Zespołowe przedsięwzięcie inżynierskie
3IS P3
Państwowa Wyższa Szkoła Zawodowa w Nowym Sączu

Tytuł projektu: Aplikacja usprawniająca komunikację osób niepełnosprawnych za pomocą klawiatury wirtualnej wykorzystując biblioteki OpenCV

Tytuł roboczy: KlawiaturaCV

Skład grupy: Adrian Święs, Patryk Matusik, Adam Szczepański

Prowadzący: mgr inż. Nikodem Bulanda

1.Cel

Osoba korzystająca z aplikacji będzie w stanie samodzielnie za pomocą ruchu oczu oraz powiek w bardzo prosty i łatwy sposób komunikować się. Aplikacja skierowana jest w stronę osób niepełnosprawnych z paraliżem czterokończynowym, lecz nie tylko dla nich. Z aplikacji będzie mogła skorzystać każda osoba. Obecnie na rynku istnieje niewiele aplikacji, które wspomagają pracę niepełnosprawnych. Termin wykonania projektu szacuje się na maj bieżącego roku. Podczas tworzenia, aby usprawnić pracę w zespole korzystaliśmy z platformy Discord, a efekty naszej pracy w postaci kodu źródłowego zamieszczaliśmy na platformie GitHub. Termin ukończenia projektu szacowany był na 10.05.2022 r..

2.Zakres

2.1 Analiza wymagań

Zastosowanie jednej z bibliotek Pythona do klasyfikacji twarzy za pomocą kamery internetowej a na koniec dodanie punktów orientacyjnych twarzy, aby śledzić oczy użytkownika aplikacji w celu określenia świadomości. Pierwszym krokiem będzie wytrenowanie modelu klasyfikacji za pomocą biblioteki Pythona. Kolejnym krokiem będzie użycie tej biblioteki w transmisjach wideo na żywo. Utworzenie środowiska wirtualnego aby projekt miał własne zależności i nie kolidował z żadnym innym projektem. Następnie pobranie wymaganych bibliotek i pakietów. Będzie to służyło do identyfikacji twarzy w kamerce internetowej. Po uruchomieniu programu będziemy widzieć trzy różne okna zawierające odpowiednio: klawiaturę, białą tablicę i obraz z kamery internetowej . Klawiatura jest podzielona na dwie części i przed naciśnięciem każdej litery będziemy musieli wybrać część. Lewa strona będzie zawierała litery: Q, W, E, R, T, A, S, D, F, G, Z, X, C, V. Prawa strona będzie zawierała litery: Y, U, I, O, P, H, J, K, L, V, B, N, M i spacja. Lewa oraz prawa strona klawiatury będzie miała trzy klawisze, które będą bezpośrednio odpowiadać za powrót do wyboru strony, automatyczne przeniesienie do witryny Google oraz znak “ . “ . Kamerka internetowa będzie kontrolowała pozycję oczu, więc jeśli będziemy chcieli wybrać lewą stronę, trzeba spojrzeć w lewo przez około sekundę lub w przeciwną stronę jeśli będziemy chcieli wybrać prawą stronę. Załóżmy, że wybierzemy prawą stronę. Zobaczymy litery znajdujące się po prawej stronie. Aby nacisnąć klawisz, będziemy musieli poczekać, aż klawisz się zaświeci, a następnie mrugać lewym okiem . Trzymać go zamkniętego, dopóki nie usłyszymy dźwięku. Ten dźwięk będzie oznaczał, że klawisz został naciśnięty. Za każdym razem, gdy naciśniemy klawisz, musimy powtórzyć krok 1 i 2. Każde naciśnięcie klawisza będzie dodawane do tablicy. Prawym okiem poprzez mrugnięcie przez dłuższy czas będzie usuwał ostatni znak z tablicy(działanie identyczne jak po naciśnięciU na klawiaturze klawisza “BACKSPACE”).

2.2 Wymagania funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne

- Określenie możliwości jakie oferuje ludzki wzrok w kontekście typowych scenariuszy interakcji człowieka. Wybór elementów wzroku przydatnych w komunikacji.
- Zdefiniowanie atrybutów elementów wzroku oraz odpowiadających im cech charakterystycznych na obrazie lub wideo, pozwalających na rozpoznawanie gestów oczu.
- Zaproponowanie struktury i elementów składowych systemu automatycznego rozpoznawania gałek ocznych.
- Opracowanie metod wyodrębniania z obrazu twarzy cech charakterystycznych, odpowiadających atrybutom rozpoznawanych elementów.
- Opracowanie metod rozpoznawania gestów wzroku wykorzystujących wyodrębnione cechy.
- Usystematyzowanie czynników wpływających na skuteczność rozpoznawania oraz opracowanie metody adaptacji systemu do człowieka oraz zmieniających się warunków otoczenia.

Wymagania niefunkcjonalne

- Jak najlepsze zoptymalizowanie kodu programu.
- Zadbanie o responsywność projektu.
- Łatwość użycia.
-

Gwarancja możliwości korzystania z aplikacji.

•
Odzwierciedlenie oczekiwań w aplikacji.

•
Łatwość zmiany lokalizacji aplikacji, środowiska, eksportu danych.

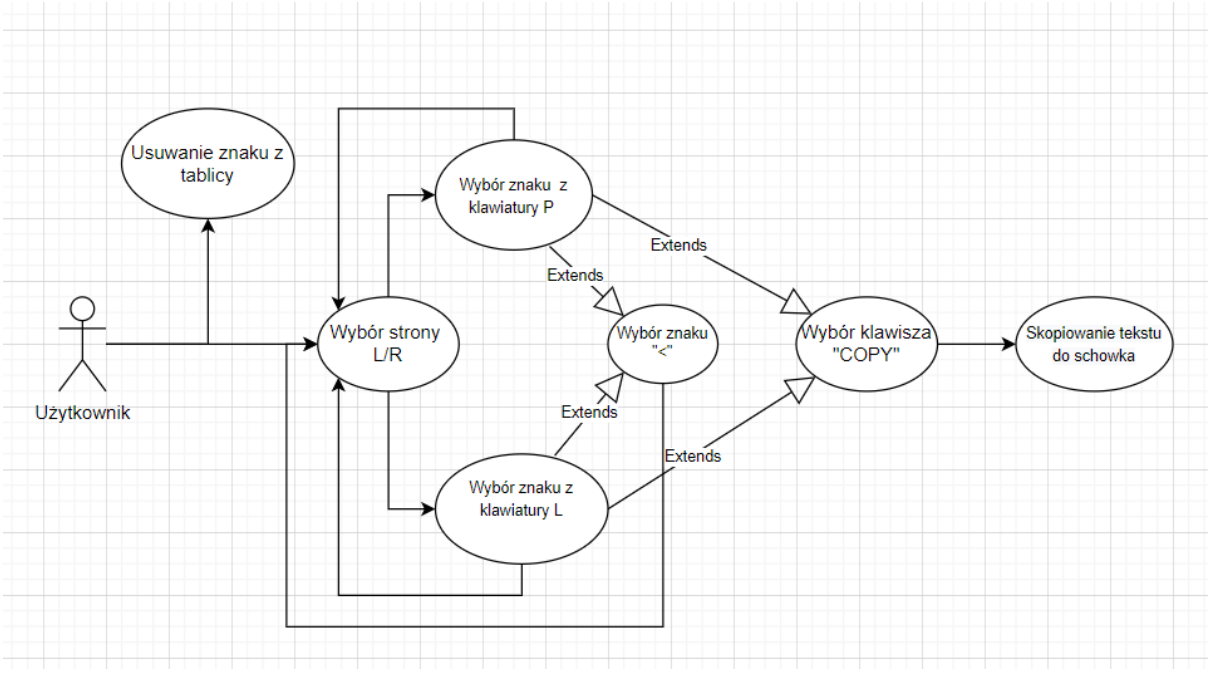
•
Założenia i wymagania jakie musi spełniać środowisko aplikacji aby użytkownicy mogli

z niej korzystać przy założonym obciążeniu w kolejnych okresach.

