МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних
технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

Система підтримки і прийняття рішень в енергетичних системах

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 41

 Кульковець В.В.
(прізвище та ініціали)

Керівник Кіктєв М.О.
(прізвище та ініціали)

(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захистурішенням кафедри *інтелектуальних технологій*
Протокол №_13__від_05.06.2023 р.
зав. кафедри_____доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кульковцю Віктору Васильовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

«Система підтримки і прийняття рішень в енергетичних системах»
затверджена протоколом засідання кафедри від « 11 » листопада 2022 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 07.05.2023 року

3. Вихідні дані до проекту (роботи)

Де і скільки електроенергії споживають та виробляють в Україні
https://businessviews.com.ua/ru/studies/id/da-budet-svet-gde-i-skolko-elektroenergii-potrebljajut-v-ukraine-1166/
Прогнозований баланс електроенергії на 2022 рік - https://vse.energy/news/hotnews/1892-balance-ee-2022

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Аналіз існуючих інформаційних систем для визначення чи генерації графіку погодинних відключень
Аналіз області застосування розроблюваної системи
Постановка задачі на розробку СППР у енергетичних системах
Класифікація СППР
Розробка механізму розрахунку часу погодинних відключень електроенергії
Розробка архітектури
Проектування графічної складової програми
Реалізація СППР енергетичних систем для генерації графіку погодинних відключень електроенергії

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Область застосування розроблюваної системи
Функціональні та нефункціональні вимоги
Детальне пояснення механізму розрахунку часу погодинних відключень електроенергії
Розрахунок енергоспоживання електроенергії в Україні 2022року
Інструменти для реалізації створення СППР
Варіанти використання СППР енергетичних системах, дерево функцій, карта процесів
IDEF0 та DFD для створюваної СППР в енергетичних системах
Структурна схема СППР енергетичних систем
Результат створення графіку погодинних відключень

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 16 лютого 2023 року

Керівник _____ / Кіктєв М.О.
(підпис) (ПІБ)

Завдання прийняв до виконання  / Кульковець В.В. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	І процентування	16.02.2023 –01.03.2023	
2	ІІ процентування	01.04.2023 –10.04.2023	
3	ІІІ процентування	01.05.2023 –10.05.2023	
4	Передзахист	19.05.2023 – 28.05.2023	
5	Перевірка готових дипломних робіт на плагіат.	30.05.2023 – 06.06.2023	Після перевірки на плагіат правки у роботу вносити не можна.
6	Здача готових дипломних робіт на кафедру.	07.06.2023 – 11.06.2023	• підписана керівником та студентом кваліфікаційна робота. • підписаний відгук керівника • підписана рецензія • архів з програмним продуктом • презентація доповіді

Студент-дипломник  / Кульковець В.В /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Кіктєв М.О. /
(підпис) (ПІБ)

Анотація

Кульковець Віктор Васильович виконав випускну кваліфікаційну роботу на тему «Система підтримки і прийняття рішень в енергетичних системах» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних методів створення систем підтримки прийняття рішень, розглянуто готові конкурентні системи на ринку та їх унікальність, розроблено інформаційне та програмне забезпечення, що виконує генерацію графіку погодинних відключень.

Ключові слова: система підтримки прийняття рішень, енергетична система, графік погодинних відключень.

Summary

The degree project: "Support and Decision-Making System in Energy Systems" has completed by Viktor Kulkovets specialty 122 - "Computer Science".

The graduation thesis analyzes modern methods for creating support and decision-making systems, examines existing competitive systems in the market and their uniqueness, and develops informational and software solutions for generating hourly blackout schedules.

Keywords: support and decision-making system, energy system, hourly blackout schedules.

ЗМІСТ

Вступ

Сучасний світ безперечно є чудовим і складним механізмом, який дозволяє людям отримувати максимум користі з електроенергії. Проте, дійсність полягає в тому, що з технологічним прогресом виникають нові виклики і проблеми, з якими нам доводиться стикатися. Агресивна поведінка деяких країн може призвести до руйнування енергетичних інфраструктури, що має негативний вплив на розвиток економіки та побутового сектору.

Після того, як Україна стала об'єктом військової агресії, енергетичні системи країни були піддані серйозному тиску та знищенню. Атаки на енергетичні структури призвели до відключення електропостачання у багатьох населених пунктах, що створило надзвичайну ситуацію та погіршило якість життя мільйонів українців.

Проблеми з енергопостачанням в Україні справді мають серйозні економічні та соціальні наслідки. Недостатнє енергопостачання може призвести до значних збитків для підприємств, зменшення виробництва та втрати робочих місць. Відсутність надійного енергопостачання також може негативно вплинути на якість життя громадян, оскільки багато аспектів повсякденного життя стають залежними від електроенергії.

Для подолання цих проблем в Україні потрібно приділяти увагу розвитку і модернізації енергетичної інфраструктури, впровадженню ефективних систем управління та моніторингу, а також посиленню енергоефективності та використанню відновлювальних джерел енергії. Розвиток інтегрованих систем підтримки прийняття рішень може сприяти ефективнішому розподілу та використанню енергоресурсів, зменшенню ризиків аварій та підвищенню надійності енергетичних систем.

Для вирішення проблеми розподілу електроенергії потрібно розробити систему підтримки прийняття рішень.

Об'єктом дослідження цієї роботи є процеси розподілення електроенергії в енергетичній системі України.

Предметом дослідження є формування графіку погодинних відключень областей в залежності від споживання електроенергії.

Метою роботи є розробка системи підтримки прийняття рішень в енергетичних системах для генерації графіку погодинних відключень електроенергії.

Для досягнення даної мети необхідно вирішити наступні задачі:

- Аналіз систем підтримки прийняття рішень
- Розробка механізму розрахунку часу погодинних відключень електроенергії
- Створення архітектури СППР у енергетичних системах
- Реалізація СППР енергетичних системах для генерації графіку погодинних відключень електроенергії

1. АНАЛІЗ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

1.1 Аналітичний огляд літератури за темою дослідження

СППР - інтерактивна прикладна система, яка надає можливість користувачам, котрі мають намір прийняти рішення, легкий, швидкий та зручний доступ до даних і моделей з метою прийняття рішень у слабо структурованих і неструктурованих ситуаціях з різних галузей людської життєдіяльності.[1]

Для вирішення поставленої задачі необхідно виконати аналіз вхідних структур та на основі цих даних вибрати ту, що забезпечує найкращий і найшвидший пошук результату.

Існує велика кількість видів СППР через наявність різних типів та форм по підтримці прийняття рішень. Однак, кожну із них об’єднують ці компоненти:

- Підсистема введення та аналізу користувача
- Підсистема оброблення запитів та генерації результатів
- База знань та даних
- Підсистема виведення результатів у читабельній формі[1]

Підсистема введення та аналізу повинна містити в собі опис кожного із припустимих запитів та зручне подання. Є обов’язковим вказування типу вхідної інформації та обґрунтування його вибору.

Підсистема оброблення даних і генерації результатів є основною частиною СППР, яке бере на себе задачу у обробленню запитів користувача. Система звертається до бази даних та знань за потрібними алгоритмами, умовами, критеріями, виконує операцію по обробленню вхідних даних і передає кінцеві значення в систему подання разом із інформацією щодо правильного типу подання кінцевих результатів.[2]

База знань та даних відіграє не менш важливу роль у всій системі. Вона містить як інформацію, що була структурно організована, так і можливі алгоритми обробки цих даних.

І останній елемент, підсистема ППР, враховує всі типи запитів і можливі форми виведення оброблених результатів для зручного сприйняття користувачем.

Задачі по прийняттю рішень постійно виникають та розв'язуються у навколишньому середовищі. Їх можна зустріти у біологічних, хімічних, фізичних, екологічних, соціальних, економічних та енергетичних системах, різної складності процесах та явищах.[2]

Рішенням системи вважають безпомилковий та обґрунтований набір зі сторони користувача, що приймає рішення. Ці початкові дії спрямовані взаємодіяти з об'єктом та системою для пошуку бажаного стану або досягнення встановленої мети.

Розумові діяльність, яка є проявом людини також називають рішенням. Можна визначити такі основні ознаки рішення:

- Достатня кількість альтернатив
- Є основна ціль
- Необхідність вольового акту ОПР[1]

Також варто не забувати важливість класифікації рішень, адже кожне із них по своєму має унікальні характеристики. За існуючими підходами їх можна класифікувати як:

- Гарно структуровані, погано структуровані та неструктуровані
- Статичні та динамічні
- Умови визначеності, ризику чи невизначеності
- Одна особа чи багато осіб
- Стратегічне чи тактичне[1]

Слід зауважити, що рішення мають певну модель. Під цією моделлю мають на увазі формальне подання поставленої задачі та процесу прийняття

рішень. Існує фундаментальна проблема прийняття рішень, що зосереджена у походженні критерію оптимальності.[2] У теорії прийняття рішень були досліджені основні задачі: опису, аналізу, вибору та теоретичних конструкцій як корисність, перевага, тощо. У сучасно світі рішення індивідів та колективу завжди потрібно досліджувати окремо, так як другі несуть певний вплив на їх прийняття.

Наявні кілька основних етапів у процесі прийняття рішень:

- Постановка задачі
- Формування рішень
- Вибору рішень[1]

Сама постановка задачі складається із важливих фаз аналізу та діагностики проблеми і визначення основних цілей для рішення. Цей етап відповідає за виявлення та опис проблемної ситуації, збір всієї необхідної інформації та даних. Визначаються всі цілі, серед яких вибирають найкращі, які відповідають нашому напряму пошуків, при цьому знищуються гірші, що не відповідають їм.

Етап формування рішень по своєму унікальний. Він складається з фаз формулювання обмежень, критеріїв для прийняття рішень та визначення альтернатив рішень. Тепер же за нами ціль у розподіленні прийнятних та неприйнятних рішень, за умовами, які визначаються визначеними обмеженнями. Розпочинається процес формування критеріїв, які нам допомагають відсортувати серед кращих придатних значень нам необхідні. Та залишається останній процес формування множини всіх альтернатив, які будуть задіяні для пошуку та розробки альтернативних варіантів рішення.[1]

І нарешті, етап вибору рішення, котрий складається з фаз оцінки існуючих альтернатив та кінцевого вибору рішення. Саме тут за обраною множиною критеріїв для наших альтернатив відбувається остаточне оцінювання, яке повинно вплинути на наш остаточний вибір рішення. За нормальних умови ми завжди матимемо фаворита, до якого будемо найбільше

прислухатися та прагнути його вибрати як результат, але через специфіку критеріїв зазвичай виникають ситуації, що ускладнюють вибір.

1.2 Аналіз існуючих інформаційних систем для визначення чи генерації графіку погодинних відключень

У наш час для вирішення кожної поставленої задачі було знайдено не один метод. Але існують задачі, які несуть унікальний характер та використовуються так рідко, що складно знайти альтернативу у вільному доступі, бо швидше за все вони є лише приватними розробками. Так, для моєї поставленої задачі потрібно знаходити методи генерації графіків погодинних відключень електроенергії по всій Україні. Виникла ця потреба за умов дефіциту вільної електроенергії у системи. Якщо брати будь-яку країну світу, то за нормальних умов не використовується вирішення даної задачі, так як всі критично-важливі і не тільки структури мають доступ до безмежної кількості електроенергії. Але під час катаклізмів чи унікальних подій, як війна, люди отримують нові задачі що потребують вирішення.

Використовуючи джерела інтернет та сотні пошукових запитів різного типу, з використовуванням різних формувань на різним мовах Світу не було знайдено ні єдиного рішення для даної задачі. Звісно, враховуючи дії української системи управління енергією, вони мають та використовують стратегії по рівномірному розподіленню живлення між кожною з областей та важливими інфраструктурами. Але, на превеликий жаль, дана інформація у закритому доступі і звичайним людям вона недоступна. Я вважаю через те, що там як змінні на вхід, чи допоміжні параметри використовується інформація, що заборонена до загальних публікацій.

За своєю структурою певною мірою СППР для енергетичних систем може бути схожою на генерацію розкладу для навчання. Так як ми маємо

дні(які можна інтерпретувати як області) та навчальні дисципліни (як режим увімкненого світла). Хоч система паралелей дуже складна, і здається невдалою, це найкращий можливий сценарій порівняння на сьогоднішній день з урахуванням всіх існуючих факторів.

1. Free College Schedule Maker - цей веб-додаток надає можливість створювати безкоштовні щотижневі розклади занять. Ви можете зберігати ці розклади на своєму комп'ютері. Крім того, ви можете імпортувати збережений розклад, якщо бажаєте змінити свої курси або предмети. За допомогою безкоштовної програми розкладу коледжів ви зможете налаштувати свій розклад занять, змінивши такі параметри, як день початку тижня, тривалість занять та формат часу (12-годинний або 24-годинний). Також ви зможете налаштувати вигляд свого розкладу, включивши або виключивши межі, зменшивши висоту розкладу та відображаючи вихідні дні.

Особливості:

- Створюйте щотижневі розклади занять
- Друк розкладів
- Експортуйте, щоб зберегти розклад на комп’ютері
- Імпортувати, щоб завантажити розклад, збережений на комп’ютері
- Зберегти розклад як зображення [13]

2. Конструктор розкладів - чудовий інструмент для онлайн-планування, що дозволяє безкоштовно створювати графіки. Ви зможете створити до п'яти щоденних або тижневих графіків. Розклад можна зберегти як зображення або PDF-файл, а також його можна роздрукувати на папері. Додаток підтримує дев'ять мов, включаючи англійську, французьку, шведську, російську та інші. Ви зможете налаштувати свій розклад, вибравши власне фонове зображення. Крім того, доступні відео-посібники, які допоможуть вам крок за кроком створити свій розклад.

Особливості:

- Розклад друку

- Збережіть до п’яти графіків
- Діліться графіком
- Збережіть розклад як зображення та PDF
- Графік імпорту / експорту [8]

3. Adobe Spark - цей безкоштовний веб-додаток дозволяє вам створювати власні графіки. Ви можете складати розклади занять, робочі графіки або особисті плани з використанням цього онлайн-інструменту для планування. Додаток надає можливість створювати розклади з обранням зображень, текстів і логотипів за вашим бажанням. Ви можете вибрати шаблон, додавати текст і змінювати розміри документів зручним інтерфейсом перетягування. За допомогою простого візуального редактора ви зможете переглядати дизайн та вносити зміни до свого графіка.

Особливості:

- Розробити індивідуальний графік
- Підтримка логотипів, типографіки та зображень
- Додавання / редагування розділів
- Збережіть, поділіться або роздрукуйте розклад [14]

Кожен із додатків по своєму унікальний і має сенс використовувати у тому чи іншому завданні. Спробуємо порівняти та проаналізувати вже існуючі застосунки, було визначено такі необхідні характеристики:

- Зручність у використанні
- Кросплатформеність
- Імпорт та експорт результатів
- Націленість на звичайних людей
- Конвертація до типу зображення
- Безкоштовність

Таблиця 1. Порівняльний аналіз існуючих рішень

	Free College Schedule Maker	Конструктор розкладів	Adobe Spark
Зручність у використанні	-	+	+
Кросплатформеність	+	+	-
Імпорт та експорт результатів	+	+	-
Націленість на звичайних людей	+	+	-
Конвертація до типу зображення	+	+	+
Безкоштовність	+	+	+

Нам вдалося розглянути вище застосунки, ціль яких є допомога у створенні простих речей звичайним користувачам. Кожен із представлених варіантів є безкоштовним, по своєму унікальним та зручним. Ці розроблені додатки повинні надихати мене як користувача до створення подібних систем у своїй роботі, поліпшивши одну або декілька сторін кожного з них.

Ціль додатку, що створюється у даній дипломній роботі це допомогти вирішити складну проблему з побудовою графіку погодинних відключень електроенергії в Україні за умов нестачі електрики та наявності певних типів пошкоджень. Даний додаток націлений на вирішення вузького типу задач, але такий необхідний у свій час.

1.3 Область застосування розроблюваної системи

Мета цієї дипломної роботи полягає у розробці та реалізації СППР, яка здатна згенерувати графік погодинних відключень в енергетичних системах. Система буде отримувати вхідні параметри щодо стану отримання електроенергії на кожну з областей, такі як попит на енергію та стан мережі.

