JOHANNES KEPLER
UNIVERSITÄT LINZ

Eingereicht von:
Johannes Hacker
Moritz Neuwirth
Julian Lumetsberger

Gruppe:
Gruppe 4

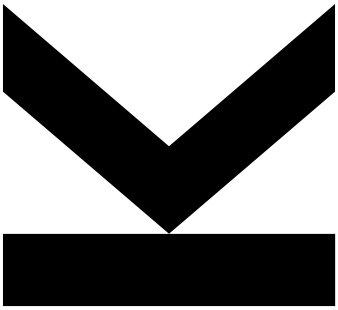
Angefertigt am:
Institut für Wirtschaftsinformatik – Software Engineering

Beurteiler / Beurteilerin:
Dr. Johannes Sametinger

Mitbetreuung
-

Datum
18.07.2023

GO-SPIEL



Projektdokumentation

PR Software Engineering
259.035
SS 2023

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
jku.at
DVR 0093696

JOHANNES KEPLER
UNIVERSITÄT LINZ

Inhaltsverzeichnis

1. Projektübersicht

Projektname: Go-Spiel

Projektziel: Das Ziel dieses Projekts ist es, ein Go-Spiel zu entwickeln, das mithilfe der JavaFX-Bibliothek eine benutzerfreundliche und ansprechende Benutzeroberfläche bietet. Go ist ein strategisches Brettspiel aus Asien, bei dem zwei Spieler abwechselnd Steine auf einem Spielbrett platzieren, um Gebiete zu kontrollieren und Gefangene zu machen. Das Spiel soll sowohl für Anfänger als auch für erfahrene Go-Spieler geeignet sein. Das Spiel soll auch bestehende Spiele laden und gespielte Spiele speichern können. Außerdem sollte man durch Züge navigieren können mit entsprechenden Zugerklärungen.

Projektteam:
- Johannes Hacker
- Moritz Neuwirth
- Julian Lumetsberger

Projektphasen:
- Vorbereitung
- Release 1
- Release 2
- Release 3
- Final Release

Projektziele:
- Entwicklung eines Go-Spiels mit einer benutzerfreundlichen JavaFX-GUI.
- Implementierung einer Spiellogik, die die Regeln des Go-Spiels korrekt umsetzt.
- Speichern und Laden von Spielständen in JSON-Formaten.
- Erstellung einer Dokumentation für Benutzer und Entwickler.

Projektergebnisse:
Das Projekt soll ein funktionsfähiges Go-Spiel liefern, das eine intuitive Benutzeroberfläche bietet und alle grundlegenden Regeln des Spiels implementiert. Die Spieler sollen in der Lage sein, das Spiel zu starten, Steine auf dem Brett zu platzieren, durch Züge zu navigieren und das Spiel zu speichern bzw. zu laden.

Projektrisiken:
Komplexität der Spiellogik: Die korrekte Implementierung der komplexen Regeln des Go-Spiels kann eine Herausforderung darstellen.
UI/UX-Design: Die Gestaltung einer benutzerfreundlichen und ästhetisch ansprechenden Benutzeroberfläche erfordert sorgfältige Planung und Abstimmung.
Zeitliche Einschränkungen: Das Projekt ist zeitlich begrenzt, daher ist eine effiziente Planung und Umsetzung erforderlich, um die gesetzten Ziele zu erreichen.

2. Vorbereitung

Das Projekt wurde in fünf aufeinanderfolgende Projektphasen unterteilt, um einen strukturierten und effizienten Entwicklungsprozess zu gewährleisten. Jede Phase hatte spezifische Ziele und Meilensteine, die das Team erreichen musste, um das Projekt erfolgreich abzuschließen. Die Phasen waren wie folgt:

2.1. Projektplan

In der Vorbereitungsphase setzte sich das Projektteam intensiv mit den eingesetzten Tools und Technologien auseinander. Dazu gehörten die Einarbeitung in JavaFX für die Erstellung der Benutzeroberfläche, die Verwendung von IntelliJ als Entwicklungsumgebung, die Versionierung mit Git und die Zeitverfolgung mit Clockify. Das Team legte außerdem die grundlegende Projektstruktur fest und definierte die Rollen und Verantwortlichkeiten jedes Teammitglieds. Die Vorbereitungsphase diente als Grundlage für einen reibungslosen Start in die Umsetzung des Projekts.

