Altklausuren-Lernportal für den FB Medizin der Goethe Uni

Projektdokumentation – Gruppe 18

Amelie Oberkirch <amelie.oberkirch@stud.tu-darmstadt.de>
Justin Karkossa <justin.karkossa@stud.tu-darmstadt.de>
Justin Peschke <justin.peschke@stud.tu-darmstadt.de>
Linda Pöhlmann <linda.poehlmann@stud.tu-darmstadt.de>
Marcel Mittenbühler <marcel.mittenbuehler@stud.tu-darmstadt.de>

Teamleitung:
Cynthia Rapp <cynthia.rapp@stud.tu-darmstadt.de>

Auftrag:
Prof. Dr. Guido Rößling <guido@tk.informatik.tu-darmstadt.de>
Telekooperation
Johannes Schreiber <j.schreiber@stud.uni-frankfurt.de>
Fachschaft Medizin der Goethe Uni Frankfurt

31.03.2021

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelorpraktikum
Wintersemester 2020/21
Fachbereich Informatik

Inhaltsverzeichnis

1 Projektbeschreibung

Kreuzen ist ein bestehendes Online-Portal, das die Studierenden des Fachbereichs Medizin der Goethe-Universität in Frankfurt bei der Klausurenvorbereitung unterstützt. Medizinische Prüfungen bestehen ausschließlich aus Single- und Multiple-Choice-Fragen, sodass es auch in der Klausurvorbereitung sinnvoll erscheint, dass derartige Fragen beantwortet, d.h. „gekreuzt“ werden. Diese Funktionalität setzt Kreuzen um, indem durch Studierende Altklausuren eingegeben werden und daraufhin beantwortet werden können.

Entstanden ist Kreuzen als studentisches Hobby-Projekt, um den Studierenden eine große Unterstützung für die Prüfungsvorbereitung an die Hand zu geben - gleichzeitig sorgen schlechter Code und veraltete Technologien dafür, dass Kreuzen nur eingeschränkt nutzbar, wartungs- und monitoringfähig ist. Dennoch verfügt die Plattform über 4.500 registrierte Nutzende, die bereits über 21 Mio. Fragen beantwortet haben. Die Probleme bei dem Nutzen der Plattform äußern sich dabei beispielsweise in:

- der Unübersichtlichkeit und daraus resultierenden, fehlenden Nutzerfreundlichkeit der Lernplattform (Abb. 1.1)
- den langen Wartezeiten bei dem Öffnen der Website
- den SQL Errors, welche regelmäßig den Nutzenden angezeigt werden (Abb. 1.2)

Aus diesen Gründen soll die Plattform eine komplette Neuentwicklung mit Erhalt der bisher entstandenen Inhalte erfahren. Die Motivation für das Bachelorpraktikum besteht also darin, die bereits vorhandene Idee aufzugreifen und in der Programmierung professionell zu modernisieren. Das führt zu einer grundlegenden Neuentwicklung unter Zuhilfenahme plattformunabhängiger Sprachen und Bibliotheken. Konkret werden Spring für die Backend- und React mit TypeScript für die Frontend-Entwicklung eingesetzt. Das Backend ist als Monolith aufgebaut. Die Kommunikation zwischen Front- und Backend geschieht über eine RESTful API, die mit Swagger dokumentiert ist. Postgres wird als SQL Datenbank genutzt, wobei das Datenschema neu entworfen wird, um gute Performance zu erzielen.

Kronkret wird für das Backend Spring, ein Java Framework, verwendet, da dieses ein etabliertes Framework für Webanwendungen ist. Java wurde aufgrund der plattformunabhängigkeit gewählt. Das Backend wird als Monolith aufgebaut, um die Entwicklung zu beschleunigen. PostgreSQL wird als SQL Datenbank genutzt, da es sich um eine performante, open-source Implementierung handelt, die ein sehr großes Spektrum an Funktionalitäten bereitstellt. Das Datenschema wird neu entwofen, um eine bessere Performance, Stabilität und Konsistenz zu erzielen. Hierbei wird darauf geachtet, mindestens die 3. Normalform zu erreichen.
Das Frontend wird in React mit TypeScript implementiert, da sich mit React moderne Single Page Applications (SPA) programmieren lassen und die in JavaScript fehlende Typsicherheit über TypeScript sichergestellt wird. Die Kommunikation zwischen Front- und Backend geschieht über eine RESTful API, die mit Swagger dokumentiert ist.
Die Architektur ist schematisch in Abb. 1.3 zu sehen.

Zunächst muss also für die Medizin-Studierenden der Goethe-Universität in Frankfurt eine neu aufgebaute

Plattform entstehen. Im Mittelpunkt steht hierbei der Wunsch, eine stabile Webseite zur Unterstützung der Prüfungsvorbereitung zu schaffen. Daneben soll eine modernisierte optische Gestaltung mit hellen Farben und intuitiven Menüführungen das Lernen unterstützen und eine angenehme User Experience bieten.
Zukünftig sollen auch Studierende anderer Fachbereiche und Universitäten von dem Projekt profitieren können. Ein wichtiges Ziel ist es daher, die Software portabel und erweiterbar zu gestalten, um auf Herausforderungen, wie vielfältige Fragenarten, vorbereitet zu sein.

Um diese Ziele effizient erreichen zu können, wird das Projektteam in kleinere Sub-Teams aufgeteilt, die sich einerseits intensiv mit der Neuentwicklung des Frontends und andererseits mit der Neuentwicklung des Backends beschäftigen. Dabei ist zu Beginn der Umsetzung eine ausführliche Einarbeitung in die bereits genannten Technologien notwendig.
Zu Beginn erfolgt eine Festlegung der Milestones in Authentifikation, User-Verwaltung, Semester- und Modulverwaltung, Klausuren, Fragen und abschließend Sessions. Diese Milestones werden stufenweise implementiert, da die Features aufeinander aufbauen. Zum Beispiel müssen Kurse implementiert sein, damit Klausuren für diese angelegt werden können.

Nach Abschluss des Projekts wird den Studierenden ein Portal zur Verfügung stehen, in denen die Funktionalitäten des alten Kreuzen-Portals erhalten bleiben und verbessert wurden. Das bedeutet: Altklausuren und auch individuelle Fragen von Studierenden können angelegt werden. Aus dem bestehenden Fragenpool können sich die Lernenden dann Übungsklausuren zusammenstellen und beim „Kreuzen“ ihren Wissensstand überprüfen, sowie Fragen in individuelle Kategorien einordnen.
Die bereits gesammelten Daten von Nutzenden und Altklausuren sollen in die neue Datenbank migriert werden. Um die Daten aus dem alten Datenschema in das neue zu überführen, wurde ein Skript geschrieben, welches sich mit der alten und neuen Datenbank verbindet und die Daten überführt. Dabei werden problematische Daten, z.B. wenn Konsistenzbedingungen verletzt werden, in ein File geschrieben, damit diese manuell übertragen werden können.

