Allgemeine Hinweise und Hilfestellungen:

1. Dieses Template ist ein exemplarisches Beispiel, wie ein Dokument strukturiert werden kann. Sie können dieses Dokument sinnvoll erweitern oder Inhalte neu anordnen, falls es aus Ihrer Sicht notwendig ist. Holen Sie sich Anregungen und adaptieren Sie: <https://www.arc42.de/overview/> Die Reihenfolge im Dokument kann der Reihenfolge der Abarbeitung entsprechen, wird es in der Regel jedoch nicht. Arbeiten Sie die einzelnen Punkte im Dokument so ab, wie sie Ihnen zeitlich und in der Reihenfolge besser passt. Die geforderten Punkte zu den einzelnen Praktika-Terminen sind dabei jedoch zu berücksichtigen!
2. Bedenken Sie: Alles, was Sie schreiben, macht Arbeit und das mehrfach! (beim Erstellen, beim Lesen, beim Kommentieren, beim Korrigieren, beim Nachpflegen etc.). Also gelten folgende Grundsätze:
   1. „In der Kürze liegt die Würze.“ oder „Qualität statt Quantität.“
   2. Nutzen Sie technische Hilfsmittel, wie toolbasierte Rechtschreibkorrektur!
   3. Sorgen Sie für Qualitätssicherung im Team (Gegenlesen etc.).
   4. Fassen Sie Informationen zweckmäßig zusammen und nutzen Sie dafür sinnvolle Gestaltungsmöglichkeiten (Tabellen, Grafiken, Aufzählungen, Verweise)

Versuchen Sie Inhalte sachlich und auf den Punkt zu formulieren. Nehmen Sie sich die Zeit, Sie sparen diese im späteren Verlauf des Projektes mit Faktor X wieder ein.

1. Vermeiden Sie Wiederholungen von Inhalten. Referenzieren Sie im Dokument! Nutzen Sie Verweise (auch bei Abbildungen und Tabellen) und eineindeutige Bezeichner. Beispiel: anstatt stumpf „ID 1“ besser „Req\_01“ z.B. für eine Nummerierung einer Anforderung.
2. Wählen Sie für wiederkehrende Begrifflichkeiten gute, einheitliche Bezeichner und dokumentieren Sie diese (Glossar, Abkürzungsverzeichnis, IO-Tabelle etc). Am besten lassen sich diese (gut gewählten) Bezeichner bis hin in den späteren Quellcode verwenden. Also keine Umlaute, Leerzeichen, Sonderzeichen etc. Wenn Kurzbezeichner intuitiv lesbar sind, erhöht das einerseits die Lesbarkeit und verringert andererseits die Schreibarbeit! Beispiel: „Mit freundlichen Grüßen“ à „MfG“
3. Vermeiden Sie Synonyme! Ja, es klingt dann manchmal nicht elegant, aber Sie schreiben ein technisches Dokument, keine Prosa. Also X bleibt X und wird nicht plötzlich zu X\_neu oder gar Y.
4. Jeder Kapitelüberschrift folgt ein kleiner einleitender Text. Verschaffen Sie dem Leser einen Überblick über das Kapitel, damit er selbst entscheiden kann, ob das Kapitel für ihn relevant ist (siehe z.B. der typische Aufbau von Lehr- und Fachbüchern).
5. Es ist empfehlenswert dem Leser das zu entwickelnde System verständlich zu erklären. Dabei hilft z.B. das Prinzip Top-Down und die Wahl sinnvoller Diagramme aus dem SE-Kontext. Wir werden Sie aus diesem Grund dazu auffordern, auch ein Grobkonzept zu erstellen! Das hilft auch manchmal im Team, um eine Diskussionsgrundlage zu haben.
6. Entfernen Sie aus ihrem Dokument alle roten Bearbeitungshinweise, bevor Sie das Dokument weiterreichen oder veröffentlichen (z.B. an Kunde, Betreuer, etc.). Zusätzlich aktualisieren Sie immer das Inhaltsverzeichnis.
7. Dokumentation und daraus abgeleitete Artefakte (Code, Tests, usw.) sind immer konsistent!
8. Versuchen Sie sich bestmöglich an die oben aufgeführten Punkte zu halten. Trainieren Sie technische Dokumentationen zu schreiben und starten Sie damit jetzt. Viel Erfolg!

