

Softwareprojekt Wintersemester 2019/2020

am Fachgebiet Software Engineering, Leibniz Universität Hannover

Anforderungsspezifikation

GameDev1

SWP-WS1920-GameDev-Spec-v01.docx

Vorgelegt

am 04.11.19
von Team GameDev

Ausführende:

<i>Nachname</i>	<i>Vorname</i>	<i>Rolle</i>
<i>Andrae</i>	<i>Dominik</i>	<i>Projektleiter</i>
<i>Holze</i>	<i>Christian</i>	<i>Qualitätsbeauftragter</i>
<i>Curth</i>	<i>Leon</i>	<i>Entwickler</i>
<i>Niehus</i>	<i>Lukas</i>	<i>Entwickler</i>
<i>Gaire</i>	<i>Amrit</i>	<i>Entwickler</i>
<i>Allerman</i>	<i>Lukas</i>	<i>Entwickler</i>
<i>Hollmann</i>	<i>Tim</i>	<i>Entwickler</i>
<i>Volodarskis</i>	<i>Felix</i>	<i>Entwickler</i>

Das Dokument enthält <Zutreffendes bitte ankreuzen>

- ☒ Die Anforderungen aus Kundensicht (User Requirements)
- ☒ Anforderungen, wie das zu System zu gestalten ist (System Requirements)

Datum, Unterschrift des Projektleiters, auch für die anderen Projektangehörigen

Kunden-Bewertung

Der Kunde, Prof. Dr. Kurt Schneider, bestätigt mit seiner Unterschrift, diese Anforderungsspezifikation erhalten, geprüft und für inhaltlich ☐ in Ordnung | ☐ weitgehend in Ordnung | ☐ deutlich zu verbessernd | ☐ nicht akzeptabel befunden zu haben.

Datum, Unterschrift des Kunden; evtl. Vermerk.

Inhaltsverzeichnis

1	Mission des Projekts	3
1.1	Erläuterung des zu lösenden Problems	3
1.2	Wünsche und Prioritäten des Kunden	3
1.3	Domänenbeschreibung	3
1.4	Maßnahmen zur Anforderungsanalyse	3
2	Rahmenbedingungen und Umfeld	4
2.1	Einschränkungen und Vorgaben	4
2.2	Anwender	4
2.3	Schnittstellen und angrenzende Systeme	4
3	Funktionale Anforderungen	4
3.1	Use Case-Diagramm	4
3.2	Use Case-Beschreibungen	4
	<Use Case 1: Gesamtsystem starten>	5
4	Qualitätsanforderungen	6
4.1	Qualitätsziele des Projekts	6
4.2	Qualitäts-Prioritäten des Kunden	6
4.3	Wie Qualitätsziele erreicht werden sollen	6
5	Hinweise zur Umsetzung	7
6	Probleme und Risiken	7
7	Optionen zur Aufwandsreduktion	7
7.1	Mögliche Abstriche	7
7.2	Inkrementelle Arbeit	7
8	Glossar	7
9	Abnahme-Testfälle	8



1 Mission des Projekts

1.1 Erläuterung des zu lösenden Problems

Das Ziel des Projekts GameDev ist es, ein Programm zu entwickeln, das durch Vergabe von Achievements und Punkten über Kollektoren wie Git oder Jira Arbeitsschritte belohnt. Dies soll Mitarbeiter*innen mehr Motivation für die Arbeit geben, da ihre Bemühungen stets durch Achievements und Punkte Anerkennung bekommen und sie sich im Highscore mit anderen Mitarbeiter*innen messen können.

1.2 Wünsche und Prioritäten des Kunden

Als Kollektoren sollen Git und Jira auf jeden Fall angebunden werden, andere Kollektoren sollen allerdings auch später ohne besonderen Aufwand hinzugefügt werden können. Zusätzlich sollen über push- oder pull-Trigger die Änderungen regelmäßig erfasst und die Achievements aktualisiert werden.

Als Authentifizierungssystem soll LDAP genutzt werden. Es soll eine Webanwendung erstellt werden, auf welcher der Progress eingesehen werden kann.

1.3 Domänenbeschreibung

Das System läuft auf einem lokalen Server, der sich im selben Netzwerk befindet, wie alle Benutzer*innen. Das System sollte mind. 40 Nutzer*innen gleichzeitig verwalten können, die sowohl Git und Jira, als auch ähnliche Systeme verwenden. Besonders zu beachten ist, dass das System intuitiv benutzbar ist **oder** eine übersichtliche Anleitung zur Verfügung stellt.

1.4 Maßnahmen zur Anforderungsanalyse

Ein Website-Entwurf wurde als Mockup implementiert und vorgestellt. Das Mockup enthält alle wichtigen Funktionen zur Benutzung des Programms und zeigt exemplarisch, wie die Benutzeroberfläche aussehen könnte. Das Mockup war für den Kunden sehr zufriedenstellend und kann in dieser Art ohne große Veränderungen implementiert werden.

Zudem wurde eine Skizze zum Systemaufbau vorgestellt, die ebenfalls umgesetzt **werden kann.**

2 Rahmenbedingungen und Umfeld

2.1 Einschränkungen und Vorgaben

Die SW läuft auf einem lokalen Server und soll Entwicklerteams im Größenbereich von ca 8 bis 40 Entwickler*innen verwalten können. Es wurden keine genaueren Ressourcen bzw. Kapazitäten angegeben, aber es ist davon auszugehen, dass die zur Verfügung stehenden Ressourcen definitiv ausreichend für das Projekt sind. Da das System gegebenenfalls weiterentwickelt werden soll, muss darauf geachtet werden, dass man weitere Kollektoren und Achievements möglichst einfach hinzufügen kann. Um dies zu erreichen ist eine möglichst modulare Architektur wünschenswert. Für die Authentifizierung sollen die schon bestehenden Login-Daten der Entwickler genutzt werden.

2.2 Anwender

Die Anwender*innen sind SW-Entwickler*innen, dementsprechend kann man gute Kompetenz in der Bedienung von Computern und Webanwendungen voraussetzen, jedoch sollte natürlich trotzdem darauf geachtet werden, dass die Weboberfläche einfach und intuitiv bedienbar ist.

Die Initialisierung des Systems wird von einem Admin auf der Kommandozeile ausgeführt. Ein*e Administrator*in sollte über genügend Erfahrung mit der Kommandozeile verfügen, sodass eine Dokumentation zur Inbetriebnahme des Systems genügt, um das System zu initialisieren.

2.3 Schnittstellen und angrenzende Systeme

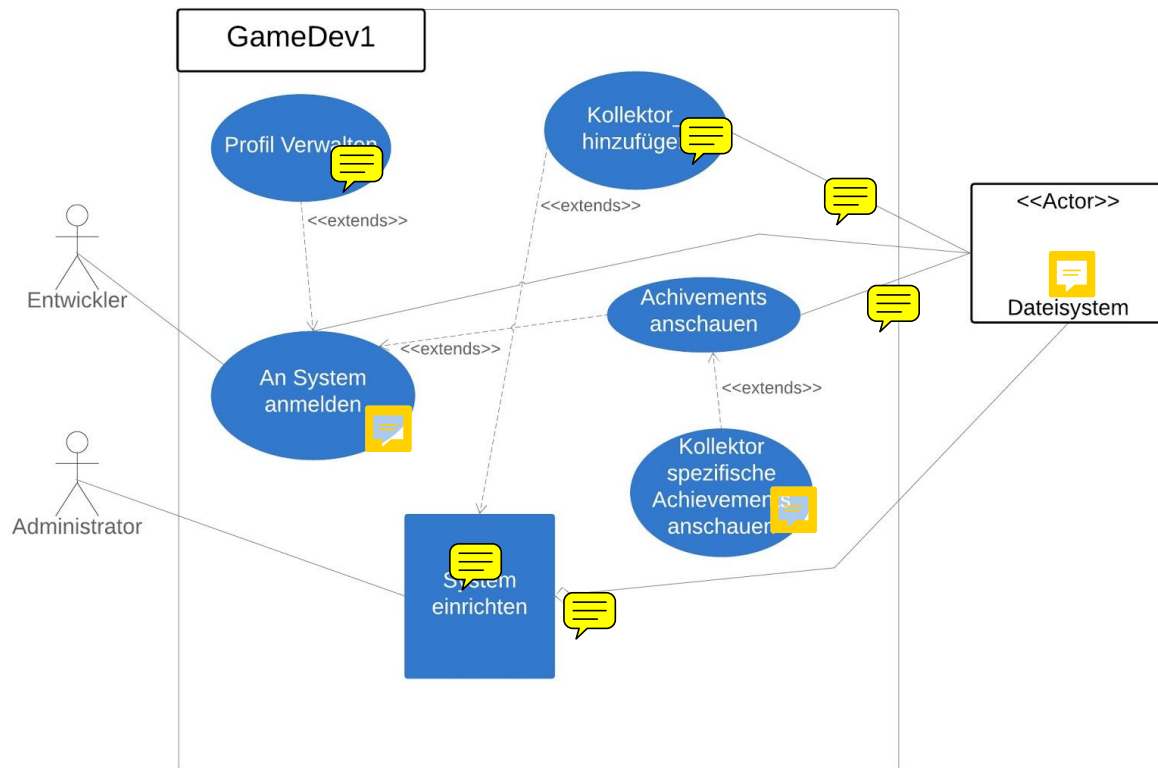
Jegliche Authentifizierungsdaten für einzelne Benutzer*innen sind über LDAP erreichbar. Desweiteren sind die Entwickler*innen in LDAP auch schon in die entsprechenden Entwicklergruppen eingeteilt.

Die SW soll Daten von Entwicklungstools wie GIT oder Jira auslesen können und gegebenenfalls um weitere Schnittstellen zu anderen Entwicklungstools erweiterbar sein.

Die SW benötigt einen Server mit Dockerengine, auf dem sie laufen kann.

3 Funktionale Anforderungen

3.1 Use Case-Diagramm



3.2 Use Case-Beschreibungen

<Use Case 1: Systemeinrichtung>

Use Case 1	System einrichten
Erläuterung	Der Administrator richtet das System auf dem Server ein.
Umfeld	Der Administrator startet die Software auf dem Server zum ersten Mal.
Systemgrenzen (Scope)	Der Server, auf dem die Software gestartet wird.
Ebene	Hauptaufgabe
Vorbedingung	Der Server läuft und unterstützt Docker-Container. Die Configdatei wurde ausgefüllt.
Mindestgarantie	Die Software teilt mit, ob sie erfolgreich eingerichtet wurde oder nicht.
Erfolgsgarantie	Die Software ist richtig eingerichtet und betriebsbereit.
Stakeholder	Administrator : möchte die Software für anderen developer bereitstellen. Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwicklern motivieren, um Produktivität zu steigern.
Hauptakteur	Administrator
Auslöser	Administrator startet die Applikation zum ersten Mal.
Hauptszenario	1. Administrator startet die Applikation zum ersten Mal. 2. Das System führt den internen Setup durch und informiert den User, ob alle Dienste wie erwartet laufen.
Erweiterungen	2b: WENN ein Fehler aufgetreten ist, DANN soll eine Fehlermeldung ausgegeben werden und weiter bei 1.
Priorität	unverzichtbar
Verwendungshäufigkeit	einmalig am Anfang pro Team

Use Case 2	Kollektor hinzufügen
Erläuterung	Der Administrator möchte einen Kollektor hinzufügen.
Umfeld	Der Administrator startet die Kollektor-Software zum ersten Mal.
Systemgrenzen (Scope)	Der Server, auf dem das Hauptsystem läuft.
Ebene	Nebenfunktion
Vorbedingung	Der Server läuft und unterstützt Docker-Container und die Hauptsoftware läuft.
Mindestgarantie	Die Software teilt mit, ob sie erfolgreich eingerichtet wurde oder nicht.
Erfolgsgarantie	Die Software ist richtig eingerichtet und betriebsbereit.
Stakeholder	Administrator : möchte die Software für anderen developer bereitstellen. Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwicklern motivieren, um Produktivität zu steigern.
Hauptakteur	Administrator
Auslöser	Administrator startet die Kollektor-Applikation zum ersten Mal.
Hauptszenario	1. Administrator startet die Kollektor-Applikation zum ersten Mal. 2. Das System führt den internen Setup durch und informiert den User, ob der Dienste wie erwartet laufen.
Erweiterungen	2b: WENN ein Fehler aufgetreten ist, DANN soll eine Fehlermeldung ausgegeben werden und weiter bei 1.
Priorität	hoch
Verwendungshäufigkeit	Einmal pro neues Subsystem

