МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Обслуговування комп'ютерних систем і мереж»

Група: 4КС-58

Дипломний проект

здобувача освіти денної форми навчання
КС.58.20.000.ДП

ЦЮСЬМАКА МИКОЛАЯ АНДРІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Обслуговування комп'ютерних систем і мереж»**

Група: **4КС-58**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання

Проектний матеріал складається з пояснювальної записки на 75 сторінках та графічного (презентаційного) матеріалу на 13 аркушах (слайдах)

Дипломник _____ (Цюсьмак М.А.)

Керівник _____ (Нестеренко В.Д)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Красножупська К.Г.)

Захист «28» червня 2025 р. Протокол ЕК № 7

Оцінка ЕК 5 (відмінно) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Обслуговування комп'ютерних систем та мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Цюсьмаку Миколаю Андрійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання*

затверджена наказом по коледжу від "14" листопада 2024 р. № 246

2. Термін здачі закінченого проекту 16.06.2025

3. Вихідні данні до проекту *1. Реалізувати програму для моніторингу стану декількох мережевих інтерфейсів; 2. Реалізувати періодичне опитування параметрів якості з'єднання; 3. Забезпечити можливість налаштування порогових значень якості; 4. Передбачити автоматичне перемикання між мережами при погіршенні якості активного з'єднання; 5. Реалізувати механізм вибору найкращого альтернативного інтерфейсу; 6. Передбачити візуалізацію результатів моніторингу у вигляді графіків;*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз та класифікація систем моніторингу якості мережевого з'єднання; Аналітичний огляд методів вимірювання якості мережі; Обґрунтування критеріїв автоматичного перемикання інтерфейсу; Огляд і вибір засобів реалізації програмного забезпечення; Аналіз засобів візуалізації метрик і результатів моніторингу; Розробка та обґрунтування архітектури застосунку; Реалізація програмного забезпечення та тестування сценаріїв перемикання

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Діаграма взаємодії для віртуального асистента; Діаграма процесу формування документів; Модель захисту віртуального асистента; Інтегрована архітектура віртуального асистента; Гексагональна архітектура віртуального асистента; ER-діаграма сутностей і зв'язків у БД віртуального асистента; Діаграма станів віртуального асистента; Блок-схема алгоритму забезпечення безпеки при формуванні текстових документів; Приклади роботи розробленого віртуального асистента для безпечного формування текстових документів

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Нестеренко В.Д.*		
Економічний розділ	*Канський М.Ю*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 05.04.2025

Керівник *Нестеренко В.Д* _____
(підпис)

Завдання прийняв до виконання *Цюсьмак М.А* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Вступ. Постановка задачі проєктування	05.04.2025	Виконав
2	Аналіз існуючих систем моніторингу	10.04.2025	Виконав
3	Вибір параметрів та методів їх вимірювання	14.04.2025	Виконав
4	Огляд і вибір засобів реалізації	21.04.2025	Виконав
5	Обґрунтування значень для рішень про перемикання	29.04.2025	Виконав
6	Розробка архітектури застосунку і структури модулів	04.05.2025	Виконав
7	Реалізація опитування інтерфейсів та збору метрик	07.05.2025	Виконав
8	Реалізація модуля логіки прийняття рішень	10.05.2025	Виконав
9	Розробка графічного інтерфейсу користувача	20.05.2025	Виконав
10	Побудова графіків якості з'єднання (ping, loss, jitter)	31.05.2025	Виконав
11	Реалізація журналювання подій і повідомлень	05.06.2025	Виконав
12	Тестування стабільності	10.06.2025	Виконав
13	Розробка питань з охорони праці та техніки безпеки	11.06.2025	Виконав
14	Підготовка мультимедійної презентації проєкту	14.06.2025	Виконав
15	Вступ. Постановка задачі проєктування	15.06.2025	Виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 58. 20 000. 00 ДП	Дипломний проект		
А4			КС 58. 20 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 58. 20 000. 00 ДП		
Розробив		Цюсьмак М.А.		23.06	Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання	Літ.	Аркуш 4 / Аркушів 75
Перевірив		Нестеренко В.Д		23.06		НДП	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КС-58	
Затверд.		Кривченко Ю.В.		23.06			

Зміст

ВСТУП

Сучасне середовище інформаційних технологій вимагає безперебійного доступу до мережі Інтернет для роботи з хмарними сервісами, участі в онлайн-зустрічах та обробки даних у реальному часі. Разом із цим дедалі частіше зустрічаються сценарії, коли на одному пристрої доступні одразу декілька мережевих інтерфейсів (Ethernet, Wi-Fi, мобільний інтернет), але штатні засоби операційної системи не забезпечують автоматичного перемикання при погіршенні якості з'єднання. У результаті користувачі стикаються з падінням швидкості, нестабільністю з'єднання та зниженням продуктивності робочих процесів.

У рамках даного дипломного проєкту розроблено програмний засіб для моніторингу мережевих адаптерів і автоматичного перемикання між ними на основі вимірювання таких параметрів, як затримка (ping), втрата пакетів і стабільність з'єднання. Проєкт реалізовано мовою C# із використанням WinForms та .NET Framework 4.8, що дозволило забезпечити гнучку взаємодію з Windows API, високу швидкодію та зручний графічний інтерфейс. Розроблений продукт здатний працювати у фоновому режимі, зберігати історію подій та налаштувань, а також надавати користувачеві інформативні сповіщення про зміни стану мережі.

Розробка цього рішення спирається не лише на технічні аспекти, а й на аналіз потреб кінцевих користувачів, які потребують надійного інтернет-з'єднання в складних або змінних умовах. Зокрема, було проведено порівняння з існуючими утилітами та встановлено, що більшість доступних рішень або занадто громіздкі в налаштуванні, або не адаптуються швидко до коливань якості мережі.

Особливу увагу в роботі приділено розробці механізму адаптивного порогового аналізу, який дозволяє враховувати не лише миттєві значення ping, але й тренди зміни якості з'єднання. Такий підхід дозволяє зменшити кількість помилкових перемикань і підвищити загальну стабільність роботи без додаткового навантаження на систему.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз проблеми та цілей

Стабільне та якісне інтернет-з’єднання є невід’ємною складовою сучасного цифрового середовища. Щодня мільйони користувачів покладаються на доступ до мережі Інтернет для виконання робочих завдань, участі в онлайн-зустрічах, перегляду контенту, спілкування або віддаленого навчання. Однак, навіть за наявності декількох доступних мереж (Wi-Fi, Ethernet, мобільні мережі), більшість пристроїв не перемикаються між ними автоматично на основі якості з'єднання. У результаті користувач стикається з проблемами:

падіння швидкості доступу до інтернету, що впливає на якість відеозв’язку, завантаження файлів або стримінг;

тимчасова втрата з'єднання, що призводить до збоїв у роботі додатків або переривання сесій;

неоптимальне використання доступних підключень, коли система залишає користувача на слабкому Wi-Fi, ігноруючи стабільне Ethernet-з’єднання.

1.1.1 Огляд проблематики

Операційна система Windows надає базові можливості перемикання мереж, але:

не враховує якість з’єднання (наприклад, затримки або втрачені пакети);

не аналізує трафік або стабільність у динаміці;

потребує втручання користувача для зміни активного підключення;

не забезпечує журналювання змін або виявлення шаблонів поганої якості.

Ці обмеження роблять процес перемикання неефективним у реальних умовах, особливо у сферах з високими вимогами до безперервності: віддалена робота, стрімінг, онлайн-ігри, IP-телефонія.

1.1.2 Мета розробки:

Створити програмне забезпечення, яке:

1. Автоматично моніторить активні мережеві інтерфейси на основі:

середнього часу відповіді (Ping),

втрати пакетів (Packet Loss),

ширини каналу (Bandwidth),

стабільності з'єднання протягом часу.

2. Приймає рішення про перемикання мережі на основі заданих порогових значень.
3. Має зручний інтерфейс, у якому користувач може:

обирати пріоритетні мережі;

задавати власні критерії перемикання;

переглядати лог змін стану з'єднання;

налаштовувати частоту перевірки.

4. Працює у фоновому режимі та не вимагає взаємодії користувача.
5. Зберігає інформацію про з'єднання для подальшого аналізу (статистика, лог-файли).

1.2 Огляд існуючих рішень

На сучасному ринку програмного забезпечення існує низка продуктів, які пропонують функції моніторингу мережі та, частково, автоматичного керування з’єднанням. Проте більшість із них мають обмежений функціонал або не орієнтовані на індивідуального користувача. В цьому розділі розглянемо найбільш поширені приклади та їхні обмеження.

1.2.1 NetBalancer (SeriousBit)

NetBalancer — це потужний інструмент для моніторингу та керування мережевим трафіком на рівні процесів і портів. Основне призначення програми полягає в наданні користувачеві гнучкого контролю над тим, як і які програми використовують мережу. Користувач може вручну виставляти пріоритети трафіку для окремих застосунків, задавати правила обмеження швидкості, переглядати реальне навантаження у графічному інтерфейсі та застосовувати фільтри за портами або процесами. Це робить NetBalancer корисним інструментом у ситуаціях, коли потрібно забезпечити стабільну роботу критично важливих додатків або контролювати інтернет-активність.

Серед ключових переваг — розширений контроль над трафіком, зручний графічний інтерфейс для візуального аналізу навантаження, а також можливість тонкого налаштування правил за допомогою фільтрації. Проте, незважаючи на ці функції, програма має і певні обмеження. Зокрема, вона не підтримує автоматичне перемикання між мережами в разі погіршення якості з'єднання, а також не має системи прийняття рішень на основі показників, таких як ping чи втрати пакетів. Іншими словами, NetBalancer працює лише в рамках одного активного з'єднання і не реагує на його деградацію.

Крім того, повний доступ до функціональності програми можливий лише за наявності платної ліцензії — безкоштовна версія має низку обмежень, що може бути суттєвим бар'єром для окремих користувачів. Таким чином, хоча NetBalancer є ефективним засобом для контролю трафіку, він не вирішує завдання динамічного управління мережевими інтерфейсами, залишаючи нішу для спеціалізованих рішень із автоматичним перемиканням на основі якості підключення.

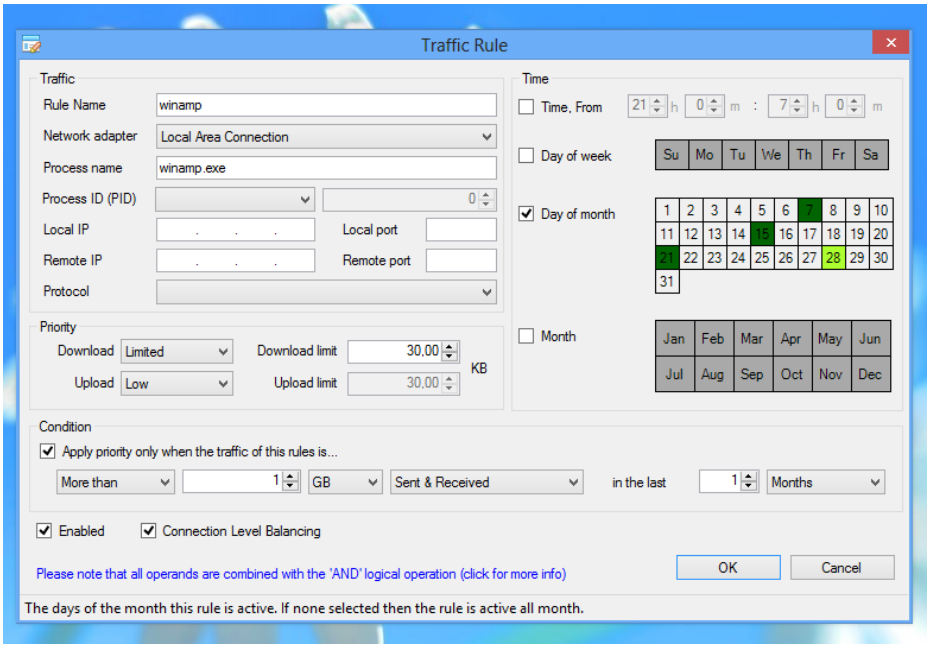


Рисунок 1.1 Головне меню NetBalancer

1.2.2 Speedify

Speedify — це кросплатформовий застосунок, який поєднує кілька доступних мережевих підключень, зокрема Wi-Fi, Ethernet та мобільний інтернет,

в єдине об'єднане з'єднання з метою підвищення стабільності, швидкості передачі даних та надійності доступу до Інтернету. Програма також вбудовано забезпечує базовий VPN-захист, шифруючи трафік і приховуючи IP-адресу користувача.

Однією з головних переваг Speedify є автоматичне поєднання доступних каналів і розподіл трафіку між ними для забезпечення максимальної продуктивності. При виявленні відмови або нестабільності одного з підключень система майже миттєво перемикає трафік на альтернативний інтерфейс, забезпечуючи безперервність з’єднання. Простий інтерфейс користувача, наявність мобільних версій для iOS та Android, а також підтримка кількох платформ роблять програму доступною для широкого кола користувачів.

Втім, рішення має і певні обмеження. Головним недоліком є те, що робота застосунку обов’язково передбачає використання VPN-з’єднання, що не завжди бажано або дозволено, особливо в корпоративних мережах або в середовищах, де VPN фільтрується або обмежується. Крім того, Speedify працює за моделлю платної підписки, що може бути бар’єром для користувачів, які шукають безкоштовні або повністю локальні рішення. Також користувач має обмежений доступ до внутрішньої логіки програми — відсутня можливість налаштовувати критерії перемикання, аналізувати параметри якості з’єднання або розширювати функціональність. Програма має закритий код і не допускає кастомізації поведінки під потреби конкретного середовища.

Таким чином, хоча Speedify є ефективним інструментом для підвищення стабільності мережі та зручним рішенням «із коробки», він більше орієнтований на кінцевого користувача, ніж на розробника чи адміністратора, і не підходить у випадках, коли потрібен повний контроль над логікою перемикання й прозорість процесів.

1.2.3 Windows Network Settings / Windows Network Profile Priority

Операційна система Windows дозволяє вручну змінювати пріоритет мереж, або через реєстр/PowerShell, або шляхом налаштування “Metric” у властивостях мережевого адаптера. Система автоматично обирає підключення з нижчим значенням метрики.

Операційна система Windows дійсно передбачає можливість налаштування пріоритету мереж вручну — через редактор реєстру, PowerShell або шляхом встановлення параметра “Metric” у властивостях мережевого адаптера. За цією логікою система обирає підключення з найнижчим значенням метрики як основне. Такий механізм інтегрований у ОС і дозволяє реалізувати базову автоматизацію — наприклад, автоматичне підключення до Ethernet замість Wi-Fi при їх одночасній наявності.

Основною перевагою цього підходу є його доступність "із коробки", без необхідності встановлення стороннього ПЗ, а також можливість задати порядок мереж відповідно до сценарію користувача. Проте система має низку істотних обмежень, які роблять її непридатною для складніших сценаріїв або динамічного середовища. По-перше, налаштування потребують ручного втручання або знань роботи з PowerShell. По-друге, рішення базується виключно на числовому значенні метрики без урахування реальної якості підключення — таких як ping, втрати пакетів або стабільність. Крім того, система не надає механізмів моніторингу, журналювання подій або графічного інтерфейсу для користувача, що ускладнює як контроль, так і діагностику проблем.

Наявні сторонні рішення, які намагаються вирішити цю задачу, часто обмежені у функціональності. Більшість з них орієнтовані на корпоративне використання зі складною конфігурацією та інтеграцією з Active Directory, або працюють у VPN-режимі без повного контролю над фізичними інтерфейсами. Також трапляються утиліти, які не здійснюють перемикання на основі параметрів якості мережі, а лише фіксують доступність інтернету.

Ці обмеження формують явну нішу для нового програмного продукту, який поєднує простоту використання з реальним моніторингом стану мережі. Такий застосунок має бути прозорим у логіці прийняття рішень, оснащений графічним інтерфейсом, можливістю асинхронного аналізу параметрів з’єднання (ping, втрати, стабільність) та здатністю автоматично й без втручання користувача

перемикатися між мережами при погіршенні якості активного каналу. Це забезпечує комфортну роботу в динамічному середовищі та підвищує стабільність доступу до мережевих ресурсів.