Requirements / Design and Test Documentation

(RDT)

Version 0.8

ESEP – Praktikum – Semester Jahr

Team – Team-Bezeichner

(z.B. ESEP-WS2024-3.1 [3->Praktikumsgruppe, 1->Teamnummer])

*Die Teamnummer erhaltet ihr spätestens am ersten Praktikumstermin!*

Name, Vorname, Matrikel-Nr., E-Mail

Name, Vorname, Matrikel-Nr., E-Mail

Name, Vorname, Matrikel-Nr., E-Mail

Name, Vorname, Matrikel-Nr., E-Mail

Änderungshistorie:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Erstellt | Autor | Kommentar |
| 0.1 | 2018-03-12 | LMN | Initiale Version des Templates. |
| 0.2 | 2020-03-15 | DAI | Überarbeitung wegen Corona. |
| 0.3 | 2022-02-24 | LMN | Anpassungen für Sommersemester. Anforderungen an Requirements reduziert auf Ergänzungen. |
| 0.4 | 2022-11-22 ff. | CHRS | Neustrukturierung des Templates, Schriftgrößen vereinheitlicht, Erweiterungen: Hinweise am Anfang des Dokuments, Unterkapitel Hardware und technische Gegebenheiten, Unterkapitel Analyse des Kundenwunsches, Unterkapitel Nachrichten und Signale, allg. Abnahmetest Text + Tabelle, Unterkapitel Abbildungsverzeichnis |
| 0.5 | 2023-08-22 | CHRS | Korrekturen Rechtschreibung/Ausdruck, Erweiterung der Hilfestellungen, Erweiterung um „Grobkonzept“, Umformulierung und Erweiterung von Texten, Anpassung Titelblatt |
| 0.6 | 2023-09-18 | CHRS | Redaktionelle Bearbeitung |
| 0.7 | 2023-09-27 | CHRS | Bsp. für Test-Tabelle hinzugefügt |
| 0.8 | 2024-08-28 | CHRS | Redaktionelle Bearbeitung |
|  |  |  | <weitere Versionen> |

Inhaltsverzeichnis:

[1 Teamorganisation 7](#_Toc146699740)

[1.1 Verantwortlichkeiten 7](#_Toc146699741)

[1.2 Absprachen 7](#_Toc146699742)

[1.3 Repository-Konzept 7](#_Toc146699743)

[2 Projektmanagement 8](#_Toc146699744)

[2.1 Prozess 8](#_Toc146699745)

[2.2 Projektorganisation 8](#_Toc146699746)

[2.3 Risiken 8](#_Toc146699747)

[2.4 Qualitätssicherung 8](#_Toc146699748)

[3 Problemanalyse 9](#_Toc146699749)

[3.1 Analyse des Kundenwunsches 9](#_Toc146699750)

[3.1.1 Stakeholder 9](#_Toc146699751)

[3.1.2 Systemkontext des Systems 9](#_Toc146699752)

[3.1.3 Anforderungen 9](#_Toc146699753)

[3.1.4 Use Cases / User Stories 9](#_Toc146699754)

[3.2 Anlage: Analyse der technischen Gegebenheiten 10](#_Toc146699755)

[3.2.1 Technischer Aufbau und Hardwarekomponenten 10](#_Toc146699756)

[3.2.2 Werkstücke 10](#_Toc146699757)

[3.2.3 Anforderungen aus dem Verhalten und technischen Besonderheiten 10](#_Toc146699758)

[3.3 Softwareebene 11](#_Toc146699759)

[3.3.1 Systemkontext der Software 11](#_Toc146699760)

[3.3.2 Resultierende Anforderungen an die Software 11](#_Toc146699761)

[3.3.3 Schnittstellen: Nachrichten und Signale 11](#_Toc146699762)

[4 Grobkonzept des technischen Systementwurfes 12](#_Toc146699763)

[5 Software-Design 13](#_Toc146699764)

[5.1 Software Architektur 13](#_Toc146699765)

[5.2 Software Struktur 13](#_Toc146699766)

[5.3 Verhaltensmodellierung 13](#_Toc146699767)

[6 Implementierung: Besonderheiten 14](#_Toc146699768)

[7 Qualitätssicherung 15](#_Toc146699769)

[7.1 Teststrategie 15](#_Toc146699770)

[7.2 Testszenarien/Abnahmetest 15](#_Toc146699771)

[7.3 Testprotokolle und Auswertungen 15](#_Toc146699772)

[8 Technische Schulden 16](#_Toc146699773)

[9 Lessons Learned 16](#_Toc146699774)

[10 Anhang 16](#_Toc146699775)

[10.1 Glossar 16](#_Toc146699776)

[10.2 Abkürzungen 16](#_Toc146699777)

# Teamorganisation

Überlegen Sie, welche Regeln Sie für die Zusammenarbeit aufstellen und welche Rollen Sie im Team verteilen wollen. Dokumentieren Sie diese hier zusammen mit weiteren Anmerkungen der Teamorganisation.

## Verantwortlichkeiten

Benennen Sie Verantwortliche innerhalb des Projekts (Projektleiter, Tester, Implementierer, etc.).

|  |  |
| --- | --- |
| Verantwortlichkeit | Person/en |
| *<Verantwortlichkeit>* | *<Teammitglied, …>* |
|  |  |

## **Absprachen**

Listen Sie hier die Absprachen im Team auf, z. B. Jour Fixe, Kommunikation, Response-Latenz, ....

## Repository-Konzept

Überlegen Sie sich, wie Sie das Repository und die Ordner organisieren wollen. Welche Regeln wollen Sie beim Umgang mit Branches, Auslieferungen, Nachrichten an den Commits usw. im Team einhalten? Überlegen Sie, wie die Arbeitsabläufe sein sollen bei der Umsetzung von Arbeitsaufträgen oder bei der Behebung von Fehlern im Zusammenhang mit dem Repository. Listen Sie diese Absprachen hier auf.

# Projektmanagement

In diesem Kapitel sollten organisatorische Punkte beschrieben und festgelegt werden.

## Prozess

Legen Sie den Prozess fest, nach dem Sie das Projekt umsetzen wollen. Geben Sie ggf. grobe Schritte an, wie Planungsrunden, Sprints, oder ähnliches.

## Projektorganisation

Projektstrukturplan, Ressourcenplan, Zeitplan, Abhängigkeiten von Arbeitspaketen, eventueller Zeitverzug, Visualisierung des Projektstandes, etc.

## Risiken

Wenn Sie eine Risk-Matrix für Ihr Projekt erstellen, dann fügen Sie die Tabelle hier ein.

## Qualitätssicherung

Überlegen Sie, wie Sie Qualität in Ihrem Projekt insgesamt, nicht nur im Code, sicherstellen wollen. Listen Sie die Maßnahmen hier auf. Beachten Sie, dass diese Maßnahmen für die unterschiedlichen Artefakte und Ebenen entsprechend unterschiedlich sein können.

# Problemanalyse

Die Anforderungen aus der Aufgabenstellung sind nicht vollständig. Die Struktur der nachfolgenden Kapitel soll Sie bei der Strukturierung der Analyse unterstützen. Dokumentieren Sie die Ergebnisse der Analysen entsprechend.

## Analyse des Kundenwunsches

Dieses Unterkapitel ist der Schritt zwischen Systemanalyse und dem Softwareentwurf. Was hat der Kunde an Informationen bereitgestellt, was wünscht er sich? Welche Informationen sind bekannt, welche fehlen? Deckt sich der Kundenwunsch mit den Möglichkeiten der Anlage?

