

Fakultät **Sprach- , Literatur- und Kulturwissenschaften**

**Institut für Information und** Medien**, Sprache und Kultur (I:IMSK)  
Lehrstuhl für Medieninformatik**

Praxisseminar Master Medieninformatik

MEI-M 26.1 (M. Sc.)

SS 2018

Leitung: Florin Schwappach

Usability Testing Plattform – OBS Plugin

Dominik Deller

Matr.-Nr: 1679917

3. Semester M.Sc. Medieninformatik

Email: dominik.deller@stud.uni-regensburg.de

Julia Grötsch

Matr.-Nr: 1673540

3. Semester M.Sc. Medieninformatik

Email: julia.groetsch@stud.uni-regensburg.de

Khang Ho

Matr.-Nr: 1686639

3. Semester M.Sc. Medieninformatik

Email: khang.ho@stud.uni-regensburg.de

Philipp Weber

Matr.-Nr: 1699855

3. Semester M.Sc. Medieninformatik

Email: philipp1.weber@stud.uni-regensburg.de

Abgegeben am 07.10.2018

Inhalt

1. Einleitung 4
2. Angaben zu Ressourcen und Dokumenten 5
3. Problemstellung und Projektbeschreibung 5
4. Projektmanagement 6
   1. Agiles Projektmanagement 6
   2. Projektmanagement-Tool 6
      1. Taiga 7
      2. Asana und Instagantt 7
      3. Projektplan 8
      4. Codeverwaltung 9
   3. Meetings 10
      1. Meetings mit Kursleiter und Stakeholder 10
      2. Sprints 13
5. Architektur und Implementierung 16
   1. Anforderungsanalyse 16
      1. Contextual Inquiry 17
      2. User Stories 17
      3. Mock-Ups 19
   2. Implementierung 22
      1. Funktionalitäten 23
      2. Systemarchitektur 27
   3. Spezifikation 28
6. Evaluation 28
   1. Testing mit Mitkommilitonen 28
   2. Evaluation mit Stakeholder 30
7. Setup-Anweisungen 31
8. Ausblick 31

Abbildungen

[Abbildung 1 - Taiga Übersicht](#bookmark) 7

[Abbildung 2 - Anforderungsanalyse](#bookmark1) 8

[Abbildung 3 – Projektplanung](#bookmark2) 8

[Abbildung 4 – Projektdurchführung](#bookmark3) 9

[Abbildung 5 – Design](#bookmark4) 9

[Abbildung 6 – Evaluation](#bookmark5) 9

[Abbildung 7 - Sonstige Miletones](#bookmark6) 9

[Abbildung 8 - Projektplanübersicht](#bookmark7) 10

[Abbildung 9: Hauptseite](#bookmark8) 20

[Abbildung 10 – Testleiter Ansicht im Mockup](#bookmark9) 21

[Abbildung 11 – Testpersonen Ansicht im Mockup](#bookmark10) 22

[Abbildung 12 – Export Seite im Mockup](#bookmark11) 23

[Abbildung 13 – Hauptseite der Anwendung](#bookmark12) 24

[Abbildung 14 – Tutorial Seite der Anwendung](#bookmark13) 25

[Abbildung 15 – Testleiter Ansicht der Anwendung](#bookmark14) 25

[Abbildung 16 Testmanager – Einen bestehenden Test wählen](#bookmark15) 26

[Abbildung 17 Testmanager – Einen neuen Test anlegen](#bookmark16) 27

[Abbildung 18 Testmanager – Tasks](#bookmark17) 27

[Abbildung 19 Testmanager – Einen freien Test starten](#bookmark18) 28

[Abbildung 20 – Taskansicht auf Testperson Seite](#bookmark19) 28

# Einleitung

Evaluierungen sowie Usability Testing sind ein zentraler Bestandteil des Softwareengineering Prozesses, weshalb Remote-Testing ein sehr wichtiger Aspekt ist, da nicht jeder Usability Test vor Ort absolviert werden kann. Für dieses Szenario existiert noch keine Open Source Software an der Universität Regensburg, weshalb eine Implementierung einer solchen Software erwünscht war. Aufgabe war es, eine Usability Testing Plattform zu gestalten, die es dem Nutzer ermöglicht, mit möglichst wenig Aufwand einen Usability Test zu absolvieren. Umgesetzt wurde im Rahmen des Projekts eine Homepage, die von Testleiter, sowie Proband abgerufen werden muss. Dabei unterscheiden sich die beiden Ansichten. De Testleiter stehen mehr Funktionalitäten zur Verfügung und er sieht zusätzlich den Bildschirm des Probanden, welcher durch die Open Broadcaster Software[[1]](#footnote-1) übertragen wird. Der Proband dagegen benötigt zwei Monitore, um einerseits die Tasks durchzuführen, die ihm im Rahmen des Tests gestellt werden, und auf der anderen Seite mit dem Testleiter zu kommunizieren. Die einzigen konkreten Kosten die bei diesem System bilden, wären die Kosten für einen Server, der aber evtl. auf dem Server des Lehrstuhls Medieninformatik laufen könnte. Die Software kann von jeder Person genutzt werden, es werden keinerlei Lizenzen oder sonstige Berechtigungen benötigt. Durch zusätzliche Skripte oder Plugins für OBS soll es zudem möglich sein, weitere Funktionalitäten hinzuzufügen.

# Angaben zu Ressourcen und Dokumenten

Um die Software zu benutzen, werden einige zusätzliche Elemente benötigt. Da es Ziel der Software ist, Tests zu ermöglichen, die nicht immer vor Ort ablaufen, sondern zum Teil auch über größere Distanzen, wird ein Server benötigt, um den OBS-Stream darauf zu hosten und die Kommunikation zwischen Testleiter und Proband zu ermöglichen. Testweise wurde dies während der Projektdurchführung auf der „Digital Ocean“-Cloud[[2]](#footnote-2) realisiert, zur realen Inbetriebnahme dieser Software müsste der Stream auf einen eigenen Server gehostet werden. Auf dem Server müssen PHP, … , … und installiert werden. Eine genaue Erläuterung zur Inbetriebnahme der Software sowie alle benötigten Schritte entnehmen Sie bitte dem beigefügten Dokument. Theoretisch wäre es möglich über einen Raspberry Pi zu streamen, doch dieser würde nach längerer Zeit überlasten, weshalb dies keine sinnvolle Alternative ist. (würde ich hier nicht mit reinnehmen)

# Problemstellung und Projektbeschreibung

Großer Wert bei der Umsetzung des Projekts wurde darauf gelegt, eine möglichst kostenfreie Usability Plattform zu gestalten, damit verschiedene Nutzergruppen diese Software nutzen können. Hierbei sollen grundlegende Elemente der Usability im Vordergrund stehen: Es soll möglich sein, eine Kommunikation zwischen Testleiter und Proband zu realisieren, sowie der Testperson Aufgaben zu stellen. Dies ist einerseits durch einen Chat realisiert, aber wir empfehlen für einen realen Usability Test Voice Chat zu benutzen, damit die Kommunikation schneller und effizienter ablaufen kann. Unsere Zielgruppe sind zunächst Studenten bzw. Dozenten, die ihre Tests Remote absolvieren wollen, aber auch andere Nutzergruppen sollten ohne Probleme die Anwendung nutzen können.

Es hat sich relativ schnell herausgefiltert, dass eine Implementierung der Anforderungen rein auf OBS Seite nicht möglich ist, daher wurde früh entschieden, die meisten Funktionalitäten auf einer Homepage auszulagern. Daher ist ein Server unbedingt nötig, um die Anwendungen zu nutzen. Das Projekt umfasst eine Implementierung der Anwendung auf Server Ebene. Hierbei existiert eine Webseite mit verschiedenen Ansichten, welche für die jeweiligen Teilnehmer (Testleiter sowie Proband) einige Funktionalitäten bietet.

# Projektmanagement

Das Projektmanagement ist ein wesentlicher Bestandteil des Softwareengineering Prozesses und ist vor allem für die organisatorische Ebene zuständig. Dabei ist es von besonderer Bedeutung, einen Zeitplan sowie einen Meilensteinplan zu erstellen, um den Fortschritt des Projektes zu gewährleisten und diesen auch einzuhalten, um weitere Kosten zu vermeiden. Dabei wurde eine leicht abgeänderte Form des agilen Projektmanagements benutzt.

## Agiles Projektmanagement

Als Projektmanagement Methode haben wir SCRUM gewählt (vgl. Projektmanagement: Definitionen, Einführungen und Vorlagen), wobei wir uns nicht strikt an die Vorgaben vom SCRUM Modell gehalten haben, sondern eine abweichende Form des SCRUMs angewendet haben, die besser an uns angepasst war. Wöchentliche Sprint-Meetings wurden gehalten, wobei diese teils persönlich und teils über das Internet (Teamspeak) absolviert worden sind.

Im laufe des Projektes wurde das Projektmanagement nach und nach an unsere Bedürfnisse angepasst. Zunächst haben wir uns in wöchentlichen Spring-Meetings getroffen, um den Fortschritt des Projektes zu gewährleisten. Dies wurde aber gegen Mitte der Projektphase zunehmend schwieriger, aufgrund von persönlichen Gründen. In dieser Zeit fanden die Meetings sehr unregelmäßig statt und es wurde meist nur über Teamspeak kommuniziert. Aus Erfahrung können wir sagen, dass das persönliche Treffen effektiver ist, da man nicht mit der Versuchung in Berührung kommt, etwas anderes zu machen. Teamspeak Meetings waren hingegen besser, wenn man sich nur kurz austauschen bzw. die Arbeitspakete untereinander austauschen wollte. Letztendlich ist eine Mischung aus beidem die beste Lösung, da die Arbeitspakete unterschiedlich viel Zeit beanspruchen. Da es sich bei unserer Methode um ein agiles Management handelt, war es uns möglich etwas flexibler zu handeln, weshalb auch die kurzzeitige Phase kein Problem war. Diese wurde durch intensivere Meeting Dichte gegen Ende unseres Projektprozesses kompensiert.

