeting	Meeting mit Betreuer, um über den aktuellen Stand des Projekts zu informieren und ggf. Rückmeldung zu erhalten.	Jeden Montag um 17.00 Uhr
Sprint Retrospective	Nach Abschluss des Sprints wird dessen Zielerreichung von den Projektmitarbeitern evaluiert.	Montag nach Abschluss von Sprint
Sprint Planung	Die Projektmitarbeiter planen die Aufgaben für den neuen Sprint und erfassen diese in Jira.	Montag nach Abschluss von Sprint
Risiken neu bewerten und dokumentieren	Die Risikoanalyse in der Doku aktualisieren. Neue Risiken hinzufügen, oder bestehende Risiken anpassen oder löschen.	Gegen Ende des Sprints
Entscheidungsprotokoll	Entscheidungsprotokoll für wöchentliches Meeting an Betreuer senden.	Jeden Dienstag

Tabelle 12.2: Prozesse

12.4 Langzeitplanung

Das Projekt wird gemäss RUP in vier Phasen unterteilt:

Phase	Start	Ende
Inception	2024-02-19	2024-03-03
Elaboration	2024-03-04	2024-03-31
Construction	2024-04-01	2024-06-02
Transition	2024-06-03	2024-06-14

Tabelle 12.3: Phasen

12.4.1 Inception Phase - Sprint 1

In dieser Phase werden der Langzeitprojektplan und die Aufgabenstellung genau definiert. Anschliessend folgt die Ausarbeitung der groben Muss-Anforderungen. Die Erarbeitung der groben Muss-Anforderungen hat die höchste Priorität in dieser Phase, da diese notwendig sind, um mit der Elaboration-Phase weiter machen zu können.

12.4.2 Elaboration Phase - Sprint 2, 3

Zu Beginn von Sprint 2 wird aus den groben Anforderungen eine detaillierte Anforderungsanalyse erstellt. Daraufhin ist es leichter, die Risiken zu identifizieren und erste Architekturentscheidungen zu treffen. In der Hälfe von Sprint 2 wird damit begonnen,

einen Papierprototyp für das UI auszuarbeiten. Mit diesem Prototyp werden Testszenarien erstellt und an einigen Testpersonen durchgeführt. Das soll helfen, offensichtliche Fehler im UI zu identifizieren. Zudem soll eine Umfrage erstellt werden, um einen Bedarf für eine solche Applikation zu prüfen. Die Umfrage soll Fragen bezüglich Einkaufsgewohnheiten beinhalten und an möglichst viele Studenten versendet werden.

Zu Beginn von Sprint 3 werden Frameworks für die Entwicklung einer Android-App evaluiert und anschliessend wird ein Git Repository erstellt. Wenn möglich, soll gleich die CI/CD Pipeline erstellt werden; ist in dieser Phase des Projekts aber nicht von hoher Priorität. Wichtiger ist die Ausarbeitung einer App, welche mit der Gerätekamera ein Bild aufnehmen kann und anschliessend den Text auf diesem Bild auslesen. Ziel ist es, etwas Übung und Routine im Umgang mit dem ausgewählten Framework zu erhalten und eine Basis für den weiteren Projektverlauf zu schaffen.

12.4.3 Construction Phase - Sprint 4, 5, 6, 7

In Sprint 4 geht es darum, herauszufinden, wie man strukturierten Text aus Fotos von Kassenbelegen extrahieren kann. Die Herausforderung besteht darin, dass Kassenbelege unterschiedliche Formate haben. Es gibt die Möglichkeit, dass der Benutzer den Artikelbereich mit Produktnamen, Preisen und Mengen selbst markiert, oder man könnte Technologien einsetzen, welche diesen Bereich automatisch erkennen.

Da der Scan von Text auf Bildern nicht zu 100 Prozent genau ist, werden Fehler auftreten. Als Benutzer möchte man eine Möglichkeit haben, diese Fehler manuell korrigieren zu können. Das ist wichtig, weil Preisvergleiche zweier Produkte nur funktionieren, wenn die Produktnamen gleich sind. Die Ausarbeitung einer entsprechenden Funktion ist das Ziel von Sprint 5.

Ziel von Sprint 6 ist es, dass die erfassten bzw. korrigierten Daten persistiert werden können. Daten sollen lokal auf dem Gerät persistiert werden, auf dem die App ausgeführt wird.

Im letzten Sprint der Construction Phase soll die Möglichkeit implementiert werden, erfasste Belege jederzeit anschauen zu können. Zudem soll auf dem Beleg die Preisänderung eines Produkts seit dem letzten Einkauf angezeigt werden, falls es eine Preisänderung gab. Dabei werden nur Produkte mit dem gleichen Namen und im gleichen Verzeichnis (=Laden) verglichen werden, weil ein gleichnamiges Produkt in einem anderen Laden grundsätzlich einen anderen Preis haben kann.

12.4.4 Transition Phase - Sprint 8

Zu Beginn von Sprint 8 soll eine App vorhanden sein, mit der man Belege erfassen kann und Preisänderungen angezeigt bekommt, sofern diese vorhanden sind. Die wichtigsten Risiken und die grössten Herausforderungen beim Entwickeln dieser App wurden

identifiziert. Ebenfalls wichtig ist, dass man die Vorteile und Nachteile verschiedener Technologien evaluiert, welche folgende Probleme lösen:

- Erkennung von Text aus Bildern von Kassenbelegen.
- Detektieren des Bereichs auf dem Foto, auf dem sich die Produktnamen, Stückpreise und Mengenangaben finden.

Im weiteren Verlauf der Transition Phase werden die Dokumentation und alle anderen Dokumente, welche eingereicht werden müssen, fertiggestellt.

