PLAN TESTÓW AKCEPTACYJNYCH – „MyTest” w wersji 1.0

METRYKA TESTU

Nr ID:

MT.0.1v1-win_W10_lin_Pop!_OS_22.04_LTS_Ubuntu__22.04_LTS-ipABC_dhcp_connect_core_ram_bios_host_-2023.11.12

Autor:

Agnieszka Iżuk

Testowany Kod:

Maciej 43125

Data:

12.11.2023

Historia zmian:

Dokument w wersji 1.0

Przyjęty format nazewnictwa plików:

<skrót_testowanego_oprogramowania>.<wersja_oprogramowania>v<wersja_dokumentacji>-<system_operacyjny>.<wersja_systemu_operacyjnego>-<kluczowe_parametry>-<data>

np. MT.0.1v1-win_W10- ipABC_ dhcp_con_core_ram_bios_host_-2023.11.03

- <skrót_testowanego_oprogramowania> - to pierwsze litery nazwy oprogramowania:
    - dla „MyTest” - MT
- <wersja_oprogramowania>- składa się z dwóch liczb rozdzielonych kropkami
- <wersja_dokumentacji> - składa się z jednej lub dwóch liczb rozdzielonych kropkami
- <system_operacyjny> - to skrócona nazwa systemu operacyjnego:
    - dla Windows – win
    - dla Linux – lin
- <wersja_systemu_operacyjnego> - to informacja o wersjach systemu operacyjnego dla których domyślnie jest przygotowany test, oddzielamy je znakiem podkreślenia _ lub zakresem_od..._do...:
    - dla Windows – _odW7_doW11
    - dla Linux np. _Pop!_OS_22.04_LTS

- <parametry> - to lista przyjętych skrótów dla wstępnych parametrów systemów operacyjnych, które będą weryfikowane, oddzielonych znakiem podkreślenia _:
    - dla adresu IPv4 – ip<klasy> np. ipABC:
        - adresy prywatne z klasy A
          (zakres 10.0.0.0 – 10.255.255.255, maska: 255.0.0.0/8)
        - adresy prywatne z klasy B
          (zakres 172.16.0.0 – 172.31.255.255, maska: 255.240.0.0/12)
        - adresy prywatne z klasy C
          (zakres 192.168.0.0 – 192.168.255.255, maska: 255.255.0.0/16)
    - dla sposobu przydzielenia adresu IP (dynamicznie / statycznie) – dhcp
    - dla typu połączenia (Ethernet / WiFi) – connect
    - dla uruchomionego proxy – proxy
    - dla liczby rdzeni – core
    - dla pamięci RAM – ram
    - dla wersji BIOS – bios
    - dla nazwy hosta – host
- <data> to data utworzenia pliku w formacie RRRR.MM.DD

WPROWADZENIE DO PROJEKTU

Oprogramowanie ma za zadanie dostarczyć użytkownikowi informacji na temat:

- przydzielonego adresu IPv4
- sposobu dzierżawy adresu IPv4: statyczne/dynamiczne
- rodzaju użytej karty do połączenia z Internetem: wi-fi/ethernet
- czy jest uruchomione Proxy
    - w przypadku uruchomionego proxy jakie IPv4 jest pobierane
    - w przypadku uruchomionego proxy jaki port jest wykorzystywany
- typu systemu operacyjnego
- wersji systemu operacyjnego
- liczba rdzeni procesora
- ilości pamięci RAM
- wersja systemu Bios jeśli jest dostępny
- nazwy hosta

Użytkownik powinien otrzymać informację w wersji wizualnego komunikatu wypisanego na ekranie:

- dla GUI: tekstu w okienku programu
- dla konsoli: tekstu w konsol

PLAN TESTÓW

I. Opis funkcjonalności:

- Program powinien być zgodny z platformą systemu Windows lub Linux i uruchamiać się w formie graficznej.
- Program powinien przyjmować polecenia z wiersza poleceń.
- Program powinien dostosowywać swój rozmiar i położenie zgodnie z standardowymi funkcjami okna programu windows lub linux: maksymalizuj, minimalizuj, zamknij, dostosuj rozmiar samodzielnie poprzez strzałki szerokości i wysokości, przenoszenie okna przy pomocy strzałki.
- Program podaje użytkownikowi informacje o:
  - przydzielonym adresie IPv4
  - sposobu dzierżawy adresu IPv4: statycznej lub dynamicznej
  - rodzaju użytej karty do połączenia z Internetem: wi-fi lub ethernet
  - tym czy jest uruchomione Proxy
    - w przypadku uruchomionego proxy o tym jakie IPv4 jest pobierane
    - w przypadku uruchomionego proxy o tym jaki port jest wykorzystywany
  - typie systemu operacyjnego
  - wersji systemu operacyjnego
  - liczbie rdzeni procesora
  - ilości pamięci RAM
  - wersji systemu Bios jeśli jest dostępny
  - nazwie hosta

II. Opis niefunkcjonalności:

- Program powinien działać szybko i nie obciążać zbytnio zasobów komputera → maksymalne obciążenie dla systemu Windows to ‘Umiarkowany’ lub dla systemu Linux to 2.5 %CPU.
- Program powinien być łatwy w instalacji i użytkowaniu, a jego interfejs powinien być intuicyjny
- Program powinien w czytelny sposób prezentować informację użytkownikowi..
- Program powinien być odporny na przewidywane błędy i awarie, a w razie ich wystąpienia powinien informować użytkownika o przyczynie i sposobie rozwiązania problemu.

III. Potrzebne zasoby:

Wymagania systemowe:

- Program wymaga systemu operacyjnego Windows 10 z prawami administratora.
- Program wymaga użytkownika z uprawnieniami sudo na systemie Linux

WYMAGANIA ŚRODOWISKA TESTOWEGO:

Środowisko testowe składa się z komputera wyposażonego w kartę Ethernetową oraz WiFi, RAM, procesor minimum 1 rdzeniowy, z dostępem do Internetu, możliwością skonfigurowania proxy oraz opartego na systemie BIOS.

Wymagane zasoby osobowe:

Tester manualny - osoba odpowiedzialna za wykonanie i raportowanie testów. Tester powinien mieć umiejętności techniczne związane z testowaniem aplikacji okienkowych i analityczne na poziomie umożliwiającym ocenę oprogramowania pod kątem kryteriów projektowych oraz standardów ISTQB, a także znajomość podstaw systemu Windows oraz sieci.

Kierownik testów - osoba odpowiedzialna za nadzorowanie i koordynowanie całego procesu testowania. Kierownik testów powinien mieć umiejętności zarządzania i liderowania, a także znajomość standardów i najlepszych praktyk testowania. Kierownik testów powinien również monitorować postęp i jakość testów, a także zarządzać zasobami, ryzykami i problemami testowymi.

Analityk testów - osoba odpowiedzialna za analizę wymagań, ryzyk, celów i kryteriów testów. Analityk testów powinien mieć umiejętności komunikacyjne i biznesowe, a także znajomość technologii i procesów testowania. Analityk testów powinien również współpracować z innymi zainteresowanymi stronami, takimi jak programiści, użytkownicy, klienci, itp.

Kierownik projektu rozwoju

Kierownik grupy rozwoju

Kierownik grupy testów systemowych

Kierownik działu płacowego korporacji

Kierownik działu płacowego korporacji

HARMONOGRAM TESTÓW

Testy zostaną przeprowadzone przed wdrożeniem programu zaplanowanym na 20.12.2023.

Zadanie	Poprzednie zadania	Specjalne umiejętności	Odpowiedzialność / Weryfikacja	Wysiłek	Data zakończenia
1 Przygotowanie zarysu planu testów	Ukończony: ☐ opis projektu programu MyTest przygotowany na podstawie wymagań zebranych od klienta ☐ wstępny plan rozwoju programu MyTest ☐ zaakceptowany plan finansowania projektu MyTest	—	Kierownik testów, Starszy analityk testów	4	08-10-2023
2 Przygotowanie specyfikacji testów	Zadanie 1	Wiedza na temat działania systemu Linux lub Windows oraz sieci	Starszy analityk testów	9	12-10-2023
3 Przygotowanie przypadków testowych	Zadanie 2	—	Analityk testów	4	21-10-2023
4 Przygotowanie procedury testowej	Zadanie 3	—	Analityk testów	6	25-10-2023
5 Przygotowanie	Zadanie 4	—	Analityk testów	6	31-10-2023

opisu wymaganych kompetencji i składu osobowego zespołu					
6 Zakończenie transmisji pozycji testowej	Ukończ testy integracyjne	—	Kierownik projektu rozwoju	1	01-11-2023
7 Sprawdź wszystkie procedury kontroli zadań wymaganych do uruchomienia systemu	Zadanie 6	Doświadczenie w kontroli pracy	Tester manualny	1	02-11-2023
8 Wykonaj procedury testowe wprowadzania danych	Zadanie 5, Zadanie 8	—	Analityk testów	1	04-11-2023
9 Wykonać procedury testów wsadowych	Zadanie 5, Zadanie 8	—	Tester manualny	3	07-11-2023
10 Sprawdź wyniki testów partii	Zadanie 10	Wiedza na temat wymagań dotyczących raportów płacowych	Analityk testów	1	08-11-2023

11 Rozwiązywanie raportów z incydentów testowych	Zadanie 9, Zadanie 11	—	Kierownik grupy rozwoju, Kierownik grupy testów systemowych, Kierownik działu płacowego korporacji	2	10-11-2023
12 Powtarzaj zadania (6)–(12), aż wszystkie procedury testowe zakończą się pomyślnie	Zadanie 12	—	—	2	12-11-2023
13 Napisz raport podsumowujący testy systemu	Zadanie 13	—	Kierownik grupy testów systemowych, Kierownik działu płacowego korporacji	1	13-11-2023
14 Prześlij całą dokumentację testową i dane testowe do grupy zarządzającej konfiguracją	zadanie 14	_	Tester automatyczny	2	20-11-2023

I. Testy zostaną przeprowadzone przed wdrożeniem programu.
II. Zasoby testowe: Komputer z systemem operacyjnym Windows lub Linux.
III. Procedury testowe: Testy zostaną przeprowadzone zgodnie z opisanymi przypadkami testowymi.
IV. Metryki testowe: Liczba testów przeprowadzonych, liczba błędów znalezionych.

SPECYFIKACJA TESTÓW:

Cel ogólny:

Sprawdzić, czy program poprawnie wyświetla informacje o adresie IP, typie przydzielenia adresu IP, typie połączenia, uruchomionym proxy, wersji systemu operacyjnego, liczbie rdzeni i pamięci RAM, wersji BIOS oraz nazwie hosta dla systemu Windows i czy jest zgodny z wymaganiami użytkownika oraz założeniami projektu.

Cele szczegółowe:

V. Sprawdzenie, czy program uruchamia się w trybie graficznym, z poprawnie widocznymi wszystkimi komponentami.
VI. Sprawdzenie, czy działa dostosowanie rozmiaru okna do indywidualnych oczekiwań użytkownika.
VII. Sprawdzenie, czy działają przyciski paska tytułu okna programu.
VIII. Sprawdzenie, czy program poprawnie wyświetla adres IP użytkownika.
IX. Sprawdzenie, czy program poprawnie wyświetla typ przydzielenia adresu IP.
X. Sprawdzenie, czy program poprawnie wyświetla typ połączenia.
XI. Sprawdzenie, czy program poprawnie wyświetla informacje o uruchomionym proxy.
XII. Sprawdzenie, czy program poprawnie wyświetla wersję systemu operacyjnego.
XIII. Sprawdzenie, czy program poprawnie wyświetla liczbę rdzeni i pamięć RAM.
XIV. Sprawdzenie, czy program poprawnie wyświetla wersję BIOS.
XV. Sprawdzenie, czy program poprawnie wyświetla nazwę hosta.
XVI. Sprawdzenie, czy program obsługuje wszystkie komendy z wiersza poleceń.
XVII. Sprawdzenie, czy informacje uzyskane z wiersza poleceń są identyczne z tymi uzyskanymi z GUI.

Cele jakościowe:

- Sprawdzenie, czy program posiada pomoc z której może skorzystać użytkownik.
- Sprawdzenie, czy informację są wyświetlane użytkownikowi w sposób czytelny i nie zawierają literówek lub błędów ortograficznych, składniowych itp.
- Sprawdzenie, czy program działa szybko i płynnie, bez opóźnień i zawieszeń.

- Sprawdzenie, czy program jest zgodny ze standardami i normami dotyczącymi bezpieczeństwa, prywatności i ochrony danych użytkownika.
- Sprawdzenie, czy program jest łatwy w utrzymaniu i rozbudowie, z możliwością aktualizacji i dodawania nowych funkcji.
- Sprawdzenie, czy program jest atrakcyjny i przyjazny dla użytkownika, z intuicyjnym i spójnym interfejsem graficznym.
- Sprawdzenie, czy program wyświetla kod błędu lub informację zwrotną dla użytkownika.

Zakres dla celów szczegółowych:

Testy obejmują następujące funkcjonalności programu:

- Wykrywanie adresu IP i typu przydzielenia adresu IP (statycznie lub dynamicznie)
- Wykrywanie typu karty wykorzystywanej do połączenia (Ethernet, Wi-Fi)
- Wykrywanie czy ustawienie proxy jest aktywne i jego adresu oraz portu jeżeli jest aktualnie włączone
- Wykrywanie rodzaju oraz wersji systemu operacyjnego
- Wykrywanie liczby rdzeni procesora i ilości pamięci RAM dostępnej w systemie
- Wykrywanie wersji BIOS, o ile jest dostępny
- Wykrywanie nazwy hosta
- Minimalizacja okna
- Maksymalizacja okna
- Zamykanie okna
- Dostosowanie rozmiaru okna do indywidualnych oczekiwań użytkownika

Sposób testowania:

Testy będą przeprowadzane manualnie na różnych konfiguracjach systemowych, zgodnie z następującymi krokami:

- Uruchomienie programu i sprawdzenie, czy wyświetla się okno z informacjami o systemie
- Sprawdzenie czy wszystkie przyciski działają zgodnie z oczekiwaniami:
    - IPv4 → wyświetlenie informacji na temat IPv4
    - Proxy → wyświetlenie informacji na temat Proxy
    - SO info → wyświetlanie informacji na temat Systemu operacyjnego
    - BIOS → wyświetlenie informacji na temat Biosu
    - HOST → wyświetlenie informacji na temat nazwy komputera
    - minimalizuj „-” → okno aplikacji się minimalizuje na pasku startu
    - maksymalizuj „□” → okno aplikacji rozszerza się na całe okno
    - zamknij „X” → okno aplikacji zamyka się kończąc działanie procesu powiązanego z aplikacją
- Porównanie wyświetlanych informacji z rzeczywistymi danymi systemowymi, uzyskanymi za pomocą innych narzędzi lub poleceń
- Sprawdzenie, czy program poprawnie reaguje na zmiany w systemie, takie jak zmiana adresu IP, typu połączenia, uruchomienie lub wyłączenie proxy, itp.

- Sprawdzenie, czy program nie powoduje błędów, awarii lub niepożądanych efektów ubocznych w systemie

Kryteria akceptacji:

Program spełnia kryteria akceptacji, jeśli:

- Wyświetla poprawne i aktualne informacje o systemie dla każdej z testowanych konfiguracji
- Nie wykazuje żadnych błędów, awarii lub niepożądanych efektów ubocznych w systemie
- Nie wymaga nieuzasadnionych uprawnień lub zasobów systemowych

CELE TESTÓW

XVIII. Plan testów

A. Harmonogram testów: Testy zostaną przeprowadzone przed wdrożeniem programu.
B. Zasoby testowe: Komputer z systemem operacyjnym Windows lub Linux.
C. Procedury testowe: Testy zostaną przeprowadzone zgodnie z opisanymi przypadkami testowymi.
D. Metryki testowe: Liczba testów przeprowadzonych, liczba błędów znalezionych.

XIX. Środowisko testowe

A. Konfiguracja sprzętowa: Komputer z procesorem Intel Core i5 lub wyższym, 8 GB RAM, dyskiem twardym o pojemności 256 GB lub większej.
B. Konfiguracja oprogramowania: System operacyjny Windows 10 lub Linux Pop!_OS 22.04 LTS.
C. Konfiguracja sieciowa:Komputer podłączony do sieci Wi-Fi lub Ethernet.

PROCEDURY TESTOWE

Procedury testowe dla testów funkcjonalnych:

- Uruchom program.
- Sprawdź, czy program poprawnie wyświetla adres IP użytkownika.
- Sprawdź, czy program poprawnie wyświetla typ przydzielenia adresu IP.
- Sprawdź, czy program poprawnie wyświetla typ połączenia.
- Sprawdź, czy program poprawnie wyświetla informacje o uruchomionym proxy.
- Sprawdź,czy program poprawnie wyświetla wersję systemu operacyjnego.
- Sprawdź, czy program poprawnie wyświetla liczbę rdzeni i pamięć RAM.
- Sprawdź,czy program poprawnie wyświetla wersję BIOS.
- Sprawdź,czy program poprawnie wyświetla nazwę hosta dla systemu Windows i Linux.

Procedury testowe dla testów niefunkcjonalnych:

- Uruchom program.
- Sprawdź, czy program posiada funkcję pomoc zawierającą minimum podstawowe informację o tym jak korzystać z programu, opis poszczególnych przycisków.
- Sprawdź, czy informację są wyświetlane w sposób czytelny: czcionka bezszeryfowa, tekst powinien zmieniać swoją wielkość wraz ze zmianą wielkości okna.
- Sprawdź, czy informacje nie zawierają literówek lub błędów ortograficznych albo składniowych.
- Sprawdź, czy program działa szybko i płynnie, bez opóźnień i zawieszeń.
- Sprawdź, czy program jest zgodny ze standardami i normami dotyczącymi bezpieczeństwa, prywatności i ochrony danych użytkownika.
- Sprawdź, czy program jest łatwy w utrzymaniu i rozbudowie, z możliwością aktualizacji i dodawania nowych funkcji.
- Sprawdź, czy program jest atrakcyjny i przyjazny dla użytkownika, z intuicyjnym i spójnym interfejsem graficznym.
- Sprawdź, czy program wyświetla kod błędu lub informację zwrotną dla użytkownika.

Raportowanie wyników testów

- Format raportu: Tekstowy raport w formacie PDF.

- Zawartość raportu: Raport powinien zawierać informacje o wynikach testów oraz błędach znalezionych podczas testów.
- Harmonogram raportowania: Raport powinien być dostarczony do końca dnia roboczego następującego po przeprowadzeniu testów.
- Zarządzanie ryzykiem
- Identyfikacja ryzyka: Ryzykiem jest niepoprawne działanie programu na systemie operacyjnym, na którym nie był testowany.

Analiza ryzyka: Ryzyko jest niskie, ponieważ program został przetestowany na systemach Windows i Linux.

Planowanie reakcji na ryzyko: W przypadku wystąpienia problemów z działaniem programu na innym systemie operacyjnym, program zostanie przetestowany na tym systemie.

ZAŁOŻENIA TESTOWE

Ważne: Przy uruchamianiu programu na urządzeniu z systemem Linux do poprawnego działania musimy założyć posiadanie uprawnień root lub co najmniej usera z dostępem sudo.

Przykładowa komenda na dodanie usera do odpowiedniej grupy :
sudo usermod -aG sudo agnieszkaizuk

Testy zostały napisane w C++

```
void test_ipv4_info_on_linux() {
    QString ip = "192.168.1.8";
    QString is_static = "Statyczne";
    QString interface = "Ethernet";

    QString result = QString("IP: %1\n%2 IP\nInterfejs: %3\n----------").arg(ip, is_static,
interface);
    text_output->setPlainText(result);

    TEST - wynik
    // 192.168.44.21  lub 192.168.1.8
    // Dynamiczne IP
    // Interfejs: Ethernet
}
```

Wynik testu: Pozytywny -> 192.168.1.8, Dynamiczne IP
Negatywny - > Niezgodność (192.168.44.21), Statyczne IP, dynamiczne IP

```
void test_proxy_info_on_linux() {
    bool is_error = true;

    // Symulacja błędu - status proxy nie jest dostępny
    if (is_error) {
        QString error_message = "Błąd pobierania statusu proxy";
        QString result = QString("Błąd: %1\n----------").arg(error_message);
        text_output->setPlainText(result);
    } else {
        QString proxy_status = "Proxy jest włączone";
        QString result = QString("Status Proxy: %1\n----------").arg(proxy_status);
        text_output->setPlainText(result);

}
```

Wynik testu:
Pozytywny - >Proxy jest włączone
Negatywny -> błąd pobrania statusu

WINDOWS

```
void test_ipv4_info_on_windows() {
    QString ip = "192.168.1.100";
    QString is_static = "Statyczne";
    QString interface = "Ethernet";

    QString result = QString("IP: %1\n%2 IP\nInterfejs: %3\n----------").arg(ip, is_static,
interface);
    text_output->setPlainText(result);

    // Przykładowy wynik testu:
    // IP: 192.168.1.53 lub 192.168.44.2
    // Statyczne IP
    // Interfejs: Ethernet
    // ----------
}
```

Wynik testu: Pozytywny -> 192.168.1.53, statyczne IP

Negatywny - > Niezgodność (192.168.44.21)
Dynamiczne IP lub Statyczne

```
void test_proxy_info_on_windows() {
    bool is_error = true;

     if (is_error) {
        QString error_message = "Błąd pobierania statusu proxy";
        QString result = QString("Błąd: %1\n----------").arg(error_message);
        text_output->setPlainText(result);
    } else {
        QString proxy_status = "Proxy jest włączone";
        QString result = QString("Status Proxy: %1\n----------").arg(proxy_status);
        text_output->setPlainText(result);
    }
}
```

Wynik testu:
Pozytywny - >Proxy jest włączone
Negatywny -> błąd pobrania statusu

PRZYPADKI TESTOWE

1. Dla aplikacji MyTestApp.py

```
import unittest

class Test(unittest.TestCase):
    def test_ipv4_info(self):
        app = QApplication([])
        window = MyTestApp(app)
        window.get_ipv4_info()
        text_output = window.text_output.toPlainText()
        self.assertIn("IP:", text_output)
        self.assertIn("Statyczne", text_output) [1]
        self.assertIn("Interfejs:", text_output)[2]
        app.quit()

    def test_system_info(self):
        app = QApplication([])
        window = MyTestApp(app)
        window.get_system_info()
        text_output = window.text_output.toPlainText()
        self.assertIn("Wersja SO:", text_output)
```

[1] Sprawdzamy, czy informacja o typie IP jest wyświetlona

[2] Sprawdzamy, czy informacja o interfejsie jest wyświetlona

```
        self.assertIn("Typ Systemu:", text_output)
        self.assertIn("Rdzenie:", text_output)
        self.assertIn("RAM:", text_output)
        app.quit()

    def test_bios_info(self):
        app = QApplication([])
        window = MyTestApp(app)
        window.get_bios_info()
        text_output = window.text_output.toPlainText()
        self.assertIn("Wersja Biosu:", text_output)
        app.quit()

    def test_hostname(self):
        app = QApplication([])
        window = MyTestApp(app)
        window.get_hostname()
        text_output = window.text_output.toPlainText()
        self.assertIn("Nazwa Hosta:", text_output)
        app.quit()

if __name__ == '__main__':
    unittest.main(argv=[''], exit=False)
```

2. Przypadki Testowe opis:

Testowanie funkcji `get_ipv4_info()`

Cel testu:

- Sprawdzenie poprawności zbierania informacji o adresie IP, typie IP (statycznym/dynamicznym) oraz interfejsie sieciowym.

Kroki testowe:

1. Uruchomienie aplikacji.

2. Wywołanie metody `get_ipv4_info()` w interfejsie graficznym.

Oczekiwane wyniki:

- Wyświetlenie adresu IP.

- Wyświetlenie informacji o typie IP (statyczny/dynamiczny).

- Wyświetlenie nazwy interfejsu sieciowego, jeśli jest dostępna.

Testowanie funkcji `get_system_info()`

Cel testu:

- Weryfikacja prawidłowości zbierania informacji o systemie operacyjnym, typie systemu, liczbie rdzeni procesora oraz ilości pamięci RAM.

Kroki testowe:

1. Uruchomienie aplikacji.

2. Wywołanie metody `get_system_info()` w interfejsie graficznym.

Oczekiwane wyniki:

- Wyświetlenie informacji o wersji systemu operacyjnego.

- Wyświetlanie typu systemu (architektura).

- Wyświetlenie liczby rdzeni procesora.

- Wyświetlanie ilości dostępnej pamięci RAM.

Testowanie funkcji `get_bios_info()`

Cel testu:

- Sprawdzenie poprawności pobierania informacji o wersji BIOSu.

Kroki testowe:

1. Uruchomienie aplikacji.

2. Wywołanie metody `get_bios_info()` w interfejsie graficznym.

Oczekiwane wyniki:

Wyświetlenie informacji o wersji BIOSu.

Cel testu:

Weryfikacja prawidłowości pobierania nazwy hosta.

Kroki testowe:

1. Uruchomienie aplikacji.

2. Wywołanie metody `get_hostname()` w interfejsie graficznym

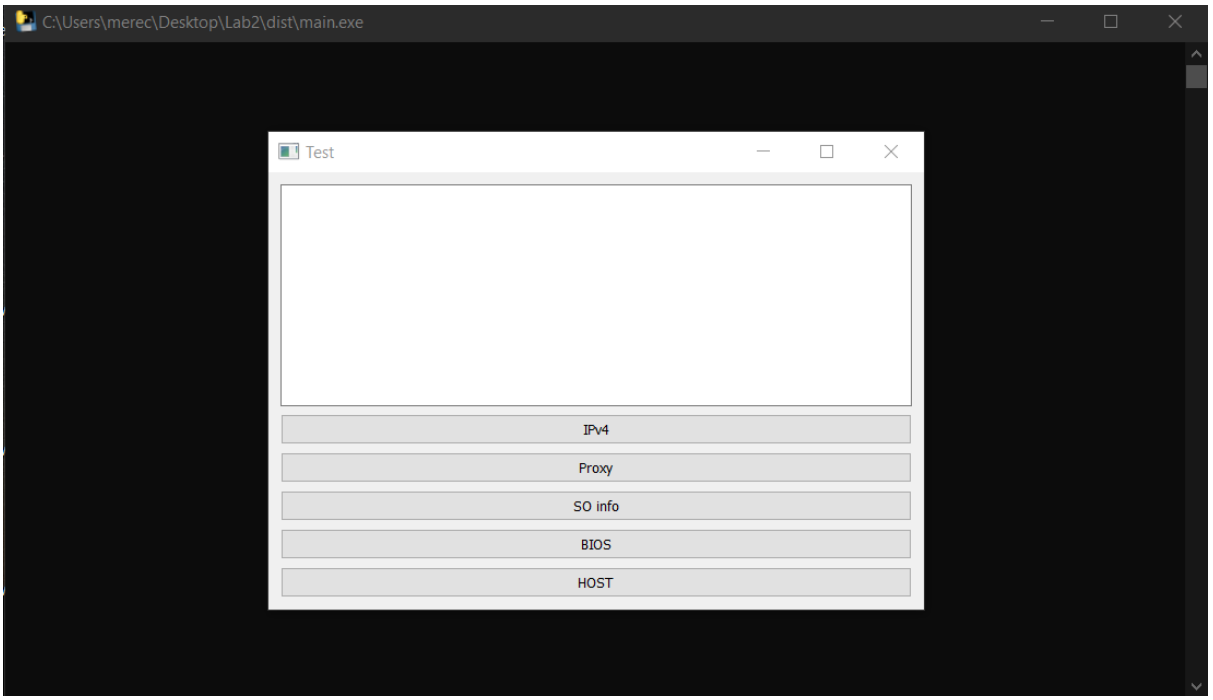
Oczekiwane wyniki:

- Wyświetlenie nazwy hosta systemu.

RAPORTY O BŁĘDACH

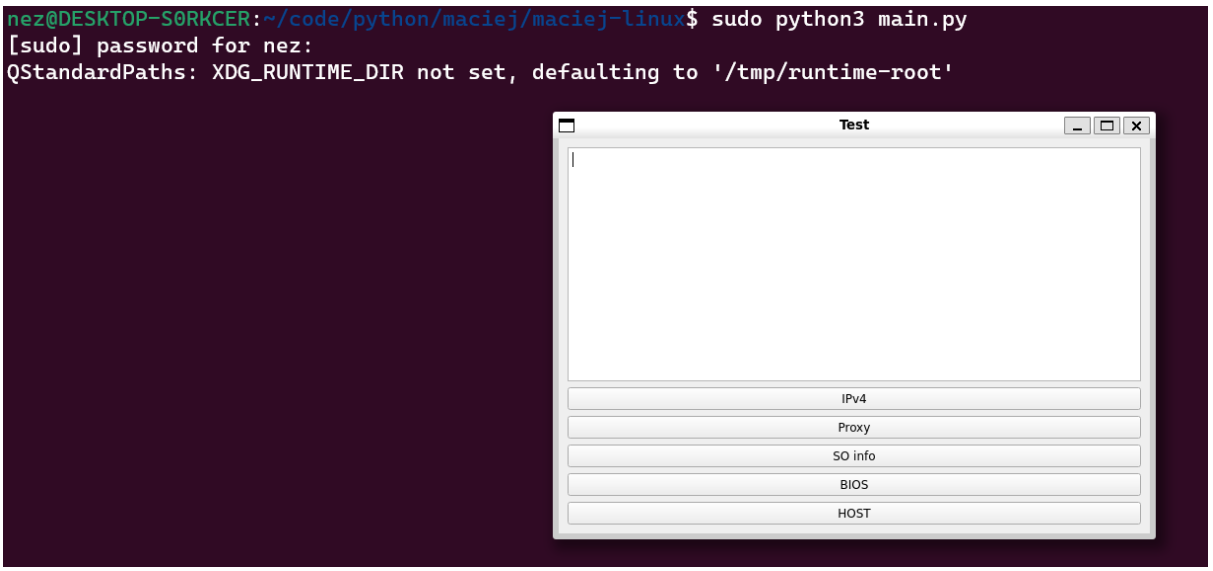
1. Sprawdzenie, czy program uruchamia się w trybie graficznym, z poprawnie widocznymi wszystkimi komponentami.

   Windows:



WYNIK: Zgodnie z oczekiwaniami

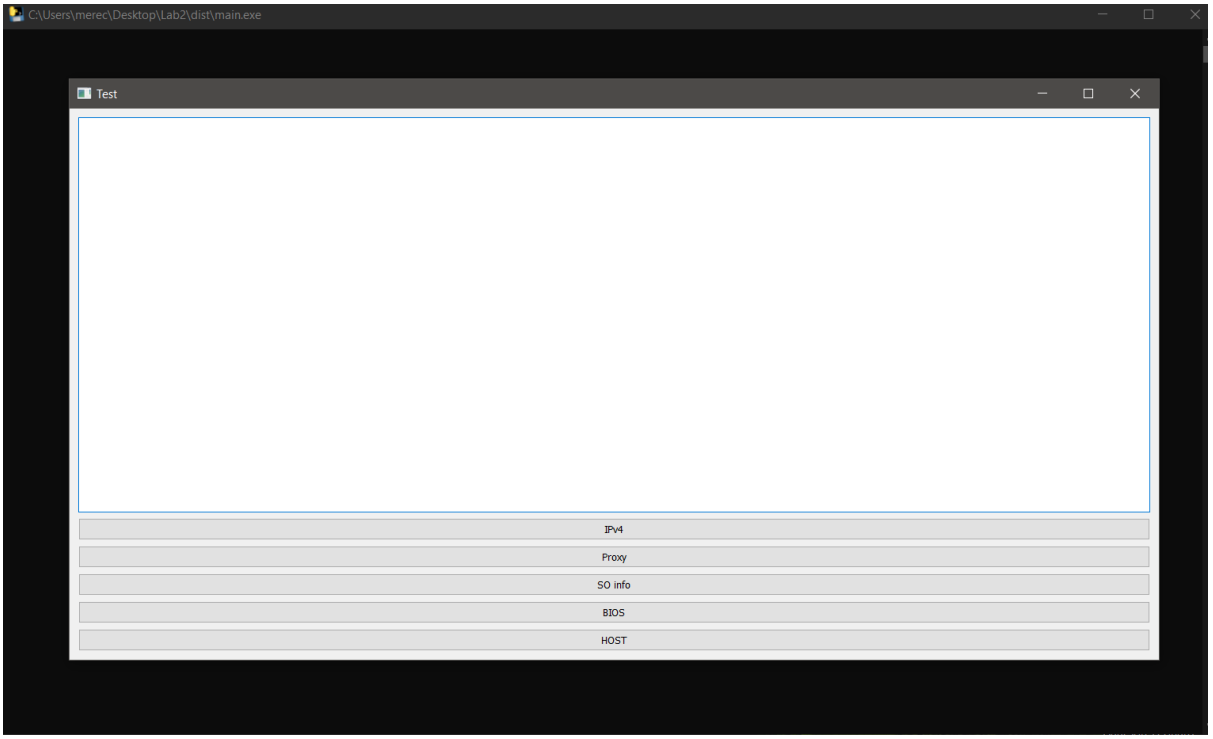
   Linux:



WYNIK: Zgodnie z oczekiwaniami

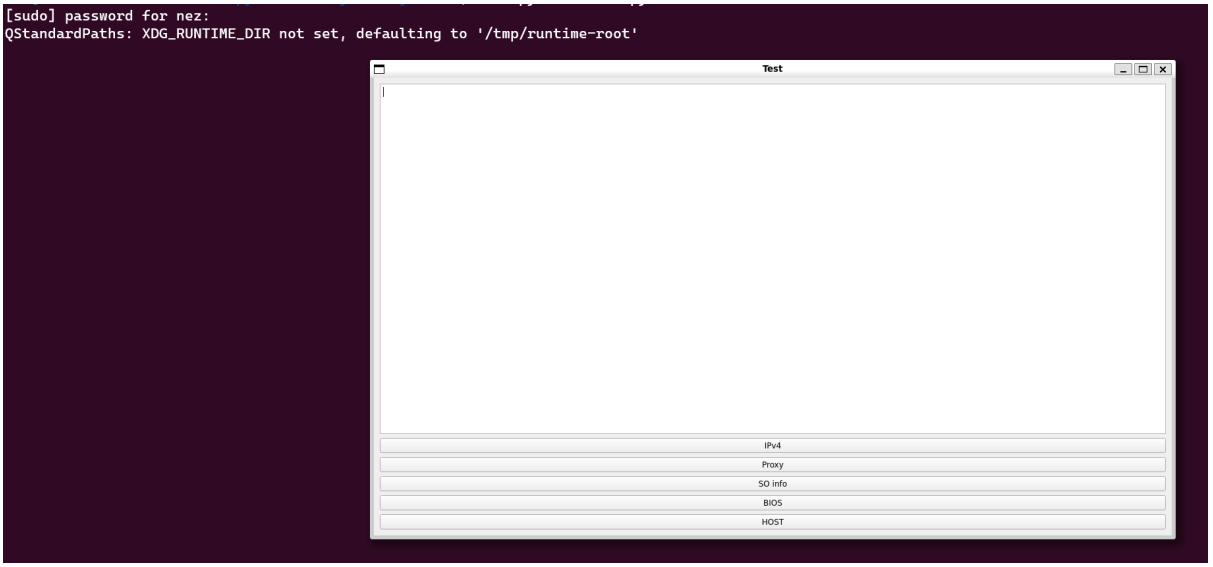
2. Sprawdzenie, czy działa dostosowanie rozmiaru okna do indywidualnych oczekiwań użytkownika.

   Windows:



WYNIK: Zgodnie z oczekiwaniami

   Linux:



WYNIK: Zgodnie z oczekiwaniami

3. Sprawdzenie, czy działają przyciski paska tytułu okna programu.

   Windows:

→ Przycisk powoduje ukrycie okna, na pasku startowym pojawia się ikona programu.

WYNIK: Zgodnie z oczekiwaniami

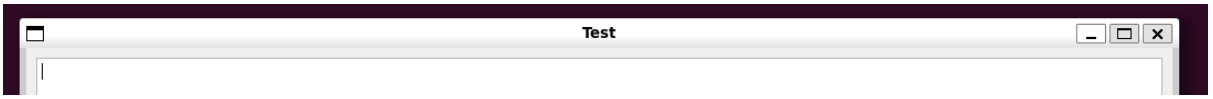
→ Przycisk powoduje maksymalizację okna okna, okno zajmuje 100% przestrzeni pulpitu.

WYNIK: Zgodnie z oczekiwaniami

→ Przycisk powoduje zamknięcie okna oraz zamknięcie procesu systemowego przypisanego do programu, na pasku startowym brak ikona programu.

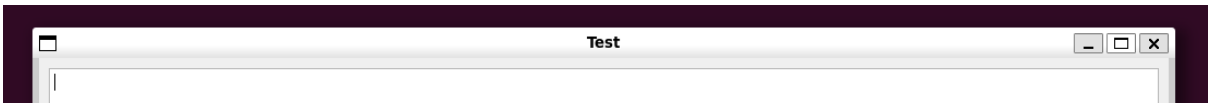
WYNIK: Zgodnie z oczekiwaniami

   Linux:



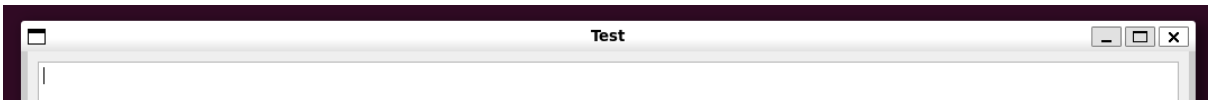
→ Przycisk powoduje ukrycie okna, na pasku startowym pojawia się ikona programu.

WYNIK: Zgodnie z oczekiwaniami



→ Przycisk powoduje maksymalizację okna okna, okno zajmuje 100% przestrzeni pulpitu.

WYNIK: Zgodnie z oczekiwaniami

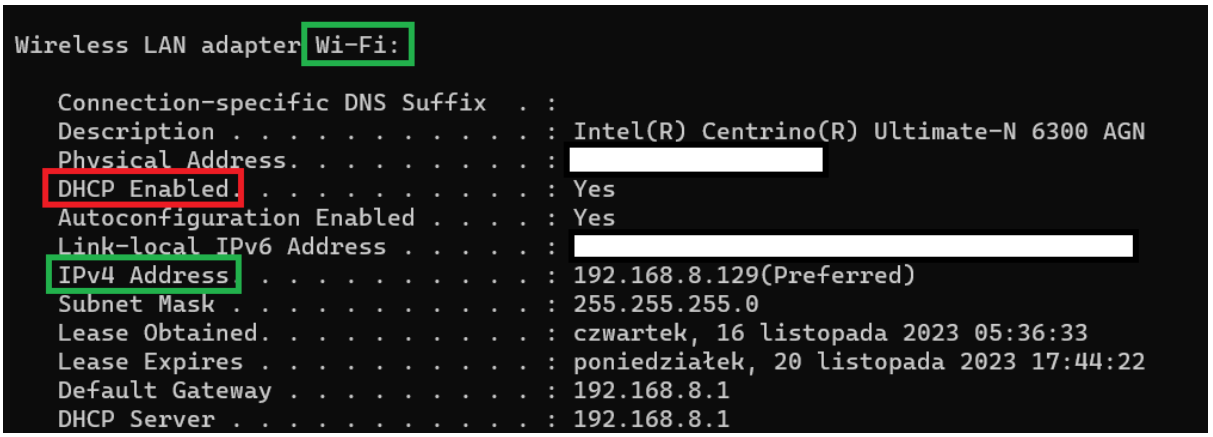


→ Przycisk powoduje zamknięcie okna oraz zamknięcie procesu systemowego przypisanego do programu, na pasku startowym brak ikona programu.

