**Anforderungs- und Entwurfsspezifikation (“Pflichtenheft”)**

Hinweis: Die Spezifikation ist keine Arbeitsbeschaffungsmaßnahme, sondern dient dazu, Sachverhalte so früh wie möglich vor der Implementierung abzuklären. Denn je später ein Fehler entdeckt wird, desto aufwändiger ist es, diesen Fehler zu beheben.

* Titel, Autoren, Inhaltsverzeichnis

**1 Einführung**

**1.1 Beschreibung**

* Elevator-Pitch Statement kopieren

**1.2 Ziele**

* Anwendungsbereiche, Motivation, Umfang, Alleinstellungsmerkmale, Marktanforderungen
* Informationen zu Zielbenutzergruppen und deren Merkmale (Bildung, Erfahrung, Sachkenntnis)
* Abgrenzung (Was ist das Softwaresystem *nicht*)
* ggfs. SWOT-Analyse

Die Anwendungsbereiche von V.I.G-Mini liegen in der automatisierten und dokumentierten Zucht diverser Pflanzenarten, insbesondere in der Aufzucht von Setzlingen. Durch seine Kompakten Größe eignet es sich ideal für den Innenbereich wie bspw. Wohnzimmern, Küchen, oder Kinderzimmern. Durch die vielseitigen Funktionalitäten bietet sich V.I.G-Mini für Klein- und Hobbygärtner an, die Umfangreiche Daten über ihre Pflanzen sammeln möchten und repetitive Aufgaben automatisieren wollen. Nutzer benötigten keine Fortschrittlichen Kenntnisse in der Zucht um das Gewächshaus benutzten zu können. Auch der Technische Aspekt von V.I.G-Mini ist intuitiv gehalten und hält sich an die Regel „Einstecken und loslegen“ (Plug and Play). Das Rudimentäre Verständnis vom Ablauf und bedienen IT-technischer Geräte/Clients ist empfehlenswert. Jedoch ist ein Grundverständnis über Mathematik erforderlich, um die, von V.I.G-Mini erstellten Graphen, interpretieren zu können.

Abgrenzung

V.I.G-Mini eignet sich, aufgrund der Größe des Gewächshauses wie das „Mini“ vermuten lässt, nicht für das Kommerzielle Anbauen von Pflanzen. Auch für die Agrarwirtschaft ist V.I.G nicht geeignet.

