OST
Ostschweizer
Fachhochschule

Campus Rapperswil-Jona
8640 Rapperswil

Studienarbeit

Automated Lesson-Feedback

Testat-Abgabe-Checker

Herbstsemester
2021

Teamname: SA ALF

Autor: Roman Spring
Pascal Gsell

Betreut von: Thomas Corbat
Nicola Jordan

Abgabe: 24.12.2021

Version: 1.0

Abstract

Problem

Im Modul Cpp der OST können Studierende ihre Lösungen von Übungsaufgaben mit einem automatisierten Prototyp-Tool überprüfen lassen. Aktuell ist leider die Verwaltung dieser Aufgaben für die Dozierenden nicht direkt zugänglich. Die Erfassung neuer oder die Anpassung bestehender Übungen ist nur von einem Administrator durchführbar.

Ziel

Der existierende Prototyp des ALF-Tools soll durch eine verbesserte Version abgelöst werden. Die bewährte Funktionalität soll für die Studierenden erhalten bleiben. Zudem sollen sie die Möglichkeit haben, ihre Abgaben zur Korrektur einzureichen. Für die Dozierenden soll die Verwaltung der Aufgaben und der Zugriff auf die abgegebenen Lösungen über eine Benutzerschnittstelle ermöglicht werden.
In der Entwicklung soll auch darauf geachtet werden, dass das Tool skalierbar ist. Zudem soll das Tool mit dem Gedanken an weitere Programmiersprachen entwickelt werden.

Methode / Vorgehen

Um das Ziel dieser Arbeit zu erreichen, wird neben dem Aufstellen und Analysieren der Anforderungen eine Analyse des alten ALF-Tools durchgeführt. Um frühzeitig Technologieprobleme angehen zu können, wird zu Beginn ein Durchstich durch alle Technologien gemacht. Dieser Durchstich soll folgendes beinhalten; Docker-Compose, Einreichung einer Abgabe auf dem Frontend, eine Verarbeitung eines leeren CMake-C++-Cute Projekts im Hintergrund und einer visuellen Anzeige der Antwort auf dem Frontend. Der Durchstich wird im Laufe des Projekts stets mit weiteren Features ausgestattet, bis zum aktuellen Stand des Produkts.

Wesentliche Ergebnisse

Auf dem entwickelten ALF-Tool können nun Dozierende Projekte selbst anlegen, diese modifizieren und testen. Die Studierenden können, nachdem sie eine Abgabe getätigt haben, alle ihre hochgeladenen Abgaben einsehen. Neben der Übersicht über die bestanden Unit-Tests, können die Studierenden ihre hochgeladenen Abgaben auch wieder herunterladen. Die Dozierenden sehen über ihre Übersicht den Stand des Projekts und können alle Abgaben auf einmal herunterladen. Zusätzlich können sie die Resultate der einzelnen Abgaben direkt über das Tool betrachten.
Das ganze Tool wurde so implementiert, dass eine Erweiterung durch zusätzliche Sprachen ohne grössere Schwierigkeiten und Anpassungen möglich ist. Somit steht der Verwendung in anderen Modulen nichts mehr im Weg.
Die Implementation wurde mit den folgenden Technologien umgesetzt; Python, Django, Bootstrap, SQLite, Traefik und Docker-Compose.

Empfehlungen

Das Projekt benötigt einen praktischen Einsatz. Dazu sollte es im bereits implementierten Modul *Cpp* praktisch eingesetzt werden. Damit würden weitere Erfahrungen gewonnen und das Tool würde eine erste Etablierung erfahren. Weiter sollten mehrere Projekttypen implementiert werden, sodass das Tool auch in anderen Modulen verwendet werden kann. Ebenfalls ist es anzudenken, dass das Tool in Moodle integriert wird.

Schlüsselwörter

ALF, automated lesson-feedback, Python, Django, Docker-Compose, Cpp, OST, Studienarbeit, Traefik, Bootstrap, SQLite

Management Zusammenfassung

Ausgangslage

Das aktuelle ALF-Tool (Automated Lesson-Feedback) wurde für das Modul Cpp entwickelt. Dieses Tool ermöglicht es den Studierenden für ein Projekt, zum Beispiel ein Testat, ihre Lösungen hochzuladen. Das Tool wertet den hochgeladenen Sourcecode anhand der zuvor definierten Unit-Tests aus. Die Studierenden bekommen daraufhin mitgeteilt, welche der Tests erfolgreich sind und welche nicht. Somit können sie sicher sein, dass sie die Aufgabe erledigt haben. Anschliessend müssen die Studierende ihre Lösungen noch zusätzlich dem Dozenten abgeben. Das bestehende Tool hat mehrere schwerwiegende Probleme, welche im nachfolgenden aufgeführt sind.

Statisches Tool
Falls der Dozent ein zusätzliches Projekt erstellen will, muss der Administrator des Tools einen Teil des Sourcecodes kopieren und diesen direkt im Code auf das neue Projekt anpassen. Dies verhindert zum einen, dass der Dozent selbständig Projekte erstellen und verwalten kann. Zum anderen wird durch das Kopieren des Sourcecodes das Tool immer weiter aufgeblasen und es erschwert erheblich die Wartung.

Ungetesteter Sourcecode
Bei der Erstellung des ALF-Tools wurde auf Unit-Tests verzichtet. Durch den Verzicht der Tests, kann deshalb bei einer Änderung des Sourcecodes nicht sichergestellt werden, dass dieser immer noch richtig funktioniert. Der Verzicht auf die Tests ist darauf zurückzuführen, dass das Tool eher als eine schnelle Lösung implementiert wurde.

Keinen Clean-Code
Da das Tool schnell und eher als ein Prototyp geschrieben wurde, entspricht der Sourcecode nicht unbedingt der gängigen Clean Code Regeln.

Beschränkung auf C++
Das Tool ist für die Sprache C, beziehungsweise C++, geschrieben worden. Eine Ausweitung auf andere Sprachen ist durch die statische Form fast nicht möglich.

Nachträgliche Betrachtung
Den Studierenden ist es nicht möglich vergangene Auswertungen zu betrachten. Sie können lediglich die aktuelle Auswertung einsehen. Dies erschwert ihnen den Überblick über ihre Fortschritte zu behalten.

Zusätzliche Abgabe für Studierende
Die Studierenden müssen nach dem Überprüfen mit dem Tool ihren Sourcecode zusätzlich dem Dozenten über eine andere Plattform einreichen.

Ergebnisse

In diesem Abschnitt werden die Umsetzungen der einzelnen Probleme vom Abschnitt Ausgangslage aufgeführt.
Die nachfolgenden Abbildungen beinhalten Musterwerte und können auch leicht vom endgültigen Resultat der Arbeit abweichen.

Statisches Tool

Durch die dynamische Programmierung des Tools ist es nun möglich, dass ein Dozent selbständig ein Projekt anlegen und dieses modifizieren kann. In der nachfolgenden Abbildung (Abbildung 1. Ändern eines Projekts) ist ersichtlich wie das Formular für das Ändern der Konfiguration eines Projekts aussieht.

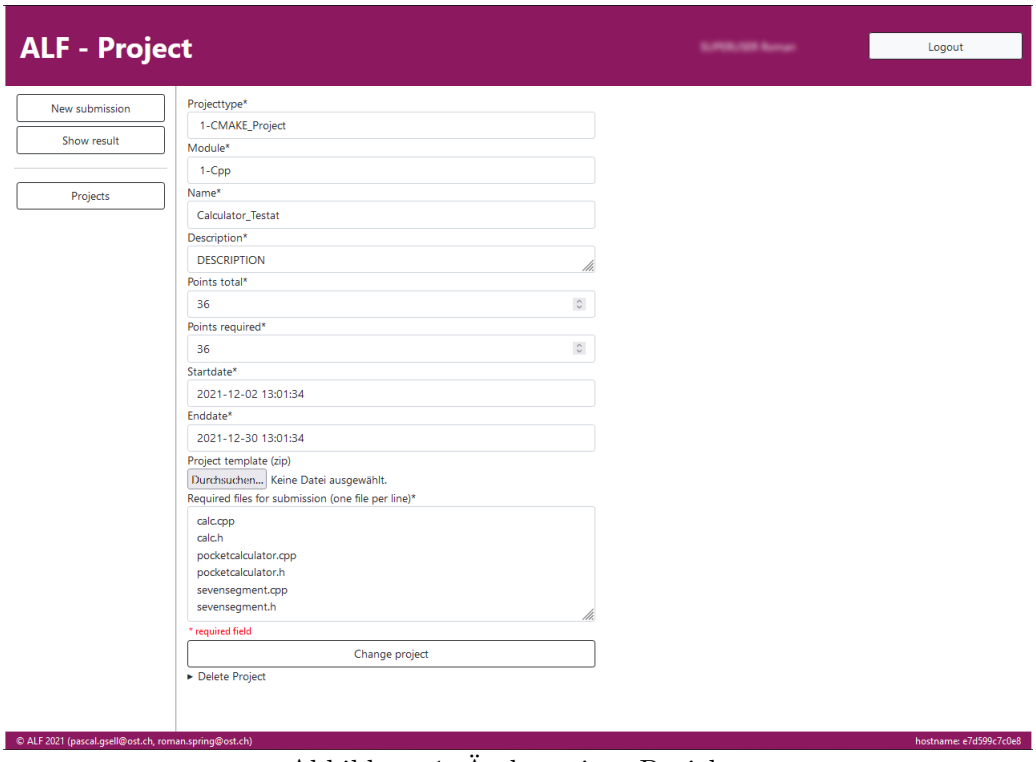


Abbildung 1: Ändern eines Projekts

Ungetesteter Sourcecode

Während der Entwicklung wurde stets auf eine gute Abdeckung durch Unit-Tests geachtet. Dadurch werden von 825 geschriebenen Statements alle 825 durch Tests abgedeckt. Die genaue Aufschlüsselung kann unter Abschnitt 6.4. Test-Coverage nachgelesen werden.

Keinen Clean-Code

Durch regelmässige Reviews im Entwicklerteam wurde versucht den ALF-Sourcecode so gut strukturiert und verständlich wie möglich zu halten. Dies soweit es die Erfahrung und aktuelle Ausbildung zugelassen hat.

Beschränkung auf C++

Das Tool wurde so implementiert, dass es theoretisch ohne grössere Umstände auf weitere Sprachen erweitert werden kann. Dafür genügt es einen Runner und ein Parser für die neue Sprache zu schreiben und einzupflegen. Dies ist unter dem Abschnitt B.3. Zusätzliche Projekttypen ersichtlich.

Nachträgliche Betrachtung
In der neuen Implementierung des Tools werden alle abgegebenen Dateien auf dem Dateisystem des Servers gespeichert. Zusätzlich werden die Ergebnisse der Auswertung in einer Datenbank abgespeichert. Dies ermöglicht es den Studierenden alle ihre Abgaben jederzeit wieder einzusehen und die abgegebenen Dateien auch herunterzuladen. Die Studierenden bekommen für jedes Projekt eine Auflistung ihrer Abgaben und können sich aussuchen, welche sie ansehen möchten. Dies ist in der nachfolgenden Abbildung (Abbildung 2. Auswahl einer Abgabe) zu sehen.

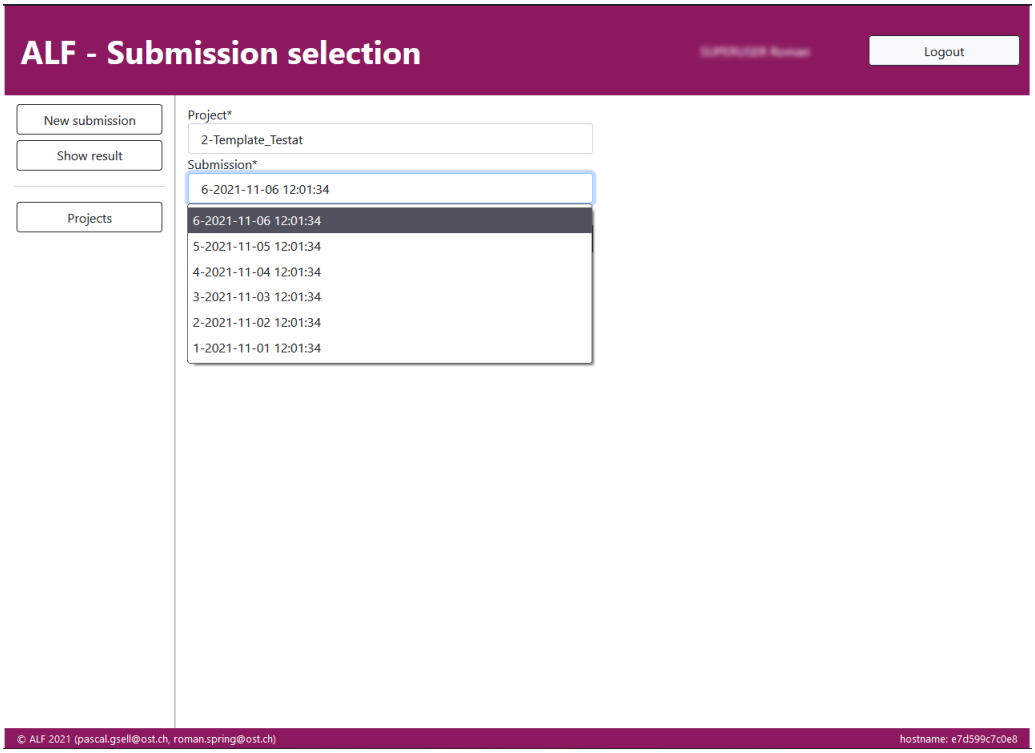


Abbildung 2: Auswahl einer Abgabe

Nach der Wahl der Abgabe bekommen die Studierenden eine Abgabenübersicht und können die hochgeladenen Dateien wieder herunterladen. Dies ist in der untenstehenden Abbildung (Abbildung 3. Ansicht der Resultate einer Abgabe) ersichtlich.

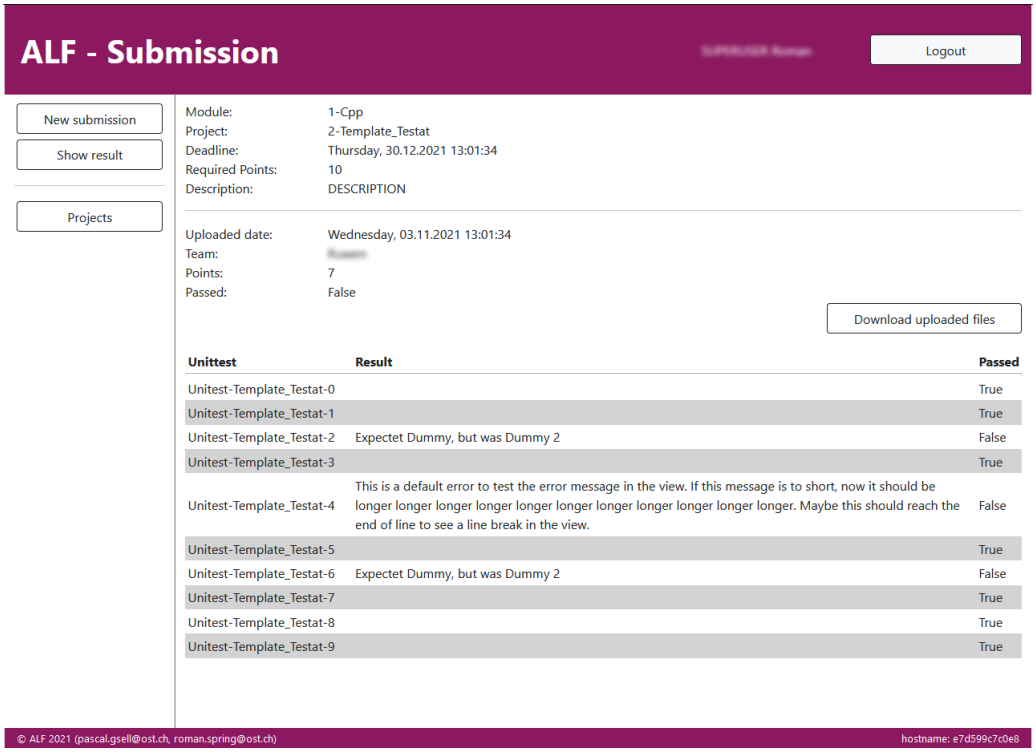


Abbildung 3: Ansicht der Resultate einer Abgabe

Zusätzliche Abgabe für Studierende
Alle Abgaben werden auf dem Server und in der Datenbank gespeichert. Damit ist es dem Dozenten auch möglich diese jederzeit und auch nach dem Projektende einzusehen. Somit entfällt die zusätzliche Abgabe für die Studierenden über eine zusätzliche Plattform. Der Dozent kann sich entweder die Abgaben einzeln herunterladen oder er lädt alle Abgaben eines Projekts auf einmal herunter. Auf der folgenden Abbildung (Abbildung 4. Ansicht aller Abgaben für einen Dozenten) ist die Übersicht eines Dozenten über alle Abgaben eines Projekts zu sehen.

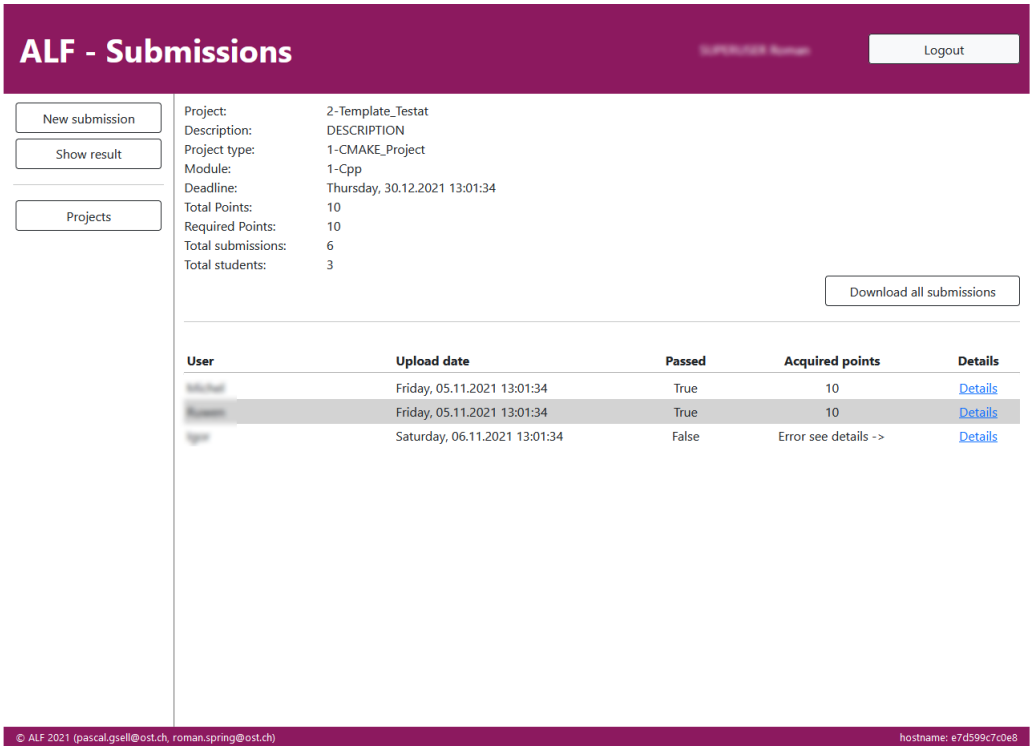


Abbildung 4: Ansicht aller Abgaben für einen Dozenten

Danksagung

An dieser Stelle möchten wir all jenen danken, die durch Ihren Einsatz zum erfolgreichen Gelingen dieser Studienarbeit beigetragen haben.

Zuerst gebührt unser Dank unseren beiden Betreuern Thomas Corbat und Nicola Jordan, welche uns jederzeit unterstützt und hilfreiche Inputs gegeben haben. Auch bedanken wir uns für Ihr Verständnis, wenn etwas nicht so funktioniert hat wie erhofft, auch nach einem erneuten Hinweis darauf.

Für die Korrekturlesungen bedanken wir uns bei Alexandra Gimmi, David Gsell, Celine Müller und Fränzi Arnold.

Zudem wollen wir uns bei Martin Kunz für das Erstellen unseres Logos bedanken. Ohne seinen Einsatz wäre das Logo nur ein einfacher Text geworden.

Historie

Version	Datum	Beschreibung	Wer
0.1	20.09.2021	**Dokumentation Initialversion**	Team
1.0	24.12.2021	**Dokumentation Abgabeversion**	Team

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Hinweise

Hier wird der eine oder andere Hinweis bezüglich dieser Dokumentation aufgeführt.

Referenzen
Die Referenzen sind Dokumente, welche im Allgemeinen zur Hilfe genommen wurden, jedoch nicht eindeutig einem Kapitel oder Abschnitt zugeordnet werden können.

- Guidelines Dokumentation [1], siehe Abschnitt F.3
- Leitfaden für Bachelor- und Studienarbeiten [2], siehe Abschnitt F.4
- Informationen zur SA-Abgabe HS21 [3], siehe Abschnitt F.5
- Studienarbeiten: Termine Herbstsemester 2021/22 [4], siehe Abschnitt F.6
- Python 3.10.0 Dokumentation [5]
- Pytest 6.2.x Dokumentation [6]
- Django 4.0 Dokumentation [7]
- PEP8 Dokumentation [8]

Gendering
Für die bessere Lesbarkeit und Verständlichkeit wird innerhalb dieser Dokumentation auf das Gendern verzichtet. Wegen ihre kürzere und prägnantere Verwendung wird die Männliche Form verwendet. Selbstverständlich sind bei der Verwendung der männlichen Bezeichnung alle Geschlechter gleichermassen angesprochen und gemeint.

Kapitel 1

Einleitung

Zu Beginn des Projekts muss die Ausgangslage, sowie der Auftrag definiert werden. Damit sich das Projekt nicht verläuft, benötigt es auch noch Rahmenbedingungen, welche das Projekt eingrenzen.

1.1 Ausgangslage

Die aktuelle Version des ALF-Tools wurde von Nicola Jordan mehrheitlich als Nebenprojekt, respektive Prototyp erarbeitet. Das Frontend wurde in Vue. Die Studenten können sich auf dem Server via GitLab-OST-OAuth einloggen. Nach dem Anmelden können sie ihre Lösungsdateien hochladen und erhalten dann eine Auflistung der Testresultate. Das Backend wurde mit Django entwickelt und bietet nur die allernötigsten Features. Die Schnittstellen zu den Funktionen wurde mittels OpenFaaS und die Services über Docker-Compose realisiert. Jedes Projekt musste durch Nicola Jordan einzeln in das Gesamtprojekt integriert werden. Dies bietet allerdings viele Möglichkeiten für Fehler und viele Anpassungen waren für neue Abgaben nötig. Aufgrund der Tatsache, dass die aktuelle Version ein Prototyp darstellt und das ALF-Tool eine grössere Verwendung mittels Moodle erreichen könnte, wurde dieses Projekt als Studien- oder Bachelorarbeit ausgeschrieben.

1.2 Aufgabe

In dieser Studienarbeit wird eine Neufassung des ALF-Tools behandelt. Das ALF-Tool soll einer Überarbeitung der bestehenden Applikation oder einer kompletten Neuimplementation unterzogen werden. Die bekannten Features sollen beibehalten werden und zusätzlich soll die Applikation um eine Administrationsschnittstelle erweitert werden. Eine angedachte Erweiterung wäre auch eine direkte Abgabemöglichkeit eines Testats über das ALF-Tool.

Die verwendeten Technologien von OpenFaaS, Docker und Python sollen eingearbeitet werden, wobei jedoch eine Neuimplementation nicht zwingend diese Technologien verwenden muss. Nicola Jordan begleitet dieses Projekt bezüglich Technologischen-Details.

1.3 Rahmenbedingungen

Dieses Projekt wird im Rahmen einer Studienarbeit im Herbstsemester 2021 durchgeführt. Die Studienarbeit an der Hochschule OST setzt dem Projekt gewisse Grenzen, welche eingehalten werden müssen.
Grenzen der Studienarbeit:

- **Arbeitsstunden**
  Eine Studienarbeit wird mit 8 ECTS-Punkte pro Studenten gewertet. Jeder ECTS-Punkt entspricht 30h, somit ergibt sich pro Studenten ein Soll-Stundensatz von 240h. Da diese Studienarbeit von zwei Studenten bestritten wird, ergibt sich somit einen Soll-Stundensatz von 480h. Die Arbeitszeiten sind über das ganze Projekt hinweg erfasst (siehe Abschnitt 7.3. Zeiterfassung).
- **Durchführungszeitraum**
  Eine Studienarbeit wird jeweils innerhalb eines Semesters geschrieben, sprich der erste Tag im Semester ist der Startzeitpunkt, sowie der letzte Schultag das Ende für das Projekt ist. In diesem Fall bedeutete dies, dass die Durchführung vom 20.09.2021 bis 24.12.2021 dauert.
- **Beratungsgespräch**
  Der Leitfaden gibt den Studierenden das Anrecht auf ein Beratungsgespräch pro Woche. Dieses Beratungsgespräch pro Woche wird auf Wunsch der Betreuer und Studierenden durchgeführt. Diese Gespräche werden jeweils protokolliert, die Protokolle befinden sich unter Anhang E. Sitzungsprotokolle.
- **Plakat und Abstract**
  In einer arbeit muss neben der Dokumentation jeweils auch ein Plakat und ein Abstract für die Veranschaulichung, beziehungsweise für die Plakatwände erstellt werden.
- **Abgabe**
  Auf Wunsch der Betreuer ist die Studienarbeit digital, sowie ein Exemplar als Ausdruck an Thomas Corbat abzugeben.

Kapitel 2

Analyse

Der erste Schritt in einem Projekt ist die Analyse. In diesem Kapitel wird die Aufgabenstellung durch eine Problemdomäne und den Systemkontext analysiert. Aus dieser Analyse sind die Anforderungen und Use-Cases definiert, welche das MVP konkretisieren. Während eines Projekts können Risiken auftreten, welche den erfolgreichen Abschluss des Projekts gefährden. Durch die Risikoanalyse ist diesen Risiken entgegengewirkt.

IDs in den Abschnitten
Die verwendeten IDs in einzelnen Abschnitten sind nach einem Schema vergeben. Die IDs bestehen aus drei Teilen. Die erste Bezeichnung ist für die Zugehörigkeit, die zweite grenzt diese durch eine Kategorie oder den Verwendungszweck noch mehr ein. Der letzte Teil der ID ist eine Laufnummer, welche innerhalb des Verwendungszwecks hochgezählt wird. Die Kürzel sind in den jeweiligen Kapiteln erklärt oder sie verstehen sich aus dem Kontext.

2.1 Problemdomäne

Da eine Ausführung auf einem Server selbst nicht gefordert ist, ist der Load-Balancer und der Reverse-Proxy kein Teil der Problemdomäne. Das Anmelden ist Teil der Problemdomäne des ALF-Tools, wobei noch unklar ist, ob die Authentifizierung vom Tool selbst oder durch ein Drittanbieter-Tool ausserhalb der Domäne geprüft wird. Die Problemdomäne ist in der nachfolgenden Abbildung (Abbildung 2.1. Problemdomäne) ersichtlich.

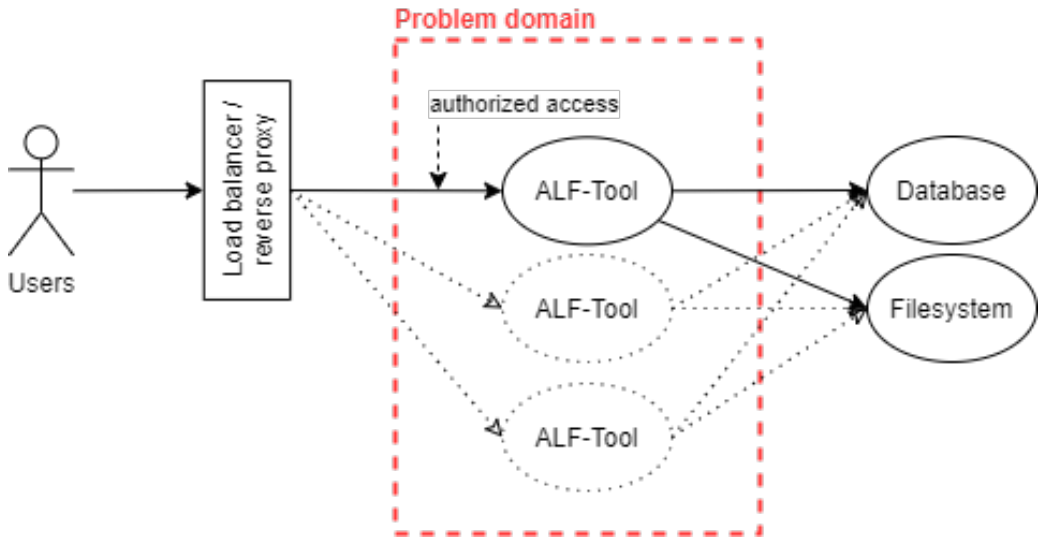


Abbildung 2.1: Problemdomäne

2.2 System-Kontext

Das Kontextdiagramm zeigt das Umfeld der Applikation auf. Bei dem in Abbildung 2.2. Systemkontextdiagramm gezeigten Kontextdiagramm sind der Unit-Coordinator und Job-Handle bereits getrennt, da diese ebenfalls eine zu realisierende Schnittstelle besitzen.

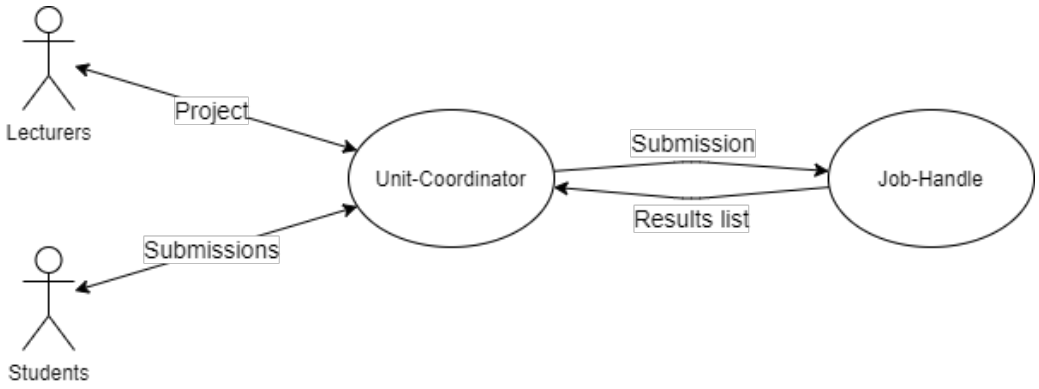


Abbildung 2.2: Systemkontextdiagramm

Die Akteure in diesem Projekt sind die Dozenten und Studenten der OST. Diese beiden Akteure verwenden das ALF-Tool. Vollständigkeit halber muss noch der Administrator erwähnt werden. Dieser ist für die Verwaltung der Datenbank und des Dateisystems zuständig.
Der Hauptbestandteil in der Applikation ist der Unit-Coordinator. Dieser bietet dem Dozenten eine Übersicht über bereits existierende Projekte und ermöglicht es neue Projekte einzurichten. Der Unit-Coordinator interagiert ebenfalls mit den Studenten. Diese können neue Abgaben einreichen und ihnen werden bereits eingereichte Abgaben angezeigt. Als ausführende Instanz ist der Job-Handle angedacht. Dieser kommuniziert ausschliesslich mit dem Unit-Coordinator und nicht direkt mit den Dozenten oder Studenten. Wünschenswert ist, dass sowohl der Unit-Coordinator wie auch der Job-Handle in Docker-Compose realisiert werden.

2.3 Minimum Viable Product - MVP

Das MVP stellt die minimale Applikation dar, welche die gewünschten Rahmenbedingungen einhält. In diesem Projekt wird das MVP durch die funktionalen und nicht funktionalen Anforderungen definiert, welche eine Gewichtung "MUSS" besitzen. Diese Definitionen sind unter Abschnitt 2.4. Anforderungen zu finden. Die Erreichung des MVP ist das Hauptziel dieses Projekts welches im Rahmen der Studienarbeit durchgeführt wird.
Ein ansprechendes Design gehört nicht zu den Anforderungen. Jedoch ist die Bedingung, dass das ALF-Tool gut zu bedienen und logisch aufgebaut ist.

2.4 Anforderungen

In den nachfolgenden Abschnitten wird auf unterschiedliche Anforderungen eingegangen. Die einzelnen Abschnitte sind nach den Anforderungsgruppen Allgemein, Student und Dozent unterteilt, wobei die Anforderungen mit Gewichtungen und Abhängigkeiten aufgelistet sind.

2.4.1 Funktionale Anforderungen

Nr.	Anforderung	Gewichtung	Abhängigkeit
Allgemein			
FA.A.01	Die Applikation wertet CMake-C++-Cute Projekte aus.	MUSS	—
FA.A.02	Das System wertet die Studenten-Lösungen aus.	MUSS	FA.A.01
FA.A.03	Alle Studenten-Lösungen werden gespeichert.	MUSS	—
FA.A.04	Dateien werden über ein Frontend hochgeladen.	SOLL	—
FA.A.05	Das System speichert alle Auswertungen.	SOLL	FA.A.02
Student			
FA.S.01	Der Student lädt seine Lösung einer Aufgabe hoch.	SOLL	—
FA.S.02	Der Student erhält eine Auswertung seiner Lösung.	SOLL	FA.S.01, FA.A.02
FA.S.03	Der Student lädt ein Testat zur Bewertung hoch.	KANN	FA.S.01, FA.A.02, FA.A.03
FA.S.04	Der Student kann mit anderen Studenten ein Team bilden für eine Abgabe.	KANN	—
FA.S.05	Der Student sieht bei Projekten die jeweiligen Deadlines.	KANN	—
Dozent			
FA.D.01	Der Dozent erstellt ein Projekt eigenständig.	MUSS	—
FA.D.02	Der Dozent löscht ein Projekt eigenständig.	MUSS	FA.D.01
FA.D.03	Der Dozent ändert seine Projekte eigenständig.	MUSS	FA.D.01
FA.D.04	Der Dozent sieht die Auswertungen der Aufgaben.	MUSS	FA.A.02
FA.D.05	Der Dozent erstellt ein Projekt über das Frontend.	SOLL	FA.A.04, FA.D.01
FA.D.06	Der Dozent löscht ein Projekt über das Frontend.	SOLL	FA.A.04, FA.D.02
FA.D.07	Der Dozent ändert seine Projekte über das Frontend.	SOLL	FA.A.04, FA.D.03
FA.D.08	Der Dozent sieht die Auswertungen der Aufgaben im Frontend.	SOLL	FA.A.02, FA.D.04
FA.D.09	Der Dozent lädt die hochgeladenen Abgaben herunter.	KANN	FA.A.03

Tabelle 2.1: Funktionale Anforderungen

2.4.2 Nicht funktionale Anforderungen

Nr.	Anforderung	Gewichtung	Abhängigkeit
Allgemein			
NFA.A.01	Nur die Applikation hat direkten Zugriff auf das Dateisystem und die Datenbank.	MUSS	—
NFA.A.02	Die Applikation läuft 24/7 mit einer maximalen Ausfallrate von 1% (Auf 1 Jahr ca. 3 Tage Ausfall).	SOLL	—
NFA.A.03	Die Applikation ist dockerisiert.	SOLL	—
NFA.A.04	Ein Fehler bei der Auswertung führt nicht zu einem Absturz der Applikation.	SOLL	FA.A.02
NFA.A.05	Die Applikation bricht die Berechnung der Resultate ab, wenn sie ein Zeitlimit überschreitet.	SOLL	NFA.A.04
Student			
NFA.S.01	Der Student wird autorisiert.	MUSS	—
NFA.S.02	Der Student sieht nur die Resultate seiner Abgaben.	MUSS	—
NFA.S.03	Der Student sieht nach spätestens 5 Sekunden eine Reaktion auf den Aufruf der Übersichtsseite.	KANN	—
NFA.S.04	Der Student erhält spätestens 5 Sekunden nach dem Hochladen einer Abgabe eine Reaktion.	KANN	—
NFA.S.05	Der Student sieht nach spätestens 5 Sekunden nach dem Time-Out der Berechnungen ein Resultat für die Abgabe.	KANN	NFA.A.05
Dozent			
NFA.D.01	Der Dozent wird autorisiert.	MUSS	—
NFA.D.02	Der Dozent kann nur eigene Projekte einsehen und ändern.	MUSS	—
NFA.D.03	Der Dozent sieht nach spätestens 5 Sekunden eine Reaktion beim Erstellen oder Ändern eines Projekts.	KANN	NFA.D.02
NFA.D.04	Der Dozent sieht nach spätestens 5 Sekunden eine Reaktion beim Aufruf der Übersichtsseite.	KANN	NFA.D.02

Tabelle 2.2: Nicht funktionale Anforderungen

2.5 Use-Cases / Szenarios

Die aufgeführten Use-Cases decken den Umfang der in Abschnitt 2.4. Anforderungen aufgeführten funktionalen und nicht funktionalen Anforderungen ab. Die aufgeführten Szenarios beschreiben eine angedachte Nutzung des Produkts.

2.5.1 Student lädt seine Lösung hoch

Use-Case	**Beschreibung**
ID	UC.S.1
Umfang	Unit-Coordinator
Primärer Akteur	Student
Stakeholder und Interessierte	Student: Möchte seine Lösung hochladen und erwartet eine Rückmeldung, ob das Hochladen funktioniert hat.
Vorbedingungen	• Der Student benötigt ein GitLab-OST-Konto. • Das Projekt wurde bereits erfasst.
Erfolgsgarantie	• Nach erfolgreichem Hochladen der Lösung, wird dem Studenten eine direkte Rückmeldung zum Hochladen seiner Lösung angezeigt. • Ist das Hochladen der Lösung nicht erfolgreich, wird dem Studenten eine Fehlermeldung angezeigt.
Hauptszenario (erfolgreich)	1. Der Student meldet sich in der Applikation an. 2. Der Student wählt das entsprechende Projekt aus. 3. Der Student wählt die erforderlichen Dateien seiner Lösung aus. 4. Der Student startet den Übermittlungsvorgang. 5. Die Applikation zeigt ihm das erfolgreiche Hochladen an.
Erweiterungen	5a. Die Applikation stellt fest, dass das Hochladen nicht korrekt ist oder etwas schief gegangen ist. I. Die Applikation bricht das Hochladen ab. II. Die Applikation zeigt dem Studenten eine Fehlermeldung an.
Häufigkeit des Auftretens	Da dies ein notwendiger Schritt zum Erreichen einer der Kernfunktionalitäten der Applikation ist, tritt dieser häufig auf.

Tabelle 2.3: Use-Case: Student lädt seine Lösung hoch

2.5.2 Student betrachtet die Resultate seiner Abgabe

Use-Case	Beschreibung
ID	UC.S.2
Umfang	Unit-Coordinator
Primärer Akteur	Student
Stakeholder und Interessierte	Student: Möchte die Resultate seiner Abgabe betrachten.
Vorbedingungen	• Der Student benötigt ein GitLab-OST-Konto. • Das Projekt wurde bereits erfasst. • Der Student hat eine Lösung hochgeladen.
Erfolgsgarantie	• Die Resultate der zuvor hochgeladenen Abgabe wird dem Studenten angezeigt.
Hauptszenario (erfolgreich)	1. Der Student meldet sich in der Applikation an. 2. Der Student wählt das entsprechende Projekt aus. 3. Die Applikation zeigt dem Studenten die Resultate seiner Abgabe.
Erweiterungen	3a. Der Student hat bereits mehrere Lösungen hochgeladen. I. Die Applikation zeigt dem Studenten die Resultate aller Abgaben an.
Häufigkeit des Auftretens	Dieser Use-Case wird häufig auftreten, da dies eine gewollte Rückmeldung der Applikation ist, welche dem Studenten zur Verfügung stehen sollte.

Tabelle 2.4: Use-Case: Student betrachtet die Resultate seiner Abgabe

2.5.3 Dozent erstellt ein Projekt

Use-Case	Beschreibung
ID	UC.D.1
Umfang	Unit-Coordinator
Primärer Akteur	Dozent
Stakeholder und Interessierte	Dozent: Möchte sein Projekt erstellen.
Vorbedingungen	• Der Dozent benötigt ein GitLab-OST-Konto und entsprechende Rechte.
Erfolgsgarantie	• Nach erfolgreichem Erstellen des Projekts wird dem Dozenten eine direkte Rückmeldung zum Erstellen des Projekts angezeigt. • Ist das Erstellen des Projekts nicht erfolgreich, wird dem Dozenten eine Fehlermeldung angezeigt.
Hauptszenario (erfolgreich)	1. Der Dozent meldet sich in der Applikation an. 2. Der Dozent konfiguriert das neue Projekt. 3. Der Dozent lädt die erforderlichen Dateien für das Projekt hoch. 4. Der Dozent speichert das neue Projekt. 5. Die Applikation zeigt ihm das erfolgreiche Speichern an.
Erweiterungen	4a. Die Applikation stellt fest, dass das Hochladen nicht korrekt ist oder etwas schiefgegangen ist. I. Die Applikation bricht das Hochladen ab. II. Die Applikation zeigt dem Dozenten eine Fehlermeldung an.
Häufigkeit des Auftretens	Projekte müssen nur selten neu erstellt werden, daher tritt dieser Use-Case nicht oft auf.

Tabelle 2.5: Use-Case: Dozent erstellt ein Projekt

2.5.4 Dozent ändert ein Projekt

Use-Case	Beschreibung
ID	UC.D.2
Umfang	Unit-Coordinator
Primärer Akteur	Dozent
Stakeholder und Interessierte	Dozent: Möchte ein existierendes Projekt ändern und erwartet eine Rückmeldung, ob die Änderungen übernommen wurden.
Vorbedingungen	• Der Dozent benötigt ein GitLab-OST-Konto und entsprechende Rechte. • Der Dozent hat bereits ein Projekt erstellt.
Erfolgsgarantie	• Nach erfolgreichem Ändern des Projekts wird dem Dozenten eine direkte Rückmeldung zur Änderung des Projekts angezeigt. • Ist die Änderung des Projekts nicht erfolgreich, wird dem Dozenten eine Fehlermeldung angezeigt.
Hauptszenario (erfolgreich)	1. Der Dozent meldet sich in der Applikation an. 2. Der Dozent wählt ein existierendes Projekt aus. 3. Der Dozent ändert die gewünschten Daten für das Projekt. 4. Der Dozent speichert das Projekt. 5. Die Applikation zeigt dem Dozenten die erfolgreiche Änderung an.
Erweiterungen	5a. Die Applikation stellt fest, dass die Änderung nicht korrekt ist oder etwas schiefgegangen ist. I. Die Applikation zeigt dem Dozenten eine Fehlermeldung an.
Häufigkeit des Auftretens	Projekte müssen häufig nach einem erfolgreichen Hochladen angepasst werden. Sobald aber ein Projekt korrekt eingerichtet ist, wird dieser Use-Case nur noch selten auftreten.

Tabelle 2.6: Use-Case: Dozent ändert ein Projekt

2.5.5 Dozent erhält die Resultate eines Projekts

Use-Case	Beschreibung
ID	UC.D.3
Umfang	Unit-Coordinator
Primärer Akteur	Dozent
Stakeholder und Interessierte	Dozent: Möchte von einem Projekt die Resultate der Studentenlösungen erhalten.
Vorbedingungen	• Der Dozent benötigt ein GitLab-OST-Konto und entsprechende Rechte. • Der Dozent hat bereits ein Projekt hochgeladen.
Erfolgsgarantie	• Die Resultate der Studentenlösungen werden dem Dozenten zur Verfügung gestellt.
Hauptszenario (erfolgreich)	1. Der Dozent meldet sich in der Applikation an. 2. Der Dozent wählt ein existierendes Projekt aus. 3. Der Dozent startet den Generierungsprozess. 4. Die Applikation übergibt dem Dozenten die Resultate.
Erweiterungen	3a. Die Applikation stellt fest, dass der Generierungsprozess nicht korrekt ist oder etwas schiefgegangen ist. I. Die Applikation zeigt dem Dozenten eine Fehlermeldung an.
Häufigkeit des Auftretens	Dieser Use-Case wird häufig auftreten, da Dozenten zu jeder Zeit während eines Projekts Auswertungen generieren.

