Frankfurt University of Applied Sciences

BSRN-Werkstück A

Alternative 3: Buzzword-Bingo

Aouatif Adnane[1530522], Saida Čovrk[1509852], Aisha Khan[1532696], Merwa Tahanur[1505432], Mariam Zadran[1518535]

Studiengang: Wirtschaftsinformatik Sommersemester 2024

Prüfer: Prof. Dr. Christian Baun

Abgabedatum: 30.06.2024

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Dokumentation beschreibt die gezielten Überlegungen und Herangehensweisen, welche die Gruppe bei der Entwicklung der Lösung anwandte. Nach genauer Analyse und sorgfältiger Abwägung der sechs Alternativen, entschied sich die Gruppe für die dritte Alternative: Ein Buzzword-Bingo-Spiel mit Interprozesskommunikation. Die Entscheidung fiel insbesondere deshalb auf diese Alternative, da sie sich durch ihre optimalen Visualisierungsmöglichkeiten der Daten auszeichnete. Als Programmiersprache entschied die Gruppe sich für Python, da die Mitglieder bereits erste Erfahrungen damit hatten und das Modul Objektorientierte Programmierung sie mit der ähnlichen Programmiersprache Java vertraut machte.
Im Verlauf dieser Dokumentation werden die analytischen Schritte zur Lösung und ihre Vorbereitungen detailiert erläutert. Es werden die primären Probleme und ihre Herausforderungen, die während der Durchführung des Projekts auftraten, beschrieben und analysiert. Darüber hinaus präsentiert die Gruppe ihre daraus entwickelten Lösungen und das endgültige Projekt-Ergebnis.
Durch diese detailreiche Darstellung soll ein Verständnis für die Entscheidungen und die technischen Implementierungen der Arbeitsgruppe vermittelt werden.
Die Dokumentation hat zum Ziel das strategische Vorgehen des Teams und seine Kompetenz bei der Bewältigung der komplexen Projektanforderungen zu veranschaulichen.

2 Vorbereitung

In der Anfangs- und Kennenlernphase des Projekts widmete die Gruppe sich der Aufschlüsselung sämtlicher Anforderungen an den Simulator, um eine klare Vorstellung von den zu erledigenden Aufgaben zu bekommen. Dabei analysierte das Team die notwendigen Funktionalitäten des Simulators und legte die Spezifikationen fest, die er erfüllen sollte.
In dieser ersten Phase wurden ausgiebige und tiefgründige Recherchen durchgeführt, um sicherzustellen, dass alle relevanten Bedingungen berücksichtigt und abgedeckt waren. Anschließend wurden

die Aufgaben gezielt an alle Gruppenmitglieder verteilt. Dabei erhielt jedes Gruppenmitglied spezifische Aufgaben, basierend auf individuellen Stärken und Fachkenntnissen. Diese Verteilung umfasste sowohl die Entwicklung der ausgewählten Projektalternative, die Dokumentation mit dem LaTeX-Editor sowie das Testen der jeweiligen Bestandteile des Simulators.

2.1 Anforderungen

Für die Erstellung des Simulators sind verschiedene Anforderungen gegeben. Im Folgenden werden diese Anforderungen als Stichpunkte zusammengefasst, um einen Überblick der zu erfüllenden Kriterien zu gewährleisten.

- **Spielgestaltung:** Das Programm soll eine Kommandozeilenanwendung sein und soll auf einem Computer mit mindestens zwei Spielern gespielt werden können.
- **Erstellen der Bingo-Karte:** Jeder Spieler bekommt eine individuelle Bingokarte in der Shell generiert. Die Größe der Karte wird zu Beginn vom Benutzter festgelegt. Die Wörter sollen aus einer Textdatei eingelesen werden und zufällig generiert sein.
- **Kommunikation der Spieler:** Alle Spieler sind ein eigener Prozess. Zur Interprozesskommunikation werden Nachrichtenwarteschlangen genutzt.
- **Benutzeroberfläche:** Zur grafischen Ausgabe auf der Kommandozeile wird die curses Bibliothek genutzt. Felder werden mit den Pfeiltasten der Tastatur ausgewählt.
- **Gewinnlogik:** Sobald ein Spieler eine komplette Spalte, Zeile oder Diagonale markiert hat, erscheint eine Gewinnanimation bei allen Spielern mit dem Spielernamen des Gewinners. Daraufhin wird das Spiel beendet.
- **Fehlerbehebung:** Das Programm beinhaltet Fehlermeldungen und Debug-Prints, um einen reibungslosen Spielablauf zu gewährleisten.
- **Nutzung des Programms:** Das Buzzword-Bingo-Spiel ist ausschließlich auf Linux-Systemen spielbar, da POSIX-IPC genutzt wird.