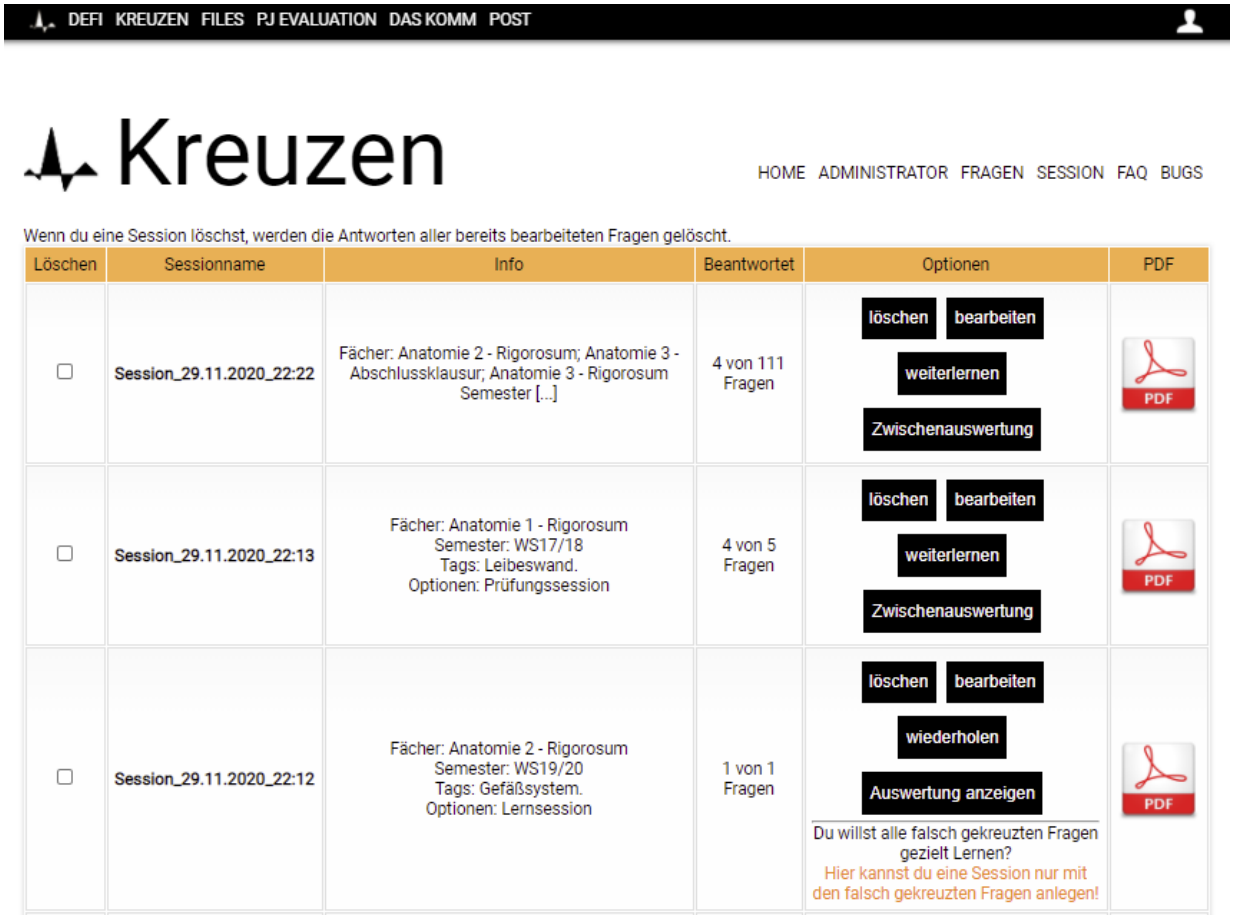

Abbildung 1.1: Unübersichtliche Menüs in Kreuzen

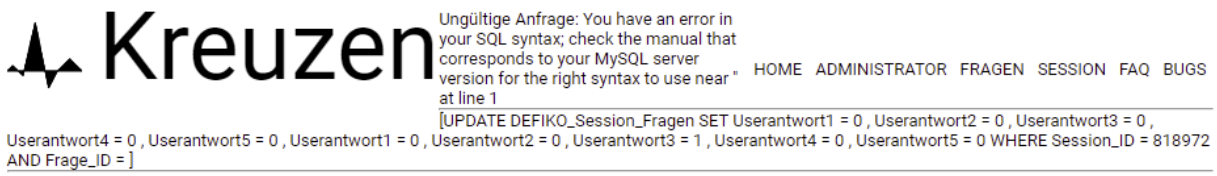

Abbildung 1.2: SQL-Error in Kreuzen

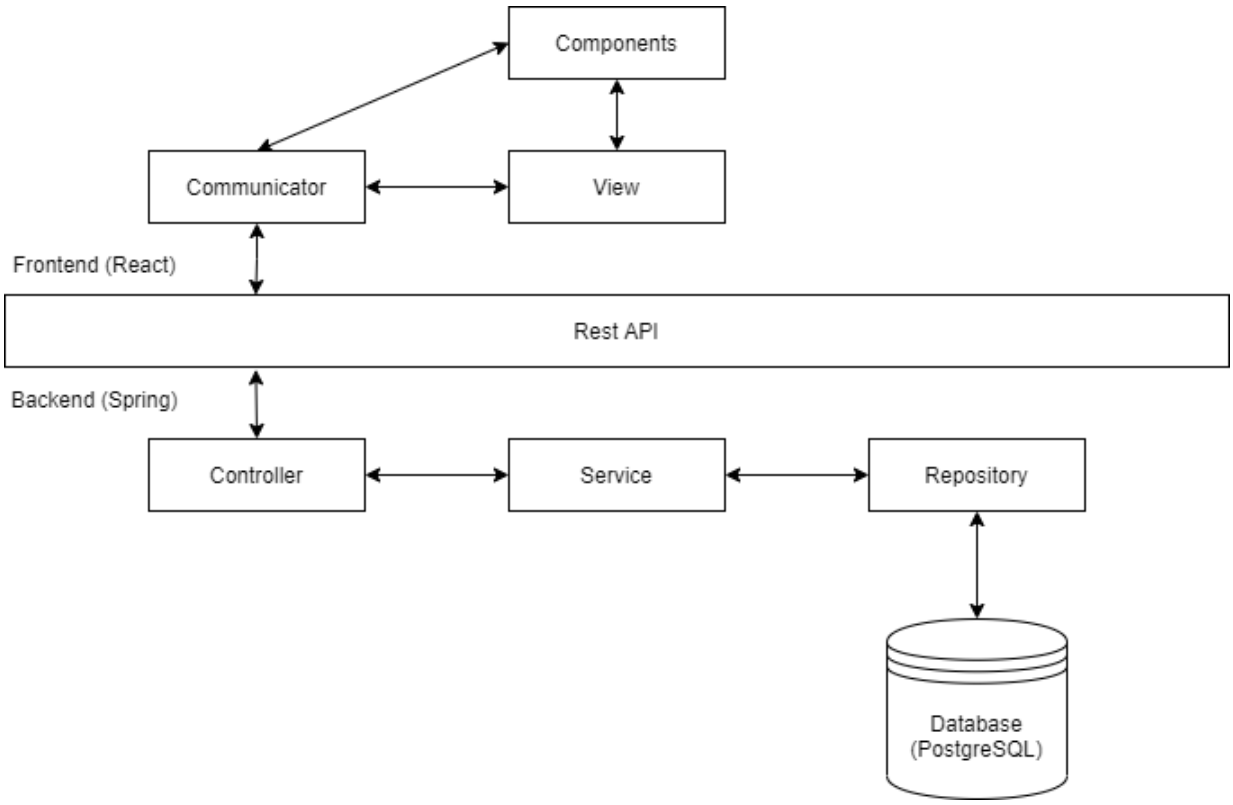

Abbildung 1.3: Architektur

2 Qualitätssicherung

Damit Kreuzen von den zukünftigen Entwickler*innen nahtlos weiterentwickelt werden kann und kein erneuter Neustart notwendig wird, setzen wir kontinuierliche Qualitätssicherung ein. Hierbei liegt das Hauptaugenmerk auf der Wartbarkeit und Portabilität der Applikation sowie der Datensicherheit aller gespeicherten Daten. Im Folgenden wird beschrieben, warum wir diese Ziele gewählt haben, welche Maßnahmen zur Sicherstellung der Qualitätssicherungsziele getroffen werden und wie der Qualitätssicherungsprozess abläuft.

Um parallel an mehreren Features zu arbeiten, wird Branching eingesetzt, wobei für jede User Story ein neuer Feature-Branch vom Develop-Branch erstellt wird, dann das Feature implementiert und nach Sicherstellung der QS Ziele wieder auf den Develop gemerged wird. Über Pull-Requests wird signalisiert, dass ein Feature fertig implementiert ist und die Qualitätssicherung durchgeführt werden kann. Zum Abschluss jeder Iteration wird dann der Develop auf den Master gemerged. Schematisch ist der Ablauf in Abb. 2.1 dargestellt.