**2 Anforderungen**

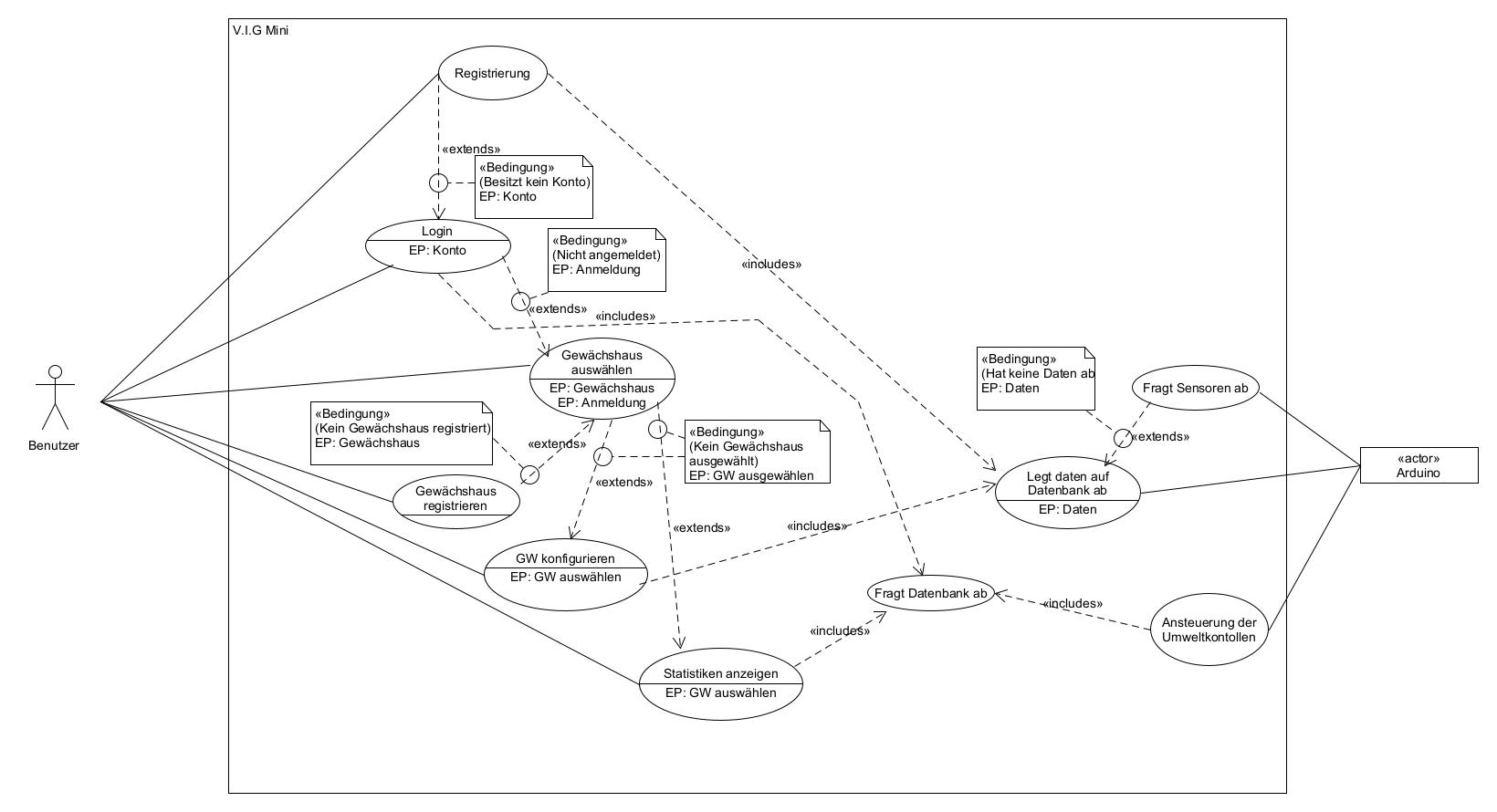
**2.1 Stakeholder**

| **Funktion / Relevanz** | **Name** | **Kontakt / Verfügbarkeit** | **Wissen** | **Interessen / Ziele** |
| --- | --- | --- | --- | --- |
| **Verantwortlicher für Serversystem** | Lukas Beckmann |  | Kenntnisse in OracleSql, Linux, SQL-Modellierung | Stabiles System, einfacher Zugang |
| **Programmierer für eingebettetes System** | Kai-Ulrich Stieler | Kai.stieler@outlook.de | Grundlegende Kenntnisse in elektro-Technik, Kenntnisse in OOP, C/C++, Netzwerktechnik, Skriptsprachen wie Python, |  |
| **Programmierer für Front und Backend** | Nils Jollet | Nils-jollet@gmx.de | Kenntnisse in OOP, C/C++, Netzwerktechnik, Skriptsprachen wie Python, Grundkenntnisse html, | Web-Applikation |
| **Programmierer für Front und Backend** | Lukas Beckmann | lukas-beckmann@gmx.net | Kenntnisse in OOP, C/C++, Netzwerktechnik, Skriptsprachen wie Python, Grundkenntnisse html, | Android-App |
| **Verantwortlicher** | Kai-Ulrich Stieler | Kai.stieler@outlook.de | Erfahrung mit Teamleitung, | Projekterfolg |
| **Dokumentation** | Kai-Ulrich Stieler | Kai.stieler@outlook.de | Erfahrung im schreiben von Sachtexten | Sachliche und detaillierte Projektdokumentation |
| **Dokumentation** | Nils Jollet | Nils-jollet@gmx.de | Erfahrung im schreiben von Sachtexten | Sachliche und detaillierte Projektdokumentation |
| **Zeitmanager** | Lukas Beckmann | lukas-beckmann@gmx.net |  | Einhalten von Fristen, Zeitpuffer hinzufügen |
| **Personalwesen** | Lukas Beckmann | lukas-beckmann@gmx.net |  | Freundliches Arbeitsklima |
| **(extern) Vertriebler** | Freddy Kruger | Freddy.kruger@gmx.de | --- | Ein funktionstüchtiges, gemäß des Pflichtenhefts erfülltes, Produkt, Ansprechpartner |
| **(extern) Kunde** | PflanzenUndMehr GmbH | info@pum.de | Marktführer in Automatisierungen im | Geldgeber |

**Beispiel**

**2.2 Funktionale Anforderungen**

* Strukturierung der Anforderungen in funktionale Gruppen
* Definition der Akteure
  + Benutzer (Interagiert mit dem Client und macht eingaben/ bekommt ausgaben)
  + Aduino (Kommuniziert mit dem Server erhält daten über die Sensoren. Steuert Licht/Wasserpumpe/Lüfter an.
* ggfs. Use-Case Diagramme



* Akteure sowie andere Begriffe der implementierten Fachdomäne definieren
* Begriffe konsistent in der Spezifikation verwenden
* Begriffe im Glossar am Ende des Dokuments darstellen

**2.3 Nicht-funktionale Anforderungen**

**2.3.1 Rahmenbedingungen**

* Normen, Standards, Protokolle, Hardware, externe Vorgaben
* Hardware:
* W-Lan fähiger, 32-Bit-Mikrocontroller (ESP32), über ihm sind die Sensoren, Lüfter usw. angesprochen
* Luftfeuchtigkeitssensor, Temperatursensor, Bodenfeuchtigkeitssensor, Lichtsensor, zur Aufzeichnung von Messwerten
* Steckbretter, Leitungen, Transistoren, Widerstände, Datenkabel, Netzkabel
* Server zum hosten der Server-Anwendung
* Software:
* Server-Anwendungen wie:
* Datenbankserver, zur Verwaltung/speichern der Nutzeraccount, Gewächshäuser und Messwerten
* Web-Server, um dem Nutzer einen grafische Benutzeroberfläche bieten zu können
* Java, C/C++, Intellij-IDE, GUI-Framework,
* ggf. flask, django, kotlin
* ArduinoIDE
* externe Vorgaben:
* Bedienungsfreundlichkeit
* „Plug and Play“
* Datenschutz von Personenbezogenen Daten

**2.3.2 Betriebsbedingungen**

* Vorgaben des Kunden (z.B. Web Browser / Betriebssystem Versionen, Programmiersprache)
* Browseranwendung für Google Chrome und Firefox
* Programmiersprachen benutzen wie: Java, C/C++,php, html, javascript, ggf. flask, django, kotlin
* Computerclient für Windows10 in Java
* Android App
* Open Source Code benutzen

**2.3.3 Qualitätsmerkmale**

* Externe Qualitätsanforderungen (z.B. Performance, Sicherheit, Zuverlässigkeit, Benutzerfreundlichkeit)

| **Qualitätsmerkmal** | **sehr gut** | **gut** | **normal** | **nicht relevant** |
| --- | --- | --- | --- | --- |
| **Zuverlässigkeit** |  |  |  |  |
| Fehlertoleranz |  | X | - | - |
| Wiederherstellbarkeit |  | - | -X | - |
| Ordnungsmäßigkeit | X | - | - | - |
| Richtigkeit | X | - | - | - |
| Konformität | X |  | - | - |
| **Benutzerfreundlichkeit** |  |  |  |  |
| Installierbarkeit | X | - |  | - |
| Verständlichkeit | X | - | - | - |
| Erlernbarkeit | - | X | - | - |
| Bedienbarkeit | - | X | - | - |
| **Performance** |  |  |  |  |
| Zeitverhalten | - | - |  | X |
| Effizienz | - | - | - | X |
| **Sicherheit** |  |  |  |  |
| Analysierbarkeit |  | - | X | - |
| Modifizierbarkeit | - | - | -X |  |
| Stabilität | X | - | - | - |
| Prüfbarkeit |  | X | - | - |

**2.4 Graphische Benutzerschnittstelle**

* GUI-Mockups passend zu User Stories
* Screens mit Überschrift kennzeichnen, die im Inhaltsverzeichnis zu sehen ist
* Unter den Screens darstellen (bzw. verlinken), welche User Stories mit dem Screen abgehandelt werden
* Modellierung der Navigation zwischen den Screens der GUI-Mockups als Zustandsdiagramm
* Mockups für unterschiedliche Akteure
* Mockups für unterschiedliche Frontends (Mobil, Web, Desktop)

