Automatisiertes Proctoring digitaler Prüfungen mit Machine Learning

Studienarbeit

Studiengang Informatik
OST - Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Semester: Herbstsemester 2023

Autoren: Nicolas Gattlen
Kevin Pfister

Betreuer: Prof. Dr. Frieder Loch
Version: 1.0
Datum: 22.12.2023

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Inhaltsverzeichnis

Abstract

Aufgabenstellung

Das Ziel dieser Arbeit bestand darin, ein Tool oder eine Methode zu entwickeln, die es ermöglicht, mittels spezialisierter Software zu erkennen, ob ein Prüfungsteilnehmer oder eine Prüfungsteilnehmerin während einer Online-Prüfung betrügerische Handlungen begeht. Der Einsatz einer solchen Software bietet den Vorteil, dass Online-Prüfungen effektiv durchgeführt werden können. Zudem ermöglicht es, dass Programmierprüfungen nicht mehr manuell, sondern in einem kontrollierten und überwachten Umfeld absolviert werden können. Der Fokus dieser Arbeit lag speziell auf der Überwachung der Offline-Aktivitäten der Prüfungsteilnehmenden. Dies umfasst die Beobachtung von auffälligen Körperbewegungen und Sprachaktivitäten.

Vorgehen / Technologien

Unser Ansatz zielte darauf ab, die Benutzerfreundlichkeit unserer Software sowohl für Studierende als auch für Lehrende zu maximieren und einen strukturierten Prozess zu etablieren. Der Ablauf beginnt damit, dass der Studierende die Software aktiviert, die mittels einer Webcam ein Video aufzeichnet und über ein ausgewähltes Mikrofon den Ton erfasst. Nachdem die Prüfungsaufgaben gelöst wurden, wird die Software beendet. Der Studierende erhält dadurch eine Video- und Audiodatei, die er eigenständig an einen schulischen Server übermitteln kann. Nachdem der Schulserver alle Audio und Videodateien erhalten hat. Kann der Professor mittels einer Webseite die Video und Audiodateien analysieren. Während der Analyse wird bei jedem Prüfungsteilnehmer die Video und Audiodatei analysiert. Dabei wurden bestimmte Kriterien von uns ausgewählt auf welche sich die Videoanalyse stützen soll.

Gesichtserkennung: In jedem Frame des Videos soll überprüft werden, ob genau eine Person anwesend ist. Dies ist wichtig, da Prüfungen in der Regel Einzelarbeiten sind. Sollte keine oder mehr als eine Person erkannt werden, wird dies als Betrugsversuch gewertet.

Munderkennung: Das System muss ständig den Mund erkennen können. Dies dient dazu, heimliche Gespräche aufzudecken, auch wenn das Mikrofon während der Prüfung absichtlich ausgeschaltet ist. Ein Betrugsversuch wird deklariert, sobald der Mund sich öffnet, um die Spracherkennung zu schützen.

Kopfneigungserkennung: Die Kopfposition wird kontinuierlich überwacht, um auffällige Bewegungen, die auf einen zweiten Bildschirm oder ein externes Gerät hinweisen könnten, als Betrugsversuche zu identifizieren.

Iriserkennung: Diese Technik dient dazu, ungewöhnliches Augenschillen zu erkennen. Sie soll Prüflinge identifizieren, die zwar ihren Kopf nicht bewegen, aber dennoch nicht auf die Prüfungsaufgaben schauen.

Resultat

Derzeit ist eine vollständig automatisierte Proctoring-Lösung noch nicht realisierbar. Menschliches Eingreifen bleibt notwendig, um zu entscheiden, ob in bestimmten Situationen Betrug vorliegt. Unsere Software hat jedoch den Analyseprozess von Audio- und Videodateien erheblich effizienter gestaltet. Als nächsten Schritt zur Verbesserung der Software könnte die Erfassung von Online-Aktivitäten in Betracht gezogen werden.

Danksagung

Ein Grosses Dankeschön geht an folgende Institutionen, Programme und Personen:

- Professor Doktor Loch für die ausgezeichnete Betreuung und Unterstützung während des Projekts. Er hat uns alle Fragen immer sofort beantwortet und war eine grosse Hilfe während dem Projekt.
- Joel Stohler, Sabrina Forster, Fabian Freitag, Jason Benz, Josip di Benedetto und Valerio Falvella für das Testing unserer Software.
- Der Ostschweizer Fachhochschule für die Ausschreibung dieser Arbeit und die bereitstellung der Server.
- ChatGPT für das korrigieren unserer Texte.

Teil I

Management Summary

Management Summary

Problemstellung

Unser Ziel war es, eine zuverlässige proctoring Lösung zu entwickeln, die es Schülern ermöglicht, Prüfungen von zu Hause aus online zu absolvieren, ohne dass der Verdacht des Betrugs für die Lehrperson entsteht. Durch den Einsatz von Online-Aufnahmen und automatisierten Skripten können Betrugsversuche identifiziert und anschliessend vom Lehrer analysiert werden. Das Hauptziel ist es, die Zeit, die für die Video- und Audioanalyse benötigt wird, zu reduzieren. Ein automatisiertes Skript hebt alle potenziellen Betrugsversuche hervor, sodass die Lehrperson eigenständig entscheiden kann, ob in einem bestimmten Fall tatsächlich betrogen wurde.

Lösungsansatz

Als Lösung haben wir eine Webseite entwickelt, auf der Schülerinnen und Schüler ihre Aufnahmen automatisch nach der Prüfung hochladen können. Diese Videos werden auf der Webseite angezeigt und können durch das Klicken eines Buttons auf potenzielle Betrugsversuche hin analysiert werden. Unsere Analysetools basieren auf Sensoren, die sowohl Video- als auch Audiodateien auswerten. Jede Videodatei wird beispielsweise daraufhin überprüft, ob während der gesamten Prüfung nur eine Person vor der Webcam ist. Zusätzlich führen wir eine Iriserkennung durch, die kontinuierlich die Iris verfolgt und auffällige Bewegungen registriert. Ähnliches gilt für die Überwachung der Kopfneigung. Ausserdem haben wir einen Spracherkennungssensor integriert, der die Audiodaten analysiert und gesprochene Worte in Text umwandelt. Wenn das Mikrofon stummgeschaltet ist, überprüft das System, ob der Mund sichtbar bleibt und löst Alarm aus, sobald sich der Mund öffnet. Mit der Kombination all dieser Sensoren können Lehrpersonen selbst entscheiden, ob während der Prüfung betrogen wurde. Zudem verkürzt sich die Analysezeit, da nicht mehr das gesamte Video angesehen werden muss.

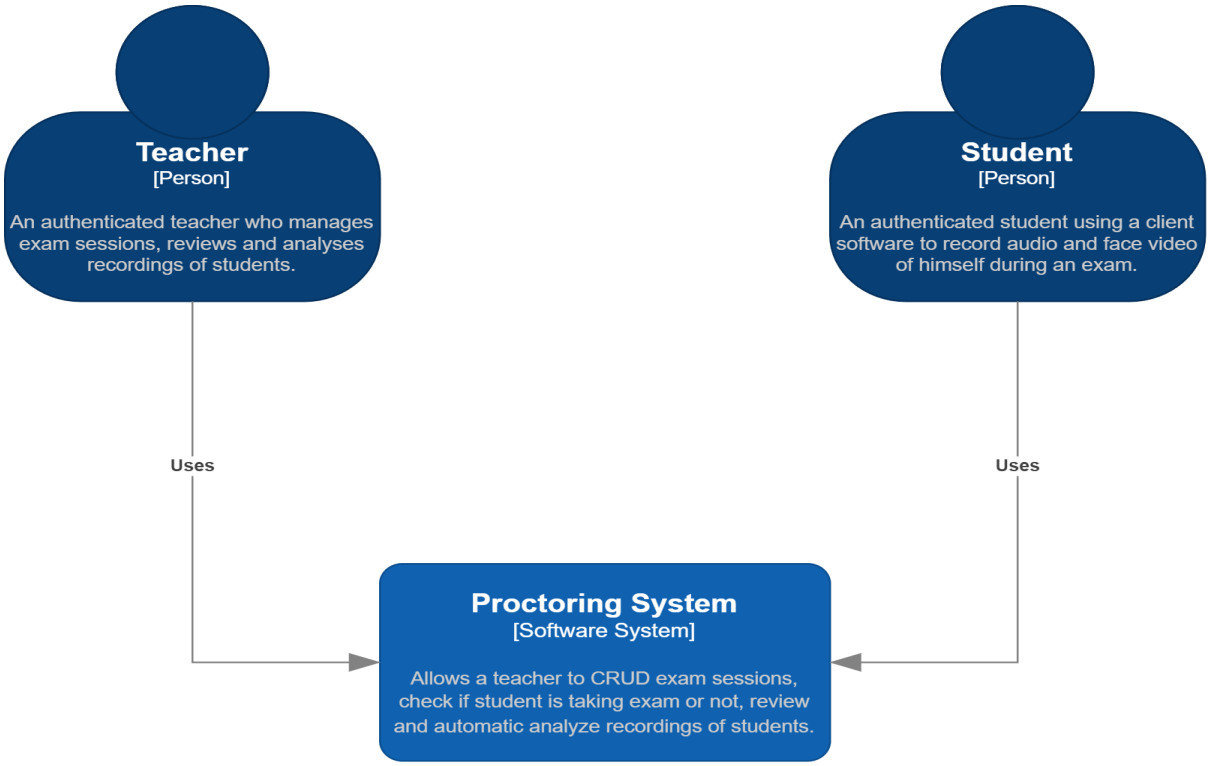


Abbildung 1: Proctoring System Lösung

Resultate

Die Webseite funktionierte erfolgreich und war in der Lage, Betrugsversuche zu erkennen. Allerdings stiessen wir auch auf Probleme, die nur begrenzt beeinflussbar sind. Wir stellten beispielsweise fest, dass sowohl die Lichtverhältnisse im Video als auch der Abstand zur Kamera entscheidend für die Zuverlässigkeit der Analyseergebnisse sind. Darüber hinaus gibt es Spielraum für Verbesserungen bei einigen Parametern der Sensoren, um die Genauigkeit noch weiter zu steigern. Insgesamt ist es uns jedoch gelungen, Betrugsfälle zu identifizieren und die Analysezeit für Lehrpersonen deutlich zu reduzieren.

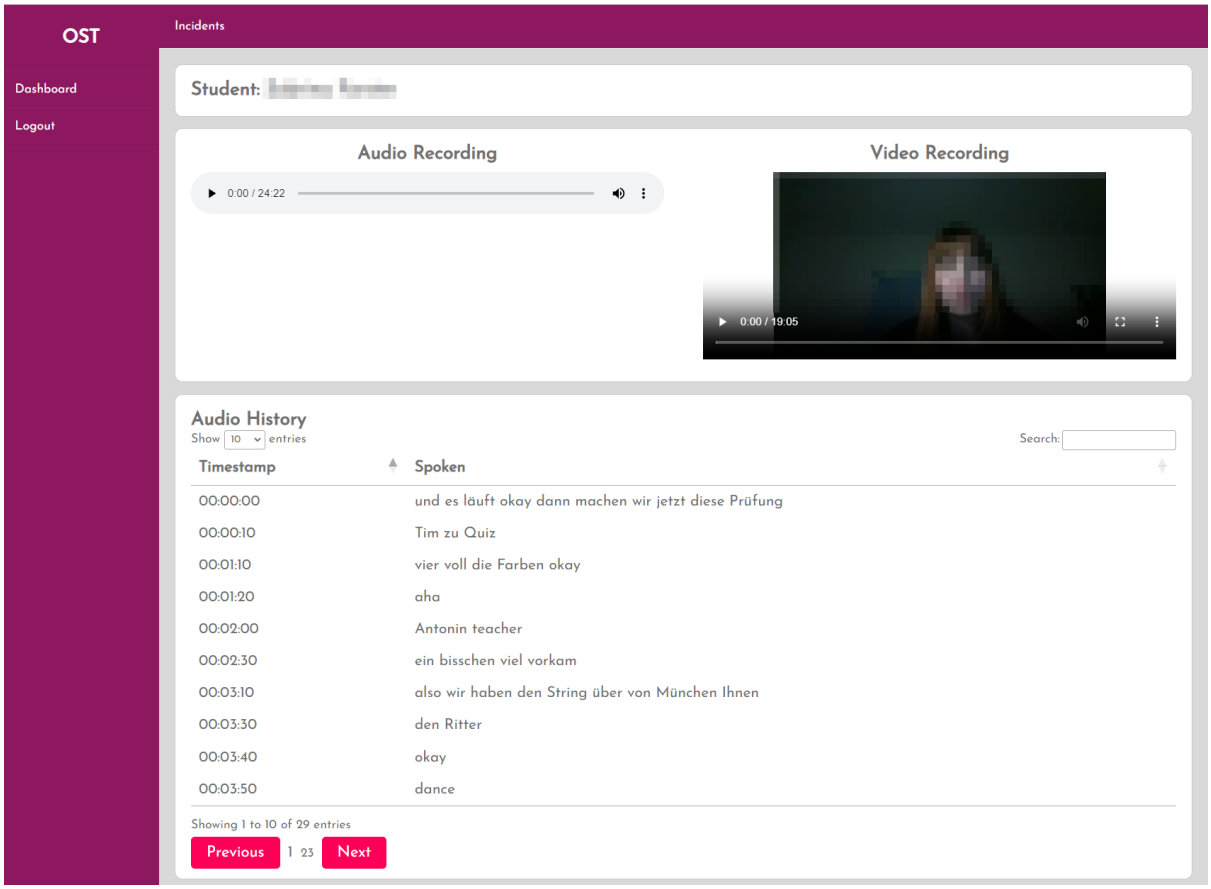


Abbildung 2: Bild der fertigen Video- und Audio Analyse

Fortsetzung des Projektes

Stand heute, am 21. Dezember 2023, ist es noch ungewiss, ob die Weiterentwicklung dieses Projekts fortgesetzt wird. Die möglichen Verbesserungen umfassen die Optimierung der Sensoren sowie die Überarbeitung des Client-Systems. Ziel ist es, Studierenden in Zukunft die Möglichkeit zu bieten, ihre Aufnahmen direkt im Browser zu erstellen, anstatt dafür ein externes Programm herunterladen zu müssen.

Teil II

Produkt Dokumentation

Kapitel 1: Voraussetzungen

1.1 Use Case Beschreibung

In Bildungseinrichtungen, insbesondere an der OST (Ostschweizer Fachhochschule), soll in Zukunft vermehrt auf digitalen Prüfungen gesetzt werden. Proctoring-Software spielt hierbei eine entscheidende Rolle, um die Integrität der Prüfung zu gewährleisten. Dabei sollen in einem Beispielszenario die Schülerin oder der Schüler die Möglichkeit bekommen, Programmierprüfungen auf einem Online System wie zum Beispiel Moodle zu schreiben.

Durch die Durchführung von Programmierprüfungen in einer Online-Umgebung, wie beispielsweise Moodle, anstatt auf Papier, erhalten Informatikstudierende die Möglichkeit, ihre Prüfungen in einer realistischeren und praxisnäheren Umgebung abzulegen. Ein wesentlicher Vorteil dieser Methode ist, dass Studierende ihre Programme vor der Abgabe testen und gegebenenfalls verbessern können. Dieser Prozess bietet den Studierenden ein sichereres Gefühl hinsichtlich der Qualität und Funktionsfähigkeit ihrer Arbeit. Da dieser Ansatz auch die reale Arbeitsweise von Softwareentwicklern widerspiegelt, bei der das Testen und debuggen von Code ein integraler Bestandteil des Entwicklungsprozesses ist. Es bietet dem Studierenden auch eine realere Erfahrung in Hinsicht auf die Softwareentwicklung und die zukünftige Arbeitswelt.

Während der Prüfung wird der Student jedoch von einem Proctoring-System so überwacht, dass es nicht möglich ist zu betrügen, um die Chancengleichheit zu gewährleisten. Wichtig ist, dass die Studierenden darüber informiert sind, dass die Proctoring-Software im Hintergrund läuft. Jedoch sollte diese die Leistung der Studierenden nicht beeinträchtigen, etwa durch Softwareabstürze oder lange Wartezeiten während der Prüfung.

Nach der Prüfung können Lehrende die Aufnahmen der Studierenden einsehen und selbst beurteilen, ob Betrug vorlag. Die verschiedenen Arten von Betrugsversuchen und Auffälligkeiten werden in einem späteren Abschnitt detaillierter beschrieben.

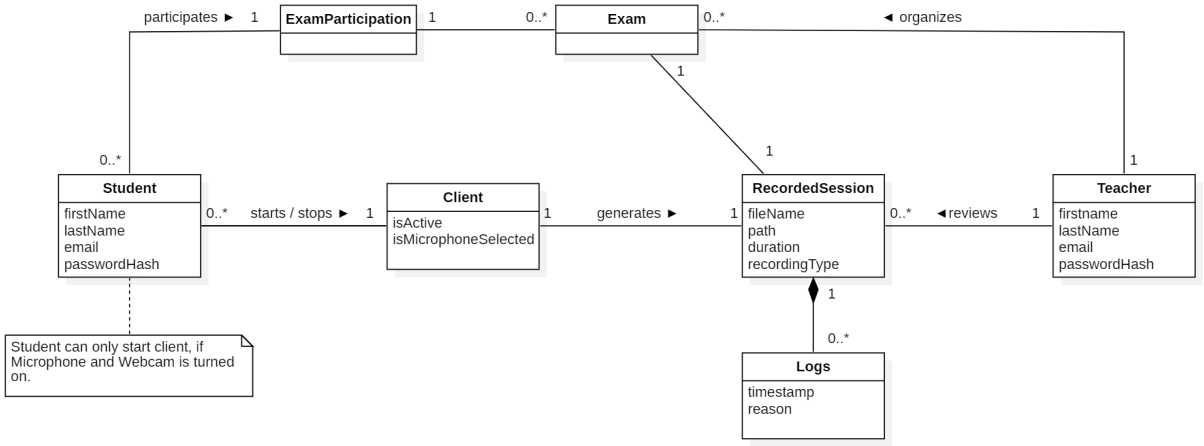


Abbildung 1.1: Domain Model Prüfungsverlauf

1.2 Akteure

1.2.1 Hauptakteur: Schüler/Schülerin

Die Schülerin oder der Schüler ist ein Endbenutzer der Software. Die Rolle besteht darin, problemlos seine Prüfung schreiben zu können. Im besten Fall läuft die Software im Hintergrund und hindert den Schüler nicht daran, seine Prüfung zu schreiben. Im Falle eines Betrugsversuches erstellt die Software eine Momentaufnahme und protokolliert dies intern. Der Schüler bekommt jedoch nichts davon mit.

1.2.2 Hauptakteur: Lehrperson

Die Lehrperson ist ebenfalls ein Endbenutzer der Software. Die Aufgabe für die Lehrperson ist es, den Überblick nach der Prüfung zu behalten. Dabei soll es keine Rolle spielen über welches System eine Prüfung durchgeführt wird. Sie soll nach der Prüfung die Möglichkeit haben, Auffälligkeiten zu finden. Mittels einer Logfunktion ist es der Lehrperson möglich, nur die gekennzeichneten Auffälligkeitsmomente zu betrachten und nicht die ganze Aufnahme.

1.2.3 Hauptakteur: Organisation

Da jede Prüfung von einer Schule oder einer anderen Bildungseinrichtung durchgeführt wird. Ist die Organsisation der Akteur, welche die Prüfungsaufnahmen erhält und verwaltet.

1.2.4 Supportakteur: Kamera und Mikrofon

Damit während der Prüfung Aufnahmen möglich sind, werden Supportsysteme benötigt. Dazu gehört zum einen die Webcam, die dazu dient, das Gesicht und die Gestik der Schülerin oder des Schülers zu überwachen. Das Mikrofon dient dazu, Geräusche aufzunehmen. Da die Umgebung während einer Prüfung ruhig sein sollte, soll das Mikrofon als Sensor dienen, welcher sofortige Auffälligkeiten erfasst.

1.3 Use Case Diagramm

In diesem Abschnitt wird das Use Case Diagramm erstellt und es wird vereinzelt auf die einzelnen Use Cases eingegangen.

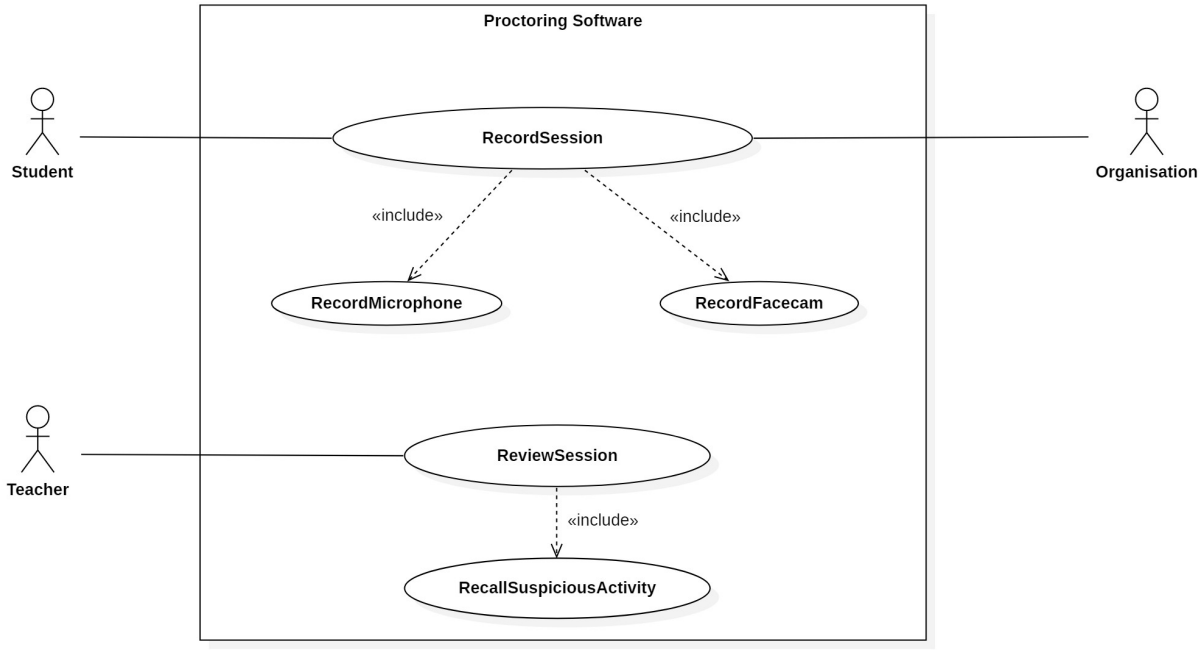


Abbildung 1.2: Use Case Diagramm

RecordSession: Der Schüler oder die Schülerin startet vor dem Lösen der Prüfung die Software und initiiert somit eine Recordsession. Diese Recordsession ist für die Überwachung einzelner Supportactors verantwortlich. Das beinhaltet die Überwachung der Hardwarekomponenten Kamera und Mikrofon.

RecordFacecam: Um Offline-Betrugsversuche vorzubeugen, soll die Kamera aufzeichnen. Auf diese Weise sollen Auffälligkeiten wie eine zweite Person im Hintergrund oder auffällige Kopfbewegungen erkannt werden. Die Software soll jedes Mal einen Log-Eintrag machen, sollte sie eine zweite Person im Hintergrund erkennen, eine auffällige Kopfbewegung feststellen (zum Beispiel langes Wegschauen vom Bildschirm), ein auffälliges Schielen, eine Bewegung des Mundes oder ein Nichterkennen des Mundes.

RecordMicrophone: Um eine möglichst realistische Prüfungsumgebung zu schaffen, soll das Mikrofon aufgezeichnet werden. Da während einer Prüfung normalerweise nicht gesprochen werden darf, soll die Software einen Log-Eintrag machen, sollte das Mikrofon eine Stimme erkennen. Dadurch wird Betrug durch Online- und Offline-Kommunikation verhindert.

ReviewSession: Die Lehrperson hat die Möglichkeit, alle Aufnahmen nach Abschluss der Prüfung durchzugehen. Dadurch soll im Nachhinein festgestellt werden können, ob es Betrugsfälle gegeben hat.

RecallSuspiciousActivity: Da während der Aufnahme Log-Einträge für auffälliges Verhalten erstellt werden, hat die Lehrperson die Möglichkeit, diese Stellen während einer Aufnahme zu überprüfen. Dadurch wird der Lehrperson die Arbeit erleichtert, da sie sich nicht die gesamten Aufnahmen ansehen muss.

1.4 Prüfungsverlauf

In diesem Abschnitt wird beschrieben, wie eine Prüfung mittels der zu entwickelnden Proctoring-Software ablaufen soll. Der Schüler oder die Schülerin muss eine Programmierprüfung ablegen, bei der das Bildungsinstitut festlegt, ob die Prüfung zu Hause oder in der Schule stattfindet. Die Prüfung wird dabei über ein Online-Prüfungssystem wie zum Beispiel Moodle durchgeführt, bei dem der Schüler die Möglichkeit hat seinen Code zu testen und debuggen. Bevor die Prüfung beginnt, muss der Schüler oder die Schülerin die Proctoring-Software starten. Beim Starten der Software ist eine Anmeldung erforderlich. Anschliessend muss das Fach der Prüfung sowie ein Mikrofon ausgewählt werden. Bei der Auswahl der Kamera wird darauf geachtet, dass die Standardkamera ausgewählt wird, um die Verwendung virtueller Kameras zu verhindern. Nachdem alle Komponenten ausgewählt wurden, kann der Schüler den Aufnahme-Knopf drücken, und die Software startet die Aufzeichnung. Während dieser Zeit kann der Schüler seine Prüfung online im Browser schreiben. Die Software zeichnet dabei Video und Audio über Kamera und Mikrofon auf.