Die Anforderungen des Projektes wurden in 4 Teilbereiche aufgeteilt:

- Funktionale Anforderungen an die GUI (siehe Abb. 1 & 2)
- Funktionale Anforderungen an die Spiellogik (siehe Abb. 3)
- Nicht-funktionale Anforderungen (siehe Abb. 4)
- „Nice to have“-Anforderungen (siehe Abb. 5)

Während in der Vorbereitungsphase noch die Rede von einem Aufteilen in Singleplayer und Multiplayer war, wurde diese Idee im Projektverlauf schnell verworfen. Die Einteilung der Aufgaben erfolgte daher im ersten Releaseplan noch diesem Schema.

Anforderung	Priorität (1-10)	Release (1-3)
Menü zum Einstellen des Spielmodus (SP / MP)	1	1
Button zum Laden der Spieldatei (SP)	5	1
Eingabe der Spielfeldgröße (9x9,13,x13,19x19) (MP)	3	1
Eingabe der Handycap-Steine (MP)	8	1

Abb. 1: Funktionale Anforderungen an die GUI (Menü)

Anforderung	Priorität (1-10)	Release (1-3)
Spielfelder in den gewünschten Dimensionen anzeigen (SP / MP)	3	1
GO-typisches Spielbrett inklusive Handycappunkte (SP / MP)	1	1
Variabel vergrößerbare Darstellung des Spielbretts (SP / MP)	7	1
Anzeige der Steine (SP / MP)	2	1
Spiel speichern Button (exportieren als Datei) (SP / MP)	5	1
Anzeige des Gewinners und der Punkte (nach Spielende) (SP / MP)	6	1
Anzahl der gefangenen Steine anzeigen (SP / MP)	4	1
Züge vor und zurückspringen (SP)	5	1
Erklärung von Zügen Ein- / Ausblenden (SP)	5	1
Aufgeben – Button	9	1
Passen - Button	3	1

Abb. 3: Funktionale Anforderungen an die GUI (Spiel)

Anforderung	Priorität (1-10)	Release (1-3)
Setzen	1	2
Passen	3	2
Schlagen	2	3
Erkennen von Selbstmord (Erlaubt / Nicht erlaubt)	5	3
Feststellung des Siegers	7	3
Erkennung der Punkte jedes Spielers	7	3
Kö Regel	10	FINAL
Spiel protokollieren und Protokoll exportieren (Speichern)	4	3
Spielprotokoll einlesen	4	3
Aufgeben	8	2

Abb. 2: Funktionale Anforderungen an die Logik

Anforderung	Priorität (1-10)	Release (1-3)
Datenerhaltung bei Absturz	6	FINAL
Fehlertoleranz	8	FINAL
Flüssigkeit des Spielbetriebs	5	FINAL
Benutzerfreundliches Design	4	1

Abb. 4: Nicht-funktionale Anforderungen

Anforderung
Computergegner im Multiplayer (Chat GPT Anbindung)
Benutzeranmeldung mit Speicherständen
Turniere und Meisterschaften
Ranglisten
Verschiedene Spielmodi (Blitz, Normal, …)
Bevorzugte Regeln auswählen (Japanisch, Chinesisch, …)
Komi auswählbar
Übergrößen und nicht-quadratische Spielfeldgrößen
Import von Speicherständen eines anderen Datentyps

Abb. 5: „Nice to have“-Anforderungen

2.2. Marktanalyse

Im Zuge der Vorbereitung wurde auch eine Marktanalyse der bereits vorhandenen Go-Spiele durchgeführt. Dabei kam zum Vorschein, dass es bereits etliche kleinere Offline-Projekte gibt in verschiedenen Programmiersprachen, diese jedoch oft nur Grundlegende Spiellogiken wie Setzen und Schlagen implementiert haben, jedoch kein vollständiges Spiel mit allen Regeln darstellen. Es werden auch verschiedene Regelwerke bei den Anbietern unterschieden, wobei die häufigste Variante das japanische, koreanische oder chinesische ist.
Im Online-Bereich sind einige Anbieter vertreten, die sehr ausgereifte Lösungen für Online-Multiplayer Spiele zu Verfügung stellen. Gerade im asiatischen Raum sind auf diesen Plattformen auch sehr viele Spieler vertreten.

