МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«_____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА МОНІТОРИНГУ СТАБІЛЬНОСТІ ІНТЕРНЕТ-КОНФЕРЕНЦІЇ В УМОВАХ БАГАТОЗАДАЧНОСТІ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21910202

Виконала студентка 4-го курсу, групи 402
_____***О. О. Бухтіярова***
«21» червня 2023 р.

Керівник: д-р техн. наук, професор
_____***І. М. Журавська***
«21» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Бухтіяровій Олені Олегівні.

1. Тема кваліфікаційної роботи «Система моніторингу стабільності інтернет-конференції в умовах багатозадачності».

Керівник роботи Журавська Ірина Миколаївна, д-р техн. наук, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20_ р.

3. Вхідні (початкові) дані до роботи: інтернет-конференція з різним програмним забезпеченням. .

Очікуваний результат: система для моніторингу стабільності інтернет-конференції в умовах багатозадачності.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз предметної сфери;

– методи та інформаційні технології для вирішення поставленої задачі;

– моделювання та проєктування застосунку;

– програмна реалізація та тестування.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Вимоги до роботи з комп'ютерним обладнанням, проведення навчання з питань охорони праці за допомогою інтернет-конференцій та дії в надзвичайних ситуацій».

7. Консультанти розділів робот

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Викладач кафедри екології Медичного інституту ЧНУ імені Петра Могили Боженко А. Л.	

Керівник роботи __д-р техн. наук, проф. Журавська І. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Бухтіярова О. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Система моніторингу стабільності інтернет-конференції в умовах багатозадачності

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	29.10.2022	29.10.2022	виконано
2	Отримання завдання на виконання БКР	23.11.2022	23.11.2022	виконано
3	Складання календарного плану роботи на весь період виконання БКР	8.12.2022	8.12.2022	виконано
4	Отримання завдання на переддипломну практику	29.04.2023	29.04.2023	виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	виконано
7	Виконання БКР: аналіз предметної сфери, дослідження ресурсоспоживання відеоконференцій, аналіз застосунків-аналогів, огляд існуючих технологій, проєктування моделі застосунку, розробка ПЗ, тестування готового ПЗ	15.05.2023	19.06.2023	виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробила студентка Бухтіярова О. О. _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи д-р техн. наук, проф. Журавська І. М. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

« 08 » 12 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студенки групи 402 ЧНУ ім. Петра Могили

Бухтіярової Олени Олегівни

Тема: «Система моніторингу стабільності інтернет-конференції в умовах багатозадачності»

На сьогоднішній день більшість людей перейшли на дистанційну роботу та навчання, тому збільшилось використання засобів для віддалено спілкування, а саме – відеоконференцій. Часто користувачі стикаються з такими проблемами, як старі моделі персональних комп’ютерів та нестабільне інтернет-з’єднання. Тому моніторинг стабільності роботи інтернет-конференцій в умовах багатозадачності комп'ютера є актуальним, і полягає в тому, що за допомогою такого аналізу можна оцінити продуктивність роботи комп'ютера та покращити стабільність роботи з іншими програмами й сайтами.

Об'єкт роботи – ризики аварійного завершення інтернет-конференцій через надмірне споживання ресурсів ПК, а також трафіку передачі даних.

Предмет роботи – параметри ОС, що реєструють перенавантаження комп'ютерної системи, а також програмні інструменти для розроблення застосунку, призначеного для попередження користувача про загрозу аварійного завершення інтернет-конференції.

Метою бакалаврської кваліфікаційної роботи є дослідження та аналіз стабільності інтернет-конференції в умовах багатозадачності за допомогою системи моніторингу на мові програмування C# та платформі Windows Forms.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та двох додатків.

У першому розділі розглядається аналіз предметної сфери, огляд та аналіз застосунків-аналогів та постановка задачі.

У другому розділі визначається вибір програмних засобів реалізації для створення застосунку.

У третьому розділі розроблено моделі за допомогою UML та описано структуру застосунку-моніторингу.

У четвертому розділі представлено графічний інтерфейс та функціонал застосунку, результати роботи та тестування ПЗ при різних умовах.

В результаті розроблено систему для моніторингу стабільності інтернет-конференції в умовах багатозадачності.

Бакалаврська кваліфікаційна робота містить 82 сторінки (без додатків), 54 рисунків, 12 таблиць, 25 використаних джерел та 2 додатка.

Ключові слова: відеоконференції, моніторинг, C#, Windows Forms, .NET.

ABSTRACT

of the Bachelor's Thesis

"A system for monitoring the stability of an Internet conference in a multitasking environment"

Student of group 402: Bukhtiiarova Olena

This thesis is devoted to the nowadays, most people have switched to remote work and study, so video conferencing has increased. Many people face problems such as old PC and laptop models and unstable Internet connections. Therefore, monitoring the stability of Internet conferencing in the context of computer multitasking is relevant, as it helps to assess computer performance and improve the strength of work with other programs and websites.

The object of the work is the risks of crashing Internet conferences due to excessive consumption of PC resources and data traffic.

The subject of the work is the OS parameters that register the overload of a computer system, as well as software tools for developing an application designed to warn the user about the threat of an emergency termination of an Internet conference.

The purpose of this work is the study and analyse the stability of an Internet conference in a multitasking environment using a monitoring system in the C# programming language and the Windows Forms platform.

The explanatory note consists of an introduction, four chapters, conclusions, a list of reference sources and two appendices.

The first section analyzes the field of the subject area, a review and analysis of analogue applications, and a task statement.

The second section is devoted to the analysis and selection of software tools for creating an application.

The third section describes the process of modelling an information system using UML and describes the structure of the monitoring application.

The fourth section describes the developed graphical interface and functionality of the application, the results of software operation and testing under various conditions.

The bachelor's thesis contains 82 pages (without appendices), 54 figures, 12 tables, 25 references and 2 appendices.

Key words: video conference, monitoring, C#, Windows Forms, .NET.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних
БКР – бакалаврська кваліфікаційна робота
ОЗП – оперативний запам’ятовуючий пристрій
ПЗ – програмне забезпечення
ПК – персональний комп’ютер
СКБД – система керування базами даних
ЦП – центральний процесор

AI – Artificial Intelligence
API – Application Programming Interface
CPU – Central Processing Unit
GUI – Graphical User Interface
IDE – Integrated Development Environment
UML – Unified Modeling Language

ВСТУП

Актуальність моніторингу стабільності роботи інтернет-конференцій в умовах багатозадачності комп'ютера полягає в тому, що такий аналіз дозволяє оцінити продуктивність роботи комп'ютера, визначити алгоритми роботи для зниження навантаження. Такий аналіз дає можливість покращити стабільність роботи інтернет-конференцій з іншими програмами й сайтами.

На сьогоднішній день це є дуже актуально, тому що багато людей користуються відеоконференціями у повсякденному житті, для роботи, для навчання, але не мають стабільного інтернет-з’єднання та використовують різні персональні комп’ютери (ПК) та ноутбуки, у тому числі доволі старі моделі.

Об'єкт роботи – ризики аварійного завершення інтернет-конференцій через надмірне споживання ресурсів ПК, а також трафіку передачі даних.

Предмет роботи – параметри ОС, що реєструють перенавантаження комп'ютерної системи, а також програмні інструменти для розроблення застосунку, призначеного для попередження користувача про загрозу аварійного завершення інтернет-конференції.

Метою кваліфікаційної бакалаврської роботи є дослідження та аналіз стабільності інтернет-конференції в умовах багатозадачності за допомогою системи моніторингу на мові програмування C# та платформі Windows Forms.

Для досягнення поставленої мети необхідно виконати наступні **завдання**:

– дослідити предметну сферу, провести аналіз літератури та публікацій;

– дослідити вплив інтернет конференцій на споживання ресурсів пристроїв: пам’яті, центрального процесора (ЦП), а також трафіку передачі даних та ін.;

– проаналізувати аналогічні системи для аналізу стабільності інтернет-конференції;

– визначити функціонал застосунку, визначити засоби для реалізації система моніторингу;

– розробити застосунок за допомогою обраних технологій;

– протестувати розроблене програмне забезпечення (ПЗ).

Практичне значення – для запобігання зависання персонального комп’ютера виконується інформування користувача о можливих аварійних ситуаціях та втрати даних.

Робота пройшла апробацію під час XI Міжнародної науково-практичної конференції «Free and Open Source Software (FOSS’2023)» (Харків, 14–16 лютого 2023 р.) та Всеукраїнської науково-практичної конференції «Інформаційні технології та інженерія» (Миколаїв, 07–10 лютого 2023 р.). За результатами конференцій опубліковані тези доповідей [1, 2].

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Дослідження використання відеоконференцій, як інструменту зв’язку

Інтернет-конференції – один з найпоширеніших видів зв’язку сьогодення, який об'єднує людей з усього світу в онлайн середовищі для обговорення спільних інтересів та проблем. Онлайн-конференції використовуються у різних сферах: освіта, дослідження, бізнес, політика, культура тощо.

Через пандемію COVID-19, воєнний стан та інші виклики сьогодення багато організацій, підприємств та навчальних закладів перейшли на нову структуру онлайн-роботи. Зустрічі відбуваються через системи відеоконференцій, які дозволяють сотням мільйонів користувачів спілкуватися онлайн за допомогою аудіо- та відеопотоків.

Відеоконференції, зазвичай, проводяться через спеціальну онлайн-платформу, яка дозволяє учасникам приєднатися до зустрічі з будь-якого місця, де є доступ до Інтернету. Платформа включає такі інструменти взаємодії, як відеозв’язок, чат, спільний доступ до файлів і документів, а також опитування.

Онлайн-зустрічі можуть відбуватися у різних форматах, включаючи вебінари, круглі столи та онлайн-курси, і можуть бути відкритими для всіх або закритими (лише за запрошеннями), а також існують великі міжнародні онлайн-конференції з сорока тисячами учасників з різних країн.

Онлайн-освіта все більше стає популярною, тому використання відеоконференцій – чудовий варіант для проведення навчального процесу дистанційно. Такі конференції дозволяють приймати участі у навчальному процесі незалежно від місця проживання та географічного розташування навчального закладу. Також кожен має багато своїх справ, а таке навчання допомагає заощадити час [3, 4].

Відеоконференції швидко стають важливим бізнес-інструментом. Статистика показує, що відеоконференції, які раніше були необов'язковим

нововведенням у діловому світі, швидко пережили стрімке зростання, особливо серед підприємств із віддаленим режимом роботи.

Багато компаній і підприємств, що працюють віддалено, використовують відеозустрічі для найрізноманітніших цілей. Це включає в себе ділові дзвінки один на один, групові зустрічі, всі зустрічі персоналу і навіть повні події та події, які транслюються в прямому ефірі для онлайн-супроводжуючих.

Перевага онлайн-конференцій полягає в тому, що вони заощаджують гроші та час на відвідування заходів, які раніше можна було відвідати лише офлайн. Вони також дають можливість людям з усього світу збиратися разом, співпрацювати та обмінюватися знаннями.

Інтернет-конференції мають також і недоліки. Під час проведення відеоконференцій можуть виникати технічні проблеми зі зв’язком, звуком, відео. Тому для отримання максимальної ефективності від інтернет-конференцій необхідно забезпечити високу якість технічного та інформаційного забезпечення [5].

Нині для тих, хто працює віддалено або навчається, доступно безліч відеоплатформ: Skype, Zoom, Microsoft Teams і Google Meet – це лише кілька варіантів. Ці застосунки надають основні функції спільної роботи з відеозв’язком.

Відповідно до інтересів покупців, топ-8 лідерів на ринку відеоконференцій у 2022 році виглядає наступним чином (рис. 1.1):

1) Zoom;
2) Microsoft Teams;
3) Webex Meetings;
4) TeamViewer;
5) Google Hangouts Meet;
6) GoToMeeting;
7) BlueJeans Meetings.

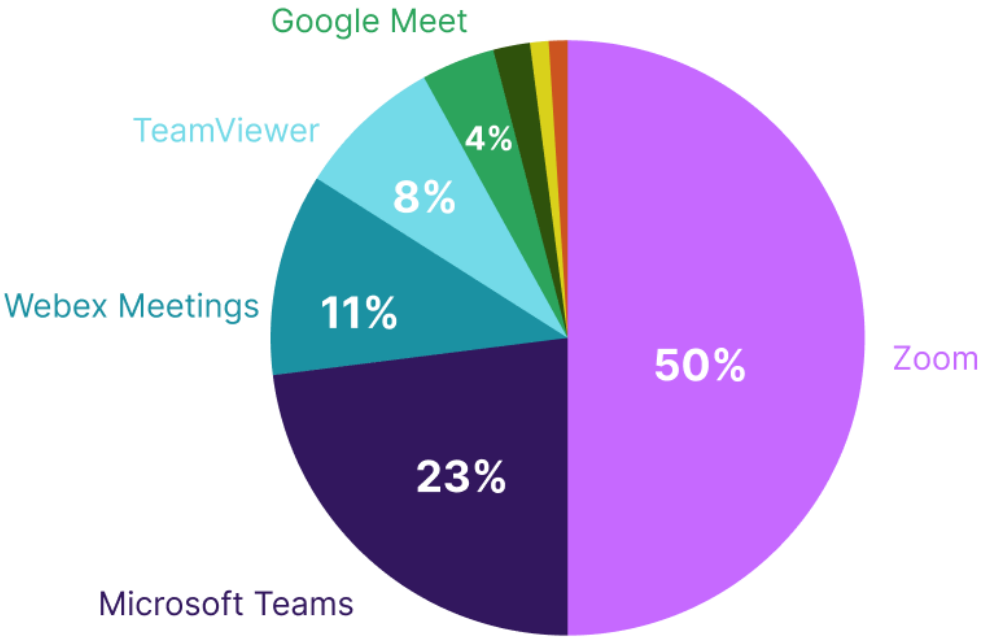


Рисунок 1.1 – Топ-8 лідерів на ринку відеоконференцій

Далі наведено статистику щодо використання відеоконференцій у повсякденному житті:

– відеоконференції покращили комунікацію для 99 % людей;

– 90 % користувачів відеодзвінків вважають, що їм легше донести свою думку, якщо їх бачать;

– 65 % усіх відеодзвінків здійснюються лише зі звуком;

– до 51 % людей вважають компанії, які використовують відеоконференції, більш інноваційними;

– до 76 % віддалених співробітників використовують відеоконференції для роботи [6, 7].

Глобальні виклики останніх роки змусили суспільство перейти до дистанційної освіти та роботи, зокрема до зв’язку через відеоконференції. Тому підтримка стабільної роботи інтернет-конференцій при будь-яких обставинах за допомогою аналізу та керування навантаженням пристроїв в умовах багатозадачності є актуальним завданням.

1.2 Аналіз ресурсоспоживання відеоконференцій

Для оцінки продуктивності інтернет-конференцій було досліджено ресурсоспоживання різних платформ для відеоконференцій.

Проведені експерименти з якості обслуговування за допомогою мережевих вимірювань на стороні клієнта та відправника на трьох популярних платформах, а саме: Google Meet, Microsoft Teams і Zoom. Хоча кожна платформа має свої переваги, але жоден застосунок не є ідеальним. Користувач може вибрати відповідну платформу в залежності від того, що саме важливіше для нього: аудіо, відео чи пропускна здатність мережі, процесор чи пам'ять. Для даного дослідження було взято комп’ютери з наступними параметрами (табл. 1.1).

Таблиця 1.1 – Параметри комп’ютерів для дослідження

Параметри	**Відправник**	**Отримувач**
ЦП	Intel i5-8265U	Intel i5-8250U
Оперативна пам'ять, Гбайт	16	16
Операційна система	Windows 10	Windows 10
Широкосмуговий інтернет	WLAN 802.11ac 150 Мбіт/с	WLAN 802.11ac 150 Мбіт/с
4G мобільний інтернет, Мбіт/с	11	10,5
Акумулятор, Вт·год	41	41
Браузер	Google Chrome	Google Chrome

В результаті дослідження було встановлено, що Microsoft Teams і Zoom поводяться однаково (включно зі споживанням ресурсів) як при використанні мобільного 4G-інтернету, так і кабельного широкосмугового. Однак Google Meet обмежує свою пропускну здатність при підключені через мобільний інтернет, що це знижує якість відеопотоку. А також він більше споживає пам'ять пристрою.

Натомість, Zoom і Microsoft Teams забезпечують найкращу можливу якість відео при будь-якому підключенні до мережі Інтернет.

Визначено, що швидкість передачі пакетів у Google Meet не така стабільна, як у двох інших сервісів, тому відбувається більша буферизацію/падіння кадрів, і як наслідок більше споживання пам'яті. Що стосується аудіо, то Google Meet покращує якість такого зв’язку, порівняно з відео.

Якщо аудіо для користувача важливіше, то Google Meet і Zoom є кращими варіантами при кабельному широкосмуговому зв'язку. Якщо ж важливіший відеозв’язок, Microsoft Teams і Zoom кращі варіанти. Користувачеві, який зацікавлений в економії трафіку, потрібно вибрати Google Meet, особливо при використанні мобільного інтернету.

Далі порівняємо споживання ресурсів на стороні відправника. Microsoft Teams має стабільно високе завантаження процесора порівняно з Zoom та Google Meet. Для Google Meet спостерігається значне збільшення обсягу пам'яті, що використовується, при увімкненій камері (табл. 1.2).

Таблиця 1.2 – Споживання ресурсів на стороні відправника при використанні кабельного широкосмугового зв'язку

Тип тесту	**Платформа**	**Завантаження ЦП, %**	**Споживання пам’яті, Мбайт**	**Споживання акумулятора, %**
Мікрофон OFF камера OFF	Google Meet	13,35	336,61	5,00
	MS Teams	26,58	294,68	8,00
	Zoom	15,91	355,23	6,00
Мікрофон ON камера OFF	Google Meet	22,68	388,71	6,00
	MS Teams	29,21	312,39	9,00
	Zoom	16,05	358,82	10,00
Мікрофон OFF камера ON	Google Meet	25,14	555,40	7,00
	MS Teams	30,89	360,77	10,00
	Zoom	20,54	370,17	9,00
Мікрофон ON камера ON	Google Meet	25,22	575,90	8,00
	MS Teams	31,74	372,20	10,00
	Zoom	22,88	382,85	10,00

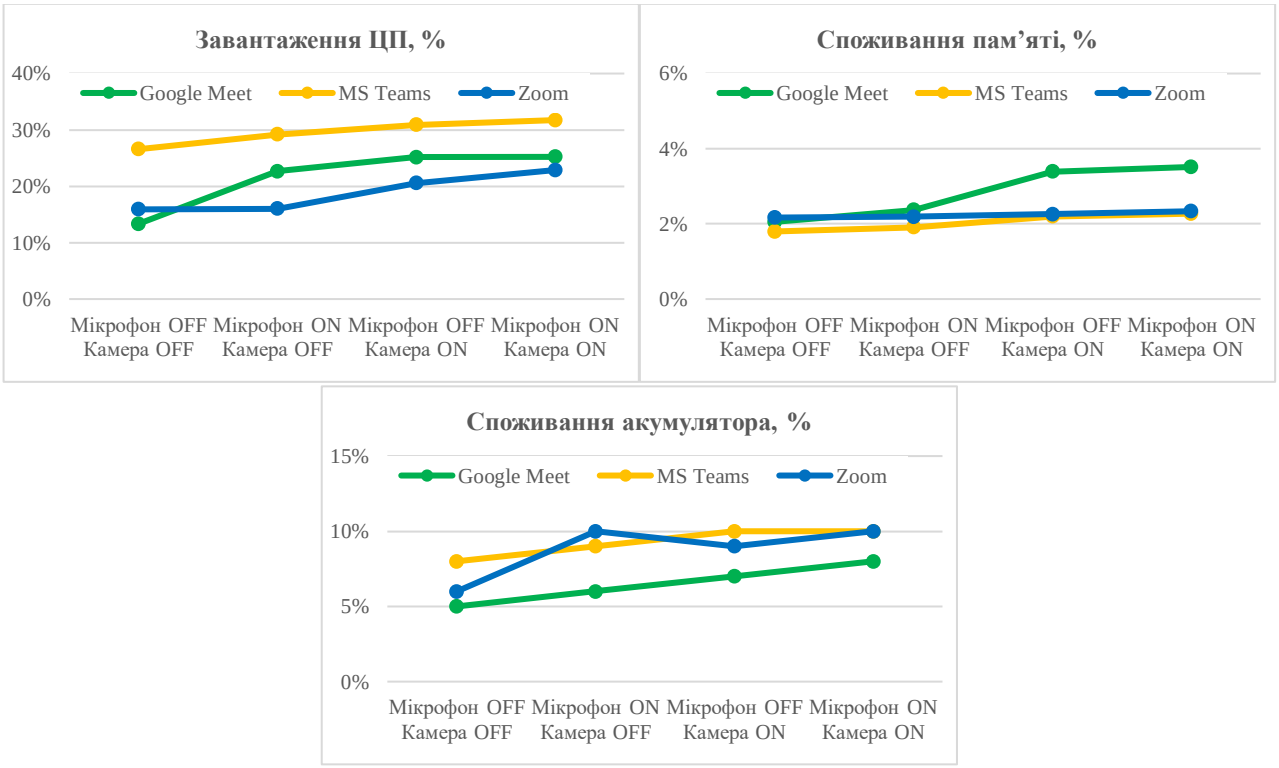


Рисунок 1.2 – Споживання ресурсів через кабельной широкосмуговой зв'язок

Наступним кроком проаналізуємо споживання ресурсів на стороні приймача для кабельного широкосмугового зв'язку. Microsoft Teams мінімально використовує ресурси процесора порівняно з Zoom та Google Meet. Після запуску Google Meet з увімкненою камерою спостерігається значне збільшення обсягу пам'яті, що використовується (табл. 1.3).

Таблиця 1.3 – Споживання ресурсів на стороні приймача при використанні кабельного широкосмугового зв'язку

Тип тесту	**Платформа**	**Завантаження ЦП, %**	**Споживання пам'яті, Мбайт**	**Споживання акумулятора, %**
Мікрофон OFF камера OFF	Google Meet	2,31	233,13	5,00
	MS Teams	4,64	209,33	7,00
	Zoom	9,93	232,85	6,00
Мікрофон ON камера OFF	Google Meet	6,07	269,09	6,00
	MS Teams	6,90	222,17	7,00
	Zoom	11,52	246,97	7,00

Кінець таблиці 1.3

Мікрофон OFF камера ON	Google Meet	12,99	489,22	7,00
	MS Teams	9,11	275,53	8,00
	Zoom	13,81	258,24	8,00
Мікрофон ON камера ON	Google Meet	13,51	514,17	7,00
	MS Teams	9,98	291,55	8,00
	Zoom	14,30	263,09	8,00

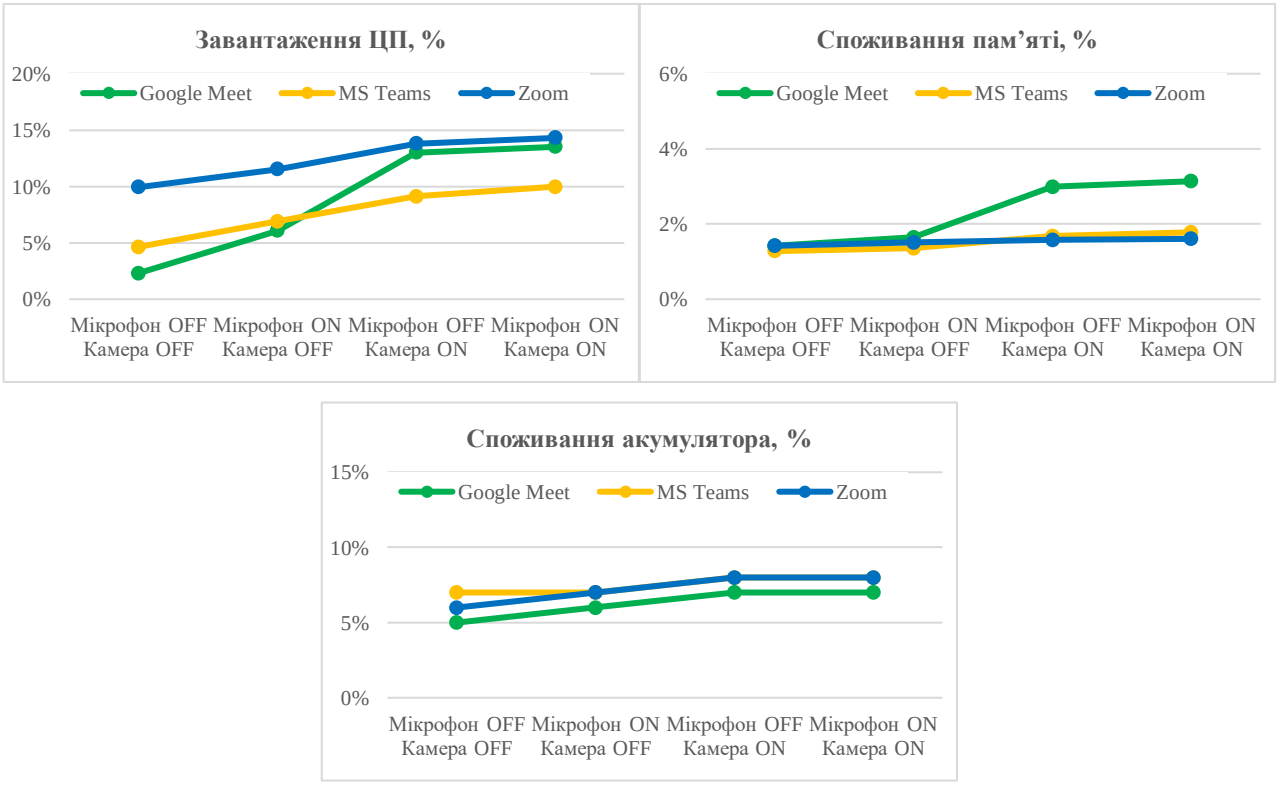


Рисунок 1.3 – Споживання ресурсів через кабельной широкосмуговой зв'язок

Також перевіримо споживання ресурсів на стороні відправника при підключені через мобільний інтернет Google Meet і Zoom мають схоже використання процесора, тоді як Microsoft Teams має порівняно більше використання процесора та має низьке споживання пам'яті. Google Meet значно збільшує споживання пам'яті, коли увімкнено камеру (табл. 1.4).

Таблиця 1.4 – Споживання ресурсів на стороні відправника через 4G

Тип тесту	Платформа	Завантаження ЦП, %	Споживання пам'яті, Мбайт	Споживання акумулятора, %
Мікрофон OFF камера OFF	Google Meet	12,46	336,69	7,00
	MS Teams	20,17	243,62	8,00
	Zoom	10,11	276,11	7,00
Мікрофон ON камера OFF	Google Meet	12,82	356,45	8,00
	MS Teams	26,99	267,86	9,00
	Zoom	12,86	296,28	8,00
Мікрофон OFF камера ON	Google Meet	19,18	455,40	7,00
	MS Teams	31,29	365,27	9,00
	Zoom	17,33	452,39	10,00
Мікрофон ON камера ON	Google Meet	19,69	517,89	9,00
	MS Teams	33,32	394,63	10,00
	Zoom	18,83	460,35	11,00

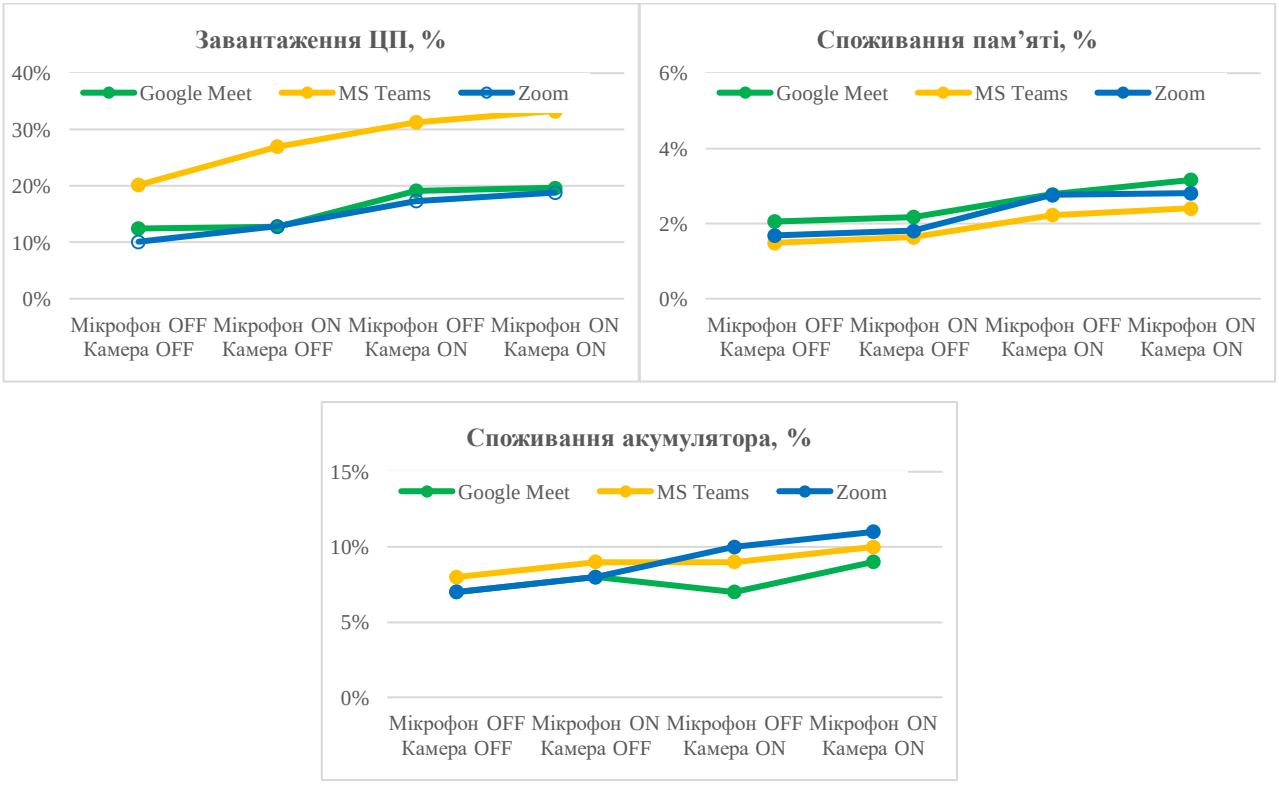


Рисунок 1.4 – Споживання ресурсів через мобільний інтернет 4G

Далі порівняємо споживання ресурсів на стороні приймача при підключені через мобільний інтернет. Усі застосунки використовують процесор приблизно однаково, причому Microsoft Teams трохи менше, ніж Google Meet та Zoom. Google Meet має порівняно більше споживання пам'яті. Споживання пам'яті Google Meet значно збільшується, коли камеру увімкнено (табл. 1.5).

