# Dokumentacja techniczna Programu OfficeHelper

Inez Małecka (52738)

Stworzono przy użyciu LaTeXw programie TeXmaker
na licencji GNU GPL.

# 1 Opis projektu, Przewidywane aktualizacje

## 1.1 Opis projektu, wprowadzenie

Program OfficeHelper powstał jako projekt na zaliczenie zajęć "Programowanie w języku Python"prowadzonych w trakcie pierwszego semestru informatyki w trybie niestacjonarnym (nabór zimowy 2023/2024). W trakcie tworzenia projektu znalazł on funkcjonalne zastosowanie, dzięki czemu przewiduje się ciągły rozwój oraz aktualizację programu.

Początkowym celem projektu było stworzenie prostej aplikacji desktopowej przyjmującej wpis w oknie, a następnie zapisujący go w pliku.txt mającym działać jako prowizoryczna baza danych. Wraz z postępem tworzenia projektu, pomysł na program ewoluował aż do obecnej wersji.

Obecna wersja programu OfficeHelper jest programem katalogowym mającym na celu rejestrowania dokumentów wpływających oraz wychodzących z firmy, zarządzaniem użytkownikami z poziomu aplikacji oraz przeglądaniem rejestru już wprowadzonych dokumentów.

**Uwaga! Program działa jedynie na komputerach z systemem MacOS**

## 1.2 Funkcje zaimplementowane

Z racji na wczesne stadium w/w programu, nie wszystkie funkcje zostały dotychczas zaimplementowane, niektóre zostały zaimplementowane jedynie w zubożałej formie. Obecnie program umożliwia:

- Korzystanie z interaktywnego UI,
- Logowanie do programu przy użyciu danych wprowadzonych do zewnętrznej bazy danych (hasła w bazie są odpowiednio zaszyfrowane).
- Wybór między użytkownikami o różnych uprawnieniach w trakcie logowania.
- Pracę na interaktywnym formularzu, generującym dla każdego dokumentu unikalny kod katalogowy oraz odpowiadający mu kod kreskowy
- W formularzu istnieje możliwość wydruku kodów kreskowych na drukarce do etykiet oraz zapisaniu go w PDF poprzez systemowe okno dialogowe.
- Użytkownikowi z uprawnieniami administratora, udzielono dostępu do panelu administratora gdzie może dodać, usunąć oraz zmienić dane użytkownika.

## 1.3 Planowane implementacje oraz aktualizacje

- Dodanie bazy danych (rejestru) katalogującego wpisy do rejestru za pomocą formularza. Umożliwienie podglądu tej bazy z okna startowego oraz stworzenie dodatkowego okna "Rejestr"umożliwiającego szczegółowe wyszukiwanie w bazie.
- Zmiana okien interfejsu a responsywne - obecnie responsywny jest jedynie panel administratora
- Dodanie widocznej jedynie dla administratorów historii logowań oraz poczynionych zmian w rejestrach.

- Przeniesienie baz danych na serwer sieciowy oraz zapewnienie integralności między kilkoma programami uruchomionymi jednocześnie.
- Dodanie funkcji kalendarza, z listą "do zrobienia", możliwość tworzenia grup, wątków oraz przydzielania zadań do użytkowników.
- Dodanie możliwości przechowywania dokumentów wraz z rejestrem - jeśli dokument jest w formie elektronicznej powinna być możliwość przeciągnąć dokument do formularza co umożliwi ich połączenie i łatwe odnajdywanie.
- Dodanie możliwości tworzenia wątków w formularzach, które wyświetlają powiadomienie o dodaniu wszystkich zainteresowanych użytkowników.
- Dodanie formy komunikacji, najpierw wewnętrzego systemu nadawania e-mail, w późniejszej wersji prosty komunikator wewnętrzny.
- Dodanie aplikacji na smartphone bądź webowej dla panelu administratora i szybkiego przeglądania statystyk.

**Uwaga! przed podjęciem się dalszych prac, należy porównać środowiska programistyczne oraz dostępne języki programowania. Należy dokonać ewaluacji spójności kodu oraz przestrzeni czasowej oraz pamięciowej algorytmów.**

## 1.4 Środowisko programistyczne

Projekt ten został stworzony w zintegrowanym środowisku programistycznym (IDE) PyCharm 2023.3.2 (Professional Edition) z wykorzystaniem języka Python3.12. Do stworzenia aplikacji z nowoczesnym interfejsem użyto framework PyQt 6.6.1 oraz w programie QtCreator 12.1.0 dla biblioteki Qt 6.6.0 .