2.2 Vorgehen

Um eine unkomplizierte Zusammenarbeit zu gewährleisten, vereinbarte die Gruppe regelmäßige Meetings. Diese nutzten die Mitglieder sowohl in den angebotenen Übungsstunden, wie auch in privaten Meetings und Calls über den Discord-Server, zum Besprechen der Fortschritte und zur Lösung auftretender Probleme. Zudem nutzte die Gruppe GitHub, um den Überblick über die Aufgabenverteilung und die einzelnen Fortschritte zu behalten. So stellte das Team sicher, dass alle Teammitglieder stets über den aktuellen Stand des Projekts informiert waren und dadurch effizient zusammenarbeiten konnten. Diese strukturierte Vorgehensweise ermöglichte es der Arbeitsgruppe, eine Grundlage für die Entwicklung der Projektaufgabe zu schaffen und sicherzustellen, dass alle Aspekte von Beginn an berücksichtigt und erfüllt wurden.

3 Aufbau und Nutzen des Programms

Bevor man die Ausführung des Codes startet, muss vorrausgesetzt sein, dass Python, die curses- Bibliothek und posix_ipc auf dem System installiert ist. Außerdem müssen die Spielskripte auf dem Computer vorhanden sind.Da das Bingo-Spiel Posix-IPC Nachrichtenwarteschlangen nutzt, welche ausschließlich auf Linux-Systemen funktioniert, braucht man ein Linux-Subsystem falls man ein anderes System nutzt.
Um das Programm zu starten, öffnet man das Terminal des Computers. Daraufhin wechselt man in das Verzeichnis, in dem sich die Spielskripte befinden, also die Server-und Client Datei. Daraufhin führt man zunächst das Spielskript für den Server aus. Der Befehl gestaltet sich folgendermaßen. python3 server.py ¡roundfile¿ ¡Log-Pfad¿ ¡Zeilen¿ ¡Spalten¿ ¡max.Spieleranz.¿ Ist dies erfolgt, öffnet man weitere Terminals für die Clients. Auch in diesen Terminals wechselt man zunächst in das richtige Verzeichnis. Ist dies erfolgt, so gibt man folgenden Befehl ein, um das Spiel zu starten. python3 client.py ¡player$_name$ > Nachdem dies abgeschlossen ist, kann das Spiel beginnen.

4 Hauptspiel

Die Idee des Spiels basiert auf dem klassichen Bingo-Spiel. Statt das traditionelle "Bingoßu rufen, wird in diesem technischen Bingo-Spiel ein Bingo-Button gedrückt, um das Gewinnsignal zu geben. Das Buzzword-Bingo-Spiel ist interaktiv gestaltet und bietet mehreren Spielern (mindestens Zwei) die Möglichkeit gegeneinander anzutreten. Es ist eine Anwendung, die für Unterhaltung während der Vorlesungen sorgen soll. Dabei erhalten alle Spieler eine Karte mit verschiedenen zufälligen Wörtern, welche dann markiert werden, sobald diese während der Vorlesung erwähnt werden. Das Bingo-Spiel wird mit einem Intro eröffnet. Dabei haben die Spieler die Möglichkeit, dass Spiel sofort zu beginnen, eine Spielanleitung durchzulesen oder das Spiel zu verlassen. Zu Beginn des Spiels, bekommen die Spieler eine Bingo-Karte mit zufällig generierten Wörtern. Sind Zeilen und Spalten der Bingo-Karte ungerade, so bekommt der Spieler ein Joker-Feld in der Mitte der Karte. Nun können die Spieler beginnen, Wörter zu markieren, sobald sie diese in der Vorlesung hören. Das Spiel wird so lange gespielt, bis ein Spieler eine Wörterreihe diagonal, vertikal oder horizontal markiert. Ist dies erfüllt, drückt man Bingo und hat das Spiel gewonnen. Die Gewinnnachricht wird dann bei allen Spielern als Gewinnanimation angezeigt. Das Geamtkonzept des Bingo-Spiels wird durch das Nutzen von Interprozesskommunikation unterstützt. So können mehrere Spieler auf einem Computer gemeinsam miteinander spielen.

