Definition of Ready

Erstellt eure Team-eigene DoR --> hier sollte JEDES Teammitglied beteiligt sein und mitbestimmen dürfen!! Definiert es so, dass ihr das für JEDES Issue auch einhalten könnt. Es scheint zunächst aufwändig, aber es spart euch letztlich wiederum Zeit, denn es vermeidet Fehler und Missverständnisse durch die klare Formulierung von Erwartungen

- Muss erfüllt sein, bevor das Issue von der Liste "Open" in die Liste "Backlog" übergehen darf
- Nur Tickets aus der Liste "Backlog" dürfen in den nächsten Sprint übergehen
- Allgemeine Good Practice ist die Orientierung an den sog. INVEST-Kriterien:
  - Independent (unabhängig)
  - Egal, an welchem Backlog-Punkt du gerade arbeitest, er darf nicht von einer anderen Aufgabe abhängig sein, d. h. der Backlog-Punkt muss eigenständig sein. Auf diese Weise kann dein Team unnötige Arbeit vermeiden.
  - Negotiable (verhandelbar)
  - Eine Aufgabe sollte nicht starr sein. Du musst flexibel genug sein, um andere Optionen in Betracht zu ziehen, die das Team anbieten könnte.
  - Valuable (wertvoll)
  - Deine Arbeit muss einen Zweck haben. Noch wichtiger ist, dass sie dem Produkt, dem Kunden und dem Unternehmen einen Mehrwert liefert.
  - Estimable (schätzbar)
  - Die Aufgabe muss machbar, erreichbar und messbar sein. Dein Team muss wissen, wie viel Zeit und Aufwand es einplanen muss. Wenn der Sprint mehrere Aufgaben erfordert, gilt dies für jede einzelne Aufgabe.
  - Klein
  - Die Arbeit muss überschaubar sein. Wenn eine Aufgabe komplex ist, solltest du sie in kleinere Aufgaben unterteilen können. Dadurch werden "Brandbekämpfungen" und hektisches Arbeiten zur Einhaltung unangemessener Deadlines verhindert. Außerdem schützt du so dein Team vor einem Burn-out.
  - Testable (testbar)
  - Spezifiziere die Erfolgs- und Abschlusskriterien auf Grundlage der Unternehmens- und Benutzeranforderungen. Anhand dieser Kriterien kann dein Team beurteilen, ob die Aufgabe abgeschlossen ist.

Quelle: *https://www.atlassian.com/de/agile/project-management/definition-of-ready*

Eine pragmatische konkrete Umsetzung für dieses Projekt wären folgende Kriterien in der DoR:

- Issue hat eine knappe aber aussagekräftige Überschrift
- Issue hat mind. Ein Akzeptanzkriterium (mehr dazu nachfolgend)
- Issue ist einem Meilenstein zugeordnet (der wiederum eine definierte Laufzeit und somit Deadline hat)
- Issue hat ein Due Date (innerhalb des Meilensteins!)
- Aufwand des Issues ist geschätzt (in h; ein Limit setzen, z.B. nicht größer als 10)
- Daumenregel: Ein Issue muss zeitlich von einer Person in einem Sprint erledigt werden können!

Tipps zur Formulierung von Akzeptanzkriterien

- Liste an Kriterien, damit wir Product Owner und ihr im Team die Aufgabe als erfüllt akzeptiert.
- Formulieren, was das Ergebnis / der Zielzustand sein soll
- Klar und präzise sein: Akzeptanzkriterien sollten eindeutig und verständlich formuliert sein, um Missverständnisse zu vermeiden.
- Messbar und überprüfbar: Sie sollten so formuliert sein, dass man klar feststellen kann, ob sie erfüllt sind oder nicht.
- Benutzerperspektive einnehmen: Formuliere die Kriterien aus der Sicht des Endbenutzers, um sicherzustellen, dass die Anforderungen den tatsächlichen Bedürfnissen entsprechen.
- Konsistent bleiben: Verwende eine einheitliche Struktur und Terminologie, um die Verständlichkeit zu erhöhen.
- Zusammenarbeit fördern: Arbeite eng mit dem Product Owner, Entwicklern und Testern zusammen, um sicherzustellen, dass die Akzeptanzkriterien vollständig und realistisch sind.
- Iterativ verbessern: Überarbeite und verfeinere die Akzeptanzkriterien kontinuierlich basierend auf Feedback und neuen Erkenntnissen.

Generelle Fragestellungen zur Formulierung von Akzeptanzkriterien:

1. Was ist das Ziel der Anforderung?
- Was soll durch die Implementierung dieser Anforderung erreicht werden?
- Welchen Nutzen bringt die Anforderung für den Endbenutzer oder das System?

2. Wie wird die Anforderung genutzt?
- Welche Schritte oder Prozesse sind notwendig, um die Anforderung zu nutzen?
- Gibt es spezifische Eingaben oder Aktionen, die erforderlich sind?

3. Welche Bedingungen müssen erfüllt sein?
- Unter welchen Bedingungen soll die Anforderung arbeiten?
- Gibt es spezifische Vorbedingungen oder Nachbedingungen?