2.3. Softwarewerkzeuge

Bereits am Beginn haben wir und festgelegt welche Softwarewerkzeuge (siehe Abb. 6) wir für das Projekt benutzen möchten. Dabei setzten wir grundlegend auf sehr gängige Produkte, die sich schon lange auf dem Markt etabliert haben.

Name	Beschreibung
IntelliJ	IDE, Code coverage
Maven	Build tool
Github	VCS
Scene Builder	JavaFX GUI Builder
Kommunikationstools	Sharepoint, Whatsapp, Zoom, Clockify
SONARQUBE	Codequalitätsprüfung
JUnit	Unit-Tests

Abb. 6: Softwarewerkzeuge

Im Laufe des Projektes gab es bei der Verwendung der Werkzeuge auch nur kleine Änderungen. So haben wir uns für OneDrive anstelle von SharePoint entschieden, da dies von der JKU gehostet wird. Sonarcube zur Codequalitätsprüfung wurde aus zeitlichen Gründen nicht eingesetzt. Als Planungswerkzeug haben wir zusätzlich noch das Kanban-Board (siehe Abb. 7), welches in GitHub Projects integriert ist, verwendet den Status unserer verschiedenen Tasks darzustellen.

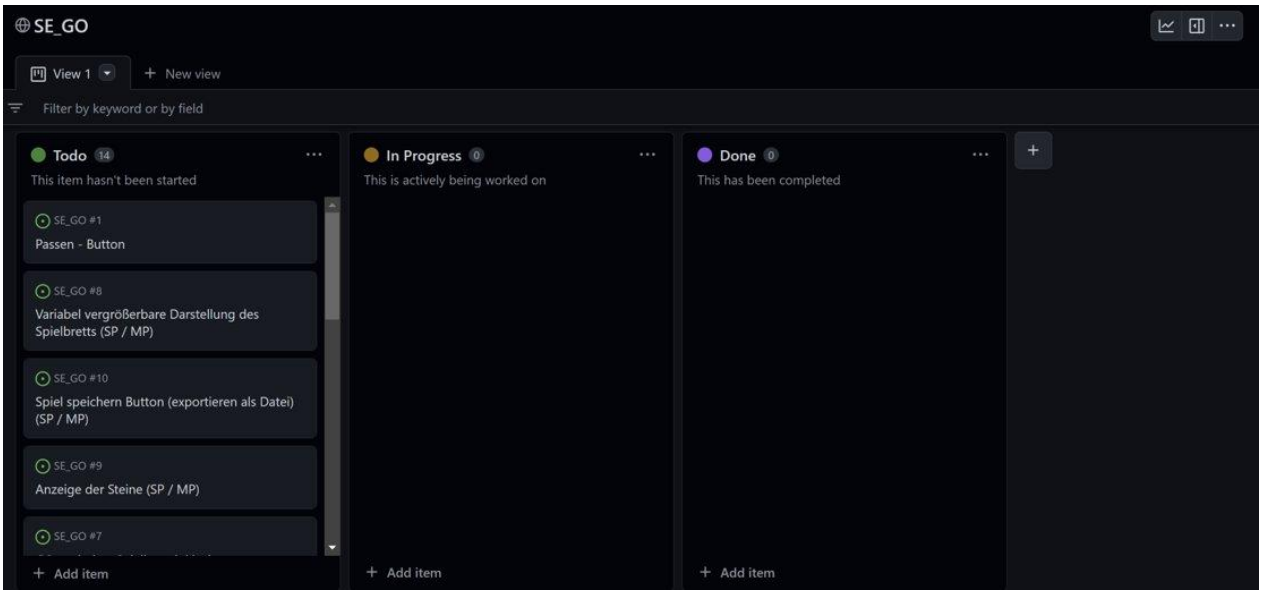


Abb. 7: Kanban-Board

3. Release 1

Im Release 1 lag der Fokus auf der GUI. Diese sollte hier bereits vollständig fertig sein und im Laufe des Projektes nur noch mit kleineren Änderungen im Zuge der Logik-Implementierungen auskommen. Zwar war die GUI anfangs schnell grundlegend fertig gestellt, traten aber auch bereits erste Probleme auf, speziell mit der variablen Größe des Spielfeldes und des Nicht-vorhandenseins der zugehörigen Logik. Daher konnten wir nicht ganz alle geplanten Anforderungen (siehe Abb. 8 & 9) für diesen Release implementieren und mussten diese in den Release 2 verschieben.