## Projektmanagement-Tool

Als Projektmanagement Tools werden meist externe Softwares bezeichnet, die zur Dokumentation des Projektes verwendet werden. Dabei werden Milestones sowie Arbeitspakete unterteilt, um eine bessere Übersicht zu bekommen und den Fortschritt des Projektes zu gewährleisten. Zunächst haben wir uns für das Tool „Taiga „(taiga.io) entschieden. Nach einiger Zeit haben wir uns umentschieden und sind auf die Plattformen Asana (asana.com) und Instagantt (instagantt.com) umgestiegen, da diese besser zu unseren Anforderungen gepasst haben.

### Taiga

Das Projektmanagement wurde zunächst auf Taiga durchgeführt, hier wurden die ersten Milestones und sämtliche Abläufe der bis zu diesem Zeitpunkt absolvierten Meetings dokumentiert. Im Laufe des Projektes haben wir jedoch festgestellt, dass besser angepasste Alternativen für unser Projekt existieren, weshalb wir schlussendlich zu Asana bzw. Instagantt gewechselt sind. Diese bieten die Möglichkeit aus dem Projektplan einen Milestone Plan bzw. ein Ganttdiagramm zu generieren, was uns den Aufwand erspart, einen eigenen zu erstellen. Dies war der Hauptgrund, wieso wir von Taiga zu Instagantt wechselt sind.

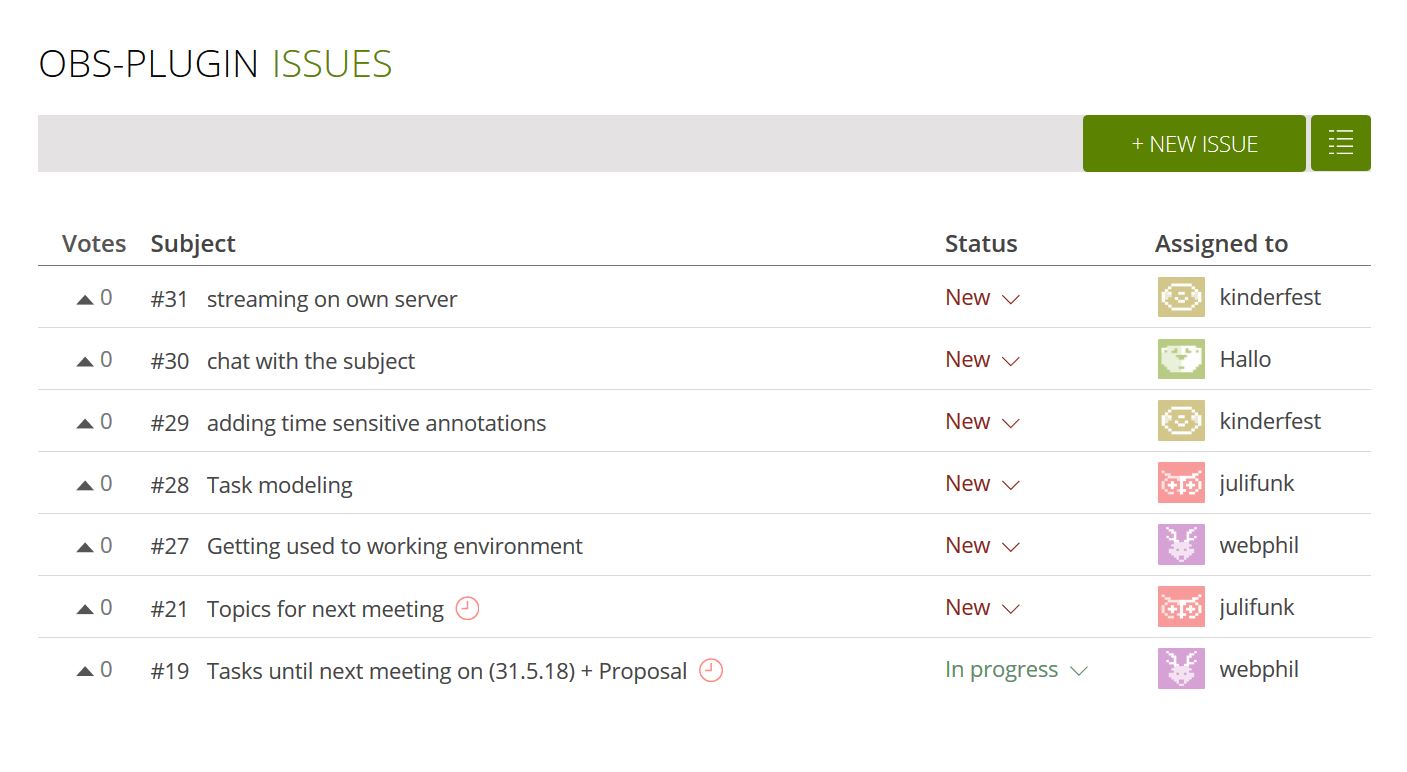


Abbildung 1 - Taiga Übersicht

### Asana und Instagantt

Auf Asana haben wir uns einen detaillierten Projektplan angelegt, welcher sehr strikt eingehalten, jedoch wurde es gegen Mitte des Projektes etwas nachlässiger, aufgrund von privaten Angelegenheiten der Mitglieder. Dies wurde aber im späteren Verlauf des Projektes zunehmend aufgefangen, um die Implementierung der Anwendung zu gewährleisten. Dies Geschah durch eine erhöhte Intensität sowie Dichte der Meetings innerhalb einer Woche. Der Projektplan dient hier als Orientierungsanker für das Projekt, dieser wurde zeitlich so aufgeteilt, dass genügend Zeit für die einzelnen Tasks eingeplant wurde und diese ohne Probleme zu bewältigen waren. Ebenfalls diente der Plan zur Kontrolle des Fortschritts des Projektes, es wurde stets darauf geachtet, dass nicht zu viele Arbeitspakete nicht rechtzeitig abgeschlossen werden, da es sonst zum „Stau“ kommt und den weiteren Arbeitsablauf beeinträchtigt.

### Projektplan

Der erste grobe Projektplan wurde auf Asana erstellt und immer weiter spezifiziert, hier wurden im Wesentlichen in fünf größere Abschnitte und zahlreiche, kleinere Arbeitspakete unterteilt, die im Folgenden genauer erklärt werden:

****

Abbildung 2 - Anforderungsanalyse

Die Anforderungsanalyse beinhaltet das Erheben von Daten für das Projekt. Hierbei ging es primär darum, mit dem Stakeholder zu reden, um die Anforderungen zu bekommen und die Machbarkeit des Projektes zu verifizieren.

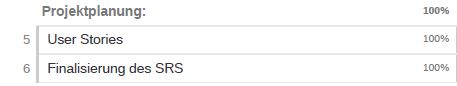


Abbildung 3 – Projektplanung

Hierbei ging es generell noch um die Anforderungen sowie die Abgabe des SRS. Dieser Schritt gehört noch zur Anforderungsanalyse.



Abbildung 4 – Projektdurchführung

Die Projektdurchführung war der längste Teil und zog sich von Mitte Juni bis einschließlich Mitte September durch. Hier wurden einige Änderungen an der Umsetzung vorgenommen, die im späteren Verlauf der Dokumentation erläutert werden.



Abbildung 5 – Design

Die Design Phase lief über den gesamten Ablauf des Projektes und wurde stetig angepasst. Aus den ersten MockUps entstand der erste Prototyp auf HTML-Basis, welcher dann erweitert wurde.



Abbildung 6 – Evaluation

Dies ist die letzte Phase des Projektes. Hier werden die letzten Feedbacks von den Endnutzern erhoben und im Endprodukt berücksichtigt.

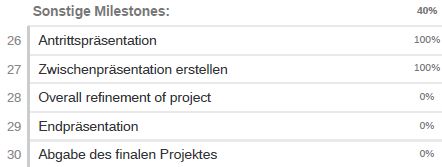


Abbildung 7 - Sonstige Miletones

Dies sind Arbeitspakete, die nicht zu den oben genannten Schritten zugeordnet werden können, weshalb diese extra kategorisiert wurden.

