Mathematisch-
Naturwissenschaftliche
Fakultät

Human-Computer Interaction

Bachelorarbeit

Erstellung und Bewertung eines Webbrowser-basierten Frameworks zur Datenakquise für Studien

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Human-Computer Interaction
Oliwia Oles, oliwia.oles@student.uni-tuebingen.de, 2020

Bearbeitungszeitraum: 15.08.2020 - 14.12.2020

Betreuer/Gutachter: Prof. Dr. Enkelejda Kasneci, Universität Tübingen
Zweitgutachter: Dr. Wolfgang Fuhl

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Oliwia Oles (Matrikelnummer 4186564), December 9, 2020

Abstract

Diese Arbeit beschäftigt sich mit der Ertsellung und Bewertung eines Frameworks zur Durchführung und Datenerfassung von Studien, wie beispielswweise eine Blickpostionsvorhersage oder Emotionserkennung. Der Eyetracker basiert auf der webgazer.js Biblitohek der Brown Universität und die Emotionserkennung beruht auf einer FrontEnd-Emotionserkennungs Bibliothek auf Grundlage von der Google Shape Detection API. Beide Bibiliotheken, wurden implementiert, um sie für Studien zu gebrauchen. Zusätzlich zu diesen Daten werden auch noch Angaben wie das Alter und das Geschlecht, jedoch nur auf freiwilliger Basis, erfasst, um mehr Aussagekraft zu erhalten. Anschließend wird das Framework getestet und bewertet, um zu erfahren, ob es genauso gut in der Praxis wie in der Theorie funktioniert, da es sich hierbei um komplett clientseitige Anwendungen handelt. Im Normalfall benötigen solche Anwendungen nämlich noch zusätzliche Hardware, um zu funktionieren. Die verwendeten Biblitoheken kommen jedoch ohne diese aus, fordern nur eine funktionierende Webcam und bestehen nur auf JavaScript Code. Deshalb stellt sich hier nun die Frage, ob eine rein clientseitige Anwendung vergleichbar gut mit anderen Anwednungen in diesen Bereichen funktioniert und ob sich die Daten gut erfassen lassen.

Contents

1. Einleitung **9**

2. Grundlagen **13**
2.1. HTML 13
2.2. JavaScript 15
2.3. AJAX 16
2.4. Eyetracking via WebGazer 17
2.5. Emotionserkennung 18

3. Methoden **21**
3.1. Übersicht 21
3.2. Tutorial 31
3.3. Kommunikation mit dem Server 32
3.4. Funktionen und GUI 33
3.4.1. start.js 33
3.4.2. index.html 35
3.4.3. trackerLib.js 36
3.4.4. eyetracker.html 38
3.4.5. emotiondetection.html 39
3.4.6. webgazer.js 40
3.4.7. delete.html und showData.php 40
3.4.8. Zusammenhang zwischen den Dokumenten 41
3.5. Server und Datenbank 42

4. Evaluation **45**
4.1. Eyetracker 45
4.2. Emotionserkennung 46
4.3. Laufzeit 47

5. Diskussion **49**

A. Zusammenfassung **51**

1. Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Beschreibung und Bewertung eines webbrowserbasierten Frameworks zur Datenakquise und Durchführung von Studien [PN09]. Im Besonderen beschäftigt sich diese Ausarbeitung vor allem mit der Erfassung von Blickpositionen [Fuh19, FRE20, FSK+18, FGK20a] und der Erkennung von Emotionen [GG15, FGRK19, FRK19, FCZ+18]. Ziel ist es, eine funktionierende Webplattform zu erstellen, die es ermöglicht, diese Daten für verschiedene Probanden zu speichern und auszuwerten.
Im Rahmen dieser Bachelorarbeit wurde zum einen eine Blickpunktpositions- und zum anderen eine Emotionserkennungs-Bibliothek implementiert, um Studien dazu durchzuführen und die erfassten Daten speichern zu können. Zudem wird zu Anfang dem Nutzer die Möglichkeit gegeben, noch einige persönliche Informationen anzugeben, die jedoch völlig freiwillig sind und nur zu Studienzwecken dienen.
Bereits im späten 19. Jahrhundert gab es erste, mechanische Eyetracker. Der Kopf der Person musste stillgehalten und das Auge anästhesiert werden. Dies war verständlicherweise nicht so bequem wie der Eyetracker, der hier verwendet wird und rein clientseitig aufgebaut ist. Erst Anfang des 20. Jahrhunderts wurde damit begonnen Fotografien von Lichtreflektionen im Auge aufzunehmen,wohingegen sich in den 50er Jahren einige verschiedene Methoden für Eyetracking etablierten [HNA+11]. Heute wird dies jedoch über das bildbasierte Eye Tracking gelöst [FSR+16, FKH+17, FGS+18, FGK20b, FSK17b].
Für die Blickpositionserkennung in diesem Framework wurde die webgazer.js-Bibliothek der Brown Universität verwendet, diese dient einer Blickpunktvorhersage im Webbrowser und erfordert keine zusätzliche Hardware, sodass sie auf jedem Computer verwendet werden kann. Diese Form des Eye Trackings wird auch Remote Eye Tracking genannt [FSG+16, FSG+17, FSK17a, FEH+18]
Für die Emptionserkennung wurde die FrontEnd-Emotiondetection-Bibliothek von Kevin Hsiao auf github verwendet, die ebenfalls keine zusätzliche Hardware benötigt. Diese Emotionserkennung basiert auf neuronalen Netzen [FKRK20, **?**, FK20b, FK20a].
Der Vorteil davon, diese beiden Bibliotheken in einem Framework zu haben ist, dass die Daten direkt gespeichert und aufgerufen werden können, ohne auf Dritte angewiesen zu sein, die diese Daten bei sich speichern. Sollte etwas mal nicht funktionieren, kann der Fehler leichter gefunden und behoben werden, statt eventuell lange darauf zu warten, dass ein anderer sich darum kümmert. Außerdem werden noch zusätzliche Daten gespeichert, wie beispielsweise das Alter und das Geschlecht, die es ermöglichen genauere Statistiken aufzustellen, anstelle der Speicherung von nur vielen Blickpunkten, die nie auf einen Proband zurückgeführt werden kön-

nten. Natürlich fallen solche Daten unter den Datenschutz [FBK20], weshalb der Anwender einwilligen muss, bevor die Software verwendet werden kann.

Mit diesem Framework ist es möglich, Studien durchzuführen und auf Drittanbieter zu verzichten, die Daten speichern. Auf diesem Weeg können die Daten direkt gespeichert werden, ohne über Dritte auf sie zugreifen zu müssen. Die Daten werden auf einem Webserver gespeichert, den Roufayda Salaheddine im Rahmen ihrer Bachelorarbeit "Erstellung und Bewertung einer Webplattformbasierten Datenhaltungs- und Bewertungssoftware für Studien" aufgesetzt hat. Dort werden die Ergebnisse der Blickposition und die Ergebnisse der Emotionserkennung gesammelt, welche später nach Belieben ausgewertet werden können, um beispielsweise die Genauigkeit der Emotionserkennung zu bestimmen.
Doch wofür wird so eine Blickpunktvorhersage eigentlich überhaupt verwendet? Damit kann erfasst werden, wo der User öfter hinschaut, was seinen Blick mehr anzieht und was seltener angeschaut wird. Dadurch können auch die idealen Punkte herausgelesen werden, wo es sich am meisten lohnt, Werbung zu platzieren und an welchen Stellen Werbung eher erfolgslos wäre. Vor allem der Marketing Bereich profitiert davon, wie reagieren Nutzer auf Produkte, was zieht den Blick in einer Werbung am meisten an. Jedoch wird Eyetracking auch im Gesundheitswesen verwendet. Beispielsweise besteht die Möglichkeit, über einen Eyetracker einen Computer zu bedienen, wenn es dem Patienten nicht oder kaum möglich ist, sich zu bewegen. Das Einsatzgebiet ist also nicht gerade klein und erweitert sich stetig.[Ebe18]
Genauso hat auch die Emotionserkennung einen größeren Nutzen als vielleicht am Anfang erwartet wird. Damit kann beispielsweise ermittelt werden, wie verschiedene Kunden auf bestimmte Werbung und Produkte reagieren und diese somit dann eventuell anpassen, um die gewünschte Reaktion auf ein Produkt von einem Kunden zu erzielen. [Sti20, Fuh20] Zudem ist es auch das Ziel, eine bessere Mensch-Computer Intelligenz Interaktion zu erreichen, das heißt, die Interaktion zwischen Mensch und Computer soll sich der Interaktion zwischen Mensch und Mensch annähern. Dafür müssen Emotionen, Tonfall und Gestik interpretiert werden können, um dies zu erreichen. Kann der Computer echte Emotionen besser interpretieren, so ist das Ziel einen kleinen Schritt näher gerückt. [SSLG04]

Dieses Framework hat jedoch nicht das Ziel, herauszufinden, wo Werbung am salientesten platziert werden kann oder wie ein User auf Profukte reagiert, hier soll nur generell der Blickpunkt und die Emotion erfasst werden [FKB+18, FKS+15, GFSK17, FKSK18]. Natürlich gibt es auch Alternativen zu diesem Framework, die im Internet zu finden sind. Schon allein die wbgazer.js-Biblitohek wird sehr häufig im Internet verwendet. Die Brown Universität hat auch eigene Beispiele bereitgestellt, wie die webgazerbib verwendet werden kann: Zum Beispiel nur zur Blickpositionerkennung, so wie es auch hier eingesetzt wird oder um ein Ballkollisionspiel zu spielen, wo der Ball nur mit dem eigenen Blick bewegt wird. Außerdem gibt es die Blickpositionerkennung im Webbrowser auch von anderen Anbietern mit anderen Bibliotheken, beispielsweise gibt es da die Alternative hawkeye 2.0 die eine

Blickpositonerkennung ohne zusätzliche Hardware für iPhone und iPad anbietet. Genauso wie bei der Blickposition gibt es auch für die Emotionserkennung diverese Bibliotheken und funktionierende Alternativen. Hier fängt es schon damit an, dass die Gesichter und Emotionen entweder in einem Live-Video erkannt werden oder auf, von einem User hochgeladenen, Bildern. Zudem gibt es auch diverese Alternativen zu der FrontEnd - Emotiodetection-Bibliothek, die in diesem Framework verwendet wurde. [pro]

2. Grundlagen

Im Web gibt es drei Standard-Technologien: HTML, CSS und JavaScript. Sobald eine nicht-statische Webseite erstellt werden soll, wird in den meisten Fällen JavaScript verwendet. Deshalb wurde dieses webbbasierte Framework auf Grundlage von HTML und JavaScript inklusive Ajax zur Kommunikation mit dem Server entwickelt. HTML wird im Browser verwendet, um Webseiten zu strukturieren und Layouten, CSS dient dazu, Stilregeln für die Webseite zu definieren und diese ansprechend für den User zu gestalten. JavaScript liefert Funktionen, um dynamische Updates und viele andere Funktionen zu realisieren. Es bietet außerdem alle Funktionalitäten, die benötigt werden, um sowohl den Eyetracker als auch die Emotiondetection zum Laufen zu bringen. Im Folgenden werden sowohl die Programmiersprachen, als auch die beiden möglichen Beipsiele zur Datenerfassung erläutert.

2.1. HTML

HTML wurde im Jahre 1992 vom World Wide Web Consortium (W3C) veröffentlicht und dient seitdem als Grundbaustein bei der Entwicklung von Webseiten. Die aktuelle Version, HTML 5.2, die 2017 veröffentlicht wurde, wird bereits von vielen Webbroswern untertsützt.
Die Grundidee von HTML war es damals, Dokumente schnell und einfach zwischen Personen auf elektronischem Wege zu übertragen. Begonnen hat das Projekt 1989 damit, dass der Wunsch aufkam, Forschungsergebnisse mit anderen Mitarbeitern der Europäischen Organisation für Kernforschung (CERN) zu teilen und von den beiden Standorten in Frankreich und in der Schweiz aus zugänglich zu machen. Drei Jahre später erfolgte die Veröffentlichung von HTML. [wik20c]

Über HTML

HTML, kurz für HyperText Markup Language, ist eine textbasierte Auszeichnungssprache, die den strukturellen Aufbau einer Webseite bzw. eines elektronischen Dokuments bestimmt und den Inhalt dieser beschreibt und definiert. HTML setzt sich aus den beiden Bausteinen HyperText und Markup zusammen:

- ***HyperText***
  Hiermit sind die (Hyper-)links gemeint, die Webseiten/HTML-Dokumente

miteinander in Verbindungen setzen und aufeinander verweisen, sei es nun innerhalb einer Seite oder zwischen mehreren Webseiten im Netz.

- ***Markup***
  Dies bezieht sich darauf, dass verschiedene Textteile ausgezeichnet werden, um dem ganzen eine Struktur zu geben. Die Auszeichnung erfolgt durch verschiedene Tags<>, die den Inhalt für die Anzeige im Netz kommentieren und aus verschiedenen Elementen wie <title>, <video>, <p>, etc. bestehen. Diese Tags stellen entweder Inhalte dar, die für den Nutzer sichtbar sind, wie <p>, das einen Pragraphen definiert, der Text enthät oder nicht-sichtbare Inhalte, Metainformationen, wie bspw. <meta>, die Informationen über Autor, Sprache, usw. enthalten. Die Tags sind nicht case-sensitive, das heißt, dass die Groß- und Kleinschreibung der Tags keinerlei Rolle spielt.

