

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ УПРАВЛІННЯ
ПОРТФЕЛЯМИ ПРОЄКТІВ ЕНЕРГОЗБЕРЕЖЕННЯ НА
МЕТАЛУРГІЙНИХ ПІДПРИЄМСТВАХ

SOFTWARE IMPLEMENTATION OF METHODS FOR
MANAGING PORTFOLIOS OF ENERGY SAVING PROJECTS
AT METAL WORKS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-210

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

АЛЕКСЄЄВ О.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ДУБРОВІН В.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент КАЗУРОВА А.Є.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

АЛЕКСЄЄВА Олександра Олександровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація методів управління портфелями проєктів енергозбереження на металургійних підприємствах. Software Implementation of Methods for Managing Portfolios of Energy Saving Projects at Metal Works

керівник проєкту (роботи) к.т.н., професор ДУБРОВІН Валерій Іванович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань роботи.

2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування і експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ДУБРОВІН В.І., професор		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “15” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олександр АЛЕКСЄЄВ
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Валерій ДУБРОВІН
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
69 с., 6 табл., 40 рис., 2 дод., 11 джерел.

ПОРТФЕЛІ ПРОЄКТІВ, ЕНЕРГОЗБЕРЕЖЕННЯ, ПІДПРИЄМСТВО,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МЕТАЛУРГІЯ.

Об'єктом дослідження є процес енергоспоживання та енергозбереження на металургійних підприємствах.

Предмет дослідження – комп'ютерні програми для управління портфелями проєктів енергозбереження на металургійних підприємствах.

Мета роботи – створення програмного забезпечення для управління проєктами енергозбереження на металургійних підприємствах.

Матеріали, методи та технічні засоби: мова програмування Python, IDE PyCharm Community Edition, персональний комп'ютер з процесором AMD Phenom II X4 965 під управлінням операційної системи Microsoft Windows 7.

Результати. Створено програмний застосунок, який дозволяє виділити найпріоритетніші проєкти енергозбереження на металургійному виробництві і, на основі отриманих даних, зменшити кількість енергоспоживання.

Галузь використання – управління проєктами енергозбереження на металургійних підприємствах.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 69 pages, 6 tables, 40 figures, 2 appendixes, 11 sources.

PROJECT PORTFOLIOS, ENERGY SAVING, ENTERPRISE, SOFTWARE, METALLURGY.

The object of research is a process of energy consumption and energy saving at metallurgical enterprises.

The subject of the research is computer programs for managing portfolios of energy saving projects at metallurgical enterprises.

The purpose of the work is to create software for managing energy saving projects at metallurgical enterprises.

Materials, methods, and tools: Python programming language, PyCharm Community Edition IDE. a personal computer with an AMD Phenom II X4 965 processor running Microsoft Windows 7 operating system.

Results. A software application has been created to identify the highest-priority energy saving projects in metallurgical production and, based on the data obtained, reduce energy consumption.

The field of use is a management of energy saving projects at metallurgical enterprises.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз проблеми та постановка завдань роботи 10	10
1.1 Аналіз предметної області 10	10
1.2 Огляд існуючих аналогів 10	10
1.2.1 Microsoft project 10	10
1.2.2 Oracle primavera 11	11
1.2.3 Planview projectplace.....13	13
1.3 Постановка завдання роботи 14	14
2 Розробка архітектури програми 15	15
2.1 Вибір мови програмування.....15	15
2.2 Визначення функціональних вимог до застосунку 16	16
2.3 Вибір бази даних.....18	18
3 Основні рішення щодо реалізації компонентів системи 20	20
3.1 Алгоритм роботи програми 20	20
3.2 Опис бази даних.....21	21
3.3 Опис алгоритмів роботи програми 25	25
3.3.1 Реєстрація та авторизація 25	25
3.3.2 Додавання процесів 25	25
3.3.3 Змінення цін 26	26
3.3.4 Додавання проєкту 26	26
3.3.5 Сорткування проєктів 26	26

3.3.6 Запуск проєкту 27

3.3.7 Видалення проєктів 27

4 Експлуатація, тестування та експериментальне дослідження програми.....28

4.1 Призначення застосунку та умови його виконання 28

4.2 Виконання роботи із застосунком 28

4.3 Тестування застосунку та його результати 39

Висновки.....40

Перелік джерел посилання 41

Додаток А Текст програми 42

Додаток Б Слайди презентації.....63

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE – Integrated Development Environment;

SQL – Structured Query Language;

ОС – Операційна система;

ПЗ – Програмний застосунок;

РБД – Реляційна база даних;

СУБД – Система управління базами даних.

ВСТУП

В умовах сучасної економіки питання енергозбереження стають дедалі актуальнішими для промислових підприємств, особливо в галузі металургії, яка є однією з найбільш енерговитратних.

Металургійна галузь займає близько 30–35% у загальному споживанні всієї електроенергії і 25–35% у споживанні теплоенергії в промисловості [1]. Ці величезні витрати мають тенденцію на зростання через низку факторів, наприклад: зношування обладнання складність інтеграції нових технологій зростання цін на ресурси.

Таке складне становище галузі змушує вдаватися до запровадження проєктів економії енергоресурсів і оптимізації виробництва. Враховуючи масштаби підприємств, виникає необхідність у засобах для більш комфортного керування цими процесами.

Впровадження інноваційних технологій, оптимізація виробничих процесів та використання альтернативних джерел енергії є ключовими напрямками, які можуть дозволити металургійним підприємствам досягти значних успіхів у сфері енергозбереження.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ РОБОТИ

1.1 Аналіз предметної області

Управління проєктами — це комплексний процес, який включає планування, організацію, контроль і завершення проєктів. Враховуючи масштаб і складність металургійного виробництва, успішне керування проєктами потребує чіткого планування, координації та контролю на всіх етапах — від розробки концепції до завершення та введення в експлуатацію. Проєктний підхід дозволяє підвищити ефективність використання ресурсів, знизити ризики та забезпечити досягнення поставлених цілей в рамках визначених термінів і бюджету.

Основними аспектами управління проєктами в металургії є планування, контроль та оптимізація ресурсів, а також забезпечення безпеки та відповідності екологічним стандартам.

Успішне управління проєктами у металургійній галузі сприяє підвищенню продуктивності, зниженню витрат та покращенню якості продукції, що в свою чергу допомагає підприємствам залишатися конкурентоспроможними на світовому ринку.

1.2 Огляд існуючих аналогів

1.2.1 Microsoft Project

Microsoft Project - це програмне забезпечення для управління проєктами, яке використовується для створення графіків, планів проєктів, управління ресурсами та відстеження часу. Він має такі функції, як діаграми Ганта, дошки канбан і календарі проєктів для професіоналів з управління проєктами [2].

На рис. 1.1 зображено загальний вигляд застосунку.

Переваги:

- розширені можливості планування;
- гнучкість і налаштування;

- робота з великими проектами.

Недоліки:

- висока вартість;
- складність освоєння.

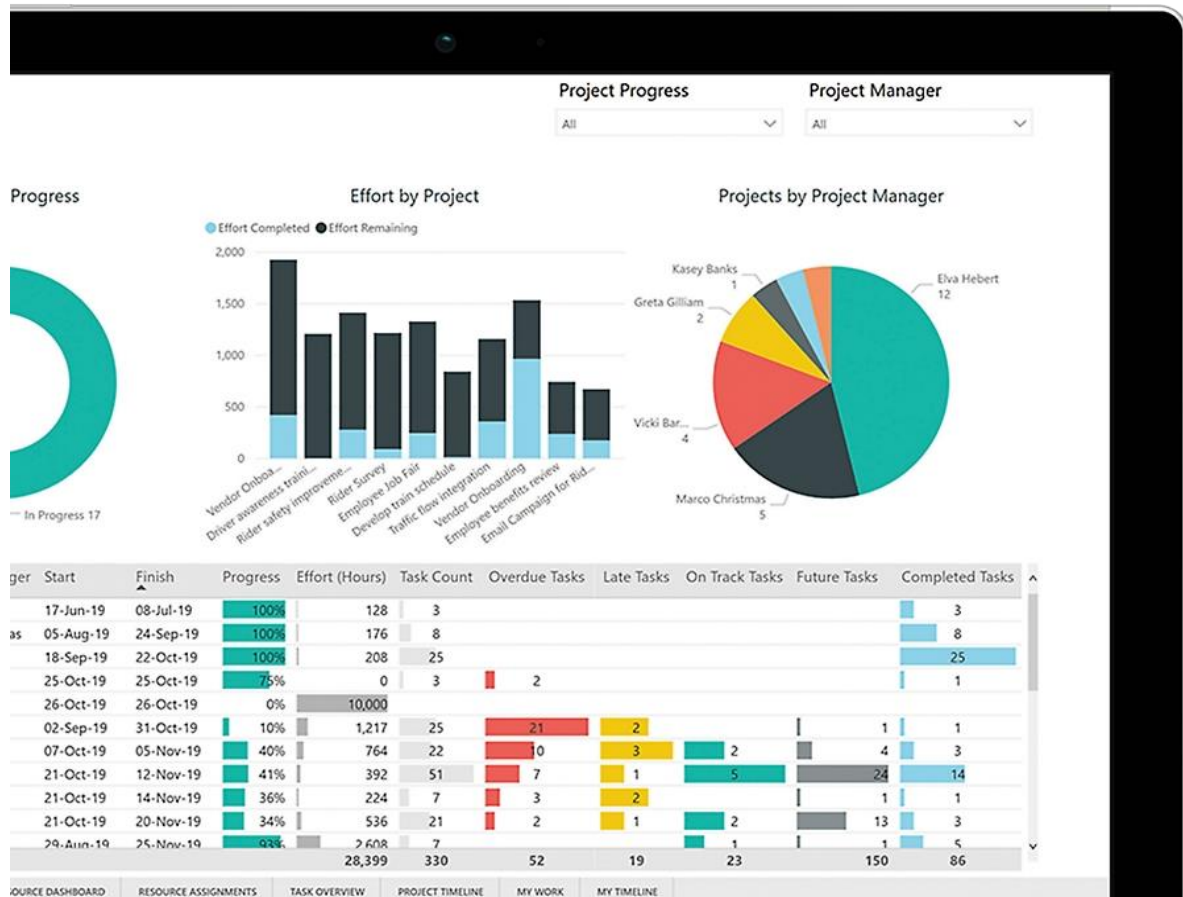


Рисунок 1.1 – Застосунок «Microsoft Project» [3]

1.2.2 Oracle Primavera

Primavera - це програмне забезпечення для управління портфелем проектів підприємства. Воно включає управління проектами, планування, аналіз ризиків, управління можливостями, управління ресурсами, спільну роботу та можливості контролю. Primavera була запущена в 1983 році компанією Primavera Systems Inc., яка була придбана корпорацією Oracle в 2008 році [4].

На рис. 1.2 зображено загальний вигляд застосунку.

Переваги:

- можливості для управління ризиками;
- підтримка багатокористувацької роботи;
- висока гнучкість налаштувань.

Недоліки:

- висока вартість;
- складність освоєння;
- високі вимоги до апаратного забезпечення.

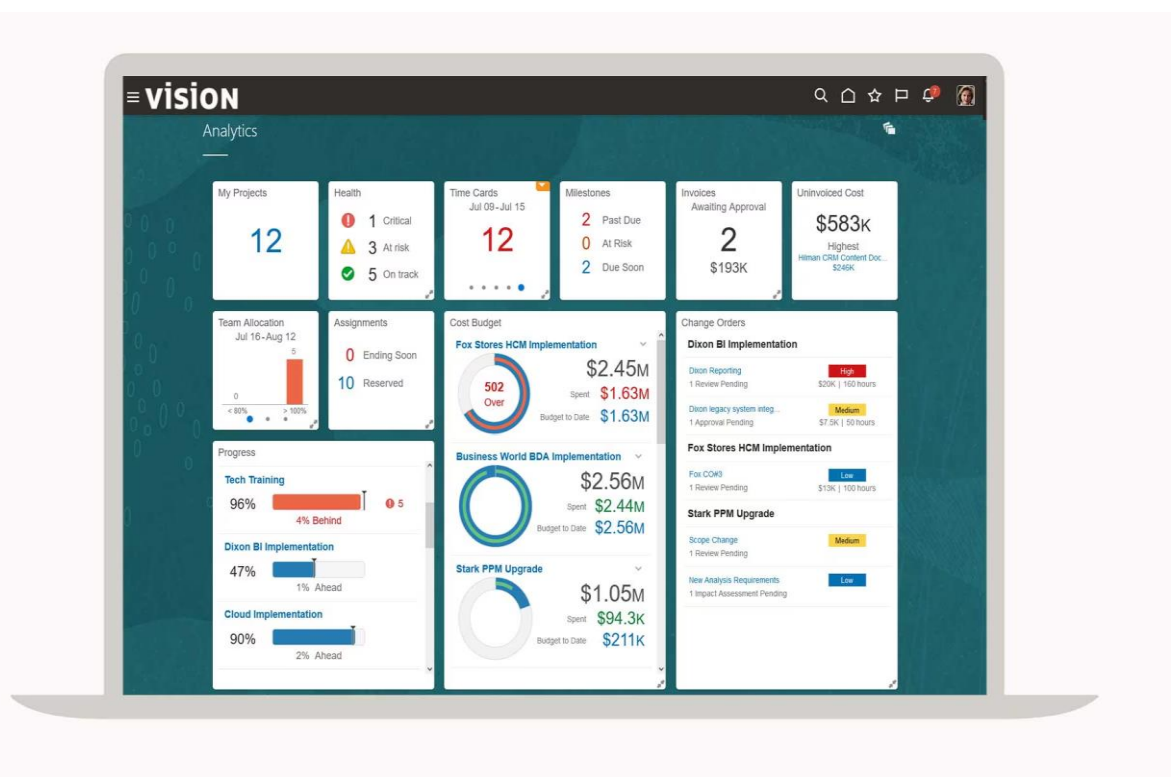


Рисунок 1.2 – Застосунок «Oracle Primavera» [5]

1.2.3 Planview Projectplace

Projectplace - це програмний застосунок для управління спільною роботою від Planview, яке пропонує програмне забезпечення для управління проектами та рішення для спільної роботи. Projectplace інтегрує в собі такі функції, як програми для управління завданнями, дошки Kanban та діаграми Ганта [6]. На рис. 1.3 зображено загальний вигляд застосунку.