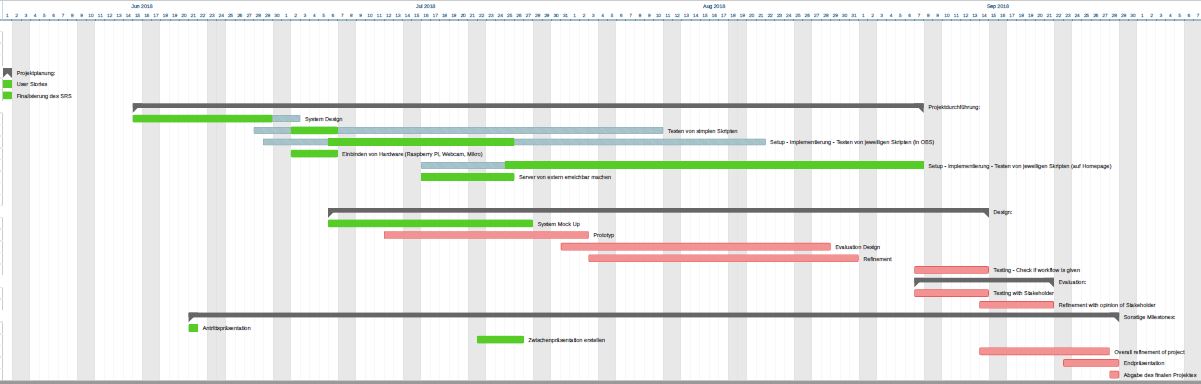


Abbildung 8 - Projektplanübersicht

### Codeverwaltung

Als Code Verwaltungsplattform wurde GitHub benutzt. In Git wurde das Projekt aktuell gehalten, damit jedes Teammitglied jederzeit an der Anwendung arbeiten konnte. Dafür wurden jeweils eigene Branches erstellt und am Ende wurden diese „gemerged“, um größere Probleme aus dem Weg zu gehen. Issues wurden hier intensiv gepflegt und es wurde explizit darauf geachtet, dass diese auch rechtzeitig erfüllt worden sind. Zusätzlich haben wir die externe Anwendung Sourcetree verwendet, die für uns etwas einfacher zu bedienen war.

## Meetings

Es wurden zahlreiche Meetings mit dem Stakeholder und Kursleiter ausgemacht, um den Stand der Dinge des Projektes zu klären.

### Meetings mit Kursleiter und Stakeholder

Die Meetings mit dem Stakeholder sowie Kursleiter wurde dazu genutzt, um die Machbarkeit der einzelnen Arbeitspakete sowie den Projektstand zu diskutieren. Es fanden zahlreiche Meetings mit dem Kursleiter sowie mit dem Stakeholder statt, welche in der TechBase stattgefunden haben.

#### Meeting 1 – 21.05.2018

Im ersten Meeting mit unserem Kursleiter haben wir allgemeine Informationen über das Projekt erhalten, es wurde spezifisch über die Rahmenbedingungen und Architektur der Software diskutiert. Da wir wenig bis gar keine Erfahrung mit C/C++ hatten, musste eine Alternative gefunden werden, da die meisten Skripte für OBS auf C/C++ Basis waren. Hier haben wir als Alternativlösung einige Möglichkeiten gefunden, die Skripte auch in Python einzubinden. Zusätzlich wurde uns hier erklärt, was sich der Kursleiter vorstellt und wir haben dem Kursleiter kurz unsere Ansicht vom Projekt erläutert.

* Python Skripte lassen sich in OBS einbinden
* Grobe Anforderungen des Kursleiters erhalten
* Fragen sammeln für Stakeholder für nächstes Meeting

#### Meeting 2 – 24.05.2018

In diesem Meeting haben wir unseren Stakeholder kennen gelernt und wir haben mit ihm darüber diskutiert, was er sich genau unter der Anwendung vorstellt. Dieses Gespräch lief als eine sehr freie Form des Contextual Inquiries (vgl. Kapitel 5.1.1) ab, der Stakeholder hat uns erklärt, inwiefern er die Applikation während der Arbeit benutzen würde. Der Stakeholder nannte uns hier die verschiedenen Grundfunktionalitäten, die er sich wünscht. Diese Informationen waren für uns sehr wichtig, darauf entstand Schluss endlich die Anforderungsliste bzw. die User Stories, die als Basis für unsere Anwendung gilt. Die Applikation soll als Open Source Software für alle zur Verfügung stehen, da momentan nur Morae als Alternative für Studentische Zwecke existiert und die Lizenz nicht sehr kostengünstig ist. Daher sollte auch der Aufbau des Codes so gestaltet werden, dass dieser von anderen Personen ohne Probleme erweitert werden können.

* Task modellieren und stellen 🡪 wird dem User im Interface angezeigt
* Bildschirm + Keylogs + Webcam + Mouse Events; Areas/Hotspots loggen
* Als Testleiter Timeline loggen; Metadaten eingeben
* Vgl. Morae 🡪 Kein Manager
* Secondary: Chat
* Machbarkeit des Projektes allgemein testen

#### Meeting 3 – 07.06.2018

Bei diesem Meeting hatten wir Zeit, uns über das Thema zu informieren und die Umsetzbarkeit des Themas zu diskutieren. Hier wurde die momentane Situation beschrieben und das Vorgehen, wie wir es uns vorgestellt haben dem Stakeholder erläutert. Hierbei wurden folgende Probleme mit dem Stakeholder diskutiert:

* Delay von Stream (sehr wichtig für einen Usability Test)
* Nur Zuschauer sieht kompletten Screen 🡪 Zweiter Monitor wird benötigt
* Für unsere Zwecke mit Twitch verbinden, für die tatsächliche Testumgebung wäre ein eigener Server besser (zunächst Raspberry Pi als Alternative)
* Evtl. einen „Wrapper“, der Skripte zusammenführt anstatt einzelner Skripte (Soll einen flüssigen Workflow geben)
* Workflow muss gegeben sein (Sehr wichtig!)

#### Meeting 4 – 05.07.2018

Das zentrale Thema dieses Meeting war die Besprechung des Projektplans, den wir auf Asana bzw. Instagantt erstellt haben und die groben Änderungen, die wir uns im Laufe des Projektes vorgenommen haben. Dafür haben wir uns mit unserem Kursleiter getroffen, um die Details des Vorgehens zu besprechen. Der Projektplan war im Allgemeinen in Ordnung, dennoch sollte er etwas genauer untergliedert sein, da die Arbeitspakete noch etwas zu groß waren. Der Plan an sich war schlüssig, es gab keine weiteren negativen Einwände des Kursleiters. Wichtig war es zudem, dass die einzelnen Arbeitspakete in dem Zeitraum abgeschlossen werden, die angegeben wurde. Zusätzlich wurde eine Zeiteinschätzung für die jeweiligen Pakete angegeben, um einen groben Überblick darüber zu geben, wie viel Zeit und Aufwand eingeschätzt werden muss, um ein Paket abzuschließen. Die tatsächlich gebrauchte Zeit variiert jedoch ein wenig, da es sich hierbei nur um Schätzungen handelt. Das zweite Thema der Besprechung war die Auslagerung mehrerer Funktionalitäten auf eine externe Webpage. Nach einigen Recherchen sind wir auf den Entschluss gekommen, dass es nicht möglich sei, alle Funktionalitäten als „Plug-Ins“ in OBS zu implementieren, da der Testleiter lediglich den Stream sieht und daher es nicht möglich wäre, den Stream selbst zu manipulieren. Folglich war der Plan, die Funktionalitäten auf einer externen von uns gehosteten Webseite auszulagern. Dies sollte zunächst auf einen Raspberry Pi geschehen.

* Projektplan genauer untergliedern
* Zeitschätzungen anpassen
* Projektplan abgeben
* Aufteilung der Funktionalitäten:
  + Webpage erstellen mit Funktionalitäten (Chat, Tasks, Video)
  + Server soll auf einem Raspberry laufen
  + Auf OBS Ebene: Stream und Keylogger

#### Meeting 5 – 19.07.2018

Dieses Treffen fand nur mit unserem Stakeholder statt und hatte als Ziel, den momentan Stand zu präsentieren sowie mögliche Änderungen zu diskutieren. Der Stakeholder wurde über unsere groben Änderungen informiert, dass wir einige Funktionen auslagern wollen und dass es verschiedene Ansichten der Webpage geben wird. Einerseits wird es eine Testleiter Ansicht und eine Probanden Ansicht geben. Diese unterscheiden sich von ein paar wenigen Funktionalitäten, da der Testleiter mehr Funktionen braucht. Diese beiden Ansichten sollen über einen Chat kommunizieren können. Dabei soll der Testleiter den Stream sehen, sowie Tasks erstellen und Annotationen hinzufügen können, während der Proband nur die Tasks einsehen und mit dem Testleiter über den Chat kommunizieren kann. Des Weiteren haben wir darüber diskutiert, inwiefern der Raspberry Pi auf längerer Sicht genutzt werden kann, da dieser Hardware technisch nur bedingt geeignet ist für eine längere Nutzung der Anwendung. Alternativ müsste ein Server gemietet werden, um das Streaming dort zu hosten.

#### Meeting 6 – 24.09.2018

Dieses Treffen war zur Evaluation der Anwendung im Alpha Stadium geplant. Wir präsentierten dem Stakeholder unsere Anwendung, dabei sollte er selbständig alle Funktionalitäten testen. Dies war unser letztes Treffen, weshalb wir möglichst viel Feedback von unserem Stakeholder bekommen wollten, um die Anwendung noch besser an die Anforderungen anzupassen. Den detaillierten Verlauf sowie die einzelnen Erkenntnisse können Sie im Kapitel 6.2 entnehmen.

### Sprints

Die wöchentlichen Sprints dienten der Gewährleistung des Projektfortschritts. Hierbei trafen wir uns entweder persönlich oder in einem Teamspeak und haben Arbeitspakete durchgeführt oder besprochen. Der allgemeine Aufbau der Sprints war wie folgt:

* Allgemeine Besprechung: Was wurde geschafft? Was wurde nicht geschafft?
* Momentan Stand des Projektes – Wurde der Projektplan eingehalten?
* Was sind die nächsten Schritte?
* Kurzes Sprint Review für alle Mitglieder
* Aufgaben Aufteilung bis zur nächsten Sitzung

Während des Projektes sind einige signifikante Änderungen vorgenommen worden, da manche Umsetzungen nicht so funktioniert haben, wie wir es uns vorgestellt haben. Eine komplette Auflistung der Sprints würde den Rahmen dieser Dokumentation sprengen, weshalb nur die signifikantesten Sprints aufgelistet werden, um unsere Entscheidungen besser nachvollziehen zu können.

