



Universität Regensburg

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	Zusammenfassung	4
3	Angaben zu Ressourcen und Dokumenten	5
4	Problemstellung und Projektbeschreibung	5
5	Projektmanagement	6
5.1	Agiles Projektmanagement	6
5.2	Projektmanagement-Tools.....	6
5.2.1	Taiga	7
5.2.2	Asana und Instagantt	8
5.2.3	Projektplan	8
5.2.4	Codeverwaltung	10
5.3	Meetings	10
5.3.1	Meetings mit Kursleiter und Stakeholder	11
5.3.2	Sprints	14
6	Architektur und Implementierung	18
6.1	Anforderungsanalyse	18
6.1.1	Contextual Inquiry.....	18
6.1.2	User Stories	19
6.1.3	MockUps	20
6.2	Implementierung	24
6.2.1	Funktionalitäten	25
6.2.2	Systemarchitektur	30
6.3	Evaluation.....	31
6.3.1	Evaluation mit Mitkommilitonen	31
6.3.2	Evaluation mit Stakeholder	33
7	Setup-Anweisungen	34
7.1	Setup Server	34
7.2	Setup Keylogger	37
8	Ausblick	38

Abbildungen

Abbildung 1 - Taiga Übersicht	7
Abbildung 2 – Anforderungsanalyse	8
Abbildung 3 – Projektplanung.....	8
Abbildung 4 – Projektdurchführung.....	9
Abbildung 5 – Design	9
Abbildung 6 – Evaluation	9
Abbildung 7 - Sonstige Milestones.....	10
Abbildung 8 – Projektplanübersicht.....	10
Abbildung 9 – Hauptseite.....	21
Abbildung 10 – Testleiter Ansicht im Mockup	22
Abbildung 11 – Testpersonen Ansicht im Mockup	23
Abbildung 12 – Export Seite im Mockup.....	24
Abbildung 13 – Hauptseite der Anwendung	25
Abbildung 14 – Tutorial Seite der Anwendung	26
Abbildung 15 – Testleiter Ansicht der Anwendung	26
Abbildung 16 - Testmanager – Einen bestehenden Test wählen	27
Abbildung 17 - Testmanager – Einen neuen Test anlegen.....	28
Abbildung 18 - Testmanager – Tasks	28
Abbildung 19 - Testmanager – Einen freien Test starten	29
Abbildung 20 – Taskansicht auf Testperson Seite.....	29

1 Einleitung

Evaluierungen und die dazu gehörigen *Usability Tests* sind ein zentraler Bestandteil des Softwareengineering Prozesses (Schatten et al.), weshalb *Remote-Tests* einen sehr wichtigen Aspekt darstellen, 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 als Thema im Rahmen des Praxisseminars des Masterkurses Medieninformatik ausgeschrieben wurde. Zunächst werden in dieser Dokumentation allgemeine Informationen über das Projekt und dessen Rahmenbedingungen dargelegt, um einen groben Überblick zu verschaffen. Anschließend wird der Prozess des Projektmanagements erläutert, welche Methode verwendet wurde und die einzelnen Meetings beschrieben, damit die Entscheidungen während des Projektes nachvollzogen werden können. Im Anschluss daran wird die Durchführungsphase der Implementierung sowie die Systemarchitektur detailliert beschrieben. Abschließend werden die Setups für wichtige Komponenten der Anwendung erklärt und einen kurzen Ausblick gegeben, wie die Anwendung noch weiter erweitert werden kann. Diese Software soll open source und frei verfügbar sein, hierbei werden keinerlei Lizenzen oder sonstige Berechtigungen benötigt. Durch zusätzliche Plugins soll es zudem möglich sein, weitere Funktionalitäten hinzuzufügen.

2 Zusammenfassung

Die konkrete Aufgabe bestand darin, eine *Usability Testing* Plattform zu gestalten, die es dem Nutzer ermöglicht, mit möglichst wenig Aufwand, einen Usability Test zu absolvieren, welcher sowohl lokal als auch online stattfinden kann. Dies wurde unter der Nutzung der Software *Open Broadcaster Software* (fortan mit OBS abgekürzt) und einer Homepage realisiert, die vom Testleiter sowie Proband abgerufen werden muss und jeweils eine andere Ansicht besitzt. Dabei sollten verschiedene Funktionalitäten implementiert werden (vgl. Kapitel 6.2.1). Der Testleiter hat dabei mehr Funktionalitäten zur Verfügung und sieht den Bildschirm des Probanden im Livestream, während der Proband den Test auf zwei Monitore absolviert. Auf einem werden ihm seine Tasks aufgelistet und der Chat mit dem Testleiter angezeigt und auf dem Anderen absolviert er die ihm gestellten Aufgaben. Bevor die Anwendung verwendet werden kann, muss ein Server konfiguriert werden, sowohl um den Stream darauf zu hosten, als auch um die Webseite verwenden zu können. Während der Entwicklung der Software wur-

de dafür ein *webhosted* Server verwendet, der zwar voll funktionsfähig ist, aber nur noch zeitlich begrenzt zur Verfügung steht.

Der Server sollte, bevor die Software verwendet werden kann, anhand der enthaltenen Anleitung, auf einem eigens dafür gedachten Server der Universität, implementiert werden.

3 Angaben zu Ressourcen und Dokumenten

Um diese Software nutzen zu können, werden einige zusätzliche Elemente benötigt. Da die jeweiligen Tests nicht immer lokal ablaufen, sondern zum Teil remote, auch über größere Distanzen, wird ein Server benötigt, um über diesen den Stream zu hosten. Testweise wurde dies mit der „Digital Ocean“¹ Plattform realisiert, zur realen Inbetriebnahme dieser Software sollte der Stream auf einem eigenen Server gehostet werden. Auf dem Server müssen PHP², Apache³, NGINX⁴ und MySQL⁵ installiert werden. Eine genaue Erläuterung zur Inbetriebnahme der Software sowie alle benötigten Schritte entnehmen Sie bitte dem beigefügten Dokument (Server_setup.pdf).

4 Problemstellung und Projektbeschreibung

Der Fokus bei der Entwicklung dieser Anwendung lag darauf, eine kostenfreie *open source Usabilitytesting* Software zu entwickeln, damit verschiedene Nutzergruppen der Universität Regensburg diese für Ihre Zwecke nutzen können. Es soll möglich sein, Tasks zu erstellen und eine Kommunikation zwischen Testleiter und Proband existieren. Dies ist einerseits durch einen Chat realisiert, aber für einen realen Usability Test wäre ein Voice Chat unter Umständen die bessere Alternative. Die Zielgruppe sind zunächst Studenten und Dozenten der Universität Regensburg, 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 *Open Broadcaster Software* 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.

¹ <https://www.digitalocean.com/>

² <http://php.net/>

³ <https://httpd.apache.org/>

⁴ <https://www.nginx.com/>

⁵ <https://www.mysql.com/de/>

5 Projektmanagement

Das Projektmanagement ist ein wesentlicher Bestandteil des Softwareengineering Prozesses (Schatten et al., Kapitel 4) 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.

5.1 Agiles Projektmanagement

Als Projektmanagement Methode wurde die SCRUM Methode gewählt (vgl. Projektmanagement: Definitionen, Einführungen und Vorlagen), wobei hier nicht strikt alle Vorgaben des SCRUM Modells eingehalten wurde, sondern eine abweichende Form angewendet wurde, die besser an die Projektlage angepasst war. Wöchentliche Sprint-Meetings wurden gehalten, wobei diese teils in Person und teils via Kommunikationsapplikationen (Teamspeak⁶) absolviert worden sind. Im Laufe des Projektes wurde das Projektmanagement nach und nach an die Bedürfnisse angepasst. Zunächst wurden wöchentliche Spring-Meetings abgehalten, um den Fortschritt des Projektes zu gewährleisten. Gruppenmeetings waren vor allem sinnvoll um das Projekt gezielt voran zu treiben und vor allem zu Beginn nützlich, um den Grundstein zu legen. Teamspeak Meetings kamen hingegen hauptsächlich gegen Ende des Projekts zum Einsatz, um kurze Statusupdates der Teammitglieder zu erhalten. Letztendlich ist eine Mischung aus beidem die beste Lösung, da die Arbeitspakete unterschiedlich viel Zeit beanspruchen. Da es sich bei der gewählten Methode um ein agiles Management handelt, war es dadurch möglich etwas flexibler zu handeln. Gegen Ende war insgesamt eine intensivere Meeting-Dichte zu verzeichnen.

5.2 Projektmanagement-Tools

Projektmanagement Tools erleichtern die Dokumentation des Projektes erheblich und sind ebenfalls sehr gut geeignet, um den Stand des Projektes als Teammitglied mitzuverfolgen. Dabei werden Milestones sowie Arbeitspakete unterteilt, um eine bessere Übersicht zu be-

⁶ <https://www.teamspeak.com/en/>

kommen und den Fortschritt des Projektes zu gewährleisten. Zunächst wurde mit dem Tool Taiga⁷ gearbeitet. Nach dem ersten Monat kam es zu der Entscheidung, auf die Plattformen Asana⁸ und Instagantt⁹ umzusteigen, da diese besser an die Anforderungen angepasst sind.