Переваги:

- зручний інтерфейс та простота використання;
- командна робота в режимі реального часу;
- наявність хмарної платформи.

Недоліки:

- вартість;
- обмежені можливості для великих проектів;
- необхідність постійного інтернет-з'єднання.

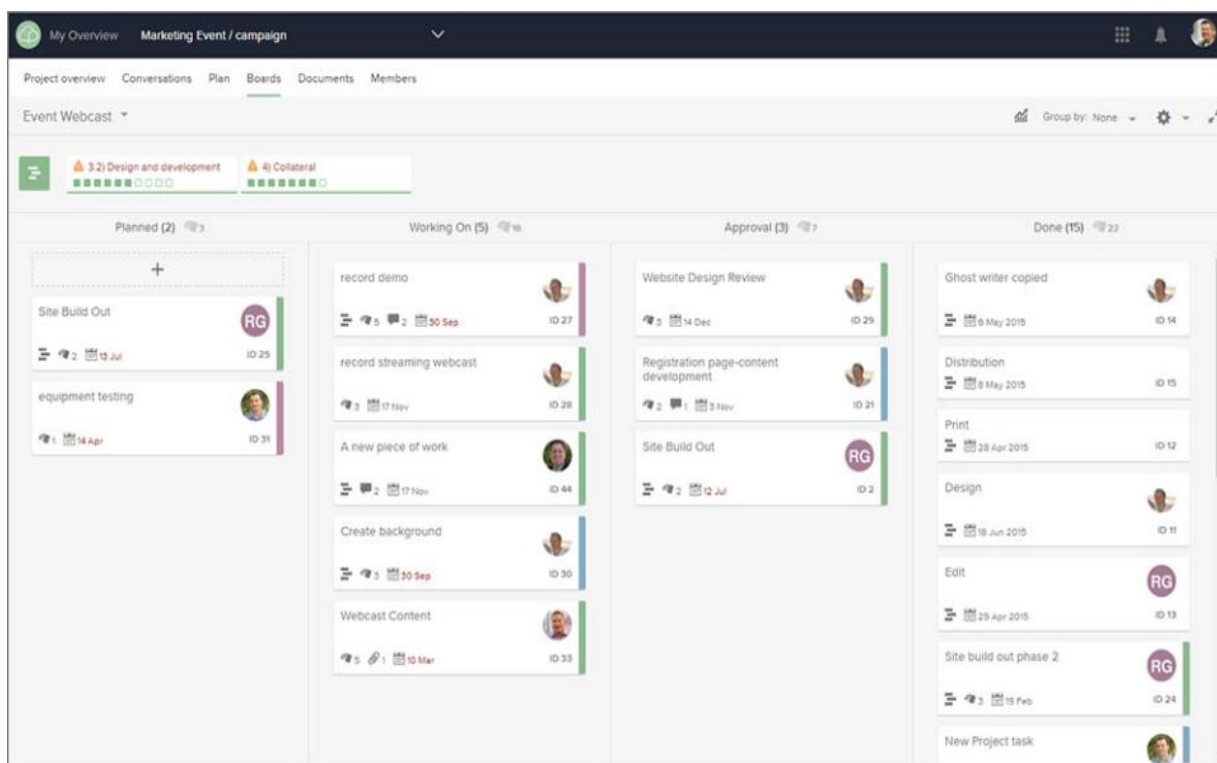


Рисунок 1.3 – Застосунок «Planview Projectplace» [7]

1.3 Постановка завдання роботи

За результатами огляду існуючих аналогів, завдання до роботи – це розробка застосунку для управління проєктами енергозбереження.

Застосунок повинен забезпечувати виконання наступного функціоналу:

- додавання нових процесів;
- перегляд детальної інформації про процеси;
- перегляд інформації про проєкти;
- додавання нових проєктів;
- відстежування ефективності проєктів;
- керування проєктами.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір мови програмування

Python (найчастіше вживане прочитання — «Пайтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, аспектно-орієнтована та функціональна.

Розробка мови Python була розпочата в кінці 1980-х років співробітником голландського інституту CWI Гвідо ван Россумом. Для розподіленої ОС Амоса потрібна була розширювана скриптова мова, і Гвідо почав писати Python на дозвіллі, запозичивши деякі напрацювання для мови АВС (Гвідо брав участь у розробці цієї мови, орієнтованої на навчання програмування). У лютому 1991 року Гвідо опублікував вихідний текст в групі новин alt.sources. Мова почала вільно поширюватися через Інтернет і сподобалася іншим програмістам. З 1991 року Python є цілком об'єктно-орієнтованим. Python також запозичив багато рис таких мов, як С, С++, і Ісрп, й окремі риси функціонального програмування з Ліспу [8].

У табл. 2.1 наведено порівняння мови програмування Python з іншими мовами програмування.

Таблиця 2.1 — Порівняння Python з іншими мовами програмування

Критерій порівняння	Python	Java	C++
Продуктивність	Висока	Висока	Середня
Складність розробки	Низька	Середня	Висока
Безпека та надійність	Висока	Висока	Низька
Підтримка спільноти	Висока	Висока	Середня

2.2 Визначення функціональних вимог до застосунку

Функціональні вимоги до застосунку:

- користувач повинен мати можливість реєструватися та авторизуватися;
- користувач повинен мати можливість додавати нові процеси та пристрої;
- користувач повинен мати можливість змінювати ціни на енергоресурси;
- користувач повинен мати можливість переглядати дані про наявні енерговитратні процеси;
- користувач повинен мати можливість видаляти інформацію про наявні енерговитратні процеси;
- користувач повинен мати можливість додавати варіанти проєктів енергозбереження;
- користувач повинен мати можливість видаляти варіанти проєктів енергозбереження;

- користувач повинен мати можливість сортувати проекти енергозбереження за їх ефективністю;
- користувач повинен мати можливість застосовувати проекти до енерговитратних процесів на виробництві;
- користувач повинен мати можливість переглядати варіанти проектів енергозбереження.

На основі вище зазначених функціональних вимог було побудовано діаграму прецедентів. Діаграма прецедентів наведена на рис. 2.1.

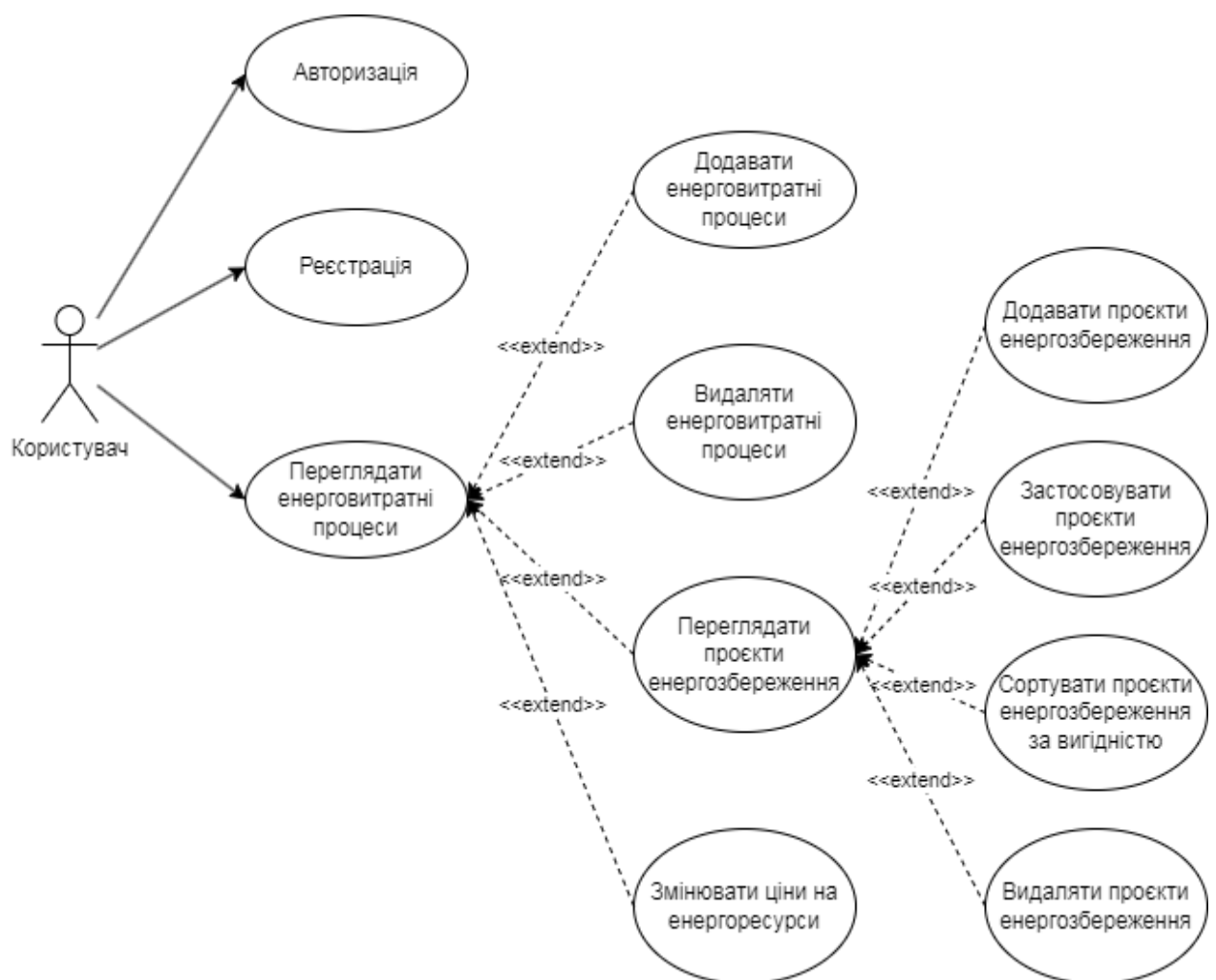


Рисунок 2.1 — Діаграма прецедентів

2.3 Вибір бази даних

Одними з найпопулярніших типів баз даних є реляційна та нереляційна.

Реляційні бази даних (РБД) – один із найпоширеніших типів баз даних, що використовуються для зберігання, організації та управління інформацією. Засновані на реляційній моделі даних, яку було запропоновано Едгаром Коддом у 1970 році. Швидкий доступ і управління даними реалізуються за допомогою SQL-запитів. Це робить реляційні бази даних ідеальними для широкого спектра додатків, від обліку та управління замовленнями до зберігання інформації про клієнтів і багато чого іншого [9].

NoSQL database (або «Not Only SQL») — це група баз даних, які відрізняються від традиційних реляційних баз даних (SQL) способом організації, зберігання та доступом до даних. NoSQL бази даних зазвичай потрібні для зберігання великих обсягів неструктурованих або слабоструктурованих даних. Це, наприклад, текстові файли, зображення та відео [10].

У табл. 2.2 наведена порівняльна характеристика між цими двома видами бази даних.

Таблиця 2.2 – Порівняльна характеристика реляційної та нереляційної бази даних

Опис	Реляційна база даних	Нереляційна база даних
Структура даних	Дані представлені у вигляді таблиць	Дані можуть бути представлені у вигляді документів, графів та тощо
Гнучкість	Менш гнучка	Більш гнучка
Нормалізація	Потребує	Не потребує
Масштабованість	Менш масштабована	Більш масштабована

В якості бази даних було обрано SQL через відсутність необхідності у зберіганні великих обсягів неструктурованих або слабоструктурованих даних.

В якості СУБД було обрано SQLite – бібліотеку, яка надає легку дискову базу даних, яка не потребує окремого серверного процесу та дозволяє отримувати доступ до бази даних за допомогою нестандартного варіанту мови запитів SQL. Деякі програми можуть використовувати SQLite для внутрішнього зберігання даних [11].

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм роботи програми

На рисунку 3.1 зображено схему роботи застосунку

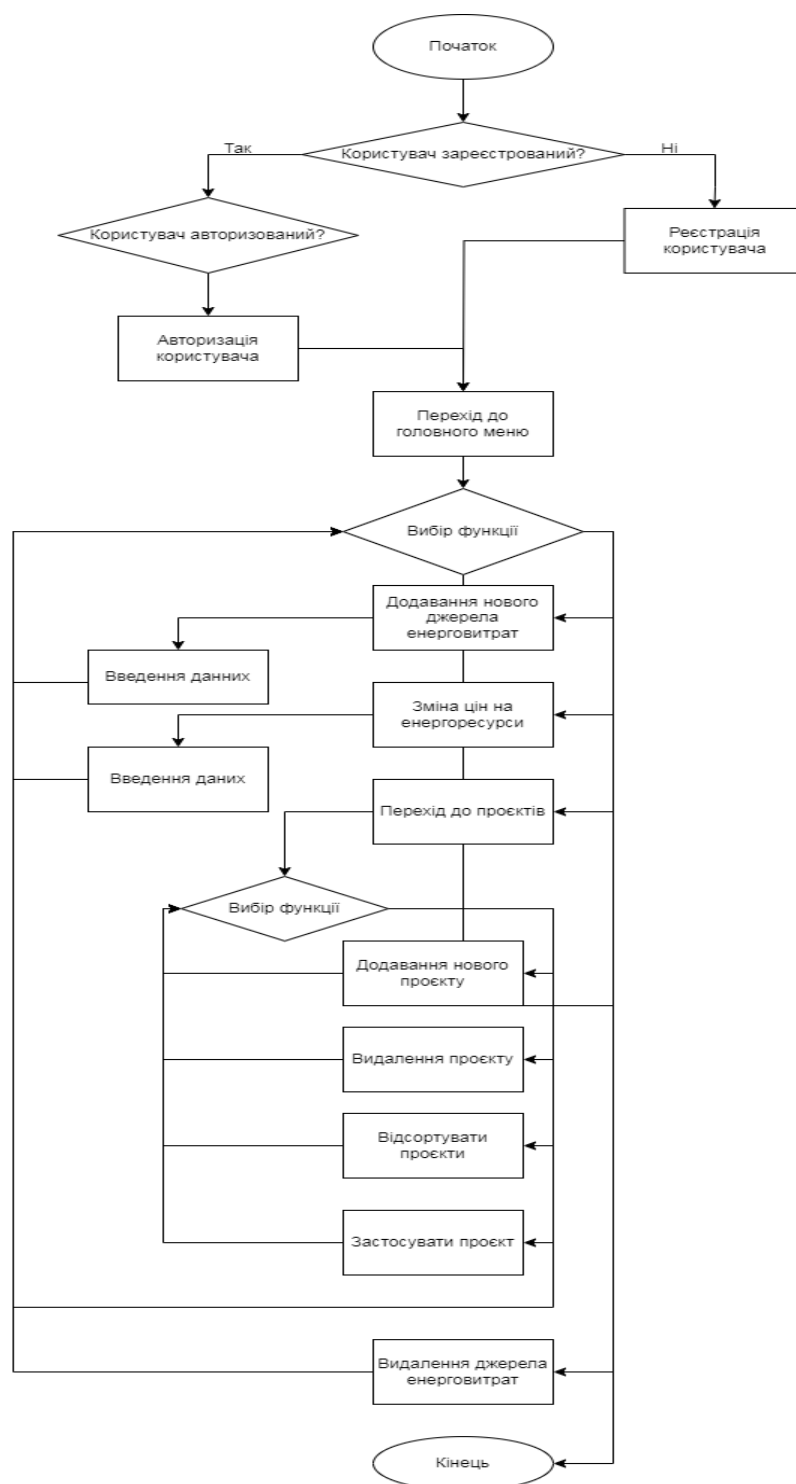


Рисунок 3.1 – Схема роботи застосунку

3.2 Опис бази даних

База даних складається з таких таблиць:

- users;
- user_data;
- user_rates;
- projects.