Die Sprache dient jedoch nicht dazu, den Inhalt zu formatieren, dies ist die Aufgabe von CSS. HTML zielt stattdessen vor allem darauf ab, den Inhalt einem User zur Verfügung zu stellen.
HTML unterscheidet sich von den gängigen Programmiersprachen insofern, dass es sich hierbei um eine AUszeichungssprache handelt, die dazu dient, ein Dokument zu strukturieren anstatt ein Programm zu schreiben. Eine Gemeinsamkeit jedoch ist, dass für die meisten Programmiersprachen und Auszeichnungssprachen ein einfacher Texteditor reicht, um die Quelldokumente zu bearbeiten.
Ein HTML Dokument besteht aus 3 Teilen: Die Dokumenttypdeklaration, Head und Body. In der Dokumenttypdeklaration wird festgelegt um welche HTML Version es sich handelt, z.b. HTML 5.
Im Head werden vor allem Informationen gespeichert, die der User später nicht sieht, wie bspw. Metainformationen oder eingebundene externe Dateien.
Im Body befindet sich der Inhalt, der für den Nutzer sichtbar ist, der also im Anzeigebereich des Browsers zu sehen ist. Dies können sowohl ein Bild, als auch Hyperlinks, Texte usw. sein. [htm] [Con05a]

HTML Erweiterung

Da HTML nur dazu dient, den strukturellen Aufbau einer Seite zu beschreiben, jedoch weder die Präsenation dieser noch Funktionalitäten beeinflusst, gibt es verschiedene Techniken die dies tun.
Um das Aussehen einer Webseite zu ändern, werden sogenannte Cascading Styling Sheets verwendet, darin wird das Aussehen von Text, Seite, etc. definiert. In HTML gibt es zwar Tags, die das Aussehen zu einem kleinen Teil beeinflussen können, doch das Ziel ist es, die Präsentation eines HTML-Dokuments gänzlich seperat in einem CSS-Dokument zu beschreiben.
Sollen dem Dokument verschiedene Funktionen hinzugefügt werden, wie bsp. eine Live-Uhr, das aktuelle Datum, etc., so wird dafür eine Skriptsprache verwendet, in

aller Regel handelt es sich dabei um JavaScript. Sobald JS-Funktionen verwendet werden, spricht man von dynamischen HTML.
Sowohl CSS-Dateien, als auch JS-Dateien werden im Header des HTML-Dokuments über entsprechende Tags eingebunden. [wik20c]

2.2. JavaScript

JavaScript, im Folgenden JS, ist eine Skriptsprache, die Mitte der Neunziger, genauer genommen im Jahre 1995 entwickelt wurde, um die Möglichkeiten von HTML und CSS zu erweitern und einen größeren Funktionsumfang zu bieten.
JS wurde früher primär clientseitig für die Entwicklung dynamischer Webseiten eingesetzt, jedoch kann es mittlerweile auch mit dem Framework Node.js serverseitig eingesetzt werden. In dieser Arbeit wurde JS allerdings nur clientseitig verwendet, daher wird sich dieser Abschnitt auch nur mit der clientseitigen Verwendung von JS beschäftigen. [Con05b]
Die Skriptsprache basiert auf dem ECMAScript, das von allen gängigen Browsern mindestens ECMAScript 3, meist aber eher ECMAScript 5.1 unterstützen. ECMAScript ist selbst eine Skriptsprache, die als Standard für Sprachen wie JS dient und immer weiter entwickelt wird. Diese Skriptsprache ist eine general-purpose programming language, das heißt sie ist für viele Anwendungsfälle geeignet, in den meisten Fällen wird sie in Webbrowsern integriert, jedoch wird diese auch immer öfter serverseitig oder in eingebetteten Systemen eingestezt. [ecm]
Jedoch hieß JS nicht immer so, die Sprache wurde erst nach der Veröffentlichung nochmals umbenannt. Im selben Jahr, in dem JS veröffentlich wurde, kam auch eine andere Programmiersprache auf den Markt, Java. JS hieß währenddessen noch LiveScript. Die beiden Sprachen sind in den meisten Aspekten nicht und sind grundlegend verschieden. Dennoch wurde LiveScript in JavaScript umbenannt, nachdem Java so ein großer Erfolg war und die beiden Sprachen sich in Ausdruckssyntax, den Namenskonventionen und den elementaren Kontrollstrukturen gleichen. [wik20d]

Über JavaScript

JS ist eine objektorientierte Programmiersprache, die, im Gegensatz zu Java, nicht zwischen veschiedenen Typen von Objekten unterscheidet. Während bei Java die Vererbung strikt über die Klassenhierarchie erfolgt, wird in JS über den Prototypen-Mechanismus vererbt, das heißt, dass die Vererbung dynamisch vonstattengeht, es kann also variieren, was vererbt wird und Objekte können dynamisch und Methoden und Eigenschaften erweitert werden.
Ein weiterer großer Unterschied zu Java ist, dass JS eine eher systaktisch freie Sprache ist. Das heißt, dass Variablen, Klassen und Methoden nicht deklariert werden müssen, zudem muss auch nicht darauf geachtet werden, ob diese privat, öffentlich oder

geschützt sind. Zudem müssen Variablen, Parameter und Rückgabewerte nicht typisiert werden, Java hingegen ist eine auf schnelle Ausführung und strenge Typsicherheit ausgelegte, klassenbasierte Programmiersprache. Dieser Unterschied macht das Programmieren mit JS deutlich einfacher als das Programmieren mit Java. Da JS zur dynamischen Webprogrammierung verwwendet wird, benötigt je nach Komplexität der Webseite verschiedene Bibliotheken und Frameworks, die die Sprache um verschiedene Funktionen erweitern, um dem Programmierer mehr Möglichkeiten beim Erstellen der Seite dienen.
Typische JS Anwendungsgebiete sind unter anderem die Manipulation von Webseiten über DOM, die Prüfung von Daten bei der Formulareingabe bevor diese an den Server geschickt werden, um fehlende und fehlerhafte Eingaben zu vermeiden, das Senden und Empfangen von Daten über Ajax, die Verschleierung vom E-Mail-Adressen und noch viele mehr.
Jedoch gibt es auch JS-Anwendungen, die zum Missbrauch verwendet werden und gegen den Wunsch des Nutzer sprechen oder gegen das Principle of Least Surprise verstoßen, das heißt, dass diese Anwendungen den Nutzer überraschen, was so nicht passieren sollte. Daher gibt es aber in Browsern verschiedene Funktionen, um gegen eventuellen Missbrauch vorzugehen, z.B. das Deaktivieren des Kontextmenüs oder der Kopierfunktion, das Verbieten von Pop-Up-Fenstern, die den User behindern und noch viele andere.
Ein großes Manko, was die Sprache mit sich bringt, ist die Tatsache, dass JS in Browsern beliebig ein- und ausgeschlatet werden kann, sodass Webseiten nicht immer (gleich) funktionieren, wie der Entwickler es erwartet und wünscht. [Con19]

2.3. AJAX

AJAX steht für **A**synchronous **J**avaScript **a**nd **X**ML und dient der asynchronen Datenübertragen zwischen einem Broswer und einem Server.[w3ab] Ganz allgemein gesagt ist AJAX dafür gedacht, die Kommunikation zwischen Client und Server zu ermöglichen, wobei clientseitig JavaScript verwendet wird. Serverseitig wurde für den Server die Programmiersprache PHP eingesetzt. AJAX ist eine Technik, um Daten an einen Server zu senden oder von einem Server einzuholen, diese ist jedoch keine eigene Programmiersprache. Der größte Nutzen von AJAX liegt darin, dass es dadurch möglich ist, Teile einer Webseite zu aktualisieren ohne die ganze Seite neu laden zu müssen, daher ist es asynchron. Dies führt dazu, dass die Wartezeit um einiges kürzer ist als bei statischen Webseiten, die das Neuladen des kompletten Dokuments erfordern. Die verkürzte Wartezeit führt zu einer verbesserten User-Experience, da ein Nutzer immer sehr kurze bis gar keine Wartezeit erwartet, weil sonst das Interesse verloren geht. AJAX setzt sich aus folgenden Bausteinen zusammen: JavaScript, HTML DOM und XMLHttpRequest-Objekt.
Um die Datenübertragung zu starten, wird zuerst ein XMLHttpRequest-Objekt erstellt mit *new XMLHttpReuest()*. [w3aa] Mit .open() wird die Methode(POST, GET, ...) und Ziel-URL, dabei handelt es sich im Normalfall um eine PHP-Datei, gesetzt

und danach ein Request per GET oder POST an den Server gesendet: XMLHttpRequest().send(). Dieser wird dort verarbeitet, folglich wird eine Antwort (Response) erstellt und an den Clienten zurückgeschickt. JavaScript liest die Antwort und führt die entsprechende Aktion aus, beispielsweise das Anzeigen einer Information aus einer Datenbank auf einer Webseite, ohne dass der Nutzer die Seite neu laden muss. [sel20] [wik20a]

2.4. Eyetracking via WebGazer

Da dieses Projekt das Ziel hat, Daten zu erfassen, wurde der Eyetracker als Beispiel möglicher Daten verwendet, die gesammelt werden können [FCK18a, FCK18b, FK18, FBH+19, FCK+19]. Der Eyetracker basiert auf der WebGazer.js Bibliothek, die seine Ursprünge in der Brown Universität hat und seitdem konstant weiterentwickelt wird.
Der Eyetracker kann nur lokal oder mit einer https-Verbindung verwendet werden, deshalb gibt es auch eine entsprechende Meldung dazu sobald dieser aufgerufen wird(Figure 2.1.).

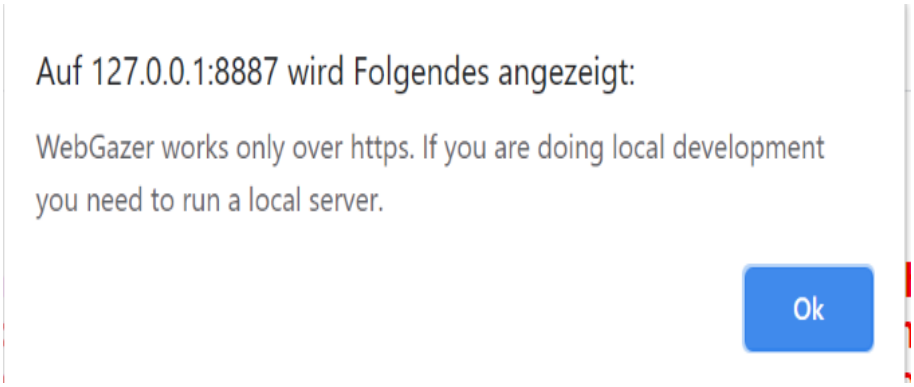

Figure 2.1.: Fehlermeldung der Eyetracker Software, da diese https für die Serverkommunikation benötigt. Dies wird erzwungen, da es sich um persönliche Daten handelt.

Die Bibliothek verwendet die Webcam eines Nutzers, um die Blickposition eines Webseitenbesuchers in Echtzeit zu erfassen und vorherzusagen. Das Modell kalibiriert sich von selbst, sobald ein Nutzer mit einer Webseite interagiert, die WebGazer nutzt. Die Bibliothek ist ausschließlich mit JavaScript geschrieben, sodass diese problemlos nur clientseitig verwendet werden kann. Natürlich muss erst die Erlaubnis zur Verwendung der Webcam gegeben werden, bevor das Modell die Daten sammeln kann.
Frühere Studien haben ergeben, dass die Blickposition eines Nutzers und die Mausposition auf dem Bildschirm in einem Zusammenhgang stehen. Laut diesen Studien,

weicht die Blickpositon meist nur wenig von der Mausposition ab und folgt dem Mauszeiger meist leicht verzögert. Aufgrund dieser Ergebnisse, wurde dies auch mehr oder weniger für die Bibliothek verwendet. Die Blickpositionsvorhersage erfasst die Pupillen des Users und die Positon des Mauszeigers und berechnet daraus einen Punkt, den der User anschaut, der meist in der Nähe des Mauszeigers liegen sollte. [PSL+16, FK19]
Diese Bibiliothek ist ein Open Source Projekt und steht somit jedem zur Verfügung und kann ganz leicht mit wenig Code in eine Webseite integriert werden. Es besteht die Möglichkeit, eine der beispielhaften HTML-Dokumente zu verwenden, die mit dem Webgazer-Projekt geliefert werden oder ein eigenes Dokument zu schreiben, wie es für dieses Projekt getan wurde.
Die Bibliothek erfordert keine besondere bzw. zusätzliche Hardware, eine normale Webcam ist schon ausreichend und läuft auf den meisten verbreiteten Browsern wie Google Chrome, Safari, Mozilla Firefox, Opera und Microsoft Edge. Die Bibliothek bietet verschiedene Prediction-Modelle und viele verschiedene zusätzliche Funktionen. Beispielweise kann dem User im Videofeedback ein Kasten angezeigt werden, der dazu dient, dem User mitzuteilen, ob dieser von der Kamera gut oder schlecht erkannt wird.
Sobald der WebGazer integriert ist, wird das WebGazer-Ojekt in das globale Namespace eingeführt. Die Bibliothek bietet viele verschiedene Methoden zur Steuerung des Webgazers(beispielsweise um dieses zu starten mit *webgazer.begin()*), um Callbacks hinzuzuüfgen oder um Module austauschen zu können. Es gibt auch mehr als eine Möglichkeit, um die Blickposition zu erfassen, entweder wird durch ein Callback alle paar Millisekunden die Blickposition erfasst oder die Blickposition wird nur dann erfasst, wenn diese gebraucht wird.
Grundsätzlich muss zuerst der Zugriff auf die Kamera erlaubt werden, sobald dies geschehen ist, wird bei vielen Anwendungen dem User ein Video von sich selbst gezeigt(muss jedoch nicht ziwngend sein). Ab diesem Zeitpunkt fängt das Modell an, die Daten vom Blickpunkt zu sammeln. Es gibt viele verschiedene Möglichkeiten, inwiefern der WebGazer verwendet werden kann: bei Suchmaschinen, um herauszufinden, wo der Nutzer öfter oder seltener hinschaut; Spiele, die nur mit den Augen gespielt werden; und viele andere Anwendungszwecke. [web, FRM+20]

2.5. Emotionserkennung

Der Emotiondetector basiert auf der FrontEnd-Emotiondetection Bibliothek von Kevin Hsiao auf GitHub. Diese nutzt die TensorFlow.js Bibliothek, welche ein zuvor trainiertes Modell lädt. Die TensorFlow.js ist eine JavaScript-Bibliothek für maschinelles Lernen.
Die FrontEnd Emotiondetection Bibliothek bietet drei verschiedene Modelle, um Emotionen zu erkennen und es werden zwei verschiedene Module zur Erkennung benutzt.

Zum einen wird die Chrome Shape Detection API verwendet, die es ziemlich einfach macht, Gesichter, Worte und Barcodes in Bildern zu erkennen. [Ste19]
Zum anderen wird die face-api.js Bibliothek verwendet, dies ist ebenfalls eine JavaScript Bibliothek, die zur Erkennung von Gesichtern dient und noch viele andere Features bietet, wie Emotionserkennung, Altervorhersage, Geschlechtserkennung uvm. und es ermöglicht, im Web und Mobilgeräten verwendet zu werden. [**?**] [ES19]
Es werden zudem zwei verschiedene Datensätze verwendet, um das Emotionserkennungs-Modell zu trainieren: Microsoft FERPlus und Real-world Affective Faces Database (RAF-DB), bei beiden handelt es sich um Sammlungen an Fotos, die verschiedene Emotionen oder auch Alter und Geschlecht liefern, um das Modell zu trainiren.
Die drei verschiedene Modelle sind:

- MobileNetImage.html
  Dieses Modell verwendet die Chrome Shape Detection API. Hier wird ein Bild hochgeladen und das Modell erkennt Gesichter auf dem Bild.
- MobileNetWebcam.html
  Dieses Modell verwendet ebenfalls die Chrome Shape Detection API. Bei dieser Variante muss der Zugriff auf die Webcam zugelassen werden, anschließend gibt es ein Videofeedback und ein Duplikat dessen, auf dem in Echtzeit die Emotion erkannt wird.
- TinyFaceDetectWebcam.html
  Dieses Modell verwendet die face-api.js Bibliothek. Hier gibt es ebenfalls ein Videofeedback, wo die Emotion in Echtzeit erkannt wird.

