Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка програмного продукту оцінки та прогнозування проєктів»

Виконав: студент групи К19-1
Спеціальність 122 «Комп'ютерні науки»
Рухлов А. В.
(прізвище та ініціали)
Керівник:
к. ф.-м. н., доц. Лебідь О. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент: Спеціалізоване управління розробки та супроводження програмного забезпечення Департаменту з питань цифрового розвитку, цифрових трансформацій та цифровізації ДМСУ
(місце роботи)
Головний державний інспектор відділу розробки програмного забезпечення
(посада)
Бахтін О. В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Рухлов А. В. Розробка програмного продукту оцінки та прогнозування проєктів.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2023.

Метою кваліфікаційної роботи є створення програми для отримання інформації щодо кінцевої дати закінчення проєкту та наочний показ даних для розуміння етапів проєкту та ступеня виконання.

У роботі проведено аналіз можливих методів прогнозування завершення робіт по проєкту, причини щодо запізнення завершення, варіанти покращення оцінки та прогнозу. Приділено увагу психологічному аспекту планування та помилок, які найчастіше допускаються. Для розроблення програми використані сучасні середовища розробки. Архітектура програми побудована за сучасними принципами та має технологічну структуру. Інтерфейс зроблений зрозумілим користувачеві і не має візуальної навантаженості. Розроблені функції дають змогу користувачу скористатися інструментами для аналізу плану робіт. Були показані методи для експорту та імпорту даних.

У кваліфікаційному проєкті можна побачити програму для оцінки стану роботи проєкта та приблизних строків його завершення. Користувач має можливість розробити план свого проєкту, строки етапів, подивитись на успішність втілювання проєкту. Це допоможе для планування різноманітних завдань у будь-якій сфері діяльності та убереже від помилок у проєктуванні виконання плану. Кінцевий продукт показує просту, надійну та функціональну програму, яка підійде широкому колу користувачів.

Ключові слова: ПРОГНОЗ, АНАЛІЗ, ГРАФІК, ТРЕНД, ОЦІНКА, ПЛАН, СПРИНТ.

ANNOTATION

Rukhlov A. V. Development of a software product for project evaluation and forecasting.

Thesis for obtaining a bachelor’s degree in the specialty 122 "Computer science". – University of Customs and Finance, Dnipro, 2023

The purpose of the qualification work is to create a program for obtaining information about the final date of the end of the project and visual display of data to understand the stages of the project and the degree of implementation.

An analysis of possible methods of forecasting the completion of work on the project, reasons for late completion, options for improving the estimate and forecast was carried out. Attention was paid to the psychological aspect of planning and the most common mistakes. Modern development environments were used to develop the program. The architecture of the program is built according to modern principles and has a technological structure. The interface is made clear to the user and has no visual load. The developed functions allow the user to use tools for analyzing the work plan. Methods for exporting and importing data were shown.

In the qualification project, you can see the program for evaluating the project’s work status and the approximate terms of its completion. The user has the opportunity to develop a plan for his project, the terms of the stages, to see the success of the implementation of the project. This will help to plan a variety of tasks in any field of activity and will protect against mistakes in planning the implementation of the plan. The final product shows a simple, reliable and functional application that will suit a wide range of users.

Keywords: FORECAST, ANALYSIS, SCHEDULE, TREND, ESTIMATE, PLAN, SPRINT.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI – User interface (інтерфейс користувача)

API – application programming interface (інтерфейс програмування додатків)

ПЗ – програмне забезпечення

IDE – Integrated development environment (інтегроване середовище розробки)

OS – Operating system (операційна система)

PDF – Portable document format (універсальний формат файлів)

GUI – Graphical user interface (графічний інтерфейс користувача

ЗМІСТ

ВСТУП

З розвитком галузі менеджменту у бізнесі та побуті, з’являються рішення, що дозволяють користувачам ефективно аналізувати виконання завдань, передбачати завершення проєктів та оцінювати їх результативність. Ці програми набувають все більшої популярності й корисності для широкого кола користувачів, починаючи від окремих професіоналів до великих підприємств

Актуальність кваліфікаційної роботи полягає у тому, що вона спрямована на розробку додатку, який забезпечуватиме аналіз даних про проєкти і надаватиме інформацію про їх етапи та строк закінчення. З урахуванням зростаючої популярності та значущості програм, які допомагають управляти проєктами, ця робота стає актуальною для широкого спектру користувачів. Незалежно від того, чи це індивідуальний професіонал або велике підприємство, наявність інструменту, що забезпечує детальний аналіз стану проєктів та запланованих завдань, стає необхідністю.

Важливість програми виявляється в тому, що додаток, який розробляється, буде забезпечувати користувачам ретельний огляд стану їх проєктів та задач. Він буде надавати інформацію про тривалість завдань, прогрес їх виконання, оцінку їх складності. Це дозволить користувачам оперативно оцінювати поточний стан проєкту й приймати належні рішення для покращення його ефективності. Завдяки додатку користувачі зможуть вести ефективну проєктну діяльність, уникати затримок та збитків і досягати поставлених цілей вчасно.

Таким чином, розробка додатку, який аналізуватиме дані про проєкт і надаватиме інформацію про його етапи та строк закінчення, має велику актуальність у сучасному світі бізнесу та менеджменту. Цей додаток стане незамінним інструментом для ефективного управління проєктами, що дозволить користувачам бути більш продуктивними, ефективними та успішними у своїй діяльності. Тому, дана тема є актуальною.

Об’єктом дослідження є технології розробка десктопного застосунку.

Предмет дослідження: розробка програми з аналізу та прогнозування проєктів.

Метою є створення та побудова простого для використання додатка, який має зрозумілий інтерфейс та дозволяє припускати строки закінчення справи, імпортувати та експортувати дані, мати можливість показати наочно аналізовану інформацію.

Завданням кваліфікаційної роботи є:

– проаналізувати проблемне питання розробки програмного забезпечення для оцінки та прогнозування проєктів;

– сформулювати постановку завдання;

– описати вимоги до програмного засобу;

– провести аналіз оцінювання багатоетапної роботи;

– розглянути математичні засоби для оцінки прогнозування;

– провести аналіз сучасних технологій створення десктопних програм;

– розробити архітектуру програмної системи;

– описати та реалізувати інтерфейс користувача для програмного продукту оцінки та прогнозування проєктів;

– провести тестування та описати можливі сценарії роботи користувача.