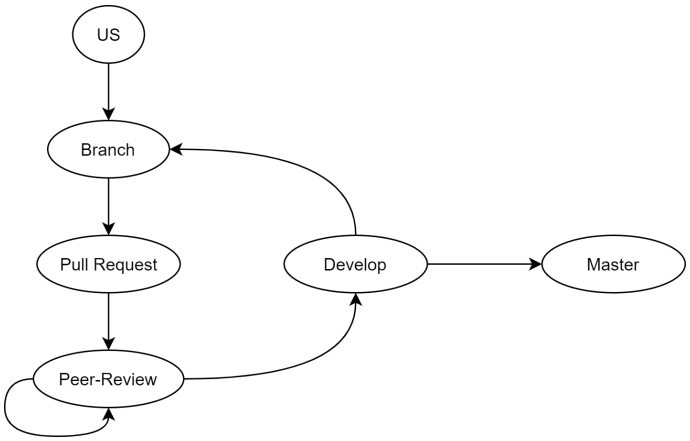

Abbildung 2.1: Ablauf der Implementierung

2.1 Qualitätsziel 1: Wartbarkeit

Bezug zum Projekt: Kreuzen soll auch nach Vollendung des Bachelorprojekts weiterentwickelt werden und wird zudem als Open-Source Software bereitgestellt. Daher ist es wichtig, die Wartbarkeit des Codes zu gewährleisten, sodass neue Features problemlos implementiert werden können. Hierfür muss sichergestellt werden, dass sich Entwickelnde in den Code einarbeiten können, dieser also ausreichend dokumentiert und kommentiert ist. Außerdem sind Tests zu implementieren, die die Funktionalität des Codes sicherstellen, damit angesichts folgender Updates keine alten Features unnutzbar werden.

Maßnahme: Für die Dokumentation des Codes werden JavaDocs verwendet, die für jede Funktion und Klasse vorliegen müssen. Zur Sicherstellung der Funktionalität werden im Backend automatisierte Tests implementiert, wobei eine Line-Coverage von mindestens 80 Prozent in jedem Package erreicht werden muss. Außerdem wird nach erfolgter Implementierung einer User Story ein Code Review durchgeführt. Zum Abschluss eines Features werden im Frontend außerdem manuelle Tests durchgeführt, welche als Integration-Tests die Zusammenarbeit von Frontend, Backend und Datenbank überprüfen.