**2.5 Anforderungen im Detail**

* User Stories mit Akzeptanzkritierien
* Optional: Name (oder ID) und Priorität (“Must”, “Should”, “Could”, “Won’t”)
* Strukturierung der User Stories in funktionale Gruppen
* Sicherheit: Misuse-Stories formulieren
* -

1. Gewächshaus:
   1. Senden/empfangen von Daten zur Datenbank mittels Wlan
   2. Produktcode hinzufügen
   3. Sensoren und andere Hardware ans Gewächshaus anbringen
2. Sensoren (Gewächshaus):
   1. Messen von Luftfeuchtigkeit
   2. Messen von Bodenfeuchtigkeit
   3. Messen von Temperatur
3. aktive Funktionen (Gewächshaus):
   1. Lüfter steuern können (in Abhängigkeit der gewünschten Werte des Nutzers)
   2. LED steuern (in Abhängigkeit des Zeitplans)
   3. Pumpe zum bewässern Steuern (in Abhängigkeit der gewünschten Werte des Nutzers)
4. Datenbank (Server) (siehe ER-Modell)
   1. Nutzeraccount speichern
   2. Nutzer mit Gewächshäusern verknüpfen
   3. Gewächshaus-Registercode speichern
   4. die Möglichkeit Nutzereinstellungen zu speichern
   5. die Möglichkeit Messwerte pro Gewächshaus zu speichern
5. Web-Server, Schnittstelle PHP, (Server)
   1. Daten bereitstellen für Client
   2. Daten empfangen von ESP32
   3. Messwerte speichern in Datenbank
   4. Zeitplan speichern in Datenbank
   5. Messgrenzen speichern in Datenbank
6. Web-Server, Benutzeroberfläche, progressive Web-App(Server)
   1. Nutzer
      1. Login
      2. Nutzer kann sich registrieren
      3. Nutzer kann Passwort ändern
      4. Nutzer kann email ändern
      5. Nutzer kann Vor- und Nachname ändern
   2. Gewächshaus
      1. Gewächshaus registrieren/verknüpfen
      2. Statistiken anzeigen (Tages-, Wochen- und Monatsverlauf)
      3. Zeitplan erstellen (um welche Uhrzeit soll belüftet, gewässert, beleuchtet werden)
      4. Funktionen aus 5d) nur in Abhängigkeit der Messwerte ausführen (bis auf Licht)
7. Android App
   1. Nutzer
      1. Login
      2. Nutzer kann sich registrieren
      3. Nutzer kann Passwort ändern
      4. Nutzer kann email ändern
      5. Nutzer kann Vor- und Nachname ändern
   2. Gewächshaus
      1. Gewächshaus registrieren/verknüpfen
      2. Statistiken anzeigen (Tages-, Wochen- und Monatsverlauf)
      3. Zeitplan erstellen (um welche Uhrzeit soll belüftet, gewässert, beleuchtet werden)
      4. Funktionen aus 5d) nur in Abhängigkeit der Messwerte ausführen (bis auf Licht)
8. JavaClient
   1. Nutzer
      1. Login
      2. Nutzer kann sich registrieren
      3. Nutzer kann Passwort ändern
      4. Nutzer kann email ändern
      5. Nutzer kann Vor- und Nachname ändern
   2. Gewächshaus
      1. Gewächshaus registrieren/verknüpfen
      2. Statistiken anzeigen (Tages-, Wochen- und Monatsverlauf)
      3. Zeitplan erstellen (um welche Uhrzeit soll belüftet, gewässert, beleuchtet werden)
      4. Funktionen aus 5d) nur in Abhängigkeit der Messwerte ausführen (bis auf Licht)