WYNIK: Zgodnie z oczekiwaniami

4. Sprawdzenie, czy program poprawnie wyświetla adres IP użytkownika.
5. Sprawdzenie, czy program poprawnie wyświetla typ przydzielenia adresu IP.
6. Sprawdzenie, czy program poprawnie wyświetla typ połączenia.

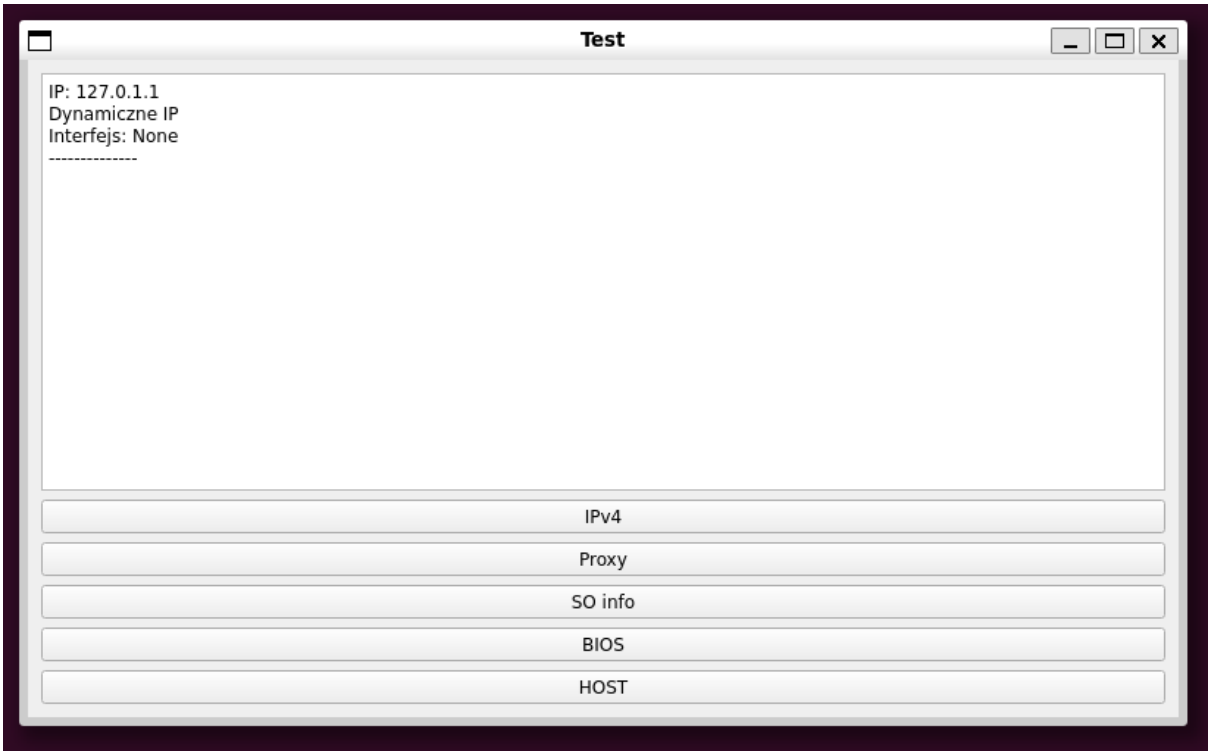
Windows:



WYNIK: NIE zgodnie z oczekiwaniami

Linux:

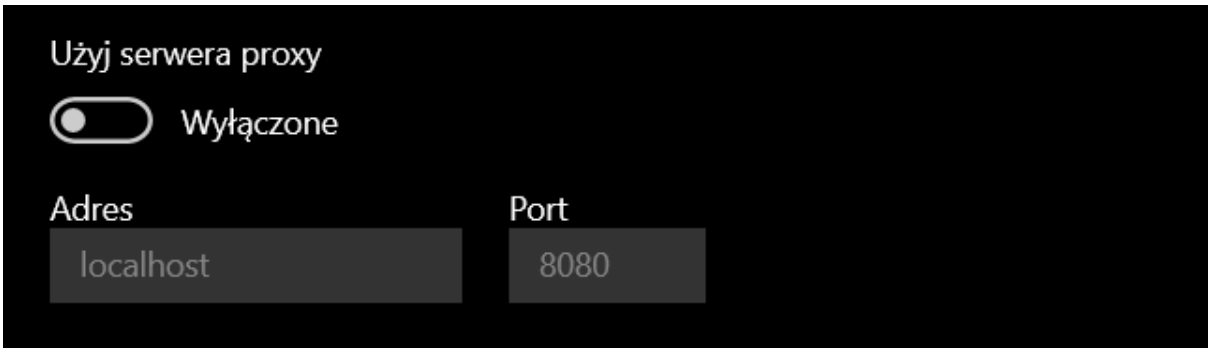
```
nez@DESKTOP-S0RKCER:~$ ip -br addr show
lo               UNKNOWN        127.0.0.1/8 ::1/128
eth0             UP             172.28.237.217/20 [redacted]
docker0          DOWN           172.17.0.1/16
nez@DESKTOP-S0RKCER:~$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether [redacted] brd ff:ff:ff:ff:ff:ff
    inet 172.28.237.217/20 brd 172.28.239.255 scope global eth0
```

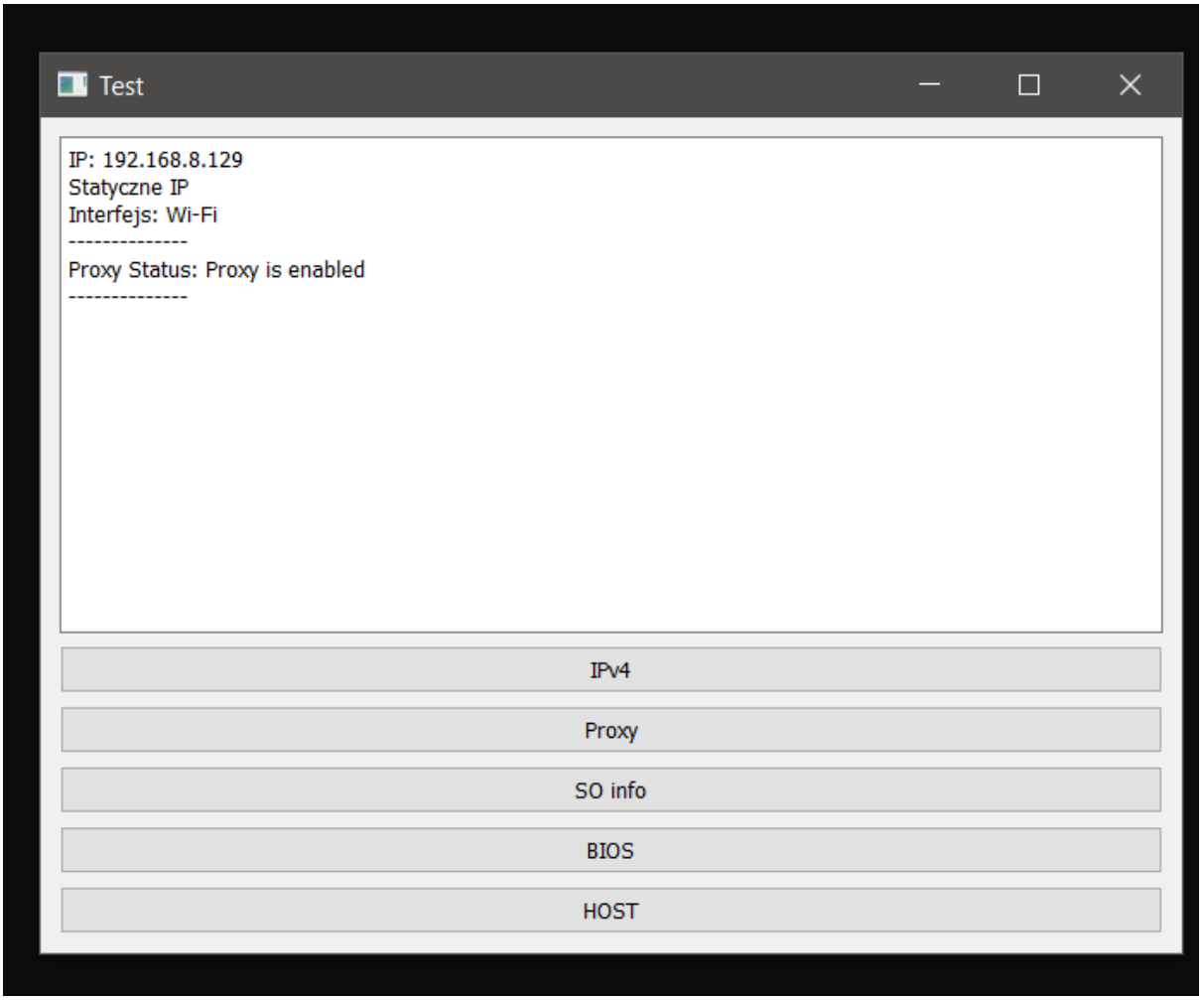


WYNIK: NIE zgodnie z oczekiwaniami

7. Sprawdzenie, czy program poprawnie wyświetla informacje o uruchomionym proxy.

   Windows:

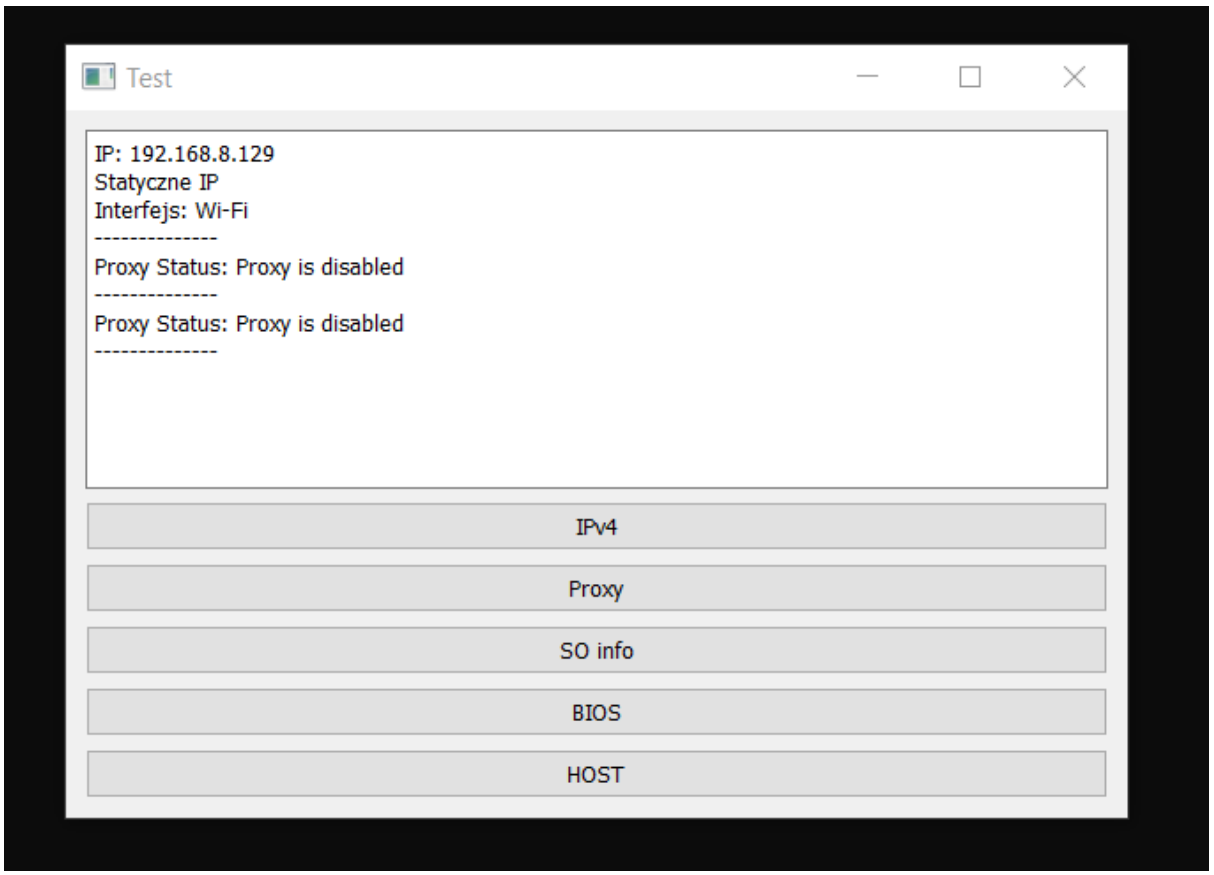



Test
IP: 192.168.8.129
Statyczne IP
Interfejs: Wi-Fi

Proxy Status: Proxy is enabled

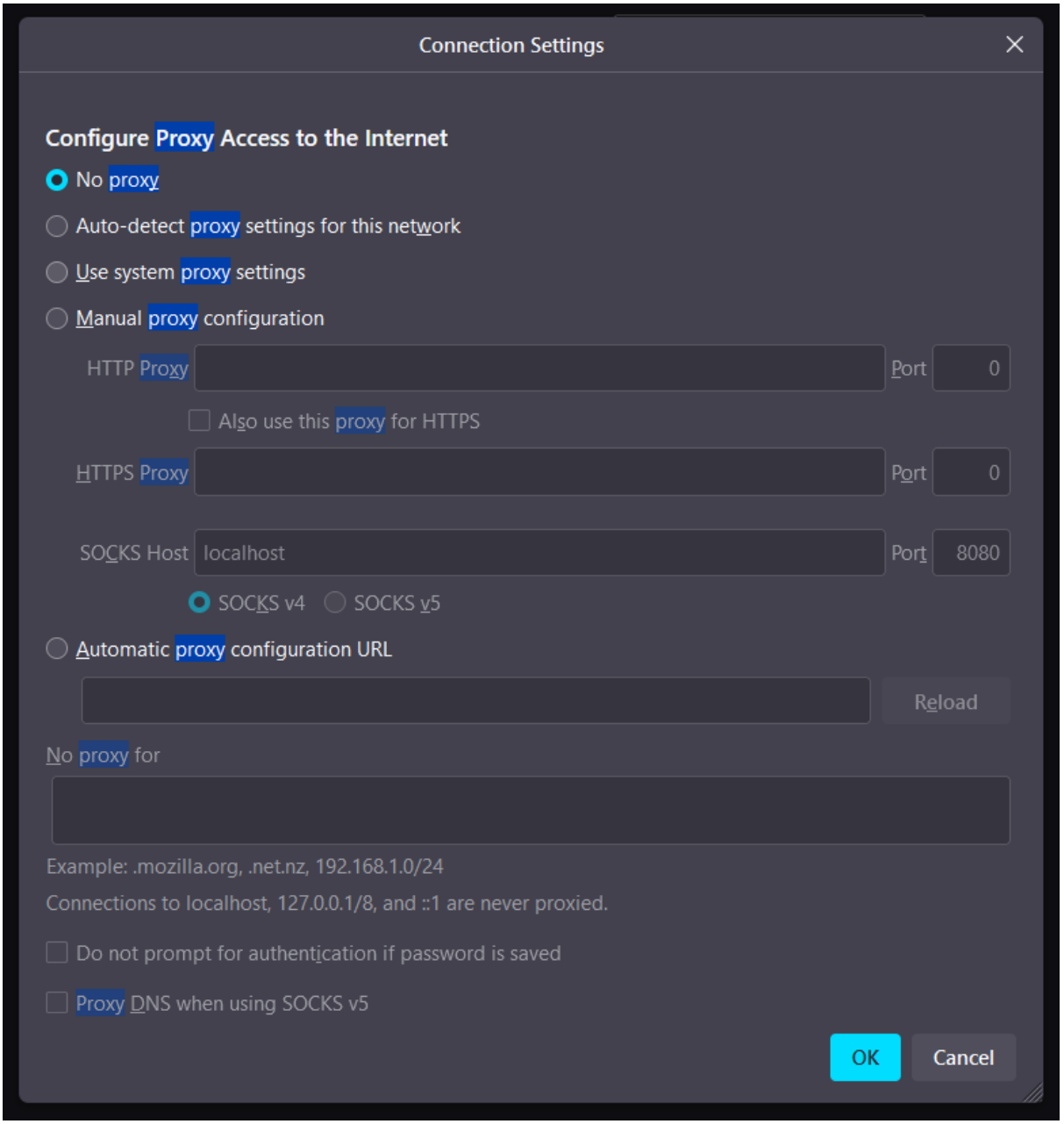
IPv4
Proxy
SO info
BIOS
HOST

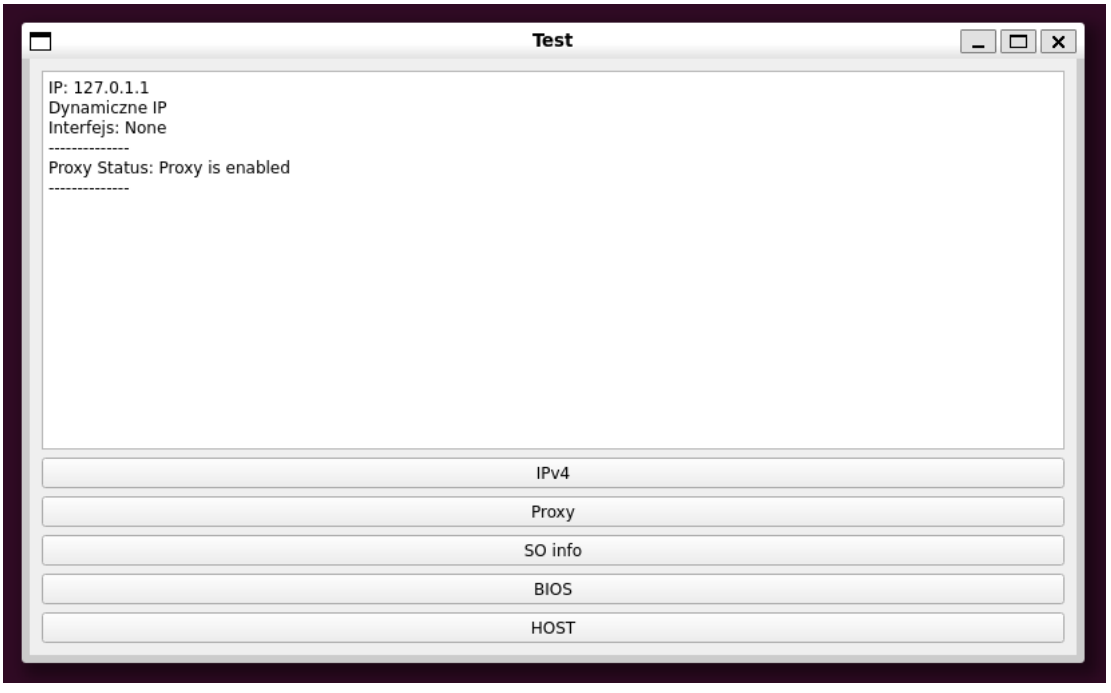

Użyj serwera proxy
Włączone
Adres
localhost
Port
8080



WYNIK: NIE zgodnie z oczekiwaniami

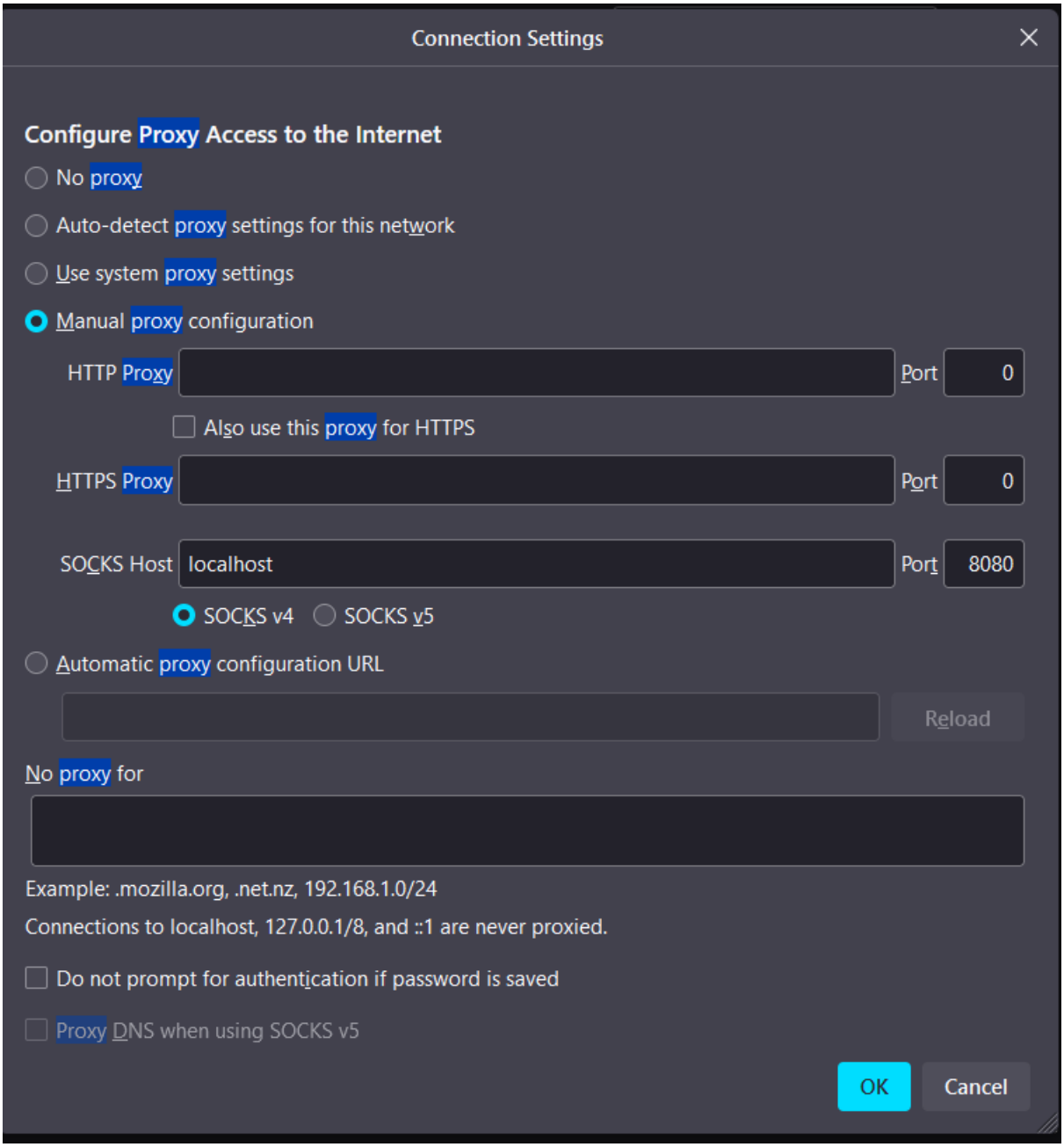
Linux:

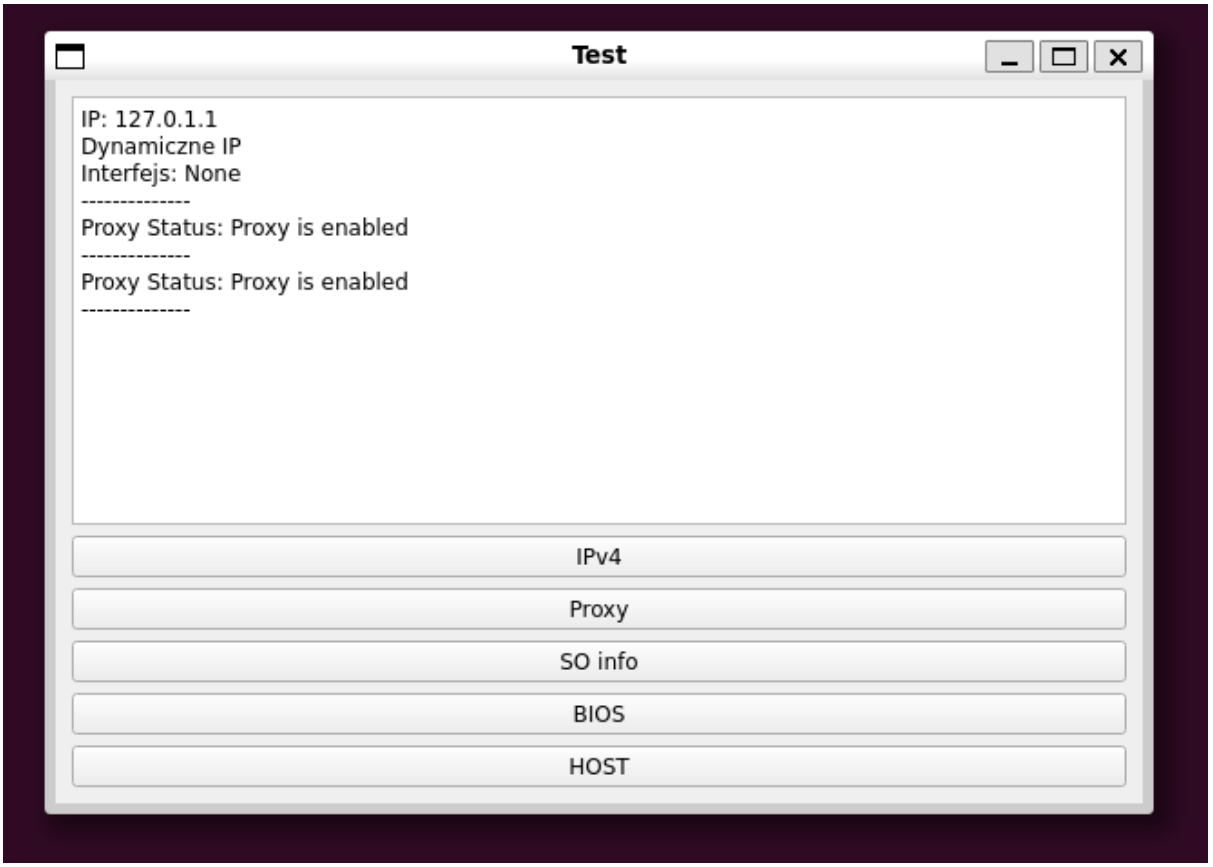

Connection Settings
Configure Proxy Access to the Internet
No proxy
Auto-detect proxy settings for this network
Use system proxy settings
Manual proxy configuration
HTTP Proxy
Port
0
Also use this proxy for HTTPS
HTTPS Proxy
Port
0
SOCKS Host
localhost
Port
8080
SOCKS v4
SOCKS v5
Automatic proxy configuration URL
Reload
No proxy for
Example: .mozilla.org, .net.nz, 192.168.1.0/24
Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.
Do not prompt for authentication if password is saved
Proxy DNS when using SOCKS v5
OK
Cancel


Test
IP: 127.0.1.1
Dynamiczne IP
Interfejs: None

Proxy Status: Proxy is enabled

IPv4
Proxy
SO info
BIOS
HOST

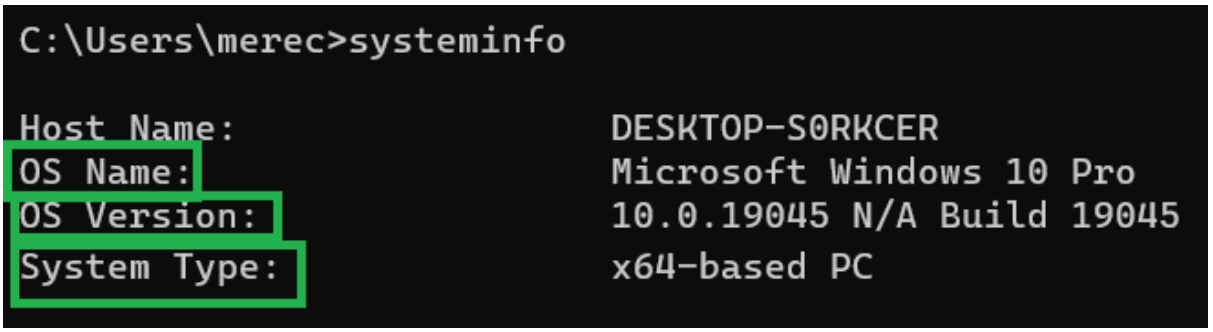

Connection Settings
Configure Proxy Access to the Internet
No proxy
Auto-detect proxy settings for this network
Use system proxy settings
Manual proxy configuration
HTTP Proxy
Port
0
Also use this proxy for HTTPS
HTTPS Proxy
Port
0
SOCKS Host
localhost
Port
8080
SOCKS v4
SOCKS v5
Automatic proxy configuration URL
Reload
No proxy for
Example: .mozilla.org, .net.nz, 192.168.1.0/24
Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.
Do not prompt for authentication if password is saved
Proxy DNS when using SOCKS v5
OK
Cancel

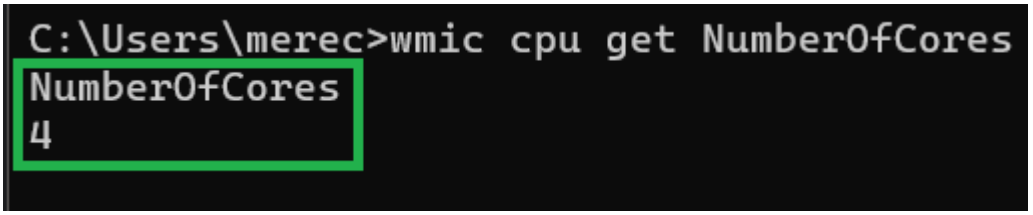


WYNIK: NIE zgodnie z oczekiwaniami

8. Sprawdzenie, czy program poprawnie wyświetla wersję systemu operacyjnego.
9. Sprawdzenie, czy program poprawnie wyświetla liczbę rdzeni i pamięć RAM.

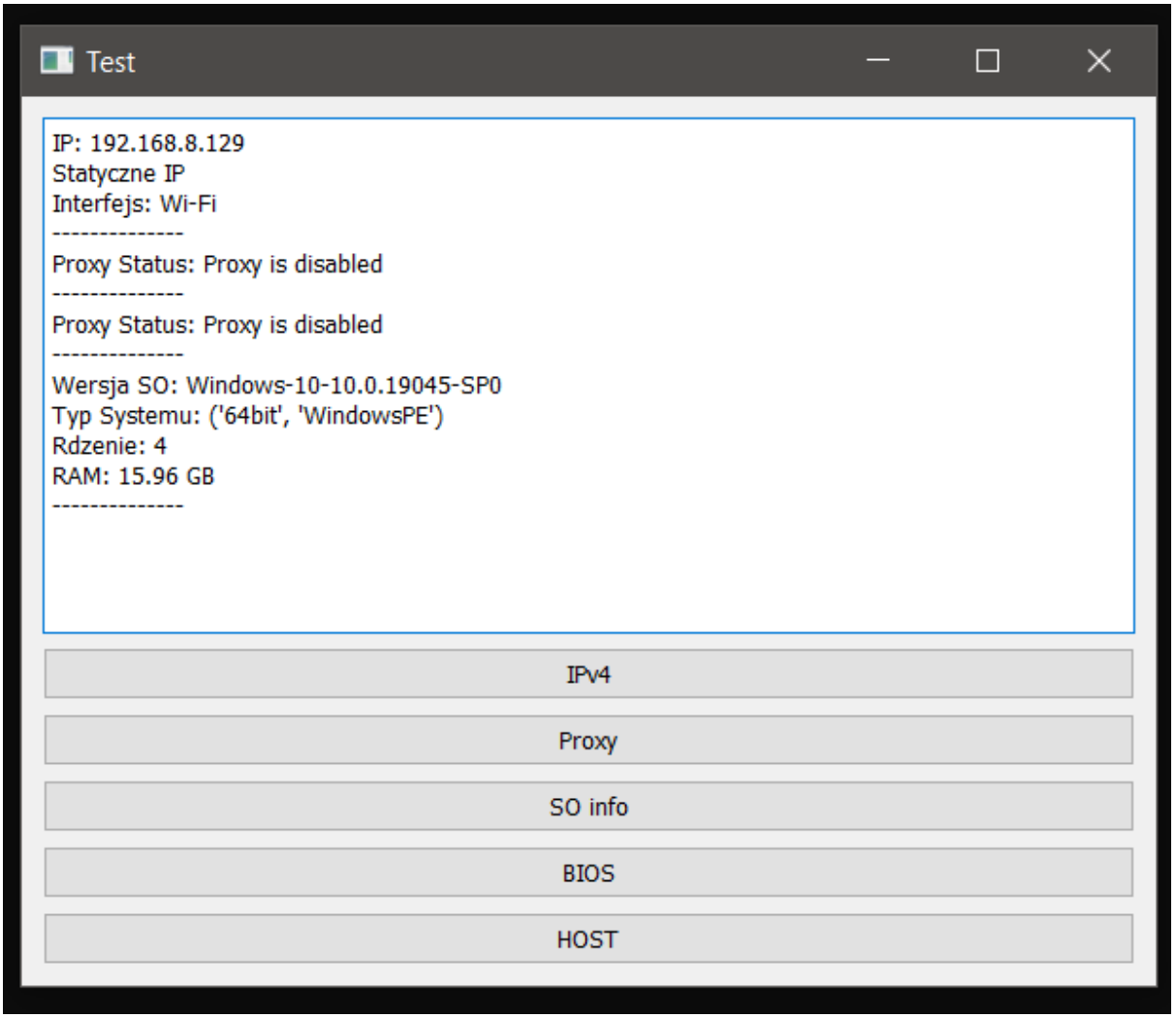
   Windows:





```
C:\Users\merec>systeminfo | findstr /C:"Total Physical Memory"
Total Physical Memory:     16 341 MB
```

```
C:\Users\merec>wmic ComputerSystem get TotalPhysicalMemory
TotalPhysicalMemory
17134448640
```



Powodem jest zastosowanie w kodzie zaokrąglenia do 2 miejsca po przecinku, w przypadku Windows uważam to za błąd ponieważ żadne wypisanie informacji w systemie (graficzne / w konsoli) nie podaje wyniku w takiej formie.

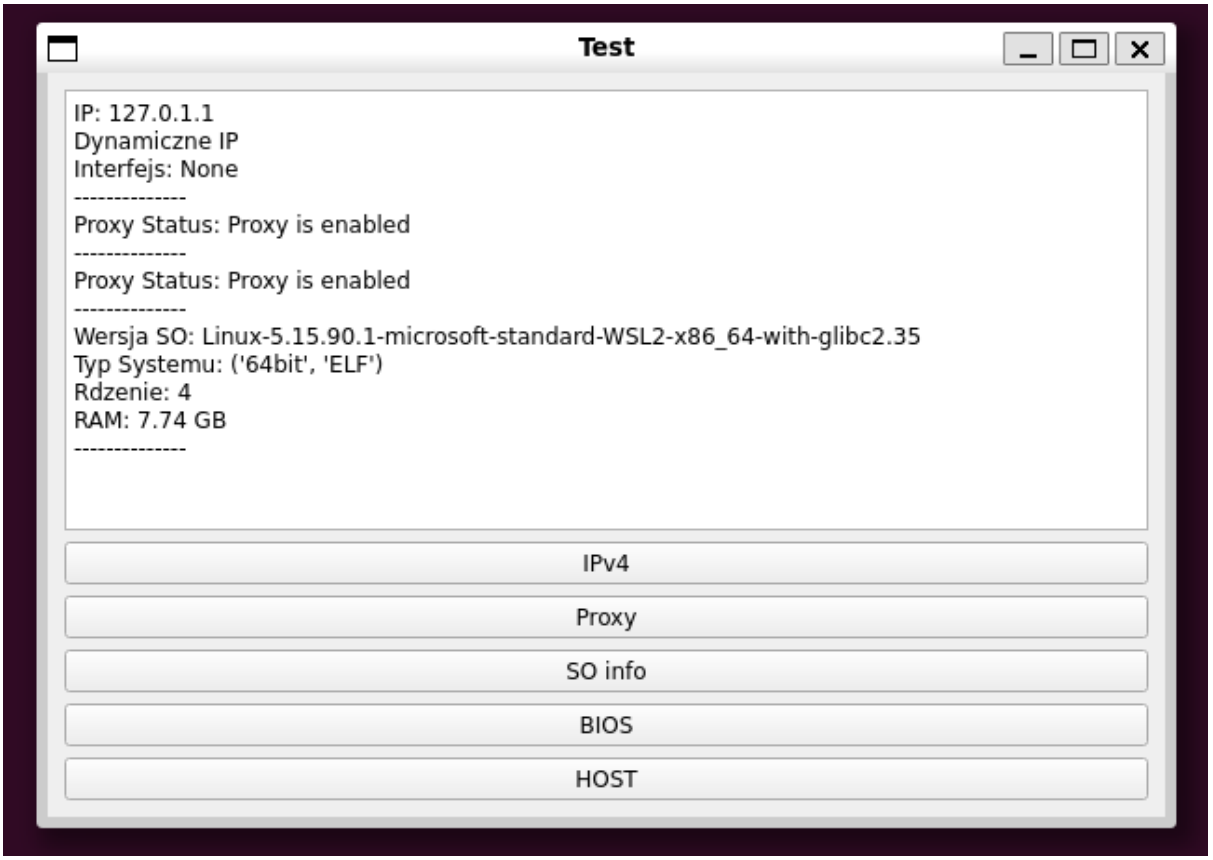
WYNIK: NIE zgodnie z oczekiwaniami

Linux:

```
nez@DESKTOP-S0RKCER:~$ uname -a
Linux DESKTOP-S0RKCER 5.15.90.1-microsoft-standard-WSL2 #1 SMP Fri Jan 27 02:56:13 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

```
nez@DESKTOP-S0RKCER:~$ iscpu
Command 'iscpu' not found, did you mean:
  command 'lscpu' from deb util-linux (2.37.2-4ubuntu3)
Try: sudo apt install <deb name>
nez@DESKTOP-S0RKCER:~$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         36 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  8
  On-line CPU(s) list:   0-7
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-2720QM CPU @ 2.20GHz
    CPU family:          6
    Model:               42
    Thread(s) per core:  2
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            7
    BogoMIPS:            4390.03
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp
                         lm constant_tsc arch_perfmon rep_good nopl xtopology cpuid pni pclmulqdq ssse3 cx16 pdcm pcid sse4_1 sse4_2 popcnt aes
                         xsave avx hypervisor lahf_lm pti ssbd ibrs ibpb stibp xsaveopt flush_l1d arch_capabilities
Virtualization features:
  Hypervisor vendor:     Microsoft
  Virtualization type:   full
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
```

```
nez@DESKTOP-S0RKCER:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           7.7Gi       3.1Gi       217Mi        28Mi       4.4Gi       4.3Gi
Swap:          2.0Gi        34Mi       2.0Gi
```

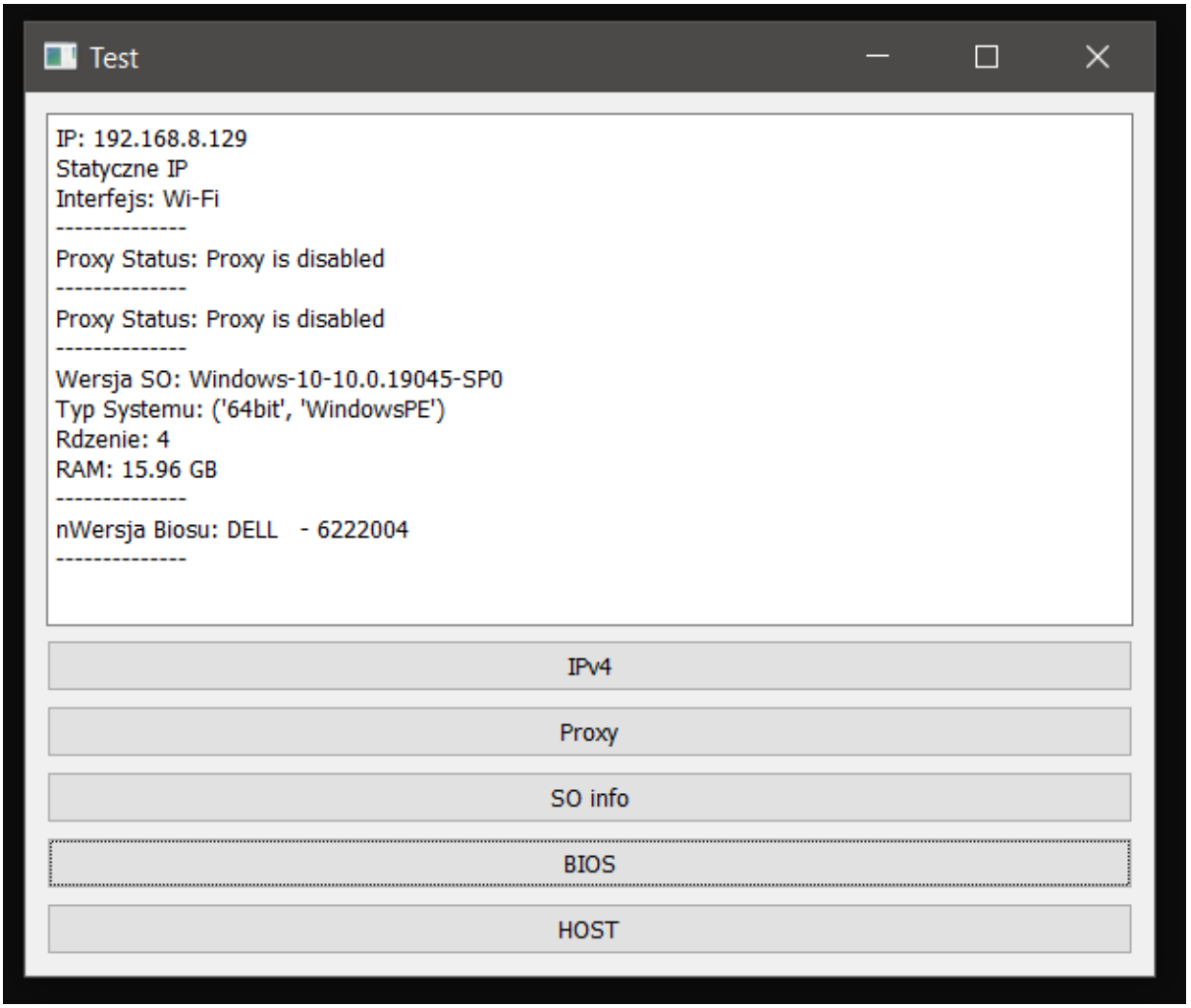


WYNIK: Zgodnie z oczekiwaniami

10. Sprawdzenie, czy program poprawnie wyświetla wersję BIOS.

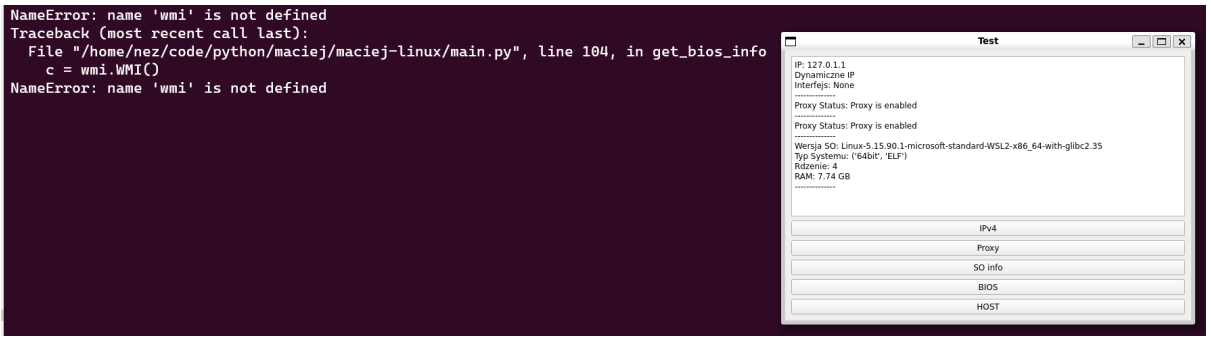
    Windows:

```
C:\Users\merec>wmic bios get smbiosbiosversion
SMBIOSBIOSVersion
A19
```



WYNIK: NIE zgodnie z oczekiwaniami

Linux:

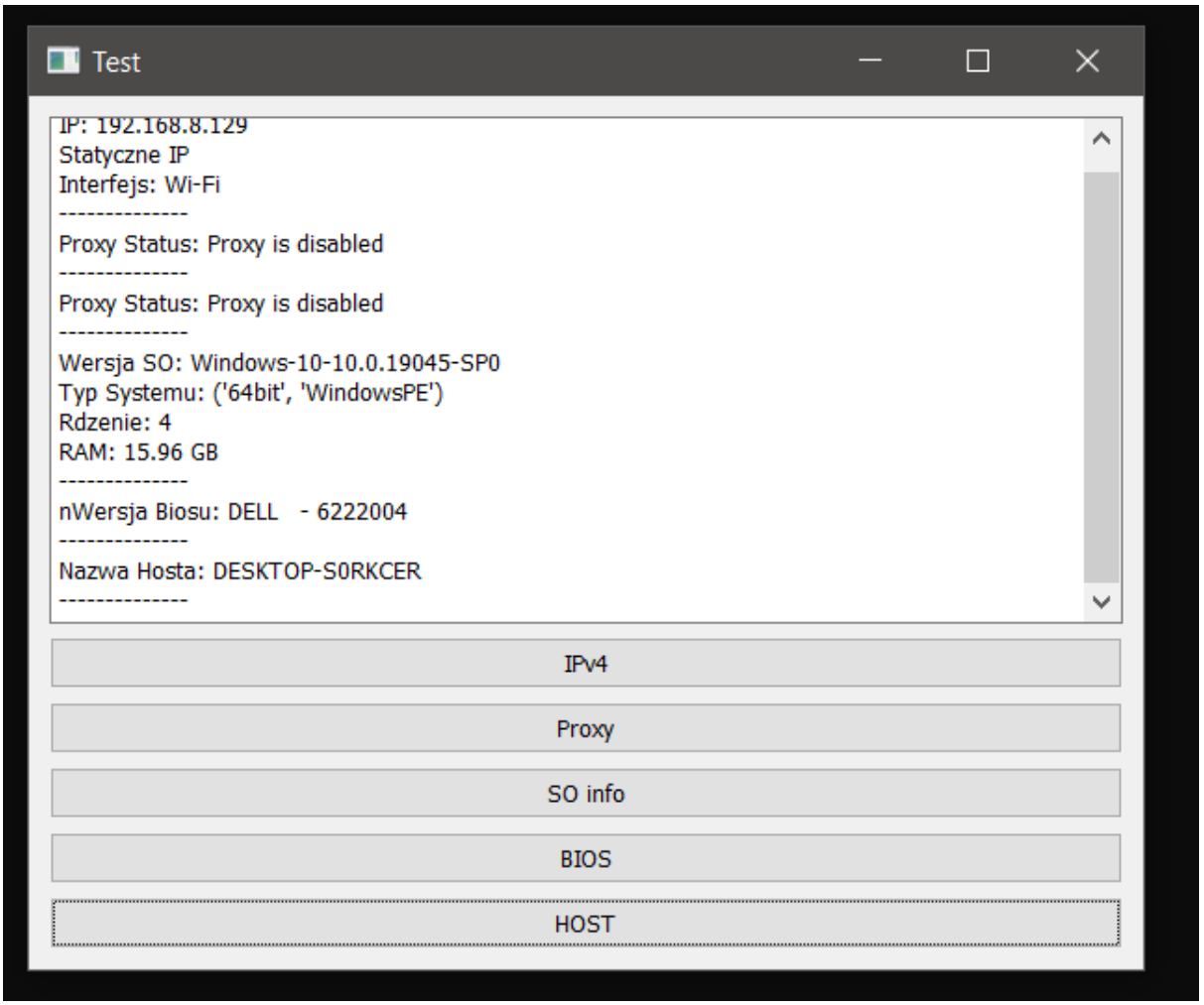


Program powstał z myślą o systemie Windows, moduł ‘wmi’ nie działa na systemie Linux.

WYNIK: NIE zgodnie z oczekiwaniami

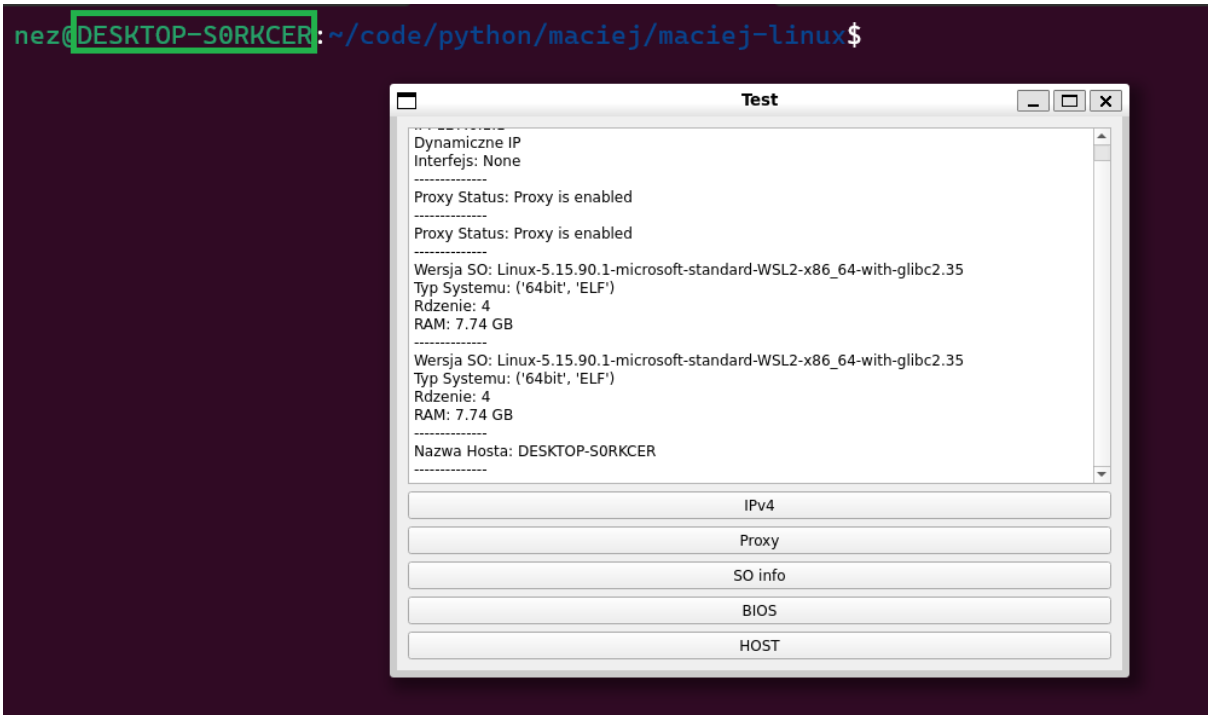
11. Sprawdzenie, czy program poprawnie wyświetla nazwę hosta.

    Windows:



WYNIK: Zgodnie z oczekiwaniami

    Linux:



WYNIK: Zgodnie z oczekiwaniami

12. Sprawdzenie, czy program obsługuje wszystkie komendy z wiersza poleceń.

    Windows:

Nie obsługuje komend z wiersza poleceń.

WYNIK: NIE zgodnie z oczekiwaniami

    Linux:

Nie obsługuje komend z wiersza poleceń.

WYNIK: NIE zgodnie z oczekiwaniami

13. Sprawdzenie, czy informacje uzyskane z wiersza poleceń są identyczne z tymi uzyskanymi z GUI.

    Windows:

Brak zaimplementowanej funkcjonalności obsługi komend z wiersza poleceń.

WYNIK: NIE zgodnie z oczekiwaniami

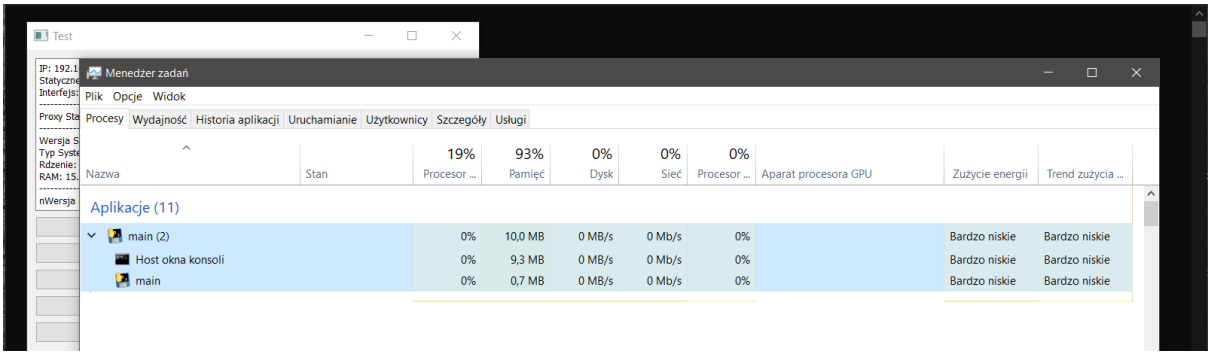
    Linux:

Brak zaimplementowanej funkcjonalności obsługi komend z wiersza poleceń.

WYNIK: NIE zgodnie z oczekiwaniami

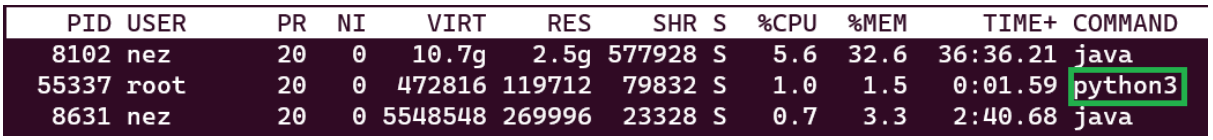
14. Program powinien działać szybko i nie obciążać zbytnio zasobów komputera → maksymalne obciążenie dla systemu Windows to ‘Umiarkowany’ lub Linux to poniżej 2.5 %CPU.

    Windows:



WYNIK: Zgodnie z oczekiwaniami

    Linux:



WYNIK: Zgodnie z oczekiwaniami

15. Program powinien być łatwy w instalacji i użytkowaniu, a jego interfejs powinien być intuicyjny i czytelny.

    Windows:

Program odpalany jest za pomocą podwójnego kliknięcia jest to rozwiązanie bardzo intuicyjne dla użytkowników Windows. Konstrukcja okienkowa jest wzorowana na aplikacjach windowsowych.

WYNIK: Zgodnie z oczekiwaniami

    Linux:

Program wykorzystuje graficzne środowisko Linuxa wzorowane na łatwym i intuicyjnym mechanizmie Windowsa.

WYNIK: Zgodnie z oczekiwaniami

16. Program powinien być odporny na przewidywane błędy i awarie, a w razie ich wystąpienia powinien informować użytkownika o przyczynie i sposobie rozwiązania problemu.

    Windows:

Użytkownik otrzymuje jedynie błędy systemowe.

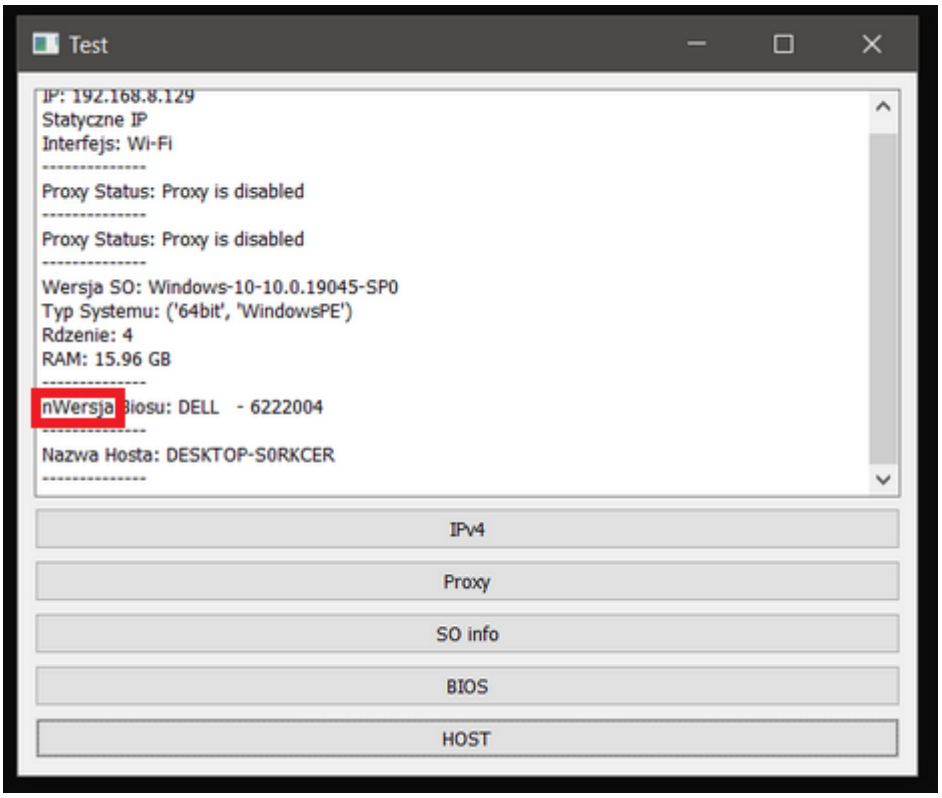
WYNIK: NIE zgodnie z oczekiwaniami

Linux:

Użytkownik otrzymuje jedynie błędy systemowe.

WYNIK: NIE zgodnie z oczekiwaniami

17. Program nie powinien zawierać literówek, błędów ortograficznych lub fleksyjnych.



Windows:

Po usunięciu literówki można zmienić status na zaakceptowany.

WYNIK: NIE zgodnie z oczekiwaniami

Linux:

Brak możliwości przetestowania - błędna implementacja funkcjonalności.

WYNIK: NIE zgodnie z oczekiwaniami

WYNIKI POMIARÓW

Przypadek testowy	System	Stopień koniecznych poprawek	Wynik
1	Windows	Zgodnie z wytycz…	Zatwierd…
1	Linux	Zgodnie z wytycz…	Zatwierd…
2	Windows	Zgodnie z wytycz…	Zatwierd…
2	Linux	Zgodnie z wytycz…	Zatwierd…
3	Windows	Zgodnie z wytycz…	Zatwierd…
3	Linux	Zgodnie z wytycz…	Zatwierd…
4	Windows	Błędna implemen…	Odrzucono
4	Linux	Błędna implemen…	Odrzucono
5	Windows	Błędna implemen…	Odrzucono
5	Linux	Błędna implemen…	Odrzucono
6	Windows	Błędna implemen…	Odrzucono
6	Linux	Błędna implemen…	Odrzucono
7	Windows	Błędna implemen…	Odrzucono
7	Linux	Błędna implemen…	Odrzucono
8	Windows	Zgodnie z wytycz…	Zatwierd…
8	Linux	Zgodnie z wytycz…	Zatwierd…
9	Windows	Błędna implemen…	Odrzucono
9	Linux	Zgodnie z wytycz…	Zatwierd…
10	Windows	Błędna implemen…	Odrzucono
10	Linux	Błędna implemen…	Odrzucono
11	Windows	Zgodnie z wytycz…	Zatwierd…

Przypadek testowy	System	Stopień koniecznych poprawek	Wynik
11	Linux	Zgodnie z wytycz…	Zatwierd…
12	Windows	Błędna implemen…	Odrzucono
12	Windows	Błędna implemen…	Odrzucono
13	Linux	Błędna implemen…	Odrzucono
13	Windows	Błędna implemen…	Odrzucono
14	Linux	Zgodnie z wytycz…	Zatwierd…
14	Windows	Zgodnie z wytycz…	Zatwierd…
15	Linux	Zgodnie z wytycz…	Zatwierd…
15	Windows	Zgodnie z wytycz…	Zatwierd…
16	Linux	Błędna implemen…	Odrzucono
16	Windows	Błędna implemen…	Odrzucono
17	Linux	Drobna korekta	Odrzucono
17	Windows	Drobna korekta	Odrzucono

WNIOSKI PODSUMOWUJĄCE

Wyniki testu manualnego na środowisku Linux są negatywne. Program nie uruchamia się na systemach innej dystrybucji niż Windows, z powodu braku modułu WMI”

```
File "/home/user/Downloads/maciej (1)/maciej-linux/main.py", line 8, in <module>
    import wmi
ModuleNotFoundError: No module named 'wmi '
```

Moduł WMI (Windows Management Instrumentation) w systemie Windows odpowiada za zarządzanie i monitorowanie zasobami, aplikacjami oraz usługami na komputerze.

Dodatkowo wykryto problemy z zestawem znaków UTF-8
SyntaxError: Non-UTF-8 code starting with '\xcd' in file /home/blackmen/Downloads/maciej (1)/maciej-linux/dist/main on line 2, but no encoding declared; see https://python.org/dev/peps/pep-0263/ for details

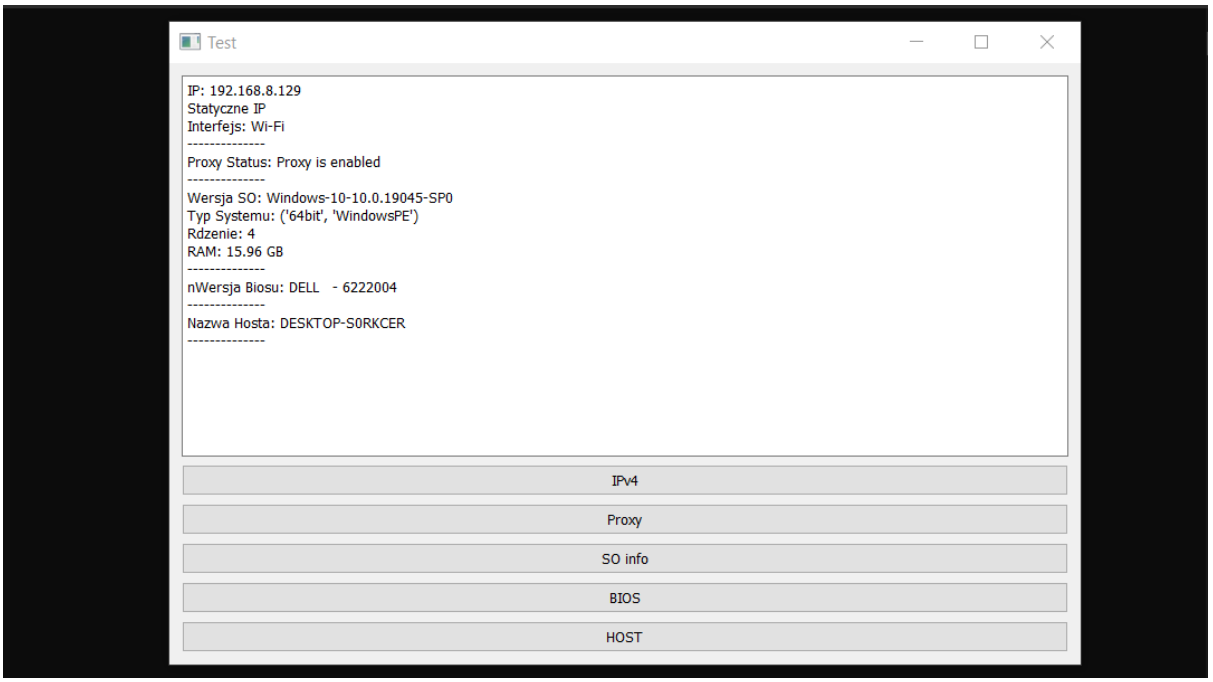
Obydwa problemy można rozwiązać komentując w pliku MyTest.py wartości sprzeczne tj.

A) dodając informacje o deklaracji zestawu znaków # -*- coding: utf-8 -*-
B) komentując moduł WMI

Po zastosowaniu powyższych rozwiązań program uruchamia się, jednak występują błędy techniczne tj. get_ipv4_info pokazuje błędny adres, dodatkowo get_proxy_info zwraca błędną wartość. Funkcje get_system_info, get_bios_info oraz get_hostname działają zgodne z oczekiwanymi założeniami.

Program działa poprawnie na systemie Windows. Funkcje zostały przetestowane z uprawnieniami administratora, bez uprawnień również nie wykryto błędów.

Widoczne są błędy wizualne jak dodatkowa litera N przy wersji biosu.



Zauważalnym jest fakt nieścisłości niektórych danych z powodu zaokrąglania wyników w programie otrzymanym do testowania. W następstwie analizy kodu źródłowego owe nieścisłości nie uznaje jako błąd, lecz przyjęcie innej formy wartości parametrów przy założeniach programu.

Agnieszka Iżuk 43120
DSW University of Lower Silesia
Github: https://github.com/student43120/Py_Test/tree/main