



POLSKO-JAPÓŃSKA AKADEMIA
TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria oprogramowania, procesów biznesowych i baz danych

Weronika Barczykowska

Nr albumu s22380

Paweł Barczykowski

Nr albumu s22943

**Stressless VR – aplikacja wspomagająca
oswajanie stresujących sytuacji z
wykorzystaniem technologii wirtualnej
rzeczywistości**

Praca magisterska napisana pod
kierunkiem:

dr. inż. Mariusza Trzaski, prof. PJATK

Warszawa, lipiec, 2026

Streszczenie

Niniejsza praca magisterska przedstawia koncepcję wykorzystania technologii wirtualnej rzeczywistości jako narzędzia wspomagającego proces adaptacji do stresogennych sytuacji. Symulacje przeprowadzane w realistycznym, a zarazem bezpiecznym środowisku cyfrowym zapewniają odpowiednie warunki do treningu i mogą przyczyniać się do lepszego radzenia sobie w rzeczywistych okolicznościach, na co wskazują wyniki przeanalizowanych badań.

W pracy przeprowadzono analizę dostępnych na rynku aplikacji, a następnie opracowano autorskie rozwiązanie obejmujące analizę potrzeb, koncepcję aplikacji oraz implementację prototypu. Powstały prototyp aplikacji zawiera przykładową scenę przedstawiającą realistyczne otoczenie sprzyjające oswojeniu się z sytuacjami takimi jak rozmowa o pracę czy egzamin ustny. Przygotowana scena dostępna jest w dwóch wersjach: wersji o wysokim poziomie szczegółowości, przeznaczonej na komputery osobiste oraz gogle VR, a także wersji *low poly*, przeznaczonej na urządzenia mobilne.

Symulacje nie są ograniczone do sztywnych scenariuszy dzięki wykorzystaniu sztucznej inteligencji. Każda z nich poprzedzona jest etapem konfiguracji, który obejmuje m.in. wybór modelu językowego, wprowadzenie *promptu* oraz opcjonalne dodanie pliku wykorzystywanego przez mechanizm RAG. Elementy te pozwalają określić kontekst symulacji oraz zakres wiedzy wykorzystywanej przez model. Użytkownik ma zatem możliwość weryfikowania swojej wiedzy na dany temat, a zarazem oswojenia się ze stresem związanym z daną sytuacją poprzez prowadzenie swobodnej rozmowy z cyfrowym awatarem zasilanym sztuczną inteligencją. Po każdej symulacji system dostarcza raport z rozmowy zawierający statystyki dotyczące poziomu oszacowanego stresu oraz podsumowanie. Na podstawie wyników przeanalizowanych badań można zatem stwierdzić, że opracowana aplikacja ma potencjał do zastosowania zarówno w pracy indywidualnej, jak i jako narzędzie wspierające specjalistów.

Aplikacja charakteryzuje się wysokim poziomem generyczności oraz elastyczności dzięki zastosowaniu architektury modułowej. Stronę kliencką oparto na lekkim, otwartoźródłowym (ang. *open-source*) i dobrze zoptymalizowanym, silniku Godot, który zapewnia wsparcie technologii VR. Natomiast aplikacja serwerowa napisana jest w języku Python z wykorzystaniem bibliotek, takich jak Vosk, Coqui TTS oraz Phonemizer, na których oparte są poszczególne funkcjonalności. Umożliwia to ich łatwą wymianę, rozbudowę oraz dostosowanie do przyszłych potrzeb projektu. Uzyskane rezultaty stanowią zatem punkt wyjścia do dalszego rozwoju prototypu o nowe sceny, animacje, biblioteki do analizy mowy i dźwięku czy integracje z kolejnymi chmurowymi modelami językowymi.

Słowa kluczowe:

Wirtualna rzeczywistość (VR), sztuczna inteligencja (AI), duże modele językowe (LLM), stres, zarządzanie stresem, symulacje, przetwarzanie języka naturalnego (NLP)

Podziękowania

Pragniemy podziękować promotorowi, dr. inż. Mariuszowi Trzasce, prof. PJATK, za nieocenioną pomoc i wsparcie merytoryczne podczas realizacji niniejszej pracy.

Spis treści

1.	Wstęp	5
1.1.	Motywacja	5
1.2.	Cel pracy	5
1.3.	Rozwiązania przyjęte w pracy	5
1.4.	Rezultaty pracy	6
1.5.	Organizacja pracy	6
1.6.	Podział prac	7
2.	Przegląd literatury i podstawy do realizacji projektu	8
2.1.	Stres – definicja, mechanizmy i skutki	8
2.2.	Strategie radzenia sobie ze stresem	9
2.3.	Stres związany z wystąpieniami publicznymi	10
2.4.	Technologie wspierające zarządzanie stresem	10
2.4.1	Rola inteligentnych zegarków w monitoringu stresu	11
2.4.2	Aplikacje mobilne jako narzędzie treningowe do oswajania stresu	11
2.4.3	Wirtualna rzeczywistość (VR) w redukcji stresu	12
2.4.4	Podsumowanie	14
3.	Analiza rynku i porównanie istniejących aplikacji	15
3.1.	Aplikacje wykorzystujące technologię wirtualnej rzeczywistości	15
3.1.1	Symulatory firmy Visus VR	15
3.1.2	Symulatory firmy 4 HELP VR	17
3.1.3	Symulatory EPICVR	19
3.1.4	iPro VR - „Trudne rozmowy”	19
3.1.5	Trening rozmów rekrutacyjnych - tematyczna aplikacja szkoleniowa VR	20
3.1.6	VR Explorer 2	21
3.1.7	VirtualSpeech	24
3.1.8	Ovation	26
3.2.	Aplikacje mobilne	28
3.2.1	Orai for Public Speaking, Pres	28
3.2.2	TalkReady: AI Interview Prep	30
3.3.	Podsumowanie	33
4.	Koncepcja aplikacji Stressless VR	34
4.1.	Słownik pojęć	34
4.2.	Wymagania funkcjonalne	35
4.3.	Wymagania нефункционалне	39
4.4.	Wysokopoziomowa architektura rozwiązania	40
4.5.	Charakterystyka aktorów	43
4.6.	Przypadki użycia	43
4.6.1	Konfiguracja symulacji	44
4.6.2	Przebieg symulacji	48
4.6.3	Raport i statystyki	49
4.7.	Diagramy sekwencji	53
4.8.	Integracje	55
5.	Narzędzia i technologie zastosowane przy realizacji pracy	57
5.1.	Wykorzystane narzędzia	57
5.1.1	Enterprise Architect	57
5.1.2	Godot	58
5.1.3	PyCharm	59
5.1.4	Git	59
5.1.5	Gitea	60
5.1.6	Docker	60

5.1.7	Komodo	62
5.2.	Wykorzystane technologie	62
5.2.1	Python	62
5.2.2	Biblioteka Vosk	62
5.2.3	Biblioteka Coqui TTS	63
5.2.4	Biblioteka Phonemizer	64
5.2.5	Biblioteka openSMILE	64
5.2.6	Modele LLM	65
6.	Opis implementacji	66
6.1.	Przegląd architektury	66
6.2.	Aplikacja serwerowa (Python)	67
6.2.1	Architektura i komunikacja WebSocket	67
6.2.2	Moduł mowy – STT, TTS i fonemizacja	69
6.2.3	Moduł analizy akustycznej	71
6.2.4	Moduł RAG	73
6.2.5	Integracja z modelami językowymi (Ollama / Gemini)	74
6.3.	Aplikacja kliencka (Godot)	76
6.3.1	Komunikacja z serwerem i obsługa audio	76
6.3.2	Synchronizacja ruchu ust	78
6.3.3	Sceny	79
6.3.4	Generyczne animacje awatara	81
6.3.5	Wskaźnik stanu i napisy	82
7.	Opis prototypu aplikacji	83
7.1.	Ekran startowy	83
7.2.	Konfiguracja ustawień	83
7.3.	Rozmowa egzaminacyjna	85
7.4.	Rozmowa kwalifikacyjna – low poly	88
7.5.	Ekran podsumowania statystyk	90
8.	Podsumowanie i perspektywy rozwoju	94
8.1.	Trudności napotkane podczas realizacji pracy	94
8.2.	Wady i zalety rozwiązania	94
8.3.	Możliwe kierunki rozwoju aplikacji	96
	Bibliografia	98
	Dodatki	102
	Dodatek A: Spis rysunków	102
	Dodatek B: Spis tabel	104
	Dodatek C: Spis listingów	105

1. Wstęp

Stres jest nieodłącznym elementem życia każdego człowieka, który niesie za sobą zarówno pozytywne, jak i negatywne skutki. Z jednej strony może motywować do działania. Z drugiej zaś nadmierny lub długotrwały stres może prowadzić do negatywnych skutków psychicznych i fizycznych, takich jak obniżenie nastroju, spadek efektywności, wypalenie zawodowe, a nawet rozwój poważnych chorób.

Współczesny styl życia, intensywne tempo pracy i presja społeczna powodują, że coraz więcej osób doświadcza trudności w radzeniu sobie ze stresem. Ponadto w wyniku długotrwałej izolacji związanej z rosnącym trendem pracy zdalnej wiele osób odczuwa lęk przed interakcjami na żywo.

W odpowiedzi na te wyzwania coraz częściej poszukuje się nowoczesnych technologii wspierających człowieka w zarządzaniu emocjami i przygotowaniu się do stresujących sytuacji. Jedną z obiecujących technologii w tym zakresie jest wirtualna rzeczywistość (VR), która pozwala na symulowanie różnych zdarzeń w bezpiecznym i kontrolowanym środowisku.

1.1. Motywacja

Motywacją autorów do podjęcia wyzwania jakim jest realizacja projektu wspomagającego oswajanie stresujących sytuacji jest chęć stworzenia użytecznego rozwiązania z wykorzystaniem nowych technologii, które będzie odpowiadać na realne problemy. Stres, będący nieodłącznym elementem życia każdego człowieka oraz niejednokrotnie towarzyszący mu w najważniejszych momentach, może znacząco obniżać komfort funkcjonowania i utrudniać realizację istotnych zadań.

Zdaniem autorów jednym z najważniejszych czynników wpływających na znaczącą redukcję poziomu stresu podczas różnych sytuacji, takich jak rozmowa o pracę czy obrona pracy dyplomowej, jest przygotowanie się do nich, rozumiane jako ugruntowanie wiedzy oraz oswojenie się z potencjalnym otoczeniem, a także możliwymi scenariuszami danej sytuacji.

1.2. Cel pracy

Celem niniejszej pracy magisterskiej jest pogłębienie tematyki związanej ze stresem oraz strategiami zaradczymi, a następnie zaprojektowanie i implementacja prototypu aplikacji „Stressless VR – aplikacja wspomagająca oswajanie stresujących sytuacji z wykorzystaniem technologii wirtualnej rzeczywistości”, wspomagającej użytkowników w oswajaniu się z sytuacjami stresowymi poprzez realistyczne symulacje w wirtualnej rzeczywistości. Aplikacja ma umożliwiać trening reakcji i zachowań w trudnych sytuacjach, co może przełożyć się na większy spokój, opanowanie i skuteczność w realnych warunkach.

1.3. Rozwiązania przyjęte w pracy

Na całość projektu składają się dwa główne moduły - aplikacja kliencka odpowiedzialna za prezentowanie scen i interakcję użytkownika z AI oraz aplikacja serwerowa odpowiadająca za przetwarzanie mowy, komunikację z LLM, zarządzanie pamięcią RAG oraz generowanie mowy (TTS).

Stronę kliencką oparto na lekkim, otwartoźródłowym (ang. *open-source*) i dobrze zoptymalizowanym silniku Godot, który zapewnia wsparcie technologii VR, a także bezproblemowe i łatwe budowanie aplikacji na różne platformy takie jak Windows czy MacOS oraz architektury takie

jak ARM czy x86. Natomiast aplikacja serwerowa napisana jest w języku Python z wykorzystaniem bibliotek takich jak Vosk, Coqui TTS oraz Phonemizer.

Architektura rozwiązania jest opisana w rozdziale czwartym, natomiast szczegóły dotyczące zastosowanych narzędzi i technologii znajdują się w rozdziale piątym.

1.4. Rezultaty pracy

Głównym rezultatem pracy są wnioski z analizy problematyki związanej ze stresem, które pozwoliły na opracowanie koncepcji rozwiązania odpowiadającego na zidentyfikowane problemy i potrzeby. Analiza objęła literaturę dotyczącą tematyki stresu oraz badania naukowe, których wyniki stanowiły podstawę do oceny zasadności realizacji prototypu zaproponowanego rozwiązania.

Pozostałe produkty, stanowiące podstawę realizacji pracy, obejmują analizę rynku wraz z porównaniem istniejących aplikacji.

1.5. Organizacja pracy

Praca magisterska została podzielona na siedem rozdziałów opisujących zarówno podstawy teoretyczne pracy, wyniki analizy, jak i wykorzystane narzędzia oraz szczegóły implementacyjne.

Rozdział pierwszy stanowi wprowadzenie do tematyki pracy, określa jej cel, motywacje autorów do jej podjęcia, przyjęte rozwiązania projektowe, a także prezentuje osiągnięte rezultaty.

Rozdział drugi zawiera przegląd literatury oraz podstawy teoretyczne związane z tematyką stresu, jego mechanizmami, sposobami radzenia sobie z nim, a także technologiami wspierającymi zarządzanie stresem – ze szczególnym uwzględnieniem wirtualnej rzeczywistości (VR).

W rozdziale trzecim dokonano analizy rynku oraz porównania istniejących aplikacji wykorzystujących m.in. technologię wirtualnej rzeczywistości, które mają na celu przeszkolenie użytkownika w różnych dziedzinach, a co za tym idzie – redukcję stresu związanego z daną sytuacją. Następnie przedstawiono wyniki analizy funkcjonalność, zastosowań i ograniczeń wybranych rozwiązań.

Rozdział czwarty przedstawia koncepcję projektowanej aplikacji Stressless VR, w tym definicję wymagań funkcjonalnych i нефункциональных, architekturę rozwiązania, przypadki użycia, a także diagramy opisujące działanie systemu.

W rozdziale piątym opisano narzędzia i technologie, które zostały wykorzystane podczas realizacji prototypu.

Rozdział szósty zawiera szczegółowy opis prototypu aplikacji, jego funkcjonalności oraz sposobu działania.

Rozdział siódmy prezentuje przykładowe scenariusze użycia aplikacji wraz z szczegółowym opisem poszczególnych ekranów.

Rozdział ósmy stanowi podsumowanie pracy, przedstawia napotkane trudności, zalety i wady rozwiązania, a także możliwe kierunki dalszego rozwoju aplikacji.

Na końcu pracy umieszczono bibliografię oraz dodatki, takie jak spis rysunków, tabel i listingów.

1.6. Podział prac

Niniejsza praca została wykonana wspólnie przez oboje autorów, którzy wzajemnie wspierali się zarówno w zakresie implementacji systemu, jak i redakcji dokumentu. Mimo że podział odpowiedzialności nie był równy w każdym obszarze – Paweł Barczykowski skoncentrował się przede wszystkim na implementacji prototypu, natomiast Weronika Barczykowska na opracowaniu dokumentacji – oboje aktywnie uczestniczyli we wszystkich etapach realizacji projektu. Poniżej przedstawiono orientacyjny podział odpowiedzialności w ramach projektu.

Weronika Barczykowska wspierała przede wszystkim w tworzeniu warstwy klienckiej aplikacji w silniku Godot – opracowanie scenariuszy symulacji, konfigurację i dostrajanie modelu postaci przygotowanego w oprogramowaniu Character Creator w wersji 5 oraz budowę scen trójwymiarowych. W zakresie animacji zajęła się dostrajaniem wag fonemów i czasów wypowiedzi, regulacją zakresów animacji realizowanych przez mechanizm mieszania kształtów (ang. *blend shapes*) oraz ogólnym dopracowaniem ruchu awatara. Zaprojektowała koncepcję interfejsu użytkownika aplikacji klienckiej, określając układ ekranów oraz przepływ interakcji. W ramach pracy pisemnej odpowiadała za rozdziały drugi i trzeci, obejmujące przegląd literatury dotyczącej tematyki stresu, strategii zaradczych oraz technologii wspierających zarządzanie stresem, a także analizę rynku i porównanie istniejących aplikacji. Opracowała diagramy przypadków użycia w narzędziu Enterprise Architect wraz z ich opisem oraz scenariuszami, a także wysokopoziomą architekturę rozwiązania przedstawioną w rozdziale czwartym.

Paweł Barczykowski odpowiadał za całość aplikacji serwerowej zrealizowanej w języku Python, obejmującej moduł rozpoznawania mowy z wykorzystaniem biblioteki Vosk, moduł syntezy mowy opartej na bibliotece Coqui TTS z fonemizacją realizowaną przez Phonemizer, moduł analizy akustycznej oparty na bibliotece openSMILE oraz moduł RAG z integracją z modelami językowymi Ollama i Gemini. Zajął się również implementacją warstwy komunikacyjnej WebSocket po obu stronach systemu – zarówno po stronie serwera, jak i aplikacji klienckiej Godot – w tym kodowaniem protokołu wymiany wiadomości, mechanizmów połączenia oraz zachowań aplikacji w odpowiedzi na odbierane zdarzenia. W ramach pracy pisemnej odpowiadał za rozdział szósty opisujący implementację prototypu.

Analiza przypadków użycia prowadzona była wspólnie – Paweł Barczykowski służył jako źródło wiedzy technicznej dotyczącej działania systemu, natomiast Weronika Barczykowska modelowała i opisywała przypadki użycia, tworzyła scenariusze oraz diagramy sekwencji i maszynę stanową przedstawione w rozdziale czwartym. Wspólnie opracowano model działania aplikacji, wymagania funkcjonalne i нефункционалне oraz podjęto kluczowe decyzje projektowe dotyczące funkcjonalności i architektury systemu. Rozdział piąty dotyczący narzędzi i technologii był redagowany wspólnie.

2. Przegląd literatury i podstawy do realizacji projektu

W niniejszym rozdziale przedstawiono definicje stresu, jego potencjalne skutki, a także wybrane strategie zaradcze, które są stosowane w przeciwdziałaniu skutkom nadmiernego napięcia emocjonalnego. Autorzy szczególną uwagę poświęcają nowoczesnym technologiom, które w ostatnich latach zyskały na znaczeniu w obszarze wspierania zdrowia psychicznego. W tym kontekście został omówiony potencjał technologii wirtualnej rzeczywistości (VR) jako narzędzia wspierającego zarządzanie stresem.

Rozdział ten stanowi podstawę teoretyczną dla zaprojektowania prototypu aplikacji „Stressless VR – aplikacja wspomagająca oswojanie stresujących sytuacji z wykorzystaniem technologii wirtualnej rzeczywistości”.

2.1. Stres – definicja, mechanizmy i skutki

Stres, będący terminem powszechnie używanym i intuicyjnie rozumianym, nie jest jednak pojęciem prostym ani możliwym do jednoznacznego zdefiniowania, gdyż może być rozpatrywany w różnych ujęciach. Na stres można spojrzeć z perspektywy fizjologicznej, rozumianej jako reakcja organizmu objawiająca się m.in. pobudzeniem, potliwością czy suchością w ustach. Jedną z pierwszych definicji ujmowała stres właśnie z perspektywy biologicznej. Została ona przedstawiona przez Hansa Selyego, który określił stres jako „nieswoista reakcja organizmu na wszystkie stawiane mu żądania” [1]. Z drugiej strony stres może być analizowany w kontekście bodźców – sytuacji lub zdarzeń odbieranych jako wymagające, trudne bądź zagrażające – które wywołują tego typu reakcje.

Oba te podejścia – reakcyjne i bodźcowe – mogą być niepełne i niejednoznaczne w interpretacji. Jeśli chodzi o pierwsze z nich, to objawy fizjologiczne takie jak pobudzenie, potliwość czy suchość w ustach mogą być spowodowane wieloma przyczynami. Natomiast jeśli chodzi o sytuacje stresowe, to każdy może inaczej na nie reagować np. student przygotowany do egzaminu będzie mniej zdenerwowany niż student, który się nie uczył przez cały semestr.

Richard Lazarus i Susan Folkman zaproponowali natomiast model transakcyjny, który według [1] należy rozumieć jako „relację między osobą a otoczeniem, która jest oceniana przez jednostkę jako obciążająca lub przekraczająca jej zasoby oraz zagrażająca jej dobrostanowi”.

Współcześnie stres jest rozumiany jako kompilacja wcześniej przedstawionych podejść – reakcji, bodźca oraz transakcji. Zgodnie z [2] „stres nie jest lokalizowany ani w jednostce, ani w otoczeniu, ale dotyczy określonego rodzaju relacji (interakcji, transakcji) między nimi”. Zatem zaczynamy się stresować wówczas, gdy oceniamy narzucone przez daną sytuację wymagania lub nasze wewnętrzne standardy jako wykraczające poza aktualne możliwości.

W literaturze można znaleźć następującą klasyfikację stresu – stres ostry i stres przewlekły. Podczas gdy krótkotrwały stres może nieść pozytywne efekty np. działać mobilizująco, chroniczny stres może prowadzić do szeregu negatywnych skutków zdrowotnych – od zaburzeń snu, przez osłabienie układu odpornościowego, aż po rozwój depresji, a z perspektywy zawodowej do wypalenia. Szczególnie niepokojące są skutki stresu przewlekłego w środowiskach zawodowych, a także edukacyjnych. Z tego względu istotne staje się rozwijanie skutecznych metod radzenia sobie ze stresem, który stanowi nieodłączny element współczesnego życia.

2.2. Strategie radzenia sobie ze stresem

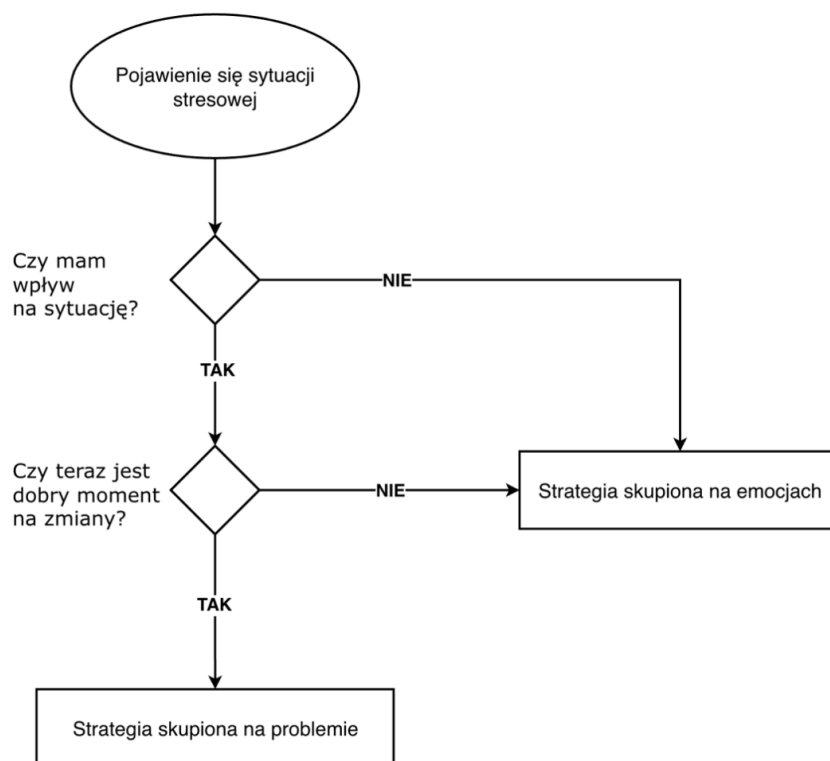
W literaturze [3] można odnaleźć dwa podstawowe sposoby radzenia sobie ze stresem, opisane przez Richarda Lazarusa i Susan Folkman w ramach prac nad wcześniej wspomnianym transakcyjnym modelem stresu.

Pierwszy z nich zorientowany jest na problem (ang. *problem-focused coping*), co należy rozumieć jako aktywne poszukiwanie możliwych rozwiązań problemu będącego źródłem negatywnych emocji, w tym stresu. Jeśli jako przykład przyjąć egzamin dyplomowy oraz studenta, który nie opanował materiału w stopniu spełniającym jego wewnętrzne standardy, aktywnym podejściem do problemu byłoby poświęcanie określonej ilości czasu na systematyczne powtarzanie wiedzy. Jeżeli konieczność wystąpienia przed komisją byłaby kolejnym źródłem stresu, student mógłby dodatkowo ćwiczyć potencjalne scenariusze rozmowy poprzez symulację tego wydarzenia. Zarówno powtarzanie materiału, jak i symulowanie sytuacji egzaminacyjnej mogłyby przyczynić się do zmniejszenia poziomu stresu oraz zwiększenia pewności siebie studenta.

Drugie podejście skoncentrowane na emocjach (ang. *emotion-focused coping*) stosowane jest dla sytuacji, których nie da się opanować. Kiedy są one poza zasięgiem, nie da się ich kontrolować jedynym rozwiązaniem pozostaje zarządzanie negatywnymi emocjami, w tym stresem, poprzez techniki takie jak:

- unikanie (oczekiwanie na poprawę sytuacji),
- dystansowanie się (nieprzejmowanie się problemem),
- reinterpretację wydarzeń (szukanie pozytywów sytuacji),
- samokontrola (wstrzymywanie emocji np. poprzez ćwiczenia oddechowe),
- poszukiwanie wsparcia społecznego (rozmowa z bliskimi czy przyjaciółmi na temat danej sytuacji).

Rysunek 1 przedstawia uproszczony schemat doboru strategii w zależności od rodzaju problemu.



Rysunek 1 Algorytm wyboru strategii radzenia sobie ze stresem. Źródło: Opracowanie własne.

Podczas gdy strategie skoncentrowane na emocjach mogą być skuteczne i przynosić szybkie rezultaty, mogą nie stanowić najlepszego rozwiązania długoterminowego, w szczególności, jeśli istnieje możliwość podjęcia aktywnego działania.

2.3. Stres związany z wystąpieniami publicznymi

W artykule Grieve, Woodley, Hunt i McKay [4] wskazano, że wystąpienia przed publicznością, będące nieodłącznym elementem edukacji, a w późniejszych etapach także rozwoju zawodowego, należą do czynności mogących budzić szczególnie silne obawy. Autorzy podkreślają, że negatywne doświadczenia związane z wystąpieniami publicznymi mogą być czynnikiem przyczyniającym się do problemów ze zdrowiem psychicznym oraz dobrostanem studentów.

Ponadto autorzy przywołują wyniki ankiety przeprowadzonej wśród studentów dwóch brytyjskich uniwersytetów, z której wynika, że aż 80% badanych uznało prezentacje ustne za źródło lęku społecznego wpływającego na naukę i dobrostan.

W ramach badania jakościowego Grieve i współautorów zastosowano cztery pytania otwarte, w tym pytanie dotyczące strategii wykorzystywanych w celu zmniejszenia lęku przed wystąpieniami publicznymi, w tym prezentacjami ustnymi. Wśród odpowiedzi wskazano następujące strategie:

- ćwiczenia oddechowe,
- ćwiczenie przed rówieśnikami,
- głośne omawianie całej prezentacji,
- nagrywanie próbnej prezentacji w celu jej późniejszej analizy,
- przedstawianie prezentacji małej grupie osób postrzeganych jako onieśmielające, w celu przygotowania się do wystąpienia przed większą grupą odbiorców lub specjalistów z danej dziedziny.

Natomiast w odpowiedzi na pytanie dotyczące tego, w jaki sposób uniwersytet mógłby wspierać studentów odczuwających lęk przed wystąpieniami publicznymi, wskazano m.in.:

- więcej możliwości ćwiczenia,
- sesje ćwiczeniowe w małych grupach,
- możliwość ćwiczenia wystąpień publicznych w spokojnej i wspierającej atmosferze.

Kluczowym wnioskiem z przywołanego badania jest zatem potrzeba zapewnienia odpowiedniego wsparcia w zakresie wystąpień publicznych osobom odczuwającym lęk przed tego typu sytuacjami. Wyniki omawianych badań potwierdzają zatem przypuszczenia autorów niniejszej pracy, że stres związany z sytuacjami społecznymi, takimi jak wystąpienia publiczne, a także – przez analogię – egzaminy ustne czy rozmowy o pracę, stanowi realny problem wpływający na dobrostan wielu osób. Jednocześnie wskazują one na potrzebę stworzenia bezpiecznej przestrzeni do ćwiczenia tego typu sytuacji, co potwierdza zasadność realizacji projektu Stressless VR.

2.4. Technologie wspierające zarządzanie stresem

Rozwój technologii oraz wzrost zainteresowania dobrostanem (ang. *well-being*) przyczyniły się do powstania nowych możliwości w zakresie monitorowania i zarządzania stresem. Za przykład można wskazać coraz popularniejsze urządzenia typu *wearables*, takie jak inteligentne zegarki (ang. *smartwatch*) czy opaski fitness (ang. *smartband*), które z wykorzystaniem aplikacji mobilnych pozwalają skutecznie monitorować i kontrolować szereg różnych parametrów, w tym poziom stresu. Na rynku dostępnych jest również coraz więcej aplikacji dedykowanych urządzeniom VR, oferujących

realistyczne i bezpieczne środowisko, w którym użytkownik dzięki immersyjnym symulacjom może się zrelaksować, a także przećwiczyć określone sytuacje.

2.4.1 Rola inteligentnych zegarków w monitoringu stresu

Zgodnie z wynikami przeglądu systematycznego [5] inteligentne zegarki wykazują duży potencjał w zakresie monitorowania zdrowia psychicznego oraz zarządzania stresem. Pomimo istniejących ograniczeń, takich jak:

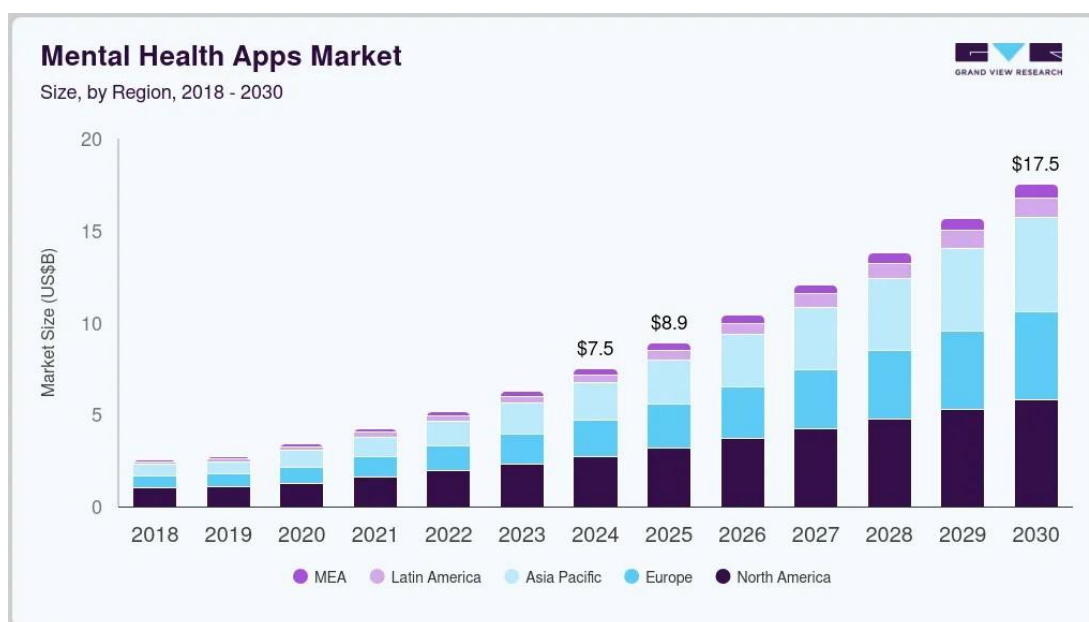
- dokładność sensorów zbierających dane,
- ograniczony czas pracy baterii, w wyniku czego dane nie mają charakteru ciągłego,
- niedoskonałość algorytmów uczenia maszynowego, które mogą błędnie interpretować odczytane parametry,

urządzenia te przyczyniają się do poprawy codziennych nawyków użytkowników. W razie wykrycia podwyższonego poziomu stresu są w stanie wysłać alert z propozycją ćwiczeń oddechowych, które mogą pomóc zredukować napięcie.

Funkcje oferowane przez tego typu urządzenia mogą przyczyniać się do budowania większej świadomości użytkownika oraz doraźnie wspierać opanowywanie emocji związanych z bieżącą sytuacją, co wpisuje się we wcześniej opisane strategie skoncentrowane na emocjach. Inteligentne zegarki nie rozwiązują jednak bezpośrednio źródła problemu, czyli samego stresora. W kolejnych podrozdziałach autorzy przeanalizują zatem potencjał innych narzędzi i technologii w kontekście długoterminowego radzenia sobie ze stresem.

2.4.2 Aplikacje mobilne jako narzędzie treningowe do oswajania stresu

Z raportu opracowanego przez firmę badawczo-analityczną Grand View Research [6] wynika, że rosnące zainteresowanie dobrostanem przyczyniło się do rozwój rynku aplikacji mobilnych wspierających zdrowie psychiczne i zarządzanie stresem. Rysunek 2 przedstawia szacowaną wartość rynku aplikacji zdrowia psychicznego w latach 2018-2024 wraz z prognozą do roku 2030.

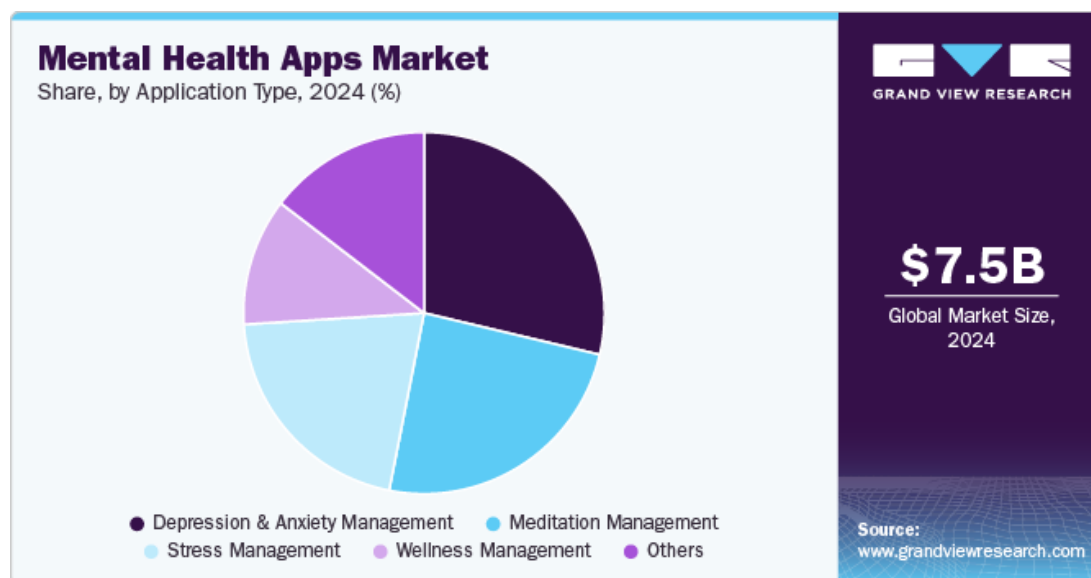


Rysunek 2 Szacowana wartość rynku aplikacji zdrowia psychicznego z prognozą do roku 2030. Źródło: [6]

Szacuje się, że globalna wartość rynku aplikacji dotyczących zdrowia psychicznego w 2024 r. wyniosła 7,48 mld USD, a do 2030 r. ma ona osiągnąć wartość 17,52 mld USD, przy średniorocznej

stopie wzrostu (CAGR) na poziomie 14,6% w latach 2025–2030. Zmieniające się podejście od tradycyjnych metod opieki zdrowotnej w kierunku opieki skoncentrowanej na pacjencie oraz spersonalizowanego wsparcia sprzyja popularyzacji aplikacji wspierających zdrowie psychiczne. Aplikacje oferujące m.in. medytację, zarządzanie depresją i lękiem oraz wspieranie dobrostanu mogą pozytywnie wpływać na utrzymanie ogólnego zdrowia użytkownika, promowanie zdrowszego stylu życia oraz redukcję stresu.

Zgodnie z raportem aplikacje wspierające zarządzanie stresem stanowią drugą najpopularniejszą kategorię po aplikacjach skoncentrowanych na zarządzaniu depresją i lękiem, co przedstawiono na Rysunek 3.



Rysunek 3 Struktura rynku aplikacji zdrowia psychicznego według typu aplikacji w 2024 roku. Źródło: [6]

Aplikacje te podobnie jak wcześniej opisane inteligentne zegarki pomagają zarządzać stresem, ale nie eliminują źródła problemu. Z tego powodu oprócz aplikacji skoncentrowanych na relaksacji, medytacji czy monitorowaniu nastroju coraz większą popularność zyskują również rozwiązania ukierunkowane na aktywne przygotowanie użytkownika do sytuacji wywołujących stres.

Istotnymi zaletami aplikacji mobilnych jest ich łatwa dostępność oraz możliwość korzystania z nich w dowolnym miejscu i czasie, co sprzyja regularnemu treningowi i systematycznemu rozwijaniu umiejętności radzenia sobie ze stresem. Dzięki czemu użytkownicy mogą stopniowo oswajać sytuacje wywołujące napięcie.

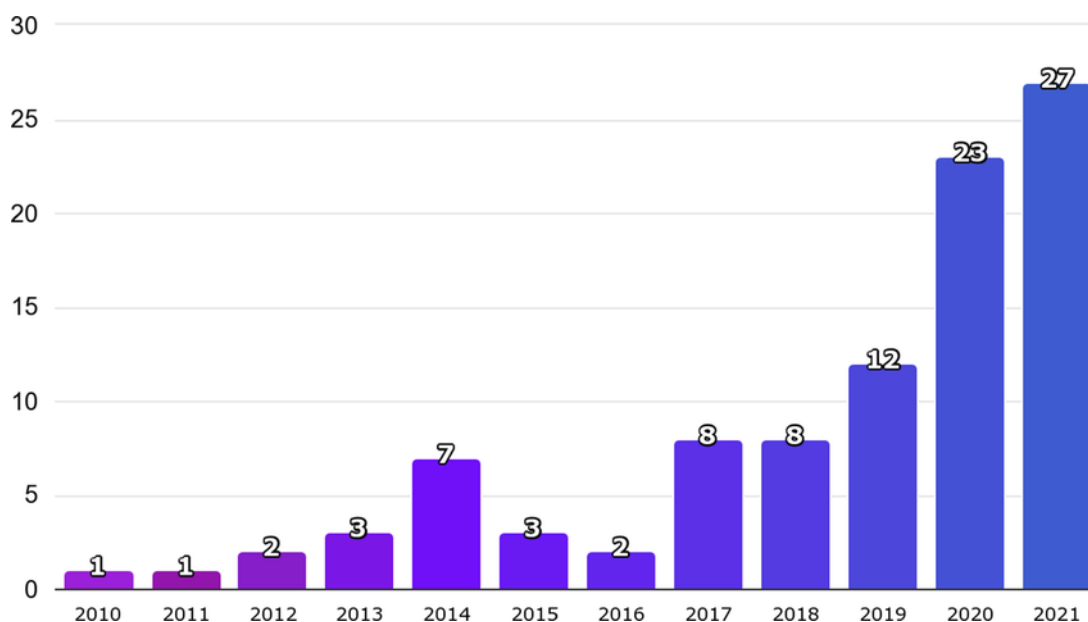
W literaturze podkreśla się również, że aktywne przygotowanie do sytuacji stresowej może być skuteczniejszym rozwiązaniem długoterminowym niż strategie oparte wyłącznie na chwilowej redukcji napięcia.