5.2.1 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 wurde jedoch festgestellt, dass es besser angepasste Alternativen für das Projekt existieren, weshalb schlussendlich zu Asana bzw. Instagantt gewechselt wurde. Diese bieten die Möglichkeit aus dem Projektplan einen Milestone Plan bzw. ein Gantt-Diagramm zu generieren, was den Managementaufwand deutlich reduzierte und bessere Ergebnisse lieferte.

OBS-PLUGIN ISSUES








				+ NEW ISSUE	
Votes	Subject	Status	Assigned to		
▲ 0	#31 streaming on own server	New ▾	 kinderfest		
▲ 0	#30 chat with the subject	New ▾	 Hallo		
▲ 0	#29 adding time sensitive annotations	New ▾	 kinderfest		
▲ 0	#28 Task modeling	New ▾	 julifunk		
▲ 0	#27 Getting used to working environment	New ▾	 webphil		
▲ 0	#21 Topics for next meeting ⌚	New ▾	 julifunk		
▲ 0	#19 Tasks until next meeting on (31.5.18) + Proposal ⌚	In progress ▾	 webphil		

Abbildung 1 - Taiga Übersicht

⁷ <https://taiga.io/>

⁸ <https://asana.com/de>

⁹ <https://instagantt.com/app>

5.2.2 Asana und Instagantt

Auf Asana wurde ein detaillierter Projektplan angelegt, welcher zu Beginn der Projektphase sehr strikt eingehalten werden konnte. Gegen Mitte der Projektlaufzeit wurden hier Nachlässigkeiten verzeichnet, welche sich auf private Umstände der Projektmitglieder zurückführen lassen. Dies wurde aber im späteren Verlauf des Projektes zunehmend aufgefangen, um die Implementierung der Anwendung zu garantieren. Dies geschah durch eine erhöhte Intensität sowie Dichte der Meetings innerhalb der finalen Wochen. 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.

5.2.3 Projektplan

Der erste detaillierte 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:

OBS-Plugin		Sub Tasks	Task Prog.
Anforderungsanalyse:			100%
1	Sprechstunden wahrnehmen		100%
2	Gespräche mit Stakeholder		100%
3	Machbarkeitsstudie		100%

Abbildung 2 – Anforderungsanalyse

Die Anforderungsanalyse beinhaltet das Erheben von Daten für das Projekt. Hierbei ging es primär darum, mit den Stakeholdern zu reden, die Anforderungen zu spezifizieren und die Machbarkeit des Projektes zu verifizieren.

Projektplanung:		100%
5	User Stories	100%
6	Finalisierung des SRS	100%

Abbildung 3 – Projektplanung

Teil der Anforderungsanalyse war es ebenfalls, User Stories zu erstellen, sowie das SRS zu finalisieren, um das weitere Vorgehen im Projekt besser planen zu können.

Projektdurchführung:		75%
	System Design	3 100%
	Testen von simplen Skripten	2 100%
	Setup - Implementierung - Testen von jeweiligen Skr...	3 100%
11	Einbinden von Hardware (Raspberry Pi, Webcam, M...	100%
	Setup - Implementierung - Testen von jeweiligen Skr...	4 100%
13	Server von extern erreichbar machen	100%

Abbildung 4 – Projektdurchführung

Die Projektdurchführung war der zeitaufwändigste Teil des Projektes und beanspruchte die meiste Zeit. Hier wurden einige Änderungen an der Umsetzung vorgenommen, die im späteren Verlauf der Dokumentation erläutert werden.

Design:		20%
17	System Mock Up	100%
18	Prototyp	0%
19	Evaluation Design	0%
20	Refinement	0%
21	Testing - Check if workflow is given	0%

Abbildung 5 – Design

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

Evaluation:		0%
23	Testing with Stakeholder	0%
24	Refinement with opinion of Stakeholder	0%

Abbildung 6 – Evaluation

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

Sonstige Milestones:		40%
26	Antrittspräsentation	100%
27	Zwischenpräsentation erstellen	100%
28	Overall refinement of project	0%
29	Endpräsentation	0%
30	Abgabe des finalen Projektes	0%

Abbildung 7 - Sonstige Milestones

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

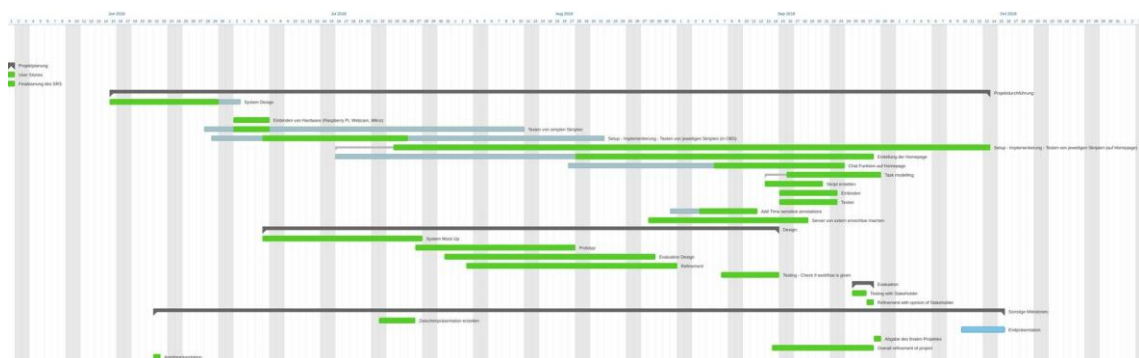


Abbildung 8 – Projektplanübersicht

5.2.4 Codeverwaltung

Als Code Verwaltungsplattform wurde GitHub¹⁰ verwendet, wodurch das Projekt stets aktuell gehalten wurde, damit die Teammitglieder jederzeit an der Anwendung arbeiten konnten. Issues und Milestones wurden intensiv gepflegt und es wurde explizit darauf geachtet, dass diese auch rechtzeitig erfüllt worden sind. Zusätzlich zu GitHub wurde noch die Anwendung SourceTree¹¹ benutzt.

5.3 Meetings

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

¹⁰ <https://github.com/>

¹¹ <https://www.sourcetreeapp.com/>

5.3.1 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. Ebenfalls wurden Fragen zum Projekt geklärt und Änderungen am weiteren Vorgehen besprochen.

5.3.1.1 Meeting 1 – 21.05.2018

Im ersten Meeting mit dem Kursleiter wurden allgemeine Informationen über das Projekt dargelegt, es wurde spezifisch über die Rahmenbedingungen und Architektur der Software diskutiert. Da wenig bis gar keine Erfahrung mit C/C++ vorherrschte, musste eine Alternative gefunden werden, da die meisten Skripte für OBS auf C/C++ Basis waren. Hier wurde als Alternativlösung einige Möglichkeiten gefunden, die Skripte via Python einzubinden. Zusätzlich wurde erörtert, welche Vorstellungen zur Anwendung der Stakeholder hat:

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

5.3.1.2 Meeting 2 – 24.05.2018

In diesem Meeting wurde, durch den Stakeholder, die Anforderungen des Projekts dargelegt. Dieses Gespräch lief als eine sehr freie Form des Contextual Inquiries (vgl. Kapitel 5.1.1) ab, der Stakeholder erklärte, inwiefern er die Applikation während der Arbeit benutzen würde. Der Stakeholder nannte die verschiedenen Grundfunktionalitäten, die die finale Version des Projekts enthalten muss. Auf Basis dieser Informationen, entstanden die Anforderungsliste bzw. die User Stories, die als Basis der Anwendung gelten. 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 besonders günstig 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 speichern
- Als Testleiter Timeline loggen; Metadaten eingeben
- Vgl. Morae → Kein Manager

- Secondary: Chat
- Machbarkeit des Projektes allgemein testen

5.3.1.3 Meeting 3 – 07.06.2018

Bei diesem Meeting wurden weitere Informationen dargelegt und die Umsetzbarkeit des Themas diskutiert. Es wurde die momentane Situation beschrieben und das geplante weitere Vorgehen dem Stakeholder erläutert. Hierbei wurden folgende Probleme mit dem Stakeholder diskutiert:

- Delay des Stream (sehr wichtig für einen Usability Test)
- Nur Zuschauer sieht kompletten Screen → Zweiter Monitor wird benötigt
- Twitch als erster Streamingdienst zu Testzwecken, 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!)