Таблиця 1.5 – Споживання ресурсів на стороні приймача через 4G

Тип тесту	**Платформа**	**Завантаження ЦП, %**	**Споживання пам’яті, Мбайт**	**Споживання акумулятора, %**
Мікрофон OFF камера OFF	Google Meet	5,71	224,17	6,00
	MS Teams	4,92	211,21	7,00
	Zoom	6,50	222,84	6,00
Мікрофон ON камера OFF	Google Meet	6,23	228,78	6,00
	MS Teams	5,94	217,57	8,00
	Zoom	6,95	246,37	7,00
Мікрофон OFF камера ON	Google Meet	9,22	439,14	8,00
	MS Teams	7,78	291,25	8,00
	Zoom	7,45	248,44	8,00
Мікрофон ON камера ON	Google Meet	10,13	466,71	9,00
	MS Teams	8,13	293,86	9,00
	Zoom	10,31	261,96	9,00

Отже, в результаті цього дослідження було визначено, що Microsoft Teams використовує приблизно на 10 % більше корисного навантаження, ніж Google Meet та Zoom для кабельного широкосмугового зв'язку. Це дозволяє Microsoft Teams надавати відео з вищою роздільною здатністю, ніж Google Meet. Zoom може забезпечити кращу якість відео зі значно меншим корисним навантаженням. Google Meet зазбечує гіршу якість відео, але Google Meet забезпечує кращу якість звуку.

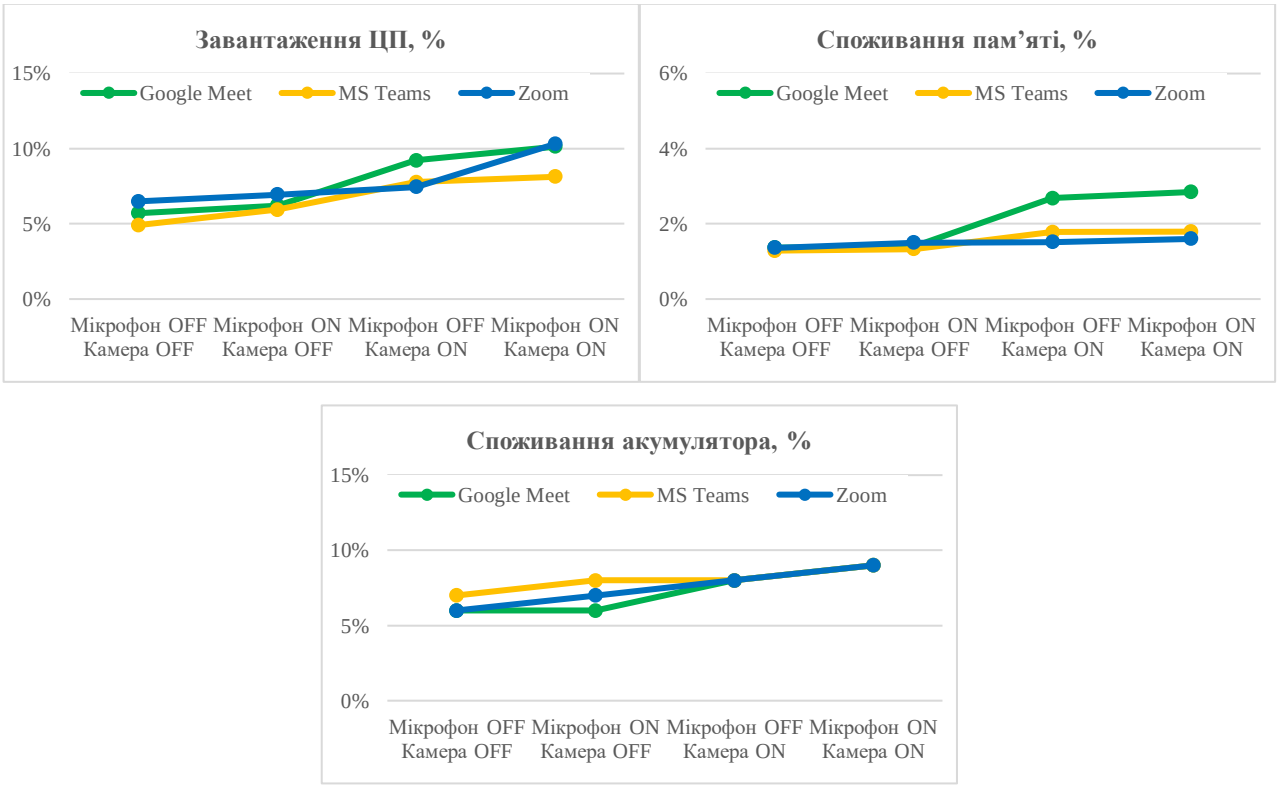


Рисунок 1.5 – Споживання ресурсів через мобільний інтернет 4G

Microsoft Teams і Zoom поводяться однаково (включаючи споживання ресурсів) у мобільному 4G-інтернеті, як і в кабельному широкосмуговому. Однак Google Meet обмежує свою пропускну здатність під час використання мобільного інтернету 4G, тим самим ще більше знижує якість відео [8, 9].

1.3 Огляд та аналіз застосунків-аналогів

Програмне забезпечення для моніторингу системи – це інструмент, який відстежує, реєструє й аналізує системні ресурси на всіх рівнях IT-середовища – від операційних систем і вбудованого ПЗ у нижній частині стека до системних застосунків і служб на середньому рівні та користувацького програмного забезпечення, що працює на вершині.

Розробники, адміністратори та звичайні користувачі повинні мати можливість отримати детальне уявлення про робочий стан пристроїв, відстежуючи певні показники продуктивності системи, такі як завантаженість процесора,

пам'ять і дисковий простір тощо. Це, своєю чергою, допоможе ефективно керувати системними ресурсами, підвищуючи загальну продуктивність.

Далі розглянемо декілька популярних програмних забезпечень для моніторингу системи.

Диспетчер задач – це менеджер запуску, системний монітор і диспетчер завдань. Він фіксує навантаження та відомості про пам'ять, мережеву активність і статистику, зареєстрованих користувачів і системні служби, а також надає інформацію про продуктивність комп'ютера, запущені застосунки, процеси та використання ЦП. Диспетчер завдань також може керувати властивостями процесора, запускати та зупиняти служби, встановлювати пріоритети процесів і примусово завершувати процеси.

На рис. 1.6 зображено інтерфейс даного ПЗ.

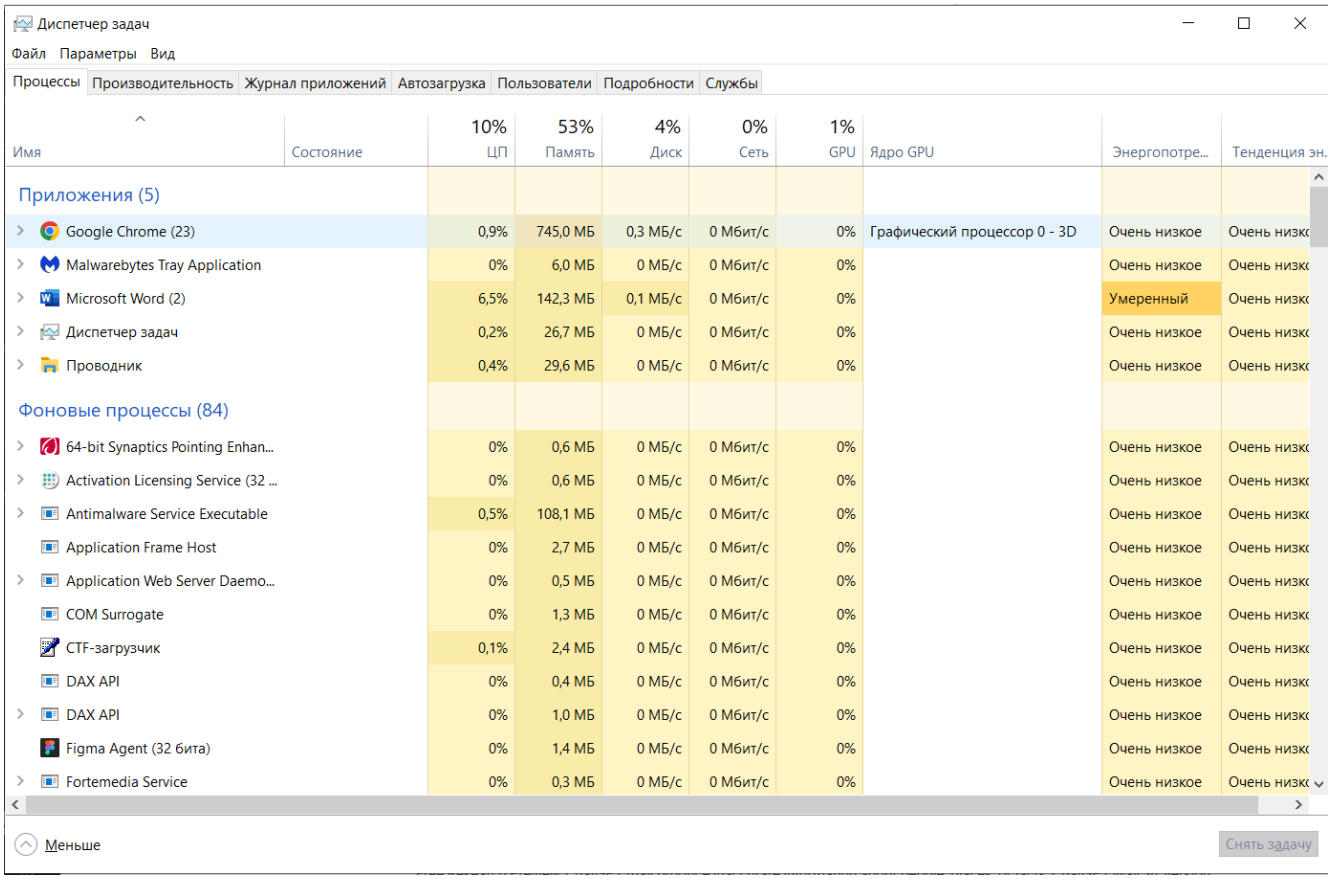


Рисунок 1.6 – Інтерфейс диспетчера задач

Основні функції:

– відображення список запущених процесів, дозволяючи переключатися на них та закривати;

– надання статистичних даних щодо використання ЦП, пам’яті, диска, Wi-Fi та графічного процесора, включаючи графіки та додаткову інформацію.

Недоліки:

– відсутність діалогу з користувачем;

– немає рекомендацій щодо стабільності роботи застосунків в умовах багатозадачності;

– відсутність накопичення даних та їх перегляд.

SolarWinds Server & Application Monitor – це інструмент моніторингу апаратного забезпечення, який може виявити проблеми з продуктивністю в мережі. Інструмент може відстежувати роботу процесора, пам'яті, фізичного дискового простору, швидкість обертання вентиляторів та живлення. Вся ця інформація відображається у вигляді інформаційної панелі, яка забезпечує повну видимість стану мережі та використання Windows-застосунків, Linux-застосунків і серверних ресурсів, незалежно від того, розташовані вони локально або в хмарі [10].

На рис. 1.7 зображено інтерфейс даного застосунку.

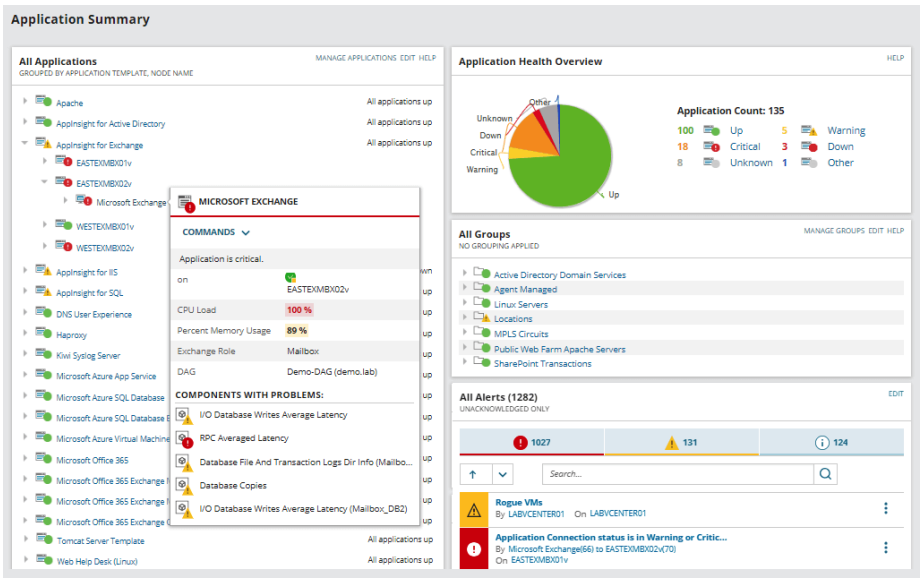


Рисунок 1.7 – Інтерфейс SolarWinds Server & Application Monitor

Основні функції:

– відстежує ключові статуси серверів;
– інтегрується з модулями управління мережею;
– система сповіщень;
– привабливі графіки даних.

Недоліки:

– відсутність діалогу з користувачем;
– немає рекомендацій щодо стабільності роботи застосунків в умовах багатозадачності;
– відсутність накопичення даних та їх перегляд.

CPUID HWMonitor – один з найпоширеніших інструментів моніторингу обладнання в цьому списку. HWMonitor відстежує стан здоров'я, напругу, температуру та стан вентиляторів підключених пристроїв. Інформація відображається у вигляді списку з розбивкою по продуктивності підключених пристроїв. Всі дані відображаються в режимі реального часу, щоб могли бути в курсі продуктивності пристроїв у мережі [10].

Основні функції:

– відстежує стан обладнання;
– простий у використанні;
– безкоштовна версія.

Недоліки:

– відсутність діалогу з користувачем;
– немає рекомендацій щодо стабільності роботи застосунків в умовах багатозадачності;
– відсутність накопичення даних та їх перегляд.

На рис. 1.8 зображено інтерфейс даного застосунку.

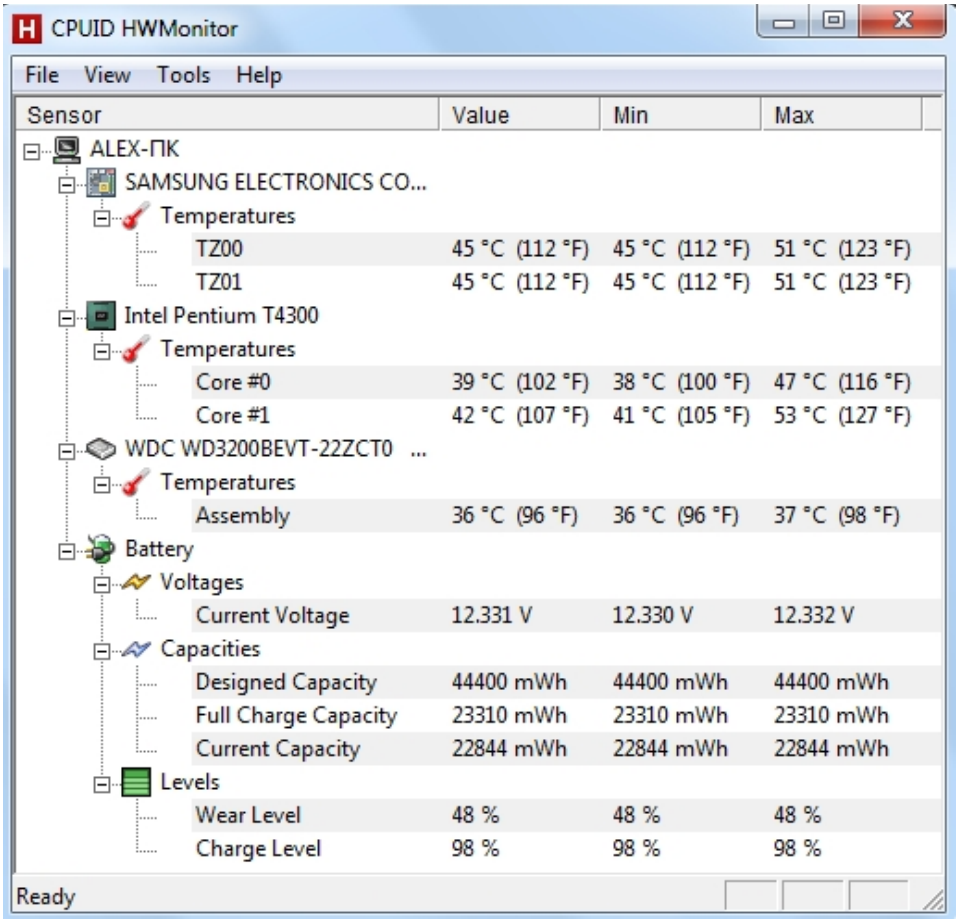


Рисунок 1.8 – Інтерфейс CPUID HWMonitor

В результаті аналізу аналогів систем для моніторингу комп’ютерів та інтернет мережі було визначено, що необхідно розробити такий застосунок, щоб кожен користувач міг користуватися, щоб він був інтуїтивно зрозумілим та мав сучасний дизайн. Всі програмні застосунки, які існують більш спрямовані на професійних користувачів такі як, розробники, адміністратори та інші люди які в цьому розбираються. Звичайні користувачі, які навчаються або працюють в інших сферах, буде важко зрозуміти як працюють дані програмні забезпечення. Також ці програмні забезпечення мають різноманітний функціонал, який може не знадобитися звичайному користувачу комп’ютера.

Отже, визначено, що застосунок буде розроблений такий, щоб кожен міг з легкістю в ньому розібратися, щоб функціонал був лише з найбільш важливих функцій, щоб був інтуїтивно зрозумілим та з гарним, приємний дизайном.

1.4 Постановка задачі

Сформуємо технічне завдання:

1) назва БКР: «Система моніторингу стабільності інтернет-конференції в умовах багатозадачності»;

2) мета роботи: дослідження та аналіз стабільності інтернет-конференції в умовах багатозадачності за допомогою системи моніторингу на мові програмування C# та платформі Windows Forms;

3) завдання роботи:

а) дослідити предметну сферу, провести аналіз літератури та публікацій;

б) дослідити вплив інтернет конференцій на споживання ресурсів пристроїв: пам’яті, центрального процесора (ЦП), а також трафіку передачі даних та ін.;

в) проаналізувати аналогічні системи для аналізу стабільності інтернет-конференції;

г) визначити функціонал застосунку, визначити засоби для реалізації система моніторингу;

д) розробити застосунок за допомогою обраних технологій;

е) протестувати розроблене програмне забезпечення (ПЗ);

ж) скласти звіт;

4) функціональні вимоги до застосунку:

а) виведення інформації про такі параметри, як CPU, фізична й віртуальна пам’яті, інтернет трафік та інформації про всі активні процеси;

б) візуалізація отриманої інформації, за допомогою інформативних графіків;

в) виведення повідомлень при досягненні критичних значень параметрів (графічні та аудіо);

5) технічні вимоги до застосунку:

а) мова програмування: C#;

б) інтерфейс користувача: графічний;

в) платформа: Windows Forms;

г) вихідні дані: звіт про стан інтернет-конференцій (стабільність, швидкість завантаження тощо) та про фізичний стан комп'ютера (ЦП, пам'ять тощо);

6) етапи реалізації застосунку:

а) дослідження предметної сфери та пошук необхідної літератури;

б) розробка прототип застосунку;

в) написання коду та тестування застосунку;

г) налагодження програмного застосунку;

д) складання звіту;

7) терміни виконання проєкту:

а) дослідження предметної сфери та пошук необхідної літератури – 2 тижня;

б) розробка прототип застосунку – 2 дні;

в) написання коду та тестування застосунку – 2 тижня;

г) налагодження програмного застосунку – 2 дні;

д) складання звіту – 1 тиждень;

8) очікуваний результат:

а) розроблений застосунок, що дозволить:

б) виведення інформації про такі параметри, як CPU, фізична й віртуальна пам'яті, інтернет трафік та інформації про всі активні процеси;

в) візуалізація отриманої інформації, за допомогою інформативних графіків;

г) виведення повідомлень при досягненні критичних значень параметрів (графічні та аудіо);

9) вимоги до виконавця проєкту:

а) знання мови програмування C#;

б) вміння працювати з Windows Forms;

10) основні ризики проєкту:

а) затримка в розробці через технічні проблеми;

б) проблеми з якістю даних, що може призвести до неточностей;

в) зміна у вимогах до застосунку;

11) основні користувачі програмного продукту: користувачі різних сфер;

12) критерії успішності проєкту:

а) розроблений застосунок має високу точність моніторингу;

б) застосунок має простий та зручний інтерфейс;

в) застосунок має працювати стабільно та швидко;

г) розроблений застосунок виконує весь визначений функціонал;

д) застосунок розроблений в зазначені терміни.

Було встановлено основні задачі та вимоги до роботи, для досягнення поставленої мети – дослідження та аналізу стабільності інтернет-конференції в умовах багатозадачності за допомогою системи моніторингу на мові програмування C# та платформі Windows Forms.

Висновки до розділу 1

Під час написання першого розділу було досліджено предметну сферу, а саме використання відеоконференцій та їх ресурсоспоживання. Визначено, що інтернет-конференції – один з найпоширеніших видів зв’язку сьогодення, який об'єднує людей з усього світу в онлайн середовищі для обговорення спільних інтересів та проблем.

Досліджено споживання ресурсів комп’ютера при використанні різних платформ для конференцій. Здійснено порівняльний аналіз ресурсоспоживання інтернет-конференцій.

Проведено аналіз аналогічних систем моніторингу, визначено їх переваги та недоліки. На основі цього дослідження було визначено вимоги до застосунку:

– простий та інтуїтивно зрозумілий інтерфейс з сучасним дизайном;

– логічно продуманий і не перевантажений зайвими можливостями функціонал.

Було встановлено основні задачі та вимоги до виконання бакалаврської кваліфікаційної роботи для досягнення поставленої мети.

2 МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Засоби реалізації застосунку

Для досягнення поставленої мети розробки застосунку потрібно визначити засоби реалізації. Для цього було проведено аналіз існуючих засобів та обрано необхідні.

Для розробки застосунку обрано:

– мову програмування C#;
– середовище розробки Visual Studio;
– платформу Windows Forms;
– програмну технологію .NET Framework;
– базу даних SQLite.

2.1.1 Мова програмування C#

C# – мова програмування яка є об'єктно-орієнтованою і типобезпечною. Дана мова допомагає розробляти різноманітні типи безпечних та надійних застосунків, які працюють в .NET. Ця мова походить із сімейства мов C і буде знайома тим, хто володіє мовами C, C++, JavaScript та Java [11].

C# – є мовою високого рівня загального призначення і підтримує декілька парадигм. Дана мова буда розроблена Андерсом Хейлсбергом компанією Microsoft та випущена в 2000 році як частина платформи Microsoft .NET. У 2002 році була затверджена як міжнародний стандарт Ecma (ECMA-334) та у 2003 році як ISO/IEC (ISO/IEC 23270). Компанія Microsoft разом з Visual Studio та .NET Framework представила мову C#. Через 10 років було випущено ще різні програмні продукти такі, як Visual Studio Code, Roslyn та Unified .NET Platform. Усі вони підтримують C# та є безкоштовними, кросплатформними та з відкритим кодом.

C# розроблена для спільної мовної інфраструктури (CLI), яка описує виконуваний код та середовище виконання. Це дозволяє використовувати декілька

мов високого рівня на різних комп'ютерних платформах та архітектурах.

CLR – це компонент віртуальної машини, який керує виконанням програм, написаних на мовах, що використовують .NET Framework. Деякі з його прикладів включають C#, F# тощо.

Вихідний код .NET компілюється в CLI. Код виконується під час виконання після того, як CIL перетворюється на нативний код. Це робиться за допомогою компілятора JIT (рис.2.1).

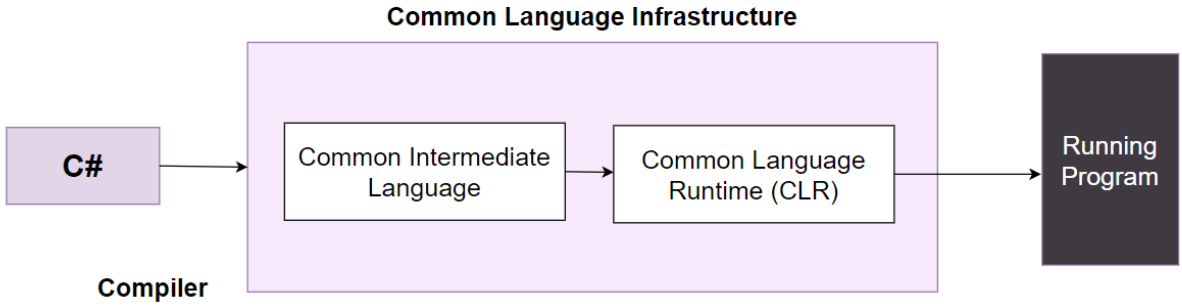


Рисунок 2.1 – Компілювання коду в CLI

Кожна мова програмування має свої переваги та недоліки, а тому потрібно визначити їх [12].

До переваг відносяться:

– вказівники не є обов'язковими в C#, але вони можуть бути використані;
– класи можуть бути визначені всередині класів у C#;
– програми менш схильні до помилок, оскільки небулеві змінні не можуть використовуватися як умови;
– C# має можливості рефлексії, тобто може переглядати та змінювати свою структуру під час виконання;
– у C# немає необхідності оголошувати функції та класи;
– класи та функції можуть бути визначені в будь-якому порядку в C#;
– застосунки на C# можна виконувати в обмеженій пісочниці;

– усі змінні в C# автоматично ініціалізуються значеннями за замовчуванням перед використанням;

– у C# не існує глобальних змінних або функцій, оскільки всі вони належать до класу.

До недоліків C# можна віднести наступне:

– код на C# потрібно компілювати щоразу, коли в нього вносяться зміни. Це може призвести до багатьох помилок або багів, якщо код не буде ретельно протестований кожного разу;

– застосунок .NET потребує платформи Windows для виконання, і це також стосується C#, оскільки вона є частиною застосунку .NET. Однак, більшість компаній надають перевагу Linux-серверам, оскільки вони дешевші;

– старіші фреймворки .NET не підтримуються Microsoft після кількох оновлень операційної системи. Крім того, оновлення пов'язані з великими витратами, оскільки вони повинні бути протестовані та затверджені перед тим, як їх можна буде розгортати.

У C# набагато більше переваг у порівнянні з недоліками. Це через те, що C# можна використовувати для вебзастосунків, служб Windows тощо, а також мова корисна для масштабування.

C# має велику кількість сфер застосувань. Дану багатоцільову мову програмування можна використовувати для створення:

– клієнтських програми, бібліотек, компонентів та сервісів для Windows;
– вебзастосунків;
– API-інтерфейс;
– застосунків для iOS;
– програм для Android;
– інструментів сумісності (наприклад, віджети та скрипти Excel);
– ігрових систем;
– ігор;
– машинного навчання та програм штучного інтелекту;

– блокчейну.

C# є досить популярною мовою, оскільки використовується для веброзробки, мобільної розробки тощо. Також WPF, UWP, WinForms можна використовувати для створення різноманітних Windows-застосунків. Застосунки для Android, iOS можна створювати за допомогою мобільної розробки на C#. Отже, C# можна використовувати для створення будь-якого типу програмного забезпечення. Вона є однією з найкращих та затребуваних мов в IT-сфері.

2.1.2 Середовище розробки Visual Studio

Visual Studio – це потужний інструмент розробника, за допомогою якого можна виконати весь цикл розробки в одному місці. Це комплексне інтегроване середовище розробки (IDE), за допомогою якого можна писати, редагувати, налагоджувати та збирати код, а потім розгортати програму. Окрім редагування та налагодження коду, Visual Studio включає компілятори, інструменти для завершення коду, контроль вихідних кодів, розширення та багато інших функцій для покращення кожного етапу процесу розробки ПЗ [13].

Visual Studio надає розробникам багатофункціональне середовище розробки для ефективного та спільного створення високоякісного коду:

– інсталятор на основі робочого навантаження;

– потужні інструменти та функції кодування;

– підтримка багатьох мов – код на C++, C#, JavaScript, TypeScript, Python та інших;

– крос-платформна розробка;

– інтеграція контролю версій.

Visual Studio IDE дозволяє програмістам створювати та редагувати код спільно. Вона має розширення та включає інструменти, які допомагають у створенні коду. Це дозволяє бачити в реальному часі, над чим працюють колеги по команді, щоб зменшити надмірність.

У ньому є завершення коду з підсвічуванням синтаксису, модель кодування

з AI для допомоги в програмуванні та інструмент аналізу, який допоможе з налагодженням.

Переваги використання середовища для розробки Visual Studio:

– підтримується на різних платформах;

– підтримує широкий спектр мов програмування;

– має багато вбудованих візуальних інструментів для розробки інтерфейсу користувача;

– надає різні інтегровані інструменти для управління проєктом;

– має зручний редактор коду.

Дане середовище існує вже понад 25 років, тому існує безліч ресурсів, які полегшать життя за допомогою цього програмного забезпечення.

Доступні сотні розширень, які допоможуть зробити будь-що – від інтеграції GitHub до потужного інструменту для підвищення продуктивності та полегшення роботи. Це також найпопулярніша IDE для C++, яка має безкоштовну версію.

Існують спеціальні спільноти, де можна спілкуватися з іншими розробниками і навіть домовлятися про зустрічі. З такою потужною системою підтримки можна бути впевненим, що будь-які запитання та проблеми будуть вирішеними.

2.1.3 Програмна технологія .NET Framework

.NET розроблений компанією Microsoft і являє собою програмний фреймворк. Він працює переважно на Microsoft Windows. Цей фреймворк також допоміг у створенні різних платформ, які працюють з вбудованими пристроями, мобільними обчисленнями, плагінами для веббраузерів тощо [14].

Визначимо деякі з завдань, які вирішуються за допомогою фреймворку .NET:

– фреймворк .NET забезпечує середовище, яке усуває різні проблеми, що виникають через інтерпретовані або скриптові середовища;

– фреймворк .NET забезпечує об'єктно-орієнтоване середовище програмування. У цьому середовищі об'єктний код виконуватися та зберігатися локально. Крім того, код може виконуватися віддалено;

– фреймворк .NET також надає середовище, яке забезпечує однаковий досвід розробників у різних типах застосунків, які можуть суттєво відрізнятися, таких як вебзастосунки, застосунки на базі Windows тощо;

– завдяки середовищу виконання, яке забезпечує платформа .NET, розгортання програмного забезпечення, а також конфлікти версій є мінімальними;

– фреймворк .NET створено повністю на основі галузевих стандартів, тому код на основі .NET може бути інтегрований з будь-яким кодом;

– середовище .NET забезпечує безпечне виконання коду. Це стосується і коду, створеного невідомими особами.