#### Besprechung des Themas

Die ersten Besprechungen wurden geführt, um in das Thema einzuführen und uns über OBS bzw. den Anforderungen zu informieren. Wir haben uns hierbei zunächst einige Beispiel Skripte angeschaut, wie diese in OBS eingebunden werden. Die meisten Skripte basierten auf Lua bzw. C/C++, weshalb wir uns noch nicht sicher waren, ob wir dieses Projekt tatsächlich umsetzen wollen, da wir keinerlei Erfahrung mit den beiden Sprachen haben. Dennoch gab es einige Anzeichen, dass man auch über Python Skripte einfügen kann, was für uns ein Indiz war, dass dieses Thema eventuell doch umsetzbar ist.

#### Machbarkeit des Projektes

Um sicherzustellen, dass wir das Projekt auch abschließen können, mussten wir die Machbarkeit des Projektes überprüfen. Dafür mussten wir intensive Recherchen betreiben und einige Setups bzw. Skripte testen. Da wir die Anforderungen zuvor von unserem Kursleiter erhalten und nähere Spezifikationen vom Stakeholder mitgeteilt bekommen haben, konnten wir grob die Architektur der Anwendung erstellen und somit die gebrauchten Frameworks sowie Anwendungen beschaffen, um die ersten Tests durchzuführen. Diese Tests waren ausschlaggebend für uns, ob wir dieses Thema tatsächlich absolvieren oder uns ein anderes Thema gesucht hätten, da wir keinerlei Erfahrungen mit OBS oder dem Streamen selbst hatten.

#### Nacharbeit der Anforderungsanalyse

Aus den zuvor erhobenen Daten der Meetings konnten wir die ersten groben Anforderungen generieren und die ersten User Stories sowie Epics (Funktionen) in Taiga erstellen. Hier haben wir die Funktionen in reale Szenarien verpackt und als User Stories aufgestellt, um einen besseren Bezug zu der tatsächlichen Nutzung zu bekommen. Hierbei haben wir zwei Grundfunktionen gefiltert, die wir auf die Machbarkeit testen müssen, bevor wir das Projekt durchführen können. Dies war einerseits das Einbinden von Skripten auf Python Basis und andererseits das Hosten eines Streams. Diese beiden grundlegenden Funktionen müssen für uns umsetzbar sein, damit wir dieses Projekt abschließen können.

#### Skripteinbindung mit Python

Der erste Test war die Einbindung eines simplen Skripts (Countdown Zähler) in OBS. Das Skript selbst war in Python geschrieben, damit wir sicher gehen können, dass wir auch ohne Kenntnisse in Lua bzw. C Anwendungen schreiben können, die in OBS laufen. Anfangs hatten wir kleinere Probleme, da das Skript nicht erkannt wurde, aber durch längeres Probieren haben wir es schlussendlich geschafft, das Skript zum laufen zu bringen. Dies war für uns der Startschuss, da wir nun davon ausgingen konnten, dass wir das Projekt durchführen konnten. Dennoch musste noch ein zweiter Test gemacht werden, da wir den Stream noch auf einen Server hosten mussten, und wir auch dort noch keinerlei Erfahrung hatten, dennoch waren wir uns relativ sicher, dass wir das Projekt nun durchführen konnten.

#### Stream hosten via Raspberry Pi

Das Hosten des Streams war das nächste große Arbeitspaket, dass wir uns vorgenommen hatten. Der erste Gedanke war es, den Stream auf einen privaten Raspberry Pi zu konfigurieren und zu hosten. Hierzu haben wir den Raspberry Pi in einem unserer Netzwerke angeschlossen und konfiguriert. Nach anfänglichen Schwierigkeiten konnten wir den Raspberry Pi so konfigurieren, dass der Stream gehostet werden konnte. Der Stream konnte als Medienquelle in OBS gestreamt werden und auch von außerhalb des Netzwerkes erreicht werden. Doch nach einiger Zeit konnten Performance Probleme erkannt werden, da Streaming an sich selber relativ viel Rechenleistung beansprucht und ein Raspberry Pi auf längerer Sicht nicht dafür geeignet ist. Auch die Konfiguration über Ports bei dem Router war sehr komplex, weshalb eine andere Lösung angestrebt wurde.

#### Grober Projektplan erstellt

Nachdem wir die Grundvoraussetzungen des Projektes erfüllt haben und uns sicher waren, dass wir das Projekt auch abschließen können, haben wir unseren Projektplan erstellt, der auf Asana bzw. Instagantt erstellt wurde. Die Details zu diesem Plan sind im Kapitel 4.2.3 näher erläutert.

#### Erster Grundbauplan der Anwendung

Bei diesem Sprint haben wir uns über den Grundbauplan der Anwendung Gedanken gemacht. Erste Unterteilungen der Anwendung wurde schon bei der Erstellung des Projektplans gemacht und hier kam uns die Erkenntnis, dass es nicht möglich war, alle Funktionalitäten auf OBS Seite zu implementieren, da der Testleiter keinen Zugriff auf den Stream hat, sondern nur als Zuschauer fungiert. Deshalb haben wir uns darauf geeinigt, eine externe Webpage zu gestalten, um dort die Kommunikation zu ermöglichen. Die Kommunikation ist ein Schlüsselelement des Usability Testing, da Fragen gestellt werden können und Tasks übermittelt werden müssen. Folglich waren es keine Plugins an sich selbst mehr, sondern eine Auslagerung der Funktionen auf einer Webpage. Dies ermöglicht uns flexibler zu sein und durch eine externe Webpage können einfacher weitere Elemente hinzugefügt werden. Lediglich der Keylogger musste noch separat bearbeitet werden. Dies wurde von Kursleiter sowie Stakeholder akzeptiert. Die nächsten Schritte wurden geplant und die jeweiligen Arbeitspakete auf die Mitglieder verteilt.

#### Design der Webpage

Das Design der Webpage wurde anfangs als MockUps erstellt (vgl. Kapitel 5.1.3) und später genauer spezifiziert. Der erste Prototyp der Webpage wurde als Grundlage genutzt und im weiteren Verlauf des Projektes stetig angepasst und modifiziert. Die Funktionalitäten wurden hierbei noch nicht hinzugefügt, lediglich das Gerüst der Webpage wurde erstellt, um einen Groben Plan zu bekommen, welche Funktionalitäten wie funktionieren sollen.

#### Streaming hosten via RMTP

Bei den ersten Tests den Stream zu hosten, sind wir auf einige Probleme gestoßen. Der Raspberry Pi bzw. das Streamen selbst funktioniert nur auf RTMP Basis, da wir über den JW-Player streamen. Die bessere Methode wäre HLS …

#### Server Hosting anstatt Raspberry Pi

Nach längerer Diskussion haben wir uns dazu entschlossen, den Raspberry Pi aufgrund von Performance Problemen durch einen Server zu ersetzen. Hierbei haben wir uns für die Server Plattform „Digital Ocean“ entschieden. Nachdem das Streamen erfolgreich verknüpft wurde, wurden die ersten Funktionalitäten implementiert.

#### Taskmodellierung

Die Taskübermittlung war ebenfalls ein zentraler

#### Chat

#### Server Aufsetzen

#### Datenbankanbindung

#### Finalisierung der Anwendung

Nachdem wir unsere Anwendung dem Stakeholder präsentiert haben, haben wir die abschließenden Modifizierungen hinzugefügt. Es wurden sämtliche Anforderungen des Kursleiters sowie dem Stakeholder implementiert und auf die jeweiligen Wünsche des Stakeholders eingegangen. Da der Stakeholder keine größeren Mängel an unserer Anwendung bemerkt hatte und zufrieden mit dem Ergebnis war, konnten wir die Finalisierung der Anwendung relativ zügig abschließen und für die Abgabe vorbereiten. Hierzu wurden kleinere Bugs gefixt sowie Quellenangaben hinzugefügt und der Code aufgeräumt, damit dieser auch für andere Leute schlüssig ist.

# Architektur und Implementierung

Das Projekt untergliedert sich in wesentliche Teilblöcke, die sukzessiv abgearbeitet worden sind. Zunächst wurde die grundlegende Anforderungsanalyse durchgeführt, um einen Überblick über das Thema zu bekommen sowie weitere Informationen über Funktionalitäten sowie Anforderungen zu erheben. Daraus resultierten die ersten Mock-Ups sowie erste Annäherungen des Designs.

## Anforderungsanalyse

Die Anforderungsanalyse wurde relativ früh mit unserem Stakeholder durchgeführt. Dies geschah in insgesamt drei Meetings, da die jeweiligen Anforderungen einer Machbarkeitsstudie unterlaufen mussten und wir die Anforderungen näher spezifiziert haben. Die Funktionalitäten wurden dann durch User-Stories dargestellt, um die Funktionalitäten in reellen Situationen widerzuspiegeln.

### Contextual Inquiry

Die grundlegenden Anforderungen wurde vor den ersten Meetings schon von dem Kursleiter bereitgestellt, weshalb die ersten groben Spezifikationen der Systemarchitektur vorgenommen und vorgestellt werden konnten. In den ersten Meetings wurde uns das Vorhaben der Anwendung erläutert. Dies geschah durch eine relativ freie Form des Contextual Inquiries (Beyer & Holtzblatt, 1997, S.37f), bei welcher der Stakeholder kurz erklärte, wie die Anwendung im Alltag genutzt werde und uns so einen Überblick gegeben hat, welche Funktionalitäten wichtig sind. Auf Basis dessen entstanden so genannte User Stories, die im folgendem zusammengefasst werden.