5.3.1.4 Meeting 4 – 05.07.2018

Das zentrale Thema dieses Meeting war die Besprechung des Projektplans, welcher auf Asana bzw. Instagantt erstellt wurde und die Änderungen, die im Laufe des Projektes vorgenommen wurden. Dafür fand ein weiteres Meeting mit dem Kursleiter statt, 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. 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. Wichtig war es zudem, dass die einzelnen Arbeitspakete in dem Zeitraum abgeschlossen werden, die angegeben wurde. Die tatsächlich gebrauchte Zeit variiert jedoch ein wenig, da es sich hierbei nur um Schätzungen handelt. Der Plan an sich war schlüssig, es gab keine weiteren negativen Einwände des Kursleiters

Das zweite Thema der Besprechung war die Auslagerung mehrerer Funktionalitäten auf eine externe Webpage. Nach einigen Recherchen wurde der Entschluss gefasst, 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

wurde der Plan, die Funktionalitäten auf eine extern gehostete Webseite auszulagern, gefasst. 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

5.3.1.5 Meeting 5 – 19.07.2018

Dieses Treffen fand nur mit dem Stakeholder statt und hatte als Themen, den momentanen Stand zu präsentieren sowie mögliche Änderungen zu diskutieren. Der Stakeholder wurde über die durchgeführten Änderungen am Konzept informiert. Unter anderem sollten einige Funktionen ausgelagert werden und verschiedene Ansichten der Webpage erstellt werden. Einerseits wird es eine Testleiter Ansicht und eine Probanden Ansicht geben. Diese unterscheiden sich um ein paar wenigen Funktionalitäten, da der Testleiter mehr Funktionen benötigt. 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 wurde darüber diskutiert, inwiefern der Raspberry Pi auf längerer Sicht genutzt werden kann, da dieser, Hardware technisch nur bedingt für eine längere Nutzung der Anwendung geeignet ist. Alternativ müsste ein Server gemietet werden, um das Streaming dort zu hosten.

5.3.1.6 Meeting 6 – 24.09.2018

Dieses Treffen war zur Evaluation der Anwendung im Alpha Stadium geplant. Dabei wurde dem Stakeholder die Anwendung präsentiert und alle Funktionalitäten getestet. Dies war das letzte Treffen vor der finalen Abgabe des Projekts, weshalb möglichst viel Feedback gesammelt werden sollte, um die Anwendung noch weiter an die Anforderungen anzupassen. Den detaillierten Verlauf sowie die einzelnen Erkenntnisse können Sie dem Kapitel 6.2 entnehmen.

5.3.2 Sprints

Die wöchentlichen Sprints dienten der Gewährleistung des Projektfortschritts. Hier fanden persönliche Treffen statt, alternativ kam es zu Meetings via der Kommunikationsapplikation Teamspeak. Es wurden 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 sie geplant waren. Eine komplette Auflistung der Sprints würde den Rahmen dieser Dokumentation sprengen, weshalb nur die signifikantesten Sprints aufgelistet werden, um die getroffenen Entscheidungen besser nachvollziehen zu können.

5.3.2.1 *Besprechung des Themas*

Die ersten Besprechungen wurden geführt, um in das Thema einzuführen und das Projektteam über OBS bzw. die Anforderungen zu informieren. Zu Beginn wurden zunächst einige Beispiel-Skripte analysiert. Die meisten Skripte basierten auf Lua bzw. C/C++. Allerdings wurde auch die Möglichkeit, Skripte via Python einzubinden diskutiert und als vermutlich beste Lösung festgehalten.

5.3.2.2 *Machbarkeit des Projektes*

Um sicherzustellen, dass das Projekt auch abschließbar ist, musste die Machbarkeit des Projektes überprüft werden. Dafür mussten intensive Recherchen betrieben und einige Setups bzw. Skripte getestet werden. Da die Anforderungen zuvor vom Kursleiter gestellt und nähere Spezifikationen durch den Stakeholder angeführt wurden, konnte grob die Architektur der Anwendung geplant und somit die benötigten Frameworks sowie Komponenten überdacht werden,

um die ersten Tests zu planen. Diese Tests waren ausschlaggebend, ob dieses Thema tatsächlich absolviert werden kann, oder eine Alternative gefunden werden muss.

5.3.2.3 Nacharbeit der Anforderungsanalyse

Aus den zuvor erhobenen Daten der Meetings konnten die ersten groben Anforderungen generiert und die ersten *User Stories* sowie *Epics* (Funktionen) in Taiga erstellt werden. Hier wurden die Funktionen in reale Szenarien verpackt und als *User Stories* aufgestellt, um einen besseren Bezug zu der tatsächlichen Nutzung zu bekommen. Hierbei haben sich zwei Grundfunktionen herauskristallisiert, die auf ihre Machbarkeit getestet werden müssen. Dies war einerseits das Einbinden von Skripten auf Python Basis und andererseits das Hosten eines Streams. Diese beiden grundlegenden Funktionen müssen umsetzbar sein, damit dieses Projekt abgeschlossen werden kann.

5.3.2.4 Skripteinbindung mit Python

Der erste Test war die Einbindung eines simplen Skripts (Countdown Zähler) in OBS. Das Skript selbst war in Python geschrieben, damit sichergestellt werden kann, dass auch ohne Kenntnisse der Sprache Lua bzw. C Anwendungen geschrieben werden können. Anfangs gab es kleinere Probleme, da das Skript nicht erkannt wurde, nach ein paar Testläufen wurde das Skript jedoch korrekt ausgeführt. Dennoch musste noch ein zweiter Test gemacht werden, da der Stream noch auf einem Server gehostet werden musste.

5.3.2.5 Stream hosten via Raspberry Pi

Das Hosten des Streams war das nächste große Arbeitspaket, das durchgeführt wurde. Der erste Gedanke war, den Stream auf einem privaten Raspberry Pi zu konfigurieren und zu hosten. Hierzu wurde der Raspberry Pi in einem privaten Netzwerk angeschlossen und konfiguriert. Nach anfänglichen Schwierigkeiten konnten der Raspberry Pi so konfigurieren werden, dass der Stream gehostet werden konnte. Der Stream konnte als Medienquelle in OBS eingebunden werden und auch von außerhalb des privaten Netzwerkes erreicht werden. Nach einiger Zeit traten jedoch Performance Probleme auf, da Streaming an sich relativ viel Rechenleistung beansprucht und ein Raspberry Pi auf längerer Sicht nicht dafür geeignet ist.

5.3.2.6 Projektplan erstellt

Nachdem die Grundvoraussetzungen des Projektes erfüllt waren, wurde der Projektplan erstellt, welcher auf Asana bzw. Instagantt einzusehen war. Die Details zu diesem Plan wurden im Kapitel 4.2.3 näher erläutert.

5.3.2.7 Erster Grundbauplan der Anwendung

Bei diesem Sprint wurde der Grundbauplan der Anwendung ausgearbeitet. Erste Unterteilungen der Anwendung wurde schon bei der Erstellung des Projektplans gemacht und hier kam 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 wurde sich darauf geeinigt, eine externe Webpage zu gestalten, um dort die Kommunikation zu ermöglichen. Die Kommunikation ist von hoher Bedeutung beim *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 es flexibler zu sein und durch eine externe Webpage können einfacher weitere Elemente hinzugefügt werden. Lediglich der Keylogger musste noch separat bearbeitet werden. Dies wurde vom Kursleiter sowie Stakeholder akzeptiert. Die nächsten Schritte wurden geplant und die jeweiligen Arbeitspakete auf die Mitglieder verteilt.

5.3.2.8 Design der Webpage

Das Design der Webpage wurde anfangs als *MockUp* 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.

5.3.2.9 Streaming hosten via RTMP

Der Raspberry Pi bzw. das Streamen selbst funktioniert nur auf RTMP Basis, da über den JW-Player gestreamt wird. Die weitere Methode wäre das Streaming via HLS (HTTP Live

Streaming), welches für zukünftige Erweiterungen oder Verbesserungen in Betracht gezogen werden sollte.

5.3.2.10 Server Hosting anstatt Raspberry Pi

Nach längerer Diskussion fiel der Entschluss, den Raspberry Pi aufgrund von Performance Problemen durch einen Server zu ersetzen. Hierbei fiel die Entscheidung auf die Server Plattform „Digital Ocean“. Nachdem das Streamen erfolgreich verknüpft wurde, wurden die ersten Funktionalitäten implementiert.

5.3.2.11 Chat

Hier schreiben beide Chatpartner in ein .txt File. Dessen Inhalt wird dann auf der Webseite angezeigt. Diese Methode ermöglicht es am Ende des Tests, dieses File zum Download bereitzustellen, um somit dem Testleiter im Nachhinein die Möglichkeit zu bieten, dieses einfach und schnell auszuwerten.

5.3.2.12 Taskmodellierung

Von der Grundfunktion her funktionieren die Tasks genauso wie der Chat, die Daten werden in ein File geschrieben und anschließend ausgelesen und auf der Webseite angezeigt!