Use Case 3	Vollständige Nutzung der Weboberfläche
Erläuterung	Der Benutzer führt alle gängigen Verwendungszwecke nacheinander aus.
Umfeld	Weboberfläche.
Systemgrenzen (Scope)	Browser.
Ebene	Hauptfunktion
Vorbedingung	Der Server läuft und unterstützt Docker-Container und die Hauptsoftware läuft.
Mindestgarantie	Die Daten des Nutzers werden korrekt angezeigt oder es wird ein Fehler angezeigt
Erfolgsgarantie	Die Daten des Nutzers werden korrekt angezeigt
Stakeholder	Administrator : möchte die Software für anderen developer bereitstellen. Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwicklern motivieren, um Produktivität zu steigern.
Hauptakteur	Entwickler
Auslöser	Der Benutzer befindet sich auf der Website..
Hauptszenario	<ol style="list-style-type: none"> 1. Der Benutzer gibt seine Login-Daten ein und klickt auf "Login" 2. Das System prüft die Login-Daten und meldet den Benutzer an 3. Das System öffnet die Hauptseite 4. Der Benutzer klickt auf "Profil" (evtl. Benutzername) 5. Das System öffnet die Profil-Seite 6. Der Benutzer ändert seinen Namen 7. Das System speichert die Änderung und gibt eine Bestätigung zurück 8. Der Benutzer klickt auf "Home" (evtl. Haus-Item) 9. Das System öffnet die Hauptseite
Erweiterungen	2a. WENN die Login-Daten nicht validiert werden konnten, DANN kehrt das System zurück zu 1. 9a. WENN der Token abgelaufen ist, DANN kehrt das System zurück zu 1.
Priorität	unverzichtbar
Verwendungshäufigkeit	Regelmäßig

Use Case 4	Achievements anschauen
Erläuterung	Der User möchte die kollektor spezifische Achievements einsehen.
Umfeld	Der User ist in dem System eingeloggt und findet sich in Hauptseite.
Systemgrenzen (Scope)	Browser
Ebene	Nebenfunktion
Vorbedingung	Der Server läuft und unterstützt Docker-Container. Der User ist jetzt im System eingeloggt.
Mindestgarantie	Die webseite wird geladen.
Erfolgsgarantie	Der User kann seine Kollektor spezifische Achievements einsehen.
Stakeholder	Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können.
Hauptakteur	Developer/User
Auslöser	User klickt auf "Haus".
Hauptszenario	<ol style="list-style-type: none"> 1. User klickt auf "Haus". 2. Das System öffnet die Achievement-Seite. 3. User klickt auf die gewünschte Kollektor. 4. Das System öffnet die Achievements-Seite je nach der Wahl des Users.
Erweiterungen	
Priorität	unverzichtbar
Verwendungshäufigkeit	häufig

4 Qualitätsanforderungen

4.1 Qualitätsziele des Projekts

Die Programmierung selbst unterliegt einer klar formulierten **Code Convention**, an die sich jedes Team-Mitglied hält und die so Qualität des Codes selbst **garantiert**. Diese Convention umfasst u.a. klare Regeln bzgl. Kommentierung und Dokumentation des Codes, die einheitliche Benennung von Variablen und Methoden und die Regelung der verwendeten GIT-Repository.