4.1 Benutzeroberfläche (GUI)

Die Gruppe nutzte curses[1] zur Visualisierung der Benutzeroberfläche. Die GUI ermöglicht die Interaktion zwischen den Spielern und dem Spiel und verwaltet zudem verschiedene Askpekte des Spiels. Dazu gehören die Verarbeitung der Spielereignisse, die Handhabung der Fenstergrößen sowie die Aufzeichnung der Aktionen. Die Hauptaufgabe besteht allerdings besonders darin, eine visuell eindrucksvolle und leicht-bedienbare Visualisierung des Programms anzubieten. Da die curses- Bibliothek eine interaktive Benutzereingabe und -ausgabe ermöglicht, war sie ideal für die Entwicklung des Spiels.

[1] https://github.com/zephyrproject-rtos/windows-curses

- **Intro:** Zeigt ein Menü zu Beginn des Spiels an. Hat die Auswahlmöglichkeiten: *Play, Bedienung und Exit.*
- **Kopfinformationen:** Zeigt den Spielernamen auf der Bingo-Karte der Clients an.
- **Beschriftungen:** Zeigt Texte und Fehlermeldungen an.
- **Schaltflächen:** Sind die interaktiven Buttons, welche vom Spieler gedrückt werden können. Sie ändern bei Markierung ihre Farbe in ein leuchtendes Grün. Außerdem ermöglichen sie das Bestätigen des Bingo-Buttons, welches notwendig für das Gewinnsignal ist.

4.2 Gewinnfunktionen

Damit die Gewinnlogik des Programms ordentlich ausgeführt werden kann, werden verschiedene Funktionen implementiert, die diese garantieren. Um das Spielfeld auf mögliche Gewinnbedingungen zu überprüfen, wird die Funktion `check_for_win` verwendet. Sie überprüft alle Gewinnmöglichkeiten horizontal, vertikal und diagonal, indem sie überprüft ob alle Knöpfe in der jeweiligen Reihe gedrückt wurden. Ist dies der Fall, so wird `self.bingo_reached` auf true (wahr) gesetzt, was bedeutet, dass ein Gewinn erreicht wurde. Wenn allerdings keine der Überprüfungen einen Gewinn beinhaltet, so gibt die Methode false (falsch) zurück und das Spiel wird fortgesetzt. Weiterhin gibt es die Methode `check_for_win_and_register`, welche Gewinnüberprüfung und Gewinnregistrierung miteinander verbindet. Gibt es einen Gewinn, so wird dieser mit der Methode `broadcast_win` registriert und als Gewinnmeldung an Server und Clients gesendet. Der genaue Vorgang dessen, wird in der Interprozesskommunikation erläutert. Zuletzt ist die Methode `gewonnen_animation` von Bedeutung. Diese zeigt eine Gewinnanimation auf dem Bildschirm an, sobald ein Spieler das Spiel gewonnen hat. Dafür wird mit der Funktion `clear_screen` der Bildschirm gelöscht, um die Gewinnanimation anzuzeigen. Sie besteht aus einem Text, der den Gewinner beglückwünscht und einer Sternchen-Animation. Sobald die Animation vorbei ist, wird das Spiel beendet. Weitere Gewinnfunktionen, die Interprozesskommunikation beinhalten, werden dort weiter aufgegriffen.

