TU WIEN Informatics

Sketching Interfaces

Generating High Fidelity Prototypes and Mockups from Sketches

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Alex Untertrifaller
Matrikelnummer 01115807

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Thomas Grechenig
Mitwirkung: Christoph Wimmer

Wien, 1. Mai 2020

Alex Untertrifaller	Thomas Grechenig

Sketching Interfaces

Generating High Fidelity Prototypes and Mockups from Sketches

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Masterstudium Software Engineering & Internet Computing

eingereicht von

Alex Untertrifaller

Matrikelnummer 01115807

ausgeführt am

Institut für Information Systems Engineering

Forschungsbereich Business Informatics

Forschungsgruppe Industrielle Software

der Fakultät für Informatik der Technischen Universität Wien

Betreuer: Thomas Grechenig

Mitwirkung: Christoph Wimmer

Wien, 01. Mai 2020

Technische Universität Wien
Karlsplatz 13 | 1040 Wien | +43-1-58801-0 | www.tuwien.at

Erklärung zur Verfassung der Arbeit

Alex Untertrifaller

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschlieÿlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Mai 2020

Alex Untertrifaller

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während des Erstellens dieser Masterarbeit unterstützt und motiviert haben.

Ich bedanke mich bei Herrn Professor Thomas Grechenig für das Betreuen meiner Masterarbeit.

Mein Dank gebührt Herrn Dipl.-Ing. Christoph Wimmer, der meine Masterarbeit betreut und begutachtet hat. Ich möchte mich für die hilfreichen Anregungen und eine konstruktive Kritik herzlich bedanken.

Ebenfalls möchte ich allen TeilnehmerInnen meiner Studie danken, ohne die die Auswertung dieser Arbeit nicht hätte entstehen können. Mein Dank gilt ihrer Bereitschaft, an der Studie teilzunehmen und ihren interessanten Beiträgen und Antworten auf meine Fragen.

Außerdem möchte ich mich bei meiner Lebenspartnerin Patrizia Riegler bedanken, die mich während des Schreibens der Arbeit unterstützt und motiviert hat.

Abschließend möchte ich mich bei meinen Eltern bedanken, die mir mein Studium durch ihre Unterstützung ermöglicht haben.

Kurzfassung

Zur Steuerung einer Software benötigt man ein Interface. Im Laufe der Zeit hat sich dieses von einer einfachen Kommandozeile mit Texteingabe zu einer grafischen Oberfläche verändert. Dadurch hat sich auch der Aufwand für die Erstellung einer guten Benutzeroberfläche erhöht. Eine übliche Vorgehensweise ist, die Oberfläche zuerst zu planen und zu analysieren und dann erst zu implementieren. In der Regel ist der Designprozess einer Benutzeroberfläche ein langer Prozess. Zu Beginn eines Projektes erarbeiten die Stakeholder die Requirements, dabei entstehen mit jeder Iteration detailliertere Artefakte mit steigendem Detailgrad. Das Skizzieren mit der Hand ist ein intuitiver und schneller Ansatz für das Erstellen von ersten Entwürfen und immer noch ein sehr beliebtes Mittel zur Ausarbeitung einer Benutzeroberfläche.

Die Idee dieser Arbeit ist es, diese Schritte zu verkürzen, sodass die Software gleich eine Oberfläche und somit etwas Greifbares zum Testen und Analysieren bietet. Ziel dieser Arbeit ist es, ein Tool zu entwickeln, das die mit Hand skizzierten Benutzeroberflächen erkennen und diese auf Basis eines Design Guides in fertige Prototypen umwandeln soll. Dadurch kann die Oberfläche der Software sofort betrachtet und analysiert und der Designprozess verkürzt werden. Die Motivation für diese Arbeit ist es, eine schnellere Herangehensweise zu finden, um den Designprozess zu verbessern und somit eine gute Kommunikation zwischen Kunden und Softwareentwicklern zu schaffen.

Das im Rahmen dieser Arbeit entwickelte Tool hilft somit, Ideen und Problemstellungen besser zu veranschaulichen, da man sofort ein erstes Bild einer Idee erhält. Das Tool wird jedoch nicht den eigentlichen Designprozess ersetzen, sondern unterstützen. Mit Hilfe von Testpersonen wird ermittelt, ob das Tool diese Umwandlung von Skizzen in Prototypen ideengerecht ausführen kann. Die Testpersonen müssen User Interfaces skizzieren, und das Tool wandelt diese in ein fertiges Design um. Zusätzlich zu qualitativen Beobachtungen der Testpersonen bei der Erledigung der Aufgaben werden Fragebögen zur Sammlung von Feedback eingesetzt.

Die Antworten der Fragebögen zeigen, dass das Tool für das Umwandeln der Skizzen in ein fertiges Design geeignet ist. Das Tool kann die Handskizzen der ProbandInnen ideengerecht in fertige Prototypen umwandeln und somit helfen, den Designprozess zu verkürzen.

Schlüsselwörter

Handskizzierte Benutzeroberflächen, Beschleunigung Designprozess, Umwandlung Skizzen in fertiges Design

Abstract

An interface is required to control software. Over time, the user interface has evolved from a simple command-line interface into graphical user interfaces. As a consequence, user interface design has become more time consuming and complex. A typical approach for creating and designing user interfaces is to start with planning, design, and analysis before implementing the user interface. Thus, the design of user interfaces is typically a time-consuming process spanning several iterations. Based on the requirements established at the beginning of a project with the involvement of relevant stakeholders, increasingly detailed and refined artifacts are created within each iteration. In early design phases, sketching by hand is an intuitive and fast approach to generate early drafts and it is still one of the most used techniques to elaborate a user interface.

The idea of this thesis is to shorten the process of moving from rough, early sketches to detailed mockups which more closely resemble the final product, in order to create more detailed and specific representations of the final user interface for discussion, evaluation, and analysis. The goal of this thesis is to develop a software tool that translates hand-drawn user interface sketches into high fidelity mockups and prototypes based on a style guide. The motivation of this thesis is to speed up and support the design process and in doing so, to enhance communication and understanding between the various stakeholders involved.

The tool created for this thesis enables improved representation and visualization of early ideas, concepts, and problem solutions. However, the tool is not intended to replace the design process, but to support it.

To evaluate the correct functioning and utility of the tool in translating sketches to prototypes, test participants with different knowledge and experience of user interface design participated in a study. All participants had to draw sketches of user interfaces and the tool converted them into a finished design. In addition to the observations made during the study, participants had to complete a questionnaire to gather additional feedback. The results of the study and the responses to the questionnaire indicate that the tool is useful and functioning properly. The tool was able to successfully translate the hand-drawn sketches of participants into detailed high fidelity mockups and therefore it can help to speed up the design process.

Keywords

sketching user interface, design process speedup, converting sketches to finished designs

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Problemstellung

Um eine Software steuern zu können, benötigen wir ein Interface. Im Laufe der Zeit hat dieses sich von einer einfachen Kommandozeile mit Texteingabe zu einer grafischen Oberfläche mit Visualisierungen und verschiedenen Eingabemöglichkeiten verändert. Dadurch hat sich auch der Aufwand für die Erstellung einer guten Benutzeroberfläche erhöht. Eine übliche Vorgehensweise ist, die Oberfläche erst zu planen und analysieren und dann zu implementieren.

Das Skizzieren mit der Hand ist ein intuitiver und schneller Ansatz für das Erstellen von ersten Entwürfen. Auch wenn es mittlerweile verschiedene Software-Lösungen für das Erstellen von Entwürfen gibt, so ist das Skizzieren mit der Hand immer noch ein sehr beliebtes Mittel zur Ausarbeitung einer Benutzeroberfläche. Laut der Studie *The Tools Designers Are Using Today* aus dem Jahre 2015 wird vor allem in den frühen Phasen des Designprozesses auf Skizzen mit Stift und Papier gesetzt [1].

Mittels Skizzen wird versucht, die richtige Idee zu finden und diese dann immer weiter zu verfeinern, bis am Ende ein detaillierter Prototyp ausgearbeitet ist. Ein Risiko von iterativen Designprozessen ist, dass eine erste Idee unabhängig von ihrer Qualität und Eignung zur Lösung eines Problems über mehrere Iterationen hinweg verfolgt wird, ohne mit der nötigen Gewissenhaftigkeit nach alternativen, eventuell besseren Lösungsansätzen zu suchen. Dieses Vorgehen kann dazu führen, dass man am Ende einen bestmöglichen Prototypen zu einer Idee erstellt hat, aber dieser Prototyp nicht die bestmögliche Lösung zu einem Problem darstellt. Der bereits investierte Zeitaufwand ist groß, da der Prototyp bereits erarbeitet werden musste.

In der Regel ist der Designprozess einer Benutzeroberfläche ein langer Vorgang, der verschiedene Disziplinen der Softwareentwicklung durchläuft, bis das eigentliche Produkt

betrachtet werden kann. Zu Beginn eines Projekts werden von den Stakeholdern die Requirements für die Software analysiert, anschließend werden diese zu Features und schlussendlich zu Skizzen und/oder Prototypen ausgearbeitet. Dabei entstehen immer wieder neue Artefakte mit steigendem Detailgrad. Jeder der Akteure spielt eine wichtige Rolle für die Erstellung eines Software Interfaces. Empirische Studien haben gezeigt, dass dieser Prozess aufwendig, zeitintensiv und fehleranfällig ist, insbesondere wenn das Design und die Implementierung von verschiedenen Teams durchgeführt werden [2–5].

Die Idee dieser Arbeit ist es, den Designprozess zu verkürzen. Ziel dieser Arbeit ist es, ein Tool zu entwickeln, das die mit Hand skizzierten Benutzeroberflächen erkennt und diese auf Basis eines Design Guides in fertige Prototypen umwandeln kann. Die Oberfläche der Software kann somit gleich betrachtet und analysiert werden.

1.2 Motivation

Bill Buxton schreibt in seinem Buch *Sketching User Experiences: Getting the Design Right and the Right Design*: "Generally the last thing that you should do when beginning to design an interactive system is write code."[6]. Um ein User Interface zu erstellen, sollte zunächst ein Design ausgearbeitet werden; üblicherweise wird dafür ein Mockup erstellt. Dies kann mittels Software oder auch mit Stift und Papier erfolgen.

Die Motivation für diese Arbeit ist es, eine schnellere Herangehensweise zu finden, um den Designprozess zu verbessern und somit eine gute Kommunikation zwischen den unterschiedlichen Stakeholdern zu schaffen. Da das Skizzieren mit der Hand ein intuitiver Prozess ist, können alle Stakeholder am Designprozess teilnehmen, indem Sie selbst auf Papier ihre Ideen skizzieren. Auch sollten so neue Ideen ohne lange Planungs- und Implementierungszeiten schnell getestet werden können. Des Weiteren können so Änderungen an der Oberfläche schnell umgesetzt werden.

Das im Rahmen dieser Arbeit entwickelte Tool kann somit helfen, erste Ideen und Problemstellungen besser zu veranschaulichen, da man sofort ein erstes Bild einer Idee und somit etwas Greifbares erhält. Das Tool wird jedoch nicht den eigentlichen Designprozess ersetzen, sondern unterstützen.

1.3 Zielsetzung

Ziel dieser Arbeit ist es, ein Tool zu entwickeln, das handgemalene Skizzen eines User Interfaces in ein fertiges Design umwandelt. Beim fertigen Design handelt es sich um eine HTML Seite, die aus der Skizze generiert wird.

Mit Hilfe von Testpersonen wird analysiert, ob das Tool die Handskizzen ideengerecht in fertige Prototypen umwandeln kann. Des Weiteren wird analysiert, ob die Testpersonen so User Interfaces erstellen und ihre Ideen umsetzen können. Es soll eine explorative

Herangehensweise für die Evaluierung von Ideen und konkreten Problemstellungen für die ProbandInnen möglich sein.

Diese Herangehensweise wird mittels einer Studie, bei der die Testpersonen mehrere Aufgaben ausführen und anschließend einen Fragebogen ausfüllen müssen, evaluiert. Eigene Beobachtungen während der Durchführung der Studie werden genannt und beschrieben.

Es wird analysiert, ob diese Herangehensweise bei einem Designprozess eine Beschleunigung und damit eine Verbesserung des Designprozesses ermöglicht.

1.4 Methodische Vorgehensweise

In diesem Abschnitt wird die Problemlösung vorgestellt. Es wird auf das Design und die Architektur des Tools eingegangen und die verwendeten Technologien werden kurz beschrieben. Des Weiteren werden die Durchführungs- und Auswertungsmethoden der Studie erläutert und die Studienumgebung, Aufgabenstellung und der Fragebogen für die ProbandInnen vorgestellt. Am Ende des Kapitels wird die Auswertungsmethode für die Fragestellung beschrieben.

1.4.1 Tool-Design

Im Zuge dieser Arbeit wird das Sketching Interfaces Tool entwickelt, das handskizzierte Mockups in fertige Designs umwandelt. Fertige Designs sind in diesem Fall fertige Webseiten in HTML und CSS unter Verwendung eines Frontend Frameworks. Webseiten und mobile Applikationen sollen skizziert werden können, um so schnell einen Prototypen zu erhalten. Es muss ein Tool entwickelt werden, das die einzelnen Komponenten einer Skizze erkennt und klassifiziert und diese anschließend richtig in einem Layout anordnet. Die klassifizierten Komponenten müssen unter Verwendung eines Frontend Frameworks in HTML- und CSS-Elemente umgewandelt werden. Als Ergebnis soll ein fertiges Design zu sehen sein.

1.4.2 Implementierung

Für die konkrete Implementierung wird das Tool in mehrere Module unterteilt. Abbildung 1.1 zeigt eine Übersicht über die Architektur des Tools.

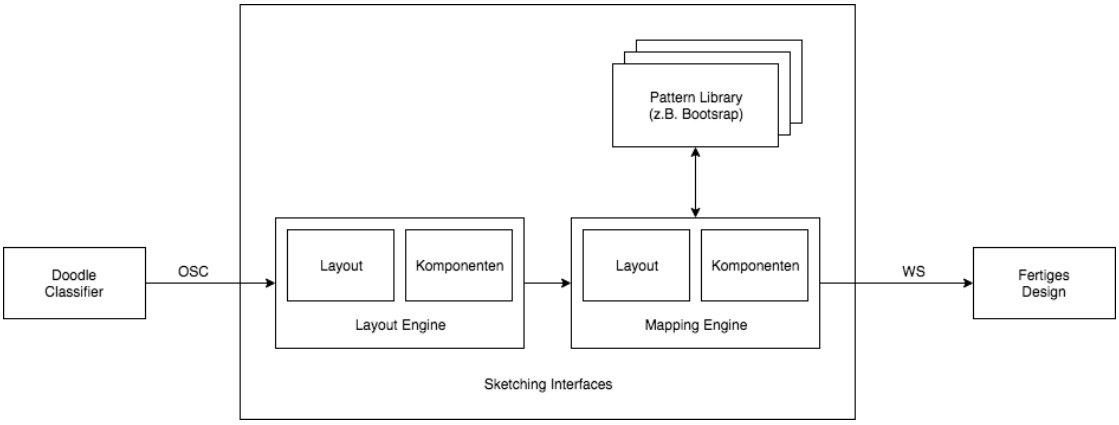


Abbildung 1.1: Architektur

Der **Doodle Classifier** ist ein Tool aus der Machine Learning for Artists - Open Frameworks (ml4a-ofx) Kollektion und wird verwendet, um die Skizzen mittels einer Kamera aufzunehmen und zu klassifizieren. Es werden die einzelnen gezeichneten Elemente auf der gesamten Skizze identifiziert und einzeln klassifiziert. Für die Klassifizierung werden Trainingsdaten aufgenommen und benannt, sodass am Ende der Doodle Classifier ein Label mit den Koordinaten pro erkannter Komponente an das nächste Modul weiterleitet. Um alle benötigten Daten im nächsten Schritt zu erhalten, muss der Doodle Classifier angepasst werden. Die Daten werden mittels Open Sound Control (OSC) Protokoll weitergeleitet. Dieses Protokoll ist bereits im Doodle Classifier implementiert und wird daher auch verwendet.

Für diese Arbeit wird das Tool **Sketching Interfaces** implementiert, das wiederum aus mehreren Teilen besteht:

- Die **Layout Engine** erhält die Daten vom Doodle Classifier und ist für das Erstellen des Layouts zuständig. Die Position der erkannten UI Komponenten wird ermittelt. Es soll erkannt werden, wo sich die Komponenten auf der Zeichnung befinden, um sie im fertigen Design richtig anzuordnen. Dazu muss erkannt werden, ob sich Komponenten untereinander oder nebeneinander befinden. Das Ergebnis der Layout Engine ist eine Folge von UI Komponenten und *kombinierten Objekten*, diese bestehen ebenfalls aus einer Reihe von UI Komponenten. *Kombinierte Objekte* können aus einem rechten und einem linken Teil bestehen, um so die Anordnung der UI Komponenten nebeneinander zu definieren.
- Die **Mapping Engine** ist für das Umwandeln der Labels, die der Doodle Classifier erkannt hat, in HTML Komponenten zuständig. Dafür greift die Mapping Engine auf vordefinierte Mappings im JSON Format zu. Für jedes verwendete Framework wird eine Mappingdatei erstellt. Dadurch ist es möglich, weitere Frontend Frameworks hinzuzufügen. Im Zuge dieser Arbeit werden zwei Frameworks, das Bootstrap und das Material Design Lite Framework, implementiert.

Das Ergebnis des Tools ist das **fertige Design** in HTML und CSS. Dieses wird vom Tool generiert und angezeigt.

Für die Umsetzung werden folgende Technologien verwendet: Das Tool wird in Node.js [7] implementiert und in einen Electron Container [8] verpackt. Für das Visualisieren des fertigen Designs wird HTML und CSS verwendet. Um Änderungen der Skizzen im fertigen Design upzudaten, wird ein Websocket (WS) verwendet, dafür wird im Frontend AngularJS [9] verwendet.

1.4.3 Studie und Umgebungsaufbau

Um das korrekte Umwandeln der Skizzen in ein fertiges Design zu testen, wird eine Studie durchgeführt. Für die Studie werden ProbandInnen aus verschiedenen Berufsgruppen ausgewählt, somit kann das unterschiedliche Vorgehen beobachtet werden.

Für die Durchführung der Studie wird eine Studienumgebung aufgebaut. Ein Whiteboard, auf dem die ProbandInnen ihre Skizzen zeichnen müssen, wird aufgestellt. Über das Whiteboard wird eine Webkamera auf einem Stativ montiert, sodass sie auf das Whiteboard gerichtet ist und alles aufzeichnet, was skizziert wird. Zum Zeichnen wird ein löschbarer Stift verwendet, damit Änderungen an der Skizze vorgenommen werden können. Die Webkamera wird an einen Computer angeschlossen, auf dem der Doodle Classifier und das Tool laufen und das fertige Design angezeigt wird.

1.4.4 Aufgabenstellung und Fragebogendesign

Alle ProbandInnen, die an der Studie teilnehmen, erhalten eine Aufgabenstellung, die für alle gleich ist, ein Infoblatt und einen Fragebogen.

Die ProbandInnen zeichnen die ersten Aufgaben eigenständig unter Verwendung eines Infoblattes, auf dem alle zur Verfügung stehenden User Interface (UI) Komponenten aufgelistet werden und erläutert wird, wie sie skizziert werden sollten. Das Infoblatt dient als Hilfestellung für die ProbandInnen. Die ersten drei Aufgaben geben ein Mockup einer Webseite oder App vor, die die ProbandInnen nachskizzieren sollen. Mithilfe dieser Aufgaben soll erkannt werden, ob das fertige Design den Erwartungen der ProbandInnen entspricht. Dafür müssen die UI Komponenten vom Tool richtig erkannt und angeordnet werden. Bei der letzen Aufgabe handelt es sich um eine Gruppenaufgabe, die die ProbandInnen zu zweit lösen müssen. Diese Aufgabe ist eine offene Aufgabenstellung, bei der der explorative Ansatz für das Erstellen eines Designs getestet wird. Die Durchführungsdauer für die gesamte Aufgabenstellung wird ca. 30 Minuten in Anspruch nehmen.

Am Ende erhalten alle ProbandInnen einen Fragebogen, den sie ausfüllen müssen und der für die Evaluierung des Tools verwendet wird.

1.4.5 Auswertungsmethode

Mittels des von den ProbandInnen ausgefüllten Fragebogens und den eigenen Beobachtungen während der Durchführung der Studie wird das Tool evaluiert, und es wird bestimmt, ob diese Herangehensweise zu den gewünschten Ergebnissen führt und die richtigen fertigen Designs liefert. Probleme, die bei der Durchführung der Studie aufgetreten sind, werden genannt und begründet.

Es wurde die Evaluierung mittels ProbandInnen im Zuge einer Studie gewählt, da so erkannt werden kann, ob die Zeichnungen verschiedener Personen erkannt und richtig umgewandelt werden können. So kann auch beobachtet werden, wie die ProbandInnen mit dem Tool interagieren und ob explorative Skizzieren sinnvoll ist und eine Beschleunigung für den Designprozess bedeutet.

1.5 Aufbau der Arbeit

Im Kapitel 2 werden zwei verwandte Arbeiten vorgestellt, in denen Screenshots von Benutzeroberflächen in fertige Designs umgewandet werden, und vier Arbeiten, in denen Handskizzen in fertige Designs umgewandet werden.

Kapitel 3 behandelt sowohl Grundlagen als auch Definitionen, auf denen diese Arbeit basiert, und bietet einen Überblick über den aktuellen Stand der für diese Arbeit relevanten Technik. Zudem werden UI Patterns, die für diese Arbeit relevant sind, beschrieben.

In Kapitel 4 werden die Tools und Technologien, die für die Erstellung des im Zuge dieser Arbeit erstellten Tools verwendet wurden, erläutert. Weiters werden die Architektur und ihre einzelnen Komponenten vorgestellt. Es werden alle verwendeten Kommunikationsprotokolle, die für diese Arbeit relevant sind, aufgeführt und es wird erklärt, wie das Tool erweitert werden kann.

In Kapitel 5 wird die Studie vorgestellt. Die Aufgabenstellung für die ProbandInnen und die Studienumgebung werden beschrieben und die verwendete Hardware wird vorgestellt. Die ProbandInnen, die die Studie durchgeführt haben, werden einzeln aufgeführt.

Kapitel 6 stellt die Ergebnisse der Studie vor, die detailliert analysiert werden.

Die Stärken und Schwächen des Tools, die sich aus der Studie ergeben haben, werden in Kapitel 7 angesprochen.

Abschließend fasst Kapitel 8 die wesentlichen Erkenntnisse zusammen und gibt einen Ausblick in die Zukunft.

KAPITEL 2

Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten, die einen ähnlichen Ansatz haben, vorgestellt und die Unterschiede zu dem im Zuge dieser Arbeit erstellten Tool aufgezeigt.

2.1 Pix2code

Pix2Code ist ein Tool, das im Rahmen des Papers *pix2code: Generating Code from a Graphical User Interface Screenshot* von Tony Beltramelli [10] erstellt wurde, um aufzuzeigen, dass es möglich ist, ein User Interface mittels Deep Learning Methoden aus einem Input Bild zu generieren. Ein grafisches User Interface, das von einem Designer erstellt wurde, wird oft als Screenshot an eine/n ProgrammiererIn weitergegeben, die/der dieses dann implementiert. Laut Beltramelli kann sich die/der ProgrammiererIn mehr auf die Logik konzentrieren, wenn ein Tool das Umwandeln des User-Interface-Screenshots in ein User Interface übernimmt, und somit wird die/der ProgrammiererIn entlastet. Daher versuchte Beltramelli, mittels Deep Learning Methoden ein Tool zu entwickeln, das das Umwandeln der Screenshots in User Interfaces übernimmt. Mit dem erstellten Tool konnte aufgezeigt werden, dass die Input Bilder mit einer 77%igen Genauigkeit für verschiedene Plattformen (iOS, Android und Web-basierte Anwendungen) umgewandelt werden konnten. Das von ihm entwickelte Tool analysiert die Input Bilder anhand ihrer Pixel, da diese leicht aus den Bildern extrahiert werden können. Es wurden Convolutional Neural Networks zur Klassifizierung der Bilder [11] verwendet. Eine leichtgewichtige Domänenspezifische Sprache (DSL) wurde erstellt, um die klassifizierten Input Bilder zu repräsentieren, wodurch Layouts und Komponenten erkannt und in den DSL Code umgewandelt werden. Textwerte sowie Bilder werden ignoriert und erhalten einen Platzhaltertext. Abbildung 2.1 zeigt die Repräsentation des Screenshots eines iOS User Interfaces (links im Bild) im DSL Code (rechts im Bild). Der DSL Code

wird verwendet, um den Source Code zu generieren. Dieser Ansatz ermöglicht es, den DSL Code in verschiedenen Programmiersprachen umzuwandeln [10].

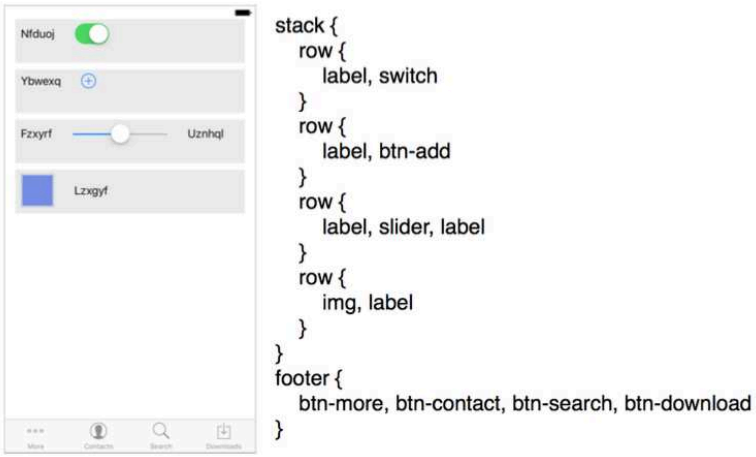


Abbildung 2.1: Pix2Code [10]

Abbildung 2.2 zeigt ein Beispiel einer Umwandlung, links im Bild der Screenshot wie er in das Tool pix2code eingelesen wurde und rechts im Bild das generierte User Interface [10].

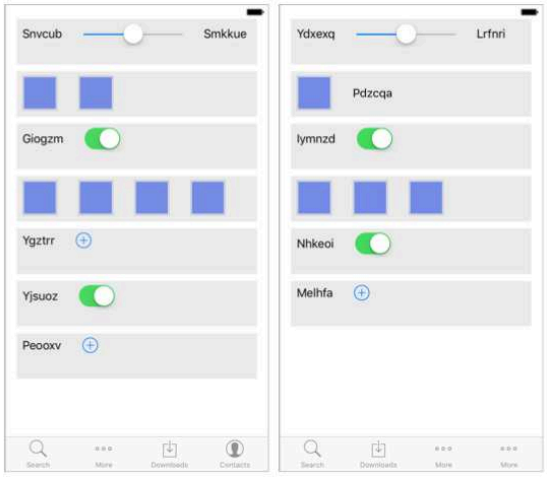


Abbildung 2.2: Pix2Code – Links Screenshot und rechts das generierte UI [10]

2.2 REMAUI

Reverse Engineer Mobile Application User Interfaces (REMAUI) können ebenfalls aus Screenshots von User Interfaces User Interfaces generieren [4]. Auch hier ist der Ansatz derselbe wie im Abschnitt 2.1 beschrieben; die GrafikerInnen erstellen User Interface Designs und geben diese den ProgrammiererInnen als Bild weiter, die/der diese wiederum implementieren muss. Da die Screenshots von User Interfaces mit Tools wie Adobe InDesign oder ähnlichen Programmen erstellt werden, sind sie im Gegensatz zu Handskizzen genauer (keine schiefen Linien, gleiche Komponenten variieren kaum), was das Erkennen im Gegensatz zu Handskizzen stark erleichtert. REMAUI for Android wurde entwickelt, um aus Bildern einen Android Code zu generieren. Es werden sowohl die UI Komponenten als auch die Textwerte und Bilder extrahiert und im generierten User Interface eingebaut. 488 Screenshots aus 100 bekannten Apps wurden mit dem Tool in User Interfaces umgewandelt; alle generierten UIs waren ähnlich [4].

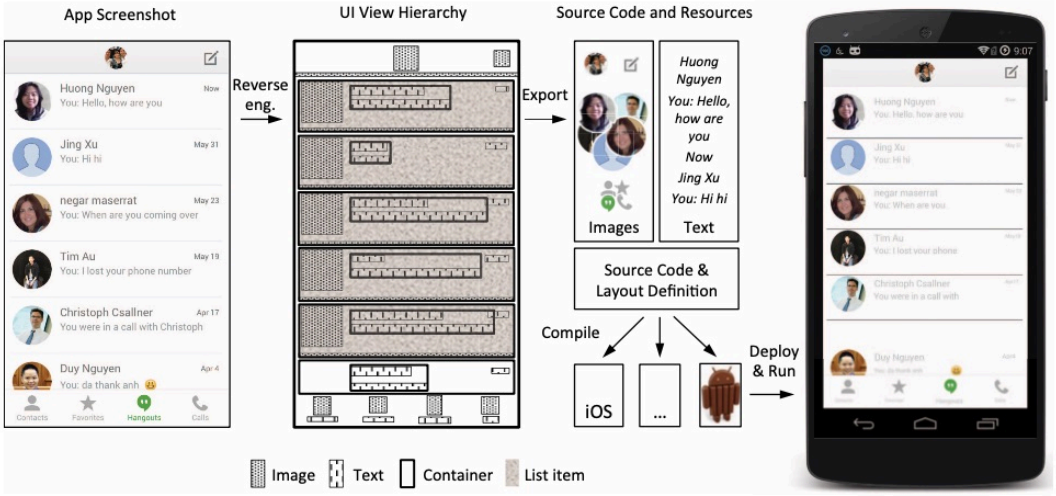


Abbildung 2.3: REMAUI [4]

Die Abbildung 2.3 zeigt den Vorgang der Umwandlung anhand eines Beispiels. Links ist der App-Screenshot dargestellt, so wie er vom Tool eingelesen wird. UI Elemente und Layouts werden klassifiziert, daraus werden die benötigten Informationen ausgelesen, um anschließend ein User Interface zu generieren [4].

2.3 SILK: Sketching Interfaces Like Krazy

SILK *Sketching Interfaces Like Krazy*: ist ein Tool zum Zeichnen von Skizzen am Computer, das von James A. Landay im Artikel *SILK: Sketching Interfaces Like Krazy* [12] 1996 veröffentlicht wurde. Mithilfe eines elektronischen Pads und eines Stylus war es möglich, Skizzen zu zeichnen. Das Tool erleichtert das schnelle Erstellen von Prototypen, anders als auf Papier ermöglicht es, Aktionen und Storyboards zu erstellen. Die Skizzen werden nicht auf Papier sondern mit Stylus und Pads erstellt, was den Vorteil hat, dass die Zeichnungen immer wieder verwendet werden können. Das Archivieren der Skizzen wird vereinfacht, da diese mit Labels versetzt und digital abgelegt werden können [12]. Ein weiterer Vorteil des Verwendens eines Stylus und Pads ist, dass die Zeichnungen bearbeitet werden können was hingegen auf Papier schwieriger ist [13].

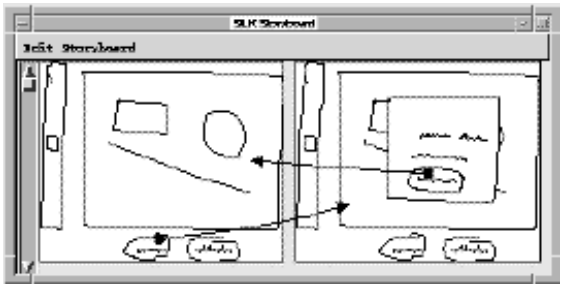


Abbildung 2.4: SILK Screenshot [12]

2.4 Airbnb

Der Design Technologe Benjamin Wilkins, der für das Unternehmen Airbnb tätig ist, beschreibt im Artikel *Sketching Interfaces - Generating code from low fidelity wireframes* seine Ansichten wie Artificial Intelligence den Designer zukünftig bei der Erstellung von Prototypen unterstützen wird [14].

”The time required to test an idea should be zero”. Die Idee des Teams hinter diesem Artikel ist, den Designprozess zu verkürzen und einen intuitiven Weg zu finden, um Prototypen zu entwickeln. Handskizzen sind laut Benjamin Wilkins der intuitivste und schnellste Weg, um Ideen festzuhalten und zu visualisieren. Dazu wurden alle UI Komponenten aus der Airbnb Sammlung in das Tool eingespielt. Dabei handelt es sich um Komponenten wie z.B. Buttons, Textfelder und Bilder. Whiteboards wurden verwendet, um die Skizzen zu erstellen. Mittels einer Kamera wurden die Skizzen aufgezeichnet und in das Tool eingespielt. Das Tool wandelt die aufgenommen Skizzen in High-fidelity Mocks um. Benjamin Wilkins erwähnt in seinem Artikel, dass er davon überzeugt ist, dass zukünftig die Artificial Intelligence den Designer dabei unterstützen wird, bessere und schnellere Software zu entwickeln [14].

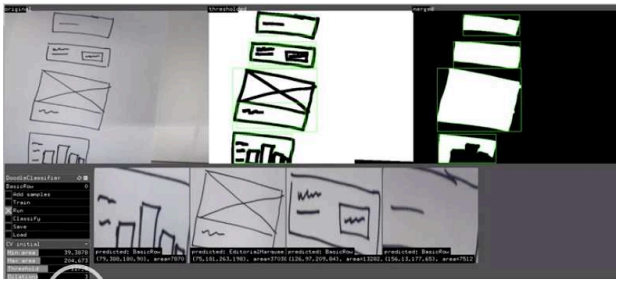


Abbildung 2.5: Airbnb-Tool Input [14]

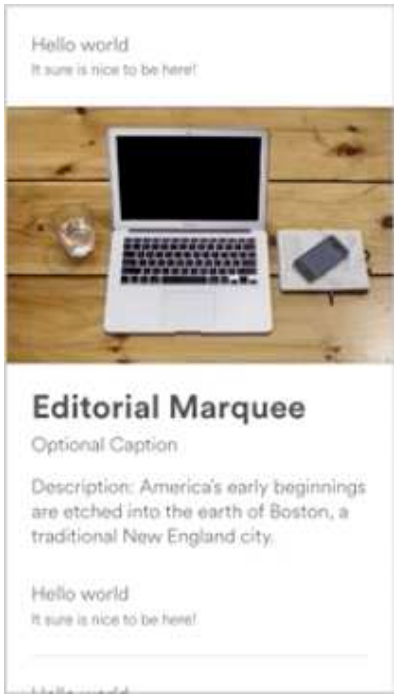


Abbildung 2.6: Airbnb-Tool Output [14]

Die Abbildung 2.5 zeigt das Tool mit dem die Handskizzen eingelesen und weiterverarbeitet werden. In der Abbildung 2.6 ist das Ergebnis der Auswertung der Handskizze zu sehen.

2.5 XING Engineering

Der Autor Markus Siering beschreibt in seinem Artikel *Teaching a machine to convert wireframes into code* [15] das Erstellen eines Tools für das Umwandeln von Skizzen in einen Source Code:

Dieser Artikel wurde von dem von Airbnb erstellten Prototypen inspiriert, der im Abschnitt 2.4 genauer beschrieben wurde. Das Tool besteht aus drei Teilen:

- Maschine Learning Instanz, die mit Trainingsdaten gefüllt wird.
- Einen Node.js Server der die Klassifizierungen entgegennimmt und an den Client weiterleitet.

- Eine Client App die die Klassifizierungen entgegennimmt und in die Komponenten umwandelt.

Machine Learning

Für das Machine Learning wurde die Bibliothek ml4a *(machine learning for artists)* verwendet. Der DoodleClassifier aus dieser Sammlung wurde für die Klassifizierung der Handskizzen verwendet. Der DoodleClassifier musste etwas angepasst werden, damit dieser die Positionskoordinaten der Komponenten ebenfalls weiterleitet. Da dieses Tool nur mobile Oberflächen erkennen und umwandeln soll, war hier nur die y-Koordinate wichtig. Trainiert wurde der DoodleClassifier zuerst nur von den Skizzen des Autors. Sobald weitere Skizzen von Kollegen hinzugefügt wurden, bekam der DoodleClassifier Probleme damit, die Skizzen zu identifizieren. Der Autor beschreibt, dass dies auf zu wenig Trainingsdaten zurückzuführen ist. Der Designer von Airbnb, Benjamin Wilkins, antwortet mit einem Twitter Post auf dieses Verhalten: "This was our experience too. Didn't take much when we were only working from a one person drawing. Looking forward to seeing your results with multiple people drawing!"[16]. Ein möglicher Lösungsansatz wäre die Maschine mit einem spezifischen Design Style zu trainieren oder viele Trainingsdaten zu verwenden.

Node.js Server

Der Node.js Server liest die Daten über die OSC Schnittstelle ein und sendet diese über einen Web Socket weiter an den Client.

Client Side App

Die Client Side App empfängt die Daten über den Web Socket und wandelt diese in ein fertiges Design um. In der Abbildung 2.7 ist die Client App zu sehen.

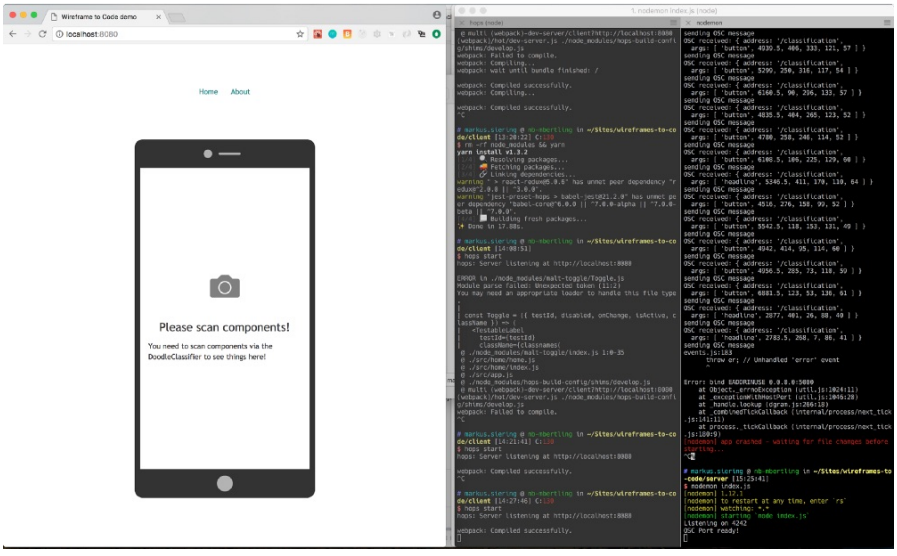


Abbildung 2.7: Client Side App [15]

2.6 Uizard

Uizard.io [17] ist eine Webapplikation, die handgemalte Skizzen in Prototypen umwandelt. Ein Foto einer Handskizze kann hochgeladen werden und das Tool wandelt dieses Foto auf Basis eines ausgewählten Styleguides in ein fertiges Design um. Die Styleguides können in der Weboberfläche des Tools bearbeitet oder es können neue erstellt werden. Farben, Formen und Icons der zur Verfügung stehenden UI Komponenten können bearbeitet werden. Es stehen nur wenige UI Komponenten zur Verfügung, die skizziert werden können. Text, Buttons, Bilder, Navigationsleisten, Formkomponenten und Icons können skizziert und bearbeitet werden. Es gibt keine Möglichkeit, weitere Komponenten in das Tool einzubauen.

Die erkannten UI Komponenten aus der Skizze werden in ein fertiges Design umgewandelt, wo sie manuell verschoben werden können. Das Tool bietet die Möglichkeit, aus den einzelnen Aufnahmen einen interaktiven Prototypen zu erstellen. Dazu können in der Weboberfläche des Tools Verlinkungen eingefügt werden. Eine Export-Funktion ermöglicht das Exportieren des Prototypen auf verschiedene Plattformen, wie z. B. HTML und CSS, Android oder Sketch. Derzeit können Skizzen von mobilen Applikationen umgewandelt werden, das Umwandeln von Desktop Applikationen befindet sich im Moment noch in der Beta Phase.

Das Tool ermöglicht das Einlesen von Bildern (z. B. Foto einer Skizze), es ist aber nicht möglich, während des Skizzierens direkte Ergebnisse der Umwandlung zu erhalten. Bei jeder Änderung müssen neue Fotos erstellt werden und in das Tool hochgeladen werden.

Abbildung 2.8 zeigt ein Beispiel einer Umwandlung mit dem Uizard Tool.