•
Dokładność aplikacji.
•
Skierowanie aplikacji do grupy osób niepełnosprawnych

•
Aplikacja obsługuje wszystkie rozmiary oczu użytkowników.

2.3 Diagram przypadków użycia.



2.4 Dobór technologii

- Python- język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. Python wspiera różne paradygmaty programowania: obiektowy, imperatywny oraz w mniejszym stopniu funkcyjny.
Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią, będąc w tym podobnym do języków Perl, Ruby, Scheme czy Tcl. Podobnie jak inne języki dynamiczne jest często używany jako język skryptowy. Interpretery Pythona są dostępne na wiele systemów operacyjnych.

- OpenCV- wieloplatformowa biblioteka funkcji wykorzystywanych podczas obróbki obrazu, oparta na otwartym kodzie i zapoczątkowana przez Intela. Autorzy jej skupiają się na przetwarzaniu obrazu w czasie rzeczywistym.

- GitHub- hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji Git. Stworzony został przy wykorzystaniu frameworka Ruby on Rails i języka Erlang. Serwis działa od kwietnia 2008 roku. GitHub udostępnia darmowy hosting programów open source i prywatnych repozytoriów (część funkcji w ramach prywatnych repozytoriów jest płatna).

NumPy- to biblioteka Pythona używana do pracy z tablicami. Posiada również funkcje do pracy w dziedzinie algebry liniowej, transformaty Fouriera i macierzy. NumPy został stworzony w 2005 roku przez Travisa Oliphanta. Jest to projekt open source i możesz z niego swobodnie korzystać. NumPy to skrót od Numerical Python.

Pyglet **-** jest łatwą w użyciu, ale potężną biblioteką do tworzenia bogatych wizualnie aplikacji GUI, takich jak gry, multimedia itp. w systemach Windows, Mac OS i Linux. Ta biblioteka jest tworzona wyłącznie w Pythonie i obsługuje wiele funkcji, takich jak okna, obsługa zdarzeń interfejsu użytkownika, joysticki, grafika OpenGL, ładowanie obrazów i filmów oraz odtwarzanie dźwięków i muzyki. pyglet jest dostarczany na licencji open source BSD, co pozwala na używanie go

zarówno w komercyjnych, jak i innych projektach open source z bardzo małymi ograniczeniami.

Time- moduł Pythona,który udostępnia wiele sposobów przedstawiania czasu w kodzie, takich jak obiekty, liczby i ciągi znaków. Zapewnia również funkcje inne niż reprezentowanie czasu, takie jak oczekiwanie podczas wykonywania kodu i mierzenie wydajności kodu.

Tkinter – biblioteka Pythona umożliwiająca tworzenie interfejsu graficznego (GUI). Tkinter jest dołączony do standardowych instalacji Pythona w systemach Linux, Microsoft Windows i Mac OS X. Nazwa Tkinter pochodzi od interfejsu Tk. Biblioteka ta została napisana przez Fredrika Lundha. Tkinter to darmowe oprogramowanie wydane na licencji Pythona.

3.Scenariusze

Scenariusz nr:	1
Tytuł:	Test aplikacji
Aktor:	Użytkownik
Warunek:	Włączenie aplikacji
Opis/Przebieg:	1.Użytkownik uruchamia kamerę. 2.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 3. Użytkownik poprzez zamknięcie powiek wybiera literę “A”. 4. Użytkownik ponownie wybiera klawiaturę z przedziału “LEFT”. 5. Po poprawnym zamknięciu powiek użytkownik wybrał literę “B”. 6. Użytkownik postanowił usunąć ciąg znaków więc wybrał klawiaturę z przedziału “RIGHT”. 7. Użytkownik usunął tekst używając klawisza “del”.

	8. Użytkownik kończy pracę w programie
Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

Scenariusz nr:	2
Tytuł:	Pisanie sekwencji znaków
Aktor:	Użytkownik
Warunek:	1. Włączenie aplikacji 2. Uruchomienie kamery
Opis/Przebieg:	1.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 2. Użytkownik poprzez zamknięcie powiek wybiera literę “H” 3. Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 4. Użytkownik poprzez zamknięcie powiek wybiera literę “E” 5.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 6.Użytkownik poprzez zamknięcie powiek wybiera literę “L” 7.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 8.Użytkownik poprzez zamknięcie powiek wybiera literę “L” 9.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 10. Użytkownik powraca do wyboru stron klawiatury wirtualnej “<”. 11.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 12.Użytkownik poprzez zamknięcie powiek wybiera literę “O” 13. Użytkownik kończy pracę w programie

Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

Scenariusz nr:	3
Tytuł:	Modyfikowanie tekstu
Aktor:	Użytkownik
Warunek:	1. Włączenie aplikacji 2. Uruchomienie kamery
Opis/Przebieg:	1.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 2. Użytkownik poprzez zamknięcie powiek wybiera literę “H” 3. Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 4. Użytkownik poprzez zamknięcie powiek wybiera literę “E” 5.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 6.Użytkownik poprzez zamknięcie powiek wybiera literę “L” 7.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 8.Użytkownik poprzez zamknięcie powiek wybiera literę “P” 9.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 10. Użytkownik usunął znak pisarski używając klawisza “del”. 11.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 12. Użytkownik usunął znak pisarski używając klawisza “del”. 11.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT”

	12.Użytkownik poprzez zamknięcie powiek wybiera literę “Y” 13. Użytkownik kończy pracę w programie
Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

Scenariusz nr:	4
Tytuł:	Pisanie sekwencji znaków
Aktor:	Użytkownik
Warunek:	1. Włączenie aplikacji 2. Uruchomienie kamery
Opis/Przebieg:	1.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 2. Użytkownik poprzez zamknięcie powiek wybiera literę “C” 3. Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 4. Użytkownik poprzez zamknięcie powiek wybiera literę “A” 5.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 6.Użytkownik powraca do wyboru stron klawiatury wirtualnej “<”. 7.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 8.Użytkownik poprzez zamknięcie powiek wybiera literę “P” 9.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 10. Użytkownik usunął znak pisarski używając klawisza “del”. 11.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 12.Użytkownik poprzez zamknięcie powiek wybiera literę “R”

	13. Użytkownik kończy pracę w programie
Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

Scenariusz nr:	5
Tytuł:	Pisanie sekwencji znaków
Aktor:	Użytkownik
Warunek:	1. Włączenie aplikacji 2. Uruchomienie kamery
Opis/Przebieg:	1.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 2. Użytkownik poprzez zamknięcie powiek wybiera literę “C” 3. Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 4. Użytkownik poprzez zamknięcie powiek wybiera literę “A” 5.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 6.Użytkownik powraca do wyboru stron klawiatury wirtualnej “<”. 7.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 8.Użytkownik poprzez zamknięcie powiek wybiera literę “T” 9.Użytkownik kończy pracę w programie
Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