Konkret sollen folgende **funktionale** Qualitätsmerkmale erzielt werden:

- System kann auf einem lokalen Server in Docker eingerichtet werden
- Kollektoren für neue Subsysteme können modular hinzugefügt werden
- User können ihren Progress auf einer Weboberfläche einsehen

Außerdem werden folgende **nicht-funktionale** Qualitätsmerkmale festgelegt:

- Die Weboberfläche soll einfach bedienbar sein
- Die Wartbarkeit des Systems soll gewährleistet sein

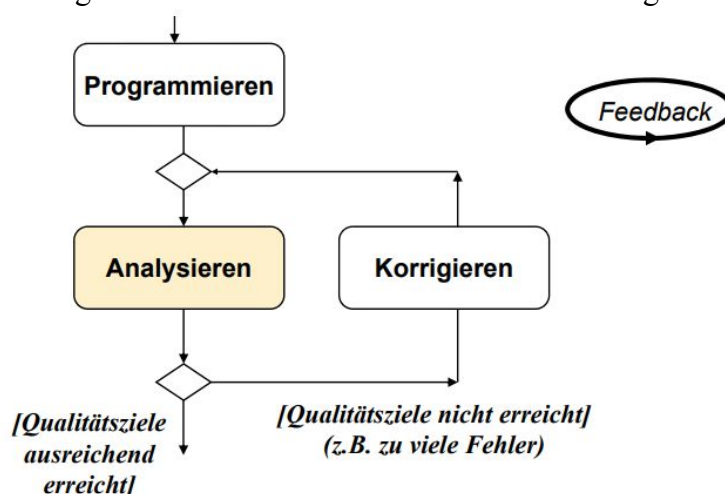
4.2 Qualitäts-Prioritäten des Kunden

Die Qualitätsziele sind wie folgt absteigend priorisiert:

1. System ist auf lokalem Server funktionsfähig
2. User können ihren Progress auf einer Weboberfläche einsehen
3. Kollektoren für neue Subsysteme können modular hinzugefügt werden
4. Die Weboberfläche soll einfach bedienbar sein
5. Die Wartbarkeit des Systems soll gewährleistet sein

4.3 Wie Qualitätsziele erreicht werden sollen

Die Qualitätsziele werden für jeden Teilschritt anhand von Analysen und Tests überprüft und ggf. korrigiert. Dieser iterative Arbeitsschritt ist in folgendem Diagramm abgebildet:



Bereits in den frühen Projektphasen werden Tests der einzelnen Module durchgeführt. Anhand der formulierten Qualitätsziele wird jeweils geprüft, ob diese erreicht wurden. Ist dies nicht der Fall, werden Fehler korrigiert, bis die gewünschten Testergebnisse erzielt werden.

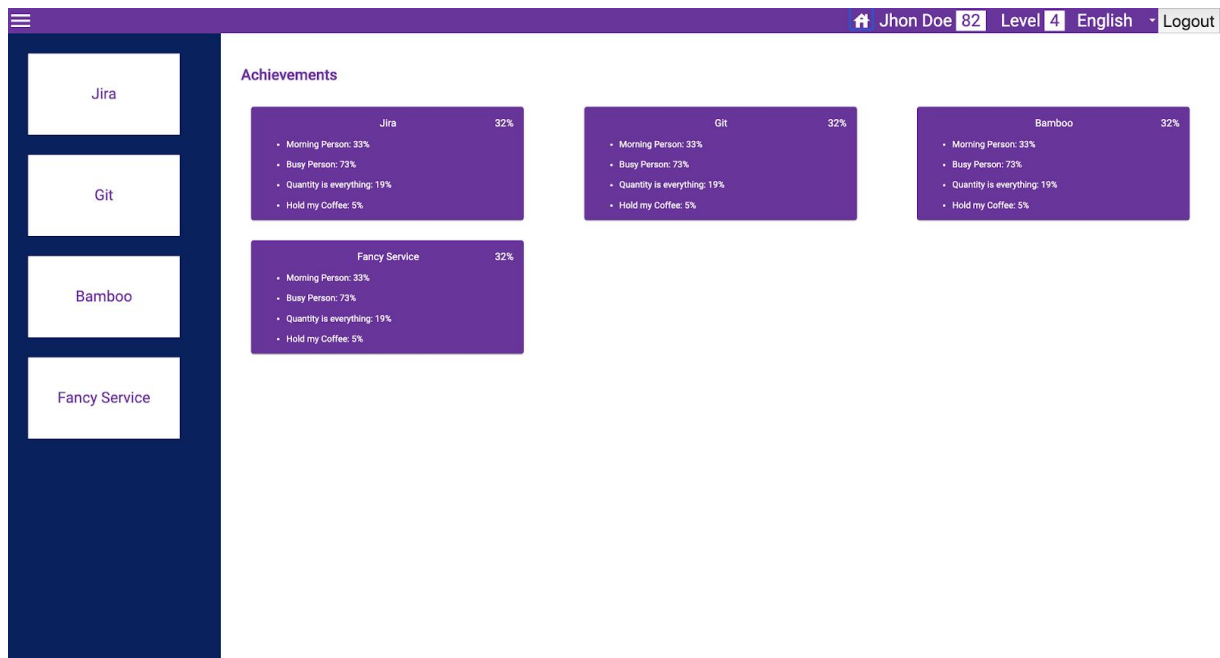
Neben den einzelnen Modultests werden auch in diesen Phasen die Integrationstests mit anderen bereits bestehenden Modulen bzw. der vorausgesetzten Systemumgebung durchgeführt. Vor Abschluss des Projekts findet außerdem ein Systemtest statt, der die Anwendung in ihrer Gesamtheit testet und die Erfüllung aller Anforderungen gewährleistet.