# 2 Informacje szczegółowe

## 2.1 Deklaracja bibliotek wykorzystanych w projekcie

Wycinek z main.py - biblioteki

```
import os
import sys
import random
from datetime import datetime
import hashlib
import sqlite3
# Poniżej, import elementów framework'u PyQt:
from PyQt6 import QtWidgets, uic
from PyQt6.QtCore import Qt, QTimer
from PyQt6.QtPrintSupport import QPrinter QPrintDialog
from PyQt6.QtWidgets import QWidget, QTableWidget, QTableWidgetItem
import code128
import io
from PIL import Image, ImageDraw, ImageFont
from PyQt6.QtGui import QPixmap, QPainter, QImage
```

Na powyższym wycinku kodu można zaobserwować wszystkie biblioteki jakie zostały wykorzystane do stworzenia niniejszego programu. Wszystkie są niezbędne do prawidłowego działania programu w sposób zaprojektowany przez autorkę.

- **os**: Biblioteka os umożliwia interakcję z systemem operacyjnym.
- **sys**: Moduł sys zapewnia dostęp do niektórych zmiennych i funkcji związanych ze środowiskiem uruchomieniowym Pythona.
- **random**: Biblioteka random umożliwia generowanie liczb pseudolosowych oraz wykonywanie operacji losowych.
- **datetime**: Moduł datetime umożliwia operacje na datach i godzinach.
- **hashlib**: Biblioteka hashlib umożliwia obliczanie funkcji skrótu (hash) dla danych.
- **sqlite3**: Moduł sqlite3 zapewnia interfejs do obsługi bazy danych SQLite z poziomu Pythona.
- **PyQt6**: Biblioteka PyQt6 to zestaw narzędzi do tworzenia aplikacji graficznych w języku Python z wykorzystaniem biblioteki Qt.
- **code128**: Biblioteka code128 umożliwia generowanie kodów kreskowych w formacie Code 128.
- **PIL (Python Imaging Library)**: Biblioteka PIL umożliwia manipulację obrazami w formacie bitmapowym.

## 2.2 Klasy i główne funkcje programu:

Główne klasy programu:

- PoczątkowaBazaDanych - Inicjalizuje bazę danych SQLite i tworzy tabelę użytkowników jeśli te nie istnieją.
- OknoLogowania - Obsługuje proces logowania użytkownika.
- OknoGlowne - Wyświetla główne okno aplikacji po zalogowaniu, umożliwiając dostęp do funkcji.
- FormularzDokumentu - Tworzy formularz generowania kodów katalogowych i umożliwia ich wydrukowanie.
- Okno_Panel_Administracyjny - Zapewnia interfejs do zarządzania użytkownikami, ich dodawania, usuwania i modyfikacji.

Funkcje globalne programu:

- sprawdz_dostep_dekorator - Dekorator funkcji sprawdzający uprawnienia użytkownika..
- Generowanie_numeru - Generuje unikatowy numer katalogowy na podstawie aktualnej daty i losowej liczby.

## 2.3 Krótka analiza programu:

- Aplikacja wykorzystuje interfejs graficzny PyQt6 do stworzenia interfejsu użytkownika.
- Bazuje na bazie danych SQLite do przechowywania informacji o użytkownikach.
- Hasła użytkowników są zapisywane w postaci zahaszowanej (sha256).
- Kod kreskowy generowany jest na podstawie unikatowego numeru i jest wyświetlany w interfejsie graficznym.
- Okno Panelu Administracyjnego umożliwia zarządzanie użytkownikami, ich dodawanie, usuwanie oraz zmianę danych.
- Wydruk kodu kreskowego jest obsługiwany przez systemowe okno dialogowe.
- Aplikacja korzysta z dekoratorów do sprawdzania uprawnień użytkownika w różnych funkcjach.
- Wykorzystuje również moduły do obsługi grafiki oraz generowania kodów kreskowych.

# 3 Procedury testowe

## 3.1 Opis procedur testowych

Procedury testowe stanowią kluczowy element procesu tworzenia oprogramowania. Opracowywane są one w celu przetestowania różnych aspektów systemu podczas jego rozwijania. Procedury testowe obejmują zestawy kroków i testów, które są wykonywane w celu sprawdzenia, czy oprogramowanie działa zgodnie z oczekiwaniami i spełnia określone wymagania.

Podczas opracowywania procedur testowych uwzględniono różne scenariusze i warunki, które mogą wystąpić podczas użytkowania programu OfficeHelper. Procedury te obejmują testowanie wszystich funkcji dotychczas zaimplementowanych, interakcji użytkownika, obsługi błędów oraz wydajności systemu.