Scenariusz nr:	6

Tytuł:	Usuwanie tekstu po błędzie
Aktor:	Użytkownik
Warunek:	1. Włączenie aplikacji 2. Uruchomienie kamery
Opis/Przebieg:	1.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 2. Użytkownik poprzez zamknięcie powiek wybiera literę “C” 3. Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 4. Użytkownik poprzez zamknięcie powiek wybiera literę “O” 5.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 6.Użytkownik poprzez zamknięcie powiek wybiera literę “O” 7.Wybór strony przedziału liter z klawiatury wirtualnej “RIGHT” 8.Użytkownik usunął znak pisarski używając klawisza “del”. 9.Wybór strony przedziału liter z klawiatury wirtualnej “LEFT” 10.Użytkownik poprzez zamknięcie powiek wybiera literę “W” 11.Użytkownik kończy pracę w programie
Zakończenie:	Poprawne napisanie sekwencji znaków
Zakończenie alternatywne:	Użytkownik zaktualizował sekwencje znaków oraz zakończył działanie aplikacji

4. Estymacja czasowa

1.Budowa podstawowej aplikacji w języku Python
2. Implementacja potrzebnych bibliotek opencv.
3. Ustalenie odpowiednich parametrów Kamery oraz wdrożenie jej do aplikacji.
4. Stworzenie funkcji aplikacji odpowiedzialnej za wykrywanie oczu

5. Wykrywanie drugiego oka.
6. Wykrywanie i wyświetlanie mrugnięcia.
7. Wykrywanie kierunku spojrzenia.
8. Wyświetlanie kierunku spojrzenia.
9.Implementacja wirtualnej klawiatury.
10.Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze.
11.Umieszczenie kodu aplikacji podzielonego na wirtualną klawiaturę i wykrywanie mrugania oraz spojrzenia oczu do jednego pliku.
12. Implementacja wirtualnej klawiatury.
13. Stworzenie plików mp4 do wykonywanych komend podczas wyborów użytkownika.
14. Zadbanie o responsywność aplikacji
15.Implementacja Podziału klawiatury na dwie części L/P
16. Dodawanie komentarzy do kodu programu
17. Optymalizowanie kodu
18. Implementacja przycisku na klawiaturze, który odpowiedzialny jest za przenoszenie do witryny Google oraz kopiowanie tekstu z tablicy do schowka.
19.Implementacja przycisku do usuwania ostatnio napisanej litery z tablicy.
20. Zmiana funkcji mrugania, aby zaimplementować mruganie lewym okiem.
21. Usuwanie liter prawym okiem
22. Optymalizacja kodu.

Sprint 1 (6dni / 48h) / 24h

1.Budowa podstawowej aplikacji w języku Python (3 dni)
2. Implementacja potrzebnych bibliotek opencv. (3 dni)
3. Ustalenie odpowiednich parametrów Kamery oraz wdrożenie jej do aplikacji. (4 dni)
4. Stworzenie funkcji aplikacji odpowiedzialnej za wykrywanie oczu (3 dni)

Sprint 2 (5 dni / 40h) / 38h

1. Wykrywanie drugiego oka. (2 dni)
2. Wykrywanie i wyświetlanie mrugnięcia. (1 dni)
3. Wykrywanie kierunku spojrzenia. (1 dni)

4. Wyświetlanie kierunku spojrzenia. (1 dni)

Sprint 3 (7 dni / 56h) 37h

1.Implementacja wirtualnej klawiatury (2 dni)
2. Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze. (2 dni)
3. Umieszczenie kodu aplikacji podzielonego na wirtualną klawiaturę i wykrywanie mrugania oraz spojrzenia oczu do jednego pliku (3 dni)

Sprint 4 (7 dni / 56h) 36h

1. Implementacja wirtualnej klawiatury (2 dni)
2. Stworzenie plików mp4 do wykonywanych komend podczas wyborów użytkownika (1 dni)
3. Zadbanie o responsywność aplikacji(1 dni)
4.Implementacja podziału klawiatury na dwie części L/P (3 dni)

Sprint 5 (4 dni / 32h) 19h / 17h 15 min

1. Dodawanie komentarzy do kodu programu (1 dni)
2. Optymalizowanie kodu (1 dni)
3. Implementacja przycisku na klawiaturze, który odpowiedzialny jest za przenoszenie do witryny Google oraz kopiowanie tekstu z tablicy do schowka. (1 dni)
4. Implementacja przycisku do usuwania ostatnio napisanej litery z tablicy. (1 dni)

Sprint 6 (3 dni / 24h) 5h

1. Zmiana funkcji mrugania, aby zaimplementować mruganie lewym okiem.(1 dni)
2. Usuwanie liter prawym okiem (1 dni)
3. Optymalizacja kodu.(1 dni)

MVP

Aplikacja będzie wykrywać ruch oczu oraz powiek z obrazu kamery na żywo. Ruchem gałek ocznych będziemy wybierać stronę klawiatury(l/p) oraz mrugnięciem będzie można wybierać litery z klawiatury

wyświetlanej na ekranie. Na ekranie pojawi się tablica, w którym będą nadpisywane litery wybrane przez użytkownika.
Termin oddania projektu szacuje się na 10.05.2022 r.