Таблиця «users» зберігає інформацію про користувача, таку як його номер, логін і пароль. Поля та їхній тип наведено у табл. 3.1.

Таблиця 3.1 – Таблиця «users»

Поле	Тип
user_id	int
username	text
password	text

Таблиця «user_data» зберігає інформацію про назву процесів та пристроїв що витрачають такі енергоресурси, як природний газ, теплова енергія і електроенергія та в яких кількостях. Поля та їхній тип наведено у табл. 3.2.

Таблиця 3.2 – Таблиця «user_data»

Поле	Тип
id	int
user_id	int
process_or_device	text
natural_gas	real
solid_energy	real
electrical_energy	real
liquid_energy	real

Таблиця «user_rates» зберігає інформацію про ціни на такі ресурси, як природний газ, теплова енергія і електроенергія. Поля та їхній тип наведено у табл. 3.3.

Таблиця 3.3 – Таблиця «user_rates»

Поле	Тип
id	int
user_id	int
natural_gas	real
solid_energy	real
electrical_energy	real
liquid_energy	real

Таблиця «projects» зберігає інформацію про назву проєкту, його вартість, обсяги зменшення споживання енергоресурсів, перелік процесів і пристроїв на які впливатиме проєкт, кількість зменшення грошових витрат, кількість місяців, за яку проєкт окупиться і коефіцієнт окупності. Поля та їхній тип наведено у табл. 3.4.

Таблиця 3.4 – Таблиця «projects»

Поля	Тип
1	2
id	int
user_id	int
project_name	text
project_cost	real
gas_reduction	real

Кінець 3.4

1	2
solid_energy_reduction	real
electrical_energy_reduction	real
liquid_energy_reduction	real
processess	text
cost_reduction_per_month	real
payback_period	real
coefficient	real

На рис. 3.2 зображено схему бази даних і зв'язки між таблицями.

Зв'язки таблиці «users» з таблицями «user_data» і «projects» один до багатьох тому, що на одному підприємстві відбувається багато процесів і розглядається декілька проєктів.

Зв'язок таблиці «users» з таблицею «user_rates» один до одного тому, що застосовується тільки один варіант цінових показників.

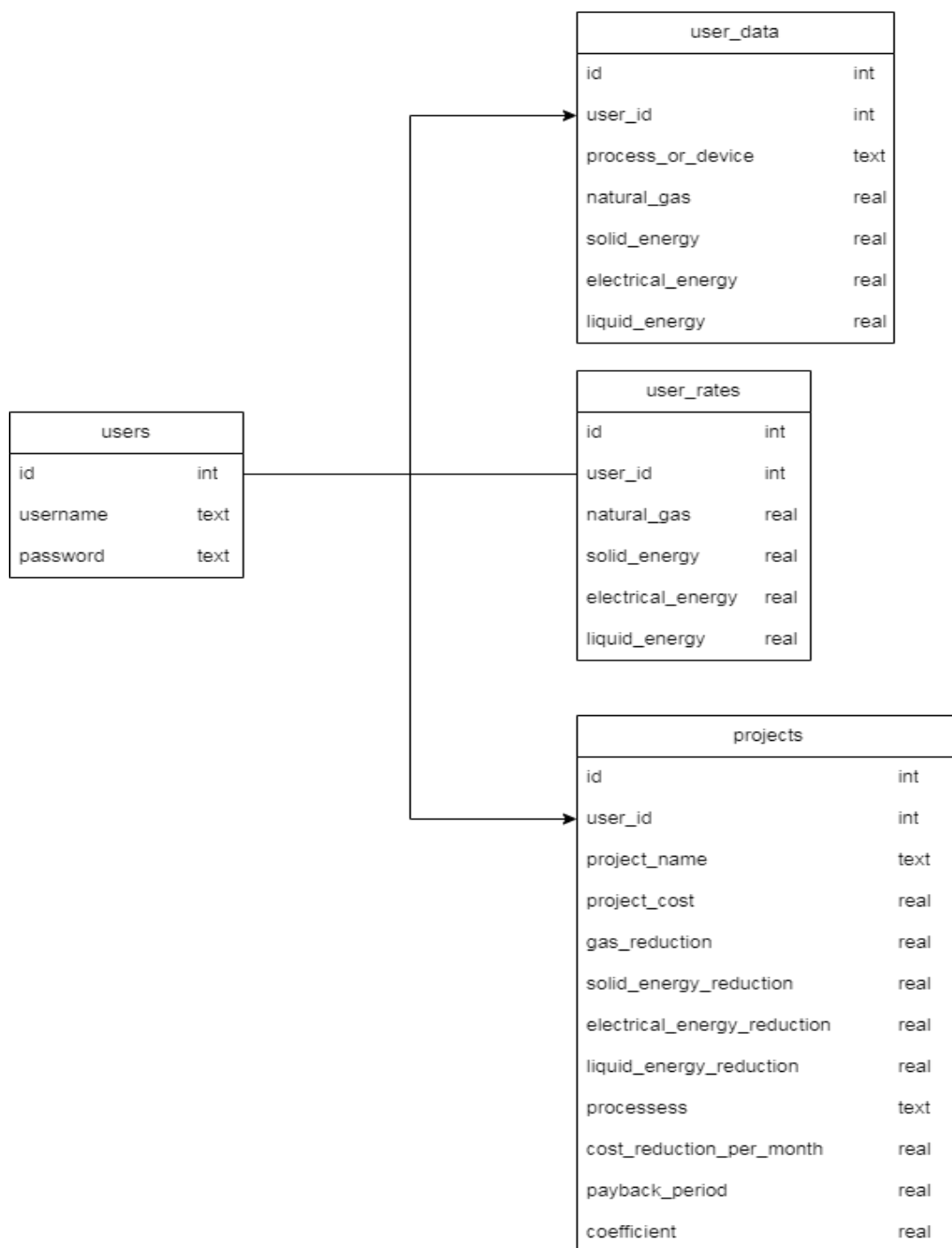


Рисунок 3.2 – Схема бази даних

3.3 Опис алгоритмів роботи програми

3.3.1 Реєстрація та авторизація

1. Користувач вводить назву підприємства і пароль. Якщо користувач обирає авторизацію – відбувається перехід до пункту 2, а якщо реєстрацію – до пункту 3.

2. Відбувається звернення до бази даних з метою перевірки наявності у ній користувача. Якщо користувач присутній – відбувається перехід до пункту 6, якщо ні – до пункту 4.

3. Відбувається звернення до бази даних з метою перевірки у ній користувача. Якщо користувач присутній – відбувається перехід до пункту 4, якщо ні – до пункту 5.

4. Користувач отримує повідомлення про відповідну помилку.

5. Дані користувача заносяться до бази даних. Перехід до кроку 6.

6. Відкривається головне меню.

3.3.2 Додавання процесів

1. Користувач натискає кнопку «Додати процес/пристрій».

2. Відкривається вікно «add_process_window».

3. Користувач вводить дані в поля і натискає кнопку «Зберегти». Якщо користувач залишає поле «Процес/пристрій» порожнім або вводить в інші поля не числові значення – перехід до пункту 8, якщо ні – до пункту 4.

4. Дані заносяться до бази даних.

5. Новий запис заноситься до таблиці у вікні.

6. Оновлюються дані про витрати.

7. Вікно «add_process_window» закривається.

8. Користувач отримує повідомлення про відповідну помилку.

3.3.3 Змінення цін

1. Користувач натискає кнопку «Змінити ціни».
2. Відкривається вікно «price_window».
3. Користувач заповнює поля і натискає кнопку «Зберегти». Якщо користувач заповнює їх числами – перехід до пункту 4, якщо ні – до пункту 5.
4. Дані в таблиці «user_rates» замінюються на нові, вікно «price_window» закривається.
5. Користувач отримує повідомлення про помилку.

3.3.4 Додавання проєкту

1. Користувач натискає кнопку «Проекти».
2. Відкривається вікно «project_window».
3. Користувач вводить дані у поля і натискає кнопку «Зберегти». Якщо користувач заповнює поля «Назва проєкту» і «Процеси/пристрої» – перехід до пункту 5, якщо ні – до пункту 4.
4. Користувач отримує повідомлення про відповідну помилку.
5. Вираховуються значення збереження коштів, періоду окупності і коефіцієнт окупності.
6. Дані заносяться до таблиці «projects».
7. Новий запис заноситься до таблиці у вікні «project_window».

3.3.5 Сортування проєктів

1. Користувач натискає кнопку «Сортувати за пріоритетністю».
2. Відбувається звернення до бази даних, таблиці «projects».
3. Відбувається сортування елементів таблиці за коефіцієнтом окупності (від більшого до меншого).
4. Таблиця у вікні оновлюється.

3.3.6 Запуск проєкту

1. Користувач обирає проєкт в таблиці.
2. Користувач натискає кнопку «Запустити проєкт».
3. Дані про енергозбереження обраного проєкту застосовуються до відповідних процесів.
4. Зміни заносяться до бази даних, таблиці «user_data».
5. Оновлюється таблиця у вікні «main_window».
6. Оновлюються дані про витрати.

3.3.7 Видалення проєктів

1. Користувач обирає проєкт.
2. Користувач натискає кнопку «Видалити вибраний проєкт».
3. Обраний проєкт видаляється з бази даних.
4. Таблиця у вікні «project_window» оновлюється.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення застосунку та умови його виконання

Застосунок призначений для того, щоб користувачі могли додавати проекти енергозбереження для металургійних підприємств і досліджувати їх вплив на обсяги енергоспоживання і витрати на нього.

Для забезпечення нормальної роботи застосунку необхідний персональний комп'ютер з операційною системою Windows.

4.2 Виконання роботи із застосунком

Для того щоб запустити застосунок користувач повинен запустити файл «Управління проектами енергозбереження». Після цього відобразиться вікно авторизації та реєстрації (рис. 4.1).

Користувачу необхідно ввести назву підприємства і пароль, після чого натиснути або на кнопку «Реєстрація», або на кнопку «Увійти». Потім, якщо користувач успішно зареєструвався або авторизувався, відкривається головне меню (рис. 4.2).

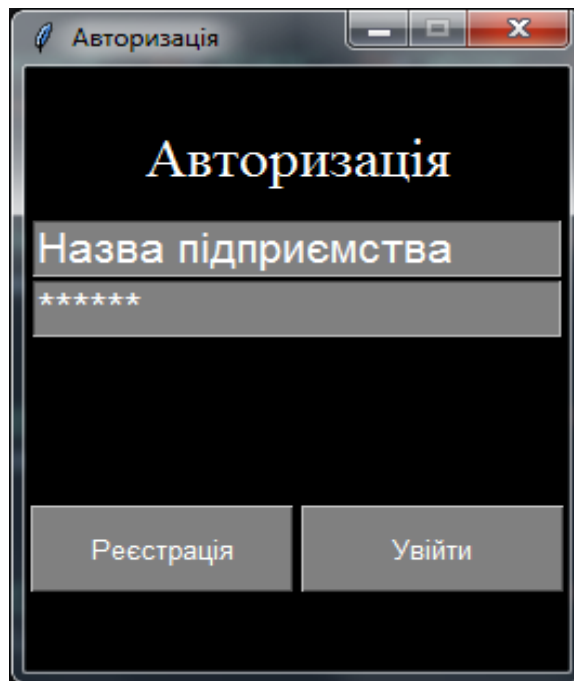


Рисунок 4.1 — Вікно реєстрації і авторизації

ПАТ "Запоріжсталь"				
Процес або пристрій	Природний газ, куб.м	Тверде паливо, т	Електроенергія, кВт*год	Рідке паливо, л
Мартенівська піч 1	30000.0	150.0	0.0	0.0
Транспортування	0.0	0.0	0.0	400.0
Освітлення	0.0	0.0	100.0	0.0
Додати процес/пристрій Змінити ціни Проекти Видалити вибраний процес/пристрій Ціни: Природний газ: 8.0 грн за 1 куб.м Тверде паливо: 16800.0 грн за 1 т Електроенергія: 4.3 грн за 1 кВт*год Рідке паливо: 4.3 грн за 1 л Всього: 30000.0 куб.м газу (240000.0грн) 150.0 т твердого пального (2520000.0грн) 100.0 кВт*год електроенергії (430.0грн) 400.0 л рідкого пального (24000.0грн) Сума: 2784430.0грн				

Рисунок 4.2 — Головне вікно застосунку

Якщо користувач спробував зареєструватися або авторизуватися не заповнивши необхідні поля, намагався авторизуватися, не зареєструвавшись, або зареєструватися, використовуючи дані вже існуючого користувача — виводиться повідомлення про відповідну помилку (рис. 4.3 – 4.5).