### User Stories

Aus den vorher erhobenen Daten konnten User Stories erstellt werden, welche reale Szenarien darstellen. Diese dienen als Grundlage für die Anforderungsliste:

* As User I want to be able to see the subject's screen during the test in realtime
* As User I want to be able to save a recording of the subject's screen during the test
* As user I want to be able to model tasks to show them on the tester’s screen
* As user I want to be able to edit tasks
* As User I want to get keylogs of all activity of the subject during the test
* As User I want to record all mouse events during the test
* As User I want to be able to record the subject’s facial expressions during the test
* As User I want to be able to get an overview of hotspots/main areas of interest of the subject
* As User I want to be able to log a timeline for the test.
* As User I want to be able to add time-sensitive metadata to the recording of the test.
* As user I want to be able to have a live-chat with the subject during the test.

Aus den oben genannten User Stories sind folgende Anforderungen und dessen Prioritäten resultiert:

**Stream the subject’s screen**

Das Streamen des Bildschirmes der Testperson ist die Grundfunktion dieser Anwendung und hat somit die höchste Priorität, da ohne diese Funktion ein Remote-Test nicht stattfinden kann.

**Record mouse and key events**

Diese Funktion ist ein “nice to have“, da es den Workflow eines Tests nicht beeinflusst, sondern vor allem für die Nacharbeit eine wichtige Rolle spielt. Dadurch, dass ein Test auch ohne einen Keylogger funktionieren kann, hat diese Funktion nur eine mittlere Priorität.

**Chat with subject**

Kommunikation zwischen Testleiter sowie Testperson ist ein Schlüsselelement bei einem Usability Test, weshalb diese Anforderung eine sehr hohe Priorität hat.

**Task modelling**

Das Gestalten von Tasks sowie deren Bearbeitung hat ebenfalls eine sehr hohe Priorität, da es den Workflow beeinflusst. Wenn diese Funktion nicht einwandfrei funktioniert, ist es nicht möglich, einen effizienten Test durchzuführen, weshalb es sehr wichtig ist, dass diese Anforderung erfüllt wird.

**Add time-sensitive annotations**

Diese Anforderung hat eine mittlere Priorität, da es den Prozess des Tests nicht beeinflusst. Der Testleiter könnte theoretisch auch externe Notizen machen und diese für die Nacharbeit mit einbeziehen.

**Screen recording**

Diese Anforderung musste von uns nicht übernommen werden, da OBS eine Funktion hat, um den Stream zu recorden. Der Testleiter kann den Screen selbst recorden.

**Record the subject’s face**

Auch diese Anforderung wurde von OBS übernommen, die es ermöglicht eine Webcam als Medienquelle hinzuzufügen.

Diese sind die Hauptfunktionalitäten der Anwendung und dienen als Grundlage für die Anwendung.

### Mock-Ups

Aus den ersten Gesprächen mit dem Stakeholder haben wir Informationen über die Funktionalitäten der Anwendung sammeln können und die ersten Mock-Ups generiert. Diese dienten als Grundlage für die weitere Designspezifizierung und wurde nach und nach designtechnisch angepasst. Auch die verschiedenen Core-Funktionalitäten wurden abgebildet.

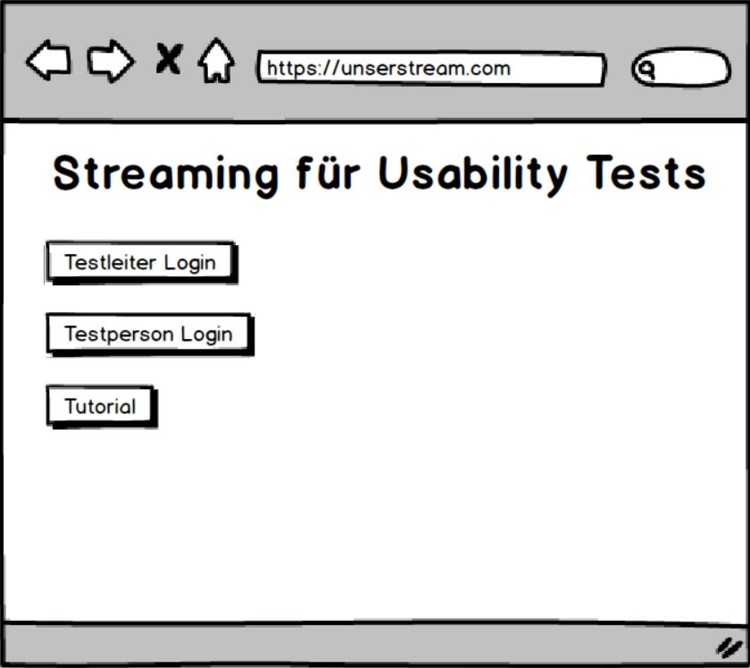


Abbildung 9: Hauptseite

In der obigen Abbildung sieht man den Hauptscreen der Webpage. Die Webpage untergliedert sich in 2 wesentlichen Screens, die im folgendem noch näher erläutert werden. Auf dem Hauptscreen kann man zwischen der Testleiterseite sowie Testpersonenseite auswählen. Zusätzlich soll noch ein kurzes Tutorial dem Nutzer die Applikation erklären.

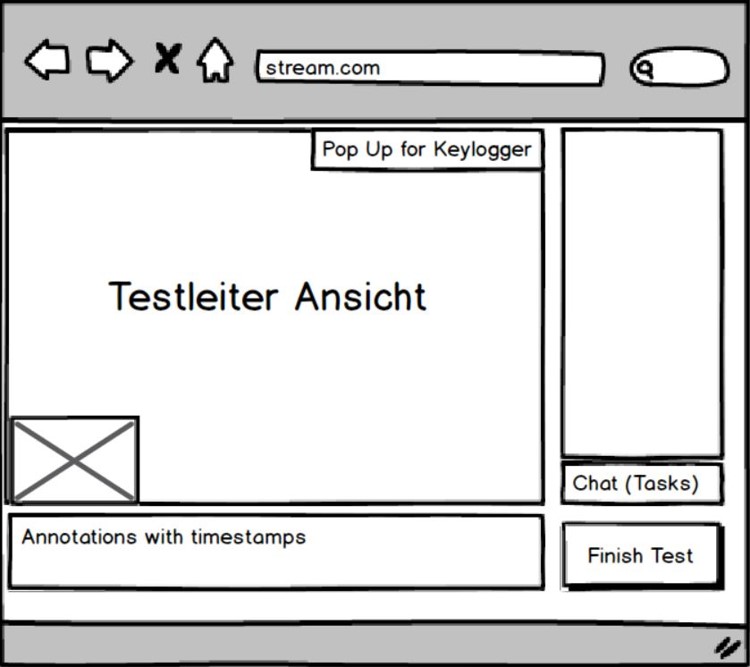


Abbildung 10 – Testleiter Ansicht im Mockup

Hier sieht man die Testleiter Ansicht der Webseite. Drei grundlegende Funktionalitäten sind hierbei auf der Seite abgebildet:

* Der Bildschirm des Probanden soll angezeigt werden, dabei nimmt dieser Teil den größten Abschnitt der Seite ein
* Es soll möglich sein, zeitsensitive Daten zu erfassen, dies kann der Leiter durch den „Annotationsabschnitt“ tätigen
* Der Testleiter braucht eine Möglichkeit, mit dem Probanden zu kommunizieren, hierbei ist der Chat zur Verfügung gestellt worden
* Der Tester soll einen Test abschließen können



Abbildung 11 – Testpersonen Ansicht im Mockup

Die Testpersonen Seite ist ähnlich der Seite des Testleiters aufgebaut. Einerseits soll die Testperson den Bildschirm streamen, auf welcher er die zu testende Software am laufen hat und zusätzlich einen Bildschirm haben, um Fragen bzw. die Tasks anzeigen zu lassen und ggf. mit dem Testleiter kommunizieren. Diese Ansicht hat deutlich weniger Funktionalitäten und hat hauptsächlich die Kommunikation im Vordergrund.



Abbildung 12 – Export Seite im Mockup

Diese Ansicht sollte dazu dienen, die jeweiligen Logs herunterzuladen, wurde jedoch im Laufe des Projektes verworfen und mit einem Button ersetzt.

