

# Dokumentacja Techniczna

---

Tytuł projektu: Aplikacja do automatycznego oceniania testów na podstawie obrazu

Autorzy:

Miłosz Okarma

Krystian Paluch

Michał Rosa

## 1. Cel aplikacji

Celem aplikacji jest automatyczne ocenianie graficznych testów uczniowskich na podstawie wczytanego szablonu wzorcowego. System analizuje skany testów, wykrywa odpowiedzi w zadanych lokalizacjach i generuje szczegółowe wyniki oraz raporty.

## 2. Technologie

- Język programowania: Python
- GUI: tkinter, customtkinter
- Przetwarzanie obrazu: OpenCV, PIL (Pillow)
- Porównywanie obrazów: cv2.matchTemplate, skimage.metrics.structural\_similarity
- Generowanie raportu PDF: fpdf
- Obsługiwane formaty plików: .png, .jpg, .pdf

## 3. Główne funkcjonalności

3.1 Wczytywanie szablonu testu

3.2 Wczytywanie testów uczniowskich

3.3 Ocena testów

3.4 Generowanie wyników

3.5 Generowanie raportu PDF

3.6 Porównanie różnic

#### 4. Konfiguracja i zmienne globalne

- question\_number – liczba pytań w teście
- score\_points – maksymalna liczba punktów
- points\_table – punktacja dla każdego pytania (opcjonalna)
- pass\_threshold – próg zaliczenia (np. 50%)
- grade\_table – mapa ocen punktowych (1–5)

#### 5. Sposób działania algorytmu oceny

1. Każde pytanie opisane jest przez prostokąt (x, y, w, h).
2. Wycinany jest odpowiadający obszar w teście oraz szablonie.
3. Obliczany jest stopień podobieństwa (cv2.matchTemplate).
4. Jeżeli podobieństwo  $\geq 0.80$ , przyznawany jest punkt.
5. Po zsumowaniu punktów aplikacja przypisuje ocenę końcową i status (zaliczony/niezaliczony).

#### 6. Testowanie i walidacja

##### logic.py

test\_convert\_to\_opencv\_rgb\_input – sprawdza poprawność konwersji obrazu PIL do formatu OpenCV.

test\_find\_corners\_simple – testuje wykrywanie narożników na prostym obrazie.

test\_find\_corners\_arbitrary\_order – testuje działanie algorytmu przy losowej kolejności narożników.

test\_process\_image\_result\_shape – weryfikuje, czy przetworzony obraz ma oczekiwany rozmiar.

test\_process\_image\_type – sprawdza, czy typ danych obrazu to uint8.

test\_process\_image\_corner\_order – sprawdza, czy wynik przetwarzania obrazu to tablica NumPy.

test\_image\_warping\_dimension – sprawdza wymiary obrazu po przekształceniu perspektywy.

##### Ocena i progi punktowe

test\_score\_to\_grade\_5 – testuje przydzielenie oceny 5.

test\_score\_to\_grade\_4 – testuje przydzielenie oceny 4.

test\_score\_to\_grade\_3 – testuje przydzielenie oceny 3.

test\_score\_to\_grade\_2 – sprawdza, czy wynik przekracza próg zaliczenia.

test\_score\_to\_grade\_1 – sprawdza, czy wynik jest poniżej progu zaliczenia.

test\_failed\_test\_detection – wykrywa niezaliczony test.

test\_passed\_test\_detection – wykrywa zaliczony test.

test\_grade\_table\_length – sprawdza długość tablicy ocen.

### **buttons.py**

test\_exit\_program\_calls\_destroy – weryfikuje poprawne zamknięcie aplikacji.

test\_toggle\_help\_text\_behavior – testuje działanie pokazywania/ukrywania pomocy.

test\_show\_page1\_behavior – testuje przełączenie widoku na stronę 1.

test\_toggle\_settings\_option\_behavior – testuje przełączanie widoczności opcji ustawień.

test\_toggle\_test\_settings\_behavior – testuje działanie opcji konfiguracji testu.

test\_show\_page2\_behavior – testuje przełączenie widoku na stronę 2.

test\_approve\_color\_template\_sets\_color – sprawdza zmianę koloru ramki na zielony.

### **main.py**

test\_ctk\_appearance\_mode\_and\_theme – sprawdza ustawienia trybu i motywu aplikacji.

## **7. Struktura GUI**

Aplikacja zawiera interfejs graficzny zbudowany z kilku głównych zakładek dostępnych w górnym pasku nawigacyjnym. Struktura GUI opiera się na następujących widokach:

### **Górny pasek zakładek:**

- Opcje
- Ustawienia testu
- Wczytaj szablon
- Wczytaj testy
- Oceń

- Stwórz raport

#### **Widok – ustawienia testu:**

W centralnym oknie pojawia się panel konfiguracyjny z następującymi polami:

- Liczba pytań – wartość liczbową,
- Liczba punktów – wartość liczbową,
- Progi procentowe ocen – wartości dla ocen 5, 4, 3 i 2 (np. 0.9, 0.7 itd.),
- Dwa przyciski: Zapisz i Anuluj.

#### **Widok – wczytaj testy:**

Ekran podzielony jest na dwie główne sekcje:

- Lewa strona – wyświetla załadowaną kartę odpowiedzi (szablon).
  - Podgląd obrazu karty odpowiedzi,
  - Dwa przyciski: Zatwierdź i Zmień.
- Prawa strona – lista załadowanych testów (nazwy plików graficznych),
  - Przycisk „X” przy każdej nazwie umożliwia usunięcie testu,
  - Dwa przyciski poniżej listy: Zatwierdź i Dodaj.

#### **Widok - stwórz raport:**

- Lista ocenionych testów, każda pozycja zawiera:
  - Nazwę pliku (np. IMG\_0095.png),
  - Liczbę punktów,
  - Ocenę,
  - Status (Zaliczony),
  - Ikona lupy (przycisk do podglądu).
- Panel po prawej stronie z podsumowaniem:
  - Łączna liczba ocenionych testów,
  - Liczba zaliczonych i niezaliczonych,
  - Średnia ocena,
  - Średnia liczba punktów.

## **8. Zależności i instalacja**

pip install opencv-python

pip install pillow

pip install customtkinter

pip install fpdf

pip install scikit-image

Dodatkowo:

- Wymagana czcionka: Assets/DejaVuSans.ttf

## 9. Struktura plików projektu

ZTP\_Projekt-TESTOWA/

- |— .idea/
- |— Assets/
  - | |— DejaVuSans.ttf
  - | |— passwords.json
  - | |— raport.pdf
  - | |— tests.py
- |— Images/
  - | |— IMG\_0095.png
  - | |— IMG\_0099.png
  - | |— IMG\_0990.png
  - | |— IMG\_0996.png
  - | |— left-arrow.png
  - | |— right-arrow.png
  - | |— loupe.png
- |— Imports/
  - | |— buttons.py
  - | |— constants.py
  - | |— imports.py
  - | |— logic.py
  - | |— raport.py

|— main.py

|— README.md