5.3.2.13 Server

Als Grundlage für den Server wurde ein einfaches *Droplet* der Plattform „Digital Ocean“ verwendet. Bei dem verwendeten Betriebssystem handelt es sich um die Ubuntu Version 18.04. Das genau Setup des entnehmen Sie bitte dem Dokument „Server Setup“.

5.3.2.14 Datenbankanbindung

Die Anbindung der MySQL Datenbank erfolgt über *phpmyadmin* und ist Bestandteil der Serverkonfiguration. Im Code selbst fungiert hier PHP als Schnittstelle zwischen der Webpage (Frontend) und der Datenbank (Backend)

5.3.2.15 Finalisierung der Anwendung

Nachdem die Anwendung dem Stakeholder präsentiert wurde und die abschließenden Evaluationen mit Kommilitonen sowie Stakeholder abgeschlossen waren, wurden die finalen Anpassungen getätigt. Es wurden sämtliche Anforderungen des Kursleiters sowie des Stakeholders implementiert und auf die jeweiligen Wünsche der evaluierten Personen eingegangen. Da es keine weiteren Bemerkungen bezüglich der Anwendung gab, konnte die Finalisierung der Anwendung abgeschlossen werden. Hierzu wurden kleinere Bugs gefixt sowie Quellenangaben hinzugefügt und der Code aufgeräumt, damit dieser auch schlüssig ist.

6 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 *MockUps* sowie erste Annäherungen des Designs, woraufhin die Implementierung startete. Abschließend wurden Evaluationen durchgeführt und die letzten Anpassungen getätigt.

6.1 Anforderungsanalyse

Die Anforderungsanalyse wurde relativ früh mit dem Stakeholder durchgeführt. Dies geschah in insgesamt drei Meetings (vgl. Kapitel 5.3), da die jeweiligen Anforderungen einer Machbarkeitsstudie unterlaufen mussten und die Anforderungen näher spezifiziert wurden. Die Funktionalitäten wurden dann durch User-Stories dargestellt, um die Funktionalitäten in reellen Situationen widerzuspiegeln.

6.1.1 Contextual Inquiry

Die grundlegenden Anforderungen wurden vor dem ersten Meeting schon von dem Kursleiter bereitgestellt, weshalb die ersten groben Gedanken der Systemarchitektur vorgestellt werden konnten. In den ersten Meetings wurden Informationen über die Anwendung selbst sowie ein Beispielablauf eines Tests gesammelt. Dies geschah durch eine relativ freie Form des Contextual Inquiries (Beyer & Holtzblatt, 1997, S.37f), indem Stakeholder kurz erklärte, wie die An-

wendung im Alltag Verwendung findet und so einen Überblick gegeben hat, welche Funktionalitäten wichtig sind. Auf Basis dessen entstanden so genannte *User Stories*, die im folgendem zusammengefasst werden.

6.1.2 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 Bildschirms der Testperson ist die Grundfunktion dieser Anwendung und hat somit die höchste Priorität, da ohne diese Funktion ein *Remote-Test* nicht stattfinden kann.

Record key events

Diese Funktion ist nicht zwingend erforderlich, da sie keinen Einfluss auf den Workflow hat, dennoch sollte es bei Gelegenheit implementiert werden. Die Nacharbeit und Auswertung von Daten gehören ebenfalls zum Softwareengineering-Prozess, weshalb diese Funktionen eine mittlere Priorität besitzen.

Chat with subject

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

Task modelling

Das Gestalten von Tasks sowie deren Bearbeitung hat ebenfalls eine sehr hohe Priorität, da es den Workflow direkt 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

OBS ermöglicht das Aufnehmen von dem eigenen Bildschirm, weshalb diese Funktion nicht explizit implementiert werden musste.

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.

6.1.3 MockUps

Aus den ersten Gesprächen mit dem Stakeholder konnten die Anforderungen gefiltert und im Anschluss die ersten *MockUps* generiert werden. Diese dienten als Grundlage für die weitere Designspezifizierung und wurde nach und nach designtechnisch angepasst. Auch die verschiedenen Core-Funktionalitäten wurden abgebildet.

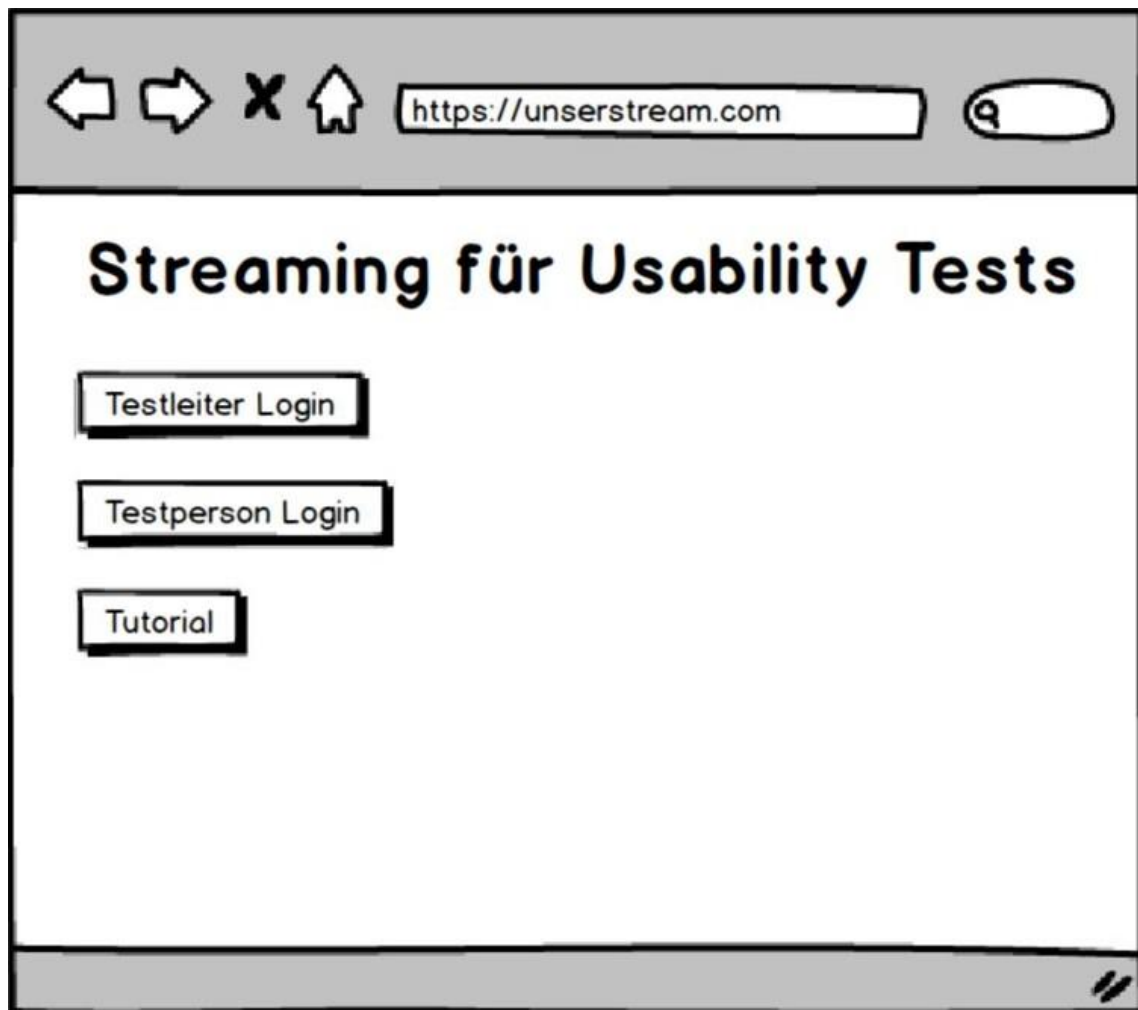


Abbildung 9 – Hauptseite

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.