Die Rückmeldung der Emotion erfolgt über einen Kasten, der um das erkannte Gesicht gezeichnet wird und darüber wird die Emotion eingeblendet.
Die Modelle, die die Chrome Shape Detection API verwenden, laufen nur auf dem Chrome Browser für Windows10, MacOS oder Chrome auf Android. Zudem müssen dafür die Experimental Web Platform Features aktiviert werden via chrome://flags/#enable-experimental-web-platform-features. Dies sollte so aussehen (Figure 2.2.):

Sollten diese Features jedoch nicht aktiviert sein, wird dem Nutzer eine Meldung vom Browser angeziegt (Figure 2.3.):

Wird hingegen die face-api verwendet, so läuft dieses Modell zusätzlich zu Chrome noch auf Safari und Firefox.
Für diese Arbeit wurde das MobileNetWebcam Model verwendet zusammen mit der Chrome Shape Detection API. [Hsi18]

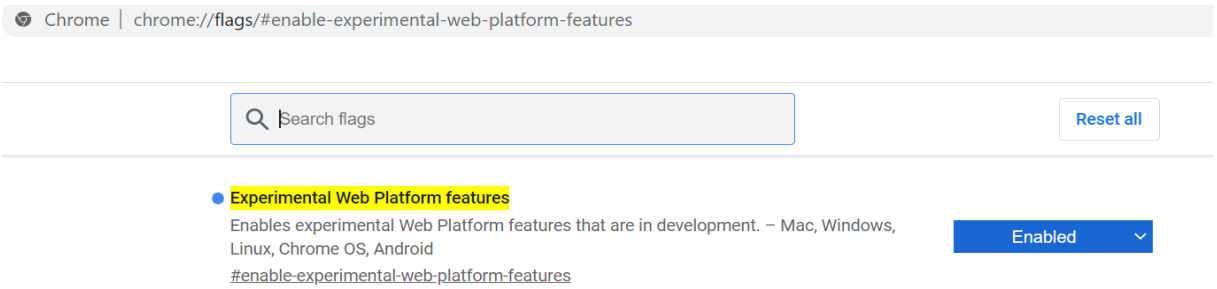

Figure 2.2.: Experimental Features aktiviert

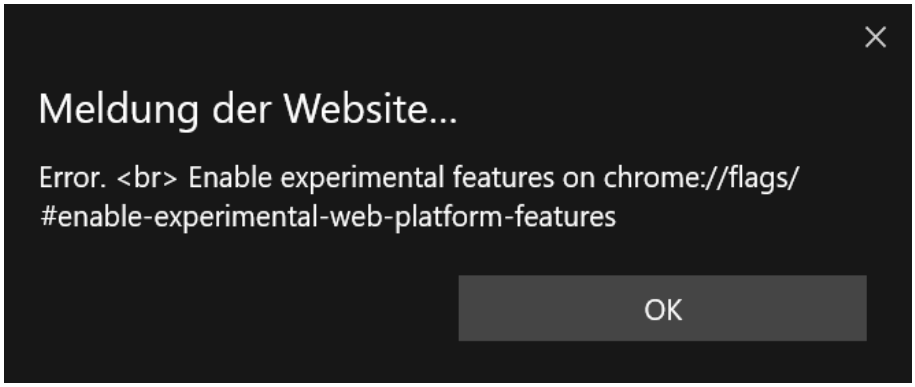

Figure 2.3.: Meldung vom Browser, wenn die Experimental Features in Chrome nicht aktiviert sind

3. Methoden

3.1. Übersicht

1. Über das Framework

Dieses Framework wurde erstellt, um Studien durchführen und auswerten zu können. Im Rahemn dieser Bachelorarbeit wurden dafür ein Eyetracker und eine Emotionserkennung integriert. Das Framework wurde mit JavaScript umgesetzt und die Kommunikation mit dem Server, der mit PHP erstellt wurde, erfolt über AJAX. Im Folgenden findet sich eine ausführliche Übersicht über die Funktion und den Ablauf der einzelnen Varianten, die zu diesem Framework gehören.

2. Zu Beginn

Datenschutz
Zu aller erst wird die Startseite(index.html) geöffnet und es wird direkt ein PopUp-Fenster angezeigt, das Informationen zum Datenschutz enthält. Diese könnnen akzeptiert oder abgelehnt werden, werden sie jedoch abgelehnt, wird der User zu Google weitergeleitet. Sobald dieser jedoch akzeptiert wird, kann mit der Startseite interagiert werden (Figure 3.1.).

Erklärungen
Rechts auf der Seite steht ein kurzer Überblick über alle Möglichkeiten, welche Daten erfasst werden und wie diese funktionieren. Zudem ist links unten noch ein Löschen-Button, um die bisherigen Daten, die erhoben wurden, mittels einer ID, die dem User mitgeteilt wird, zu löschen (Figure 3.2.).

Auswahl und Datenangabe
Links auf der Webseite kann der User sich dann zwischen den vier Varianten Einfaches Foto, Exetracker, Emotiondetection, beides entscheiden. Diese Aushwal muss geroffen werden, ansonsten kann der User nicht weitergeleitet werden.
Zusätzlich kann der User noch verschiedene freiwillige Angaben für die Statistik angeben wie Alter, Geschlecht, Status an der Universität. Hat der User alles aus-

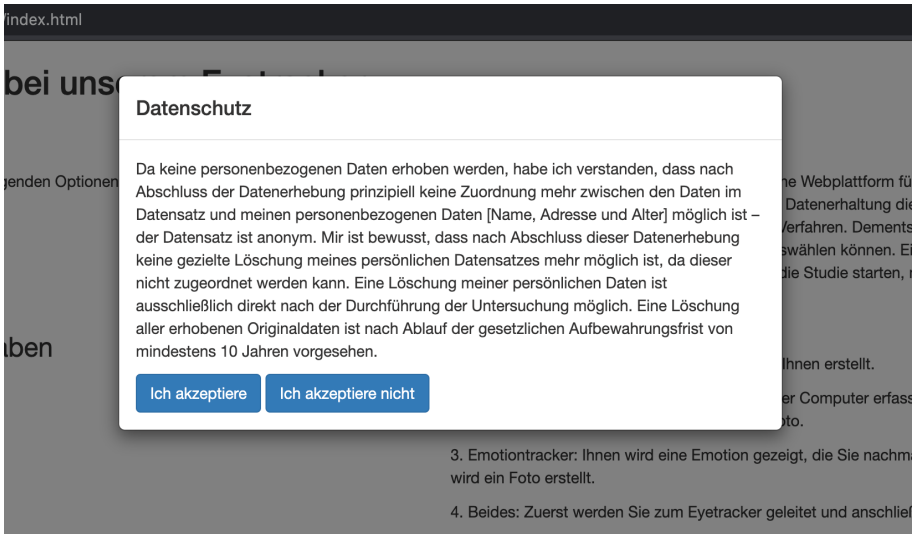

Figure 3.1.: PopUp mit dem Datenschutzhinweis, der akzeptiert werden muss, bevor das Framework verwendet werden kann

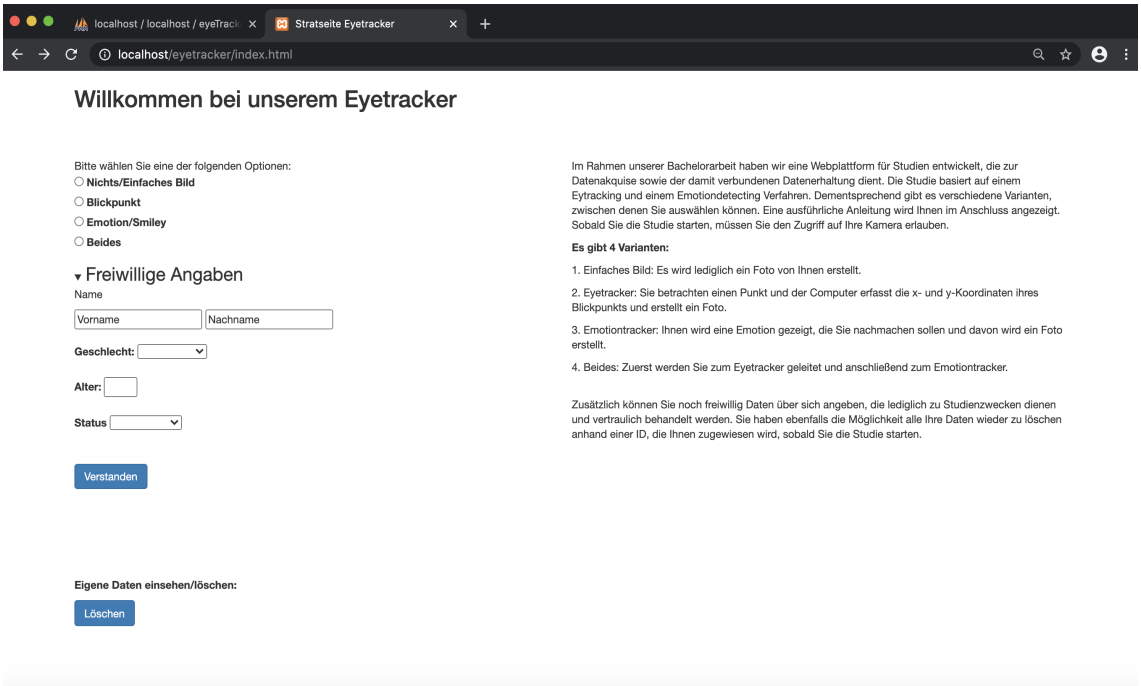

Figure 3.2.: Die Startseite vom Framework(index.html), die der User zuerst zu sehen bekommt und welche die verschiedenen Variaten kurz erklärt

gewählt und gegebenenfalls angegeben, wird er zu seinem ausgewählten Modell weitergeleitet.

3. Die Varianten

Zentriert unter der Überschrift wird bei allen Varianten auf der Seite dem User noch zusätzlich seine aktuelle Session-ID angezeigt, anhand dessen er später noch seine Daten und Bilder anschauen und/oder löschen kann.

Einfaches Foto

Zu Anfang erscheint wieder ein PopUp-Fenster, das der User sich durchlesen sollte, statt es wegzuklicken, wie es oftmals der erste Impuls ist, damit dem Benutzer bewusst ist, was nun zu tun ist. Hat sich der Nutzer dieses durchgelesen, verstanden und den Zugriff auf die Kamera erlaubt, verschwindet das Fenster und der Nutzer kann mit der Studie beginnen.
Links oben erscheint ein Videofeedback vom User mit einem Kasten, der Rückmeldung darüber gibt, ob die Versuchsperson gut von der Kamera erkannt wird. Der User kann nun beliebig viele Fotos von sich machen, indem er auf den "Foto machen"-Button klickt (Figure 3.3.).

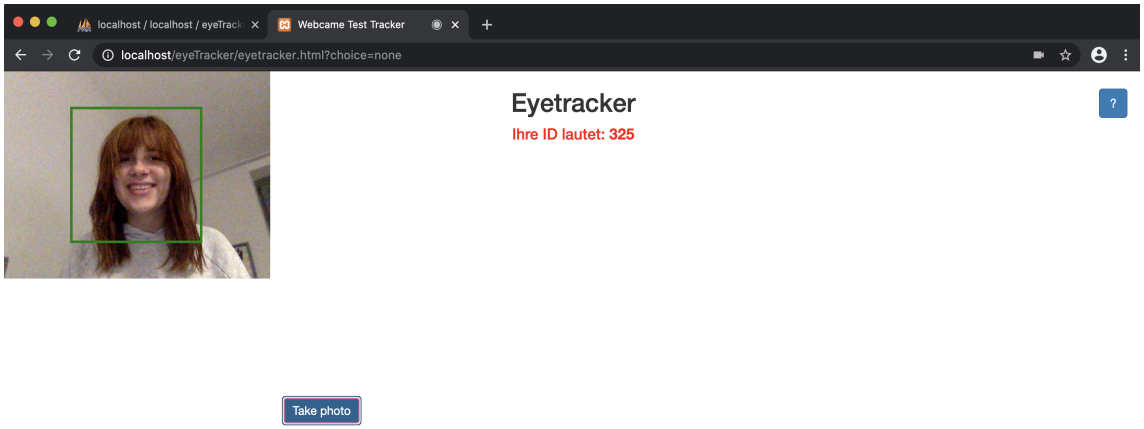

Figure 3.3.: Die erste Variante: Ein einfaches Bild wird vom User aufgenommen ohne jegliche weitere Features

Blickpunkt

Auch hier erscheint am Anfang wieder ein PopUp-Fenster, das der User sich durchlesen sollte, hierbei handelt es sich um dasselbe PopUp wie bei der vorherigen Variante. Das PopUp beinhaltet sowohl die Erklärung zum einfachen Bild als auch zum Bild mit Blickpunkt.
Dieses mal muss der Nutzer ebenfalls alles lesen und den Zugriff auf die Kamera erlauben.
Das Videofeedback befindet sich ebenfalls links oben im Fenster und gibt wieder mit einem Viereck Rückmeldung über die optimale Position des Users. Dieses Mal muss der User jedoch den schwarzen Punkt in der Mitte des Bildschirms anschauen und dabei ein Foto von sich machen. Dies geschieht, indem der Nutzer den schwarzen Punkt, den er anschaut auch anklickt. Im Hintergrund berechnet währenddessen die WebGazer Bibliothek die x- und y-Koordinaten des Blickpunkts vom Nutzer und die Mauszeigerposition und sagt somit einen Punkt vorher, den der User anschaut. Dieser wird auch auf dem Bildschirm durch einen roten sich bewegenden Punkt dargestellt.
Auch hier können beliebig viele Fotos gemacht werden (Figure 3.4.).

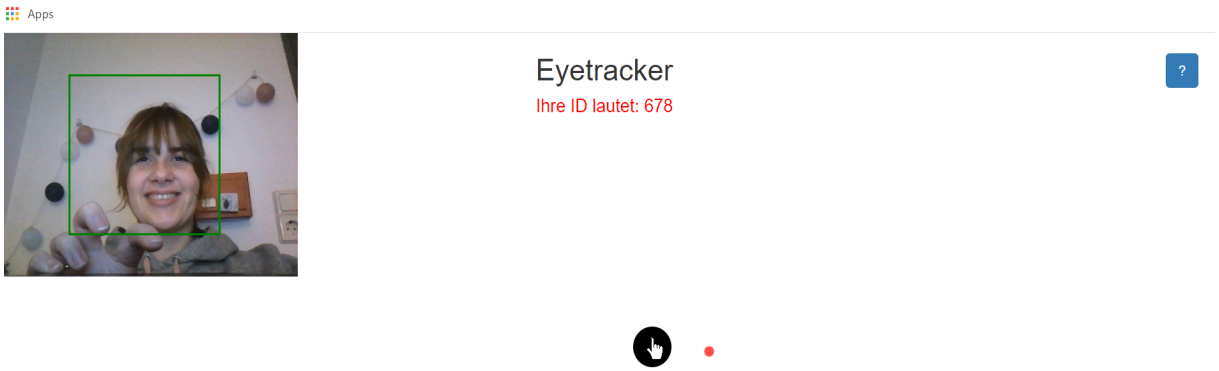

Figure 3.4.: Zweite Variante: Blickpunktvorhersage. Hier schaut der User einen Punkt auf dem Bildschirm an und mithilfe der Webcam und des Mauszeigers wird erkannt, wo der User hinschaut.