Область застосування розроблюваної системи підтримки прийняття рішень в енергетичних системах є широкою і різноманітною. Нижче наведено деякі з основних областей, де ця система може знайти своє застосування:

- Енергодистрибуція та управління мережею: Система може бути використана операторами енергетичних мереж для оптимізації розподілу ресурсів та управління мережею. Вона допоможе зменшити навантаження на ділянки мережі, які переживають пікові навантаження, шляхом розподілу енергетичного навантаження та планування погодинних відключень.
- Енергоспоживачі та промислові підприємства: Система може бути корисною для промислових підприємств та великих споживачів електроенергії. Вона допоможе їм планувати своє виробництво та споживання енергії з урахуванням графіка погодинних відключень, що дозволить знизити витрати на електроенергію та підтримувати стабільну роботу в умовах обмеженого постачання.
- Відновлювальна енергетика: З урахуванням зростання використання відновлювальних джерел енергії, система підтримки прийняття рішень може допомогти ефективно інтегрувати ці джерела в енергетичну систему. Вона буде здатна враховувати прогнози виробництва енергії з відновлювальних джерел та визначати оптимальну стратегію використання цих джерел у контексті погодинних відключень та потреб споживачів.

- Енергетична ефективність: Система підтримки прийняття рішень також може знайти застосування в полі енергетичної ефективності. Вона може допомогти виявляти області зайвого споживання енергії та рекомендувати оптимальні заходи для його зниження. Шляхом аналізу даних про споживання, погодинні відключення та енергетичні навантаження, система може ідентифікувати енергоефективність споживачів та розробляти плани для поліпшення енергоефективності в енергетичній системі.
- Екстрені ситуації та аварійне управління: Система підтримки прийняття рішень може бути незамінним інструментом для управління екстреними ситуаціями та аваріями в енергетичних системах. Вона може надати операторам мережі необхідну інформацію для швидкого виявлення проблем та прийняття ефективних рішень щодо відключення, перерозподілу енергії та відновлення нормального режиму роботи системи.

Застосування системи підтримки прийняття рішень у енергетичних системах має значний потенціал для покращення ефективності, стабільності та надійності енергопостачання. Вона дозволяє операторам мережі здійснювати оптимальне планування та управління ресурсами, а також реагувати на зміни в попиті та умовах енергетичної системи. В результаті цього досягається зниження витрат, оптимальне використання ресурсів та підвищення задоволеності споживачів електроенергії.

1.4 Постановка задачі на розробку створення системи підтримки прийняття рішень в енергетичних системах

В першу чергу визначимо вимоги для вхідних даних. Вони повинні задовольняти декільком критеріям:

1. Достовірність - вхідні дані повинні бути точними та відповідати дійсності, справжнім необхідним та наданим кількостям електроенергії.
2. Акуратність - вхідні дані повинні бути правильно введені та форматовані. Їх значення повинні бути задані у певних межах, невід’ємні та чисельні.
3. Релевантність - вхідні дані повинні бути пов'язані з темою системи підтримки прийняття рішень в енергетичних системах.
4. Комплексність - вхідні дані повинні включати всі необхідні аспекти для прийняття висновків та розробки рішень.
5. Надійність - вхідні дані повинні бути стійкими до втрати та пошкодження.
6. Структурованість - вхідні дані повинні бути структуровані згідно з вимогами системи.
7. Величина - вхідні дані повинні бути величинами, зрозумілими та легко інтерпретованими системою.
8. Кількість - вхідні дані повинні включати достатню кількість параметрів для розробки висновків та прийняття рішень.
9. Частота - вхідні дані повинні оновлюватись з достатньою частотою, щоб мати актуальну інформацію.
10. Повнота - всі необхідні дані повинні бути представлені для виконання завдань системи.
11. Швидкість - система повинна забезпечувати швидкий доступ до даних та швидке їх оброблення.
12. Готовність - дані повинні бути доступними в будь-який момент, коли система потребує їх для роботи.
13. Розширюваність - система повинна забезпечувати можливість додавання нових даних та їх оброблення без необхідності переробки всіх існуючих даних.

14. Конфіденційність - система повинна забезпечувати захист конфіденційної інформації від несанкціонованого доступу.

Також у нас є нефункціональні вимоги, які необхідно використати і вони повинні як мінімум включати в себе:

1. Надійність: система повинна бути надійною та стійкою до відмов, щоб забезпечити безперебійну роботу при різних умовах використання.
2. Ефективність: система повинна бути швидкою та ефективною у виконанні завдань, щоб забезпечити швидке відгук і задоволення користувачів. Орієнтований час для роботи програми до 5с.
3. Безпека: система повинна бути захищеною від несанкціонованого доступу та зломів, щоб запобігти витоку конфіденційної інформації. Це можна забезпечити за допомогою встановлення систем захисту від зломів.
4. Масштабованість: система повинна бути здатною масштабуватися в залежності від зростаючих потреб користувачів, щоб забезпечити надійну роботу при збільшенні обсягів даних.
5. Кросплатформність: система повинна бути здатною взаємодіяти з іншими системами та платформами, щоб забезпечити безперебійний обмін даними та інформацією з іншими системами.
6. Сумісність: система повинна бути сумісною з різними пристроями, платформами та програмними засобами, що дозволить користувачам використовувати її на різних пристроях.
7. Доступність: система повинна бути доступною для користувачів з обмеженими можливостями, таких як люди з вадами зору або слуху, щоб забезпечити доступність для всіх користувачів.
8. Юзабіліті: система повинна бути простою і зрозумілою для користувачів, щоб забезпечити легкість користування та максимальну продуктивність.

Функціональні вимоги будуть складатися з таких пунктів:

1. Система повинна дозволяти вносити та зберігати інформацію про споживання електроенергії в різних областях України.
2. Система повинна визначати місця, де може виникнути нестача електроенергії, на основі даних про споживання.
3. Система повинна розробляти графік погодинних відключень для кожної області, щоб уникнути нестачі електроенергії.
4. Система повинна бути здатна прогнозувати можливі несправності в енергетичних структурах та запобігати їх виникненню.
5. Система повинна забезпечувати можливість редагувати початкові параметри, які впливають на процес генерації графіку погодинних відключень.
6. Система повинна мати можливість редагувати графік погодинних відключень в разі зміни умов.
7. Система повинна забезпечувати захист від несанкціонованого доступу до даних та можливість резервного копіювання інформації.
8. Система повинна бути здатна працювати в режимі реального часу.
9. Система повинна мати можливість проводити симуляцію відключень для оцінки можливих наслідків та вибору найбільш оптимального варіанту.
10. Система повинна забезпечувати можливість надсилання автоматичних повідомлень різним службам та працівникам енергетичних структур.
11. Система повинна забезпечувати можливість відстеження виконання графіку погодинних відключень та розрахунку необхідної електроенергії для повного функціонування області.

1.5 Класифікація СППР

На сьогоднішній день існує різноманітні підходи до класифікації систем підтримки прийняття рішень (СППР), і немає єдиної загальної класифікації. Розглянемо деякі основні види класифікацій за різними характеристиками.[9]

На рівні користувача виділяють такі типи СППР:

- Активна СППР дозволяє користувачам зробити пропозицію щодо вибору оптимального рішення.
- Пасивна СППР допомагає користувачам у процесі ухвалення рішення, але не надає можливості внести пропозицію щодо вибору рішення.
- Кооперативні СППР дозволяють лідерам і керівникам змінювати, доповнювати або поліпшувати рішення, що пропонуються системою, та внести ці зміни для подальшої перевірки. [9]

На технічному рівні виділяють такі типи СППР:

- СППР рівня підприємства, яка підключена до великих сховищ даних та обслуговує багатьох менеджерів підприємства.
- Персональна настільна СППР, яка є малою системою, що обслуговує лише один комп'ютер користувача.

На концептуальному рівні виділяють такі типи СППР:

- Керована повідомленнями (Communication-Driven DSS) - система, що підтримує групу користувачів, що працюють над виконанням загальної задачі. [9]
- Керована даними (Data-Driven DSS, Data-oriented DSS) - система, що орієнтується на доступ та маніпуляції з даними. [9]
- Керована документами (Document-Driven DSS) - система, яка забезпечує пошук та маніпуляцію неструктурованою інформацією у різних форматах документів. [9]

- Керована знаннями (Knowledge-Driven DSS) - система, яка надає рішення на основі фактів, правил та процедур. [9]

Керована моделями (Model-Driven DSS) - система, яка забезпечує доступ та маніпуляцію з математичними моделями, такими як статистичні, фінансові, оптимізаційні та імітаційні моделі. [9]

1.6 Висновки

У першому розділі було проведено аналіз системи підтримки прийняття рішень в енергетичних системах. Було розглянуто основну інформацію за системами підтримки прийняття рішень, їх специфіку, види та особливості. Для зручного створення програми та бачення її серед конкурентів, було знайдено альтернативи, кращі якості яких ми порівняли між собою для глибокого аналізу, щоб створити максимально ефективний власний додаток, наповнивши його особливими компонентами, що відрізнятиме від інших робіт. Було розглянуто можливі області застосування розроблюваної системи, у наш час найактуальнішою є робота системи у екстремальних умовах, які і спонукали на розробку застосунку. Пошкодження інфраструктури та зміни в енергетичній системі України потребують максимально швидкого рішення для розв’язання проблеми з постачанням електроенергії і побудовою графіку погодинних відключень. Також було визначено функціональні та нефункціональні вимоги, а також основні до вхідних даних. Вхідні дані повинні мати мінімум трудомісткості, об’ємність, відповідність, різноманітність. Нефункціональні вимоги: безкоштовність, мультиплатформенність, зручність, безпечність відповідність та інші. А от функціональні містять всі вимоги до програмного забезпечення, які описують його внутрішню роботу, певною мірою поведінку.

2. ПРОЕКТУВАННЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В ЕНЕРГЕТИЧНИХ СИСТЕМАХ

2.1 Вибір найкращого варіанту для проектування СППР

Предметною областю роботи системи є графік погодинних відключень електроенергії. Під час здійсненням аналізу предметної області було визначено такі основні особливості:

1. Предметна область містить інформацію про області України та їх споживання електроенергії.
2. На вхід системи подається інформація про максимальну необхідну кількість споживання електроенергії та її реальне надходження.
3. Кожна область буде відображатися на графіку з трьома чергами по часу використання світла.
4. Надходження електроенергії на області є в межах [0, максимальна необхідна кількість споживання електроенергії]

Кожну область можна представити як вектор, який містить мінімум три важливих компоненти – назву, необхідну кількість електроенергії та отриману кількість електроенергії. Для зручної взаємодії системи можна записати їх у новий вектор загальної інформації. Тоді ці вектори будуть у вигляді матриці, де буде зберігатися основна інформація по областям. Дана система повинна допомагати у процесі ухвалення рішення, але не приймати його. На технічному рівні обслуговувати лише один комп’ютер користувача та на концептуальному рівні в основному орієнтуватися на доступ та маніпуляції з даними.

Взявши до уваги цей аналіз та пройдену попередню інформацію щодо СППР можна зробити висновок, що найкращим варіантом для вибору проектування є пасивна персональна настільна СППР з керованими даними.

2.2 Розробка механізму розрахунку часу погодинних відключень електроенергії

Для даної предметної області було розроблено власний механізм розрахунку часу з врахуванням необхідної та наданої кількості електроенергії на область. Варто зазначити, що потрібно сформувати графіки навантаження по всім 3-м чергам. Для зручності та рівномірного розподілення часу роботи системи, графік буде відображати час для чотирьох часових періодів (00:00-06:00, 06:00-12:00, 12:00-18:00, 18:00-00:00), зображено на рисунку 2.1

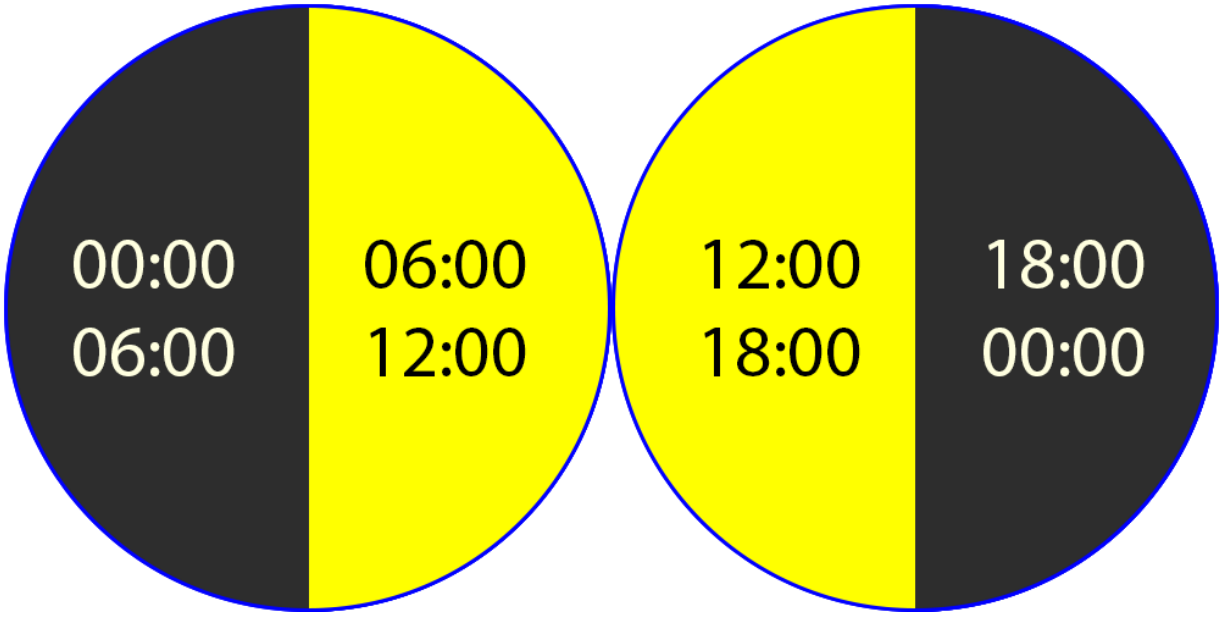


Рисунок 2.1 – Зображення доби, що розбита на 4 часові проміжки

Кожен з часових періодів має три черги. Черги повинні мати однаковий розподіл використання електроенергії та рівномірний за часом розподіл протягом кожного з часових періодів. Тому, черга під номером 1 буде розташована на початку кожного з часових періодів. Черга номер 2 між першою та третьою, а остання розташована у кінці, що можна спостерігати на рисунку 2.2

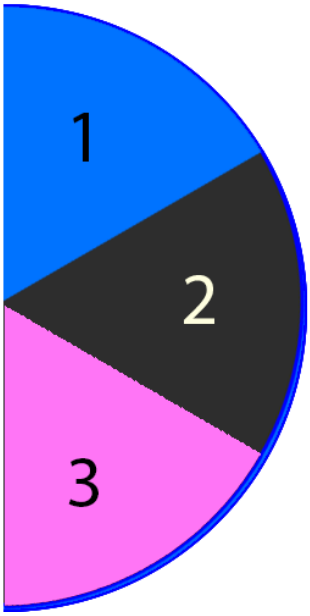


Рисунок 2.2 – Зображення ¼ доби, що розбита на 3 черги

Кожна черга являє собою певну групу населення області, які очікують на надходження електроенергії. Ці черги можуть перетинатися, якщо певна область має більше 33% живлення, при 100% вони повністю всі накладаються, так як час наданої електроенергії черги 1 рухається за годинниковою стрілкою, черги 3 проти годинникової стрілки та черги 2 у обидва напрями одночасно. За умови, якщо кількість електроенергії буде менша за 33% від максимальної, то будуть проміжки часу з повною відсутністю енергопостачання.

Для розрахунку часу надання електроенергії, який буде надаватися для області відповідно з кожного проміжку часу на кожну чергу використовується наступний алгоритм:

1. Розрахунок часу роботи області у хвилинах
2. Розподіл часу на години та хвилини
3. Розрахунок часу в залежності від черги та часового проміжку

Для знаходження активного часу роботи заданої області на період у хвилинах скористаємося формулою (2.1):

$$Time(NeedPower, Power) = \left(\frac{Power}{NeedPower}\right) * TimeH * TimeM * Part \quad (2.1)$$

де NeedPower – необхідна кількість електроенергії для цілодобової роботи області;

Power – надана потужність (кількість електроенергії);

TimeH – кількість годин у добі;

TimeM – кількість хвилин на годину;

Part – коефіцієнт, що складає четверту частину доби і має значення ¼.

Отримане значення буде використовуватися для подальших розрахунків за формулою (2.2) та (2.3), для точної кількості часу у годинах та хвилина, які можуть бути надані для використання світла.

$$WorkingH = \frac{Time(NeedPower,Power)}{60}, \quad (2.2)$$

$$WorkingM = Time(NeedPower, Power)\%60. \quad (2.3)$$

де WorkingH – час роботи у годинах черги області у певний часовий проміжок;

WorkingM – час роботи у хвилинах черги області у певний часовий проміжок.