Prozessbeschreibung: Die oben genannten Maßnahmen werden grundsätzlich von einer an der Implementierung unbeteiligten Person durchgeführt, die im selben Aufgabengebiet arbeitet, d.h. Entwickelnde aus dem Frontend reviewen Frontend-Code, Entwickelnde aus dem Backend reviewen Backend-Code. Das Review wird dann anhand der folgenden Checkliste durchgeführt:

- ☐ Struktur
  - ☐ Entspricht der Code-Stil dem zuvor festgelegten?
  - ☐ Ist der Code korrekt formatiert?
  - ☐ Ist der Code gut strukturiert?
  - ☐ Existieren ungenutzte Funktionen oder unerreichbarer Code?
  - ☐ Gibt es Wiederholungen im Code, die durch Auslagerung entfernt werden können?
  - ☐ Gibt es ungenutzte Variablen?
  - ☐ Werden Konstanten in eine Konfigurationsdatei ausgelagert?
  - ☐ (Backend) Werden Fehlermeldungen aus der i18l-Datei geladen?
- ☐ Dokumentation
  - ☐ Existiert über jeder Funktion ein aussagekräftiger JavaDoc Kommentar?
  - ☐ Ist die Beschreibung der einzelnen Funktionen verständlich?
  - ☐ Liegen an komplizierten Codeabschnitten Kommentare vor, die das Verhalten erläutern?
  - ☐ Sind Variablen, Klassen und Funktionen so benannt, dass ihre Funktion ersichtlich wird?
  - ☐ (Backend) Liegt die Swagger Dokumentation für alle Endpunkte vor?
- ☐ Architektur
  - ☐ Wird die zuvor festgelegte Architektur eingehalten?
  - ☐ Hat jede Klasse nur eine Zuständigkeit?
  - ☐ Wird Dependency Injection genutzt?
- ☐ Formalien
  - ☐ Wird 80% Line-Coverage im Package erreicht?
  - ☐ Werden Corner-Cases getestet?

Nachdem der*die Entwickelnde ein Feature implementiert hat, öffnet diese*r einen Merge-Request. Das Review für den Merge-Request ist innerhalb von drei Tagen durchzuführen, oder spätestens zum Ende einer Iteration. Hierfür geht der*die Reviewende die Checkliste durch. Im Frontend ist zudem abschließend durch manuelle Tests zu belegen, dass die Funktionalität erfüllt wird. Falls Mängel vorliegen, gibt der*die Reviewende diese an den Entwickelnden weiter, wobei wenn möglich konkrete Positionen im Code anzugeben sind. Der*die

Entwickelnde hat die Nachbesserungen innerhalb von drei Tagen durchzuführen. Nach Behebung der Mängel wird das Review wiederholt durchgeführt, bis keine weiteren Mängel vorliegen. Abschließend wird das Feature in die aktuelle Code-Basis gemerged.

2.2 Qualitätsziel 2: Portabilität

Bezug zum Projekt: Ein grundlegender Bestandteil einer Lernplattform ist die große Anzahl an unterschiedlichen Nutzer*innen, welche auf dieser registriert sind. Da die Nutzer*innen der Applikation unterschiedliche Browser und Geräte verwenden, ist es sehr wichtig, die Applikation für alle Nutzenden korrekt anzuzeigen. Es gilt also darauf zu achten, Kreuzen für jede*n Nutzer*in zugänglich zu machen und somit nicht auf verschiedene Hard- oder Software-Konstellationen zu beschränken.
Darüber hinaus ist eine plattformunabhängige Website obligatorisch, da somit eine größere Nutzendenbasis erreicht werden kann. Aus diesem Grund liegt ein Schwerpunkt unserer Qualitätsziele darin, die Plattformunabhängigkeit von Kreuzen sicherzustellen.
Mit Hilfe der Portabilität wird den Arbeitgebenden zusätzlich die Möglichkeit geboten, flexibel bei einer sich ändernden Verfügbarkeit von Servern zu bleiben und sich somit nicht auf eine bestimmte Plattform festzulegen. Dies ist vor Allem im Hinblick darauf bedeutend, dass die Arbeitgebenden langfristig noch nicht beschlossen haben, wie das Projekt in Zukunft deployed werden wird.

Maßnahme: Allgemein gilt, dass sowohl für das Front- als auch das Backend während der Implementierung darauf zu achten ist, keine Libraries zu nutzen, die plattformabhängig sind. Die Implementierung im Backend erfolgt in Java 11, im Frontend wird mit React Bootstrap gearbeitet. Sowohl Java 11 als auch React Bootstrap sind auf allen gängigen Plattformen verfügbar und tragen so maßgeblich zu einer hohen Portabilität bei.