Emotionserkennung

Hier erscheint natürlich auch zu Beginn ein PopUp, das erklärt, was zu tun ist. Zudem muss wieder der Kamerazugriff erlaubt werden und zusätzlich muss der User die Experimental Web Platform Features über chrome://flags/#enable-experimental-web-platform-features aktivieren, daürber wird der User aber auch über ein Alert informiert.
Sobald das erledigt ist, sieht der User zwei Videofeedbacks, eins davon zeigt dem User an, welche Emotion dieser gerade zeigt. In der Mitte zwischen den beiden Videos sieht der User einen Smiley, der eine der folgenden Emotionen zeigt: Angst, Glücklich, Traurigkeit, Neutral und Überraschung. Die Emotion, die der Smiley zeigt, soll der User nachahmen und davon dann mit "Foto machen" ein Bild erstellen. Mit dem Button "Neuer Smiley" kann der User sich einen neuen Smiley mit neuer Emotion anzeigen lassen.
Ebenfalls kann der User dann wieder beliebig viele Fotos von sich machen (Figure 3.5.).

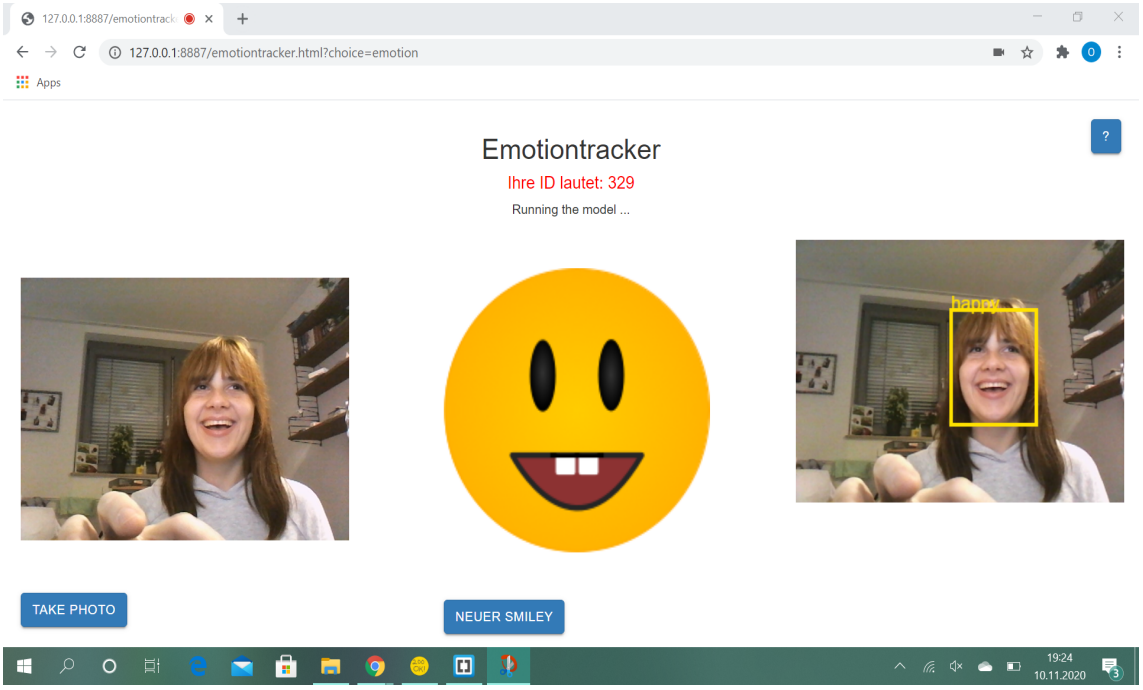

Figure 3.5.: Dritte Variante: Emotionserkennung. Hier wird dem User eine Emotion vorgegeben und diese soll nachgemacht werden, mit der Webcam wird versucht, die gezeigte Emotion zu erkennen.

Beides

Die Variante "Beides" beinhaltet zum einen den Blickpunkt und zum anderen die Emotion. Zuerst wird der User auf die Seite mit dem Blickpunkt weitergeleitet und kann mit der Seite interagieren(für nähere Informationen siehe den Punkt *Blickpunkt*) und sobald der User da ein (oder auch mehrere) Bild(er) gemacht hat, erscheint ein Button, der ihn dann zu der Variante Emotionserkennung(für weitere Informationen siehe Punkt *Emotionserkennung*) weiterleitet (Figure 3.6.).

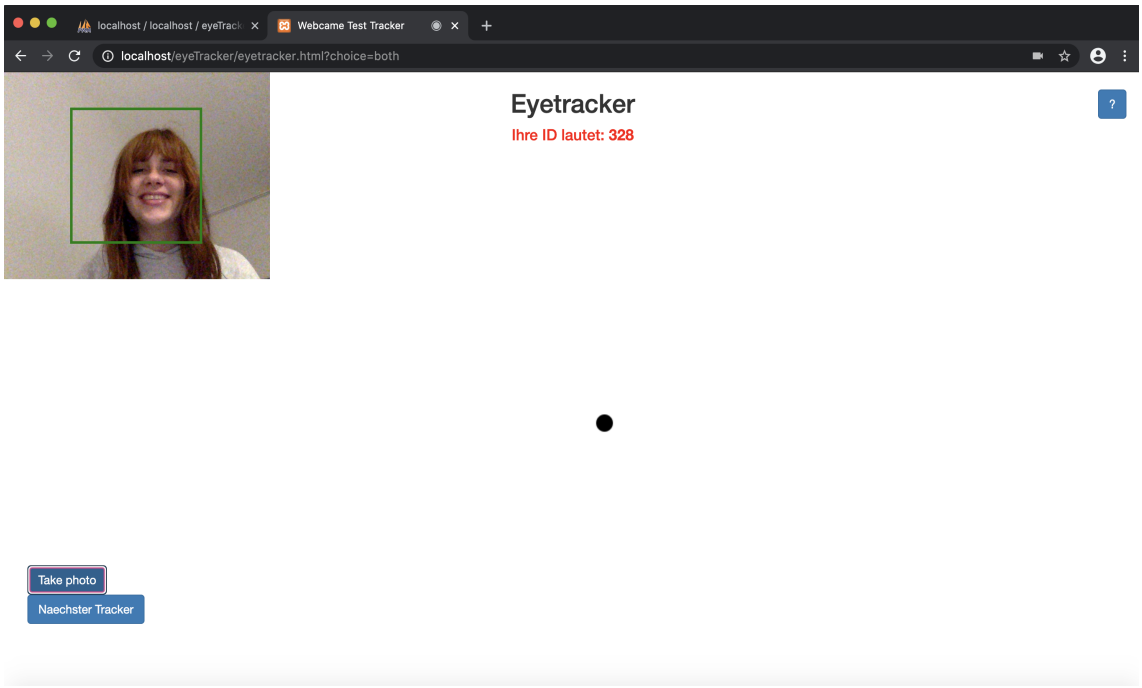

Figure 3.6.: Variante Beides: Blickounktvorhersage und Emotionserkennung. Erst kann der User die Blickpunkterkennung durchführen und danach die Emotionserkennung, somit werden die Daten direkt auf einmal erfasst.

4. Daten löschen/anschauen

Wie schon zu Anfang erwähnt, hat ein Nutzer die Möglichkeit, seine Daten, die er angegeben hat und die Bilder, die er gemacht hat, mithilfe einer Session-ID, die ihm während der Studie zugeteilt wurde.

Auf der Startseite befindet sich links unter der Auswahl der Varianten und den Freiwilligen Angaben ein Button, der für das Löschen der Daten steht. Hat der User den Wunsch, seine Daten zu löschen, so kann er diesen anklicken und wird dann auf eine Seite geleitet, auf der ein Feld zu sehen ist, in das der User seine Session-ID eingeben kann. Handelt es sich um eine ungültige ID, so wird das direkt mitgeteilt,

ansonsten wird der User auf eine Seite geleitet, auf der all seine angegebenen Daten und gemachten Bilder angezeigt werden. Dort befindet sich dann auch ein Button, um diese zu löschen.
Sollte sich der Nutzer gegen das Löschen entscheiden, kann er einfach über einen "Zurück"-Button wieder zurück zur Startseite springen (Figure 3.7.).

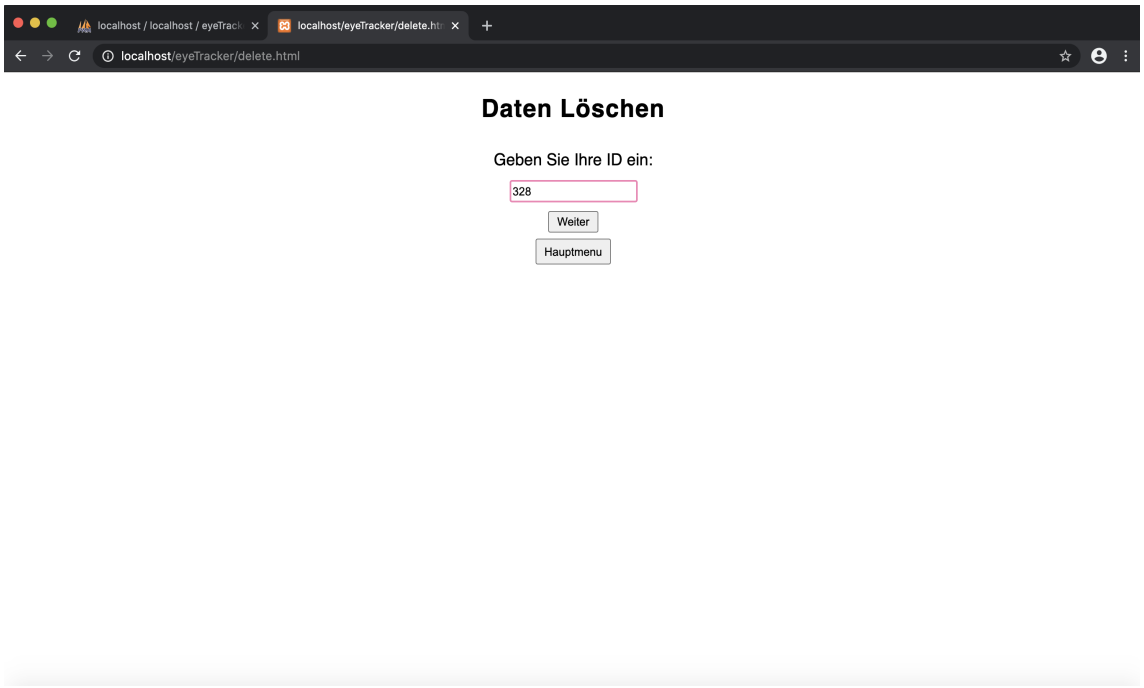

Figure 3.7.: Die Möglichkeit zur Löschung der Daten führt zu einer Seite, wo die ID des Users eingegeben wird, anhand derer die Daten aus der Datenbank ausgelesen werden.

Sobald der Nutzer auf "Weiter" drückt, wird er zur nächsten Seite geleitet, eine PHP-Datei, wo alle Bilder und Daten, die zu dieser ID gehören, aufgelistet werden (Figure 3.8.).

Je nachdem, wie viele Eintrage zu dieser ID gehören, kann runtergescrollt werden. Am Ende befindet sich unter allen Daten dann ein Button, um die oberen Daten wieder zu löschen (Figure 3.9.).

Welche Daten werden wie gespeichert?

In der Datenbank von Roufayda Salaheddine wurden mehrere Tabellen aufgesetzt, in denen die verschiedenen erfassten Daten gespeichert werden. Eine dieser Tabellen ist die Probanden-Tabelle, dort werden alle Teilnehmer und deren freiwilligen Angaben gespeichert (Figure 3.10.).

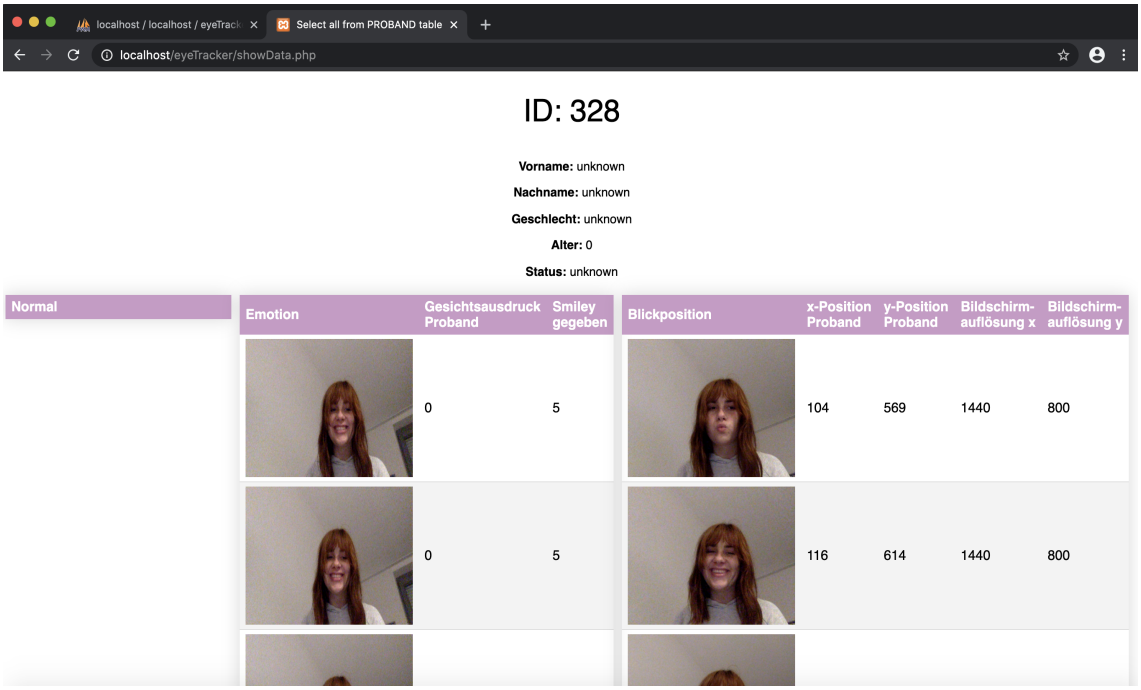

Figure 3.8.: Möchte der Nutzer seine Daten löschen, werden diese ihm zuvor anhand der vorherig eingegebenen ID aufgelistet.

In jeder Tabelle werden verschiedene Werte gespeichert, um später Studien auswerten und Statistiken aufstellen zu können. Im Folgenden wird erklärt, welche Werte auf welche Art und Weise gespeichert werden:

Zu Beginn werden die frewilligen Angaben gespeichert: Vor- und Nachname, Alter, Status an der Uni und Geschlecht. Diese Angaben muss der Nutzer nicht machen, sie dienen allein statistischden Zwecken und sind für die Studien nicht von größter Bedeutung. Diese Daten werden als Integer oder als String gespeichert.