Nachdem der Schüler oder die Schülerin die Prüfung abgeschlossen hat, sendet er oder sie die Aufnahmen an einen Schulserver. Auf dem Schulserver werden Logdateien erstellt, in denen Auffälligkeiten protokolliert werden. In der Nachbearbeitung überprüft die Lehrperson die Aufnahmen und überwacht die Protokolle. Die Protokolle sollen der Lehrperson dabei helfen, mögliche Unregelmässigkeiten zu erkennen. Dies spart der Lehrperson Zeit, da sie nicht die gesamte Aufnahme jedes Schülers oder jeder Schülerin ansehen muss, sondern nur die Protokolle, die auf Auffälligkeiten hinweisen

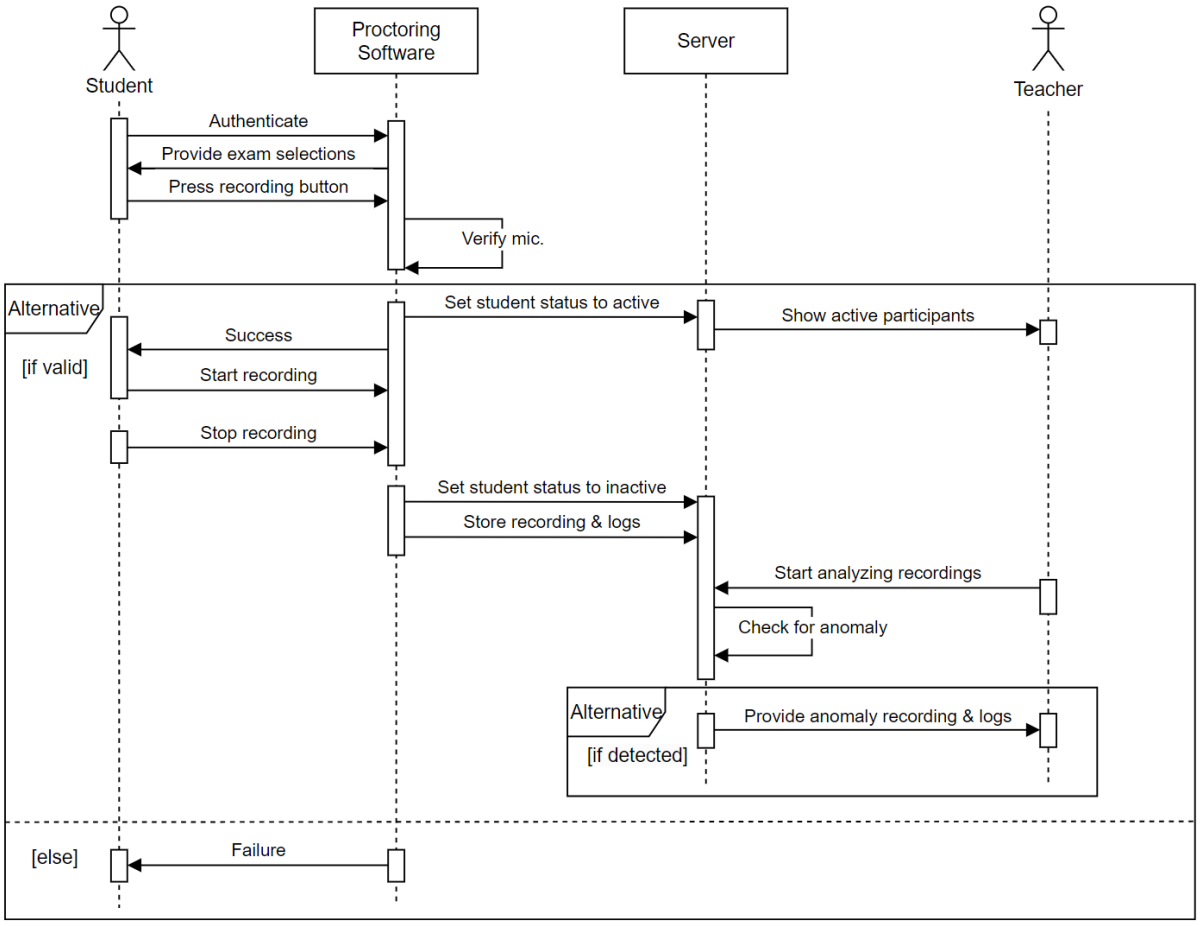


Abbildung 1.3: Sequenzdiagramm Prüfungsverlauf

1.5 Nicht-funktionale Anforderungen

In diesem Abschnitt lesen Sie über die Nicht-funktionale Anforderungen. Als Orientierung für unsere Voraussetzungen bedienen wir uns bei "Boehms Software Characteristics Tree", um eine genaue Kategorisierung unserer Voraussetzungen zu machen.

Tabelle 1.1: NFR-1

ID	**NFR-1**
Thema	End User kann die Software in kurzer Zeit starten.
Voraussetzung	Performance und Zeitverhalten
Wichtigkeit	Hoch
Beschreibung	Das Starten der Software soll möglichst effizient sein. Zum Start gehört das Eingeben des Namens + Hardware Verification. Der Schüler soll den Fokus nicht auf den Client legen müssen sondern auf seine Prüfung.
Überprüfbarkeit	Start der Software soll mit Login und Zeitpunkt der Aufnahme bei funktionstüchtigem Gerät nicht länger als 3 Minuten dauern.
Status	Erfüllt

Tabelle 1.2: NFR-2

ID	**NFR-2**
Thema	Die Software muss Betrugsfälle erkennen können.
Voraussetzung	Zuverlässigkeit
Wichtigkeit	Sehr Hoch
Beschreibung	Die Software soll erkennen wenn kein oder mehr als ein User vor der Kamera sind.
Überprüfbarkeit	Im Bezug auf die Konfusionsmatrix, soll eine Genauigkeit von 60 Prozent erreicht werden.
Status	Nicht Erfüllt, Genauigkeit von 44 Prozent erreicht. Siehe Auswertung

Tabelle 1.3: NFR-3

ID	**NFR-3**
Thema	Die Software muss Betrugsfälle erkennen können
Voraussetzung	Zuverlässigkeit
Wichtigkeit	Sehr Hoch
Beschreibung	Die Software soll erkennen wenn sich die Kopfneigung um mehr als 10 Prozent zum Referenzwert(wird alle 30 Frames neu berechnet) verändert.
Überprüfbarkeit	Im Bezug auf die Konfusionsmatrix, soll eine Genauigkeit von 60 Prozent erreicht werden.
Status	Erfüllt, Genauigkeit von 89 Prozent.

Tabelle 1.4: NFR-4

ID	**NFR-4**
Thema	Die Software muss Betrugsfälle erkennen können.
Voraussetzung	Zuverlässigkeit
Wichtigkeit	Sehr Hoch
Beschreibung	Die Software soll erkennen wenn in einer Aufnahme gesprochen wurde.
Überprüfbarkeit	Im Bezug auf die Konfusionsmatrix, soll eine Genauigkeit von 60 Prozent erreicht werden.
Status	Erfüllt, Genauigkeit von 98 Prozent.

Tabelle 1.5: NRF-5

ID	**NFR-5**
Thema	Die Software muss Betrugsfälle erkennen können.
Voraussetzung	Zuverlässigkeit
Wichtigkeit	Sehr Hoch
Beschreibung	Die Software soll erkennen wenn in einer Aufnahme geschielt wird oder die Augen nicht auf die Prüfung gerichtet sind.
Überprüfbarkeit	Im Bezug auf die Konfusionsmatrix, soll eine Genauigkeit von 60 Prozent erreicht werden.
Status	Erfüllt, Genauigkeit von 78 Prozent.

Tabelle 1.6: NRF-6

ID	**NFR-6**
Thema	Der Hardware Verification Check muss bei funktionstüchtigen Geräten funktionieren.
Voraussetzung	Benutzerfreundlichkeit und Zugänglichkeit
Wichtigkeit	Sehr Hoch
Beschreibung	Bei Notebooks und Desktop Pc's soll der Verification Test funktionieren, vorausgesetzt man besitzt eine Webcam und ein Mikrofon.
Überprüfbarkeit	Der Hardware verification check muss bei funktionstüchtigen Geräten funktionieren. Die Erfolgsrate muss bei mindestens 80 Prozent liegen.
Status	Erfüllt

Tabelle 1.7: NFR-7

ID	**NFR-7**
Thema	Nach dem Stop einer Aufnahme, soll die Aufnahme an einen externen Server geschickt werden.
Voraussetzung	Kompatibilität und Interoperabilität
Wichtigkeit	Mittel
Beschreibung	Nachdem der Schüler die Aufnahme stoppt, soll die Aufnahme automatisch an einen externen Server geschickt werden
Überprüfbarkeit	Aufnahmen sollen in 95 Prozent der Fälle ankommen. Falls das nicht möglich ist, besitzt der Schüler die Aufnahme und kann sie an die Lehrperson senden. Die Lehrperson kann diese anschliessend hochladen.
Status	Erfüllt

Tabelle 1.8: NFR-8

ID	**NFR-8**
Thema	Datenschutz bei der Übertragung von Aufnahmen.
Voraussetzung	Datensicherheit und Datenschutz
Wichtigkeit	Gering
Beschreibung	Bei der Übertragung der Aufnahmen an dem Server muss der Datenschutz gewährleistet sein. Dies umfasst die Verschlüsselung der Daten während der Übertragung sowie die Sicherstellung, dass keine unbefugten Dritten Zugriff auf die Daten erhalten.
Überprüfbarkeit	Die Vewendung von SFTP (Secure File Transfer Protokoll) gewährleistet eine sichere Art der Datenübetragung, die durch die Verschlüsselung der Daten und sichere Authentifizierungsmethoden den Datenschutz verbessert.
Status	Erfüllt

Kapitel 2: Recherche

2.1 Wissenschaftliche Literatur

2.1.1 An Automated Online ExamProctoring System [KYM+22]

Einführung Im Bericht "An Automated Online Exam Proctoring System" wird das Problem des Betrugs bei Online-Prüfungen in Bildungseinrichtungen erläutert. Aufgrund von Betrugsbedenken während Online-Prüfungen bevorzugen Bildungseinrichtungen, die Prüfung nicht online durchzuführen, sondern auf andere Prüfungsmethoden zurückzugreifen. Laut einer Studie von Dr. McCabe hat eine hohe Anzahl amerikanischer Studenten während Online-Prüfungen betrogen. Um das Problem zu lösen, haben die Autoren ein webbasiertes Prüfungsaufsichtssystem namens ProctorEx entwickelt, das verschiedene Erkennungsalgorithmen verwendet. Das Programm ermöglicht es, verdächtige Aktivitäten während der Prüfung zu überwachen und in Echtzeit Warnungen an den Dozenten zu senden. Dadurch wird das Risiko von Betrug bei Online-Prüfungen reduziert und Bildungseinrichtungen haben die Möglichkeit, die Prüfung online durchzuführen. Unsere Studienarbeit verfolgt ähnliche Ziele wie der Bericht, indem sie sich mit Methoden zur Bekämpfung von Betrug bei Online-Prüfungen auseinandersetzt und Lösungen zur Verbesserung der Prüfungsaufsicht untersucht.

Softwarevergleich Im Bericht werden drei beliebte Proctoring-Softwarelösungen verglichen, die AI-Technologien verwenden. Die genannten Proctoring-Softwarelösungen sind:

- ProctorU
- Proctorio
- ExamSoft

Alle diese Softwarelösungen erfassen Bewegungen und nutzen Bilderkennungsalgorithmen, um mögliche verdächtige Aktivitäten zu erkennen. Ein gemeinsames Merkmal aller Softwarelösungen ist, dass die Prüfungsaufnahmen für eine spätere Überprüfung durch einen Dozenten angesehen werden, anstatt verdächtige Aktivitäten in Echtzeit zu überprüfen. Trotz ihrer nützlichen Funktionen weisen die Softwarelösungen auch gemeinsame Nachteile auf. Studierende beklagen sich über Latenzprobleme, da die Software Zeit benötigt, um die Überwachung zu starten. Zudem reagieren die Programme sehr sensibel selbst auf kleinste Geräusche, was zu unnötigen Warnmeldungen führen kann. Des Weiteren werden Kosten- und Datenschutzaspekte thematisiert. Die gespeicherten Aufnahmen stellen nicht nur eine finanzielle Belastung für Bildungseinrichtungen dar, sondern auch für die Studierenden, da ihre Privatsphäre während der Prüfung stark eingeschränkt wird. In unserem Fall wird der ethische Aspekt ignoriert.

Features Insgesamt wurden vier Features in ProctorEx implementiert. Im Folgenden werden die einzelnen Features aufgelistet und beschrieben, wie sie implementiert wurden, welches Vorgehen dabei angewendet wurde und welche Probleme auftraten.

Tabelle 2.1: Features von ProctorEx

Feature	Beschreibung
Gesichtserkennung	ProctorEx verwendet den OpenCV Deep Neural Network Algorithmus für die Gesichtserkennung mit einer Genauigkeit von 80 Prozent oder höher. Das System erkennt Gesichter anhand von vordefinierten Modellen und zeichnet Begrenzungsrahmen um erkannte Gesichter. Aufgrund dieses optimierten Deep Neural Network Algorithmus kann eine schnelle und präzise Gesichtserkennung durchgeführt werden. Eine Herausforderung besteht jedoch darin, dass bei unterschiedlichen Lichtverhältnissen Schwierigkeiten auftreten können, was die Genauigkeit beeinträchtigen könnte.
Blickverfolgung	ProctorEx isoliert die Augen anhand von Gesichtslandmarken und überwacht die Bewegung der Iris. Das Programm verwendet binäre Bildverarbeitung und OpenCV Funktionen, um die Position und Bewegung der Iris zu verfolgen. Diese Verfolgung der Iris ermöglicht es festzustellen, ob Studierende nach rechts, links, oben oder nach unten schauen. Auch hier kann die Genauigkeit der Iriserkennung beeinträchtigt werden, wenn sich die Lichtverhältnisse oder die Kopfhaltung ändern.
Kopfhaltungsschätzung	Das Programm verwendet fünf charakteristische Punkte im Gesicht, um die Kopfhaltung zu schätzen. Dabei nutzt ProctorEx die solvePNP-Funktion von OpenCV, um die Euler-Winkel zu extrahieren und zu überprüfen, ob Studierende den Kopf zur Seite drehen. Durch diese Funktion kann die Genauigkeit bei der Erkennung von Veränderungen in der Kopfhaltung verbessert werden, beispielsweise wenn der Student sich vom Bildschirm wegdreht. Die Herausforderung hierbei ist, dass komplexe Kopfbewegungen zu ungenauen Schätzungen führen können.
Browseraktivität	Das System überwacht Browseraktivitäten, wie das Wechseln von Tabs oder das Aufteilen des Bildschirms. ProctorEx verwendet Algorithmen, um Browseraktivitäten zu überwachen und verdächtige Aktivitäten zu erkennen. Es ermöglicht die Identifizierung von betrügerischen Handlungen im Zusammenhang mit der Bildschirmnutzung. Allerdings kann das System auf harmlose Aktivitäten empfindlich reagieren, was zu Fehlalarmen führen könnte.

Evalution Im Projekt wurde ein UAT (User Acceptance Testing) durchgeführt, bei dem ausgewählte Benutzer das Programm getestet haben, um die Benutzerfreundlichkeit zu bewerten und Feedback zu sammeln. Es wurden drei Benutzer ausgewählt. Der erste Tester identifizierte Fehler, die behoben wurden, und gab anschliessend Feedback zur Verbesserung der Benutzeroberfläche. Der zweite Tester bemängelte die Geschwindigkeit bei schlechter Internetverbindung, woraufhin die Webcam-Bildrate reduziert wurde, um die Systemleistung zu verbessern. Der letzte Tester lieferte wertvolle Einsichten für zukünftige Verbesserungen.

Abschluss ProctorEx ermöglicht Gesichtserkennung, Blickverfolgung, Kopfhaltungsschätzung und Browserüberwachung. Positiv anzumerken ist die Echtzeitüberwachung ohne Drittanbieter-Intervention sowie die Benutzerfreundlichkeit für Lehrer und Studierende.

Die Herausforderungen liegen in den Webcam-Abhängigkeiten, Datenschutzbedenken, der Genauigkeit der Erkennungsalgorithmen und möglichen Belastungen für Studierende durch Fehlbeschuldigungen. Einige Verbesserungspunkte beinhalten die technische Zuverlässigkeit, die Entwicklung von Webcam-Alternativen und die Optimierung der Privatsphäre.

2.1.2 An Intelligent System For Online Exam Monitoring [PNSB16]

Einführung Das Amrita E-Learning Research Lab (AERL) hat sich mit einer ähnlichen Aufgabe befasst wie wir in unserer Arbeit. In ihrer Arbeit haben sie eine Proctoring-Lösung entwickelt, die auf bestehenden Modellen basiert. Da das Projekt im Jahr 2016 veröffentlicht wurde, bietet es uns Einblicke in die verwendeten Methoden und ermöglicht es uns, die technischen Unterschiede zu betrachten.

Softwarevergleich In der Arbeit wurden verschiedene Softwares hinsichtlich der angebotenen Funktionen verglichen, anstatt die Gesamtfunktionalität der Softwares zu betrachten. Am Ende der Tabelle wird eine Spalte "Proposed System"vorgestellt, in der Ihre erstellte Software im Hinblick auf die Funktionalität mit den anderen verglichen wurde. Es ist zu beachten, dass alle Firmen neben dem technischen Proctoring auch ein physisches Proctoring anbieten. Das bedeutet, dass ein Beauftragter während der Prüfung die Möglichkeit hat, manuelles Proctoring durchzuführen, um eine zusätzliche Unterstützung zur technischen Möglichkeit darzustellen. Das Proposed System bietet kein physisches Monitoring an. Es ist zu vermuten, dass der Grund dafür sein könnte, dass es sich nicht um ein kommerzielles Produkt handelt und man sich auf die technischen Funktionalitäten fokussieren wollte. Ein weiterer interessanter Aspekt ist, dass das Proposed System alle anderen Funktionalitäten anbietet und diese abdecken kann, was bei den kommerziellen Produkten nicht der Fall ist. Eine Erklärung hierfür könnte sein, dass diese Funktionalitäten möglicherweise nicht zuverlässig genug funktionieren und man sich deshalb gegen eine solche Technologie entschieden hat, und stattdessen das Erkennen von Betrugsfällen durch physisches Proctoring lösen möchte.

ONLINE PROCTORING TOOLS FEATURE COMPARISON				
Features	Kryterion	Software Secure	ProctorU	Proposed System
Physical Proctor for monitoring	YES	YES	YES	NO
Voice recognition	NO	NO	NO	YES
Usage of webcam	YES	YES	YES	YES
Continuous Internet	YES	NO	YES	YES
Automatic active window capture	NO	NO	NO	YES

Abbildung 2.1: EXISTING AND PROPOSED ONLINE PROCTORING TOOL COMPARISON. Quelle: [PNSB16]

Lösungsansatz Der Lösungsansatz für die erstellte Software bestand darin, sich auf die Eingaben über Video, Mikrofon und Bildschirm zu konzentrieren. Dabei sollten alle Daten, die aus diesen drei Datenquellen stammen, verarbeitet werden, um zu überprüfen, ob es sich um verdächtige Aktivitäten handelt.

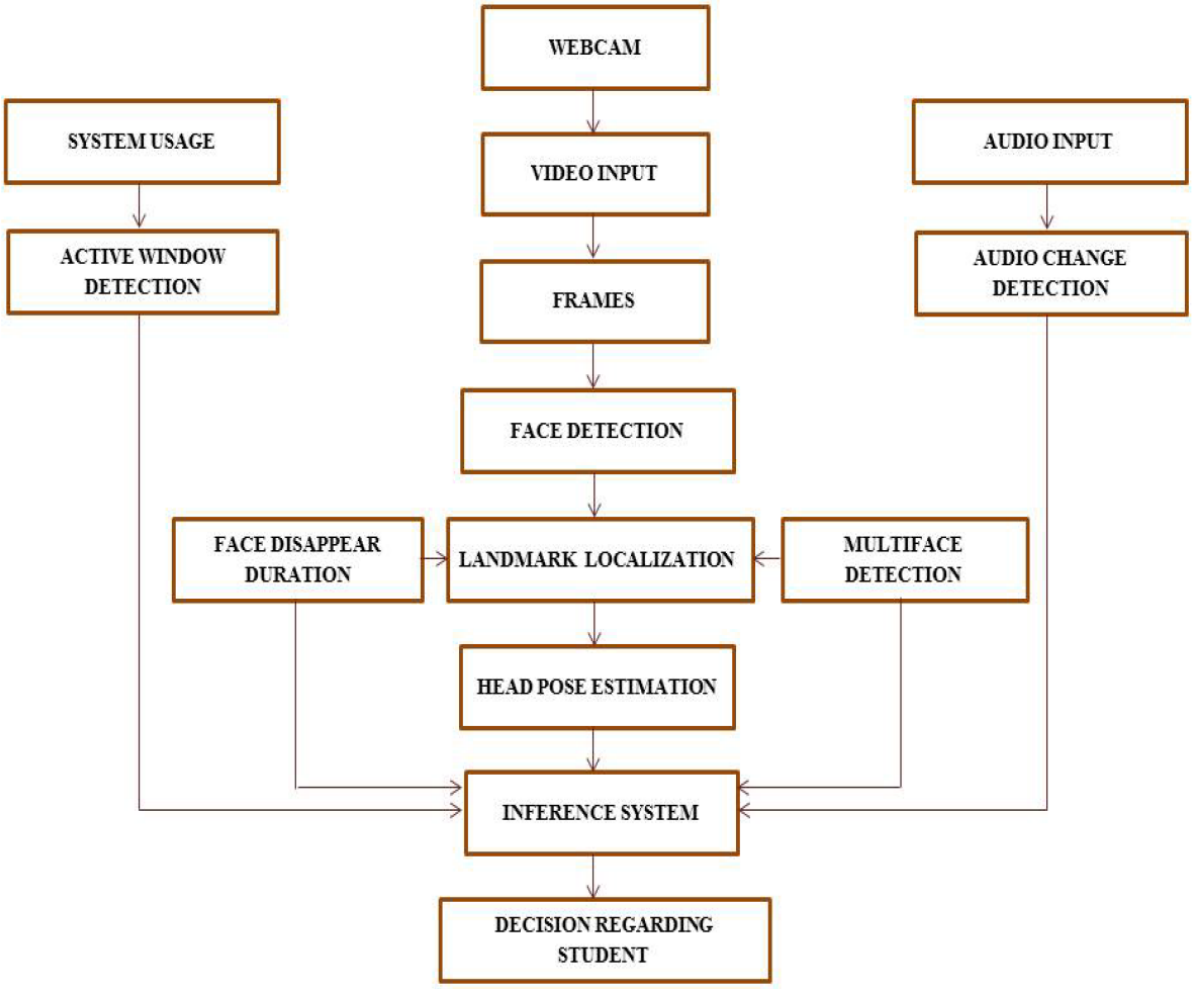


Abbildung 2.2: Architecture Design of the Proposed Multi-Modal Online Proctoring System. Quelle: [PNSB16]

Systemcheck Beim Systemcheck geht es darum, die Daten auszuwerten, die vom System erzeugt werden. In diesem Zusammenhang werden Log-Einträge erstellt, wenn der Benutzer oder die Benutzerin ein neues Programm öffnet oder ein externes System anschliesst. Es wurde im Bericht jedoch nicht näher dokumentiert, wie dies umgesetzt wurde.

Videoverarbeitung Bei der Videoverarbeitung geht es darum, Betrugsfälle mittels Videodaten zu erkennen. Im Projekt wurde dazu eine normale Webcam verwendet. Das Video wird in einzelne Frames zerlegt und diese einzeln verarbeitet. Das bedeutet, dass jedes einzelne Bild im Video analysiert wird. Diese Methode wurde verwendet, um Gesichter zu erkennen. Dabei wird jedes Bild auf Gesichter untersucht, und es werden Koordinatenpunkte für die Umrisse des Kopfes als Ausgabe generiert. Dadurch kann nicht nur die Gesichtserkennung durchgeführt werden, sondern auch der Standort und die Neigung des Kopfes erfasst werden. Dies ist in Ihrem Projekt hilfreich, um mehr Betrugsfälle zu erkennen, da ein Betrüger möglicherweise versucht, seinen Kopf zur Seite zu neigen, um einen Spickzettel anzusehen.

Audio Verarbeitung Um eine ruhige Prüfungsatmosphäre zu gewährleisten und Gespräche während der Prüfung zu verhindern, wurde in Ihrer Software die Aktivierung eines Mikrofons implementiert. Das Mikrofon nimmt die Geräusche während der Prüfung auf und hat einen eingestellten Schwellwert für die Lautstärke. Dieser Schwellwert dient dazu, unnötige Geräusche zu filtern, da das Mikrofon auch unkontrollierte Geräusche von aussen wahrnehmen kann, die falsche Warnsignale erzeugen könnten. Wenn der Schwellwert überschritten wird, protokolliert das Programm die Zeitspanne, in der der Schwellwert überschritten wurde. Dies könnte ein Hinweis darauf sein, dass es sich um einen Betrugsversuch handelt.

Kombination der Input Daten Um eine zuverlässige Meldung bei Betrugsversuchen zu gewährleisten, wurden die Eingangsdaten miteinander verglichen. Das bedeutet, wenn beispielsweise die Mikrofonerkennung und die Bilderkennung Alarm schlagen, gibt das System einen Betrugsversuch an. Gleiches gilt in Kombination mit dem Systemcheck. Dadurch sollte die Anzahl an falsch positiven Meldungen reduziert werden. Die Systeme gaben jedoch auch Alarm, wenn ein "Sensor" einen schwerwiegenderen Betrugsversuch in einem System erkannte.

2.1.3 Literaturdiskussion

In den beiden Forschungsarbeiten wurde ein Fokus auf die drei Eingabekanäle Video, Audio und System (Browser, Bildschirmaktivität) gelegt. Es zeigt sich ein Schwerpunkt auf Algorithmen in der Videoverarbeitung, die Kopf und Gesicht analysieren. Beide Arbeiten beschreiben, wie sie einzelne Frames auswerten und diesen Algorithmus über das gesamte Video hinweg anwenden. Bei der Audioüberprüfung wird auf Mikrofongeräusche zugegriffen, wobei Hintergrundgeräusche zu falschen Auffälligkeiten führen können. Daher wäre eine separate Spracherkennung sinnvoller. Zudem ist eine eigene Spracherkennung im Videomaterial erforderlich, da das Mikrofon einfach stummgeschaltet werden kann. Im Audiobereich gibt es deutliche Möglichkeiten zur Verbesserung, um falsche Detektionen zu minimieren. Bei der Systemüberwachung waren die verschiedenen Ansätze besonders interessant. Während Audio- und Videoverarbeitung fast identisch funktionieren, gibt es bei der Systemüberwachung unterschiedliche Herangehensweisen. Im Bericht "An automated Online Exam Proctoring System"werden Browseraktivitäten überwacht, während im Text "An Intelligent System for Online Exam Monitoring" Log-Einträge bei der Öffnung neuer Programme erstellt werden. Eine Kombination beider Methoden könnte sich lohnen, um sowohl den Browser als auch externe Programme effektiv zu überwachen.