Anforderung	Priorität (1-3)
Buttons zur Auswahl des Spielmodus	1
Button zum Laden der Spieldatei	2
Eingabe der Spielfeldgröße (9x9,13x13,19x19)	3
Eingabe der Handicap-Steine	3

Abb. 8: Release 1 Anforderungen – GUI (Menü)

Anforderung	Priorität (1-3)
Spielfelder in den gewünschten Dimensionen anzeigen	1
GO-typisches Spielbrett inklusive Handicappunkte	1
Variabel vergrößerbare Darstellung des Spielbretts	3
Anzeige der Steine	1
Spiel speichern Button (exportieren als Datei)	2
Anzeige des Gewinners und der Punkte (nach Spielende)	2
Anzahl der gefangenen Steine anzeigen	1
Züge vor und zurückspringen	2
Erklärung von Zügen Ein- / Ausblenden	2
Aufgeben – Button	3

Abb. 9: Release 1 Anforderungen

Um die anfängliche Struktur schon einmal grob festzulegen, haben wir uns mit Hilfe von IntelliJ auch ein erstes Klassendiagramm (siehe Abb. 10) erzeugen lassen. Dieses hat sich mit dem Implementieren der Logik in späteren Releases jedoch sehr stark verändert.

Als Fazit für diesen Release lässt sich sagen, dass der anfängliche Rechercheaufwand für JavaFX