Die folgenden Daten werden immer gespeichert, unterliegen also nicht der freiwilligen Angabe des Users. Diese gehören zu den Studien und müssen demnach im Rahmen einer Studie auch erfasst werden:
Im Rahmen des Eyetrackers(inkl. Einfaches Bild) wird zum einen ein Bild auf dem Server gespeichert, in der dazugehörigen Tabelle wird die url zum Ordner mit dem Bild gespeichert. Zum Anderen wird die Bildschirmauflösung gespeichert, in der Mitte der x- und y-Achse des Bildschirms befindet sich der Punkt, den der Proband

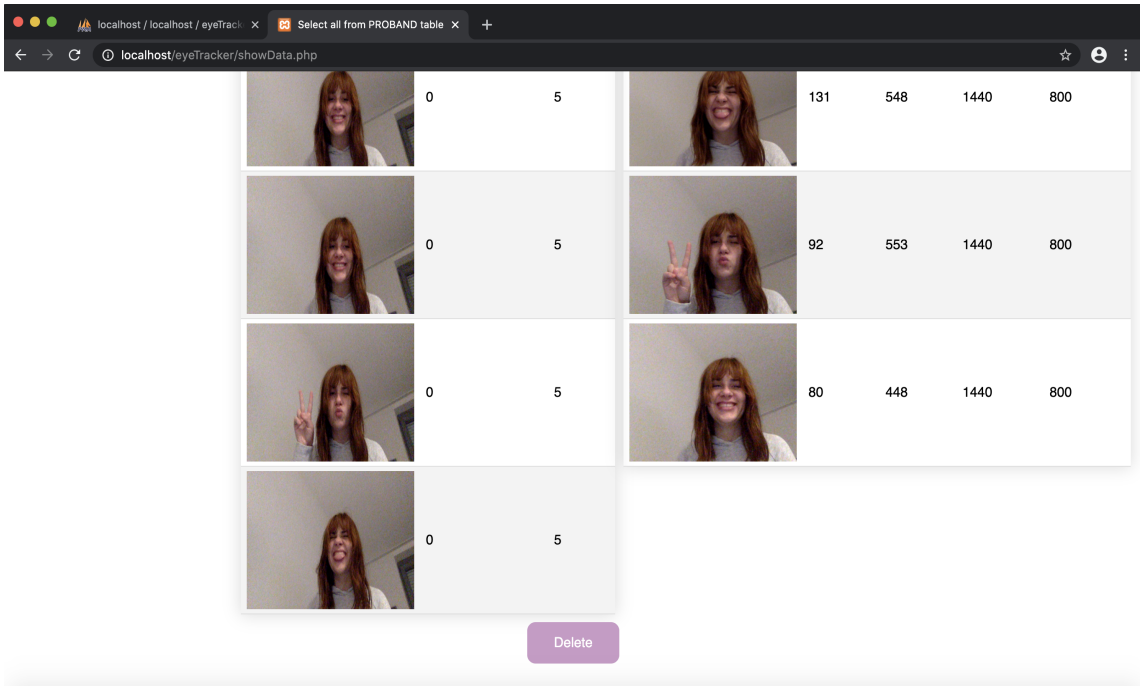

Figure 3.9.: Die angezeigten Daten können schlussendlich gelöscht werden, der Nutzer kann sich jedoch auch dagegenentscheiden.

+ Optionen

	id	f_name	l_name	gender	age	status
Bearbeiten Kopieren Löschen	313	Anna	Müller	w	25	student
Bearbeiten Kopieren Löschen	314	Renate	Schmidt	w	50	mitarbeiter
Bearbeiten Kopieren Löschen	315	unknown	unknown	unknown	0	unknown
Bearbeiten Kopieren Löschen	317	Alexander	unknown	m	0	student
Bearbeiten Kopieren Löschen	318	Oliwia	Oles	w	23	student
Bearbeiten Kopieren Löschen	319	Ralf	Weber	m	43	sonstige
Bearbeiten Kopieren Löschen	320	Roufayda	Salaheddine	w	21	student
Bearbeiten Kopieren Löschen	321	unknown	unknown	unknown	0	unknown
Bearbeiten Kopieren Löschen	322	unknown	unknown	unknown	0	unknown

Figure 3.10.: Eine von mehreren Tabellen, die in der Datenbank existieren. In dieser Tabelle werden die Probanden gespeichert, die an einer Studie teilgenommen haben.

anschaut. Zusätzlich dazu wird die Position des Blickpunktes des Probanden in Pixel

als Integer gespeichert. Mit diesen vier Angaben kann später berechnet werden, wie genau der Eyetracker die Augenposition erfasst hat und ob der Nutzer in erster Linie überhaupt während der Erstellung des Fotos auf den schwarzen Punkt geschaut hat oder nicht.
Im Rahmen der Emotionserkennung wird ebenfalls ein Bild gespeichert, genauso wie beim Eyetracker. Außerdem wird hier gespeichert, welche Emotion der Smiley dem Probanden vorgegeben hat. Die Smileys wurden mit dem Programm InkScape erstellt und sind Vektorgraphiken. Es gibt fünf Emotionen, die angezeigt werden können und diese werden als Integer Wert zwischen 1 und 5 gespeichert. Dabei entspricht :

1 = Glücklich/Happy
2 = Traurig/Sad
3 = Ängstlich/Fear
4 = Überrascht/Surprise
5 = Neutral
(Figure 3.11. und Figure 3.12.)

(a) (b) (c)

Figure 3.11.: Emotionen: Glücklich, Traurig, Ängstlich

(a) (b)

Figure 3.12.: Emotionen: Überrascht und Neutral

Es wird jedoch nicht nur die vorgegebene Emotion gespeichert, sondern auch die

Emotion, die der User zeigt, bzw. die Emotion, die vom Computer erkannt wird. Diese wird ebenfalls als Integer zwischen 1 und 5 gespeichert. Anhand dieser beiden Werte kann später verglichen werden, wie oft/genau die Emotionserkennung die richtige Emotion erkannt hat (unter der Annahme, dass der User nicht mit Absicht die falsche Emotion gezeigt hat).

3.2. Tutorial

In diesem Abschinitt folgt eine kurze, schrittweise Anleitung zur Nutzung des Frameworks zur Durchführung einrt Studie. Das Tutorial ist in drei größere Abschnitte eingeteilt, die jeweils die Startseite, die verschiedenen Varianten und die Möglichkeit zum Löschen der Daten erklären.

1. Start

1. Datenschutz durchlesen und akzeptieren/ablehnen
2. Anleitung und Erklärung rechts durchlesen
3. Links die gewünschte Variante auswählen
4. Freiwillige Angaben wie Geschlecht und Alter
5. "Verstanden"-Button drücken

2. Die Varianten

Für alle Varianten gilt:

1. Kamerazugriff erlauben
2. Anleitung durchlesen und "Verstanden"-Button anklicken
3. Einmalige Session-ID merken

Variante Einfaches Bild:

1. In die Kamera schauen, sodass der Kasten um das Gesicht grün ist
2. Mit "Foto machen" beliebig viele Bilder von sich machen.

Variante Bild mit Blickpunkt:

1. In die Kamera schauen, sodass der Kasten um das Gesicht grün ist
2. Den schwarzen Punkt auf dem Bildschirm anschauen
3. Den schwarzen Punkt anklicken, um ein Foto zu machen

Variante Emotionserkennung:

1. Experimental Web Platform Features über chrome://flags/#enable-experimental-web-platform-features aktivieren
2. In die Kamera schauen und die Emotion nachmachen, die der Smiley vorgibt
3. Nach Belieben einen neuen Smiley mit neuer Emotion generieren lassen über "Neuer Smiley"
4. Mit "Foto machen" beliebig viele Bilder von sich machen.

Variante Beides:

1. Siehe Variante Anleitung Blickpunkt
2. Nach Beendigung erscheint ein Button, der zur zweiten Variante weiterleitet, sobald dieser angeklickt wird
3. Siehe Variante Emotionserkennung

3. Löschen

1. Button "Daten löschen" anklicken
2. Session-ID angeben
3. Angegebene Daten und gemachte Bilder anschauen
4. Über den "Löschen"-Button diese bei Wunsch löschen
5. Über "Zurück" Button zurück zur Startseite springen

3.3. Kommunikation mit dem Server

Die Kommunikation mit dem Server, den Roufayda Salaheddine im Rahmen ihrer Bachelorarbeit aufgestezt hat, erfolgt klassisch über AJAX und das XMLHttpRequest-Objekt. Die grundlegende Struktur des Requests sieht folgendermaßen aus:

```
1       var xr = new XMLHttpRequest();
2       var url = 'Destination_PHP_Document';
3       var data = new FormData();
4       data.append('keyword', value);
5
6       xr.open("POST", url, true);
7       xr.send(data);
```

Bei diesem allgemeinen Beispiel handelt es sich um ein POST-Request, das heißt, dass Daten auf dem Server modifiziert werden sollen, beispielsweise dadurch, dass eine der Tabellen um einen Eintrag erweitert werden. Das PHP-Skript empfängt den

Request und verarbeitet diesen. Dies kann dann folgendermaßen aussehen:

```
1  if (@$_POST['prenom'] == "Vorname" || @$_POST['prenom'] == "") {
2      $f_name = 'unknown';
3  } else {
4      $f_name = $db->real_escape_string($_POST['prenom']);
5      echo $f_name;
6  }
```

Dieses Beispiel zeigt, wie und wann etwas in eine Tabelle eingetragen wird. In diesem Fall handelt es sich um den Vornamen: Wurde auf der Startseite das Feld für den Vornamen leer gelassen oder der default Wert "Vorname" nicht geändert, so wird in der enstprechenden Probanden Tabelle für den Vornamen "unknown" eingetragen.
Wenn der User jedoch etwas eingetragen hat, so tritt der else-Fall ein und der eingetragene Wert wird übernommen und in der Tabelle an der richtigen Stelle einegragen.

3.4. Funktionen und GUI

Im Folgenden werden die einzelnen Funktionen, die das alles ermöglicht haben und welchen Steuerlementen, die zur GUI gehören, diese zugrunde liegen. (engl. für graphical user interface) auf deutsch grafische Benutzeroberfläche beschreibt die Benutzerschnittstelle eines Computers. Dafür werden verschiedene Icons, Buttons und noch viele andere Elemente verwendet. [gui19, wik20b] Die verschiedenen Elemente werden nach JavaScript-, HTML oder PHP-Datei sortiert, um besser zuordnen zu können, wo sie sich abspielen.

3.4.1. start.js

Diese Datei ist dafür da, um die Startseite(index.html) zum Laufen zu bringen, mit Inhalt zu füllen und um die eingegeben Daten im Anschluss zu speichern. Sie besteht lediglich aus zwei einfachen Funktionen:

- getData():
  Diese Funktion ist dazu da, um den Text über den Datenschutz aus der zugehörigen Tabelle in der Datenbank aufzurufen und auf der Startseite in einem PopUp anzuzeigen.
  Zudem wird aus dieser Tabelle auch der Text aufgerufen, der die Anleitungen und Ekrlärungen über die verschiedenen Varianten enthält.

  Dafür wird je ein neues Request Objekt erstellt über new XMLHttpRequest(); und über eine zurvor vergegebene ID in einem HTML-Tag wird in dieses Tag

dann der Text eingefügt, der aus der entsprechenden PHP-Datei herausgelesen wird.

```
var oReq2 = new XMLHttpRequest(); // New request object
oReq2.onload = function() {
    var anleitungX = document.getElementById("anleitung");
    anleitungX.innerHTML = this.responseText;
};
oReq2.open("get", "tutorial.php", true);
oReq2.send();
```

- myFunction():
  In dieser Funktion werden die eingegebenen Daten des Nutzers der Form entnommen und dann per xm an den Server gesendet, in dem ebenfalls ein neues Request Objekt erstsellt wird, um diese zu speichern.
  Beispiel anhand des Alters:

```
var age = document.getElementById("alter").value;
var xr = new XMLHttpRequest();
var url = 'insert.php';
var data = new FormData();
data.append('age', age);
xr.open("POST", url, true);
xr.send(data);
```

Zudem wird über eine if-Abfrage herausgefunden, für welche Variante sich der User entschieden hat und abhängig davon, wird er dann zum richtigen Modell weitergeleitet. Bei der Weiterleitung wird an den eigentlichen Link noch die gewählte Variante über "?blub=xxx" angehängt, um später die weiteren Funktionen zu erleichtern.
Zuerst wird geprüft ob ein Form ausgewählt wurde oder nicht:

```
var mychoice;
    var choiceNone = document.getElementById("none").value;
```

Danach wird die if-Abfrage durchlaufen:

```
//check which option was chosen

if(document.getElementById("none").checked){
            mychoice = choiceNone;

    //redirect to exetracker with chosen option
        window.document.location.href = "eyetracker.html?choice=" + mychoice;
 } else if(document.getElementById("dot").checked){
            mychoice = choiceDot;

    //redirect to exetracker with chosen option
        window.document.location.href = "eyetracker.html?choice=" + mychoice;
    } else if(document.getElementById("emotion").checked){
            mychoice = choiceEmotion;

    //redirect to exetracker with chosen option
        window.document.location.href = "emotiontracker.html?choice=" + mychoice;
    } else{
            mychoice = choiceBoth;

```

```
21          //redirect to exetracker with chosen option
22              window.document.location.href = "eyetracker.html?choice=" + mychoice;
23          }
```

Diese Funktionen sind an die Steuerelemente aus der index.html gebuden:

3.4.2. index.html

Auf der Startseite befinden sich die meisten Steuerlemente, ein Großteil der Funktionen die dazugehören, befinden sich in der start.js.

Dialog

Zu Anfang des Seitenaufrufs erscheint dem User erst mal ein Dialogfenster, welches den Datenschutzhinweis enthält. Dialoge sind in aller Regel modale Schaltflächen, das heißt, dass der User keine Eingaben auserhalb dieses Fensters machen kann, solange dieses nicht ausgeblendet wird. Das Fenster kann über Buttons geschlossen werden. Das Fenster wird über eine jQuery und die dazugehörige ID, hier ist es *myModal*, geöffnet, sobald die Startseite aufgerufen wird.

```
1   $(document).ready(function(){
2                   $("#myModal").modal('show');
3           });
```

Buttons

Zusammen mit den Buttons im Dialogfenster gibt es 4 Buttons auf der Startseite, die angeklickt werden können.
Im Dialogfenster gibt es einen Akzeptieren- und Nicht Akzeptieren-Button(siehe Bild bei 3.1.2 *Datenschutz*. Sollte der Datenschutzhinweis nicht akzeptiert werden, so wird der User zu Google weitergeleitet. Wird dieser jedoch akzeptiert, so kann der User schließlich die ganze Startseite sehen und mit dieser interagieren.
Ansonsten befinden sich auf der Startseite noch zwei weitere Buttons, beide mit ein wenig Abstand untereinander auf der linken Seite.
Beim oberen der beiden handelt es sich um einen "Verstanden"-Button, den der User betätigt, sobald er sich die Erklärungen durchgelsen und seine Wahl getroffen hat. Die Funktion *myFunction()* wird aufgerufen, sobald der Button angeklickt wird.
Mit dem Löschen-Button kann der Benutzer seine eigenen Daten einsehen und gegebenenfalls löschen, dafür wird der User auf eine neue Seite weitergeleitet. Die Funktion *success()* wird dabei aufgerufen:

```
1   function success() {
2                   if(document.getElementById("deleteID").value==="") {
3                       document.getElementById('btn').disabled = true;
4                   } else {
5                       document.getElementById('btn').disabled = false;
```

```
6                   }
7               }
```

Steuerelemente, die durch den Nutzer verändert werden, liegen in einem sogenannten Formular, das später an den Server geschickt wird. Dazu gehören hier die Radiobuttins und Textfelder.