2.2 Vergleich bestehender Proctoring Software

In diesem Abschnitt werden bestehende Proctoring-Softwarelösungen verglichen. Dabei werden die bekanntesten Hersteller mit den von uns erstellten Kategorien geprüft.

2.2.1 Auswahl der Software

Für die Tests wurden Softwarehersteller ausgewählt welche in Suchfunktionen zuerst auftauchen und eine Demoversion anbieten. Damit sollen potenzielle User vor dem Erwerb der Software einen ersten Eindruck durch unsere Testergebnisse bekommen. Bei einigen Softwareherstellern wurde bei einer nicht vorhandenen Testversion persönlich angefragt ob eine zur Verfügung gestellt werden kann.

Resultate Die Unternehmen, von denen wir die Proctoring-Software testen wollten, stellten uns keine Testversion zur Verfügung. Der übliche Vorgang war wie folgt: Zunächst suchte man eine Firma aus, die eine Proctoring-Lösung anbietet. Bei der Anforderung einer Demo oder einer Testversion mussten die Organisation, für die man arbeitet, und der genaue Verwendungszweck der Software angegeben werden. Da unser Interesse ausschliesslich dem Testing der Software galt und nicht dem Erwerb nach einer Demoversion, erhielten wir regelmässig Absagen oder wurden ignoriert. Wenn es zu weiteren Gesprächen kam, wurde als Begründung angeführt, dass man keine Probeversion herausgeben möchte aus Angst vor schlechter Performance der Software und einem potenziell negativen PR-Einfluss auf das Unternehmen. Diese Entscheidung blieb auch dann bestehen, als wir anboten, das Programm unter einem anonymen Namen in unserer Arbeit zu erwähnen. Daher war es uns nicht möglich, irgendeine Software zu testen. In den nächsten Abschnitten wird dennoch unser Vorgehen beschrieben, wie ein Test der Software ausgesehen hätte.

2.2.2 Kategorien

Um den Vergleich fair zu gestalten, haben wir eine Tabelle entwickelt, in der Kriterien aufgeführt sind, die objektiv beurteilt werden können. Diese Kriterien werden in den folgenden Abschnitten beschrieben.

Stimmenerkennung

Die Umgebung einer Prüfung ist für gewöhnlich ruhig, und es darf nicht geschwatzt werden. Deshalb setzen wir eine gute Stimmerkennung voraus. Geprüft werden soll das Ganze, indem wir während des Tests Schwatzen und externe Stimmen durch Videos oder andere Personen im Hintergrund laufen lassen. Die Bewertung funktioniert folgendermassen: Zwei Minuszeichen, falls keine Stimme erkannt werden kann. Ein Minuszeichen, wenn das Mikrofon getestet wird, es aber einfach umgangen werden kann (bsp. durch selbstständiges Muten). Ein Pluszeichen, wenn mindestens 50 Prozent der Fälle erkannt werden. Zwei Pluszeichen bei mehr als 80 Prozent und Drei Pluszeichen bei einer Erkennung von 100 Prozent der Betrugsfälle.

Gesten und Personenerkennung

Um Offline-Betrugsfälle zu erkennen, ist eine Gesten- und Personenerkennung mittels Facecam notwendig. Die zu prüfenden Softwares sollen erkennen, falls jemand zu lange weg vom Bildschirm schaut. Dazu gehört das Wegschauen auf die Seite, nach oben und nach unten. Damit sollen Offline-Betrugsmöglichkeiten wie die Verwendung von zusätzlichen elektronischen Geräten oder unerlaubten Notizen erkannt werden. Ebenfalls soll die Software erkennen, wenn sich eine zweite Person vor der Kamera befindet. Hier gilt die folgende Bewertung: Drei Minuszeichen bei keiner Facecam-Verwendung. Zwei Minuszeichen bei einfachem Umgehen der Kameraerkennung

(bspw. mittels Bild vor der Kamera). Ein Minuszeichen bei Nichterkennung von zusätzlichen Personen. Ein 'O' wenn mehr als 0 und weniger als 50 Prozent der Fälle erkannt werden. Ein Pluszeichen, wenn mindestens 50 Prozent der Fälle erkannt werden. Zwei Pluszeichen bei mehr als 80 Prozent. Drei Pluszeichen bei einer Erkennung von 100 Prozent der Betrugsfälle.

Bildschirmcheck

Online-Betrugsfälle werden mittels einem Bildschirmcheck überprüft. Die Software muss erkennen, wenn externe Programme geöffnet werden oder Webseiten als unerlaubte Hilfe benutzt werden. Das Kriterium hier funktioniert folgendermassen: Drei Minuszeichen, falls keine automatische Bildschirmüberprüfung stattfindet. Zwei Minuszeichen bei Nichterkennung von extern benutzter Software. Ein Minuszeichen bei einfacher Umgehung des Bildschirmchecks (bspw. Benutzen eines zweiten Bildschirms). 'O' wenn mehr als 0 und weniger als 50 Prozent der Fälle abgedeckt werden. Ein Pluszeichen, wenn mehr als 50 Prozent der Fälle abgedeckt werden können. Zwei Pluszeichen, wenn mehr als 80 Prozent der Fälle abgedeckt werden können. Drei Pluszeichen, wenn 100 Prozent der Fälle abgedeckt werden. Es wurde sich dafür entschieden, noch einen Extra-Punkt mittels einem Sternzeichen zu geben, falls es mit der Software möglich ist, erlaubte externe Ressourcen zu benutzen. Ein Beispiel hierfür wäre, wenn man eine Programmierprüfung schreibt und dem Schüler oder der Schülerin die Möglichkeit gibt, eine IDE zu benutzen.

Usability

Eine Proctoring-Software muss selbsterklärend und einfach zu bedienen sein. Da der Schüler oder die Schülerin sich auf die Prüfung konzentrieren will, soll möglichst wenig Zeit verwendet werden, um die Software zu starten. Um hier eine objektive Bewertung abgeben zu können, soll die Zeit zwischen dem Start der Software und dem Beginn der Prüfung gemessen werden. Als Bewertende für diesen Test gelten die Autoren dieser Arbeit und willkürlich ausgewählte Studenten. Dabei gibt es folgende Bewertung: Ein Minuszeichen, wenn das Starten der Software länger als 5 Minuten dauert. Ein Extra Minuszeichen, wenn die Testnutzer im Durchschnitt das Test-UI schlecht finden. Ein Pluszeichen, wenn das Starten der Software 5 Minuten oder kürzer dauert. Ein Extra Pluszeichen, wenn das User Interface im Durchschnitt positiv bewertet wird. Ein "O" Zeichen, wenn sich die Plus- und Minuszeichen aufheben.

Kapitel 3: Architektur

3.1 Funktion der Software

Unsere Software besteht aus drei Hauptkomponenten. Die erste Komponente ist der 'Client', der speziell für Studierende konzipiert ist. Der Hauptzweck des Clients besteht darin, Aufnahmen zu erstellen und diese zur Analyse an den Server zu senden. Die zweite wichtige Komponente ist der 'Analyse Server'. Dieser teilt sich in zwei Unterbereiche: die Analyse-Software und das Backend. Diese Aufteilung wurde vorgenommen, da beide Bereiche unabhängig voneinander operieren können. In den folgenden Unterabschnitten wird über die einzelnen Komponenten genauer gesprochen.

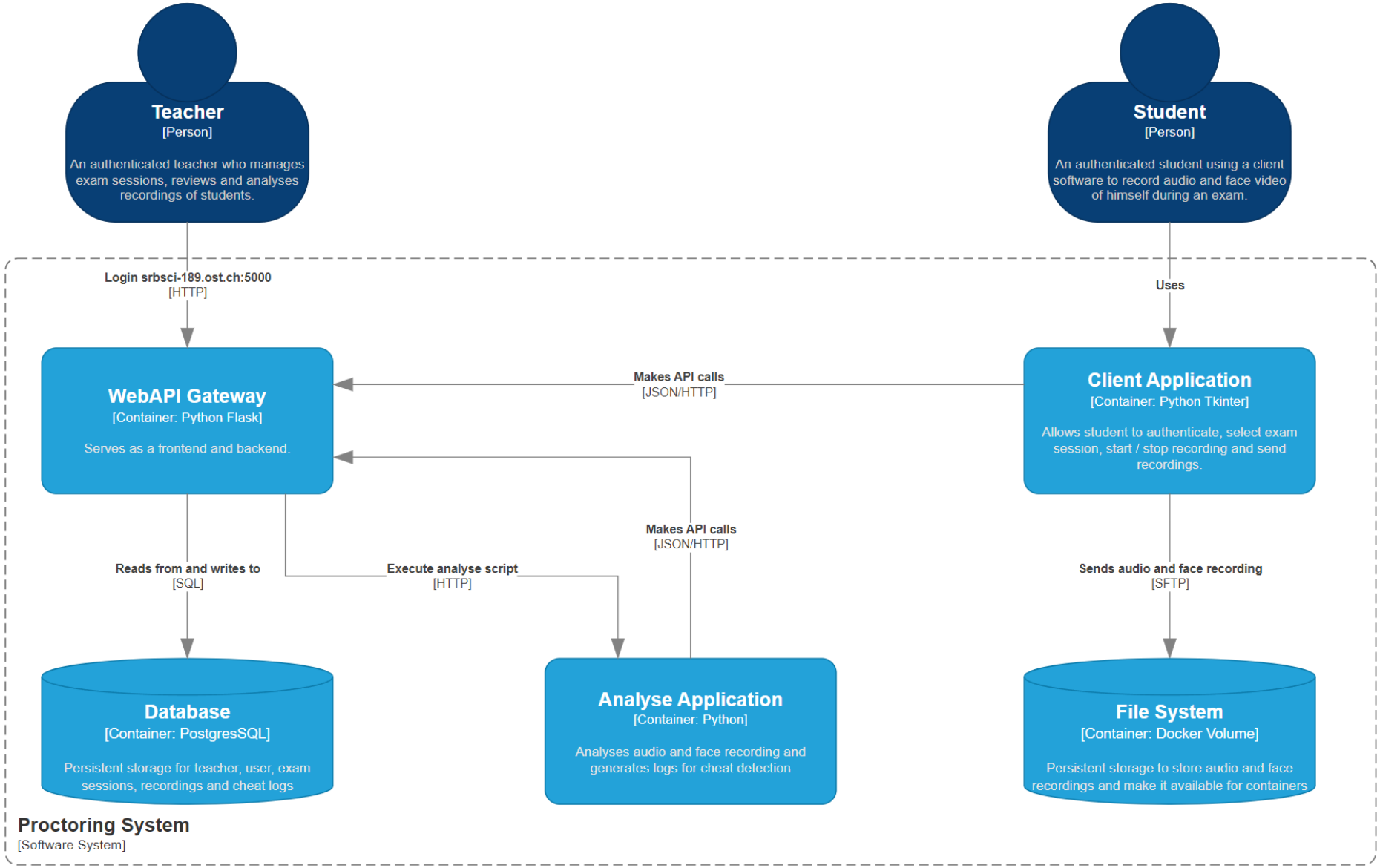


Abbildung 3.1: C4 Diagramm Layer 2 - Software System

3.1.1 Client

Der Client unserer Software ist speziell für Studierende entwickelt worden. Vor Beginn der Prüfung muss der Studierende den Client starten. Dies geschieht durch die Eingabe der Login-Daten auf dem Startbildschirm. Anschliessend wählt der Studierende aus einem Dropdown-Menü die entsprechende Prüfung und aus einem weiteren Menü das zu verwendende Mikrofon aus. Nachdem diese Einstellungen vorgenommen wurden, aktiviert der 'Start Recording'-Button die Aufnahmefunktion. Ab diesem Moment werden sowohl die Webcam als auch das Mikrofon aufgezeichnet. Für die Webcam-Aufnahme wird automatisch die primäre Kamera des Geräts verwendet, was bei vorhandener Webcam unproblematisch ist. Bei der Audioaufnahme kommt das zuvor ausgewählte Mikrofon zum Einsatz. Nach Abschluss der Prüfung kann der Studierende die Aufnahme durch Klicken auf 'Stop Recording' beenden. Dadurch werden zwei Dateien erstellt: eine MP4-Datei mit dem Video der Webcam und eine WAV-Datei mit der Audioaufnahme vom Mikrofon. Zum Schluss kann der Studierende durch Klicken auf 'Send Video to Server' die Aufnahmen direkt an den Analyse-Server übermitteln.

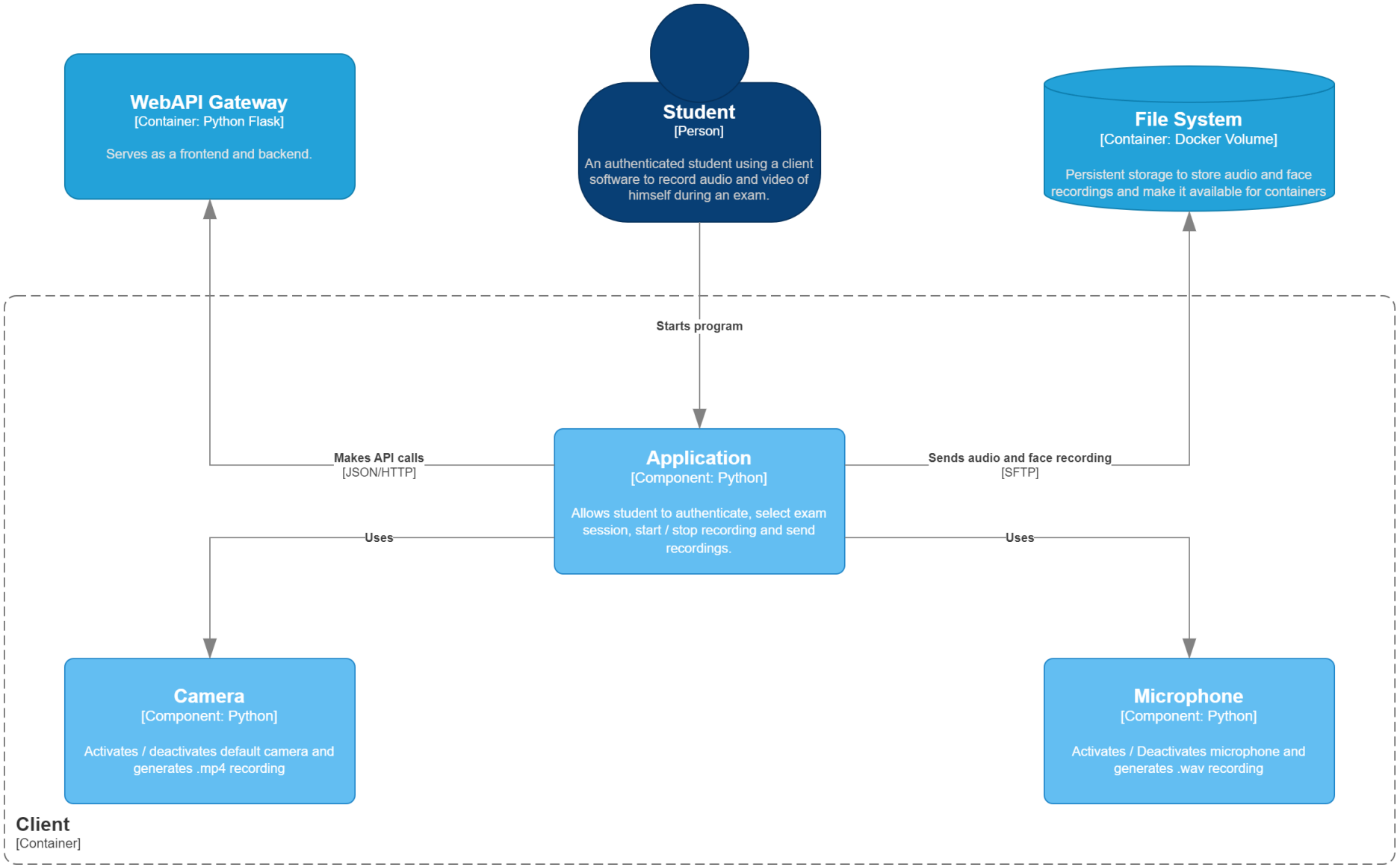


Abbildung 3.2: C4 Diagramm Layer 3 - Client

3.1.2 Analyse Server

Die Analysefunktion unserer Software ist in mehrere Abschnitte gegliedert. Zuerst werden die einzelnen Sensoren beschrieben, die für die Datenerfassung verwendet werden. Jeder Sensor wird dabei individuell betrachtet, um seine spezifischen Funktionen und die Art der erfassten Daten zu verstehen.

Anschliessend erfolgt die Zusammenführung dieser Daten zu einem kohärenten Datensatz. Diese Integration ist entscheidend, um einen umfassenden Überblick über die gesammelten Informationen zu erhalten.

Abschliessend wird die Aufbereitung der Daten für das Backend erläutert. Hierbei wird beschrieben, wie die verarbeiteten Daten für die weitere Analyse durch das Backend-System übermittelt werden.

Facedetection

Im Facedetection Sensor geht es darum aus einzelnen Frames die Anzahl Gesichter zu erkennen. Dazu wurde die Library von OpenCV verwendet. Sie haben eine Gesichtserkennung welche auf den Haarcascade Algorithmus basiert.

Für die Gesichtserkennung müssen dem Facedetection-Modell spezifische Parameter übergeben werden. Diese Parameter sind entscheidend, um eine präzise Erkennung von Gesichtern zu gewährleisten, während gleichzeitig eine effiziente Verarbeitungszeit pro Frame sichergestellt wird.

Tabelle 3.1: Parameter für Gesichtserkennung

Parameter	Beschreibung
ScaleFactor	Dieser Parameter kontrolliert die Skalierung des Bildes. Ein typischer Wert wie 1.15 bedeutet, dass das Bild in jeder Iteration um 15 Prozent verkleinert wird, was es ermöglicht, Gesichter unterschiedlicher Grösse zu erkennen.
MinNeighbors	Durch die Festlegung dieser Anzahl wird bestimmt, wie viele Nachbarbereiche ebenfalls als Gesichter erkannt werden müssen, um ein Gebiet als Gesicht zu klassifizieren. Ein höherer Wert reduziert Falschmeldungen. In Unserem Projekt wurde sich aufgrund verschiedener Versuche für den Wert 4 entschieden.
MinSize	Dies definiert die minimale Grösse der zu erkennenden Gesichter im Bild. Kleinere Gesichter als der festgelegte Schwellenwert werden ignoriert, was die Effizienz steigert. Im Projekt wurde der Wert 30x30 Pixel festgelegt.

Die Parameterwahl für die Gesichtserkennung wurde sorgfältig getroffen, um Genauigkeit und Leistung auszugleichen. Eine überhöhte Genauigkeit kann die Verarbeitungszeit pro Frame verlängern, während zu schnelle Erkennung die Genauigkeit beeinträchtigen kann. Die endgültigen Werte wurden nach verschiedenen Tests während der Entwicklungsphase festgelegt, um optimale Ergebnisse zu erzielen.

Kopfhaltung Überwachung

Die Erkennung der Kopfneigung ist ein wesentlicher Bestandteil unserer Software, um sicherzustellen, dass Studierende während der Prüfung nicht abgelenkt werden. Für diese Funktion wurde die Mediapipe-Bibliothek eingesetzt. Diese Bibliothek liefert präzise Gesichtspunkte, von denen jedoch für unsere Zwecke nur drei benötigt werden: die Punkte an beiden Augen und der Punkt auf der Nasenspitze. Basierend auf diesen drei Punkten berechnen wir die Neigung des Kopfes, die als Referenz für die folgenden 30 Frames dient.

Während der nächsten 30 Frames wird kontinuierlich überprüft, ob sich die Neigung des Kopfes um mehr als 10 Prozent im Vergleich zum Referenzwert ändert. Eine solche Änderung wird als auffällige Kopfbewegung interpretiert. Diese Methode ermöglicht es uns, effektiv zu erkennen, wenn ein Studierender seinen Blick von der Prüfung abwendet. Somit trägt die Methode zur Aufrechterhaltung der Integrität des Prüfungsprozesses bei.

```
if self.reference_angle is not None:
    angle_difference = abs(angle - self.reference_angle)

    if angle_difference > 10:
        detection.append(True)
    else:
        detection.append(False)
```

Abbildung 3.3: Berechnung der Veränderung zum Referenzwert

Munderkennung

Um eine zuverlässige Munderkennung zu gewährleisten, haben wir in unserer Software zwei spezielle Sensoren integriert. Der erste Sensor ist darauf ausgerichtet, zu erkennen, ob der Mund sichtbar ist. Der zweite Sensor überprüft, ob gesprochen wird. Diese zweifache Sensortechnik zielt darauf ab, Gespräche zu detektieren, selbst wenn das Mikrofon ausfällt oder stummgeschaltet ist.

Die Implementierung der Munderkennung stellte eine Herausforderung dar, da kein spezialisiertes Modell für diese Aufgabe gefunden werden konnte. Daher entschieden wir uns für die Nutzung des Mediapipe Landmarkmodells. Obwohl dieses Modell keine direkte Munderkennung bietet, ermöglicht es die Identifikation einzelner Mundpunkte.

Ursprünglich wurde versucht, einen Algorithmus zu entwickeln, der aktiv wird, wenn die Mundpunkte nicht erkannt werden. Dies erwies sich jedoch als unpraktikabel, da die Gesichtspunkte des Mundes oft basierend auf den Positionen anderer Gesichtsmerkmale geschätzt werden. Das bedeutet, dass selbst wenn der Mund nicht sichtbar ist, dennoch Punkte an den mutmasslichen Mundpositionen erkannt werden, sofern andere Teile des Gesichts erkennbar sind. Aufgrund dieser Einschränkung war es notwendig, einen alternativen Ansatz zu verfolgen, um eine effektive Munderkennung zu realisieren.

Angesichts des hohen Ressourcen- und Zeitaufwands, der mit der Entwicklung eines neuen Modells für die Munderkennung verbunden wäre, entschieden wir uns für einen praktikableren Ansatz. Wir nutzten die Farbwerte der Landmarkpunkte des Mundes, die vom Mediapipe-Modell bereitgestellt werden. Konkret wurden die Farbwerte aller Mundpunkte zusammengezählt, um daraus einen Durchschnittswert zu berechnen. Dieser Durchschnittswert diente dann als Refe-

renz für die folgenden 30 Frames. Während dieser 30 Frames wurde kontinuierlich überprüft, ob der aktuelle Farbwert des Mundes signifikant vom Referenzwert abweicht. Eine solche Abweichung würde darauf hindeuten, dass der Mund verdeckt ist. In diesem Fall wird der spezifische Sensor aktiviert, der anzeigt, dass der Mund nicht sichtbar ist. Dieser Ansatz ermöglichte es uns, eine effiziente und zuverlässige Lösung für die Herausforderung der Munderkennung zu implementieren, ohne die Notwendigkeit, ein komplett neues Modell zu entwickeln.

Der zweite Sensor in unserem System wurde entwickelt, um festzustellen, ob gesprochen wird. Hierfür griffen wir erneut auf das Landmarkmodell von Mediapipe zurück. Unser Ziel war es, Veränderungen im Bereich des Mundes zu detektieren, speziell das Öffnen des Mundes während des Sprechens. Dafür wählten wir zwei spezifische Landmarkpunkte aus: den Mittelpunkt der Oberlippe und den Mittelpunkt der Unterlippe. Die Distanz zwischen diesen beiden Punkten wurde gemessen und als Basiswert für die folgenden 30 Frames festgelegt. In Übereinstimmung mit dem Prinzip des ersten Sensors wurde während dieser 30 Frames überwacht, ob sich die Distanz zwischen den beiden Lippenpunkten signifikant vom Referenzwert unterscheidet. Eine solche signifikante Abweichung deutet darauf hin, dass gesprochen wird. In diesem Fall wird der Sensor aktiviert, was darauf hinweist, dass eine Sprechaktivität stattfindet. Dieses Verfahren ermöglicht es uns, effektiv zu überwachen, ob der Studierende spricht, selbst wenn das Mikrofon nicht verfügbar oder stummgeschaltet ist.

```
if last_face_landmarks:
    if mouth_covered:
        self.detection.append(True)
    else:
        mouth_landmarks = [(int(last_face_landmarks.landmark[i].x * frame.shape[1]),
                            int(last_face_landmarks.landmark[i].y * frame.shape[0]))
                           for i in [13, 14]]
        current_distance = self.calculate_distance(mouth_landmarks[0], mouth_landmarks[1])

        if self.last_distance is not None and abs(current_distance - self.last_distance) > self.distance_threshold:
            self.detection.append(True)
        else:
            self.detection.append(False)

        self.last_distance = current_distance
```

Abbildung 3.4: Berechnung der Distanz zwischen Ober- und Unterlippe

Iris Erkennung

Die Iriserkennungsfunktion in unserer Software nutzt das Mediapipe Landmark-Modell, um zu kontrollieren, dass Studierende während der Prüfung nicht auf andere Bildschirme blicken. Dabei konzentrieren wir uns auf die Identifikation der Iris sowie der äusseren Augenwinkel beider Augen.

Wir berechnen den Abstand zwischen der Iris und dem jeweiligen äusseren Augenwinkel. Diese Distanzmessung ist ein wichtiger Indikator dafür, wohin die Studierenden schauen. Ein auffällig kleiner oder grosser Abstand kann darauf hinweisen, dass der Blick nicht auf den Prüfungsbildschirm gerichtet ist.

```python
def iris_position(self, iris_center, right_point, left_point):
    center_to_right_dist = self.euclidean_distance(iris_center, right_point)
    total_distance = self.euclidean_distance(right_point, left_point)
    ratio = center_to_right_dist / total_distance
    iris_position = ""

    if ratio < 2.6 or ratio > 3.05:
        self.detection.append(True)
    else:
        self.detection.append(False)

    return iris_position, ratio
```

Abbildung 3.5: Berechnung der Iris Ratio

Erkennt unser System eine solche abweichende Blickrichtung, wird die Iriserkennung aktiv und signalisiert einen potenziellen Fehler. Dieser Mechanismus ist entscheidend, um die Aufmerksamkeit der Studierenden auf die Prüfung zu lenken und sicherzustellen, dass sie nicht durch externe Bildschirme oder Ablenkungen beeinträchtigt werden.

