NEXTGEN
- DEVELOPMENT -

Qualitätssicherungsplan

NextGen Development

Version 1.0

4. April 2025

Teammitglieder:
Julian Lachenmaier
Ayo Adeniyi
Din Alomerovic
Cedric Balzer
Rebecca Niklaus

Verantwortlich für dieses Dokument:
Rebecca Niklaus (Qualitätssicherung)

Inhaltsverzeichnis

1 Einleitung

Die Qualitätssicherung spielt eine zentrale Rolle im gesamten Projektverlauf. Dadurch soll sichergestellt werden, dass alle Ergebnisse den definierten Anforderungen entsprechen. Dieser Qualitätssicherungsplan beschreibt die Maßnahmen, Prozesse und Standards, die angewendet werden, um eine hohe Qualität der Software sowie der zugehörigen Dokumentation zu gewährleisten.

Der Plan umfasst verschiedene Aspekte der Qualitätssicherung, darunter das Projektmanagement, die Einhaltung von Coding-Standards, Versionierung, Teststrategien sowie die Dokumentation. Durch die Implementierung bewährter Methoden und den Einsatz geeigneter Tools wird sichergestellt, dass das Endprodukt nicht nur funktional, sondern auch stabil, wartbar und benutzerfreundlich ist.

Darüber hinaus legt dieser Plan die Verantwortlichkeiten innerhalb des Teams fest und beschreibt, wie Qualitätssicherungsmaßnahmen kontinuierlich überwacht und verbessert werden. Ziel ist es, potenzielle Fehler frühzeitig zu erkennen, Risiken zu minimieren und eine transparente sowie effiziente Zusammenarbeit zu fördern.

2 Projektmanagement

Ziel des Projektmanagements ist, Risiken zu begrenzen und Projektziele unter Verwendung der verfügbaren Ressourcen zu erreichen. Das Endergebnis des Projekts soll den Anforderungen des Kunden entsprechen. Im folgenden Abschnitt werden Maßnahmen definiert, um dieses auf Seiten des Projektmanagements zu erreichen.

Um einen organisierten Projektablauf zu gewährleisten, werden häufig Methoden wie Scrum oder Kanban verwendet. Dieses Projekt orientiert sich an der Kanban-Vorgehensweise.

Kanban ist eine agile Methode. Das heißt, dass man die Arbeitsweise im Laufe des Projekts durch kleine Änderungen laufend verbessert [Dec24]. Zur Visualisierung der Aufgaben des Teams wird ein Kanban-Board verwendet. Diese werden in drei Zustände unterteilt: To-Do, in Bearbeitung und vollständig. Für die Projektplanung wird das Tool **ClickUp** verwendet.

2.1 Ressourcen-Planung

Für die Bearbeitung eines Projekts stehen begrenzte Ressourcen zur Verfügung. Im Rahmen dieses Projekts sind Personal- und zeitliche Ressourcen relevant.

Das Zeitmanagement stellt sicher, dass Abgabefristen eingehalten werden können [Hel23]. Jedem der fünf Projektmitglieder steht wöchentlich 10 Stunden zur Verfügung. Um diese möglichst sinnvoll zu nutzen, wird die Aufgabenverteilung auf die Fähigkeiten und Erfahrungen der Mitglieder angepasst. Die Zeiterfassung erfolgt in ClickUp.

2.2 Projektphasen und Meilensteine

Das Projekt wird in mehrere Phasen unterteilt, um eine strukturierte Vorgehensweise sicherzustellen. Jeder Phase sind klare Meilensteine zugewiesen, um den Fortschritt messbar zu machen. Diese sind im Projektplan festgelegt.

2.3 Kommunikationsmanagement

Grundvorraussetzung für einen erfolgreichen Projektablauf stellt die Zusammenarbeit und das Wohlbefinden der einzelnen Mitglieder dar. Sollten sich Mitglieder über- oder unterfordert fühlen, soll dies direkt kommuniziert werden können. Daher werden zwei Teams-Meetings á 15 Minuten pro Woche fest eingeplant, um mögliche Engpässe frühzeitig zu beheben. Außerdem besteht jederzeit die Möglichkeit, sich über Teams oder WhatsApp an sein Team zu wenden.