Виділимо деякі з застосунків і служб, які можна створити за допомогою .NET framework:

– програми Windows Presentation Foundation (програми WPF);
– консольні програми;
– сервісно-орієнтовані програми (з використанням WCF);
– програми Windows Forms;
– служби Windows;
– програми ASP.NET;
– програми з підтримкою робочого процесу (з використанням WF).

Відомі технологічні компанії, а також найкращі програми Microsoft, як відомо, були створені за допомогою технологій .NET та Windows CE. Платформа програмування .NET (з такими реалізаціями, як .NET Core, NET Framework, Xamarin, Mono, Unity або Godot) дозволяє розробникам програмувати в єдиній кодовій базі та створювати програми, які працюють під Mac OS Windows та Linux.

Переваги використання .NET Framework для розробки застосунків полягають у наступному:

– спільні бібліотеки коду;

– відсутність «написання з нуля»;
– безпека доступу до коду;
– відкритий вихідний код (безкоштовний на GitHub);
– вбудовані елементи керування інтерфейсом користувача;
– гнучкі/затребувані застосунки та сервіси;
– один з найкращих програмних фреймворків Microsoft.

2.1.4 Платформа Windows Forms

Windows Forms – це платформа для інтерфейсу користувача для створення класичних застосунків Windows. Дана платформа забезпечує ефективний спосіб способів розробки класичних застосунків за допомогою конструктора візуального в Visual Studio. У Windows Forms можна розробляти графічно складні застосунки. Там передбачено безліч елементів для управління, які можна додавати в форми та взаємодіяти з ними за допомогою коду. За допомогою середовища розробки Visual Studio, можна створювати інтелектуальні клієнтські застосунки Windows Forms, які виводять інформацію, запитують, що необхідно вести користувачу та взаємодіють з ним [15].

WinForms – це фреймворк інтерфейсу користувача, який використовує Windows API для створення десктопних застосунків у Windows. Це традиційний фреймворк інтерфейсу для десктопних застосунків Windows.

Інформаційні технології стали важливою частиною повсякденого життя. Вони принесли у світ багато нових технологій. Головне, що вони надали світові – це мову програмування та кодування, за допомогою якої розробляються різноманітні програми. Однією з таких мов є .Net, яка є дуже корисним фреймворком, що допомагає у створенні хороших застосунків та вебсайтів. За допомогою .Net можна робити багато речей. Вона включає бібліотеку класів Windows Forms, яка надає різні компоненти графічного інтерфейсу користувача (GUI).

WinForms – це, по суті, бібліотека класів у .Net. Вона включає в себе всі

компоненти графічного інтерфейсу, які в подальшому використовуються при створенні застосунків для різних клієнтів. Багато клієнтів хочуть, щоб Net виконувався так, щоб забезпечував кращу функціональність і зовнішній вигляд. Діаграми Windows Forms є одним з таких корисних інструментів, який дозволяє користувачеві отримати кращий графічний інтерфейс для програми. Windows Forms працює досить добре, допомагаючи користувачеві ефективно отримувати доступ до програми та керувати нею. Це є одним з найкращих способів реалізації програми. Він має багато переваг, що робить його ідеальним для використання [16].

WinForms працює відповідно до фреймворку .Net для забезпечення функціональності. В основному він чекає на введення даних, а потім допомагає внести зміни. Він також відомий під назвою Winforms, він допомагає додавати всі компоненти графічного інтерфейсу до .Net, до яких в подальшому може отримати доступ користувач. Форма Windows має різні методи у своїй бібліотеці класів, які надають можливість додавати компоненти графічного інтерфейсу.

Переваги Windows Forms:

– легкість використання;
– широкий набір елементів керування;
– створює баланс між функціональністю та зовнішнім виглядом програми;
– підтримка мов програмування;
– інтеграція з іншими технологіями Microsoft;
– робить завдання додавання компонентів інтерфейсу дуже простим;
– створює простий спосіб встановлення графічного інтерфейсу відповідно до вимог програми;
– ефективний спосіб забезпечити ефективну роботу графічного інтерфейсу навіть на старих комп'ютерах.

Отже, Windows Forms дозволяє швидко створювати прості та середньої складності застосунки з графічним інтерфейсом користувача на платформі Windows. Дана платформа підходить для широкої сфери проєктів та дозволяє

розробникам зосередитися на функціональності застосунку, не витрачаючи багато часу на навчання складних графічних технологій.

2.2 Класи та методи для вирішення задачі

Застосунок для моніторингу стабільності інтернет-конференції передбачає отримання значень про стан процесора, пам’яті, диска, інтернет-трафіку та про процеси.

System.Diagnostics є простором імен в .NET Framework, який надає класи і інструменти для взаємодії з діагностикою системи, процесами та іншими системними ресурсами в середовищі .NET. Цей простір імен містить різні класи, які можна використати для моніторингу та налагодження застосунків [17].

в System.Diagnostics існують наступні класи:

– клас *Process* дозволяє отримувати доступ до інформації про процеси, керувати параметрами процесу та взаємодіяти з ними;

– клас *EventLog* дозволяє записувати події в системний журнал подій і читати записані події з журналу;

– клас *Stopwatch* вимірювати час виконання фрагментів коду з високою точністю. Дозволяє розробникам визначати час виконання певних операцій або весь час виконання програми;

– клас *PerformanceCounter* дозволяє отримувати доступ до параметрів використання процесора, пам'яті, дискового простору тощо. Він дозволяє вимірювати та моніторити різні показники продуктивності.

Щоб отримати відомості про поточний стан процесора, диска та пам’яті, буде використано клас *PerformanceCounter*. Для отримання даних про процеси буде використано клас *Process*. Даний клас дозволяє виконувати дії з процесами, зокрема завантажити інформацію про процеси, потоки та прив'язані до процесу модулі [18].

Дані про інтернет-трафік будуть отримані за допомогою простором імен *System.Net.NetworkInformation*.

System.Net.NetworkInformation є простором імен в .NET Framework, який надає класи, які мають інформацію про мережеві параметри та доступ до окремих мережевих ресурсів. Він дозволяє взаємодіяти з мережевими інтерфейсами, отримувати статус підключення, отримувати інформацію про IP-адреси, DNS, маршрутизацію, стан з'єднання та інше [19].

Основні класи, які доступні в *System.Net.NetworkInformation*:

– клас *NetworkInterface* дозволяє отримувати інформацію про мережеві інтерфейси на пристрої, такі як назва інтерфейсу, фізична адреса, швидкість передачі даних, стан підключення;

– клас *IPAddress* представляє IP-адресу (IPv4 або IPv6) і надає різноманітні функції для роботи з адресами, отримання інформації про тип адреси, порівняння адрес тощо;

– клас *Ping* дозволяє виконувати перевірку доступності хоста або IP-адреси в мережі за допомогою протоколу ICMP;

– клас *Dns* дозволяє отримувати інформацію про систему доменних імен, виконувати DNS-розрішення (перетворення доменного імені на IP-адресу) та виконувати операції пов'язані з ним.

При досягненню критичних значень, які будуть встановлені, буде голосове оповіщення про те, скільки відсотків процесора використовується, також буде виводитися повідомлення.

Для голосового оповіщення використовується *System.Speech.Synthesis*. *System.Speech.Synthesis* є простором імен в .NET Framework, який надає класи та функціональність для синтезу мовлення (Text-to-Speech) в програмах, розроблених на платформі .NET [20].

Основні компоненти *System.Speech.Synthesis*:

– клас *SpeechSynthesizer* є ключовим компонентом System.Speech.Synthesis і представляє синтезатор мовлення. Він дозволяє генерувати мовлення з тексту, налаштовувати параметри синтезу, контролювати вимову, регулювати швидкість та голос синтезованого мовлення;

– клас *InstalledVoice* представляє встановлені голоси на системі, які використовуються для синтезу мовлення. Він надає інформацію про доступні голоси, такі як назва голосу, мова, вимоги до ресурсів, швидкість та інші атрибути.

Для виведення повідомлення використовується клас *MessageBox* у Winforms, який надає простий спосіб виводити повідомлення користувачеві у застосунку . Він дозволяє відображати вікно з текстом повідомлення, іконкою, кнопками та іншими елементами для спілкування з користувачем. *MessageBox* є простим, але потужним засобом взаємодії з користувачем у віконних застосунках. Він може бути використаний для показу повідомлень, попереджень або підтверджень.

2.3 Обгрунтування вибору бази даних

Мова структурованих запитів (SQL) – це стандартизована мова програмування, яка використовується для керування реляційними базами даних та виконання різних операцій над даними в них. Вперше створена в 1970-х роках, SQL регулярно використовується не тільки адміністраторами баз даних, але й розробниками, які пишуть скрипти для інтеграції даних, а також аналітиками даних, які прагнуть налаштувати і запустити аналітичні запити [21, 22].

SQL використовується для наступного:

– модифікації структур таблиць та індексів бази даних;
– додавання, видалення та оновлення рядків даних;
– отримання підмножин інформації з реляційних СКБД.

При виборі бази даних (БД) було враховано зручність та швидкість використання. Обиралося серед двох баз MySQL та SQLite. Так як застосунок націлений на моніторинг працездатності та забезпечення стабільності, тому обрано БД SQLite через те, що MySQL потрібно підключати за допомогою сервера, а це додаткове навантаження на процесор, тому для даного застосунку обрано БД SQLite.

SQLite – це найпоширеніша база даних у світі, яка має велику кількість застосувань, включаючи кілька гучних проєктів.

SQLite – це вбудований рушій баз даних SQL. Порівнюючи з іншими БД, SQLite не має окремого сервера. Дана БД записує та читає безпосередньо до звичайних дискових файлів. Повна SQL база даних з декількома таблицями, індексами, тригерами і представленнями міститься в одному файлі на диску. Це роить SQLite популярною базою [23].

Основні особливості SQLite:

– дуже простий у використанні. Можна створювати базу даних, таблиці та виконувати запити без необхідності встановлення та налаштування сервера бази даних;

– зберігає всю інформацію у єдиному файлі бази даних, що дозволяє легко переносити та розповсюджувати базу даних з одного середовища на інше;

– підтримується на багатьох платформах і має бібліотеки для різних мов програмування;

– невеликий розмір та низьки витрати ресурсів;

– підтримка стандартів SQL.

Отже, SQLite є гарним вибором, тому що дана база даних забезпечує простоту, швидкодію та легкість у використанні.

Висновки до розділу 2

Під час написання другого розділу було визначено основні засоби для реалізації застосунку для моніторингу стабільності інтернет-конференцій. Для реалізації ПЗ було обрано: мову C#, програмну технологію .NET Framework, середовище розробки Visual Studio, платформу Windows Forms та базу даних SQLite.

Визначено переваги та недоліки C# та де використовується дана мова. Описано технологію .NET Framework, які завдання вирішуються за допомогою неї та які застосунки можна створити. Визначено що таке Winforms та переваги використання. Виділено переваги використання середовище для розробки Visual Studio IDE.

Порівняно дві бази даних та обрано ту, яка краще підходить для розробки майбутнього застосунку. Визначено, що БД SQLite для застосунку-моніторингу працездатності комп'ютера є кращою ніж MySQL, тому що вона забезпечує простоте використання, швидкодію, легкість у використанні та не потребує підключення до сервера.

Описано класи, за допомогою яких отримуються параметри комп'ютера. Для параметрів поточного стану ЦП, диска та пам'яті використано клас *PerformanceCounter*. Дані про інтернет-трафік отримані за допомогою простору імен *System.Net.NetworkInformation*.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ЗАСТОСУНКУ

Застосунок призначений для моніторингу працездатності комп'ютера має бути інтуїтивно зрозумілий, зручний у використанні та інформативним.

3.1 Розробка моделі програмного забезпечення для моніторингу стабільності інтернет-конференції в умовах багатозадачності

Уніфікована мова модифікації або Unified Modeling Language (UML) – це стандартизована мова моделювання загального призначення в галузі об'єктно-орієнтованої розробки програмного забезпечення. UML є методом візуального представлення структури, проєктування та реалізації складних програмних систем. Коли програма містить тисячі рядків коду, може бути важко спостерігати за взаємозв'язками та ієрархіями в програмній системі. Ця програмна система може бути розділена на компоненти та підкомпоненти за допомогою діаграм UML [24].

UML пропонує стандартний спосіб написання схем системи, включаючи концептуальні речі, такі як бізнес-процеси та системні функції, а також конкретні речі, такі як оператори мови програмування, схеми баз даних і повторно використовувані програмні компоненти [25].

Існує дві основні переваги UML:

– зручність читання: представлення всієї архітектури програмного забезпечення в блок-схемі, діаграмі класів, діаграмах ER робить проєкт більш читабельним;

– можливість повторного використання: після того, як система стане більш читабельною та буде розбита на частини, стане легше ідентифікувати зайві та подібні модулі; таким чином підвищується можливість повторного використання.

Існує два основних способи використання діаграм UML:

– передній дизайн. Моделювання та проєктування виконуються перед кодуванням програми. Зазвичай прямий дизайн використовується, щоб допомогти розробникам краще побачити систему, яку вони намагаються створити;

– зворотній дизайн. Моделювання виконується після написання коду, а діаграми UML служать документацією для робочого процесу проєкту. Це може допомогти розробникам побачити розвиток проєкту таким, яким він був насправді, щоб покращити його в майбутньому.

Незалежно від того, чи використовуються діаграми UML до чи після кодування чи проєкту, вони забезпечують спосіб візуалізації багатьох аспектів проєкту та того, хто за яку діяльність відповідає.

Існують різні види діаграм, за допомогою яких можна побудувати модель програмного забезпечення.

Діаграма прецедентів або діаграма варіантів використання (use case diagram) – використовуються на етапі аналізу проєкту для визначення та розподілу функціональності системи. Вони поділяють систему на акторів та варіанти використання. На рис. 3.1 зображено діаграму прецедентів.

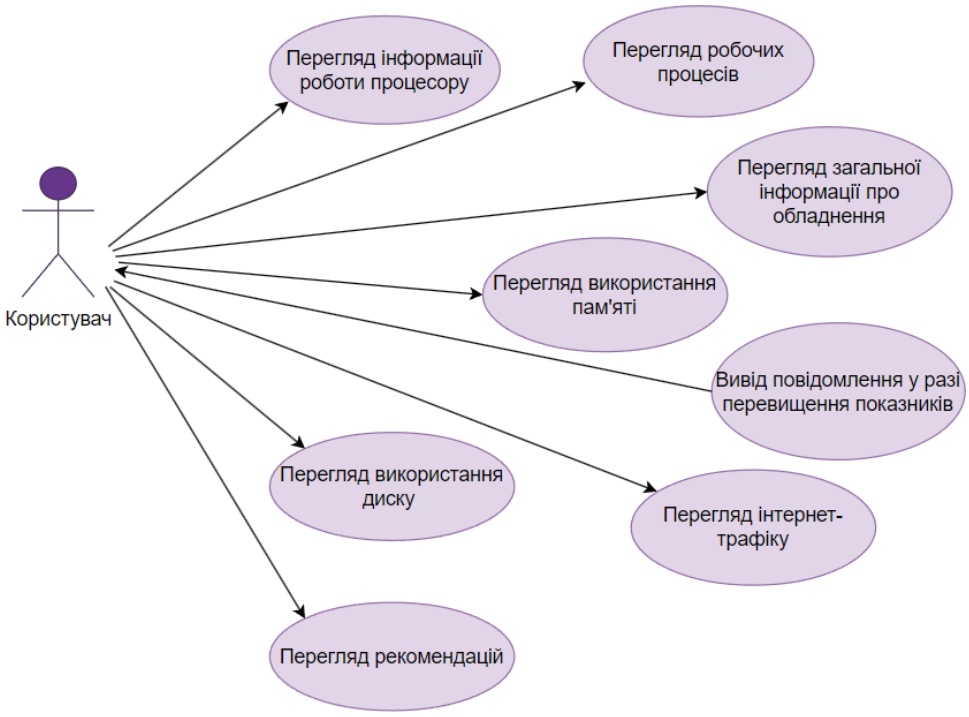


Рисунок 3.1 – Діаграма прецедентів

Актори представляють ролі, які можуть виконувати користувачі системи. Такими користувачами можуть бути люди, інші комп'ютери, апаратні засоби чи навіть інші програмні системи. Єдиним критерієм є те, що вони повинні бути зовнішніми по відношенню до частини системи, яка розділена на варіанти використання. Вони повинні надсилати дії до системи і отримувати від неї назад вихідні дані.

Діаграма варіантів використання описує поведінку системи, коли один із цих учасників надсилає один конкретний стимул. Ця поведінка описана текстово. Він описує природу стимулу, який запускає варіант використання; вхідні дані та вихідні дані для інших акторів, а також поведінка, яка перетворює вхідні дані на виходи. У тексті варіанту використання також зазвичай описується все, що може піти не так під час певної поведінки, і які дії для виправлення стану вживатиме система.

Діагарми класів (class diagram) – найкращий спосіб ідентифікувати класи – розглядати всі «іменники» у випадках використання як класи, «дієслова» як методи класів, а відношення між акторами можна використовувати для визначення зв'язку між класами. Відношення або зв'язок між класами може бути або зв'язком «є-є», або «має-є», який можна легко ідентифікувати з випадків використання. На рис. 3.3 зображено діаграма класів.

Оскільки програмне забезпечення зазвичай базується на об'єктно-спеціальному програмуванні, діаграма класів є найбільш часто використовуваною діаграмою UML.

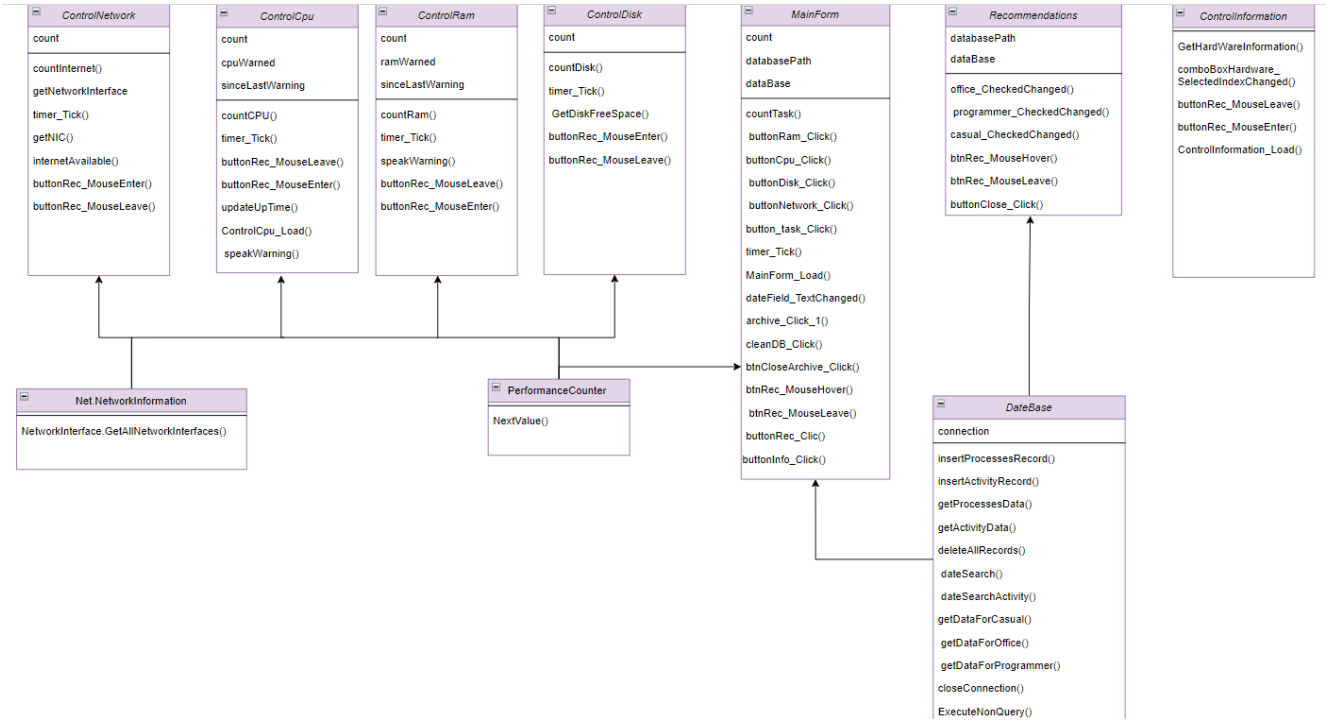


Рисунок 3.3 – Діаграма класів

Діаграми послідовності (sequence diagram) – використовуються для представлення або моделювання потоку повідомлень, подій і дій між об’єктами або компонентами системи. Час представлено у вертикальному напрямку, показуючи послідовність взаємодії елементів заголовка, які відображаються горизонтально у верхній частині діаграми. На рис. 3.2 представлено діаграма послідовності.

Діаграми послідовності використовуються в основному для розробки, документування та перевірки архітектури, інтерфейсів і логіки системи шляхом опису послідовності дій, які необхідно виконати для виконання завдання або сценарію. Діаграми послідовності є корисними інструментами проєктування, оскільки вони забезпечують динамічне уявлення про поведінку системи, яке може бути важко отримати зі статичних діаграм або специфікацій.

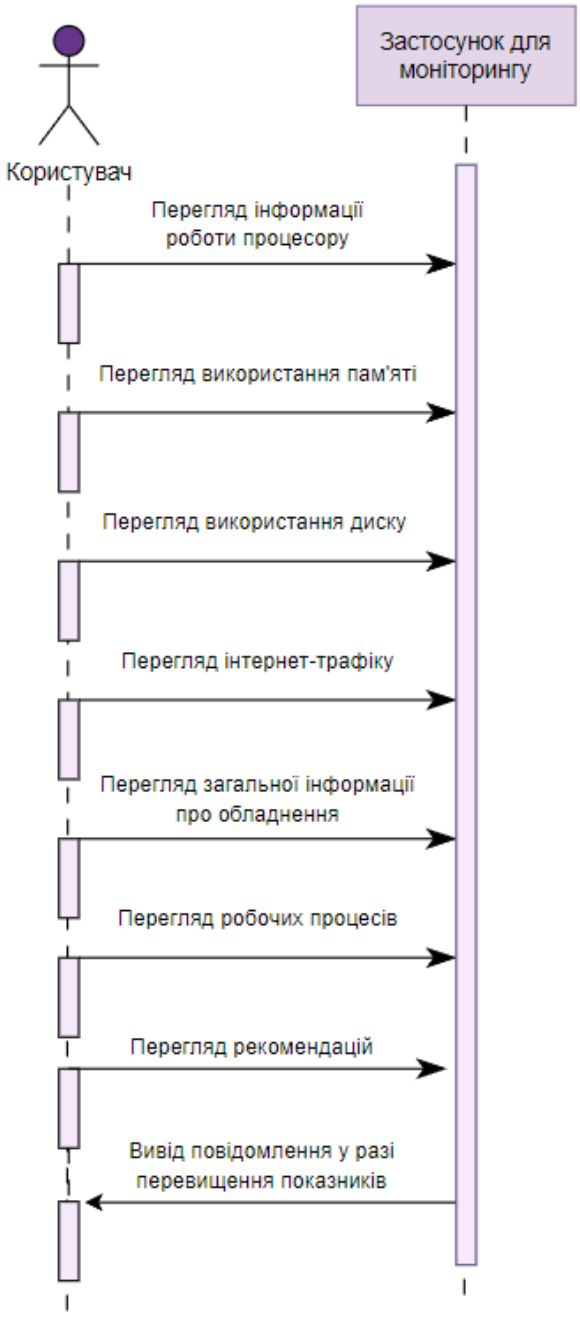


Рисунок 3.2 – Діаграма послідовності

Діаграма діяльності (activity diagram) – зазвичай використовується для моделювання бізнес-процесів, для моделювання логіки одного варіанту використання або для візуалізації детальної логіки бізнес-правила. На рис. 3.4 представлено діаграму активностей.

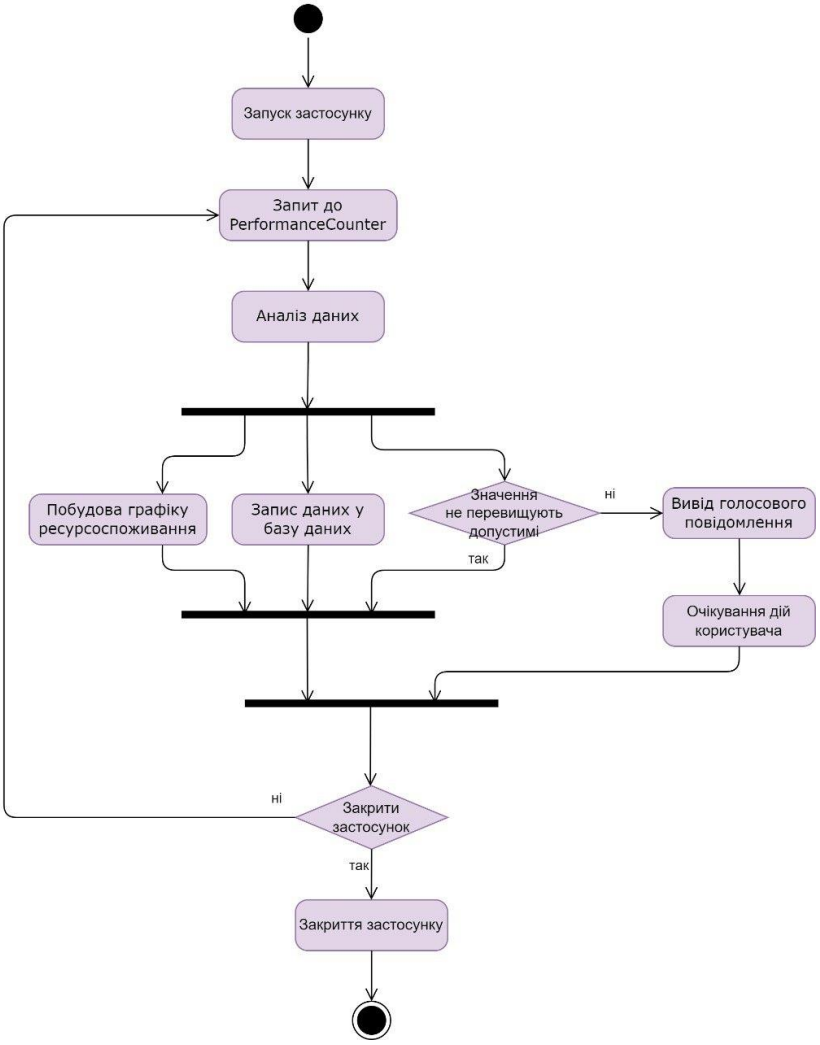


Рисунок 3.4 – Діаграма активностей

Складні потоки процесів у системі відображаються на діаграмі діяльності. Подібно до діаграми станів, діаграма діяльності також складається з дій, рішень, переходів, початкового та кінцевого станів і умов. Але відмінність полягає в тому, що діаграми стану знаходяться в контексті моделювання, а діаграма діяльності дає детальне уявлення про бізнес-логіку.

Діаграми кінцевого автомата (state machine diagram) – зображують стани, в яких може перебувати об’єкт, і переходи між цими станами. Фактично, в інших мовах моделювання цей тип діаграми прийнято називати діаграмою переходів станів або навіть просто діаграмою станів. На рис. 3.5 представлено діаграму станів.

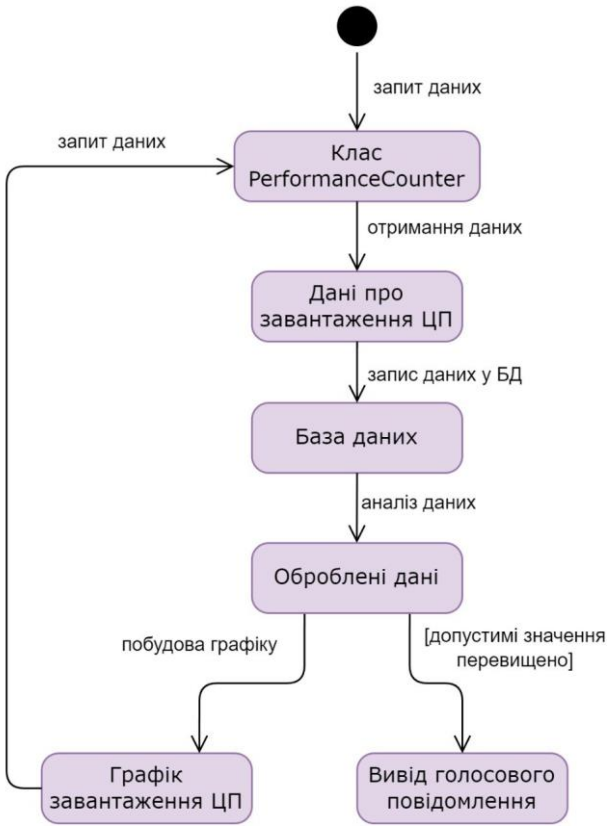


Рисунок 3.5 – Діаграма станів

Стан представляє етап у шаблоні поведінки об’єкта, і, подібно до діаграм активності, може мати початкові та кінцеві стани. Початковий стан, який також називають станом створення, – це стан, у якому перебуває об’єкт, коли його вперше створено, тоді як кінцевий стан – це стан, з якого не ведуть переходи. Перехід – це перехід від одного стану до іншого, який буде спричинено внутрішньою або зовнішньою подією об’єкта.

За допомогою різних UML-діаграм спроєктовано модель застосунку для моніторингу працездатності комп’ютера.

3.2 Розробка застосунку-моніторингу

У процесі розробки застосунку, який моніторить працездатність комп’ютера, був створений класичний застосунок для пристроїв на базі операційної системи Windows. В якості користувацького інтерфейсу виступає розроблений інтерфейс Windows Forms. Застосунок складається з головної форми, форми з

рекомендаціями та п’яти контролерів, а саме: контролер з інформацією про використання процесора, контролер з інформацією про використання пам’яті, контролер з інформацією про використання диска, контролер з інформацією про інтернет-трафік та контролер із загальною інформацією параметрів комп’ютера.

Даний застосунок містить наступні бібліотеки:

1) *System* – містить основні класи що визначають типи значень події та обробники подій події атрибути та інтерфейси які часто використовуються при створенні застосунків;

2) *System.IO* – містить типи що дають змогу здійснювати читання і запис у файли та потоки даних а також типи для базової підтримки файлів і папок;

3) *System.Runtime.InteropServices* – надає різноманітні члени, що підтримують COM-взаємодію і служби виклику платформи;

4) *System.Windows.Forms* – містить класи для створення застосунків Windows, які дозволяють оптимізувати розширені можливості інтерфейсу користувача операційної системи Microsoft Windows;

5) *System.Management* – надає доступ до багатого набору управлінської інформації та подій про систему пристрої та програми що входять до інфраструктури Windows Management Instrumentation;

6) *Microsoft.Win32* – надає два типи класів ті які обробляють події викликані операційною системою і ті які керують системним реєстром;

7) *System.Diagnostics* – надає класи що дають змогу здійснювати взаємодію із системними процесами журналами подій і лічильниками продуктивності;

8) *System.Net.NetworkInformation* – надає доступ до даних мережевого трафіку відомостей про мережеві адреси та повідомлень про зміну адрес для локального компютера;

9) *Microsoft.VisualBasic.Devices* – містить типи, які підтримують об'єкти My, пов'язані з пристроями у Visual Basic.

3.2.1 Опис бази даних

Під час розробки застосунку було створено базу даних «activity.db». Спочатку в БД було створено 2 таблиці «processes» та «activity».

Опис таблиці «processes», яка містить в собі інформацію щодо активних процесів та їх споживання пам’яті, представлений в табл. 3.1.

Таблиця 3.1 – Опис таблиці «processes»

Назва поля	Тип даних	Значення
processName	text	Назва активного процесу
memoryUsage	text	Використання пам’яті для кожного процесу
status	text	Статус процесу
upTime	text	Дата запуску процесу
createdDate	text	Дата створення запису

Опис таблиці «activity», яка містить в собі інформацію щодо використання ЦП та пам’яті в певний час, представлений в табл. 3.2.

Таблиця 3.1 – Опис таблиці «activity»

Назва поля	Тип даних	Значення
cpu	text	Використання ЦП
ram	text	Використання пам’яті
createdDate	text	Дата створення запису

Усі дані записуються до БД автоматично, тому за необхідністю можна відкрити усі записані дані та переглянути їх. Для запобігання зайвого витрачання ресурсів комп’ютера передбачено кнопку для очищення бази даних.

Далі створено ще три таблиці, які містять дані про різні програмні забезпечення, які використовуються найбільше в трьох різних сфер діяльності. Додано наступні таблиці: «casual», «office», «programmer». Опис таблиці «casual», яка містить в собі інформацію про програмні забезпечення, якими користуються

звичайні користувачі та їх вимоги до системи, представлений в табл. 3.3.

Таблиця 3.3 – Опис таблиці «casual»

Назва поля	Тип даних	Значення
program	text	назва ПЗ
os	text	вимоги до ОС
cpu	text	вимоги до процесора
ram	text	вимоги до ОЗП
gpu	text	вимоги до графічного процесора
memory	text	вимоги до наявності пам’яті

Опис таблиці «office», яка містить в собі інформацію про програмні застосунки, якими користуються офісні працівники, та їх вимоги до системи, представлений в табл. 3.4.

Таблиця 3.4 – Опис таблиці «office»

Назва поля	Тип даних	Значення
program	text	назва ПЗ
os	text	вимоги до ОС
cpu	text	вимоги до процесора
ram	text	вимоги до ОЗП
gpu	text	вимоги до графічного процесора
memory	text	вимоги до наявності пам’яті

Опис таблиці «programmer», що містить в собі інформацію про програмні застосунки, якими користуються програмісти, та їх вимоги до системи, представлений в табл. 3.5.

Таблиця 3.5 – Опис таблиці «programmer»

Назва поля	Тип даних	Значення
program	text	назва ПЗ
os	text	вимоги до ОС
cpu	text	вимоги до процесора
ram	text	вимоги до ОЗП
gpu	text	вимоги до графічного процесора
memory	text	вимоги до наявності пам’яті

Дані таблиці містять інформацію про найбільш популярне ПЗ, яким користуються користувачі різних сфер діяльності. Також в них зазначено вимоги для ОС, до процесора, до ОЗП, до графічного процесора та до наявності вільної пам’яті для того, щоб забезпечити стабільну роботу інтернет-конференцій в умовах багатозадачності.

3.2.2 Опис класів застосунку

Застосунок складається з 2 форм, 5 контролерів та класу БД. На головній формі розташовано вивід активних процесів та їх використання пам’яті, використання ЦП та пам’яті в певний час, вивід даних з БД, пошук даних за датою додавання до БД. Вивід рекомендацій для покращення стабільності роботи. Структура класу «MainForm» зображено на рис. 3.6.

Функції даного класу:

а) *countTask()* – визначення активних процесів, їх використання пам’яті, використання ЦП та ОЗП, запис до бази даних;

б) *MainForm_Load()* – формування таблиць з активними процесами, їх використання ОЗП та ЦП;

в) *timer_Tick()* – запуск лічильника;

г) *archive_Click_1()* –вивід даних з БД;

д) *btnRec_MouseHover()* – відображення рекомендацій;

е) *btnRec_MouseLeave()* – приховання рекомендацій;

ж) *buttonRec_Clic()* – відкриття вікна з рекомендаціями;

з) *cleanDB_Click()* – очищення таблиць БД;

и) *dateField_TextChanged()* – пошук даних в таблицях з БД;

к) *button_task_Click()* – перехід до активних процесів та вивід даних з БД;

л) *buttonInfo_Click()* – перехід до вікна з інформацією про апаратне забезпечення;

м) *buttonDisk_Click()* – перехід до вікна з розрахунком для диска;

н) *buttonRam_Click()* – перехід до вікна з розрахунком ОЗП;

о) *buttonCpu_Click()* – перехід до вікна з розрахунком ЦП;

п) *buttonNetwork_Click()* – перехід до вікна з розрахунком інтернет-трафіку.

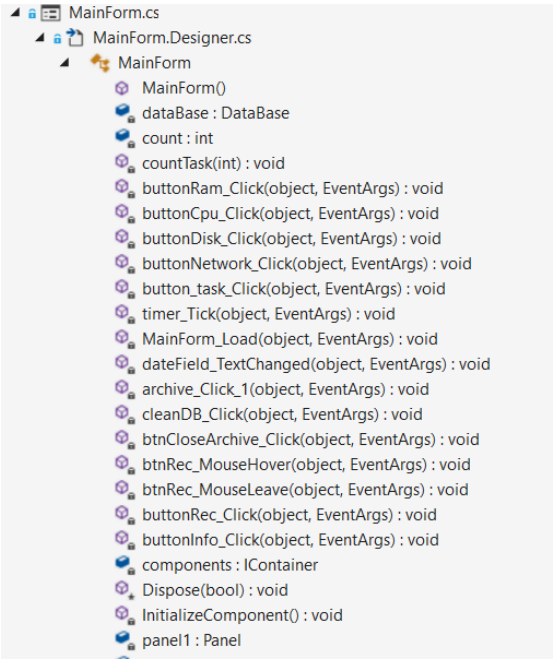


Рисунок 3.6 – Клас «MainForm»

На головній формі також розташовано усі контролери для зручного перемикання. На контролері «ControlCpu» розташовано інформація про процесор, вивід даних щодо використання процесора та графік, а також надання рекомендацій для стабільної роботи інтернет-конференцій в умовах багатозадачності. Структура класу «ControlCpu» зображено на рис. 3.7.

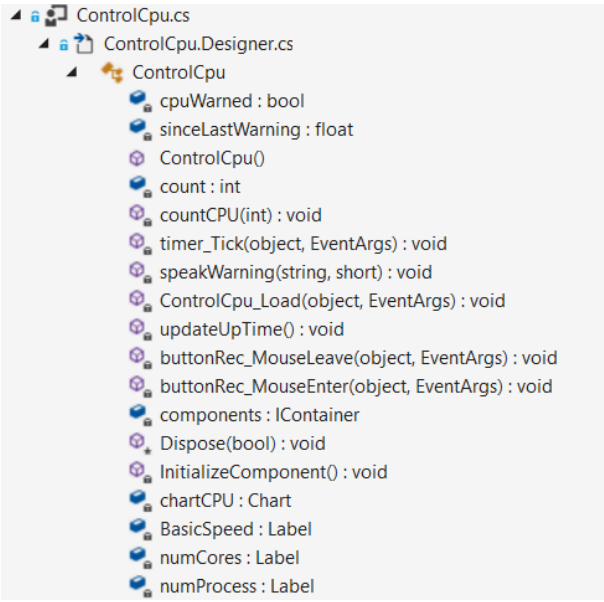


Рисунок 3.7 – Клас «ControlCpu»

Функції даного класу:

а) *countCPU()* – розрахунок використання процесора, інформація про процесор;

б) *timer_Tick()* – запуск лічильника;

в) *speakWarning()* – визначає голосове попередження;

г) *ControlCpu_Load()* – завантаження форми;

д) *buttonRec_MouseEnter()* – відображення рекомендацій;

е) *buttonRec_MouseLeave()* – приховання рекомендацій;

ж) *updateUpTime()* – розрахунок часу роботи комп’ютера.

На контролері «ControlRam» розташована інформація про ОЗП, вивід використання пам’яті та графік, а також надання рекомендацій для стабільної роботи інтернет-конференцій в умовах багатозадачності. Структура класу «ControlRam» зображено на рис. 3.8.

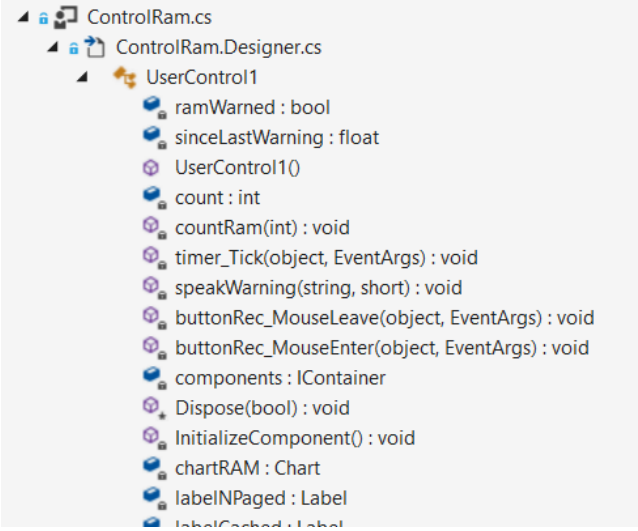


Рисунок 3.8 – Клас «ControlRam»

Функції даного класу:

а) *countRam()* – розрахунок використання оперативної пам’яті;

б) *buttonRec_MouseEnter()* – відображення рекомендацій;

в) *buttonRec_MouseLeave()* – приховання рекомендацій;

г) *timer_Tick()* – запуск лічильника.

На контролері «ControlDisk» розташована інформація про роботу дисків, а також надання рекомендацій для стабільної роботи інтернет-конференцій в умовах багатозадачності. Структура класу «ControlDisk» зображено на рис. 3.9.

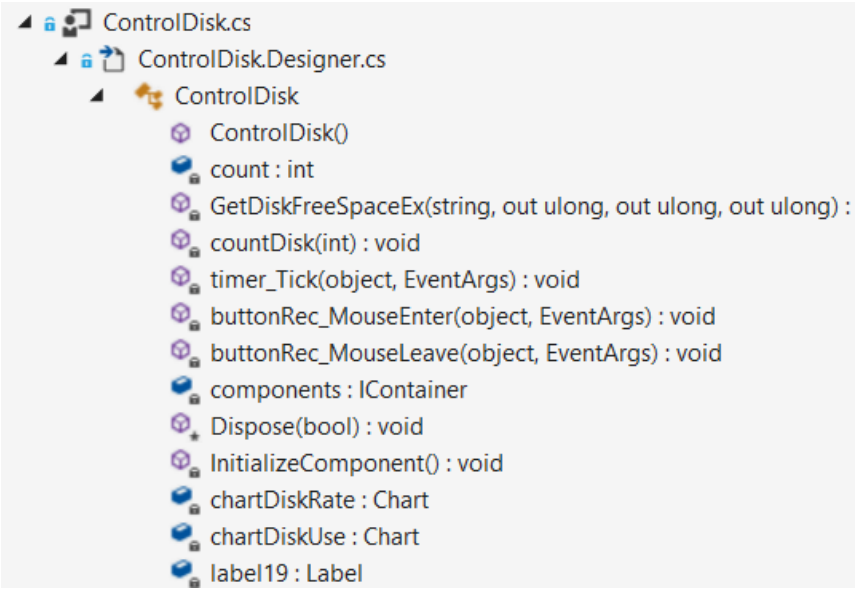


Рисунок 3.9 – Клас «ControlDisk»

Функції даного класу:

а) *countDisk()* – розрахунок використання диска;

б) *timer_Tick()* – запуск лічильника;

в) *buttonRec_MouseEnter()* – відображення рекомендацій;

г) *buttonRec_MouseLeave()* – приховання рекомендацій.

На контролері «ControlNetwork» розташована інформація про інтернет-мережу, вивід інтернет-трафіка та його графіки, а також надання рекомендацій для стабільної роботи інтернет-конференцій в умовах багатозадачності. Структура класу «ControlNetwork» зображено на рис. 3.10.

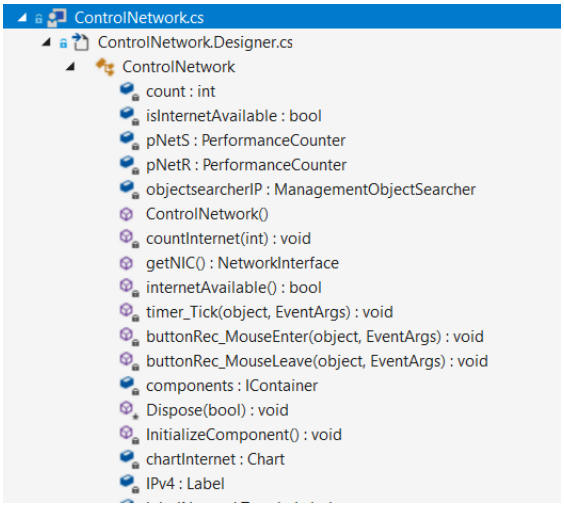


Рисунок 3.10 – Клас «ControlNetwork»

Функції даного класу:

а) *countInternet()* – розрахунок інтерней-трафіку;

б) *getNic()* – отримання перелік усіх мережевих інтерфейсів;

в) *internetAvailabl()* – визначення підключення до інтернету;

г) *timer_Tick()* – запуск лічильника;

д) *buttonRec_MouseEnter()* – відображення рекомендацій;

е) *buttonRec_MouseLeave()* – приховання рекомендацій.

На контролері «ControlInformation» розташована інформація про апаратне забезпечення комп’ютера, а також надання рекомендацій. Структура класу «ControlInformation» зображена на рис. 3.11.

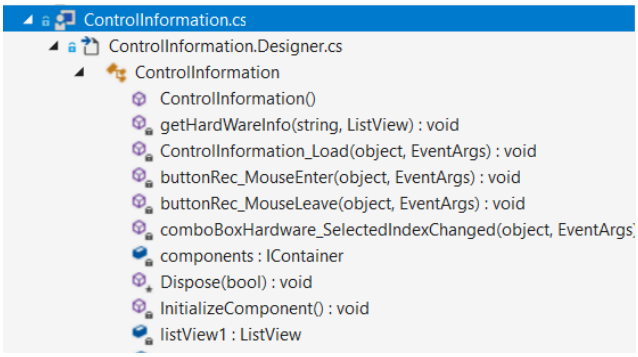


Рисунок 3.11 – Клас «ControlInformation»

Функції даного класу:

а) *GetHardWareInfo()* – отримання інформації про апаратне забезпечення;

б) *ControlInformation_Load()* – відображення інформації при завантаженні форми;

в) *comboBoxHardware_SelectedIndexChanged()* – вибір необхідної інформації;

г) *btnRec_MouseEnter()* – відображення рекомендацій;

д) *btnRec_MouseLeave()* – приховання рекомендацій.

Вікно «Recommendations» містить рекомендації для трьох різних сфер діяльності. Рекомендації призначені для покращення стабільності роботи інтернет-конференцій в умовах багатозадачності. Виділено найбільш популярні програми, які використовують в цих сферах, надано для ПЗ вимоги, щоб користувач міг ознайомитися та забезпечити стабільність роботи. Структура класу «Recommendations» зображено на рис. 3.12.

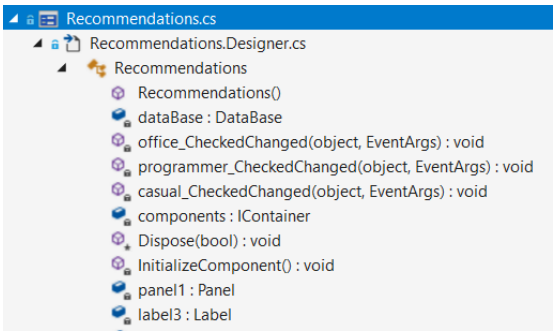


Рисунок 3.12 – Клас «Recommendations»

Функції даного класу:

а) *casual_CheckedChanged()* – рекомендації для звичайного користувача;

б) *office_CheckedChanged()* –рекомендації для офісного працівника;

в) *programmer_CheckedChanged()* – рекомендації для програміста.

Клас «DateBase» містить підключення до бази даних, усі функції, які забезпечують вивід з бази, запис до БД та видалення даних з таблиць. Структура класу «DateBase» зображено на рис. 3.13.

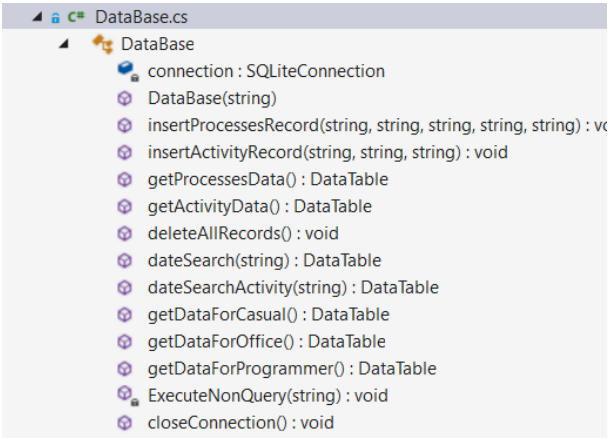


Рисунок 3.13 – Клас «DateBase»

Функції даного класу:

а) *insertProcessesRecord()* – запис даних про активні процеси до БД;

б) *insertActivityRecord()* – запис даних про використання ЦП та ОЗП протягом дня до БД;

в) *getProcessesData()* – отримання даних з БД про активні процеси;

г) *getActivityData()* – отримання даних з БД про використання ЦП та ОЗП;

д) *deleteAllRecords()* – видалення даних з таблиць;

е) *dateSearch()* – пошук по даті запису даних про активні процеси;

ж) *dateSearchActivity()* – пошук по даті запису даних про використання ЦП та ОЗП;

з) *getDataForCasual()* – отримання даних для звичайного користувача;

и) *getDataForOffice()* – отримання даних для офісного працівника;

к) *closeConnection()* – закриття підключення;

л) *ExecuteNonQuery()* – запит до бази даних;

м) *getDataForProgrammer()* – отримання даних для програміста.

Визначено усі функції класів та визначено їх функціонал. В наступному розділі основні функції будуть описані більш детально.

Висновки до розділу 3

Під час написання третього розділу було розроблено моделі програмного забезпечення. Для моделювання процесів роботи ПЗ та подальшої розробки архітектури було побудовано UML-діаграми за допомогою вебресурсу diagrams.net.

Було побудовано 5 наступних діаграм: діаграма прецедентів, діаграма класів, діаграма послідовності, діаграма активностей, діаграма станів. За допомогою різних UML-діаграм спроєктовано модель застосунку для моніторингу працездатності комп'ютера.

На основі отриманих моделей проаналізовано складні процеси, що відбуваються під час роботи застосунку, та всі можливі варіанти поведінки, які можуть відбуватися у рамках змодельованих процесів. Завдяки моделюванню та аналізу продуктивності на основі UML, підвищено ефективність та передбачуваність процесів системи моніторингу стабільності інтернет-конференції в умовах багатозадачності.

Також було визначено структуру застосунку, кількість вікон та визначено основні функції для кожного класу. Було описано базу даних та її структуру таблиць. На основі створених моделей застосунку та його взаємодій з персональним комп'ютером розроблено програмне забезпечення.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

Застосунок призначений для моніторингу працездатності комп'ютера має бути інтуїтивно зрозумілий, зручний у використанні та інформативним.

4.1 Опис функцій розробленого застосунку для моніторингу

Код програми наведено в додатку А, нижче описано основні функції застосунку та інтерфейс.

4.1.1 Опис функцій застосунку

Використання процесора та відображення інформації про процесор виконується за допомогою функції *countCPU()*. Фрагмент коду зображено на рис. 4.1. В даній функції отримуються поточні значення використання ЦП, дескрипторів та потоків. Оновлюються та додаються нові значення використання процесора до графіка. Також функція отримує назву процесора з реєстру системи Windows, отримує значення швидкості процесора, отримує загальну кількість логічних процесорів та обчислює кількість ядер. Ця функія ще має граничне значення використання процесора, у разі досягненння – виводится повідомлення про надмірне навантаження на процесор.

```
labelCPU.Text = string.Format("{0:0.00}%", cpu);
labelHandle.Text = string.Format("{0:0}", handle);
labelProcess.Text = string.Format("{0:0}", process);

if (count >= 100)
    chartCPU.Series["CPU"].Points.RemoveAt(0);
chartCPU.Series["CPU"].Points.AddY(cpu);
object result = Registry.GetValue("HKEY_LOCAL_MACHINE\\HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0", "ProcessorNameString", "");
if (result != null)
{
    labelNameCPU.Text = result.ToString();
}
object result2 = Registry.GetValue("HKEY_LOCAL_MACHINE\\HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0", "~MHz", "");
if (result2 != null)
{
    BasicSpeed.Text = "Швидкість: " + result2 + " ГГц".ToString();
}
```

Рисунок 4.1 – Фрагмент функції *countCPU()*

Вимірювання та відображення даних про використання диска і швидкість читання/запису виконується за допомогою функції *countDisk()*. Фрагмент коду зображено на рис. 4.2. Дана функція забезпечує отримання значення використання диска, швидкості читання та швидкості запису. Оновлюються та додаються нові значення до відповідних графіків. Отримує інформацію про диски, що доступні в системі, а також їх розмір.

```
private void countDisk(int count)
{
    float pdisk = pDISK.NextValue();
    float dRead = pReadSpd.NextValue();
    float dWrite = pWriteSpd.NextValue();

    labelDiskUse.Text = string.Format("{0:0}%", pdisk);
    labelReadSpd.Text = string.Format("{0:0.0} КБ/с", dRead / 1024);
    labelWriteSpd.Text = string.Format("{0:0.0} КБ/с", dWrite / 1024);

    if (count >= 100)
    {
        chartDiskUse.Series["DiskUse"].Points.RemoveAt(0);
        chartDiskRate.Series["DiskRead"].Points.RemoveAt(0);
        chartDiskRate.Series["DiskWrite"].Points.RemoveAt(0);
    }
    chartDiskUse.Series["DiskUse"].Points.AddY(pdisk);
    chartDiskRate.Series["DiskRead"].Points.AddY(dRead / 1024);
    chartDiskRate.Series["DiskWrite"].Points.AddY(dWrite / 1024);

    DriveInfo[] allDrives = DriveInfo.GetDrives();
```

Рисунок 4.2 – Фрагмент функції *countDisk()*

Вимірювання та відображення швидкості передачі та отримання даних, а також статус підключення до Інтернету виконується за допомогою функції *countInternet()*. Фрагмент коду зображено на рис. 4.3. Дана функція отримує значення відправки та отримання (байтів на секунду); відображає швидкості відправки та отримання (кбіт/с). Оновлюються та додаються нові значення до графіка. Виводить інформацію про мережевий інтерфейс. Перевіряє підключення до інтернету та виводить необхідне повідомлення.

```
pNetS.CategoryName = "Network Interface";
pNetS.CounterName = "Bytes Sent/sec";
pNetS.InstanceName = name;
pNetR.CategoryName = "Network Interface";
pNetR.CounterName = "Bytes Received/sec";
pNetR.InstanceName = name;
float fsend = pNetS.NextValue();
float freceive = pNetR.NextValue();

labelNetS.Text = string.Format("{0:0.0} Кбіт/с", fsend * 8 / 1024);
labelNetR.Text = string.Format("{0:0.0} Кбіт/с", freceive * 8 / 1024);

if (count >= 100)
{
    chartInternet.Series["Send"].Points.RemoveAt(0);
    chartInternet.Series["Receive"].Points.RemoveAt(0);
}
chartInternet.Series["Send"].Points.AddY(fsend * 8 / 1024);
chartInternet.Series["Receive"].Points.AddY(freceive * 8 / 1024);
```

Рисунок 4.3 – Фрагмент функції *countInternet()*

Вимірювання та відображення даних про використання оперативної пам'яті виконується за допомогою функції *countRam()*. Фрагмент коду зображено на рис. 4.4. Дана функція отримує та виводить значення пам’яті, що використовується, доступна, кешована та виконана, а також значення вивантаженого та невивантаженого пула. Оновлюються та додаються нові значення використання оперативної пам’яті до графіка.

```
private void countRam(int count)
{
    ComputerInfo cpI = new ComputerInfo();
    float ram = pRAM.NextValue();
    float ramCmt = pRAMcmt.NextValue();
    float ramAvai = pRAMavai.NextValue();
    float cached = pCached.NextValue();
    float paged = pPaged.NextValue();
    float nPaged = pNPaged.NextValue();


    labelRAM.Text = string.Format("{0:0.0} ГБ", (cpI.TotalPhysicalMemory - cpI.AvailablePhysicalMemory) / (1024 * 1024 * 1024));
    labelCmt.Text = string.Format("{0:0.0} ГБ", ramCmt / (1024 * 1024 * 1024));
    labelRamAvai.Text = string.Format("{0:0.0} ГБ", ramAvai / 1024);
    labelCached.Text = string.Format("{0:0.0} ГБ", cached / (1024 * 1024 * 1024));
    labelPaged.Text = string.Format("{0:0} МБ", paged / (1024 * 1024));
    labelNPaged.Text = string.Format("{0:0} МБ" +
        "", nPaged / (1024 * 1024));

    if (count >= 100)
        chartRAM.Series["RAM"].Points.RemoveAt(0);
    chartRAM.Series["RAM"].Points.AddY(ram);
```

Рисунок 4.4 – Фрагмент функції *countRam()*

Вимірювання та відображення даних про активні процеси та використання процесора та пам’яті протягом дня виконується за допомогою функції *counTask()*. Фрагмент коду зображено на рис. 4.5. Дана функція визначає активні процеси, їх використання пам’яті, статус процесу, дату запуску та дату запису даних. Всі ці дані записуються до бази даних, після чого їх можна переглянути в будь-який момент. Також функція визначає використання процесора та пам’яті загальний протягом дня, створюється запис з датою та використання параметрів, після цього дані записуються до БД. Таким чином можна переглянути записані дані, зробити аналіз та забезпечити стабільність роботи конференцій та системи.

```
private void countTask(int count)
{
    ComputerInfo cpI = new ComputerInfo();
    float cpu = CPU.NextValue();

    Process[] processes = Process.GetProcesses();

    ListViewItem item2 = new ListViewItem(string.Format("{0:0.00}%", cpu));
    item2.SubItems.Add(string.Format("{0:0.0} GB", (cpI.TotalPhysicalMemory - cpI.AvailablePhysicalMemory) / (1024 * 1024 * 1024)));
    item2.SubItems.Add(DateTime.Now.ToString());
    listViewProcesses2.Items.Add(item2);
    dataBase.insertActivityRecord(string.Format("{0:0.00}%", cpu),
                          string.Format("{0:0.0} GB", (cpI.TotalPhysicalMemory - cpI.AvailablePhysicalMemory) / (1024 * 1024 * 1024)),
                          DateTime.Now.ToString());
```

Рисунок 4.5 – Фрагмент функції *coutTask()*

Визначено та описано основні функції, які розраховують параметри, виводять інформацію, різні значення та графіки у реальному часі, а також зберігають дані у БД, це дає можливістю перегляду у будь-який час, щоб проаналізувати ресурсоспоживання ПЗ та забезпечити стабільність роботи.

4.1.2 Опис інтерфейсу застосунку

На рис. 4.1 зображено головне вікно. На цьому вікні виводяться активні процеси, їх використання пам’яті, використання ЦП та дискового простору протягом дня. На даному вікні є кнопка, яка дає можливість завантажити усі додані дані з БД до таблиць (рис. 4.2), також передбачено можливість пошуку даних за датою запису для того, щоб легше було переглядати дані та аналізувати їх.

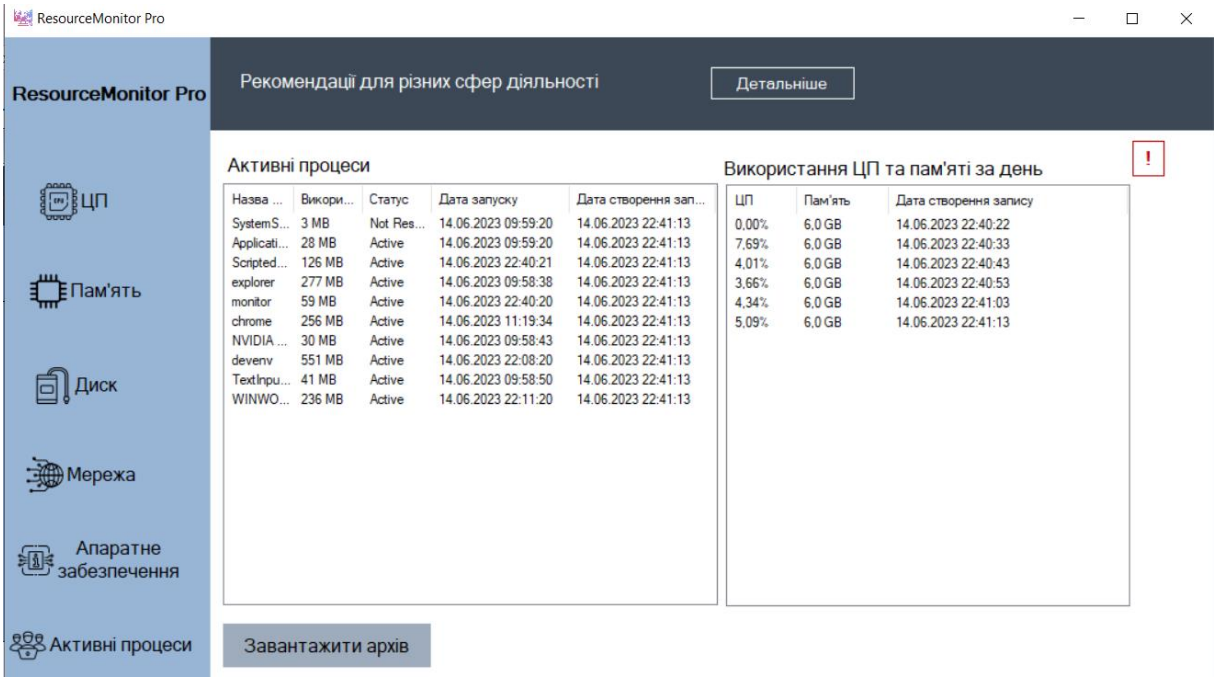


Рисунок 4.1 – Головне вікно застосунку

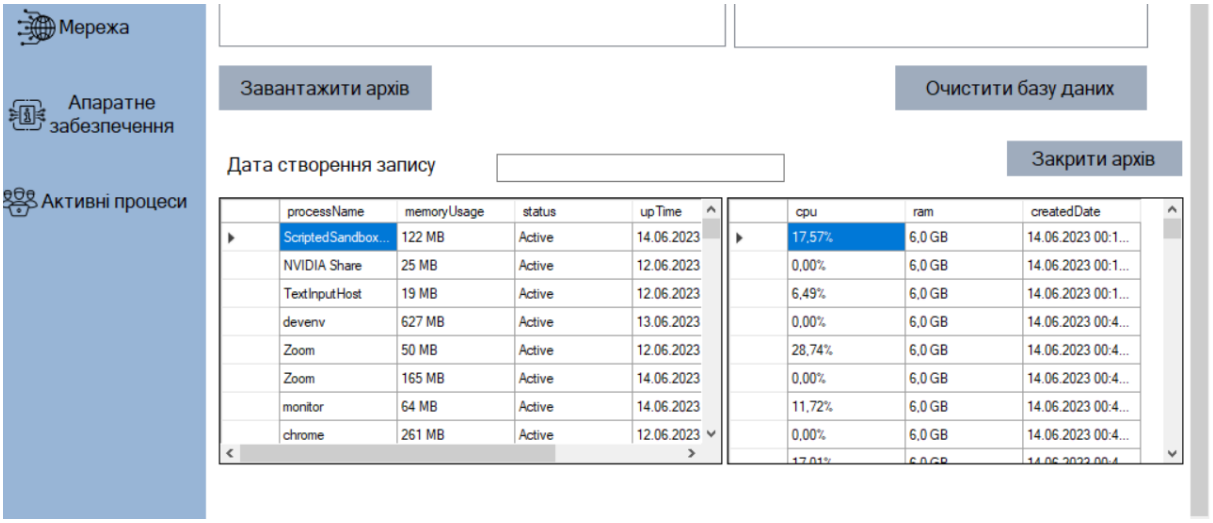


Рисунок 4.2 – Завантаження даних з БД на головному вікні

На рис. 4.3 зображено вікно з інформацією про використання процесора та дані про процесор.