Таблиця 1.1. Порівння додатків для діагностики мережі

Критерій	NetBalancer	Speedify	Windows Network Settings
Призначення	Моніторинг і керування трафіком	Об’єднання декількох каналів із VPN	Вбудоване керування пріоритетом мереж
Автоматичне перемикання мереж	не підтримує	так (виявлення відмов та миттєве перемикання)	тільки на основі фіксованої метрики
Аналіз якості з’єднання	немає системи рішень за якістю	базові перевірки стабільності	відбувається лише за значенням метрики
Контроль трафіку за процесами	підтримка фільтрації за процесами	немає деталізації	немає
Графічний інтерфейс	інформативні графіки	простий інтерфейс, мобільні додатки	тільки в панелі властивостей адаптера
Логи / історія подій	немає (лише миттєві показники)	відсутні	відсутні
Налаштування пріоритету	детальні правила	автоматичне, без ручного налаштування	через Metric або PowerShell
Ліцензія / Вартість	Платна (обмежена безкоштовна версія)	Платна підписка	Безкоштовно, входить в ОС
Відкритість коду	закритий	закритий	(вбудоване в ОС)
Ніша для нового рішення	Автоматичне перемикання за якістю, глибокий контроль	Прозора логіка роботи, без обов’язкового VPN	Динамічне перемикання без ручних налаштувань

1.3 Обґрунтування вибору мови програмування та середовища розробки

При розробці програмного забезпечення, яке здійснює моніторинг мережевих підключень і автоматичне перемикання між ними, критично важливим є вибір правильного інструментарію. До основних вимог, які впливали на вибір технологій, належать:

доступ до системних мережевих інтерфейсів і адаптерів;

можливість низькорівневого моніторингу параметрів з'єднання;

стабільна робота у фоновому режимі з мінімальним використанням ресурсів;

наявність засобів створення графічного інтерфейсу користувача (GUI);

хороша підтримка спільноти, документація, сумісність з Windows.

1.3.1 Вибір мови програмування: C#

C# було обрано як основну мову програмування для реалізації застосунку з огляду на низку технічних і практичних переваг, що безпосередньо впливають на ефективність розробки та стабільність роботи програми. Однією з ключових причин є глибока інтеграція з Windows API — мова дозволяє легко взаємодіяти з системними мережевими налаштуваннями за допомогою бібліотек .NET, а також використовувати P/Invoke для виклику низькорівневих функцій Windows, таких як керування адаптерами або отримання розширеної інформації через WMI.

C# також має розвинену екосистему з безліччю готових бібліотек, що охоплюють типові задачі для такого типу програм. Зокрема, бібліотеки для виконання ping-запитів, взаємодії з WMI, запуску команд PowerShell або Netsh, а також запису логів значно спрощують розробку і скорочують час реалізації. Це дозволяє зосередитися на бізнес-логіці замість низькорівневої реалізації кожного функціонального модуля з нуля.

Інструментарій Visual Studio, який є рідним середовищем розробки для C#, забезпечує зручний цикл створення застосунків — включно з візуальним конструктором інтерфейсів (WinForms/WPF), потужним відлагоджувачем і

засобами для побудови установників, що особливо важливо для розповсюдження десктопних програм.

Окрема перевага полягає у підтримці створення як традиційних WinForms, так і сучасніших WPF-інтерфейсів, що дозволяє адаптуватися під складність проєкту та вимоги до дизайну. C# також демонструє високу продуктивність у фоновому режимі, перевершуючи багато інтерпретованих мов, таких як Python, що робить його оптимальним для постійного моніторингу без надмірного навантаження на систему.

Крім того, C# має вбудовану підтримку асинхронного програмування через async/await, що є критично важливим для забезпечення плавної роботи інтерфейсу під час виконання мережевих операцій. Усе це робить мову C# логічним і технологічно виправданим вибором для реалізації системи автоматичного перемикання мереж.

1.3.2 Вибір середовища розробки: WinForms + .NET Framework

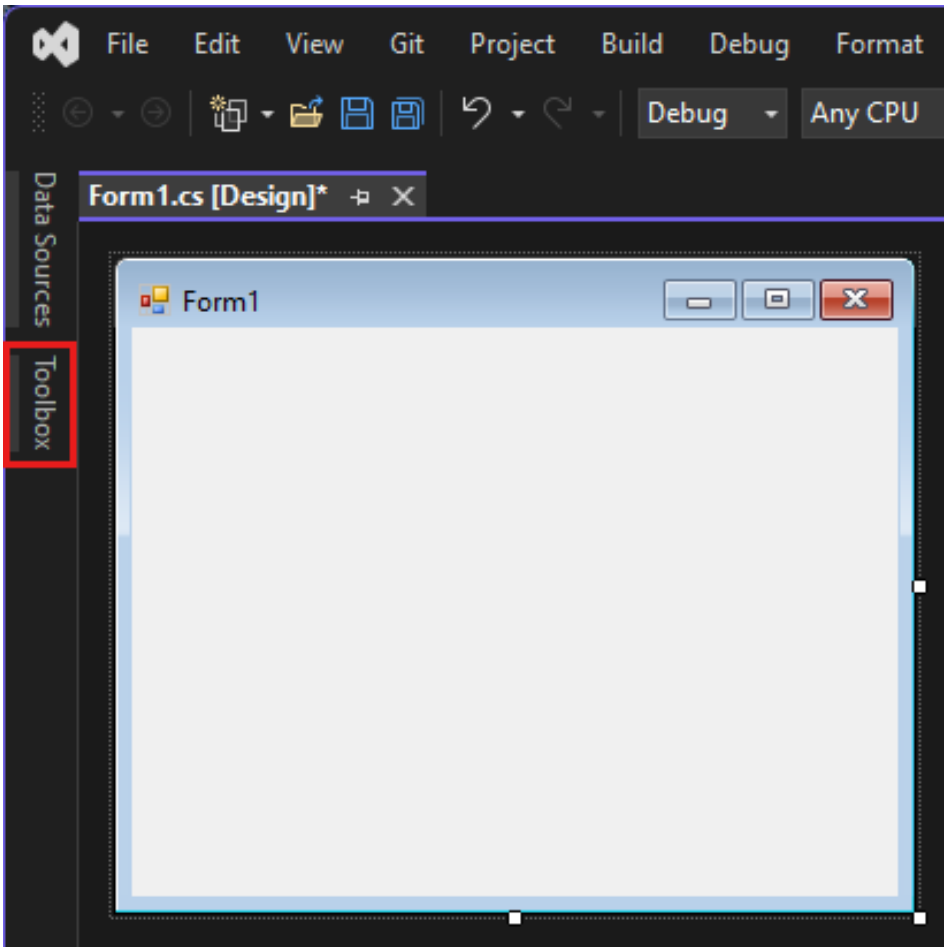


Рисунок 1.2 Visual studio з Початковою формою

Для реалізації графічного інтерфейсу користувача було обрано платформу WinForms, яка оптимально підходить для створення класичних настільних застосунків у середовищі Windows. Основними причинами такого вибору стали простота реалізації GUI, швидкість побудови форм, можливість безпосередньої інтеграції елементів керування (таблиць, кнопок, випадаючих списків тощо) і зручна система обробки подій. На відміну від WPF, WinForms не потребує вивчення XAML, що значно зменшує поріг входу для розробника та прискорює розробку інтерфейсу, оскільки вся логіка, розмітка та обробники знаходяться в одному кодовому контексті.

В якості платформи виконання було обрано .NET Framework версії 4.8, оскільки ця версія широко підтримується всіма актуальними редакціями Windows і не вимагає встановлення додаткових компонентів або залежностей від нових середовищ, таких як .NET Core або .NET 5/6+. Крім того, саме .NET Framework 4.8 включає в себе повний набір API та класів для роботи з мережею, взаємодії з WMI (Windows Management Instrumentation), управління процесами та адаптерами, що критично необхідно для реалізації функцій автоматичного моніторингу та перемикання мережевого з’єднання.

Альтернативна ідея реалізації застосунку на Python була відхилена, оскільки попри його популярність та наявність великої кількості бібліотек, робота з системними мережевими інтерфейсами в Windows викликає низку проблем. Зокрема, інтеграція з WMI або Network Interface Control у Python є складною, потребує сторонніх модулів і часто супроводжується нестабільністю, що є неприпустимим для програми, яка повинна працювати у фоновому режимі з високим рівнем надійності.

Таким чином, поєднання WinForms для створення інтерфейсу та .NET Framework 4.8 як платформи виконання забезпечує оптимальний баланс між стабільністю, функціональністю та швидкістю розробки, повністю відповідаючи вимогам до проєкту системи автоматичного керування мережевими з’єднаннями.

1.3.3 Альтернативи, що розглядалися

Аналіз альтернативних технологій показав, що хоча такі платформи, як Python із Tkinter або PyQt, пропонують високу гнучкість і швидку розробку, вони мають суттєві недоліки в контексті роботи з драйверами мережевих адаптерів та взаємодії з системними інтерфейсами Windows. Реалізація доступу до WMI, зміни налаштувань мережі або роботи з привілеями в Python вимагає використання нестабільних або погано документованих бібліотек, що створює ризики для надійності застосунку.

Java, хоча й має сильну кросплатформеність, не забезпечує достатнього рівня підтримки для низькорівневої взаємодії з системою Windows. Робота з мережевими інтерфейсами, PowerShell, чи зміна системних налаштувань мережі вимагає використання JNI або сторонніх бібліотек, що ускладнює підтримку і робить додаток менш прозорим.

Electron/JavaScript, у свою чергу, добре підходить для створення веб-орієнтованих настільних інтерфейсів, однак має суттєвий недолік — високе споживання ресурсів. Для програми, яка повинна працювати у фоновому режимі з мінімальним впливом на систему, використання Node.js, Chromium і всього середовища Electron є надмірним і неефективним рішенням.

У зв’язку з цим було прийнято рішення використовувати C# у поєднанні з WinForms та .NET Framework 4.8, що забезпечує оптимальний баланс між функціональністю, стабільністю та продуктивністю. Такий стек технологій гарантує повноцінну інтеграцію з Windows API, легкий доступ до WMI, можливість асинхронного виконання системних операцій та просту побудову графічного інтерфейсу — усе це критично важливо для реалізації стабільного та адаптивного застосунку для автоматичного керування мережевими з’єднаннями.

1.4 Опис вимог до застосунку

1.4.1 Загальна постановка задачі

Сучасні комп’ютерні системи можуть мати декілька одночасно доступних мережевих інтерфейсів, таких як Ethernet, Wi-Fi або навіть віртуальні адаптери. Однак штатними засобами Windows перемикання між ними здебільшого

відбувається вручну або відповідно до фіксованого пріоритету. У випадках, коли активна мережа втрачає якість (високий ping, втрата пакетів), операційна система не вживає жодних дій для автоматичного перемикання. Це призводить до проблем зі з'єднанням, втрати сесій у месенджерах, зависання відеозв'язку тощо.

Завданням даного програмного забезпечення є створення системи, що безперервно аналізує доступні мережеві підключення, оцінює їхню якість та, за необхідності, автоматично перемикає активне з'єднання на альтернативне — стабільніше.

1.4.2 Цілі та функціонал

Основною функцією застосунку є забезпечення користувача стабільним інтернет-з'єднанням шляхом моніторингу стану мережевих адаптерів і виконання автоматичного перемикання між ними на основі заданих критеріїв. Для цього застосунок має реалізовувати наступний функціонал:

Робота застосунку для автоматичного перемикання мережевого з'єднання побудована як послідовність взаємопов'язаних етапів, що забезпечують надійний моніторинг і керування адаптерами на основі реальних показників якості мережі. Основу функціональності становить періодичне виконання ping-запитів до одного або кількох фіксованих зовнішніх ресурсів, таких як 8.8.8.8 (Google DNS) або 1.1.1.1 (Cloudflare). Це дозволяє з високою точністю вимірювати середній час відповіді (RTT), оцінювати відсоток втрати пакетів, перевіряти наявність IP-адреси та впевнено визначати, чи є активне з'єднання з мережею Інтернет.

Для кожного адаптера програма вимірює ці параметри окремо, що дозволяє побачити не тільки загальний стан системи, а й стан кожного інтерфейсу окремо. Після кожного циклу опитування результати порівнюються з пороговими значеннями, які користувач може самостійно налаштувати. Наприклад, якщо допустимий рівень ping складає не більше 150 мс, а втрати не перевищують 5%, то лише адаптери, які відповідають цим критеріям, вважаються придатними до використання. Якщо кілька вимірювань поспіль демонструють відхилення — модуль прийняття рішень фіксує погіршення якості активного підключення.

У цей момент програма перевіряє всі доступні альтернативні інтерфейси на відповідність критеріям стабільності. Якщо знайдено краще з’єднання — застосовується рішення про перемикання, враховуючи виставлений пріоритет (наприклад, Ethernet вищий за Wi-Fi) та поточні показники. Саме перемикання виконується через зміну метрики маршрутизації (interface metric) за допомогою PowerShell або Netsh. За потреби нестабільний адаптер тимчасово вимикається, а активний шлюз пере означається для правильного маршруту.

Усі зміни миттєво відображаються в графічному інтерфейсі. Користувач бачить тип активної мережі, її технічні параметри (ping, втрати, IP-адреса), а також індикатор стану з’єднання — зелений, жовтий або червоний, залежно від якості. В окремому блоці виводиться історія подій, що дозволяє простежити, коли й чому відбувалося перемикання.

Логування є невід’ємною частиною системи: кожна подія, включаючи виявлення проблем, рішення про перемикання, його результат (успішний або ні), а також усі помилки, записується у файл з мітками часу. Це дає змогу зручно аналізувати поведінку системи й виявляти причини можливих збоїв.

Усі налаштування, включаючи критерії перевірки, список адаптерів з пріоритетами та інтервал моніторингу, зберігаються між сесіями запуску. Завдяки цьому користувачеві не потрібно щоразу повторно конфігурувати систему. Застосунок працює у фоновому режимі з мінімальним навантаженням на ресурси системи, що дозволяє йому бути постійно активним без помітного впливу на продуктивність. Крім того, реалізована можливість автозапуску разом із завантаженням Windows через відповідні записи в реєстрі або task scheduler, що гарантує надійний старт і готовність до моніторингу одразу після запуску системи.

1.4.3 Вимоги до середовища виконання

Розроблений програмний застосунок орієнтований на сумісність із операційною системою Windows версії 10 або новішої. Такий вибір зумовлений тим, що Windows 10 залишається однією з найпоширеніших ОС серед користувачів як у корпоративному, так і в домашньому середовищі. Вона

забезпечує сучасні засоби керування мережею, підтримку багатозадачності та інтеграцію з оновленими бібліотеками API. Зважаючи на ці чинники, для розробки було обрано платформу .NET Framework, яка є надійною технологічною основою для створення застосунків під Windows. Вона надає розробнику доступ до широкого спектру системних можливостей, дозволяє ефективно працювати з апаратними та мережевими ресурсами й водночас гарантує стабільну роботу програми в межах операційної системи.

Мовою реалізації обрано C#, що є сучасною об'єктно-орієнтованою мовою програмування, тісно інтегрованою з .NET Framework. Ця мова поєднує в собі зручний синтаксис, високу швидкість розробки, безпечне керування пам’яттю та потужні засоби для обробки подій, взаємодії з інтерфейсом користувача та роботи з потоками. Саме завдяки можливостям C# і .NET Framework стало можливим реалізувати всі необхідні функції застосунку, пов’язані з моніторингом мережевих інтерфейсів, збиранням технічних параметрів з’єднання, динамічною обробкою даних і оперативним відображенням результатів роботи у зручному графічному середовищі.

Для створення графічного інтерфейсу користувача було використано технологію Windows Forms, яка є одним із найпоширеніших засобів розробки десктопних застосунків у середовищі .NET Framework. Ця технологія забезпечує підтримку чіткої логіки обробки подій, широкого спектра візуальних компонентів і можливість швидкої розробки інтуїтивно зрозумілого інтерфейсу. Під час проєктування інтерфейсу було враховано сучасні принципи зручності використання (usability) та доступності, що дозволяє користувачу легко орієнтуватися в програмі та ефективно взаємодіяти з усіма ключовими функціями. Завдяки цьому застосунок залишається доступним навіть для користувачів без технічної підготовки.

Завдяки поєднанню перевірених технологій і сучасних принципів розробки, створений застосунок вирізняється стабільністю, високою продуктивністю та готовністю до подальшого розширення функціональності.

