HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

PRAKTIKUMSBERICHT

Herr
Hannes Steiner

Entwicklung einer Dokumenten-Scanner-App

Digitalisierung der Verwaltung an der Hochschule Mittweida

2020

Fakultät **Angewandte Computer- und Biowissenschaften**

PRAKTIKUMSBERICHT

Entwicklung einer Dokumenten-Scanner-App

Digitalisierung der Verwaltung an der Hochschule Mittweida

Autor:
Hannes Steiner

Studiengang:
Softwareentwicklung

Seminargruppe:
IF17wS-B

Matrikelnummer:
46540

Erstprüfer:
Prof. Dr. Mark Ritter

Zweitprüfer:
N.N.

Mittweida, März 2020

Bibliografische Angaben

Steiner, Hannes: Entwicklung einer Dokumenten-Scanner-App, Digitalisierung der Verwaltung an der Hochschule Mittweida, 29 Seiten, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Praktikumsbericht, 2020

Dieses Werk ist urheberrechtlich geschützt.

Referat

In dem vorliegendem Praktikumsbericht...

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

bzw. ..... beziehungsweise, Seite 7

CRUD ..... Das Akronym CRUD steht für Create/Erstellen, Read/Lesen, Update/Aktualisieren und Delete/Löschen und umfasst die vier grundlegenden Operationen persistenter Speicher, Seite 9

Fakultät CB ..... Fakultät für angewandte Computer- und Biowissenschaften, Seite 7

HSMW ..... Hochschule Mittweida - university of applied sience, Seite 3

MVVM ..... Model View ViewModel, Seite 5

OCR ..... optical character recognition, deutsch: optische Zeichen Erkennung und im deutschen Synonym für Texterkennung, Seite 9

1 Einleitung

Digitalisierung wird gängig als Integration von digitaler Technologie in den Alltag verstanden, und soll helfen Zeit einzusparen [1]. Mit diesem Gedanken initiierten die Mitarbeiter Holger Langner und Falk Schmidsberger der Hochschule Mittweida, das Projekt *Memo Space*. Im Zuge dessen sollen kleinere Forschungsergebnisse entstehen, die richtungsweisend für die Digitalisierung der Verwaltung von Lehr- und Forschungseinrichtung sind.

Eine der ersten Ideen ist es, die Arbeit von Klausur-Prüfern zu erleichtern. Diese müssen, nachdem die Klausuren kontrolliert wurden, die Zensur, sowie die Eckdaten der Studenten, in ein digitales Format bringen. Grund dafür ist, dass die Noten in das Notensystem der Einrichtung eingetragen werden müssen.

Im Rahmen eines zwölfwöchigem Forschungspraktikums an der Hochschule Mittweida arbeiteten der Student Tobias Kallauke und der Verfasser, gemeinsam an einer Lösung zur Digitalisierung dieses Arbeitsschrittes.

... Beide arbeiteten zum einem Teil zusammen und auch getrennt an Aufgaben... Sie entwickelten... (Klausuren-Vorlage, App, Server, OCR? evtl.) / Dabei entstanden .../ Der Verfasser programmierte...

Was noch?

2 Hochschule Mittweida

Die Hochschule Mittweida - university of applied sience (HSMW) ist ...

- was gibt es da

- wer leitet die hochschule

- was machen meine betreuer in der hochschule

- aktuelles?

3 Grundlagen

In diesem Kapitel werden wichtige Grundlagen vermittelt...

3.1 Model-View-ViewModel

Das Entwurfsmuster Model View ViewModel (MVVM) entstand bei *Microsoft* im Jahr 2005 mit der *Windows Presentation Foundation* (WPF) und *Silverlight-Technologien*. MVVM verwendet das Konzept eines Schichtmodells und ist eine abstrakte Darstellung einer Benutzeroberfläche, in Form einer Klasse, wie man sie unter objektorientierten Programmiersprachen kennt. Diese Klasse enthält Daten, die auf der Benutzeroberfläche angezeigt werden sollen und Anweisungen, die auf der Benutzeroberfläche aufgerufen werden können. Dieses sogenannte ViewModel, weiß nichts von Views, wie es sonst bei anderen Entwurfsmustern üblich ist, um Daten auf der Benutzeroberfläche anzuzeigen. Stattdessen verwendet eine MVVM-View eine Bindungsfunktion (data binging) zur bidirektionalen Zuordnung von Daten aus dem ViewModel zu den jeweiligen Eigenschaften auf der View. Z. B. Einträge in einer Dropdown-Menü. Aber auch das binden von Daten aus dem Model zu Benutzereingaben durch Maus, Tastatur oder Touch-Screens ist möglich. Beispielsweise kann ein Mausklick eine Anweisung in dem ViewModel auslösen. Diese verändert einen Wert im Model, wodurch die View durch data binding aktualisiert wird.