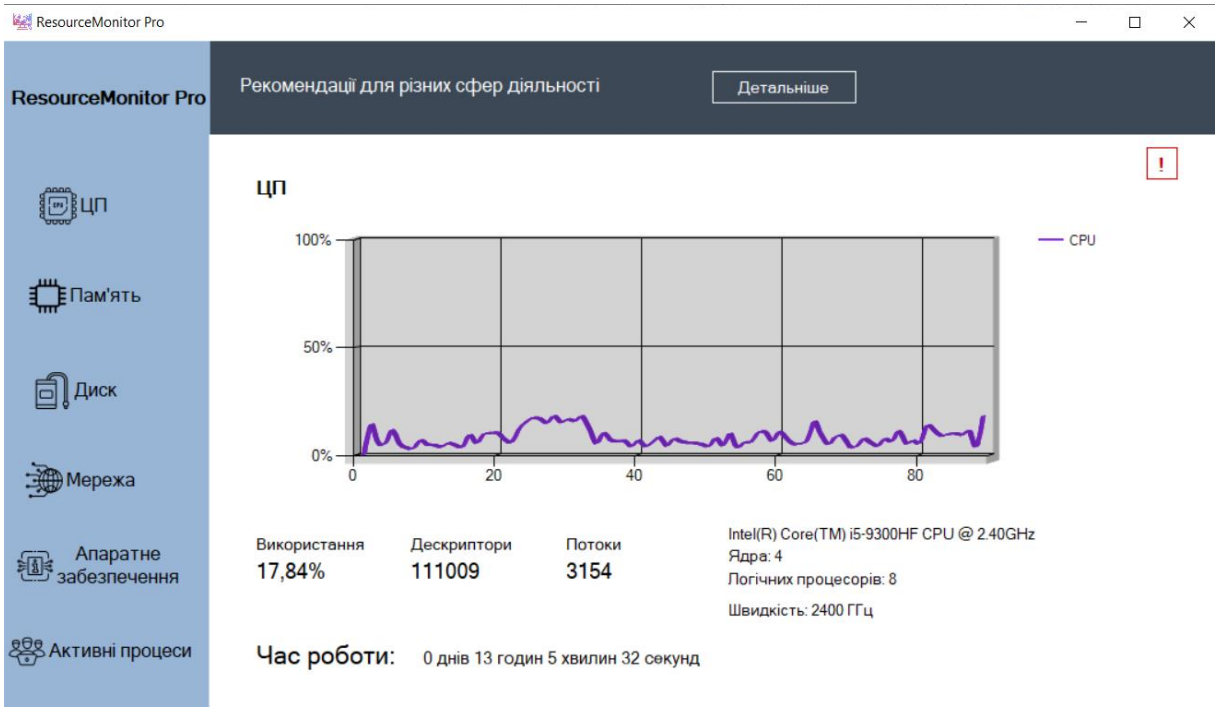


Рисунок 4.3 – Вікно «ЦП»

На рис. 4.4 зображено вікно з інформацією про використання пам’яті.

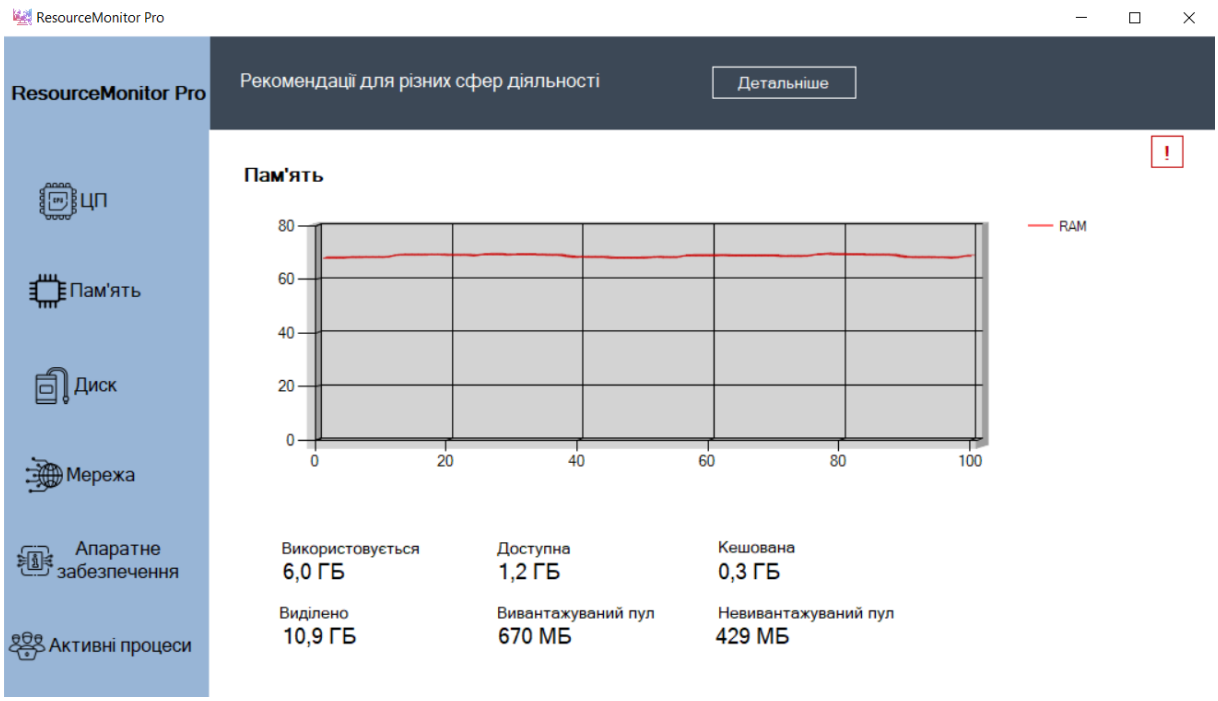


Рисунок 4.4 – Вікно «Ram information»

На рис. 4.5 зображено вікно з інформацією про використання диска та швидкість обміну даними з диском, а також інформація про наявні диски.

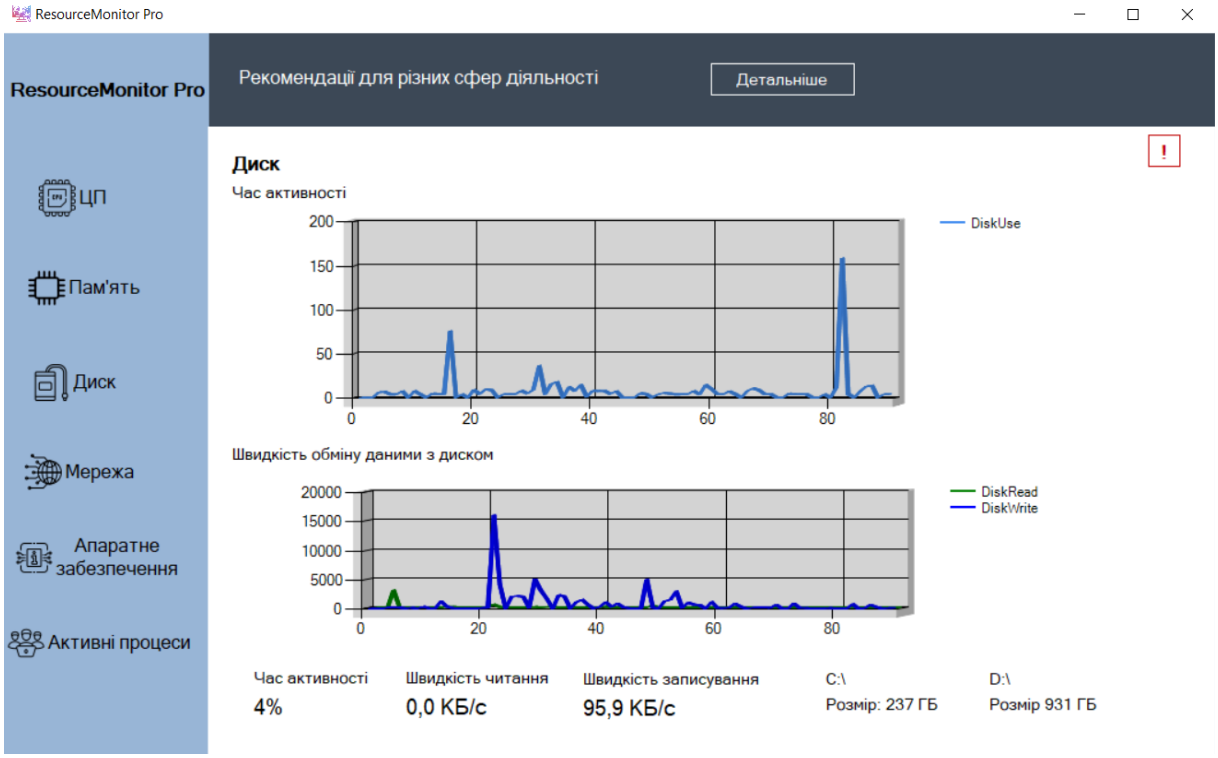


Рисунок 4.5 – Вікно «Disk information»

На рис. 4.6 зображено вікно з інформацією про мережу, швидкість відправки та отримання даних з інтернету.

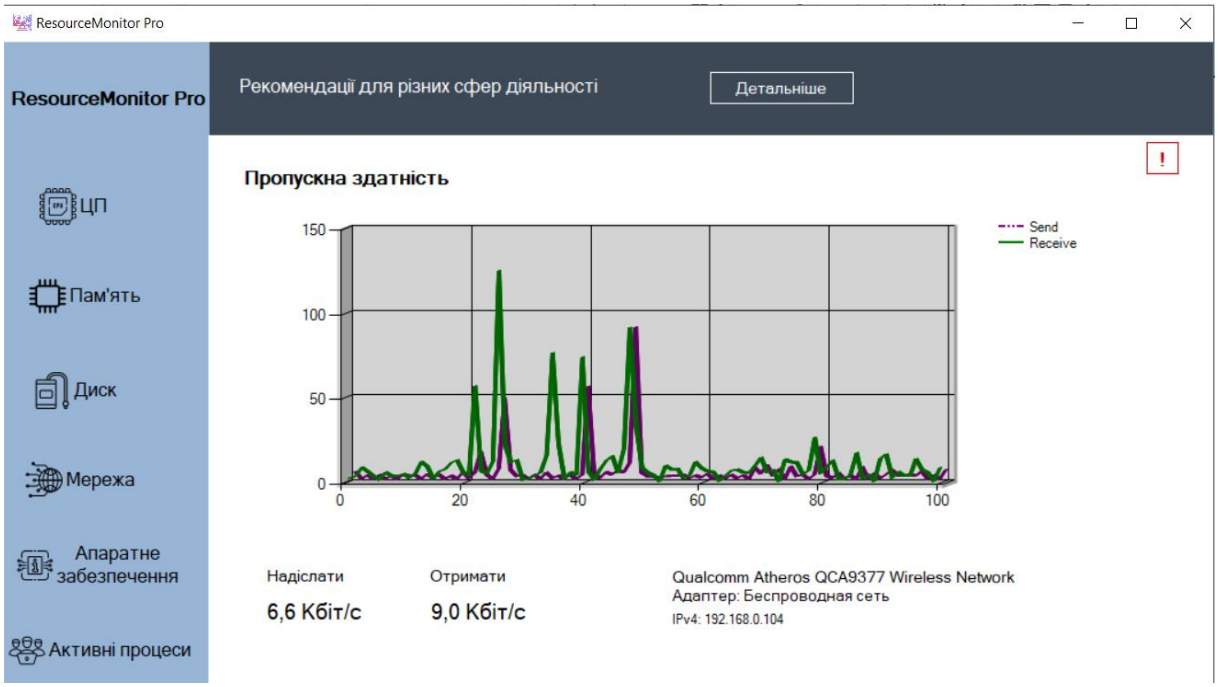


Рисунок 4.6 – Вікно «Network information»

На рис. 4.7 зображено вікно з інформацією про апаратне забезпечення.

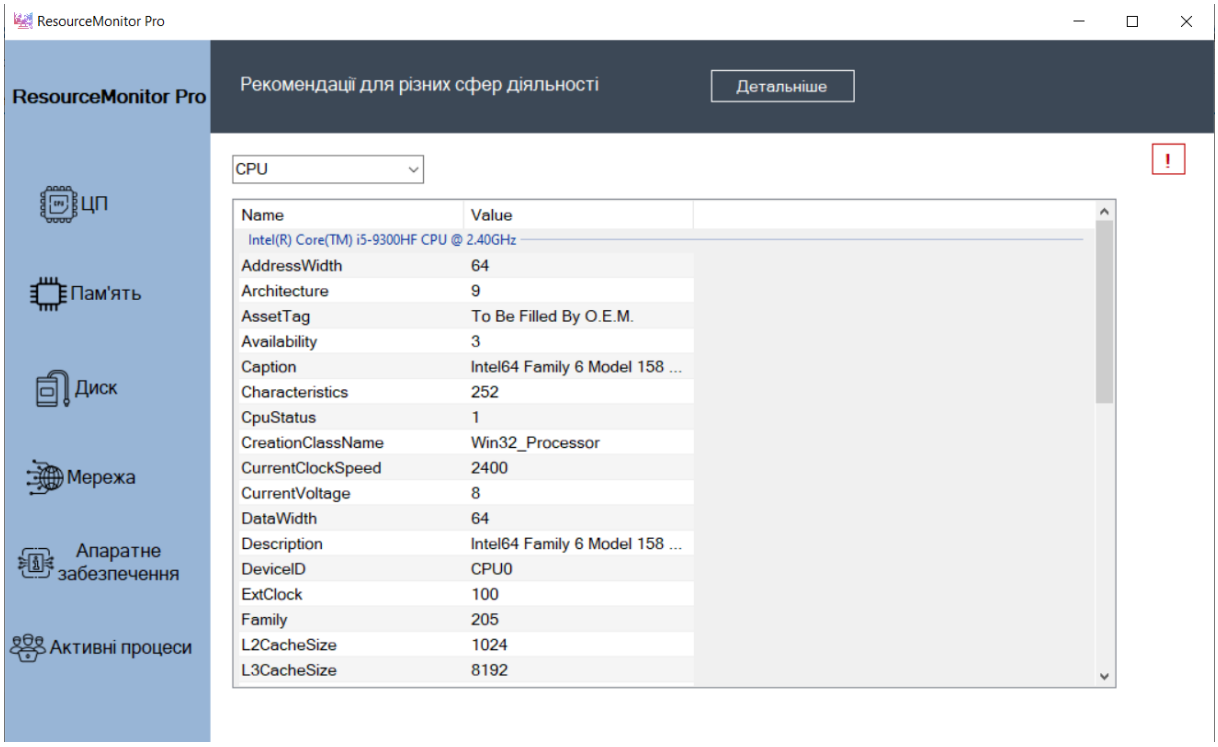


Рисунок 4.7 – Вікно «Hardware information»

Кожне вікно містить рекомендації для користувача для забезпечення стабільності роботи інтернет-конференцій. На рис. 4.8 зображено вивід рекомендації щодо активних процесів, використання ЦП та пам’яті за день.

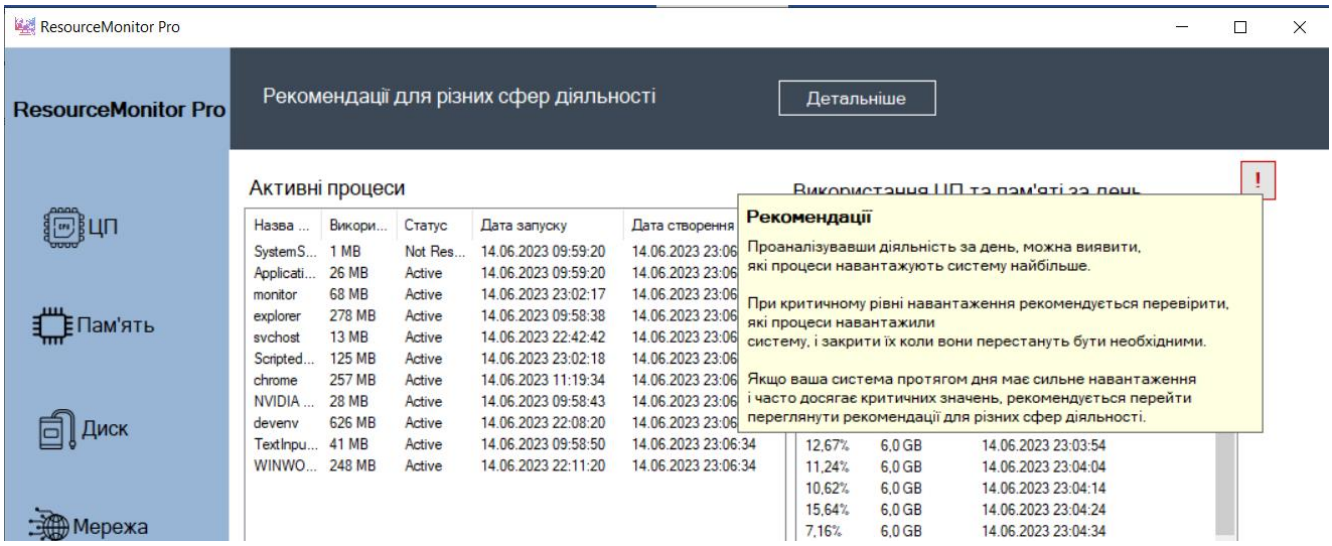


Рисунок 4.8 – Вивід рекомендацій на головному вікні

На рис. 4.9 зображено рекомендації щодо використання ЦП.

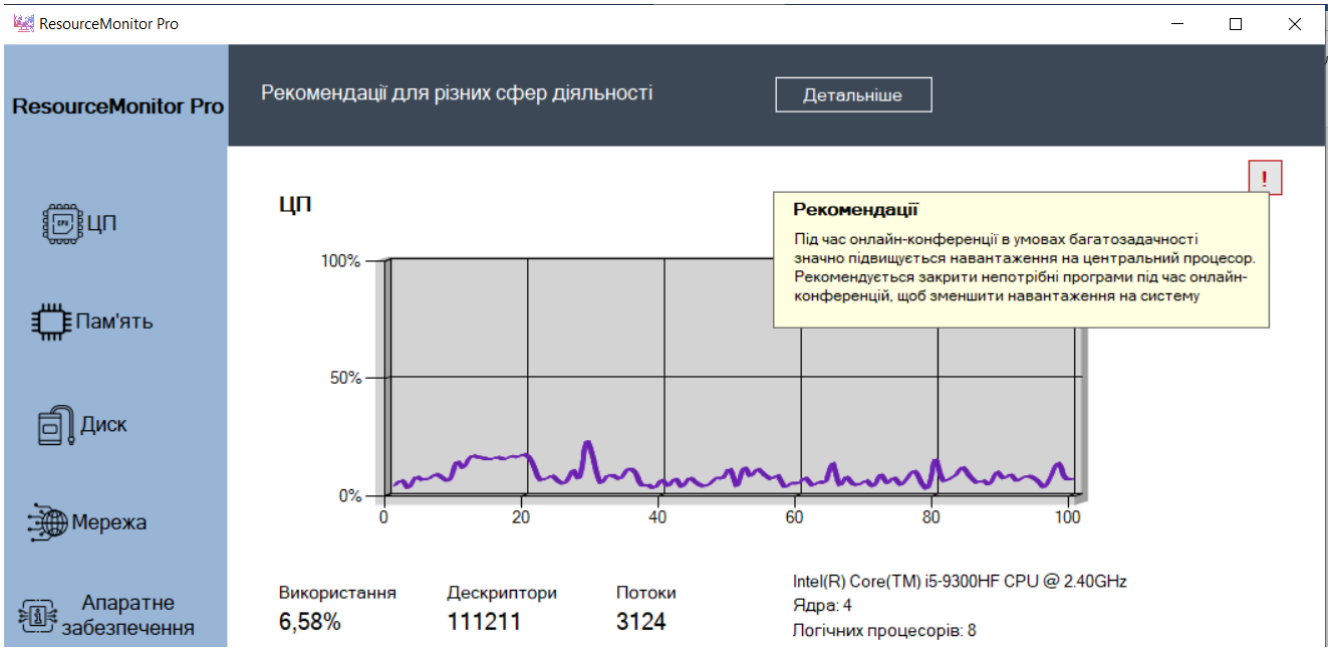


Рисунок 4.9 – Вивід рекомендацій щодо ЦП

На рис. 4.10 зображено рекомендацій щодо використання пам’яті.

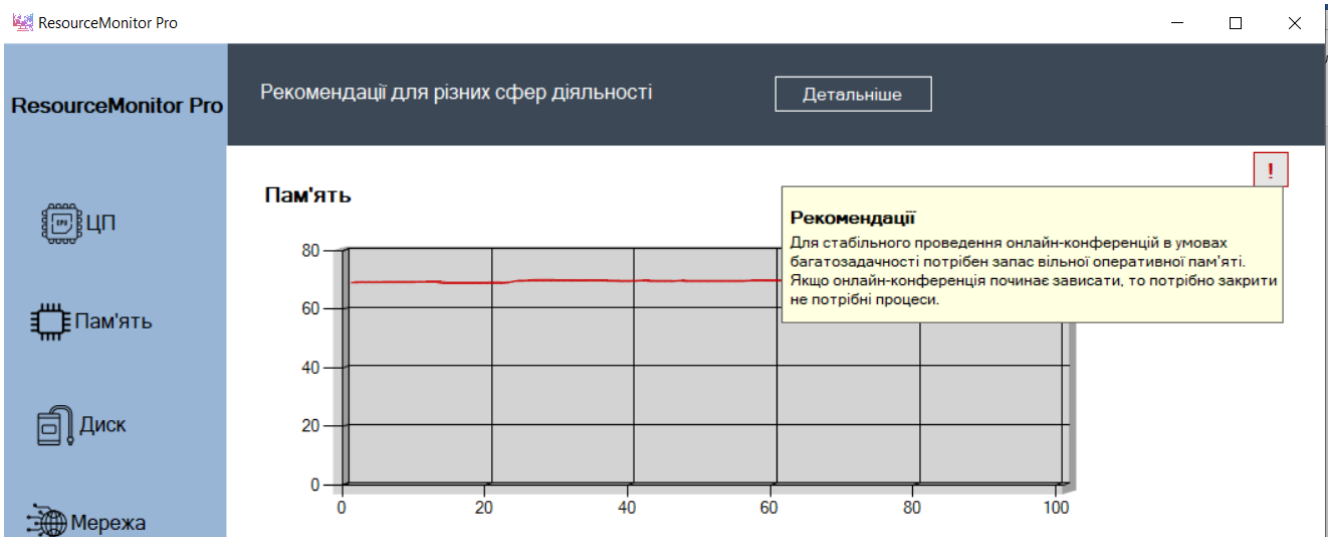


Рисунок 4.10 – Вивід рекомендацій щодо пам’яті

На рис. 4.11 зображено рекомендацій щодо використання диска.

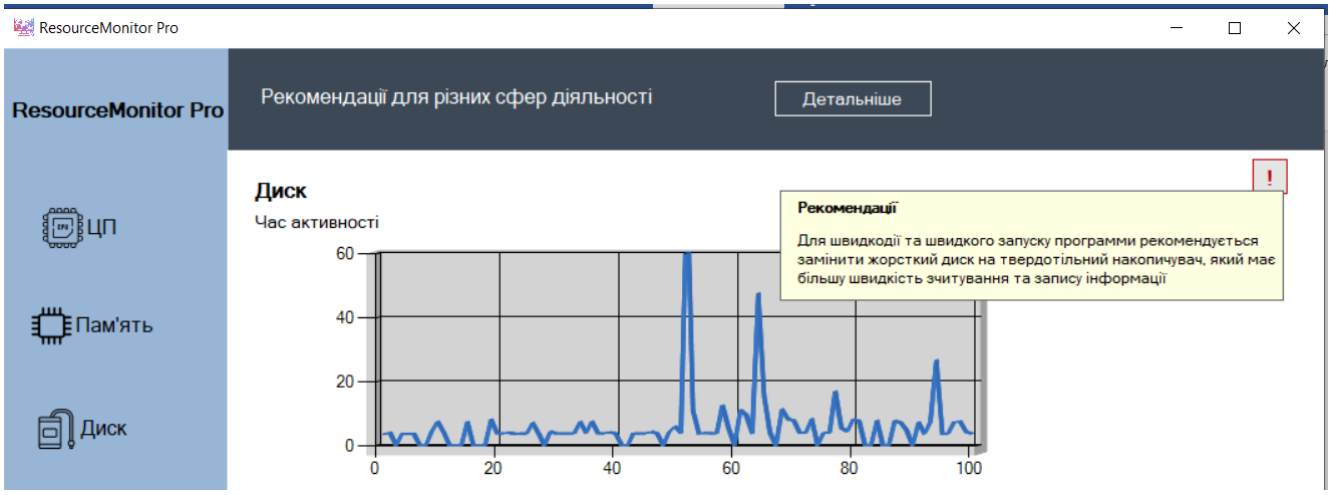


Рисунок 4.11 – Вивід рекомендацій щодо диска

На рис. 4.12 зображено появлення push-рекомендацій щодо інтернет-мережі.

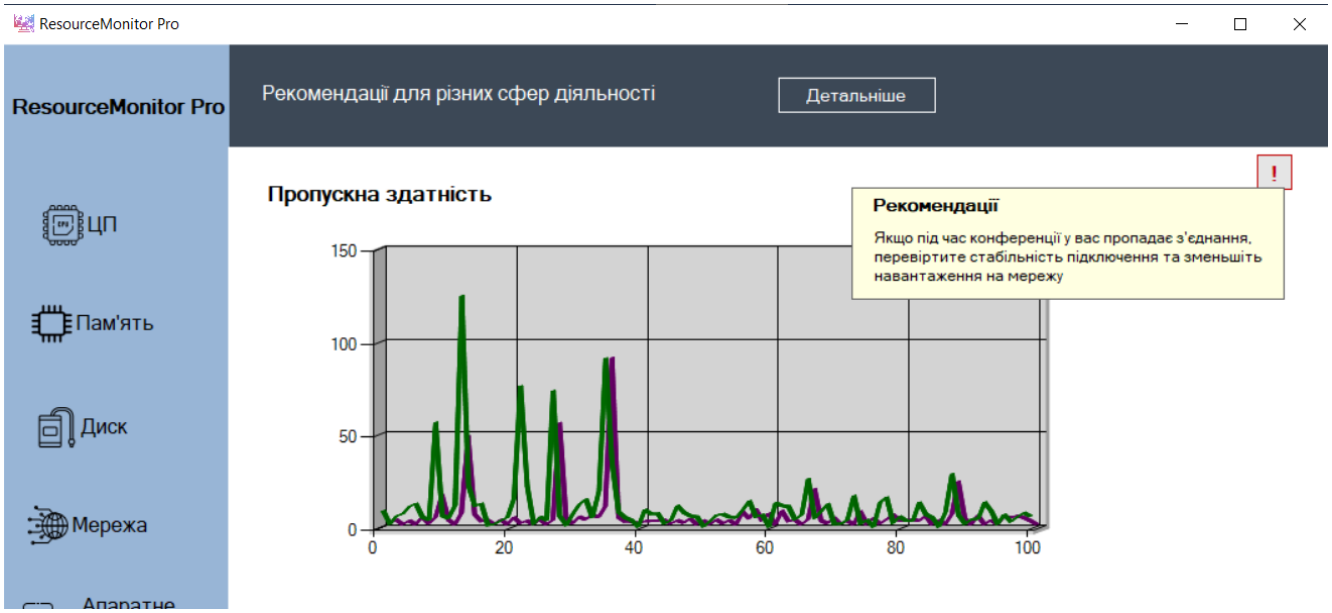


Рисунок 4.12 – Вивід рекомендацій щодо мережі

На рис. 4.13 зображено появлення push-рекомендацій щодо апаратного забезпечення.