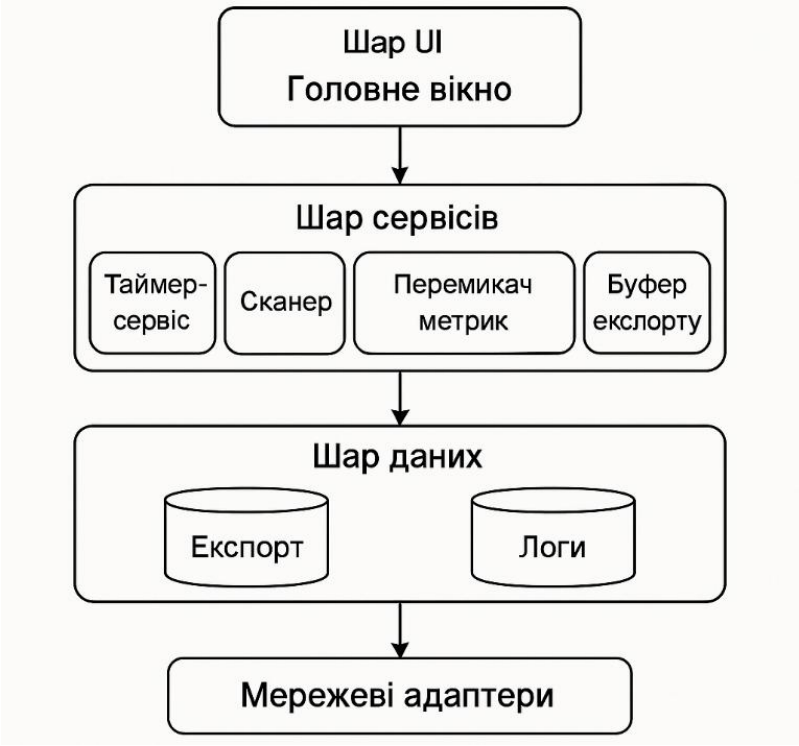


Рисунок 1.3 Відображення архітектури проєкту

Інтерфейс користувача повинен бути зрозумілим, мінімалістичним та підтримувати українську або англійську мову. Програма повинна функціонувати коректно як у вигляді активного вікна, так і згорнутою до системного трея.

1.4.4 Нефункціональні вимоги

Розроблений застосунок має забезпечувати стабільну роботу в режимі 24/7 без витоків пам’яті, зависань або критичних помилок, що можуть призвести до його зупинки. Робота у фоновому режимі повинна бути максимально оптимізованою — навантаження на процесор не повинно перевищувати 5% у середньому, щоб уникнути впливу на продуктивність системи загалом. Оновлення графічного інтерфейсу користувача має відбуватися плавно, без мерехтіння, зависань або блокування елементів керування.

Всі мережеві перевірки, зокрема виконання ping-запитів до зовнішніх хостів, повинні виконуватись асинхронно — це дозволить не блокувати головний потік програми та забезпечити її швидку реакцію на дії користувача. Надійність системи має підтримуватись через повноцінну обробку винятків і помилок. Усі важливі події та збої повинні логуватись у відповідному форматі із

зазначенням часу, типу події та деталей ситуації. Якщо помилка потребує втручання користувача, необхідно забезпечити виведення інформативного повідомлення з описом проблеми й можливими шляхами її вирішення. Такий підхід гарантує високу надійність, передбачуваність і зручність використання програми в реальних умовах.

1.4.5 Обмеження

Робота програми обмежується рамками користувацьких прав Windows. У випадках, коли система не дозволяє змінювати пріоритети інтерфейсів або вимикати адаптери без адміністративних прав, програмне забезпечення повинне запропонувати відповідні дії користувачеві або виконати команду через підвищення прав доступу.

Програма не аналізує політики корпоративної безпеки або мережеві VPN-з'єднання, а також не гарантує повної безперервності з'єднання на рівні TCP-сесій.

1.5 Архітектура та логіка роботи програми

Розроблений застосунок для автоматичного перемикання мережевого з’єднання побудований за модульною архітектурою, яка забезпечує гнучкість, масштабованість і зручність у подальшій підтримці. Модульний підхід дозволяє кожному компоненту відповідати лише за свою вузьку функціональність, з мінімальною залежністю від інших частин системи. Завдяки цьому кожен блок може бути незалежно протестований, оновлений або замінений без ризику порушити загальну логіку роботи застосунку.

В межах архітектури можна виокремити кілька основних логічних модулів. Модуль моніторингу відповідає за постійний аналіз стану мережевих адаптерів і збір технічних параметрів, таких як ping, втрати пакетів, швидкість передавання та статус підключення. Модуль прийняття рішень обробляє отримані дані, порівнюючи їх з пороговими значеннями, і вирішує, чи потрібно перемикатися на інше з’єднання. Для реалізації самого перемикання використовується модуль керування адаптерами, який здійснює увімкнення, вимкнення, зміну метрики або вибір активного адаптера через системні команди та API. Паралельно з цим

модуль логування фіксує всі значущі події для подальшого аналізу та діагностики. Збереження налаштувань користувача між сесіями забезпечується окремим модулем конфігурації, а графічний інтерфейс користувача дає змогу взаємодіяти з усіма функціями системи в зручному вигляді.

Завдяки такій структурі система залишається гнучкою до змін, придатною для розширення новими функціями та стабільною в роботі за рахунок чітко розмежованих відповідальностей між її складовими.

1.5.1 Основні компоненти системи

Модуль моніторингу мережі виконує ключову функцію безперервного збору та аналізу параметрів усіх доступних мережевих інтерфейсів. Його основне завдання полягає в регулярному оцінюванні якості з'єднання кожного адаптера, що дозволяє програмі своєчасно реагувати на погіршення зв'язку. Моніторинг охоплює кілька основних метрик: середній час відповіді (ping) до заданого контрольного хосту, такого як 8.8.8.8 (Google DNS) або cloudflare.com; відсоток втрати пакетів під час серії ping-запитів; загальний статус мережевого адаптера, включно з інформацією про його активність, наявність IP-адреси та готовність до передачі даних; а також, за потреби, орієнтовну швидкість передавання даних, яку можна визначити на основі системної статистики або спеціалізованих запитів.

Уся система перевірки параметрів мережевого з'єднання реалізована з використанням асинхронного підходу. Це означає, що всі операції збору та аналізу даних виконуються у фоновому режимі без блокування головного потоку програми. Такий підхід є критично важливим для забезпечення стабільної роботи графічного інтерфейсу користувача, особливо в умовах регулярних мережевих запитів, які можуть мати змінну тривалість у часі.

Інтервал між циклами моніторингу задається вручну в параметрах застосунку і за замовчуванням становить, наприклад, п'ять секунд. Після завершення кожного циклу виконується аналіз зібраної інформації по всіх активних мережевих інтерфейсах. Отримані дані включають значення затримки (ping), відсоток втрати пакетів, наявність IP-адреси, а також інші ключові показники стабільності з'єднання.

Для кожного адаптера формується окремий звіт, який може бути виведений у вигляді таблиці або переданий в лог-файл. Це дозволяє системі зберігати історію станів мережі, а також приймати обґрунтовані рішення щодо перемикання між інтерфейсами або сповіщення користувача про погіршення з'єднання.

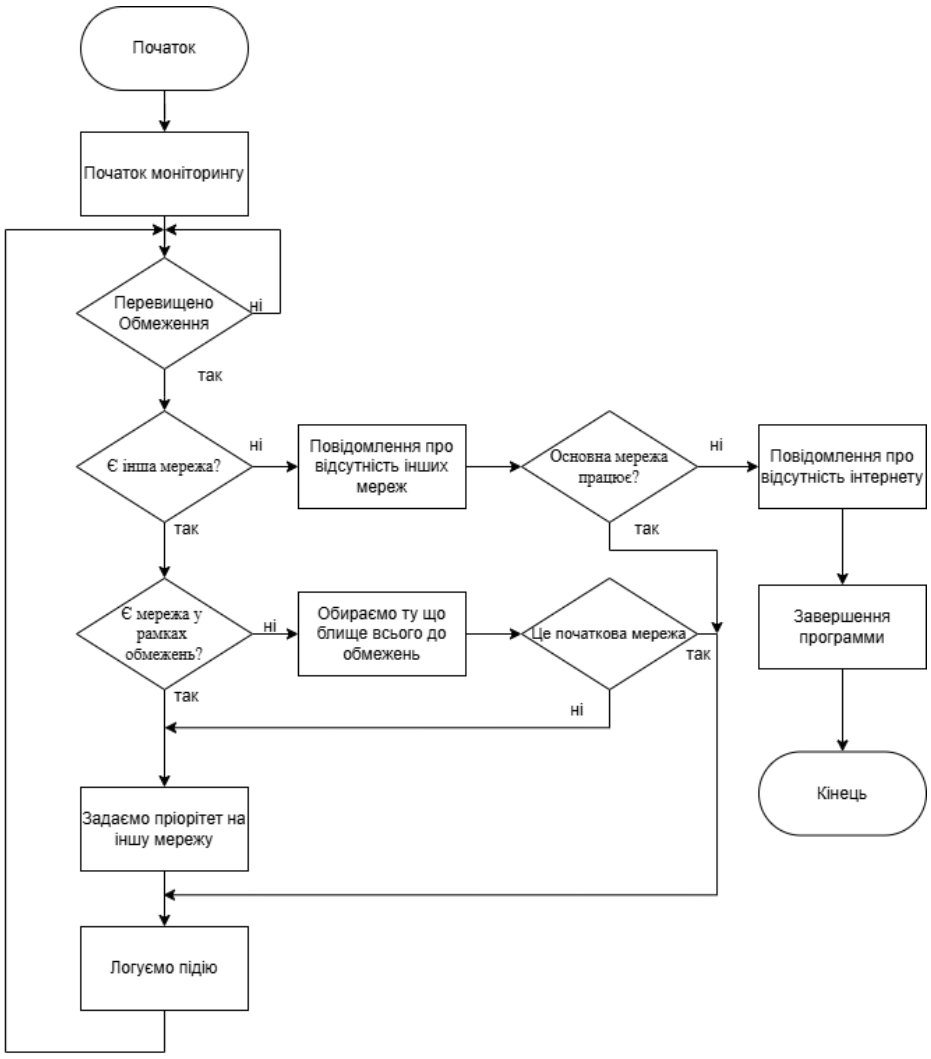


Рисунок 1.4 Загальна діаграма роботи програми

Модуль прийняття рішень є центральним елементом логіки перемикання між мережами. Його робота базується на аналізі даних, що надходять від модуля моніторингу — зокрема значень ping, рівня втрат пакетів і загальної стабільності з'єднання. Після кожного циклу опитування програма порівнює поточні параметри активного підключення з пороговими значеннями, встановленими користувачем у налаштуваннях. Якщо зафіксовано, що ping перевищує допустимий рівень, або втрати пакетів стають критичними, система активує механізм перемикання.

Рішення про зміну мережі не ухвалюється миттєво — модуль додатково враховує кілька ключових чинників. По-перше, береться до уваги встановлений пріоритет адаптерів, що дозволяє програмі обрати найбільш бажане з'єднання, якщо їх доступно кілька. По-друге, перед перемиканням обов’язково виконується перевірка стабільності альтернативного інтерфейсу, щоб упевнитися, що нове з’єднання справді є кращим і не створить ще більших проблем. По-третє, реалізовано захист від флапінгу — тобто частого й безперервного перемикання між мережами в умовах нестабільного сигналу. Система аналізує попередні перемикання, вводить затримку між ними або накладає тимчасову блокаду на повторне підключення до тієї ж мережі.

Такий підхід дозволяє досягти балансу між швидкістю реакції на погіршення зв’язку та стабільністю роботи програми, уникаючи хаотичної поведінки в динамічних умовах мережевого середовища.

Модуль керування адаптерами відповідає за безпосереднє управління мережевими інтерфейсами системи та забезпечує технічну основу для автоматичного перемикання між з’єднаннями. Його функціональність охоплює увімкнення та вимкнення мережевих адаптерів, що реалізується за допомогою викликів через PowerShell або WMI. Такий підхід дозволяє програмно впливати на стан інтерфейсів без ручного втручання користувача.

Окрім базового керування активністю адаптерів, модуль виконує налаштування пріоритетів мереж шляхом зміни «метрики» (network metric) через командний інтерфейс Netsh. Це дозволяє визначати, який адаптер буде використовуватися системою як основний у разі наявності кількох активних з’єднань. Визначення поточної активної мережі здійснюється шляхом запитів до системних API — зокрема через Windows API або клас Win32_NetworkAdapterConfiguration, що надає доступ до розширеної інформації про стан і параметри кожного інтерфейсу.

Оскільки частина команд, пов’язаних із керуванням мережею, вимагає адміністративних прав, модуль побудований таким чином, що в разі потреби він ініціює виконання окремих операцій з підвищеними привілеями. Це реалізується

через запуск окремих процесів або команд з правами адміністратора, що дозволяє зберігати безпеку та стабільність роботи програми, не порушуючи політик доступу Windows. Така архітектура забезпечує повний контроль над адаптерами та дозволяє динамічно змінювати активне підключення на основі результатів моніторингу мережевої якості.

Графічний інтерфейс користувача реалізовано з використанням технології WinForms, що забезпечує швидкий і зручний доступ до основних функцій програми. Основне вікно містить індикатор активного підключення, де виводиться назва поточної мережі, IP-адреса, а також значення ping у режимі реального часу. Це дозволяє користувачеві миттєво оцінити стан з'єднання без потреби звертатися до системних утиліт.

У межах інтерфейсу також передбачено панель налаштувань, де користувач може змінювати параметри перемикання мережі — зокрема граничне значення ping, допустимий відсоток втрат пакетів і таймаути, які впливають на алгоритм прийняття рішень системою моніторингу. Окремий розділ відведено для журналу подій: лог відображається у вигляді прокручуваного списку з можливістю фільтрації записів за датою, що значно полегшує перегляд історії з'єднань і діагностику проблем.

Інтерфейс також містить перемикач, який дозволяє активувати або вимкнути автоматичний запуск програми при старті операційної системи Windows. Для зручності користувача передбачено вкладку "Про програму", де відображається поточна версія застосунку та інша довідкова інформація.

Усі елементи GUI динамічно оновлюються на основі подій, що надходять із модуля моніторингу. Це забезпечує актуальність інформації без потреби ручного оновлення та дозволяє підтримувати високий рівень інтерактивності й зручності під час роботи з програмою.

Модуль логування відповідає за фіксацію всіх важливих подій, що відбуваються під час роботи програми, забезпечуючи повну історію змін і діагностики. До таких подій належать виявлення погіршення якості з'єднання, перемикання між мережами, помилки системного рівня, а також увімкнення або

вимкнення мережевих адаптерів. Кожен запис у лог-файлі супроводжується точною міткою часу, що дозволяє відстежувати хронологію подій.

Лог-файл містить структуровану інформацію, включаючи тип події (наприклад, “Switch”, “Warning”, “Error”, “ConnectionLost”), назву або ідентифікатор адаптера, що був задіяний під час події, а також технічні параметри з’єднання — наприклад, ping, втрати пакетів чи рівень джітера на момент перемикання. У разі виникнення помилок або попереджень запис додається з відповідною позначкою, що полегшує аналіз проблем під час налагодження або перевірки працездатності системи.

Завдяки цьому модулю забезпечується прозорість у роботі застосунку, можливість зворотного аналізу мережевих збоїв і зручність для користувачів та розробників при діагностиці нештатних ситуацій.

Логи зберігаються у форматі *.txt або *.csv у локальній директорії користувача.

Модуль збереження налаштувань реалізовано за допомогою вбудованого механізму конфігурації .NET, зокрема через файл Settings.settings. Цей підхід дозволяє надійно зберігати параметри між сесіями програми без потреби у створенні окремих файлів або баз даних. Усі користувацькі значення — зокрема пороги якості з’єднання (наприклад, максимальний ping, втрати пакетів), порядок пріоритетів мережевих адаптерів, інтервали опитування та інші опції — автоматично зчитуються під час запуску програми та застосовуються до відповідних компонентів. Таким чином, користувачеві не потрібно повторно вводити налаштування при кожному запуску, що забезпечує зручність і послідовність у роботі застосунку. У разі зміни параметрів під час роботи, оновлені значення зберігаються через виклик відповідного методу збереження конфігурації, що гарантує актуальність збережених даних.

1.5.2 Потік роботи програми

Після запуску програма автоматично ініціалізує всі наявні мережеві адаптери та перевіряє, які інтерфейси наразі активні. На основі зібраної інформації визначається основне підключення, що використовується за

замовчуванням, і паралельно починається збір статистики для всіх доступних інтерфейсів. У задані проміжки часу програма надсилає запити ping до контрольного хосту, окремо для кожного адаптера, щоб оцінити якість з'єднання в реальному часі.

Якщо система виявляє погіршення параметрів основного підключення — наприклад, високий ping, втрати пакетів або нестабільність — запускається процедура перевірки якості альтернативних мереж. У разі якщо хоча б одна з них має кращі характеристики, програма автоматично виконує перемикання з'єднання на більш стабільну мережу. Факт перемикання реєструється у лог-файлі з детальним описом події, включно з часом, параметрами з'єднань і причиною зміни.

Після цього інтерфейс користувача оновлюється автоматично, відображаючи актуальний стан мережі, а моніторинг продовжується у фоновому режимі без необхідності втручання з боку користувача. Такий підхід забезпечує безперервний контроль якості з'єднання та дозволяє динамічно адаптуватися до змін у мережевому середовищі.

1.5.3 Взаємодія компонентів

Комунікація між модулями здійснюється через подієву модель (event-driven), що дозволяє уникнути блокувань інтерфейсу. Компоненти не залежать один від одного безпосередньо, а взаємодіють через делегати та callback-методи. Це спрощує підтримку коду та дозволяє за потреби легко замінити окремий модуль (наприклад, замінити логування у файл на логування в базу даних).

1.6 Опис головних модулів і реалізації

Структура розробленого застосунку ґрунтується на чіткому розділенні відповідальностей між окремими логічними модулями. Це дозволяє забезпечити гнучкість, масштабованість, простоту супроводу коду, а також полегшує відлагодження та тестування системи. Нижче подано опис основних модулів, які формують архітектурну основу застосунку.

1.6.1 Модуль моніторингу мережевих підключень

Цей модуль є ядром усієї системи. Він відповідає за періодичний збір інформації про поточний стан усіх доступних мережевих адаптерів. Реалізація включає виконання ping-запитів до заздалегідь визначених IP-адрес (наприклад, 8.8.8.8 або 1.1.1.1). Результати кожного запиту аналізуються для визначення таких показників:

- середній час відповіді (Round Trip Time),
- наявність або відсутність втрат пакетів,
- стабільність відгуку протягом останніх кількох циклів,
- стан мережевого інтерфейсу (активний / неактивний),
- наявність IP-адреси та шлюзу.

Модуль реалізовано у вигляді окремого класу NetworkMonitorService, що працює у власному потоці з можливістю асинхронної обробки.

Реалізовано класом NetworkMonitorService, що запускає фоновий таск і викликає колбек з DTO InterfaceMetrics.

```
public class NetworkMonitorService : IDisposable
{
   private readonly string[] _hosts = { "8.8.8.8", "1.1.1.1" };
   private readonly TimeSpan _interval = TimeSpan.FromSeconds(5);
   private CancellationTokenSource _cts;
   public event Action<InterfaceMetrics> OnMetrics;
   public void Start()
   {
      _cts = new();
      Task.Run(async () =>
      {
         while (!_cts.Token.IsCancellationRequested)
         {
            foreach (var ni in NetworkInterface.GetAllNetworkInterfaces()
                     .Where(n => n.OperationalStatus==OperationalStatus.Up))
            {
               OnMetrics?.Invoke(await MeasureAsync(ni, _hosts, _cts.Token));
            }
            await Task.Delay(_interval, _cts.Token);
         }
      });
   }
   public void Stop() => _cts.Cancel();
```

```
    private async Task<InterfaceMetrics> MeasureAsync(
        NetworkInterface ni, string[] hosts, CancellationToken ct)
    {
        var ping = new Ping();
        var times = new List<long>();
        int sent=0, recv=0;

        foreach (var h in hosts)
            for (int i=0; i<2; i++) // по 2 пінги на хост
            {
                if (ct.IsCancellationRequested) break;
                sent++;
                var r = await ping.SendPingAsync(h, 500);
                if (r.Status==IPStatus.Success) { recv++; times.Add(r.RoundtripTime); }
            }

        var avg = times.Any() ? times.Average() : double.MaxValue;
        var loss = sent>0 ? (1- (double)recv/sent)*100 : 100;
        var stable = times.Distinct().Count() <= 1;

        var props = ni.GetIPProperties();
        return new InterfaceMetrics
        {
            InterfaceName = ni.Name,
            AvgRtt        = avg,
            PacketLoss    = loss,
            IsStable      = stable,
            IsUp          = true,
            HasIp                                                                    =
props.UnicastAddresses.Any(a=>a.Address.AddressFamily==System.Net.Sockets.Ad
dressFamily.InterNetwork),
            HasGateway    = props.GatewayAddresses.Any(),
            Timestamp     = DateTime.Now
        };
    }

    public void Dispose() => Stop();
}
public record InterfaceMetrics(
    string InterfaceName,
    double AvgRtt,
    double PacketLoss,
    bool IsStable,
```

```
    bool IsUp,
    bool HasIp,
    bool HasGateway,
    DateTime Timestamp
);
```

1.6.2 Модуль аналізу якості та логіки перемикання

Цей компонент відповідає за прийняття рішень щодо зміни активного мережевого інтерфейсу. Він працює на основі результатів, що надходять із модуля моніторингу. Алгоритм передбачає визначення порушення стабільності з'єднання (наприклад, якщо три поспіль перевірки дають ping понад 200 мс або втрата пакетів перевищує 10%).

Після виявлення порушення система:

1. Оцінює альтернативні адаптери.
2. Перевіряє їхній поточний стан і параметри.
3. У разі виявлення стабільного інтерфейсу ініціює процедуру перемикання.

Модуль ConnectionDecisionEngine реалізує логіку, яка враховує виставлені пріоритети адаптерів (на основі user-defined ranking), уникає надмірно частих перемикань (захист від флапінгу), а також записує подію у лог.

```
public class ConnectionDecisionEngine
{
    private readonly int _pingTh = 200, _lossTh = 10;
    private readonly TimeSpan _minInterval = TimeSpan.FromSeconds(30);
    private DateTime _lastSwitch = DateTime.MinValue;
    private readonly Dictionary<string, List<InterfaceMetrics>> _hist
        = new();
    public event Action<string> SwitchTo;
    public event Action<string> Log;
    public void Process(InterfaceMetrics m)
    {
        if (!_hist.ContainsKey(m.InterfaceName))
            _hist[m.InterfaceName] = new List<InterfaceMetrics>();
        var h = _hist[m.InterfaceName];
        h.Add(m);
        if (h.Count > 3) h.RemoveAt(0);
        if (h.Count < 3) return;
        bool unstable = h.All(x => x.AvgRtt > _pingTh || x.PacketLoss > _lossTh);
        if (!unstable || DateTime.Now - _lastSwitch < _minInterval) return;
        var best = _hist
```

```
        .Where(kv => kv.Key != m.InterfaceName && kv.Value.Last().AvgRtt <=
_pingTh && kv.Value.Last().PacketLoss <= _lossTh)
        .OrderBy(kv => kv.Value.Last().AvgRtt + kv.Value.Last().PacketLoss)
        .Select(kv => kv.Key)
        .FirstOrDefault();
    if (best != null)
    {
        Log?.Invoke($"Switch {m.InterfaceName}→{best}");
        SwitchTo?.Invoke(best);
        _lastSwitch = DateTime.Now;
        h.Clear();
    }
```

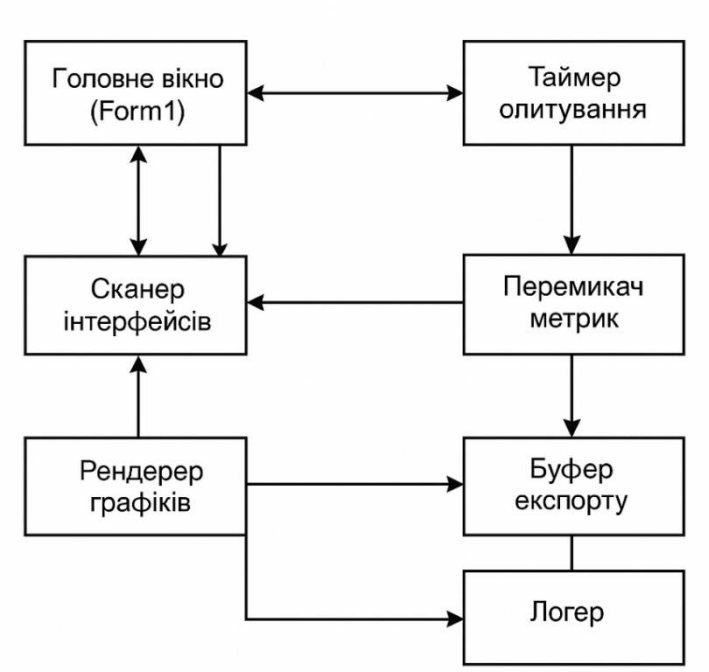


Рисунок 1.5 Діаграма взаємодії модулів

1.6.3 Модуль керування адаптерами

Цей модуль виконує ключову роль у забезпеченні безпосередньої взаємодії з мережевими інтерфейсами операційної системи. Його функціональність охоплює керування мережевими адаптерами, зокрема можливість їхнього програмного увімкнення або вимкнення, а також зміну пріоритетів маршрутів шляхом коригування метрик доступу до мережі. Для цього використовується набір системних інструментів, таких як Netsh або PowerShell, що дозволяють впливати на таблицю маршрутизації. Окремо модуль відповідає за визначення поточного активного маршруту до інтернету, тобто обчислення default gateway, а

також може ініціювати оновлення IP-конфігурації у випадках, коли це необхідно для підтримання стабільного з’єднання.

У середовищі C# зазначені дії реалізовано за допомогою класу System.Management, який забезпечує доступ до Windows Management Instrumentation (WMI), а також через ProcessStartInfo, що дозволяє запускати зовнішні команди у межах окремих процесів. Зокрема, для зміни метрики маршруту застосунок формує відповідну команду та передає її до PowerShell або Netsh, де вона виконується у фоновому режимі з поверненням результату до основного модуля.

У ситуаціях, коли для виконання певних дій потрібні підвищені права доступу, програма автоматично ініціює запуск обгортки з адміністративними привілеями. Це досягається шляхом встановлення параметра UseShellExecute = true і задання значення Verb = "runas", що змушує систему запитати підтвердження користувача на запуск із правами адміністратора. Такий підхід забезпечує гнучке та безпечне управління критичними мережевими параметрами в рамках політики безпеки Windows.

1.6.4 Модуль логування

Цей компонент фіксує всі значущі події у текстовий лог-файл. До таких подій належать:

- запуск програми;
- запуск або зупинка моніторингу;
- результат ping-перевірки;
- погіршення якості активного з’єднання;
- переключення мережі;
- помилки та винятки.

Формат запису містить час події, тип події, ім’я адаптера, деталі (ping, IP, причина тощо). Для створення логів використовується клас StreamWriter у поєднанні з власною структурою LogEntry.

```
public class Logger : IDisposable
{
```

```
    private readonly StreamWriter _writer;
    private readonly object _lock = new();

    public Logger(string filePath)
    {
        _writer = new StreamWriter(filePath, append: true) { AutoFlush = true };
    }

    public void Log(string eventType, string adapter, string details)
    {
        var entry = new LogEntry
        {
            Timestamp = DateTime.Now,
            EventType = eventType,
            Adapter   = adapter,
            Details   = details
        };
        lock (_lock)
        {
            _writer.WriteLine(entry.ToString());
        }
    }

    public void Dispose() => _writer.Dispose();
}
```

1.6.5 Модуль користувацьких налаштувань

Модуль конфігурації відповідає за збереження та завантаження користувацьких налаштувань, необхідних для коректної роботи системи моніторингу. До таких налаштувань належать порогові значення затримки (ping) та втрати пакетів (packet loss), інтервал між циклами перевірки, список доступних мережевих адаптерів разом із визначеними для них пріоритетами, стан прапорця автозапуску програми при старті системи, а також IP-адреса або доменне ім’я хосту, до якого здійснюється перевірка з’єднання.

Для реалізації збереження цих параметрів використовується вбудований механізм .NET Settings — зокрема, клас Properties.Settings.Default, що надає можливість зберігати пари ключ–значення у форматі XML у конфігураційному файлі користувача. Такий підхід дозволяє зручно зчитувати та оновлювати

налаштування без необхідності ручного редагування файлів, забезпечуючи стабільність і простоту використання системи.

1.6.6 Графічний інтерфейс (GUI)

Інтерфейс побудований на базі WinForms. Він складається з таких основних елементів:

- головне вікно з табличним відображенням стану адаптерів;
- панель параметрів для налаштування критеріїв перемикання;
- вкладка "Журнал подій" з можливістю фільтрації;
- індикатор статусу з'єднання (візуальна шкала або колір);
- системний трей з контекстним меню для швидкого доступу.

Користувач має змогу вручну вимкнути/увімкнути моніторинг, змінити налаштування, переглянути лог або вийти з програми. Комунікація між GUI та логікою реалізована через події та інтерфейси, щоб уникнути прямої прив’язки компонентів.

1.7 Алгоритм визначення якості з’єднання

Для реалізації ефективного перемикання між мережами на основі об’єктивної оцінки якості з’єднання було розроблено алгоритм, який виконує аналіз кількох ключових показників. Алгоритм працює циклічно, з заданим інтервалом (наприклад, кожні 5 секунд), і обробляє дані для кожного активного мережевого інтерфейсу.

1.7.1 Основні параметри оцінки

У системі автоматичного моніторингу мережевих з’єднань ключову роль відіграють кілька технічних метрик, які дозволяють об’єктивно оцінити якість роботи адаптерів і своєчасно реагувати на погіршення зв’язку.

Ping (RTT) — це базовий показник затримки, який відображає середній час проходження ICMP-пакету до контрольного вузла (наприклад, 8.8.8.8) і назад. Чим менше значення RTT, тим стабільніше та швидше працює з’єднання. Типовим вважається поріг у 150 мс, однак він може бути змінений користувачем відповідно до своїх потреб або умов середовища. Моніторинг виконується

періодично, а результати порівнюються з установленим порогом, щоб оцінити якість поточного підключення.

Втрати пакетів (Packet Loss) — ще один критично важливий показник, що фіксує нестабільність з'єднання. Якщо в серії із 5 ping-запитів хоча б один не отримує відповіді, фіксується втрата. У разі, якщо втрати повторюються протягом кількох послідовних перевірок (наприклад, 3 цикли поспіль), адаптер автоматично класифікується як нестабільний. Такий підхід дозволяє уникнути фальшивих спрацьовувань при короткочасних збоях і зосередитись на повторюваних проблемах.

Доступність шлюзу (Gateway Reachability) — допоміжний критерій, який перевіряє, чи пристрій має фізичний доступ до маршрутизатора або точки доступу. Це дозволяє точно визначити, чи є мережеве підключення дійсно активним, навіть якщо інтернет недоступний. Перевірка виконується через локальну IP-адресу шлюзу, яка зчитується з системних налаштувань мережі.

IP-конфігурація адаптера — включає перевірку наявності дійсної IP-адреси. Якщо система отримала адресу з діапазону APIPA (169.254.x.x), це означає, що DHCP-сервер недоступний і адаптер втратив зв'язок із мережею. Також враховується загальний стан маршрутизації — чи наявний маршрут до Інтернету через поточний інтерфейс.

Сукупність цих параметрів дозволяє системі оцінити не лише формальну доступність мережі, а й її реальну якість. Це критично важливо для ухвалення обґрунтованого рішення щодо перемикання на альтернативне з'єднання без втручання користувача.

1.7.2 Логіка обробки

Алгоритм прийняття рішення про перемикання мережі побудований на багатоступеневій логіці, що дозволяє поєднати точність оцінки якості з'єднання з уникненням фальшивих спрацювань і циклічного перемикання. Першим етапом є збір результатів, під час якого для кожного активного мережевого адаптера виконується серія з 4–5 ping-запитів до контрольного вузла. На основі відповідей розраховується середній час затримки (RTT) і фіксується кількість втрат пакетів.

Після збору даних виконується оцінка якості: якщо середній ping перевищує встановлений поріг (наприклад, 150 мс), а втрати перевищують 10%, адаптер позначається як потенційно нестабільний. Проте перемикання не відбувається миттєво — у системі передбачено механізм накопичення спостережень. Тільки після кількох послідовних виявлень нестабільності (наприклад, у трьох циклах моніторингу підряд) адаптер офіційно визнається непридатним для використання. Це дозволяє уникнути реакцій на короткочасні збої або коливання якості, які не мають критичного значення.

У разі підтвердження нестабільності активного з'єднання програма переходить до пошуку кращої мережі. Для цього аналізуються всі доступні адаптери з урахуванням кількох критеріїв: найменше значення ping, відсутність або мінімальний рівень втрат пакетів, стабільність показників у попередніх циклах, а також пріоритет, виставлений користувачем (наприклад, Ethernet важливіший за Wi-Fi).

На основі цієї інформації приймається рішення: якщо знайдено адаптер з кращими показниками, виконується перемикання мережі — наприклад, через зміну маршрутизаційної метрики, відключення поточного адаптера або активацію альтернативного.

Для запобігання циклічному перемиканню (флапінгу) між двома адаптерами зі схожими характеристиками реалізовано механізм тайм-ауту стабілізації. Після кожного перемикання активується блокування повторного перемикання на певний час (наприклад, 30 секунд), що дозволяє стабілізувати роботу мережі та уникнути надмірної кількості переходів. Такий підхід забезпечує зважене, стабільне та передбачуване функціонування системи в реальних умовах.

1.7.3 Програмна реалізація

У коді алгоритм реалізовано у вигляді методу EvaluateConnectionQuality() у класі ConnectionDecisionEngine. Для обробки статистики застосовуються списки об'єктів типу PingResult, які містять:

- час перевірки,

- успішність відповіді,
- тривалість (у мс).

Середнє значення та відсоток втрат обчислюються за допомогою LINQ-виразів. Прийняте рішення сигналізується у головний модуль, який, у разі потреби, запускає перемикання через NetworkAdapterManager.

1.8 Інтерфейс користувача: структура, компоненти та взаємодія

Інтерфейс користувача (Graphical User Interface, GUI) — важлива складова застосунку, оскільки забезпечує зручність взаємодії, візуалізацію поточного стану з'єднання та доступ до налаштувань. Інтерфейс реалізовано на базі бібліотеки Windows Forms (WinForms), яка є частиною .NET Framework. Основною вимогою до інтерфейсу була простота, інформативність і швидке реагування без навантаження системи.

1.8.1 Загальна структура інтерфейсу

Інтерфейс програми побудований із урахуванням зручності щоденного використання та швидкого доступу до основної інформації. Він складається з кількох логічно відокремлених розділів, кожен з яких виконує свою функцію. У центрі взаємодії знаходиться головне вікно, у якому в режимі реального часу відображається стан усіх активних мережевих інтерфейсів. Тут користувач може побачити назву адаптера, тип з'єднання, IP-адресу, середній ping, відсоток втрат пакетів і індикатор якості з'єднання, що візуально сигналізує про стабільність (зелений, жовтий або червоний статус).

Окремим блоком передбачена панель налаштувань, яка дозволяє задавати ключові параметри для логіки перемикання. Користувач може вручну встановити порогові значення для ping, втрат пакетів, обрати частоту перевірки інтерфейсів, пріоритет мереж (наприклад, Ethernet вище за Wi-Fi), а також активувати або деактивувати механізм флапінг-захисту. Налаштування зберігаються між сесіями й застосовуються автоматично.

Ще один важливий елемент — вкладка журналу подій, де відображається історія дій програми: виявлення погіршення з'єднання, запуск перевірок, рішення

про перемикання, помилки, час подій тощо. Це дає змогу відслідковувати поведінку системи та виконувати діагностику без глибокого технічного аналізу.

Для зручності програма має системне меню у треї Windows, що дозволяє зменшити візуальне навантаження на робочий стіл. Після запуску застосунок автоматично згортається в трей та продовжує роботу у фоновому режимі. У будь-який момент користувач може викликати головне вікно, двічі клацнувши на іконці у системній панелі. Таке рішення дозволяє програмі бути постійно активною без нав'язливості й водночас забезпечує швидкий доступ до її функцій.

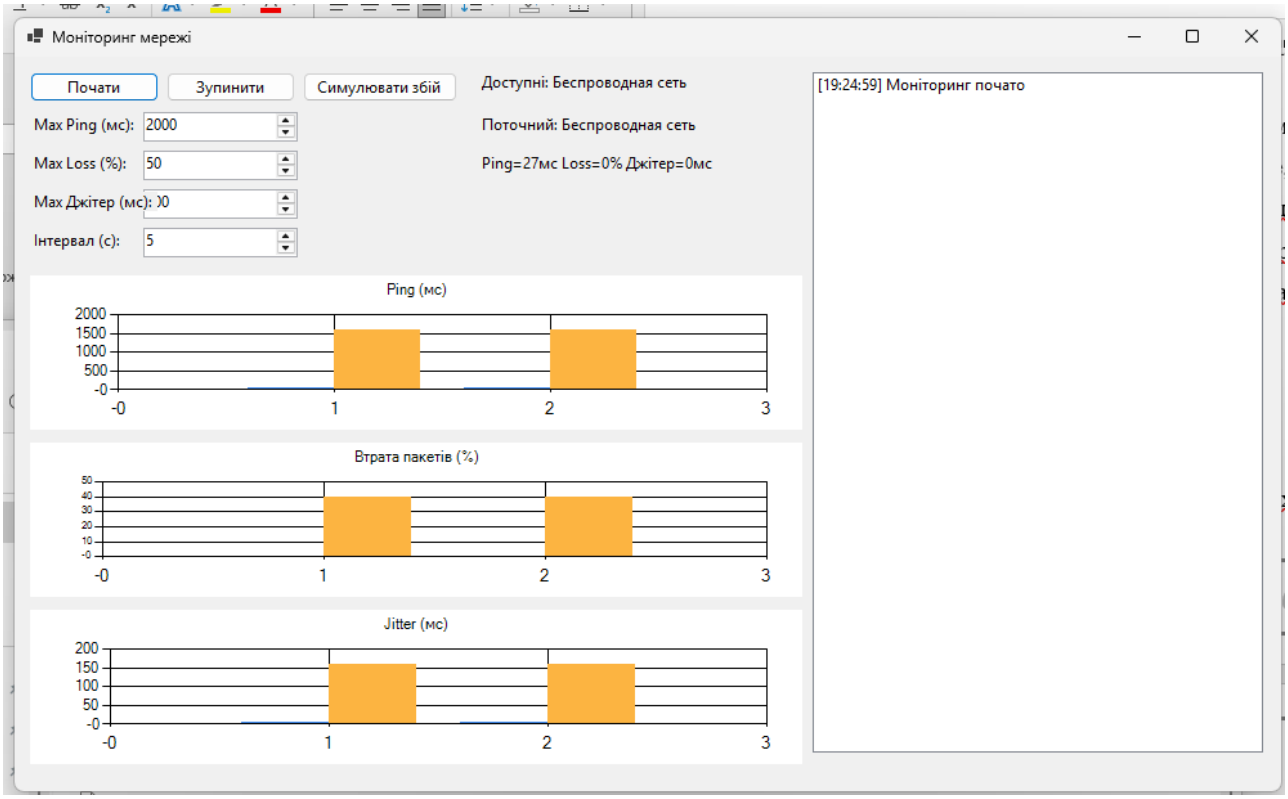


Рисунок 1.5. Загальний вигляд програми

1.8.2 Опис основних елементів GUI

Список мережевих інтерфейсів у програмі реалізований у вигляді зручного табличного представлення з використанням компонента DataGridView, що дозволяє динамічно виводити й оновлювати інформацію про кожен адаптер у режимі реального часу. Кожен рядок таблиці відповідає одному мережевому інтерфейсу та містить ключові поля: назва адаптера (наприклад, "Intel(R) Ethernet"), тип підключення (Ethernet або Wi-Fi), поточна IP-адреса, середній ping у мілісекундах, відсоток втрат пакетів та стан з'єднання — активний або неактивний.

Цей список оновлюється автоматично після кожного завершеного циклу перевірки, щоб відображати актуальні дані з мінімальною затримкою. У разі появи або зникнення інтерфейсу

(наприклад, при підключенні USB-модему або відключенні Wi-Fi) оновлення відбувається з урахуванням нової конфігурації.

Для зручності користувача активне з’єднання виділяється в таблиці: наприклад, кольоровим рядком або спеціальним графічним індикатором у відповідному стовпці. Це дозволяє з першого погляду визначити, який саме адаптер наразі використовується системою як основний, і швидко порівняти його параметри з іншими доступними мережами. Такий підхід забезпечує прозору візуалізацію стану мереж і дозволяє швидко реагувати на зміни або аналізувати ситуацію без потреби переходити в системні налаштування Windows.

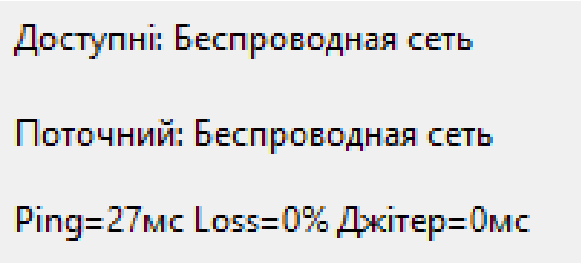


Рисунок 1.6. Список мережевих інтерфейсів та поточні данні

Панель налаштувань є важливою складовою інтерфейсу, що дозволяє користувачеві тонко налаштувати поведінку програми відповідно до своїх вимог і типу мережевого середовища. Вона складається з декількох логічних блоків.

Перший блок зосереджений на заданих критеріях якості з'єднання, які визначають, коли система повинна вважати мережу нестабільною. Тут розташовані поля для введення:

– максимального ping, що вимірюється в мілісекундах, наприклад, 150 мс це межа, після якої зв'язок вважається затриманим;

– максимального відсотку втрати пакетів — значення, що визначає, який рівень втрат ще допускається для нормального функціонування;

– інтервалу перевірки, тобто частоти, з якою програма виконуватиме аналіз мережевого стану. Зазвичай цей параметр задається в секундах, і чим менше значення, тим чутливіше реагує система.

Другий блок панелі — це список мережевих інтерфейсів, які виявляються при запуску програми. Користувач може змінювати їх пріоритет, визначаючи, який адаптер слід вважати основним у випадку рівнозначної якості. Список реалізовано у зручній формі, яка підтримує зміну порядку елементів за

допомогою кнопок або drag-and-drop дій (перетягування мишкою), що дозволяє швидко і наочно керувати налаштуванням пріоритетів.

Цей розділ панелі забезпечує зрозумілий і доступний спосіб впливати на поведінку програми без потреби втручання у код або системні параметри, залишаючи користувачеві повний контроль над логікою перемикання.

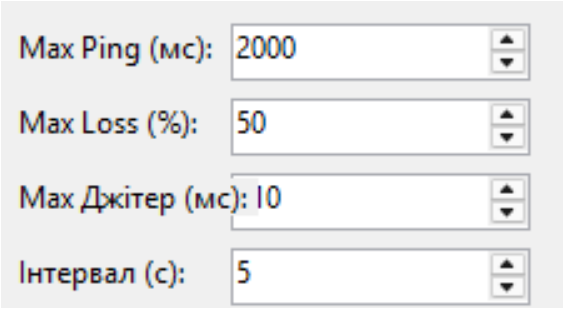


Рисунок 1.7. параметри оцінки

Журнал подій є окремим функціональним розділом інтерфейсу, реалізованим у вигляді вкладки, що надає користувачеві повну прозорість стосовно роботи застосунку в реальному часі. У цьому журналі фіксуються всі значущі події, пов’язані з мережею та автоматичними рішеннями системи.

Кожен запис у журналі містить дату і час, що дозволяє точно простежити, коли саме відбулася та чи інша подія. Далі вказується тип події — наприклад, виявлено нестабільність з'єднання, відбулося автоматичне перемикання адаптера, виникла помилка при спробі підключення, або ж операція завершилась успішно. Деталі події містять додаткову інформацію: назву адаптера, його IP-адресу, значення ping і втрат пакетів на момент спрацювання, а також прийняте рішення (перемикання, ігнорування, тайм-аут тощо).

Інтерфейс журналу побудований таким чином, щоб користувач мав можливість не лише переглядати історію, а й взаємодіяти з нею. Передбачені опції збереження в файл для подальшого аналізу або звітності, а також очищення журналу — зручна функція у випадках тестування, відлагодження чи оновлення налаштувань.

Таким чином, журнал подій виконує роль центрального джерела інформації про рішення системи та її поведінку у відповідь на зміну стану мережі.

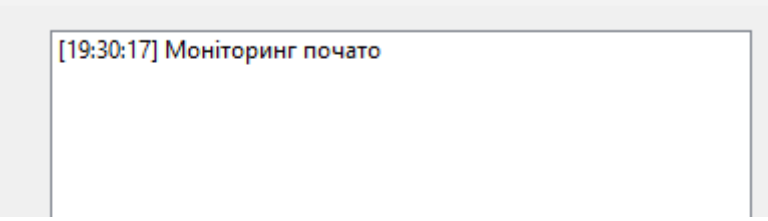


Рисунок 1.8. Панель логів

Меню в системному треї реалізоване для зручної взаємодії з програмою, яка більшість часу працює у фоновому режимі. Після запуску застосунок автоматично згортається до піктограми в системному треї Windows — це дозволяє уникнути зайвого навантаження на інтерфейс користувача та водночас забезпечити швидкий доступ до основних функцій.

Натискання правою кнопкою миші на піктограмі відкриває компактне контекстне меню, що містить такі пункти:

- “Зупинити моніторинг” — тимчасово призупиняє всі фонові перевірки якості з’єднання без закриття застосунку.
- “Перезапустити перевірку” — примусово ініціює миттєвий запуск аналізу адаптерів незалежно від інтервалу, заданого у налаштуваннях.
- “Вийти з програми” — повністю завершує роботу застосунку, вивантажуючи його з пам’яті та зупиняючи всі модулі.

Це меню забезпечує швидкий контроль над функціональністю програми без потреби відкривати основне вікно, що особливо зручно для користувачів, які бажають звести до мінімуму взаємодію із застосунком під час звичайної роботи за комп’ютером.

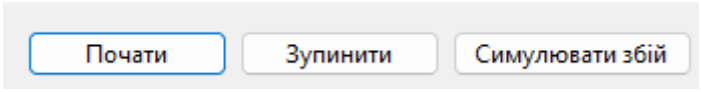


Рисунок 1.9. Меню системного трею

Повідомлення користувача Реалізовано за допомогою системних сповіщень (NotifyIcon.ShowBalloonTip()), які інформують про перемикання мережі, виявлення проблем або помилки. Наприклад: “Перемикання на Ethernet – ping Wi-Fi перевищив 200 мс”.

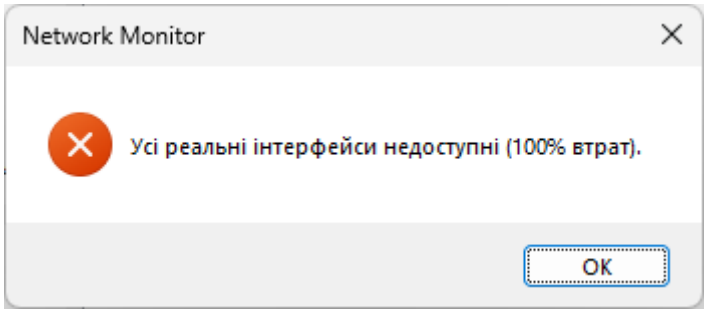


Рисунок 1.10. Повідомленні при недоступності жодної мережі

1.8.3 Логіка взаємодії елементів

Інтерфейс взаємодіє з логікою програми через подієву модель. Наприклад, при кожному циклі перевірки оновлюється список адаптерів, а зміна їхнього стану автоматично призводить до оновлення елементів GUI. Команди з інтерфейсу (наприклад, натискання кнопки "Зберегти налаштування") викликають відповідні методи бізнес-логіки.

Також передбачено механізм обробки винятків: при виникненні критичних помилок користувачу відображається вікно з описом проблеми, а деталі фіксуються у лог.

1.8.4 Особливості реалізації

Усі елементи інтерфейсу оновлюються динамічно в реальному часі, без потреби у перезавантаженні вікна чи взаємодії з користувачем. Це дозволяє постійно відображати актуальну інформацію про стан мережевих адаптерів, затримки, втрати пакетів і результати перевірок без затримок і навантаження на систему.

Застосунок спроєктовано таким чином, щоб після первинного налаштування участь користувача була мінімальною. Програма автоматично здійснює моніторинг, приймає рішення про перемикання, оновлює інтерфейс і веде логування, не потребуючи втручання.

Основний акцент зроблено на автоматичному режимі роботи — користувач лише контролює стан і, за потреби, коригує параметри через інтерфейс. Усі налаштування адаптерів, порогів, частоти перевірок і пріоритетів доступні через зручну панель параметрів.

Окремо передбачено підтримку локалізації — усі текстові елементи інтерфейсу винесені в ресурсні файли формату .resx, що дозволяє швидко додавати переклади та перемикати мову інтерфейсу без зміни логіки програми. Це робить застосунок гнучким і зручним для користувачів з різними мовними уподобаннями.

1.9 Логування, обробка винятків та збереження історії роботи

Наявність повноцінної системи логування та обробки винятків є критично важливою частиною будь-якого програмного забезпечення, особливо у випадках, коли воно працює у фоновому режимі без постійного контролю з боку користувача. У розробленому застосунку реалізовано власну систему фіксації подій, помилок, а також механізм збереження історії мережевої активності для подальшого аналізу.

1.9.1 Мета логування

Логування в системі автоматичного перемикання мереж є ключовим інструментом для забезпечення прозорості та діагностики. Завдяки веденню журналу подій програма фіксує усі значущі дії — від моменту запуску, виявлення проблем із мережею, до рішень про перемикання або виникнення виняткових ситуацій. Це дозволяє не лише виявляти й усувати потенційні помилки, а й надавати користувачу обґрунтовану інформацію про те, чому була обрана та чи інша дія.

Крім того, лог-файли можуть бути використані для ретроспективного аналізу: наприклад, для виявлення нестабільних інтерфейсів, підрахунку частоти перемикань або збору статистики щодо параметрів з'єднання. Такий підхід дозволяє на основі реальних даних вдосконалювати алгоритм логіки перемикання в майбутніх версіях програми.

1.9.2 Структура логів

Кожен запис у журналі подій створюється у структурованому форматі, що дозволяє легко аналізувати інформацію як вручну, так і за допомогою автоматизованих інструментів. Основними складовими кожного запису є:

Дата та час — фіксується точна мітка часу (до мілісекунд), коли подія відбулася, що дозволяє відтворити хронологію дій

Тип події — класифікація за рівнем важливості: *інформація* (наприклад, запуск моніторингу), *попередження* (нестабільне з'єднання), *помилка* (невдала спроба перемикання), *критичний збій* (відмова адаптера, падіння застосунку).