Розроблене програмне забезпечення повинно обробляти початкові та поточні дані про стан роботи у проєкті та мати можливість прогнозувати якість і строки його завершення.

Важливою задачею є розробити таку структуру та спроєктувати такий інтерфейс користувача, щоб застосунок був легкий та ненавантажений функціями і елементами взаємодії.

Потрібно зробити наголос на візуальному відображенні даних, на простоті та швидкості аналізу та на гнучкості вхідних даних. Користувач може сам вибрати усі початкові параметри. Кінцеві дані забезпечують розгорнуті звіти, які дозволяють оцінювати продуктивність, ефективність та відповідність метам проєктів. Крім того, ці рішення допомагають виявляти

можливі ризики, затримки та проблеми, що можуть виникнути під час виконання завдань.

Необхідно проаналізувати: потреби користувача, щоб зрозуміти вимоги та очікування, що дасть більш точне та ефективне задоволення потреб і покращення користування продуктом. Треба зробити аналіз концепції програми, яка полягає у створенні додатку, тому що створення плану дій, оцінка задач та орієнтація у часі проєкта є важливими етапами управління проєктом. Вони допомагають зорієнтуватися у меті та ресурсах, встановити пріоритети та дедлайни, забезпечують керівництво командою та спрощують моніторинг прогресу. Це забезпечує ефективне виконання завдань, уникнення затримок та досягнення успіху проєкту.

С технічного зору, потрібно розробити програму, яка зможе:

– приймати вхідні дані через ручне введення;
– приймати вхідні дані через автоматичний імпорт даних;
– видавати дані через експорт у декількох варіантах;
– будувати таблиці;
– змінювати таблиці;
– малювати графіки;
– мати захист від неочікуваних дій користувача;
– спроєктована у простому та приємному інтерфейсі;

Методи дослідження: спостереження, порівняння, рахунок, вимірювання, експеримент, узагальнення, абстрагування, формалізація, аналіз і синтез, індукція і дедукція, аналогія, моделювання, ідеалізація, ранжирування, а також аксіоматичний, гіпотетичний, історичний і системні методи.

Робота підтверджує наступні програмні результати навчання, що відповідають освітній програмі 122 «Комп’ютерні науки»:

– ПР1. Застосовувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів

вилучення, аналізу, обробки та синтезу інформації в предметній області комп’ютерних наук.

– ПР2. Використовувати сучасний математичний апарат неперервного та дискретного аналізу, лінійної алгебри, аналітичної геометрії, в професійній діяльності для розв’язання задач теоретичного та прикладного характеру в процесі проєктування та реалізації об’єктів інформатизації.

– ПР5. Проєктувати, розробляти та аналізувати алгоритми розв’язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій.

– ПР8. Використовувати методологію системного аналізу об’єктів, процесів і систем для задач аналізу, прогнозування, управління та проєктування динамічних процесів в макроекономічних, технічних, технологічних і фінансових об’єктах.

– ПР9. Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв’язання задач в галузі комп’ютерних наук.

Структура кваліфікаційної роботи бакалавра: вступ, три розділи, висновки, список використаних джерел, що містить 45 посилань та додатку. Робота містить 28 рисунків, обсяг – 54 сторінки.

РОЗДІЛ 1.
ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ОЦІНКИ ТА ПРОГНОЗУВАННЯ ПРОЄКТІВ

1.1 Постановка завдання

Завдання програми обумовлені сучасними технологіями та потребами користувачів.

По-перше, людині потрібно мати простір для запису своїх завдань. У голові неможливо тримати всі пункти, і для цього зазвичай використовуються блокноти, записані книги або щоденники.

По-друге, є необхідність оцінки складності завдань, так як всі завдання індивідуальні, для них знадобиться різна кількість часу і зусиль до виконання. Також корисними будуть нотатки про дати створення завдання та планова дата завершення. Це допоможе в орієнтації в плані виконання, дасть можливість зрозуміти, на якому етапі знаходиться проєкт, скільки часу ще потрібно для завершення та загальний період виконання кожного завдання та всіх разом.

По-третє, має бути можливість перенесення даних. Для цього обрано Ексель та PDF. Експорт даних в Ексель дозволить взаємодіяти з ними надалі, а також збережена інформація дає варіант для імпорту в інші сучасні додатки. Експортовані дані до PDF дає можливість для звіту та компактного огляду даних у текстовому вигляді. Імпорт даних з Excel прискорить процес заповнення початкової таблиці та скоротить час для подальших розрахунків.

По-четверте, потрібна візуалізація даних розрахунків, для цього ідеально підходять графіки. Діаграми мають бути зрозумілі, прості та наочні, щоб швидше доносити інформацію.

Виходячи з усіх перелічених вище вимог, список завдань, які повинна виконувати програма, наступний:

1) Запис усіх задач для проєкту.

2) Оцінка складності задач.

3) Зазначення дат початку та кінця задач.

4) Підрахунок довжини етапів

5) Підрахунок кількості етапів

6) Позначення дат початку та кінця етапу

7) Підрахунок усієї кількості роботи

8) Підрахунок кількості зробленої роботи за всі етапи

9) Підрахунок кількості зробленої роботи за конкретний етап

10) Наочне зображення інформації на графіках.

11) Зображення на графіках лінії тренду що дозволяє спрогнозувати дату закінчення усіх етапів.

12) Експорт та імпорт даних з Excel.

13) Експорт даних до PDF файлу.

1.2 Вимоги до програмного забезпечення

Програма повинна бути потужним інструментом для планування та управління проєктами. Вона має надавати широкий спектр функціональності, яка допомагає організувати та ефективно виконувати завдання.

Продукту потрібно зберігати всі задачі, пов’язані з проєктом, в одному місці, що буде спрощувати їх організацію та контроль. Треба розробити функції щоб легко створювати, відстежувати та оновлювати список задач, а також надавати їм оцінку складності і позначати статус виконання. Це дозволить бачити загальну картину і мати доступ до актуальної інформації.