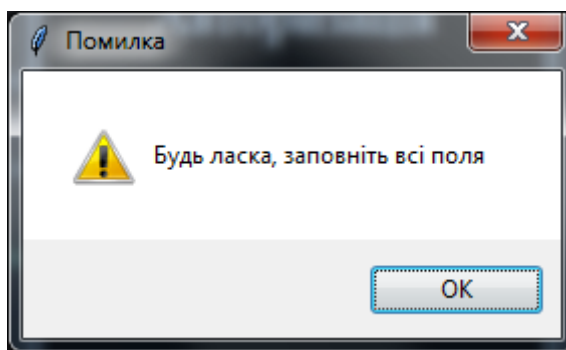


Рисунок 4.3 — Повідомлення про незаповнені поля

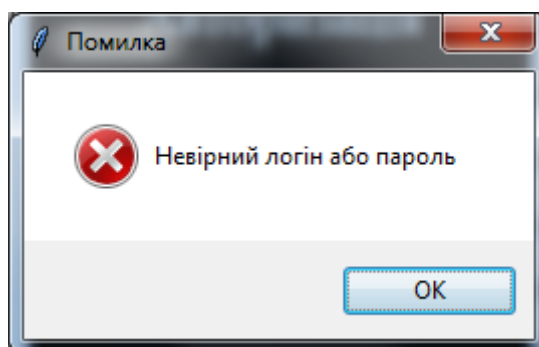


Рисунок 4.4 — Повідомлення про неправильно введені дані

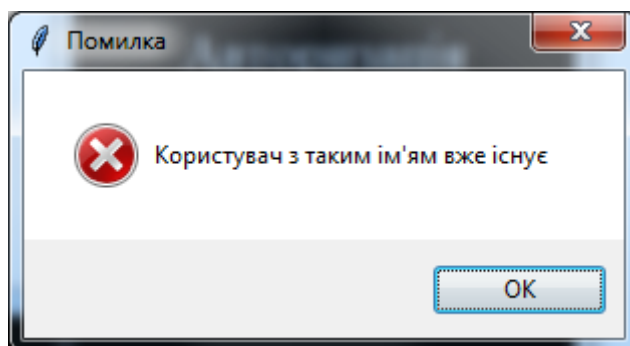


Рисунок 4.5 — Повідомлення про невдалу спробу реєстрації

У головному меню користувачеві доступні такі функції, як «Додати процес/пристрій», «Змінити ціни», «Проекти» і «Видалити вибраний процес/пристрій». Внизу відображаються ціни на енергоресурси і обсяги їх споживання.

При натисканні кнопки «Додати процес/пристрій» відкривається вікно для додавання нового джерела витрат енергоресурсів (рис. 4.6). В ньому

користувач заповнює заповнює необхідні поля і натискає кнопку «Зберегти», після чого нові дані заносяться до таблиці.

Якщо користувач не заповнив поле «Процес/пристрій», він отримує повідомлення про помилку (рис. 4.7). Якщо користувач заповнив поля для запису витрат енергоресурсів не числовими значеннями, він отримує повідомлення про відповідну помилку (рис. 4.8). При успішному введенні даних до таблиці додається новий запис (рис. 4.9).

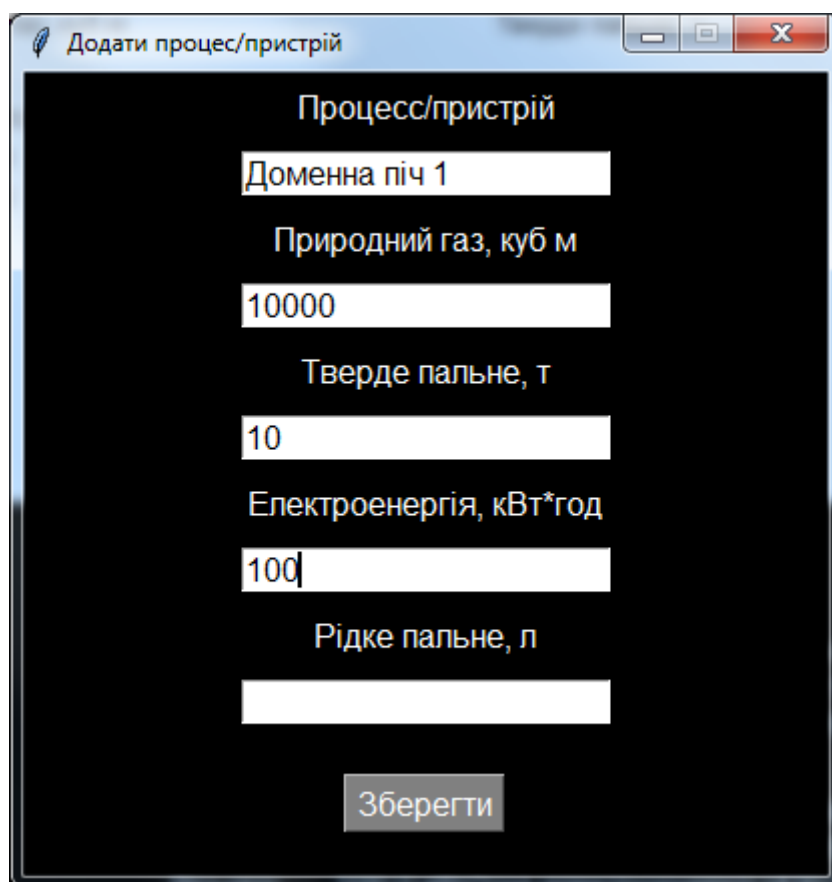


Рисунок 4.6 — Вікно додавання нового процесу

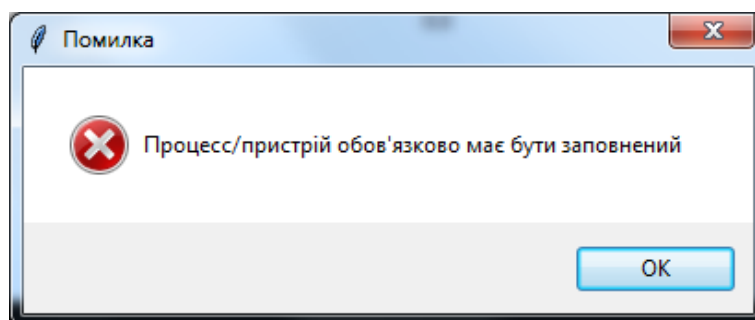


Рисунок 4.7 — Повідомлення про відсутність вводу назви процесу

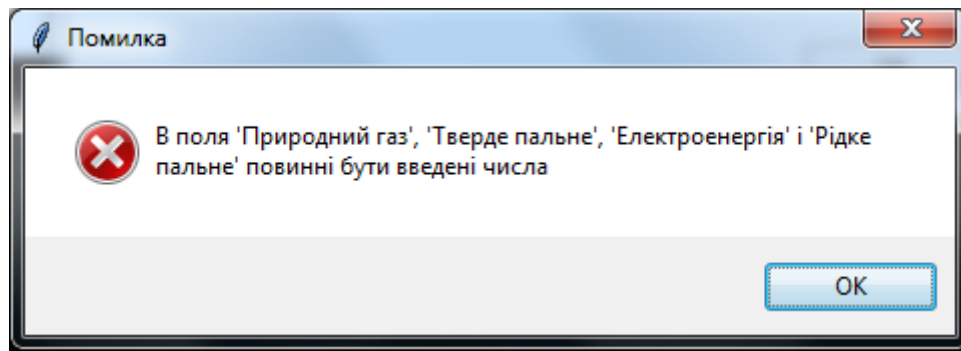


Рисунок 4.8 — Повідомлення про заповнення полів не числами

Головне вікно

ПАТ "Запоріжсталь"

Процес або пристрій	Природний газ, куб.м	Тверде паливо, т	Електроенергія, кВт*год	Рідке паливо, л
Мартенівська піч 1	30000.0	150.0	0.0	0.0
Транспортування	0.0	0.0	0.0	400.0
Освітлення	0.0	0.0	100.0	0.0
Доменна піч 1	10000.0	10.0	100.0	0.0

Додати процес/пристрій Змінити ціни Проекти Видалити вибраний процес/пристрій

Ціни:

Природний газ: 8.0 грн за 1 куб.м

Тверде паливо: 16800.0 грн за 1 т

Електроенергія: 4.3 грн за 1 кВт*год

Рідке паливо: 60.0 грн за 1 л

Всього:

40000.0 куб.м газу (320000.0грн)

160.0 т твердого пального (2688000.0грн)

200.0 кВт*год електроенергії (860.0грн)

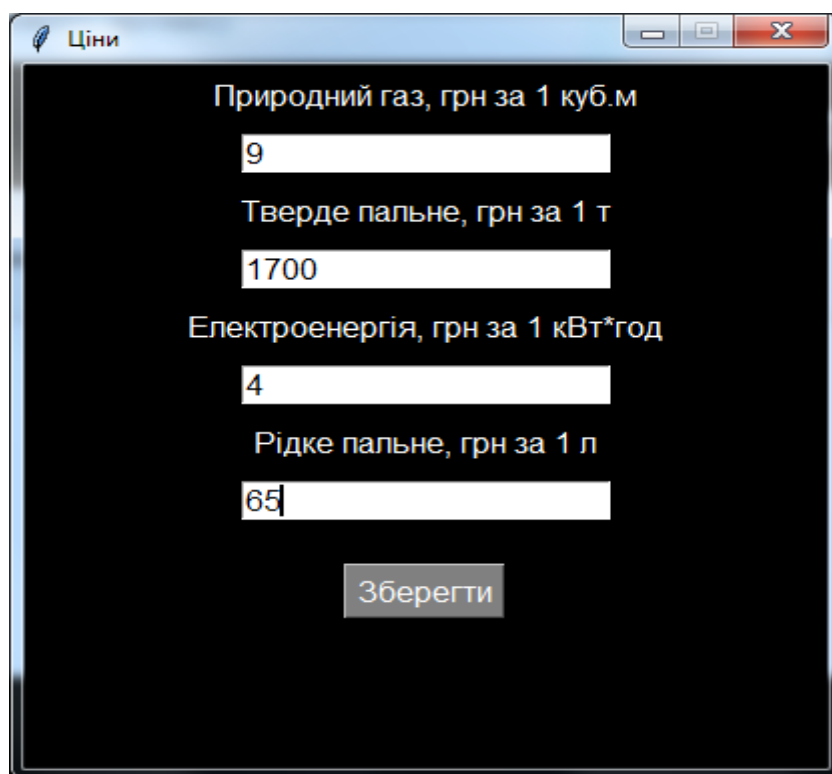
400.0 л рідкого пального (24000.0грн)

Сума: 3032860.0грн

Рисунок 4.9 — Додавання нового запису до таблиці

При натисканні кнопки «Змінити ціни» виводиться вікно для введення цін на енергоресурси (рис. 4.10). У ньому користувач заповнює поля числовими даними і натискає кнопку «Зберегти».

Якщо користувач заповнив поля не числовими даними, виводиться повідомлення про відповідну помилку (рис. 4.11).



Ціни

Природний газ, грн за 1 куб.м
9

Тверде пальне, грн за 1 т
1700

Електроенергія, грн за 1 кВт*год
4

Рідке пальне, грн за 1 л
65

Зберегти

Рисунок 4.10 — Вікно зміни цін

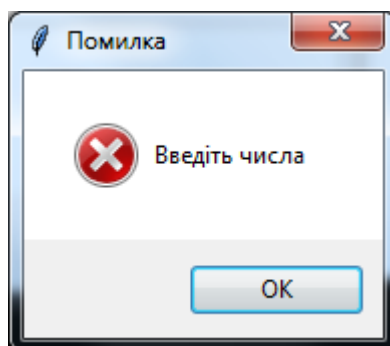


Рисунок 4.11 — Повідомлення про некоректне введення даних

При успішному введенні цін змінюється відображення цін і витрат на головному вікні (рис. 4.12).

Головне вікно

ПАТ "Запоріжсталь"

Процес або пристрій	Природний газ, куб.м	Тверде паливо, т	Електроенергія, кВт*год	Рідке паливо, л
Мартенівська піч 1	30000.0	150.0	0.0	0.0
Транспортування	0.0	0.0	0.0	400.0
Освітлення	0.0	0.0	100.0	0.0
Доменна піч 1	10000.0	10.0	100.0	0.0

Додати процес/пристрій Змінити ціни Проекти Видалити вибраний процес/пристрій

Ціни:
 Природний газ: 9.0 грн за 1 куб.м
 Тверде паливо: 1700.0 грн за 1 т
 Електроенергія: 4.0 грн за 1 кВт*год
 Рідке паливо: 4.0 грн за 1 л

Всього:
 40000.0 куб.м газу (360000.0грн)
 160.0 т твердого пального (272000.0грн)
 200.0 кВт*год електроенергії (800.0грн)
 400.0 л рідкого пального (26000.0грн)
 Сума: 658800.0грн

Рисунок 4.12 — Зміна відображення цін і витрат

При натисканні кнопки «Проекти» відкривається вікно для управління проектами енергозбереження (рис. 4.13). У ньому користувач може переглянути і видалити існуючі проекти, додати нові, обрати найвигідніший і запустити його.

При спробі додавання нового проекту без вказання його назви користувач отримає повідомлення про помилку (рис. 4.14).

Проекти

Проекти

Проект	Вартість, грн	Зменшення споживання газу, куб.м	Зменшення споживання твердого пального, т	Зменшення споживання електроенергії, кВт*год	Зменшення споживання рідкого пального, л	Процеси/пристрої	Зменшення витрат за місяць, грн	Окупиться за ... місяців
--------	---------------	----------------------------------	---	--	--	------------------	---------------------------------	--------------------------

Назва проекту:

Вартість, грн:

Зменшення споживання газу, куб.м:

Зменшення споживання твердого пального, т:

Зменшення споживання електроенергії, кВт*год:

Зменшення споживання рідкого пального, л:

Процеси/пристрої:

Рисунок 4.13 — Вікно «Проекти»

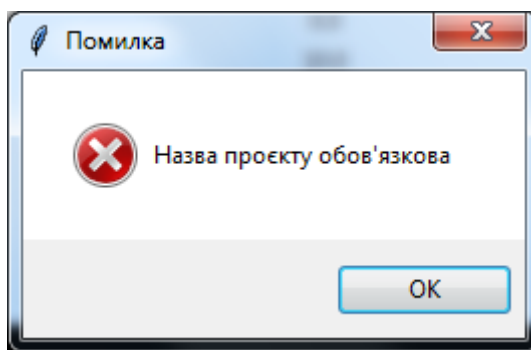


Рисунок 4.14 — Повідомлення про відсутність назви проєкту

При відсутності процесів, на які впливатиме проєкт, користувач також отримає повідомлення про помилку (рис. 4.15).