Маючи цю інформацію можна розраховувати час надання електроенергії для першої черги. Для цього скористаємося формулою (2.4):

$$T(\text{WorkingH}, \text{WorkingM}) = (STH + \text{WorkingH}) : (\text{WorkingM}) \quad (2.4)$$

де T – часовий діапазон графіку роботи світла області у проміжок часу;

STH – Початковий час у годинах часового проміжку.

Для розрахунку часу для третьої черги використовується формула (2.5):

$$T(\text{WorkingH}, \text{WorkingM}) = (LTH - \text{WorkingH} - 1) : (60 - \text{WorkingM}) \quad (2.5)$$

де LTH – кінцевий час у годинах часового проміжку.

Так як час у хвилинах може сягати 0, то буде створений спеціальний метод, яких це буде перевіряти та корегувати час на годину вперед, так як формула буде зайвий раз віднімати годину.

Та найскладніший у реалізації це розрахунок часу для другої черги, так як вона знаходиться між іншими та при збільшенні енергопостачання повинене час збільшувати у два напрями. Це досягається механізмом зворотного поєднання часу до хвилин, знаходження його половини з повторним перетворенням до годин з хвилинами та застосування попередньо згаданих формул з врахуванням початкового та кінцевого часу, як середньоарифметичне між ними та їх використанням. Так, початок часу роботи світла визначається за формулою (2.6):

$$T() = \left(\frac{LTH+STH}{2} - \text{WorkingH} - 1\right) : (60 - \text{WorkingM}) \quad (2.6)$$

де T – початковий часовий діапазон графіку роботи світла області у проміжок часу;

WorkingH та WorkingM перераховані години та хвилини роботи черги.

Для знаходження кінцевого часу роботи світла у другій черзі за визначеним діапазоном скористаємося формулою (2.7):

$$T() = \left(\frac{LTH+STH}{2} + \text{WorkingH}\right) : (\text{WorkingM}) \quad (2.7)$$

де T – кінцевий часовий діапазон графіку роботи світла області у проміжок часу.

Детальніше можна переглянути алгоритм на рисунку 2.3

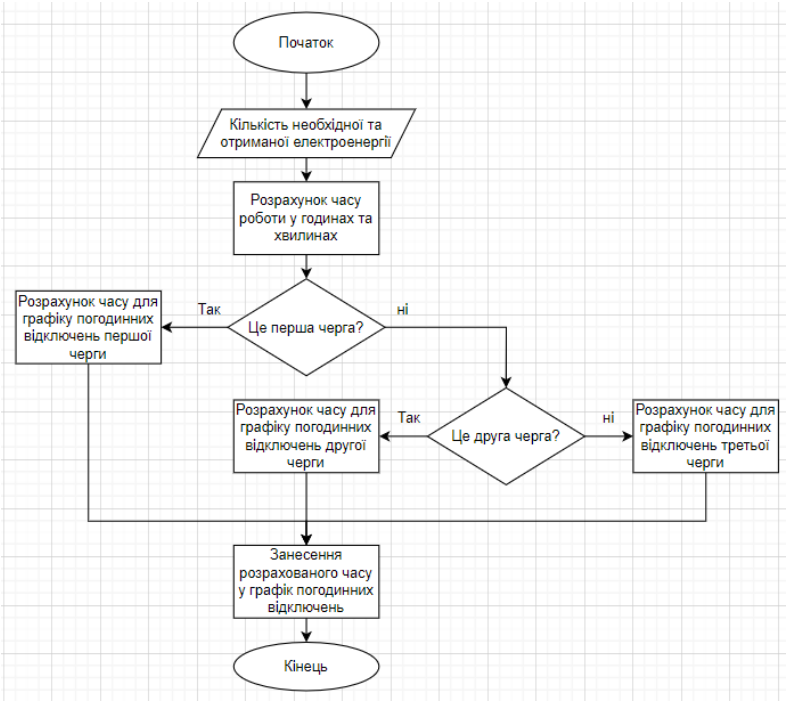


Рисунок 2.3 – Опис алгоритму розрахунку часу погодинних відключень електроенергії у вигляді блок-схеми

Нехай маємо для Київської області 2000 одиниць необхідної кількості електроенергії для споживання та 1000 одиниць реального мережевого надходження. Тоді час роботи області у хвилинах:

$$Time(NeedPower, Power) = \left(\frac{1000}{2000}\right) * 60 * 24 * \frac{1}{4} = 180$$

Робочі цілі години будуть розраховані як:

$$\text{WorkingH} = \frac{180}{60} = 3$$

Робочі хвилини можна знайти за формулою(2.3), або з математичних міркувань віднявши від кількості хвили всі цілі години, помноживши їх на 60.

$$\text{WorkingM} = 180\%60 = 0$$

Нехай визначення береться для першого часового проміжку для всіх трьох черг. Цей часовий проміжок є [00:00-06:00]. Тож для знаходження роботи першої черги початок цього проміжку береться як 00:00, а його закінчення розраховується:

$$T(\text{WorkingH}, \text{WorkingM}) = (0 + 3): (0) = 03: 00$$

У результаті графік роботи першої черги для часового проміжку 1 складатиме [00:00-03:00]. Знаходження часу роботи третьої черги відбувається:

$$T(\text{WorkingH}, \text{WorkingM}) = (6 - 3 - 1): (60 - 0) = 03: 00$$

Для цієї черги розраховується вже початок для роботи світла, так як його закінчення обмежується кінцем 1 часового діапазону. Тому маємо результат роботи в [03:00-06:00].

Найскладнішим у цьому буде розрахування часу для другої черги, яка повинна бути розташована рівно по центру між двома попередніми. ЇЇ початковий час для розрахунків є середнім арифметичним між початком та кінцем часового діапазону.

$$T() = \frac{(0+6)}{2} : \frac{(0+0)}{2} = 03{:}\,00$$

В результаті будемо мати [03:00-03:00], далі використовуємо за алгоритмом попередні формули (2.6) та (2.7) і отримуємо:

$$T(Start) = (3-1-1){:}\,(60-30) = 01{:}\,30$$

$$T(End) = (3+1){:}\,(30) = 04{:}\,30$$

Так, час роботи другої черги складатиме 3 години і працюватиме для першого часового проміжку в діапазоні [01:30-04:30].

Об’єднавши отримані результати ми отримаємо данні, що зображені у таблиці 2.1

Таблиця 2.1 Графік роботи світла області у перший часовий проміжок

Область	Черга	[00:00-06:00]
Київська	1	[00:00-03:00]
	2	[01:30-04:30]
	3	[03:00-06:00]

Отже, можемо будувати графік погодинних відключень електроенергії за розробленим механізмом розрахунку часу погодинних відключень електроенергії.

2.3 Розрахунок приблизного енергоспоживання електроенергії в Україні по кожній області за 2022 рік

Серед відкритих та точних джерел інформації було знайдено данні за 2014 рік з детальним зображенням на карті інформації про кількість енергоспоживання у млрд кіловат-год на кожну з областей. Внесені данні можна спостерігати у таблиці 2.2 в першому стовпчику.[7]

Так, у 2014 році сума споживання електроенергії країною дорівнює сумі споживання по кожній з областей, а це S1=134.6 млрд кіловат-год.

За офіційними даними у 2022 році використання електроенергії становить S2=153.7 млрд кіловат-год.[12]

Маючи цю інформацію та порівнюючи відому інформацію про виробництво електроенергії станом на 2022 рік можна з допомогою певних механізмів розрахувати приблизну кількість спожиття електроенергії по областях на сьогоднішній день.

Розрахуємо коефіцієнт збільшення споживання електроенергії для кожної з областей. Для цього достатньо використати формулу (4.1)

$$k = \frac{S1}{S2}, \tag{4.1}$$

де S1 – сума споживання електроенергії в 2014 році;

S1 – сума споживання електроенергії в 2022 році.

Так маємо, $k = \frac{134.6}{153.7} \sim 1.14$

До дамо в таблицю 4.1 нові перераховані дані з додатковим значенням споживання не тільки протягом року а й за годину. Для знаходження Мвт/год необхідно млрд кВт годин помножити на мільйон та поділити на 8760 (кількість годин у році).

Дана інформація буде використовуватися у подальшому як початкова необхідна кількість електроенергії на кожну з областей.

Таблиця 2.2 Енергоспоживання областей України

	млрд кВт год/рік (2014 рік)	*1,14 млрд кВт год / рік (2022рік)	МВт/год (2022р)
Вінницька	2,9	3,306	377,3972603
Волинська	1,6	1,824	208,2191781
Дніпропетровська	27,8	31,692	3617,808219
Донецька	18,1	20,634	2355,479452
Житомирська	2,6	2,964	338,3561644
Закарпатська	1,9	2,166	247,260274
Запорізька	8,9	10,146	1158,219178
Івано-Франківська	2,2	2,508	286,3013699
Київська	14,7	16,758	1913,013699
Кіровоградська	3,2	3,648	416,4383562
Луганська	7,8	8,892	1015,068493
Львівська	4,4	5,016	572,6027397
Миколаївська	3	3,42	390,4109589
Одеська	6,2	7,068	806,8493151
Полтавська	5,3	6,042	689,7260274
Рівненська	2,4	2,736	312,3287671
Сумська	2,2	2,508	286,3013699
Тернопільська	1,3	1,482	169,1780822
Харківська	7	7,98	910,9589041
Херсонська	2,4	2,736	312,3287671
Хмельницька	2,3	2,622	299,3150685
Черкаська	3,2	3,648	416,4383562
Чернівецька	1,3	1,482	169,1780822
Чернігівська	1,9	2,166	247,260274
	134,6	153,7	

2.4 Розробка архітектури

Спочатку спрогнозуємо основні варіанти використання системи підтримки прийняття рішень в енергетичних системах користувачем. Для роботи з системою користувач повинен мати можливість на перегляд та редагування початкових вхідних даних для генерації графіку погодинних відключень електроенергії, а також саму можливість генерації. Дана модель представлена на рисунку 2.4

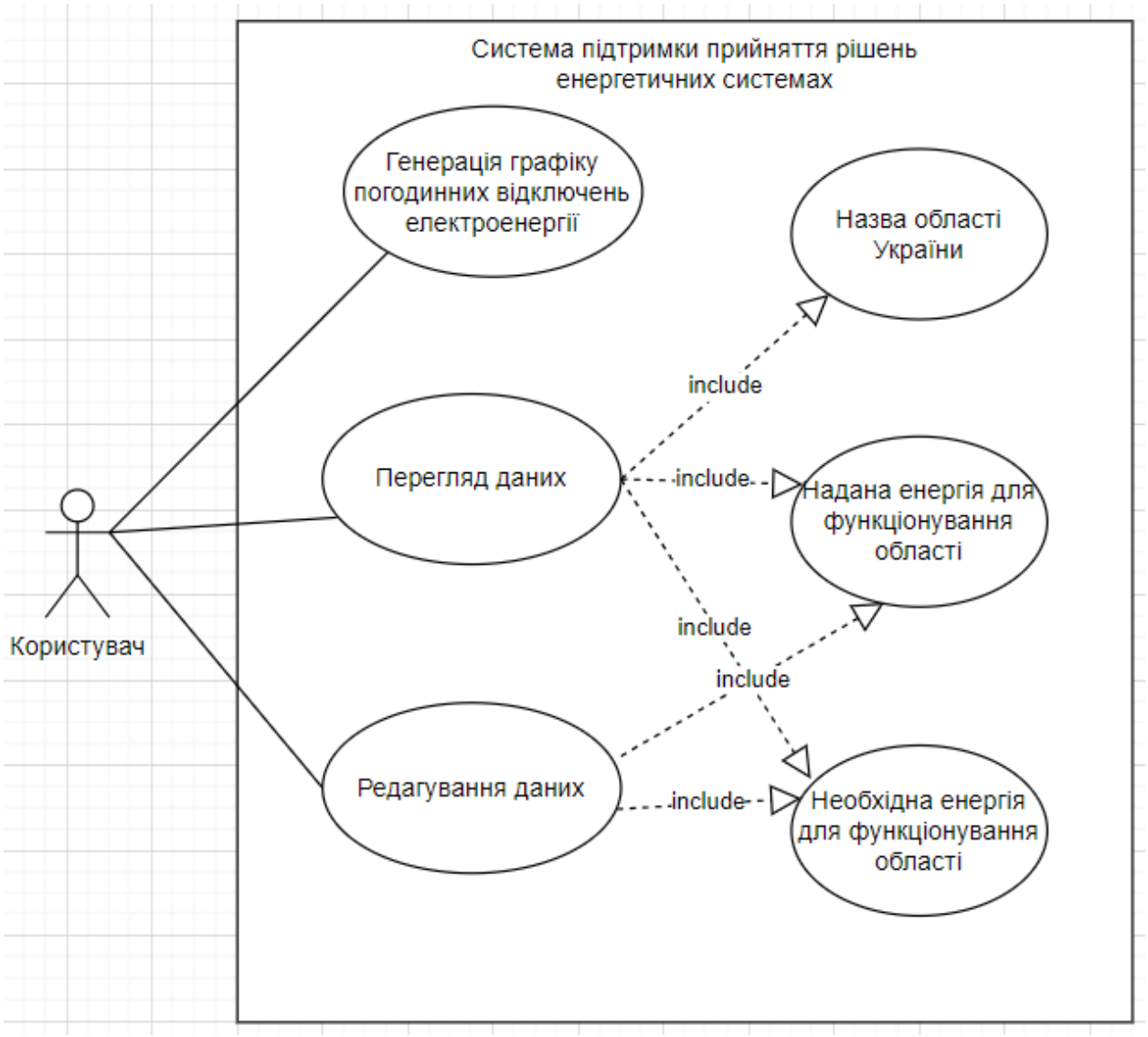


Рисунок 2.4 – Варіанти використання СППР енергетичних системах

З отриманої моделі варіантів використання можна визначити, що основною взаємодією користувача є перегляд даних, який включає в себе перегляд інформації про області України, надану та необхідну електроенергію для функціонування області. Редагування даних має лише можливість зміни параметрів наданої та необхідної кількості електроенергії. Та основною та важливою функцією використання є сама генерація графіку погодинних відключень електроенергії.

На основі варіантів використання СППР побудуємо дерево функцій (рисунок 2.5) та визначимо основні його частини.

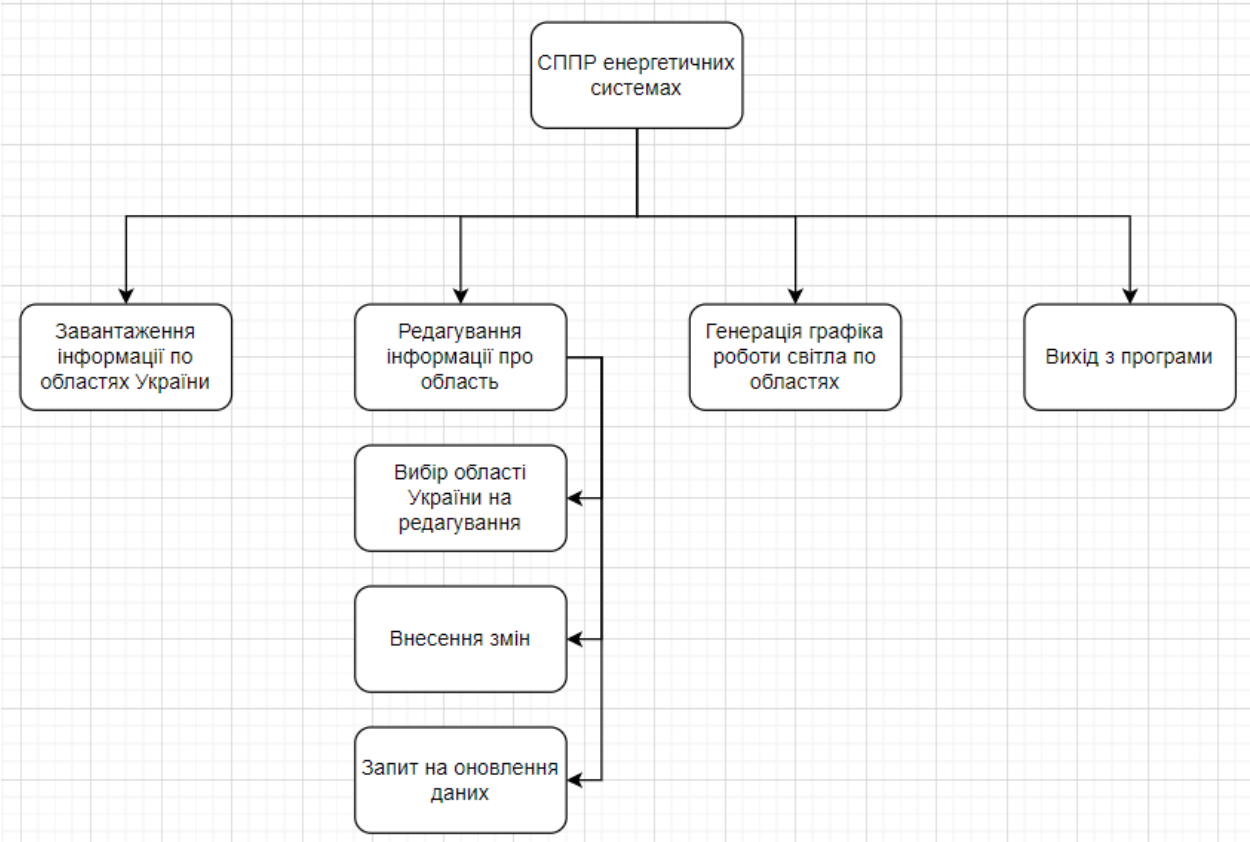


Рисунок 2.5 – Дерево функцій СППР енергетичних системах

Функція завантаження інформації по областях України наддасть користувачеві візуальне відображення стану про їх необхідну кількість електроенергії та рівень надходження. Цей процес буде відбуватися у отриманні з внутрішніх збережених даних програми інформації про області України та їх зображення у відповідному полі програми для можливості подальших маніпуляцій.