Radiobuttons

Da es vier Varianten zur Auswahl gibt und damit der User nur eine davon auswählen kann, wird dies mithilfe von Radiobuttons gelöst. Hiermit ist nur eins auswählbar. Solange der User keine Variante auswählt, kann er auch nicht weitergeleitet werden.

Abgesehen von den verpflichtenden Angaben wie Auswahl der Variante und Betätigung des Verstanden-Buttons, gibt es noch weitere Steuerelemente, die eine freiwillige Angabe ermöglichen und nur der Statistik dienen.

Textfeld/Kombinationsfeld Dropdown

Es gibt Textfelder für Vor- und Nachnamen sowie Listenfelder für Status und Geschlecht. Das Alter ist ein Auswahlfeld, in das entweder etwas eingetippt oder über Pfeile das Alter hoch oder runter gezählt werden kann.

Werden ein oder mehrere dieser Felder nicht ausgefüllt, so wird für dieses Feld in der Datenbank der Wert "unknown" gespeichert.

3.4.3. trackerLib.js

Die trackerLib beinhaltet fünf Methoden, die zusätzliche Funktionen für den Eyetracker und die Emotiondetection bietet.
Zum Einen werden dort die Smileys oder der Punkt in der Mitte des Bildes generiert und zum Anderen wird dort ein Bild des User erstellt und an die Datenbank geschickt.

- createImg():
  Diese Funktion wird durch die Funtkionen fromPoint() oder fromEmotion() aufgerufen. Die Funktion kriegt als Parameter die gewählte Variante und ein CanvasElement übergeben. Aus dem Canvaselement wird über toDataURL() ein Bild bzw ein base64 Code von einem Bild herausgezogen.

```
1       var img = HTMLCanvasElement.toDataURL().toString();
```

Dieser Base64 Code wird anschließend wieder per Xm an den Server gesendet, dekodiert und dann in der Datenbank gespeichert.

- fromPoint():
  fromPoint() wird in der ecetracker.html aufgerufen, sobald der Nutzer auf "Foto machen" klickt. Als Parameter wird der Funktion die gewählte Variante "dot" oder "both" übergeben, die ist wichtig, damit nach der Aufnahme des Bildes, der Button zur Weiterleitung zur Emotiondetection erscheint, sollte die Variante Beides gewählt worden sein.

```
1     if(choice == "both"){
2       var nextTrack = document.getElementById("nextPage");
3       nextTrack.innerHTML = "<a␣href='emotiontracker.html?choice=emotion'><button␣class='btn␣btn-primary'>
4 ␣␣␣␣␣Naechster␣Tracker</button></a>"
5       }
```

Zudem wird hier das CanvasElement herausgezogen, was in dieser Funtkion dann an createImg() als Parameter übergegeben wird. Zudem werden hier die x- und y-Kooridinaten des Blickpunkts über den Webgazer in Erfahrung gebracht und über xm an die Datenbank geschickt:

```
1     var xpos =webgazer.getCurrentPrediction().x;
```

- fromEmotion():
  In dieser Funktion wird ebenfalls wie in der fromPoint() das CanvasElement an die createImg übergeben und zudem die erkannte Emotion per xm wieder an die Datenback geschickt.
  Als Parameter werden das CanvasElement und die Emotion an diese Funktion übergeben.
- Generator():
  Hier wird per Zufall über Math.Random einer der fünf Smileys generiert, die eine der folgenden fünf Emotionen zeigen: Angst, Glücklich, Trauer, Neutral und Überraschung. Der Smiley wird in ein Html-Tag mit der passenden ID gemalt.

```
1   var placement = document.getElementById("smiley");
2               var num = Math.random();
3               if(num < 0.2){
4                     placement.innerHTML ="<img␣src='smileys/sad.png'␣alt='sad'␣width='300'␣height='300'>␣";
5               }else if(num >= 0.2 && num < 0.4){
6                   placement.innerHTML ="<img␣src='smileys/happy.png'␣alt='happy'␣width='300'␣height='300'>␣";
7               }else if(num >= 0.4 && num < 0.6){
8                     placement.innerHTML ="<img␣src='smileys/fear.png'␣alt='angry'␣width='300'␣height='300'>␣";
9               }else if(num >= 0.6 && num < 0.8){
```

```
10                  placement.innerHTML ="<img src='smileys/surprise.png' alt='surprise' width='300' height='300'> ";
11              } else {
12                  placement.innerHTML ="<img src='smileys/neutral.png' alt='neutral' width='300' height='300'> ";
13              }
```

Diese Funktion wird in der emotiondetection.html schon aufgerufen, während die Seite noch geladen wird (onload). Zudem wird diese Funktion über den Button "Neuer Smiley" aufgerufen.

- generateDot():
  Da sowohl die Variante "Einfaches Bild" als auch "Eyetracker" bzw. "Beides" auf die selbe Html-Datei weiterleiten, muss dafür gesorgt werden, dass der Blickpunkt nur bei der zweiten und dritten Variante angezeigt wird. Dies wird mit dieser Funktion bewerkstelligt.
  Als Parameter kriegt die Funktion die Variante als String übergeben, die aus dem Link entnommen wird. Handelt sich dabei um die Variante "dot" oder "both", so wird die if-Abfrage durchlaufen und kommt nicht in den else-Fall, da passiert einfach nichts. Wenn jedoch die Bedingung erfüllt ist, so wird in der Mitte des Bildschirms ein schwarzer Punkt gezeichnet, den der User dann anschauen soll.

```
1   if(link == "dot" || link == "both") {
2           /*var canvheight = 480; //height of video element
3           var canv = document.getElementById("chosen");
4
5           //get screen size
6           var x = screen.width;
7           var y = screen.availHeight;
8
9           //set size of canvas where circle is drawn
10          canv.width = x;
11          canv.height = (y-canvheight);
12
13              // draw circle
14                      var ctx = document.getElementById('chosen').getContext("2d");
15                      ctx.beginPath();
16                      ctx.arc(x/2, canv.height/2, 10, 0, 2 * Math.PI, true);
17                      ctx.fill(); */
18                      var x = document.createElement("BUTTON");
19           x.onclick = function(){trackerLib.fromPoint(linka)};
20          x.className = "dotbutton";
21          document.body.appendChild(x);
22      } else {
23       document.getElementById("body").innerHTML ="<button  onclick='trackerLib.fromPoint(link)' id
24  =  'takepic' class='btn btn-primary' style='position: absolute; top:300px;'>Take Photo</button>";
25      }
```

3.4.4. eyetracker.html

Der Eyetracker hat als einzige Steuerelemnete drei Buttons und ein Dialogfenster.

Dialog

Zu Beginn öffnet sich ein modales Dialogfenster, das die Anleitung zur Variante

Blickpunkt und Einfaches Bild enthält.

Button

Der erste Button befindet sich im Dialogfenster zu Anfang. Mit diesem Button wird das Fenster geschlossen, das läuft genauso ab wie in der index.html.
Ein weiterer Button mit dem Text "Take Photo" befindet sich auf der linken Seite, unter dem Videofeedback, jedoch nur bei der Variante *Einfaches Bild*. Bei der Variante *Blickposition* ist dieser Button nicht da, sondern nur ein schwarzer Punkt in der Mitte des Bildes. Dieser Punkt ist gleichzeitig ein Button, der dieselbe Funktion erfüllt wie der "Take Photo"-Button. Mit diesen Buttons wird jeweils die Funktion fromPoint() aufgerufen, um zum einen ein Foto zu erstellen und zum anderen, um die Blickposition zu erfassen, wenn nötig.
Der dritte Button befindet sich oben in der rechten Ecke auf der Seite, der ein Fragzeiechen zeigt. Mit diesem Button wird das Dialogfenster vom Anfang nochmak aufgerufen, falls der User sich die Anleitung nochmal durchlesen will.

Spezialfall Beide Varianten Wurde diese Variante ausgewählt, so erscheint ein weiterer Button "Nächster Tracker", sobald das erste Foto aufgenommen wurde. Natürlich können noch mehr Bilder aufgenommen werden, bevor dieser Button betätigt wird. Dieses Steuerelement dient der Weitrleitung zur Emotiondetection.

3.4.5. emotiondetection.html

Auf der Seite der Emotiondetection befinden sich insgesamt fünf Steuerelemente, vier Buttons und ein Dialogfenster.

Dialog

Auch hier öffnet sich zu Beginn ein Dialogfenster mit der Anleitung zur Emotiondetection, damit der Nutzer genau weiß, was zu tun ist.

Button

Im Dialogfenster befindet sich einer der vier Buttons, dieses Steuerelement schließt den Dialog, damit der User mit der Seite weiter interagieren kann.
Wie schon beim Eyetracker, befindet sich auf der linken Seite unter dem Videofeedback ein Button, der dieses Mal jedoch die Funktion from Emotion() aufruft ebenfalls ein Foto erstellt und statt der Blickposition die angezeigte und vom Modell erkannte Emotion erfasst.

Der dritte Button befindet sich ungefähr in der Mitte unter dem Smiley, der zwischen den beiden Videofeedbacks sitzt. Dieser Button ruft die Funktion Generator() auf, um einen neuen Smiley anzeigen zu lassen.
Oben in der rechten Ecke auf der Seite ist der letzte Button zu finden, dieser enthält ein Fragzeichen. Mit diesem Button wird das Dialogfenster vom Anfang nochmal aufgerufen, damit der User sich nochmal die Anleitung durchlesen kann.

3.4.6. webgazer.js

Zusätzlich zu dem Videofeedback Eyetracker bietet die webgazer Bibliothek noch viele weitere nützliche Funktionen:
Es kann beispielsweise ein Feeback Kasten um das Gesicht im Video gezeichnet werden, damit der User sieht, ob er gut im Bild positioniert ist und direkt von der Kamera gesehen wird oder nicht. Dies wird durch die Farbe des Kasten signalisiert. Grün heißt logischwerweise gut und rot bedeutet nicht gut.

Zudem kann auch das Gesicht des Users nachgezeichnet werden, da ist dann direkt klar, ob das Gesicht erkannt wird oder nicht. Dieses Feature wurde in diesem Framework jedoch ausgeschaltet.

Außerdem gibt es zwei Varianten um die Blickposition zu erfassen. Einerseits kann der Blickpunkt alle paar Millisekunden erfasst werden oder nur auf Abfrage. Hier wird die Blickposition nur auf Abfrage erfasst, indem der Nutzer ein Bild von sich machen lässt, davor wird diese Methode nicht aufgerufen.

3.4.7. delete.html und showData.php

In diesem HTML-Dokument kann der Nutzer den Befehl geben, seine Daten zu löschen. Im PHP-Dokuemnt werden dem Nutzer all seine Daten angeziegt, die anhand einer ID aufgerufen werden und diese können dann gelöscht werden.
Textfeld

Oben in der Mitte auf der Seite befindet sich ein Textfeld, in das eine ID eingetragen werden kann, sofern ein Nutzer, seine Daten anschauen und/oder löschen möchte.

Buttons

Hier sind in der Mitte zwei Buttons zu finden, der "Zurück"-Button leitet den User zurück auf die Startseite(index.html).
Der "Löschen"-Button hingegen leitet den User zu einer PHP-Datei, die dem User erst alle Daten anzeigt, die zur ID gehören. Am Ende der Seite befindet sich dann

enbenfalls zwei Buttons. Der "Zurück"-Button leitet den User wieder zur delete.html zurück. Der "Delete"-Button löscht die angezeigten Daten vom Nutzer, dies geschieht serverseitig über PHP.
Sollte die ID, die der Benutzer angegeben hat jedoch nicht existieren, wird das dem User mitgeteilt.

3.4.8. Zusammenhang zwischen den Dokumenten

In diesem Abschnitt wird kurz erklärt, wie die einzelnen Dokumente miteinander verknüpft sind und sich beeinflussen. Dies ist auch in der folgenden Grafik dargestellt (Figure 3.13.):

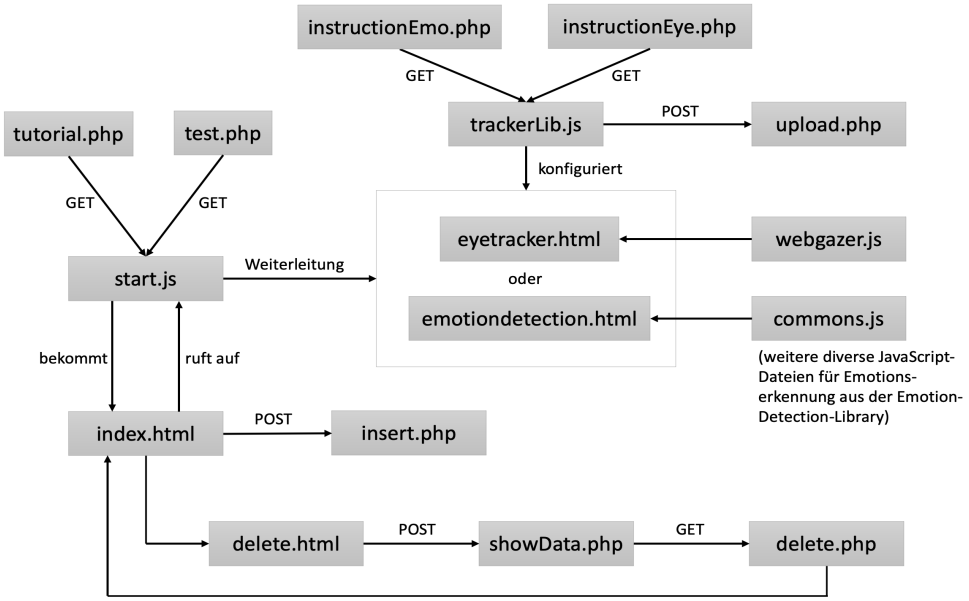

Figure 3.13.: Exemplarische Darstellung, wie die verschiedenen Dokumente miteinander zusammenhängen und sich gegenseitig beeinflussen.