12.5 Meilensteine

Phase	Ziel	Bis
Inception	Aufgabenstellung definieren, Risikomatrix erstellen, Muss-Requirements festlegen, groben Langzeit Projektplan erstellen.	2024-03-03
Elaboration	Detaillierte Requirement Analyse, wichtige Architekturentscheidungen treffen, hohe Risiken vermindern, erste UI Tests auf Papier durchführen und Umfrage zu Kaufverhalten durchführen.	2024-03-17
Elaboration	Code Repository mit CI/CD Pipeline aufsetzen. Prototyp erstellen: Starten der Kamera und Zuschneiden des Bildes funktioniert. Man kann mittels OCR Library den unstrukturierten Text erkennen.	2024-03-31
Construction	Daten aus Scan (zugeschnittenes Bild) sollen strukturiert extrahiert werden, mit Produktname, Menge und Preis.	2024-04-21
Construction	Einmaliges Korrigieren der Daten nach dem Scan funktioniert. Danach können Daten nicht mehr bearbeitet werden.	2024-05-05
Construction	Die ausgelesenen bzw. korrigierten Daten können strukturiert in einer Datenbank persistiert werden.	2024-05-19
Construction	Persistierte Daten Kassenbelege können jederzeit angeschaut werden und die Preisänderungen sollen angezeigt werden, falls vorhanden.	2024-06-02
Transition	Dokumentation und weitere Dokumente wie Plakat, Abstracts, etc. bereitstellen.	2024-06-14

Tabelle 12.4: Meilensteine

12.6 Risikoanalyse

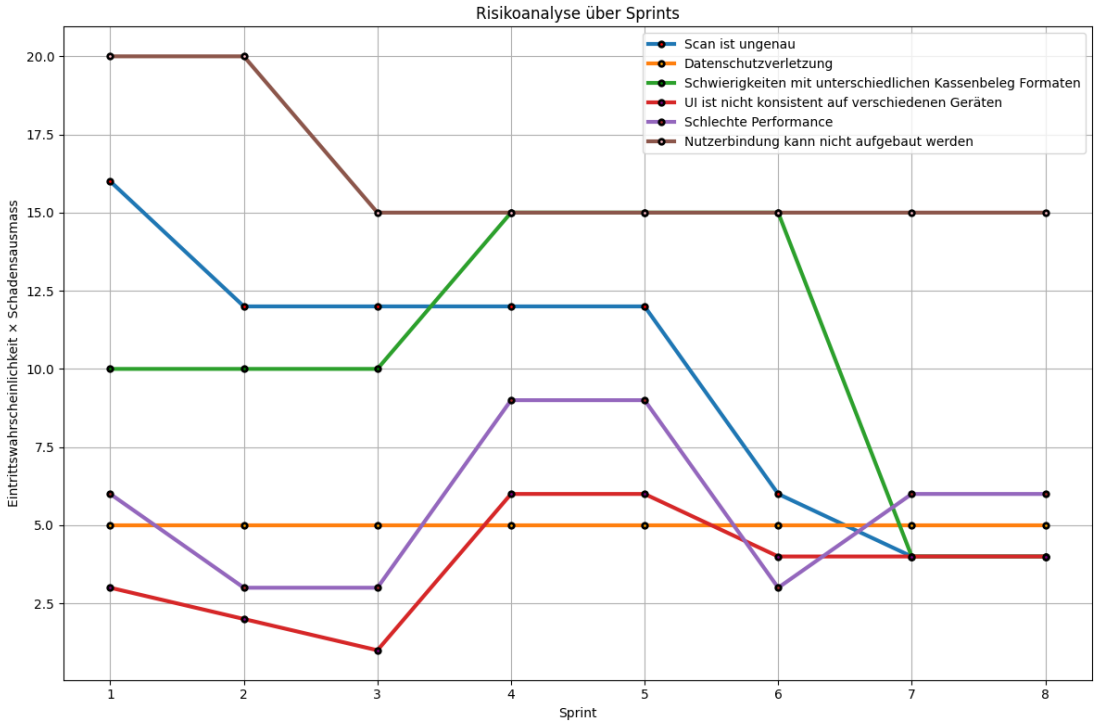


Abbildung 12.1: Risiken

12.6.1 Protokoll

Genauigkeit des Scans

Kernfunktionalität der App hängt davon ab, wie genau der Kassenzettel gescannt wird. Man benötigt Produktname und Preis. Ein ungenauer Scan führt zu falschen Berechnungen.

Eintrittswahrscheinlichkeit	2 von 5
Schadensausmass	2 von 5
Auswirkung	Preise und Namen werden fehlerhaft erfasst / berechnet und dadurch wird Statistik verfälscht.
Minimierung	Nach dem Scan werden Daten angezeigt und Benutzer kann diese manuell bearbeiten und korrigieren.

Tabelle 12.5: Genauigkeit des Scans

- Update 2024-03-06: Es wird definitiv eine Möglichkeit geben, damit der Benutzer

nach dem Scan die Daten einmalig manuell bearbeiten kann gemäss 5.7.

- Update 2024-03-20: Man weiss zu diesem Zeitpunkt, dass das Erkennen der relevanten Bereiche (Preis und Produktname) technisch umsetzbar ist. Geeignete Tools sind generelle AI Modelle wie ChatGPT, Gemini oder Claude. Diese können Bilder entgegennehmen und die relevanten Bereiche darauf erkennen. Es gibt noch die Möglichkeit, ein spezifisches Modell zu trainieren, welches automatisch die relevanten Bereiche erkennt und die Textzeichenerkennung durchführt. Ein solches Modell ist bspw. EasyOCR, welches neben einem trainierbaren OCR Modell ein Detection Modell anbietet (um die relevanten Bereiche zu erkennen).

- Man hat festgestellt, dass eine homogene Beleuchtung sehr wichtig ist, um ein gutes Scan-Resultat zu erhalten. Es lässt sich eine Funktion in der Kameravorschau implementieren, welche den Benutzer anweist, die Aufnahme zu wiederholen, wenn die Beleuchtung schlecht ist. Das führt zu weniger Fehlern und weniger manuellen Korrekturen. Daher wurde das Schadensausmass auf von 3 auf 2 minimiert.

Datenschutz und Sicherheit

Man möchte nicht, dass andere Benutzer sehen, was man kauft. Daten sollen nicht einem Benutzer zugeordnet werden können. Falls es ein Datenleck gibt, sollte man sich ggf. über Rechtliches machen.