5.Implementacja

```python
#Projekt ZPI
"""
Zespołowe przedsięwzięcie inżynierskie

Autorzy:
Adam Szczepański
Patryk Matusik
Adrian Święs

kierunek: Informatyka Stosowana
"""
import cv2
import numpy as np
import dlib
import pyglet
import time
from math import hypot
from tkinter import *
import webbrowser

okno = Tk()


#Ładowanie dźwięków
dzwiek = pyglet.media.load("dzwiek.wav", streaming=False)
lewo_dzwiek = pyglet.media.load("lewo.wav", streaming=False)
prawo_dzwiek = pyglet.media.load("prawo.wav", streaming=False)

#Wczytwanie ustawień kamery
cap = cv2.VideoCapture(0)
frame_width = 1280
frame_height = 720
fps = 30.0

tablica = np.zeros((100, 1400), np.uint8)
tablica [:] = 255

#Punkty orientacyjne twarzy
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("68_punktow_orientacyjnych_twarzy.dat")


#Ustawienia klawiatury
klawiatura = np.zeros((600, 1210, 3), np.uint8)
ustawienie_klawiszy_1 = {0: "Q", 1: "W", 2: "E", 3: "R", 4: "T",
                         5: "DEL", 6: "A", 7: "S", 8: "D", 9: "F", 10: "G",
                         11: "BACK", 12: "Z", 13: "X", 14: "C", 15: "V", 16: "B", 17: "COPY"}
ustawienie_klawiszy_2 ={0: "Y", 1: "U", 2: "I", 3: "O", 4: "P",
                         5: "DEL", 6: "H", 7: "J", 8: "K", 9: "L", 10: "?", 11:"BACK",
                         12: "V", 13: "B", 14: "N", 15: "M", 16: "_", 17: "COPY"}

def litera(litera_index, tekst, podswietlenie):
#Klawisze
    if litera_index == 0:
        x = 0
        y = 0
        font_scale = 10
    elif litera_index == 1:
        x = 200
        y = 0
```

```
60          y = 0
61          font_scale = 10
62      elif litera_index == 2:
63          x = 400
64          y = 0
65          font_scale = 10
66      elif litera_index == 3:
67          x = 600
68          y = 0
69          font_scale = 10
70      elif litera_index == 4:
71          x = 800
72          y = 0
73          font_scale = 10
74      elif litera_index == 5:
75          x = 1010
76          y = 0
77          font_scale = 4
78      elif litera_index == 6:
79          x = 0
80          y = 200
81          font_scale = 10
82      elif litera_index == 7:
83          x = 200
84          y = 200
85          font_scale = 10
86      elif litera_index == 8:
87          x = 400
88          y = 200
89          font_scale = 10
90      elif litera_index == 9:
91          x = 600
92          y = 200
93          font_scale = 10
94      elif litera_index == 10:
95          x = 800
96          y = 200
97          font_scale = 10
98      elif litera_index == 11:
99          x = 1010
100         y = 200
101         font_scale = 4
102     elif litera_index == 12:
103         x = 0
104         y = 400
105         font_scale = 10
106     elif litera_index == 13:
107         x = 200
108         y = 400
109         font_scale = 10
110     elif litera_index == 14:
111         x = 400
112         y = 400
113         font_scale = 10
114     elif litera_index == 15:
115         x = 600
116         y = 400
117         font_scale = 10
118     elif litera_index == 16:
119         x = 800
```

```
        y = 400
        font_scale = 10
    elif litera_index == 17:
        x = 1010
        y = 400
        font_scale = 4


    width = 200 #szerokość liter
    height = 200 #wysokośc liter
    th = 3 #grubość liter

#Ustawienia tekstu
    font_letter = cv2.FONT_HERSHEY_PLAIN
    font_th = 4
    text_size = cv2.getTextSize(tekst, font_letter, font_scale, font_th)[0]
    width_text, height_text = text_size[0], text_size[1]
    text_x = int((width - width_text) / 2) + x
    text_y = int((height + height_text) / 2) + y
#Podświetlenie klawiszy
    if podswietlenie is True:
        cv2.rectangle(klawiatura, (x + th, y + th), (x + width - th, y + height - th), (255, 255, 255), -1)
        cv2.putText(klawiatura, tekst, (text_x, text_y), font_letter, font_scale, (51, 51, 51), font_th)
    else:
        cv2.rectangle(klawiatura, (x + th, y + th), (x + width - th, y + height - th), (255, 0, 0), th)
        cv2.putText(klawiatura, tekst, (text_x, text_y), font_letter, font_scale, (255, 255, 255), font_th)
#Menu
def rys_menu():
    rows, cols, _ = klawiatura.shape
    th_lines = 4 # grubość linii
    cv2.line(klawiatura, (int(cols/2) - int(th_lines/2), 0),(int(cols/2) - int(th_lines/2), rows),
             (51, 51, 51), th_lines)
    cv2.putText(klawiatura, "LEWO", (160, 300), font, 6, (255, 255, 255), 5)
    cv2.putText(klawiatura, "PRAWO", (160 + int(cols/2), 300), font, 6, (255, 255, 255), 5)



def srodek(p1 ,p2):
    return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2)

font = cv2.FONT_HERSHEY_PLAIN

#Zdefiniowanie funkcji do wykrywania mrugania
def get_blinking_ratio(eye_points, facial_landmarks):