Die index.html bindet die start.js ein, welche die Anleitungen und den Datenschutz über GET-Anfragen aus einer Datenbank bezieht, um diese dann auf der Startseite anzuzeigen. Über eine POST-Anfrage werden dann die ausgewählte Variante und die Angaben über den User an die insert.php übergeben, die diese Daten dann in die Datenbank einträgt.
Über die start.js wird der User dann auch von der index.html an die eyetracker.html oder an die emotiondetection.html weitergeleitet. Sowohl die eyetracker.html als

auch die emotiondetection.html binden verschiedene JavaScript-Dateien ein, die zu den entsprechenden Bibliotheken gehören. Beide HTML-Dokumente binden zudem die trackerlib.js ein, die verschiedene Funktionen bietet, die relevant für die Datenspeicherung, wie bspw. die Erstellung der Fotos, sind. Die trackerlib.js sendet dann diese Daten über eine POST-Anfrage an die upload.php, damit die Bilder, Blickpunkte, Emotionen usw. in die richtigen Tabellen in der Datenbank eingeragen werden. Über GET-Anfragen, werden die Anleitungen für die verschiedenen Varianten erfragt und in die entspechenden HTML-Dokumente eingetragen.
Von der index.html kann der User auch an die delete.html gesendet werden, in der er seine ID angeben muss, wenn der Nuter den Wunsch hat, seine bisherigen gespeicherten Daten anzuschauen und eventuell zu löschen. Ist dies der Fall, so wird die eingetragene ID über eine POST-Anfrage an die showData.php gesendet und dem User werden seine Bilder und Daten angezeigt. Von der delete.php wird der User dann zurück zur index.html geleitet, sobald alles gelöscht ist.

3.5. Server und Datenbank

Der Server mit der dazugehörigen Datenbank wurde von Roufayda Salaheddine im Rahmen iherer Bachelorarbeit aufgesetzt. Um eine Datenbank aufzusetzen, braucht es zuerst eienn Webserver. Dieser wurde mit Xampp aufgesetzt. Im Folgenden wird kurz erklärt, wie die Datenbank aufgebaut ist.
Die Datenbank, die erstellt wurde, beruht auf dem relationalen Datenbankmodell und deshalb gibt es mehrere verschiedene Tabellen, die miteinander in Relation stehen.
Unter Berücksichtigung des Entity-Relationsship-Modells ergibt sich eine Datenbank "eyetracker", die sich aus sechs verschiedenen Tabellen zusammensetzt (Figure 3.14.).
Diese Tabellen speichern entweder Daten, die von einem Studienteilnehmer erfasst werden oder sie beinhalten Daten, die aus der Datenbank ausgelesen werden, um sie dem Nutzer zu zeigen.
So speichern die beiden Tabellen *text* und *emotions* Daten, die später von der Webseite aufgerufen und dem User angezeigt werden. Beispielsweise speichert *text* den Text für den Datenschutz, der ganz zu Anfang dem User in einem Dialogfenster angezeigt wird.
Im Gegensatz dazu speichern die Tabellen *proband*, *pbImage*, *pbEmotion* und *pbViewingPos* die Daten, die von dem Nutzer erfasst werden. So wird in der *proband*-Tabelle zum Beispiel der Name und das Alter eines Studienteilnehmers gespeichert. In *pbEmotion* werden zusätzlich zu den aufgenommenen Bilder von der Emotionserkennung auch noch die erkannte und vorgegebene Emotion(beides als Integer) gespeichert. In der *pbViewingPos* werden ebenfalls die aufgenommenen Bilder, diesmal jedoch von dem Eyetracker, gespeichert. Hier wird auch die Bildschirmauflösung und die vorhergesagte Blickposition gespeichert, um später beurteilen zu können, wie genau der Eyetracker sich verhält.

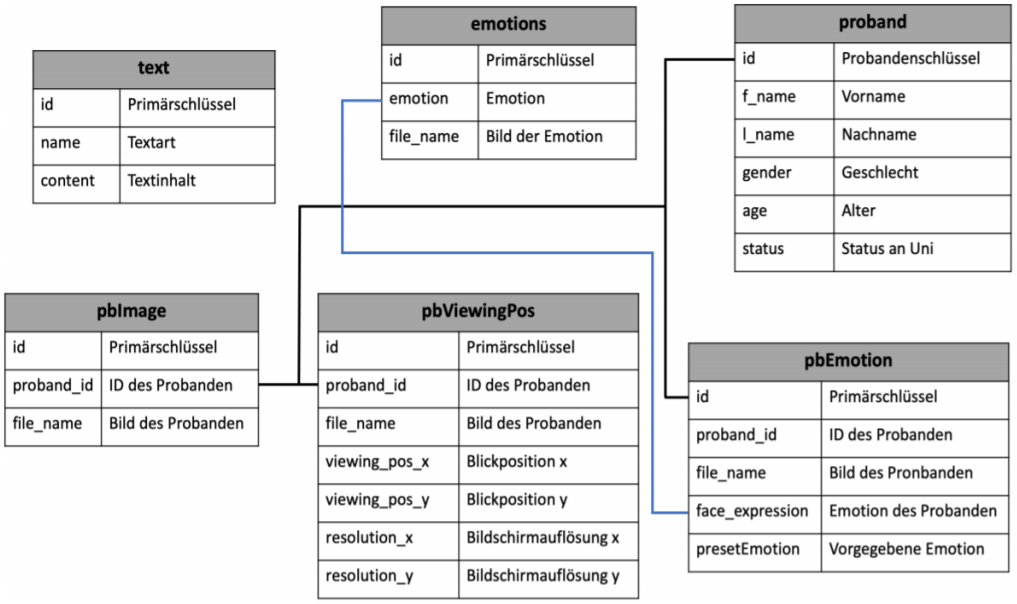

Figure 3.14.: Die Tabellen in der Datenbank mit den entsprechenden Parametern

4. Evaluation

In diesem Kapitel wird die Genauigkeit des Eyetrackers und der Emotionserkennung begutachtet und die Laufzeit des Frameworks evaluiert.
Um die Genauigkeit des Eyetrackers und der Emotionserkennung zu testen, wurden mehrere Bilder und Daten aufgenommen und dann mit den gewünschten Werten verglichen.

4.1. Eyetracker

Im Falle des Eyetrackers wurden ca. 25 Bilder gemacht und mithilfe der erkannten Blickposition und den gespiecherten Bildschirmauflösung konnte berechnet werden, wie genau der Punkt vom User angeschaut wurde. Der Punkt befindet sich in der Mitte des Bildschirms, dafür muss von der x-Achsen-Bildschirmauflösung und y-Achsen-Bildschirmauflösung jeweils die Hälfte genommen werden, das ist dann die Position der schwarzen Punktes. Diese Daten wurden dann mit den erkannten Blickpunkt des Nutzers verglichen und lieferte folgendes Ergebnis:
Vor allem ist aufgefallen, dass die Blickpunktvohersage fast immer links vom Mauszeiger ist. Das liegt vermutlich daran, dass die Vorhersage unter der Annahme getroffen wird, dass die Augen immer dem Mauszeiger von links nach rechts folgen. Zudem ist aufgefallen, dass die Vorhersage so gut wie nie direkt auf dem Mauszeiger war, auch wenn dieser direkt angeschaut wurde. Die 25 Mal, als die Blickposition erfasst wurde, führte zu dem Ergebnis, dass der Blickpunkt meist um einige wenige Pixel vom schwarzen Punkt abweicht. Eine Abweichung von ca 20 bis 50 Pixel wurde in den meisten Fällen aufgezeichnet. Um diese Abweichung zu erhalten, wurde verglichen, bei welchen Pixeln die Bildschirmmitte liegt und welche Blickposition vorhergesagt wurde. Diese wurde jeweils für die x- und die y-Achse gemacht. Auch wichtig zu bemerken ist, dass je mehr Bilder gemacht wurden, desto genauer wurde die Blickpunktvorhersage, was an der Selsbt-Kalibration des Models liegt, das heißt, es lernt während es verwendet wird und funktioniert deshalb später besser als am Anfang. In der folgenden Abbildung ist nochmal exemplarisch dargestellt, wo der Blickpunkt meist vorhergesagt wurde (Figure 4.1.).

Dieses Ergebnis ist ziemlich zufriedenstellend, unter Betrachtung der Tatsache, dass Eyetracking normalerweise spezielle Hardware benötigt, um zu funktionieren. Dieses komplett clientseitige Modell sagt die Blickposition ziemlich genau vor, auch

Figure 4.1.: Beispielhafte Darstellung, wo der Blickpunkt oftmals vorhergesagt wird im Vergleich zum anzuschauenden Punkt

wenn es trotzdem Abweichungen gibt. Diese können jedoch außer Acht gelassen werden, da es alles in allem doch sehr gut funktioniert und nicht etwas ganz falsches vorhersagt. Mit der Erweiterung der Broswerfunktionen und der Software ist es in der zukunft bestimmt möglich, es noch genauer zu messen.

4.2. Emotionserkennung

Auch für die Emotionserkennung wurden ungefähr 25 Bilder aufgenommen. Diesmal wurde jedoch verglichen, ob die vorgegebene Emotion mit der erkannten übereinstimmt oder nicht, deshalb wird sowohl die vorgegebene als auch die erkannte Emotion gespeichert. Dies hat zu folgenden Ergebnissen geführt:
Die Emotionen Happy und Surprise wurden in allen Fällen erkannt und nie falsch betitelt. Die Emotion Fear ist schwieriger zu erkennen, da es öfter mit Surprise verwechselt wird vom Computer, in ungefähr der Hälfte der Fälle wurde es richtig erkannt. Sad wurde oft nur schwierig erkannt, in vielen Fällen wurde es als Neutral bewertet statt als Sad. Und Neutral wird auch nur in der Hälfte der Fälle erkannt, da es immer wieder mit Happy oder Sad verwechselt wird. Dies führt zu dem Ergebnis, dass in einem Drittel der Fälle die Emotion nicht richtig oder nur schwierig erkannt wird. Schwierig bedeutet in diesem Fall, dass es durchgehend zwischen zwei Emotionen schwankt. Manche Emotionen werden besser erkannt als andere.

Vor allem bei Surprise und Fear wundert das nicht, weil beide in der Regel mit aufgerissenen Augen und offenem Mund dargestellt werden, da müssen die Unterschiede schon sehr fein erkannt werden. Da es sich hier um eine rein clientseitige Anwendung handelt, darf auch davon ausgegangen werden, dass diese nicht so genau und gut läuft wie bei anderen Anwendungen, die nicht nur im Browser laufen und zusätzliche Hardware nutzen.

4.3. Laufzeit

Die Laufzeit der Funktionen wurde mit der JavaScript Funktion *performance.now()* gemessen. Das sieht im Falle der Funktion getData(), die verschiedene Texte wie bspw. den Datenschutz aus der Datenbank abfragt, so aus:

```
1  getData: function(){
2  var start = performance.now();
3   var oReq = new XMLHttpRequest(); // New request object
4      oReq.onload = function() {
5           var priv = document.getElementById("privacyTest");
6           priv.innerHTML = this.responseText;
7      };
8      oReq.open("get", "test.php", true);
9      oReq.send();
10
11   var oReq2 = new XMLHttpRequest(); // New request object
12     oReq2.onload = function() {
13          var anleitungX = document.getElementById("anleitung");
14          anleitungX.innerHTML = this.responseText;
15     };
16     oReq2.open("get", "tutorial.php", true);
17     oReq2.send();
18     var end = performance.now();
19     console.log("getData(): " + (end-start));
20   }
```

Pro Funktion wurden dann jeweils 20 Werte gesammelt und davon der Durchschnitt ausgerechnet. Für die Funktion getData() berechnete sich dadurch folgender Wert: 0,70ms. Das heißt, dass es im Schnitt ca. 0,7ms dauert, bis die Daten aus der Datenbank herausgelesen und dem Nutzer angezeigt wurden.

In der folgenden Tabelle ist die durchschnittliche Dauer einiger Funktionen festgehalten und unterscheidet sich auch je nach Variante (Tabelle 4.1).

Table 4.1.: Dauer in Millisekunden für unterschiedliche Funktionen.

Funktion	Einfaches Bild	Eyetracker	Emotiondetection
createImg()	47,78	190,77	14,70
fromPoint()	48,45	191,32	-
fromEmotion()	-	-	15,12
Generator()	-	-	0,22
generateDot()	-	0,26	-
eyeAnleitung()	0,48	0,48	-
getProbandenID()	0,66	0,66	0,66

Es ist genau zu erkennen, dass es wesentlich länger dauert ein Bild zu machen als beispielweise einen Punkt in die Mitte des Bildschirms zu malen, was auch nicht weiter verwunderlich ist, da das deutlich mehr Arbeit ist. Alles in allem sind die Laufzeiten aber ziemlich gering und werden von einem Nutzer meistens gar nicht bemerkt, sodass das keine negative Auswirkung auf die User Experience hat.

5. Diskussion

Der Eyetracker, der auf Basis der Webgazer.js-Bibliothek und eine FrontEnd-Emotionserkennung, die auf Basis der ChromeShapeAPI erstellt wurden, wurden verwendet, um ein webbasiertes Framework zu entwickeln, das es erlaubt Studien durchzuführen und die Daten zu speichern.

Das Framework funktioniert für beide Modelle zusammen nur im Chrome Browser (Version 86.0.4240.198), denn die Emotionserkennung ist nur für diesen Broswer ausgelegt. Der Eyetracker hingegen funktioniert auch in anderen Browsern wie Safari oder Firefox, da jedoch für die Emotionserkennung nur Chrome verwendet werden kann, wird empfohlen für beide Modelle direkt Chrome zu nutzen.
Die Emotiondetection basiert auf einer von drei verschiedenen Bibliotheken von Kevin Hsiao (kevinsbest auf GitHub). Verwendet wurde die MobileNetWebcam, da MobileNetImage nur Emotionen auf einem hochgeladenen Bild erkennt und kein Videofeedback liefert und weil die TinyFaceDetectionWebcam, die eigentlich auf verschiedenen Browsern funktionieren sollte, einen Fehler bringt, der vom Autor auch nicht behoben wurde, nachdem darauf hingewiesen wurde. TinyFace war ursprünglich die favorisierte Wahl, da es auf verschiedenen Broswern und ohne extra Features funktionieren sollte. Die MobileNetWebcam hingegen funtkioniert schon erwähnt nur in Chrome und zuvor müssen die Experimental Features in Chrome aktiviert werden, da sonst die Chrome Shape Detection API nicht funktioniert. Zudem funktioniert die Emotionerkennung micht auf den neueren MacOs Versionen (Catalina), was die Verwendungsmöglichkeit davon stark reduziert. Zusätzlich produziert die Emotionserkennung machnal Fehler und erkennt plötzlich nur noch die Emotion "Wütend". Eine wirkliche Lösung konnte dafür nicht gefunden werden, es hat jedoch geholfen die dazugehörigen Dateien erneut hochzuladen. Ein weiteres Problem ist die Genauigkeit der Erkennung, oftmals kann das Modell beispielsweise nicht zwischen "Erschreckt/Angst" und "Überraschung" unterscheiden und wechselt oft zwischen den beiden EMotionen hin und her.
Die Eyetracker-Bibliothek hingegen funktioniert von Anfang an sehr gut und lässt sich auch ohne Probleme verwenden. Es ist schnell und leicht möglich, die richtigen Funktionen zu nutzen, die nötig sind, um die Blickpunkt Erkennung zu starten. Auch ein großer Vorteil ist es, dass es möglich ist, den Blickpunkt dauerhaft oder nur für einen bestimmten Moment anzufragen, denn für diese Arbeit war immer nur der Blickpunkt zum Zeitpunkt der Erstellung des Fotos nötig. Der Eyetracker funktioniert nur lokal oder über eine https-Verbindung, dies hat die Arbeit mit Roufayda Salaheddine erschwert, da ein gemeinsamer Webserver benötigt wurde,

damit sowohl sie als auch ich Zugriff auf die Webseiten haben, jedoch gab es online keinen , der ohne weitere Kosten https unterstützt. Auch hat es nicht geklappt, einen ftp-Zugang zu Roufayda Salaheddines Xampp Server aufzubauen, sodass sie die Zusammenarbeit ziemlich umständlich gestaltet hat, da immer nur an einem Computer gearbeitet werden konnte.
Auch zu kritisieren ist die Tatsache, dass der Cache ziemlich schnell gefüllt war und somit oft geleert werden musste, da sonst Änderungen im Code nicht mehr angenommen wurde und Chrome weiterhin eine alte Version des Codes verwendet hat. Die hat dazu geführt, das länger als nötig nach Lösungen gesucht wurde als nötig.
Doch all diese Hürden wurden schlussendlich überwunden und sowohl der Eyetracker als auch die Emotionserkennung können abhängig vom Computer getestet werden. Und auch wenn der Eyetracker besser funktioniert und auf verschiedenen Computern funktioniert, so ist die Emotionserkennung beeindruckender, wenn die Personen befragt werden, die es beide Modelle ausgetestet haben.

