Srum Master:
– Marvin Glaser (4424114, marvin.glaser91@gmail.com)
Project Owner:
– Zoë Lange (5621688, zoe.lange@stud.uni-frankfurt.de)
Entwicklungsteam:
– Gian-Marco D'Agostino (6604769, gian-marco.dagostino@outlook.de)

zu Aufgabe 2.5

Sprint Review

Realzeit 16. Feb 21:05 - 22:16;
Simulierte Zeit 24. Mai 13:00 - 15:34

Dokumentation – Feedback der Vereinsvorstände:

zu ID6: Es wäre gut, wenn es für die Buchhaltung eine zentralisierte Seite gäbe, auf der der Status von regelmäßigen Zahlungen aller Mitglieder eingesehen werden könnte.
zu ID7: Außerdem sollten die Nutzer automatisch eine Mahnung als E-Mail bekommen, wenn sie eine Zahlung vergessen/nicht getätigt haben
zu ID9: Es wäre auch gut, wenn es eine Möglichkeit gäbe seine Bankdaten anzugeben, so dass das System dann automatisch alle ausstehenden Zahlungen einzieht
zu vorherigem Review: Die Anmerkungen, die ich letztes Mal gemacht habe wurden ja alle umgesetzt. Allerdings bin ich noch nicht zufrieden wie umständlich es ist eine Rückbuchung durchzuführen. Das sollte man doch mit einem Knopfdruck umsetzen können, oder?

Sprint Retrospektive

Realzeit 16. Feb. 22:17 - 22:55;
Simulierte Zeit 24. Mai 15:45 - 16:30

Was seit der letzten Retrospektive besser gelaufen ist:

- **Mehr Zeit für Tests:** Die extra Zeit fürs Testen war sehr sinnvoll. Viele Fehler konnten so gefunden und Konzepte besser überarbeitet werden, so dass die Software stabiler und ressourcensparender designed ist.

- **Mehr Zeit für Implementierungen:** Die anfänglichen Problematiken mit der

Implementierung wurden dadurch eingedämmt.

- **Aufgabenverteilung nach Erfahrungswerten der Programmierer richten:** Die Programmierer konnten ihrer Spezialisierung nach bessere und schnellere Erfolge beim Programmierung nachweisen, als dies zuvor der Fall gewesen ist. Da die Aufgabenverteilung nicht immer funktioniert, müssen auch Aufgaben erledigt werden, die nicht in die Spezialgebiete der entsprechenden Programmierer passen.

- **Ein Entwickler sollte an größeren Aufgaben(-paketen) arbeiten:** Die Programmierer konnten sich somit besser in ihre momentane Aufgabe hineinarbeiten und -denken, wodurch sich Konzepte verbesserten und Aufgaben auch konzentrierter erledigt wurden.

- **Dokumentation von Bugs:** Die Dokumentation war sehr hilfreich, da Bugs sehr schnell gefunden wurden und weitere Fehler durch die dokumentierte Erfahrung vorgebeugt wurden. Bugs, die in eine ähnliche Klasse fallen, sollten nicht mehr ausführlich dokumentiert werden, um Zeit zu sparen.

- **Pair-Programming:** Programmierer 1 und 2 arbeiten gut zusammen. Programmierer 1 und 3 konnten weniger gut zusammenarbeiten. Schlussendlich lässt sich sagen, dass Pair-Programming zur Vermeidung von Fehlern und Bugs führt, die Arbeitsgeschwindigkeit wird dadurch etwas gedrosselt, was sich aber im Endeffekt zu einer schnelleren Fertigstellung der Software und einer größeren Stabilität und Sicherheit führt. Zudem konnten die Programmierer voneinander lernen, wenn die Zusammenarbeit gut funktioniert hat (dies war wie gesagt, leider nicht bei jedem *pair* der Fall). Pair-Programming sollte nur noch für kompliziertere Aufgaben genutzt werden, da die Zeitersparnis bei kleineren Aufgaben nicht gegeben war.

- **Flexible Zeit für diverse Problematiken:** Die freiverfügbare Zeit konnte sehr gut genutzt werden und es war sehr hilfreich, die Zeit flexibler nutzen zu können. So konnte sich auf Problematiken konzentriert werden, ohne den aktuellen Zeitplan zu verletzen.

- **Versionierungssoftware:** Es wurde sich für Git entschieden, da mehr Erfahrung mit der Arbeit mit Git vorliegt. Es hat ein wenig Zeit gekostet alle hinreichend einzuarbeiten, aber es hat sich gelohnt, da nun weniger Verwirrung über aktuelle Versionenen im Enwicklerteam besteht und der Product Owner die Stake Holder auf dem Laufen halten kann ohne die Programmierer andauernd nach dem Stand fragen zu müssen.