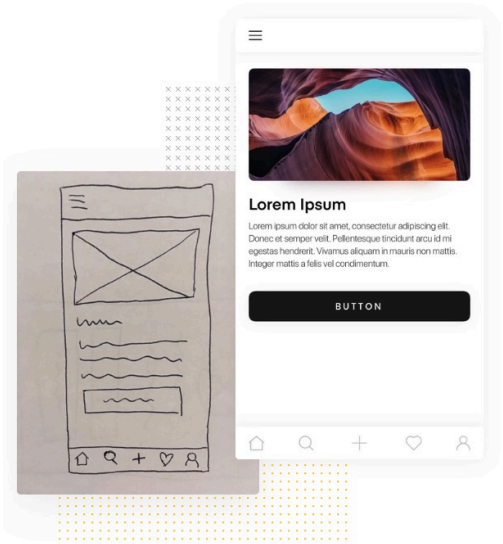


Abbildung 2.8: Uizard [17]

2.7 Zusammenfassung

Das Ziel dieser Projekte ist es, den Designprozess zu verkürzen und einen intuitiven Weg zu finden, um an schnelle Prototypen einer Software zu gelangen.

Die Tools *pix2code* und *REMAUI* wandeln Screenshots eines User Interfaces in ein fertiges Design um, im Gegensatz zu den in dieser Arbeit erstellten Tools werden jedoch keine Handskizzen umgewandelt. Da Screenshots genauer als Handskizzen sind und gleiche UI Komponenten kaum variieren, sind diese leichter zu klassifizieren als handgemalte Skizzen, die in ihrem Aussehen variieren können.

Das Tool *SILK* erkennt per Hand gemalte Skizzen, die mittels Stylus und Pad gezeichnet werden, jedoch nicht Skizzen auf Papier. Dies bringt den Vorteil, dass Änderungen leicht vorgenommen werden können. Durch die Verwendung eines Whiteboards mit löschbarem Stift hat das Tool, das für diese Arbeit erstellt wurde, ebenfalls diesen Vorteil.

Bei den Prototypen von *Airbnb* und *XING Engineering* wurden die gleichen Technologien verwendet, um die Tools zu erstellen. Beide Arbeiten können Skizzen in Prototypen umwandeln, jedoch wird keine Rücksicht auf das Layout der Handskizze genommen. Es wird lediglich Rücksicht auf die y-Koordinate der gescannten Objekte genommen, dadurch ist es nur möglich, die Komponenten untereinander anzuordnen. In dieser Arbeit wird sowohl die y- als auch die x-Koordinate der Komponenten betrachtet und versucht, diese korrekt anzuordnen.

Ein weiterer wichtiger Unterschied zu den vorher genannten Tools ist die Möglichkeit, verschiedene Frontend Frameworks auswählen zu können. Im Zuge dieser Arbeit wurden zwei Frontend Frameworks implementiert, das Bootstrap und das Material Design Lite Framework. Das Tool wurde jedoch so erstellt, dass es leicht mit weiteren Frontend Frameworks erweitert werden kann, so können auch selbst erstellte Designs von UI Komponenten in das Tool eingebaut werden, wodurch das fertige Design sofort dem Endprodukt ähnelt.

KAPITEL 3

Aktueller Stand der Technik

3.1 Designprozesse für User Interfaces

Zum Steuern einer Software benötigt man ein Interface. Im Laufe der Zeit hat sich das Interface von einer einfachen Kommandozeile mit Texteingabe zu einer grafische Oberfläche mit verschiedenen Komponenten für die Ein- und Ausgabe von Informationen weiterentwickelt. Dadurch hat sich auch der Aufwand für das Erstellen eines guten User Interfaces erhöht. In der Regel wird zuerst analysiert und geplant und dann implementiert.

Auch wenn es für das Erstellen von Mockups verschiedene Software-Lösungen gibt, so ist das Skizzieren mit der Hand immer noch ein beliebtes Mittel für das Erstellen eines ersten Entwurfs, da dies ein schneller und intuitiver Ansatz ist. Ein Risiko von iterativen Designprozessen ist, dass eine erste Idee unabhängig von ihrer Qualität und Eignung zur Lösung eines Problems über mehrere Iterationen hinweg verfolgt wird, ohne mit der nötigen Gewissenhaftigkeit nach alternativen, eventuell besseren Lösungsansätzen zu suchen. Dieses Vorgehen kann dazu führen, dass man am Ende einen bestmöglichen Prototypen zu einer Idee erstellt hat, aber dieser Prototyp nicht die bestmögliche Idee zu einem Problem darstellt.

Bill Buxton schreibt *"Getting the design right and getting the right design"* in seinem Buch *Sketching User Experiences: Getting the Design Right and the Right Design* [6] und meint damit Folgendes: Erstellt man ein Design für eine Benutzeroberfläche, das ein Problem *gut* löst, dieses Problem jedoch gar nicht existiert, kann man es nicht als Erfolg zählen. Hat man eine Lösung für ein Problem und man erstellt das Design so, dass es niemand verwenden kann, kann man dies auch nicht als Erfolg zählen. *Getting the right Design:* In dieser Phase muss herausgefunden werden, wieso der/die BenutzerIn das tun, was sie tun. Verschiedene Konzepte sollten durchgetestet werden, sodass man am Ende eine gute Lösung für ein Problem findet. *Getting the design Right:* In dieser Phase sollte

darauf geachtet werden, dass man die Lösung für ein Problem so umsetzt, dass ein gutes Design für eine Problemlösung entsteht [6].

Der Designprozess einer Benutzeroberfläche ist in der Regel ein langer Prozess, der durch verschiedene Phasen gehen muss, sodass am Ende ein fertiger Prototyp entsteht. Dabei sind mehrere Stakeholder involviert, die zusammen die Requirements analysieren und mit jeder Iteration neue Artefakte mit steigenden Detailgraden erzeugen.

3.1.1 Sketching vs. Prototyping

Skizzen sind meist händisch gemalte, grobe Andeutungen der Funktionalität der Software; Prototypen sind hingegen detaillierter und enthalten neben den Funktionen auch Designs. In der ersten Phase des Designprozesses ist die Umsetzung der Anforderungen nicht gegeben, und es muss zuerst eine Idee zur grafischen Umsetzung der Funktionen erkundet werden. Ein exploratives Vorgehen mit Hilfe von Skizzen ist hier von Vorteil, um ein gutes Konzept für die Softwareanforderungen zu finden. Prototypen auf der anderen Seite werden zu einem späteren Zeitpunkt erstellt, zu dem das Grundkonzept der Umsetzung der Software bereits feststeht, daher geht man beim Prototyping zielgerichteter vor. Abbildung 3.1 zeigt, wie zu Beginn durch ein exploratives Skizzieren verschiedene Ideen verfolgt werden. Das Prototyping kommt danach zum Einsatz, wenn aus diesen Ideen die beste weiter verfolgt und verfeinert wird [18].

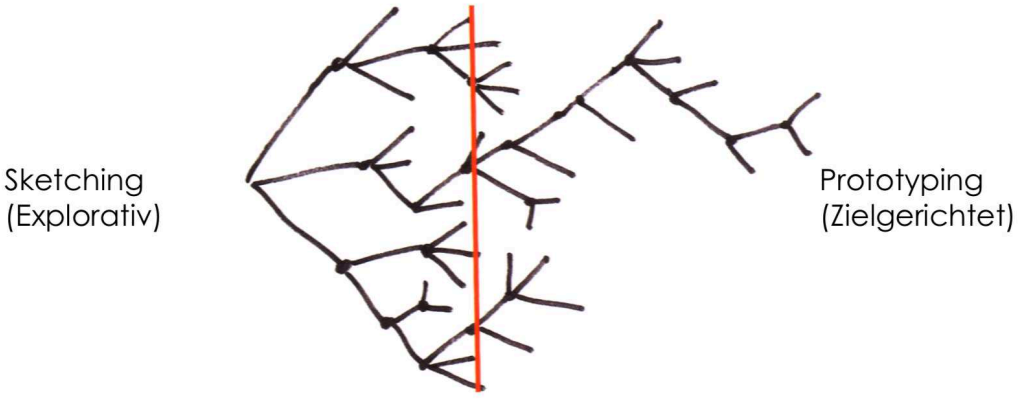


Abbildung 3.1: Sketching vs Prototyping [18]

3.1.2 Sketching

Das Skizzieren im Design-Prozess hilft den Entwicklern und Designern beim explorativen Suchen des richtigen Designs. Dabei wird weniger auf Details als auf das Gesamtkonzept geachtet. Durch das Verwenden von Stift und Papier können Ideen schnell auf Papier gebracht werden, um so etwas Greifbares für weitere Überlegungen zu schaffen. Das Skizzieren mit Stift und Papier benötigt kaum Zeit und dient daher auch in unserem digitalen Zeitalter immer noch hervorragend als erster Schritt, um Ideen zu visualisieren [18].

Die Studie *2019 Design Tools Survey* hat aufgezeigt, dass 86% aller befragten Personen Handskizzen auf Papier oder auf einem Whiteboard verwenden, um erste Ideen zu skizzieren. Nur sehr wenige verwenden in den frühen Phasen des Designprozesses eine Software-Lösung wie Sketch, Figma oder Adobe XD [19].

3.1.3 Prototyping

Prototypen sind nicht immer voll ausgereift erste Versionen eines Produktes. Prototypen dienen dazu, ein Produkt zu optimieren und verbessern. Es wird zielgerichtet eine Idee erweitert und optimiert. Ein Prototyp stellt ein detailliertes Design des endgültigen Produktes dar. Dabei unterscheidet man folgende Typen [18]:

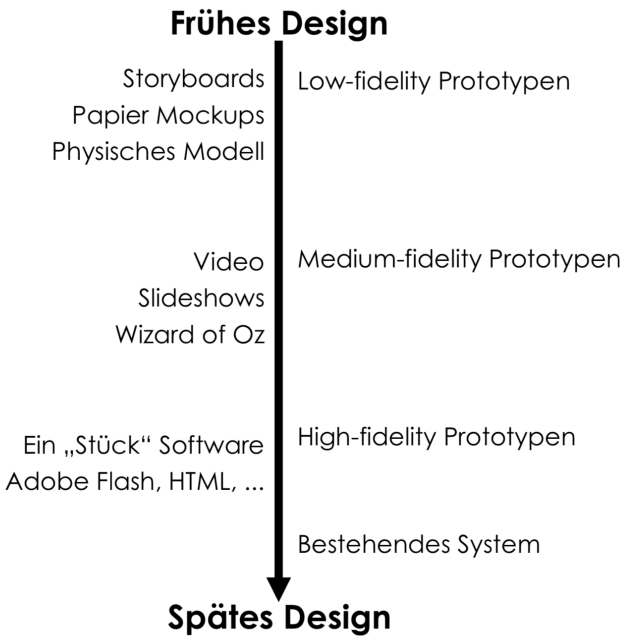


Abbildung 3.2: Fidelity Prototypen [18]

Low-fidelity

Bei Low-fidelity-Prototypen handelt es sich um grobe Ausarbeitungen und Entwürfe mit geringem Detailgrad, die in den frühen Phasen des Designprozesses erstellt werden. Bei diesen Prototypen liegt der Fokus auf der Funktionalität und nicht auf dem detaillierten Graphikdesign. Das bringt vor allem in den frühen Entwicklungsphasen viele Vorteile, dadurch können nämlich verschiedene Designkonzepte schnell und mit einem geringen Kostenaufwand evaluiert werden. Sie werden als Kommunikationsmedium zwischen Entwicklern und deren Kunden eingesetzt, um die Funktionen und deren Verwendungszwecke zu visualisieren. Durch die geringen Entwicklungskosten der Low-fidelity-Prototypen sind Änderungen leicht und schnell vornehmbar [18].

Einer der Nachteile dieser Art von Prototypen ist, dass durch die groben Skizzen keine genauen Spezifikationen festgehalten werden. Außerdem sind der Workflow und die Navigation nur beschränkt in den Skizzen zu sehen [18].

High-fidelity

Bei den High-fidelity-Prototypen handelt es sich um detaillierte Prototypen, die funktional und interaktiv sind und bereits nahe an das finale Endprodukt herankommen. Sie werden oft am Ende des Designprozesses verwendet, um die Verwendbarkeit der Software zu überprüfen und um Fehler im Workflow der Software festzustellen. Ein Nachteil dieser Prototypen ist die teure Entwicklung, da das Erstellen zeitaufwändig ist [18].

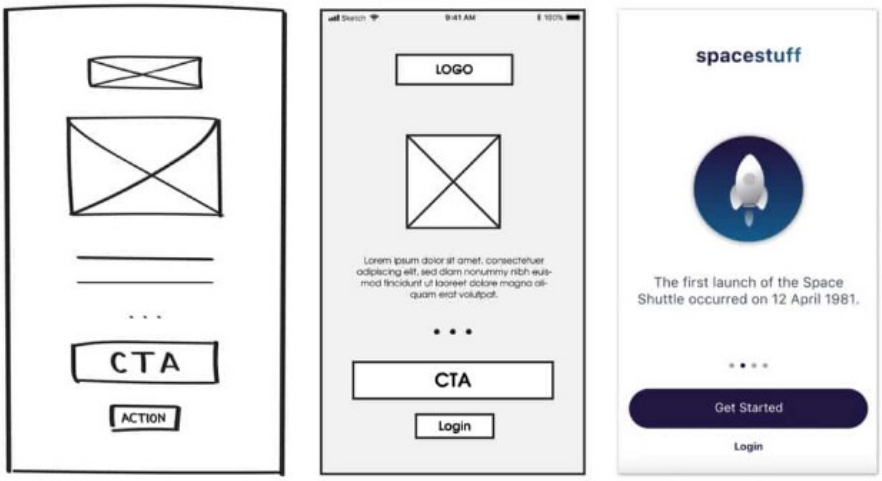


Abbildung 3.3: Fidelity Prototypen [20]

Abbildung 3.3 zeigt die verschiedenen Detailgrade von Prototypen. Links im Bild der Low-fidelity-Prototyp, mittig der Medium-fidelity-Prototyp und rechts der High-fidelity-

Prototyp. In der Abbildung werden die Unterschiede in der Genauigkeit des Prototypen deutlich dargestellt. Im ersten Bild werden die Funktionen des Prototypen angedeutet, das Design aber vernachlässigt. Der High-fidelity-Prototyp sieht bereits wie die fertige Software aus [20].

3.2 Frontend Frameworks

Pattern Libraries helfen dabei, Konsistenz herzustellen, ein gemeinsames Vokabular für alle am Projekt Beteiligten zu schaffen, die Wiederverwendung zu fördern und das Testen zu erleichtern [21].

Für DesignerInnen und EntwicklerInnen ist es eine Herausforderung, Oberflächen zu erstellen, die in einer Vielzahl von verschiedenen Bildschirmgrößen, Geräten, Browsern und Umgebungen ansprechend aussehen und funktionieren sollen. Aus diesem Grund ist es hilfreich, die Oberflächen in ihre kleinsten Elemente zu zerlegen [22]. Ausgehend von diesen Elementen können wiederum immer komplexere Komponenten geschaffen werden.

Im Folgenden werden die Begriffe „UI Komponenten“ und „Layouts“ erläutert.

UI Komponenten

Benutzeroberflächenkomponenten sind wiederverwendbare Bedienungselemente, die eine grafische Interaktion mit einer Software ermöglichen. Benutzeroberflächenkomponenten wie Buttons oder Textinputs sind mittlerweile in fast jeder Benutzeroberfläche vorhanden. Kombinationen aus verschiedenen Benutzeroberflächenkomponenten haben neue Komponenten geschaffen, Suchfelder z. B. sind in den meisten Fällen eine Kombination aus Textfeldern und Buttons.

Layouts

Das Layout ist eine Gestalltungsvorlage zum Erstellen von Benutzeroberflächen, es dient als Grundgerüst für eine Benutzeroberfläche. Die Komponenten einer Benutzeroberfläche werden in einem Layout angeordnet. Abhängig von der Benutzeroberfläche werden verschiedene Layout Vorlagen verwendet. Bei Responsive Designs, passt sich das Layout an die Größe des Displays an und ordnet so die Komponenten der Benutzeroberfläche um, sodass diese immer optimal angeordnet werden.

3.2.1 Frontend Frameworks

Frontend Frameworks sind eine Sammlung von vorgefertigten UI-Komponenten, die verwendet werden können, um eine Webseite zu erstellen. Sie sollen den Entwickler und den Designer dabei unterstützen Zeit zu sparen.

Eigenschaften eines guten Frontend Frameworks [23]:

1. Ein Framework sollte maßgeschneidert und auf die Bedürfnisse der zu erstellenden Software zugeschnitten sein.
2. Ein Framework sollte verwendbar sein, und zwar sowohl für die/den BenutzerIn als auch für die/den EntwicklerIn zur Implementierung.
3. Ein Framework sollte erweiterbar sein.

Es gibt zwei grundlegende Regeln wie ein Frontend Framework verwendet werden sollte [23]:

1. Der Dokumentation folgen
2. Keinen Framework Code überschreiben

Das im Rahmen dieser Arbeit entwickelte Tool erkennt die verschiedenen UI Komponenten und wandelt diese in HTML Komponenten um. Das Tool wurde so entwickelt, dass verschiedene Frontend Frameworks verwendet werden können, um die UI Komponenten darzustellen. Im Zuge dieser Arbeit wurden zwei dieser Frontend Frameworks ausgewählt und eingebunden:

Bootstrap

Bootstrap wurde von den Entwicklern von Twitter Mark Otto and Jacob Thornton im Jahre 2011 unter dem Namen Twitter Blueprint entwickelt. Es wurde unter einer Open-Source Lizenz veröffentlicht und ist nun als Bootstrap bekannt. Bootstrap besteht aus HTML, CSS und JS Komponenten und wurde für die mobile first responsive Webseite optimiert [24, 25].

Material Design Lite

Material Design Lite ist ein Framework, das das von Google entwickelte Material Design implementiert [26]. Material Design Lite ist ein Framework das für Mobile Anwendungen optimiert ist, jedoch auch für Webanwendungen verwendet werden kann.

Das Tool wurde so erstellt, dass weitere Frontend Frameworks auf einfache Weise hinzugefügt werden können. Wie im Abschnitt 4.2.3 beschrieben, wird dies über eine Mapping-Datei realisiert.

3.2.2 Im Tool verwendete UI Komponenten

In diesem Abschnitt werden UI Komponenten vorgestellt, die im Prototyp implementiert wurden. Um die UI Komponenten der Handskizzen bestmöglich erkennen zu können, müssen diese so ähnlich wie möglich an die im folgenden Abschnitt gezeigten Trainings Skizzen gezeichnet werden. Das Tool erlaubt es, weitere Komponenten hinzuzufügen. Es werden die skizzierten Zeichnungen, die für das Training verwendet werden, und die jeweiligen UI Komponenten der Frameworks vorgestellt:

Button

Der *Button* ermöglicht es Aktionen durchzuführen. Es handelt sich hier um eine der wichtigsten Komponenten die in jedem User Interface Gebrauch finden. Die Handskizze links in der Abbildung 3.4 zeigt einen *Button*, die Beschriftung ist bewusst weggelassen worden um die Skizze besser von der Input Skizze in der Abbildung 3.5 zu unterscheiden. Rechts in Abbildung 3.4 zeigt die *Buttons* wie sie vom Tool generiert werden, es handelt sich hier um einen *Button* aus dem Bootstrap Framework und einen *Button* aus Material Design Lite Framework.

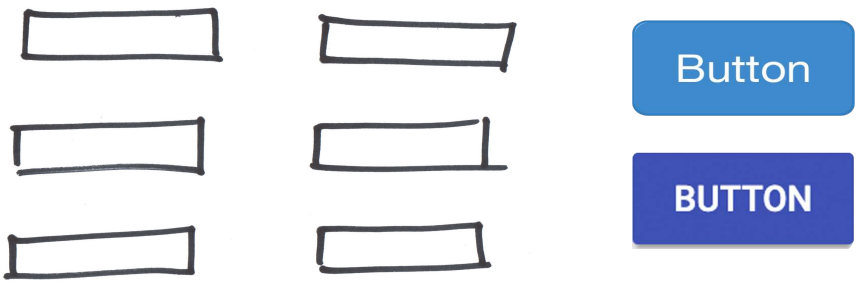


Abbildung 3.4: UI Komponente – Button

Input

Die *Input* Komponente ermöglicht das Eingeben von Texten in einem User Interface. Links in der Abbildung 3.5 ist die Handskizze zu sehen und rechts die UI Komponenten aus den Frameworks: Bild oben aus dem Bootstrap Framework und das Bild darunter aus dem Material Design Lite Framework. Die Komponente ist ähnlich wie ein Button zu zeichnen mit dem Unterschied, dass hier noch Text in das Rechteck geschrieben werden muss.

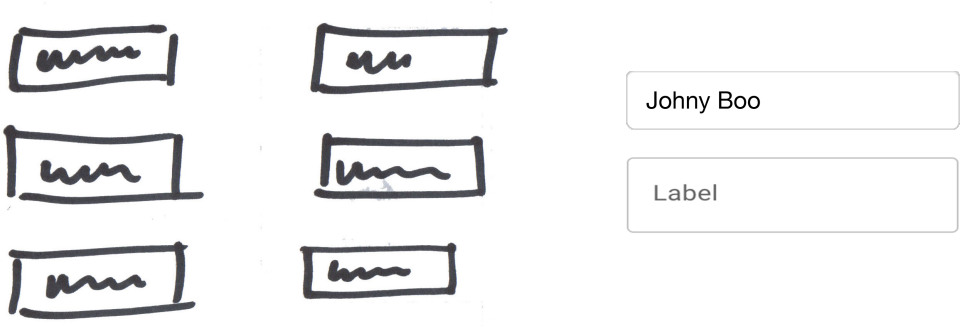


Abbildung 3.5: UI Komponente – Input

Image

Bei der UI Komponente *Bild* handelt es sich um eine Komponente die in den meisten User Interfaces Gebrauch findet, sie wird verwendet um Bilder in einer Benutzeroberfläche einzubinden. Links in der Abbildung 3.6 sind unterschiedliche Möglichkeiten aufgezeigt, um ein Bild mit der Hand zu skizzieren.

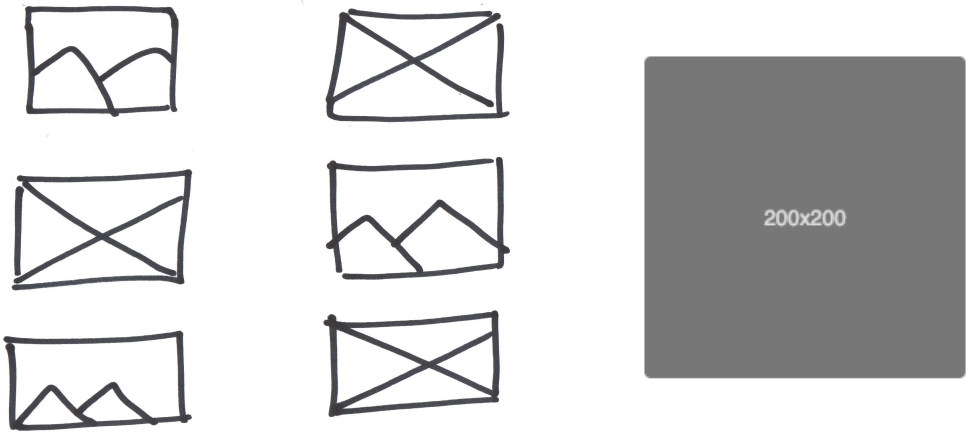


Abbildung 3.6: UI Komponente – Image

Text

Der *Text* wird verwendet, um Texte in einem User Interface darzustellen. Es handelt sich um eine einfache jedoch unverzichtbare Komponente, denn Texte findet man in fast jedem User Interface. Das Tool wandelt die Textskizze in einen Platzhaltertext um.

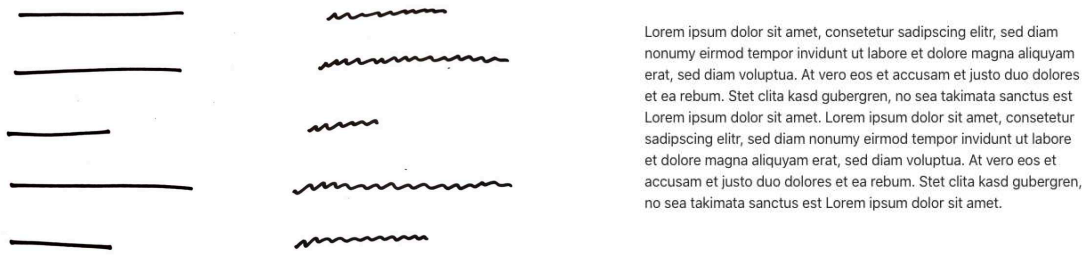
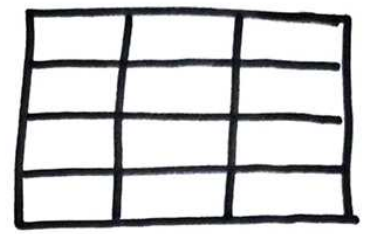


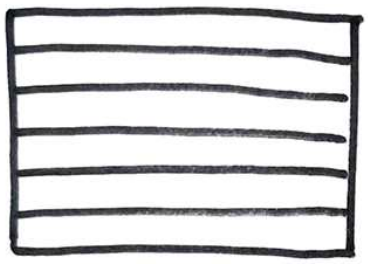
Abbildung 3.7: UI-Komponente – Text

Tabelle

Die *Tabellenkomponente* ermöglicht das Visualisieren von Informationen, wodurch diese strukturiert dargestellt werden können. Die Abbildung 4.25 zeigt links die Skizze der Tabelle, rechts oben die Tabelle aus dem Bootstrap Framework und rechts unten die Tabelle aus dem Material Design Lite Framework.



#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter



First	Last	Handle
Mark	Otto	@mdo
Jacob	Thornton	@fat
Larry	the Bird	@twitter

Abbildung 3.8: UI-Komponente – Tabelle

Carousel

Das *Carousel* wird verwendet um mehrere Bilder anzuzeigen, diese wechseln nach einer bestimmten Zeit automatisch zum nächsten. Meist wird auch eine Möglichkeit geboten manuell die Bilder in der Slideshow umzuschalten. Das *Carousel* findet meist auf der Startseite einer Webseite Gebrauch. Links in der Abbildung 3.9 sind einige Möglichkeiten aufgezeigt um ein *Carousel* mit der Hand zu skizzieren.

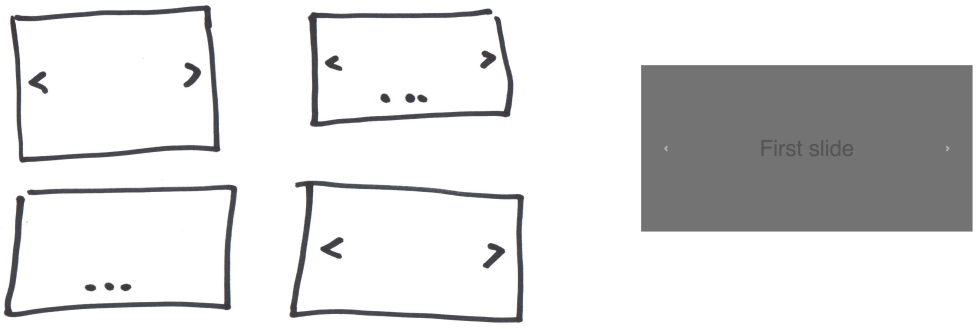


Abbildung 3.9: UI Komponente – Carousel

Navbar

Die *Navbar* wird verwendet um in einem User Interface zu navigieren, es dient als Navigationsmenü und wird oft auf jeder Unterseite angezeigt. Oben in Abbildung 3.10 sind einige Möglichkeiten dargestellt um die *Navbar* zu skizzieren.

Abbildung 3.10: UI Komponente – Navbar

Info Card

Die *Info Card* Komponente ist eine Komponente die ein Bild mit Text bzw. anderen UI Komponenten kombiniert und zum Anzeigen von Informationen verwendet werden kann. Links in der Abbildung 3.11 sind die Trainings Handskizzen zu sehen und auf der rechten Seite oben die UI Komponente aus dem Bootstrap Framework und rechts unten die Material Design Lite UI Komponente.

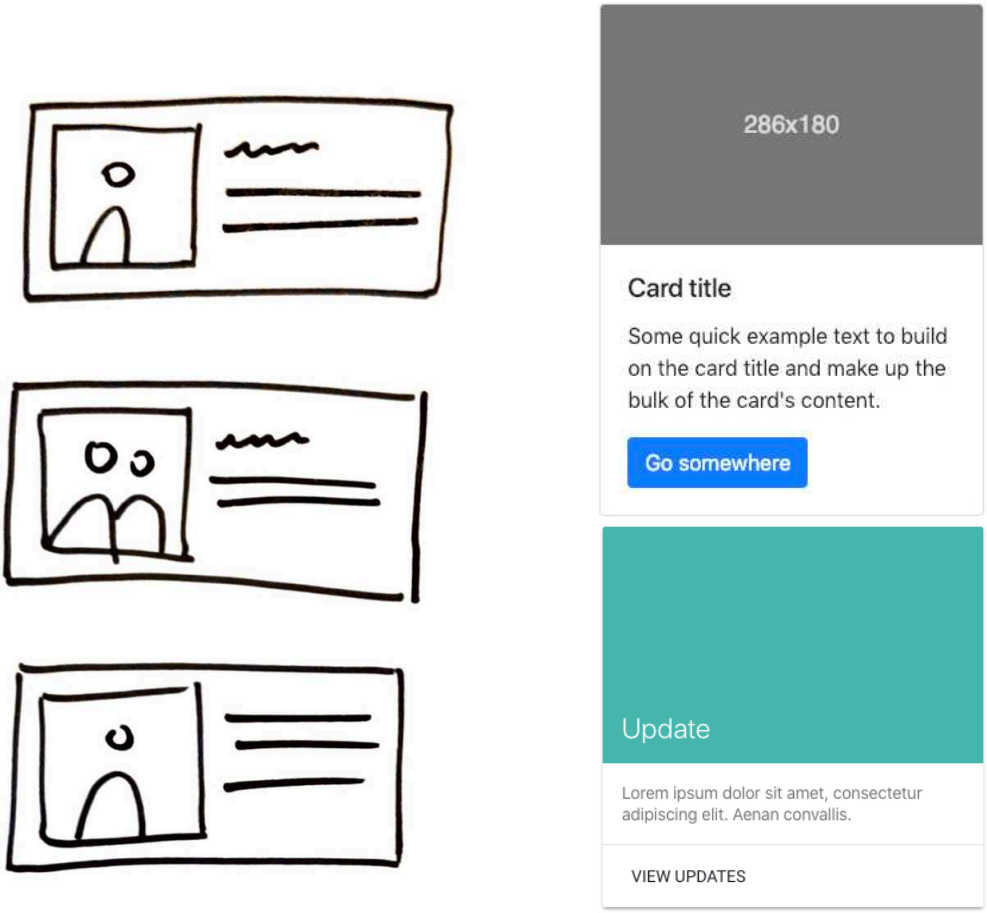


Abbildung 3.11: UI Komponente – Info Card

3.3 Convolutional Neural Networks

Für das Erkennen und Klassifizieren der Skizzen auf Papier wurde ein existierendes Tool, der Doodle Classifier, verwendet. Dieser benutzt Convolutional Neural Networks (CNN) für das Klassifizieren der Skizzen. Im folgenden Abschnitt wird das CNN näher erläutert.

Convolutional Neural Networks ist ein Deep-Learning-Modell zum Analysieren von Bildern. CNN ist auf eine ähnliche Weise aufgebaut, wie wir Menschen unsere Umgebung wahrnehmen. Zum besseren Verständnis, wie ein Computer Bilder analysiert, ist es erforderlich, zuerst die menschliche Analyse von Bildern zu betrachten. Im folgenden Abschnitt wird auf Basis des Buches [27], des Konferenz-Papers [28] und dem Paper [29] ein Einblick in das Thema CNN gegeben.

Die Lichtrezeptoren in unserem Auge senden über optische Nerven die Informationen, die unser Auge aufnimmt, an den primären visuellen Cortex, der die Bilder analysiert. Ein tiefes Netzwerk von Neuronen und Verbindungen spielt eine wichtige Rolle beim Speichern und Benennen von Objekten. Das menschliche Gehirn analysiert die Bilder in verschiedenen Ebenen mit steigender Komplexität. Die erste Ebene erkennt z. B. Linien und Kurven, in höheren Ebenen erkennt das Gehirn, dass die Kombination aus Linien und Farben ein bestimmtes Objekt ergibt.

Der Computer hingegen „sieht“ ein Bild als ein Array mit Werten. Normalerweise handelt es sich hier um ein 3-dimensionales Array, was den RGB-Werten eines Bildes entspricht. Die nachstehende Abbildung 3.12 spiegelt z. B. ein 6x6 RGB-Bild wider. Jedes Pixel hat einen Rot-Wert, einen Grün-Wert und einen Blau-Wert.

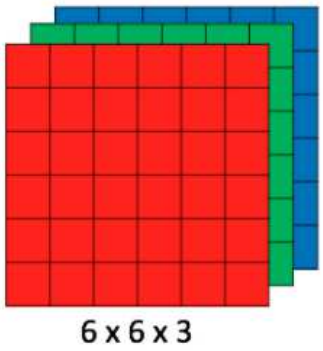


Abbildung 3.12: 6x6 RGB Bild [30]

Bei einem 3-dimensionalen Array handelt es sich um die Repräsentation des Bildes am Computer. Das Convolutional Neural Networks kann dieses Bild mit Hilfe von Filtern klassifizieren. Ähnlich wie beim menschlichen Gehirn werden verschiedene Filter in verschiedenen Ebenen angewendet, um diese Bilder zu klassifizieren. In den ersten Ebenen erkennen die Filter Linien und Ecken, in höheren Ebenen können komplexere Objekte erkannt werden.

Anhand des Beispiels *32x32x3* (32 Breit - 32 Hoch - 3 Farbkanäle) werden im folgenden Abschnitt die verschiedenen Schichten genauer beschrieben:

Convolution

Ziel ist es, eine Funktionskarte zu erhalten. In den ersten Ebenen wird eine geringe Anzahl an Filtern verwendet, um Low-Level-Objekte zu erkennen. Je tiefer wir in das CNN gehen, desto mehr Filter werden angewendet, um detailliertere Objekte zu erkennen. Die Filter werden mittels Matrixmultiplikation angewendet, um eine Funktionskarte zu erhalten. Abbildung 3.13 zeigt, wie ein Convolutional Filter angewendet wird und welche Feature-Map daraus resultiert. Nehmen wir an es werden 12 Filter auf das Beispielbild angewendet, so erhalten wir eine Feature-Map der Größe *32x32x12*.

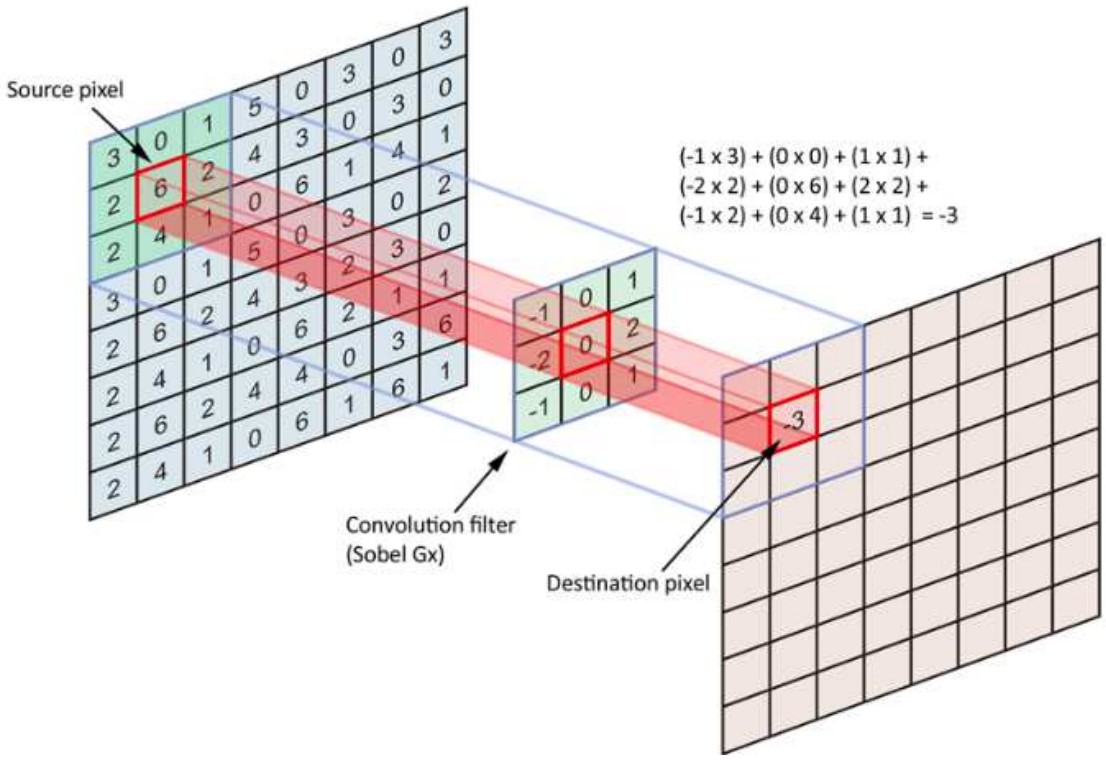


Abbildung 3.13: Anwendung von Convolutional Filter [30]

Zum besseren Verständnis betrachten wir diesen Vorgang anhand eines kleineren Beispiels. Nehmen wir an, wir haben ein Bild der Größe 5x5x1 und einen Filter der Größe 3x3x1, siehe Abbildung 3.14.

Das erste Bild aus der Abbildung 3.15 zeigt den ersten Schritt: Um den ersten Wert der resultierenden Matrix zu erhalten, wird folgende Berechnung vorgenommen: $(3 \cdot 0) + (3 \cdot 1) + (2 \cdot 2) + (0 \cdot 2) + (0 \cdot 2) + (1 \cdot 0) + (3 \cdot 0) + (1 \cdot 1) + (2 \cdot 2) = 12$. Im nächsten Schritt

Abbildung 3.14: Convolutional Filter [31]

wird die gleiche Operation durchgeführt, jedoch wird der Filter nach rechts verschoben. Dieser Vorgang wiederholt sich so lange, bis der rechte Rand der Input Matrix erreicht wurde, dann wird der Filter nach unten verschoben und der Vorgang beginnt erneut von links nach rechts. Aus einer 5x5x1 Matrix ergibt sich so eine 3x3x1 Matrix. Abbildung 3.15 veranschaulicht dieses Vorgehen [31].

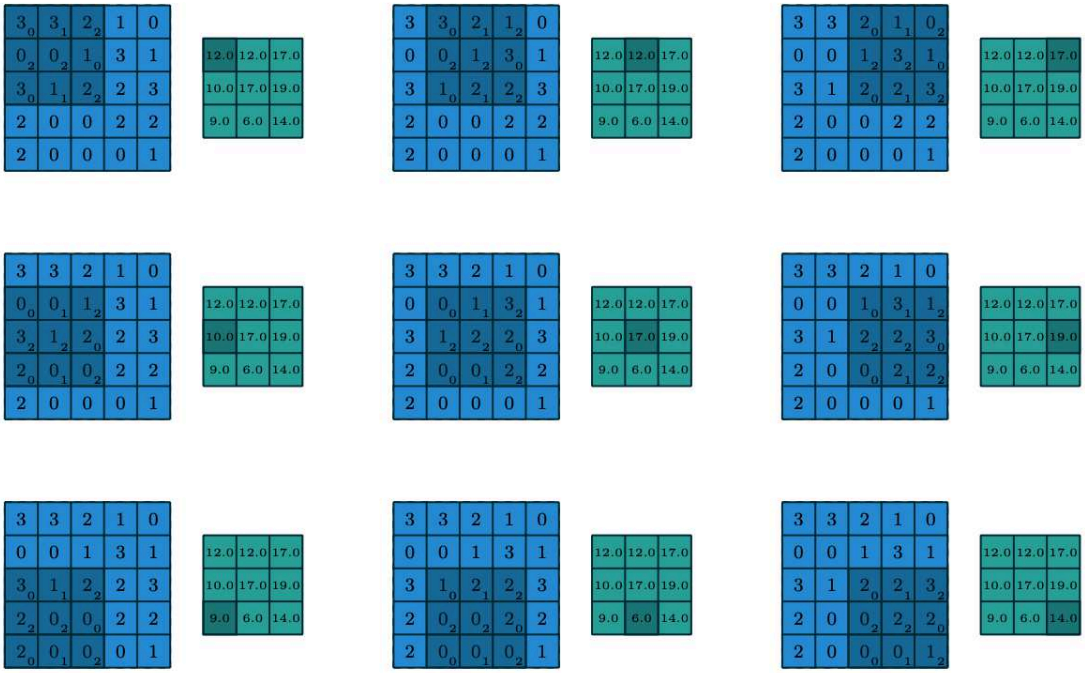


Abbildung 3.15: Anwendung von Convolutional Filtern [31]

Non Linearity (ReLU)

ReLU steht für *Rectified Linear Unit*, eine nicht lineare Operation: f(x) = max(0,x).
Die ReLU-Funktion wird angewendet, um eine Nichtlinearität in unser Convolutional Neural Network einzuführen. Damit das CNN lernen kann, braucht es eine nicht lineare Funktion [32]. Neben den ReLU-Funktionen gibt es auch andere nicht lineare Funktionen,

wie *tanh* oder *sigmoid*, die anstelle der ReLU-Funktionen verwendet werden können. Die Größe des Arrays ändert sich dabei nicht. Abbildung 3.16 zeigt ein Beispiel zum besseren Verständnis.