## Implementierung

Zunächst war der Plan alle Funktionalitäten innerhalb OBS zu implementieren, doch nach ausgiebiger Recherche haben wir uns dazu entschlossen, die Funktionalitäten auszulagern und auf einer externen Homepage zu gestalten, da eine direkte Implementierung in OBS selber nicht möglich ist für unsere Zwecke. Problematisch hierbei war es, dass der Proband selber seinen Bildschirm streamt und es daher nicht möglich ist, von extern (also als Testleiter) darauf zu zugreifen. Daher wurden einige Funktionalitäten auf der externen Homepage transferiert. Lediglich der Key Logger muss beim Probanden eingebunden werden, welcher dann nach Abschluss des Tests manuell an den Testleiter gesendet werden muss. \*

### Funktionalitäten

Auf Basis der Anforderungsanalyse sowie zahlreichen Meetings wurden grundlegende Funktionalitäten herausgefiltert, die in unserer Anwendung implementiert wurden. Folgend wird kurz ein Überblick über alle Funktionalitäten auf der Webpage gegeben, die auf der Basis der User Stories beruhen:

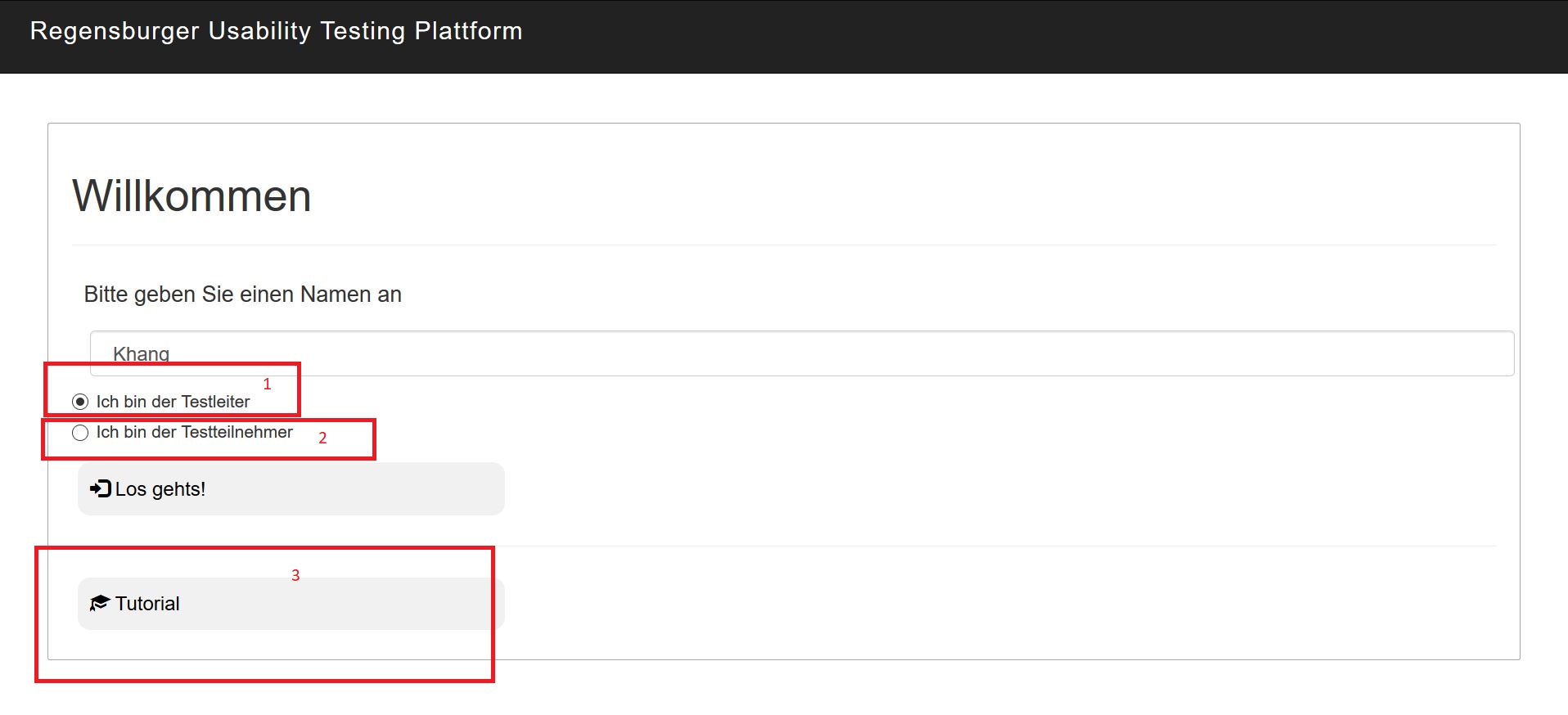


Abbildung 13 – Hauptseite der Anwendung

Zunächst muss ein Name angegeben werden, damit sich die jeweiligen Personen identifizieren können. Auf der Hauptseite der Webpage gibt es dann folgende Möglichkeiten:

1. Zur Testleiter Ansicht zu kommen
2. Zur Testperson Ansicht zu gelangen
3. Auf die Tutorial Seite zu gelangen

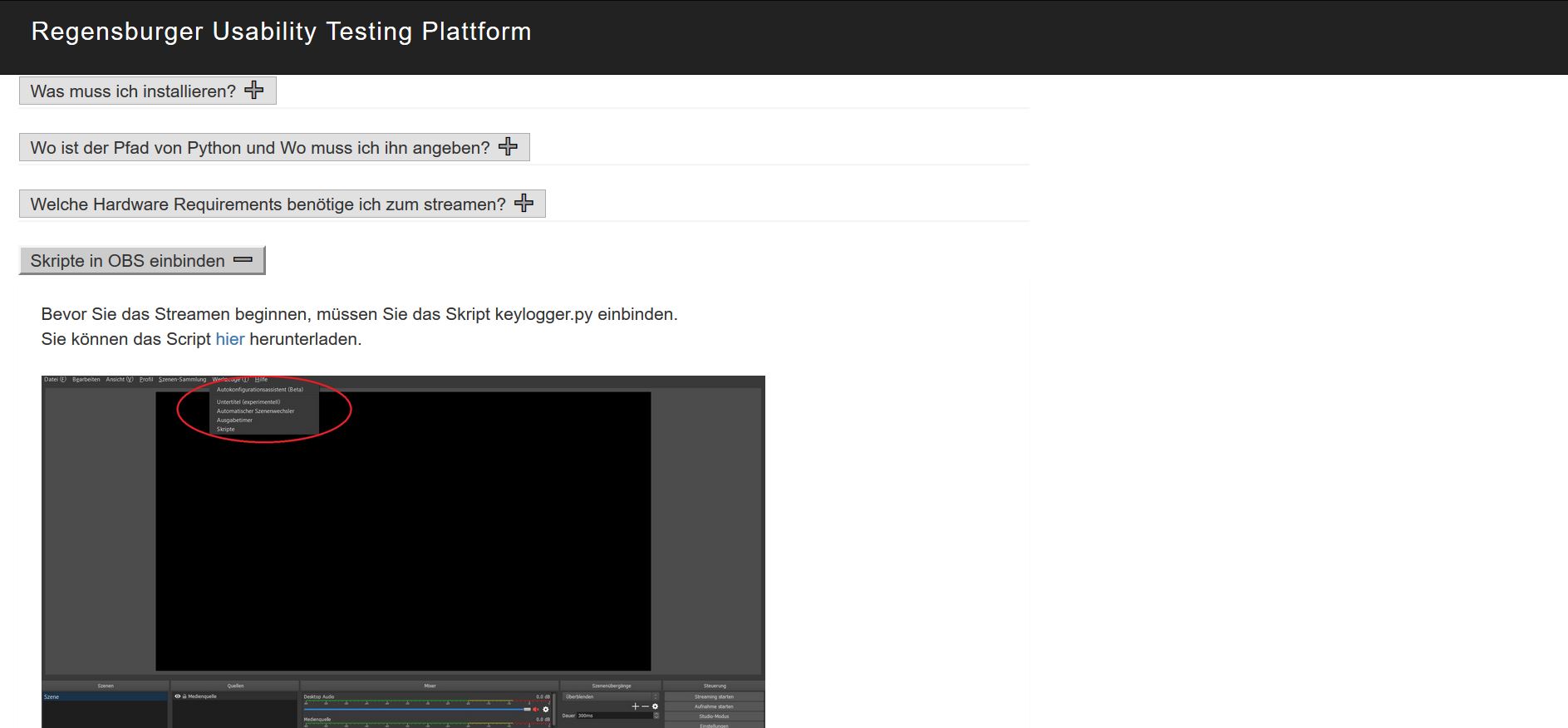


Abbildung 14 – Tutorial Seite der Anwendung

Auf der Tutorial Seite sind grundlegende Fragen beantwortet, wie die Anwendung funktioniert. Ebenfalls kann man sich hier die Keylogger Datei herunterladen und es gibt verschiedene kurze Anleitungen wie man OBS konfiguriert, um den Stream ordnungsgemäß zu starten. Ebenfalls wird kurz erläutert, welche Elemente die Webpage besitzt.