Tabelle 2.7: Use-Case: Dozent erhält die Resultate eines Projekts

2.6 Bestehende Infrastruktur

Das Projekt "ALF - Automated Lesson-Feedback" wurde bereits Ende 2019 begonnen. Auf dieses Projektarchiv kann während der ganzen Projektlaufzeit zugegriffen werden. Das Projekt ist bereits auf einem Server aufgesetzt und in Betrieb. Bereits vieles ist in rudimentärer Form umgesetzt und kann gut als Nachschlagewerk verwendet werden. In Abbildung 2.3. Ansicht bestehendes ALF-Tool ist eine Ansicht des bestehenden ALF-Tools zu sehen, bei welcher gerade eine Abgabe getätigt wurde.

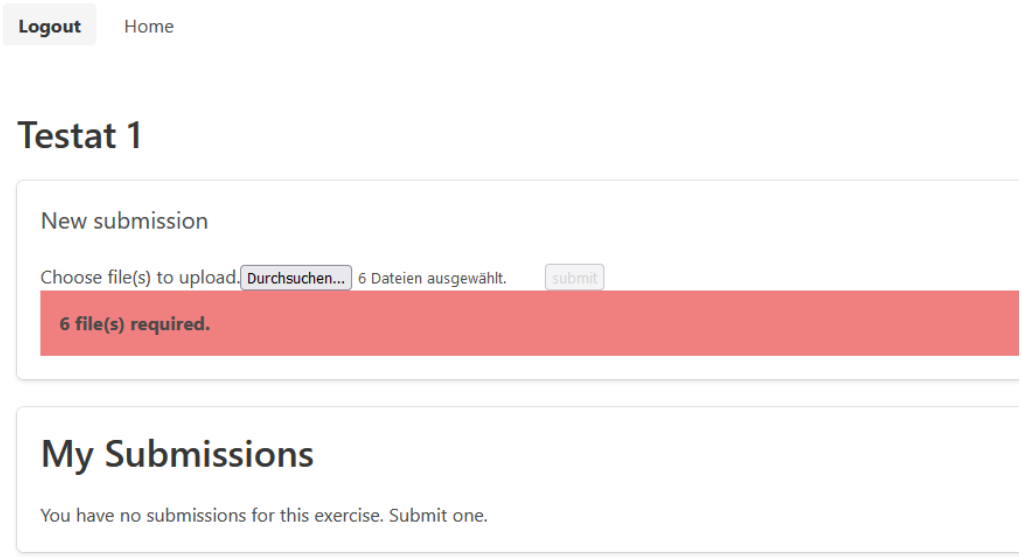


Abbildung 2.3: Ansicht bestehendes ALF-Tool

Als Frontend-Framework wird Vue eingesetzt, welches GitLab-OST für die Authentifizierung verwendet. Dieses regelt die Hauptansichten und verweist auf die Unterseiten. Als Backend-Framework wird Django eingesetzt, welches die hochgeladenen Dateien prüft und verarbeitet. Für jedes unterschiedliche Projekt ist ein eigenständiger OpenFaaS-Service eingerichtet, welcher jeweils bei einer neuen Abgabe gestartet wird. Als Datenbank wird PostgreSQL verwendet. Nginx wird als Reverse-Proxy eingesetzt.

2.7 Risikoanalyse

Bezeichnungen in den IDs:

U/M/P: Umwelt, Management oder Projekt

Risikoklassen:

1: Geringes Risiko 2: Normales Risiko 3: Mittleres Risiko 4: Hohes Risiko 5: Besonders hohes Risiko

Die Tabelle 2.10. Risikoanalyse - Bestimmung von Risiken und Klassifizierung zeigt die am meisten zu betrachtenden Risiken bei der Umsetzung des Projekts. Da dieses Projekt in Form einer Studienarbeit durchgeführt wird, ist der Spielraum im Umgang mit Risiken nur gering. Ebenfalls können aufgrund dieser Form keine weiteren Mitarbeiter hinzugezogen werden, um Rückstände aufzuholen, oder Teile der Arbeit an eine weitere Gruppe abzugeben. In der Tabelle 2.11. Risikoanalyse - Gegenmassnahmen und Vorgehen werden zu den genannten Risiken Gegenmassnahmen und Vorgehen aufgelistet.

Nummer	Beschreibung	Schaden	%	Klasse
R.U.01	Das GitLab-OST ist nicht erreichbar und es gibt keinen Zugriff auf die Daten.	0.5h	0.05	1
R.U.02	Häufige Änderung der Anforderungen oder Zielen.	4h	0.20	1
R.M.01	Die Konfiguration und / oder Projekt-Templates können nicht so generalisiert werden, dass sie durch die Dozenten selbst erstellt und / oder konfiguriert werden können.	1w	0.05	1
R.M.02	Die Leistungsanforderung für eine Reaktion des Frontends der Applikation kann nicht eingehalten werden.	1w	0.05	1
R.M.03	Es entstehen unvorhersehbare Zeiteinbussen.	1d	0.10	1
R.P.01	Der erste Meilenstein (Erster Durchstich) wird nicht erreicht und verschiebt sich nach hinten.	1w	0.20	2
R.P.02	Es werden unnötigen Features entwickelt.	0.5h	0.05	1

Tabelle 2.10: Risikoanalyse - Bestimmung von Risiken und Klassifizierung

Eine grössere Tragweite zeigt das Risiko R.P.01. Trifft dieses Risiko ein, kann die erfolgreiche Abgabe des MVP gefährdet sein. Alle weiteren Risiken haben eine tiefe Wahrscheinlichkeit oder können in absehbarer Zeit abgehandelt werden. Dabei wird eine realistische Projektplanung vorausgesetzt.

Nummer	Klasse	Gegenmassnahmen	Vorgehen bei Eintritt
R.U.01	1	Jeder Entwickler hat von jedem Projektarchiv einen lokalen Klon auf dem Computer.	Aufregen, fluchen und dann weiterarbeiten.
R.U.02	1	Frühzeitig Änderungen erkennen, mit Betreuern kontinuierlich besprechen und Moving-Targets kontinuierlich beobachten.	Gewünschte und / oder nötige Änderungen festhalten, Änderungen nach Aufwand und Möglichkeiten abschätzen und entsprechend umsetzen.
R.M.01	1	Frühzeitig Schnittstellen definieren und mit Betreuern diskutieren.	Mit Betreuern nicht realisierbare Probleme besprechen, mögliche Kompromisse erarbeiten und entscheiden.
R.M.02	1	Abläufe für die entsprechenden Reaktionen kurzhalten und bereits bekannte lange Operationen parallelisieren oder auslagern.	Mit Betreuern Reaktionszeiten besprechen und neue setzen oder langsame Abläufe parallelisieren, wenn möglich.
R.M.03	1	Bei der aktuellen Planung Reservezeit einplanen.	Verzögerung mit Betreuern besprechen und Gegenmassnahmen bestimmen.
R.P.01	2	Probleme bei der Umsetzung direkt mit Betreuern besprechen und nicht auf nächste Sitzung verschieben. Bei der Umsetzung auf nötige Features den Fokus setzten.	Die Verzögerung von weiteren Meilensteinen mit Betreuern diskutieren. Wenn nötig MVP anpassen oder andere Einsparungen bestimmen.
R.P.02	1	Vor einer zusätzlichen Funktionsimplementierung Notwendigkeit prüfen und bei einer Unsicherheit mit einem Teampartner besprechen.	Nicht notwendige Features können im Projekt verbleiben, sollten aber entsprechend dokumentiert werden, dass sie zurzeit nicht verwendet oder im Frontend eingebunden werden.

Tabelle 2.11: Risikoanalyse - Gegenmassnahmen und Vorgehen

Kapitel 3

Design

Bevor mit der Programmierung gestartet werden kann, müssen verschiedene grundsätzliche Punkte überlegt und definiert werden. Wie sieht eine normale Sequenz des Tools aus? Wie soll die Architektur des Tools aussehen? Wie sieht die Benutzeroberfläche aus? Wie werden die Daten persistiert? Diese Überlegungen sind in diesem Kapitel zusammengefasst.

3.1 Entwurf der Lösung

3.1.1 Projektstruktur

Die Projektstruktur (Abbildung 3.1. Projektstruktur) dient der Übersicht über alle nötigen Themenbereiche. Diese beinhaltet eine grobe Einteilung in die Kategorien Koordination, Infrastruktur, Applikation und Ablage.

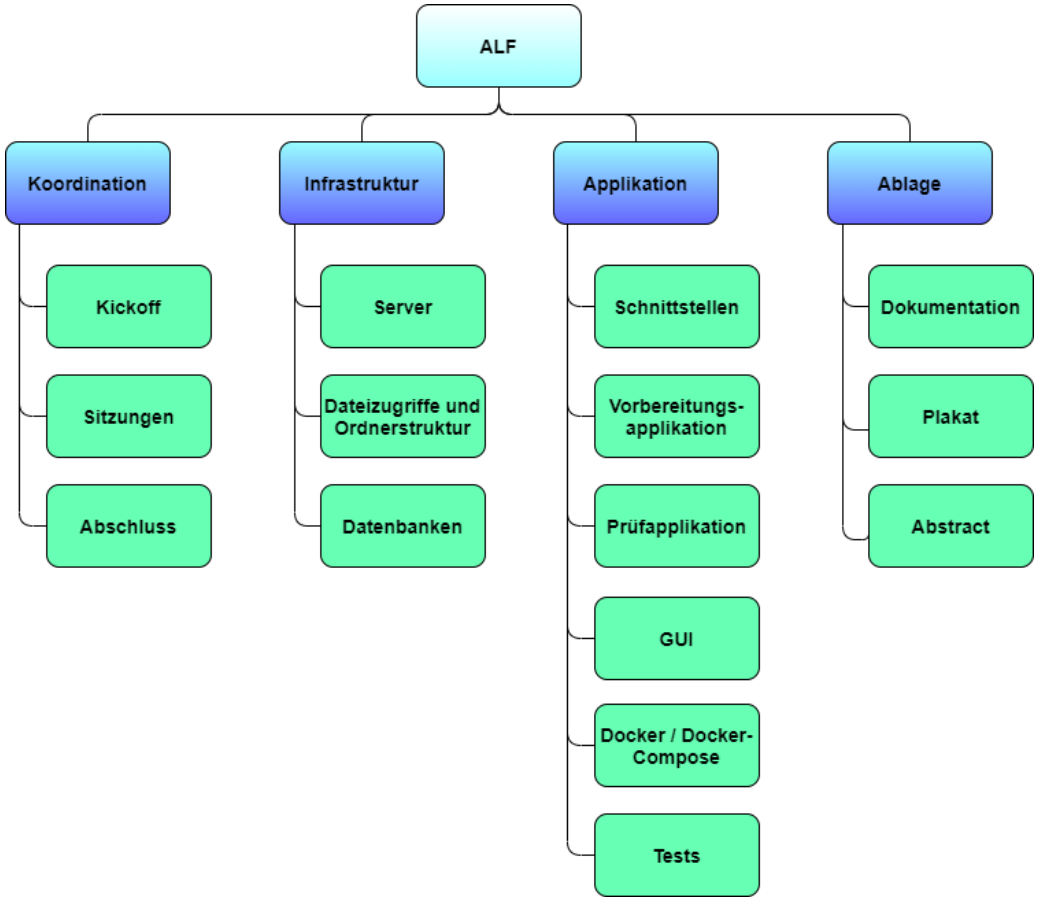


Abbildung 3.1: Projektstruktur

3.2 Architektur-Übersicht

Das System des ALF-Tools soll möglichst alle angedachten Use-Cases abdecken. Die Abbildung 3.2. Containerdiagramm veranschaulicht die Architektur. Das System ist auf zwei einzelne Applikationen aufgeteilt, wobei der Unit-Coordinator alle administrativen Arbeiten ausführt und der Job-Handle die konkreten Abgaben prüfen soll.
Im MVP werden nur CMake-C++-Cute Projekte ausgewertet. Allerdings sind die einzelnen Komponenten so entwickelt, dass es zu einem späteren Zeitpunkt möglich ist, weitere Projekttypen hinzuzufügen.
Das System verfügt im MVP über keinerlei Abhängigkeiten zu anderen Systemen. Eine Anbindung an Moodle könnte zukünftig angedacht werden, welche beim Design und der Umsetzung berücksichtigt werden kann.

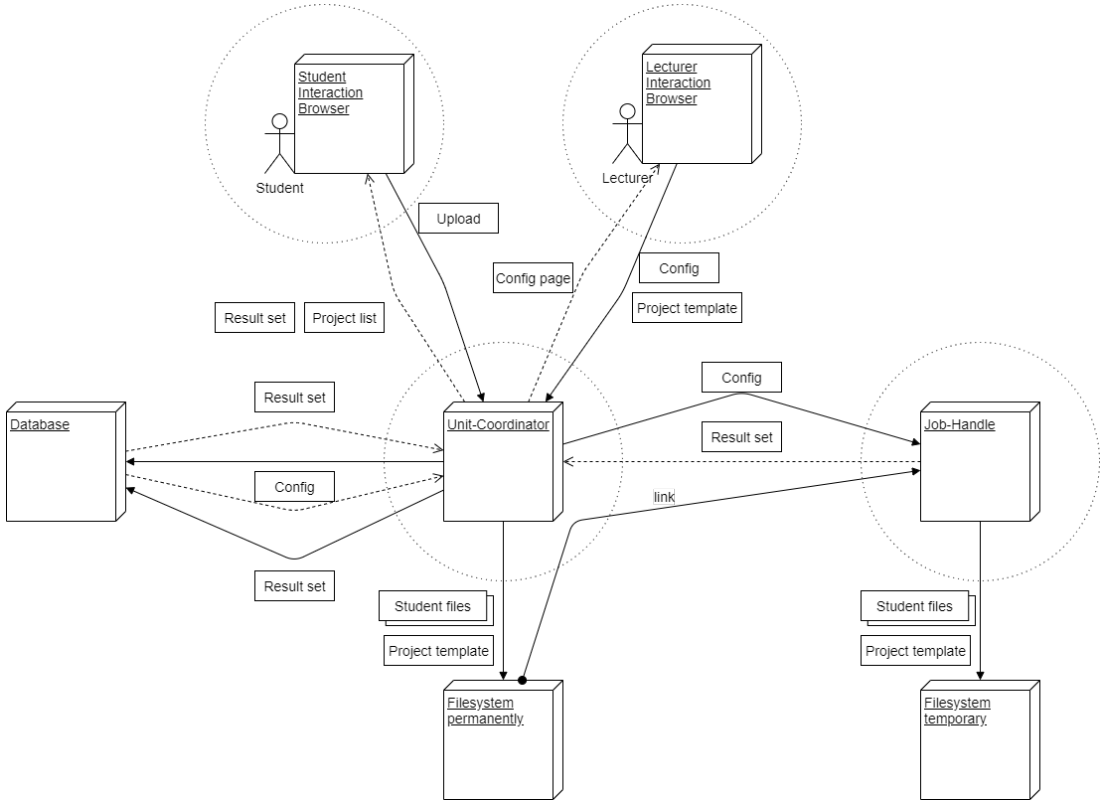


Abbildung 3.2: Containerdiagramm

Der Unit-Coordinator ist für alle Aufgaben im GUI-Bereich, sowie der Speicherung von Daten und Resultate zuständig. Der Unit-Coordinator speichert die Projekte und die Abgaben der Studenten in einem permanenten Dateisystem. Konfigurationen und Resultat-Sets speichert er in einer Datenbank. Bei einer Abgabe eines Studenten übermittelt der Unit-Coordinator einem Job-Handle einen Pfad mit den Dateien und wie diese auszuwerten sind. Nach der Antwort des Job-Handle speichert und visualisiert der Unit-Coordinator die Resultat-Sets. Als Authentifizierung wird das OAuth 2.0 Protokoll über GitLab-OST angedacht. Alternativen dazu wären das Package *django-allauth* oder OAuth.

Der Job-Handle ist für das Auswerten einer Abgabe zuständig. Er bekommt einen Pfad zu den benötigten Dateien, sowie eine Anweisung, wie die Dateien auszuwerten sind. Der Job-Handle führt anschliessend die Unit-Tests aus und generiert, als Antwort für den Unit-Coordinator, ein Resultat-Set.

Der "Student-Interaction-Browser" ist das standardmässige Frontend des Unit-Coordinators, welches bei den Studenten angezeigt wird. Über dieses GUI können die Studenten neue Lösungen hochladen oder Resultate ihrer früheren Abgaben einsehen.

Der "Lecturer-Interaction-Browser" ist ein weiteres Frontend des Unit-Coordinators, welches nur denjenigen Nutzern angezeigt wird, welche die entsprechenden Rechte besitzen. Dozenten können über dieses GUI neue Konfigurationen erstellen / ändern oder neue Projekte hochladen.

3.3 Sequenz-Übersicht

3.3.1 Neue Abgabe

Das Hochladen einer neuen Abgabe ist der Hauptanwendungsfall in diesem Tool, siehe Abbildung 3.3. Sequenzdiagram - Neue Abgabe. Bei diesem Anwendungsfall sind alle Komponenten betroffen. Die Hauptaufgabe hat der Unit-Coordinator, welcher die Abgabe und deren Resultate in die Datenbank speichert. Neben dem Schreiben in die Datenbank erstellt der Unit-Coordinator auch das vom Job-Handle auszuführende Projekt. Nach dem Erstellen des Projekts startet der Unit-Coordinator den Job-Handle mit diesem Projekt.
Der Ablauf für das Testen eines Projekts durch den Dozenten hat den gleichen Ablauf, wie die Abgabe durch den Studenten, mit der Ausnahme, dass nichts in der Datenbank persistiert wird und dem Dozenten auch keine "received page" gesendet wird, sondern direkt die "result page".
Die "pending page" bekommt der Student nur, wenn er eine Abgabe anfragt, welche noch nicht ausgewertet ist. Ansonsten bekommt er die "result page" (siehe Unterabschnitt 3.3.2. Resultate anzeigen).

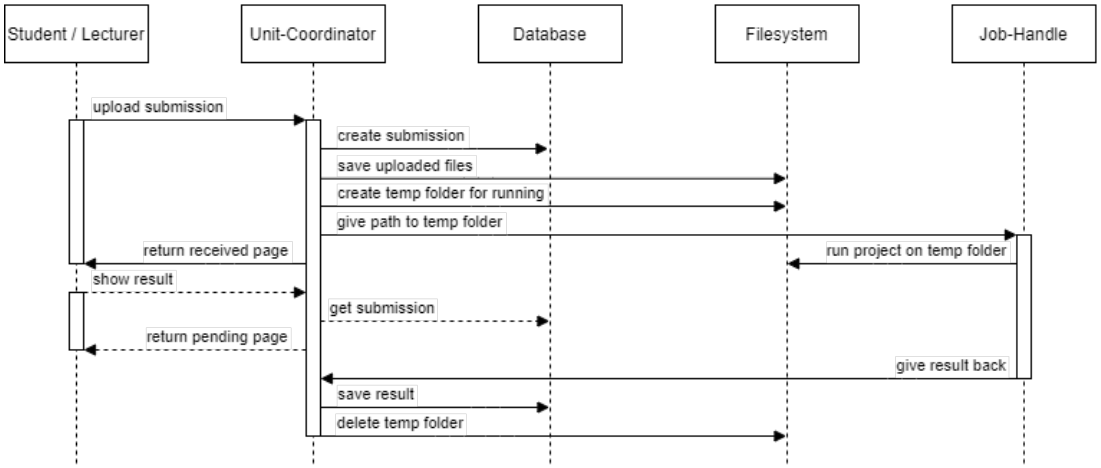


Abbildung 3.3: Sequenzdiagram - Neue Abgabe

3.3.2 Resultate anzeigen

Neben der Abgabe ist das Anzeigen der Resultate einer der wichtigsten Anwendungsfälle. Bei einer Anfrage an den Unit-Coordinator bezüglich der Resultate holt sich dieser die Daten aus der Datenbank und gibt die "result page" zurück. Dieser Vorgang ist in der Abbildung 3.4. Sequenzdiagram - Resultate anzeigen ersichtlich.

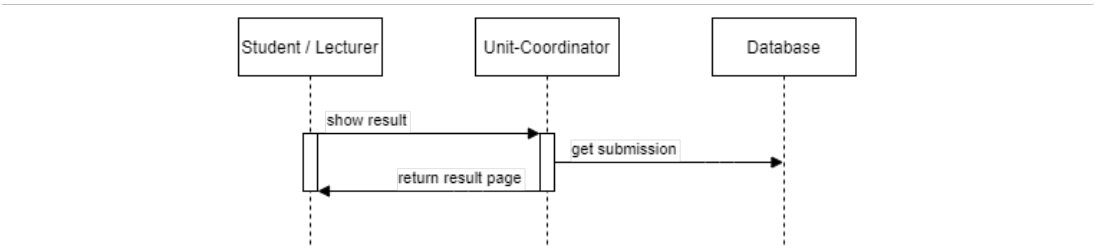


Abbildung 3.4: Sequenzdiagram - Resultate anzeigen

3.3.3 Abgabe herunterladen

Ein Student oder Dozent kann die hochgeladenen Dateien auch wieder herunterladen. Dabei greift der Unit-Coordinator auf das Dateisystem zu und erstellt ein Zip-Archiv, welches er zurückgibt. Das Herunterladen einer Abgabe ist in der Abbildung 3.5. Sequenzdiagram - Abgabe herunterladen aufgezeigt.

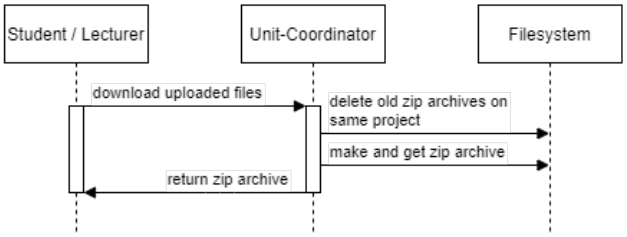


Abbildung 3.5: Sequenzdiagram - Abgabe herunterladen

3.4 GUI Design

Die Studenten dürfen nicht auf die gleichen Funktionen wie die Dozenten Zugriff haben. Um dies zu gewährleisten, könnte man zwei komplett getrennte Oberflächen zur Verfügung stellen. Eine zweite Variante ist die Sicherstellung der Zugriffsrechte mit einer Trennung aller Funktionen.
Im Rahmen dieses Projekts ist die zweite Variante umgesetzt. Dozenten und Studenten sind beides Benutzer und je nach Berechtigung werden ihnen mehr oder weniger Funktionen zur Verfügung gestellt. Um dies umzusetzen, werden ganze Unterseiten ein- beziehungsweise ausgeschaltet.
Die Abbildung 3.6. Wireframe - Neues Projekt erstellen zeigt ein geplantes Wireframe des Formulars zur Erstellung eines neuen Projekts, welches nur für einen Dozenten aufrufbar ist.

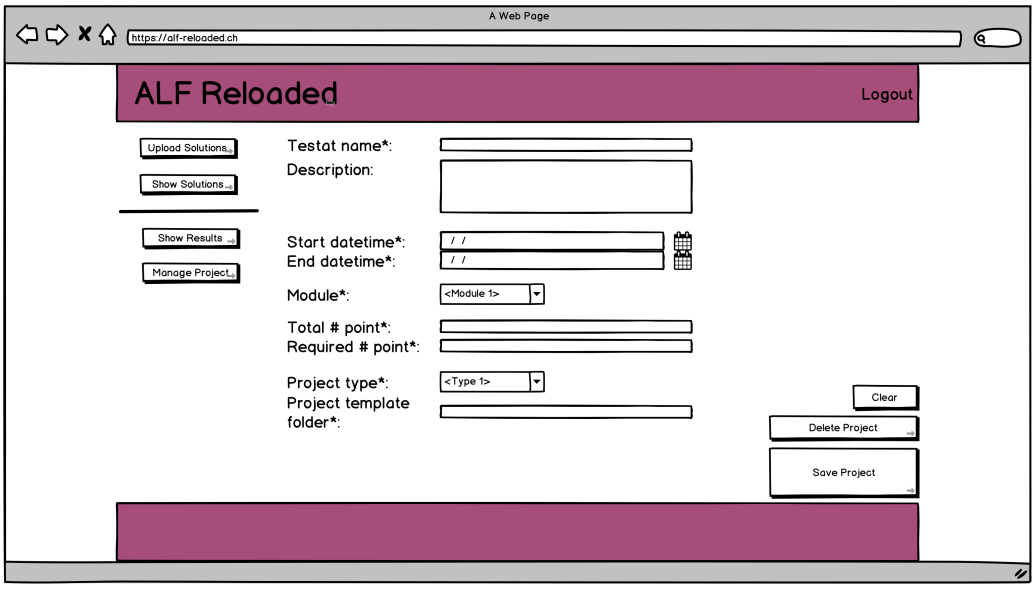


Abbildung 3.6: Wireframe - Neues Projekt erstellen

Unter Abschnitt F.1. Wireframes kann man die Wireframes der Webseite einsehen. Die Wireframes widerspiegeln nur die Grundüberlegungen in der Planung, dementsprechend weicht die finale Lösung davon ab.

3.5 Datenbank Design

3.5.1 Datenbank Wahl

Für die Wahl der Datenbank kommt es in erster Linie darauf an, welche Datenbanken von Django unterstützt werden. Django unterstützt gemäss offizieller Dokumentation fünf Datenbanken [7].

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

Zusätzlich ist es das Ziel, eine Datenbank zu verwenden, mit welcher in der Vergangenheit bereits Erfahrungen gesammelt wurden. Dies ergibt folgende Schnittmenge mit Django: PostgreSQL, MariaDB, MySQL und SQLite. MariaDB und SQLite sind beide dateibasiert, sprich sie speichern die komplette Datenbank innerhalb einer Datei. Dies vereinfacht die Handhabung während der Entwicklung. Da SQLite die Standarddatenbank von Django ist, ist SQLite ebenfalls in diesem Projekt verwendet.

3.5.2 Tabellenaufteilung innerhalb der Datenbank

Der folgende Abschnitt bezieht sich auf die Abbildung 3.7. Entitäten-Beziehungsdiagramm der Datenbank.
Die Datenbanktabelle *projects*, welche für das Speichern der Konfiguration der Projekte verantwortlich ist, stellt die Haupttabelle dar. Sie ist nur von der Tabelle *modules* abhängig. Die Tabelle *modules* ist nur zu Filterzwecken und wegen dem Gedanken an spätere Verwendungen vorhanden. Die beiden Tabellen *projecttypes* und *requiredfiles* dienen der weiteren Konfiguration des Projekts. Jede Abgabe wird in *submissions* gespeichert, die Ergebnisse der einzelnen Unit-Tests hingegen in der Tabelle *results*. Die Studenten und Dozenten werden in einer Tabelle von Django selbst abgespeichert. Dies erlaubt es die Authentifizierung der Benutzer einfacher und sicherer zu handhaben.

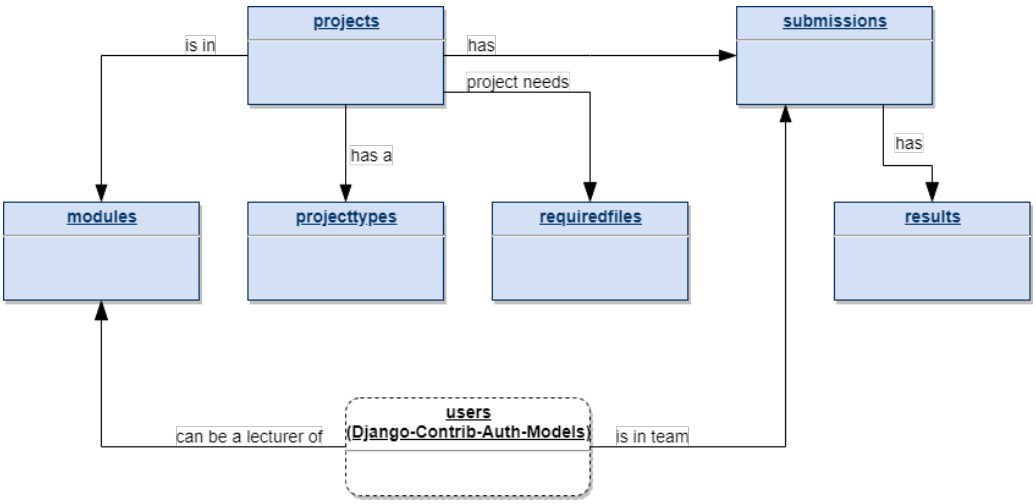


Abbildung 3.7: Entitäten-Beziehungsdiagramm der Datenbank

Kapitel 4

Implementation

Die Implementation ist der Kern jedes Softwareprojekts. In diesem Kapitel wird die Implementation des ALF-Tools behandelt. Die Implementation ist so formuliert, dass von den eigentlichen Technologien kein vertieftes Wissen notwendig ist. Neben den Technologien sind die Entwicklervereinbarungen aufgeführt, an welche sich die Entwickler halten. Anschliessend ist in der Sektion Implementationsaspekte die eigentliche Entwicklung beschrieben. Zum Ende des Kapitels ist aufgeführt, wie das Tool getestet ist.

4.1 Technologien

4.1.1 Unit-Coordinator und Job-Handle

Die beiden Teilprojekte sind in der Sprache Python geschrieben. Während der Job-Handle in Plain-Python geschrieben ist, ist der Unit-Coordinator mit Django realisiert. Da im ehemalige ALF-Tool Django verwendet wird, wird Django auch für die Neuimplementierung verwendet.

4.1.2 Versionierung

Die Versionierung findet über GitLab-OST statt. Der komplette Sourcecode (ALF-Sourcecode) ist versioniert. Die Dokumentation ist in einem anderen Projektarchiv (ALF-Dokumentation) versioniert. Über GitLab-OST laufen zudem die CI-Runner, welche alle relevanten Unit-Tests bei jedem Commit prüfen, die ALF-Pages mit der Test-Coverage des ALF-Sourcecode veröffentlichen, die Docker-Images erstellen und diese auf Docker Hub veröffentlichen. Auf dem Sourcecode-Projekt (ALF-Sourcecode) werden mehrere Branches für die Entwicklung verwendet, welche in Tabelle 4.1. Übersicht der Branches im ALF-Sourcecode aufgeführt sind.

Branch	Beschreibung
master	Nach jedem Meilenstein werden hier alle Branches zusammengefügt und es wird jeweils ein Marker für den Meilenstein gesetzt. Bei jedem Commit auf diesen Branch werden die Docker-Images erstellt und hochgeladen. Zusätzlich wird die Auswertung der Test-Coverage auf die ALF-Pages hochgeladen.
alf*	Auf diesen Branches werden die Packages entwickelt und versioniert.
job_handle und unit_coordinator	Diese beiden Branches sind für die Entwicklung der Hauptkomponenten des ALF-Sourcecodes.

Tabelle 4.1: Übersicht der Branches im ALF-Sourcecode

4.1.3 Dokumentation

Für das Schreiben der ALF-Dokumentation wird LaTeX verwendet. Dies ermöglicht es, dass mehrere Personen parallel an einer Dokumentation arbeiten können. Ein weiterer grosser Vorteil ist, dass die einzelnen Dateien ebenfalls durch GitLab-OST versioniert werden können.

4.1.4 Authentifizierung

Die Authentifizierung der OST-Studenten und OST-Dozenten erfolgt über GitLab-OST. Dabei wird das Package *django-allauth* genutzt, um grundsätzliche Funktionalitäten zu verwenden. Dieses Package verwendet das OAuth 2.0-Protokoll, welches durch Einfachheit überzeugt und mit GitLab-OST direkt interagiert. GitLab-OST wird hauptsächlich verwendet, da dort bereits nur OST-Angehörige registriert sind und somit eine Kontrolle für Studenten und Dozenten entfällt.

4.1.5 Server

Der Server ist nicht direkt gefordert in diesem Projekt. Dieser dient jedoch dazu, die Skalierbarkeit der Docker-Container zu zeigen und zu validieren. Für dieses Projekt wird ein virtueller Server beantragt, welcher selbst eingerichtet und verwaltet wird. Auf dem Server läuft ein Ubuntu-Betriebssystem, auf welchem die Docker-Images laufen. Die Docker-Images werden über Docker-Compose verwaltet und skaliert.

4.2 Entwicklervereinbarungen

Als Grundlage gelten die Style Guides von PEP 8 für Python Code [8].

4.2.1 Parameterchecks

Alle Parameter einer Funktion müssen mit Typen so eindeutig wie möglich typisiert werden. Somit hilft einem die IDE beim Programmieren, da sie die richtigen Vorschläge machen kann. In Funktionen, welche für die Verwendung ausserhalb der Packages gedacht sind, müssen die Parameterranges überprüft werden.

4.2.2 from ... import ...

Alle Imports sollen so spezifisch wie möglich formuliert werden. Eine Ausnahme stellt hierbei pytest dar, da diese zum einen Decorators verwenden und zum anderen nur in den Testdateien verwendet wird.

4.3 Implementationsaspekte

4.3.1 Schnittstellenverträge

Unit-Coordinator ↔ Job-Handle

Bei der Schnittstelle zwischen Unit-Coordinator und Job-Handle fungiert der Job-Handle als Provider. Dies ist in der Abbildung 4.1. Schnittstellenvertrag: Unit-Coordinator und Job-Handle graphisch abgebildet. Bei dieser Schnittstelle handelt es sich um eine Docker-Compose interne. Es ist nicht möglich von ausserhalb auf den Job-Handle zuzugreifen, daher ist die Schnittstelle nicht versioniert.
Die Anfrage besteht aus dem Projekttypen, damit dem Job-Handle klar ist, wie er die Auswertung vorzunehmen hat, sowie einem Pfad zum auszuführenden Projekt. Dieser Ordner muss alle benötigten Dateien für die Ausführung beinhalten.
Als Antwort erhält der Unit-Coordinator einen Error-String, welcher nur etwas beinhaltet, falls beim Auswerten etwas nicht funktioniert hat. In der Antwort ist eine Liste der Resultate enthalten. Jedes Resultat besteht aus einem Namen (der Name des Unit-Tests), einem Resultat (falls der Unit-Test fehlgeschlagen ist) und einem binären Wert (zeigt ob der Test erfolgreich durchgelaufen ist).

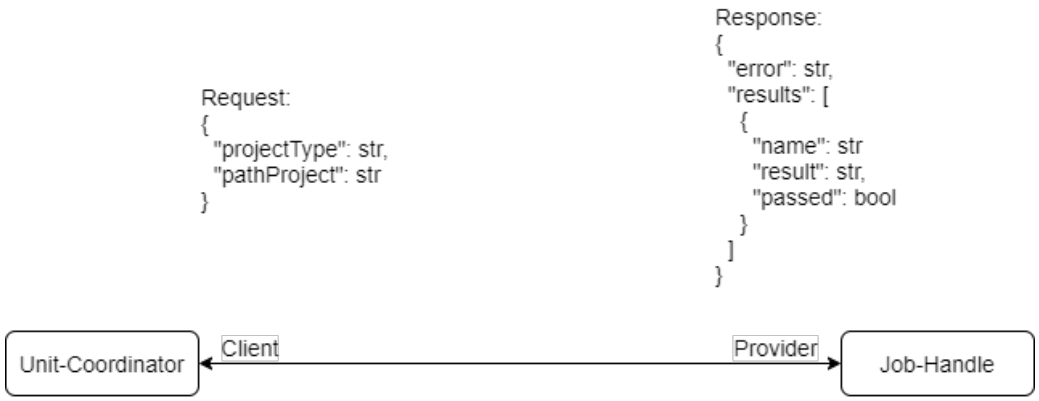


Abbildung 4.1: Schnittstellenvertrag: Unit-Coordinator und Job-Handle

4.3.2 Projektübersicht

In der Abbildung 4.2. Komponentendiagramm des Projekts werden alle Packages und ihre Abhängigkeiten gezeigt. Dabei ist zu erkennen, dass jedes implementierte Package in beiden Teilprojekten verwendet wird. In den folgenden Unterabschnitten werden die beiden Teilprojekte und verwendeten Packages genauer betrachtet. Die Implementation wird nur oberflächlich angesprochen, da ein tieferer Einblick bereits sehr nahe am ALF-Sourcecode wäre. Genauere Implementationsaspekte sind dem ALF-Sourcecode zu entnehmen.

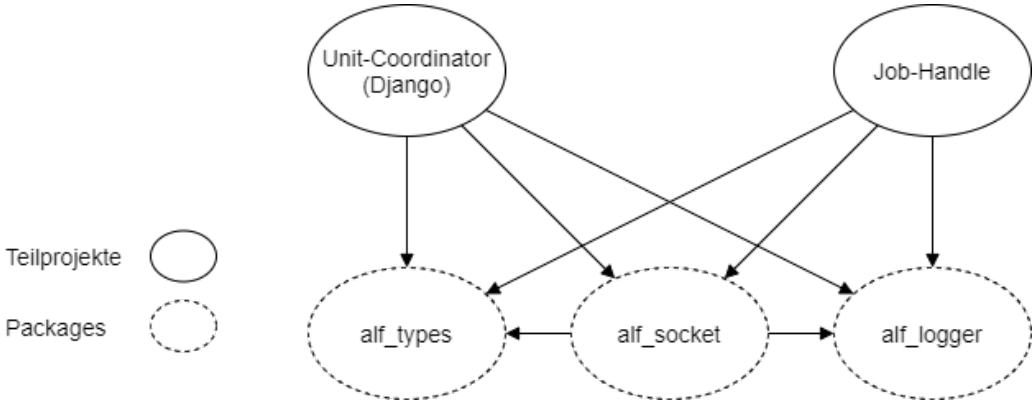


Abbildung 4.2: Komponentendiagramm des Projekts

4.3.3 Unit-Coordinator

Die Abbildung 4.3. Modulübersicht Unit-Coordinator zeigt die Module und ihre Abhängigkeiten im Unit-Coordinator. Im folgenden wird auf die Module eingegangen.

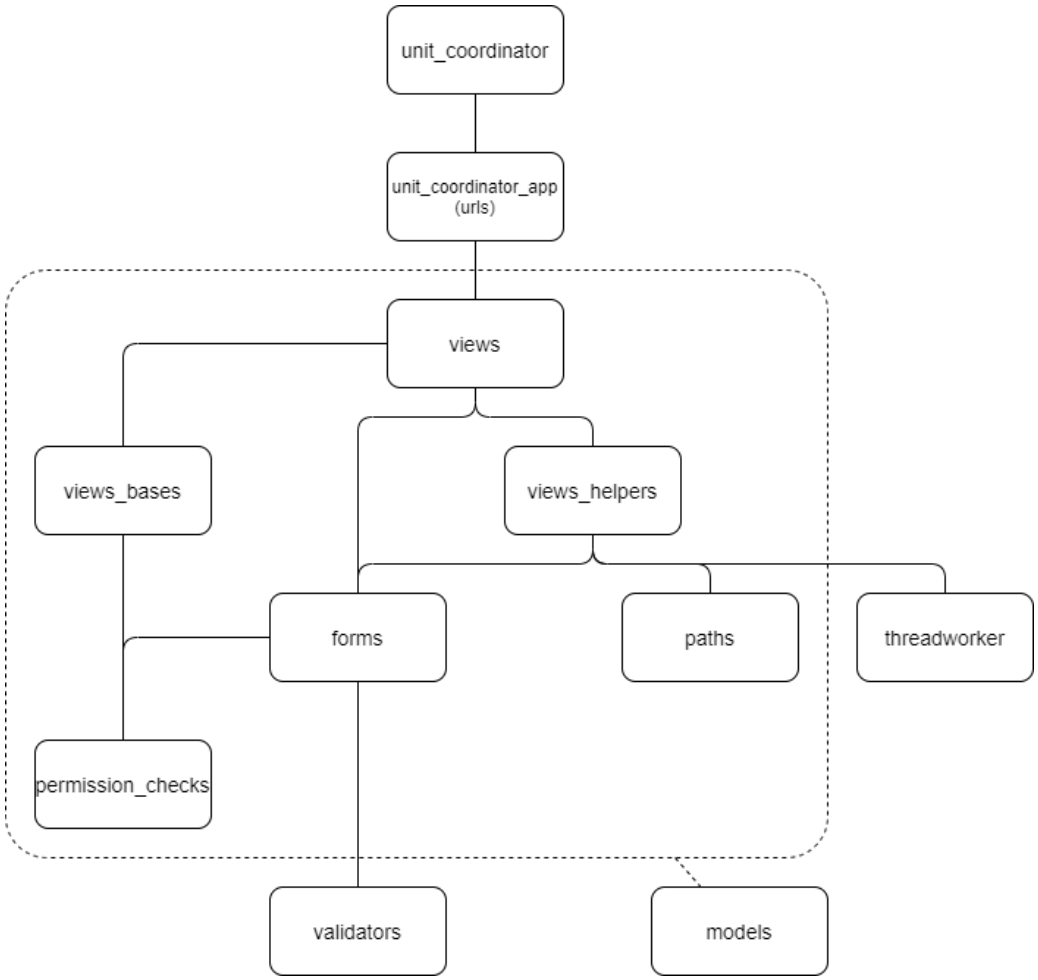


Abbildung 4.3: Modulübersicht Unit-Coordinator

unit_coordinator

Dieses Package stellt den Wurzel-Knoten des Django-Projekts dar. Dieser sorgt für die globalen Einstellungen des Django-Projekts in der *settings.py* und *urls.py* Datei. Erwähnenswert ist ebenfalls die *social_accounts_adapters.py*, welche für das Anmelden über GitLab-OST verwendet wird. Diese Verknüpft das *SocialLogin* mit einem Benutzer und leitet die Anfrage auf die Index-Seite weiter.

unit_coordinator_app

Dieses Package stellt den Wurzel-Knoten für die Implementation dieses Projekts dar. In dieser App sind Ansichten, Formulare, Validatoren und Modelle implementiert.

views

Dieses Modul stellt ein Standard-Django-Modul dar und fasst alle nötigen Klassen und Funktionen für die *urls* zusammen. *views* verwendet *views_bases* für die Abstraktion der Basisklassen. Das *views_helpers* Modul wird für die Abstraktion von unterstützenden Funktionen verwendet. Hingegen das *forms* Modul wird für die Verknüpfung der Formulare mit den Ansichten verwendet.

views_bases

Dieses Modul beinhaltet die Basisklassen für alle Ansichten, welche in *views* erstellt werden. Diese Basisklassen werden nicht instanziiert, da sie vorwiegend zur Minderung von Codezeilen und Wiederverwendung von Funktionen dienen. Dieses Modul verwendet das Modul *permission_checks* um Zugriffe auf Ansichten zu prüfen und zu managen.

views_helpers

Dieses Modul beinhaltet Hilfsfunktionen für Klassen oder Funktionen in *views*. Hilfsfunktionen haben meist nur eine Verwendung und dienen dazu, dass Klassen und Funktionen in *views* kürzer und besser lesbar werden.

paths

Dieses Modul fasst Funktionen zusammen, welche Pfad-Objekte generieren. Die Funktionen sind hierarchisch rekursiv aufgebaut, wobei die Pfade der Ordnerstruktur auf dem Dateisystem entsprechen (siehe Abschnitt 4.3.6. Dateistruktur).

thread_worker

Dieses Modul stellt die Funktionalität zur Verfügung, um über einen Thread mit dem Job-Handle zu kommunizieren. Die Antwort wird über eine Callback-Funktion verarbeitet.

permission_checks

Dieses Modul beinhaltet Funktionen, um Berechtigungen zu prüfen. Es enthält Funktionen zur Prüfung ob ein Benutzer ein Dozent ist, auf ein Projekt oder Abgabe zugreifen darf, sowie die Überprüfung eines Projektdatums.

forms

Dieses Modul stellt ein Standard-Django-Modul dar und fasst alle nötigen Formulare und ihre Verarbeitung zusammen. Es verwendet das Modul *permission_checks*, um bei Formularen nur die nötigen Inhalte anzuzeigen. Das Modul *validators* wird verwendet, um die Eingaben bei Formularen zu überprüfen.

validators

Dieses Modul stellt ein Standard-Django-Modul dar und beinhaltet Klassen, welche Eingaben von Formularen prüfen. Es beinhaltet zurzeit nur eine Klasse (*RequiredFileValidator*). Sie wird für die hochgeladenen Dateien bei Abgaben verwendet, um die Dateinamen inklusive Erweiterungen auf ihre Richtigkeit zu überprüfen.

models

Dieses Modul stellt ein Standard-Django-Modul dar und fasst alle Modelle zusammen. Die Modell-Klassen werden mit Django-ORM ebenfalls für die Erstellung des Datenbankschemas verwendet und bieten eine automatisch generierte Datenbank-API auf den Modellen an. Dieses Modul wird von praktisch allen Modulen verwendet (siehe Abbildung 4.3. Modulübersicht Unit-Coordinator),

4.3.4 Job-Handle

Der Job-Handle ist der Teil des ALF-Tools, welcher für die Ausführung und Auswertung der hochgeladenen Lösungen zuständig ist. Die Abbildung 4.4. Modulübersicht Job-Handle ist eine Übersicht der im Nachfolgenden erläuterten Module.