Spracherkennung

Unser System nutzt die Google Speech Recognition Library, um gesprochene Worte in Audioaufnahmen zu erkennen. Wir haben das Spracherkennungsmodell dieser Bibliothek so eingestellt, dass es Deutsch (Schweiz) versteht. Wir stiessen jedoch auf das Problem, dass Schweizerdeutsch vom Modell nicht immer richtig erkannt wird. Für unsere Zwecke ist dies jedoch kein grosses Problem. Unser Hauptziel ist es, festzustellen, ob in der Aufnahme überhaupt gesprochen wird. Daher ist die Fähigkeit des Systems gesprochene Sprache zu erkennen, ausreichend.

```python
text = self.recognizer.recognize_google(audio, language='de-CH')
if text:
    timestamps.append((segment_start / frame_rate, text))
```

Abbildung 3.6: Aufruf der Google Speech Recognition API

Zusammenführung einzelner Sensoren

Zur Datenaufbereitung für das Backend und zur Durchführung genauer Zeitstempel wurde eine Hauptklasse implementiert. Diese Klasse durchläuft alle Videos und teilt sie in 10-Sekunden-Fragmente auf. Jedes Fragment wird dann von jedem Sensor analysiert, wobei der Sensor innerhalb dieser 10 Sekunden alle Auffälligkeiten protokolliert. Nach Abschluss der Videoanalyse wird der Prozentsatz der Auffälligkeiten berechnet. Dies ist wichtig, um die Anzahl der Fehldetektionen zu verringern. Insbesondere bei Auffälligkeiten, die nur in einem Frame auftreten, besteht eine hohe Wahrscheinlichkeit, dass es sich um einen sogenannten „False Positive“ handelt. Nach der Auswertung aller Daten wird ein JSON-Segment erstellt, das angibt, welche Sensoren in den 10 Sekunden Auffälligkeiten erkannt haben und welche nicht. Nach der Bearbeitung aller Fragmente ergibt sich eine JSON-Datei mit Metadaten und den Timestamps der erkannten Auffälligkeiten.

Ähnlich verhält es sich mit der Audiodatei. Diese wird ebenfalls in Zehn-Sekunden-Segmente unterteilt, die jeweils einzeln analysiert und dann zusammengefasst werden. Der wesentliche Unterschied besteht jedoch darin, dass sie nicht durch mehrere Sensoren, sondern ausschliesslich durch die Spracherkennung verarbeitet wird. Nach der Auswertung der Audiodatei wird eine separate JSON-Datei erstellt, in der die Zeitstempel zusammen mit den erkannten Wörtern aufgeführt sind. Die beiden JSON Dateien werden Anschliessend an Backend übergeben.

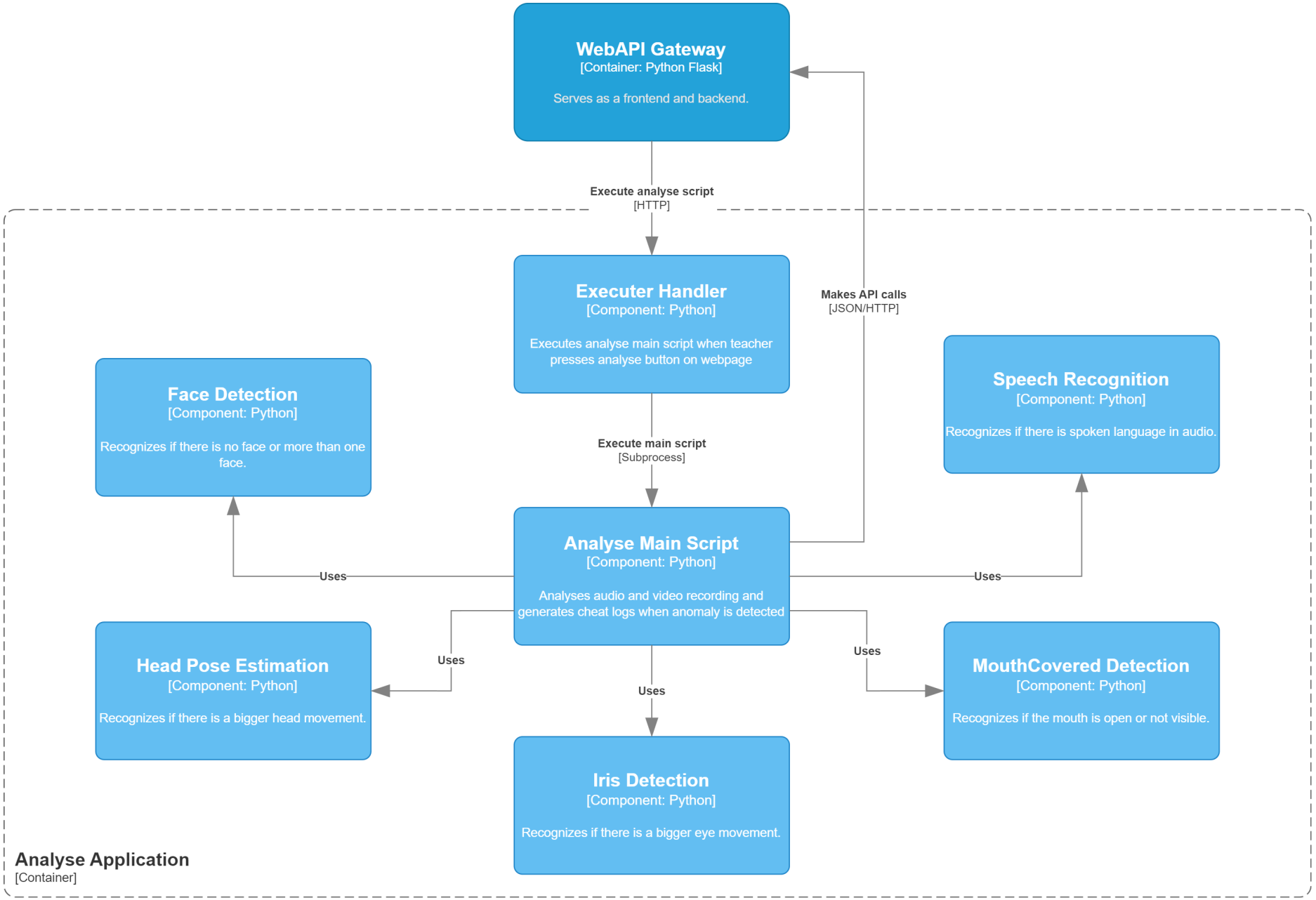


Abbildung 3.7: C4 Diagramm Layer 3 - Analyse Server

3.1.3 Frontend

Für die Darstellung und Auswertung von Daten wird ein Dashboard entwickelt, das ausschliesslich für den Zugriff durch Dozenten bestimmt ist. Im Frontend kommt das Flask-Framework zum Einsatz, um die HTML-Seiten zu rendern und die Benutzeroberfläche zu gestalten. Darüber hinaus spielt das Frontend eine wesentliche Rolle bei der Organisation und Überwachung von Prüfungssessions. Im Folgenden wird erläutert, wie das Frontend von Beginn der Prüfung bis hin zur Auswertung der Daten eingesetzt wird.

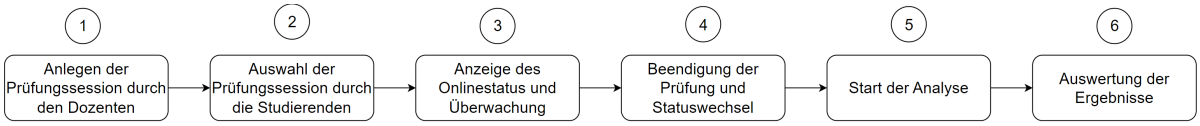


Abbildung 3.8: Flussdiagramm - Ablauf einer Prüfungssession im Frontend

Tabelle 3.2: Ablauf einer Prüfungssession im Frontend

Nummerierung	**Beschreibung**
1.	Der Dozent legt über das Frontend eine Prüfungssession an und bestimmt spezifische Informationen wie z.B. Datum und Uhrzeit. Dieser Schritt dient der organisatorischen Zuordnung und Vorbereitung der Prüfung für die Studierenden. Dabei geht es nicht um die Erstellung von Prüfungsfragen, sondern um die organisatorische Planung der Prüfungssession.
2.	Nach der Vorbereitung wählen die Studierenden im Client die festgelegte Prüfungssession aus und beginnen mit der Aufnahme der Prüfung.
3.	Das Frontend zeigt den Onlinestatus der Studierenden als „aktiv“ an, sobald sie die Prüfung starten. Dies ermöglicht dem Dozenten eine Echtzeitüberwachung der Prüfungsdurchführung.
4.	Nach Beendigung der Prüfung und dem Hochladen der Aufnahmen auf dem Server wechselt der Status der Studierenden zu „Offline“. Dies signalisiert, dass alle Prüfungsteilnehmer die Prüfung abgeschlossen haben.
5.	Der Dozent startet die Analyse auf der „Exam Session Page“, nachdem alle Studierenden offline sind.
6.	Nach Abschluss der Analyse sind detaillierte Auswertungen der Aufnahmen für jede individuelle Prüfungssession verfügbar, einschliesslich der Gesamtergebnisse und spezifischer Vorfälle oder Besonderheiten einzelner Studierenden.

Wireframes

Um eine klare Richtlinie für das Design und die Struktur des Dashboards zu schaffen, werden Wireframes entwickelt. Die Wireframes dienen dazu, dass das Team nicht unstrukturiert mit der Umsetzung beginnt, sondern von Anfang an einen klaren Überblick über die Layout- und Funktionsaspekte des Dashboards hat. Diese sollten als Inspirationsquelle betrachtet werden, die als Leitfaden für das Design dient und auch Raum für kreative Anpassungen und Verbesserungen lässt. Die Wireframes werden im Anhang hinzugefügt.

Aufbau Dashboard

Das Userflow Diagramm bietet einen gesamten Überblick darüber, wie auf die einzelnen Seiten zugegriffen wird. Es visualisiert den Pfad, den ein Dozent durch die verschiedenen Abschnitte der Webseite nimmt, inklusive der Entscheidungspunkte und Navigationsmöglichkeiten. Die genauen Funktionen oder Aufgaben jeder Seite, sowie das Aussehen der Webseite werden in den folgenden Abschnitten beschrieben.

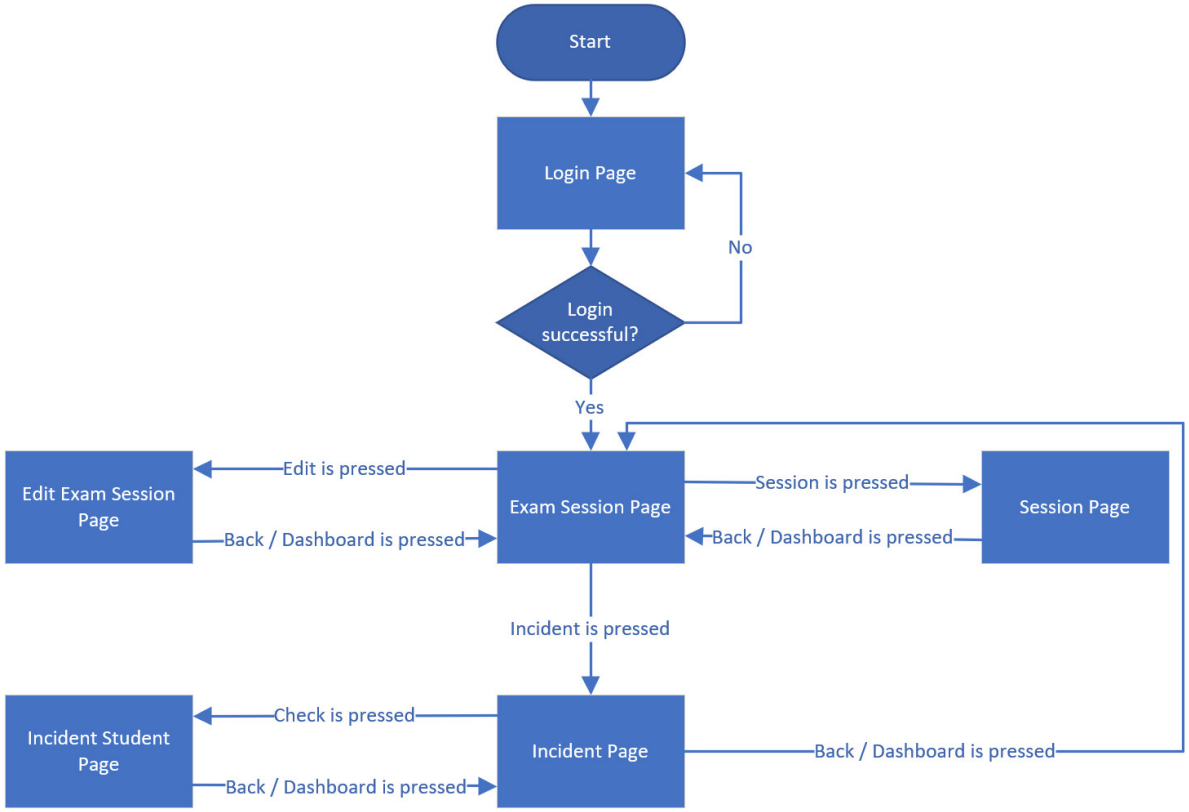


Abbildung 3.9: Website User Flow Diagramm

Dashboard - Exam Session Page

Nach der Login Page sieht der Dozent alle seine Prüfungsessionen. Hier hat er die Möglichkeit seine Prüfungsession zu verwalten. Sobald eine Prüfungsession vorbei ist, kann oben rechts den Button "Start Analyzing Recordings" gedrückt werden, damit der Analyse Server die Recordings von den einzelnen Studenten analysieren und auswerten kann.

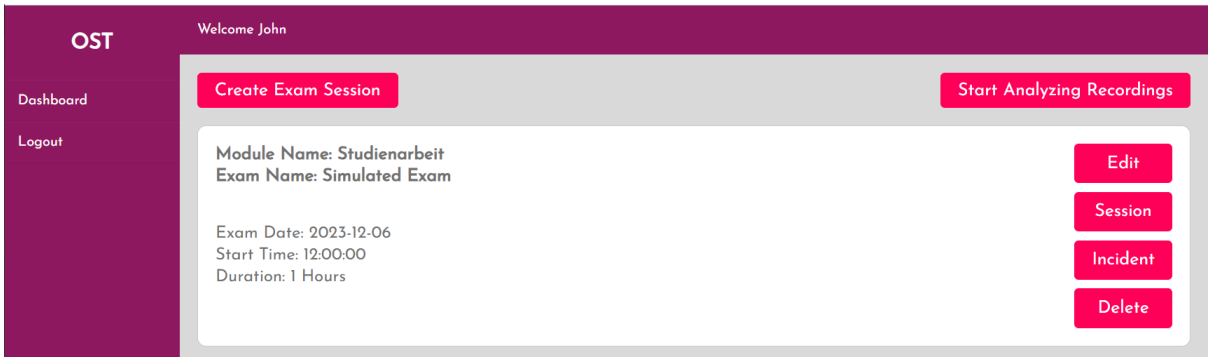


Abbildung 3.10: Dashboard - Exam Session Page

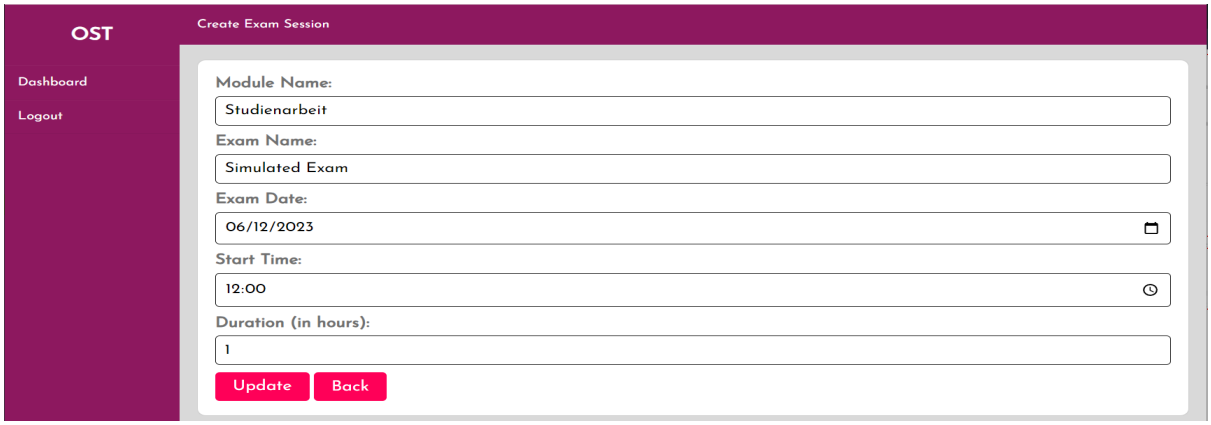


Abbildung 3.11: Dashboard - Verwaltung Exam Session Page

Dashboard - Session Page

Wenn der Button "Session" in einer Prüfungssitzung gedrückt wird, wird der Aktivitätsstatus der Studierenden angezeigt, um anzuzeigen, ob sie die Prüfung noch ablegen oder bereits abgeschlossen haben

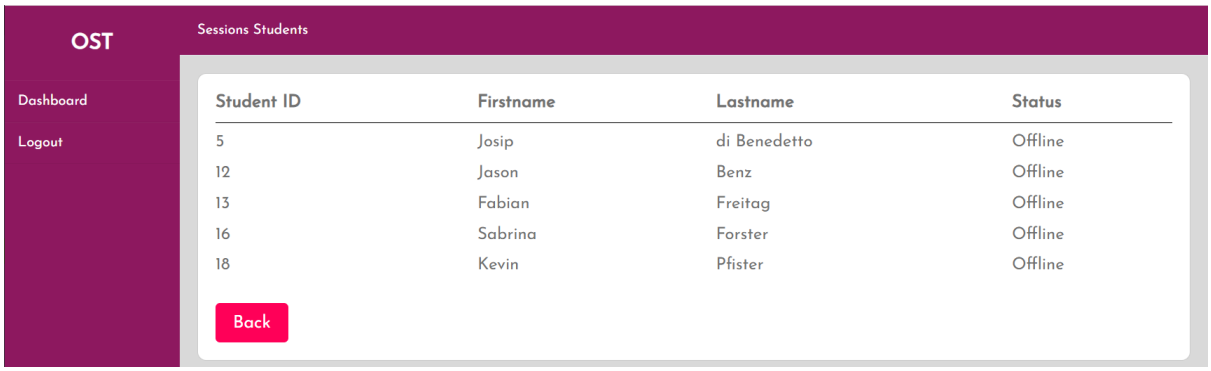


Abbildung 3.12: Dashboard - Aktivitätstatus der Studenten

Dashboard - Incident Page

Auf der Incident page wird die Anzahl aller Betrugsfsällen, sowie die Anzahl Betrugsfälle einzelner Studenten aufgelistet. Genauere Details über die Betrugsfälle eines Studenten kann über den Link "Check" hervorgerufen werden.

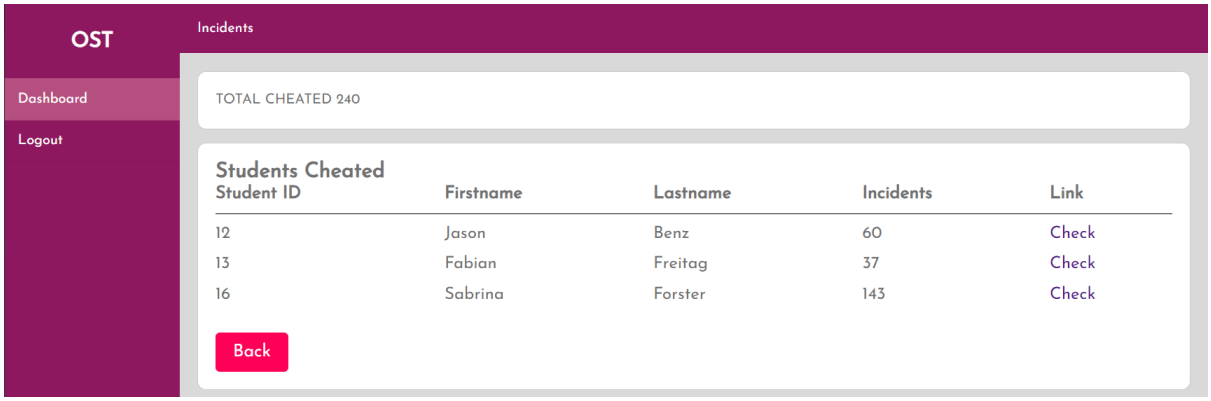


Abbildung 3.13: Dashboard - Incident Page

Dashboard - Incident Page einzelner Studenten

Hier werden die genauen Details der Betrugsfälle von Studierenden aufgezeigt, die während der Prüfung aufgetreten sind. Zuoberst wird die Audio- und Videoaufnahme gezeigt, die während der Prüfung aufgezeichnet wurde. Anschliessend werden zwei Tabellen dargestellt, die die Auswertung der Betrugsfälle auflisten und den Zeitpunkt ihres Auftretens in der Aufnahme angeben. Eine Tabelle für die Audioaufnahme und eine andere für die Videoaufnahme. Die Tabellen sind so aufgebaut, dass die Einträge gefiltert werden können.Zusätzlich wird beim Klicken auf den Zeitstempel automatisch zur entsprechenden Stelle in der Aufnahme vorgespult.

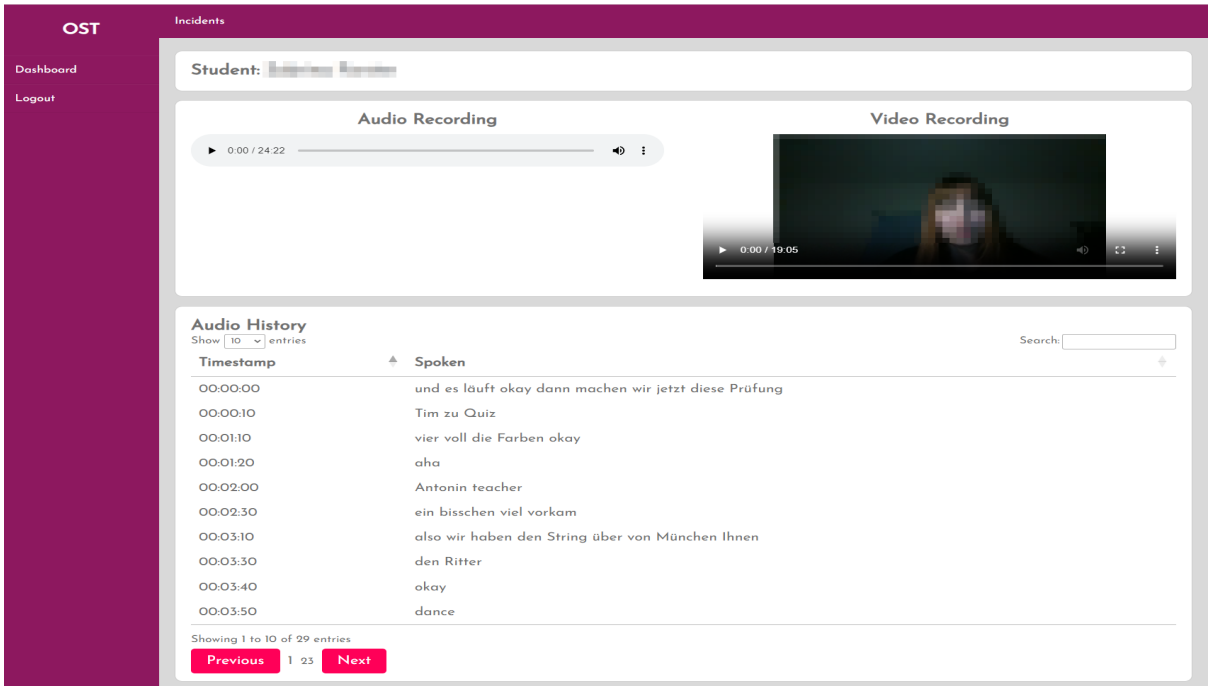


Abbildung 3.14: Dashboard - Incident Page einzelner Studenten Teil 1

Abbildung 3.15: Dashboard - Incident Page einzelner Studenten Teil 2

Verwendung von Flask Inheritance

Das Dashboard wird nicht als Single Page Application konzipiert, sondern nutzt das Flask Vererbungsprinzip für die Strukturierung. Durch Vererbung können Templates effizient wiederverwendet werden und die Gestaltung der Benuterzoberfläche optimiert werden. Die folgende Grafik veranschaulicht den Aufbau des Dashboards und zeigt, welche HTML-Seiten von welchen Vorlagen erben. Insgesamt sind 6 HTML-Seiten erstellt. Die Wichtigsten Funktionen vom Dashboard werden im nächsten Abschnitt erklärt.

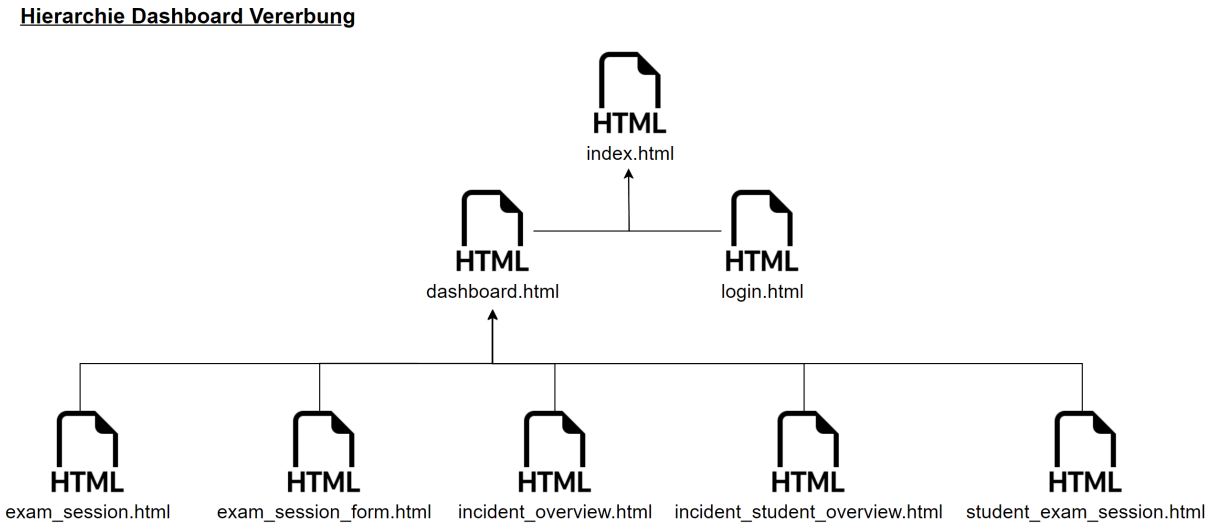


Abbildung 3.16: Hierarchy Dashboard Vererbung

3.1.4 Backend

Analog zur Verwedung von Flask im Frontend wird auch im Backend Flask eingesetzt. Das Backend der Anwendung folgt dem Model-View-Controller Konzept. Auf der nächsten Seite ist das C4 Diagramm zu sehen und zeigt wie das Backend aufgebaut ist.