4.3 Logdatei

Erstellung der Log-Datei
Der Logger Code ist ein wichtiger Bestandteil, um die Aktivitäten eines Spielers im Spiel zu protokollieren. Es dokumentiert des Spielbeitritts, also der Name des Spielers, die Spielfeldgröße mit Zeilen und Spalten, die Aktionen auf dem Spielfeld, also das Anklicken eines Buttons sowie das Zurückklicken, dabei werden die Koordinaten des Buttons mitgegeben. Und als Letztes das Wichtigste, das Spielergebnis, wer gewonnen hat. Es wird sichergestellt, dass jede Protokollierung einen Zeitstempel hat, um die Ereignisse genau nachverfolgen zu können.
Um den Logger Code umzusetzen, würden die Bibliotheken datetime und os implementiert. Die Bibliothek datetime wird genutzt, um Zeitstempel in Logger-Datei zu protokollieren. Um die Dateien zu erstellen und zugreifen, ist das os zuständig.

5 Interprozesskommunikation

5.1 Umsetzung

In diesem Programm wird die Interprozesskommunikation (IPC) durch die POSIX-Nachrichtenwarteschlangen (posix_ipc.MessageQueue) realisiert. Diese Art von Nachrichtenwarteschlangen ermöglicht es Prozesse, Nachrichten zu unterschiedlichen Zeiten, also ungleichzeitig, zu senden und zu empfangen. Die IPC hierbei wird verwendet, um die Kommunikation zwischen einem Server und mehreren Clients zu ermöglichen. Sowohl im Server- als auch im Client-Code erfolgt die Umsetzung der IPC.

1. **Einrichtung der Nachrichtenwarteschlangen**
   Server-Code
   Die Nachrichtenwarteschlange *„/serverQueue“* wird verwendet, wodurch das Empfangen der Nachrichten von den Clients ermöglicht wird.
   - **„QUEUE_SERVER“** definiert den Namen der Server-Warteschlange
   - **„queue_server“** erstellt die Nachrichtenwarteschlange
   - **posix_ipc.O_CREAT** stellt sicher, dass diese Warteschlange erstellt wird, wenn sie nicht bereits existiert.

Client-Code
Es gibt mehrere Clients, die jeweils eine eigene Nachrichtenwarteschlange basierend auf dem Spielernamen mit einem eindeutigen Namen erstellen. Der Client sendet eine Join-Nachricht an die Server-Warteschlange, um dem Server mitzuteilen, dass er dem Spiel beigetreten ist.

- **client_queue_name** erstellt den Namen der Client-Warteschlange unter Verwendung des Spielernamens.
- **queue_client** erstellt die Client-Warteschlange.
- **join_message** erstellt eine Nachricht im Format "JOIN—client_queue_name".
- **queue_server.send** sendet die Join-Nachricht an die Server-Warteschlange.

2. **Nachrichtenübermittlung zwischen Server und Clients**
   Der Server empfängt die Join-Nachricht von einem Client, fügt den Client zur Liste der Teilnehmer hinzu und sendet die Spielparameter an den Client.
   - **queue_server.receive** empfängt Nachrichten aus der Server-Warteschlange.
   - **message.decode** decodiert die empfangene Nachricht.
   - **client_queue_name** extrahiert den Namen der Client-Warteschlange aus der Nachricht.
   - **queue_client.send** sendet die Spielparameter an die Client-Warteschlange.
3. **Nutzung von Threads in den Clients**
   Um die Nachrichten ungleichzeitig zu empfangen und gleichzeitig die Spiel-Benutzeroberfläche **User Interface (UI)** zu betreiben, nutzen die Clients ***Threads***. Dies ermöglicht eine parallele Verarbeitung und sorgt dafür, dass die Benutzerreaktionsfähig bleibt.
   - **message_listener** ist eine Funktion, die in einem separaten Thread abläuft und kontinuierlich Nachrichten aus der Client-Warteschlange empfängt.
   - **threading.Thread** erstellt einen neuen Thread mit message_listener als Ziel.
   - **listener_thread.start** startet den Thread.