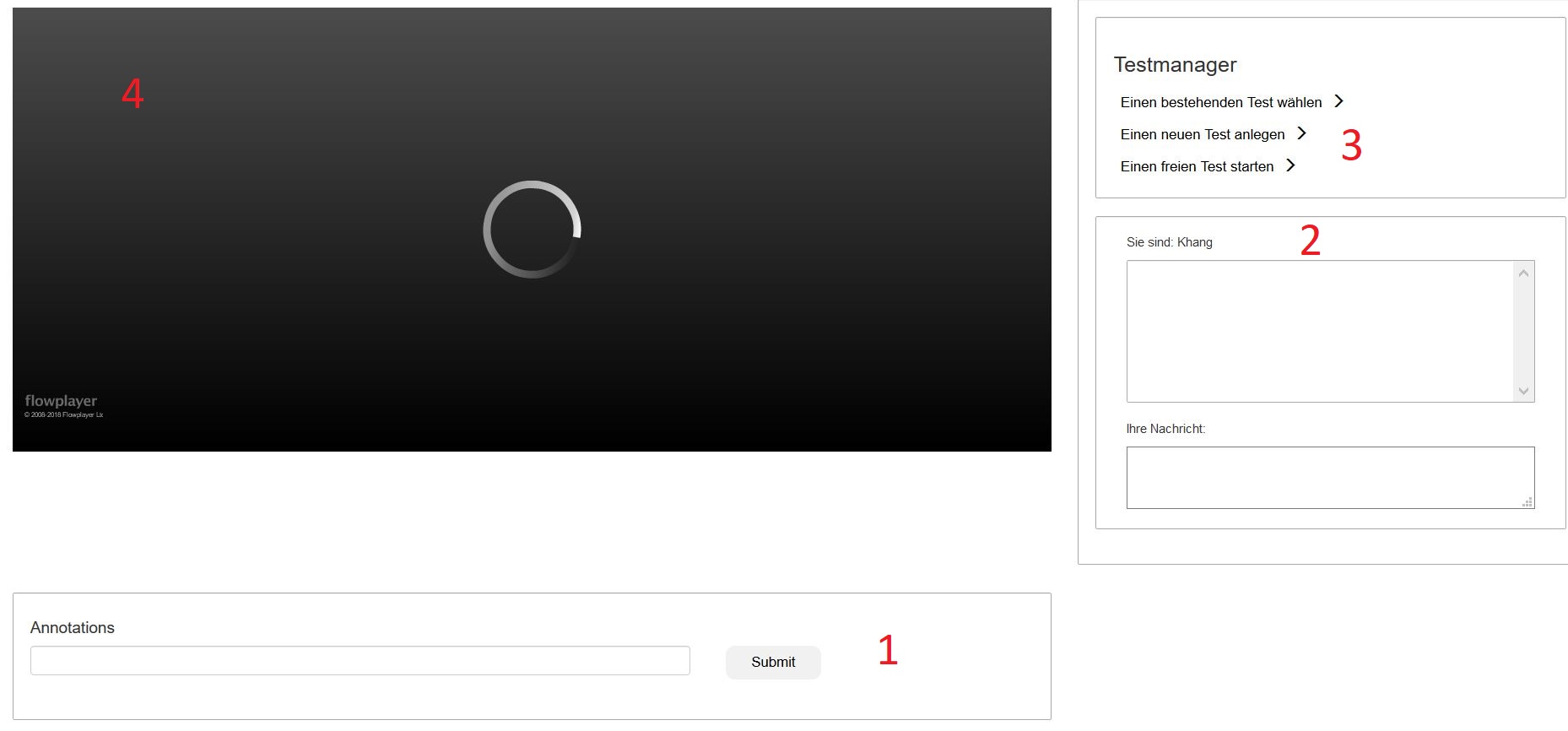


Abbildung 15 – Testleiter Ansicht der Anwendung

Abbildung 8 zeigt die Ansicht des Testleiters. Dieser benötigt lediglich einen Bildschirm, jedoch würde ein zweiter Bildschirm ebenfalls sinnvoll sein, um etwaige Informationen zu notieren bzw. den Stream zu maximieren und den Chat auf der rechten Seite zu öffnen. Folgende Aktionen können hier durchgeführt werden:

1. Zeitsensitive Annotationen können notiert werden, welche in einer Text Datei lokal abgelegt werden. Diese können zur Nacharbeit benutzt werden
2. Der Chat dient zur Kommunikation zwischen Testleiter und Testperson, da bei Remote-Testing sich Teilnehmer nicht gegenüber sitzen 🡪 Voice Talk evtl. auch sinnvoll
3. Der Taskmanager ist nur in der Testleiter Ansicht verfügbar und wird später näher erläutert
4. Der Stream ist ein Hauptelement dieser Ansicht und zeigt den Screen der Testperson an. Der Stream ist im Regelfall um einige Sekunden verzögert, sollte aber für ein Testing irrelevant sein

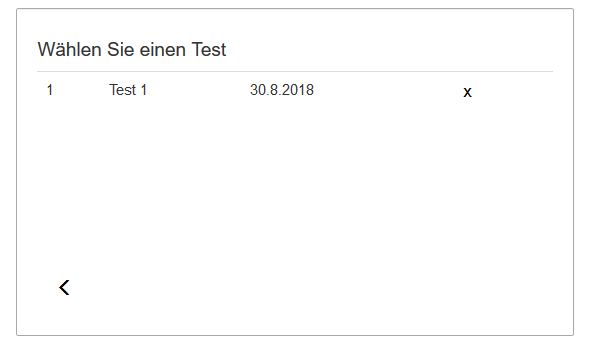


Abbildung 16 Testmanager – Einen bestehenden Test wählen

Hier kann ein bestehender Test gewählt werden mit den jeweiligen Tasks. Dies spart Zeit, wenn man mehrere Probanden mit demselben Testdurchlauf absolvieren will.

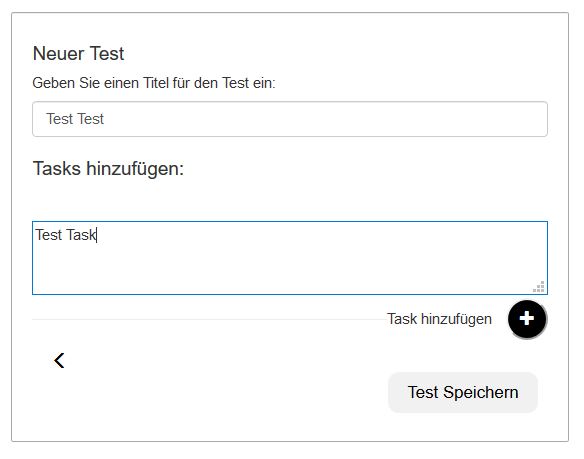


Abbildung 17 Testmanager – Einen neuen Test anlegen

Hier werden neue Tests angelegt, die dann in der Datenbank abgelegt werden. Es können beliebig viele Tasks hinzugefügt werden durch den „Task hinzufügen – Button“. Durch „Test Speichern“ wird der Test gespeichert.

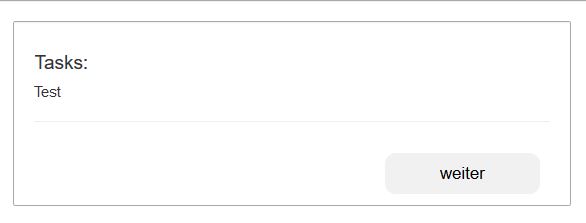


Abbildung 18 Testmanager – Tasks

Nachdem ein existierender Test gewählt bzw. ein neuer Test angelegt wurde, können diese gestartet werden. Die eingegebenen Tasks werden nacheinander der Testperson übermittelt, indem der „weiter-Button“ geklickt wird. Man kann erst wieder auf den Button klicken, sofern die Testperson den Task als „completed“ markiert hat.

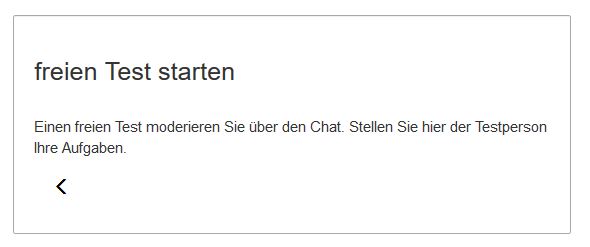


Abbildung 19 Testmanager – Einen freien Test starten

Ein Freies Test wird über den Chat durchgeführt, dies findet vor allem bei unstrukturierten Interviews Verwendung und wird meist von der Testperson geleitet.

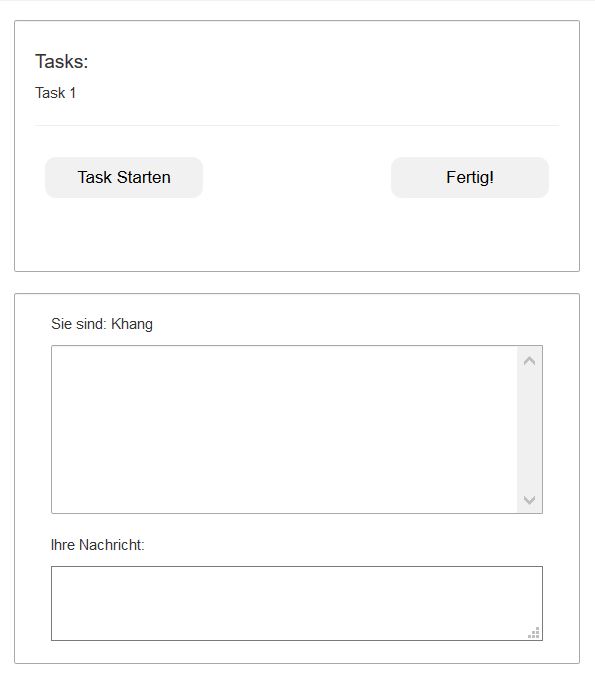


Abbildung 20 – Taskansicht auf Testperson Seite

Die Testpersonen Ansicht ist relativ simpel gestaltet. Der untere Bereich gestaltet sich aus dem Chat, der obere Bereich beinhaltet die Taskübermittlung. Sobald ein Task gestartet wird, läuft im Hintergrund ein Counter. Sobald der Task abgeschlossen ist, soll der „Fertig“ Button betätigt werden, sodass die Task Completion Time notiert werden kann. Diese Datei wird lokal abgelegt und muss noch hochgeladen werden. Dabei kann der Testleiter erst einen neuen Task übermitteln, wenn der vorherige abgeschlossen ist.

### Systemarchitektur

Beschreiben Sie die Systemarchitektur der Anwendung. Wie sind die verschiedenen Komponenten aufgebaut? Welche Frameworks oder Bibliotheken haben Sie eingesetzt und wie wurden diese in das System integriert. Sie sollten an dieser Stelle einen Überblick über die technische Umsetzung des Projekts und den implementierten Quellcode liefern. Fassen Sie größere logische Module Ihrer Software zusammen und beschreiben Sie die interne Kommunikation innerhalb Ihrer Anwendung. Der Abschnitt sollte dabei als Einführung und Ergänzung zur Ihrer Code-Base dienen. Unabhängig davon muss Ihr Code selbst auch sinnvoll dokumentiert und kommentiert sein. Verweisen Sie an dieser Stelle auf mögliche externe Dokumentationen, die Sie aus den Quellcode-Kommentaren generiert haben.

Das Ziel dieses Kapitel ist es, Dritte (z.B. Auftraggeber(!)) in die technische Implementierung Ihrer Anwendung einzuführen. Zusammen mit der Code-Dokumentation und der Setup-Anweisung sollte ein fähiger Programmierer in der Lage sein, Ihren Code nachzuvollziehen und diesen zu warten bzw. zu erweitern.

## Spezifikation

Beschreibung der auf der Basis der Anforderungen entwickelten Spezifikation für Ihr Vorhaben. Die Spezifikation hängt natürlich stark von der Art des Projektes ab (welche Art Software wird entwickelt, welche Plattformen werden genutzt, welche Rolle spielt das Interaktions- oder UX-Design etc.).

# Evaluation

Die Evaluation ist ein iterativer Prozess, welcher von Beginn an durchgeführt wurde, um den Anforderungen gerecht zu werden. Die ersten spezifischen Evaluationen wurden mit Mitkommilitonen durchgeführt, die im Laufe ihres Studiums schon einige Usability Test absolviert haben, daher ist das Feedback dieser Probanden signifikant und relevant, da diese schon Erfahrungen mit Usability Tests haben. Eine ausführliche Evaluation wurde dann mit dem Stakeholder durchgeführt, da dieser der Endnutzer ist und somit sein Feedback von hoher Bedeutung ist. Die erhobenen Daten wurden im Endprodukt berücksichtigt.

## Testing mit Mitkommilitonen

Im Rahmen des Projektes wurden sieben Medieninformatik Studenten getestet. Aus einer Studie von Virzi (1992) kann man entnehmen, dass auch schon bei einer geringen Anzahl an Probanden qualitative Daten entnommen werden können da die signifikantesten Probleme schon bei den ersten 4-5 Probanden erkannt werden. Hauptsächlich ging es hierbei um die Bedienbarkeit der Seite sowie Designtechnisches Feedback der Probanden.

Aus der Anforderungsanalyse sowie die daraus entstehenden User Stories haben wir Use Cases generiert, welche die Hauptfunktionalitäten abprüfen. Hierbei wurden folgende Funktionalitäten getestet:

* Auf die Testleiter bzw. Testteilnehmer Seite gelangen
* Einen Stream starten (mit einer kurzen Anleitung)
* Chatten
* Skript hinzufügen zu OBS
* Annotationen hinzufügen

Es wurden zwei Laptops für die Testpersonen Seite sowie einen Laptop für die Testleiterseite zur Verfügung gestellt. Es wurde eine Anwendung in einem prototypischen Zustand vorgestellt, welcher Kernfunktionalitäten beinhaltete. Das Feedback von der Zwischenpräsentation wurde berücksichtigt.

**Aufgabe 1:**

Die jeweiligen Probanden sollten von der Hauptseite in die jeweiligen Ansichten (Testleiter und Testteilnehmer) gehen. Hierbei konnten alle Probanden die Aufgabe ohne weitere Probleme absolvieren.

**Aufgabe 2:**

Die zweite Aufgabe war es, einen Stream zu starten. Dabei wurden den Probanden alle nötigen Informationen sowie ein kurzes Tutorial zur Verfügung gestellt. Hierbei konnten fünf Probanden ohne Probleme den Stream starten, zwei Probanden hatten anfangs Probleme, da Sie nicht wussten, was Sie in den jeweiligen Felder angeben müssen. Diese konnten aber auf Hinweis der Testleiter auf das Tutorial, die Aufgabe ohne weitere Probleme beenden. Hierbei konnte man erkennen, dass einige Probanden schon Erfahrungen mit dem Streaming haben und keinerlei Hilfe von dem Tutorial benötigten.

**Aufgabe 3:**

Eine weitere Kernfunktion war das Chatten zwischen Testleiter sowie Testperson. Dies wurde von unseren Probanden schnell gefunden und ohne Probleme erkannt. Die Chatfunktion wurde überwiegend positiv empfunden, jedoch wurden von einigen Probanden das persönliche Gespräch (oder auch Voicechat) bevorzugt, da man sich dann besser auf die Tasks konzentrieren kann.

**Aufgabe 4:**

Bei diesem Task wurde primär darauf geachtet, ob der Proband ein externes Skript in OBS einbinden kann. Hierbei wurde dem Probanden zunächst keine Hilfestellung gegeben, nur bei Bedarf wurde dem Probanden eine kurze Anleitung zur Verfügung gestellt. Vier Probanden konnte nach einer kurzen Zeit das Skript (dies war auf dem Desktop vorgegeben) erfolgreich einbinden. Die restlichen Probanden waren sich unsicher und haben nach der Hilfestellung die Aufgabe erfolgreich absolvieren können. Hierbei ist besonders zu vermerken, dass zwei der vier Probanden, die es geschafft haben da Skript einzubinden, nie zuvor OBS benutzt haben.

**Aufgabe 5:**

Diese Aufgabe wurde von allen Probanden ohne weitere Probleme abgeschlossen.

Zusammenfassend kann man sagen, dass alle Probanden die Tasks abschließen konnten. Die Tutorials sind sehr hilfreich und wurden von allen Leuten positiv empfunden. Da die Anwendung von verschiedenen Nutzergruppen benutzt werden soll, ist die Anwendung selber sowie das Tutorial relativ simpel gestaltet, damit jeder – auch medial weniger bewandte Personen - diese Anwendung nutzen kann.

## Evaluation mit Stakeholder

Diese Evaluation wurde gegen Ende durchgeführt, um auf letzte Wünsche des Stakeholders einzugehen. Die Anwendung war in einem Alpha Stadium, welches alle gewünschten Funktionalitäten beinhaltete. Dies war eine sehr freie Evaluation, der Stakeholder durfte sich eigenhändig die Anwendung anschauen und sollte bei Bedarf an den User Stories sich orientieren bzw. die Leiter fragen. Der Stakeholder explorierte alle Funktionalitäten eigenhändig, konnte die Kernfunktionen eigenständig ausführen und ohne Probleme erkennen. Zunächst hat sich der Proband die Testleiter Seite angeschaut und hat einige Tasks erstellt, welches kein Problem darstellte. Daraufhin hat er den Chat sowie die Annotationen getestet, die in einem separaten Log-File gespeichert werden. Auch dies verlief reibungslos ab. Ebenfalls wurde die Ansicht der Testperson exploriert, spezifisch wurde das Setup des Streams selber auf OBS Ebene erkundigt. Hier wurde dem Probanden kurz erläutert, wie er vorgehen müsste, dieser verstand den Prozess ohne Probleme. Es wurde darauf geachtet, dass alle Funktionen auf Basis der User Stories (vgl. Kapitel x) vom Probanden getestet werden, dabei konnten alle Funktionalitäten gefunden sowie getestet werden. Im groben und ganzen war der Stakeholder positiv überrascht und konnte keine signifikanten Mängel feststellen. Dennoch wurden einige Punkt erwähnt, die für das Endprodukt relevant sind und noch geändert werden mussten:

* Login Testperson bzw. Testleiter sollte umbenannt werden 🡪 irritiert, da man sich per se nicht „einloggt“ 🡪 Test durchführen bzw. an Test teilnehmen
* Nachdem die Tasks beendet werden, sollte ein Download Button erscheinen für die Log Files
* Das Tutorial musste erweitert werden
* Nachdem ein Test gespeichert wurde, sollte es die Möglichkeit geben, diese zu Clearen oder zu einem existierenden Test zu wechseln
* Chat noch fehlerhaft 🡪 wurde mehrmals geschrieben
* Completion time der Task als CSV ausgeben zur einfacheren Auswertung
* Der gleiche Test mit mehreren Probanden sollte entweder gleich nummeriert werden oder in einem Ordner gepackt werden

Abschließend lässt sich sagen, dass der Stakeholder sehr zufrieden mit unserer Anwendung ist, da alle Anforderungen sowie Funktionalitäten erfüllt sind. Die oben genannten Punkte wurden beim Endprodukt berücksichtigt. Das Aufsetzen des Servers zur Nutzung dieser Software wird im nächsten Kapitel genauer erläutert.

# Setup-Anweisungen

Je nach Art Ihres Projekts beschreiben Sie hier welche Schritte notwendig sind, um Ihr Projekt auszuführen bzw. verweisen auf eine entsprechende Web-Adresse, Datei o.ä. Ihre Anwendung sollte ggf. als ausführbare Datei oder über eine entsprechende Setup-Anwendung zugänglich sein.

Die hier adressierten Dateien sollen es dem Gutachter bzw. dem Auftraggeber ermöglichen Ihre Anwendung problemlos auszuführen und zu testen.

# Ausblick

Die Anwendung ist einem nutzbaren Zustand, kann dennoch weiter verfeinert werden und neue Funktionalitäten umgesetzt werden.

**Quellen:**

Beyer, H., & Holtzblatt, K. (1997). *Contextual design: defining customer-centered systems*. Elsevier.

Virzi, R. A. (1992). *Refining the test phase of usability evaluation: How many subjects is enough? Human factors*, *34*(4), 457-468.

Projektmanagement: Definitionen, Einführungen und Vorlagen. Retrieved from

<http://projektmanagement-definitionen.de/glossar/scrum/> [03.10.2018]

1. https://obsproject.com/de[30.09.2018] im Folgenden OBS [↑](#footnote-ref-1)
2. https://www.digitalocean.com/[30.09.2018] [↑](#footnote-ref-2)