4. Wie wird der Erfolg gemessen?
- Welche Kriterien müssen erfüllt sein, damit die Anforderung als erfolgreich betrachtet wird?
- Gibt es spezifische Metriken oder Indikatoren, die den Erfolg anzeigen?

5. Welche Fehlerfälle müssen berücksichtigt werden?
- Welche möglichen Fehlerszenarien gibt es?
- Wie soll das System auf diese Fehler reagieren?

6. Welche Einschränkungen oder Grenzen gibt es?
- Gibt es technische, rechtliche oder andere Einschränkungen, die berücksichtigt werden müssen?
- Welche Grenzen gibt es hinsichtlich der Leistung, Sicherheit oder Benutzerfreundlichkeit?

7. Welche Abhängigkeiten gibt es?
- Gibt es andere Funktionen oder Systeme, von denen diese Anforderung abhängt?
- Müssen bestimmte Voraussetzungen erfüllt sein, bevor die Anforderung genutzt werden kann?

8. Wie wird die Anforderung getestet?
- Welche Testfälle und Szenarien müssen abgedeckt werden, um die Akzeptanzkriterien zu überprüfen?
- Gibt es spezifische Testdaten oder Umgebungen, die benötigt werden?

9. Welche Performance-Anforderungen gibt es?
- Welche Leistungsanforderungen müssen erfüllt sein (z.B. Antwortzeiten, Durchsatz)?
- Gibt es spezifische Last- oder Stresstests, die durchgeführt werden müssen?

10. Welche Sicherheitsanforderungen gibt es?
- Welche Sicherheitsmaßnahmen müssen implementiert werden (z.B. Authentifizierung, Autorisierung, Verschlüsselung)?
- Gibt es spezifische Compliance-Anforderungen, die erfüllt werden müssen?

11. Welche Usability-Anforderungen gibt es?
- Welche Anforderungen gibt es hinsichtlich der Benutzerfreundlichkeit (z.B. einfache Navigation, Barrierefreiheit)?
- Gibt es spezifische Usability-Tests, die durchgeführt werden müssen?

12. Welche Wartungsanforderungen gibt es?
- Welche Anforderungen gibt es hinsichtlich der Wartbarkeit des Systems (z.B. Codequalität, Dokumentation)?
- Wie soll das System auf zukünftige Änderungen vorbereitet werden?

13. Welche Skalierbarkeitsanforderungen gibt es?
- Wie gut muss das System skalieren können, um zukünftige Lasten zu bewältigen?
- Gibt es spezifische Anforderungen an die horizontale oder vertikale Skalierbarkeit?

Beispiel für nicht-funktionale Akzeptanzkriterien:

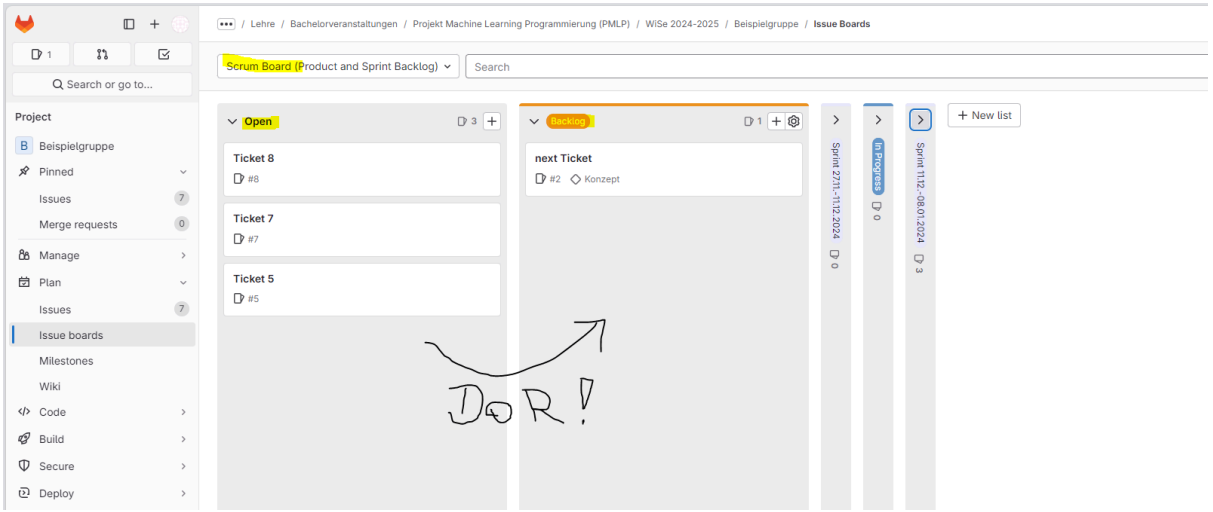
Anforderung: Das System muss eine hohe Verfügbarkeit gewährleisten.
Akzeptanzkriterien:
1. Das System muss eine Verfügbarkeit von 99,9% im Jahresdurchschnitt erreichen.
2. Geplante Wartungsfenster dürfen nicht länger als 2 Stunden pro Monat dauern.
3. Das System muss innerhalb von 5 Minuten nach einem Ausfall wiederhergestellt werden können.
4. Es müssen regelmäßige Backups durchgeführt werden, die eine Wiederherstellung der Daten innerhalb von 30 Minuten ermöglichen.
5. Sicherheitsupdates müssen innerhalb von 24 Stunden nach Veröffentlichung eingespielt werden.

