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.58.14.000.ДП

МАННАПОВА
ДЕНИСА ХАМІТОВИЧА

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 програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК

Проєктний матеріал складається з пояснювальної записки на 74 сторінках та графічного (презентаційного) матеріалу на 15 аркушах (слайдах)

Дипломник _____ (Маннапов Д.Х.)

Керівник _____ (Шувалова І.О)

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 _____ (Краснокутська К.Г.)

Захист «24» червня 2025 р. Протокол ЕК № 4

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 і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Маннапову Денису Хамітовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК*

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

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *Використання середи розробки Microsoft Visual Studio; Використання мови програмування C#; Використання принципів ООП; Розробка інтерфейсу програми; Розробка принципу розрахунку потужності для блоків живлення ПК; Реалізація розрахунків; Тестування розробляємого ПЗ*

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

Аналіз існуючих систем розрахунку потужності БЖ; Вибір інструментів розробки; Структура розробляємого ПЗ; Принцип розрахунку потужності для блоків живлення ПК; Реалізація інтерфейсу ПЗ; Реалізація основних елементів ПЗ; Реалізація розрахунку потужності блоків живлення у ПЗ; Тестування розробленого ПЗ

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

Аналіз існуючих рішень; Інструменти розробки; Структура розробляємого ПЗ; Принцип розрахунку потужності блоків живлення; Процес реалізації інтерфейсу ПЗ; Процес реалізації основних елементів ПЗ; Тестування ПЗ; Скріншоти розробленого ПЗ

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

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

7. Дата видачі завдання 02.06.2025

Керівник *Шувалова І.О.* _____
(підпис)

Завдання прийняв до виконання *Манапов Д.Х.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*14.05.2025*	Виконав
2	*Аналіз існуючих аналогів*	*16.05.2025*	Виконав
3	*Вибір та налаштування засобів розробки*	*17.05.2025*	Виконав
4	*Проектування структури розробляємого ПЗ*	*20.05.2025*	Виконав
5	*Розробка принципу розрахунку потужності БЖ*	*22.05.2025*	Виконав
6	*Реалізація інтерфейсу ПЗ*	*28.05.2025*	Виконав
7	*Реалізація основних елементів ПЗ*	*01.06.2025*	Виконав
8	*Введення системи розрахунку потужності БЖ*	*03.06.2025*	Виконав
9	*Відлагодження розроблених елементів ПЗ*	*06.06.2025*	Виконав
10	*Тестування працездатності ПЗ*	*10.06.2025*	Виконав
11	*Виправлення виявлених помилок*	*11.06.2025*	Виконав
12	*Аналіз результатів, підготовка слайдів презентації*	*12.06.2025*	Виконав
13	*Економічні розрахунки та питання з охорони праці*	*13.06.2025*	Виконав
14	*Підготовка графічної частини проекту*	*14.06.2025*	Виконав
15	*Підготовка проекту до захисту та тестування ПЗ*	*16.06.2025*	Виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 58. 14 000. 00 ДП	Дипломний проєкт		
А4			КС 58. 14 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 58. 14 000. 00 ДП		
Розробив		Маннапов Д.Х.		16.06	Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК	Літ.	Аркуш 4 / Аркушів 74
Перевірив		Шувалова І.О		16.06.		Н Д П	
Н. контр.		Петрашова В.І.		16.06		ВСП "ОТФК ОНТУ" гр.4КС-58	
Затверд.		Кривченко Ю.В.		16.06.			

ЗМІСТ

ВСТУП

Промисловість комп’ютерної техніки є невід’ємною складовою світової ІТ-галузі. Щоб забезпечити коректну роботу персональних комп’ютерів важливо якісно обирати комплектуючі, зокрема блок живлення, який є ключовим елементом у системі, бо його основна задача це забезпечення живленням усіх комплектуючих персонального комп’ютера відповідно до їх енергоспоживання. Саме тому є висока потреба у розрахунку потужності блоку живлення персонального комп’ютера.

Водночас відбувається постійне удосконалення комп’ютерних комплектуючих, внаслідок чого зростає їх потреби у потужності джерел живлення. Розвиток технологій дозволяє автоматизувати розрахунки блоку живлення за допомогою програмного забезпечення.

На сьогоднішній час існує багато онлайн-додатків, які спрощують розрахункову частину, однак більшість із них є іншомовними, мають обмежений функціонал або не враховують нові компоненти. Отже зростає актуальність створення власної програми для підрахунку потужності блока живлення персонального комп’ютера, адаптованої до потреб користувача.

У зв’язку зі спрощенням розробки програмного забезпечення, створення подібного додатка може бути виконано навіть однією особою. Завдяки сучасним мовам програмування та зручними середовищами розробки можна реалізувати інтуїтивно зрозумілу програму для розрахунку потужності блока живлення та можливість оновлення даних про енергоспоживання комплектуючих. Для реалізації такої програми достатньо використання сучасної середи розробки та створення і використання електронної бази даних.

Планується реалізувати програмне забезпечення, яке дасть змогу обирати різні варіанти комплектуючих для персонального комп’ютеру, кількість носіїв інформації, систем охолодження, та отримувати розрахунок потужності блоків живлення для системи. Потужність, має бути виведена у вигляді фактичних та рекомендованих величин.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз існуючих систем розрахунку потужності БЖ

Перед тим, як почати аналіз аналогів, потрібно визначити осередки додатку для розрахунку потужності блока живлення персонального комп'ютера. Це інструмент який має використовуватися для полегшення процесу підрахунку потужностей комп'ютерних комплектуючих та використання цих даних для вибору відповідного блоку живлення.

Комп'ютерні комплектуючі – це складові модулі системного блока персонального комп'ютера, які мають власне призначення при роботі комп'ютера. В цьому випадку від комп'ютерних комплектуючих залежить загальне енергоспоживання усього системного блока, а як наслідок – блока живлення. Додаток для розрахунку потужності блока живлення повинен підраховувати потужності таких комплектуючих: центральний процесор (ЦП або CPU), відеокарта або графічний процесор (ГП або GPU), Накопичувачі (Жорсткий диск (HDD), Твердотілий Накопичувач (SSD), Гібридний жорсткий диск (SSHD)), та кількість охолоджуючих вентиляторів на корпусі системного блоку і центрального процесора, тобто для розрахунку блока живлення потрібно враховувати енергоспоживання процесора, відеокарти, накопичувачів та вентиляторів.

1.1.1 Аналіз Web-додатку «be quiet!»

Серед інтернет-застосунків перший якісний аналог у пошуковій системі був додаток «be quiet!». До його функціоналу входять такі функції як вибір комплектуючих і збереження конфігурації обраного комп'ютерного апаратного устаткування у текстовий файл та можливість завантажити конфігурацію з подібного текстового файлу.

Додаток має інтуїтивно зручний інтерфейс та функціонал який зображений на рисунку 1.1. У ньому можна обрати процесори досить старої лінійки процесорів таких як AMD Opteron або Intel Pentium 4, та найновіші процесори такі як AMD Ryzen 9 9950X3D та Intel Ultra 9285K. Також можна обрати відеокарту і вибір

буде аналогічно широкий, від старих NVIDIA GeForce GTX 480 або ATI Radeon HD 5870 до найновіших NVIDIA GeForce RTX 5090 або AMD Radeon RX 9070 XT. Присутній вибір кількості вентиляторів або водяного охолоджування, носіїв та оперативної пам’яті.

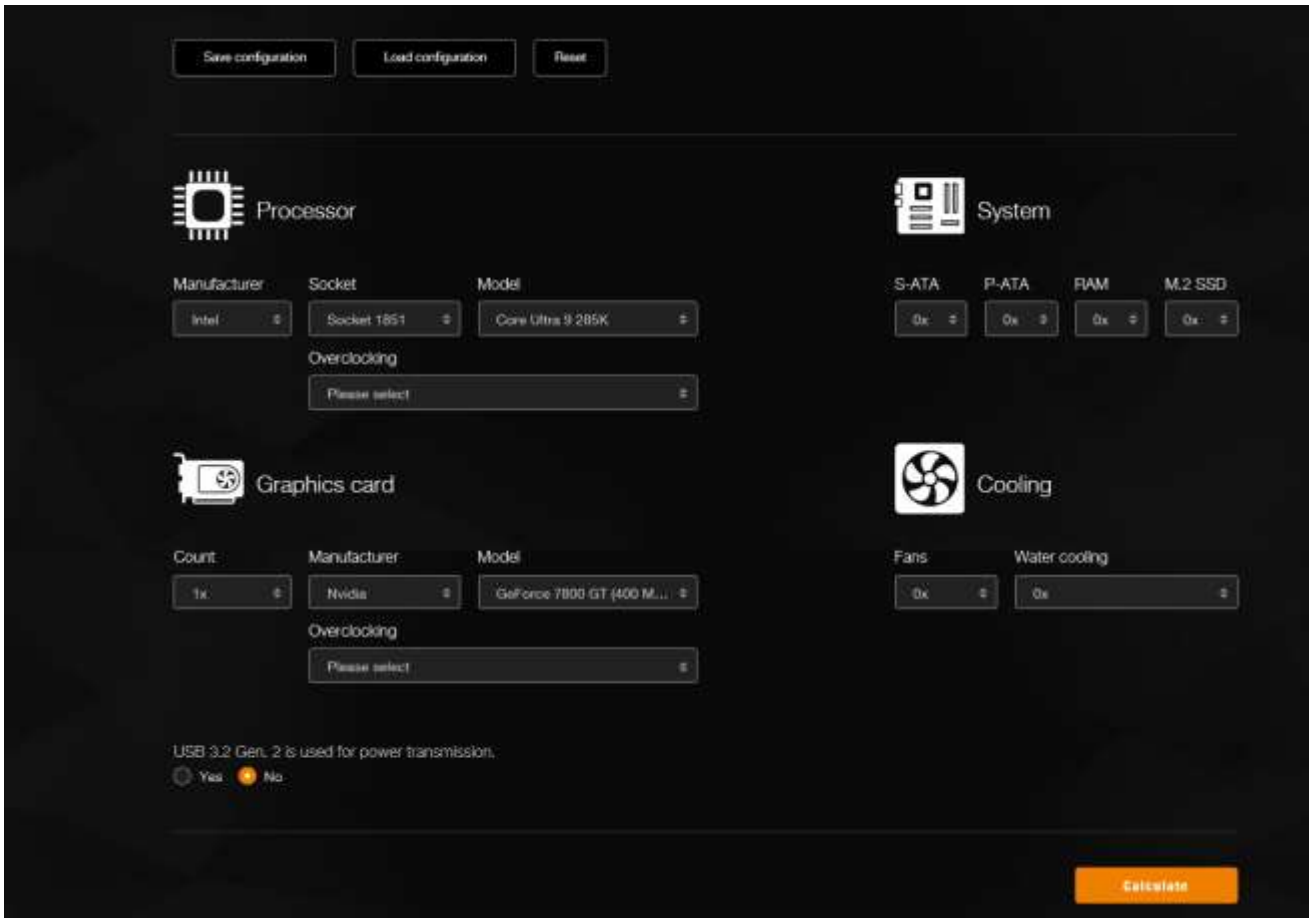


Рисунок 1.1. Скріншот додатку «be quiet!»

За підсумком, програма має зручний та якісний дизайн який охоплює усі елементи для взаємодії, дуже широкий вибір комплектуючих, які можна обрати за виробником, сокетом (для процесорів) і моделлю. Недоліком цього застосунку є процес обчислення потужності, він складається з підрахунку потужності компонентів та пошуку блока живлення від кількості цієї потужності незважаючи на рекомендацію виробників комплектуючих. Тобто сервіс може вибрати блок живлення який буде працювати на свою повну потужність, що може призвести до виходу блоку живлення з ладу достроково. З цієї ж причини будь-яка модернізація системного блока буде потребувати заміни і блока живлення, що є досить коштовною дією.

1.1.2 Аналіз Web-додатку від «Corsair»

Наступний додаток який є аналогом програми для розрахунку потужності блоку живлення персонального комп'ютера – це застосунок від «Corsair» зображений на рисунку 1.2.

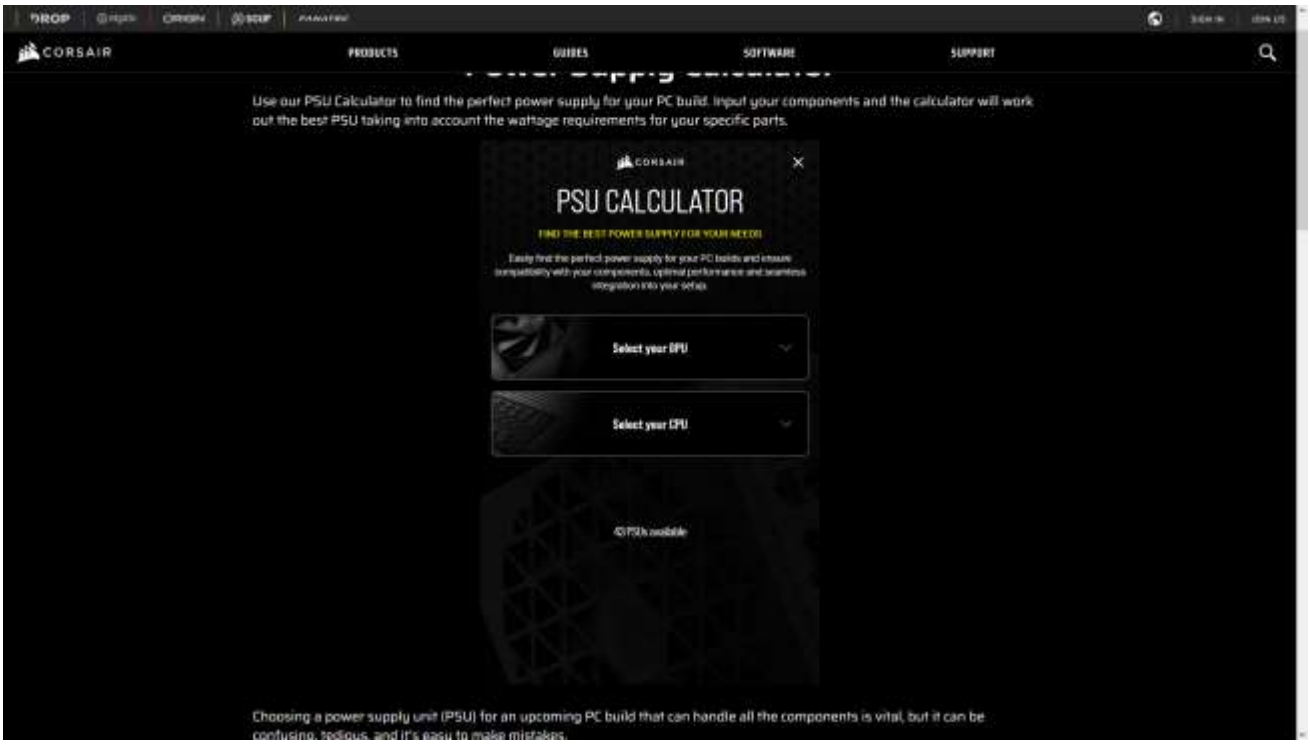


Рисунок 1.2. Скріншот сторінки додатку від «Corsair»

Обираючи процесор потрібно обрати форм-фактор системного блоку та можливість розгону системи, так як це дуже впливає на потужність блоку живлення. Нижче є повідомлення для користувача про те, що у розрахунки вже додані енергоспоживання до 4 модулей оперативної пам'яті, до 4 M.2 накопичувачів, до 2 S-ATA накопичувачів, водяного охолодження, та до 10 RGB вентиляторів.

Після вибору потрібних комплектуючих та додаткових характеристик веб додаток показує потужність комплектуючих при роботі та рекомендовану потужність блоку живлення разом із рекомендованими блоками живлення від виробника розрахункового додатку.

На відміну від «be quiet!», у якому є можливість обрати як дуже старі так і нові комп'ютерні комплектуючі, у додатку від «Corsair» можна обрати лише найновіші відеокарту та процесор що зменшує його попит серед власників

комп’ютерних комплектуючих вироблених більше ніж кілька років тому. Недоліком також є розбіжності у електроспоживанні компонентів та неможливість керувати кількістю додаткових компонентів, таких як носії та охолодження, що призводить до більших значень у витратах і вимогам до потужності блока живлення.

1.1.3 Аналіз Web-додатку від «Cooler Master»

Додаток від «Cooler Master» має поля для вибору комп’ютерних комплектуючих з можливістю обрати виробника та сокету (для процесора), форм-фактор материнської плати, кількість і тип оперативної пам’яті, об’єм та форм-фактор твердотілого накопичувача або жорсткого диску. Переглянути додаток можна на рисунку 1.3.

Після обрання комп’ютерних комплектуючих у додатку підраховується необхідна мінімальна потужність блоку живлення у реальному часі. Під потужністю з’являються блоки живлення від компанії виробника додатку, які задовольняють вимоги до мінімальної потужності до енергопостачання зазначеного комп’ютерної конфігурації з маленьким або достатнім запасом по потужності.

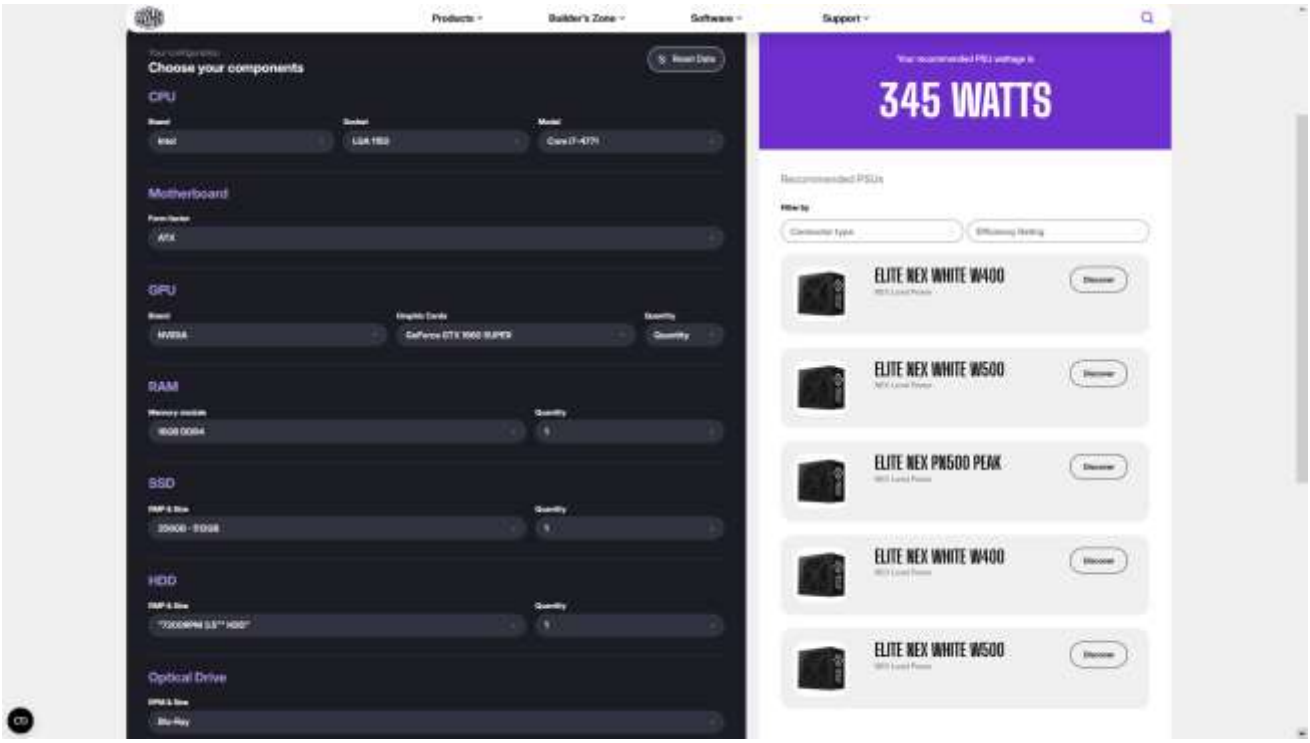


Рисунок 1.3. Скріншот сторінки додатку від «Cooler Master»

1.1.4 Результати аналізу

Після аналізу наявних web-додатків для розрахунку потужності блока живлення персонального комп'ютера можна виділити кілька спільних рис і недоліків. Усі три розглянуті сервіси (від be quiet!, Corsair та Cooler Master) орієнтовані на автоматизований вибір комплектуючих та розрахунок споживаної потужності з подальшою рекомендацією блока живлення. Інтерфейси в більшості випадків зручні, підтримується вибір популярних комплектуючих.

Основні відмінності полягають у гнучкості налаштувань, актуальності бази комплектуючих та підході до обрахунку потужності. Додаток «be quiet!» дозволяє обирати як застарілі, так і нові компоненти, але не враховує запас потужності згідно з рекомендаціями виробників. Сервіс від «Corsair» обмежений новими моделями й має фіксовані значення для деяких типів комплектуючих, що може призводити до неточностей. «Cooler Master» пропонує доволі точний розрахунок у реальному часі, але також як і «Corsair» фокусується лише на власних блоках живлення.