Оцінка складності задач також повинна стати ще однією корисною функцією цієї програми. Необхідно дати користувачу можливість призначити складність кожній задачі на основі її обсягу, труднощів та ресурсів, необхідних для її виконання. Це дозволить краще розподілити свій час та реалістично оцінити тривалість проєкту.

Додатковою функцією цієї програми буде можливість встановлювати дати початку та кінця задач. Буде змога надавати конкретні терміни виконання

і керувати графіком проєкту. Це спростить планування робіт, допоможе визначити терміни завершення проєкту та вчасно виявити можливі затримки.

Програма також буде надавати можливість підрахунку довжини етапів та кількості етапів. Наприклад, розділити проєкт на логічні частини і зорганізувати їх послідовно. Це дозволить вам контролювати прогрес робіт, оцінювати тривалість кожного етапу та визначити, які ресурси необхідно виділити на кожний з них.

Однією з важливих можливостей цієї програми стане підрахунок усієї кількості роботи, це дозволить користувачу оцінити загальний обсяг завдань, необхідних для завершення проєкту. Це стане важливим фактором при плануванні ресурсів.

Крім того, програма повинна надавати можливість підрахунку кількості зробленої роботи за всі етапи, а також за кожен окремий етап. Це дозволить користувачу оцінити прогрес проєкту, визначити, наскільки етапи виконуються відповідно до плану та виявити можливі відхилення.

Остання функція, але не менш важлива, це наочне зображення інформації на графіках. Користувач зможе візуалізувати графіки, що демонструють прогрес проєкту, терміни виконання завдань, розподіл роботи між етапами та багато іншого. Графічна репрезентація допоможе вам краще розуміти дані та зробити обґрунтовані рішення.

Отже, ця програма буде корисною для будь-якого, хто займається управлінням проєктами – менеджерів проєктів, командних лідерів, планувальників, аналітиків та учасників проєкту, або звичайних людей. Вона надасть надійні інструменти для планування, контролю та оцінки роботи, що допомагає ефективно керувати проєктами та досягати поставлених цілей.

Системні вимоги додатку. Програма повинна бути не вимогливою до комп’ютера, тому системні вимоги прості:

– Операційна система Windows 7 або більш пізня версія.
– Процесор Intel Core 2 Duo кращий.
– Оперативна пам’ять мінімум 512 МБ RAM.

– Вільне місце на жорсткому диску: мінімум 100 МБ.

– Відеокарта сумісна з DirectX 9 або вищою версією.

– Звуковий пристрій сумісний з DirectX 9 або вищою версією.

– Монітор з роздільною здатністю 1024x768 пікселів або вище.

– Клавіатура та миша сумісні з Windows.

– Програмне забезпечення: Microsoft .NET Framework версії 4.0 або вище.

1.3 Опис продукту

Мовою програмування буде C#, це одна з найпопулярніших мов програмування, яка є незалежною від платформи та має широку екосистему. У виборі технології обрана WinForms, тому що вона сумісна з багатьма версіями Windows а також має зручний інтерфейс для звичайного користувача. Формою програми між хмарним та локальним був обраний десктопний варіант.

У десктопних додатках є свої переваги, а саме:

– незалежність від Інтернет-з’єднання: десктопні додатки працюють безпосередньо на комп’ютері користувача, тому вони не вимагають постійного Інтернет-з’єднання для своєї роботи. Це особливо корисно в умовах обмеженого або непостійного доступу до Інтернету;

– більш висока швидкодія: десктопні додатки мають тенденцію працювати швидше, оскільки вони виконуються на локальному комп’ютері, а не в хмарному середовищі. Вони можуть миттєво реагувати на дії користувача і швидко обробляти великі обсяги даних;

– більша контрольованість та безпека даних: оскільки десктопні додатки працюють локально, користувач має більший контроль над своїми даними. Вони можуть зберігати дані безпосередньо на локальному пристрої, що дозволяє більш точно контролювати доступ до них та забезпечувати вищий рівень безпеки.

Програма розраховується для використання звичайними користувачами, тому повинна мати простий та чіткий інтерфейс. Це допоможе новим користувачам швидше опанувати програму, їм не потрібно буде бентежитися про велику кількість інтерфейсів.

Також додаток повинен мати захист від помилок користувача, тобто потрібно передбачити усі можливі сценарії роботи та варіанти введення даних. У разі незапланованих дій чи некоректної поведінки, користувач має отримувати повідомлення задля комфортного користування.

1.4 Висновки до першого розділу

У першому розділі були розглянуті сучасні технології та потреби користувачів у керуванні завданнями та проєктами. Програма, яка розробляється, має на меті забезпечити користувачам простір для запису завдань, оцінку їх складності, планування та візуалізацію даних. Крім того, вона надасть можливість експорту та імпорту даних з Excel, а також експорту до PDF для зручності звітування. Цей розділ виокремлює важливі функціональні можливості додатку, спрямовані на поліпшення організації та керування завданнями у сучасному робочому середовищі.

Також були визначені основні вимоги та функціональні можливості програми для планування та управління проєктами. Важливими функціями є підрахунок тривалості етапів та загальної кількості роботи, а також візуалізація даних на графіках.

Для розробки додатку обрано мову програмування C# та технологію WinForms. Десктопний варіант додатка був вибраний з метою незалежності від Інтернет-з’єднання, забезпечення більшої швидкодії та контролю над даними. Програма буде мати простий інтерфейс для зручного використання користувачами, а також захист від помилок та некоректної поведінки. Загальна мета – забезпечити комфортне користування програмою широким колом користувачів.

РОЗДІЛ 2.
АНАЛІЗ ПИТАНЬ ОЦІНЮВАННЯ ТА ПРОГНОЗУВАННЯ ПРОЄКТІВ

2.1 Проблеми та труднощі з оцінюванням декількаетапної роботи

Кожна людина у своєму житті будує план для якоїсь мети. Це може бути невеликий задум, як список покупок в магазин, а може бути великий проєкт, як для будівництва будинку. Тоді звичайним списком вже не обійтися, і потрібно розписати все до дрібниць, адже це дорогий процес, і його тяжкість у реалізації також відображається у довгостроковому плануванні, загальний час виконання якого може тривати кілька років.

