ANFORDERUNGSSPEZIFIKATION

NEUROFEEDBACK-OPTIMIZED SELF-SCHEDULING PLATFORM

15. JUNI 2023

TECHNISCHE HOCHSCHULE ASCHAFFENBURG

1. Einführung

1.1. Projekteinführung

Die Anwendung „Neurofeedback-Optimized Self-Scheduling Platform“ befasst sich mit der optimierten Aufgabenzuteilung einer Arbeitsgruppe, wobei jedes Gruppenmitglied selbst für die eigene Zuteilung von Aufgaben verantwortlich ist. Die Software soll einem Masteranden bei seiner Forschungsarbeit unterstützen. Um eine bestmögliche Software zu entwickeln, werden in diesem Dokument die Anforderungen der Software näher geschildert.

1.2. Projektziele

Ziel des Projektes ist es, herauszufinden, ob mit Hilfe von Neurofeedback eine optimale Aufgabenverteilung in einer bestimmten Arbeitsgruppe gewährleistet werden kann. Mit der Software soll geprüft werden, wie Probanden einen bestimmten Aufgabengraph absolvieren. Hierbei ist es wichtig, diese zum einen mit Hilfe von Neurofeedback zu unterstützen und einen Vergleich bezüglich der Abarbeitung ohne Neurofeedback zu erzeugen. Um einen reibungslosen Ablauf der Software zu bieten, werden im Folgenden funktionale und nicht-funktionale Anforderungen an das Software-System geschildert.

1.3. Stakeholder

- Auftraggeber/Kunde: Prof. Dr. Alexander Biedermann, Elias Mende
- Entwicklerteam: Lukas Arnold, Tristan Denk, Sebastian Nesse, Semijon Mujan, Hannes Weber
- Product Owner: Hannes Weber
- Scrum Master: Semijon Mujan
- Anwendungsnutzer: Probandengruppen

2. Anforderungen

2.1. funktionale Anforderungen

Die Software soll folgende funktionale Anforderungen mit sich bringen:
Damit eine reibungslose „Spielsitzung“ stattfinden kann, kommunizieren alle Clients mit einem Server, welcher konsolenbasiert ausgeführt werden soll. Dieser soll zum einen Verbindungsanfragen der verschiedenen Clients akzeptieren, aber auch die Aufgabenverteilung des gesamten Systems steuern. Damit keine fehlerhaften Ausführungen, wie das zeitgleiche Bearbeiten einer Task von zwei Nutzern eintritt, steuert der Server die komplette Abarbeitungsstruktur des Task Graphen. Über die Verbindung des Servers zu den verschiedenen Clients sollen alle nötigen Informationen an die Spieler übertragen werden, sodass diese im User Interface sauber angezeigt werden können. Neben den oben genannten Aufgaben soll der Server auch alle möglichen Events der verbundenen Clients in einer csv-File strukturiert speichern. Dies setzt voraus, dass alle Spiele und andere Events von den jeweiligen Client-Anwendungen über eine TCP-Verbindung an den Server übertragen werden. Die Client-Anwendung soll dem User bei Ausführung die Möglichkeit geben, die jeweilige IP, den Port und seinen Namen, unter dem er spielen wird, anzugeben. Dies ist hilfreich, wenn verschiedene Sitzungen unter verschiedenen Hosts laufen. Der User soll dann, nach erfolgreicher Anmeldung, die Möglichkeit haben, Tasks aus der Ansicht zu pullen, die er spielen kann. Andere angemeldete Spieler können sehen, welche Aufgabe gepullt wurden und ggf. bei hoher Auslastung eines Spielers Aufgaben von diesem abnahmen, bevor das gesamte Spiel durch die Niederlage eines Spiels verloren wird. Die linke Spalte des Taskbereiches in der UI soll dem User eine Übersicht des gesamten Graphen darstellen, um den aktuellen Stand der Abarbeitung visuell sehen zu können. Zudem soll links neben dem Spielfeld, das sich in 4 Felder unterteilt, eine Ansicht gezeigt werden, die die aktuell angemeldeten Nutzer mit den aktuell ausgeführten Tasks zeigt. Hier sollen auch Tasks von anderen Nutzern abgenommen werden können. Welcher Task von einem Nutzer gespielt werden kann liegt daran, welche Art von Spiel im Moment von diesem bearbeitet wird, da zeitgleich nur eines einer bestimmten Kategorie bearbeitet werden kann. Neben den aufgezeigten Features der User Overview soll diese auch in einer speziellen Client-Version die Auslastung eines jeden Users anzeigen. Darüber soll ersichtlich werden, welcher Nutzer in einer Stresssituation ist, und welcher noch Puffer für weitere Aufgaben hat. Die hoch ausgelasteten Nutzer sollen dann von einem anderen Spieler durch das Wegnehmen von Tasks entlastet werden können. Bei den Spielen handelt es sich um verschiedene Spielarten. Die Anwendung soll dabei zum einen ein Spiel implementiert haben, das dem Nutzer die Möglichkeit gibt, über Tasteneingaben eine Kugel innerhalb einer Linie zu bewegen. Dabei soll der Nutzer eine bestimmte Zeit/Distanz auf der Linie zurücklegen, um das Spiel zu gewinnen. Zudem soll es ein Spiel geben, das dem klassischen Memory ähnelt. Hierbei werden dem Spieler eine Matrix an Karten gezeigt, aus denen er Paare an zusammenpassenden Bildern bestimmen muss. Der Spieler muss dabei eine bestimmte Anzahl an Paaren finden, um das Spiel zu gewinnen. Ein weiteres Spiel ist Back Track, wobei der Nutzer eine bestimmte Anzahl an Zahlen gezeigt bekommt und die n-te Zahl davor eingeben muss. Auch hier muss der Spieler eine bestimmte Anzahl an korrekten Eingaben vollziehen, um das Spiel zu gewinnen. Zudem hat der Spieler aber auch eine maximale Anzahl an versuchen, für die er einmal fehlschlagen kann. Wenn diese überschritten wird, zählt das Spiel als verloren. Auch Teil der Anwendung soll der klassische Brick Bracker sein, bei dem der Nutzer einen „Balken“ steuert, um mit einer Kugel verschiedene Blöcke zu eliminieren. Auch hier gibt es eine bestimmte Bedingung, wann das Spiel gewonnen wurde.

Das letzte Spiel nennt sich Tip Tornado. Hier soll der Nutzer innerhalb einer vorgegebenen Zeit eine bestimmte Anzahl an Wörtern eines Textes abschreiben, ohne dabei Fehler zu machen. Das Spiel zählt als gewonnen, wenn der Nutzer die Wortgrenze unter der Maximalzahl an Fehlern erreicht. Wenn zu viele Schreibfehler eingebaut wurden, wird das Spiel verloren.

2.2. nicht-funktionale Anforderungen

Zuverlässigkeit
Das System soll zuverlässig Nachrichten zwischen Server und Client übertragen, um Verlust von Daten zu vermeiden. Zudem soll die Aufgabenverteilung und auch die Ausführung des Programms zuverlässig ablaufen, um Fehler zu vermeiden.

Kompatibilität
Die Software soll Windows Systemen ausführbar sein, damit das Forschungsprojekt in einem Rechenlabor der Technischen Hochschule Aschaffenburg durchgeführt werden kann.

Skalierbarkeit
Mit dem System sollen sich mindestens 20 Clients verbinden lassen, ohne dass es zu Verzögerungen oder anderen Problemen kommt

Leistung
Das Softwaresystem soll reibungslos laufen, ohne dass es zu Wartezeiten beim Übertrag von Nachrichten oder ähnlichem kommt. Zudem soll die Client-Anwendung so gebaut sein, dass die Ausführung von diversen Spielen zeitgleich kein Problem darstellt, und diese flüssig abläuft.

Sicherheit
Da das Projekt nicht deployt wird, ist der Faktor der IT-Sicherheit zweitrangig. Sicherheit muss aber darin gewährleistet sein, dass die Anwendung nur mit den Clients kommuniziert und Nachrichten immer ankommen.

3. User Stories

- Als Sitzungsleiter möchte ich die Möglichkeit haben, eine beliebige Anzahl an Nutzern mit meinem Server verbinden zu lassen.
- Als Sitzungsleiter möchte ich verschiedenen End- und Abbruchkriterien bei den verschiedenen Spielen individuell zu verändern.
- Als Sitzungsleiter möchte ich alle Ereignisse, die während des Programmablaufs ausgeführt werden, gesammelt in einer Log-File am Server auslesen können. Hierzu zählen neben Spiel-Ausgaben auch Key-Events und ähnliches.
- Als Sitzungsleiter möchte ich den Taskgraphen individualisiert gestalten. Hierzu zählen das Einfügen, Verändern und Entfernen von Tasks, um die Größe und Schwierigkeit des Graphen zu variieren.
- Als Spieler möchte ich eine Übersicht des Taskgraphen sehen, die mir anzeigt, welche Task gerade verfügbar ist und welche aktuell bearbeitet wird. Hier will ich meine neuen Tasks über einen einfachen „Klick“ auswählen können.
- Als Spieler möchte ich neben der Hauptansicht auch eine Übersicht des gesamten Graphen sehen, die mir die aktuelle Position des Scroll Fensters anzeigt. Zudem soll dieser dieselben Features haben, wie der Hauptgraph. Es soll nur keine Tasks darüber zugeteilt werden können.
- Als Spieler möchte ich im Taskgraphen sehen, welcher Spieler aktuell welche Task bearbeitet.
- Als Spieler will ich einem unter großem Stress ausgesetzten Kollegen Aufgaben abnehmen können. Dies soll zum einen über die Nutzeransicht, aber auch über den Task Graph möglich sein.
- Als Spieler möchte ich auf der linken Seite des User Interfaces eine Ansicht haben, in der ich alle aktuell verbundenen Benutzer sehe. Ich will dabei an oberster Stelle stehen.
- Als Spieler möchte ich Anhand der Useransicht sehen, in welchem Feld welche Aufgabe von einem bestimmten User bearbeitet wird.
- Als Spieler möchte ich auch über die Useransicht einem Arbeitskollegen eine Aufgabe abnehmen können, wenn ich sehe, dass dieser in einer erhöhten Stresssituation ist.