3.2 Redux.js

... Dieser Container besteht aus verschiedenen States. Diese sind dazu da einer View oder mehrere zusammenhängende Views die benötigten Daten bereit zustellen. Die Daten können allerdings nur ge - https://github.com/reduxjs/redux - https://redux.js.org/introduction/core-concepts - https://redux.js.org/introduction/three-principles

- MVVM

- data binding

- redux state container, actions, unidirectional data flow

- publisher und subsciber

4 Problemstellung

Hochschulmitarbeiter sitzen zum Ende eines Semesters über Tage an der Kontrolle von Klausuren. Diese Aufgabe muss stets mit hoher Konzentration erledigt werden, und lässt sich aber in den meisten Fällen nur schwer durch Maschinen ersetzen. Unter keinen Umständen dürfen bei der Bewertung Fehler vorkommen, was jedoch bei der kognitiven Last der Prüfer immer wieder passiert. Auch nach der Durchsicht der Prüfungsaufgaben ist eine hohe Achtsamkeit wichtig. Denn anschließend wird die Benotung in eine digitale Tabelle überführt. In diese muss die Matrikelnummer, der Vor- und Nachname, sowie die Note des Studenten eingetragen werden. Hier kommt es vor allem bei der Matrikelnummer und der Zensur auf die Richtigkeit jedes Zeichens drauf an.

4.1 Digitalisieren der Klausur-Daten

Für genau diesen Vorgang des Digitalisierens wird eine Lösung gesucht. Die Prüfer sollen so bequem und möglichst zeitsparend diese Aufgabe verrichten, ohne dabei ihre Aufmerksamkeitsspanne zu überlasten. Des Weiteren müssen die Ergebnisse der Prüfungen, sowie die Eckdaten der Studenten in ein geeignetes digitales Format gebracht werden, um es der Notenfreigabe weiterzuleiten. Darüber hinaus empfiehlt es sich, digitale Kopien der Klausuren abzuspeichern, da/um ... (warum genau, sollen die gespeichert werden? meine/unsere Idee Online Klausureneinsicht -> ins Fazit/Ausblick (hat viele ”Probleme”))

4.2 Klausuren-Vorlage

Eine Klausuren-Vorlage bzw. ein Gestaltungsleitfaden für Klausuren der Fakultät *Angewandte Computer- und Biowissenschaften* (Fakultät CB) bietet außerdem die Möglichkeit der Kontrolle des Prüfers an. Genauer ist es durch das vorgegebene Layout der Klausur möglich, einen Teil der Arbeit des Prüfenden auf Fehler zu untersuchen. Auf dem Deckblatt der Klausur ist eine Tabelle, mit drei Zeilen und für jede Aufgabe eine Spalte. In der ersten Zeile befinden sich die Nummern der Aufgaben. In der zweiten, die zu erreichenden Punkte der Aufgabe. Und in der dritten Zeile trägt der Prüfer die erbrachten Punkte des Studenten ein. Unter der Tabelle befindet sich ein Feld für die erreichte Gesamtpunktzahl, sowie ein Feld für die aus den Punkten resultierende Note. Die Punkte pro Aufgabe, die Gesamtpunktzahl und die Zensur stehen in (Korrelation?) Relation zu einander, sodass aus den Punkten der Aufgaben die beiden anderen Felder errechnet werden und mit den Ergebnissen des Prüfers abgeglichen werden könnten. Eine weiteres Merkmal der Klausuren-Vorlage ist ein Feld, für die vom Studenten erreichten Punkte über jeder Aufgabenstellung. Die dortige Angabe sollte mit der, in der

Tabelle auf dem Deckblatt übereinstimmen und bietet somit noch eine weitere Möglichkeit der Kontrolle an.

4.3 Vorlage verbessern

Templates entwerfen, die optimiert für die Digitalisierungs-Prozesse sind ...

4.4 Weitere Anmerkungen

TODO:

Ferner soll bei der Lösung von der Anschaffung neuer Technologie und Geräte abgesehen werden. Grund dafür sind neben den Anschaffungskosten, die Idee, dass das Ergebnis des Forschungsprojekts in weiteren Lehr- und Forschungseinrichtung Anwendung finden sollte.

Was noch?

5 Anforderungen

Es soll eine iOS-App entstehen mit der Klausuren digitalisiert werden können. Genauer müssen die, für die Notenfreigabe relevanten Daten der Klausur, in ein tabellarisches Format gebracht werden. Um unterschiedliche Klausuren oder auch andere Dokumente zu unterstützen, soll das Anlegen von Scan-Vorlagen in der App möglich sein. Hierfür (aber auch zum Einscannen) benötigt die App die Berechtigung für das Kamera-System. Diese muss, bei der ersten Benutzung einmalig erteilt werden.

Die Scan-Vorlagen bestehen aus zwei Komponenten. Die eine sind zugeschnittene Fotos der Seiten eines Dokuments. Und die zweite sind die Regionen auf den Bildern, wo sich die zu digitalisierenden Informationen befinden. Dem Benutzer muss es möglich sein, zuerst die Fotos zu machen und anschließend Regionen auf den Seiten zu markieren. Weiter benötigen die Regionen Namen bzw. Typen, die der Benutzer angeben muss. Die Angabe des Typs ist wichtig, da dadurch eindeutig wird, ob es sich um die Note oder die Matrikelnummer des Studenten handelt. Diese Eindeutigkeit wird nicht nur für die automatische Erstellung der Tabelle benötigt, sondern auch für die Texterkennung.

Optical character recognition (OCR), auf deutsch optische Zeichen Erkennung, soll dazu benutzt werden, die Informationen der Regionen zu digitalisieren. Um die Texterkennung zu verbessern, soll der Typ der Regionen verwendet werden. Damit könne dem OCR mögliche Ergebnisse mitgeteilt oder die Ergebnisse angepasst werden. Beispielsweise bestehen Zensuren immer aus Dezimalzahlen von 1 bis 6 und mit nur einer Nachkommastelle. Wird der Buchstabe O anstelle der Ziffer 0 erkannt, kann so der Fehler korrigiert werden.

Zudem muss die Texterkennung auf dem Gerät selbst statt finden. Allerdings darf die App auch Texterkennung auf externen Servern unterstützen. Hierfür müssen dann die entsprechenden API-Schnittstellen implementiert oder für einen eigenen Server entwickelt werden.

Die digitalisierten Daten, die für die Texterkennung entstandenen Dokumenten-Bilder und die erstellten Scan-Vorlagen, soll außerhalb der App auf einem Server gespeichert werden. Somit ist die Möglichkeit gegeben, die Vorlagen wieder zu verwenden und anderen Benutzern der App zur Verfügung zu stellen. Des Weiteren hat diese zentralen Stelle den Vorteil alle anfallenden Daten zu verwalten, was die Benutzung der App trotz vieler Benutzer vereinfacht.

Damit Synchronität der Daten auf den Geräten und dem Server gewährleistet werden kann, benötig die App hierfür ebenfalls Schnittstellen. Diese sollten den standardmäßigen *CRUD*-Operationen entsprechen.

Da die gesamte Kommunikation über das Internet geschieht, muss das Softwaresystem die üblichen Datenschutz- und Datensicherheit-Richtlinien entsprechen, bzw. implementieren.

6 Konzept der Dokumenten-Scanner-App

Wie in der Problemstellung und Anforderung beschreiben, soll die Dokumenten-Scanner App wichtigen Daten auf dem Deckblatt einer Klausur, wie Vor- und Nachname des Studenten, seine Matrikelnummer sowie die Note erkennen, digitalisieren und in ein für die Notenfreigabe geeignetes Format bringen. Die digitalisierten Daten sollen anschließend an einen Server gesendet werde, wo sie und die beim Einscannen entstandenen Bilder gespeichert werden. Verwendet eine einzuscannende Klausur die Klausuren-Vorlage der Fakultät CB oder ähnliche, die eine Punkteübersicht haben, dann soll es außerdem möglich sein, die Punkte sowie die Note auf Richtigkeit zu überprüfen.

