Державний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Програмний модуль моніторингу комп'ютерної системи»

Студента 4 курсу, 7 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» — Васильєва Богдана Олександровича

_____
підпис студента

Науковий керівник, кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки. — Палагута Катерина Олексіївна

_____
підпис керівника

Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки — Рзаєва Світлана Леонідівна

_____
підпис гаранта

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Васильєва Богдана Олександровича
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмний модуль моніторингу комп'ютерної системи»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту є розробка зручного та ефективного програмного модуля моніторингу, що забезпечить користувачам простий та зрозумілий інтерфейс для контролю та аналізу стану комп'ютерної системи.

Об'єкт дослідження є інформаційно-технологічна інфраструктура та робота комп'ютерних систем.

Предмет дослідження є проектування та розробка функціональності програмного модуля, що дозволить користувачам зручно та ефективно контролювати та моніторити роботу комп'ютерної системи.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ПРОГРАМНОГО МОДУЛЮ

1.1. Характеристика роботи програми с моніторнгу комп'ютерних систем

1.2. Опис проблеми

1.3. Технічне завдання

1.4. Висновок до розділу 1

РОЗДІЛ 2. ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

2.1. Архітектура програми

2.2. Дизайн програми

2.3 Проектування програмного продукту

2.4. База даних програми

2.5. Висновок до розділу 2

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Розробка системи сихронизації бази даних та програми

3.2. Розробка основних функції програми

3.3 Розробка додаткових функції програми

3.4. Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз предметної області застосування програмного модулю*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Вибір програмних засобів та проєктування програмного модулю*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Реалізація програмного забезпечення*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	*02.06.2023*
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*	*20.06.2023*	*20.06.2023*

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Криворучко О.В
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Васильєв Б.О.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____ Васильєв Б.О.

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Рзаєва С.Л.

(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Робота присвячена розробці та реалізації програмного модуля моніторингу комп'ютерної системи. Метою роботи було створення зручного та ефективного інструменту для контролю та аналізу роботи комп'ютерних систем.

В результаті порівняльного аналізу аналогічних рішень визначено основні функціональні вимоги до програмного модуля. Для реалізації модуля використані сучасні технології та мови програмування, що дозволило досягти високої продуктивності та надійності системи. Проведено серію тестувань, які підтвердили функціональність та якість модуля.

Готовий програмний модуль моніторингу комп'ютерної системи було успішно протестовано. Отриманий модуль представляє собою ефективне рішення для користувачів, яке дозволяє зручно та швидко контролювати та аналізувати стан комп'ютерної системи. Результати роботи можуть бути використані для поліпшення ефективності, безперебійності та надійності роботи комп'ютерних систем.

Ключові слова: програмний модуль, моніторинг, комп'ютерна система, функціональні вимоги, архітектура, користувачі, інтерфейс, продуктивність, надійність, тестування, контроль, аналіз.

ABSTRACT

The work is devoted to the development and implementation of a computer system monitoring software module. A convenient and effective tool for monitoring and analyzing the operation of computer systems was created using the work method.

As a result of a comparative analysis of similar solutions, the main functional requirements for the software module were determined. For the implementation of the module, modern technologies and programming languages were used, which made it possible to achieve high performance and reliability of the system. A series of tests were conducted, which confirmed the functionality and quality of the module.

The finished computer system monitoring software module was successfully tested. The resulting module is an effective solution for users, which allows you to conveniently and quickly monitor and analyze the state of the computer system. The results of the work can be used to improve the efficiency, continuity and reliability of computer systems.

Key words: software module, monitoring, computer system, functional requirements, architecture, users, interface, performance, reliability, testing, control, analysis.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

SQL (Structured Query Language) - мова програмування для управління маніпулювання реляційними базами даних.

WinForms - це бібліотека графічного інтерфейсу для розробки програм під управлінням операційної системи Windows.

C# - мова програмування, розроблена Microsoft, яка поєднує можливості C++ та зручність використання мови Java.

БД (база даних) - це структурована колекція даних, організована та доступна для збереження, управління та маніпуляцій з використанням певної системи управління базами даних (СУБД).

СУБД (система управління базами даних) - програмне забезпечення для організації, керування та маніпуляцій з базами даних.

ЗМІСТ

ВСТУП **3**

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ПРОГРАМНОГО МОДУЛЮ **5**

1.1. Характеристика роботи програми с моніторнгу комп'ютерних систем 5

1.2. Опис проблеми 6

1.3. Технічне завдання 8

1.4. Висновки до розділу 1 10

РОЗДІЛ 2 ..... **11**

ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ **11**

2.1. Дизайн програми 11

2.2. Архітектура проекту 20

2.3. Проектування програмного продутку 21

2.4. База даних програми 24

2.4. Висновки до розділу 2 29

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ **31**

3.1. Розробка системи синхронизації бази даних та програми 31

3.2. Розробка основних функції програми 33

3.3. Розробка додаткових функціх програми 37

3.4. Висновки до розділу 3 38

ВИСНОВКИ ТА ПРОПОЗИЦІЇ **39**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **40**

ДОДАТКИ **41**

ВСТУП

Програмний модуль моніторингу комп'ютерної системи є важливою складовою сучасних інформаційних технологій, що забезпечують ефективне керування та контроль за роботою комп'ютерних систем. У контексті інформаційно-технологічних інфраструктур, моніторинг системи грає ключову роль у забезпеченні безперебійності та надійності роботи комп'ютерних ресурсів.

Об'єктом дослідження є інформаційно-технологічна інфраструктура та робота комп'ютерних систем.

Предметом дослідження є проектування та розробка функціональності програмного модуля, що дозволить користувачам зручно та ефективно контролювати та моніторити роботу комп'ютерної системи.

Основною метою проекту є розробка зручного та ефективного програмного модуля моніторингу, що забезпечить користувачам простий та зрозумілий інтерфейс для контролю та аналізу стану комп'ютерної системи.

Основними завданнями проекту є: аналіз вимог до програмного модуля, проектування архітектури системи моніторингу, розробка функціональності модуля, тестування та підтримка програмного модуля.

Практична значущість проекту полягає в покращенні ефективності та безперебійності роботи комп'ютерної системи, забезпеченні швидкого виявлення та вирішення проблем, а також у підвищенні загальної продуктивності та ефективності використання ресурсів.

Тема проекту є актуальною та добре вивченою в науковому та практичному аспекті. Існують різноманітні програмні модулі для моніторингу комп'ютерних систем, але зазначений модуль має специфічну спрямованість

на аналіз та контроль комп'ютерних ресурсів, що робить його унікальним у своєму роді. Модуль може бути корисним як системним адміністраторам, так і звичайним користувачам, які бажають забезпечити стабільну та ефективну роботу своєї комп'ютерної системи.

Дослідження та розробка програмного модуля моніторингу комп'ютерної системи має великий потенціал для практичного використання, а також для подальшого розвитку і вдосконалення у майбутньому. Задача розробки модуля передбачає вирішення складних технічних завдань, що дозволить розробнику ознайомитись з новими технологіями та методами, які можуть знадобитись у майбутньому.

Отже, розробка програмного модуля моніторингу комп'ютерної системи має великий потенціал для покращення ефективності та безперебійності роботи комп'ютерних систем, а також може сприяти подальшому розвитку та вдосконаленню інформаційних технологій в сфері моніторингу та управління комп'ютерними ресурсами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ПРОГРАМНОГО МОДУЛЮ

1.1. Характеристика роботи програми с моніторнгу комп’ютерних систем

У сучасному світі, де комп'ютери використовуються в різних сферах діяльності, від особистих комп'ютерів до корпоративних мереж, моніторинг систем ПК набуває особливої важливості. Завдяки програмам моніторингу, адміністратори та користувачі можуть відстежувати стан комп'ютерів, забезпечувати безпеку, виявляти проблеми та забезпечувати ефективну роботу систем.

Однак існуючі програми моніторингу систем ПК мають свої обмеження та недоліки, що вимагають подальшого вдосконалення. Наприклад, деякі програми не надають детальної інформації про ресурси системи, інші не підтримують достатньо широкий спектр функціональних можливостей, а деякі не забезпечують належного рівня безпеки та конфіденційності даних.

Таким чином, існує потреба у розробці програмного продукту, який би вирішував ці проблеми та надавав адміністраторам та користувачам надійний та зручний інструмент для моніторингу систем ПК.

Метою даної дипломної роботи є розробка програмного продукту для моніторингу систем ПК, який відповідатиме вимогам сучасних користувачів. Головною метою є створення зручного та функціонального інструменту, який дозволить адміністраторам та користувачам відстежувати стан

комп'ютерів, аналізувати ресурси системи, виявляти проблеми та забезпечувати ефективну роботу систем.

Програмний продукт розроблено з використанням мови програмування C# та технології WinForms для створення інтерфейсу користувача. Для зберігання даних про моніторинг систем ПК буде використовується технологія SQL для створення та керування базою даних. Цей підхід дозволить забезпечити ефективне зберігання та обробку великої кількості даних про ресурси систем, їх стан, активність та інші параметри[2].

1.2. Опис проблеми

Програми моніторингу систем ПК, відіграють важливу роль у сучасному інформаційному середовищі, де залежність від комп'ютерних систем та їх надійна робота стають все більш важливими для багатьох організацій та користувачів. Проте, при розробці та експлуатації таких програм виникають різні проблеми, які потребують уваги та вирішення.

Достовірність і точність зібраної інформації викликає значні проблеми. Програма моніторингу системи ПК повинна мати здатність збирати та відображати різноманітні дані (наприклад, використання ЦП, мережевий трафік, пам’ять, дисковий простір), але, враховуючи складність і різновиди комп’ютерних систем, інформація може бути відсутньою, неправильно прочитаною або іншим чином ненадійною. . Це може призвести до неправильних рішень користувача та неправильних результатів. Глибоке розуміння комп’ютерних систем, перевірка отриманої інформації та розробка точних алгоритмів моніторингу є важливими для вирішення цієї проблеми.

Іншою проблемою, з якою стикаються програми моніторингу систем ПК, є складність та неоднорідність взаємодії з різними типами систем. Кожна комп'ютерна система може мати свої особливості та вимоги щодо моніторингу. Наприклад, певні системи можуть використовувати специфічні API для доступу до інформації про систему, або мати власні формати даних для зберігання.

Різноманітні ПК-системи та їхні різні версії операційних систем, архітектур процесорів і апаратних конфігурацій створюють проблеми для програм моніторингу ПК. Універсальний підхід неможливий, що змушує розробників шукати рішення для програмного забезпечення, сумісного з низкою систем ПК.

Розробники повинні враховувати версії операційних систем, архітектури процесорів і пристроїв, забезпечуючи при цьому сумісність з різними драйверами. Існує також проблема швидкого та ефективного програмного забезпечення під час моніторингу системи ПК. Перевантаження системи та затримки можуть бути результатом значної кількості даних у реальному часі, які потребують аналізу. Дуже важливо створити оптимальні алгоритми збору, обробки та зберігання даних, щоб забезпечити швидкий і ефективний моніторинг систем ПК без погіршення продуктивності контрольованої системи.

Безпека є ще однією важливою проблемою для програм моніторингу систем ПК. Збір та збереження конфіденційних даних про систему може створювати потенційну загрозу безпеці. Важливо забезпечити надійне шифрування даних, контроль доступу та аутентифікацію користувачів, щоб

запобігти несанкціонованому доступу до цієї інформації. Розробники також повинні уважно відслідковувати та реагувати на потенційні загрози безпеки, такі як вторгнення, злом або шкідливе програмне забезпечення. Це включає в себе розробку механізмів виявлення аномальної активності, моніторинг мережевого трафіку, а також впровадження системи сповіщень та журналування подій для аналізу та відстеження потенційних інцидентів.