    left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
    center_top = srodek(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2]))
    center_bottom = srodek(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_points[4]))

    hor_line_lenght = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
    ver_line_lenght = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1]))

    ratio = hor_line_lenght / ver_line_lenght
    return ratio

#Zdefiniowanie funkcji konturów oczu
def kontury_oczu(facial_landmarks):
    left_eye = []
    for n in range(36, 42):
```

```python
        x = facial_landmarks.part(n).x
        y = facial_landmarks.part(n).y
        left_eye.append([x, y])
    for n in range(42, 48):
        x = facial_landmarks.part(n).x
        y = facial_landmarks.part(n).y
    left_eye = np.array(left_eye, np.int32)
    return left_eye
#Zdefiniowanie funkcji do wykrywania współczynnika spojrzenia
def get_gaze_ratio(eye_points, facial_landmarks):
    left_eye_region = np.array([(facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y),
                                (facial_landmarks.part(eye_points[1]).x, facial_landmarks.part(eye_points[1]).y),
                                (facial_landmarks.part(eye_points[2]).x, facial_landmarks.part(eye_points[2]).y),
                                (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y),
                                (facial_landmarks.part(eye_points[4]).x, facial_landmarks.part(eye_points[4]).y),
                                (facial_landmarks.part(eye_points[5]).x, facial_landmarks.part(eye_points[5]).y)], np.int32)

    height, width, _ = frame.shape
    mask = np.zeros((height, width), np.uint8)
    cv2.polylines(mask, [left_eye_region], True, 255, 2)
    cv2.fillPoly(mask, [left_eye_region], 255)
    eye = cv2.bitwise_and(gray, gray, mask=mask)

    min_x = np.min(left_eye_region[:, 0])
    max_x = np.max(left_eye_region[:, 0])
    min_y = np.min(left_eye_region[:, 1])
    max_y = np.max(left_eye_region[:, 1])

    gray_eye = eye[min_y: max_y, min_x: max_x]
    _, threshold_eye = cv2.threshold(gray_eye, 70, 255, cv2.THRESH_BINARY)
    height, width = threshold_eye.shape
    left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]
    left_side_white = cv2.countNonZero(left_side_threshold)

    right_side_threshold = threshold_eye[0: height, int(width / 2): width]
    right_side_white = cv2.countNonZero(right_side_threshold)

    if left_side_white == 0:
        gaze_ratio = 1
    elif right_side_white == 0:
        gaze_ratio = 5
    else:
        gaze_ratio = left_side_white / right_side_white
    return gaze_ratio

#Liczniki
frames = 0
litera_index = 0
mruganie = 2
frames_to_blink = 7
aktywna_ramka = 12


#Ustawienia tekstu i klawiatury
tekst = ""
zaznaczona_litera = True
wybrana_klawiatura = "lewo"
ostatnio_wybrana_klawiatura = "lewo"
wybor_menu = True
```

```python
wybor_klawiatury = 0


while True:
    _, frame = cap.read()
    rows, cols, _ = frame.shape
    klawiatura[:] = (26, 26, 26)
    frames += 1
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    #Rysowanie białej przestrzeni dla paska ładowania
    frame[rows - 50: rows, 0: cols] = (255, 255, 255)



    if wybor_menu is True:
        rys_menu()

    #Wybranie klawiatury
    if wybrana_klawiatura == "lewo":
        keys_set = ustawienie_klawiszy_1
    else:
        keys_set = ustawienie_klawiszy_2
    zaznaczona_litera = keys_set[litera_index]


    #Wykrywanie twarzy
    faces = detector(gray)
    for face in faces:
        landmarks = predictor(gray, face)

        left_eye = kontury_oczu(landmarks)

        #Wykrywanie mrugania
        left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)
        blinking_ratio = left_eye_ratio

        #Kontur Oczu (Czerwony)
        cv2.polylines(frame, [left_eye], True, (0, 0, 255), 2)

        if wybor_menu is True:
            #Wykrywanie spojrzenia, aby wybrać lewą lub prawą stronę klawiatury
            gaze_ratio_left_eye = get_gaze_ratio([36, 37, 38, 39, 40, 41], landmarks)
            gaze_ratio_right_eye = get_gaze_ratio([42, 43, 44, 45, 46, 47], landmarks)
            gaze_ratio = (gaze_ratio_right_eye + gaze_ratio_left_eye) / 2

            if gaze_ratio <= 0.9:
                wybrana_klawiatura = "prawo"
                wybor_klawiatury += 1
                #Jeśli użytkownjik patrzy w jedną stronę więcej niż 18 klatek, przejdź do klawiatury
                if wybor_klawiatury == 18:
                    wybor_menu = False
                    prawo_dzwiek.play()
                    #Ustaw liczbę klatek na 0, gdy wybrana jest klawiatura
