

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Кафедра програмних засобів

(найменування кафедри)

КУРСОВИЙ ПРОЄКТ  
(РОБОТА)

з дисципліни «Операційні системи»

(назва дисципліни)

на тему: «Застосунок аналізу системних ресурсів»

Студента 3 курсу КНТ-122 групи  
спеціальності 121 Інженерія  
програмного забезпечення  
освітня програма (спеціалізація)  
інженерія програмного забезпечення  
Онищенко О. А.

(прізвище та ініціали)  
(прізвище та ініціали)

Керівник Степаненко О. О.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_  
Кількість балів: \_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_

Члени комісії

|                         |                        |
|-------------------------|------------------------|
| <u>Степаненко О. О.</u> | (прізвище та ініціали) |
| <u>(підпис)</u>         | (прізвище та ініціали) |
| <u>(підпис)</u>         | (прізвище та ініціали) |

2024 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»  
(повне найменування закладу вищої освіти)

Інститут, факультет ПРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н., проф.

**С.О. Субботін**

“ \_\_\_\_ ” 20 \_\_\_\_ року

**З А В Д А Н Н Я**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА**

Онишенко О. А

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Застосунок аналізу системних ресурсів  
керівник проекту (роботи) Степаненко Олександр Олексійович,  
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)  
затверджені наказом закладу вищої освіти від \_\_\_\_\_
2. Срок подання студентом проекту (роботи) 03 грудня 2024 року
3. Вихідні дані до проекту (роботи) створити застосунок аналізу системних ресурсів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області; 2. Аналіз програмних засобів;  
3. Основні рішення з реалізації компонентів системи; 4. Керівництво  
програміста; 5. Керівництво користувача; 6. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | Прізвище, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|---|----------------|---------------------------|
|                     |   | завдання видав | прийняв виконане завдання |
| 1-5 Основна частина | Степаненко О. О.                          |                |                           |
|                     |   |                |                           |
|                     |   |                |                           |

7. Дата видачі завдання 16 жовтня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів курсового проєкту (роботи)  | Срок виконання етапів проєкту (роботи ) | Примітка          |
|-------|--|---|-------------------|
| 1.    | Аналіз індивідуального завдання.   | 1 тиждень                               |                   |
| 2.    | Аналіз програмних засобів, що будуть використовуватись в роботі.                     | 2 тиждень                               |                   |
| 3.    | Аналіз структур даних, що необхідно використати в курсовій роботі.                   | 3 тиждень                               |                   |
| 4.    | Затвердження завдання  | 4 тиждень                               |                   |
| 5.    | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-9 тиждень                             |                   |
| 6.    | Аналіз вимог до апаратних засобів  | 9 тиждень                               |                   |
| 7.    | Розробка програмного забезпечення  | 9-13 тиждень                            |                   |
| 8.    | Проміжний контроль   | 10 тиждень                              | Розділи 1-2<br>ПЗ |
| 9.    | Оформлення, відповідних пунктів пояснлюальної записки.                               | 10-14 тиждень                           | Розділи 1-5<br>ПЗ |
| 10.   | Захист курсової роботи.  | 15 тиждень                              |                   |
|       |  |   |                   |
|       |  |   |                   |

Студент Онищенко О. А.  
( підпис ) (прізвище та ініціали)

Керівник проєкту (роботи) Степаненко О. О.  
( підпис ) (прізвище та ініціали)

## РЕФЕРАТ

Проект «Застосунок аналізу системних ресурсів» є комплексним застосунком, розробленим з використанням мови програмування Python. Застосунок призначений для полегшення процесу отримання актуальних та оновлювальних даних про поточні ресурси системи користувача у цифровому вигляді. Це зручна платформа, яка дозволяє користувачам переглядати інформацію про систему, її поточні характеристики та властивості, а також показники роботи системи на конкретний момент часу. Застосунок також дозволяє отримати візуальне представлення зміни показників системи, що може полегшити сприйняття користувачем відповідних параметрів системи при аналізі системних ресурсів. Проект має на меті спростити процес витягання та взаємодії із ресурсами системи користувача за допомогою візуальних елементів інтерфейсу.

Система, ресурси системи, Python, курсова робота, курсовий проект, Windows застосунок, застосунок аналізу системних ресурсів.

## ЗМІСТ

|   |    |
|---|----|
| РЕФЕРАТ .....   | 4  |
| ЗМІСТ .....   | 5  |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ<br>І ТЕРМІНІВ.....  | 7  |
| ВСТУП .....   | 8  |
| 1 ОГЛЯД І АНАЛІЗ ТЕМАТИКИ .....   | 9  |
| 1.1 Аналіз витягання системних ресурсів, як основи предметної області ..... | 9  |
| 1.2 Огляд існуючих методів вирішення завдання.....                          | 10 |
| 1.2.1 Передмова .....   | 10 |
| 1.2.2 Система «Cpu-Z» .....   | 10 |
| 1.2.3 Система «Open Hardware Monitor» .....                                 | 13 |
| 1.2.4 Висновки огляду існуючих застосунків.....                             | 16 |
| 2 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ.....   | 17 |
| 3 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ.....                             | 20 |
| 3.1 Системні дані.....  | 20 |
| 3.2 Інформація про процесор .....   | 20 |
| 3.2.1 Теоретична інформація .....   | 21 |
| 3.2.2 Показники компонентів.....  | 21 |
| 3.3 Інформація про пам'ять.....   | 22 |
| 3.3.1 Оперативна пам'ять .....  | 23 |
| 3.3.2 Пам'ять підкачки.....   | 23 |
| 3.4 Накопичувачі .....  | 23 |
| 3.4.1 Детальна інформація.....  | 24 |
| 3.4.1 Загальна інформація.....  | 25 |
| 3.5 Мережа .....  | 25 |
| 3.5.1 Детальна інформація.....  | 25 |
| 3.5.1 Загальна інформація.....  | 26 |
| 3.6 Інші відомості.....   | 26 |
| 3.6.1 Мова інтерфейсу.....  | 26 |
| 3.6.2 Структура інтерфейсу .....  | 27 |
| 4 ПРОГРАМНА РЕАЛІЗАЦІЯ.....   | 28 |
| 4.1 Вибір засобів .....   | 28 |
| 4.2 Компоновка проекту .....  | 28 |

|  |           |
|--|-----------|
| <b>5 КЕРІВНИЦТВО ОПЕРАТОРА .....</b>                                     | <b>32</b> |
| 5.1 Встановлення Python .....  | 32        |
| 5.2 Завантаження коду .....  | 32        |
| 5.3 Встановлення пакетів .....   | 33        |
| 5.4 Запуск програми.....   | 33        |
| <b>6 КЕРІВНИЦТВО ПРОГРАМІСТА .....</b>                                   | <b>34</b> |
| 6.1 Інструкція з модернізації.....                                       | 34        |
| 6.2 Таблиця функцій .....  | 35        |
| <b>ВИСНОВКИ.....</b>   | <b>36</b> |
| <b>ПЕРЕЛІК ПОСИЛАНЬ .....</b>  | <b>37</b> |
| <b>ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ .....</b>                               | <b>38</b> |
| <b>ДОДАТОК Б – ТЕКСТ ПРОГРАМИ.....</b>                                   | <b>45</b> |
| Б1 – run.py .....  | 45        |
| <b>ДОДАТОК В – МЕТОДИКА ТА РЕЛУЗЛЬТАТИ ТЕСТУВАННЯ<br/>ПРОГРАМИ .....</b> | <b>50</b> |
| В1 Системні дані.....  | 51        |
| В2 Дані про процесор .....   | 52        |
| В3 Дані про пам'ять.....   | 58        |
| В4 Дані про накопичувачі .....   | 61        |
| В5 Інформація про мережу .....   | 66        |
| В6 Графічні елементи.....  | 71        |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

*ОС* – операційна система, системне програмне забезпечення, що надає інтерфейс між користувачем та апаратним забезпеченням системи.

*Пам'ять підкачки* – (Англійською *swap memory*) розділ фіксованого накопичувача системи, що використовується операційною системою для збереження інформації з оперативної пам'яті, коли та вичерпує доступні ресурси.

*NiceGUI* – бібліотека Python для розробки графічних інтерфейсів.

*Python* – мова програмування.

*psutil* – бібліотека Python для доступу до ресурсів системи.

*pywin32* або *win32api* – бібліотека Python для доступу до системних ресурсів.

## **ВСТУП**

Застосунок аналізу системних ресурсів є цифровою платформою надання коректних поточних даних про характеристики користувачької системи та поточних показників конкретних компонентів. Оскільки наявність комп'ютера є корисною можливістю, аналіз ресурсів системи для системних адміністраторів або користувачів, які дбають про систему самостійно може бути корисним інструментом. Такий застосунок може полегшити отримання та опрацювання інформації про поточні характеристики системи та її показники на конкретний момент часу.

# 1 ОГЛЯД І АНАЛІЗ ТЕМАТИКИ

Системні ресурси є частиною системи. Вони відображають характеристики системи, її конкретних компонентів, а також надають інформацію про поточні показники компонентів системи.

## 1.1 Аналіз витягання системних ресурсів, як основи предметної області

Витягання системних ресурсів є основною підзадачею аналізу ресурсів системи користувача. Для витягання ресурсів із системи розробники операційних систем зазвичай надають системні функції та методи для роботи з компонентами системи. Оскільки застосунок розробляється під платформу Windows, програма використовує системні функції та методи, зокрема WMI, для доступу до ресурсів системи, для витягання інформації про компоненти та аналіз їх поточних характеристик.

Для повного та точного надання інформації про системи програма має надавати наступну інформацію:

- Дані про систему;
- Інформацію про процес;
- Дані про пам'ять;
- Інформацію про накопичувачі;
- Дані про мережу.

Дані про пам'ять мають включати підрозділ з інформацією про пам'ять підкачки (Англійською *swap memory*). Пам'ять підкачки є простором на фізичному диску користувача, який використовується операційною системою для збереження інформації з оперативної пам'яті в разі вичерпання її наявних ресурсів.

Для кращого розуміння поставленої задачі та цілей проекту краще розглянути вже існуючі рішення аби зрозуміти можливі функції системи та перейняти досвід інших розробників.

## **1.2 Огляд існуючих методів вирішення завдання**

### ***1.2.1 Передмова***

Оскільки задача аналізу системних ресурсів є необхідною для користувачів які обслуговують системи власноруч, розробники з різних країн зробили подібні системи для витягання та аналізу ресурсів користувальських систем.

Нижче розглянуто кілька аналогічних систем для оцінки вже наявних рішень та можливого перейняття досвіду від інших розробників. Розглянуті системи такі:

- Застосунок CPU-Z;
- Застосунок Open Hardware Monitor.

Для кожної системи надано опис, її переваги та недоліки, та приклади використання системи для оцінки ресурсів персонального комп’ютера.

### ***1.2.2 Система «CPU-Z»***

CPU-Z є безкоштовною програмою для Windows, яка збирає інформацію про основні компоненти системи: процесор, материнська плата, пам'ять. CPU-Z повністю підтримується на Windows 11.

Переваги:

- Детальна та точна інформація.

Недоліки:

- Відсутність графічних елементів.

Робота програми наведена нижче.

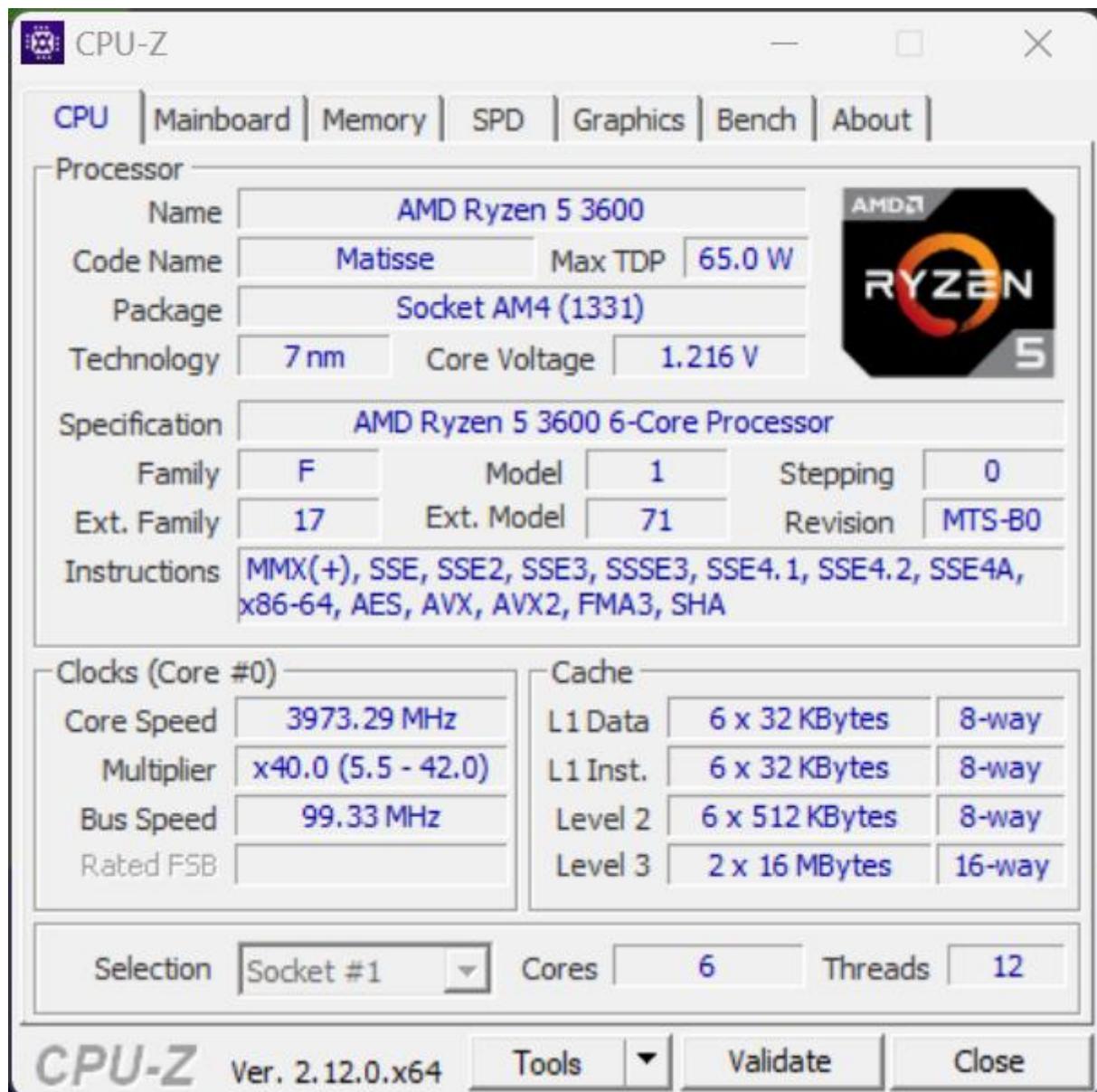


Рисунок 1.1 – Робота програми «Срп-Z»

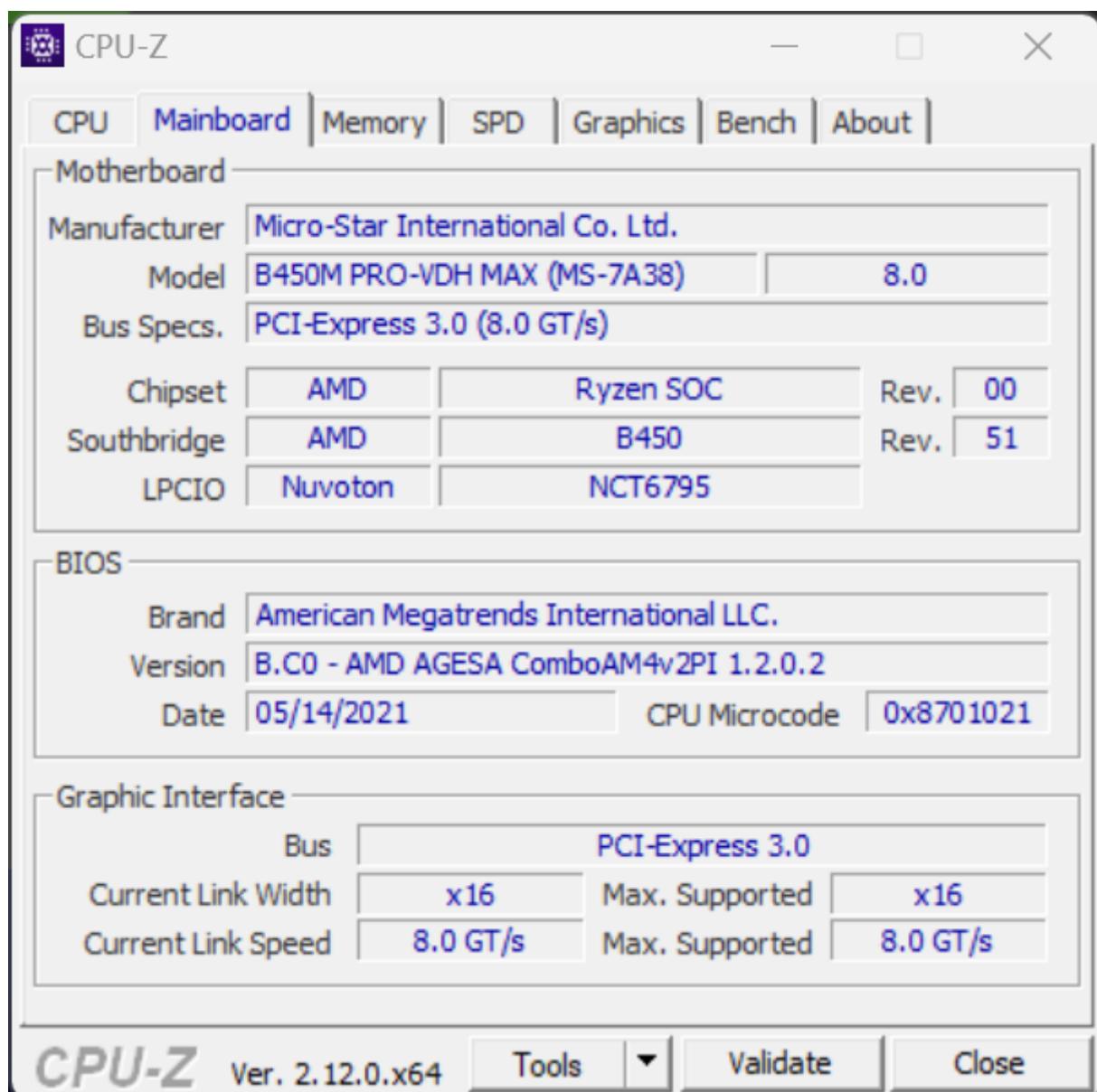


Рисунок 1.2 – Работа программы «Сри-Z»

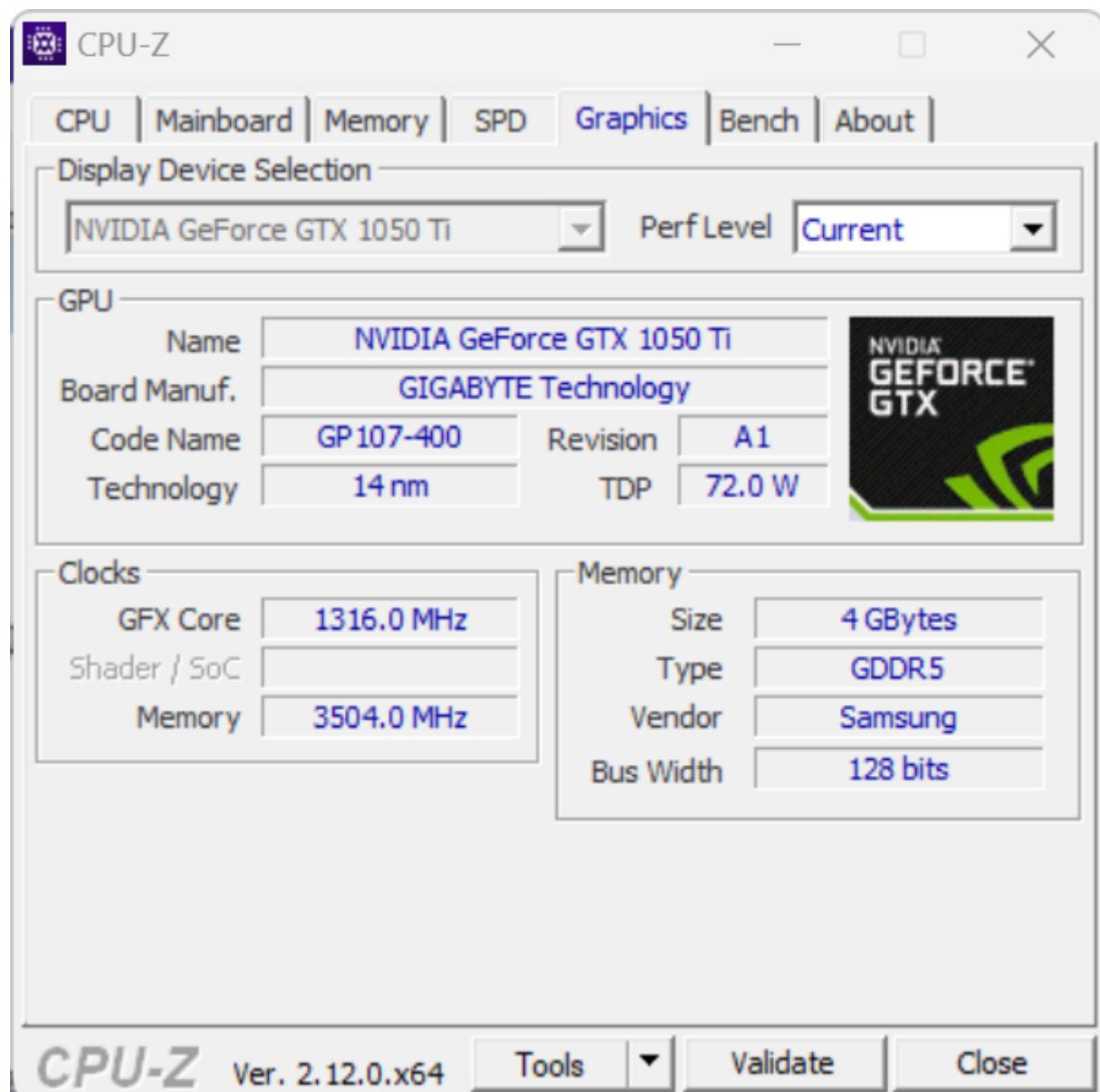


Рисунок 1.3 – Робота програми «Срі-З»

### 1.2.3 Система «Open Hardware Monitor»

Open Hardware Monitor є безкоштовним програмним забезпеченням з відкритим вихідним кодом, яке відстежує температурні датчики, швидкість обертання вентиляторів, напругу, навантаження і тактову частоту комп'ютера.

Переваги:

- Детальна інформація;
- Дані чітко структуровано;

- Наявність графічних елементів.

Робота програми наведена нижче.

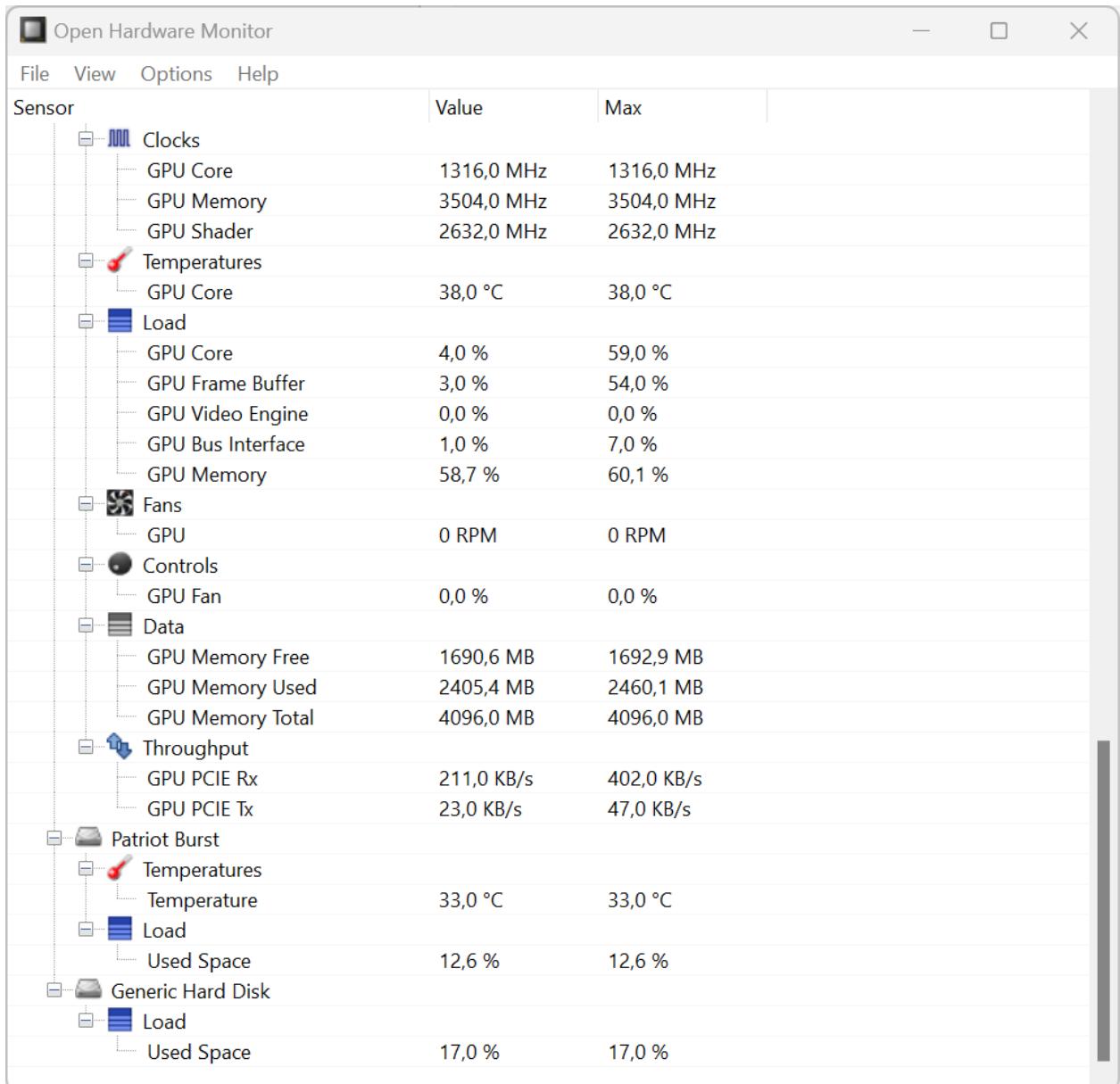


Рисунок 1.4 – Робота програми «Open Hardware Monitor»

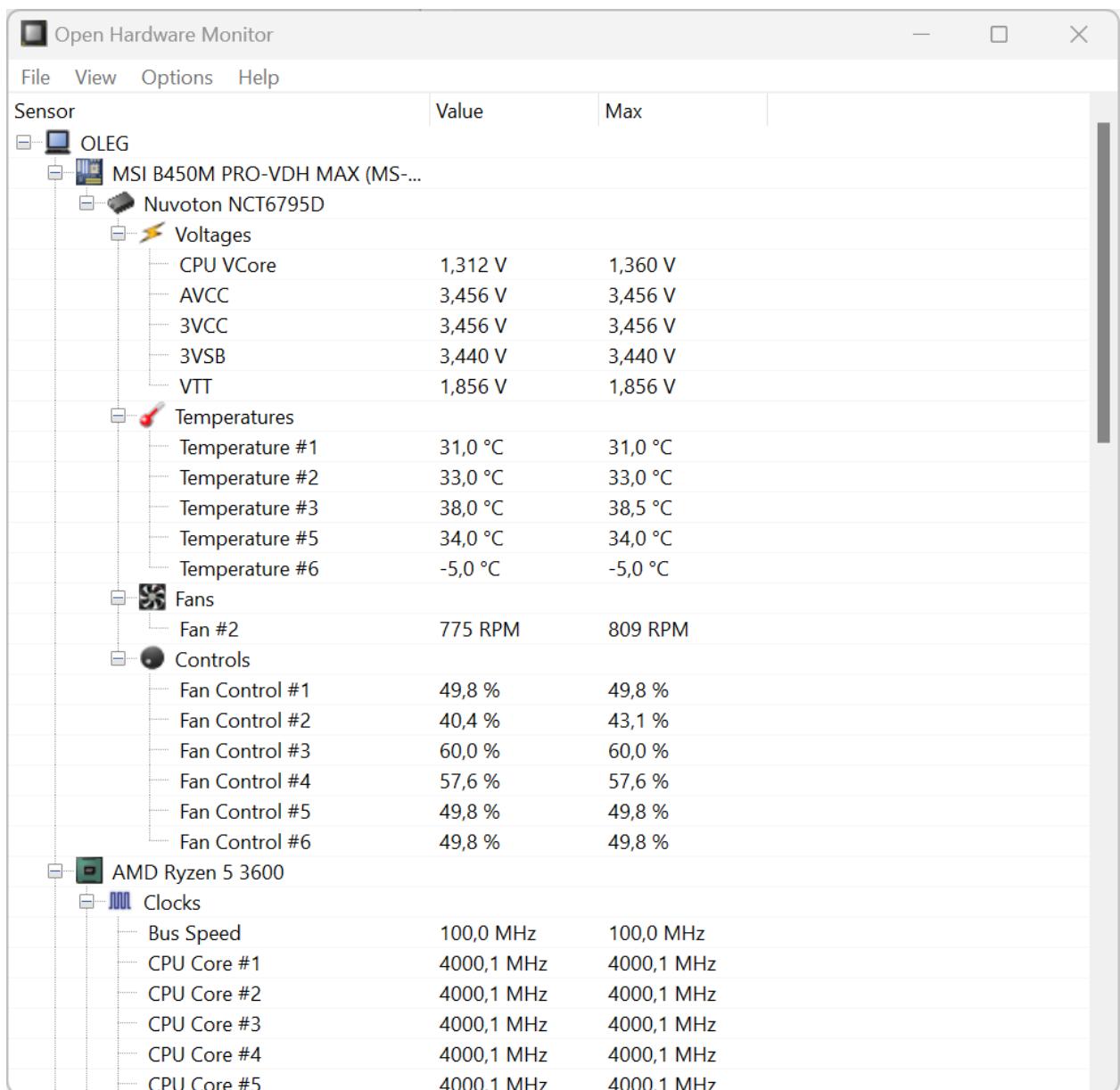


Рисунок 1.5 – Работа программы «Open Hardware Monitor»

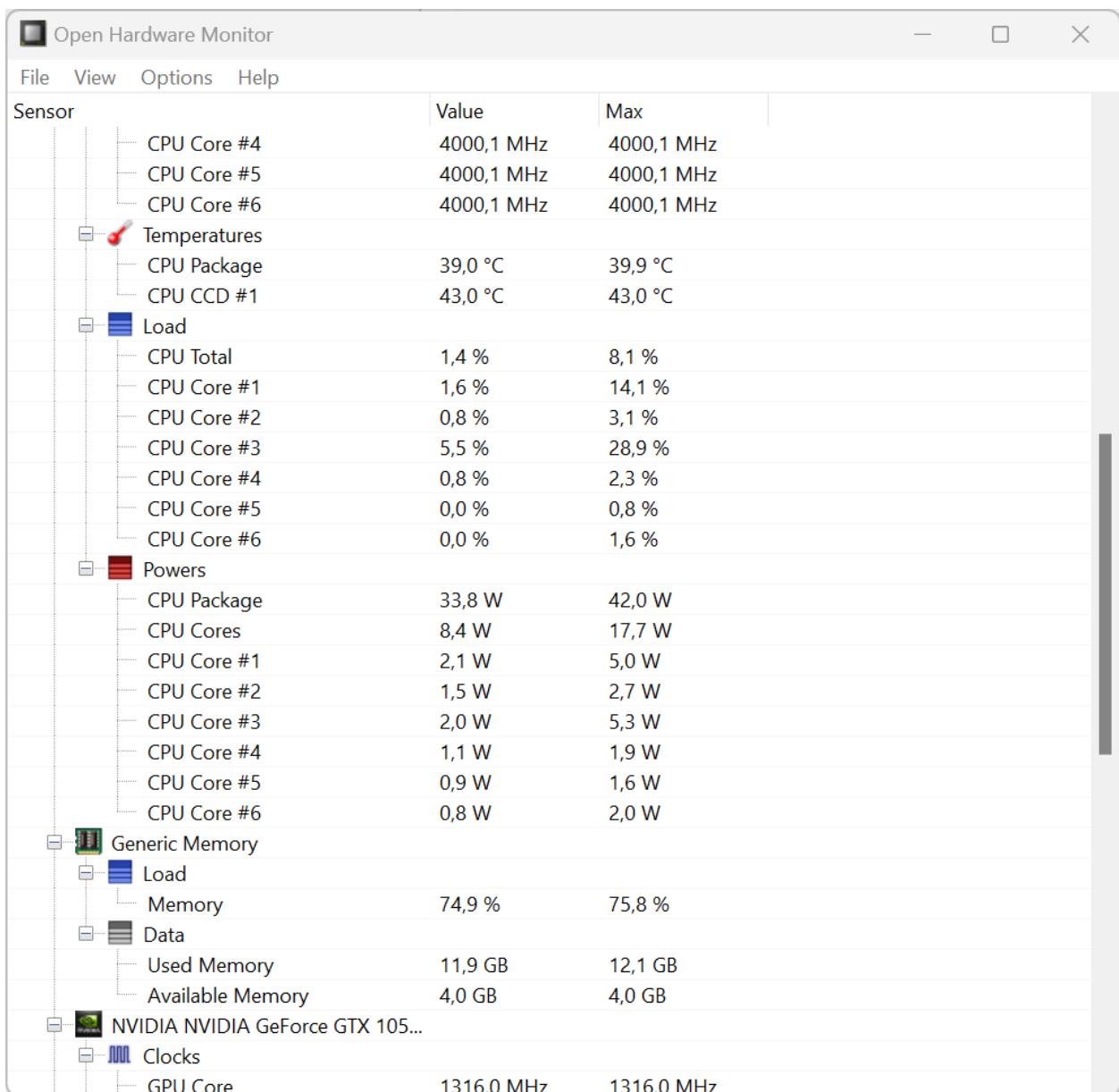


Рисунок 1.6 – Робота програми «Open Hardware Monitor»

#### **1.2.4 Висновки огляду існуючих застосунків**

Після огляду наявних систем зрозуміло що структурування інформації та її оновлюваність є ключовими показниками оцінки подібної системи. Наявність графічних елементів відображення змін показників системи також є корисною функцією.

## 2 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Технічне завдання передбачає використання мови програмування Python, яку і було застосовано до розробки програмної системи.

Серед функціональних вимог у завданні прописані наступні:

*Як вихідну інформація програма має виводити у графічний інтерфейс користувача всі зазначені характеристики для кожного пристрою у поточний момент часу:*

*1. Процесор*

- a. Назва*
- b. Кількість ядер*
- c. Кількість потоків*
- d. Тактова частота*
- e. Температура*
- f. Відсоток завантаженості*

*2. Графічний процесор*

- a. Назва*
- b. Тактова частота*
- c. Температура*
- d. Швидкість обертів вентиляторів*
- e. Обсяг оперативної пам'яті*

*3. Оперативна пам'ять*

- a. Загальний обсяг*
- b. Доступний обсяг*
- c. Швидкість*

*4. Дисковий накопичувач*

- a. Температура*
- b. Загальний обсяг*

- c. Доступний обсяг
- 5. Пристрій охолодження
  - a. Швидкість обертів

При виконанні програми вимогу було скореговано під наявні технічні засоби та методи доступні у бібліотеках розробки програмного продукту. В результаті вимоги наступні:

1. Процесор;
  - a. Назва;
  - b. Платформа;
  - c. Кількість ядер;
  - d. Кількість потоків;
  - e. Тактова частота;
  - f. Відсоток завантаженості.
2. Оперативна пам'ять;
  - a. Загальний обсяг;
  - b. Доступний обсяг;
  - c. Використаний обсяг.
3. Пам'ять підкачки;
  - a. Загальний обсяг;
  - b. Доступний обсяг;
  - c. Використаний обсяг.
4. Дискові накопичувачі;
  - a. Для кожного диску:
    - i. Назва;
    - ii. Літера диску;
    - iii. Тип файлової системи;
    - iv. Загальний обсяг;
    - v. Використаний обсяг;

vi. Вільний обсяг.

- b. Загальна кількість завантажених даних;
- c. Загальна кількість відвантажених даних.

5. Мережа.

- a. Для кожного пристрою мережі:
  - i. Адреса;
  - ii. Маска;
  - iii. Адреса трансляції (за наявності).
- b. Загальна кількість завантажених даних;
- c. Загальна кількість відвантажених даних.

Технічне завдання також передбачає використання графічного інтерфейсу користувача та його оновлюваність. Ці пункти передбачено у розробленій програмній системі.

Загальну схему алгоритму роботи програми збережено згідно технічного завдання.

## 3 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

Система має бути виконана у веб-браузері. Сторінка має бути умовно розділена на такі теоретичні секції:

- Системні дані;
- Інформація про процесор;
- Інформація про пам'ять;
- Інформація про накопичувачі;
- Інформація про мережу.

Більш детальна інформація про кожний підрозділ програми наведена нижче у відповідних розділах.

### 3.1 Системні дані

У розділі Системні дані користувач має бачити основну інформація про систему, а саме:

- Тип системи;
- Ім'я авторизованого користувача;
- Номер випуску системи;
- Повна версія системи;
- Тип процесора;
- Дата та час запуску системи.

### 3.2 Інформація про процесор

Розділ інформації про процесор має розділятися на дві умовні секції:

- Теоретична інформація;
- Показники характеристик на поточний момент часу.

### ***3.2.1 Теоретична інформація***

Усі теоретичні дані системи користувача мають бути виведенні з використанням таблиць даних, в яких перша колонка міститиме назву характеристики, а друга міститиме значення характеристики.

Теоретична інформація про процесор користувача має включати такі дані:

- Повна назва процесора;
- Назва платформи процесора;
- Кількість ядер процесора;
- Кількість потоків процесора.

### ***3.2.2 Показники компонентів***

Характеристики процесор мають бути розділені на дві оновлювальні підсекції:

- Інформація про частоту процесора;
- Інформація про використовуваність процесора.

Інформація про частоту процесора має бути представлена у вигляді таблиці та включати такі характеристики:

- Мінімальна частота;
- Максимальна частота;
- Поточна частота.

Такий набір характеристик зумовлен тим, що частота роботи процесора може змінюватися в ході роботи системи. Для відображення таких змін програма має передбачати виведення як максимальної, так і мінімальної частот для полегшення процесу відлагодження системи в разі наявності несправностей.

Підсекція інформації про частоту процесора також має містити графічний елемент – кругову діаграму, що показуватиме поточну частоту процесора порівняно з мінімальною та максимальною. Така кругова діаграма має мати початкове значення як мінімальна частота процесора, поточне значення як поточна частота процесора, і максимальне значення як максимальна тактова частота процесора у Мегагерцах.

Інформація про використовуваність (Англійською usage або застосування) усіх потоків процесора має бути наведена у вигляді таблиці та включати таку інформацію:

- Для кожного потоку процесора процент використовуваності.

Таблиця інформації про застосуваність процесора має бути динамічно оновлюваною і таблиця має включати дані про всі доступні потоки процесора користувача. Дані таблиці також мають бути оновлюваними щосекунди, що є інтервалом оновлення інтерфейсу користувача.

### **3.3 Інформація про пам'ять**

Підрозділ інформації про пам'ять має логічно розділятися на дві секції, а саме:

- Інформація про оперативну (віртуальну) пам'ять;
- Інформація про пам'ять підкачки.

### **3.3.1 Оперативна пам'ять**

Дані про оперативну пам'ять мають бути наведені у вигляді таблиці та містити таку інформацію:

- Загальний обсяг оперативної пам'яті;
- Обсяг доступної пам'яті;
- Кількість наразі використаної оперативної пам'яті;
- Відсоток використаної оперативної пам'яті.

Підрозділ про оперативну пам'ять також має містити кругову діаграму, що показуватиме поточний відсоток використання оперативної пам'яті на системі користувача.

### **3.3.2 Пам'ять підкачки**

Дані про пам'ять підкачки мають містити таку інформацію:

- Загальний обсяг пам'яті підкачки;
- Вільний обсяг пам'яті підкачки;
- Використано місця пам'яті підкачки;
- Відсоток використаного місця з загального обсягу пам'яті підкачки.

Для секції пам'яті підкачки також має бути кругова діаграма, подібна на ту, що у підрозділі віртуальної (оперативної) пам'яті.

## **3.4 Накопичувачі**

Розділ інформації про накопичувачі має складатися з двох умовних секцій:

- Інформація про кожен накопичувач;
- Загальні дані.

### ***3.4.1 Детальна інформація***

У розділі інформації про накопичувачі для кожного доступного фізичного накопичувача системи має бути надана детальна інформація двох типів:

- Дані про накопичувач;
- Інформація про місце на накопичувачі.

Дані про накопичувач мають включати такі відомості:

- Назва накопичувача;
- Тип файлової системи диску;
- Умовна позначка (літера) накопичувача.

Інформація про обсяг і місце накопичувача має містити такі дані:

- Загальний обсяг диску;
- Використаний простір накопичувача;
- Вільний обсяг диску;
- Відсоток використання обсягу накопичувача.

Кожна секція накопичувача має бути представлена випадним меню, яке, при натисканні, має показувати інформація про накопичувач.

Секція накопичувача також має містити кругову діаграма вільного місця на диску, де поточне значення є відсоток заповненості дискового накопичувача.

### ***3.4.1 Загальна інформація***

Загальні дані накопичувачів мають містити такі значення:

- Загальна кількість прочитаних даних;
- Загальна кількість записаних даних.

## **3.5 Мережа**

Інформація про мережу також має бути організована у дві умовні підсекції:

- Інформація про кожен пристрій мережі;
- Загальні дані про мережу.

### ***3.5.1 Детальна інформація***

Дані про накопичувач, аналогічно секції з дисками, мають бути закриті у випадні меню. Такий підхід обрано оскільки пристрой може бути багато (так само як і дисків), а кожен має містити певний обсяг інформації, який, якщо відображати на сторінці, може ускладнити загальне сприйняття інформації про показники системи.

Інформація про мережевий пристрій має містити такі дані:

- Назва пристрою;
- Адреса пристрою;

- Маска пристрою;
- Адреса трансляції пристрою (якщо є).

### ***3.5.1 Загальна інформація***

Загальні дані про мережу мають містити такі дані:

- Загальна кількість прочитаних даних;
- Загальна кількість записаних даних.

## **3.6 Інші відомості**

### ***3.6.1 Мова інтерфейсу***

Мова інтерфейсу застосунку обрана Англійська для покращення інтернаціоналізації та сприйняття даних застосунку серед людей.

Назви розділів Англійською наведені нижче:

- System
- Processor
  - Frequencies
  - Usage
- Virtual Memory
- Swap Memory
- Disks
  - Data
  - Space
- Network
  - Data

Усі інші підказки та підписи також зроблено Англійською мовою з відповідним текстом позначки кожного елементу.

### ***3.6.2 Структура інтерфейсу***

Інтерфейс застосунку структуровано таким чином:

- У верхній секції показано інформаційні блоки, які організовано по стовпчиках. Кожен стовпчик може мати підстовпчики якщо секція містить додаткові дані (наприклад, секція процесора має два підстовпчика: частоти та використованість).
- Нижня частина екрану містить графічні показники: три графіки в яких динамічно відображені поточні показники різних компонентів системи.

Графічні частини програми містять такі графіки:

- Завантажуваність процесора;
- Завантажуваність оперативної пам'яті;
- Використання мережі.

Графік мережі містить дві лінії: швидкість відвантаження даних та швидкість завантаження даних.

Кожен графік оновлюється раз на секунду з новими отриманими даними про показники системи.

## 4 ПРОГРАМНА РЕАЛІЗАЦІЯ

### **4.1 Вибір засобів**

При виконанні проекту було обрано наступні технічні засоби:

- Мова програмування Python;
- Редактор коду Visual Studio Code;
- Бібліотека NiceGUI для інтерфейсу;
- Бібліотеки psutil, platform, та win32api для інформації про систему;
- Бібліотеки datetime, collections для додаткових функцій.

### **4.2 Компоновка проекту**

Робота проекту починається з імпортuvання використаних модулів для роботи програми.

З модулю collections програма імпортує контейнери defaultdict та namedtuple.

Контейнер defaultdict буде потрібен для заповнення таблиць даних, де ключі можуть динамічно змінюватися.

Контейнер namedtuple потрібен для формування моделей даних переважно для легшої взаємодії зі статичними даними програми.

З модулю datetime програма імпортує datetime.

Модуль datetime буде необхідний для розрахування поточного часу для оновлення графіків змін показників компонентів у графчному інтерфейсі користувача.

З модулю nicegui програма імпортує ui.

Елемент `ui` є основним способом роботи з бібліотекою NiceGUI і надає доступ до усіх (наче) доступних елементів інтерфейсу та методів взаємодії з ними.

Також програма імпортує модулі `win32api`, `platform`, та `psutil`. Обидри ці модулі потрібні будуть для витягання системних даних для заповнення інтерфейсу користувача.

Після імпортування модулів у програмі оголошено константні змінні: класи для центрування об'єктів `CENTER_CLASSES` та стандартні стовпці для таблиць інтерфейсу `COLUMNS`.

Класи для центрування об'єктів містять пару класів TailwindCSS, які використовуються бібліотекою NiceGUI для зміни стилів елементів. Ці класи будуть використані для центрування кругових діаграм та їх підписів на сторінці. Один з класів визначено не повністю, бо значення класу змінюється в залежності від елементу інтерфейсу. Якщо клас застосовується до підпису діаграми, то потрібно вказати значення `text-center`, якщо ж до самої діаграми, то значення має бути `self-center`. Тому на початку змінної залишено лише частинку `-center`, а у коді при застосуванні цих класів вказано необхідний модифікатор. Інший клас який застосовується це `w-full`, який надає елементу повну доступну ширину.

Змінна стандартних стовпців таблиці є списком з словників. Кожен словник містить стандартні необхідні дані для оголошення стовпця елементу інтерфейсу таблиця з бібліотеки NiceGUI.

Далі оголошується функція `get_formatted_size`. На вході вона приймає кількість байт `bytes`, зазвичай отриману від бібліотеки витягання системних даних, а також суфікс `suffix`, за замовчуванням стоїть “`B`” як `Bytes`.

Далі система оголошує відповідні змінні даних: `system`, `boot_time`, `cpu_frequencies`, `virtual_memory`, `swap_memory`, `partitions`, `disks`, `networks`, `network`, `network_sent`, `network_received`. Кожна з цих змінних використовує бібліотеки для витягання системних даних.

Після цього програма оголошує моделі даних `System`, `Processor`, `ProcessorFrequencies`, `VirtualMemory`, `SwapMemory`, `Disks`, `Network` та заповнює їх даними `system_data`, `processor_data`, `processor_frequencies_data`, `virtual_memory_data`, `swap_memory_data`, `disks_data`, `network_data`.

Далі програма оголошує функцію `get_rows`. На вході вона приймає словник з даними `data`. На виході функція формує рядки для таблиці для кожного елементу словника.

Після цього програма оголошує функцію `update_ui`. На вході ця функція не отримує нічого, але перезаписує глобальні змінні даних новими значеннями та оновляє інтерфейс з новими значеннями.

Далі у програмі прописані елементи інтерфейсу відповідно до завдання. Усі елементи згруповані або у рядки, або у стовпці. Для кожного елементу інтерфейсу використовуються дані: або статичні, або динамічні. Якщо дані динамічні, вони оновлюються у функції `update_ui`. Також дані можуть бути оголошенні всередині блоку створення елементів інтерфейсів у випадках коли елементи даних можуть динамічно оновлюватися в залежності від користувальської системи.

Після цього програма також оголошує рядок з трьома графіками зміни характеристик компонентів: графік застосованості процесора `processor_usage_plot`, графік застосованості оперативної пам'яті `memory_usage_plot`, та графік швидкостей мережі `network_speed_plot`. Для

кожного з цих графіків додаються початкові та умовно максимальні значення на початку аби коректно відобразити шкалу можливих значень.

Насамкінець програма запускає застосунок на сервері та виконує таймер, який оновлюється щосекунди та викликає функцію `update_ui`.

## **5 КЕРІВНИЦТВО ОПЕРАТОРА**

Для використання програми необхідно зробити кілька кроків:

- Встановити мову програмування Python на систему;
- Завантажити код програми;
- Встановити використані пакети програми;
- Запустити програму на сервері.

Опис виконання та результатів кожного з кроків надано нижче у відповідному пункті.

### **5.1 Встановлення Python**

Для встановлення мови програмування Python необхідно перейти на офіційний сайт та завантажити програму-встановлючав останньої версії.

Після виконання програми-встановлювача на системі користувача має бути встановлений виконувач програм типу .ру.

Цей крок необхідно виконувати лише якщо система користувача ще не має встановленої версії мови програмування Python.

### **5.2 Завантаження коду**

Після встановлення виконувача програм, написаних мовою програмування Python, необхідно завантажити сам вихідний код програми. Завантажити його можна [за посиланням](#).

Після завантаження коду необхідно перенести його на зручне місце на локальному диску для подальшої роботи.

### 5.3 Встановлення пакетів

Для встановлення використаних пакетів для програми необхідно відкрити теку із завантаженим проєктом та у консолі запустити наступну команду:

```
pip install -r requirements.txt
```

Після запуску така команда має встановити усі необхідні пакети на систему користувача.

### 5.4 Запуск програми

Для запуску програми потрібно виконати файл run.py у завантаженій текі проєкту. Виконання цього файлу запустить сервер проєкту і відкриє нову вкладку в стандартному браузері користувача з інтерфейсом проєкту.

## 6 КЕРІВНИЦТВО ПРОГРАМІСТА

### 6.1 Інструкція з модернізації

Для переробки або модернізації проєкту необхідно дотриматися певних початкових умов:

1. Мати Python встановленим на комп’ютері;
2. Завантажити код програми;
3. Встановити пакети залежності для програми.

Для завантаження та встановлення мови Python необхідно завантажити програму-встановлювач з офіційного сайту мову та виконати її, дотримуючись усіх наданих інструкцій.

Для завантаження коду програми необхідно клонувати репозиторій [за посиланням](#) або завантажити ZIP архів з файлами програми [за посиланням](#).

Для встановлення пакетів, використовуваних у програмному застосунку необхідно ввести у консолі наступну команду (переконайтесь що локація консолі зазначена у теці клонованого проєкту на системі)

```
pip insall -r requirements.txt
```

Після виконання усіх попередніх кроків можна приступити до модернізації та переробки коду. Структура програми наведена детальніше у розділі 4 цього документу.

## 6.2 Таблиця функцій

Програма містить наступні функції:

| Назва              | Параметри  | Дії  |
|--------------------|--|--|
| get_rows           | data: словник з даними про розділ програми                       | Формує список рядків для елементу інтерфейсу таблиця                           |
| update_ui          | -  | Оновлює дані програми та відповідні графічні елементи, до яких вони прив'язані |
| get_formatted_size | bytes: кількість байтів,<br>suffix: суфікс, за замовчуванням 'B' | Формує розмір у відповідній величині, приймає кількість байтів на вході        |

Таблиця 6.1 – Функції програми

## ВИСНОВКИ

Під час виконання курсового проекту благодаттю Господа нашого Ісуса Христа було розроблено проект аналізу системних ресурсів мовою програмування Python із використанням зовнішніх пакетів NiceGUI, psutil, та win32api. Програма має візуальний графічний інтерфейс, оновлює дані щосекунди, має графічні елементи відображення змін параметрів системи та детальний опис інформації системи користувача.

## ПЕРЕЛІК ПОСИЛАНЬ

- 1) How to Get Hardware and System Information in Python [Електронний ресурс]. – Режим доступу: <https://thepythoncode.com/article/get-hardware-system-information-python>
- 2) psutil documentation [Електронний ресурс]. – Режим доступу: <https://psutil.readthedocs.io/en/latest/?badge=latest#>
- 3) NiceGUI Documentation [Електронний ресурс]. – Режим доступу: <https://nicegui.io/documentation>
- 4) How to Make a Network Usage Monitor in Python [Електронний ресурс]. – Режим доступу: <https://thepythoncode.com/article/make-a-network-usage-monitor-in-python>
- 5) Is it possible to get in Python the CPU/core/processor id that the python program itself is using? - Quora [Електронний ресурс]. – Режим доступу: <https://www.quora.com/Is-it-possible-to-get-in-Python-the-CPU-core-processor-id-that-the-python-program-itself-is-using>
- 6) Python get cpu info [Електронний ресурс]. – Режим доступу: <https://www.programcreek.com/python/?CodeExample=get%20cpu%20info>
- 7) How to use the psutil.net\_if\_addrs function in psutil | Snyk [Електронний ресурс]. – Режим доступу: [https://snyk.io/advisor/python/psutil/functions/psutil.net\\_if\\_addrs](https://snyk.io/advisor/python/psutil/functions/psutil.net_if_addrs)

## **ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ**

### **РОЗРОБКА ПРОГРАМИ, ЯКА РЕАЛІЗУЄ ОСНОВНІ ЗАДАЧІ АНАЛІЗУ СИСТЕМНИХ РЕСУРСІВ ПЕОМ**

#### **Мета розробки та її призначення**

Метою розробки програми є отримання необхідної системою інформації за запитом користувача. Особливістю розроблюваної програми є створення десктопного застосунку для швидкого отримання точної інформації про поточний статус системи та про роботу її окремих пристройів.

#### **Змісти теоретичних посилок до розробки**

Системні ресурси описують доступні можливості Персональної електронно-обчислювальної машини (ПЕОМ). Для їх аналізу операційні системи дають доступ до функцій, через які можна отримати інформацію про поточний статус пристройів.

Пристрої системи зазвичай включають дискові накопичувачі, процесори, оперативну пам'ять, графічні процесори, пристрої охолодження. Кожен з цих пристройів має свої характеристики, інформація про які може бути потрібна користувачеві. Для таких ситуацій розроблюється програма цього проекту. Вона дозволятиме користувачеві побачити інформацію про пристрої системи та їх поточні характеристики. Наприклад, програма має надавати доступ до назви процесора машини, назви графічного процесора, обсягу оперативної пам'яті, поточної швидкості роботи процесора, і так далі.

Проект розроблюється під операційну систему Windows, яка надає доступ до необхідної інформації через системний реєстр, засоби Windows Management Instrumentation (WMI) або через засоби Windows API.

Кожен пристрій системи може мати різні характеристики. Наприклад, оперативна пам'ять має тактову частоту, вимірюну у мегагерцах; процесор має

кількість ядер, кількість потоків, поточну швидкість у мегагерцах; дисковий накопичувач має температуру у цельсіях або фаренгейтах, загальний простір вимірюється в гігабайтах та доступний простір вимірюється в гігабайтах; пристрій охолодження має поточну швидкість обертів вимірювання у обертах за хвилину (revolutions per minute або rpm). Всі ці характеристики можуть змінюватися з часом роботи програми, тому маємо це враховувати.

## **Основні вимоги до програми**

Програма має коректно працювати з операційною системою Windows, а також з усіма типами операційних систем, які мають встановлену програму браузера. Система також має передбачати графічний інтерфейс користувача та використовувати системні методи для отримання необхідної інформацію про пристрой.

Як початкові дані програма має використовувати отриманні за запитом до системи дані про ресурси системи, а саме інформація про процесор, графічний процесор, оперативну пам'ять, дискові накопичувачі та пристрой охолодження. Для кожного пристрою програма має виводити його поточні характеристики.

Як вихідну інформація програма має виводити у графічний інтерфейс користувача всі зазначені характеристики для кожного пристрою у поточний момент часу:

1. Процесор
  - a. Назва
  - b. Кількість ядер
  - c. Кількість потоків
  - d. Тактова частота
  - e. Температура
  - f. Відсоток завантаженості

2. Графічний процесор
  - a. Назва
  - b. Тактова частота
  - c. Температура
  - d. Швидкість обертів вентиляторів
  - e. Обсяг оперативної пам'яті
3. Оперативна пам'ять
  - a. Загальний обсяг
  - b. Доступний обсяг
  - c. Швидкість
4. Дисковий накопичувач
  - a. Температура
  - b. Загальний обсяг
  - c. Доступний обсяг
5. Пристрій охолодження
  - a. Швидкість обертів

Ці дані програма має виводити у графічному вигляді засобами графічного інтерфейсу та оновлювати їх періодично для отримання точного статусу системи на кожен момент часу.

Графічний інтерфейс має складатися з елементів подання інформації у вигляді дерева або таблиці. Для кожного компоненту має бути виділено окреме місце інтерфейсу та розписані його поточні характеристики. Ці характеристики можна зазначити або як окремі таблиці у інтерфейсі, або у вигляді дерева де корінь це назва системи користувача.

### **Алгоритм функціонування програми**

При запуску програма має зчитувати всю необхідну інформацію з системи. Це можна зробити вбудованими засобами системи Windows. Варіанти витягання необхідної інформації про систему наступні:

- Використати засоби WMI. Цей варіант є найбільш простим бо надає необхідну інформацію за доступом до відповідних змінних.
- Використання засобів Windows API. Цей варіант найскладніший бо його підтримка обраними засобами розробки обмежена.
- Доступ до системного реєстру. Цей варіант є середнім за складністю і також надає інформацію за доступом до змінних.

При реалізації програми може виникнути потреба використанні комбінації усіх цих засобів аби отримати повну інформацію. Також при розробці може виникнути ситуація коли треба буде додавати додаткові способи витяганні інформації якщо вже зазначені не надаватимуть якихось даних.

Після отримання всієї необхідної інформації про кожну з компонент системи програма має вивести її користувачеві у доступному та зрозумілому вигляді. Це має здійснюватись шляхом використання графічного інтерфейсу. Користувач має чітко бачити окремі компоненти та їх характеристики у кожен момент часу.

При використанні програма має додатково запитувати необхідну інформацію періодично та оновлювати інтерфейс з новими даними. Така функція необхідна для отримання актуальної поточної інформації про систему. Це потрібно через те що використання системи може змінювати її характеристики. Наприклад навантаження на процесор може змінюватись при використанні різних програм, які потребують більше ресурсів; або обсяг доступної оперативної пам'яті може змінюватись при використанні програм які потребують більше доступної оперативної пам'яті для роботи.

## **Обґрунтування вибору апаратно-технічних засобів, операційної системи та мови програмування**

Операційна система вибору для поточного проекту є Windows 10 або 11. Обрана саме ця операційна система через використання її на машині розробки. Обрання іншої операційної системи потребуватиме використання засобів емуляторів або встановлення та використання іншої системи для розробки програмного забезпечення. Жоден з цих варіантів не є бажаним вибором.

Програма має розроблюватися інструментами розробки Python або C#. Ці засоби розробки надають доступ до усіх необхідних функцій системи, мають широку розповсюдженість серед розробників, надають багато доступних ресурсів у системі інтернет для використання та витягування потрібної інформації в процесі розробки. Також ці мови програмування містять необхідні можливості створення графічних інтерфейсів користувача. Для мови програмування C# засобами розробки графічного інтерфейсу можуть бути Windows Forms або WPF. В разі використанні мови Python також доступні різні бібліотеки для написання графічного інтерфесу, як от PyQt або tkinter або NiceGUI.

Скоріш за все для проєкту буде використана мова програмування Python, а разом з нею засоби розробки інтерфейсу NiceGUI. Використання Python для розробки обґрунтоване її знайомістю для розробника та доступністю джерел інформації і бібліотек для витягання необхідної інформації про систему. Мова програмування Python містить доступний та простий синтаксис що має спростити процес розробки та відповідно скоротити його загальний час.

Використання мови Python дозволяє застосувати методи розробки Microsoft Visual Studio Code які мають всі необхідні засоби та методи для швидкого та просто процесу розробки проєкту.

## **Точно визначені системні функції програми**

Конкретні функції програми можуть змінюватись в процесі розробки програм. Найбільш необхідні функції програми такі:

- Витяганні інформації про систему. Ця функція має запитувати засоби системи про поточні характеристики кожного з компонентів. Така функція може бути реалізована багатьма окремими функціями або класами та модулями програми. Конкретний алгоритм роботи функції залежить від обрання даних які необхідно витягнути. При завершенні роботи ця функція має повернати всі необхідні дані у вигляді об'єкту JSON або словнику значень у вигляді ключ: значення.
- Виведення необхідної інформації. Ця функція має приймати як аргумент усі витягнені дані з попередньої функції та виводити їх на екран користувача. Виведення даних має здійснюватись через оголошення та налаштування відповідних засобів інтерфейсу користувача. Ці засоби мають бути використані з наявних методів обраної бібліотеки проектування графічного інтерфейсу. Розробка цією функції також може передбачати використання багатьох класів та модулів коду для кращої модульності та незалежності кожної з компонентів системи.
- Оновлення даних. Ця функція має комбінувати роботу двох попередніх функцій. Такий підхід необхідний для забезпечення надання точної інформації про систему під час запиту а також під час використання програми. Ця функція може використати попередньо розроблені методи та функції першої функції а також методи другої функції для оновлення інтерфейсу користувача.

Комбінація цих трьох функцій має забезпечити коректну та точну роботу програми завдяки якій користувач може отримати необхідну поточну інформація про статус системи з деталями про кожен компонент, його роботу та його характеристики у необхідний момент часу.

## **Обмеження на установку та використання**

Встановлення програмного застосунку не має займати багато часу оскільки застосунок має бути виконуваним у веб-браузері на системі користувача. Застосунок має коректно працювати на будь-якій операційній системі, яка має встановлений веб-браузер. Для запуску застосунку і його роботи на сторінці необхідно провести встановлення застосунку на сервер. На сервері потрібно виконати встановлення використаних програмних пакетів які застосунок використовує. Назви пакетів та їх версії мають бути прописані у окремому текстовому файлі з відповідною назвою. Після встановлення пакетів та запуску застосунку на сервері користувачі мають мати змогу використовувати застосунок через введення його адреси у пошуковий рядок системного браузера.

При бажанні використати застосунок на мобільному пристрой обов'язковим є його встановлення на віддаленому сервері з отриманням статичної URL адреси. Без використання серверу для встановлення застосунку його використання на мобільному пристрой не може бути зроблене через обмеження функціоналу мобільних операційних систем, зокрема через те, що вони не передбачають встановлення мови програмування Python.

Використання застосунку має проходити із використанням простого світлого графічного інтерфейсу користувача на сторінці веб-браузера. Взаємодія з компонентами має бути виконана через графічні візуальні елементи сторінки користувача. Елементи також мають бути оновлені з певним інтервалом для забезпечення актуальності інформації на веб-сторінці користувача.

## ВИСНОВКИ

По закінченню роботи було отримано документ що містить усю необхідну інформацію про проект та його можливі майбутні характеристики. Наявність такого документу технічного завдання може дозволити перейти до розробки програмного забезпечення із зазначеними вимогами.

## ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

### **Б1 – run.py**

```

from collections import defaultdict, namedtuple
from datetime import datetime
from nicegui import ui
import win32api
import platform
import psutil

CENTER_CLASSES: str = '-center w-full'
COLUMNS: list[dict] = [
    {'id':'property','label':'Property','field':'property','align':'left'},
    {'id':'value','label':'Value','field':'value','sortable':True},
]

def get_formatted_size(bytes,suffix='B'):
    factor=1024
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes<factor: return f'{bytes:.2f} {unit}{suffix}'
        bytes/=factor

system=platform.uname()
boot_time=datetime.fromtimestamp(psutil.boot_time())
cpu_frequencies=psutil.cpu_freq()
virtual_memory=psutil.virtual_memory()
swap_memory=psutil.swap_memory()
partitions=psutil.disk_partitions()
disks=psutil.disk_io_counters()
networks=psutil.net_if_addrs()
network=psutil.net_io_counters()
network_sent, network_received=network.bytes_sent, network.bytes_recv

System=namedtuple('System','type user release version machine booted')
system_data=System(
    type=system.system,
    user=system.node,
    release=system.release,
    version=system.version,
    machine=system.machine,
    booted=f'{boot_time.day}.{boot_time.month}.{boot_time.year}'
    {(boot_time.hour:02d}:{boot_time.minute:02d}:{boot_time.second:02d}')
)

Processor=namedtuple('Processor','name platform cores threads')
processor_data=Processor(
    name=system.processor,
    platform=system.machine,
    cores=psutil.cpu_count(logical=False),
    threads=psutil.cpu_count(logical=True),
)

```

```

    )

ProcessorFrequencies=namedtuple('ProcessorFrequencies','min max current')
processor_frequencies_data=ProcessorFrequencies(
    min=cpu_frequencies.min,
    max=cpu_frequencies.max,
    current=cpu_frequencies.current,
)

VirtualMemory=namedtuple('VirtualMemory','total available used percentage')
virtual_memory_data=VirtualMemory(
    total=get_formatted_size(virtual_memory.total),
    available=get_formatted_size(virtual_memory.available),
    used=get_formatted_size(virtual_memory.used),
    percentage=f'{virtual_memory.percent}%',
)
SwapMemory=namedtuple('SwapMemory','total free used percentage')
swap_memory_data=SwapMemory(
    total=get_formatted_size(swap_memory.total),
    free=get_formatted_size(swap_memory.free),
    used=get_formatted_size(swap_memory.used),
    percentage=f'{swap_memory.percent}%',
)
Disks=namedtuple('Disks','read write')
disks_data=Disks(
    read=get_formatted_size(disks.read_bytes),
    write=get_formatted_size(disks.write_bytes),
)
Network=namedtuple('Network','sent received')
network_data=Network(
    sent=get_formatted_size(network.bytes_sent),
    received=get_formatted_size(network.bytes_recv),
)
def get_rows(data:dict):
    return [{k.capitalize():v} for k,v in data.items()]

def update_ui():
    current_time=datetime.now().timestamp()

    cpu_frequencies=psutil.cpu_freq()
    processor_frequencies_data=ProcessorFrequencies(
        min=cpu_frequencies.min,
        max=cpu_frequencies.max,
        current=cpu_frequencies.current,
    )

    processor_frequencies_table.rows=get_rows(processor_frequencies_data._asdict())
    processor_frequencies_circle.value=cpu_frequencies.current

    threads_usage=psutil.cpu_percent(percpu=True)
    processor_usage_data={
        f'Core {index}': usage
        for index,usage in enumerate(threads_usage)
    }
    processor_usage_table.rows=get_rows(processor_usage_data)

```

```

cpu_usage=psutil.cpu_percent()
processor_usage_plot.push([current_time], [[cpu_usage]])

virtual_memory=psutil.virtual_memory()
virtual_memory_data=VirtualMemory(
    total=get_formatted_size(virtual_memory.total),
    available=get_formatted_size(virtual_memory.available),
    used=get_formatted_size(virtual_memory.used),
    percentage=f'{virtual_memory.percent}%',
)
virtual_memory_table.rows=get_rows(virtual_memory_data._asdict())
virtual_memory_circle.value=virtual_memory.percent
memory_usage_plot.push([current_time], [[virtual_memory.percent]]))

swap_memory=psutil.swap_memory()
swap_memory_data=SwapMemory(
    total=get_formatted_size(swap_memory.total),
    free=get_formatted_size(swap_memory.free),
    used=get_formatted_size(swap_memory.used),
    percentage=f'{swap_memory.percent}%',
)
swap_memory_table.rows=get_rows(swap_memory_data._asdict())
swap_memory_circle.value=swap_memory.percent

BITS_TO_KILOBITS=10**-3
global network_sent, network_received
current_network=psutil.net_io_counters()

current_sent, current_received=current_network.bytes_sent, current_network.bytes_recv
    download_speed, upload_speed=(current_received-
network_received)/1, (current_sent-network_sent)/1

download_speed, upload_speed=download_speed*BITS_TO_KILOBITS, upload_speed*BITS_TO_KILOBITS

network_speed_plot.push([current_time], [[download_speed], [upload_speed]])
    network_sent, network_received=current_sent, current_received

with ui.row().classes('flex gap-3'):
    with ui.column():

        ui.table(columns=COLUMNS, rows=get_rows(system_data._asdict()), title='System')

        with ui.column():

            ui.table(columns=COLUMNS, rows=get_rows(processor_data._asdict()), title='Processor')

            with ui.row().classes('flex w-full'):
                with ui.column():

processor_frequencies_table=ui.table(columns=COLUMNS, rows=get_rows(processor_frequencies_data._asdict()), title='Frequencies (MHz)')

                    ui.label('Processor
Frequency').classes('text'+CENTER_CLASSES)

processor_frequencies_circle=ui.circular_progress(min=cpu_frequencies.min,

```

```

max=cpu_frequencies.max,value=cpu_frequencies.current).classes('self'+CENTER_CLASSES)

    processor_usage_data={
        f'Core {index}': usage
        for index,usage in
        enumerate(psutil.cpu_percent(percpu=True))
    }

processor_usage_table=ui.table(columns=COLUMNS,rows=get_rows(processor_usage_data),title='Usage (%)').classes('flex-1')

    with ui.column():

virtual_memory_table=ui.table(columns=COLUMNS,rows=get_rows(virtual_memory_data._asdict()),title='Virtual Memory').classes('w-full')

        ui.label('Virtual Memory Usage').classes('text'+CENTER_CLASSES)

virtual_memory_circle=ui.circular_progress(value=virtual_memory.percent,max=100).classes('self'+CENTER_CLASSES)

swap_memory_table=ui.table(columns=COLUMNS,rows=get_rows(swap_memory_data._asdict()),title='Swap Memory')

        ui.label('Swap Memory Usage').classes('text'+CENTER_CLASSES)

swap_memory_circle=ui.circular_progress(value=swap_memory.percent,max=100).classes('self'+CENTER_CLASSES)

    with ui.column():
        with ui.card():
            ui.label('Disks').classes('q-table__title')
            disk_tables=defaultdict(dict)
            for partition in partitions:
                try: usage_data=psutil.disk_usage(partition.mountpoint)
                except: continue

partition_name=win32api.GetVolumeInformation(partition.device) [0]
            disk_data={
                'device':partition.device,
                'name':partition_name,
                'file system':partition.fstype,
            }
            space_data={
                'total':get_formatted_size(usage_data.total),
                'used':get_formatted_size(usage_data.used),
                'free':get_formatted_size(usage_data.free),
                'percentage':f'{usage_data.percent}%',
            }
            with ui.expansion(partition_name):

disk_tables[partition_name]['disk']=ui.table(columns=COLUMNS,rows=get_rows(disk_data),title=f'{partition_name} Data').classes('w-full')

disk_tables[partition_name]['space']=ui.table(columns=COLUMNS,rows=get_rows(space_data),title=f'{partition_name} Space').classes('w-full')

        ui.label(f'{partition_name}
Usage').classes('text'+CENTER_CLASSES)

```

```

ui.circular_progress(value=usage_data.percent,max=100,min=0).classes('self
'+CENTER_CLASSES)

disks_table=ui.table(columns=COLUMNS,rows=get_rows(disks_data._asdict()),r
ow_key='name').classes('w-full')

with ui.column():
    with ui.card():
        ui.label('Network').classes('q-table__title')
        network_tables=defaultdict(str)
        for interface_name,interface_addresses in networks.items():
            interface_addresses=[a for a in interface_addresses if
a.family.name=='AF_INET' or a.family.name=='AF_PACKET']
            for address in interface_addresses:
                interface_data={
                    'IP Address' if address.family.name=='AF_INET'
else 'MAC Address':address.address,
                    'netmask':address.netmask,
                    'Broadcast IP' if address.family.name=='AF_INET'
else 'Broadcast MAC':address.broadcast,
                }
                with ui.expansion(interface_name):
                    network_tables[interface_name]=ui.table(columns=COLUMNS,rows=get_rows(int
erface_data),title=f'{interface_name} Data')

network_table=ui.table(columns=COLUMNS,rows=get_rows(network_data._asdict(
)),row_key='name').classes('w-full')

with ui.row():

processor_usage_plot=ui.line_plot(n=1,figsize=(4.7,2.47)).with_legend(['CP
U Usage %'],loc='upper center',ncol=1)
    processor_usage_plot.push([datetime.now().timestamp()],[[0]])
    processor_usage_plot.push([datetime.now().timestamp()],[[100]])

memory_usage_plot=ui.line_plot(n=1,figsize=(4.7,2.47)).with_legend(['RAM
Usage %'],loc='upper center',ncol=1)
    memory_usage_plot.push([datetime.now().timestamp()],[[0]])
    memory_usage_plot.push([datetime.now().timestamp()],[[100]])

network_speed_plot=ui.line_plot(n=2,figsize=(4.7,2.47)).with_legend(['Down
load Speed','Upload Speed'],loc='upper center',ncol=2)
    network_speed_plot.push([datetime.now().timestamp()],[[0],[0]])
    network_speed_plot.push([datetime.now().timestamp()],[[100],[100]])

if __name__ in {"__main__","_mp_main__"}:
    ui.timer(1,update_ui,active=True)
    ui.run(title='System Resources Analysis',favicon='💻')

```

## ДОДАТОК В – МЕТОДИКА ТА РЕЛУЗЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМИ

Загальна структура графічного інтерфейсу застосунку наведена нижче у вигляді зображення:

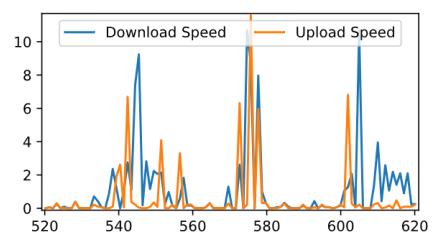
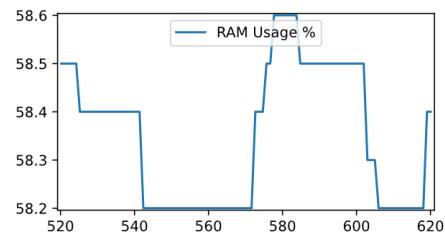
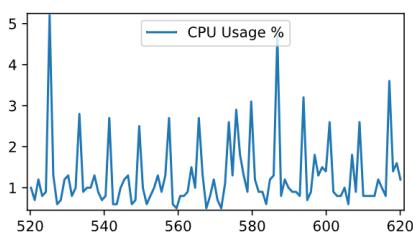


Рисунок В.0 – Структурна схема графічного інтерфейсу застосунку

Методика тестування програми наступна: для кожної умовної секції програми наведено сценарій виконання. Якщо секція має динамічні показники (ті, що змінюються з часом виконання програми), секція має містити дані у різні моменти часу.

Перед тестуванням окремих секцій нижче наведено вигляд застосунку у його цілісному вигляді:

| System   |                    | Processor           |  | Virtual Memory |                   | Disks                |                | Network       |                             |  |  |  |  |
|----------|--------------------|---------------------|--|----------------|-------------------|----------------------|----------------|---------------|-----------------------------|--|--|--|--|
| Property | Value              | Property            | Value  | Property       | Value             | Systemnyj            | ▼              | Ethernet      | ▲                           |  |  |  |  |
| Type     | Windows            | Name                | AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD | Total          | 15.93 GB          | Robochyj             | ▲              | Ethernet Data | Ethernet                    |  |  |  |  |
| User     | Oleg               | Platform            | AMD64  | Available      | 6.63 GB           | Robochyj Data        | Robochyj       |               |                             |  |  |  |  |
| Release  | 11                 | Cores               | 6  | Used           | 9.30 GB           |                      |                |               |                             |  |  |  |  |
| Version  | 10.0.22631         | Threads             | 12   | Percentage     | 58.4%             |                      |                |               |                             |  |  |  |  |
| Machine  | AMD64              | Frequencies (MHz)   |  | Usage (%)      |                   | Virtual Memory Usage |                |               |                             |  |  |  |  |
| Booted   | 6.12.2024 08:09:57 | Property            | Value  | Property       | Value             | 58.4                 | ▼              | Ethernet Data | Ethernet                    |  |  |  |  |
|          |                    | Min                 | 0  | Core 0         | 4.7               | Swap Memory          |                |               |                             |  |  |  |  |
|          |                    | Max                 | 3600   | Core 1         | 7.8               | Property             | Value          |               |                             |  |  |  |  |
|          |                    | Current             | 3600   | Core 2         | 0                 | Total                | 13.00 GB       |               |                             |  |  |  |  |
|          |                    | Processor Frequency |  | Core 3         | 0                 | Free                 | 12.87 GB       |               |                             |  |  |  |  |
|          |                    | 3600                | Core 4   | 0              | Used              | 134.07 MB            | Robochyj Space | Robochyj      | vEthernet (Default Switch)  |  |  |  |  |
|          |                    |                     | Core 5   | 1.6            | Percentage        | 1.0%                 |                |               |                             |  |  |  |  |
|          |                    |                     | Core 6   | 0              | Swap Memory Usage |                      |                |               |                             |  |  |  |  |
|          |                    |                     | Core 7   | 0              | 1                 | ▼                    |                |               |                             |  |  |  |  |
|          |                    |                     | Core 8   | 0              | Robochyj Usage    |                      |                |               |                             |  |  |  |  |
|          |                    |                     | Core 9   | 0              | 14.2              | ▼                    | Google Drive   | Google Drive  | Loopback Pseudo-Interface 1 |  |  |  |  |
|          |                    |                     | Core 10  | 0              | Google Drive      |                      |                |               |                             |  |  |  |  |
|          |                    |                     | Core 11  | 0              | Property          | Value                |                |               |                             |  |  |  |  |
|          |                    |                     |  |                | Read              | 16.40 GB             |                |               |                             |  |  |  |  |
|          |                    |                     |  |                | Write             | 9.66 GB              |                |               |                             |  |  |  |  |



### Рисунок В.1 – Вигляд цілої сторінки програми

Нижче наведено результати тестування кожної окремої секції застосунку.

## B1 Системні дані

Інформація про систему є статичною і не оновлюється з часом виконання.

Дані подані у вигляді таблиці та виглядають наступним чином:

| System   |                    |
|----------|--------------------|
| Property | Value              |
| Type     | Windows            |
| User     | Oleg               |
| Release  | 11                 |
| Version  | 10.0.22631         |
| Machine  | AMD64              |
| Booted   | 6.12.2024 08:09:57 |

Рисунок В.2 – Інформація про систему

## B2 Дані про процесор

Процесорні дані розділені на дві підсекції:

- Загальна інформація;

- Оновлювана інформація.

Вигляд загальної інформації такий:

| Property | Value  |
|----------|--|
| Name     | AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD |
| Platform | AMD64  |
| Cores    | 6  |
| Threads  | 12   |

Рисунок В.3 – Загальна інформація про процесор

Оновлювана інформація про процесор містить такі показники:

- Інформація про частоти;
- Дані про завантаженість.

Інформація про частоти виглядає наступним чином у різні моменти часу:

## Frequencies (MHz)

| Property | Value |
|----------|-------|
| Min      | 0     |
| Max      | 3600  |
| Current  | 3600  |

### Processor Frequency



Рисунок В.4 – Інформація про частоти

| Frequencies (MHz) |       |
|-------------------|-------|
| Property          | Value |
| Min               | 0     |
| Max               | 3600  |
| Current           | 3600  |

Processor Frequency



Рисунок В.5 – Інформація про частоти через якийсь час

Інформація про використання ресурсів процесора виглядає наступним чином:

| Property | Value |
|----------|-------|
| Core 0   | 1.5   |
| Core 1   | 1.5   |
| Core 2   | 0     |
| Core 3   | 0     |
| Core 4   | 3.1   |
| Core 5   | 4.6   |
| Core 6   | 0     |
| Core 7   | 0     |
| Core 8   | 0     |
| Core 9   | 1.6   |
| Core 10  | 0     |
| Core 11  | 0     |

Рисунок В.6 – Інформація про завантаженість процесора

| Usage (%) |       |
|-----------|-------|
| Property  | Value |
| Core 0    | 1.5   |
| Core 1    | 13.8  |
| Core 2    | 4.6   |
| Core 3    | 1.5   |
| Core 4    | 1.5   |
| Core 5    | 3.1   |
| Core 6    | 0     |
| Core 7    | 1.5   |
| Core 8    | 0     |
| Core 9    | 0     |
| Core 10   | 0     |
| Core 11   | 0     |

Рисунок В.7 – Інформація про завантаженість процесора після якогось часу та відкривання додаткової програми

### B3 Дані про пам'ять

Дані про пам'ять містять дві умовні секції:

- Дані про оперативну пам'ять;
- Дані про пам'ять підкачки.

Обидві секції є оновлюваними, тому екранні формули для обидвох наведені у різні моменти часу.

Вигляд секції даних про оперативну пам'ять:

| Virtual Memory |          |
|----------------|----------|
| Property       | Value    |
| Total          | 15.93 GB |
| Available      | 6.57 GB  |
| Used           | 9.36 GB  |
| Percentage     | 58.7%    |

Virtual Memory Usage

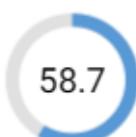


Рисунок В.8 – Дані про оперативну пам'ять

| Virtual Memory |          |
|----------------|----------|
| Property       | Value    |
| Total          | 15.93 GB |
| Available      | 5.70 GB  |
| Used           | 10.23 GB |
| Percentage     | 64.2%    |

### Virtual Memory Usage

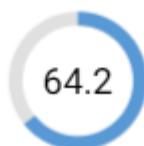


Рисунок В.9 – Дані про оперативну пам’ять через якийсь час та після запуску додаткових програм

Дані про пам’ять підкачки виглядать так:

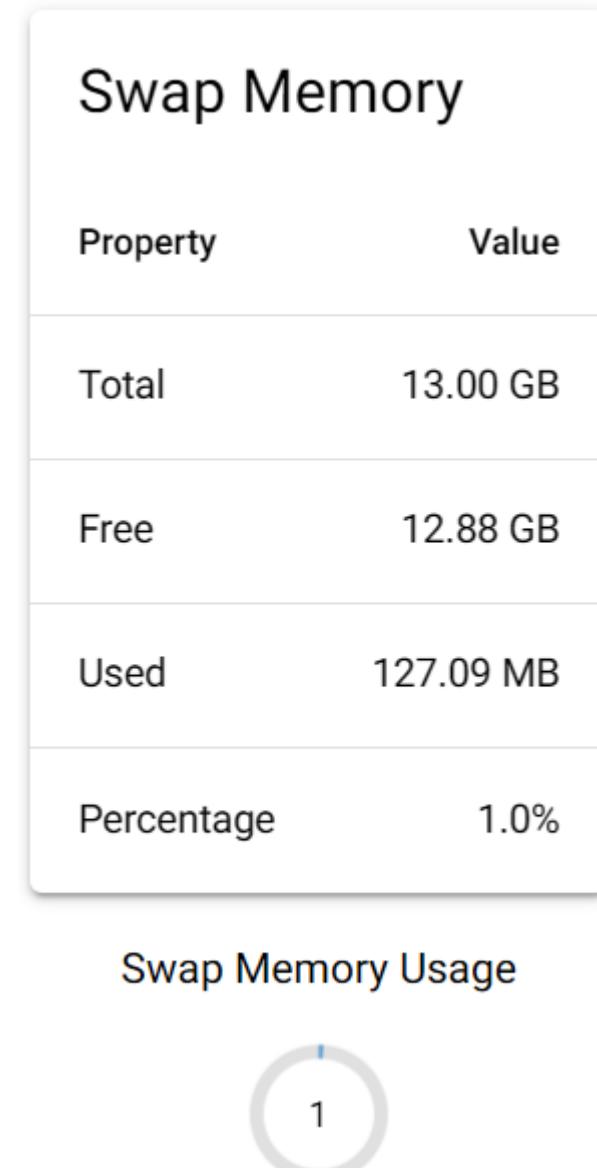


Рисунок В.10 – Дані про пам'ять підкачки

| Property   | Value    |
|------------|----------|
| Total      | 13.00 GB |
| Free       | 12.91 GB |
| Used       | 89.58 MB |
| Percentage | 0.7%     |

### Swap Memory Usage



Рисунок В.11 – Дані про пам’ять підкачки через якийсь час

### B4 Дані про накопичувачі

Інформація про накопичувачі згрупована у одну коробочку інтерфейсу. Інформація про кожен накопичувач схована у випадному меню. Коробка з даними про накопичувачі з усіма окремим у згорнутому вигляді виглядає так:

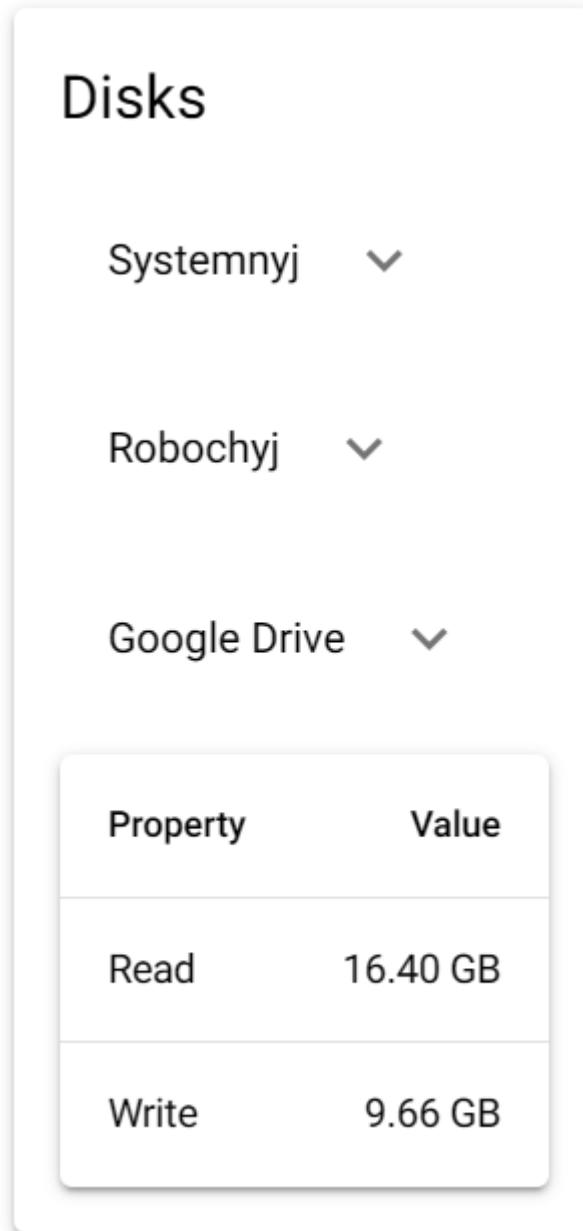


Рисунок В.12 – Інформація про накопичувачі

Дані про кожен накопичувач у розгорнутому вигляді виглядають так:

## Systemnyj

### Systemnyj Data

| Property    | Value     |
|-------------|-----------|
| Device      | C:\       |
| Name        | Systemnyj |
| File system | NTFS      |

### Systemnyj Space

| Property   | Value     |
|------------|-----------|
| Total      | 930.65 GB |
| Used       | 159.44 GB |
| Free       | 771.21 GB |
| Percentage | 17.1%     |

### Systemnyj Usage



Рисунок В.13 – Інформація про системний накопичувач

Robochyj ^

### Robochyj Data

| Property    | Value    |
|-------------|----------|
| Device      | E:\      |
| Name        | Robochyj |
| File system | NTFS     |

### Robochyj Space

| Property   | Value     |
|------------|-----------|
| Total      | 111.79 GB |
| Used       | 15.88 GB  |
| Free       | 95.91 GB  |
| Percentage | 14.2%     |

### Robochyj Usage

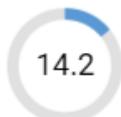


Рисунок В.14 – Інформація про робочий диск

## Google Drive ^

### Google Drive Data

| Property    | Value        |
|-------------|--------------|
| Device      | G:\          |
| Name        | Google Drive |
| File system | FAT32        |

### Google Drive Space

| Property   | Value    |
|------------|----------|
| Total      | 15.00 GB |
| Used       | 1.41 GB  |
| Free       | 13.59 GB |
| Percentage | 9.4%     |

### Google Drive Usage



Рисунок В.15 – Інформація про мережевий накопичувач

## B5 Інформація про мережу

Інформація про мережу також містить інформацію про кожен мережевий пристрій, згорнуту за замовчуванням.

Загальна інформація про мережу зі всіма пристроями у згорнутому вигляді виглядає так:

| Network                     |           |
|-----------------------------|-----------|
| Ethernet                    | ▼         |
| Ethernet 2                  | ▼         |
| vEthernet (Default Switch)  | ▼         |
| Loopback Pseudo-Interface 1 | ▼         |
| Property Value              |           |
| Sent                        | 635.15 MB |
| Received                    | 2.05 GB   |

Рисунок В.16 – Інформація про мережу

Інформація про мережеві пристрой, кожен у розгорнутому вигляді, виглядає так:

## Ethernet

**Ethernet Data**

| Property     | Value         |
|--------------|---------------|
| Ip address   | 192.168.31.70 |
| Netmask      | 255.255.255.0 |
| Broadcast ip |               |

Рисунок В.17 – Інформація про мережевий пристрій

## Ethernet 2 ^

**Ethernet 2 Data**

| Property     | Value         |
|--------------|---------------|
| Ip address   | 192.168.56.1  |
| Netmask      | 255.255.255.0 |
| Broadcast ip |               |

Рисунок В.18 – Інформація про мережевий пристрій

## vEthernet (Default Switch)



### vEthernet (Default Switch) Data

| Property     | Value         |
|--------------|---------------|
| Ip address   | 172.24.64.1   |
| Netmask      | 255.255.240.0 |
| Broadcast ip |               |

Рисунок В.19 – Інформація про мережевий пристрій

## Loopback Pseudo-Interface 1



### Loopback Pseudo-Interface 1 Data

| Property     | Value     |
|--------------|-----------|
| Ip address   | 127.0.0.1 |
| Netmask      | 255.0.0.0 |
| Broadcast ip |           |

Рисунок В.20 – Інформація про мережевий пристрій

### B6 Графічні елементи

На нижній частині сторінки розташовані графіки динамічних показників системи:

- Застосованість процесора
- Використованість пам'яті
- Швидкості мережі

Кожен графік наведено на рисунку нижче:

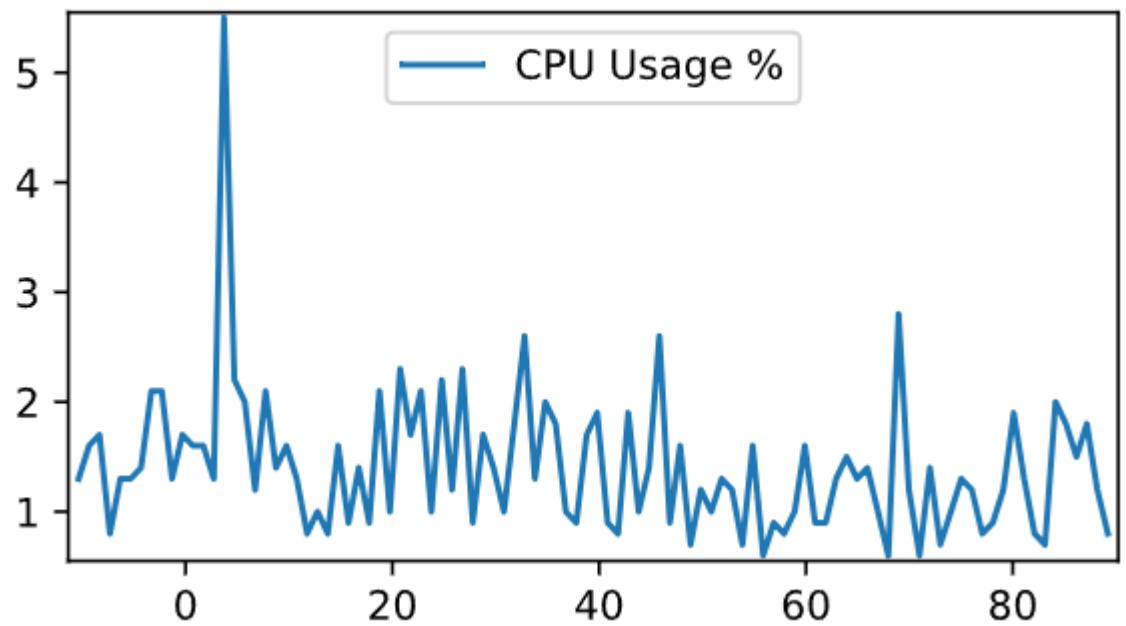


Рисунок В.21 – Графік застосованості процесора

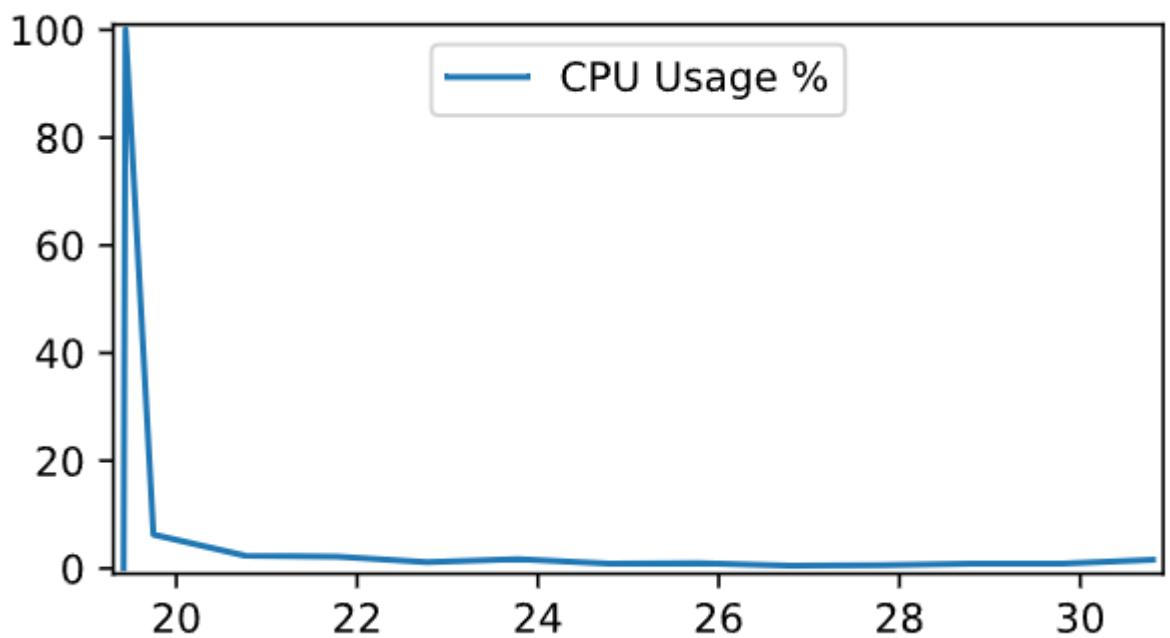


Рисунок В.22 – Графік використання процесора на початку роботу програми

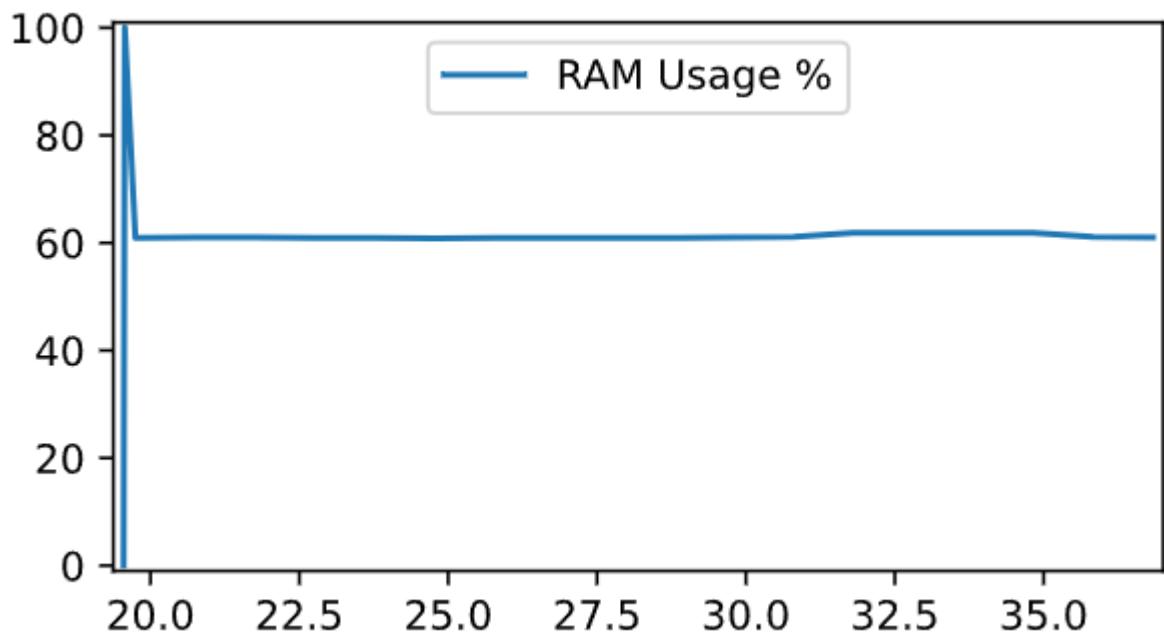


Рисунок В.23 – Графік застосованості пам'яті на початку роботу програми

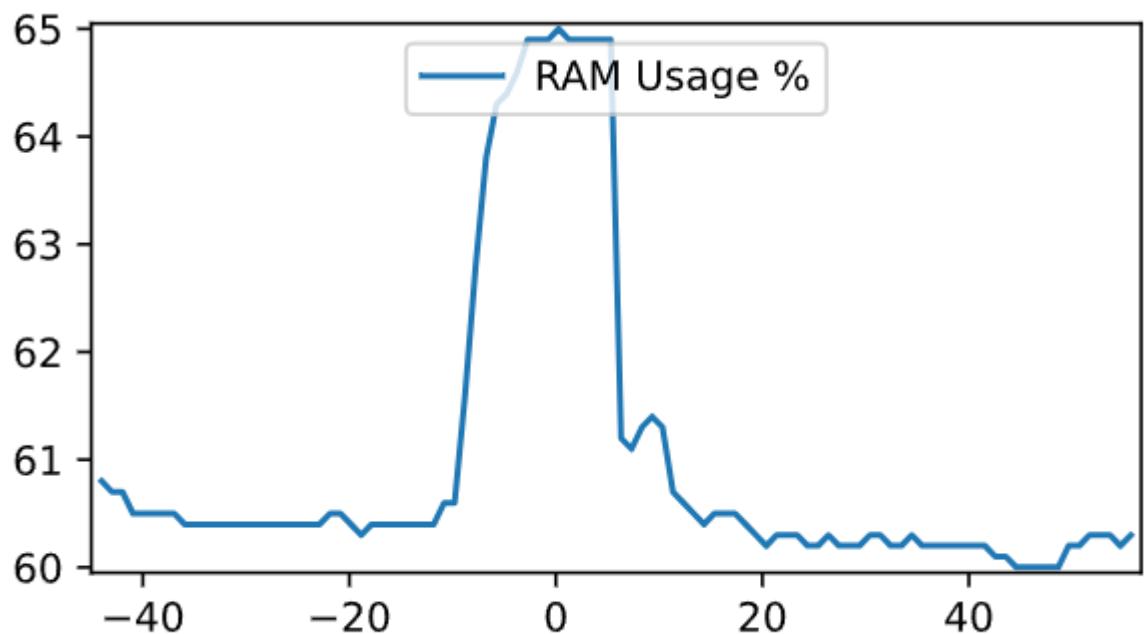


Рисунок В.24 – Графік застосування пам'яті через якийсь час та після  
відкриття програми

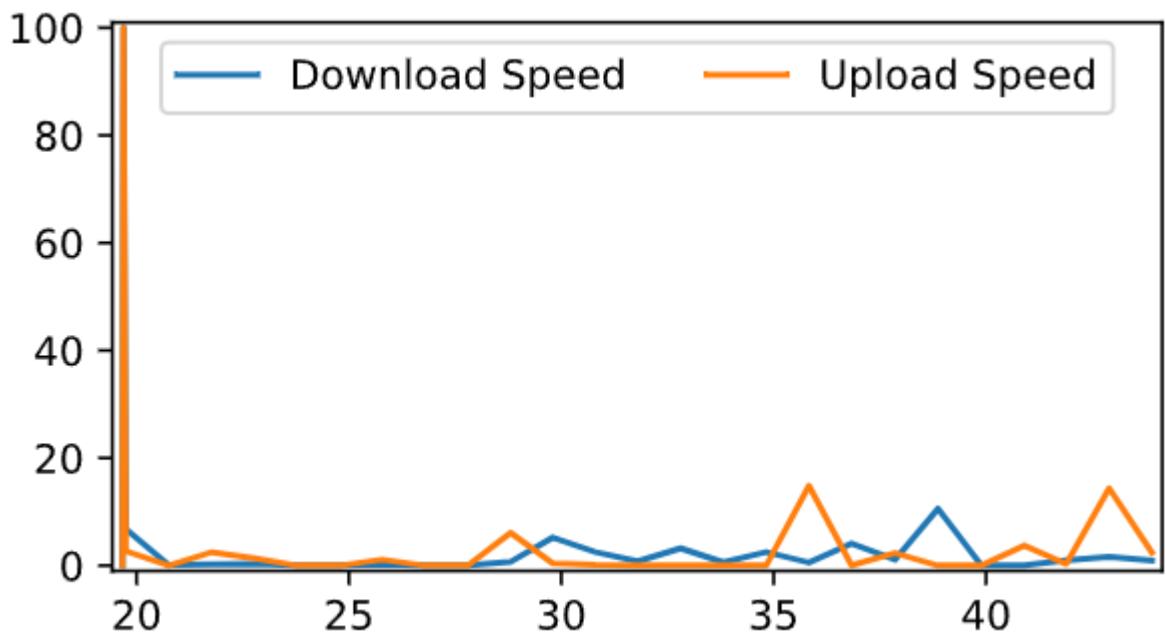


Рисунок В.25 – Графік швидкостей мережі на початку роботу програми

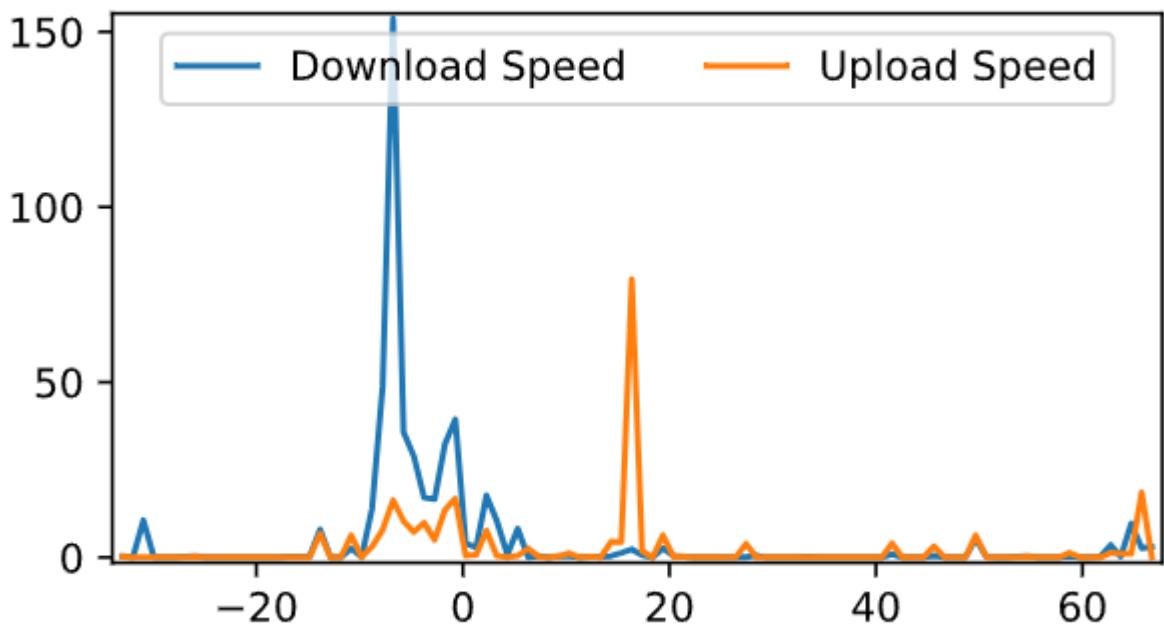


Рисунок В.26 – Графік швидкостей мережі через якийсь час та після відкривання програми