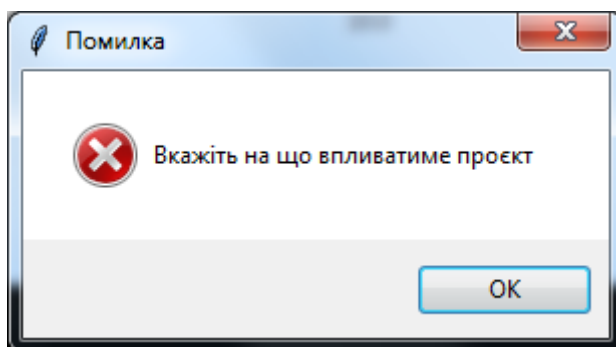


Рисунок 4.15 — Повідомлення про відсутність вказаних процесів

При заповненні полів «Вартість», «Зменшення споживання газу», «Зменшення споживання твердого пального», «Зменшення споживання електроенергії», «Зменшення споживання рідкого пального» не числовими даними користувач отримує повідомлення про помилку (рис. 4.16).

При успішному додаванні нового проєкту запис про нього з'явиться на таблиці (рис. 4.17)

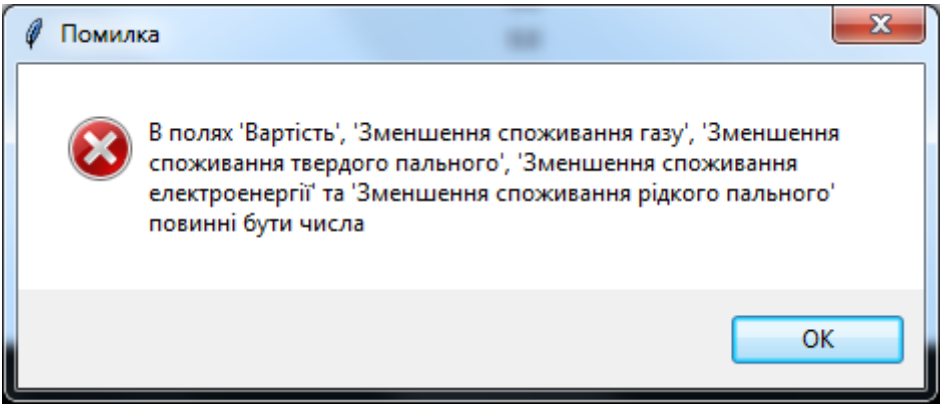


Рисунок 4.16 — Повідомлення про некоректність введених даних

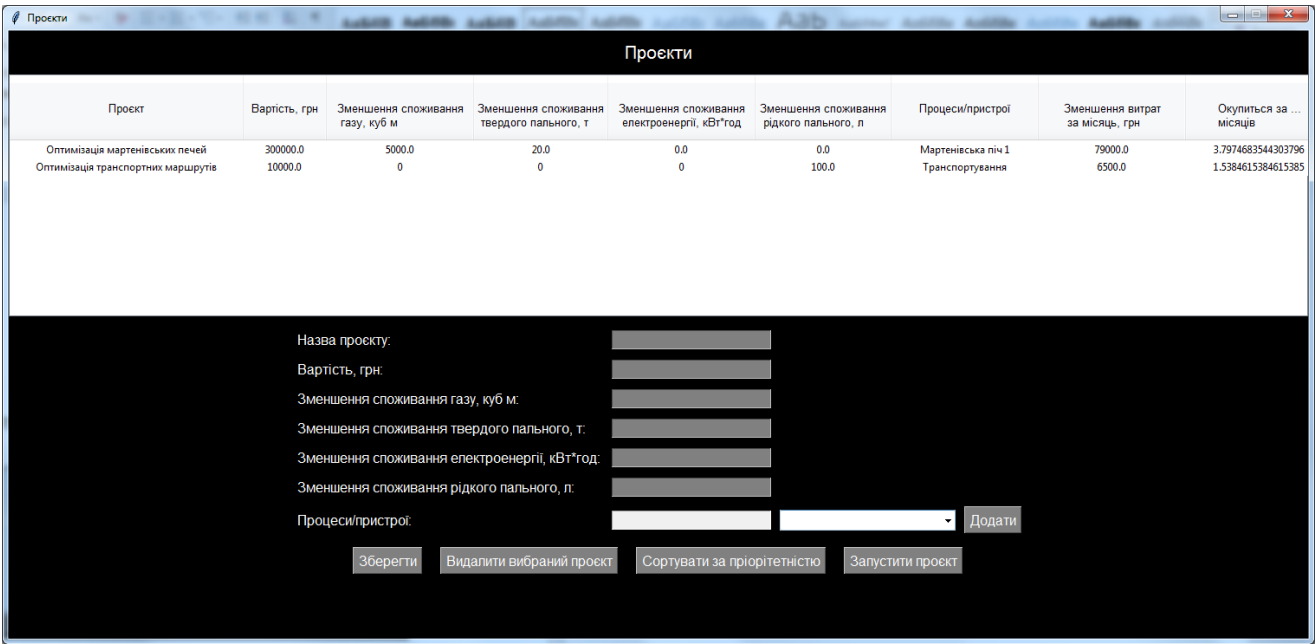


Рисунок 4.17 — Додавання записів до таблиці

При натисканні кнопки «Сортувати за пріоритетністю» таблиця відсортується за результатом ділення сумарного зменшення витрат на термін окупності (рис. 4.18) .

Проекти

Проект	Вартість, грн	Зменшення споживання газу, куб м	Зменшення споживання твердого пального, т	Зменшення споживання електроенергії, кВт*год	Зменшення споживання рідкого пального, л	Процеси/пристрої	Зменшення витрат за місяць, грн	Окупиться за ... місяців
Оптимізація мартенівських печей	300000.0	5000.0	20.0	0.0	0.0	Мартенівська піч 1	79000.0	3.7974683544303796
Модернізація доменних печей	150000.0	3000.0	2.0	10.0	0.0	Доменна піч 1	30440.0	4.92772667542707
Оптимізація транспортних маршрутів	10000.0	0.0	0.0	0.0	100.0	Транспортування	6500.0	1.5384615384615385

Назва проекту:

Вартість, грн:

Зменшення споживання газу, куб м:

Зменшення споживання твердого пального, т:

Зменшення споживання електроенергії, кВт*год:

Зменшення споживання рідкого пального, л:

Процеси/пристрої:

Рисунок 4.18 — Результат сортування

При виборі у таблиці проекту і натискання кнопки «Видалити вибраний проект» обраний проект буде видалено з таблиці (рис. 4.19).

При виборі у таблиці проекту і натискання кнопки «Запустити проект» обраний проект буде застосовано для зміни обсягів витрат (рис. 4.20).

Проекти

Проект	Вартість, грн	Зменшення споживання газу, куб м	Зменшення споживання твердого пального, т	Зменшення споживання електроенергії, кВт*год	Зменшення споживання рідкого пального, л	Процеси/пристрої	Зменшення витрат за місяць, грн	Окупиться за ... місяців
Оптимізація мартенівських піч	300000.0	5000.0	20.0	0.0	0.0	Мартенівська піч 1	79000.0	3.7974683544303796
Модернізація доменних печей	150000.0	3000.0	2.0	10.0	0.0	Доменна піч 1	30440.0	4.92772667542707

Назва проекту:

Вартість, грн:

Зменшення споживання газу, куб м:

Зменшення споживання твердого пального, т:

Зменшення споживання електроенергії, кВт*год:

Зменшення споживання рідкого пального, л:

Процеси/пристрої:

Рисунок 4.19 — Видалення проекту

ПАТ "Запоріжсталь"				
Процес або пристрій	Природний газ, куб.м	Тверде паливо, т	Електроенергія, кВт*год	Рідке паливо, л
Мартенівська піч 1	25000.0	130.0	0.0	0.0
Транспортування	0.0	0.0	0.0	400.0
Освітлення	0.0	0.0	100.0	0.0
Доменна піч 1	10000.0	10.0	100.0	0.0
Додати процес/пристрій Змінити ціни Проекти Видалити вибраний процес/пристрій				
Ціни: Природний газ: 9.0 грн за 1 куб.м Тверде паливо: 1700.0 грн за 1 т Електроенергія: 4.0 грн за 1 кВт*год Рідке паливо: 65.0 грн за 1 л 35000.0 куб.м газу (315000.0грн) 140.0 т твердого пального (238000.0грн) Всього: 200.0 кВт*год електроенергії (800.0грн) 400.0 л рідкого пального (26000.0грн) Сума: 579800.0грн				

Рисунок 4.20 — Результат проекту

При виборі процесу або пристрою і натисканні кнопки «Видалити вибраний процес/пристрій» користувач може видалити обраний процес або пристрій (рис. 4.21).

ПАТ "Запоріжсталь"				
Процес або пристрій	Природний газ, куб.м	Тверде паливо, т	Електроенергія, кВт*год	Рідке паливо, л
Мартенівська піч 1	25000.0	130.0	0.0	0.0
Транспортування	0.0	0.0	0.0	400.0
Доменна піч 1	10000.0	10.0	100.0	0.0
Додати процес/пристрій Змінити ціни Проекти Видалити вибраний процес/пристрій				
Ціни: Природний газ: 9.0 грн за 1 куб.м Тверде паливо: 1700.0 грн за 1 т Електроенергія: 4.0 грн за 1 кВт*год Рідке паливо: 65.0 грн за 1 л 35000.0 куб.м газу (315000.0грн) 140.0 т твердого пального (238000.0грн) Всього: 100.0 кВт*год електроенергії (400.0грн) 400.0 л рідкого пального (26000.0грн) Сума: 579400.0грн				

Рисунок 4.21 — Видалення процесу

4.3 Тестування застосунку та його результати

Для того, щоб перевірити якість застосунку було проведено тестування верстки і функціональне тестування.

Для кожного тестування були створені чеклісти, де було зазначено на яких ос було проведено тестування, список перевірок та результати перевірки. На рис. 4.22 та 4.23 наведено чеклісти тестування верстки та функціонального тестування. Як результат, тестування було пройдено успішно і жодна із перевірок не отримала статус «Не пройдено».

Перевірка	Windows 7	Windows 10
Перевірити коректне відображення вікон застосунку	Пройдено	Пройдено
Перевірити коректне відображення тексту та шрифтів	Пройдено	Пройдено
Перевірити коректне відображення цін	Пройдено	Пройдено
Перевірити коректне відображення витрат на ресурси	Пройдено	Пройдено
Перевірити коректне відображення таблиць	Пройдено	Пройдено
Перевірити коректне відображення кнопок	Пройдено	Пройдено
Перевірити коректне відображення випадаючого меню	Пройдено	Пройдено
Перевірити коректне відображення полів для вводу	Пройдено	Пройдено

Рисунок 4.22 – Чекліст тестування верстки

Перевірка	Windows 7	Windows 10
Перевірити реєстрацію	Пройдено	Пройдено
Перевірити авторизацію	Пройдено	Пройдено
Перевірити додавання процесів	Пройдено	Пройдено
Перевірити видалення процесів	Пройдено	Пройдено
Перевірити можливість зміни цін	Пройдено	Пройдено
Перевірити розрахунок сумарних витрат	Пройдено	Пройдено
Перевірити можливість додавати проекти	Пройдено	Пройдено
Перевірити можливість видалення проектів	Пройдено	Пройдено
Перевірити можливість сортування проектів	Пройдено	Пройдено
Перевірити можливість застосовувати проекти	Пройдено	Пройдено
Перевірити відображення повідомлень про помилки:		
Невірний логін	Пройдено	Пройдено
Невірний пароль	Пройдено	Пройдено
Заповнення цінових значень не числами	Пройдено	Пройдено
Відсутність назви процесу або проекту	Пройдено	Пройдено
Відсутність процесів, на які впливатиме проект	Пройдено	Пройдено

Рисунок 4.23 – Чекліст функціонального тестування

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи було розроблено програмний застосунок для управління портфелями проєктів енергозбереження на металургійних підприємствах.

Було виконано аналіз предметної області, існуючих аналогів, їхні переваги та недоліки, визначено функціональні вимоги до застосунку.

Для розробки було обрано мову програмування Python, IDE PyCharm та СУБД SQLite.

Створений застосунок дозволяє додавати проєкти енергозбереження і аналізувати їх ефективність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Економічний огляд металургійної галузі України [Електрон. ресурс]. — Режим доступу: <https://rating.zone/ekonomichnyj-ohliad-metalurhijsnoi-haluzi-ukrainy/>
2. What Is Microsoft Project? Uses, Features and Pricing [Electronic resource]. — Access mode: <https://www.projectmanager.com/blog/what-is-microsoft-project>
3. Керування проектами [Електрон. ресурс]. — Режим доступу: <https://www.microsoft.com/uk-ua/microsoft-365/project/project-management>
4. Primavera [Electronic resource]. — Access mode: [https://en.wikipedia.org/wiki/Primavera_\(software\)#Products_and_versions](https://en.wikipedia.org/wiki/Primavera_(software)#Products_and_versions)
5. Product Tour—Oracle Project Management [Електрон. ресурс]. — Режим доступу: <https://www.oracle.com/ua/erp/project-portfolio-management-cloud/product-tour/>
6. Projectplace (software) [Електрон. ресурс]. — Режим доступу: [https://en.wikipedia.org/wiki/Projectplace_\(software\)](https://en.wikipedia.org/wiki/Projectplace_(software))
7. Planview Projectplace [Electronic resource]. — Access mode: <https://ppm.winmill.com/planview/planview-projectplace/>
8. Python [Електрон. ресурс]. — Режим доступу: <https://uk.wikipedia.org/wiki/Python>
9. Реляційні бази даних усе, що необхідно про них знати [Електрон. ресурс]. — Режим доступу: <https://foxminded.ua/reliatsiini-bazy-danykh/>
10. Що таке NOSQL бази даних? [Електрон. ресурс]. — Режим доступу: <https://freehost.com.ua/ukr/faq/wiki/chto-takoe-bazi-dannih-nosql/>
11. sqlite3 — інтерфейс DB-API 2.0 для баз даних SQLite [Електрон. ресурс]. — Режим доступу: <https://docs.python.org/uk/3.9/library/sqlite3.html>

ДОДАТОК А

Текст програми

A1. registration_window

```

import tkinter as tk
from tkinter import messagebox
import sqlite3
import main_window

conn = sqlite3.connect('user_data.db')
cursor = conn.cursor()
cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE NOT NULL,
        password TEXT NOT NULL
    )
''')
conn.commit()
conn.close()

def register_user():
    username = username_entry.get()
    password = password_entry.get()
    if not username or not password or username == "Назва підприємства" or password == "Пароль":
        messagebox.showwarning("Помилка", "Будь ласка, заповніть всі поля")
        return

    try:
        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()
        cursor.execute('INSERT INTO users (username, password)
VALUES (?, ?)', (username, password))
        conn.commit()
        user_id = cursor.lastrowid
        conn.close()
        root.destroy()
        main_window.main_window(user_id)
    except sqlite3.IntegrityError:
        messagebox.showerror("Помилка", "Користувач з таким ім'ям вже існує")

def login_user():
    username = username_entry.get()
    password = password_entry.get()
    if not username or not password or username == "Назва підприємства" or password == "Пароль":
        messagebox.showwarning("Помилка", "Будь ласка, заповніть всі поля")

```

```

        return

    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM users WHERE username = ? AND
password = ?', (username, password))
    user_id = cursor.fetchone()
    conn.close()

    if user_id:
        root.destroy()
        main_window.main_window(user_id[0])
    else:
        messagebox.showerror("Помилка", "Невірний логін або
пароль")

root = tk.Tk()
root.title("Авторизація")
root.geometry("250x300")

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

x_position = (screen_width - 250) // 2
y_position = (screen_height - 300) // 2
root.geometry(f"250x300+{x_position}+{y_position}")
root.resizable(False, False)
root.configure(bg='black')
auth_label = tk.Label(root, text="Авторизація", bg='black',
fg='white', font=("Times New Roman", 20))
auth_label.place(relx=0.5, rely=0.15, anchor='center')

username_entry = tk.Entry(root, bg='gray', fg='white',
font=("Arial", 16))
username_entry.insert(0, "Назва підприємства")
username_entry.bind('<FocusIn>', lambda event:
on_entry_click(event, username_entry, "Назва підприємства"))
username_entry.bind('<FocusOut>', lambda event: on_focusout(event,
username_entry, "Назва підприємства"))
username_entry.place(relx=0.5, rely=0.3, anchor='center')
password_entry = tk.Entry(root, bg='gray', fg='white',
font=("Arial", 16), show="*")
password_entry.insert(0, "Пароль")
password_entry.bind('<FocusIn>', lambda event:
on_entry_click(event, password_entry, "Пароль"))
password_entry.bind('<FocusOut>', lambda event: on_focusout(event,
password_entry, "Пароль"))
password_entry.place(relx=0.5, rely=0.4, anchor='center')

button_font = ("Arial", 10)
button_width = 14

```

```

button_height = 2

register_button = tk.Button(root, text="Реєстрація",
                             command=register_user, bg='gray', fg='white', font=button_font,
                             width=button_width,
                             height=button_height)
register_button.place(relx=0.25, rely=0.8, anchor='center')

login_button = tk.Button(root, text="Увійти", command=login_user,
                           bg='gray', fg='white', font=button_font,
                           width=button_width, height=button_height)
login_button.place(relx=0.75, rely=0.8, anchor='center')

def on_entry_click(event, entry, placeholder):
    if entry.get() == placeholder:
        entry.delete(0, "end")
        entry.config(fg='black')

def on_focusout(event, entry, placeholder):
    if entry.get() == '':
        entry.insert(0, placeholder)
        entry.config(fg='grey')

root.mainloop()

```

A2. main_window

```

import tkinter as tk
from tkinter import ttk
import sqlite3
from add_process_window import add_process_window
from price_window import price_window
from project_window import project_window

def main_window(user_id):
    def load_user_name():
        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()
        cursor.execute('SELECT username FROM users WHERE id = ?',
            (user_id,))
        user = cursor.fetchone()
        conn.close()
        return user[0] if user else "Користувач"

    def load_data():
        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()

```

```

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS user_data (
                id INTEGER PRIMARY KEY,
                user_id INTEGER,
                process_or_device TEXT,
                natural_gas REAL,
                solid_energy REAL,
                electrical_energy REAL,
                liquid_energy REAL,
                FOREIGN KEY (user_id) REFERENCES
users(id)
            )
        """)
        cursor.execute('SELECT * FROM user_data WHERE user_id =
?', (user_id,))
        data = cursor.fetchall()
        conn.close()
        return data

def load_rates():
    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS user_rates (
            id INTEGER PRIMARY KEY,
            user_id INTEGER UNIQUE,
            natural_gas REAL,
            solid_energy REAL,
            electrical_energy REAL,
            liquid_energy REAL,
            FOREIGN KEY (user_id) REFERENCES users(id)
        )
    ''')
    cursor.execute('SELECT natural_gas, solid_energy,
electrical_energy, liquid_energy FROM user_rates WHERE user_id =
?',
                    (user_id,))
    rates = cursor.fetchone()
    conn.close()
    return rates if rates else (0, 0, 0, 0)

def update_prices_labels():
    rates = load_rates()
    price_values_label.config(
        text=f"Природний газ: {rates[0]} грн
за 1 куб.м\n Тверде пальне: {rates[1]} грн за 1 т\n
Електроенергія: {rates[2]} грн за 1 кВт*год\n Рідке пальне:
{rates[3]} грн за 1 л",
    )

```

```

def update_total_label():
    total_gas, total_solid_energy, total_electrical_energy,
total_liquid_energy = calculate_total()
    rates = load_rates()
    total_gas_value = total_gas * float(rates[0])
    total_solid_value = total_solid_energy * float(rates[1])
    total_electrical_value = total_electrical_energy *
float(rates[2])
    total_liquid_value = total_liquid_energy * float(rates[3])
    total_value = total_gas_value + total_electrical_value +
total_solid_value + total_liquid_value
    total_values_label.config(
        text=f"{total_gas} куб.м газу (
{total_gas_value}грн)\n {total_solid_energy} т твердого пального
({total_solid_value}грн)\n {total_electrical_energy} кВт*год
електроенергії ({total_electrical_value}грн)\n
{total_liquid_energy} л рідкого пального
({total_liquid_value}грн)\nСума: {total_value}грн")

def refresh_table():
    table.delete(*table.get_children())
    data = load_data()
    for row in data:
        table.insert("", "end", values=row[2:])

def calculate_total():
    data = load_data()
    total_gas = sum(float(row[3]) for row in data)
    total_solid_energy = sum(float(row[4]) for row in data)
    total_electrical_energy = sum(float(row[5]) for row in
data)
    total_liquid_energy = sum(float(row[6]) for row in data)
    return total_gas, total_solid_energy,
total_electrical_energy, total_liquid_energy

def delete_selected_row():
    selected_item = table.selection()
    if selected_item:
        item = table.item(selected_item)
        values = item['values']
        process_or_device = values[0]

        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()
        cursor.execute('DELETE FROM user_data WHERE user_id =
? AND process_or_device = ?',
                        (user_id, process_or_device))
        conn.commit()
        conn.close()

```

```

refresh_table()
update_total_label()

main_window = tk.Tk()
main_window.title("Головне вікно")
main_window.geometry("1450x600")
screen_width = main_window.winfo_screenwidth()
screen_height = main_window.winfo_screenheight()
x_position = (screen_width - 1450) // 2
y_position = (screen_height - 600) // 2
main_window.geometry(f"1450x600+{x_position}+{y_position}")
main_window.resizable(False, False)
style = ttk.Style()
style.configure("Treeview.Heading", font=("Arial", 10),
padding=(0, 30))
main_window.configure(bg='black')
user_name = load_user_name()
tk.Label(main_window, text=user_name, bg='black', fg='white',
font=("Arial", 30)).pack(pady=10)
table_frame = tk.Frame(main_window, bg='black')
table_frame.place(relx=0.5, rely=0.3, anchor='center')
columns = ("Процес або пристрій", "Природний газ, куб.м",
"Тверде пальне, т", "Електроенергія, кВт*год", "Рідке пальне, л")
table = ttk.Treeview(main_window, columns=columns,
show='headings')
for col in columns:
    table.heading(col, text=col)
    table.column(col, width=300, anchor='center')
table.pack()
refresh_table()

def open_process_window():
    add_process_window(user_id, refresh_table,
update_total_label)

    add_button = tk.Button(main_window, text="Додати
процес/пристрій", command=open_process_window, bg='gray',
fg='white',
font=("Arial", 12))
    add_button.place(relx=0.081, rely=0.61, anchor='center')

def open_price_window():
    price_window(user_id, update_prices_labels,
update_total_label)

    price_button = tk.Button(main_window, text="Змінити ціни",
command=open_price_window, bg='gray',
fg='white', font=("Arial", 12))
    price_button.place(relx=0.18, rely=0.61, anchor='center')

```

```

price_frame = tk.Frame(main_window, bg='black')
price_frame.place(relx=0.5, rely=0.75, anchor='center')

price_label = tk.Label(price_frame, text="Ціни: ", bg='black',
fg='white', font=("Arial", 12))
price_label.pack(side="left")

rates = load_rates()
price_values_label = tk.Label(price_frame,
                                text=f"Природний газ: {rates[0]}
грн за 1 куб.м\n Тверде пальне: {rates[1]} грн за 1 т\n
Електроенергія: {rates[2]} грн за 1 кВт*год\n Рідке пальне:
{rates[3]} грн за 1 л",
                                bg='black', fg='white',
font=("Arial", 12))
price_values_label.pack(side="left")

total_frame = tk.Frame(main_window, bg='black')
total_frame.place(relx=0.5, rely=0.90, anchor='center')

total_gas, total_solid_energy, total_electrical_energy,
total_liquid_energy = calculate_total()
total_gas_value = total_gas * float(rates[0])
total_solid_value = total_solid_energy * float(rates[1])
total_electrical_value = total_electrical_energy *
float(rates[2])
total_liquid_value = total_liquid_energy * float(rates[3])
total_value =
total_gas_value+total_electrical_value+total_solid_value+total_liq
uid_value

total_label = tk.Label(total_frame, text=f"Всього: ",
bg='black', fg='white', font=("Arial", 12))
total_label.pack(side="left")

total_values_label = tk.Label(total_frame,
                                text=f"{total_gas} млн куб.м
газу ( {total_gas_value}грн)\n {total_solid_energy} т твердого
пального ( {total_solid_value}грн)\n {total_electrical_energy}
кВт*год електроенергії ( {total_electrical_value}грн)\n
{total_liquid_energy} л рідкого пального
( {total_liquid_value}грн)\n Сума: {total_value}грн",
                                bg='black', fg='white',
font=("Arial", 12))
total_values_label.pack(side="left")

def open_project_window():
    project_window(user_id, refresh_table, update_total_label)

```

```

        create_project_button = tk.Button(main_window, text="Проекти",
command=open_project_window, bg='gray',
                                         fg='white', font=("Arial",
12))
        create_project_button.place(relx=0.2403, rely=0.61,
anchor='center')

        delete_button = tk.Button(main_window, text="Видалити вибраний
процес/пристрій", command=delete_selected_row,
                                bg='gray', fg='white',
font=("Arial", 12))
        delete_button.place(relx=0.889, rely=0.61, anchor='center')

main_window.mainloop()

```

A3. project_window

```

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3

def project_window(user_id, refresh_table_callback, update_label):
    def save_project():
        project_name = project_name_entry.get()
        project_cost = project_cost_entry.get()
        gas_reduction = gas_reduction_entry.get()
        solid_energy_reduction =
solid_energy_reduction_entry.get()
        electrical_energy_reduction =
electrical_energy_reduction_entry.get()
        liquid_energy_reduction =
liquid_energy_reduction_entry.get()
        processes = processes_entry.get()

        if not project_name:
            tk.messagebox.showerror("Помилка", "Назва проекту
обов'язкова")
            return
        if not processes:
            tk.messagebox.showerror("Помилка", "Вкажіть на що
впливатиме проєкт")
            return

        try:
            project_cost = float(project_cost) if project_cost
else 0

```

```

        gas_reduction = float(gas_reduction) if gas_reduction
else 0
        solid_energy_reduction = float(solid_energy_reduction)
if solid_energy_reduction else 0
        electrical_energy_reduction =
float(electrical_energy_reduction) if electrical_energy_reduction
else 0
        liquid_energy_reduction =
float(liquid_energy_reduction) if liquid_energy_reduction else 0

except ValueError:
    messagebox.showerror("Помилка",
                        "В полях 'Вартість',
                        'Зменшення споживання газу', 'Зменшення споживання твердого
пального', 'Зменшення споживання електроенергії' та 'Зменшення
споживання рідкого пального' повинні бути числа")
    return

    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute("SELECT natural_gas, solid_energy,
electrical_energy, liquid_energy FROM user_rates WHERE user_id=?",
                    (user_id,))
    rates = cursor.fetchone()
    if rates:
        gas_price, solid_energy_price,
electrical_energy_price, liquid_energy_price = rates
    else:
        gas_price = solid_energy_price =
electrical_energy_price = liquid_energy_price = 0

    processes_list = processes.split(", ")
    num_processes = len(processes_list)

    total_cost_reduction = num_processes * (
        (gas_reduction * gas_price) +
        (solid_energy_reduction * solid_energy_price) + (
            electrical_energy_reduction *
electrical_energy_price) + (liquid_energy_reduction *
liquid_energy_price))
    if total_cost_reduction != 0:
        payback_period = project_cost / total_cost_reduction
    else:
        payback_period = 0

    if payback_period != 0:
        coefficient = total_cost_reduction / payback_period
    else:
        coefficient = 0

    cursor.execute("""
        INSERT INTO projects (user_id, project_name,
project_cost, gas_reduction, solid_energy_reduction,