Zur Digitalisierung und Kontrolle der Daten werden Scan-Vorlagen verwendet. Diese muss vor dem Einscannen der ausgefüllten Klausuren erstellt werden, um zu gewährleisten, dass die richtigen Daten digitalisiert werden. Beim Erstellen einer Scan-Vorlage geht man wie folgt vor:

- Zuerst wird jede Seite der Klausur fotografiert. Dabei wird in jedem Bild das Dokument erkannt, vom Hintergrund getrennt oder genauer gesagt ausgeschnitten und perfekt ausgerichtet. Zudem wird der Kontrast erhöht, so dass die Schrift leichter lesbarer wird.
- Anschließend markiert man diejenigen Regionen auf jeder Seite, die zu digitalisieren und/oder kontrollieren sind. Zusätzlich muss jeder Region ein Name und einer der folgenden Typen zugeordnet werden: Unbekannt, Name, Matrikelnummer, Seminargruppe, Punkte und Note. Mithilfe des Typs wird die Texterkennung verbessert und falsche Ergebnisse automatisch korrigiert. Dazu später mehr.
- Des Weiteren kann man aus allen markierten Regionen diese auswählen, die in Relation stehen. Diesen Relationen werden danach noch weitere Eigenschaften zugeteilt, um sie später bei der Kontrolle richtig zu analysieren. So eine Eigenschaft könnte sein, dass es sich um einen Vergleich zwischen zwei in Relation stehenden Regionen handelt. Z. B. ist die eine Region die Zelle in der Punkteübersichts-Tabelle, in der die erreichten Punkte zu Aufgabe 1 rein geschrieben werden sollen. Und die zweite Region die Stelle über der Aufgabenstellung von Aufgabe 1, in der ebenfalls die erreichten Punkte eingetragen werden sollen. Weitere Beispiele für solche Relationen und Eigenschaften sind die Summanden für die erreichte Gesamtpunktzahl und die Summe selbst, oder auch noch die daraus resultierende Note.
- Als letztes speichert man die Vorlage ab, welche dann automatisch an einen Server gesendet wird, so dass andere diese Vorlage ebenfalls nutzen können.

Nach der Erstellung einer Vorlage kann das Dokument digitalisiert werden. Dazu scannt man die Seiten in derselben Reihenfolge, wie in der Scan-Vorlage ein. Die App digita-

lisiert mithilfe von OCR den Inhalt, in den vordefinierten Regionen. Der Typ der Region nimmt nun Einfluss auf das Ergebnis. Durch ihn wird während der Worterkennungsphase eine Liste an vordefinierten Ergebnissen eingespeist. Diese Liste hat Vorrang vor dem Standard-Lexikon, welches verwendet wird. Ein Beispiel für solch eine Liste könnten, wie in den Anforderungen beschrieben, alle möglichen Noten sein. Auch vorstellbar ist, dass alle Seminargruppen oder Namen von Personen, die an der Klausur teilgenommen haben, dort verwendet werden.

Weiter wäre eine zusätzliche Möglichkeit der Korrektur erdenklich. Angenommen, jemand verschreibt sich bei seiner Seminargruppe und vergisst ein Zeichen. Die Texterkennung erkennt zwar jedes Zeichen richtig, jedoch wäre es für die Notenfreigabe eine nicht korrekte Seminargruppe. Deshalb könnte man auch im Nachhinein das Ergebnis mit einer Seminargruppe ersetzt, die die größte Übereinstimmung mit dem erkannten Wort hat. Dieses Vorgehen kann man dadurch verbessern, wenn man dafür nur die Seminargruppen-Bezeichnungen verwendet, die auch tatsächlich die Klausur geschrieben haben. Ähnliches gilt auch für die Namen oder Matrikelnummern der Studenten.

Um wirklich sicher zu gehen, ist es aber im Anschluss an die Texterkennung möglich die Resultate noch einmal zu überprüfen und zu korrigieren, bevor es an den Server zum Abspeichern gesendet wird. Dazu wird außerdem die confidence des OCR bei jeder Region angezeigt. Diese sagt aus, wie sicher sich die Texterkennung ist, das das Ergebnis stimmt. Jedoch kann es auch hier falsche Positiv-Ergebnisse geben, weshalb auch das kein perfekter Indikator für die Richtigkeit ist.

Ein wichtiger Vorteil einer App gegenüber anderen Lösungen ist, dass so gut wie jeder Mitarbeiter an einer Lehr- und Forschungseinrichtung ein eigenes oder Zugang zu einem Smartphone oder Tablet hat.

... noch mehr? (Auf Tobias warten...)

- Noch mehr zu OCR sagen: was es ist, wie es geht, komplett eigenes Kapitel?

7 Entwicklung des ersten Prototyps

Vor Beginn des Praktikums saßen die Praktikum-Betreuer Falk Schmidsberger, Robert Manthey der Hochschule Mittweida, mit Tobias Kallauke und mir zusammen. Wir diskutierten, welches Thema für das Praktikum geeignet ist, wie eine Art Durchführbarkeits- / Machbarkeitsstudie und entschieden uns für den Dokumenten-Scanner. Da ich privat Erfahrung in *Swift* sowie *iOS*-Entwicklung gesammelt habe, sollte ich damit anfangen den Scanner als *iOS*-App zu entwickeln. Wir machten uns auch direkt noch Gedanken über die Anforderungen an die App und definierten, wie das zukünftige Software-System mit Server und Backend aussehen könnte.

7.1 Planung und Vorbereitung

In der aller ersten Woche habe ich mich hauptsächlich mit der Problemstellung auseinander gesetzt, Ideen entwickelt, Problemanalyse betrieben und einen kleinen Prototypen entwickelt. Ich begann damit die Dokumentation von *Vision*, *VisionKit* und *PhotoKit* von *Apple* an zu schauen. Danach hatte ich einen guten Überblick darüber, was ich noch zu entwickeln habe und was mir Frameworks abnehmen können. Zum einen war das Erkennen und gerade ziehen von Dokumenten in Echtzeit in der Kamera, sowie die Texterkennung auf Bildern schon in den Frameworks implementiert. Nebenbei entdeckte ich eine Beispiel-App von *Apple* [2], in der es um Erkennen von Objekten in Standbildern ging. Diese schaute ich mir an, um das Vision Framework genauer zu verstehen, sodass ich mich gut eingearbeitet habe.

Bei meiner Recherche stellte ich fest, dass die App nur Geräte mit iOS 13.0 oder neuer unterstützen werden kann. Grund dafür sind die *Apple* Frameworks *SwiftUI* und *VisionKit*. *SwiftUI* bietet die Möglichkeit, Benutzeroberflächen für alle *Apple*-Plattformen in *Swift* zu erstellen. Die deklarative *Swift*-Syntax ist für mich einfach zu lesen und schnell zu schreiben, so dass es möglich ist, die App in wenigen Wochen für iPhone und iPad zu schreiben. Als Alternative gibt es noch *UIKit* oder auch *AppKit*, die unter alle iOS Versionen funktioniere. Diese Frameworks sind allerdings nicht deklarativ sodass, Views sowohl im Code als auch in Interface-Dateien getrennt voneinander erstellt und konfiguriert werden müssen [4]. Dadurch dauert die Entwicklung einer App, im Gegensatz zu SwiftUI deutlich länger. *VisionKit* dagegen, ist das Framework zum Scannen der Dokumente. Auch hierfür gibt es eine Alternative. Das Framework ist von den Entwicklern von *WeTransfer* und funktioniert auch auf älteren iOS Versionen. Allerdings unterstützt *WeScan* noch kein stapelweises Scannen. Das bedeutet man kann immer nur ein Foto machen, welches erst abgespeichert werden muss, bevor man das nächste machen kann. Das ist für den Benutzer nicht bequem und spart wahrscheinlich auch keine Zeit.

- Link zu weScan und den andern Frameworks? - Video-Link noch mit rein. (https://www.youtube.com/watch?v
TiLDZo) zur Verdeutlichung

Um das Projekt gut durch zu planen benutze ich *Scrum* als Projektmanagement. Dies ist für agile Softwareentwicklung gut geeignet. Durch die gesammelte Erfahrung mit Scrum aus den vorherigen Semester, fiel mir die Planung leicht. Mit GitHub Projects und Issues konnte ich so dem Sprint Aufgaben zuordnen und Fortschritt nachvollziehen.

7.2 Entwurf und Design

Nach der Vorbereitung begann ich damit, mir Gedanken zu dem Workflow der App zu machen und erstellte den ersten Sprint bis zum nächsten Meeting. Für den Workflow fertigte ich zu jeder View ein grobes Design an, welches sehr schnell in *SwiftUI* umzusetzen ist. Das Aussehen der App sollten sich dann im Laufe der Zeit noch ändern, jedoch stand erstmal die Umsetzung des Konzepts im Vordergrund.

Aus den Designs und dem Workflow heraus entwickelte ich einen groben Plan, wie die Daten in der App gehandhabt werden sollten. Jedoch stellte ich schnell fest, dass meine Idee des Datenflusses nicht ideal ist, wenn noch mehr Views hinzukommen. Außerdem stellte die Kommunikation mit dem Server ein Problem dar, da diese immer asynchron abläuft. Aus diesen Gründen suchte ich nach einer Lösung. Bei der Suche entdeckte ich *Redux.js* eine JavaScript-Bibliothek zur Verwaltung von Zustandsinformationen in Webanwendung. ...

Da *SwiftUI* das Entwurfsmuster MVVM umsetzt, ist es möglich einen *Redux.js* ähnlichen State Container, als ViewModel zu implementieren. Und weil die App an einen Server gekoppelt sind und durch die Synchronisation mit diesem viele

7.3 Implementierung

Einzelne Schritte als eigene Section mit Verlauf der jeweiligen Entwicklung: erst das, dann kam das, dann musste es so, ...

7.3.1 Template erstellen

7.3.2 Template speichern

7.3.3 Template verwenden

...

Die Beispiel-App von *Apple* nahm ich mir als Vorbild für die Umsetzung der Texterkennung.

...

7.4 ”Abnahme”

Abnahme ...

Weitere Punkte die in Kapitel 6 rein könnten/sollten:

- beschreiben wie die einzelnen Views/Seiten nun aussehen und funktionieren? oder eher den Prozess der Entwicklung?

- Bild mit Wireframe aller Views und deren Workflow?

- AppState erklären? (Single source of truth) -> später AppStore mit mehreren States

- erklären deskew / Dokument ausrichten?

- Umrechnung der Bilder vom Template aus und zuschneiden der neuen Regionen erklären

- Attribute hinzufügen (den Vorgang) -> Rechteck einzeichnen...

- Verwendete Programme, Sprache, etc.

- ein paar Worte, wie gut die Entwicklung lief (Simulator vs. echtes Gerät), wo sind die Grenzen des Simulators (Keine Kamera), wo waren Probleme mit dem echten Gerät...

8 Weitere Entwicklung und Besonderheiten

8.1 App-Architektur

- redux like app store mit states erklären? (single source of truth)

8.2 API

- auf die API eingehen? oder nur tobias?

8.3

-

9 Grenzen der App

9.1 Probleme beim Erkennen von Dokumenten

- Gleich-farbiger Hintergrund

- Wenig Licht

- Hintergrund mit starken Kanten

- Runde Ecken, keine Ecken

- Starke Kanten im Bild (schwarzer Kreditkartenstreifen)

- zu jedem möglichem vlt. dann ein Beispiel Bild

9.2 Probleme der Klausur-Vorlage beim Scannen

- Schrift zu klein

- zu wenig Platz

- zu viel Platz

- Überschneidungen

- ...

9.3 Weitere ...

- Abstürze, Memory Leaks,

10 Templates

- Template und den Entwicklungsprozess vorstellen,

- wieso weshalb warum muss das nun so aussehen?

- Was kann an der neuen Vorlage immer noch optimiert werden?

- Welche Probleme konnten behoben werden?

11 Ausblick

- Es müssen nicht nur Klausuren sein, sondern alles mögliche (Krankenscheine, Urlaubsscheine, was auch immer, nur DB muss angepasst werden. (oder automatisches generieren von DB-Tabellen an Hand der Template Attribute))

- Server mit Bildern als Klausuren-Einsicht nutzen -> allerdings viele Probleme (Klausuren würden kopiert werden -> Profs mehr Arbeit, keinen direkten Kontakt zum Prof wegen Fragen, Verbesserungen oder Anmerkungen, ...)

- QR-Code-Idee um Vorlagen weg zu lassen -> Programm oder Plugin (Word/LaTex) was die QR-Codes dann automatisch erstellt und richtig einfügt. (Wichtige Daten sind da hinterlegt, Vor- und Nachteile von QR-Codes, ...)

- Ist die App nun schneller als der normale Umgang bleibt offen -> Bachelorarbeit knüpft da an...

Anhang A: Tätigkeitsbericht

Datum	Tätigkeit
24.02. - 01.03.	Ich habe mich mit der Problemstellung auseinander gesetzt, Ideen gesammelt, Problemanalyse betrieben und einen kleinen Prototypen entwickelt. Dazu erstellte ich einen minimalen Projektplanung, arbeitete mich in die Frameworks *Vision* und *VisionKit* ein und setzte eine Versionsverwaltung auf. Zusätzlich suchte ich nach einer passenden App-Architektur, die geeignet für das deklarative GUI-Framwwork SwiftUI, sowie für asynchrone Aufgaben, wie z. B. API-Aufrufe ist. Dabei stieß ich auf *Cleancode Architecture* und *Redux*. Am Ende der Woche konnte man in der App Vorlagen mit einer Seite erstellen. Das heißt man konnte ein Foto machen, aus welchem das Dokument rausgeschnitten und anschließend ausgerichtet wurde. Weiter war es möglich Regionen auf dem Dokument zu markieren und diese in der Vorlage abspeichern. Ansonsten konnten die Vorlage schon dazu benutzt werden, um die Regionen auf dem neuen Foto heraus zu schneiden.
02.03. - 08.03.	In dieser Woche habe ich die Texterkennung auf den berechneten Regionen eines neuen Fotos implementiert, den Workflow sowie viele andere Kleingikeiten in der App verbessert und alle Fehler der letzten Woche behoben, sodass ich neue Dinge implementieren konnte. Zudem probierte ich CI sowie Lint für das Projekt aus. Da CI für eine iOS-App mit *GitHub Actions* schwer aufzusetzen war und ab April etwas kosten würde, lies ich es sein. Des Weiteren pflegte ich das Projekt Management durch *Issues* und *Project-Boards* in GitHub. Anschließend programmierte ich den App-Workflow so um, dass nun mehr als eine Seite aufgenommen und analysiert werden konnte. Abgesehen von neuem Quellcode fing ich an den Praktikumsbericht zu schreiben und arbeitet mich dafür in LaTeX und die Bachelorarbeits-Vorlage für LaTeX der Hochschule Mittweida ein.
09.03. - 15.03.	Zu Beginn der dritten Woche schaute ich mir Möglichkeiten für serverseitiges OCR an. Genauer sammelte ich Informationen zu dem Framework Vapor und Swift unter Linux. Jedoch funktionieren die Frameworks *Vision* und *CoreML* von *Apple* unter Linux nicht, weshalb sich IronOCR als beste Option herausstellte. Ich entwickelte ein Datenbankmodell, mithilfe der in der App verwendeten Datentypen und erstellte dazu noch eine JSON-Struktur die später für die APIs verwendet werden könnte. Außerdem gab es ein Meeting, in dem wir unseren aktuellen Stand präsentieren sollten, um weitere Schritte und Aufgaben zu planen.Bis zum Ende der Woche arbeitete ich weiter an meinem Beleg und schrieb den Datenfluss in der App um. Nun ähnelt er sehr dem Redux-Model.

Datum	Tätigkeit
16.03. - 22.03.	Anfangs habe ich weiter an meinem Praktikumsbericht geschrieben, neue Issues hinzugefügt und bearbeitet. Außerdem gepflegte ich die Dokumentation und betrieben Projekt Management, um nun Links zwischen Regionen hinzuzufügen. Dabei entstanden neue Views und der Redux-Store musste dadurch angapsst werden. Es kam eine Erweiterung für die Texterkennung hinzu, so dass man durch die Auswahl eines Datentyps, das Resultat der Erkennung verbessern konnte. Des Weiteren habe ich bis zum Ende der Woche die Vergleich-Links vollständig implementiert und die App auf Fehler und Abstürze kontrolliert, sowie den Beleg um einige Kapitel erweitert.
23.03. - 30.03.	...

Literaturverzeichnis

[1] Michael Graf, Partner bei PwC
http://www-cs-faculty.stanford.edu/~uno/abcde.html

[2] Apple Developer - Sample Code, Detecting Objects in Still Images
https://developer.apple.com/documentation/vision/detecting_objects_in_still_images

[3] BRAGGE, Matti, et al. Model-View-Controller architectural pattern and its evolution in graphical user interface frameworks. 2013.
https://lutpub.lut.fi/handle/10024/92156

[4] Thomas Sillmann - Einsteig in SwiftUI. 22.11.2019
https://www.heise.de/developer/artikel/Einstieg-in-SwiftUI-4594018.html

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 13.03.2020