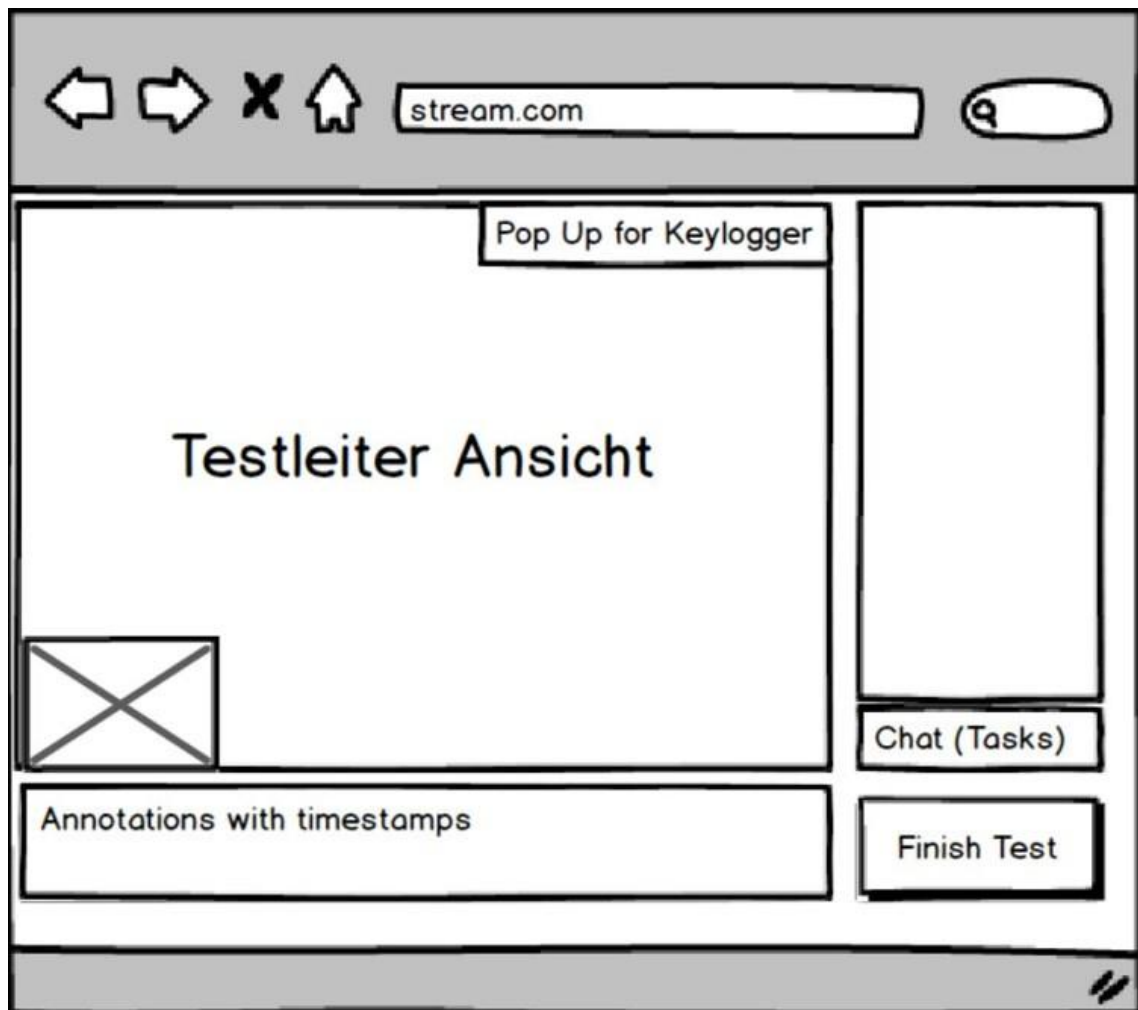


Abbildung 10 – Testleiter Ansicht im Mockup

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

- Stream
- Chat
- Annotationsmöglichkeit



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 ausführt und zusätzlich einen Bildschirm haben, um Fragen bzw. die Tasks anzeigen zu lassen und ggf. mit dem Testleiter kommunizieren. Diese Ansicht hat weniger Funktionalitäten und hat hauptsächlich die Kommunikation im Vordergrund.

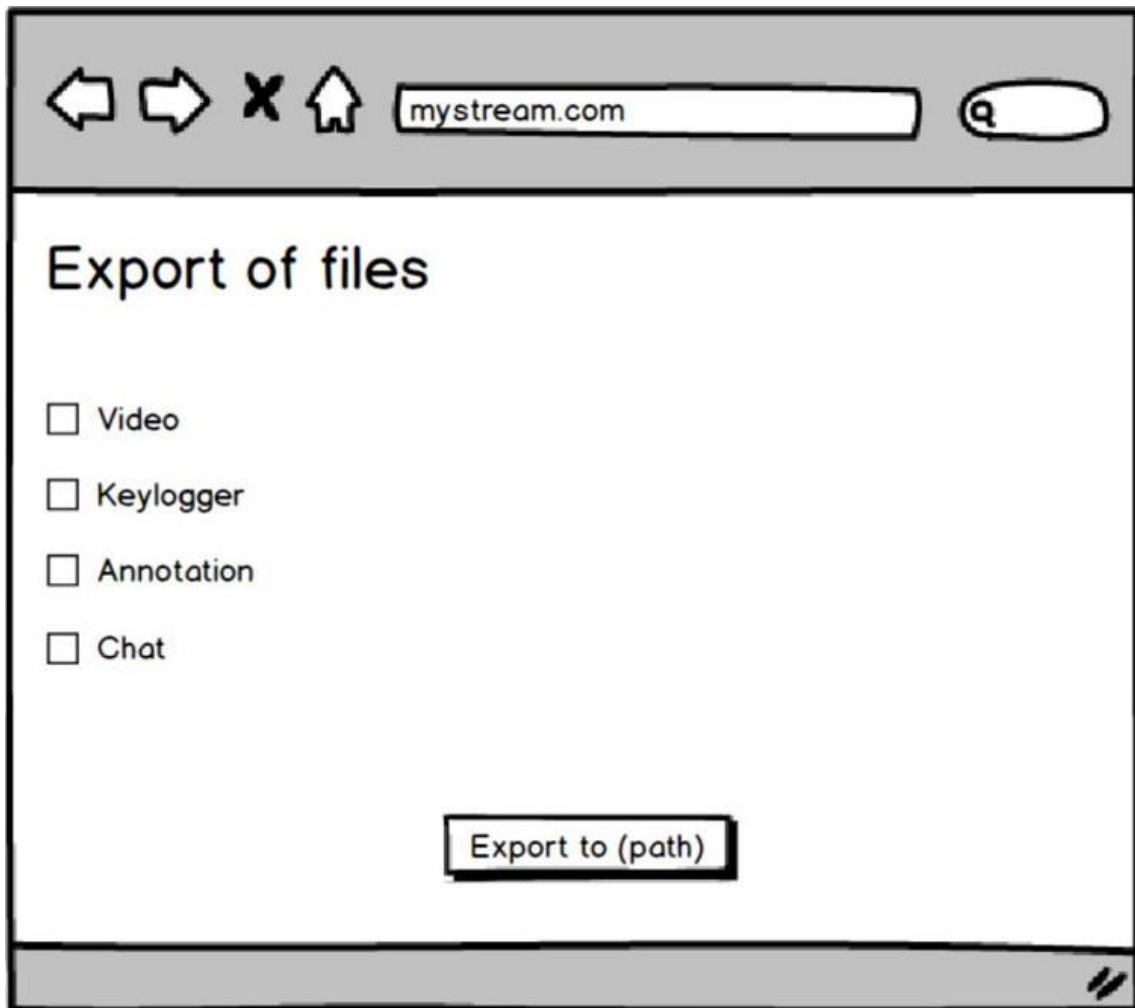


Abbildung 12 – Export Seite im Mockup

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

6.2 Implementierung

Die ersten Konzeptionen der Anwendung wurden schon während der Anforderungsanalyse erstellt und in der Implementierungsphase ausgebaut und spezifiziert. Das Grundgerüst wurde erstellt und die jeweiligen Funktionalitäten wurden anschließend implementiert. Im groben lässt sich die Implementierungsphase wie folgt untergliedern (genauer kann dem Kapitel 5.3.2 entnommen werden):

- Server konfigurieren
- Stream hosten
- Designspezifizierung

- Datenbankanbindung
- Funktionalitäten implementieren
- Evaluation

6.2.1 Funktionalitäten

Als nächstes wird ein Überblick über alle Funktionalitäten auf der Webpage gegeben, die auf der Basis der User Stories beruhen:

Regensburger Usability Testing Plattform

Willkommen

Bitte geben Sie einen Namen an

Khang

☒ Ich bin der Testleiter 1
 ☐ Ich bin der Testteilnehmer 2

Los gehts!