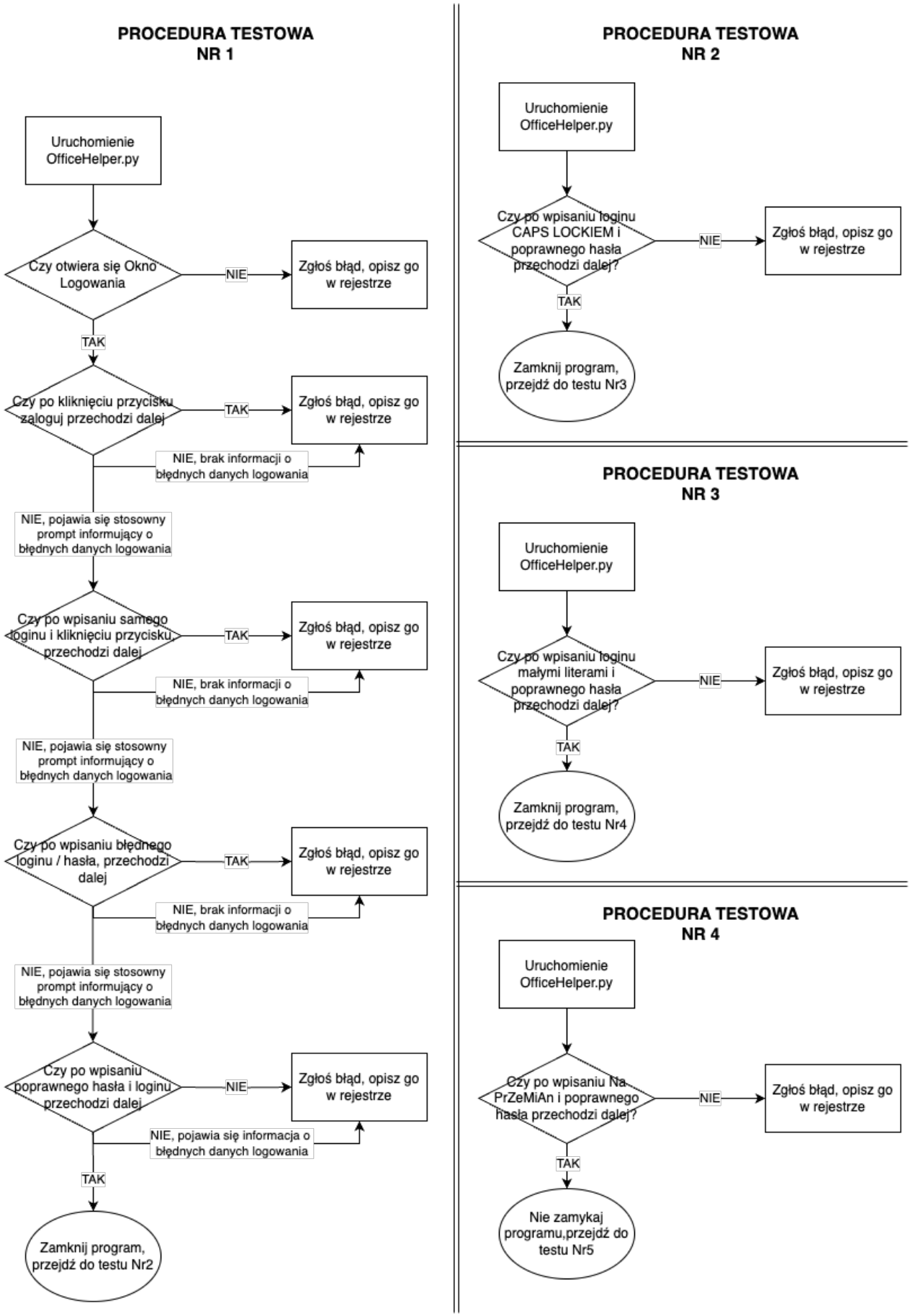
Schemat opracowanych procedur testowych znajduje na końcu dokumentu. Do testów poproszono osobę nie związaną z kodem by przeprowadziła test zgodnie z przedstawionym schematem oraz próbowała używać funkcji programu w sposób jak najbardziej zbliżony do naturalnego użytkowania kodu by sprawdzić czy wszystkie procedury zostały odpowiednio zawarte.

Tester aplikacji przeprowadził 20 prób na różnej rozdzielczości ekranu na 2 komputerach z systemem MacOS - MacbookAir M1 oraz Macbook Pro M2 i nie zgłosił błędów.

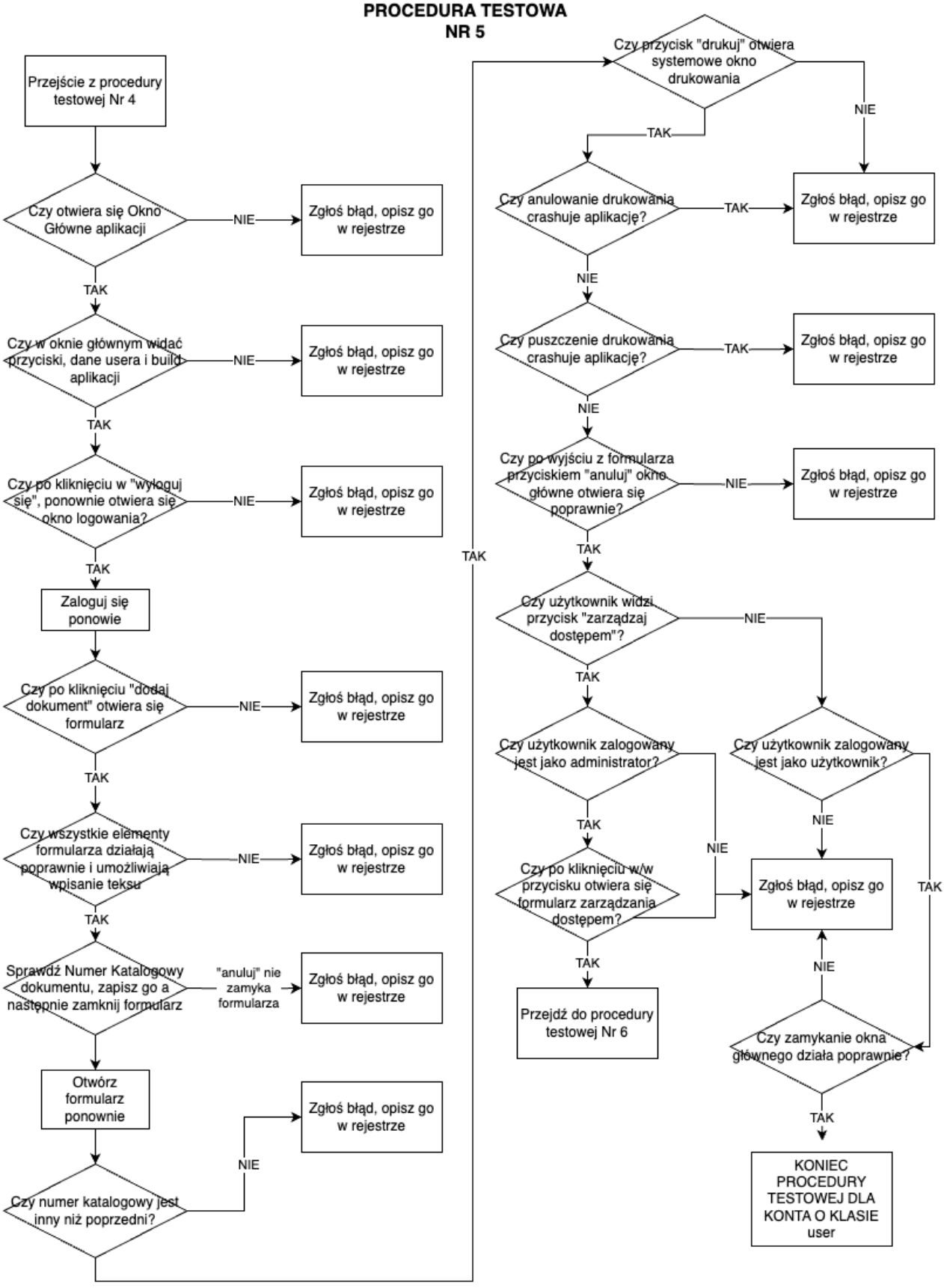
## 3.2 Zestawienie testów

| Numer testu | Rozdzielczość ekranu | Błąd w procedurze nr1 | Błąd w procedurze nr2 | Błąd w procedurze nr3 | Błąd w procedurze nr4 | Błąd w procedurze nr5 | Błąd w procedurze nr6 | Błąd w procedurze nr7 | Jakość obrazu |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1280 x 800 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 2 | 1280 x 800 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 3 | 1280 x 800 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 4 | 1280 x 800 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 5 | 1280 x 800 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 6 | 1140 x 900 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 7 | 1140 x 900 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 8 | 1140 x 900 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 9 | 1140 x 900 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 10 | 1140 x 900 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | wysoka |
| 11 | 1650 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | niska |
| 12 | 1650 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | niska |
| 13 | 1650 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | niska |
| 14 | 1650 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | niska |
| 15 | 1650 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | niska |
| 16 | 1680 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | średnia |
| 17 | 1680 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | średnia |
| 18 | 1680 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | średnia |
| 19 | 1680 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | średnia |
| 20 | 1680 x 1050 | NIE | NIE | NIE | NIE | NIE | NIE | NIE | średnia |