Програмне забезпечення для моніторингу систем ПК часто стикається з проблемою інтеграції з різноманітними джерелами даних і системами. Щоб отримати повний огляд працездатності системи, потрібна уніфікація інформації з багатьох каналів, включаючи операційну систему, драйвери, програмне забезпечення сторонніх розробників і журнали подій. Це, у свою чергу, може передбачати створення адаптерів для різноманітних джерел даних, оптимізацію протоколів стандартизації та нормалізації даних, а також розробку інтерфейсів, які забезпечують інтеграцію із зовнішніми системами.

1.3. Технічне завдання

Технічна специфікація (ТЗ) містить огляд специфікацій і вимог до майбутнього програмного продукту, який буде створено. Технічне завдання є важливою частиною роботи, яка служить для багатьох цілей:

Обсяг робіт програмного продукту визначається специфікацією, яка окреслює компоненти, що розробляються, встановлює необхідні вимоги та визначає очікувані функції.

Специфікація програмного забезпечення визначає очікування користувачів. Це набір різноманітних вимог до продукту, який може бути функціональним або нефункціональним. Перше стосується здатності програмного забезпечення вирішувати завдання, а друге включає такі властивості, як швидкість, безпека та надійність.

Включно з такими компонентами, як система, структура бази даних, інтерфейси користувача та залежності, архітектурні рішення викладені в технічному завданні.

Загальні відомості:

1.1. Найменування системи PC_Monitor:

1.1.1. Повне найменування системи: Система моніторингу стану комп'ютерів «MonitorPC».

1.1.2. Скорочене найменування системи: PCM.

1.2. Планові терміни початку та закінчення робіт: 01.04.2023 - 30.05.2023.

1.3. Порядок оформлення і пред’явлення результатів робіт: Програмний продукт та документація.

1.4. Головний бенефіціар та потенційні користувачі системи: Компанії, що займаються обслуговуванням комп'ютерів, IT-підрозділи компаній, приватні користувачі.

Мета та призначення створення системи:

2.1. Призначення системи: Моніторинг стану комп'ютерів для попередження виникнення проблем та швидкого реагування на них.

2.2. Мета створення системи: Забезпечення ефективного моніторингу та управління станом комп'ютерів, зниження витрат часу та коштів на обслуговування комп'ютерів, забезпечення безпеки та захисту від зловмисників.

Вимоги до системи:

3.1. Вимоги до системи в цілому: Система повинна забезпечувати швидкий та надійний моніторинг стану комп'ютерів.

Система повинна бути простою та зручною у використанні. Система повинна бути масштабованою та здатною працювати з великою кількістю комп'ютерів.

3.1.1. Вимоги до структури та функціонування системи, перелік підсистем: Система повинна забезпечувати моніторинг стану комп'ютерів за допомогою збору та аналізу інформації про комп'ютер, такої як: поточне навантаження, стан системних ресурсів, присутність вірусів.

1.4. Висновки до розділу 1.

У розділі «Аналіз предметної області застосування програмного модулю» був детально розібраний програмний продукти, в який буде використовуватися. Під час дослідиження були проаналізовані вимоги та потреби користувачів, основні аспекти, існуючі рішення та проблеми.

Функціональність і якість продукту значною мірою залежать від нашого розуміння його контексту. Завдяки аналізу сутностей ми отримали глибоке розуміння суті та контексту використання. Ми зосередилися на потребах користувачів, оскільки ми оцінювали їхні очікування та вимоги.

РОЗДІЛ 2

ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

2.1. Дизайн програми

З програмним забезпеченням для моніторингу системи ПК на основі WinForms ми стверджуємо, що інтерфейс користувача має бути інтуїтивно зрозумілим і простим у використанні. Інтерфейс має бути зрозумілим і слідувати логічній послідовності, дозволяючи користувачам швидко та ефективно знаходити системну інформацію.

ЦП, пам’ять, дискові процеси, кількість потоків – усе це повинно бути легко доступним на основному інтерфейсі. Загальна інформація про стан системи – це те, що має надавати екран.

Розробка програмного забезпечення для моніторингу системи ПК, створена в WinForms, має свої переваги, однією з яких є можливість перегляду історії показників за певний період часу. Це дає змогу користувачам отримати уявлення про продуктивність системи та визначити будь-які процеси, які можуть негативно вплинути на неї[6].

Простий і логічний інтерфейс, що дозволяє користувачам швидко та легко знайти потрібну інформацію про стан їх системи. Можливість перегляду історії показників за певний період часу, що дозволяє зрозуміти, як система працювала в минулому та з'ясувати, які процеси можуть впливати на її продуктивність.

Можливість налаштування системи оповіщення про критичні ситуації, що дозволяє користувачам бути в курсі стану своєї системи та вчасно приймати рішення.

Windows Forms, або WinForms, — це ефективний спосіб створення різноманітних графічних компонентів і керування ними для програм, що працюють у Microsoft Windows. Ця структура присвячена пропонуванню розширеного контролю та обробки вікон у графічному інтерфейсі. Це чудовий варіант для створення та адміністрування подій, GUI та інших елементів керування.

Створення модуля моніторингу для вашої комп'ютерної системи за допомогою технології WinForms є надзвичайно корисним подвигом. Цей модуль має можливість відстежувати численні змінні, такі як використання компонентів (тобто оперативна пам'ять, процесор, дисковий простір і підключення до мережі), а також загальний стан системи. За допомогою WinForms ви можете створити інтерактивний інтерфейс користувача, який буде одночасно зручним і ефективним для моніторингу стану вашої системи.

Модуль моніторингу, отриманий через WinForms, завдяки своїй потужності та гнучкості пропонує всі розширені функції, необхідні для розробки інтерактивного інтерфейсу для моніторингу комп'ютерної системи. Задоволення потреб користувача є кінцевою метою, і це відображається на зручності та ефективності модуля.

Графіки та діаграми: WinForms має потужні бібліотеки для створення графіків та діаграм, таких як лінійні графіки, кругові діаграми, стовпчасті діаграми тощо. Це дозволяє візуалізувати дані моніторингу в зручній формі.

Мінуси дизайну ПЗ моніторингу систем ПК, зробленого у WinForms, можуть включати:

1. Обмежені можливості стилізації та візуалізації інформації, що може зробити інтерфейс менш привабливим для користувачів.
2. Обмежена масштабованість програми, що може стати проблемою при обробці великої кількості даних або розширенні функціональності.
3. Обмежена підтримка крос-платформеності, що може зробити програму менш доступною для користувачів, які використовують інші операційні системи.
4. У цілому, дизайн ПЗ моніторингу систем ПК, зробленого у WinForms, має свої плюси та мінуси, і варто бути уважним при виборі програмного продукту для моніторингу своєї системи.

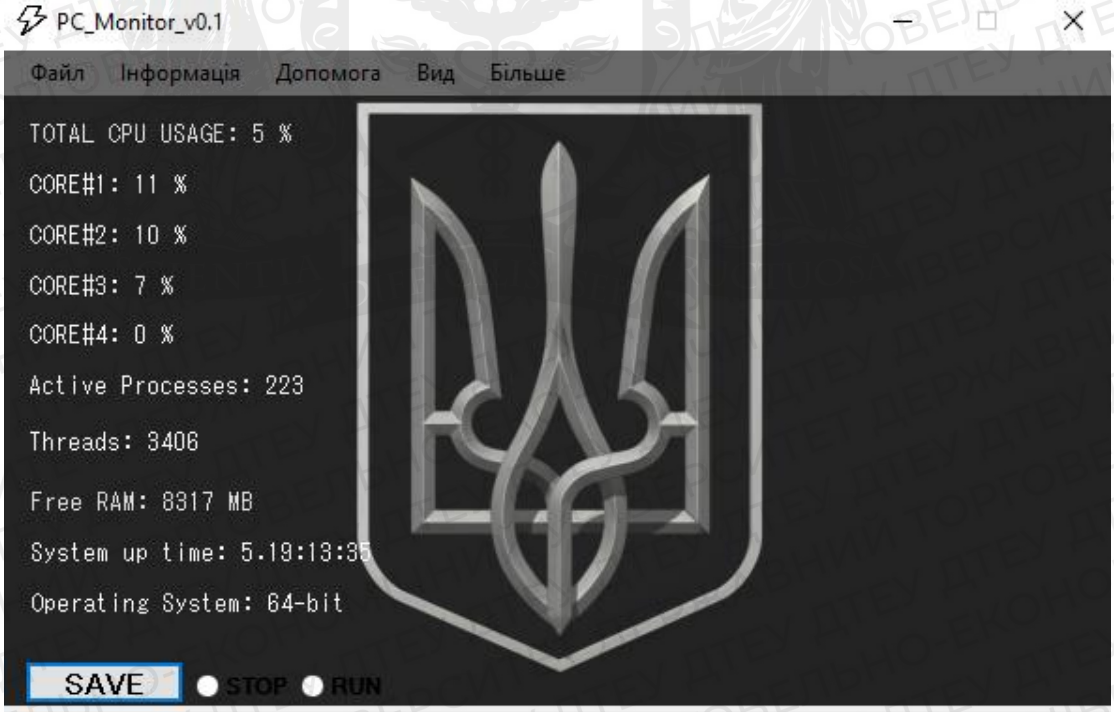


Рисунок 2.1. **Розроблений дизайн у WinForms**
Джерело: побудовано автором

Окремо розглянемо дизайн вікна авторизації, де користувач повинен ввести ключ для доступу до програми.

Позитивні сторони такого дизайну можуть включати:

1. Безпека: введення ключа є механізмом захисту програми від несанкціонованого доступу, що може бути важливим для деяких користувачів.
2. Простота використання: зазвичай вікно авторизації має простий та зрозумілий інтерфейс, що дозволяє користувачам швидко та легко ввести ключ.
3. Підтримка ліцензування: дизайн вікна авторизації дозволяє розробникам ПЗ легко контролювати кількість ліцензованих копій програми та обмежувати доступ до неї.

Однак, такий дизайн може мати деякі недоліки, такі як:

1. Неприємність для користувачів: введення ключа може бути додатковою перешкодою для користувачів, особливо якщо вони забули свій ключ або потребують авторизації декілька разів.
2. Вразливість: якщо ключ не зберігається в безпечному місці, він може бути вкрадений з ПК користувача та використаний для несанкціонованого доступу до програми.

У загальному, дизайн вікна авторизації з введенням ключа має свої сімлістичний дизайн, та досить зручний у використанні.

