**Розділ 1. Аналітичний огляд**

1. **Аналіз методів, відомих розробок для моніторингу системних ресурсів**

Розглянемо три популярні програми для моніторингу системних ресурсів: їхній опис, функціонал, переваги та недоліки.

**1.1 HWMonitor**

Зображення, що містить текст, знімок екрана, програмне забезпечення, монітор

Вміст, створений ШІ, може бути неправильним.

Рис 1.1 Інтерфейс HWMonitor

1.1.1 Короткий опис:  
HWMonitor – це безкоштовна утиліта для моніторингу апаратних параметрів комп’ютера, таких як температура процесора, напруга, швидкість обертання вентиляторів, а також використання CPU, GPU і пам’яті.

1.1.2 Функціонал:  
- відображення температури, напруги та частоти роботи апаратних компонентів;  
- моніторинг у реальному часі;  
- підтримка широкого спектра датчиків апаратного забезпечення;  
- експорт даних у текстовий файл;  
- відображення мінімальних і максимальних значень параметрів за сесію.

1.1.3 Переваги та недоліки:  
Переваги:

* простий і зрозумілий інтерфейс;
* безкоштовний доступ до базового функціоналу;
* підтримка більшості сучасного обладнання;
* легке споживання ресурсів.

Недоліки:

* відсутність автоматичних сповіщень про критичні стани;
* немає функцій активного управління ресурсами (наприклад, завершення процесів);
* обмежені можливості налаштування;
* відсутність графіків чи детальної історії використання;
* деякі датчики можуть некоректно відображатися на нових системах.  
  Вартість: Безкоштовно, є платна версія HWMonitor Pro за $22,95 (доларів США) із додатковими функціями (віддалений моніторинг).

**1.2 Task Manager (Диспетчер завдань Windows)**

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

Рис 1.2 Інтерфейс Task Manager (Диспетчер завдань Windows)

1.2.1 Короткий опис:  
Task Manager – вбудований інструмент Windows для моніторингу та управління системними ресурсами, процесами й продуктивністю системи.

1.2.2 Функціонал:  
- відображення використання CPU, пам’яті, диска та мережі в реальному часі;  
- список запущених процесів із можливістю їх завершення;  
- моніторинг продуктивності через графіки;  
- управління автозавантаженням програм;  
- перегляд мережевої активності та споживання ресурсів окремими програмами.

1.2.3 Переваги та недоліки:  
Переваги:

* вбудований у систему, не потребує встановлення;
* простий у використанні для базового моніторингу;
* можливість завершувати процеси вручну;
* безкоштовний.

Недоліки:

* відсутність детальних сповіщень чи автоматизації;
* обмежена інформація про апаратні параметри (наприклад, температура);
* немає журналів чи звітів для аналізу;

**1.3 AIDA64**

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

Рис 1.3 Інтерфейс AIDA64

1.3.1 Короткий опис:  
AIDA64 – це потужне програмне забезпечення для діагностики, моніторингу та бенчмаркінгу системних ресурсів, розроблене компанією FinalWire Ltd. Воно надає детальну інформацію про апаратне забезпечення, дозволяє відстежувати продуктивність у реальному часі та проводити стрес-тести для оцінки стабільності системи.

1.3.2 Функціонал:  
- моніторинг завантаження процесора, оперативної пам’яті, дисків і мережі в реальному часі;  
- відстеження температури, напруги та швидкості обертання вентиляторів;  
- детальна інформація про апаратні компоненти (CPU, GPU, материнська плата тощо);  
- бенчмаркінг процесора, пам’яті та накопичувачів;  
- стрес-тестування системи для перевірки стабільності;  
- експорт даних у формати HTML, CSV, XML;  
- підтримка віддаленого моніторингу через смартфони та планшети.

1.3.3 Переваги та недоліки:  
Переваги:

* найточніший у своєму класі механізм виявлення апаратного забезпечення (база даних із понад 250 000 пристроїв);
* широкий набір інструментів для моніторингу та тестування;
* зручний інтерфейс із можливістю кастомізації (SensorPanel, OSD);
* підтримка більш ніж 250 сенсорних пристроїв для моніторингу температури та напруги;
* можливість створення детальних звітів.

Недоліки:

* повна функціональність доступна лише в платній версії (безкоштовна версія обмежена 30-денним пробним періодом);
* високе споживання ресурсів під час стрес-тестів;
* відсутність автоматичного управління ресурсами (наприклад, завершення процесів);
* складність для новачків через велику кількість опцій;
* ціна може бути високою для домашніх користувачів (AIDA64 Extreme коштує $59.95, Engineer – $219.90).  
  Вартість: Безкоштовна пробна версія на 30 днів, платні версії від $59.95 (Extreme) до $219.90 (Engineer).

**1.4 Resource Monitor (Монітор ресурсів Windows)**

**Зображення, що містить текст, знімок екрана, програмне забезпечення, Комп’ютерна піктограма

Вміст, створений ШІ, може бути неправильним.**

Рис. 1.4. Інтерфейс Resource Monitor (Монітор ресурсів Windows)

1.4.1 Короткий опис

Resource Monitor – це вбудований інструмент Windows, який є частиною операційної системи і доступний через Task Manager (Диспетчер завдань) або через прямий запуск (resmon.exe). Він призначений для детального моніторингу системних ресурсів і використання їх окремими процесами, надаючи глибший аналіз, ніж Task Manager.

1.4.2 Функціонал

* Відображення використання CPU, пам’яті, диска та мережі в реальному часі з розбивкою по процесах;
* Графіки використання ресурсів для кожної категорії (CPU, RAM, диск, мережа);
* Детальна інформація про кожен процес: кількість потоків, використання пам’яті, дискові операції, мережевий трафік;
* Фільтрація процесів за критеріями (наприклад, процеси, що найбільше використовують мережу);
* Перегляд асоційованих дескрипторів і модулів для кожного процесу;
* Моніторинг загальної продуктивності системи, включаючи час очікування CPU і статистику дискових операцій.

1.4.3 Переваги та недоліки

Переваги:

* Вбудований у Windows, не потребує встановлення;
* Детальний аналіз використання ресурсів окремими процесами;
* Зручні графіки для оцінки продуктивності в реальному часі;
* Безкоштовний;
* Дозволяє швидко виявляти проблемні процеси (наприклад, ті, що спричиняють високе навантаження на диск).

Недоліки:

* Відсутність автоматичних сповіщень про критичні стани;
* Немає функцій експорту даних чи створення звітів;
* Обмежена інформація про апаратні параметри, такі як температура чи напруга;
* Не підтримує активне управління ресурсами (наприклад, завершення процесів можливе лише через Task Manager);
* Інтерфейс може бути складним для новачків через велику кількість технічної інформації.

**1.5 MSI Afterburner**

**Зображення, що містить текст, знімок екрана, електроніка, мультимедіа

Вміст, створений ШІ, може бути неправильним.**

Рис. 1.5. Інтерфейс MSI Afterburner

1.5.1 Короткий опис

MSI Afterburner – це популярна програма для моніторингу та розгону відеокарт, розроблена компанією MSI у співпраці з RivaTuner. Вона широко використовується геймерами та ентузіастами для відстеження продуктивності GPU, CPU, RAM і розгону апаратного забезпечення. Програма підтримує відеокарти всіх виробників (NVIDIA, AMD).