З огляду на це, у рамках дипломного проєкту варто врахувати: необхідність актуальної бази комплектуючих, можливість ручного регулювання кількості споживачів енергії, а також логіку обчислення, яка включає запас потужності відповідно до реальних умов експлуатації та можливих майбутніх оновлень конфігурації системного блоку.

1.2 Вибір інструментів розробки

1.2.1 Вибір середи розробки

Кожен процес розробки програмного забезпечення починається з вибору середовища, у якому буде створюватись програма. Це ключовий етап, оскільки він визначає не тільки мову програмування, яка буде використовуватись, але і доступні інструменти, підхід до створення інтерфейсу, спосіб організації коду, методи тестування, налагодження та подальшої підтримки проєкту. Від правильності вибору середовища залежать зручність роботи, швидкість розробки, стабільність програми та простота її масштабування. Для реалізації утиліти,

призначеної для розрахунку необхідної потужності блока живлення персонального комп’ютера, було обрано інтегроване середовище розробки Visual Studio. Це середовище добре відоме серед розробників C# і надає широкий набір можливостей для створення десктопних застосунків. Воно підтримує роботу з Windows Forms, що дозволяє швидко проєктувати графічний інтерфейс, інтегрувати логіку обробки даних і реалізувати обчислювальні алгоритми.

Visual Studio надає повноцінну підтримку платформ .NET Framework і .NET Core, що забезпечує гнучкість при виборі архітектури застосунку. Усі основні компоненти – редактор коду з підсвіткою синтаксису, система підказок, автоматичне виявлення помилок, відлагоджувач, візуальний редактор форм, інструменти побудови і компіляції – зібрані в єдиному вікні. Це значно спрощує процес розробки та усуває потребу у зовнішніх засобах. Крім того, середовище включає підтримку систем керування версіями (наприклад, Git), автоматичні оновлення проєктів, інтеграцію з системами контролю помилок, можливість додавання сторонніх розширень, а також широкий вибір плагінів, які покращують або розширюють функціональність середовища. Завдяки цьому зменшується кількість рутинних дій і скорочується загальний час реалізації програмного забезпечення.

Серед можливих альтернатив розглядалися інші середовища: JetBrains Rider, SharpDevelop, MonoDevelop та Visual Studio Code. Rider – потужне середовище з розширеним функціоналом, але воно орієнтоване переважно на командну розробку, потребує платної ліцензії та має більший обсяг системних вимог. SharpDevelop і MonoDevelop – проєкти з відкритим кодом, але вони вже не підтримуються і не відповідають сучасним вимогам, особливо в частині інтеграції з новими версіями .NET. Visual Studio Code – зручний легкий редактор, але для повноцінної роботи з C# потребує встановлення додаткових розширень, ручного налаштування процесу збирання, компіляції та налагодження, що ускладнює розробку невеликого настільного застосунку з графічним інтерфейсом.

У результаті було обрано Visual Studio Community Edition – безкоштовну версію середовища, яка містить повний набір необхідних функцій для реалізації

проєкту. Вона підтримує мову C#, забезпечує глибоку інтеграцію з .NET, дозволяє швидко створювати графічний інтерфейс за допомогою Windows Forms і налагоджувати логіку програми. Такий вибір дозволяє без додаткових витрат зосередитися на реалізації основної функціональності, а не на технічній підготовці середовища чи усуненні проблем сумісності.

1.2.2 Вибір системи бази даних

Для збереження і обробки даних, які використовуються у програмі, була обрана реляційна система управління базами даних – Microsoft SQL Server. Це поширена і стабільна СУБД, яка добре інтегрується з середовищем Visual Studio і з мовою C#. Основна перевага – можливість напряму працювати з базою даних за допомогою технології ADO.NET, без потреби у зовнішніх драйверах або сторонніх бібліотеках. Visual Studio містить вбудовані засоби для створення з'єднання з SQL Server, побудови таблиць, написання SQL-запитів, перегляду та зміни вмісту бази, а також налагодження взаємодії бази з формами, полями введення, кнопками та іншими елементами інтерфейсу.

Microsoft SQL Server підтримує всі необхідні функції, які потрібні для роботи невеликої програми: збереження параметрів користувача, результатів обчислень, журналів, налаштувань. Є можливість створювати зв'язки між таблицями, задавати типи полів, обмеження на значення, використовувати зовнішні ключі, індекси для пришвидшення запитів, зберігані процедури, тригери та інші інструменти для організації цілісності і керування даними. Система також підтримує транзакції, що дозволяє контролювати послідовність виконання запитів при оновленні або видаленні даних, виключаючи ризик пошкодження або втрати інформації при помилках.

Для реалізації взаємодії програми з базою даних передбачено можливість підключення до серверного SQL-сховища через стандартні засоби доступу, зокрема через ADO.NET. У загальному випадку це дозволяє працювати з віддаленим сервером бази даних, що зручно для масштабованих рішень або корпоративних систем. Однак у цьому проєкті база даних використовується не локально, з підключенням до зовнішнього сервера. Обрано SQL Server Express –

безкоштовне видання, яке встановлюється сервер компанії розробника. Воно забезпечує повну функціональність для роботи з базою, але потребує окремого серверного середовища. З'єднання з базою відбувається так само, як і при роботі з сервером, але дані обробляються локально. Такий підхід спрощує розгортання, дозволяє тестувати функції прямо в середовищі Visual Studio, а також підвищує актуальність даних, що важливо для якісного ПЗ.

1.3 Структура розробляємого ПЗ

1.3.1 Загальна структура програми

Для подальшої розробки програми підрахунку потужності блока живлення потрібно визначити логіку роботи, яка включає як внутрішню обробку даних, так і взаємодію з користувачем. Структура програми має складатися з окремих етапів, кожен з яких виконує чітку функцію – від ініціалізації інтерфейсу до обробки введених даних і формування результату.

При запуску програма виконує ініціалізацію всіх елементів інтерфейсу. Завантажуються основні компоненти вікна, включаючи текстові позначення (Label), викидні списки (ComboBox), керувальні кнопки (Button), та повідомлення (MessageBox), які будуть використовуватись для повідомлень і попереджень.

Далі користувач взаємодіє з інтерфейсом, обираючи відповідні комп'ютерні комплектуючі: процесор, відеокарту, материнську плату, оперативну пам'ять, накопичувачі, вентилятори охолодження. Вибір кожного елемента відбувається через відповідні викидні списки або прапорці вибору. Для кожного з обраних компонентів в інтерфейсі передбачено поле введення або вибору, що дозволяє точно вказати тип і кількість.

Після завершення вибору користувач натискає кнопку розрахунку, що запускає основний обчислювальний алгоритм. Програма отримує вхідні дані, перевіряє їх коректність (наявність обов'язкових компонентів, допустимі значення тощо) і виконує підрахунок сумарної потужності, необхідної для стабільної роботи всієї конфігурації. Після обробки програма виводить результат у зручному для користувача форматі. Це може бути окреме текстове поле або

повідомлення у вигляді MessageBox, де вказується рекомендована потужність блока живлення, з урахуванням резерву для пікового навантаження обраної конфігурації. Алгоритм програми зображений на рисунку 1.4.

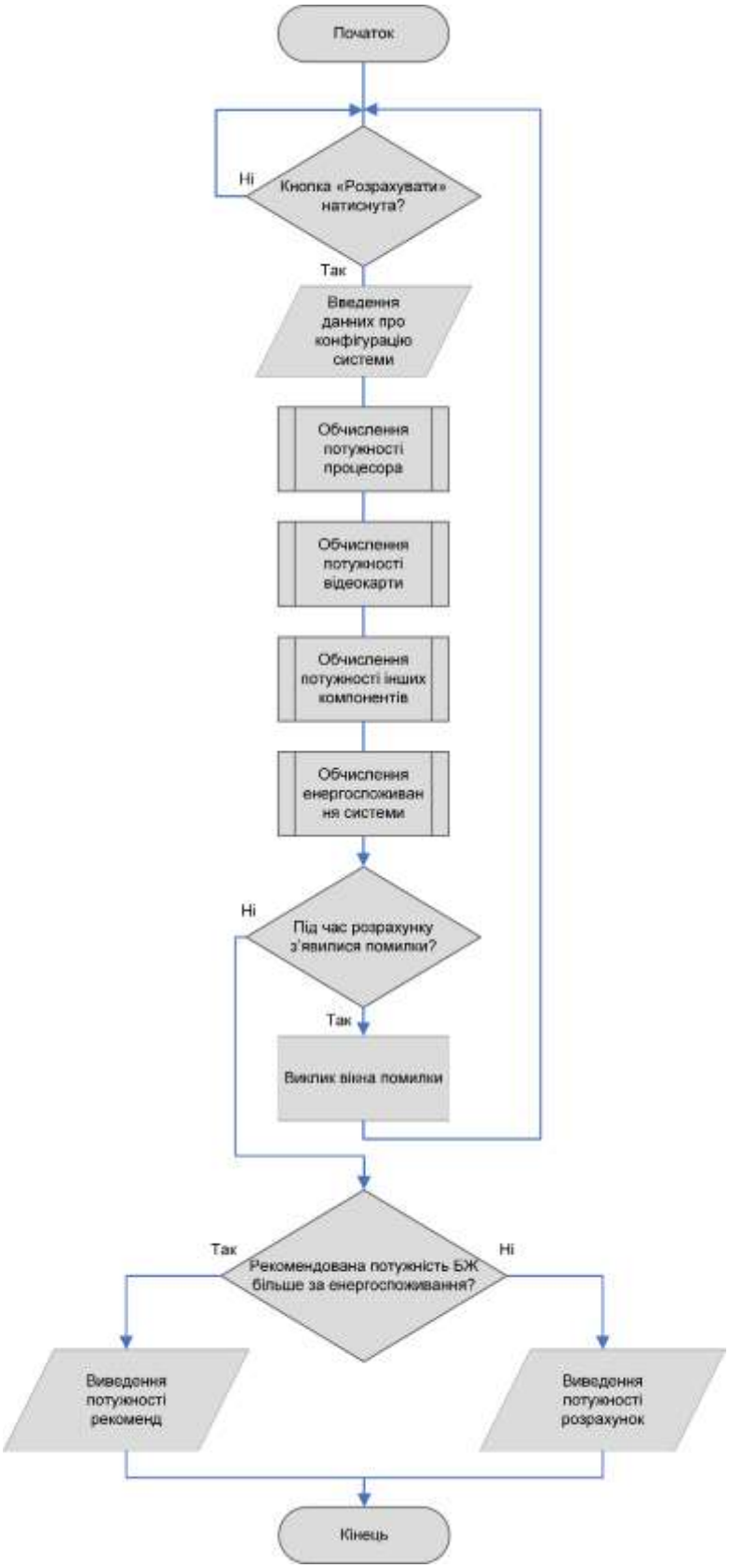


Рисунок 1.4. БСА роботи у програмі для розрахунку потужності БЖ

Як описано у алгоритмі, спочатку до програми надходять данні про вибрану користувачем конфігурацію системи у вигляді назви комп'ютерних комплектуючих, далі програма послідовно обчислює дані після чого виводить енергоспоживання до користувача.

Заносити дані до програми зовнішній користувач буде через такі елементи користувацького інтерфейсу як викидний список (ComboBox), у якому можна буде обрати комп'ютерний компонент за назвою (для процесора та відеокарти) або кількість додаткових комплектуючих (твердотілі накопичувачі, жорсткі диски, кількість модулів оперативної пам'яті, та кількість додаткових вентиляторів охолодження), з даними яких буде працювати обчислювальна частина програми. Модуль вибору комп'ютерних компонентів можна побачити на рисунку 1.5.

Рисунок 1.5. Модуль вибору комп'ютерних комплектуючих

Після того як зовнішній користувач обрав необхідні дані, ці значення потрібно передати на обробку згідно з закладеним алгоритмом. Обробка виконується у внутрішній логіці програми, після чого результат слід вивести назад до інтерфейсу. Для цього достатньо використати звичайний елемент Label, який оновлюється відповідно до отриманих розрахунків. Значення в Label змінюється динамічно, залежно від результату, який формує розрахункова частина програми.

1.3.2 Середовище зберігання даних

Для коректного функціонування програми потрібно забезпечити її середовищем, яке буде зберігати в собі дані про комп'ютерні комплектуючі, такі як процесор та відеокарта, а саме назви їх моделей які в подальшому будуть автоматично з'являтись у викидному списку та відповідно їх максимального енергоспоживання.

На початку розробки програми було прийняте рішення використовувати списки (enum) як середовище зберігання назв та їх відповідного значення у цілочисельному форматі, що підходить для виміру потужності. Приклад списку можна побачити на рисунку 1.6.

```
public enum CPU
{
    Ryzen3_1200 = 30, Ryzen5_1600 = 80, Ryzen5_2600 = 80, Ryzen5_3600 = 80,
    Ryzen5_5600 = 80, Ryzen5_5600X = 90, Ryzen7_1700 = 80, Ryzen7_2700X = 90,
    Ryzen7_3700X = 90, Ryzen7_5800X = 140, Ryzen7_5800X3D = 90, Ryzen9_3900X = 140,
    Ryzen9_5900X = 150, Ryzen9_5950X = 150, Ryzen5_7600 = 90, Ryzen5_7600X = 120,
    Ryzen7_7700 = 90, Ryzen7_7700X = 140, Ryzen7_7800X3D = 90, Ryzen9_7900 = 90,
    Ryzen9_7900X = 200, Ryzen9_7950X = 260, Core_i3_6100 = 60, Core_i3_8100 = 70,
    Core_i3_10100 = 70, Core_i3_13100 = 70, Core_i5_6600K = 100, Core_i5_7600K = 100,
    Core_i5_8600K = 100, Core_i5_9600K = 100, Core_i5_10600K = 130, Core_i5_11600K = 130,
    Core_i5_12400 = 90, Core_i5_12600K = 150, Core_i5_13400 = 100, Core_i5_13600K = 180,
    Core_i7_6700K = 100, Core_i7_7700K = 100, Core_i7_8700K = 100, Core_i7_9700K = 100,
    Core_i7_10700K = 130, Core_i7_11700K = 130, Core_i7_12700K = 190, Core_i7_13700K = 250,
    Core_i9_9900K = 130, Core_i9_10900K = 130, Core_i9_11900K = 130, Core_i9_12900K = 240,
    Core_i9_13900K = 250, Core_i9_12900KS = 280
}
```

Рисунок 1.6. Приклад списку процесорів

Однак списки мають багато недоліків. Наприклад до списку можна занести тільки два типи даних: строкові (назва елементу списку), що буде зберігати назву компонента і цілочисельні (величина яка присвоюється до елементу списку), що буде зберігати потужність компонента. Назва елементу списку має бути лише одним словом, тому назва має бути без пробілів (наприклад замість «GTX 1080 TI» має бути «GTX1080TI» або «GTX_1080_TI») і для подальшого використання списку з цими назвами потрібно буде прописувати кожен елемент списку до елементів випадаючого списку вручну, через те виникає ще більше проблем і незручностей. Потенційне оновлення програми як мінімум буде включати редагування кожного списку з комп'ютерними компонентами та редагування

переліку елементів у кожному викидному списку, на додачу пробіли у назвах компонентів будуть замінені на «_» («GTX_1080_TI» а не «GTX 1080 TI»), для того щоб витягувати дані з списку.

Тому була створена база даних у середовищі Microsoft SQL Server. У ній реалізовано одну таблицю для зберігання назв компонентів комп'ютера, таких як процесори та відеокарти, а також їхніх характеристик. Основна мета – централізоване зберігання даних і повна відмова від списків у коді. Кожен запис у таблиці містить повну назву компонента, наприклад «Intel Core i7-11700K» або «NVIDIA GeForce RTX 3080», а також числове значення максимальної потужності у ватах. Це дозволяє не стискати назви, не замінювати пробіли символами підкреслення і не обмежуватись одним словом, як це було у випадку зі списками.

У Visual Studio налаштовано з'єднання з сервером бази даних через стандартний SQL-провайдер. Дані з таблиці підтягуються під час запуску програми автоматично. Запит до бази виконується через SQL-запит SELECT, який отримує всі доступні компоненти. Після отримання результату дані передаються до елементів інтерфейсу, зокрема до випадаючих списків. Таким чином користувач бачить повні назви компонентів без скорочень, у нормальному форматі, як у звичайних специфікаціях. Вибір компонента автоматично підтягує його енергоспоживання, яке потім використовується в розрахунках.

MySQL на початковому етапі також розглядався як варіант, але при спробі підключення до Visual Studio виникли проблеми з драйверами та несумісністю з внутрішніми інструментами розробки. Наприклад, стандартний DataSet не працює з MySQL без сторонніх рішень, а інтеграція з ORM-інструментами типу Entity Framework нестабільна. Крім того, у процесі налагодження були проблеми з обробкою запитів, через що вирішено повністю відмовитись від MySQL.

Microsoft SQL Server, навпаки, одразу працює з Visual Studio без додаткових налаштувань. Таблиці можна створювати прямо у середовищі розробки, також доступне редагування, перегляд вмісту, побудова запитів, генерація моделей даних. Це значно спростило процес роботи з даними і зробило систему гнучкою. Будь-які оновлення в таблиці автоматично впливають на роботу інтерфейсу, при

цьому змінювати код не потрібно. Це дозволяє підтримувати актуальність компонентів без зайвих зусиль.

Microsoft SQL Server Management Studio – це програма, яка використовується для повного керування базами даних Microsoft SQL Server. У ній можна створювати нові бази, писати запити, додавати таблиці, змінювати структуру, переглядати дані і виконувати обслуговування. Інтерфейс простий: зліва відображається дерево об’єктів сервера, справа – вікно запитів або редактор таблиць. Усе можна зробити як через SQL-запити, так і через графічні форми без необхідності писати код. Приклад роботи у середовищі для роботи з Microsoft SQL Server, Microsoft SQL Server Management Studio зображено на рисунку 1.7.

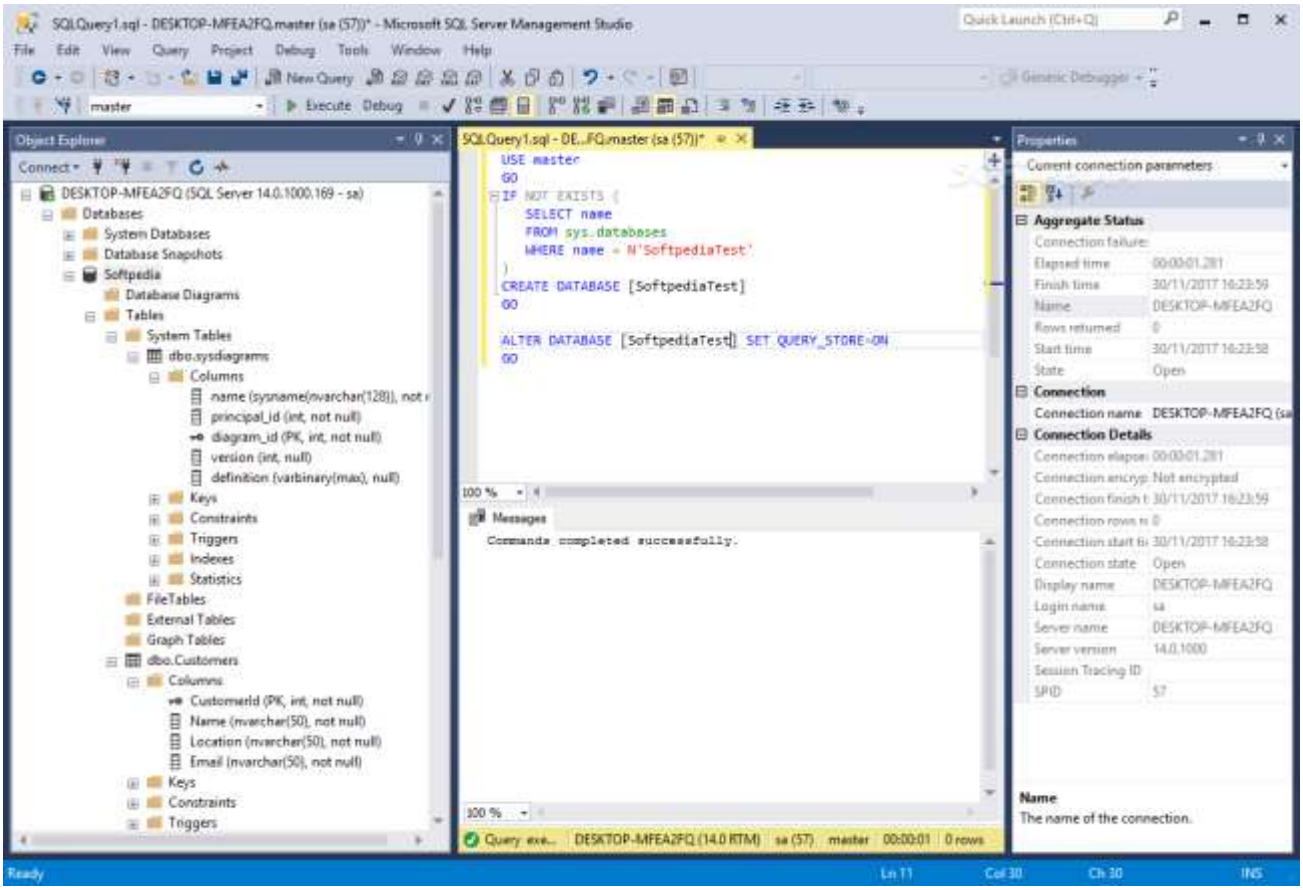


Рисунок 1.7. Приклад роботи у Microsoft SQL Server Management Studio

Перед початком реалізації бази даних, необхідно було розробити схему зв’язків між таблицями, полями бази даних та програмою для розрахунку потужності блоків живлення. Оскільки база даних використовується як внутрішній файл для проекту, свої дані вона буде передавати безпосередньо до елементів інтерфейсу майбутньої програми. На рисунку 1.8 зображена схема

зв’язків між складовими бази даних та елементами програми.