Beispiel für Akzeptanzkriterien bei funktionalen Anforderungen:

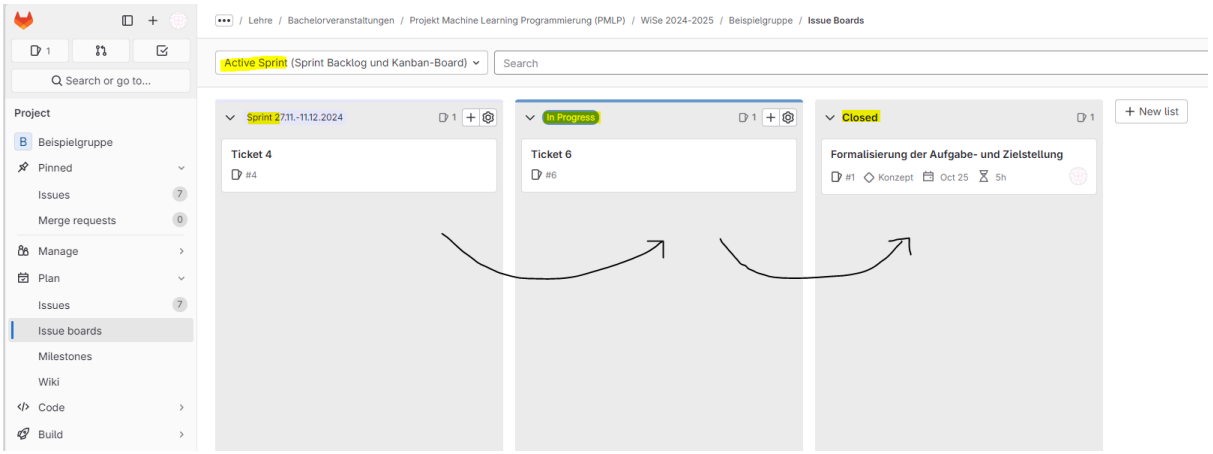
User Story: Als Benutzer möchte ich mich in das System einloggen können, um auf meine persönlichen Daten zuzugreifen.
Akzeptanzkriterien:
1. Der Benutzer muss ein gültiges Benutzername-Passwort-Paar eingeben, um sich erfolgreich einzuloggen.
2. Bei erfolgreichem Login wird der Benutzer auf die Startseite weitergeleitet.
3. Bei falschen Anmeldedaten wird eine Fehlermeldung angezeigt, die den Benutzer darauf hinweist, dass die Anmeldedaten ungültig sind.
4. Das System muss nach drei fehlgeschlagenen Login-Versuchen den Benutzer für 5 Minuten sperren.
5. Die Login-Seite muss den Anforderungen an Barrierefreiheit entsprechen (z.B. durch Unterstützung von Screenreadern).

Scrum-Prozess

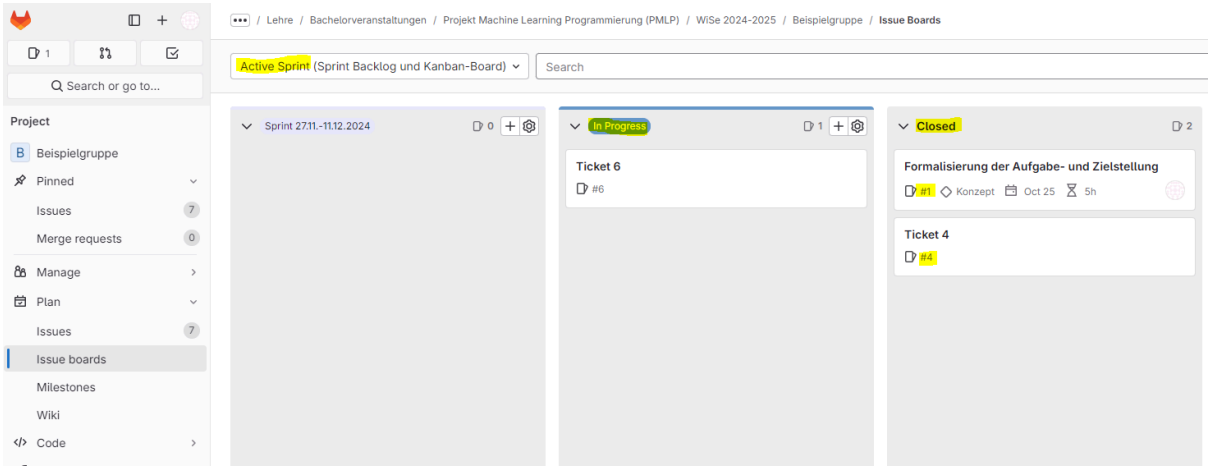
Refinement:



Während des Sprints:



Review:



Planning: