1. Cel stworzenia projektu

Głównym celem stworzenia projektu jest sterowanie komputerem za pomocą mowy. Program wykonuje odpowiednio przekazane mu głosowo polecenia. Słowo wywołujące po uruchomieniu programu to “Computer”, po którym należy wypowiedzieć skonfigurowaną przez nas komendę. Program następnie wykona zadanie oraz do momentu wyłączenia będzie wyczekiwał następnej instancji słowa wywołującego.

2. Zakres pracy

11.03.2024 (pierwszy commit) - 19.06.2024

3. Wykorzystane technologie

- Python 3.12
- Django-ninja API - framework API dla Django, umożliwiający tworzenie API opartych na modelach Django
- Flet - framework do tworzenia aplikacji Flutter w Pythonie
- PyAudio - biblioteka umożliwiająca komunikacje języka Python z API PortAudio19, wielo-platformowej biblioteki I/O audio
- DoomEmacs - potężne środowisko programistyczne i edytor tekstu
- Discord Api - aplikacja użyta do zdalnego sterowania projektem

4. Struktura projektu

Cały program składa się z głównego folderu zawierającego plik app_run.py, służącego do uruchomienia całej aplikacji.

Następnie podfolder api zawiera konfigurację Django API oraz model documents służący do zapisywania komend skonfigurowanych przez użytkownika w bazie danych.

Podfolder app_run_config zawiera konfigurację oraz uruchamia potrzebne zależności, API oraz interfejs graficzny.

Podfolder gui zawiera konfigurację oraz główny kod interfejsu graficznego.

Podfolder discord_bot zawiera konfigurację oraz główny kod bota na platformie Discord komunikającego się z API.

Podfolder audio_bot zawiera konfigurację oraz główny kod bota rozpoznającego dynamicznie mowe.

```
.
├── api/
│   ├── api/
│   └── documents/
├── app_run_config/
│   └── audio_config/
├── discord_bot/
│   └── bot_config/
├── audio_bot/
│   └── audio_config/
├── gui/
│   └── gui_config/
└── app_run.py
```

5. Instrukcja obsługi

Starting app development
Copy the .env.example file:

```
cp .env.example .env
```

Modify the environment variables to suit your requirements. Launching services Install app requirements

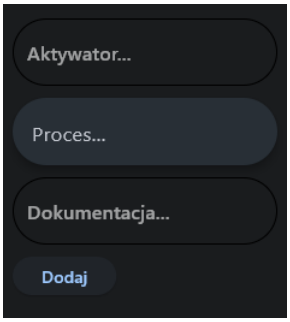
```
python app_run.py --install
```

run app

```
python app_run.py
```

6. Dane wejściowe do działania projektu

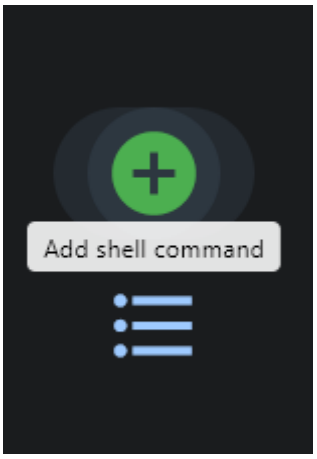
Jako dane wejściowe projekt potrzebuje nazwy, bądź frazy którą będziemy używać do wywołania komendy, ścieżki lub komendy shell/cmd (w zależności od systemu użytkownika) oraz opcjonalnej dokumentacji na czym polega tworzona komenda.



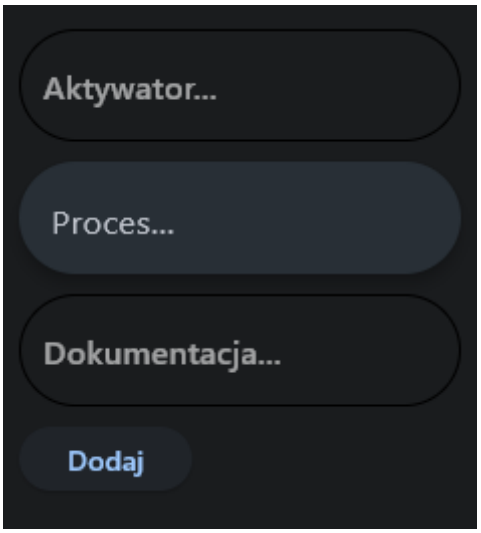
7. Przykłady działania

7.1. Gui

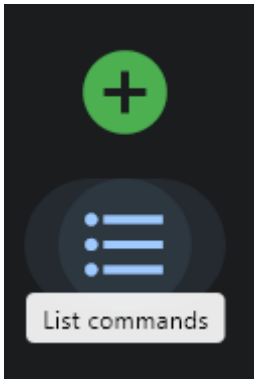
Na początku, zaraz po uruchomieniu programu nie będziemy w stanie od razu wywoływać głosem żadnej komendy, gdyż baza danych jest jeszcze pusta. Aby rozpocząć użytkowanie programu należy dodać komendę, do czego służy pierwszy główny przycisk “Dodaj komendę”.

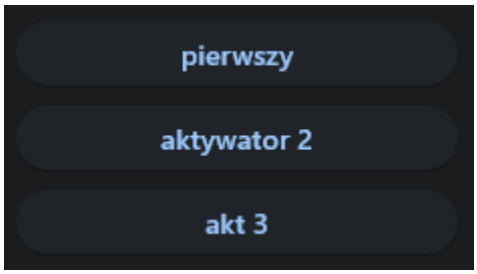


Następnie musimy wpisać odpowiednio opisane już wcześniej interesujące nas parametry oraz kliknąć przycisk “Dodaj”



Jeśli jesteśmy tym zainteresowani, możemy ujrzeć listę wszystkich komend, które zostały dodane do bazy danych i możemy wykonywać. Jest to możliwe za pomocą drugiego głównego przycisku “Wyświetl komendy”.





Następnie wystarczy wypowiedzieć słowa wywołujące oraz komendę, którą wcześniej skonfigurowaliśmy.

7.2. audio_bot

Mikrofon systemowy analizuje mowe i po krótkiej ciszy, wyszukuje podaną frazę w bazie danych, jeśli ją znajdzie, wykona komende przypisaną do tej frazy.

7.3. discord_bot

Użytkownik wpisuje w okienku chatu discorda frazę, po czym jest wyszukiwana w bazie. Przypisana komenda do tej frazy się wykonuje.

7.4. DoomEmacs

Użytkownik jest w stanie do bazy dodać komendy elisp, które potrafią zautomatyzować proces programowania, przy uruchomionym emacs serverze.

8. Podsumowanie

Projekt oferuje możliwość zautomatyzowania codziennych czynności i interakcji z zewnętrznymi urządzeniami.

Wiele dostępnych funkcjonalności jest dopiero bazą, którą można w przyszłości rozwinąć.

8.1. Podział pracy w projekcie

- Rafał Bazan - Team leader, automatyzacja pracy, implementacja Django-ninja API, działalności: PyAudio, discord-bot, DoomEmacs (backend)
- Jakub Piasek - design Graficzny interfejsu Flet, funkcjonalność i implementacja gui, działalności discord-bot (frontend)

Repozytorium na GitHub