                    frames = 0
                    wybor_klawiatury = 0

                if wybrana_klawiatura != ostatnio_wybrana_klawiatura:
```

```
                ostatnio_wybrana_klawiatura = wybrana_klawiatura
                wybor_klawiatury = 0

            else:
                wybrana_klawiatura= "lewo"
                wybor_klawiatury += 1
                #Jeśli użytkownik patrzy w jedną stronę więcej niż 18 klatek, przejdź do klawiatury
                if wybor_klawiatury == 18:
                    wybor_menu = False
                    lewo_dzwiek.play()
                    #Ustaw liczbę klatek na 0, gdy wybrana jest klawiatura
                    frames = 0
                if wybrana_klawiatura != ostatnio_wybrana_klawiatura:
                    ostatnio_wybrana_klawiatura = wybrana_klawiatura
                    wybor_klawiatury = 0

        else:
            #Wykryj mruganie, aby wybrać klawisz, który się świeci

            if blinking_ratio > 5:
                mruganie += 1
                frames -= 1
                #Pokaż zielony kontur oczu, gdy są zamknięte
                cv2.polylines(frame, [left_eye], True, (0, 255, 0), 2)

                #Wpisywanie litery
                if mruganie == frames_to_blink:
                    if zaznaczona_litera != "BACK" and zaznaczona_litera != "_" and zaznaczona_litera != "COPY" and zaznaczona_litera != "<-" and zaznaczona_litera != "DEL" :
                        tekst += zaznaczona_litera
                    if zaznaczona_litera == "COPY":
                        okno.clipboard_clear()
                        okno.clipboard_append(tekst)
                        webbrowser.open('http://google.com/')
                    if zaznaczona_litera == "_":
                        tekst += " "
                    if zaznaczona_litera:
                        if zaznaczona_litera == "DEL":
                            tekst = tekst[: -1]
                            tablica = np.zeros((100, 1400), np.uint8)
                            tablica [:] = 255
                    time.sleep(0.3)
                    dzwiek.play()
                    wybor_menu = True
            else:
                mruganie = 0

    #Pokaż tekst, który piszemy na tablicy
    cv2.putText (tablica, tekst, (20, 50), cv2.FONT_HERSHEY_SIMPLEX, 2,(0, 0, 0), 5)

    #Wyświetlaj litery na klawiaturze
    if wybor_menu is False:
        if frames == aktywna_ramka:
            litera_index += 1
            frames = 0
        if litera_index == 18:
            litera_index = 0


    for i in range(18):
        if i == litera_index:
```

```
                    light = True
                else:
                    light = False
                litera(i, keys_set[i], light)

        #Migający pasek ładowania
        percentage_blinking = mruganie / frames_to_blink
        loading_x = int(cols * percentage_blinking)
        cv2.rectangle(frame, (0, rows - 50), (loading_x, rows), (51, 51, 51), -1)

        cv2.imshow("Okno Kamery", frame)
        cv2.imshow("Wirtualna klawiatura", klawiatura)
        cv2.imshow("Tablica", tablica)
        #Wyłaczenie aplikacji
        key = cv2.waitKey(1)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

6.Testy i ich wyniki

7. Postęp pracy w Jira.

Projekty / projekt / Raporty

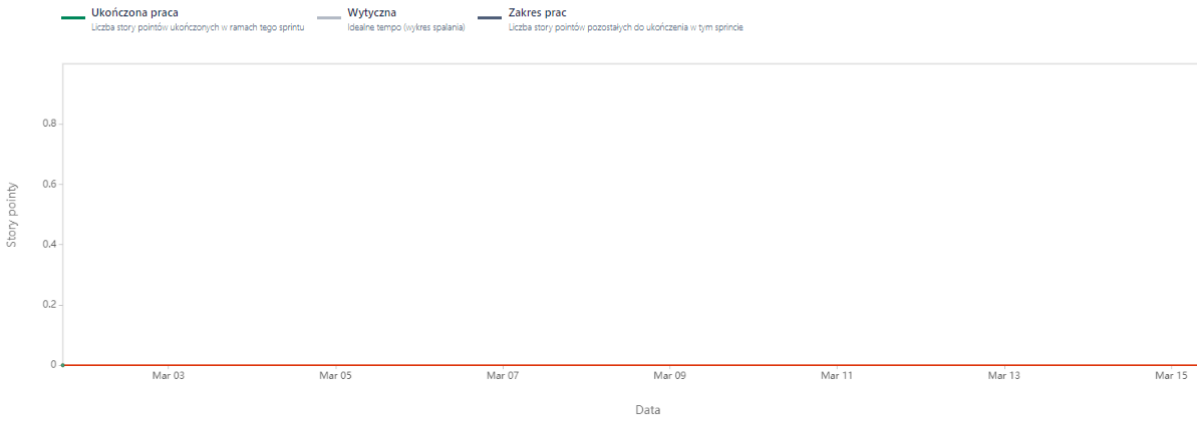
Raport Burnup

Jak czy

Sprint: Sprint 1

Pole oszacowania: Story pointy

Data - 1 marca 2022 - 15 marca 2022



Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Mar 01 2022, 5:41pm	Sprint rozpoczęty		0	0
Tue, Mar 15 2022, 5:39pm	Dodano do sprintu	PROJ3-1 Budowa podstawowej aplikacji w języku Python	0	0
Tue, Mar 15 2022, 5:39pm	Dodano do sprintu	PROJ3-2 Implementacja potrzebnych bibliotek opencv	0	0
Tue, Mar 15 2022, 5:39pm	Zgłoszenie zamknięto	PROJ3-1 Budowa podstawowej aplikacji w języku Python	0	0
Tue, Mar 15 2022, 5:39pm	Zgłoszenie zamknięto	PROJ3-2 Implementacja potrzebnych bibliotek opencv	0	0
Tue, Mar 15 2022, 5:40pm	Dodano do sprintu	PROJ3-3 Ustalenie odpowiednich parametrów Kamery oraz wdrożenie jej do aplikacji.	0	0
Tue, Mar 15 2022, 5:40pm	Dodano do sprintu	PROJ3-4 Stworzenie funkcji aplikacji odpowiedzialnej za wykrywanie oczu	0	0
Tue, Mar 15 2022, 5:41pm	Zgłoszenie zamknięto	PROJ3-3 Ustalenie odpowiednich parametrów Kamery oraz wdrożenie jej do aplikacji.	0	0
Tue, Mar 15 2022, 5:41pm	Zgłoszenie zamknięto	PROJ3-4 Stworzenie funkcji aplikacji odpowiedzialnej za wykrywanie oczu	0	0
Tue, Mar 15 2022, 5:44pm	Ukończony sprint	PROJ3-1 Budowa podstawowej aplikacji w języku Python PROJ3-2 Implementacja potrzebnych bibliotek opencv PROJ3-3 Ustalenie odpowiednich parametrów Kamery oraz wdrożenie jej do aplikacji. PROJ3-4 Stworzenie funkcji aplikacji odpowiedzialnej za wykrywanie oczu	0	0