```

```
electrical_energy_reduction, liquid_energy_reduction, processes,
cost_reduction_per_month, payback_period, coefficient)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

```
        """ , (
            user_id, project_name, project_cost, gas_reduction,
solid_energy_reduction, electrical_energy_reduction,
liquid_energy_reduction, processes, total_cost_reduction,
payback_period, coefficient))
        conn.commit()
        conn.close()
        table.insert("", "end", values=(
            project_name, project_cost, gas_reduction,
solid_energy_reduction, electrical_energy_reduction,
liquid_energy_reduction, processes, total_cost_reduction,
payback_period))
```

```
        project_name_entry.delete(0, tk.END)
        project_cost_entry.delete(0, tk.END)
        gas_reduction_entry.delete(0, tk.END)
        solid_energy_reduction_entry.delete(0, tk.END)
        electrical_energy_reduction_entry.delete(0, tk.END)
        liquid_energy_reduction_entry.delete(0, tk.END)
        processes_entry.config(state="normal")
        processes_entry.delete(0, tk.END)
        process_combobox.delete(0, tk.END)
        processes_entry.config(state="readonly")
        update_process_combobox()
```

```
def load_data():
    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute("""
```

```
                                CREATE TABLE IF NOT EXISTS projects
(
                                id INTEGER PRIMARY KEY,
                                user_id INTEGER,
                                project_name TEXT,
                                project_cost REAL,
                                gas_reduction REAL,
                                solid_energy_reduction REAL,
                                electrical_energy_reduction
REAL,
                                liquid_energy_reduction REAL,
                                processes TEXT,
                                cost_reduction_per_month REAL,
                                payback_period REAL,
                                coefficient REAL,
                                FOREIGN KEY (user_id)
REFERENCES users(id)
                                )
                                """)
```

```

        cursor.execute("SELECT natural_gas, solid_energy,
electrical_energy, liquid_energy FROM user_rates WHERE user_id=?",
(user_id,))
        rates = cursor.fetchone()

        cursor.execute(
            "SELECT id, project_name, project_cost, gas_reduction,
solid_energy_reduction, electrical_energy_reduction,
liquid_energy_reduction, processes FROM projects WHERE user_id=?",
            (user_id,))
        projects = cursor.fetchall()

        for project in projects:
            project_id, project_name, project_cost, gas_reduction,
solid_energy_reduction, electrical_energy_reduction,
liquid_energy_reduction, processes = project

            processes_list = processes.split(", ")
            cost_reduction_per_month = 0

            for process in processes_list:
                cursor.execute("SELECT natural_gas, solid_energy,
electrical_energy, liquid_energy FROM user_data WHERE
process_or_device=?", (process,))
                usage = cursor.fetchone()
                if usage:
                    cost_reduction_per_month += (
                        (gas_reduction * rates[0]) +
                        (solid_energy_reduction * rates[1]) +
                        (electrical_energy_reduction * rates[2]) +
                        (liquid_energy_reduction * rates[3])
                    )

            payback_period = project_cost /
cost_reduction_per_month if cost_reduction_per_month != 0 else 0
            coefficient = cost_reduction_per_month /
payback_period if payback_period != 0 else 0

            cursor.execute("""
                UPDATE projects
                SET cost_reduction_per_month=?, payback_period=?,
coefficient=?
                WHERE id=?
            """, (cost_reduction_per_month, payback_period,
coefficient, project_id))

            conn.commit()
            cursor.execute(
                "SELECT project_name, project_cost, gas_reduction,
solid_energy_reduction, electrical_energy_reduction,
liquid_energy_reduction, processes, cost_reduction_per_month,
payback_period, coefficient FROM projects WHERE user_id=?",

```

```

        (user_id,))
    data = cursor.fetchall()
    conn.close()
    return data

def load_processes():
    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute("SELECT DISTINCT process_or_device FROM
user_data WHERE user_id=?",
                    (user_id,))
    processes = [row[0] for row in cursor.fetchall()]
    conn.close()
    return processes

def refresh_table():
    table.delete(*table.get_children())
    projects = load_data()
    for row in projects:
        table.insert("", "end", values=row)

def delete_selected_row():
    selected_item = table.selection()
    if selected_item:
        item = table.item(selected_item)
        values = item['values']
        project_name = values[0]

        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()
        cursor.execute('DELETE FROM projects WHERE user_id = ?
AND project_name = ?', (user_id, project_name))
        conn.commit()
        conn.close()
        refresh_table()

def sort_table():
    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute("SELECT project_name, project_cost,
gas_reduction, solid_energy_reduction,
electrical_energy_reduction, liquid_energy_reduction, processes,
cost_reduction_per_month, payback_period, coefficient FROM
projects WHERE user_id=? ORDER BY coefficient DESC", (user_id,))
    sorted_data = cursor.fetchall()
    conn.close()

    table.delete(*table.get_children())
    for row in sorted_data:
        table.insert("", "end", values=row)

```

```

def update_process_combobox():
    process_combobox['values'] = load_processes()

def add_process():
    processes_entry.config(state="normal")
    selected_process = process_combobox.get()
    if selected_process:

        current_processes = processes_entry.get()
        if current_processes:
            updated_processes = f"{current_processes},
{selected_process}"
        else:
            updated_processes = selected_process
        processes_entry.delete(0, tk.END)
        processes_entry.insert(0, updated_processes)
        processes_entry.config(state="readonly")
        process_combobox.delete(0, tk.END)
        process_combobox['values'] = [p for p in
process_combobox['values'] if p != selected_process]

def run_project():
    selected_item = table.selection()
    if not selected_item:
        tk.messagebox.showerror("Помилка", "Виберіть проект
для запуску")
        return

    item = table.item(selected_item)
    values = item['values']
    processes = str(values[6]).split(", ")
    gas_reduction = float(values[2])
    solid_energy_reduction = float(values[3])
    electrical_energy_reduction = float(values[4])
    liquid_energy_reduction = float(values[5])

    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()

    for process in processes:
        cursor.execute(
            "SELECT natural_gas, solid_energy,
electrical_energy, liquid_energy FROM user_data WHERE
process_or_device=?",
            (process,))
        usage = cursor.fetchone()
        if usage:
            gas_usage, solid_energy_usage,
electrical_energy_usage, liquid_energy_usage = usage
            new_gas_usage = max(0, gas_usage - gas_reduction)
            new_solid_energy_usage = max(0, solid_energy_usage
- solid_energy_reduction)

```

```

        new_electrical_energy_usage = max(0,
electrical_energy_usage - electrical_energy_reduction)
        new_liquid_energy_usage = max(0,
liquid_energy_usage - liquid_energy_reduction)

        cursor.execute("""
            UPDATE user_data
            SET natural_gas=?, solid_energy=?,
            electrical_energy=?, liquid_energy=?
            WHERE process_or_device=?
            """, (new_gas_usage, new_solid_energy_usage,
new_electrical_energy_usage, new_liquid_energy_usage, process))
        conn.commit()
    conn.close()
    refresh_table_callback()
    update_label()

project_window = tk.Toplevel()
project_window.title("Проекти")
project_window.geometry("1500x700")
project_window.configure(bg='black')

screen_width = project_window.winfo_screenwidth()
screen_height = project_window.winfo_screenheight()

x_position = (screen_width - 1500) // 2
y_position = (screen_height - 700) // 2
project_window.geometry(f"1500x700+{x_position}+{y_position}")

project_window.resizable(False, False)

tk.Label(project_window, text="Проекти", font=("Arial", 16),
bg='black', fg='white').pack(pady=10)

table_frame = tk.Frame(project_window, bg='black')
table_frame.place(relx=0.5, rely=0.4, anchor='center')

columns = (
    "Проект", "Вартість, грн", "Зменшення споживання\n газу,
куб м",
    "Зменшення споживання\n твердого пального, т",
    "Зменшення споживання\n електроенергії, кВт*год",
    "Зменшення споживання\n рідкого пального, л",
    "Процеси/пристрої",
    "Зменшення витрат\n за місяць, грн",
    "Окупиться за ...\nмісяців")
table = ttk.Treeview(project_window, columns=columns,
show='headings')
for col in columns:
    table.heading(col, text=col)
    table.column(col, width=170, anchor='center')

```

```

table.pack()

input_frame = tk.Frame(project_window, bg='black')
input_frame.pack(pady=10)

tk.Label(input_frame, text="Назва проекту:", font=("Arial",
12), bg='black', fg='white').grid(row=0, column=0,

padx=5, pady=5,

sticky='w')
project_name_entry = tk.Entry(input_frame, font=("Arial", 12),
bg='gray', fg='white')
project_name_entry.grid(row=0, column=1, padx=5, pady=5)

tk.Label(input_frame, text="Вартість, грн:", font=("Arial",
12), bg='black', fg='white').grid(row=1, column=0,

padx=5, pady=5,

sticky='w')
project_cost_entry = tk.Entry(input_frame, font=("Arial", 12),
bg='gray', fg='white')
project_cost_entry.grid(row=1, column=1, padx=5, pady=5)

tk.Label(input_frame, text="Зменшення споживання газу, куб
м:", font=("Arial", 12), bg='black',
fg='white').grid(row=2, column=0, padx=5, pady=5,
sticky='w')
gas_reduction_entry = tk.Entry(input_frame, font=("Arial",
12), bg='gray', fg='white')
gas_reduction_entry.grid(row=2, column=1, padx=5, pady=5)

tk.Label(input_frame, text="Зменшення споживання твердого
паливного, т:", font=("Arial", 12), bg='black',
fg='white').grid(row=3, column=0, padx=5, pady=5,
sticky='w')
solid_energy_reduction_entry = tk.Entry(input_frame,
font=("Arial", 12), bg='gray', fg='white')
solid_energy_reduction_entry.grid(row=3, column=1, padx=5,
pady=5)

tk.Label(input_frame, text="Зменшення споживання
електроенергії, кВт*год:", font=("Arial", 12), bg='black',
fg='white').grid(row=4, column=0, padx=5, pady=5,
sticky='w')
electrical_energy_reduction_entry = tk.Entry(input_frame,
font=("Arial", 12), bg='gray', fg='white')
electrical_energy_reduction_entry.grid(row=4, column=1,
padx=5, pady=5)

```

```

    tk.Label(input_frame, text="Зменшення споживання рідкого
пального, л:", font=("Arial", 12), bg='black',
            fg='white').grid(row=5, column=0, padx=5, pady=5,
sticky='w')
    liquid_energy_reduction_entry = tk.Entry(input_frame,
font=("Arial", 12), bg='gray', fg='white')
    liquid_energy_reduction_entry.grid(row=5, column=1, padx=5,
pady=5)

    tk.Label(input_frame, text="Процеси/пристрої:", font=("Arial",
12), bg='black', fg='white').grid(row=6, column=0,

padx=5, pady=5,

sticky='w')
    processes_entry = tk.Entry(input_frame, font=("Arial", 12),
bg='gray', fg='black', state="readonly")
    processes_entry.grid(row=6, column=1, padx=5, pady=5)

    process_combobox = ttk.Combobox(input_frame, font=("Arial",
12), values=load_processes())
    process_combobox.grid(row=6, column=2, padx=5, pady=5)

    add_process_button = tk.Button(input_frame, text="Додати",
command=add_process, bg='gray', fg='white',
                                font=("Arial", 12))
    add_process_button.grid(row=6, column=3, padx=5, pady=5)

    buttons_frame = tk.Frame(project_window, bg='black')
    buttons_frame.pack()

    save_button = tk.Button(buttons_frame, text="Зберегти",
command=save_project, bg='gray', fg='white',
                                font=("Arial", 12))
    save_button.pack(side=tk.LEFT, padx=10)

    delete_button = tk.Button(buttons_frame, text="Видалити
вибраний проект", command=delete_selected_row, bg='gray',
                                fg='white', font=("Arial", 12))
    delete_button.pack(side=tk.LEFT, padx=10)

    sort_button = tk.Button(buttons_frame, text="Сортувати за
пріоритетністю", command=sort_table, font=("Arial", 12),
                                bg='gray', fg='white')
    sort_button.pack(side=tk.LEFT, padx=10)

    run_project_button = tk.Button(buttons_frame, text="Запустити
проект", command=run_project, bg='gray', fg='white',
                                font=("Arial", 12))
    run_project_button.pack(side=tk.LEFT, padx=10)

    update_process_combobox()

```

```
refresh_table()
project_window.mainloop()
```

A4. price_window

```
import tkinter as tk
import sqlite3

def price_window(user_id, update_prices, update_labels):
    def save_data():
        natural_gas = natural_gas_entry.get().strip()
        solid_energy = solid_energy_entry.get().strip()
        electrical_energy = electrical_energy_entry.get().strip()
        liquid_energy = liquid_energy_entry.get().strip()

        try:
            natural_gas = float(natural_gas) if natural_gas else
0.0
            solid_energy = float(solid_energy) if solid_energy
else 0.0
            electrical_energy = float(electrical_energy) if
electrical_energy else 0.0
            liquid_energy = float(liquid_energy) if liquid_energy
else 0.0
        except ValueError:
            tk.messagebox.showerror("Помилка", "Введіть числа")
            return

        conn = sqlite3.connect('user_data.db')
        cursor = conn.cursor()
        cursor.execute('''
            INSERT INTO user_rates (user_id, natural_gas,
solid_energy, electrical_energy, liquid_energy)
            VALUES (?, ?, ?, ?, ?)
            ON CONFLICT(user_id) DO UPDATE SET
            natural_gas=excluded.natural_gas,
            solid_energy=excluded.solid_energy,
            electrical_energy=excluded.electrical_energy,
            liquid_energy=excluded.liquid_energy
        ''', (user_id, natural_gas, solid_energy,
electrical_energy, liquid_energy))
        conn.commit()
        conn.close()
        update_prices()
        update_labels()
        price_window.destroy()