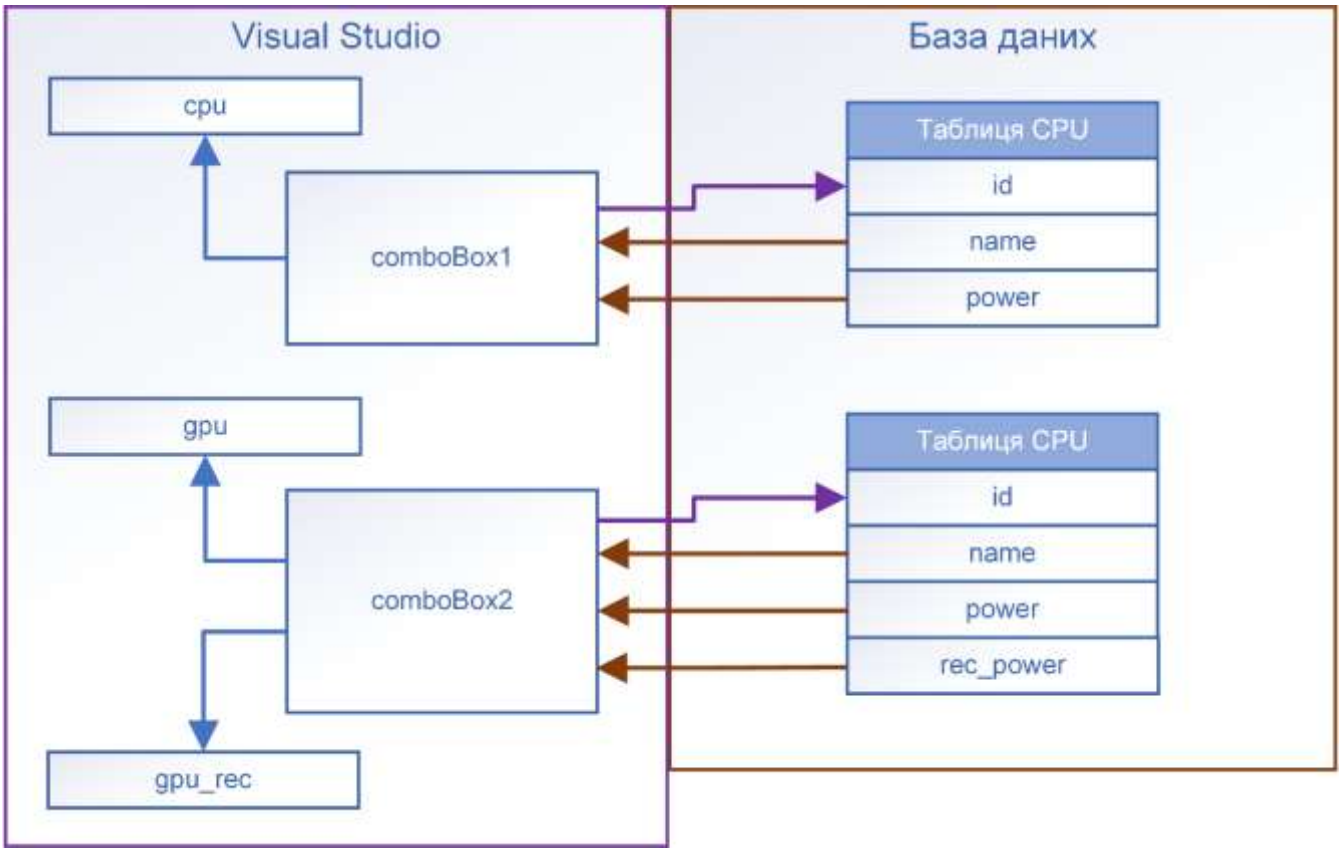


Рисунок 1.8. Схема зв’язків між складовими бази даних та елементами програми

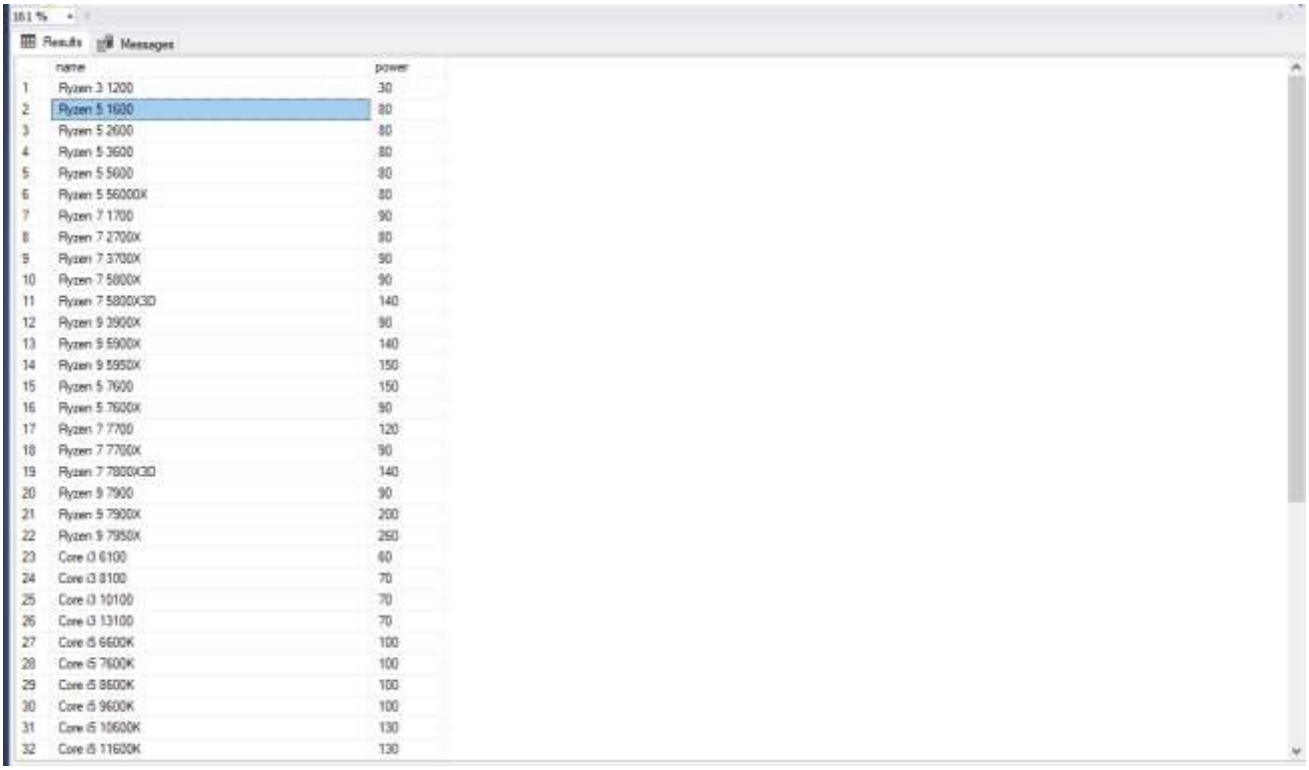
Щоб створити нову базу даних, спочатку потрібно запустити SSMS і підключитись до локального SQL Server. Після запуску програма запитує сервер і тип автентифікації. Якщо сервер встановлено на тому ж комп’ютері, достатньо обрати localhost і залишити тип авторизації Windows за замовчуванням. Після успішного підключення відкривається доступ до всіх баз і об'єктів сервера.

Далі в лівій частині вікна потрібно знайти розділ "Databases", натиснути на нього правою кнопкою миші і вибрати створення нової бази. Відкривається просте вікно, де достатньо вказати назву бази, наприклад Power. Після підтвердження база з’являється в дереві об’єктів.

Після цього можна створити таблицю для зберігання даних про комп’ютерні компоненти. Для цього потрібно перейти до нової бази, знайти розділ "Tables", відкрити редактор нової таблиці. У вікні створення таблиці задаються колонки: наприклад, одна для назви (процесора чи відеокарти) і одна для потужності у ватах. Всі поля вводяться вручну через інтерфейс. Ідентифікатор задається як int

з автозаповненням, назви та тип – як текстові значення (nvarchar), потужність – як ціле число (int). Після завершення таблиця зберігається з довільною назвою, наприклад CPU.

Потім до таблиці можна додавати дані. Це можна зробити двома способами: або через графічний редактор, де таблиця відкривається у вигляді звичайної табличної форми, або за допомогою SQL-запитів типу INSERT INTO. У будь-якому випадку дані зберігаються в базі і доступні для подальшого використання програмою. Вигляд таблиць бази даних зображений на рисунку 1.9. та 1.10.



	name	power
1	Ryzen 3 1200	30
2	Ryzen 5 1600	80
3	Ryzen 5 2600	80
4	Ryzen 5 3600	80
5	Ryzen 5 5600	80
6	Ryzen 5 5600GX	80
7	Ryzen 7 1700	90
8	Ryzen 7 2700X	80
9	Ryzen 7 3700X	90
10	Ryzen 7 5800X	90
11	Ryzen 7 5800X3D	140
12	Ryzen 9 3900X	90
13	Ryzen 9 5900X	140
14	Ryzen 9 5950X	150
15	Ryzen 5 7600	150
16	Ryzen 5 7600X	90
17	Ryzen 7 7700	120
18	Ryzen 7 7700X	90
19	Ryzen 7 7800X3D	140
20	Ryzen 9 7900	90
21	Ryzen 9 7900X	200
22	Ryzen 9 7950X	250
23	Core i3 6100	60
24	Core i3 8100	70
25	Core i3 10100	70
26	Core i3 13100	70
27	Core i5 6600K	100
28	Core i5 7600K	100
29	Core i5 8600K	100
30	Core i5 9600K	100
31	Core i5 10600K	130
32	Core i5 11600K	130

Рисунок 1.9. Таблиця CPU

Подібна таблиця створюється й для комплектуючих відеокарт але з додатковим стовпцем який буде зберігати дані про рекомендовану потужність блоку живлення для комп’ютера. Окрім додаткового стовпця також є стовпець із показниками потужності для відеокарти. Це дасть змогу виконувати потрібні розрахунки з виводу даних щодо номінальної потужності блоку живлення для обраної конфігурації персонального комп’ютера, та рекомендованої потужності блоку живлення.

	name	power	rec_power
1	GeForce GTX 950	90	350
2	GeForce GTX 960	120	400
3	GeForce GTX 970	145	500
4	GeForce GTX 980	165	500
5	GeForce GTX 980 Ti	250	600
6	GeForce GTX 1050	75	300
7	GeForce GTX 1050 Ti	75	300
8	GeForce GTX 1060	120	400
9	GeForce GTX 1070	150	500
10	GeForce GTX 1070 Ti	180	500
11	GeForce GTX 1080	180	500
12	GeForce GTX 1080 Ti	250	600
13	GeForce GTX 1650	75	300
14	GeForce GTX 1660	120	450
15	GeForce GTX 1660 Ti	120	450
16	GeForce GTX 1660 Super	125	450
17	GeForce RTX 2060	160	500
18	GeForce RTX 2060 Super	175	550
19	GeForce RTX 2070	175	550
20	GeForce RTX 2070 Super	215	650
21	GeForce RTX 2080	225	650
22	GeForce RTX 2080 Super	250	650
23	GeForce RTX 2080 Ti	300	650
24	GeForce RTX 3050	130	550
25	GeForce RTX 3060	170	550
26	GeForce RTX 3060 Ti	220	600
27	GeForce RTX 3070	240	650
28	GeForce RTX 3070 Ti	310	850
29	GeForce RTX 3080	340	750
30	GeForce RTX 3080 Ti	350	750
31	GeForce RTX 3090	350	750
32	GeForce RTX 3090 Ti	450	850

Рисунок 1.9. Таблиця GPU

Наприкінці база даних готова до підключення з боку застосунку. У Visual Studio задається підключення до сервера, після чого можна виконувати запити до створеної таблиці, витягати назви компонентів, типи і енергоспоживання, та підставляти їх у елементи інтерфейсу. Завдяки такому підходу майже вся логіка зберігання винесена за межі коду, а зміна даних не потребує редагування самої програми.

1.4 Принцип розрахунку потужності для блоків живлення ПК

1.4.1 Мета розрахунку

Основна задача – обчислити необхідну потужність блока живлення, виходячи з конфігурації комп’ютера. Підхід базується не на типових шаблонних значеннях, а на використанні конкретних параметрів компонентів із бази даних, де для кожної моделі вказано її типове енергоспоживання. Що дозволяє точно адаптувати розрахунок під реальну систему користувача.

Програма також враховує рекомендації виробників відеокарт щодо живлення системи. У базі даних для кожної моделі GPU зберігається рекомендована потужність блока живлення. Це значення порівнюється з

обрахованим і використовується, якщо воно більше або дорівнює потужності з урахуванням запасу.

1.4.2 Структура розрахунку

У структурі розрахунку потужності блока живлення основну роль відіграє послідовна обробка вхідних даних з елементів інтерфейсу користувача, підрахунок сумарного споживання енергії та визначення кінцевої рекомендованої потужності. Вся логіка реалізована у вигляді обробника події з використанням конструкції try-catch, що дозволяє уникати аварійного завершення програми у випадку незаповнених полів.

На початку відбувається прив'язка властивості ValueMember для comboBox2 до поля «power», яке містить значення споживаної потужності графічного адаптера. Після цього зчитуються значення:

- потужності процесора (comboBox1.SelectedValue);
- потужності відеокарти (comboBox2.SelectedValue) ;
- кількості HDD (comboBox3.Text), де кожен диск використовує 10 Ват.;
- кількості SSD (comboBox5.Text), де кожен носій використовує 15 Ват.;
- кількості RAM (comboBox4.Text), де кожен диск використовує 5 Ват.;
- кількості вентиляторів (comboBox6.Text), де кожен вентилятор додає 15 Ват.

Усі ці значення конвертуються у цілі числа та підсумовуються. Таким чином, обчислюється базова потужність всієї системи без урахування запасу. Змінна power є агрегованим значенням і становить основу для подальших розрахунків.

Далі comboBox2.ValueMember змінюється на «rec_power», що дозволяє отримати рекомендовану виробником потужність блока живлення для обраної відеокарти. Це значення фіксується в змінній gpu_rec.

Проміжний результат – сумарна потужність – відображається у першій мітці (label1) з відповідним текстом. Потім виконується множення на коефіцієнт 1.25 для врахування запасу, ділиться на 1000, округлюється до одного знаку після коми, а потім знову приводиться до значення у ватах. Це потрібно для того щоб

отримати число рекомендованого блоку живлення у сотнях ват, бо блоки живлення зазвичай конструюються з потужністю кратною сотні.

Ключовий етап – порівняння отриманої з урахуванням запасу потужності з рекомендованою виробником відеокарти. Якщо значення з коефіцієнтом менше або дорівнює рекомендованому – система виводить саме gpu_rec як рекомендовану потужність. Якщо більше – виводиться розраховане значення з коефіцієнтом. Це дозволяє гарантувати, що не буде перевищено вимог виробника, водночас не відбудеться зниження стабільності роботи через недостатнє енергоживлення. Структуру розрахунку програми можна побачити на рисунку 1.10.

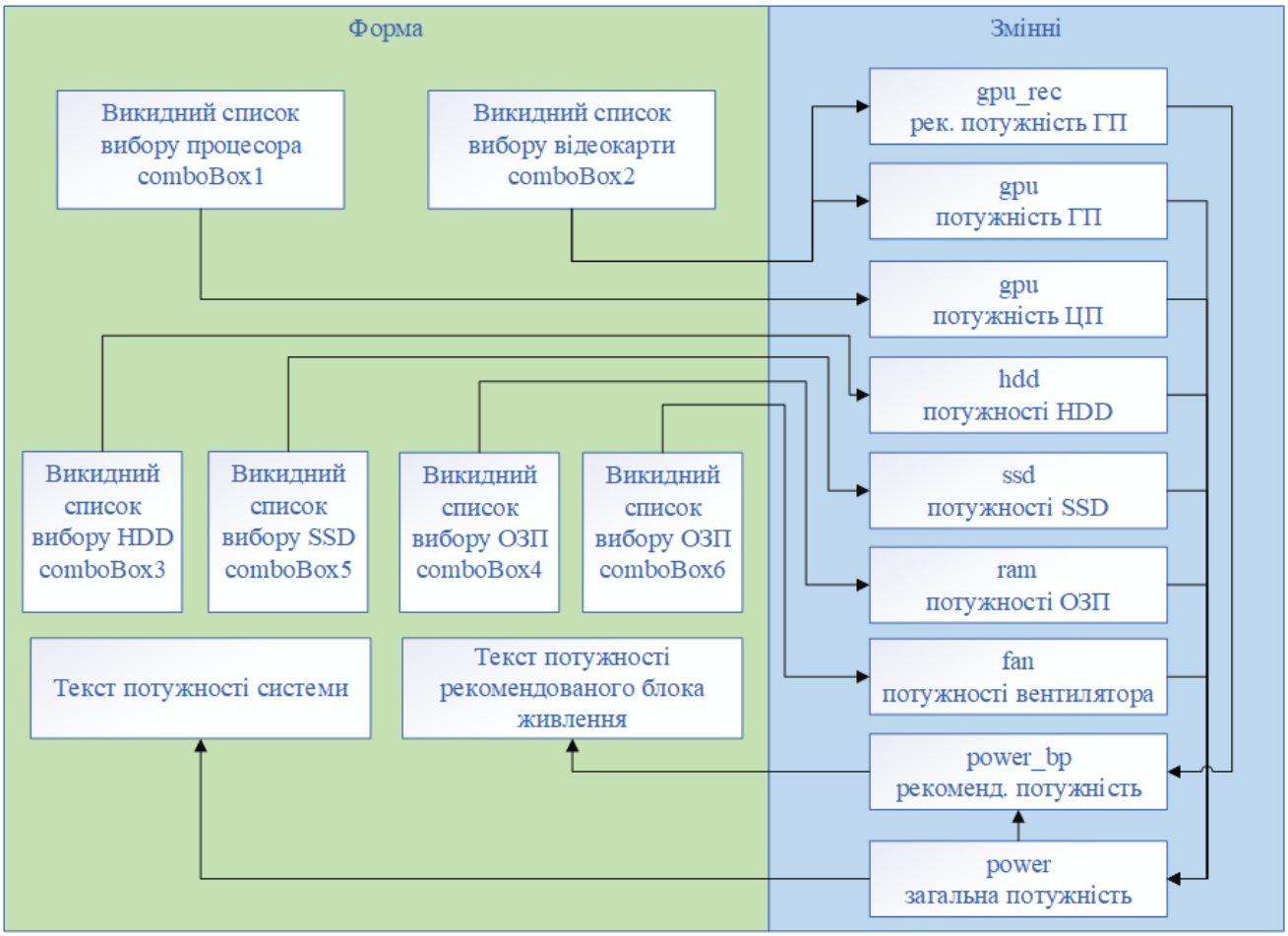


Рисунок 1.10. Структура розрахунку програми

У випадку, якщо будь-яке з полів не заповнене або містить некоректні значення, спрацьовує блок catch, який видає повідомлення через MessageBox, нагадуючи користувачу про необхідність обрати всі компоненти. Це обумовлено тим, що у програмі немає прямого вводу даних користувача до програми, воно

здійснюється за допомогою викидного списку у якому як би не намагався не вийде вписати неіснуюче значення наприклад або не вписати нічого. Єдиний випадок коли програма може ініціювати блок catch, коли користувач не вибрав процесор та відеокарту, тоді програма не зможе отримати дані з бази даних без назви компоненту, як наслідок не зможе здійснити розрахунки. Тому тільки у випадку коли користувач не вибере процесор або відеокарту(або процесор та відеокарту) буде виводитися вікно нагадуючи користувачу що потрібно обрати комплектуючі. У блоках з жорстким диском, твердотілим накопичувачем, оперативною пам’яттю, і вентиляторами охолодження за замовчуванням виставленні положення «0».