Опис — стислий текст, що пояснює суть події, наприклад: "Перемикання на Ethernet", "Ping перевищив 250 мс", "Адаптер Wi-Fi недоступний".

Контекст — допоміжна інформація: ім'я або ID адаптера, значення ping, відсоток втрат, обрана альтернатива, тощо.

Такий підхід дозволяє ефективно ідентифікувати причини дій програми та виявляти закономірності у нестабільності мереж.

1.9.3 Реалізація логування

Логування в програмі реалізоване через спеціалізований сервіс LoggerService, який інкапсулює всю логіку обробки подій. Такий підхід дозволяє централізовано керувати записами, забезпечуючи як продуктивність, так і надійність у роботі.

Всі повідомлення спочатку поміщаються в оперативний буфер, що дозволяє мінімізувати кількість операцій запису на диск. Далі, у фоновому режимі, вони поступово записуються у файл через StreamWriter, відкритий у режимі дописування. Потік автоматично закривається після завершення кожної сесії запису, щоб уникнути витоку ресурсів.

Передбачено механізм контролю розміру лог-файлів — при досягненні порогового об'єму (наприклад, 5 МБ) створюється архівна копія з часовою міткою, а поточний файл очищується для подальшого використання. Крім того, реалізовано ведення щоденних журналів із динамічним іменуванням файлів відповідно до дати, що спрощує пошук і аналіз історичних подій.

Файли зберігаються у підкаталозі Logs, розміщеному у внутрішній директорії користувача (AppData/Local/НазваПрограми/Logs), що забезпечує стабільний доступ незалежно від рівня доступу до системного кореня. Запис у

файл здійснюється в окремому потоці, щоб уникнути блокування основного UI або модулів моніторингу.

1.9.4 Обробка винятків

Для запобігання аварійному завершенню роботи застосунку всі критично важливі фрагменти коду реалізовано з використанням конструкцій обробки винятків try-catch. Такий підхід дозволяє у випадку виникнення помилки не лише уникнути припинення виконання програми, а й виконати необхідні дії для збереження стабільності системи. Якщо під час роботи виявляється виняткова ситуація, система автоматично фіксує подію у лог-файлі із зазначенням часу, типу помилки та контексту її виникнення. У разі необхідності користувачеві відображається інформативне повідомлення з коротким описом проблеми, що дозволяє йому зорієнтуватися у ситуації. Після цього застосунок або намагається відновити своє функціонування, якщо це можливо без втрати даних, або коректно завершує роботу з попереднім збереженням усіх важливих станів.

У процесі реалізації було враховано низку потенційно небезпечних ситуацій, зокрема таких, як відсутність доступу до одного чи кількох мережевих інтерфейсів, обмеження прав користувача на виконання операцій із системними адаптерами, збої при виконанні ping-запитів або під час взаємодії з інструментами командного рядка PowerShell, а також проблеми, пов’язані з некоректною IP-конфігурацією мережі. Крім того, передбачено можливість виникнення помилок під час доступу до файлів журналювання або конфігураційних файлів, які також обробляються окремо із повідомленням про несправність. Така гнучка система обробки винятків дозволяє забезпечити надійність і стійкість застосунку навіть у разі виникнення несподіваних технічних проблем.

Всі необроблені винятки перехоплюються глобальним обробником AppDomain.CurrentDomain.UnhandledException.

1.9.5 Журнал подій у GUI

Крім збереження логів у файл, основні події також виводяться у вкладку “Журнал” в інтерфейсі програми. Користувач може:

- переглядати останні події у вигляді списку;
- копіювати або експортувати події у зовнішній файл;
- фільтрувати записи за типом (успішні перемикання, помилки, попередження).

Це дозволяє не лише виявляти проблеми, а й аналізувати, наскільки стабільною була мережа протягом певного часу.

1.9.6 Збереження історії

Програма підтримує збереження агрегованої історії мережевих подій — кількість перемикань, середній ping за добу, частота втрат, тощо. Ці дані можуть зберігатися у форматі .csv або .json для подальшого аналізу.

Потенційно ці дані можуть бути використані для побудови графіків або візуалізації стабільності інтернету за період (наприклад, тиждень чи місяць), якщо передбачити додатковий модуль статистики.

1.10 Автоматизація запуску, робота у фоновому режимі та взаємодія з Windows

Оскільки програмне забезпечення призначене для постійного моніторингу якості мережевого з’єднання, воно має запускатися автоматично при старті системи, працювати без активного вікна та мати мінімальний вплив на продуктивність комп’ютера. У цьому розділі розглянуто, як ці вимоги реалізовано у взаємодії із середовищем операційної системи Windows.

1.10.1 Автоматичний запуск разом із системою

Для забезпечення безперервної роботи застосунку після кожного увімкнення комп’ютера, реалізовано механізм автозапуску. Його суть полягає у додаванні ярлика програми до системного реєстру Windows або до спеціальної папки автозавантаження користувача.

Реалізовано два підходи:

1. Через реєстр Windows (рекомендується для стабільності)

   Додається запис до ключа:

   зі значенням повного шляху до виконуваного файлу програми.

2. Через папку автозавантаження

Програма автоматично створює ярлик у папці:

Увімкнення чи вимкнення автозапуску керується у налаштуваннях програми через відповідний чекбокс.

1.10.2 Робота у фоновому режимі

Після запуску програма не відкриває головне вікно, а автоматично мінімізується в системний трей. При цьому:

- відбувається ініціалізація усіх основних служб (моніторингу, логіки перемикання, логування);
- запускається цикл перевірки мереж;
- оновлюється іконка в системному треї відповідно до стану підключення.

Завдяки цьому користувач може користуватися комп'ютером без будь-яких відволікань — застосунок виконує свою роботу непомітно.

Іконка в треї слугує як візуальний індикатор (наприклад, зелена — стабільне з'єднання, червона — проблеми), і містить контекстне меню для швидкого доступу до функцій (вихід, перезапуск моніторингу, налаштування тощо).

1.10.3 Обмеження прав доступу

Деякі функції застосунку (зокрема, перемикання мереж, вимкнення адаптерів, зміна метрик) потребують підвищених прав у системі. Для їх реалізації передбачено:

- перевірку рівня доступу при запуску;
- виклик окремих скриптів або команд через runas;
- повідомлення користувача у разі недостатніх прав з пропозицією перезапуску з адміністративним доступом.

У випадках, коли права адміністратора недоступні, програма обмежує функціонал і працює у режимі моніторингу без активного втручання в налаштування мережевих адаптерів.

1.10.4 Мінімізація навантаження

Усі процеси, пов'язані з моніторингом стану мережевих інтерфейсів, збором технічних метрик, логуванням подій та аналізом отриманих даних,

реалізовано у фонових потоках із використанням механізмів BackgroundWorker, Task та System.Timers.Timer. Такий підхід дозволяє уникнути блокування головного потоку інтерфейсу користувача, забезпечуючи безперебійну та плавну роботу програми навіть у разі тимчасових затримок мережевих відповідей. Це особливо важливо для застосунків із графічним інтерфейсом, де затримка в обробці подій може призвести до некоректного виведення даних або заморожування вікна програми.

Інтервал перевірки, тобто час між окремими циклами опитування мережевих адаптерів, обирається таким чином, щоб досягти балансу між швидкою реакцією на втрату зв'язку чи погіршення якості з'єднання та оптимальним використанням ресурсів комп'ютера. У типовій конфігурації цей інтервал становить від 5 до 10 секунд, однак він може бути змінений користувачем у параметрах застосунку залежно від цілей моніторингу — від постійного контролю в корпоративному середовищі до епізодичної перевірки в домашніх умовах.

Особливістю реалізації є повна відмова від використання нескінченних циклів (while (true) із Thread.Sleep) для підтримки періодичності перевірок. Замість цього застосунок використовує подієво-орієнтовану модель, де обробники подій прив'язані до таймера або результату асинхронного запиту. Такий підхід дозволяє значно зменшити навантаження на процесор, підвищити енергоефективність (особливо актуально для ноутбуків і мобільних пристроїв), а також зробити код програми більш чистим, модульним і придатним до розширення.

Кожен фоновий процес виконує чітко визначене завдання — окремі задачі відповідають за отримання ping-запитів, оцінку втрати пакетів, аналіз отриманих результатів на відповідність пороговим значенням, запис логів та оновлення графічних елементів інтерфейсу. У разі виявлення критичних значень або втрати з'єднання, відповідний потік формує повідомлення, яке через механізм диспетчеризації (наприклад, Invoke або SynchronizationContext) безпечно передається в основний UI-потік для відображення користувачу.

1.11 Тестування і результати роботи застосунку

Після завершення етапу розробки всі ключові функціональні компоненти програмного забезпечення було піддано ретельному тестуванню у реальному середовищі експлуатації. Основною метою цього процесу була перевірка того, наскільки поведінка програми відповідає попередньо визначеним функціональним вимогам. Особлива увага приділялася оцінці стабільності роботи системи за умов змінної якості мережевого з'єднання. Під час випробувань аналізувалося, як застосунок реагує на втрату зв'язку, суттєве підвищення затримок, появу або зникнення мережевих адаптерів, а також на інші нетипові ситуації, які можуть виникнути у реальній практиці.

Також перевірялася коректність механізму логування, зокрема відповідність записів реальним подіям та своєчасність їх фіксації. Усі журнали подій виявилися повними, структурованими й доступними для подальшого аналізу. Окремо було протестовано збереження та відновлення налаштувань користувача після перезапуску, що підтвердило коректність роботи відповідних модулів. Таким чином, результати тестування засвідчили функціональну відповідність програмного продукту поставленим завданням і готовність до використання в умовах реального навантаження.

1.11.1 Методика тестування

Тестування програмного забезпечення здійснювалося на фізичному комп'ютері, який був обладнаний двома одночасно активними мережевими інтерфейсами — Wi-Fi та Ethernet. У процесі випробувань були змодельовані різні типові сценарії, що імітують характерні проблеми з мережевим з'єднанням. Зокрема, перевірялася реакція системи на повне відключення мережевого адаптера вручну, на штучне підвищення затримки (ping) шляхом емуляції високого навантаження, на втрату доступу до мережі внаслідок вимкнення маршрутизатора, а також на зникнення сигналу Wi-Fi, що є поширеною ситуацією у середовищах з нестабільним бездротовим покриттям.

Перед початком тестування було визначено граничні порогові значення для параметрів, за якими система оцінює якість з'єднання. Максимально допустимий

рівень затримки було встановлено на рівні 150 мілісекунд, допустима втрата пакетів не повинна була перевищувати 10 відсотків, а інтервал між перевірками складав 5 секунд. Для забезпечення стабільності було також задано пріоритет використання мереж, згідно з яким Ethernet вважався основним каналом зв'язку, а Wi-Fi використовувався як резервний варіант. Такий підхід дозволив максимально наблизити умови тестування до реального середовища експлуатації та оцінити ефективність системи в умовах змінної якості мережевого з'єднання.

1.11.2 Тестові сценарії та результати

Сценарій 1: Нормальна робота Wi-Fi

- Ping у межах 25–30 мс.
- Жодних втрат.
- Програма не ініціює перемикання.
- Стан відображається як "стабільний".
- Результат: Успішно**.**

Сценарій 2: Втрати пакетів на Wi-Fi

- Емуляція втрат (відключення інтернету при збереженні підключення до маршрутизатора).
- Ping перевищив 300 мс, втрати понад 50%.
- Через 3 цикли відбувається автоматичне перемикання на Ethernet.
- Лог містить відповідний запис з деталями.
- Результат: Успішно.

Сценарій 3: Вимкнення Ethernet

- Фізичне відключення кабелю.
- Програма виявляє втрату адаптера.
- У разі погіршення Wi-Fi — немає доступного резерву.
- Стан позначається як "нестабільне з'єднання".
- Результат: Успішно, перемикання не виконано (відсутні альтернативи).

Сценарій 4: Перемикання туди-назад

- Початково: Wi-Fi — нестабільне, Ethernet — стабільне.
- Перемикання на Ethernet.

- Через 1 хвилину Wi-Fi стабілізується.
- Захист від флапінгу не допускає повторного перемикання до завершення періоду стабілізації (30 сек).
- Результат: Успішно, система стабільно утримує підключення.

Сценарій 5: Вихід з програми і перезапуск

- Програма закривається.
- Після повторного запуску всі налаштування збережені.
- Моніторинг відновлюється автоматично.
- Журнал подій продовжується.
- Результат**:** Успішно.

1.11.3 Спостереження щодо стабільності

Програма демонструє стабільну роботу протягом тривалого періоду, що підтверджено результатами тестування, яке тривало понад 12 годин без необхідності перезапуску. У фоновому режимі споживання системних ресурсів залишається на низькому рівні: завантаження центрального процесора коливається в межах 1–3 відсотків, а обсяг використаної оперативної пам’яті не перевищує 80 мегабайтів. Протягом усього часу спостереження не зафіксовано жодних витоків пам’яті, що свідчить про ефективне керування ресурсами в межах застосунку.

Окрему увагу приділено веденню журналів подій: лог-файли формуються коректно, своєчасно оновлюються, залишаючись доступними для читання та аналізу. Вони не блокуються сторонніми процесами і можуть бути відкриті навіть у момент активного запису. Інтерфейс користувача працює синхронно з фоновими модулями моніторингу: всі індикатори відображають поточний стан мережевих інтерфейсів без затримок чи розбіжностей, що забезпечує користувачеві достовірну інформацію в режимі реального часу.

Лог-файли створюються коректно, не блокуються іншими процесами. Індикатори в GUI працюють узгоджено з реальним станом мереж

2. ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

У цьому дипломному проєкті виконана розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання. Розроблений застосунок стабільно моніторить доступні інтерфейси, своєчасно фіксує погіршення якості з’єднання та безперебійно перемикає активне підключення на оптимальний канал. Застосунок успішно пройшов тестування в різних сценаріях — від емуляції втрати пакету до фізичного відключення кабелю — та продемонстрував надійну роботу без витоків пам’яті й зайвого навантаження на систему. Відзначено, що інтеграція з існуючими системами моніторингу та управління мережею, а також можливість розгортання в рамках Active Directory, відкриває перспективи масштабного використання розробленого продукту в корпоративному сегменті.

Ефективність програмного забезпечення (ПЗ) визначається не лише його якісними характеристиками, але й ефективністю процесу розробки. Якість ПЗ оцінюється за кількома критеріями: потреби та враження користувачів - наскільки ПЗ відповідає очікуванням і зручне у використанні: економічність використання ресурсів - наскільки ефективно ПЗ використовує системні ресурси; відповідність встановленим вимогам- чи відповідає ПЗ технічним специфікаціям та стандартам.

З точки зору користувача, важливим аспектом оцінки якості є витрати на розробку, що включають обсяг трудових витрат та загальні фінансові вкладення. У цьому розділі ми представимо розрахунок вартості розробленого програмного забезпечення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість створення програмного продукту визначається низкою чинників, серед яких – обсяг робіт, рівень трудомісткості, кваліфікація розробників, а також строки, що задаються ринковими умовами. Для оцінки обсягу програмного забезпечення використовується метод структурної

аналогії, згідно з яким на основі спеціалізованих каталогів аналогів визначається кількість умовних машинних команд для подібного програмного продукту (у тисячах команд).

