Review von Feature 2 der Gruppe 2

Reviewer: Andreas Heckl (Gruppe 3)

Januar 2020

Vorbemerkung: Zunächst möchte ich positiv hervorheben, dass der Support der Gruppe 2 sehr gut war. Leider gab es zunächst beim sogenannten Generator bei Schritt 6 Probleme. Diese konnten allerdings mit ein paar Hinweisen schnell umgangen werden und alles hat im Anschluss gut funktioniert. Sogar ein Scaffold mit bereits generierten Daten für alle Features wurde bereitgestellt. Es ist mir wichtig dies zu betonen, da ich es mir auch selbst wünschen würde, dass die uns reviewende Gruppe auf unseren Support eingeht, falls es Probleme mit unserem eigenen Feature geben sollte.

1 Review zur Systemanalyse im Tabellenformat

Zum Zeitpunkt der Abgabe lag noch keine Ausarbeitung in Fließtextform vor, deshalb wird hier nun das tabellarische Schema reviewt.

1.1 High Level Review

Ich finde die Systembeschreibung generell gut gelungen. Die Biases und Argumente sind meist nachvollziehbar und gut durchdacht. Auch der Verification Teil ist sinnvoll erläutert.

Genereller Kritikpunkt: Eine Aufgabe hat ja in der Praxis meist nicht nur einen Schwierigkeitsgrad, sondern auch eine Priorität. Ihr solltet vielleicht in euren Ausführungen klar machen, ob es in eurer Firma auch für jede Aufgabe eine Priorität gibt oder nicht. An vereinzelten Stellen ist nämlich die Rede davon, aber bei manchen Argumenten hat man den Eindruck, dass eine Aufgabe als einziges Attribut einen Schwierigkeitsgrad hat und eben keine Priorität. Falls es nämlich auch eine Priorität gibt, spielt das meiner Meinung nach eine wichtige Rolle für die Pro/Contra Argumente. Vielleicht ist sogar eine Umgestaltung des Features sinnvoll, insofern dass es anhand einer Gewichtung von Schwierigkeit und Priorität die Warnungen vergibt. Details dazu unter 1.2.

1.2 Detaillierte Review

Im Preexisting Bias der ersten Fragestellung ist von Prioritäten die Rede, welche schon vorgeschrieben sind. Sind hier wirklich Prioritäten gemeint oder eher die Schwierigkeitsgrade? Dies kommt für mich nicht klar rüber. Generell solltet ihr eventuell auf die Prioritäten in den Biases stärken eingehen (oder alternativ im Universum klarstellen, dass es gar keine Prioritäten gibt, um die ganze Sache zu vereinfachen), da ja nicht nur schwierige Aufgaben hohe Priorität haben, und daher die Erledigung leichter Aufgaben in Summe vielleicht sogar mehr Wert für das Unternehmen schafft, was ja letztendlich der Maßstab sein sollte.

Die Ausarbeitung der Biases der zweiten Fragestellung sowie die vortheoretische Deliberation finde ich sehr gut gelungen. Hier fällt mir intuitiv nichts mehr ein, was zwingend ergänzt werden sollte.

Das zweite deontologische pro Argument verstehe ich nicht ganz. Ich lese heraus, dass durch die Einführung des Features die Auftraggeber gleich behandelt werden. Das heißt im Umkehrschluss, dass die Auftraggeber unter Umständen aktuell nicht gleich behandelt werden. Allerdings handelt es sich bei der Aufteilung der Aufgaben ja um rein interne Prozesse, die Resultate sind ja die gleichen, lediglich die Aufteilung unter

den zuständigen Mitarbeitern ändert sich. Die restlichen Argumente finde ich einfach nachvollziehbar und ausführlich genug beschrieben.