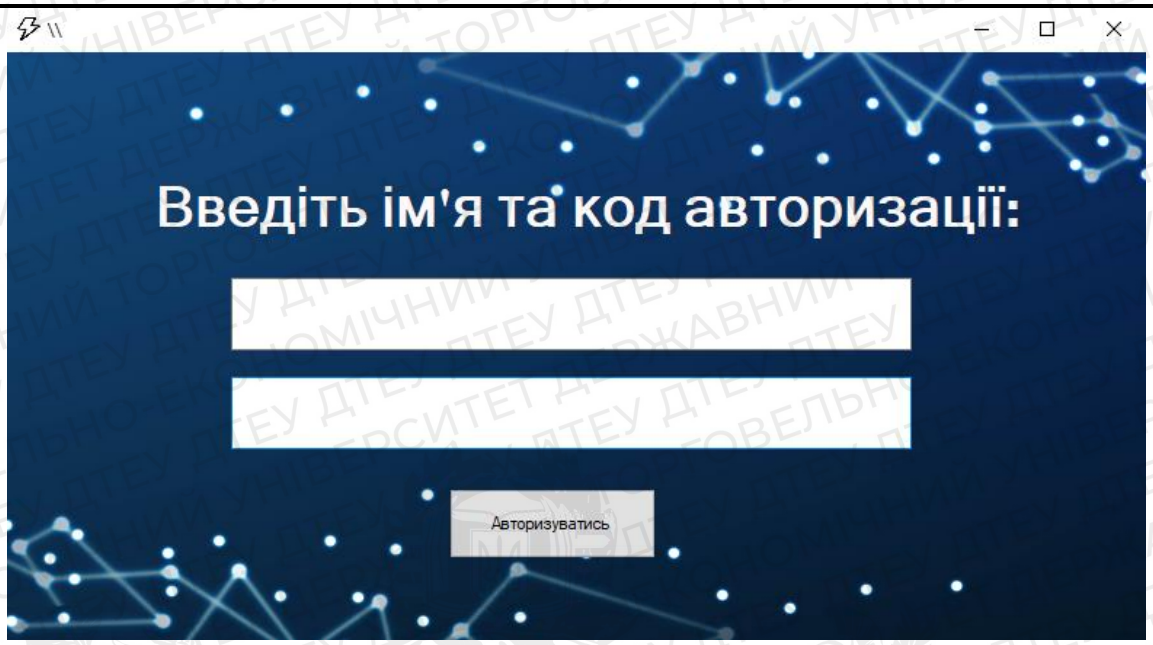


Рисунок 2.2. **Дизайн вікна авторизації**
Джерело: побудовано автором

Вікно, де будується графіки CPU Usage, є важливою частиною моніторингової системи ПК. Це вікно має такі елементи дизайну:

1. Графік: на графіку показуються значення відсоткового використання процесора на протязі часу.
2. Інформаційне поле: вікно містить інформаційне поле, що показує загальну нагрженість процессорв.

Позитивні сторони дизайну вікна з графіком CPU Usage можуть включати:

1. Візуалізація даних: графік може допомогти користувачам швидко визначити, які процеси використовують більше процесорного часу, що дозволяє їм ефективніше управляти процесами та забезпечувати оптимальну продуктивність ПК.
2. Доступність інформації: наявність інформаційних полів та легенди графіка дозволяє користувачам швидко визначати кількість запущених процесів та їх відсоткове використання процесора.

Проте, дизайн вікна з графіком CPU Usage може мати деякі недоліки, такі як:

1. Обмеження функціональності: це вікно може бути обмеженим у функціональності, що дозволяє користувачам лише відслідковувати використання процесора.
2. Зайнятість ресурсів: якщо програма постійно відображає графіки та інформаційні поля, це може займати більше ресурсів системи та призводити до повільної роботи ПК.
3. Складність для новачків: вікно може бути складним для новачків, оскільки вимагає розуміння термінології та функцій процесів, що використовують процесор.

Загалом, дизайн вікна з графіком CPU Usage є важливим елементом моніторингової системи ПК, який може допомогти користувачам ефективніше управляти процесами та забезпечувати оптимальну продуктивність ПК.

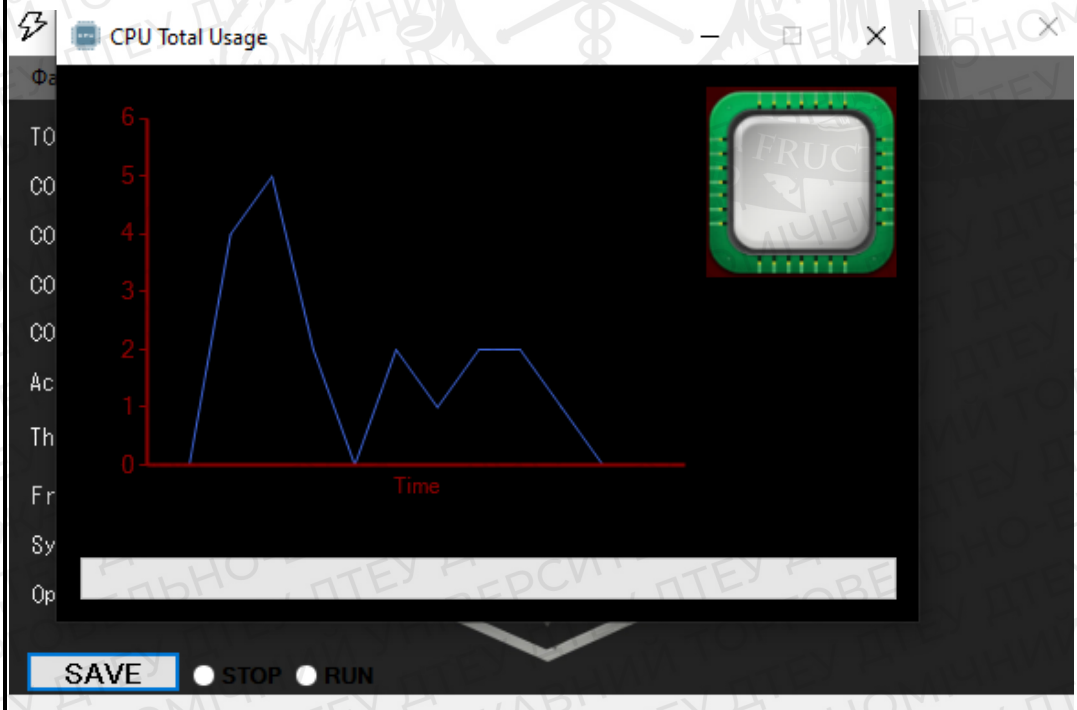


Рисунок 2.3. **Дизайн вікна графіків**
Джерело: побудовано автором

Головне вікно моніторингової системи ПК на базі WinForms містить інформацію про загальне використання CPU, кількість активних процесів та потоків, обсяг вільної оперативної пам'яті, час роботи системи та розрядність операційної системи. На головному вікні зазвичай розміщується графік, який відображає загальне використання процесора (Total CPU Usage) за останню секунду. Також, на цьому вікні відображена інформація про використання CPU по ядрам. Ця функція дозволяє користувачу зрозуміти, які процеси використовують більше ресурсів на кожному ядрі процесора. Також на головному вікні відображена кількість активних процесів та потоків. Ця інформація дозволяє користувачу зрозуміти, скільки процесів та потоків працює в системі в даний момент.

Додатково, на головному вікні відображена інформація про вільний обсяг оперативної пам'яті (Free Ram). Це дозволяє користувачу зрозуміти, скільки оперативної пам'яті залишилось в системі та скільки можна використовувати для запуску нових процесів. Окрім того, на головному вікні відображена інформація про час роботи системи (System Up Time). Ця інформація дозволяє користувачу зрозуміти, скільки часу працює система без перезавантаження.

Нарешті, на головному вікні відображена інформація про розрядність операційної системи. Це важлива інформація для користувача, оскільки вона дозволяє зрозуміти, які програми та процеси можуть бути використані на комп'ютері. Одним з головних плюсів такої моніторингової системи є можливість оперативного контролю та відстеження використання ресурсів комп'ютера. Також, моніторингова система ПК зроблена у WinForms має зручний та простий інтерфейс, що дозволяє користувачеві швидко зорієнтуватись у відображеній інформації та з легкістю користуватись всіма доступними опціями та налаштуваннями.

Незважаючи на свою ефективність, система моніторингу має обмежені можливості та функціональність порівняно з новими та більш продуктивними системами. Це може спричинити проблеми з продуктивністю та перешкодити майбутньому розширенню та модифікації програмного забезпечення під час використання WinForms.

Не дивлячись на ці обмеження, система моніторингу ПК, побудована з використанням WinForms, все ще може служити цінним інструментом для оперативного контролю та моніторингу використання ресурсів комп'ютера. Це дозволяє користувачеві відстежувати надмірне використання ресурсів і оперативно реагувати, що, у свою чергу, підвищує продуктивність і швидкість комп'ютера.

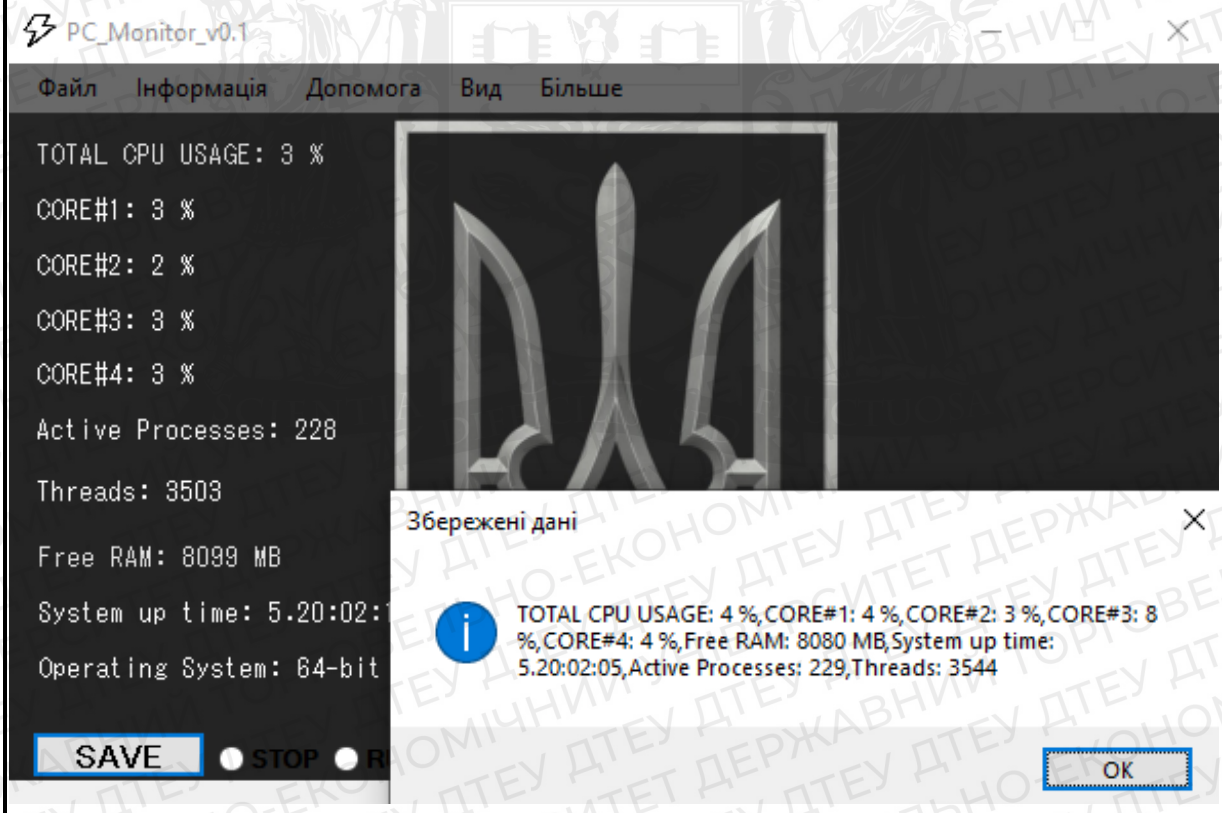


Рисунок 2.4. **Дизайн головного вікна**
Джерело: побудовано автором

Вікно інформації про жорсткий диск, в якому відображається інформація про Disk time, Disk read time, Disk write time, bytes read, bytes written, має наступний дизайн.

Зверху вікна розміщується заголовок "Інформація про жорсткий диск". Далі, по центру вікна, розміщується таблиця, яка містить наступні стовпці: "Disk time", "Disk read time", "Disk write time", "Bytes read", "Bytes written". Кожен з цих стовпців містить дані про відповідні показники роботи жорсткого диска.

Загалом, дизайн вікна інформації про жорсткий диск повинен бути зрозумілим та легко читатися користувачем. Він повинен допомагати відслідковувати роботу жорсткого диска та вчасно реагувати на проблеми.

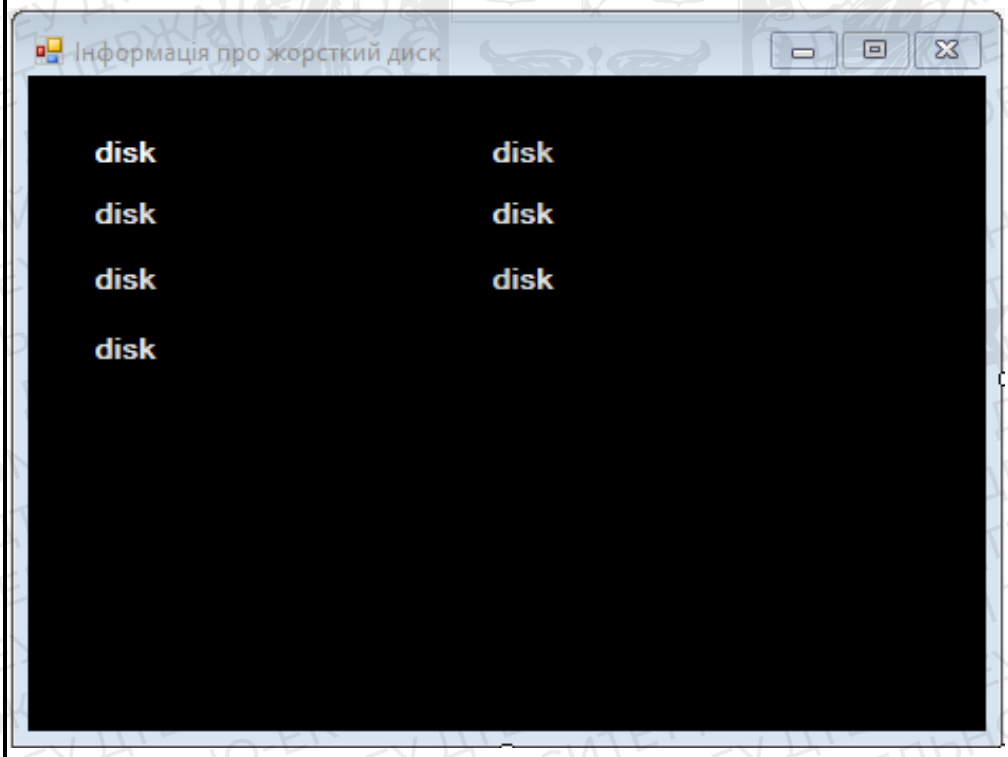


Рисунок 2.5. **Дизайн вікна інформації про жорсткий диск**
Джерело: побудовано автором

2.2. Архітектура проекту

Метою цього проекту є створення функціональних можливостей, які дозволяють користувачам відстежувати продуктивність системи ПК у реальному часі. Для цього використовується архітектура MVC (Model-View-Controller).

Model-View-Controller (MVC) — це архітектурний шаблон, який поділяє програму на три основні логічні компоненти: модель, представлення та контролер. Звідси і акронім MVC. Кожен компонент архітектури призначений для обробки певного аспекту розробки програми. MVC відокремлює бізнес-логіку та рівень презентації один від одного. Він традиційно використовується в графічних інтерфейсах користувача (GUI). Сьогодні архітектура MVC стала популярним способом розробки додатків і мобільних додатків[1].

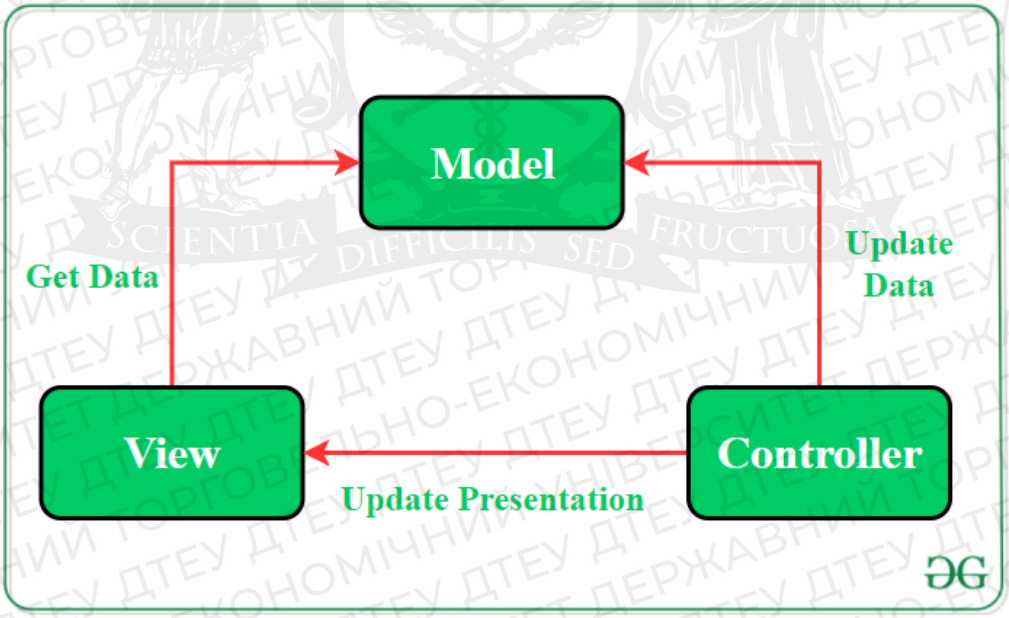


Рисунок 2.6. **Графічне зображення роботи MVC архітектури**
Джерело: побудовано автором

2.3. Проектування програмного продукту

Процес розробки програмного забезпечення охоплює серію дій, які спрямовані на встановлення внутрішньої роботи програми та навіть заглиблення в дрібниці її спостережуваних частин. Створення індивідуального проекту дозволяє розробникам отримати чітке уявлення про запланований курс дій, створити необхідну технологічну архітектуру та розробити комплексний набір методологій і тактик.

Люди часто плутають проектування з прототипом. Однак створення прототипу — це окремий процес, необхідний для розуміння візуальної естетики програми, розміщення елементів і того, як вони взаємодіють один з одним. З іншого боку, проектування передбачає створення комплексного плану дій, вибір представлення даних, архітектурних шаблонів і стилів, а також визначення стратегій для вирішення проблем. Складається технічна документація, встановлюються основні правила документації програмного забезпечення та вибираються методи тестування.

Виконується визначення відповідного стеку технологій і методи розробки для вашого проекту, а також необхідний розмір команди. Процес зменшує потенційні ризики під час розробки та усуває будь-які зайві дії, які споживають дорогоцінний час і ресурси. Розуміючи внутрішню роботу програми, визначаючи її потенціал для розширення та покращуючи її функціональність, ви можете отримати чітке розуміння того, як вона працює. Ведення документації на кожному етапі та створення детального плану дій допомагає мінімізувати конфлікти між замовником та виконавцем.

Для створення програмного забезпечення моніторингу системи ПК необхідно використання багатьох діаграм для розуміння архітектури, потоків даних і взаємодії між компонентами.

Діаграми класів є одними з найпоширеніших типів діаграм у програмному моделюванні. Це візуальне представлення класів, їхніх атрибутів, методів і зв’язків між ними.

Вона дозволяє визначити структуру системи, що розробляється, тобто її складові елементи, їх взаємодію та ієрархію успадкування. Класи можуть мати різні типи зв’язків, наприклад агрегацію, композицію, один до одного, один до багатьох, багато до багатьох тощо.

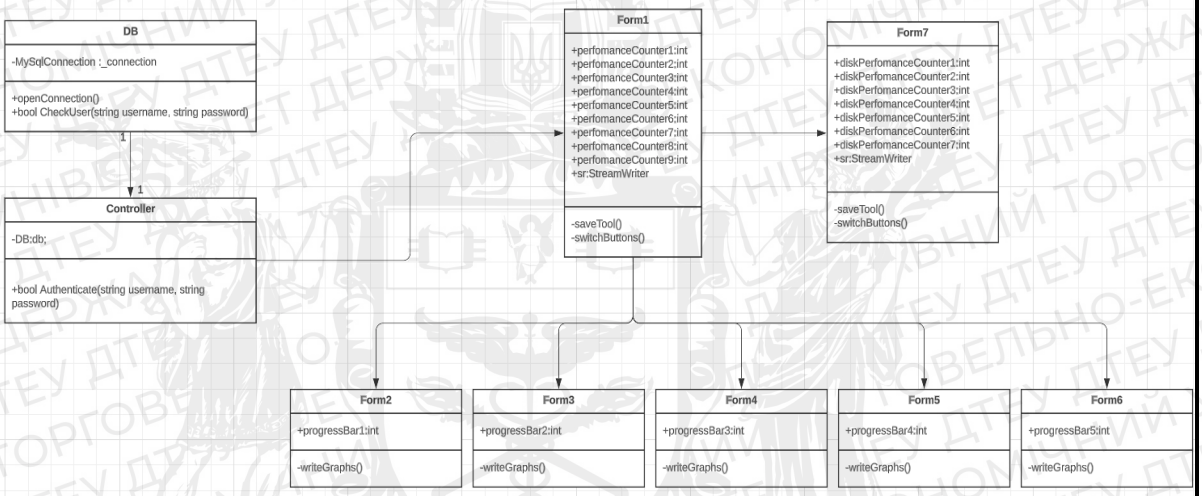


Рисунок 2.8. **Діаграмма класів**
Джерело: побудовано автором

Архітектурна діаграма клієнт-сервер — це графічне зображення, яке використовується для ілюстрації взаємодії між клієнтами (користувачами) і серверами в системі. Він демонструє, як клієнти та сервери обмінюються даними та взаємодіють один з одним у розподіленій архітектурі.

На схемі показані різні компоненти системи. Він може показувати фізичну або логічну структуру системи, залежно від контексту. Схеми архітектури сервера користувача допомагають візуалізувати архітектурні концепції системи та зрозуміти, як різні компоненти взаємодіють один з одним. Це корисний інструмент спілкування між розробниками

дизайнерами, архітекторами та іншими зацікавленими сторонами, який допомагає досягти консенсусу щодо системи та плану роботи.

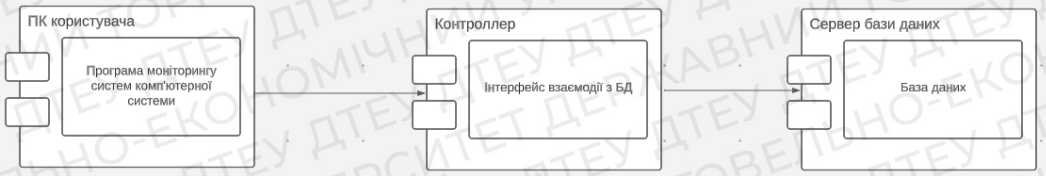


Рисунок 2.9. **Діаграма користувацько-серверної архітектури**
Джерело: побудовано автором

Еталонна модель програми — це діаграма, схема або концептуальна модель, яка визначає загальну структуру та поведінку програмного продукту. Це найвищий рівень абстракції, який визначає основні компоненти, взаємодії та поведінку програми без конкретної реалізації.

Еталонна модель програми, яка може бути представлена у вигляді діаграми, опису або специфікації, визначає основні компоненти програми, зв’язки між ними та основні правила роботи. Він може охоплювати такі аспекти, як архітектура, модульність, інтерфейси, логіка бізнес-процесів тощо.

Еталонна модель програми допомагає групі розробників і зацікавленим сторонам узгодити загальну концепцію програми, встановлює правила та принципи системи та забезпечує загальне розуміння проекту. Він служить основою для подальшого впровадження програмного продукту, визначає функціональні вимоги та допомагає уникнути проблем під час розробки та інтеграції компонентів.

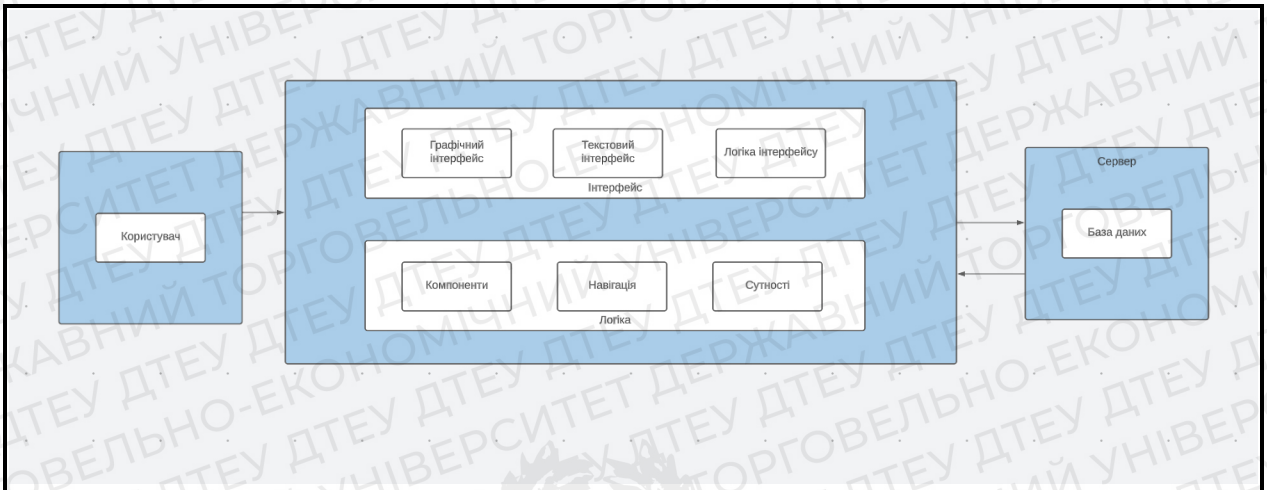


Рисунок 2.10. **Еталонна модель програми**
Джерело: побудовано автором

Еталонна модель програми допомагає групі розробників і зацікавленим сторонам узгодити загальну концепцію програми, встановлює правила та принципи системи та забезпечує загальне розуміння проекту. Він служить основою для подальшого впровадження програмного продукту, визначає функціональні вимоги та допомагає уникнути проблем під час розробки та інтеграції компонентів.

2.4. База даних програми

База даних програми системного моніторингу ПК містить таблицю, в якій зберігаються дані про користувачів програми. Ця таблиця містить лише логіни та паролі.

ID є унікальним ідентифікатором для кожного користувача в системі і використовується для ідентифікації користувача в базі даних. Пароль — це конфіденційний код, який користувач вводить під час входу в програму моніторингу системи ПК. Після введення правильного ідентифікатора та пароля користувачі можуть отримати доступ, щоб переглядати та аналізувати дані про системи своїх ПК[5].

Система моніторингу системи покладається на ідентифікатор і пароль для автентифікації користувача та безпеки ідентифікації. Унікальні ідентифікатори (ID) призначаються користувачам під час реєстрації, як правило, це комбінація букв або цифр. Потім ці ідентифікатори підключаються до профілю користувача, що зберігається в базі даних системи.

Під час реєстрації або під час першого входу користувач вибирає конфіденційний код, відомий як пароль. Користувач повинен ввести цей конфіденційний код під час наступних входів, щоб підтвердити свою особу. Рекомендується використовувати надійний пароль, який складається з комбінації літер, цифр і спеціальних символів, щоб зменшити ризик отримання зловмисниками доступу.

Система моніторингу системи ПК розблоковує пост-автентифікацію користувача, надаючи доступ до складних даних щодо функцій і поведінки комп’ютера. Інформація про процесор, оперативну пам’ять, диск.

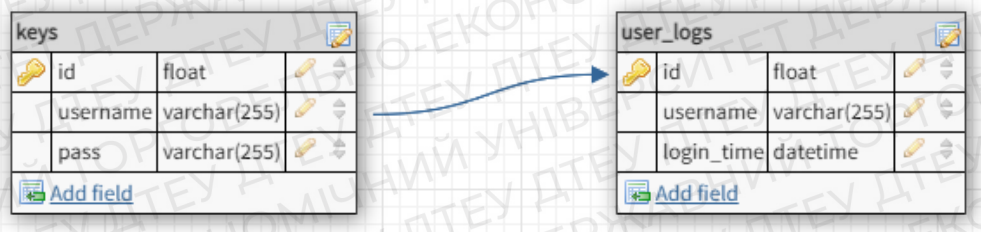


Рисунок 2.11. **Логічна модель бази даних**
Джерело: побудовано автором

У таблиці «Час входу користувача» зберігається інформація про час входу користувача в систему. Обидві таблиці пов'язані полем імені користувача, що дозволяє пов'язувати записи в цих таблицях. Ця логічна схема бази даних є основою для розробки програми, яка забезпечуватиме автентифікацію користувачів і відстежуватиме час їх входу.

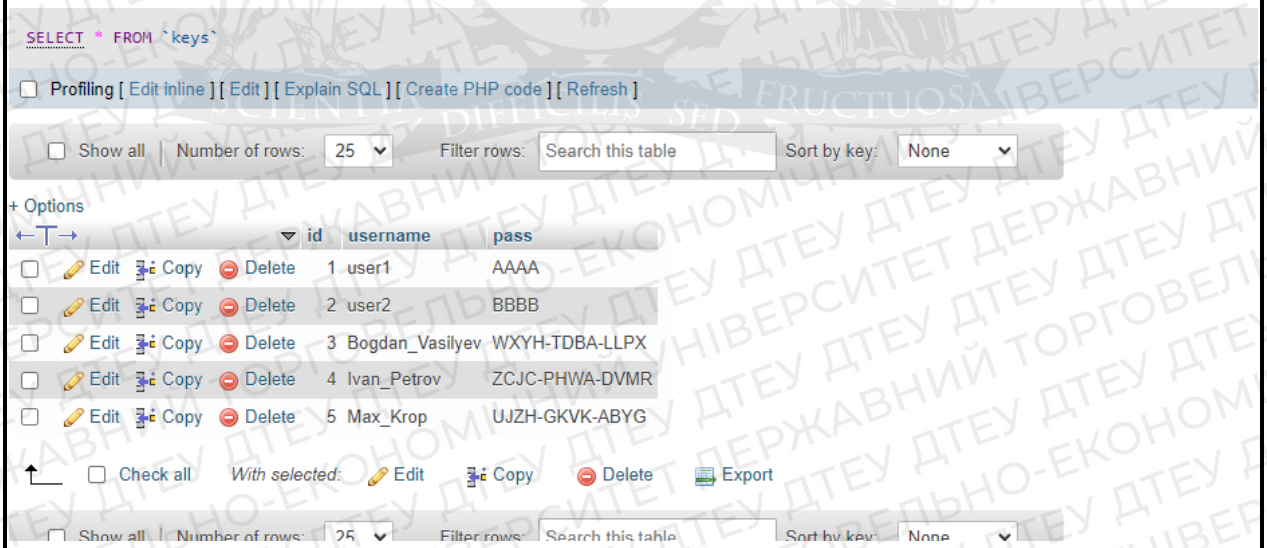


Рисунок 2.12. **Таблиця для зберігання данних користувача**
Джерело: побудовано автором

Таблиці для зберігання часу входу користувачів є важливою частиною логічної схеми бази даних з кількох причин:

1. Аналіз активності користувачів: економія часу входу дозволяє відстежувати дії користувачів і встановлювати моделі поведінки. Цю інформацію можна використовувати для аналізу активності, залученості користувачів і визначення популярного часу.
2. Зберігаючи записи про вхід користувачів, ви можете підвищити безпеку системи. Ці дані допомагають ідентифікувати заборонений вхід, зламані облікові записи або підозрілу поведінку. Перехресно посилаючись на ці дані, ви можете визначити, які дії користувачі виконали в системі, і перевірити будь-які інциденти на потенційні загрози.
3. Функція журналу підтримки та статистики: час входу дозволяє відстежувати та записувати операції користувача в системі. Це можна використовувати для створення журналів подій, відновлення даних, відстеження змін і статистичного аналізу активності користувачів[3].

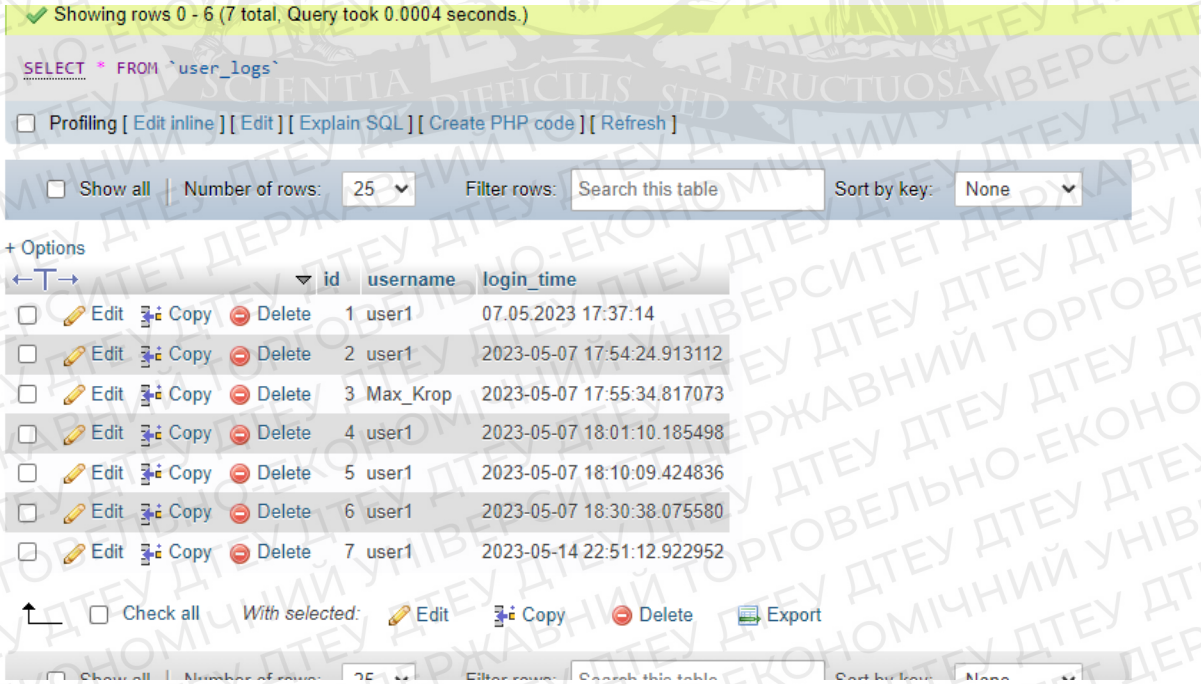


Рисунок 2.13. **Таблиця для зберігання часу входу користувача**
Джерело: побудовано автором

Вікно авторизації – це інтерфейс, який відображається, коли користувач намагається запустити програму моніторингу системи ПК. У цьому вікні користувач повинен ввести ключ, що відповідає його ключу з бази даних. Вікно авторизації може містити поле для введення коду і кнопку «увійти». Якщо ключ неправильний, системний монітор ПК не запуститься, і користувач повинен повторити вхід із правильним ідентифікатором і паролем. Крім того, забуті паролі можна відновити за допомогою відповідних посилань або інших механізмів, які дозволяють користувачам відновити доступ до своїх облікових записів, якщо їх забули.

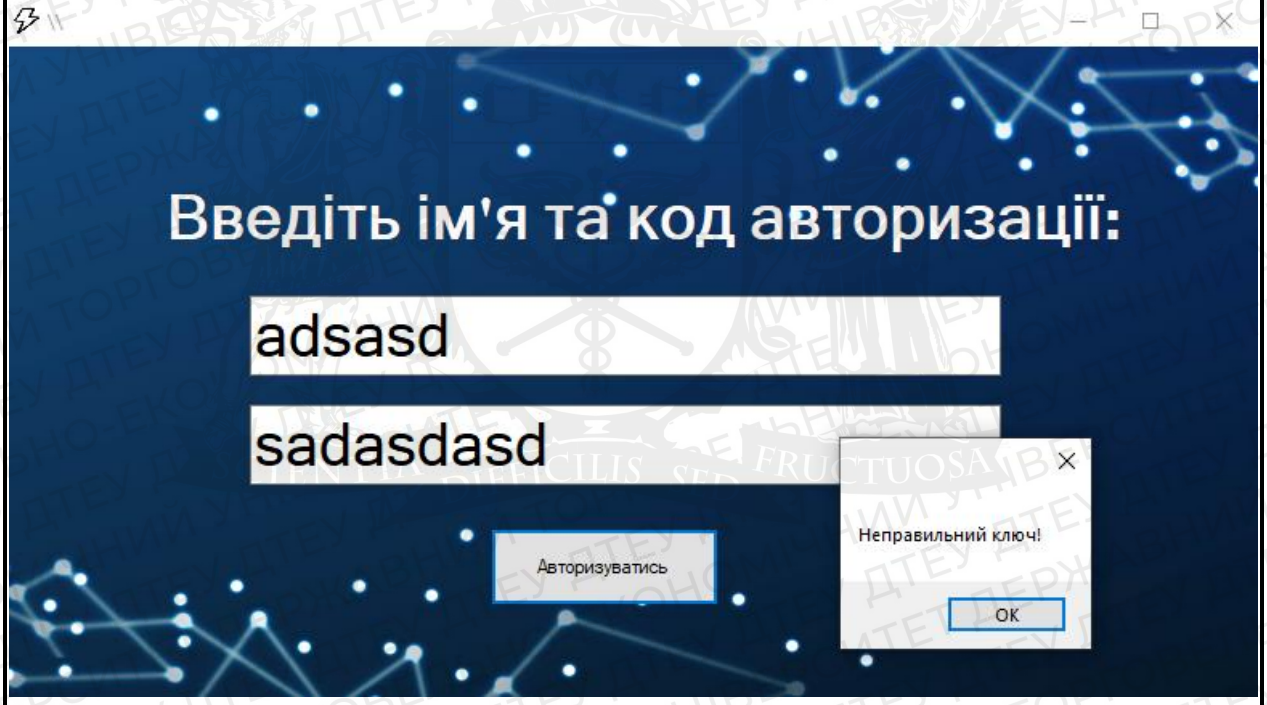


Рисунок 2.14. **Демонстрація введення невалідного ключа**
Джерело: побудовано автором

Усі дані, які вводяться в це вікно, повинні бути передані зашифрованими на сервер бази даних, щоб забезпечити безпеку даних та уникнути можливого несанкціонованого доступу до системи.

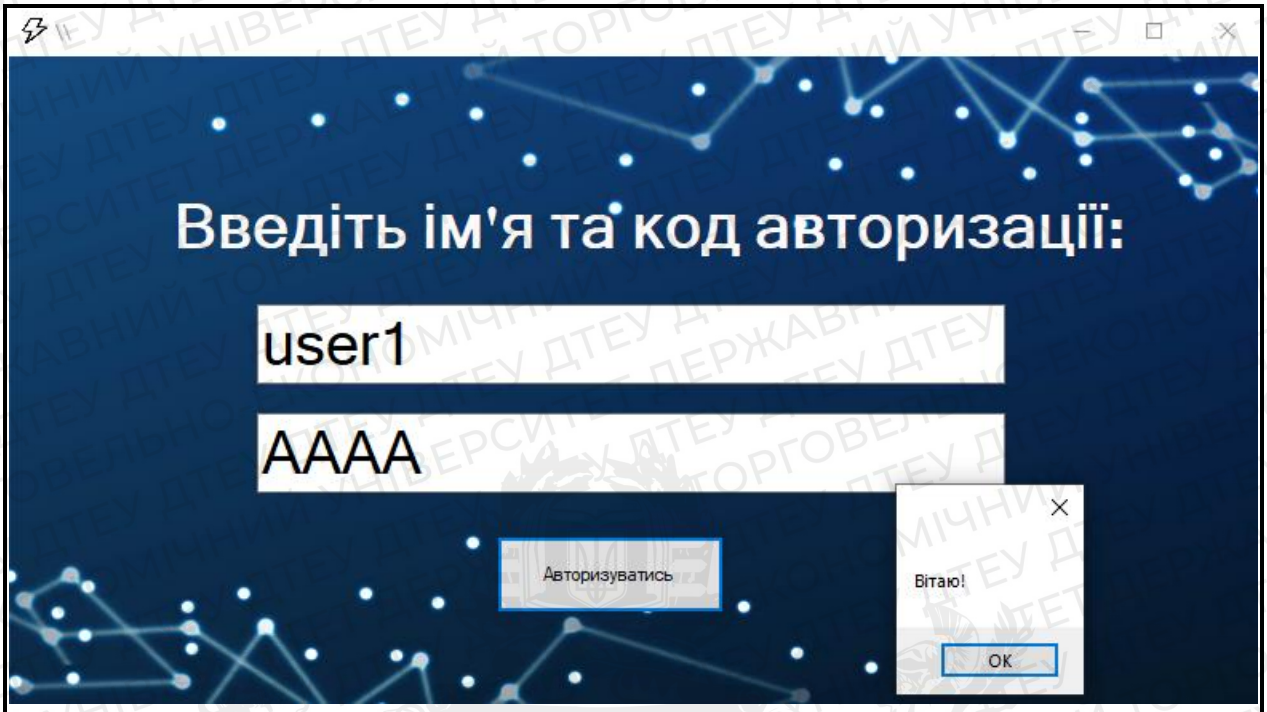


Рисунок 2.15. **Демонстрація введення валідного ключа**
Джерело: побудовано автором

2.4. Висновки до розділу 2.

Розробляючи план структури проекту та взаємодії компонентів, було розроблено архітектуру проекту. Це включало відображення клієнтського інтерфейсу, серверного компонента та модулів обробки даних. Наша зосередженість як на зручності використання, так і на функціональності призвела до розробки інтерфейсу користувача, який враховував і те, і інше. Ми також розробили структуру бази даних, яка враховує сутності, зв'язки та властивості для кожної таблиці. Нарешті ми інтегрували системи та компоненти. Безпечна, масштабована та ефективна розробка проекту була кінцевою метою, що призвело до серії дій. У процесі розробки проекту, приділялась увага аспектам безпеки. Були впроваджені механізми аутентифікації та авторизації, щоб гарантувати, що лише правомірні користувачі матимуть доступ до системи і її функціональності. Застосування шифрування даних та інших методів захисту було важливою складовою

безпечності проекту. Оскільки було прагнення до масштабованості, архітектура була ретельно спроектована, щоб забезпечити її гнучкість та легкість розширення. Були використані підходи, такі як мікросервісна архітектура, щоб розділити різні компоненти системи на окремі модулі. Це дозволило горизонтально масштабувати окремі компоненти при необхідності, що сприяє ефективному розподілу навантаження та забезпечує стабільну роботу системи під час зростання обсягів даних або користувачів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
3.1. Розробка системи синхронизації бази даних та програми

Програма починає свою роботу з вікна авторизації, де користувач має ввести свої данні і завдяки наступному скрипту відбувається перевірка чи були введені дані коректними. Даний скрипт перевіряє по базі даних, чи існує взагалі такий користувач і якщо все вірно, допускає до роботи з програмою, якщо ні то з'являється вікно з помилкою.

У формі можна знайти різні елементи керування, такі як кнопки, смуги прокрутки, списки, таблиці тощо. Ці елементи керування можна розташувати в різних контейнерах, таких як таблиці, групи та панелі, для відображення складних макетів елементів. Різні засоби керування для полегшення взаємодії з користувачем і виконання дій можуть бути включені в програму через форму заявки. Нижче наведено кілька таких елементів керування.

Певні команди або дії можна виконувати за допомогою кнопок. Ці кнопки можуть включати «Зберегти», «Видалити», «Надіслати» тощо.

Щоб переглянути всі дані або вміст форми, користувач може використовувати смуги прокручування, коли доступного місця не вистачає. Вони дозволяють здійснювати вертикальне прокручування вгору та вниз. Вибір користувача з певних параметрів доступний за допомогою списків із такими параметрами, як розкривний список або кілька. Прикладами є розкривний список країн або список пунктів меню.

З’єднання з базою даних можна встановити шляхом створення класу, який задає параметри та ініціалізує з’єднання. Щоб гарантувати належне підключення, розробник може встановити адресу сервера, пароль і дані для входу. Наприклад, це може зробити конструктор.

Вхідні дані можуть бути перевірені методом, наданим цим класом. Після того, як користувач введе свій логін і пароль, викликається метод. Ця перевірка включає перевірку даних у базі даних і виконання інших необхідних перевірок для забезпечення правильності.

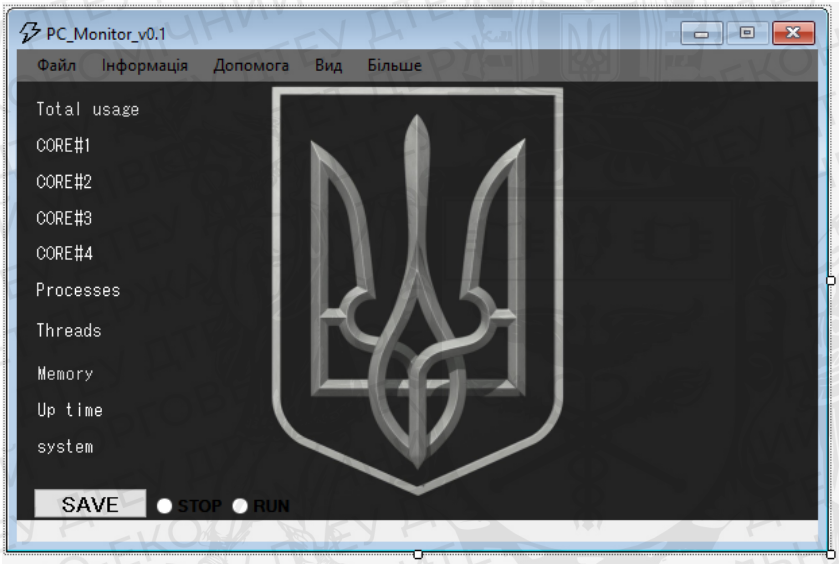
Після завершення перевірки та схвалення користувача йому надається доступ до головного інтерфейсу програми. Потім вони матимуть можливість використовувати функції програми, виконувати кілька дій і отримувати відгуки від програми. Після успішного входу користувач не матиме жодних обмежень щодо наданого доступу. По суті, вони матимуть повну можливість використовувати всі послуги та переваги, які пропонує програма.

Виконання різноманітних дій і використання різних функцій програми без обмежень стає можливим після того, як користувач увійшов у систему та отримав доступ до основного інтерфейсу. Взаємодія з інтерфейсом, введення даних, виконання операцій і отримання результатів від програми – все це можна зробити легко.

Після доступу до основного інтерфейсу програми через успішний вхід, здатність користувача взаємодіяти та виконувати дії залежить від наданих йому прав доступу. Однак після надання доступу користувач матиме необмежену функціональність у системі.

3.2. Розробка основних функції програми

Після того, як користувач успішно проходить авторизацію, він отримує доступ до програми і має змогу здійснювати моніторинг своєї комп'ютерної системи. Для відображення всіх данних було використано єлементи фреймворку WinForms, що називається PerfomanceCounter. Завдяки ньому є можливість напряму відображати системні показники з максимальною дочністю.



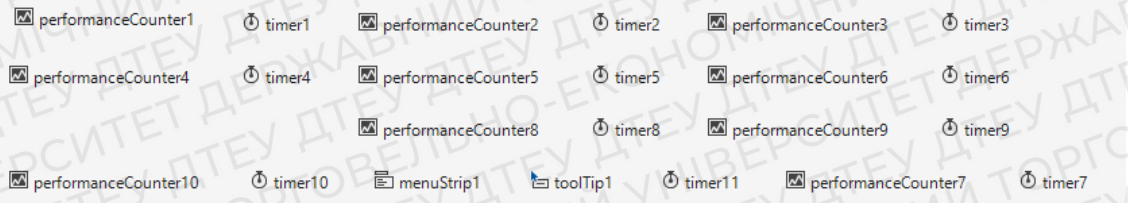


Рисунок 3.4. **Демонстрація використання єлементів WinForms**

Джерело: побудовано автором

PerformanceCounter в WinForms - це клас, який надає доступ до системних лічильників продуктивності (performance counters) в операційній системі Windows. Він дозволяє отримувати різноманітну інформацію про продуктивність комп'ютера, таку як використання процесора, обсяг вільної пам'яті, кількість операцій з диском тощо.

```
Ссылок: 1
private void timer1_Tick(object sender, EventArgs e)
{
    lbl1.Text = "TOTAL CPU USAGE: " + performanceCounter1.NextValue().ToString("0") + " %";
}
Ссылок: 1
private void timer2_Tick(object sender, EventArgs e)
{
    lbl2.Text = "CORE#1: " + performanceCounter2.NextValue().ToString("0") + " %";
}
Ссылок: 1
private void timer3_Tick(object sender, EventArgs e)
{
    lbl3.Text = "CORE#2: " + performanceCounter3.NextValue().ToString("0") + " %";
}
Ссылок: 1
private void timer4_Tick(object sender, EventArgs e)
{
    lbl4.Text = "CORE#3: " + performanceCounter4.NextValue().ToString("0") + " %";
}
Ссылок: 1
private void timer5_Tick(object sender, EventArgs e)
{
    lbl5.Text = "CORE#4: " + performanceCounter5.NextValue().ToString("0") + " %";
}
Ссылок: 1
private void timer6_Tick(object sender, EventArgs e)
{
    lbl6.Text = "Free RAM: " + performanceCounter6.NextValue() + " MB";
}
Ссылок: 1
```

Рисунок 3.5. **Демонстрація роботи клас PerfomanceCounter**
Джерело: побудовано автором

System.Diagnostics - це простір імен в .NET Framework (і .NET Core) який містить класи і інтерфейси для роботи з діагностикою, профілювання та моніторингом програм в середовищі .NET.У цьому просторі імен знаходяться класи для взаємодії з подіями, процесами, лічильниками продуктивності, реєстрацією подій, відлагодженням, моніторингом продуктивності та іншими діагностичними функціями.

```
Ссылок: 1
private void btn1_Click(object sender, EventArgs e)
{
    SaveFileDialog spremi = new SaveFileDialog();

    spremi.Title = "Save file";
    spremi.Filter = "Text Files (*.txt) |*.txt";


    if (spremi.ShowDialog() == DialogResult.OK){

        StreamWriter write = new StreamWriter(File.Create(spremi.FileName));

        write.Write(lbl1.Text );
        write.Write(lbl2.Text );
        write.Write(lbl3.Text );
        write.Write(lbl4.Text );
        write.Write(lbl5.Text );
        write.Write(lbl6.Text );
        write.Write(lbl8.Text );
        write.Write(lbl9.Text );
        write.Write(lbl10.Text );



        write.Dispose();
    }
}
```

Рисунок 3.6. **Демонстрація зберігання даних**
Джерело: побудовано автором

Користувач має змогу отримати та зберігти у текстовому файлі свої показники у реальному часі. Для цього було використано класс StreamWriter, що дозволяє створювати текстові документи та записувати туди дані.

Для відображення графіків було використано елементи фреймворку Chart. Charting в WinForms (Windows Forms) - це функціональність, яка дозволяє створювати та відображати графіки і діаграми в програмах, розроблених на платформі Windows Forms.

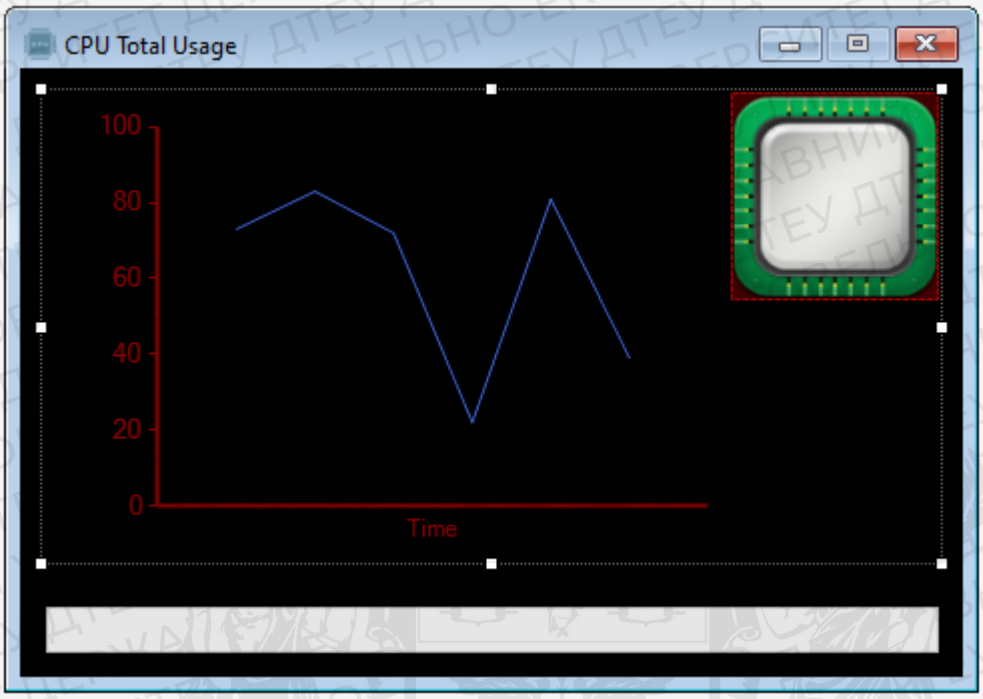


Рисунок 3.7. **Демонстрація графіків Charting**
Джерело: побудовано автором

Кожну секунду графік оновлюється і по точками малює криву, що відображає роботу системи у часі.

```
namespace CPU_Counter
{
    Ссылок: 5
    public partial class Form2 : Form
    {
        Ссылок: 1
        public Form2()
        {
            InitializeComponent();
        }

        Ссылок: 1
        private void timer1_Tick(object sender, EventArgs e)
        {
            float c = performanceCounter1.NextValue();
            int a = (int)c;

            progressBar1.Value = a;

            chart1.Series["Series1"].Points.AddY(a);
        }
```

Рисунок 3.9. **Демонстрація коду роботи графіків**
Джерело: побудовано автором

3.3. Розробка додаткових функціх програми

Для відображення інформації про стан та показники жорсткого диску, було використані єлементи PerfomanceCounter, що дають змогу швидко та досить точно відобразити стан систем.

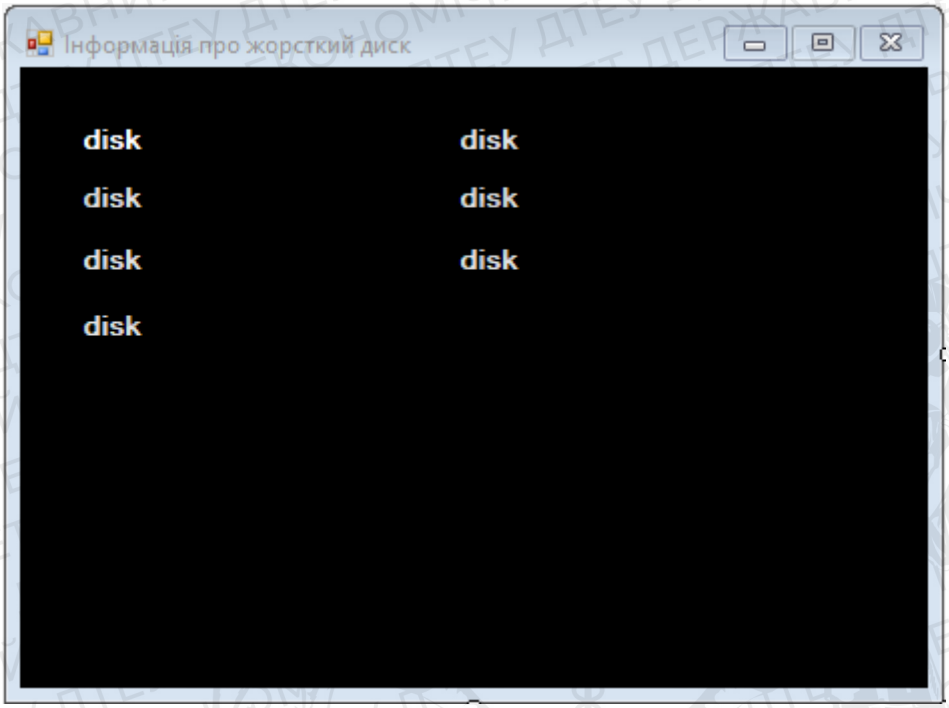


Рисунок 3.8. **Демонстрація вікна інформаціїї про жорсткий диск**
Джерело: побудовано автором

```
namespace CPU_Counter
{
    public partial class Form7 : Form
    {
        public Form7()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            label1.Text = "Disk time: " + performanceCounter1.NextValue().ToString("0.0") + " %";
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            label2.Text = "Disk read time: " + performanceCounter2.NextValue().ToString("0.0") + " %";
        }

        private void timer3_Tick(object sender, EventArgs e)
        {
            label3.Text = "Disk write time: " + performanceCounter3.NextValue().ToString("0.0") + " %";
        }
```

Рисунок 3.11. **Демонстрація коду відображення інформаціїї про жорсткий диск**
Джерело: побудовано автором

3.4. Висновки до розділу 3.

Після заглиблення в дослідження та розробку програми з використанням WinForms і C# була створена ефективна програма для перевірки систем ПК. Ця нова програма відрізняється надійністю та ефективністю, коли йдеться про моніторинг систем ПК. Інформацію про стан процесорів, оперативної пам’яті та жорстких дисків, а також про використання ресурсів можна збирати та аналізувати з високою ефективністю. Маючи зручний та інтуїтивно зрозумілий інтерфейс, програма моніторингу надійно виконує свої функції. За допомогою цієї програми є можливість легко отримати доступ до детальної інформації про систему. Відстежується поточний стан ПК, і завдяки цій програмі можна своєчасно реагувати на будь-які проблеми або незвичні ситуації, надаючи користувачам можливість вжити відповідних заходів.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

На основі проведеного дослідження ми дійшли до деяких заключних зауважень. Проведення експертизи поточних програм моніторингу системи виявило необхідність побудови нової системи моніторингу ПК. Причина, чому Windows стала основним варіантом для програми моніторингу системи ПК, полягає в тому, що вона є кращою операційною системою для персональних комп'ютерів, отже, вона повсюдна.

1. Після аналізу потреб користувачів визначено основні функціональні вимоги до програми моніторингу системи ПК.

2. Було проведено аналіз для визначення вимог і потреб користувачів, що дозволило визначити ключові функціональні вимоги до програми моніторингу системи ПК.

3. Визначення основних функціональних вимог до програми моніторингу системи ПК стало можливим завдяки ретельному аналізу вимог і потреб користувачів.

4. За допомогою технології WinForms та програмування на C# створено систему моніторингу роботи ПК під час реалізації проекту.

5. Для вищевказаної програми розроблено графічний інтерфейс.

6. Інформація користувачів зберігається за допомогою бази даних SQL.

Загальний висновок полягає в тому, що розроблена програма моніторингу систем ПК на базі C#, WinForms та SQL є ефективним інструментом для відстеження та аналізу стану систем ПК.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MVC Framework - Introduction [Електронний ресурс] // tutorialspoint.com – Режим доступу до ресурсу: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm.
2. C# documentation [Електронний ресурс] // learn.microsoft.com – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/.
3. Microsoft SQL [Електронний ресурс] // techtarget.com – Режим доступу до ресурсу: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server.
4. Марголін О. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] / Олександр Марголін // evergreens – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.
5. Проектування баз даних [Електронний ресурс] // rdb.dp.ua – Режим доступу до ресурсу: https://rdb.dp.ua/uk/chapter_04.
6. WinForms [Електронний ресурс] // mono-project.com ua – Режим доступу до ресурсу: https://www.mono-project.com/docs/gui/winforms/.

					ДТЕУ 121 07-02.БР		
Зм.	Аркуш	№ докум.	Підпис	Дата			
Зав. каф.		Криворучко О.В.		27.01.23	Програмний модуль моніторингу комп'ютерної системи	Стадія	Аркуш	Аркушів
Керівник		Палагута К.О		27.01.23		Р1	40	40
Гарант		Рзаєва С.Л.		27.01.23		Факультет інформаційних технологій 4 курс, 7 група		
Розробив		Васильєв Б.О		23.12.22	Список використаних джерел			

ДОДАТКИ

ДОДАТОК А

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace CPU_Counter
{
    public class DB
    {
        MySqlConnection _connection = new
MySqlConnection("server=localhost;port=3306;username=root;" +
            "password=root;database=pcmonitor");
        public void OpenConnection()
        {
            if( _connection.State == System.Data.ConnectionState.Closed)
            {
                _connection.Open();
            }
        }
        public void CloseConnection()
        {
            if (_connection.State == System.Data.ConnectionState.Open)
            {
                _connection.Close();
            }
        }
        public MySqlConnection getConnection()
        {
            return _connection;
        }
        public bool CheckUser(string username, string password)
        {
            DataTable dataTable = new DataTable();

            MySqlDataAdapter dataAdapter = new MySqlDataAdapter();

            MySqlCommand command = new MySqlCommand("SELECT * FROM `keys` WHERE
`username` = @uN AND `pass` = @pU", _connection);

            command.Parameters.Add("@uN", MySqlDbType.VarChar).Value = username;
            command.Parameters.Add("@pU", MySqlDbType.VarChar).Value = password;

            dataAdapter.SelectCommand = command;
            dataAdapter.Fill(dataTable);

            return dataTable.Rows.Count > 0;
        }

        public void AddUserLog(string username, DateTime loginTime)
        {
            MySqlCommand logCommand = new MySqlCommand("INSERT INTO `user_logs`
(`username`, `login_time`) " +
                "VALUES (@username, @login_time)", _connection);

            logCommand.Parameters.Add("@username", MySqlDbType.VarChar).Value =
username;
```

```
    logCommand.Parameters.Add("@login_time", MySqlDbType.DateTime).Value =
loginTime;

            try
            {
                _connection.Open();
                logCommand.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка: " + ex.Message);
            }
            finally
            {
                _connection.Close();
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CPU_Counter.Controller
{
    public class Controller
    {
        private DB db;

        public Controller()
        {
            db = new DB();
        }

        public void Authenticate(string password, string username)
        {
            if (db.CheckUser(username, password))
            {
                DateTime loginTime = DateTime.Now;

                db.AddUserLog(username, loginTime);

                MessageBox.Show("Вітаю!");
                Form1 form1 = new Form1();
                form1.Show();
                //this.Hide();
            }
            else
            {
                MessageBox.Show("Неправильний ключ!");
            }
        }
    }
}
```

```
using CPU_Counter.Controller;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CPU_Counter
{
    public partial class LoginForm : Form
    {
        private CPU_Counter.Controller.Controller controller;

        public LoginForm()
        {
            InitializeComponent();
            controller = new CPU_Counter.Controller.Controller();
        }

        private void LoginForm_Load(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string password = passField.Text;
            string username = nameField.Text;
            controller.Authenticate(password, username);
        }

        private void textBox1_TextChanged_1(object sender, EventArgs e)
        {

        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;



namespace CPU_Counter
{
    public partial class Form1 : Form
    {
        int a = 10;

        public Form1()
        {
             InitializeComponent();
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            lbl1.Text = "TOTAL CPU USAGE: " +
performanceCounter1.NextValue().ToString("0") + " %";
        }
        private void timer2_Tick(object sender, EventArgs e)
        {
            lbl2.Text = "CORE#1: " + performanceCounter2.NextValue().ToString("0") +
" %";
        }
        private void timer3_Tick(object sender, EventArgs e)
        {
            lbl3.Text = "CORE#2: " + performanceCounter3.NextValue().ToString("0") +
" %";
        }
        private void timer4_Tick(object sender, EventArgs e)
        {
            lbl4.Text = "CORE#3: " + performanceCounter4.NextValue().ToString("0") +
" %";
        }
        private void timer5_Tick(object sender, EventArgs e)
        {
            lbl5.Text = "CORE#4: " + performanceCounter5.NextValue().ToString("0") +
" %";
        }
        private void timer6_Tick(object sender, EventArgs e)
        {
            lbl6.Text = "Free RAM: " + performanceCounter6.NextValue() + " MB";
        }
        private void timer8_Tick(object sender, EventArgs e)
        {
            float s = performanceCounter8.NextValue();

            int ss = (int)s;

            TimeSpan t = TimeSpan.FromSeconds(ss);

            lbl8.Text = "System up time: " + t.ToString();
        }
        private void timer9_Tick(object sender, EventArgs e)
```

```
{
            lbl9.Text = "Active Processes: " + performanceCounter9.NextValue();
        }
        private void timer10_Tick(object sender, EventArgs e)
        {
            lbl10.Text = "Threads: " + performanceCounter10.NextValue();
        }
        private void btn1_Click(object sender, EventArgs e)
        {
            SaveFileDialog spremi = new SaveFileDialog();

            spremi.Title = "Save file";
            spremi.Filter = "Text Files (*.txt) |*.txt";


            if (spremi.ShowDialog() == DialogResult.OK){

                StreamWriter write = new StreamWriter(File.Create(spremi.FileName));

                write.Write(lbl1.Text );
                write.Write(lbl2.Text );
                write.Write(lbl3.Text );
                write.Write(lbl4.Text );
                write.Write(lbl5.Text );
                write.Write(lbl6.Text );
                write.Write(lbl8.Text );
                write.Write(lbl9.Text );
                write.Write(lbl10.Text );



                write.Dispose();
            }
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                timer1.Enabled = false;
                timer2.Enabled = false;
                timer3.Enabled = false;
                timer4.Enabled = false;
                timer5.Enabled = false;
                timer6.Enabled = false;
                timer8.Enabled = false;
                timer9.Enabled = false;
                timer10.Enabled = false;

            }
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton2.Checked)
            {
                timer1.Enabled = true;
                timer2.Enabled = true;
                timer3.Enabled = true;
                timer4.Enabled = true;
                timer5.Enabled = true;
                timer6.Enabled = true;
                timer8.Enabled = true;
                   timer9.Enabled = true;
```

```
                timer10.Enabled = true;

            }
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StreamWriter sr = new StreamWriter("upis.txt");

sr.Write(lbl1.Text+","+lbl2.Text+","+lbl3.Text+","+lbl4.Text+","+lbl5.Text+","+lbl6.
Text+","+lbl8.Text+","+lbl9.Text+","+lbl10.Text);
            sr.Close();
            MessageBox.Show("Збережено!");
        }

        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void creditsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Vasilyev Bohdan");
        }

        private void aboutTheProgramToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            MessageBox.Show("PC_Monitor_V0.1");
        }

        private void versionHistoryToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            MessageBox.Show("v1.0");
        }

        private void helpToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Натискання клавіші STOP зупиняє лічильники значень,
натискання клавіші RUN їх перезапускає.\n Ви можете зберегти дані двома способами,
файл/зберегти або натиснувши клавішу SAVE, де ви вибираєте назву даних, які ви
зберігаєте, і вибираєте місце призначення.");
        }

        string procitano;
        private void historyToolStripMenuItem_Click(object sender, EventArgs e)
        {

            try
            {
                StreamReader sr = new StreamReader("upis.txt");
                MessageBox.Show(procitano, "Збережені дані", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                procitano = sr.ReadToEnd();

                sr.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Неіснуючий файл!, «Помилка!");
}
        }
```

```
        private void progressBar1_Click(object sender, EventArgs e)
        {

        }


        //private void timer11_Tick(object sender, EventArgs e)
        //{

         //   a = a - 1;

        //   if (a == 0)
        //    {
        //        panel1.Visible = false;
       //     }

        //}

        private void totalUsageToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 f2 = new Form2();

            f2.Show();
        }

        private void cORE1ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form3 f3 = new Form3();

            f3.Show();
        }

        private void cORE2ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form4 f4 = new Form4();

            f4.Show();
        }

        private void cORE3ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form5 f5 = new Form5();

            f5.Show();
        }

        private void cORE4ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form6 f6 = new Form6();

            f6.Show();
        }

        private void hDDInfoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form7 f7 = new Form7();
f7.Show();
        }

        private void timer7_Tick(object sender, EventArgs e)
        {
           bool a1 = Environment.Is64BitOperatingSystem;

            if(a1 == true)
```

```
            {
                label1.Text = "Operating System: 64-bit";
            }
            else
            {
                label1.Text = "Operating System: 32-bit";
            }
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void lbl10_Click(object sender, EventArgs e)
        {

        }

        private void lbl9_Click(object sender, EventArgs e)
        {

        }

        private void lbl8_Click(object sender, EventArgs e)
        {

        }

        private void lbl6_Click(object sender, EventArgs e)
        {

        }

        private void lbl5_Click(object sender, EventArgs e)
        {

        }

        private void lbl4_Click(object sender, EventArgs e)
        {

        }
    }
```