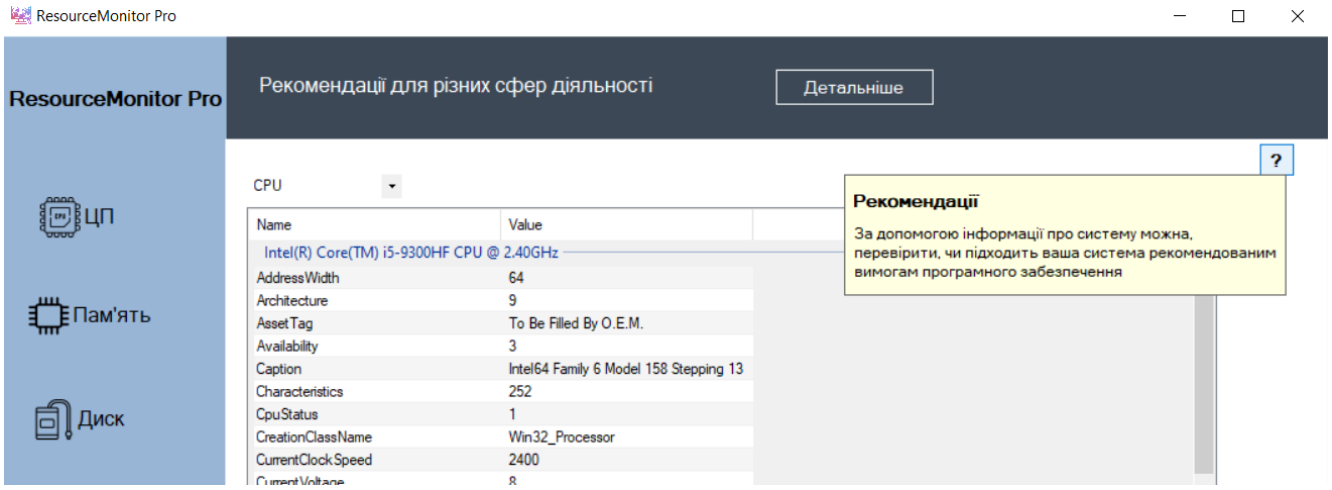


Рисунок 4.13 – Вивід рекомендацій щодо апаратного забезпечення

На рис. 4.14 зображено вікно з рекомендаціями для різних сфер діяльності. На даному вікні містяться вимоги до ПЗ для різних користувачів та рекомендації, які допоможуть забезпечити стабільну роботу інтернет-конференції.

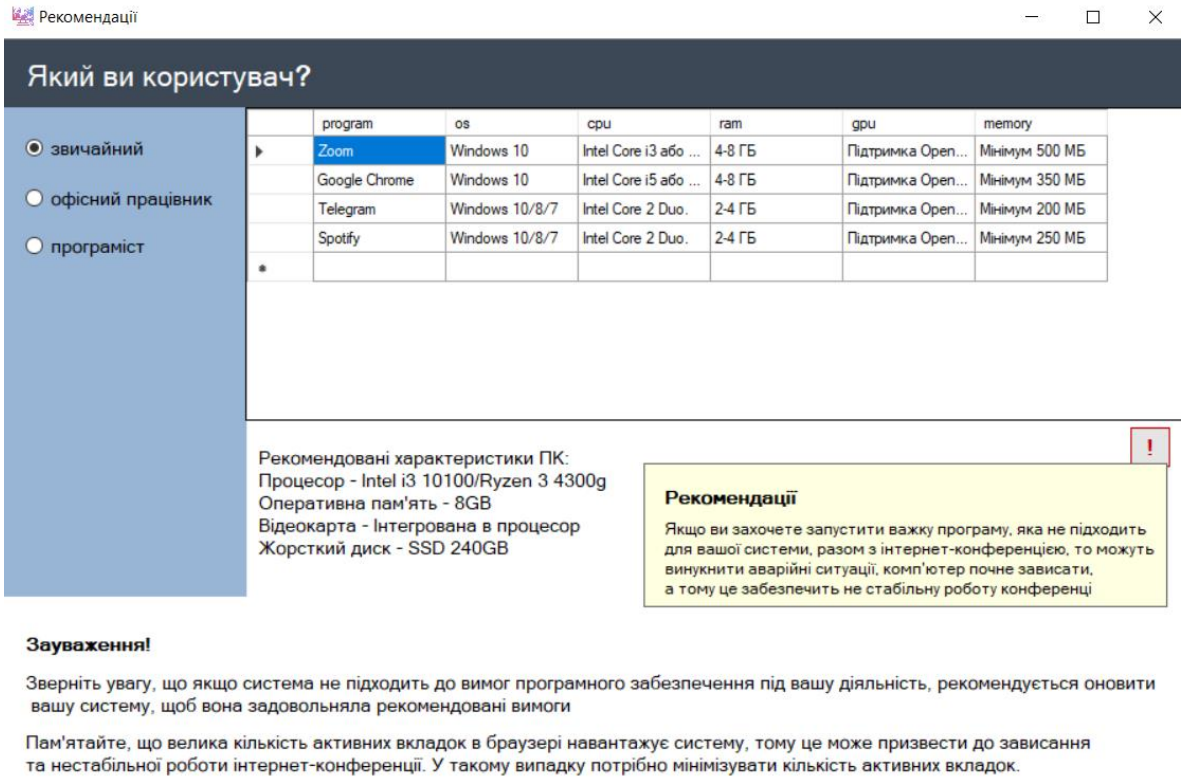


Рисунок 4.14 – Вікно з рекомендаціями для різних користувачів

На рис. 4.15 зображено вивід повідомлення при використання процесора на 100 %.

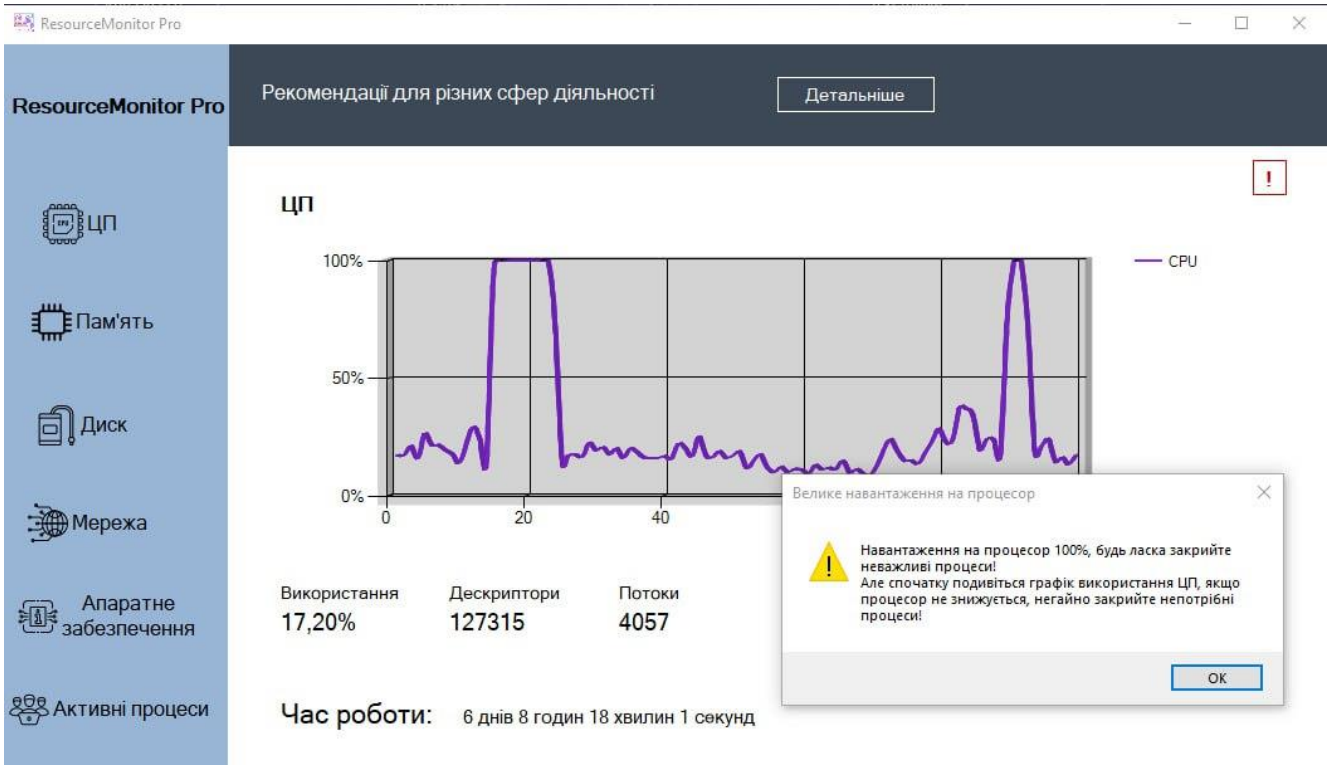


Рисунок 4.15 – Повідомлення під час використання процесора на 100 %

На рис. 4.16 зображено вивід повідомлення при відключенні інтернету

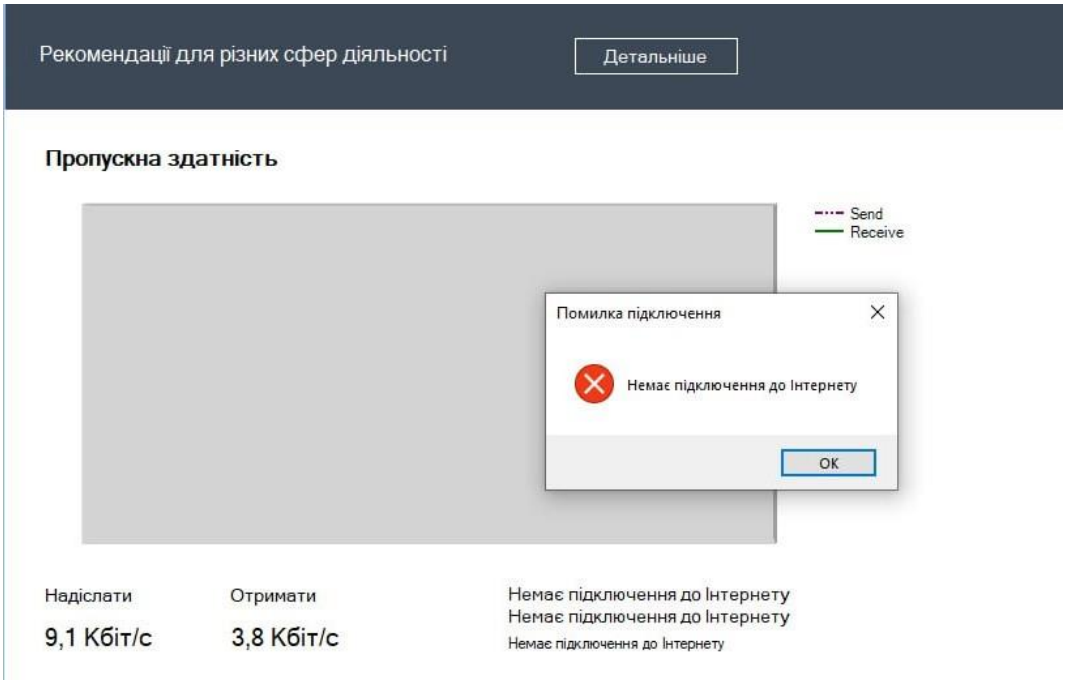


Рисунок 4.16 – Повідомлення при відсутності підключення до інтернету

На рис. 4.17 зображено вивід повідомлення при підключенню до інтернету.

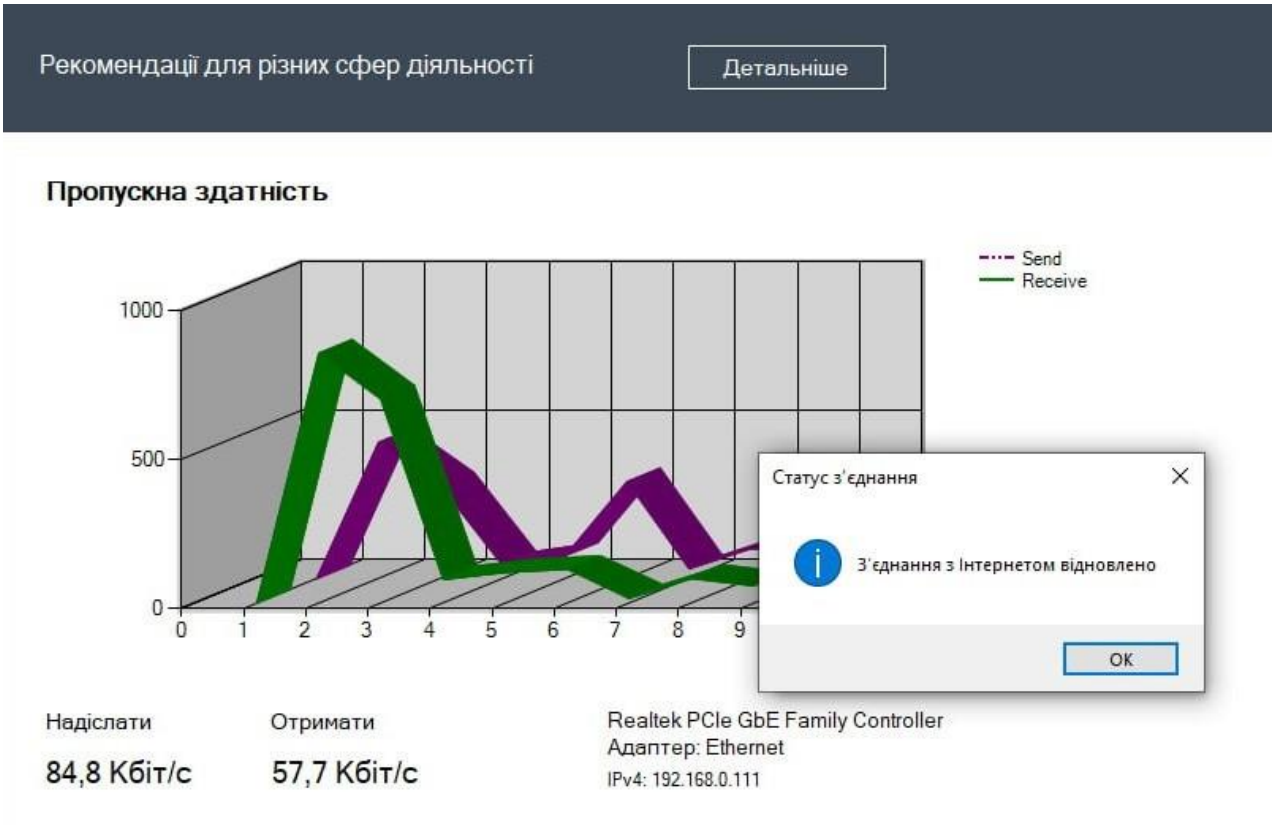


Рисунок 4.17 – Повідомлення при підключенню до інтернету

4.2 Тестування розробленого застосунку-моніторингу стабільності інтернет-конференції в умовах багатозадачності

Проведемо тестування розробленого застосунку для моніторингу стабільності інтернет-конференції в умовах багатозадачності. Для цього оберемо конференцію Zoom, різні ПЗ та браузер, щоб перевірити навантаженння на систему та моніторинг стабільності роботи конференції.

Запустимо Zoom, Visual Studio, Chrome та Word. У Chrome відкриємо декілька вкладок. Відкриваємо розроблений застосунок та переглядаємо навантаження та використання ресурсів.

На рис. 4.18 зображено навантаження на процесор при роботі усіх ПЗ.

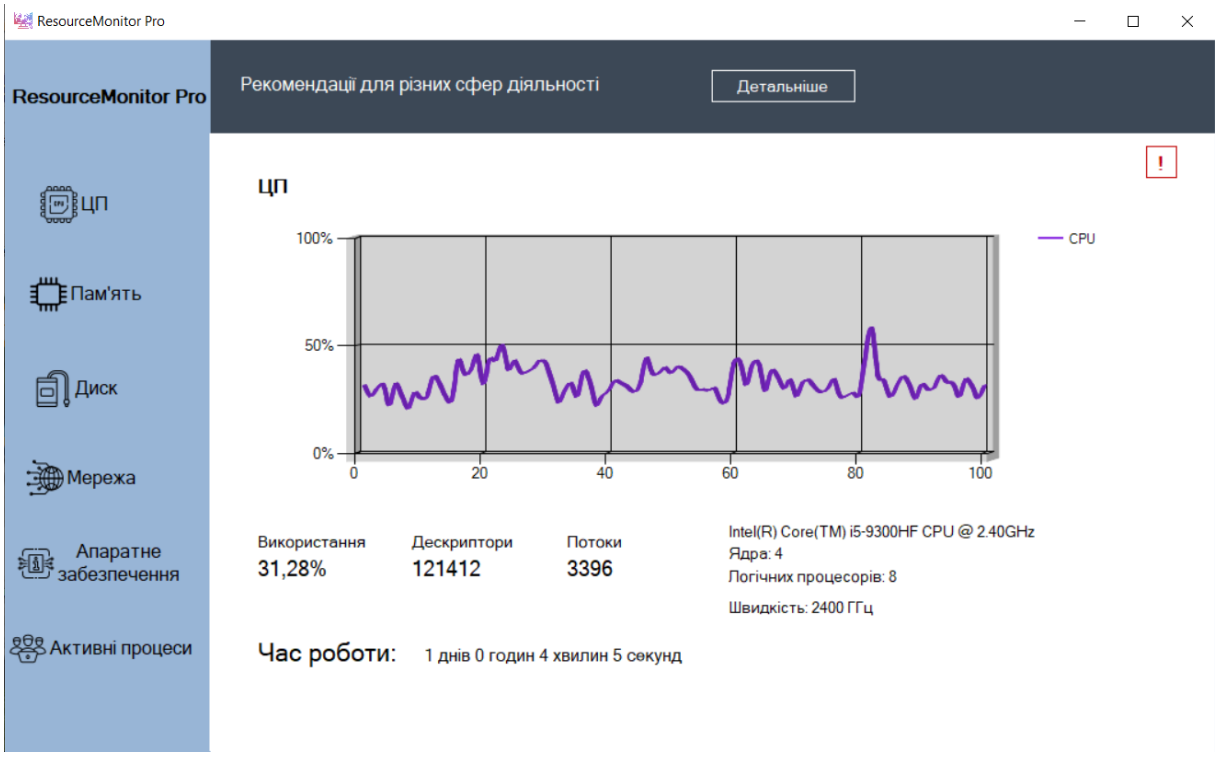


Рисунок 4.18 – Навантаження на процесор

На рис. 4.19 зображено навантаження на пам’ять в умові багатозадачності комп’ютера.

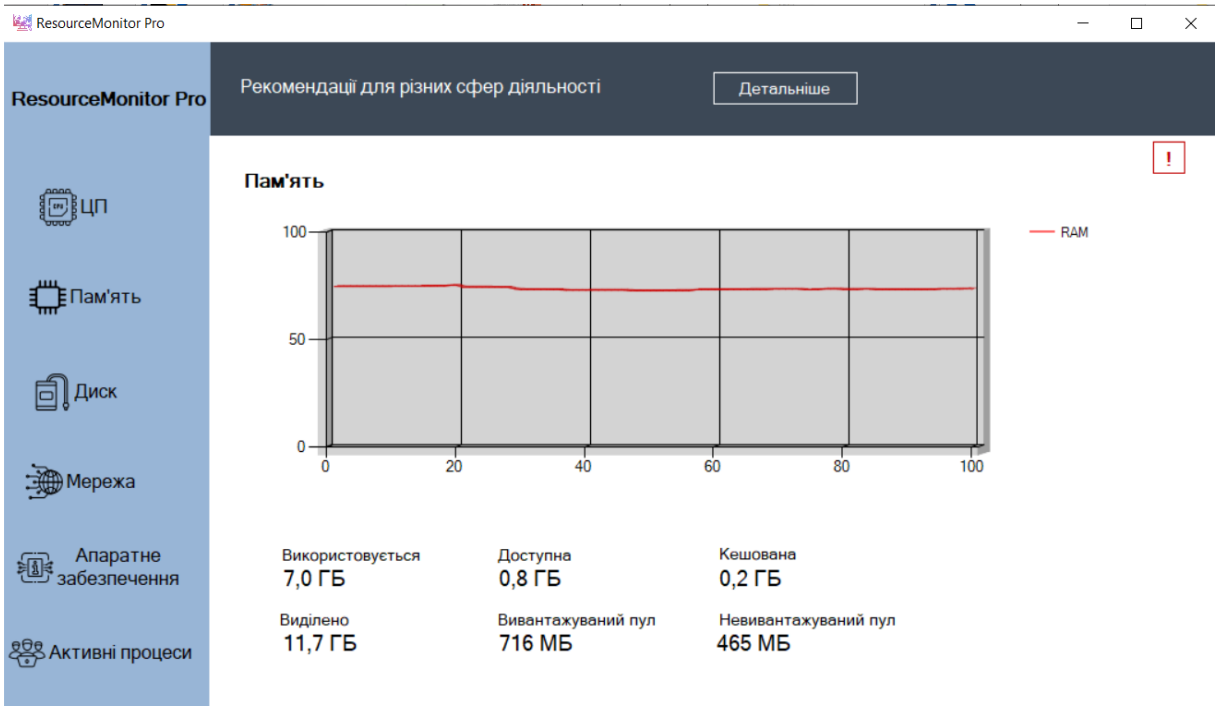


Рисунок 4.19 – Навантаження на пам’ять

На рис. 4.20 зображено швидкість відправки та отримання інтернету при роботі конференції в умовах багатозадачності.

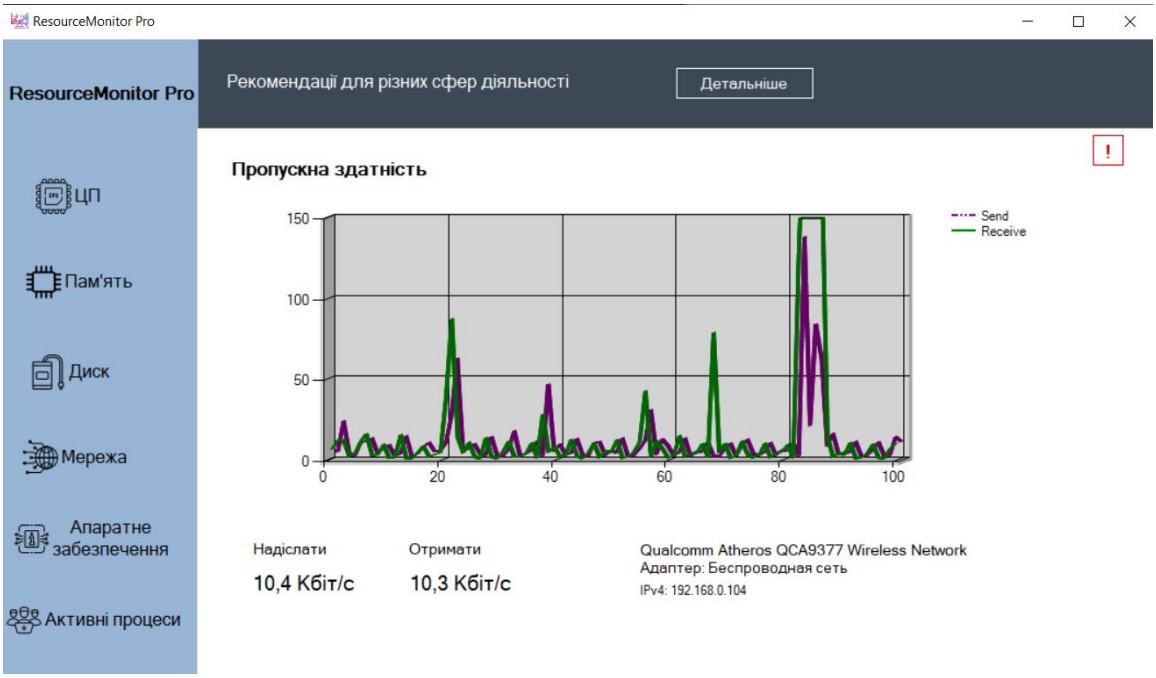


Рисунок 4.20 – Швидкість відправки та отримання інтернету

На рис. 4.21 зображено активні процеси та загальне навантаження на пам’ять та процесор.

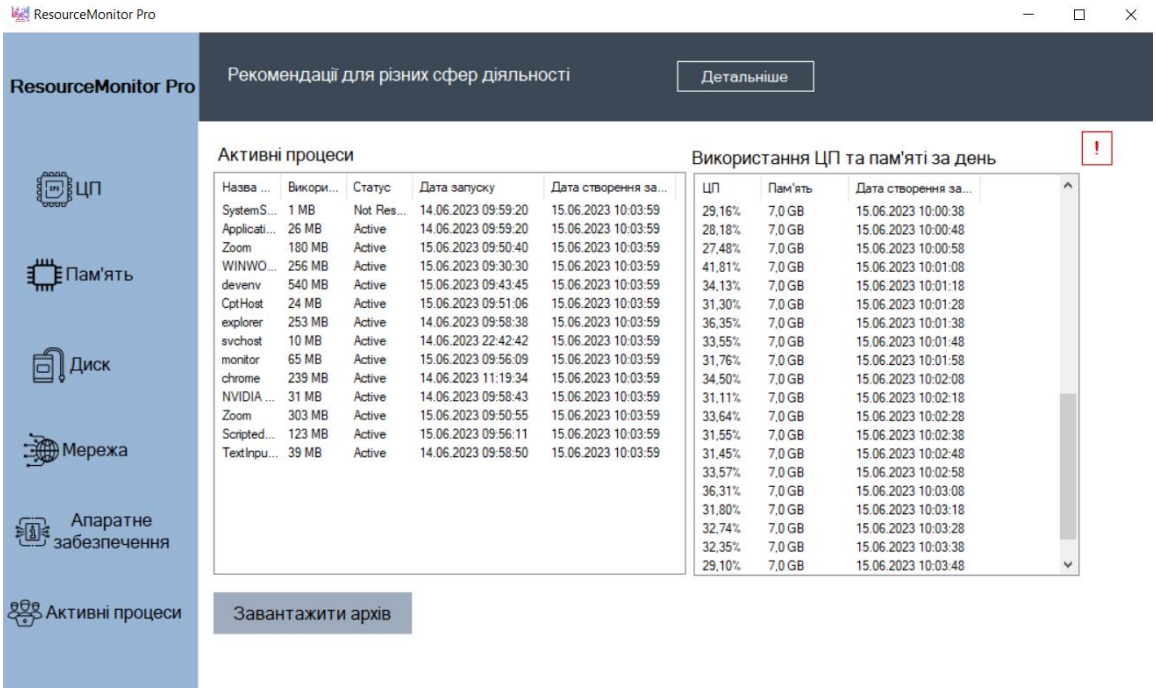


Рисунок 4.21 – Активні процеси та навнтаження на систему

Дане тестування було проведено на ноутбуці з процесором Intel Core i5-9300HF, 8 Гбайт оперативної пам’яті, з використанням бездротового інтернету. Згідно графікам бачимо, що максимальне значення до якого доходило значення процесора 52 %, пам’ять 6 Гбайт, швидкість відправки відправки повідомлень до інтернету доходило до 150 кбіт/с.

При даному аналізу інтернет-конференція мала стабільну роботу, система не була сильно навантажена, адже усі ПЗ підходять за вимогами для даного ноутбука, щоб стабільно працювала система. З даними вимогами та рекомендаціями можна ознайомитися у вікні з рекомендаціями для різних сфер діяльності, а також можна переглянути інформацію про апаратне забезпечення комп’ютера для того, щоб краще знати параметри свого пристроя, щоб забезпечити стабільність роботи конференції.

Важливо зазначити, що якщо проводити це тестуванні на ноутбуці, який має нижчі характеристи, тоді могли виникнути аварійні ситуації такі як, перевантаження процесора та пам’яті, в такому випадку це привело до зависання та нестабільної роботи інтерне-конференції, а розроблений застосунок проаналізував би навантаження та повідомив користувачу про необхідність знизити навантаження на систему, закрити неважлиів процеси, щоб забезпечити стабільність інтернет-конференції. Для користувача забезпечено різні рекомендації на кожному вікні для того, щоб він зміг з ними ознайомитися та скористатися для забезпечення стабільності роботи конференцій.

Проведемо ще одне дослідження, проаналізуємо стабільність роботи конференції при включенню звука та відео, порівняємо використання оперативної пам’яті для конференції, споживання ЦП загальний, споживання пам’яті загальний при кожному типу тестуванню.

Таблиця 4.7 – Ресурсоспоживання системи в умовах багатозадачності

Тип тесту	**Платформа**	**Завантаження ЦП загальний, %**	**Споживання пам'яті загальний, Гбайт**	**Споживання пам'яті, Мбайт**
Мікрофон OFF камера OFF	Google Meet Zoom	23,38 22,17	6,0 6,0	267 298
Мікрофон ON камера OFF	Google Meet Zoom	25,25 28,62	6,0 6,0	267 303
Мікрофон OFF камера ON	Google Meet Zoom	42,22 32,95	6,0 6,0	262 359
Мікрофон ON камера ON	Google Meet Zoom	42,04 48,63	6,0 6,0	263 366
Мікрофон ON камера ON + демонстрація екрану	Google Meet Zoom	48,73 52,73	6,0 6,0	274 449

Як можна побачити з результатів тестування, найбільше навантаження конференція дає тоді, коли увімкнено звук, відео та демонстрацію екрану. Щоб забезпечити стабільну роботу для користувача, який хоче працювати саме в такій конференції в умовах багатозадачності, потрібно подбати про навантаження на пристрій, а також переглянути усі рекомендації, які зазначено у застосунку. У разі аварійних ситуацій буде повідомлено про надмірне навантаження на систему, і щоб забезпечити стабільність необхідно буде закрити непотрібні процеси.

Далі проведемо порівняльний аналіз споживання ресурсів інтернет-конференції на різних пристроях. Для початку визначимо характеристики комп'ютерів, на яких будемо проводити тестування (табл. 4.8).

Таблиця 4.8 – Параметри пристроїв для тестування

	HP 255 G7 Notebook	**Lenovo IdeaPad Gaming 3**	**Персональний комп'ютер**
ОС	Windows 10	Windows 10	Windows 10
Процесор	Ryzen 3 2200U	Intel Core i5-9300HF	Ryzen 5 1600

Кінець таблиці 4.8

ОЗП	8 GB	8 GB	16 GB
Графічний процесор	AMD Radeon RX Vega 3	NVIDIA GTX 1650 4GB	NVIDIA GTX 1650Super 4GB
Накопичувач	SSD 250GB	SSD 250GB + HDD 1TB	SSD 500GB + HDD 1TB
Тип Інтернет-з’єднання	Бездротове	Бездротове	Ethernet

Далі порівняємо навантаження на пристрої при проведенні конференції Zoom при роботі з Visual Studio та Chrome. На рис. 4.22 зображено навантаження на процесор для першого пристрою.