Для наочного прикладу необхідності плану виберемо мету – підготуватися до іспиту. Це також важке завдання і воно складне психологічно, адже від результату залежитиме подальші рішення щодо життя. Не дуже важливо, який це іспит: випускний після школи, складання на автомобільні права, або іспит з мови.

Підготовка вимагає великої уважності, адже важливо на першому етапі розписати, яким чином проходитиме підготовка, як багато часу їй приділятиметься, і наскільки складною вона буде [1].

Однак, відразу не зрозуміло, як розбити велике завдання. Є спокус піти в дві крайності - зробити величезну кількість дрібних завдань, так не дійшовши до головних, і навпаки, розбити всього на пару штук, заспокоюючи себе, що завдання загалом нескладне, і зробити його легко.

Якщо ж все ж таки вдалося спроєктувати достатню кількість підзавдань, поступово розподілити їхню складність, то настає така складність, а саме планування їх виконання. Немає впевненості в тому, що будуть виконані всі завдання в строк, і що з появою нових графік робіт не зіб’ється, адже дуже рідко заздалегідь можна продумати абсолютно всі деталі та можливі аспекти, які з’являтимуться під час виконання робіт.

Для цього потрібно передбачити, з якою частотою з’являтимуться нові завдання, і наскільки посильно встигатимуть їх робити.

Повернемося до підготовки до іспитів. Допустимо, знаємо всю програму іспиту, і найзрозумілішим шляхом тренування буде вирішення тестів попередніх років. Тоді нам потрібно:

1) Скласти «пакет» тестів попередніх років.

2) Зрозуміти, скільки часу йде на рішення одного білету.

3) Прорахувати приблизну дату завершення всієї підготовки.

Однак такий план є ненадійним, незважаючи на його очевидну простоту, яка інтуїтивно приваблива.

Під час вирішення одного з квитків, можна усвідомити, що є прогалина знань у певній темі з предмета, і якщо не "залатати" цю дірку, то це може відгукнутися на іспиті. Отже, слід виділити час на додаткову підготовку, але це зрушує графік. Також є індивідуальні можливі похибки, у екзаменаційного листка це може бути складність завдань, що зростає. Якщо неправильно прорахувати час виконання одного листка виходячи тільки з перших завдань, це також позначиться на глобальній помилці розрахунку.

У сучасному світі, де час – цінний ресурс, а конкуренція неймовірно висока, розробка чіткого плану своїх дій стає надзвичайно важливою для успіху у будь-якій справі чи проєкті. Будь то особиста мета, бізнес-проєкт, академічна робота або будь-яка інша сфера діяльності, наявність плану, який містить точні та чесні задачі, допомагає уникнути розчарувань та досягнути бажаного результату [2].

Спочатку важно зрозуміти, що таке план дій. План – це структурований набір кроків та дій, які потрібно виконати для досягнення певної мети. Це своєрідна дорожня карта, яка надає орієнтири та направлення на шляху до успіху. Планування своїх дій допомагає зосередитися на головних завданнях, уникнути розкидання сил та ресурсів, а також визначити пріоритети.

Однією з переваг чіткого планування є створення конкретних та досяжних цілей. Задачі, які вказані у плані, повинні бути SMART:

специфічними (specific), вимірюваними (measurable), досяжними (achievable), релевантними (relevant) та обмеженими за часом (time-bound). Такий підхід дозволяє уникнути неясностей та розмитості в цілях, а також допомагає зрозуміти, коли конкретна мета досягнута [3].

Крім того, планування забезпечує систематичний підхід до роботи та дозволяє ефективно управляти часом. Відповідно розподілити ресурси та обов'язки між учасниками проєкту. Планування дозволяє заздалегідь передбачити можливі складнощі, ризики та перешкоди, що допомагає готуватися до них та знайти шляхи їх подолання. Він створює структуру і організовує процес роботи, що відбивається на якості та ефективності виконання завдань.

Однак, навіть найкраще спланований проєкт може зіткнутися з непередбаченими обставинами та невдачами. Люди мають тенденцію розчаровуватися, коли щось не йде за планом. Важливо розуміти, що план – це не незмінний документ, а живий інструмент, який може адаптуватися до нових обставин та змін у ході виконання проєкту. Гнучкість та готовність до змін – це ключові якості, які дозволяють подолати труднощі та шукати альтернативні шляхи досягнення мети.

Найважливіше – не занепадати духом та використовувати невдачі як можливості для вдосконалення. Вони можуть стати цінним досвідом та допомогти зрозуміти помилки або недоліки в плануванні та виконанні завдань. Відновлення після невдачі – це важлива частина процесу розвитку та досягнення успіху.

2.2 Математичні засоби для оцінки прогнозування

Для оцінки та прогнозу дати завершення проєкту використовуються лінії тренду.

Лінія тренду є важливим інструментом в аналізі даних, який дозволяє виявити та прогнозувати загальний напрямок руху певного явища або набору

даних. Вона допомагає встановити зв’язок між змінними та визначити, чи існує тренд, якщо так, то якого типу. Далі буде докладніше описаний процес будування лінії тренду та різні види трендів.

Перш за все, буде описаний загальний процес будування лінії тренду. Щоб побудувати лінію тренду, нам потрібно мати набір даних, що складається з великої кількості спостережень або значень. Ми використовуємо ці дані для аналізу та знаходження загального зв’язку між ними [4].

Існує декілька способів побудови лінії тренду, але одним з найпоширеніших є метод найменших квадратів. Цей метод полягає в пошуку лінії, яка мінімізує суму квадратів відхилень між реальними значеннями і значеннями, які передбачає лінія тренду. Цей метод був обраний як самий простий, який підходить для завдання роботи.

Метод найменших квадратів є статистичним методом, що дозволяє знайти найкращу апроксимацію лінії для набору даних шляхом мінімізації суми квадратів відхилень між спостережуваними значеннями та прогнозованими значеннями лінії тренду.