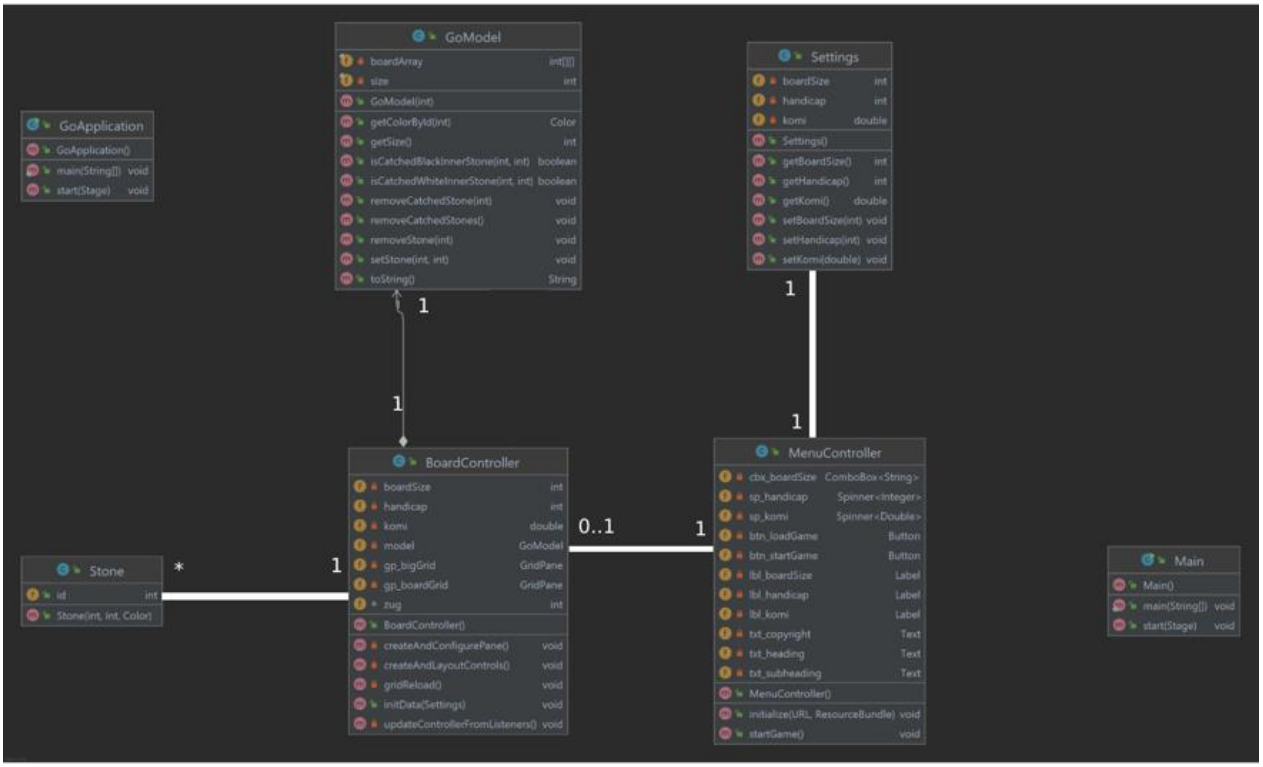


Abb. 10: UML-Klassendiagramm

enorm hoch war. Das FrontEnd-Design mit JavaFX bzw. dem SceneBuilder ist oft mühsam und lässt sich an einigen Stellen nur schwer nachvollziehen. Auch Build-Fehler, die bei der Maven-Synchronisation auftraten, ließen sich oft nur sehr schwer lösen, da die Ursache nicht bekannt war. Generell lässt sich das MVC-Pattern sehr gut für JavaFX-Anwendungen verwenden und die Versionskontrolle mit GitHub und IntelliJ funktionierte nach kleineren anfänglichen Schwierigkeiten auch sehr gut.

4. Release 2

Der Plan für Release 2 bestand auf 6 Anforderungen (siehe Abb. 11), welche für grundlegende Logiken des Spiels notwendig sind. Weiters sollten bereits erste Unit-Tests für die Hauptklassen erstellt werden. Für diese wurde JUnit 5 verwendet. Zu Beginn war der Plan weitere Tests mit einem für JavaFX spezialisierten Test-Framework durchzuführen, dieses war jedoch durch mangelnde Dokumentation und fehlenden Use-Cases für und nicht verwendbar. Weiters kam auch die Frage auf, welche Klassen bei einem JavaFX-Projekt sinnvoll zu testen sind, da Controller und View Zusammenspiel eher schwierig zu testen ist bzw. dabei sehr viele unnötige Testfälle entstehen würden. Nach einer Diskussion mit den anderen Teams wurde sich daher darauf geeinigt, dass es ausreichend ist das Model mit einer möglichst hohen Coverage zu testen. Die Unit-Tests zogen sich von Release 2 bis zum Final Release, da immer wieder Funktionalitäten hinzugefügt wurden, die ebenfalls Tests bedurften.

Die Anforderungen, die aus Release 1 noch in diesen Release mitgenommen wurden, konnten vollständig implementiert werden.

Anforderung	Priorität (1-3)
Aufgeben	3
Schlagen	2
Passen	1
Setzen	1
Unit-Test für Hauptklassen	2
GUI-Issues aus Release 1	1

Abb. 11: Release 2 Anforderungen

Abschließend lässt sich sagen, dass der Release 2 wie geplant ablief und es zu keinen Verzögerungen gekommen ist. Allerdings mussten manch fertig implementierten Funktionalitäten aufgrund von Anforderungen, die in späteren Releases noch hinzukamen, nochmals komplett neu ausprogrammiert werden, um eine zusammenspielendes Gesamtprodukt zu erhalten.

5. Release 3

Im Release 3 standen die erweiterten Funktionalitäten (siehe Abb. 12), sowie die Anpassung der Grundfunktionen auf dem Plan. Im Zuge einer Code Review wurden wir auch noch auf einige Mängel in der Umsetzung des MVC-Patterns hingewiesen, welche wir ebenfalls in diesem Release behoben. Aufgrund der steigenden Komplexität unserer Algorithmen und Logik wurde die Implementierung von neuen Funktionalitäten oftmals eine Herausforderung und kostete uns sehr viel mehr Zeit als ursprünglich geplant. Daher konnten auch mehrere wesentliche Anforderungen in diesem Release nicht fertig oder vorerst nur fehlerhaft eingefügt werden.

Eine große Errungenschaft war das Implementieren eines Protokolls mit dem die gespielten Spiele gespeichert und auch wieder geladen werden konnten. Dieses wurde schließlich immer wieder

erweitert um auch die Beschreibung der Züge, sowie geschlagene Steine und Steingruppen korrekt abspeichern zu können.