2.4.3 Wirtualna rzeczywistość (VR) w redukcji stresu

Znaczny rozwój technologii wirtualnej rzeczywistości oraz udostępnienie jej na rynku konsumenckim otworzyły szereg nowych możliwości. Często technologia ta kojarzona jest z rozrywką bądź edukacją, jednakże w ostatnich latach coraz częściej znajduje zastosowanie w obszarze wsparcia zdrowia psychicznego, w tym w zarządzaniu stresem. Jej potencjał wynika z możliwości tworzenia realistycznych i bezpiecznych środowisk, które angażują zmysły użytkownika i pozwalają na pełne zanurzenie w wybranej scenarii.

Rosnące uznanie potencjału technologii wirtualnej rzeczywistości we wspieraniu standardowej praktyki klinicznej w dziedzinie psychiatrii i psychologii oraz wzrost zainteresowania badaczy tematyką

redukcji stresu, potwierdza przegląd zakresowy autorstwa Ladakisa i współautorów [7]. Rysunek 4 przedstawia histogram zaprezentowany w przeglądzie, obrazujący znaczący wzrost liczby badań dotyczących systemów i podejść służących redukcji stresu.



Rysunek 4 Histogram badań według roku publikacji. Źródło: [7]

Szczególny wzrost nastąpił w latach 2020-2021, na które przypada okres pandemii COVID-19. Autorzy przeglądu wskazują, że mogło to wynikać z obciążeń psychologicznych doświadczanych przez wiele osób w tym czasie, m.in. w związku z izolacją społeczną oraz zmianą warunków codziennego funkcjonowania.

Autorzy przeglądu przeanalizowali różne zastosowania wirtualnej rzeczywistości w kontekście redukcji stresu oraz zarządzania nim. Pomimo trudności związanych z definiowaniem, wykrywaniem i oceną stresu wyniki wskazują na obiecujący potencjał tej technologii.

Systematyczny przegląd badań przeprowadzony przez Meshkat i współautorów [8] również potwierdza wyraźny potencjał interwencji wykorzystujących technologię wirtualnej rzeczywistości w redukcji poziomu stresu. Chociaż część badań przyniosła niejednoznaczne rezultaty, w wielu przypadkach zaobserwowano jednak zmniejszenie poziomu stresu, lęku oraz poprawę dobrostanu uczestników. Autorzy zwracają jednocześnie uwagę, że analizowane badania były prowadzone w zróżnicowanych kontekstach oraz obejmowały różne grupy uczestników i rodzaje interwencji, co może częściowo wyjaśniać niejednoznaczność części wyników. Do przeglądu włączono bowiem 50 badań obejmujących blisko 2,9 tysiąca uczestników, a wśród najczęściej stosowanych metod można wymienić m.in.:

- relaksację,
- zadania powtórzeniowe,
- sesje uważności (mindfulness),
- terapię poznawczo-behawioralną, w tym terapię ekspozycyjną (np. symulacje sytuacji społecznych, takich jak wystąpienia publiczne).

Autorzy podkreślili również, że potencjalne korzyści wynikające z terapii ekspozycyjnej – będącej jedną z bardziej wymagających form interwencji – nie wydają się wiązać z ryzykiem wystąpienia negatywnych skutków u uczestników. Wynika to z możliwości konfrontacji ze stresorami w bezpiecznym i kontrolowanym środowisku.

W podobnym tonie wypowiadają się autorzy metaanalizy poświęconej ocenie skuteczności terapii ekspozycyjnej z wykorzystaniem rzeczywistości wirtualnej (ang. *Virtual Reality Exposure Therapy, VRET*) w redukcji objawów zaburzeń lęku społecznego [9]. Metaanaliza wykazała, że skuteczność terapii ekspozycyjnej prowadzonej w środowisku wirtualnym jest porównywalna ze skutecznością tradycyjnych metod leczenia, takich jak ekspozycja *in vivo*, terapia poznawczo-behawioralna, psychoedukacja czy poradnictwo. Ponadto autorzy podkreślają, że połączenie terapii ekspozycyjnej opartej na wirtualnej rzeczywistości z klasycznymi elementami terapii poznawczo-behawioralnej może przynosić korzystniejsze efekty terapeutyczne. Dodatkową zaletą VRET może być jej forma, która nie wymaga bezpośredniego kontaktu twarzą w twarz z terapeutą, co dla niektórych pacjentów może stanowić istotną barierę w rozpoczęciu terapii. Metoda ta jest efektywna czasowo i wiąże się z mniejszymi wymaganiami organizacyjnymi, dzięki czemu skuteczna terapia ekspozycyjna może stać się dostępna dla szerszej populacji pacjentów. Jednak, podobnie jak we wcześniej wspomnianych przeglądach, autorzy metaanalizy wskazują na potrzebę przeprowadzenia dodatkowych badań na większych grupach uczestników oraz z uwzględnieniem długoterminowej obserwacji.

2.4.4 Podsumowanie

Z punktu widzenia niniejszej pracy szczególnie interesującą technologią wykorzystywaną do redukcji poziomu stresu jest wirtualna rzeczywistość, a najistotniejszą techniką jest ekspozycja, która wykracza poza bierną relaksację i skupia się na aktywnym ćwiczeniu sytuacji wywołujących stres. Szczególnie ciekawym kierunkiem wykorzystania VR w zarządzaniu stresem wydają się zatem aplikacje umożliwiające wielokrotne ćwiczenie określonych sytuacji stresowych w realistycznym, a jednocześnie bezpiecznym środowisku.

Aplikacje wykorzystujące technologię VR do redukcji i zarządzania stresem mimo wciąż istniejących luk badawczych oraz trudnościach związanych z ewaluacją wyników już przeprowadzonych badań, stanowią obiecujące narzędzie. Wirtualne środowiska mogą bowiem tworzyć bezpieczną przestrzeń do treningu i stopniowej ekspozycji, a dotychczasowe wyniki wskazują na potencjalne korzyści tego typu interwencji, przy jednoczesnym braku negatywnych skutków u uczestników.

3. Analiza rynku i porównanie istniejących aplikacji

Wzrost zainteresowania tematyką dobrostanu (ang. *well-being*) oraz zarządzania stresem przyczynił się do poszukiwania rozwiązań, które mogłyby stanowić wsparcie w radzeniu sobie z napięciem, budowaniu odporności psychicznej oraz poprawie jakości codziennego funkcjonowania. Obok standardowych aplikacji przeznaczonych na urządzenia mobilne coraz większe zastosowanie znajduje technologia wirtualnej rzeczywistości. Jej dynamiczny rozwój oraz upowszechnienie na rynku konsumenckim przyczyniły się do powstania rosnącej liczby aplikacji umożliwiających użytkownikom ćwiczenie różnorodnych sytuacji w bezpiecznym, a jednocześnie realistycznym środowisku.

W dalszej części rozdziału przedstawione zostaną wybrane rozwiązania dostępne na rynku, obejmujące zarówno aplikacje mobilne, jak i narzędzia wykorzystujące technologię VR. Analiza obejmie omówienie mocnych i słabych stron tych rozwiązań, a także identyfikację potencjalnych luk rynkowych, które mogą stanowić punkt wyjścia do zaprojektowania nowego rozwiązania.

3.1. Aplikacje wykorzystujące technologię wirtualnej rzeczywistości

Niniejszy podrozdział poświęcony jest analizie wybranych aplikacji wykorzystujących technologię wirtualnej rzeczywistości do oswajania użytkowników z różnorodnymi sytuacjami wymagającymi emocjonalnie.

3.1.1 Symulatory firmy Visus VR

Firma Visus VR Sp. z o.o. [10] we współpracy z EPICVR S.A. stworzyła system interaktywnych szkoleń dotyczących udzielania pierwszej pomocy i ratownictwa z zastosowaniem zaawansowanych technik wirtualnej rzeczywistości oferując kilka scenariuszy szkoleniowych takich jak:

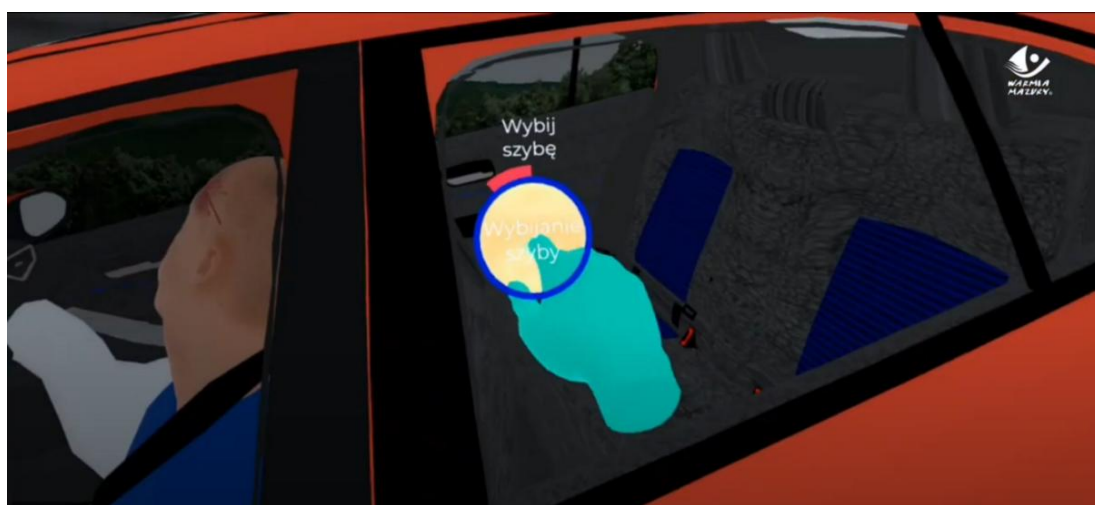
- wypadek drogowy,
- udar mózgu w pociągu,
- nagłe zatrzymanie krążenia w szkole,
- nagłe zatrzymanie krążenia w biurze,
- udzielanie pomocy w scenariuszu baśniowym,
- wstrząs anafilaktyczny w gabinecie stomatologicznym,
- medycyna pola walki przeznaczone dla służb mundurowych.

Aplikacja szkoleniowa wykorzystuje technologię śledzenia dłoni (ang. *hand tracking*) umożliwiając użytkownikom intuicyjną obsługę interfejsu, naturalne wejście w interakcje w cyfrowym środowisku, a przede wszystkim przeprowadzanie symulacji z wykorzystaniem fizycznych fantomów (Rysunek 5). Wykorzystanie pełnej haptyki zwiększa realizm symulacji przyczyniając się do poprawy efektywności szkoleń i przygotowuje użytkowników do działania w rzeczywistych sytuacjach.



Rysunek 5 Symulacja resuscytacji krążeniowo-oddechowej z wykorzystaniem fizycznego fantomu oraz gogli VR. Źródło: [11]

Rozwiązanie stworzone przez firmę Visus VR dostarcza predefiniowanych scenariuszy, które użytkownik może przećwiczyć w pełni bezpiecznym i kontrolowanym środowisku. Użytkownik musi postępować zgodnie z następującymi po sobie krokami składającymi się na symulację danej sytuacji. Rysunek 6 zawiera kadr z materiału wideo przedstawiającego symulację wypadku drogowego. Widoczny jest na nim jeden z etapów postępowania, który może być wymagany w tego rodzaju sytuacji.



Rysunek 6 Symulacja wypadku drogowego. Źródło: [10].

Użytkownik ma zatem ograniczoną elastyczność w zakresie swobody działania w przestrzeni wirtualnej, co jednak może być postrzegane jako zaleta w przypadku sytuacji takich jak udzielanie pierwszej pomocy czy działania ratownicze, w których procedury postępowania są ściśle określone i nie powinny być modyfikowane.

3.1.2 Symulatory firmy 4 HELP VR

Podobne narzędzie stworzyła firma 4 HELP VR [12], które z wykorzystaniem technologii wirtualnej rzeczywistości pomaga w nauce i treningu z zakresu udzielania pierwszej pomocy. Przy użyciu gogli VR użytkownicy mogą w bezpiecznym, a zarazem realistycznym wirtualnym środowisku przećwiczyć procedury ratunkowe, takie jak:

- opatrywanie ran,
- tamowanie krwotoków,
- wypadki komunikacyjne,
- nagłe zatrzymanie krążenia,
- przeprowadzanie RKO na wirtualnym fantomie,
- ćwiczenie układania poszkodowanego w pozycji bezpiecznej.

Rysunek 7 przedstawia symulację resuscytacji krążeniowo-oddechowej z wykorzystaniem fizycznego fantomu oraz gogli VR.



Rysunek 7 Symulacja RKO z wykorzystaniem VR oraz fizycznego fantomu. Źródło: [12]

Rysunek 8 przedstawia natomiast jeden z kroków będących częścią procedury udzielania pierwszej pomocy, widoczny z perspektywy środowiska wirtualnego.



Rysunek 8 Symulacja udzielania pierwszej pomocy osobie nieprzytomnej w wirtualnej rzeczywistości. Źródło: [12]

Aplikacja oferuje również programy szkoleniowe dotyczące BHP, pracy na wysokości, szkolenie z zakresu ergonomii pracy biurowej, ochrony przeciwpożarowej oraz szkolenie dla strażaków.

Rysunek 9 przedstawia ćwiczenia praktyczne dotyczące ochrony przeciwpożarowej z użyciem gogli VR oraz podręcznego sprzętu gaśniczego.



Rysunek 9 Szkolenie z zakresu PPOŻ. Źródło: [12]

Symulatory firmy 4 HELP VR, podobnie jak rozwiązanie firmy Visus VR, ograniczają swobodę działania użytkownika. Ze względu na charakter symulacji oraz obszar ich zastosowania ograniczenie to może być jednak postrzegane jako zaleta. Dodatkowym atutem rozwiązania jest wykorzystanie fizycznych artefaktów podczas symulacji, które może zwiększać poziom realizmu odtwarzanej sytuacji.

3.1.3 Symulatory EPICVR

EPICVR [13] to doświadczony VR Software House, który specjalizuje się w projektowaniu oraz wdrażaniu zaawansowanych rozwiązań szkoleniowych z wykorzystaniem zaawansowanej technologii wirtualnej rzeczywistości. Firma opracowała liczne interaktywne programy szkoleniowe mające na celu odwzorowanie rzeczywistych warunków pracy, procedur awaryjnych oraz scenariuszy wymagających podejmowania szybkich decyzji. Szkolenia opracowane przez EPICVR znajdują zastosowanie m.in. w branżach medycznej, przemysłowej, lotniczej oraz w obszarach związanych z bezpieczeństwem i higieną pracy (BHP) i ochroną przeciwpożarową (PPOŻ).

Rysunek 10 przedstawia kadr z symulacji będącej częścią szkolenia dotyczącego badania miejsca zdarzenia drogowego.



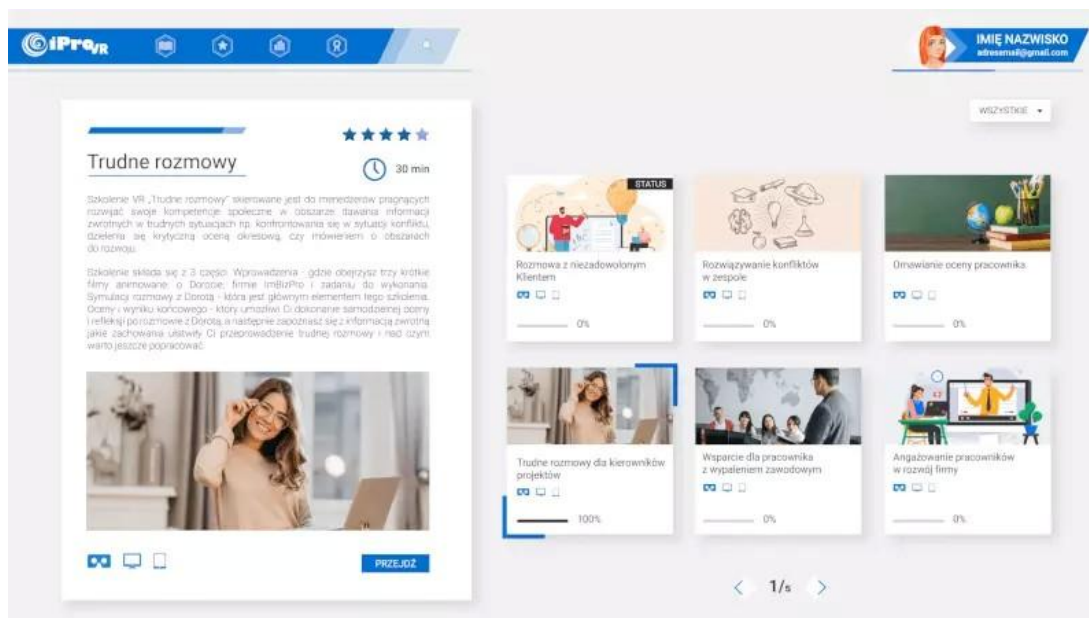
Rysunek 10 Symulacja badania miejsca zdarzenia drogowego. Źródło: [13]

3.1.4 iPro VR - „Trudne rozmowy”

Firma iPro VR [14] opracowała natomiast innowacyjne narzędzie oparte na technologii wirtualnej rzeczywistości, którego domeną są rozmowy menadżerskie. Celem szkolenia „Trudne rozmowy” jest przygotowanie uczestników do prowadzenia wymagających konwersacji w środowisku zawodowym poprzez ćwiczenia kompetencji miękkich w realistycznym i bezpiecznym wirtualnym środowisku. Każda z symulacji skupiona jest na rozwoju określonych umiejętności, takich jak:

- radzenie sobie z oporem,
- reagowanie na krytykę i atak,
- udzielanie informacji zwrotnej,
- efektywne komunikowanie zmian,
- skuteczne wyrażanie prośb i oczekiwań,
- asertywne wyznaczanie granic w komunikacji.

Rysunek 11 przedstawia jeden z ekranów aplikacji, na którym widoczny jest opis szkolenia oraz możliwość wyboru zagadnienia do przećwiczenia w ramach kursu.



Rysunek 11 Ekran z aplikacji - wybór rozmowy. Źródło: [14]

Dzięki zastosowaniu wirtualnego środowiska uczestnik ma możliwość przećwiczenia różnych scenariuszy rozmów w realistycznym, a jednocześnie w pełni kontrolowanym środowisku, bez ryzyka negatywnych konsekwencji. Rozwiązanie stwarza zatem bezpieczną przestrzeń do eksperymentowania, w której użytkownik może popełniać błędy bez stresu i obaw.

W krytycznych momentach dialogu podczas symulacji przed użytkownikiem pojawia się kilka gotowych wariantów wypowiedzi reprezentujących inną strategię komunikacyjną, np. reakcję agresywną, postawę uległą lub próbę manipulacji. Rozwiązanie dostarczone przez iPro VR nie oferuje zatem możliwości prowadzenia w pełni swobodnych rozmów, lecz koncentruje uwagę użytkownika na potencjalnych konsekwencjach wyboru określonej strategii komunikacyjnej.

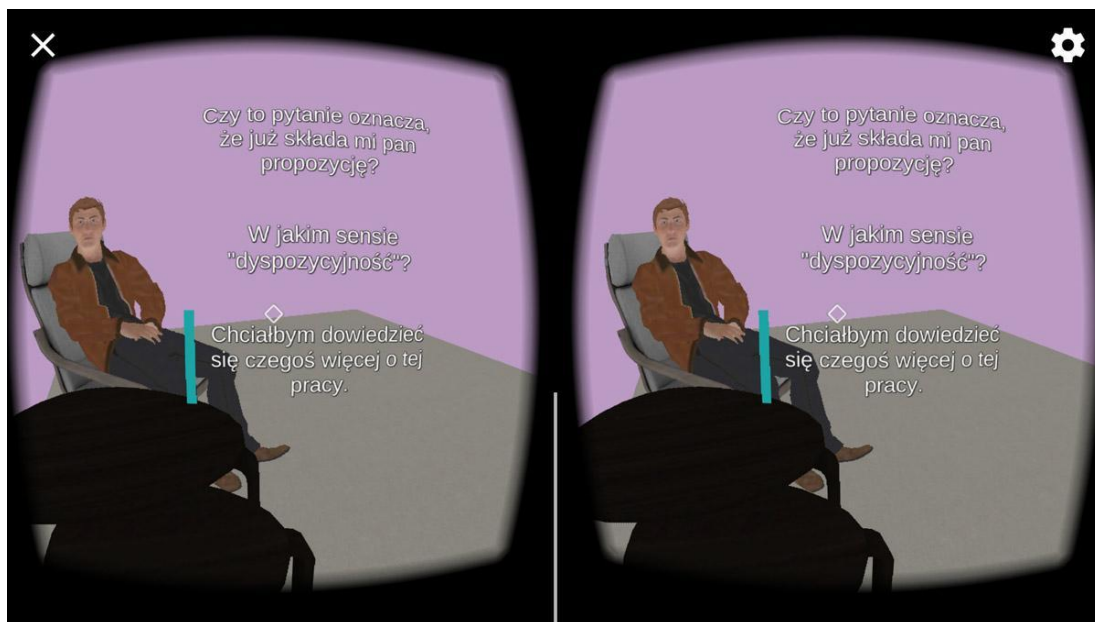
Zaletą takiego podejścia jest skłanianie uczestnika do analizy subtelnych różnic w słownictwie oraz sposobie formułowania wypowiedzi, co budować świadomość komunikacyjną i przyczynić się do rozwinięcia umiejętności miękkich.

Z innej perspektywy brak możliwości swobodnego formułowania wypowiedzi sprawia, że użytkownik nie ma pełnej przestrzeni do ćwiczenia spontanicznego mówienia w warunkach napięcia emocjonalnego. Co jest kluczowe, ponieważ w sytuacjach stresowych samo posiadanie wiedzy na temat właściwej reakcji często może okazać się niewystarczające, jeśli użytkownik nie miał wcześniej okazji przećwiczyć jej samodzielnego zastosowania w praktyce.

W tym kontekście potencjalną lukę stanowi brak rozwiązania, które łączyłoby edukacyjny charakter wyboru strategii komunikacyjnych z możliwością swobodnego wypowiadania się i trenowania reakcji werbalnych w realistycznych warunkach stresu.

3.1.5 Trening rozmów rekrutacyjnych - tematyczna aplikacja szkoleniowa VR

Podobne rozwiązanie dostarczyła firma VR Training Sp. z o.o. [15], która opracowała szkolenie skupione na treningu rozmów rekrutacyjnych. Użytkownik bierze udział w rozmowie wchodząc w interakcję z wirtualnym rekruterem poprzez wybór predefiniowanych pytań i odpowiedzi. Rysunek 12 przedstawia kadr z symulacji rozmowy rekrutacyjnej.



Rysunek 12 Symulacja rozmowy rekrutacyjnej. Źródło: [15]

Zaletą aplikacji może być jej wysoka dostępność sprzętowa, gdyż nie wymaga ona korzystania z zaawansowanego zestawu VR. Rozwiązanie umożliwia bowiem wykorzystanie smartfona umieszczonego w niskokosztowym uchwycie pełniącym funkcję gogli VR.

Natomiast podobnie jak w przypadku narzędzia iPro VR ograniczenie symulacji do wyboru opcji z listy nie daje użytkownikowi pełnej możliwości treningu danej sytuacji stresowej. Użytkownik nie formułuje samodzielnie odpowiedzi, co ogranicza możliwość ćwiczenia spontanicznego wypowiedzania się w warunkach zbliżonych do rzeczywistej rozmowy rekrutacyjnej.

3.1.6 VR Explorer 2

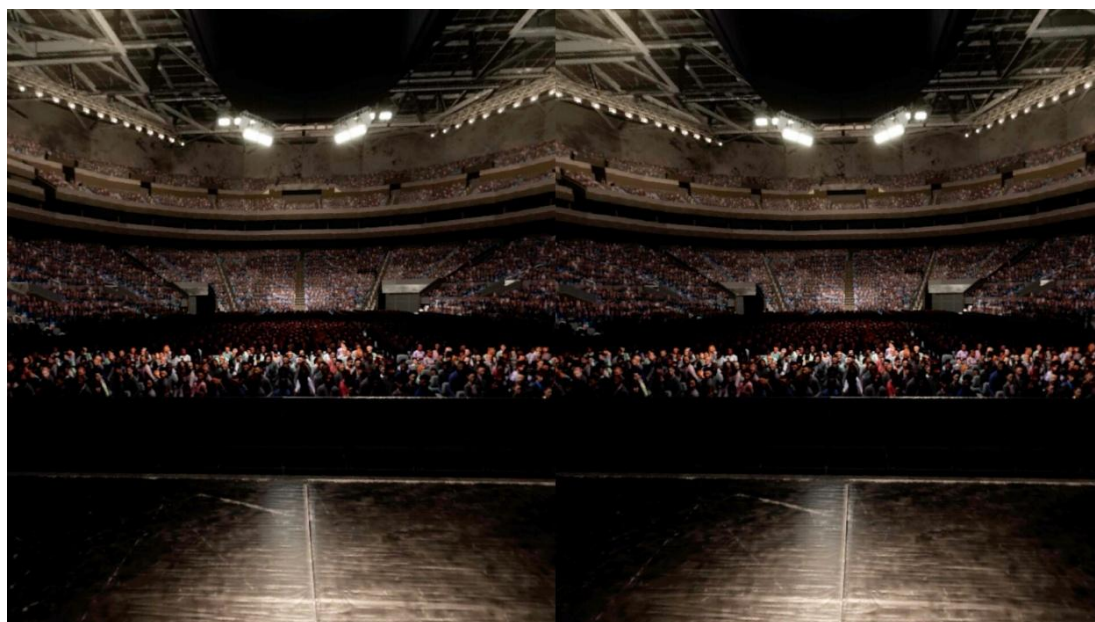
Kolejnym wartym uwagi rozwiązaniem jest darmowa aplikacja VR Explorer 2 [16], która została opracowana przez dr. Chrisa Macdonalda, dyrektora Immersive Technology Lab działającego przy Lucy Cavendish College, University of Cambridge. Aplikacja powstała w odpowiedzi powszechny problem związany z lękiem przed wystąpieniami publicznymi.

Celem opracowanego rozwiązania jest redukcja strachu związanego z wystąpieniami publicznymi oraz zwiększenie pewności siebie poprzez ćwiczenia w bezpiecznym i realistycznym otoczeniu. W związku z tym rozwiązanie oferuje szeroką gamę środowisk szkoleniowych, które można podzielić na dwie kategorie: ogólną oraz specjalistyczną. Rysunek 13 przedstawia ekran startowy aplikacji, na który składają się dwie listy, spośród których można wybrać odpowiednie otoczenie.



Rysunek 13 Ekran startowy aplikacji VR Explorer 2. Źródło: [16]

Za pomocą pierwszej listy użytkownik może wybrać uniwersalne środowisko zawierające nagrania 360° publiczności o różnej liczebności. Aplikacja umożliwia wybór publiczności składającej się zarówno z pojedynczych osób, jak i większych grup, liczących 100, 200, 500 lub 1000 osób. Rysunek 14 przedstawia widok z symulacji wystąpienia publicznego przed dużą publicznością.



Rysunek 14 Widok z symulacji wystąpienia przed dużą publicznością. Źródło: [16]

Druga lista zawiera otoczenia specjalistyczne odpowiadające sytuacjom, takim jak:

- rozmowa w radiu,
- odczytywanie treści z promptera w studiu telewizyjnym,
- rozmowa o pracę, której kadr przedstawia Rysunek 15.



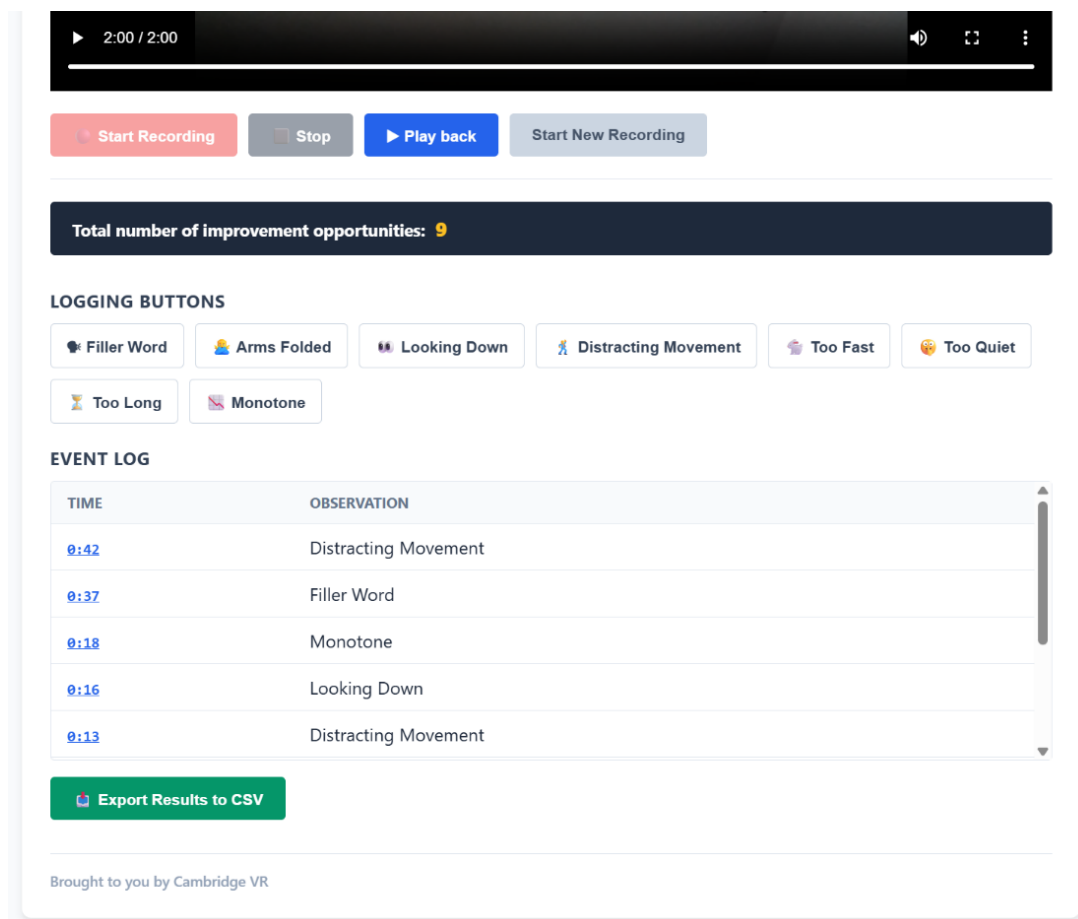
Rysunek 15 Widok z symulacji rozmowy o pracę. Źródło: [16]

W czasie symulacji rozmowy o pracę oraz rozmowy radiowej w nagraniu pojawiają się ogólne pytania z przerwami na udzielenie odpowiedzi. Przerwy te nie są dostosowane do długości wypowiedzi, co może prowadzić do rozproszenia użytkownika, gdy będzie musiał oczekiwać na kolejne pytanie lub skończy odpowiedź na dłuższą chwilę przed czasem. Ze względu na charakter aplikacji nie dochodzi zatem do żadnej interakcji użytkownika z wirtualnym środowiskiem.

Rozwiązanie nie generuje również raportów z wystąpienia. Użytkownik może dokonać samooceny wystąpienia korzystając z osobnej aplikacji „Presentation Analyst App” zbudowanej w ramach całości projektu dr. Chrisa Macdonalda. Należy nagrać wystąpienia, a następnie poprzez funkcję ponownego odtwarzania użytkownik ma możliwość oznaczenia momentów wystąpienia wymagających poprawy pod kątem elementów, takich jak:

- głośność,
- monotonia,
- tempo mówienia,
- krzyżowanie rąk,
- opuszczanie wzroku,
- rozpraszające ruchy.

Rysunek 16 przedstawia przykładową ocenę wystąpienia.



Rysunek 16 Przykładowa ocena wystąpienia w aplikacji Presentation Analyst App. Źródło: Opracowanie własne.

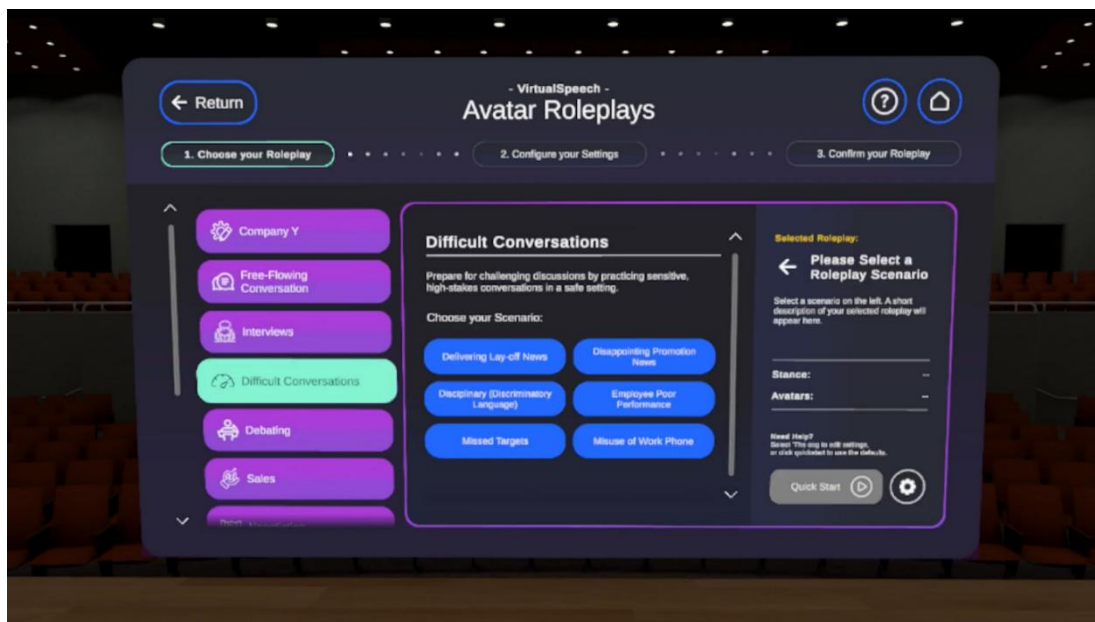
Istotną zaletą aplikacji VR Explorer 2 jest jej wysoka dostępność sprzętowa. Rozwiązanie nie wymaga bowiem korzystania z zaawansowanego zestawu VR, ponieważ umożliwia wykorzystanie smartfona umieszczonego w niskokosztowym uchwycie pełniącym funkcję gogli VR. Dzięki temu bariera wejścia dla użytkownika jest znacząco niższa niż w przypadku aplikacji wymagających dedykowanego zestawu VR. Plusem jest również duży wybór różnego rodzaju realistycznych symulacji. Jednakże symulacje te nie są interaktywne, co jest dużym minusem, w przypadku, gdy użytkownik chciałby poćwiczyć prowadzenie dynamicznej rozmowy.

3.1.7 VirtualSpeech

Kolejną wartą uwagi aplikacją jest VirtualSpeech [17], która pod względem założeń jest jednym z rozwiązań najbliższych projektowi Stressless VR. VirtualSpeech to narzędzie treningowe wspomagające rozwój kompetencji miękkich z wykorzystaniem wirtualnej rzeczywistości oraz awatarów zasilanych sztuczną inteligencją. Platforma umożliwia użytkownikom ćwiczenie m.in.:

- wystąpień publicznych,
- negocjacji biznesowych,
- rozmów kwalifikacyjnych,
- komunikacji w sytuacjach zawodowych
- przekazywania informacji zwrotnej (ang. *feedback*).

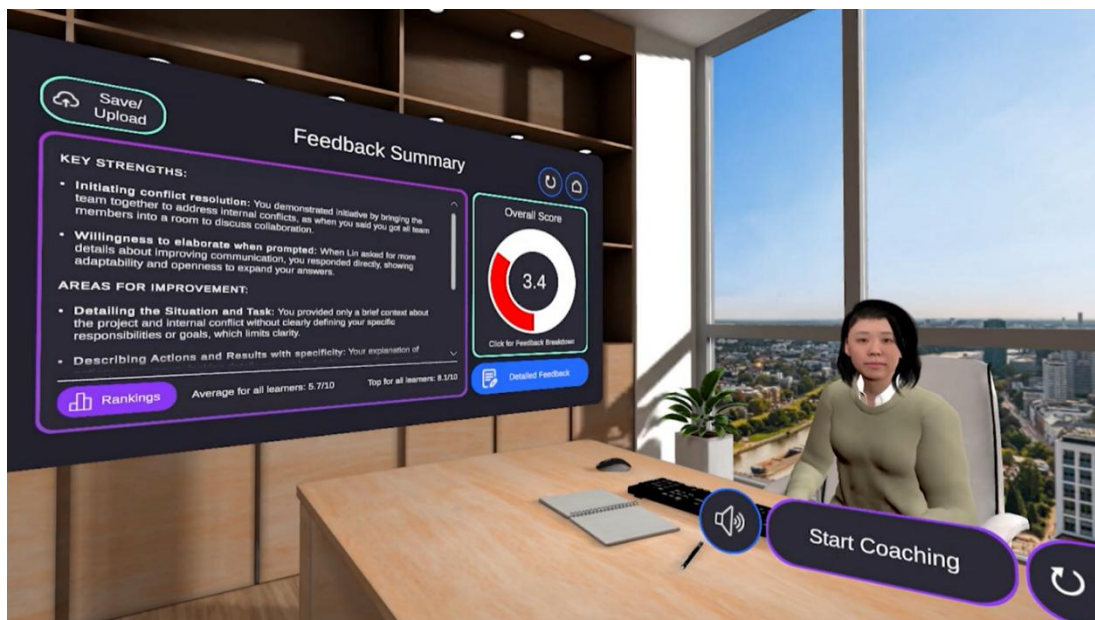
Aplikacja umożliwia elastyczne dostosowanie symulacji m.in. poprzez wybór ich typu, lokalizacji oraz awataru. Rysunek 17 przedstawia ekran wyboru typu symulacji, który stanowi pierwszy z kroków konfiguracji rozmowy.



Rysunek 17 przedstawia kadr z wideo z wyborem symulacji. Źródło: [17]

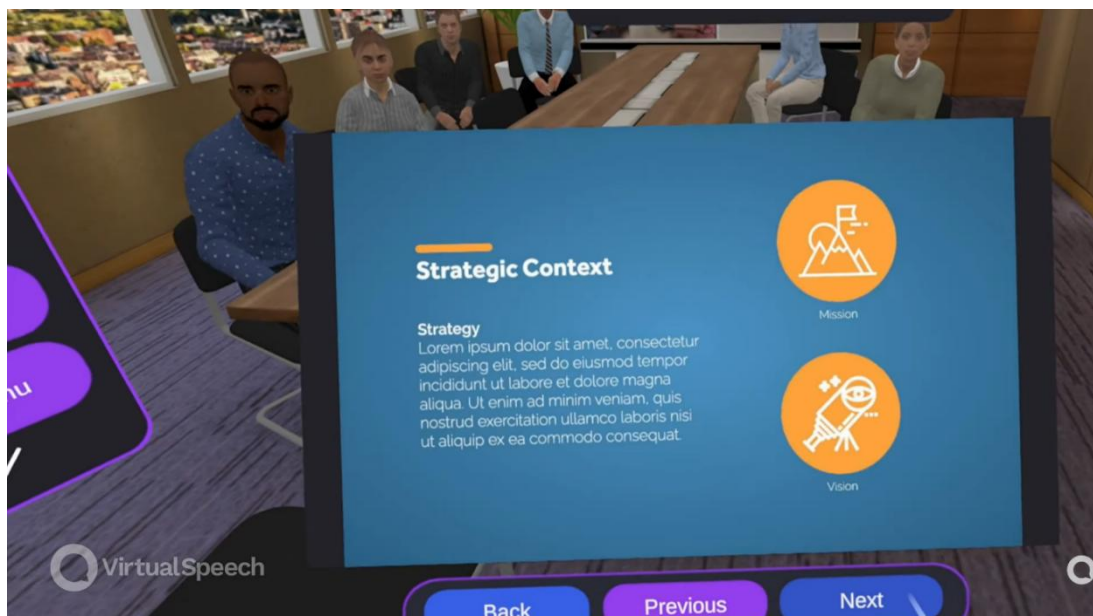
Na Rysunek 18 widoczny jest natomiast kadr z podsumowaniem symulacji, które są dokonywane przez sztuczną inteligencję i uwzględniają m.in.:

- ocenę zachowania i mowy ciała (kontakt wzrokowy, mowa ciała);
- analizę wraz z oceną treści i struktury wypowiedzi (trafność argumentów, logika, perswazja);
- analizę mowy (ton głosu, tempo mówienia, kontakt wzrokowy, użycie słów-wypełniaczy).