Редагування інформації про область відбувається за стандартним алгоритмом:

1. Використовуючи клавіші управління вибір області України на редагування
2. Внесення числових змін навантаження області
3. Запит на оновлення інформації

Коли процес досягає успіху ми отримуємо у завантаженій інформації про області України зміни щодо однієї з них. Якщо користувач введе данні з похибкою чи залишить поля пустими, програма повинна від сповістити про помилку.

Функція генерації графіка роботи світла по областях являється основним та заключним етапом роботи програми, що повинно вивести на екран додатку сформований результат, яким може користувач скопіювати та використовувати для власних потреб.

Графік повинен в себе включати назву області, всі три черги та чотири часові проміжки. Орієнтовне зображення на рисунку 2.6

Область	черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Волинська	1	00:00-02:00	06:00-08:00	12:00-14:00	18:00-20:00
	2	02:00-04:00	08:00-10:00	14:00-16:00	20:00-22:00
	3	04:00-06:00	10:00-12:00	16:00-18:00	22:00-00:00

Рисунок 2.6 – Прогнозоване зображення графіку погодинних відключень електроенергії по одній з областей України

Вихід з програми забезпечує безпечне закриття СППР енергетичних системах зі збереженням зміненої інформації та генерованого результату графіку роботи світла.

Для зображення всіх етапів діяльності додатку та зрозуміння, як вони пов'язані між собою скористаємося картою процесів. Головною метою карти процесів є визначення потреб у змінах і поліпшеннях.

Карта процесів надає змогу детально описати кожен окремий процес, включаючи всі етапи його виконання та взаємозв'язки з іншими процесами. Допомагає виявити недоліки та перешкоди, які заважають ефективному виконанню процесів, що дозволяє підприємству удосконалювати свою діяльність та зменшувати витрати.

Побудуємо карту процесів для СППР енергетичних системах та переглянемо її на рисунку 2.7

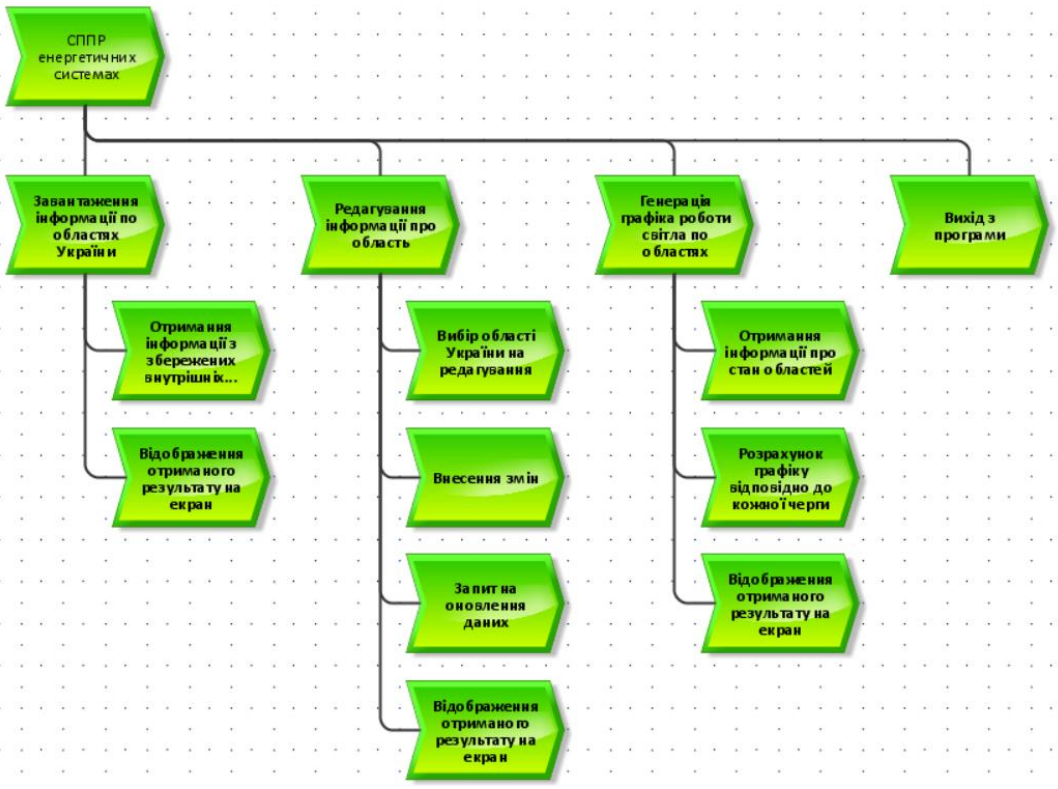


Рисунок 2.7 – Карта процесів СППР енергетичних системах

Дана карта процесів допомогла нам у візуалізації взаємозв'язків між різними кроками та етапами в процесі. Можна чітко спостерігати закономірність, що будь-яка дія користувача, окрім виходу з програми, закінчується візуальним відображенням результату, що до виконаної роботи користувачем. Це дозволяє у режимі реального часу максимально ефективно впливати на побудову графіку погодинних відключень електроенергії, так як ми бачимо відразу зміни.

2.5 Проектування СППР енергетичних системах за допомогою методології IDEF0 та DFD

Контекстна діаграма є однією з важливих моделей в процесі аналізу та проектування системи. Вона дозволяє описати систему на високому рівні, показуючи її взаємодію з зовнішніми системами та інтерфейсами. Контекстна діаграма є першим етапом в аналізі системи та надає загальне уявлення про її функціонування. Контекстну діаграму СППР можна побачити на рисунку 2.8

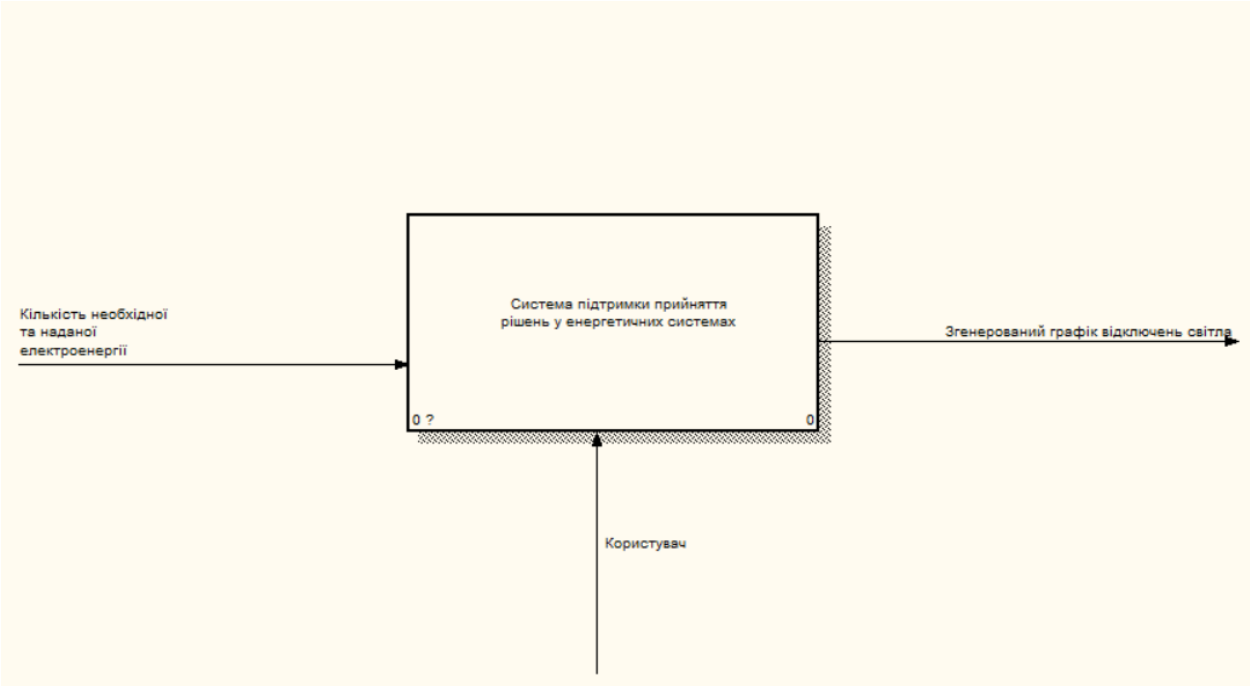


Рисунок 2.8 – Контекстна діаграма процесу " Система підтримки прийняття рішень в енергетичних системах"

У контекстній діаграмі процесу "Система підтримки прийняття рішень в енергетичних системах" показано взаємодію даної системи з користувача з системою, яка у результаті надає згенерований графік відключень світла по областях у 3-х чергах. На вхід ми отримуємо основну інформацію, яка збережена у самій програмі та яка може бути редагованою користувачем, при необхідності в оновлені інформації. Детальніше можна переглянути процеси роботи програми при декомпозиції контекстної діаграми на рисунку 2.9

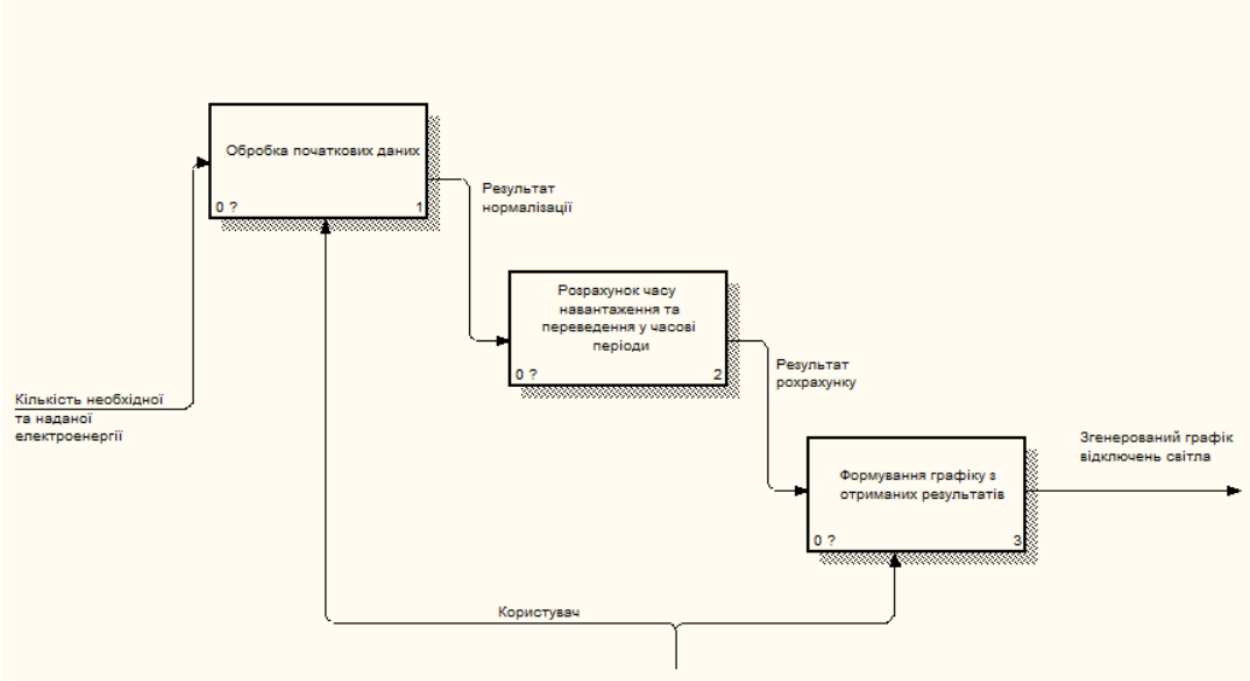


Рисунок 2.9 – Декомпозиція контекстної діаграми процесу "Система підтримки прийняття рішень енергетичних системах"

Розглянемо докладніше кожен процес у вигляді DFD діаграми. Для отримання нормалізованих даних внутрішніх збережень, необхідних для роботи системи підтримки прийняття рішень в енергетичних системах, спочатку відбирається інформація про максимальну кількість використаної електроенергії на область. Далі, на основі цієї інформації, отримується інформація про надані квоти у попередньому використанні. Після ініціалізації процесу, всі дані потрапляють у меню редагування в реальному часі, де

користувач може змінювати моделі та інформацію, яка його цікавить. Після підтвердження дій користувача, дані нормалізуються та зберігаються для подальшого використання системою. Описану декомпозицію процесу "Обробка початкових даних" можна переглянути на рисунку 2.10

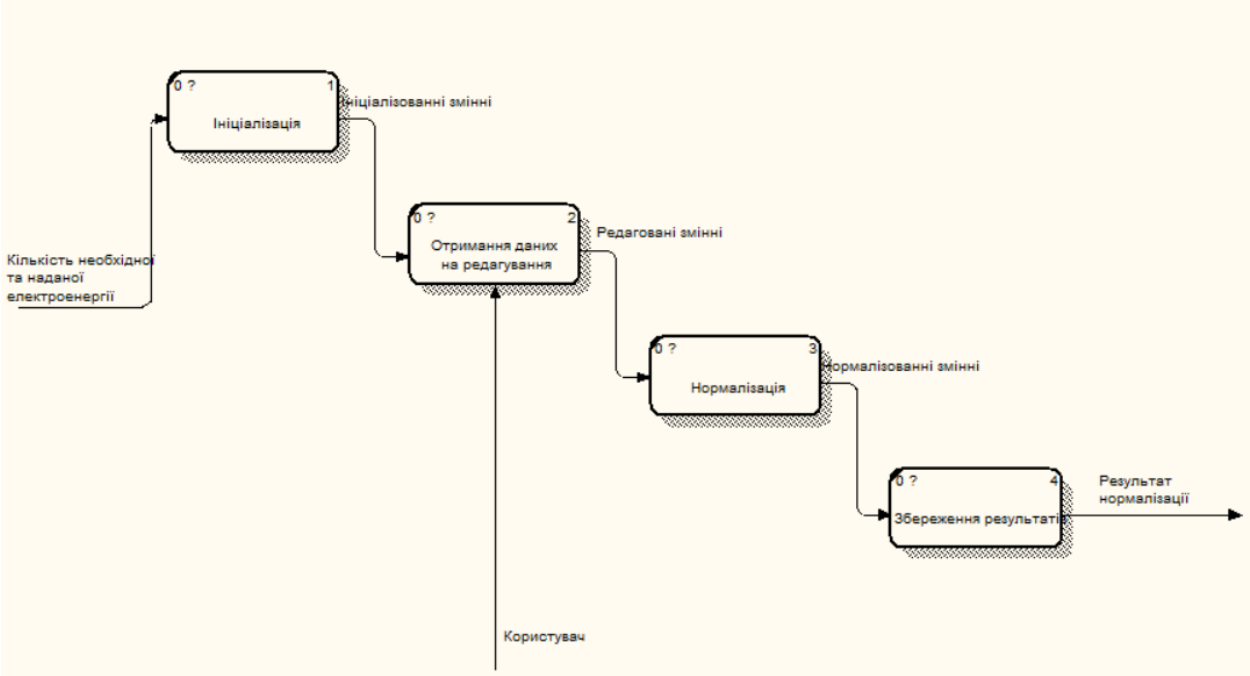


Рисунок 2.10 – Декомпозиція процесу "Обробка початкових даних"