**Schablone für User Stories**

| **Name** | **In meiner Rolle als…** | **…möchte ich…** | **…, so dass…** | **Erfüllt, wenn…** | **Priorität** |
| --- | --- | --- | --- | --- | --- |
| Statistiken | Benutzer | Unkompliziert und übersichtlich die Statistiken angezeigt bekommen | ich eventuell Veränderungen an den Einstellungen vornehmen kann | Eine Tabelle aller wichtigen Daten angezeigt wird und diese sortierbar ist | Muss |
| Bedienbarkeit | Benutzer | Einfach zu bedienende Oberfläche | Ich mich nicht einarbeiten muss | Die GUI selbsterklärend ist | Sollte |
|  | Benutzer | Mich um nichts mehr kümmern | Ich mich auf andere Sachen konzentrieren kann | Das Gewächshaus automatisiert abläuft | Muss |
|  | Benutzer | Dass ich mich überall anmelden kann | Ich immer mein Gewächshaus im blick hab | Die Datenbank von überall erreichbar ist | Should |
| Sensoren | Benutzer | Die Luftfeuchtigkeit, Bodenfeuchtigkeit und die Temperatur meines Gewächshauses kontrollieren können | Ich das Wachstum meiner Pflanzen besser kontrollieren kann | Die Sensordaten werden regelmäßig überprüft und in einer Datenbank abgespeichert | Muss |
| Steuerung | Benutzer | Das Gewächshaus steuern können | Meine Pflanzen den bestmöglichen Ort zum Gedeihen haben | Der Benutzer durch einen Zeitplan oder durch einstellen der Grenzwerte, den Lüfter, die Wasserpumpe und/oder die Lampe einstellen können | Muss |

**3 Technische Beschreibung**

**3.1 Systemübersicht**

* Systemarchitekturdiagramm (z.B. als UML-Deployment-Diagramm)
* Kommunikationsprotokolle, Datenformate

Das Diagramm in Kapitel “Systemübersicht” ist statisch und nicht dynamisch und stellt daher keine Abläufe dar. Abläufe werden im Kapitel “Abläufe” dargestellt. Im Kapitel “Systemübersicht” soll genau ein Diagramm dargstellt werden. Das “Box-and-Arrow”-Diagramm soll als Systemarchitekturdiagramm eine abstrakte Übersicht über das Softwaresystem geben. Dazu stellt es die Rechnerknoten und deren Kommunikationsbeziehungen (Protokoll (z.B. HTTP), Datenformat (z.B. JSON)) dar. Also Rechtecke und gerichtete Pfeile. Ähnlich einem UML-Deployment-Diagramm, aber noch abstrakter, denn es zeigt nicht die Verteilung der Softwarebausteine auf die Rechnerknoten. So erlangt der Leser einen schnellen und guten Überblick über das Softwaresystem.

**3.2 Softwarearchitektur**

* Darstellung von Softwarebausteinen (Module, Schichten, Komponenten)

Hier stellen Sie die Verteilung der Softwarebausteine auf die Rechnerknoten dar. Das ist die Softwarearchitektur. Zum Beispiel Javascript-Software auf dem Client und Java-Software auf dem Server. In der Regel wird die Software dabei sowohl auf dem Client als auch auf dem Server in Schichten dargestellt.

* Server
  + Web-Schicht
  + Logik-Schicht
  + Persistenz-Schicht
* Client
  + View-Schicht
  + Logik-Schicht
  + Kommunikation-Schicht

Die Abhängigkeit ist bei diesen Schichten immer unidirektional von “oben” nach “unten”. Die Softwarearchitektur aus Kapitel “Softwarearchitektur” ist demnach detaillierter als die Systemübersicht aus dem Kapitel “Systemübersicht”. Die Schichten können entweder als Ganzes als ein Softwarebaustein angesehen werden. In der Regel werden die Schichten aber noch weiter detailliert und in Softwarebausteine aufgeteilt.

**3.3 Schnittstellen**

* Schnittstellenbeschreibung (API)
* Auflistung der nach außen sichtbaren Schnittstelle der Softwarebausteine

Hier sollen sämtliche Schnittstellen definiert werden:

* die externen Schnittstellen nach außen. Über welche Schnittstelle kann z.B. der Client den Server erreichen?
* die internen Schnittstellen der unter 3.2 definierten Softwarebausteine

**3.3.1 Ereignisse**

* In Event-gesteuerten Systemen: Definition der Ereignisse und deren Attribute

**3.4 Datenmodell**

* Konzeptionelles Analyseklassendiagramm (logische Darstellung der Konzepte der Anwendungsdomäne)
* Modellierung des implementierungsnahhen Datenmodells
  + RDBMS: ER-Diagramm bzw. Dokumentenorientiert: JSON-Schema

**3.5 Abläufe**

* Ggfs. Aktivitätsdiagramme für relevante Use Cases
* Aktivitätsdiagramm für den Ablauf sämtlicher Use Cases
* Aktivitätsdiagramm mit Swimlanes sind in der Regel hilfreich  
  für die Darstellung der Interaktion von Akteuren der Use Cases / User Stories
* Abläufe der Kommunikation von Rechnerknoten (z.B. Client/Server)  
  in einem Sequenz- oder Aktivitätsdiagramm darstellen
* Modellieren Sie des weiteren die Diagramme, die für das (eigene) Verständnis des  
  Softwaresystems hilfreich sind.

**3.6 Entwurf**

* Detaillierte UML-Diagramme für relevante Softwarebausteine

**3.7 Fehlerbehandlung**

* Mögliche Fehler / Exceptions auflisten
* Fehlercodes / IDs sind hilfreich
* Nicht nur Fehler technischer Art (“Datenbankserver nicht erreichbar”) definieren, sondern auch im Hinblick auf  
  Kapitel 3.8 sind fachliche Fehler wie “Kunde nicht gefunden”. “Nachricht wurde bereits gelöscht” o.ä.

**3.8 Validierung**

* Relevante (Integrations)-Testfälle, die aus den Use Cases abgeleitet werden können
* Testfälle für
  + Datenmodell
  + API
  + User Interface
* Fokusieren Sie mehr auf Integrationstestfälle als auf Unittests
* Es bietet sich an, die IDs der Use Cases / User Stories mit den Testfällen zu verbinden,  
  so dass erkennbar ist, ob Sie alle Use Cases getestet haben.

**4 Projektorganisation**

**4.1 Annahmen**

* Nicht durch den Kunden definierte spezifische Annahmen, Anforderungen und Abhängigkeiten
* Verwendete Technologien (Programmiersprache, Frameworks, etc.)
* Aufteilung in Repositories gemäß Software- und Systemarchitektur und Softwarebbausteinen
* Einschränkungen, Betriebsbedingungen und Faktoren, die die Entwicklung beeinflussen (Betriebssysteme, Entwicklungsumgebung)
* Interne Qualitätsanforderungen (z.B. Softwarequalitätsmerkmale wie z.B. Erweiterbarkeit)

**4.2 Verantwortlichkeiten**

* Zuordnung von Personen zu Softwarebausteinen aus Kapitel “Systemübersicht” und “Softwarearchitektur”
* Rollendefinition und Zuordnung

| **Softwarebaustein** | **Person(en)** |
| --- | --- |
| Komponente A | Thomas Mustermann |

**Rollen**

**Softwarearchitekt**

Entwirft den Aufbau von Softwaresystemen und trifft Entscheidungen über das Zusammenspiel der Softwarebausteine.

**Frontend-Entwickler**

Entwickelt graphische oder andere Benutzerschnittstellen, insbesondere das Layout einer Anwendung.

**Backend-Entwickler**

Implementiert die funktionale Logik der Anwendung. Hierbei werden zudem diverse Datenquellen und externe Dienste integriert und für die Anwendung bereitgestellt.

**Rollenzuordnung**

| **Name** | **Rolle** |
| --- | --- |
| Kai-U Stieler | Softwarearchitekt |
| Nils Jollet | Frontend/Backend-Entwickler |
| Lukas Beckmann | Frontend/Backend-Entwickler |

**4.3 Grober Projektplan**

* Meilensteine

**Meilensteine**

* KW 43 (21.10)
  + Abgabe Pflichtenheft
* KW 44 (28.10) / Projekt aufsetzen
  + Repository Struktur
* KW 45 (4.11) / Implementierung
  + Implementierung #3 (Final)
* KW 48 (18.12) / Abnahmetests
  + manuelle Abnahmetestss
  + Präsentation / Software-Demo

**5 Anhänge**

**5.1 Glossar**

* Definitionen, Abkürzungen, Begriffe

**5.2 Referenzen**

* Handbücher, Gesetze
* z.B. Datenschutzgrundverordnung

**5.3 Index**