Rysunek 18 Kadr z wideo przedstawiającego symulację rozmowy o pracę. Źródło: [17]

Po opłaceniu dodatkowych funkcjonalności rozwiązanie umożliwia również dodanie własnych prezentacji czy notatek, które będą częścią symulacji (Rysunek 19).



Rysunek 19 Kadr z wideo przedstawiającego symulację prezentacji biznesowej. Źródło: [17]

Dużą zaletą rozwiązania jest jego elastyczność, osiągnięta dzięki zasileniu awatarów sztuczną inteligencją. Dzięki temu symulacje nie ograniczają się do sztywnych, z góry zaprogramowanych scenariuszy, tak jak miało to miejsce w przypadku wcześniej opisanych rozwiązań. Dodatkowym atutem są dostępne po zakończeniu symulacji raporty z rozmowy, które dostarczają cennych informacji zwrotnych. Pozwala to nie tylko na opanowanie stresu związanego z sytuacją, ale i na rozwój szeroko rozumianych umiejętności komunikacyjnych.

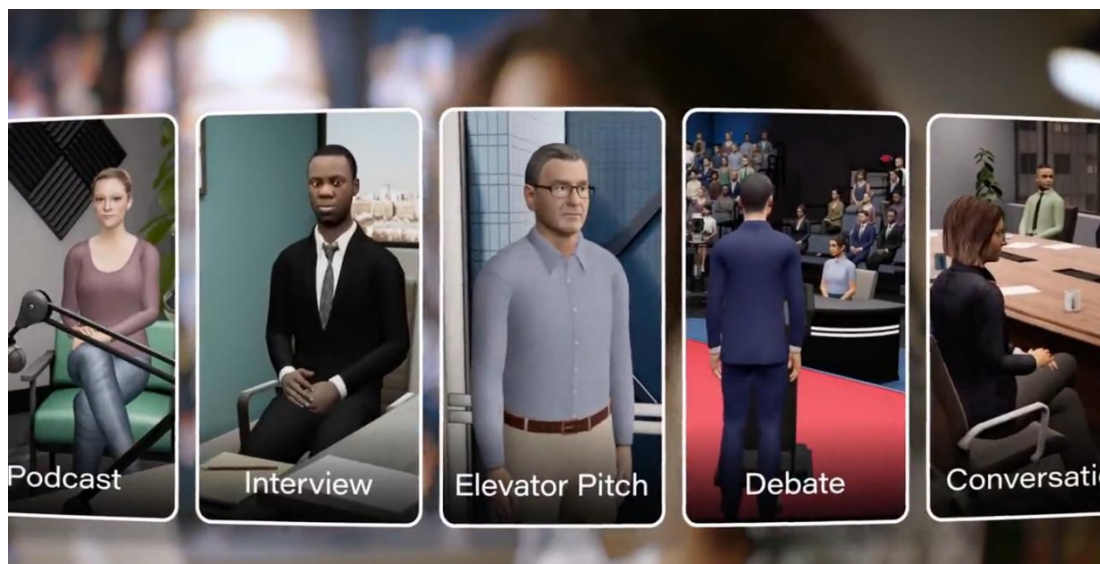
Natomiast potencjalne ograniczenie rozwiązania VirtualSpeech może stanowić brak wsparcia zarówno dla mobilnych rozwiązań VR opartych na smartfonach, takich jak Google Cardboard czy Merge VR, jak i dla klasycznych aplikacji mobilnych niewykorzystujących technologii VR. Oznacza to, że użytkownik nie może korzystać z platformy ani w formie prostego rozwiązania VR uruchamianego za pomocą smartfona, ani jako standardowej aplikacji mobilnej, co może zmniejszać elastyczność korzystania z rozwiązania oraz ograniczać jego dostępność dla osób, które nie posiadają dedykowanego zestawu VR lub komputera spełniającego wymagania platformy.

3.1.8 Ovation

Ovation [18] to kolejna aplikacja podobna do VirtualSpeech oraz bliska założeniom projektu Stressless VR. Rozwiązanie to pozwala na trening różnorodnych sytuacji, takich jak:

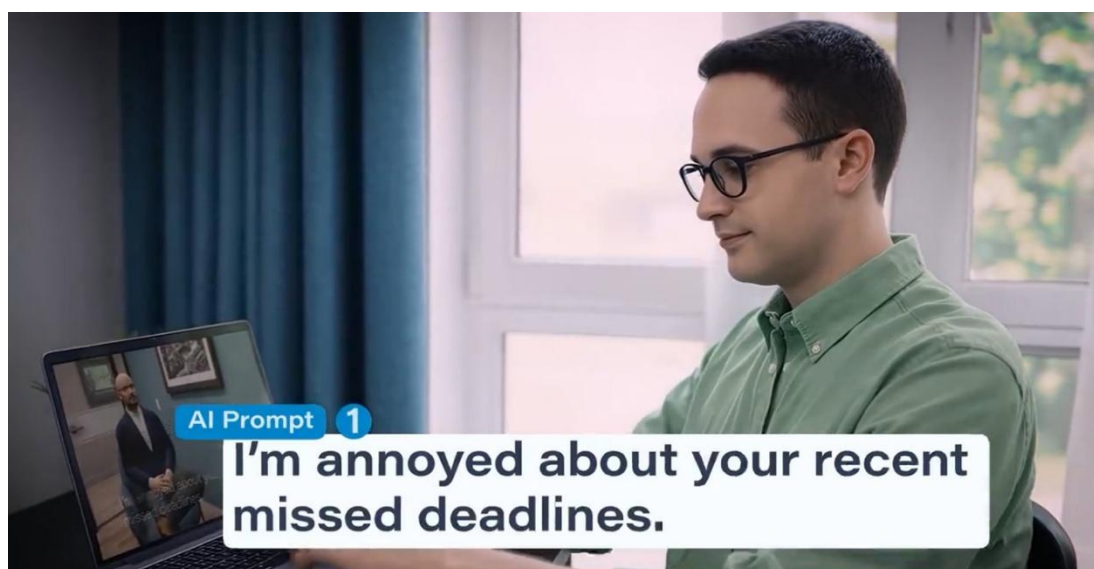
- debata,
- rozmowa o pracę,
- konferencja prasowa,
- rozmowa z inwestorem,
- wystąpienia w sali sądowej,
- rozmowa z pacjentem w gabinecie lekarskim,
- krótka prezentacja pomysłu (ang. *elevator pitch*),
- spotkania indywidualne z przełożonym (ang. *1-on-1*).

Pierwszy etap konfiguracji symulacji obejmuje wybór otoczenia, co zaprezentowano na Rysunek 20.



Rysunek 20 Kadr z wideo prezentującego działanie aplikacji Ovation - wybór otoczenia. Źródło: [19]

Po wyborze otoczenia użytkownik zostaje przeniesiony do wirtualnego środowiska, w którym może rozpocząć trening. Awatary będące częścią symulacji mogą pełnić funkcję słuchaczy, rozmówców, rekruterów, inwestorów lub innych uczestników danej sytuacji. W trakcie ćwiczenia użytkownik reaguje na pytania, wypowiada przygotowane treści lub prowadzi interakcję zgodnie z wybranym scenariuszem określonym przy użyciu *promptów* (Rysunek 21).

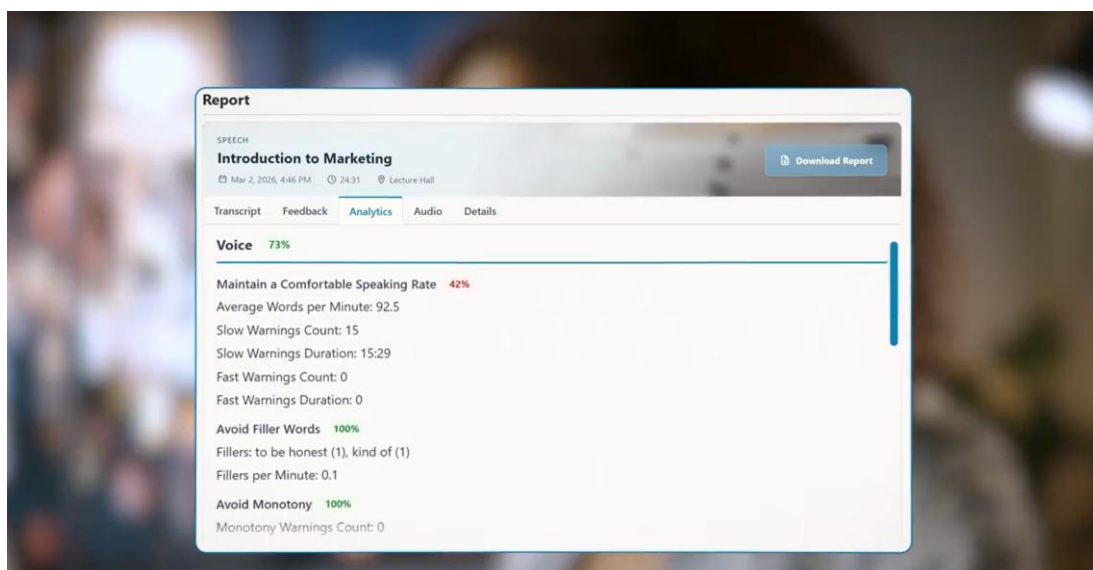


Rysunek 21 Kadr z wideo prezentującego działanie aplikacji Ovation - wprowadzanie promptu. Źródło: [19]

Po zakończeniu symulacji aplikacja generuje raport podsumowujący przebieg ćwiczenia (Rysunek 22), który zawiera elementy, takie jak:

- transkrypcja rozmowy,
- dane dotyczące nagrania audio,
- szczegóły przeprowadzonej symulacji,
- informacje zwrotne (ang. *feedback*) dotyczące części merytorycznej wypowiedzi,
- analizę wystąpienia obejmującą m.in. tempo mówienia czy wykryte słowa-wypełniacze.

Dzięki temu użytkownik ma możliwość nie tylko lepszego opanowania stresu związanego z daną sytuacją, lecz także analizy własnego zachowania, identyfikacji mocnych i słabych stron oraz dalszego doskonalenia umiejętności komunikacyjnych.



Rysunek 22 Kadr z wideo prezentującego działanie aplikacji Ovation - podsumowanie rozmowy. Źródło: [19]

Zalety i ograniczenia aplikacji Ovation są zatem w dużej mierze analogiczne do rozwiązania VirtualSpeech. Ovation, podobnie jak VirtualSpeech, nie jest przeznaczona na urządzenia mobilne, ponieważ treningi mogą być prowadzone z wykorzystaniem gogli VR lub komputera/laptopa. Może to ograniczać dostępność rozwiązania dla użytkowników, którzy nie posiadają odpowiedniego sprzętu lub oczekują możliwości korzystania z aplikacji w bardziej mobilnej formie.

3.2. Aplikacje mobilne

Niniejszy podrozdział poświęcony jest analizie wybranych aplikacji mobilnych, których celem jest przygotowanie użytkowników do różnego rodzaju sytuacji stresowych np. sytuacji związanych z komunikacją.

3.2.1 Orai for Public Speaking, Pres

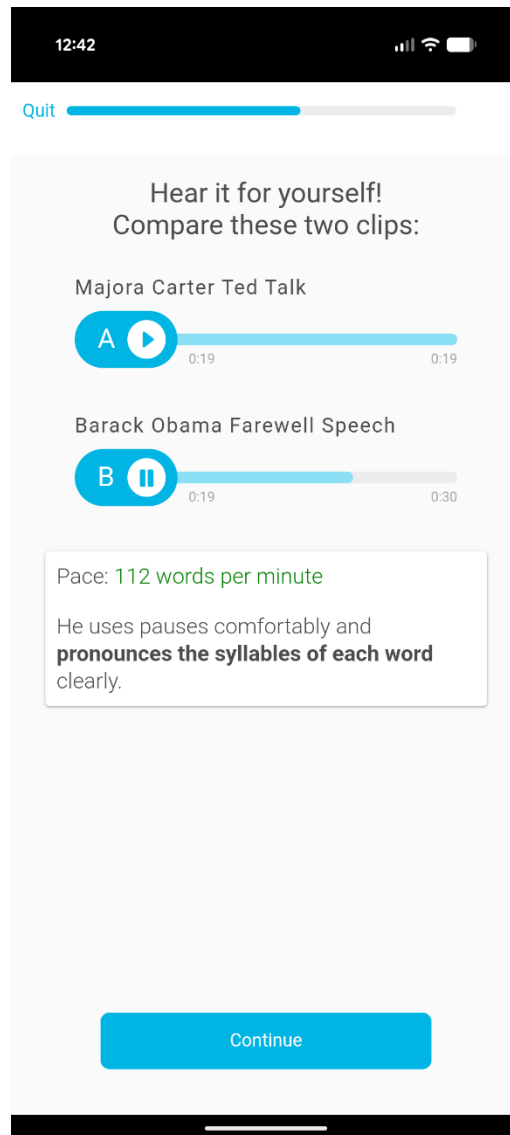
Orai [20] to aplikacja mobilna umożliwiająca rozwijanie umiejętności związanych zarówno z komunikacją biznesową, jak i codzienną. Aplikacja oferuje różnego rodzaju ćwiczenia pozwalające na stopniowe budowanie pewności siebie oraz doskonalenie określonych elementów wypowiedzi, takich jak:

- zwięzłość treści,
- stosowanie pauz,
- tempo mówienia,
- intonacja i głośność,
- redukcja użycia słów-wypełniaczy.

Użytkownik wybiera element, nad którym chciałby popracować, a następnie realizuje określone ćwiczenia, takie jak:

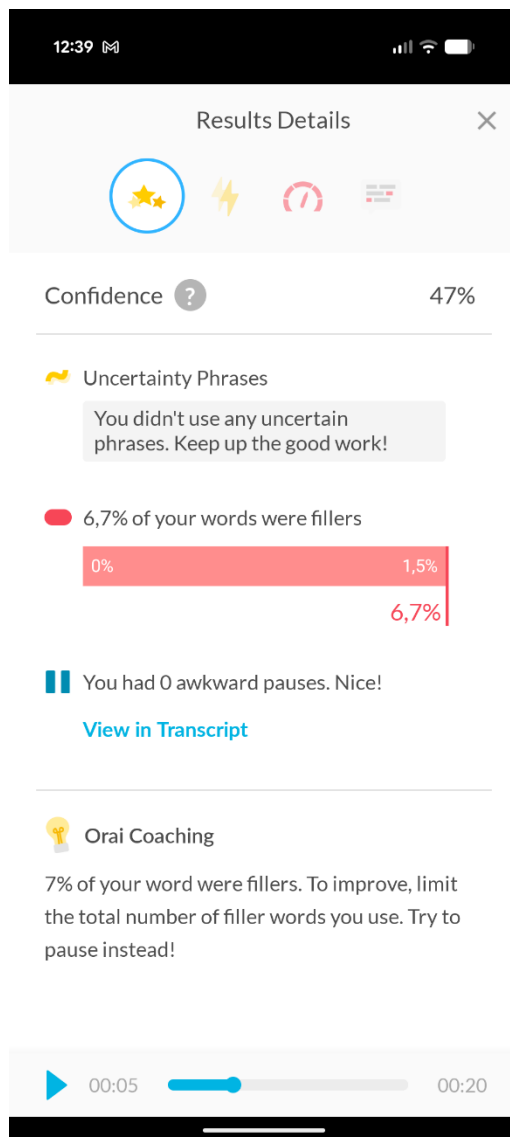
- swobodne wypowiadanie się na dany temat;

- słuchanie przemówień ze wskazaniem ich mocnych i słabych stron (Rysunek 23);
- odczytywanie fragmentu tekstu ukierunkowane np. na stosowanie prawidłowych pauz w wypowiedzi.



Rysunek 23 Porównanie przemówień. Źródło: Opracowanie własne.

Użytkownik nagrywa swoje wypowiedzi, które następnie są analizowane przez AI czego wynikiem są raporty wskazujące na elementy wymagające poprawy. Rysunek 24 przedstawia przykładową analizę.

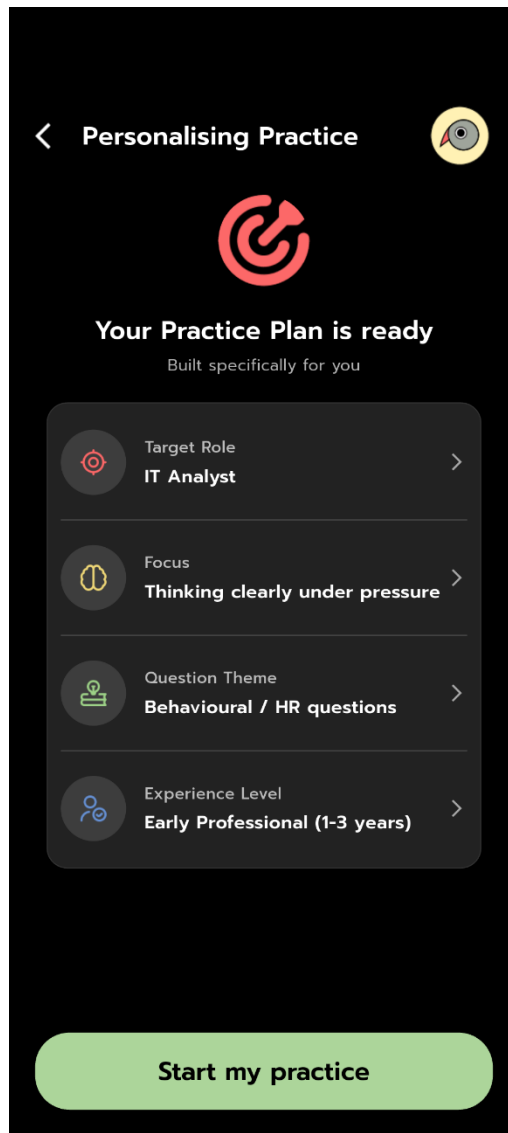


Rysunek 24 Raport z wystąpienia. Źródło: Opracowanie własne.

Mocną stroną aplikacji Orai są raporty podsumowujące każdą wypowiedź, dzięki którym użytkownik otrzymuje informację zwrotną na temat obszarów wymagających dalszej pracy. Aplikacja analizuje m.in. strukturę, czytelność, trafność oraz skuteczność komunikacyjną wypowiedzi. Należy jednak zauważyć, że rozwiązanie to nie weryfikuje w pełni merytorycznej poprawności prezentowanych treści. Ograniczeniem aplikacji jest również brak możliwości prowadzenia dialogu, co może być istotne w przypadku potrzeby ćwiczenia dwustronnej rozmowy.

3.2.2 TalkReady: AI Interview Prep

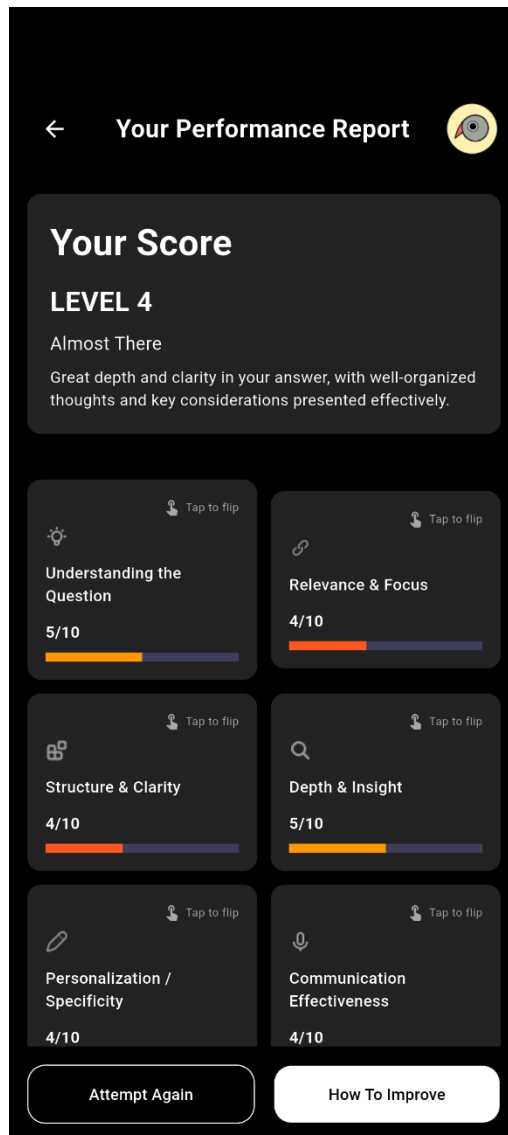
TalkReady [21] to aplikacja mobilna wykorzystująca sztuczną inteligencję, która umożliwia przygotowanie do rozmowy kwalifikacyjnej. Użytkownik ma możliwość wyboru stanowiska, którego ma dotyczyć rozmowa, oraz konfiguracji jej parametrów. Rysunek 25 przedstawia przykładową konfigurację rozmowy na stanowisko Analityka IT.



Rysunek 25 Konfiguracja rozmowy kwalifikacyjnej. Źródło: Opracowanie własne.

Użytkownik nagrywa odpowiedź na wygenerowane pytanie, a następnie otrzymuje ocenę elementów (Rysunek 26), takich jak:

- zrozumienie pytania,
- struktura i przejrzystość,
- głębia odpowiedzi i poziom refleksji,
- indywidualizacja i konkretność odpowiedzi.



Rysunek 26 Podsumowanie odpowiedzi. Źródło: Opracowanie własne.

Podobnie jak Orai, TalkReady wyróżnia się raportami umożliwiającymi użytkownikowi dalszy rozwój w obszarach, które uzyskały niezadowalające oceny. W przeciwieństwie do Orai, TalkReady koncentruje się jednak przede wszystkim na analizie merytorycznej treści odpowiedzi, obejmującej m.in. dopasowanie do pytania, poziom szczegółowości, logiczną strukturę oraz stopień wykorzystania konkretnych przykładów, a nie wyłącznie na sposobie jej przekazania.

Analogicznie do Orai, aplikacja nie umożliwia prowadzenia dialogu z cyfrowym awatarem, co może stanowić istotne ograniczenie w kontekście ćwiczenia dwustronnej rozmowy. Dodatkowo należy zauważyć, że TalkReady jest rozwiązaniem wyspecjalizowanym w jednym typie sytuacji stresowej, jaką jest rozmowa kwalifikacyjna. W odróżnieniu od bardziej elastycznych platform nie obejmuje szerszego zakresu scenariuszy komunikacyjnych, takich jak wystąpienia publiczne, rozmowy z przełożonym, negocjacje czy interakcje społeczne.

3.3. Podsumowanie

Dostępne na rynku rozwiązania wspierające osvajanie użytkowników z sytuacjami stresowymi można analizować z kilku perspektyw. Pierwszą z nich jest poziom immersyjności, który może istotnie wpływać na skuteczność treningu. Wykorzystanie wysokiej jakości środowiska wirtualnego stanowi dużą zaletę, ponieważ pozwala użytkownikowi w większym stopniu zanurzyć się w symulowanej sytuacji i przeciwżyć reakcje w warunkach zbliżonych do rzeczywistych.

Drugą istotną perspektywą jest poziom specjalizacji danego rozwiązania oraz stopień jego elastyczności, który coraz częściej wiąże się z wykorzystaniem sztucznej inteligencji. Zastosowanie AI pozwala tworzyć bardziej dynamiczne środowiska treningowe, mniej zależne od sztywnych, wcześniej zaprogramowanych scenariuszy. Zdecydowana większość dostępnych na rynku rozwiązań opartych na technologii wirtualnej rzeczywistości charakteryzuje się jednak wysokim stopniem specjalizacji. Symulacje są najczęściej realizowane według konkretnego scenariusza, co dobrze sprawdza się w przypadku szkoleń proceduralnych, lecz może ograniczać możliwość swobodnego reagowania użytkownika.

Wyjątek stanowią m.in. aplikacje VirtualSpeech oraz Ovation. Chociaż koncentrują się one głównie na tematyce komunikacji, dzięki wykorzystaniu wirtualnych awatarów wspieranych przez sztuczną inteligencję nie ograniczają się wyłącznie do jednego, z góry określonego przebiegu rozmowy. Pozwalają tym samym na większą elastyczność interakcji oraz lepsze dopasowanie symulacji do potrzeb użytkownika. Jednakże narzędzi, które w elastyczny sposób umożliwiałyby stopniowe osvajanie się z różnorodnymi sytuacjami stresowymi dnia codziennego, zarówno w życiu prywatnym, jak i zawodowym, nadal jest stosunkowo niewiele.

Jeszcze większy stopień specjalizacji widoczny jest w przypadku aplikacji mobilnych, nawet tych ukierunkowanych na rozwój kompetencji komunikacyjnych. Przykłady aplikacji Orai i TalkReady pokazują wyraźne różnice w założeniach tego typu narzędzi. Orai skupia się przede wszystkim na analizie sposobu mówienia, obejmującej m.in. tempo wypowiedzi, ton głosu, głośność, stosowanie pauz oraz użycie słów-wypełniaczy. TalkReady koncentruje się natomiast głównie na analizie merytorycznej treści odpowiedzi, uwzględniając jej trafność, strukturę, poziom szczegółowości oraz mocne i słabe strony wypowiedzi. Oba rozwiązania dostarczają użytkownikowi wartościowych informacji zwrotnych, jednak nie oferują pełnej, dynamicznej rozmowy z cyfrowym rozmówcą.

Istotnym kryterium oceny analizowanych rozwiązań jest również ich dostępność. Konieczność posiadania odpowiedniego sprzętu, takiego jak gogle VR lub komputer spełniający określone wymagania techniczne, może ograniczać możliwość korzystania z aplikacji opartych na wirtualnej rzeczywistości. Z tego względu dużą zaletą aplikacji mobilnych jest ich łatwiejsza dostępność oraz możliwość uruchomienia na urządzeniu, które większość użytkowników posiada na co dzień. Niektóre aplikacje VR, np. Ovation, dostępne są również w wersji desktopowej, jednak komputer lub laptop nie zapewnia takiego poziomu mobilności jak smartfon.

Najbardziej pożądanym kierunkiem wydaje się zatem model hybrydowy, który łączyłby zalety środowiska wirtualnego z dostępnością charakterystyczną dla aplikacji mobilnych. Takie rozwiązanie mogłoby umożliwiać użytkownikowi trening w immersyjnym środowisku, a jednocześnie nie wymagać korzystania ze specjalistycznego sprzętu.

Luka rynkowa widoczna jest zatem szczególnie w obszarze narzędzi, które pozwalałyby na elastyczne, stopniowe osvajanie się z różnorodnymi sytuacjami stresowymi, umożliwiały swobodne formułowanie wypowiedzi oraz zapewniały dostępność na urządzeniach mobilnych.

4. Koncepcja aplikacji Stressless VR

W ramach niniejszego rozdziału omówiono projekt techniczny aplikacji Stressless VR. Przedstawione zostały m.in. wymagania funkcjonalne, niefunkcjonalne, scenariusze przypadków użycia oraz architektura rozwiązania. Treść została wsparta matrycami walidacji w ramach scenariuszy przypadków użycia, a także diagramem przypadków użycia, aktywności i stanów.

4.1. Słownik pojęć

Tabela 1 zawiera słownik pojęć.

Tabela 1 Słownik pojęć. Źródło: Opracowanie własne.

Pojęcie	Definicja pojęcia
AI	AI (ang. <i>Artificial Intelligence</i>), czyli sztuczna inteligencja, najczęściej rozumiana jako systemy, które wykazują inteligentne zachowanie poprzez analizowanie swojego otoczenia i podejmowanie działań – z pewnym stopniem autonomii – w celu osiągnięcia określonych celów [22]. Termin ten często stosuje się wymiennie z nazwami dyscyplin podrzędnych, do których należą uczenie maszynowe (ang. <i>machine learning</i> , <i>ML</i>), a także uczenie głębokie (ang. <i>deep learning</i>) [23].
Akustyczna analiza sposobu wypowiedzi	Akustyczna analiza sposobu wypowiedzi to automatyczna analiza cech sygnału głosowego użytkownika, obejmująca między innymi częstotliwość podstawową, głośność, tempo mowy, pauzy, jitter, shimmer, HNR oraz wartości formantów. Jej wyniki są wykorzystywane do wyznaczania metryk opisujących sposób wypowiedzi [24].
F0	F0 (podstawowa częstotliwość głosu, ang. <i>fundamental frequency</i>) to najniższa częstotliwość drgań fałd głosowych podczas fonacji, wyrażana w hercach (Hz). Odpowiada za postrzeganą wysokość głosu – wyższe F0 oznacza wyższy ton wypowiedzi. Stabilność i zakres zmienności F0 są wskaźnikami pewności siebie oraz ekspresji mówcy [25].
F1 i F2	F1 i F2 (formanty, ang. <i>formants</i>) to charakterystyczne pasma częstotliwościowe wzmocnione przez rezonans traktu głosowego (jamy ustnej, gardłowej i nosowej). Pierwszy formant F1 związany jest głównie z pionowym położeniem języka – niskie F1 odpowiada wysokiej pozycji języka, wysokie F1 – niskiej. Drugi formant F2 odzwierciedla poziome położenie języka – wysokie F2 charakteryzuje samogłoski przednie (np. <i>i</i>), niskie F2 – tylne (np. <i>u</i>). Wartości F1 i F2 są kluczowymi wskaźnikami jakości artykulacji samogłosek i precyzji wymowy [25].
Jitter	Jitter, czyli miara krótkookresowych zmian długości kolejnych okresów drgań fałdów głosowych. Podwyższone wartości jittera mogą wskazywać na niestabilność częstotliwości głosu [26].
Kurtosis	Kurtosis, czyli miara statystyczna opisująca kształt rozkładu wartości oraz występowanie obserwacji znacząco odbiegających od wartości typowych. W systemie jest wykorzystywana do analizy zmienności jittera i shimmera [27].

Pojęcie	Definicja pojęcia
LLM	LLM (ang. <i>Large Language Model</i>), czyli duży model językowy będący jednym z powszechnie stosowanych modeli sztucznej inteligencji, który ma na celu m.in generowanie odpowiedzi, streszczeń czy tłumaczeń w języku naturalnym zrozumiałym dla człowieka na podstawie danych wejściowych przekazanych w postaci <i>promptów</i> [28].
Metryka akustyczna	Metryka akustyczna to wskaźnik opisujący wybrany aspekt sposobu wypowiedzi użytkownika, wyznaczany na podstawie jednej lub kilku cech sygnału audio.
MoSCoW	Metoda MoSCoW to jedna z metod priorytetyzacji zadań i wymagań, wykorzystywana podczas analizy biznesowej oraz podczas tworzenia oprogramowania. Pozwala ona dzielić poszczególne elementy systemu na cztery główne kategorie: 1) Must (Musi być), 2) Should (Powinno być), 3) Could (Może być), 4) Won't (Nie będzie / Odłożone w czasie). Przypisanie odpowiedniego priorytetu ułatwia skupienie się na dostarczeniu niezbędnych funkcjonalności w określonym terminie [29].
Prompt	Zapytanie lub instrukcja skierowana do modelu sztucznej inteligencji będąca częścią interakcji człowiek-AI i inicjująca proces generowania odpowiedzi przez wybrany model.
RAG	RAG (ang. <i>Retrieval-Augmented Generation</i>) to technika łącząca wyszukiwanie informacji (ang. <i>retrieval</i>) z generowaniem tekstu (ang. <i>generation</i>). W przeciwieństwie do tradycyjnych modeli językowych, które generują odpowiedzi głównie na podstawie wiedzy zdobytej podczas procesu trenowania, podejście RAG rozszerza ten proces o dodatkowy etap wyszukiwania informacji w zewnętrznych źródłach danych. W pierwszej kolejności system identyfikuje dokumenty lub fragmenty dokumentów istotne dla zadanego zapytania, a następnie wykorzystuje je jako dodatkowy kontekst podczas generowania odpowiedzi. Dzięki temu możliwe jest uwzględnienie bardziej aktualnych i specyficznych informacji, które nie były dostępne na etapie trenowania modelu [30].
Shimmer	Shimmer, czyli miara krótkookresowych zmian amplitudy kolejnych okresów drgań fałdów głosowych. Podwyższone wartości shimmera mogą wskazywać na niestabilność głośności głosu [26].

4.2. Wymagania funkcjonalne

Tabela 2 zawiera wymagania funkcjonalne dla aplikacji Stressless VR, którym zostały nadane priorytety zgodnie z metodą MoSCoW.

Tabela 2 Wymagania funkcjonalne aplikacji Stressless VR. Źródło: Opracowanie własne.

ID	Nazwa wymagania	Opis wymagania	Priorytet
Konfiguracja symulacji			
WF.01	Wybór sceny	1) Aplikacja umożliwia użytkownikowi wybór sceny, w ramach której będzie ćwiczył określoną sytuację. a) W aplikacji są dostępne sceny, takie jak: biuro oraz sala lekcyjna. 2) W aplikacji są dostępne dwie kategorie scen: a) sceny zoptymalizowane pod kątem urządzeń mobilnych, wykonane w stylu <i>low poly</i>, b) sceny o wyższym poziomie szczegółowości, przeznaczone dla urządzeń o większej wydajności. 3) Lista wyboru scen zawiera pozycje odpowiednie dla urządzenia, na którym została uruchomiona aplikacja. a) Na urządzeniach mobilnych sceny są ograniczone do kategorii <i>low poly</i>.	M
WF.02	Wprowadzenie promptu	1) Aplikacja umożliwia wprowadzenie krótkiego opisu sytuacji lub celu treningu. 2) Wprowadzony <i>prompt</i> jest wykorzystywany przez model językowy do określenia kontekstu oraz przebiegu symulacji.	M
WF.03	Dodanie źródła wiedzy	1) Aplikacja umożliwia użytkownikowi dodanie pliku, który będzie wykorzystywany przez wybrany model językowy jako kontekst źródłowy w mechanizmie RAG. a) Odpowiedzi cyfrowego awatara odnoszą się do informacji zawartych w przesłanym pliku. 2) Aplikacja akceptuje następujące rozszerzenia plików: .txt, .pdf. a) Aplikacja blokuje dodanie pliku o rozszerzeniu innym niż dopuszczalne.	S
WF.04	Wybór modelu LLM	1) Aplikacja umożliwia użytkownikowi wybór modelu językowego, który będzie odpowiadał za generowanie wypowiedzi cyfrowego awatara w trakcie symulacji. 2) Użytkownik ma możliwość wyboru jednego z dwóch typów modeli:	C

		a) modelu chmurowego Gemini, dostępnego za pośrednictwem zewnętrznego API; b) lokalnego modelu językowego uruchamianego i zarządzanego za pomocą Ollamy. 3) W przypadku wyboru modelu Gemini aplikacja umożliwia wprowadzenie klucza API, który będzie wykorzystywany do autoryzacji zapytań kierowanych do zewnętrznej usługi.	
WF.05	Wybór inicjatora rozmowy	1) Aplikacja umożliwia wybór inicjatora rozmowy, czyli określenie, czy symulację rozpoczyna użytkownik, czy cyfrowy awatar.	C
Przebieg symulacji			
WF.06	Uruchomienie symulacji	1) Aplikacja umożliwia użytkownikowi uruchomienie symulacji w momencie, gdy wszystkie obligatoryjne parametry konfiguracyjne są uzupełnione. 2) Aplikacja prezentuje stosowny komunikat i blokuje uruchomienie symulacji w przypadku braku konfiguracji obligatoryjnych parametrów.	M
WF.07	Rozmowa	1) Aplikacja umożliwia użytkownikowi swobodne wypowiedzianie się w trakcie symulacji za pomocą dedykowanego przycisku. 2) Aplikacja rozpoznaje mowę użytkownika, a następnie na jej podstawie generuje odpowiedź cyfrowego awataru. a) W czasie generowania odpowiedzi aplikacja wyświetla stosowną ikonę informacyjną. 3) Aplikacja prezentuje użytkownikowi treści wypowiedzi generowanych przez cyfrowego awataru w trakcie symulacji.	M
WF.08	Zakończenie i zapis symulacji	1) Aplikacja umożliwia użytkownikowi przerwanie symulacji w dowolnym momencie za pomocą dedykowanego przycisku. 2) Po zakończeniu symulacji aplikacja: a) zapisuje jej konfigurację oraz transkrypcję rozmowy, b) generuje raport (patrz WF.10).	M

		3) Po zakończeniu symulacji aplikacja wyświetla wygenerowany raport (patrz WF.09). a) W czasie generowania raportu aplikacja wyświetla ekran ładowania (ang. <i>loader</i>).	
Raport i statystyki			
WF.09	Wyświetlenie listy symulacji	1) Aplikacja umożliwia wyświetlenie listy przeprowadzonych symulacji.	C
WF.10	Wyświetlenie raportu	1) Aplikacja umożliwia wyświetlenie raportu z wybranej symulacji zawierającego akustyczną analizę sposobu wypowiedzi oraz merytoryczne podsumowanie przeprowadzonej rozmowy. 2) Akustyczna analiza sposobu wypowiedzi obejmuje następujące metryki wyznaczone dla każdej wypowiedzi użytkownika: a) poziom stresu głosowego – wyznaczany na podstawie jittera, shimmera oraz stosunku harmonicznego do szumów (ang. <i>Harmonics-to-Noise Ratio, HNR</i>); b) pewność siebie – oceniana na podstawie stabilności podstawowej częstotliwości głosu (ang. <i>fundamental frequency, F0</i>), głośności oraz aktywności fonacji; c) płynność mowy – określana na podstawie tempa wypowiedzi, stosunku czasu mówienia do czasu ciszy oraz długości pauz; d) ekspresja głosu – mierzona zakresem zmienności F0 oraz dynamiką głośności; e) napięcie głosowe – wyznaczone na podstawie kurtosis i skośności jittera oraz shimmera; f) jakość artykulacji – oceniana na podstawie wartości formantów F1 i F2. 3) Dla każdej metryki wyznaczany jest również trend zmian w trakcie sesji, informujący o tym czy jakość mowy użytkownika poprawiała się czy pogarszała w trakcie rozmowy. 4) Merytoryczne podsumowanie rozmowy generowane jest przez model językowy na podstawie transkrypcji sesji i obejmuje: a) omówienie głównych tematów poruszonych w rozmowie;	M

		b) ocenę jakości i spójności wypowiedzi użytkownika; c) mocne i słabe strony komunikacji; d) praktyczną wskazówkę dotyczącą poprawy skuteczności wypowiedzi.	
WF.11	Wyświetlenie transkrypcji	1) Aplikacja umożliwia wyświetlenie transkrypcji wybranej symulacji.	S
WF.12	Wyświetlenie konfiguracji symulacji	1) Aplikacja umożliwia wyświetlenie szczegółów wybranej symulacji z listy przeprowadzonych symulacji. 2) Szczegóły symulacji obejmują m.in. datę jej przeprowadzenia, użyty model LLM, wprowadzony prompt oraz powiązany raport (patrz WF.10).	C

4.3. Wymagania niefunkcjonalne

Tabela 3 zawiera wymagania niefunkcjonalne dla aplikacji Stressless VR, którym zostały nadane priorytety zgodnie z metodą MoSCoW.

Tabela 3 Wymagania niefunkcjonalne aplikacji Stressless VR. Źródło: Opracowanie własne.