1.5.2 Функціонал

* Моніторинг у реальному часі завантаження GPU, CPU, RAM, температури, частоти ядра та пам’яті відеокарти;
* Відображення графіків продуктивності для всіх параметрів;
* Накладання інформації на екран (OSD) під час ігор (наприклад, FPS, температура GPU);
* Розгін відеокарти: налаштування частоти ядра, пам’яті, напруги та швидкості вентиляторів;
* Створення профілів для різних сценаріїв (наприклад, ігровий режим, енергозбереження);
* Запис відео та створення скріншотів через вбудовану утиліту (RivaTuner Statistics Server);
* Підтримка кастомізації інтерфейсу (зміна скінів).

1.5.3 Переваги та недоліки

Переваги:

* Безкоштовний і доступний для всіх користувачів;
* Потужний інструмент для розгону відеокарт із детальним контролем параметрів;
* Зручне накладання інформації (OSD) для моніторингу під час ігор;
* Підтримка всіх сучасних відеокарт (NVIDIA, AMD);
* Можливість кастомізації інтерфейсу та створення профілів;
* Низьке споживання ресурсів під час моніторингу.

Недоліки:

* Відсутність автоматичних сповіщень про критичні стани (наприклад, перегрів GPU);
* Обмежений моніторинг інших компонентів (наприклад, накопичувачів чи мережі);
* Складність для новачків, особливо в налаштуванні розгону;
* Розгін може призвести до перегріву або пошкодження обладнання, якщо неправильно налаштувати;
* Немає функцій управління процесами чи експорту даних у звіти.

Таблиця 1.1. Порівняльний аналіз програм для моніторингу системних ресурсів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Характеристика | HWMonitor | Task Manager | AIDA64 | Resource Monitor | MSI Afterburner |
| Моніторинг CPU | Так | Так | Так | Так | Так |
| Моніторинг RAM | Так | Так | Так | Так | Так |
| Моніторинг GPU (температура) | Так | Ні | Так | Ні | Так |
| Моніторинг мережі | Ні | Так | Так | Так | Ні |
| Моніторинг накопичувачів | Обмежено | Так | Так | Так | Ні |
| Автоматичні сповіщення | Ні | Ні | Обмежено | Ні | Ні |
| Експорт даних | Так | Ні | Так | Ні | Ні |
| Управління процесами | Ні | Так | Ні | Ні | Ні |
| Розгін апаратного забезпечення | Ні | Ні | Ні | Ні | Так |
| Споживання ресурсів | Низьке | Низьке | Високе | Низьке | Низьке |
| Вартість | Безкоштовно | Безкоштовно | Платно ($59.95–$219.90) | Безкоштовно | Безкоштовно |

**Розділ 2. Обгрунтування розробки програмної системи**

**2.1 Короткий опис**

**2.1.1 Назва проекту**

Розробка програмного забезпечення для моніторингу та контролю системних ресурсів операційної системи.

**2.1.2 Мета розробки**

Метою проекту є створення легкого, функціонального програмного забезпечення для моніторингу та управління системними ресурсами (CPU, RAM, GPU, накопичувачі, мережа) у реальному часі. Програма забезпечує стабільну роботу системи шляхом автоматичного виявлення критичних станів, сповіщення користувача та надання інструментів для аналізу й оптимізації продуктивності.

**2.1.3 Актуальність розробки**

Зі зростанням складності апаратного забезпечення, обсягів даних і вимог до продуктивності комп’ютерних систем зростає потреба в ефективних інструментах моніторингу. Багато наявних рішень, таких як HWMonitor і Task Manager, не пропонують автоматичних сповіщень чи детального аналізу всіх компонентів системи, а професійні інструменти, як AIDA64, є дорогими та складними для пересічних користувачів. Розробка програми, яка поєднує простоту, автоматизацію та широкий функціонал, є актуальною для забезпечення стабільності систем у різних сценаріях використання.

**2.2 Проблематика та необхідність розробки**

Користувачі стикаються з такими проблемами під час моніторингу системних ресурсів:

* Відсутність автоматичних сповіщень про критичні стани в безкоштовних програмах, таких як HWMonitor.
* Обмежений функціонал вбудованих інструментів, як Task Manager, який не відображає температуру чи стан накопичувачів.
* Висока ціна та складність професійних рішень, як AIDA64, що робить їх недоступними для домашніх користувачів.
* Надмірне споживання ресурсів деякими програмами під час моніторингу.

Розроблена програма усуває ці недоліки, пропонуючи безкоштовне рішення з автоматичними сповіщеннями, низьким споживанням ресурсів і простим інтерфейсом.

**2.3 Галузь застосування**

Програма призначена для:

* Домашніх користувачів: Оптимізація ПК для ігор чи роботи з ресурсоємними програмами.
* Системних адміністраторів: Моніторинг серверів і робочих станцій у корпоративних мережах.
* Розробників вбудованих систем: Контроль продуктивності в IoT-пристроях.
* Технічних ентузіастів: Аналіз системи для розгону чи тестування компонентів.

Програма підтримує Windows 7, 10, 11 (32/64 біти) і сумісна з більшістю сучасних апаратних компонентів.

**2.4 Порівняння з аналогами**

Розроблена програма поєднує переваги аналогів і усуває їхні недоліки:

* HWMonitor: Програма додає сповіщення та моніторинг мережі, зберігаючи низьке споживання ресурсів. На відміну від HWMonitor, який не має автоматичних сповіщень і графіків історії використання, розроблена програма забезпечує ці функції, що робить її зручнішою для користувачів, які потребують активного контролю.
* Task Manager: Надає детальнішу інформацію про GPU, накопичувачі та можливість експорту звітів. Task Manager не відображає температуру компонентів і не підтримує експорт даних, тоді як розроблена програма усуває ці недоліки, пропонуючи розширені можливості для аналізу.
* AIDA64: Є безкоштовною, простішою у використанні, хоча поступається в бенчмаркінгу. AIDA64 має високу ціну та складність для пересічних користувачів, тоді як розроблена програма пропонує безкоштовне рішення з інтуїтивним інтерфейсом, зберігаючи широкий функціонал.
* Resource Monitor: Додає автоматичні сповіщення, експорт даних і моніторинг температури GPU та накопичувачів. Resource Monitor добре підходить для детального аналізу процесів, але не має функцій сповіщень, експорту звітів і відображення температури, що робить розроблену програму більш універсальною.
* MSI Afterburner: Розширює функціонал моніторингу мережі, накопичувачів і сповіщень, зберігаючи низьке споживання ресурсів. MSI Afterburner фокусується на GPU і розгоні, але не підтримує моніторинг мережі чи накопичувачів і не має автоматичних сповіщень, що обмежує його універсальність порівняно з розробленою програмою.