Anforderung	Priorität (1-3)
Spiel speichern/Protokoll	2
Spiel laden	2
Unit Tests	1
MVC Pattern sauber implementieren	1
Punkte zählen	1
Selbstmord Erkennung	3
Züge vor- und zurückspringen	2

Abb. 12: Release 3 Anforderungen

Die nicht abgeschlossenen Anforderungen wurden in den Final Release mitgenommen.

6. Final Release

Das wir aus dem Release 3 noch einige größere Funktionalitäten in den Final Release mitgenommen haben, wurde der Zeitplan etwas durcheinandergebracht. Allerdings konnten die erweiterten Funktionalitäten noch rechtzeitig implementiert werden, sodass genügend Zeit für die Erstellung der Dokumentationen zur Verfügung stand. Konkret wurde eine Benutzerdokumentation ausgearbeitet, die den Nutzer der Anwendung bei der Installation und Bedienung anleitet. In der Systemdokumentation wurde Bezug auf die wichtigsten Klassen und deren Funktionalitäten genommen. Die Projektdokumentation beinhaltet die wesentlichen Meilensteine und Planungen des Projektverlaufes. Der Code selber wurde mittels JavaDoc ausreichend beschrieben und auch ein am Ende durchgeführtes Code-Cleanup half wesentlich zur besseren Lesbarkeit.

Zum Abschluss konnten beinahe alle Funktionalitäten implementiert werden, lediglich die Kö-Regel, die eine sehr spezielle Konstellation im Go-Spiel darstellt, konnte aus zeitlichen Gründen nicht mehr umgesetzt werden.

7. Zeitaufzeichnung

Während des gesamten Projekts wurde die Arbeitszeit des Teams mit Hilfe von Clockify erfasst. Insgesamt wurden etwa 350 Stunden für die Entwicklung des Go-Spiels aufgezeichnet. Diese Zeit teilt sich annähernd gleichmäßig auf die verschiedenen Teammitglieder auf und ist ein Indiz einer gelungenen Arbeitsteilung.

Die Verwendung von Clockify erwies sich als äußerst hilfreich und effizient. Clockify ist eine kostenlose Zeiterfassungssoftware, die dem Team ermöglichte, die investierte Zeit genau zu dokumentieren. Die Plattform bot eine benutzerfreundliche Oberfläche und eine übersichtliche Darstellung der erfassten Zeiten. Dies half dem Team, den Überblick über den Projektfortschritt zu behalten und sicherzustellen, dass die Zeit effektiv genutzt wurde.

Während des Projekts gab es Phasen mit unterschiedlichem Arbeitsaufwand, abhängig von den Anforderungen des Teams und dem Stressniveau der Universität. In stressigeren Zeiten, beispielsweise während Prüfungsphasen oder Abgabefristen, konnte die Arbeitszeit für das Projekt reduziert sein, während in ruhigeren Phasen mehr Zeit für die Projektarbeit zur Verfügung stand.

Insgesamt erwies sich die Zeitaufzeichnung als wertvolles Instrument für die Projektsteuerung und half dem Team, das Go-Spielprojekt erfolgreich abzuschließen. Die Möglichkeit, die Zeit zu dokumentieren und auszuwerten, half dabei, die Arbeitsabläufe zu optimieren und das Projektziel effizient zu erreichen.

8. Kommunikation

Die effektive Kommunikation spielte eine entscheidende Rolle für den Erfolg des Projekts. Das Team nutzte verschiedene Kommunikationsmittel, um miteinander in Verbindung zu bleiben und Informationen auszutauschen.

8.1. Meetings

Im Verlauf des Projekts wurden regelmäßige Meetings abgehalten, um den Fortschritt zu besprechen und die nächsten Schritte zu planen. Diese Meetings erwiesen sich als wichtige Kommunikations- und Koordinationsmöglichkeit für das Team.

Die Meetings fanden in regelmäßigen Abständen statt und wurden vor allem vor geplanten Präsentationen durchgeführt. Dies ermöglichte es dem Team, den aktuellen Stand des Projekts zu bewerten und sicherzustellen, dass alle notwendigen Aufgaben erledigt wurden, um eine erfolgreiche Präsentation durchzuführen.

Darüber hinaus wurden auch während der Projektarbeit Meetings abgehalten, um Fragen zu klären, Probleme zu besprechen und gemeinsam Lösungen zu erarbeiten. Diese Meetings halfen dabei, mögliche Engpässe oder Hindernisse frühzeitig zu identifizieren und zu bewältigen.