Процес розрахунку часу навантаження та переведення у часові періоди є важливим етапом у створенні графіку погодинних відключень електроенергії. Цей процес включає в себе ряд дій, що допомагають сформувати часові періоди, необхідні для фінального етапу генерації графіку. Спочатку з нормалізованої інформації ми визначаємо навантаження, які потрібні для кожної області. Потім ці навантаження перетворюються на часові періоди за допомогою спеціального блоку обробки даних.. Ці періоди зберігаються для подальшого використання у фінальній генерації графіку погодинного відключення електроенергії. Декомпозицію цього процесу ретельніше можна переглянути на рисунку 2.11

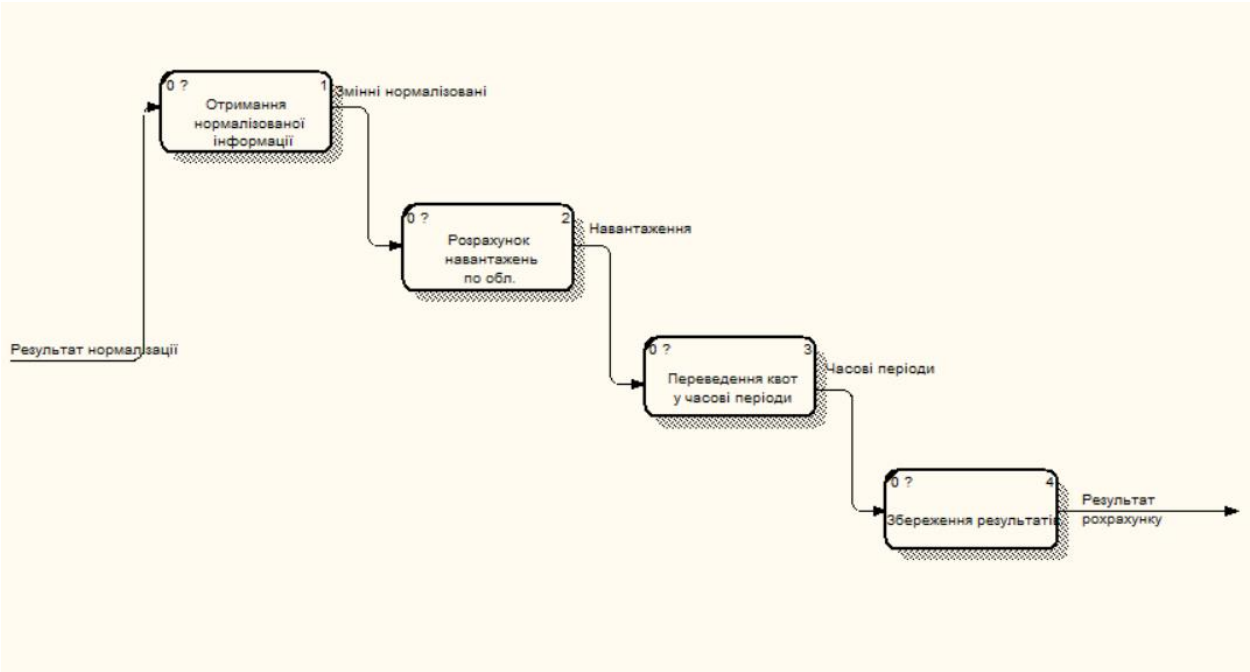


Рисунок 2.11 – Декомпозиція процесу "Розрахунок часу навантаження та переведення у часові періоди"

Останнім етапом виконання системи є процес формування графіку з отриманих результатів, який можна відобразити на екрані. Для досягнення цієї мети ми використовуємо часові періоди, що були отримані на попередньому етапі. На вході графіку вказуються параметри для визначення розкладу погодинних відключень, алгоритм побудови якого розраховується на основі цих параметрів та часових періодів.

Отриманий графік подається на фінальний етап формування, де він піддається обробці та візуалізації. У цьому етапі графік приймає остаточний вигляд, який можна відобразити на екрані для подальшого використання. Декомпозицію процесу «Формування графіку з отриманих результатів» переглянемо на рисунку 2.12

Необхідно зазначити, що процес формування графіку погодинних відключень є важливим етапом в енергетичних системах, оскільки дозволяє ефективно розподіляти енергоресурси та зменшувати витрати. Таким чином, система підтримки прийняття рішень в енергетичних системах, зокрема

процес формування графіку погодинних відключень, є невід'ємною складовою у забезпеченні стабільності та ефективності енергетичного сектору.

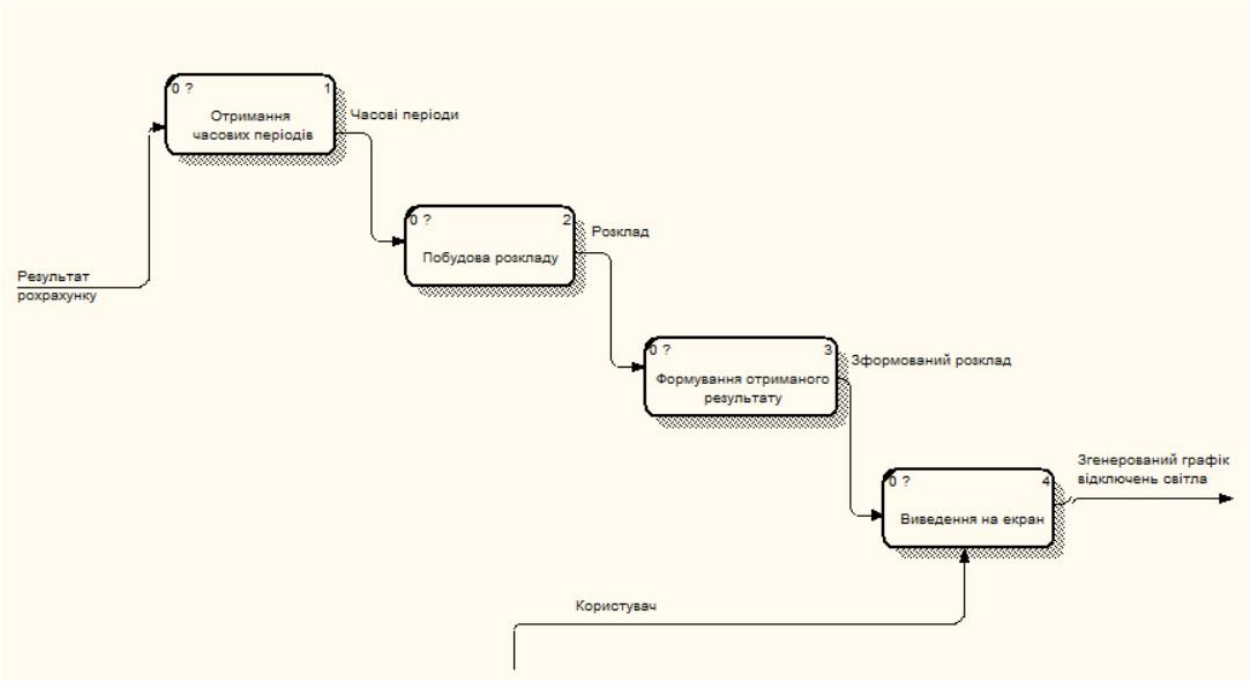


Рисунок 2.12 – Декомпозиція процесу «Формування графіку з отриманих результатів»

Дана модель є надійним інструментом для аналізу та підтримки прийняття рішень в енергетичній системі. Вона забезпечує користувача широким спектром можливостей, що дозволяє редагувати інформацію, створювати симуляції подій, та навіть прогнозувати майбутні проблеми. Це робить систему універсальною та ефективною, зокрема, у вирішенні складних завдань.

Проте, для поліпшення роботи системи, необхідно врахувати деякі фактори. Наприклад, важливо додати можливість автоматичного реагування на пошкодження важливих енергетичних структур. Це дозволить системі надійно захистити систему від випадкових помилок та негараздів, що може допомогти зберегти електроенергію та запобігти виникненню серйозних проблем.

Крім того, систему також можна поліпшити, додавши нові алгоритми та методи аналізу даних, що дозволить забезпечити ще більш точний та ефективний аналіз інформації. Такі оновлення покращать швидкість генерації графіку погодинних відключень та дозволять найефективніше розподіляти електроенергію у той непростий час.

В цілому, дана модель є важливим кроком у розвитку енергетичної системи та може стати важливим інструментом для вирішення складних проблем та підвищення ефективності енергетичної інфраструктури.

2.6 Проектування інтерфейсу користувача СППР ЕС

При проектуванні графічної складової важливо створити програму максимально зручною у використанні. Всі елементи повинні бути доступними на головному екрані та додаткових. Початковим набором для побудови UI було 11 компонентів. Зображення набору розглянемо на рисунку 2.13

Рисунок 2.13 – Графічне представлення компонентів проекту

Кожен компонент відігравав важливу роль. Слід зазначити, що хоч ми і маємо дві області для виведення інформації на екран, було задіяно ще і для

цього поля введення. Поля введення напряму взаємодіють з користувачем, надаючи йому можливості вводити інформацію, яка у майбутньому буде змінювати параметри системи. Для поліпшення роботи користувач може натиснути клавішу автозаповнення, що наповнить поля введення необхідною інформацією і користувачу буде достатньо вести йому потрібну інформацію, не витрачаючи час на загальну структуру. Первинне зображення інтерфейсу можна переглянути на рисунку 2.14, а на рисунку 2.15 зображення інтерфейсу з визначенням областей для взаємодії з СППР

Рисунок 2.14 – Первинне зображення інтерфейсу

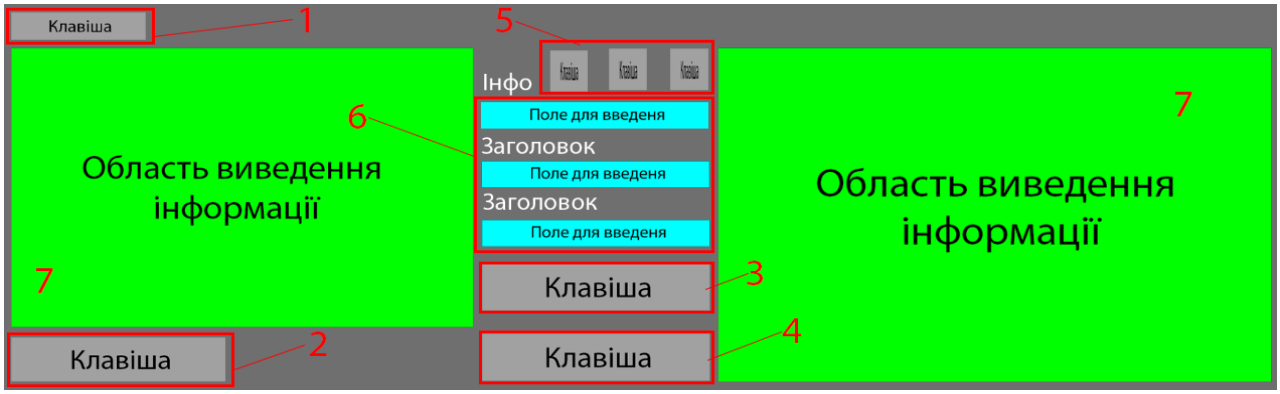


Рисунок 2.15 – Області взаємодії користувача СППР, що виділені на інтерфейсі програми

Важливо провести інтерпретацію розташованих об’єктів:

1. Вихід з програми
2. Заповнення області 7(зліва) інформацією про області України
3. Дозволяє оновлювати значення що до області, що вказана у 6(1 поле),

використовуючи значення у полях 6(2,3).

4. Дозволяє генерувати графік погодинних відключень, який буде зображено в області 7(з права)
5. Дана область містить 3 клавіші для зручного управління областями, що зображені в 7(зліва)
    1. Автоматично встановлює на перший елемент списку областей, заповнює всі поля у 6
    2. Дозволяє рухатися у «лівому» напрямку списку областей, також заповнює поля
    3. Дозволяє рухатися у «правому» напрямку списку областей, також заповнює поля
6. Область, що містить три поля для введення інформації. Ці поля також можуть виводити інформацію, яку можна скопіювати.
7. Області виведення інформації на екран

У програмуванні часто буває, що при роботі програми виникає велика кількість різних помилок. Одні з них критичні, інші – ні, і дозволяють без проблем працювати у подальшому програмі. Так, було вирішено створити додаткове діалогове вікно, котре буду відображати помилку та інформацію про неї, яке можна переглянути на рисунку 2.16

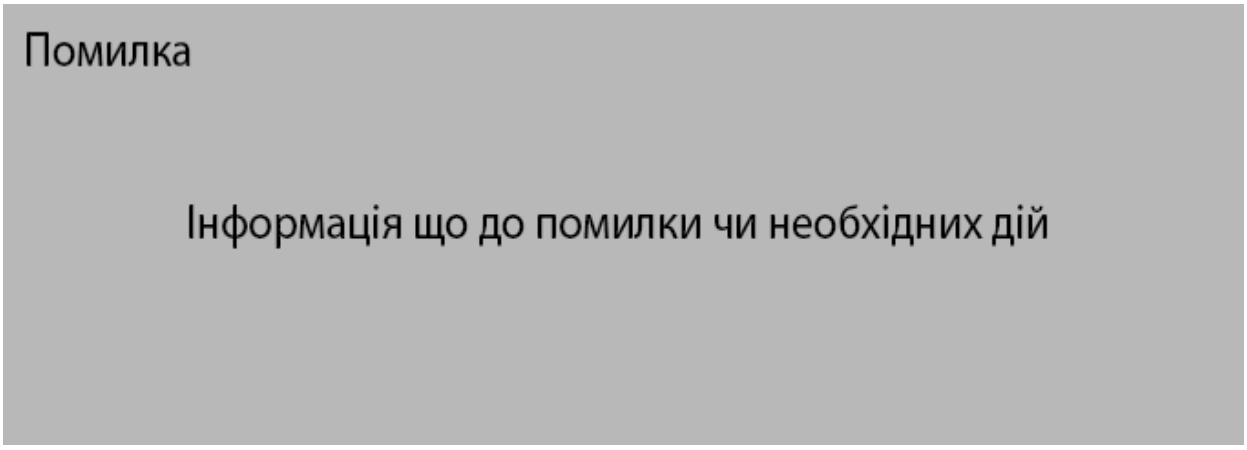


Рисунок 2.16 – Графічне представлення помилки

2.7 Висновки

Завдяки класифікації СППР було вибрано найкращим варіантом для вибору проектування це пасивна персональна настільна СППР з керованими даними. Було розроблено власний механізм розрахунку часу для кожного часового проміжку з врахуванням черг. Було вибрано мову програмування, середовище розробки та один з фреймворків для графічного представлення програмного застосунку. Було проаналізовано та розраховано приблизну кількість споживання електроенергії Україною в 2022 році, що буде використовуватися у системі в якості початкових налаштувань. Підчас розробки архітектури СППР було створено варіанти використання системи, побудовано дерево функцій та карту процесів. Також було для аналізу процесу створення системи підтримки прийняття рішення використано методологію IDEF0 та DFD. Було спроектовано графічну складову програми з всіма активними клавішами та полями виводу інформації. Оброблено помилку вводу інформації, якщо одне із полів є пустим.

3. РЕАЛІЗАЦІЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В ЕНЕРГЕТИЧНИХ СИСТЕМАХ

3.1 Вибрані інструменти для реалізації

Враховуючи постановку завдання, необхідно створити програмну реалізацію СППР. Було вибрано мову програмування C++ для досягнення цієї цілі.

Ця мова програмування є потужною високорівневою мовою з підтримкою декількох парадигм програмування, зокрема об'єктно-орієнтованої та процедурної парадигм. Вона надає розробникам гнучкість і можливість використовувати різні стилі програмування в залежності від потреб проекту. [10]

Об'єктно-орієнтована парадигма дозволяє створювати класи, об'єкти та взаємодіяти з ними, забезпечуючи структурованість та модульність коду. Ви можете використовувати успадкування, поліморфізм та інші концепції об'єктно-орієнтованого програмування для організації коду в логічні блоки.

Процедурна парадигма дозволяє описувати послідовність дій та алгоритми, розділяючи код на функції або процедури. Це дозволяє легше структурувати та управляти кодом, особливо для великих проектів.

Завдяки цим парадигмам, C++ мова програмування надає розробникам багато можливостей і гнучкість при реалізації різноманітних проектів різного масштабу і складності.

Для роботи над графічним інтерфейсом з використанням мови програмування c++ у середовищі розробки Microsoft Visual Studio було вибрано Microsoft Windows Forms. Це однин із фреймворків, які доступні в Visual Studio для розробки графічних інтерфейсів користувача (GUI) для

програм, призначених для операційної системи Windows. Він є частиною платформи .NET Framework і надає розробникам зручні інструменти для створення віконних додатків з використанням різноманітних елементів управління та функціональності. [11]

Особливості Microsoft Windows Forms:

- Розширена набір елементів управління: Windows Forms надає багатий вибір готових елементів управління, таких як кнопки, тексти, списки, таблиці, вкладки і багато інших. Ці елементи можуть бути легко розташовані на формі і настроєні за допомогою властивостей.
- Події і обробники подій: Windows Forms дозволяє реагувати на різні події, такі як натискання кнопки чи зміна значення поля введення. Розробники можуть призначати обробники подій для взаємодії з користувачем або здійснення певних дій.
- Простота використання і швидкість розробки: Завдяки візуальному дизайнеру в Visual Studio, Windows Forms дозволяє швидко створювати і налаштовувати інтерфейс користувача. Розробники можуть перетягувати і розміщати елементи управління на формі, встановлювати їх властивості та додавати логіку до подій без необхідності вручну писати весь код.

Розширені можливості налаштування: Windows Forms дозволяє встановлювати різні властивості для елементів управління, такі як розмір, колір, шрифт, положення і багато інших. Крім того, розробники можуть створювати власні елементи управління, розширюючи функціональність.

3.2 Структурна схема СППР енергетичних системах

Для відображення розробленої системи підтримки прийняття рішень енергетичних системах було побудовано структурну схему, яка допоможе побачити взаємодію між модулями програми і зображена на рисунку 3.1

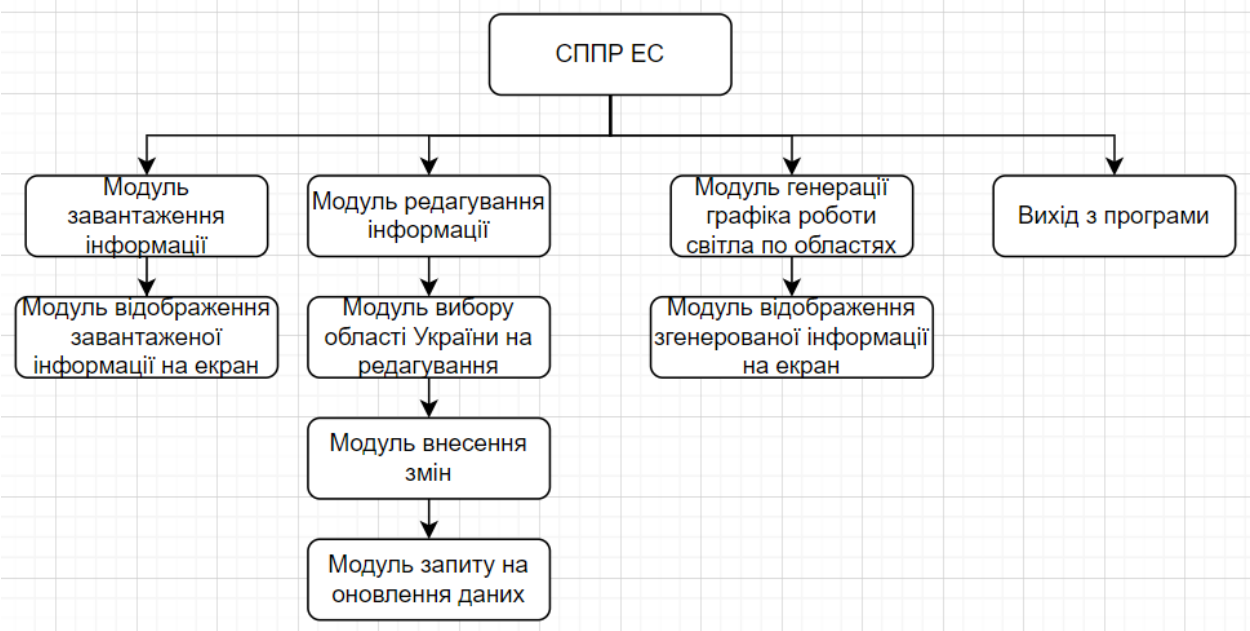


Рисунок 3.1 – Структурна схема СППР енергетичних системах

Розглянемо реалізацію основних модулів роботи СППР.

Модуль завантаження інформації передбачає генерацію таблиці для відображення інформації і саме занесення завантажених даних. Завантаження інформації по областях України відбувається після натиску на клавішу «Завантажити».

Це в свою чергу викликає внутрішній метод модулю відображення інформації, який під завантажує з системи програми інформацію про області України у щойно з генеровану таблицю та зображує її на екрані користувача.

Модуль редагування інформації передбачає взаємодію з областями. Для цього потрібно встановити початковий маркер на першу з областей, цього можна досягнути натиском на клавішу «First», яка запустить для цього відповідний метод.

Спостерігаємо після встановлення додаткову функцію, що перетворює виведену інформацію у полях користувача для взаємодії. Інформація була перетворена з представлення чисел з крапкою у представленні з комою, для зручного розуміння звичайним користувачем. Код цього процесу має вигляд:

Модуль запиту на оновлення даних відбувається після введення необхідної інформації і натиску на клавішу «Внести зміни», але спочатку відбувається перевірка на те, чи введена інформація не є пустою. Якщо ніяких проблем не виявлено відбувається оновлення інформації з повторним запуском методу, який був описаний раніше і оновлює інформацію у таблиці для користувача.

Модуль генерації графіку відключень електроенергії відбувається після натиску на клавішу «Розрахувати графік». Спочатку згенерується таблиця у яку буде заноситися вся інформація про часові проміжки роботи для кожної з областей та відповідних черг. А далі відбувається сам процес розрахунку часу погодинних відключень, код якого можна переглянути у додатку. Відображення отриманого результату відбувається автоматично на екрані користувача і не потребує додаткових дій з його сторони.

Вихід з програми можна здійснити за стандартним методом, використовуючи відповідну запрограмовану для цього клавішу або же скористатися хрестиком, який є у більшості програм справа зверху у куті.

3.3 Керівництво користувача СППР енергетичних систем

Для комфортного використання програми користувачу необхідно слідувати звичайній послідовності дій. Спочатку, щоб відкрити програму, потрібно здійснити подвійний натиск миші на зображення розробленого додатку. Це запустить саму програму та наддасть змогу користувачу взаємодіяти з системою. Графічний інтерфейс програми можна спостерігати на рисунку 3.2

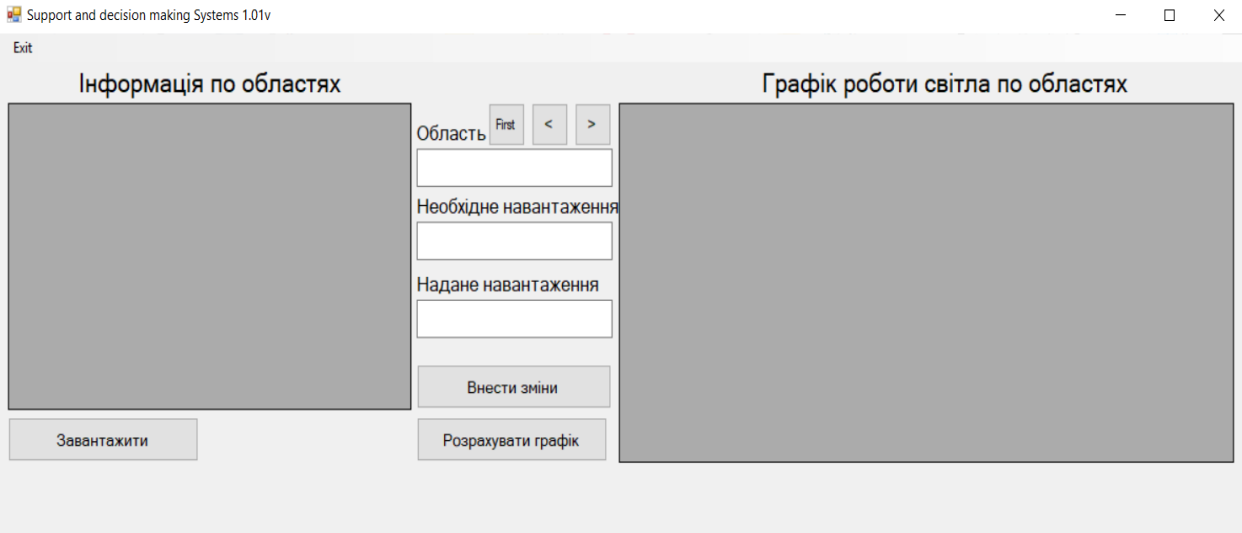


Рисунок 3.2 – Графічний інтерфейс програми

Для завантаження інформації про стан областей, їх необхідну кількість електроенергії та надану, потрібно натиснути на клавішу завантажити, що можна спостерігати на рисунку 3.3

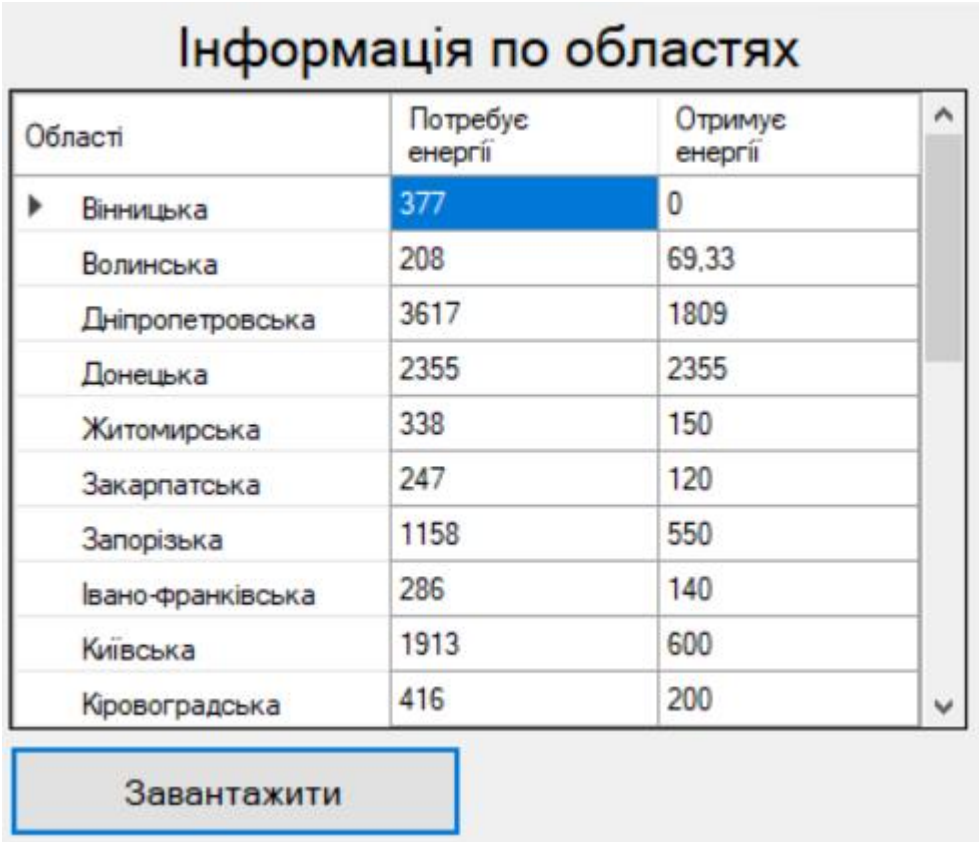


Рисунок 3.3 – Детальна інформація по областях України

Для взаємодії з областями використовується додаткова панель з трьома вхідними полями та чотирма клавішами, три з яких відповідають за переміщення у списку областей та четверта за оновлення інформації у вибраній області. Переглянути можна на рисунку 3.4

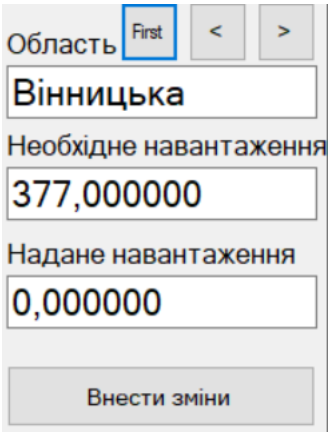
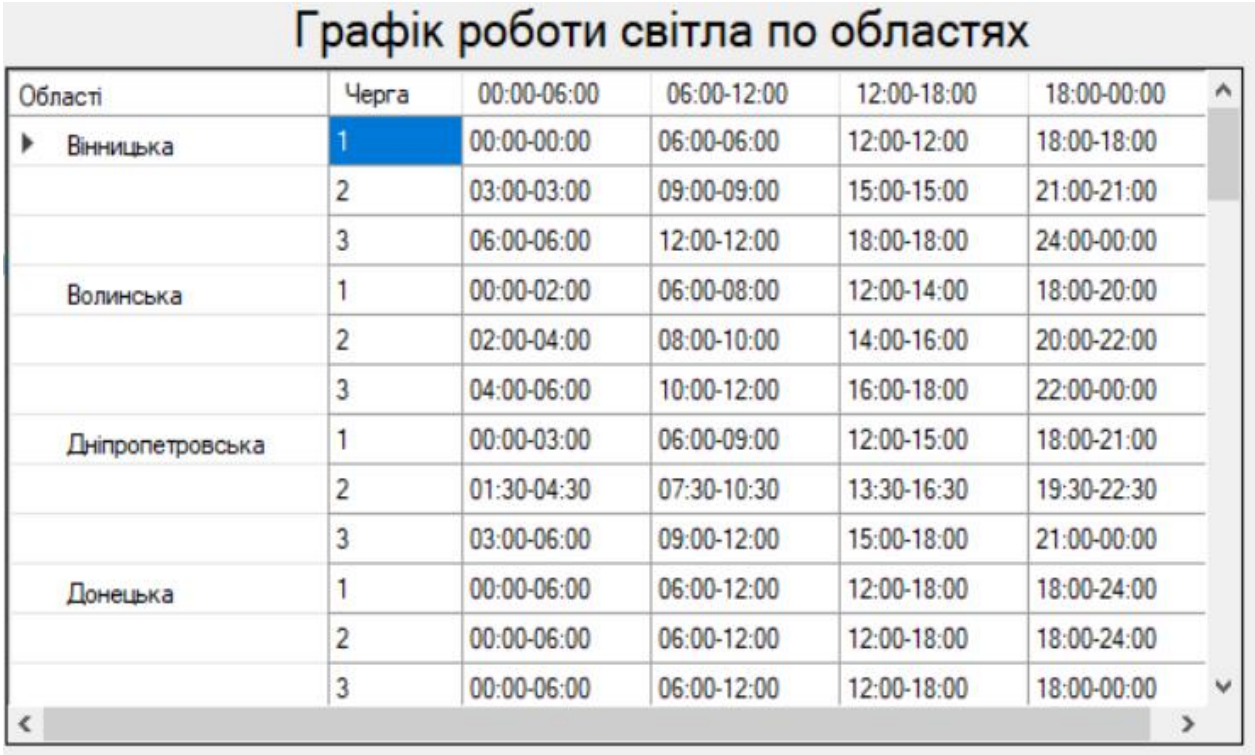


Рисунок 3.4 – Область для редагування параметрів областей України

Кінцевим результатом нашої програми є графік погодинних відключень електроенергії, який можна згенерувати натиснувши на клавішу «Розрахувати графік», який зображено на рисунку 3.5



Області	Черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Вінницька	1	00:00-00:00	06:00-06:00	12:00-12:00	18:00-18:00
	2	03:00-03:00	09:00-09:00	15:00-15:00	21:00-21:00
	3	06:00-06:00	12:00-12:00	18:00-18:00	24:00-00:00
Волинська	1	00:00-02:00	06:00-08:00	12:00-14:00	18:00-20:00
	2	02:00-04:00	08:00-10:00	14:00-16:00	20:00-22:00
	3	04:00-06:00	10:00-12:00	16:00-18:00	22:00-00:00
Дніпропетровська	1	00:00-03:00	06:00-09:00	12:00-15:00	18:00-21:00
	2	01:30-04:30	07:30-10:30	13:30-16:30	19:30-22:30
	3	03:00-06:00	09:00-12:00	15:00-18:00	21:00-00:00
Донецька	1	00:00-06:00	06:00-12:00	12:00-18:00	18:00-24:00
	2	00:00-06:00	06:00-12:00	12:00-18:00	18:00-24:00
	3	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00

Рисунок 3.5 –Виведення таблиці з графіком роботи світла по областях

Здійснюється також обробка первинних помилок, яка реагує на пусті введення полів з інформацією. Повідомлення про помилку можна спостерігати на рисунку 3.6

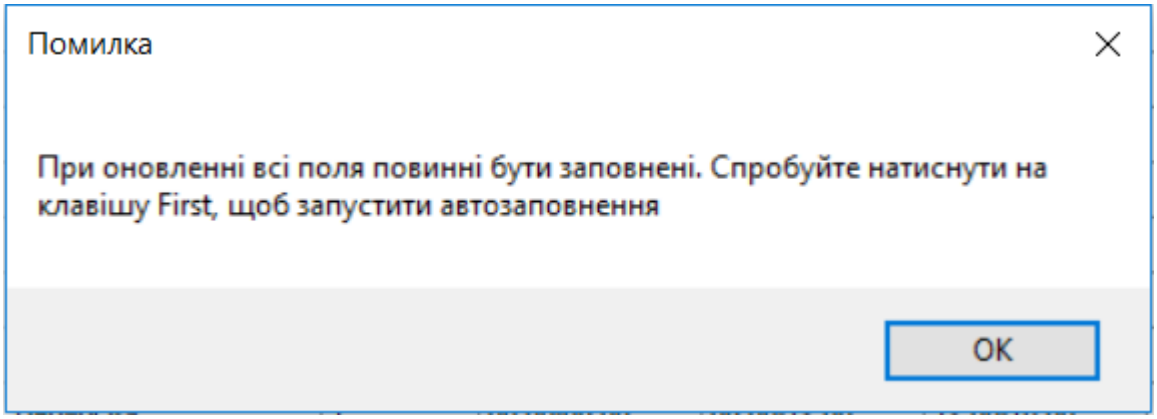


Рисунок 3.6 – Повідомлення про помилку

3.4 Тестування роботи СППР енергетичних систем для генерації графіку погодинних відключень електроенергії

Перевірку здійснимо для всього набору даних. Спочатку перевіримо можливість змінювати параметри у вже існуючих областях та на їх основі будувати новий графік. З Рисунку 3.3 змінимо параметри для перших 4 вхідних областей та побудуємо графік. Отримаємо результат, який можна спостерігати на рисунку 3.7

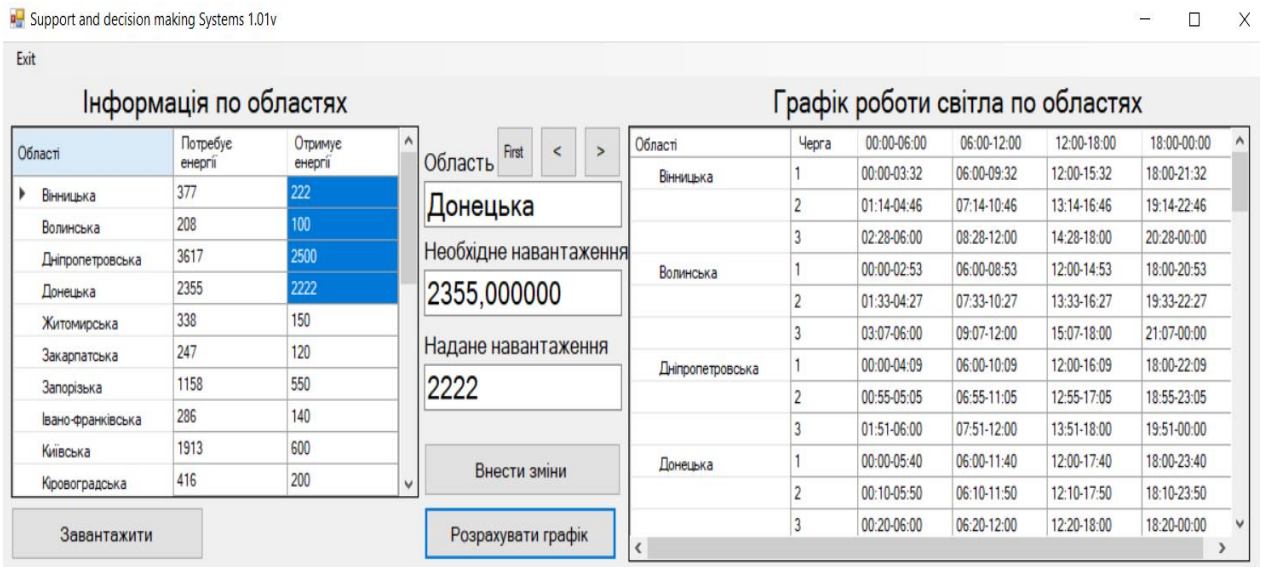


Рисунок 3.7 – Результат роботи програми

Перевіримо результати використовуючи параметри Волинської області. Час роботи у хвилинах рівний int(100/208*24*60)=692, так як час ми розбиваємо на 4 часових проміжки, то маємо 692/4=173 хв, що складає 2 години 53 хвилини на один проміжок. Це означає, що для Черги 1 Волинської області час роботи у перший проміжок від 00:00 до 02:53, Що чітко співпадає з отриманими результатами. Перевіряючи і інші параметри, результат аналогічний.

Тепер перевіримо всі унікальні випадки при розподілу електроенергії та за їх умови побудови графіків.

Нехай Вінницька область потребує 377 МВт енергії та отримує 0, тоді прогнозований графік матиме вигляд, як на таблиці 3.1

Таблиця 3.1 Прогнозований графік роботи світла у Вінницькій області за умов нульового отримання електроенергії

Область	черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Вінницька	1	00:00-00:00	06:00-06:00	12:00-12:00	18:00-18:00
	2	03:00-03:00	09:00-09:00	15:00-15:00	21:00-21:00
	3	06:00-06:00	12:00-12:00	18:00-18:00	24:00-00:00

Отриманий результат роботи програми за заданих умов можна переглянути на рисунку 3.8

Області	Черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Вінницька	1	00:00-00:00	06:00-06:00	12:00-12:00	18:00-18:00
	2	03:00-03:00	09:00-09:00	15:00-15:00	21:00-21:00
	3	06:00-06:00	12:00-12:00	18:00-18:00	24:00-00:00

Рисунок 3.8 - Результат генерації графіку за умов (377,0)

Нехай Волинська область потребує 208 МВт енергії та отримує 69.33, тоді прогнозований графік матиме вигляд, як на таблиці 3.2

Таблиця 3.2 Прогнозований графік роботи світла у Волинській області за умов 69.33 МВт одиниць отримання електроенергії

Область	черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Волинська	1	00:00-02:00	06:00-08:00	12:00-14:00	18:00-20:00
	2	02:00-04:00	08:00-10:00	14:00-16:00	20:00-22:00
	3	04:00-06:00	10:00-12:00	16:00-18:00	22:00-00:00

Отриманий результат роботи програми за заданих умов можна переглянути на рисунку 3.9

Волинська	1	00:00-02:00	06:00-08:00	12:00-14:00	18:00-20:00
	2	02:00-04:00	08:00-10:00	14:00-16:00	20:00-22:00
	3	04:00-06:00	10:00-12:00	16:00-18:00	22:00-00:00

Рисунок 3.9 - Результат генерації графіку за умов (208, 69.33)

Нехай Дніпропетровська область потребує 3617 МВт енергії та отримує 1809, тоді прогнозований графік матиме вигляд, як на таблиці 3.3

Таблиця 3.3 Прогнозований графік роботи світла у Дніпропетровській області за умов 1809 МВт одиниць отримання електроенергії

Область	черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Дніпро-петровська	1	00:00-03:00	06:00-09:00	12:00-15:00	18:00-21:00
	2	01:30-04:30	07:30-10:30	13:30-16:30	19:30-22:30
	3	03:00-06:00	09:00-12:00	15:00-18:00	21:00-00:00

Отриманий результат роботи програми за заданих умов можна переглянути на рисунку 3.10

Дніпропетровська	1	00:00-03:00	06:00-09:00	12:00-15:00	18:00-21:00
	2	01:30-04:30	07:30-10:30	13:30-16:30	19:30-22:30
	3	03:00-06:00	09:00-12:00	15:00-18:00	21:00-00:00

Рисунок 3.10 - Результат генерації графіку за умов (3617, 1809)

Нехай Донецька область потребує 2355 МВт енергії та отримує 2355, тоді прогнозований графік матиме вигляд, як на таблиці 3.4

Таблиця 3.4 Прогнозований графік роботи світла у Донецькій області за умов 2355 МВт одиниць отримання електроенергії

Область	черга	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
Донецька	1	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
	2	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00
	3	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00

Отриманий результат роботи програми за заданих умов можна переглянути на рисунку 3.11

Донецька	1	00:00-06:00	06:00-12:00	12:00-18:00	18:00-24:00
	2	00:00-06:00	06:00-12:00	12:00-18:00	18:00-24:00
	3	00:00-06:00	06:00-12:00	12:00-18:00	18:00-00:00

Рисунок 3.11 - Результат генерації графіку за умов (2355, 2355)

В ході тестування роботи СППР було виявлено повну відповідність генерації прогнозованого графіку погодинних відключень електроенергії до того, що був створений вручну. Це говорить про правильну побудову та реалізацію СППР енергетичних системах, що добре виконує свою поставлену задачу.

Висновки

У рамках дипломної роботи була проведена аналітична робота з аналізу системи підтримки прийняття рішень в енергетичних системах України. В результаті проведеного аналізу було виявлено проблеми, пов'язані зі складністю процесів прийняття рішень в енергетичній галузі, в тому числі і з плануванням відключень електроенергії. Для вирішення цих проблем була розроблена система підтримки прийняття рішень, яка базується на аналізі даних про відключення електроенергії та прогнозуванні їх можливих наслідків. Було здійснено класифікацію СППР та розроблено власний метод розрахунку часу для формування графіку. Система включає в себе дерево функцій та карту процесів, які допомагають орієнтуватися в складних процесах планування відключень та прийняття рішень. Результатом роботи стало створення програмного продукту з графічним інтерфейсом, який дозволяє зручно та ефективно проводити планування відключень та аналізувати їх наслідки. Впровадження системи підтримки прийняття рішень в енергетичній галузі України виявиться корисним кроком для поліпшення безпеки та надійності роботи енергетичних систем. Ця система допоможе зменшити ризик виникнення аварій і покращити ефективність управління енергосистемами. Розробка такої системи є важливим завданням, яке вирішує важливі проблеми в енергетичній галузі, сприяючи безпечному та ефективному функціонуванню енергетичних систем.

Список використаних джерел

1. П. І. Бідюк, О. Л. Тимощук, А. Є. Коваленко, Л. О. Коршевнюк // Система підтримки прийняття рішень// Затверджено Вченою радою КПІ ім. Ігоря Сікорського як підручник - 2022.С.610

2. Системи підтримки прийняття рішень [Текст] : навчальний посібник для самостійного вивчення дисципліни / [уклад.: С. М. Братушка, С. М. Новак, С. О. Хайлук] ; Державний вищий навчальний заклад "Українська академія банківської справи Національного банку України". – Суми : ДВНЗ "УАБС НБУ", 2010. – 265 с.

3. Виробництво електроенергії в Україні за І півріччя 2021 року [Електронний ресурс] - https://vse.energy/news/pek-news/electro/1716-power-generation-202106

4. Структура електрогенерації в Україні [Електронний ресурс] - https://tek.energy/news/struktura-elektrogeneratsii-v-ukraini-ta-ii-zvyazok-iz-tarifami-na-elektroenergiyu

5. Як влаштована енергосистема України [Електронний ресурс] - https://projects.censor.net/energy/

6. Дефіцит потужності електроенергії [Електронний ресурс] – https://suspilne.media/327230-deficit-potuznosti-elektroenergii-dovedenij-dla-zakarpatta-perevisue-50-u-pikovi-navantazenna-ponad-70-bilak/

7. Хай буде світло: де і скільки електроенергії споживають та виробляють в Україні [Електронний ресурс] - https://businessviews.com.ua/ru/studies/id/da-budet-svet-gde-i-skolko-elektroenergii-potrebljajut-v-ukraine-1166/

8. Інструменти онлайн-планувальника [Електронний ресурс] - https://uk.myservername.com/top-8-best-free-online-schedule-maker-tools

9. ОСНОВНІ ПОЛОЖЕННЯ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ [Електронний ресурс] - https://lib.chmnu.edu.ua/pdf/posibnuku/313/3.pdf

10. Огляд і основи мови програмування C++ [Електронний ресурс] - http://www.znannya.org/?view=Cpp_basics

11. Desktop Guide (Windows Forms .NET) [Електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0

12. Прогнозований баланс електроенергії на 2022 рік [Електронний ресурс] - https://vse.energy/news/hotnews/1892-balance-ee-2022

13. College Schedule Maker [Електронний ресурс] - https://gizmoa.com/college-schedule-maker/

14. Adobe Spark [Електронний ресурс] - https://www.adobe.com/express/learn/blog/welcome-to-adobe-spark

Додатки

Повний код програмної реалізації додатку

```
#include "Support_and_decision_making_Systems.h"
#include <msclr\marshal_cppstd.h>
using namespace System;
using namespace System::Windows::Forms;
using namespace msclr::interop;
class RegionsOfUkraine {
public:
    std::string Name;
    float NeedPower;
    float ThisPower;
    float WorkingH;
    float WorkingM;
    RegionsOfUkraine(std::string name, float needPower, float thisPower) {
        Name = name;
        NeedPower = needPower;
        ThisPower = thisPower;
        SetTime();
    }
    std::string Get_Name() {
        return Name;
    }
    float Get_NeedPower() {
        return NeedPower;
    }
    float Get_ThisPower() {
        return ThisPower;
    }
    void Set_NeedPower(float power) {
        NeedPower = power;
        SetTime();
    }
    void Set_ThisPower(float power) {
        ThisPower = power;
        SetTime();
    }
private: void SetTime() {
    float REALTime = round((ThisPower / NeedPower) * 24 * 60 / 4);
```

```
   WorkingH = int(REALTime / 60);
   WorkingM = int(REALTime) % 60;
}
};



[STAThreadAttribute]
void Main(array<String^>^ args) {
        Application::EnableVisualStyles();
        Application::SetCompatibleTextRenderingDefault(false);


        SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems form;
        Application::Run(% form);
}
std::vector<RegionsOfUkraine> Ukraine = { {"Вінницька", 377, 0}, {"Волинська", 208, 69.33},{"Дніпропетровська", 3617, 1809},{"Донецька", 2355, 2355},{"Житомирська", 338, 150},{"Закарпатська", 247, 120},
                        {"Запорізька", 1158, 550}, {"Івано-франківська", 286, 140},{"Київська", 1913, 600},{"Кіровоградська", 416, 200},{"Луганська", 1015, 500},{"Львівська", 572, 280},
                        {"Миколаївська", 390, 200}, {"Одеська", 806, 400},{"Полтавська", 690, 350},{"Рівненська", 312, 150},{"Сумська", 286, 140},{"Тернопільська", 170, 85},
                        {"Харківська", 910, 450}, {"Херсонська", 312, 150},{"Хмельницька", 299, 150},{"Черкаська", 416, 200},{"Чернівецька", 169, 85},{"Чернігівська", 247, 120} };



System::Void SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::dataGridView1_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e)
{
        return System::Void();
}

System::Void SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
        dataGridView1->RowCount = Ukraine.size();
```

```
            dataGridView1->ColumnCount = 2;

            Show();
            dataGridView1-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
            dataGridView1->AutoResizeColumns();

            return System::Void();
}

void SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::Show()
{
   dataGridView1->TopLeftHeaderCell->Value = "Області";
   dataGridView1->Columns[0]->HeaderCell->Value = "Потребує енергії";
   dataGridView1->Columns[1]->HeaderCell->Value = "Отримує енергії";
            for (int i = 0; i < Ukraine.size(); i++) {
                        for (int j = 0; j < 2; j++) {
           dataGridView1->Rows[i]->HeaderCell->Value = marshal_as<String^>(Ukraine[i].Name);
           if (j == 0) { dataGridView1->Rows[i]->Cells[j]->Value = Convert::ToString(Ukraine[i].NeedPower); }
           else { dataGridView1->Rows[i]->Cells[j]->Value = Convert::ToString(Ukraine[i].ThisPower); }
                        }
            }
}

void SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::PointToKoma()
{
   std::string textBoxOne = msclr::interop::marshal_as<std::string>(textBox2->Text);
   std::replace(textBoxOne.begin(), textBoxOne.end(), '.', ',');
   std::string textBoxOne1 = msclr::interop::marshal_as<std::string>(textBox3->Text);
   std::replace(textBoxOne1.begin(), textBoxOne1.end(), '.', ',');
   textBox2->Text = marshal_as<String^>(textBoxOne);
   textBox3->Text = marshal_as<String^>(textBoxOne1);
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::exitToolStripMenuIte
m_Click(System::Object^ sender, System::EventArgs^ e)
{
   Application::Exit();
}
```

```
System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::textBox1_TextChange
d(System::Object^ sender, System::EventArgs^ e)
{
   return System::Void();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::textBox2_TextChange
d(System::Object^ sender, System::EventArgs^ e)
{
   return System::Void();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::textBox3_TextChange
d(System::Object^ sender, System::EventArgs^ e)
{
   return System::Void();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button2_Click(System
::Object^ sender, System::EventArgs^ e)
{
   if ((this->textBox1->Text == String::Empty)|| (this->textBox2->Text == String::Empty)|| (this->textBox3->Text
== String::Empty)) {
      MessageBox::Show("При оновленні всі поля повинні бути заповнені. Спробуйте натиснути на клавішу
First, щоб запустити автозаповнення", "Помилка");
   }
   else {
      std::string textBoxOne = msclr::interop::marshal_as<std::string>(textBox1->Text);
      float textBoxTwo = Convert::ToDouble(textBox2->Text);
      float textBoxThree = Convert::ToDouble(textBox3->Text);
      for (int i = 0; i < Ukraine.size(); i++) {
         if (Ukraine[i].Name == textBoxOne) {
            if (textBoxTwo) {
               Ukraine[i].Set_NeedPower(textBoxTwo);
            }
            Ukraine[i].Set_ThisPower(textBoxThree);
         }
```

```
        }
        Show();
    }
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button3_Click(System
::Object^ sender, System::EventArgs^ e)
{
    textBox1->Text = marshal_as<String^>(Ukraine[0].Name);
    textBox2->Text = marshal_as<String^>(std::to_string(Ukraine[0].NeedPower));
    textBox3->Text = marshal_as<String^>(std::to_string(Ukraine[0].ThisPower));
    PointToKoma();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button4_Click(System
::Object^ sender, System::EventArgs^ e)
{
    std::string textBoxOne = msclr::interop::marshal_as<std::string>(textBox1->Text);
    for (int i = 0; i < Ukraine.size(); i++) {
        if (Ukraine[i].Name == textBoxOne) {
            if (i - 1 > 0) {
                textBox1->Text = marshal_as<String^>(Ukraine[i-1].Name);
                textBox2->Text = marshal_as<String^>(std::to_string(Ukraine[i-1].NeedPower));
                textBox3->Text = marshal_as<String^>(std::to_string(Ukraine[i-1].ThisPower));
            }
            else {
                textBox1->Text = marshal_as<String^>(Ukraine[Ukraine.size()-1].Name);
                textBox2->Text = marshal_as<String^>(std::to_string(Ukraine[Ukraine.size() - 1].NeedPower));
                textBox3->Text = marshal_as<String^>(std::to_string(Ukraine[Ukraine.size() - 1].ThisPower));
            }
        }
    }
    PointToKoma();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button5_Click(System
::Object^ sender, System::EventArgs^ e)
{
```

```
    std::string textBoxOne = msclr::interop::marshal_as<std::string>(textBox1->Text);
    for (int i = 0; i < Ukraine.size(); i++) {
        if (Ukraine[i].Name == textBoxOne) {
            if (i + 1 > 23) {
                textBox1->Text = marshal_as<String^>(Ukraine[0].Name);
                textBox2->Text = marshal_as<String^>(std::to_string(Ukraine[0].NeedPower));
                textBox3->Text = marshal_as<String^>(std::to_string(Ukraine[0].ThisPower));
            }
            else {
                textBox1->Text = marshal_as<String^>(Ukraine[i+1].Name);
                textBox2->Text = marshal_as<String^>(std::to_string(Ukraine[i+1].NeedPower));
                textBox3->Text = marshal_as<String^>(std::to_string(Ukraine[i+1].ThisPower));
            }
        }
    }
    PointToKoma();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::dataGridView2_CellC
ontentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e)
{
    return System::Void();
}

System::Void
SupportanddecisionmakingSystemsWindowsForm::Support_and_decision_making_Systems::button6_Click(System
::Object^ sender, System::EventArgs^ e)
{
    dataGridView2->RowCount = Ukraine.size()*3;
    dataGridView2->ColumnCount = 5;

    dataGridView2->TopLeftHeaderCell->Value = "Області";
    dataGridView2->Columns[0]->HeaderCell->Value = "Черга";
    dataGridView2->Columns[1]->HeaderCell->Value = "00:00-06:00";
    dataGridView2->Columns[2]->HeaderCell->Value = "06:00-12:00";
    dataGridView2->Columns[3]->HeaderCell->Value = "12:00-18:00";
    dataGridView2->Columns[4]->HeaderCell->Value = "18:00-00:00";
    for (int i = 0; i < Ukraine.size()*3; i++) {
            if (i%3 == 0) {
                dataGridView2->Rows[i]->HeaderCell->Value = marshal_as<String^>(Ukraine[i / 3].Name);
```

```
            dataGridView2->Rows[i]->Cells[0]->Value = "1";
            for (int j = 1; j < 5; j++) {
               std::string time;
               if (j == 1) {
                  if (Ukraine[i / 3].WorkingM) {
                     if (int(Ukraine[i / 3].WorkingM) > 9) {
                        time = "00:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH)) + ":" + std::to_string(int(Ukraine[i / 3].WorkingM));
                     }
                     else {
                        time = "00:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH)) + ":0" + std::to_string(int(Ukraine[i / 3].WorkingM));
                     }
                  }
                  else {
                     time = "00:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH)) + ":00";
                  }
               }
               else if (j == 2) {
                  if (Ukraine[i / 3].WorkingM) {
                     if (int(Ukraine[i / 3].WorkingH + 6)>9) {
                        if (int(Ukraine[i / 3].WorkingM) > 9) {
                           time = "06:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 6)) + ":" + std::to_string(int(Ukraine[i / 3].WorkingM));
                        }
                        else {
                           time = "06:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 6)) + ":0" + std::to_string(int(Ukraine[i / 3].WorkingM));
                        }
                     }
                     else {
                        if (int(Ukraine[i / 3].WorkingM) > 9) {
                           time = "06:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH + 6)) + ":" + std::to_string(int(Ukraine[i / 3].WorkingM));
                        }
                        else {
                           time = "06:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH + 6)) + ":0" + std::to_string(int(Ukraine[i / 3].WorkingM));
                        }
                     }
                  }
```

```
                else {
                    if (int(Ukraine[i / 3].WorkingH+ 6) > 9) {
                        time = "06:00-" + std::to_string(int(Ukraine[i / 3].WorkingH)+6) + ":00";
                    }
                    else {
                        time = "06:00-0" + std::to_string(int(Ukraine[i / 3].WorkingH)+6) + ":00";
                    }
                }
            }
            else if (j == 3) {
                if (Ukraine[i / 3].WorkingM) {
                    if (int(Ukraine[i / 3].WorkingM) > 9) {
                        time = "12:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 12)) + ":" +
std::to_string(int(Ukraine[i / 3].WorkingM));
                    }
                    else {
                        time = "12:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 12)) + ":0" +
std::to_string(int(Ukraine[i / 3].WorkingM));
                    }
                }
                else {
                    time = "12:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 12)) + ":00";
                }
            }
            else {
                if (Ukraine[i / 3].WorkingM) {
                    if (int(Ukraine[i / 3].WorkingM) > 9) {
                        time = "18:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 18)) + ":" +
std::to_string(int(Ukraine[i / 3].WorkingM));
                    }
                    else {
                        time = "18:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 18)) + ":0" +
std::to_string(int(Ukraine[i / 3].WorkingM));
                    }
                }
                else {
                    time = "18:00-" + std::to_string(int(Ukraine[i / 3].WorkingH + 18)) + ":00";
                }
            }
            dataGridView2->Rows[i]->Cells[j]->Value = marshal_as<String^>(time);
        }
```

```
        }
        else if (i%3 == 1) {
            dataGridView2->Rows[i]->Cells[0]->Value = "2";
            for (int j = 1; j < 5; j++) {
                std::string time;
                float Time = round((Ukraine[i / 3].WorkingH * 60 + Ukraine[i / 3].WorkingM) / 2);
                int TimeH = int(Time / 60);
                int TimeM = int(Time) % 60;
                if (j == 1) {
                    if (TimeM>50) {
                        time = "0" + std::to_string(int(3-TimeH-1))+":0" + std::to_string(int(60-TimeM)) + "-0" +
std::to_string(int(3 + TimeH)) + ":" + std::to_string(int(TimeM));
                    }
                    else if(TimeM==0){
                        time = "0" + std::to_string(int(3 - TimeH)) + ":00-0" + std::to_string(int(3 + TimeH)) + ":00";
                    }
                    else if (10 > TimeM > 0) {
                        time = "0" + std::to_string(int(3 - TimeH - 1)) + ":" + std::to_string(int(60 - TimeM)) + "-0" +
std::to_string(int(3 + TimeH)) + ":0" + std::to_string(int(TimeM));
                    }
                    else {
                        time = "0" + std::to_string(int(3 - TimeH-1)) + ":" + std::to_string(int(60 - TimeM)) + "-0" +
std::to_string(int(3 + TimeH)) + ":" + std::to_string(int(TimeM));
                    }
                }
                else if (j == 2) {
                    if (TimeH) {
                        if (TimeM > 50) {
                            time = "0" + std::to_string(int(9 - TimeH-1)) + ":0" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(9 + TimeH)) + ":0" + std::to_string(int(TimeM));
                        }
                        else if (TimeM == 0) {
                            time = "0" + std::to_string(int(9 - TimeH)) + ":00-" + std::to_string(int(9 + TimeH)) + ":00";
                        }
                        else if (10 > TimeM > 0) {
                            time = "0" + std::to_string(int(9 - TimeH - 1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(9 + TimeH)) + ":0" + std::to_string(int(TimeM));
                        }
                        else {
                            time = "0" + std::to_string(int(9 - TimeH-1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(9 + TimeH)) + ":" + std::to_string(int(TimeM));
```

```
                }
            }
            else {
                if (TimeM > 50) {
                    time = "0" + std::to_string(int(9 - TimeH-1)) + ":0" + std::to_string(int(60 - TimeM)) + "-0" +
std::to_string(int(9 + TimeH)) + ":" + std::to_string(int(TimeM));
                }
                else if (TimeM == 0) {
                    time = "0" + std::to_string(int(9 - TimeH)) + ":00-0" + std::to_string(int(9 + TimeH)) + ":00";
                }
                else if (10 > TimeM > 0) {
                    time = "0" + std::to_string(int(9 - TimeH - 1)) + ":" + std::to_string(int(60 - TimeM)) + "-0" +
std::to_string(int(9 + TimeH)) + ":0" + std::to_string(int(TimeM));
                }
                else {
                    time = "0" + std::to_string(int(9 - TimeH-1)) + ":" + std::to_string(int(60 - TimeM)) + "-0" +
std::to_string(int(9 + TimeH)) + ":" + std::to_string(int(TimeM));
                }
            }
        }
        else if (j == 3) {
            if (TimeM > 50) {
                time = "" + std::to_string(int(15 - TimeH-1)) + ":0" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(15 + TimeH)) + ":" + std::to_string(int(TimeM));
            }
            else if (TimeM == 0) {
                time = "" + std::to_string(int(15 - TimeH)) + ":00-" + std::to_string(int(15 + TimeH)) + ":00";
            }
            else if (10 > TimeM > 0) {
                time = "" + std::to_string(int(15 - TimeH - 1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(15 + TimeH)) + ":0" + std::to_string(int(TimeM));
            }
            else {
                time = "" + std::to_string(int(15 - TimeH-1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(15 + TimeH)) + ":" + std::to_string(int(TimeM));
            }
        }
        else {
            if (TimeM > 50) {
                time = "" + std::to_string(int(21 - TimeH-1)) + ":0" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(21 + TimeH)) + ":" + std::to_string(int(TimeM));
```

```
                }
                else if (TimeM == 0) {
                    time = "" + std::to_string(int(21 - TimeH)) + ":00-" + std::to_string(int(21 + TimeH)) + ":00";
                }
                else if (10 > TimeM > 0) {
                    time = "" + std::to_string(int(21 - TimeH - 1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(21 + TimeH)) + ":0" + std::to_string(int(TimeM));
                }
                else {
                    time = "" + std::to_string(int(21 - TimeH-1)) + ":" + std::to_string(int(60 - TimeM)) + "-" +
std::to_string(int(21 + TimeH)) + ":" + std::to_string(int(TimeM));
                }
            }
            dataGridView2->Rows[i]->Cells[j]->Value = marshal_as<String^>(time);
        }
    }
    else {
        dataGridView2->Rows[i]->Cells[0]->Value = "3";
        for (int j = 1; j < 5; j++) {
            std::string time;
            if (j == 1) {
                if (Ukraine[i / 3].WorkingM) {
                    if (int(Ukraine[i / 3].WorkingM) > 50) {
                        time = "0" + std::to_string(int(6-(Ukraine[i / 3].WorkingH)-1)) + ":0" + std::to_string(int(60-
Ukraine[i / 3].WorkingM))+"-06:00";
                    }
                    else {
                        time = "0" + std::to_string(int(6-Ukraine[i / 3].WorkingH-1)) + ":" + std::to_string(int(60-
Ukraine[i / 3].WorkingM)) + "-06:00";
                    }
                }
                else {
                    time = "0" + std::to_string(int(6-Ukraine[i / 3].WorkingH)) + ":00-06:00";
                }
            }
            else if (j == 2) {
                if (Ukraine[i / 3].WorkingH >=2) {
                    if (Ukraine[i / 3].WorkingM) {
                        if (int(Ukraine[i / 3].WorkingM) > 50) {
                            time = "0" + std::to_string(int(12 - (Ukraine[i / 3].WorkingH) - 1)) + ":0" + std::to_string(int(60
- Ukraine[i / 3].WorkingM)) + "-12:00";
```

```
                    }
                    else {
                        time = "0" + std::to_string(int(12 - Ukraine[i / 3].WorkingH - 1)) + ":" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-12:00";
                    }
                }
                else {
                    if (Ukraine[i / 3].WorkingH == 2) {
                        time = "" + std::to_string(int(12 - Ukraine[i / 3].WorkingH)) + ":00-12:00";
                    }
                    else {
                        time = "0" + std::to_string(int(12 - Ukraine[i / 3].WorkingH)) + ":00-12:00";
                    }
                }
            }
            else {
                if (Ukraine[i / 3].WorkingM) {
                    if (int(Ukraine[i / 3].WorkingM) > 50) {
                        time = "" + std::to_string(int(12 - (Ukraine[i / 3].WorkingH) - 1)) + ":0" + std::to_string(int(60
- Ukraine[i / 3].WorkingM)) + "-12:00";
                    }
                    else {
                        time = "" + std::to_string(int(12 - Ukraine[i / 3].WorkingH - 1)) + ":" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-12:00";
                    }
                }
                else {
                    time = "" + std::to_string(int(12 - Ukraine[i / 3].WorkingH)) + ":00-12:00";
                }
            }
        }
        else if (j == 3) {
            if (Ukraine[i / 3].WorkingM) {
                if (int(Ukraine[i / 3].WorkingM) > 50) {
                    time = "" + std::to_string(int(18 - (Ukraine[i / 3].WorkingH) - 1)) + ":0" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-18:00";
                }
                else {
                    time = "" + std::to_string(int(18 - Ukraine[i / 3].WorkingH - 1)) + ":" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-18:00";
                }
```

```
                }
                else {
                    time = "" + std::to_string(int(18 - Ukraine[i / 3].WorkingH)) + ":00-18:00";
                }
            }
            else {
                if (Ukraine[i / 3].WorkingM) {
                    if (int(Ukraine[i / 3].WorkingM) > 50) {
                        time = "" + std::to_string(int(24 - (Ukraine[i / 3].WorkingH) - 1)) + ":0" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-00:00";
                    }
                    else {
                        time = "" + std::to_string(int(24 - Ukraine[i / 3].WorkingH - 1)) + ":" + std::to_string(int(60 -
Ukraine[i / 3].WorkingM)) + "-00:00";
                    }
                }
                else {
                    time = "" + std::to_string(int(24 - Ukraine[i / 3].WorkingH)) + ":00-00:00";
                }
            }
            dataGridView2->Rows[i]->Cells[j]->Value = marshal_as<String^>(time);
        }
    }
}
dataGridView2-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
dataGridView2->AutoResizeColumns();

return System::Void();
}
```