   Durch die Verwendung von Threads kann der Client Nachrichten ungleichzeitig empfangen und gleichzeitig andere Aufgaben,

wie die Aktualisierung der Benutzeroberfläche, durchführen. Dies verbessert die Reaktionsfähigkeit und Effizienz des Systems erheblich.

5.2 Erläuterung der Funktionsweise

POSIX Message Queues Der code verwendet POSIX Message Queues (‘posix_ipc.MessageQueue‘), um Nachrichten zwischen einem Server und mehreren Clients auszutauschen. Sie ermöglichen eine asynchrone und zuverlässige Kommunikation zwischen den Prozessen.

- Der Server erstellt eine zentrale Message Queue (‘/serverQueue‘), die von allen Clients verwendet wird, um Beitritts- und Gewinnnachrichten zu senden.
- Jeder Client erstellt eine eigene Message Queue (‘/client_¡player_name¿‘), die der Server verwendet, um Spielparameter und Statusmeldungen zu senden.

Nachrichtentypen und Handhabung Es gibt zwei Haupttypen von Nachrichten, die zwischen dem Server und den Clients ausgetauscht werden:

1.Beitrittsnachrichten:

- Clients senden ‘JOIN—¡client_queue_name¿‘ Nachrichten an den Server, um dem Spiel beizutreten.
- Der Server empfängt diese Nachrichten, speichert die Client-Queue-Namen und sendet die Spielernamen zurück.

2. Gewinnnachrichten:

- Wenn ein Client gewinnt, wird eine Nachricht wie ‘¡player_name¿ hat gewonnen¡ an den Server gesendet.
- Der Server empfängt diese Nachricht und leitet sie an alle Clients weiter.

5.3 Nachrichtenvermittlung

Die Nachrichtenvermittlung zwischen dem Server und dem Client ist ein wichtiger Bestandteil der Interprozesskommunikation. Sie ermöglicht es, dass der Server und die Clients effizient miteinander kommunizieren.

Server Der Server ist für den Nachrichtenempfang und für die Nachrichtenverarbeitung zuständig. Er wartet kontinuirrlichin auf Nachrichten in der Serverwarteschlange. Dies erfolgt in einer Schleife, die sicherstellt, dass der Server bereit ist, Nachrichten von Clients zu empfangen und zu verarbeiten, solange nicht alle Spieler dem Beigetreten sind und der nächste Schritt eingeleitet werden kann. Nachdem der Server eine Nachricht aus der Warteschlange empfängt, wird diese dekodiert und analysiert. Wenn es sich bei der Nachricht um eine "JOIN"Nachricht handelt, wird der Client zur Liste der Spieler hinzugefügt und die Spielparameter werden an diesen Client gesendet.

Client Der Client ist der Nachrichtensender und -empfänger. Damit die Clients in der Lage sind, Nachrichten zu senden und zu empfangen, ohne dabei die Hauptanwendung zu blockieren, werden Threads verwendet. Dies ist besonders wichtig um eine reibungslose Benutzererfahrung zu gewährleisten. Die UI des Spiels bleibt weiterhin flüssig, während Nachrichten im Hintergrund empfangen und verarbeitet werden. Nachricht senden: Die Clients senden eine "JOIN"Nachricht an die Serverwarteschlange, um dem Server mitzuteilen, dass sie dem Spiel beigetreten sind. Nachricht empfangen: Ein separater Thread wartet kontinuirlich in der Client-Warteschlange darauf Nachrichten zu empfangen. Die empfangene Nachricht wird dann dekodiert und entsprechend verarbeitet, z.B. um den Gewinn eines Spielers anzuzeigen.