У цьому коді використовуються формули найменших квадратів для визначення коефіцієнтів нахилу (slope) та зсуву (intercept) лінії тренду. Потім ці коефіцієнти використовуються для побудови лінії тренду на графіку.

Далі будується формула для побудови лінії тренду використовуючи метод найменших квадратів. Припускається, що є набір даних зі значеннями x і y. Мета – знайти лінію тренду у вигляді y = mx + b, де m – це коефіцієнт нахилу (slope), а b – це зміщення (intercept).

Формула для розрахунку цих значень є наступною:

$$m = (n * \Sigma(xy) - \Sigma x * \Sigma y) / (n * \Sigma(x^2) - (\Sigma x)^2),$$
$$b = (\Sigma y - m * \Sigma x) / n,$$

де n – кількість спостережень, Σ – сума, x – значення незалежної змінної, y – значення залежної змінної.

Ці формули дозволяють обчислити коефіцієнт нахилу (slope) та зміщення (intercept) для лінії тренду. Після того, як ми знайдемо ці значення, ми можемо використовувати їх для побудови лінії тренду на графіку [5].

У проєкті використовується саме метод найменших квадратів.

Залежно від того, як змінюються дані у наборі, можуть виникати різні типи трендів. Основні типи трендів включають наступні.

Позитивний тренд. В такому випадку значення залежної змінної збільшуються зі зростанням значення незалежної змінної. Це може вказувати на позитивну залежність між двома змінними, де збільшення однієї змінної спричинює збільшення іншої.

Негативний тренд. Тут значення залежної змінної зменшуються зі зростанням значення незалежної змінної. Це свідчить про негативну залежність між змінними, де збільшення однієї змінної призводить до зменшення іншої.

Горизонтальний тренд. У цьому випадку значення залежної змінної майже не змінюються незалежно від зміни незалежної змінної. Це може свідчити про відсутність залежності між двома змінними або про наявність сталої рівноваги.

Нелінійний тренд. Це сценарій, де залежність між змінними не може бути описана простою лінією. Тут можуть виникати складніші форми трендів, такі як параболічні, експоненціальні або логарифмічні.

На рисунку 2.1. зображена схема видів трендів.

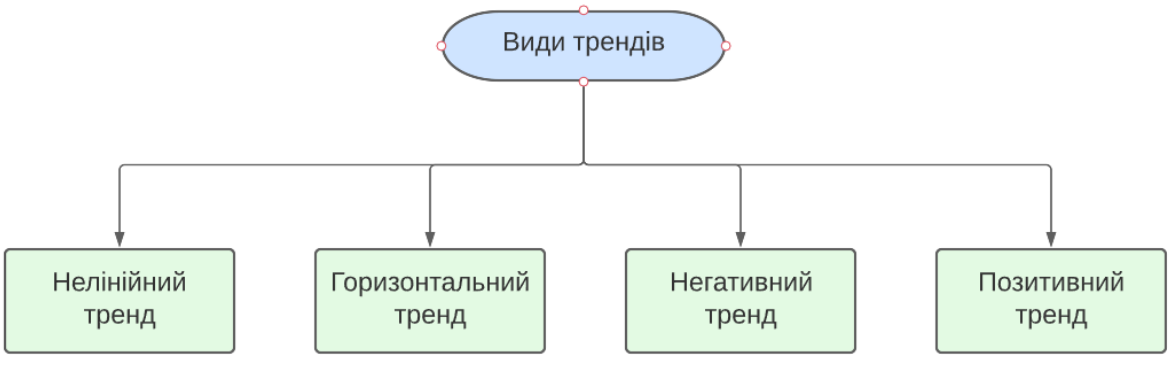


Рисунок 2.1 – Види трендів

Кожен тип тренду має свої характеристики та інтерпретацію. Важливо аналізувати дані та визначати, який тип тренду найкраще відповідає ваших даним [6].

Отже, лінія тренду є помічним інструментом для аналізу даних та прогнозування майбутніх значень. Вона допомагає з'ясувати, як змінюються дані з часом і встановлює зв'язок між ними. За допомогою формул та методу найменших квадратів ми можемо побудувати лінію тренду та використовувати її для прогнозування та прийняття рішень.

2.3 Технології реалізації

Microsoft Visual Studio 2022 - це інтегроване середовище розробки (IDE), яке надає розробникам зручні інструменти для створення програмного забезпечення. Вона є однією з найпопулярніших інструментальних платформ серед програмістів, і це не дивно, зважаючи на багатий функціонал та низку переваг, які надає це середовище.

Однією з ключових особливостей Microsoft Visual Studio 2022 є його широкий спектр підтримуваних мов програмування. Відомо, що це середовище підтримує мови програмування, такі як C#, Visual Basic, C++, F#, JavaScript, Python і багато інших. Це дає розробникам можливість працювати з різними мовами програмування в одному інтегрованому середовищі, що спрощує розробку складних проєктів та полегшує спільну роботу між різними командами [7].

Крім того, Microsoft Visual Studio 2022 надає широкий набір інструментів для побудови та налагодження програм. Зокрема, це включає в себе розвинений редактор коду з розширеною функціональністю, автоматичне завершення коду, підказки та перевірки помилок, що допомагають розробникам писати якісний код швидше та ефективніше. Крім того, візуальні інструменти для створення графічного інтерфейсу дозволяють швидко створювати зручні та привабливі користувацькі інтерфейси.

Ще одна перевага Microsoft Visual Studio 2022 – це його інтеграція з іншими сервісами та платформами Microsoft. Наприклад, розробники можуть легко підключати свої проєкти до хмарних сервісів, таких як Azure, що дозволяє швидко розгортати та масштабувати програми. Крім того, середовище має інтегровану систему керування версіями (Team Foundation Server), що спрощує спільну роботу над проєктами у команді та дозволяє ефективно керувати версіями коду.