Таблиця 2.1. -Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3 ПП введення інформації	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткість визначаємо на основі табл.2.2

Таблиця.2.2

Обсяг ПЗ, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначаємо укрупнену норму часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПЗ, тобто в умовах комп'ютера, Кк=0,7÷0,8), для нашого варіанта виділено сірим кольором:

Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначаємо по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

Трудомісткість технічного завдання

$$T_{тз}=Ta*L_1*K_н=183,2 *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=Ta*L_2*K_н=183,2 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$T_{рп}=Ta*L_З*К_н*К_т=183,2 *0,61*0,7*0,6= 46,94$ (люд/годин) (2.3)

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=27 (стр), розробка робочого проекту $N_{рп}$=11 (стр), пояснювальна записка відповідно $N_{пз}$ 21 (стр)

Розрахунок зведений у таблицю 2.6

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=0,15*2=0,3$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*27=18,9$	$T_{нк}=0,15*N_{тп}=0,15*27=4,05$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*11=7,7$	$T_{нк}=0,15*N_{рп}=0,15*11=1,65$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*21=31,5	$T_{кк}=0,7*N_{тз}=0,7*21=14,7$	$T_{нк}=0,15*N_{пз}=0,15*21$ =3,15
Усього, в т.ч.:	159,78		
- на розробку	$\Sigma T_р$=107,93		
- контроль		$\Sigma T_{кк}$=42,7	
- нормоконтроль			$\Sigma T_{нк}$=9,15

2.3. Розрахунок ціни програмного продукту

Щоб визначити загальну вартість розробки програмного забезпечення, ми розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати виконавців наведено в Таблиці 2.7. Згідно зі статтею 8 Закону України "Про Державний бюджет України на 2025 рік", мінімальна заробітна плата з 1 січня 2025 року становить 8000 гривень на місяць, а мінімальна погодинна тарифна ставка — 48,00 гривень.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	107,93	48,00	5180,64
2.Контроль керівника	42,7	90,00	3843,00
3.Нормоконтроль	9,15	100,00	915,5
Усього	-	-	ΣЗо= 9938,64

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8

Таблиця 2.8.- Розрахунок матеріальних витрат на розробку ПП

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	61	5.0	305,00
Разом	-	-	-	$В_{м\,і}$ =305,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*305=30,5
Усього				$В_м=В_{мі}+В_{тр\ з=}$ 335,50

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	335,5	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	9938,64	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	993,86	$Зд = 0{,}1 \times Зо$ = 9938,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2405,15	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (9938,64+993,86)
5. Накладні витрати	3975,46	$Внак. = 0{,}4 \times Зо$ =0,4 * 9938,64
6. Повна собівартість	17648,61	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 335,5+9938,64+993,86+2405,15 +3975,46

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100=(17648,61*15)/100=2647,29$ грн (2.4)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П$ = 17648,61+2647,29 = 20295,9 грн; (2.5)

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$Ц_р=Ц_о+ПДВ$ = 20295,9 + 20295,9 *0.2 = 24355,08 грн; (2.6)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Забезпечення безпеки життя та здоров'я громадян під час виконання ними трудових обов'язків, а також створення умов, що не шкодять здоров'ю, є одним із пріоритетних завдань держави. Кожне робоче місце повинно бути оснащене таким чином, щоб гарантувати зручність і безпеку співробітників. Наприклад, виробниче обладнання, обслуговування якого вимагає переміщення персоналу, слід обладнати надійними і зручними проходами, майданчиками, сходами та поручнями. Водночас експлуатація устаткування не повинна перевищувати встановлені норми викидів шкідливих речовин і створювати загрози пожеж або вибухів.

Метою даного дипломного проекту є Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці даного програмного комплексу

При розробці програмного комплексу проводиться ретельний аналіз можливого впливу виробничих чинників, які можуть зашкодити здоров'ю програміста. Згідно зі стандартом ГОСТ 12.1.003-74, до небезпечних факторів належать ті, що можуть спричинити раптове погіршення здоров'я або навіть летальний результат під час роботи.

Аналіз також враховує різні якісні характеристики робочого середовища — фізичні параметри приміщень, такі як температура, вологість, електричний опір підлоги, а також дані щодо концентрації іонів і забруднювачів у повітрі.

3.2 Гігієнічні вимоги до виробничого середовища

Сучасне виробництво вимагає створення оптимальних санітарних умов, які забезпечують плідну роботу співробітників без надмірного навантаження. Це досягається організацією комфортного робочого місця з чистим повітрям, правильною освітленістю, а також заходами щодо захисту від шумових та вібраційних впливів.

3.2.1 Мікроклімат

Недотримання норм мікроклімату негативно позначається на здоров'ї людини, що може призвести до зниження працездатності або її повної втрати. Показники мікроклімату мають відповідати нормам, визначеним у ДСН 3.3.6.042-99. Згідно з чинними нормативними документами (ДСанПіН 3.3.2-007-98), у холодні періоди температура повітря повинна перебувати від –22 до +24°С, швидкість руху повітря – близько 0,1 м/с, а відносна вологість – у межах 40–60%.

В теплий сезон допустимі значення температури складають 23–25°С при збереженні вологості та швидкості повітря (0,1–0,2 м/с) на тих же рівнях. Підвищення кількості позитивних іонів у робочій зоні також може негативно впливати на здоров'я, тому оптимальний рівень аероіонізації вважається в межах від 150 до 5000 легких аерофонів на 1 см³.

Вплив на покращення складу робочого повітря здійснюється примусовою вентиляцією, застосуванням захисних екранів із заземленням або іонізаторів, а також можливістю регулювання основних параметрів мікроклімату.

3.2.2 Освітлення

Правильно організоване освітлення позитивно впливає на центральну нервову систему, сприяє зниженню енергетичних витрат організму під час виконання завдань і покращує продуктивність праці. Надмірне або недостатнє освітлення може призводити до перенапруження зору, втоми, зниження швидкості робочих процесів та, з часом, до розвитку захворювань очей, таких як короткозорість.

Тому освітлення робочих приміщень має відповідати нормам СниП II.4-79. Забезпечення рівномірного світлового потоку досягається за допомогою відбитого або розсіяного світла, яке слід поєднувати з природним освітленням, дотримуючись нормативного рівня в 300–500 лк. При цьому необхідно уникати відблисків від клавіатури, екрана та інших пристроїв, спрямованих у бік очей користувача.

3.2.3 Шум

Деякі пристрої, що працюють з ВДТ, можуть стати джерелами різних звукових коливань – як у чутному, так і в ультразвуковому діапазоні. Постійний або тривалий вплив такого шуму спричиняє зниження працездатності, погіршення концентрації, збільшення кількості помилок, зорову втому, зміну сприйняття кольорів та появу головного болю.

Нормативним показником для робочого місця є рівень шуму до 50 дБ, а для його зниження слід застосовувати заходи, як-от усунення причин шуму на етапі проектування, використання звукопоглинаючих матеріалів та оптимізація планування виробничих приміщень.

3.2.4 Вимоги до організації робочого місця працівника

Під час виконання паяльних робіт потрібно суворо дотримуватися норм організації робочого місця. Кожен елемент робочої зони має бути розташований так, щоб забезпечити максимальний комфорт і безпеку, усуваючи зайві предмети, які можуть створювати перешкоди.

Паяльне обладнання, робочі інструменти і деталі, а також засоби індивідуального захисту повинні перебувати у справному стані та відповідати стандартам охорони праці. Паяльні роботи проводяться з використанням електропаяльника, який живиться від мережі 220 В і має споживання не більше 100 Вт. Використання кислот або рідин на основі кислотних розчинів суворо заборонено.

Під час ремонтних робіт обладнання повинно бути повністю відключене від електроживлення (штк. вилка вилучається з розетки), а всі доступні елементи – ізольовані від мережі. Через застосування різних припоїв і флюсів, що містять шкідливі компоненти (свинець, цинк, літій, калій, натрій, кадмій тощо), робочі місця паяльників повинні бути обладнані додатковими локальними витяжними системами.

3.2.5 Електробезпека

Для запобігання ураженню електричним струмом необхідно чітко дотримуватися правил безпечного виконання робіт і експлуатації техніки.

Оператор повинен бути захищений від доступу до частин обладнання, що працюють під високою напругою, а також до неізольованих елементів, які не підключені до захисного заземлення. Електроживлення комп'ютерної техніки має підключатися виключно через спеціальні штекери із заземленням.

3.3 Пожежна безпека

До систем гасіння пожеж належать як внутрішні пожежні водопроводи (крани-ПК), так і різні типи вогнегасників, зокрема вуглекислотні, порошкові, а також сухий пісок. У будівлях пожежні крани розташовують у коридорах та на сходових майданчиках; кожен кран комплектується пожежним рукавом і встановлюється у спеціальних ящиках, розташованих на певній висоті від підлоги.

На початкових стадіях пожеж застосовують вогнегасники, найбільш ефективними серед яких є вуглекислотні пристрої, що дозволяють не лише гасити загоряння, але й зберігати електрообладнання. Такі засоби мають бути розміщені у легкодоступних місцях, на відповідній висоті від підлоги. Крім того, виробничі приміщення повинні мати запасні виходи, де двері мають бути позначені освітленим написом «Запасний вихід», а схема евакуації – розміщена біля основного виходу.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани - ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1,35 м від полу.

Пожежна безпека є комплексною системою заходів, спрямованих на запобігання займання, своєчасне виявлення пожежі та ефективне ліквідування загоряння. До основних засобів цього захисту відносяться як внутрішні пожежні водопроводи (крани-ПК), так і різноманітні типи вогнегасників: вуглекислотні, порошкові, а також засоби за основою сухого піску. У будівлях пожежні крани зазвичай розташовують у коридорах та на сходових майданчиках; кожен кран комплектується пожежним рукавом і встановлюється в спеціально обладнаних

ящиках, оптимально розміщених на висоті приблизно 1,35 м від підлоги. Таке розташування забезпечує легкий доступ до засобів гасіння у разі надзвичайної ситуації.

Рисунок 3.1. Протипожежна безпека

На початкових стадіях загоряння застосовують вогнегасники, найбільш ефективними з яких є вуглекислотні пристрої. Вони дозволяють не лише швидко знищити загоряння, але й мінімізувати можливі пошкодження електрообладнання, що особливо важливо у виробничих приміщеннях. Водночас, стандартними засобами гасіння пожежі можуть виступати і порошкові вогнегасники, а також сухий пісок, які використовуються для локалізації загоряння до прибуття аварійних служб.

ВИСНОВКИ

Виконана робота підтвердила ефективність обраного підходу до автоматичного перемикання мереж на основі адаптивного аналізу параметрів з’єднання. Розроблений застосунок стабільно моніторить доступні інтерфейси, своєчасно фіксує погіршення якості з’єднання та безперебійно перемикає активне підключення на оптимальний канал. Застосунок успішно пройшов тестування в різних сценаріях — від емуляції втрати пакету до фізичного відключення кабелю — та продемонстрував надійну роботу без витоків пам’яті й зайвого навантаження на систему.

Подальші напрямки вдосконалення можуть включати розширення аналітичного модуля та інтеграцію візуалізації статистики в реальному часі, реалізацію підтримки VPN-з’єднань і багатомовний інтерфейс. Запропоноване рішення може бути корисним як для індивідуальних користувачів, так і для корпоративних середовищ із підвищеними вимогами до стабільності мереж, забезпечуючи автоматичний моніторинг і миттєве реагування на зміни якості інтернет-з’єднання.

В процесі експлуатації було виявлено, що застосунок також може слугувати інструментом діагностики для сисадмінів та IT-служб, додаючи прозорості у виявленні проблем із мережевою інфраструктурою. Ведення журналу перемикань і метрик дозволяє отримувати докладні звіти про поведінку мережі в різних умовах експлуатації.

Відзначено, що інтеграція з існуючими системами моніторингу та управління мережею, а також можливість розгортання в рамках Active Directory, відкриває перспективи масштабного використання розробленого продукту в корпоративному сегменті. Це підтверджує практичну значущість проєкту та обґрунтовує подальші дослідження в напрямку автоматизації мережевих операцій.

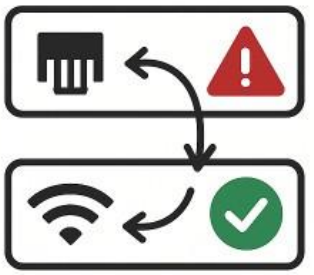
ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. SeriousBit. NetBalancer — Software for network traffic control and monitoring [Електронний ресурс]. — Режим доступу: https://netbalancer.com
2. Speedify. Fast Bonding VPN — Combine Wi-Fi, Ethernet, Cellular & more [Електронний ресурс]. — Режим доступу: https://speedify.com
3. Microsoft Docs. Configure the order of network adapters in Windows [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com
4. Microsoft Docs. Set Interface Metric via PowerShell [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/powershell
5. Microsoft. Windows Management Instrumentation (WMI) [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/windows/win32/wmisdk
6. Microsoft. NetworkInterface Class (System.Net.NetworkInformation) [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/dotnet/api/system.net.networkinformation.networkinterface
7. Google Public DNS [Електронний ресурс]. — Режим доступу: https://developers.google.com/speed/public-dns
8. Cloudflare DNS — 1.1.1.1 [Електронний ресурс]. — Режим доступу: https://1.1.1.1
9. Lipsum.com. Lorem Ipsum Generator [Електронний ресурс]. — Режим доступу: https://www.lipsum.com/feed/html

ДОДАТОК А. Слайди мультимедійної презентації

Актуальністьі мета роботи

- Проблема нестабільного інтернету
- Windows не перемикає мережі автоматично
- Зниження якості онлайн-роботи
- Мета — створити застосунок для моніторингу та перемикання
- Працює без участі користувача



Інтелектуальне перемикання мережевих з'єднань у Windows-середовищі

Здобувач: Цюсьмак М.А.

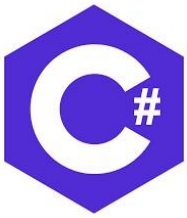
Керівник: Нестеренко вд

Аналіз існуючих рішень

NetBalancer забезпечує контроль мережевого трафіку, але не підтримує автоматичне перемикання між підключеннями.
Speedify використовує VPN-технології для об'єднання каналів, однак має обмежену гнучкість у налаштуваннях.
Система Windows Metric дозволяє вручну задавати пріоритети для мережевих інтерфейсів, проте не забезпечує автоматизації цього процесу.

Вибір технологій

Проєкт реалізовано мовою C# з використанням платформи .NET Framework 4.8 і технології WinForms для створення графічного інтерфейсу.
Вибір відмови від Python та Electron обумовлений міркуваннями стабільності, продуктивності й низького споживання ресурсів.
Застосунок забезпечує глибоку інтеграцію з Windows API, що дозволяє ефективно взаємодіяти з системними компонентами.



Використані протоколи та інструменти

У проєкті використано декілька технологій для забезпечення функціональності моніторингу мережі:

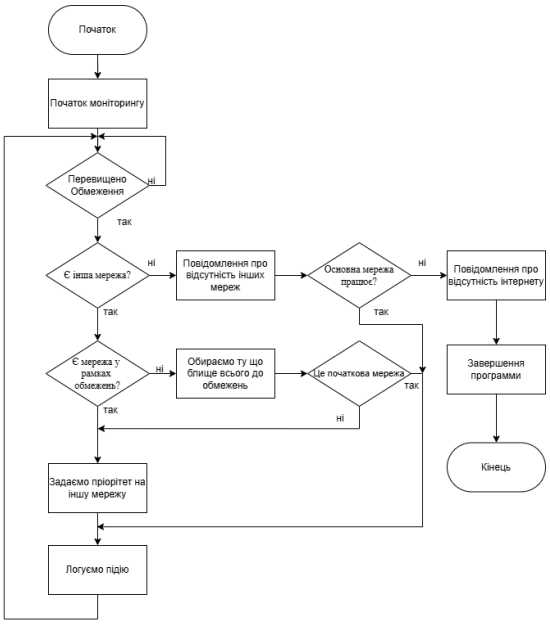
Ping (ICMP) застосовується для оцінки затримки,

Netsh і PowerShell використовуються для зміни метрик підключень,

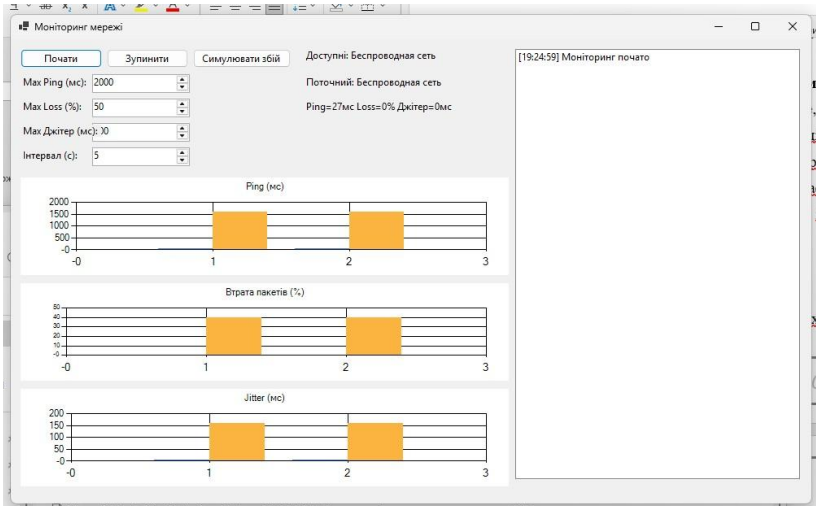
WMI забезпечує моніторинг IP-адрес та стану мережевих адаптерів

.NET API дозволяє виконувати асинхронне опитування мережі. Для реалізації графічного інтерфейсу використовується WinForms із компонентом DataGridView.

Алгоритм роботи застосунку



Інтерфейс користувача



Реалізація перемикання

Результати тестування

Сценарій	Результат
Втрата Wi-Fi сигналу	Перемикання на Ethernet
Штучне підвищення ping	Перемикання на провайдера 2
Зникнення IP	Перемикання на провайдера 1

Висновки і перспективи

Реалізовано стабільне автоматичне перемикання між мережами.
Система має зрозумілий інтерфейс, підтримує логування та не потребує ручного втручання.
Подальші плани: мобільна версія, інтеграція з хмарними службами, підтримка VPN.
Рішення придатне як для звичайних користувачів, так і для корпоративного середовища.