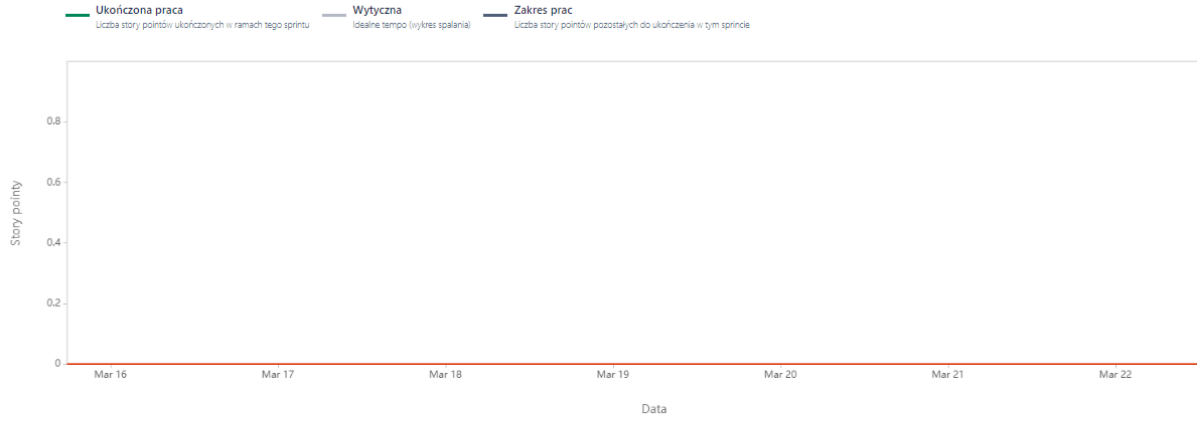
Sprint

Sprint 2

Pole oszacowania

Story pointy

Data - 15 marca 2022 - 22 marca 2022

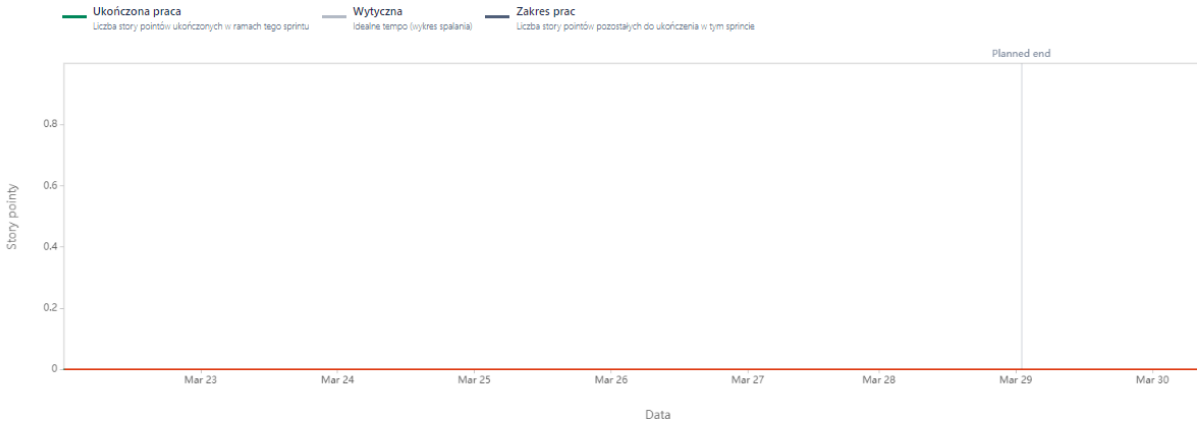


Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Mar 15 2022, 5:47pm	Sprint rozpoczęty		0	0
Tue, Mar 15 2022, 5:58pm	Dodano do sprintu	PROJ3-11 Wykrywanie drugiego oka	0	0
Fri, Mar 18 2022, 5:27pm	Dodano do sprintu	PROJ3-14 Wykrywanie Mrugania	0	0
Fri, Mar 18 2022, 5:32pm	Dodano do sprintu	PROJ3-15 Wykrywanie kierunku spojrzenia	0	0
Fri, Mar 18 2022, 5:39pm	Zgłoszenie zamknięto	PROJ3-11 Wykrywanie drugiego oka	0	0
Fri, Mar 18 2022, 5:41pm	Dodano do sprintu	PROJ3-16 Wyświetlanie kierunku spojrzenia	0	0
Fri, Mar 18 2022, 5:42pm	Zgłoszenie zamknięto	PROJ3-14 Wykrywanie Mrugania	0	0
Sat, Mar 19 2022, 5:55pm	Zgłoszenie zamknięto	PROJ3-15 Wykrywanie kierunku spojrzenia	0	0
Sun, Mar 20 2022, 5:12pm	Zgłoszenie zamknięto	PROJ3-16 Wyświetlanie kierunku spojrzenia	0	0
Tue, Mar 22 2022, 6:01pm	Ukończony sprint	PROJ3-11 Wykrywanie drugiego oka PROJ3-14 Wykrywanie Mrugania PROJ3-15 Wykrywanie kierunku spojrzenia PROJ3-16 Wyświetlanie kierunku spojrzenia	0	0

Sprint: Sprint 3

Pole oszacowania: Story pointy

Data - 22 marca 2022 - 29 marca 2022



Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Mar 22 2022, 12:00am	Sprint rozpoczęty		0	0
Tue, Mar 22 2022, 4:36pm	Dodano do sprintu	PROJ3-17 Implementacja wirtualnej klawiatury	0	0
Tue, Mar 22 2022, 4:36pm	Dodano do sprintu	PROJ3-18 Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze	0	0
Tue, Mar 22 2022, 4:36pm	Dodano do sprintu	PROJ3-20 Umieszczenie kodu aplikacji podzielonego na wirtualną klawiaturę i wykrywanie mrugania oraz spojrzenia oczu do jednego pliku	0	0
Fri, Mar 25 2022, 12:39pm	Zgłoszenie zamknięto	PROJ3-17 Implementacja wirtualnej klawiatury	0	0
Mon, Mar 28 2022, 12:06pm	Zgłoszenie zamknięto	PROJ3-20 Umieszczenie kodu aplikacji podzielonego na wirtualną klawiaturę i wykrywanie mrugania oraz spojrzenia oczu do jednego pliku	0	0
Mon, Mar 28 2022, 12:17pm	Zgłoszenie zamknięto	PROJ3-18 Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze	0	0
Mon, Mar 28 2022, 12:17pm	Zgłoszenie otwarte ponownie	PROJ3-18 Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze	0	0
Mon, Mar 28 2022, 12:18pm	Zgłoszenie zamknięto	PROJ3-18 Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze	0	0
Wed, Mar 30 2022, 5:27pm	Ukończony sprint	PROJ3-17 Implementacja wirtualnej klawiatury PROJ3-18 Utworzenie funkcji aplikacji do podświetlania klawisza na wirtualnej klawiaturze PROJ3-20 Umieszczenie kodu aplikacji podzielonego na wirtualną klawiaturę i wykrywanie mrugania oraz spojrzenia oczu do jednego pliku	0	0

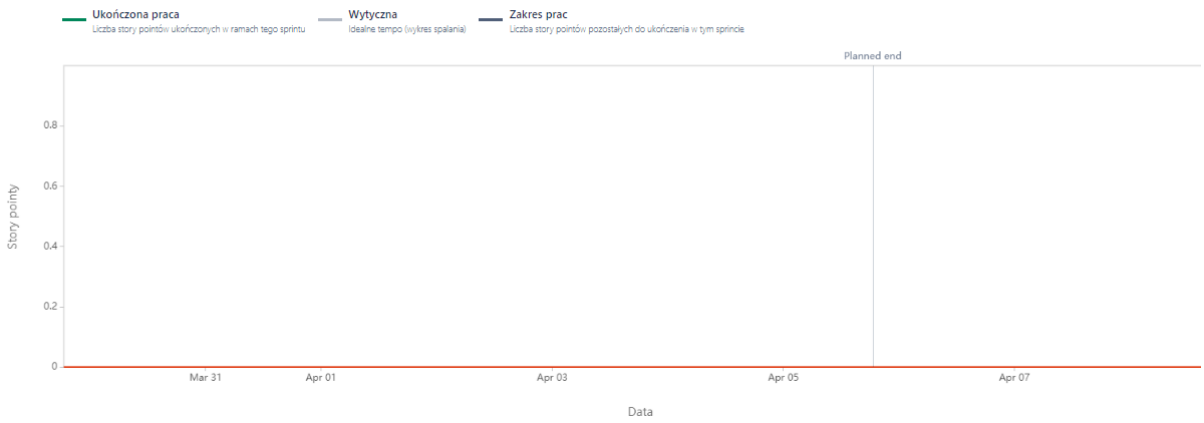
Sprint

Sprint 4

Pole oszacowania

Story pointy

Data - 29 marca 2022 - 5 kwietnia 2022



Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Mar 29 2022, 6:33pm	Sprint rozpoczęty		0	0
Thu, Mar 31 2022, 6:22pm	Dodano do sprintu	PROJ3-21 Stworzenie tablicy do umieszczania liter z wirtualnej klawiatury oraz wykrywanie naciśnięcia klawisza podczas mrugania	0	0
Thu, Mar 31 2022, 6:27pm	Dodano do sprintu	PROJ3-22 Dodanie dźwięku do projektu sygnalizującego kiedy naciskamy klawisz podczas mrugania oraz dźwięku podczas wyboru strony klawiatury	0	0
Thu, Mar 31 2022, 6:28pm	Dodano do sprintu	PROJ3-23 Podział wirtualnej klawiatury na stronę Lewą i Prawą	0	0
Thu, Mar 31 2022, 6:29pm	Dodano do sprintu	PROJ3-24 Implementacja obrysu oczu	0	0
Fri, Apr 01 2022, 11:14am	Zgłoszenie zamknięto	PROJ3-21 Stworzenie tablicy do umieszczania liter z wirtualnej klawiatury oraz wykrywanie naciśnięcia klawisza podczas mrugania	0	0
Fri, Apr 01 2022, 8:59pm	Zgłoszenie zamknięto	PROJ3-22 Dodanie dźwięku do projektu sygnalizującego kiedy naciskamy klawisz podczas mrugania oraz dźwięku podczas wyboru strony klawiatury	0	0
Sat, Apr 02 2022, 4:26pm	Zgłoszenie zamknięto	PROJ3-23 Podział wirtualnej klawiatury na stronę Lewą i Prawą	0	0
Sat, Apr 02 2022, 4:26pm	Zgłoszenie otwarte ponownie	PROJ3-23 Podział wirtualnej klawiatury na stronę Lewą i Prawą	0	0
Sat, Apr 02 2022, 4:26pm	Zgłoszenie zamknięto	PROJ3-23 Podział wirtualnej klawiatury na stronę Lewą i Prawą	0	0
Sat, Apr 02 2022, 4:27pm	Zgłoszenie zamknięto	PROJ3-24 Implementacja obrysu oczu	0	0
Fri, Apr 08 2022, 6:04pm	Ukończony sprint	PROJ3-21 Stworzenie tablicy do umieszczania liter z wirtualnej klawiatury oraz wykrywanie naciśnięcia klawisza podczas mrugania PROJ3-22 Dodanie dźwięku do projektu sygnalizującego kiedy naciskamy klawisz podczas mrugania oraz dźwięku podczas wyboru strony klawiatury PROJ3-23 Podział wirtualnej klawiatury na stronę Lewą i Prawą PROJ3-24 Implementacja obrysu oczu	0	0