Eintrittswahrscheinlichkeit	1 von 5
Schadensausmass	5 von 5
Auswirkung	Rechtliche Konsequenzen können eintreten.
Minimierung	Keine - App wird während der Arbeit nicht veröffentlicht.

Tabelle 12.6: Datenschutz und Sicherheit

- Update 2024-03-06: Die App wird während der Arbeit nicht veröffentlicht, daher werden rechtliche Punkte nicht beachtet.

- Update 2024-06-04: Momentan verwendet die App die Gemini API, um die Daten aus dem Bild zu extrahieren. Diese Daten werden für die Verbesserung von Gemini benötigt. Das ist lediglich beim Free Tier der Fall und dieser ist deshalb in Europa nicht nutzbar (nur über VPN). Die bezahlte Version verwendet keine Nutzerdaten für die Verbesserung von Gemini und ist konform mit der DSGVO und kann deshalb auch in Europa genutzt werden. Da die App nicht veröffentlicht wird, wurde dieses Risiko nicht erhöht.

Unterschiede in den Formaten der Kassenbelege

Verschiedene Läden haben verschiedene Formate für Ihre Kassenbelege. Die App muss mit den einzelnen Unterschieden gut umgehen könnten. Wichtig ist, dass auf dem Beleg

der Produktname und der Preis angegeben sind. Das sollte zumindest für die etablierten Läden (Migros, Coop, Spar, Aldi, Lidl etc.) gut funktionieren. Andere Läden (bspw. Dönerbude), welche Benutzer ggf. selbst hinzufügen können, sind momentan weniger wichtig.

Eintrittswahrscheinlichkeit	2 von 5
Schadensausmass	2 von 5
Auswirkung	Produkte können nicht richtig erfasst werden, weil bspw. der Preis bei Anbieter A auf der linken Seite ist und bei Anbieter B auf der rechten Seite.
Minimierung	Eventuell verschiedene Datenstrukturen für verschiedene Beleg-Formate verwenden, oder Verarbeitung an einen externen Dienst auslagern.

Tabelle 12.7: Unterschiede in den Formaten der Kassenbelege

- Update 2024-04-24: Schadensausmass wurde von 2 auf 3 erhöht, weil beim Parsen des Textes erkannt wurde, dass Zeilenumbrüche in Belegen sehr problematisch sind, weil sich dann die Produktpreise um eine Zeile nach unten verschieben und überall wieder manuell korrigiert werden müssen. Zahlen werden sehr gut erkannt, Texte werden zum allergrössten Teil gut erkannt.
- Update 2024-06-04: Eintrittswahrscheinlichkeit wurde von 5 auf 2 gesenkt und Schadensausmass wurde von 3 auf 2 gesenkt. Momentan wird das Bild an Gemini gesendet, um die Daten aus dem Bild zu extrahieren, was gut funktioniert mit verschiedenen Beleg-Formaten. Gemini erkennt zusammenhängende Zeilenumbrüche besser, aber es kann trotzdem noch zu Fehlern kommen. Insgesamt ist die Fehlerrate diesbezüglich kleiner und somit verringert sich die Anzahl manueller Korrekturen, die man machen muss, was das Schadensausmass mildert.

UI Konsistenz auf verschiedenen Geräten

Das UI sollte auf möglichst vielen verschiedenen Android-Geräten gleich aussehen. Da es viele unterschiedliche Grössen gibt, sind absolute Werte eventuell weniger gut geeignet.

Eintrittswahrscheinlichkeit	2 von 5
Schadensausmass	2 von 5
Auswirkung	Kann Benutzererlebnis beeinträchtigen.
Minimierung	Tests mit mehreren echten und virtuellen Geräten durchführen.

Tabelle 12.8: UI Konsistenz auf verschiedenen Geräten

- Update 2024-02-23: Prüfen, ob ein Multiplatform Framework wie Flutter oder MAUI verwendet werden kann.

- Update 2024-04-03: Man verwendet nicht die Systemkamera-App, sondern erstellt eine eigene Kamera View. Das wurde bereits gemacht und mit zwei verschiedenen Android Geräten getestet (Samsung S und A Reihe).
- Update 2024-04-24: Beim Implementieren der Kamera View ist aufgefallen, dass es schwer ist, diese im Vollbild Modus zu implementieren, weil verschiedene Geräte verschiedene Seitenverhältnisse haben. Beim Anpassen des Seitenverhältnisses wird die CameraPreview je nach Gerät verzerrt. Es wurde noch keine Lösung für dieses Problem gefunden. Deshalb wurde die Eintrittswahrscheinlichkeit von 1 auf 3 erhöht und das Schadensausmass wurde von 1 auf 2 erhöht, weil das UI dadurch auf verschiedenen Geräten nicht ganz konsistent ist.
- Update 2024-03-20: Die Kamera Preview Page wurde angepasst, sodass sie konsistent auf Geräten mit verschiedenen Seitenverhältnissen ist.

Schlechte Performance

Eintrittswahrscheinlichkeit	2 von 5
Schadensausmass	3 von 5
Auswirkung	Kann die Benutzer-Erfahrung beeinträchtigen oder Benutzung verunmöglichen
Minimierung	Nur Android Versionen zulassen, bei denen wir wissen, dass man die App vernünftig verwenden kann.

Tabelle 12.9: Schlechte Performance

- Update 2024-03-06: Google ML Kit wurde mit altem Gerät (HTC One M7 Android 5) getestet und die Texterkennung hat gut funktioniert.
- Update 2024-04-24: Das Erkennen des Texts mittels OCR Technologie ist sehr schnell, auch auf alten Geräten mit schwacher Hardware. Was eher problematisch sein könnte, ist das Preprocessing des Bildes, bevor der Text extrahiert wird. Das Preprocessing soll dem OCR-Scanner helfen, den Text auf dem Bild besser zu erkennen. Einige Preprocessing Schritte dauern lange und funktionieren noch nicht wie gewünscht, deshalb wurde die Eintrittswahrscheinlichkeit von 1 auf 3 erhöht.
- Update 2024-04-30: Preprocessing wurde entfernt, weil die Performance auf dem Smartphone eher schlecht ist. Geeigneter wäre ein Server, der diese Aufgabe übernimmt. Ein weiteres Problem ist, dass der Output des Preprocessings (mit dem gleichen Bild) je nach Gerät anders aussehen kann. Ein gutes Bild benötigt nicht zwingend Preprocessing, da das Resultat meistens gut genug ist.
- Update 2024-06-04: Momentan wird das Bild an Google Gemini gesendet, um die relevanten Informationen aus dem Beleg zu extrahieren. Vorher wurde das

mit einer Regex gemacht, welche allerdings nur für ein bestimmtes Format funktioniert. Dafür war es deutlich schneller. Gemini kann besser mit verschiedenen Beleg-Formaten umgehen, aber ist langsamer (Verarbeitung dauert 3-5 Sekunden). Deshalb wurde die Eintrittswahrscheinlichkeit von 1 auf 2 erhöht.