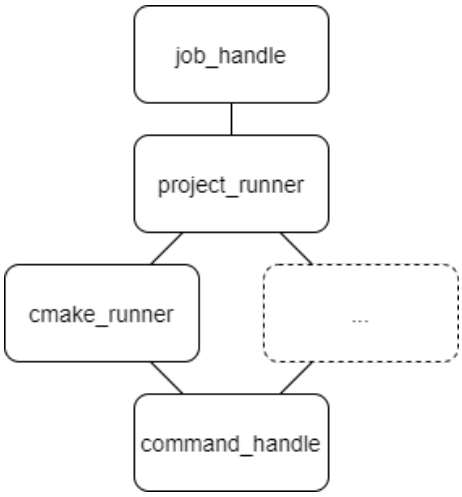


Abbildung 4.4: Modulübersicht Job-Handle

job_handle

Dieses Modul dient hauptsächlich dem Koordinieren und Aufrufen der einzelnen Funktionen des Job-Handles. Zum einen startet es den Websocket-Server über das *alf_socket*-Package. Zum anderen beinhaltet dieses Modul die Handle-Funktion, welche bei einer Anfrage aufgerufen wird um diese zu bearbeiten. Die Handle-Funktion gibt die Anfrage dem *project_runner* Modul weiter.

project_runner

Dieses Modul macht nichts anderes, als für den geforderten Projekttypen das richtige Modul, beziehungsweise Funktion, aufzurufen. Diese Funktion ist für die Auswertung der Abgabe zuständig. Für diese Studienarbeit ist dieses Modul nicht zwangsweise notwendig. Allerdings ermöglicht es eine einfache Einbindung weiterer Runner für andere Projekttypen.

cmake_cpp_cute_runner

Die beiden Aufgaben dieses Moduls sind das Auswerten des Projekts und das Parsen in die Antwortform (siehe Abschnitt 4.3.1. Unit-Coordinator ↔ Job-Handle).
Für ein CMake-Projekt mit C++ und Cute wäre dies, das Projekt zu erstellen und anschliessend die Tests auszuführen. Das Resultat der Tests ist ein riesiger Text, welcher im Anschluss in die Antwortform geparst wird.

command_handle

Das *command_handle*-Modul stellt eine Funktion zur Verfügung mit welcher man Shell-Befehle ausführen kann.

4.3.5 Allgemeine Packages

Alle folgenden Packages sind mit Python-Wheels erstellt. Die erstellten Packages sind nicht veröffentlicht und befinden sich nur auf dem Projektarchiv des Sourcecodes (ALF-Sourcecode).

alf_logger

Dieses Package stellt nur eine Funktion zur Verfügung. Mit dieser Funktion kann ein Logger erstellt werden, ohne dass man sich um seine Konfiguration zu kümmern braucht. Dieser Logger schreibt die Log-Datei standardmässig an einen Pfad, welcher in den Umgebungsvariablen definiert ist.

alf_types

Dieses Package stellt verschiedene Typen für das Projekt zur Verfügung. Die Typisierung erfolgt mit der Hilfe des Typing-Packages, welches direkt in Python integriert ist. Durch die definierten Typen im *alf_socket*-Package werden die Entwickler unterstützt. Diese helfen den Schnittstellen-Vertrag einzuhalten.

alf_socket

Das *alf_socket*-Package stellt dem Projekt die Funktionalitäten des Websockets zur Verfügung. Zusätzlich übernimmt dieses Package die korrekte Konvertierung zwischen dem Python-Dictionary und dem zu übermittelnden JSON.
Das Package besteht primär aus zwei Funktionen:
Die eine Funktion ist für den API-Provider (Job-Handle). Diese läuft nach dem Aufrufen asynchron(endlos). Sie nimmt Anfragen auf und ruft mit den Anfragen eine vordefinierte Handle-Funktion auf. Die Antwort wird anschliessend zurückgeschickt.
Die andere Funktion ist für den API-Client (Unit-Coordinator). Sie sendet einem API-Provider eine Anfrage. Nach dem Senden wird asynchron auf die Antwort des API-Providers gewartet. Diese Antwort wird anschliessend zurückgegeben.

4.3.6 Server

Um eine möglichst genaue Implementierung vorzunehmen, wird ein Ubuntu-Server vom OST – Virtual Server Kiosk verwendet. Dieser Server hat einen Lebenszyklus bis zum 31.01.2022.

Konfiguration

Der Server ist auf https konfiguriert. Alle unverschlüsselten Anfragen werden auf https umgeleitet. Dies ist mit dem Einsatz von Traefik realisiert.
Um eine skalierbare Umsetzung zu demonstrieren, sind beide Container im Prod-Modus skaliert. Im Dev-Modus gibt es keine Skalierung, beide Container laufen mit einer Instanz. Die Skalierung ist in der Tabelle 4.2. Skalierung der Container ersichtlich. Dabei ist der Server so konfiguriert, dass die jeweiligen Container im Round-Robin Verfahren aufgerufen werden.

	Anzahl Instanzen	
Name	**Dev-Modus**	**Prod-Modus**
Unit-Coordinator:	1	2
Job-Handle:	1	5

Tabelle 4.2: Skalierung der Container

Dateistruktur

Die beiden Teilprojekte (Unit-Coordinator und Job-Handle) arbeiten beide auf dem gleichen Dateisystem, siehe Abbildung 4.5. Ordnerstruktur: Root. Der Unit-Coordinator sieht und hat dabei Lese-/ Schreibrechte für die vier Ordner; */alf/alf/*, */alf/db/*, */alf/temp/* und */alf/logs/*. Der Job-Handle sieht hingegen nur den */alf/logs/* und */alf/temp/* Ordner. Auf dem */alf/temp/* hat er nur Lesezugriff. Dies verhindert, dass bei der Ausführung Dateien von der Datenbank oder den Projekten ausgelesen oder verändert werden können. Auf dem */alf/logs/* Ordner haben sowohl der Unit-Coordinator wie auch der Job-Handle vollen Lese- und Schreibzugriff, damit diese dort ihre Log-Dateien ablegen können.

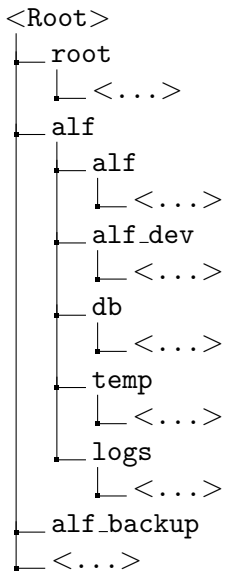


Abbildung 4.5: Ordnerstruktur: Root

/alf/alf_dev/ und /alf_backup/

Bei jedem Starten des Dev-Modus wird der */alf/alf_dev/* Ordner durch den */alf_backup/* Ordner ersetzt. So wird das Dateisystem bei jedem Neustart des Dev-Modus aufgeräumt.
Die Datei-Struktur des */alf/alf_dev/* ist gleich der des */alf/alf/* (siehe Abbildung 4.7. Ordnerstruktur: /alf/alf/). Der Unterschied liegt darin, dass im */alf/alf_dev/* bereits Projekttypen, Projekte und Abgaben vordefiniert sind (für genauere Informationen siehe Abschnitt B.2. Server)

/alf/db/
Dies ist der Speicherort aller Dateien der Datenbank. Die Ordnerstruktur ist schlicht gehalten, da mit SQLite nur jeweils eine Datei erstellt und genutzt wird. Diese Struktur ist in Abbildung 4.6. Ordnerstruktur: /alf/db/ dargestellt.

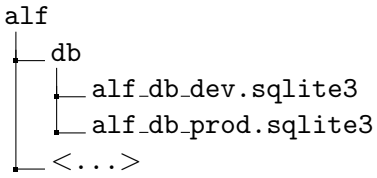


Abbildung 4.6: Ordnerstruktur: /alf/db/

/alf/alf/
In diesem Ordner werden alle Dateien der Projekte abgelegt. Neben den Dateien, welche durch die Studenten oder Dozenten hochgeladen werden, befinden sich hier auch die Dateien, die von den Projekten benötigt werden. In der nachfolgenden Aufzählung ist die Struktur eines Projekttyps und eines Projekts beispielhaft erläutert. Die Abbildung 4.7. Ordnerstruktur: /alf/alf/ zeigt eine entsprechende Ordnerstruktur auf.

- /alf/alf/
  Für jeden Projekttypen liegt an diesem Pfad ein Ordner. Da in dieser Studienarbeit nur der Projekttyp "CMake_Cpp_Cute" Priorität hat, existiert hier nur ein Ordner.
- .../<*Projecttype-ID*>_<*Projecttype-Name*>/projects/
  Für jedes Projekt des Projekttypen existiert ein Ordner.
- .../<*Projecttype-ID*>_<*Projecttype-Name*>/template/
  Spezifische Dateien für den Projekttypen. Für "CMake_Cpp_Cute" liegt hier zum Beispiel der Ordner für Cute.
- .../projects/<*Project-ID*>_<*Project-Name*>/archives/
  In diesem Ordner werden die Zip-Archive für den Download erstellt.
- .../projects/<*Project-ID*>_<*Project-Name*>/submissions/
  Für jede Abgabe des Projekts liegt in diesem Pfad ein Ordner.
- .../projects/<*Project-ID*>_<*Project-Name*>/template/
  An diesem Pfad liegen spezifische Dateien für das Projekt. Zum Beispiel zusätzliche Dateien für das Projekt, aber vor allem die Unit-Tests.

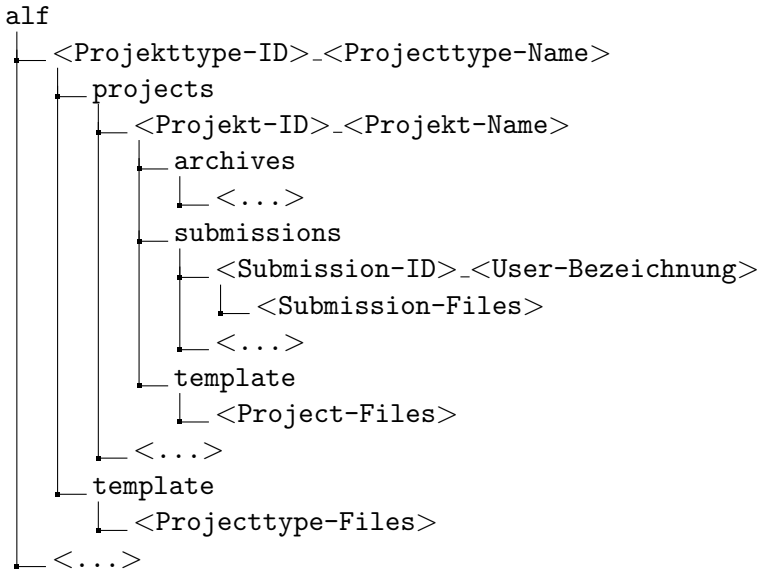


Abbildung 4.7: Ordnerstruktur: /alf/alf/

/alf/temp/

Der */alf/temp/* Ordner wird für die Ausführung der Projekte verwendet. Eine beispielhafte Ordnerstruktur ist in der Abbildung 4.8. Ordnerstruktur: /alf/temp/ abgebildet. Für jede Ausführung wird vom Unit-Coordinator ein neuer Ordner erstellt, in welchen alle relevanten Dateien von */alf/alf/* kopiert werden. Die relevanten Pfade sind in der Tabelle 4.3. Pfade für ein vollständiges Projekt aufgelistet (für die genauen Pfade siehe Abbildung 4.7. Ordnerstruktur: /alf/alf/). Der neu erstellte Ordner in */alf/temp/* wird nach der Ausführung des Job-Handles wieder gelöscht. Dementsprechend ist der */alf/temp/* im Ruhezustand immer leer.

Die benötigten Dateien setzen sich aus den folgenden Pfaden zusammen:

Inhalt	**Pfad**
Template des Projekttypen:	/alf/alf/<*Projekttype-ID*>_<*Projecttype-Name*>/template/
Template des Projekts:	/alf/alf/.../projects/<*Projekt-ID*>_<*Projekt-Name*>/template/
Abgaben der Studenten:	/alf/alf/.../submissions/<*Submission-ID*>_<*User-Bezeichnung*>/

Tabelle 4.3: Pfade für ein vollständiges Projekt

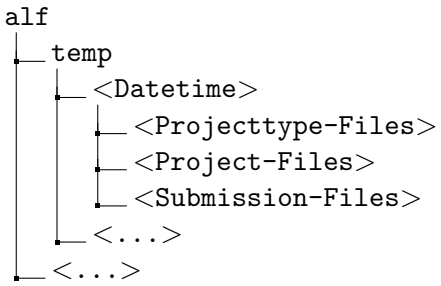


Abbildung 4.8: Ordnerstruktur: /alf/temp/

/alf/logs/

In diesem Ordner werden die Log-Dateien des gesamten ALF-Tools abgelegt. Es wird pro Container eine Datei pro Woche angelegt. Dies geschieht so lange bis es fünf Log-Dateien von einem Container gibt. Anschliessend wird jeweils die älteste der fünf Dateien gelöscht. Jede Log-Datei hat folgende Namenskonversion: *YYYY-MM-DD_<HOSTNAME>.log*, wobei der *HOSTNAME* der Name des ausführenden Containers ist.

Nebenläufigkeit

Da die einzelnen Teilprojekte schon in Docker-Containern umgesetzt sind, können diese bereits nebenläufig und unabhängig skaliert werden. Dieses Limit ist durch den verwendeten Server gegeben. Der Unit-Coordinator, als Django-Projekt, bietet bereits eine gewisse Nebenläufigkeit der eingehenden Anfragen. Da die eigentliche Funktion des Projekts auf der Einreichung von Abgaben besteht, werden prinzipiell nur Daten auf dem Dateisystem angelegt oder die Datenbank mit neuen Einträgen gefüllt. Der Job-Handle kann nur eine Anfrage gleichzeitig verarbeiten. Daher wird empfohlen, mehrere Instanzen des Job-Handles zu verwenden. Der Job-Handle hat jedoch einen Nachrichten-Puffer von 32 Anfragen.

Nachfolgend sind die bestehenden Risiken und Einschränkungen der Nebenläufigkeit aufgeführt:

- **Unit-Coordinator - Dateisystem**
  Da Abgaben von Studenten wie auch von Dozenten heruntergeladen werden können, stellt dies ein Risiko in der Nebenläufigkeit dar. Zurzeit wird der *archives*-Ordner stets vor einem neuen Download aufgeräumt, um Datenmüll zu minimieren. Fragen also mehrere Personen im selben Projekt gleichzeitig einen Download an, kann es zu Verlusten von Daten im *archives*-Ordner kommen, jedoch nie im */alf/alf/* selbst. Stellt sich dieses Problem als kritischer heraus, könnte man das Löschen der *archives*-Ordner nur zu bestimmten Zeiten durchführen. Mit dieser Änderung würden Kollisionen nur noch sehr selten auftreten.

- **Unit-Coordinator - Datenbank**
  Neben dem Dateisystem werden ebenfalls Einträge in der Datenbank verändert. Die Datenbankabfragen werden über das Django-ORM verwaltet und auch synchronisiert. Somit werden die meisten Kollisionen bereits dort abgefangen.
  Die meisten Änderungen finden statt, wenn Dozenten ein Projekt umbenennen oder wenn Resultate vom Job-Handle eintreffen. Die Änderung der Abgabe bei eintreffenden Resultaten stellt kein Problem dar, da über das Frontend nur neue Abgaben eingetragen werden können und jeweils nur ein Thread auf Resultate des Job-Handles wartet. Beim Anzeigen von Abgaben wird die Datenbank nur lesend benutzt.
  Benennen Dozenten ein Projekt jedoch um, werden gleichzeitig die Pfade zum Projekt und den Abgaben geändert. Dies stellt ein grösseres Problem der Nebenläufigkeit dar, da gleichzeitig Studenten Abgaben hochladen könnten.
  Daher empfiehlt es sich Projekte bereits beim Erstellen richtig zu benennen. Eine weiter Möglichkeit ist es, vor einer Umbenennung, das Enddatum in die Vergangenheit oder das Startdatum in die Zukunft zu setzen. Alle weiteren Angaben haben keinen Einfluss auf die Ordnerstruktur und können auch später ohne Probleme noch geändert werden.

4.3.7 Datenbank

Es ist anzumerken, dass in Django-Projekten Modelle erstellt werden, um die Struktur zu definieren. Ein vereinfachtes Datenbankschema ist in der Abbildung 4.9. Vereinfachtes Datenbankschema dargestellt, wobei die Datenbanktabellen gleichzeitig die Modelle darstellen. Im weiteren werden die verwendeten Typen in der Tabelle 4.4. Datenbanktypen erläutert. Mittels dem Django-ORM wird das Datenbankschema aus den Modellen jeweils generiert oder migriert. Das Datenbankschema ist also nicht direkt in der Datenbank eingerichtet, wodurch die Datenbank mit wenigen Konfigurationsänderungen gewechselt werden kann. Unter Anhang D. Datenbankschema ist das komplette Datenbankschema mit allen Attributen zu finden.

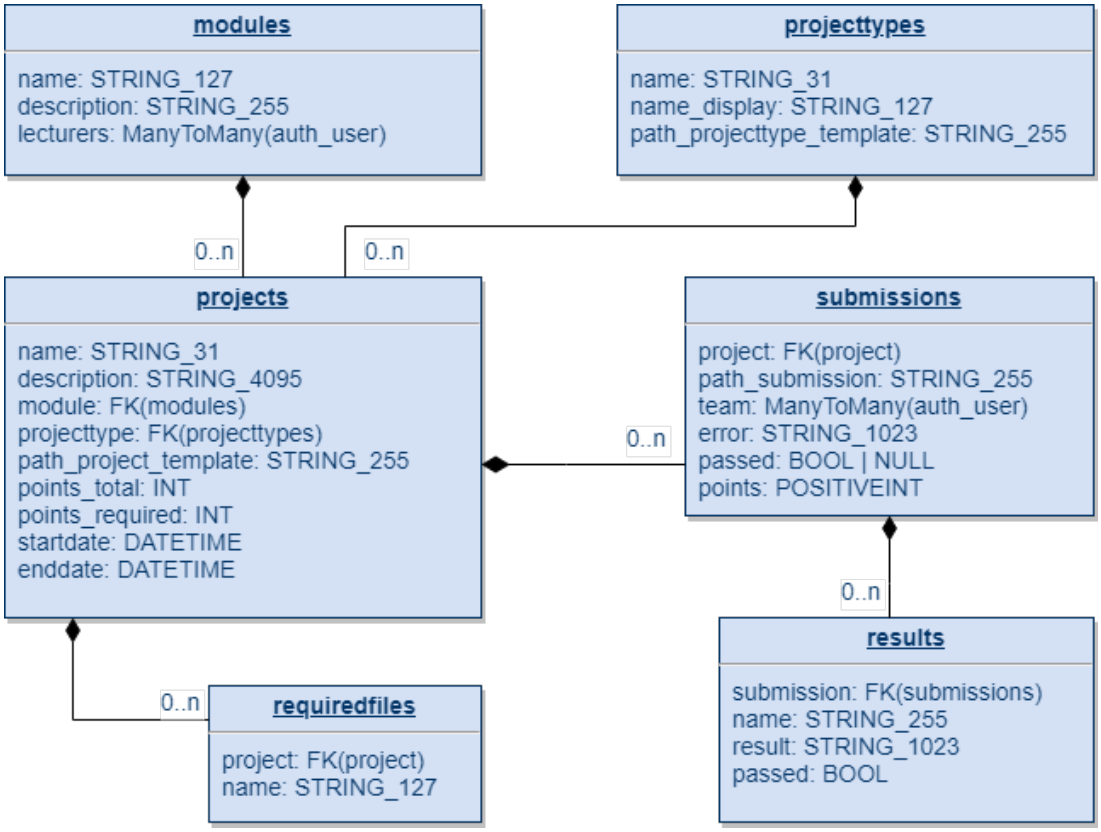


Abbildung 4.9: Vereinfachtes Datenbankschema

In der folgenden Tabelle werden die verwendeten Typen erläutert.

Bezeichnungen	**Erläuterung**
BIGINT_PK	Ein Integer-Feld, welches von Django selbst angelegt und als primären Schlüssel verwendet wird.
NULL	Dieses Feld darf alternativ nicht ausgefüllt werden und wird dann mit dem None-Objekt befüllt.
BOOL	Ein Feld für einen binären Wert.
INT	Ein Ganzzahl-Feld mit normaler Grösse.
STRING_XXXX	Eine Text-Repräsentation, welche XXXX Zeichen aufnehmen kann.
DATETIME	Ein Feld mit Datum- und Zeitangaben. Aufgrund der gesetzten Einstellung von *USE_TZ* wird ein Datum-Zeit-Objekt verwendet, welches die Zeitzonen berücksichtigt.
FK(...)	Dieses Feld stellt ein Fremdschlüssel zu einem anderen Modell (einer anderen Tabelle) dar. Auf dem referenzierten Modell kann diese Beziehung ebenfalls über das Attribut .MODELNAME_set abgefragt werden, sofern dies nicht mit *related_name* überschrieben wurde. Siehe auch Reverse(..).
ManyToMany(...)	Dies stellt eine ManyToMany-Beziehung zu einem anderen Modell (einer anderen Tabelle) dar. Auf dem referenzierten Modell kann diese Beziehung ebenfalls über das Attribut .MODELNAME_set abgefragt werden, sofern dies nicht mit *related_name* überschrieben wurde. Siehe auch Reverse(..).
Reverse(...)	Dies ist kein direktes Feld auf dem Modell, sondern stellt eine Gegenseite zu einem referenzierten Modell dar. Diese Felder werden durch ein FK(...) oder ManyToMany(...) auf einem anderen Modell gesetzt.

Tabelle 4.4: Datenbanktypen

4.4 Testen des ALF-Sourcecode

Das Ziel bei den Unit-Tests ist es eine Test-Coverage von über 90% zu erreichen. Die erreichte Test-Coverage ist unter Abschnitt 6.4. Test-Coverage zu finden. Dabei ist zu erwähnen, dass eine hohe Test-Coverage nichts darüber aussagt, ob die Tests sinnvoll sind und die nötigen Fälle abdecken. Um dem entgegenzuwirken, wurden die Tests jeweils durch einen zweiten Entwickler geprüft und wenn nötig angepasst.
Zusätzlich zu den Unit-Tests sind die Module *project_runner* und *cmake_cpp_cute_runner* mit diversen Integration-Tests geprüft.
Durch die knappe Zeit während der Studienarbeit wurden nur Unit-Tests und Integrations-Tests automatisiert durchgeführt, sowie unstrukturierte System-Tests durch die Entwickler.
Die Integrations-Tests decken folgende Fälle ab:

Projekttyp	**Test-Case**	**Beschreibung**
CMake_Cpp_Cute	fully_passed	Alle Unit-Tests müssen erfolgreich durchlaufen.
CMake_Cpp_Cute	not_fully_passed	Einige Unit-Tests müssen anschlagen.
CMake_Cpp_Cute	failed_compilation	Das Erstellen des Projekts muss fehlschlagen.
CMake_Cpp_Cute	too_long_execution	Bei der Ausführung muss ein Time-Out ausgelöst werden.
CMake_Cpp_Cute	wrong_testat	Das Erstellen des Projekts muss fehlschlagen.
CMake_Cpp_Cute	invalid_memory_write	Beim Ausführen muss ein Fehler festgestellt werden.
CMake_Cpp_Cute	infinity_memory_leak	Bei der Ausführung muss ein Time-Out ausgelöst werden oder ein Fehler festgestellt werden.
CMake_Cpp_Cute	missing_allocation	Beim Ausführen muss ein Fehler festgestellt werden.
CMake_Cpp_Cute	stack_overflow	Beim Ausführen muss ein Fehler festgestellt werden.

Tabelle 4.5: Integrations-Tests für die Module *project_runner* und *cmake_cpp_cute_runner*

Kapitel 5

Projektmanagement

Durch das Projektmanagement wird versucht einen reibungslosen Ablauf des Projekts sicherzustellen. Das Projektmanagement umfasst nicht nur das Vorgehen unterschiedlicher Prozesse, sondern auch die langfristige Planung durch Meilensteine und die Projektorganisation.

5.1 Vorgehen

5.1.1 Prozess

Da das Entwicklerteam nur aus zwei Personen besteht, wird auf eine komplett durchstrukturierte Sprint-Planung à la Scrum verzichtet. Es wird zwar wöchentlich entschieden, welche Tasks in der nächsten Woche Priorität haben. Diese werden jedoch auf dem Agile-Board nicht innerhalb eines Sprints gelistet. Daher existiert nur ein grosses Agile-Board, auf welchem alle Tasks zu finden sind, welches als Custom-Kanbanboard eingerichtet ist. Das Verwenden des Boards wird unter Unterabschnitt 5.1.3. Workflow definiert.
Die Zeiterfassung ist in vier verschiedene Kategorien aufgeteilt. Diese ermöglichen es einen Überblick über die verschiedenen Tätigkeiten zu erhalten. Die Kategorien dienen nicht nur der Übersicht, sondern ermöglichen es auch eine detailliertere Zeitauswertung vorzunehmen (siehe Abschnitt 7.3. Zeiterfassung). Die Auswertungen ermöglichen es für zukünftige Projekte Schlüsse zu ziehen.
Folgende Kategorien sind vorhanden:

- Dokumentation
- Meeting
- Recherche
- Entwicklung

5.1.2 Review

Bevor der Task geschlossen wird, muss von einem anderen beteiligten Entwickler ein Review durchgeführt werden. Dabei wird vor allem auf folgende Punkte geachtet:

- Das gewählte Code-Design stimmt zum restlichen Code der Applikation.
- Die Style-Guides der jeweiligen Sprache werden eingehalten.
- Der Code funktioniert so wie er soll, aus der Sicht des Entwicklers und Nutzers.
- Der Code kann nicht sinnvoll vereinfacht werden und ist gut lesbar.
- Der geschriebene Code ist mit Unit-Tests abgedeckt.
- Die vorhandenen Kommentare bringen einen Mehrwert.
- Die wichtigsten Funktionen sind in der ALF-Dokumentation erläutert.
- Die DoD ist erfüllt.

5.1.3 Workflow

Das Kanban-Board teilt sich in folgende fünf Spalten auf:

- Backlog
- Offen
- In Bearbeitung
- Erledigt
- Verifiziert

Alle Tasks werden in der Spalte *Backlog* erstellt. Tasks, welche nächste Woche bearbeitet werden sollen, kommen in die Spalte *Offen*. Sobald an diesen gearbeitet wird, werden sie nach *In Bearbeitung* verschoben, wo sie bis zum DoD verbleiben. Ist bei einem Task das DoD erreicht, wird er zur Verifikation in die Spalte *Erledigt* verschoben. Nun ist es die Aufgabe eines anderen Entwicklers gemäss Unterabschnitt 5.1.2. Review diesen Task zu verifizieren.
Bei den Kategorien *Dokumentation* und *Meeting* wird anstatt eines Reviews nur die DoD kontrolliert. Die Tasks der Kategorien *Recherche* und *Organisation* können selbständig von *In Bearbeitung* nach *Verifiziert* verschoben werden.

5.1.4 Definition of Done (DoD)

Sourcecode

- CI auf GitLab-OST läuft durch:
  - Der komplette Code ist hochgeladen.
  - Für alle Funktionen müssen Unit-Tests geschrieben sein.
  - Alle Unit-Tests müssen durchlaufen und bestanden werden.
- Es muss ein erfolgreiches Review durchgeführt worden sein.
- Die funktionalen Anforderungen und nicht funktionalen Anforderungen müssen, auf das Modul bezogen, erfüllt werden.
- Jedes Modul muss repräsentativ dokumentiert sein.
- Für das Modul muss ein Task erstellt sein, auf welchem die Zeiten erfasst wurden.

Dokumentation

- Für die jeweilige Sektion muss ein Task erstellt sein, auf welchem die Zeiten erfasst wurden.
- Die Sektion muss mindestens von einer anderen Person gegengelesen sein.
- Das PDF muss ohne einen Fehler gebildet werden können.
- Die CI auf GitLab-OST muss durchlaufen.

Sitzung

- Für jede wöchentliche Sitzung wird ein Task erstellt.
- Das Protokoll ist geschrieben.
- Die Zeiten für Vorarbeit / Sitzung / Nachbesprechung und Schreiben des Protokolls sind erfasst.

5.2 Projektplan

Die Projektdauer der Studienarbeit beträgt 14 Wochen. Die Entwicklung der Applikation ist vertikal umgesetzt, damit so früh wie möglich ein Durchstich durch die ganze Applikation besteht. Im Zeitraum des Projekts wurden während der Planungsphase mehrere Meilensteine definiert, welche sukzessive zu erreichen sind. Die Meilensteine stellen dabei die langfristige Planung des Projekts dar und garantieren einen stetigen Fortschritt. Während den Iterationen wird auf die Erreichung des nächsten Meilensteins hingearbeitet. Eine Iteration ist jeweils eine Woche lang, zwischen den Sitzungen, welche montags stattfinden.

5.2.1 Meilensteine

Die Meilensteine stellen Fixpunkte in der Entwicklung dar. Das Hauptziel jeder Iteration ist die Erreichung des nächsten Meilensteins.
Folgende Meilensteine wurden definiert:

MS	Datum (SW)	Bezeichnung und Beschreibung
MS.01	25.10.21 (06)	**Erster Durchstich** 1. Der Student verifiziert sich nur mit dem Namen. 2. Der Student startet eine Auswertung (Ohne Hochladen). 3. Der Unit-Coordinator leitet den Auftrag weiter an den Job-Handle. 4. Der Job-Handle gibt eine Musterauswertung zurück. 5. Der Unit-Coordinator gibt die Auswertung auf dem Frontend aus.
MS.02	01.11.21 (07)	**Anmeldung über GitLab-OST** 1. Der Student verifiziert sich über GitLab-OST. 2. Nach erfolgreichem Verifizieren hat der Student Zugriff auf die Applikation. **Dateisystem** 1. Der Student startet eine Auswertung (mit Hochladen). 2. Der Unit-Coordinator speichert die Dateien. 3. Der Unit-Coordinator kopiert die Dateien des auszuführenden Projekts in ein temporäres Verzeichnis. 4. Der Unit-Coordinator gibt dem Job-Handle den Auftrag für die Auswertung. 5. Der Unit-Coordinator löscht das temporäre Verzeichnis. 6. Der Job-Handle gibt eine Default-Auswertung zurück. 7. Der Unit-Coordinator gibt die Auswertung auf dem Frontend aus.
MS.03	15.11.21 (09)	**Auswertung durch Job-Handle** 1. Der Job-Handle erhält einen Auftrag. 2. Der Job-Handle führt das Projekt aus. 3. Der Job-Handle gibt das Resultat der Ausführung dem Unit-Coordinator zurück. **Datenbank** 1. Der Unit-Coordinator holt die Daten für den Auftrag für den Job-Handle aus der Datenbank. 2. Der Unit-Coordinator speichert die erhaltene Auswertung in der Datenbank.

MS	Datum (SW)	Bezeichnung und Beschreibung
MS.04	29.11.21 (11)	**Erstellen eines Projekts** 1. Der Dozent verifiziert sich über GitLab-OST. 2. Der Dozent kann ein Projekt erstellen und konfigurieren. 3. Der Unit-Coordinator speichert die Konfiguration in der Datenbank. 4. Der Unit-Coordinator speichert die Dateien des Projekts auf dem Dateisystem ab. **Abfragen der Auswertungen** 1. Der Dozent will die Auswertungen eines Projekts einsehen. 2. Der Unit-Coordinator stellt aus den Daten in der Datenbank die Auswertung zusammen. 3. Der Unit-Coordinator gibt die zusammengestellten Daten den Dozenten aus.
MS.05	06.12.21 (12)	**MVP** • Erreichung allen Anforderungen für das MVP.
MS.06	24.12.21 (14)	**Projektabgabe** • Dokumentation • Projekt • Abstract • Plakat

Tabelle 5.1: Meilensteine

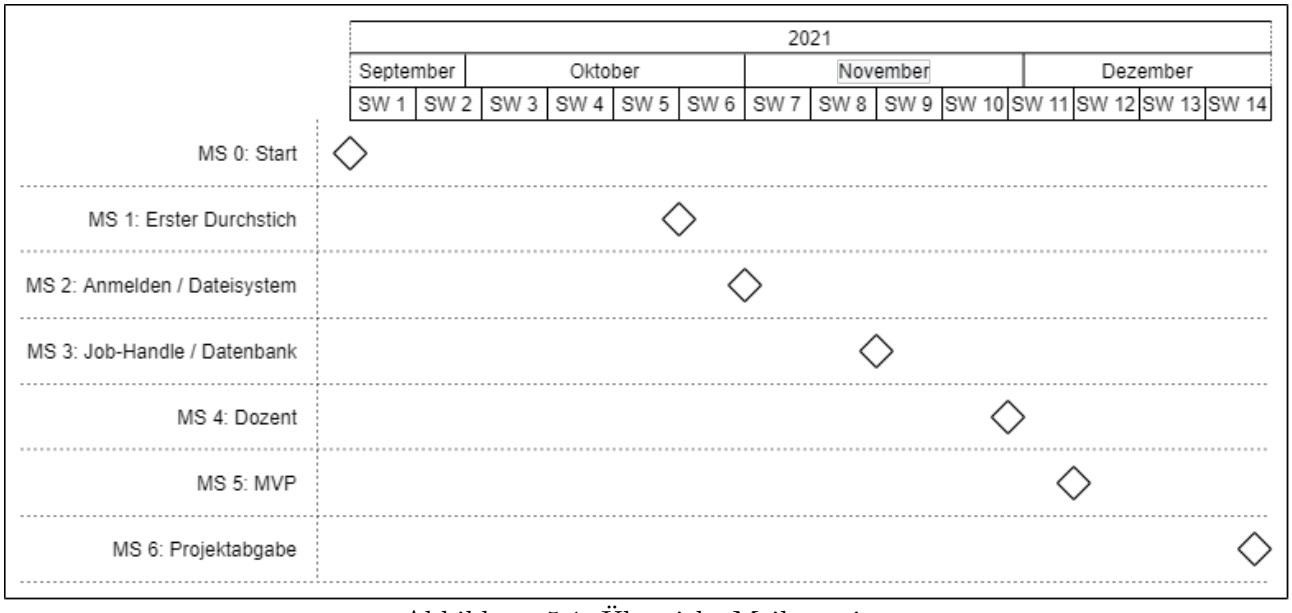


Abbildung 5.1: Übersicht Meilensteine

5.3 Projektorganisation

5.3.1 Organisationsstruktur

Roman Spring (roman.spring@ost.ch):	Entwickler
Pascal Gsell (pascal.gsell@ost.ch):	Entwickler
Thomas Corbat (thomas.corbat@ost.ch):	Betreuer (Studienarbeit)
Nicola Jordan (nicola.jordan@ost.ch):	Betreuer (Fachwissen)

5.3.2 Externe Schnittstellen

Betreuer (Studienarbeit):	Thomas Corbat (thomas.corbat@ost.ch)
Betreuer (Fachwissen):	Nicola Jordan (nicola.jordan@ost.ch)
OST – Virtual Server Kiosk:	Christian Spielmann (christian.spielmann@ost.ch)

5.3.3 Dokumentation

Die Dokumentation befindet sich in einem GitLab-OST Projekt: ALF-Dokumentation.

5.3.4 Entwicklung

Das Entwicklungsprojekt befindet sich in einem GitLab-OST Projekt: ALF-Sourcecode.

5.3.5 Besprechungen

Um den Projektfortschritt zu sichern, wird mit den Betreuern einmal in der Woche, zu Beginn der Iteration, eine Sitzung gehalten. Die Sitzungen sind darauf ausgelegt, die vergangene Woche zu reflektieren und die kommende Woche zu besprechen. Die jeweiligen Protokolle sind unter Anhang E. Sitzungsprotokolle zu finden.

Kapitel 6

Resultate

Dieses Kapitel beinhaltet den Soll-Ist-Vergleich der Benutzeroberfläche, der funktionalen und nicht funktionalen Anforderungen, sowie eine Analyse der Erreichung der Meilensteine. Zudem wird das Testen des ALF-Sourcecodes aufgegriffen und analysiert. Am Schluss des Kapitels wird noch kurz auf die zusätzlichen Funktionen eingegangen.

6.1 Benutzeroberfläche

6.1.1 Vergleich zu den Wireframes

Hier wird ein kurzer Vergleich zwischen den Wireframes (Abschnitt F.1. Wireframes) und dem schlussendlichen Resultat der Studienarbeit gemacht.
Der Grundgedanke des Produkts entspricht im Grossen und Ganzen den Wireframes. Hauptsächlich sind die Namensgebungen und das Design leicht angepasst. Als Beispiel sehen Dozenten nun keine zwei Menüpunkte mehr, da entschieden wurde, dass auch ein Menüpunkt genügt.
Ein anderes Beispiel wären die Formulare, welche in den Wireframes mit zwei Spalten gezeichnet sind, aber nur mit einer Spalte implementiert wurden. Zu dem sind bei den Formularen zum Teil die Typen der Felder angepasst oder neue Felder hinzugefügt worden. Das Resultat dieser Anpassungen sieht man als Beispiel in der Abbildung 6.1. Ändern eines Projekts.

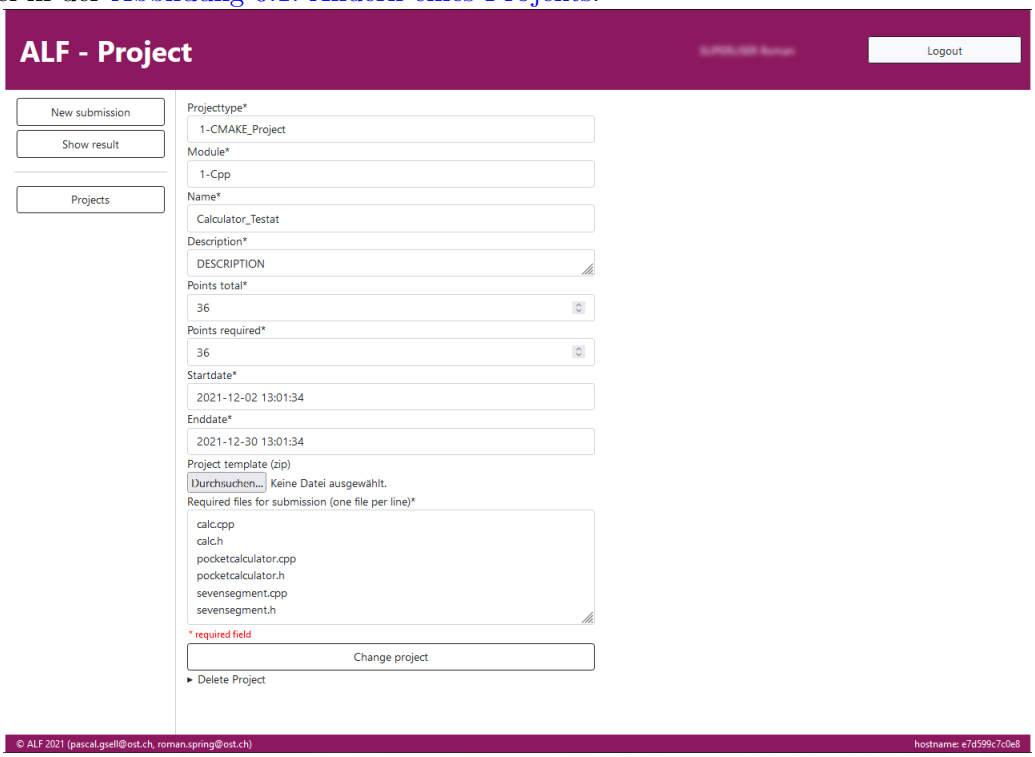


Abbildung 6.1: Ändern eines Projekts

6.1.2 Dropdowns

Um die Benutzer des Tools nicht zu überfordern und die Bedienung zu vereinfachen, wurde die Auswahl in den Dropdowns der Formulare beschränkt.

New Submission - Project pick
Für die Abgabe müssen die Studenten das Projekt auswählen, in welchem sie ihre Abgabe tätigen wollen (siehe Abbildung C.5. Auswahl eines Projekts).
Es werden nur die Module angezeigt, welche auch gültige Projekte besitzen, da die Auswahl eines Moduls ohne gültige Projekte keinen Sinn ergibt. Nach der Auswahl des Moduls werden alle gültigen Projekte zu diesem Modul angezeigt. Als gültige Projekte gelten alle, welche ein Startdatum in der Vergangenheit haben und bei welchen noch nicht das Enddatum erreicht ist.
Die Dozenten sehen hier nur die Module, in welchen sie eingetragen sind. Dementsprechend sehen sie auch nur die Projekte von diesen Modulen. Dies wurde entschieden, weil die Dozenten nur auf diese Formulare gehen, um zu sehen, wie die Studentensicht ihrer Projekte ist.

New Submission - Submit
Eine Abgabe kann nicht nur von einem Studenten, sondern auch von einem Team abgegeben werden (siehe Abbildung C.6. Hochladen einer Lösung).
Das Auswahlfeld der Benutzer für das Team wird nicht gefiltert. Es werden alle registrierten Studenten aufgeführt. Dies kommt daher, dass man keine Möglichkeit hat, diese Liste zu filtern, da jeder Student ein potenzielles Teammitglied sein kann.

Show Result
Damit die Studenten das Resultat einer Abgabe einsehen können, müssen sie diese zuerst auswählen (siehe Abbildung C.8. Auswahl einer Abgabe).
Den Studenten werden nur die Abgaben angezeigt, welche sie getätigt haben oder im Team markiert wurden. Den Dozenten werden alle Abgaben von den Modulen angezeigt, in welchen sie als Dozent eingetragen sind. Es werden alle Projekte ausgeblendet, welche keine Abgaben aufweisen.

Projects - Project selection
Die Dozenten möchten ein Projekt bearbeiten, testen oder einfach die Abgaben einsehen (siehe Abbildung C.11. Auswahl des Projekts).
Den Dozenten werden alle Module angezeigt, bei welchen sie als Dozent eingetragen sind. Nach einer Auswahl eines Moduls werden die Projekte anhand dieses Moduls gefiltert.

Superuser
Der Superuser stellt bei den Dropdowns einen Sonderfall dar. Er sieht ohne Einschränkungen alle Optionen und kann diese auch verwenden.

6.2 Auswertung der Anforderungen

Das Hauptziel einer jeder Arbeit ist es, die definierten Anforderungen zu erfüllen. Das ALF-Tool, welches im Rahmen dieser Studienarbeit erstellt wurde, erfüllt alle Anforderungen der Stufe MUSS, SOLL und KANN vollumfänglich.

Im Nachfolgenden sind die Anforderung und ihre Erfüllungsgrade aufgelistet:

6.2.1 Funktionale Anforderungen

Nr.	Anforderung	Gewichtung	Erreicht
Allgemein			
FA.A.01	Die Applikation wertet CMake-C++-Cute Projekte aus.	MUSS	Ja
FA.A.02	Das System wertet die Studenten-Lösungen aus.	MUSS	Ja
FA.A.03	Alle Studenten-Lösungen werden gespeichert.	MUSS	Ja
FA.A.04	Dateien werden über ein Frontend hochgeladen.	SOLL	Ja
FA.A.05	Das System speichert alle Auswertungen.	SOLL	Ja
Student			
FA.S.01	Der Student lädt seine Lösung einer Aufgabe hoch.	SOLL	Ja
FA.S.02	Der Student erhält eine Auswertung seiner Lösung.	SOLL	Ja
FA.S.03	Der Student lädt ein Testat zur Bewertung hoch.	KANN	Ja
FA.S.04	Der Student kann mit anderen Studenten ein Team bilden für eine Abgabe.	KANN	Ja
FA.S.05	Der Student sieht bei Projekten die jeweiligen Deadlines.	KANN	Ja
Dozent			
FA.D.01	Der Dozent erstellt ein Projekt eigenständig.	MUSS	Ja
FA.D.02	Der Dozent löscht ein Projekt eigenständig.	MUSS	Ja
FA.D.03	Der Dozent ändert seine Projekte eigenständig.	MUSS	Ja
FA.D.04	Der Dozent sieht die Auswertungen der Aufgaben.	MUSS	Ja
FA.D.05	Der Dozent erstellt ein Projekt über das Frontend.	SOLL	Ja
FA.D.06	Der Dozent löscht ein Projekt über das Frontend.	SOLL	Ja
FA.D.07	Der Dozent ändert seine Projekte über das Frontend.	SOLL	Ja
FA.D.08	Der Dozent sieht die Auswertungen der Aufgaben im Frontend.	SOLL	Ja
FA.D.09	Der Dozent lädt die hochgeladenen Abgaben herunter.	KANN	Ja

Tabelle 6.1: Überprüfung funktionaler Anforderungen

6.2.2 Nicht funktionale Anforderungen

Nr.	Anforderung	Gewichtung	Erreicht
Allgemein			
NFA.A.01	Nur die Applikation hat direkten Zugriff auf das Dateisystem und die Datenbank.	MUSS	Ja
NFA.A.02	Die Applikation läuft 24/7 mit einer maximalen Ausfallrate von 1% (Auf 1 Jahr ca. 3 Tage Ausfall).	SOLL	—
NFA.A.03	Die Applikation ist dockerisiert.	SOLL	Ja
NFA.A.04	Ein Fehler bei der Auswertung führt nicht zu einem Absturz der Applikation.	SOLL	Ja
NFA.A.05	Die Applikation bricht die Berechnung der Resultate ab, wenn sie ein Zeitlimit überschreitet.	SOLL	Ja
Student			
NFA.S.01	Der Student wird autorisiert.	MUSS	Ja
NFA.S.02	Der Student sieht nur die Resultate seiner Abgaben.	MUSS	Ja
NFA.S.04	Der Student erhält spätestens 5 Sekunden nach dem Hochladen einer Abgabe eine Reaktion.	KANN	Ja
NFA.S.05	Der Student sieht nach spätestens 5 Sekunden nach dem Time-Out der Berechnungen ein Resultat für die Abgabe.	KANN	Ja
Dozent			
NFA.D.01	Der Dozent wird autorisiert.	MUSS	Ja
NFA.D.02	Der Dozent kann nur eigene Projekte einsehen und ändern.	MUSS	Ja
NFA.D.03	Der Dozent sieht nach spätestens 5 Sekunden eine Reaktion beim Erstellen oder Ändern eines Projekts.	KANN	Ja
NFA.D.04	Der Dozent sieht nach spätestens 5 Sekunden eine Reaktion beim Aufruf der Übersichtsseite.	KANN	Ja

Tabelle 6.2: Überprüfung nicht funktionaler Anforderungen

Bemerkung (NFA.A.02): Ist in einer Studienarbeit von 14 Wochen nicht testbar, da der nötige Meilenstein 5 erst am Ende der Projektzeit steht. Somit kann die Anforderung auch nicht prozentual geprüft werden.

6.2.3 Übersicht aller Anforderungen

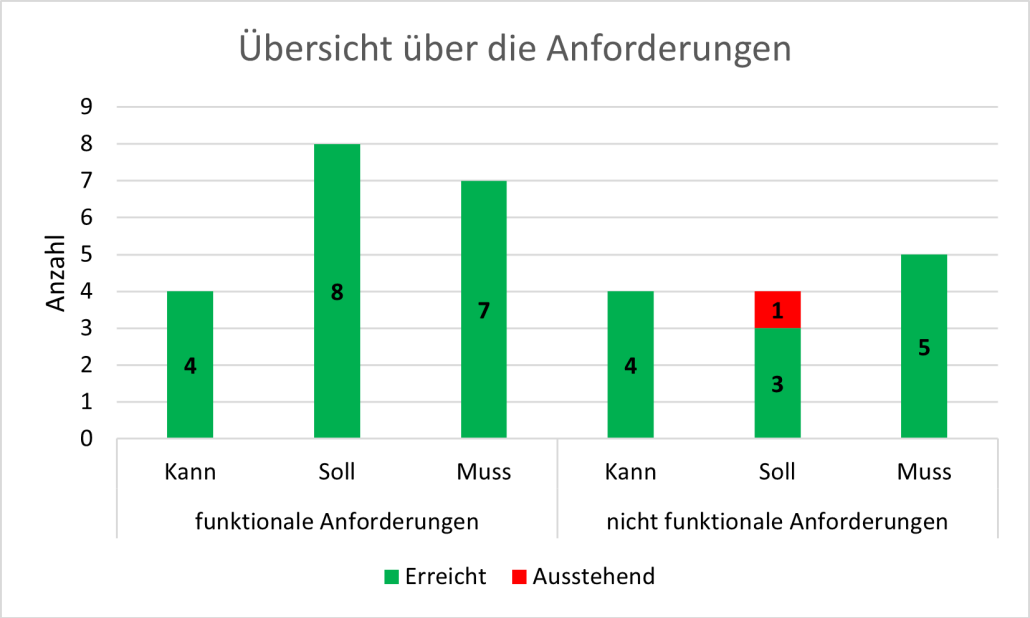


Abbildung 6.2: Übersicht aller Anforderungen

6.3 Zielerreichung Meilensteine

Alle folgenden Punkte beziehen sich auf Unterabschnitt 5.2.1. Meilensteine. Die einzelnen Verzögerungen und Abhängigkeiten sind zusätzlich in der Abbildung 6.3. Meilensteine Soll-Ist-Vergleich visualisiert.

- **Meilenstein 1 - Erster Durchstich**
  Nach langer Unklarheit, was die genauen Ziele der Studienarbeit sind, wurde dennoch der erste Meilenstein rechtzeitig erreicht. Durch die Erreichung dieses Meilensteins wurde zugleich eines der grössten Risiken eliminiert (siehe R.P.01 in Abschnitt 2.7. Risikoanalyse). Mit dieser Erreichung konnte aufgezeigt werden, dass die grundsätzliche Idee hinter der Architektur funktioniert.

- **Meilenstein 2 - Anmelden über GitLab-OST / Dateisystem**
  Während der Implementation des zweiten Meilensteins wurde festgestellt, dass dieser nicht ohne die Datenbank, welche erst im nächsten Meilenstein implementiert werden sollte, möglich ist. Dies rührt daher, dass in Django das Frontend, beziehungsweise die Formulare, jeweils Datenbankmodelle benötigen. Um die Planung nicht komplett zu verwerfen, wurde entschieden, dass der Meilenstein 2 mit dem Meilenstein 3 abgeschlossen wird. Daher hat sich die Erreichung des zweiten Meilensteins um zwei Wochen nach hinten verschoben.

- **Meilenstein 3 - Datenbank / Auswertung Job-Handle**
  Wie beim zweiten Meilenstein erwähnt, wurde der Meilenstein 3 mit dem Meilenstein 2 abgeschlossen. Beide Meilensteine wurden pünktlich zum Termin des dritten Meilensteins erreicht.

- **Meilenstein 4 - Erstellen eines Projekts / Abfragen der Auswertungen**
  Die Umstrukturierung des kompletten Frontends auf Bootstrap und die Konzentration auf die ALF-Dokumentation hätte beinahe dazu geführt, dass der vierte Meilenstein erst verspätet erreicht worden wäre. Mit etwas zusätzlichem Einsatz wurde der Meilenstein dennoch pünktlich erreicht.

- **Meilenstein 5 - MVP**
  Der Meilenstein 5 wurde wegen kleineren Fehler im ALF-Sourcecode nicht termingerecht erreicht. Das Korrigieren dieser Fehler benötigte zwei weitere Tage, wodurch sich die Erreichung des Meilensteins um diese beiden Tage verschoben hat.

- **Meilenstein 6 - Projektabgabe**
  Da der Meilenstein 6 die Abgabe der Studienarbeit selbst darstellt, wurde dieser zwangsweise eingehalten.

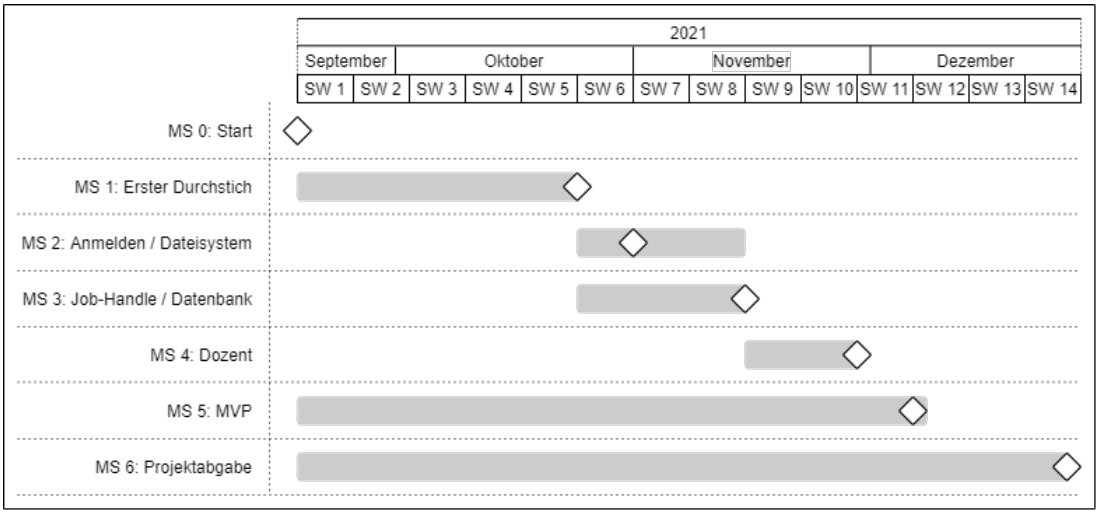


Abbildung 6.3: Meilensteine Soll-Ist-Vergleich

6.4 Test-Coverage

In den ALF-Pages kann der aktuelle Stand der Test-Coverage auf dem Master-Branch eingesehen werden. Die verwendeten Pfadangaben sind im Unterabschnitt 4.3.6. Server ausführlicher beschrieben.

Eine hohe Test-Coverage sagt aus, wie gut der ALF-Sourcecode abgedeckt ist. Sie sagt aber nichts darüber aus, wie gut die einzelnen Funktionen getestet sind. Um möglichst viele Fälle abzudecken, wurden die Unit-Tests von einem zweiten Entwickler nachgeprüft, in der Hoffnung möglichst viele relevante Fälle abzudecken.
Wenn bei der Implementation ein Fehler auftauchte, wurde während dem Beheben zugleich ein entsprechender Unit-Test geschrieben, welcher den Fehler in Zukunft abfangen sollte.
Im folgenden Abschnitt ist die momentane Test-Coverage dokumentiert.

Job-Handle
Das Teilprojekt Job-Handle beinhaltet Tests für alle Module innerhalb des Projekts (siehe Tabelle 6.3. Test-Coverage des Job-Handles). Die Module *start*, *job_handle* und *cmake_cpp_cute_runner* werden komplett durch Unit-Tests abgedeckt. Die Module *command_handle* und *project_runner* hingegen werden zusätzlich mit Integrations-Tests geprüft. Die meisten Module werden mit positiven und negativen Fällen abgedeckt, was zu einer guten Testabdeckung aller Funktionen beiträgt. Wo sinnvoll, wurden Tests parametriert, um unterschiedliche Kombinationen zu testen.

Datei	**Statements**		**Test-Coverage**
	Anzahl	Ungetestet	
job_handle/src/cmake_cpp_cute_runner.py	80	0	100%
job_handle/src/command_handle.py	25	0	100%
job_handle/src/job_handle.py	25	0	100%
job_handle/src/project_runner.py	19	0	100%
job_handle/src/start.py	14	0	100%
Total	163	0	**100%**

Tabelle 6.3: Test-Coverage des Job-Handles

Unit-Coordinator
Das Teilprojekt Unit-Coordinator beinhaltet Tests nur für eigen geschriebene Module und auch nur, wenn sie nicht Django spezifisch sind. Daher wurden alle Konfigurationen im *unit_coordinator/unit_coordinator/* Package nicht getestet, sowie auch die Module *admin*, *apps*, *models* und *urls* im *unit_coordinator/unit_coordinator_app/* Package (siehe Tabelle 6.4. Test-Coverage des Unit-Coordinators). Die Module *forms*, *paths*, *permission_checks*, *thread_worker* und *validators* aus dem Package *unit_coordinator/unit_coordinator_app/* wurden im Verhältnis zu anderen Modulen besser getestet, da diese in mehreren Modulen importiert werden. Das Modul *views* aus dem Package *unit_coordinator/unit_coordinator_app/* wurde hauptsächlich für die verwendeten Funktionalitäten getestet, wodurch nicht alle möglichen Fälle abgedeckt sind.

Datei	Statements		Test-Coverage
	Anzahl	Ungetestet	
unit_coordinator/forms.py	89	0	100%
unit_coordinator/paths.py	25	0	100%
unit_coordinator/permission_checks.py	10	0	100%
unit_coordinator/thread_worker.py	15	0	100%
unit_coordinator/validators.py	25	0	100%
unit_coordinator/views.py	197	0	100%
unit_coordinator/views_bases.py	41	0	100%
unit_coordinator/views_helpers.py	175	0	100%
Total	577	0	**100%**

Tabelle 6.4: Test-Coverage des Unit-Coordinators

Allgemeine Packages

Die Module *alf_logger* und *alf_socket* wurden gegenüber anderen Modulen sehr ausführlich getestet. Die Test-Dateien sind jeweils mehr als dreifach so lange, wie die eigentlichen Dateien. Dies wurde angestrebt, da diese Packages in beiden Teilprojekten Unit-Coordinator und Job-Handle verwendet werden und einen wichtigen Bestandteil des Produkts darstellen. Trotzdem sind dies keine Garantien, dass alle nötigen Fälle abgedeckt sind und die allgemeinen Packages keine Fehler mehr aufweisen. In der Tabelle 6.5. Test-Coverage der allgemeinen Packages sind die Test-Coverages der einzelnen Packages aufgeführt.

Datei	Statements		Test-Coverage
	Anzahl	Ungetestet	
alf_logger/alf_logger.py	23	0	100%
alf_socket/alf_socket.py	62	0	100%
Total	85	0	**100%**

Tabelle 6.5: Test-Coverage der allgemeinen Packages

Zusammenfassung

In der nachfolgenden Tabelle sind die Test-Coverages der einzelnen Teilprojekte zusammengefasst.

Teilprojekt	Statements		Test-Coverage
	Anzahl	Ungetestet	
Job-Handle	163	0	100%
Unit-Coordinator	577	0	100%
Allgemeine Packages	85	0	100%
Total	825	0	**100%**

Tabelle 6.6: Zusammenfassung der Test-Coverage

6.5 Zusätzliche Features

Während der Studienarbeit wurden gewisse Features zusätzlich zu den Anforderungen entwickelt. Diese werten das ALF-Tool zusätzlich auf und sollten daher nicht unerwähnt bleiben.

6.5.1 Dokumentierte Features

Studentenansicht - Auswahl der Abgabe
Der Student kann alle jemals getätigten Abgaben durch einfache Auswahl von Projekt und Abgabe betrachten. Die Abgabe kann auch nach Projektende noch ausgewählt und betrachtet werden. Dies ist im Unterabschnitt C.3.2. Einsehen einer abgegebenen Lösung zu sehen.

Studentenansicht - Download früherer Abgaben
Zusätzlich zur Betrachtung der früheren Abgaben, können die damalig hochgeladenen Dateien erneut heruntergeladen werden. Die Abgaben können über einen Download-Button, auf der Detailansicht der Abgabe selbst, wieder heruntergeladen werden. Dies ist im Unterabschnitt C.3.2. Einsehen einer abgegebenen Lösung ersichtlich.

Dozentenansicht - Testen eines Projekts
Die Dozenten haben die Möglichkeit, ein von ihnen eingerichtetes Projekt direkt im ALF-Tool zu testen. Dies ist im Unterabschnitt C.4.5. Projekt testen zu sehen. Dabei können die Dozenten direkt prüfen ob alle nötigen Dateien richtig konfiguriert wurden. Die Testresultate werden nicht in der Datenbank persistiert und keine Datei auf dem Dateisystem zurückgelassen. Sind die Resultate vorhanden, wird den Dozenten die gleiche Ansicht wie für Studenten gezeigt, womit diese ebenfalls überprüft werden kann.

6.5.2 Undokumentierte Features

Dank enthusiastischem Einsatz der Entwickler wurden die nachfolgenden Features erst kurz vor Abgabe des Projekts noch entwickelt. Daher sind die folgenden Features in dieser Dokumentation nicht widerspruchsfrei dokumentiert und das erreichte Produkt weicht vom dokumentierten Produkt leicht ab.

Dozentenansicht - Enddatum vor Startdatum
Wird ein neues Projekt angelegt oder ein bestehendes Projekt geändert, kann dabei das Enddatum nicht vor dem Startdatum gewählt werden.

Dozentenansicht - Download der Template-Dateien
Möchten Dozenten ein Projekt für das Tool entwickeln, können sie die Projettyp-Template-Dateien und Projet-Template-Dateien herunterladen. Dazu wird allerdings ein bestehendes Projekt benötigt. Möchte ein Dozent nur die Projekttyp-Template-Dateien herunterladen, erstellt er ein neues Projekt und lädt einen leeren ZIP-Ordner als Projekt-Template hoch. Danach kann er sich die Template-Dateien herunterladen, welcher so nur aus den Projekttyp-Template-Dateien besteht.

Team aktivieren / deaktivieren
Durch das hinzufügen eines binären Wertes zum Projekt, kann nun die Teambildung aktiviert / deaktiviert werden. Dadurch kann das Problem (Spammen von Mitstudierenden) eingeschränkt werden.

Kapitel 7

Zusammenfassung

Das Kapitel Zusammenfassung ist dem Kapitel Resultate ähnlich, nur das hier im Allgemeinen das Projekt und nicht die einzelnen Ergebnisse zusammengefasst werden. Neben dem Abschnitt Allgemeine Zusammenfassung ist hier auch der Abschnitt Zeiterfassung enthalten. Da ein Projekt, egal wie viel Zeit es bekommt, nie fertig wird, sind zudem die Abschnitte Offene Punkte und Ausblick aufgeführt.

7.1 Allgemeine Zusammenfassung

Im Rahmen dieser Arbeit wurde ein bestehendes Tool neu entwickelt, welches Abgaben von Studenten zu spezifischen Projekten mit Unit-Tests prüft und analysiert. In diesem wurden bestehende, sowie neue Anforderungen umgesetzt.
Zu Beginn wurde die Analyse des bestehenden Tools durchgeführt. Mit der Analyse der neuen Anforderungen stellte dies die Grundlage der weiteren Arbeit dar. Da bereits früh mit einem Durchstich durch alle Technologien begonnen wurde, konnten die verwendeten Technologien rasch verifiziert werden. Eine frühe Risikoanalyse zeigte auf, dass die verwendeten Technologien auch zukünftig keine grossen Überraschungen bergen sollten. Im weiteren Verlauf des Projekts wurden stets weitere Features hinzugefügt.
Als erste Features kamen das Anmelden über GitLab-OST und das Dateisystem hinzu. Für das Dateisystem wurde ein separates Package geführt, welches sich im Verlaufe des Projekts aber als unnötig herausstellte und eingestellt wurde. Diese Features deckten ein Hochladen der Dateien, sowie eine Auswertung durch den Job-Handle ab. In dieser Stufe wurden übermittelte Dateien noch ignoriert und ein Muster-Resultat zurück übermittelt, welches der Unit-Coordinator dann präsentierte.
In einem nächsten Schritt wurde die Ausführung von CMake-C++-Cute Projekten implementiert. Diese dient als beispielhafte Implementation, sowie deckt sie den alten Funktionsumfang des vorangegangenen Tools ab. Um die nun ausgewerteten Resultate ebenfalls im Unit-Coordinator abzubilden, wurden die Resultate nun ebenfalls in der Datenbank gespeichert.
In einem letzten Schritt wurden die Verwaltung der Projekte, das Testen eines Projekts, sowie die Anzeige aller Abgaben für Dozenten implementiert. Mit dem Erreichen dieser Features wurde das MVP erreicht.
Am Schluss wurden noch kleinere Fehlerbehebungen und Implementationen von zusätzlich gewünschten Feature durchgeführt.

Mit dem erreichten Produkt können Dozenten selbstständig Projekte einrichten und lancieren. Dozenten können den Verlauf der Abgaben beobachten und haben die Möglichkeit alle Dateien der Abgaben herunterzuladen. Dozenten wie Studenten können sich über GitLab-OST authentifizieren und benötigen so kein separates Konto. Die Pflege von Studenten benötigt deswegen keinen Administrationsaufwand, nur für das Erteilen der Rechte für Dozenten und das Erstellen von Modulen benötigt es einen Administrator (siehe Unterabschnitt B.1.1. Dozenten registrieren und Unterabschnitt B.1.2. Modul anlegen). Studenten können Abgaben durchführen und haben die Möglichkeit frühere Abgaben erneut einzusehen, sowie die Dateien der Abgaben wieder herunterzuladen. Mit diesen Features deckt das entwickelte Produkt nicht nur die Anforderung des bestehenden Tools, sondern auch die zusätzlichen Anforderungen ab.

7.2 Offene Punkte

Das entwickelte Produkt weist keine offenen Punkte gemäss den Anforderungen auf.
Idealerweise würde dieser Satz hier alleine stehen, aber auch dieses Produkt ist nicht vollkommen.

Nebenläufiges Downloaden
Werden nebenläufig zwei oder mehrere Downloads innerhalb derselben Projekte angefordert, treten Überschneidungen beim Löschen des *archives* Ordners auf. Dadurch werden noch benötigte Dateien von einer zweiten Anfrage gelöscht, wodurch die erste Anfrage nicht korrekt abgeschlossen werden kann. Dies kann dazu führen, dass Archive nicht alle Dateien enthalten oder Anfragen abstürzen und unbeantwortet bleiben.
Durch ein periodisches Löschen der Dateien, zum Beispiel jeden frühen Morgen, welches extern gestartet wird, könnte das Problem behoben werden. Eine andere Möglichkeit wäre, das Löschen gänzlich weg zu lassen, dies würde aber bei langlebigen Projekten zu enormem Datenmüll führen.

Unsicherer GET in *django-allauth*
Nach dem Meilenstein 5 wurde das Package *django-allauth* noch aktualisiert, wobei auf eine mögliche Sicherheitslücke hingewiesen wird. Die direkte Weiterleitung bei einer GET-Anfrage zu einer Dritt-Anbieter-Software (GitLab-OST) kann ungenügend sicher sein. Daher wird standardmässig eine Zwischenseite angezeigt, bei welcher der Benutzer ein Button klicken muss, um eine POST-Anfrage abzusetzen. Zurzeit ist der Unit-Coordinator so konfiguriert, dass er GET-Anfragen zulässt.

Studentenansicht - Spammen von Mitstudierenden
Möchte ein Student seinen Mitstudierenden Schaden zufügen oder dem Dozenten einfach mehr Arbeit aufhalsen, kann ein Student kurz vor Endtermin eines Projekts eine schlechte Abgabe übermitteln. Bei dieser schlechten Abgabe könnten alle Mitstudierenden in das Team hinzugefügt werden, wodurch damit diese Abgabe als letzte gültige Abgabe für jeden Mitstudierenden gilt. Dies führt dazu, dass die Übersicht über alle Abgaben für den Dozenten unbrauchbar wird. Der Dozent kann noch durch alle Abgaben durchgehen oder kann alle Abgaben herunterladen und dort alle einzeln durchgehen, aber das Tool verliert damit seine nützliche Funktionalität.
Trifft dies ein, ist es am einfachsten die entsprechende Abgabe von einem Administrator löschen zu lassen.

7.3 Zeiterfassung

7.3.1 Arbeitszeit über die Semesterwochen

In der Semesterwoche sechs wurde von einem anderen Modul eine Testatabgabe erwartet, daher ist dort ein Rückgang zu verzeichnen. Da nach der Semesterwoche sieben die Meilensteine 2 und 3 anstanden, wurde in dieser Woche mehr gearbeitet, um die fehlende Zeit wieder aufzuholen. Zusätzlich wurde in den letzten Wochen mehr gearbeitet um die Dokumentation auf den entsprechenden Stand zu bringen. Da der Ausdruck der Studienarbeit bereits Donnerstag stattfand, fehlten in der letzten Woche die Hauptarbeitstage (Freitag, Samstag und Sonntag).
Die nachfolgenden Abbildungen, Abbildung 7.1. Arbeitszeit über Semesterwochen und Abbildung 7.2. Summierter Zeitverlauf nach Arbeitstyp, zeigen den Arbeitseinsatz über die Semesterwochen, sowie eine Verteilung nach Arbeitstyp.

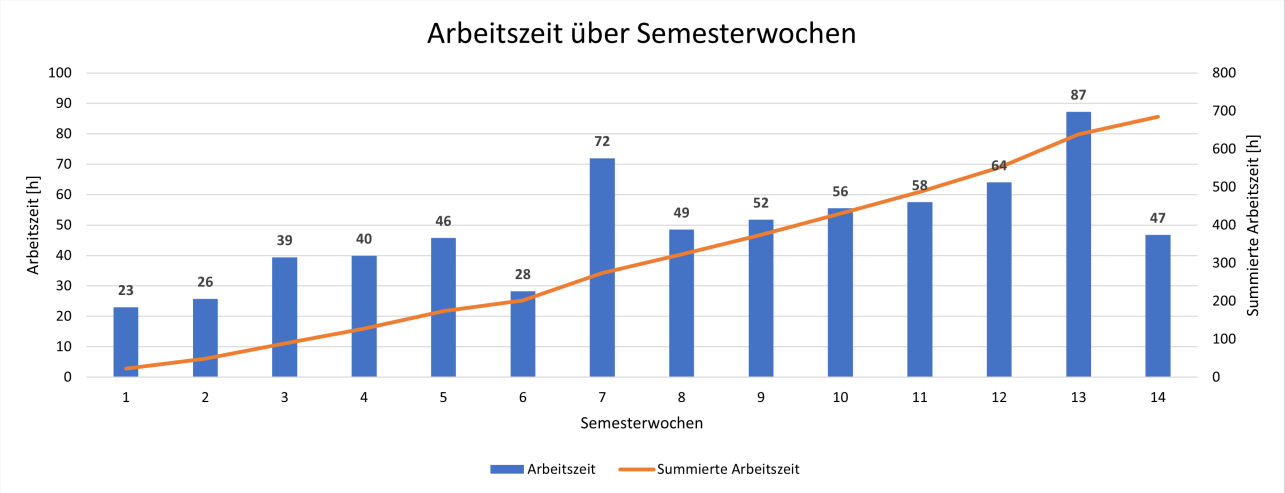


Abbildung 7.1: Arbeitszeit über Semesterwochen

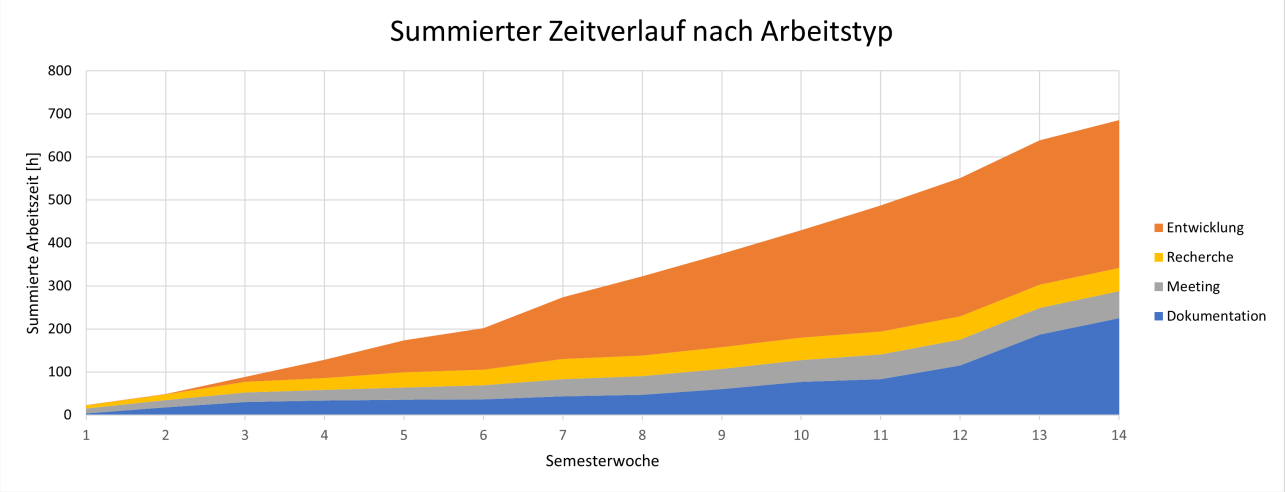


Abbildung 7.2: Summierter Zeitverlauf nach Arbeitstyp

7.3.2 Vergleich der Entwickler

Im Grossen und Ganzen ist die Arbeitsteilung unter den Entwicklern erstaunlich gut gelungen. Alle haben mehr oder weniger gleich viel gearbeitet. Auch die Aufteilung der Gebiete ist über die ganze Arbeit hinweg ausgeglichen.
Die nachfolgenden Abbildungen, Abbildung 7.3. Vergleich der Gesamtarbeitszeiten pro Entwickler und Abbildung 7.4. Vergleich der Gesamtarbeitszeiten pro Arbeitstyp, zeigen die Vergleiche der Gesamtarbeitszeit und nach Arbeitstyp.

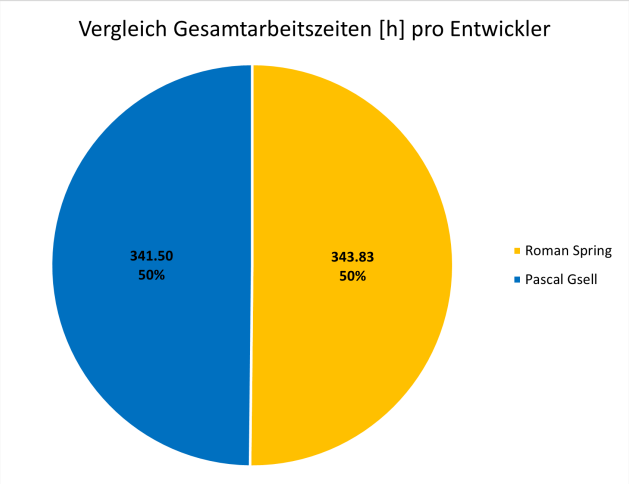


Abbildung 7.3: Vergleich der Gesamtarbeitszeiten pro Entwickler

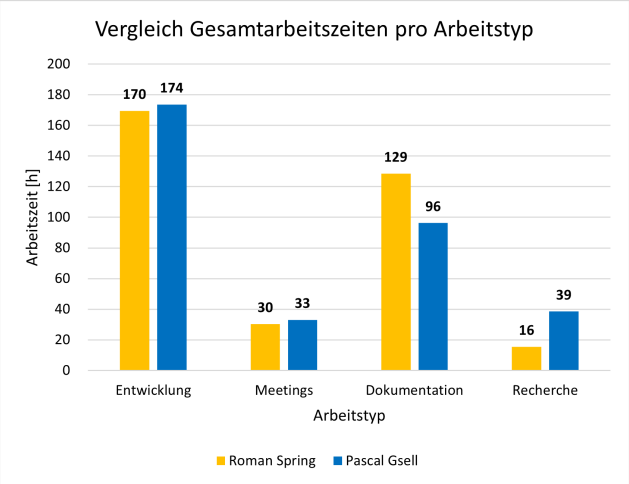


Abbildung 7.4: Vergleich der Gesamtarbeitszeiten pro Arbeitstyp

7.3.3 Aufteilung der Arbeitstypen

Wie man bei einem Softwareprojekt sieht, bei welchem der Hauptfokus auf der Entwicklung liegt, sind der Entwicklungs- und Dokumentationsanteil die Grössten (siehe Abbildungen 7.5 - 7.8). Bis auf die letzten Wochen war der Entwicklungsanteil mit Abstand am grössten (siehe Abbildung 7.2. Summierter Zeitverlauf nach Arbeitstyp).

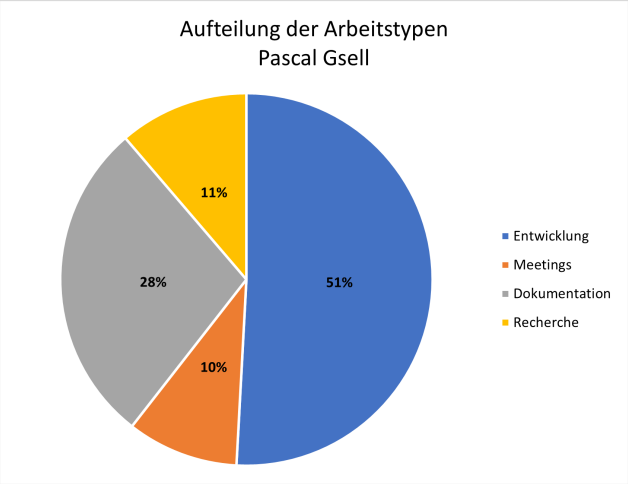


Abbildung 7.5: Aufteilung der Arbeitstypen - Pascal Gsell

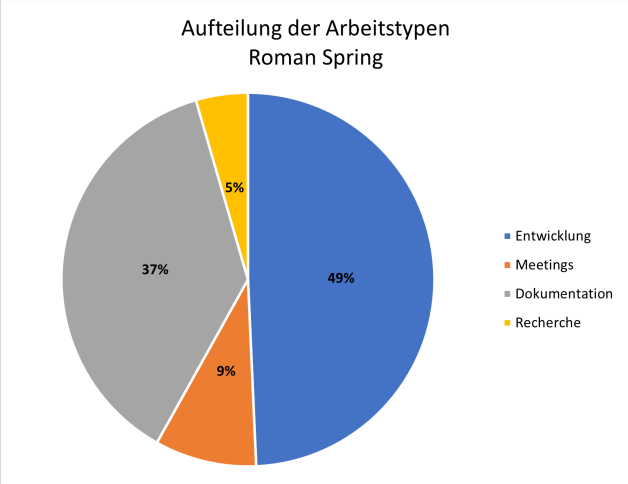


Abbildung 7.6: Aufteilung der Arbeitstypen - Roman Spring

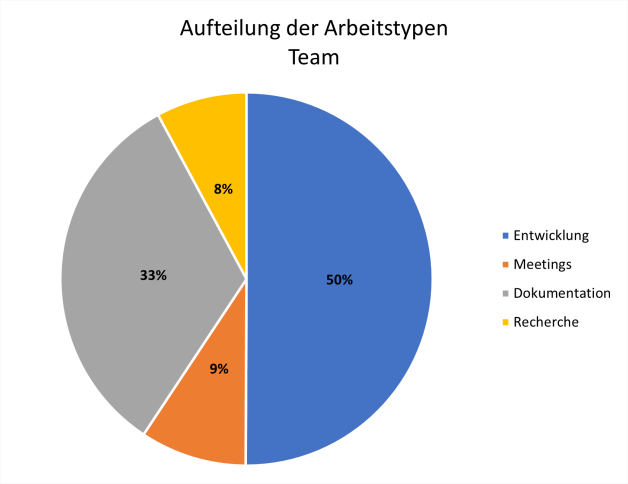


Abbildung 7.7: Aufteilung der Arbeitstypen - Team

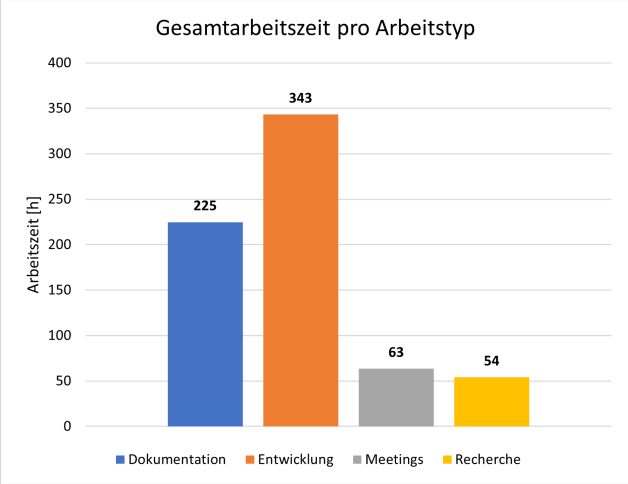


Abbildung 7.8: Gesamtarbeitszeit pro Arbeitstyp

7.3.4 Zeittabellen

Die folgenden Tabellen, Tabelle 7.1. Arbeitszeit über die Semesterwochen nach Arbeitstypen und Tabelle 7.2. Arbeitszeit über die Arbeitstypen nach Entwickler, sind der Vollständigkeit halber angefügt. Alle Abbildungen im Abschnitt 7.3. Zeiterfassung sind aus diesen Daten generiert.

		Arbeitszeit [h]				
Datum	**SW**	**Total**	**Entwicklung**	**Meeting**	**Dokumentation**	**Recherche**
20.09 - 26.09	1	23.00	1.00	11.00	3.50	7.00
27.09 - 03.10	2	25.75	0.00	5.50	14.50	5.75
04.10 - 10.10	3	39.42	9.75	4.67	12.50	12.50
11.10 - 17.10	4	39.92	31.42	3.50	2.75	2.25
18.10 - 24.10	5	45.75	32.75	3.00	2.00	8.00
25.10 - 31.10	6	28.25	21.75	4.50	1.00	1.00
01.11 - 07.11	7	72.00	47.25	7.25	7.00	10.50
08.11 - 14.11	8	48.50	40.50	3.00	4.00	1.00
15.11 - 21.11	9	51.75	32.00	3.50	13.25	3.00
22.11 - 28.11	10	55.50	33.50	4.00	16.50	1.50
29.11 - 05.12	11	57.50	43.25	7.00	6.50	0.75
06.12 - 12.12	12	64.00	29.00	2.50	31.75	0.75
13.12 - 19.12	13	87.25	13.50	2.00	71.75	0.00
20.12 - 26.12	14	46.75	7.50	1.50	37.75	0.00
Total		**685.33**	**343.17**	**63.42**	**224.75**	**54.00**

Tabelle 7.1: Arbeitszeit über die Semesterwochen nach Arbeitstypen

Arbeitstyp	**Roman Spring**	**Pascal Gsell**
Entwicklung	169.50	173.67
Meeting	30.33	33.08
Dokumentation	128.50	96.25
Recherche	15.50	38.50
Total	**343.83**	**341.50**

Tabelle 7.2: Arbeitszeit über die Arbeitstypen nach Entwickler

7.4 Ausblick

7.4.1 Weiterführende Schritte

Server aufsetzen
Zurzeit läuft das ALF-Tool nur auf einem temporären Server (siehe Unterabschnitt 4.3.6. Server). Ein erster Schritt wäre es, das Tool auf einen Server zu migrieren, welcher permanent laufen kann. Eine mögliche Vorgehensweise findet sich im *readme.md* auf dem Projektarchiv des ALF-Sourcecode.

Tests
Durch die knappe Zeit während der Studienarbeit wurden nur Unit-Tests und Integrations-Tests durchgeführt, sowie unstrukturierte System-Tests durch die Entwickler. Um eine volle Funktionalität zu bescheinigen, müssten noch strukturierte System-, Usability- und Performance-Tests durchgeführt werden.

7.4.2 Weiterentwicklung

Mehr Projettypen
Das ALF-Tool unterstützt im aktuellen Stand nur CMake mit C++ und Cute. Eine mögliche Weiterentwicklung wäre zum Beispiel, dass weitere Projekttypen zusätzlich programmiert werden. Da an der OST zu einem grossen Teil mit Java unterrichtet wird, würde sich diese Sprache anbieten. Um dies umzusetzen, muss ein neuer Runner unter dem *project_runner* geschrieben werden (siehe Abschnitt 4.3.4. project_runner). Anschliessend muss noch der Datenbankeintrag für den neuen Projekttyp erstellt werden, was nur von einem Administrator des Systems durchgeführt werden kann (siehe Abschnitt B.3. Zusätzliche Projekttypen). Zum Schluss benötigt der neue Projekttyp noch einen Ordner in der Hierarchie (siehe Abbildung 4.7. Ordnerstruktur: /alf/alf/) und entsprechende Template-Dateien.

Studentensicht - Viele Module und Projekte
Möchten Studenten eine neue Abgabe tätigen, müssen sie zuvor ein Modul, sowie ein Projekt auswählen (siehe Unterabschnitt C.3.1. Hochladen einer Lösung). Da zurzeit keine Möglichkeit besteht, den Studenten spezifische Module oder Projekte zuzuweisen, können diese beiden Listen sehr lange werden. Diese Zuweisung würde mit den jetzigen Möglichkeiten einen zu grossen administrativen Aufwand bedeuten. Mit einer Anbindung an Moodle könnte diese Filterung vermutlich durchgeführt werden.

Studentensicht - Viele Studenten
Beim Ausfüllen des Abgabeformulars müssen die Studenten die Teamkollegen auswählen (siehe Unterabschnitt C.3.1. Hochladen einer Lösung). Diese Liste wird lediglich durch das Entfernen von Dozenten gefiltert, wodurch diese Liste rasch sehr lange werden kann. Mit einer Anbindung an Moodle könnte diese Filterung optimiert werden.

Dozentenansicht - Testen eines Projekts
Führen Dozenten einen Test eines Projekts durch, wird die aktuelle Verbindung aufrecht gehalten, da das Resultat nicht in der Datenbank und nicht auf dem Dateisystem gespeichert werden soll (siehe Unterabschnitt C.4.5. Projekt testen). Durch die Aufrechterhaltung der Verbindung kann es aber zu Überschreiten des Time-Outs des Browsers kommen. Dauert eine Auswertung eines Tests länger als das Browser-Time-Out, dann wird vom Browser die Verbindung zurückgesetzt und die Auswertung kann von den Dozenten nicht mehr eingesehen werden. Eine mögliche Behebung dieses Punktes könnte durch das Implementieren des gleichen Verhaltens, wie bei einer normalen Abgabe durch einen Studenten, erreicht werden. Dadurch hätte der Dozent aber seine Tests ebenfalls auf der Abgabeübersicht und im Download der Abgaben. Dies könnte durch eine Möglichkeit zum Löschen von Abgaben durch den Dozenten behoben werden.

Dozentenansicht - Dateienüberfluss beim Download
Bei einem Download aller Abgaben durch die Dozenten wird der komplette Inhalt des *submissions* Ordners in das Archiv gepackt (siehe Unterabschnitt C.4.2. Abgaben einsehen). Somit finden die Dozenten nicht nur die letzten, sondern alle Abgaben in dem Archiv. Dies könnte behoben werden, wenn der Archiv-Funktion nicht einfach *submissions*, sondern eine Liste spezifischer Ordner übergeben würde.

Statistiken
Eine Erweiterung der Anzeige anhand von Statistiken könnte den Dozenten hilfreiche Informationen aufzeigen. *War die Aufgabe zu schwer? / Wie viele Versuche (Abgaben) hat eine Gruppe im Schnitt benötigt? /...* Somit könnte er in der nächsten Durchführung des Moduls die Vorlesung oder Aufgabe anpassen.

Volles responsives Design
Im Rahmen dieser Studienarbeit wurde die Priorität primär auf das Backend gelegt. Daher ist das Frontend praktischer Natur. Das Frontend sieht zwar durch Bootstrap in allen Fenstergrössen brauchbar aus, jedoch wurde das Frontend weder für die verschiedenen Grössen optimiert, noch wurde es strukturiert getestet.

Zurücksetzen eines Projekts
Im aktuellen Feature-Umfang ist es für die Dozenten nicht möglich das Projekt für die nächste Iteration des Moduls zurückzusetzen. Momentan müssen die Dozenten das Projekt löschen und neu anlegen oder in Kauf nehmen, dass weiterhin die Abgaben der ehemaligen Studenten zu sehen sind. Mit der Möglichkeit ein Projekt zurückzusetzen, würden alle Abgaben ausgeblendet oder gelöscht werden. Alternativ würde die Funktionalität des Kopierens eines Projekts dieses Problem auch lösen.

Kapitel 8

Literaturverzeichnis

[1] Thomas Corbat, *Guideline Dokumentation*.

[2] OST, *Leitfaden für Bachelor- und Studienarbeiten*.

[3] OST, *Informationen zur SA-Abgabe*.

[4] OST, *Studienarbeiten: Termine Herbssemester 2021/22*.

[5] Python Software Foundation, *Python 3.10.0 Documentation*, 2021. Adresse: https://docs.python.org/3.10/ (besucht am 28. 11. 2021).

[6] Holger Krekel und Pytest-Dev Team, *Pytest 6.2.x Documentation*, 2021. Adresse: https://docs.pytest.org/en/6.2.x/contents.html (besucht am 13. 12. 2021).

[7] Django Software Foundation und einzelne Mitwirkende, *Django 4.0 Documentation*, 2021. Adresse: https://docs.djangoproject.com/en/4.0/ (besucht am 13. 12. 2021).

[8] Python Software Foundation, *PEP 8 – Style Guide for Python Code*, 2021. Adresse: https://www.python.org/dev/peps/pep-0008/ (besucht am 08. 10. 2021).