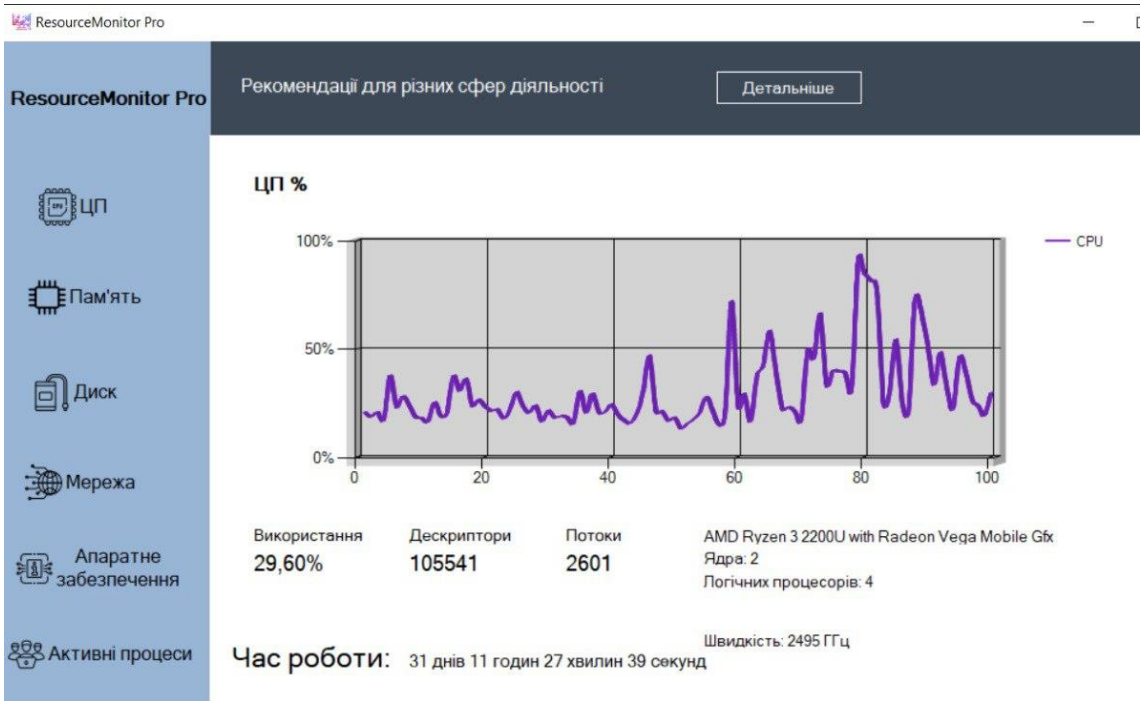


Рисунок 4.22 – Навантаження на процесор на ноутбуці HP

На рис. 4.23 зображено навантаження на пам’ять для третього пристрою.

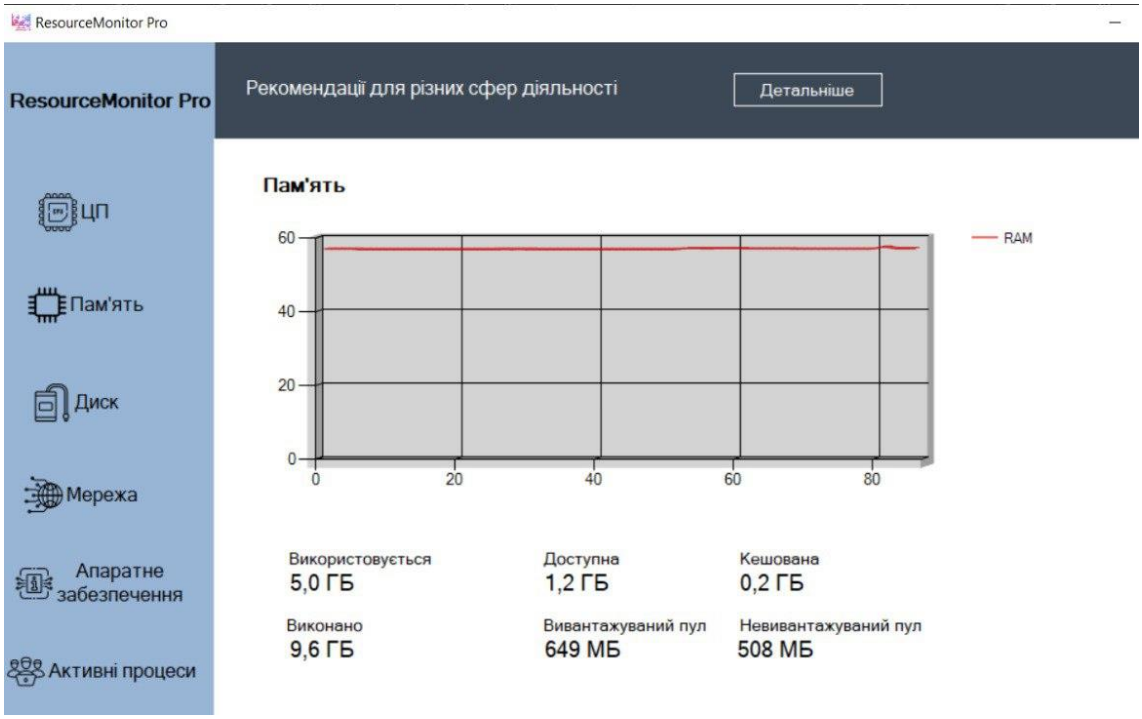


Рисунок 4.23 – Навантаження на пам’ять на ноутбуці HP

На рис. 4.24 зображено навантаження на процесор для другого пристрою.

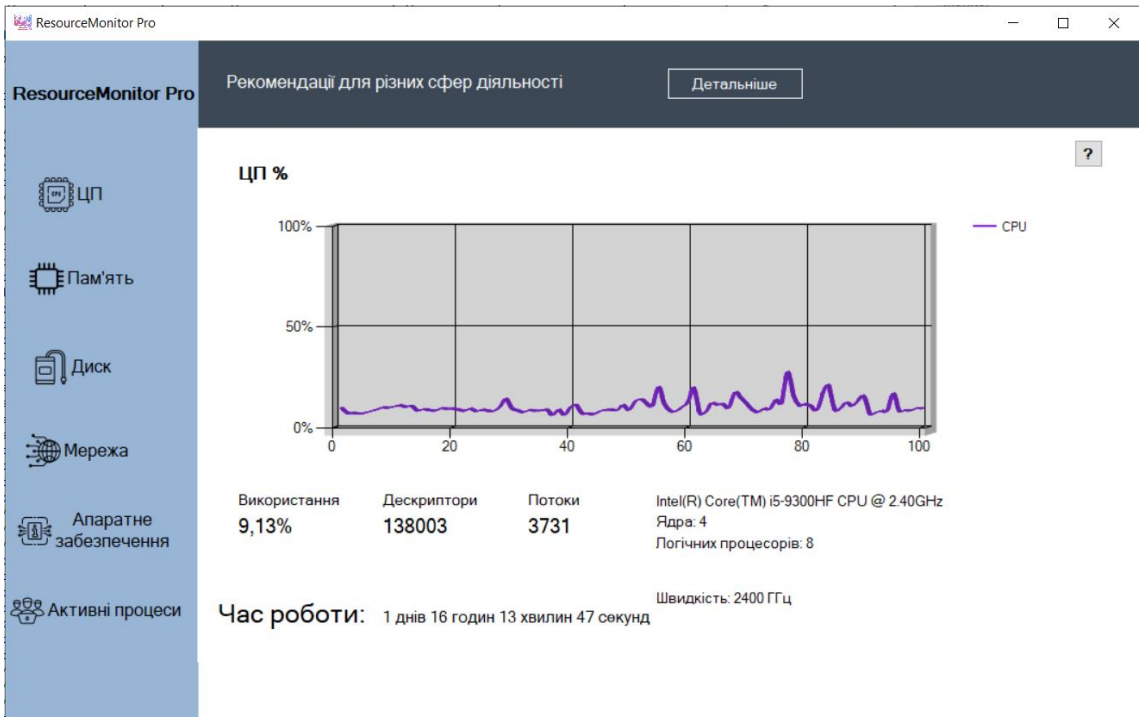


Рисунок 4.24 – Навантаження на процесор на ноутбуці Lenovo

На рис. 4.25 зображено навантаження на пам’ять для другого пристрою.

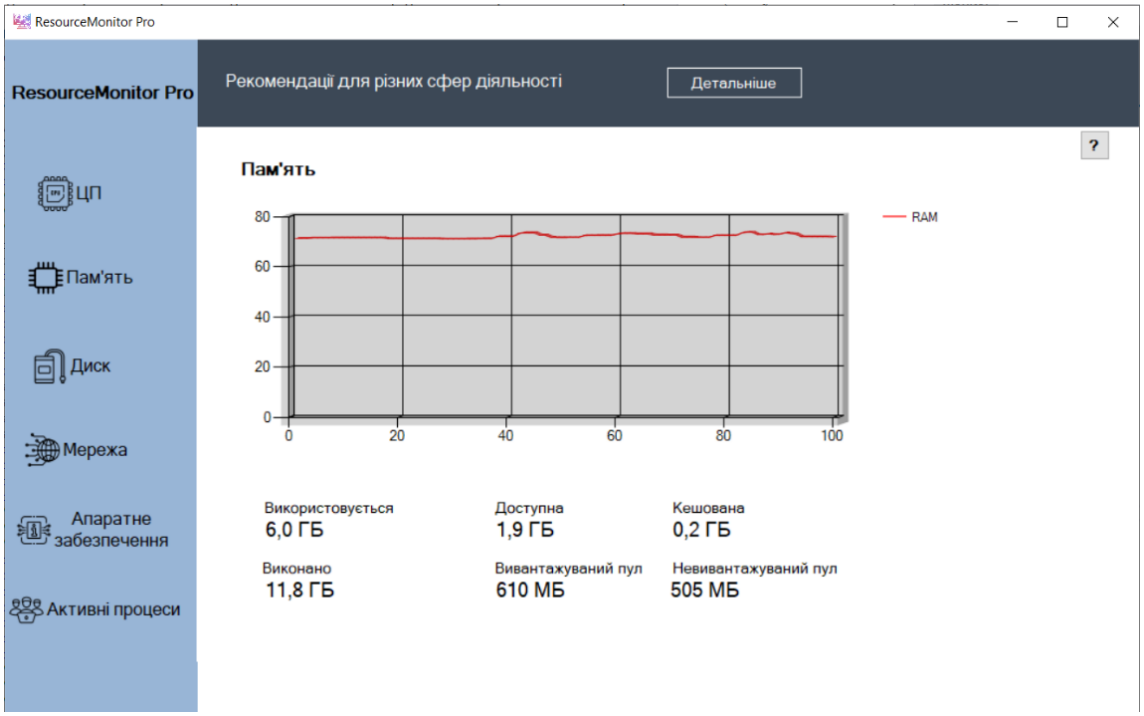


Рисунок 4.25 – Навантаження на пам’ять на ноутбуці Lenovo

На рис. 4.26 зображено навантаження на процесор для третього пристрою.

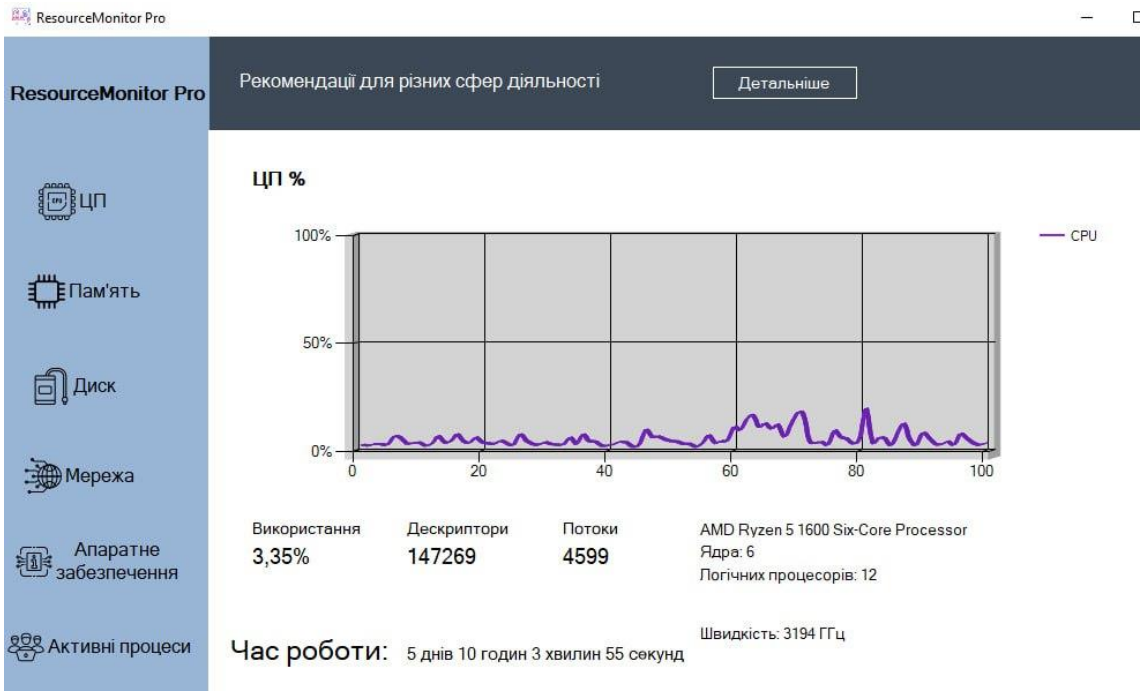


Рисунок 4.26 – Навантаження на процесор на ПК

На рис. 4.27 зображено навантаження на пам’ять для третього пристрою.

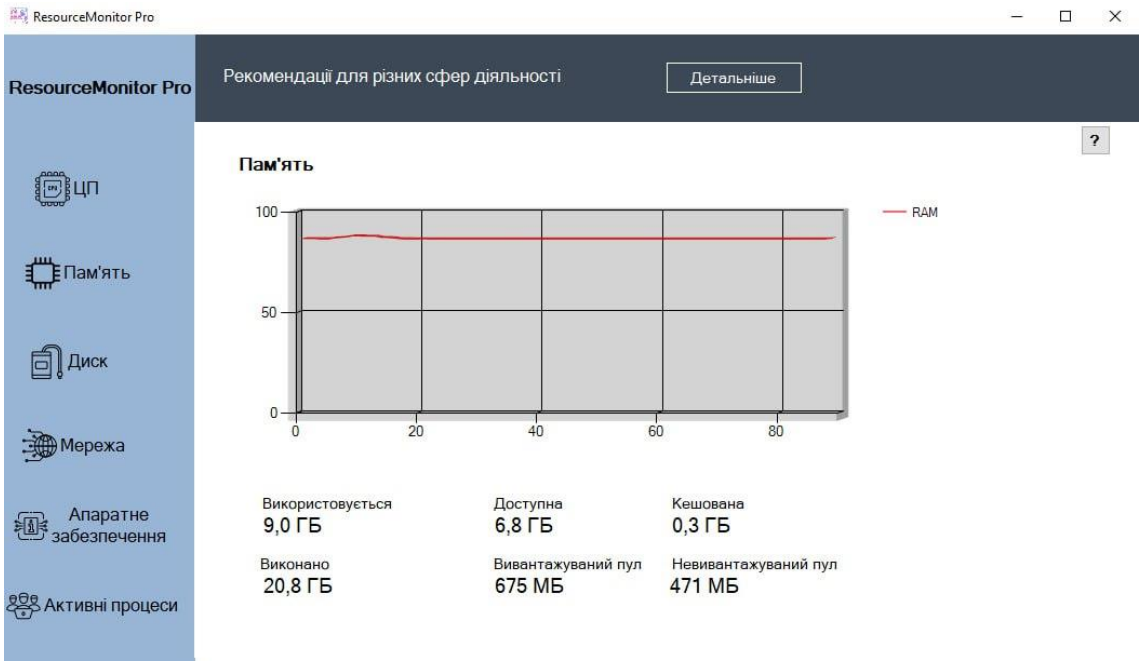


Рисунок 4.27 – Навантаження на пам’ять на ПК

Згідно даного тестування можна зробити висновки, що з однаковим навантаженням різні пристрої поводяться по-різному. Перший навантажується найбільше, а третій найменше. Для того щоб система не навантажувалась, а конференція проводилась стабільно, необхідно дотримуватись зазначеним рекомендаціям у застосунку.

Висновки до розділу 4

Під час написання четвертого розділу було розглянуто та описано основну структуру, інтерфейс та повний функціонал розробленої системи моніторингу стабільності інтернет-конференції в умовах багатозадачності.

Окрім можливості моніторінгу основних параметрів пристрою, також є наступний функціонал:

– підключення до БД, з можливістю завантажити усі додані дані, також провести пошук даних за датою запису;

– виведення повідомлень про надмірне навантаження на систему у разі аварійних ситуацій;

– рекомендації для користувача для забезпечення стабільності роботи інтернет-конференцій на конкретному пристрої.

Було проведено декілька різних тестів для аналізу стабільність роботи конференції при різних умовах (аудіо, відео, шейрінг екрану). У розробленому застосунку. Для тестувань було обрано різне ПЗ (Zoom, Visual Studio, Chrome, Word тощо), щоб імітувати реальне навантаженння на систему.

Також було проведено порівняльний аналіз споживання ресурсів інтернет-конференції на різних пристроях та на основі нього виведено основі рекомендації для користувачів щодо забезпечення стабільності роботи інтернет-конференцій на пристроях з певними характеристиками.

ВИСНОВКИ

В результаті виконання бакалаврської кваліфікованої роботи було досягнуто поставленої мети – досліджено та проведено аналіз стабільності інтернет-конференції в умовах багатозадачності за допомогою розробленої системи моніторингу на мові програмування C# та платформі Windows Forms

Зазначену мету досягнуто завдяки розв’язанню наступних задач:

– було досліджено предметну сферу, проведено аналіз літератури та публікацій;

– було досліджено вплив інтернет-конференцій на споживання ресурсів пристроїв: пам’яті, центрального процесора (ЦП), а також трафіку передачі даних та ін.;

– було проаналізовано аналогічні системи для аналізу стабільності інтернет-конференції;

– було визначено функціонал необхідного застосунку та засоби для його реалізації;

– було розроблено застосунок за допомогою обраних технологій;

– було протестовано розроблене програмне забезпечення.

Було досліджено предметну сферу, а саме використання відеоконференцій та їх ресурсоспоживання, і визначено, що інтернет-конференції – один з найпоширеніших видів зв’язку сьогодення, який об'єднує людей з усього світу в онлайн середовищі для обговорення спільних інтересів та проблем. Тому підтримка стабільної роботи інтернет-конференцій при будь-яких обставинах за допомогою аналізу та керування навантаженням пристроїв в умовах багатозадачності є актуальним завданням.

Було проведено аналіз аналогічних систем моніторингу, визначено їх переваги та недоліки. На основі цього дослідження було визначено наступні вимоги до застосунку:

– простий та інтуїтивно зрозумілий інтерфейс з сучасним дизайном;

– логічно продуманий і не перевантажений зайвими можливостями функціонал;

– застосунок із забезпеченням діалогу з користувачем при аварійних ситуацій;

– виведення рекомендацій для забезпечення стабільної роботи інтернет-конференцій в умовах багатозадачності.

У другому розділу було визначено основні засоби для реалізації застосунку для моніторингу стабільності інтернет-конференцій та обрано базу даних для зберігання даних системи моніторингу.

У наступному розділу було розроблено UML-діаграми моделі програмного забезпечення, для моделювання процесів роботи системи та подальшої розробки архітектури. Також було визначено структуру застосунку та основні функції для кожного класу.

В останьому розділі було описано інтерфейс та основний функціонал розробленої системи моніторингу стабільності інтернет-конференції. Серед основного функціоналу системи можна зазначити наступне:

– виведення у головному вікні активних процесів, використання пам’яті, ЦП та дискового простору протягом дня;

– виведення більш детальної інформації про споживання ресурсів простою у графічному вигляді в окремих вікнах;

– підключення до БД, з можливістю завантажити усі додані дані, також провести пошук даних за датою запису;

– виведення повідомлень про надмірне навантаження на систему у разі аварійних ситуацій;

– рекомендації для користувача для забезпечення стабільності роботи інтернет-конференцій на конкретному пристрої.

Далі було проведено декілька різних тестів для аналізу стабільність роботи конференції при різних умовах (аудіо, відео, шейрінг екрану). У розробленому

застосунку. Для тестувань було обрано різне ПЗ (Zoom, Visual Studio, Chrome, Word тощо), щоб імітувати реальне навантаженння на систему.

Також було проведено порівняльний аналіз споживання ресурсів інтернет-конференції на різних пристроях та на основі нього виведено основні рекомендації для користувачів щодо забезпечення стабільності роботи інтернет-конференцій на пристроях з певними харантеристиками.

Практичне значення полягає в тому, що було реалізовано інформування користувача при аварійних ситуаціях, розроблено рекомендації, що надають користувачу можливі дії у разі наставання аварійних ситуацій.

Розроблений застосунок в режимі реального часу повідомляє користувача за допомогою Windows Forms про перенавантаженість ресурсів ПК та засобами прийнятної візуалізації надає необхідну інформацію для зменшення кількості виконуваних задач з метою забезпечення стабільності інтернет-конференції.

Забезпечує користувача рекомендаціями для покращення роботи стабільності конференцій в умовах багатозадачності.

Результати кваліфікаційної бакалаврської роботи пройшли апробацію під час XI Міжнародної науково-практичної конференції «Free and Open Source Software (FOSS’2023)» (Харків, 14–16 лютого 2023 р.) та Всеукраїнської науково-практичної конференції «Інформаційні технології та інженерія» (Миколаїв, 07–10 лютого 2023 р.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бухтіярова О. О., Журавська І. М., Обухова К. О. Програмне забезпечення для моніторингу стабільності відеоконференцій в умовах багатозадачності на базі Winforms. *Free and Open Source Software (FOSS'2023)* : тези доп. XI Міжнар. наук.-практ. конф. / Харків. нац. економ. ун-т ім. Семена Кузнеця. Харків, 14–16 лютого 2023 р. Харків : ХНЕУ ім. Семена Кузнеця, 2023. С. 42–44.

2. Бухтіярова О. О., Обухова К. О. Моніторинг стабільності інтернет-конференції в умовах багатозадачності. І*нформаційні технології та інженерія* : тези доп. Всеукр. наук.-практ. конф. Миколаїв, 07–10 лют. 2023 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2023. С. 10–12.

3. What is the real impact of video conferences on the new work environment? URL: https://www.mixvoip.com/what-data-reveals-the-real-impact-of-video-conferences-on-the-new-work-environment (Last accessed: 04.05.2023).

4. Aristovnik A., Keržič D., Ravšelj D., Tomaževič N., Umek L. Impacts of the COVID-19 pandemic on life of higher education students: A global perspective. *Sustainability*. 2020. No. 12(20). DOI: 10.3390/su12208438.

5. Factors Affecting Video Conferencing. URL: https://www.geeksforgeeks.org/factors-affecting-video-conferencing/ (Last accessed: 04.05.2023).

6. Belyh A. The Ultimate List of Video Conferencing Statistics for 2023. URL: https://www.founderjar.com/video-conferencing-statistics/ (Last accessed: 06.05.2023)

7. Sadler M. Video Conferencing Statistics. URL: https://www.trustradius.com/vendor-blog/web-conferencing-statistics-trends (Last accessed: 06.05.2023).

8. Bieringa R., Radhakrishnan A., Singh T., Vos S., Donkervliet J., Iosup A. An Empirical Evaluation of the Performance of Video Conferencing Systems. *ICPE '21: ACM/SPEC International Conference on Performance Engineering*. 2021. P. 65–71. DOI: 10.1145/3447545.3451186.

9. Kumar R., Nagpal D., Naik V., Chakraborty D. Comparison of popular video DOI: conferencing apps using client-side measurements on different backhaul networks. *MobiHoc '22: Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 2022. P. 241-246. DOI: 10.48550/arXiv.2210.09651.

10. Keary T. Best PC & Hardware Monitoring Software & Tools for 2023. URL: https://www.comparitech.com/net-admin/best-hardware-monitoring-software/ (Last accessed: 09.05.2023).

11. Голуб Б. М. C#. Концепція та синтаксис : навч. посібник. Львів : Видавничий центр ЛНУ імені Івана Франка, 2016. 136 с.

12. What Is C# URL: https://www.designveloper.com/blog/what-is-c-sharp/ (Last accessed: 15.05.2023).

13. What is Visual Studio? URL: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022 (Last accessed: 15.05.2023).

14. About .NET Framework URL: https://www.knowledgehut.com/tutorials/csharp/csharp-introduction#:~:text=Disadvantages%20of%20C%23Some%20of,is%20a%20part%20of%20the%20 (Last accessed: 15.05.2023).

15. Windows Forms .NET. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0 (Last accessed: 15.05.2023).

16. What Is Windows Forms And Its Benefits. URL: https://issuu.com/arctionltdchart/docs/what_is_windows_forms_and_its_benefits (Last accessed: 15.05.2023).

17. System.Diagnostics Namespace. URL: https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics?view=net-7.0 (Last accessed: 17.05.2023).

18. Performance Counters. URL: https://learn.microsoft.com/en-us/dotnet/api/system.net.networkinformation?view=net-7.0 (Last accessed: 17.05.2023).

19. System.Net.NetworkInformation Namespace. URL: https://learn.microsoft.com/en-us/dotnet/api/system.net.networkinformation?view=net-7.0 (Last accessed: 17.05.2023).

20. System.Speech.Synthesis. URL: https://learn.microsoft.com/en-us/dotnet/api/system.speech.synthesis.speechsynthesizer?view=netframework-4.8.1 (Last accessed: 17.05.2023).

21. Chamberlin D. Early history of SQL. *IEEE Annals of the History of Computing*. 2012. Vol. 34, Is .4. P. 78–82. DOI: 10.1109/MAHC.2012.61.

22. Kreibich J. Using SQLite. Small. Fast. Reliable. Choose Any Three. *O'Reilly Media*, 2010. 530 p.

23. Loshin P. Structured Query Language (SQL). URL: https://www.techtarget.com/searchdatamanagement/definition/SQL (Last accessed: 19.05.2023).

24. Rumpe B. Modeling with UML. *Springer Cham*, 2016. 281 p. DOI: 10.1007/978-3-319-33933-7.

25. UML Diagram Types Guide. URL: https://creately.com/blog/diagrams/uml-diagram-types-examples/ (Last accessed: 19.05.2023).

ДОДАТОК А

Код програми

DataBase.cs

```
public class DataBase
    {

        private SQLiteConnection connection;

        public DataBase(string databasePath)
        {
            connection = new SQLiteConnection("Data Source=" + databasePath);
            connection.Open();
            string createProcessesTable = "CREATE TABLE IF NOT EXISTS processes (processName TEXT,
memoryUsage TEXT, status TEXT, upTime TEXT, createdDate TEXT)";
            ExecuteNonQuery(createProcessesTable);

            string createActivityTable = "CREATE TABLE IF NOT EXISTS activity (cpu TEXT, ram TEXT,
createdDate TEXT)";
            ExecuteNonQuery(createActivityTable);
        }

        public void insertProcessesRecord(string processName, string memoryUsage, string status,
string upTime, string createdDate)
        {
            string insertProcessesRecord = "INSERT INTO processes (processName, memoryUsage, status,
upTime, createdDate) VALUES (@processName, @memoryUsage, @status, @upTime, @createdDate)";
            using (SQLiteCommand command = new SQLiteCommand(insertProcessesRecord, connection))
            {
                command.Parameters.AddWithValue("@processName", processName);
                command.Parameters.AddWithValue("@memoryUsage", memoryUsage);
                command.Parameters.AddWithValue("@status", status);
                command.Parameters.AddWithValue("@upTime", upTime);
                command.Parameters.AddWithValue("@createdDate", createdDate);
                command.ExecuteNonQuery();
            }
        }

        public void insertActivityRecord(string cpu, string ram, string createdDate)
        {
            string insertActivityRecord = "INSERT INTO activity (cpu, ram, createdDate) VALUES
(@cpu, @ram, @createdDate)";
            using (SQLiteCommand command = new SQLiteCommand(insertActivityRecord, connection))
            {
                command.Parameters.AddWithValue("@cpu", cpu);
                command.Parameters.AddWithValue("@ram", ram);
                command.Parameters.AddWithValue("@createdDate", createdDate);
                command.ExecuteNonQuery();
            }
        }
        public DataTable getProcessesData()
        {
            string selectProcessesData = "SELECT * FROM processes";
            using (SQLiteCommand command = new SQLiteCommand(selectProcessesData, connection))
            {
                using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
                {
                    DataTable processesTable = new DataTable();
                    adapter.Fill(processesTable);
                    return processesTable;
```

```
        }
    }
}

public DataTable getActivityData()
{
    string selectActivityData = "SELECT * FROM activity";
    using (SQLiteCommand command = new SQLiteCommand(selectActivityData, connection))
    {
        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
            DataTable activityTable = new DataTable();
            adapter.Fill(activityTable);
            return activityTable;
        }
    }
}
public void deleteAllRecords()
{
    string deleteProcessesRecords = "DELETE FROM processes";
    string deleteActivityRecords = "DELETE FROM activity";
    ExecuteNonQuery(deleteProcessesRecords);
    ExecuteNonQuery(deleteActivityRecords);
}

public DataTable dateSearch(string searchDate)
{
    string selectQuery = "SELECT * FROM processes WHERE createdDate LIKE @searchDate";

    using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))
    {
        command.Parameters.AddWithValue("@searchDate", searchDate + "%");

        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
            DataTable searchResultsTable = new DataTable();
            adapter.Fill(searchResultsTable);
            return searchResultsTable;
        }
    }

}
public DataTable dateSearchActivity(string searchDate)
{
    string selectQuery2 = "SELECT * FROM  activity  WHERE createdDate LIKE @searchDate";
    using (SQLiteCommand command = new SQLiteCommand(selectQuery2, connection))
    {
        command.Parameters.AddWithValue("@searchDate", searchDate + "%");

        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
            DataTable searchResultsTable2 = new DataTable();
            adapter.Fill(searchResultsTable2);
            return searchResultsTable2;
        }
    }
}

public DataTable getDataForCasual()
{
    string selectProcessesData = "SELECT * FROM casual";
    using (SQLiteCommand command = new SQLiteCommand(selectProcessesData, connection))
    {
        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
```

```
            DataTable processesTable = new DataTable();
            adapter.Fill(processesTable);
            return processesTable;
        }
    }
}
public DataTable getDataForOffice()
{
    string selectProcessesData = "SELECT * FROM office";
    using (SQLiteCommand command = new SQLiteCommand(selectProcessesData, connection))
    {
        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
            DataTable processesTable = new DataTable();
            adapter.Fill(processesTable);
            return processesTable;
        }
    }
}
public DataTable getDataForProgrammer()
{
    string selectProcessesData = "SELECT * FROM programmer";
    using (SQLiteCommand command = new SQLiteCommand(selectProcessesData, connection))
    {
        using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(command))
        {
            DataTable processesTable = new DataTable();
            adapter.Fill(processesTable);
            return processesTable;
        }
    }
}

private void ExecuteNonQuery(string query)
{
    using (SQLiteCommand command = new SQLiteCommand(query, connection))
    {
        command.ExecuteNonQuery();
    }
}

public void closeConnection()
{
    if (connection != null && connection.State == ConnectionState.Open)
    {
        connection.Close();
        connection.Dispose();
        connection = null;
    }
}

   }
```