**2.5 Функціональні та нефункціональні вимоги**

**2.5.1 Функціональні вимоги**

На основі аналізу потреб користувачів і недоліків аналогів, програмне забезпечення включає наступні функції:

1. Моніторинг завантаження процесора в реальному часі

Опис:  
Програма відстежує рівень завантаження процесора з оновленням даних у реальному часі для оцінки продуктивності системи.

Функціональні можливості:

* Відображення відсоткового завантаження CPU для кожного ядра.
* Графічне представлення у вигляді діаграми чи графіка.
* Оновлення даних із заданою частотою (наприклад, кожну секунду).
* Виділення пікових навантажень кольором для швидкого аналізу.

Обґрунтування необхідності:  
Моніторинг CPU є ключовим для оцінки продуктивності системи, оскільки перевантаження процесора може призвести до уповільнення роботи або збоїв. Ця функція дозволяє користувачам (від геймерів до адміністраторів серверів) швидко виявляти проблемні процеси. На відміну від HWMonitor, який не пропонує налаштування сповіщень, і Task Manager, який має обмежені графіки, програма забезпечує детальну візуалізацію та підтримує подальшу автоматизацію (наприклад, сповіщення), що сприяє стабільності системи.

2. Відстеження використання оперативної пам’яті

Опис:  
Програма показує, скільки оперативної пам’яті використовується, щоб користувач міг виявити витоки чи перевантаження.

Функціональні можливості:

* Відображення загального обсягу RAM, використаної та вільної пам’яті.
* Показник у відсотках і мегабайтах/гігабайтах.
* Графік використання RAM у реальному часі.
* Інформація про програми, що споживають найбільше пам’яті.

Обґрунтування необхідності:  
Недостатня кількість вільної оперативної пам’яті може спричинити уповільнення роботи програм або аварійне завершення процесів. Ця функція необхідна для виявлення ресурсоємних додатків і оптимізації роботи системи. Task Manager показує лише базову статистику RAM, тоді як програма пропонує детальні графіки та список споживачів пам’яті, що робить її корисною для користувачів, які працюють із багатозадачними середовищами чи серверами.

3. Інформація про GPU

Опис:  
Програма моніторить температуру відеокарти та графік використання GPU.

Функціональні можливості:

* Отримання даних із апаратних датчиків (за наявності).
* Відображення поточної, мінімальної та максимальної температури.
* Графічне відображення використання GPU.
* Графічне відображення змін температури з часом.

Обґрунтування необхідності:  
GPU є критичним компонентом для геймерів, дизайнерів і користувачів, які працюють із графічними програмами. Перегрів або перевантаження GPU може призвести до зниження продуктивності чи апаратних збоїв. Task Manager має обмежений моніторинг GPU, а HWMonitor не завжди точно відображає температуру нових моделей. Програма заповнює цю прогалину, надаючи детальну інформацію, що допомагає запобігти перегріву та оптимізувати роботу відеокарти.

4. Аналіз мережевої активності

Опис:  
Програма відстежує вхідний і вихідний мережевий трафік для виявлення аномалій чи надмірного споживання.

Функціональні можливості:

* Показ швидкості завантаження та віддачі (в Мбіт/с).
* Графік мережевої активності в реальному часі.
* Список програм, що використовують мережу, із їхнім споживанням.
* Попередження про незвичну активність (наприклад, різке зростання трафіку).

Обґрунтування необхідності:  
Мережева активність впливає на продуктивність системи, особливо в корпоративних мережах або при обмеженій пропускній здатності. Аномалії в трафіку можуть вказувати на шкідливе ПЗ чи неефективні програми. HWMonitor не підтримує моніторинг мережі, а Task Manager показує лише базові дані. Ця функція дозволяє користувачам виявляти проблеми, оптимізувати мережеве навантаження та підвищувати безпеку системи.

5. Моніторинг стану накопичувачів (HDD/SSD)

Опис:  
Програма перевіряє стан жорстких дисків і твердотільних накопичувачів для оцінки їхньої працездатності.

Функціональні можливості:

* Відображення вільного та зайнятого місця на дисках.
* Інформація про температуру та стан здоров’я (SMART-аналіз).
* Попередження про низький рівень вільного місця (наприклад, <10%).
* Графік активності читання/запису на диск.

Обґрунтування необхідності:  
Накопичувачі є критичними для зберігання даних, і їхній збій може призвести до втрати інформації. Недостатній вільний простір або перегрів знижують продуктивність. Task Manager не відображає температуру чи SMART-стан, а AIDA64 робить це лише в платній версії. Ця функція необхідна для своєчасного виявлення проблем із дисками, що забезпечує надійність і довговічність апаратного забезпечення.

6. Перегляд загальної інформації про систему

Опис:  
Програма надає користувачу базову інформацію про апаратне та програмне забезпечення системи.

Функціональні можливості:

* Виведення даних про процесор (модель, частота), обсяг RAM, відеокарту.
* Інформація про операційну систему (версія, розрядність).
* Список встановлених драйверів і їхній стан.
* Можливість копіювання даних у буфер обміну.

Обґрунтування необхідності:  
Загальна інформація про систему потрібна для швидкої діагностики апаратного та програмного забезпечення, особливо для технічних спеціалістів і ентузіастів. HWMonitor і Task Manager надають обмежені дані про систему, тоді як програма пропонує ширший набір інформації, включаючи стан драйверів. Це спрощує аналіз конфігурації та виявлення застарілих компонентів.

7. Відображення часу роботи системи

Опис:  
Програма показує, як довго система працює без перезавантаження, для оцінки стабільності.

Функціональні можливості:

* Виведення часу роботи у форматі "дні:години:хвилини".
* Попередження про тривалу роботу без перезавантаження (наприклад, >7 днів).
* Історія перезавантажень (дата й час).
* Відображення часу останнього запуску системи.

Обґрунтування необхідності:  
Тривала робота системи без перезавантаження може призвести до накопичення помилок і зниження продуктивності, особливо на серверах. Жоден із аналогів (HWMonitor, Task Manager, AIDA64) не відображає історію перезавантажень чи попереджає про тривалу роботу. Ця функція допомагає користувачам планувати перезавантаження, забезпечуючи стабільність і надійність системи.

8. Сповіщення про високе завантаження CPU, RAM

Опис:  
Програма інформує користувача про критичні стани, коли ресурси системи перевантажені.

Функціональні можливості:

* Налаштування порогів для CPU та RAM (наприклад, 90% завантаження).
* Візуальні (спливаючі вікна) та звукові сповіщення.
* Рекомендації щодо зменшення навантаження (наприклад, закриття програм).
* Журнал сповіщень із часом і причиною.

Обґрунтування необхідності:  
Автоматичні сповіщення є критично важливими для запобігання перевантаженням і збоїв, особливо в серверних чи високонавантажених системах. HWMonitor і Task Manager не мають цієї функції, а AIDA64 пропонує лише обмежені сповіщення в платній версії. Ця функція підвищує автоматизацію та дозволяє користувачам швидко реагувати на проблеми, що сприяє стабільності.

9. Експорт даних у текстовий файл

Опис:  
Програма дозволяє зберігати статистику й логи для подальшого аналізу чи передачі.

* Функціональні можливості:
* Експорт поточних даних про CPU, RAM, мережу тощо у TXT-файл.
* Можливість вибору часового діапазону для експорту.
* Автоматичне створення звітів за розкладом.
* Збереження файлів із датою та часом у назві (наприклад, "Report\_2025-03-25.txt").

Обґрунтування необхідності:  
Експорт даних необхідний для аналізу продуктивності системи в довгостроковій перспективі, наприклад, для системних адміністраторів чи технічної підтримки. Task Manager не підтримує експорт, а HWMonitor і AIDA64 мають обмежені формати звітів. Ця функція забезпечує гнучкість для документування стану системи та обміну даними, що є важливим для професійного використання.

10. Перегляд активних процесів

Опис:  
Програма показує список запущених процесів із деталями їхнього впливу на систему.

Функціональні можливості:

* Список процесів із назвою, ID, споживанням CPU та RAM.
* Можливість завершувати процеси вручну.
* Сортування за рівнем споживання ресурсів.
* Підсвічування процесів із надмірним навантаженням.

Обґрунтування необхідності:  
Перегляд і управління процесами дозволяють користувачам швидко виявляти та завершувати ресурсоємні програми, що знижують продуктивність. Task Manager пропонує подібний функціонал, але без підсвічування проблемних процесів. HWMonitor і AIDA64 не дозволяють завершувати процеси. Ця функція є ключовою для активного управління системними ресурсами, що відповідає меті програми — забезпечення стабільності.

**2.5.2 Нефункціональні вимоги**

* Інтерфейс: Інтуїтивно зрозумілий графічний інтерфейс із підтримкою української мови, налаштуванням частоти оновлення даних і порогових значень сповіщень.
* Продуктивність: Споживання CPU < 1%, RAM < 50 МБ для мінімального впливу на систему.
* Сумісність: Підтримка Windows 7, 10, 11 (32/64 біти), сумісність із апаратними компонентами Intel, AMD, NVIDIA.
* Надійність: Стабільна робота без збоїв під час тривалого моніторингу, захист від витоку даних.
* Безпека: Відсутність доступу до даних користувача без дозволу, захист від несанкціонованого завершення процесів.

**2.6 Очікувані результати**

Програма забезпечить:

* Підвищення стабільності систем шляхом своєчасного виявлення перевантажень.
* Зручний моніторинг завдяки автоматизації та інтуїтивному інтерфейсу.
* Економію коштів порівняно з платними аналогами, як AIDA64.
* Гнучкість для домашніх користувачів, адміністраторів і розробників.

**Розділ 3. Розробка програмної системи**

**3.1 Вибір мови програмування**

Для розробки програми обрано мову програмування Python. Цей вибір ґрунтується на таких ключових факторах:

* Простота та читабельність: Python має зрозумілий синтаксис, що спрощує написання, читання та підтримку коду. Це особливо важливо для проєктів з обмеженим часом на розробку.
* Кросплатформність: Python підтримує роботу на різних операційних системах (Windows, Linux, macOS), що забезпечує сумісність із Windows 7, 10, 11 (32/64 біти).
* Широка екосистема бібліотек: Python пропонує численні бібліотеки для роботи з системними ресурсами (psutil), графічними інтерфейсами (Tkinter) і візуалізацією даних (matplotlib), що полегшує реалізацію функціональних вимог.
* Підтримка взаємодії із зовнішніми утилітами: Python дозволяє легко викликати зовнішні програми, такі як smartctl.exe, через модуль subprocess, що необхідно для SMART-аналізу накопичувачів.
* Швидке прототипування: Завдяки простоті та великій кількості готових інструментів, Python дозволяє швидко створювати робочі прототипи.
* Спільнота та документація: Велика спільнота розробників і доступна документація допомагають вирішувати технічні питання.

Альтернативи: Розглядалися мови, такі як C++ і Java. C++ має складніший синтаксис і вимагає більше часу на розробку, а Java менш підходить для легких застосунків через більші витрати ресурсів. Python, навпаки, відповідає вимозі про низьке споживання ресурсів і забезпечує швидший старт розробки.

**3.2 Технології, бібліотеки та зовнішні утиліти**

Для реалізації програми використано набір технологій, бібліотек і зовнішніх утиліт, які оптимально відповідають поставленим завданням.

3.2.1 Tkinter

* Роль: Створення графічного інтерфейсу користувача (GUI).
* Чому обрано: Tkinter є частиною стандартної бібліотеки Python, що усуває потребу в додаткових залежностях. Це спрощує встановлення та використання програми.
* Функціонал: Використовується для створення головного вікна, вкладок (наприклад, для CPU, RAM, дисків, мережі), кнопок (експорт даних, завершення процесів) і сповіщень.
* Переваги: Легкість, інтеграція з matplotlib для графіків, відповідність вимозі про інтуїтивно зрозумілий інтерфейс.

3.2.2 Matplotlib

* Роль: Візуалізація даних у реальному часі через графіки.
* Чому обрано: Matplotlib — одна з найпотужніших бібліотек для побудови графіків у Python, із підтримкою інтеграції в Tkinter через FigureCanvasTkAgg.
* Функціонал: Створює графіки завантаження CPU, RAM, мережі тощо у відповідних вкладках програми.
* Переваги: Гнучкість, можливість оновлення графіків у реальному часі, відповідність вимозі про графічне представлення даних.

3.2.3 Psutil

* Роль: Збір даних про системні ресурси.
* Чому обрано: Psutil — стандартна бібліотека для моніторингу системних ресурсів у Python із простим API для доступу до інформації про CPU, пам’ять, диски та мережу.
* Функціонал: Використовується в класі ResourceMonitor для отримання даних через методи, як-от psutil.cpu\_percent(), psutil.virtual\_memory(), psutil.disk\_usage().
* Переваги: Кросплатформність, низьке споживання ресурсів, відповідність вимозі про стабільність і надійність.

3.2.4 Threading

* Роль: Забезпечення асинхронного моніторингу.
* Чому обрано: Модуль threading дозволяє виконувати моніторинг у фоновому потоці, не блокуючи графічний інтерфейс.
* Функціонал: Клас ResourceMonitor запускає моніторинг у окремому потоці, зберігаючи чутливість SystemMonitorGUI до дій користувача.
* Переваги: Забезпечує стабільність інтерфейсу та низьке споживання ресурсів.

3.2.5 Subprocess (для використання smartctl.exe)

* Роль: Виклик зовнішньої утиліти smartctl.exe для SMART-аналізу накопичувачів.
* Чому обрано: Модуль subprocess є частиною стандартної бібліотеки Python і дозволяє легко запускати зовнішні програми та отримувати їхній вивід. Це необхідно для інтеграції з smartctl.exe.
* Функціонал: Використовується в модулі monitor.py для виклику smartctl.exe із потрібними параметрами, щоб отримати SMART-дані накопичувачів, такі як стан здоров’я, температура, кількість відпрацьованих годин тощо.
* Переваги: Забезпечує доступ до низькорівневих даних накопичувачів, що відповідає функціональній вимозі про SMART-аналіз.

3.2.6 Smartctl.exe

* Роль: Отримання SMART-даних накопичувачів.
* Чому обрано: Smartctl.exe є частиною пакета smartmontools, який є стандартним інструментом для роботи з SMART-даними на різних платформах. Він надає детальну інформацію про стан накопичувачів, що неможливо отримати через psutil у повному обсязі.
* Функціонал: Використовується для моніторингу параметрів накопичувачів, таких як стан здоров’я ("PASSED" або "FAILED"), температура, кількість циклів увімкнення/вимкнення, а також попередження про потенційні проблеми (наприклад, перевищення порогу зносу SSD). Утиліта викликається через subprocess.run() із параметром --all для отримання повних даних.
* Переваги:
  + Надійність: Smartctl є промисловим стандартом для роботи з SMART-даними.
  + Доступність: Безкоштовна утиліта, яка легко інтегрується в проєкт.
  + Відповідність вимозі про моніторинг накопичувачів із SMART-аналізом.
* Обмеження: Для роботи smartctl.exe необхідно встановити пакет smartmontools на систему користувача, що додає залежність. Однак це виправдано, адже без цього інструменту повноцінний SMART-аналіз неможливий.
* Альтернативи: Розглядалися бібліотеки, такі як pySMART, але вони менш надійні та мають обмежену підтримку в Windows. Smartctl.exe є більш універсальним і точним рішенням.

**3.3 Структурна схема програми**

Структурна схема програми (Рис. 3.1) відображає модульну архітектуру системи моніторингу системних ресурсів, включаючи основні модулі, їхні функції та зв’язки між ними. Вона створена з використанням нотації UML (діаграма компонентів) і демонструє, як організовано взаємодію між різними частинами програми. Схема включає п’ять основних модулів: main.py, gui.py, monitor.py, utilities.py і config.py, а також зв’язки між ними, які показують потік даних і залежності.

Зображення, що містить схема, текст, ряд, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рис. 3.1. Структурна схема програми для моніторингу системних ресурсів

Опис модулів

Програма має модульну структуру, що складається з п’яти основних модулів, кожен із яких виконує чітко визначену роль. Такий підхід забезпечує розподіл відповідальності, полегшує розробку, тестування та подальше розширення системи.

* Модуль **main.py**: Точка входу програми. Відповідає за початкове налаштування системи, запускаючи графічний інтерфейс і модуль моніторингу. Цей модуль координує роботу всієї програми, забезпечуючи її правильний запуск і завершення.
* Модуль **gui.py**: Відповідає за створення та управління графічним інтерфейсом користувача. Реалізує відображення даних про системні ресурси (CPU, RAM, GPU, накопичувачі, мережа, системна інформація), включаючи вкладки, графіки, сповіщення та кнопки для взаємодії з користувачем, що відповідає функціональним вимогам.
* Модуль **monitor.py**: Виконує моніторинг системних ресурсів у реальному часі. Збирає дані про завантаження CPU, RAM, мережу та накопичувачі за допомогою бібліотеки psutil і утиліти smartctl.exe для SMART-аналізу. Передає зібрані дані в модуль інтерфейсу для відображення.
* Модуль **utilities.py**: Містить допоміжні функції, такі як логування подій, експорт даних у текстовий файл, побудова графіків і управління процесами. Використовується іншими модулями для виконання другорядних операцій.
* Модуль **config.py**: Зберігає конфігураційні параметри програми, такі як інтервал оновлення даних, порогові значення для сповіщень і шлях до утиліти smartctl.exe. Забезпечує централізоване налаштування для всіх модулів.

Опис взаємодії між модулями

Структурна схема програми (Рис. 3.1) ілюструє взаємодію між модулями, яка забезпечує функціонування системи. Модуль main.py виступає координатором, запускаючи графічний інтерфейс і модуль моніторингу. Після запуску модуль моніторингу періодично збирає дані про системні ресурси та передає їх у модуль інтерфейсу через механізм зворотного виклику, що дозволяє оновлювати відображення даних у реальному часі.

Модуль інтерфейсу активно взаємодіє з допоміжними функціями, викликаючи їх для побудови графіків, експорту звітів і завершення процесів. Модуль моніторингу також використовує допоміжні функції для запису подій у журнал, що полегшує налагодження та аналіз роботи програми. Усі модулі отримують налаштування з модуля конфігурації, що забезпечує єдине джерело параметрів, таких як інтервал оновлення чи порогові значення для сповіщень. Ця взаємодія дозволяє системі працювати як єдине ціле, ефективно виконуючи поставлені завдання.

**3.4 Реалізація модулів**

3.4.1 Модуль **main.py**

Роль: Точка входу програми.

Реалізація:

* + Імпортує класи SystemMonitorGUI з gui.py і ResourceMonitor з monitor.py.
  + Функція main() створює екземпляри цих класів, викликає setup\_gui() для ініціалізації графічного інтерфейсу і start\_monitoring() для запуску моніторингу, після чого запускає головний цикл Tkinter через root.mainloop().

Взаємодія: Координує запуск gui.py і monitor.py, забезпечуючи початкову ініціалізацію програми.

Фрагмент коду:

from gui import SystemMonitorGUI

from monitor import ResourceMonitor

def main():

gui = SystemMonitorGUI()

monitor = ResourceMonitor(gui.update\_gui)

gui.setup\_gui()

monitor.start\_monitoring()

gui.root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

3.4.2 Модуль **gui.py**

Роль: Управління графічним інтерфейсом.

Реалізація:

* + Містить клас SystemMonitorGUI, який ініціалізує головне вікно Tkinter (root), створює вкладки через ttk.Notebook для відображення даних (CPU, RAM, GPU, диски, мережа, системна інформація).
  + Використовує matplotlib для побудови графіків через FigureCanvasTkAgg (наприклад, графіки завантаження CPU).
  + Метод update\_gui(data) оновлює інтерфейс на основі даних, отриманих через callback від ResourceMonitor. Наприклад, оновлює графіки, текстові поля та викликає show\_alert() для сповіщень.
  + Використовує функції з utilities.py для оновлення списків процесів (update\_process\_list()), завершення процесів (kill\_process()) та експорту даних (export\_data()).

Взаємодія:

* + Отримує дані від monitor.py через callback.
  + Викликає utilities.py для допоміжних функцій.
  + Читає параметри з config.py, наприклад, UPDATE\_INTERVAL.

3.4.3 Модуль **monitor.py**

Роль: Моніторинг системних ресурсів у реальному часі.

Реалізація:

* Містить клас ResourceMonitor, який працює в окремому потоці (threading.Thread) для асинхронного збору даних.
* Збирає дані через psutil:
  + - get\_cpu\_usage(): Викликає psutil.cpu\_percent() для завантаження CPU.
    - get\_ram\_usage(): Викликає psutil.virtual\_memory() для використання RAM.
    - get\_network\_stats(): Викликає psutil.net\_io\_counters() для мережевих даних.
* Для SMART-аналізу накопичувачів використовує smartctl.exe через subprocess:
  + - Метод get\_disk\_info() викликає smartctl.exe --all /dev/sdX (або еквівалент для Windows, наприклад, smartctl.exe -a /dev/sda), отримує вивід і парсить дані (стан здоров’я, температура, знос).
    - Приклад виклику:

import subprocess

def get\_disk\_info(self):

try:

result = subprocess.run(

["smartctl.exe", "-a", "/dev/sda"],

capture\_output=True,

text=True

)

output = result.stdout

# Парсинг виводу для отримання температури, стану здоров’я тощо

health\_status = self.parse\_health\_status(output)

temperature = self.parse\_temperature(output)

return {"health": health\_status, "temperature": temperature}

except Exception as e:

return {"health": "Unknown", "temperature": "Unknown"}

Передає зібрані дані в SystemMonitorGUI через callback.

Взаємодія:

* Використовує psutil і subprocess для збору даних.
* Викликає utilities.py для логування через setup\_logging().

- Читає параметри з config.py, наприклад, UPDATE\_INTERVAL.

3.4.4 Модуль **utilities.py**

Роль: Допоміжні функції для логування, експорту та управління процесами.

Реалізація:

* Функція setup\_logging() налаштовує логування у файл system\_monitor.log за допомогою модуля logging.
* Функція export\_data() зберігає зібрані дані у текстовий файл (наприклад, report.txt).
* Функція kill\_process(pid) завершує процес через psutil:

import psutil

def kill\_process(pid):

try:

process = psutil.Process(pid)

process.terminate()

return True

except psutil.NoSuchProcess:

return False

Функція create\_plot() створює графіки через matplotlib.

Взаємодія:

* Використовується gui.py і monitor.py.
* Читає параметри з config.py, наприклад, AUTO\_EXPORT\_INTERVAL.

3.4.5 Модуль **config.py**

Роль: Зберігання конфігураційних параметрів.

Реалізація:

* Містить константи, такі як:
  + - UPDATE\_INTERVAL = 3 (інтервал оновлення даних у секундах).
    - CPU\_THRESHOLD = 90 (поріг для сповіщень про CPU, %).
    - MAX\_HISTORY = 60 (максимальна кількість точок у графіках).
    - SMARTCTL\_PATH = "C:/Program Files/smartmontools/bin/smartctl.exe" (шлях до smartctl.exe).
* Використовується всіма модулями для налаштування поведінки.

Взаємодія: Надає параметри для всіх модулів.

**3.5.1 Діаграма класів**

Діаграма класів (Рис. 3.1) відображає основні класи програми для моніторингу системних ресурсів, їхні атрибути, методи та зв’язки між ними. Вона ілюструє архітектуру програми на рівні класів, показуючи, як організовано взаємодію між графічним інтерфейсом і модулем моніторингу. Діаграма створена з використанням нотації UML і включає два основні класи: SystemMonitorGUI і ResourceMonitor, які є ключовими компонентами системи.

Зображення, що містить текст, знімок екрана, Шрифт, квитанція

Вміст, створений ШІ, може бути неправильним.

Рис. 3.1 Діаграма класів

**Опис структури діаграми класів**

Діаграма класів представляє архітектуру програми на рівні об’єктно-орієнтованого дизайну, демонструючи два основні класи: SystemMonitorGUI і ResourceMonitor. Кожен клас має набір атрибутів і методів, які відповідають за певну функціональність системи. Між класами встановлено зв’язок типу "Callback", що забезпечує передачу даних від модуля моніторингу до графічного інтерфейсу. Нижче наведено детальний опис кожного класу, його елементів і зв’язків.

Клас **SystemMonitorGUI**

Клас **SystemMonitorGUI** відповідає за створення та управління графічним інтерфейсом користувача (GUI). Він реалізує відображення даних про системні ресурси, графіки, сповіщення та взаємодію з користувачем, що відповідає вимогам про інтуїтивно зрозумілий інтерфейс і відображення даних.

Атрибути класу **SystemMonitorGUI**:

* *- root: Tk:* Об’єкт типу Tk із бібліотеки Tkinter, який представляє головне вікно програми.  
  Роль: Забезпечує базову структуру для всіх елементів GUI, таких як вкладки та кнопки. Використовується для ініціалізації графічного інтерфейсу та запуску головного циклу через root.mainloop().
* *- notebook: ttk.Notebook:* Віджет із бібліотеки Tkinter, який створює вкладки для відображення даних про CPU, RAM, GPU, накопичувачі, мережу та системну інформацію.  
  Роль: Дозволяє організувати інтерфейс у вигляді вкладок, що робить його зручним для користувача.
* *- cpu\_frame: ttk.Frame:* Контейнер для вкладки CPU, який містить графік і текстові поля для відображення даних.  
  Роль: Використовується для компонування елементів на вкладці CPU, таких як графіки та мітки.
* *- cpu\_plot: FigureCanvasTkAgg:* Об’єкт із бібліотеки matplotlib, який вбудовує графік у вкладку Tkinter.  
  Роль: Відображає графік завантаження CPU у реальному часі.
* *- history: dict:* Словник для зберігання історії даних (наприклад, завантаження CPU, RAM) для побудови графіків.  
  Роль: Зберігає обмежену кількість точок (визначається параметром MAX\_HISTORY із config.py), щоб уникнути перевантаження пам’яті, що відповідає вимозі про низьке споживання ресурсів

Методи класу **SystemMonitorGUI**:

* *+ setup\_gui():* Метод для ініціалізації графічного інтерфейсу.  
  Реалізація: Створює вкладки через ttk.Notebook, додає графіки через matplotlib (за допомогою функції create\_plot() із utilities.py) і налаштовує кнопки для експорту даних та завершення процесів.  
  Роль: Забезпечує початкову підготовку інтерфейсу перед запуском моніторингу.
* *+ update\_gui(data):* Метод для оновлення інтерфейсу на основі отриманих даних.  
  Реалізація: Приймає словник data із даними від ResourceMonitor (наприклад, {"cpu": 75, "ram": 60, "disk": {...}}), оновлює графіки та текстові поля, а також перевіряє порогові значення (наприклад, CPU\_THRESHOLD) для виклику сповіщень.  
  Роль: Забезпечує відображення даних у реальному часі.
* *+ show\_alert(message):* Метод для виведення сповіщень користувачу.  
  Реалізація: Використовує messagebox.showwarning() із Tkinter для відображення сповіщень (наприклад, "Високе завантаження CPU: 92%").  
  Роль: Реалізує функціональну вимогу про сповіщення при перевищенні порогових значень.

Клас **ResourceMonitor**

Клас **ResourceMonitor** відповідає за збір даних про системні ресурси в реальному часі. Він працює в окремому потоці, щоб не блокувати графічний інтерфейс, і передає зібрані дані в SystemMonitorGUI через механізм callback. Клас включає методи для моніторингу CPU, RAM, накопичувачів (з використанням smartctl.exe) тощо.

Атрибути класу **ResourceMonitor**:

* - *callback:* function: Функція зворотного виклику, передана з SystemMonitorGUI.  
  Роль: Використовується для передачі зібраних даних (наприклад, self.callback(data)), щоб оновити графічний інтерфейс. Це забезпечує зв’язок між класами.
* *- cpu\_usage\_history: list:* Список для зберігання історії завантаження CPU.  
  Роль: Зберігає значення завантаження CPU для побудови графіків у SystemMonitorGUI. Довжина списку обмежена параметром MAX\_HISTORY із config.py (наприклад, 60 точок), що допомагає економити пам’ять.
* *- ram\_usage\_history: list:* Список для зберігання історії використання RAM.  
  Роль: Аналогічно до cpu\_usage\_history, зберігає дані для графіків використання оперативної пам’яті.
* *- running: bool:* Логічна змінна, яка вказує, чи активний процес моніторингу.  
  Роль: Використовується для управління циклом моніторингу в методі monitor\_resources(). Якщо running = False, цикл зупиняється, що дозволяє коректно завершити роботу програми.

Методи класу ResourceMonitor:

* *+ start\_monitoring():* Метод для запуску моніторингу.  
  Реалізація: Встановлює running = True і запускає цикл моніторингу в окремому потоці через threading.Thread, викликаючи метод monitor\_resources().  
  Роль: Забезпечує асинхронний моніторинг, що відповідає вимозі про низьке споживання ресурсів і стабільність інтерфейсу.
* *+ stop\_monitoring():* Метод для зупинки моніторингу.  
  Реалізація: Встановлює running = False, що зупиняє цикл у monitor\_resources().  
  Роль: Дозволяє коректно завершити роботу програми при закритті.
* *+ get\_cpu\_usage():* Метод для отримання завантаження CPU.  
  Реалізація: Викликає psutil.cpu\_percent() для збору даних про завантаження CPU у відсотках.  
  Роль: Реалізує функціональну вимогу про моніторинг CPU.
* *+ get\_ram\_usage():* Метод для отримання використання RAM.  
  Реалізація: Викликає psutil.virtual\_memory() для збору даних про використання оперативної пам’яті.  
  Роль: Реалізує функціональну вимогу про моніторинг RAM.
* *+ get\_disk\_info():* Метод для отримання SMART-даних накопичувачів.  
  Реалізація: Викликає smartctl.exe через subprocess.run() із параметром --all для отримання повних SMART-даних (стан здоров’я, температура, знос). Парсить вивід для витягування потрібних параметрів, таких як температура та статус ("PASSED"/"FAILED").

Роль: Реалізує функціональну вимогу про SMART-аналіз накопичувачів.

3. Зв’язок між класами

Між класами **ResourceMonitor** і **SystemMonitorGUI** встановлено зв’язок типу Callback, який позначений стрілкою на діаграмі класів. Цей зв’язок означає, що ResourceMonitor передає зібрані дані в SystemMonitorGUI через функцію зворотного виклику (callback).

* Реалізація зв’язку:  
  Під час ініціалізації ResourceMonitor отримує метод update\_gui() із SystemMonitorGUI як параметр callback (наприклад, monitor = ResourceMonitor(gui.update\_gui) у main.py). У циклі моніторингу ResourceMonitor викликає self.callback(data), передаючи словник із даними (наприклад, {"cpu": 75, "ram": 60, "disk": {...}}), що дозволяє SystemMonitorGUI оновити інтерфейс.
* Роль зв’язку:  
  Забезпечує односторонню передачу даних від ResourceMonitor до SystemMonitorGUI, що відповідає принципу слабкого зв’язку (loose coupling) між компонентами. Це дозволяє ізолювати логіку моніторингу від логіки відображення, що полегшує тестування та модифікацію програми.

Відповідність вимогам

Діаграма класів відображає архітектуру, яка забезпечує виконання всіх ключових вимог програми:

* Функціональні вимоги:
  + Моніторинг CPU, RAM, GPU, мережі, накопичувачів реалізовано через методи get\_cpu\_usage(), get\_ram\_usage(), get\_disk\_info() у ResourceMonitor.
  + Сповіщення реалізовано через метод show\_alert() у SystemMonitorGUI.
  + Експорт даних підтримується через взаємодію з utilities.py, хоча це не відображено на діаграмі класів.
* Нефункціональні вимоги:
  + Низьке споживання ресурсів (<1% CPU, <50 МБ RAM) забезпечується завдяки асинхронному моніторингу (start\_monitoring() у ResourceMonitor) та обмеженню історії даних (history у SystemMonitorGUI).
  + Інтуїтивно зрозумілий інтерфейс реалізовано через використання вкладок (notebook) і графіків (cpu\_plot).

**3.5.2 Діаграма послідовності**

Діаграма послідовності (Рис. 3.2) ілюструє взаємодію між компонентами програми для моніторингу системних ресурсів під час оновлення даних у реальному часі. Вона створена з використанням нотації UML і відображає послідовність дій між основними компонентами: точкою входу програми, графічним інтерфейсом, модулем моніторингу та зовнішньою утилітою для SMART-аналізу. Діаграма показує, як програма ініціалізується, запускає моніторинг і періодично оновлює дані, включаючи SMART-інформацію про накопичувачі.

Зображення, що містить текст, знімок екрана, число, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рис. 3.2. Діаграма послідовності для оновлення даних у реальному часі

Опис учасників діаграми

Діаграма послідовності включає чотирьох учасників, кожен із яких відіграє певну роль у процесі оновлення даних:

* Точка входу програми: Представляє модуль main.py, який є координатором програми. Відповідає за ініціалізацію всіх компонентів і запуск основного циклу роботи.
* Графічний інтерфейс: Представляє клас **SystemMonitorGUI** із модуля gui.py. Відповідає за відображення даних, оновлення графіків і сповіщень, а також взаємодію з користувачем.
* Модуль моніторингу: Представляє клас **ResourceMonitor** із модуля monitor.py. Виконує збір даних про системні ресурси, включаючи SMART-аналіз накопичувачів, і передає їх для відображення.
* Зовнішня утиліта SMART-аналізу: Представляє утиліту smartctl.exe, яка використовується для отримання низькорівневих даних про накопичувачі, таких як стан здоров’я та температура.

Опис послідовності дій

Діаграма послідовності (Рис. 3.2) описує процес оновлення даних у реальному часі, який складається з кількох етапів: ініціалізації, запуску моніторингу та періодичного збору й відображення даних.

* Ініціалізація програми:  
  Процес починається з того, що точка входу програми ініціалізує графічний інтерфейс. Це включає створення головного вікна, вкладок і графіків, які необхідні для відображення даних. Після цього точка входу ініціалізує модуль моніторингу, створюючи екземпляр відповідного класу та передаючи йому функцію для передачі даних у графічний інтерфейс. Цей етап забезпечує підготовку всіх компонентів до роботи.
* Запуск моніторингу:  
  Графічний інтерфейс дає команду модулю моніторингу розпочати збір даних. Це запускає асинхронний процес, який працює в окремому потоці, щоб не блокувати відображення даних і взаємодію з користувачем. Такий підхід відповідає вимозі про стабільність і низьке споживання ресурсів.
* Періодичний збір і оновлення даних:  
  Після запуску моніторингу програма переходить у цикл, який повторюється з певним інтервалом (наприклад, кожні 3 секунди, як визначено в налаштуваннях). У межах цього циклу відбуваються такі дії:
  + Модуль моніторингу збирає дані про завантаження центрального процесора (CPU), обсяг використаної оперативної пам’яті (RAM) та інші параметри системи за допомогою відповідних бібліотек.
  + Модуль моніторингу звертається до зовнішньої утиліти SMART-аналізу, щоб отримати дані про накопичувачі, такі як стан здоров’я ("PASSED"/"FAILED") і температура. Утиліта повертає ці дані модулю моніторингу для подальшої обробки.
  + Після збору всіх даних модуль моніторингу передає їх графічному інтерфейсу через механізм зворотного виклику. Дані включають інформацію про CPU, RAM, накопичувачі тощо.
  + Графічний інтерфейс обробляє отримані дані, оновлюючи вкладки, текстові поля та графіки в реальному часі. Наприклад, графік завантаження CPU змінюється відповідно до нових значень, а вкладка накопичувачів відображає оновлені SMART-дані.

Цикл повторюється, доки програма не буде зупинена, забезпечуючи безперервне оновлення даних для користувача.

**3.9 Інтерфейс програми**

Інтерфейс програми для моніторингу системних ресурсів розроблений з використанням бібліотеки Tkinter і має інтуїтивно зрозумілу структуру, що відповідає вимозі про зручність використання. Основна мета інтерфейсу — забезпечити користувачу зручний доступ до даних про системні ресурси в реальному часі, включаючи графіки, текстову інформацію, SMART-дані накопичувачів і сповіщення про критичні стани. Інтерфейс складається з головного вікна, яке містить вкладки для різних категорій ресурсів, а також кнопки для додаткових функцій, таких як експорт даних і завершення процесів.

Опис елементів інтерфейсу

Головне вікно програми є основним робочим простором користувача. У верхній частині вікна розташована панель із вкладками, створена за допомогою віджета ttk.Notebook, що дозволяє перемикатися між різними категоріями даних: CPU, RAM, GPU, накопичувачі, мережа та системна інформація. Над вкладками розміщені кнопки для експорту даних у текстовий файл і завершення вибраних процесів, що відповідає функціональним вимогам. Кожна вкладка містить специфічні елементи для відображення даних, які оновлюються в реальному часі з інтервалом, визначеним у конфігурації (наприклад, кожні 3 секунди).

На Рис. 3.3 зображено головне вікно програми, де видно панель із вкладками та кнопки керування у верхній частині. Вкладка CPU відкрита за замовчуванням і показує основні елементи інтерфейсу.

Зображення, що містить текст, знімок екрана, число, ряд

Вміст, створений ШІ, може бути неправильним.

Рис. 3.3. Головне вікно програми для моніторингу системних ресурсів

**Вкладка CPU**

Вкладка CPU призначена для моніторингу центрального процесора і відображає його завантаження у відсотках. У верхній частині вкладки розташований графік, створений за допомогою бібліотеки matplotlib, який показує зміну завантаження CPU у реальному часі. Під графіком розміщене текстове поле, що відображає поточне значення завантаження (наприклад, "Завантаження CPU: 75%"). У нижній частині вкладки є список активних процесів із інформацією про їхнє споживання CPU, що дозволяє користувачу ідентифікувати найбільш ресурсоємні процеси. Користувач може вибрати процес із списку та завершити його за допомогою кнопки "Завершити процес", розташованої у верхній частині вікна.

На Рис. 3.4 зображено вкладку CPU із графіком завантаження, текстовим полем і списком процесів, що відповідає вимогам про моніторинг CPU і управління процесами.

Зображення, що містить текст, знімок екрана, Графік, схема

Вміст, створений ШІ, може бути неправильним.

Рис. 3.4. Вкладка моніторингу CPU із графіком

**Вкладка RAM**

Вкладка RAM відображає дані про використання оперативної пам’яті. Аналогічно до вкладки CPU, вона містить графік, який показує зміну обсягу використаної пам’яті у відсотках або мегабайтах, і текстове поле з поточними значеннями (наприклад, "Використання RAM: 6.2 ГБ з 8 ГБ"). Додатково відображається інформація про доступну пам’ять і обсяг, зайнятий кешем, що допомагає користувачу оцінити загальний стан пам’яті.

Зображення, що містить текст, знімок екрана, монітор, число

Вміст, створений ШІ, може бути неправильним.

Рис. 3.5. Вкладка моніторингу RAM

**Вкладка накопичувачів**

Вкладка накопичувачів надає інформацію про диски, включаючи їхній обсяг, використання та SMART-дані. Для кожного накопичувача відображається текстова інформація, така як "Використання: 120 ГБ з 500 ГБ", а також SMART-параметри, отримані за допомогою утиліти smartctl.exe: стан здоров’я ("PASSED"/"FAILED"), температура (наприклад, "Температура: 35°C") і кількість відпрацьованих годин. Ця вкладка дозволяє користувачу швидко оцінити стан накопичувачів і виявити потенційні проблеми, що відповідає вимозі про SMART-аналіз.

На Рис. 3.5 зображено вкладку накопичувачів із відображенням SMART-даних, таких як стан здоров’я і температура, що дозволяє користувачу контролювати стан дисків.

Зображення, що містить текст, знімок екрана, ряд, Графік

Вміст, створений ШІ, може бути неправильним.

Рис. 3.6. Вкладка моніторингу накопичувачів із SMART-даними

**Вкладка GPU**

Вкладка GPU: Відображає завантаження графічного процесора (якщо підтримується), температуру та список процесів, що використовують GPU.

Зображення, що містить текст, знімок екрана, Графік, схема

Вміст, створений ШІ, може бути неправильним.

Рис. 3.7. Вкладка моніторингу завантаження графічного процесора

Зображення, що містить текст, знімок екрана, число, Графік

Вміст, створений ШІ, може бути неправильним.

Рис. 3.8. Вкладка моніторингу пам’яті графічного процесора

Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.

Рис. 3.9. Вкладка моніторингу температури графічного процесора

**Вкладка мережі**

Вкладка мережі: Показує швидкість передачі даних (вхідний і вихідний трафік) у реальному часі через графік, а також загальний обсяг переданих даних.

**Зображення, що містить текст, знімок екрана, число, ряд

Вміст, створений ШІ, може бути неправильним.**

Рис. 3.10. Вкладка моніторингу марежі

**Вкладка системної інформації**

Вкладка системної інформації: Містить статичні дані, такі як версія операційної системи, модель процесора, обсяг встановленої пам’яті та інформація про диски.

Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

Рис. 3.11. Вкладка системної інформації

**Сповіщення**

Програма включає функцію сповіщень, яка активується, коли певні параметри перевищують порогові значення (наприклад, завантаження CPU > 90%). Сповіщення відображаються у вигляді діалогового вікна, що з’являється поверх головного вікна, із повідомленням, наприклад, "Високе завантаження CPU: 92%". Це відповідає вимозі про сповіщення користувача.

Зображення, що містить текст, знімок екрана, Шрифт, монітор

Вміст, створений ШІ, може бути неправильним.

Рис. 3.12. Сповіщення користувача