Nutzerbindung kann nicht aufgebaut werden

Der Benutzer sollte nicht das Interesse an der App verlieren. Der Benutzer muss einen Grund haben, die App zu verwenden, also einen persönlichen Vorteil. Das kann mit einem Punktesystem oder sonstigem Belohnungssystem erreicht werden.

Eintrittswahrscheinlichkeit	3 von 5
Schadensausmass	5 von 5
Auswirkung	Benutzer sehen keinen Sinn darin, die App zu verwenden. Das führt zu schlechten Bewertungen und erschwert den Aufbau einer Nutzerbasis.
Minimierung	End to End Tests mit Benutzern durchführen und prüfen, welche Vorteile die App einem Benutzer persönlich bringt. Eventuell eine Art Punktesystem einführen oder aufzeigen, wie viel der Benutzer sparen kann, wenn er bspw. ein teures Produkt durch ähnliches aber günstigeres ersetzt etc.

Tabelle 12.10: Nutzerbindung kann nicht aufgebaut werden

- Update 2024-03-18: Anhand einer Umfrage und Papier UI-Tests wurde festgestellt, dass ein Bedarf an der App besteht und dass das erarbeitete UI ansprechend ist. Daher wurde die Eintrittswahrscheinlichkeit von 5 auf 3 gesenkt.

Kapitel 13

Anhang

13.1 Eigenständigkeitserklärung

OST
Ostschweizer
Fachhochschule

Erklärung zur Urheberschaft

Erklärung	Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich trage die Verantwortung für die Qualität des Textes sowie die Auswahl aller Inhalte und habe sichergestellt, dass Informationen und Argumente mit geeigneten wissenschaftlichen Quellen belegt bzw. gestützt werden. Die aus fremden Quellen übernommenen Texte, Gedankengänge, Konzepte, Grafiken usw. in meinen Ausführungen habe ich als solche eindeutig gekennzeichnet und mit vollständigen Verweisen auf die jeweilige Quelle versehen. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.
KI-Einsatz ohne Kennzeichnungspflicht	Ich bin mir bewusst, dass die Nutzung maschinell generierter Texte keine Garantie für die Qualität von Inhalten und Text gewährleistet. Ich versichere daher, dass ich mich textgenerierender KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Textpassagen vollumfänglich selbst. Ich versichere, dass ich keine KI-Schreibwerkzeuge verwendet habe, deren Nutzung der Prüfer / die Prüferin explizit schriftlich ausgeschlossen hat.
Verstoss	Mir ist bekannt, dass ein Verstoss gegen die genannten Punkte prüfungsrechtliche Konsequenzen haben und dazu führen kann, dass die Prüfungsleistung mit “nicht ausreichend” bzw. „nicht bestanden“ bewertet wird.
Ort/Datum	Rapperswil, 9. Juni 2024
Unterschrift Verfasser/ Verfasserin	Fabian Freitag Joisp Di Benedetto

Abbildung 13.1: Die unterschriebene Eigenständigkeitserklärung

13.2 Nutzungsrecht

OST
Ostschweizer
Fachhochschule

Vereinbarung

Ohne anderslautende Vereinbarungen stehen die Schutzrechte und das Know-how an der Studienarbeit oder Bachelorarbeit (nachfolgend ‚Arbeit' genannt) und an der in diesem Rahmen geschaffenen Güter, wie Software, sowohl dem Rechtsträger der OST Ostschweizer Fachhochschule, dem für die Arbeit verantwortlichen Professoren sowie dem Verfasser der Arbeit resp. Entwickler der in diesem Rahmen geschaffenen Güter, wie Software, zu.

Die genannten Parteien übertragen sich gegenseitig nicht exklusiv, jedoch unentgeltlich, weltweit, sachlich und zeitlich unbeschränkt die jeweiligen Schutzrechte und das Know-how an der Arbeit und an der in diesem Rahmen geschaffenen Güter, wie Software, einschliesslich dem Recht zur Weiterübertragung, ab. Entsprechend steht es jeder Partei zu, sämtliche Schutzrechte an der Arbeit resp. an der in diesem Rahmen geschaffenen Güter, wie Software, beliebig weltweit, zeitlich und sachlich unbeschränkt zu verwerten. Darunter fällt namentlich aber nicht abschliessend das Recht zur Lizenzierung in jeder Art, Umfang und Form, das Recht zur Bearbeitung und damit zur Nutzung z. B. der Software oder Komponenten hiervon als Grundlage eines neuen schutzfähigen Guts. Die Parteien erklären sich gegenseitig den Verzicht auf Namensnennung bei der Verwertung der Schutzrechte und des Know-how durch eine oder mehrere Parteien gemeinsam und stimmen namentlich zu, dass jede Partei allein unter ihrem eigenen Namen die Schutzrechte resp. das Know-how verwertet. Die vorliegende gegenseitige unentgeltliche Übertragung der Schutzrechte resp. des Know-how bezieht sich auch auf Verwertungsarten, welche heute noch nicht bekannt sind.

Rapperswil, den 10.06.2024 .....
Die Studentin/der Student

Rapperswil, den 11.06.2024 Markus Stolze .....
Der Betreuer / die Betreuerin

Abbildung 13.2: Die unterschriebene Nutzungsrechtvereinbarung

13.3 Einverständniserklärung EPrint

OST
Ostschweizer
Fachhochschule

Einverständniserklärung Publikation auf eprints.ost.ch

○ SA

☒ BA

Titel der Arbeit:	Einkaufshelfer Android App
Team:	Josip Di Benedetto, Fabian Freitag
Betreuer:	Prof. Dr. Markus Stolze

Wir sind mit der Publikation unserer Arbeit auf eprints.ost.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.
Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.ost.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 10.06.2024

Name(n)	Unterschrift(en)
Josip Di Benedetto	
Fabian Freitag	

Abbildung 13.3: Die unterschriebene Einverständniserklärung für den EPrint

13.4 Lizenztexte

13.4.1 MIT

MIT License

Copyright (c) [year] [fullname]

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

13.4.2 Apache-2

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of

this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort

(including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

13.4.3 BSD-3-Clause

Copyright <YEAR><COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

13.5 Testprotokolle

13.5.1 Erste Papier UI-Tests

Benutzer scannt Kassenbeleg ein und speichert diesen in einem Verzeichnis - Benutzer LT

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Tippt auf Kamera-Button.	OK	-	1
Platziert Kamera über Kassenzettel und erfasst Foto	OK	-	2
Markiert Bereich mit Produktnamen, Mengen und Preisen	OK	-	3
Bestätigt markierte Auswahl mit OK	OK	-	3
Wird auf Kamera Preview geleitet, wo zweites Bild aufgenommen werden kann	OK	War etwas verwirrt, warum wieder auf Kameraansicht, hat es aber schnell verstanden, wegen der "Vorschau unten links	4
Bestätigt Scan durch Antippen des Hakens	OK	-	4
Wird auf Listenansicht mit gescannten Produkten geleitet und tippt auf "weiter"	OK	-	6
Bestätigt in der Listenansicht, dass alle Daten korrekt sind	OK	Anmerkung: Daten sollten auch später bearbeitbar sein	8
Belässt Standardnamen und wählt Verzeichnis aus	OK	-	10
Speichert Scan mit Standardnamen und ausgewähltem Verzeichnis	OK	-	9

Tabelle 13.1: scannt-beleg-ein-lt

Benutzer löscht den gescannten Kassenbeleg mit dem Namen lorem_ipsum - Benutzer LT

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Startseite	OK	-	13
Tippt bei lorem_ipsum auf die drei Punkte	OK	-	12
Wählt im Drop-down-Menü löschen	OK	-	12
Bestätigt Löschvorgang	OK	Anmerkung: Nicht klar, warum das benötigt wird, vorgang ist aber klar	14

Tabelle 13.2: löscht-beleg-lt

Benutzer zeigt die Preisänderungen des Migros Belegs mit dem Namen 2024-03-01 - Benutzer LT

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Belege	OK	Macht nichts, war bereits auf Belege	16
Klappt die Spalte für Migros auf und tippt auf Beleg mit Namen 2024-03-01	OK	-	16
Sieht alle Preisänderungen der Produkte in Prozent	OK	Anmerkung: Gut markiert, welche Preise rauf bzw. runter sind	18

Tabelle 13.3: zeigt-beleg-preisänderungen-lt

Benutzer zeigt Preisverlauf für Cola 500ml an - Benutzer LT

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Produkte	OK	-	16
Sucht Eintrag für Cola 500ml	OK	-	19
Tippt auf Cola 500ml und sieht den Preisverlauf	OK	-	19
Sieht alle Preise für Cola 500ml	OK	-	20

Tabelle 13.4: zeigt-alle-preisänderungen-lt

Benutzer scannt Kassenbeleg ein und speichert diesen in einem Verzeichnis - Benutzer KF

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Tippt auf Kamera-Button.	OK	-	1
Platziert Kamera über Kassenzettel und erfasst Foto	OK	-	2
Markiert Bereich mit Produktnamen, Mengen und Preisen	Tippt direkt auf OK	Nicht klar, was Cropping Markierung bedeutet	3
Bestätigt markierte Auswahl mit OK	OK	Siehe oben	3
Wird auf Kamera Preview geleitet, wo zweites Bild aufgenommen werden kann	n. OK	War etwas verwirrt, warum wieder auf Kameransicht, erst nach Bemerkung klar	4
Bestätigt Scan durch Antippen des Hakens	OK	-	4
Wird auf Listenansicht mit gescannten Produkten geleitet und tippt auf "weiter"	OK	-	6
Bestätigt in der Listenansicht, dass alle Daten korrekt sind	OK	-	8
Belässt Standardnamen und wählt Verzeichnis aus	Direkt auf fertig getippt	Nicht offensichtlich, dass Verzeichnis gewählt werden muss	10
Speichert Scan mit Standardnamen und ausgewähltem Verzeichnis	OK	-	9

Tabelle 13.5: scannt-beleg-ein-kf

Benutzer löscht den gescannten Kassenbeleg mit dem Namen lorem_ipsum - Benutzer KF

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Startseite	OK	-	13
Tippt bei lorem_ipsum auf die drei Punkte	Hat auf Beleg selbst getipp	Hat nicht direkt 3 Punkte Menu erkannt	12
Wählt im Drop-down-Menü löschen	OK	-	12
Bestätigt Löschvorgang	OK	-	14

Tabelle 13.6: löscht-beleg-kf

Benutzer zeigt die Preisänderungen des Migros Belegs mit dem Namen 2024-03-01 - Benutzer KF

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Belege	OK	War bereits auf Belege	16
Klappt die Spalte für Migros auf und tippt auf Beleg mit Namen 2024-03-01	OK	-	16
Sieht alle Preisänderungen der Produkte in Prozent	OK	-	18

Tabelle 13.7: zeigt-beleg-preisänderungen-kf

Benutzer zeigt Preisverlauf für Cola 500ml an - Benutzer KF

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Produkte	OK	-	16
Sucht Eintrag für Cola 500ml	OK	-	19
Tippt auf Cola 500ml und sieht den Preisverlauf	OK	-	19
Sieht alle Preise für Cola 500ml	OK	-	20

Tabelle 13.8: zeigt-alle-preisänderungen-kf

Benutzer scannt Kassenbeleg ein und speichert diesen in einem Verzeichnis - Benutzer DD