Im Diagramm werden vier Controller verwendet: der Authentication Controller, der Student Exam Session Controller, der Incident Controller und der Exam Session Controller, die zur Interaktion mit dem Dashboard verwendet werden und ein API Controller, der die ganzen Requests vom Analyse Server sowie Client bearbeitet. In der folgenden Tabellen werden die Aufgaben des Controllers erklärt:

Tabelle 3.3: Beschreibung Backend Controllers

Controller	**Aufgabe**
Authentication Controller	Der Authentication Controller ist für die Authentifizierung der Dozenten zuständig. Bei einer erfolgreichen Authentifizierung wird der Dozent zum Exam Session Controller weitergeleitet.
Exam Session Controller	Dieser Controller ermöglicht es den Dozenten, die Prüfungssession zu verwalten und bietet Zugriff auf die Incident- sowie Session-Seiten.
Incident Controller	Der Incident Controller ist für die Darstellung von Betrugsfällen verantwortlich und zeigt auf, welche Studierenden betrogen haben könnten und auf welche Weise.
Student Exam Session Controller	Der Student Exam Session Controller rendert für die Prüfungssession die teilnehmenden Studierenden und zeigt den Online-/Offline-Status der Studierenden an.
API Controller	Der API Controller agiert im Hintergrund und verbindet die Analyse- sowie Client-Anwendung mit der Datenbank, um Daten zu speichern oder zu aktualisieren.

Ergänzend wurden Service-Komponenten für jede Tabelle in der Datenbank implementiert, die eine Vielzahl von CRUD-Operationen ausführen können. Diese Architektur erleichtert die Verwaltung des Codes und stellt sicher, dass das System einfach gewartet und erweitert werden kann.

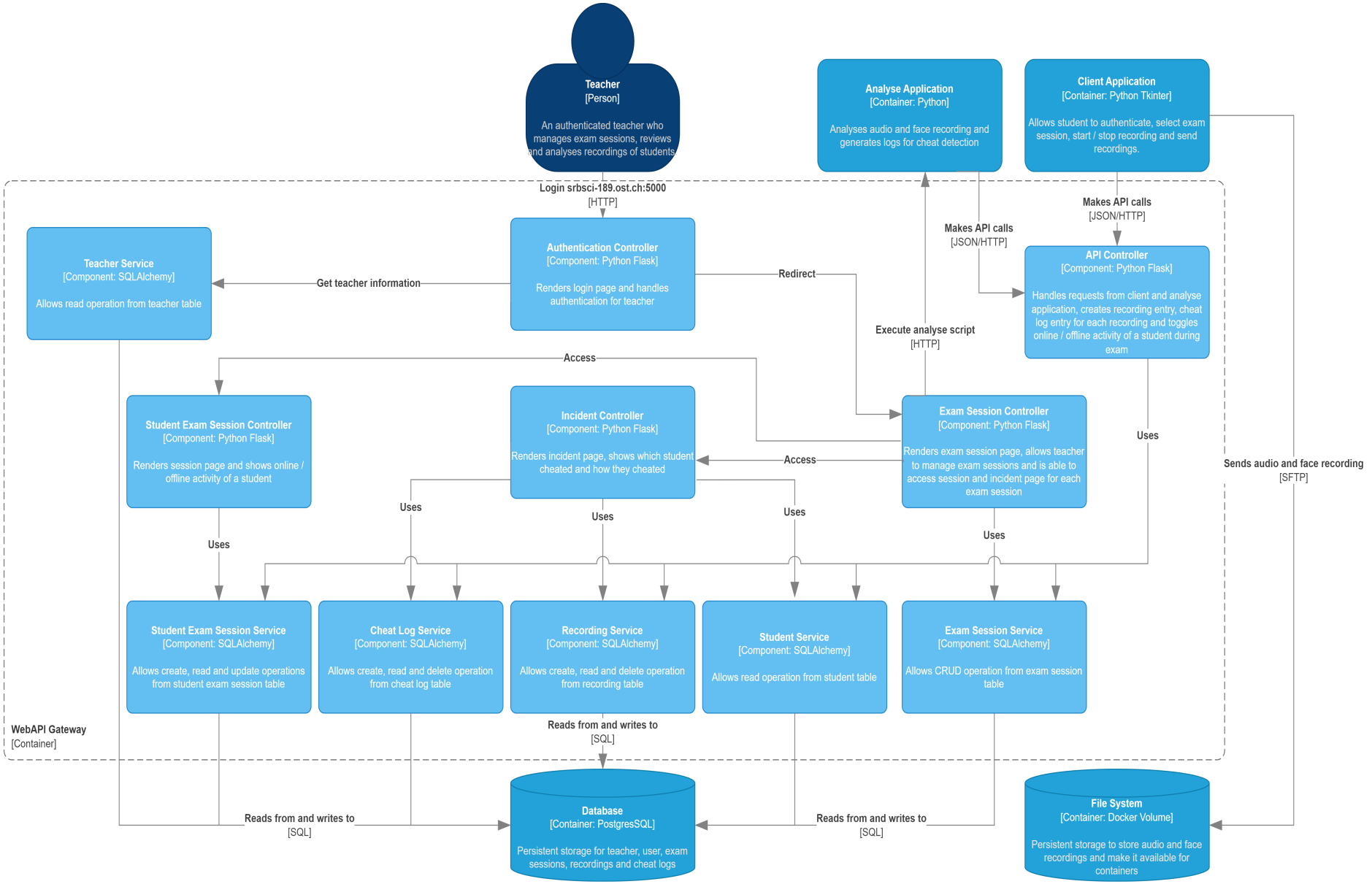


Abbildung 3.17: C4 Diagram Layer 3 - WebAPI Gateway

3.1.5 Datenbank

Als Datenbank wird eine PostgreSQL-Datenbank verwendet. Im ERD-Diagramm werden die Tabellen aufgelistet, die in der Datenbank für das Proctoring-System vorhanden sind. Das System ist entscheidend für die Speicherung und Zuordnung von Betrugsfällen zu den entsprechenden Aufnahmen und Studierenden, was notwendig ist, damit die Einträge im Dashboard angezeigt werden können. Während die Datenbank nur die Pfade der Audio- und Videoaufnahmen speichert, befinden sich die physischen Dateien auf einem separaten Dateisystem auf dem Server. Diese Dateien werden als Docker-Volume für die anderen Container zur Verfügung gestellt.

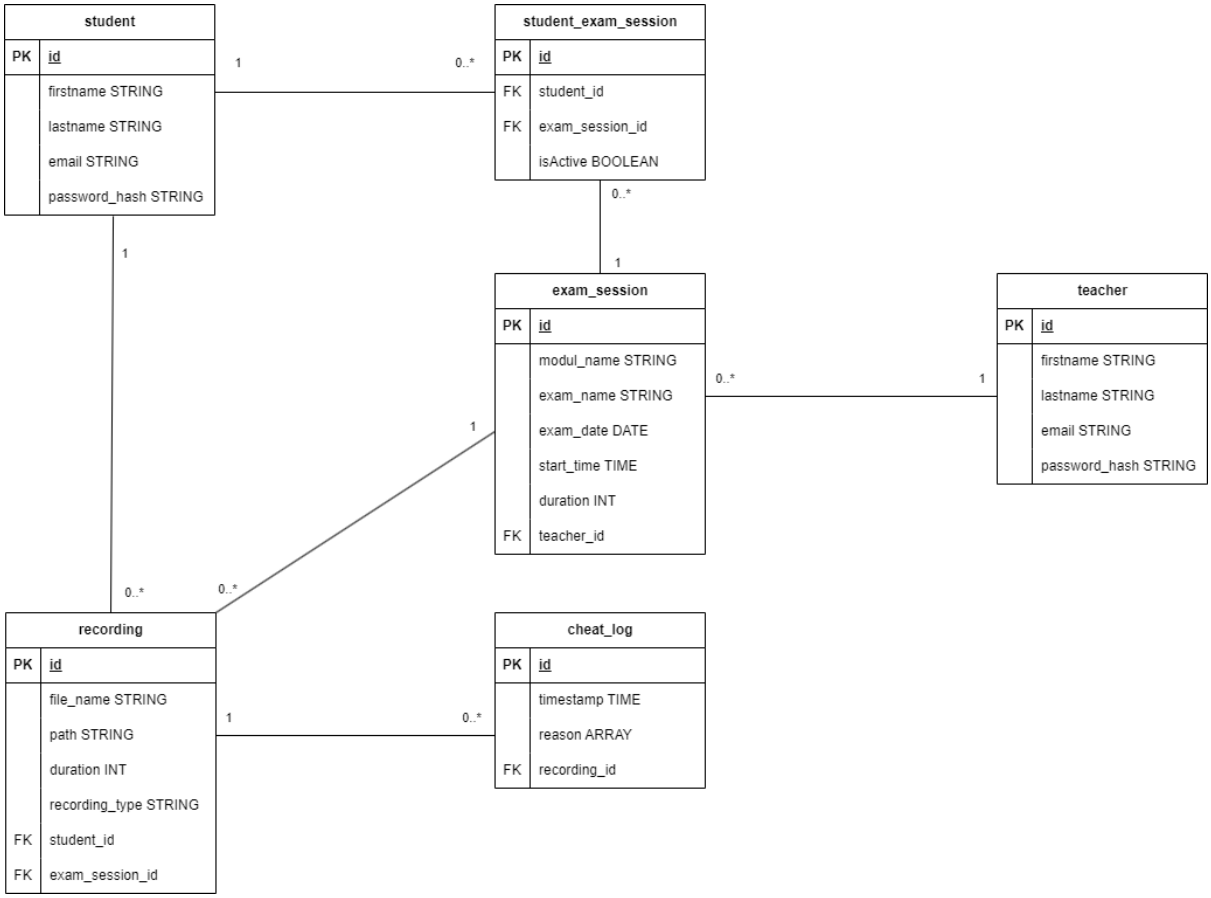


Abbildung 3.18: ERD Diagramm

3.2 Tools / Libraries

In diesem Abschnitt werden alle wichtigen Tools / Libraries beschrieben, welche verwendet werden um die einzelen Use Cases umzusetzten. Im Anschluss wird nachsstehend auch beschrieben, wieso diese Tools ausgewählt werden und wie diese zum Einsatz kommen.

Tabelle 3.4: Beschreibung Tools

Tool / Library	**Kurzbeschreibung**
PyAudio	PyAudio bietet eine Schnittstelle für den Zugriff auf Audiohardware. Damit kann Audio mit dem Mikrofon aufgenommen und verarbeitet werden.
SpeechRecognition	In Kombination mit PyAudio können Aufnahmen in Text umgewandelt werden.
OpenCV	OpenCV ist eine beliebte Bibliothek für Computer Vision und Bildverarbeitung. Damit können Videoaufnahmen sowie Bildverarbeitungstechniken verwendet werden.
TKinter	Mit Tkinter können grafische Benutzeroberflächen erstellt werden.
MediaPipe	Mediapipe ist ein Face Recognition Model, welches Gesichtspunkte einzeichnen und erkennen kann .
Flask	Flask ist ein Mikro-Webframework für Python, das sich besonders für kleine bis mittelgrosse Projekte sowie für die Erstellung von RESTful APIs eignet.
PostgreSQL	Postgres ist ein relationales Datenbanksystem, um Daten abzuspeichern. Die wird benötigt, damit die Auswertungen für das Dashboard angezeigt werden kann.

3.2.1 Tool-Einsatz und Anwendungsweise

Tabelle 3.5: Beschreibung Tools zu Use Case RecordSession

Use Case	RecordSession
Tools	• Tkinter
Ziel / Idee	Mithilfe von Tkinter kann eine grafische Benutzeroberfläche erstellt werden, damit die Studenten die Aufnahme starten und stoppen können. Es vereinfacht die Bedienung des Programms und hilft den Studenten Zeit zu sparen.

Tabelle 3.6: Beschreibung Tools zu Use Case RecordFacecam und FaceRecognition

Use Case	RecordFacecam und FaceRecognition
Tools	• Open-CV und Mediapipe
Ziel / Idee	OpenCV bietet eine Gesichtserkennungsfunktion, um Gesichter im Kamerabild zu erkennen. Dies kann mithilfe des Haar-Cascade-Algorithmus implementiert werden. Wenn eine zweite Person erkannt wird, kann ein entsprechender Log-Eintrag erstellt werden. Mediapipe hat eine Face Landmark function. Das bedeutet dass es in einem Gesicht die einzelnen Punkte erkennen kann.

Tabelle 3.7: Beschreibung Tools zu Use Case RecordMicrophone

Use Case	RecordMicrophone
Tools	• PyAudio • SpeechRecognition
Ziel / Idee	Mithilfe dieser beiden Tools kann die Sprache aufgenommen und in Text umgewandelt werden. Damit kann ein Schwellenwert für die Sprachaktivitätserkennung festgelegt werden. Niedrige Werte deuten darauf hin, dass gesprochen wurde.Bei Werten die den Schwellenwert überschreiten, wird das Gesprochene in Text umgewandelt und geloggt.

Kapitel 4: Qualitätsmassnahmen

4.1 Organisatorisch

4.1.1 Definiton of Done

Ein Feature oder eine User Story gilt als "done", wenn folgende Kriterien erfüllst sind:

- Der Code wurde geschrieben und alle Änderungen wurden im Git dokumentiert.
- Der Code wurde von einem anderen Entwickler überprüft und Feebacks wurden eingepflegt.
- Neue Tests wurden geschrieben, welche die Funktionalität des Features sinnvoll überprüfen.
- Alle Tests wurden erfolgreich ausgeführt und die Codeabdeckung ist angemessen.
- Alle notwendigen Dokumentationen wurden aktualisiert.

4.2 Tools zur Qualitätskontrolle

4.2.1 Linter

Als Linter wird Flake8 verwendet, welches sowohl für den Client- als auch für den Backend-Code eingesetzt wird. Flake8 ermöglicht es, einheitliche Codierungsstandards sicherzustellen und potenzielle Probleme im Code frühzeitig zu erkennen.

4.2.2 Code Coverage

Für die Testabdeckung kommt das Tool Pytest zum Einsatz, um die Anwendung gründlich zu testen. In der CI/CD-Pipeline wird ein Code-Coverage-Bericht erstellt. Dieser Bericht gibt einen klaren Überblick darüber, welche Teile des Codes durch die Tests abgedeckt werden und welche nicht. Er wird automatisch generiert und als Artefakt in Gitlab gespeichert. Die Berichte werden im HTML-Format zur Verfügung gestellt und können im Browser angezeigt und analysiert werden. Der Bericht ist im Anhang zu finden.

4.3 CI/CD

4.3.1 Dokumentation Repository

Das Dokumentations-Repository verfügt über keine CI/CD-Pipeline, da die Dokumentation über Overleaf erstellt wird. Mithilfe von Overleaf kann die Dokumentation kollaborativ bearbeitet werden. Zudem ist Overleaf mit unserem GitHub-Dokumentations-Repository synchronisiert. Falls die Overleaf-Website nicht verfügbar ist, kann auf die Dokumentation über GitHub zugegriffen und lokal daran gearbeitet werden.

4.3.2 Software Repository

Das Software-Repository ist in zwei Branches aufgeteilt. Der Main-Branch wird für stabile Releases der Applikation verwendet, während der Staging-Branch für tägliche Änderungen genutzt wird. Neue Funktionen werden über Feature-Branches implementiert, die mit "feature/"gekennzeichnet sind. Die daraus resultierenden Merge Requests werden von anderen Teammitgliedern überprüft. Die CI/CD-Pipeline dockerisiert eine PostgreSQL-Datenbank, den Analyse-Server und einen Flask-Server, der als Webserver fungiert. Die Codebasis wird in der CI/CD-Pipeline zunächst mit Flake8 überprüft. Anschliessend werden die Backend-Unit-Tests ausgeführt. Schliesslich gibt es auch einen Deployment-Schritt, der nur für den Main-Branch ausgeführt wird. In diesem Schritt wird die Pipeline über SSH mit einem On-Premise-Server verbunden und der Backend-Service (Flask, Analyse-Server und PostgreSQL) gestartet.

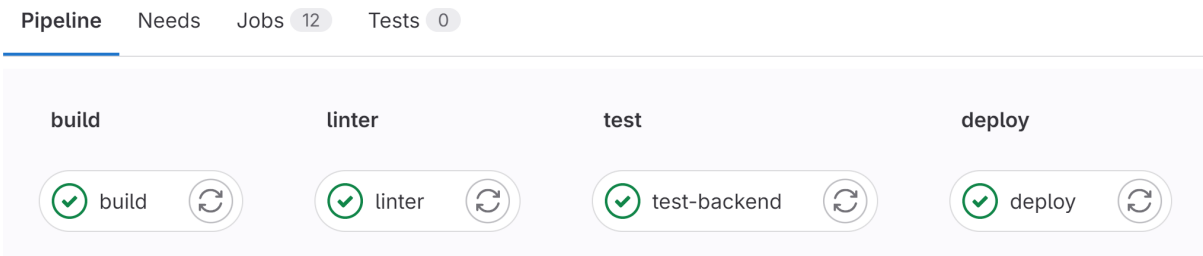


Abbildung 4.1: CI/CD Pipeline

4.4 Teststrategie

4.4.1 Unit Testing

Die Unit Tests für das Flask Backend werden entweder automatisch in einer GitLab CI/CD Pipeline ausgeführt oder manuell über ein Makefile. Das Ziel dabei ist, für alle Controller entsprechende Tests zu entwickeln, um nicht nur die korrekte Speicherung der Daten zu gewährleisten, sondern auch sicherzustellen, dass diese Daten richtig im Dashboard dargestellt werden.

4.4.2 End User Tests

Für die End User tests wurden vier Studierende gebeten, mit unserem Client eine Prüfung zu schreiben. Dazu wurde ihnen ein Login für ein Moodle-System eingerichtet. In Moodle musste eine von uns vorgegebene Prüfung absolviert werden, die eine Programmieraufgabe und vier Wahr-oder-Falsch-Fragen beinhaltete. Bevor die Studierenden die Prüfung begannen, mussten sie den Client starten, der für die Durchführung von Audio- und Videoaufnahmen verantwortlich war. Den Studierenden wurde mitgeteilt, dass sie auf jede erdenkliche Weise versuchen sollten zu schummeln, wobei das Übertragen des Prüfungsbildschirms oder das Öffnen neuer Programme oder Tabs auf demselben Bildschirm nicht gestattet war. Der Einsatz von Zweitgeräten oder zusätzlichen Bildschirmen war erlaubt und wurde auch genutzt.

Auswertung der Resultate

Die Auswertung der Ergebnisse erfolgte mittels einer Konfusionsmatrix. Für diese Analyse wurden die Videos erneut gesichtet und durch menschliche Beurteilung entschieden, welche Reaktionen die Sensoren in den jeweiligen Videoabschnitten zeigen sollten. Da diese Bewertung auf menschlichem Urteil basiert, ist es schwierig, die Objektivität und Korrektheit der Beurteilung in jedem Fall zu garantieren. Trotzdem wurden Fehlerberichte in den einzelnen Sensorbereichen detailliert analysiert. Die zusammengefassten Ergebnisse aus allen Video- und Audiodateien finden Sie im Bericht und die detaillierten Resultate jedes einzelnen Probanden im Anhang.

Es werden die Werte Genauigkeit, Präzision, Sensitivität und Spezifität berechnet, um genaue Aussagen über die Sensorenergebnisse zu treffen. Die Formeln sehen dabei folgendermassen aus:

$$\text{Genauigkeit} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Präzision} = \frac{TP}{TP + FP}$$

$$\text{Sensitivität} = \frac{TP}{TP + FN}$$

$$\text{Spezifität} = \frac{TN}{TN + FP}$$

Gesichtserkennung

Tabelle 4.1: Konfusionsmatrix Gesichtserkennung

	True	**False**
Positive	1	165
Negative	129	0

Genauigkeit: 130 / 295 = 44%
Präzision: 1 / 166 = 0%
Sensitivität: 1 / 1 = 100%
Spezifität: 129 / 294 = 44%

Diskussion: Bei der Auswertung der Ergebnisse zeigte sich, dass bei der Gesichtserkennung eine Genauigkeit von 44 Prozent erreicht wurde. Dieser Wert ist jedoch irreführend, da die Gesichtserkennung in einer Videoaufnahme nicht funktionierte und in jedem Fragment ein 'False Positive' auftrat. Das Versagen der Gesichtserkennung könnte darauf zurückzuführen sein, dass sowohl die Lichtverhältnisse als auch der Abstand zur Kamera eine wesentliche Rolle spielen. So funktionierte die Gesichtserkennung in dunklen Räumen nur sehr eingeschränkt, und bei einer zu nahen Position zur Kamera wurden ebenfalls einige 'False Positives' erzeugt.

Aufgrund der geringen Stichprobengrösse lassen sich nur begrenzte Aussagen über die Präzision und Sensitivität treffen. Jedoch legen die Ergebnisse nahe, dass die Wichtigkeit der Gesichtserkennung möglicherweise nicht so hoch ist, wie ursprünglich angenommen. In den analysierten 295 Videofragmenten gab es nur in einem einzigen Fall kein Gesicht im Bild, und es kam nie vor, dass mehr als ein Gesicht gleichzeitig erkannt wurde.

Ähnlich wie bei der Genauigkeit lässt sich auch über die Spezifität nur eingeschränkt urteilen. Insbesondere fiel eine Videoaufnahme mit schlechten Lichtverhältnissen auf, bei der eine hohe Anzahl an 'False Positives' die Spezifität in Frage stellt. Dies deutet darauf hin, dass das Gesichtserkennungsmodell empfindlich auf Umgebungsfaktoren reagiert. Faktoren wie die Entfernung zur Kamera und die Lichtverhältnisse spielen eine entscheidende Rolle für die Leistungsfähigkeit der Gesichtserkennung.

Munderkennung

Tabelle 4.2: Konfusionsmatrix Munderkennung

	True	**False**
Positive	84	44
Negative	142	25

Genauigkeit: 242 / 295 = 77%
Präzision: 84 / 128 = 66%
Sensitivität: 84 / 109 = 77%
Spezifität: 142 / 186 = 76%

Diskussion: Die Genauigkeit der Munderkennung beträgt 77 Prozent, wobei die meisten Fälle korrekt identifiziert wurden. Dies ist auf die Verwendung von Farbwerten der Landmark-Punkte durch den Sensor zurückzuführen. Allerdings wurden False Positives durch veränderliche Lichtverhältnisse oder starken Kopfbewegungen ausgelöst. Ein weiterer Faktor, der zu False Positives beitragen kann, sind Bärte. Einige Probanden führten Mundbewegungen durch, die dazu führten, dass ihre Lippen kleiner erschienen und das Modell irrtümlich die Landmark-Punkte der Bärte erfasste. Dies führte zu einer signifikanten Veränderung der Farbwerte und damit zur Auslösung eines False Positives. Die meisten False Negatives traten auf, wenn kurz gesprochen wurde, aber der Mund nur geringfügig geöffnet war. Dies schien eine zu kurze Distanz zwischen der Oberlippe und Unterlippe zu sein um einen True Positive zu erhalten. Um eine höhere Sensibilität zu erreichen, könnte die Distanztoleranz zwischen Ober- und Unterlippe verringert werden.

Mit einer Präzision von 66 Prozent ist die Leistung im Bereich der Munderkennung im Vergleich zu anderen Messwerten eher gering. Es ist herausfordernd, Verbesserungen für eine höhere Präzision zu identifizieren, da Faktoren wie starke Kopfbewegungen, Lichtverhältnisse und Bärte ausserhalb unserer Kontrolle liegen. Ein vielversprechender neuer Ansatz könnte die Entwicklung eines spezialisierten Modells sein, das ausschliesslich auf die Erkennung von Mündern trainiert ist. Dies würde die Abhängigkeit von Farbwerten reduzieren und ermöglichen, das Modell spezifisch für verschiedene Lichtverhältnisse zu trainieren.

Die Sensitivität des Tests liegt bei 77 Prozent. Wie bereits erwähnt, könnten False Negatives reduziert werden, indem der Schwellenwert für die Distanz zwischen Ober- und Unterlippe niedriger angesetzt wird. Diese Anpassung würde dazu beitragen, die Rate an falsch negativen Ergebnissen zu verringern.

Die Spezifität des Munderkennungssystems liegt bei 77 Prozent. Es zeigte sich, dass True Negatives also Fälle, in denen korrekterweise kein Mund erkannt wurde besonders gut bei klarer Sicht auf den Mund identifiziert werden konnten. Die Zuverlässigkeit der Munderkennung scheint besonders hoch zu sein, wenn der Mund nicht bewegt wird und die Sichtverhältnisse gut sind.

Iriserkennung

Tabelle 4.3: Konfusionsmatrix Iriserkennung

	True	**False**
Positive	128	31
Negative	101	35

Genauigkeit: 229 / 295 = 78%
Präzision: 128 / 159 = 81%
Sensitivität: 128 / 163 = 79%
Spezifität: 101 / 132 = 77%

Diskussion: Die Iris Analyse menschlich auszuwerten war am schwierigsten von allen Sensoren, da man sich manchmal unsicher war ob es sich in diesem Fall um Betrug handeln kann oder ob man sich an eine anderer Stelle in der Prüfung orientiert. Nichts destotrotz wurde während der Auswertung dieser Sensor als einer der wichtigsten angesehen. Da regelmässige Irisbewegungen in die gleiche Richtung schon auffällig erscheinen.