Однією з найновіших інновацій Microsoft Visual Studio 2022 є підтримка розробки для платформи .NET 6. .NET 6 – це оновлена версія платформи розробки, яка пропонує швидшу та більш глибоку розробку програмного забезпечення для різних платформ, включаючи Windows, Linux та macOS. Завдяки підтримці .NET 6, розробники можуть використовувати останні технології та функціонал для створення сучасних додатків.

Узагалі, Microsoft Visual Studio 2022 є потужним та вдосконаленим інструментом для розробки програмного забезпечення. Він надає розробникам зручне та ефективне середовище для написання, налагодження та впровадження програм. Його широкий функціонал, підтримка різних мов програмування та інтеграція з іншими сервісами роблять його незамінним інструментом для професіоналів та початківців у сфері програмування [8].

Завдяки постійному розвитку та оновленням, Microsoft Visual Studio 2022 продовжує займати провідні позиції серед інтегрованих середовищ розробки та залишається незамінним інструментом для всіх, хто працює у сфері розробки програмного забезпечення.

Мова програмування C# є однією з найпопулярніших мов програмування у світі сучасного програмування. Розроблена компанією Microsoft, C# була представлена в 2000 році як основна мова програмування для платформи .NET. Протягом останніх двадцяти років вона набула широкого визнання і стала важливим інструментом для розробки різноманітних додатків та програмного забезпечення. Давайте розглянемо історію, особливості та переваги мови програмування C#.

Історія C# починається з бажання Microsoft створити мову програмування, яка б поєднувала силу C++ з простотою інтерпретованих мов, таких як Visual Basic. У 1999 році команда розробників, на чолі з Андерсом Гейлсбергом, розпочала роботу над новою мовою, яка отримала назву C#. C# була створена з метою розширення можливостей розробки програмного забезпечення на платформі .NET, а також для полегшення роботи розробників, що працюють з мовою C++ [9].

Однією з ключових особливостей мови C# є її синтаксис, який нагадує синтаксис мови C++ та Java. Це робить C# зрозумілою для програмістів, які вже знайомі з цими мовами. C# підтримує об'єктно-орієнтований підхід до програмування, де об'єкти взаємодіють один з одним через методи та властивості. Крім того, мова має вбудовану підтримку делегатів, подій, атрибутів та багато інших функцій, що допомагають розробникам створювати складні та цікаві додатки.

Ще однією вагомою перевагою C# є його висока продуктивність. Мова програмування C# компілюється в проміжний код, який виконується на віртуальній машині .NET. Це дозволяє досягти оптимальної продуктивності та ефективності роботи програми. Крім того, C# має вбудовану систему збирання сміття, яка автоматично вивільняє пам'ять, що дозволяє уникнути проблем з утриманням пам'яті та покращує швидкодію програм.

Окрім того, C# має велику кількість стандартних бібліотек, які надають розробникам широкі можливості. Наприклад, бібліотека Windows Forms дозволяє швидко створювати десктопні додатки з графічним інтерфейсом користувача, а ASP.NET дозволяє розробляти веб-додатки та веб-сервіси. Крім того, наявність різних сторонніх бібліотек та фреймворків робить C# ще більш привабливою для розробки різноманітних застосунків [10].

Завдяки активній підтримці Microsoft, мова C# постійно розвивається та оновлюється. Вона має велику спільноту розробників, яка активно співпрацює, ділиться знаннями та надає підтримку одне одному. Це робить C# ідеальним вибором для початківців та професіоналів у світі програмування.

Загалом, мова програмування C# виявилася чудовим інструментом, який дозволяє розробникам створювати якісне та ефективне програмне забезпечення. Її синтаксис, продуктивність, широкий функціонал та активна спільнота розробників роблять її важливим інструментом для будь-якого, хто зацікавлений у програмуванні. C# продовжує займати провідну позицію серед мов програмування та залишається вибором багатьох розробників по всьому світу.

Платформа Windows Forms (WinForms) є однією з найпопулярніших та найширше використовуваних платформ для розробки десктопних додатків у середовищі Windows. WinForms надає розробникам зручні інструменти для створення графічних інтерфейсів користувача та функціональних можливостей для їх взаємодії.

Однією з головних переваг платформи WinForms є її простота в використанні. Завдяки візуальному дизайнеру, розробники можуть швидко створювати і налаштовувати елементи інтерфейсу користувача, такі як кнопки, поля введення, списки та інші, простим перетягуванням і розміщенням на формі. WinForms надає розширені можливості для керування властивостями та подіями елементів, що дозволяє розробникам налаштовувати поведінку програми за допомогою коду.

Ще однією перевагою WinForms є його висока сумісність з платформою Windows. Додатки, розроблені за допомогою WinForms, легко запускаються та працюють на різних версіях операційної системи Windows без необхідності внесення значних змін або адаптації. Це дозволяє розробникам швидко поширювати свої додатки серед користувачів, нехтуючи особливостями різних версій OS.

WinForms також надає широкий набір вбудованих елементів керування, які спрощують створення функціональних інтерфейсів користувача. Вона підтримує різні типи вікон, меню, панелі, вкладки та інші компоненти, що дозволяють розробникам створювати багатофункціональні додатки зі зручним та зрозумілим інтерфейсом.

Однією з особливостей WinForms є можливість розробки додатків мовою програмування C#. Це означає, що розробники можуть використовувати всю функціональність мови C#, таку як об’єктно-орієнтований підхід, делегати, події та багато іншого, для створення розширених та ефективних додатків.

WinForms також підтримує можливість використання сторонніх компонентів і бібліотек, що розширює його функціональність і дозволяє розробникам використовувати готові рішення для реалізації певних функцій у своїх додатках [11].

Загалом, платформа WinForms є інструментом для розробки десктопних додатків у середовищі Windows. Її простота в використанні, сумісність з платформою Windows, широкий набір елементів керування та підтримка мови програмування C# роблять її відмінним вибором для розробників, які шукають зручний інструмент для створення десктопних додатків.

2.4 Висновки до другого розділу

У другому розділі була розглянута важливість планування та розробки чіткого плану дій для досягнення успіху. План дій є орієнтиром і дорожньою картою, яка спрямовує нас до мети. Він допомагає створити конкретні цілі, розподілити ресурси та визначити пріоритети. Планування забезпечує систематичний підхід та ефективне управління часом. Воно допомагає передбачити складнощі та ризики, що дозволяє готуватися до них. Проте, важливо бути гнучким та готовим до змін. Використання невдач як можливості для розвитку є важливою частиною процесу досягнення успіху. Отже, розробка та виконання чіткого плану дій є необхідною умовою для досягнення бажаного результату в будь-якій сфері діяльності.

Також була приділена увага використанню ліній тренду для оцінки та прогнозу дати завершення проєкту. Лінія тренду є важливим інструментом аналізу даних, що допомагає встановити загальний напрямок руху та

прогнозувати майбутні значення. Побудова лінії тренду включає визначення зв’язку між змінними та застосування методу найменших квадратів. Цей метод дозволяє знайти лінію, яка мінімізує відхилення між реальними та передбаченими значеннями. Для цього обчислюються коефіцієнт нахилу (slope) та зміщення (intercept) за допомогою спеціальних формул. Залежно від змінення даних, можуть виникати різні типи трендів, такі як позитивний, негативний, горизонтальний та нелінійний. Кожен тип тренду має свої характеристики та інтерпретацію. Розуміння та аналіз цих трендів допомагає прогнозувати та приймати рішення. Лінія тренду є корисним інструментом для аналізу даних та прогнозування майбутніх значень у проєктах та інших областях діяльності.

Мова C# обрана як високо функціональна технологія розробки з високою продуктивністю та широким функціоналом. Наявність стандартних бібліотек та підтримка фреймворків дозволяють швидко створювати графічні та веб-додатки. Активна підтримка Microsoft забезпечує постійні оновлення та розвиток мови, роблячи її гнучкою та сучасною.

РОЗДІЛ 3.
ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОДУКТУ ДЛЯ ОЦІНКИ ТА ПРОГНОЗУВАННЯ ПРОЄКТІВ

3.1 Архітектура програмної системи

WinForms – це технологія, яка дозволяє розробляти клієнтські додатки для операційної системи Windows, використовуючи мову програмування C# або Visual Basic.NET. Архітектура додатка на WinForms базується на шаблоні Model-View-Controller або Model-View-Presenter, хоча існують й інші підходи, які можуть бути використані [12].

Основні складові архітектури додатка на WinForms включають:

Модель: представляє логіку бізнес-процесів, дані та операції, пов’язані з обробкою цих даних. Вона може включати класи, структури, інтерфейси та інші компоненти, які відповідають за збереження та маніпулювання даними. Модель не залежить від інтерфейсу користувача.

Представлення: відповідає за графічний інтерфейс користувача (GUI) та його відображення. Воно складається з форм, елементів керування (наприклад, кнопок, текстових полів, списків тощо) та інших компонентів, які відображають дані та дозволяють користувачеві взаємодіяти з додатком.

Контролер або презентер: компонент використовується для координації взаємодії між моделлю та представленням. Він обробляє події, які виникають у представленні (натискання кнопок, введення тексту тощо), і взаємодіє з моделлю для отримання або оновлення даних. Контролер також може включати додаткову логіку, пов’язану з обробкою даних перед передачею їх до моделі або представлення [13].

Програма складається з трьох вкладок:

1. Початкова вкладка з таблицею План та кнопками керування.
2. Таблиця з формулами для розрахунків спринтів та даних для графіків.
3. Вкладка з двома графіками.

В програмі є робота з Екселем та PDF файлами. Для реалізації усіх функції потрібні були бібліотеки Microsoft.Office.Interop.Excel та iTextSharp

Бібліотека Microsoft.Office.Interop.Excel є популярним інструментом для роботи з електронними таблицями у програмі Excel. Ця бібліотека надає розширені можливості для автоматизації рутинних завдань, маніпулювання даними та створення складних звітів і документів. У цій доповіді ми розглянемо основні функції та можливості цієї бібліотеки [14].

Microsoft.Office.Interop.Excel є інтероперабельною бібліотекою, що дозволяє взаємодіяти з програмою Excel з допомогою мови програмування, таких як C# або Visual Basic.NET. Ця бібліотека надає доступ до об’єктів Excel, таких як робочі книги, аркуші, діапазони, клітинки тощо, що дозволяє виконувати різноманітні операції з даними.

Однією з ключових можливостей бібліотеки є зчитування та запис даних у електронні таблиці Excel. За допомогою методів і властивостей цієї бібліотеки можна зчитувати значення з клітинок, записувати нові значення, форматувати дані, створювати діаграми, застосовувати фільтри та багато іншого. Це дозволяє автоматизувати процес обробки даних у програмі Excel і значно збільшити продуктивність роботи.

Додатково, Microsoft.Office.Interop.Excel дозволяє працювати з формулами у електронних таблицях. За допомогою бібліотеки можна виконувати обчислення, вставляти формули у клітинки, обчислювати значення формул та отримувати результати. Це відкриває широкі можливості для автоматизації складних обчислень та аналізу даних.

Окрім роботи з даними, Microsoft.Office.Interop.Excel дозволяє керувати зовнішнім виглядом електронних таблиць. Ви можете змінювати шрифти, кольори, стилі, розташування та багато іншого. Також можна створювати звіти з використанням різних форматувань, включаючи таблиці, графіки, діаграми, заголовки та підзаголовки. Це дозволяє створювати професійні та зрозумілі візуальні представлення даних.

У підсумку, бібліотека Microsoft.Office.Interop.Excel є чудовим інструментом для автоматизації роботи з електронними таблицями у програмі Excel. Вона надає розширені можливості для маніпулювання даними, створення звітів та документів, роботи з формулами та виглядом таблиць. Використання цієї бібліотеки дозволяє підвищити ефективність роботи з даними та забезпечити швидкий та точний аналіз інформації у програмі Excel.

1. Вкладка План (рисунок 3.1.)