Während der Meetings wurden auch die Aufgaben für die nächste Phase oder den nächsten Sprint verteilt. Jedes Teammitglied hatte die Möglichkeit, seine Fortschritte zu teilen, Fragen zu stellen und Feedback von anderen Teammitgliedern zu erhalten.

Besonders effektiv war die gemeinsame Erstellung der Präsentationen für die verschiedenen Projektphasen. Jedes Teammitglied trug sein Wissen und seine Erfahrungen bei, um eine umfassende und aussagekräftige Präsentation zu erstellen. Dies förderte nicht nur das Verständnis für das Projekt bei allen Teammitgliedern, sondern half auch dabei, das Projekt nach außen hin gut zu präsentieren.

Insgesamt erwiesen sich die Meetings als eine wertvolle Zeitinvestition, um den Projektfortschritt zu verfolgen, die Zusammenarbeit zu fördern und mögliche Herausforderungen frühzeitig zu bewältigen. Die klare Kommunikation und die regelmäßige Abstimmung ermöglichten es dem Team, das Projekt effizient voranzutreiben und erfolgreich abzuschließen.

8.2. Whatsapp-Gruppe:

Die Teammitglieder kommunizierten regelmäßig über eine Whatsapp-Gruppe. Hier konnten allgemeine Fragen gestellt, Informationen geteilt und schnelle Updates gegeben werden. Die Whatsapp-Gruppe diente als informeller Kanal für die tägliche Kommunikation und half dabei, dringende Angelegenheiten zeitnah zu klären.

8.3. Meetings über Zoom:

Für formelle Besprechungen und Meetings verwendete das Team die Plattform Zoom. Zoom bot eine stabile Video- und Audiokommunikation und ermöglichte es den Teammitgliedern, sich persönlich auszutauschen, auch wenn sie sich nicht physisch am gleichen Ort befanden. Die Meetings wurden vor geplanten Präsentationen, während der Projektarbeit und bei Bedarf abgehalten, um den Fortschritt zu besprechen und Probleme zu lösen.

8.4. Dateimanagement über Onedrive:

Onedrive wurde als zentrale Plattform für das Dateimanagement genutzt. Hier wurden alle Projektdateien, Dokumente, Präsentationen und Code-Snippets gespeichert. Die Verwendung einer gemeinsamen Cloud-Speicherlösung ermöglichte es den Teammitgliedern, auf die benötigten Ressourcen zuzugreifen und Änderungen in Echtzeit zu verfolgen. Dies erleichterte die Zusammenarbeit und verhinderte Versionierungskonflikte.

Die klare Kommunikation über diese verschiedenen Kanäle half dem Team, gut organisiert zu bleiben, den Überblick über den Projektfortschritt zu behalten und rechtzeitig auf Herausforderungen zu reagieren. Die Kombination aus informeller Kommunikation über Whatsapp und formellen Meetings über Zoom schuf eine effiziente und effektive Kommunikationsstruktur, die eine reibungslose Zusammenarbeit ermöglichte. Die Nutzung von Onedrive als zentralem Dateispeicher erleichterte den Zugriff auf gemeinsame Ressourcen und förderte die gemeinsame Bearbeitung von Dokumenten. Insgesamt trug die kluge Wahl der Kommunikationsmittel dazu bei, dass das Projektteam effizient und produktiv zusammenarbeiten konnte.

9. Fazit

Für sämtliche Teammitglieder war das gemeinsame Arbeiten in einem größeren Softwareprojekt unbekannt. Daher war es eine gute Erfahrung, da es eine Ähnlichkeit mit dem Arbeiten im beruflichen Leben hatte und man somit erste Erfahrungen sammeln konnte. Der regelmäßige Austausch mit den anderen Gruppen war stets ein willkommener Input, um neue Ansätze zu hören und sich von den Ideen andere inspirieren zu lassen. Das Thema des Projektes, GO, war für uns alle Neuland und lag nicht gerade in unserem Interessensbereich. Hier hätten wir uns zu Beginn eine Auswahlmöglichkeit gewünscht, evtl. aus zwei Projekten, die thematisch sehr verschieden sind.

10. Abbildungsverzeichnis