Kapitel 9

Glossar

Das Glossar ist auf der Stufe eines Informatikstudenten oder -dozenten der OST geschrieben, daher werden gewisse Nomenklaturen bereits vorausgesetzt. Die Texte in den Glossareinträgen stammen von den einzelnen Webseiten und sind ins Deutsche übersetzt oder leicht angepasst.

9.1 Glossar

ALF-Dokumentation
Dies ist ein GitLab-OST-Projektarchiv, welches die gesamte Dokumentation beinhaltet.
URL: https://gitlab.ost.ch/roman.spring/sa_alf_docu/

ALF-Pages
Eine einfache Übersicht über die Test-Coverage über alle Packages des Projekts. Diese Pages wurden mit GitLab-Pages erstellt.
URL: http://roman.spring.pages.gitlab.ost.ch/sa_alf/

ALF-Sourcecode
Dies ist ein GitLab-OST-Projektarchiv, welches den gesamten Sourcecode aller Packages beinhaltet.
URL: https://gitlab.ost.ch/roman.spring/sa_alf/

C++
C++ ist eine von der ISO genormte Programmiersprache. C++ ermöglicht sowohl die effiziente und maschinennahe Programmierung als auch eine Programmierung auf hohem Abstraktionsniveau. Der Standard definiert auch eine Standardbibliothek, zu der verschiedene Implementierungen existieren.
URL: https://isocpp.org/

Clean Code Regel
Eine Initiative für mehr Professionalität in der Softwareentwicklung.
URL: https://clean-code-developer.com/

CMake
CMake ist eine quelloffene, plattformübergreifende Familie von Tools zum Erstellen, Testen und Verpacken von Software. CMake wird verwendet, um den Softwarekompilierungsprozess mit einfachen plattform- und compilerunabhängigen Konfigurationsdateien zu steuern und native Makefiles und Workspaces zu erzeugen, die in unterschiedlichen Compilerumgebungen verwendet werden können.
URL: https://cmake.org/

Cute
Das Cute-Framework ist eines der Besten C++ Unit Testing Frameworks. Das Cute-Framework ist nicht nur sehr einfach zu verwenden, sondern nutzt auch moderne C++-Bibliotheken und -Funktionen.
URL: https://cute-test.com/

Django
Django ist ein High-Level-Python-Web-Framework, das eine schnelle Entwicklung und ein sauberes, pragmatisches Design fördert. Es wurde von erfahrenen Entwicklern entwickelt und nimmt einen Grossteil der mühsamen Webentwicklung ab, sodass man sich auf die Entwicklung der Anwendung konzentrieren kann, ohne das Rad neu erfinden zu müssen.
URL: https://djangoproject.com/

Docker Hub
Docker Hub ist der weltweit einfachste Weg, Container-Anwendungen zu erstellen, zu verwalten und bereitzustellen.
URL: https://hub.docker.com/

GitLab-OST
GitLab-OST ist eine Instanz von GitLab, welche durch die OST gehostet wird.
URL: https://gitlab.ost.ch/

GitLab-Pages
Mit GitLab-Pages können statische Websites direkt aus einem Projektarchiv in GitLab veröffentlicht werden. GitLab-Pages ist für jede private oder geschäftliche Website geeignet und kann einen beliebigen Static Site Generator (SSG) oder einfaches HTML verwenden.
URL: https://docs.gitlab.com/ee/user/project/pages/

JetBrains
JetBrains ist ein multinationales Software-Unternehmen mit Niederlassungen in Prag, Sankt Petersburg, Novosibirsk, Moskau, Boston und München. JetBrains wurde im Jahr 2000 von den russischen Softwareentwicklern Sergey Dmitriev, Eugene Belyaev und Valentin Kipiatkov in Prag gegründet.
URL: https://jetbrains.com/

OAuth 2.0
OAuth 2.0 ist das branchenübliche Protokoll für die Autorisierung. OAuth 2.0 konzentriert sich auf die Einfachheit der Client-Entwickler und bietet gleichzeitig spezifische Autorisierungsabläufe für Webanwendungen, Desktop-Anwendungen, Mobiltelefone und Wohnzimmergeräte. Diese Spezifikation und ihre Erweiterungen werden von der IETF OAuth Working Group entwickelt.
URL: https://oauth.net/2/

OpenFaaS
OpenFaaS macht es Entwicklern leicht, ereignisgesteuerte Funktionen und Microservices in Kubernetes bereitzustellen, ohne dass sie sich wiederholende Kodierungen vornehmen müssen. Verpacken Sie Ihren Code oder eine vorhandene Binärdatei in ein Docker-Image, um einen hoch skalierbaren Endpunkt mit automatischer Skalierung und Metriken zu erhalten.
URL: https://openfaas.com/

OST – Virtual Server Kiosk
Ein Kiosk für virtuelle Server der OST.
URL: https://vkiosk.i.ost.ch/

pytest
pytest ist ein ausgereiftes, voll funktionsfähiges Python-Testwerkzeug, das hilft, bessere Programme zu schreiben. Das pytest-Framework macht es einfach, kleine Tests zu schreiben, und ist dennoch skalierbar, um komplexe funktionale Tests für Anwendungen und Bibliotheken zu unterstützen.
URL: https://docs.pytest.org/

Python-Wheel
Wheels sind der neue Standard der Python-Distribution und sollen eggs ersetzen. Unterstützung wird in pip $\geq$ 1.4 und setuptools $\geq$ 0.8 angeboten.
URL: https://pythonwheels.com/

Traefik
Traefik ist ein Open-Source-Edge-Router, der die Veröffentlichung von Diensten zu einer einfachen und unterhaltsamen Erfahrung macht. Er empfängt Anfragen im Namen des Systems und findet heraus, welche Komponenten für deren Bearbeitung zuständig sind.
URL: https://doc.traefik.io/traefik/

Vue

Vue ist ein progressives Framework zur Erstellung von Benutzeroberflächen. Im Gegensatz zu anderen monolithischen Frameworks ist Vue von Grund auf so konzipiert, dass es schrittweise übernommen werden kann. Die Kernbibliothek konzentriert sich nur auf die View-Schicht und ist leicht zu übernehmen und in andere Bibliotheken oder bestehende Projekte zu integrieren. Auf der anderen Seite ist Vue in Kombination mit modernen Werkzeugen und unterstützenden Bibliotheken durchaus in der Lage, anspruchsvolle Single-Page-Anwendungen zu realisieren.
URL: https://vuejs.org/

YouTrack

YouTrack ist ein Projektmanagement-Tool von JetBrains, das an Prozesse angepasst werden kann und hilft, hervorragende Produkte zu liefern. Projekte und Aufgaben können im Blick behalten werden.
URL: https://www.jetbrains.com/de-de/youtrack/

9.2 Abkürzungen

ALF
Automated Lesson-Feedback

CI
Continuous Integration

FA
Funktionelle Anforderungen

MS
Meilenstein

NFA
Nicht funktionelle Anforderungen

OST
Ostschweizerische Fachhochschule

SW
Semesterwoche

Kapitel A

Persönliche Berichte

Die persönlichen Berichte zeigen die Erfahrungen und Schlussfolgerungen der einzelnen Entwickler auf. Da diese Berichte persönlich sind und die Meinung eines Entwicklers widerspiegeln, sind sie in der Ich-Perspektive (bzw. Wir-Perspektive) verfasst.

A.1 Pascal Gsell

A.1.1 Start des Projekts

Bereits vor dem Semesterstart konnte ein Kickoff-Meeting durchgeführt werden, wodurch wir rasch Zugang zu allem Nötigen bekommen hatten. Unter anderem wurde uns das GitLab-OST-Projekt vom bestehenden ALF-Tool freigegeben. Zu Beginn schien mir alles fremd und ohne grosse Logik. Die Übersicht über die bestehende Lösung fiel mir schwer. Dies relativierte sich nach weiterer Einarbeitungszeit und nach einer Einführung von Nicola. Die erste Woche verging wie im Flug und ich, genau so wie Roman, hatte noch keine Ahnung wie die Architektur, die Schnittstellen und Docker-Container aussehen, geschweige denn programmiert werden sollten. Mit was sollten wir beginnen, was sollte früh geklärt werden und sollten wir top-down oder bottom-up arbeiten. All diese Unklarheiten waren eine grosse Herausforderung, gerade zu Beginn des Projekts. Nicht nur da wir beide noch nicht mit Docker oder Docker-Compose gearbeitet hatten, sondern auch weil wir zu Beginn vieles klären mussten, bevor wir überhaupt einzelne Funktionen ausprogrammieren konnten.

A.1.2 Erst mal keinen Code

Wie bereits erwähnt, hatten wir beim Start des Projekts Mühe den Überblick über bestehendes zu finden. Eine weitere Herausforderung war, dass wir fast frei in den Anforderungen waren. Somit mussten wir nicht nur das bestehende Tool kennenlernen, sondern wir mussten uns auch Gedanken machen, was das Produkt am Ende können soll. Wir haben über einige Features mehrmals diskutiert, ist es nun SOLL oder KANN, haben wir etwas vergessen und kann das überhaupt zusammen ein gutes Produkt ergeben. Nachdem sich die Anforderungen herauskristallisiert hatten, mussten wir uns noch einigen, ob wir das Projekt nun top-down oder bottom-up bestreiten. Ich hätte gerne bottom-up gearbeitet, ich denke Roman Spring ebenso, aber Thomas Corbat und Nicola Jordan empfahlen uns top-down. Top-Down hat den klaren Vorteil, dass bereits früh Technologien verifiziert werden können und bereits vieles an Schnittstellen und den wichtigsten Funktionen mindestens musterhaft implementiert werden müssen. Für uns hiess dies aber, dass wir uns zuerst mit Python, Django, Docker-Compose, einem Proxy und dem Server anfreunden mussten. Das bedeutete viel Recherche für die neuen, uns noch nicht sehr geläufigen, Technologien und noch keinen Code.

A.1.3 Alles neu

Bereits im Engineering-Projekt mussten wir uns mit GitLab-OST, der GitLab-OST-CI und der Zeiterfassung in GitLab-OST auseinandersetzen. Roman Spring und mir war klar, dass wir GitLab-OST nur für das nötigste einsetzen werden, folglich mit dem ALF-Sourcecode-Projektarchiv und der GitLab-OST-CI. Die GitLab-OST-CI wurde uns einige Male zum Verhängnis und kostete uns viel Einrichtungszeit.
Weiter mussten wir ein neues Tool für die Zeiterfassung finden. Roman Spring hatte bereits etwas Erfahrung mit YouTrack, ein Produkt von JetBrains und somit für uns kostenlos. Mir erschien das Tool von Beginn an sehr angenehm, daher mussten wir nicht lange diskutieren.

Docker-Compose und Traefik, sowie Nginx, durften wir kurz in *Distributed Systems & Blockchain* kennen lernen. Wir konnten aber in diesem Modul die Technologien nicht in die Praxis umsetzen, sodass wir uns auch in diesen Technologien noch Einlesen mussten. Uns fehlte dabei einiges Wissen über konkrete Umsetzungen und wir stiessen dabei auf einige Probleme, was die richtige Konfiguration in einer produktiven Umgebung betrifft.
Python war mir nicht ganz fremd, da ich im Vorsemester das Modul *Python* abschliessen konnte. Grundlegendes Wissen von Packages, Modulen und Klassen oder Funktionen konnte ich mitbringen, dies war aber auch schon alles, was ich vom *Python*-Modul in dieses Projekt einbringen konnte. Da wir Django als Framework nutzten und ich die ersten Implementationen in diesem Framework machen durfte, musste ich mich bereits vor meiner ersten Zeile Code in Django und Python einlesen.

A.1.4 Neues Gelernt

Da so ziemlich alles in diesem Projekt für mich neu war, konnte ich in allen Bereichen Neues erlernen.

Projektstruktur
Wir haben uns darauf geeinigt nur zwei Projektarchive, ALF-Dokumentation und ALF-Sourcecode, einzurichten. Die ALF-Dokumentation von dem ALF-Sourcecode zu trennen, war sicherlich eine gute Idee und werde ich in Zukunft gleich umsetzen. Allerdings war es im Nachhinein keine gute Idee, nur ein Projektarchiv für den ganzen ALF-Sourcecode zu nutzen. Bereits nach nur wenigen Wochen pflegten wir einige Branches, welche unterschiedliche Ansprüche hatten. Diese Ansprüche in einem Projektarchiv zu vereinen, war stets eine Herausforderung. In einem nächsten Projekt würde ich das ALF-Sourcecode-Archiv ebenfalls sinnvoll auftrennen, um es leichtgewichtiger zu machen.

GitLab-OST-CI
Die GitLab-OST-CI hat definitiv zu viel Zeit in Anspruch genommen. Die bereits erwähnte Projektstruktur hat uns dabei jedes Mal vor Herausforderungen gestellt. Nicht nur, dass jeder Branch unterschiedlich gehandhabt werden musste, sondern auch dass eigene Packages in Branches verwendet wurden, womit eine Abhängigkeit untereinander entstand und dies in nur einer Konfiguration. Über den Verlauf des Projekts konnten wir aber eine laufende GitLab-OST-CI einrichten. Die nun vorliegende Implementation verwendet wiederverwendbare Skripts, Requirements-Installationen, Tests mit pytest, Deployments über Docker, Coverage-Reports in Artefakten und publizieren der Coverage-Reports auf den ALF-Pages. Für mich dient diese Implementation als Vorlage für zukünftige Implementationen, wobei ich aber definitiv ein Augenmerk auf mögliche Trennungen des Projektarchivs legen werde. Trotz vielen gewonnen Einblicken in die GitLab-OST-CI, kenne ich nur das Nötigste über die Möglichkeiten und werde vermutlich auch in Zukunft einige Zeit in dieses Tool verschwenden.

Docker-Compose und Traefik
In diesem Projekt konnten wir unsere erste in Produktion laufende Docker-Compose Umgebung inklusive Traefik umsetzen. Ich konnte vieles über Docker-Compose und Traefik, sowie die Einschränkungen von diesen und Nginx, lernen. Eine grosse Unsicherheit war zu Beginn die Kommunikation über Container hinweg, welche sich aber als nicht kompliziert herausstellte und einfach umgesetzt werden konnte. Aber auch in diesem Bereich würde ich behaupten, dass ich nur gerade das Nötigste kenne, um Docker-Compose und Traefik einzusetzen und ich in weiteren Projekten neues erlernen werde.

Django und Python
Da Python sich gegenüber den bereits bekannten Programmiersprachen recht unterscheidet, konnte ich an die Programmierung anders heran gehen. Nach den doch einigen geschrieben Zeilen ALF-Sourcecode gefällt mir die Programmiersprache sehr gut und die pythonische Art Code zu schreiben fällt mir nicht mehr schwer. Manchmal ganz im Missfallen von Roman Spring, da Ihm diese Art nicht so zusagt wie mir und sie manchmal schon mit wenigen Zeichen auskommt.
In Django konnten wir die Authentifizierung über GitLab-OST, Modelle für die Datenbank und Formulare umsetzen. Hinzu kam die eigene Implementation eines Validators, welcher für Formularfelder verwendet wird. Auch hatten wir zu Beginn die *views* als Funktionen geschrieben und sie im Verlauf des Projekts zu Klassen umgeschrieben, um duplizierten Code zu mindern.

Datenbank und Schnittstellen
Bereits zu Beginn mussten wir für den Technologiedurchstich die Datenbank und Schnittstelle zwischen Unit-Coordinator und Job-Handle in einer ersten Fassung definieren und implementieren. Bereits bis zum zweiten Meilenstein mussten wir die Datenbank zehn mal anpassen, auch das Package *alf_socket*, welches die Schnittstelle zwischen Unit-Coordinator und Job-Handle implementiert, erreichte eine hohe Version von 3.0.2. Wir haben während dem ganzen Projekt nichts so häufig geändert wie die Datenbank und Schnittstellen. In der ersten Fassung war die Datenbank viel zu umfangreich und unnötig komplex aufgebaut. Zum Ende hin wurde sie schlank und enthält jetzt nur noch das Wesentlichste, um den aktuellen Funktionsumfang des Produkts zu unterstützen. In einem nächsten Projekt würde ich mehr Zeit in die Analyse der Datenbank und Schnittstellen investieren und erhoffe mir daraus, dass im Verlaufe des Projekts nicht so viele Änderungen gemacht werden müssen.

A.1.5 Abschluss

Wir meldeten uns auf dieses Projekt, da wir beide das frühere ALF-Tool kannten und dort bereits Testate vom Modul *Cpp* getestet haben. Wir fanden beide das Tool sehr hilfreich und haben uns gefreut es weiter zu entwickeln. Schnell wurde klar, dass wir das Tool nicht direkt weiter entwickeln sollen, sondern auf einen Stand bringen, welcher dann auch weiter verwendet und für andere Module eingesetzt werden kann. Wir sahen das Potential dieses Tool für andere Module zu verwenden und freuen uns bei diesem ersten Schritt in diese Richtung mitgewirkt zu haben. Trotz allen Schwierigkeiten am Anfang mit neuen Technologien und dem geplanten Technologiedurchstich bereits im ersten Meilenstein konnten wir das Projekt erfolgreich abschliessen.
Ich habe in allen Bereichen vieles mitgenommen und einiges über GitLab-OST, Docker-Compose, Django und natürlich Python gelernt. Aber neben allen Erfahrungen mit der Programmiersprache Python habe ich auch einiges über Projektstruktur, Schnittstellen und Datenbanken gelernt und gefestigt. Mir hat die Arbeit an diesem Projekt sehr gefallen und die Zusammenarbeit mit Roman Spring war immer sehr angenehm. Wir konnten immer ausführlich, sachlich und freundlich miteinander diskutieren und fanden stets einen gemeinsamen Nenner.

A.2 Roman Spring

A.2.1 Aller Anfang ist schwer...

Zu Beginn der Studienarbeit waren wir in Bezug auf das Projekt recht aufgeschmissen. Zwar war das Hauptziel der Arbeit klar, dennoch gab es mehrere Punkte, welche bei uns Kopfzerbrechen verursachten.

Keine saubere Aufgabenstellung
Da wir keine sauber aufgeführte Aufgabenstellung hatten, mussten wir zu Beginn erst einmal evaluieren, was überhaupt gewünscht ist. Bis wir komplett durchgeblickt hatten, was überhaupt das genaue Ziel der Arbeit ist, vergingen die ersten paar Wochen wie im Flug. Frei von einer Aufgabenstellung zu sein hat seine Vorteile, denn so konnten wir die Wünsche und Anforderungen an das Projekt selbst ein wenig steuern. Dennoch denke ich, dass die ersten paar Wochen um einiges produktiver hätten eingesetzt werden können, wenn eine Aufgabenstellung vorhanden gewesen wäre. Wir haben auch während des ganzen Projekts untereinander immer wieder Diskussionen geführt, ob wir nun dieses oder jenes Feature implementieren sollten.
Während der Implementation erweiterten wir den Funktionsumfang des Tools stetig, bis wir irgendwann gemerkt haben, dass die Zeit langsam knapper wird und wir uns auf die primären Anforderungen konzentrieren sollten.
Im Nachhinein muss ich sagen, dass ich eine gut definierte Aufgabenstellung vorgezogen hätte, da wir die selbst gesetzten Anforderungen unterschätzt haben. Dies führte dazu, dass wir zum Ende hin die geforderte Zeit überschritten haben, um das Projekt sauber abzuschliessen.

Wo beginnt man?
Am Anfang des Projekts waren wir ziemlich planlos, wie und wo wir starten sollten. Dies zum einen, weil wir keine definierten Anforderungen hatten, zum anderen, weil wir mit Softwareprojekten noch nicht viele Erfahrungen hatten. Ein Hinweis von Nicola Jordan, dass wir mit einem vertikalen Durchstich beginnen sollten, half uns schlussendlich einen Startpunkt zu finden. Durch diesen Durchstich bekamen wir schnell einen Eindruck, was alles erledigt werden musste. Zudem zeigte der Durchstich, dass unsere Architektur-Idee auch funktionierte.
Bei der weiteren Entwicklung konnten wir diesen Durchstich anschliessend horizontal erweitern, wodurch nach und nach das ALF-Tool entstand. Die Methode des vertikalen Durchstichs finde ich interessant, vor allem für die Entwicklung eines Prototyps. Allerdings finde ich sie bei der Entwicklung eines endgültigen Produkts heikel, da durch diese Methode die Kopplung der einzelnen Komponenten erhöht wird. Durch den vertikalen Durchstich sind die Komponenten nicht so sauber getrennt, wie wenn jede Komponente einzeln für sich entwickelt worden wäre. Dies wäre jedoch für eine Weiterentwicklung förderlich. Daher werde ich die vertikale Entwicklung in Zukunft für Prototypen oder Machbarkeitsstudien weiterverwenden, jedoch bei der Entwicklung wahrscheinlich auf eine horizontale Entwicklung setzen.

A.2.2 I have never ever...

Innerhalb dieses Projekts habe ich (beziehungsweise wir) viele neue Technologien verwendet, welche ich zuvor noch nie angewendet hatte. Dadurch musste ich viel recherchieren, bevor ich mit der eigentlichen Arbeit beginnen konnte. Dies führte auch dazu, dass das Projekt nicht so schnell voranschritt wie ich es erwartete. Bei der Planung dachten wir, dass wir genügend Reservezeit eingeplant hätten. Jedoch erreichten wir jeden Meilenstein immer nur knapp. Dies nicht nur wegen den Recherchen, sondern auch weil sich durch das bessere Verständnis bessere Möglichkeiten anboten, welche anschliessend implementiert wurden.
Im nachfolgenden gehe ich auf die neuen Technologien ein, welche ich mir im Laufe das Projekts aneignen musste.

Django und Python
Ich habe für dieses Projekt das erste Mal mit Python gearbeitet. Somit musste ich eine komplette Programmiersprache von Grund auf erlernen, welche sich doch recht von den gewohnten Sprachen wie Java oder C# unterscheidet. Sobald klar war, welches Projekt wir innerhalb der Studienarbeit umsetzen, habe ich privat ein Miniprojekt entwickelt, um die Sprache kennen zu lernen. Im Nachhinein muss ich sagen, dass mir dies nur bezüglich des Syntax was gebracht hat. Im Laufe des Projekts musste ich feststellen, dass Python Klassen nur verwendet werden, wenn diese sinnvoll sind. Im Allgemeinen musste ich lernen, dass Python die Tendenz hat, nur das zu machen was auch wirklich benötigt wird. Unterdessen würde ich sagen, dass ich langsam den Grundgedanken von Python verstehe, obwohl ich immer wieder versucht bin die OO-Welt reinzubringen.
Pascal Gsell hat sich zu Beginn primär mit dem Django-Framework auseinandergesetzt. Nach dem der Job-Handle erledigt war, konnte ich durch die Zusammenarbeit innerhalb des Unit-Coordinators auch viel über das Django-Framework lernen.

Docker, Docker-Compose und Traefik
Im Modul *Distributed Systems & Blockchain* hatten wir zwar Docker und Docker-Compose angeschaut, leider jedoch nur im schnellen Überflug. In diesem Überflug wurde zwar ein einfacher Container erstellt, allerdings half dies nicht für die tatsächliche Implementierung. Dies gilt nicht nur für Container, sondern auch für Docker-Compose und Traefik, welches nur theoretisch durchgenommen wurde.
Durch das Recherchieren und Experimentieren ist das Verständnis von Docker und Docker-Compose bei uns enorm angestiegen. Durch diese Arbeit haben wir nun auch ein Template für weitere Projekte. Ein weiteres Problem war die Umstellung von http zu https. Die Theorie dahinter hatten wir in mehreren Modulen angeschaut. Jedoch keines hat mir beigebracht, wie ich dies umzusetzen habe. Diese Umsetzung hat ebenfalls mehr Zeit in Anspruch genommen, als sie nach dieser Ausbildung hätte sein dürfen.

CI
Wie auch Docker wurde die GitLab-OST-CI in vielen Modulen angesprochen, jedoch in keinem Modul wirklich erklärt und gelernt. Da ich mich im Engineering Projekt nicht für die GitLab-OST-CI interessiert habe, begann ich in dieser Studienarbeit nur mit dem Wissen, wie eine YML-Datei aufgebaut ist. Das Recherchieren ging schnell von der Hand. Das Experimentieren hingegen hat enorm viel Zeit benötigt, da jedes Mal ein Runner von GitLab-OST gestartet und laufen gelassen werden musste. Dieser Umstand hat Zeit am laufenden Band vernichtet.
Zum aktuellen Zeitpunkt würde ich sagen, dass ich das Nötigste in der GitLab-OST-CI umsetzen kann. Verglichen mit dem, was ich gesehen habe und was möglich wäre, ist mein Wissen jedoch nicht existent. Es genügt jedoch um Projekte zu kompilieren, zu testen und zu veröffentlichen.

YouTrack
Da wir schlechte Erfahrungen innerhalb des Engineering Projekts mit der Zeiterfassung über GitLab-OST gemacht haben, war von Anfang an klar, dass wir dies nicht mehr machen. Nach einer kurzen Recherche bezüglich Zeiterfassung, haben wir uns für YouTrack entschieden.
Diese Entscheidung war kein Fehler. Im Nachhinein hätte man sie jedoch besser treffen können. Zum einen ist dieses Tool extrem mächtig und kann daher mehr als jemals irgendeine Person benötigt. Dieser enorme Funktionsumfang benötigte wiederum viel Zeit, um sich einzuarbeiten und sie zu überblicken. Zum anderen ist dieses Tool auf agile Projektarbeit ausgelegt und unsere Studienarbeit war eher semi-agil, da mit dem Erreichen des MVPs auch das Projekt endet. Da wir eher mit dem Kanban-Prinzip gearbeitet haben, wäre für die Entwicklung ein Kanbanboard mit Zeiterfassung besser geeignet gewesen.

A.2.3 Wenn ich du wäre ...

Während der Studienarbeit haben wir Entscheidungen getroffen, welche ich in einem zukünftigen Projekt wahrscheinlich anders treffen werde.

Ein Projektarchiv für den Sourcecode
Wir haben uns entschieden, dass wir den ganzen ALF-Sourcecode in einem Projektarchiv von GitLab-OST verwalten. Dies ist auf der einen Seite angenehm, da das ganze Projekt zur Verfügung steht. Spätestens jedoch beim Mergen zweier Branches, welche unterschiedliche Versionen der Teilprojekte besitzen, gibt es einen enormen Mehraufwand. Nicht nur beim Mergen selbst, sondern auch bei relevanten Fehlerbehebungen in den Branches. Zwei Möglichkeiten bestehen, um relevante Fehlerbehebungen in mehreren Branches durchzuführen. Zum einen kann im entsprechenden Branch der Fehler korrigiert und anschliessend diesen Branch in alle anderen Branches zusammengeführt werden. Zum anderen kann der Fehler in allen Branches einzeln korrigiert werden. Beide Möglichkeiten sind zeitaufwändig und nicht sauber.
In einem zukünftigen Projekt werde ich mich auf jeden Fall dafür entscheiden, für jedes Teilprojekt ein eigenes Projektarchiv zu erstellen (sofern sinnvoll). Mit der Hoffnung, dass die einzelnen Projektarchive viel Nerven und Zeit einsparen werden.

Mehrzeit in verbindende Elemente investieren
In diesem Projekt haben wir nichts so viel geändert wie die Schnittstelle zwischen dem Unit-Coordinator und Job-Handle, vielleicht noch das Datenbankmodell. Einen Teil der Änderungen an der Schnittstelle hätte verhindert werden können, wenn wir mehr Zeit in das Ausarbeiten dieser investiert hätten. Für das Erarbeiten einer Datenbank für eine Software fehlte mir die Erfahrung, um zu erkennen, dass das ursprünglich geplante Datenbankmodell nicht funktionieren kann, beziehungsweise dass es zu gross war. Auch bezüglich Datenbankmodell hätte eine intensivere Analyse des Problems wahrscheinlich geholfen, die Änderungen zu minimieren. Durch die Anpassungen der Software an das neue Interface oder an das neue Datenbankmodell ging viel Zeit verloren, welche in zusätzliche Features oder in eine noch vollständigere Testabdeckung hätte investiert werden können.

Zeit- und Task-Trackingtool
Wie bereits erwähnt war die Wahl für die Zeiterfassung und das Task-Tracking, YouTrack zu verwenden, kein Reinfall. Dennoch würde ich bei einem nächsten Projekt ein leichteres Tool verwenden. Auch haben wir alles (jeden Meilenstein und die gesamte ALF-Dokumentation) in einem einzigen Board verwaltet. Dieses wurde dadurch schnell unübersichtlich. Bei einem nächsten Projekt werde ich zumindest die Erfassung für die ALF-Dokumentation auf einem eigenen Board verwalten.
Uns stellte sich während des Projekts auch immer wieder die Frage, wie fein granular sollten die einzelnen Tasks sein. Wenn sie zu fein sind, steigt der Erfassungsaufwand der Tasks und Arbeitszeit enorm an. Sind sie zu grob, bringen sie keinen Mehrwert mehr. Gestartet haben wir mit fein granularen Tasks. Durch Änderungen während dem Projekt, kam es immer wieder dazu, dass diese Tasks dann gar nicht mehr aktuell waren oder sogar überflüssig wurden. Im Laufe des Projekts haben wir immer mehr auf eher grob granulare Tasks gesetzt (à la ein Task für ein Python-Modul in einem Meilenstein). Diese Aufteilung hat gut funktioniert und ich werde dies so voraussichtlich für die nächste Zeiterfassung auch beibehalten, sofern keine anderen Anforderungen bestehen.

Dokumentation
Innerhalb dieser Studienarbeit mussten wir diverse Kapitel, beziehungsweise Texte schreiben, welche unserer Meinung nach, keinen Mehrwert bringen. Da diese zwar für die Entwicklung eines Projekts wichtig sind, aber für den Leser der Dokumentation nur mehr Text zum Lesen darstellen.
Das Problem findet sich wahrscheinlich in der unterschiedlichen Meinung der Art der Dokumentation. Eine Produktdokumentation würde unserer Meinung nach ausreichen, hingegen wünschten unsere Betreuer eine Projektdokumentation. Was nicht bedeutet, dass die Projektentscheidungen und -definitionen nicht getroffen und definiert werden sollten.
Die Frage, welche ich mir stelle, ist; "Wäre es in einem nächsten Projekt nicht besser zwei Dokumentationen zu schreiben?". Eine Dokumentation über das Projekt selbst, und die andere nur über das Ergebnis.

A.2.4 Schlusswort

Mein Ziel für die Studienarbeit war es, etwas zu erarbeiten, was einen Nutzen erbringt und keine reine Beschäftigungstherapie ist. Da wir aus dem Modul *Cpp* das ALF-Tool bereits kannten und auch bereits mit diesem gearbeitet haben, freuten wir uns dieses weiterzuentwickeln. Schon im Kickoff-Meeting erfuhren wir, dass der aktuelle Stand des ALF-Tools eher einem Prototyp gleicht und wir die Wahl zwischen Weiterentwickeln oder Neuimplementieren haben. Die Entscheidung mit einer Neuimplementierung war die Richtige. Denn nun sollte das ALF-Tool bereit für eine Weiterentwicklung sein.
Ich bin froh, dass das Projekt ein erfolgreiches Ende gefunden hat. Denn auch wenn ich gerne am Projekt gearbeitet habe, war es dennoch anstrengend und intensiv. Leider konnten wir die Arbeit nicht in der gesetzten Arbeitszeit erledigen und wir benötigten einiges mehr an Zeit. Ich denke dies lag vor allem daran, dass wir unsere Anforderungen selbst gewählt haben und durch unsere mangelnde Erfahrung den Aufwand falsch eingeschätzt haben. Durch die harmonierende Zusammenarbeit mit Pascal Gsell konnten wir immer ausführlich und freundlich miteinander diskutieren und kamen stets auf einen gemeinsamen Nenner. Dies erlaubte uns, trotz des Mehraufwands, das Projekt termingerecht abzuschliessen.
Vom erarbeiteten Wissen und Erfahrung nehme ich insbesondere die Erfahrung mit Python und Docker mit. Auch die Erfahrung, dass man eine Schnittstelle oder auch ein Datenbankmodell wohl überlegen sollte und diese nicht einfach im Durchlaufen definieren sollte, nehme ich mit.
Am meisten an der ganzen Studienarbeit hat mich die Tatsache gestört, dass andere Studienarbeit-Teams nur eine Produktdokumentation schreiben mussten. Während andere Teams nicht einmal 30 Seiten erreichen, mussten wir so viele schreiben, dass wir über 150 Seiten erreichen. Dies sollte unbedingt in den Richtlinien der Studien- und Bachelorarbeiten genauer definiert werden, da es so wie es momentan ist, einen faden Beigeschmack hinterlässt.

Kapitel B

Entwickler Dokumentation

Die Entwickler Dokumentation richtet sich an die Administratoren und Entwickler des ALF-Tools. Es wird aufgeführt, wie und was man im Backend machen kann, wie der Server funktioniert und wie man einen neuen Projekttypen hinzufügt.

Die folgenden Bilder dienen zur Veranschaulichung und können vom aktuellen Stand des Tools leicht abweichen.

B.1 Backendadmins

Die Unterseiten des Backendadmins stellen das Backend des ALF-Tools dar. Sie sind erreichbar durch <Webseite>/backendadmins, zum Zeitpunkt der Entwicklung ist dies https://sinv-56026.rj.ost.ch/backendadmins. Im Backend können sich nur Administratoren oder Dozenten (für die Dozenten muss ein Passwort im Backend erfasst sein) anmelden. Für Dozenten nützt das Backend nicht viel, ausser dass sie sich für das ALF-Tool anmelden können. Ein Administrator hingegen kann dort die Datenbankeinträge einsehen, anpassen oder neue anlegen.

Im Backend stehen folgende wichtige Punkte zur Verfügung (irrelevante Punkte wurden hier weggelassen):

- Users
- Social applications
- Modules
- Projecttypes
- Projects
- Requiredfiles
- Submissions
- Results

Das Backend sollte nur im Notfall oder bei Punkten verwendet werden, wenn diese nicht über das Frontend erledigt werden können.

B.1.1 Dozenten registrieren

Voraussetzungen:

- Der gewünschte Benutzer hat sich bereits einmal über GitLab-OST angemeldet (siehe Abschnitt C.1. Anmelden).

Alle über GitLab-OST registrierten Benutzer sind standardmässig normale Studenten. Um einen Studenten zu einem Dozenten zu machen, müssen folgende Schritte befolgt werden.

1. Anmelden als Administrator im Backend
2. Zum Benutzer navigieren: *Users* → <Benutzer>
3. *Staff status* anwählen: **Achtung: nicht das Feld *Superuser status***
4. Abspeichern: unten rechts → *Save*

Die Superuser-Berechtigungen sind mit Vorsicht einzusetzen, da durch sie viele Möglichkeiten geboten werden, welche zu Fehlverhalten des ALF-Tools führen können.

B.1.2 Modul anlegen

Voraussetzungen:

- Jedes Modul, welches angelegt wird, benötigt mindestens einen Dozenten. Daher müssen immer zuerst die Dozenten registriert werden (siehe Unterabschnitt B.1.1. Dozenten registrieren).

Ein neues Modul kann über "*Modules* → *Add module*" angelegt werden. Während das Feld *Name* nur für die Datenbank ist, ist der *Name display* für das Frontend.

B.1.3 Projekttyp anlegen

Ein neuer Projekttyp kann über "*Projecttypes* → *Add projecttype*" angelegt werden. Wie beim Modul (siehe Unterabschnitt B.1.2. Modul anlegen) wird das Feld *Name* nicht im Frontend verwendet. Für den Namen auf dem Frontend wird das Feld *Name display* verwendet. Der Pfad für die Template-Dateien kann bei der Erstellung ignoriert werden, dieser wird automatisch erstellt, sobald der Projekttyp gespeichert wird. Der Pfad wird im Abschnitt B.3. Zusätzliche Projekttypen benötigt, sollte daher notiert oder gemerkt werden.

B.2 Server

B.2.1 Modi

Der Server besitzt zwei Modi, ein Dev- und Prod-Modus. Beide Modi verwenden getrennte Dateisysteme und Datenbanken.
Beim Starten des Dev-Modus wird auf dem Dateisystem der Ordner */alf/alf_dev/* gelöscht und vom Ordner */alf_backup/* kopiert, somit ist das Dateisystem beim Starten des Dev-Modus immer im selben Zustand. Auch wird beim Starten des Dev-Modus die verwendete Datenbank zurückgesetzt und mit Musterdaten befüllt, welche mit dem Dateisystem übereinstimmen.
Der Prod-Modus hingegen persistiert alle Einträge der Datenbank über mehrere Sitzungen hinweg und nutzt den Ordner */alf/alf/*. Im Prod-Modus werden die Container skaliert, die konfigurierten Skalierungen können unter Tabelle 4.2. Skalierung der Container nachgelesen werden.

Es sollte darauf geachtet werden, dass im Livebetrieb nie der Dev-Mudos läuft, da dort Benutzer mit erhöhten Rechten existieren, welche kein sicheres Passwort besitzen, sowie auch Debug-Informationen direkt im Frontend angezeigt werden.

B.2.2 Musterdaten

Die Musterdaten werden nur im Dev-Modus erstellt.

Benutzer
In der aktuellen Konfiguration werden folgende Benutzer angelegt:

Typ / Berechtigungen	Benutzername	Passwort
Dozent	Ueli	ostostost
Dozent	Sepp	ostostost
Dozent	Hans	ostostost
Student	Michel	ostostost
Student	Igor	ostostost
Student	Ruwen	ostostost

Tabelle B.1: Musterbenutzer

Module
Jeder der drei Dozenten besitzt sein eigenes Modul.

- Ueli → Cpp - Cpp
- Sepp → CppA - Cpp Advanced
- Hans → CppUA - Cpp Ultimate Advanced

Projekttyp
Es wird der Projekttyp für ein CMake-C++-Cute Projekt angelegt.

Projekt
Im Dev-Modus werden zwei Projekte angelegt. Zum einen ein *Template-Testat* und zum anderen ein Testat aus dem *Cpp* Modul, das *Calculator Testat*. Beide gehören zum Mustermodul *Cpp*. Das *Template Testat* kann zwar gestartet werden, es existieren jedoch keine Unit-Tests, daher ist eine Abgabe auf diesem Projekt nie bestanden.

Abgaben
Als Musterabgaben werden sechs erstellt, welche verschiedene Abgabetermine haben und von verschiedenen Kombinationen der Musterstudenten abgegeben wurden. Zudem sind die Resultate der einzelnen Abgaben unterschiedlich, um die verschiedenen Möglichkeiten aufzuzeigen.

B.2.3 Starten des Servers

1. Anmelden über Putty oder Webkonsole
2. Das Shell-Script *./start.sh* ausführen
3. Anweisungen befolgen

Über dasselbe Shell-Script kann auch das Docker-Compose heruntergefahren oder aufgeräumt werden.

```
root@sinv-56026:~# ./start.sh
Development, Production, Stop, DOWN or Clean? [d,p,s,down,c]
```

Abbildung B.1: Shell-Script für das Starten des Servers

B.2.4 Konfigurationsübersicht

Traefik ist für beide Modi gleich definiert, daher existiert dafür nur eine einzige Konfiguration. Das */alf/traefik.yml* beinhaltet diese. Für das Funktionieren von Traefik ist der Ordner */alf/traefik-public-certificates/* genauso wichtig, denn dieser enthält das Zertifikat für die Verschlüsselung der Verbindung.

Die Dateien für die Umgebungsvariablen (*/alf/.env.dev/* und */alf/.env.prod/*) und die Konfiguration des Docker-Compose (*/alf/docker-compose.dev.yml* und */alf/docker-compose.prod.yml*) sind dem jeweiligen Modus zugeordnet.

Zusätzlich existiert pro Modus jeweils ein Shell-Script (*/alf/start-dev.sh* und */alf/start-prod.sh*), welches für den Start des jeweiligen Modus verantwortlich ist. Das Shell-Script des Dev-Modus ist auch für das aufräumen der Datenbank und des Dateisystems verantwortlich.

B.2.5 Log-Dateien

Die Log-Dateien sind unter */alf/logs/* zu finden. Es existieren pro Container bis zu fünf Dateien, welche mit dem Erstellungsdatum und dem Hostname (Containername) beschriftet sind.

B.3 Zusätzliche Projekttypen

Um einen zusätzlichen Projekttypen anzubieten, müssen folgende Schritte befolgt werden.

1. Neuer Runner programmieren
2. *project_runner* anpassen
3. Datenbankeintrag anlegen
4. Ordner auf dem Dateisystem erstellen
5. Spezifische Dateien auf dem Dateisystem ablegen

Neuen Runner programmieren
Damit der neue Projekttyp auch verwendet werden kann, muss nun für diesen ein neuer Runner geschrieben werden. Dieser Runner benötigt eine Funktion. Die Funktion bekommt einen Pfad zu einem temporären Ordner, welcher alle Dateien (Projekttyp-Tempalte, Projekt-Template und die Abgabe) enthält. Die Rückgabe muss vom Typen *ResponseDict* sein, ein Auszug des *alf_types* Package ist in der folgenden Abbildung B.2. Type-Klassen für ResponseDict gezeigt.

```
from typing import TypedDict

class ResultDict(TypedDict):
name: str
result: str
passed: bool

class ResponseDict(TypedDict):
error: str
results: list[ResultDict]
```

Abbildung B.2: Type-Klassen für ResponseDict

***project_runner* anpassen**
Damit der Job-Handle für jeden Projekttypen die richtige Funktion aufruft, muss das *PROJECT_RUNNER_DICT* im Modul *project_runner* ergänzt werden (siehe Abbildung B.3. Mapping von Projekttyp und Runner-Funktion). Der Key des Dictionarys ist dabei der Name des Projekttyps und das Value ist die geschriebene Runner-Funktion.

Datenbankeintrag anlegen
Im Unterabschnitt B.1.3. Projekttyp anlegen ist beschrieben wie dies zu erfolgen hat. Durch das Anlegen des Datenbankeintrags, wird der Pfad für das Dateisystem erstellt, welcher für die folgenden Schritte relevant ist.

Ordner auf dem Dateisystem erstellen
Als nächstes muss der Ordner *.../<projecttype_id>_<projecttype_name>/template* auf dem Dateisystem des Servers erstellt werden.

Spezifische Dateien auf dem Dateisystem ablegen
Falls der Projekttyp spezifische Dateien benötigt, können diese im entsprechenden *.../<projecttype_id>_<projecttype_name>/template* Ordner abgelegt werden. Spezifische Dateien für einen Projekttypen sind zum Beispiel Dateien, welche jedes Projekt von diesem Projekttyp benötigt. Im Fall von *CMake_Cpp_Cute* wäre dies die Cute.

```
RUNNER_DICT_TYPE = dict[str, Callable[[Path], Awaitable[ResponseDict]]]
PROJECT_RUNNER_DICT: RUNNER_DICT_TYPE = {
        "CMAKE_CPP_Cute": run_cmake_cpp_cute_project
}
```

Abbildung B.3: Mapping von Projekttyp und Runner-Funktion

Kapitel C

Benutzer Dokumentation

Die Benutzer Dokumentation richtet sich an die Studenten und Dozenten, welche das ALF-Tool verwenden werden. Es wird im Allgemeinen erklärt, wie man sich bei dem Tool anmeldet und wie man sich darauf bewegt.
Für die Studenten ist erklärt, wie man eine Abgabe tätigt und wie man die Lösungen seiner Abgaben einsehen kann. Den Dozenten wird aufgezeigt, wie man mit der zusätzlichen Unterseite agiert. Im Wesentlichen werden Fragen wie "Wie wird ein Projekt aufgesetzt?", "Wie wird ein bestehendes Projekt geändert?", "Wie kann ein Projekt getestet werden?" und "Wie können die Abgaben der Studenten angesehen werden?" geklärt.
Die folgenden Bilder dienen zur Veranschaulichung und können vom aktuellen Stand des Tools leicht abweichen.

C.1 Anmelden

Voraussetzung:

- Student oder Dozent der OST mit einem Konto auf GitLab-OST

Das komplette Abgabeportal ist gesichert und nur für Studenten oder Dozenten der OST zugänglich (siehe Abbildung C.1. Anmeldeaufforderung). Die Authentifizierung wird nicht durch das ALF-Tool geregelt, sondern durch GitLab-OST. Wenn man unautorisiert auf das Portal zugreifen will, wird man aufgefordert sich über GitLab-OST anzumelden.

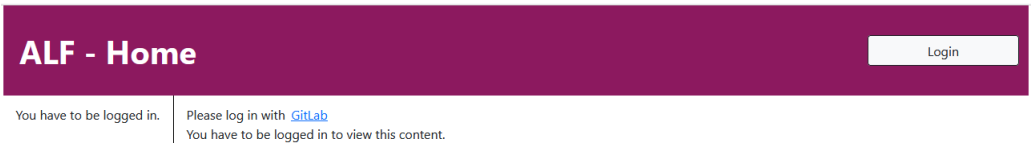


Abbildung C.1: Anmeldeaufforderung

Nach dem erfolgreichen Anmelden auf GitLab-OST wird man gefragt, ob man dem ALF-Tool den Read-Only Zugriff auf die Benutzerdaten geben möchte (siehe Abbildung C.2. GitLab-OST Autorisierung). Das Portal verwendet nur den Benutzername, die E-Mail-Adresse und den vollen Namen. Alle Daten dienen nur dem Portal, sie werden keinesfalls weitergegeben. **Es ist zwingend Notwendig das ALF-Tool zu autorisieren, da sonst die Anmeldung und Verwendung nicht möglich ist.**

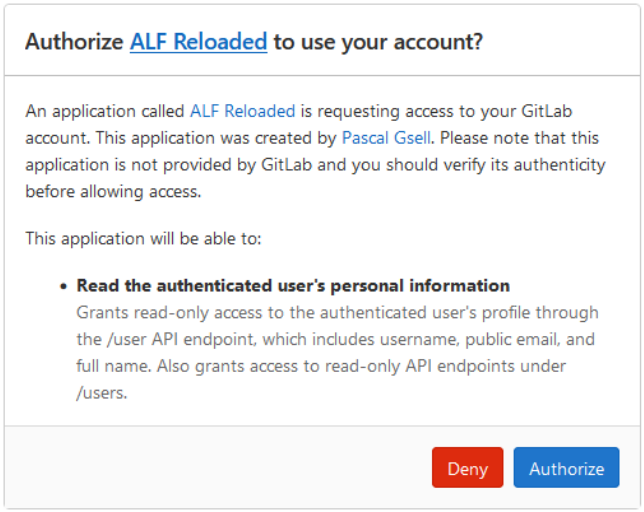


Abbildung C.2: GitLab-OST Autorisierung

Anschliessend wird man automatisch zurück auf die Homeseite vom ALF geleitet. Wenn in der oberen rechten Ecke der Benutzername steht, war die Anmeldung erfolgreich und ALF kann nun verwendet werden (siehe Abbildung C.3. Erfolgreiches Einloggen).

Abbildung C.3: Erfolgreiches Einloggen

C.2 Navigation

Voraussetzungen:

- Erfolgreiches anmelden über GitLab-OST

Alle verfügbaren Funktionen des ALF-Tools sind über die Navigation auf der linken Seite erreichbar (siehe Abbildung C.4. Navigation auf ALF). Falls man als Student eingeloggt ist, sind nur die beiden Funktionen Hochladen einer Lösung und Einsehen einer abgegebenen Lösung verfügbar. Als Dozenten ist zusätzlich die Funktion Auswahl des Projekts verfügbar. Unter diesem kann der Dozent seine Projekte managen.

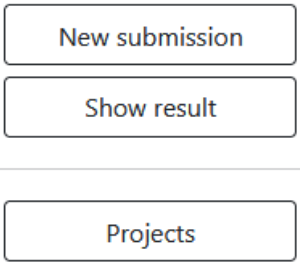


Abbildung C.4: Navigation auf ALF

C.3 Student

Voraussetzungen:

- Erfolgreiches anmelden über GitLab-OST

C.3.1 Hochladen einer Lösung

In diesem Teil des Tools können die Studenten ihre Lösung zur Überprüfung und Auswertung hochladen, eine mögliche Auswahl eines Projekts ist in Abbildung C.5. Auswahl eines Projekts abgebildet. Zuallererst muss man das gewünschte Modul auswählen, erst dann werden die Verfügbaren Projekte angezeigt.

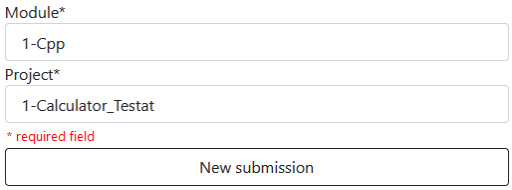


Abbildung C.5: Auswahl eines Projekts

Auf der danach folgenden Seite kann man anschliessend seine Abgabe hochladen (siehe Abbildung C.6. Hochladen einer Lösung). Als erstes sieht man noch einmal das Modul, sowie das Projekt mit seiner Beschreibung. Im nachfolgenden gibt es jeweils für jede benötigte Datei ein Auswahlfeld. **Die Dateinamen und Dateitypen müssen mit den geforderten Dateien übereinstimmen, sonst kann das Formular nicht erfolgreich abgeschickt werden.** Zum Schluss ist es noch möglich zusätzliche Teammitglieder für die Abgabe anzuwählen.

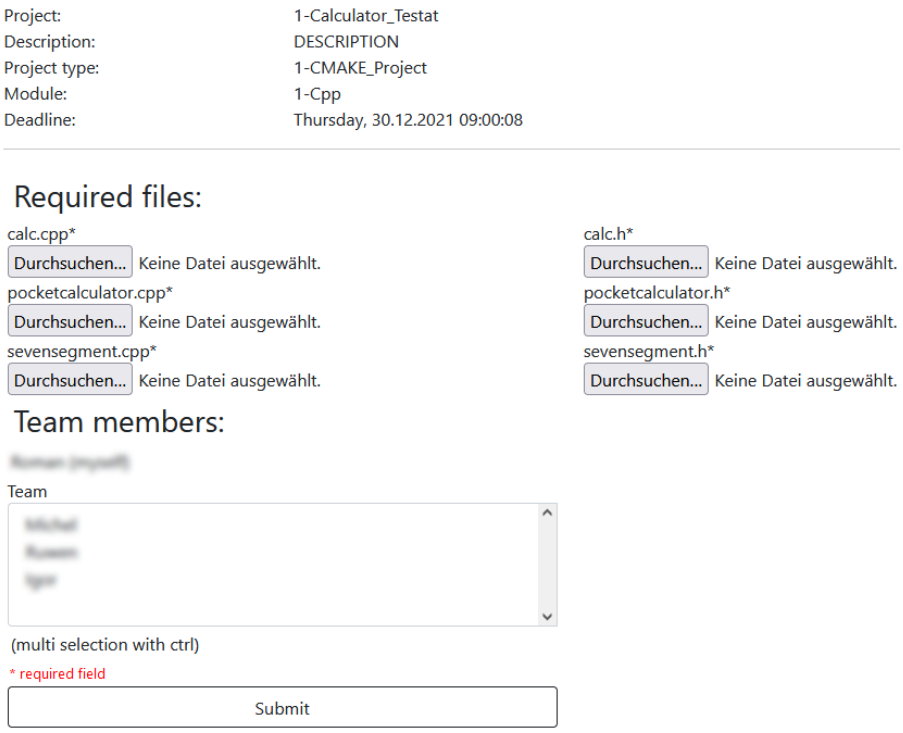


Abbildung C.6: Hochladen einer Lösung

Nach erfolgreichem Ausfüllen, wird die Abgabe intern ausgeführt und man bekommt eine Meldung, dass das Abgeben erfolgreich war (siehe Abbildung C.7. Erfolgreiches Hochladen einer Abgabe). Über den angezeigten Link kommt man direkt zu Auswertung.

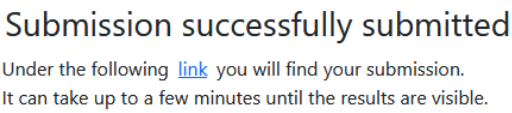


Abbildung C.7: Erfolgreiches Hochladen einer Abgabe

C.3.2 Einsehen einer abgegebenen Lösung

Ähnlich wie beim Hochladen einer Lösung muss auch hier zu Beginn die Abgabe selektiert werden, eine mögliche Auswahl einer Abgabe ist in Abbildung C.8. Auswahl einer Abgabe abgebildet. Es werden nur die Projekte angegeben, welche auch abgegeben wurden, ob als Hochlader oder als Teammitglied. Das Abgabe-Dropdown füllt sich auch hier erst nach der Wahl des Projekts.

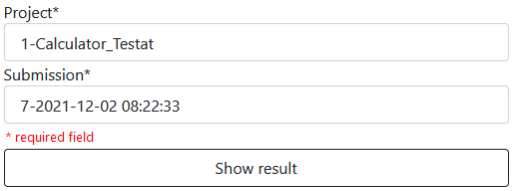


Abbildung C.8: Auswahl einer Abgabe

Das Auswerten der Abgabe kann je nach Belastung des ALF-Tools ein paar Minuten in Anspruch nehmen. Im Fall, dass man zu früh auf der Auswertungsseite ist, bekommt man den Hinweis, dass die Auswertung noch im Gange ist (siehe Abbildung C.9. Ausstehende Auswertung einer Abgabe).

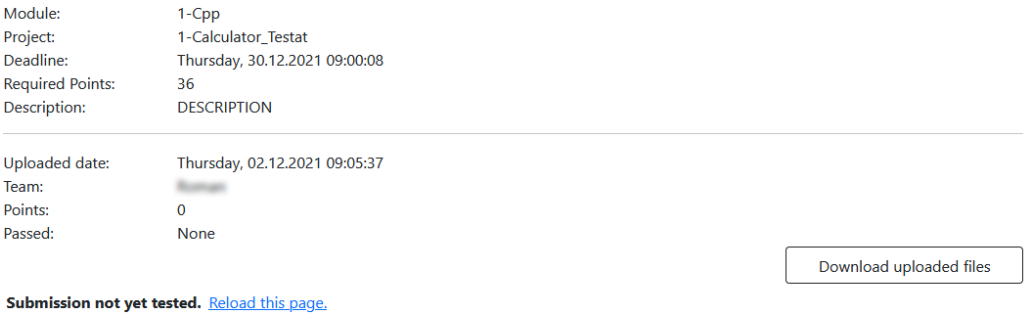


Abbildung C.9: Ausstehende Auswertung einer Abgabe

Wenn die Auswertung zu Ende ist, sieht man entweder die Unit-Tests und deren Resultate oder falls was nicht funktioniert hat, zum Beispiel das Compilieren des Codes, sieht man die Fehlermeldung, welche geworfen wurde. In der folgenden Abbildung ist eine Mögliche Abgabe und ihre Fehlermeldungen dargestellt.

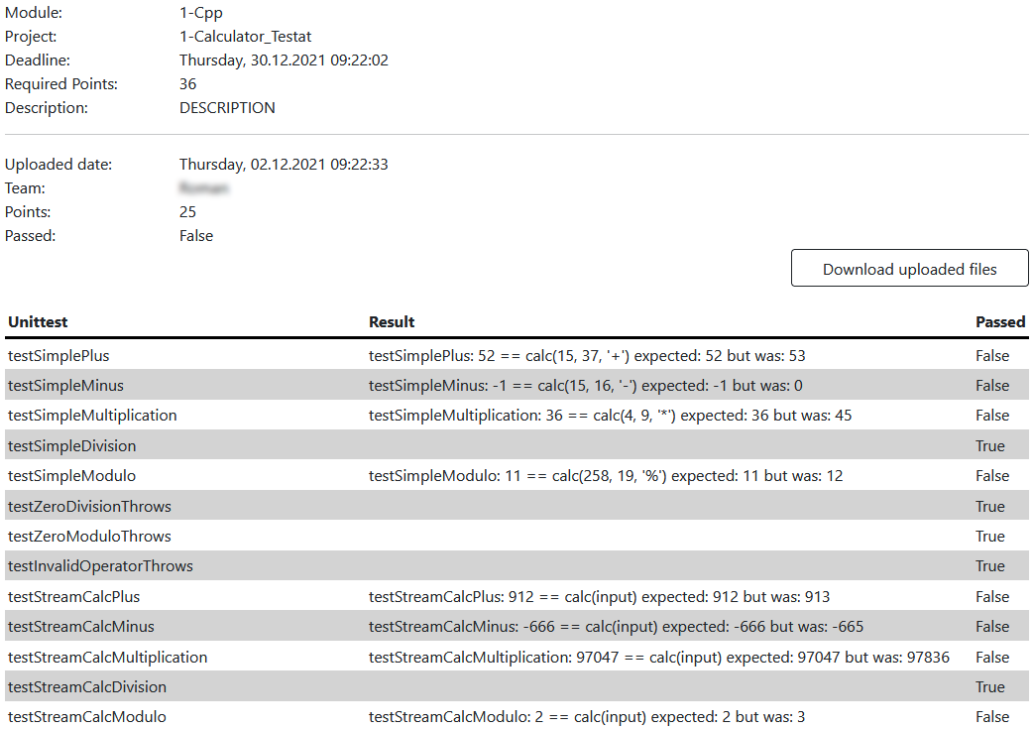


Abbildung C.10: Detailansicht einer Abgabe

Zusätzlich ist in der Abbildung C.10. Detailansicht einer Abgabe einen Download-Button zu sehen. Ein Klick auf diesen erlaubt einem die frühere Abgabe wieder herunterzuladen.

C.4 Dozent

Voraussetzungen:

- Erfolgreiches anmelden über GitLab-OST
- Registriert als Dozent

C.4.1 Auswahl des Projekts

Als Dozenten hat man den zusätzlichen Menüpunkt Project (siehe Abbildung C.4. Navigation auf ALF). Unter diesem Punkt kann man ein neues Projekt erstellen, oder ein bestehendes Projekt modifizieren, testen oder sich die Abgaben anzeigen lassen. In der folgenden Abbildung ist eine Auswahl eines Projekts und die Buttons gezeigt.

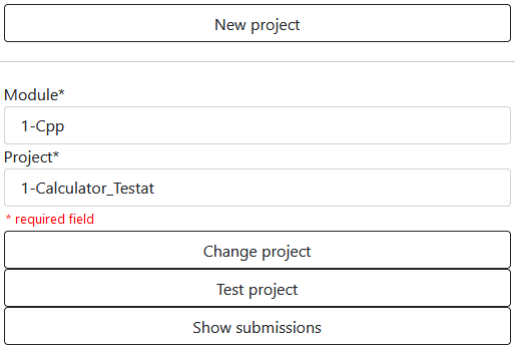


Abbildung C.11: Auswahl des Projekts

C.4.2 Abgaben einsehen

Auf dieser Seite bekommt man einen Überblick über alle Abgaben auf diesem Projekt. Eine beispielhafte Übersicht ist in der Abbildung C.12. Abgaben einsehen zu sehen. Über einen Klick auf "Details" kommt man auf die Abgabe selbst (siehe Abbildung C.10. Detailansicht einer Abgabe). Einen Klick auf "Download all Submissions" lädt alle, zu diesem Projekt gehörenden, Abgaben herunter.

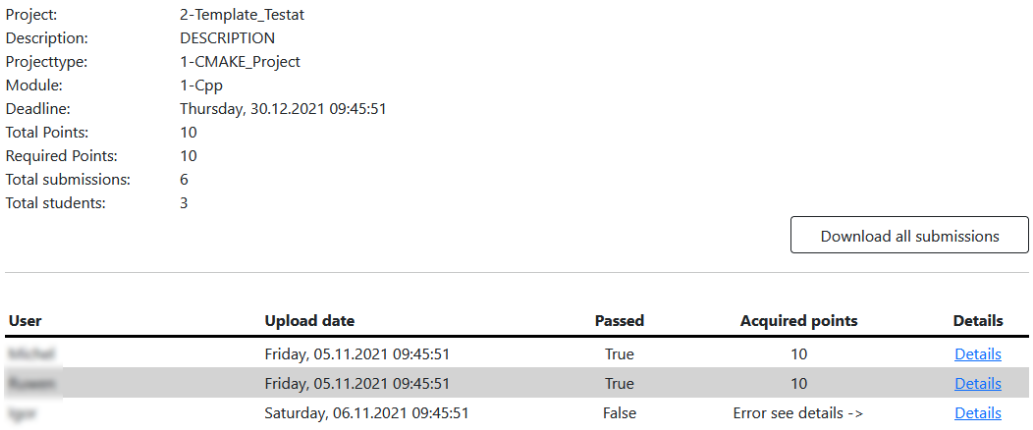


Abbildung C.12: Abgaben einsehen

C.4.3 Projekt erstellen

Projekte können nur zu bestehenden Projekttypen und Modulen hinzugefügt werden. Während ein Modul einfach vom Administrator erstellt werden kann, ist das hinzufügen eines neuen Projekttypen eine grössere Arbeit und benötigt die Arbeit eines Entwicklers. In der Abbildung C.13. Erstellen eines Projekts ist das Formular für ein neues Projekt dargestellt, welches automatisch mit Standardwerten befüllt ist.

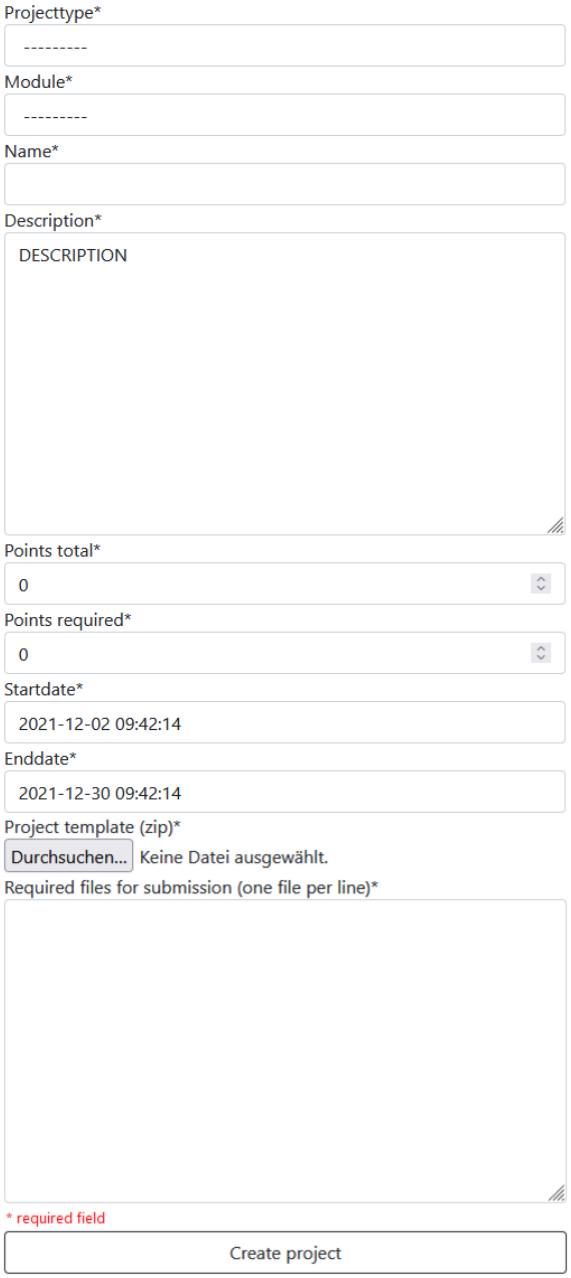


Abbildung C.13: Erstellen eines Projekts

Der Name des Projekts darf 32 Zeichen lang sein, während die Beschreibung bis zu 4095 Zeichen beinhalten kann. Die Beschreibung kann man beschränkt mit Markdown schreiben. Das Styling mit Markdown wird auf HTML-Tags übersetzt, in der folgenden Tabelle sind die verfügbaren Möglichkeiten aufgeführt.

Markdown	Beispiel	HTML-Umsetzung
Umbruch	\crlf	
Doppelumbruch	\crlf \crlf	Text vor und nach Umbruch kommt in ein
Titel	# Titel	Titel

Markdown	Beispiel	HTML-Umsetzung
Titel 2	## Titel 2	Titel 2
Titel ...	##... Titel ...	<h...> Titel ... </h...>
Titel 6	###### Titel	Titel 6
Kursiv	*Text*	Text
Fett	**Text**	Text
Blockquotes	> Text	<blockquote> Text </blockquote>
Geordnete Liste	1. Punkt	Punkt
Ungeordnete Liste	- Punkt	Punkt
Sourcecode	`Ich bin Code`	<code> Ich bin Code </code>
Links	[Beispiel Code](gitlab.ost.ch)	 Beispiel Code

Tabelle C.1: Auflistung der Styling-Möglichkeiten der Beschreibung

Die totalen Punkte werden primär für das Anzeigen benötigt, sie entsprechen der Gesamtanzahl an Unit-Tests im Projekt. Die benötigten Punkte hingegen werden für die Auswertung der Abgaben benötigt, eine Abgabe ist erst dann bestanden, wenn diese Punktzahl erreicht wird.
Das Startdatum bestimmt, ab wann das Projekt den Studenten angezeigt wird. Dies ist standardmässig auf den Zeitpunkt des Abrufen des Formulars gesetzt. Das Enddatum bestimmt bis zu welchem Zeitpunkt die Studenten eine Abgabe tätigen können. Aktuell implementiert ist der Standardwert gleich des Standardwerts des Startdatums plus 4 Wochen, sprich 28 Tage. Beim Datum ist es wichtig, dass die vorgegebene Konfession eingehalten wird (yyyy-mm-dd hh:mm:ss).
Das Zip-Archiv muss alles beinhalten, was zur Validierung des Projekts benötigt wird.
Im letzten Feld des Formulars können die Dateien definiert werden, welche von den Studenten hochgeladen werden müssen. Mit einem Zeilenumbruch wird jeweils eine weitere Datei verlangt. Bei der Eingabe ist darauf zu achten, dass die Benennung genau stimmt, da die Studenten gezwungen werden, diese Namen zu verwenden.
Folgende Dateien werden für die jeweiligen Projekttypen zwingend benötigt:

CMake_Cpp_Cute Project

- CMakeList.txt:

```
cmake_minimum_required(VERSION 3.14)
project(Testat1Alf)

set(CMAKE_CXX_STANDARD 17)

include_directories(cute)

add_executable(test
        Test.cpp
        calc.cpp
        calc_suite.cpp
        pocket_calc_suite.cpp
        pocketcalculator.cpp
        seven_segment_suite.cpp
        sevensegment.cpp)
```

Abbildung C.14: Beispielinhalt einer CMakeList.txt

C.4.4 Projekt modifizieren

Es wird das gleiche Formular (siehe Abbildung C.13. Erstellen eines Projekts) für das Erstellen und Bearbeiten eines Projekts verwendet. Beim Modifizieren eines Projekts werden allerdings die Felder mit den vorhandenen Werten vorausgefüllt. Zusätzlich wird ein Spoiler-Dropdown eingeblendet (siehe Abbildung C.15. Löschen eines Projekts - Spoiler eingeklappt), durch dessen Aufklappen ein Button für das Löschen des Projekts eingeblendet wird (siehe Abbildung C.16. Löschen eines Projekts - Spoiler ausgeklappt). Ein Klick auf diesen Button löscht unwiderruflich alle Einträge in der Datenbank bezüglich Projekt und Abgaben zu diesem. Die Dateien auf dem Server bleiben bestehen und können durch einen Administrator noch heruntergeladen werden.

Abbildung C.15: Löschen eines Projekts - Spoiler eingeklappt

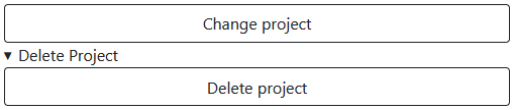


Abbildung C.16: Löschen eines Projekts - Spoiler ausgeklappt

C.4.5 Projekt testen

Das Testen eines Projekts funktioniert genau gleich wie das Hochladen einer Abgabe von den Studenten. Der Unterschied liegt darin, dass kein Team gewählt werden kann und die Abgabe nicht in der Datenbank persistiert wird. Daher wird man auch erst nach dem Erhalt der Auswertung weitergeleitet. In der folgenden Abbildung ist ein beispielhafter Test des Projekts *Calculator Testat* zu sehen.

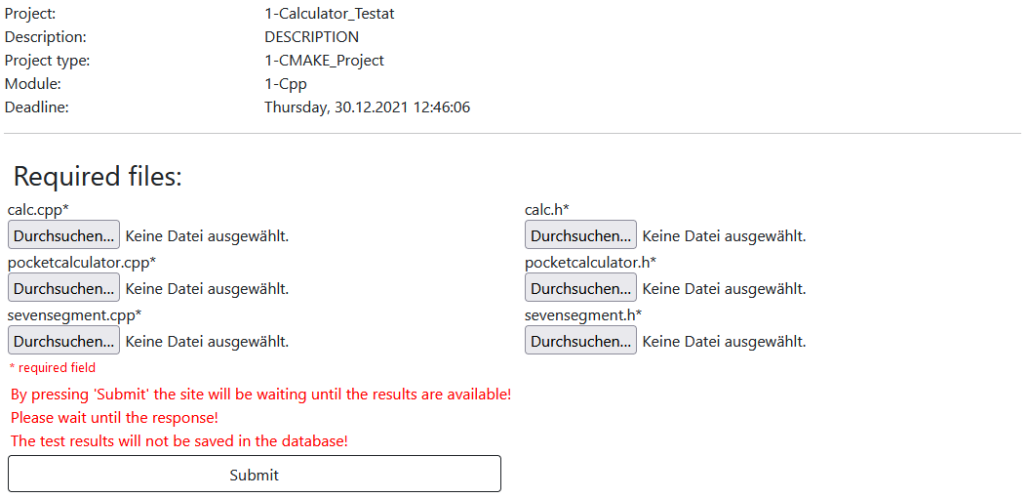


Abbildung C.17: Testen eines Projekts

Kapitel D

Datenbankschema

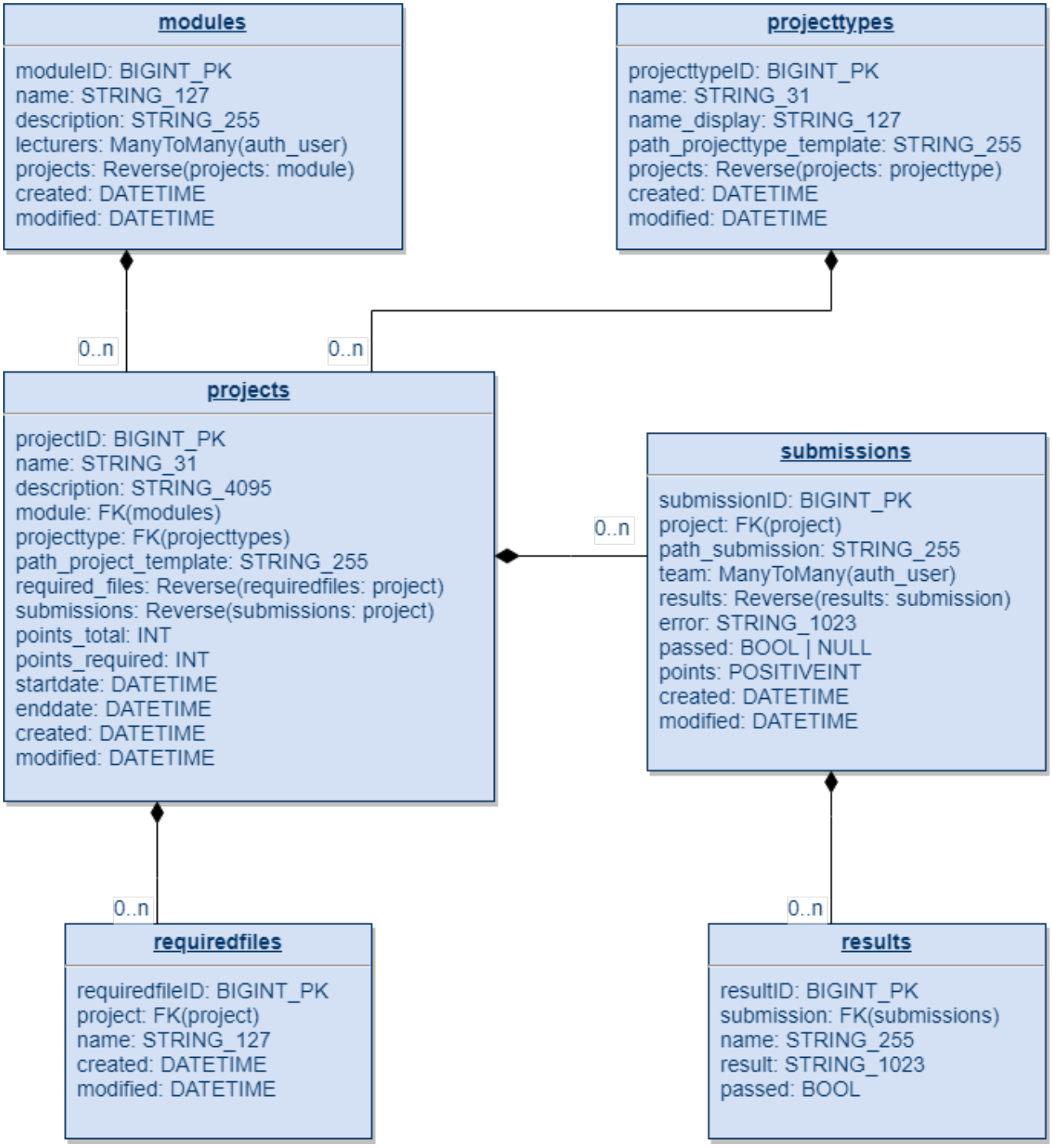


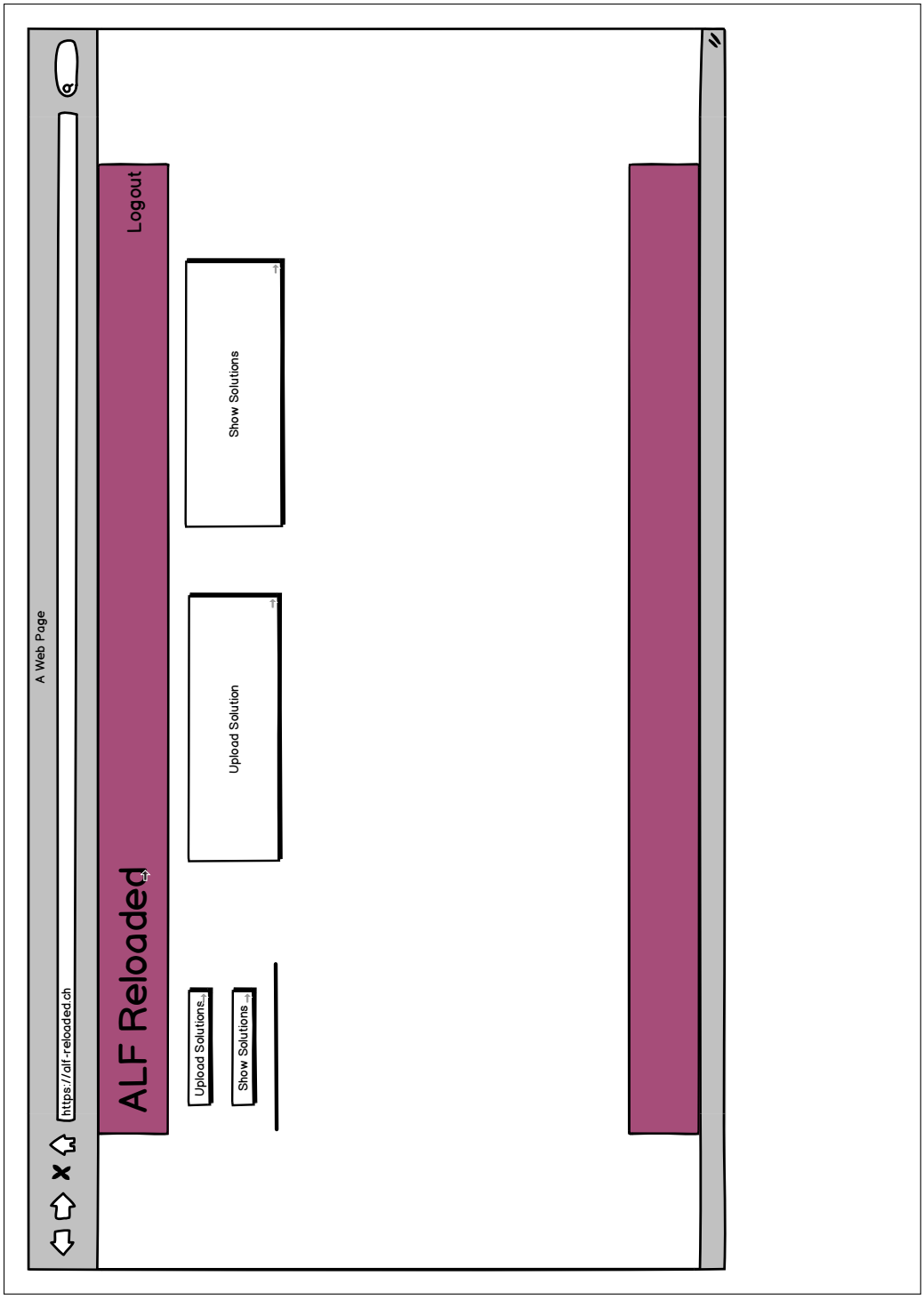
Abbildung D.1: Datenbankschema mit allen Attributen

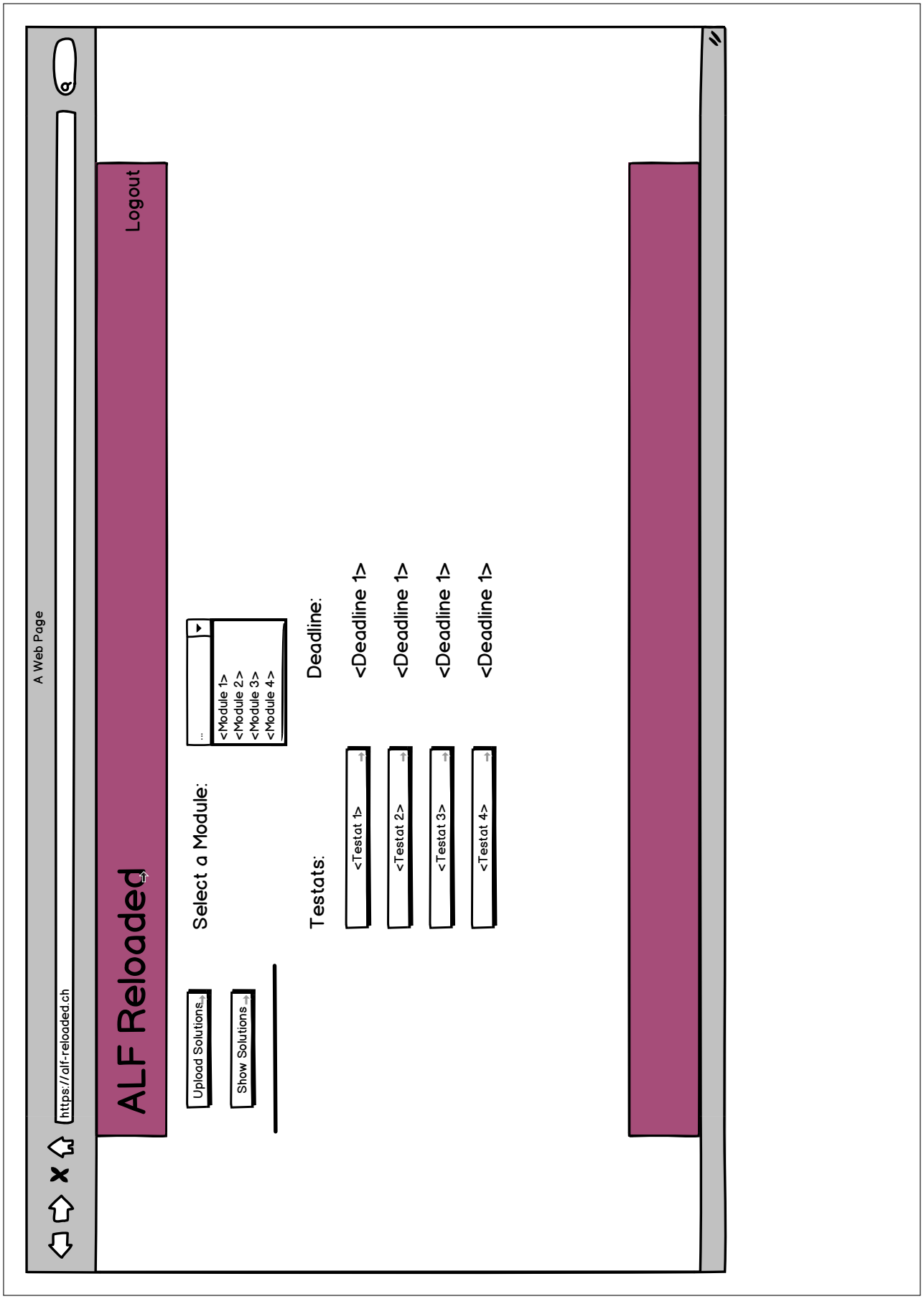
Kapitel F

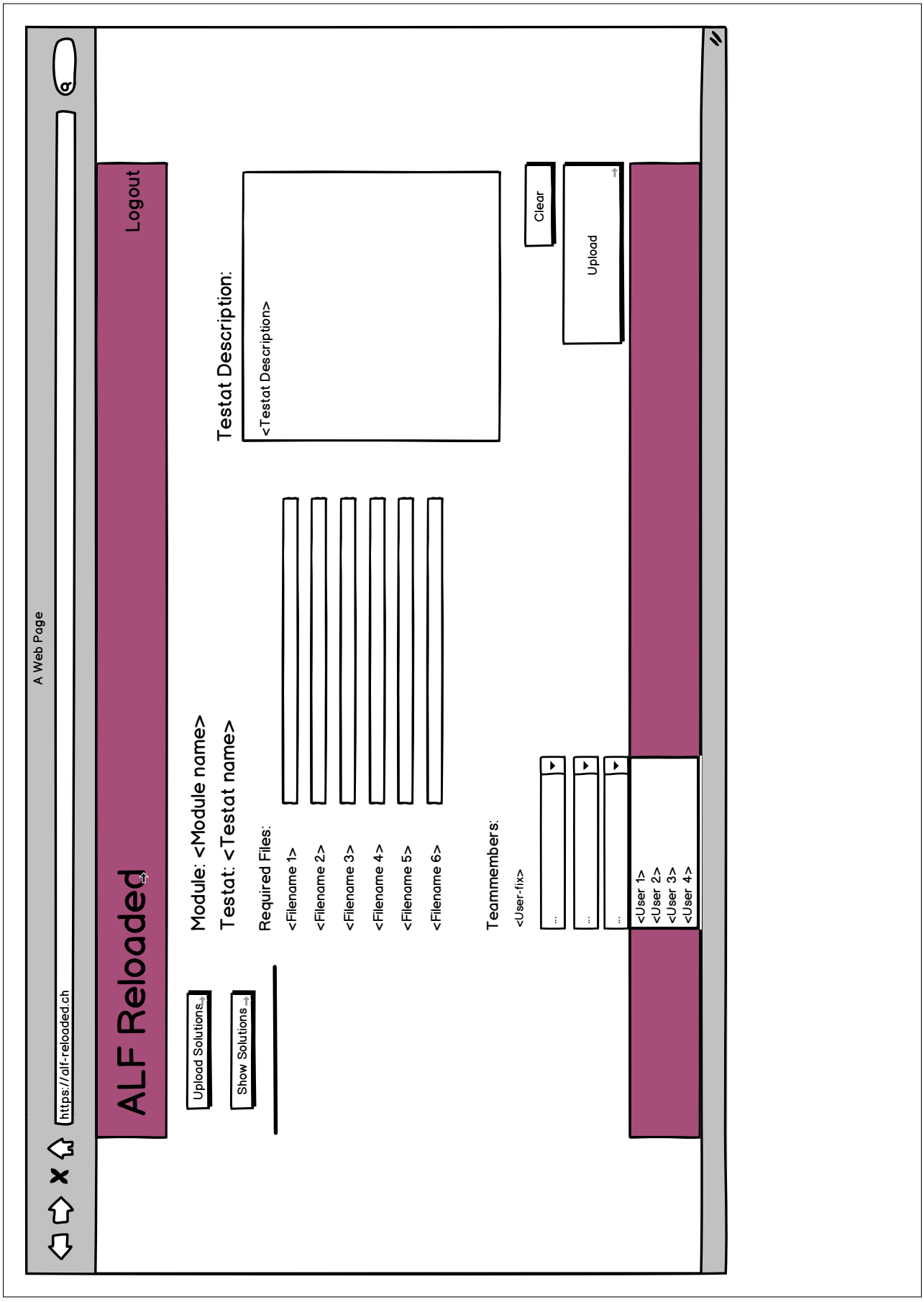
Zusätzliche Dateien

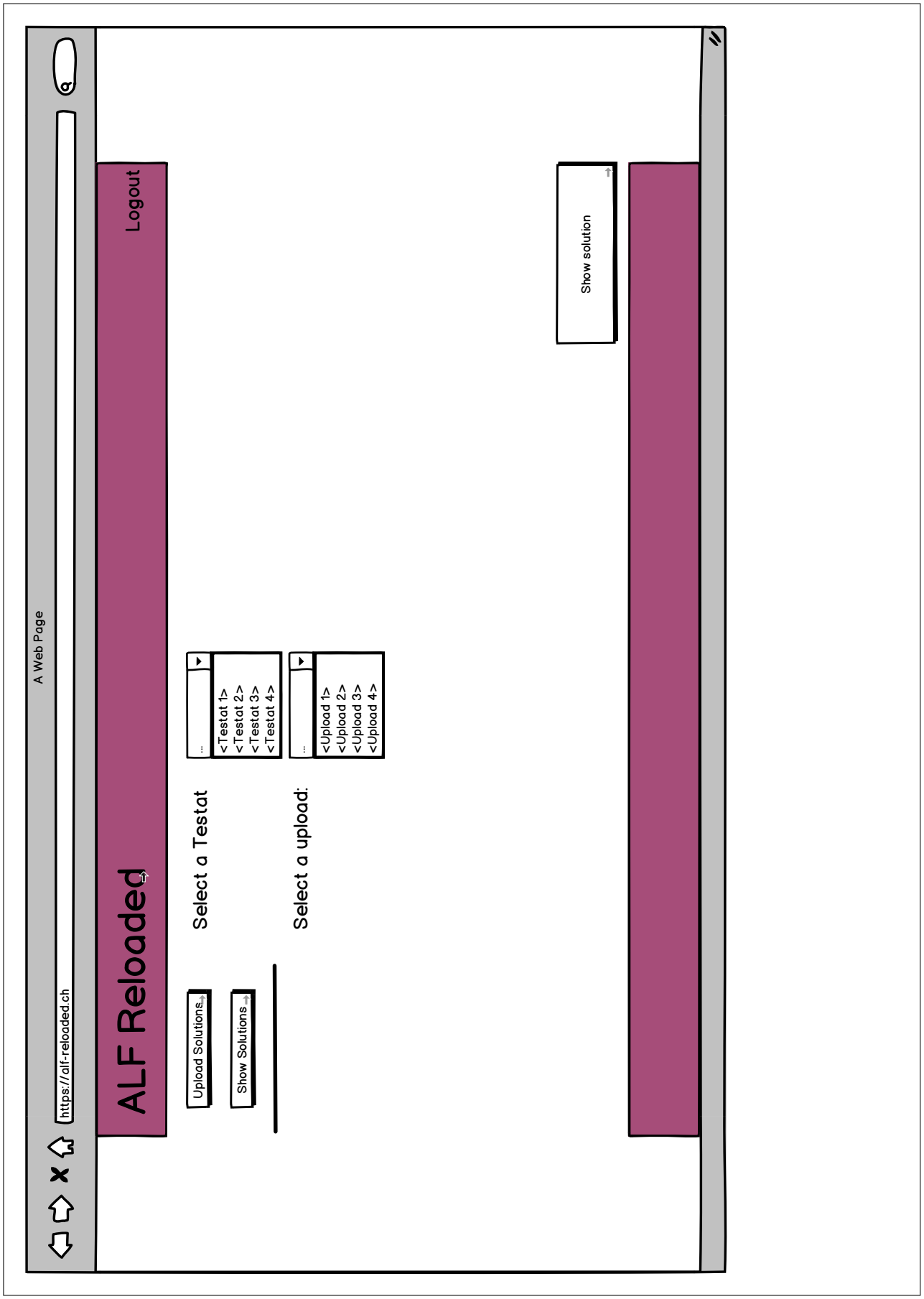
F.1 Wireframes

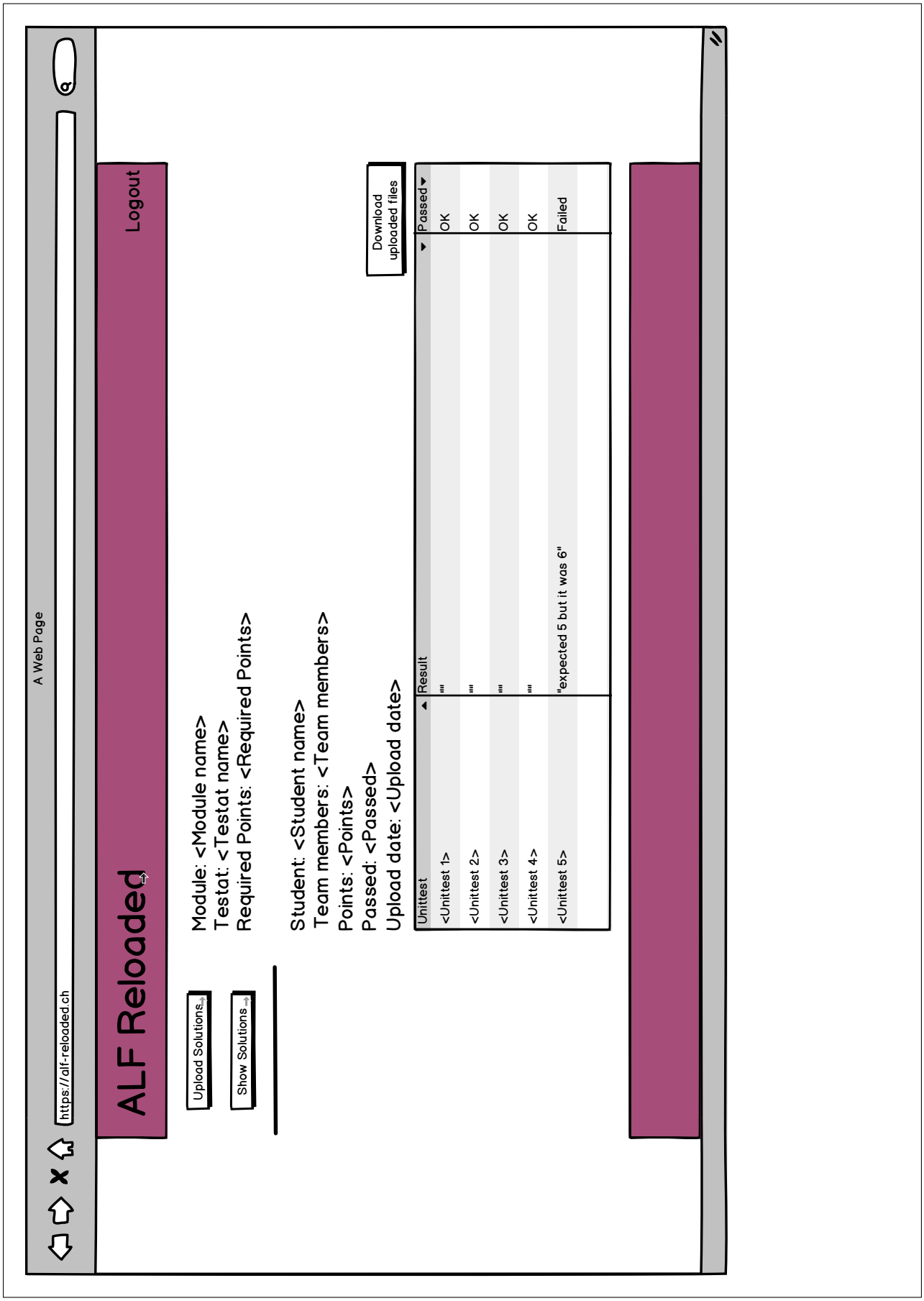
Die Wireframes stellen nur einen ersten Entwurf des Frontend dar. Das schlussendliche Frontenddesign kann an bestimmten Punkten abweichen.

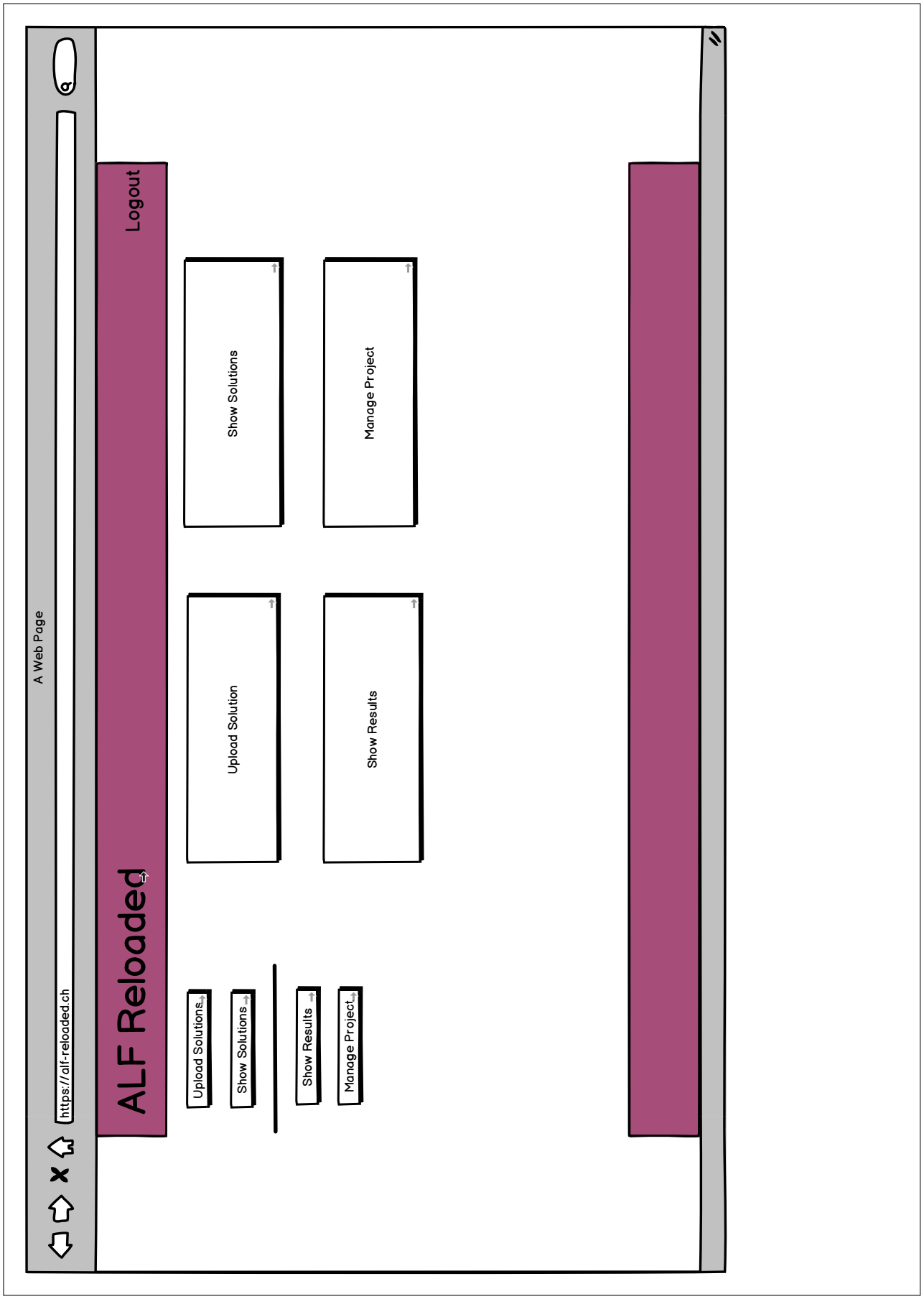


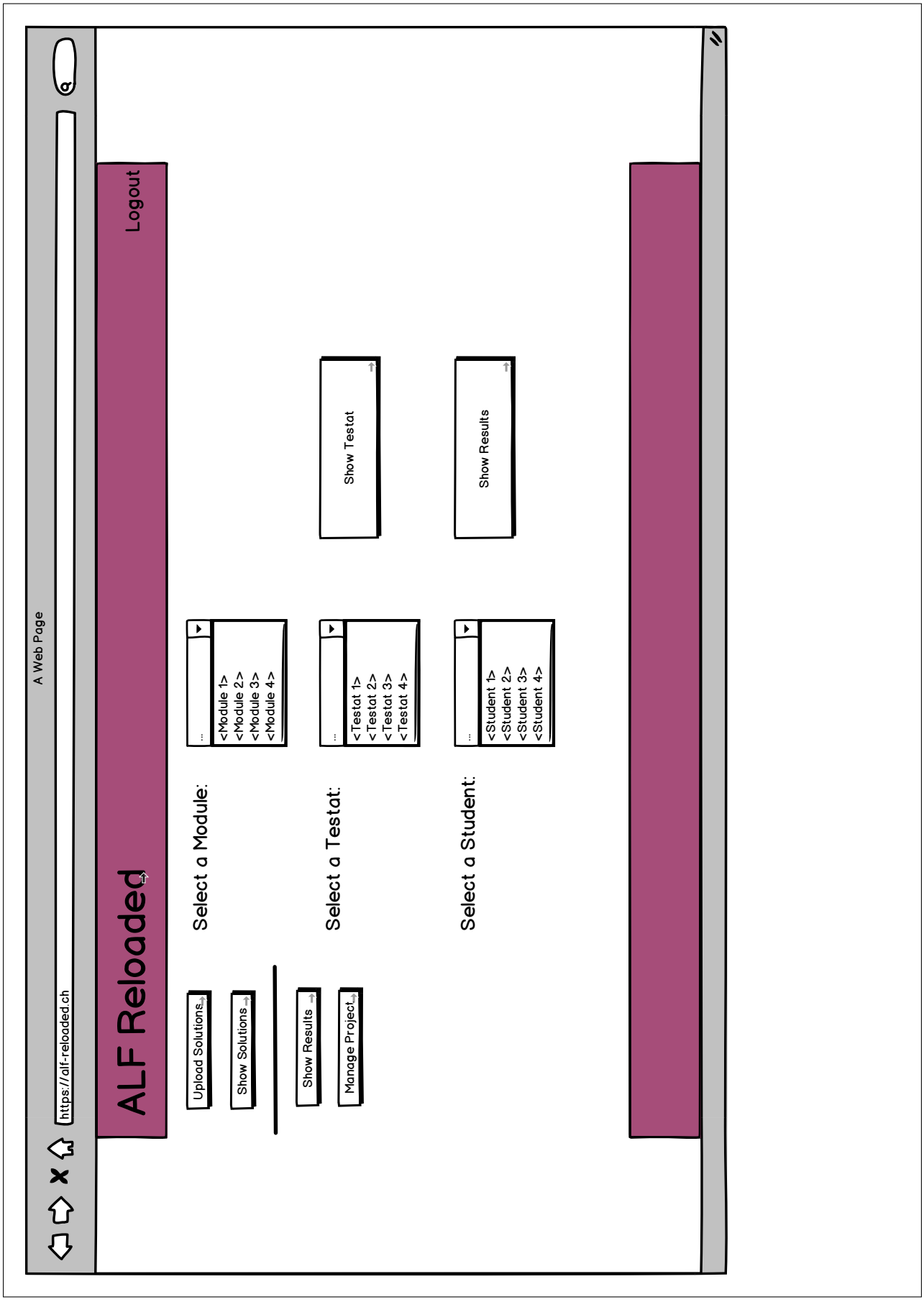

A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Select a Module:
<Module 1>
<Module 2>
<Module 3>
<Module 4>
Testats:
Deadline:
<Testat 1>
<Testat 2>
<Testat 3>
<Testat 4>
<Deadline 1>
<Deadline 1>
<Deadline 1>
<Deadline 1>


A Web Page
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Module: <Module name>
Testat: <Testat name>
Required Files:
<Filename 1>
<Filename 2>
<Filename 3>
<Filename 4>
<Filename 5>
<Filename 6>
Testat Description:
<Testat Description>
Clear
Upload
Teammembers:
<User-fix>
...
...
...
<User 1>
<User 2>
<User 3>
<User 4>

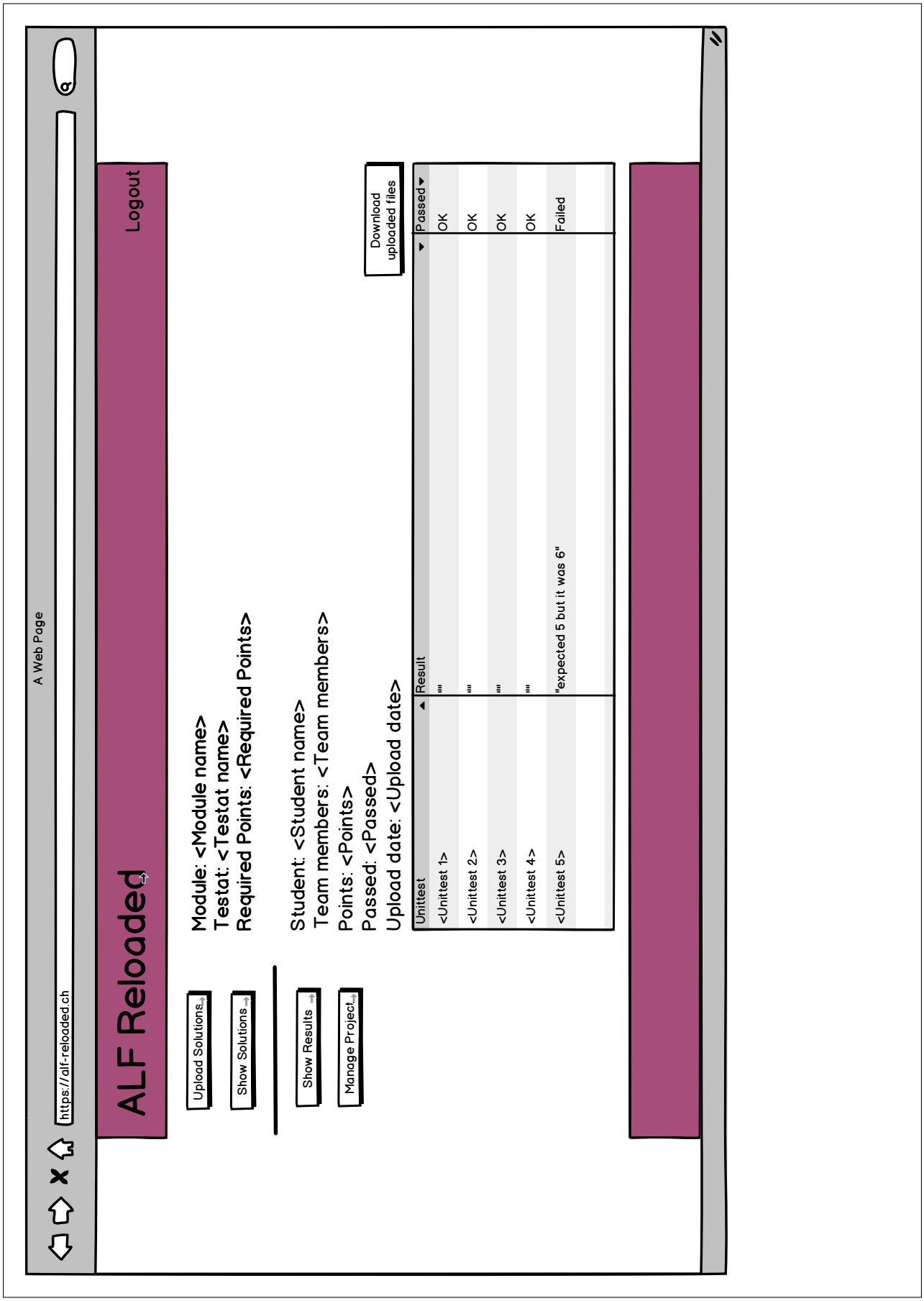

A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions...
Show Solutions
Select a Testat
<Testat 1>
<Testat 2>
<Testat 3>
<Testat 4>
Select a upload:
<Upload 1>
<Upload 2>
<Upload 3>
<Upload 4>
Show solution

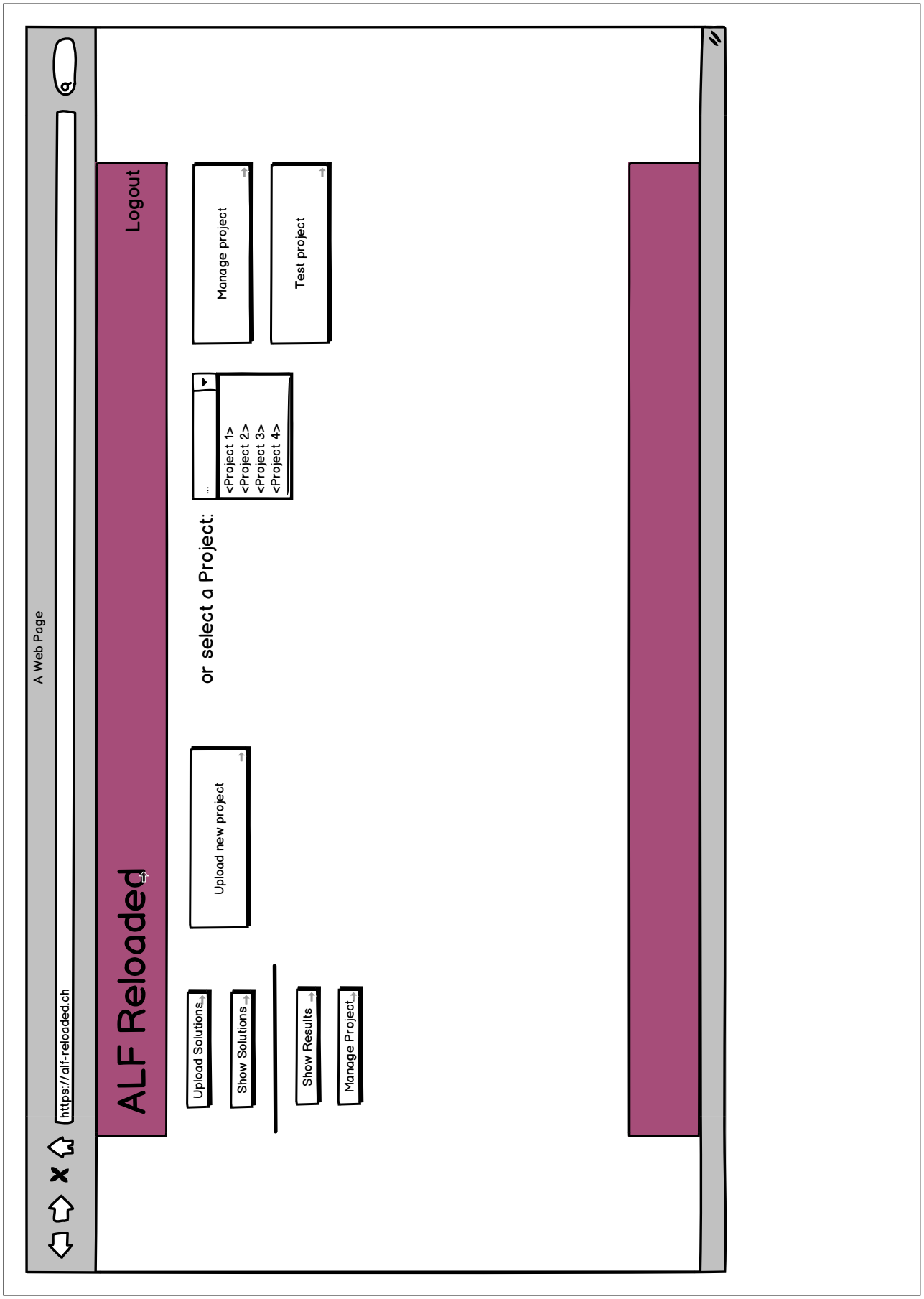

A Web Page
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Module: <Module name>
Testat: <Testat name>
Required Points: <Required Points>
Student: <Student name>
Team members: <Team members>
Points: <Points>
Passed: <Passed>
Upload date: <Upload date>
Download uploaded files
Unittest
Result
Passed
<Unittest 1>
""
OK
<Unittest 2>
""
OK
<Unittest 3>
""
OK
<Unittest 4>
""
OK
<Unittest 5>
"expected 5 but it was 6"
Failed

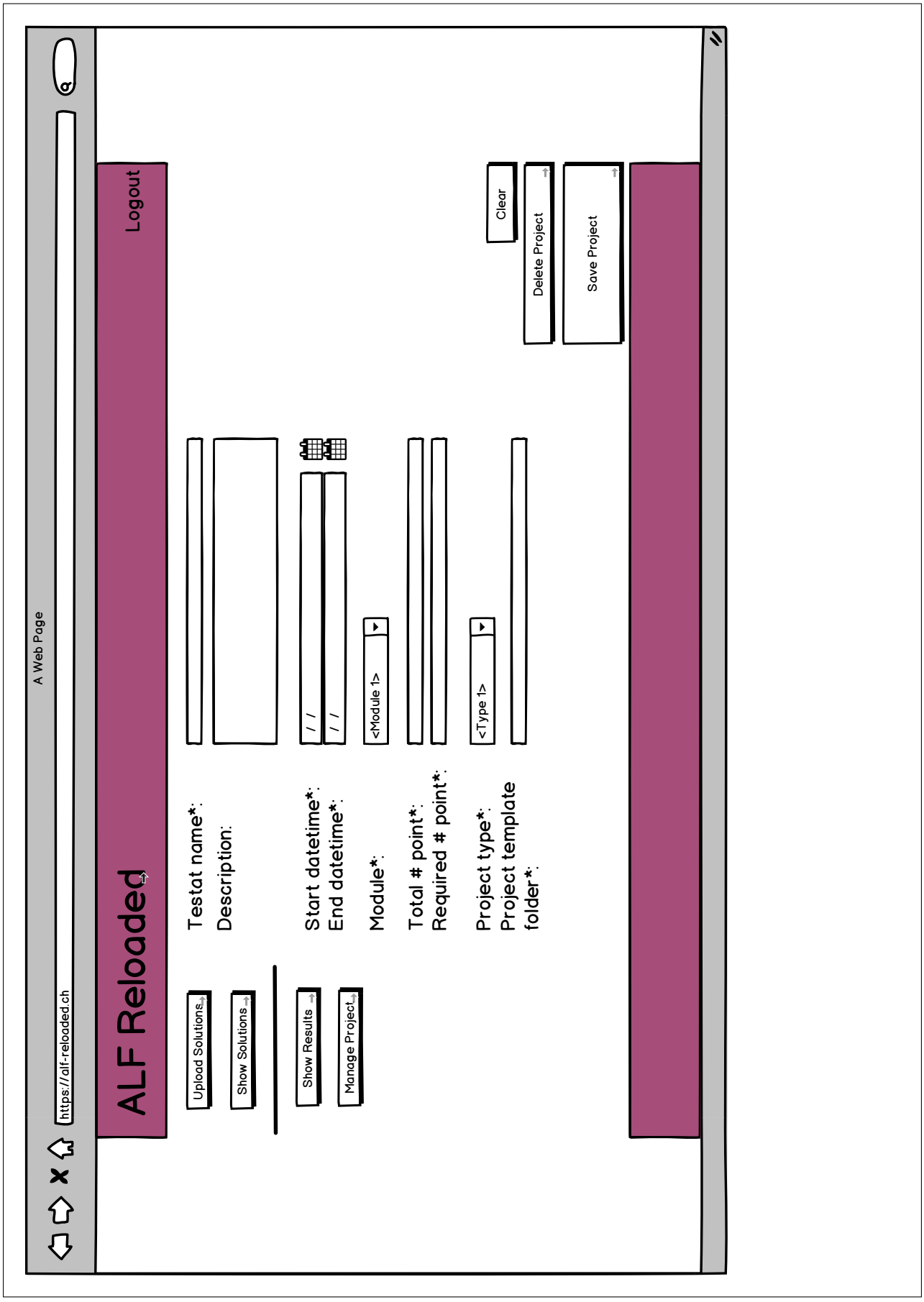

A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Upload Solution
Show Results
Show Solutions
Manage Project

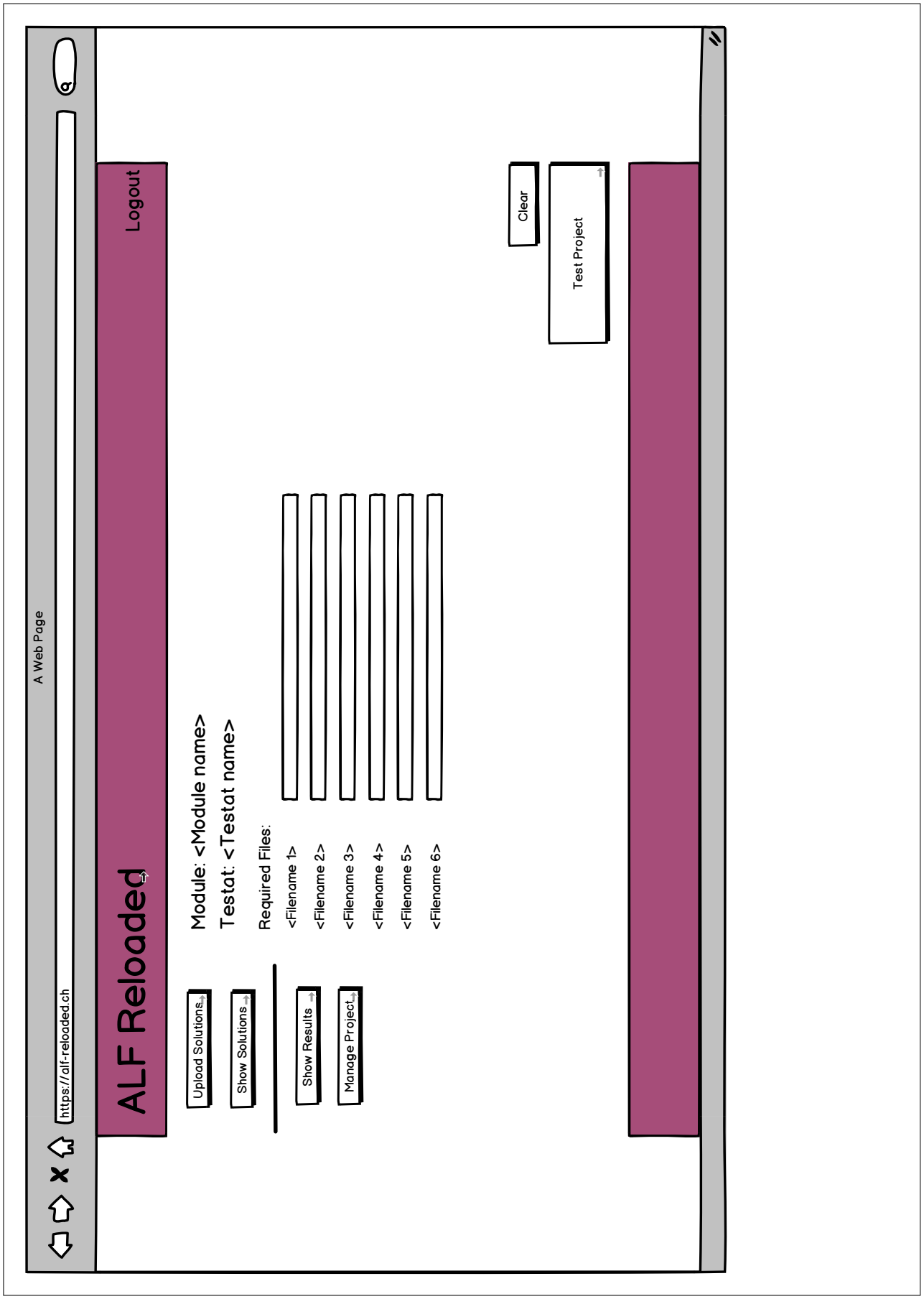

A Web Page
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Select a Module:
<Module 1>
<Module 2>
<Module 3>
<Module 4>
Select a Testat:
<Testat 1>
<Testat 2>
<Testat 3>
<Testat 4>
Select a Student:
<Student 1>
<Student 2>
<Student 3>
<Student 4>
Show Testat
Show Results

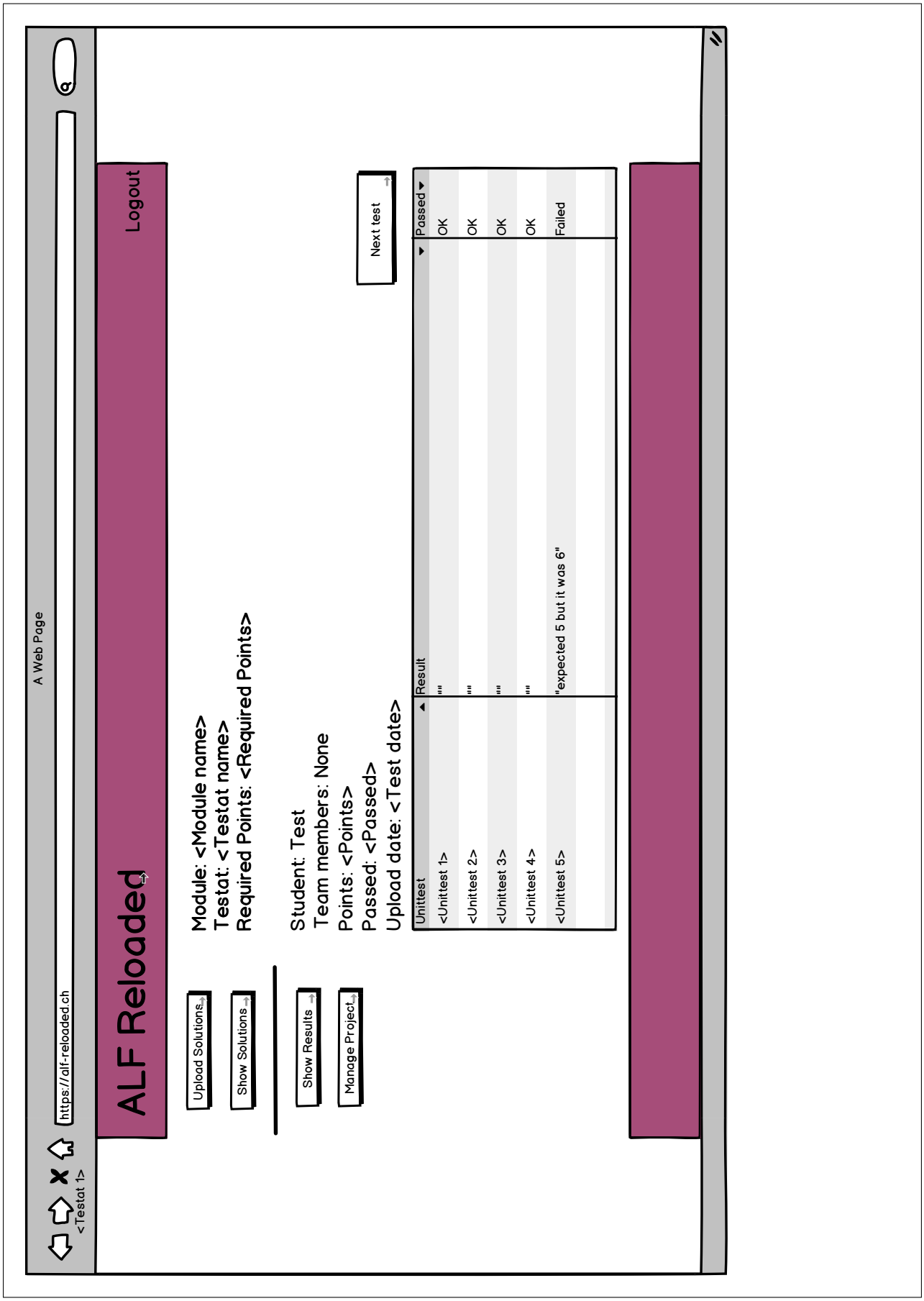
A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Module: <Module name>
Testat: <Testat name>
Total Submissions: <Submissions number>
Total Students: <Students number>
Download all uploaded Files
Student Team Upload date Passed Details Download uploaded files
<User 1> <Team members 1> <Date 1> <Passed 1> Details Download
<User 2> <Team members 2> <Date 2> <Passed 2> Details Download
<User 3> <Team members 3> <Date 3> <Passed 3> Details Download
<User 4> <Team members 4> <Date 4> <Passed 4> Details Download
<User 5> <Team members 5> <Date 5> <Passed 5> Details Download
<User 6> <Team members 6> <Date 6> <Passed 6> Details Download
<User 7> <Team members 7> <Date 7> <Passed 7> Details Download


A Web Page
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Module: <Module name>
Testat: <Testat name>
Required Points: <Required Points>
Student: <Student name>
Team members: <Team members>
Points: <Points>
Passed: <Passed>
Upload date: <Upload date>
Download uploaded files
Unittest
Result
Passed
<Unittest 1>
""
OK
<Unittest 2>
""
OK
<Unittest 3>
""
OK
<Unittest 4>
""
OK
<Unittest 5>
"expected 5 but it was 6"
Failed


A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions...
Show Solutions...
Upload new project
Show Results
Manage Project...
or select a Project:
...
<Project 1>
<Project 2>
<Project 3>
<Project 4>
Manage project
Test project


A Web Page
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Testat name*:
Description:
Start datetime*:
End datetime*:
Module*:
Total # point*:
Required # point*:
Project type*:
Project template folder*:
/ /
/ /
<Module 1>
<Type 1>
Clear
Delete Project
Save Project


A Web Page
https://alf-reloaded.ch
ALF Reloaded
Logout
Upload Solutions...
Show Solutions
Show Results
Manage Project...
Module: <Module name>
Testat: <Testat name>
Required Files:
<Filename 1>
<Filename 2>
<Filename 3>
<Filename 4>
<Filename 5>
<Filename 6>
Clear
Test Project


A Web Page
<Testat 1>
https://alf-relooded.ch
ALF Reloaded
Logout
Upload Solutions
Show Solutions
Show Results
Manage Project
Module: <Module name>
Testat: <Testat name>
Required Points: <Required Points>
Student: Test
Team members: None
Points: <Points>
Passed: <Passed>
Upload date: <Test date>
Next test
Unittest
Result
Passed
<Unittest 1>
""
OK
<Unittest 2>
""
OK
<Unittest 3>
""
OK
<Unittest 4>
""
OK
<Unittest 5>
"expected 5 but it was 6"
Failed

F.2 Aufgabenstellung

Aufgabenstellung für «ALF Reloaded»
Pascal Gsell / Roman Spring

1. Supervisor and Industry Partner

Diese Studienarbeit wird an der OST, Ostschweizer Fachhochschule, im Herbstsemester 2021 durchgeführt. Sie wird von Thomas Corbat (thomas.corbat@ost.ch) betreut.

Technischer Advisor: Nicola Jordan, Institut für Software (nicola.jordan@ost.ch)

2. Studierende

Das Projekt wird im Rahmen des Moduls «Studienarbeit Informatik» im Departement Informatik von folgenden Studierenden durchgeführt:

- Pascal Gsell (pascal.gsell@ost.ch)
- Roman Spring (roman.spring@ost.ch)

3. Einführung

In verschiedenen Modulen an der OST müssen Studierende Zulassungsbedingungen erfüllen, um die Prüfung des zugehörigen Fachs ablegen zu dürfen. In den Modulen CPl und CPIA sind die sogenannten Testate abzugebende Lösungen für Übungsaufgaben. Diese werden dann durch die Dozierenden korrigiert und die Studierenden erhalten ein Feedback [1].

Funktionale Aspekte der zu lösenden Aufgaben können mittels automatisierter Unit Tests verifiziert werden. Diese sind seitens der Betreuer vorhanden und unter anderem werden die Abgaben dagegen geprüft. Damit die Studierenden bereits eine vorgängige Selbstkontrolle vor der finalen Abgabe durchführen können, wurde am Institut für Software eine Web-Applikation entwickelt, welche es den Studierenden ermöglicht ihre Lösung hoch zuladen und zu schauen, ob sie die Tests erfolgreich besteht. Dies ist möglich ohne, dass die Tests den Studierenden zur Verfügung gestellt werden müssen.

Die Web-Applikation, genannt Automatisiertes Lern-Feedback/Automated Lesson-Feedback (ALF) [2], funktioniert sehr gut für die Studierenden und die subjektive Wahrnehmung der Dozierenden lässt auf eine sehr grosse Verbesserung der Qualität der Abgabe vermuten.

Leider verfügt die Plattform über den Nachteil, dass sie aktuell nur eine Benutzerschnittstelle für die Studierenden bietet. Sprich sie können ihre Lösungen hochladen und das Ergebnis anschauen, jedoch fehlt eine Benutzeroberfläche für die Dozierenden, um die Testate zu verwalten. Aktuell muss dies immer von einem Mitarbeiter des IFS gemacht werden.

4. Ziele des Projektes

In dieser Studienarbeit soll der oben genannte Mangel von ALF verbessert werden. Dabei steht es den Studierenden offen, ob sie ALF komplett neu entwickeln oder auf dem existierenden System aufbauen möchten. Die Wahl der Technologien steht ihnen grundsätzlich offen, diese muss aber in Absprache mit Nicola Jordan erfolgen und von ihm für gut befunden werden.

Der Funktionsumfang sollte mindestens die bestehende Funktionalität für Studierende bieten und zudem das Erstellen, Verwalten und Löschen von Projekten für Dozierende ermöglichen. Es sind viele weitere Features denkbar:

- Statistiken
- Gruppenabgaben
- Direkte Abgabe von Testaten über das Tool
- Plagiatscheck
- Korrektur online im Tool
- und viele mehr

Die umzusetzende Funktionalität muss jeweils mit dem Betreuer abgesprochen werden.

5. Dokumentation

Das Projekt muss entsprechend der Richtlinien des Departments Informatik dokumentiert werden [3]. Dies umfasst alle Analyse, Design, Implementation und Projekt Management, usw. Kapitel. Die Dokumentation kann wahlweise auf Deutsch oder Englisch verfasst werden. Ein Projektplan muss am Anfang ausgearbeitet und entsprechend dem effektiven Fortschritt angepasst werden. Schlussendlich müssen alle Resultate komplett auf den Archiv-Server der OST hochgeladen werden.

Eine gedruckte Ausgabe der Dokumentation muss dem Betreuer abgegeben werden (farbig und doppelseitig gedruckt, gebunden).

6. Wichtige Daten

* Die URLs könnten noch angepasst werden.

20.09.2021	**Semesterstart und Beginn des Projektes**
Bis 21.12.2021	**Erfassen des Abstracts im Online-Tool und Überprüfung durch den Betreuer [4]**
24.12.2021, 17.00	**Finale Abgabe über den Archiv-Server [5]**

7. Bewertung

Bei erfolgreichem Bestehen der Studienarbeit erhalten die Studierenden 8 ECTS-Punkte. Der geschätzte Aufwand für ein ECTS beträgt 30 Arbeitsstunden (vgl. Modulbeschreibung [6]). Der Betreuer ist verantwortlich für die Bewertung der geleisteten Arbeit.

Kriterium	Weight
1. Organisation, Ausführung	1/5
2. Bericht (Abstract, Management Summary, technischer und persönlicher Bericht) sowie Struktur, Visualisierung und Sprache der ganzen Dokumentation.	1/5
3. Inhalt der Arbeit	3/5

Weiter gelten die generellen Regularien für Studienarbeiten im Departement Informatik.

8. Referenzen

[1] https://www.ost.ch/de/studium/informatik/bachelor-informatik/studieninhalt-und-aufbau

[2] https://alf-uploader.sifs0005.infs.ch

[3] https://ostch.sharepoint.com/:b:/r/teams/TS-StudiengangInformatik/Freigegebene%20Dokumente/Studieninformationen/Studien-%20und%20Bachelorarbeiten/Informationen/Leitfaden%20BA%20SA%20v1.0.pdf?csf=1&web=1&e=1qAL6o

[4] https://abstract.rj.ost.ch

[5] https://avt.i.ost.ch

[6] https://studien.rj.ost.ch/allModules/24386_M_SAI14.html

Rapperswil, 20. September 2021

Thomas Corbat

Lehrbeauftragter
OST – Ostschweizer Fachhochschule

F.3 Guideline Dokumentation

Struktur Dokumentation

Hier bei handelt es sich um eine Richtlinie, von welcher wenn sinnvoll in begründeten Fällen abgewichen werden soll. Teile und Inhalte können, abhängig vom Projekt wegfallen oder hinzu kommen.

Abstract

- Der Abstract richtet sich an den Spezialisten auf dem entsprechenden Gebiet und beschreibt daher in erster Linie die (neuen, eigenen) Ergebnisse und Resultate der Arbeit. (Aus Anleitung Dokumentation FS21 vom SG-I).
- Der Umfang beträgt in der Regel eine halbe Seite Text
- Keine Bilder

Management Summary

- Das Management Summary richtet sich in der Praxis an die "Chefs des Chefs", d.h. an die Vorgesetzten des Auftraggebers (diese sind in der Regel keine Fachspezialisten). Die Sprache soll knapp, klar und stark untergliedert sein. (Aus Anleitung Dokumentation FS21 vom SG-I)
- Der Umfang beträgt in der Regel 3-5 Seiten.
- Bilder sind hier sinnvoll

Table of Contents

- Inhaltsverzeichnis

Introduction

- Beschreibung der Ausgangslage
- Beschreibung der Aufgabe
- Rahmenbedingungen
- Vorarbeiten
- Übersicht über die kommende Kapitel

Analysis

- Beschreibung des System-Kontexts
- Funktionale und nicht-Funktionale Anforderungen
- Use Cases/Scenarios
- Bestehende Infrastruktur
- Abhängig vom Projekt: Risikoanalyse

Design

- Beschreibung des Entwurfs der Lösung
- Architektur-Übersicht
- Internes Design (Subsysteme, Komponenten, Klassen)
- Extenes Design (UI)

Implementation

- Beschreibung interessanter Implementaionsaspekte

- Verwendete Technologien
- Nebenläufigkeit
- Vorgehen beim Testing

Resultate

- Zielerreichung
- Auswertung Erfüllung der Anforderungen
- Projektmetriken

Conclusion

- Zusammenfassung
- Evaluation der Ergebnisse
- Zielerreichung/offene Punkte
- Ausblick, weiterführende Schritte

Project Management

- Vorgehen (Prozess, Reviews, Workflows, Qualitätssicherung)
- Projektplan, Vergleich ursprüngliche Planung, effektive Ausführung
- Zeiterfassung (Stunden pro Woche/Stunden pro Task-Kategorie, wie Implementation, Doku, Meeting, etc.)
  Hinweis: Keine Liniendiagramme für die Darstellung von Zeit/Arbeitsaufwand pro Woche

Glossary

- Kurze Erklärung der projektspezifischen Begriffe und Abkürzungen

Bibliography

- Referenzen der Quellen im Dokument

Appendices

- Relevante Anhänge
- Meeting Protokolle
- Vereinbarungen
- (SA/BA): Persönliche Berichte
- Entwicklerdokumentation
- User Dokumentation
- Sonstige Protokolle (z.B. von Usability Tests)

Generelle Hinweise

- Kapitelüberschriften folgen nie direkt aufeinander. Sprich zwischen Überschrift eines Kapitels und eines Subkapitels gibt es immer Text.
- Bei Top-Level-Kapiteln (Introduction, Analysis, bis Conclusion) beschreibt der erste Paragraph die Struktur des Kapitels und fasst zusammen welche Information in welchem Unterkapitel zu finden sind.
- Tabellen, Abbildungen und Code Listings verfügen jeweils über eine Beschriftung. Diese muss im Text referenziert werden.
- Aktive Formulierungen sind passiven vorzuziehen.

F.4 Leitfaden für BA und SA

Leitfaden für Bachelor- und Studienarbeiten

Bachelorstudiengang Informatik an der OST – Ostschweizer Fachhochschule

Erlassen durch die Studiengangleitung des Bachelorstudiengangs Informatik an der OST – Ostschweizer Fachhochschule in Ausführung von Art. 2 der Ausführungsbestimmungen des Bachelorstudiengangs Informatik an der OST – Ostschweizer Fachhochschule.

1. Zweck

1.1 Zweck des Leitfadens

Dieser Leitfaden gilt für Arbeiten, die für die Module Studienarbeit und Bachelorarbeit im Bachelorstudiengang Informatik erstellt werden. Er definiert

- die Abläufe bei der Vergabe der Studien- und Bachelorarbeiten,
- die Aufgaben der Betreuerinnen und Betreuer von Studien- oder Bachelorarbeiten sowie der Gegenleserinnen und Gegenleser als auch der Expertinnen und Experten
- Richtlinien für die Bewertung der Arbeiten

1.2 Zweck der Bachelor- und Studienarbeiten

Die Arbeiten sollen die selbständige Problemlösungsfähigkeit unter Anwendung ingenieurmässigen Methoden im Bereich Informatik nachweisen. Entsprechend umfasst die Arbeit einen konzeptionellen und theoretischen sowie einen praktischen Anteil.

Neben fachlichen und technischen Fähigkeiten nach den neuesten Erkenntnissen von Wissenschaft und Praxis sollen die Studierenden auch die Fähigkeit zur Reflexion, Vermittlung und Einordnung der eigenen Leistung in einen grösseren Kontext sowie ein Verantwortungsbewusstsein für wirtschaftliche, soziale und auf Nachhaltigkeit bezogene Fragestellungen nachweisen.

2. Administratives

2.1 Zulassung

Zur Studienarbeit wird zugelassen, wer die Kategorie «SE Practices» bestanden sowie zum Zeitpunkt der Anmeldung 90 Credits erzielt hat.

Zur Bachelorarbeit wird zugelassen, wer die Studienarbeit bestanden sowie zum Zeitpunkt der Anmeldung 120 Credits erzielt hat.

OST
Ostschweizer
Fachhochschule

2.2 Rahmentermine

Für Arbeiten, die im Herbstsemester stattfinden, gelten folgende Rahmentermine:

- KW 18 Freitag, 17 Uhr, Eingabe von Themenvorschlägen
- KW 22 Vorstellung der Themen und Eröffnung der Bewerbungsphase
- KW 27 Freitag, 17 Uhr, Bewerbungsschluss für Studierende
- KW 35 Montag, 12 Uhr, Veröffentlichung der Themenzuteilung
- KW 38 Beginn der Arbeit
- KW 51 Freitag, 17 Uhr, Abgabe der Studienarbeit
- KW 02 Freitag, 17 Uhr, Abgabe der Bachelorarbeit

Für Arbeiten, die im Frühjahrssemester stattfinden, gelten folgende Rahmentermine:

- KW 46 Freitag, 17 Uhr, Eingabe von Themenvorschlägen
- KW 50 Vorstellung der Themen und Eröffnung der Bewerbungsphase
- KW 01 Freitag, 17 Uhr, Bewerbungsschluss für Studierende
- KW 05 Montag, 12 Uhr, Veröffentlichung der Themenzuteilung
- KW 08 Beginn der Arbeit
- KW 22 Freitag, 17 Uhr, Abgabe der Studienarbeit
- KW 24 Freitag, 17 Uhr, Abgabe der Bachelorarbeit

Ausnahmen davon sind in begründeten Fällen auf Antrag an die Studiengangleitung möglich.

2.3 Betreuungspersonen

Bachelor- und Studienarbeiten werden von durch die Studiengangleitung zugelassene Betreuungspersonen betreut. Dies sind in der Regel Dozierende, die im Studiengang Informatik unterrichten und über die entsprechende Prozess- und Fachkompetenz verfügen, Arbeiten im Bereich Informatik zu betreuen.

Die Betreuungsperson kann bis zu 75% der Betreuung an eine Co-Betreuungsperson abgeben, bspw. wissenschaftliche Mitarbeitende. Die volle Verantwortung für die Arbeit bleibt bei der Betreuungsperson.

Betreuungspersonen werden bei Studienarbeiten mit 30h (0.8 SWS) je betreuter Studentin/betreutem Studenten vergütet, bei Bachelorarbeiten mit 37.4h (1 SWS). Dies umfasst den Aufwand für die Vorbereitung der Aufgabenstellung, die wöchentlichen Besprechungen, die Beurteilung der Arbeit und die Prüfung. Bei Bachelorarbeiten ist zudem die Arbeit als Gegenleserin/Gegenleser für jeweils eine andere Bachelorarbeit enthalten und wird nicht gesondert vergütet.

OST
Ostschweizer
Fachhochschule

2.4 Expertinnen und Experten

Bei Bachelorarbeiten schlägt die Betreuungsperson der/dem Studiengangleitenden eine Expertin oder einen Experten vor, die/der die Bachelorarbeit begutachtet. Expertinnen und Experten sollen mindestens über einen Master-Abschluss in Informatik oder ähnlichen Disziplinen bzw. über vergleichbare Berufserfahrung verfügen. Sie dürfen zu der entsprechenden Arbeit und daran beteiligten Personen in keinem geschäftlichen oder familiären Verhältnis stehen. Insbesondere dürfen sie nicht unmittelbar von den Ergebnissen der Arbeit profitieren, keine Angestellten des Industriepartners sein, usw.

Die Expertin / der Experte begutachtet die Arbeit unabhängig von der Betreuungsperson anhand der abgegebenen Dokumente und der Präsentation der Bachelorarbeit. Sie/er soll auch die Praxistauglichkeit der Resultate der Studierenden in die Beurteilung einfliessen lassen. Die Note der Expertin / des Experten wird schriftlich festgehalten. Können sich die Expertin / der Experte und die Betreuungsperson nicht auf eine gemeinsame Note einigen, gilt die Note der Betreuungsperson, die in diesem Fall eine Begründung für die Diskrepanz verfassen muss.

2.5 Gegenleserinnen und Gegenleser

Die Gegenleserin / der Gegenleser einer Bachelorarbeit wird durch die Studiengangleitung in den ersten Semesterwochen bestimmt. In der Regel handelt es sich dabei um eine Betreuungsperson einer anderen Bachelorarbeit im gleichen Semester. Die Aufgabe des Gegenlesers / der Gegenleserin besteht im Unterschied zum Experten / zur Expertin darin, eine weitere Meinung zur Note einzubringen, nicht jedoch eine eigene Bewertung vorzunehmen.

2.6 Rechte und allfällige Geheimhaltung

Das Urheberrecht an der Arbeit steht den beteiligten Studierenden gleichermassen zu.

Die Studierenden treten sämtliche Nutzungsrechte an der Arbeit und allen im Rahmen dieser Arbeit erzeugten Artefakten an die OST ab, insbesondere an Software, Bildern, Dokumenten, Audiodateien und Videodateien. Die OST gewährt allen Angehörigen der OST in der Regel vollumfängliche Nutzungsrechte daran.

Abweichend davon können Nutzungsrechte an bestimmten Artefakten teilweise oder vollständig von der Studiengangleitung eingeschränkt werden. Insbesondere können auf Antrag der Betreuungsperson abweichende Lizenzvereinbarungen genehmigt werden, bei Software-Artefakten beispielsweise eine Open-Source-Lizenz.

Die Studiengangleitung kann im Rahmen einer Arbeit, die mit Partnern durchgeführt wird, die Nutzungsrechte an Artefakten teilweise oder vollständig an diese Partner abtreten, auch exklusiv. Entsprechende Vereinbarungen müssen in schriftlicher Form durch die Partner unterzeichnungsfähig und vor der Ausschreibung der Arbeit bei der Studiengangleitung über die Betreuungsperson eingereicht werden.

Partner können mit der Betreuungsperson weitere besondere Bestimmungen vereinbaren, bspw. erhöhte Vertraulichkeit (Non-Disclosure Agreement). Die Partner anerkennen, dass die Arbeiten im Grundsatz öffentlich sind: Die Studierenden müssen über ihre Leistung unter Auslassung vertraulicher Details so berichten können, dass sie für eine nicht an der Arbeit beteiligte Fachperson im Wesentlichen nachvollziehbar ist; Bachelorarbeiten werden anlässlich ihrer Ausstellung einem allgemeinen Publikum präsentiert und über das eprints-Portal der OST veröffentlicht: https://eprints.ost.ch

Die Vereinbarungen sind von allen Beteiligten in der Regel vor Beginn der Arbeit, spätestens jedoch in der ersten Woche der Arbeit zu unterzeichnen. Zum Schutz der Studierenden dürfen danach Vereinbarungen nur noch mit Zustimmung der Studiengangleitung unterzeichnet werden.

Non-Compete-Klauseln sind für Bachelor- und Studienarbeiten im Studiengang Informatik grundsätzlich nicht zulässig.

3. Themen

3.1 Ausschreibung von Themen

Themen müssen immer von zugelassenen Betreuungspersonen ausgeschrieben werden.

Der Ausschreibe- und Vergabeprozess für die Studien- und Bachelorarbeiten wird vom Arbeitsverwaltungstool (AVT) unterstützt: https://avt.hsr.ch [1]

Alle Themen werden von den jeweiligen Betreuungspersonen im AVT ausgeschrieben und zum Stichtag freigeschaltet. Danach werden keine weiteren Themen ausgeschrieben. Bei Fragen zu einzelnen Themen können, wenn nichts anderes angegeben, die Ausschreibenden per Mail kontaktiert werden. Von Fragen ausserhalb dieses Prozesses, insbesondere vor Freischaltung der Ausschreibungen, sollten die Studierenden absehen, um unnötigen Mehraufwand zu vermeiden.

Die Mehrzahl der Themen wird von den Betreuungspersonen in einem kurzen Pitch unmittelbar vor der Freischaltung präsentiert. Die Teilnahme an dieser Veranstaltung ist für Ausschreibende und Studierende freiwillig, wird aber stark empfohlen.

3.2 Anforderungen an Themen

Themen müssen eine abgeschlossene Arbeit der Studierenden ermöglichen, die unabhängig von den Leistungen anderer bewertet werden kann. Sie müssen immer einen konzeptionellen und theoretischen Anteil umfassen; rein ausführende Arbeiten, z.B. nur Implementierung bzw. Testen, sind nicht zulässig. Ebenso sind auch rein theoretische Abhandlungen nicht zugelassen; es muss zumindest einen Proof-of-Concept geben.

Themen, die einen Interessenskonflikt bei der Betreuungsperson oder der OST auslösen, sind nicht zulässig. Ethisch bedenkliche Themen müssen dem Ethikausschuss der OST bekannt gemacht werden.

Bei jedem Thema müssen die fachlichen Anforderungen an die Studierenden benannt werden. Studierende, die sich auf ein Thema bewerben, erklären damit, dass sie die genannten Anforderungen erfüllen. Es ist zulässig, dass Studierende bestimmte Kenntnisse erst während der Durchführung erwerben. Die Entscheidung darüber obliegt der Betreuungsperson.

Sollten bei Arbeiten zusätzliche Verpflichtungen der Studierenden notwendig sein, müssen diese ihrem Charakter nach in der Ausschreibung erwähnt sein. (Bspw. «Studierende müssen ein NDA unterschreiben.» Das konkrete NDA muss dort dann aber noch nicht abgebildet sein.)

[1] Dieser URL wird sich demnächst ändern. Detaillierte Informationen werden zu gegebener Zeit kommuniziert.

OST Ostschweizer Fachhochschule

3.3 Themenvorschläge

Studierende bzw. Externe dürfen Themenvorschläge einbringen. Diese müssen bis zum oben angegebenen Termin beim Studiengangsekretariat eingereicht werden. Das Sekretariat macht die Vorschläge den Betreuungspersonen zugänglich.

Es gibt keine allgemeine Zusage, dass eingereichte Themenvorschläge ausgeführt werden. Eine notwendige Bedingung ist, dass sich eine zugelassene Betreuungsperson bereit erklärt, das Thema auszuschreiben.

3.4 Anmeldung und Bewerbung auf Themen

Studierende, die zur Bachelor- oder Studienarbeit zugelassen werden wollen, müssen sich wie gewöhnlich im entsprechenden Modul anmelden. Zusätzlich müssen sie sich selbst in Zweiergruppen organisieren und die Gruppe auf die im Arbeitsverwaltungstool ausgeschriebenen Themen bis zum entsprechenden Anmeldetermin bewerben.

In Ausnahmefällen ist es auf begründeten Antrag an die Studiengangleitung möglich, als Dreiergruppe oder allein ein Thema zu bearbeiten. Von Einzelarbeiten raten wir aber in den meisten Fällen ab.

Zugang zum AVT erhalten nur diejenigen Studierenden, deren Leistungen darauf hindeuten, dass sie die Zulassungsbedingungen erfüllen werden.

Studierende, die nicht zum vorzeitigen Vergabeprozess im AVT zugelassen werden, zu Semesterbeginn jedoch die geforderten Voraussetzungen erreichen, werden dann einer Arbeit zugeteilt. Ebenso werden Studierende, die zum Vergabeprozess zugelassen werden, die Zulassungsbedingungen jedoch zu Semesterbeginn nicht erfüllen, von ihrem zugewiesenen Thema wieder entfernt. Verbleibende Teampartnerinnen oder Teampartner können dann in Absprache mit der Betreuerin oder dem Betreuer entweder am selben Thema weiterarbeiten oder sich einer anderen Gruppe anschliessen.

3.5 Zuweisung der Themen

Nach Ablauf der Bewerbungsfrist priorisieren die Betreuungspersonen die eingegangenen Bewerbungen auf ihre Themen. Diese Priorisierungen sind intern und werden nicht veröffentlicht. Anhand der Priorisierungen und unter Berücksichtigung weiterer Faktoren wie bspw. Auslastung der Betreuungspersonen besprechen die Professorinnen und Professoren im Departement Informatik eine faire Zuteilung der Gruppen und Themen.

Die finale Zuweisung erfolgt durch die Studiengangleitung und wird zum angegebenen Termin veröffentlicht. Vor diesem Termin kann kein Anspruch auf Durchführung einer Arbeit abgeleitet werden.

Gruppen können auch Themen zugewiesen bekommen, auf die sie sich nicht beworben haben, insbesondere dann, wenn sich die Gruppe auf keine Themen beworben hat oder alle Themen, auf die sie sich beworben hat, bereits anderen Gruppen zugewiesen wurden. Fühlt sich eine Gruppe in diesem Fall nicht in der Lage, das Thema angemessen zu bearbeiten, soll sie unverzüglich mit der Studiengangleitung Kontakt aufnehmen, um eine Lösung zu finden.

Themen, die von Studierenden eingebracht und einer Betreuungsperson ausgeschrieben wurden, werden in der Regel diesen Studierenden zugewiesen. Ein solches Thema wird nie anderen Studierenden zugewiesen.

Bachelorarbeiten, die thematisch an die Studienarbeit derselben Gruppe anknüpfen, werden prioritär zugewiesen.

In den beiden letztgenannten Fällen (von Studierenden eingebrachten Themen oder Fortsetzungsarbeiten mit derselben Gruppe) wird das Thema für die entsprechende Gruppe reserviert. Andere Reservationen sind aus Gründen der Gleichbehandlung nicht zulässig. Insbesondere dürfen keine Vorabsprachen zwischen Betreuenden und Studierenden über Reservationen getroffen werden, und es gibt keine Regel «first come, first served».

4. Aufgabenstellung

4.1 Bekanntgabe der Aufgabenstellung

Die konkrete Aufgabenstellung wird den Studierenden in der ersten Semesterwoche von der Betreuungsperson schriftlich mitgeteilt und im ersten wöchentlichen Beratungsgespräch (s. 5.2) mündlich vorgestellt.

Bei Unklarheiten dürfen die Studierenden eine Präzisierung der Aufgabenstellung verlangen, die ihnen bis zum Beratungsgespräch der zweiten Semesterwoche ausgehändigt werden muss.

Erfolgt die Bekanntgabe der Aufgabenstellung in dieser Weise, lässt sich kein Anspruch auf Verlängerung des Abgabetermins ableiten. Weitere Verspätungen der Aufgabenstellung hingegen berechtigen zu einer entsprechenden Verlängerung des Abgabetermins.

4.2 Inhalt der Aufgabenstellung

Die Aufgabenstellung soll ein Problem beschreiben, das mit ingenieurwissenschaftlichen Methoden und Verfahren, Konzepten und/oder Technologien der Informatik gelöst werden soll. Den Studierenden muss genügend Freiraum für die eigene Lösungsfindung zur Verfügung stehen.

Die Aufgabenstellung muss mindestens die folgenden Punkte umfassen:

- Titel der Arbeit
- Problembeschrieb
- Formulierung eines konkreten Auftrags
- Umfang und Form der erwarteten Resultate bei Abgabe der Arbeit, wenn abweichend von 5.5
- Beteiligte Personen (Studierende, Betreuungsperson, weitere Beteiligte soweit bekannt, namentlich Industriepartner)
- Anfangs- und Abgabetermin
- Zulässige Hilfsmittel und weitere Betreuung (z.B. durch externen Partner)

4.3 Allgemeine Grundsätzliche Bestandteile der Aufgabenstellung (AGBs)

Die folgenden Teile sind impliziter Bestandteil jeder Aufgabenstellung und müssen nicht explizit aufgeführt werden. Sie dürfen durch die Aufgabenstellung in ihrem Wesen nicht verändert werden.

Die Arbeit muss im Wesentlichen selbständig erfolgen. Quellen und Beiträge anderer sind in jedem Fall deutlich und unmissverständlich als solche zu kennzeichnen. Insbesondere müssen bei Arbeiten, die in starkem Bezug zu eigenen oder fremden Vorarbeiten stehen, diese klar von der eigenen Leistung im Rahmen der Bachelor- oder Studienarbeit abgegrenzt werden.

Es soll ingenieurwissenschaftlich vorgegangen werden. Dazu gehört insbesondere:

a. Annahmen und Entscheide sind zu plausibilisieren sowie deren Implikationen aufzuzeigen.
b. Fehlende Kenntnisse müssen selbständig erarbeitet werden, auch ausserhalb der Informatik, soweit im Rahmen der Arbeit zumutbar.
c. Fehlende, unzulängliche oder widersprüchliche Anforderungen dürfen nicht als Begründung eines unbrauchbaren Ergebnisses dienen, sondern müssen mit der Betreuungsperson und ggf. den Partnern erörtert werden.
d. Software-Artefakte und ähnliches sind nach dem Stand der Technik zu erstellen und zu dokumentieren.

Die Verantwortung für das Ergebnis der Arbeit liegt ungeachtet der Hinweise der Betreuungsperson bei den Studierenden.

Die Arbeit und ihr Wert müssen in das Fachgebiet eingeordnet werden. Insbesondere müssen verwandte Arbeiten in angemessenem Umfang dargestellt werden (bspw. gleicher Ansatz / andere Domäne oder anderer Ansatz / gleiches Problem). Es soll auf Aspekte gesellschaftlicher, nachhaltiger und/oder wirtschaftlicher Natur eingegangen werden.

5. Durchführung

5.1 Generelles

Die Studierenden sollen grundsätzlich selbständig die vorgegebene Aufgabenstellung nach bestem Wissen und Gewissen bearbeiten. Die Verantwortung für die Arbeitsergebnisse liegt immer bei den Studierenden. Insbesondere treffen die Studierenden alle Entscheidungen während der Arbeit in eigener Verantwortung, auch wenn die Betreuungsperson jederzeit Hinweise einbringen kann. Bei Unklarheiten oder offensichtlichen Unsinnigkeiten ist Rücksprache mit der Betreuungsperson zu nehmen. Keinesfalls dürfen Studierende solche als Begründung für unzulängliche Arbeitsergebnisse verwenden.

Aufgabe der Betreuungsperson ist es, die Studierenden bei der selbständigen Arbeit zu führen und z.B. auf mögliche Probleme hinzuweisen. Insbesondere soll sie die Studierenden frühzeitig warnen, wenn die Ergebnisse der Arbeit erheblich von den Erwartungen abweichen.

5.2 Wöchentliches Beratungsgespräch

Die Studierenden haben ein Anrecht auf ein wöchentliches Beratungsgespräch, das durchschnittlich eine Stunde dauert. Die Studierenden entscheiden, ob sie das Beratungsgespräch wahrnehmen wollen. Die Betreuungsperson kann davon abweichend ein Gespräch in jeder zweiten Woche verpflichtend ansetzen.

Die Betreuungsperson kann verlangen, zu Beginn des Beratungsgesprächs über den Stand der Arbeit informiert zu werden.

Die Studierenden sollen über die Gespräche Protokoll führen und dieses der Betreuungsperson in digitaler Form zukommen lassen.

5.3 Zwischenpräsentation

In den Semesterwochen 7 bis 9 führen die Studierenden bei Bachelorarbeiten eine Zwischenpräsentation durch, in der sie den die Aufgabenstellung und den Stand ihrer Arbeiten vorstellen, insbesondere die Überlegungen zum Konzept. Die Teilnahme ist für Betreuungsperson und Gegenlesende verbindlich, für Expertinnen und Experten empfohlen.

Ziel ist es, von den Teilnehmenden Feedback zur Arbeit zu erhalten. Die Zwischenpräsentation fliesst nicht in die Bewertung ein, soll von den Studierenden aber dennoch ordentlich vorbereitet und durchgeführt werden.

Spätestens an der Zwischenpräsentation soll der Termin für die Bachelorprüfung festgelegt werden. Der Termin der Zwischenpräsentation wird zu Beginn des Semesters vereinbart.

5.4 Aufwand und Abgabetermin

Der Aufwand für die Bachelor-Arbeit beträgt 360 Stunden, bei der Studienarbeit 240 Stunden (30 Stunden je Credit). Die Studierenden sollen diese Arbeitszeit in etwa einhalten und darüber Buch führen. Die Betreuungsperson entscheidet, ob und welchem Detaillierungsgrad die Aufzeichnungen Eingang in den Schlussbericht finden.

Die Arbeitszeit ist bei Studienarbeiten gleichmässig im Semester zu verteilen (240h / 14w ≈ 17h/w), was in der Semesterplanung berücksichtigt werden sollte. Die Arbeitszeit der Bachelorarbeiten teilt sich in die Semesterzeit und zwei zusätzliche Blockwochen im Anschluss daran, in denen Vollzeit gearbeitet werden sollte (40h/w). Im Semester ergibt sich bei Bachelorarbeiten somit eine Wochenarbeitszeit von (360h – 80h) / 14w = 20h/w.

In den Frühjahrsferien (KW 15) soll nicht an Studien- oder Bachelorarbeiten gearbeitet werden.

Studierende können bei der Studiengangleitung aus entschuldbaren Gründen und unter Darstellung der geleisteten Arbeitsstunden einen Antrag auf Verlängerung des Abgabetermins stellen. Der Antrag muss soweit möglich unmittelbar nach Eintreten der entsprechenden Umstände gestellt werden, spätestens jedoch eine Woche vor dem regulären Abgabetermin. Er wird über die Betreuungsperson gestellt, die dazu auch Stellung nimmt.

5.5 Umfang und Form der Abgabe

Bei Bachelorarbeiten müssen folgende Dokumente erstellt werden:

- Projektplan
- Bericht
- Poster
- Abschlusspräsentation

- Broschüren-Abstract

Projektplan, Bericht, Poster und Abschlusspräsentation werden in jeweils einem separaten PDF-File abgegeben; die Abschlusspräsentation erst zur Bachelorprüfung. Die Studierenden bestätigen im Bericht der Bachelor-Arbeit, dass sie die Arbeit selbstständig und nur unter Benützung der angeführten Quellen und Hilfsmittel angefertigt haben. Sämtliche Entlehnungen sind durch Quellenangaben festzuhalten.

Der Bericht enthält einen wissenschaftlichen Abstract, der für Fachleute in diesem Gebiet die wesentlichen Leistungen der Arbeit umreisst. Zusätzlich soll der Bericht ein Management Summary oder ein Lay Summary enthalten. Ersteres richtet sich dabei an Entscheidungsträger in grösseren Organisationen – fokussiert also eher auf den wirtschaftlichen Nutzen –, letzteres an ein sehr allgemeines Publikum mit Schwerpunkt auf leichte Verständlichkeit und einen gesellschaftlichen Nutzen (siehe Artikel in populären Tageszeitungen).

Vom wissenschaftlichen Abstract zu unterscheiden ist der Broschüren-Abstract, der in der BA-Broschüre erscheinen wird und über https://abstract.rj.ost.ch erstellt wird. Ebenso wie das Poster richtet er sich an ein interessiertes Laienpublikum, sollte aber etwas tiefer gehen als ein Lay Summary. Im Gegensatz zu diesem ist er auch mit Überschriften gegliedert und enthält Bildmaterial.

Alle weiteren Artefakte sollen in einem ZIP-File mit angemessener Verzeichnisstruktur abgegeben werden.

5.6 Bachelorprüfung

Bei Bachelorarbeiten führen Betreuerin/Betreuer, Gegenleserin/Gegenleser und Expertin/Experte mit den Studierenden eine Prüfung durch. In der Regel findet diese auf einem Campus der OST unter Ausschluss der Öffentlichkeit statt.

Die Prüfung besteht aus zwei Teilen: der Abschlusspräsentation und der mündlichen Fachprüfung. Vertretungen der Partner der Arbeit dürfen immer an der Abschlusspräsentation teilnehmen. Auf Wunsch der Studierenden können im Einverständnis mit der Betreuungsperson weitere Personen an der Abschlusspräsentation teilnehmen. Die Fachprüfung findet immer unter Ausschluss weiterer Personen statt.

Der Zeitrahmen für die Präsentation kann zwischen der Betreuungsperson und den Studierenden frei vereinbart werden; wenn nichts vereinbart wird, soll die Präsentation etwa 30 Minuten dauern. Die Fachprüfung soll pro Person etwa 10 Minuten in Anspruch nehmen. Die Studierenden einer Gruppe werden in der Regel gemeinsam geprüft. Auf Wunsch der Studierenden können auch separate Prüfungen vorgenommen werden.

In der Abschlusspräsentation sollen die Studierenden ihre Ergebnisse und Lösungswege für Fachpersonen darstellen. Die Fachprüfung dient dazu zu überprüfen, ob die Studierenden eine entsprechende Tiefe im Fachgebiet der Bachelorarbeit erreicht haben.

6. Bewertung

6.1 Grundsätze der Benotung

Die Leistungen werden in voneinander unabhängigen Teilbereichen bewertet. Für jede Arbeit wird die Gewichtung der Teilbereiche definiert. Für jeden Teilbereich werden die Anforderungen an die Noten 4, 5 und 6 definiert. Die Noten entsprechen dabei folgenden Grundgedanken:

- Note 6: Herausragende, publikationswürdige Leistung, die einer grösseren Öffentlichkeit ohne weiteres präsentiert werden kann.
- Note 5: Übliche Leistung, die innerhalb einer technisch geprägten Arbeitsumgebung erwartet werden darf, z.B. innerhalb einer Entwicklungsabteilung eines Unternehmens.
- Note 4: Mindest-Anforderungen an eine professionelle Arbeitsweise im Rahmen des Stands der Technik.

Leistungen, die zwischen den Anforderungen dieser Noten liegen, können mit entsprechend interpolierten Viertelnoten bewertet werden. Die Anforderungen einer besseren Note umfassen immer die Anforderungen an eine niedrigere Note.

6.2 Notenfindung

Verantwortlich für die Benotung ist immer die Betreuungsperson. Expertinnen/Experten nehmen davon unabhängig eine eigene Bewertung vor. Das Bewertungsschema ist grundsätzlich verbindlich; begründete Abweichungen davon können vor Beginn der Arbeit mit der Studiengangleitung vereinbart werden und müssen den Studierenden im Rahmen der Aufgabenstellung mitgeteilt werden.

Weichen die Bewertungen von Expertin / Experte und Betreuungsperson voneinander ab, sollen im Gespräch Gründe dafür gesucht und überprüft werden. Kommt auch dann keine Einigung zustande, gilt die Bewertung der Betreuungsperson, aber die Studiengangleitung ist entsprechend zu informieren.

6.3 Bekanntgabe der Note

Die Note wird den Studierenden über unterricht.rj.ost.ch bekannt gegeben. Die Betreuungsperson darf vor diesem Zeitpunkt angeben, ob die Arbeit als bestanden/nicht bestanden gilt, sowie vorwarnen, wenn das Risiko auf eine ungenügende Note besteht. In diesen Fällen soll auch die Studiengangleitung informiert werden.

6.4 Teile der Studienarbeiten

Studienarbeiten werden nach vier Teilen bewertet:

1. Organisation und Durchführung: 20%
2. Formale Qualität des Berichts: 20%
3. Analyse, Entwurf und Auswertung: 20%
4. Technische Umsetzung: 40%

OST Ostschweizer Fachhochschule

6.5 Teile der Bachelorarbeit

Bachelorarbeiten werden nach fünf Teilen bewertet:

1. Organisation und Durchführung: 10%
2. Formale Qualität des Berichts: 10%
3. Analyse, Entwurf und Auswertung: 20%
4. Technische Umsetzung: 40%
5. Bachelorprüfung: 20%

6.6 Bewertungsraster Organisation und Durchführung

Die Note 6 wird erreicht, wenn die Studierenden in höchstem Masse selbständig arbeiten und aussergewöhnlich hohen Einsatz zeigen, von sich auf Stakeholder zugehen und deren Interessen in der Arbeit berücksichtigen.

Die Note 5 wird erreicht, wenn die Studierenden das Projekt proaktiv durchführen, indem sie bspw. selbständig Aufgaben identifizieren oder auf auftretende Problemstellungen aufmerksam machen und angemessene Lösungsvorschläge anbieten.

Die Note 4 wird erreicht, wenn die Studierenden in der Lage sind, den Projektplan aufzustellen, ihm zu folgen und regelmässig zu aktualisieren. Sie sollen sich gegenüber Betreuenden sowie Projektpartnerinnen und -partnern professionell verhalten und über den Projektfortschritt unterrichten.

6.7 Bewertungsraster Formale Qualität des Berichts

Die Note 6 wird erreicht, wenn der Bericht den formalen Anforderungen an eine technische oder wissenschaftliche Publikation genügt oder einem externem Fachpublikum vorgestellt werden könnte. Dazu gehören neben einer gehobenen Sprache einwandfreie Rechtschreibung und Grammatik sowie ein ansprechendes Layout. Die Literaturrecherche zeugt von einer eingehenden Beschäftigung mit dem aktuellen Stand im Fachgebiet.

Die Note 5 wird erreicht, wenn der Bericht einer internen Publikation entsprechen würde; kleinere sprachliche und formale Mängel sind akzeptabel. Der Bericht soll logisch strukturiert und verständlich geschrieben sein sowie sämtliche wichtigen Aspekte der Arbeit darstellen. Wo nötig sollen Grafiken und Tabellen verwendet werden, die komplexe Sachverhalte unterstützend erklären. Die Literaturrecherche bezieht sich neben Grundlagen auch auf weiterführende Literatur.

Die Note 4 wird vergeben, wenn die Arbeit in grossen Teilen dokumentiert wurde. Die Sprache hält gutes umgangssprachliches Niveau ein, sprachliche Fehler sind in gewissem Masse tolerabel. Verweise auf verwandte Arbeiten oder Grundlagen sind an wichtigen Stellen angegeben.

6.8 Bewertungsraster Analyse, Entwurf und Auswertung

Die Note 6 wird erreicht, wenn die Studierenden eine Kompetenz in der Analyse der Aufgabenstellung, dem Lösungsentwurf und der Auswertung der Ergebnisse ihrer Arbeit auf wissenschaftlichem Niveau nachweisen. Der Bericht enthält eine umfangreiche Darstellung der der für die Arbeit relevanten Domäne und stellt die Problematik unter Berücksichtigung verschiedener Aspekte dar, bspw. technische,

wirtschaftliche oder gesellschaftliche. Die Studierenden stellen an allen kritischen Orten verschiedene Varianten in entsprechender Tiefe vor und motivieren ihre Entscheidung anhand wissenschaftlicher Kriterien. Sie bewerten ihre Arbeit kritisch und selbstreflektierend, ordnen diese in den Stand der Wissenschaft und Technik ein und geben einen fundierten Ausblick auf Folgearbeiten.

Die Note 5 wird erreicht, wenn der Bericht einer technischen Dokumentation entspricht, die Fachleuten verständlich ist. Domäne und Probleme, die aus der Aufgabenstellung resultieren, werden eingänglich dargestellt und bei Bedarf mit Grafiken erläutert. Die Studierenden finden für wesentliche Probleme verschiedene Lösungsvarianten und erklären schlüssig ihre Entscheidung für eine davon. Sie bewerten die Resultate ihrer Arbeit in Bezug auf die Aufgabenstellung und verwandte Arbeiten und zeigen verschiedene interessante Möglichkeiten der Weiterverwertung ihrer Resultate auf.

Die Note 4 wird erreicht, wenn der Bericht die Beschreibung der folgenden Punkte enthält: Aufgabenstellung in der Domäne, Lösungskonzept mit Begründung, Darstellung der Resultate und Abgleich mit der Aufgabenstellung sowie vereinzelte Anknüpfungspunkte für weitere Arbeiten.

6.9 Bewertungsraster Technische Umsetzung

Die Note 6 wird erreicht, wenn die Arbeit höchsten Ansprüchen genügt. Die Studierenden müssen

- die Aufgabenstellung übererfüllt haben, bspw. weitere, selbst definierte Use Cases oder nicht-funktionale Anforderungen
- die Fehlerfreiheit ihrer Arbeit nachweisen, bspw. durch formale Methoden oder umfangreiche automatisierte Tests (Verifikation)
- die Nützlichkeit ihrer Arbeit nachweisen, bspw. durch User-Experience-Tests, Einordnung in einen wissenschaftlichen Kontext oder eine Wirtschaftlichkeitsanalyse (Validierung)
- alle Artefakte frei von technischer Schuld (*technical debt*) abgeben, die in dieser Form ohne weiteres releast werden könnten

Die Note 5 wird erreicht, wenn die Arbeit solide umgesetzt wurde. Die Studierenden müssen

- die Aufgabenstellung vollständig erfüllt haben (oder nachweisen, dass das nicht mit vertretbarem Aufwand möglich ist)
- plausibel machen, dass die Arbeit grösstenteils fehlerfrei ist, z.B. durch eine angemessene Zahl an manuellen und/oder automatisierten Tests
- darstellen, dass und wie die Arbeit ihren Zweck erfüllt
- alle Artefakte in einem Zustand abgeben, der es erlaubt diese mit wenig zusätzlichem Aufwand zu veröffentlichen

Die Note 4 wird erreicht, wenn die Arbeit gerade noch den Ansprüchen der Professionalität genügt. Die Studierenden müssen

- die Aufgabenstellung ihrem Wesen nach erfüllt haben
- ausreichend viele manuelle und/oder automatisierte Tests durchgeführt und dokumentiert haben
- alle Artefakte so abgeben, dass andere Fachpersonen mit vertretbarer Einarbeitungszeit daran weiterarbeiten können

6.10 Bewertungsraster Bachelorprüfung

Die Note 6 wird erreicht, wenn die Studierenden sich und ihre Arbeit optimal vorstellen können. Die Studierenden weisen eine fachliche Tiefe nach, die sie als Expertinnen bzw. Experten im Fachgebiet der Arbeit ausweisen. Die Arbeit wird auf einem nachvollziehbaren Abstraktionsniveau und in einem angemessenen Zeitrahmen vorgestellt. Die Präsentationsmaterialien sind professionell gestaltet und gegliedert, die Studierenden reden frei und flüssig auf sprachlich hohem Niveau.

Die Note 5 wird erreicht, wenn die Studierenden ihre Arbeit vollumfänglich präsentieren, so dass Personen mit einem Informatik-Grundverständnis ohne weiteres den Inhalt der Arbeit und ihre Einordnung in das entsprechende Fachgebiet verstehen können. Sie können Zusammenhänge in der Arbeit und im Fachgebiet darstellen und vermitteln den Eindruck, dass sie als Fachleute in diesem Gebiet selbständig produktiv sein können. Die Präsentation ist ansprechend gestaltet, grösstenteils logisch gegliedert und die Studierenden haben einen guten Redefluss auf einem angemessenen sprachlichen Niveau.

Die Note 4 wird erreicht, wenn die Studierenden ihre Arbeit so darstellen können, dass es Fachleuten anhand der vorbereiteten Materialien gelingt, die wesentlichen Punkte der Arbeit nachzuvollziehen. Die Studierenden weisen nach, dass sie über hinreichende Kenntnisse im Fachgebiet verfügen, um in diesem tätig werden zu können. Die Präsentationsweise ist auf studentischem Niveau, Sätze werden oft abgelesen, der Redefluss kann noch teilweise stocken.

7. Anhang

7.1 Auszug aus den Ausführungsbestimmungen

Art. 17 Durchführung von Bachelor- oder Studienarbeiten mit Partnern

[1] Bachelor- oder Studienarbeiten können in Zusammenarbeit mit Partnern ausserhalb und innerhalb der OST durchgeführt werden.

[2] Die Arbeit ist keine Auftragsarbeit, die Partner sind nicht Empfänger einer verabredeten Leistung. Es dürfen gegenüber den Partnern keine Zusicherungen hinsichtlich der Resultate der Arbeit abgegeben werden, insbesondere nicht über Umfang, Art und Qualität derselben.

[3] Die Partner können bei der Formulierung der Aufgabenstellung involviert werden. Sie können den Studierenden bei der Beantwortung von Fragen zur Verfügung stehen und sowohl auf Mängel und Fehler als auch auf mögliche Lösungen hinweisen. Der Beitrag der Partner muss von den Studierenden vollständig angegeben werden und wird bei der Bewertung der Eigenleistung der Studierenden berücksichtigt.

[4] Angehörige der OST und Studierende dürfen keinen unmittelbaren finanziellen oder geldwerten Vorteil aus der Arbeit ziehen. Mit der Arbeit unmittelbar zusammenhängende Spesen können von den Partnern vergütet werden, insbesondere Fahrt- und Verpflegungskosten sowie Material- und Lizenzkosten.

[5] Zwischen den Partnern und dem Studiengang kann eine Vereinbarung getroffen werden, die Geheimhaltung, Nutzungsrechte und Patentfragen regeln. Eine solche Vereinbarung ist den beteiligten Studierenden spätestens 2 Wochen vor Beginn der Arbeit mitzuteilen. Akzeptieren die Studierenden Verpflichtungen, die ihnen aus dieser Vereinbarung entstehen nicht, können sie eine Alternativarbeit verlangen. Die OST übernimmt grundsätzlich keine Haftung für Verstösse gegen die Vereinbarung seitens der Studierenden oder Betreuungspersonen.

[6] Soll basierend auf den Ergebnissen der Arbeit ein Patent angemeldet werden, müssen die Partner eine schriftliche Vereinbarung mit dem Studiengang treffen, die insbesondere eine Regelung zur angemessenen Vergütung des Anteils der Studierenden und Angehörigen der OST enthält.

F.5 Informationen über die SA Abgabe HS21

OST
Ostschweizer
Fachhochschule

Informationen zur SA-Abgabe

Der späteste Abgabetermin für die Studienarbeit ist der 24.12.21, 17 Uhr.

Alle Informationen, Termine, Vorlagen und zu unterschreibende Erklärungen finden Sie auf Teams.

Erfassung Abstract über das Online-Tool:

Die SA-Abstracts werden über das Online-Tool erfasst. Für diese Abstracts sind Sie aufgefordert, uns eine kurze Zusammenfassung Ihrer Arbeit sowie eigene Bilder zur Verfügung zu stellen. Die Abstracts werden auf der OST-Webseite publiziert.

Erfassung des Abstracts:
Die Erfassung der Abstracts erfolgt mit einem speziellen Online-Tool, https://abstract.rj.ost.ch/
Loggen Sie sich dazu über das OST Netzwerk mit Ihrem OST Account ein oder von extern über Cisco AnyConnect Client
https://intranet.rj.ost.ch/it-informatikdienste/netzwerkzugang/zugang-von-extern-vpn/ Bitte kontrollieren Sie die bereits eingegebenen Metadaten (Titel, Angaben Industriepartner) und melden allfällige Unstimmigkeiten rasch möglichst.

Abgabetermin:
Wir bitten Sie bis zum 21.12.21 um Erfassung des Abstracts und Weitergabe an den Betreuer/die Betreuerin. Der Betreuer/die Betreuerin und/oder das Studiengangsekretariat haben die Möglichkeit, Ihnen den Abstract zu Korrekturzwecken zurückzusenden. Kontrollieren Sie also auch nach der Abgabe des Abstracts regelmässig Ihr OST-E-Mail-Konto. Der definitive und vom Examinator geprüfte Abstract muss bis spätestens am 23.12.21 an das Studiengangsekretariat weitergeleitet werden.

Publikation und Archivierung: SA Abgabe über https://avt.i.ost.ch/

Für die Archivierung erfolgt die Abgabe der SA/BA über https://avt.i.ost.ch/
Logen Sie sich dazu mit Ihrem OST Account ein. Von extern über VPN.

Für die Abgabe der SA und BA https://avt.i.ost.ch/ benötigen wir folgende Dateien:
Vorlagen finden Sie auf Teams

- **Archivdatei mit den von Betreuern und Experten verlangten Unterlagen.**
  Das Archiv darf nicht Passwort geschützt sein.
  Archiv Format: Zip.

- **Eigenständigkeitserklärung** (als PDF Scan)

- **Urheber- und Nutzungsrechte** (als PDF Scan)

- **Publizierbare Version der Arbeit für** http://eprints.ost.ch/
  - **Bericht als PDF (nur für Arbeiten die nicht vertraulich sind)**
    Die Publikation auf eprints erfolgt durch den Studiengang. Aus Datenschutzgründen soll dieser Bericht keine persönlichen Daten wie Sitzungsprotokolle, Daten von Industriepartnern, Eigenständigkeitserklärungen, Unterschriften etc. enthalten.
  - **Plain-Text Abstract**
    Der Plan-Text Abstract wird auch bei vertraulichen Arbeiten publiziert.

- **Einverständniserklärung** (als PDF Scan)
  Damit wir Ihre Arbeit von aus auf http://eprints.ost.ch/ publizieren, muss eine von allen Teammitgliedern unterzeichnete Einverständniserklärung vorliegen (PDF Scan).

  Bei vertraulichen Arbeiten, wird nur der Abstract publiziert.

- **Poster A0**

Abgabe Bericht in gedruckter Form:

Einige Betreuer möchten den Bericht in gedruckter Form. Der Betreuer legt die Anzahl der abzugebenden Berichte fest. Der Aufbau des Berichts ist im Leitfaden beschrieben. Diesen finden Sie auf Teams.

Druck Bericht:

Wir geben Ihnen die Möglichkeit die vom Betreuer verlangten abzugebenden Berichte auf Kosten des Studiengangs zu drucken. Sie können in den folgenden Zeitfenstern im Studiengangsekretariat 1.173 vorbeikommen (bitte USB-Stick mit PDF-Version mitbringen):

- **Donnerstag, 23-12.21, 13 – 15 Uhr**

OST
Ostschweizer
Fachhochschule

Ihre Checkliste für Abgabe von SA/BA-Dokumenten

Für den Betreuer

Der Betreuer/die Betreuerin legt fest, in welcher Form ihm/ihr der Bericht abgegeben werden soll sowie die Anzahl abzugebender gedruckten Berichte.

Dokumente, die auf https://avt.i.ost.ch/ **hochgeladen werden:**
Die Vorlagen finden Sie auf Teams.

Für die Archivierung:

- ❑ **ZIP Archivdatei mit den von Betreuern und Experten verlangten Unterlagen**
- ❑ Eigenständigkeitserklärung (PDF Scan)
- ❑ Urheber- und Nutzungsrechte (PDF Scan)
- ❑ Für A0 Poster
  Die PPT-Postervorlage auf Teams dient als Vorlage für das Poster, welches im Format A0 als PDF abgegeben werden muss.

Zusätzlich für die Publikation auf : http://eprints.ost.ch/ (Publikation erfolgt durch SG I)

- ❑ **Plain-Text Abstract**
  Der Plain-Text Abstract wird auch bei vertraulichen Arbeiten publiziert.
- ❑ **Bericht als PDF (nur für Arbeiten, die nicht vertraulich sind)**
  Das PDF enthält keine persönlichen Daten und Source Codes.
- ❑ **Einverständniserklärung (PDF Scan)**
  Für die Publikation auf http://eprints.ost.ch/

Für die Erfassung des Abstracts für die Publikation auf der OST Webseite
Die Erfassung erfolgt über das Abstract-Tool. Die BA-Abstracts werden ebenfalls für Bachelorarbeitsbroschüre verwendet.

- ❑ **Abstract mit DAB-Tool**

29.11.21 FUCL

F.6 Termine Studierende HS21 Studienarbeiten

Studienarbeiten: Termine Herbstsemester 2021/22

bis 23.04.2021	Eingabe Themen von Studierenden und externen Partnern
26.05.2021, 12 Uhr	Präsentation der Themen
26.02.2021, 13 Uhr	Die Themen werden unter https://avt.hsr.ch veröffentlicht.
bis 04.07.2021	Die Studierenden, die aufgrund ihrer Vorleistungen zum Vergabeprozess im AVT zugelassen sind, können sich auf die ausgeschriebenen Themen im AVT bewerben
23.08.2021	Veröffentlichung der Themenzuteilung
bis 17.09.2021	Einrichten der Arbeitsplätze in den Labors und Vorbesprechung mit Betreuer
20.09.2021	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch den Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf Teams zur Verfügung.
21.12.2021	Die Studierenden erfassen den Abstract im Online Tool http://abstract.rj.ost.ch/ und geben den Abstract zur Kontrolle an ihren Betreuer/ihre Betreuerin frei.
23.12.2021	Der Betreuer/die Betreuerin gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei.
24.12.2021	Abgabe des Berichts an den Betreuer/die Betreuerin und hochladen aller Dokumente auf https://archiv-i.hsr.ch/Overview/All bis 17 Uhr