## Високорівневий опис реалізації майбутнього програмного рішення згідно розроблених вимог

1. Архітектура програми:

- Програма "MoniSy" буде реалізована за допомогою поєднання мов програмування Python та C++ для забезпечення оптимальної продуктивності та швидкодії.

- В основі архітектури буде модульна структура, що дозволить розділити різні аспекти моніторингу та аналізу на окремі компоненти.

2. Моніторинг та аналіз ресурсів:

- Для моніторингу використання CPU, пам'яті, дискового простору та мережевого трафіку програма використовуватиме системні виклики та бібліотеки, доступні у мові програмування C++ та Python, для збору статистики в реальному часі.

- Дані про використання ресурсів будуть зберігатися відповідно до принципів оптимальної організації даних, щоб забезпечити швидкий доступ до них для аналізу та відображення.

3. Інтерфейс користувача:

- Графічний інтерфейс програми буде розроблений з використанням бібліотеки Tkinter/ PyQt/ Kivy/ wxPython у мові програмування Python для забезпечення зручності та доступності для користувача. Кінцевий варіант GUI бібліотеки буде обраний після тестування можливостей та продуктивності, так як потрібна велика швидкість реагування для відображення графіків у реальному часі.

- Інтерфейс буде налаштовуватися, дозволяючи користувачеві вибирати та налаштовувати показники, які він бажає відслідковувати, а також встановлювати порогові значення для сповіщень.

4. Системні дані та журнали:

- Для збору системних подій та журналів програма буде використовувати системні API Windows, щоб отримати доступ до відповідних журналів та подій.

- Інформація про систему, така як версія ОС, тип процесора, обсяг пам'яті та інші, буде отримана за допомогою відповідних системних викликів та бібліотек.

5. Сповіщення та аналіз даних:

- Система сповіщень буде налаштована з використанням механізмів оповіщення ОС Windows, щоб надавати користувачам повідомлення про перевищення порогових значень.

- Для аналізу даних та виявлення проблем програма використовуватиме алгоритми машинного навчання та статистичні методи для виявлення аномальної активності та рекомендацій щодо оптимізації використання ресурсів.

6. Підтримка та поширення:

- Програма буде підтримувати операційні системи Windows починаючи з версії Windows 7 для обох архітектур, 32-х та 64-бітних.

- Для поширення програма буде доступна для завантаження з веб-сайту розробника, а також можливе розміщення у відповідних інтернет-магазинах.

Цей підхід дозволить створити комплексне та ефективне програмне рішення "MoniSy", яке задовольнить потреби користувачів у моніторингу та аналізі їхніх систем Windows.

## Структура програмного рішення; архітектуру класів, їх призначення і взаємодію, опис функцій

1. \*\*Клас `SystemMonitor`\*\*:

- Призначення: Головний клас, який управляє всіма аспектами моніторингу системи.

- Методи:

- `start\_monitoring()`: Починає моніторинг системи.

- `stop\_monitoring()`: Зупиняє моніторинг системи.

- `update\_system\_info()`: Оновлює інформацію про систему.

- `set\_thresholds()`: Встановлює порогові значення для сповіщень.

2. \*\*Клас `CPUUsageMonitor`\*\*:

- Призначення: Моніторинг використання CPU.

- Методи:

- `get\_cpu\_usage()`: Отримує інформацію про використання CPU.

- `notify\_if\_threshold\_exceeded()`: Повідомляє, якщо порогове значення використання CPU перевищено.

3. \*\*Клас `MemoryUsageMonitor`\*\*:

- Призначення: Моніторинг використання пам'яті.

- Методи:

- `get\_memory\_usage()`: Отримує інформацію про використання пам'яті.

- `notify\_if\_threshold\_exceeded()`: Повідомляє, якщо порогове значення використання пам'яті перевищено.

4. \*\*Клас `DiskSpaceMonitor`\*\*:

- Призначення: Моніторинг використання дискового простору.

- Методи:

- `get\_disk\_space()`: Отримує інформацію про використання дискового простору.

- `notify\_if\_threshold\_exceeded()`: Повідомляє, якщо порогове значення використання дискового простору перевищено.

5. \*\*Клас `NetworkTrafficMonitor`\*\*:

- Призначення: Моніторинг мережевого трафіку.

- Методи:

- `get\_network\_traffic()`: Отримує інформацію про мережевий трафік.

- `notify\_if\_threshold\_exceeded()`: Повідомляє, якщо порогове значення мережевого трафіку перевищено.

6. \*\*Клас `SystemEventMonitor`\*\*:

- Призначення: Моніторинг системних подій та журналів.

- Методи:

- `get\_system\_events()`: Отримує інформацію про системні події та журнали.

- `notify\_if\_critical\_event\_detected()`: Повідомляє, якщо виявлено критичну подію або помилку.

7. \*\*Клас `TemperatureAndFanMonitor`\*\*:

- Призначення: Моніторинг температури та вентиляції.

- Методи:

- `get\_temperature\_and\_fan\_info()`: Отримує інформацію про температуру та вентиляцію.

- `notify\_if\_threshold\_exceeded()`: Повідомляє, якщо порогове значення температури або швидкості вентилятора перевищено.

8. \*\*Клас `SystemInformation`\*\*:

- Призначення: Збір та представлення системної інформації.

- Методи:

- `get\_basic\_system\_info()`: Отримує базову інформацію про систему (операційну систему, тип процесора, обсяг пам'яті тощо).

- `get\_installed\_devices\_info()`: Отримує інформацію про встановлені пристрої та їх параметри.

- `get\_network\_info()`: Отримує інформацію про мережеві параметри та з'єднання.

9. \*\*Клас `NotificationManager`\*\*:

- Призначення: Керування сповіщеннями.

- Методи:

- `send\_notification()`: Відправляє сповіщення користувачеві в разі перевищення порогових значень.

Це лише загальна структура програми, і реальна реалізація може варіюватися в залежності від конкретних вимог та архітектурних рішень.

## Перша спроба реалізації програми

Реалізація програми знаходиться у файлах main.py та functions.py

Для справної роботи потрібно встановити залежності за допомогою команди

‘pip install -r re quirements.txt’