Id	Nazwa wymagania	Opis wymagania	Priorytet
Wydajność			
WNF.01	Wydajność aplikacji	Aplikacja działa płynnie na urządzeniach spełniających minimalne wymagania sprzętowe, tj. Intel Core i5-8400 / AMD Ryzen 5 2600, 8 GB RAM, NVIDIA GTX 1060 6 GB / AMD RX 580 8 GB, 4 GB VRAM, 10 GB przestrzeni dyskowej, Windows 10 64-bit / macOS 12, Vulkan 1.0.	M
WNF.02	Czas ładowania symulacji	Aplikacja powinna zapewniać czas ładowania symulacji nieprzekraczający 10 s na urządzeniach spełniających minimalne wymagania sprzętowe.	S
WNF.03	Czas odpowiedzi modelu językowego	Aplikacja powinna zapewniać czas oczekiwania na odpowiedź cyfrowego awatara nieprzekraczający 10 s przy stabilnym połączeniu z wybranym modelem językowym, uruchomionym na maszynie spełniającej minimalne wymagania niezbędne do jego płynnego działania.	M

WNF.04	Czas generowania raportu	Aplikacja powinna zapewniać czas generowania raportu nieprzekraczający 30 s przy stabilnym połączeniu z wybranym modelem językowym, uruchomionym na maszynie spełniającej minimalne wymagania niezbędne do jego płynnego działania.	S
Bezpieczeństwo			
WNF.05	Bezpieczeństwo klucza API	W przypadku wykorzystania modelu chmurowego aplikacja przechowuje klucz API w sposób ograniczający ryzyko dostępu przez osoby nieuprawnione.	M
WNF.06	Ochrona danych użytkownika	Aplikacja chroni dane użytkownika przechowywane lub przetwarzane w systemie, w tym transkrypcje rozmów, raporty oraz pliki dodawane jako źródło wiedzy.	M
Kompatybilność			
WNF.07	Kompatybilność z urządzeniami mobilnymi	Aplikacja jest możliwa do uruchomienia na wybranych urządzeniach mobilnych spełniających minimalne wymagania techniczne, tj.: 2 GB RAM, 512 MB VRAM lub pamięci RAM współdzielonej, Snapdragon 660 / MediaTek Helio G85 lub odpowiednik, Android 8.0+, 3 GB przestrzeni dyskowej.	M
Skalowalność			
WNF.08	Modularność integracji z LLM	Integracja z modelami językowymi jest zaprojektowana w sposób umożliwiający dodanie kolejnych dostawców modeli chmurowych oraz lokalnych.	M
WNF.09	Możliwość rozbudowy scen	Struktura aplikacji umożliwia dodanie kolejnych scen w przyszłości bez konieczności przebudowy głównych mechanizmów działania systemu.	M
Użyteczność			
WNF.10	Czytelność raportu	Raport z symulacji jest przedstawiony w sposób przejrzysty, umożliwiając łatwą identyfikację mocnych stron oraz obszarów wymagających poprawy.	M

4.4. Wysokopoziomowa architektura rozwiązania

Na Rysunek 27 przedstawiono wysokopoziomową architekturę aplikacyjną opracowanego rozwiązania. System składa się z aplikacji klienckiej Stressless VR, utworzonej z wykorzystaniem

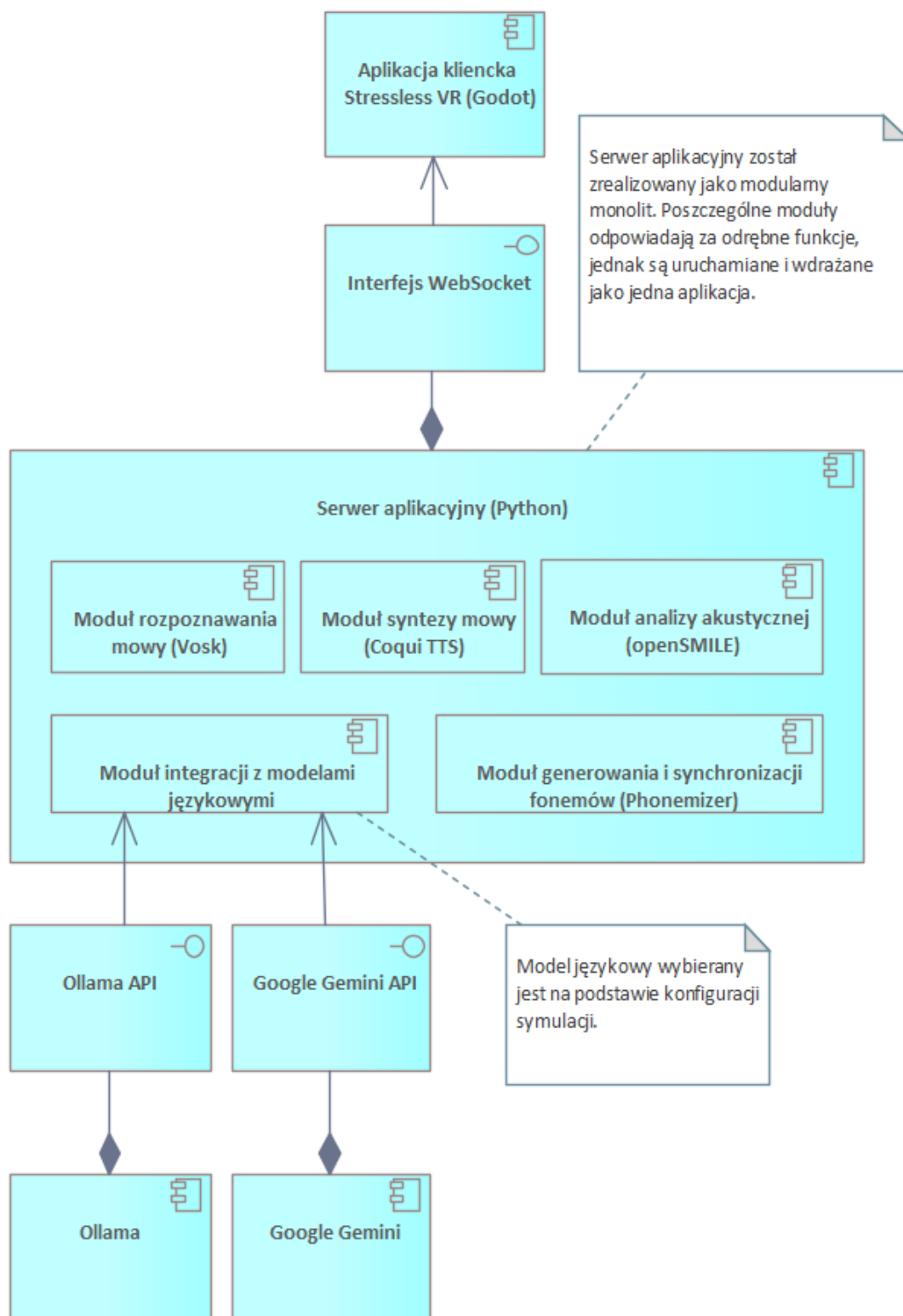
silnika Godot, serwera aplikacyjnego zaimplementowanego w języku Python oraz zewnętrznych modeli językowych. Komunikacja pomiędzy aplikacją kliencką a serwerem odbywa się za pośrednictwem interfejsu WebSocket, który umożliwia dwukierunkowe przesyłanie danych, w tym nagrań wypowiedzi użytkownika, wygenerowanego dźwięku oraz informacji wykorzystywanych do animowania cyfrowego awatara.

Serwer aplikacyjny został zrealizowany zgodnie z podejściem modularnego monolitu. Oznacza to, że poszczególne obszary funkcjonalne zostały wydzielone w postaci odrębnych modułów, jednak są uruchamiane i wdrażane jako jedna aplikacja. W ramach serwera wyróżniono:

- moduł rozpoznawania mowy wykorzystujący bibliotekę Vosk,
- moduł syntezy mowy oparty na Coqui TTS,
- moduł analizy akustycznej wykorzystujący openSMILE,
- moduł generowania i synchronizacji fonemów wykorzystujący bibliotekę Phonemizer,
- moduł integracji z modelami językowymi.

Moduł integracji z modelami językowymi odpowiada za komunikację z wybranym modelem LLM. Rozwiązanie umożliwia wykorzystanie lokalnie uruchamianych modeli udostępnianych za pośrednictwem interfejsu Ollama API lub modelu Google Gemini dostępnego przez Google Gemini API. Model używany w trakcie danej symulacji jest wybierany na podstawie jej konfiguracji. Taka budowa umożliwia wymianę lub dodawanie kolejnych modeli językowych bez konieczności wprowadzania istotnych zmian w pozostałych modułach systemu.

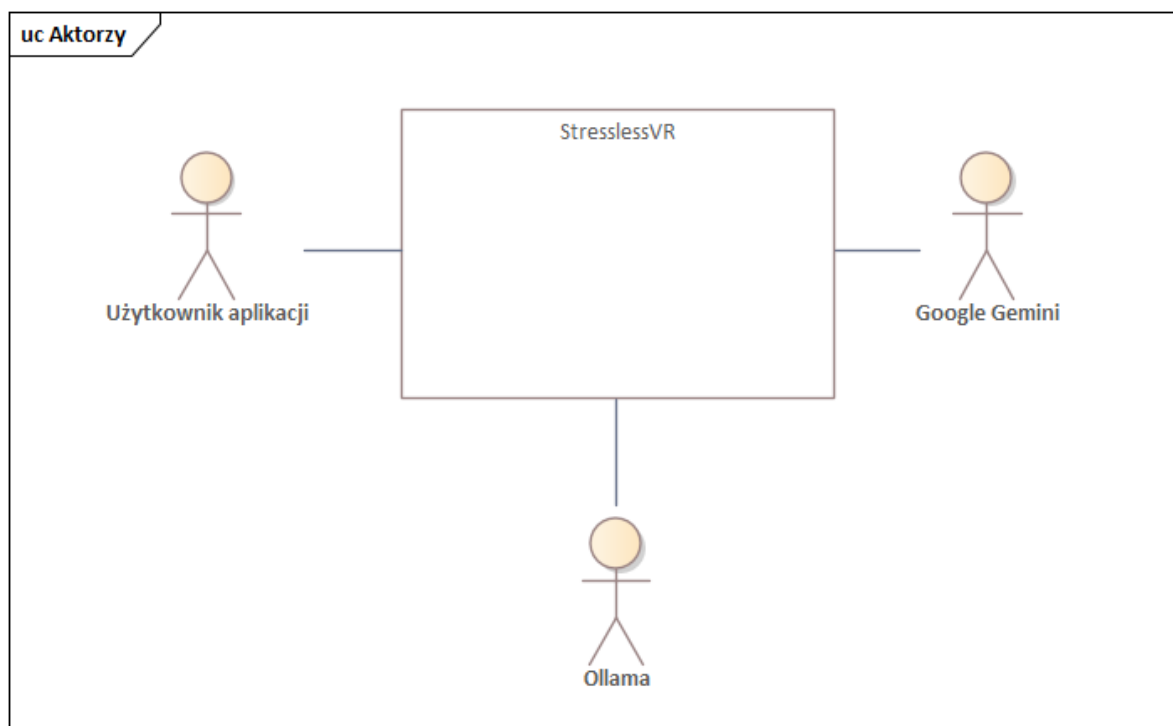
application Wysokopoziomowa architektura aplikacyjna rozwiązania



Rysunek 27 Wysokopoziomowa architektura aplikacyjna rozwiązania. Źródło: Opracowanie własne.

4.5. Charakterystyka aktorów

Rysunek 28 przedstawia aktorów aplikacji Stressless VR.



Rysunek 28 Diagram aktorów. Źródło: Opracowanie własne.

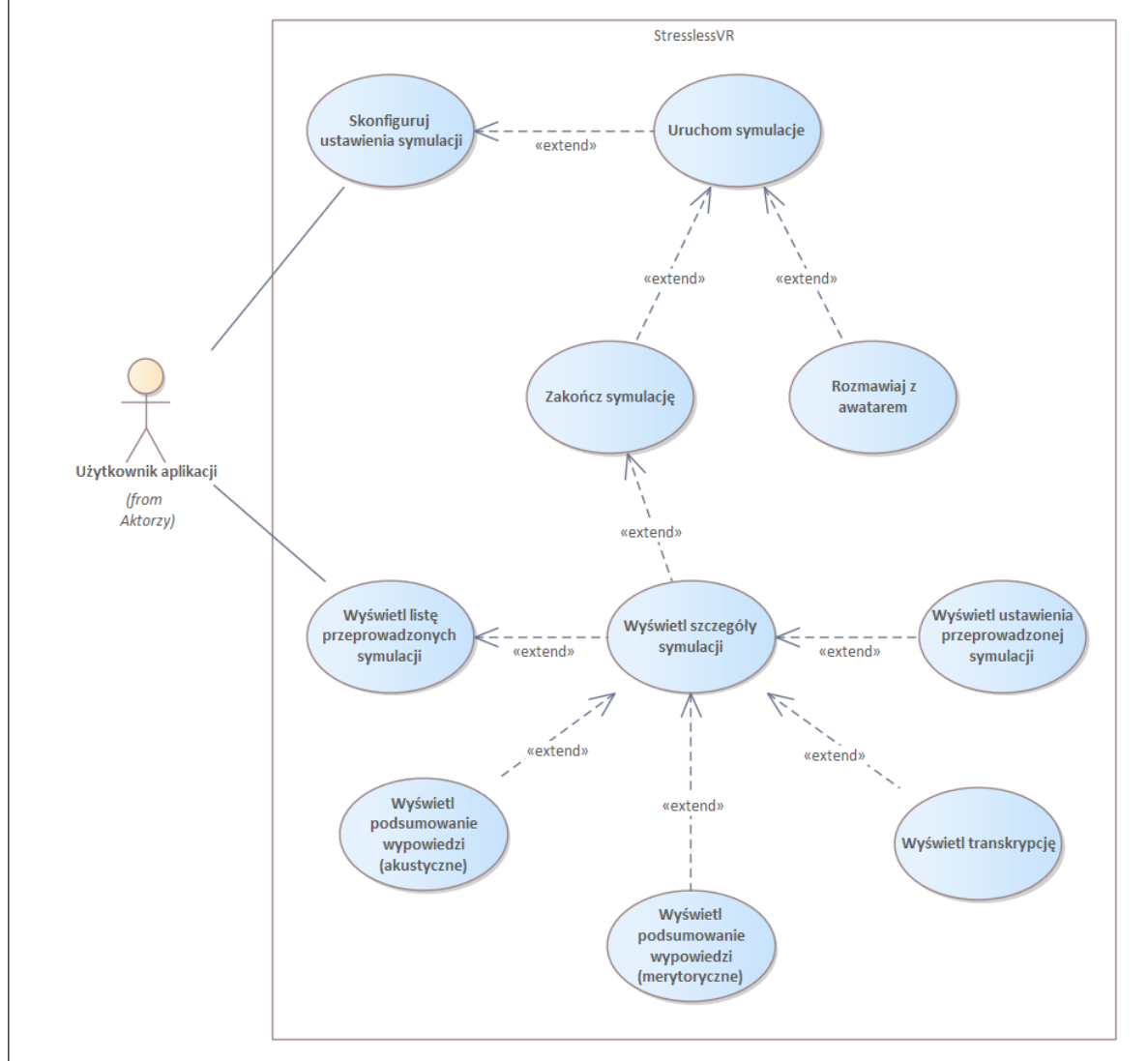
Głównym aktorem biznesowym jest użytkownik aplikacji, który korzysta z dostępnych funkcji systemu, między innymi przeprowadza symulacje, prowadzi rozmowę z cyfrowym awatarem oraz przegląda raporty i transkrypcje zakończonych rozmów. Na diagramie przedstawiono również aktorów systemowych reprezentujących rozwiązania zewnętrzne współpracujące z systemem StresslessVR.

Google Gemini oraz Ollama umożliwiają wykorzystanie modeli językowych przez aplikację. System StresslessVR komunikuje się z nimi w celu generowania odpowiedzi na kolejne wypowiedzi użytkownika. Modele językowe są również wykorzystywane do przygotowywania podsumowań symulacji na podstawie transkrypcji rozmowy oraz wyników analizy akustycznej wypowiedzi użytkownika.

W trakcie jednej symulacji wykorzystywany jest model językowy wybrany na podstawie jej konfiguracji. Zastosowanie dwóch alternatywnych rozwiązań pozwala korzystać zarówno z modeli uruchamianych lokalnie za pośrednictwem platformy Ollama, jak i z modelu Google Gemini udostępnianego jako usługa zewnętrzna.

4.6. Przypadki użycia

W niniejszym podrozdziale przedstawiono scenariusze przypadków użycia w podziale na grupy funkcjonalne. Rysunek 29 przedstawia diagram przypadków użycia.



Rysunek 29 Diagram przypadków użycia. Źródło: Opracowanie własne.

4.6.1 Konfiguracja symulacji

Tabela 4 zawiera scenariusz przypadku użycia związanego z konfiguracją symulacji.

Tabela 4 Scenariusz przypadku użycia PU.KONFIG.01. Źródło: Opracowanie własne.

ID przypadku użycia	PU.KONFIG.01
Nazwa przypadku użycia	Skonfiguruj ustawienia symulacji
ID wymagania	WF.01-WF.05
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia konfigurację ustawień symulacji.

Wyzwalacze	Aktor wybiera opcję „Start”.
Warunki początkowe	Aktor znajduje się w menu głównym aplikacji.
Warunki końcowe	Skonfigurowane ustawienia symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System prezentuje widok konfiguracji symulacji 3) Aktor konfiguruje ustawienia symulacji
Scenariusz alternatywny	Alt 3 Aktor nie zmienia domyślnej konfiguracji → END
Informacje dodatkowe	1) Parametry konfiguracyjne symulacji są domyślnie zdefiniowane zgodnie z matrycą walidacji. 2) Aktor może uruchomić symulację bez modyfikowania domyślnych parametrów konfiguracyjnych. 3) Na ekran główny składają się następujące przyciski: a) Start b) Statistics c) Exit

Tabela 5 przedstawia matrycę walidacji dla przypadku użycia „Skonfiguruj ustawienia symulacji”.

Tabela 5 Matryca walidacji do PU.KONFIG.01. Źródło: Opracowanie własne.

Nazwa pola	Typ pola	Czy obowiązkowe	Właściwości	Błąd użytkownika	Komunikat walidacyjny	Informacje dodatkowe
Select Scene	Item List	Tak	Lista opcji: 1) University 2) Office	-	-	Lista scen na urządzeniach mobilnych zawężona jest do scen <i>low poly</i> .
Server IP	Input	Tak	-	-	-	Domyślnie w polu wprowadzony jest adres 127.0.0.1
System Prompt	Textarea	Tak	Pole dopuszcza wprowadzenie do 1000 znaków.	-	-	1) W polu wprowadzona jest domyślna zawartość sugerująca modelowi językowemu, że jest rekrutem IT. 2) Po usunięci wartości domyślnej w polu widoczny jest placeholder „Enter system prompt for AI...” 3) W przypadku gdy użytkownik usunie domyślną wartość do modelu wysyłany jest następujący prompt „You are an AI avatar acting as a

Nazwa pola	Typ pola	Czy obowiązkowe	Właściwości	Błąd użytkownika	Komunikat walidacyjny	Informacje dodatkowe
						professional IT recruiter.”
RAG Document (optional)	Button	Tak	Dopuszczalne formaty to .txt, .pdf	Użytkownik dodaje plik o niepoprawnym formacie.	Only PDF and TXT files are allowed.	-
AI starts conversation (greet you first)	Checkbox	Nie	-	-	-	Domyślnie wyłączony.
AI Model	Dropdown	Tak	Lista opcji: 1) Ollama (local) 2) Gemini (Google API)	-	-	Domyślnie wybraną opcją jest Ollama (local).
Google API Key	Input	Tak	Wartości wpisana w polu jest maskowana	Użytkownik nie uzupełnił pola.	Gemini requires API key!	1) W polu widoczny jest placeholder „Enter your Gemini API key...” 2) Pole jest widoczne, gdy wybrano model Gemini.

4.6.2 Przebieg symulacji

Tabela 6 – Tabela 8 zawierają scenariusze przypadków użycia związanych z symulacją określonej sytuacji stresowej.

Tabela 6 Scenariusz przypadku użycia PU.SYMULACJA.01. Źródło: Opracowanie własne.

ID przypadku użycia	PU.SYMULACJA.01
Nazwa przypadku użycia	Uruchom symulację
ID wymagania	WF.06
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia uruchomienie symulacji.
Wyzwalacze	Aktor kliknął w przycisk „Connect to Server”.
Warunki początkowe	Wszystkie obligatoryjne parametry konfiguracyjne symulacji są uzupełnione (wykonany <i>PU.KONFIG.01 Skonfiguruj ustawienia symulacji</i>).
Warunki końcowe	Uruchomiona symulacja zgodna z parametrami konfiguracyjnymi
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System uruchamia symulację zgodną z parametrami konfiguracyjnymi
Scenariusz alternatywny	Alt 2 Nie wszystkie parametry konfiguracyjne symulacji zostały uzupełnione lub zawierają błędy - System blokuje uruchomienie symulacji oraz wyświetla komunikaty walidacyjne zgodne z matrycą walidacji dla <i>PU.KONFIG.01 Skonfiguruj ustawienia symulacji</i> → END
Informacje dodatkowe	-

Tabela 7 Scenariusz przypadku użycia PU.SYMULACJA.02. Źródło: Opracowanie własne.

ID przypadku użycia	PU.SYMULACJA.02
Nazwa przypadku użycia	Rozmawiaj z awatarem
ID wymagania	WF.07
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia aktorowi wypowiedzenie się w trakcie symulacji.
Wyzwalacze	Aktor wciska przycisk interakcji
Warunki początkowe	Aktor znajduje się w widoku symulacji.
Warunki końcowe	Cyfrowy awatara udzielił odpowiedzi.

Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) Aktor wypowiada komunikat. 3) System rejestruje wypowiedź aktora, rozpoznaje ją, a następnie przekazuje do dalszego przetwarzania. 4) System przekazuje odpowiedź do cyfrowego awatara 5) Cyfrowy awatar wypowiada odpowiedź, a system jednocześnie prezentuje transkrypcję jego wypowiedzi
Scenariusz alternatywny	-
Informacje dodatkowe	Podczas mówienia cyfrowego awatara system prezentuje dodatkowo transkrypcję jego wypowiedzi.

Tabela 8 Scenariusz przypadku użycia PU.SYMULACJA.03. Źródło: Opracowanie własne.

ID przypadku użycia	PU.SYMULACJA.03
Nazwa przypadku użycia	Zakończ symulację
ID wymagania	WF.08
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia aktorowi zakończenie trwającej symulacji.
Wyzwalacze	Aktor nacisnął odpowiedni przycisk (w zależności od urządzenia)
Warunki początkowe	Aktor jest w trakcie symulacji.
Warunki końcowe	System zakończył symulację, a następnie wyświetlił ekran z podsumowaniem.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System wyświetla okno modalne z prośbą o potwierdzenie czynności 3) Aktor potwierdza zakończenie symulacji 4) System kończy trwającą symulację, a następnie prezentuje ekran z podsumowaniem (przejsie do PU.RAPORT.01)
Scenariusz alternatywny	Alt 3 Aktor anuluje czynność - System zamyka okno modalne → END Alt 4 Aktor zakończył symulację bez rozmowy z awatarem - System kończy symulację, a następnie prezentuje główne menu aplikacji → END
Informacje dodatkowe	-

4.6.3 Raport i statystyki

Tabela 11 – Tabela 14 zawierają scenariusze przypadków użycia związanych wyświetlaniem raportów z symulacji.

Tabela 9 Scenariusz przypadku użycia PU.RAPORT.01. Źródło: Opracowanie własne.

ID przypadku użycia	PU.RAPORT.01
Nazwa przypadku użycia	Wyświetl listę przeprowadzonych symulacji
ID wymagania	WF.09
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie listy symulacji.
Wyzwalacze	Aktor nacisnął przycisk „Statistics”.
Warunki początkowe	Aktor znajduje się w menu głównym aplikacji.
Warunki końcowe	Wyświetlona lista symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System prezentuje listę symulacji
Scenariusz alternatywny	Alt 2 Aktor nie przeprowadził żadnej symulacji - System prezentuje komunikat „No saved sessions.” → END
Informacje dodatkowe	Na liście prezentowane są kolejno: 1) data przeprowadzenia symulacji, 2) godzina przeprowadzenia symulacji, 3) czas trwania symulacji, 4) liczba wypowiedzi, 5) ogólny wynik symulacji.

Tabela 10 Scenariusz przypadku użycia PU.RAPORT.02. Źródło: Opracowanie własne.

ID przypadku użycia	PU.RAPORT.02
Nazwa przypadku użycia	Wyświetl szczegóły symulacji
ID wymagania	WF.10
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie szczegółów symulacji.
Wyzwalacze	Aktor wybrał pozycję z listy symulacji LUB Przejsie z PU.SYMULACJA.03
Warunki początkowe	Aktor znajduje się na liście symulacji ORAZ Na liście znajduje się co najmniej jedna pozycja
Warunki końcowe	Wyświetlone szczegóły symulacji

Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System wyświetla szczegóły symulacji
Scenariusz alternatywny	-
Informacje dodatkowe	Na ekran szczegółów składają się: 1) sekcja zawierająca podstawowe informacje o symulacji, w tym datę i godzinę jej przeprowadzenia, 2) sekcja z ocenami: a) stresu głosowego, b) pewności siebie, c) płynności mowy, d) ekspresji głosu, e) napięcia głosowego, f) jakości artykulacji. 3) zakładki z transkrypcją rozmowy, konfiguracją przeprowadzonej symulacji oraz podsumowaniem AI elementów, takich jak: a) merytoryka rozmowy (słabe i mocne strony), b) przekaz wypowiedzi (płynność, stres, ekspresja),

Tabela 11 Scenariusz przypadku użycia PU.RAPORT.03. Źródło: Opracowanie własne.

ID przypadku użycia	PU.RAPORT.03
Nazwa przypadku użycia	Wyświetl podsumowanie wypowiedzi (akustyczne)
ID wymagania	WF.10
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie podsumowania akustycznego symulacji.
Wyzwalacze	Aktor wybrał zakładkę „AI speech review”.
Warunki początkowe	Aktor znajduje się w widoku szczegółów symulacji.
Warunki końcowe	System wyświetlił raport z podsumowaniem akustycznym symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System wyświetla podsumowanie dla wybranej symulacji
Scenariusz alternatywny	-
Informacje dodatkowe	Zakładka jest domyślnie otworzona na wejściu na ekran szczegółów symulacji.

Tabela 12 Scenariusz przypadku użycia PU.RAPORT.04. Źródło: Opracowanie własne.

ID przypadku użycia	PU.RAPORT.04
Nazwa przypadku użycia	Wyświetl podsumowanie wypowiedzi (merytoryczne)
ID wymagania	WF.10
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie podsumowania merytorycznego symulacji.
Wyzwalacze	Aktor wybrał zakładkę „AI conversation review”.
Warunki początkowe	Aktor znajduje się w widoku szczegółów symulacji.
Warunki końcowe	System wyświetlił raport z podsumowaniem merytorycznym symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System wyświetla podsumowanie dla wybranej symulacji
Scenariusz alternatywny	-
Informacje dodatkowe	-

Tabela 13 Scenariusz przypadku użycia PU.RAPORT.05. Źródło: Opracowanie własne.

ID przypadku użycia	PU.RAPORT.05
Nazwa przypadku użycia	Wyświetl transkrypcję
ID wymagania	WF.11
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie transkrypcji z określonej symulacji.
Wyzwalacze	Aktor wybrał zakładkę „Conversation Log”.
Warunki początkowe	Aktor znajduje się w widoku szczegółów symulacji.
Warunki końcowe	System wyświetlił transkrypcję symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System prezentuje transkrypcję
Scenariusz alternatywny	-
Informacje dodatkowe	-

Tabela 14 Scenariusz przypadku użycia PU.RAPORT.06. Źródło: Opracowanie własne.

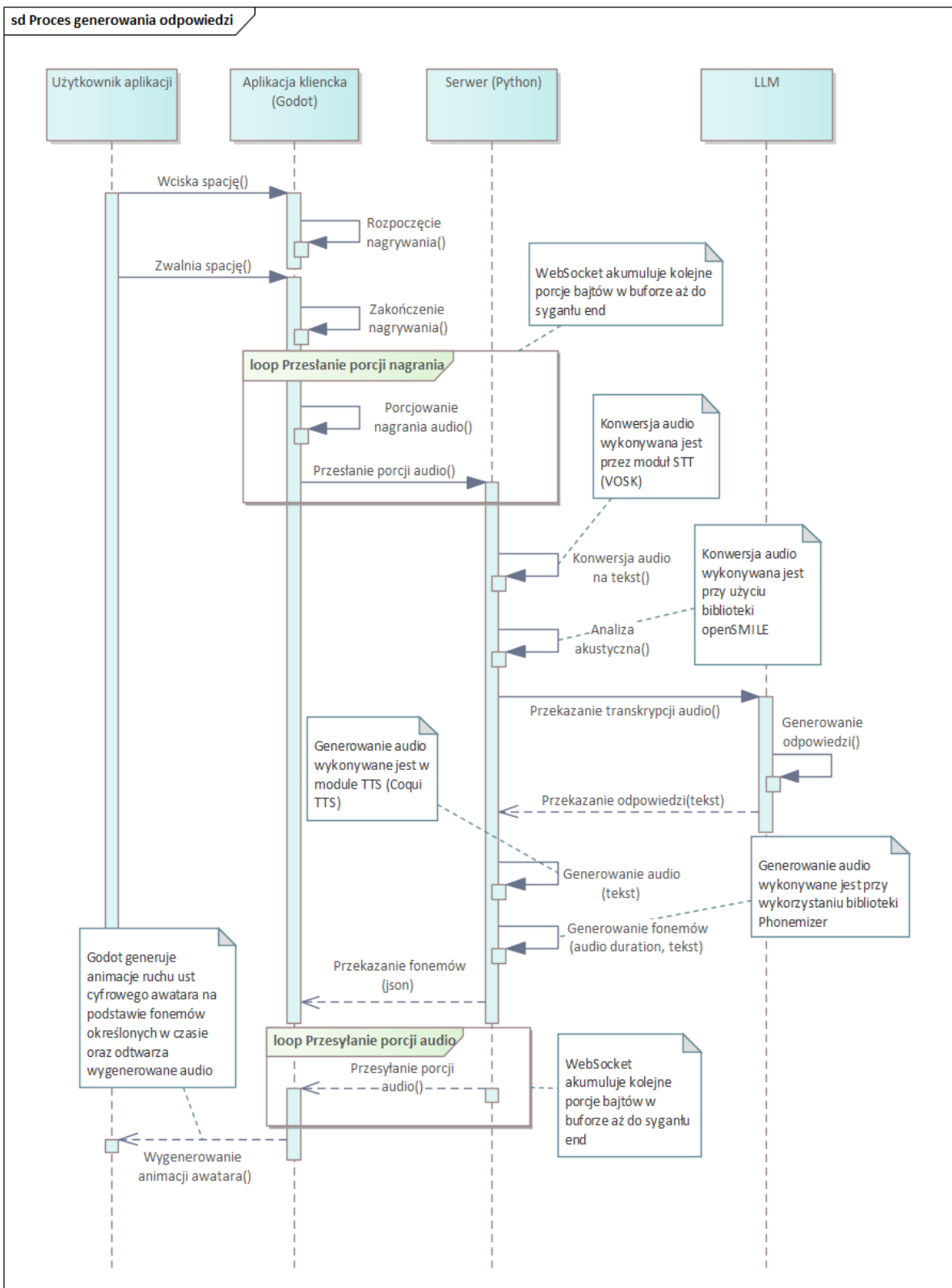
ID przypadku użycia	PU.RAPORT.06
Nazwa przypadku użycia	Wyświetl ustawienia przeprowadzonej symulacji
ID wymagania	WF.12
Aktorzy	Użytkownik aplikacji
Opis	Przypadek użycia umożliwia wyświetlenie ustawień z określonej symulacji.
Wyzwalacze	Aktor wybrał opcję „Configuration”.
Warunki początkowe	Aktor znajduje się w widoku szczegółów symulacji.
Warunki końcowe	System wyświetlił ustawienia symulacji.
Scenariusz główny	1) Aktor uruchamia przypadek użycia 2) System prezentuje ustawienia wybranej symulacji
Scenariusz alternatywny	-
Informacje dodatkowe	-

4.7. Diagramy sekwencji

Jednym z najważniejszych elementów aplikacji jest proces generowania odpowiedzi. W momencie naciśnięcia spacji Godot zaczyna rejestrować wypowiedź użytkownika aż do momentu jej zwolnienia. Następnie aplikacja kliencka przekazuje nagranie audio do serwera, gdzie następuje przede wszystkim jego konwersja na tekst. Dodatkowo każda wypowiedź podlega analizie akustycznej, na podstawie, której na końcu symulacji aplikacja generuje raport z podsumowaniem.

Kolejny krok stanowi przekazanie tekstu do wybranego modelu LLM, określonego w konfiguracji symulacji. Model LLM generuje odpowiedź dla przekazanej treści, bazując na przekazanym prompcie oraz ewentualnym pliku rozszerzającym kontekst. Wygenerowana odpowiedź zwracana jest na serwer, gdzie z wykorzystaniem biblioteki Coqui TTS generowane jest audio.

Następnie określone są fonemy w czasie, z wykorzystaniem tekstu otrzymanego od modelu językowego oraz wygenerowanego w poprzednim kroku audio. Serwer zwraca do aplikacji klienckiej komunikat JSON z określonymi w czasie fonemami oraz wygenerowane audio. Na ich podstawie Godot generuje animację ruchu ust cyfrowego awatara, zsynchronizowaną z odtwarzanym dźwiękiem. Rysunek 30 przedstawia opisaną powyżej sekwencję.



Rysunek 30 Diagram sekwencji - proces generowania odpowiedzi. Źródło: Opracowanie własne.

4.8. Integracje

W niniejszym podrozdziale opisano zewnętrzne integracje wykorzystane w prototypie aplikacji Stressless VR. Tabela 15 przedstawia szczegółowy opis integracji z Gemini API firmy Google.

Tabela 15 Integracja z Google Gemini. Źródło: Opracowanie własne.

Udostępniający	Google LLC
Autoryzacja	Klucz API w nagłówku x-goog-api-key generowany w Google AI Studio
Przeznaczenie	Generowanie odpowiedzi tekstowych przez model językowy Gemini
Typ metody	POST
Adres	https://generativelanguage.googleapis.com/v1beta/models/{model}:generateContent
Dane wejściowe	system_instruction – prompt systemowy contents – historia konwersacji z polami role (user/model) i parts generationConfig – opcjonalne parametry generowania
Format żądania	JSON
Przykładowy request	<pre>{ "system_instruction": { "parts": [{ "text": "You are a job interview examiner. Respond in English." }] }, "contents": [{ "role": "model", "parts": [{ "text": "Can you tell me about your experience?" }] }, { "role": "user", "parts": [{ "text": "Of course! I have been working in software development for five years." }] }, { "role": "model", "parts": [{ "text": "What technologies do you use most often?" }] }] }</pre>

Przykładowy response	<pre>{ "candidates": [{ "content": { "role": "model", "parts": [{ "text": "Thank you for coming today. Could you please start by telling me a little about yourself and your professional background?" }] }, "finishReason": "STOP", "index": 0 }], "usageMetadata": { "promptTokenCount": 87, "candidatesTokenCount": 34, "totalTokenCount": 121 }, "modelVersion": "gemini-3.1-flash-lite-preview" }</pre>
Obsługa błędów	Wszelkie błędy zwrócone przez API są przekazywane do aplikacji klienckiej.

5. Narzędzia i technologie zastosowane przy realizacji pracy

W niniejszym rozdziale omówiono narzędzia i technologie, które zostały wykorzystane podczas realizacji projektu aplikacji StresslessVR. Opisy zostały uzupełnione uzasadnieniem wyboru danego narzędzia bądź technologii.

5.1. Wykorzystane narzędzia

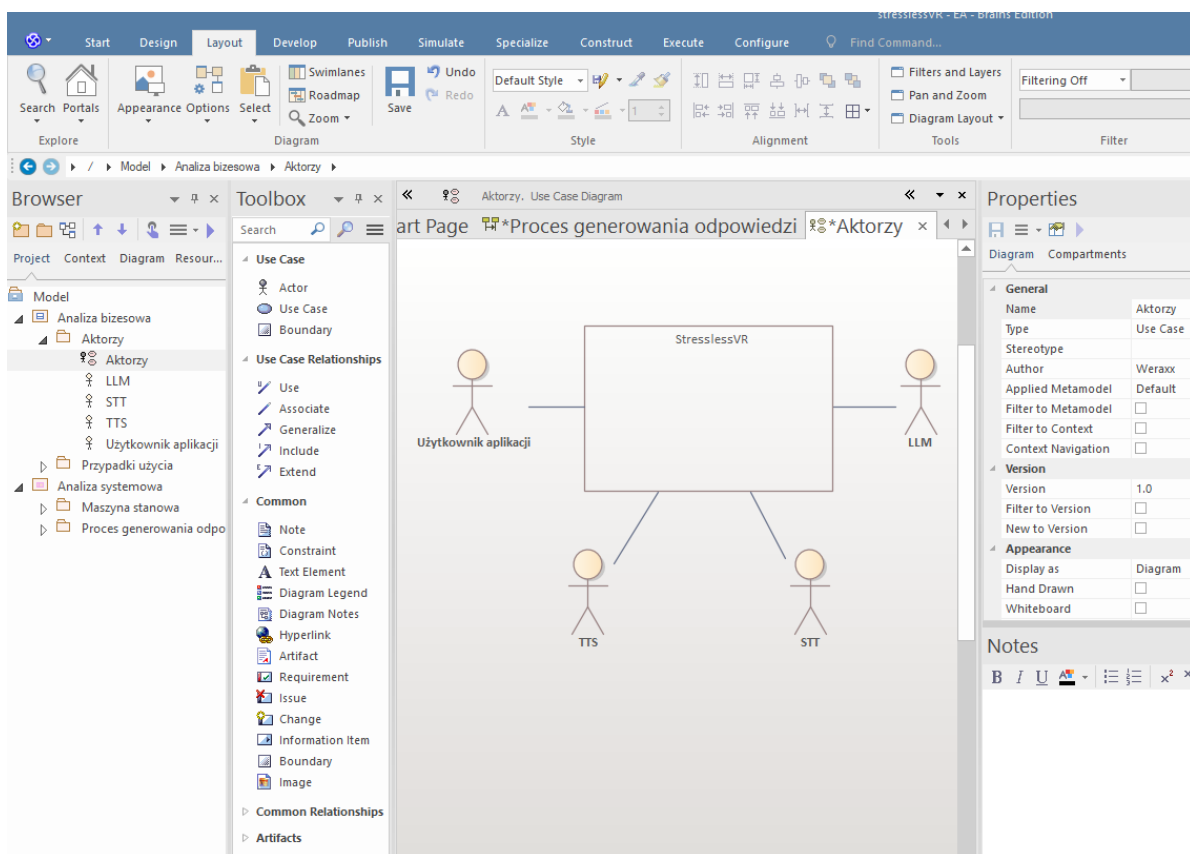
W niniejszym podrozdziale opisano narzędzia wykorzystane do realizacji kolejnych etapów projektu Stressless VR – od analizy po implementację.

5.1.1 Enterprise Architect

Do zamodelowania systemu Stressless VR wykorzystany został Enterprise Architect, będący jednym z najpopularniejszych narzędzi do modelowania systemów na rynku. Narzędzie to umożliwia modelowanie na różnych poziomach abstrakcji, w tym na poziomie biznesowym, architektonicznym oraz systemowym.

Enterprise Architect wspiera szeroki zakres notacji, w tym UML, BPMN, SysML oraz ArchiMate. Dzięki temu narzędzie może być wykorzystywane zarówno do modelowania wymagań i procesów biznesowych, jak również struktury systemu oraz architektury rozwiązania [31].

Rysunek 31 przedstawia interfejs Enterprise Architect wraz z przykładowym diagramem UML.