5.4 Gewinnmeldung

Es sind verschienede Methoden implementiert, welche für die Übermittlung und das Senden der Gewinnnachricht zuständig sind. Im Folgenden sind die wichtigsten Methoden beschrieben.

broadcast_win

- Erstellen der Gewinnnachricht: Die Nachricht wird formatiert und codiert.
- Die Gewinnnachricht wird an die Server-Queue gesendet.
- Senden der Gewinnmeldung an alle Clients.

– Die Server-Queue wird geschlossen und entfernt. Das Spiel wird beendet.

handle_win_message Diese Methode verarbeitet eine Gewinnnachricht, indem sie diese Nachricht an alle verbundenen Clients weiterleitet und anschließend das Spiel beendet. Sie funktioniert folgendermaßen:

– Die Server-Queue wird geöffnet.
– Senden der Nachricht an die Client-Queues.
– Schließen und Entfernen der Server-Queue.
– Ordnungsgemäßes Beenden des Spiels.

run_game Diese Methode startet das Spiel und einen `listener_thread`, der auf Gewinnnachrichten lauscht. Der Thread ruft die Methode `message_listener` auf, die auf Gewinnnachrichten wartet und bei Erhalt einer Gewinnnachricht dann die Methode `handle_win` aufruft.

6 Herausforderungen

Während des Projekts ergab sich ein Problem bei der Übergabe der Parameter im Client. Es stellte sich heraus, dass jeder Spieler mit pyTermTk die Möglichkeit hatte zu gewinnen, wobei dei Gewinnbedingung für jeden Spiler separat geprüft wurde.Ein wesentliches Problem trat jedoch auf, als benannte Pipes die Nachrichten nicht korrekt versendeten, was die Kommunikation zwischen den Prozessen beeinträchtigte. Dies führte dazu, dass die Gewinnachricht nicht zuverlässig übertragen wurden, was die Spielerfahrung negativ beeinflusste.

6.1 Interprozesskommunikation

Eine der Hauptherausforderungen bei der Entwicklung des Buzzword-Bingo-Spiels ist die Interprozesskommunikation. Da das Spiel von mehreren Spielern gleichzeitig gespielt wird, müssen die verschiedenen Instanzen des Spiels, also Server und Clients miteinander kommunizieren können. Dies erfordert die Implementierung von prozessübergreifender Kommunikation, um Daten zwischen den Prozessen auszutauschen. Dafür wurden in der Anforderung an den Simulator verschiedene Möglichkeiten, wie Nachrichtenwarteschlangen oder

Pipes angeboten. Die Gruppe entschied sich anfangs für bennante Pipes. Da die Pipes einige Probleme verursachten, hat die Gruppe sie sich im späteren Verlauf für Posix-IPC-Nachrichtenwarteschlangen entschieden. Mit den benannten Pipes war es nicht möglich, zentral zu prüfen, wer das Spiel gewonnen hat, da die Gewinnbedingung für jeden Spieler individuell geprüft wurde. Dies hatte zur Folge, dass nicht nur ein Spieler gewinnnen konnte, sondern alle. Desweiteren war die Datenübertragung ineffizient, da Pipes eine sequenzielle Kommunikation erfordern.So gab es Schwierigkeiten damit, die Nachrichten zwischen Host und Clients ordnungsgemäß über die Pipes zu versenden. Außerdem wurden die Pipes an einigen Stellen im Code nicht richtig geschlossen und an anderen Stellen nicht richtig geöffnet, weswegen es immer wieder zu Fehlermeldungen bei der Code-Ausführung kam. Zwar fiel die Implementierung der Pipes nicht schwer, die damit einhergehenden Fehlermeldungen bezüglich der Gewinnanimation und weiteren Methoden im Code machten das Coden jedoch zeit- und kraftintensiver. Die POSIX-IPC-Nachrichtenwarteschlangen boten eine einfachere Lösung, um die Kommunikation zwischen Prozessen zu verwalten, die Gewinnprüfung zu zentralisieren und die Systemstabilität zu verbessern. Ihre Implementierung war wesentlich einfacher und machte das Endprodukt ersehnlicher.