Sprint

Sprint 5

Pole oszacowania

Story pointy

Data - 5 kwietnia 2022 - 12 kwietnia 2022

Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Apr 05 2022, 6:04pm	Sprint rozpoczęty	PROJ3-25 Optymalizowanie Kodu PROJ3-26 Dodanie komentarzy do kodu programu	0	0
Fri, Apr 08 2022, 6:06pm	Zgłoszenie zamknięto	PROJ3-26 Dodanie komentarzy do kodu programu	0	0
Fri, Apr 08 2022, 9:26pm	Dodano do sprintu	PROJ3-30 Implementacja przycisku na klawiaturze, który odpowiedzialny jest za przenoszenie do witryny Google oraz za kopiowanie tekstu z tablicy do schowka.	0	0
Sun, Apr 10 2022, 3:07pm	Zgłoszenie zamknięto	PROJ3-30 Implementacja przycisku na klawiaturze, który odpowiedzialny jest za przenoszenie do witryny Google oraz za kopiowanie tekstu z tablicy do schowka.	0	0
Mon, Apr 11 2022, 3:23am	Dodano do sprintu	PROJ3-31 Implementacja przycisku do usuwania ostatnio napisanej litery z tablicy	0	0
Mon, Apr 11 2022, 3:26am	Zgłoszenie zamknięto	PROJ3-31 Implementacja przycisku do usuwania ostatnio napisanej litery z tablicy	0	0
Mon, Apr 11 2022, 8:52am	Zgłoszenie zamknięto	PROJ3-25 Optymalizowanie Kodu	0	0
Mon, Apr 11 2022, 8:52am	Zgłoszenie otwarte ponownie	PROJ3-25 Optymalizowanie Kodu	0	0
Tue, Apr 12 2022, 1:00am	Zgłoszenie zamknięto	PROJ3-25 Optymalizowanie Kodu	0	0
Tue, Apr 12 2022, 1:00am	Zgłoszenie otwarte ponownie	PROJ3-25 Optymalizowanie Kodu	0	0
Tue, Apr 12 2022, 1:01am	Zgłoszenie zamknięto	PROJ3-25 Optymalizowanie Kodu	0	0
Tue, Apr 12 2022, 1:01am	Ukończony sprint	PROJ3-25 Optymalizowanie Kodu PROJ3-26 Dodanie komentarzy do kodu programu PROJ3-30 Implementacja przycisku na klawiaturze, który odpowiedzialny jest za przenoszenie do witryny Google oraz za kopiowanie tekstu z tablicy do schowka. PROJ3-31 Implementacja przycisku do usuwania ostatnio napisanej litery z tablicy	0	0

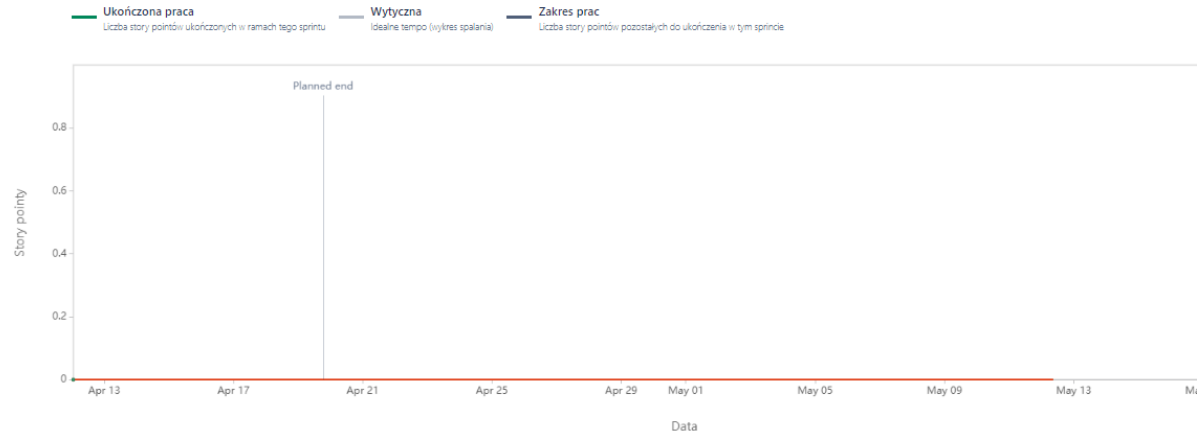
Sprint

Sprint 6

Pole oszacowania

Story pointy

Data - 12 kwietnia 2022 - 19 kwietnia 2022



Data	Zdarzenie	Zgłoszenie	Ukończono	Zakres
Tue, Apr 12 2022, 1:02am	Sprint rozpoczęty		0	0
Tue, Apr 12 2022, 6:43pm	Dodano do sprintu	PROJ3-33 Usuwanie liter prawym okiem	0	0
Tue, Apr 12 2022, 6:43pm	Dodano do sprintu	PROJ3-34 Zmiana funkcji mrugania, aby zaimplementować mruganie lewym okiem.	0	0
Tue, Apr 12 2022, 6:43pm	Dodano do sprintu	PROJ3-35 Optymalizacja kodu.	0	0
Mon, May 09 2022, 9:15pm	Zgłoszenie zamknięto	PROJ3-34 Zmiana funkcji mrugania, aby zaimplementować mruganie lewym okiem.	0	0
Mon, May 09 2022, 9:38pm	Zgłoszenie zamknięto	PROJ3-33 Usuwanie liter prawym okiem	0	0
Mon, May 09 2022, 9:38pm	Zgłoszenie zamknięto	PROJ3-35 Optymalizacja kodu.	0	0
Mon, May 09 2022, 9:38pm	Zgłoszenie otwarte ponownie	PROJ3-33 Usuwanie liter prawym okiem	0	0
Tue, May 10 2022, 9:23am	Zgłoszenie zamknięto	PROJ3-33 Usuwanie liter prawym okiem	0	0

Wykres przepływu

Projekty / projekt / Raporty

Wykres przepływu skumulowanego

Jak czytać ten raport

Filtr dat

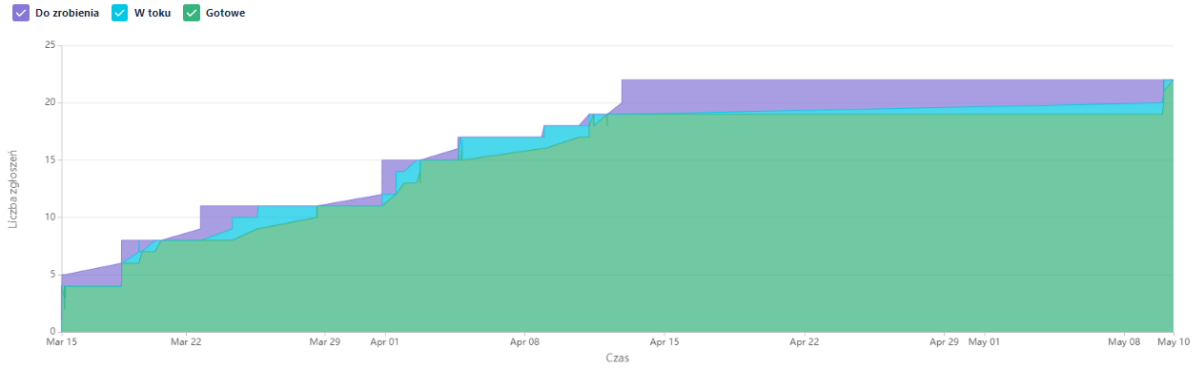
All Time

Od daty

2/18/1993

Do daty

2/18/1993



8.Podsumowanie i bilans

(MVP vs rzeczywistość)

Projekt udało się ukończyć. Założenia początkowe projektu udało się wykonać, ponadto aplikacja została lepiej dopracowana niż to było wstępnie oszacowane. W celu dopracowania projektu zatwierdzanie liter poprzez mruganie dwoma oczami zostało odpowiednio zmienione na zatwierdzanie liter poprzez mrugnięcie lewym okiem oraz usuwanie ostatniego znaku z tablicy poprzez mrugnięcie prawym okiem.

Projekt był dość czasochłonny, a implementacja kodu źródłowego przysporzyła nam trochę problemów, lecz udało nam się je rozwiązać. Wykonanie projektu oszacowaliśmy na 256 godzin, a udało się nam go zrealizować w 157 godzin. Pogłębiliśmy swoją wiedzę w dziedzinie programowania. Aplikacja w przyszłości może zostać dopracowana, aby w szerszym zakresie pomagać ludziom niepełnosprawnym. Dopracowanie mankamentów aplikacji oraz utworzenie rozszerzenia kompatybilnego z witryną Google Chrome byłoby dużym usprawnieniem w spełnianiu celu naszej aplikacji, którym jest niewątpliwie ułatwienie sposobu komunikacji osób niepełnosprawnych, bądź pomoc w czynnościach wykonywanych na co dzień, gdyż internet pełni coraz większą rolę w życiu i coraz bardziej jesteśmy od niego zależni.

9.Link do repozytorium.

https://github.com/Vaxler2k/ProjectDisabled

10.Bibliografia

1. https://pl.wikipedia.org/wiki/Python
2. https://pl.wikipedia.org/wiki/OpenCV
3. https://pl.wikipedia.org/wiki/GitHub
4. https://pl.w3hmong.com/python/numpy_intro.htm
5. https://www.geeksforgeeks.org/introduction-to-pyglet-library-for-game-development-in-python/
6. https://pl.wikipedia.org/wiki/Tkinter