Дякую за увагу

ДОДАТОК Б. Код Логіки програми

Form1.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace NetworkMonitorWinForms
{
    public partial class Form1 : Form
    {
        private Timer timer;
        private int maxPing = 200;
        private int maxPacketLoss = 30;
        private int minSignalStrength = 50;
        private Random rnd = new Random();
        private int fakePingCounter = 0;
        private int timerTickCount = 0;
        private int currentInterfaceIndex = 0;

        private NetworkInterface[] activeInterfaces = new NetworkInterface[2];
        private string[] interfaceNames = new string[2];
        private string logFilePath;

        public Form1()
        {
            InitializeComponent();

            chart1.Series.Clear();
            chart1.Series.Add("Interface1");
            chart1.Series["Interface1"].ChartType                                    =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
            chart1.Series["Interface1"].BorderWidth = 2;
            chart1.Series["Interface1"].Color = System.Drawing.Color.Blue;

            chart1.Series.Add("Interface2");
            chart1.Series["Interface2"].ChartType                                    =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
            chart1.Series["Interface2"].BorderWidth = 2;
            chart1.Series["Interface2"].Color = System.Drawing.Color.Orange;

            timer = new Timer();
            timer.Interval = 1000;
            timer.Tick += Timer_Tick;

            DetectInterfaces();
            SetupLogFile();
        }

        private void SetupLogFile()
        {
            string  logDir  =  Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"logs");
            Directory.CreateDirectory(logDir);
            string date = DateTime.Now.ToString("yyyy-MM-dd");
```

```
            logFilePath = Path.Combine(logDir, $"{date}_log.txt");
        }

        private void DetectInterfaces()
        {
            var all = NetworkInterface.GetAllNetworkInterfaces()
                .Where(ni => ni.OperationalStatus == OperationalStatus.Up &&
                             (ni.NetworkInterfaceType                                  ==
NetworkInterfaceType.Wireless80211 ||
                              ni.NetworkInterfaceType                                  ==
NetworkInterfaceType.Ethernet))
                .ToArray();

            if (all.Length > 0) activeInterfaces[0] = all[0];
            if (all.Length > 1) activeInterfaces[1] = all[1];

            interfaceNames[0] = activeInterfaces[0]?.Name ?? "Ethernet1";
            interfaceNames[1]    =    activeInterfaces[1]?.Name    ??    "Ethernet3
(эмуляция)";

            lblInterface.Text = $"Интерфейс 1: {interfaceNames[0]}\nИнтерфейс 2:
{interfaceNames[1]}";
        }

        private void Timer_Tick(object sender, EventArgs e)
        {
            timerTickCount++;
            int ping1 = rnd.Next(30, 60);
            if (timerTickCount >= 30)
                ping1 += rnd.Next(250, 350);

            int ping2 = activeInterfaces[1] != null
                        ? PingViaInterface("8.8.8.8", activeInterfaces[1])
                        : (fakePingCounter < 15 ? rnd.Next(50, 150) : rnd.Next(250,
350));

            if (activeInterfaces[1] == null) fakePingCounter++;

            chart1.Series["Interface1"].Points.AddY(ping1);
            chart1.Series["Interface2"].Points.AddY(ping2);

            if (chart1.Series["Interface1"].Points.Count > 60)
                chart1.Series["Interface1"].Points.RemoveAt(0);
            if (chart1.Series["Interface2"].Points.Count > 60)
                chart1.Series["Interface2"].Points.RemoveAt(0);

            int previousIndex = currentInterfaceIndex;
            currentInterfaceIndex = (ping1 > maxPing && activeInterfaces[1] != null)
? 1 : 0;

            if (currentInterfaceIndex != previousIndex)
                Log($"Перемикання     с      {interfaceNames[previousIndex]}      на
{interfaceNames[currentInterfaceIndex]}");

            int activePing = currentInterfaceIndex == 0 ? ping1 : ping2;
            int signal = GetWifiSignalStrength();

            lblStatus.Text = (activePing > maxPing || signal < minSignalStrength)
                             ? "Warning: Poor connection!"
                             : "Connection OK.";
```

```
            lblActiveConnection.Text     =     "Текущее     подключение:     "     +
interfaceNames[currentInterfaceIndex];

            Log($"Interface:   {interfaceNames[currentInterfaceIndex]}   |   Ping:
{activePing}ms | Signal: {signal}% | Active: Interface{currentInterfaceIndex + 1}");
        }

        private void Log(string text)
        {
            string time = DateTime.Now.ToString("[HH:mm:ss]");
            File.AppendAllText(logFilePath,                                  $"{time}
{text}{Environment.NewLine}");
        }

        private int PingViaInterface(string host, NetworkInterface nic)
        {
            try
            {
                var ipProps = nic?.GetIPProperties();
                var addr = ipProps?.UnicastAddresses
                            .FirstOrDefault(ip   =>   ip.Address.AddressFamily   ==
AddressFamily.InterNetwork);
                if (addr == null) return 0;

                Ping ping = new Ping();
                PingOptions options = new PingOptions();
                byte[] buffer = new byte[32];
                int timeout = 1000;

                PingReply reply = ping.Send(host, timeout, buffer, options);
                return (int)(reply.RoundtripTime);
            }
            catch { return 0; }
        }

        private int GetWifiSignalStrength()
        {
            try
            {
                ProcessStartInfo  psi  =  new  ProcessStartInfo("netsh",  "wlan  show
interfaces");
                psi.RedirectStandardOutput = true;
                psi.UseShellExecute = false;
                psi.CreateNoWindow = true;
                var process = Process.Start(psi);
                string output = process.StandardOutput.ReadToEnd();
                var match = Regex.Match(output, @"Signal\s*:\s*(\d+)");
                if (match.Success)
                {
                    return int.Parse(match.Groups[1].Value);
                }
            }
            catch { }
            return 0;
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            int.TryParse(txtMaxPing.Text, out maxPing);
            int.TryParse(txtMaxLoss.Text, out maxPacketLoss);
            int.TryParse(txtMinSignal.Text, out minSignalStrength);
            timer.Start();
```

```
        }
    }
```

Program.cs

```
using System;
using System.Windows.Forms;

namespace NetworkMonitorWinForms
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Form1.Designer.cs

```
namespace NetworkMonitorWinForms
{
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.Windows.Forms.Button btnStart;
        private System.Windows.Forms.TextBox txtMaxPing;
        private System.Windows.Forms.TextBox txtMaxLoss;
        private System.Windows.Forms.TextBox txtMinSignal;
        private System.Windows.Forms.Label lblStatus;
        private System.Windows.Forms.Label lblInterface;
        private System.Windows.Forms.Label lblActiveConnection;
        private System.Windows.Forms.Label lblMaxPing;
        private System.Windows.Forms.Label lblMaxLoss;
        private System.Windows.Forms.Label lblMinSignal;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null)) components.Dispose();
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.chart1                                =                                new
System.Windows.Forms.DataVisualization.Charting.Chart();
            this.btnStart = new System.Windows.Forms.Button();
            this.txtMaxPing = new System.Windows.Forms.TextBox();
            this.txtMaxLoss = new System.Windows.Forms.TextBox();
            this.txtMinSignal = new System.Windows.Forms.TextBox();
            this.lblStatus = new System.Windows.Forms.Label();
            this.lblInterface = new System.Windows.Forms.Label();
            this.lblActiveConnection = new System.Windows.Forms.Label();
            this.lblMaxPing = new System.Windows.Forms.Label();
            this.lblMaxLoss = new System.Windows.Forms.Label();
            this.lblMinSignal = new System.Windows.Forms.Label();

            ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
            this.SuspendLayout();
```

```
            var chartArea = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            chartArea.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea);
            this.chart1.Location = new System.Drawing.Point(12, 12);
            this.chart1.Name = "chart1";
            this.chart1.Size = new System.Drawing.Size(760, 300);

            this.btnStart.Location = new System.Drawing.Point(670, 320);
            this.btnStart.Size = new System.Drawing.Size(100, 25);
            this.btnStart.Text = "Старт";
            this.btnStart.Click += new System.EventHandler(this.btnStart_Click);

            this.lblMaxPing.Location = new System.Drawing.Point(12, 320);
            this.lblMaxPing.Size = new System.Drawing.Size(140, 23);
            this.lblMaxPing.Text = "Макс. пінг (мс):";

            this.txtMaxPing.Location = new System.Drawing.Point(150, 320);
            this.txtMaxPing.Size = new System.Drawing.Size(50, 23);
            this.txtMaxPing.Text = "200";

            this.lblMaxLoss.Location = new System.Drawing.Point(210, 320);
            this.lblMaxLoss.Size = new System.Drawing.Size(140, 23);
            this.lblMaxLoss.Text = "Макс. втрата (%):";

            this.txtMaxLoss.Location = new System.Drawing.Point(350, 320);
            this.txtMaxLoss.Size = new System.Drawing.Size(50, 23);
            this.txtMaxLoss.Text = "30";

            this.lblMinSignal.Location = new System.Drawing.Point(410, 320);
            this.lblMinSignal.Size = new System.Drawing.Size(170, 23);
            this.lblMinSignal.Text = "Мін. сигнал Wi-Fi (%):";

            this.txtMinSignal.Location = new System.Drawing.Point(580, 320);
            this.txtMinSignal.Size = new System.Drawing.Size(50, 23);
            this.txtMinSignal.Text = "50";

            this.lblStatus.Location = new System.Drawing.Point(12, 350);
            this.lblStatus.Size = new System.Drawing.Size(500, 23);
            this.lblStatus.Text = "Статус:";

            this.lblInterface.Location = new System.Drawing.Point(12, 380);
            this.lblInterface.Size = new System.Drawing.Size(600, 30);
            this.lblInterface.Text = "Інтерфейси:";

            this.lblActiveConnection.Location = new System.Drawing.Point(12, 410);
            this.lblActiveConnection.Size = new System.Drawing.Size(600, 23);
            this.lblActiveConnection.Text = "Поточне підключення:";

            this.ClientSize = new System.Drawing.Size(784, 450);
            this.Controls.Add(this.chart1);
            this.Controls.Add(this.btnStart);
            this.Controls.Add(this.lblMaxPing);
            this.Controls.Add(this.txtMaxPing);
            this.Controls.Add(this.lblMaxLoss);
            this.Controls.Add(this.txtMaxLoss);
            this.Controls.Add(this.lblMinSignal);
            this.Controls.Add(this.txtMinSignal);
            this.Controls.Add(this.lblStatus);
            this.Controls.Add(this.lblInterface);
            this.Controls.Add(this.lblActiveConnection);
            this.Text = "Монітор мережі";
```

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Цюсьмака Миколи Андрійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Обслуговування комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Нестеренко Володимир Дмитрович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання*

Обсяг розрахунково-пояснювальної записки *71* сторінок

Обсяг графічної (презентаційної) частини *12* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації,

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 12 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Охоплено всі етапи Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання*

Забезпечено аналіз поточних даних від мережевих адаптерів, їх обробки та реалізація зміни пріоритету підключень

д) основні недоліки дипломного проекту

Не реалізовано можливості для відстеження результатів моніторингу за іншими IP-адресами. Не реалізована можливість самостійної зміни метрик через графічний інтерфейс. Деякі недоліки оформлення пояснювальної записки

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

«27» 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Цюсьмак Микола Андрійович

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Обслуговування комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 63 сторінок. У пояснювальній записці описано етапи Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Цюсьмак Микола поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Цюсьмак Микола під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Цюсьмак Микола показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як Microsoft Visual Studio, Microsoft PowerPoint, Microsoft Visio та ін.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Нестеренко Володимир Дмитрович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«23» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Цюсьмак Микола Андрійович
здобувач освіти гр. 4КС-58, та

Нестеренко Володимир Дмитрович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка пристрою діагностики стану ПК на платформі Arduino» (автор роботи – Цюсьмак М.А., керівник роботи – Нестеренко В.Д.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Цюсьмак М.А. /

Керівник _____ / Нестеркно В.Д /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КС-58

Цюсьмака Миколая Андрійовича

на тему *Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *23.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 20.06.2025 р. значення коефіцієнту подібності в роботі становить 13,92%, коефіцієнт цитування – 1,45%.*

(підпис) *23.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Цюсьмака М.А.* (П.І.Б.)

проведена « *23* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/20/2025
Дата редагування 6/20/2025

Документ прийнятий

Звіт подібності

метадані

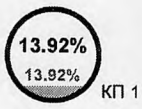
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка застосунку для моніторингу якості та автоматичного перемикання мережевого з'єднання

Автор
Цюсьмак Микола Андрійович
Науковий керівник / Експерт
Нестеренко Володимир Дмитрович

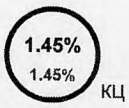
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25	15560	129762
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		33
Інтервали	A→	0
Мікропробіли	·	45
Білі знаки		0
Парафрази (SmartMarks)	a	133

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	124 0.80 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	70 0.45 %
3	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	58 0.37 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	57 0.37 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	54 0.35 %

6	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	47 0.30 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	43 0.28 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	42 0.27 %
9	https://blog.csdn.net/qq_39656138/article/details/90450672	41 0.26 %
10	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	40 0.26 %

з домашньої бази даних (0.07 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка анімованої веб-вікторини до 95-річчя ВСП "ОТФК ОНТУ" [redacted] 6/19/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	11 (2) 0.07 %

з програми обміну базами даних (0.25 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Диплом - Новаковский.docx [redacted] 6/18/2021 Kryvyi Rih National University (Кафедра моделювання і програмного забезпечення)	23 (3) 0.15 %
2	ООП 2024 ІПЗС11 Кундля Б.Р [redacted] 12/7/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	10 (1) 0.06 %
3	content-110-1718-21728-Qh6NEDCu.txt [redacted] 5/11/2023 Yuriy Fedkovych Chernivtsi National University(CNU) course papers (Deanery)	6 (1) 0.04 %

з Інтернету (13.60 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download	575 (51) 3.70 %
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**410 (11) 2.63 %**
3	**https://blog.csdn.net/qq_39656138/article/details/90450672**	**126 (12) 0.81 %**
4	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**100 (4) 0.64 %**
5	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**90 (3) 0.58 %**
6	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**85 (5) 0.55 %**
7	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**82 (4) 0.53 %**
8	http://www.nfex.ru/index.php/Csharp/C_Sharp_by_API/System.Windows.Forms/Form	81 (11) 0.52 %
9	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**72 (3) 0.46 %**
10	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content**	**51 (2) 0.33 %**

11	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**47 (4) 0.30 %**
12	https://blog.csdn.net/u011555996/article/details/113665810	41 (4) 0.26 %
13	**https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download**	**40 (1) 0.26 %**
14	https://knowledge.allbest.ru/programming/3c0a65625a3ac69a4c53a89521206d26_0.html	40 (5) 0.26 %
15	https://card-file.ontu.edu.ua/bitstreams/11562741-24e6-4201-bc41-a00c8013fca1/download	37 (4) 0.24 %
16	https://videouroki.net/razrabotki/sobytiinoie-proghrammirovaniie.html	30 (3) 0.19 %
17	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**28 (1) 0.18 %**
18	https://social.msdn.microsoft.com/Forums/en-US/d66e0386-8870-4f19-be6e-391a7bd0f7e2/dialogbox-with-user-input	26 (2) 0.17 %
19	**https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download**	**26 (1) 0.17 %**
20	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	23 (3) 0.15 %
21	https://social.msdn.microsoft.com/forums/vstudio/tr-TR/c6efdc87-d44e-4579-8af1-71d5b037b359/gei-efekti	22 (4) 0.14 %
22	https://card-file.ontu.edu.ua/bitstreams/aed610a6-43ef-47e0-9066-e85c89456f3e/download	18 (2) 0.12 %
23	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	17 (1) 0.11 %
24	https://blog.csdn.net/ytuyzh/article/details/105534241	11 (2) 0.07 %
25	https://card-file.ontu.edu.ua/bitstreams/dfa57ac3-98fa-4c22-86e7-0549d1254d89/download	10 (1) 0.06 %
26	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	9 (1) 0.06 %
27	https://gist.github.com/gvaduha/b05f61d299cf1db9f6d3fdb8f8e42e75	7 (1) 0.04 %
28	https://eep8a.de/scripts/08_Formulare/	6 (1) 0.04 %
29	https://card-file.ontu.edu.ua/bitstreams/63ee88cb-a3d0-4005-9cf2-0cff89f28c0d/download	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Обслуговування комп'ютерних систем і мереж»
Група: 4КС- 58

Дипломний проєкт
здобувача **освіти денної форми навчання** КС. 58.20.000. ДП

ЦЮСЬМАКА
МИКОЛАЯ АНДРІЙОВИЧА

м. Одеса
2025 **р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»