MainForm.cs

```
public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
            string databasePath = "Activity.db";
            dataBase = new DataBase(databasePath);

        }
        private DataBase dataBase;
        private int count = 0;
        private void countTask(int count)
        {
            ComputerInfo cpI = new ComputerInfo();
            float cpu = CPU.NextValue();

            Process[] processes = Process.GetProcesses();

            ListViewItem item2 = new ListViewItem(string.Format("{0:0.00}%", cpu));
            item2.SubItems.Add(string.Format("{0:0.0} GB", (cpI.TotalPhysicalMemory -
cpI.AvailablePhysicalMemory) / (1024 * 1024 * 1024)));
            item2.SubItems.Add(DateTime.Now.ToString());
            listViewProcesses2.Items.Add(item2);
            dataBase.insertActivityRecord(string.Format("{0:0.00}%", cpu),
                                    string.Format("{0:0.0} GB", (cpI.TotalPhysicalMemory -
cpI.AvailablePhysicalMemory) / (1024 * 1024 * 1024)),
                                    DateTime.Now.ToString());


            listViewProcesses.Items.Clear();


            foreach (Process aprocess in processes)
            {
                if (aprocess.MainWindowHandle != IntPtr.Zero)
                {
                    ListViewItem item1 = new ListViewItem(aprocess.ProcessName);
                    item1.SubItems.Add(string.Format("{0} MB", aprocess.WorkingSet64 / (1024 *
1024)));
                    item1.SubItems.Add(aprocess.Responding ? "Active" : "Not Responding");
                    item1.SubItems.Add(aprocess.StartTime.ToString());
                    item1.SubItems.Add(DateTime.Now.ToString());
                    listViewProcesses.Items.Add(item1);
                    dataBase.insertProcessesRecord(aprocess.ProcessName,
                                        string.Format("{0} MB", aprocess.WorkingSet64 / (1024 *
1024)),
                                        aprocess.Responding ? "Active" : "Not Responding",
                                        aprocess.StartTime.ToString(),
                                        DateTime.Now.ToString());
                }
            }
        }

        private void buttonRam_Click(object sender, EventArgs e)
        {
            userControl11.BringToFront();
        }
        private void buttonCpu_Click(object sender, EventArgs e)
        {
```

```
   controlCpu1.BringToFront();
}

private void buttonDisk_Click(object sender, EventArgs e)
{
    controlDisk1.BringToFront();
}

private void buttonNetwork_Click(object sender, EventArgs e)
{
    controlNetwork1.BringToFront();
}

private void button_task_Click(object sender, EventArgs e)
{
    panelTask.BringToFront();
}

private void timer_Tick(object sender, EventArgs e)
{
    countTask(count);

    count++;
    timer.Interval = 10000;
}

private void MainForm_Load(object sender, EventArgs e)
{
    panel3.Visible = false;
    cleanDB.Visible = false;
    timer_Tick(null, null);

    listViewProcesses.View = View.Details;
    listViewProcesses2.View = View.Details;


    listViewProcesses.Columns.Add("Назва процесу");
    listViewProcesses.Columns.Add("Використання пам'яті");
    listViewProcesses.Columns.Add("Статус");
    listViewProcesses.Columns.Add("Дата запуску", 120);
    listViewProcesses.Columns.Add("Дата створення запису", 120);


    listViewProcesses2.Columns.Add("ЦП");
    listViewProcesses2.Columns.Add("Пам'ять", 80);
    listViewProcesses2.Columns.Add("Дата створення запису", 120);
}


private void dateField_TextChanged(object sender, EventArgs e)
{
    string searchDate = dateField.Text;
    DataTable searchResults = dataBase.dateSearch(searchDate);
    dataGridViewProcesse.DataSource = searchResults;
    DataTable searchResults2 = dataBase.dateSearchActivity(searchDate);

    dataGridViewActivity.DataSource = searchResults2;

}

private void archive_Click_1(object sender, EventArgs e)
{
```

```
            panel3.Visible = true;
            cleanDB.Visible = true;
            DataTable processesTable = dataBase.getProcessesData();
            dataGridViewProcesse.DataSource = processesTable;
            processesTable = dataBase.getActivityData();
            dataGridViewActivity.DataSource = processesTable;
        }

        private void cleanDB_Click(object sender, EventArgs e)
        {
            dataBase.deleteAllRecords();
        }

        private void btnCloseArchive_Click(object sender, EventArgs e)
        {

            panel3.Visible = false;
            cleanDB.Visible = false;
        }

        private void btnRec_MouseHover(object sender, EventArgs e)
        {
            panelInfo.Visible = true;
        }

        private void btnRec_MouseLeave(object sender, EventArgs e)
        {
            panelInfo.Visible = false;
        }

        private void buttonRec_Click(object sender, EventArgs e)
        {
            Recommendations recommendations = new Recommendations();
            recommendations.Show();
        }

        private void buttonInfo_Click(object sender, EventArgs e)
        {
            controlInformation1.BringToFront();
        }

           }
```

ControlCpu.cs

```
public partial class ControlCpu : UserControl
    {
        private bool cpuWarned = false;
        private static float sinceLastWarning = 0;

        public ControlCpu()
        {
            InitializeComponent();
        }
        private int count = 0;

        private void countCPU(int count)
        {
            float cpu = CPU.NextValue();
            float handle = pHandle.NextValue();
            float process = pProcess.NextValue();
```

```
            labelCPU.Text = string.Format("{0:0.00}%", cpu);
            labelHandle.Text = string.Format("{0:0}", handle);
            labelProcess.Text = string.Format("{0:0}", process);

            if (count >= 100)
                chartCPU.Series["CPU"].Points.RemoveAt(0);
            chartCPU.Series["CPU"].Points.AddY(cpu);
            object result =
Registry.GetValue("HKEY_LOCAL_MACHINE\\HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0",
"ProcessorNameString", "");
            if (result != null)
            {
                labelNameCPU.Text = result.ToString();
            }
            object result2 =
Registry.GetValue("HKEY_LOCAL_MACHINE\\HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0",
"~MHz", "");
            if (result2 != null)
            {
                BasicSpeed.Text = "Швидкість: " + result2 + " ГГц".ToString();
            }

            int pprocess = Environment.ProcessorCount;
            numProcess.Text = "Логічних процесорів: " + pprocess.ToString();
            numCores.Text = "Ядра: " + (pprocess / 2).ToString();

            if (cpu >= 100 && cpuWarned == false)
            {
                speakWarning("Увага!Навантаження процесора досягло 100 відсотків", 1);

                MessageBox.Show("Навантаження на процесор 100%, будь ласка закрийте неважливі
процеси! \nАле спочатку подивіться графік використання ЦП, якщо процесор не знижується,
негайно закрийте непотрібні процеси!", "Велике навантаження на процесор",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            else
                    if (cpu >= 80 && cpuWarned == false)
                speakWarning(String.Format("Навантаження на процесор {0:0.00}відсотків", cpu),
1);

            sinceLastWarning += 0.016666667f;
            if (sinceLastWarning >= 1)
            {
                cpuWarned = false;

            }

        }


            private void timer_Tick(object sender, EventArgs e)
            {
                updateUpTime();
                countCPU(count);

                count++;
            }

            private void speakWarning(string warningMessage, short warningType)
            {
```

```
                cpuWarned = (warningType == 1);
                sinceLastWarning = 0;

                SpeechSynthesizer synth = new SpeechSynthesizer();
                synth.SelectVoiceByHints(VoiceGender.Neutral);
                synth.SpeakAsync(warningMessage);
            }

            private void ControlCpu_Load(object sender, EventArgs e)
            {
                timer_Tick(null, null);
                updateUpTime();
            }

            private void updateUpTime()
            {

                float upTimeValue = perfCountUpTime.NextValue();
                TimeSpan upTimeSpan = TimeSpan.FromSeconds(upTimeValue);
                string upTimeText = String.Format("{0} днів {1} годин {2} хвилин {3} секунд",
(int)upTimeSpan.TotalDays, (int)upTimeSpan.Hours, (int)upTimeSpan.Minutes,
(int)upTimeSpan.Seconds);
                lblUpTime.Text = upTimeText;
            }

            private void buttonRec_MouseLeave(object sender, EventArgs e)
            {
                panelInfo.Visible = false;
            }

            private void buttonRec_MouseEnter(object sender, EventArgs e)
            {
                panelInfo.Visible = true;
            }

               }
```

ControlNetwork.cs

```
public partial class ControlNetwork : UserControl
    {
        private int count = 0;
        private bool isInternetAvailable = true;
        private PerformanceCounter pNetS = new PerformanceCounter();
        private PerformanceCounter pNetR = new PerformanceCounter();
        private ManagementObjectSearcher objectsearcherIP = new
ManagementObjectSearcher("SELECT * FROM Win32_NetworkAdapterConfiguration");

        public ControlNetwork()
        {
            InitializeComponent();
        }

        private void countInternet(int count)
        {
            if (internetAvailable())
            {
                if (!isInternetAvailable)
                {
                    isInternetAvailable = true;
```

```
                    MessageBox.Show("З'єднання з Інтернетом відновлено", "Статус з'єднання",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                NetworkInterface nic = getNIC();
                String name = String.Copy(nic.Description);
                name = name.Replace("(", "[");
                name = name.Replace(")", "]");
                try
                {
                    pNetS.CategoryName = "Network Interface";
                    pNetS.CounterName = "Bytes Sent/sec";
                    pNetS.InstanceName = name;
                    pNetR.CategoryName = "Network Interface";
                    pNetR.CounterName = "Bytes Received/sec";
                    pNetR.InstanceName = name;
                    float fsend = pNetS.NextValue();
                    float freceive = pNetR.NextValue();

                    labelNetS.Text = string.Format("{0:0.0} Кбіт/с", fsend * 8 / 1024);
                    labelNetR.Text = string.Format("{0:0.0} Кбіт/с", freceive * 8 / 1024);

                    if (count >= 100)
                    {
                        chartInternet.Series["Send"].Points.RemoveAt(0);
                        chartInternet.Series["Receive"].Points.RemoveAt(0);
                    }
                    chartInternet.Series["Send"].Points.AddY(fsend * 8 / 1024);
                    chartInternet.Series["Receive"].Points.AddY(freceive * 8 / 1024);
                }
                catch (InvalidOperationException e)
                {
                    Console.WriteLine(e.Message);
                }
                labelNetworkType.Text = String.Copy(nic.Description);
                labelNetworkType1.Text = "Адаптер: " + String.Copy(nic.Name);

                foreach (ManagementObject queryObj in objectsearcherIP.Get())
                {
                    if (queryObj["IPAddress"] == null)
                    {
                    }
                    else
                    {
                        String[] arrIPAddress = (String[])(queryObj["IPAddress"]);
                        foreach (String arrValue in arrIPAddress)
                        {
                            if (arrValue.StartsWith("192") || arrValue.StartsWith("172") ||
arrValue.StartsWith("10."))
                            {
                                IPv4.Text = "IPv4: " + arrValue;
                            }
                        }
                    }
                }
            }
            else
            {
                if (isInternetAvailable)
                {
                    isInternetAvailable = false;
                    MessageBox.Show("Немає підключення до Інтернету", "Помилка підключення",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }

            chartInternet.Series["Send"].Points.Clear();
            chartInternet.Series["Receive"].Points.Clear();
            labelNetworkType.Text = "Немає підключення до Інтернету";
            labelNetworkType1.Text = "Немає підключення до Інтернету";
            IPv4.Text = "Немає підключення до Інтернету";
        }
    }

    public NetworkInterface getNIC()
    {
        NetworkInterface[] nics = NetworkInterface.GetAllNetworkInterfaces();
        foreach (NetworkInterface nic in nics)
        {
            if (nic.OperationalStatus.ToString().Equals("Up"))
            {
                return nic;
            }
        }
        return null;
    }

    private bool internetAvailable()
    {
        try
        {
            using (Ping ping = new Ping())
            {
                PingReply reply = ping.Send("www.google.com", 1000);
                return (reply != null && reply.Status == IPStatus.Success);
            }
        }
        catch
        {
            return false;
        }
    }

    private void timer_Tick(object sender, EventArgs e)
    {
        countInternet(count);
        count++;
    }



    private void buttonRec_MouseEnter(object sender, EventArgs e)
    {
        panelInfo.Visible = true;
    }

    private void buttonRec_MouseLeave(object sender, EventArgs e)
    {
        panelInfo.Visible = false;
    }

   }
```

Recommendations.cs

```
public partial class Recommendations : Form
    {
        public Recommendations()
        {
            InitializeComponent();
            string databasePath = "Activity.db";
            dataBase = new DataBase(databasePath);
        }
        private DataBase dataBase;


        private void office_CheckedChanged(object sender, EventArgs e)
        {
            DataTable officeUserTable = dataBase.getDataForOffice();
            dataGridViewRecomm.DataSource = officeUserTable;
            label3.Text = $"Рекомендовані характеристики ПК:\nПроцесор - Intel i3 10100/Ryzen
3 4300g \nОперативна пам'ять - 8GB \nВідеокарта - Інтегрована в процесор \nЖорсткий диск - SSD
500GB";

        }

        private void programmer_CheckedChanged(object sender, EventArgs e)
        {
            DataTable programmerUserTable = dataBase.getDataForProgrammer();
            dataGridViewRecomm.DataSource = programmerUserTable;
            label3.Text = $"Рекомендовані характеристики ПК:\nПроцесор - Intel i5 10400/Ryzen
5 5600 \nОперативна пам'ять - 16GB \nВідеокарта - GTX 1650/RX 6400 \nЖорсткий диск - SSD 256GB
+ 1TB HDD";
        }

        private void casual_CheckedChanged(object sender, EventArgs e)
        {
            DataTable casualUserTable = dataBase.getDataForCasual();
            dataGridViewRecomm.DataSource = casualUserTable;
            label3.Text = $"Рекомендовані характеристики ПК:\nПроцесор - Intel i3 10100/Ryzen
3 4300g \nОперативна пам'ять - 8GB \nВідеокарта - Інтегрована в процесор \nЖорсткий диск - SSD
240GB";

        }

        private void btnRec_MouseHover(object sender, EventArgs e)
        {
            panelInfo.Visible = true;
        }

        private void btnRec_MouseLeave(object sender, EventArgs e)
        {
            panelInfo.Visible = false;
        }

        private void buttonClose_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

       }
```

ДОДАТОК Б

Матеріали апробації роботи

Б.1 XI Міжнародна науково-практична конференція «FOSS'2023»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАТЕРІАЛИ

XIV-ої МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

«FREE AND OPEN SOURCE SOFTWARE»

Дякуємо за підтримку

IDCM epam

14-16 лютого 2023 р.
м. Харків

- Потрібно більше серверних ресурсів, оскільки Next.js формує HTML на стороні сервера перед надсиланням його клієнту.

Ще одним відомим недоліком Next.js іноді називають той факт, що це самостійний фреймворк, тобто він має певний метод і набір інструментів, яких потребує від розробників при створенні додатків. Однак ці потреби не виходять за межі сучасних вимог у веб-розробці, проте можливості Next.js цілком задовольняють більшості проектів [7].

Наразі навколо Next.js сформувалося велике та активне комюніті, а сам фреймворк активно підтримується та розвивається розробниками. Всі оновлення публікуються в блозі і детально описані в документації, поряд із мінорними інтерактивним туторіалом. Останні оновлення мають адаптацію React 18, нові файлові конвенції, паралельні запити даних та багато іншого. Next.js активно використовують та підтримують великі компанії на кшталт Google, а Vercel пропонує зручну інфраструктуру для деплою додатків. Отже можна з упевненістю сказати, що Next.js буде і далі популярним у найближчому майбутньому.

Література

[1] Stepan Prokopiak: Багатосторінкові та односторінкові додатки, їх переваги та недоліки [Електронний ресурс]. – Режим доступу до ресурсу: https://dou.ua/forums/topic/25444/

[2] React. Офіційна документація. [Електронний ресурс]. – Режим доступу до ресурсу: https://reactjs.org

[3] Офіційна сторінка фреймворку NextJS [Електронний ресурс]. – Режим доступу до ресурсу: https://nextjs.org/

[4] SSR - SSG - ISR - CSR in Next.js-The Ultimate Guide [Електронний ресурс]. – Режим доступу до ресурсу: https://dev.to/dmzhavh/ssr-ssg-isr-csr-in-nextjs-the-ultimate-guide-256m

[5] Сторінка бета-документації проекту ReactJS [Електронний ресурс]. – Режим доступу до ресурсу: https://beta.reactjs.org/

[6] What does CSR, SSR, SSG and ISR means and why should you care? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.flatirons.com/data-building-strategy-for-nextjs-app/

[7] Олександр Жеребцов, Антон Іванов. «Використання функціонального програмування у JavaScript та фреймворках», 2022 International Conference on Innovative Solutions in Software Engineering (ICISSE). Vasyl Stefanyk Precarpathian National University. Ivano-Frankivsk. Ukraine. Nov. 29-30, 2022. pp. 162-169 с.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ СТАБІЛЬНОСТІ ВІДЕОКОНФЕРЕНЦІЙ В УМОВАХ БАГАТОЗАДАЧНОСТІ НА БАЗІ WINFORMS

Бухтіярова О.О., Журавська І.М., Обухова К.О.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ СТАБІЛЬНОСТІ ВІДЕОКОНФЕРЕНЦІЙ В УМОВАХ БАГАТОЗАДАЧНОСТІ НА БАЗІ WINFORMS

Бухтіярова О.О.
Керівники: Журавська І.М., Обухова К.О.
E-mail: lenab9666@gmail.com
Миколаїв, Чорноморський національний університет імені Петра Могили

Через пандемію COVID-19 та інші виклики сьогодення багато організацій, підприємств та навчальних закладів перейшли на нову структуру онлайн-роботи [1]. Зустрічі відбуваються через системи відеоконференцій, які дозволяють сотням мільйонів користувачів спілкуватися онлайн за допомогою аудіо- та відеопотоків.

Актуальність аналізу ресурсоспоживання відеоконференцій [2] в умовах багатозадачності комп'ютера полягає в тому, що такий аналіз дозволяє оцінити рівень загальної продуктивності роботи комп'ютера, визначити стандартні алгоритми для зниження навантаження, і як наслідок покращити стабільність роботи з інтернет-конференціями та іншими програмами й сайтами. На сьогоднішній день це є дуже актуальним, адже багато

людей не мають стабільного інтернет-з'єднання та використовують різні ПК та ноутбуки, у тому числі дуже старі моделі.

Метою роботи є створення програми для аналізу стабільності інтернет-конференції в умовах багатозадачності на мові програмування C# та платформі Windows Forms.

Windows Forms – це платформа для створення графічного інтерфейсу класичних застосунків Windows. Забезпечує один з найефективніших способів створення класичних застосунків на допоміжною візуального конструктора в Visual Studio.

Даний застосунок буде мати наступний функціонал:

- виведення інформації про такі параметри, як CPU, фізична й віртуальна пам'ять, інтернет трафік та інформації про всі активні процеси;
- візуалізація отриманої інформації, за допомогою інформативних графіків;
- виведення повідомлень при досягненні критичних значень параметрів (графічні та аудіо).

Щоб отримати відомості про поточний стан процесору, фізичної та віртуальної пам'яті буде використано клас PerformanceCounter із простору імен System.Diagnostics. Для отримання даних про запущені процеси буде використано клас Process із простору імен System.Diagnostics. Цей клас дозволяє аналізувати дії з процесами, зокрема завантажити інформацію про запущені процеси, потоки та прив'язані до процесу модулі.

Дані про інтернет-трафік будуть отримані за допомогою бібліотеки System.Net.NetworkInformation. Ця бібліотека надає класи, які містять інформацію про мережеві параметри та доступ до окремих мережевих ресурсів.

Рисунок 1 – Прототип програми

Для оцінки продуктивності інтернет-конференцій було досліджено аналіз проведення оцінки ресурсоспоживання систем відеоконференцій у 2022 році. В результаті цього аналізу було досліджено, що Microsoft Teams і Zoom поводяться однаково у мобільному 4G-інтернеті, і в дротовому широкосмуговому, включно зі споживаними ресурсів. Однак Google Meet обмежує свою пропускну здатність у мобільному інтернеті 4G, тому це знижує якість відео, а також він більше споживає пам'ять. Zoom і Microsoft Teams забезпечують кращу якість відео.

Якщо аудіо для користувача важливіше, то Google Meet і Zoom є кращими варіантами для дротового широкосмугового зв'язку. Якщо відео важливіше, Microsoft Teams і Zoom кращі варіанти. Користувачеві, який зацікавлений в економії трафіку, потрібно вибрати Google Meet, особливо при використанні мобільного інтернету 4G.

При дротовому широкосмуговому зв'язку, Microsoft Teams має стабільновисоке навантаження процесора порівняно з Zoom та Google Meet. Спостерігається значне збільшення обсягу пам'яті, що використовується Google Meet, коли камера увімкнена.

Test Type	Platform	CPU Load (%)	Memory Consumption (MB)	Battery Consumption (%)
Mic OFF Cam OFF	Google Meet	13.35	339.61	5.00
	MS Teams	26.58	294.68	8.00
	Zoom	15.91	355.23	6.00
Mic ON Cam OFF	Google Meet	22.68	388.71	6.00
	MS Teams	29.21	312.39	9.00
	Zoom	16.05	358.82	10.00
Mic OFF Cam ON	Google Meet	25.14	555.40	7.00
	Teams	30.89	369.77	10.00
	Zoom	20.54	370.17	9.00
Mic ON Cam ON	Google Meet	25.22	573.90	8.00
	MS Teams	31.74	372.20	10.00
	Zoom	22.88	382.85	10.00

Рисунок 2 – Споживання ресурсів при дротовому широкосмуговому зв'язку

При мобільному інтернеті 4G, Google Meet і Zoom мають схоже використання процесора, тоді як Microsoft Teams має порівняно більше використання процесора та має більше споживання пам'яті. Google Meet значно збільшує споживання пам'яті, коли увімкнено камеру.

Test Type	Platform	CPU Load (%)	Memory Consumption (MB)	Battery Consumption (%)
Mic OFF Cam OFF	Google Meet	12.457	336.608	7.00
	MS Teams	28.173	243.624	8.00
	Zoom	10.199	276.107	7.00
Mic ON Cam OFF	Google Meet	12.820	356.434	8.00
	MS Teams	36.993	267.860	9.00
	Zoom	12.841	296.275	8.00
Mic OFF Cam ON	Google Meet	19.182	455.40	7.00
	MS Teams	31.294	365.27	9.00
	Zoom	17.327	402.39	10.00
Mic ON Cam ON	Google Meet	19.842	537.890	9.00
	MS Teams	33.321	394.626	10.00
	Zoom	18.826	460.348	11.00

Рисунок 3 – Споживання ресурсів при мобільному інтернеті 4G

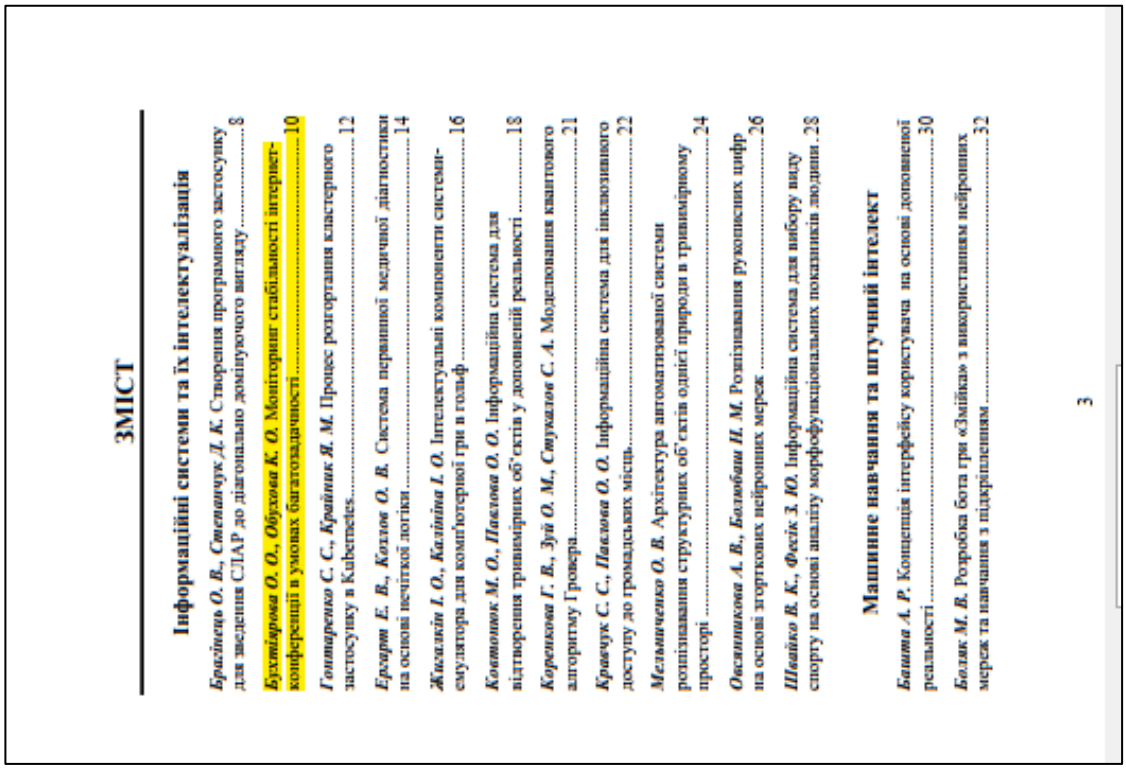
Глобальні виклики останніх років змусили суспільство перейти до дистанційної освіти та роботи, зокрема до зв'язку через відеоконференції. Тому підтримка стабільної роботи інтернет-конференцій при будь-яких обставинах та допомогого аналізу та керування навантаженим пристроїв в умовах багатозадачності є актуальним завданням.

Література

[1] Aristovnik A., Keržič D., Ravšelj D., Tomaževič N., Umek L. Impacts of the COVID-19 pandemic on life of higher education students: A global perspective. Sustainability 2020. № 12(20). DOI: 10.3390/su12208438.

[2] Bierinen R., Radhakrishnan A., Singh T., Vos S., Donlercviat J., Iosup A. An Empirical Evaluation of the Performance of Video Conferencing Systems. ICPE '21: ACM/SPEC International Conference on Performance Engineering. 2021. P. 65–71. DOI: 10.1145/3427345.3431186.

Б.2 Всеукраїнська науково-практична конференція «Інформаційні технології та інженерія»



ЗМІСТ

Інформаційні системи та їх інтелектуалізація

Машинне навчання та штучний інтелект

Міністерство освіти і науки України
Чорноморський національний університет
імені Петра Могили

PETRO MOHYLA BLACK SEA NATIONAL UNIVERSITY
COMPUTER SCIENCE FACULTY Est. 1996

«Інформаційні технології та інженерія»

Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів

ТЕЗИ

7–10 лютого 2023 року

Миколаїв – 2023

УДК 004.738.5:004.3

Бухтіярова О. О., Обухова К. О.
Чорноморський національний університет ім. Петра Могили, Миколаїв, Україна

МОНІТОРИНГ СТАБІЛЬНОСТІ ІНТЕРНЕТ-КОНФЕРЕНЦІЇ В УМОВАХ БАГАТОЗАДАЧНОСТІ

Актуальність аналізу працездатності інтернет-конференції в умовах багатозадачності комп'ютера полягає в тому, що такий аналіз дозволяє оцінити рівень загальної працездатності комп'ютера, визначити його міри протидії навантаженням та покращити продуктивність користування інтернет-конференціями та іншими програмами та сайтами. На сьогоднішній день це є дуже актуально, адже багато людей знаходяться не вдома, не має постійного інтернету, має застарілі ноутбуки. Тому моніторинг навантаженості комп'ютерної системи та розробка спеціалізованого програмного забезпечення для повідомлення користувача про необхідність обмеження ресурсів, що витрачаються, буде корисним у сьогоднішніх умовах.

Програма для аналізу працездатності інтернет-конференції в умовах багатозадачності може бути написана за допомогою мови програмування C# та платформи Windows Forms [1, 2].

Windows Forms – це платформа користувача інтерфейсу для створення класичних застосунків Windows. Вона забезпечує один з найефективніших способів створення класичних застосунків за допомогою візуального конструктора в Visual Studio.

Розроблений застосунок для моніторингу задіяних ресурсів комп'ютера виконує наступні функції: виведення параметрів CPU, фізичної та віртуальної пам'яті; виведення графіків параметрів; виведення повідомлення при досягненні критичних значень параметрів; отримання всіх запущених процесів; виведення інформації про інтернет-графік (рис. 1).

Щоб отримати значення навантаженості центрального процесору, фізичної та віртуальної пам'яті доцільно використати клас PerformanceCounter із простору імен System.Diagnostics.

Для отримання даних про запущені процеси використано клас Process із простору імен System.Diagnostics. Це дозволяє виконати дії з процесами, зокрема завантажити інформацію про запущені процеси, потоки та прив'язані до процесу модулі.

Дані про інтернет-графік отримуються за допомогою бібліотеки System.Net.NetworkInformation. Ця бібліотека надає класи,

які містять інформацію про мережеві параметри та доступ до окремих мережевих ресурсів. За допомогою цієї бібліотеки можна отримувати інформацію про мережеві параметри.

Рисунок 1 – Прототип застосунку

Слід зазначити, що затримку голосу та/або відео під час користування інструментами вебконференцій (Google Meet, Microsoft Teams, Zoom, Facebook, Skype та інших комерційних інструментів або інструментів з відкритим кодом) не вдається зменшити або виключити, якщо вона викликана вибором серверів [3]. Але користувач може прийняти рішення щодо зменшення кількості задач, що виконуються на комп'ютері одночасно з інтернет-конференцією.

Розроблений застосунок в режимі реального часу повідомляє користувача за допомогою Windows Forms про перевантаженість ресурсів ПК та засобами прийнятної візуалізації надає необхідну інформацію для зменшення кількості використаних задач з метою забезпечення стабільності інтернет-конференції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yosifovich P. Solomon D. A., Ionescu A. Windows Internals: System architecture, processes, threads, memory management, and more. Part 1 (Developer Reference). 7th Ed. Microsoft Press. 2017. 800 p.
2. Farrell J. Microsoft Visual C#: An Introduction to Object-Oriented Programming. Cengage Learning. 2017. 784 p.
3. Tsioutas K., Xylomenos G. Audio delay in web conference tools. *Web Audio Conference WAC-2022*. July 6–8, 2022. Cannes, France. DOI: