



ClansCore

Weiterentwicklung

Discord-Bot für Vereine

Studien- und Bachelorarbeit im
Herbst-Semester 2025

Ausdruck:

10.10.2025

Team / Autoren:

Vanessa Alves und Joel Thoma

Referent:

Prof. Dr. Frieder Loch

Koreferent:

Julia Czerniak-Wilmes

Gegenleser:

Severin Dellsperger

Inhaltsverzeichnis

I Produkt Dokumentation	1
1 Einleitung	1
1.1 Problemstellung	1
1.2 Vorarbeiten	1
1.3 Ziel der Arbeit und Abgrenzung	2
1.4 Rahmenbedingungen	2
1.5 Stand der Technik	2
2 Erweitertes Gamification-Konzept	4
2.1 Interviews zur Motivation von Vereinsmitgliedern	4
2.1.1 Methodisches Vorgehen	4
2.1.2 Interviewleitfaden	5
2.1.3 Mehrwert gegenüber dem Vorprojekt	5
3 Anforderungen	6
3.1 Priorisierung der Anforderungen	6
3.2 Legende zum Erfüllungsgrad der Anforderungen	6
3.3 User Stories	6
3.3.1 User Story: Vorstandsmitglied	6
3.4 Functional Requirements	7
3.4.1 Discord-Bot	7
3.4.2 Admin-Dashboard	7
3.4.3 Neue Plattform	9
3.5 Use Cases	9
3.6 Non-Functional Requirements	14
4 Systemanalyse und Architektur	17
5 Qualitätssicherung	18
6 Implementation	19
7 Fazit	20
II Projekt Dokumentation	21
8 Projekt Plan	21
8.1 Zusammenarbeit	21
8.2 Meetings	21
8.3 Minimum Viable Product (MVP)	21
8.4 Long-Term Plan	22
8.4.1 Inception (Initiierung)	23
8.4.2 Elaboration (Ausarbeitung)	23
8.4.3 Construction (Konstruktion)	23
8.4.4 Transition (Übergang)	23
8.4.5 Presentation (Präsentation)	23
8.4.6 Meilensteine	23
8.5 Risikomanagement	24
8.5.1 Ausweichplan	27
9 Arbeitszeitnachweis	28
10 Tooling und Technologie-Entscheidungen	29
10.1 Jira	29
10.2 Teams	29
10.3 GitHub	29
10.3.1 GitHub Guidelines	30
10.4 Dokumentation	30
10.4.1 Dokumentation Guidelines	30
10.4.2 Dokumentation Deployment	30
10.5 Code und Entwicklung	31
10.5.1 Code Guidelines	31

10.5.2 Setup	31
10.5.3 Hosting	31
10.5.4 Code Deployment	31
10.6 Programmiersprachen und Frameworks	32
10.7 Datenbank	33
10.8 KI-Tools	33
11 Sitzungsprotokolle	34
12 Persönliche Erfahrungsberichte	36
12.1 Bericht Joel Thoma	36
12.2 Bericht Vanessa Alves	36
III Glossar und Abkürzungsverzeichnis	37
IV Literaturverzeichnis	38
V Abbildungsverzeichnis	39
VI Tabellenverzeichnis	40
VII Code-Listings	43
VIII Anhang	44

Kapitel I

Produkt Dokumentation

1 Einleitung

Digitale Kommunikationsplattformen wie Discord haben sich in den vergangenen Jahren zunehmend als zentrale Werkzeuge für die Organisation und Interaktion innerhalb von Online-Communities und Vereinen etabliert. Insbesondere im Bereich der Freizeit- und Gaming-Vereine bieten sie die Möglichkeit, Kommunikation, Eventplanung und Mitgliederverwaltung zu bündeln. Trotz dieser Vorteile stossen bestehende Plattformfunktionen bei wachsender Mitgliederzahl, zunehmender organisatorischer Komplexität und steigenden Anforderungen an Datensicherheit und Automatisierung an ihre Grenzen.

Vor diesem Hintergrund befasst sich die vorliegende Bachelorarbeit mit der Weiterentwicklung eines im Vorprojekt erstellten Discord-Bots, der vereinsrelevante Prozesse automatisiert und Gamification-Mechanismen zur Steigerung des Engagements nutzt. Ziel der Arbeit ist es, das bestehende System, um ein plattformunabhängiges Admin-Dashboard zu erweitern, welches die sichere, benutzerfreundliche und rollenbasierte Verwaltung von Mitgliederdaten und Gamification-Statistiken ermöglicht. Damit wird ein Beitrag zur Entkopplung der Vereinsverwaltung von der Kommunikationsplattform Discord geleistet und die Grundlage für eine modulare, erweiterbare Systemarchitektur geschaffen.

1.1 Problemstellung

Das im Vorprojekt entwickelte System ermöglicht die Automatisierung einiger zentraler Vereinsprozesse, ist jedoch ausschliesslich innerhalb der Discord-Umgebung funktionsfähig. Dadurch ist die Nutzung stark Discord-zentriert und für Personen ohne Discord-Konto oder mit eingeschränkter technischer Erfahrung nur begrenzt zugänglich.

Die Bearbeitung vereinsrelevanter Daten erfolgt derzeit ausschliesslich über direkten Zugriff auf die MongoDB-Datenbank. Dies erfordert spezifische technische Kenntnisse, birgt Sicherheitsrisiken und verhindert eine kontrollierte, benutzerfreundliche Datenverwaltung. Zudem existiert keine zentrale Oberfläche, die eine visuelle und statistisch fundierte Auswertung des Vereinsengagements ermöglicht.

Darüber hinaus erschwert die aktuelle Architektur die Anbindung zusätzlicher Plattformen oder externer Systeme, etwa zur Erweiterung um Web- oder Mobile-Anwendungen. Diese Einschränkungen verdeutlichen den Bedarf nach einer modularen, skalierbaren und plattformunabhängigen Systemlösung, die sowohl technische als auch organisatorische Herausforderungen adressiert.

1.2 Vorarbeiten

Das Vorprojekt „Discord-Bot für Vereine“ ([Alves & Schnider, 2025](#)) bildete die Grundlage für die vorliegende Arbeit. Es verfolgte das Ziel, zentrale Vereinsprozesse wie Mitgliederverwaltung, Eventorganisation und Gamification in einem Discord-Bot zu integrieren. Der entwickelte Prototyp implementierte ein Rollen- und Berechtigungssystem, eine Anbindung an die Google Calendar API sowie ein auf wissenschaftlichen Erkenntnissen basierendes Gamification-Konzept.

Während die Machbarkeit und der Nutzen eines solchen Systems bestätigt werden konnten, blieb dessen Erweiterbarkeit auf andere Plattformen sowie die administrative Steuerung offen. Das Admin-Dashboard wurde im Vorprojekt lediglich als potenzielle Weiterentwicklung beschrieben, jedoch nicht realisiert. Ebenso fehlten Mechanismen zur Messung und Visualisierung des Engagements sowie eine Entkopplung der Discord-spezifischen Logik von der Datenbankstruktur.

Die vorliegende Arbeit schliesst an diese Ergebnisse an und adressiert die genannten Defizite durch eine Neugestaltung der Architektur, eine verbesserte Datenkonsistenz zwischen Discord, Dashboard und Datenbank sowie durch die Integration einer administrativen Benutzeroberfläche. Gleichzeitig werden Erkenntnisse aus Interviews mit Präsidenten anderer Vereine in die konzeptionelle Weiterentwicklung des Gamification-Systems einbezogen, um dessen Übertragbarkeit und Nachhaltigkeit zu erhöhen.

1.3 Ziel der Arbeit und Abgrenzung

Ziel dieser Bachelorarbeit ist die Konzeption und Entwicklung einer plattformunabhängigen, administrativen Erweiterung des bestehenden Discord-Bots, die eine effiziente und sichere Verwaltung vereinsrelevanter Daten ermöglicht. Hierbei liegt der Fokus auf der funktionalen Umsetzung eines Admin-Dashboards sowie auf der architektonischen Modularisierung des Systems, um künftige Integrationen weiterer Plattformen leichter zu ermöglichen. Die neue Softwarelösung wird mit dem Namen ClansCore adressiert.

Im Einzelnen verfolgt die Arbeit folgende Teilziele:

- Entwurf und Implementierung eines funktionsfähigen Admin-Dashboards in ClansCore, zur Verwaltung von Mitgliedern, Rollen, Events und Gamification-Daten
- Refaktorisierung der ClansCore-Backend-Architektur zur Entkopplung von Discord-spezifischer Logik und zur Vorbereitung auf eine plattformübergreifende Erweiterung
- Entwicklung einer sicheren Schnittstellenkommunikation zwischen Bot, Dashboard und Datenbank zur Gewährleistung konsistenter Datenzustände innerhalb des gesamten ClansCore-Systems
- konzeptionelle und technische Erweiterung des Gamification-Systems, insbesondere zur Messung des Mitgliederengagements
- Evaluation von ClansCore durch Usability-Tests mit Vereinsmitgliedern

Diese Arbeit unterscheidet sich vom Vorprojekt durch ihren erweiterten technischen und analytischen Anspruch. Während das Vorprojekt primär die Funktionalität innerhalb von Discord validierte, liegt der Schwerpunkt der Bachelorarbeit auf der systematischen Generalisierung und Professionalisierung der Lösung als Gesamt-System namens ClansCore. Damit wird die Grundlage geschaffen, um das System langfristig als universelles Vereinsmanagement-Tool einsetzen zu können.

1.4 Rahmenbedingungen

Die Arbeit wurde im Rahmen der Bachelor- und Studienarbeit des Studiengangs Informatik an der Ostschweizer Fachhochschule (OST) durchgeführt.

- **Arbeitstyp:** Bachelorarbeit (360 Stunden) + Studienarbeit (240 Stunden)
- **Gesamtarbeitsaufwand:** 600 Stunden
- **ECTS-Punkte:** 12 ECTS (Bachelorarbeit) und 8 ECTS (Studienarbeit)

Der Schwerpunkt liegt auf der Konzeption, Entwicklung und Evaluation eines funktionalen Software-Systems mit wissenschaftlich begründetem Gamification-Ansatz, welches den Namen ClansCore trägt. Die Arbeit vereint methodische Ansätze aus den Bereichen Software-Engineering, Usability und Motivationsforschung und ist als entwicklungsorientierte empirische Arbeit mit prototypischem Charakter einzuführen.

1.5 Stand der Technik

Der Stand der Technik im Bereich digitaler Vereinsorganisation wurde im Vorprojekt umfassend untersucht und bildet die Grundlage der vorliegenden Arbeit. Die damalige Marktanalyse zeigte, dass bestehende Systeme entweder auf die Interaktion und Gamification innerhalb von Discord oder auf die administrative Vereinsverwaltung über klassische Softwarelösungen fokussieren, jedoch keine integrative Verbindung beider Ansätze bieten.

Seit Abschluss des Vorprojekts im Juli 2025 sind zwar Weiterentwicklungen einzelner Systeme erkennbar, eine Lösung, welche Gamification, Datenverwaltung und plattformübergreifende Nutzung in einem kohärenten Konzept vereint, wurde bisher jedoch nicht öffentlich dokumentiert.

Im Bereich der Discord-Bots zählt MEE6 zu den meistgenutzten Anwendungen und bietet über das Plugin „Statistics Channels“ grundlegende Auswertungen zur Serveraktivität sowie ein Levelsystem zur Förderung der Nutzerinteraktion (MEE6-Contributors, 2025). Auch der Bot Arcane integriert vergleichbare Gamification-Funktionen in Form von Belohnungssystemen (Arcane-Contributors, 2025). Beide Systeme bleiben jedoch auf die Discord-Plattform beschränkt und verfügen über keine dokumentierten Schnittstellen zu externen Administrations- oder Datenverwaltungssystemen.

Demgegenüber adressieren klassische Vereinsmanagement-Lösungen wie ClubDesk oder WildApricot die organisatorischen Anforderungen von Vereinen, insbesondere in den Bereichen Mitgliederverwaltung, Eventplanung und Buchhaltung (ClubDesk-Contributors, 2025; WildApricot-Contributors, 2025). Motivierende oder interaktive Komponenten, wie sie in Gamification-Konzepten beschrieben werden, sind in diesen Systemen kein grosser Bestandteil. Diskussionen über eine

mögliche Erweiterung durch Belohnungsmechanismen existieren lediglich in Anwenderforen, was die bislang geringe praktische Umsetzung solcher Konzepte verdeutlicht ([WildApricot-Community, 2015](#)).

Die vorliegende Arbeit positioniert sich zwischen diesen beiden Ansätzen, indem sie den organisatorischen Verwaltungsfokus klassischer Systeme mit der motivierenden Wirkung von Gamification-Elementen in [ClansCore](#) kombiniert. Durch die laufende Einführung des vorherigen Systems (Discord-Bot) beim Verein [EGSwiss](#) über das Teammitglied Vanessa Alves, während der Sommermonate 2025, konnten Rückmeldungen des Vorstandes gesammelt und praxisrelevante Anforderungen sowie Verbesserungsvorschläge identifiziert werden. Diese sind in der bestehenden Softwarelösung bislang unberücksichtigt geblieben sind. Damit leistet die Arbeit einen Beitrag zur Integration von Motivation, Datenmanagement und Vereinsorganisation in einem einheitlichen, plattformunabhängigen Anwendungskonzept ClansCore.

2 Erweitertes Gamification-Konzept

Bereits im Vorprojekt wurden erste Erkenntnisse zu Motivation, Hindernissen und Gamification im Vereinskontext erhoben. Diese basierten auf eine Nutzergruppen-Analyse mit Nicht-Mitglieder, Vereins-Mitglieder und Vorstands-Mitglieder. Die Ergebnisse lieferten eine breite, quantitative Datengrundlage, die allgemeine Tendenzen sichtbar machen, beispielsweise welche Aktivitäten besonders motivierend wirken oder welche Risiken bei der Einführung von Gamification-Systemen bestehen ([Alves & Schnider, 2025](#)).

Für die vorliegende Arbeit war es jedoch notwendig, über diese Vorarbeit hinauszugehen. Während die Umfragen des Vorprojekts vor allem die Perspektive der Mitglieder und Erfahrungswerte von Vanessa Alves als Präsidentin von [EGSwiss](#) abbildeten, sollen nun die Erfahrungen und Einschätzungen anderer Vereinsvorstände, systematisch untersucht werden. Präsident:innen und Vorstände sind zentrale Entscheidungsträger in Vereinen und können eine strategische Sicht auf Motivation und Engagement bieten. Ihre Sichtweise ist entscheidend, um zu verstehen, welche strukturellen und organisatorischen Massnahmen tatsächlich umsetzbar sind und wie digitale Tools wie der Discord-Bot oder ein Admin-Dashboard sie dabei unterstützen können.

2.1 Interviews zur Motivation von Vereinsmitgliedern

Das Ziel der Interviews ist es, einen praxisnahen Abgleich zwischen theoretischen Konzepten und organisatorischer Realität zu schaffen. Während die Vorarbeit vor allem beschrieb, was Mitglieder motiviert, soll jetzt untersucht werden, wie Vorstände Motivation fördern und welche Unterstützung sie dabei benötigen.

Konkret sollen die Interviews:

- **Motivationsstrategien der Vereine erfassen:** Welche Massnahmen nutzen Vorstände aktuell, um Mitglieder langfristig an den Verein zu binden?
- **Herausforderungen aus Leitungsperspektive beleuchten:** Welche strukturellen Hürden sehen Vorstände bei Engagement und Motivation?
- **Einsatz digitaler Werkzeuge untersuchen:** Welche Rolle spielen Plattformen wie Discord oder Social Media für Vorstände selbst?
- **Poteniale von Gamification im Management identifizieren:** Welche Chancen sehen Vorstände in Belohnungssystemen, und welche Risiken gilt es zu vermeiden?
- **Anforderungen an digitale Unterstützung ableiten:** Welche Features im Discord-Bot und welche Funktionen im Admin-Dashboard würden Vorstände konkret entlasten oder Motivation steigern?

Damit erweitern die Interviews den Erkenntnisstand aus der Vorarbeit um externe, strategische Führungsperspektiven.

2.1.1 Methodisches Vorgehen

Die Interviews wurden als halb-strukturierte, qualitative Gespräche konzipiert. Im Unterschied zur standardisierten Umfrage im Vorprojekt ermöglicht diese Methode detaillierte Einblicke und erlaubt es, individuell auf die Antworten einzugehen.

Die Auswahl der Interviewpartner erfolgte gezielt. Befragt wurden Präsident:innen und Vorstands-Mitglieder von Gaming-Vereinen, da sie einerseits die operative Situation im Verein kennen und andererseits Entscheidungen über zukünftige Massnahmen treffen. Diese Zielgruppe liefert somit nicht nur Einschätzungen zur Motivation der Mitglieder, sondern auch Anforderungen an Management- und Organisationswerkzeuge.

Die Interviews dauerten 30 bis 45 Minuten und folgten einem thematisch gegliederten Leitfaden. Die Gesprächspartner wurden dabei schrittweise von allgemeinen Fragen über Motivation und Aktivitäten hin zu digitalen Tools und schliesslich zu hypothetischen Feature-Ideen geführt.

2.1.2 Interviewleitfaden

Der Leitfaden wurde in sieben Blöcke gegliedert, die einen roten Faden gewährleisten:

1. **Allgemeine Fragen:** Kontext zu Rolle, Vereinsgrösse, Gründungsjahr, um eine Grundlage zur Vergleichbarkeit der Antworten zu schaffen.
2. **Vereinsn motivation:** aktuelle Beitragsgründe, Bindungsfaktoren, Herausforderungen im Verein.
3. **Aktivitäten und Engagement:** motivierende Angebote, Unterschiede zwischen neuen und erfahrenen Mitgliedern, Umgang mit Passivität.
4. **Digitale Tools und Gamification:** Nutzung digitaler Plattformen, Erfahrungen mit spielerischen Elementen, Chancen und Risiken.
5. **Zukunft und Anpassungen:** Wünsche nach Belohnungen, nachhaltige Fördermassnahmen.
6. **Digitale Brücke:** Hypothetische Fragen zu einem digitalen Tool, Kennzahlen und gewünschte Features.
7. **Abschluss:** Platz für offene Ergänzungen.

Wichtig ist, dass die Interviewten erst in den letzten Blöcken mit der Möglichkeit digitaler Lösungen konfrontiert wurden. Dadurch konnten zunächst unverfälschte Aussagen zur Vereinsn motivation gewonnen werden, bevor ein Bezug zu einem Discord-Bot oder Admin-Dashboard hergestellt wurde.

2.1.3 Mehrwert gegenüber dem Vorprojekt

Die Interviews schaffen einen klaren Mehrwert gegenüber der Vorarbeit. Während die Umfragen des Vorprojekts Mitgliederperspektiven in quantitativer Form erfassten, liefern die Interviews qualitative Einblicke aus Führungssicht. Anstelle allgemeiner Trends stehen hier strategische und organisatorische Aspekte im Vordergrund, zum Beispiel wie Vorstände Motivation gezielt steuern oder wo sie digitale Unterstützung benötigen. Die Ergebnisse sind nicht nur eine Wiederholung, sondern dienen dazu, konkrete Anforderungen für die Weiterentwicklung des Discord-Bots und Erstellung des Admin-Dashboards abzuleiten. Die Ergebnisse liefern einen wertvollen Einblick in die Fragen nach Kennzahlen und Managementfunktionen im Dashboard.

Damit stellen die Interviews ein zentrales Bindeglied zwischen Theorie (Motivations- und Gamification-Modelle), Praxis (Mitgliederumfragen aus der Vorarbeit) und der konkreten Softwareentwicklung in dieser Bachelorarbeit dar.

3 Anforderungen

Dieses Kapitel stellt die Anforderungen an ClansCore für den Verein systematisch dar. Die Anforderungen basieren auf klar definierten User Stories, welche die Bedürfnisse der Nutzer abbilden. Aus diesen User Stories werden die Functional Requirements abgeleitet, welche die konkreten Funktionen des Bots und des Dashboards spezifizieren. Darauf aufbauend definieren die Use Cases, wie diese Anforderungen in der geplanten Umsetzung realisiert werden. Neben den Functional Requirements werden auch die Non-Functional Requirements berücksichtigt, um Qualitätsmerkmale wie Sicherheit, Skalierbarkeit und Benutzerfreundlichkeit sicherzustellen. Zudem erfolgt eine Priorisierung der Anforderungen, damit die wichtigsten Funktionen frühzeitig implementiert werden.

3.1 Priorisierung der Anforderungen

Damit der Entwicklungsprozess auf die wichtigsten Features fokussiert bleibt, erhalten gewisse Anforderungen eine höhere Priorität als andere. Dadurch kann zügig ein Prototyp entwickelt, getestet und auf Basis von Nutzerfeedback iterativ verbessert werden. Die nicht optionalen Use Cases und Non-Functional Requirements bilden das MVP (Minimum Viable Product) und stellen die funktionale Basis des Projekts dar.

Die Nachfolgende Auflistung beschreibt die Prioritäts-Kategorien und deren Bedeutung:

Prioritäts-Kategorie	Beschreibung
Hoch (<u>MVP</u>) - Essenziell für die erste Version	Diese Anforderungen sind direkt mit den Kernanforderungen des MVP verknüpft und müssen für eine funktionale Erstversion vorhanden sein.
Mittel - Wichtige Funktionen nach dem MVP	Diese Anforderungen ergänzen das System sinnvoll, sind jedoch nicht kritisch für die erste Version.
Niedrig - Erweiterungen für spätere Versionen	Diese Anforderungen verbessern die Benutzerfreundlichkeit und Verwaltung, sind aber nicht erforderlich für den ersten MVP-Release.

Tab. 1: Beschreibung der Prioritäts-Kategorien

3.2 Legende zum Erfüllungsgrad der Anforderungen

Die Resultate der Anforderungen werden jeweils in den Anforderungstabellen dokumentiert. Zusätzlich wird der Status farblich hervorgehoben, um eine schnelle Übersicht über den Erfüllungsgrad zu ermöglichen.

Farbe	Status
grün	Ziel der Anforderung wurde erreicht.
gelb	Ziel der Anforderung wurde zum Teil oder über einen anderen Weg erreicht.
rot	Ziel der Anforderung wurde nicht erreicht.

Tab. 2: Beschreibung der Resultats-Kategorien

3.3 User Stories

Die User Stories beschreiben die Bedürfnisse und Erwartungen der Nutzer an ClansCore und formulieren sie aus der Perspektive der Endanwender. Sie bilden die Grundlage für die Functional Requirements, die daraus abgeleitet werden. Die Analyse konzentriert sich auf die Nutzerrollen Vorstand, Mitglied und Nicht-Mitglied.

3.3.1 User Story: Vorstandsmitglied

Als Vorstandsmitglied...

- möchte ich, dass Mitglieder automatisch eine Erinnerung für den fälligen Mitgliederbeitrag bekommen, damit ich diese nicht selber schicken muss.
- möchte ich eine Zahlungsbestätigung erhalten nach der Einzahlung von einem Mitgliederbeitrag, damit ich die Zahlungen nicht von Hand durchsuchen muss.

- möchte ich, über eine Webseite Mitglieder, Rollen und Events verwalten können, um die Organisation übersichtlicher und effizienter zu gestalten.
- möchte ich über eine Webseite neue Aufgaben mit Belohnungen erstellen, um die Interaktion im Verein zu fördern.
- möchte ich Gamification-Parameter erfassen können, damit die automatische Punkteanzahl für eine Aufgabe berechnet werden kann.
- möchte ich, detaillierte Gamification-Statistiken und Mitgliedsaktivitäten einsehen damit ich diese analysieren und Entscheidungen treffen kann.
- möchte ich, dass man sich nur mit einer Zwei-Faktor-Authentifizierung direkt mit der Datenbank verbinden kann, damit sensible Daten vor unautorisiertem Zugriff geschützt sind.
- möchte ich, dass alle zusammenhängenden Datenbankoperationen in Transaktionen ausgeführt werden, damit die Datenintegrität gewährleistet ist und keine fehlerhaften oder unvollständigen Daten im System gespeichert werden.

Als Mitglied...

- möchte ich eine Erinnerung für meinen Mitgliederbeitrag erhalten, damit ich diesen nicht vergesse zu zahlen.
- möchte ich aktive Aufgaben annehmen können, damit ich Punkte sammeln kann.
- möchte ich aktuelle Aufgaben, meine Punkte und die Rangliste auf einer Webseite einsehen können, damit ich immer auf dem aktuellen Stand bin.

Als Nicht-Mitglied...

- möchte ich ein Formular online ausfüllen, um mich für den Verein zu bewerben, damit ich beitreten kann und die entsprechende Rolle erhalte.

Als Verein...

- möchte ich nicht nur auf Discord angewiesen sein für meine Vereinsorganisation.

3.4 Functional Requirements

Die Functional Requirements leiten sich aus den User Stories ab und spezifizieren die erwarteten Funktionen von ClansCore. Sie beschreiben, welche Features implementiert werden müssen, um die definierten Anforderungen zu erfüllen. Zudem sind sie eine zentrale Grundlage für die Use Cases, die detailliert darstellen, wie Nutzer mit ClansCore interagieren.

3.4.1 Discord-Bot

ID	FR1
Name	Zahlungserinnerung
Beschreibung	<ul style="list-style-type: none"> Der Discord-Bot muss nach Ablauf eines Vereinsjahres eine Zahlungserinnerung an das Mitglied verschicken

Tab. 3: Beschreibung Functional Requirement 1

ID	FR2
Name	Zahlungsbestätigung
Beschreibung	<ul style="list-style-type: none"> Der Discord-Bot muss nach einer erfolgreichen Einzahlung von einem Mitglied eine Zahlungsbestätigung an den Vorstand verschicken

Tab. 4: Beschreibung Functional Requirement 2

3.4.2 Admin-Dashboard

ID	FR3
Name	Login
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss ein Loginsystem bereitstellen.

Tab. 5: Beschreibung Functional Requirement 3

ID	FR4
Name	Mitglied Bewerbung
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss ermöglichen sich für den Verein zu bewerben.

Tab. 6: Beschreibung Functional Requirement 4

ID	FR5
Name	Passwort Zurücksetzen
Beschreibung	<ul style="list-style-type: none"> Es muss eine Möglichkeit geben, um sein Passwort zurückzusetzen.

Tab. 7: Beschreibung Functional Requirement 5

ID	FR6
Name	Passwort Ändern
Beschreibung	<ul style="list-style-type: none"> Es muss eine Möglichkeit geben, um sein Passwort zu ändern.

Tab. 8: Beschreibung Functional Requirement 6

ID	FR7
Name	Verwaltung von Mitgliedern, Rollen sowie Events
Beschreibung	<p>Ein Web-Interface (Admin-Dashboard) muss bereitgestellt werden, in dem Vorstandsmitglieder folgende Funktionen zur Verfügung haben:</p> <ul style="list-style-type: none"> Mitglieder und deren Rollen verwalten Events inklusive Erinnerungen erstellen, bearbeiten und löschen

Tab. 9: Beschreibung Functional Requirement 7

ID	FR8
Name	Steuerung der Bot-Funktionen
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss ermöglichen, die Zugriffsrechte auf Befehle zu bearbeiten (welche Rollen welche Befehle nutzen dürfen). Das Admin-Dashboard muss ermöglichen, automatisierte Nachrichten zu konfigurieren.

Tab. 10: Beschreibung Functional Requirement 8

ID	FR9
Name	Gamification-Parameter Verwalten
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss die Erfassung und Bearbeitung von Gamification-Parametern ermöglichen.

Tab. 11: Beschreibung Functional Requirement 9

ID	FR10
Name	Gamification-Datenverwaltung
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss ermöglichen, Aktivitäten zu erstellen und automatisch dazu eine Punkteanzahl zu empfehlen. Das Admin-Dashboard muss ermöglichen, detaillierte Gamification-Statistiken und Mitgliedsaktivitäten einzusehen und zu analysieren.

Tab. 12: Beschreibung Functional Requirement 10

ID	FR11
Name	Gamification-Ansicht
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss ermöglichen aktuelle Ranglisten und Aufgaben sowie die eigenen Punkte anzuzeigen.

Tab. 13: Beschreibung Functional Requirement 11

ID	FR12
Name	Aufgaben Anmeldung
Beschreibung	<ul style="list-style-type: none"> Das Admin-Dashboard muss die Anmeldung für Aufgaben ermöglichen.

Tab. 14: Beschreibung Functional Requirement 12

ID	FR13
Name	Datenbank Transaktionen
Beschreibung	<ul style="list-style-type: none"> Das System stellt sicher, dass alle zusammenhängenden Datenbankoperationen innerhalb von Transaktionen ausgeführt werden, um die Datenintegrität zu gewährleisten.

Tab. 15: Beschreibung Functional Requirement 13

ID	FR14
Name	Datenbank Zwei-Faktor-Authentifizierung
Beschreibung	<ul style="list-style-type: none"> Das System muss für alle direkten Datenbankanmeldungen eine Zwei-Faktor-Authentifizierung erzwingen.

Tab. 16: Beschreibung Functional Requirement 14

3.4.3 Neue Plattform

ID	FR15
Name	Discord-Bot Funktionen für eine Neue Plattform
Beschreibung	<p>Die neue Plattform wird prototypisch mit den nachfolgenden Funktionen implementiert:</p> <ul style="list-style-type: none"> Bewerbungsformular für neue Mitglieder Personenbezogene Informationsabfrage aus Datenbank

Tab. 17: Beschreibung Functional Requirement 15

3.5 Use Cases

Die Use Cases beschreiben detailliert die Interaktionen zwischen Nutzern und ClansCore und zeigen, wie die Functional Requirements umgesetzt werden. Darüber hinaus unterstützen sie die Systemarchitektur, indem sie Anforderungen an Sicherheit, Skalierbarkeit und Benutzerfreundlichkeit konkretisieren.

ID	UC1
Name	Benachrichtigungen und Informationen zu Zahlungen (FR1 , FR2)
Akteur	Mitglied, Vorstand, Discord-Bot
Beschreibung	Mitglieder zahlen jährlich einen Mitgliederbeitrag an den Verein. Alle Zahlungen an den Verein zu durchsuchen und zu schauen, dass jeder seinen Beitrag gezahlt hat ist eine zeitintensive Aktivität. Der Discord-Bot soll Zahlungserinnerung an die Mitglieder und Zahlungsbestätigungen an den Vorstand verschicken.
Voraussetzung	Das Mitglied hat seinen fälligen Mitgliederbeitrag noch nicht gezahlt.
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Discord-Bot schickt dem Mitglied eine Zahlungserinnerung 2. Das Mitglied zahlt seinen jährlichen Mitgliederbeitrag 3. Die Zahlung trifft korrekt auf dem Konto des Vereins ein 4. Der Discord-Bot speichert das Datum der Zahlung in der Vereinsdatenbank 5. Das Vorstandsmitglied wird über die Zahlung benachrichtigt
Alternativ-Verlauf	<ul style="list-style-type: none"> • Bei einem neuen Mitglied <ol style="list-style-type: none"> 6. Das neue Mitglied wird durch das Vorstandsmitglied angenommen 7. Die Rolle wird automatisch zugewiesen • Das Vorstandsmitglied kann nach der Einzahlung vom Mitglied die Bewerbung trotzdem ablehnen und die Zahlung manuell rückerstatten • Falls das Mitglied nach der Erinnerung durch den Discord-Bot den Beitrag immer noch nicht einzahlt, sieht der Vorstand dies in der Liste der offenen Beiträge und kann nach eigenem Ermessen vorgehen • Falls die Einzahlung vom Mitglied nicht richtig zugeteilt werden konnte muss der Vorstand manuell eintragen, dass das Mitglied den Beitrag gezahlt hat
Nachbedingung	Das Datum der Zahlung des Mitgliedes ist in der Vereinsdatenbank registriert.
Priorität	Medium
Resultat	

Tab. 18: Beschreibung Fully dressed Use Case 1

ID	UC2
Name	Login (FR3)
Akteur	Mitglied, Vorstand
Beschreibung	Die Benutzer kommen nur über ein Login auf die ClansCore Seite.
Voraussetzung	Der Benutzer welcher sich versucht einzuloggen besitzt bereits Logindaten.
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Benutzer gibt die Logindaten ein 2. ClansCore überprüft die Korrektheit der eingegebenen Daten 3. Nach erfolgreicher Überprüfung wird der Benutzer auf die Startseite von ClansCore weitergeleitet
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls die eingegebenen Logindaten keinem gültigem Login entsprechen, wird eine Fehlermeldung ausgegeben • Falls der Benutzer sich nicht mehr an die Logindaten erinnern kann muss die Passwort-Zurücksetzung gestartet werden. (UC4)
Nachbedingung	Der Benutzer ist eingeloggt.
Priorität	Hoch
Resultat	

Tab. 19: Beschreibung Fully dressed Use Case 2

ID	UC3	
Name	Mitglied Bewerbung (<u>FR4</u>)	
Akteur	Nicht-Mitglied	
Beschreibung	Nicht-Mitglieder können sich für eine Mitgliedschaft beim Verein bewerben.	
Voraussetzung	Das Admin-Dashboard ist eingerichtet und mit der Vereinsdatenbank verbunden.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Das Nicht-Mitglied füllt das Bewerbungsformular aus 2. Die Daten werden in der Vereinsdatenbank gespeichert und der Vorstand wird informiert 3. Der Vorstand prüft die Bewerbung und nimmt das Mitglied in den Verein auf 4. Die Rollen werden entsprechend automatisch gesetzt 5. Das neue Mitglied wird über die Aufnahme in den Verein informiert 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls das Bewerbungsformular unvollständig oder fehlerhaft ist, wird eine entsprechende Fehlermeldung ausgegeben und es wird eine Korrektur angefordert. • Falls die Bewerbung abgelehnt wird, erhält das Nicht-Mitglied eine Benachrichtigung. 	
Nachbedingung	Das Mitglied ist in der Vereinsdatenbank registriert und hat eine zugewiesene Rolle oder wurde abgelehnt.	
Priorität	Hoch	
Resultat		

Tab. 20: Beschreibung Fully dressed Use Case 3

ID	UC4	
Name	Passwort Zurücksetzung (<u>FR5</u>)	
Akteur	Mitglied, Vorstand	
Beschreibung	Wenn ein Benutzer sein Passwort vergessen hat, muss er dieses Zurücksetzen können.	
Voraussetzung	Der Benutzer besitzt einen Account für das Login.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Auf der Login-Seite wird die Passwort-Zurücksetzung gestartet. 2. Der Benutzer bekommt eine Möglichkeit sein Passwort zu ändern. 3. Der Benutzer kann sich erfolgreich einloggen. (<u>UC2</u>) 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls der Benutzer noch keinen Account besitzt kann kein neues Passwort verschickt werden und Vorgang wird abgebrochen. 	
Nachbedingung	Das neue Passwort wurde gespeichert.	
Priorität	Hoch	
Resultat		

Tab. 21: Beschreibung Fully dressed Use Case 4

ID	UC5	
Name	Passwort Ändern (<u>FR6</u>)	
Akteur	Mitglied, Vorstand	
Beschreibung	Benutzer welche einen Account besitzen müssen ihr Passwort ändern können.	
Voraussetzung	Der Benutzer ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Benutzer muss das alte Passwort wie auch ein neues Passwort eingeben 2. Das alte Passwort wird auf Korrektheit überprüft 3. Die Änderung des Passwortes wird gespeichert 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls das alte Passwort nicht korrekt eingegeben wurde wird eine entsprechende Fehlermeldung ausgegeben 	
Nachbedingung	Das neue Passwort wurde gespeichert.	
Priorität	Mittel	
Resultat		

Tab. 22: Beschreibung Fully dressed Use Case 5

ID	UC6	
Name	Verwaltung von Mitgliedern und Bot-Funktionen im Admin-Dashboard (FR7 , FR8)	
Akteur	Vorstand	
Beschreibung	Der Vorstand kann Mitglieder, Rollen und die Rechte über ein Admin-Dashboard verwalten.	
Voraussetzung	Das Admin-Dashboard ist eingerichtet und mit der Vereinsdatenbank verbunden. Das Vorstandsmitglied ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Vorstand kann über das Admin-Dashboard Mitglieder, Rollen, Events, und die Rechte verwalten 2. Änderungen werden in die Datenbank übernommen 	
Alternativ-Verlauf	Falls falsche Eingaben gemacht werden, zeigt das Dashboard eine Fehlermeldung.	
Nachbedingung	Änderungen sind korrekt gespeichert und umgesetzt.	
Priorität	Hoch	
Resultat		

Tab. 23: Beschreibung Fully dressed Use Case 6

ID	UC7	
Name	Gamification-Parameter Verwalten (FR9)	
Akteur	Vorstand	
Beschreibung	Um eine akkurate Punkteanzahl bei der Erstellung von Aufgaben vorzuschlagen müssen Gamification-Parameter verwaltet werden können.	
Voraussetzung	Das Vorstandsmitglied ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Die Gamification-Parameter werden bearbeitet 2. Die Daten werden in der Vereinsdatenbank gespeichert 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls die eingegebenen Daten unvollständig oder fehlerhaft sind, wird eine entsprechende Fehlermeldung ausgegeben und es wird eine Korrektur angefordert. 	
Nachbedingung	Änderungen sind korrekt gespeichert.	
Priorität	Hoch	
Resultat		

Tab. 24: Beschreibung Fully dressed Use Case 7

ID	UC8	
Name	Verwaltung von Aufgaben und Belohnungen (FR10)	
Akteur	Vorstand	
Beschreibung	Der Vorstand kann Aufgaben verwalten.	
Voraussetzung	Das Admin-Dashboard ist eingerichtet und mit der Vereinsdatenbank verbunden. Das Vorstandsmitglied ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Vorstand gibt Daten zur Aufgabenerstellung an 2. Es wird ein Punktevorschlag dargestellt der durch die eingegebenen Angaben berechnet wurde 3. Der Vorstand erstellt und veröffentlicht die Aufgabe 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls die Gamification-Parameter nicht gesetzt sind kann kein Punktevorschlag erstellt werden. • Falls falsche Eingaben gemacht werden, zeigt das Dashboard eine Fehlermeldung an. 	
Nachbedingung	Änderungen sind korrekt gespeichert und umgesetzt.	
Priorität	Hoch	
Resultat		

Tab. 25: Beschreibung Fully dressed Use Case 8

ID	UC9	
Name	Ansicht Aufgaben und Ranglisten (FR10 , FR11)	
Akteur	Mitglied, Vorstand	
Beschreibung	Mitglieder können aktuelle Aufgaben, Ranglisten sowie die eigenen Punkte einsehen.	
Voraussetzung	Das Admin-Dashboard ist eingerichtet und mit der Vereinsdatenbank verbunden. Der Benutzer ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Mitglieder erhalten Punkte für Aktivitäten (z. B. Event-Teilnahme, Spenden) 2. Mitglieder können Ranglisten und ihren aktuellen Punktestand einsehen 	
Alternativ-Verlauf	<ol style="list-style-type: none"> 1. Der Vorstand kann dazu auch detaillierte Gamification-Statistiken und Mitgliedsaktivitäten einzusehen 	
Nachbedingung	Punkte werden korrekt verwaltet und angezeigt.	
Priorität	Hoch	
Resultat		

Tab. 26: Beschreibung Fully dressed Use Case 9

ID	UC10	
Name	Aufgaben Anmeldung (FR12)	
Akteur	Mitglied	
Beschreibung	Damit Mitglieder Punkte sammeln können müssen sich diese für Aufgaben anmelden	
Voraussetzung	Das Mitglied ist eingeloggt.	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Es wird eine Aufgabe angewählt und sich dafür angemeldet 2. Die Anmeldung wird in der Vereinsdatenbank gespeichert 	
Alternativ-Verlauf	<ul style="list-style-type: none"> • Falls ein Fehler bei der Anmeldung geschieht, wird eine entsprechende Fehlermeldung ausgegeben. 	
Nachbedingung	Anmeldung ist korrekt gespeichert.	
Priorität	Hoch	
Resultat		

Tab. 27: Beschreibung Fully dressed Use Case 10

ID	UC11	
Name	Datenbank Transaktionen (FR13)	
Akteur	Mitglied, Vorstand	
Beschreibung	Sicherstellung der Datenkonsistenz durch Transaktionen	
Voraussetzung	Es wird eine Datenbankoperation durchgeführt	
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Benutzer initiiert eine Datenbankoperation welche keine Leseoperation ist 2. Das System startet eine Datenbanktransaktion 3. Das System führt alle zugehörigen Datenbankoperationen aus 4. Wenn alle Operationen erfolgreich sind, bestätigt das System die Transaktion 	
Alternativ-Verlauf	<ol style="list-style-type: none"> 4. Falls eine Operation fehlschlägt rollt das System die Transaktion zurück 5. Das System informiert den Benutzer über den Fehler 	
Nachbedingung	Die Datenbank befindet sich nach der Ausführung in einem konsistenten Zustand	
Priorität	Mittel	
Resultat		

Tab. 28: Beschreibung Fully dressed Use Case 11

ID	UC12
Name	Datenbank Zwei-Faktor-Authentifizierung (FR14)
Akteur	Vorstand, Admin
Beschreibung	Bei einer direkten Verbindung mit der Datenbank kann sich nur mit Zwei-Faktor-Authentifizierung angemeldet werden.
Voraussetzung	Der Benutzer startet eine direkte Verbindung zur Datenbank. Die Datenbank hat die Zwei-Faktor-Authentifizierung aktiviert.
Standard-Verlauf	<ol style="list-style-type: none"> 1. Der Benutzer gibt seine Anmelde Daten ein 2. Das System prüft die Zugangsdaten und fordert den zweiten Faktor an 3. Der Benutzer authentifiziert sich mit dem zweiten Faktor 4. Die Datenbank prüft die 2FA-Daten und erlaubt bei Erfolg den Zugriff
Alternativ-Verlauf	<ol style="list-style-type: none"> 2. Bei falschem Benutzername oder Passwort wird der Zugriff verweigert und eine Fehlermeldung wird angezeigt 4. Bei ungültigem zweiten Faktor wird der Zugriff verweigert und eine Fehlermeldung wird angezeigt
Nachbedingung	Benutzer ist mit der Datenbank verbunden
Priorität	Mittel
Resultat	

Tab. 29: Beschreibung Fully dressed Use Case 12

ID	UC13
Name	Neue Plattform(FR15)
Akteur	Mitglied, Neue Plattform
Beschreibung	Auf einer neuen Plattform werden prototypisch wenige Funktionen des Discord-Bot implementiert. Es wird das hinzufügen eines neuene Mitgliedes und die Informationsabfrage aus der Datenbank implementiert.
Voraussetzung	Die neue Plattform ist mit der Vereinsdatenbank verbunden.
Standard-Verlauf	<ol style="list-style-type: none"> 1. Nicht-Mitglied füllt Bewerbungsformular aus. 2. Nach einer expliziten Zustimmung der Datennutzungs-Bedingungen wird das Formular abgesendet. (UC7) 3. Der Discord-Bot speichert die Daten in der Vereinsdatenbank und informiert den Vorstand. 4. Der Vorstand prüft die Bewerbung und setzt den Status des Mitgliedes 5. Das Mitglied kann nun mit einem Befehl die gespeicherten Daten von sich selber anfragen.
Alternativ-Verlauf	<ul style="list-style-type: none"> • Die Bewerbung wird abgelehnt und es wird kein neues Mitglied hinzugefügt
Nachbedingung	Das neue Mitglied ist in der Datenbank gespeichert
Priorität	Niedrig
Resultat	

Tab. 30: Beschreibung Fully dressed Use Case 13

3.6 Non-Functional Requirements

Die Non-Functional Requirements definieren die Qualitätsmerkmale von ClansCore, darunter Sicherheit, Performance, Skalierbarkeit und Usability. Sie beeinflussen die Architektur- und Designentscheidungen, um eine zuverlässige und effiziente Nutzung sicherzustellen. Zudem stellen sie sicher, dass das System auch bei steigender Nutzung robust und erweiterbar bleibt.

ID	NFR1	
Beschreibung	Die Code Qualität entspricht dem im Linter definierten Anforderungen.	
Anforderungen	Maintainability - Analyzability	
Priorität	Hoch	
Messung	Es wird manuell überprüft ob der Linter Fehlermeldungen ausgibt.	
Testen	Nach jedem Commit	
Resultat		

Tab. 31: Beschreibung Non-Functional Requirement 1

ID	NFR2	
Beschreibung	Die Testabdeckung der Applikation beträgt mindestens 70%.	
Anforderungen	Maintainability - Testability	
Priorität	Hoch	
Messung	Test Coverage wird mittels des verwendeten Testing-Frameworks bestimmt.	
Testen	Am Ende	
Resultat		

Tab. 32: Beschreibung Non-Functional Requirement 2

ID	NFR3	
Beschreibung	Fehler werden gefangen und entsprechend behandelt. Es sollte nicht zum Absturz kommen.	
Anforderungen	Reliability - Fault tolerance	
Priorität	Hoch	
Messung	Es werden Usability-Tests durchgeführt, bei denen die Tester gezielt aufgefordert werden den Discord-Bot, das Admin Dashboard und die neue Plattform zum Absturz zu bringen.	
Testen	Am Ende	
Resultat		

Tab. 33: Beschreibung Non-Functional Requirement 3

ID	NFR4	
Beschreibung	Das Design des Admin-Dashboards ist auf Desktop-Systeme ausgelegt. Es werden etablierte Design-Praktiken verwendet um die Usability des Dashboards zu gewährleisten.	
Anforderungen	Usability	
Priorität	Hoch	
Messung	Es werden Usability-Tests durchgeführt.	
Testen	Am Ende	
Resultat		

Tab. 34: Beschreibung Non-Functional Requirement 4

ID	NFR5	
Beschreibung	Das Admin-Dashboard kann auf den Browsern Google Chrome und Firefox verwendet werden.	
Anforderungen	Compatibility - Interoperability	
Priorität	Hoch	
Messung	Es werden Usability-Tests durchgeführt.	
Testen	Am Ende	
Resultat		

Tab. 35: Beschreibung Non-Functional Requirement 5

ID	NFR6	
Beschreibung	Der Datenschutz der Nutzer ist gewährleistet.	
Anforderungen	Security	
Priorität	Hoch	
Messung	Es wird ein Sicherheitskonzept erstellt, welche als Grundlage für die Implementation verwendet wird. Zudem werden Usability-Tests durchgeführt.	
Testen	Am Ende	
Resultat		

Tab. 36: Beschreibung Non-Functional Requirement 6

ID	NFR7	
Beschreibung	Das Gamification-System ist gegen Manipulation geschützt.	
Anforderungen	Security	
Priorität	Hoch	
Messung	Es wird ein Gamification-Konzept erstellt, welches Schritte zur Erkennung und Verhinderung von Manipulation enthält. Dieses Konzept wird als Grundlage für die Implementation verwendet. Zudem werden Usability-Tests durchgeführt.	
Testen	Am Ende	
Resultat		

Tab. 37: Beschreibung Non-Functional Requirement 7

ID	NFR8	
Beschreibung	Nach einem Commit auf den Main Branch und erfolgreich ausgeführten Tests wird ein neues Image erstellt und das Deployment ausgelöst.	
Anforderungen	Maintainability	
Priorität	Hoch	
Messung	Es wird geprüft, ob die neueste Version auf dem Server ist und die Pipeline keinen Fehler ausgibt.	
Testen	Nach jedem Commit auf den Main Branch	
Resultat		

Tab. 38: Beschreibung Non-Functional Requirement 8

ID	NFR9	
Beschreibung	Das System soll Logging von Datenbankaktivitäten gewährleisten, um Auditierbarkeit, Sicherheit und Nachvollziehbarkeit aller Datenzugriffe und -änderungen sicherzustellen..	
Anforderungen	Security	
Priorität	Mittel	
Messung	Es wird manuell ein Test durchgeführt	
Testen	Am Ende	
Resultat		

Tab. 39: Beschreibung Non-Functional Requirement 9

4 Systemanalyse und Architektur

5 Qualitätssicherung

6 Implementation

7 Fazit

Kapitel II

Projekt Dokumentation

8 Projekt Plan

8.1 Zusammenarbeit

Als Projektmanagement-Methode wird Kanban verwendet welches es erlaubt das Projekt agil durchzuführen. Aufgrund der Teamgrösse von zwei Personen wurde sich gegen Scrum entschieden. Der zusätzliche organisatorische Mehraufwand durch Daily Meetings, Sprint Reviews und weitere Scrum Meetings würde keinen grossen Nutzen bringen. Dank der geringen Teamgrösse ist es kein Problem den Überblick über das Projekt zu behalten, sodass eine schlanke Methode wie Kanban eine passende Wahl ist.

Die Zusammenarbeit und Aufgabenaufteilung wird mittels Jira organisiert.

8.2 Meetings

Jede Woche findet ein Meeting unter den Teammitgliedern und ein Meeting mit dem Referenten statt.

Im Team-Meeting wird besprochen, welche Aufgaben fertiggestellt wurden und diese gegebenenfalls zusammen anschaut. Zudem werden die Aufgaben für kommende Woche verteilt.

Beim Meeting mit dem Referenten wird der aktuelle Stand besprochen. Ausserdem kann dieses genutzt werden, um allfällige Fragen zu stellen.

8.3 Minimum Viable Product (MVP)

Das Minimum Viable Product (MVP) beinhaltet die Kernfunktionen des Projektes. Diese sind folgendermassen definiert:

- **Erweiterung Discord-Bot:** Ziel ist es, den bestehenden Discord-Bot basierend auf dem während des Einführungsprozesses für den Verein EGSwiss erhaltenen Feedback gezielt weiterzuentwickeln. Dabei sollen Funktionen neu erstellt, ergänzt und verbessert werden.
- **Erweiterung Gamification:** Ziel ist es, das bestehende Gamification-System auszubauen und zu untersuchen, welche Erweiterungen die Motivation und das Engagement im Verein steigern können.
- **Plattform-Unabhängigkeit:** Ein Admin-Dashboard soll erstellt werden, welches dem Vorstand Anpassungen an bereits erfassten Daten ermöglicht, ohne direkten Zugriff auf die Datenbank. Das Admin-Dashboard soll unabhängig von Discord nutzbar sein und somit die Applikation plattformunabhängig machen.

8.4 Long-Term Plan

Der Long-Term Plan wurde in 5 verschiedenen Phasen aufgeteilt. Diese werden in der folgenden Tabelle genauer beschrieben. Optionale Aufgaben stehen in Klammern.

Phase	Woche	Datum	Ziele	Meilenstein
Inception	1	15.09 - 21.09.	Einrichtung, MVP definieren	
	2	22.09. - 28.09.	Projektplan, Risikomanagement, Long-Term-Plan	M1
Elaboration	3	29.09. - 05.10.	Requirements, Einleitung, Stand der Technik	
	4	06.10. - 12.10.	Interviews andere Vereine, Sicherheitskonzept	
	5	13.10. - 19.10.	Gamificationkonzept, Architektur, Mockups GUI	
	6	20.10. - 26.10.	Testkonzept, Prototyp Admin-Dashboard (Proof of Concept)	M2
Construction	7	27.10. - 02.11.	Refactoring, 1.1, 1.2, 7.1	
	8	03.11. - 09.11.	2.1, 2.2, 2.3, 3.1, 3.2, 3.3	
	9	10.11. - 16.11.	4.1, 4.2, 6.1, 6.2, 7.2	M3
	10	17.11. - 23.11.	4.3, 4.4, (5.1), (5.2), (6.3), 7.3	
	11	24.11. - 30.11.	(7.4), 8.1, 8.2, (8.3)	
	12	01.12. - 07.12.	Feedback aus Usability Test umsetzen und Implementation beenden	M4
Transition	13	08.12. - 14.12.	Dokumentation, Abschluss	
	14	15.12. - 19.12.	Fertigstellung, Abgabe Abstract und Dokumentation	M5
Presentation	15	20.12. - 28.12.	Präsentation und A0-Poster	
	16	29.12.2025 - 04.01.2026		
	17	05.01. - 09.01.		M6

Tab. 40: Long-Term Plan

Die einzelnen Komponenten und Funktionen sind wie folgt definiert:

1. Backend

- 1.1: Feedback / Neue Features ergänzen
- 1.2: Auto. Zahlungsbestätigung über Bank

2. Datenbank

- 2.1: Schema anpassen
- 2.2: DB-Login verbessern (2FA)
- 2.3: DB-Transactions / Rollbacks

3. Discord-Bot

- 3.1: Feedback / Neue Features ergänzen
- 3.2: Darstellung Events verbessern
- 3.3: Logs einsehen

4. Admin-Dashboard

- 4.1: Login erstellen
- 4.2: Darstellung / Bearbeitung DB-Daten (Mitglieder, Rollen, Events, Logs)
- 4.3: Erstellung des Punktesystems (Punkte, Belohnungen, Rangliste)
- 4.4: Darstellung / Bearbeitung für Mitglieder

5. Neue Plattform (optional)

- 5.1: Wahl der Plattform
- 5.2: Implementierung gleicher Features wie Discord-Bot

6. Unit-Tests

- **6.1:** Discord-Bot
- **6.2:** Dashboard
- **6.3:** Neue Plattform

7. Pipelines CI/CD

- **7.1:** Dokumentation
- **7.2:** Discord-Bot
- **7.3:** Dashboard
- **7.4:** Neue Plattform

8. Usability Tests

- **8.1:** Discord-Bot
- **8.2:** Dashboard
- **8.3:** Neue Plattform

8.4.1 Inception (Initiierung)

In der Inception-Phase werden die Anforderungen für das Projekt und der Projektplan erstellt, sowie die Risiken identifiziert. In dieser Phase wird zudem die generelle Herangehensweise und die Machbarkeit des Projektes bestimmt.

8.4.2 Elaboration (Ausarbeitung)

In der Elaboration-Phase werden die Anforderungen des Projektes im Detail analysiert. Hier wird das Gamification- und Sicherheitskonzept ausgearbeitet. Die Architektur wird mittels des C4-Modells definiert und ein erster Softwareprototyp wird erstellt. Dieser wird verwendet, um erste Systeme zu testen, auf denen dann später aufgebaut wird.

8.4.3 Construction (Konstruktion)

Die Construction-Phase wird dazu verwendet das Projekt technisch umzusetzen. In dieser Phase werden die geplanten Features umgesetzt und systematisch getestet. Usability-Tests werden durchgeführt auf dessen Basis Anpassungen vorgenommen werden. Die umgesetzten Features und Tests werden dokumentiert.

8.4.4 Transition (Übergang)

In der Transition-Phase wird das Projekt abgeschlossen. Die Dokumentation wird fertiggestellt und der Abstract wird geschrieben. Die Dokumentation wird in dieser Phase abgegeben.

8.4.5 Presentation (Präsentation)

Da das Projekt neben einer Studienarbeit auch eine Bachelorarbeit ist, wird in der Presentation-Phase die Präsentation erstellt. In dieser Phase arbeitet Vanessa Alves, welche die Bachelorarbeit durchführt, an der Präsentation und am Poster alleine.

8.4.6 Meilensteine

Die folgenden Meilensteine dienen als Hilfe, um die definierten Ziele des Projektes zu erreichen. Sie gelten als erfüllt, sobald die einzelnen Komponenten durch das Team in Jira als erledigt gekennzeichnet und vom Referenten eingesehen wurden.

M1: Projekt Initiierung

- Der Projektplan ist definiert.
- Die Risiken sind definiert.
- Das MVP ist definiert.
- Der Long-Term-Plan ist definiert.

M2: Prototyp

- Die Requirements sind definiert.
- Die Architektur ist definiert.
- Mockups für das Admin-Dashboard sind erstellt.
- Das Gamificationkonzept, Datenschutzkonzept und Testkonzept ist erstellt.
- Der erste Prototyp für das Admin-Dashboard ist erstellt.

M3: Erreichung des MVP

- Die Discord-Bot Erweiterungen sind umgesetzt.
- Das Admin-Dashboard ist umgesetzt.
- Das MVP wird erreicht.

M4: Fertigstellung der Implementation

- Alle geplanten, nicht-optionalen Features sind umgesetzt.
- Alle Funktionen sind erfolgreich getestet.
- Usability-Tests sind durchgeführt und Feedback ist umgesetzt.
- Die Testabdeckung entspricht dem definiertem Wert.

M5: Abgabe

- Die umfassende Dokumentation ist fertiggestellt.
- Das Abstract ist fertiggestellt.

M6: Präsentation

- Die Präsentation ist fertiggestellt.
- Das Poster ist fertiggestellt.

8.5 Risikomanagement

Die folgenden Tabellen zeigen die Risiken des Projektes auf und die Strategien die verwendet werden, um diese zu vermindern.

Wahrscheinlichkeit / Schweregrad	1 - Sehr Unwahrscheinlich	2 - Unwahrscheinlich	3 - Gelegentlich	4 - Wahrscheinlich	5 - Oft
4 - Katastrophal	R2				
3 - Kritisch		R3, R7	R9, R11		
2 - Signifikant			R4, R8, R13	R1, R12	
1 - Gering		R5		R6, R14	

Tab. 41: Risiko Matrix

ID	R1
Risiko	Lernkurve
Kommentar	Der Einsatz neuer Technologien (z. B. Frameworks, CI/CD, Web-Technologien) führt zu einer erhöhten Einarbeitungszeit.
Prävention	Frühzeitige Prototypen reduzieren die Einarbeitungszeit. Wissen wird im Team dokumentiert und geteilt.
Korrektur	Komplexe Features werden in kleinere Arbeitspakete unterteilt. Bei Problemen erfolgt gegenseitige Unterstützung im Team.

Tab. 42: Beschreibung Risiko 1

ID	R2
Risiko	Probleme bei Discord oder der neuen Plattformen
Kommentar	Discord oder alternative Plattformen (z. B. MS Teams, Matrix) können temporär nicht verfügbar sein.
Prävention	Evaluierung alternativer Plattformen und Absicherung kritischer Funktionen über die Web-Applikation.
Korrektur	Risiko muss weitgehend akzeptiert werden. Bei längeren Ausfällen wird die Nutzung der Web-Applikation verstärkt.

Tab. 43: Beschreibung Risiko 2

ID	R3
Risiko	Ressourcen limitiert
Kommentar	Ein Teammitglied fällt temporär oder dauerhaft aus.
Prävention	Alle Arbeitsschritte und Entscheidungen werden systematisch dokumentiert, um Wissenstransfer sicherzustellen.
Korrektur	Aufgaben werden durch verbleibende Teammitglieder übernommen. Fokus liegt auf priorisierten Features, um Projektziele trotz reduzierter Kapazität zu erreichen.

Tab. 44: Beschreibung Risiko 3

ID	R4
Risiko	Scope Creep
Kommentar	Anforderungen werden unklar definiert oder nachträglich erweitert, was den Projektumfang vergrößert.
Prävention	Detaillierte Anforderungsdefinition und Scope-Management im Rahmen der Meetings.
Korrektur	Nachträglich eingebrachte oder optionale Anforderungen mit niedriger Priorität werden in spätere Phasen verschoben oder gestrichen.

Tab. 45: Beschreibung Risiko 4

ID	R5
Risiko	Kompatibilitätsprobleme
Kommentar	Unterschiedliche Versionen von Tools, Bibliotheken oder Frameworks können zu Integrationsproblemen führen.
Prävention	Einheitliche Dokumentation der eingesetzten Versionen und Verwendung von Lockfiles (z. B. package-lock.json).
Korrektur	Versionskonflikte werden durch Abstimmung im Team gelöst, eine einheitliche Version wird verbindlich definiert.

Tab. 46: Beschreibung Risiko 5

ID	R6
Risiko	Zeiteinschätzung
Kommentar	Arbeitsaufwände werden zu optimistisch oder pessimistisch eingeschätzt.
Prävention	Iterative Abschätzung in den wöchentlichen Meetings sowie Nutzung von Erfahrungswerten und Vergleich mit vorherigen Projekten.
Korrektur	Optional geplante Features werden verschoben oder gestrichen, um Kernziele nicht zu gefährden.

Tab. 47: Beschreibung Risiko 6

ID	R7
Risiko	Änderungen in der Discord API oder andere Technologien
Kommentar	Fundamentale Änderungen in der Discord API oder anderen zentralen Technologien (z. B. Datenbank, Framework).
Prävention	Keine Verwendung von Funktionen, die als „deprecated“ markiert sind und Monitoring von Release Notes.
Korrektur	Anpassung des Codes und Refactoring bei API-Änderungen, auch wenn zusätzlicher Aufwand entsteht.

Tab. 48: Beschreibung Risiko 7

ID	R8
Risiko	Der bestehende Code ist stark an Discord gekoppelt, was die Entwicklung einer Web-App erschwert.
Kommentar	Der bisherige Code ist nicht komplett von Discord entkoppelt.
Prävention	Frühzeitiges Refactoring und Einführung einer modularen Architektur.
Korrektur	Falls Aufwand höher als geplant, Anpassung des Zeitplans und Reduktion optionaler Features.

Tab. 49: Beschreibung Risiko 8

ID	R9
Risiko	Dashboard ist nicht benutzerfreundlich
Kommentar	Dashboard erfüllen nicht die Erwartungen der Mitglieder/Vorstände hinsichtlich Usability.
Prävention	Entwicklung mit Usability-Tests und Einbindung von Stakeholdern.
Korrektur	Anpassung der Benutzeroberfläche und Interaktionskonzepte anhand von Feedback.

Tab. 50: Beschreibung Risiko 9

ID	R10
Risiko	Manipulation Gamification-System
Kommentar	Mitglieder könnten das Punktesystem missbrauchen (z. B. durch Mehrfachaktionen).
Prävention	Definition klarer Regeln, technische Prüfungen gegen Mehrfacheinträge, Monitoring verdächtiger Aktivitäten durch den Vorstand.
Korrektur	Rücksetzen oder Anpassung von Punkten bei Missbrauch und Anpassung der Regeln.

Tab. 51: Beschreibung Risiko 10

ID	R11
Risiko	Datenschutz
Kommentar	Potenzielle Risiken im Umgang mit personenbezogenen Daten.
Prävention	Erstellung eines Datenschutzkonzepts in Anlehnung an DSGVO und Schweizer Datenschutzgesetz sowie die Minimierung der erhobenen Daten.
Korrektur	Einhaltung von Einverständniserklärungen.

Tab. 52: Beschreibung Risiko 11

ID	R12
Risiko	Stakeholder-Anforderungen nicht berücksichtigt
Kommentar	Feedback von Mitgliedern oder Vorstand wird nicht ausreichend eingebunden.
Prävention	Priorisierung der Anforderungen, Usability-Tests mit realen Mitgliedern.
Korrektur	Anpassungen anhand von Feedback, Verschiebung von Features niedriger Priorität.

Tab. 53: Beschreibung Risiko 12

ID	R13
Risiko	Plattform-Integration komplexer als erwartet
Kommentar	Die geplante Entkopplung auf Dashboard oder anderen Plattformen verursacht höheren Aufwand oder technische Probleme.
Prävention	Proof-of-Concept frühzeitig erstellen, modulare Architektur.
Korrektur	Fokus auf Admin-Dashboard, weitere Plattformintegration in spätere Phase verschieben.

Tab. 54: Beschreibung Risiko 13

ID	R14
Risiko	Hosting oder CI/CD Ausfall
Kommentar	Server oder CI/CD-Pipeline fallen aus und blockieren Entwicklung oder Betrieb.
Prävention	Monitoring und automatisierte Backups, klare Verantwortlichkeiten im Team.
Korrektur	Deployment auf alternative Cloud-Umgebung oder manuelles Deployment als Notfalllösung.

Tab. 55: Beschreibung Risiko 14

8.5.1 Ausweichplan

Der Ausweichplan dient dazu systematisch auf Risiken zu reagieren, die im Vorfeld nicht bestimmt werden konnten. Der Ablauf ist wie folgt definiert:

1. Das Problem innerhalb einer Aufgabe wird identifiziert und mit dem Teammitglied besprochen.
2. Es wird versucht das Problem so schnell wie möglich zu lösen.
3. Kann das Problem nicht gelöst werden, wird es im nächsten Meeting mit dem Referenten besprochen.

Ziel dieses Ausweichplans ist es, bei auftretenden Problemen, die den Projektverlauf beeinträchtigen, eine klare und strukturierte Vorgehensweise zu gewährleisten. Dadurch kann effizient reagiert und ein reibungsloser Projektfortschritt sichergestellt werden.

9 Arbeitszeitnachweis

10 Tooling und Technologie-Entscheidungen

Dieses Kapitel beschreibt die im Rahmen der Bachelorarbeit eingesetzten Werkzeuge, Technologien und Entwicklungsprozesse. Es begründet die wesentlichen Entscheidungen hinsichtlich Projektorganisation, Dokumentation, Code-Qualität und Deployment. Grundlage bilden die Erfahrungen aus dem Vorprojekt, welche in dieser Arbeit systematisch weiterentwickelt werden.

10.1 Jira

Die Zusammenarbeit und Aufgabenverteilung wird mit Jira¹ verwaltet. Für jede Aufgabe wird im Kanban Board ein Issue erstellt mit den folgenden Merkmalen:

- **Epic:** Die Phase zu dem das Issue gehört.
- **Typ:** Der Typ des Issues (Meeting, Aufgabe).
- **Geschätzte Zeit:** Die Soll-Zeit des Issues.
- **Verwendete Zeit:** Die tatsächlich aufgewendete Zeit um, den Issue zu beenden.
- **Zugewiesene Person:** Die Person, die das Issue durchführt.

Die Issues werden alle zuerst im Backlog definiert. Im wöchentlichen Team-Meeting werden die Issues, welche in dieser Woche geplant sind in die „Todo“-Spalte geschoben. Wenn die Person, der das Issue zugewiesen wurde, mit der Arbeit beginnt, wird dieser in die „In Progress“-Spalte geschoben. Wurde das Issue umgesetzt, kommt er in die „Review“-Spalte. Alle Issues in der Review Spalte werden im wöchentlichen Meeting besprochen und bei einer erfolgreichen Umsetzung in die „Done“-Spalte verschoben. Falls es noch Verbesserungspotenzial gibt, kommt das Issue zurück in die „Todo“-Spalte.

10.2 Teams

Für die interne Kommunikation, spontane Absprachen und die Planung von ausserordentlichen Meetings wird Microsoft Teams² verwendet. Es dient als zentraler Kommunikationskanal und Speicherort für geteilte Dokumente. Damit wird ein kontinuierlicher Informationsfluss sichergestellt, was insbesondere bei verteilter Arbeit im Rahmen der Bachelorarbeit entscheidend ist.

10.3 GitHub

Das Vorprojekt wurde zu Beginn der Arbeit von GitLab³ auf GitHub⁴ migriert, um die Entwicklungs- und Veröffentlichungsprozesse langfristig unabhängig von der OST-internen Infrastruktur zu gestalten. Die zuvor auf der OST-GitLab gehostete Umgebung war nur innerhalb der Hochschule zugänglich, wodurch externe Zusammenarbeit und eine spätere Öffnung des Projekts erschwert wurden. GitHub bietet im Gegensatz dazu eine öffentlich zugängliche, weit verbreitete Plattform, die sich in der Open-Source-Community etabliert hat. Durch die Nutzung von GitHub Organisations und integrierten CI/CD-Workflows (GitHub Actions) kann die Projektstruktur klar gegliedert und zukünftige Erweiterungen effizient verwaltet werden. Neben diesen Vorteilen diente der Wechsel ausserdem dem Erkenntnisgewinn für das Team.

Für die Projektstruktur wurde eine eigene Organisation namens „ClansCore“⁵ erstellt. Innerhalb dieser Organisation befinden sich die nachfolgenden Repositories:

Repository	Beschreibung
clanscore ⁶	Der Quellcode der Applikation.
clanscore-dok ⁷	Die Projek-Dokumentation.
clanscore-dok-pub ⁸	Die automatisch generierten PDF-Versionen der Dokumentation und Sitzungs-Protokolle, welche über GitHub Pages veröffentlicht werden (siehe Abschnitt 10.4.2).

¹<https://www.atlassian.com/software/jira>

²<https://www.microsoft.com/de-ch/microsoft-teams/log-in?market=ch>

³<https://gitlab.ost.ch/>

⁴<https://github.com/>

⁵<https://github.com/ClansCore>

⁶<https://github.com/ClansCore/clanscore>

⁷<https://github.com/ClansCore/clanscore-doc>

Tab. 56: Übersicht der Repositories innerhalb der GitHub-Organisation ClansCore

Während der Bearbeitung dieser Arbeit sind die Haupt-Repositories privat gestellt, um den Entwicklungsprozess geschützt zu halten. Die Organisation selbst ist jedoch öffentlich, wodurch Transparenz, Auffindbarkeit und spätere Zusammenarbeit gewährleistet bleiben.

10.3.1 GitHub Guidelines

1. Auf dem Dokumentations-Repository hat jedes Teammitglied seinen eigenen Branch im Format „dev-Initialen“ (z.B. dev-jt)
2. Auf den Code-Repositories (Discord-Bot, Admin-Dashboard) soll für jedes Feature oder Fehler ein jeweiliger Branch im Format „feature-FeatureName“ oder „bug-BugName“ erstellt werden.
3. Das Deployment der Repositories wird je auf einem eigenen Branch namens „deploy“ umgesetzt. Die Sammlung der Commits werden beim mergen des Main-Branches mithilfe der Funktion „Squash“ zusammengefasst, um die Anzahl ähnlicher Commit-Nachrichten zu reduzieren.
4. Bevor Änderungen beim Code in den Develop-Branch gemerged werden, müssen diese von einem anderen Teammitglied überprüft werden.
5. Schreibe Commit-Nachrichten auf Englisch.
6. Beim wöchentlichen Meeting werden die Änderungen vom Develop-Branch in den Main-Branch gemerged.

10.4 Dokumentation

Die Projektdokumentation wird vollständig mit Typst⁹ erstellt. Die Entscheidung für Typst erfolgte aufgrund der einfacheren Syntax im Vergleich zu Latex¹⁰, der nativen Git-Integration und der bereits im Vorprojekt begonnenen Typst-Vorlage.

10.4.1 Dokumentation Guidelines

1. Dokumentiere auf Deutsch.
2. Dokumentiere mit Typst.
3. Verwende die während diesem Projekt entwickelte Vorlage.
4. Unterteile den Text in eine Typst-Datei pro grösseres Kapitel.
5. Benenne die einzelnen Typst Dateien in Englisch und verwende hierfür kebab-case.
6. Verwende aussagekräftige Titel, welche den Text gut unterteilen.
7. Versehe alle nötigen Titel mit einer kurzen und klaren Referenz-Variable im snake_case.
8. Verwende eine saubere Formatierung, die dabei, hilft den Text gut lesen zu können.
9. Kennzeichne Referenzen aus externen Quellen als solche.
10. Schreibe für jede Abbildung und Tabelle eine Beschriftung.
11. Schreibe Referenz-Namen der Beschriftungen für Abbildungen und Tabellen im kebab-case.

10.4.2 Dokumentation Deployment

Im Rahmen dieser Arbeit wurde eine automatisierte Dokumentations-Pipeline implementiert, welche die kontinuierliche Erstellung und Veröffentlichung der Projektdokumentation gewährleistet. Ziel war es, den zuvor im Vorprojekt auf GitLab basierenden CI/CD-Prozess auf die GitHub-Infrastruktur zu migrieren und an die Anforderungen des neuen Projektsetups anzupassen (siehe Abschnitt 10.3).

Die Pipeline dient der automatischen Kompilierung und Veröffentlichung der Typst-Dokumentation, bestehend aus dem Hauptbericht und den Sitzungsprotokollen. Dabei werden sämtliche Schritte, vom Erstellen der PDF-Dateien bis zur Bereitstellung auf GitHub Pages, automatisiert ausgeführt. Da GitHub Pages für private Repositories in der kostenfreien Version nicht verfügbar ist, wurde eine Split-Repository-Strategie umgesetzt. Das private Repository (clanscore-doc) enthält den vollständigen Typst-Quellcode, während ein separates öffentliches Ziel-Repository (clanscore-doc-pub) ausschliesslich die generierten Ausgabedateien beherbergt.

⁸<https://github.com/ClansCore/clanscore-doc-pub>

⁹<https://typst.app/>

¹⁰<https://www.latex-project.org/>

Die Pipeline wurde mittels GitHub Actions realisiert und befindet sich im privaten Repository. Der Workflow wird bei jedem Push auf den Main-Branch ausgelöst. Für die Authentifizierung und das Deployment wurde eine eigene GitHub App innerhalb der Organisation ClansCore erstellt. Diese App besitzt ausschliesslich die für das Deployment benötigten Berechtigungen auf das Ziel-Repository. Das App-Zugriffstoken wird zur Laufzeit über eine Action generiert und anschliessend verwendet, um den Inhalt des public-/Ordners in den Main-Branch des öffentlichen Repositories zu pushen.

Die GitHub Pages des Ziel-Repositories wurden so konfiguriert, dass der Main-Branch automatisch als statische Website veröffentlicht wird. Damit wird bei jeder Änderung im privaten Repository die Dokumentation neu erzeugt und unmittelbar online verfügbar gemacht.

Dieser Ansatz stellt eine klare und sichere CI/CD-Lösung dar. Der Quellcode und interne Inhalte bleiben privat, während die erzeugten Enddokumente öffentlich zugänglich sind. Durch die Verwendung einer GitHub App wird zudem auf persönliche Zugriffstokens verzichtet, was eine nachhaltige und wartungsarme Authentifizierung gewährleistet.

10.5 Code und Entwicklung

Der Code für dieses Projekt wird in der hier beschriebenen Form erstellt.

10.5.1 Code Guidelines

1. Schreibe TypeScript Variablen- und Funktionsnamen im camelCase.
2. Schreibe TypeScript Konstanten in Grossbuchstaben, wobei einzelne Wörter mit Unterstrichen aufgeteilt werden.
3. Schreibe HTML-Element-Bezeichner im kebab-case.
4. Gib Arrays einen Namen im Plural.
5. Schreibe kurze Funktionen mit wenigen Argumenten. Eine Funktion soll nur einen Zweck erfüllen.
6. Verwende let oder const anstelle von var.
7. Schreibe Vergleiche mit === oder !==.
8. Verwende bei Variablen-Zuweisungen Leerzeichen zwischen dem Gleichheitszeichen.
9. Schreibe bei Funktionsheadern die geschweifte Klammer auf die gleiche Zeile.
10. Schreibe Kommentare nur wo es nötig ist, der Code sollte selbsterklärend geschrieben werden.
11. Verwende für gleiche Konzepte die gleichen Wörter.
12. Überprüfe Änderungen vor dem pushen mit Prettier und ESLint.

10.5.2 Setup

Um den Coding-Style einzuhalten werden die Standardregeln von Prettier und ESLint für TypeScript verwendet. Zudem sind bei den Coding Projekten bestimmte Prozesse automatisiert, um die Entwicklung zu erleichtern.

10.5.3 Hosting

Um ClansCore dauerhaft online zu halten und beispielsweise auf Discord-Ereignisse zu reagieren oder zuverlässigen Zugriff auf das Admin-Dashboard zu gewährleisten, ist ein geeignetes Hosting erforderlich. Während dem Vorprojekt wurde der Discord-Bot auf einem Linux-Server des Vereins EGSwiss gehostet, durch manuelle Deployments über den Vereinsinternen Serveradmin. Da dieser Serveradmin zum Zeitpunkt der Arbeit anderweitig beschäftigt war sowie die Teammitglieder keinen direkten Zugriff auf den Server hatten, fiel die Entscheidung auf einen temporären virtuellen Server der Fachhochschule OST. Dieser läuft in einer stabilen Umgebung, ist kostenfrei und garantiert dem Team vollen Zugriff für die Einführung eines automatischen Deployments (siehe Abschnitt 10.5.4).

10.5.4 Code Deployment

Im Vorprojekt wurde das Deployment mit Docker manuell durch eine Drittperson auf einem dedizierten Linux-Server durchgeführt. Da ein manuelles Deployment fehleranfällig, langsam und ineffizient ist, wird ein automatisiertes Deployment umzusetzen (AutoMQ-Contributors, 2025).

Mit GitHub Actions kann das Deployment automatisiert und sichergestellt werden, dass bei einem Push auf den Main-Branch die Änderungen in die Produktivumgebung übernommen werden. Docker wird für das Containermanagement verwendet,

da es eine zuverlässige und plattformunabhängige Bereitstellung der Anwendung ermöglicht ([Docker-Contributors, 2025](#)) und das Team bereits Erfahrung mit Docker hat.

10.6 Programmiersprachen und Frameworks

ClansCore baut auf einem bereits im [Vorprojekt](#) entwickelten Discord-Bot auf. Dieser wurde in TypeScript programmiert und verwendet die Bibliothek Discord.js, welche als die am weitesten verbreitete und aktiv gepflegte Schnittstelle zur Discord API gilt ([Discord.js-Contributors, 2025a](#)). Die Bibliothek abstrahiert die Kommunikation über REST- und WebSocket-Schnittstellen und ermöglicht dadurch eine effiziente und stabile Implementierung der Bot-Funktionalitäten.

Discord.js wird fortlaufend weiterentwickelt und erhält regelmässig sicherheits- und funktionsbezogene Updates ([Discord.js-Contributors, 2025b](#)). Durch die positiven Erfahrungen aus dem Vorprojekt, den aktiven Community-Support und die hohe Kompatibilität mit TypeScript, wird Discord.js weiterhin als technologische Basis verwendet. Diese Entscheidung stellt sicher, dass bestehende Codebestandteile übernommen und nachhaltig weiterentwickelt werden können, ohne eine kostspielige Migration auf eine alternative Bibliothek durchführen zu müssen.

Für die Entwicklung des Admin-Dashboards standen verschiedene moderne Web-Technologien zur Auswahl. Da ein Framework den Entwicklungsprozess strukturiert, die Wiederverwendbarkeit erhöht sowie die Wartung erleichtert, wurde ein komponentenbasiertes Framework eingesetzt. Die Auswahl erfolgte anhand von Kriterien wie Verbreitung, Funktionsumfang, Zukunftssicherheit und vorhandene Teamerfahrung.

Web-Technologie	Beschreibung
React	Eine weit verbreitete JavaScript-Bibliothek für User Interfaces mit Millionen von Anwendern weltweit. React ist komponentenbasiert, kombiniert Logik und Markup in derselben Datei und profitiert von einer grossen Entwickler-Community. Die Flexibilität der Bibliothek ermöglicht unterschiedliche Architekturansätze, erfordert jedoch zusätzliche Tools für Routing, State-Management und Build-Prozesse. (React-Contributors, 2025)
Angular	Ein umfassendes Web-Framework, das eine vollständige Entwicklungsumgebung für skalierbare Web-Applikationen bietet. Angular wird von Google gepflegt, unterstützt TypeScript nativ und beinhaltet bereits Funktionen wie Routing, Dependency Injection und Formularverwaltung. Es ermöglicht eine klare Strukturierung und eine nachhaltige Code-Architektur. (Angular-Contributors, 2025)
Vue.js	Ein progressives JavaScript-Framework zur Entwicklung von User Interfaces. Vue kombiniert einfache Syntax mit hoher Flexibilität und eignet sich sowohl für kleine als auch komplexe Anwendungen. Es besitzt eine engagierte Community, wird jedoch primär von Einzelentwicklern und kleineren Teams gepflegt. (Vue.js-Contributors, 2025)

Tab. 57: Beschreibung der evaluierten Web-Technologien

Nach einer systematischen Bewertung wurde entschieden, das Admin-Dashboard mit Angular zu entwickeln. Diese Entscheidung beruht auf mehreren Faktoren:

- **TypeScript-Integration:** Angular ist vollständig in TypeScript implementiert, was eine einheitliche Codebasis zwischen Frontend und Backend ermöglicht. Dadurch können Typdefinitionen und Schnittstellen gemeinsam genutzt werden, was die Konsistenz und Fehlersicherheit erhöht.
- **Vollständige Framework-Struktur:** Im Gegensatz zu React, das primär eine UI-Bibliothek darstellt, bietet Angular ein integriertes Framework mit klaren Architekturvorgaben, standardisierten Projektstrukturen und eingebautem Tooling (z. B. Testing, Routing, Dependency Injection). Dies reduziert den Integrationsaufwand externer Bibliotheken und erhöht die Wartbarkeit.
- **Skalierbarkeit und Nachhaltigkeit:** Angular ist auf gross angelegte, modular aufgebaute Anwendungen ausgelegt und gewährleistet damit eine langfristig stabile Codebasis.
- **Teamkompetenz:** Beide Teammitglieder verfügten bereits über Erfahrung im Umgang mit Angular, was den Einarbeitungsaufwand erheblich reduzierte und einen produktiven Entwicklungsstart ermöglichte. Mit React und Vue.js liegen hingegen keine oder nur geringe praktische Erfahrungen vor.

Zusammenfassend wurde Angular aufgrund der klaren Struktur, nativen TypeScript-Unterstützung, hohen Skalierbarkeit und der bestehenden Teamexpertise als geeignetstes Framework für die Umsetzung des Admin-Dashboards bewertet. Diese

Wahl stellt sicher, dass das Dashboard langfristig wartbar bleibt und sich konsistent in die bestehende ClansCore-Systemarchitektur integriert.

10.7 Datenbank

Im [Vorprojekt](#) wurde eine MongoDB-Datenbank verwendet. MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die Daten in JSON-ähnlichen Dokumenten speichert ([MongoDB-Contributors, 2025](#)). Eine zentrale Funktion von MongoDB ist die horizontale Skalierbarkeit, was insbesondere bei einer steigenden Nutzung des Bots einen entscheidenden Vorteil darstellt. Die Erweiterungen welche für ClansCore umgesetzt wurden erforderten keine fundamentalen Änderungen an der Datenbank, sodass kein Anlass bestand, eine alternative Datenbanklösung in Erwägung zu ziehen.

10.8 KI-Tools

KI-Tools wurden eingesetzt, um die Effizienz zu steigern und die Qualität der Ergebnisse zu verbessern. Die Einsatzbereiche umfassen:

- **Technische Entscheidung:** Unterstützung bei der Bewertung von Frameworks, Bibliotheken und Architekturansätzen.
- **Programmierung:** Hilfe bei der Fehlersuche, Optimierung und Codegenerierung.
- **Dokumentation:** Unterstützung beim Strukturieren und Formulieren von Textabschnitten, insbesondere für technische Beschreibungen, Kapitelüberschriften und sprachliche Korrektheit.

Die KI wurde als Assistenzwerkzeug eingesetzt und diente nicht zur automatisierten Generierung vollständiger Inhalte. Alle durch die KI erzeugten Vorschläge wurden kritisch geprüft und bei Bedarf angepasst.

Folgende KI-Tools wurden verwendet:

- OpenAI ChatGPT¹¹
- Google Gemini¹²
- GitHub Copilot¹³

¹¹<https://chatgpt.com/>

¹²<https://gemini.google.com>

¹³<https://github.com/features/copilot>

11 Sitzungsprotokolle

Besprechung vom 19.09.2025

Anwesenheitsliste:	Frieder Loch, Joel Thoma, Vanessa Alves
Autor:	Joel Thoma

Inhalt der Sitzung:

- Aufgabenstellung
 - Was kommt als nächstes
 - Abgabetermine
 - Feedback SA FS25
 - Sonstiges
-

Aufgabenstellung

Ein konkretes Projekt finden. Bei anderen onlinebasierten Vereinen Umfragen und Recherche durchführen um Konzept zu verbessern.

Eine Idee ist ein Admin-Dashboard zu machen um von Discord zu entkoppeln und keine Datenbankzugriffe mehr nötig sind. Eine Anbindung an Microsoft Teams oder die Open Source Plattform Matrix wäre denkbar.

Was kommt als nächstes

Problembeschreibung, Zeitplan und Meilensteine definieren.

Abgabetermine

Wegen der SA von Joel ist der Abgabetermin für die Arbeit am 19. Dezember und Vanessa hat noch für die Präsentation und Poster bis am 9. Januar Zeit.

Feedback SA FS25

Wird noch bereitgestellt.

Sonstiges

Änderungen zur Aufgabenstellung:

Gegenleser: Severin Dellspurger

Koreferent: Julia Czerniak-Wilmes

Besprechung vom 29.09.2025

Anwesenheitsliste:	Frieder Loch, Joel Thoma, Vanessa Alves
Autor:	Joel Thoma

Inhalt der Sitzung:

- Besprechung aktueller Stand
 - ToDo
-

Besprechung aktueller Stand

- Meilensteine könnte man messbar machen (klar definieren wann diese erreicht sind)
- Im MVP sollte klar definiert sein welche Teile neu sind (Nur die Plattform-Unabhängigkeit in das MVP?)

ToDo

- Self-Determination-Theory: über dieses oder ähnliche Frameworks Research betreiben und in das Interview einbauen. (vor allem in Verbindung mit Competence, Autonomy, Relatedness)
- Problemstellung am Anfang sollte folgendes enthalten:
 - die zentralen technischen Probleme finden und beschreiben. Wo könnte es Probleme geben und wieso ist es schwierig. (Kernprobleme, da unterschiedliche Lösungen finden und analysieren) (vlt Synchronisierung, Modularisierung?)
 - Alles Begründen. Alle Handlungsmöglichkeiten anschauen, analysieren und die Entscheidungen dokumentieren.
 - Das Produkt gut verkaufen. Was ist das endgültige Produkt. Gamification und Produkt gut zusammenpacken.

Besprechung vom 03.10.2025

Anwesenheitsliste:	Frieder Loch, Joel Thoma, Vanessa Alves
Autor:	Joel Thoma

Inhalt der Sitzung:

- Besprechung aktueller Stand
-

Besprechung aktueller Stand

- Es wurde der aktuelle Stand geschildert.
- Verlängerung der Elaboration Phase und Verschiebung vom Gamification Konzept

12 Persönliche Erfahrungsberichte

12.1 Bericht Joel Thoma

12.2 Bericht Vanessa Alves

Kapitel III

Glossar und Abkürzungsverzeichnis

ClansCore	ClansCore ist das Gesamt-System und der Produktnname für die Erweiterung des Discord-Bots aus dem <u>Vorprojekt</u> . Es ist das End-Produkt dieser Arbeit und verbindet sämtliche Teil-Komponenten wie Backend, Datenbank, Discord-Bot, Admin-Dashboard und weitere Plattformen.
EGSwiss	Ein Schweizer Gaming Verein, wobei das Teammitglied Vanessa Alves Präsidentin ist und daher die nötige Expertise für die Vorarbeit und aktuelle Weiterentwicklung mitbringt sowie Erfahrungen aus dem Vereinsalltag einfließen lässt.
CI/CD	Continuous Integration (CI) und Continuous Deployment/Delivery (CD). Ein automatisierter Prozess, bei dem Codeänderungen kontinuierlich integriert, getestet und automatisch oder mit manueller Freigabe in die Produktionsumgebung publiziert werden.
REST-Schnittstelle	REST (Representational State Transfer) ist eine Architekturform für webbasierte Schnittstellen, bei der Daten über standardisierte HTTP-Methoden (z. B. GET, POST, PATCH, DELETE) ausgetauscht werden. REST-APIs sind zustandslos, d. h. jede Anfrage enthält alle notwendigen Informationen, um sie unabhängig von früheren Interaktionen zu verarbeiten. In <u>ClansCore</u> wird die Discord REST-API verwendet, um gezielte Aktionen wie das Senden von Nachrichten oder das Verwalten von Rollen durchzuführen.
WebSocket-Schnittstelle	WebSockets sind bidirektionales Kommunikationsprotokolle (RFC 6455), welches eine dauerhafte Verbindung zwischen Client und Server ermöglicht. Im Gegensatz zu REST, das auf einzelnen HTTP-Anfragen basiert, erlaubt WebSocket eine kontinuierliche Übertragung von Daten in Echtzeit. In <u>ClansCore</u> wird über das Discord-Gateway eine WebSocket-Verbindung aufgebaut, um Ereignisse (z. B. neue Nachrichten, Interaktionen oder Mitgliederänderungen) unmittelbar zu empfangen und darauf zu reagieren.

Kapitel IV

Literaturverzeichnis

Alves, V., & Schnider, M. (2025). *Discord-Bot für Vereine*.

Angular-Contributors. (2025,). *What is Angular?*. <https://angular.dev/overview>

Arcane-Contributors. (2025,). *Arcane Discord Bot*. <https://arcane.bot/>

AutoMQ-Contributors. (2025,). *Common Issues with Manual Deployment*. <https://www.automq.com/blog/fully-automated-deployment-vs-manual-deployment-comparison#common-issues-with-manual-deployment>

ClubDesk-Contributors. (2025,). *Vereinssoftware für die einfache Vereinsverwaltung*. <https://www.clubdesk.de/de/startseite>

Discord.js-Contributors. (2025b,). *Commits*. <https://github.com/discordjs/discord.js/commits/main/>

Discord.js-Contributors. (2025a,). *The most popular way to build Discord bots*. <https://discord.js.org/>

Docker-Contributors. (2025,). *Developers bring their ideas to life with Docker*. <https://www.docker.com/why-docker/>

MEE6-Contributors. (2025,). *Statistics Channels Plugin*. <https://wiki.mee6.xyz/en/plugins/statistics-channels>

MongoDB-Contributors. (2025,). *Introduction to MongoDB*. <https://www.mongodb.com/docs/manual/introduction/>

React-Contributors. (2025,). *React*. <https://react.dev/>

Vue.js-Contributors. (2025,). *What is Vue?*. <https://vuejs.org/guide/introduction>

WildApricot-Community. (2015,). *Gamify Member's Site Participation*. <https://forums.wildapricot.com/forums/308932-wishlist/suggestions/10321731-gamify-member-s-site-participation>

WildApricot-Contributors. (2025,). *Membership Management Software*. <https://www.wildapricot.com/>

Kapitel V

Abbildungsverzeichnis

Kapitel VI

Tabellenverzeichnis

Tab. 1 Beschreibung der Prioritäts-Kategorien	6
Tab. 2 Beschreibung der Resultats-Kategorien	6
Tab. 3 Beschreibung Functional Requirement 1	7
Tab. 4 Beschreibung Functional Requirement 2	7
Tab. 5 Beschreibung Functional Requirement 3	7
Tab. 6 Beschreibung Functional Requirement 4	8
Tab. 7 Beschreibung Functional Requirement 5	8
Tab. 8 Beschreibung Functional Requirement 6	8
Tab. 9 Beschreibung Functional Requirement 7	8
Tab. 10 Beschreibung Functional Requirement 8	8
Tab. 11 Beschreibung Functional Requirement 9	8
Tab. 12 Beschreibung Functional Requirement 10	8
Tab. 13 Beschreibung Functional Requirement 11	8
Tab. 14 Beschreibung Functional Requirement 12	9
Tab. 15 Beschreibung Functional Requirement 13	9
Tab. 16 Beschreibung Functional Requirement 14	9
Tab. 17 Beschreibung Functional Requirement 15	9
Tab. 18 Beschreibung Fully dressed Use Case 1	10
Tab. 19 Beschreibung Fully dressed Use Case 2	10
Tab. 20 Beschreibung Fully dressed Use Case 3	11
Tab. 21 Beschreibung Fully dressed Use Case 4	11
Tab. 22 Beschreibung Fully dressed Use Case 5	11
Tab. 23 Beschreibung Fully dressed Use Case 6	12
Tab. 24 Beschreibung Fully dressed Use Case 7	12
Tab. 25 Beschreibung Fully dressed Use Case 8	12

Tab. 26 Beschreibung Fully dressed Use Case 9	13
Tab. 27 Beschreibung Fully dressed Use Case 10	13
Tab. 28 Beschreibung Fully dressed Use Case 11	13
Tab. 29 Beschreibung Fully dressed Use Case 12	14
Tab. 30 Beschreibung Fully dressed Use Case 13	14
Tab. 31 Beschreibung Non-Functional Requirement 1	15
Tab. 32 Beschreibung Non-Functional Requirement 2	15
Tab. 33 Beschreibung Non-Functional Requirement 3	15
Tab. 34 Beschreibung Non-Functional Requirement 4	15
Tab. 35 Beschreibung Non-Functional Requirement 5	15
Tab. 36 Beschreibung Non-Functional Requirement 6	16
Tab. 37 Beschreibung Non-Functional Requirement 7	16
Tab. 38 Beschreibung Non-Functional Requirement 8	16
Tab. 39 Beschreibung Non-Functional Requirement 9	16
Tab. 40 Long-Term Plan	22
Tab. 41 Risiko Matrix	24
Tab. 42 Beschreibung Risiko 1	24
Tab. 43 Beschreibung Risiko 2	25
Tab. 44 Beschreibung Risiko 3	25
Tab. 45 Beschreibung Risiko 4	25
Tab. 46 Beschreibung Risiko 5	25
Tab. 47 Beschreibung Risiko 6	25
Tab. 48 Beschreibung Risiko 7	26
Tab. 49 Beschreibung Risiko 8	26
Tab. 50 Beschreibung Risiko 9	26
Tab. 51 Beschreibung Risiko 10	26
Tab. 52 Beschreibung Risiko 11	26
Tab. 53 Beschreibung Risiko 12	26
Tab. 54 Beschreibung Risiko 13	27

Tab. 55 Beschreibung Risiko 14	27
Tab. 56 Übersicht der Repositories innerhalb der GitHub-Organisation ClansCore	29
Tab. 57 Beschreibung der evaluierten Web-Technologien	32

Kapitel VII

Code-Listings

Kapitel VIII

Anhang