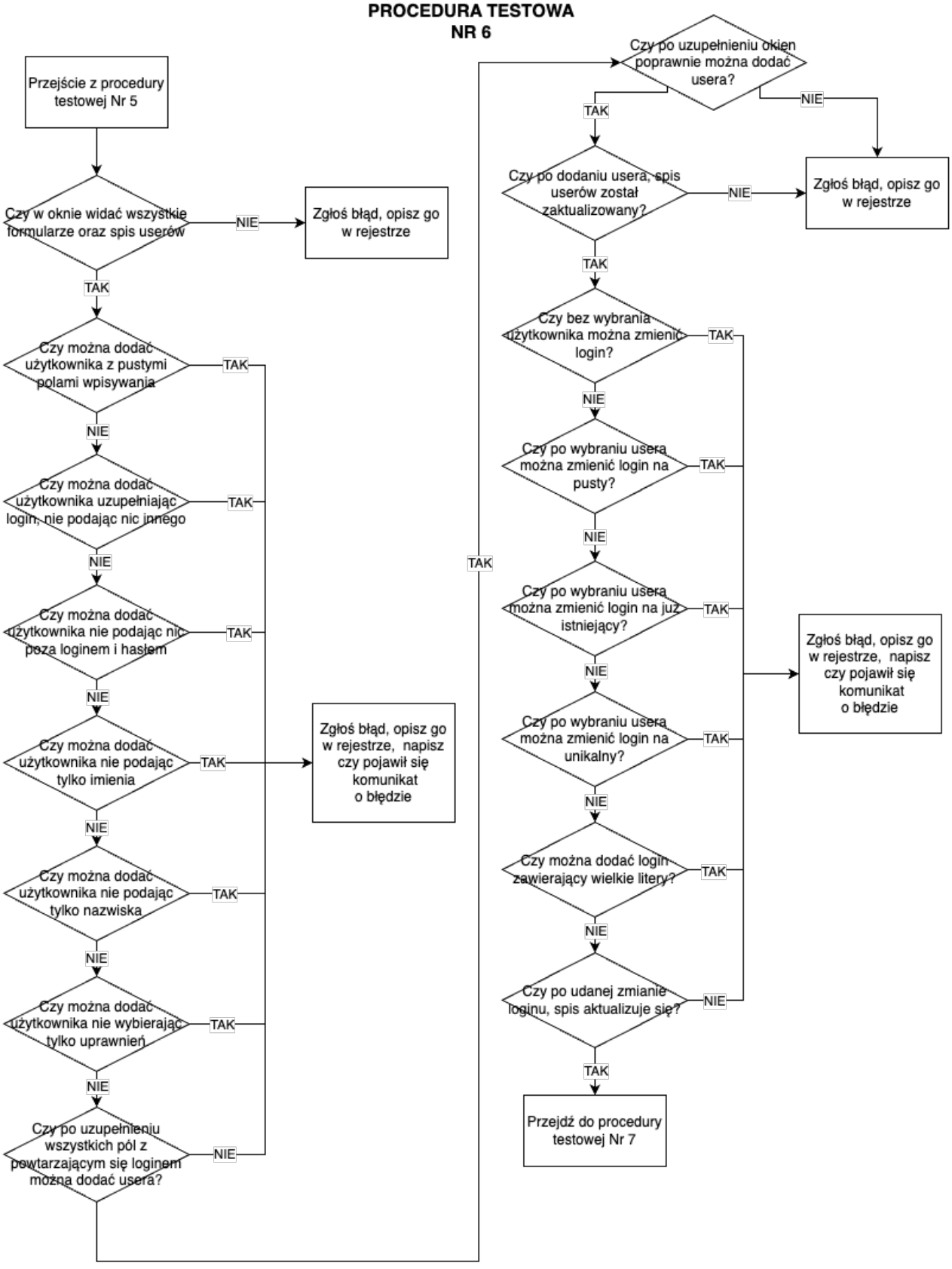
## 3.3 Schematy procedur testowych



**PROCEDURA TESTOWA**
**NR 5**



**PROCEDURA TESTOWA**
**NR 6**



**PROCEDURA TESTOWA**
**NR 7**