1.5 Процес реалізації інтерфейсу ПЗ

1.5.1 Реалізація структури програмного забезпечення

Щоб реалізувати структуру, яка була описана у попередньому розділі, потрібно зосередитися на тому, з яких елементів складається програма. Вона складається з викидних списків (ComboBox), написів (Label), та кнопки (Button). Програму буде реалізовано як WindowsForm, у середовищі розробки Visual Studio.

Для початку, потрібно створити проєкт у Visual Studio. Йому було дано назву «PidrahunokPotujnosti», що підходить до теми та ідеї програми. Після чого, на формі було розміщено за допомогою внутрішнього інструментарію Visual Studio, а саме «Toolbox», потрібні елементи – comboBox (для процесора, відеокарти, жорсткого диску, твердотілого накопичувача, оперативної пам’яті, вентилятора охолодження), button (для виконання кнопки «Розрахувати»), і label (для написів щодо «Оберіть процесор» та «Оберіть відеокарту», також для додаткових комплектуючих «HDD, SSD, ОЗП, Вентилятори», і для написів про потужність комп’ютера та блока живлення, для більшої зручності користувача). Після розташування всіх елементів на формі, їх налаштування розмірів та позицій, результатом є не налаштований вигляд програми, який можна побачити на рисунку 1.11.

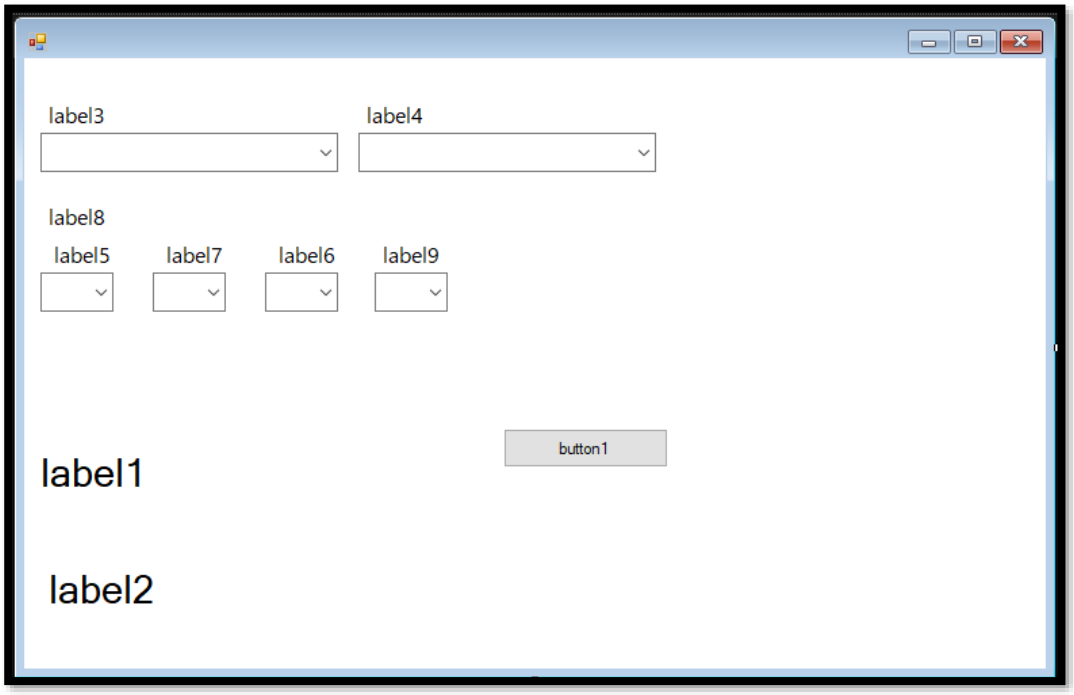


Рисунок 1.11. Форма без налаштувань

Після чого йде тривале налаштування компонентів. Важливо зробити написи так, щоб інтерфейс був інтуїтивно зрозумілий для користувача. Почнемо з верху до низу, від основних компонентів:

Label3 – у розділі Appearance потрібно вибрати властивість «Text» і вписати туди щось, що буде спонукати користувача до вибору моделі процесора, наприклад «Оберіть процесор», також це можна зробити і не в розділі Properties графічно, а при запуску форми, наприклад label3.Text = “Оберіть процесор”;.

Label4 – так само у розділі Appearance треба вибрати властивість «Text» та написати текст, який буде підштовхувати користувача до обирання моделі відеокарти, не потрібно далеко відходити від стилю яким був написаний текст про процесор, нехай це буде «Оберіть відеокарту», так само як і з процесором можна не змінювати назву за допомогою зручного інтерфейсу середовища Visual Studio, а за допомогою коду, під час ініціалізації форми змінити властивість напису рядком коду: label4.Text = “Оберіть відеокарту”; .

Далі потрібно налаштувати написи додаткових компонентів, але перед цим йде напис який показує користувачу що далі йде вибір додаткових компонентів, щоб користувач знав що він буде обирати у викидних списках нижче, адже comboBox-и містять в собі лише цифру, їх ширина замала для напису типу

«Оберіть оперативну пам’ять» або «Оберіть ОЗП», це занадто широко. Тому напис має містити загальну назву усіх компонентів нижче:

Label8 – як було зроблено з процесором та відеокартою, переходимо до вкладки Properties, до категорії Appearance обираємо параметр Text, змінюємо його на «Оберіть додаткові компоненти», і як з процесором або відеокартою можна змінити без графічного інтерфейсу, рядком label8.Text = “Оберіть додаткові компоненти”; .

Написи label5, label7, label6, label9 мають показувати які саме додаткові комплектуючі обираються:

Label5 перший із списку додаткових компонентів, він має відповідати за жорсткий диск, тому змінюємо параметр, як і у попередніх написах, по шляху Properties – Appearance – Text на «HDD». Цей параметр можна змінити рядком у методі void Form1_Load: label5.Text = “HDD”;.

Label7 відповідає за твердотілі накопичувачі, тому налаштування властивості буде аналогічне попередньому напису. Змінюється параметр Text рядком або через графічний інтерфейс середовища розробки на «SSD».

Label6 показує що нижче обирається кількість планок оперативної пам’яті, отже і параметр Text потрібно змінити на «ОЗП» через інтерфейс або рядком – label6.Text = “ОЗП”;.

Label9 це останній напис у категорії додаткові компоненти. Він показує на вибір кількості додаткових вентиляторів охолодження, відповідно і параметр Text має бути змінений на «Кулер», що не є таким коротким текстом як попередні назви комплектуючих, але все ж відповідає вимогам до довжини. Так як і усі минулі написи, може бути змінений графічним інтерфейсом або рядком label9.Text = “Кулер”;

Label1 виводить максимальну потужність системи яку вибрав користувач, тому напис має доносити цю інформацію до нього. Перед цим напис Напис змінюється аналогічно попереднім написам, але так як напис не є статичним і взаємодіє з програмою, під час розрахунків зручніше буде прописувати його значення рядком у коді окрім статичної частини «Внесіть дані», яка має бути

встановлена через графічний інтерфейс середовища розробки або рядком у методі ініціалізації форми: label1.Text = “Внесіть дані”; У коді після натискання кнопки button1, та після подальших розрахунків має бути рядок, який змінює значення label1 з «Внесіть дані» на «Потужність ПК: X Wh». Рядок виглядає так: label1.Text = “Потужність ПК: ”+ power.ToString() + “ Wh”. Цілочисельного типу змінна power, як вже було описано на рисунку 1.13, має підраховувати суму всіх потужностей комплектуючих, отже має в собі дані про потужність усієї системи, щоб змінна коректно виводилася необхідно її конвертувати до строкового типу даних, це робить метод «ToString». Фрагмент “ Wh” позначає одиниці виміру потужності – Ват на годину.

Label2 виконує поставлену у цьому проєкті функцію: показує якої потужності блок живлення рекомендовано встановити користувачу, щоб його системна конфігурація мала достатньо живлення, працювала коректно, і не потребувала заміни блока живлення у випадку її модифікації. Напис про рекомендовану потужність буде з’являтися після натискання кнопки button1, тоді за замовчуванням властивість Text елементу label2 має містити порожнє значення « ». На рисунку 1.4. зображено розгалуження перед виводом інформації. Коли змінна gpu_rec (відповідає за рекомендовану виробником відеокарти потужність блоку живлення у системі з цією відеокартою) більше за power_bp (розрахована потужність оптимального блок живлення із «запасом» потужності для цієї конфігурації), виводиться число зі змінної gpu_rec, інакше – зі змінної power_bp. Рядки коду які будуть виконуватися відповідно умові такі:

label2.Text = “Рекомендована потужність блоку живлення “+gpu_rec.ToString()+” Wh”; (для випадку коли gpu_rec => power_bp).

label2.Text = “Рекомендована потужність блоку живлення “+power_bp.ToString()+” Wh”; (для випадку коли gpu_rec < power_bp).

Наступним кроком перед налаштуванням викидних списків є налаштування кнопки button1.

Button1 відповідає за запуск розрахунку загальної потужності системи та визначення рекомендованої потужності блока живлення. Користувач натискає

кнопку після того, як обрав усі комплектуючі зі списків. Напис на кнопці має чітко вказувати, що вона запускає процес розрахунку. Щоб змінити напис, потрібно у властивостях кнопки знайти категорію Appearance і змінити параметр Text, наприклад, на «Розрахувати». Те саме можна зробити у коді під час ініціалізації форми рядком: button1.Text = "Розрахувати"; Після натискання виконується метод, пов’язаний з подією Click. У цьому методі зчитуються вибрані значення з comboBox-ів, обчислюється сумарна потужність, результат виводиться у label1. Також визначається рекомендована потужність блока живлення і виводиться у label2.

Після налаштування тексту написів та тексту кнопки отримаємо вигляд форми без дизайну графічного інтерфейсу. Форма зображена на рисунку 1.12.

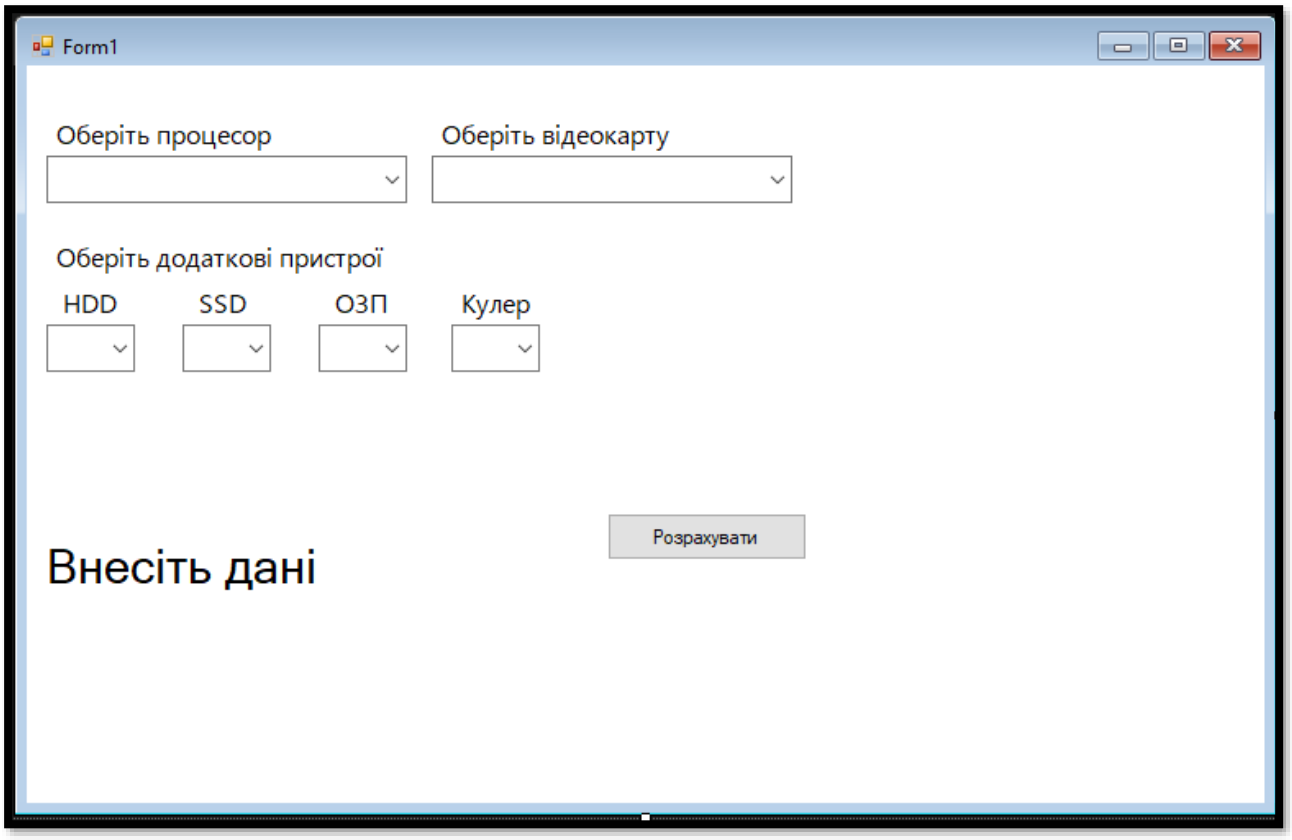


Рисунок 1.12. Скріншот програми без графічного дизайну

ComboBox1 відповідає за вибір відеокарти. Дані для нього беруться не вручну, а з таблиці GPU, яка зберігається в базі даних MS SQL Server. Для цього спочатку підключається сама база. У Visual Studio потрібно відкрити панель Server Explorer (натиснути Ctrl+Alt+S). Додається нове з’єднання через Add

Connection. Тип джерела – Microsoft SQL Server. Далі вказується назва сервера, наприклад localhost, і база Power, в якій є таблиця CPU. У цьому випадку база знаходиться локально, тому зручно використовувати Windows Authentication. Якщо сервер віддалений або використовується обліковий запис SQL Server – вводиться логін і пароль. Після успішного тесту з'єднання з'єднання додається до проєкту. Це зображено на рисунку 1.13.

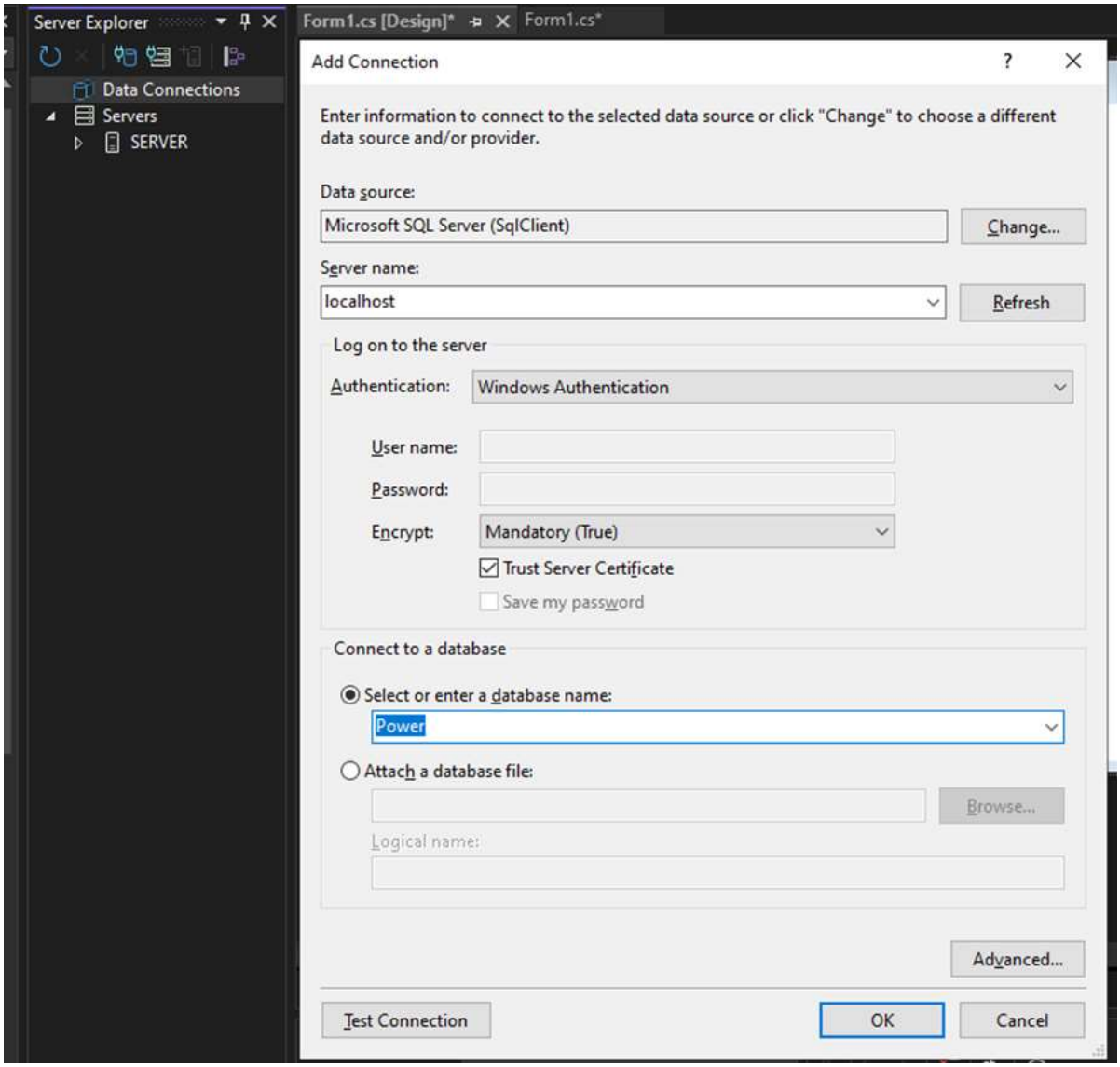


Рисунок 1.13. Вікно під'єднання до бази даних Microsoft SQL Server

Visual Studio автоматично створює набір класів для таблиць бази. Для отримання даних із таблиці CPU зручно використати DataSet. У меню Visual Studio відкривається Data Sources, обирається Add New Data Source, далі – Database, після чого знову вказується наявне з'єднання з базою. Серед таблиць потрібно вибрати CPU. В результаті у проєкті з'являється клас, який містить

об’єкт TableAdapter і доступ до даних. Цей адаптер дозволяє витягнути всі записи з таблиці CPU в DataTable. У методі Form1_Load створюється об’єкт CPUTableAdapter. Викликається метод GetData(), який повертає усі записи з таблиці. Вибирається лише поле name, тобто назва процесора. Ці назви послідовно додаються в comboBox1, щоб користувач мав змогу вибрати один із варіантів. Після того як користувач зробить вибір компонентів, при натисканні кнопки оброблюється подія. З адаптера знову отримується таблиця, далі знаходиться рядок, де поле name відповідає вибраному значенню. З цього рядка витягується поле power, яке відповідає споживанню енергії процесором. Отримане значення зберігається у змінній cpu. Ця змінна буде використана для розрахунку загальної потужності системи.

ComboBox2 використовується для вибору процесора. Джерелом даних для нього є таблиця GPU, яка зберігається в базі даних MS SQL Server. Щоб дані з'явилися в списку одразу після запуску форми, потрібно налаштувати з’єднання з базою та підтягнути потрібні значення. У Visual Studio створюється підключення до бази через Server Explorer. Якщо з’єднання вже є – достатньо переконатися, що воно активне. Якщо ні – додається нове, де задається назва SQL-сервера, тип аунтефікації, вибирається база, і перевіряється зв’язок через кнопку тесту. Після успішного з’єднання база з’являється у вигляді дерева, де можна бачити таблиці. Далі через інструмент Data Sources додається таблиця GPU до джерел даних проєкту. Обирається варіант "Database", потім вибирається вже існуюче з’єднання з базою, вказується таблиця GPU, після чого генеруються класи доступу до неї. Це дозволяє працювати з даними через TableAdapter, без прямого SQL-коду. У методі Form1_Load створюється екземпляр CPUTableAdapter. Через метод GetData() отримується таблиця з усіма рядками. Із кожного рядка береться назва відеокарти – поле name, і додається як один із варіантів у comboBox2. Дані вставляються в цикл. Після цього comboBox містить список усіх відеокарт з бази, користувач вибирає одну з них. При натисканні button1 у коді знову створюється адаптер, витягується таблиця GPU, далі по назві відеокарти, яку вибрав користувач, знаходиться потрібний рядок. З нього витягується значення стовпців

power і rec_power, які присвоюються змінним gpu і gpu_rec відповідно. Це потрібно для розрахунку загальної потужності конфігурації і для визначення, яку потужність для блоку живлення виведе користувачу програма.

ComboBox3 використовується для вибору кількості жорстких дисків (HDD), які будуть враховані при конфігурації системи. На відміну від comboBox1 та comboBox2, які відповідають за вибір процесора і відеокарти і працюють з базою даних, comboBox3 не потребує підключення до зовнішніх джерел. Кількість HDD не впливає на розрахунки продуктивності, тому досить просто зчитати вибране значення зі списку.

Значення для comboBox3 задаються вручну під час проєктування форми. Щоб це зробити, у вікні конструктора потрібно натиснути стрілку ComboBox Tasks, вибрати Unbound Mode і натиснути Edit Items. З'явиться текстове поле, в яке вводяться можливі варіанти вибору — кожен з нового рядка. Це показано на рисунку 1.14.

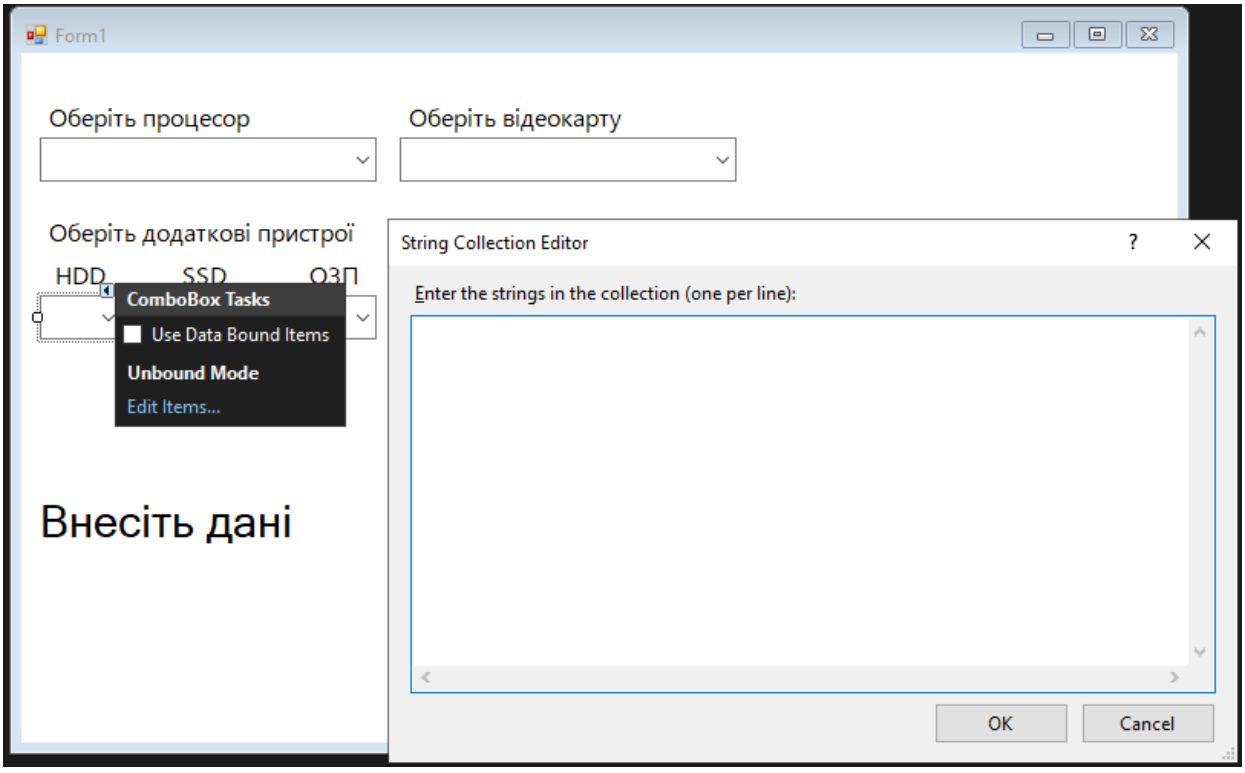


Рисунок 1.14. Редагування вмісту викидного списку

У цьому випадку список містить числа від 0 до 6, що відповідає типовій кількості дисків у звичайних ПК. Альтернативно список можна сформувати в коді

за допомогою таких команд:

comboBox3.Items.Add("0");

comboBox3.Items.Add("1");

comboBox3.Items.Add("2");

comboBox3.Items.Add("3");

comboBox3.Items.Add("4");

comboBox3.Items.Add("5");

comboBox3.Items.Add("6");

Обидва способи однаково придатні, залежно від того, як зручніше реалізувати інтерфейс.

Зчитування кількості жорстких дисків виконується за допомогою властивості Text, значення присвоюються до змінної через зчитування тексту обраного користувачем варіанту викидного списку. Приклад рядку зчитування кількості жорстких дисків:

*int hdd = Convert.ToInt32(comboBox3.Text) * 10;*

Цей рядок створює змінну цілочисельного типу (int) з іменем «hdd». змінній «hdd» присвоюється перетворений до цілочисельного типу текст обраного користувачем варіанту, та множить його на 10, щоб отримати значення потужності для подальших розрахунків. Конвертація до цілочисельного типу даних потрібна через те, що властивість «.Text» передає дані у строковому форматі (string), з яким не можна проводити розрахунки, тому для цього використовуємо метод «Convert.ToInt32», який переводить строковий формат «string» до цілочисельного формату «int» для додаткових дій з ним, у цьому випадку – множення і подальша сума.

ComboBox5 відповідає за кількість твердотілих накопичувачів (SSD) у конфагурації. Як і з викидним списком вибору кількості жорстких дисків, тут не потрібне підключення до бази данних, а усі можливі варіанти викидного списку можна вписати вручну у полі додавання елементів, що зображено на рисунку 1.14, або за допомогою рядків коду:

comboBox5.Items.Add("0");

comboBox5.Items.Add("1");

comboBox5.Items.Add("2");

comboBox5.Items.Add("3");

comboBox5.Items.Add("4");

comboBox5.Items.Add("5");

comboBox5.Items.Add("6");

Так як і викидний список вибору кількості жорстких дисків, цей викидний список буде однаково записувати дані вибрані користувачем. Приклад рядку для зчитування кількості твердотілих накопичувачів:

*int ssd = Convert.ToInt32(comboBox5.Text) * 15;*

Аналогічно comboBox3, зчитування виконується через властивість Text, множиться на 15, та записується до щойно створеної цілочисельної змінної «ssd», яка далі буде використовуватися у обчисленнях.

ComboBox4 використовується для введення користувачем кількості планок оперативної пам’яті. Так як і в двох попередніх викидних списках, comboBox4 не потрібно підключати базу даних, він має лише 7 цифр, які можна додати графічно або при ініціалізації. Він буде зберігати варіанти: 0, 1, 2, 3, 4, 5, 6.

Механізм зчитування аналогічний, дані беруться з обраного користувачем варіанту через властивість Text та конвертуються до цілочисельного формату, після чого множаться на 5, та наприкінці записуються у змінну цілочисельного формату з іменем «ram». Приклад рядку зчитування кількості оперативної пам’яті:

*int ram = Convert.ToInt32(comboBox3.Text) * 5;*

ComboBox6 потрібен для вибору користувачем кількості додаткових вентиляторів охолодження. Як і всіх викидних списках вибору додаткових компонентів, comboBox6 не потрібна база даних, він працює з списком цифр – від 0 до 6, які можна занести до варіантів методом додавання через графічний інтерфейс: у вкладці «ComboBox Tasks» обрати напис «Edit items…» та вписати туди списком 0, 1, 2, 3, 4, 5, 6. Або скористатися «.Items.Add» та додати рядком у коді. Зчитування відбувається аналогічно попереднім викидним спискам за допомогою властивості Text яка буде зчитуватися, конвертуватися до

цілочисельного формату, потім помножуватися на 15, та записуватися у змінну «fan». Приклад рядку зчитування кількості вентиляторів:

*int fan = Convert.ToInt32(comboBox6.Text) * 15;*

1.5.2 Реалізація дизайну ПЗ

Після розміщення елементів і налаштування їхніх властивостей потрібно перейти до зовнішнього вигляду форми. Стандартний вигляд WindowsForm виглядає застаріло. Щоб програма виглядала сучасніше і зручніше для користувача, потрібно внести базові зміни до візуального оформлення.

Почати варто з фону форми. За замовчуванням колір сірий. У Properties – BackColor змінено на світлий відтінок (наприклад, WhiteSmoke), щоб елементи були краще видимі. Далі змінюються кольори самих елементів. Наприклад, кнопці button1 призначено колір фону LightGray, текст зроблено чорним. Це дає контраст і зручність при натисканні.

Шрифти на формі за замовчуванням дрібні. Щоб зробити інтерфейс читабельнішим, у кожного label і comboBox в розділі Font змінюється розмір шрифту. Оптимально – 11 або 12 пт. Шрифт стандартний – Microsoft Sans Serif або Segoe UI, для того щоб не зламалася верстка на різних версіях Windows.

Для логічного групування елементів можна використовувати елемент GroupBox. Вони не мають функціональності, але дозволяють візуально відділити частини інтерфейсу. Наприклад, окремо групуються comboBox-и для CPU і GPU, окремо – для додаткових компонентів (HDD, SSD, ОЗП, кулери), ще окремо – блок виводу результату. Це спрощує орієнтацію користувача на формі. Було вирішено замінити label8 («Оберіть додаткові комплектуючі») на groupBox з написом «Оберіть кількість додаткових пристроїв», який є частиною вибору додаткових комплектуючих. Також було додано groupBox із назвою «Результати:», у якому знаходяться label1 та label2, для розуміння того, яка саме область відповідає за результати.

Заголовок форми змінюється у властивості Text – на щось зрозуміле, наприклад «Розрахунок потужності блока живлення ПК». Значок форми (іконка у верхньому лівому куті) можна теж замінити через властивість Icon. Для цього

підключається файл .ico через Resources. Іконка програми зображена на рисунку 1.15.

Рисунок 1.15. Іконка ПЗ

ComboBox-и за замовчуванням мають вузький розмір, тому їх ширина збільшується до 150–180 пікселів, щоб вміщалися повні назви моделей. Те саме з label-ами – довші назви обрізаються, якщо не вистачає ширини, тому вручну збільшується їхній Size. Особливо це стосується label1 і label2, які виводять результати – вони мають бути ширші за інші.

Далі налаштовується вирівнювання. Елементи не повинні бути розкидані по формі без порядку. В Visual Studio зручно виставити всі компоненти по сітці або з допомогою автоматичного вирівнювання по лівому краю. Важливо також залишити відступи між елементами, щоб вони не злипались і форма не виглядала перенасиченою. Для кращого сприйняття можна розділити форму горизонтально – верхня частина для вводу, нижня – для результатів. Це не робиться окремим компонентом, просто візуально розноситься блок елементів. Кнопка «Розрахувати» має бути центру роздільною лінією між вибором комплектуючих та виводом результатів.

Label1 та label2 будуть змінені з «Введіть дані» та « » на «Потужність ПК:» та «Рекомендована потужність блока живлення:» відповідно. Це потрібно, щоб користувач розумів звідки будуть виводитися потрібні йому дані, попередній варіант заплутував та не давав чіткого розуміння звідки будуть виводитися дані.

Колір тексту у всіх написах – чорний або темно-сірий, не варто використовувати кольоровий текст. Колір фону – однаковий для всієї форми, не

варто виділяти елементи кольорами, це створює візуальний шум. Ті самі принципи – мінімум кольорів, максимум простоти.

Розмір вікна буде незмінний, через те, що все необхідне присутнє у її звичайному вигляді і не потребує розширення.

Потрібна окрема кнопка для скидання введених даних, оскільки користувач може помилково вибрати неправильні параметри або просто захотіти швидко очистити всі поля без перезапуску програми. Це зменшує кількість кліків і робить інтерфейс зручнішим. Без кнопки скидання користувач буде змушений вручну повертати всі поля у початковий стан, що незручно і довго, особливо при заповненні кількох списків. Також скидання може знадобитися при повторному розрахунку з іншими параметрами. Кнопка button2 – скидання всіх параметрів форми. Розташовується зліва від button1 (розрахунок). Відстань між кнопками – 30 пікселів. Координати Location виставляються вручну, залежно від позиції button1, з урахуванням ширини обох кнопок. Висота така сама – 30 пікселів, ширина – 150. Це дозволяє зберегти однакову геометрію. Назва на кнопці – «Скинути». TextAlign по центру, шрифт той самий, що і на інших елементах (Microsoft Sans Serif, 8pt), щоб не вибивалось із загального вигляду. Параметр Anchor не виставляється, оскільки кнопка не повинна рухатись при зміні розміру форми. Позиція жорстко фіксована, як і у button1. Обробник події Click скидає всі значення у випадаючих списках comboBox1 та comboBox2 властивістю SelectedIndex = -1. Це очищає вибір процесора та відеокарти. Для викидних списків comboBox3 – comboBox6 властивість для скидання значень до 0 - SelectedIndex = 0. Необхідно очистити і значення результатів label1 і label2 – зробити їх такими як при запуску програми «Потужність ПК:» та «Рекомендована потужність блока живлення:». Це можна зробити властивістю label.Text = “Потужність ПК:” або “Рекомендована потужність блока живлення:”. Після чого усі елементи ПЗ будуть повернуті до початкового стану.

Для того, щоб користувач після натискання кнопки «Розрахувати» бачив процес обчислень візуально, треба додати ефект завантаження. Це можна зробити за допомогою курсора та методів .Delay() (для затримки процесу на певну

кількість мілісекунд) та .Wait() (для зависання усіх процесів форми до моменту кінця затримки). При натисканні «Розрахувати» буде виконуватися процес який замінює курсор на очікування та запускає методи .Delay(1000).Wait() для паузи програми на 1 секунду, тим самим користувач може побачити початок процесу розрахунку візуально.

Останній етап – перевірка відображення на екранах з різним масштабуванням (DPI). У Windows часто стоїть масштаб 125% або 150%, через що стандартні форми «розлазяться». Візуально перевіряється відображення на різних значеннях, якщо шрифти або компоненти збиваються – підбираються розміри вручну. У складних випадках можна задати AutoScaleMode = Dpi, щоб інтерфейс автоматично адаптувався.

Після усіх налаштувань інтерфейс програми стає не тільки повністю функціональним, але й зручним для користувача. Всі елементи розташовані логічно, немає нічого зайвого, що могло б відволікати або ускладнювати роботу. Графічний дизайн мінімалістичний, але не примітивний – усе виглядає просто, зрозуміло і водночас достатньо продумано для комфортної взаємодії. Користувач може одразу орієнтуватися в інтерфейсі без потреби в інструкціях або додаткових поясненнях. Програма нагадує стандартну утиліту, призначену для практичного використання. На рисунку 1.16 зображено остаточний вигляд програми з усіма оформленими графічними елементами.

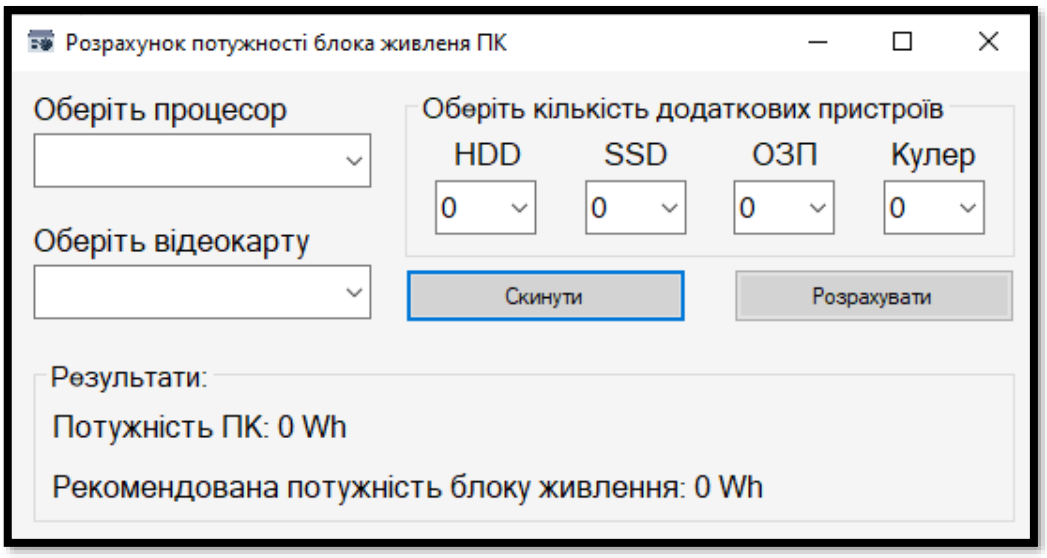


Рисунок 1.16. Остаточний вигляд програми

1.6 Реалізація основних елементів ПЗ

Початок реалізації структурної частини починається з реалізації форми, потрібно обрати форму, іконку, та її назву. Для цього було використано властивості .Text .Icon .Width .Height, Назва «Розрахунок блока живлення ПК», іконка була додана до директорії програми і має назву «psu.ico», ширина та висота відповідно 560 на 290 пікселів. Відповідний код зображений на рисунку 1.17.

```
1 reference
private void Form1_Load(object sender, EventArgs e)
{

    this.Text = "Розрахунок потужності блока живленя ПК";
    this.Icon = new Icon("psu.ico");
    this.Width = 560;
    this.Height = 290;

}
```

Рисунок 1.17. Налаштування властивостей форми

Після форми потрібно налаштувати текст у всіх написах на формі – label1, label2, label3, label4, label5, label6, label7, label9. Відповідно – «Потужність ПК:», «Рекомендована потужність блоку живлення:», «Оберіть процесор», «Оберіть відеокарту», «HDD», «ОЗП», «SSD», «Кулер». Код налаштування написів зображений на рисунку 1.18.

```
label1.Text = "Потужність ПК:";
label2.Text = "Рекомендована потужність блоку живлення:";
label3.Text = "Оберіть процесор";
label4.Text = "Оберіть відеокарту";
label5.Text = "HDD";
label6.Text = "ОЗП";
label7.Text = "SSD";
label9.Text = "Кулер";
```

Рисунок 1.18. Налаштування властивостей форми

Далі потрібно налаштувати початкові значення викидних списків, які відповідають за вибір процесора, відеокарти, та кількості додаткових комплектуючих у конфігурації комп'ютера. Це налаштовується через властивості .SelectedIndex та .Text. В цьому випадку comboBox1 і comboBox2, що

відповідають за значення моделей процесора та відеокарти мають бути змінені через властивість .SelectedIndex = -1. Інші викидні списки, які відповідають за додаткові компоненти (comboBox3, comboBox5, comboBox4, comboBox6) через властивість Text = 0. Налаштування початкових значень викидних списків зображено на рисунку 1.19.

```
comboBox1.SelectedIndex = -1;
comboBox2.SelectedIndex = -1;
comboBox3.Text = "0";
comboBox4.Text = "0";
comboBox5.Text = "0";
comboBox6.Text = "0";
```

Рисунок 1.19. Налаштування початкових значень викидних списків

Викидні списки вибору додаткових компонентів потрібно наповнити варіантами вибору, в цьому випадку – списком чисел від 0 до 6. Це можна зробити за допомогою властивості Items.Add. Для кожного викидного списку код їх заповнення зображений на рисунку 1.20, 1.21, 1.22, 1.23.

```
comboBox3.Items.Add("0");
comboBox3.Items.Add("1");
comboBox3.Items.Add("2");
comboBox3.Items.Add("3");
comboBox3.Items.Add("4");
comboBox3.Items.Add("5");
comboBox3.Items.Add("6");
```

Рисунок 1.20. Наповнення елементами comboBox3

```
comboBox4.Items.Add("0");
comboBox4.Items.Add("1");
comboBox4.Items.Add("2");
comboBox4.Items.Add("3");
comboBox4.Items.Add("4");
comboBox4.Items.Add("5");
comboBox4.Items.Add("6");
```

Рисунок 1.21. Наповнення елементами comboBox4

```
comboBox5.Items.Add("0");
comboBox5.Items.Add("1");
comboBox5.Items.Add("2");
comboBox5.Items.Add("3");
comboBox5.Items.Add("4");
comboBox5.Items.Add("5");
comboBox5.Items.Add("6");
```

Рисунок 1.22. Наповнення елементами comboBox5

```
comboBox6.Items.Add("0");
comboBox6.Items.Add("1");
comboBox6.Items.Add("2");
comboBox6.Items.Add("3");
comboBox6.Items.Add("4");
comboBox6.Items.Add("5");
comboBox6.Items.Add("6");
```

Рисунок 1.23. Наповнення елементами comboBox6

Програма не передбачає ввід власної кількості додаткових компонентів, тому потрібно заборонити користувачу вносити дані до викидного списку, через можливі помилки при подальших розрахунках. Заборона редагування вмісту поля викидного списку редагується властивістю DropDownStyle. Код редагування властивості DropDownStyle для всіх викидних списків зображений на рисунку 1.24.

```
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox2.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox3.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox4.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox5.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox6.DropDownStyle = ComboBoxStyle.DropDownList;
```

Рисунок 1.24. Код заборони редагування вмісту викидних списків

Для коректної роботи програми до викидних списків comboBox1 (список вибору процесора) та comboBox2 (список вибору відеокарти) необхідно

під’єднати базу даних. Процес підключення вже був детально описаний підрозділі 1.5.1 із зображенням та поясненням. Підключивши базу даних до методу Form1_Load додаються рядки коду, які відповідають за підключення до таблиць бази даних. Це зображено на рисунку 1.25.

```
// TODO: This line of code loads data into the 'powerDataSet.GPU' table.
// You can move, or remove it, as needed.
this.gPUTableAdapter.Fill(this.powerDataSet.GPU);
// TODO: This line of code loads data into the 'powerDataSet.CPU' table.
// You can move, or remove it, as needed.
this.cPUTableAdapter.Fill(this.powerDataSet.CPU);
```

Рисунок 1.25. Під’єднання до таблиць бази даних

Налаштування GroupBox складається із зміни їх тексту. Для groupBox1 та groupBox2 необхідно виставити значення «Оберіть кількість додаткових компонентів» та «Результати:» через властивість .Text. Налаштування значень groupBox-ів зображено на рисунку 1.26.

```
groupBox1.Text = "Оберіть кількість додаткових компонентів";
groupBox2.Text = "Результати:";
```

Рисунок 1.26. Налаштування значень GroupBox-ів

Перед завершальним етапом налаштування основних елементів потрібно налаштувати написи на кнопках, їх колір та розмір. Щоб змінити назви кнопок, потрібно змінити властивість Text, розмір – властивість Width, а за колір відповідає властивість BackColor. Налаштування кнопок зображено на рисунку 1.27.

```
button1.Text = "Розрахувати";
button1.Width = 150;
button2.Text = "Скинути";
button2.Width = 150;
button1.BackColor = Color.LightGray;
button2.BackColor = Color.LightGray;
```

Рисунок 1.27. Налаштування кнопок

Завершальний етап – це редагування стилю елементів ПЗ під обраний графічний дизайн. Редагування фонового кольору форми здійснюється через зміну властивості .BackColor на елемент списку Colors, також для форми потрібно виставити шрифт - Microsoft Sans Serif за допомогою властивості .Font. Необхідно вибрати шрифт Microsoft Sans Serif для таких елементів: label, groupBox, comboBox, button. Перед цим для зручності створюється екземпляри класу Font з потрібним шрифтом та розмірами, що зображено на рисунку 1.28.

```
Font mss8 = new Font("Microsoft Sans Serif", 8.25f);
Font mss11 = new Font("Microsoft Sans Serif", 11.25f);
Font mss12 = new Font("Microsoft Sans Serif", 12f);
```

Рисунок 1.28. Створення екземплярів шрифтів

Маючи екземпляри шрифтів можна зручно заповнювати властивості Font потрібних елементів. Зміна шрифту елементів зображена на рисунку 1.29.

```
this.Font = mss12;
label1.Font = mss12;
label2.Font = mss12;
label3.Font = mss12;
label4.Font = mss12;
label5.Font = mss12;
label6.Font = mss12;
label7.Font = mss12;
label9.Font = mss12;
groupBox1.Font = mss11;
groupBox2.Font = mss11;
comboBox1.Font = mss12;
comboBox2.Font = mss12;
comboBox3.Font = mss12;
comboBox4.Font = mss12;
comboBox5.Font = mss12;
comboBox6.Font = mss12;
button1.Font = mss8;
button2.Font = mss8;
```

Рисунок 1.29. Редагування шрифтів елементів які містять текст

1.7 Реалізація розрахунку потужності блоків живлення у ПЗ

Реалізація розрахунків відбувається після введення користувачем потрібних даних про комп'ютерні комплектуючі і натисканням кнопки «Розрахувати» (button1). Для реалізації розрахункової частини потрібно створити подію Click у вкладці Events, яка створить метод button1_Click, до якого буде записуватися вся розрахункова частина. Для зручності вона буде записуватися у блок try-catch для уникнення небажаного завершення програми. Блок try починається з графічної ілюзії процесу розрахунків, курсор змінюється на «очікування» і програма зупиняється на 1 секунду, після чого повертає звичайний курсор. Метод button1_Click і код зміни курсора та затримки у блоці try зображений на рисунку 1.30.

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        Cursor.Current = Cursors.WaitCursor;
        Task.Delay(1000).Wait();
        Cursor.Current = Cursors.Default;
    }
    catch
    {

    }
}
```

Рисунок 1.30. Редагування шрифтів елементів які містять текст

Після затримки має виконуватися код отримання значень із бази даних та викидних списків. Це реалізовано зміною параметру викидного списку, що відповідає за вибір стовпця потрібних даних у таблиці значень відеокарт. Далі йде зчитування обраних користувачем значень викидних списків та запис їх до змінних. Щоб записати дані третього стовпця таблиці відеокарт з бази даних використовується редагування властивості що відповідає за обраний стовпець, в цьому випадку «отримувати дані зі стовпця rec_power». Це зображено на рисунку 1.31 рядком comboBox2.ValueMember = "power" та comboBox2.ValueMember = "rec_power".

```
comboBox2.ValueMember = "power";
int cpu = Convert.ToInt32(comboBox1.SelectedValue);
int gpu = Convert.ToInt32(comboBox2.SelectedValue);
int hdd = Convert.ToInt32(comboBox3.Text) * 10;
int ssd = Convert.ToInt32(comboBox5.Text) * 15;
int ram = Convert.ToInt32(comboBox4.Text) * 5;
int fan = Convert.ToInt32(comboBox6.Text) * 15;
comboBox2.ValueMember = "rec_power";
int gpu_rec = Convert.ToInt32(comboBox2.SelectedValue);
```

Рисунок 1.31. Блок коду для зчитування даних

Процес розрахунку загальної максимальної потужності виконується через суму всіх потужностей комплектуючих. Але процес розрахунку рекомендованого блоку живленя має складний для розуміння алгоритм дії, з метою виводу числа з округленням до сотень. На рисунку 1.32 зображені етапи округлення до сотень.

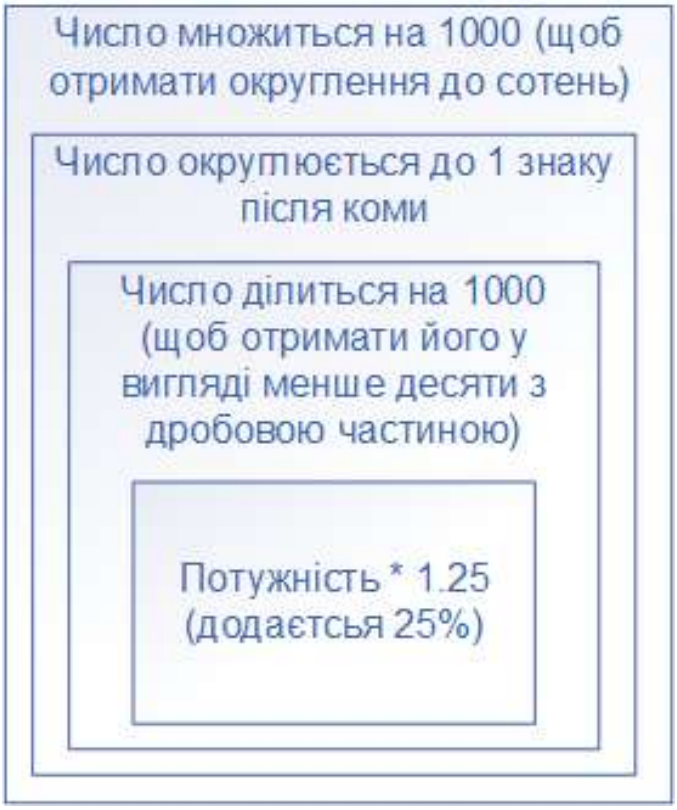


Рисунок 1.32. Опис етапів округлення до сотень

Щоб перевести число з цілочисельного типу в тип з плаваючою крапкою, було використано Convert.ToDouble. Для округлення – метод Math.Round. Блок коду який відповідає за розрахунок потужності системи та рекомендованої потужності блока живлення зображений на рисунку 1.33.

```
int power = (cpu + gpu + hdd + ssd + ram + fan);
double power_bp = (Math.Round(
    ((Convert.ToDouble(power) * 1.25) / 1000), 1)) * 1000;
```

Рисунок 1.33. Блок коду розрахунків

Для виводу потрібних значень до інтерфейсу, змінюються властивості Text написів label1 та label2. Відповідно алгоритму, якщо рекомендована потужність після розрахунків буде менша за рекомендовану потужність блока живлення від виробника відповідної відеокарти, виведеться значення рекомендованої потужності з таблиці відеокарт, інакше виведеться потужність після розрахунків. Блок коду виводу інформації до користувача зображений на рисунку 1.34.

```
label1.Text = ("Потужність ПК: " + power.ToString() + " Wh");
if (Convert.ToInt32(power_bp) <= gpu_rec)
{

    label2.Text = ("Рекомендована потужність блоку живлення: "
    + gpu_rec.ToString() + " Wh");

}
else
{
    label2.Text = ("Рекомендована потужність блоку живлення: "
    + power_bp.ToString() + " Wh");
}
```

Рисунок 1.34. Блок коду виводу інформації користувачу

Також для виводу користувачу попередження для спонукання його вибрати комплектуючі у блоці catch буде виконуватися виклик повідомлення з написом «Оберіть комплектуючі» у разі невиконання основної програми. Блок catch зображений на рисунку 1.35.

```
catch
{
    MessageBox.Show("Оберіть комплектуючі");
}
```

Рисунок 1.35. Блок catch з викликом вікна повідомлення

Реалізація кнопки скидання реалізовується подією button2_Click. До цього методу записується блок коду для редагування властивості SelectedIndex, тобто скидаючи значення викидних списків. Блок коду скидання із методом button2_Click зображений на рисунку 1.36.

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    comboBox1.SelectedIndex = -1;
    comboBox2.SelectedIndex = -1;
    comboBox3.SelectedIndex = 0;
    comboBox4.SelectedIndex = 0;
    comboBox5.SelectedIndex = 0;
    comboBox6.SelectedIndex = 0;
    label2.Text = ("Рекомендована потужність блоку живлення:");
    label1.Text = ("Потужність ПК:");
}
```

Рисунок 1.36. Блок методу button2_Click з реалізацією скидання значень

1.8 Тестування розробленого ПЗ

Під час процесу тестування потрібно перевірити коректність роботи програми. Основна функція ПЗ – це розрахувати потужність блока живлення. Для тестування цієї функції потрібно ввести значення до викидних списків, що буде зображено на рисунку 1.37.

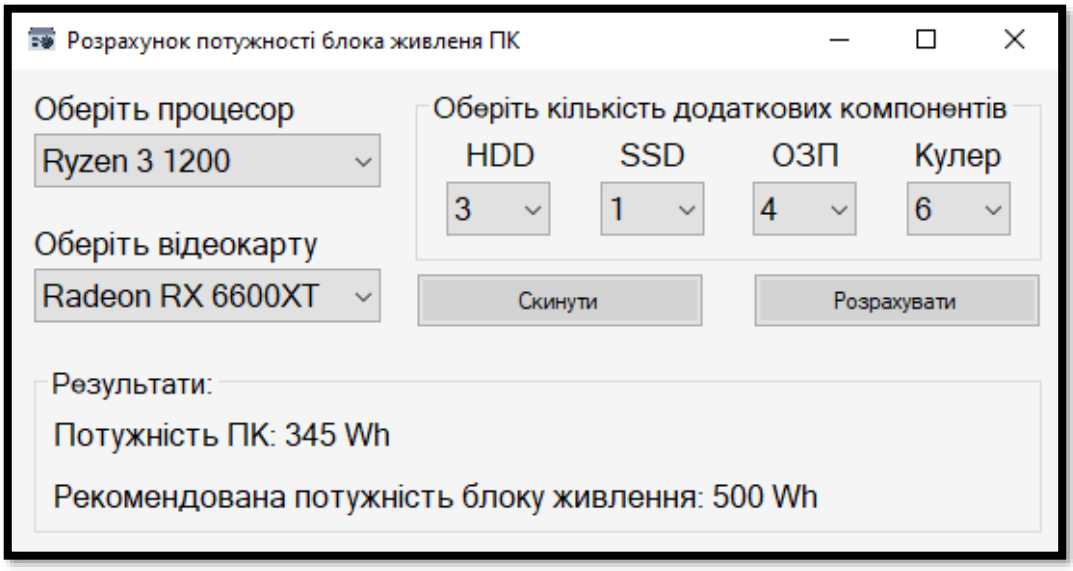


Рисунок 1.37. Тестування функції розрахунку блока живлення

Як зображено на рисунку 1.37, програма правильно розраховує

рекомендований блок живлення для цієї конфігурації. Але зараз програма виводить рекомендовану потужність, що заявлена виробником відеокарти. Потрібно перевірити інший випадок, коли навантаження системи більше за рекомендований блок живлення, що зображено на рисунку 1.38.

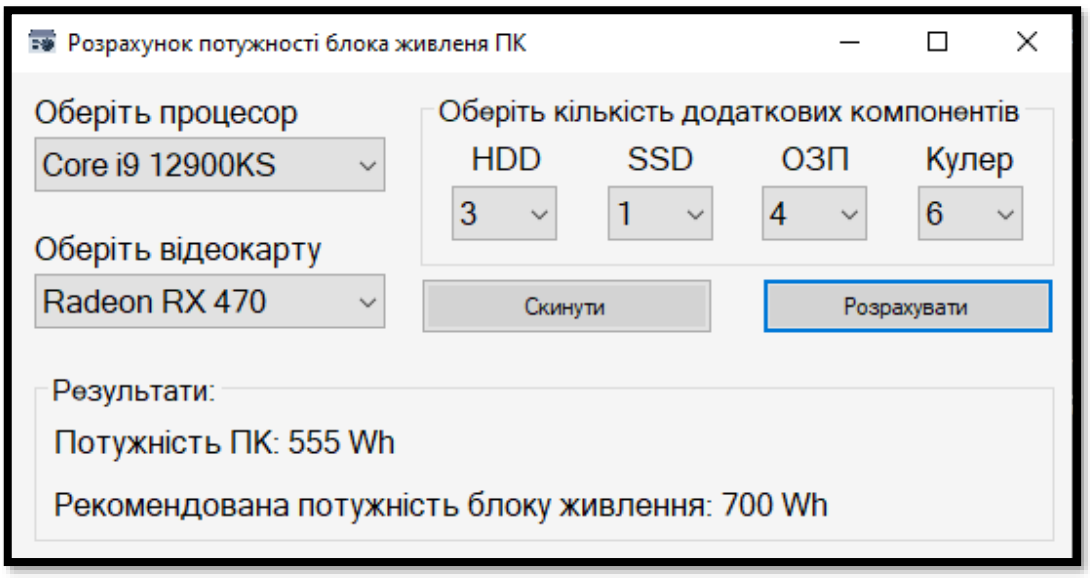


Рисунок 1.38. Тестування функції розрахунку блока живлення

Програма розрахувала рекомендовану потужність блоку живлення із запасом у цьому випадку. Перевірка функції скидання значень зображена на рисунку 1.39.

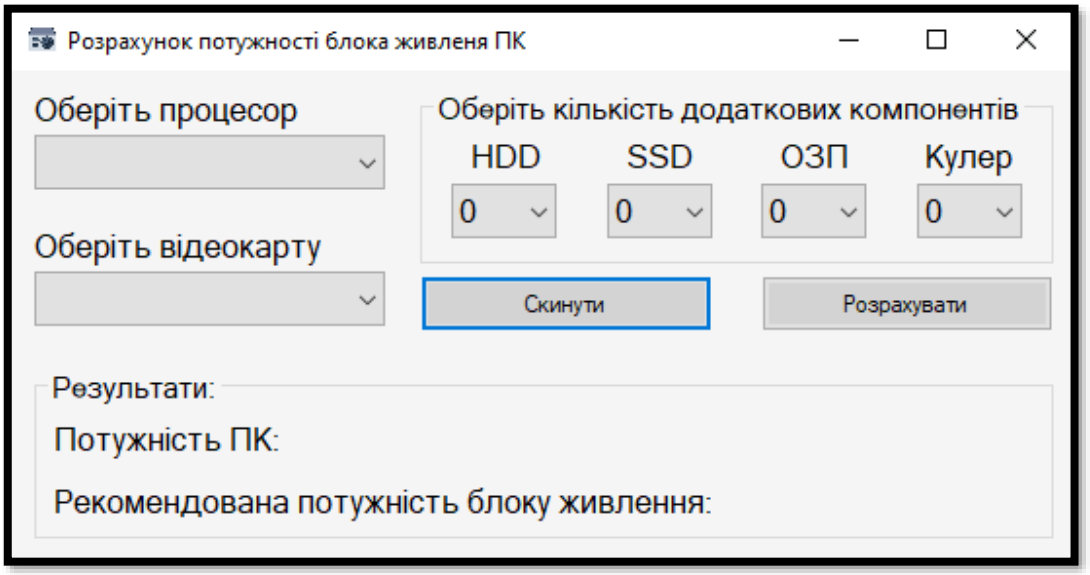


Рисунок 1.39. Тестування функції скидання значень

Після проведення процесу тестування, було виявлено, що програма виконує

свої обов’язкові та додаткові функції. Тому розробка і реалізація програмного забезпечення розрахунку рекомендованого блоку живлення персонального комп’ютера завершена.

Під час роботи можна обрати різні комбінації процесорів та відеокарт, а також комбінації додаткових компонентів для конфігурації персонального комп’ютера, що розглядається. Дані до полі процесорів та відеокарт надходять із бази даних, яку можна оновлювати новими комплектуючими, як тільки стає доступною інформація про них. Це дає змогу відразу актуалізувати програму для користувачів та реалізовувати цю актуалізацію із поставкою нових версій розробленого програмного забезпечення.

У подальшому програму можна покращити додавши до неї функцію підбору блоку живлення для обраної конфігурації персонального комп’ютера, щоби користувач міг відразу перейти до пошуку пропозицій у мережі Інтернет. Також, можна реалізувати систему онлайн-оновлення, щоби користувачі могли отримувати нові версії програми автоматично, без необхідності завантажувати інсталяційні файли програми.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У результаті виконаної роботи створено готовий до використання застосунок, який дозволяє оцінити орієнтовну потужність блоку живлення, необхідного для надійної роботи комп'ютера. Інтерфейс простий і зрозумілий, що дозволяє користуватись ним без спеціальних знань у сфері комп'ютерного обладнання. В подальшому програму дуже легко приводити до актуального стану, додаючи до неї параметри нових комп'ютерних комплектуючих.

Ефективність програмного забезпечення визначається як його якістю, так і результативністю процесу розробки. Якість ПЗ оцінюється за різними параметрами: з точки зору користувача, ефективності використання ресурсів і відповідності заданим вимогам. Для користувача важливими критеріями є витрати на розробку, зокрема обсяг трудових ресурсів. У цьому розділі розглядається розрахунок вартості розробленого програмного продукту.

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 ПЗ	*Обсяг функції ПЗ – V_o, усл. машинних командах.*
1. ПЗ автоматизації засобів по каталогу	680 – 7000
2. ПЗ автоматизованих розрахунків	1300 – 8600
3. ПЗ оптимізації розрахунків	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. Таблиця трудомісткості

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:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

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 ПЗ

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 ПЗ	1,75 – 1,2
Б	ПЗ – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПЗ 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 ПЗ 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.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

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

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н=183,2 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=183,2 *0,61*0,7*0,6= 46,94 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=3 (стр), розробка ТП $N_{тп}$=29 (стр), розробка робочого проекту $N_{рп}$=8 (стр), пояснювальна записка відповідно $N_{пз}$ 20 (стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПЗ

Найменування етапів	*Розрахунок, годин.*		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*3=2,1	$T_{нк}=0,15*N_{тз}$=0,15*3=0,45
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}$=0,7*29=20,3	$T_{нк}=0,15*N_{тп}$=0,15*29=4,35
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}$=0,7*8=5,6	$T_{нк}=0,15*N_{рп}$=0,15*8=1,2
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*20=30	$T_{кк}=0,7*N_{тз}$=0,7*20=14	$T_{нк}=0,15*N_{пз}$=0,15*20 =3
Усього, в т.ч.:	157,43		
- на розробку	$\Sigma T_р$=106,43		
- контроль керівника		$\Sigma T_{кк}$=42	
- нормоконтроль			$\Sigma T_{нк}$=9

2.3 Розрахунок ціни програмного продукту

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПЗ. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2025» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2025 року – 8000 гривень; мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	*Трудомісткість робіт, години*	*Погодинна тарифна ставка, грн.*	*Розрахунок, грн.*
1.Розробка ПЗ	106,43	48,00	5108,64
2.Контроль керівника	42	120,00	5040,00
3.Нормоконтроль	9	120,00	1080,00
Усього	-	-	ΣЗо= 11228,64

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

Таблиця 2.8 Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	*Тип, модель*	*Кількість*	*Ціна одиниці, грн.*	*Вартість, грн.*
Папір	Лист А4	60	5.0	300,0
Разом	-	-	-	$В_{м\,і}$ =300,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1*300=30,0
Усього				$В_м=В_{мі}+В_{тр\ з=}$ 330,0

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

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	*Значення, грн.*	*Формула розрахунку*
1. Матеріали	330,0	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	11228,64	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1122,86	$Зд = 0{,}1 \times Зо$ = 11228,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2717,33	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (11228,64+1122,86)
5. Накладні витрати	4491,46	$Внак. = 0{,}4 \times Зо$ =0,4 * 11228,64
6. Повна собівартість	19890,29	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 330,0+11228,64+1122,86+2717,33 +4491,46

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

$$П=(С_{пов}*Р)/100=(19890,29*15)/100=2983,54 \text{ грн.} \quad (2.4)$$

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

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = 19890,29+2983,54 = 22873,83 \text{ грн.} \quad (2.5)$$

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

$$Ц_р=Ц_о+ПДВ = 22873,83 + 22873,83 *0.2 = 27448,59 \text{ грн.} \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Науково-технічний прогрес суттєво вплинув на умови праці програмістів, зробивши їхню діяльність більш інтенсивною та напруженою, що вимагає значних витрат розумових, емоційних і фізичних ресурсів. Це зумовило необхідність комплексного підходу до питань ергономіки, гігієни, організації робочого процесу, а також впровадження чітких регламентів щодо режимів праці та відпочинку.

Сьогодні комп'ютерні технології активно використовуються у всіх сферах людської діяльності. Однак робота програміста передбачає вплив низки потенційно небезпечних факторів, таких як електромагнітне поле, інфрачервоне та іонізуюче випромінювання, шум, вібрація, статична електрика.

Важливість розробки заходів з охорони праці обумовлена впливом комп’ютерного обладнання на зір, поставу, нервову систему та загальне навантаження на організм. Потрібно враховувати ергономіку, електробезпеку, мікроклімат і режим роботи.

3.1 Аналіз небезпечних і шкідливих чинників, що впливають на програміста

На багатьох офісних підприємствах програмісти та інші фахівці, що працюють із комп'ютерними системами, стикаються з несприятливими умовами праці. Шкідливі та небезпечні виробничі фактори тісно взаємопов’язані та можуть значно впливати на самопочуття і продуктивність працівників.

Усі шкідливі фактори умовно поділяються на такі групи:

Фізичні—охоплюють різноманітні зовнішні умови, що впливають на організм.

Психофізіологічні—напружений робочий ритм, високі навантаження та стрес.

Деякі фактори є неминучими у робочому середовищі. Серед найбільш значущих:

Температурний режим, підвищена вологість та різні види випромінювання.

Електромагнітні поля.

Шумове навантаження.

Освітлення — надмірна яскравість чи недостатня освітленість, що негативно впливає на зір.

Окремо ці фактори можуть не становити серйозної загрози, але їх поєднання та тривалий вплив можуть призводити до відчутних наслідків для здоров’я працівників.

3.2 Розробка заходів з охорони праці

Охорона праці на виробництві—важливий аспект, що впливає на ефективність і безпеку діяльності підприємства. Для забезпечення комфортних умов роботи необхідно розробити заходи, спрямовані на мінімізацію ризиків. Вони визначаються відповідно до загальних завдань компанії та специфіки робочого процесу.

Одним із ключових напрямків є ергономічне проектування робочих місць із відеотерміналами. Важливо, щоб робочий простір відповідав антропометричним, фізичним і психологічним вимогам працівників. Відповідність цим критеріям сприяє зниженню втоми, підвищенню продуктивності та зменшенню негативного впливу робочого середовища.

При організації робочого місця програміста слід враховувати такі основні вимоги:

Оптимальне розташування обладнання, що забезпечує зручність у роботі.

Достатній робочий простір, який дає можливість комфортно виконувати всі необхідні дії.

Врахування характеру роботи, зокрема необхідність тривалого перебування за комп’ютером. Створення сприятливого робочого середовища сприяє покращенню здоров'я працівників та підвищенню їхньої продуктивності.

3.2.1 Виробниче освітлення

Робота з обчислювальною технікою створює певні виробничі фактори, що можуть впливати на комфорт та ефективність працівника. Серед них:

Ймовірність виникнення прямої блискості, що може спричиняти дискомфорт і підвищене навантаження на очі.

Погіршення контрастності між зображенням і фоном, що ускладнює сприйняття інформації.

Розмитість зображення на екрані, що впливає на якість роботи та візуальну зручність.

Через недостатню інтенсивність природного освітлення необхідне використання штучного світла. У дипломному проєкті запропоновано встановити два світильники типу ЛДР (2×40 Вт), загальний світловий потік яких становить 5700 лм. Таке розташування освітлення покликане забезпечити оптимальні умови роботи та зменшити вплив шкідливих факторів рис.6.1.

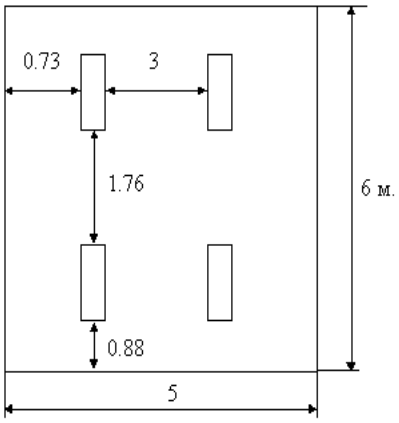


Рисунок 3.1 Схема розташування світильників

3.2.2 Шум, вібрація

На підприємствах, де в офісах стоять комп'ютери та інше обладнання, без шуму, як правило, не обходиться. Постійно працює техніка видає гучні звуки, які можуть змінювати свою інтенсивність від стану навантаження системи. Якщо людина змушена регулярно зазнавати такого впливу, то це негативно позначиться на його здоров'ї. Від сильного шуму починає боліти голова, підвищується тиск, знижується гострота слуху. Зменшення негативного впливу шуму на працівника для цього рекомендують використовувати:

– виробляти звукоізоляцію галасливих місць з допомогою використання захисних кожухів, обладнання кабінок;

– оздоблення приміщень звукопоглинаючими матеріалами;
– облицювання стелі та стін звукопоглинальним матеріалом (знижує шум на 6-8 дб);
– екранування робочого місця (постановкою перегородок, діафрагм);
– установка в комп'ютерних приміщеннях обладнання, що виробляє мінімальний шум.

3.2.3 Організація робочого місця програміста

При організації робочого місця програміста повинно бути дотримані наступні основні умови: оптимальне розміщення устаткування, що до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення (рис. 3.2).

Рисунок 3.2. Організація робочого місця програміста

Робота програміста передбачає тривале перебування в сидячому положенні, що може негативно впливати на опорно-руховий апарат через недостатню рухливість. Особливу роль у цьому процесі відіграє плечовий пояс, адже будь-яке переміщення рук впливає не тільки на м’язи спини, а й на положення хребта, тазу та навіть ніг. Щоб уникнути негативних наслідків, важливо дотримуватися режиму праці та відпочинку, регулярно змінювати положення тіла та виконувати фізичні вправи, які активізують незадіяні групи м’язів.

Щодо робочого місця, для комфортної роботи програміста необхідно, щоб його площа становила не менше 6 м², висота приміщення 4 м, а обсяг—20 м³ на

одну людину. Однак аналіз реальних робочих місць показав, що ці параметри часто не відповідають стандартам, що може негативно позначатися на продуктивності та здоров'ї працівника.

Для покращення умов роботи програміста пропонується:

Робочий стіл: висота поверхні 720 мм (бажано регульована в межах 680-780 мм).

Розміри столу: 1600 × 1000 мм.

Простір для ніг: глибина 650 мм.

Підставка для ніг: довжина 400 мм, ширина 350 мм, нахил 15°.

Розташування клавіатури: не більше 300 мм від краю столу для комфортної опори передпліч.

Відстань між очима та екраном: 40-80 см, що забезпечує оптимальну видимість без надмірного навантаження на очі.

Правильна організація робочого місця не лише сприяє збереженню здоров'я працівника, а й підвищує його продуктивність. Якщо потрібно, можу допомогти додатково оптимізувати опис або внести коригування (рис. 3.3).

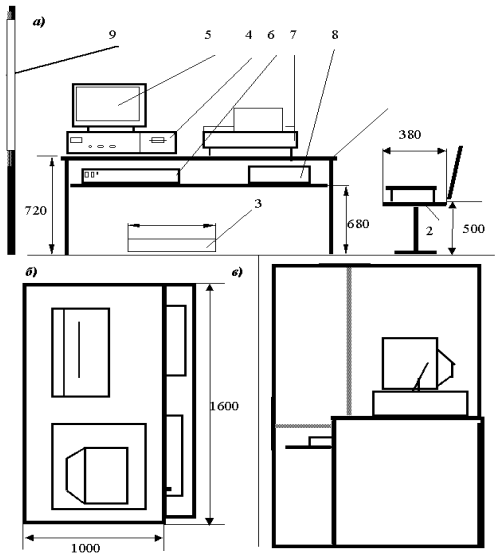


Рисунок 3.3. Розташування техніки на робочому місці програміста

ВИСНОВКИ

У процесі виконання дипломного проєкту за темою «Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК» було розроблено десктопний застосунок для розрахунку потужності блоку живлення персонального комп’ютера. Перед початком реалізації було проаналізовано існуючі рішення та проведено дослідження типових конфігурацій ПК, що дозволило визначити основні параметри, які впливають на споживання енергії.

Було розроблено структуру застосунку, визначено основні модулі, їхню функціональність та взаємодію між собою. На основі цього виконано проектування інтерфейсу користувача, який реалізовано засобами Windows Forms. Інтерфейс побудований так, щоб користувач міг швидко ввести конфігурацію свого ПК та отримати результат без зайвих дій.

Для реалізації застосунку використано середовище розробки Visual Studio та мову програмування C#. Код організовано за принципами об’єктно-орієнтованого програмування. Основна логіка винесена в окремі класи, що спрощує підтримку і можливе розширення функціоналу в майбутньому.

Під час завершаючої частини проєкту виконано тестування та відлагодження програми. Перевірено коректність розрахунків при різних варіантах введення даних. Було усунено критичні помилки, які могли спричинити некоректну роботу програми або її аварійне завершення.

У результаті виконаної роботи створено готовий до використання застосунок, який дозволяє оцінити потужність блоку живлення, у номінальних та рекомендованих значеннях. Інтерфейс простий і зрозумілий, що дозволяє користуватись ним без спеціальних знань у сфері комп’ютерного обладнання. В подальшому програму дуже легко приводити до актуального стану, додаючи до бази даних комплектуючих процесорів та відеокарт параметри нових пристроїв. Окрім того є шляхи для покращення розробленого ПЗ з точки зору реалізації функціоналу виведення рекомендованого БЖ та онлайн-оновлень.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. В.А. Ровінський, Програмування мовою C#: навчальний посібник, Івано-Франківськ, Сімик, 2016 р.;
2. Ієн Гріффітс, Програмування на C# 5.0., O`REILLY, 2016 р.;
3. Коваль В.С., Алгоритми і структури даних: Навчальний посібник, Тернопіль: ФОП Шпак В. Б., 2017 р.;
4. Козак Л. І., Костюк І. В., Стасевич С. П., Основи програмування, Львів: «Новий Світ-2000», 2020 р.;
5. Коноваленко І.В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І.В., Марущак П.О. – Тернопіль: ФОП Паляниця В. А.,
6. Матвієнко М.П., Теорія алгоритмів., Ліра К, 2018 р.;
7. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4;
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms – Boston., McGraw-Hill, 2001. – 1082p.;
9. Robert Sedgewick, Kevin Wayne. Algorithms - Addison-Wesley, 2020. – 956 p. – ISBN 978-0321573513;
10. C# Guide - .NET managed language | Microsoft Learn: [Веб-сайт]. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 10.05.2025);
11. Характеристики енергоспоживання відеокарт MSI: [Веб-сайт]. URL: https://www.msi.com/Graphics-Card/ (дата звернення: 13.05.2025);
12. Характеристики енергоспоживання процесорів Intel. [Веб-сайт]. URL: https://www.intel.com/content/ (дата звернення: 12.05.2025);
13. Характеристики енергоспоживання процесорів та відеокарт AMD. [Веб-сайт]. URL: https://www.amd.com/ (дата звернення: 17.05.2025).

ДОДАТОК А. Лістинг основних модулів ПЗ мовою C#

Скрипт Form1.cs

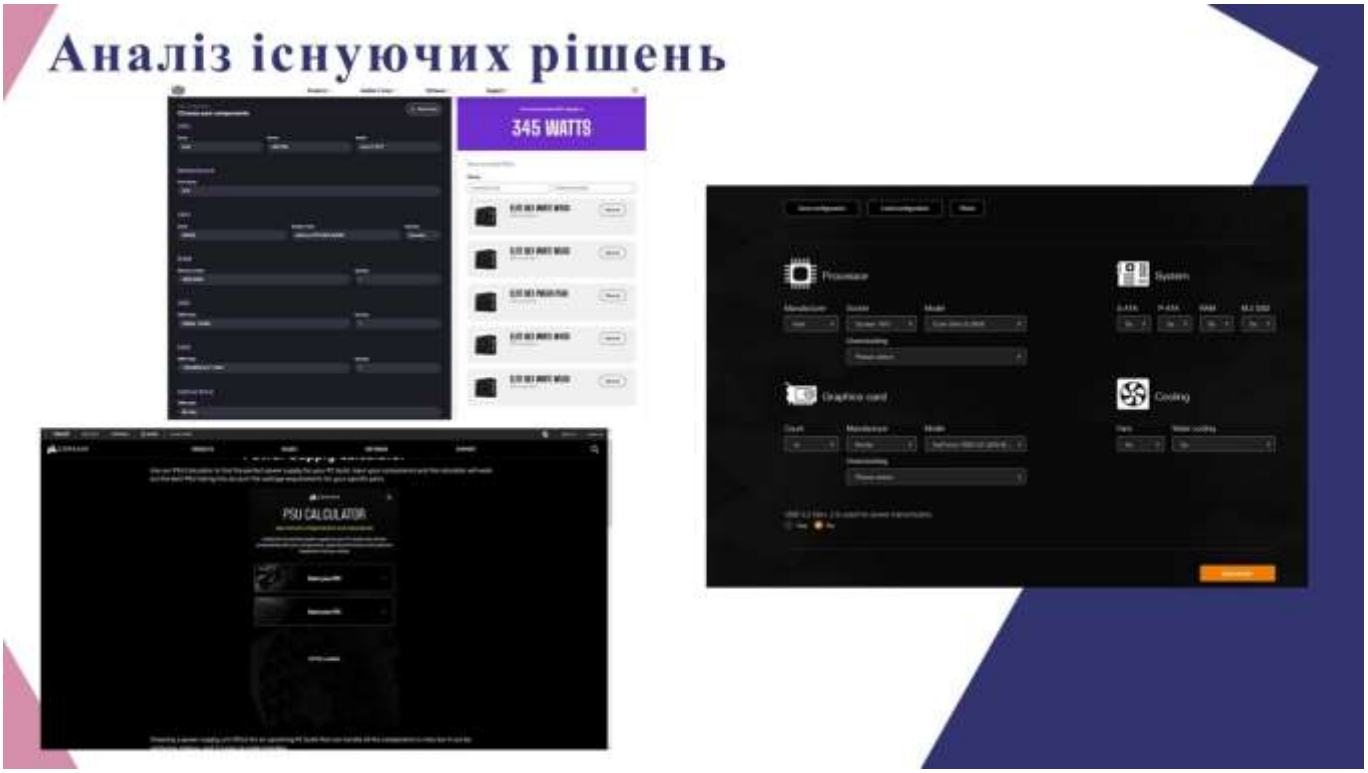
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices.ComTypes;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
namespace PidrahunokPotujnosti
{
    public partial class Form1: Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'powerDataSet.GPU' table.
            // You can move, or remove it, as needed.
            this.gPUTableAdapter.Fill(this.powerDataSet.GPU);
            // TODO: This line of code loads data into the 'powerDataSet.CPU' table.
            // You can move, or remove it, as needed.
            this.cPUTableAdapter.Fill(this.powerDataSet.CPU);

            this.Text = "Розрахунок потужності блока живленя ПК";
            this.Icon = new Icon("psu.ico");
            this.Width = 560;
            this.Height = 290;

            label1.Text = "Потужність ПК:";
            label2.Text = "Рекомендована потужність блоку живлення:";
            label3.Text = "Оберіть процесор";
            label4.Text = "Оберіть відеокарту";
            label5.Text = "HDD";
            label6.Text = "ОЗП";
            label7.Text = "SSD";
            label9.Text = "Кулер";

            comboBox1.SelectedIndex = -1;
            comboBox2.SelectedIndex = -1;
            comboBox3.Text = "0";
            comboBox4.Text = "0";
            comboBox5.Text = "0";
            comboBox6.Text = "0";

            comboBox3.Items.Add("0");
            comboBox3.Items.Add("1");
            comboBox3.Items.Add("2");
            comboBox3.Items.Add("3");
            comboBox3.Items.Add("4");
            comboBox3.Items.Add("5");
```

```
comboBox3.Items.Add("6");

comboBox4.Items.Add("0");
comboBox4.Items.Add("1");
comboBox4.Items.Add("2");
comboBox4.Items.Add("3");
comboBox4.Items.Add("4");
comboBox4.Items.Add("5");
comboBox4.Items.Add("6");

comboBox5.Items.Add("0");
comboBox5.Items.Add("1");
comboBox5.Items.Add("2");
comboBox5.Items.Add("3");
comboBox5.Items.Add("4");
comboBox5.Items.Add("5");
comboBox5.Items.Add("6");

comboBox6.Items.Add("0");
comboBox6.Items.Add("1");
comboBox6.Items.Add("2");
comboBox6.Items.Add("3");
comboBox6.Items.Add("4");
comboBox6.Items.Add("5");
comboBox6.Items.Add("6");

comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox2.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox3.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox4.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox5.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox6.DropDownStyle = ComboBoxStyle.DropDownList;

button1.Text = "Розрахувати";
button1.Width = 150;
button2.Text = "Скинути";
button2.Width = 150;
button1.BackColor = Color.LightGray;
button2.BackColor = Color.LightGray;

groupBox1.Text = "Оберіть кількість додаткових компонентів";
groupBox2.Text = "Результати:";

this.BackColor = Color.WhiteSmoke;

Font mss8 = new Font("Microsoft Sans Serif", 8.25f);
Font mss11 = new Font("Microsoft Sans Serif", 11.25f);
Font mss12 = new Font("Microsoft Sans Serif", 12f);

this.Font = mss12;
label1.Font = mss12;
label2.Font = mss12;
label3.Font = mss12;
label4.Font = mss12;
label5.Font = mss12;
label6.Font = mss12;
label7.Font = mss12;
label9.Font = mss12;
groupBox1.Font = mss11;
groupBox2.Font = mss11;
```

```
    comboBox1.Font = mss12;
    comboBox2.Font = mss12;
    comboBox3.Font = mss12;
    comboBox4.Font = mss12;
    comboBox5.Font = mss12;
    comboBox6.Font = mss12;
    button1.Font = mss8;
    button2.Font = mss8;

}
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        Cursor.Current = Cursors.WaitCursor;
        Task.Delay(1000).Wait();
        Cursor.Current = Cursors.Default;

        comboBox2.ValueMember = "power";
        int cpu = Convert.ToInt32(comboBox1.SelectedValue);
        int gpu = Convert.ToInt32(comboBox2.SelectedValue);
        int hdd = Convert.ToInt32(comboBox3.Text) * 10;
        int ssd = Convert.ToInt32(comboBox5.Text) * 15;
        int ram = Convert.ToInt32(comboBox4.Text) * 5;
        int fan = Convert.ToInt32(comboBox6.Text) * 15;
        comboBox2.ValueMember = "rec_power";
        int gpu_rec = Convert.ToInt32(comboBox2.SelectedValue);

        int power = (cpu + gpu + hdd + ssd + ram + fan);
        double power_bp = (Math.Round(
            ((Convert.ToDouble(power) * 1.25) / 1000), 1)) * 1000;

        label1.Text = ("Потужність ПК: " + power.ToString() + " Wh");
        if (Convert.ToInt32(power_bp) <= gpu_rec)
        {

            label2.Text = ("Рекомендована потужність блоку живлення: "
            + gpu_rec.ToString() + " Wh");

        }
        else
        {
            label2.Text = ("Рекомендована потужність блоку живлення: "
            + power_bp.ToString() + " Wh");
        }
    }
    catch
    {
        MessageBox.Show("Оберіть комплектуючі");
    }

}

private void button2_Click(object sender, EventArgs e)
{
    comboBox1.SelectedIndex = -1;
    comboBox2.SelectedIndex = -1;
    comboBox3.SelectedIndex = 0;
    comboBox4.SelectedIndex = 0;
    comboBox5.SelectedIndex = 0;
```

```
                comboBox6.SelectedIndex = 0;
                label2.Text = ("Рекомендована потужність блоку живлення:");
                label1.Text = ("Потужність ПК:");
            }
        }

    }
```

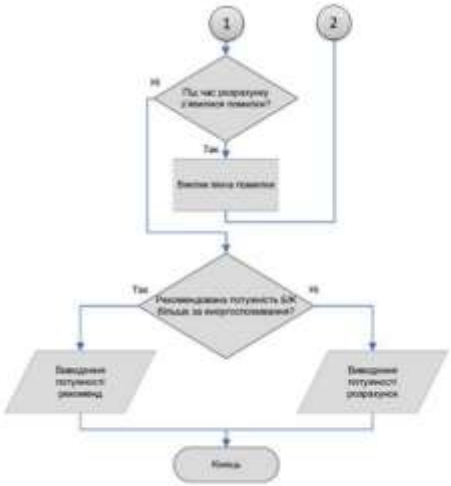
ДОДАТОК Б. Слайди мультимедійної презентації



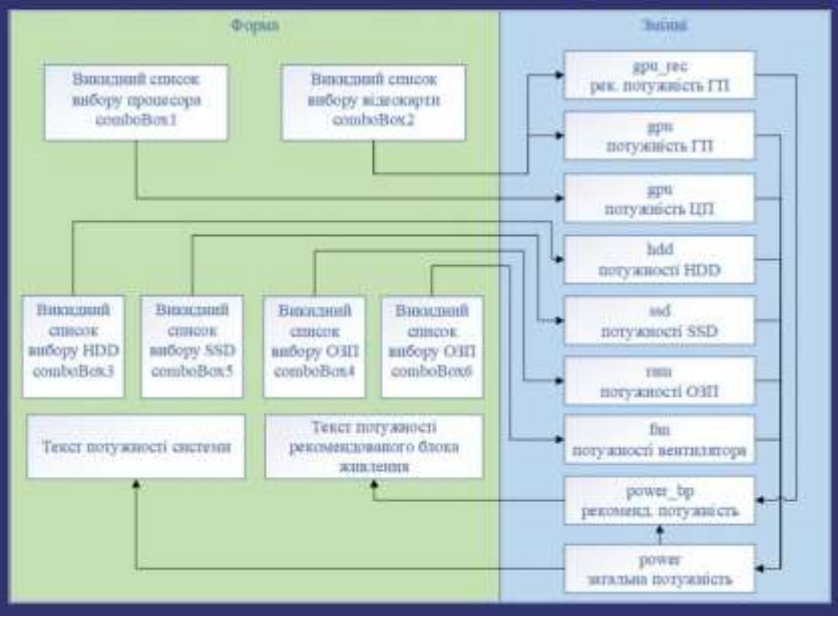
Інструменти розробки програми для розрахунку потужності БЖ

Microsoft® SQL Server®

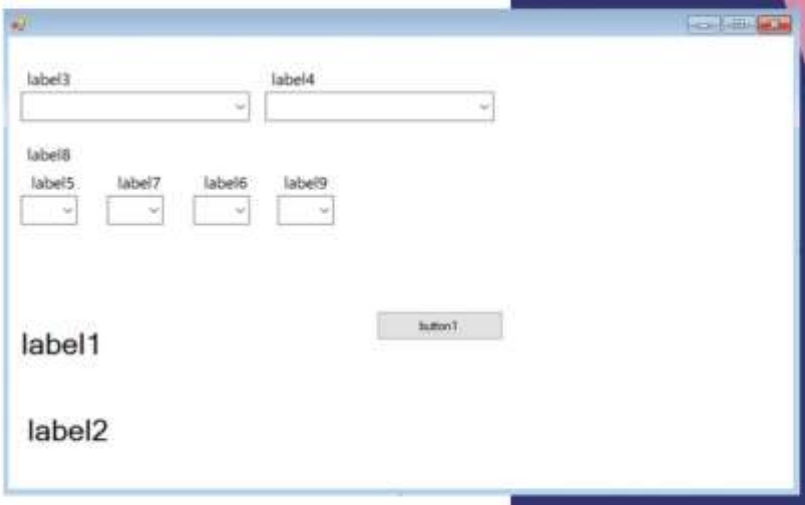
Структура розробляємого ПЗ



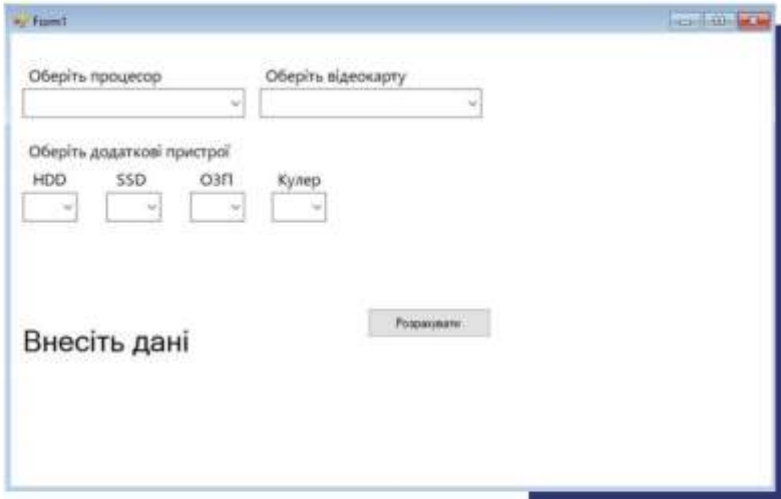
Принцип розрахунку потужності блоків живлення



Процес реалізації інтерфейсу ПЗ



Інтерфейс ПЗ із відредагованими елементами



Створення іконки програми та її заголовку

```
private void Form1_Load(object sender, EventArgs e)
{

    this.Text = "Розрахунок потужності блока живлення ПК";
    this.Icon = new Icon("pic.ico");
    this.Width = 500;
    this.Height = 290;

}
```

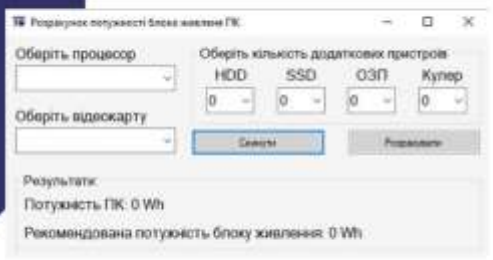
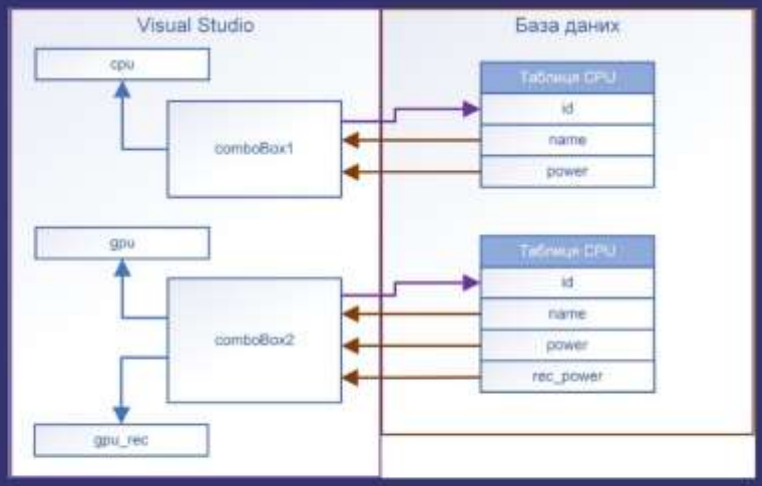
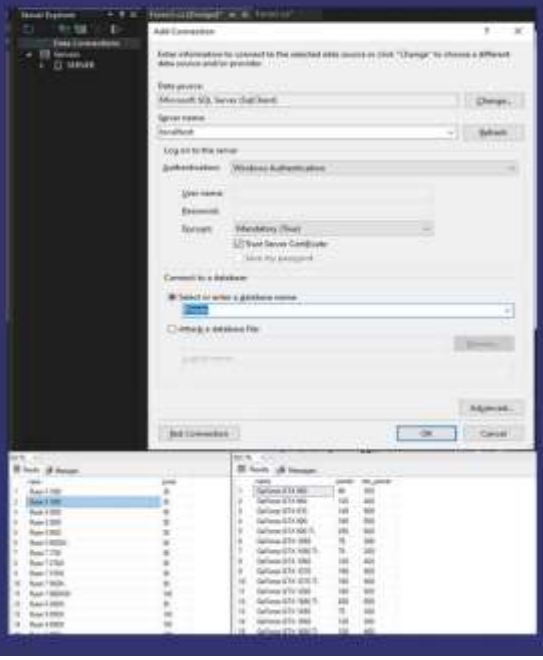


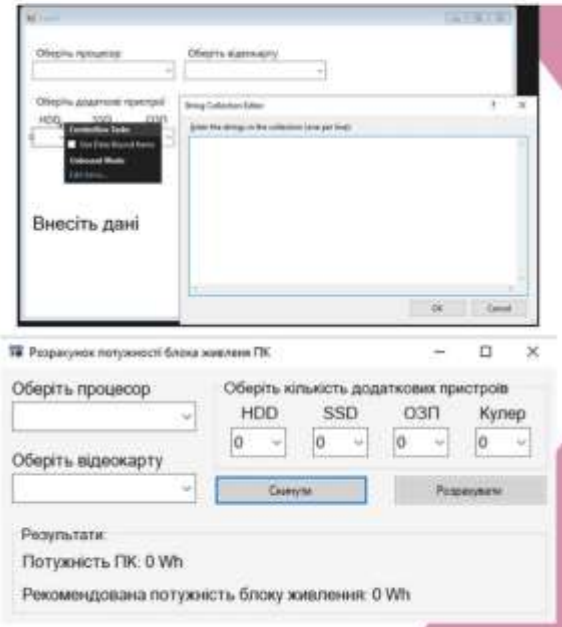
Схема зв'язків між складовими бази даних та елементами програми



Процес реалізації бази даних для ПЗ



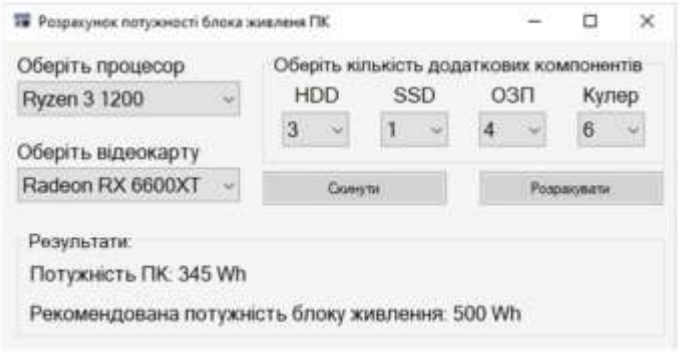
Процес реалізації основних елементів ПЗ



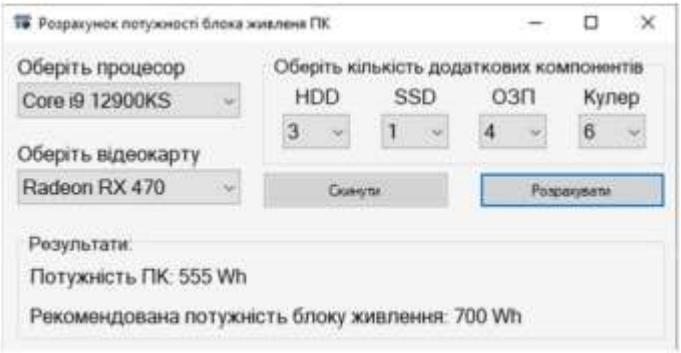
Написання коду для основних елементів ПЗ

```
comboBox2.ValueMember = "power";
int cpu = Convert.ToInt32(comboBox1.SelectedValue);
int gpu = Convert.ToInt32(comboBox2.SelectedValue);
int hdd = Convert.ToInt32(comboBox3.Text) * 10;
int ssd = Convert.ToInt32(comboBox5.Text) * 15;
int ram = Convert.ToInt32(comboBox4.Text) * 5;
int fan = Convert.ToInt32(comboBox6.Text) * 15;
comboBox2.ValueMember = "rec_power";
int gpu_rec = Convert.ToInt32(comboBox2.SelectedValue);
int power = (cpu + gpu + hdd + ssd + ram + fan);
double power_bp = (Math.Round(
    ((Convert.ToDouble(power) * 1.25) / 1000), 1)) * 1000;

label1.Text = ("Потужність ПК: " + power.ToString() + " Wh");
if (Convert.ToInt32(power_bp) <= gpu_rec)
{

    label2.Text = ("Рекомендована потужність блоку живлення: "
    + gpu_rec.ToString() + " Wh");

}
else
{
    label2.Text = ("Рекомендована потужність блоку живлення: "
    + power_bp.ToString() + " Wh");
}
```

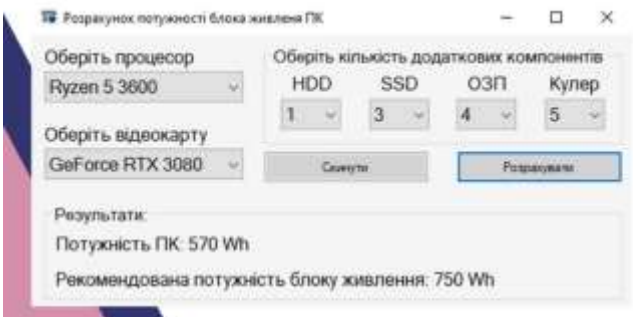
Тестування ПЗ

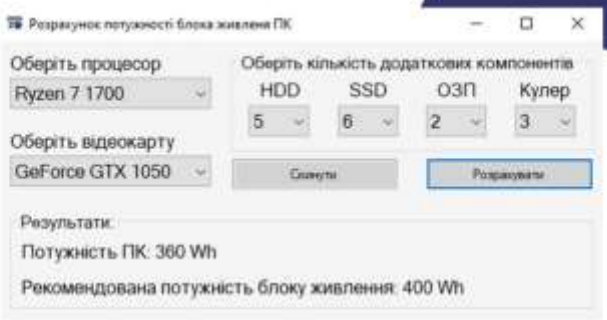


Тестування ПЗ



Скріншоти розробленого ПЗ





ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Маннапов Денис Хамітович

(прізвище, ім`я та по батькові)

Спеціальність *123 " Комп'ютерна інженерія "*

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

Керівник дипломного проекту (роботи) *Шувалова Ірина Олегівна*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК*

Обсяг розрахунково-пояснювальної записки *74* сторінок

Обсяг графічної (презентаційної) частини *15* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню *Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення програмного забезпечення для розрахунку потужності для блоків живлення ПК.*

б) характеристика виконання кожного розділу дипломного проекту (роботи) *Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання проблематики розробки програмного забезпечення у Visual Studio, створені SQL-сервера, сформовано проект програми згідно теми дипломного проекту та завданню, виконано проектування основних аспектів. За допомогою відповідного програмного забезпечення реалізовані всі намічені роботи.*

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) *Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено*

г) перелік позитивних якостей дипломного проекту (роботи) *Детально описано процес виконання розробки ПЗ; Виконано проектування елементів ПЗ із поясненнями на схемах та за допомогою коду; Розроблене ПЗ виводить рекомендовану та номінальну потужність БЖ для ПК.*

д) основні недоліки дипломного проекту (роботи) *Повторне встановлення comboBox2.ValueMember без одночасного задання DataSource — дані з БД підтягуються лише один раз, а зміна ValueMember «на льоту» не гарантує коректного перепідключення. Вся бізнес-логіка (отримання даних, обчислення) лежить у код-behind форми — відсутня шарова архітектура (розділення UI, сервісів, моделей даних). Жорстко «зашиті» розміри елементів інтерфейсу та DPI-налаштування без урахування динамічного масштабування.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

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

Підпис: _____

«23» 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Маннапов Денис Хамітович

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

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

Тема дипломного проєкту: *Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 74 сторінки. У пояснювальній записці виконано опис етапів розробки програмного застосунку для розрахунку потужності БЖ, аналіз схожих проектів, збір актуальних даних апаратної частини ПК. Графічна частина складається з 15 слайдів мультимедійної презентації, які також містять графічні матеріали, передбачені технічним завданням. Якість виконання пояснювальної записки відмінна, графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Маннапов Д.Х. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Маннапов Д.Х. під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проєкту

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проєктування здобувач освіти Маннапов Д.Х. мав змогу самостійно приймати рішення з реалізації програмного засобу для розрахунку потужності БЖ, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проєкту

Шувалова Ірина Олегівна

Місце роботи і посада керівника дипломного проєкту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії

Підпис _____

«16» 06 2025 р.

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

Ми, що нижче підписалися,

Маннапов Д.Х.
здобувач освіти гр. 4КС-58, та

Шувалова І.О.,
керівник дипломного проекту,

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

«Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК» (автор роботи – Маннапов Д.Х., керівник роботи – Шувалова І.О.)

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 _____ / Маннапов Д.Х. /

Керівник _____ / Шувалова І.О. /

«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

Маннапова Дениса Хамітовича

на тему *Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК*

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

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 15.06.2025 р. значення коефіцієнту подібності в роботі становить 11,41%, коефіцієнт цитування – 3,30%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Маннапова Д.Х.* (П.І.Б.)

проведена « *16* » *червня* 20*25* 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 6/15/2025
Дата редагування 6/15/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації

Odesa Technical Professional College of Odesa National University of Technology

Заголовок

Розробка програмного забезпечення для розрахунку потужності блоків живлення у різних конфігураціях ПК

Автор Науковий керівник / Експерт

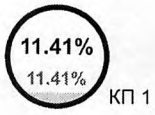
Маннапов Денис ХамітовичШувалова Ірина Олегівна

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
Довжина фрази для коефіцієнта подібності 2

11844
Кількість слів

92168
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	ß	18
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	68

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/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	100 0.84 %
2	https://card-file.ontu.edu.ua/bitstreams/c1f3e592-1123-419d-b14a-4c28662f0f1e/download	56 0.47 %
3	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	51 0.43 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	50 0.42 %
5	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	49 0.41 %

6	https://social.msdn.microsoft.com/Forums/windows/en-US/55097133-c1c1-4572-8e15-794acbc87374/datagridview-unbound-column	49 0.41 %
7	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	47 0.40 %
8	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	44 0.37 %
9	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	40 0.34 %
10	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	38 0.32 %

з домашньої бази даних (0.45 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	30 (3) 0.25 %
2	Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	12 (2) 0.10 %
3	Розробка веб-застосунку інтелектуального пошуку та сумісного перегляду аніме-контенту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	11 (2) 0.09 %

з програми обміну базами даних (0.26 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Автоматизированная система туристического агентства [redacted] 6/22/2022 Kirgiz Economic University na M. Ryskulbekov (Кафедра "Цифровая экономика и программирование")	17 (3) 0.14 %
2	ООП 2024 ІП323 Михальчук Д.Л [redacted] 12/11/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	14 (2) 0.12 %

з Інтернету (10.70 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**234 (6) 1.98 %**
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**188 (7) 1.59 %**
3	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**95 (5) 0.80 %**
4	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**94 (3) 0.79 %**
5	**https://social.msdn.microsoft.com/Forums/windows/en-US/55097133-c1c1-4572-8e15-794acbc87374/datagridview-unbound-column**	**92 (4) 0.78 %**

6	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**62 (3) 0.52 %**
7	https://poradi.ru/suspilstvo/33817-shkidlivi-virobnichi-faktori-klasifikacija.html	58 (3) 0.49 %
8	**https://card-file.ontu.edu.ua/bitstreams/c1f3e592-1123-419d-b14a-4c28662f0f1e/download**	**56 (1) 0.47 %**
9	**https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download**	**54 (3) 0.46 %**
10	**http://8ref.com/2/referat_28518.html**	**52 (4) 0.44 %**
11	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**44 (1) 0.37 %**
12	**http://www.um.co.ua/14/14-9/14-92598.html**	**30 (1) 0.25 %**
13	https://dspace.nuft.edu.ua/bitstreams/12558f74-327a-4347-9084-ed509d590d95/download	27 (4) 0.23 %
14	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	25 (2) 0.21 %
15	https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf	24 (1) 0.20 %
16	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	22 (2) 0.19 %
17	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	21 (1) 0.18 %
18	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	20 (1) 0.17 %
19	https://lektsia.com/1x54.html	14 (1) 0.12 %
20	https://www-cgrl.cs.mcgill.ca/~godfried/publications/banff.pdf	14 (1) 0.12 %
21	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	9 (1) 0.08 %
22	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	9 (1) 0.08 %
23	http://ukrefs.com.ua/print:page,1,171772-Razrabotka-i-issledovanie-imitacionnoiy-modeli-lokal-nyh-vychislitel-nyh-seteiy.html	7 (1) 0.06 %
24	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	6 (1) 0.05 %
25	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	5 (1) 0.04 %
26	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	5 (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. 58.14.000.ДП

МАННАПОВА
ДЕНИСА ХАМІТОВИЧА

м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»