Rysunek 31 Przykładowy diagram UML w Enterprise Architect. Źródło: Opracowanie własne.

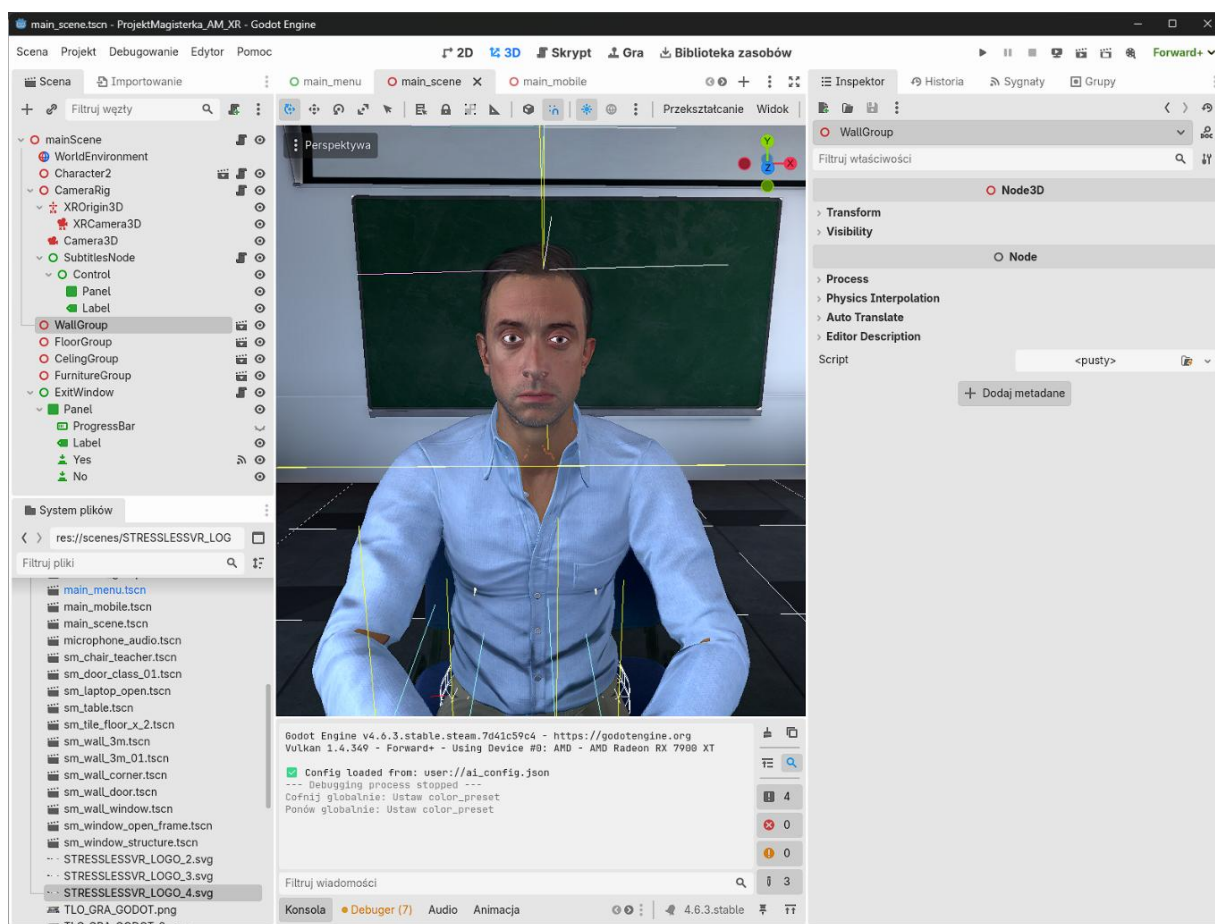
5.1.2 Godot

Godot jest bezpłatnym silnikiem do tworzenia gier, udostępnianym na licencji *open-source*. W przeciwieństwie do konkurencyjnych silników, takich jak Unity czy Unreal Engine, nie wymaga uiszczania opłat licencyjnych ani tantiem od aplikacji komercyjnych wytworzonych przy jego użyciu, niezależnie od liczby sprzedanych kopii. Godot jest dostępny dla systemów Windows, Linux oraz macOS, a także natywnie wspiera platformę Android, co umożliwia tworzenie i testowanie gier bezpośrednio na urządzeniach mobilnych. Projekty stworzone przy jego użyciu mogą być eksportowane na komputery osobiste, urządzenia mobilne oraz do przeglądarek internetowych [32].

Sam silnik Godot został napisany w języku C++, natomiast podstawowym językiem wykorzystywanym do programowania i skryptowania w tym środowisku jest autorski GDScript, którego składnia przypomina składnię języka Python. Godot wspiera również język C#. Od wersji 4.4 projekty wykorzystujące C# muszą być zgodne z platformą .NET 8 lub nowszą ze względu na przeniesienie pakietów GodotSharp na .NET 8.

Silnik Godot, bazując na standardzie OpenXR, zapewnia wbudowane wsparcie dla technologii wirtualnej rzeczywistości (ang. *virtual reality*, *VR*), rozszerzonej rzeczywistości (ang. *augmented reality*, *AR*), jak i mieszanej rzeczywistości (ang. *mixed reality*, *MR*), co pozwala na tworzenie aplikacji 3D na popularne gogle, takie jak Meta Quest czy Valve Index.

Rysunek 32 przedstawia przykładową scenę 3D w edytorze Godot.



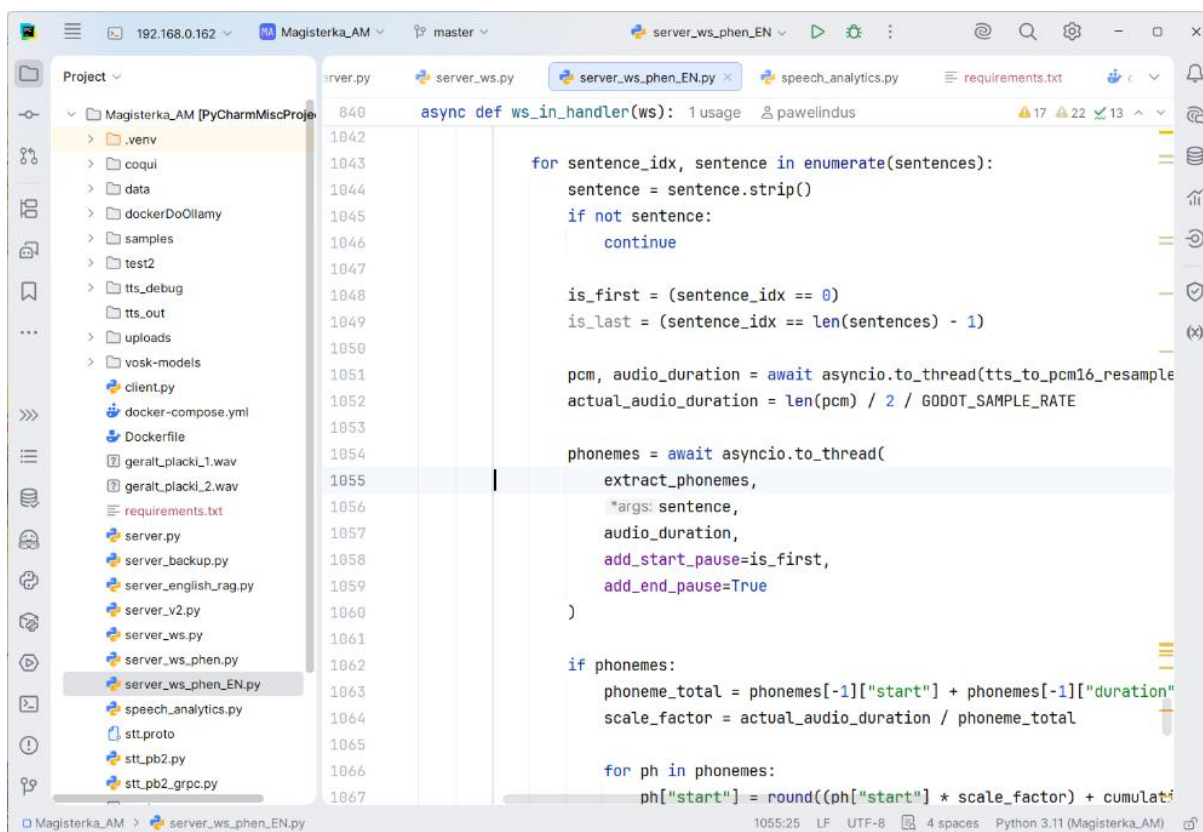
Rysunek 32 Widok przykładowej sceny symulacji w edytorze Godot. Źródło: Opracowanie własne.

5.1.3 PyCharm

PyCharm to zintegrowane środowisko programistyczne (ang. *integrated development environment*, IDE) dla języka Python, stworzone przez firmę JetBrains. Narzędzie umożliwia pisanie, testowanie oraz debugowanie kodu, a także udostępnia funkcjonalności takie jak:

- inspekcja kodu,
- zarządzanie pakietami,
- obsługa refaktoryzacji,
- analiza i kontrola typów,
- integracja z systemami kontroli wersji,
- podpowiedzi składniowe oraz uzupełnianie kodu [33] [34].

Rysunek 33 przedstawia główne okno PyCharm 2025.3.3.



Rysunek 33 Główne okno PyCharm 2025.3.3. Źródło: Opracowanie własne.

5.1.4 Git

Git to w pełni darmowy, otwartoźródłowy (ang. *open-source*), rozproszony system kontroli wersji, który sprawdzi się zarówno dla małych jak i dużych projektów [35]. Narzędzie to umożliwia śledzenie i zarządzanie zmianami w plikach aplikacji, w szczególności w kodzie źródłowym, wspierając pracę zespołową wielu programistów nad jednym projektem poprzez:

- tworzenie repozytoriów,
- pracę na gałęziach,
- rejestrowanie kolejnych wersji plików w postaci *commitów*,
- scalanie zmian wprowadzanych w różnych częściach projektu.

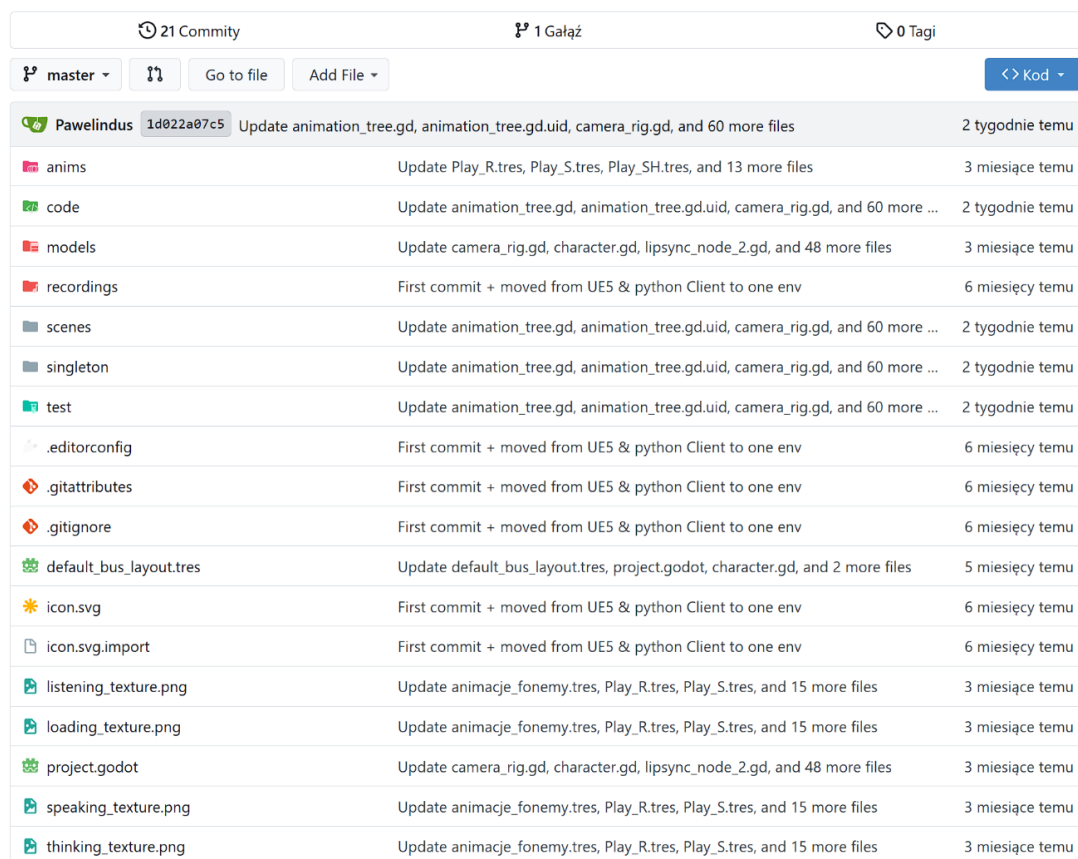
Najważniejsze zalety Git to m.in.:

- wydajność i szybkość działania;
- możliwość pracy bez dostępu do sieci, z wyjątkiem operacji pobierania zmian ze zdalnego repozytorium oraz wysyłania zmian do zdalnego repozytorium;
- niezawodność i integralność danych dzięki zastosowaniu unikalnej sumy kontrolnej dla każdego *commita* [36].

5.1.5 Gitea

Gitea to otwartoźródłowy (ang. *open-source*), kompleksowy serwis wspierający wytwarzanie oprogramowania, który umożliwia hosting repozytoriów Git na własnej infrastrukturze. Narzędzie wspiera m.in.:

- tworzenie repozytoriów i zarządzanie nimi (Rysunek 34),
- obsługę gałęzi,
- zarządzanie współpracownikami,
- recenzowanie i scalanie zgłoszonych zmian,
- przeglądanie historii commitów oraz plików kodu,
- funkcjonalności CI/CD.



21 Commity		1 Gałąź		0 Tagi	
master	Go to file	Add File		<> Kod	
Pawelindus	1d022a07c5	Update animation_tree.gd, animation_tree.gd.uid, camera_rig.gd, and 60 more files			2 tygodnie temu
anims		Update Play_R.tres, Play_S.tres, Play_SH.tres, and 13 more files			3 miesiące temu
code		Update animation_tree.gd, animation_tree.gd.uid, camera_rig.gd, and 60 more ...			2 tygodnie temu
models		Update camera_rig.gd, character.gd, lipsync_node_2.gd, and 48 more files			3 miesiące temu
recordings		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
scenes		Update animation_tree.gd, animation_tree.gd.uid, camera_rig.gd, and 60 more ...			2 tygodnie temu
singleton		Update animation_tree.gd, animation_tree.gd.uid, camera_rig.gd, and 60 more ...			2 tygodnie temu
test		Update animation_tree.gd, animation_tree.gd.uid, camera_rig.gd, and 60 more ...			2 tygodnie temu
.editorconfig		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
.gitattributes		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
.gitignore		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
default_bus_layout.tres		Update default_bus_layout.tres, project.godot, character.gd, and 2 more files			5 miesięcy temu
icon.svg		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
icon.svg.import		First commit + moved from UE5 & python Client to one env			6 miesięcy temu
listening_texture.png		Update animacje_fonemy.tres, Play_R.tres, Play_S.tres, and 15 more files			3 miesiące temu
loading_texture.png		Update animacje_fonemy.tres, Play_R.tres, Play_S.tres, and 15 more files			3 miesiące temu
project.godot		Update camera_rig.gd, character.gd, lipsync_node_2.gd, and 48 more files			3 miesiące temu
speaking_texture.png		Update animacje_fonemy.tres, Play_R.tres, Play_S.tres, and 15 more files			3 miesiące temu
thinking_texture.png		Update animacje_fonemy.tres, Play_R.tres, Play_S.tres, and 15 more files			3 miesiące temu

Rysunek 34 Repozytorium projektu Stressless VR w systemie Gitea. Źródło: Opracowanie własne.

Gitea obsługuje również wiele popularnych mechanizmów Git, takich jak *tagi*, *cherry-picking* oraz *hooki* [37]. Narzędzie to stanowi alternatywę dla platform takich jak GitHub, Bitbucket czy GitLab.

5.1.6 Docker

Docker to otwartoźródłowa (ang. *open-source*) platforma umożliwiająca deweloperom i administratorom systemów konteneryzację aplikacji. Konteneryzacja polega na uruchamianiu aplikacji

w odizolowanych środowiskach, które zawierają wszystkie niezbędne do jej działania elementy, takie jak:

- pliki aplikacji,
- biblioteki i zależności,
- narzędzia systemowe,
- środowisko uruchomieniowe.

Kontener zawierający aplikację można wdrożyć na docelowej platformie, np. na serwerze lokalnym lub w środowisku chmurowym, a następnie uruchomić w taki sam sposób niezależnie od infrastruktury. Na jednym serwerze może działać wiele kontenerów, z których każdy zawiera własną aplikację lub usługę. Kontenery są od siebie odizolowane, co zwiększa bezpieczeństwo, ogranicza ryzyko konfliktów między zależnościami oraz ułatwia utrzymanie spójnego środowiska uruchomieniowego [38].

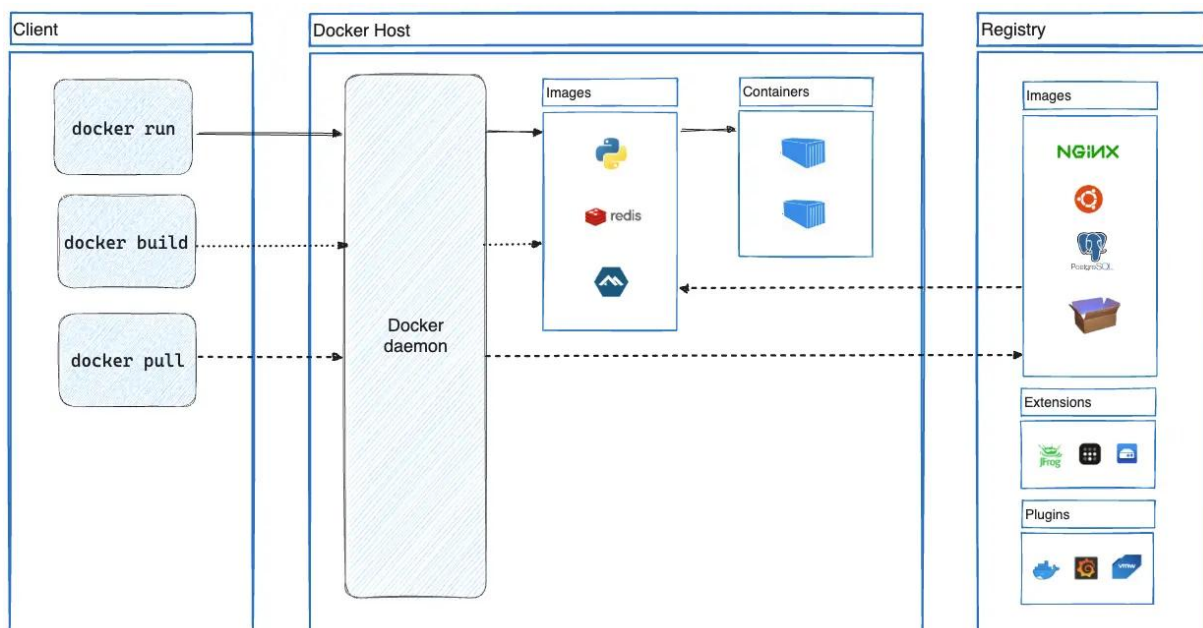
Najważniejsze korzyści korzystania z Dockera to m.in.:

- spójność,
- szybkie wdrażanie,
- integracja z CI/CD,
- izolacja i bezpieczeństwo,
- efektywne zarządzanie zasobami.

Rysunek 35 przedstawia architekturę Dockera, która opiera się na modelu klient-serwer. Klient Dockera komunikuje się z demonem Dockera (ang. *Docker daemon*), odpowiedzialnym za wykonywanie najważniejszych operacji, takich jak:

- budowanie,
- uruchamianie,
- dystrybuowanie kontenerów.

Komponenty te mogą działać w ramach tego samego systemu, jednak klient może również łączyć się ze zdalnym demonem za pomocą Docker API. Innym narzędziem pełniącym rolę klienta Dockera jest Docker Compose, który umożliwia pracę z aplikacjami składającymi się z wielu kontenerów.



Rysunek 35 Architektura Dockera. Źródło: [39]

5.1.7 Komodo

Komodo to aplikacja webowa służąca do zarządzania serwerami, *buildami*, wdrożeniami oraz zautomatyzowanymi procedurami. Umożliwia m.in.:

- podłączanie serwerów,
- wdrażanie kontenerów Docker i zarządzanie nimi,
- monitorowanie użycia procesora, pamięci i dysku wraz z obsługą alertów [40].

5.2. Wykorzystane technologie

W niniejszym podrozdziale opisano technologie wykorzystane do realizacji projektu Stressless VR.

5.2.1 Python

Zgodnie z literaturą Python to jeden z popularniejszych wysokopoziomowych, zorientowanych obiektowo języków programowania. Jego główne cechy to m.in:

- czytelna i prosta składnia,
- szeroka gama bibliotek i modułów,
- wszechstronność i elastyczność zapewniona wsparciem paradygmatów programowania obiektowego, strukturalnego, jak i funkcyjnego.

Python jest wykorzystywany m.in. do:

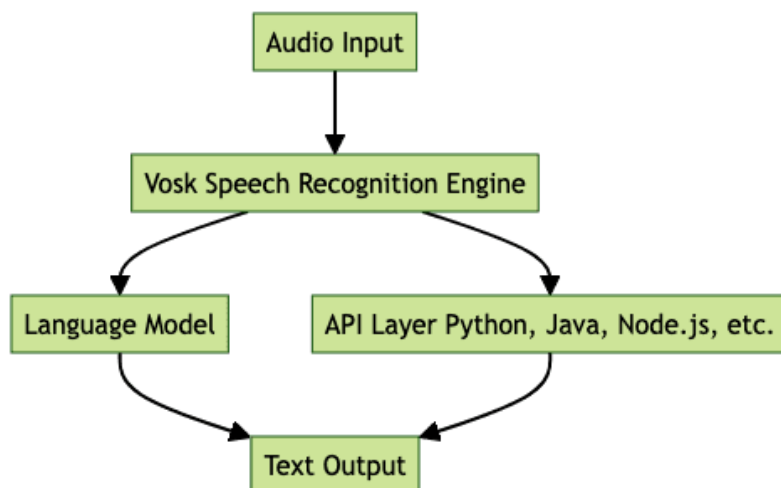
- analizy danych,
- automatyzacji zadań,
- uczenia maszynowego,
- przetwarzania języka naturalnego.

5.2.2 Biblioteka Vosk

Vosk to otwartoźródłowa biblioteka do rozpoznawania mowy, przekształcająca sygnał audio na tekst. Obsługuje ponad 20 języków i dialektów, w tym język angielski, niemiecki, francuski, hiszpański oraz polski. Biblioteka, wspierając pracę w trybie offline, nie wymaga przesyłania nagrań do zewnętrznych usług chmurowych. Dzięki strumieniowemu interfejsowi API biblioteka może być wykorzystywana do rozpoznawania mowy w czasie rzeczywistym. Vosk wspiera również konfigurowalne słownictwo oraz identyfikację mówcy.

Vosk może być uruchamiany zarówno na urządzeniach o ograniczonych zasobach, takich jak Raspberry Pi czy smartfony z systemem Android, jak i na dużych klastrach [41] [42].

Rysunek 36 przedstawia uproszczony schemat przetwarzania mowy na tekst z wykorzystaniem biblioteki Vosk.



Rysunek 36 Schemat działania biblioteki Vosk. Źródło: [43]

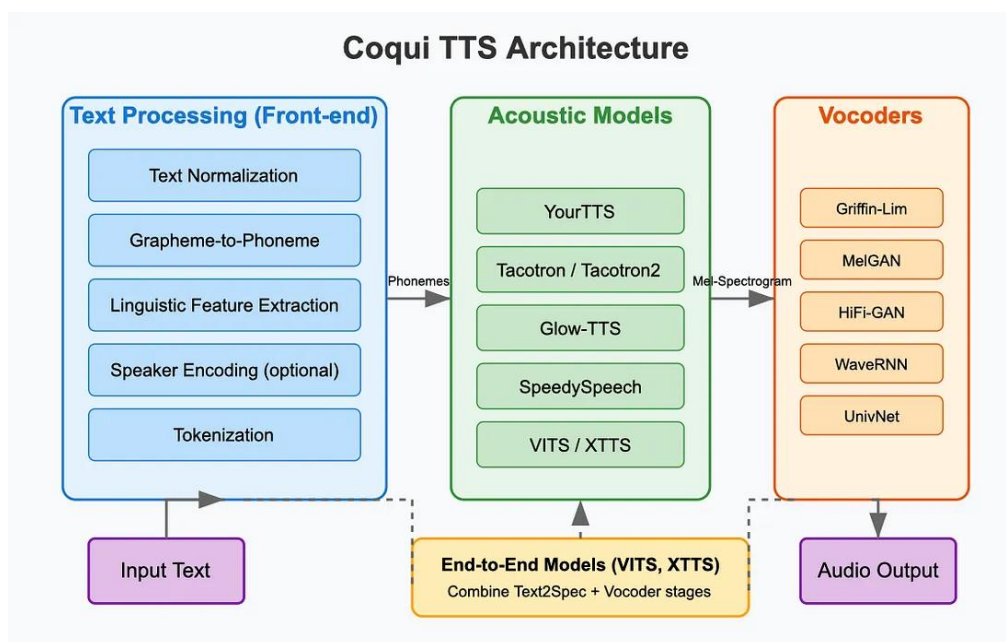
5.2.3 Biblioteka Coqui TTS

Coqui TTS to otwartoźródłowa (ang. *open-source*) biblioteka wykorzystująca sztuczną inteligencję do syntezy mowy, czyli przekształcania tekstu na sygnał dźwiękowy. Coqui TTS udostępnia wiele gotowych, wstępnie wytrenowanych modeli, które można pobrać i wykorzystać bez samodzielnego trenowania. Poszczególne modele różnią się m.in. obsługiwanymi językami, głosami, jakością i szybkością syntezy oraz możliwością klonowania głosu. Biblioteka pozwala również trenować własne modele i dostosowywać istniejące do nowych danych.

Biblioteka umożliwia m.in.:

- klonowanie głosu z próbek audio,
- generowanie głosu w czasie rzeczywistym,
- eksport wygenerowanej mowy w postaci plików WAV,
- tworzenie i projektowanie niestandardowych głosów [44].

Rysunek 37 przedstawia modułową architekturę biblioteki Coqui TTS.



Rysunek 37 Architektura Coqui TTS. Źródło: [45]

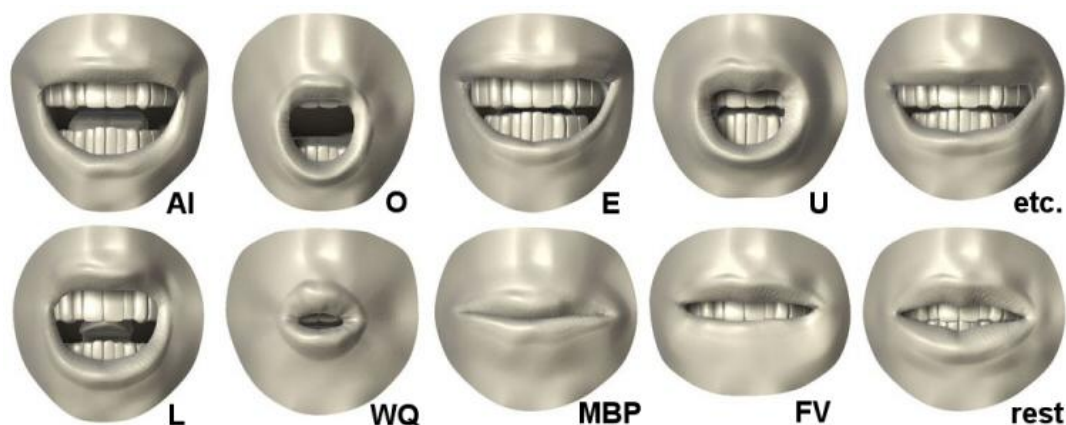
Na diagramie widoczne są trzy główne komponenty:

- Text Processing – odpowiada za przetwarzanie danych wejściowych, w tym normalizację tekstu, jego tokenizację oraz konwersję na fonemy lub inne reprezentacje wykorzystywane przez model;
- Acoustic Models – przekształcają przetworzony tekst w pośrednią reprezentację akustyczną mowy;
- Vocoders – przekształcają reprezentację akustyczną wygenerowaną przez model akustyczny w końcowy sygnał dźwiękowy, który może zostać odtworzony lub zapisany w pliku.

5.2.4 Biblioteka Phonemizer

Phonemizer to otwartoźródłowa biblioteka umożliwiająca fonemizację pojedynczych słów, a także dłuższych tekstów w wielu językach [46]. Proces fonemizacji polega na przekształceniu tekstu zapisanego za pomocą grafemów w jego reprezentację fonetyczną, określającą sposób wymowy [47].

Uzyskana reprezentacja może być następnie wykorzystana między innymi do sterowania animacją aparatu mowy cyfrowej postaci. Poszczególnym fonemom lub ich grupom przypisuje się odpowiednie wizemy, czyli charakterystyczne ułożenia ust podczas wymawiania określonych dźwięków. Przykładowe wizemy odpowiadające fonemom i grupom fonemów występującym w języku angielskim przedstawiono na Rysunek 38.



Rysunek 38 Przykładowe wizemy. Źródło: [48]

5.2.5 Biblioteka openSMILE

openSMILE (open-source Speech and Music Interpretation by Large-space Extraction) to kompletny i otwartoźródłowy zestaw narzędzi do analizy, przetwarzania i klasyfikacji dźwięku [49]. Narzędzie jest ukierunkowane w szczególności na analizę mowy i muzyki.

Biblioteka umożliwia wyznaczanie cech akustycznych sygnału mowy, takich jak:

- częstotliwość podstawowa, związana z wysokością głosu;
- głośność, intensywność i energia, określające siłę wypowiedzi;
- współczynniki opisujące barwę i charakterystyczne właściwości głosu;
- parametry opisujące niewielkie zmiany wysokości oraz głośności głosu;
- tempo mówienia oraz długość wypowiedzi i występujących w niej przerw;
- cechy przedstawiające rozkład energii sygnału pomiędzy poszczególnymi częstotliwościami oraz zależności między jego składowymi [24].

Wyodrębnione cechy mogą być następnie wykorzystywane jako dane wejściowe dla modeli uczenia maszynowego, między innymi w systemach rozpoznawania emocji, identyfikacji mówcy oraz oceny poziomu stresu na podstawie głosu.

5.2.6 Modele LLM

Duże modele językowe (ang. *Large Language Models, LLM*) będące jednym z rodzajów modeli sztucznej inteligencji przeznaczone są do przetwarzania i generowania języka naturalnego poziomem zbliżonym do ludzkiego.

Wśród najważniejszych cech dużych modeli językowych można wskazać:

- zdolność uczenia się zależności i wzorców językowych na podstawie dużych zbiorów danych, bez konieczności definiowania sztywnych reguł gramatycznych;
- wykorzystanie mechanizmu samouwagi (ang. *self-attention*), który umożliwia modelom analizowanie zależności między poszczególnymi elementami wypowiedzi oraz określanie ich znaczenia w danym kontekście poprzez przypisywanie tym zależnościom odpowiednich wag;
- generatywny charakter, polegający na tworzeniu tekstu poprzez przewidywanie kolejnych tokenów, czyli jednostek tekstu mogących odpowiadać między innymi fragmentom słów, całym słowom, znakom interpunkcyjnym lub innym znakom [50].

5.2.6.1 Gemini

Gemini to chatbot i wirtualny asystent wykorzystujący sztuczną inteligencję, opracowany przez Google. Działanie asystenta opiera się na rodzinie modeli Gemini, zdolnych do przetwarzania różnych rodzajów danych, takich jak tekst, obrazy, dźwięk, materiały wideo oraz kod źródłowy [51]. Poszczególne warianty modeli są dostosowane do różnych zastosowań i mogą różnić się między innymi zakresem możliwości, szybkością działania oraz zapotrzebowaniem na zasoby obliczeniowe.

Programistyczny dostęp do modeli jest możliwy między innymi za pomocą interfejsu Gemini API [52].

5.2.6.2 Ollama

Ollama to otwartoźródłowe (ang. *open-source*) narzędzie służące do pobierania, uruchamiania i zarządzania modelami sztucznej inteligencji, w szczególności dużymi modelami językowymi. Nie stanowi ono samodzielnego modelu, lecz środowisko uruchomieniowe umożliwiające korzystanie z modeli należących do różnych rodzin, takich jak:

- Llama,
- Qwen3,
- Gemma 4,
- DeepSeek-R1,
- gpt-oss (modele opracowane przez OpenAI) [53].

Ollama umożliwia uruchamianie modeli lokalnie na komputerach z systemami Windows, Linux oraz macOS. W przypadku korzystania z modeli lokalnych zapytania mogą być przetwarzane bezpośrednio na urządzeniu użytkownika, bez konieczności przesyłania ich treści do zewnętrznego dostawcy usługi. Szybkość generowania odpowiedzi zależy między innymi od wielkości modelu oraz dostępnych zasobów sprzętowych, takich jak pamięć operacyjna, procesor oraz karta graficzna [54].

6. Opis implementacji

W niniejszym rozdziale omówiono architekturę oraz implementację prototypu aplikacji, uwzględniając kluczowe decyzje projektowe podjęte w trakcie realizacji. Przedstawiono budowę aplikacji serwerowej odpowiedzialnej za przetwarzanie mowy, analizę akustyczną oraz integrację z modelami językowymi, a także aplikacji klienckiej zrealizowanej w silniku Godot, obejmującej scenę trójwymiarową, animacje awatara oraz komunikację z serwerem.

6.1. Przegląd architektury

Prototyp aplikacji zbudowany jest w architekturze klient-serwer, gdzie każda z warstw realizowana jest w odrębnym środowisku technologicznym dobranym do jej charakteru obliczeniowego. Warstwę serwerową stanowi aplikacja napisana w języku Python, natomiast warstwę kliencką – aplikacja zrealizowana w silniku Godot 4.

Wybór języka Python po stronie serwera podyktowany był jego dominującą pozycją w ekosystemie uczenia maszynowego (ang. *machine learning*). Biblioteki do rozpoznawania i syntezy mowy, modele osadzania (ang. *embedding*) tekstu czy integracje z dużymi modelami językowymi są w tym środowisku dojrzałe, dobrze udokumentowane i wspierają akcelerację sprzętową przez CUDA. Istotnym argumentem był również podział obliczeniowy – ciężkie operacje takie jak rozpoznawanie mowy, synteza audio, przeszukiwanie bazy wektorowej RAG czy odpytywanie modelu językowego realizowane są po stronie serwera, zwalniając klienta z konieczności wykonywania kosztownych obliczeń równoległe z renderowaniem sceny trójwymiarowej. Architektura ta umożliwia ponadto elastyczne wdrożenie – serwer może działać na dedykowanej maszynie lokalnej, serwerze domowym lub wynajętej maszynie wirtualnej w chmurze, podczas gdy klient uruchamiany jest na znacznie słabszym sprzęcie, w tym na urządzeniach mobilnych.

Jako silnik aplikacji klienckiej wybrano Godot 4 – otwartoźródłowy (ang. *open-source*) silnik do tworzenia gier i aplikacji interaktywnych. W trakcie projektowania rozważano również silnik Unreal Engine, który oferuje narzędzie MetaHuman umożliwiające łatwe uzyskanie realistycznego modelu postaci ludzkiej. Okazało się jednak, że generowanie animacji synchronizacji ust w czasie rzeczywistym na podstawie strumienia audio działa w Unreal Engine wyłącznie w trybie podglądu edytora i nie jest dostępne w skompilowanej aplikacji. Dodatkową barierą była konieczność stosowania języka C++ przy zaawansowanych integracjach takich jak obsługa WebSocket czy przechwytywanie strumienia audio z mikrofonu, co wymaga rozległej znajomości wewnętrznej architektury silnika. Silnik Unity został wykluczony ze względu na zmiany w polityce licencyjnej wprowadzone przez producenta. Silnik Godot był natomiast znany autorom z wcześniejszych projektów, oferuje natywną obsługę protokołu WebSocket oraz standardu OpenXR dla urządzeń VR, a język skryptowy GDScript okazał się wystarczający do realizacji wszystkich wymaganych integracji.

Komunikacja między warstwami opiera się na protokole WebSocket, który domyślnie wspiera dwukierunkową wymianę danych (ang. *full-duplex*) przez pojedyncze połączenie. Dzięki temu serwer może jednocześnie przysyłać syntetyzowane audio do klienta i odbierać nową wypowiedź użytkownika bez wzajemnego blokowania – całość komunikacji odbywa się przez jedno gniazdo na porcie 8765.

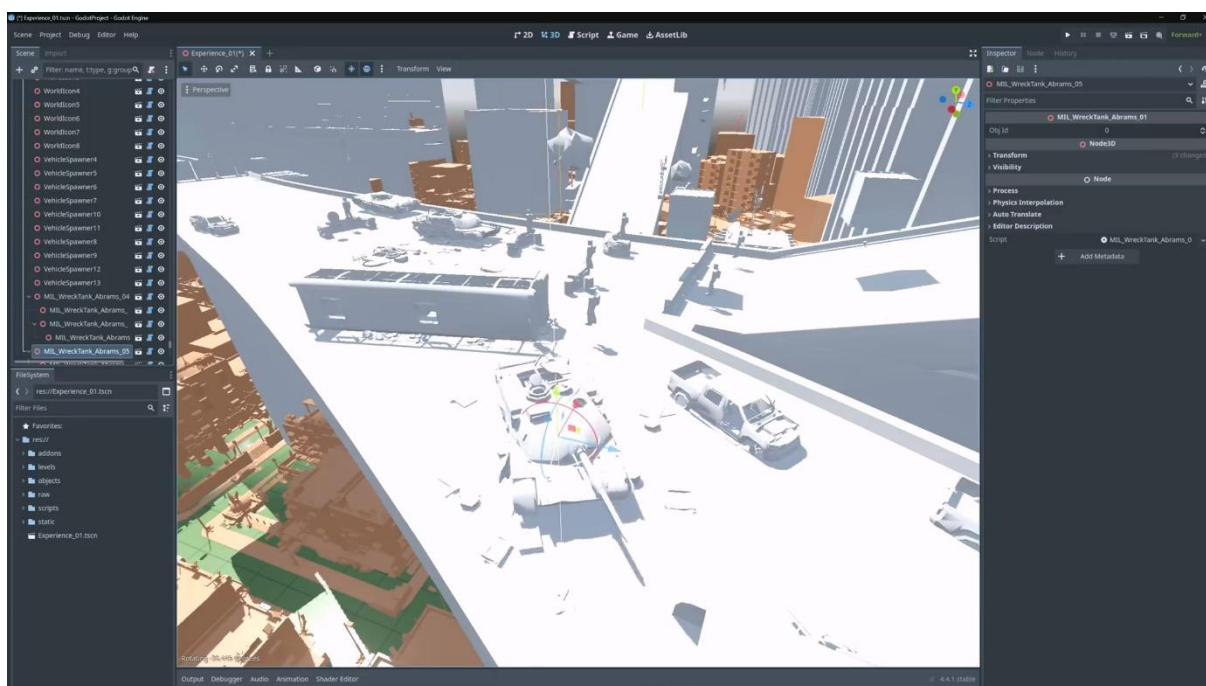
Po stronie klienta komponenty aplikacji komunikują się ze sobą za pośrednictwem systemu sygnałów (ang. *signals*) wbudowanego w silnik Godot. Mechanizm ten sprawdza się szczególnie dobrze w kontekście dynamicznego łączenia singletonów z węzłami sceny – singleton `SpeechClient` emituje sygnały niezależnie od tego, która scena jest aktualnie załadowana, a węzły obecne na scenie subskrybują je w momencie swojego utworzenia. Dzięki temu dodanie nowej sceny nie wymaga modyfikacji kodu singletonów.

Aplikacja kliencka zawiera dwie wersje sceny głównej przedstawiającej pomieszczenie biurowe z cyfrowym awatarem. Wersja pełna przeznaczona jest na komputery osobiste oraz gogle VR

i wykorzystuje szczegółowy model postaci z oprogramowania Character Creator w wersji 5, którego rozmiar po kompresji wynosi około 3 GB. Wersja uproszczona w stylu *low poly*, przeznaczona na urządzenia mobilne, korzysta z lżejszych zasobów graficznych ze sklepu Synty Store, zapewniając płynne działanie na urządzeniach z ograniczoną pamięcią operacyjną. Na małym ekranie telefonu realizm graficzny ma drugorzędne znaczenie – istotna jest możliwość przeprowadzenia symulacji w sposób płynny. Pełna immersja wizualna jest natomiast odczuwalna na dużych ekranach oraz w goglach VR, gdzie jakość modelu i animacji ma bezpośredni wpływ na realizm symulowanej sytuacji.

Architektura prototypu zakłada rozszerzalność przez społeczność użytkowników (ang. *community*). Ponieważ Godot jest otwartoźródłowy i dostępny bezpłatnie, każdy może pobrać projekt i dodać własną scenę zachowując wymagane węzły takie jak *CameraRig* czy *SubtitlesNode*. System sygnałów i singletonów zapewnia automatyczne spinanie nowej sceny z logiką aplikacji bez potrzeby modyfikacji istniejącego kodu – nowy scenariusz symulacji staje się dostępny natychmiast po dodaniu sceny do projektu.

Podobne podejście – wykorzystanie silnika Godot jako platformy do tworzenia treści przez społeczność – zastosowano w trybie Portal gry *Battlefield 6*, gdzie EA udostępniło graczom edytor poziomów *Spatial Editor* oparty na silniku Godot, umożliwiając modyfikowanie istniejących map i tworzenie własnych trybów rozgrywki [55], jak przedstawiono na Rysunek 39.



Rysunek 39 Edytor poziomów *Spatial Editor* trybu Portal w grze *Battlefield 6*, oparty na silniku Godot. Źródło: [55]

6.2. Aplikacja serwerowa (Python)

Aplikacja serwerowa stanowi rdzeń systemu odpowiedzialny za przetwarzanie mowy, generowanie odpowiedzi oraz analizę akustyczną. Została zrealizowana w języku Python z wykorzystaniem architektury asynchronicznej opartej na bibliotece *asyncio*. Poniższe podrozdziały opisują kolejne moduły składające się na tę warstwę systemu.

6.2.1 Architektura i komunikacja WebSocket

Komunikacja między aplikacją kliencką a serwerową opiera się na protokole *WebSocket*, który umożliwia dwukierunkową wymianę danych w czasie rzeczywistym. Zdecydowano się na to

rozwiązanie ze względu na konieczność ciągłego przesyłania strumienia audio oraz natychmiastowego odsyłania odpowiedzi – model żądanie-odpowiedź (ang. *request-response*) typowy dla protokołu HTTP byłby w tym kontekście niewystarczający. Protokół WebSocket jest z natury dwukierunkowy (ang. *full-duplex*), co oznacza, że obie strony mogą jednocześnie wysyłać i odbierać dane przez jedno połączenie – bez konieczności rozdzielania kanałów wejściowego i wyjściowego.

Serwer nasłuchuje na pojedynczym gnieździe (ang. *socket*) na porcie 8765, obsługując jednocześnie odbieranie audio i poleceń od klienta oraz odsyłanie odpowiedzi tekstowych, fonemów, audio oraz raportów. Połączenie obsługiwane jest przez pojedynczą funkcję asynchroniczną `ws_handler`, jak przedstawiono w Listing 1.

Listing 1 Uruchomienie serwera WebSocket nasłuchującego na pojedynczym porcie. Źródło: Opracowanie własne.

```
async with websockets.serve(ws_handler, "0.0.0.0", WS_PORT, **ws_config):
    await asyncio.Future()
```

Serwer dopuszcza wyłącznie jedno aktywne połączenie jednocześnie – próba nawiązania kolejnego połączenia podczas aktywnej sesji skutkuje jego natychmiastowym odrzuceniem. Dane audio przesyłane są jako wiadomości binarne (ang. *bytes*), natomiast polecenia sterujące oraz odpowiedzi serwera jako wiadomości tekstowe w formacie JSON. Klient przysyła kolejne porcje (ang. *chunk*) audio o rozmiarze 4096 bajtów, a zakończenie wypowiedzi sygnalizuje specjalnym znacznikiem tekstowym `__END__`, jak przedstawiono w Listing 2. Po jego odebraniu bufor audio trafia do kolejki przetwarzania.

Listing 2 Obsługa wiadomości przychodzących – rozróżnienie audio i poleceń tekstowych. Źródło: Opracowanie własne.

```
if isinstance(msg, bytes):
    audio_buffer.extend(msg)
elif isinstance(msg, str):
    if msg == "__END__":
        if audio_buffer:
            await utterance_queue.put(bytes(audio_buffer))
            audio_buffer.clear()
```

Przetwarzanie wypowiedzi odbywa się asynchronicznie za pośrednictwem kolejki (ang. *queue*) obsługiwanej przez dedykowany wątek roboczy (ang. *worker*). Dzięki temu serwer może przyjmować kolejne wiadomości od klienta bez oczekiwania na zakończenie bieżącego przetwarzania, co ma kluczowe znaczenie przy operacjach takich jak odpytywanie modelu językowego czy synteza mowy – obie potrafią trwać od ułamków sekundy do kilku sekund.

Protokół poleceń JSON obejmuje szereg typów wiadomości sterujących, które klient może wysłać do serwera. Tabela 16 przedstawia najważniejsze z nich.

Tabela 16 Typy poleceń JSON obsługiwanych przez serwer w kanale `WS_IN`. Źródło: Opracowanie własne.

Typ polecenia	Opis
<code>set_system_prompt</code>	Ustawia kontekst systemowy modelu językowego
<code>set_ai_model</code>	Wybiera backend modelu (Ollama lub Gemini)
<code>upload_file /</code> <code>upload_file_start</code>	Przesyła plik do modułu RAG

Typ polecenia

Opis

greeting_trigger

Inicjuje powitanie generowane przez model

end_session

Kończy sesję i żąda raportu

Przesyłanie plików do modułu *RAG* (ang. *Retrieval-Augmented Generation*) zrealizowano w trybie porcjowanym (ang. *chunked upload*) – plik dzielony jest na fragmenty zakodowane w formacie Base64, przesyłane kolejno jako osobne wiadomości JSON, a następnie składane po stronie serwera. Rozwiązanie to pozwala na przesyłanie plików o dowolnej wielkości bez ryzyka przekroczenia limitu rozmiaru pojedynczej wiadomości WebSocket.

6.2.2 Moduł mowy – STT, TTS i fonemizacja

Moduł mowy odpowiada za przetwarzanie głosu użytkownika oraz generowanie syntetycznej odpowiedzi awatara. Składa się z trzech współpracujących komponentów: rozpoznawania mowy (ang. *Speech-to-Text*, *STT*), syntezy mowy (ang. *Text-to-Speech*, *TTS*) oraz ekstrakcji fonemów (ang. *phoneme extraction*) wykorzystywanej do animacji ust awatara.

6.2.2.1 Rozpoznawanie mowy

Do rozpoznawania mowy zastosowano bibliotekę Vosk – otwartoźródłowy (ang. *open-source*) silnik STT działający lokalnie, bez konieczności połączenia z zewnętrznymi usługami. Przed właściwym rozpoznawaniem przeprowadzana jest wstępna filtracja ciszy na podstawie wartości RMS (ang. *Root Mean Square*) – fragmenty audio poniżej ustalonego progu są odrzucane bez angażowania modelu, co ogranicza zbędne przetwarzanie. Listing 3 przedstawia ten mechanizm.

Listing 3 Filtracja ciszy na podstawie RMS oraz rozpoznawanie mowy z wykorzystaniem biblioteki Vosk. Źródło: Opracowanie własne.

```
rms = float(np.sqrt(np.mean(audio_16k ** 2)))
RMS_THRESHOLD = 0.01
if rms < RMS_THRESHOLD:
    await send_text({"type": "no_voice", "rms": round(rms, 4)})
    return

recognizer = KaldiRecognizer(vosk_model, TARGET_SAMPLE_RATE)
recognizer.AcceptWaveform(pcm16_16k.tobytes())
res = json.loads(recognizer.Result())
text = res.get("text", "").strip()
```

Model Vosk wymaga audio w formacie 16-bitowym PCM (ang. *Pulse Code Modulation*) z częstotliwością próbkowania 16 000 Hz. Ze względu na to, że silnik Godot przesyła audio z częstotliwością 48 000 Hz, konieczne jest przepróbkowanie (ang. *resampling*) z wykorzystaniem biblioteki librosa przed przekazaniem danych do rozpoznawania.

6.2.2.2 Synteza mowy

Synteza mowy opiera się na bibliotece Coqui TTS z modelem VITS (ang. *Variational Inference with adversarial learning for end-to-end Text-to-Speech*) w wariantcie wielomówcowym VCTK. Jest to otwartoźródłowy model neuronowy generujący naturalnie brzmiącą mowę bezpośrednio z tekstu, bez pośrednich reprezentacji. Zdecydowano się na głos p228 – angielski głos męski – jako domyślny głos awatara.

Wygenerowane audio również wymaga przepróbkowania – model VITS produkuje sygnał z częstotliwością 22 050 Hz, który przed wysłaniem do klienta przeliczany jest do 48 000 Hz. Listing 4 przedstawia funkcję realizującą syntezę i przepróbkowanie.

Listing 4 Synteza mowy z wykorzystaniem modelu VITS oraz przepróbkowanie sygnału do częstotliwości wymaganej przez klienta. Źródło: Opracowanie własne.

```
def tts_to_pcm16_resampled(text: str) -> tuple[bytes, float]:
    wav = tts.tts(text=text, speaker="p228")
    wav = np.asarray(wav, dtype=np.float32)
    original_duration = len(wav) / XTTS_SR

    if XTTS_SR != GODOT_SAMPLE_RATE:
        wav = librosa.resample(wav, orig_sr=XTTS_SR,
                               target_sr=GODOT_SAMPLE_RATE)

    pcm16 = np.clip(wav * 32767, -32768, 32767).astype(np.int16)
    return pcm16.tobytes(), original_duration
```

Audio odpowiedzi wysyłane jest do klienta porcjami (ang. *chunks*) o rozmiarze odpowiadającym 20 ms sygnału, co zapewnia płynne odtwarzanie po stronie aplikacji klienckiej bez konieczności oczekiwania na wygenerowanie całej odpowiedzi.

6.2.2.3 Ekstrakcja fonemów

Aby animacja ust awatara była zsynchronizowana z generowaną mową, serwer wyznacza sekwencję fonemów wraz z ich znacznikami czasowymi. Proces ten realizowany jest przez bibliotekę Phonemizer z backendem eSpeak-ng, która konwertuje tekst na zapis fonetyczny w notacji IPA (ang. *International Phonetic Alphabet*). Następnie fonemy IPA mapowane są na uproszczony zestaw wizem (ang. *visemes*) – klas fonetycznych istotnych z punktu widzenia animacji twarzy. Tabela 17 przedstawia zastosowane odwzorowanie.

Tabela 17 Odwzorowanie fonemów IPA na wizemy wykorzystywane w animacji ust awatara. Źródło: Opracowanie własne.

Wizema	Fonemy IPA
A	a, ɑ, æ, ʌ
E	ɛ, e, ɜ, ə, ei
I	i, ɪ, j
O	ɔ, o, ɒ, ou
U	u, ʊ, w
P	p, b
M	m
F	f, v
T	t, d, n, θ, ð
S	s, z
SH	ʃ, ʒ, tʃ, dʒ

Wizema Fonemy IPA

K	k, g, ŋ, h
L	l
R	r, ɹ
SIL	cisza

Czas trwania każdego fonemu wyznaczany jest proporcjonalnie do jego wagi akustycznej względem całkowitej długości wygenerowanego audio. Samogłoski jako niosące główną energię akustyczną, otrzymują wyższe wagi niż spółgłoski zwarte. Po wyznaczeniu sekwencji fonemów dla wszystkich zdań odpowiedzi, serwer przesyła do klienta jeden zbiorczy komunikat JSON zawierający kompletną oś czasu fonemów, a następnie strumieniuje audio. Listing 5 ilustruje strukturę tego komunikatu.

Listing 5 Struktura komunikatu JSON z osią czasu fonemów przesyłanego do klienta przed strumieniowaniem audio. Źródło: Opracowanie własne.

```
phoneme_msg = {  
    "type": "phonemes",  
    "text": reply,  
    "duration": cumulative_time,  
    "data": all_phonemes  
}
```

6.2.3 Moduł analizy akustycznej

Moduł analizy akustycznej realizowany jest przez dedykowany plik `speech_analytics.py` i odpowiada za wyznaczanie obiektywnych miar jakości mowy użytkownika w trakcie każdej wypowiedzi oraz generowanie zbiorczego raportu po zakończeniu sesji. Podstawą ekstrakcji cech akustycznych jest biblioteka openSMILE (ang. *open-Source Media Interpretation by Large feature-space Extraction*) – otwartoźródłowe narzędzie szeroko stosowane w badaniach nad rozpoznawaniem emocji i analizą mowy.

6.2.3.1 Ekstrakcja cech akustycznych

Dla każdej wypowiedzi użytkownika równolegle uruchamiane są dwa zestawy cech (ang. *feature sets*):

- *eGeMAPSv02* (ang. *extended Geneva Minimalistic Acoustic Parameter Set*) – zestaw 88 cech obejmujący parametry takie jak jitter, shimmer, stosunek harmonicznych do szumów (ang. *Harmonics-to-Noise Ratio, HNR*), formanty oraz głośność;
- *IS10* – rozszerzony zestaw 1582 cech zawierający m.in. dodatkowe miary jittera (`jitterDDP`), obwiednię podstawowej częstotliwości (`F0finEnv`), logarytmiczne pasma Mela oraz statystyki struktury wypowiedzi.

Oba zestawy przetwarzane są równolegle z wykorzystaniem puli wątków (ang. *ThreadPoolExecutor*), co skraca czas analizy pojedynczej wypowiedzi. Listing 6 przedstawia mechanizm równoległej ekstrakcji.

```
fut_egemaps = _executor.submit(_smile_egemaps.process_signal, audio,
sample_rate)
fut_is10 = _executor.submit(_smile_is10.process_signal, audio,
sample_rate)
eg = fut_egemaps.result().iloc[0].to_dict()
i10 = fut_is10.result().iloc[0].to_dict()
```

6.2.3.2 Metryki jakości mowy

Na podstawie wyekstrahowanych cech wyznaczanych jest sześć metryk, każda w skali 0–100. Tabela 18 przedstawia ich zestawienie wraz z głównymi cechami wejściowymi.

Tabela 18 Metryki jakości mowy wyznaczane przez moduł analizy akustycznej. Źródło: Opracowanie własne.

Metryka	Opis	Główne cechy wejściowe
stress_score	Stres głosowy	jitter, shimmer, HNR, jitterDDP
confidence_score	Pewność siebie	stabilność F0, głośność, voicing
fluency_score	Płynność mowy	tempo, voiced ratio, długość pauz
expressiveness_score	Ekspresja głosu	zakres F0, głośności, pasma Mela
vocal_tension_score	Napięcie głosowe	Kurtoza, jitterDDP, shimmer, uplevel
articulation_score	Jakość artykulacji	formanty F1/F2, HNR, shimmer

Ocena ogólna (overall_score) stanowi ważoną średnią wszystkich metryk, przy czym stres głosowy i pewność siebie mają największy wpływ na wynik końcowy. Listing 7 przedstawia obliczanie oceny ogólnej.

Listing 7 Obliczanie ważonej oceny ogólnej jakości mowy. Źródło: Opracowanie własne.

```
overall_score = (
    (100 - stress_score) * 0.25 +
    confidence_score * 0.20 +
    fluency_score * 0.20 +
    expressiveness_score * 0.15 +
    (100 - vocal_tension_score) * 0.10 +
    articulation_score * 0.10
)
```

Warto zauważyć, że metryki stresu głosowego oraz napięcia głosowego są w formule negowane – wyższy stres i napięcie obniżają ocenę ogólną, zgodnie z intuicją oceny jakości wypowiedzi.

6.2.3.3 Raport z sesji

Po zakończeniu sesji funkcja generate_session_summary() agreguje dane ze wszystkich wypowiedzi i wyznacza wartości średnie, ekstremalne oraz trendy czasowe dla każdej metryki. Trend wyznaczany jest jako różnica średniej wartości z drugiej i pierwszej połowy sesji, co pozwala ocenić czy jakość mowy użytkownika poprawiała się, czy pogarszała w trakcie rozmowy. Listing 8 ilustruje wyznaczanie trendu.

```
def trend(attr):
    mid = n // 2
    a = sum(getattr(u, attr) for u in _session_utterances[:mid]) / mid
    b = sum(getattr(u, attr) for u in _session_utterances[mid:]) / (n -
mid)
    return round(b - a, 1)
```

Raport uzupełniany jest następnie przez dwa wywołania modelu językowego – pierwsze generuje analizę akustyczną na podstawie liczbowych wyników sesji, drugie tworzy podsumowanie merytoryczne przeprowadzonej rozmowy. Gotowy raport przesyłany jest do klienta jako pojedynczy komunikat JSON i stanowi podstawę ekranu podsumowania wyświetlanego użytkownikowi po zakończeniu symulacji.

6.2.4 Moduł RAG

Moduł RAG (ang. *Retrieval-Augmented Generation*) umożliwia wzbogacenie kontekstu modelu językowego o treści pochodzące z dokumentów dostarczonych przez użytkownika. Dzięki temu symulacja może odnosić się do konkretnej wiedzy dziedzinowej – przykładowo materiałów do egzaminu czy opisu stanowiska w przypadku rozmowy kwalifikacyjnej – bez konieczności dostrajania (ang. *fine-tuning*) samego modelu. Moduł jest opcjonalny i aktywuje się wyłącznie w przypadku przesłania pliku przez użytkownika na etapie konfiguracji.

6.2.4.1 Indeksowanie dokumentów

Obsługiwane są dwa formaty plików wejściowych: PDF oraz *plaintext* (TXT). Po odebraniu pliku przez serwer jest on wczytywany za pomocą dedykowanego *loadera* z biblioteki LangChain – odpowiednio PyPDFLoader lub TextLoader – a następnie dzielony na fragmenty (ang. *chunks*) przy użyciu klasy RecursiveCharacterTextSplitter. Zastosowano rozmiar fragmentu wynoszący 500 znaków z zakładką (ang. *overlap*) 50 znaków, co zapewnia ciągłość kontekstu na granicach podziału. Listing 9 przedstawia proces wczytywania i podziału dokumentu.

Listing 9 Wczytywanie dokumentu i podział na fragmenty z zakładką. Źródło: Opracowanie własne.

```
if file_type == "pdf":
    loader = PyPDFLoader(file_path)
elif file_type == "txt":
    loader = TextLoader(file_path, encoding='utf-8')

documents = await asyncio.to_thread(loader.load)

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=RAG_CHUNK_SIZE,
    chunk_overlap=RAG_CHUNK_OVERLAP
)
splits = await asyncio.to_thread(text_splitter.split_documents,
documents)
```

Operacje wczytywania i podziału uruchamiane są w osobnym wątku za pośrednictwem `asyncio.to_thread`, aby nie blokować głównej pętli zdarzeń serwera podczas przetwarzania większych plików.

6.2.4.2 Wektorowa baza wiedzy

Każdy fragment dokumentu przekształcany jest w wektor numeryczny (ang. *embedding*) przy użyciu modelu `sentence-transformers/all-MiniLM-L6-v2` z biblioteki HuggingFace. Jest to lekki, otwartoźródłowy model działający lokalnie, generujący wektory o wymiarze 384. Wektory przechowywane są w bazie FAISS (ang. *Facebook AI Similarity Search*) – wydajnej bibliotece do przeszukiwania przestrzeni wektorowych. W przypadku przesłania kolejnego dokumentu w trakcie sesji nowy indeks FAISS jest scalany z istniejącym, co pozwala na korzystanie z wielu źródeł wiedzy jednocześnie.

6.2.4.3 Wyszukiwanie kontekstu

Przed każdym zapytaniem do modelu językowego moduł RAG wyszukuje trzy najbardziej semantycznie zbliżone fragmenty do treści wypowiedzi użytkownika metodą przeszukiwania według podobieństwa (ang. *similarity search*). Znalezione fragmenty są następnie dołączane do zapytania jako kontekst, zgodnie ze schematem przedstawionym w Listing 10.

Listing 10 Wyszukiwanie semantycznie zbliżonych fragmentów dokumentu względem zapytania użytkownika. Źródło: Opracowanie własne.

```
def retrieve_context(query: str) -> str:
    docs = rag_vectorstore.similarity_search(query, k=RAG_TOP_K)
    context = "\n\n".join([
        f"[Document {i + 1}]\n{doc.page_content}"
        for i, doc in enumerate(docs)
    ])
    return context
```

Rozszerzony prompt przekazywany do modelu zawiera wyraźne oznaczenie granic kontekstu dokumentowego oraz oryginalną wypowiedź użytkownika, co pozwala modelowi językowemu właściwie rozróżnić wiedzę dostarczaną zewnątrz od swojej wiedzy wbudowanej. Listing 11 przedstawia strukturę tak zbudowanego zapytania.

Listing 11 Struktura zapytania do modelu językowego wzbogaconego o kontekst z modułu RAG. Źródło: Opracowanie własne.

```
user_message = f"""Context from documents:
{rag_context}

User question: {prompt}

Please answer based on the provided context when relevant."""
```

Istotną cechą zastosowanego rozwiązania jest jego niezależność od konkretnego modelu językowego – mechanizm wyszukiwania i dołączania kontekstu działa identycznie niezależnie od tego, czy jako backend wybrany został lokalny model Ollama czy chmurowy Gemini. Pozwala to na swobodną wymianę modelu bez ingerencji w logikę modułu RAG.

6.2.5 Integracja z modelami językowymi (Ollama / Gemini)

Aplikacja serwerowa wspiera współpracę z różnymi modelami językowymi (ang. *Large Language Models, LLM*) dzięki zastosowaniu wzorca fasady (ang. *facade pattern*) – pojedynczej funkcji `ask_ai()` stanowiącej punkt wejścia do warstwy modeli. Funkcja ta deleguje zapytanie do właściwego

backendu na podstawie aktualnie wybranego modelu, ukrywając przed resztą systemu szczegóły komunikacji z konkretnym dostawcą. Listing 12 przedstawia tę funkcję.

Listing 12 Funkcja fasady delegująca zapytanie do wybranego backendu modelu językowego. Źródło: Opracowanie własne.

```
def ask_ai(prompt: str) -> str:
    if AI_MODEL == "gemini":
        return ask_gemini(prompt)
    else:
        return ask_ollama(prompt)
```

Dzięki takiemu podejściu dodanie obsługi kolejnego modelu – przykładowo Claude firmy Anthropic, GPT firmy OpenAI czy dowolnego innego dostawcy udostępniającego interfejs API – sprowadza się do implementacji nowej funkcji zgodnej z sygnaturą `ask_{nazwa}(prompt: str) -> str` oraz dodania odpowiedniego przypadku w funkcji `ask_ai()`. Pozostałe moduły systemu, w tym moduł RAG, historia konwersacji czy generator raportów, działają bez żadnych modyfikacji.

6.2.5.1 Dostępne backendy LLM

W prototypie zaimplementowano dwa *backendy* LLM. Pierwszy z nich to Ollama – otwarteźródłowe (ang. *open-source*) narzędzie umożliwiające uruchamianie modeli językowych lokalnie na komputerze użytkownika. Komunikacja odbywa się przez lokalny interfejs REST (ang. *Representational State Transfer*) pod adresem `http://127.0.0.1:11434/api/chat`. Domyślnie skonfigurowany model to `gemma3`, jednak zmiana na dowolny inny model dostępny w Ollama wymaga jedynie modyfikacji pojedynczej stałej konfiguracyjnej `OLLAMA_MODEL`. Zaletą tego rozwiązania jest pełna prywatność danych – żadna wypowiedź użytkownika nie opuszcza lokalnego środowiska.

Drugi backend to Gemini – chmurowy model językowy firmy Google, dostępny przez oficjalne *SDK* (ang. *Software Development Kit*) dla języka Python. Wymaga podania klucza API (ang. *Application Programming Interface*), który użytkownik konfiguruje z poziomu aplikacji klienckiej. Listing 13 przedstawia inicjalizację modelu Gemini po otrzymaniu klucza od klienta.

Listing 13 Inicjalizacja modelu Gemini z kluczem API i promptem systemowym przekazany przez klienta. Źródło: Opracowanie własne.

```
genai.configure(api_key=api_key)
GEMINI_MODEL = genai.GenerativeModel(
    model_name='gemini-3.1-flash-lite-preview',
    system_instruction=SYSTEM_PROMPT
)
```

6.2.5.2 Zarządzanie historią konwersacji

Niezależnie od wybranego backendu, serwer utrzymuje wspólną historię konwersacji przechowywaną jako lista komunikatów w formacie zgodnym ze standardem OpenAI (`role / content`). Historia ograniczona jest do 20 ostatnich wiadomości, co zapobiega przekroczeniu limitu kontekstu modelu przy długich sesjach. Każde zapytanie do modelu zawiera pełną dostępną historię, co pozwala modelowi odnosić się do wcześniejszych fragmentów rozmowy.

6.2.5.3 Dynamiczna zmiana modelu

Model językowy może być zmieniany przez użytkownika w czasie rzeczywistym bez restartu serwera – aplikacja kliencka wysyła polecenie `set_ai_model` z nazwą backendu oraz opcjonalnym kluczem API. Serwer potwierdza zmianę komunikatem `ai_model_set` lub zwraca błąd `ai_model_error` w przypadku niepowodzenia konfiguracji, przykładowo przy nieprawidłowym kluczu API. Zmiana modelu nie wpływa na bieżącą historię konwersacji ani stan modułu RAG – sesja jest kontynuowana w sposób ciągły.

6.3. Aplikacja kliencka (Godot)

Aplikacja kliencka została zrealizowana w silniku Godot w wersji 4 – otwartoźródłowym (ang. *open-source*) silniku do tworzenia gier i aplikacji interaktywnych, oferującym natywne wsparcie dla technologii wirtualnej rzeczywistości (ang. *Virtual Reality, VR*) poprzez standard OpenXR. Architektura klienta opiera się na systemie węzłów (ang. *nodes*) i scen charakterystycznym dla silnika Godot, uzupełnionym o dwa globalne singletony dostępne w całej aplikacji poprzez mechanizm `AutoLoad`: `SpeechClient` odpowiedzialny za komunikację z serwerem oraz obsługę audio, a także `ConnectionConfig` przechowujący parametry bieżącej sesji. Poniższe podrozdziały opisują poszczególne komponenty aplikacji klienckiej.

6.3.1 Komunikacja z serwerem i obsługa audio

Skrypt `SpeechClient` stanowi centralny komponent aplikacji klienckiej, pełniąc rolę pośrednika między interfejsem użytkownika, silnikiem audio Godot a serwerem Python. Jako singleton `AutoLoad` jest dostępny globalnie z każdego miejsca w aplikacji.

Połączenie z serwerem nawiązywane jest przez pojedyncze gniazdo `WebSocket` ws odpytywane co klatkę w metodzie `_process()`, co zapewnia ciągłość komunikacji bez blokowania głównego wątku silnika. Protokół `WebSocket` jest dwukierunkowy (ang. *full-duplex*), dzięki czemu to samo połączenie służy jednocześnie do wysyłania audio i poleceń oraz odbierania odpowiedzi serwera. Weryfikację nawiązania połączenia realizuje singleton `ConnectionConfig`, który cyklicznie sprawdza stan gniazda i emituje sygnał `connected` po jego otwarciu, jak przedstawiono w Listing 14.

Listing 14 Weryfikacja nawiązania połączenia WebSocket w singletons ConnectionConfig. Źródło: Opracowanie własne.

```
func _process(delta: float) -> void:
    if working:
        ws_in.poll()
        if ws_in.get_ready_state() == WebSocketPeer.STATE_OPEN:
            connectedEmit()
```

Po nawiązaniu połączenia `SpeechClient` kolejno wysyła prompt systemowy, konfigurację modelu językowego oraz opcjonalnie wyzwała powitanie generowane przez sztuczną inteligencję, jeśli taka opcja została wybrana przez użytkownika w konfiguracji sesji.

Odebrane z serwera wiadomości tekstowe przetwarzane są przez metodę `handle_text()`, która parsuje przychodzący JSON i reaguje na każdy typ komunikatu za pomocą instrukcji `match`. Listing 15 przedstawia obsługę kluczowych typów wiadomości.

```
match msg.get("type", ""):
    "final":
        print("STT:", msg.get("text", ""))
        ai_thinking.emit()
    "phonemes":
        var phoneme_data = msg.get("data", [])
        var duration = msg.get("duration", 0.0)
        var text = msg.get("text", "")
        phonemes_received.emit(phoneme_data, duration, text)
        phoneme_timeline_ready.emit(phoneme_data, text)
    "tts_end":
        tts_ended.emit()
    "no_voice":
        tts_ended.emit()
    "session_summary":
        _save_session_summary(msg)
    "rag_loaded":
        rag_loaded.emit(msg.get("filename", ""), msg.get("chunks", 0))
    "rag_error":
        rag_error.emit(msg.get("message", ""))
```

Komunikat `final` oznacza pomyślne rozpoznanie mowy użytkownika i wyzwala sygnał `ai_thinking` zmieniający wskaźnik stanu interfejsu. Komunikat `phonemes` powoduje emisję dwóch sygnałów – `phonemes_received` trafia do systemu napisów, natomiast `phoneme_timeline_ready` trafia do węzła synchronizacji ust. Po zakończeniu sesji komunikat `session_summary` wyzwala zapis raportu do pliku JSON i nawigację do ekranu statystyk.

6.3.1.1 Nagrywanie i wysyłanie audio

Nagrywanie głosu użytkownika realizowane jest przez efekt `AudioEffectRecord` na dedykowanej magistrali (ang. *bus*) `Record` w silniku Godot. Nagrywanie rozpoczyna się po wciśnięciu klawisza spacji i kończy po jego zwolnieniu z dodatkowym opóźnieniem 0,7 sekundy, aby uniknąć obcinania końcówek wypowiedzi. Nagrane audio w formacie WAV (ang. *Waveform Audio File Format*) jest następnie konwertowane do formatu PCM (ang. *Pulse Code Modulation*) 16-bitowego mono i dzielone na porcje (ang. *chunks*) o rozmiarze 4096 bajtów, wysyłane kolejno po jednej na klatkę. Zakończenie przesyłania sygnalizowane jest znacznikiem tekstowym `__END__`. Listing 16 przedstawia przygotowanie kolejki wysyłania.

```
func prepare_send_queue(wav: AudioStreamWAV) -> void:
    var data: PackedByteArray = wav.data
    if wav.stereo:
        data = stereo_to_mono_pcm16(data)
    send_queue.clear()
    var offset: int = 0
    while offset < data.size():
        var end: int = min(offset + SEND_CHUNK_BYTES, data.size())
        send_queue.append(data.slice(offset, end))
        offset = end
```

W przypadku nagrania stereofonicznego wykonywana jest konwersja do mono przez uśrednienie próbek lewego i prawego kanału, ponieważ serwer Vosk oczekuje audio jednokanałowego.

6.3.1.2 Odtwarzanie syntetycznej mowy

Odtwarzanie odpowiedzi awatara opiera się na węźle `AudioStreamGenerator` – generatorze audio czasu rzeczywistego silnika Godot, zasilanym próbkami PCM dostarczany programowo. Odbierane z serwera porcje binarne trafiają do wewnętrznego bufora `tts_buffer`, skąd metoda `pump_tts()` wywołana co klatkę przepycha je partiami do generatora, jak pokazano w Listing 17. Zastosowanie bufora pośredniego eliminuje ryzyko utraty próbek w przypadku chwilowego przepełnienia bufora generatora.

Listing 17 Przepompowywanie próbek PCM z bufora do generatora audio silnika Godot. Źródło; Opracowanie własne.

```
func pump_tts() -> void:
    while (tts_offset + 1) < tts_buffer.size() \
        and playback.can_push_buffer(PUSH_FRAMES_BATCH):
        var frames_pushed: int = 0
        while frames_pushed < PUSH_FRAMES_BATCH \
            and (tts_offset + 1) < tts_buffer.size():
            var s: int = tts_buffer.decode_s16(tts_offset)
            var f: float = float(s) / 32768.0
            playback.push_frame(Vector2(f, f))
            tts_offset += 2
            frames_pushed += 1
        if tts_offset >= tts_buffer.size():
            tts_buffer = PackedByteArray()
            tts_offset = 0
```

Próbki PCM 16-bitowe ze znakiem dekodowane są do wartości zmiennoprzecinkowych w zakresie $-1,0$ do $1,0$ i przesyłane jako ramki stereofoniczne z identyczną wartością na obu kanałach, ponieważ syntezytor serwera produkuje dźwięk monofoniczny.

6.3.2 Synchronizacja ruchu ust

Synchronizacja ruchu ust awatara z generowaną mową realizowana jest przez skrypt `LipsyncNode` działający w oparciu o oś czasu fonemów (ang. *phoneme timeline*) przesyłaną przez serwer. Po odebraniu komunikatu JSON typu `phonemes` węzeł `SpeechClient` emituje sygnał `phoneme_timeline_ready`, który trafia do skryptu `LipsyncNode` podłączonego do węzła `AnimationPlayer` awatara.

Każdy fonem z osi czasu odwzorowany jest na odpowiednią animację o nazwie `Play_<FONEM>` – przykładowo fonem A uruchamia animację `Play_A`, a cisza (SIL) uruchamia animację `Play_SIL` odpowiadającą zamkniętym ustom. W metodzie `_process()` skrypt śledzi bieżący czas odtwarzania i przełącza animację w momencie zmiany aktywnego fonemu, z konfigurowalnym czasem mieszania (ang. *blend time*) wynoszącym domyślnie 100 ms, co zapewnia płynne przejścia między pozycjami ust. Listing 18 przedstawia logikę wyboru aktywnego fonemu.

Listing 18 Wybór aktywnej animacji fonemu na podstawie bieżącego czasu odtwarzania osi czasu. Źródło: Opracowanie własne.

```
func _process(delta):
    timeline_playback_time += delta
    var playback_time = timeline_playback_time + sync_offset
    var current_cue = get_phoneme_at_time(playback_time)

    var target_phoneme = current_cue["phoneme"]
    if target_phoneme != current_phoneme:
        current_phoneme = target_phoneme
        play_animation(target_phoneme)
```

Skrypt przewiduje również mechanizm wydłużania zbyt krótkich momentów ciszy – przerwy krótsze niż skonfigurowany próg (domyślnie 150 ms) są automatycznie rozszerzane, co zapobiega zbyt gwałtownemu zamykaniu ust między słowami. Po wydłużeniu czasów trwania pozycje startowe wszystkich kolejnych fonemów są przeliczane, aby zachować spójność osi czasu.

Wczytywanie osi czasu fonemów realizuje metoda `load_phoneme_timeline()`, która oprócz konwersji formatu danych wprowadza korekcję zbyt krótkich momentów ciszy. Listing 19 przedstawia ten mechanizm.

Listing 19 Wczytywanie osi czasu fonemów z korekcją minimalnego czasu trwania ciszy. Źródło: Opracowanie własne.

```
for i in range(phonemes.size()):
    var ph = phonemes[i]
    var duration = ph["duration"]

    if ph["phoneme"] == "SIL" and duration < min_silence_duration:
        var is_at_end = (i == phonemes.size() - 1)
        if (is_at_end and extend_silence_at_end) or \
            (not is_at_end and extend_silence_between_words):
            duration = min_silence_duration

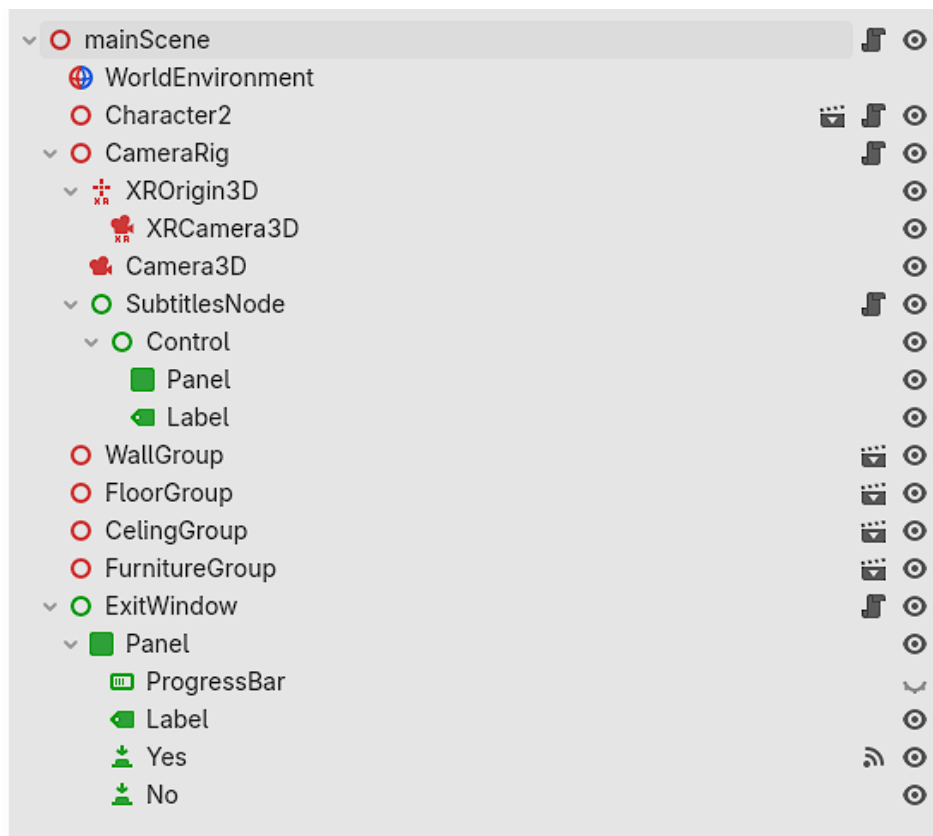
    phoneme_timeline.append({
        "phoneme": ph["phoneme"],
        "start": ph["start"],
        "end": ph["start"] + duration,
        "duration": duration
    })

    _recalculate_timeline_positions()
```

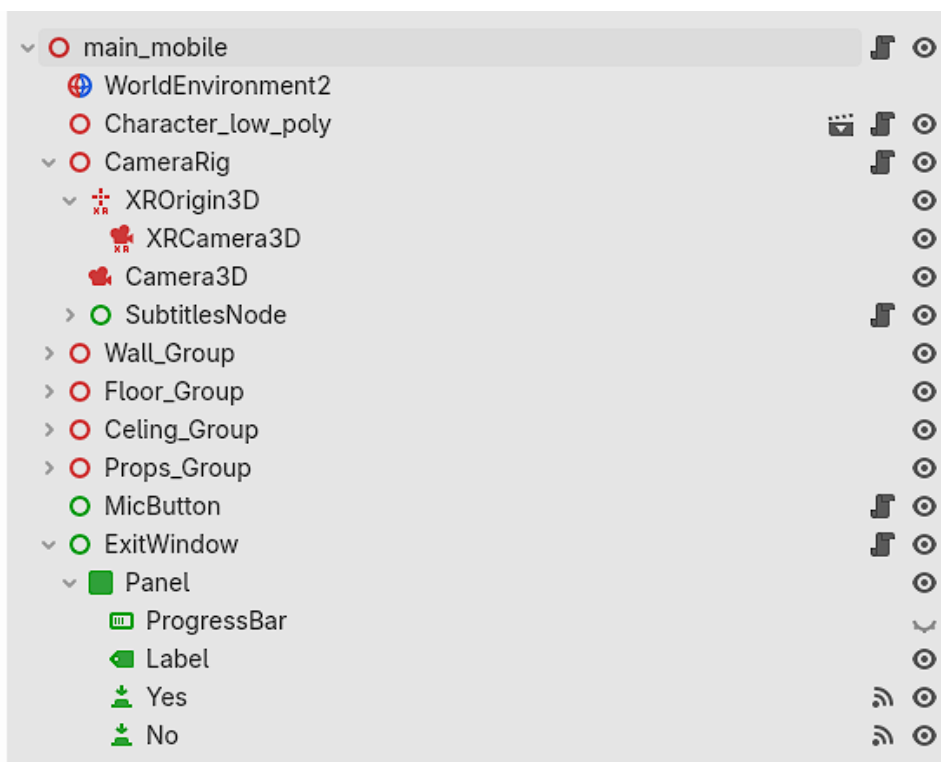
Decyzja o wydłużeniu ciszy podejmowana jest osobno dla przerw między słowami oraz ciszy końcowej, co umożliwia niezależną konfigurację obu przypadków. Po wprowadzeniu korekt metoda `_recalculate_timeline_positions()` przelicza pozycje startowe wszystkich kolejnych fonemów, zachowując spójność całej osi czasu.

6.3.3 Sceny

Aplikacja zawiera dwie wersje sceny głównej dostosowane do różnych platform docelowych, których struktury węzłów przedstawiają odpowiednio Rysunek 40 oraz Rysunek 41.



Rysunek 40 Struktura węzłów sceny głównej w wersji wysokiej szczegółowości (mainScene). Źródło: Opracowanie własne.



Rysunek 41 Struktura węzłów sceny głównej w wersji uproszczonej przeznaczonej na urządzenia mobilne (main_mobile). Źródło: Opracowanie własne.

Wersja `mainScene` przeznaczona jest na komputery osobiste oraz gogle VR i zawiera szczegółowy model środowiska złożony z grup meszy (ang. *mesh*): `WallGroup`, `FloorGroup`, `CelingGroup` oraz `FurnitureGroup`, tworzących realistycznie wyglądające pomieszczenie biurowe. Wersja `main_mobile` zawiera uproszczony model w stylu *low poly* – zredukowaną geometrię zastąpiono węzłem `Character_low_poly`, a grupy elementów środowiska zostały odpowiednio uproszczone. Zamiast przechwytywania dźwięku klawiaturą, wersja mobilna udostępnia węzeł `MicButton` – ekranowy przycisk do nagrywania dostosowany do obsługi dotykowej.

Obie sceny zawierają wspólne elementy: węzeł `CameraRig` z obsługą OpenXR umożliwiającą korzystanie z gogli VR, węzeł `SubtitlesNode` wyświetlający napisy generowanej mowy awatara oraz węzeł `ExitWindow` z oknem dialogowym potwierdzenia zakończenia sesji.

Inicjalizacja sceny obejmuje uruchomienie interfejsu OpenXR, a w przypadku niedostępności gogli – automatyczne przełączenie w tryb standardowy z widokiem z perspektywy pierwszej osoby. Skrypt sceny rejestruje również węzeł awatara w grupie `lipsync`, co umożliwia jego odnalezienie przez `SpeechClient` i podłączenie synchronizacji ust. Listing 20 przedstawia inicjalizację sceny.

Listing 20 Inicjalizacja sceny z próbą uruchomienia interfejsu OpenXR. Źródło: Opracowanie własne.

```
func _ready():
    var VR = XRServer.find_interface("OpenXR")
    if VR and VR.initialize():
        get_viewport().use_xr = true
    else:
        push_warning("OpenXR niedostępne, tryb normalny")
    SpeechClient.character =
get_tree().get_first_node_in_group("lipsync")
SpeechClient.connectToLipSync()
```

6.3.4 Generyczne animacje awatara

Aplikacja zawiera trzy niezależne systemy animacji awatara, z których każdy odpowiada za inny aspekt jego zachowania.

Pierwszym z nich jest system animacji ciała oparty na węźle `AnimationTree` z maszyną stanów (ang. *state machine*). Awatar przełącza się cyklicznie co 60 sekund między dwoma pozami spoczynkowymi: `HandsOnLaps` – siedzącą ze złożonymi na kolanach rękami oraz `HandsOnDesk` – z rękami opartymi na biurku. Przejścia między stanami realizowane są przez pośrednie stany animacji zapewniające płynny ruch, a docelowa poza wybierana jest losowo, co zapobiega mechanicznej powtarzalności zachowania awatara.

Drugim systemem jest animacja mrugania, realizowana przez skrypt `BlinkController` operujący na morfy (ang. *blend shapes*) siatki (ang. *mesh*) awatara. Mruganie wyzwalane jest w losowych odstępach czasu z zakresu od 2 do 6 sekund, a sama animacja przebiega dwufazowo – zamknięcie i otwarcie oczu – z konfigurowalnymi parametrami czasu trwania.

Trzecim systemem jest śledzenie głowy (ang. *head tracking*) realizowane przez skrypt `HeadLookAt`, który obraca kość głowy szkieletu (ang. *skeleton*) awatara w kierunku kamery gracza. Rotacja wygładzana jest interpolacją sferyczną (ang. *Spherical Linear Interpolation, SLERP*) z konfigurowalnymi ograniczeniami kątowymi osobno dla osi pionowej i poziomej, co sprawia, że awatar naturalnie podąża wzrokiem za użytkownikiem pozostając w ramach anatomicznie wiarygodnego zakresu ruchu.

6.3.5 Wskaźnik stanu i napisy

Aplikacja udostępnia dwa dodatkowe elementy interfejsu informujące użytkownika o bieżącym stanie systemu.

Wskaźnik stanu realizowany jest przez skrypt `StatusIndicatorImage`, który wyświetla ikonę w rogu ekranu za pośrednictwem warstwy płótna (ang. *CanvasLayer*) renderowanej ponad sceną 3D. Ikona zmienia się w zależności od aktualnej fazy działania systemu – nasłuchiwanie, przetwarzania mowy, generowania odpowiedzi przez model lub odtwarzania syntezy mowy. Przejścia między ikonami realizowane są animacjami tweenowania (ang. *tweening*) z efektem zanikania. Każdej fazie towarzyszy dodatkowo animacja ikony: pulsowanie podczas oczekiwania na odpowiedź modelu, obrót podczas ładowania pliku RAG oraz odbijanie podczas odtwarzania mowy. Wskaźnik subskrybuje sygnały emitowane przez `SpeechClient` i zmienia wyświetlaną ikonę reaktywnie, bez potrzeby odpytywania stanu systemu.

Przypisanie ikony i animacji do danego stanu realizowane jest w metodzie `set_status()`, przedstawionej w Listing 21. Każdy stan posiada dedykowaną teksturę oraz animację pętlową podkreślającą charakter wykonywanej operacji.

Listing 21 Zmiana ikony i animacji wskaźnika stanu w metodzie `set_status()`. Źródło: Opracowanie własne.

```
func set_status(status: String):
    if current_status == status:
        return
    current_status = status

    var texture: Texture2D = null
    match status:
        "listening": texture = listening_texture
        "thinking":  texture = thinking_texture
        "speaking":  texture = speaking_texture
        "loading":   texture = loading_texture

    texture_rect.texture = texture
    texture_rect.visible = true

    tween = create_tween()
    tween.tween_property(texture_rect, "modulate:a", 1.0, fade_duration)

    anim_player.stop()
    match status:
        "thinking": anim_player.play("anim/pulse")
        "loading":  anim_player.play("anim/spin")
        "speaking": anim_player.play("anim/bounce")
```

Tekstury ikon przypisywane są w inspektorze silnika Godot jako eksportowane właściwości węzła, co pozwala na ich wymianę bez modyfikacji kodu. Stan `listening` nie posiada dedykowanej animacji pętlowej – ikona wyświetlana jest statycznie, co wizualnie odróżnia bierny tryb nasłuchiwania od aktywnych operacji przetwarzania.

Napisy generowanej mowy wyświetlane są przez skrypt `TTSSubtitles` jako panel tekstowy z zaokrąglonymi rogami i animowanym pojawianiem się i znikaniem. Pełny tekst odpowiedzi awatara dzielony jest na zdania, które wyświetlane są kolejno w synchronizacji z czasem trwania odpowiadających im fonemów. Czas wyświetlania każdego zdania wyznaczany jest proporcjonalnie do jego długości względem całej odpowiedzi.

7. Opis prototypu aplikacji

W niniejszym rozdziale opisano prototyp aplikacji Stressless VR powstały w wyniku realizacji niniejszej pracy. Omówiono kluczowe przykłady użycia ilustrujące przebieg wybranych scenariuszy symulacji dostępnych w prototypie wraz z opisem ekranu menu głównego i podsumowania sesji.

7.1. Ekran startowy

Rysunek 42 przedstawia ekran głównego menu aplikacji Stressless VR. W górnej części ekranu widoczne jest logo aplikacji składające się z ikony gogli VR z wbudowanym wykresem fali akustycznej oraz nazwy StressLess VR. W tle rozmyty widok sceny biurowej z sylwetką awatara podkreśla charakter aplikacji już na etapie ekranu startowego.

Menu główne udostępnia trzy opcje nawigacyjne:

- Start – przejście do ekranu konfiguracji sesji, gdzie użytkownik wybiera scenę, model językowy oraz definiuje parametry symulacji;
- Statistics – przejście do ekranu historii sesji umożliwiającego przeglądanie wyników poprzednich symulacji;
- Exit – zamknięcie aplikacji.



Rysunek 42 Ekran głównego menu aplikacji Stressless VR. Źródło: Opracowanie własne.

7.2. Konfiguracja ustawień

Pierwszym krokiem wymagany przed rozpoczęciem symulacji jest konfiguracja sesji realizowana za pośrednictwem ekranu *AI Voice Assistant Setup*, przedstawionego na Rysunek 43 oraz Rysunek 44.

Ekran konfiguracji umożliwia ustawienie następujących parametrów:

- wybór sceny (*Select Scene*),

- adres IP serwera (Server IP),
- *prompt* systemowy (System Prompt),
- dokument RAG (RAG Document),
- tryb inicjacji rozmowy (AI starts conversation),
- wybór modelu językowego (AI Model).

Wybór modelu językowego wpływa bezpośrednio na dostępność dodatkowego pola konfiguracyjnego. W przypadku wybrania opcji Ollama (Local), przedstawionej na Rysunek 43, pole klucza API nie jest wyświetlane – model działa lokalnie i nie wymaga autoryzacji zewnętrznej. Natomiast wybranie opcji Gemini (Google API), przedstawionej na Rysunek 44, powoduje pojawienie się dodatkowego pola Google API Key, w którym użytkownik zobowiązany jest podać klucz autoryzacyjny wymagany do komunikacji z usługą Google.

The screenshot displays the 'AI Voice Assistant Setup' interface. At the top, it says 'Select Scene' with a dropdown menu currently showing 'University' and 'Office'. Below this, the 'Server IP' is set to '192.168.0.162'. The 'System Prompt' section contains a detailed instruction: 'You are an AI avatar acting as a professional IT recruiter. Conduct a realistic, friendly, and natural job interview for the position of IT Analyst. Start by greeting the candidate, briefly explain the purpose of the interview, and ask them to introduce themselves, including their experience, education, and motivation. Ask one question at a time and follow up naturally based on the candidate's answers. Focus'. The 'RAG Document (optional)' section shows a 'Select PDF or TXT' button and a message 'No file selected'. There is a checkbox for 'AI starts conversation (greet you first)' which is currently unchecked. The 'AI Model' dropdown is set to 'Ollama (Local)'. At the bottom, there are three buttons: 'Ready to connect', 'Connect to Server' (with a rocket icon), and 'Back to menu'.

Rysunek 43 Ekran konfiguracji sesji z wybranym modelem Ollama (Local). Źródło: Opracowanie własne.

AI Voice Assistant Setup

Select Scene

University
Office

Server IP: 192.168.0.162

System Prompt:

You are an AI avatar acting as a professional IT recruiter. Conduct a realistic, friendly, and natural job interview for the position of IT Analyst.

Start by greeting the candidate, briefly explain the purpose of the interview, and ask them to

RAG Document (optional):

Select PDF or TXT

No file selected

☒ AI starts conversation (greet you first)

AI Model: Gemini (Google API)

Google API Key:

Ready to connect

Connect to Server

Back to menu

Rysunek 44 Ekran konfiguracji sesji z wybranym modelem Gemini (Google API). Źródło: Opracowanie własne.

Po uzupełnieniu parametrów użytkownik inicjuje połączenie przyciskiem **Connect to Server**. Aplikacja kliencka nawiązuje wówczas połączenie WebSocket z serwerem, przesyła skonfigurowany prompt systemowy oraz wybrany model językowy, a następnie opcjonalnie wczytuje dokument RAG. Po pomyślnym nawiązaniu połączenia, następuje przejście do wybranej sceny.

7.3. Rozmowa egzaminacyjna

Niniejszy podrozdział przedstawia przebieg przykładowej symulacji rozmowy egzaminacyjnej przeprowadzonej w scenie **University**. Scenariusz ten symuluje sytuację egzaminu ustnego, w którym użytkownik odpowiada na pytania zadawane przez cyfrowego awatara pełniącego rolę egzaminatora.

Po nawiązaniu połączenia z serwerem i załadowaniu sceny użytkownik zostaje umieszczony w wirtualnej sali przypominającej salę wykładową. Naprzeciwko znajduje się awatar siedzący przy biurku z rękami złożonymi na blacie, jak przedstawiono na Rysunek 45. Scena utrzymana jest w realistycznej stylistyce – widoczne są tablica, projektor, zegar oraz laptop na biurku, co wzmacnia poczucie immersji i realizmu symulowanej sytuacji.

Przebieg symulacji składa się z następujących faz, zilustrowanych na Rysunek 45–Rysunek 49:



Rysunek 45 Widok sceny University po załadowaniu – awatar oczekuje na wypowiedź użytkownika. Źródło: Opracowanie własne.

Faza nasłuchiwania – użytkownik inicjuje wypowiedź przytrzymując klawisz spacji. W tym czasie na ekranie pojawia się ikona mikrofonu, widoczna na Rysunek 46, sygnalizująca, że aplikacja aktywnie rejestruje dźwięk z mikrofonu. Zwolnienie klawisza kończy nagrywanie i uruchamia przesyłanie audio do serwera.



Rysunek 46 Faza nasłuchiwania – ikona mikrofonu sygnalizuje aktywne nagrywanie wypowiedzi użytkownika. Źródło: Opracowanie własne.

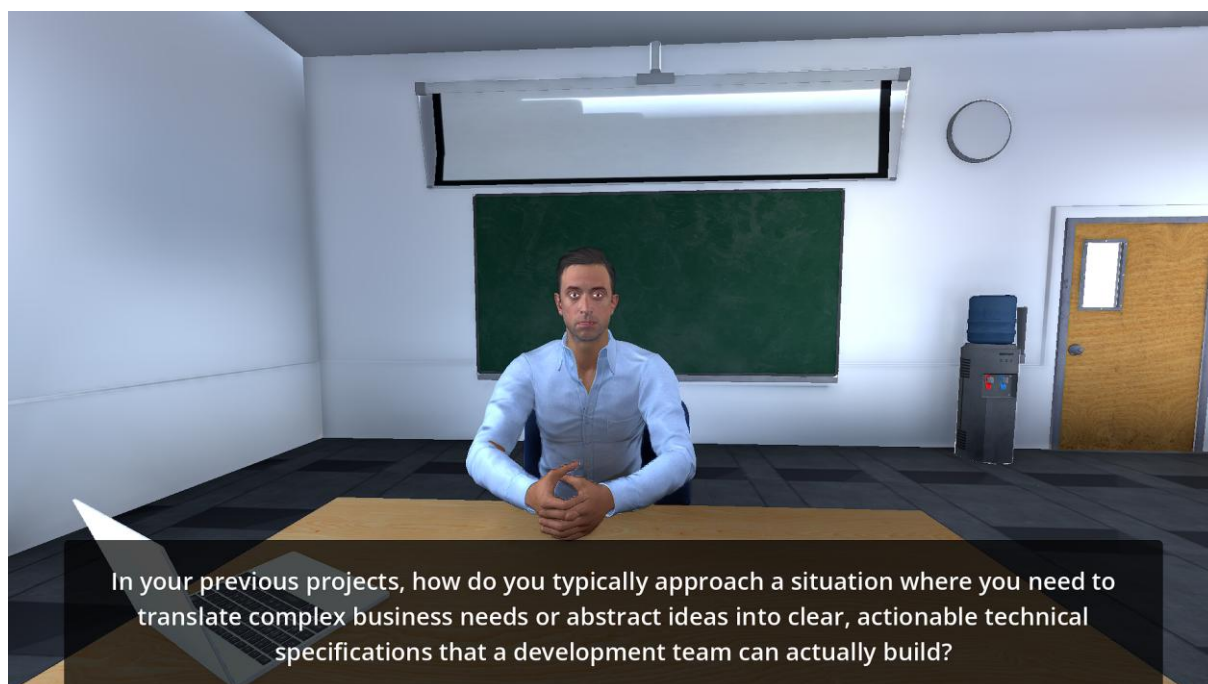
Faza przetwarzania – po przesłaniu nagrania serwer przeprowadza rozpoznawanie mowy, analizę akustyczną oraz odpytuje model językowy. W tym czasie nad awatarem wyświetlany jest dymek

z wielokropkiem, przedstawiony na Rysunek 47, informujący użytkownika, że awatar przetwarza wypowiedź i przygotowuje odpowiedź.



Rysunek 47 Faza przetwarzania – dymek nad awatarem sygnalizuje oczekiwanie na odpowiedź modelu językowego. Źródło: Opracowanie własne.

Faza odpowiedzi – po wygenerowaniu odpowiedzi przez model językowy dymek znika i awatar rozpoczyna mówienie. Ruch ust awatara zsynchronizowany jest z generowaną mową za pośrednictwem systemu synchronizacji fonemów. Jednocześnie na dole ekranu wyświetlane są napisy na żywo prezentujące treść wypowiedzi awatara, co przedstawia Rysunek 48.



Rysunek 48 Faza odpowiedzi – awatar mówi z synchronizacją ust, napisy wyświetlane są na dole ekranu. Źródło: Opracowanie własne.

Faza generowania raportu – po zakończeniu symulacji i wybraniu opcji zakończenia sesji wyświetlany jest ekran oczekiwania z animowanym wskaźnikiem postępu oraz komunikatem „Please wait...”, przedstawiony na Rysunek 49. W tym czasie serwer generuje raport akustyczny oraz merytoryczne podsumowanie rozmowy z wykorzystaniem modelu językowego. Po zakończeniu generowania aplikacja automatycznie przełącza scenę na ekran wyników.



Rysunek 49 Ekran oczekiwania na wygenerowanie raportu po zakończeniu symulacji. Źródło: Opracowanie własne.

7.4. Rozmowa kwalifikacyjna – low poly

Niniejszy podrozdział przedstawia przebieg symulacji rozmowy kwalifikacyjnej zrealizowanej w wersji *low poly* aplikacji, przeznaczonej na urządzenia mobilne. Scenariusz ten symuluje rozmowę rekrutacyjną, w której użytkownik odpowiada na pytania zadawane przez cyfrowego awatara wcielającego się w rolę rekrutera.

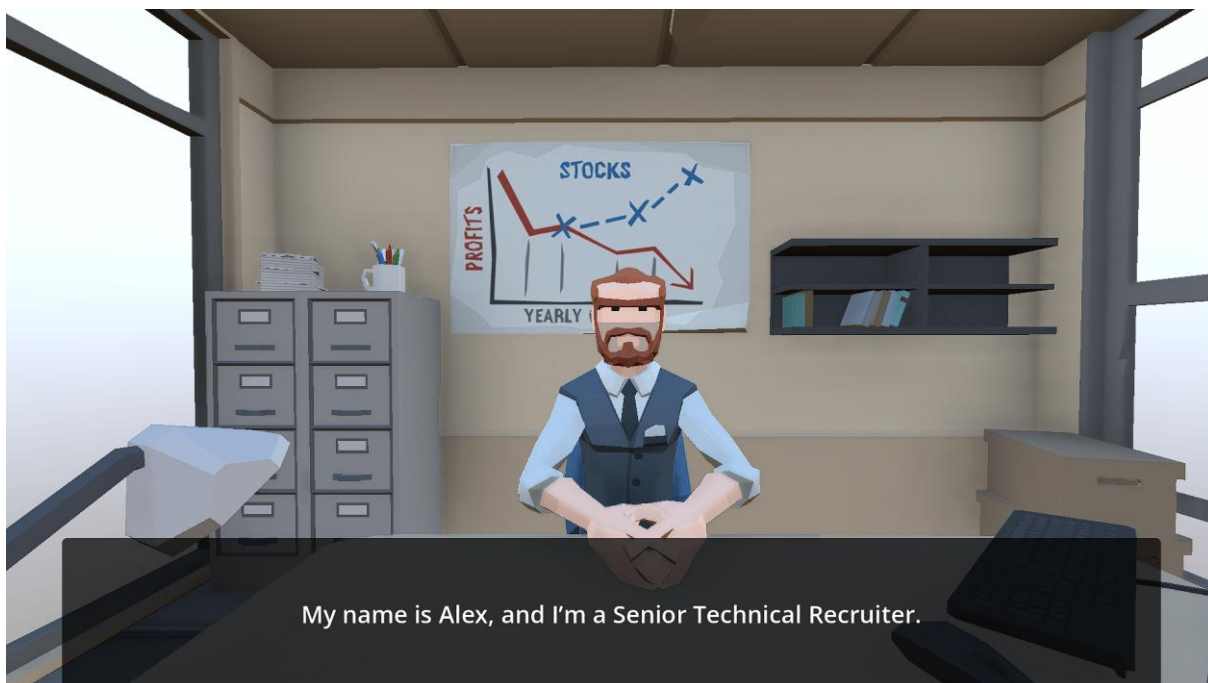
Rysunek 50 przedstawia widok sceny Office po załadowaniu – uproszczony model awatara w stylu *low poly* siedzi przy biurku w pomieszczeniu biurowym wyposażonym w szafy kartotekowe, regał z książkami oraz plakat z wykresem giełdowym na ścianie. Rysunek 51 przedstawia natomiast tę samą scenę podczas wypowiedzi awatara – na dole ekranu widoczne są napisy z treścią generowanej odpowiedzi.

Wersja *low poly* świadomie rezygnuje z animacji synchronizacji ust oraz mrugania awatara. Rezygnacja z kosztownych obliczeniowo animacji twarzy pozwala zachować płynność działania aplikacji na słabszym sprzęcie, podczas gdy kluczowe funkcjonalności symulacji – rozpoznawanie mowy, generowanie odpowiedzi przez model językowy oraz synteza audio – działają identycznie jak w wersji pełnej.

Przebieg symulacji – kolejne fazy nasłuchiwania, przetwarzania, odpowiedzi awatara oraz generowania raportu końcowego – jest identyczny jak opisany w podrozdziale 7.3.



Rysunek 50 Widok sceny Office w wersji low poly po załadowaniu – awatar oczekuje na wypowiedź użytkownika. Źródło: Opracowanie własne.



Rysunek 51 Widok sceny Office w wersji low poly podczas odpowiedzi awatara – napisy wyświetlane na dole ekranu. Źródło: Opracowanie własne.

Decyzja o wprowadzeniu sceny typu *low poly* podyktowana jest ograniczeniami sprzętowymi urządzeń mobilnych. Tabela 19 przedstawia porównanie zużycia zasobów obu wersji sceny zmierzone za pomocą wbudowanego profilera silnika Godot.

Tabela 19 Porównanie zużycia zasobów sceny w wersji pełnej i low poly. Źródło: Opracowanie własne.

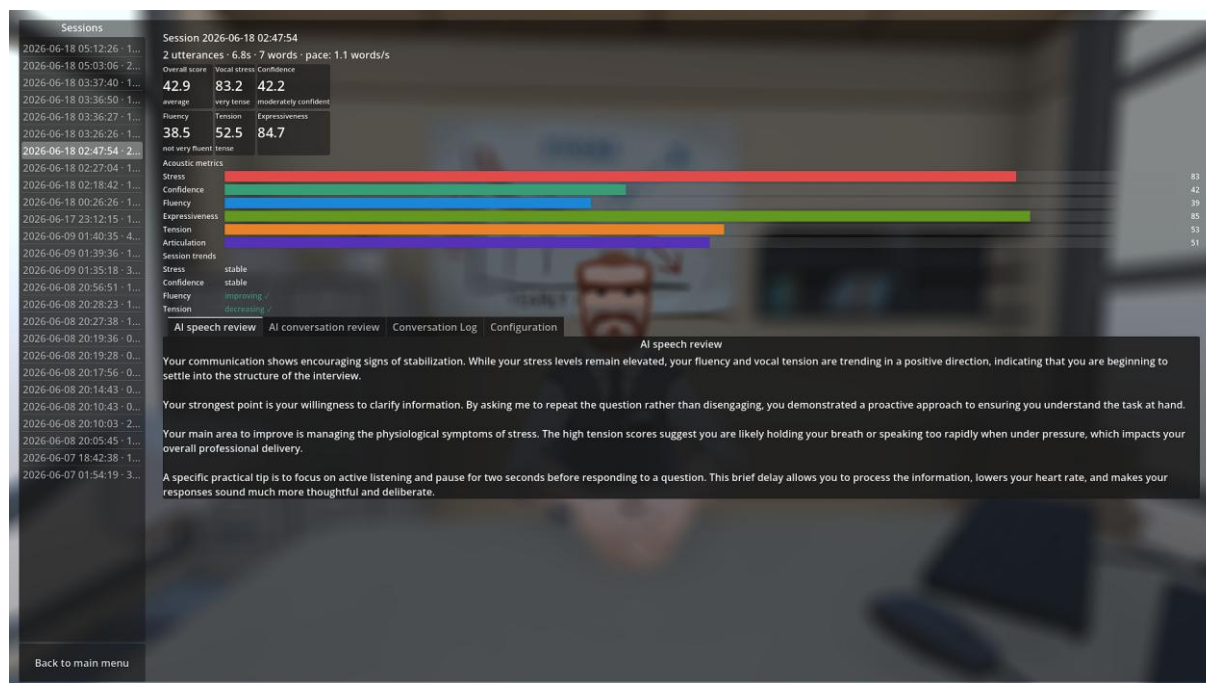
Parametr	Wersja pełna	Wersja low poly	Redukcja
Pamięć wideo (VRAM)	2,66 GiB	179,3 MiB	−93%
Pamięć tekstur	1014,5 MiB	106,3 MiB	−90%
Bufor wideo	1,15 GiB	24,1 MiB	−98%
RAM szczytowy	515,1 MiB	98,2 MiB	−81%
RAM statyczny	117,2 MiB	92,4 MiB	−21%
Liczba węzłów	645	201	−69%
FPS	175	170	~tyle samo

Redukcja zużycia pamięci wideo o 93% była kluczowym czynnikiem umożliwiającym uruchomienie aplikacji na urządzeniach mobilnych z ograniczoną pamięcią współdzieloną GPU. Rezygnacja z kosztownych obliczeniowo animacji twarzy pozwala zachować płynność działania aplikacji na słabszym sprzęcie, podczas gdy kluczowe funkcjonalności symulacji – rozpoznawanie mowy, generowanie odpowiedzi przez model językowy oraz synteza audio – działają identycznie jak w wersji pełnej.

7.5. Ekran podsumowania statystyk

Po zakończeniu symulacji i wygenerowaniu raportu przez serwer aplikacja automatycznie przełącza widok na ekran podsumowania sesji. Ekran ten stanowi główne narzędzie informacji zwrotnej dla użytkownika – prezentuje zarówno obiektywne wyniki akustycznej analizy mowy, jak i merytoryczną ocenę przeprowadzonej rozmowy.

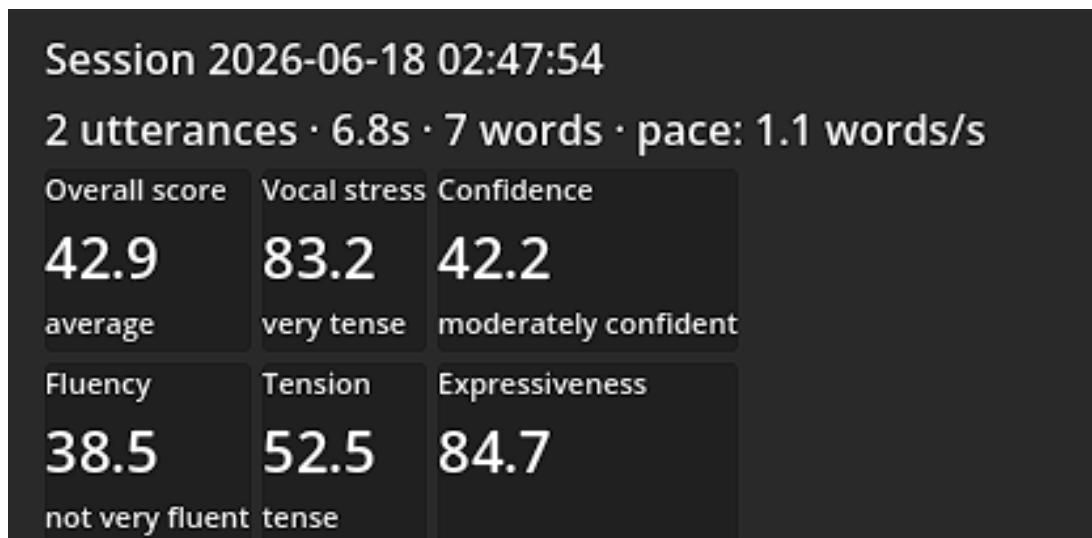
Rysunek 52 przedstawia pełny widok ekranu podsumowania przykładowej symulacji.



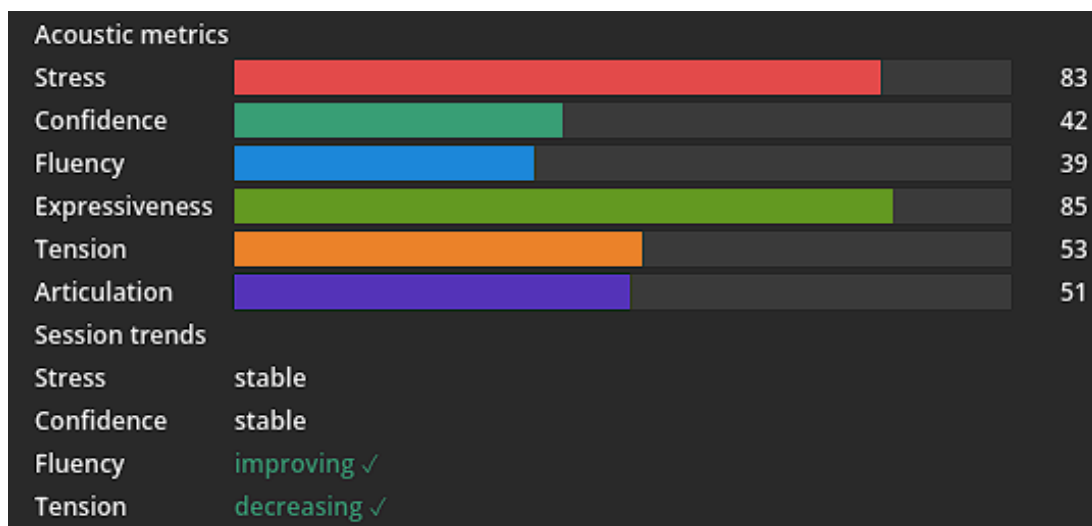
Rysunek 52 Ekran podsumowania symulacji. Źródło: Opracowanie własne.

Po lewej stronie ekranu widoczna jest lista wszystkich poprzednich sesji z datami i podstawowymi statystykami, co pozwala użytkownikowi śledzić postępy w kolejnych symulacjach i obserwować zmiany wyników w czasie.

W górnej części ekranu wyświetlane są podstawowe metadane sesji – data i godzina przeprowadzonej symulacji, liczba wypowiedzi, łączny czas trwania rozmowy oraz średnie tempo mówienia wyrażone w słowach na sekundę (Rysunek 53). Poniżej prezentowane są zbiorcze wyniki sześciu metryk akustycznych w skali 0–100 wraz z etykietami opisowymi oraz wykresem słupkowym wizualizującym ich wartości. Sekcja trendów informuje czy poszczególne metryki poprawiały się czy pogarszały w trakcie sesji (Rysunek 54).



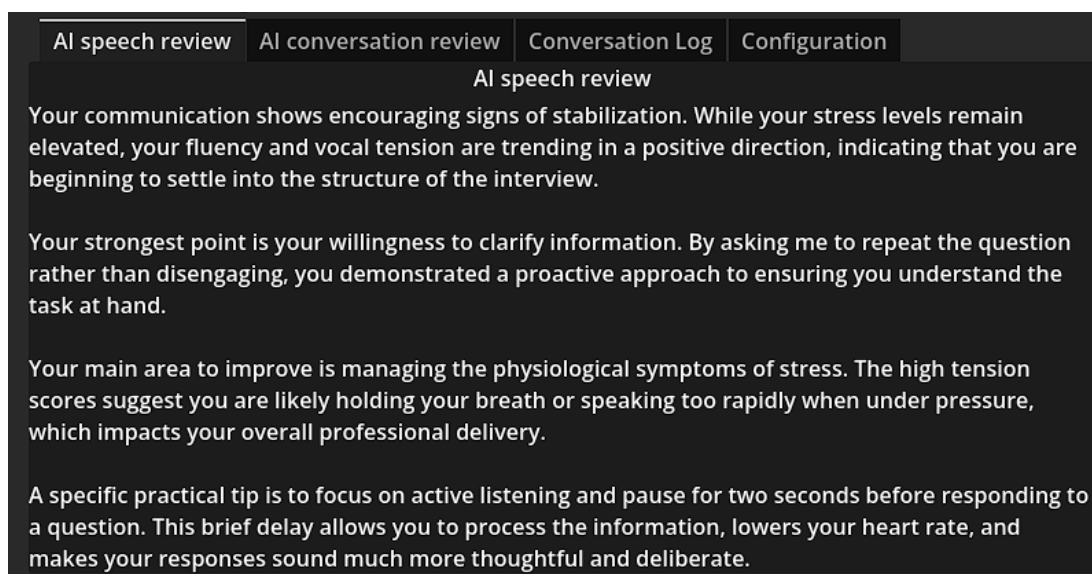
Rysunek 53 Sekcja podsumowania najważniejszych metryk. Źródło: Opracowanie własne.



Rysunek 54 Graficzna reprezentacja metryk. Źródło: Opracowanie własne.

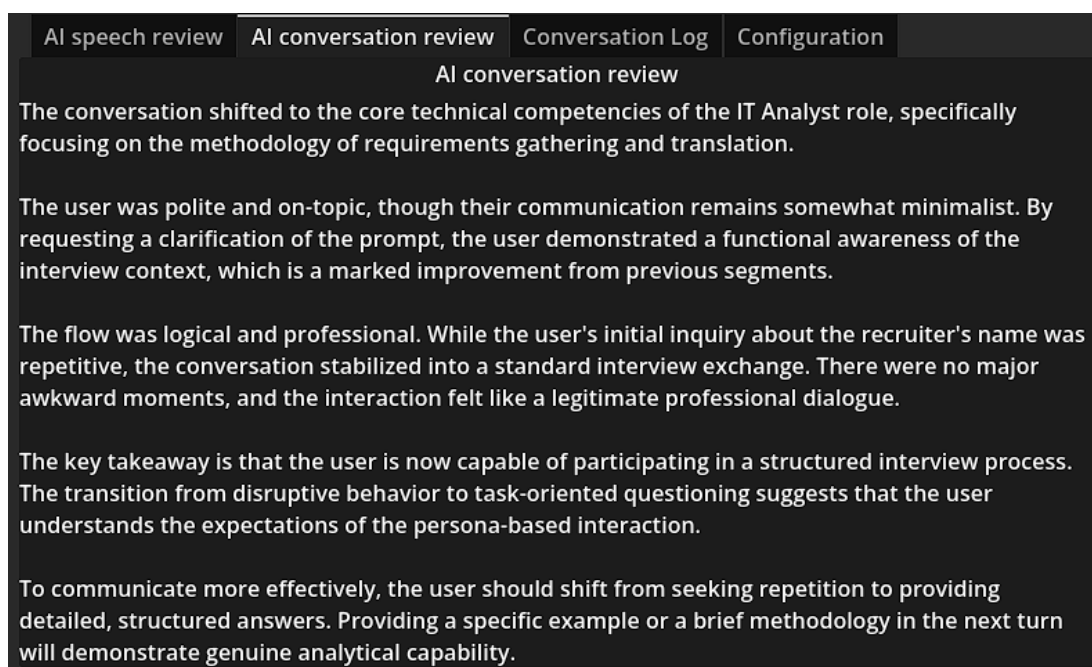
Ekran podsumowania zawiera także cztery zakładki z podsumowaniem, przedstawione na Rysunek 55 – Rysunek 58:

AI speech review (Rysunek 55) – zakładka zawiera analizę akustyczną wygenerowaną przez model językowy na podstawie liczbowych wyników sesji. Analiza obejmuje ogólną ocenę jakości mowy, wskazanie najmocniejszej strony wypowiedzi, głównego obszaru do poprawy oraz konkretną praktyczną wskazówkę dla użytkownika. Przykładowa analiza widoczna na Rysunek 55 wskazuje na podwyższony poziom stresu głosowego przy jednoczesnej poprawie płynności i pewności siebie w trakcie sesji.



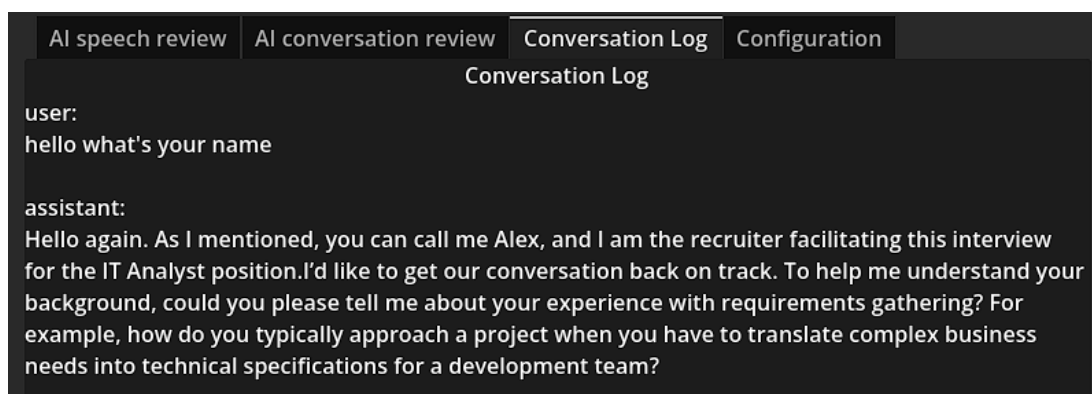
Rysunek 55 Zakładka AI speech review z akustyczną analizą mowy. Źródło: Opracowanie własne.

AI conversation review (Rysunek 56) – zakładka zawiera merytoryczne podsumowanie treści przeprowadzonej rozmowy wygenerowane przez model językowy na podstawie transkrypcji sesji. Obejmuje omówienie głównych tematów, ocenę spójności i jakości wypowiedzi użytkownika, mocne i słabe strony komunikacji oraz sugestię dotyczącą poprawy skuteczności wypowiedzi w kolejnych symulacjach.



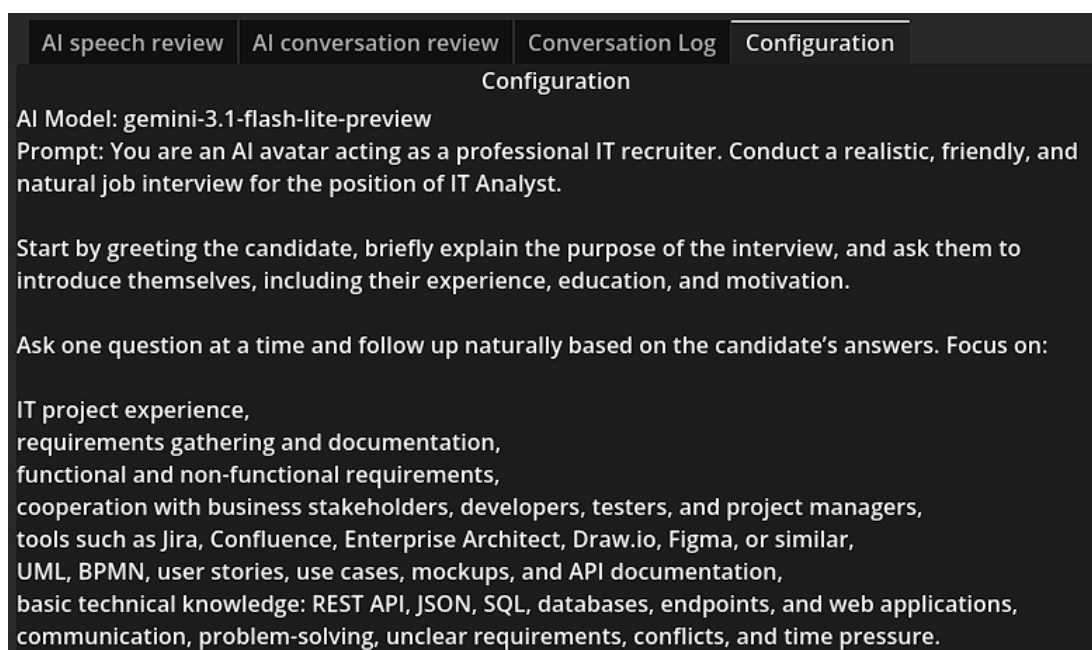
Rysunek 56 Zakładka AI conversation review z merytoryczną oceną rozmowy. Źródło: Opracowanie własne.

Conversation Log (Rysunek 57) – zakładka prezentuje pełną transkrypcję przeprowadzonej rozmowy w układzie chronologicznym z podziałem na wypowiedzi użytkownika (user) i awatara (assistant). Umożliwia to dokładną analizę przebiegu rozmowy i identyfikację konkretnych momentów wymagających poprawy.



Rysunek 57 Zakładka Conversation Log z transkrypcją rozmowy. Źródło: Opracowanie własne.

Configuration (Rysunek 58) – zakładka wyświetla parametry konfiguracyjne sesji – zastosowany model językowy, pełną treść *promptu* systemowego który definiował rolę i zachowanie awatara, informację czy awatar inicjował rozmowę oraz nazwę ewentualnie załadowanego pliku RAG. Zakładka ta umożliwia odtworzenie warunków sesji i porównanie wyników między różnymi konfiguracjami symulacji.



Rysunek 58 Zakładka Configuration z parametrami konfiguracyjnymi sesji. Źródło: Opracowanie własne.

Przedstawione przykłady użycia ilustrują kompletny przepływ interakcji użytkownika z prototypem aplikacji Stressless VR – od konfiguracji sesji, przez przeprowadzenie symulacji w wybranym scenariuszu, aż po analizę wyników w ekranie podsumowania. Elastyczność konfiguracji – możliwość wyboru sceny, modelu językowego, *promptu* systemowego oraz opcjonalnego dokumentu RAG – pozwala na dostosowanie symulacji do różnorodnych scenariuszy stresogennych bez modyfikacji kodu aplikacji. Dwupoziomowa informacja zwrotna łącząca obiektywne metryki akustyczne z merytoryczną oceną generowaną przez model językowy dostarcza użytkownikowi wartościowych wskazówek zarówno w zakresie sposobu wypowiedzi, jak i jakości merytorycznej prowadzonej rozmowy.

8. Podsumowanie i perspektywy rozwoju

Niniejszy rozdział zawiera omówienie trudności napotkanych podczas realizacji pracy, zalet i wad przyjętego rozwiązania, a także możliwych kierunków rozwoju aplikacji.

8.1. Trudności napotkane podczas realizacji pracy

Największą trudnością napotkaną podczas realizacji pracy okazała się synchronizacja dźwięku generowanego w czasie rzeczywistym z animacją mowy cyfrowego awatara.

Zgodnie z założeniami projekt „Stressless VR” miał zostać zrealizowany z wykorzystaniem Unreal Engine 5 (UE5), który jest jednym z najbardziej zaawansowanych silników na rynku. Ponadto UE5 udostępnia nowoczesny zestaw narzędzi MetaHuman, umożliwiający tworzenie realistycznych cyfrowych postaci oraz animowanie ruchów ust na podstawie sygnału dźwiękowego. Z tych powodów autorzy zdecydowali się na realizację projektu w oparciu o UE5.

Dostępne narzędzia nie pozwoliły jednak w prosty sposób wykorzystać mowy generowanej przez moduł syntezy jako bezpośredniego źródła danych do płynnej animacji ust w czasie rzeczywistym. Rozważanym rozwiązaniem problemu było strumieniowe przesyłanie dźwięku z wykorzystaniem protokołu WebSocket. Jednakże integracja tego mechanizmu z Unreal Engine 5 okazała się skomplikowana, w szczególności przy wykorzystaniu systemu wizualnego skryptowania Blueprint.

Po analizie powyższych ograniczeń autorzy zdecydowali się na przepisanie projektu z wykorzystaniem silnika Godot, który oferuje wbudowaną obsługę połączeń WebSocket. Z ich wykorzystaniem udało się przekazać cyfrowemu awatarowi dźwięk generowany w czasie rzeczywistym.

Przepisanie projektu z wykorzystaniem silnika Godot rozwiązało jeden z problemów, ale z drugiej strony wiązało się z dodatkowym nakładem pracy, który wynikał z konieczności opracowania animacji ruchów ust od podstaw. W tym celu przygotowano zestaw ułożeń ust odpowiadających poszczególnym wizerom, a ich wyświetlanie synchronizowano z odtwarzanym dźwiękiem na podstawie znaczników czasowych przekazywanych w formacie JSON. Podejście to wymagało przygotowania własnego mechanizmu animacji, ale pozwoliło spełnić przyjęte dla aplikacji założenia.

Kolejną znaczącą trudnością, która wpłynęła na pierwotne założenia projektu, było znalezienie lekkich i wydajnych modeli, które w stopniu zadowalającym rozpoznawałyby mowę, a następnie generowałyby odpowiedzi w języku polskim.

W odpowiedzi na opisany problem autorzy zdecydowali się na przejście na język angielski, dla którego dostępna jest szersza gama modeli niż w przypadku języka polskiego. Lekkie i wydajne modele zwiększyły dokładność rozpoznawania mowy oraz jakość generowanych odpowiedzi. Zastosowanie języka angielskiego można również postrzegać jako czynnik zwiększający uniwersalność prototypu i poszerzający grono jego potencjalnych użytkowników.

8.2. Wady i zalety rozwiązania

Istotną zaletę opracowanego rozwiązania stanowi jego modułowa architektura, która umożliwia szybki rozwój systemu oraz wygodną wymianę poszczególnych komponentów. Dotyczy to między innymi:

- modułu rozpoznawania mowy,
- modułu do generowania dźwięku,
- modeli wykorzystywanych do analizy wypowiedzi.

W porównaniu z dostępnymi na rynku aplikacjami zaprojektowane rozwiązanie Stressless VR charakteryzuje się wysokim stopniem uniwersalności. W ramach przygotowanego prototypu użytkownicy aplikacji mają możliwość prowadzenia swobodnego treningu różnego rodzaju sytuacji stresowych, takich jak rozmowa o pracę, egzamin ustny czy trudna rozmowa z menedżerem bądź współpracownikiem. Symulacje nie są w żadnym stopniu ograniczone do predefiniowanych scenariuszy, dzięki czemu ich przebieg może zostać dostosowany do indywidualnych potrzeb użytkownika. Aplikacja pozwala na szybką i intuicyjną konfigurację treningu umożliwiając wybór otoczenia oraz wprowadzenie promptu, który zostanie wykorzystany przez model językowy do określenia kontekstu oraz przebiegu symulacji. W celu rozszerzenia kontekstu przekazywanego modelowi LLM użytkownik może wprowadzić dodatkowy plik, który zostanie wykorzystany przez mechanizm RAG. Plik może zawierać na przykład życiorys zawodowy, materiały do egzaminu, opis stanowiska pracy lub inne informacje istotne dla przygotowywanej symulacji.

Ponadto aplikacja umożliwia elastyczne korzystanie zarówno z chmurowych, jak i lokalnych modeli językowych. W ramach prototypu opracowano integrację z chmurowym modelem Gemini, którego wykorzystanie wymaga podania klucza API. W przypadku modeli lokalnych aplikacja może natomiast łączyć się z modelami obsługiwanymi w środowisku Ollama. W tym celu wymagane jest podanie adresu serwera, na którym uruchomiono to środowisko.

W ramach prac nad prototypem nie udało się przygotować integracji z większą liczbą usług chmurowych, takich jak modele GPT firmy OpenAI czy Claude firmy Anthropic, co może być postrzegane jako ograniczenie rozwiązania. Modułowa architektura systemu umożliwia jednak dodanie ich obsługi w ramach dalszego rozwoju aplikacji.

Kolejną wartą poruszenia kwestią jest szybkość działania aplikacji, która w przypadku korzystania z modeli lokalnych jest w znacznym stopniu uzależniona od wydajności sprzętu, na którym uruchomiono serwer. Ograniczone zasoby obliczeniowe mogą prowadzić do wydłużenia czasu generowania odpowiedzi, a tym samym zmniejszenia płynności prowadzonej symulacji. Użytkownik może zatem odczuć nienaturalne przerwy w konwersacji.

Korzystanie z modeli chmurowych wiąże się natomiast z ograniczeniami narzucanymi przez dostawców usług, które mogą wpływać na jakość symulacji czy liczbę możliwych do przeprowadzenia treningów. Ograniczenia te obejmują elementy takie jak:

- limity liczby zapytań w określonym czasie,
- opłaty zależne od liczby wykorzystanych tokenów,
- maksymalną liczbę tokenów przetwarzanych w ramach pojedynczego zapytania.

Można zatem stwierdzić, że wykorzystanie AI może być postrzegane zarówno jako wada, jak i zaleta, gdyż niesie to za sobą wiele korzyści, ale i ograniczeń. Z jednej strony pozwala na wprowadzenie dużej elastyczności i uniwersalności w aplikacji. Z drugiej strony wiąże się z ograniczeniami technicznymi, a także wieloma ryzykami wynikającymi ze specyfiki działania modeli językowych, do których zaliczyć można między innymi:

- halucynacje,
- nieprzewidywalność generowanych odpowiedzi,
- odpowiedzi niezgodnych z faktami lub kontekstem symulacji,
- zależność jakości odpowiedzi od sposobu sformułowania promptu.

Mając na uwadze powyższe ryzyka i ograniczenia odpowiedzi generowane przez model nie powinny być traktowane jako całkowicie wiarygodne ani jako profesjonalna ocena zachowania użytkownika.

Ze względu na opisane we wcześniejszym podrozdziale trudności prototyp aplikacji Stressless VR został przepisany z silnika Unreal Engine 5 na silnik Godot, co z jednej strony pozwoliło to na realizację przyjętych założeń funkcjonalnych projektu, z drugiej zaś wiązało się z rezygnacją z części zaawansowanych narzędzi graficznych dostępnych w Unreal Engine 5 (w tym rozwiązań związanych z technologią MetaHuman). W konsekwencji sceny zbudowane w oparciu o prostsze modele oraz

materiały graficzne straciły nieznacznie na jakości. Mimo to opracowane rozwiązanie nie odbiega poziomem od konkurencyjnych aplikacji dostępnych obecnie na rynku. A zdaniem autorów można zaryzykować stwierdzeniem, że w ciągu dalszym przewyższa większość pod względem realizmu oraz jakości prowadzonych symulacji. Jednakże nie udało się w pełni wykorzystać potencjału technologii MetaHuman, który mógłby przede wszystkim zwiększyć płynność animacji mowy cyfrowego awatara.

Znaczącą zaletą wyróżniającą projekt jest również jego dostępność zarówno w trybie wirtualnej rzeczywistości, jak i w wersjach przeznaczonej na komputery stacjonarne czy urządzenia mobilne. Sceny opracowane zostały w dwóch kategoriach – z teksturami wysokiej jakości oraz w wersji *low poly*, przeznaczonej na urządzenia mobilne. Dzięki temu z aplikacji mogą korzystać również osoby, które nie posiadają gogli VR, co zwiększa dostępność rozwiązania oraz poszerza grupę jego potencjalnych użytkowników. Tryb VR zapewnia natomiast wyższy poziom immersji i pozwala na bardziej realistyczne odwzorowanie sytuacji stresowych.

Istotne ograniczenie może stanowić również rozmiar aplikacji, wynikający ze złożoności, a także charakteru projektu 3D. Aplikacja przechowuje bowiem obiekty i tekstury składające się na poszczególne sceny, a także model cyfrowego awatara.

8.3. Możliwe kierunki rozwoju aplikacji

Pierwszym obszarem wymagającym dalszego rozwoju jest liczba środowisk symulacyjnych dostępnych w aplikacji. Warto rozszerzyć ją o dodatkowe otoczenia odpowiadające różnym sytuacjom społecznym i zawodowym, takie jak:

- recepcja,
- aula wykładowa,
- sala konferencyjna,
- punkt obsługi klienta,
- biuro typu *open space*,
- restauracja lub kawiarnia,
- studio nagraniowe lub telewizyjne.

Poszczególne środowiska mogłyby różnić się wielkością pomieszczenia, liczbą obecnych osób, poziomem hałasu, rozmieszczeniem elementów wyposażenia oraz stopniem formalności. Pozwoliłoby to lepiej dopasować warunki symulacji do rodzaju sytuacji, do której przygotowuje się użytkownik.

Kolejnym kierunkiem rozwoju jest udoskonalenie animacji cyfrowego trenera. Oprócz ruchów ust można rozbudować mimikę twarzy, gestykulację, ruchy głowy, kontakt wzrokowy oraz sposób reagowania na wypowiedzi użytkownika. Pozwoliłoby to zwiększyć naturalność interakcji i wzmocnić poczucie obecności w symulowanej sytuacji.

Warto również umożliwić wybór trenera AI spośród kilku dostępnych postaci. Użytkownik mógłby wybrać wygląd awatara, jego głos, sposób zachowania oraz poziom formalności wypowiedzi.

Istotnym rozszerzeniem byłaby możliwość dodawania własnych materiałów do środowiska symulacyjnego. Przykładowo użytkownik przygotowujący się do wystąpienia mógłby zaimportować prezentację, która następnie zostałaby wyświetlona na ekranie znajdującym się w wirtualnym pomieszczeniu. Podobny mechanizm mógłby umożliwiać dodawanie dokumentów, notatek, pytań egzaminacyjnych lub innych materiałów związanych z ćwiczoną sytuacją.

Dalszy rozwój aplikacji może również obejmować integrację z dodatkowymi modelami rozpoznawania mowy, syntezy mowy oraz generowania odpowiedzi. Umożliwiłoby to porównywanie jakości i wydajności poszczególnych rozwiązań, a także dostosowanie systemu do możliwości sprzętowych użytkownika. W zależności od potrzeb możliwe byłoby wykorzystanie zarówno modeli lokalnych, jak i usług chmurowych.

W przyszłości warto również rozważyć opracowanie mechanizmu pozwalającego na rozszerzanie o dodatkowe sceny, modele postaci oraz moduły rozszerzających funkcjonalność aplikacji. Dzięki modułowej architekturze użytkownicy mogliby pobierać wybrane elementy bez konieczności aktualizowania całego systemu. Rozwiązanie takie ułatwiłoby rozwój aplikacji oraz pozwoliłoby dostosowywać ją do nowych obszarów zastosowań.

Bibliografia

- [1] A. Kaczmarek i P. Curyło-Sikora, „Problematyka stresu – przegląd koncepcji,” *Hygeia Public Health*, pp. 317-321, 20 10 2016.
- [2] I. Heszen, *Psychologia stresu*, Wydawnictwo Naukowe PWN, 2020.
- [3] R. M. Spielman, W. J. Jenkins, M. D. Lovett i J. Czarnota-Bojarska, *Psychologia*, Warszawa: OpenStax Poland, 2020.
- [4] R. Grieve, J. Woodley, S. E. Hunt i A. McKay, „Student fears of oral presentations and public speaking in higher education: a qualitative survey,” *Journal of Further and Higher Education*, tom 45, nr 9, p. 1281–1293, 2021.
- [5] N. Kapogianni, A. Sideraki i C.-N. Anagnostopoulos, „Using Smartwatches in Stress Management, Mental Health, and Well-Being: A Systematic Review,” *Algorithms*, Lipiec 2025.
- [6] Grand View Research, „Mental Health Apps Market Size, Share & Trends Analysis Report,” [Online]. Available: <https://www.grandviewresearch.com/industry-analysis/mental-health-apps-market-report>. [Data uzyskania dostępu: 16 Maj 2025].
- [7] L. Ioannis, D. Filos i I. Chouvarda, „Virtual reality environments for stress reduction and management: a scoping review,” *Virtual Reality*, tom 28, nr 50, 2024.
- [8] S. Meshkat, M. Edalatkhah, C. Luciano, J. Martin i e. al., „Virtual Reality and Stress Management: A Systematic Review,” *Cureus Journal of Medical Science*, 15 Lipiec 2024.
- [9] Y. L. Tan, V. Y. X. Chang, W. H. D. Ang, W. W. Ang i Y. Lau, „Virtual reality exposure therapy for social anxiety disorders: a meta-analysis and meta-regression of randomized controlled trials,” *Anxiety, Stress, & Coping*, tom 38, nr 2, pp. 141-160, 2025.
- [10] Visus VR, „Visus VR,” [Online]. Available: <https://visusvr.pl/>. [Data uzyskania dostępu: 20 Czerwiec 2026].
- [11] „Grants map EU,” [Online]. Available: <https://mapadotacji.gov.pl/projekty/1388171/?lang=en>. [Data uzyskania dostępu: 15 Czerwiec 2025].
- [12] 4helpvr, „4 Help VR,” [Online]. Available: <https://4helpvr.com/>. [Data uzyskania dostępu: 20 Styczeń 2025].
- [13] EPIC VR, „EPIC VR – Producent Oprogramowania i Gier VR/AR,” [Online]. Available: <https://epicvr.pl/pl/o-nas/>. [Data uzyskania dostępu: 20 Styczeń 2025].
- [14] iPro VR, „iPro VR,” [Online]. Available: <https://ipro-vr.com/>. [Data uzyskania dostępu: 15 Czerwiec 2025].
- [15] VR Training Sp. z o.o., „VR TRAINING,” VR TRAINING, [Online]. Available: <https://vrtraining.eu/trening-rozmow-rekrutacyjnych-tematyczna-aplikacja-szkoleniowa-vr/#>. [Data uzyskania dostępu: 15 Czerwiec 2025].
- [16] Google, „Google Play,” [Online]. Available: <https://play.google.com/store/apps/details?id=com.cambridge.vrexplorer>. [Data uzyskania dostępu: 20 Maj 2026].
- [17] VirtualSpeech, „VirtualSpeech,” [Online]. Available: <https://virtualspeech.com/>. [Data uzyskania dostępu: 25 Maj 2026].

- [18] VRSpeaking, LLC, „Ovation,” [Online]. Available: <https://www.ovationvr.com/>. [Data uzyskania dostępu: 20 Czerwiec 2026].
- [19] Meta, „Meta,” [Online]. Available: <https://www.meta.com/pl-pl/experiences/ovation/4510688418969375/>. [Data uzyskania dostępu: 26 Maj 2026].
- [20] Orai, „Orai,” [Online]. Available: <https://orai.com/>.
- [21] Knovon Consulting, „TalkReady: AI Interview Prep - Google Play,” [Online]. Available: <https://play.google.com/store/apps/details?id=com.yourcompany.talkreadystudent>. [Data uzyskania dostępu: 20 Czerwiec 2026].
- [22] European Commission, „A definition of Artificial Intelligence: main capabilities and scientific discipline,” [Online]. Available: <https://digital-strategy.ec.europa.eu/en/library/definition-artificial-intelligence-main-capabilities-and-scientific-disciplines>. [Data uzyskania dostępu: 31 Maj 2026].
- [23] Oracle, „<https://www.oracle.com/pl>,” [Online]. Available: <https://www.oracle.com/pl/artificial-intelligence/what-is-ai/>. [Data uzyskania dostępu: 31 Maj 2026].
- [24] audEERING GmbH, „About openSMILE,” audEERING GmbH, [Online]. Available: <https://audeering.github.io/opensmile/about.html>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [25] P. J. K. LADEFOGED, „A Course in Phonetics. Wyd. 6,” ISBN 978-1-4282-3126-8, Cengage Learning, 2010.
- [26] D. W. „, D. P. b. C. P. Boersma, „Phonetic Sciences,” [Online]. Available: <https://www.fon.hum.uva.nl/praat/>. [Data uzyskania dostępu: 17 Czerwiec 2026].
- [27] „, o. S. a. K. NIST/SEMATECH, „<https://www.nist.gov/>,” [Online]. Available: <https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm>. [Data uzyskania dostępu: 17 Czerwiec 2026].
- [28] Oracle, „<https://www.oracle.com/pl>,” [Online]. Available: <https://www.oracle.com/pl/artificial-intelligence/large-language-model/>. [Data uzyskania dostępu: 31 Maj 2026].
- [29] Agile Business Consortium Limited, „Agile Business,” [Online]. Available: <https://www.agilebusiness.org/resource/what-is-moscow-prioritization/>. [Data uzyskania dostępu: 20 Czerwiec 2026].
- [30] Asseco Poland S.A., „asseco,” [Online]. Available: <https://pl.asseco.com/kariera/blog/co-to-jest-rag-wstep-do-langchain-5358/>. [Data uzyskania dostępu: 31 Maj 2026].
- [31] sparxsystems, [Online]. Available: <https://sparxsystems.com/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [32] Juan Linietsky, Ariel Manzur and the Godot community (CC BY 3.0), „Dokumentacja Godot Engine 4.6 w języku polskim,” [Online]. Available: <https://docs.godotengine.org/pl/4.x/about/introduction.html>. [Data uzyskania dostępu: 16 Maj 2026].
- [33] JetBrains, [Online]. Available: <https://www.jetbrains.com/pycharm/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [34] Softonic, [Online]. Available: <https://pycharm-community-edition.softonic.pl/>. [Data uzyskania dostępu: 7 Czerwiec 2026].

- [35] „git-scm.com,” [Online]. Available: <https://git-scm.com/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [36] „git-scm.com,” [Online]. Available: <https://git-scm.com/book/pl/v2/Pierwsze-kroki-Podstawy-Git>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [37] CommitGo, Inc., „Gitea,” [Online]. Available: <https://docs.gitea.com/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [38] Oracle, „www.oracle.com,” [Online]. Available: <https://www.oracle.com/pl/cloud/cloud-native/container-registry/what-is-docker/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [39] Docker Inc., „dockerdocs,” [Online]. Available: <https://docs.docker.com/get-started/docker-overview/>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [40] Mogh Technologies Inc., „KOMODO,” [Online]. Available: <https://komo.do/docs/intro>. [Data uzyskania dostępu: 7 Czerwiec 2026].
- [41] Alpha Cephei Inc., „alphacephei,” [Online]. Available: <https://alphacephei.com/vosk/>. [Data uzyskania dostępu: 10 Czerwiec 2026].
- [42] „Python Software Foundation,” [Online]. Available: <https://pypi.org/project/vosk/>. [Data uzyskania dostępu: 10 Czerwiec 2026].
- [43] VideoSDK , „videosdk,” [Online]. Available: <https://www.videosdk.live/developer-hub/stt/vosk-speech-recognition>. [Data uzyskania dostępu: 10 Czerwiec 2026].
- [44] Coqui TTS, „Coqui TTS,” [Online]. Available: <https://coquitts.com/>. [Data uzyskania dostępu: 13 Czerwiec 2026].
- [45] „Medium,” [Online]. Available: <https://medium.com/@sudesnhm/coqui-tts-deep-dive-into-an-open-source-text-to-speech-framework-129c76a66580>. [Data uzyskania dostępu: 13 Czerwiec 2026].
- [46] Python Software Foundation, „PyPI,” [Online]. Available: <https://pypi.org/project/phonemizer/>. [Data uzyskania dostępu: 13 Czerwiec 2026].
- [47] ElevenLabs, „ElevenLabs,” [Online]. Available: <https://elevenlabs.io/pl/blog/what-is-a-phoneme>. [Data uzyskania dostępu: 13 Czerwiec 2026].
- [48] A. Silke, „Create 3D Characters,” [Online]. Available: <https://create3dcharacters.com/old-anim-week-11-lip-sync/>. [Data uzyskania dostępu: 13 Czerwiec 2026].
- [49] audEERING GmbH, „openSMILE – The Munich Open-Source Large-Scale Multimedia Feature Extractor,” [Online]. Available: <https://github.com/audEERING/opensmile>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [50] PJATK, „LLM – co to jest? Jak działają duże modele językowe?,” PJATK, [Online]. Available: <https://pja.edu.pl/llm-co-to-jest-jak-dzialaja-duze-modele-jezykowe/>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [51] S. Pichai i D. Hassabis, „Introducing Gemini: Our Largest and Most Capable AI Model,” [Online]. Available: <https://blog.google/innovation-and-ai/technology/ai/google-gemini-ai/>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [52] Google LLC, „Get started with the Gemini API,” [Online]. Available: <https://ai.google.dev/aistudio>. [Data uzyskania dostępu: 14 Czerwiec 2026].

- [53] Ollama, „Ollama,” [Online]. Available: <https://github.com/ollama/ollama>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [54] Ollama, „Quickstart,” [Online]. Available: <https://docs.ollama.com/quickstart>. [Data uzyskania dostępu: 14 Czerwiec 2026].
- [55] G. H. B. 6. P. l. y. b. y. o. w. YANG, „Epic Games Store News,” Epic Games, 25 Wrzesień 2025. [Online]. Available: <https://store.epicgames.com/news/how-battlefield-6-s-portal-lets-you-build-your-own-war>. [Data uzyskania dostępu: 2026 Czerwiec 17].

Dodatki

Dodatek A: Spis rysunków

Rysunek 1 Algorytm wyboru strategii radzenia sobie ze stresem. Źródło: Opracowanie własne.	9
Rysunek 2 Szacowana wartość rynku aplikacji zdrowia psychicznego z prognozą do roku 2030. Źródło: [6]	11
Rysunek 3 Struktura rynku aplikacji zdrowia psychicznego według typu aplikacji w 2024 roku. Źródło: [6]	12
Rysunek 4 Histogram badań według roku publikacji. Źródło: [7]	13
Rysunek 5 Symulacja resuscytacji krążeniowo-oddechowej z wykorzystaniem fizycznego fantomu oraz gogli VR. Źródło: [11]	16
Rysunek 6 Symulacja wypadku drogowego. Źródło: [10].	16
Rysunek 7 Symulacja RKO z wykorzystaniem VR oraz fizycznego fantomu. Źródło: [12]	17
Rysunek 8 Symulacja udzielania pierwszej pomocy osobie nieprzytomnej w wirtualnej rzeczywistości. Źródło: [12]	18
Rysunek 9 Szkolenie z zakresu PPOŻ. Źródło: [12]	18
Rysunek 10 Symulacja badania miejsca zdarzenia drogowego. Źródło: [13]	19
Rysunek 11 Ekran z aplikacji - wybór rozmowy. Źródło: [14]	20
Rysunek 12 Symulacja rozmowy rekrutacyjnej. Źródło: [15]	21
Rysunek 13 Ekran startowy aplikacji VR Explorer 2. Źródło: [16]	22
Rysunek 14 Widok z symulacji wystąpienia przed dużą publicznością. Źródło: [16]	22
Rysunek 15 Widok z symulacji rozmowy o pracę. Źródło: [16]	23
Rysunek 16 Przykładowa ocena wystąpienia w aplikacji Presentation Analyst App. Źródło: Opracowanie własne.	24
Rysunek 17 przedstawia kadr z wideo z wyborem symulacji. Źródło: [17]	25
Rysunek 18 Kadr z wideo przedstawiającego symulację rozmowy o pracę. Źródło: [17]	25
Rysunek 19 Kadr z wideo przedstawiającego symulację prezentacji biznesowej. Źródło: [17]	26
Rysunek 20 Kadr z wideo prezentującego działanie aplikacji Ovation - wybór otoczenia. Źródło: [19]	27
Rysunek 21 Kadr z wideo prezentującego działanie aplikacji Ovation - wprowadzanie promptu. Źródło: [19]	27
Rysunek 22 Kadr z wideo prezentującego działanie aplikacji Ovation - podsumowanie rozmowy. Źródło: [19] ..	28
Rysunek 23 Porównanie przemówień. Źródło: Opracowanie własne.	29
Rysunek 24 Raport z wystąpienia. Źródło: Opracowanie własne.	30
Rysunek 25 Konfiguracja rozmowy kwalifikacyjnej. Źródło: Opracowanie własne.	31
Rysunek 26 Podsumowanie odpowiedzi. Źródło: Opracowanie własne.	32
Rysunek 27 Wysokopoziomowa architektura aplikacyjna rozwiązania. Źródło: Opracowanie własne.	42
Rysunek 28 Diagram aktorów. Źródło: Opracowanie własne.	43
Rysunek 29 Diagram przypadków użycia. Źródło: Opracowanie własne.	44
Rysunek 30 Diagram sekwencji - proces generowania odpowiedzi. Źródło: Opracowanie własne.	54
Rysunek 31 Przykładowy diagram UML w Enterprise Architect. Źródło: Opracowanie własne.	57
Rysunek 32 Widok przykładowej sceny symulacji w edytorze Godot. Źródło: Opracowanie własne.	58
Rysunek 33 Główne okno PyCharm 2025.3.3. Źródło: Opracowanie własne.	59
Rysunek 34 Repozytorium projektu Stressless VR w systemie Gitea. Źródło: Opracowanie własne.	60
Rysunek 35 Architektura Dockera. Źródło: [39]	61
Rysunek 36 Schemat działania biblioteki Vosk. Źródło: [43]	63
Rysunek 37 Architektura Coqui TTS. Źródło: [45]	63
Rysunek 38 Przykładowe wizemy. Źródło: [48]	64
Rysunek 39 Edytor poziomów Spatial Editor trybu Portal w grze Battlefield 6, oparty na silniku Godot. Źródło: [55]	67
Rysunek 40 Struktura węzłów sceny głównej w wersji wysokiej szczegółowości (mainScene). Źródło: Opracowanie własne.	80
Rysunek 41 Struktura węzłów sceny głównej w wersji uproszczonej przeznaczonej na urządzenia mobilne (main_mobile). Źródło: Opracowanie własne.	80
Rysunek 42 Ekran głównego menu aplikacji Stressless VR. Źródło: Opracowanie własne.	83
Rysunek 43 Ekran konfiguracji sesji z wybranym modelem Ollama (Local). Źródło: Opracowanie własne.	84
Rysunek 44 Ekran konfiguracji sesji z wybranym modelem Gemini (Google API). Źródło: Opracowanie własne.	85

<i>Rysunek 45 Widok sceny University po załadowaniu – awatar oczekuje na wypowiedź użytkownika. Źródło: Opracowanie własne.</i>	86
<i>Rysunek 46 Faza nasłuchiwania – ikona mikrofonu sygnalizuje aktywne nagrywanie wypowiedzi użytkownika. Źródło: Opracowanie własne.</i>	86
<i>Rysunek 47 Faza przetwarzania – dymek nad awatarem sygnalizuje oczekiwanie na odpowiedź modelu językowego. Źródło: Opracowanie własne.</i>	87
<i>Rysunek 48 Faza odpowiedzi – awatar mówi z synchronizacją ust, napisy wyświetlane są na dole ekranu. Źródło: Opracowanie własne.</i>	87
<i>Rysunek 49 Ekran oczekiwania na wygenerowanie raportu po zakończeniu symulacji. Źródło: Opracowanie własne.</i>	88
<i>Rysunek 50 Widok sceny Office w wersji low poly po załadowaniu – awatar oczekuje na wypowiedź użytkownika. Źródło: Opracowanie własne.</i>	89
<i>Rysunek 51 Widok sceny Office w wersji low poly podczas odpowiedzi awatara – napisy wyświetlane na dole ekranu. Źródło: Opracowanie własne.</i>	89
<i>Rysunek 52 Ekran podsumowania symulacji. Źródło: Opracowanie własne.</i>	90
<i>Rysunek 53 Sekcja podsumowania najważniejszych metryk. Źródło: Opracowanie własne.</i>	91
<i>Rysunek 54 Graficzna reprezentacja metryk. Źródło: Opracowanie własne.</i>	91
<i>Rysunek 55 Zakładka AI speech review z akustyczną analizą mowy. Źródło: Opracowanie własne.</i>	92
<i>Rysunek 56 Zakładka AI conversation review z merytoryczną oceną rozmowy. Źródło: Opracowanie własne.</i>	92
<i>Rysunek 57 Zakładka Conversation Log z transkrypcją rozmowy. Źródło: Opracowanie własne.</i>	93
<i>Rysunek 58 Zakładka Configuration z parametrami konfiguracyjnymi sesji. Źródło: Opracowanie własne.</i>	93

Dodatek B: Spis tabel

<i>Tabela 1 Słownik pojęć. Źródło: Opracowanie własne.....</i>	<i>34</i>
<i>Tabela 2 Wymagania funkcjonalne aplikacji Stressless VR. Źródło: Opracowanie własne.</i>	<i>36</i>
<i>Tabela 3 Wymagania нефunkcjonalne aplikacji Stressless VR. Źródło: Opracowanie własne.....</i>	<i>39</i>
<i>Tabela 4 Scenariusz przypadku użycia PU.KONFIG.01. Źródło: Opracowanie własne.....</i>	<i>44</i>
<i>Tabela 5 Matryca walidacji do PU.KONFIG.01. Źródło: Opracowanie własne.....</i>	<i>46</i>
<i>Tabela 6 Scenariusz przypadku użycia PU.SYMULACJA.01. Źródło: Opracowanie własne.</i>	<i>48</i>
<i>Tabela 7 Scenariusz przypadku użycia PU.SYMULACJA.02. Źródło: Opracowanie własne.</i>	<i>48</i>
<i>Tabela 8 Scenariusz przypadku użycia PU.SYMULACJA.03. Źródło: Opracowanie własne.</i>	<i>49</i>
<i>Tabela 9 Scenariusz przypadku użycia PU.RAPORT.01. Źródło: Opracowanie własne.</i>	<i>50</i>
<i>Tabela 10 Scenariusz przypadku użycia PU.RAPORT.02. Źródło: Opracowanie własne.</i>	<i>50</i>
<i>Tabela 11 Scenariusz przypadku użycia PU.RAPORT.03. Źródło: Opracowanie własne.</i>	<i>51</i>
<i>Tabela 12 Scenariusz przypadku użycia PU.RAPORT.04. Źródło: Opracowanie własne.</i>	<i>52</i>
<i>Tabela 13 Scenariusz przypadku użycia PU.RAPORT.05. Źródło: Opracowanie własne.</i>	<i>52</i>
<i>Tabela 14 Scenariusz przypadku użycia PU.RAPORT.06. Źródło: Opracowanie własne.</i>	<i>53</i>
<i>Tabela 15 Integracja z Google Gemini. Źródło: Opracowanie własne.....</i>	<i>55</i>
<i>Tabela 16 Typy poleceń JSON obsługiwanych przez serwer w kanale WS_IN. Źródło: Opracowanie własne.</i>	<i>68</i>
<i>Tabela 17 Odwzorowanie fonemów IPA na wizmy wykorzystywane w animacji ust awatara. Źródło: Opracowanie własne.....</i>	<i>70</i>
<i>Tabela 18 Metryki jakości mowy wyznaczone przez moduł analizy akustycznej. Źródło: Opracowanie własne... </i>	<i>72</i>
<i>Tabela 19 Porównanie zużycia zasobów sceny w wersji pełnej i low poly. Źródło: Opracowanie własne.</i>	<i>90</i>

Dodatek C: Spis listingów

<i>Listing 1 Uruchomienie serwera WebSocket nasłuchującego na pojedynczym porcie. Źródło: Opracowanie własne.</i>	68
<i>Listing 2 Obsługa wiadomości przychodzących – rozróżnienie audio i poleceń tekstowych. Źródło: Opracowanie własne.</i>	68
<i>Listing 3 Filtracja ciszy na podstawie RMS oraz rozpoznawanie mowy z wykorzystaniem biblioteki Vosk. Źródło: Opracowanie własne.</i>	69
<i>Listing 4 Synteza mowy z wykorzystaniem modelu VITS oraz przepróbkowanie sygnału do częstotliwości wymaganej przez klienta. Źródło: Opracowanie własne.</i>	70
<i>Listing 5 Struktura komunikatu JSON z osią czasu fonemów przesyłanego do klienta przed strumieniowaniem audio. Źródło: Opracowanie własne.</i>	71
<i>Listing 6 Równoległa ekstrakcja cech akustycznych zestawami eGeMAPSv02 i IS10. Źródło: Opracowanie własne.</i>	72
<i>Listing 7 Obliczanie ważonej oceny ogólnej jakości mowy. Źródło: Opracowanie własne.</i>	72
<i>Listing 8 Wyznaczanie trendu metryki jako różnicy średnich z dwóch połów sesji. Źródło: Opracowanie własne.</i>	73
<i>Listing 9 Wczytywanie dokumentu i podział na fragmenty z zakładką. Źródło: Opracowanie własne.</i>	73
<i>Listing 10 Wyszukiwanie semantycznie zbliżonych fragmentów dokumentu względem zapytania użytkownika. Źródło: Opracowanie własne.</i>	74
<i>Listing 11 Struktura zapytania do modelu językowego wzbogaconego o kontekst z modułu RAG. Źródło: Opracowanie własne.</i>	74
<i>Listing 12 Funkcja fasady delegująca zapytanie do wybranego backendu modelu językowego. Źródło: Opracowanie własne.</i>	75
<i>Listing 13 Inicjalizacja modelu Gemini z kluczem API i promptem systemowym przekazany przez klienta. Źródło: Opracowanie własne.</i>	75
<i>Listing 14 Weryfikacja nawiązania połączenia WebSocket w singletons ConnectionConfig. Źródło: Opracowanie własne.</i>	76
<i>Listing 15 Obsługa przychodzących komunikatów JSON z serwera w metodzie handle_text(). Źródło: Opracowanie własne.</i>	77
<i>Listing 16 Przygotowanie kolejki porcji audio PCM do wysłania przez WebSocket. Źródło: Opracowanie własne.</i>	77
<i>Listing 17 Przepompowywanie próbek PCM z bufora do generatora audio silnika Godot. Źródło: Opracowanie własne.</i>	78
<i>Listing 18 Wybór aktywnej animacji fonemu na podstawie bieżącego czasu odtwarzania osi czasu. Źródło: Opracowanie własne.</i>	79
<i>Listing 19 Wczytywanie osi czasu fonemów z korekcją minimalnego czasu trwania ciszy. Źródło: Opracowanie własne.</i>	79
<i>Listing 20 Inicjalizacja sceny z próbą uruchomienia interfejsu OpenXR. Źródło: Opracowanie własne.</i>	81
<i>Listing 21 Zmiana ikony i animacji wskaźnika stanu w metodzie set_status(). Źródło: Opracowanie własne.</i>	82