6.2 Visualisierung

Die Visualisierung des Buzzword-Bingo-Spiels stellt eine weitere Herausforderung dar. Die Gestaltung einer klaren und attraktiven Benutzeroberfäche ist entscheidend. Diese beinhaltet das Layout der Bingo-Karte sowie die Farbe des Buttons beim Anklicken. Zur Visualisierung nutzte die Gruppe die PyTermTk-Bibliothek. Anfangs arbeitete sie mit TKinter, allerdings wurde nach Nachfrage beim Professor, darauf hingewiesen mit der PyTermTk- Bibliothek zu arbeiten.

6.3 Gelöste Probleme

7 Fazit

Das Buzzword-Bingo-Projekt bot eine spannende Gelegenheit, verschiedene Softwareprogramme in einem praktischen Kontext anzuwenden. Obwohl es verschiedene Schwierigkeiten und Hindernisse gab, konnte die Gruppe mit dem Einsatz von Python ein voll funktionsfähiges Spiel erstellen, welches die gegebenen Anforderungen erfüllt. Als Kommandozeilenprogramm mit grafischer Benutzeroberfläche, können mehrere Spieler ein interaktives Bingo-Erlebnis genießen. Über den gemeinsam genutzten Speicher, können alle Spieler effizient miteinander kommunizieren. Die erfolgreiche Implementierung der Interprozesskommunikation garantiert somit eine flüssige Spielerfahrung. Durch die Verwendung von GitHub[2][3], um Fortschritte miteinander zu teilen, konnten die Teammitglieder außerdem eine neue Plattform kennenlernen und sich mit ihr vertraut machen. Durch Gruppencalls auf Discord[4] und einer erstellten Gruppe auf Whatsapp, konnten die Gruppenmitglieder effizient zusammenarbeiten. Dies hatte zum Vorteil, dass Herausforderungen und Schwierigkeiten gemeinsam gelöst und bewältigt werden konnten. Das Buzzword-Bingo-Spiel forderte den Einsatz und die Förderung unserer Fähigkeiten, indem wir mehrere Lösungsansätze entwickelten und Programme mehrmals testen mussten. Dies gab uns zudem einen Einblick in die Entwicklung solcher Anwendungen und ihren Aufwand. Insgesamt stellt das Buzzword-Bingo-Spiel eine gelungene Kombination aus technologischen Aspekten und Spielspaß dar. Es zeigt deutlich, wie Softwareentwicklungsmethoden und effiziente Gruppenarbeit zu einem erfolgreichen Projekt führen können.

[2] https://github.com
[3] https://github.com/a-Bit-Of-Saida/BSRN-Gruppenaufgabe
[4] https://discord.com

8 Anhang

8.1 Kernmodule der Anwendung

Im Folgenden folgt eine Erläuterung der genutzten Module im Code.

```
1  import random
2  import posix_ipc
3  import sys
4  import time
5  import logging
6  import datetime
7  import os
8  import curses
9  import threading
10 import signal
```

- random = Generiert die Zufallswörter auf den Karten.
- posix_ipc = Interprozesskommunikation mit POSIX-Nachrichtenwarteschlangen.
- sys = Genutzt für Befehlszeilenargumente und um das Programm zu beenden.
- time = Genutzt um Verzögerungen und Zeitstempel einzuführen.
- logging = Wird verwendet für die Logdatei. Verfolgt die Ereignisse des Spiels und erfasst sie.
- datetime = Ist dafür zuständig, dass Zeitstempel für die Log-Dateinamen erzeugt werden und Ereignisse mit bestimmten Zeitstempeln erfasst werden.
- os = Wird verwendet um Dateipfade zu erstellen und Prozesse zu handhaben.
- curses = Ausgewählte Bibliothek. Zuständig für textbasierte Benutzeroberflächen.
- threading = Dadurch können kleinere Einheiten eines Prozesses gleichzeitig ausgeführt werden. Damit können Benutzereingaben verarbeitet und gleichzeitig auch Nachrichten gehört werden.
- signal = Stellt sicher, dass die Anwendung entsprechend angepasst wird, falls sich die Größe des Terminals ändert.