Für Zugriffe auf die Datenbank werden Repository Interfaces implementiert, die auf unterschiedliche SQL-Datenbanken angepasst werden können. Durch Spring werden bereits die meisten Datenbanken unterstützt. Das Schema für die Datenbank ist für PostgreSQL erstellt, kann jedoch leicht auf andere SQL-Datenbanken angepasst werden.

Im Frontend wird bei der Implementierung auf ein responsives Design geachtet, um Elemente an unterschiedliche Bildschirmgrößen sowie mobile Endgeräte anzupassen. Die Umsetzung davon erfolgt mittels React-Bootstrap. Des Weiteren findet eine kontinuierliche manuelle Überprüfung von Browser-Features auf der [MDN](https://developer.mozilla.org/en-US/) statt.
Zu den Browsern zählen dabei:

- Chrome
- Firefox
- Safari

In Absprache mit dem Arbeitgeber wurde sich geeinigt, dass Internet Explorer/Edge nicht unterstützt werden muss.

Um sicherzustellen, dass die Applikation auf möglichst vielen unterschiedlichen Geräten mit verschiedener Auflösung korrekt angezeigt wird, wurden folgende Geräte zum kontinuierlichen manuellen Testen vereinbart:

- Huawei P20 (1080 x 2244 px)

- iPad 5. Generation (1536 x 2048 px)
- Samsung Galaxy S10 (1440 x 3040 px)
- Desktop-Monitor (1024 x 768 px)
- Desktop-Monitor HD (1920 x 1080 px)
- Desktop-Monitor 4K (3840 x 2160 px)

Prozessbeschreibung: Bezüglich der Portabilität werden die oben genannten Maßnahmen ebenfalls jeweils von einer*m unabhängigen Entwickler*in, der*die im selben Aufgabengebiet arbeitet, überprüft.

Dabei wurde anhand folgender Checkliste sichergestellt, dass die Portabilität im Laufe der Entwicklung kontinuierlich vorhanden ist:

- ☐ Backend
  - ☐ Wurden neue Libraries verwendet?
  - ☐ Sind die verwendeten Libraries plattformunabhängig?
  - ☐ Wurden alle Zugriffe auf die Datenbank mittels Resporitory Interfaces implementiert?
  - ☐ Wurde die Verwendung des Java Native Interfaces vermieden?
- ☐ Frontend
  - ☐ Wurden neue Libraries verwendet?
  - ☐ Sind die verwendeten Libraries plattformunabhängig?
  - ☐ Wurde während dem graphischen Aufbau der Nutzendenoberfläche auf responsive Design (mittels React-Bootstrap) geachtet?
  - ☐ Werden alle verwendeten Browser-Features unterstützt?
- ☐ korrektes Anzeigen der Applikation auf:
  - ☐ Huawei P20 (1080 x 2244 px)
  - ☐ iPad 5. Generation (1536 x 2048 px)
  - ☐ Samsung Galaxy S10 (1440 x 3040 px)
  - ☐ Desktop-Monitor (1024 x 768 px)
  - ☐ Desktop-Monitor HD (1920 x 1080 px)
  - ☐ Desktop-Monitor 4K (3840 x 2160 px)
- ☐ Korrektes Anzeigen der Applikation mit:
  - ☐ Chrome
  - ☐ Firefox
  - ☐ Safari

Nach der Implementierung jedes Features wird durch alle am Projekt beteiligten Entwickelnden die Lernplattform auf jedem der oben genannten Geräte und Browser manuell überprüft. Dabei liegt der Schwerpunkt darin, sicherzustellen, dass die Benutzendenoberfläche an die jeweilige Bildschirmgröße angepasst wird, und somit eine gute User Experience erhalten bleibt.
Nachdem der*die Entwickelnde ein Feature implementiert hat, öffnet diese*r einen Merge-Request. Das Review für den Merge-Request ist innerhalb von drei Tagen durchzuführen, oder spätestens zum Ende einer Iteration. Hierfür geht der*die Reviewende die Checkliste durch. Falls Mängel vorliegen, gibt der*die Reviewende diese an den Entwickelnden weiter. Der*die Entwickelnde hat die Nachbesserungen innerhalb von drei Tagen durchzuführen. Nach Behebung der Mängel wird das Review wiederholt durchgeführt, bis keine weiteren Mängel vorliegen. Abschließend wird das Feature in die aktuelle Code-Basis gemerged.

2.3 Qualitätsziel 3: Datensicherheit

Bezug zum Projekt: Das Qualitätsziel „Datensicherheit“ hat für das das Projekt eine hohe Relevanz, da in der Datenbank persönliche Nutzenden- und Kontaktdaten sowie universitätsinterne Informationen gespeichert werden. Klausuren und die von Professor*innen gestellten Fragen unterliegen dem Recht auf Urheberschaft und sind daher besonders schützenswert. Außerdem werden von der Seite weitere Daten (wie Nutzungszeit, Testergebnisse o.Ä.) erfasst, die ebenfalls personenbezogene Daten darstellen und somit nicht für unberechtigte Nutzende oder Externe zugänglich sein sollen.
Weiterhin gibt es Personengruppen mit unterschiedlichen Berechtigungen, weshalb einigen Nutzenden der Zugang zu entsprechenden Features der Website verwährt bleiben soll, während Andere genau diese Zugänge benötigen.

Maßnahme: Nutzende können verschiedene Rollen haben [User*in, Moderator*in, Admin, Superadmin]. Die Maßnahmen werden zweigleisig im Front- und Backend durchgeführt, um erhöhte Sicherheit zu gewährleisten. Dies geschieht jeweils unter Berücksichtigung der festgelegten Rollen.

Backend
Bei Anfragen an die Datenbank wird das jeweilige Nutzenden-Token mitgegeben. Falls der*die Nutzende Informationen anfragt, die nicht seiner*ihrer Berechtigung unterliegen, wird anstatt eines Answer-Tokens der Fehlercode 403 (Forbidden) übergeben, der bedeutet, dass die Anfrage aufgrund unzureichender Rechte abgelehnt wurde.
Bei jeder Anfrage an das Backend wird eine Überprüfung der Nutzenden-Berechtigungen direkt mitimplementiert.

Frontend
Nutzende bekommen nur Elemente angezeigt, die den Berechtigungen des Nutzenden entsprechen und niemand erhält Weiterleitungen zu ihm*ihr unzugänglichen Seiten.
Dabei wird stets darauf geachtet, dass jede Nutzenden-Klasse nur ihren Berechtigungen entsprechende Verlinkungen bei der Nutzung sieht und auch nur auf die jeweiligen Seiten Zugriff erhält.
Eine Besonderheit war hier außerdem das Design der Navigation, die für Admin- und Superadmin-Accounts zusätzlich Verwaltungselemente/-verlinkungen enthält, während Standard-Nutzende und Moderator*innen sich nicht in den Verwaltungsbereich navigieren können.

Prozessbeschreibung: Alle Maßnahmen werden zum Ende jeder Iteration sorgfältig auf deren korrekte Funktion überprüft. Dies geschieht zum einen mit passenden Backend-Tests, bei denen ein niedrigeres

Nutzenden-Token als erforderlich bei Anfragen mitgegeben wird. Es wird überprüft, ob das Backend in diesem Fall (korrekterweise) den Zugriff verweigert.
Im Frontend wird einerseits sichergestellt, dass kein*e Nutzer*in sich mithilfe der implementierten Komponenten in Bereiche navigieren kann, für die er*sie keine Berechtigung hat. Dies geschieht mithilfe von verschiedenen Nutzenden-Accounts, die unterschiedliche Rollen zugewiesen bekommen. Außerdem wird das Verhalten der Webapp überprüft, wenn ein*e Nutzende*r manuell in die Linkzeile beispielsweise den Pfad zu einer Administrationsseite eingibt. In diesem Fall wird immer eine Seite übergeben, die die Information „Zugriff verweigert“ erhält, nicht jedoch aber dem*der Nutzer*in die für höher gestellte Nutzende implementierte Seite anzeigt.

Anmerkung: Nutzende, die nicht eingeloggt sind, haben keine Berechtigungen, Seiten außerhalb vom Login/Registrierungsbereich einzusehen.
Bezüglich der Datensicherheit werden die oben genannten Maßnahmen ebenfalls jeweils von einer*m unabhängigen Entwickler*in, der*die im selben Aufgabengebiet arbeitet, überprüft.

Dabei wurde anhand folgender Checkliste sichergestellt, dass die Datensicherheit im Laufe der Entwicklung kontinuierlich vorhanden ist:

- ☐ Backend
  - ☐ Anfragen werden korrekterweise verweigert, wenn die Berechtigung des Anfragestellenden nicht ausreicht. Testen für jede Rolle und Anfrage in Kombination
  - ☐ Es wird der entsprechende Fehler mithilfe des passenden Fehlercodes zurückgegeben (zus. ggf. Fehlermessage)
  - ☐ Wird die Berechtigung des Nutzenden kontrolliert, bevor Änderungen vorgenommen oder Daten zurückgegeben wurden?
- ☐ Frontend
  - ☐ Jede*r Nutzer*in kann sich nur zu den Pfaden navigieren, die für seine Rolle vorgesehen sind (Ausnahme: manuelle Eingabe von Links).
  - ☐ Seiten, die nicht jeder*m Nutzer*in zugänglich sein sollten, werden nur im Falle einer ausreichend hohen Rolle übergeben, ansonsten wird eine „No Access“-Seite angezeigt.
  - ☐ Die manuelle Eingabe von Pfaden funktioniert ebenfalls (für jede Seite) nur unter der Bedingung, dass der*die Nutzer*in die entsprechende Berechtigung hat.
  - ☐ Wird die Berechtigung des Nutzenden überprüft, bevor zugangsbeschränkte Seiten angezeigt oder Daten an das Backend weitergeleitet werden?
  - ☐ Sollten unabhängig von den oben genannten Verfahren Fehler bei Anfragen auftreten, werden diese in Absprache mit dem Backend verbessert und alle betreffenden oder verwandten Seiten sowie Anfragen erneut getestet.

Die Checklist wird während allen Merge Requests von einer*m unabhängigen Entwickler*in verwendet, um ein Code Review und manuelle Tests durchzuführen. Das Review für den Merge-Request ist innerhalb von drei Tagen durchzuführen, oder spätestens zum Ende einer Iteration. Im Falle von gefundenen Mängeln innerhalb der Implementierung wird dies durch den*die Reviewer*in an den*die zuständige*n Entwickler*in mittels Kommentaren unter den jeweiligen Code-Abschnitten weitergegeben. Der*die Entwickelnde hat die Nachbesserungen innerhalb von drei Tagen durchzuführen.

Des Weiteren werden nach der Implementierung jedes Features manuelle Tests von den jeweiligen Entwickelnden im Front- und Backend anhand der oben genannten Checkliste durchgeführt. Dabei wird mit unterschiedlichen Accounts, die verschiedene Rechte zu besitzen, getestet.

3 Abweichende QS Ziele

Bei der Wartbarkeit im Frontend wurde von den automatisierten Tests abgesehen, da sich diese als zu zeitintensiv herausstellten. In Rücksprache mit dem AG wurde die Maßnahme im Frontend auf die manuellen Tests reduziert, die zusammen mit dem Code Review von einer nicht an der Implementierung des Features beteiligten Person, durchgeführt werden. Diese Änderung geschah noch während der 1. Iteration.

4 Projektverlauf und Projektgefährdende Ereignisse

Projektverlauf: Das erste Planungstreffen mit den Auftraggebenden fand am 25.11.2020 statt. Für die ersten beiden Iterationen wurde im Einverständnis aller Beteiligten ein abweichender Zeitraum von je 3 Wochen gewählt. Grund dafür sind einerseits Einarbeitungszeiten in neue Programmiersprachen und Technologien, andererseits die Weihnachtspause, in der nur reduziert am Projekt gearbeitet wurde. Für die restlichen Iterationen wurde eine einheitliche Dauer von 2 Wochen festgelegt.

In der vierten Iteration wurden keine User Stories fetiggestellt, da alle Teammitglieder aufgrund von Prüfungsvorbereitung und anderen Abgabeterminen nur wenig Zeit für das Projekt investieren konnten.

Die zu Beginn festgelegten Ziele bezüglich der Funktionalitäten, welche die Lernplattform bei Abgabe an den AG erhalten sollte, wurde im Verlauf des Projekts in Absprache mit den AG auf die grundlegendsten Features reduziert. Ursache dafür war der hohe Zeitaufwand, welcher für die meisten Features zu Beginn des Projekts unterschätzt wurde.

Die vorgegebene Bearbeitungszeit von 230h für das Projekt wurde dabei von jedem Teammitglied stark überschritten. Grund dafür war zum einen die hohe Einarbeitungszeit, welche zu Beginn des Projekts nötig war, um sich mit allen Technologien bezüglich der Webentwicklung vertraut zu machen und zum anderen ein gemeinsames Bestreben, die Grundfunktionalität der Lernplattform abzudecken.

Die Ergebnisse des Projekts wurden am 17.03.2021 im finalen Treffen übergeben.

- Iteration 1: 09.12.2020 – 30.12.2020
- Iteration 2: 31.12.2020 – 20.01.2021
- Iteration 3: 21.01.2021 – 03.02.2021
- Iteration 4: 04.02.2021 – 17.02.2021
- Iteration 5: 18.02.2021 – 03.03.2021
- Iteration 6: 04.03.2021 – 17.03.2021

Projektgefährdende Ereignisse: Im Laufe des Projekts gab es keine projektgefährdenden Ereignisse.

5 Softwarelizenz

Der für dieses Projekt erstellte Code steht im Einvernehmen aller Beteiligten unter folgender Lizenz:

GNU Lesser General Public License (https://www.gnu.org/licenses/lgpl-3.0.en.html)