So stellen wir sicher, dass wir *Bottom-Up* bereits im Kleinen testen, ob einzelne Module wie gewünscht funktionieren, ob sie im nächsten Schritt miteinander korrekt interagieren und zuletzt im gesamten Systemkontext die Anforderungen erfüllen. Auf diese Weise können wir bereits im Entwicklungsprozess frühzeitig auf mögliche Fehler reagieren und ggf. korrigieren.

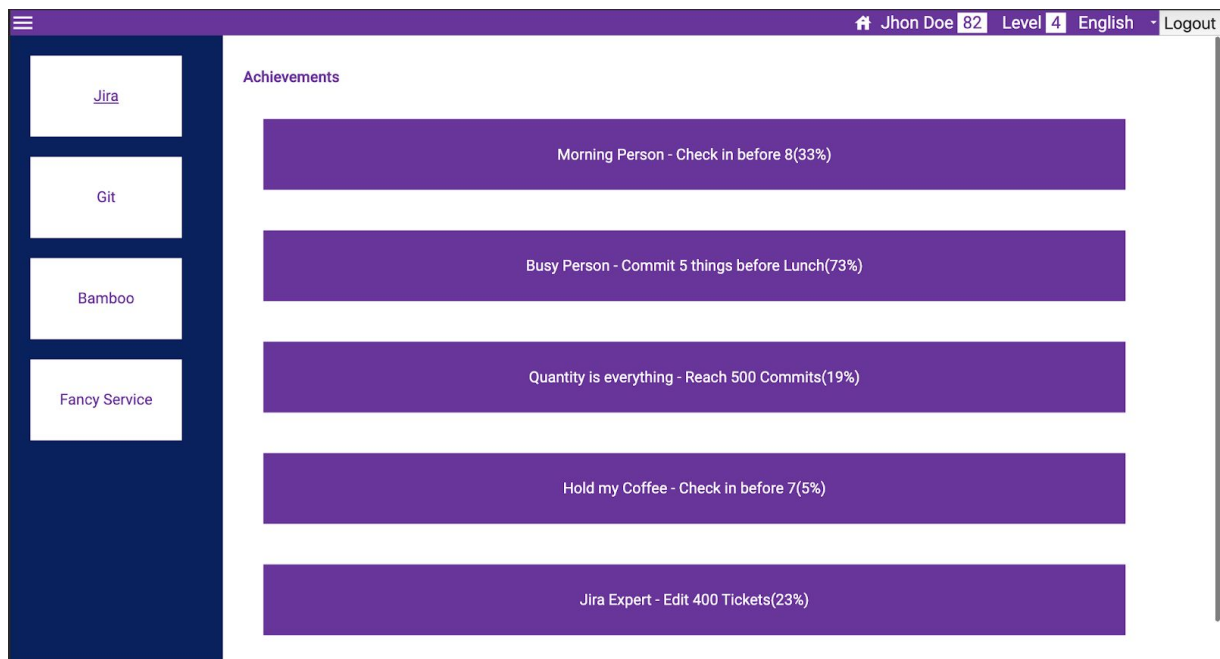
Testfälle beinhalten neben der eigentlich Funktionalität des jeweiligen Moduls mindestens die in dieser Spezifikation dokumentierten Use Cases und Abnahmetestfälle.