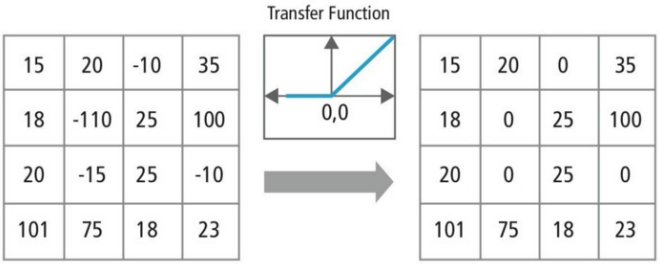


Abbildung 3.16: Non Linearity Funktion [30]

Pooling

Diese Ebene ist dafür zuständig, ein Objekt als ein Objekt zu definieren, auch wenn es in seinem Aussehen variiert; dafür wird eine Downsampling-Operation ausgeführt. Abbildung 3.17 zeigt ein Beispiel mit einem *8x8* Array, das zu einem *4x4* Array umgewandelt wird [33].

Die Input-Matrix der Größe 4x4 aus dem Beispiel in Abbildung 3.17 wird um die Hälfte reduziert. Dazu wird die Input-Matrix in vier Teile geteilt; aus jedem Bereich wird mittels Max-Pooling die größte Zahl aus diesem Bereich ausgewählt und in die neue Matrix geschrieben. Das Ergebnis ist eine neue Matrix der Größe 2x2.

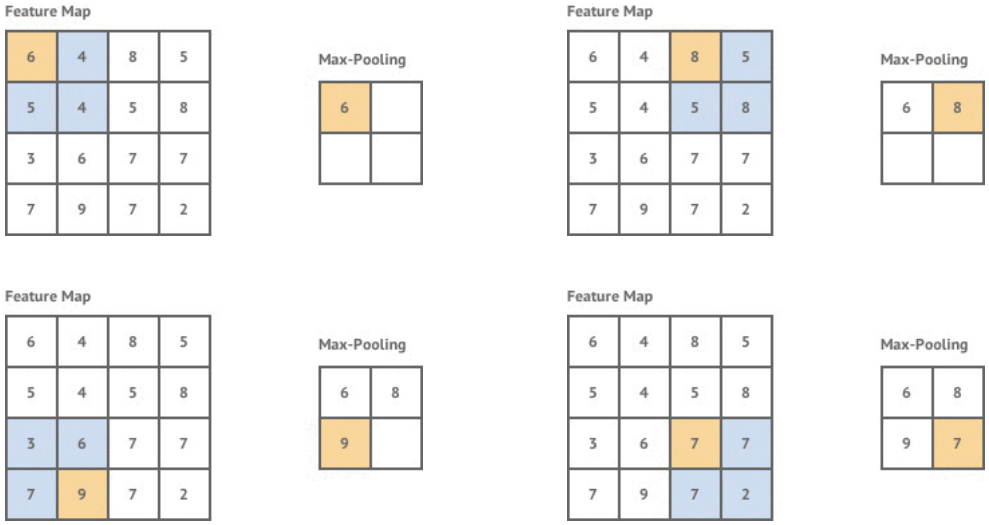


Abbildung 3.17: Downsampling Operation [30]

Fully Connected

Der Output des Poolings, die 4x4 Matrix, wird in diesem Layer zu Vektoren umgewandelt (x1, x2, x3, x4), wie in Abbildung 3.18 dargestellt. Eine Gewichtungsfunktion klassifiziert in Labels. Das Ergebnis der letzten Ebene ist ein Eindimensionaler Vektor.

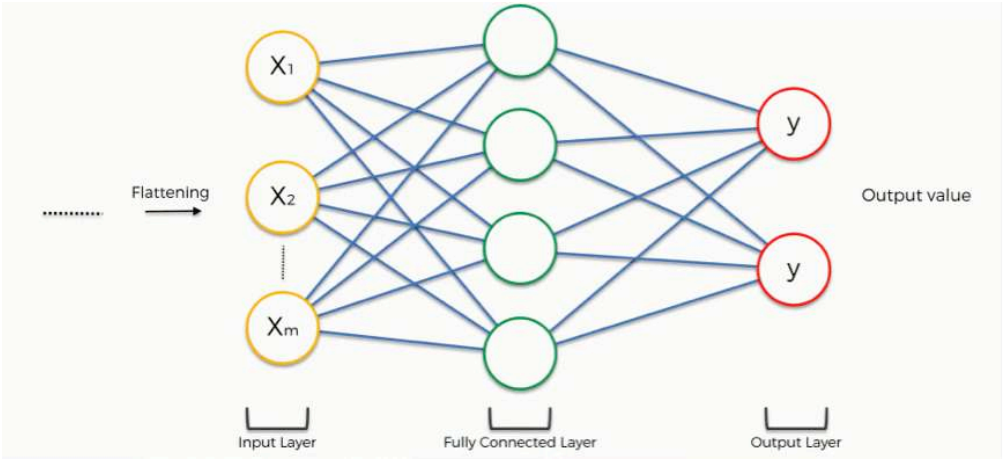


Abbildung 3.18: Fully Connected Layer [30]

Abbildung 3.19 gibt einen Überblick über die zuvor beschriebenen Ebenen des Convolutional Neural Networks.

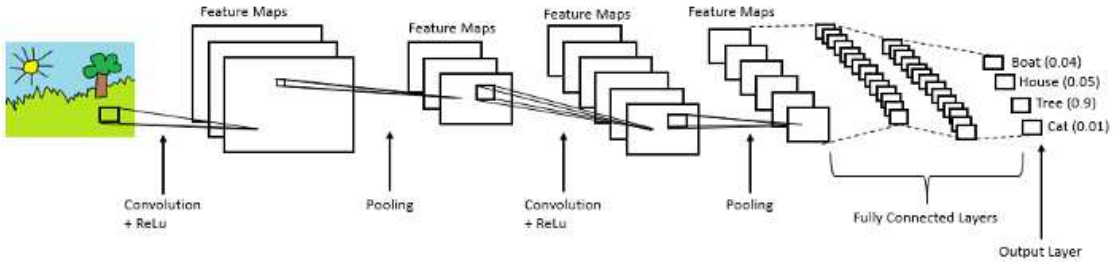


Abbildung 3.19: CNN Übersicht [30]

3.4 Source Code Generierung

In diesem Abschnitt wird ein kurzer Einblick in das Thema Source-Code-Generierung gegeben und erläutert, welcher Ansatz für diese Arbeit verwendet wurde. Die folgenden Informationen stammen aus dem Buch *Code Generation with Templates* [34] und der Arbeit *Systematic Mapping Study of Template-based Code Generation.* [35]

Seit den 1950er Jahren gibt es die Code-Generierung, die ursprünglich nur in Compilern verwendet wurde. Das Prinzip der automatischen Programmierung ist, dass der/die BenutzerIn definiert, was er vom Programm erwartet, und der Source-Code-Generator dieses Programm ohne die Unterstützung des Benutzers bzw. der Benutzerin generiert.

Automatisch generierter Source Code ist ein generischer Ansatz, bei dem der gleiche Generator mehrfach verwendet werden kann, um mittels variierender Inputs verschiedene Artefakte zu erzeugen. Die automatische Source-Code-Generierung wird verwendet, um die Entwicklungszeit zu verkürzen und die Produktivität zu steigern.

Die wichtigsten Vorteile der Source Code Generierung von User Interfaces [36]:

- **Konsistenz** – Durch die Generierung von User Interfaces werden User Interfaces generiert, die konsistent sind und der/dem BenutzerIn ein einheitliches User Interface bieten.
- **Flexibilität** – Die Verwendung eines Source Code Generators von User Interfaces bringt den Vorteil, dass bei Änderungen der Requirements oder von Schnittstellen das User Interface neu generiert werden kann.
- **Portierbarkeit** – Der Source Code Generator kann erweitert oder ausgetauscht werden, um den Source Code mit neuen Technologien zu erstellen. So ist es möglich, User Interfaces für verschiedene Plattformen zu generieren (Desktop UIs, mobile UIs oder Web-basierte User Interfaces)

Es gibt verschiedene Ansätze für die Source-Code-Generierung. Template Based Code Generation ist ein Ansatz, der einfach und schnell umgesetzt werden kann. Das Prinzip dahinter ist: *write once, produce many.* Dieses Konzept der Source-Code-Generierung wird in der Erstellung von Web-Interfaces häufig verwendet. [35]

Im Zuge dieser Arbeit wurde ein Template Based Source Code Generator erstellt. Der Source-Code-Generator erhält Informationen über die Komponenten, deren Position und das Design. Aufgrund der Daten wird automatisch eine Web-Oberfläche in HTML erstellt. Es wird auf vordefinierte Templates zugegriffen und diese werden zu einer Oberfläche kombiniert. In Abschnitt 4.2.3 wird genauer auf die Umsetzung des Source-Code-Generators für das im Zuge dieser Arbeit erstellte Tool eingegangen.

KAPITEL 4

Architektur, Implementierung und Funktionalität des entwickelten Tools

Abbildung 4.1 zeigt eine Übersicht über den Aufbau des Tools. Handskizzen werden vom DoodleClassifier eingelesen und klassifiziert und es wird deren Position bestimmt. Der DoodleClassifier leitet diese Informationen an das im Zuge dieser Arbeit erstellte Tool weiter, das die Informationen verarbeitet und ein fertiges Design erstellt. Im folgenden Kapitel wird genauer auf die einzelnen Komponenten eingegangen.

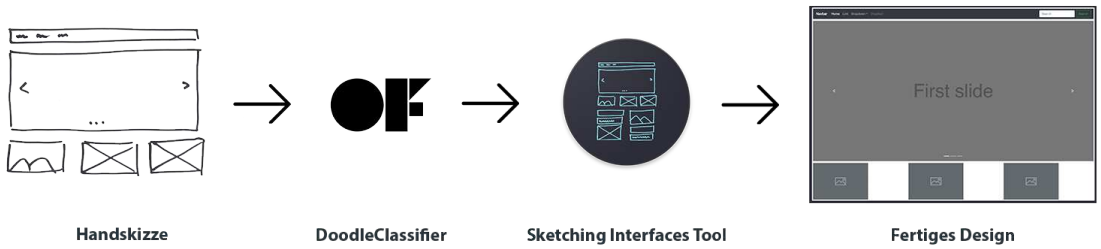


Abbildung 4.1: Tool-Übersicht

4.1 Technologie

Im folgenden Abschnitt werden die Technologien vorgestellt, die für das Erstellen des Prototypen verwendet wurden.

4.1.1 ml4a-ofx

Maschine learning for Artist (ml4a-ofx) [37] ist eine Sammlung von Echtzeitanwendungen für das Arbeiten mit Machine Learning. Alle Apps dieser Sammlung sind auf dem openFrameworks aufgebaut und in C++ implementiert. Der DoodleClassifier ist eine App aus der Sammlung mit der Handskizzen trainiert und anschließend wieder klassifiziert werden können. Im Abschnitt 4.2.1 wird der Doodle Classifier detailiert erklärt [38].

4.1.2 Web Server

Ein Web Server in Node.js [7] wird für das Umwandeln der Skizzen erstellt. Dazu ein Frontend in AngularJS für das Visualisieren der Ergebnisse der Umwandlung. Die Layout Engine und die Mapping Engine wurden in Node.js erstellt und werden im Abschnitt 4.2.2 und 4.2.3 genauer beschrieben.

4.1.3 Electron

Electron ist ein von GitHub [39] entwickeltes Opensource Framework, mit dem plattformübergreifende Desktop Applikationen in HTML, JavaScript und CSS geschrieben werden können. Es wurde mit Chromium und Node.js erstellt [8]. Ursprünglich wurde das Framework 2013 für den Texteditor Atom entwickelt und 2014 unter einer Open-Source-Lizenz veröffentlicht [40, 41].

4.1.4 OSC - Open Sound Control

Das Open Sound Control Protokoll wurde ursprünglich für die Kommunikation zwischen Computern und Sound Synthesizern sowie zwischen Computern und anderen Multimediageräten entwickelt [42].

Das OSC-Protokoll verwendet ein hierarchisches Namensschema ähnlich der URL Notation. Das Protokoll definiert keine spezifischen Objekte, die in der Hierarchie vorkommen sollen, daher bleibt es dynamisch. Jedes System, das das OSC-Protokoll verwendet, implementiert seine eigene Adresshierarchien. Es handelt sich beim OSC-Protokoll um ein open-ended Protokoll, was bedeutet, dass keine direkten Ziele angesprochen werden. Jeder Empfänger muss daher selbst bestimmen, ob die Nachricht an ihn gerichtet ist, dafür wird eine Pattern-matching Syntax verwendet, die ähnlich den Regular Expressions ist [42].

4.1.5 WebSocket

Der Websocket ist ein auf TCP aufgebautes Protokoll das eine bidirektionale Verbindung zwischen Server und Client aufbaut. Dadurch können Informationen vom Server an die Webseite gesendet werden, aber auch von der Webseite an den Server [43, 44]. Im Zuge dieser Arbeit wurde der Websocket verwendet, um die Änderungen der gescannten Skizzen an das Frontend zu senden, um immer die neuste Version der Skizze in der umgewandelten HTML-Ausgabe anzuzeigen.

4.2 Architektur und Implementierung

In diesem Abschnitt wird die Architektur des Tools sowie die verschiedenen Kommunkationsflüsse und Datenstrukturen des Tools vorgestellt:

Abbildung 4.2 zeigt einen Gesamtüberblick über die Architektur des Tools.

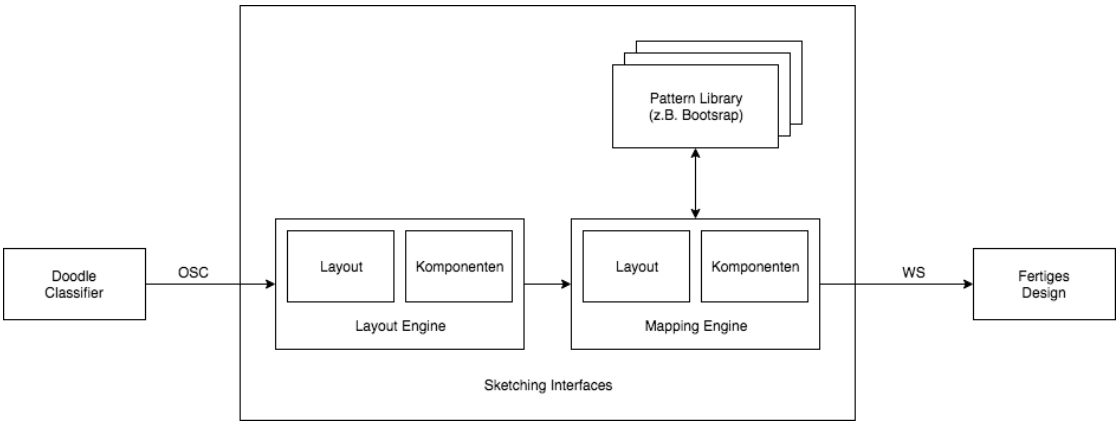


Abbildung 4.2: Architektur

4.2.1 Doodle Classifier

Der DoodleClassifier ist eine openFrameworks Application aus der *ml4a-ofx* Kollektion, die es ermöglicht, Handzeichnungen über eine Kamera zu trainieren und anschließend wieder zu erkennen und zu klassifizieren. Der DoodleClassifier wurde erstmals in einem Projekt namens DoodleTunes verwendet, in dem Musikinstrumente gezeichnet und in Ton Outputs umgewandelt wurden [37].

Für die Verwendung des DoodleClassifiers wird eine Kamera benötigt, die die Bilder zum Trainieren und für das anschließende Erkennen aufzeichnet und an den DoodleClassifier weiterleitet.

Der DoodleClassifier verwendet für die Kommunikation das OSC (open sound control) Protokoll. Dies ermöglicht das einfache Anbinden von dritten Applikationen an den DoodleClassifier.

Listing 4.1 zeigt ein Beispiel einer OSC Message wie sie vom Doodle Classifier weitergeleitet wird.

Listing 4.1: OSC Protokoll Beispiel Message

```
[ 'button', 5725, 301, 306, 120, 71,
  'picture', 7485, 433, 294, 115, 79,
  'navbar', 11506.5, 151, 81, 380, 39
]
```

Die Parameter aus der OSC Message: *Label, Area, X-Koordinate, Y-Koordinate, Breite und Höhe der Komponente* werden im Abschnitt 4.3 beschrieben.

Um die Bilder über die Kamera bestmöglich zu erkennen, gibt es einige Einstellungen im DoodleClassifier:

- **Threshold:** Bestimmt den Helligkeitsschwellwert zwischen Vordergrund und Hintergrund, um den Vordergrund vom Hintergrund zu trennen. Um dies zu verbessern, können auch Lichter verwendet werden, um die Zeichnung zu beleuchten und z.B. Schatten zu vermeiden, die möglicherweise zu fehlerhaften Inputs führen können.
- **Dilations:** Bestimmt die Dicke der erkannten Linien.
- **Min area und Max area:** Bestimmt die akzeptable Größe der Zeichnungen, die erkannt werden sollen. Setzt man die Min area zu klein werden Objekte erkannt, die keine sind, z.B. ein Staubkorn.

Training

Zum Trainieren der App benötigt es möglichst viele Beispiele für eine Klasse.

Klassen sind Gruppierungen von Objekten mit gleichen Eigenschaften. In unserem Fall ist eine Klasse eine UI Komponente, z.B. ein Button, Input oder ein Text.

Je größer die Anzahl der Beispiele, desto genauer kann das Objekt später einer Klasse zugewiesen werden. Im initialen Setup vom DoodleClassifier werden drei Objekte verwendet: Kreis, Stern und Pfeil.

Der DoodleClassifier kann mittels Kontrollfelder, Rot umrandete Bereich in der Abbildung 4.3, bedient werden. Das erste Feld ist ein Schieberegler über diesen Regler können die verschiedenen Klassen ausgewählt werden. Der Schieberegler ist für das Trainieren von Klassen notwendig [45]. Bei der Abbildung 4.3 ist die Klasse *circle* ausgewählt. Mit Stift und Papier können Beispielskizzen für diese Klasse gezeichnet und dann über die

Schaltfläche *add samples* in den DoodleClassifier eingelesen werden. Je größer die Anzahl der Skizzen für eine Klasse, desto besser kann anschließend klassifiziert werden. Dieser Vorgang muss für alle definierten Klassen wiederholt werden. Sobald alle Klassen ihre Beispielbilder eingelesen haben, kann über die Schaltfläche *Training* das Training gestartet werden. Es dauert je nach Anzahl der Klassen und Beispielbilder einige Sekunden bis Minuten. Um diesen Vorgang nicht bei jeder Ausführung des DoodleClassifiers wiederholen zu müssen, können die Trainingsdaten über die Schaltfläche *Save* gespeichert und über die Schaltfläche *Load* beim nächsten Start des Programms geladen werden.

In Abbildung 4.3 sieht man ein Screenshot des Interfaces des DoodleClassifiers. Das Originalbild der Kamera wird in der Applikation links oben angezeigt, rechts daneben das Threshold Bild. Durch das Modifizieren des Threshold Parameters kann mit diesem Bild gut erkannt werden, wann der optimale Wert für das Setup erreicht ist. Der optimale Wert für das Setup ist dann erreicht, wenn alle skizzierten Objekte im Threshold Bild gut zu erkennen sind und sich deutlich vom Hintergrund abheben. Wird der Threshold Wert zu hoch oder zu niedrig gewählt, können die verschiedenen Farben nicht voneinander abgehoben und die Skizzen nicht erkannt werden. Werden die skizzierten Komponenten richtig erkannt, dann werden sie mit einem grünen Quadrat umrandet und können klassifiziert werden. Das Bild oben rechts zeigt die erkannten Objekte, die mit einem grünen Quadrat um das Objekt gekennzeichnet sind.

Für das Erstellen des Tools wurde der Doodle Classifier mit den Trainingsdaten im Anhang trainiert: Button (A.14), Image (A.16), Carousel (A.17), Input (A.15), Info Card (A.21), Table (A.20), Text (A.19) und Navbar (A.18). Es wurden nur sehr wenige Trainingsdaten verwendet, nämlich 6 - 10 Skizzen pro Komponente. Dies war aber ausreichend, um die Skizzen anschließend klassifizieren zu können.

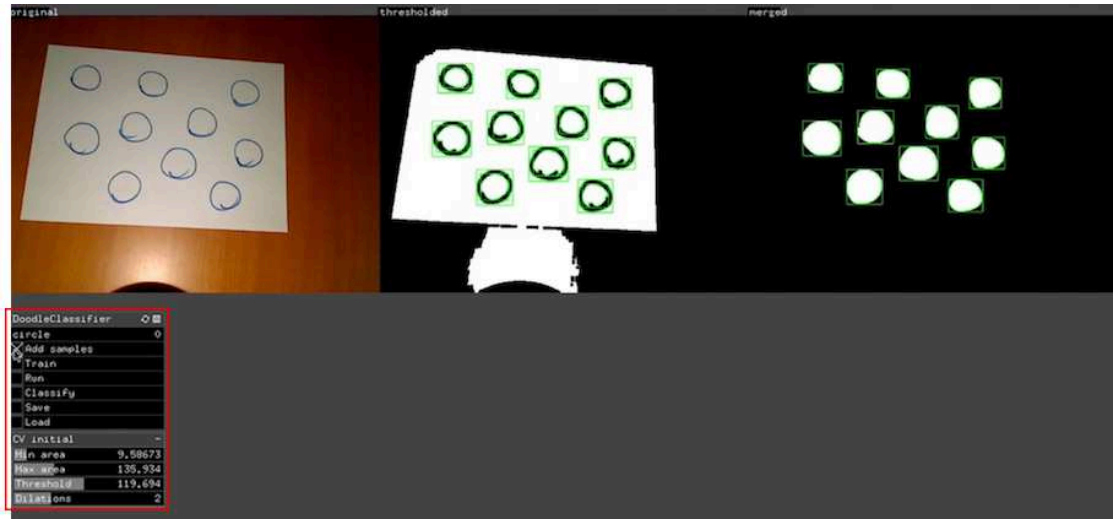


Abbildung 4.3: DoodleClassifier Interface [37]

Prediction

Wenn das Training mit allen Objekten abgeschlossen ist, kann der DoodleClassifier diese erkennen. In der Abbildung 4.4 ist zu erkennen, wie die zuvor trainierten Objekte (Kreise, Sterne und Pfeile) nun auf einer neuen Zeichnung erkannt werden. Der DoodleClassifier zeigt neben dem erkannten Objekt noch die Position der x- und y-Koordinaten des erkannten Objektes sowie die Größe an.

Die Klassifizierung der Skizzen im DoodleClassifier kann auf zwei Arten gestartet werden. Mit der Schaltfläche *run* werden die Kameraaufnahmen periodisch gescannt und klassifiziert. Mit der Schaltfläche *classify* wird hingegen das aktuelle Bild der Kamera klassifiziert.

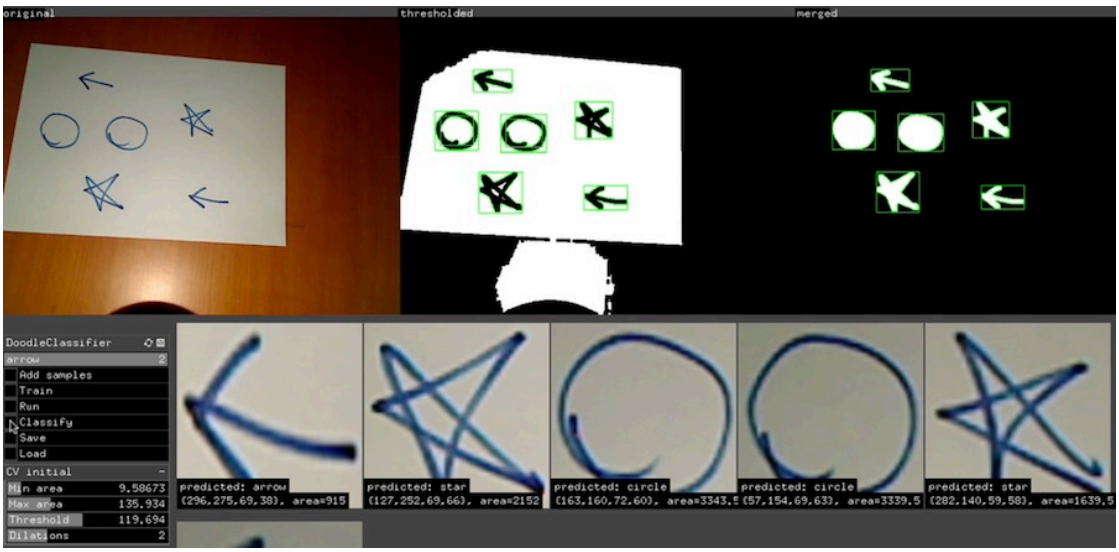


Abbildung 4.4: DoodleClassifier Klassifizierung [37]

Anpassung des Doodle Classifiers

Der Doodle Classifier wurde mit den UI Komponenten aus dem Abschnitt 3.2.2 trainiert. Dazu mussten weitere Klassen definiert und in der Doodle Classifier Config eingefügt werden.

Um alle benötigten Daten vom DoodleClassifier zu erhalten, musste dieser angepasst werden. Die Änderungen des DoodleClassifiers werden im Abschnitt 4.3 genauer beschrieben. Bei jeder Klassifizierung werden die Daten an die Sketching Interfaces Applikation weitergeleitet.

Folgende Werte werden vom DoodleClassifier ermittelt und weitergeleitet, wir betrachten diese anhand des Beispiels in Abbildung 4.6:

- **Label**: Das Objekt, das vom DoodleClassifier identifiziert wird, in diesem Beispiel: *Button*.
- **Area**: Die Größe des Objektes in Pixel. Es wird der Konturenbereich berechnet, der sich aus der Fläche der Form des Objektes ergibt. Im Beispiel 4.6 handelt es sich um die Breite mal Höhe des Buttons. Sollte der Button jedoch etwas schief gezeichnet werden, so wird nicht die Breite mal Höhe des Aufnahmebereichs berechnet, sondern die Fläche des Konturenbereichs des Buttons, also die Fläche des Buttons und nicht die Fläche des Aufnahmebereichs. In der Abbildung 4.6 rot dargestellt.

  Für ein besseres Verständnis betrachten wird das Beispiel in Abbildung 4.5, hier wird der *Area* Wert nicht anhand der grünen Umrandung des Kreises (des Aufnahmebereichs) berechnet, sondern anhand der Fläche des Kreises in Pixel (weißer Bereich).

Abbildung 4.5: Doodle Classifier – Area Wert [38]

- **X-Koordinate**: Die x-Koordinate vom linken oberen Punkt der Komponente zum linken Rand des aufgenommenen Bereichs in Pixel.
- **Y-Koordinate**: Die y-Koordinate vom linken oberen Punkt der Komponente zum oberen Rand des aufgenommenen Bereichs in Pixel.
- **Width**: Die Breite des Objekts in Pixel.
- **Height**: Die Höhe des Objekts in Pixel

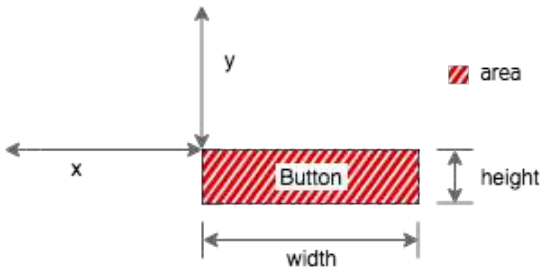


Abbildung 4.6: Ermittelte Werte vom DoodleClassifier

Diese Daten werden an die Sketching Interfaces Applikation weitergeleitet, die diese Daten benötigt, um ein Layout zu erstellen.

Nachstehend wird ein Beispiel für eine OSC Message dargestellt, die an die Sketching Interfaces Applikation weitergeleitet wird:

Listing 4.2: OSC Message vom Doodle Classifier

```
[ 'button', 5725, 301, 306, 120, 71,
  'picture', 7485, 433, 294, 115, 79,
  'navbar', 11506.5, 151, 81, 380, 39
]
```

4.2.2 Layout Engine

Die Layout Engine ist zuständig für das Erkennen des Layouts der Zeichnung. Sie muss erkennen, wie die verschieden gezeichneten UI Komponenten angeordnet sind. Dafür stehen die Information vom DoodleClassifier zur verfügung die im Abschnitt 4.2.1 beschrieben sind. Anhand des Beispiels in der Abbildung 4.7 wird die Funktionsweise der Layout Engine beschrieben.

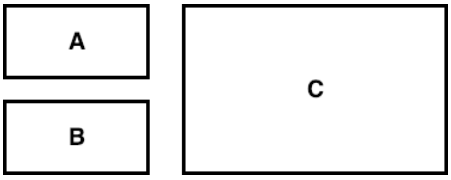


Abbildung 4.7: Layout Engine Beispiel

Das Erstellen des Layouts ist in Unterschritte unterteilt:

Finden aller Komponenten auf der gleichen Y-Achse

Alle Komponenten, die vom DoodleClassifier empfangen wurden, werden nach deren X-Position sortiert. Als erstes wird überprüft, ob sich zwei Komponenten nebeneinander befinden, dabei wird der Y-Wert der Komponenten unter Verwendung einer Toleranz überprüft. Der Toleranzwert liegt bei 25 Pixel und kann in der *layoutEngineConfig.json* Datei verändert werden. Mit diesem Schritt konnten im Beispiel in der Abbildung 4.8 die Komponenten A und C erkannt werden, und es konnte bestimmt werden, dass diese sich nebeneinander befinden.

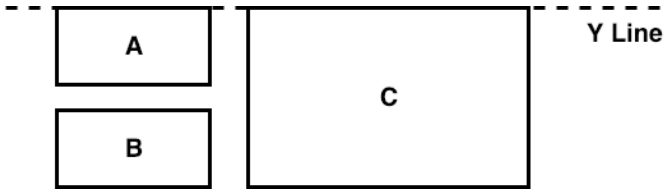


Abbildung 4.8: Layout Engine Beispiel – Y-Linie

Untersuchen, ob andere Komponenten sich in einem bereits kombinierten Objekt befinden, und diese gegebenenfalls zusammenführen

Um die Komponente B korrekt zuzuweisen, wird der Schritt des Kombinierens durchgeführt. A und C befinden sich bereits in einem kombinierten Objekt. Es wird berechnet, wie groß das neu entstandene kombinierte Objekt ist, damit lässt sich bestimmen, ob weitere Objekte, die sich noch nicht in diesem kombinierten Objekt befinden, diesem ebenfalls zugewiesen werden müssen. Dazu wird der Startpunkt F der Komponente B betrachtet, die X- und Y-Position dieses Punktes muss sich innerhalb des kombinierten Objektes befinden. Die Position von X und Y des Startpunkts F der Komponente B muss größer als die X- und Y-Position des kombinierten Objektes sein, siehe Punkt D auf dem Beispielbild 4.9, und kleiner als X + die Breite des kombinierten Objektes und kleiner als Y + die Höhe des kombinierten Objektes, siehe Punkt E im nachfolgenden Beispielbild. Dadurch kann bestimmt werden, ob sich das Objekt innerhalb oder außerhalb des kombinierten Objekts befindet. Wenn es innerhalb des kombinierten Objekts liegt, wird es dem kombinierten Objekt zugewiesen. Wenn es außerhalb des kombinierten Objekts liegt, ist es kein Teil davon sondern ein eigenständiges Objekt oder ein neues kombiniertes Objekt.

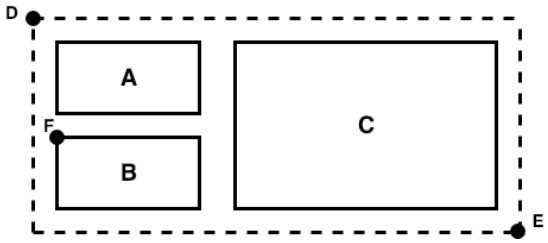


Abbildung 4.9: Layout Engine Beispiel – Kombinieren von Objekten

Der Source Code Ausschnitt 4.3 zeigt den Algorithmus der für das Erkennen weiterer Objekte in einem kombinierten Objekt zuständig ist. Dazu wird über alle skizzierten Objekte iteriert; handelt es sich bei dem Objekt um ein kombiniertes Objekt, kann das darauffolgende Objekt dahingehend überprüft werden, ob es sich im Bereich des kombinierten Objektes befindet. Der Funktion *isPointInsideRectangle* werden zwei skizzierte Objekte übergeben, sie liefert *TRUE*, sollte sich das zweite Objekt im Bereich des ersten Objektes befinden. In diesem Fall werden sie dem kombinierten Objekt hinzugefügt, ansonsten werden sie als eigenständige Objekte gespeichert.

Listing 4.3: Finden aller Objekte, die sich im kombinierten Bereich befinden

```
function findObjectsWithPositionInOtherObject(sketchedObjects) {
    let result = [];
    for (let i = 0; i < sketchedObjects.length; i++) {
        if (sketchedObjects[i].label=='combined' &&
            isPointInsideRectangle(sketchedObjects[i],
            sketchedObjects[i+1])) {
```

```
                sketchedObjects[i].objects.push(sketchedObjects[i+1]);
                result.push(sketchedObjects[i]);
                i++;
            } else
                result.push(sketchedObjects[i]);
        }
        return result
    }

    function isPointInsideRectangle(firstObject, secondObject) {
        if (firstObject.rect.x > secondObject.rect.x)
            return false;
        if (firstObject.rect.y > secondObject.rect.y)
            return false;

        return secondObject.rect.x < (firstObject.rect.x +
            firstObject.rect.width) && secondObject.rect.y < (firstObject.rect.y
            + firstObject.rect.height);
    }
```

Das Ergebnis der Zusammenführung der Komponenten aus dem Beispiel 4.9 wird in Abbildung 4.10 dargestellt. Nach dem Zusammenführen der Komponenten sind diese in einem kombinierten Objekt vereint, jedoch noch nicht an der richtigen Position. Das Positionieren wird dann im nächsten Schritt durchgeführt.

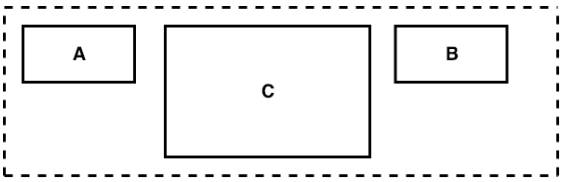


Abbildung 4.10: Layout Engine Beispiel – Kombinieren von Objekten, Resultat

Teilen des kombinierten Objektes in einen rechten und einen linken Teil

Um die Komponenten in die richtige Reihenfolge und Position zu bringen, wird nun jedes kombinierte Objekt genauer betrachtet und in eine linke und eine rechte Hälfte geteilt. Betrachten wir das Beispiel in der Abbildung 4.11 kann man beobachten, dass sich nach dem ersten Teilen des kombinierten Objektes nur ein Objekt im linken Teil befindet, im rechten Teil hingegen zwei. Nun wird die Komponente B betrachtet und aufgrund ihrer X- und Y-Position bestimmt, ob dieses Objekt dem linken oder rechten Teil zugewiesen werden kann. In diesem Beispiel wird die Komponente B dem linken Teil zugewiesen. Am Ende erhalten wir, wie in der Abbildung 4.12 zu sehen, ein kombiniertes Objekt mit jeweils zwei wiederum kombinierten Objekten, dem rechten und linken Unterobjekt. Im linken Objekt befinden sich zwei Komponenten, A und B, und im rechten Objekt die Komponente C.

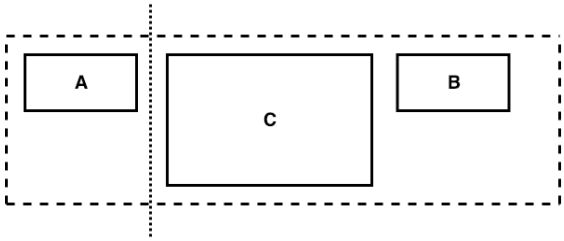


Abbildung 4.11: Layout Engine Beispiel – Teilen

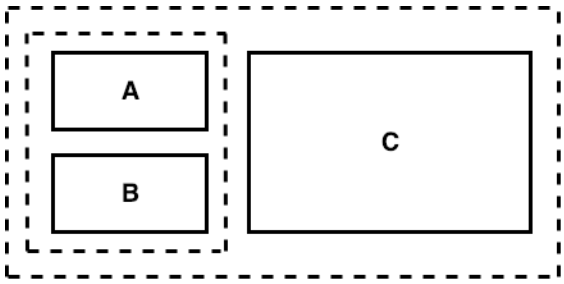


Abbildung 4.12: Layout Engine Beispiel – Resultat

Der Algorithmus im Codeausschnitt 4.4 zeigt diesen Vorgang. Es wird über alle skizzierten Objekte iteriert. Ist das Objekt ein kombiniertes Objekt und besitzt es mehr als 2 Komponenten, wird der Schritt des Teilens durchgeführt. Zwei neue kombinierte Objekte werden erstellt, ein linkes und ein rechtes kombiniertes Objekt. Im linken kombinierten Objekt befindet sich nach diesem Vorgang das erste Objekt des ursprünglichen kombinierten Objektes und im rechten kombinierten Objekt das zweite. Anschließend wird über die restlichen Objekte des ursprünglichen kombinierten Objektes iteriert, ohne die ersten zwei Objekte zu betrachten. Die *splitXPosition* ergibt sich aus der X-Position des zweiten Objektes. Dieser Punkt wird als der *Teilungs Punkt* zwischen dem linken und rechten Teil verwendet. Aufgrund dieses Punktes werden die restlichen Objekte entweder dem rechtem oder dem linkem Teil zugewiesen.

Listing 4.4: Teilen der kombinierten Objekte in einen linken und einen rechten Teil

```
function splitToLeftAndRight(sketchedObjects) {
        let result = [];
    for (let j = 0; j < sketchedObjects.length; j++) {
        if (sketchedObjects[j].label === 'combined') {
            if (sketchedObjects[j].objects.length > 2) {
                let combined = {label: "combined", objects: []};

                let combinedLeft = {label: "combined", objects: []};
                combinedLeft.objects.push(sketchedObjects[j].objects[0]);

                let combinedRight = {label: "combined", objects: []};
                combinedRight.objects.push(sketchedObjects[j].objects[1]);
```

```
                let splitXPosition = sketchedObjects[j].objects[1].rect.x;
                for (let i=2; i < sketchedObjects[j].objects.length; i++) {
                    if (sketchedObjects[j].objects[i].rect.x <
                        splitXPosition)
                        combinedLeft.objects.push(
                            sketchedObjects[j].objects[i]);
                    else
                        combinedRight.objects.push(
                            sketchedObjects[j].objects[i]);
                }

                combined.objects.push(combinedLeft);
                combined.objects.push(combinedRight);
                result.push(combined);
            } else {
                result.push(sketchedObjects[j]);
            }
        } else {
            result.push(sketchedObjects[j])
        }
    }
    return result;
}
```

Output:

Der Codeausschnitt 4.5 zeigt den Output der Layout Engine, der an die Mapping Engine weitergegeben wird. Der Codeausschnitt gibt das Beispiel in der Abbildung 4.7 wieder. Die Komponente A wird als Input dargestellt, B als Button und C als Bild:

Listing 4.5: Layout Engine Output

```
[
  {
    "label": "combined",
    "objects": [
      {
        "label": "combined",
        "objects": [
          {
            "label": "input",
            "area": 10917.5,
            "rect": { "x": 122, "y": 25, "width": 195, "height": 65 }
          },
          {
            "label": "button",
            "area": 9704,
            "rect": { "x": 122, "y": 93, "width": 192, "height": 61 }
          }
        ],
```

```
        "rect": { "x": 122, "y": 25, "width": 195, "height": 129 }
      },
      {
        "label": "combined",
        "objects": [
          {
            "label": "picture",
            "area": 20552.5,
            "rect": { "x": 315, "y": 28, "width": 182, "height": 130 }
          }
        ],
        "rect": { "x": 315, "y": 28, "width": 182, "height": 130 }
      }
    ],
    "rect": { "x": 122, "y": 25, "width": 375, "height": 133 }
  }
]
```

Das Tool bietet die Möglichkeit, das Layout einzublenden. Dadurch wird das Layout in Form von strichlierten Linien sichtbar. Diese Funktion kann in den Einstellungen des Tools aktiviert werden.