Die technische Umsetzbarkeit ist nachvollziehbar und gut durchdacht, jedoch frage ich mich, warum man sich dazu entschieden hat, sich selbst ein System von Schwierigkeitsbewertungen (1 bis 10) auszudenken. Naheliegend wäre es meiner Meinung nach, einfach Jira Issues zu verwenden, und dort zum Beispiel die Storypoints als Maßstab zu nehmen. Sicherlich ist ein eigens System auch nicht verkehrt, aber man hätte in einem Satz noch begründen können, warum man sich gegen die Verwendung von Jira Daten aus einem Open Source Projekt ausspricht.

Die Verification Passage deckt die wesentlichen Testfälle ab und dem ist daher aus meiner Sicht nichts hinzuzufügen.

2 Review der technischen Umsetzung

2.1 Review des in Jira sichtbaren Dashboards

Das User Interface des Dashboards könnte noch mit Informationen angereichert werden, z.B. auf welchen Zeitraum sich die Auswertung bezieht. Denn es wäre ja möglich, dass ein Mitarbeiter in den letzten 4 Wochen zwar überwiegend leichte Aufgaben bearbeitet hat, aber in den Wochen zuvor überwiegend schwierige.

Eine weitere Idee, um das Interface etwas spannender und aussagekräftiger zu gestalten, wäre z.B. einen Vergleich zum durchschnittlichen Firmenschwierigkeitsgrad pro Mitarbeiter darzustellen. Liegt z.B der Durchschnitt aller Tasks der letzten X Wochen bei einem Schwierigkeitsgrad von 0.6, wäre vorstellbar, dass alle Mitarbeiter eine Warnung erhalten, die darunter liegen. Wenn man diesen Firmenvergleich gegenüberstellt, hat das Dashboard noch mehr Aussagekraft. Sogar ein Ranking der Mitarbeiter wäre dann vorstellbar, natürlich muss allerdings aus ethischer Sicht geprüft werden, ob es als zulässig erachtet werden kann.

2.2 Review der Source Codes

2.2.1 Review des gadget.xml files

Hierzu gibt es aufgrund der klein gehaltenen Anzahl an HTML Codezeilen nichts anzumerken

2.2.2 Review des dashboard.js files

Kommentierung im Code Die Kommentare im Source Code finde ich insofern gelungen, dass sie zum Verständnis des Codes beitragen und nicht zu aufgebläht sind. Zudem ist positiv anzumerken, dass alle Kommentare durchgängig englisch sind und kein Sprachmix mit deutsch fabriziert wurde. Es wurde sich anscheinend bewusst gegen eine Kommentierung mit @param und @return etc. entschieden, was aufgrund des geringen Umfangs und der klaren Struktur auch nachvollziehbar ist.

Variablennamen Die Benennung der Variablen ist größtenteils so gelungen, dass sofort der Zweck der jeweiligen Variable ersichtlich ist.Auch hier wurde positiverweise alles auf englisch benannt.

Struktur Die Struktur und Semantik des Codes ist leicht verständlich, da der Umfang an Zeilen auch nicht allzu hoch ist. Als einziger Kritikpunkt ist folgendes zu nennen: der Ablauf der Funktionsaufrufe in der Ausführung des Codes ist derzeit folgende:

1. getAllUsers
2. getAllIssues
3. getCurrentUsers
4. populate

Diese Struktur könnte man meiner Meinung nach in eine main Funktion packen und dann dort den return value der oben genannten Funktionen jeweils in einer Variable speichern, und diese dann als Parameter an populate übergeben. Somit würde jede Funktion nur genau das tun, was ihr Name auch sagt. Beispielsweise würde getAllUsers dann auch tatsächlich nur die User returnen. Aktuell ist es im Code jedoch so, dass nur getAllusers aufgerufen wird, und dann anschließend diese Funktion getAllIssues aufruft und diese wieder getCurrentUsers usw. Dies erwarte ich intuitiv aber gar nicht von einer "getter" Funktion. Deshalb fände ich eine main Funktion besser, in der vier explizite Funktionsaufrufe sind.