☐ Tutorial 3

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 folgende Möglichkeiten:

- 1) Testleiter Ansicht
- 2) Testpersonen Ansicht
- 3) Tutorial Seite

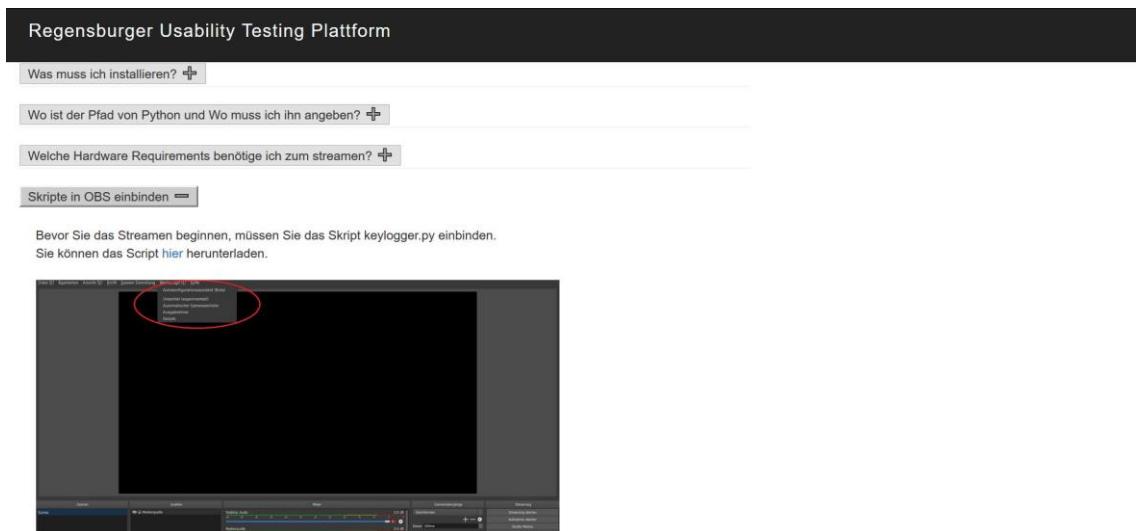


Abbildung 14 – Tutorial Seite der Anwendung

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

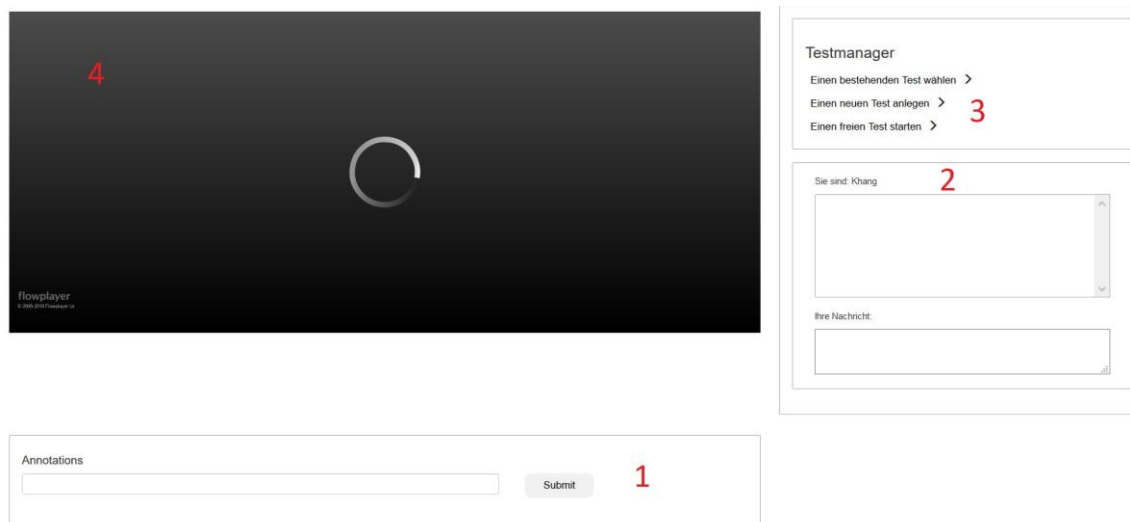


Abbildung 15 – Testleiter Ansicht der Anwendung

Abbildung 15 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 beim *Remote-Testing* sich die 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.

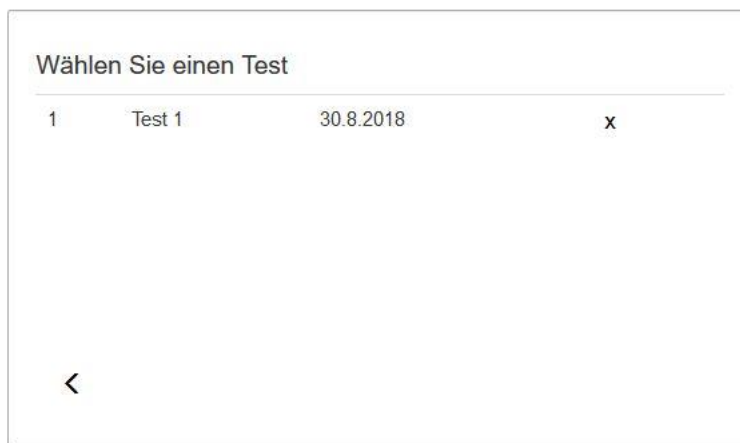


Abbildung 16 - Testmanager – Einen bestehenden Test wählen

Hier kann ein bestehender Test mit den jeweiligen Tasks gewählt werden. Dies erspart Zeit, wenn mehrere Probanden mit demselben Testdurchlauf absolviert werden sollen.

Neuer Test

Geben Sie einen Titel für den Test ein:

Test Test

Tasks hinzufügen:

Test Task

Task hinzufügen

<

Test Speichern

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.

Tasks:

Test

weiter

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. Der Button ist erst wieder klickbar, sofern die Testperson den Task als „completed“ markiert hat.

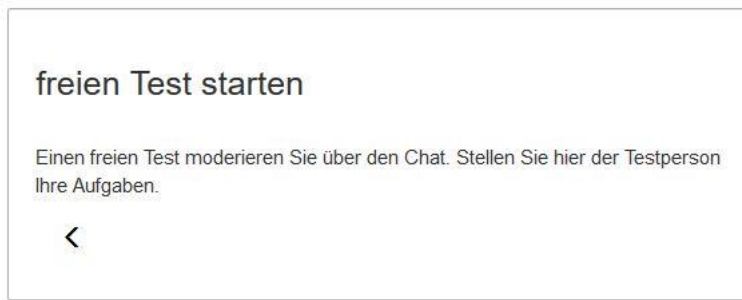


Abbildung 19 - Testmanager – Einen freien Test starten

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

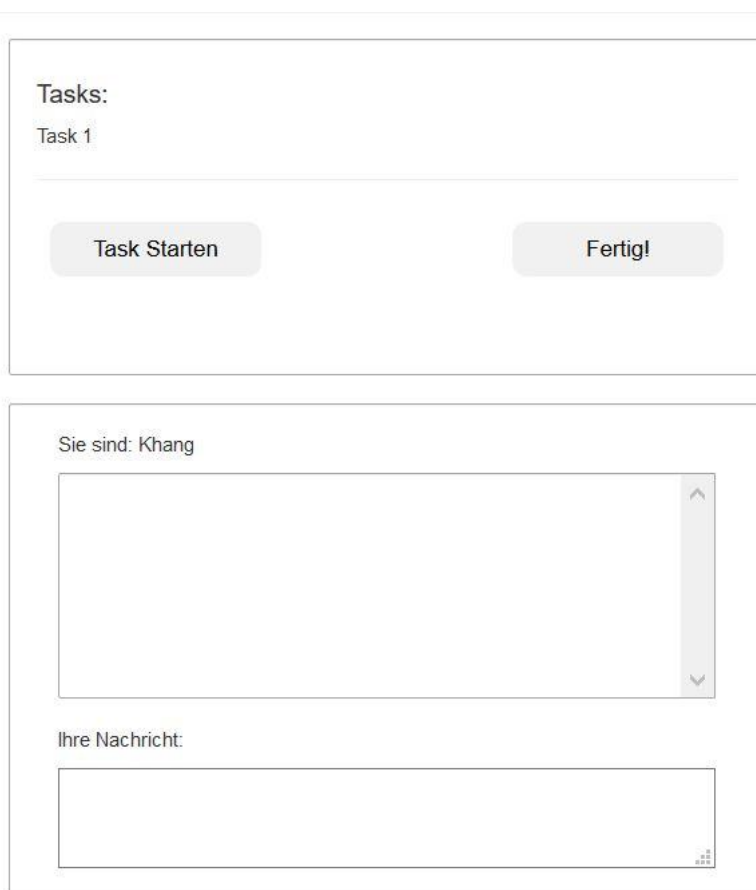


Abbildung 20 – Taskansicht auf Testperson Seite

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 und sobald der Task abgeschlossen ist, muss der „Fertig“ Button betätigt werden, damit die *task completion time* notiert werden kann. Diese Datei wird lokal

abgelegt und muss separat hochgeladen werden. Dabei kann der Testleiter erst einen neuen Task übermitteln, wenn der vorherige abgeschlossen ist.

6.2.2 Systemarchitektur

Das Projekt besteht vorrangig aus zwei Komponenten. Erstens dem Webserver, auf welchem alles gehostet wird und zweitens der Webseite, welche die Benutzung der Applikation erlaubt. Diese sind wie folgt aufgebaut. Für die Implementierung des Servers wurde ein webhosted Ubuntu 18.4 Server verwendet. Dieser wurde dann von Grund auf für das Projekt konfiguriert. Auf dem Server wurden zwei Webserver installiert. Für das benötigte Videostreaming kommt ein NGINX Webserver zum Einsatz, dieser erlaubt über das RTMP, das Übertragen von Livevideostreams über OBS. Außerdem kommt ein Apache2 Webserver zum Einsatz, der das Hosting der Webseite ermöglicht, welche das zentrale Element der Applikation darstellt.