Die Genauigkeit der Iris-Analyse beträgt 78 Prozent. Es ist schwierig festzustellen, wie diese Genauigkeit gesteigert werden könnte, da die meisten False Positives und False Negatives aus Edge-Cases resultierten. Interessanterweise wurden die Versuche bei zwei Studenten, die stark schielten, stets korrekt erkannt. Ein bemerkenswerter Aspekt ist jedoch die Augenfarbe. Bei Personen mit dunklen Augen war die Erkennung von Edge-Cases weniger präzise, was zu Fehlentscheidungen führte. Im Gegensatz dazu schien die Iriserkennung bei Personen mit blauen Augen zuverlässiger zu sein. Ausserdem spielten die Lichtverhältnisse eine wesentliche Rolle: Bei guten Lichtverhältnissen war das Tracking effektiver als bei schlechten.

Die Präzision der Iris-Analyse liegt bei 81 Prozent, ein Wert, der als zuverlässig angesehen werden kann. Die meisten False Positives resultierten aus Edge-Cases, die teilweise auf menschliche Sichtfehler bei der Beurteilung oder starke Kopfbewegungen zurückzuführen sind. Konkret bedeutet das, dass bei starken Kopfbewegungen des öfteren ein positives Ergebnis erzielt wurde, selbst wenn sich die Iris dabei nicht entsprechend stark bewegt hat.

Die Sensitivität ist 79 Prozent und die Spezifität liegt bei 77 Prozent. Da diese Werte als Zuverlässig angesehen werden und sowohl der false Positive wie auch der false Negative Wert nicht speziell heraussticht oder verändert werden kann. Wird hier kein richtiger Ansatz gefunden diese Werte speziell zu verbessern.

Kopfneigungserkennung

Tabelle 4.4: Konfusionsmatrix Kopfneigungserkennung

	True	**False**
Positive	35	11
Negative	228	21

Genauigkeit: 263 / 295 = 89%
Präzision: 35 / 46 = 76%
Sensitivität: 35 / 56 = 64%
Spezifität: 228 / 239 = 95%

Diskussion: Die Kopfneigungserkennung wurde, ähnlich wie die Gesichtserkennung, nicht oft verwendet. Es gab die Annahme, dass dieser Sensor oft auslösen würde, da viele Leute einen zweiten Bildschirm benutzen, was häufiges Drehen des Kopfes mit sich bringt. Die Ergebnisse zeigen jedoch, dass die Probanden eher dazu neigten, mit der Iris zu schielen, anstatt den ganzen Kopf zu bewegen. Dies könnte darauf hindeuten, dass die Probanden wussten, dass ihre Kopfneigung verfolgt wurde, aber nicht ihre Irisbewegungen.

Die Genauigkeit weist einen Wert von 89 Prozent auf. Dieser Wert scheint vorallem durch die vielen True Negatives und den wenigen False Positives entstanden zu sein. Da sich die Köpfe während der Prüfung oft nicht gross bewegt haben, wurden viele True Negatives erkannt.

Die Präzision liegt bei 76 Prozent. Eine Möglichkeit, die Präzision zu steigern, besteht darin, die Anzahl der False Negatives zu reduzieren, was gleichzeitig die Anzahl der True Positives erhöhen würde. Bei den False Positives besteht hingegen wenig Spielraum für Verbesserungen, da das Verhältnis zwischen True Negatives und False Positives bereits sehr gut ist.

Die Sensitivität beträgt 64 Prozent und bietet somit Raum für Verbesserungen. Eine Möglichkeit, die Sensitivität zu steigern, besteht darin, den Schwellenwert für die Kopfneigung zu senken. Dies könnte dazu beitragen, die Anzahl der False Negatives zu reduzieren und genauere Ergebnisse zu erzielen. Es ist jedoch wichtig, darauf zu achten, dass der Schwellenwert nicht zu niedrig angesetzt wird, um einen Anstieg der False Positives zu vermeiden.

Die Spezifität beträgt 95 Prozent, was darauf hindeutet, dass die Erkennung von True Negatives sehr gut funktioniert. Der niedrige Schwellenwert für die Kopfneigung, kombiniert mit der niedrigen Sensitivität ist vermutlich für diesen hohen Wert verantwortlich. Dieser hohe Spezifitätswert zeigt, dass der Sensor zuverlässig arbeitet, wenn der Kopf nicht bewegt wird.

Audioerkennung

Tabelle 4.5: Konfusionsmatrix Audioerkennung

	True	**False**
Positive	30	0
Negative	237	7

Genauigkeit: 341 / 348 = 98%
Präzision: 30 / 30 = 100%
Sensitivität: 30 / 37 = 81%
Spezifität: 311 / 311 = 100%

Diskussion: Die Audioerkennung zeigte sich als das zuverlässigste System in unserem Setup. Dies ist hauptsächlich darauf zurückzuführen, dass keine spezifischen Parameteranpassungen oder speziellen Algorithmen verwendet wurden, sondern lediglich der API-Abruf der Google Speech Recognition Library. Aufgrund dieser einfachen Konfiguration gibt es kaum Möglichkeiten zur Optimierung oder Anpassungen. Eine mögliche Änderung könnte der Wechsel zu einem anderen Spracherkennungsmodell sein.

Die Genauigkeit des Systems liegt bei 98 Prozent. Ein möglicher Grund für die wenigen False Negatives könnte sein, dass der Google Sprachassistent Schwierigkeiten hat, Schweizerdeutsch zu verstehen und es oft mit Hochdeutsch verwechselt oder in dieses übersetzt hat. Für unsere Zwecke ist dies jedoch nicht problematisch, da es uns hauptsächlich darum geht, zu erkennen, ob überhaupt gesprochen wurde und nicht um den Inhalt des Gesprochenen. Es besteht jedoch die Vermutung, dass das Sprachmodell einige schweizerdeutsche Wörter nicht korrekt zuordnen konnte und sie daher fälschlicherweise als Lärm oder Hintergrundgeräusche interpretiert wurden, dies führte dann wahrscheinlich zu Nichterkennungen.

Die Präzision und Spezifität des Systems erreichen jeweils 100 Prozent. Bei solch optimalen Ergebnissen besteht kein Verbesserungsbedarf, und man ist mit dem Resultat vollständig zufrieden.

Die Sensitivität des Systems liegt bei 81 Prozent. Dies kann, wie zuvor vermutet, darauf zurückgeführt werden, dass einige False Negatives auftraten, weil gewisse Ausdrücke nicht korrekt übersetzt oder zugeordnet werden konnten. Eine mögliche Verbesserung der Spracherkennung könnte durch den Einsatz eines Modells erreicht werden, das speziell auf Schweizerdeutsch trainiert ist. Dies könnte dazu beitragen, den Sensitivitätswert zu erhöhen.

Da nur zwei der vier Probanden True Positives auslösten, ist es schwierig, eine verlässliche Aussage darüber zu treffen, ob die Werte bei einer grösseren Anzahl von Studenten konstant bleiben würden. Es besteht die Vermutung, dass die Ergebnisse möglicherweise etwas niedriger ausfallen könnten, wenn eine grössere Stichprobengrösse vorliegt. Trotz dieser Einschränkung lässt sich feststellen, dass das Modell insgesamt sehr zuverlässig funktioniert hat.

Teil III

Projekt Dokumentation

Kapitel 5: Projektplanung

5.1 Organisation und Ressourcen

Hier werden die Ressourcen, welche uns für dieses Projekt zur Verfügung stehen, beschrieben. Dazu gehört der Zeitliche Rahmen, die Kosten und die Personas.

5.1.1 Zeitlicher Rahmen

Das Projekt begann am 18. September 2023 und dauert bis Freitag, den 22. Dezember 2023. Während dieser Zeitspanne finden Projektreviews mit dem Projektleiter Frieder Loch statt. Bei den Reviews werden die Fortschritte vorgestellt und das weitere Vorgehen wird besprochen. Der Aufwand des Projektes soll dabei 17,1 Stunden pro Woche und Person betragen.

5.1.2 Kosten

Da es sich hierbei um ein Schulprojekt handelt, entstehen weder Personal- noch Ressourcenkosten. Die benötigte Soft-/Hardware wird sowohl von den Teilnehmenden als auch von der Schule zur Verfügung gestellt.

5.1.3 Personas

In diesem Abschnitt werden die Mitarbeiter des Projektes vorgestellt:

Name: Kevin Pfister
Studium: Informatikstudium im 7. Semester
Arbeitserfahrung: Hostpoint Customer Care Support, Quereinsteiger

Name: Nicolas Gattlen
Studium: Informatikstudium im 7. Semester
Arbeitserfahrung: Lehre in der Systemtechnik. Mehrere Jahre Arbeitserfahrung in der Systemtechnik + Cybersecurity

5.1.4 Arbeitsweise

Das Projekt wird nach der agilen Arbeitsweise durchgeführt. Aufgrund der unterschiedlichen Stundenpläne kann kein klar definiertes agiles Framework eingehalten werden. SCRUM+ wird jedoch als Orientierungshilfe verwendet.

5.1.5 Meetings

Es finden wöchentlich zwei Meetings am Freitag statt:

- Betreungsmeeting jeweils um 11:05 - 12:00
- Teammeetings jeweils um 13:00 - 14:00

In Teammeetings werden die getätigten Arbeiten, aufgetretenen Probleme und anstehenden Aufgaben besprochen. Die Aufgaben werden untereinander gleichmässig aufgeteilt. Das Meeting mit dem Betreuer dient dazu, den Projektfortschritt zu zeigen sowie das weitere Vorgehen zu besprechen. Nur Meetings mit dem Betreuer werden protokolliert und können im Abschnitt Anhang eingesehen werden.

5.2 Roadmap

Für die Gesamtplanung des Projekts wird der Rational Unified Process (RUP) verwendet. Das Projekt wird in vier Phasen unterteilt, wobei jede Phase mehrere Iterationen (Sprints) haben kann. Eine Iteration dauert eine Woche. Eine detailierte Roadmap inklusiv der definierten Arbeitspaktete pro Phase ist im Anhang abgelegt.

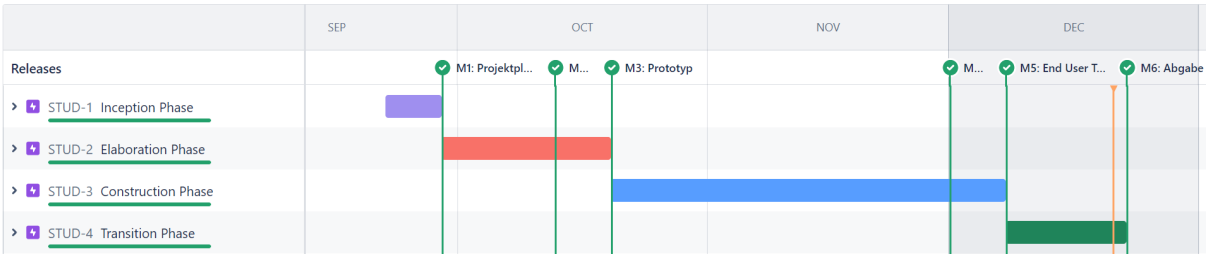


Abbildung 5.1: Roadmap

Tabelle 5.1: Daten und Dauer der einzelnen Phasen

Phase	Startdatum	Enddatum	Dauer (Tage)
Inception	22.09.2023	28.09.2023	7
Elaboration	29.09.2023	19.10.2023	21
Construction	20.10.2023	07.12.2023	49
Transition	08.12.2023	22.12.2023	14

5.2.1 Phasen

In diesem Abschnitt werden die einzelnen Phasen beschrieben. Dabei werden die Hauptaufgaben der jeweiligen Phase definiert. Zum Schluss jeder Phase werden Meilensteine gesetzt und kontrolliert. Diese dienen dazu, den aktuellen Fortschritt zu überprüfen und einen Überblick darüber zu bekommen, wo wir stehen.

Inception

In der Inception-Phase sollen alle organisatorischen Aspekte soweit bearbeitet werden, dass die Arbeit danach ohne organisatorische Unterbrechungen fortgesetzt werden kann. Um uns einen Überblick über unsere Vision zu verschaffen, sollen in der Inception-Phase erste Use Cases definiert werden und die Anforderungen sollen einen ersten Einblick in das Projekt geben. Ausserdem sollte der Produktplan sowohl kurzfristig als auch langfristig erstellt sein. Für die Inception- und Elaboration-Phasen sollte ein Short-Term-Plan bereitstehen. Dieser soll, wo nötig, von Phase zu Phase angepasst werden.

Elaboration

In dieser Phase sollen die nicht-funktionalen Anforderungen definiert werden und gegebenenfalls die funktionalen Anforderungen der Inception-Phase verfeinert werden. Neben den Anforderungen soll das Domain-Modell erstellt werden. Ausserdem soll eine Recherche bezüglich bereits bestehender Proctoring-Software gemacht werden.

Construction

Während dieser Phase werden die Use Cases umgesetzt, um das Produkt zu entwickeln. Ausserdem findet ein Beta-Release statt, um zu demonstrieren, dass die Use Cases erfolgreich umgesetzt wurden. In der vorletzten Woche wird eine Simulation durchgeführt, bei der das Produkt vom Benutzer getestet wird, um sicherzustellen, dass das Proctor-System effektiv betrügerische Aktivitäten erkennen kann.

Transition

Die Transition Phase erstreckt sich über zwei Wochen, wobei die erste Woche als Puffer dient, der im Falle eines unerwarteten Bedarfs während der Construction Phase genutzt werden kann. In der letzten Woche steht die abschliessende Dokumentation im Mittelpunkt, um sicherzustellen, dass sie für die Abgabe bereit ist.

5.2.2 Meilensteine

Im Projekt werden die folgenden 6 Meilensteine definiert. Zudem werden in der untenstehenden Tabelle alle Artefakte/Ziele aufgeführt, die bis zum Enddatum des entsprechenden Meilensteins abgearbeitet werden.

Tabelle 5.2: Meilensteine

Meilenstein	**Quality Gates**	**Termin**
M1 - Projektplan	• Es müssen mindestens 2 funktionale Anforderungen definiert und abgenommen werden. • Bis zum 28.09.2023 muss ein Use Case Diagramm erstellt und beschrieben sein. • Es müssen mindestens 3 Risiken aufgelistet und eine Risikoanalyse durchgeführt werden, die bis zum 28.09.2023 dokumentiert sein muss.	28.09.2023
M2 - Recherche	• Bis zum 5.10.203 muss ein Domain Model erstellt und dokumentiert sein. • Es werden mindestens 3 Proctoring-Software verglichen und dokumentiert. • Bis zum 12.10.2023 müssen geeignete Technologien/Tools recherchiert und dokumentiert sein. • Mindestens 3 nicht-funktionale Anforderungen erstellen und abnehmen lassen.	12.10.2023
M3 - Prototyp	• Bis zum 19.10.2023 muss eine Entwicklungsumgebung eingerichtet sein, damit in der Konstruktionsphase sofort entwickelt werden kann.	19.10.2023
M4 - Beta Release	• Bis zum 23.11.2023 müssen alle Use-Cases sowie die technischen Lösungsansätze dokumentiert sein.	30.11.2023
M5 - Endbenutzertests	• Es werden mit mindestens 3 Personen Endbenutzertests durchgeführt.	7.12.2023
M6 - Abgabe	• Bis zum 22.12.2023 müssen die Dokumentation sowie der Softwarecode abgeschlossen und abgegeben sein.	22.12.2023

5.3 Issue Management

Als Issue-Management-Tool wird Jira verwendet, welches einfach zu bedienen ist und bei dem sich alle Funktionen und Methoden auf einer Plattform befinden. Ausserdem soll das Projekt eine definierte Struktur aufweisen.

5.3.1 Issues

Im Jira gibt es verschiedene Typen von Issues, darunter Story, Epic, Task und Subtask. Im Folgenden wird beschrieben, wie diese Typen in diesem Projekt verwendet werden.

Tabelle 5.3: Beschreibung der verschiedenen Typen von Issues

Issuetypen	**Beschreibung**
Epics	Epics werden als Phasen des Gesamtprojekts dargestellt.
Story	Stories werden verwendet, sobald sich eine Anforderung oder eine Funktion auf eine spezifische Benutzeraktion oder ein Benutzerziel fokussiert.
Task	Tasks werden verwendet, wenn es sich um allgemeine Aufgaben handelt.
Subtask	Falls eine Story oder Task weiter unterteilt werden muss, kann ein Subtask dafür erstellt werden.

Die Issues in Jira werden hierarchisch strukturiert. In diesem Projekt werden Stories und Tasks einem Epic zugeordnet. Subtasks hingegen werden ausschliesslich Stories und Tasks zugewiesen.

5.3.2 Components

Jeder Issue wird einem Component zugeordnet, der dazu dient, Aufgaben zu gruppieren. Mit anderen Worten kann ein Component als Tag betrachtet werden. Nachfolgend sind die definierten Components aufgeführt. Die Liste wird regelmässig aktualisiert. Zusätzlich werden die Components für die Zeitauswertung benötigt.

5.3.3 Jira Kanban Board

Das Jira Kanban Board dient als Hilfestellung, um den Zustand und den Fortschritt eines Issues / Tasks zu überwachen, sowie auch die Erstellung von Tasks jeder Iteration vom Teammeetings.

5.4 Risikoanalyse

5.4.1 Risikomatrix

Tabelle 5.4: Risikomatrix

Auswirkung / Wahrscheinlichkeit	Schwach - 1	Minimal - 2	Moderat - 3	Hoch - 4	Schwer - 5
Sehr hoch - 4	(leer)	R4	(leer)	(leer)	(leer)
Hoch - 3	(leer)	R3	(leer)	(leer)	(leer)
Mittel - 2	(leer)	(leer)	(leer)	R1	(leer)
Niedrig - 1	(leer)	(leer)	R2	(leer)	(leer)

5.4.2 Risiken

In dieser Tabelle werden alle möglichen Risiken die im Projekt auftreten können aufgelistet.

Tabelle 5.5: Risiken

Nr.	**Risiko**	**Wahrscheinlichkeit**	**Auswirkung**
R1	Finden von guten Opensource AI Models	2	4
R2	Implementierung von bestehenden Models	1	3
R3	Testing der Software bei End-User	3	2
R4	Vereinbarung der zu benutzenden Technologien	4	2

5.4.3 Risikominderung

In diesem Abschnitt wird beschrieben, wie die oben genannten Risiken vermindert werden können.

R1: Finden von guten Opensource AI Models

Da wir keine Zeit haben eigene AI Modells zu entwerfen, greifen wir auf Opensource Modelle zurück. Die Erfahrung aus früheren Modulen hat jedoch gezeigt, dass nicht alle Opensource Modelle das erkennen, was Sie erkennen sollen. Um dieses Risiko zu vermindern, werden mehrere Modelle miteinander verglichen und das beste wird anschliessend implementiert.

R2: Implementation von bestehenden Models

Die Proctoring Software trackt viele vereinzelte Aktivitäten. Damit diese alle miteinander kombiniert werden können, braucht es eine saubere Implementation einzelner Models. Um das Risiko zu mindern ein mögliches Chaos zu verursachen, ist ein regelmässiger Austausch und ein sog. Pair programming wichtig. Damit wissen alle Beteiligten, wie die einzelnen Modelle miteinander arbeiten.

R3: Testing der Software bei End-User

Um zu überprüfen, ob die fertige Software funktioniert. Muss Sie bei End-Usern getestet werden. Für die Erstellung eines kritischen Reports, muss es eine grosse Anzahl an Test-Studenten geben, welche bereit sind unsere Software in einer Prüfungsumgebung zu testen. Um dies zu erreichen werden mehrere Testläufe mit verschiedenen Schülern durchgeführt. Dieses Risiko wird vermindert, indem der Betreuer uns ein paar seiner Arbeitskräfte für das Testing zur Verfügung stellt und Freunde aus dem Studium genug früh gefragt werden.

R4: Vereinbarung der zu benutzenden Technologien

Da wir beide keine Erfahrung mit der Erstellung einer Proctoring Software haben. Wird das Finden geeigneter Technologien ein grosses Risiko darstellen. Durch erste Erfahrungen von AI Modell Implementationen im Modul AI Applications wissen wir ungefähr was uns erwartet, wenn es um die Implementation der Open Source Modelle geht. Durch eine zusätzliche Recherche kann das Risiko hier vermindert werden.

Kapitel 6: Zeiterfassung

6.1 TimeTracker

Die Zeiterfassung wird durch eine Third Party App namens TimeTracker verwaltet, die mit Jira kompatibel ist. Anschliessend werden die Daten von Jira exportiert und mit einem Python Skript wird ein grafischer Zeitreport erstellt. Jeder Zeitreport zeigt den Ist- und Sollzustand.

6.2 Auswertung

Die Zeiterfassung wird vor den Betreungsmeetings aktuallisiert und jeweils als Screenshot in diesem Kapitel hinzugefügt.

6.2.1 Auswertung Gesamtprojekt

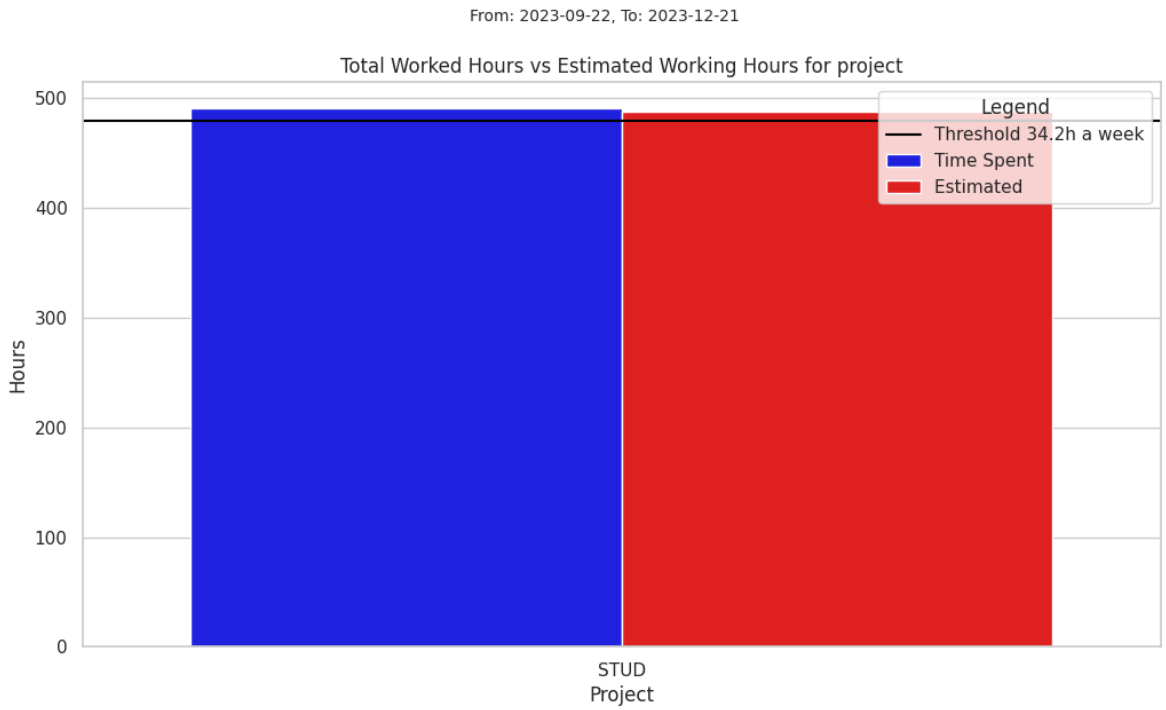


Abbildung 6.1: Auswertung Gesamtprojekt

6.2.2 Auswertung pro Phase

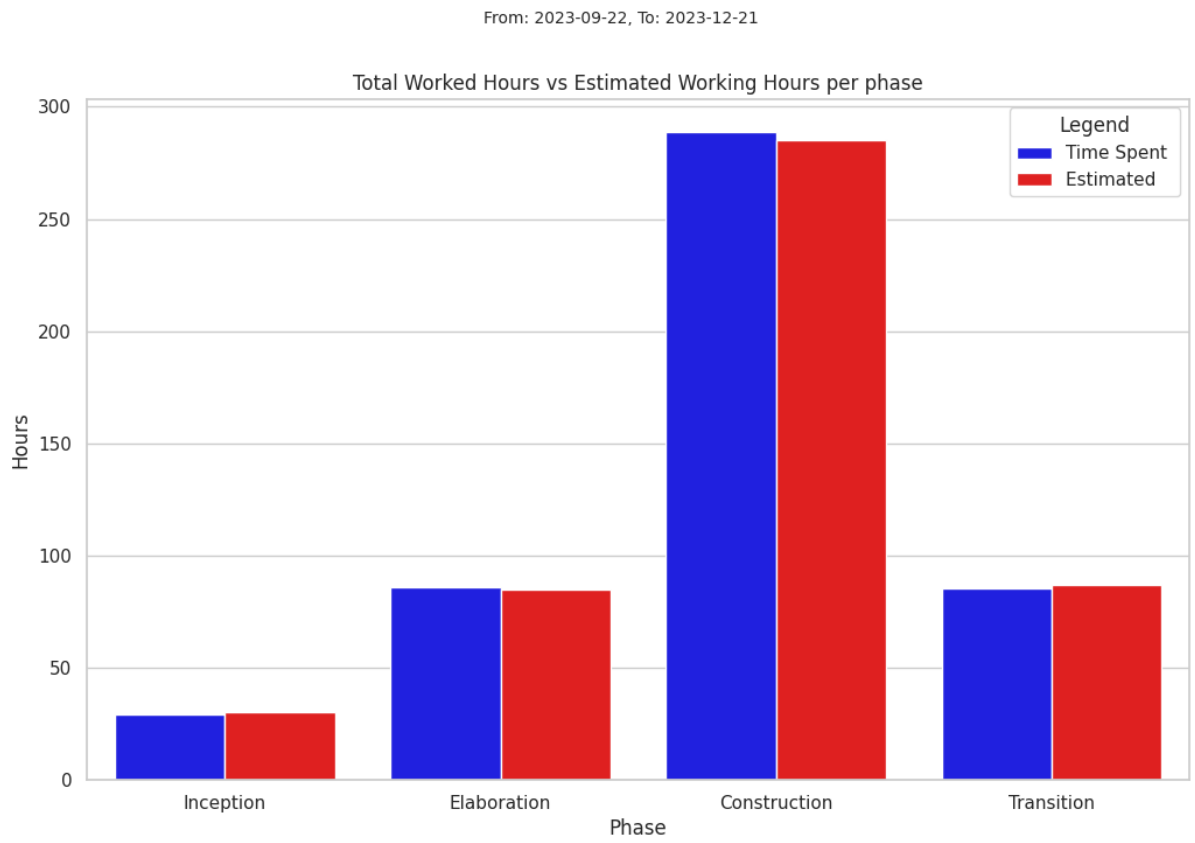


Abbildung 6.2: Auswertung pro Phase

6.2.3 Auswertung pro Person

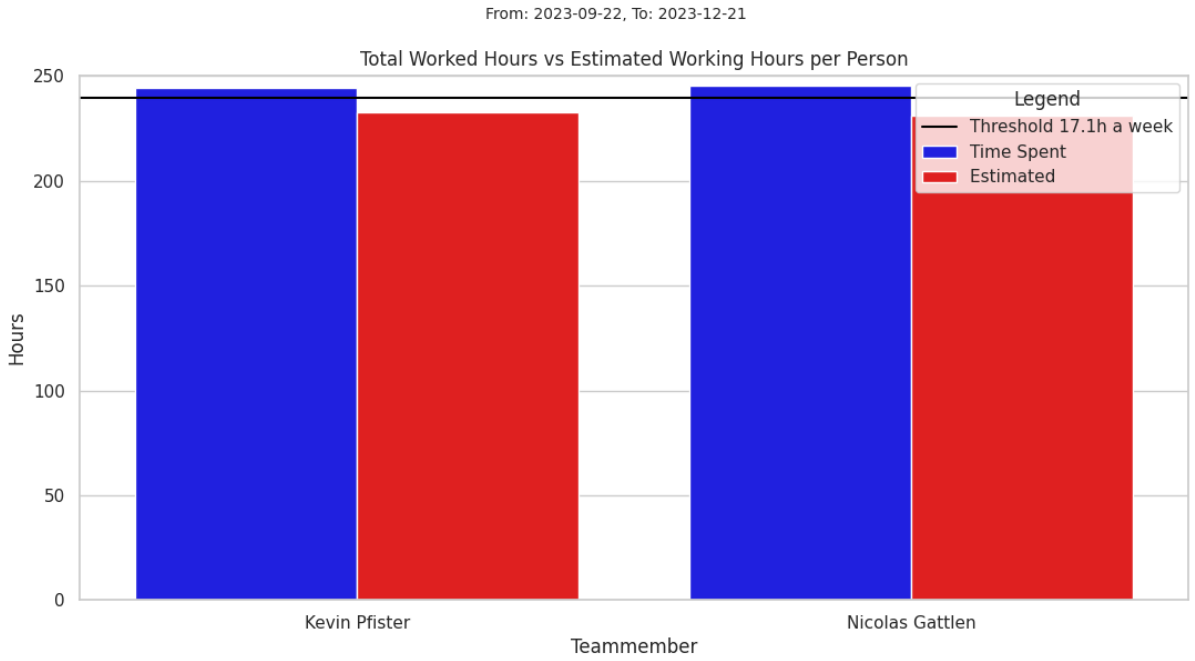


Abbildung 6.3: Auswertung pro Person

Kapitel 7: Personal Reports

7.1 Kevin Pfister

Die Arbeit an einer Online-Proctoring Lösung war eine spannende und interessante Erfahrung. Besonders fasziniert hat mich die Implementierung von Technologien wie MediaPipe, OpenCV und dem Google Speech Assistant. Die Möglichkeit, Informationen über Facelandmark-Punkte zu erhalten und daraus Sensoren zu entwickeln, war eine herausfordernde und kreative Aufgabe. Da die Erstellung eigener Modelle aufgrund von Zeit- und Ressourcenbeschränkungen nicht möglich war, mussten wir auf alternative Ansätze wie Farbwerte oder Referenzpositionen zurückgreifen. Auf diese innovativen Lösungen bin ich besonders stolz, und die mehrheitlich zufriedenstellenden Resultate waren sehr ermutigend.