Darüber hinaus wird innerhalb des Teams anhand von klaren Aufgabenverteilungen und Dokumentation der einzelnen Aufgaben sichergestellt, dass Prozesse übersichtlich und nachvollziehbar bleiben. Dazu kommen Systeme wie Jira und Git zum Einsatz.

5 Hinweise zur Umsetzung

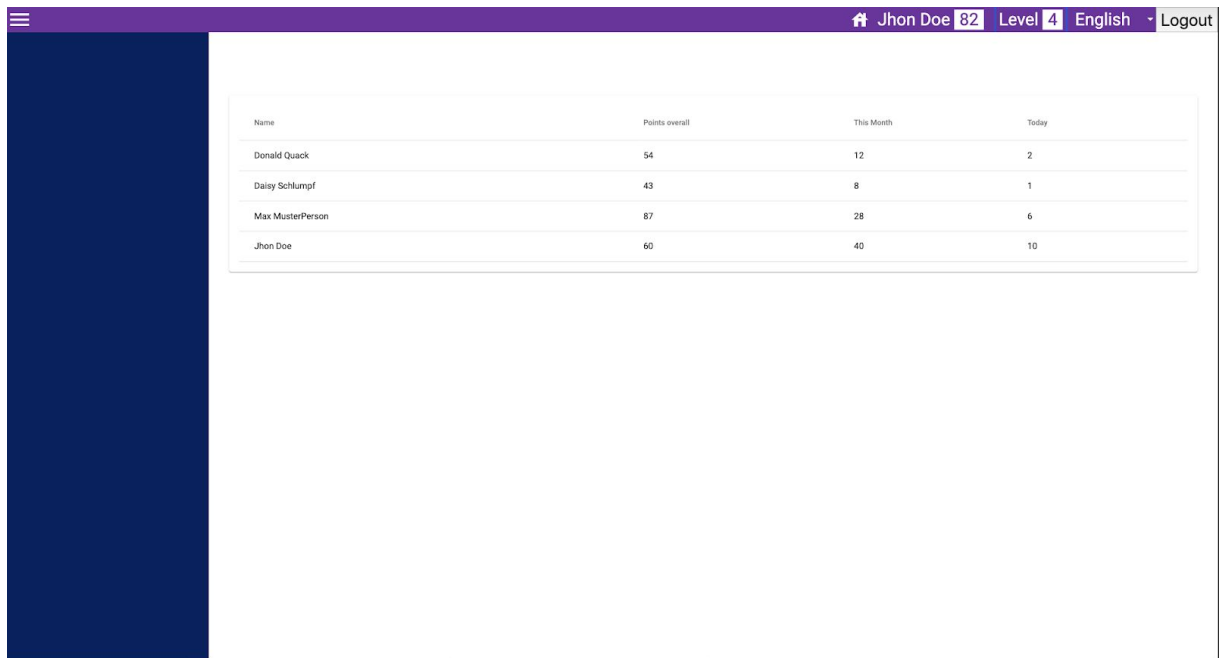


Auf der Webseite sieht man eine Übersicht aller Achievements für alle installierten Kollektoren. Außerdem sieht man in der oberen Zeile seinen Namen, die bereits erreichte Punktzahl und sein aktuelles Level. Ein Klick auf einen Kollektor führt zu einer erweiterten Ansicht.



In dieser erweiterten Ansicht werden alle Achievements für den ausgewählten Kollektor angezeigt. Zusätzliche Informationen wie ein Beschreibungstext können mit einem Klick auf das jeweilige Achievement ausgeklappt werden.

Mit einem Klick auf sein Level wird die Ranglistenübersicht angezeigt.



Name	Points overall	This Month	Today
Donald Quack	54	12	2
Daisy Schlumpf	43	8	1
Max MusterPerson	87	28	6
Jhon Doe	60	40	10

In der Ranglistenübersicht werden alle Teilnehmer des Projekts in einer Rangliste dargestellt.

6 Probleme und Risiken

WENN die Daten von den Kollektoren falsch oder nicht ausgewertet werden, DANN können Achievements nicht sinngemäß generiert werden.