Mit dem Apache Server sind außerdem eine PHP Installation, phpMyAdmin, sowie eine MySQL Datenbank verknüpft, um deren Features für die Nutzung auf der Website zur Verfügung zu stellen.

Zur Erstellung der, auf dem Server gehosteten, Website wurde, neben HTML, JavaScript und CSS, die frontend Komponenten Library Bootstrap verwendet.

Die Webseite an sich, hat mehrere nennenswerte Einzelkomponenten. Diese sind der eingebundene Videostream, die Chatfunktion und die Nutzertasks.

Der Videostream wird über RTMP und NGINX gehostet und anschließend durch den JW Player auf der Website eingebunden und abgespielt.

Die Chatfunktion wird über JavaScript ermöglicht, damit wird über AJAX mit PHP kommuniziert und die Textnachrichten auf dem Server abgespeichert und anschließend auf der Seite dargestellt. Die Chataktivität wird durch „chatUI.js“ gehandhabt. Damit werden die eingegebenen Nachrichten an „process.php“ weitergeleitet, welches anschließend die Daten auf dem Server hinterlegt. Dasselbe gilt auch für die Übertragung der Daten vom Server auf die Webseite, nur in umgekehrter Reihenfolge.

Die Taskfunktion funktioniert ähnlich wie der Chat, wobei ebenfalls über JavaScript, AJAX und PHP, die Daten zu den Tasks auf dem Server hinterlegt werden und anschließend für die Darstellung auf der Website ausgelesen werden. Dies wird von „taskmanager.js“, „createNewTest.js“ und „loadExistingTest.js“, in Verbindung mit „loadTask.php“, „loadTest.php“ und „deleteTest.php“ gesteuert, wobei somit Tasks angelegt werden und diese später dann verwendet oder gelöscht werden können.

6.3 Evaluation

Die Evaluation ist ein iterativer Prozess, welcher von Beginn an durchgeführt wurde, um flexibler auf Feedback reagieren zu können. Die ersten spezifischen Evaluationen wurden mit Mitkommilitonen durchgeführt, die im Laufe ihres Studiums schon einige Usability Tests absolviert haben. Daher ist das Feedback dieser Probanden signifikant und relevant, da diese schon Erfahrungen mit Usability Tests haben und potentielle Nutzer sind. 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.

6.3.1 Evaluation 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 wurden 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 beinhaltet. Das Feedback 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 den 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 das Skript einzubinden, nie zuvor OBS benutzt haben.

Aufgabe 5:

Alle Probanden konnte ohne Probleme Annotationen hinzufügen.

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 Anwen-

derung von verschiedenen Nutzergruppen benutzt werden soll, ist die Anwendung selbst sowie das Tutorial relativ simpel gestaltet, damit jeder – auch medial weniger bewandte Personen – diese Anwendung nutzen kann.

6.3.2 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 konnte selbständig die Anwendung untersuchen und sollte bei sich Bedarf an den User Stories orientieren bzw. die Leiter fragen. Der Stakeholder explorierte alle Funktionalitäten, 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, welches reibungslos ablief. Ebenfalls wurde die Ansicht der Testperson exploriert, spezifisch wurde das Setup des Streams selbst 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 6.1.2) vom Probanden getestet werden, dabei konnten alle Funktionalitäten gefunden sowie getestet werden. Der Stakeholder war positiv überrascht und konnte keine signifikanten Mängel feststellen. Dennoch wurden einige Punkte 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 zusammenfassen, dass der Stakeholder sehr zufrieden mit der Anwendung ist, da alle Anforderungen sowie Funktionalitäten erfüllt und vorhanden sind. Die oben genannten Punkte wurden beim Endprodukt berücksichtigt.

7 Setup-Anweisungen

In diesem Abschnitt wird kurz erläutert, wie die jeweiligen Komponenten konfiguriert bzw. hinzugefügt werden müssen, um die Anwendung ordnungsgemäß zu starten.

7.1 Setup Server

Verwendetes Operation System

Als Operation System wird die Ubuntu Version 18.04 empfohlen. Sämtliche Tests und nachfolgenden Installationsanweisungen wurden unter oben genannten OS durchgeführt.

Installation NGINX

Als Streaming Protokoll wurde in diesem Projekt RTMP (Real Time Messaging Protocol) verwendet, welches von der Firma Adobe Systems entwickelt wurde und Daten von einem Media Server an einen Flash-Player überträgt. Als Webserver, welches dieses Protokoll verwaltet wurde NGINX verwendet. Als erster Schritt der Serverkonfiguration folgt im nachstehenden die Installation eines NGINX Servers.

- **Schritt 1: Installation NGINX**

```
$ sudo apt update
$ sudo apt install nginx
```

- **Schritt 2: Konfiguration RTMP**

Öffnen Sie die die Konfigurationsdatei „nginx.conf“ und fügen sie am Ende der Datei folgende Zeilen ein:

```
rtmp {
    server{
        listen 1935;
        chunk_size 4096;
        application live {
            live on;
            record off;
        }
    }
}
```

Ebenfalls sollte der unter dem Punkt „http“ hinterlegt Port „80“, auf einen anderen Port geändert werden, da es sonst zu Komplikationen mit dem Verwendeten Apache2 Webserver kommt. In unseren Testläufen wurde der Port des NGINX Servers auf „81“ geändert. Wurden alle Änderungen an der „nginx.conf“ Datei vorgenommen, speichern und schließen Sie diese und starten anschließend den NGINX Server neu.

```
$ sudo systemctl restart nginx
```

Nachdem diese Schritte unternommen wurden kann über den installierten und Konfigurierten NGINX Server bereits gestreamt werden.

Hierfür müssen Sie lediglich im Open Broadcaster System (OBS) im Menu Datei auswählen. Dort die Einstellungen öffnen und anschließend den Reiter Stream auswählen.

Dort legen sie folgenden Eigenschaft fest:

Stream Type: Custom Streaming Server

URL: `http://[IP/URL_Ihres_Servers]/live`

Streamkey: Hier geben Sie ihren gewünschten Streamkey an. Wichtig dieser Key muss um Code anschließend hinterlegt werden, da sonst eine Kommunikation zwischen JW Player und Server nicht möglich ist. Die entsprechende Codezeile finden sie in der Datei „video.js“:

```
clip: { url: 'rtmp://[IP/URL_Ihres_Servers]/live/Ihr_Streamkey', autoplay: true, provider: 'rtmp' }
```

Installation Apache2 / PHP 7.2

Als Webserver wurde Apache2 genutzt, welchen sie mit folgenden Befehlen installieren können.

```
$ sudo apt update
$ sudo apt install apache2
```

Ubuntu 18.04 ermöglicht eine einfache Verbindung von PHP mit dem installierten PHP Server.

Diese erreichen Sie mit folgendem Befehl:

```
$ sudo apt install php libapache2-mod-php
$ sudo systemctl restart apache2
```

Nach diesen Schritten sollten sämtliche Voraussetzungen für den Webserver erfüllt sein.

Installation MySQL

Folgende Befehle sind nötig um MySQL auf Ihrem Server zu installieren.

```
$ sudo apt update
$ sudo apt install mysql-server
```

Anschließend wird empfohlen noch folgende Sicherheitskonfigurationen durchzuführen

```
$ sudo mysql_secure_installation
```

Alle hier nachfolgenden Fragen in der Konsole können mit Y beantwortet werden.

- **Anpassung User Authentication:**

```
$ sudo mysql
mysql> SELECT user, authentication_string,plugin,host FROM
mysql.user;
mysql> ALTER USER ,root'@'localhost' IDENTIFIED WITH
mysql_native_password BY ,your_password';
mysql> FLUSH PRIVILEGES;
mysql> SELECT user,authentication_string,plugin,host FROM
mysql.user;
mysql> exit
```

Installation phpMyAdmin

Als finaler Schritt der Serverkonfiguration fehlt noch die Installation von phpMyAdmin um somit die Datenbank leichter zu verwalten.

```
$ sudo apt update
$ sudo apt install phpmyadmin php-mbstring php-gettext
$ sudo phpenmod mbstring
$ sudo systemctl restart apache2
```

Nach erfolgreicher Installation von phpMyAdmin, muss über diesen noch eine Tabelle in der Datenbank angelegt werden.

Hier wird lediglich eine Tabelle mit dem Namen **Tests** benötigt. Diese Tabelle benötigt folgende Spalten:

- **TestID** (Primary Key und Auto-Increment)
- **Json** (Text / maxLength)

Nachdem all diese Schritt ausgeführt worden sind ist der Server konfiguriert. Abschließend muss die bereitgestellte Homepage nur noch in das richtige Verzeichnis kopiert werden und ab da ist sie lauffähig.

Verzeichnis:

var/www/html

Im folgendem werden alle Quellen angegeben nach deren Anleitung der Server konfiguriert wurde. Sollten bei Ihrer Konfiguration Fehler auftreten, ziehen Sie bitte diese Quellen zu Rate.