Persönlich habe ich festgestellt, dass ich noch Schwierigkeiten mit dem organisatorischen Tracking von Aufgaben habe. Ich habe es oft versäumt, abgeschlossene Tasks in Jira zu aktualisieren und das Zeiterfassungssystem zu nutzen. Dies möchte ich in Zukunft verbessern, da eine gute Projektorganisation entscheidend für einen klaren Überblick über den Arbeitsfortschritt ist.

Die Unsicherheiten bezüglich der Implementierung der Sensoren entstanden vor allem dadurch, dass ich die Tests der Sensoren ausschliesslich selbst durchführen konnte. Daher war ich mir unsicher, ob sie auch in den Umgebungen anderer Personen zuverlässig funktionieren würden, da diese variieren könnten. Es war jedoch beruhigend zu sehen, dass in der Testphase die meisten Ergebnisse positiv ausfielen.

Mein persönliches Highlight war das Testen und die Präsentation der Software. Es war ein besonderer Moment, unserem Betreuer die funktionierende Software und die erkannten Betrugsfälle vorzustellen und damit einen Beitrag zu Lösungen für Online-Proctoring zu leisten.

7.2 Nicolas Gattlen

Das Projekt fand ich sehr spannend und die Teamarbeit war sehr angenehm. Während des gesamten Projekts lagen wir gut im Zeitplan, allerdings gab es auch Situationen, in denen wir auf Probleme stiessen und mehr Zeit investieren mussten als geplant. Um diese Probleme effizient zu lösen, setzten wir auf Pair Programming. Dadurch konnten Fehler schneller behoben werden und die verlorene Zeit konnte wieder aufgeholt werden.

Während der Recherchephase nahmen wir Kontakt mit verschiedenen Firmen auf, um nachzufragen, ob wir eine Demoversion ihrer Proctoring-Software zum Testen erhalten könnten. Leider stellten die Firmen uns ihre Software nicht zur Verfügung, da sie diese nur im Falle eines Kaufs durch die OST bereitstellen würden. Von einigen Firmen erhielten wir aus verschiedenen Gründen keine Antwort. Eine Firma hat uns zu einem Webmeeting eingeladen. Zwar konnten sie uns ihre Proctoring-Software nicht direkt vorführen, erklärten uns aber deren Einsatz und Verwendung. Im Anschluss an die Präsentation konnten wir Fragen stellen, die alle beantwortet wurden.

Wenn ich das Projekt noch einmal von vorne beginnen könnte, würde ich eine Sache ändern. Aktuell dient das Backend auch als Frontend. Bei einem neuen Ansatz würde ich Frontend und Backend trennen. Der Grund, warum wir es nicht so implementierten konnten, war das wir React und Angular noch am lernen waren und wir uns unsicher fühlten diese im Projekt einzusetzen.

Auch haben wir gelernt, dass die erste Lösung nicht immer die Beste ist. Für den Datentransfer haben wir Sockets verwendet, um die Aufnahmen vom Client zum Server transferieren zu können. Dies hat funktioniert, jedoch hat es eine Weile gedauert, bis die ankamen. In der Testingphase haben zwei Studierende versucht ihre Aufnahmen an unserem Server zu senden, jedoch kamen diese nie an aufgrund der 100 Prozent Auslastung der CPUs von unserem virtuellen Server. Als alternative haben wir SFTP verwenden und haben auch festgestellt, dass dies eine bessere Lösung ist als Sockets zu verwenden. Mit SFTP kamen die Aufnahmen sicher und schneller an.

Insgesamt hat es Spass gemacht an diesem Projekt zu arbeiten. Ich konnte viel daraus lernen. Auch wenn es Situationen gibt in denen man nicht weiterkommt, sollte man versuchen Hilfe zu holen, um das Problem gemeinsam zu lösen.

Teil IV

Appendix

Kapitel 8: Anhang

8.1 Meeting Protokoll

8.1.1 Meeting 1 - 22. September 2023 11:00 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	20min

Besprechung

- Fragen zur Aufgabenstellung besprochen:
  - Was sollte genau detektiert werden? → Maus, Tastatur und Bildschirm
  - Gibt es bereits Daten/Aufnahmen? → Daten müssen selbst gesammelt werden
  - Dürfen Use Cases ohne ML abgedeckt werden bsp Secure Broswer ? → Ja
- Für Anfang ein Use Case definieren → kann erweitert werden während Projekt
- Zeiterfassung generell halten
- Prüfungsformat Beispiel Programmieraufgabe
- Machine Learning sollte auch im nachhinein funktionieren
- Logs sollten gefiltert werden → z.B von 100 Studenten auf 5 reduziert

---

Nächste Ziele

- Use Cases brainstormen
- Risikomatrix erstellen
- Projektplanung mit Meilensteine definieren

Nächstes Meeting

29. September 2023 um 11:05 Uhr

8.1.2 Meeting 2 - 29. September 2023 11:05 Uhr

Ort	Gebäude 1 - Raum 1.171
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Use Case Diagram besprochen
- Projektplanung besprochen
  - Meilensteine anpassen → Wie quantifizieren ?
- Risikoanalyse besprochen
  - Risioktabelle anpassen → Wie quantifizieren ?
  - Als Orientierung Screenshot anschauen
- Konkreter Prüfungsverlauf beschreiben

Nächste Ziele

- Domain Model erstellen
- Risikotabelle anpassen gemäss Template
- Meilensteine messbar machen
- Prüfungsverlauf beschreiben
- Marktführer der Proctoring Software suchen
- Kategorien für die Bewertung aussuchen bezüglich Proctoring Software vergleich
- Sequenzdiagramm für Proctoring System

Nächstes Meeting

6. Oktober 2023 um 11:05 Uhr

8.1.3 Meeting 3 - 6. Oktober 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Research Paper besprochen
- Moodle Test besprochen
- Softwarevergleich
  - Softwarelizenz verspätung, Aufgrund keiner Rückmeldungen von Firmen

---

Nächste Ziele

- 1 - 2 weitere Research Papers durchlesen und zusammenfassen
- Software vergleichen und dokumentieren
- Server bestellen

Nächstes Meeting

13. Oktober 2023 um 11:05 Uhr

8.1.4 Meeting 4 - 13. Oktober 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Bestehende Dokumentation schicken
- Moodle Test besprochen
  - Soll Student dazu aufzufordern zu betrügen während der Durchführung des Tests
  - Da keine "Tutorials"vorhanden waren, wurde mehr Zeit benötigt um Tests einzurichten
- Softwarevergleich
  - Softwarelizenz verspätung, aufgrund keiner Rückmeldungen von Firmen

---

Nächste Ziele

- Vielleicht auf ein bestimmtem Use-case fokusieren um Komplexität einzuschränken ?

Nächstes Meeting

20. Oktober 2023 um 11:05 Uhr

8.1.5 Meeting 5 - 20. Oktober 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Proctoring Software von Firmen
  - Schwer eine Demo Software zu erhalten, da Firma B2B bevorzugen.
  - Ein Meeting mit ProctorEdu und konnten Fragen dazu stellen
- Use-Case
  - Fokus wird auf Kamera gesetzt → Head Tracking, Iris Tracking, Person Detection
- ProctorEdu Moodle Plugin
  - Anleitung um ProctorEdu Software in Moodle einzubinden
  - Da wir keine Berechtigung haben, wird Loch anschauen und abklären

---

Nächste Ziele

- Authentication für Dozent einrichten
- Warten auf Rückmeldung von Loch wegen ProctorEdu Moodle Plugin
- Kameraeinbindung vorberereiten

Nächstes Meeting

27. Oktober 2023 um 11:05 Uhr

8.1.6 Meeting 6 - 27. Oktober 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Authenication für Dozent eingerichtet
  - Demo Login Seite gezeigt
  - Loch hat Vorgeschlagen, Wireframes zu erstellen für Design der Webseite
- Dokumentation
  - Loch wird Dokumentation durchlesen und Feedback geben übers Wochenende
- Client - Kamera
  - Kamera wurde erstellt aber Länge des Video ist nicht realtime

---

Nächste Ziele

- Wireframe erstellen für Website und abnehmen lassen
- Backend so schnell wie möglich einrichten für Kevin
- Kameraeinbindung vorberereiten

Nächstes Meeting

03. November 2023 um 11:05 Uhr

8.1.7 Meeting 7 - 03. November 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Client - Kamera
  - Kamera gefixxt, Videos haben jetzt die gleiche Länge wie die Dauer der Aufnahme
  - Mit ersten Facedetection Models gespielt.
- Wireframes
  - Wireframes gezeigt und abgenommen lassen

Nächste Ziele

- Dashboard - Exam Session Page entwickeln
- Mikrofonanbindung
- Sprachassistent einrichten

Nächstes Meeting

10. November 2023 um 11:05 Uhr

8.1.8 Meeting 8 - 10. November 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	N.Gattlen und K.Pfister
Entschuldigungen	F. Loch
Dauer	30 min

Besprechung

- Client - Mikrofon
  - Mikrofon wurde hinzugefügt + Sprachassisten wurde eingebaut
- Dashboard
  - Demo Exam Session Page

Nächste Ziele

- Einbindung der Mouth detection
- Dashboard Studen Exam Session Page einrichten
- Dashboard Incident Page einrichten

Nächstes Meeting

17. November 2023 um 11:05 Uhr

8.1.9 Meeting 9 - 17. November 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	N.Gattlen und K.Pfister
Entschuldigungen	F. Loch
Dauer	30 min

Besprechung

- Ausgefallen, da Dozent im Ausland ist

Nächste Ziele

- Weiter an der Construction Phase arbeiten

Nächstes Meeting

24. November 2023 um 11:05 Uhr

8.1.10 Meeting 10 - 24. November 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	N.Gattlen und K.Pfister
Entschuldigungen	F. Loch
Dauer	30 min

Besprechung

- Client
  - Mouthdetection wurde vollständig implementiert und Iris detection wurde auch implementiert
- Dashboard
  - Demo Student Exam Session Page
  - Demo Incident Page
  - Demo Session Page

Nächste Ziele

- Klassen zusammenfügen und Daten für das Backend bereitstellen
- Client Bereit machen für Probanden
- Datenanbindung

Nächstes Meeting

1. Dezember 2023 um 11:05 Uhr

8.1.11 Meeting 11 - 1. Dezember 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	20 min

Besprechung

- MVP Produkt
  - Client ist Fertig und Produkt ist bereit für das Testing
  - MVP Produkt, war nicht zu 100% fertig, da viel Zeit investiert wurde um den Client zu debuggen

---

Nächste Ziele

- Testpersonen suchen für die Simulation
- Bedienungsanleitung für Testpersonen schreiben
- Testing des Servers + Analyse Skripts

Nächstes Meeting

8. Dezember 2024 um 11:05 Uhr

8.1.12 Meeting 12 - 8. Dezember 2023 11:05 Uhr

Ort	Geb aude 1 - Raum 1.171
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	60 min

Besprechung

- Auswertung Testergebnisse - Procotring System
- Vorführung der Software
  - Demonstration der Auswertungen von verschiedenen Studenten inklusiv Aufnahmen angeschaut
  - Erklärt, welche Sensoren gute Ergebnisse erzielt haben und welche als besonders empfindlich eingestuft wurden
  - Erwähnt, dass virtueller OST Server Ressourcenprobleme hatte
    - Videos kamen nicht an, da CPU voll ausgelastet war
    - Virtueller Server mehr Ressourcen zugewiesen
  - Um Filetransfer zu beschleunigen wurde SFTP in Client Program eingebaut
- Client Application
  - Demonstration des Client Application

Nächste Ziele

- Dokumentation Abschnitt Architektur beschreiben
- Testergebnisse dokumentieren, sobald alle Studenten Tests durchgeführt haben

---

Nächstes Meeting

15. Dezember 2024 um 11:05 Uhr

8.1.13 Meeting 13 - 15. Dezember 2023 11:05 Uhr

Ort	Microsoft Teams
Teilnehmer	F. Loch, N.Gattlen und K.Pfister
Entschuldigungen	Keine
Dauer	30 min

Besprechung

- Dokumentation
  - C4-Diagramme vorgestellt und gezeigt
  - Abschnitt Architektur gezeigt und überflogen
  - Dozent wird Dokumentation über das Wochenende durchlesen und Feedback dazu geben
- OST Abstrakt
  - Abstrakt fertig geschrieben und zur Abgabe bereitgestellt
  - Dozent wird den Abstract durchlesen und kontrollieren

Nächste Ziele

- Dokumentation abschliessen und für die Abgabe vorbreiten

Nächstes Meeting

22. Dezember 2024 um 11:05 Uhr

8.2 Wireframes

8.2.1 Login Page

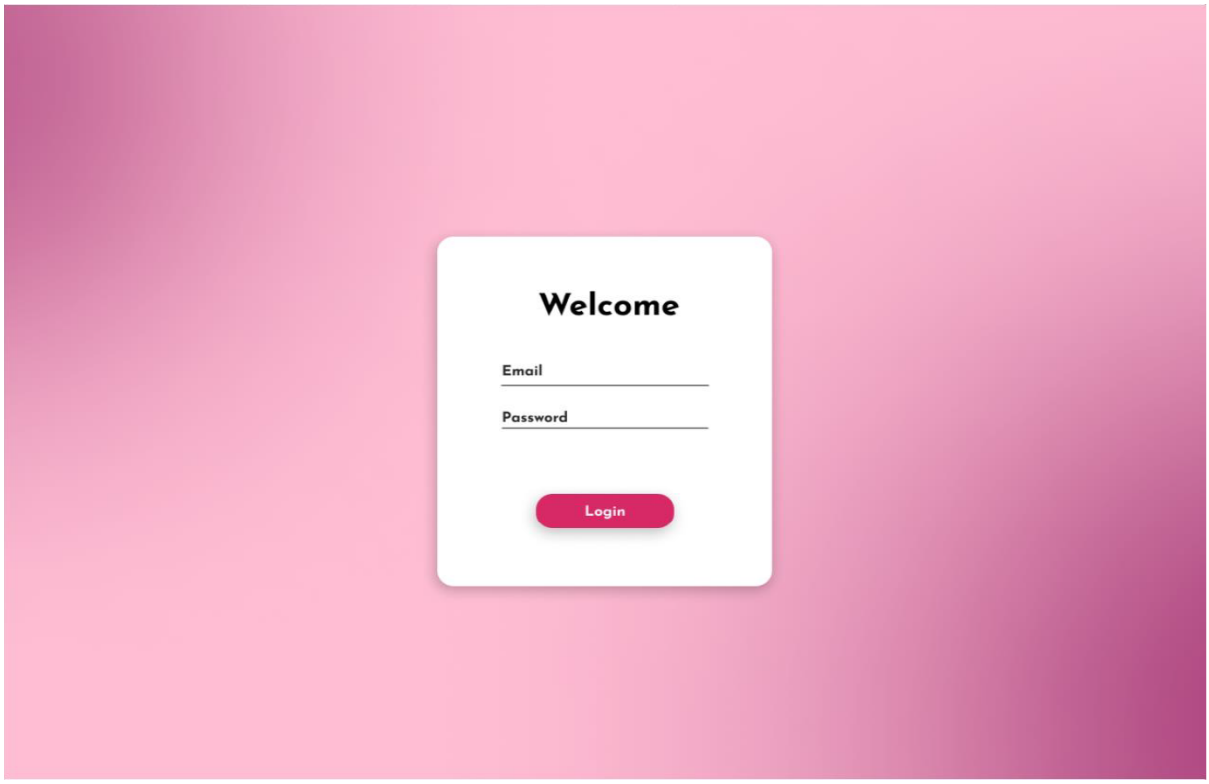


Abbildung 8.1: Wireframe - Login Page

8.2.2 Exam Session Page

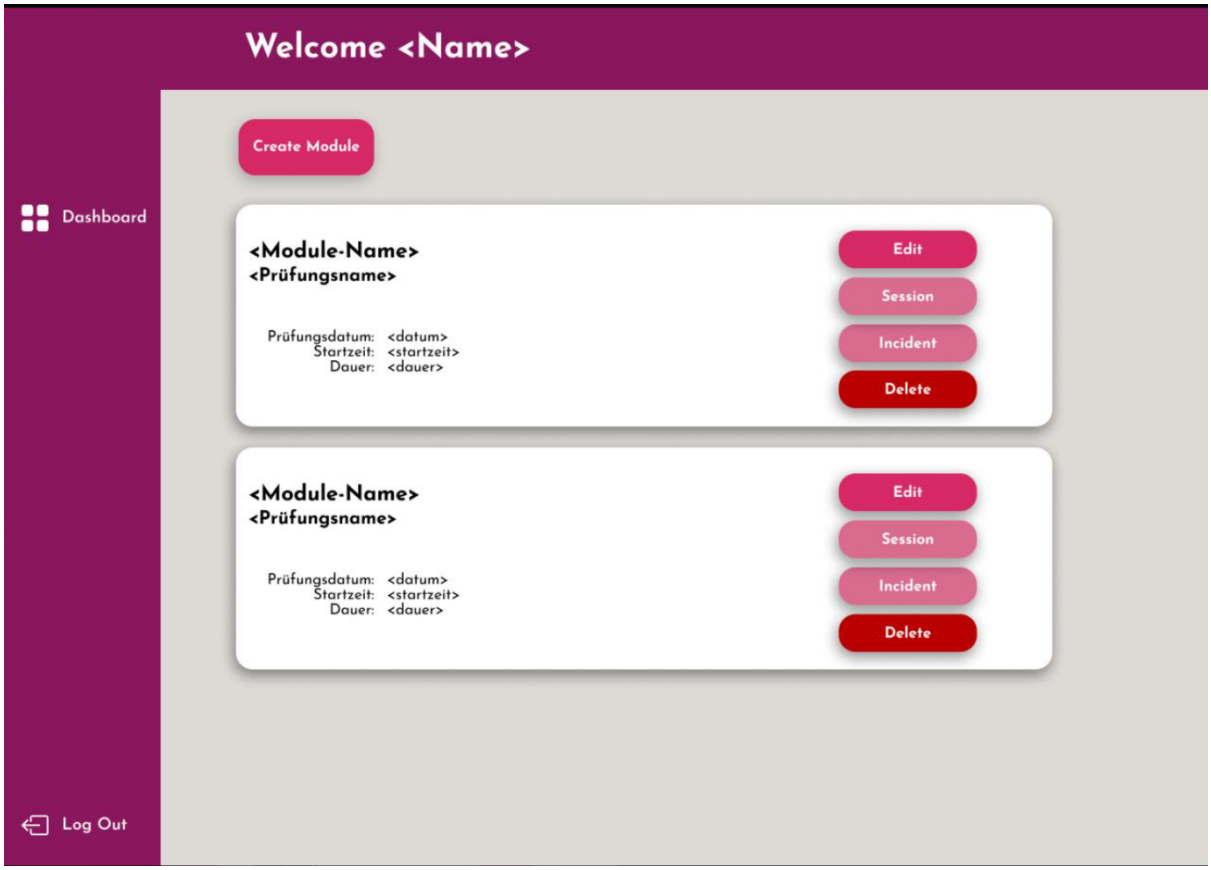


Abbildung 8.2: Wireframe - Exam Session Page

8.2.3 Session Page

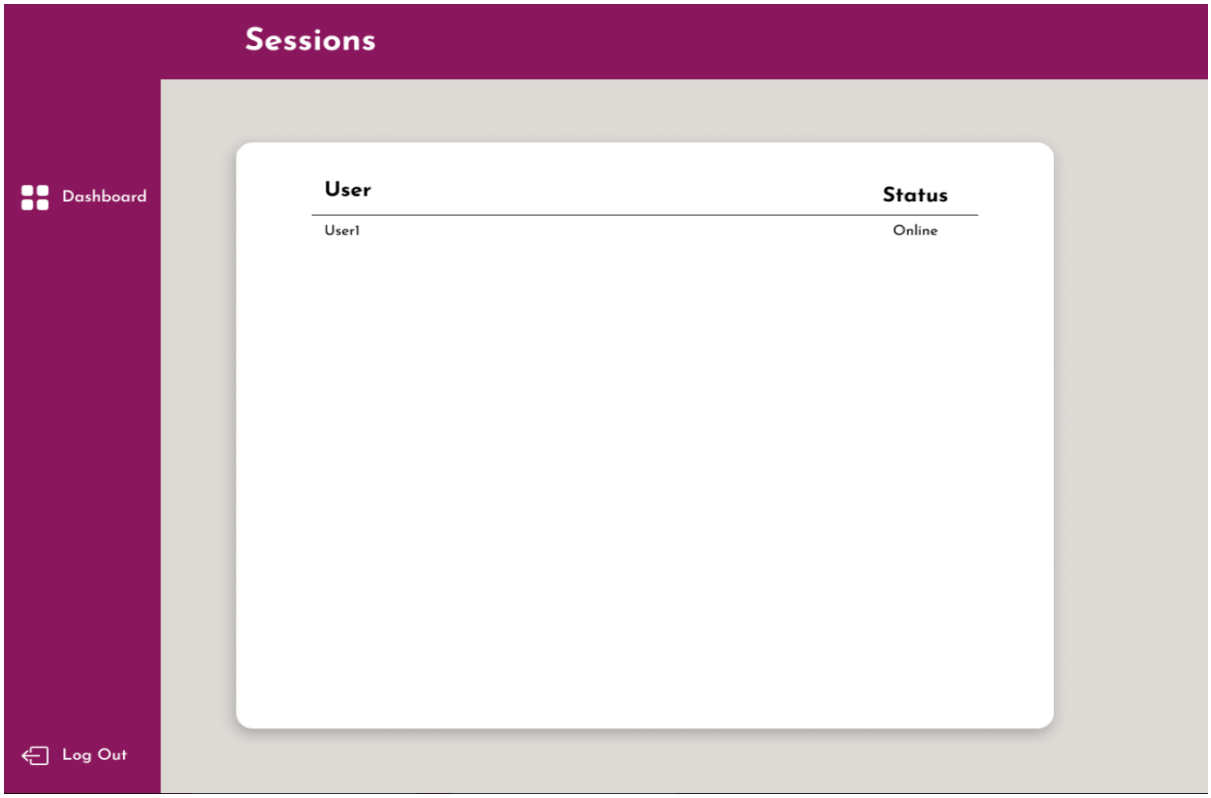


Abbildung 8.3: Wireframe - Session Page

8.2.4 Incident Overview

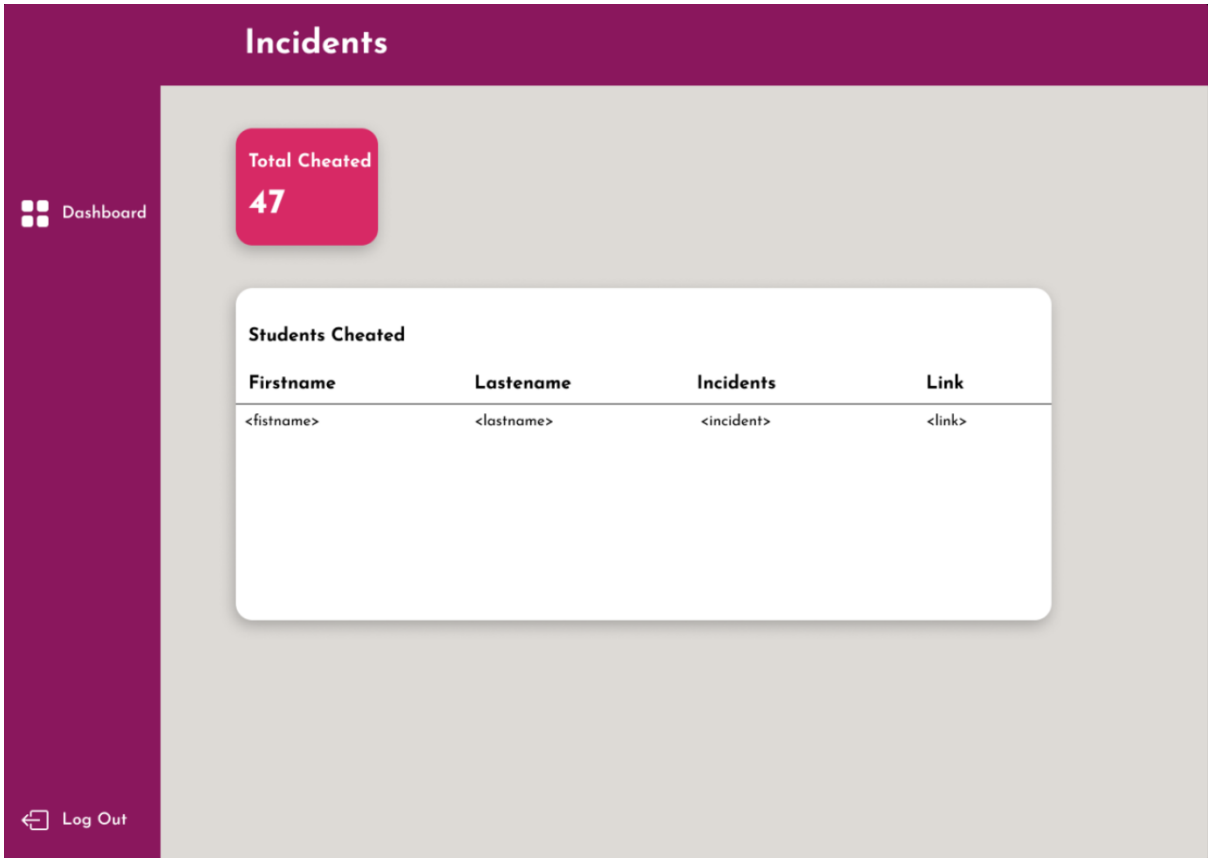


Abbildung 8.4: Wireframe - Incident Overview Page

8.2.5 Incident Student Overview Page

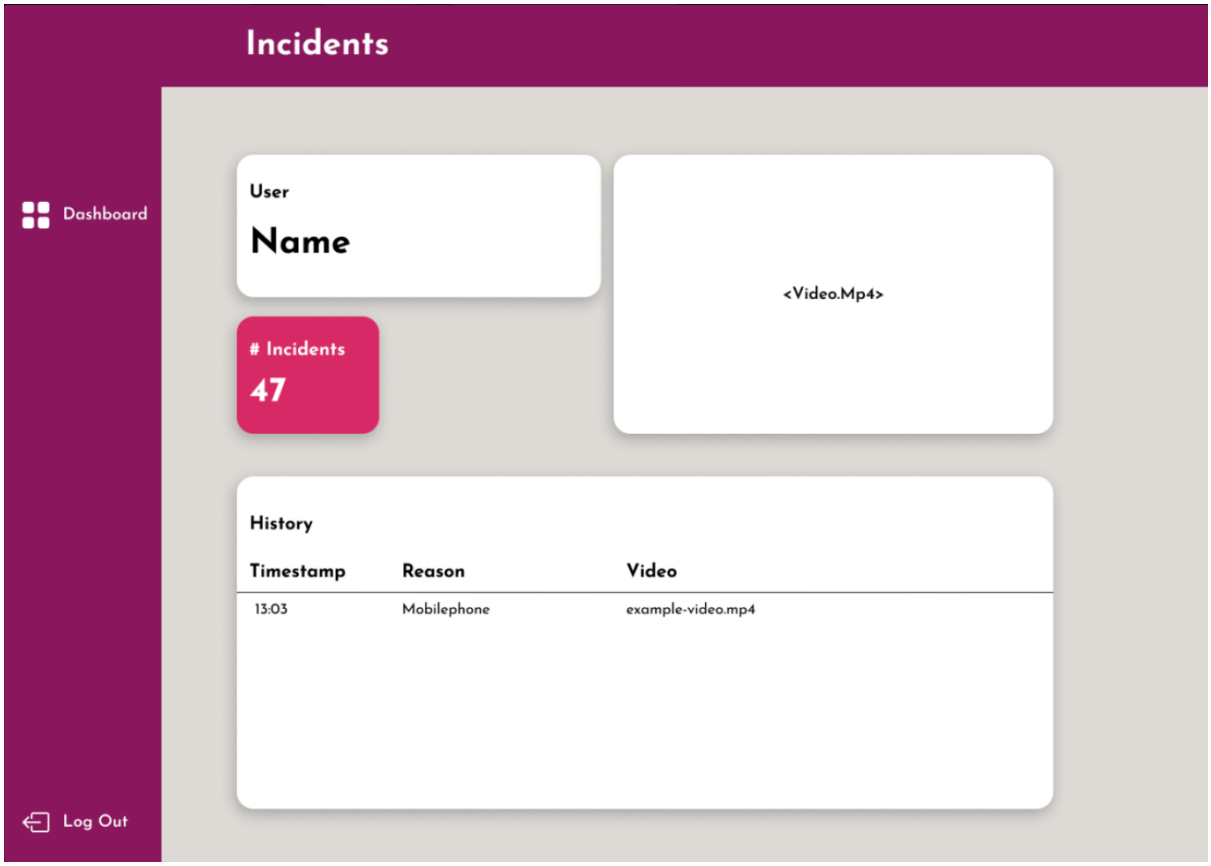


Abbildung 8.5: Wireframe - Incident Student Overview Page

8.3 Testauswertung

8.3.1 Test Abdeckung Backend

Coverage report: 97%

coverage.py v7.3.2, created at 2023-12-06 22:14 +0000

Module ↑	statements	missing	excluded	coverage
__init__.py	1	0	0	100%
app.py	29	1	0	97%
controllers/__init__.py	0	0	0	100%
controllers/api_controller.py	76	4	0	95%
controllers/authentication_controller.py	21	1	0	95%
controllers/exam_session_controller.py	39	4	0	90%
controllers/incident_controller.py	16	0	0	100%
controllers/student_exam_session_controller.py	13	0	0	100%
extensions.py	6	0	0	100%
models/__init__.py	0	0	0	100%
models/cheat_log.py	11	1	0	91%
models/exam_session.py	14	1	0	93%
models/recording.py	15	1	0	93%
models/student_exam_session.py	8	1	0	88%
models/student.py	12	1	0	92%
models/teacher.py	15	2	0	87%
routes/api_bp.py	21	1	0	95%
routes/authentication_bp.py	6	0	0	100%
routes/exam_session_bp.py	24	1	0	96%
routes/incident_bp.py	12	0	0	100%
routes/student_exam_session_bp.py	8	0	0	100%
services/__init__.py	0	0	0	100%
services/cheat_log_service.py	17	0	0	100%
services/exam_session_service.py	19	1	0	95%
services/recording_service.py	14	0	0	100%
services/student_exam_session_service.py	14	0	0	100%
services/student_service.py	11	0	0	100%
services/teacher_service.py	5	0	0	100%
tests/__init__.py	0	0	0	100%
tests/conftest.py	40	0	0	100%
tests/test_api.py	30	0	0	100%
tests/test_authentication.py	13	0	0	100%
tests/test_exam_session.py	27	0	0	100%
tests/test_incident.py	18	0	0	100%
tests/test_student_exam_session.py	19	0	0	100%
Total	**574**	**20**	**0**	**97%**

Abbildung 8.6: Coverage Report für Flask Backend

8.3.2 End Benutzer Tests Auswertung

Joel

Kopfneigungserkennung

Tabelle 8.1: Konfusionsmatrix Kopfneigungserkennung - Joel

	True	**False**
Positive	22	4
Negative	52	9

Genauigkeit: 74 / 87 = 85%
Präzision: 22 / 26 = 84%
Sensitivität: 22 / 31 = 71%
Spezifität: 52 / 56 = 92%

Gesichtserkennung

Tabelle 8.2: Konfusionsmatrix Gesichtserkennung - Joel

	True	**False**
Positive	1	48
Negative	38	0

Genauigkeit: 39 / 87 = 44%
Präzision: 1 / 49 = 2%
Sensitivität: 1 / 1 = 100%
Spezifität: 38 / 86 = 44%

Munderkennung

Tabelle 8.3: Konfusionsmatrix Munderkennung - Joel

	True	**False**
Positive	35	30
Negative	20	2

Genauigkeit: 55 / 87 = 63%
Präzision: 35 / 65 = 53%
Sensitivität: 35 / 37 = 95%
Spezifität: 20 / 50 = 40%

Iriserkennung

Tabelle 8.4: Konfusionsmatrix Iriserkennung - Joel

	True	**False**
Positive	54	5
Negative	22	6

Genauigkeit: 76 / 87 = 87%
Präzision: 54 / 59 = 91%
Sensitivität: 54 / 60 = 90%
Spezifität: 22 / 27 = 81%

Audioerkennung

Tabelle 8.5: Konfusionsmatrix Audioerkennung - Joel

	True	**False**
Positive	1	0
Negative	93	0

Genauigkeit: 94 / 94 = 100%
Präzision: 1 / 1 = 100%
Sensitivität: 1 / 1 = 100%
Spezifität: 93 / 93 = 100%

Jason

Kopfneigungserkennung

Tabelle 8.6: Konfusionsmatrix Kopfneigungserkennung - Jason

	True	**False**
Positive	2	6
Negative	51	0

Genauigkeit: 53 / 59 = 89%
Präzision: 2 / 8 = 25%
Sensitivität: 2 / 2 = 100%
Spezifität: 51 / 57 = 89%

Gesichtserkennung

Tabelle 8.7: Konfusionsmatrix Gesichtserkennung - Jason

	True	**False**
Positive	0	4
Negative	55	0

Genauigkeit: 55 / 59 = 93%
Präzision: 0 / 4 = 0%
Sensitivität: 0 / 0 = Nicht möglich
Spezifität: 55 / 59 = 93%

Munderkennung

Tabelle 8.8: Konfusionsmatrix Munderkennung - Jason

	True	**False**
Positive	2	7
Negative	50	0

Genauigkeit: 52 / 59 = 89%
Präzision: 2 / 9 = 22%
Sensitivität: 2 / 2 = 100%
Spezifität: 50 / 57 = 88%

Iriserkennung

Tabelle 8.9: Konfusionsmatrix Iriserkennung - Jason

	True	**False**
Positive	19	2
Negative	23	15

Genauigkeit: 42 / 59 = 71%
Präzision: 19 / 21 = 90%
Sensitivität: 19 / 34 = 56%
Spezifität: 23 / 25 = 92%

Audioerkennung

Tabelle 8.10: Konfusionsmatrix Audioerkennung - Jason

	True	**False**
Positive	0	0
Negative	68	0

Genauigkeit: 68 / 68 = 100%
Präzision: 0 / 0 = nicht möglich
Sensitivität: 0 / 0 = nicht möglich
Spezifität: 68 / 68 = 100%

Fabian

Kopfneigungserkennung

Tabelle 8.11: Konfusionsmatrix Kopfneigungserkennung - Fabian

	True	**False**
Positive	0	0
Negative	37	0

Genauigkeit: 37 / 37 = 100%
Präzision: 0 / 0 = Nicht möglich
Sensitivität: 0 / 0 = Nicht möglich
Spezifität: 37 / 37 = 100%

Gesichtserkennung

Tabelle 8.12: Konfusionsmatrix Gesichtserkennung - Fabian

	True	**False**
Positive	0	1
Negative	36	0

Genauigkeit: 36 / 37 = 97%
Präzision: 0 / 1 = 0%
Sensitivität: 0 / 0 = Nicht möglich
Spezifität: 36 / 37 = 97%

Munderkennung

Tabelle 8.13: Konfusionsmatrix Munderkennung - Fabian

	True	**False**
Positive	0	1
Negative	36	0

Genauigkeit: 36 / 37 = 97%
Präzision: 0 / 1 = 0%
Sensitivität: 0 / 0 = Nicht möglich
Spezifität: 36 / 37 = 97%

Zufall dass Resultate gleich wie bei der Gesichtserkennung sind

Iriserkennung

Tabelle 8.14: Konfusionsmatrix Iriserkennung - Fabian

	True	**False**
Positive	13	1
Negative	19	4

Genauigkeit: 32 / 37 = 86%
Präzision: 13 / 14 = 92%
Sensitivität: 13 / 17 = 76%
Spezifität: 19 / 20 = 95%

Audioerkennung

Tabelle 8.15: Konfusionsmatrix Audioerkennung - Fabian

	True	**False**
Positive	0	0
Negative	46	0

Genauigkeit: 46 / 46 = 100%
Präzision: 0 / 0 = nicht möglich
Sensitivität: 0 / 0 = nicht möglich
Spezifität: 46 / 46 = 100%

Sabrina

Kopfneigungserkennung

Tabelle 8.16: Konfusionsmatrix Kopfneigungserkennung - Sabrina

	True	**False**
Positive	11	1
Negative	88	12

Genauigkeit: 99 / 120 = 83%
Präzision: 11 / 12 = 92%
Sensitivität: 11 / 23 = 48%
Spezifität: 88 / 89 = 99%

Gesichtserkennung

Tabelle 8.17: Konfusionsmatrix Gesichtserkennung - Sabrina

	True	**False**
Positive	0	112
Negative	0	0

Genauigkeit: 0 / 112 = 0%
Präzision: 0 / 112 = 0%
Sensitivität: 0 / 0 = Nicht möglich
Spezifität: 0 / 112 = 0%

Munderkennung

Tabelle 8.18: Konfusionsmatrix Munderkennung - Sabrina

	True	**False**
Positive	47	6
Negative	36	23

Genauigkeit: 83 / 112 = 74%
Präzision: 47 / 53 = 88%
Sensitivität: 47 / 70 = 67%
Spezifität: 36 / 42 = 85%

Iriserkennung

Tabelle 8.19: Konfusionsmatrix Iriserkennung - Sabrina

	True	**False**
Positive	42	23
Negative	37	10

Genauigkeit: 79 / 112 = 70%
Präzision: 42 / 65 = 64%
Sensitivität: 42 / 52 = 81%
Spezifität: 37 / 60 = 62%

Audioerkennung

Tabelle 8.20: Konfusionsmatrix Audioerkennung - Sabrina

	True	**False**
Positive	29	0
Negative	110	7

Genauigkeit: 139 / 146 = 95%
Präzision: 29 / 29 = 100%
Sensitivität: 29 / 36 = 80%
Spezifität: 110 / 110 = 100%

8.4 Projektplanung

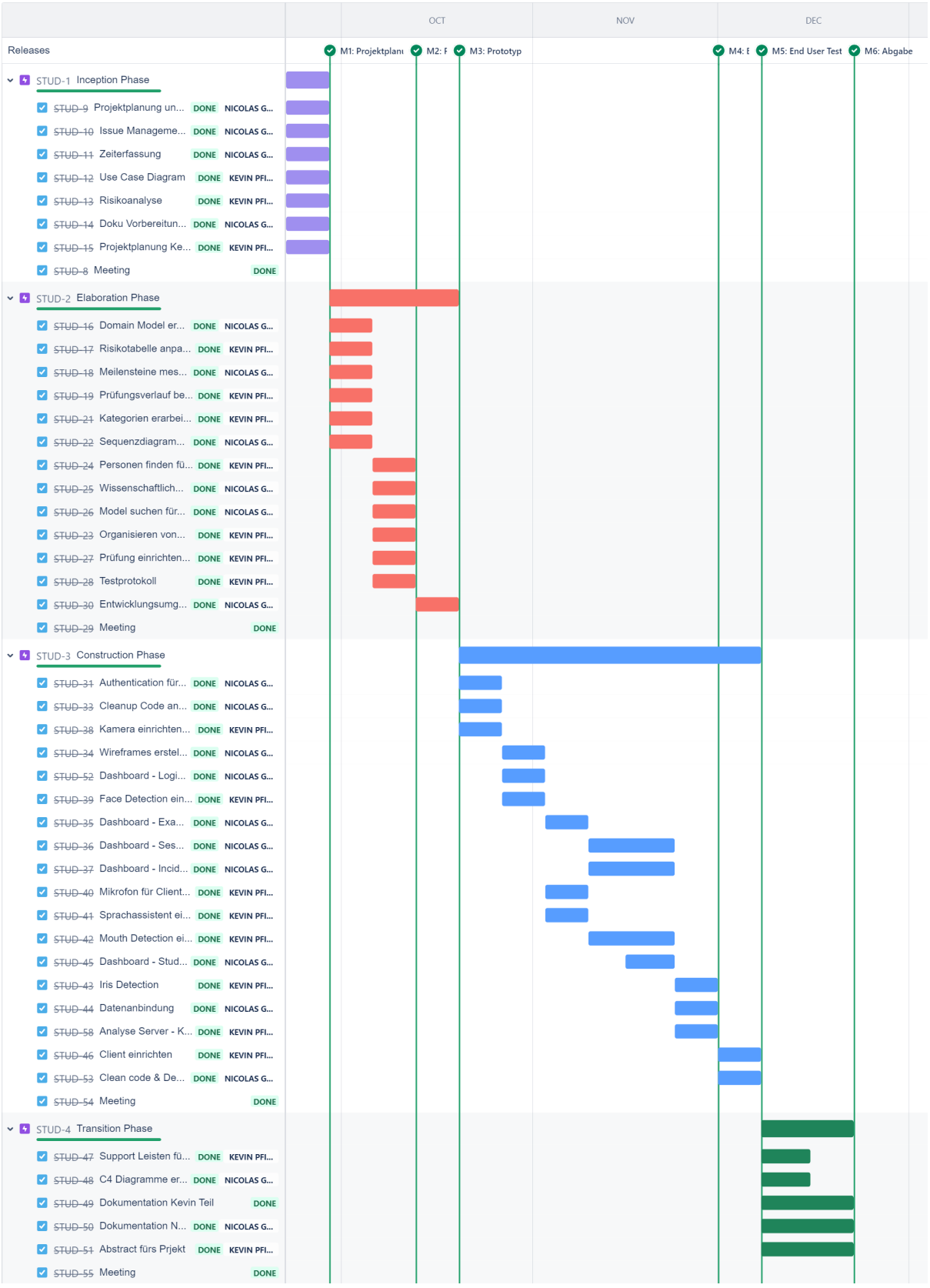


Abbildung 8.7: Projektplanung

8.5 Benutzeranleitung Prüfungsdurchführung

Anleitung zum Testing der Proctoringsoftware

Bitte lies dir die Anleitung zuerst durch, bevor du mit der Prüfung beginnst. Vor allem der Abschnitt "Spicken" ist wichtig für uns, um saubere Ergebnisse zu erhalten. Stelle sicher, dass du eine Webcam und ein Mikrofon bereit hast. Falls du den Test an einem Laptop machst, sollte dies kein Problem für dich darstellen.

Installation des Clients

1. Gehe auf folgenden Link und klone das Repository im «main» Branch.
   https://gitlab.ost.ch/nicolas.gattlen/proctoringsoftware
2. Öffne das Projekt und navigiere dich mittels des Terminals in den Client Ordner. (cd Client)
3. Erstelle ein Virtual Environment mittels folgendem Command.

```
python -m venv venv
```

4. Aktiviere deine venv mit den folgenden Commands:
   Für Windows:

```
.\venv\Scripts\Activate
```

   Für MacOs oder Linux:

```
source venv/bin/activate
```

5. Nachdem du dein virtuales Environment aktiviert hast, kannst du die benötigten Libraries mittels folgendem Command installieren

```
pip install -r requirements.txt
```

6. Starte die App.py Applikation. Wenn ein Login Fenster auftaucht, sollte das Programm funktionieren.

Login

1. Gehe auf die Webseite https://friederloch.de/moodle/login/index.php
2. Logge dich mit folgendem Login ein:
   Username: vornamenachname (Ü,Ö,Ä werden in ue, oe, ae

umgeschrieben)
PW: Ost_2023
3. Gehe in den Kurs «Testing der eigenen Proctoring Software»
4. Wähle den Proctoring Test aus und du bist bereit für die Prüfung.

Starten der Software

Nachdem du die Software installiert hast, kannst du sie folgendermassen starten:

1. Verbinde dich mit dem OST VPN, falls du den Test in der Schule machst und dich im OST-Netzwerk befindest ist dieser Schritt nicht nötig.
2. Logge dich mit folgendem Login in der Client Software ein.
E-Mail: vorname.nachname@ost.ch (Umlaute werden umgeschrieben)
PW: ost_2023
3. Wähle die Prüfung (Simulated Exam) aus.
4. Wähle dein Mikrofon aus. Falls du nicht sicher bist welches auszuwählen, wären wir froh, wenn du dies vorhin in einem externen Programm kurz testen könntest, da wir sonst keinen Ton in der Audiodatei bekommen.
5. Klicke auf Start Recording und starte den Test.
6. Nach dem Lösen des Tests kannst du auf «Stop recording» klicken und auf den Button «Send Video to Server». Wähle die MP4 und die WAV-Datei aus (das JSON muss nicht geschickt werden) und warte bis die Dateien geschickt wurden. (Die Dateien findest du im Client Ordner bei der installierten Software.) **WICHTIG:** Da die OST unserem Server wenig Bandbreite gegeben hat, kann der File Transfer sehr lange dauern (ca. 1 – 2 Stunden für ein 10 Min Video) Wir bitten dich deshalb das Programm einfach im Hintergrund laufen zu lassen und die Files einfach zu schicken. Da dies für deinen Rechner keinen grossen Rechenaufwand sein sollte, kannst du im Hintergrund gut andere Sachen machen. Falls du dir unsicher bist ob das File geschickt wurde, kannst du uns gerne kontaktieren.

Spicken

Der Sinn unserer Software ist es, Betrüger zu erkennen, um dies zu testen kannst du auf die speziellsten Arten versuchen zu betrügen. Beachte jedoch folgende Punkte:

Falls du einen neuen Tab öffnest, um dir Infos zu Googlen. Dann Sorge dazu, dass du den Tab auf einem externen Gerät öffnest oder den Tab auf einem

zweitbildschirm verschiebst. Du darfst auf deinem Hauptbildschirm nur den Test geöffnet haben.

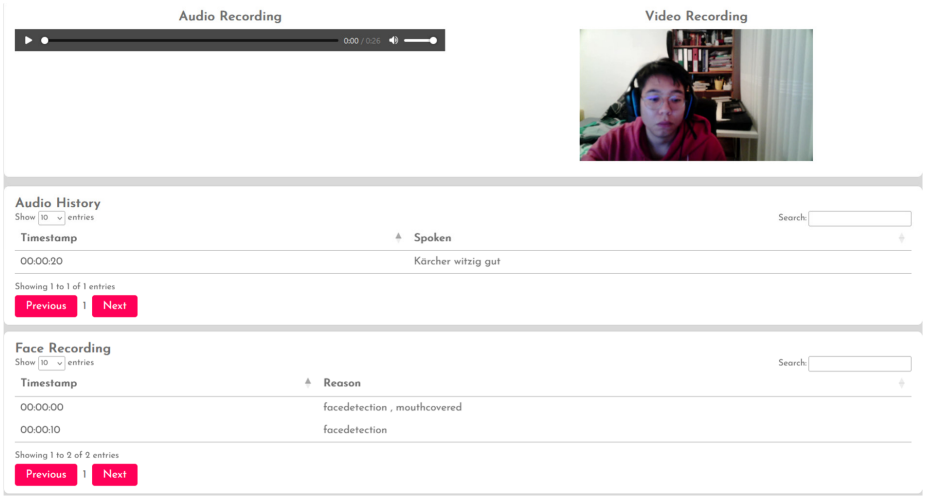
Ansonsten bist du jeglicher Kreativität freigestellt. Externe Programme dürfen verwendet werden und es darf auf alle Arten betrogen werden.

Hier ein paar Tipps wie du versuchen kannst zu betrügen:
- Löse die Prüfung zu zweit
- Mute dein Mikrofon
- Benutze dein Handy
- Sei im Discord/Teams
- Benutze einen Zweitbildschirm

WICHTIG: Versuche deine Aufnahme im Rahmen von 10 -15 Minuten zu halten. Ansonsten kann der File Transfer sehr lange dauern.

Was passiert mit deinen Aufnahmen

Nach dem du uns deine Daten geschickt hast, werden diese an einen Analyse Server geschickt, welcher deine Daten auswertet und uns eine Ausgabe gibt, zu welchem Zeitpunkt gespickt wurde. Deine Videos werden bis zur Abgabe unserer Studienarbeit (25.Dezember 2023) behalten und anschliessend gelöscht. Solltest du Interesse an deinen Resultaten haben, kannst du uns gerne kontaktieren und wir können dir einen Einblick in die Ausgabe deiner Spickversuche geben.



Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[KYM+22] Vinothini Kasinathan, Choi Ee Yan, Aida Mustapha, Vazeerudeen Abdul Hameed, Tham Hoong Ching, and Vinesh Thiruchelvam. Proctorex: An automated online exam proctoring system. *Multidisciplinary Student eJournal (MSEA)*, 71(3s2):876–889, 2022. https://www.philstat.org/special_issue/index.php/MSEA/article/view/320/316.

[PNSB16] Swathi Prathish, Athi Narayanan S, and Kamal Bijlani. An intelligent system for online exam monitoring. In *2016 International Conference on Information Science (ICIS)*, pages 138–143, Kochi, India, 2016. IEEE. https://ieeexplore.ieee.org/abstract/document/7845315.