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Tippt auf Kamera-Button.	OK	-	1
Platziert Kamera über Kassenzettel und erfasst Foto	OK	-	2
Markiert Bereich mit Produktnamen, Mengen und Preisen	OK	-	3
Bestätigt markierte Auswahl mit OK	OK	-	3
Wird auf Kamera Preview geleitet, wo zweites Bild aufgenommen werden kann	OK	Benutzer erkennt nicht, weshalb er nochmals ein Foto aufnehmen kann	4
Bestätigt Scan durch Antippen des Hakens	OK	-	4
Wird auf Listenansicht mit gescannten Produkten geleitet und tippt auf "weiter"	OK	Benutzer merkt nicht, dass er die Listeneinträge nach dem Scan bearbeiten kann	6
Bestätigt in der Listenansicht, dass alle Daten korrekt sind	OK	-	8
Belässt Standardnamen und wählt Verzeichnis aus	OK	Benutzer merkt nicht, dass Standardname geändert werden kann	10
Speichert Scan mit Standardnamen und ausgewähltem Verzeichnis	OK	-	9

Tabelle 13.9: scannt-beleg-ein-dd

Benutzer löscht den gescannten Kassenbeleg mit dem Namen lorem_ipsum - Benutzer DD

Erwartete Aktion	Tatsächliche Aktion	Notiz	Screen Nr.
Geht auf Startseite	OK	-	13
Tippt bei lorem_ipsum auf die drei Punkte	OK	-	12
Wählt im Drop-down-Menü löschen	OK	-	12
Bestätigt Löschvorgang	OK	Benutzer ist verwirrt, dass der Löschvorgang nur auf der Startseite gemacht werden kann	14

Tabelle 13.10: löscht-beleg-dd

Benutzer zeigt die Preisänderungen des Migros Belegs mit dem Namen 2024-03-01 - Benutzer DD

Erwartete Aktion	Tatsächliche Aktion	Notiz	Screen Nr.
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Belege	OK	Für den Benutzer war nicht direkt offensichtlich, dass er den Schieber auf Belege stellen soll	16
Klappt die Spalte für Migros auf und tippt auf Beleg mit Namen 2024-03-01	OK	-	16
Sieht alle Preisänderungen der Produkte in Prozent	OK	Benutzer fragte, ob das die Preisänderung ab dem ersten oder ab dem letzten Einkauf ist	18

Tabelle 13.11: zeigt-beleg-preisänderungen-dd

Benutzer zeigt Preisverlauf für Cola 500ml an - Benutzer DD

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Produkte	OK	Benutzer ist verwirrt und fragt, ob die Liste nur bestimmte Artikel oder alle Artikel von jedem Laden anzeigen	16
Sucht Eintrag für Cola 500ml	OK	-	19
Tippt auf Cola 500ml und sieht den Preisverlauf	OK	-	19
Sieht alle Preise für Cola 500ml	OK	Für den Benutzer war ab diesem Punkt offensichtlich, dass es sich um das gleiche Produkt handelt, welches in jedem Laden gekauft wurde	20

Tabelle 13.12: zeigt-alle-preisänderungen-dd

Benutzer scannt Kassenbeleg ein und speichert diesen in einem Verzeichnis - Benutzer KP

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Tippt auf Kamera-Button.	OK	-	1
Platziert Kamera über Kassenzettel und erfasst Foto	OK	-	2
Markiert Bereich mit Produktnamen, Mengen und Preisen	OK	-	3
Bestätigt markierte Auswahl mit OK	OK	-	3
Wird auf Kamera Preview geleitet, wo zweites Bild aufgenommen werden kann	OK	Benutzer erkennt nicht, weshalb er nochmals ein Foto aufnehmen kann	4
Bestätigt Scan durch Antippen des Hakens	OK	-	4
Wird auf Listenansicht mit gescannten Produkten geleitet und tippt auf "weiter"	OK	-	6
Bestätigt in der Listenansicht, dass alle Daten korrekt sind	OK	-	8
Belässt Standardnamen und wählt Verzeichnis aus	OK	-	10
Speichert Scan mit Standardnamen und ausgewähltem Verzeichnis	OK	-	9

Tabelle 13.13: scannt-beleg-ein-kp

Benutzer löscht den gescannten Kassenbeleg mit dem Namen lorem_ipsum - Benutzer KP

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Startseite	OK	-	13
Tippt bei lorem_ipsum auf die drei Punkte	OK	-	12
Wählt im Drop-down-Menü löschen	OK	-	12
Bestätigt Löschvorgang	OK	Benutzer ist verwirrt, dass der Löschvorgang nur auf der Startseite gemacht werden kann	14

Tabelle 13.14: löscht-beleg-kp

Benutzer zeigt die Preisänderungen des Migros Belegs mit dem Namen 2024-03-01 - Benutzer KP

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**	**Screen Nr.**
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Belege	OK	-	16
Klappt die Spalte für Migros auf und tippt auf Beleg mit Namen 2024-03-01	OK	-	16
Sieht alle Preisänderungen der Produkte in Prozent	OK	Benutzer fragte, ob das die Preisänderung ab dem ersten oder ab dem letzten Einkauf ist	18

Tabelle 13.15: zeigt-beleg-preisänderungen-kp

Benutzer zeigt Preisverlauf für Cola 500ml an - Benutzer KP

Erwartete Aktion	Tatsächliche Aktion	Notiz	Screen Nr.
Geht auf Seite mit allen Scans	OK	-	11
Stellt Schieber auf Produkte	OK	-	16
Sucht Eintrag für Cola 500ml	OK	-	19
Tippt auf Cola 500ml und sieht den Preisverlauf	OK	-	19
Sieht alle Preise für Cola 500ml	OK	-	20

Tabelle 13.16: zeigt-alle-preisänderungen-kp

13.5.2 End-User-Tests

Scanne den Kassenbeleg und speichere ihn im relevanten Verzeichnis - Benutzer AR

Erwartete Aktion	Tatsächliche Aktion	Notiz
Tippt auf Kamera Button	OK	-
Macht Foto von Beleg	OK	-
Überprüft erfasste Produkte	OK	-
Ev. Anpassung der Einträge	OK	-
Speichert Beleg in Verzeichnis	OK	-