Quellen

1. <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04>
2. <https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-18-04>
3. <https://linuxize.com/post/how-to-install-php-on-ubuntu-18-04/>
4. <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04>
5. <https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-18-04>
6. <https://obsproject.com/forum/resources/how-to-set-up-your-own-private-rtmp-server-using-nginx.50/>

7.2 Setup Keylogger

Auf Seiten der Testperson muss außerdem noch folgendes Setup durchgeführt werden, um das Loggen der Tastatur Tastenevents während des Tests zu ermöglichen.

Da das Logging über ein in OBS integriertes Plugin, in Form eines Pythonscripts funktioniert, wird zum Verwenden von Diesem eine Python Installation benötigt. Das Plugin wurde unter Python 3.6+ erstellt, daher sollte eine Version 3.6.x installiert werden. Alle Pythonreleases sind verfügbar unter: <https://www.python.org/> .

Das Plugin benötigt des Weiteren verschiedene Python Module, welche ebenfalls installiert werden müssen. Diese sind „**pyHook**“ und „**pywin32**“.

Für die Installation von „**pywin32**“ ist das Paketverwaltungsprogramm „**pip**“ erforderlich, dieses ist aber in Python Releases > 3.4 automatisch vorhanden.

Jetzt startet man die Eingabeaufforderung (je nach Zugriffsrechten des Nutzers, als Admin) und gibt dort

```
$ pip install pypiwin32
```

ein. Daraufhin installiert sich pywin32 von selbst.

Die Installation von „**pyHook**“ ist etwas aufwendiger. Zuerst muss die passende Version heruntergeladen werden. Alle Versionen sind hier verfügbar:

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyhook> .

Dort muss nach den Downloads für PyHook gesucht werden. Hat man diese gefunden muss man die richtige Version für die eigene Python Installation wählen. Die Benennung der Versionen liegt in folgender Form vor: „**pyHook-1.5.1-cp27-cp27m-win32.whl**“. Dabei steht die Zahl hinter „**cp**“ für die eigene Python Version. Basierend daraus muss dann die richtige Version mit

der richtigen Bitzahl des Betriebssystems (32 oder 64) gewählt werden. (Bei einer 64Bit Version mit Python 3.6+ also z.B. „pyHook-1.5.1-cp36-cp36m-win_amd64.whl“).

Anschließend über die Eingabeaufforderung zum heruntergeladenen File navigieren und dieses mit „**pip install FILENAME**“ installieren. (Bei obigem Filebeispiel also mit

```
$ pip install pyHook 1.5.1 cp36 cp36m win_amd64.whl ) .
```

Nachdem alles erfolgreich installiert wurde, muss auf der Usabilitytestingseite das dazu nötige Script heruntergeladen werden. Dies findet man unter:

<http://167.99.248.108/src/keylogger.py>

Jetzt kann OBS gestartet werden. Dort muss jetzt das Plugin integriert werden.

Dort in der Taskleiste unter „Werkzeuge“, den Reiter „Skripte“ auswählen und mit dem „+“-Icon das vorher herunter geladene Skript einfügen.

Bei der Durchführung der Tests kann jetzt das Plugin verwendet werden. Dazu vor dem Start des Videostreams erneut zu den Scripten navigieren und dort den Keylogger auswählen. Danach einen Pfad für das Logfile auswählen und anschließend auf „Starte Keylogger“ klicken. Nach dem Beenden des Tests muss der Keylogger auch wieder deaktiviert werden. Dazu zuerst den Stream beenden und dann die selben Schritte wiederholen, aber dieses Mal auf „Beende Keylogger“ klicken. Dabei schließt sich dann auch OBS automatisch.

8 Ausblick

Die aktuelle Version der Usabilitytestingsoftware ist im vorliegenden Stand funktionsfähig und erfüllt die zu Beginn des Projekts aufgestellten Anforderungen. Allerdings gibt es dennoch einige Features oder Funktionen, die im weiteren Verlauf der Software durch andere Projektteams weiterentwickelt oder hinzugefügt werden könnten, um Komfort oder Ease of Use weiter zu verbessern.

Im jetzigen Zustand kann die Software bereits Problemlos für Tests verwendet werden, aber dennoch werden sich wohl im Laufe der Zeit neue Anforderungen durch die Tests, oder die Verwendung am Lehrstuhl, ergeben.

Die Software sollte daher eher als Grundstein gesehen werden, an dem im Zeichen von Open Source weiter gearbeitet werden kann und soll. Daher ist auch eine Implementierung komplett neuer Funktionen denkbar und die Software kann, je nach Umfang der geplanten Tests, beliebig erweitert werden.

Dazu jetzt ein Paar Themen, welche in Zukunft erweitert werden könnten. Dies beinhaltet Dinge, die bereits als „nice to have“ während der Entwicklung der Software galten, aber es aus

Zeitgründen nicht in die finale Version geschafft haben, oder im Nachhinein als sinnvoll betrachtet werden.

Keylogger

Die Installation bzw. das Setup des Keyloggers ist noch relativ langwierig und kompliziert für nicht mit der Materie vertraute Nutzer. Da der Logger mit Python implementiert wurde und dieses nicht standardmäßig auf Windows PCs installiert ist, ist dort eine manuelle Installation erforderlich.

Dieser Prozess könnte entweder vereinfacht werden, oder durch ein komplettes Umschreiben des Keyloggers in z.B. C++ gelöst werden. Dann muss allerdings auch das Plugin komplett überarbeitet werden.

Auch wenn der Keylogger so beibehalten und weiterhin mit Python verwendet wird, kann der Funktionsumfang dort erweitert werden. Ein Implementieren von Logging ganzer Wörter statt einzelner Buchstaben wäre denkbar, und dem Aufbau des geloggten Files sind keine Grenzen gesetzt um die Lesbarkeit / Verwendbarkeit der Daten zu verbessern.

Eine Implementation von Mausklickevents wäre ja nach Art der Tests ebenfalls eine Option.

Testfortschritt speichern

In der aktuellen Version werden die während des Testverlaufs erfassten und in .txt-Files gespeicherten Daten beim Schließen der Webseite finalisiert. Ein Wiederaufnehmen des Tests ist daher zwar möglich, allerdings werden dann auch neue Dateien erzeugt.

Es wäre daher durchaus sinnvoll eine Unterbrechung des Tests zu erlauben, z.B. falls die Seite sich unerwartet schließt, eine Pause gemacht wird, oder der Testleiter einen Fehler macht.

Facetime / Sprachchat

Aktuell ist es der Testperson und dem Testleiter nur möglich, über den auf der Webseite integrierten Chat zu kommunizieren.

Der Testleiter sieht zwar die Testperson über den Stream und hört was diese sagt, sofern Sie eine Webcam verwendet, kann aber umgekehrt nicht mit ihr kommunizieren.

Eine Implementierung eines Sprach-/Videochats könnte deshalb eventuell von Vorteil sein.

Umstellen des Streaming Protokolls

Auf dem aufgesetzten Server wird aktuell das RTMP Protokoll zum Streamen des Videos verwendet.

An sich funktioniert dieses auch ohne Probleme, allerdings basiert es in seinen Ursprüngen auf Flash. In den letzten Jahren hat Flash allerdings viel seiner Beliebtheit eingebüßt und wird teilweise auch von besseren und neueren Technologien abgelöst. Daher könnte die aktuelle Implementierung des Videostreams evtl. in einiger Zeit veraltet sein, allerdings ist aktuell die RTMP Variante in Verbindung mit OBS, die am meisten verwendete, am beste dokumentierte und auch besten erprobte Möglichkeit.

Eine Lösung dafür wäre es, den Streaming Server auf das neue und Zukunftssichere HLS Protokoll umzustellen.

Ermöglichen von Serientests

Momentan werden für jeden Test einzeln Dateien angelegt und die Testdaten darin gespeichert.

Es wäre aber durchaus sinnvoll, diese Funktion durch das Erlauben von Serientests zu erweitern. Konkret heißt das, dass die gesammelten Daten von mehreren Testpersonen die den selben Test durchlaufen, vom System bereits automatisch, z.B. in einem Order für den jeweiligen Test hinterlegt werden würden.

Das würde das Verwalten der Daten erleichtern und die Übersicht über die Dateien verbessern.

Importfunktion für das Logfile des Keyloggers

Das auf dem PC der Testperson angelegte Logfile mit den Keyevents zu den Tests, wird aktuell von der Testperson, nach dem Test, per E-Mail an den Testleiter weitergeleitet. Dies könnte für erhöhten Komfort, in Zukunft, auch über ein Feature auf der Webseite erfolgen.

Quellen:

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

Schatten, A., Biffel, S., Demolsky, M., Gostischa-Franta, E., Östreicher, T., & Winkler, D. (2010). *Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Springer-Verlag

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]