A. Zusammenfassung

Ziel dieser Arbeit war es ein webbasiertes Framework zur Dürchführung von Studien zu erstellen und zu bewerten. Zusammenfassend lässt sich sagen, dass das Ziel grundsätzlich mit einigen Stolpersteinen erreicht werden konnte. Die Implementierung des Frameworks hat sich an manchen Stellen als schwieriger herausgestellt als erwartet. Die drei Bibliotheken, die Kevin Hsiao von kevinisbest auf github, bereitstellt für die Frontend- Emotiondetection, sind nicht alle dafür geeignet gewesen, in dieses Framework eingebunden zu werden. Während die eine Bibliothek nur zuließ, Bilder hochzuladen, statt Live-Videofeedback und die andere durchgehend einen Fehler anzeigt, erwies sie die MobildeNetWebcam auf Basis der Chrome Shape Detection API als funktionsfähig, jedoch nur unter einigen Bedingungen, sodass die Einsatzfähigkeit dieser sich stark einschränkte. Zum einen funktioniert es nur im Chromebrowser, zum anderen erfordert es zusätzlich dazu, dass die Experimental Webplatform Features in Chrome aktiviert werden. Außerdem funktioniert es nicht für neuere MacOs Versionen.
Der Eyetracker, der zuerst an der Brown Universität entwickelt wurde und auf der webgazer.js Bibliothek basiert, lieferte keine solcher Probleme. Dies ließ sich gut integrieren und lief auch schnell und problemlos.
Auch wenn auf meinem eigenen Computerw alles am Ende problemlos lief, so zeigten sich einige Probleme, als Roufayda Salaheddine es auf ihrem Webserver laufen lassen wollte. Dadurch konnte in Erfahrung gebracht werden, dass die Emotionserkennung nicht auf neuren MacOs Versionen läuft. Nachdem die Hürden übersprungen waren, konnte das Framework getestet und bewertet werden.
In den meisten Fällen merkt der Nutzer nichts von den Wartezeiten, nur wenn die Emotionserkennung zum ersten mal aufgerufen wird, dauert es unangenehm lang, bis das Modell starten kann. Das ist nicht wünschenswert, denn der User kann das Interesse verlieren, wenn die Wartezeit zu lang ist.
Die Evaluation hat gezeigt, dass der Eyetracker relativ zuverlässig und genau funktioniert, da die Abweichung von der vorhergesagt Blickpositon nie weit entfernt von dem anzuschauenden Punkt war und sich nur um einige Pixel unterschied. Dies ist ein ziemlich befriedigendes Ergebnis, da das heißt, dass der Eyetracker und auch die webgazer.js Bibliothek guten Gewissens genutzt werden können. Da bekannt ist, dass die Blickposition immer etwas weiter links vom Mauszeiger vorhergesagt wird, ist auch klar, warum der Blick nie genau auf den Koordinaten des anzuschauenden Punktes liegen wird.
Dahingegen ist die Emotionserkennung etwas weniger zuverlässig, denn in einem Drittel der Testfälle wurde die Emotion falsch erkannt. Das liegt auch daran, dass die

Software einfach nicht in der Lage ist, alle Gesichtszüge so perfekt zu interpretieren, um freine Unterschiede zu erkennen, die zwei verschiedene Emotionen voneinander zu differenzieren. Ein gutes Beispiel ist hierfür Angst Und Überraschung: Beides wird mit weit geöffneten Augen und aufgerissenem Mund symbolisiert, die Unterschiede dabei werden von dieser Software nur mit großen Schwieirgkeiten erkannt, unmöglich ist es jedoch nicht.
Diese Ergebnisse führen jedoch zu der Schlussfolgerung, dass der Eyetracker besser und zuverlässiger funktioniert als die Emotionserkennung. Damit ist auch die Frage vom Anfang geklärt, ob clientseitige Anwendungen in diesen Bereichen genauso gut funktionieren können, wie Anwendungen, die zusätzlich noch bestimmte Hardware erfordern. Nämlich: Der Eyetracker funktioniert unter diesen Umständen sehr gut und die Emotionserkennung ist leider nur befriedigend, schon allein weil die Implementation und Verwendung äußert umständlich ist. Doch die Datenerfassung hat keine solche Probleme bereitet. Somit ist zumindest dieser Teil gut umgesetzt worden und damit sind die beiden Modelle auch für Studien nutzbar.

Bibliography

[Con05a] Mozilla Contributors. Html: Hypertext markup language. https://developer.mozilla.org/de/docs/Web/HTML, 2005. Accessed on 21.10.2020.

[Con05b] Mozilla Contributors. Was ist javascript? https://developer.mozilla.org/de/docs/Learn/JavaScript/First_steps/Was_ist_JavaScript, 2005. Accessed on 19.10.2020.

[Con19] Mozilla Contributors. Einführung. https://developer.mozilla.org/de/docs/Web/JavaScript/Guide/EinfÃijhrung, 2019. Accessed on 15.10.2020.

[Ebe18] Julian Eberling. Wo eye tracking als schnittstelle zum einsatz kommt. https://www.lead-innovation.com/blog/eye-tracking-als-schnittstelle, Nov 2018. Accessed on 04.12.2020.

[ecm] Draft ecma-262 / december 3, 2020. https://tc39.es/ecma262/. Accessed on 14.10.2020.

[ES19] Cristhian Gabriel Espinosa Sandoval. Multiple face detection and recognition system design applying deep learning in web browsers using javascript. 2019.

[FBH+19] Wolfgang Fuhl, Efe Bozkir, Benedikt Hosp, Nora Castner, David Geisler, Thiago C., and Enkelejda Kasneci. Encodji: Encoding gaze data into emoji space for an amusing scanpath classification approach ;). In *Eye Tracking Research and Applications*, 2019.

[FBK20] Wolfgang Fuhl, Efe Bozkir, and Enkelejda Kasneci. Reinforcement learning for the privacy preservation and manipulation of eye tracking data. *arXiv preprint arXiv:2002.06806*, 08 2020.

[FCK18a] W. Fuhl, N. Castner, and E. Kasneci. Histogram of oriented velocities for eye movement detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.

[FCK18b] W. Fuhl, N. Castner, and E. Kasneci. Rule based learning for eye movement type detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.

[FCK+19] W. Fuhl, N. Castner, T. C. Kübler, A. Lotz, W. Rosenstiel, and E. Kasneci. Ferns for area of interest free scanpath classification. In *Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications (ETRA)*, 06 2019.

[FCZ+18] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, W. Rosenstiel, and E. Kasneci. Mam: Transfer learning for fully automatic video annotation and specialized detector creation. In *International Conference on Computer Vision Workshops, ICCVW*, 2018.

[FEH+18] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci. Bore: Boosted-oriented edge optimization for robust, real time remote pupil center detection. In *Eye Tracking Research and Applications, ETRA*, 2018.

[FGK20a] W. Fuhl, H. Gao, and E. Kasneci. Neural networks for optical vector and eye ball parameter estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[FGK20b] W. Fuhl, H. Gao, and E. Kasneci. Tiny convolution, decision tree, and binary neuronal networks for robust and real time pupil outline estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[FGRK19] W. Fuhl, D. Geisler, W. Rosenstiel, and E. Kasneci. The applicability of cycle gans for pupil and eyelid segmentation, data generation and image refinement. In *International Conference on Computer Vision Workshops, ICCVW*, 11 2019.

[FGS+18] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci. Cbf:circular binary features for robust and real-time pupil center detection. In *ACM Symposium on Eye Tracking Research & Applications*, 06 2018.

[FK18] W. Fuhl and E. Kasneci. Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools. In *Poster at Egocentric Perception, Interaction and Computing, EPIC*, 2018.

[FK19] W. Fuhl and E. Kasneci. Learning to validate the quality of detected landmarks. In *International Conference on Machine Vision, ICMV*, 11 2019.

[FK20a] Wolfgang Fuhl and Enkelejda Kasneci. Rotated ring, radial and depth wise separable radial convolutions. *arXiv*, 08 2020.

[FK20b] Wolfgang Fuhl and Enkelejda Kasneci. Weight and gradient centralization in deep neural networks. *arXiv*, 08 2020.

[FKB+18] W. Fuhl, T. C. Kübler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci. Region of interest generation algorithms for eye tracking data.

In *Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA*, 06 2018.

[FKH+17] W. Fuhl, T. C. Kübler, D. Hospach, O. Bringmann, W. Rosenstiel, and E. Kasneci. Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection. *Journal of Eye Movement Research*, 10(3), 05 2017.

[FKRK20] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci. Training decision trees as replacement for convolution layers. In *Conference on Artificial Intelligence, AAAI*, 02 2020.

[FKS+15] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Arbitrarily shaped areas of interest based on gaze density gradient. In *European Conference on Eye Movements, ECEM 2015*, 08 2015.

[FKSK18] W. Fuhl, T. Kübler, T. Santini, and E. Kasneci. Automatic generation of saliency-based areas of interest. In *Symposium on Vision, Modeling and Visualization (VMV)*, 09 2018.

[FRE20] Wolfgang Fuhl, Yao Rong, and Kasneci Enkelejda. Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction. In *Proceedings of the International Conference on Pattern Recognition*, pages 0–0, 2020.

[FRK19] W. Fuhl, W. Rosenstiel, and E. Kasneci. 500,000 images closer to eyelid and pupil segmentation. In *Computer Analysis of Images and Patterns, CAIP*, 11 2019.

[FRM+20] Wolfgang Fuhl, Yao Rong, Thomas Motz, Michael Scheidt, Andreas Hartel, Andreas Koch, and Enkelejda Kasneci. Explainable online validation of machine learning models for practical applications. In *Proceedings of the International Conference on Pattern Recognition*, pages 0–0, 2020.

[FSG+16] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, W. Rosenstiel, and E. Kasneci. Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct publication – PETMEI 2016*, 09 2016.

[FSG+17] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, and E. Kasneci. Eyelad: Remote eye tracking image labeling tool. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.

[FSK17a] W. Fuhl, T. Santini, and E. Kasneci. Fast and robust eyelid outline and aperture detection in real-world scenarios. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*, 03 2017.

[FSK17b] W. Fuhl, T. Santini, and E. Kasneci. Fast camera focus estimation for gaze-based focus control. In *CoRR*, 2017.

[FSK+18] W. Fuhl, T. Santini, T. Kuebler, N. Castner, W. Rosenstiel, and E. Kasneci. Eye movement simulation and detector creation to reduce laborious parameter adjustments. *arXiv preprint arXiv:1804.00970*, 2018.

[FSR+16] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci. Non-intrusive practitioner pupil detection for unmodified microscope oculars. *Elsevier Computers in Biology and Medicine*, 79:36–44, 12 2016.

[Fuh19] W. Fuhl. *Image-based extraction of eye features for robust eye tracking*. PhD thesis, University of Tübingen, 04 2019.

[Fuh20] Wolfgang Fuhl. From perception to action using observed actions to learn gestures. *User Modeling and User-Adapted Interaction*, pages 1–18, 08 2020.

[GFSK17] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci. Saliency sandbox: Bottom-up saliency framework. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.

[GG15] Vasavi Gajarla and Aditi Gupta. Emotion detection and sentiment analysis of images. *Georgia Institute of Technology*, 2015.

[gui19] Gui (graphical user interface). `https://www.itwissen.info/GUI-graphical-user-interface-Grafische-Benutzeroberflaeche.html`, Sep 2019. Accessend on 06.12.2020.

[HNA+11] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011.

[Hsi18] Kevin Hsiao. kevinisbest/frontend-emotiondetection. `https://github.com/kevinisbest/FrontEnd-EmotionDetection`, Oct 2018. Accessed on 22.10-2020.

[htm] Html. `https://wiki.selfhtml.org/wiki/HTML`. Accessed on 20.10.2020.

[PN09] Kara Pernice and Jakob Nielsen. How to conduct eyetracking studies. *Nielsen Norman Group*, page 945397498, 2009.

[pro] 11 alternatives to webgazer.js. `https://www.producthunt.com/alternatives/webgazer-js`. Accessed on 01.12.2020.

[PSL+16] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*, 2016.

[sel20] Javascript/ajax. https://wiki.selfhtml.org/wiki/JavaScript/Ajax, 2020. Accessed on 18.11.2020.

[SSLG04] Yafei Sun, Nicu Sebe, Michael S Lew, and Theo Gevers. Authentic emotion detection in real-time video. In *International Workshop on Computer Vision in Human-Computer Interaction*, pages 94–104. Springer, 2004.

[Ste19] Thomas Steiner. The shape detection api: a picture is worth a thousand words, faces, and barcodes. https://web.dev/shape-detection/, Jan 2019. Accessed on 22.10.2020.

[Sti20] Wolfgang Stieler, Christian und Honey. Expertenstreit über emotionserkennung durch ki. https://www.heise.de/newsticker/meldung/Expertenstreit-ueber-Emotionserkennung-durch-KI-4667496.html, Feb 2020. Accessed on 01.12.2020.

[w3aa] Ajax - the xmlhttprequest object. https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp. Accessed on 17.11.2020.

[w3ab] Ajax introduction. https://www.w3schools.com/xml/ajax_intro.asp. Accessed on 17.11.2020.

[web] webgazer.js: democratizing webcam eye tracking on the browser 2020. https://webgazer.cs.brown.edu/. Accessed on 03.12.2020.

[wik20a] Ajax (programmierung). https://de.wikipedia.org/wiki/Ajax_(Programmierung), Apr 2020. Accessed on 17.11.2020.

[wik20b] Grafische benutzeroberfläche. https://de.wikipedia.org/wiki/Grafische_Benutzeroberfl\%C3\%A4che, Dec 2020. Accessed on 05.12.2020.

[wik20c] Hypertext markup language. https://de.wikipedia.org/wiki/Hypertext_Markup_Language, Nov 2020. Accessed on 20.10.2020.

[wik20d] Javascript. https://de.wikipedia.org/wiki/JavaScript, Jul 2020. Accessed on 19.10.2020.