Auftragnehmer:
Projektgruppe 06 “Green Twelve Studios“
Ansprechpartnerin: Mirja Kühn
E-Mail: mkuehn@mail.upb.de

Abschlussdokument

Wesentliche Ideen des Teams:

Im Folgenden werden wir die wesentlichen Ideen unseres Teams im Bezug auf die Entwicklung der geforderten Dokumente und der Software eingehen.

Da eine koordinierte Zusammenarbeit, bei der alle ihren Teil zum Ergebnis beitragen, mit 12 Leuten herausfordernd ist, haben wir uns von Anfang an dazu entschieden, die Aufgaben - egal ob Dokument oder Software - in kleinere Teilaufgaben aufzuteilen. Die Arbeitspakete wurden Kleingruppen zugeteilt, welche diese bearbeiten sollten. So haben wir sichergestellt, dass unsere Arbeit effizient war, da wir jedes Mitglied in unsere Gruppe einbinden konnten. Außerdem wurde dadurch an den einzelnen Teilen der Dokumente/Softwarekomponenten parallel gearbeitet. Zusätzlich haben die Kleingruppen ihre Aufgaben untereinander auf Qualität geprüft und Verbesserungsvorschläge gegeben. Bei der Erstellung der Softwarekomponenten wurde zudem für jede Komponente ein "Beauftragter" festgelegt, der die Erstellung der jeweiligen Komponente koordinierte, kontrollierte und gleichzeitig als Ansprechpartner für Fragen innerhalb unserer Gruppe diente.

Eine weitere wesentliche Idee unseres Teams war es, durch häufige Treffen mit der gesamten Gruppe einen ständigen Fortschritt sicherzustellen. Dies setzten wir um, indem wir uns mit der gesamten Projektgruppe zweimal pro Woche trafen, um uns gegenseitig von unserem Fortschritt zu berichten und die weiteren Aufgaben zu koordinieren. Zudem kamen wir zu jeder wichtigen Deadline zusammen, um gemeinsam unsere Abgabe fertigzustellen und zu kontrollieren. Weiterhin trafen wir uns innerhalb der Kleingruppen zusätzlich oft ein- bis zweimal pro Woche.

Während des Entwicklungsprozesses nutzten wir innerhalb unserer Entwicklungsteams entweder "pair programming" oder ein Mitglied des Teams erstellte den Code alleine und ein anderer Entwickler führte eine "Code Review" durch. Beim Erstellen des Codes achteten wir meistens darauf, dass wir für neu hinzugefügte Funktionalität jeweils eine eigene "feature-branch" angelegt wurde. Diese wurde dann, wenn die Funktionalität vollständig implementiert war, auf die "develop-branch" gemergt. Dies geschah vor allem, um einen besseren Überblick über unseren Fortschritt zu erlangen, die Erstellung der Komponente besser zu koordinieren und um die parallele Weiterentwicklung zu ermöglichen.

Während der Entwicklungsphase hat unser Team vor allem die Wartbarkeit unsere Codes besonders berücksichtigt. Dies war uns wichtig, da wir unsere Software möglichst einfach gestalten wollten, damit sie für jedes Mitglied unseres Teams nachvollziehbar ist. Außerdem war es uns wichtig, dass die Software modular ist. Hierzu haben wir schon im QA-Dokument "Code Conventions" festgelegt. Es war uns besonders wichtig, dass diese während der gesamten Entwicklung unsere Codes eingehalten werden. Zudem wurde darauf geachtet, dass das Ausführen automatischer Tests im großen Umfang möglich sein sollte und auch durchgeführt wurde. Um die Modularität unseres Codes sicherzustellen, teilten wir die von Ihnen geforderten Komponenten in einzelne kleine Bausteine auf. Außerdem haben wir uns weitestgehend an die sogenannte Drei-Schichten-Architektur gehalten.

Zunächst war das Ausmaß des Projektes noch nicht gut überschaubar, weshalb zu Beginn die Tests immer nach Implementierung eines neuen Features grob die wichtigsten Funktionalitäten des Codes abdeckten. Dies wurde später dann durch Einbindung des Tools "Jacoco" verändert, mit dem sich leicht die Testabdeckung aller einzelnen Klassen anzeigen ließ. Dies half dem Test-Team, eine hohe Abdeckung durch Tests zu erreichen und dafür zu sorgen, dass bei Änderungen in jedem Fall mögliche Probleme auffielen.

Abweichung von der Definition of Done:

Von der im QA-Dokument beschriebenen Definition of Done haben wir folgende Abweichungen gemacht:
Bei den User Stories wurden neue Anforderungen oder Abweichungen nur bedingt hinzugefügt.
Die Code Conventions wurden stets eingehalten und durch Code Reviews überprüft. Bezüglich der Dokumentation wurden alle Komponenten anhand von UML-Diagrammen implementiert. Änderungen die während der Implementierung aufgefallen sind, wurden ergänzt.
Die Tests wurden zum größten Teil alle ausgeführt, jedoch zum Teil verspätet was dazu geführt hat, dass Fehler bei der Implementierung anderer Features entdeckt wurden.

Das Arbeiten mit Branches auf dem Git ist anfangs nicht genau definiert worden und deswegen wurde der "master-branch" nicht wie geplant eingesetzt. Wir haben die meisten "feature-branches" nach der Fertigstellung auf den "develop-branch" gemergt und nicht auf den "master-branch".

Zusammenfassung Rückblick:
Um die Meinungen aller Gruppenmitglieder mit einzubeziehen haben wir eine Umfrage gemacht und jeweils alle Antworten zu den folgenden Fragen zusammen gefasst.

1. Was lief gut im Praktikum?

Die meisten Mitglieder der Gruppe, fanden es gut, dass wir immer eine hohe Qualität der Abgaben erreichen konnten. Außerdem wurde oft positiv angemerkt, dass wir uns häufig getroffen haben und, dass Fragen und Fehler immer schnell gelöst/behoben wurden.
Zudem wurde erwähnt, dass die Zusammenarbeit im Team gut funktionierte und die zugeteilten Aufgaben von jedem Teammitglied zur festgelegten Deadline bearbeitet wurden.

2. Was hast du darüber gelernt im Team zu arbeiten?

Bei dieser Frage wurde am häufigsten festgestellt, dass genaue Kommunikation im Team sehr wichtig ist, sowie, dass eine feste Aufgabenverteilung und feste Ziele für die einzelnen Aufgaben gemacht werden sollte.

3. Was hast du für Arbeitstechniken gelernt ?

Für die meisten Mitglieder war das Arbeiten mit dem SCRUM-Prozess eine neue Erfahrung. Weiterhin waren das Arbeiten mit GIT und Overleaf (Latex), sowie verschiedene Programmiertechniken (z.B. Pair Programming oder Code Reviews) für viele von uns ein neuer Vorgang.

4. Was würdest du beim nächsten Mal, bezogen auf die Arbeitsweise, anders machen?

Ein großer Teil der Mitglieder unserer Gruppe würde beim nächsten Mal insbesondere die Aufteilung der Aufgaben anders gestalten. So würden viele die Aufgaben in kleinere Arbeitspakete aufteilen, den einzelnen Teilaufgaben feste Personen zuteilen und genaue Deadlines für diese festlegen, sodass die Arbeitslast am Ende besser verteilt ist. Ansonsten würden wir beim nächsten Mal das Kanban-Board aktiver nutzen. Allerdings gab es auch einige Personen, die vollkommen zufrieden mit der Arbeitsweise im Projekt waren.

5. Was ist dir positiv an der SCRUM-Arbeitsweise aufgefallen?

Am SCRUM-Prozess ist uns am häufigsten positiv aufgefallen, dass wir uns häufig getroffen haben und so gut voran kamen, bzw. koordinieren konnten, wo noch mehr Zeit investiert werden muss. Desweiteren waren alle im Team über den Großteil der einzelnen Bereiche gut informiert und, falls es eine Frage gab, konnte man sich immer an die festgelegten Ansprechpartner für die einzelnen Bereiche wenden. Auch die Möglichkeit, relativ flexibel zu Arbeiten, ist uns positiv aufgefallen.

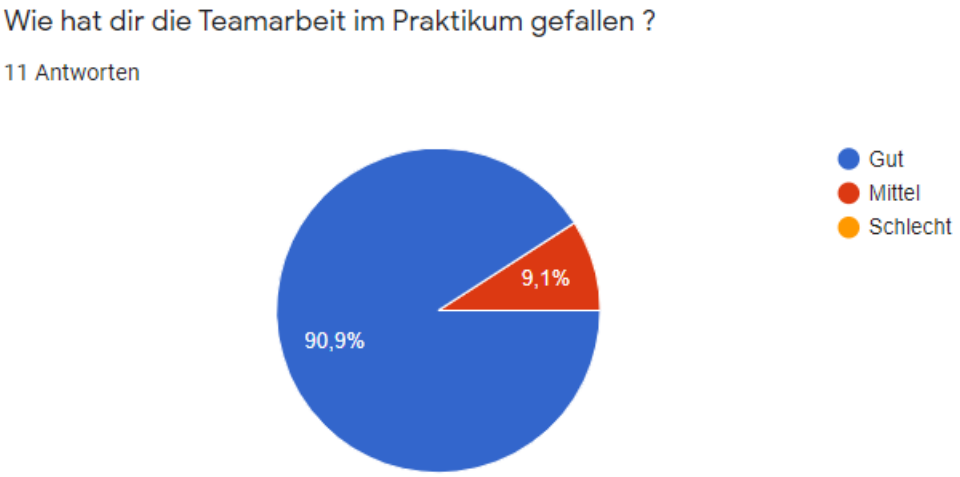
6. Was ist dir negativ an der SCRUM-Arbeitsweise aufgefallen?

Einige Mitglieder fanden die Arbeitsweise durchweg positiv. Allerdings wurde von vielen Mitgliedern bemängelt, dass die Arbeitslast relativ ungleich verteilt war und dass es ohne Erfahrung mit der SCRUM-Arbeitsweise schwer war, Rollen und Arbeitspakete richtig einzuteilen.

7. Was hast du aus dem Praktikum für die Zukunft mitgenommen?

Wir haben in diesem Praktikum gelernt, dass gute Organisation und Teamarbeit essentiell für den guten Verlauf eine Projektes sind. Zudem ist auch eine gute Absprache zwischen den einzelnen Teams im Projekt wichtig, um Missverständnisse zu vermeiden. Ein weiterer Punkt, den wir bemerkt haben, ist, dass das Testen des Codes extrem wichtig ist und dadurch viele Bugs vermieden werden können.

8. Umfragen zum Praktikum und der Teamarbeit



9. Anmerkungen zu den Wöchentlichen Treffen mit unserem Tutor, Lukas:

Alle Personen in unserer Gruppe fanden, dass Lukas ein sehr guter Tutor war, der uns bei Fragen immer weiterhelfen konnte und uns über das gesamte Praktikum hinweg gut betreut hat. Allerdings fiel uns auf, dass es teilweise etwas überflüssig war, die Treffen auf mindestens eine Stunde zu ziehen, da einige Absprachen in relativ kurzer Zeit erfolgen konnten und wir in der restlichen Zeit nicht immer wussten, was wir tun sollten (außer die Stunde abzuwarten).

10. Kritik zum Praktikum (allgemein):

Insgesamt ist uns am Praktikum negativ aufgefallen, dass die API-Spezifikation teilweise nicht optimal formuliert war und dass häufige Änderungen an dieser zu Verwirrung und zusätzlichem Aufwand führten. Zudem haben wir bemerkt, dass in manchen Fällen Wissen vorausgesetzt wurde, welches wir bisher im Studium noch nicht erlangt haben (zumindest mit dem vorhergesehenen Studienablaufplan für unsere Studiengänge), wie z.B. Java-Programmierung, Arbeiten mit GIT etc.

11. Verbesserungsvorschläge für das Praktikum (allgemein):

Als Verbesserungen für die nächste Durchführung dieses Moduls würden wir vorschlagen, dass sich das Praktikum an den vorangegangenen Modulen orientiert. So hätten wir es z.B. gut gefunden, wenn wir als Programmiersprache Python genutzt hätten, weil die meisten Mitglieder unserer Gruppe mehr Erfahrung mit Python, als mit Java hatten und bei der Benutzung von Python daher eine bessere Aufgabenverteilung möglich gewesen wäre. Außerdem hätten wir es begrüßt, wenn wir einen umfangreichen Einblick in die Bewertungskriterien und die Bewertungen unserer Abgaben gehabt hätten, um daraus noch mehr für die Zukunft lernen zu können.

Reflexion:

Abschließend können wir sagen, dass wir als Gruppe im Praktikum überwiegend positive Erfahrungen gemacht haben. Allerdings haben wir festgestellt, dass es noch einige Punkte gibt, welche wir besser umsetzen können:
Zum Einen würden wir die SCRUM-Arbeitsweise beim nächsten Mal anders anwenden. Insbesondere haben wir während des gesamten Projektes das Kanban-Board nur unregelmäßig gepflegt. Dies führte dazu, dass wir unseren Fortschritt recht schwer überblicken konnten. Im Nachhinein ist uns aufgefallen, dass eine gute Nutzung des Product Backlog uns eine strukturiertere Entwicklung ermöglicht hätte. Außerdem würden wir unsere SCRUM-Meetings besser planen, um diese effektiver Nutzen zu können und unser Projekt besser zu koordinieren. Weiterhin würden wir es in Betracht ziehen, eine kürzere Sprintlänge zu wählen, um die zeitliche Verteilung der Arbeitspakete besser planen zu können und einen besseren Überblick über die Menge der zu erfüllenden Aufgaben innerhalb eines Sprints zu haben.
Zum Anderen würden wir beim Testen von Anfang an eine ähnliche Vorgehensweise einhalten, wie wir sie am Ende des Projektes eingeführt haben, da uns aufgefallen ist, dass noch nicht getesteter Code in einer Komponente oft zum Hindernis bei der Entwicklung der anderen Komponenten wurde. Außerdem haben wir eine ausführliche Testdokumentation erst zu spät bedacht. Für den Überblick wäre es sinnvoll gewesen alle Tests und deren Ergebnisse zentral und einheitlich zu dokumentieren.
Auch die sinnvolle Nutzung der Branches auf unserem GIT ist uns anfangs schwer gefallen. Jedoch haben wir während des Projektes gelernt, diese besser zu nutzen.
Ein weiterer Punkt, welchen wir überschätzt haben, war die Zeiteinschätzung des gesamten Projektes. Das Projekt, für welches wir großzügig 2.800 Stunden geschätzt haben, haben wir mit den aktuellen 1.300 Stunden zum Ende des Projektes doch deutlich unterboten. Auch wenn diese Einschätzung mehr für uns war, so haben wir uns doch um ein großes Stück verschätzt. Dies zeigt, das man doch etwas Erfahrung braucht, um eine realistische Vorstellung davon zu bekommen, wie viel Zeit alle einzelnen Aspekte wirklich brauchen.