Tabelle 13.17: Scanne den Kassenbeleg - AR

Zeige, wie sich die Preise der Produkte, die du eben gescannt hast, verändert haben - Benutzer AR

Erwartete Aktion	Tatsächliche Aktion	Notiz
Findet Beleg auf Homescreen oder Alle Belege Seite	NOK	Nicht ganz klar, wo der gescannte Beleg ist
Klappt Rot/Grün markierte Produkte auf	OK	-

Tabelle 13.18: Zeige Preisänderung - AR

Scanne einen Beleg, der nicht unterstützt wird und erkläre die Fehlermeldung - Benutzer AR

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**
Tippt auf Kamera Button	OK	-
Macht Foto von Beleg	OK	-
Versteht die Fehlermeldung	NOK	Versucht es nochmals, Fehlermeldung ist unklar
Navigiert wieder auf den Homescreen	OK	-

Tabelle 13.19: Fehlermeldung - AR

Scanne den Kassenbeleg und speichere ihn im relevanten Verzeichnis - Benutzer LP

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**
Tippt auf Kamera Button	OK	-
Macht Foto von Beleg	OK	Hat zuerst versucht den QR-Code zu scannen
Überprüft erfasste Produkte	OK	-
Ev. Anpassung der Einträge	OK	-
Speichert Beleg in Verzeichnis	OK	-

Tabelle 13.20: Scanne den Kassenbeleg - LP

Zeige, wie sich die Preise der Produkte, die du eben gescannt hast, verändert haben - Benutzer LP

Erwartete Aktion	**Tatsächliche Aktion**	**Notiz**
Findet Beleg auf Homescreen oder Alle Belege Seite	OK	-
Klappt Rot/Grün markierte Produkte auf	OK	-

Tabelle 13.21: Zeige Preisänderung - LP

Scanne einen Beleg, der nicht unterstützt wird und erkläre die Fehlermeldung - Benutzer LP

Erwartete Aktion	Tatsächliche Aktion	Notiz
Tippt auf Kamera Button	OK	-
Macht Foto von Beleg	OK	-
Versteht die Fehlermeldung	NOK	Versucht es nochmals, Fehlermeldung ist unklar
Navigiert wieder auf den Homescreen	OK	-

Tabelle 13.22: Fehlermeldung - LP

13.6 Ergebnisse der Umfrage

13.6.1 Frage 1

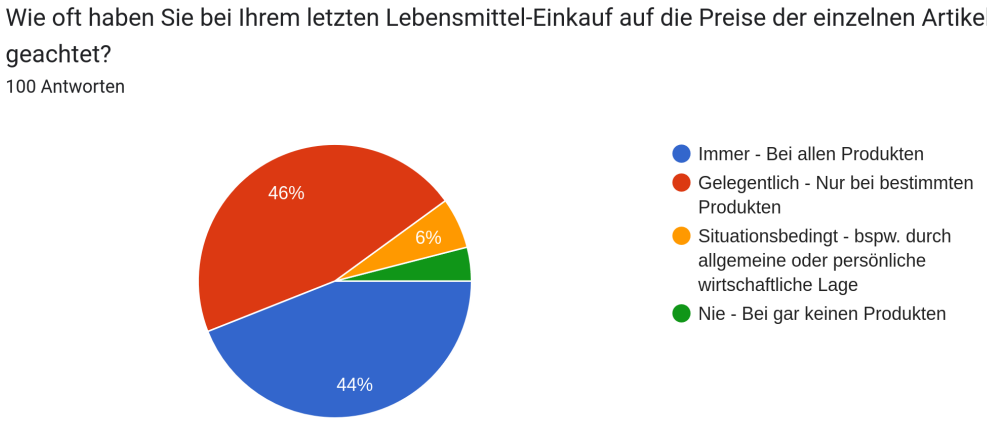


Abbildung 13.4: Antworten-Frage-1

13.6.2 Frage 2

Abbildung 13.5: Antworten-Frage-2

13.6.3 Frage 3

Abbildung 13.6: Antworten-Frage-3

13.6.4 Frage 4

Für welchen Zweck bewahren Sie Kassenbelege auf? Beispielsweise für persönliche Buchhaltung oder Ähnliches?

- Abrechnung in der WG

- Damit ich sehe, ob die Abrechnung korrekt war
- Damit ich weiss, wofür ich Geld ausgebe
- Weil ich es in Excel eintragen möchte
- Für den Fall, dass ich etwas umtauschen möchte
- Für persönliche Buchhaltung und Nachweis, dass ich einen Artikel bezahlt habe
- Um ein Gefühl zu haben, wie viel Geld ich wofür ausgebe und als Zahlnachweis
- Um Kosten zwischen WG Mitgliedern aufzuteilen
- Für persönliche Buchhaltung
- Um einen Überblick über meine Ausgaben zu haben. Ich bewahre diese nur digital auf.
- Für persönliche Buchhaltung
- Kassenbeleg wird automatisch in der App gespeichert

13.6.5 Frage 5

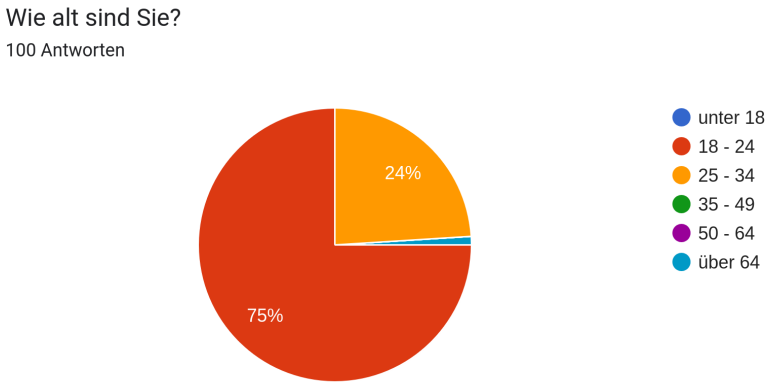


Abbildung 13.7: Antworten-Frage-5

13.7 Code Metrics

Notiz: Die angezeigten Werte beziehen sich auf die Ordner und fassen alle Files in diesem zusammen.

All files

Directory	Cyclomatic complexity	Source lines of code	Maintainability index	Number of Arguments	Maximum Nesting	Technical Debt
lib	8	77	71	0	2	0,0
lib\config\theme	2	8	80	0	1	0,0
lib\core\constants	0	0	0	0	0	0,0
lib\core\data	3	0	100	0	0	0,0
lib\core\data\dao	21	0	100	1	0	0,0
lib\core\data\mapper	9	16	90	1	0	0,0
lib\core\data\models	15	15	95	1	0	0,0
lib\core\data\repository	39	91	80	1	1	0,0
lib\core\domain\entities	3	0	100	0	0	0,0
lib\core\domain\repository	20	0	100	1	0	0,0
lib\core\domain\services	4	9	84	0	1	0,0
lib\core\domain\usecases	21	99	69	1	2	0,0
lib\core\error	3	29	79	1	1	0,0
lib\core\error\exceptions	10	5	100	0	0	0,0
lib\core\utils	3	10	86	1	1	0,0
lib\presentation	1	8	70	0	0	0,0
lib\presentation\camera	16	58	79	1	1	0,0
lib\presentation\camera\blocs	15	41	95	0	1	0,0
lib\presentation\camera\widgets	6	39	77	1	0	0,0
lib\presentation\audit_scan	9	53	82	0	1	0,0
lib\presentation\audit_scan\blocs	5	6	92	0	1	0,0
lib\presentation\audit_scan\widgets	22	91	83	1	1	0,0
lib\presentation\home	9	26	88	0	1	0,0
lib\presentation\home\blocs	4	4	93	0	1	0,0
lib\presentation\home\widgets	4	36	79	1	1	0,0
lib\presentation\receipts	15	99	77	0	1	0,0
lib\presentation\receipts\blocs	8	18	88	0	1	0,0
lib\presentation\receipts\widgets	18	73	81	0	1	0,0
lib\presentation\utore	9	29	88	0	1	0,0
lib\presentation\utore\blocs	7	29	86	0	1	0,0
lib\presentation\utore\widgets	9	37	84	0	1	0,0

Cyclomatic complexity : **318**

Source lines of code : **1006**

Maintainability index : **83**

Number of Arguments : **0**

Maximum Nesting : **1**

Technical Debt : **0**

Abbildung 13.8: Code Metrics

Quellenverzeichnis

[1] Zheng Huang et al. *ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction*. URL: https://arxiv.org/abs/2103.10213.

[2] anthropic. *Claude API Pricing*. URL: https://www.anthropic.com/api.

[3] baseflow. *Flutter permission handler Package*. URL: https://pub.dev/packages/permission_handler.

[4] Simon Binder. *Drift*. URL: https://pub.dev/packages/drift.

[5] bloclibrary. *Flutter bloc Package*. URL: https://pub.dev/packages/flutter_bloc.

[6] Thomas Burkhart. *Flutter getit Package*. URL: https://pub.dev/packages/get_it.

[7] *Cyclomatic Complexity*. URL: https://en.wikipedia.org/wiki/Cyclomatic_complexity.

[8] flutter. *Flutter camera Package*. URL: https://pub.dev/packages/camera.

[9] *Flutter Dotenv*. URL: https://pub.dev/packages/flutter_dotenv.

[10] flutter-ml. *Google ML Kit Flutter Plugin*. URL: https://pub.dev/packages/google_ml_kit.

[11] Google. *Android Data and File storage*. URL: https://developer.android.com/training/data-storage.

[12] Google. *Android Studio*. URL: https://developer.android.com/studio/intro.

[13] Google. *Flutter*. URL: https://flutter.dev/.

[14] Google. *Gemini API Pricing*. URL: https://ai.google.dev/pricing.

[15] Google. *Google Generative AI*. URL: https://pub.dev/packages/google_generative_ai.

[16] Google. *Google ML Kit*. URL: https://developers.google.com/ml-kit?hl=de.

[17] Google. *Google ML Kit Texterkennung*. URL: https://developers.google.com/ml-kit/vision/text-recognition/v2?hl=de.

[18] Google. *KameraX-Jetpack*. URL: https://developer.android.com/jetpack/androidx/releases/camera?hl=de.

[19] Google. *VSCode Flutter Plugin*. URL: https://docs.flutter.dev/tools/vs-code.

[20] *ISO / IEC 25010*. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[21] JaidenAI. *EasyOCR*. URL: https://github.com/JaidedAI/EasyOCR.

[22] JaidenAI. *EasyOCR Training*. URL: https://github.com/JaidedAI/EasyOCR/blob/master/trainer/craft/README.md.

[23] Eivind Kjosbakken. *How to generate your synthetic Dataset*. URL: https://www.freecodecamp.org/news/how-to-fine-tune-easyocr-with-a-synthetic-dataset/#how-to-generate-your-synthetic-dataset:~:text=18GB%20in%20size).-,How%20to%20Generate%20your%20Synthetic%20Dataset,-If%20you%20want.

[24] Zhou ZhiYing Loh Zhi Chang. *Robust pre-processing techniques for OCR applications on mobile devices*. URL: https://dl.acm.org/doi/10.1145/1710035.1710095.

[25] lolki3d. *Flutter iamge Package*. URL: https://pub.dev/packages/image.

[26] OnesSplit. *OneSplit*. URL: https://www.onesplit.ai/.

[27] openAI. *ChatGPT API Pricing*. URL: https://openai.com/api/pricing/.

[28] pinch. *Flutter Floor Package*. URL: https://pub.dev/packages/floor.

[29] Tekartik. *SQFLite*. URL: https://pub.dev/packages/sqflite.

[30] Danny Tuppeny. *VSCode Dart Plugin*. URL: https://dartcode.org/.

[31] Muhammad Kamran Malik Zara Nasar Seyd Waqar Jaffry. *Named Entity Recognition and Relation Extraction: State-of-the-Art*. URL: https://dl.acm.org/doi/10.1145/3445965.

Abbildungsverzeichnis

Tabellenverzeichnis