    price_window = tk.Toplevel()
    price_window.title("Ціни")
```

```

price_window.geometry("400x400")
screen_width = price_window.winfo_screenwidth()
screen_height = price_window.winfo_screenheight()
x_position = (screen_width - 400) // 2
y_position = (screen_height - 400) // 2
price_window.geometry(f"400x400+{x_position}+{y_position}")
price_window.resizable(False, False)
price_window.configure(bg='black')
tk.Label(price_window, text="Природний газ, грн за 1 куб.м",
bg='black', fg='white', font=("Arial", 12)).pack(pady=5)
natural_gas_entry = tk.Entry(price_window, font=("Arial", 12))
natural_gas_entry.pack(pady=5)

tk.Label(price_window, text="Тверде пальне, грн за 1 т",
bg='black', fg='white', font=("Arial", 12)).pack(pady=5)
solid_energy_entry = tk.Entry(price_window, font=("Arial",
12))
solid_energy_entry.pack(pady=5)

tk.Label(price_window, text="Електроенергія, грн за 1
кВт*год", bg='black', fg='white', font=("Arial", 12)).pack(pady=5)
electrical_energy_entry = tk.Entry(price_window,
font=("Arial", 12))
electrical_energy_entry.pack(pady=5)

tk.Label(price_window, text="Підке пальне, грн за 1 л",
bg='black', fg='white', font=("Arial", 12)).pack(pady=5)
liquid_energy_entry = tk.Entry(price_window, font=("Arial",
12))
liquid_energy_entry.pack(pady=5)

save_button = tk.Button(price_window, text="Зберегти",
command=save_data, bg='gray', fg='white', font=("Arial", 12))
save_button.pack(pady=20)

price_window.mainloop()

```

A5. add_process_window

```

import tkinter as tk
import sqlite3

def add_process_window(user_id, refresh_table_callback,
update_label):
    def save_data():
        process = process_entry.get().strip()
        natural_gas = natural_gas_entry.get().strip()
        solid_energy = solid_energy_entry.get().strip()
        electrical_energy = electrical_energy_entry.get().strip()
        liquid_energy = liquid_energy_entry.get().strip()

        if not process:

```

```

        tk.messagebox.showerror("Помилка", "Процесс/пристрій
обов'язково має бути заповнений")
        return

    try:
        natural_gas = float(natural_gas) if natural_gas else
0.0
        solid_energy = float(solid_energy) if solid_energy
else 0.0
        electrical_energy = float(electrical_energy) if
electrical_energy else 0.0
        liquid_energy = float(liquid_energy) if liquid_energy
else 0.0
    except ValueError:
        tk.messagebox.showerror("Помилка", "В поля 'Природний
газ', 'Тверде пальне', 'Електроенергія' і 'Рідке пальне' повинні
бути введені числа")
        return

    conn = sqlite3.connect('user_data.db')
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO user_data (user_id, process_or_device,
natural_gas, solid_energy, electrical_energy, liquid_energy)
        VALUES (?, ?, ?, ?, ?, ?)
    ''', (user_id, process, natural_gas, solid_energy,
electrical_energy, liquid_energy))
    conn.commit()
    conn.close()

    refresh_table_callback()
    update_label()
    add_window.destroy()

add_window = tk.Toplevel()
add_window.title("Додати процес/пристрій")
add_window.geometry("400x400")

screen_width = add_window.winfo_screenwidth()
screen_height = add_window.winfo_screenheight()

x_position = (screen_width - 400) // 2
y_position = (screen_height - 400) // 2
add_window.geometry(f"400x400+{x_position}+{y_position}")

add_window.resizable(False, False)

add_window.configure(bg='black')

tk.Label(add_window, text="Процесс/пристрій", bg='black',
fg='white', font=("Arial", 12)).pack(pady=5)

```

```

process_entry = tk.Entry(add_window, font=("Arial", 12))
process_entry.pack(pady=5)

tk.Label(add_window, text="Природний газ, куб м", bg='black',
fg='white', font=("Arial", 12)).pack(pady=5)
natural_gas_entry = tk.Entry(add_window, font=("Arial", 12))
natural_gas_entry.pack(pady=5)

tk.Label(add_window, text="Тверде пальне, т", bg='black',
fg='white', font=("Arial", 12)).pack(pady=5)
solid_energy_entry = tk.Entry(add_window, font=("Arial", 12))
solid_energy_entry.pack(pady=5)

tk.Label(add_window, text="Електроенергія, кВт*год",
bg='black', fg='white', font=("Arial", 12)).pack(pady=5)
electrical_energy_entry = tk.Entry(add_window, font=("Arial",
12))
electrical_energy_entry.pack(pady=5)

tk.Label(add_window, text="Рідке пальне, л", bg='black',
fg='white', font=("Arial", 12)).pack(pady=5)
liquid_energy_entry = tk.Entry(add_window, font=("Arial", 12))
liquid_energy_entry.pack(pady=5)

save_button = tk.Button(add_window, text="Зберегти",
command=save_data, bg='gray', fg='white', font=("Arial", 12))
save_button.pack(pady=20)

add_window.mainloop()

```

ДОДАТОК Б
Слайди презентації

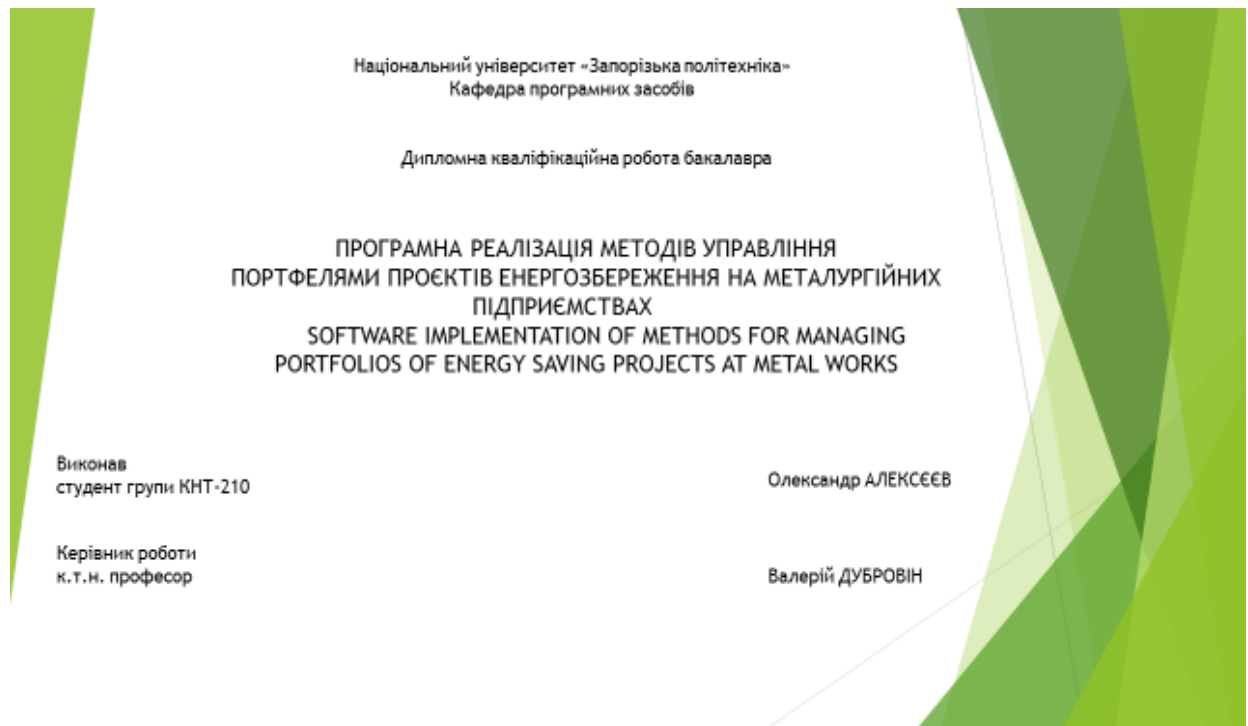


Рисунок Б.1 — Слайд 1

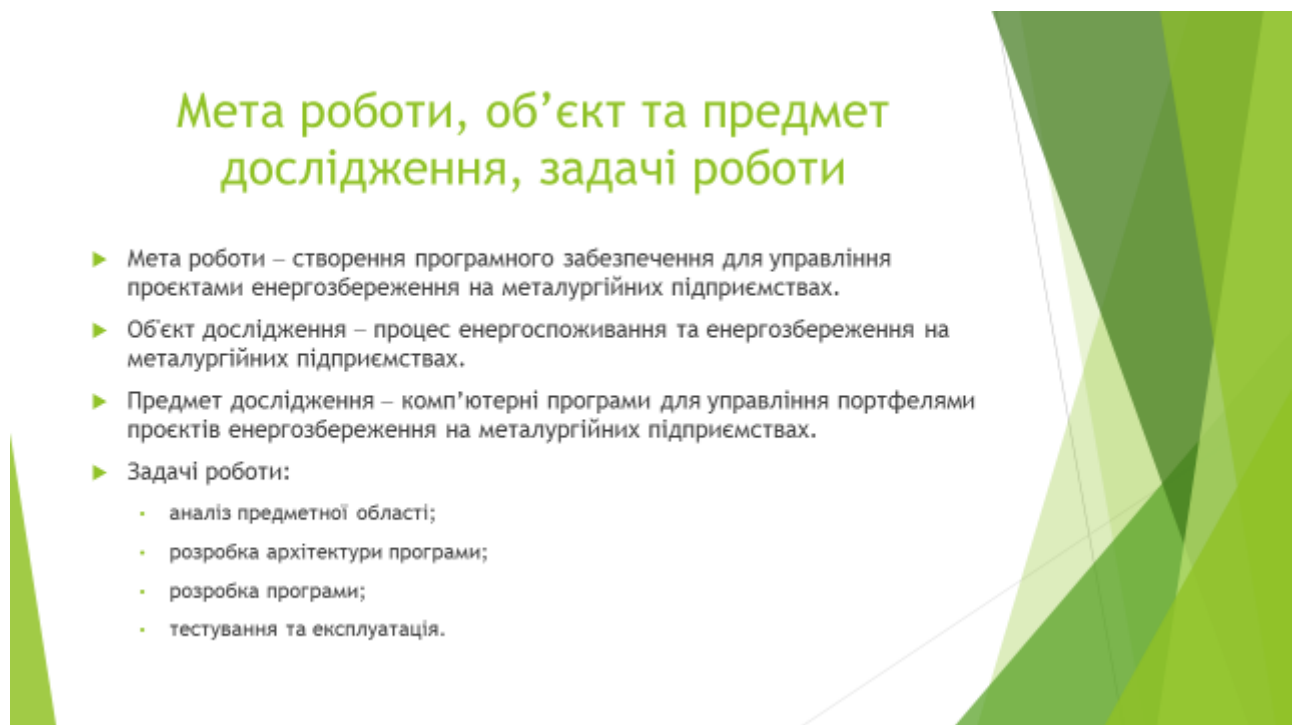


Рисунок Б.2 — Слайд 2

Вибір мови програмування

Критерій порівняння	Python	Java	C++
Продуктивність	Висока	Висока	Середня
Складність розробки	Низька	Середня	Висока
Безпека та надійність	Висока	Висока	Низька
Підтримка спільноти	Висока	Висока	Середня

Рисунок Б.3 — Слайд 3

Діаграма прецедентів

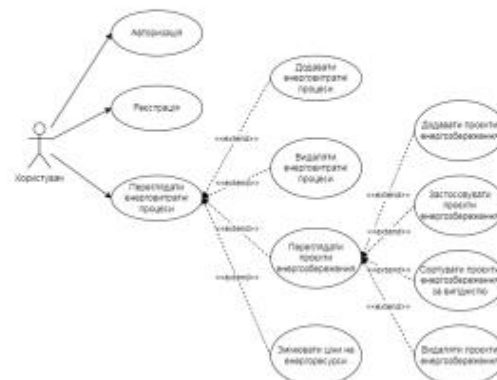


Рисунок 1 – Діаграма прецедентів

Рисунок Б.4 — Слайд 4

Вибір типу бази даних

Опис	Реляційна база даних	Нереляційна база даних
Структура даних	Дані представлені у вигляді таблиць	Дані можуть бути представлені у вигляді документів, графів та тощо
Гнучкість	Менш гнучка	Більш гнучка
Нормалізація	Потребує	Не потребує
Масштабованість	Менш масштабована	Більш масштабована

Рисунок Б.5 — Слайд 5

Схема бази даних

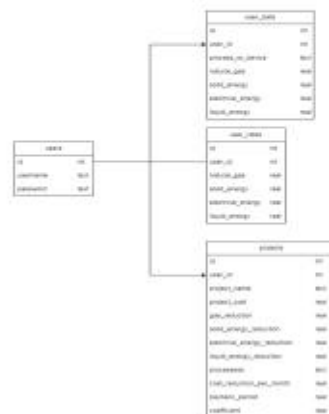


Рисунок 3 – Схема бази даних

Рисунок Б.6 — Слайд 6



Рисунок 2 – Схема функціонування застосунку

Рисунок Б.7 — Слайд 7

Інтерфейси застосунку

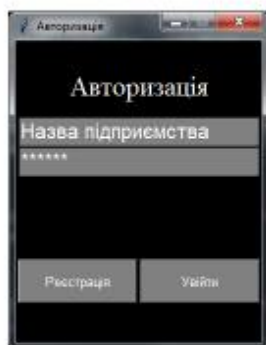


Рисунок 4 – Вікно авторизації



Рисунок 5 – Головне вікно

Рисунок Б.8 — Слайд 8

Інтерфейси застосунку

Рисунок 6 – Вікно додавання процесів

Рисунок 7 – Вікно зміни цін

Рисунок Б.9 — Слайд 9

Інтерфейси застосунку

Рисунок 8 – Вікно «Проекти»

Рисунок Б.10 — Слайд 10

Висновки

- ▶ Під час виконання дипломної кваліфікаційної роботи було розроблено програмний застосунок для управління портфелями проєктів енергозбереження на металургійних підприємствах.
- ▶ Було виконано аналіз предметної області, існуючих аналогів, їхні переваги та недоліки, визначено функціональні вимоги до застосунку.
- ▶ Для розробки було обрано мову програмування Python, IDE PyCharm та СУБД SQLite.
- ▶ Створений застосунок дозволяє додавати проєкти енергозбереження і аналізувати їх ефективність.

Рисунок Б.11 — Слайд 11