Abhilfe: Sinnvolle Fehlermeldung mit Angabe zur Fehlerquelle (falsche Credentials o.ä.) oder Behandlung von Exceptions.

WENN keine Achievements sinnvoll generiert werden können, DANN kann die Webseite nicht sinnvoll ausgefüllt werden.

Abhilfe: Sinnvolle Fehlermeldung mit Angabe zur Fehlerquelle (falsche Credentials o.ä.) oder Behandlung von Exceptions.

WENN die Authentifizierung nicht richtig überprüft werden kann, DANN kann kein Zugriff auf die Webseite erfolgen.

Abhilfe: Notfalls eigenes Authentifizierungssystem implementieren.

WENN die Daten eines Entwicklers nicht identifizierbar sind, DANN können die Achievements nicht sinnvoll zugewiesen werden.

Abhilfe: Daten zu einem Entwickler so ändern, dass diese identifizierbar sind (z.B. durch E-Mail) und die Achievements ausgeben, so dass diese trotzdem gesehen werden.

7 Optionen zur Aufwandsreduktion

7.1 Mögliche Abstriche

Komplexe Datenauswertung und die Anzahl der Achievements insgesamt können reduziert werden.

Die Webseite kann weniger komplexe Views beinhalten.

Es können weniger Kollektoren implementiert werden, da diese auch später hinzugefügt werden können.

Wir können die monatliche und wöchentliche “Challenges” und Ranglisten weglassen und nur gesamt Punkte zählen.

Achievement-Details können minimiert werden.

7.2 Inkrementelle Arbeit

Am wichtigsten ist die generelle Struktur, die es ermöglicht Kollektoren modular hinzuzufügen. Also muss zunächst ein Interface für besagte Elemente erstellt werden und ein Backend/Database/Website, die mit dem Interface schon alles anzeigen und speichern kann.

Erst später werden einige Module auch von uns implementiert.

Danach können dann bessere Auswertungen und Views entwickelt werden, die Daten attraktiver darstellen.

8 Glossar

Kollektor Modulare **Komponent** die Daten von einem **Speziellen Dienst** (zb Git) abfragt, verwertet und in Achievements umwandelt und dann an den Hauptserver sendet.

Komponente Java-Klasse, mit der sich das Resultat eines Teams starten lässt.
Muss Instanz einer Unterklasse der hier erstellten Klasse
Component sein.

8.1 Begriffserklärungen

SW: Software

9 Abnahme-Testfälle

Login



Setup: Es existiert in LDAP user mit email "abc@gmx.de" und passwort "abc",

Eingabe	Soll
User gibt bei e-mail "abc@gmx.de" und bei passwort "abc" ein	Der User wird auf die Achievementsseite vom Benutzer abc weitergeleitet.
User klickt auf seinen Namen "abc"	Profilseite von "abc" wird angezeigt
Der User ändert seinen Namen von "abc" zu "overachiever"	In der Rangliste taucht sein Name jetzt als "overachiever" auf

Setup: Es existiert kein user mit E-mail "abcd@gmx.de" und passwort "abcd"

Eingabe	Soll
User gibt email "abcd@gmx.de" und passwort "abcd" auf der Loginseite der Webseite ein	Der User kriegt eine Fehlermeldung angezeigt und bleibt auf der Loginseite.

Setup: Es existiert in LDAP user mit email "abc@gmx.de" und passwort "abc"

Eingabe	Soll
User gibt seine E-Mail "abc@gmx.de" und passwort "passwort" ein	Der User kriegt eine Fehlermeldung und bleibt auf der Loginseite

Achievements anschauen

Setup:

User ist im System eingeloggt und befindet sich auf der Webseite. Er will seine Achievements von Git anschauen.

Eingabe	Soll
User K lickt auf das Haus	Die Achievements s eite wird angezeigt
User klickt auf "Git"	Die Git-Achievements s tatistik von dem

	User werden angezeigt.
--	------------------------

Kollektor hinzufügen

Setup:
Das Hauptsystem läuft.

Eingabe	Soll
Der Administrator startet einen Kollektor auf dem Server.	Der Dienst taucht auf der Webseite unter Achievements auf.