4.2.3 Mapping Engine

Die von der Layout Engine extrahierten Informationen über Komponenten und deren Position werden von der Mapping Engine in das fertige Design umgewandelt, wobei die Mapping Engine auf die vordefinierten Mappings im JSON Format zugreift. Die Mappings sind so aufgebaut, dass verschiedene Pattern Libraries verwendet werden können.

Im Zuge dieser Arbeit wurden zwei Pattern Libraries eingebunden, das Bootstrap Framework und das Material Design Lite Framework.

Mapping Format

Das Format für das Mapping ist wie folgt aufgebaut:

Listing 4.6: Mapping Datei

```
{
  "imports": {
    "header": [
      "../node_modules/bootstrap/dist/css/bootstrap.min.css"
    ],
    "script": [
      "../node_modules/jquery/jquery.min.js",
      "../node_modules/bootstrap/dist/js/bootstrap.min.js"
    ]
  },
  "start": "<div class=\"components\">",
```

```
  "start-layout-visible": "<div class=\"components layout\">",
  "end": "</div>",
  "row-start": "<div class=\"row\">",
  "row-start-layout-visible": "<div class=\"row layout-row\">",
  "row-end": "</div>",
  "col-start": "<div class=\"col\">",
  "col-start-layout-visible": "<div class=\"col layout-col\">",
  "col-end": "</div>",
  "button": "<button type=\"button\" class=\"btn\">Button</button>",
  "info card": "<div class=\"card\" ...",
  "input": "<input type=\"email\" ...",
  "picture": "<img style=\"width: 300px; hight: 200px\" src=\"image.png\">",
  "carousel": "<div id=\"carousel\" class=\"carousel slide\" ...",
  "navbar": "<nav class=\"navbar navbar-expand-lg bg-dark\"> ..."
}
```

- **Imports**: Da die Frameworks verschiedene online Ressourcen benötigen, um korrekt zu funktionieren, können diese über die Mapping Datei eingebunden werden. Es wird zwischen *header* und *scripts* unterschieden. Im Abschnitt *header* können alle CSS Ressourcen eingebunden werden, im Abschnitt *scripts* können alle Script Dateien des Frameworks eingebunden werden. Im Beispiel 4.6 wird das Framework Bootstrap verwendet. Dies verlangt das Einbinden einer CSS Datei und von zwei Script Dateien.

  Durch das Einbinden der Ressourcen über die Mapping Datei, ist es möglich, verschiedene Frontend Frameworks zu verwenden.

- **start** und **end**: Jede Komponente, die vom Tool eingefügt wird, wird in einen *start* und *end* Tag eingepackt.
- **row-start** und **row-end**: HTML *div* Tag mit der CSS Classe *row*.
- **col-start** und **cole-end**: HTML *div* Tag mit der CSS Classe *col*.
- **button**: Implementierung des Buttons in HTML.
- **info card**: Implementierung einer Info Card in HTML.
- **input**: Implementierung eines Textfeldes für die Eingabe des/der Benutzers/Benutzerin in HTML.
- **picture**: HTML Tag zum einfügen eines Bildes, es wird ein Platzhalterbild eingefügt.
- **carousel**: Implementierung eines Carousels in HTML.
- **navbar**: Implementierung einer Navigationsleiste in HTML.

Es werden die folgenden Mappings verwendet, um das erstellte Layout zu visualisieren, dazu werden zusätzliche CSS-Klassen eingebunden:

- **start-layout-visible:** Die Klasse *layout* aus dem Listing 4.7 wird verwendet, um das Layout zu visualisieren.
- **col-start-layout-visible:** Die Klasse *layout-col* wird verwendet.
- **row-start-layout-visible:** Die Klasse *layout-row* wird verwendet.

Listing 4.7: Layout visualisierungs CSS

```
.layout {
     border: thin blue dashed;
     margin: 2px
 }
.layout-row {
    border: thin green dashed;
    margin: 2px;
    padding: 10px;
}
.layout-col {
    border: thin red dashed;
    margin: 2px;
    margin-left: 14px;
    margin-right: 14px;
}
```

Die Abbildung 4.13 zeigt das fertige Design mit dem sichtbaren Layout. Das Layout wird in Form von strichlierten Linien dargestellt.

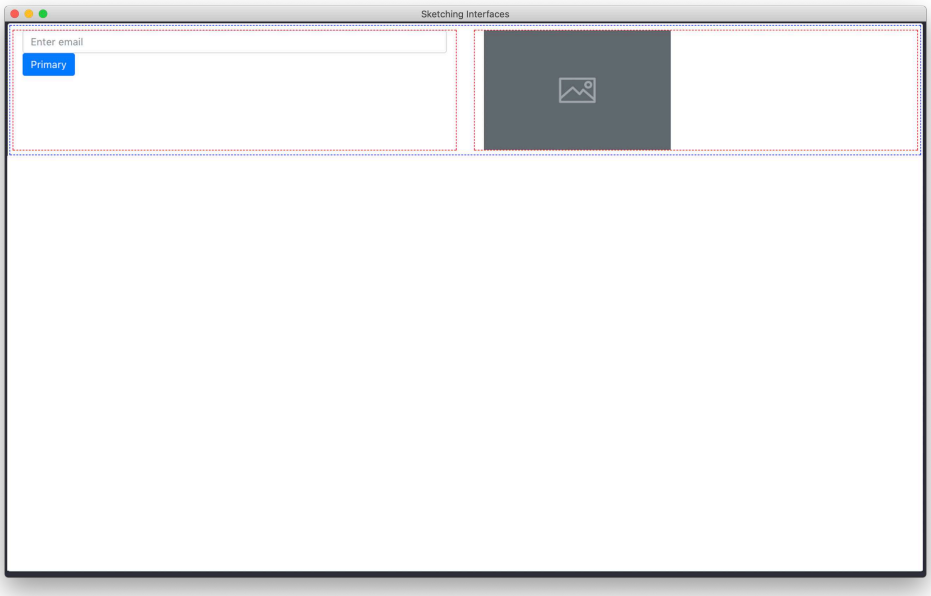


Abbildung 4.13: Fertiges Design mit sichtbarem Layout

Generieren des Source Codes

Konkret werden im Code zuerst die benötigten Libraries eingebunden. Es werden die Imports aus der Mapping-Datei geholt und zuerst die CSS-Dateien eingebunden, anschließend werden die Script-Dateien eingebunden. Im Codeausschnitt 4.8 wird abgebildet, wie die Libraries eingebunden werden:

Listing 4.8: Code generation von Header und Scripts

```
function addHeaderAndScripts() {
    let headerImports = mapping["imports"]["header"];
    let scriptImports = mapping["imports"]["script"];
    let generated = "";

    for (let i in headerImports) {
        generated += `<link rel="stylesheet" href="${headerImports[i]}">`
    }
    for (let i in scriptImports) {
        generated += `<script src="${scriptImports[i]}"></script>`
    }
    return generated;
}
```

Im Listing 4.9 ist ein Codeausschnitt abgebildet, der für das Generieren des Source Codes verwendet wird:

Listing 4.9: Code generation

```
function generate(sketchObject, callback) {
    let generated = "";
    generated += addHeaderAndScripts();
    for (let i in sketchObject) {
        if (sketchObject[i].label === "combined") {
            generated += generateCombined(sketchObject[i]);
        } else {
            if (layoutVisible)
                generated += mapping["start-layout-visible"];
            else
                generated += mapping["start"];

            generated += mapping[sketchObject[i].label];
            generated += mapping["end"];
        }
    }
    callback(generated);
}

function generateCombined(sketchObject) {
    let generated = "";
    if (layoutVisible)
        generated += mapping["start-layout-visible"];
    else
        generated += mapping["start"];
```

```
    if (layoutVisible)
        generated += mapping["row-start-layout-visible"];
    else
        generated += mapping["row-start"];

    for (let j in sketchObject.objects) {
        if (layoutVisible)
            generated += mapping["col-start-layout-visible"];
        else
            generated += mapping["col-start"];

        generated += addGeneratedPart(sketchObject.objects[j]);
        generated += mapping["col-end"];
    }
    generated += mapping["row-end"];
    generated += mapping["end"];
    return generated;
}

function addGeneratedPart(object) {
    let generated = "";
    if (object.label === 'combined') {
        for (let j in object.objects) {
            generated += addGeneratedPart(object.objects[j]);
        }
    } else
        generated += mapping[object.label];
    return generated;
}
```

Mit der Funktion *generated* aus dem Code Listing 4.9 wird die HTML-Seite generiert. Die Variable *generated* speichert einen String, der am Ende der Funktion zurückgegeben wird und die Übersetzung der Skizze in das HTML-Format enthält.

Im ersten Schritt werden alle Header und Skripts über die Funktion *addHeaderAndScripts* hinzugefügt. Das Array *sketchObject* enthält alle Informationen zu den erkannten User-Interface-Komponenten. Es wird über das Array iteriert: Handelt es sich um ein kombiniertes Objekt wird die Funktion *genertateCombined* aufgerufen und das Resultat zum *generated* String hinzugefügt. Handelt es sich um eine einfache UI-Komponente, wird mit dem Generieren fortgefahren.

Die Variable *mapping* enthält die Informationen aus der Mapping-Datei, siehe Code Listing 4.6. Je nachdem, ob die Variable *layoutVisible* true oder false ist, wird das Layout im fertigen Design visualisiert. Vor jeder UI-Komponente wird ein *start* String aus der Mapping-Datei eingebunden und am Ende jeder UI-Komponente ein *end* String. Diese beiden Strings umschließen die einzelnen Komponenten. Für den Start-String wird bei $layoutVisible = FALSE$ ein <div class=”components» verwendet und für den End-String . Ist die Variable $layoutVisible = TRUE$ wird eine zusätzliche CSS-Klasse miteingebunden, die eine gestrichelte Linie um die HTML-Elemente *div* zeichnet. Zwischen *start* und *end* wird die tatsächliche Komponente auf Basis ihres Labels eingebunden.

Ein kombiniertes Objekt besteht immer aus einem linken und einem rechten Teil. Der linke und der rechte Teil des kombinierten Objekts müssen vertikal angeordnet werden, was mit dem HTML Tag *div* und den CSS Klassen *col* und *row* realisiert wird: Die Funktion *generateCombined* funktioniert ähnlich wie die zuvor beschriebene Funktion, jedoch wird hier ein weiterer HTML-Tag eingeführt, der die gesamten Komponenten aus dem kombinierten Objekt umschließt. Es wird die CSS-Klasse *row* verwendet. Jede UI-Komponente aus dem kombinierten Objekt wird ebenfalls mit einem eigenen HTML-Tag mit der CSS-Klasse *col* umschlossen. Bei der CSS-Klasse *row* handelt es sich um eine Zeile und bei der CSS-Klasse *col* um die Spalten. Abbildung 4.14 veranschaulicht diesen Vorgang des Umschließens des Objekts, das zum Anordnen der Komponenten aus dem kombinierten Objekt dient.

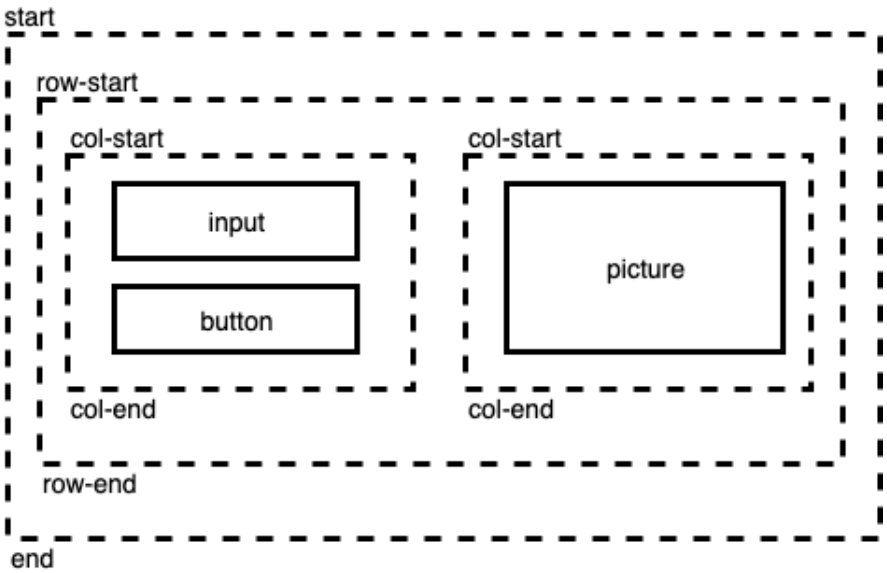


Abbildung 4.14: Beispiel Mapping Engine: HTML-Code erstellen

Das Beispiel aus dem Code Listing 4.5 würde folgende Mappings in dieser Reihenfolge aus der Mapping-Datei im Listing 4.6 abrufen:

Listing 4.10: Code-Generierung

```
start
        row-start
                col-start
                        input
                        button
                col-end
                col-start
                        picture
                col-end
        row-end
end
```

Am Ende der Funktion wird der fertige HTML-Code als String zurückgegeben und anschließend an das Frontend über den Web Socket weitergegeben, um es zu visualisieren.

4.2.4 Fertiges Design

Der Output des Tools ist ein *fertiges Design*; alle Komponenten und Layout Eigenschaften, die aus der Skizze extrahiert wurden, werden vom Tool in eine HTML-Datei umgewandelt und im Tool angezeigt.

Das Tool besitzt eine Benutzeroberfläche, mit der die verschiedenen implementierten Frontend Frameworks ausgewählt werden können. Je nach Auswahl wird das fertige Design angepasst und angezeigt. Die Abbildung 4.15 zeigt einen Screenshot der Auswahl der implementierten Frontend Frameworks.

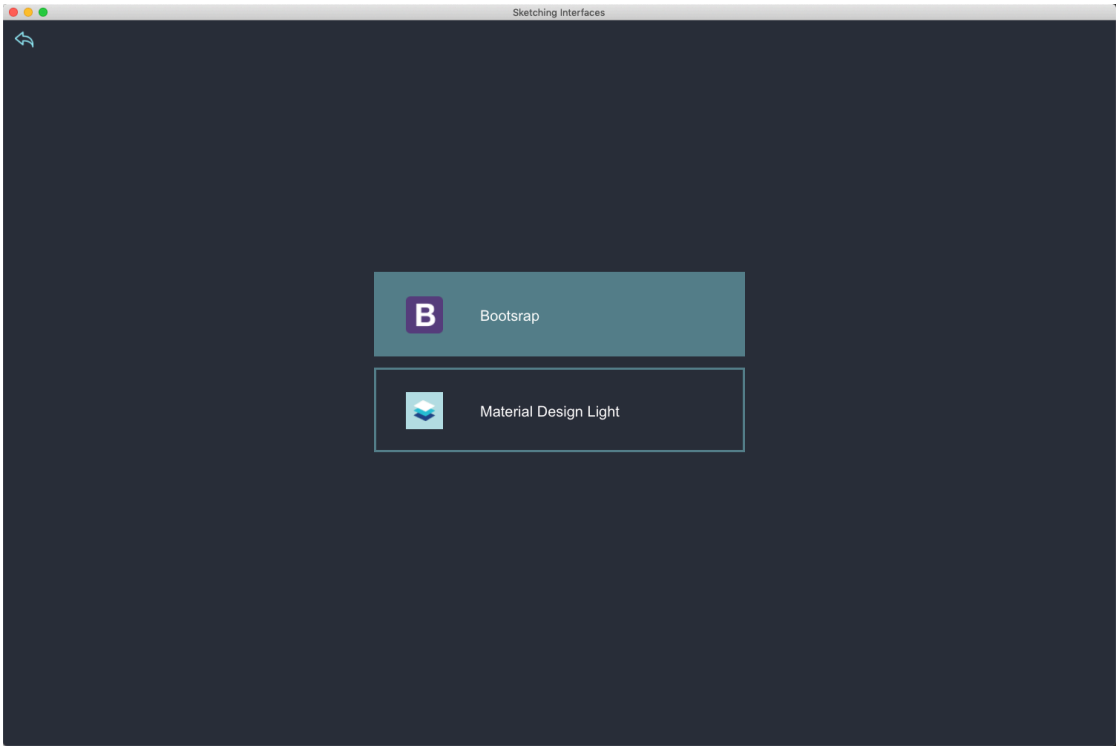


Abbildung 4.15: Screenshot des Tools für das Auswählen eines Frontend Frameworks

4.3 Kommunikation

In diesem Abschnitt wird auf die Kommunikation zwischen den einzelnen Komponenten näher eingegangen. Die Abbildung 4.16 zeigt die Kommunikationsschnittstellen in der Architektur des für diese Arbeit erstellten Tools. Der Doodle Classifier kommuniziert mittels OSC Protokoll mit dem Tool. Das Tool verwendet einen WebSocket, um die Änderungen im Frontend anzuzeigen und zu aktualisieren.

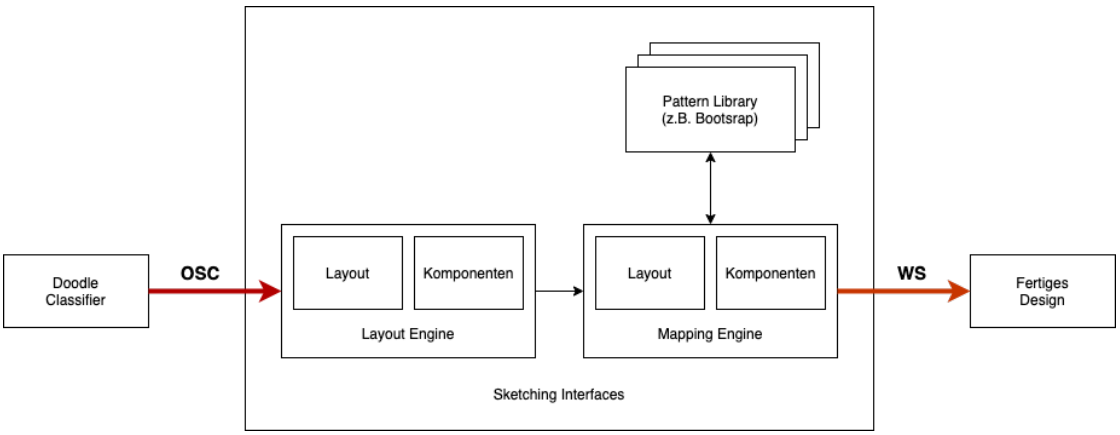


Abbildung 4.16: Kommunikation

OSC - Open Sound Control

Der DoodleClassifier verwendet das OSC-Protokill, um Daten weiterzuleiten. Der DoodleClassifier musste etwas angepasst werden, damit alle benötigten Daten weitergeleitet werden [42].

Die ursprüngliche OSC-Nachricht des Doodle Classifiers sah wie folgt aus:

Listing 4.11: Ursprüngliche OSC Message

```
[ 'button', 'picture', 'navbar']
```

Da ursprünglich nur die Labels der erkannten UI-Komponenten weitergeleitet wurden, musste der Doodle Classifier erweitert werden, um alle benötigten Informationen zu übermitteln. Zusätzlich zum Label werden nun *Area, X-Koordinate, Y-Koordinate, Breite und Höhe* jedes gefundenen Objektes übermittelt.

Der Doodle Classifier wurde in der Programmiersprache C geschrieben und verwendet OpenFrameworks. Um die OSC-Nachricht zu modifizieren, wurden die Codezeilen aus dem Listing 4.12 eingefügt.

```
1 m.addStringArg(foundSquares[i].label);
```

```
2 m.addFloatArg(foundSquares[i].area);
3 m.addIntArg(foundSquares[i].rect.x);
4 m.addIntArg(foundSquares[i].rect.y);
5 m.addIntArg(foundSquares[i].rect.width);
6 m.addIntArg(foundSquares[i].rect.height);
```

Listing 4.12: Modifikation des Doodle Classifiers

Durch die Änderungen am Source Code des DoodleClassifiers enthält die OSC-Message mehr Informationen. Das Beispiel 4.13 zeigt eine OSC-Message wie sie nach den Änderungen aussieht.

Listing 4.13: OSC-Nachricht nach Änderung des Doodle Classifiers

```
[ 'button', 5725, 301, 306, 120, 71,
  'picture', 7485, 433, 294, 115, 79,
  'navbar', 11506.5, 151, 81, 380, 39
]
```

Die Häufigkeit der Updates über das OSC-Protokoll hängt von der Verwendung des Doodle Classifiers ab. Es gibt zwei Möglichkeiten:

- Aufnahme der Skizze über die Schaltfläche **classify**: Dabei wird eine Update-Nachricht mit der aktuellen Aufnahme gesendet.
- Aufnahme der Skizze über die Schaltfläche **run**: Dabei wird periodisch eine Aufnahme der Skizze gemacht und diese über das OSC-Protokoll versendet. Es wird bei jeder Frame-Änderung ein Update gesendet.

Einstellungen für die DoodleClassifier Schnittstelle können in der *settings_doodleclassifier.xml* Datei vorgenommen werden:

Listing 4.14: settings_doodleclassifier.xml

```
<DoodleOSC>
        <ip>127.0.0.1</ip>
        <port>5000</port>
        <address>/classification</address>
    <classes>
        <class>button</class>
        <class>input</class>
        <class>picture</class>
        <class>info card</class>
        <class>carousel</class>
        <class>navbar</class>
        <class>table</class>
        <class>text</class>
    </classes>
</DoodleOSC>
```

Die IP-Adresse und der Port sowie die URL-Adresse können angepasst werden, in diesem Beispiel können die OSC Messages über die Adresse: *127.0.0.1:5000/classification* abgerufen werden.

Die Klassen für die Klassifizierung über den DoodleClassifier können ebenfalls in dieser Datei erweitert oder geändert werden. Im Abschnitt 4.5.1 wird beschrieben, wie neue UI-Komponenten hinzugefügt werden können.

WebSocket

Bei jeder Änderung, die über das OSC-Protokoll vom Doodle Classifier an das Tool gesendet wird, wird nach erfolgter Umwandlung in ein fertiges Design, dieses über den Web Socket an das Frontend gesendet und dort angezeigt. Es werden keine periodischen Updates an das Frontend gesendet, nur bei eventuellen Änderungen wird ein Update über den Web Socket gesendet.

4.4 Beschreibung der Funktionalität

Im folgenden Ab schnitt werden Screenshots des Prototypen, der im Zuge dieser Arbeit erstellt wurde, gezeigt.

Abbildung 4.17 zeigt das Hauptfenster des Tools, das auf den Input über den Doodle Classifier wartet. Über das Zahnrad oben rechts gelangt man zu den Einstellungen.

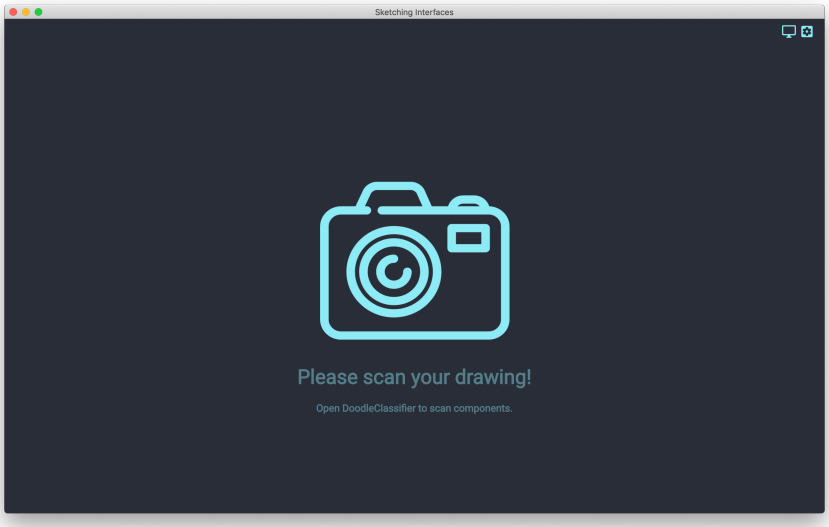


Abbildung 4.17: Screenshot – Hauptfenster

Im Tool kann zwischen zwei Ansichten für das fertige Design ausgewählt werden:

- Desktop Ansicht
- Mobile Ansicht

Mit dem Bildschirmbutton rechts oben im User Interface des Tools wird angezeigt, dass gerade die Desktopumwandlung aktiv ist. Durch Klicken auf diesen Bildschirm-Button kann auf die mobile Umwandlung umgeschaltet werden. Im Screenshot in der Abbildung 4.18 ist die Desktopumwandlung aktiv und in der Abbildung 4.19 ist die mobile Ansicht abgebildet.

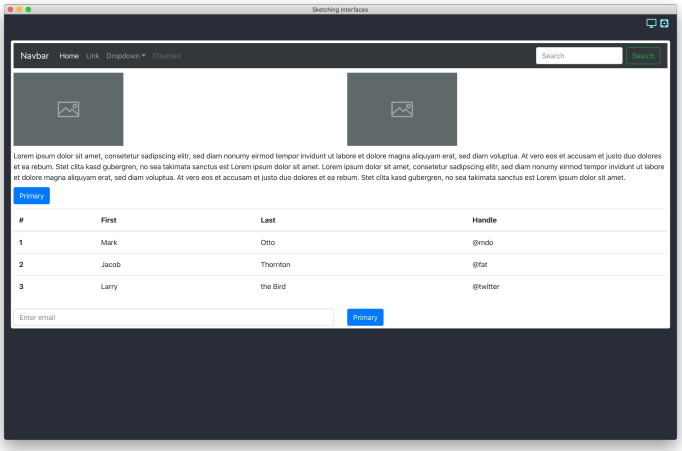


Abbildung 4.18: Screenshot – fertiges Design Desktop Ansicht

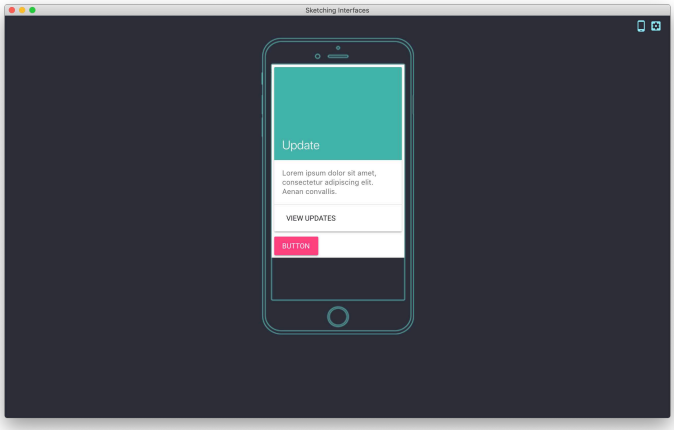


Abbildung 4.19: Screenshot – fertiges Design mobile Ansicht

Der Screenshot in der Abbildung 4.20 zeigt die Einstellungsmöglichkeiten:

- In der Einstellung **UI Patterns** können die verschiedenen UI Frameworks ausgewählt werden, siehe Abbildung 4.21.
- Die Einstellung **Layout Visibility** ermöglicht das Ein- und Ausschalten der Funktion für die Sichtbarkeit des Layouts.
- Die Schaltfläche **Layout Desktop/Mobile** ermöglicht das Umschalten des Layouts zwischen Desktop bzw. Mobiler Ansicht. Durch diese Einstellungsmöglichkeit können sowohl Desktop Applikationen als auch Mobile Apps skizziert und ein fertiges Design dazu erstellt werden.
- Mit dem Button **Relaunch DoodleClassifier** kann der DoodleClassifier neu gestartet werden, beim Start des Tools wird der DoodleClassifier aber automatisch mit gestartet. Sollte es erforderlich sein, kann der DoodleClassifier auch über diesen Button gestartet werden.

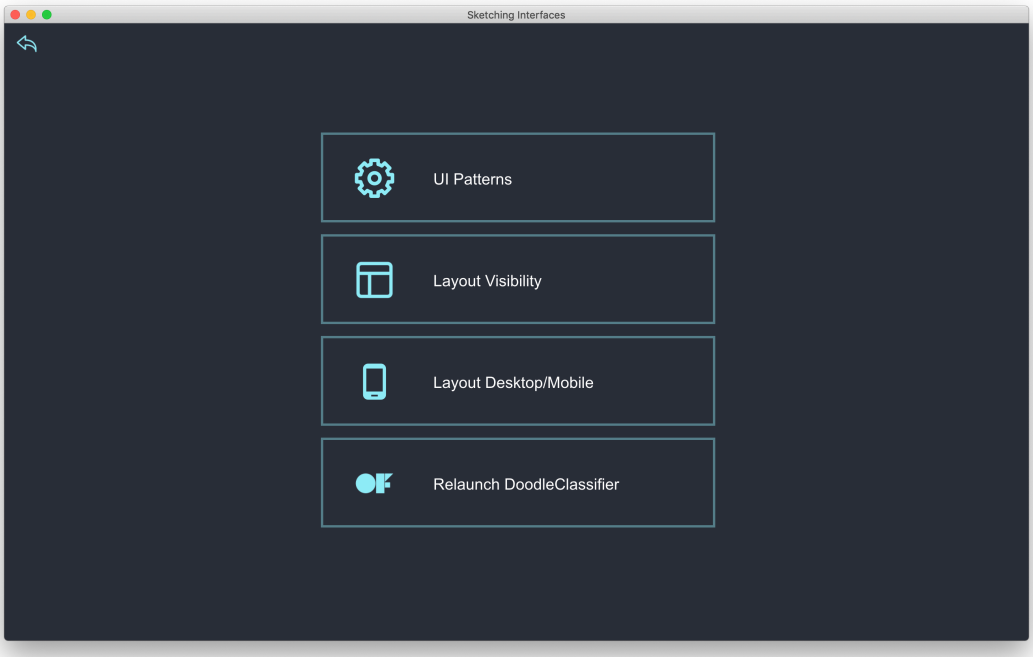


Abbildung 4.20: Screenshot – Einstellungen

Der Screenshot in der Abbildung 4.21 zeigt Auswahlmöglichkeiten von UI Frameworks, die verwendet werden können. Im Zuge dieser Arbeit wurden zwei Frameworks eingebunden, zwischen denen ausgewählt werden kann. Das Bootstrap und und das Material Design Lite Framework wurden eingebunden.

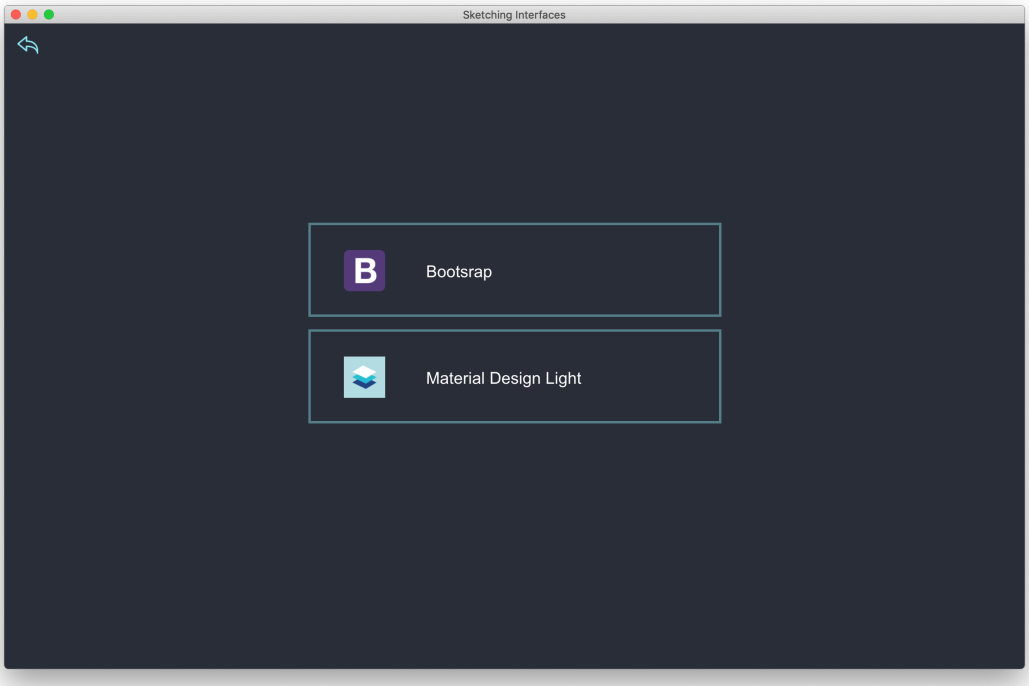


Abbildung 4.21: Screenshot – Framework Pattern Einstellung

Die Abbildung 2.4 zeigt ein fertiges Design mit sichtbarem Layout. Das Layout zu visualisieren, ist hilfreich, um den Aufbau des Layouts und die genaue Anordnung der Komponenten zu erkennen. Die Funktion, das Layout sichtbar zu machen, kann in den Einstellungen des Tools aktiviert oder deaktiviert werden.

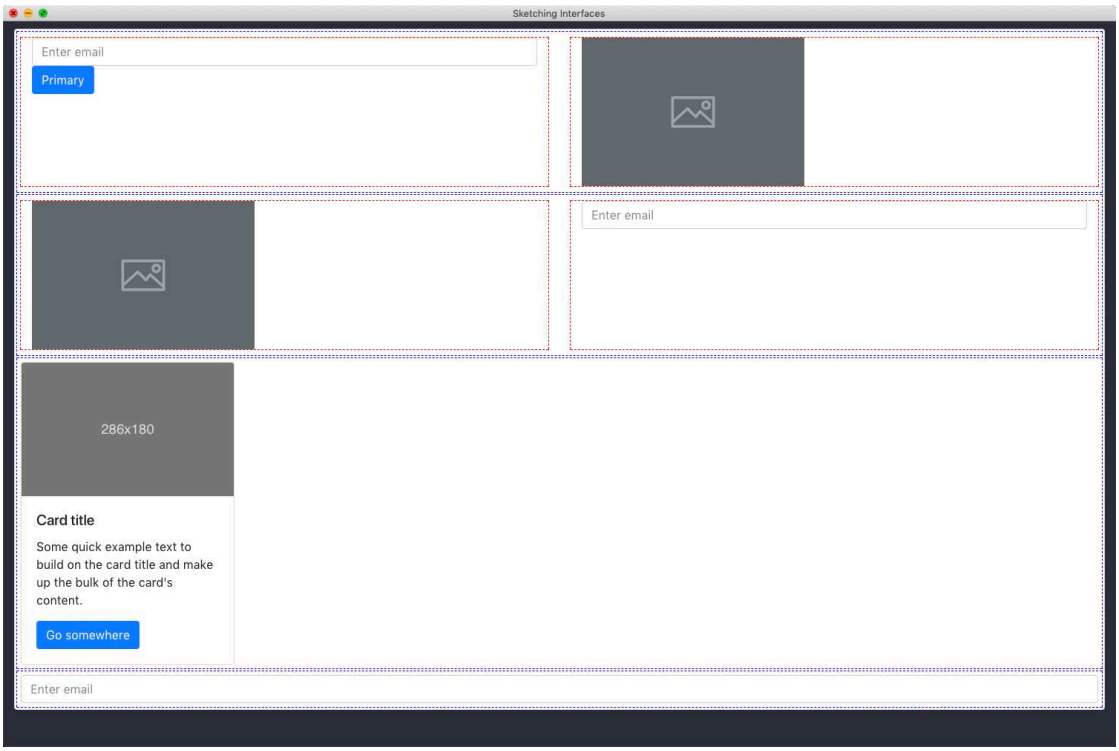


Abbildung 4.22: Screenshot – fertiges Design

Die Abbildung 4.23 zeigt ein Beispiel einer Handskizze, wie sie von dem/der BenutzerIn des Tools erstellt wird und über die Kamera in das Tool eingelesen werden kann. Diese Handskizze wurde verwendet, um das fertige Design im Screenshot in der Abbildung 4.22 zu erstellen. Beim Zeichnen wurde ein 1 mm dicker Stift verwendet, damit die Linien besser vom Doodle Classifier eingelesen werden können.

Abbildung 4.23: Beispiel einer Handskizze

Der DoodleClassifier, zu sehen in der Abbildung 4.24, ist eine wichtige Komponente dieses Tools und wird als eigenständige Applikation gestartet. Weitere Informationen zum Doodle Classifier folgen im Abschnitt 4.2.1.

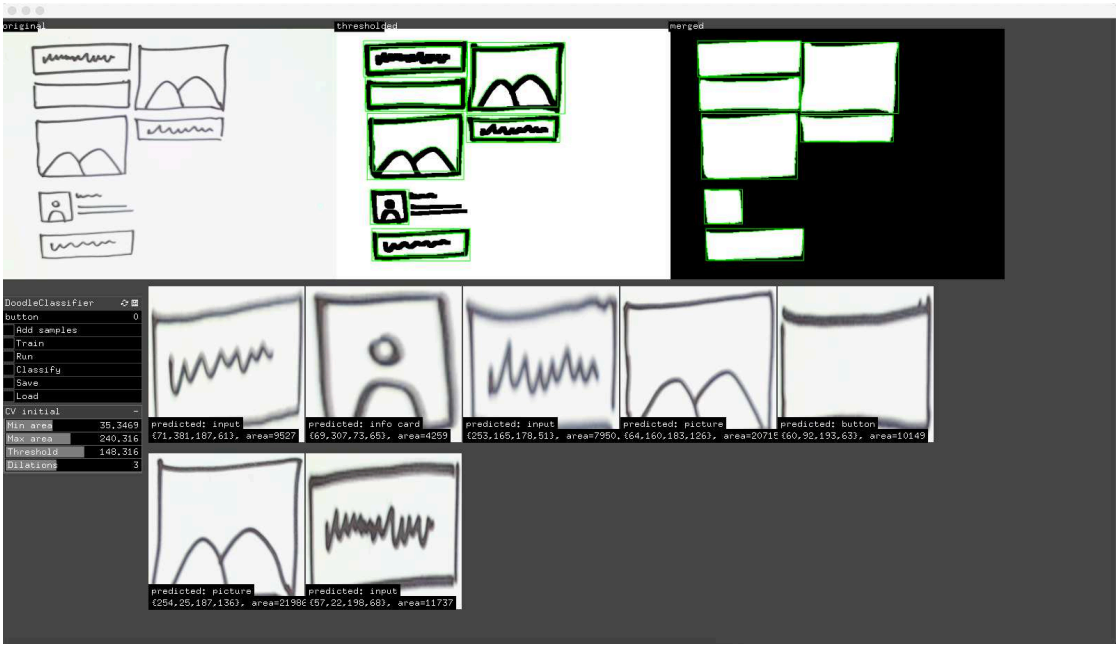


Abbildung 4.24: Screenshot – Doodle Classifier

4.5 Erweiterung des Tools um neue Komponenten

Im folgenden Abschnitt wird beschrieben, wie das Tool um neue UI-Komponenten und Frontend Frameworks erweitert werden kann.

4.5.1 Neue UI-Komponenten hinzufügen

Das Tool wurde so erstellt, dass es möglich ist, neue UI-Komponenten auf einfache Weise hinzuzufügen. Im folgenden Abschnitt wird kurz beschrieben, wie neue UI-Komponenten hinzugefügt werden können. Anhand des Beispiels einer *Tabelle* wird dies verdeutlicht.

1. Schritt: DoodleClassifier anpassen

Die Konfigurationsdatei des DoodleClassifiers *settings_doodleclassifier.xml* muss angepasst werden. Im Code Listing 4.15 ist die Datei im XML-Format zu sehen. Im Abschnitt *<classes>* muss eine neue *<class>* angelegt werden.

Wir fügen in diesem Beispiel im Abschnitt *<classes>* folgende Codezeile ein: *<class>table</class>*

Der Codeausschnitt 4.15 zeigt die bearbeitete *settings_doodleclassifier.xml* Datei. Die gelb hervorgehobene Codezeile wurde für dieses Beispiel hinzugefügt, um das Tool um die UI-Komponente *Tabelle* zu ergänzen.

Listing 4.15: Doodle Classifier Config neue Klasse hinzufügen

```
<DoodleOSC>
        <ip>127.0.0.1</ip>
        <port>5000</port>
        <address>/classification</address>
    <classes>
        <class>button</class>
        <class>input</class>
        <class>picture</class>
        <class>info card</class>
        <class>carousel</class>
        <class>navbar</class>
        <class>table</class>
        <class>text</class>
        <class>table</class>
    </classes>
</DoodleOSC>
```

2. Schritt: Trainingsdaten zeichnen und einspielen

Damit der DoodleClassifier die Zeichnungen einer Tabelle klassifizieren kann, muss er zuerst mit Skizzen einer Tabelle trainiert werden. Dazu zeichnen wir einige Skizzen einer

Tabelle. Abbildung 4.25 zeigt 4 verschiedene Skizzen einer Tabelle, die verwendet werden, um die Klasse Tabelle zu trainieren. Je mehr Tabellenskizzen beim Trainieren verwendet werden, desto genauer kann nachher klassifiziert werden.

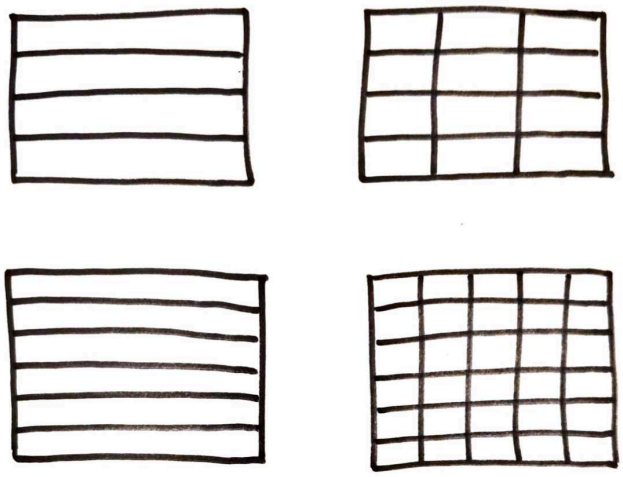


Abbildung 4.25: Tabellenskizzen

In Abschnitt 4.2.1 wird beschrieben, wie die Trainingsdaten im DoodleClassifier eingespielt werden müssen, sodass dieser die neuen UI-Komponenten erkennen kann.

3. Schritt: Mapping-Dateien anpassen

Im letzen Schritt müssen die Mapping-Dateien der Mapping Engine angepasst werden. Dies ist notwendig, damit die Mapping Engine eine HTML-Tabelle erstellen kann. Der Codeausschnitt 4.16 zeigt ein Beispiel einer Mapping-Datei, die um den Eintrag *table* erweitert werden muss.

Listing 4.16: Erweiterung einer Mapping-Datei um eine neue UI-Komponente

```
{
  "imports": {
    "header": [
      "../node_modules/bootstrap/dist/css/bootstrap.min.css"
    ],
    "script": [
      "../node_modules/jquery/jquery.min.js",
      "../node_modules/bootstrap/dist/js/bootstrap.min.js"
    ]
  },
  "start": "<div class=\"components\">",
  ...
  "table": "
```

```
                <table class="table">
                  <thead>
                    <tr>
                      <th scope="col">#</th>
                      <th scope="col">First</th>
                      <th scope="col">Last</th>
                      <th scope="col">Handle</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <th scope="row">1</th>
                      <td>Mark</td>
                      <td>Otto</td>
                      <td>@mdo</td>
                    </tr>
                    <tr>
                      <th scope="row">2</th>
                      <td>Jacob</td>
                      <td>Thornton</td>
                      <td>@fat</td>
                    </tr>
                    <tr>
                      <th scope="row">3</th>
                      <td>Larry</td>
                      <td>the Bird</td>
                      <td>@twitter</td>
                    </tr>
                  </tbody>
                </table>
  "
}
```

4.5.2 Neue Frontend Frameworks hinzufügen

Das Tool wurde so entwickelt, dass neue Frontend Frameworks einfach über eine Mapping-Datei hinzugefügt werden können. Dazu muss im Ordner Mappings des Tools eine JSON-Datei mit den Mappings, die im Abschnitt 4.2.3 beschrieben sind, angelegt werden. In der *settings.json* Datei muss der Dateiname der Mapping-Datei bei *pattern* angegeben werden, damit die richtige Mapping-Datei vom Tool ausgewählt wird.

Dadurch ist es auch möglich, nicht nur fertige Frontend Frameworks zu verwenden, sondern auch eigene UI-Komponenten mit designspezifischen Eigenschaften in das Tool einzubinden. Dies kann z. B. bei firmeninternen UI-Komponenten helfen, neue Funktionalitäten für die Software zu evaluieren. Ein Beispiel dafür liefert das Vorgehen von Airbnb wie in Abschnitt 2.4 beschrieben.

KAPITEL 5

Studie

In diesem Kapitel wird die Studie, die im Rahmen dieser Arbeit durchgeführt wurde, vorgestellt. Es wird zunächst auf die Methoden, die zur Erstellung dieser Studie verwendet wurden, eingegangen, anschließend wird die Umgebung und die verwendete Hardware beschrieben. Darauf folgt die Vorstellung aller ProbandInnen, die an der Studie teilnahmen, und es wird kurz erläutert, welche Bedeutung ihre jeweilige Tätigkeit für die Studie hat. Die Aufgabenstellung, die die ProbandInnen erhalten und durchführen mussten, wird vor der Analyse, auf die im nächsten Kapitel genauer eingegangen wird, beschrieben.

5.1 Methode der Studie

Um die Verwendbarkeit des Tools zu testen und um festzustellen, ob der Designprozess mit diesem Tool beschleunigt wird, wurde eine Studie mit 6 ProbandInnen durchgeführt. Die ProbandInnen erhielten eine Aufgabenstellung und mussten diese erledigen. Dafür wurde ihnen ein Infoblatt mit den vordefinierten User-Interface-Komponenten ausgehändigt, siehe Abbildung A.1. Nachdem die ProbandInnen die Aufgaben durchgeführt hatten, erhielten sie einen Fragebogen zum Ausfüllen. Die Auswertung der Ergebnisse dieser Fragebögen wird im Kapitel 6 vorgestellt [46].

5.2 Verwendete Hardware und Umgebungsaufbau

Für die Durchführung der Studie wurde eine 1080P Full HD Webcam verwendet. Diese wurde über USB mit einem Computer verbunden und zeichnet die Handskizzen auf, um sie in den Doodle Classifier einzulesen. Die Kamera wurde auf einem Stativ montiert und auf ein Whiteboard gerichtet. Der Abstand der Kamera zum Whiteboard beträgt ca. 100 cm. Dadurch wurde es den ProbandInnen ermöglicht, ungestört mit einem Stift auf

einem Whiteboard zu zeichnen, und die Kamera konnte die Zeichnungen direkt erfassen. Ein MacBook Pro (Retina, 13-inch, Early 2015) wurde den ProbandInnen zur Verfügung gestellt und verwendet, um das Tool auszuführen.

Auf dem Whiteboard wurde der Rahmen eingezeichnet, innerhalb dessen die Kamera das Bild erfassen konnte. Dadurch wurde verhindert, dass die ProbandInnen über den Bereich hinaus zeichnen, den die Kamera nicht mehr aufzeichnen konnte. Die ProbandInnen haben mit einem schwarzen Filzstift mit einer Strichstärke von 1 mm gezeichnet, da dieser besser über die Kamera erfasst werden konnte. Je nachdem, wie die Lichtverhältnisse sind, könnte auch ein dünnerer Stift verwendet werden, jedoch sind Zeichnungen damit über die Kamera schwerer zu erfassen. Aus diesem Grund wurde den ProbandInnen der oben beschriebene Stift zur Verfügung gestellt.

Abbildung 5.1 zeigt eine Skizze des Aufbaus der Studienumgebung. Das Whiteboard, auf dem die ProbandInnen zeichnen konnten, ist rechts im Bild zu sehen. Die Kamera wurde wie in der Abbildung auf einem Stativ auf dem Whiteboard montiert wird und mit USB am Laptop angeschlossen, wo die Ergebnisse der Zeichnungen angezeigt wurden.

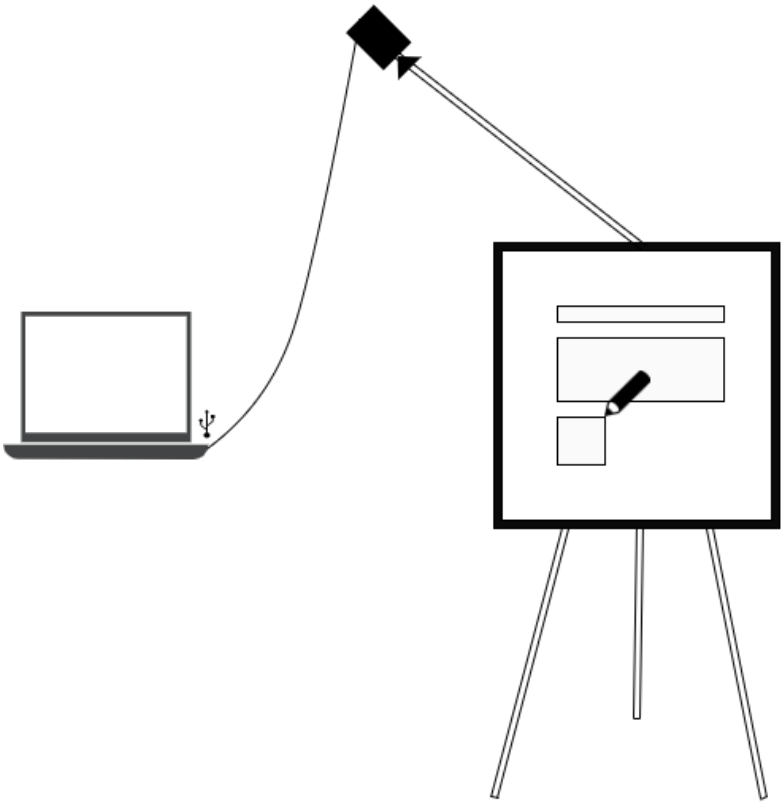


Abbildung 5.1: Aufbau der Studienumgebung

Der Doodle Classifier musste an die Lichtverhältnisse und den Abstand zur Skizze angepasst werden. Dazu wurden die Parameter, wie im Kapitel 4.2.1 beschrieben, verändert. Dies ist notwendig, um die Linien und Komponenten auf dem Blatt optimal zu erfassen, da bei einer sich verändernden Lichteinstrahlung die Kamera die Bilder verschieden aufnehmen würde. Die Parameter Min-Area und Max-Area mussten auf den Abstand der Kamera zum Whiteboard angepasst werden: Die Min-Area musste so angepasst werden, dass die kleinsten gezeichneten Komponenten erfasst werden konnten, die Max-Area wurde auf die größten Komponenten angepasst. Die Max-Area durfte nicht zu groß gewählt werden, da sonst das gesamte Whiteboard als Komponente erkannt wurde. Der Threshold und die Dilation musste so angepasst werden, dass die gezeichneten Linien optimal erfasst werden konnten. Diese Parameter sind abhängig von der Lichteinstrahlung und der Dicke der gezeichneten Linien. Durch diese Kalibrierungen mittels Doodle Classifier konnten die Handskizzen optimal aufgenommen und klassifiziert werden.

Die Fotos in der Abbildung 5.2 zeigen den Aufbau der Studienumgebung und wie zwei Probanden die Aufgabenstellung ausführen.

Abbildung 5.2: Fotos Studienumgebung

5.3 Aufgabenstellungen

Alle ProbandInnen erhielten eine Aufgabenstellung, die sie durchführen mussten, und daraufhin 8 Fragen dazu beantworten mussten.

Bei den ersten zwei Aufgaben mussten die ProbandInnen eine Webseite nachzeichnen. Die dritte Aufgabe beinhaltet das Nachzeichnen von zwei Bildschirmen einer App. Die vierte und letzte Aufgabe ist eine offene Aufgabenstellung, die in einer Gruppe aus mindestens zwei Personen durchgeführt werden musste.

Folgende Aufgaben mussten von den ProbandInnen ausgeführt werden:

1. Skizzieren Sie die folgende Webseite nach.

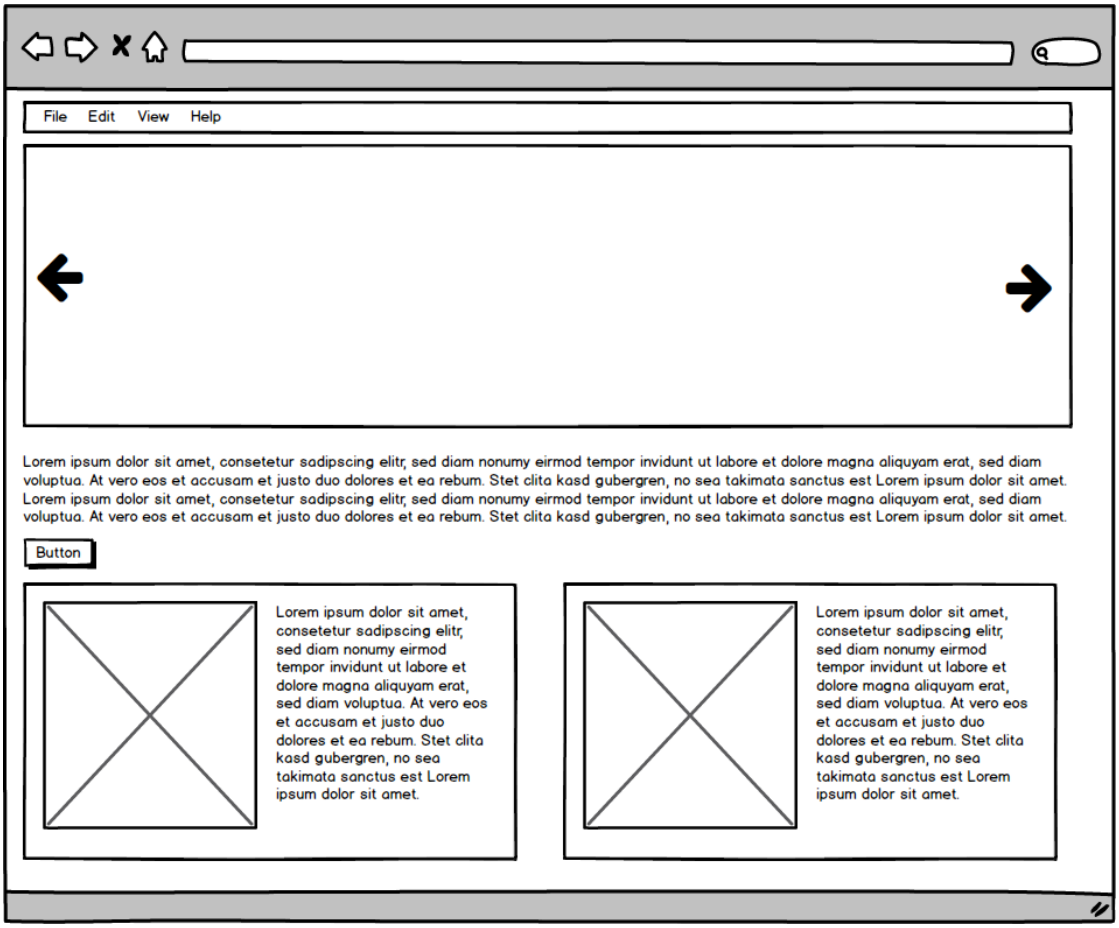


Abbildung 5.3: Screenshot – Webseite zum Nachbauen

2. Skizzieren Sie die folgende Webseite nach.

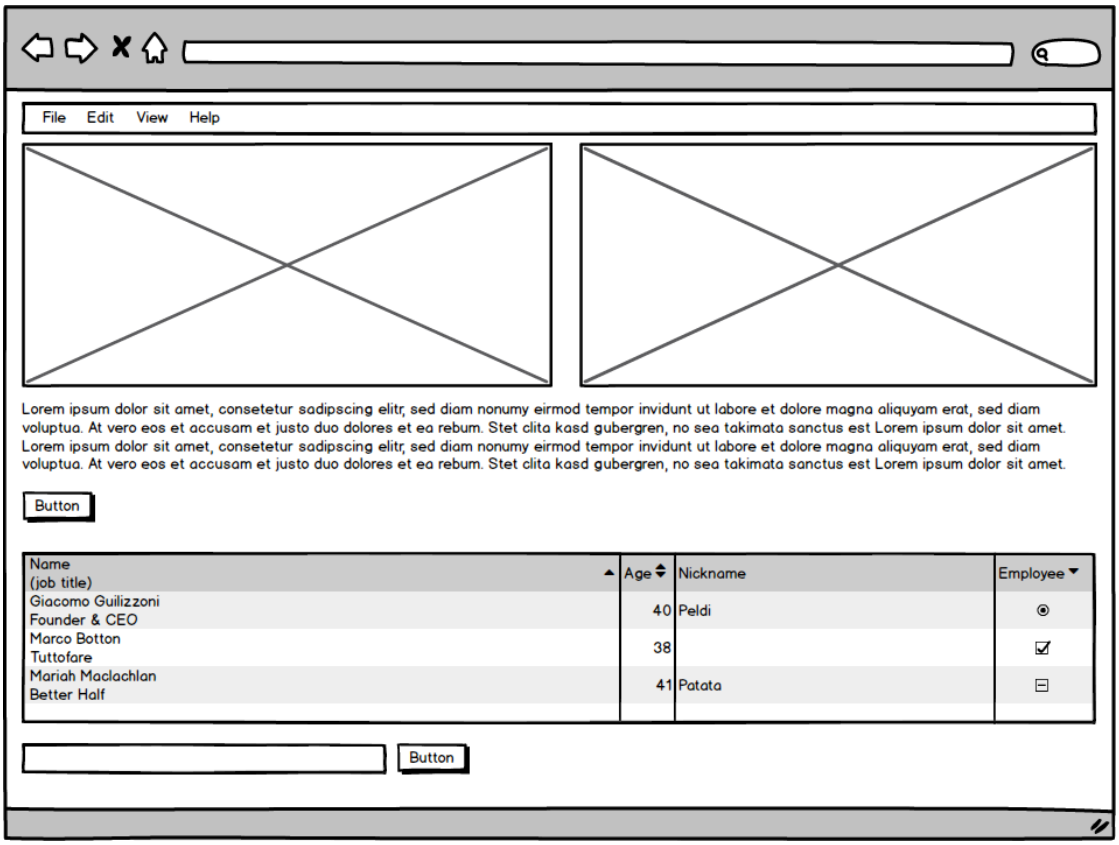


Abbildung 5.4: Screenshot – Webseite zum Nachbauen

3. Skizzieren Sie die beiden folgenden Bildschirme einer App nach.

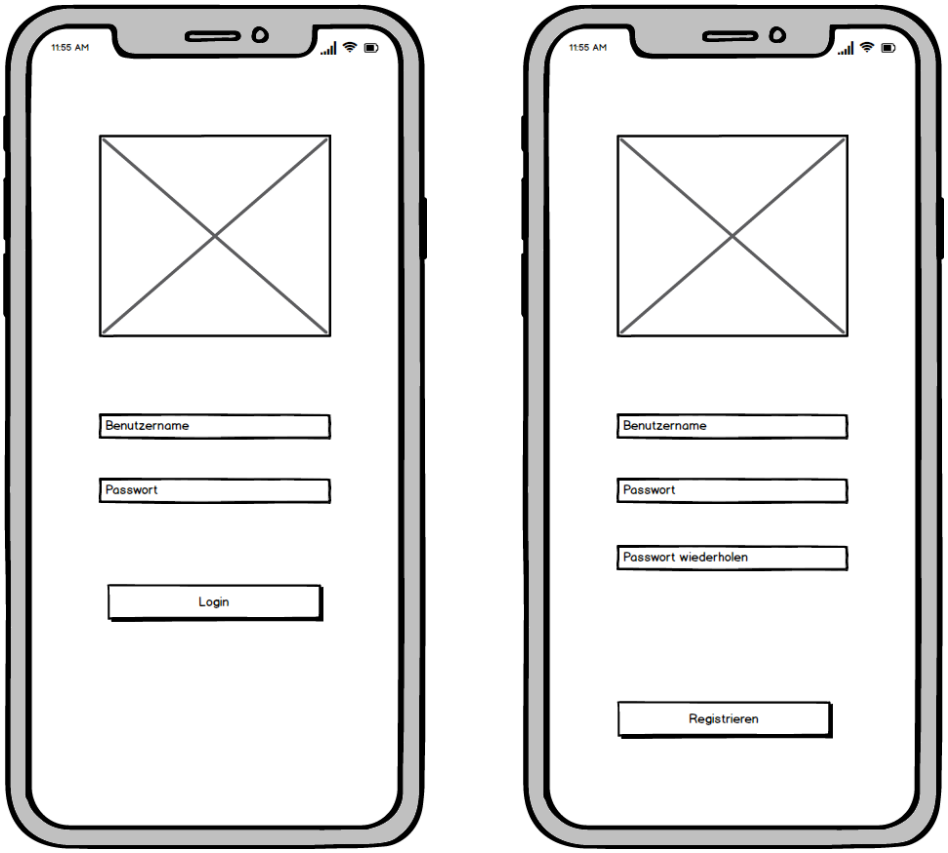


Abbildung 5.5: Screenshot – App zum Nachbauen

4. **Gruppenaufgabe:** Erstellen Sie eine App zum Festhalten von Arbeitszeiten.

   Die App benötigt folgende Funktionen:

   - Die Mitarbeiter müssen sich einloggen können (mindestens Benutzername und Passwort)
   - Die Arbeitszeiten müssen einsehbar sein
   - Die Zeit muss mit einem Start- und Stopp-Button erfasst werden können

Infoblatt

Zusätzlich zu den Aufgabenstellungen erhielten die ProbandInnen ein Infoblatt als Beispiel und Hilfe. Das Infoblatt enthält alle zur Verfügung stehenden UI-Komponenten sowie Informationen darüber, wie diese gezeichnet werden sollten. Abbildung A.1 zeigt das Infoblatt.

Fragebogen

Am Ende der Studie erhielten alle ProbandInnen einen Fragebogen, den Sie ausfüllen solleten. Die Antworten werden entweder als Bewertung von 1 bis 10, wobei 1 die schlechteste Note und 10 die beste Note ist, oder als freier Text abgegeben [46]. Folgende Fragen wurden den ProbandInnen gestellt:

1. Wie verwendbar ist das Tool?
   Antwortmöglichkeit: 1 bis 10
2. Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?
   Antwortmöglichkeit: 1 bis 10
3. Entspricht das Design des Tools den Erwartungen aus der Zeichnung?
   Antwortmöglichkeit: 1 bis 10
4. Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?
   Antwortmöglichkeit: 1 bis 10
5. Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?
   Antwortmöglichkeit: Text
6. Was funktioniert nicht wie erwartet?
   Antwortmöglichkeit: Text
7. Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?
   Antwortmöglichkeit: Text
8. Gibt es Verbesserungsvorschläge?
   Antwortmöglichkeit: Text

5.4 ProbandInnen

Für die Studie wurden ProbandInnen mit verschiedenen Vorkenntnissen herangezogen, um das unterschiedliche Vorgehen zu beobachten. Informatiker, Grafiker und Personen ohne Vorkenntnisse in den Bereichen Oberflächendesign und Programmierung mussten die Aufgaben ausführen. Die ProbandInnen mussten das Alter, das Geschlecht und den Beruf angeben. Sonstige personenbezogene Daten wurden nicht erfasst, lediglich die Initialen.

Informatiker: Zwei Probanden waren Programmierer und haben deshalb bereits Vorkenntnisse im Bereich User Interface Design. Dies war beim Zeichnen der User Interfaces von Vorteil sein.

Grafiker: Zwei Probanden waren Grafiker und haben Erfahrung im Zeichnen von User Interfaces, allerdings arbeiten sie beim Erstellen von User Interfaces nie mit Stift und Papier, sondern mit Applikationen wie Adobe InDesign oder Adobe Photoshop.

Personen ohne Vorkenntnisse: Zwei Probandinnen hatten keine Vorkenntnisse im Bereich User Interface Design. Interessant hierbei ist, zu beobachten, wie diese Personen die User Interfaces zeichnen.

5.4.1 Vorstellung der ProbandInnen

Proband MV: Der Proband MV war 29 Jahre alt und männlich. Beruflich arbeitete er als Backend Developer und hatte daher Kenntnisse in der Informatik und dem Designen von User Interfaces.

Zur Durchführung der Aufgabe brauchte er ca. 30 Minuten.

Proband PR: Die Probandin PR war 30 Jahre alt und weiblich. Sie arbeitete freiberuflich als Übersetzerin. Vorkenntnisse im Bereich User Interface Design waren keine vorhanden.

Zur Durchführung der Aufgabe brauchte sie ca. 30 Minuten.

Proband MM: Der Proband MM war 28 Jahre alt und männlich. Beruflich arbeitete er als Softwareentwickler. Der Proband hatte bereits Erfahrung beim Entwerfen von User Interfaces.

Für die Durchführung der Aufgaben brauchte der Proband ca. 25 Minuten.

Proband SU: Die Probandin SU war 26 Jahre alt und weiblich. Ihr beruflicher Werdegang als Beraterin für Arbeitssicherheit hatte nichts mit Informatik zu tun, daher besaß sie keinerlei Vorkenntnisse im Bereich User Interface Design.

Für die Durchführung der Aufgaben brauchte sie ca. 25 Minuten.

Proband MM2: Der Proband MM2 war 32 Jahre alt und männlich. Beruflich arbeitete er als Grafiker und hatte somit etwas Erfahrung im Bereich User Interface Design.

Für die Durchführung der Aufgaben brauchte er ca. 35 Minuten.

Proband DO: Der Proband DO war 32 Jahre alt und männlich. Beruflich arbeitete er als Mediendesigner und hatte Erfahrungen im Bereich User Interface Design.

Für die Durchführung der Aufgaben brauchte er ca. 35 Minuten.

KAPITEL 6

Ergebnisse der Studie

Im folgenden Kapitel werden die Ergebnisse der Studie vorgestellt. Es werden die Antworten der Fragebögen sowie die Zeichnungen der ProbandInnen aufgeführt bzw. abgebildet und ausgewertet. Zudem werden eigene Beobachtungen während der Durchführung der Studie beschrieben und es wird auf das Feedback der ProbandInnen eingegangen. Am Ende wird ein Fazit zur Studie und zu deren Ergebnissen gezogen.

6.1 Auswertung

Im folgenden Abschnitt werden die einzelnen Aufgaben und die Antworten der Fragebögen der ProbandInnen ausgewertet. Im ersten Teil des Abschnitts werden die Aufgabenlösungen der ProbandInnen vorgestellt und auf die Probleme, die bei der Umwandlung der Skizzen in das fertige Design aufgetreten sind, eingegangen. Im zweiten Teil des Abschnitts werden die Antworten der Fragebögen der ProbandInnen zitiert.

6.1.1 Auswertung der Aufgaben

Die einzelnen Aufgaben, die die ProbandInnen durchgeführt haben, werden in diesem Abschnitt vorgestellt. Dazu werden jeweils zwei Abbildungen gezeigt: links die Skizze der ProbandInnen, wie sie sie auf dem Whiteboard gezeichnet haben, und recht das fertige Design, das sich aus den Skizzen ergeben hat. Sollten Komponenten nicht richtig erkannt werden oder Layout Probleme aufgetreten sein, wird auf diese kurz eingegangen.

Aufgabe 1

Aufgabenstellung: Skizzieren Sie die Webseite in Abbildung 5.3 nach.

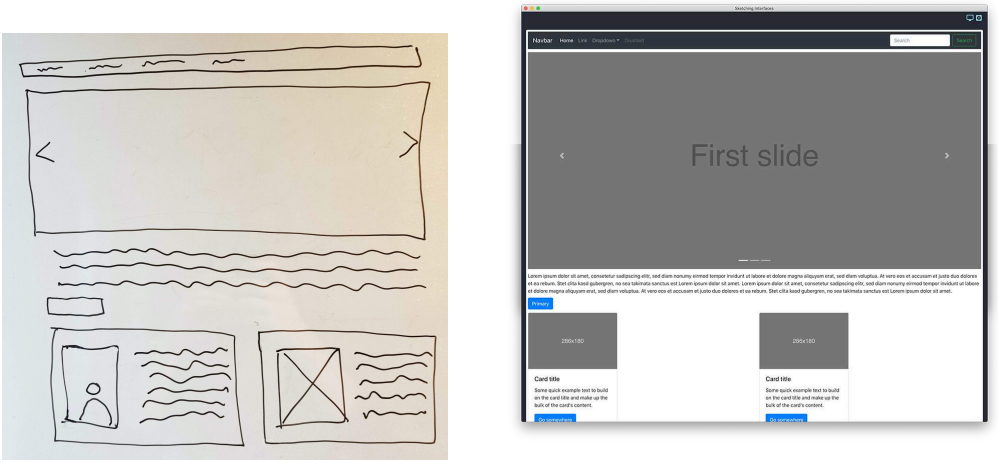


Abbildung 6.1: Ergebnisse Aufgabe 1 – Proband MV

Probleme bei der Umwandlung der Skizzen ergaben sich bei Proband PR: Die *Text* Skizze ist etwas zu kurz gezeichnet und konnte nicht als Komponente vom Doodle Classifier erkannt werden. Dieses Problem könnte gelöst werden, indem mann den Wert des *Min-area* Parameter des Doodle Classifiers verringert. Außerdem konnte das Tool die beiden *Infoboxen* nicht nebeneinander anordnen. Dieses Problem ist darauf zurückzuführen, dass der Rand der Infobox im rechten oberen Eck zu weit nach oben gezogen wurde und somit der Wert *y-position-tolerance = 10* der Layout Engine überschritten wird. Durch das Erhöhen des Wertes kann auch dieses Problem gelöst werden.

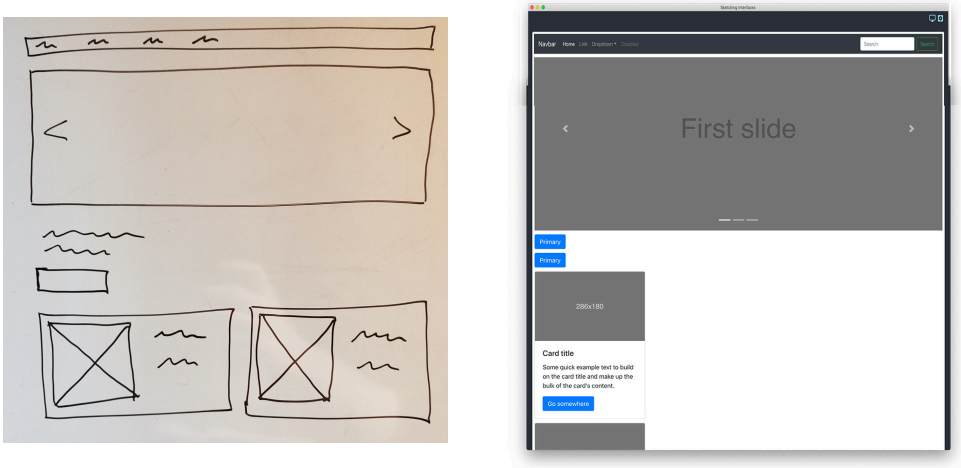


Abbildung 6.2: Ergebnisse Aufgabe 1 – Proband PR

Der *Text* der Skizze 6.3 wurde zu breit gezeichnet. Da beim Trainieren immer nur kürzere Vergleichsskizzen für die *Text* Komponente gezeichnet wurden, konnte diese nicht richtig erkannt werden.

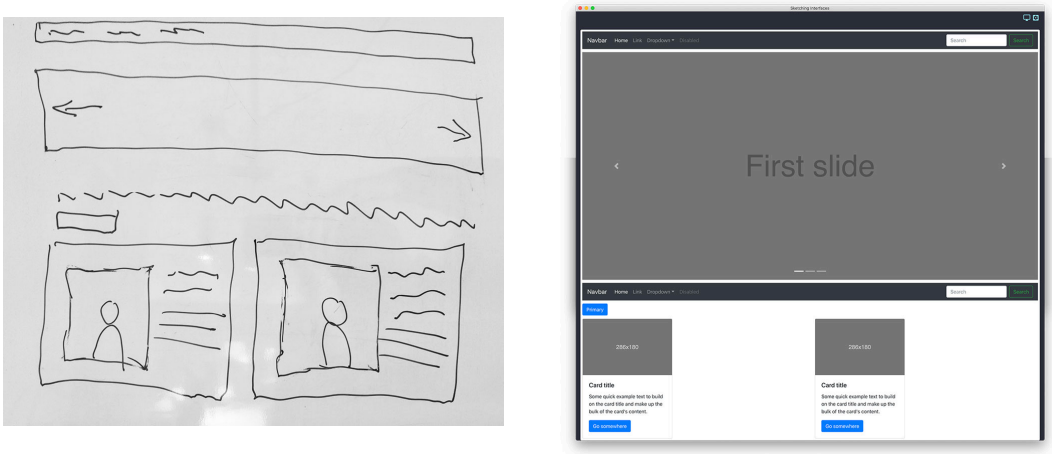


Abbildung 6.3: Ergebnisse Aufgabe 1 – Proband MM

Der Proband SU hat für den Text mehrere Linien gezeichnet, daher besteht der Text im fertigen Design aus mehreren Absätzen. Die beiden *Infoboxen* sind verschieden groß gezeichnet worden, daher konnten sie nicht nebeneinander erkannt werden und wurden im fertigen Design untereinander angeordnet.

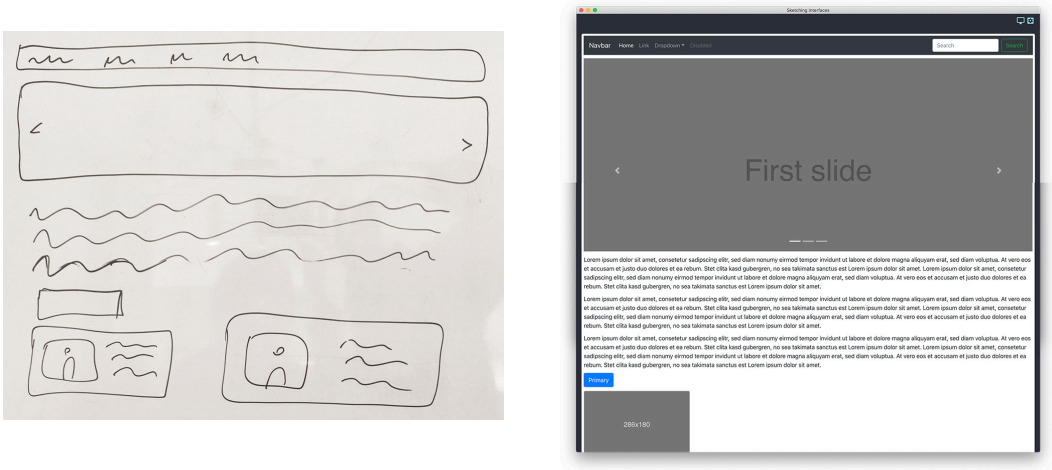


Abbildung 6.4: Ergebnisse Aufgabe 1 – Proband SU

Die Navigationsleiste aus der Skizze 6.5 wurde fälschlicherweise als Text erkannt.

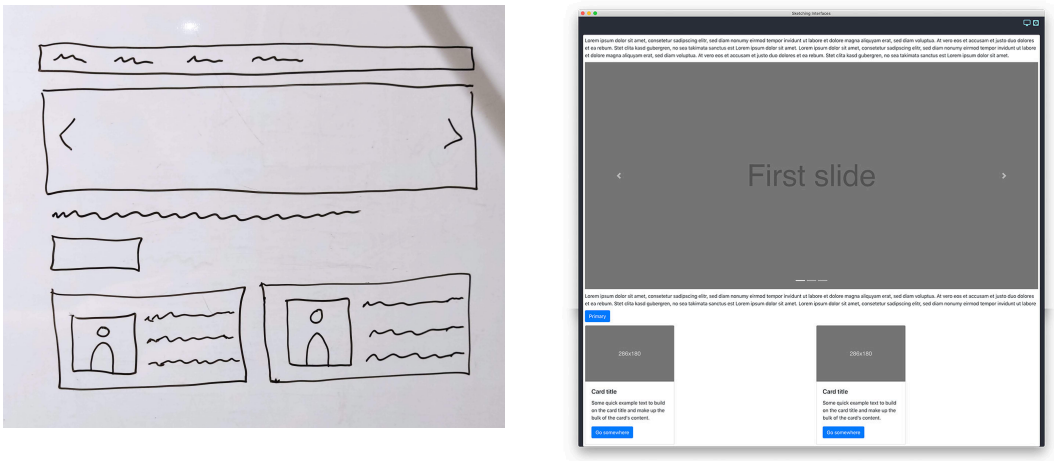


Abbildung 6.5: Ergebnisse Aufgabe 1 – Proband MM2

Bei Proband DO wurde wie in Abbildung 6.6 dargestellt vom Tool alles richtig erkannt und umgewandelt.

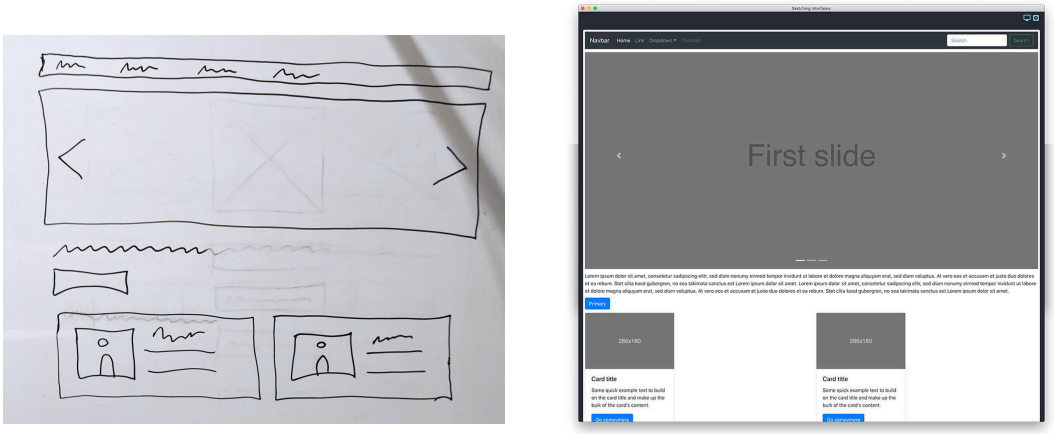


Abbildung 6.6: Ergebnisse Aufgabe 1 – Proband DO

Aufgabe 2

Aufgabenstellung: Skizzieren Sie die Webseite in Abbildung 5.4 nach.

Alle Skizzen dieser Aufgabe wurden gut erkannt und die richtigen Ergebnisse wurden als fertiges Design geliefert. Die *Textboxen* der Probanden MV und MM wurden fälschlicherweise als *Navigationsleisten* erkannt. Dieses Problem trat häufiger auf, was darauf zurückzuführen ist, dass diese beiden Komponenten ziemlich ähnlich sind. Die Abbildungen 6.7 und 6.9 zeigen die fehlerhaften *Navigationsleisten* anstelle der *Textbox*.

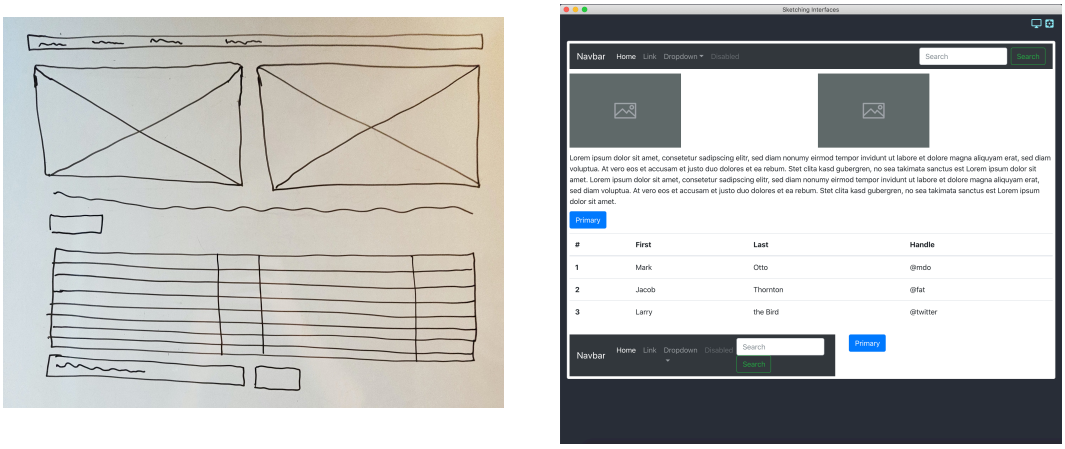


Abbildung 6.7: Ergebnisse Aufgabe 2 – Proband MV

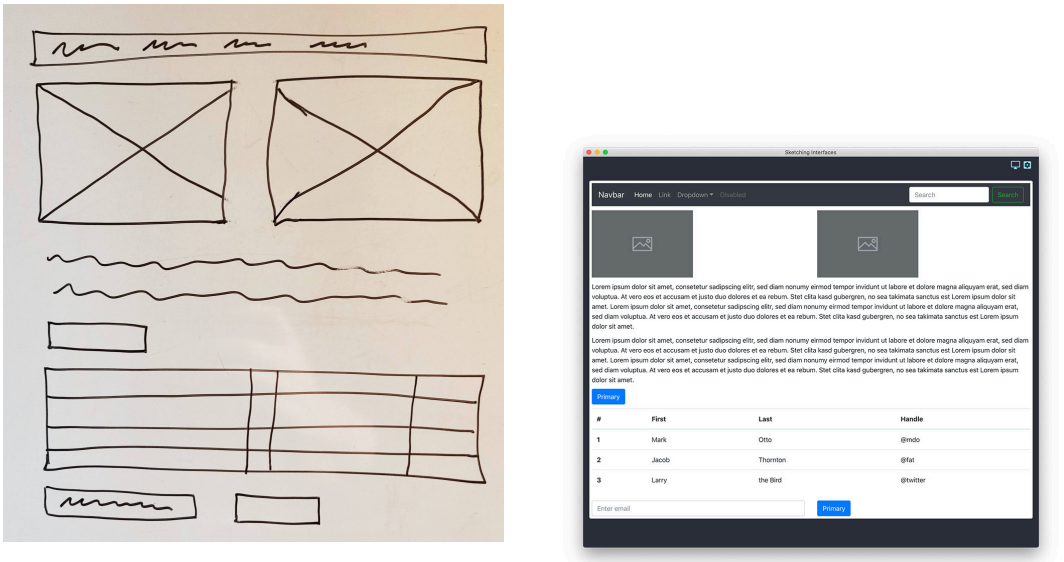


Abbildung 6.8: Ergebnisse Aufgabe 2 – Proband PR

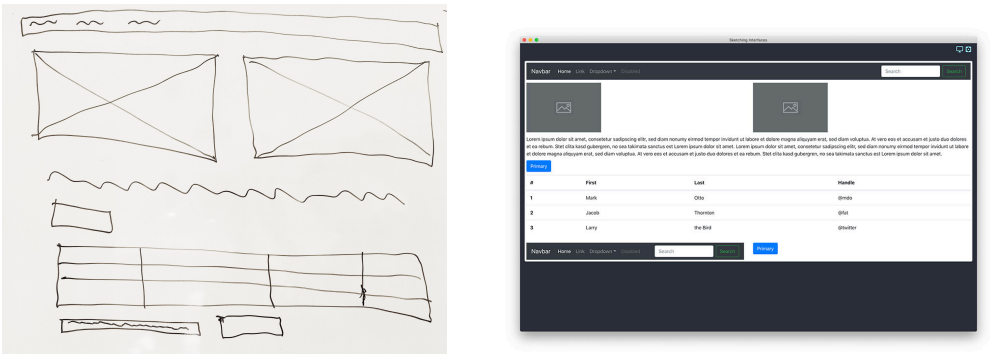


Abbildung 6.9: Ergebnisse Aufgabe 2 – Proband MM

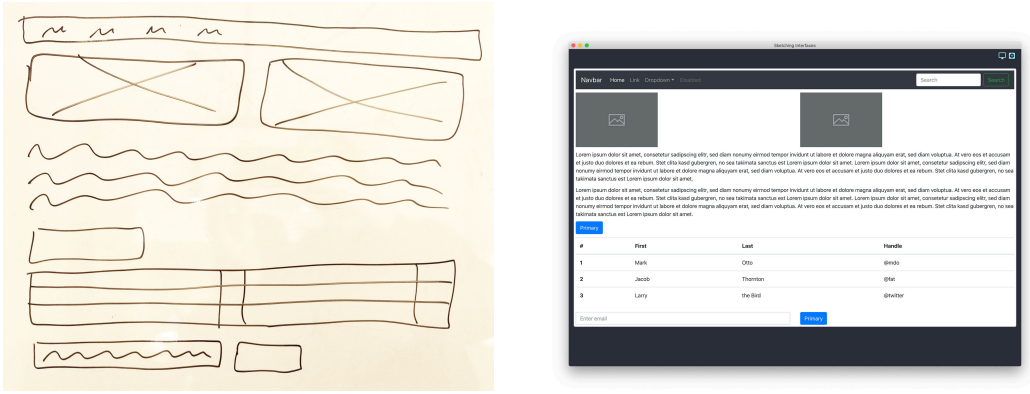


Abbildung 6.10: Ergebnisse Aufgabe 2 – Proband SU

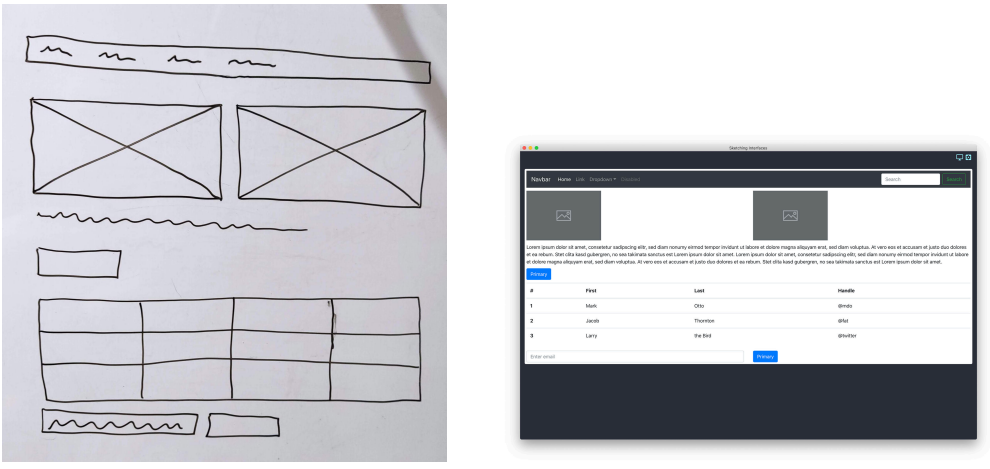


Abbildung 6.11: Ergebnisse Aufgabe 2 – Proband MM2

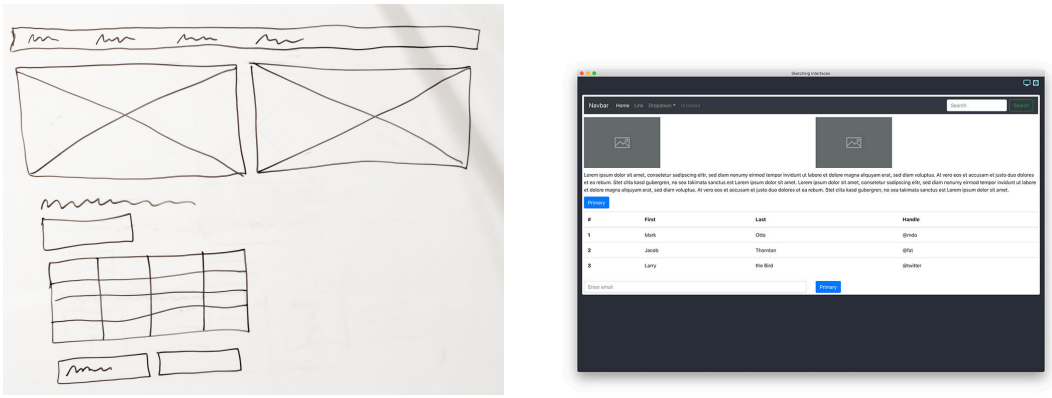


Abbildung 6.12: Ergebnisse Aufgabe 2 – Proband DO

Aufgabe 3

Aufgabenstellung: Skizzieren Sie die beiden Bildschirme einer App wie in Abbildung 5.5 nach.

Die Skizzen aus der Aufgabe 3 haben alle zu richtigen Ergebnissen geführt. Auch hier kam es lediglich bei zwei ProbandInnen zu Problemen, nämlich, dass die Komponenten *Input* mit der *Navigationsleiste* verwechselt wurden. Dieses Problem trat bei der Probandin SU und den Probanden DO auf. Die Abbildungen 6.16 und 6.18 zeigen dieses Problem.

Die Probanden MM und MM2 haben für diese Aufgabe das Material Lite Framework verwendet, alle anderen das Bootstrap Framework.

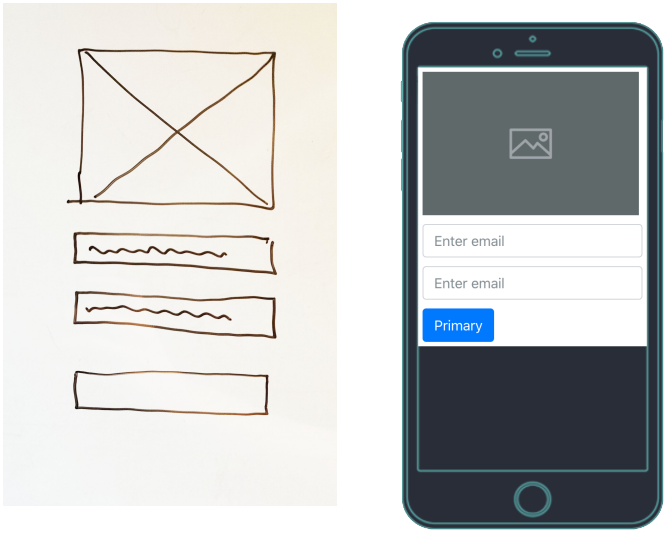


Abbildung 6.13: Ergebnisse Aufgabe 3 – Proband MV

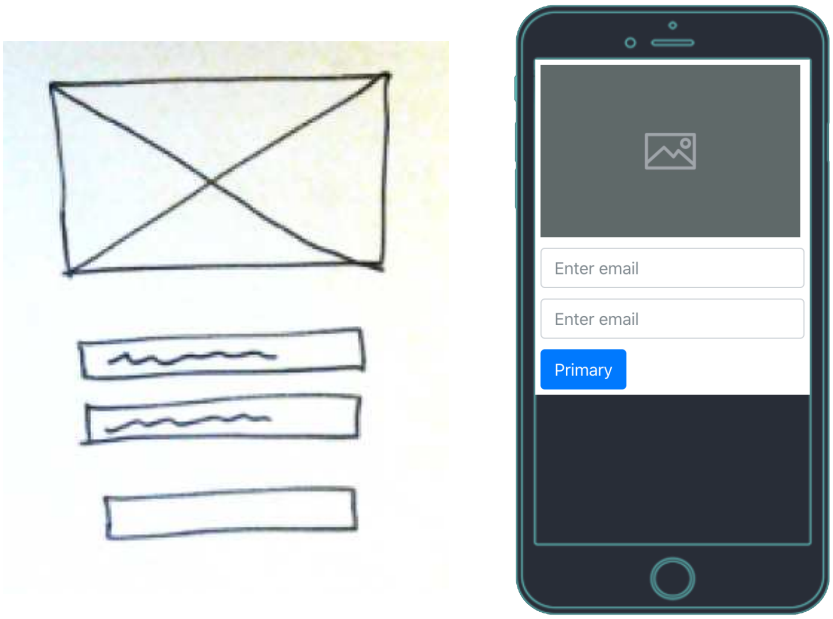


Abbildung 6.14: Ergebnisse Aufgabe 3 – Proband PR

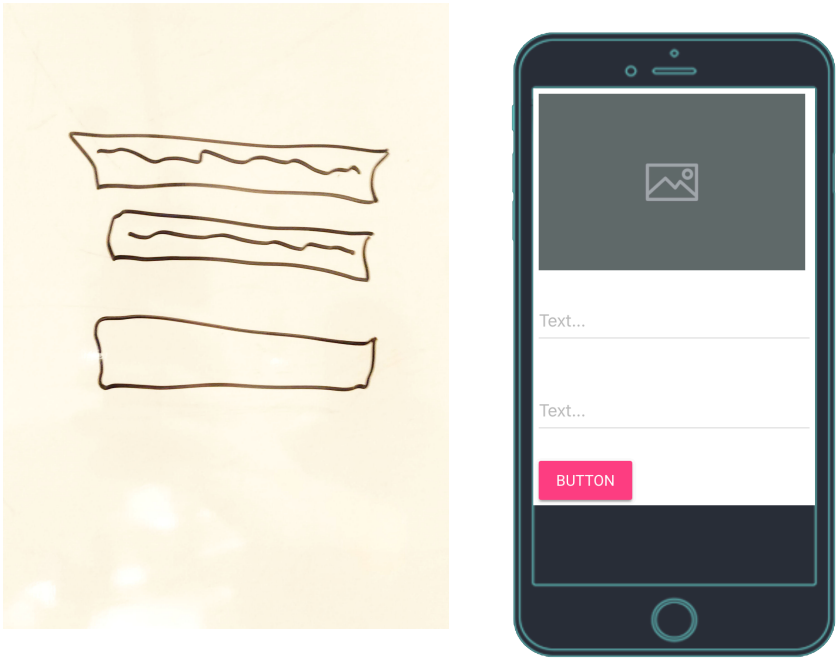


Abbildung 6.15: Ergebnisse Aufgabe 3 – Proband MM

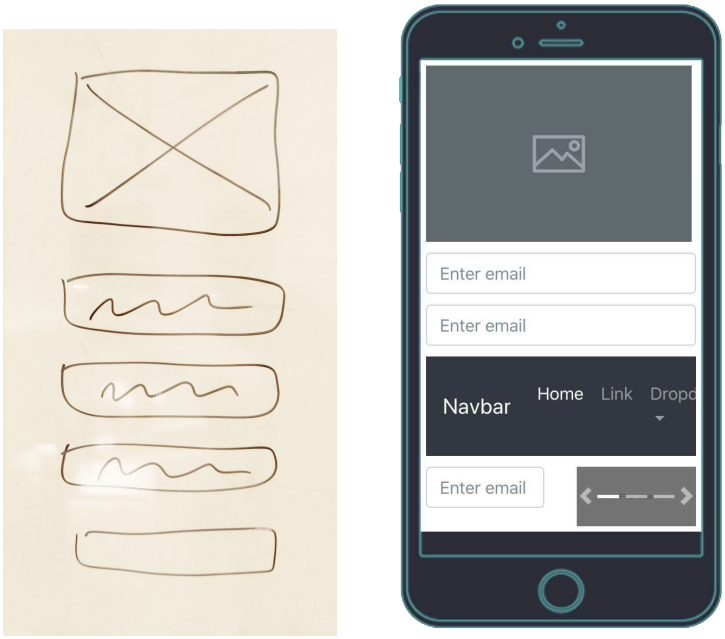


Abbildung 6.16: Ergebnisse Aufgabe 3 – Proband SU

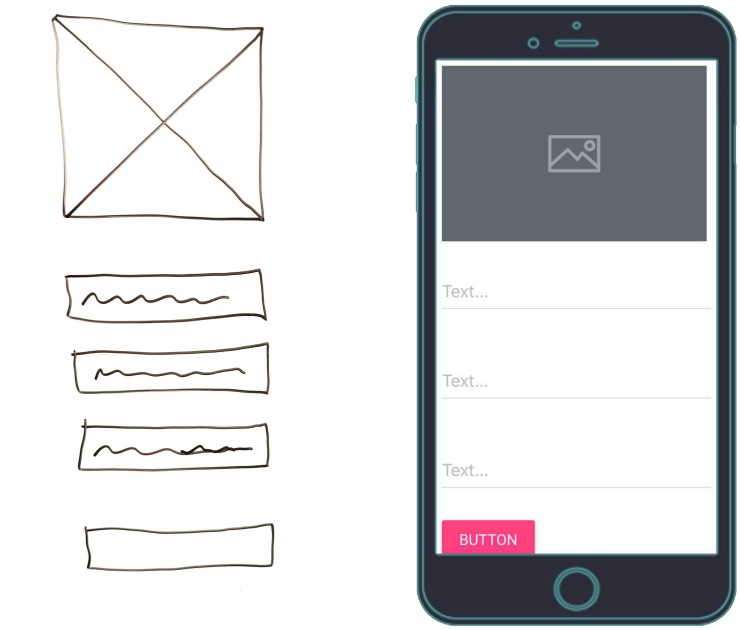


Abbildung 6.17: Ergebnisse Aufgabe 3 – Proband MM2

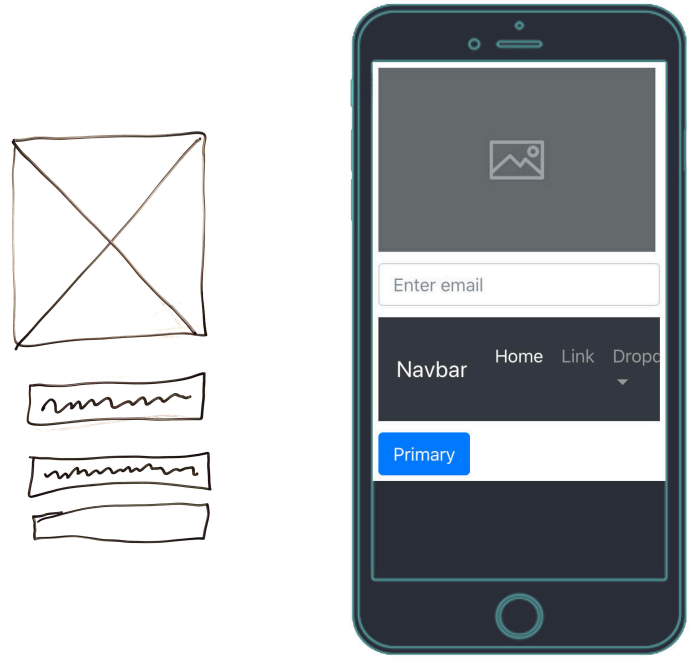


Abbildung 6.18: Ergebnisse Aufgabe 3 – Proband DO

Aufgabe 4 - Gruppenaufgabe

Aufgabenstellung:

Erstellen Sie eine App zum Festhalten von Arbeitszeiten. Die App benötigt folgende Funktionen:

- Die Mitarbeiter müssen sich einloggen können (mindestens Benutzername und Passwort)
- Die Arbeitszeiten müssen einsehbar sein
- Die Zeit muss mit einem Start- und Stopp-Button erfasst werden können

In diesem Abschnitt werden die Abbildungen zum ersten Teil, dem Login mit Benutzernamen und Passwort, weggelassen, da bereits in Aufgabe 3 ein App-Bildschirm mit Login und Registrierung gezeichnet wurde.

Bei dieser Aufgabe wurden alle UI Komponenten richtig erkannt. Was darauf zurückzuführen ist, dass sich die ProbandInnen bereits in den ersten Aufgaben mit den verschiedenen UI Komponenten vertraut gemacht haben.

Gruppe MV und PR

Der Proband MV und die Probandin PR haben diese Aufgabe in einer Gruppe gelöst. Dafür haben sie zwei Bildschirme erstellt. Im ersten Bildschirm haben sie eine Infokarte als Informationselement gewählt, um den/die eingeloggten BenutzerIn anzuzeigen. Darunter haben sie eine Tabelle gezeichnet, um alle bereits aufgezeichneten Arbeitszeiten zu visualisieren. Im zweiten Bildschirm wurde ein Bild eingefügt, das eine Uhr repräsentieren soll, und darunter zwei Buttons, einer für das Starten des Erfassens und einer für das Stoppen. Außerdem wurde ein Textfeld eingeführt, um eventuelle Beschreibungen oder Informationen zur aufgenommenen Zeit zu erfassen.

Das Ergebnis dieser Gruppe erfüllt die Aufgabenstellung und alle geforderten Funktionen sind vorhanden.

Die Abbildungen 6.19 und 6.20 zeigen die beiden Bildschirme, die der Proband MV und die Probandin PR gezeichnet haben.

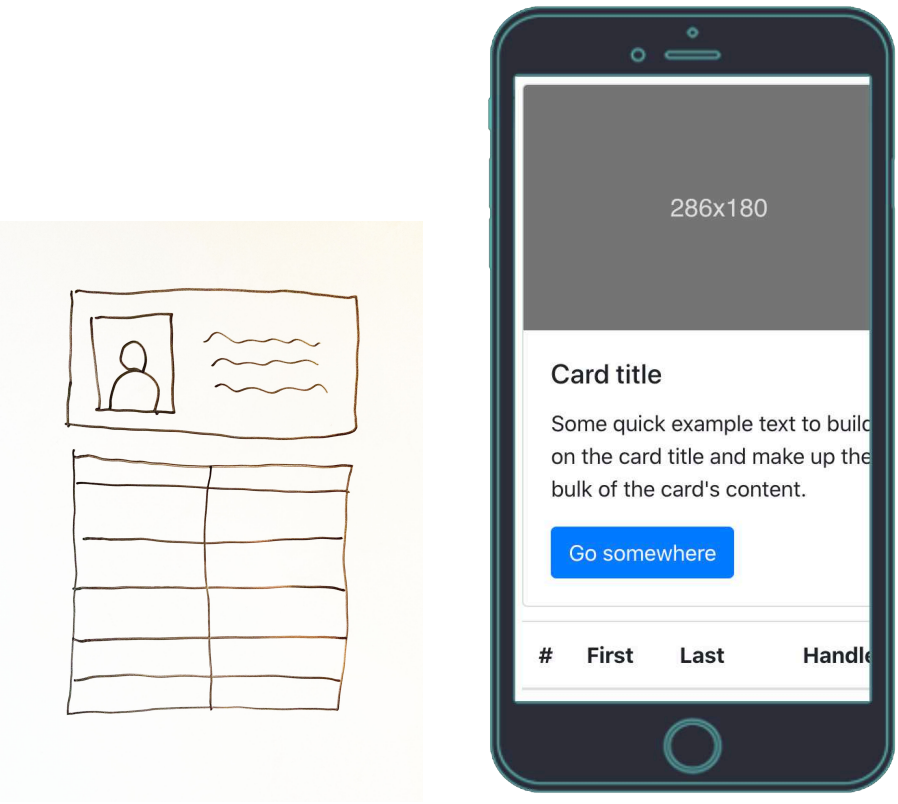


Abbildung 6.19: Ergebnisse Aufgabe 4 – Proband MV und Probandin PR

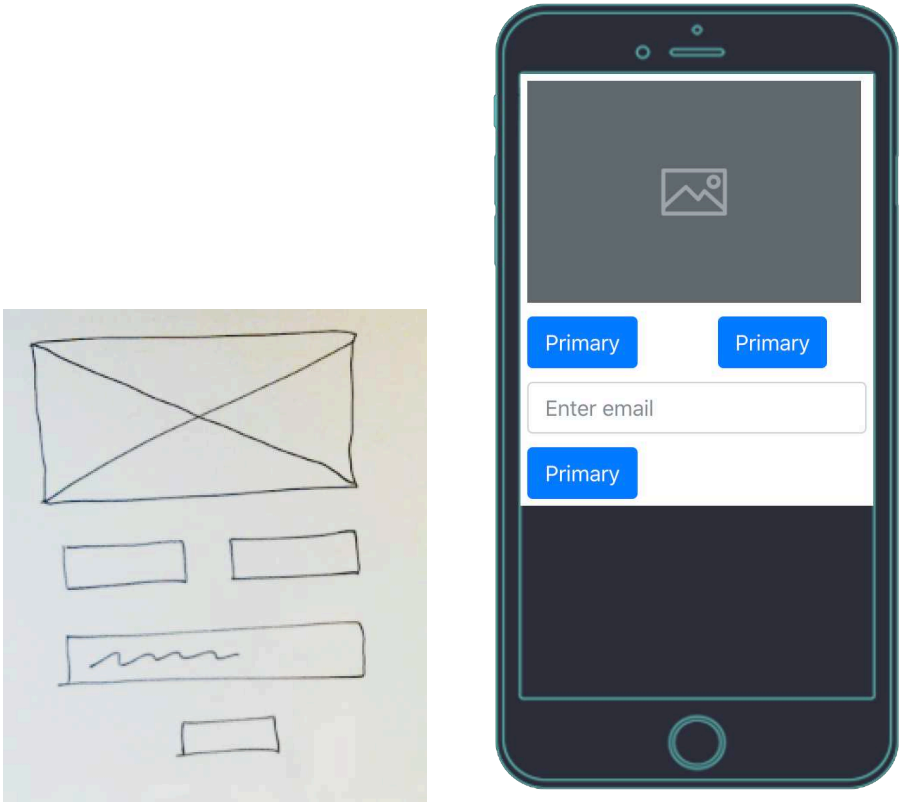


Abbildung 6.20: Ergebnisse Aufgabe 4 – Proband MV und Probandin PR

Gruppe MM und SU

Der Proband MM und die Probandin SU haben diese Aufgabe mit einem anstatt zwei Bildschirmen gelöst. Um die bereits aufgenommenen Arbeitszeiten anzuzeigen, wurde eine Tabelle gezeichnet. Zum Erfassen der Arbeitszeiten wurde ein Button eingeführt. Dieser Button wird zum Starten und zum Stoppen verwendet. Ein Button ist für sie somit ausreichend, um die Arbeitszeiten zu erfassen. Diese Gruppe hat eine kürzere Lösung mit weniger Komponenten gefunden. Die geforderte Funktionalität konnte umgesetzt werden.

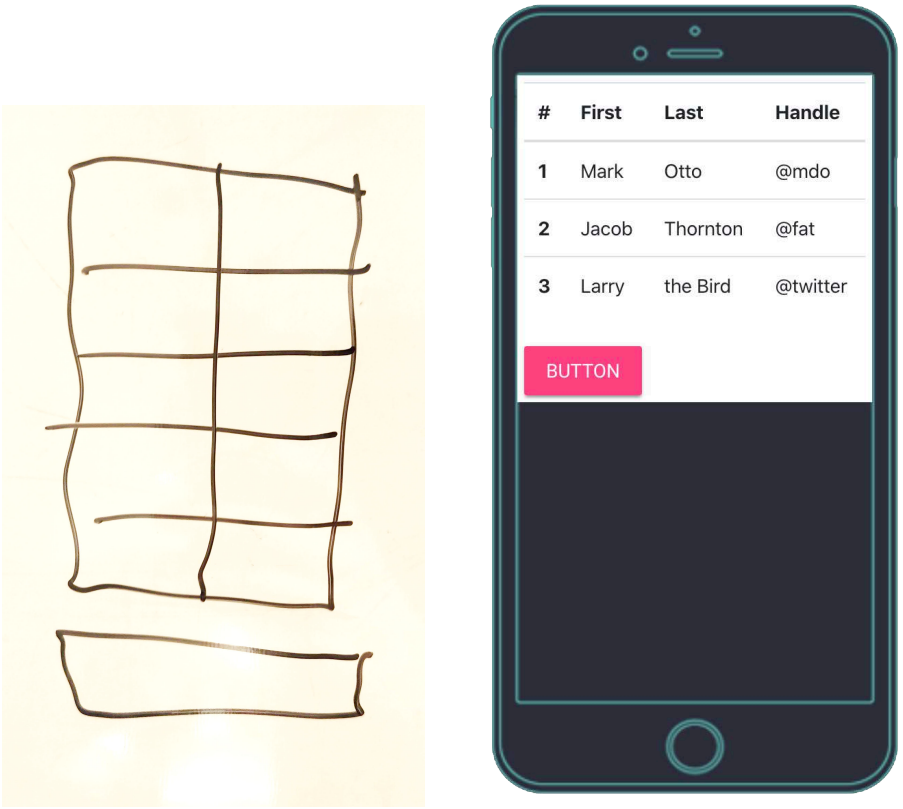


Abbildung 6.21: Ergebnisse Aufgabe 4 – Proband MM und Probandin SU

Gruppe DO und MM2

Die Probanden DO und MM2 haben die Gruppenaufgabe zusammen durchgeführt. Auch sie haben für die Umsetzung der Funktionalität nur einen Bildschirm gezeichnet. Die Abbildung 6.22 zeigt das Ergebnis: Rechts in der Abbildung 6.22 ist nur der untere Teil der App zu sehen. Die Probanden haben eine Infokarte verwendet, um den/die eingeloggten BenutzerIn anzuzeigen. Ein kurzer Text wurde als Beschriftung eingefügt. Dieser wurde vom Tool erkannt, jedoch gibt es nur eine fixe Textlänge bei der Umwandlung und somit ist er etwas zu lang ausgefallen. Zwei Buttons wurden für das Aufzeichnen eingefügt, einer zum Starten und einer zum Stoppen. Darunter befindet sich eine Tabelle zum Visualisieren der bereits aufgenommenen Arbeitszeiten. Die Probanden konnten die geforderte Funktionalität umsetzen.

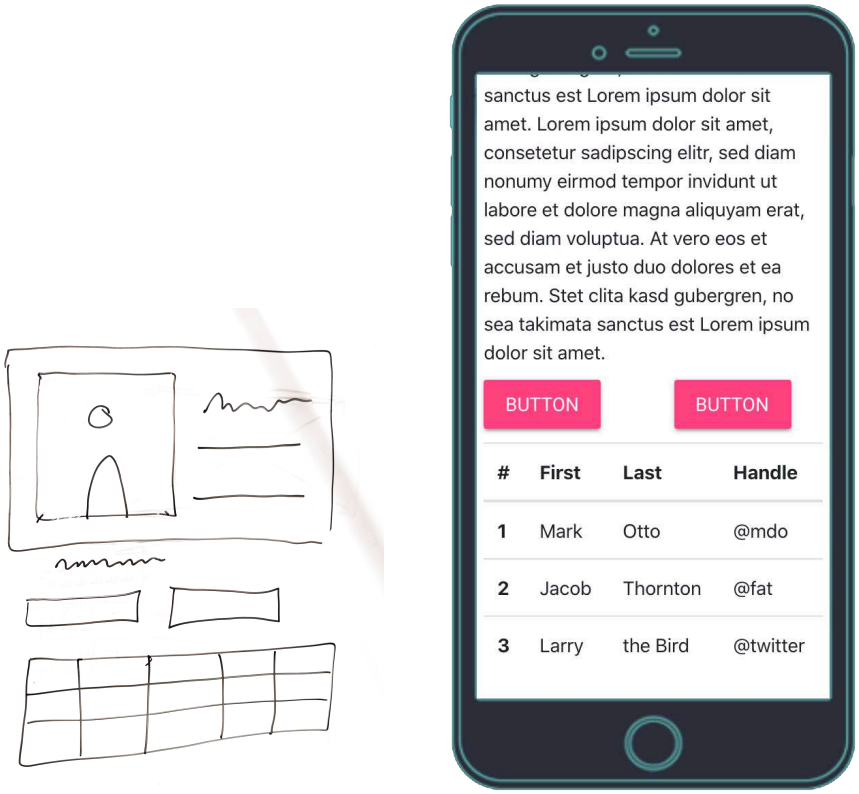


Abbildung 6.22: Ergebnisse Aufgabe 4 – Probanden DO und MM2

6.1.2 Auswertung der Fragebögen von Seiten der ProbandInnen

In diesem Abschnitt werden die Fragebögen, die die ProbandInnen nach der Ausführung der Aufgabenstellung ausgefüllt haben, ausgewertet. Die Fragen 1 bis 4 wurden mit einer Note von 1 bis 10 benotet, wobei 1 die schlechteste und 10 die beste Note war.

Frage 1

Wie verwendbar ist das Tool? Median: 7,5

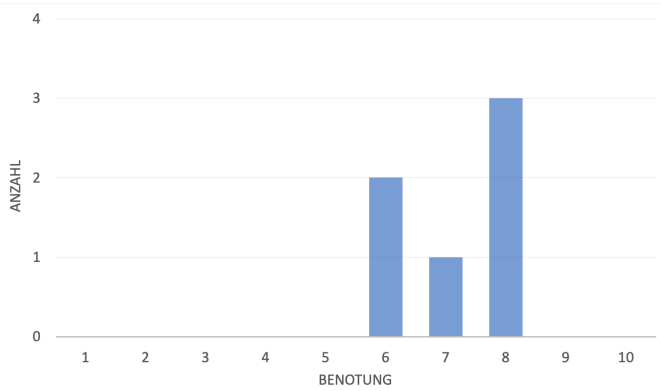


Abbildung 6.23: Ergebnisse des Fragebogens – Aufgabe 1

Frage 2

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

Median: 8

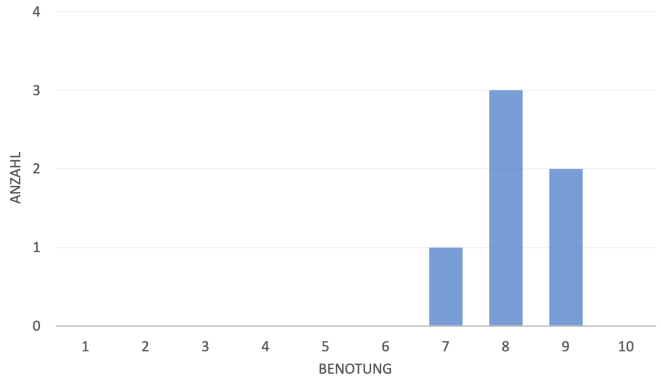


Abbildung 6.24: Ergebnisse des Fragebogens – Aufgabe 2

Frage 3

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

Median: 8

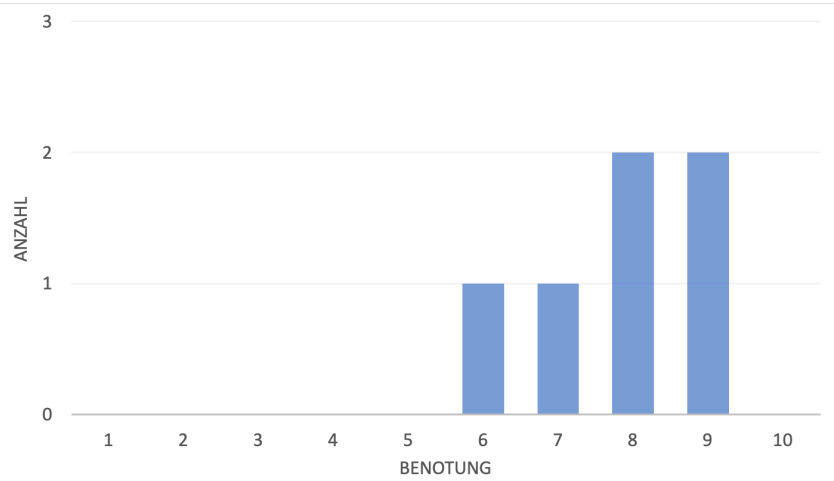


Abbildung 6.25: Ergebnisse des Fragebogens – Aufgabe 3

Frage 4

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

Median: 9

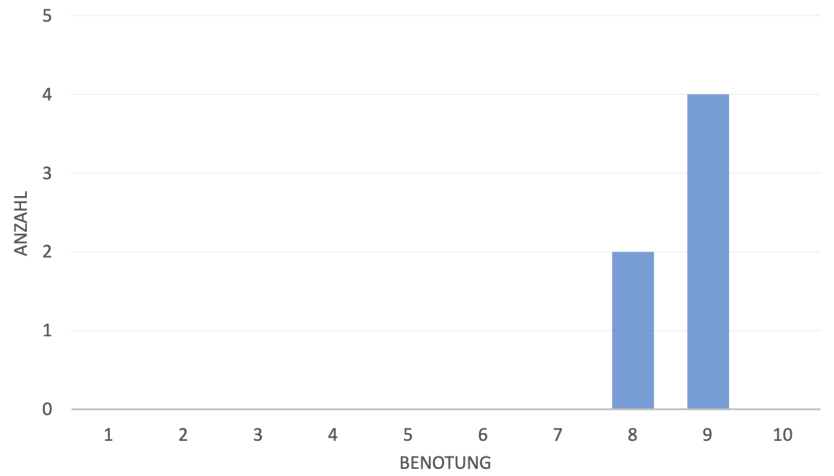


Abbildung 6.26: Ergebnisse des Fragebogens – Aufgabe 4

Im Allgemeinen haben die ProbandInnen das Tool und die Aufgaben gut bewertet. Frage 4 wurde mit einer Durchschnittsnote von 8,66 am besten bewertet, was bedeutet, dass die Funktionalität der Aufgaben gut umgesetzt werden konnte. Frage 1 wurde mit einer Durchschnittsnote von 7,16 am schlechtesten bewertet, was darauf zurückzuführen ist, dass einige Komponenten nicht korrekt umgewandelt wurden.

Frage 5

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

Proband MV: „Variable Länge der Textblöcke fehlt, ansonsten konnten die Designs umgesetzt werden. Intuitiv hätte ich mehrere Zeilen dafür gezeichnet."

Proband PR: „Bei den Aufgaben 2, 3 und 4 gab es keine Probleme - das Design entsprach den Zeichnungen. Bei Aufgabe 1 zeigte das Tool anstelle eines Textes einen Button an, weil mein "Text"zu kurz war."

Proband SU: „Die einzelnen Elemente der Zeichnungen waren teilweise zu nahe und deshalb wurden sie nicht erkannt."

Proband MM: „Großteils, z. B. wäre es gut wenn Abstände auch im Design berücksichtigt werden."

Proband DO: „- "

Proband MM2: „Ja"

Frage 6

Was funktioniert nicht wie erwartet?

Proband MV: „Inputfelder werden teilweise als Navbar erkannt."

Proband PR: „Mein Abstand zwischen den Bildern in Aufgabe 2 war zu gering - könnte sensibler reagieren. Input-Felder wurden oft als Navigationsleisten angezeigt."

Proband SU: „- "

Proband MM: „Navbar und Input werden oft verwechselt"

Proband DO: „Navigation mit Textfeld wurde vertauscht."

Proband MM2: „Navbar und Text wird nicht immer erkannt. Lange Textzeilen erkennt es nicht als Text."

Frage 7

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

Proband MV: „Die Aufgabenstellung konnte wie angegeben umgesetzt werden. Die Ausgabe der App-Skizze sollte jedoch Mobile Komponenten verwenden."

Proband PR: „Ja, die Komponenten waren ausreichend."

Proband SU: „Checkbox, Diagramme"

Proband MM: „Ja, aber ein Passwortfeld wäre noch interessant."

Proband DO: „Ja"

Proband MM2: „Ja konnte umgesetzt werden."

Frage 8

Gibt es Verbesserungsvorschläge?

Proband MV: „Abhängigkeit vom Umgebungslicht reduzieren oder Softwareseitig tweeken, da das Ergebnis dauernt beeinflusst wird."

Proband PR: „Evtl. Sensibilität bei Abständen erhöhen und die Unterschiede zwischen Input-Feldern und Navigationsleisten sollten markanter sein."

Proband SU: „Für kurzen Text eigene Zeichnung und eigene Zeichnung für langen Text."

Proband MM: „Die Navbar könnte man anstelle von Text mit Kästchen zeichnen, vielleicht wird das besser erkannt."

Proband DO: „Evaluieren von Ideen sehr sinnvoll, für Problemstellungen eher nicht."

Proband MM2: „Alles noch ziemlich roh = Prototyp. Sehr viel ausbaufähig = Hohes Potenzial. Farbschemas Vorauswahl wäre eine Idee."

Die Studie hat in den Antworten aus den Fragen 5 - 8 ein gutes Feedback erhalten. Die Antworten sind kritisch, bieten Verbesserungsvorschläge und helfen bei der Analyse der Studie im folgenden Kapitel.

6.2 Analyse

In diesem Abschnitt werden die Ergebnisse der Studie analysiert. Es wird auf die Probleme, die sich beim Skizzieren ergeben haben, eingegangen und die Fragebögen, die die ProbandInnen ausgefüllt haben, werden analysiert. Außerdem werden die Kommentare der ProbandInnen auf den Fragebögen ausgewertet und am Ende wird ein Fazit zur Studie und zu deren Ergebnissen gezogen.

Linienführung

Die ProbandInnen konnten alle Aufgaben, die Ihnen gestellt wurden, durchführen. Auffallend war, dass obwohl die ProbandInnen das Infoblatt mit den zur Verfügung stehenden Komponenten erhalten hatten, jeder die Komponenten unterschiedlich gezeichnet hat. Die Linienführung unter den ProbandInnen war unterschiedlich, so haben einige die Linien gerade gezeichnet (z. B. Probandin PR, siehe Abbildung 6.8), andere wiederum hatten eine etwas welligere Linienführung (z. B. Proband MM, siehe Abbildung 6.9) und bei wieder anderen war die Linienführung etwas schief gezeichnet, sodass die Komponente als Ganzes unförmig wurde (z. B. Proband DO, siehe Abbildung 6.18 zweiter Input und dritter Button). Diese Unregelmäßigkeiten sind vor allem auf das Zeichnen mit Stift auf einer Tafel zurückzuführen, da es die meisten ProbandInnen nicht gewohnt sind, auf einer Tafel zu schreiben bzw. zeichnen. Interessant zu beobachten war, dass die welli-ge Linienführung keinerlei Auswirkung auf das Erkennen der Komponenten hatte, eine schiefe Linienführung konnte größtenteils gut erkannt werden, verursachte jedoch teilweise bei der richtigen Anordnung der Komponenten Probleme (siehe Abbildung 6.2, die rechte Infobox hat eine nach oben gezogene Linie und wurde deshalb vom Tool darunter platziert).

Größe der Komponenten

Die ProbandInnen zeichneten die jeweils gleichen Komponenten unterschiedlich groß. Vor allem der Button und der Text wurden von einigen ProbandInnen zu klein gezeichnet, sodass sie vom Tool nicht erkannt wurden (z. B. Abbildung 6.2, hier wurde der Text zu kurz gezeichnet und dadurch nicht erkannt). Erst nach einer Anpassung der Minimalgröße des Doodle Classifiers konnte dieses Problem bei einigen behoben werden.

Abstand der gezeichneten Komponenten

Zwei Probandinnen haben die UI Komponenten in der ersten Aufgabe zu nahe beieinander gezeichnet (Probandinnen PR und SU). Das führte dazu, dass diese Komponenten vom Doodle Classifier als eine einzige Komponente erkannt wurden und nicht richtig klassifiziert werden konnten. Auch dieses Problem konnte teilweise durch Anpassen der

Parameter des Doodle Classifiers gelöst werden. Die beiden Probandinnen haben in den darauffolgenden Aufgaben mehr Abstand zwischen den Komponenten gelassen und somit trat das Problem nicht mehr auf.

Textkomponente

Die UI Komponente *Text* hat bei einigen ProbandInnen zu Verwirrung geführt. Die Textskizze, eine gewellte Linie, wandelt unabhängig von der gezeichneten Länge der Wellenlinie die Komponente in eine vorgegebene Textlänge um. Die ProbandInnen mussten bei den Aufgaben 1 und 2 einen Text aus mehreren Zeilen einfügen. Die ProbandInnen haben hier mehrere Linien anstatt nur einer Zeile gezeichnet, die zum gewünschten Ergebnis führen würde. Bei der Aufgabe 4, der Gruppenaufgabe, bei der frei gezeichnet werden konnte, haben DO und MM2 ein Label oberhalb der Buttons einfügen wollen. Das Ergebnis war ein langer Text, was nicht den Erwartungen der ProbandInnen entsprach. Um dieses Problem zu lösen, könnte man die Textkomponenten in mehrere Komponenten teilen und in zwei verschiedenen Längen anbieten, z. B. ein Label mit wenigen Worten und eine Zeile, die die ganze Breite der Seite ausnimmt. Das würde das Problem beheben und wäre für die zukünftige Verwendung des Tools sinnvoll.

Input- und Navbar-Skizzen zu ähnlich

Das häufigste Problem, das bei der Durchführung der Studie aufgetreten ist, war das Erkennen der UI Komponenten *Input* und *Navbar*. Die Skizzen dieser beiden Komponenten sind sehr ähnlich und wurden deshalb oft verwechselt. Beide Komponenten bestehen aus einem Rechteck mit gewellten Linien im Inneren. Die Navigationsleiste, hat unterbrochene, gewellte, kurze Linien, die die verschiedenen Menüpunkte repräsentieren sollen, wohingegen beim Input die gewellten Linien durchgehend sind. Des Weiteren ist die Navigationsleiste breiter und erstreckt sich über die gesamte Seite, der Input wird hingegen kürzer gezeichnet. Diese Ähnlichkeiten führen dazu, dass sich diese beiden Komponenten kaum voneinander unterscheiden und somit schwer vom Doodle Classifier richtig klassifiziert werden können.

Um dieses Problem zu lösen, gibt es mehrere Lösungsansätze:

Lösungsansatz 1: Der einfachste Weg ist, für eine der beiden Komponenten neue Skizzen zu erstellen, die sich leichter voneinander unterscheiden. Abbildung 6.27 zeigt links im Bild zwei Möglichkeiten für eine alternative Skizze der Input-Komponente und rechts im Bild zwei mögliche Alternativen für die Navbar-Skizze.

Abbildung 6.27: Alternative Skizzen – Input und Navbar

Lösungsansatz 2: Anpassen der Layout Engine, die für das Positionieren der Komponenten im fertigen Design zuständig ist. Da sich die Navigationsleiste meistens ganz oben auf der Seite oder der App befindet, könnte man darauf schließen, dass auf der restlichen Seite keine Navigationsleiste mehr eingefügt werden soll und dass es sich somit um einen Input handeln muss. Aufgrund dieser Annahme könnte man eine Bedingung einbauen und so die richtige Komponente aufgrund ihrer Position bestimmen.

Beide Lösungen würden zum gewünschten Ergebnis führen, Lösungsansatz 1 wäre dabei aber die schönere Lösung.

Position der Kamera

Bei den ersten Tests wurde die Kamera auf ein Stativ montiert, das nach unten auf ein Blatt Papier gerichtet war, somit war es möglich, die Kamera exakt in der Mitte der Zeichnung zu positionieren. Die Studie wurde dann aber auf einem Whiteboard durchgeführt und die Kamera wurde oberhalb des Whiteboards mit einem Stativ montiert. Dies führte dazu, dass die Kamera nicht exakt in der Mitte des Whiteboards positioniert war. Jedoch konnten alle Bilder aufgenommen werden und die Umwandlung hat mit beiden Varianten funktioniert. Aufgefallen ist, dass bei der Variante mit dem Whiteboard die Komponenten, die sich weiter unten auf dem Whiteboard befanden, schwerer von der Layout Engine positioniert werden konnten. Dies ist auf eine leichte Verzerrung der Kameraaufnahmen durch eine nicht ganz zentrierte Aufzeichnung zurückzuführen. Durch das Erhöhen des Parameters *y-position-tolerance* in der layoutEngineConfig.json konnte dieses Problem behoben werden.

Bessere Ergebnisse mit jeder durchgeführten Aufgabe

Auffallend war bei allen ProbandInnen, dass sich das Ergebnis mit jeder durchgeführten Aufgabe verbessert hat. Die ProbandInnen mussten sich erst an das Zeichnen der Komponenten sowie das Zeichnen auf dem Whiteboard gewöhnen. Dies war auch das Ziel, das beim Erstellen der Reihenfolge der Aufgabenstellungen verfolgt wurde. Nach dem Zeichnen der ersten drei Aufgaben waren die Komponenten schon verinnerlicht und so konnten die ProbandInnen auch schnell eine offene Aufgabenstellung lösen.

Offene Aufgabenstellung beliebteste Aufgabe

Die offene Aufgabenstellung, die 4. Aufgabe, war die unter den ProbandInnen beliebteste Aufgabe. Diese wurde in einer Gruppe bestehend aus jeweils zwei ProbandInnen durchgeführt. Die ProbandInnen konnten ihrer Kreativität freien Lauf lassen. Jede Gruppe hat diese Aufgabe unterschiedlich gelöst, mit verschiedenen Komponenten und einer unterschiedlichen Anordnungen der Komponenten. Jede der Lösungen hat die Funktionalität, die gefordert war, enthalten, jedoch waren die fertigen Designs, die sich daraus ergeben haben, unterschiedlich.

Die ProbandInnen haben bei der Gruppenaufgabe zu Beginn die Aufgabenstellung durchgelesen und haben anschließend begonnen, ihre Ideen zu diskutieren. Die ProbandInnen haben kurz diskutiert, wie sie Vorgehen sollen. Dabei ergaben sich unter anderem folgende Fragen: Wie viele Bildschirme müssen gezeichnet werden? Welche Informationen sollten angezeigt werden? Welche UI Komponenten sollten verwendet werden? Wie und wo sollten die UI Komponenten angeordnet werden? Nach kurzer Absprache der ProbandInnen untereinander, haben sie mit dem Skizzieren begonnen. Skizziert wurde meistens von einer Person, wobei der/die zweite ProbandIn daneben stand und ebenfalls Instruktionen zum Skizzieren und der Auswahl der UI Komponenten gab. Nach dem Erstellen der Skizze haben beide ProbandInnen das fertige Design betrachtet und Diskussionen über die Funktionalität sowie die Bedienbarkeit der App geführt. Daraufhin wurden Änderungen an der Skizze vorgenommen, Komponenten ausgetauscht oder Positionierungen geändert. Nach einigen Änderungen waren die ProbandInnen mit den Ergebnissen zufrieden.

Vor allem diese Aufgabe hat gezeigt, dass das Tool für ein exploratives Gestalten und Evaluieren von User Interfaces gut geeignet ist.

Vorschläge der ProbandInnen für neue Komponenten

Für das Ausführen der Aufgaben waren alle benötigten UI Komponenten vorhanden. Einige ProbandInnen haben Vorschläge für weitere UI Komponenten bei der Frage 7 des Fragebogens angegeben, die für weitere User Interfaces hilfreich sein könnten.

- Ein **Passwortfeld** für den Login- und Registrierungs-Bildschirm: Diese UI Komponente wäre eine gute Ergänzung des Tools, sie könnte beispielsweise als Input-Feld mit Sternen anstelle der gewellten Linien umgesetzt werden.
- Eine **Checkbox** war eine weitere Komponente, die genannt wurde. Diese Komponente würde das Tool ebenfalls gut ergänzen. Das Skizzieren dieser Komponente könnte über ein Quadrat mit Haken erfolgen, jedoch muss hier darauf geachtet werden, dass diese Komponente nicht zu klein gezeichnet wird, ansonsten würde sie möglicherweise nicht richtig erkannt.

- **Diagramme** sind eine häufig verwendete UI Komponente zur Visualisierung von Daten. Man könnte sie mit zwei Achsen, X und Y, und zusätzlichen Balken skizzieren.

Mobile vs. Desktop-Umwandlung

Einige ProbandInnen haben angemerkt, dass die UI Komponenten für die Apps nicht *mobile friendly* sind. Im Zuge dieser Arbeit wurden zwei Frontend Frameworks eingebunden: Bootstrap und Material Design Lite Framework. Beide Frameworks sind nicht primär für mobile Endgeräte ausgelegt. Das Material Design Lite Framework liefert die besseren mobilen Ergebnisse. Für eine bessere Mobile Lösung könnte ein weiteres Framework eingebunden werden, das für mobile Endgeräte ausgelegt ist. Hierfür würde sich zum Beispiel das Ionic- oder React Native Framework anbieten.

Vergleich der Berufsgruppen der ProbandInnen

Bei der Durchführung der Studie wurde darauf geachtet, ProbandInnen aus verschiedenen Berufsgruppen zu wählen. Informatiker, Grafiker und Personen ohne Vorkenntnisse im Bereich User Interface Design haben die Studie durchgeführt. Für die Gruppenaufgabe wurde versucht, die ProbandInnen verschiedener Berufsgruppen zusammenzuführen.

Die Informatiker haben die meisten Kenntnisse im Bereich User Interface Design, dies konnte auch bei der Ausführung der Aufgaben beobachtet werden. Alle UI Komponenten, die zur Verfügung gestellt wurden, wurden ohne weitere Erklärungen gleich erkannt und richtig verwendet. Auch bei der Gruppenaufgabe wurde detaillierter auf die Funktionalität geachtet. Informatiker konnten leichter alle Use Cases durchdenken.

Die Grafiker hatten bereis Vorkenntnisse im Bereich User Interface Design. Die UI Komponenten waren alle ohne weitere Erklärungen bekannt und wurden richtig verwendet. Die Probanden aus dieser Berufsgruppe haben zu viel vom fertigen Design erwartet, da sie vor allem viel Wert auf ein gutes Design legen. Sie würden sich als Erweiterung zum Beispiel verschiedene Farbschemen zur Auswahl wünschen oder verschiedene Schriftarten, dadurch könnte ihrer Meinung nach das Ergebnis gleich besser an die Bedürfnisse der Kunden angepasst werden.

Die Probandinnen ohne Vorkenntnisse haben die meisten UI Komponenten gleich erkannt und richtig verwendet. Interessant zu beobachten war, dass auch die Probandinnen ohne Vorkenntnisse gut skizzieren konnten und schnell zu den gewünschten Ergebnissen kamen. Da keinerlei Vorkenntnisse in diesem Bereich vorhanden waren, konnten nach meinen Beobachtungen diese Probandinnen nicht alle Use Cases für die 4. Aufgaben durchspielen.

Durch die intuitive Herangehensweise durch das Skizzieren mit Hand konnten alle ProbandInnen schnell und ohne weitere Erklärungen beginnen.

Teamwork und Kollaboration

Bei der Durchführung der Gruppenaufgabe konnte beobachtet werden, wie die ProbandInnen miteinander kollaboriert haben. Es konnte festgestellt werden, dass durch die Umwandlung der Skizze in ein fertiges Design, die ProbandInnen die Funktionalität der zu erstellenden App gleich evaluieren und gegebenenfalls Änderungen vornehmen konnten.

Die ProbandInnen haben zu Beginn dieser Aufgabe zunächst die geforderten Funktionalitäten kurz besprochen und dann angefangen zu zeichnen. Sobald die erste Skizze fertig war, haben die ProbandInnen das fertige Design begutachtet und versucht, alle möglichen Use Cases durchzuspielen. Dabei haben die ProbandInnen auch gemerkt, welche Funktionalitäten anders besser umgesetzt werden könnten, und konnten die Skizze dementsprechend ausbessern.

Der Proband MM und die Probandin SU haben beispielsweise für das Erfassen der Zeiten zu Beginn einen *start* und einen *end* Button eingefügt. Nach einigen Überlegungen haben sie aber festgestellt, dass ein Button ausreichend für die Umsetzung der Funktionalität ist, da nach einem Klick auf den *start* Button dieser auch wieder zum Stoppen verwendet werden kann.

Die Gruppe DO und MM2 hatte zu Beginn zwei Bildschirme gezeichnet, um alle Funktionalitäten umzusetzen. Nach dem Betrachten der fertigen Designs und einigen Überlegungen stellten sie fest, dass ein Bildschirm für die Aufgabe ausreichend ist. So haben sie das Design umgestellt und einen Bildschirm erstellt mit den gesamten geforderten Funktionen.

Erkennungsanalyse

Die ProbandInnen haben bei der Ausführung der Aufgaben insgesamt 129 Komponenten gezeichnet. Davon wurden 121 Komponenten richtig vom Tool erkannt und umgewandelt, 8 Komponenten wurden in falsche Komponenten umgewandelt. Das bedeutet, dass 93,8% der Komponenten richtig erkannt wurden.

Von den 129 gezeichneten Komponenten wurden 126 richtig auf dem fertigen Design positioniert, nur 3 Komponenten wurden falsch im Layout angeordnet. Das sind 97,67% der Komponenten, die im fertigen Design richtig positioniert wurden.

Die Diagramme in Abbildung 6.28 veranschaulichen die Erkennungsanalyse. Links in der Abbildung wird die Komponentenerkennungsperformance und rechts die Positionserkennungsperformance dargestellt.

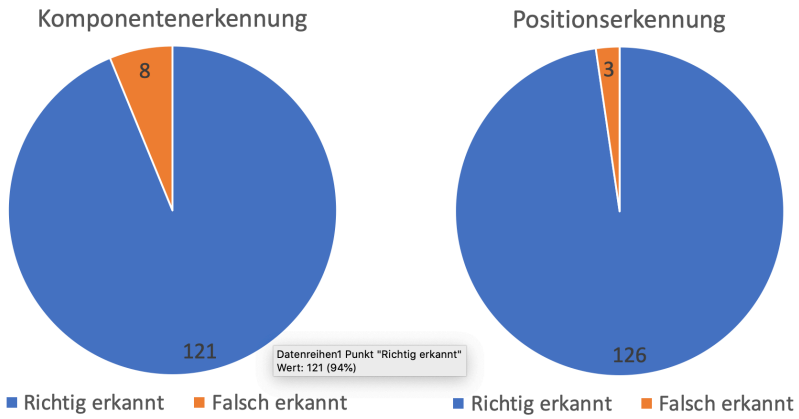


Abbildung 6.28: Komponenten- und Positionserkennung

Kann der Designprozess beschleunigt werden?

Ziel dieser Arbeit ist es, herauszufinden, ob mit dieser Herangehensweise der Designprozess beschleunigt werden kann. Da üblicherweise mehrere Stakeholder im Designprozess involviert sind und mehrere Iterationen benötigt werden, um zu einem ersten Prototypen zu gelangen, kann dies viel Zeit in Anspruch nehmen. Dabei entstehen mit steigender Iteration immer neue Artefakte mit steigendem Detailgrad. Mit dem Tool, das im Rahmen dieser Arbeit entwickelt wurde, können einige dieser Schritte übersprungen werden.

Der Projektmanager oder Designer kann mit dem Kunden zusammen erste Skizzen erstellen, die dann gleich vom Tool in ein fertiges Design umgewandelt werden. Das führt dazu, dass der Kunde gleich eine erste Version des Produktes zu Gesicht bekommt und somit gleich ein Ergebnis sieht. Dadurch können Funktionalitäten, die das Produkt beinhalten soll, gleich evaluiert und getestet werden. So kann verhindert werden, dass Funktionalitäten nicht richtig umgesetzt oder vom Projektmanager falsch verstanden werden, und Funktionalitäten können bereits während des Skizzierens hinzugefügt oder geändert werden.

Für das Auswerten von Ideen kann das Tool sehr hilfreich sein, da die Ideen schnell sichtbar werden und so Use Cases leichter durchgespielt werden können. Auch bei konkreten Problemstellungen können so Fehler im Design schnell erkannt und richtiggestellt werden.

Das Tool wurde so erstellt, dass es mit neuen Frontend Frameworks erweitert werden kann. Dadurch ist es möglich, firmeninterne UI Komponenten einzuspielen, und das fertige Design, das vom Tool generiert wird, sieht dem User Interface des Endproduktes bereits ähnlich und könnte auch für die Implementierung verwendet werden.

6.3 Fazit

Das Ergebnis der Studie hat aufgezeigt, dass das Tool die Skizzen ideengerecht in fertige Designs umwandeln kann. Der Designprozess kann vor allem in den frühen Phasen des Designprozesses unterstützt werden. Jeder Teilnehmer am Designprozess kann nach einer kurzen Eingewöhnungsphase durch den intuitiven Ansatz selbst am Designprozess teilnehmen und seine Ideen mit einbringen.

Vor allem in den frühen Phasen des Designprozesses, bei denen die verschiedenen Stakeholder (zum Beispiel Kunden und Projektmanager) die Requirements analysieren und erarbeiten, kann das Tool gut eingesetzt werden. Durch das Skizzieren und Umwandeln mit dem Tool kann schnell ein Prototyp erstellt werden. Der geringe Zeitaufwand, der dabei notwendig ist, ermöglicht es, dass verschiedene Ideen zu der Problemstellung getestet und evaluiert werden können. Dadurch kann vermieden werden, dass mit einer falschen ersten Idee gearbeitet wird, die am Ende nicht der richtige Ansatz ist, wodurch auch ein unnötiges Ausarbeiten eines Prototypen verhindert wird.

Jeder involvierte Stakeholder hat durch das Skizzieren mit der Hand die Möglichkeit, selbst seine Ideen zu Papier zu bringen. Ein unnötiges und teilweise umständliches Beschreiben seiner Vorstellungen kann so umgangen werden.

Das Tool selbst könnte für einen solchen Einsatz etwas erweitert werden. Verschiedene Farbschemen wären von Vorteil, da so leichter die vom Kunden gewünschten oder passenden Farben verwendet werden könnten, wodurch der Prototyp bessere Ergebnisse für den Kunden liefern könnte. Sollte es sich um kundenspezifische Anpassungen eines Produktes handeln, wäre es von Vorteil, produktspezifische Komponenten einzubinden, so wäre das Ergebnis ein fertiger Prototyp und könnte auch für die Implementierung weiter verwendet werden. Dies würde ein erneutes Implementieren des Interfaces unnötig machen und eine weitere Zeitersparnis mit sich bringen.

KAPITEL 7

Diskussion und Ausblick

In diesem Kapitel der Arbeit werden die Stärken und Schwächen des Tools erarbeitet und es wird erläutert, warum die ausgewählten Technologien verwendet wurden, um das Tool zu entwickeln.

7.1 Stärken und Schwächen

Das Tool funktioniert zuverlässig und schnell. Es handelt sich bei diesem Tool um einen Prototypen, der zum Evaluieren der Herangehensweise für einen schnelleren Designprozess erstellt wurde. Das Tool implementiert alle notwendigen Funktionalitäten, um die Umsetzung und Verwendbarkeit zu testen. Da es sich um einen Prototypen handelt, gibt es noch Spielraum für Verbesserungen und Erweiterungen.

Für das Training des Doodle Classifiers, der für das Klassifizieren zuständig ist, wurden nur sehr wenig Trainingsdaten verwendet, nämlich 6 - 10 verschiedene Bilder pro UI Komponenten. Je mehr Skizzen trainiert werden, desto besser kann klassifiziert werden. Diese wenigen Trainingsdaten haben aber ausgereicht, um die UI Komponenten zu erkennen und klassifizieren, daher wurde auf das Trainieren mehrerer Skizzen verzichtet. Die Trainingsdaten wurden alle von einer Person gezeichnet. Weitere Trainingsdaten von verschiedenen Personen und somit leicht abweichende Skizzen aufgrund von unterschiedlicher Linienführung oder verschiedenen Größenverhältnissen würden allerdings zu einer genaueren Klassifizierung beitragen.

7.2 Angemessenheit der verwendeten Technologien

Der Doodle Classifier, der für das Klassifizieren der gezeichneten Skizzen zuständig ist, wurde bei einem ähnlichem Projekt von Airbnb (siehe Abschnitt 2.4) verwendet. Er wurde mit einen Open Source Framework erstellt und konnte daher leicht erweitert werden. Er verfügt über alle geforderten Funktionalitäten und wurde deshalb verwendet.

Das OSC Protokoll wurde für die Übertragung der Daten zwischen Doodle Classifier und dem Tool, das im Zuge dieser Arbeit implementiert wurde, verwendet, da es bereits vom Doodle Classifier implementiert war und direkt verwendet werden konnte. Daher war eine weitere Implementierung zum Beispiel über Web Socket überflüssig. Das OSC Protokoll hat zuverlässig und schnell funktioniert.

Electron als Wrapper für die Node.js Applikation und dem Frontend in AngularJS hat den Vorteil gebracht, dass das Tool in eine Applikation verpackt werden konnte, was dem Umgang damit erleichtert hat. Der Doodle Classifier konnte integriert werden und wurde automatisch beim Starten des Programms als eigenständiges Program gestartet.

Für die Umwandlung in das fertige Design wurden Frontend Frameworks verwendet. Das Tool wurde so entwickelt, dass neue Frameworks einfach eingebunden werden können. Für diese Arbeit wurden zwei Frontend Frameworks ausgewählt, das Bootstrap und das Material Design Lite. Aufgrund der großen Anzahl unterschiedlicher Frontend Frameworks wurden diese beiden aufgrund ihrer Popularität und Verbreitung stellvertretend ausgewählt.

7.3 Future Work

Das Tool könnte in Zukunft durch einige neue Features erweitert und angepasst werden, um noch detailliertere Prototypen zu erstellen und in der Praxis eingesetzt zu werden.

Durch das Hinzufügen von Farbschemen, die der/die BenutzerIn des Tools über die Einstellungen auswählen kann, könnte das Tool an den Kunden angepasste Ergebnisse erzielen. So könnte man zum Beispiel verschiedene Buntstifte verwenden, um dieses Ziel zu erreicht.

Eine Studie, in der das Tool in einem Unternehmen getestet und mit den firmeninternen UI Komponenten erweitert wird, könnte interessante Ergebnisse liefern. Vor allem in Unternehmen, die ein Produkt an Kundenwünsche anpassen, könnte das Tool gute Ergebnisse erzielen und großen Anklang finden, da dort die spezifischen UI Komponenten eingebunden werden könnten. Das fertige Design könnte so auch gleich für die konkrete Implementierung verwendet werden.

Für eine bessere Verwendung könnte das Tool so erweitert werden, dass die aufgenommenen Skizzen abgespeichert werden und zu einem späteren Zeitpunkt wieder abrufbar

wären. Zurzeit wird nur die aktuelle Kameraaufnahme in ein fertiges Design umgewandelt.

Anstelle der Kamera könnte eine App entwickelt werden, die über die integrierte Kamera des Smartphones Fotos oder Streams aufnimmt. Diese Aufnahmen könnten zur Umwandlung in ein fertiges Design an das Tool übermittelt werden. So könnten man auch die Ergebnisse der Umwandlung wiederum auf dem Smartphone anzeigen, wo sie direkt getestet werden könnten. Ein umständliches Aufbauen und Einrichten einer Umgebung mit Whiteboard und Kamera wäre somit überflüssig und das Tool könnte überall und jederzeit verwendet werden.

Interessant für eine Erweiterung des Tools wäre das Hinzufügen von Verlinkungen (Links) in Buttons und Navigationsleisten. Das Tool könnte so angepasst werden, dass durch das Zeichnen von Pfeilen von einem Button zu einer zweiten Skizze durch Klicken auf diesen Button der zweite Bildschirm geöffnet wird. Dadurch könnten die Prototypen im Tool bedient werden, was für das Evaluieren des Designs sinnvoll wäre. Insbesondere wäre es interessant, wenn sich diese Funktionalität über die Skizzen realisieren lassen würde und nicht mittels manuellen Verknüpfens der einzelnen Bildschirme über das Tool.

KAPITEL 8

Zusammenfassung

User Interfaces sind im Laufe der Zeit immer detaillierter geworden. Während es zu Beginn nur eine einfache Kommandozeile mit Texteingabe gab, werden heute aufwändige grafische Oberflächen entwickelt. Daher hat sich auch der Aufwand für das Erstellen eines User Interfaces erhöht. Das Designen von User Interfaces kann ein langer und aufwendiger Prozess sein. Die verschiedenen Stakeholder, die dabei involviert sind, müssen die Requirements bestimmen und analysieren, dabei sind meist mehrere Iterationen notwendig, um zum gewünschten Ergebnis zu gelangen. Dabei entstehen bei jeder Iteration immer neue Artefakte mit steigendem Detailgrad.

Das Ziel dieser Arbeit war es, diesen Designprozess zu unterstützen und zu verkürzen. Dafür wurde ein Tool entwickelt, das handgemalte Skizzen eines User Interfaces in ein fertiges Design umwandelt. Jedem beteiligten Stakeholder wird es so ermöglicht, intuitiv mit Stift und Papier ein Design zu erstellen. Durch das Umwandeln der Skizze in ein fertiges Design in Form eines HTML- und CSS-basierten User Interfaces kann das Design gut beurteilt werden.

Es gibt bereits einige verwandte Arbeiten, die einen ähnlichen Ansatz verfolgen, jedoch kann keines der Tools die Komponenten in einem Layout richtig anordnen, d. h. die Komponenten werden einfach untereinander angeordnet. Das im Zuge dieser Arbeit erstellte Tool kann auch Komponenten, die nebeneinander stehen, im fertigen Design richtig anordnen. Für die Umwandlung wurden zwei Frontend Frameworks eingebaut, es ist aber auch möglich, problemlos weitere Frontend Frameworks einzubinden, was ebenfalls kein anderes Tool ermöglicht.

Die für diese Arbeit relevanten Grundlagen und Technologien beinhalten den Designprozess von Benutzeroberflächen (Sketching und Prototyping), Komponenten und Layouts von Benutzeroberflächen, Design Guidelines, Convolutional Neural Networks und Source Code Generierung. Sketching ist im Unterschied zum Prototyping ein grober Entwurf mit wenigen bis gar keinen Details. Beim Prototyping hingegen werden bereits Farben,

Beschriftungen und andere detailliertere Komponenten verwendet. Ein Prototyp kommt dem Endprodukt bereits nahe. Die Benutzeroberflächen bestehen aus verschiedenen UI Komponenten, die in einem Layout angeordnet werden. Für das Erstellen von User Interfaces können Design Guidelines verwendet werden, das sind Sammlungen von Empfehlungen für bewährte Designmöglichkeiten. Convolutional Neural Networks sind ein Deep-Learning-Modell zum Analysieren von Bildern und wurde zum Klassifizieren der Skizzen im Doodle Classifier verwendet. Für das Generieren des Source Codes wird ein Mapping Ansatz verwendet.

Für das Erkennen und Klassifizieren der UI Komponenten in der Skizze wurde ein bereits bestehendes Tool, der Doodle Classifier, verwendet. Der Doodle Classifier musste angepasst und anschließend mittels Beispielskizzen trainiert werden. Die Ergebnisse des Doodle Classifiers wurden mittels OSC Protokoll an das für diese Arbeit erstellte Tool weitergeleitet. Das Tool wurde in Node.js erstellt und in einen Electron Container verpackt. Für das Erstellen des Frontends wurde AngularJS und ein WebSocket verwendet, der alle Änderungen aus der Skizze im fertigen Design anzeigen kann. Das Tool besteht aus zwei Komponenten: Zum einen aus der Layout Engine, die die erkannten UI Komponenten vom Doodle Classifier erhält und diese versucht, richtig im fertigen Design anzuordnen. Zum anderen wurde für das Anordnen der Komponenten ein Algorithmus implementiert, der erkennt, ob sich zwei Komponenten nebeneinander oder untereinander befinden. Die Ergebnisse des Doodle Classifiers werden an die Mapping Engine weitergeleitet. Die Mapping Engine wandelt die Labels der erkannten Komponenten in HTML- und CSS-Elemente unter Verwendung eines Frontend Frameworks um. Für diese Arbeit wurden zwei Frontend Frameworks eingebunden, die im Tool ausgewählt werden können. Die Mapping Engine wurde so entworfen, dass weitere Frameworks leicht hinzugefügt werden können.

Zum Evaluieren des Tools wurde eine Studie durchgeführt. Sechs ProbandInnen haben eine Aufgabenstellung mit vier verschiedenen Aufgaben erhalten, die auf einem Whiteboard gelöst werden mussten. Bei zwei Aufgaben musste eine Webseite nachskizziert werden, eine Aufgabe bestand darin, eine App anhand einer Vorlage zu skizzieren und in eine offenen Aufgabe musste in einer Gruppe aus jeweils zwei Personen eine App gezeichnet werden. Die ProbandInnen wurden aus verschiedenen Berufsgruppen ausgesucht. Nach Ausführung der Aufgabenstellung erhielten die ProbandInnen einen Fragebogen zum Ausfüllen. Dieser Fragebogen wurde zusammen mit eigenen Beobachtungen während der Durchführung der Studie verwendet, um das Tool zu evaluieren.

Die Ergebnisse der Fragebögen haben gezeigt, dass das Tool alle Skizzen ideengerecht in fertige Designs umwandeln kann. Mit jeder Aufgabe, die die ProbandInnen ausgeführt haben, hat sich das Ergebnis verbessert. Eine schnelle Lernkurve konnte beobachtet werden, was auf die intuitive Herangehensweise durch das Skizzieren mit Stift und Papier zurückzuführen ist. Die ProbandInnen haben durch ihre Angaben auf dem Fragebogen einige gute Ideen für eine Verfeinerung und/oder Erweiterung des Tools vorgebracht. Der Doodle Classifier wurde mit nur 6 - 10 Skizzen pro UI Komponente trainiert. Diese waren jedoch ausreichend, um die Komponenten zu erkennen und zu klassifizieren.

Durch das Einspielen von weiteren Trainingsskizzen könnte das Tool noch genauere Ergebnisse liefern. Skizzen von verschiedenen Personen könnten zu einer zuverlässigeren Klassifizierung führen.

Es konnte festgestellt werden, dass dieses Tool den Designprozess, vor allem in den frühen Phasen, gut unterstützen und beschleunigen kann. Jeder Stakeholder, der involviert ist, kann so intuitiv seine Ideen zu Papier bringen und erhält durch das Tool gleich ein fertiges Design, das dabei hilft, diese Idee zu evaluieren und auszuarbeiten. Das Durchspielen von Use Cases kann anschließend dazu beitragen, eventuelle Änderungen vorzunehmen oder einen komplett neuen Ansatz zu versuchen. Da das Skizzieren schnell und intuitiv ist, geht dadurch kaum Zeit verloren.

Abbildungsverzeichnis

Listings

Abkürzungen

CNN Convolutional Neural Networks. 26

DSL Domänenspezifische Sprache. 7

ml4a-ofx Machine Learning for Artists - Open Frameworks. 4

OSC Open Sound Control. 4

UI User Interface. 5

WS Websocket. 5

Literatur

Wissenschaftliche Literatur

[2] Allen B. Tucker. *Computer Science Handbook, Second Edition*. Chapman & Hall/-CRC, 2004. ISBN: 158488360X.

[3] Brad Myers. "Challenges of HCI Design and Implementation". In: *Interactions* 1.1 (Jän. 1994), S. 73–83. ISSN: 1072-5520. DOI: 10.1145/174800.174808. URL: https://doi.org/10.1145/174800.174808.

[4] Tuan Anh Nguyen und Christoph Csallner. "Reverse Engineering Mobile Application User Interfaces with REMAUI". In: *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*. ASE '15. Lincoln, Nebraska: IEEE Press, 2015, S. 248–259. ISBN: 9781509000241. URL: https://doi.org/10.1109/ASE.2015.32.

[5] Kevin Moran u. a. "Automated Reporting of GUI Design Violations for Mobile Apps". In: *Proceedings of the 40th International Conference on Software Engineering*. ICSE '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, S. 165–175. ISBN: 9781450356381. DOI: 10.1145/3180155.3180246. URL: https://doi.org/10.1145/3180155.3180246.

[6] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN: 978-0123740373.

[10] Tony Beltramelli. "Pix2code: Generating Code from a Graphical User Interface Screenshot". In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '18. Paris, France: Association for Computing Machinery, 2018. ISBN: 9781450358972. URL: https://doi.org/10.1145/3220134.3220135.

[11] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, S. 1097–1105.

[12] James A. Landay. “SILK: Sketching Interfaces Like Krazy”. In: *In Proceedings of CHI'96 on Human factors in computer systems: common ground, ACM.* Press, 1996, S. 398–399.

[13] James A. Landay und Brad A. Myers. “Interactive Sketching for the Early Stages of User Interface Design”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, S. 43–50. ISBN: 0201847051. DOI: 10.1145/223904.223910. URL: https://doi.org/10.1145/223904.223910.

[18] Christoph Wimmer. “Interface und Interaction Design”. TU Wien. 2017/18.

[21] Anna Debenham. *FRONT-END STYLE GUIDES : creating and maintaining style guides for websites.* City: Anna Debenham, 2017. ISBN: 978-1-5272-0460-7.

[23] J.O. Meiert und E.A. Meyer. *The Little Book of HTML/CSS Frameworks.* O'Reilly Media, 2015. ISBN: 9781491920169. URL: https://books.google.at/books?id=y3v0vQEACAAJ.

[25] Silvio Moreto. *Bootstrap 4 - Responsive Web Design.* Packt Publishing, 2017. URL: https://www.xarg.org/ref/a/B073SB4X15/.

[27] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning (Adaptive Computation and Machine Learning series).* The MIT Press, 2016. URL: http://www.deeplearningbook.org.

[28] Saad Albawi, Tareq Abed Mohammed und Saad ALZAWI. “Understanding of a Convolutional Neural Network”. In: Aug. 2017. DOI: 10.1109/ICEngTechnol.2017.8308186.

[29] Yongxin Yang und Timothy M. Hospedales. “Deep Neural Networks for Sketch Recognition”. In: *ArXiv* abs/1501.07873 (2015).

[31] Vincent Dumoulin und Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).

[32] Vinod Nair und Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning.* ICML'10. Haifa, Israel: Omnipress, 2010, S. 807–814. ISBN: 9781605589077.

[33] Keiron O'Shea und Ryan Nash. “An Introduction to Convolutional Neural Networks”. In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458.

[34] Jeroen Arnoldus u. a. *Code Generation with Templates.* 1. Aufl. Berlin Heidelberg: Springer Science & Business Media, 2012. ISBN: 978-9-491-21656-5.

[35] Eugene Syriani, Lechanceux Luhunu und Houari A. Sahraoui. “Systematic Mapping Study of Template-based Code Generation”. In: *CoRR* abs/1703.06353 (2017). arXiv: 1703.06353. URL: http://arxiv.org/abs/1703.06353.

[36] Jack Herrington. *Code Generation in Action*. USA: Manning Publications Co., 2003. ISBN: 1930110979.

[38] Siddharth Kulkarni. *AI Assisted User Interface Development*. Department of Computer Science, San Jose State University, 2018. URL: http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Fall18/siddharth/CS297_Report.pdf.

[41] Kitti Kredpattanakul und Yachai Limpiyakorn. "Transforming JavaScript-Based Web Application to Cross-Platform Desktop with Electron". In: *Information Science and Applications 2018*. Hrsg. von Kuinam J. Kim und Nakhoon Baek. Singapore: Springer Singapore, 2019, S. 571–579. ISBN: 978-981-13-1056-0.

[42] Matthew Wright und Adrian Freed. "Open SoundControl: A New Protocol for Communicating with Sound Synthesizers". In: *ICMC*. 1997.

[43] Ian Fette und Alexey Melnikov. "The WebSocket Protocol". In: *RFC* 6455 (2011), S. 1–71.

[44] Vanessa Wang, Frank Salim und Peter Moskovits. "The WebSocket Protocol". In: *The Definitive Guide to HTML5 WebSocket*. Berkeley, CA: Apress, 2013, S. 33–60. ISBN: 978-1-4302-4741-8. DOI: 10.1007/978-1-4302-4741-8_3. URL: https://doi.org/10.1007/978-1-4302-4741-8_3.

[45] David Lopez-Cotarelo Flemons. *Flashlight drawing game with a neural network A.I.* 2017. URL: http://essay.utwente.nl/73063/.

[46] Valentin Ritschl, Roman Weigl und Tanja Stamm. *Wissenschaftliches Arbeiten und Schreiben: Verstehen, Anwenden, Nutzen für die Praxis*. Springer-Verlag, 2016.

Online-Referenzen

[1] Khoi Vinh. *The Tools Designers Are Using Today*. URL: http://tools.subtraction.com/ (besucht am 10. 10. 2015).

[7] *Node.js*. URL: https://nodejs.org (besucht am 13. 04. 2020).

[8] *Electron GitHub*. URL: https://github.com/electron/electron (besucht am 30. 08. 2019).

[9] *AngularJS*. URL: https://angularjs.org (besucht am 13. 04. 2020).

[14] Benjamin Wilkins. *Sketching Interfaces - Generating code from low fidelity wireframes*. URL: https://airbnb.design/sketching-interfaces/ (besucht am 30. 08. 2019).

[15] Markus Siering. *Teaching a machine to convert wireframes into code*. URL: https://tech.xing.com/teaching-a-machine-to-convert-wireframes-into-code-f9333e125e61 (besucht am 30. 08. 2019).

[16] Benjamin Wilkins. *Twitter Post - Benjamin Wilkins*. URL: https://twitter.com/thatbenlifetho/status/932663867192180736.

[17] *Uizard*. URL: https://uizard.io/ (besucht am 19. 02. 2020).

[19] *2019 Design Tools Survey*. URL: https://uxtools.co/survey-2019/#prototyping (besucht am 19. 02. 2020).

[20] Tsvetelina Lazarova. *Low Fidelity Wireframes vs High Fidelity Wireframes*. 2018. URL: https://mentormate.com/blog/low-fidelity-wireframes-vs-high-fidelity-wireframes/?utm_source=medium&utm_medium=social&utm_campaign=5-8-18%20fidelity%20wireframes%20blog.

[22] Brad Frost. *Style Guides*. 2014. URL: https://bradfrost.com/blog/post/style-guides/#patterns.

[24] *Bootstrap*. URL: https://getbootstrap.com/ (besucht am 30. 08. 2019).

[26] *Material Design Lite*. URL: https://getmdl.io/ (besucht am 30. 08. 2019).

[30] Greg Surma. *Image Classifier - Cats vs Dogs*. 2019. URL: https://towardsdatascience.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8.

[37] Gene Kogan. *Machine Learning for Artists - DoodleClassifier Guide*. URL: http://ml4a.github.io/guides/DoodleClassifier/ (besucht am 30. 08. 2019).

[39] *GitHub*. URL: https://github.com (besucht am 13. 04. 2020).

[40] *Electron*. URL: https://electronjs.org/ (besucht am 30. 08. 2019).

ANHANG A

Anhang

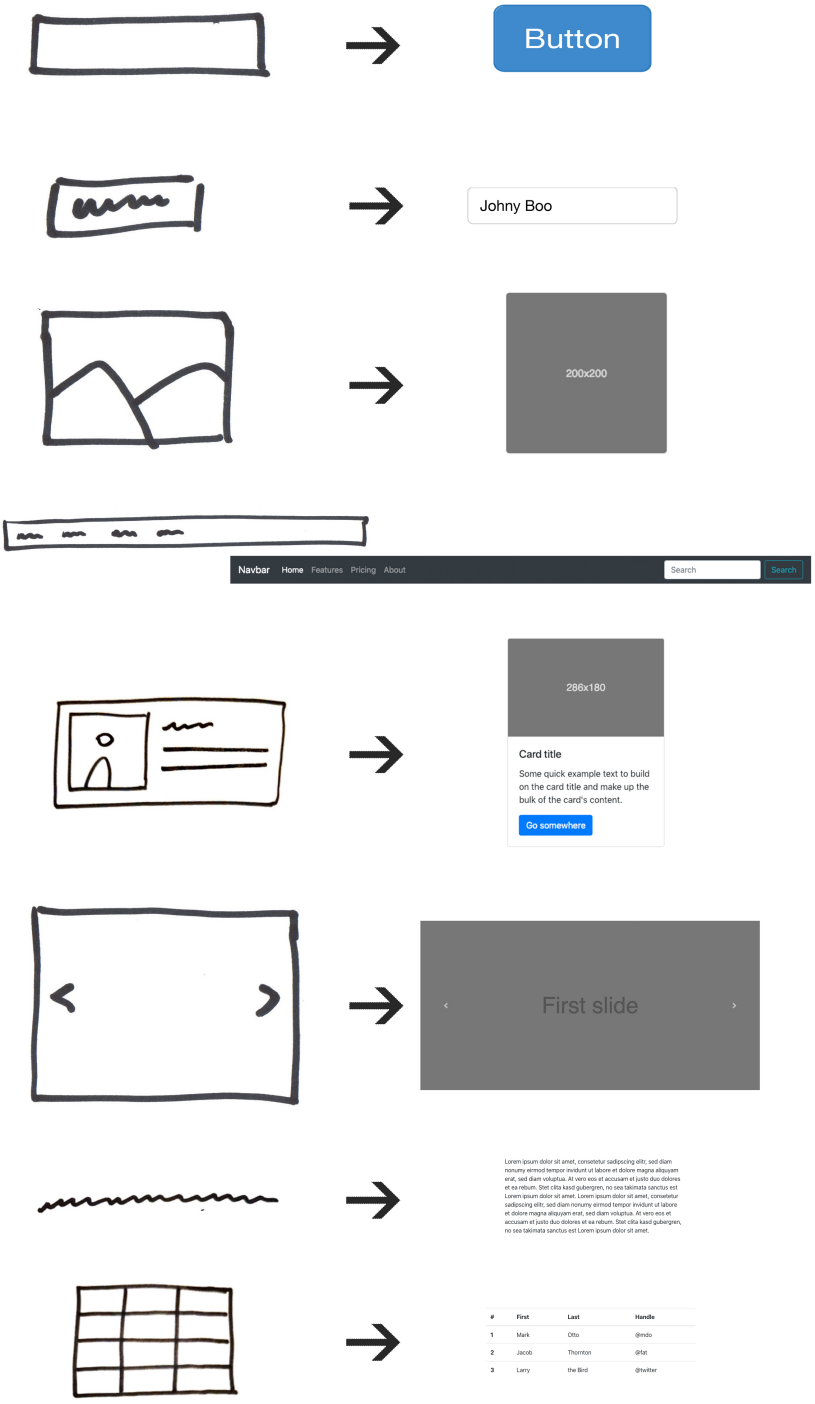


Abbildung A.1: Info Blatt mit den UI Komponenten

Proband

Initialen: MV

Alter: 29

Geschlecht: ☒ M ☐ W

Beruf: Backend Developer

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

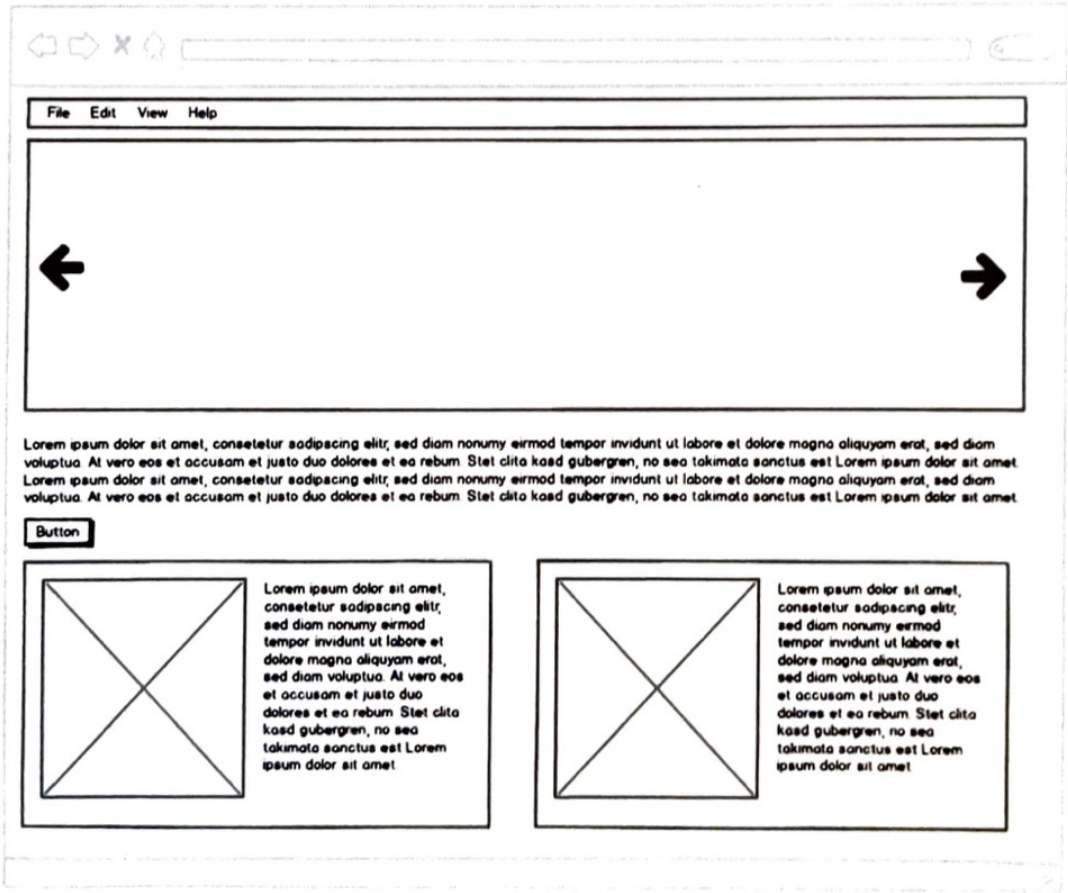


Abbildung A.2: Ausgefüllter Fragebogen – Proband MV, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

variable Länge der Textblöcke fehlte, ansonsten konnten die Designs umgesetzt werden. Intuitiv hätte ich mehrere Zeilen dafür gezeichnet.

Was funktioniert nicht wie erwartet?

Inputfelder werden teilweise als Navbar erkannt

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

Die Aufgabenstellungen konnten wie angegeben umgesetzt werden. Die Ausgabe der App-Skizze sollte jedoch Mobile Komponenten verwenden.

Gibt es Verbesserungsvorschläge?

Abhängigkeit von Umgebungslicht reduzieren oder Softwareseitig tweaken, da das Ergebnis davon beeinflusst wird.

Abbildung A.3: Ausgefüllter Fragebogen – Proband MV, Seite 2

Proband

Initialen: P.R.

Alter: 30

Geschlecht: ☐ M ☒ W

Beruf: Freiberufliche Übersetzerin

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

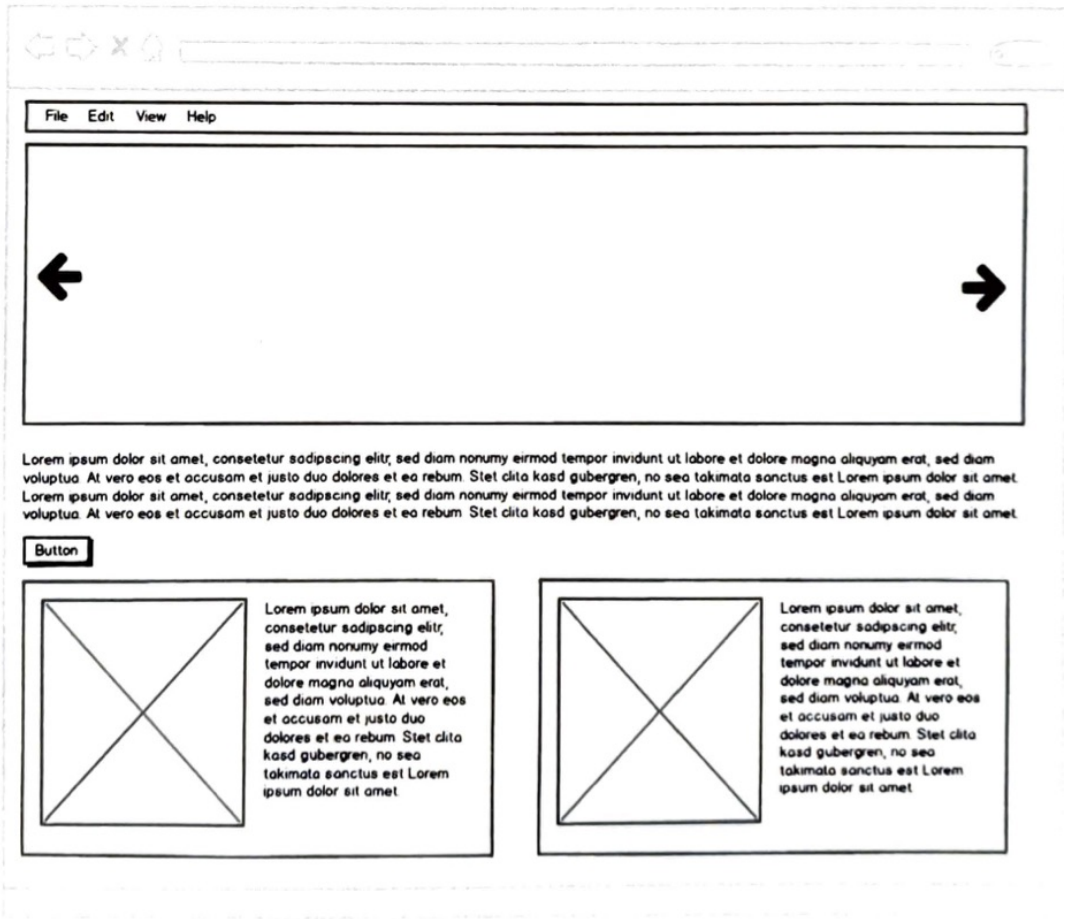


Abbildung A.4: Ausgefüllter Fragebogen – Proband PR, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

Bei den Aufgaben 2, 3 und 4 gab es keine Probleme – das Design entsprach den Zeichnungen. Bei Aufgabe 1 zeigte das Tool anstelle eines Texts einen Button an, weil mein „Text" zu kurz war.

Was funktioniert nicht wie erwartet?

Beim Abstand zwischen den Bildern im Aufgabe 2 war zu gering – könnte sensibler reagieren. Input-Felder wurden oft als Navigationsleisten angezeigt.

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

Ja, die Komponenten waren ausreichend.

Gibt es Verbesserungsvorschläge?

Evtl. Sensibilität bei Abständen erhöhen und die Unterschiede zwischen Input-Feldern und Navigationsleisten sollten markanter sein.

Abbildung A.5: Ausgefüllter Fragebogen – Proband PR, Seite 2

Proband

Initialen: SU

Alter: 26

Geschlecht: ☐ M ☒ W

Beruf: BERATUNG IN ARBEITSSICHERHEIT

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

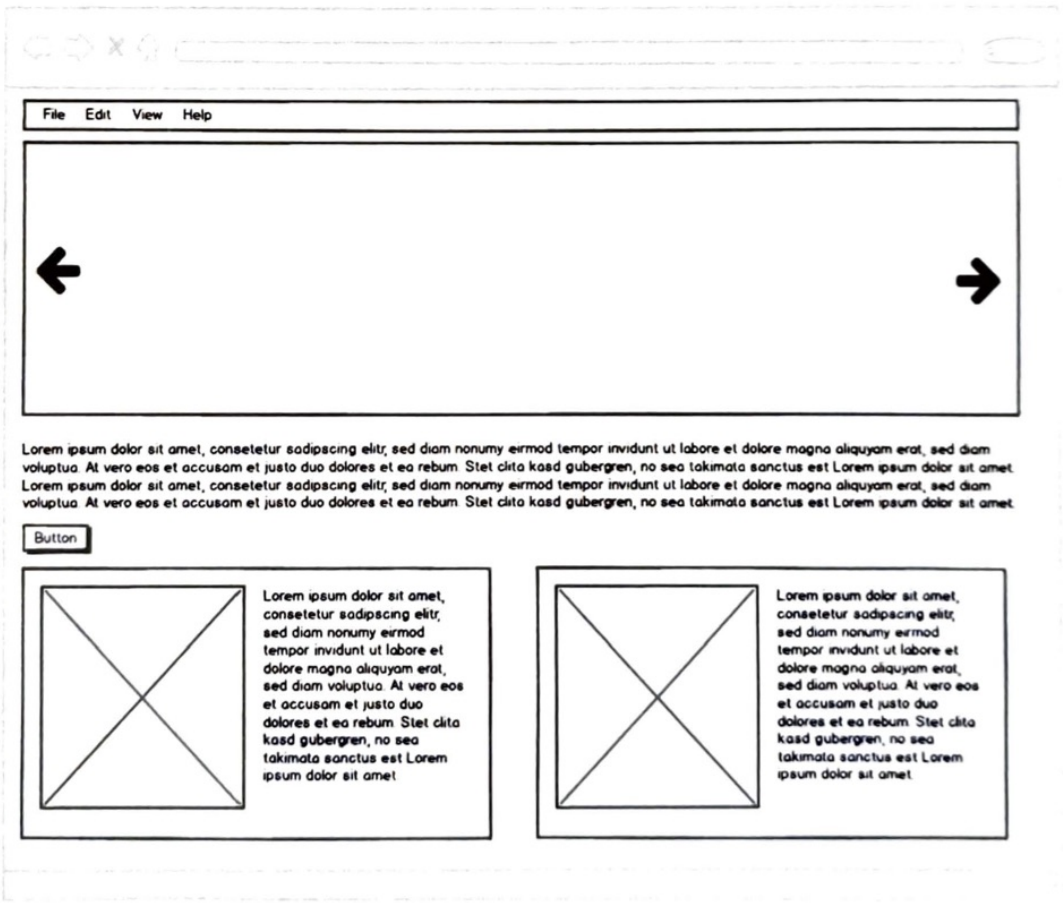


Abbildung A.6: Ausgefüllter Fragebogen – Proband SU, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

Die einzelnen Elemente der Zeichnungen waren teilweise zu nahe und deshalb wurden sie nicht erkannt.

Was funktioniert nicht wie erwartet?

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

Checkbox, Diagramme

Gibt es Verbesserungsvorschläge?

Für kurzen Text eigenes Zeichnung und eigene Zeichnung für langen Text

Abbildung A.7: Ausgefüllter Fragebogen – Proband SU, Seite 2

Proband

Initialen: MM

Alter: 28

Geschlecht: ☒ M ☐ W

Beruf: Informatiker

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

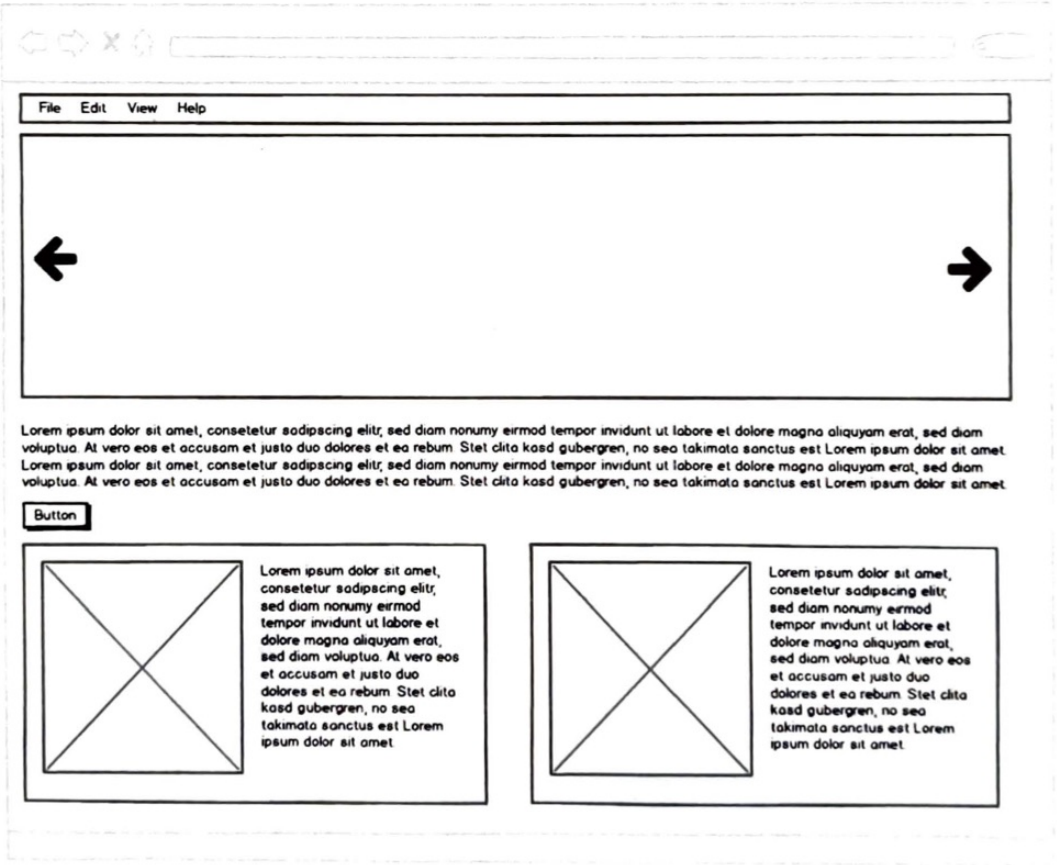


Abbildung A.8: Ausgefüllter Fragebogen – Proband MM, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

Größteils, z.B. wäre es gut wenn Abstände auch im Design berücksichtigt werden.

Was funktioniert nicht wie erwartet?

Navbar und Input werden oft verwechselt

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

Ja, aber ein Passwortfeld wäre noch interessant.

Gibt es Verbesserungsvorschläge?

Die Navbar könnte man ausfüllen von Text mit Kürzeln zeichnen, vielleicht wird das besser erkannt

Abbildung A.9: Ausgefüllter Fragebogen – Proband MM, Seite 2

Proband

Initialen: M M

Alter: 32

Geschlecht: ☒ M ☐ W

Beruf: GRAFIKER

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

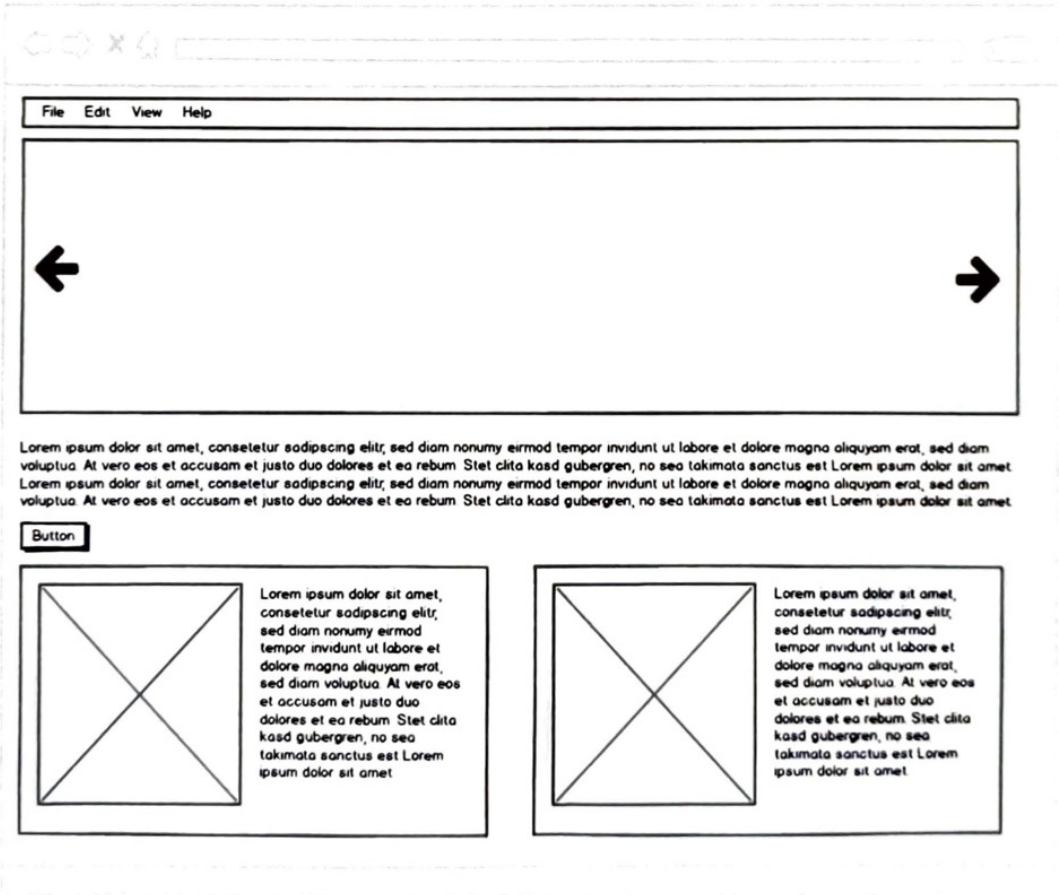


Abbildung A.10: Ausgefüllter Fragebogen – Proband MM2, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen?

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

JA

Was funktioniert nicht wie erwartet?

NAVIBAR UND TEXT WIRD WICH IMMA GRKANNT.
LANGE TEXTZEILEN ERKANNT ES NICHT ALS TEXT.

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?

JA KONNTE UMGESETZT WERDEN

Gibt es Verbesserungsvorschläge?

ALLES NOCH ZIEMLICH ROH = PROTOTYP. SEHR VIEL
AUSBAUFÄHIG = HOHES POTENZIAL
FARBSCHEMAS VORAUSWAHL WÄRE EINE IDEE

Abbildung A.11: Ausgefüllter Fragebogen – Proband MM2, Seite 2

Proband

Initialen: DO

Alter: 32

Geschlecht: ☒ M ☐ W

Beruf: MEDIENDESIGNER

Aufgabenstellung

1) Skizzieren Sie die folgende Webseite nach.

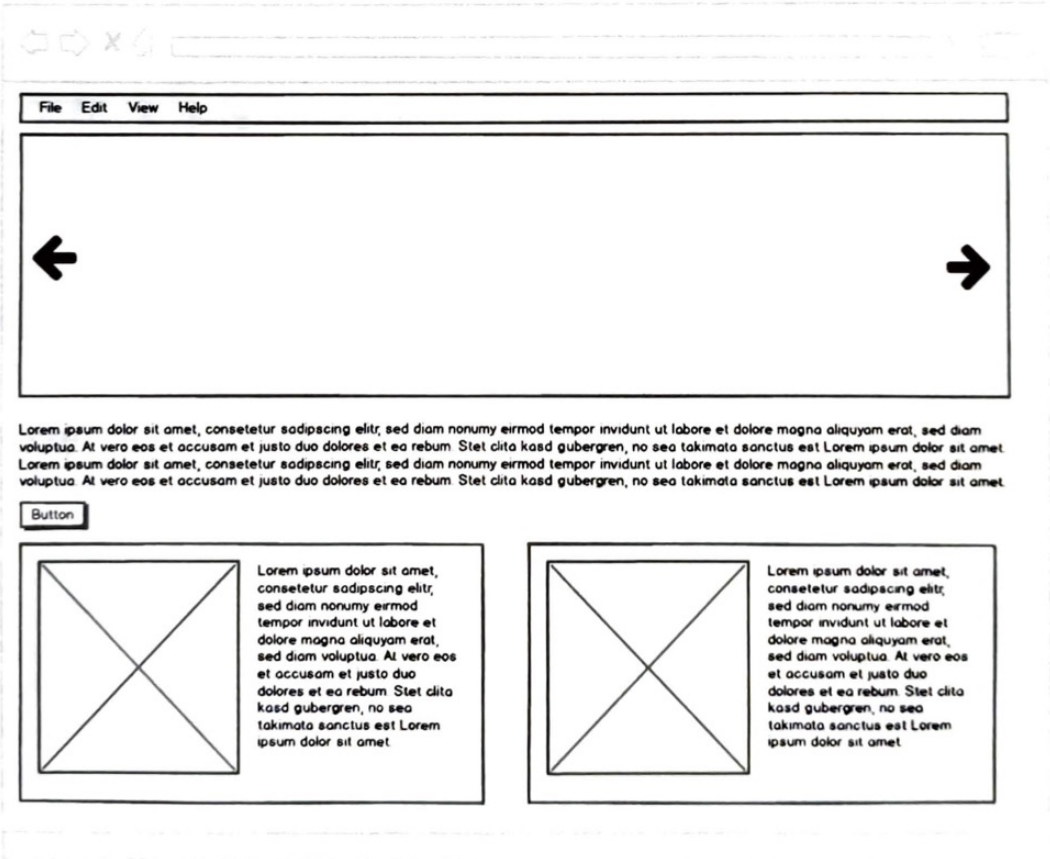


Abbildung A.12: Ausgefüllter Fragebogen – Proband DO, Seite 1

Evaluierung

Wie verwendbar ist das Tool?

1	2	3	4	5	6	7	8	9	10

Wie hilfreich ist das Tool für das Evaluieren von Ideen oder konkreten Problemstellungen? 2 8

1	2	3	4	5	6	7	8	9	10

Entspricht das Design des Tools den Erwartungen aus der Zeichnung?

1	2	3	4	5	6	7	8	9	10

Konnte die Funktionalität aus den Aufgabenstellungen umgesetzt werden?

1	2	3	4	5	6	7	8	9	10

Entspricht das fertige Design den Erwartungen aus der Zeichnung? Wenn nein, was fehlt?

Was funktioniert nicht wie erwartet?
NAVIGATION MIT TEXTFELD WURDE VERTAUSCHT

Konnten die Aufgabenstellungen mit den zur Verfügung stehenden Komponenten umgesetzt werden? Wenn nein, welche Komponenten sollten ergänzt werden?
JA

Gibt es Verbesserungsvorschläge?
EVALUIEREN VON IDEEN SEHR SINNVOLL, FÜR PROBLEMSTELLUNGEN EHER NICHT

Abbildung A.13: Ausgefüllter Fragebogen – Proband DO, Seite 2

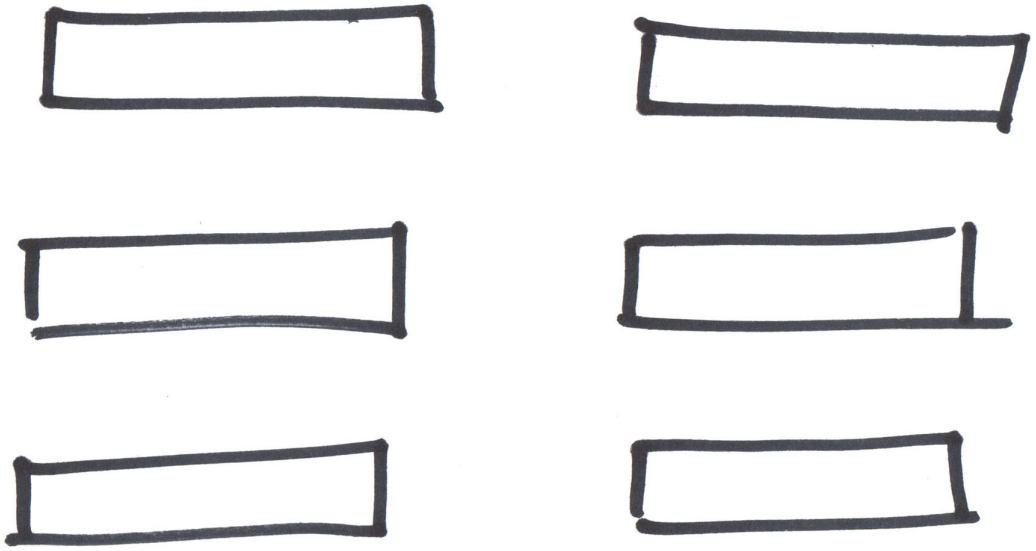


Abbildung A.14: Trainingsdaten – Button

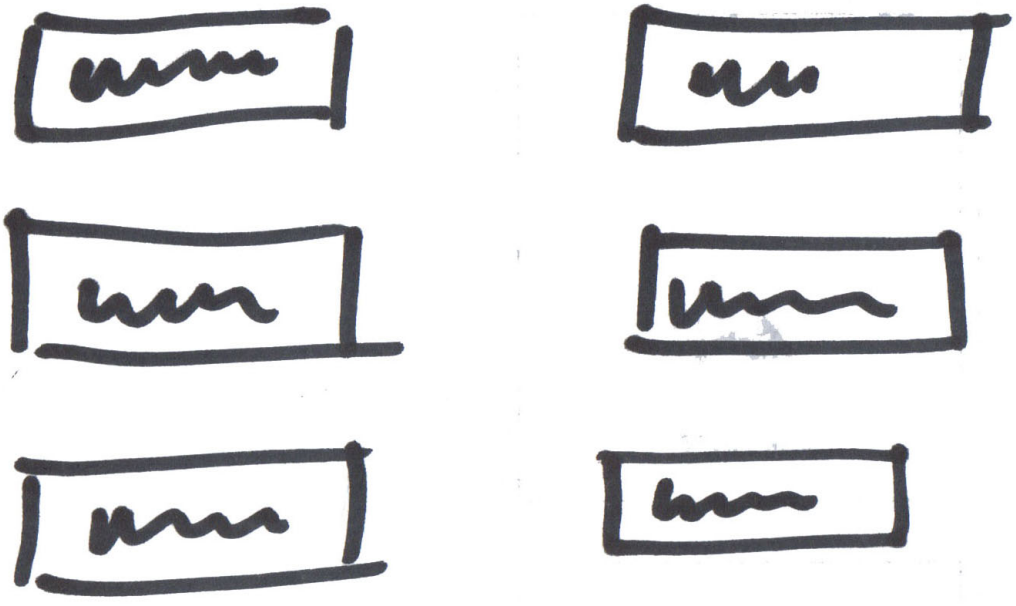


Abbildung A.15: Trainingsdaten – Input

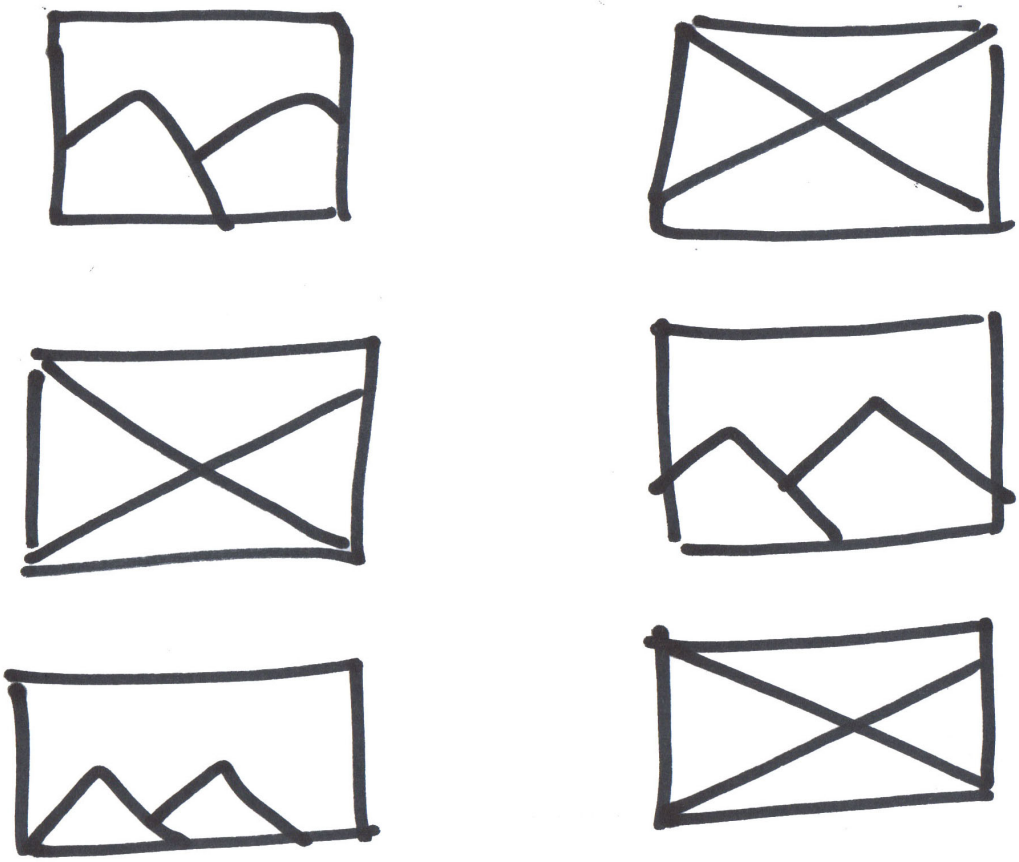


Abbildung A.16: Trainingsdaten – Image

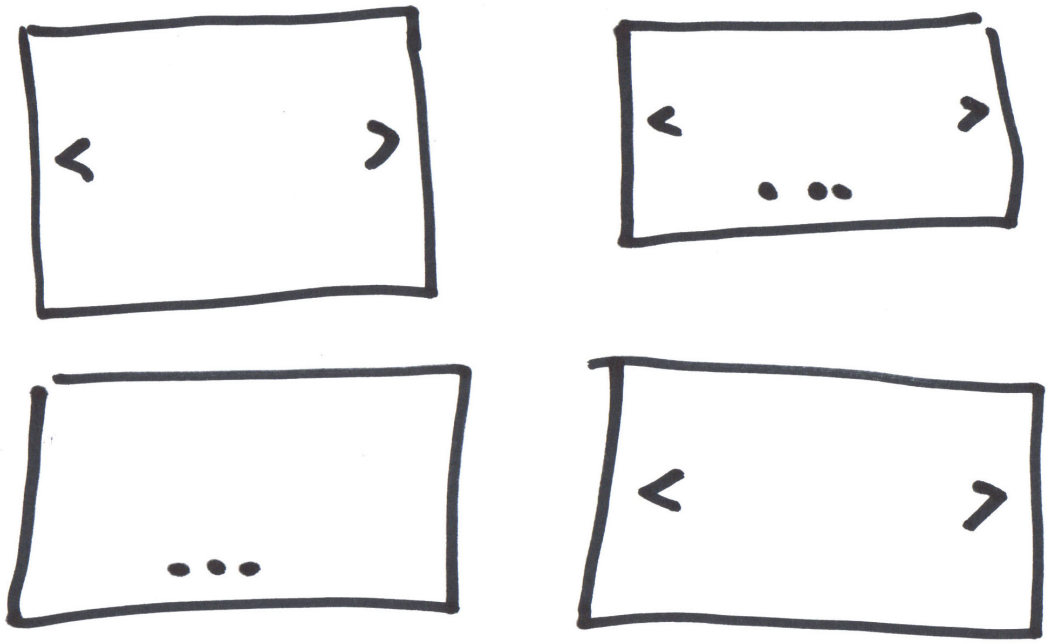


Abbildung A.17: Trainingsdaten – Carousel

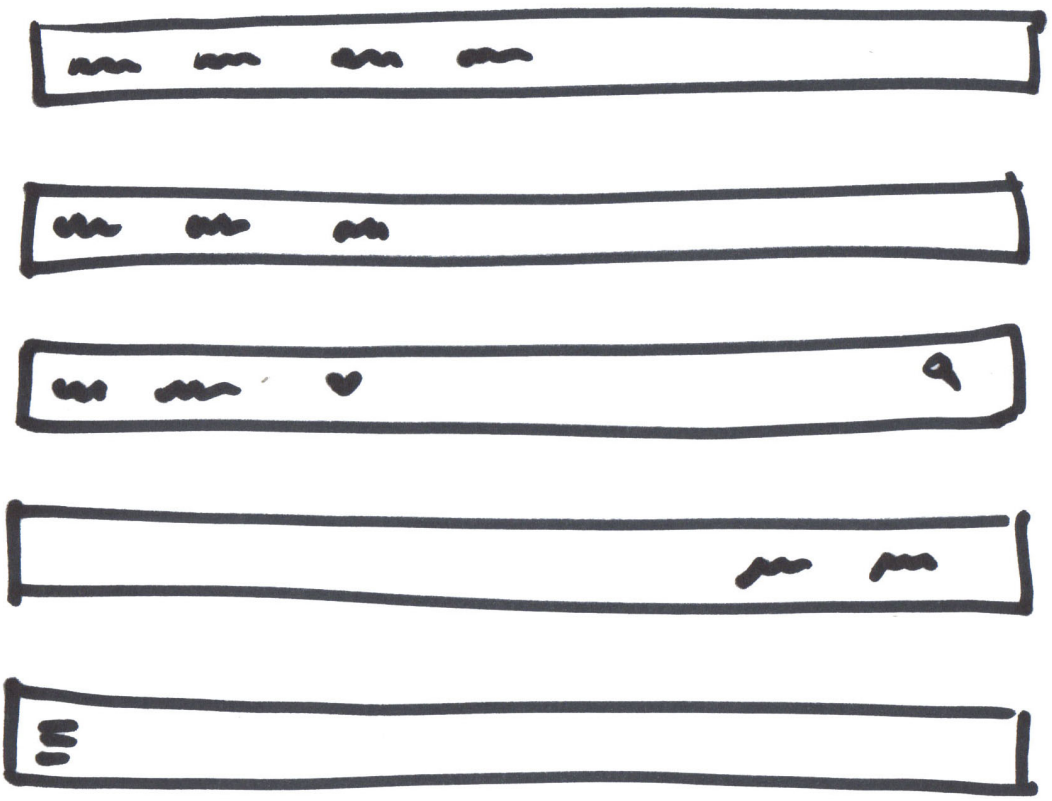


Abbildung A.18: Trainingsdaten – Navbar

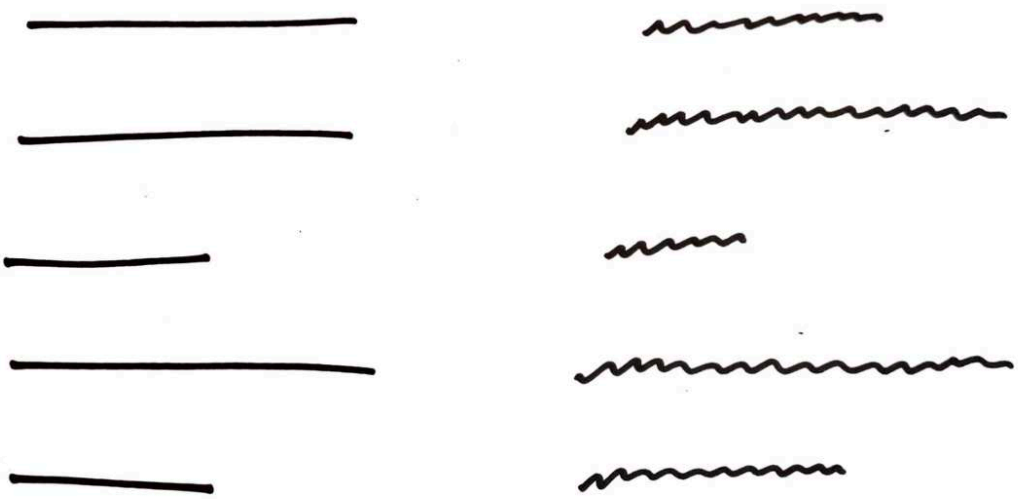


Abbildung A.19: Trainingsdaten – Text

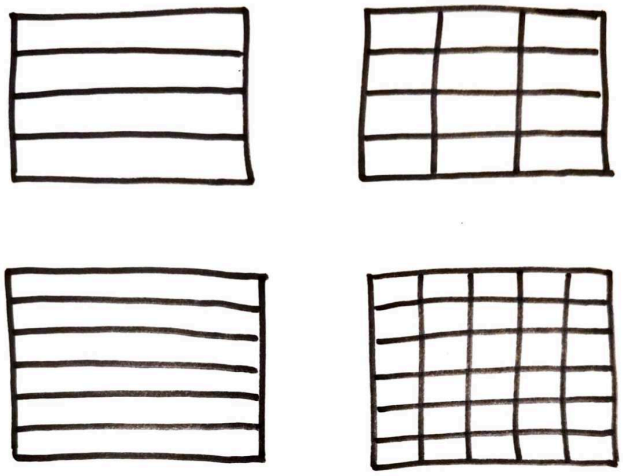


Abbildung A.20: Trainingsdaten – Table

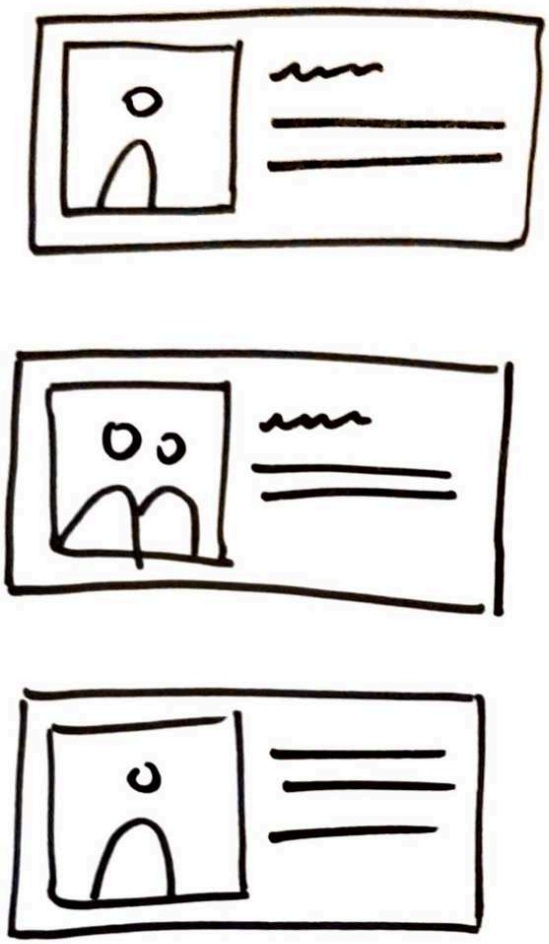


Abbildung A.21: Trainingsdaten – Info Card