### Stakeholder

Ermitteln Sie die Stakeholder für das Projekt und listen Sie diese auf.

|  |  |
| --- | --- |
| Stakeholder | Interessen |
| *<Stakeholderbezeichner>* | *<kurze Interessenbeschreibung>* |
|  |  |

### Systemkontext des Systems

Um die Kapselung Ihres Gesamtsystems klar zu dokumentieren, sollten Sie ein Systemkontext erstellen. Der Systemkontext kann sich auch in einem Use Case Diagramm wiederfinden. Die Use Cases und Test Cases müssen zu der hier verwendeten Darstellung konsistent sein.

### Anforderungen

In der Aufgabenstellung sind Anforderungen an das System gestellt, die ggf. unvollständig oder unklar sind. Dokumentieren Sie hier nur die Ergänzungen, Klarstellungen, Verfeinerungen und ähnliches als Requirements. Beachten Sie auch mögliche Fehlbedienungen und mögliche Fehlverhalten des Systems.

|  |  |
| --- | --- |
| Lfd. Nr. / ID | Beschreibung |
| *< K\_REQ\_x>* | *<Beschreibung>* |
|  |  |

### Use Cases / User Stories

Falls Sie Use Cases oder User Stories zur besseren Darstellung der Anforderungen erstellen wollen, dokumentieren Sie hier, welche Use Cases/ User Stories Sie auf der Systemebene implementieren müssen. Die Test Cases sollen später zu den Use Cases/ User Stories konsistent sein.

*< Hier kommt die genaue Beschreibung der Use Cases. Pro Anforderung eine Tabelle benutzen (äquivalent zu SE1). Die Tabelle nach Belieben vervielfältigen. >*

Die Betreuung wird maximal zwei Use Cases reviewn.

## Anlage: Analyse der technischen Gegebenheiten

Dieses Unterkapitel soll sich mit den technischen Gegebenheiten auseinandersetzen. Mögliche Fragestellungen: Was für Hardware ist vorhanden und soll genutzt werden? Was sind Besonderheiten? Für die spätere Software nötige Maße und Eigenschaften der Hardware? Es sind nur für die Durchführung des Projektes notwendige Aspekte aufzuführen! Nutzen Sie die Möglichkeit sich mit dem System vertraut zu machen. Spielen Sie Szenarien direkt an den Festo-Anlagen durch und finden Sie so heraus, wie das System reagiert, arbeitet und dokumentieren Sie ihre Ergebnisse.

### Technischer Aufbau und Hardwarekomponenten

Verschaffen Sie sich einen Überblick, um was für ein technisches System es sich handelt und wie es aufgebaut ist bzw. wie es sich zusammensetzt. Hilfreich: Skizzen, Zeichnungen, Fotos, Beschreibungen. Welche Komponenten gibt es und welche Aufgabe haben sie? Welche Schnittstellen sind existent? Vermeiden Sie jedoch zu tief ins Detail zu gehen. Dafür gibt es Datenblätter.

### Werkstücke

Sichern Sie sich ab, indem Sie Werkstücke beschreiben und definieren. Welche Auswirkungen haben Werkstücke mit ihren Eigenschaften auf das zu entwickelnde System? Gibt es Besonderheiten?

### Anforderungen aus dem Verhalten und technischen Besonderheiten

Welche Punkte aus Blickwinkel der Anlage oder der Werkstücke sind wichtig, die eventuell Aspekte der zu entwickelnden Software beeinflussen? Schauen Sie detailliert auf das Verhalten der Anlage und spielen Sie Szenarien an der Anlage durch, um Besonderheiten zu erkennen. Dokumentieren Sie ihre Ergebnisse.

|  |  |
| --- | --- |
| Lfd. Nr. / ID | Beschreibung |
| *<HW\_REQ\_x>* | *<Beschreibung>* |
|  |  |

## Softwareebene

Sie sollen Software für die Steuerung des technischen Systems erstellen. Aus den Anforderungen auf der Systemebene (Kundenwünsche etc.) und den Eigenschaften des technischen Prozesses ergeben sich Anforderungen für Ihre Software. Insbesondere wird sich eventuell die Software der beiden Anlagenteile in einigen Punkten unterscheiden.

### Systemkontext der Software

Wie sieht der Kontext Ihrer Software aus?

### Resultierende Anforderungen an die Software

Welche wesentlichen Anforderungen leiten sich aus den Systemanforderungen für Ihre Software ab? Berücksichtigen Sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems. Klare, sich aus der Aufgabenstellung ergebende Anforderungen brauchen nicht zwingend neu aufgelistet werden. Verfeinerungen, neue Anforderungen oder Anforderungen für eine bessere Klarstellung, sollten gelistet werden.

|  |  |
| --- | --- |
| Lfd. Nr. / ID | Beschreibung |
| *< SW\_REQ\_x>* | *<Beschreibung>* |
|  |  |

### Schnittstellen: Nachrichten und Signale

Welche ein- und ausgehenden Signale/Nachrichten ergeben sich aus der Analyse der Anlage und aus den Anforderungen an das SW-System? Wie erfolgt die Kommunikation mit Nachbarsystemen? Gruppieren Sie die Nachrichten und Signale sinnvoll und teilen Sie diese in einzelne Tabellen auf.

Hinweis: Führen Sie hier schon sinnvolle und nachvollziehbare Kurzbezeichner ein, vergessen Sie aber nicht diese in den Tabellen auch entsprechend zu erläutern und zu beschreiben. Es bieten sich wiederkehrende Muster/Schemata für die Vergabe von Label und Kurzbezeichner an.

Nutzen Sie diese Bezeichner spätestens ab hier dann auch fortlaufend im Dokument und Ihrem Entwicklungsprozess!

# Grobkonzept des technischen Systementwurfes

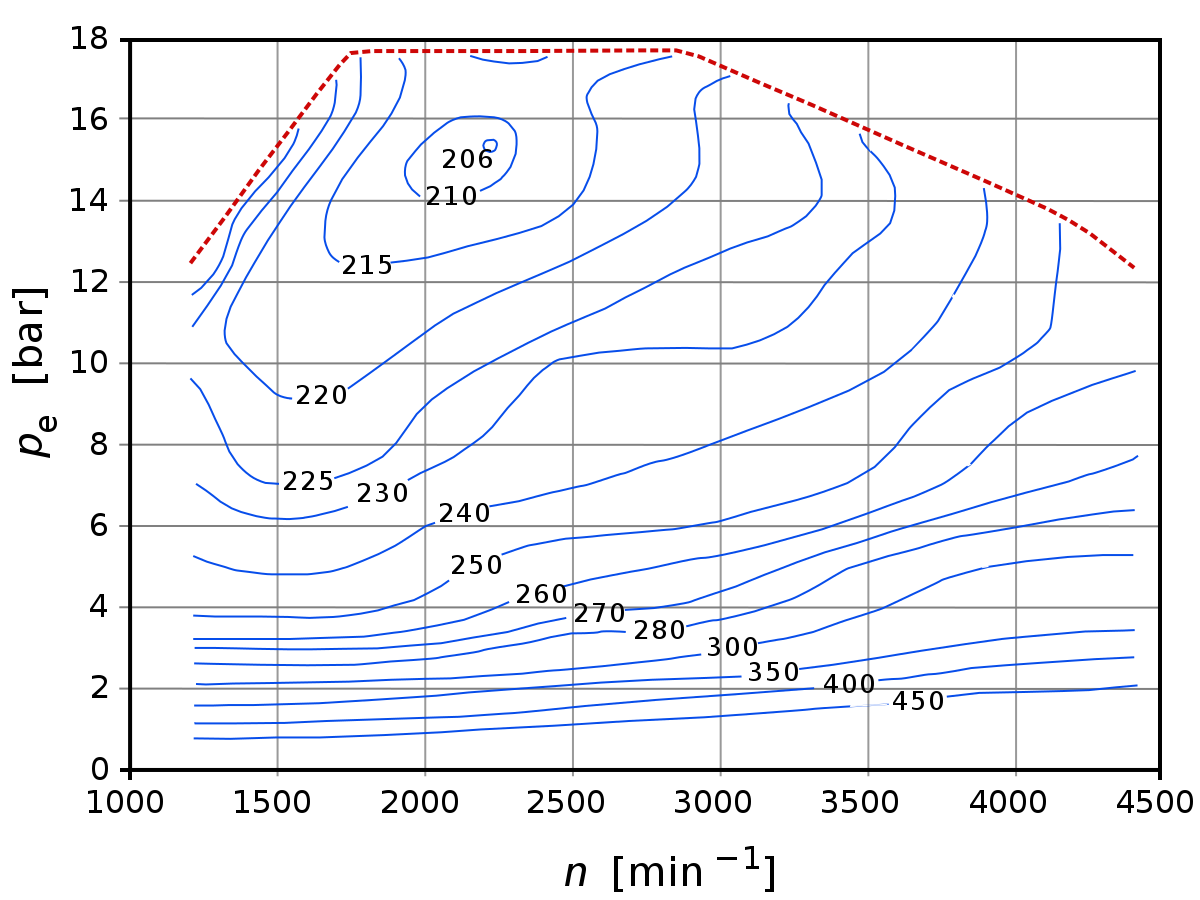
Beschreiben Sie kurz und knapp auf **EINER** Seite, wie Ihr Konzept für das finale Produkt aussehen soll. Mögliche Aspekte sind: Systemaufbau, Grobarchitektur, System-Kommunikation, Besonderheiten, etc.

Fiktives Bsp.:

*Wir bauen ein Auto. Das Auto hat Platz für 1 Person, ist ein 1-Türer und soll in der Farbe silbern angeboten werden. Wir bauen auf E-Antrieb und verzichten auf einen Kofferraum, um möglichst viel Platz für den Passagier im Innenraum zu bekommen. Das Fahrzeug wird aus einem Monocoque aufgebaut. Besonderheit: Die Zeit vergeht während der Fahrt wie im Fluge. Wir nutzen dafür …*

Ziel dieser Beschreibung:

Der Leser / der Betreuer / das andere Teammitglied kann schnell verstehen, was ihr vorhabt.



Ein Bild, das Entwurf, Landfahrzeug, Fahrzeug, Zeichnung enthält.

Automatisch generierte Beschreibung

Quellen: https://blog.atomlabor.de/das-back-to-future-delorean-design-von https://de.wikipedia.org/wiki/Verbrauchskennfeld

Ein Bild, das Entwurf, Zeichnung, Kunst, Kunststück enthält.

Automatisch generierte Beschreibung

# Software-Design

Strukturen, Verhalten und Bezeichner im Code müssen mit dem Modell übereinstimmen. Daher ist ein wohlüberlegtes Design wichtig.

Anmerkung: Die Implementierung MUSS zu Ihrem Design-Modell konsistent sein.

## Software Architektur

Erstellen Sie eine Architektur für Ihre Software. Geben Sie eine kurze Beschreibung der Komponenten und Schnittstellen an. Dokumentieren Sie hier wichtige technische Entscheidungen. Welche Patterns werden gegebenenfalls verwendet? Wie erfolgt die interne Kommunikation?

## Software Struktur

Der nächste Schritt ist die Verfeinerung der Architektur. Zeigen Sie die Struktur ihrer Software-Komponenten und dokumentieren Sie sie mit Hilfe von UML Klassendiagrammen, unter Beachtung der Designprinzipien. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier ist dann ein Übersichtsbild einzufügen.

Ergänzen Sie jeweils eine kurze textuelle Beschreibung ihrer Komponenten mit Erläuterungen der Zuständigkeiten (Responsibility), Schnittstellen usw.

## Verhaltensmodellierung

Ihre Software muss zur Bearbeitung der Aufgaben ein Verhalten aufweisen/abbilden. Überlegen Sie sich dieses Verhalten auf Basis der Anforderungen und modellieren Sie das Verhalten unter Verwendung von Verhaltensdiagrammen aus den Vorlesungen.

# Implementierung: Besonderheiten

Anmerkung: Nur wichtige Implementierungsdetails sollen hier erklärt werden. Code-Beispiele (snippets) können hier aufgelistet werden, um der Erklärung zu dienen.

Anmerkung: Bitte KEINE ganzen Programme hierhin kopieren!

# Qualitätssicherung

Machen Sie sich auf Basis Ihrer Überlegungen zur Qualitätssicherung Gedanken darüber, wie Sie die Erfüllung der Anforderungen möglichst automatisiert im Rahmen von Teststufen (Unit-Test, Komponententest, Integrationstest, Systemtest, Regressionstest und Abnahmetest) überprüfen werden.

## Teststrategie

Definieren Sie Zeitpunkte für die jeweiligen Teststufen in Ihrer Projektplanung. Dazu können Sie die Meilensteine zu Hilfe nehmen. Überlegen Sie, wie die Test-Architektur der jeweiligen Teststufen aussieht. Verwenden Sie Testmethoden wie z.B. Grenzwertanalyse, 100% Zustandsabdeckung, 100% Transitionsüberdeckung, Tiefensuche, Breitensuche, etc. Versuchen Sie Ihre Tests zu automatisieren.

## Testszenarien/Abnahmetest

Leiten Sie die Abnahmebedingungen aus den Kunden-Anforderungen her. Dokumentieren Sie hier, welche Schritte für die einzelnen Abnahmetests erforderlich sind und welches Ergebnis jeweils erwartet wird (Test Cases). Abnahmetests sind Blackbox-Tests auf Systemebene!

Nutzen Sie bitte eine tabellarische Darstellungsform. Hier eine mögliche Darstellungsform (ohne Gewähr auf Vollständigkeit!) und passen Sie diese entsprechend an oder nutzen Sie aus SE1 bekannte Tabellenformen. (z.B. Prof. Lehmann, Vorlesungsunterlagen SE1)

|  |  |
| --- | --- |
| <Test Nr> | <Titel> |
| Requirements: |  |
| Kurzbeschreibung: |  |
| Vorbedingungen: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Schritt | Aktion | Erwartung | Erfüllt |
| 1 | <Welche Aktion muss durchgeführt werden?> | <Was ist die erwartete, beobachtbare Reaktion nach dem Schritt?> |  |
| 2 |  |  |  |
| 3 |  |  |  |
|  |  |  |  |

## Testprotokolle und Auswertungen

Hier fügen Sie die Test Protokolle bei, auch wenn Fehler bereits beseitigt worden sind, ist es schön zu wissen, welche Fehler einst aufgetaucht waren. Eventuelle Anmerkung zur Fehlerbehandlung kann für weitere Entwicklungen hilfreich sein.

Das letzte Testprotokoll ist das Abnahmeprotokoll, das bei der abschließenden Vorführung erstellt wird. Es enthält eine Auflistung der erfolgreich vorgeführten Funktionen des Systems sowie eine Mängelliste mit Erklärungen der Ursachen der Fehlfunktionen und Vorschlägen zur Abhilfe

# Technische Schulden

Führen Sie bekannte technische Mängel Ihres Produktes auf. Zeigen Sie dem Kunden, dass Ihnen bewusst ist, was noch an offenen Punkten existieren. Damit zeigen Sie unter anderem, dass sie den Überblick über Ihr Projekt haben. Optional: Wieviel Zeit würde eine mögliche Beseitigung von einzelnen, offenen Arbeitspaketen umfassen?

# Lessons Learned

Führen Sie ein Teammeeting durch, in dem gesammelt wird, was gut gelaufen war, was schlecht gelaufen war und was man im nächsten Projekt (z.B. im PO) besser machen muss und will. Listen Sie für die Aspekte jeweils mindestens drei Punkte auf. Weitere Erfahrungen und Erkenntnisse können hier ebenso kommentiert werden, auch Anregungen für die Weiterentwicklung des Praktikums.

# Anhang

## Glossar

Eindeutige Begriffserklärungen, vorteilhaft ist eine alphabetische Sortierung der zu erklärenden Begriffe.

## Abkürzungen

Listen Sie alle wichtigen Abkürzungen auf, die Sie in diesem Dokument benutzt haben. Wählen Sie eine sinnvolle Unterteilung (Signale/Events, Hardwarebezeichner etc.). Nutzen Sie eine alphabetische Sortierung!