2.4 Risikomanagement

Bereits zu Beginn des Projekts werden mögliche Risiken, die im Laufe des Projekts zu Problemen führen können, analysiert. Im Bereich des Projektmanagements werden hierbei insbesondere Risiken genannt, die sich auf einen Ressourcen-Mangel beziehen. Daher wird eine Risikoanalyse durchgeführt, um präventive Maßnahmen vor Beginn des Projekts zu definieren und um auf mögliche Komplikationen vorbereitet zu sein.

3 Software-Qualität

Software-Qualität bezieht sich auf die Gesamtheit von Merkmalen und Eigenschaften, die sicherstellen, dass die Software die Anforderungen erfüllt sowie stabil und wartbar bleibt. Ein wichtiger Punkt ist die Verwaltung des Codes, die durch die Nutzung gewisser Verfahren und Tools erreicht wird. Die umfasst sowohl die Versionskontrolle als auch die Beachtung der Code-Qualität.

3.1 Versionskontrolle

n der Versionskontrolle wird GitHub als Plattform genutzt, um den effektiv zu verwalten und Änderungen nachvollziehen zu können. Das Hauptziel dabei ist, eine klare und strukturierte Entwicklung zu gewährleisten, um die Codequalität zu maximieren. Dabei sind folgende Punkte zu beachten:

- **Branching-Modell:** Ein effektives Branching-Modell ist entscheidend, um parallel an verschiedenen Funktionen und Fixes arbeiten zu können, ohne die Entwicklung anderer Bereiche zu stören. Ein gängiges Modell ist das Git-Flow-Modell oder eine angepasste Version davon. Der Hauptbranch enthält immer die Version des Codes, die produktionsbereit ist. Für jedes neue Feature wird ein eigener Branch erstellt. Nach der Fertigstellung und einer Überprüfung der Code-Qualität wird dieser in den Hauptbranch gemerged.
- **Pull-Requests und Code-Reviews:** Bevor Codeänderungen in den Hauptbranch gemerged werden, muss eine Pull-Request durchgeführt werden. So wird sichergestellt, dass der Code den Qualitätsstandards entspricht und keine bestehenden Funktionen beinträchtigt werden.
- **Merge-Requests:** Bevor eine Merge-Request akzeptiert werden kann, müssen alle Tests bestanden werden. Dadurch wird sichergestellt, dass nur getesteter Code in den Hauptbranch gelangt.

Durch die Anwendung dieser Prozesse wird die Codequalität durchgehend überwacht und verbessert, während parallel die Flexibilität und Effizienz bei der Entwicklung neuer Features erhalten bleibt.

3.2 Code-Qualität

Durch einheitliche Konfigurationen und Best Practices wird sichergestellt, dass der Code gut strukturiert, wartbar und fehlerfrei bleibt. Dafür werden folgende Ansätze und Werkzeuge verwendet:

- **Einheitliche Konfigurationen für IDEs**: Eine Nutzung einheitlicher Versionen ist essenziell, um Code-Konflikte zu vermeiden. Für ASP.NET wird hierbei die Version 8.0 verwendet, für React Version 19. So wird sichergestellt, dass keine Inkompatibilitäten auftreten.

- **Best Practices:** Außerdem sollen Best Practices genutzt werden, um eine gute Code-Qualität sicherzustellen. In React wird hierbei ESLint verwendet. ESLint ist ein Tool zur Analyse von JavaScript-Code, das Probleme findet, um Code konsistenter und fehlerfreier zu machen. Viele Probleme können automatisch behoben werden [ESL25]. C#?

4 Textkonzept

Mithilfe von Tests werden Bugs gefunden, die im Laufe der Entwicklung auftreten können. Durch sie wird ermittelt, ob ein Programm so funktioniert, wie es gedacht ist [ES24].

4.1 Testarten

Es kann zwischen folgenden Testarten unterschieden werden [ES24]:

- **Manuelles Testen:** Tests werden ohne Automatisierung von festgelegten Testern durchgeführt. Dies wird häufig für UI/UX-Tests, visuelle Überprüfungen und Interaktionstests im Frontend genutzt.
- **Unit-Test:** Testet einzelne Funktionen oder Module, z. B. die Prüfung einer E-Mail-Adresse auf Gültigkeit. Dies wird hauptsächlich im Backend angewendet.
- **Integrationstest:** Überprüft das Zusammenspiel verschiedener Module, um sicherzustellen, dass sie korrekt zusammenarbeiten. Da Frontend und Backend mithilfe von API-Calls interagieren, betrifft dieser Test beide Bereiche.
- **Systemtest:** Testet das gesamte System, einschließlich Benutzeroberfläche und Programmlogik. Dieser Test kann sowohl das Frontend als auch das Backend betreffen.
- **Akzeptanztest:** Der letzte Test vor der Auslieferung, um sicherzustellen, dass die Software alle Anforderungen erfüllt. Dieser kann sowohl das Frontend als auch das Backend abdecken.
- **Regressionstest:** Wiederholung früherer Tests, um sicherzustellen, dass Änderungen keine bestehenden Funktionen beeinträchtigen. Diese Art von Test betrifft sowohl Frontend als auch Backend.

4.2 Fehlerarten

Des Weiteren kann zwischen folgenden Fehlerarten unterschieden werden, für deren Überprüfung verschiedene Testarten benötigt werden [ES24]:

- **Funktionale Fehler:** Diese treten auf, wenn eine Funktion nicht wie vorgesehen arbeitet, beispielsweise ein nicht funktionierender Link auf einer Webseite. Solche Fehler lassen sich in der Regel gut durch automatisierte Tests, wie Unit- oder Integrationstests, aufdecken.

- **Inhaltliche Fehler:** Hierbei handelt es sich um Fehler im Text oder in der Darstellung von Inhalten, etwa Rechtschreibfehler oder falsche Bilder in einem Artikel. Während einige dieser Fehler durch automatisierte Tools (z. B. Rechtschreibprüfungen) erkannt werden können, erfordert die inhaltliche Prüfung in den meisten Fällen manuelle Überprüfung durch einen Tester oder Reviewer.

- **Visuelle Fehler:** Diese entstehen, wenn beispielsweise ein Fenster einer Webseite einen anderen Bereich überdeckt oder eine Darstellung fehlerhaft ist. Automatisierte visuelle Regressionstests können solche Probleme erkennen, dennoch ist oft eine manuelle Sichtprüfung notwendig, um subtile Designfehler auszumachen.

- **Benutzerfreundlichkeitsfehler:** Solche Fehler betreffen die Usability, etwa wenn eine Benutzerführung unnötig kompliziert oder nicht intuitiv gestaltet ist. Diese Art von Fehler lässt sich nur bedingt automatisiert testen, da sie stark von subjektiven Nutzereindrücken abhängt. Hier sind manuelle Usability-Tests und Feedback von echten Nutzern essenziell.

4.3 Umsetzung

Auch die Umsetzung kann unterschiedlich in Angriff genommen werden. Es besteht die Möglichkeit, Black Box Testing, White Box Testing oder Grey Box Testing durchzuführen. Black Box Testing wird aus Benutzersicht durchgeführt, da dort die Software unabhängig vom Quellcode durchgeführt wird. White Box Testing hingegen setzt sich mit dem Kern und Struktur der Anwendung auseinander [Ged23]. Grey Box Testing stellt eine Mischform aus beiden Testarten dar und ist ein guter Kompromiss aus Aufwand und Nutzen. Nur kritische Systeme werden als White Box genau betrachtet, nicht so wichtige als Black Box [ES24]. Daher erweist sich diese Art des Testens für dieses Projekt als am sinnvollsten.

Für automatisierte Tests werden Tools wie GitHub Actions oder Jenkins verwendet. Nach jedem Push werden verschiedene Prozesse durchlaufen, darunter auch automatisierte Tests. Somit können Regressionstests regelmäßig durchgeführt werden, um nach jeder Änderung überprüfen zu können, ob bereits realisierte Features weiterhin funktionstüchtig sind.

5 Dokumentationskonzept

Eine strukturierte und umfassende Dokumentation ist unverzichtbar, um die Verständlichkeit, Wartbarkeit und Erweiterbarkeit der Software garantieren zu können. Sie dient als Grundlage für Entwickler, Tester und weitere Stakeholder, um sich effizient in den Code einzuarbeiten. Außerdem sollten Designentscheidungen für den Kunden ersichtlich sein.

Im Projekt werden verschiedene Tools zur Dokumentation eingesetzt. Diese unterstützen sowohl die Code-Dokumentation als auch die Beschreibung der Systemarchitektur und der API-Schnittstellen. Die folgende Tabelle bietet einen Überblick über die eingesetzten Werkzeuge für die Dokumentation in ASP.NET und React.

Framework	**ASP.NET**	**React**
Code-Kommentare	XML-Dokumentation	JSDoc
Automatische Generierung	DocFX, Swagger	Storybook, Styleguidist (???)
Architektur-Dokumentation	UML, README, Confluence	README, Komponenten-Diagramme
API-Dokumentation	Swagger (OpenAPI)	OpenAPI für API-Calls

Tabelle 1: Werkzeuge für die Dokumentation in ASP.NET und React

6 Dokumente

Im Laufe des Projekts fallen mehrere Dokumente an, die sowohl zur internen Organisation als auch zur externen Kommunikation mit Betreuern und Kunden dienen. Diese Dokumente unterstützen das Projektmanagement, die Nachverfolgbarkeit von Fortschritten sowie die Qualitätssicherung. Eine klare Struktur und ein einheitlicher Umgang mit diesen Dokumenten sind daher essenziell, um Effizienz, Transparenz und Konsistenz zu gewährleisten.

Je nach Verwendungszweck lassen sich die Dokumente in regelmäßige Berichte sowie dynamische, sich weiterentwickelnde Dokumente unterteilen. Während wöchentliche Abgaben die Projektdokumentation sicherstellen, müssen zentrale Dokumente wie das Lasten- oder Pflichtenheft fortlaufend an veränderte Anforderungen angepasst werden. Die folgenden Abschnitte beschreiben, welche Dokumente erstellt werden müssen und wie der Umgang mit Änderungen organisiert ist.

6.1 Wöchentliche Abgabe

Wöchentlich abzugeben sind folgende Dokumente:

- **Wochenbericht:** Wöchentlich muss ein Wochenstatusbericht eingereicht werden, in dem die bearbeiteten Aufgaben reflektiert werden sollen. Des Weiteren wird sich auf Abweichungen vom Projektplan bezogen, um die Gestaltung der folgendenen Wochen dementsprechend anpassen zu können. Dafür wurde ein Template in Word erstellt, das in ClickUp hinterlegt wurde.

- **Aufwandsbericht:** Ebenfalls wöchentlich einzureichen ist der Aufwandsbericht. Dieser entnimmt die aufgewendete Zeit der Projektmitglieder aus ClickUp und summiert diese. Außerdem soll die insgesamt aufgewendete Zeit berechnet werden. Dies erfolgt in Excel und wird als PDF eingereicht.

6.2 Umgang mit sich weiterentwickelnden Dokumenten

Dokumente wie das Lasten- oder Pflichtenheft werden im Projektverlauf häufig an sich ändernde Anforderungen angepasst. Dies stellt sicher, dass Änderungen klar nachvollzogen werden können. Dabei sollten folgende Punkte beachtet werden:

- **Versionierung**: Jede Dokumentenänderung wird mit einer neuen Versionsnummer versehen. Dies stellt sicher, dass Änderungen klar nachvollzogen werden können, was sowohl aus Kunden- als auch Entwicklersicht relevant ist.

- **Historisierung**: Ältere Versionen bleiben zur Nachverfolgbarkeit erhalten und sollten weiterhin zur Verfügung stehen. Dies erfolgt über die frei zugängliche Website

des Teams.

- **Regelmäßige Reviews**: Um eine hohe Qualität und Konsistenz sicherzustellen, sollten Dokumente regelmäßig überprüft und aktualisiert werden. Hierbei wird überprüft, ob gewisse Standards oder Vorgaben eingehalten werden. Dies erfolgt durch definierte Reviewer, die nach dem Vier-Augen-Prinzip oder in Form von Peer Reviews arbeiten [Stu].

7 Fazit

Die Qualitätssicherung ist ein essenzieller Bestandteil des Projekts, um die Einhaltung aller Anforderungen zu sichern um das Endprodukt zuverlässig, wartbar und benutzerfreundlich zu gestalten. Durch strukturierte Prozesse wie Versionierung, Code-Reviews, automatisierte Tests und regelmäßige Dokumentationsupdates wird die Qualität kontinuierlich überwacht und verbessert.

Die Kombination aus bewährten Methoden, modernen Tools und einer klaren Kommunikation innerhalb des Teams trägt dazu bei, Fehler frühzeitig zu identifizieren und die Effizienz der Entwicklung zu steigern. Zudem ermöglicht die regelmäßige Überprüfung der Prozesse eine kontinuierliche Optimierung der Arbeitsweise.

Mit diesem Qualitätssicherungsplan wird eine solide Grundlage für das Team geschaffen, um die Projektziele erfolgreich zu erreichen und ein hochwertiges Produkt zu liefern, das den Anforderungen der Kunden gerecht wird.

Literatur

[Dec24] André Dechange. „Agiles Projektmanagement“. In: *Projektmanagement – Schnell erfasst*. Hrsg. von André Dechange. Berlin, Heidelberg: Springer, 2024, S. 265–311. ISBN: 978-3-662-68169-5. DOI: 10.1007/978-3-662-68169-5_4. URL: https://doi.org/10.1007/978-3-662-68169-5_4 (besucht am 28. 03. 2025).

[ES24] Timm Eichstädt und Stefan Spieker. „Testen“. In: *52 Stunden Informatik: Was jeder über Informatik wissen sollte*. Hrsg. von Timm Eichstädt und Stefan Spieker. Wiesbaden: Springer Fachmedien, 2024, S. 281–288. ISBN: 978-3-658-41838-0. DOI: 10.1007/978-3-658-41838-0_33. URL: https://doi.org/10.1007/978-3-658-41838-0_33 (besucht am 28. 03. 2025).

[ESL25] ESLint. *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*. 21. März 2025. URL: https://eslint.org/ (besucht am 01. 04. 2025).

[Ged23] Ekrem Gedik. *Schwarz auf Weiß - die Unterschiede zwischen White-Box und Black-Box Testing - Qcademy*. Section: Testing. 4. Aug. 2023. URL: https://qcademy.de/schwarz-auf-weiss-die-unterschiede-zwischen-white-box-und-black-box-testing/ (besucht am 04. 04. 2025).

[Hel23] Marc Helmold. „Projektmanagement“. In: *Wettbewerbsvorteile entlang der Supply Chain sichern: Best-Practice-Beispiele in Beschaffung, Produktion, Marketing und anderen Funktionen der betriebswirtschaftlichen Wertschöpfungskette*. Hrsg. von Marc Helmold. Wiesbaden: Springer Fachmedien, 2023, S. 205–212. ISBN: 978-3-658-40609-7. DOI: 10.1007/978-3-658-40609-7_11. URL: https://doi.org/10.1007/978-3-658-40609-7_11 (besucht am 28. 03. 2025).

[Stu] StudySmarter. *Dokumentationsreview: Definition & Technik*. StudySmarter. URL: https://www.studysmarter.de/ausbildung/ausbildung-in-it/fachberater-softwaretechniken/dokumentationsreview/ (besucht am 03. 04. 2025).