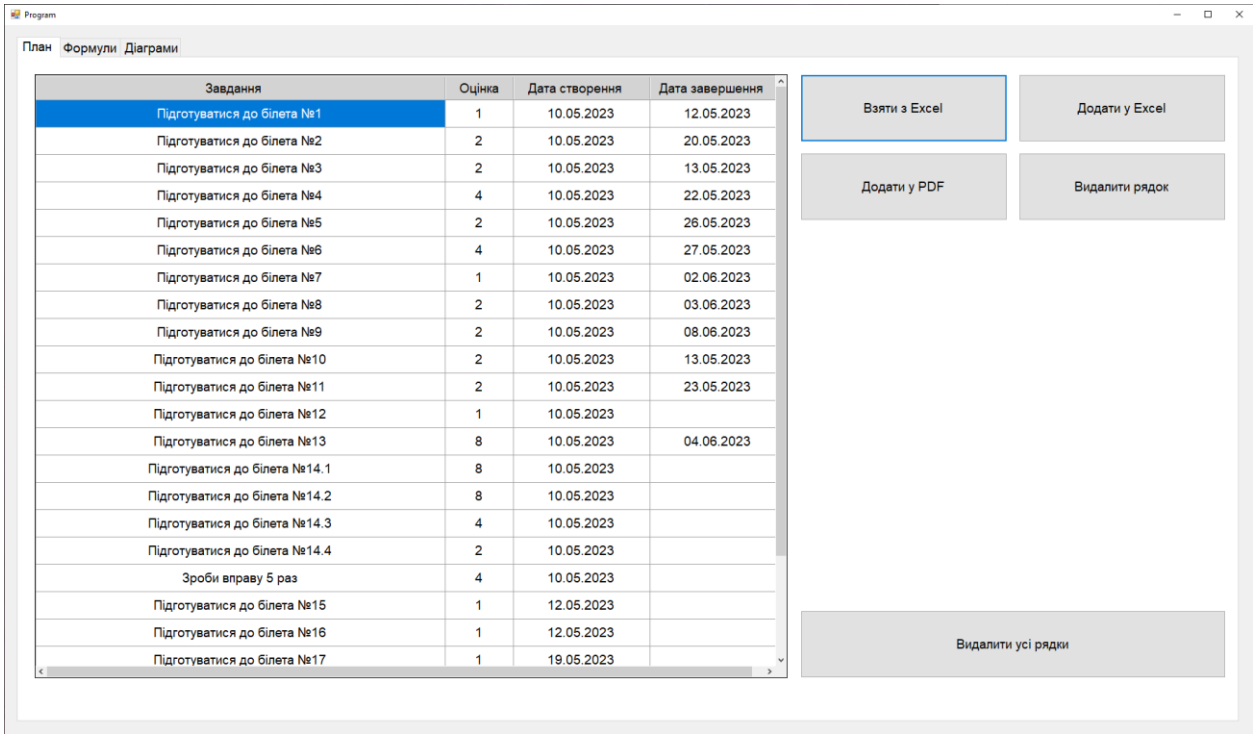


Рисунок 3.1 – Вигляд вкладки План

Основним елементом цієї вкладки є таблиця dgvPlan. Вона існує для додавання начальних даних. Для зручності користувача є функція Імпорту даних з Excel (дивись лістінг у Додатку).

Значення комірок перетворюються у потрібний формат (наприклад, дату) і додаються до DataGridView з використанням dgvPlan.Rows.Add(), (дивись лістінг у Додатку).

Після завершення зчитування даних, закривається робоча книга Excel та програма Excel за допомогою workbook.Close() та excelApp.Quit().

Звільняються ресурси, пов’язані з роботою з Excel, за допомогою Marshal.ReleaseComObject().

Ще однією функцією є експорт даних до файлу PDF.

Відкриття діалогового вікна для збереження файлу: створюється діалогове вікно для вибору розташування та імені файлу PDF з використанням SaveFileDialog (дивись лістінг у Додатку). Користувач може обрати файл та його місце збереження.

Заповнення таблиць даними: за допомогою вкладених циклів for, програма проходиться по рядках та стовпцях джерел даних (наприклад, dgvPlan). Значення комірок отримується за допомогою dgvPlan[j, i].Value і додається до відповідної комірки таблиці.

Закриття документа та збереження файлу: викликається метод document.Close() для закриття документа. Файл PDF зберігається за допомогою FileStream у вибраному місці та відкривається для перегляду за допомогою System.Diagnostics.Process.Start(filePath).

Четверта кнопка на сторінці видаляє один обраний рядок. Остання кнопка очищує усі таблиці.

Як візуально працює імпорт на експорт даних буде показано у пункті 3.3.

Для функції екпорту до PDF файлу було використано бібліотеку iTextSharp.

iTextSharp є високоефективною бібліотекою, спеціально розробленою для роботи з PDF-документами у середовищі .NET. Ця бібліотека надає широкий набір інструментів для створення, редагування та зміни PDF-файлів з безліччю можливостей. Використовуючи iTextSharp, ви зможете повністю контролювати вміст і структуру PDF-документів. Вона дозволяє додавати, видаляти та редагувати сторінки, вставляти текст, зображення та інші елементи, налаштовувати властивості та параметри PDF-файлів з легкістю і точністю. Незалежно від типу вашого .NET-застосунку, iTextSharp є незамінним інструментом для роботи з PDF-документами, що дозволяє вам

зручно і ефективно створювати, змінювати та управляти PDF-файлами для вашого проєкту [15].

Основні властивості iTextSharp включають:

Створення PDF. За допомогою iTextSharp можна створювати нові PDF-документи з нуля. Можна додавати текст, зображення, таблиці, гіперпосилання та інші елементи в PDF-файл.

Редагування та модифікація PDF: iTextSharp дозволяє змінювати вміст і структуру існуючих PDF-документів. Є функція додавання або видалення сторінки, змінювати шрифти, кольори, розміри та інші властивості елементів PDF [16].

Обробка форм: iTextSharp надає можливість створювати та заповнювати PDF-форми. Ви можете додавати текстові поля, положення вибору, перемикачі, списки та інші елементи форми до PDF-документа.

Робота з шрифтами та графічними елементами: iTextSharp має підтримку різних типів шрифтів і графічних елементів. Ви можете використовувати спеціальні шрифти для відображення тексту у PDF-документі, а також додавати зображення, графіки та інші графічні елементи.

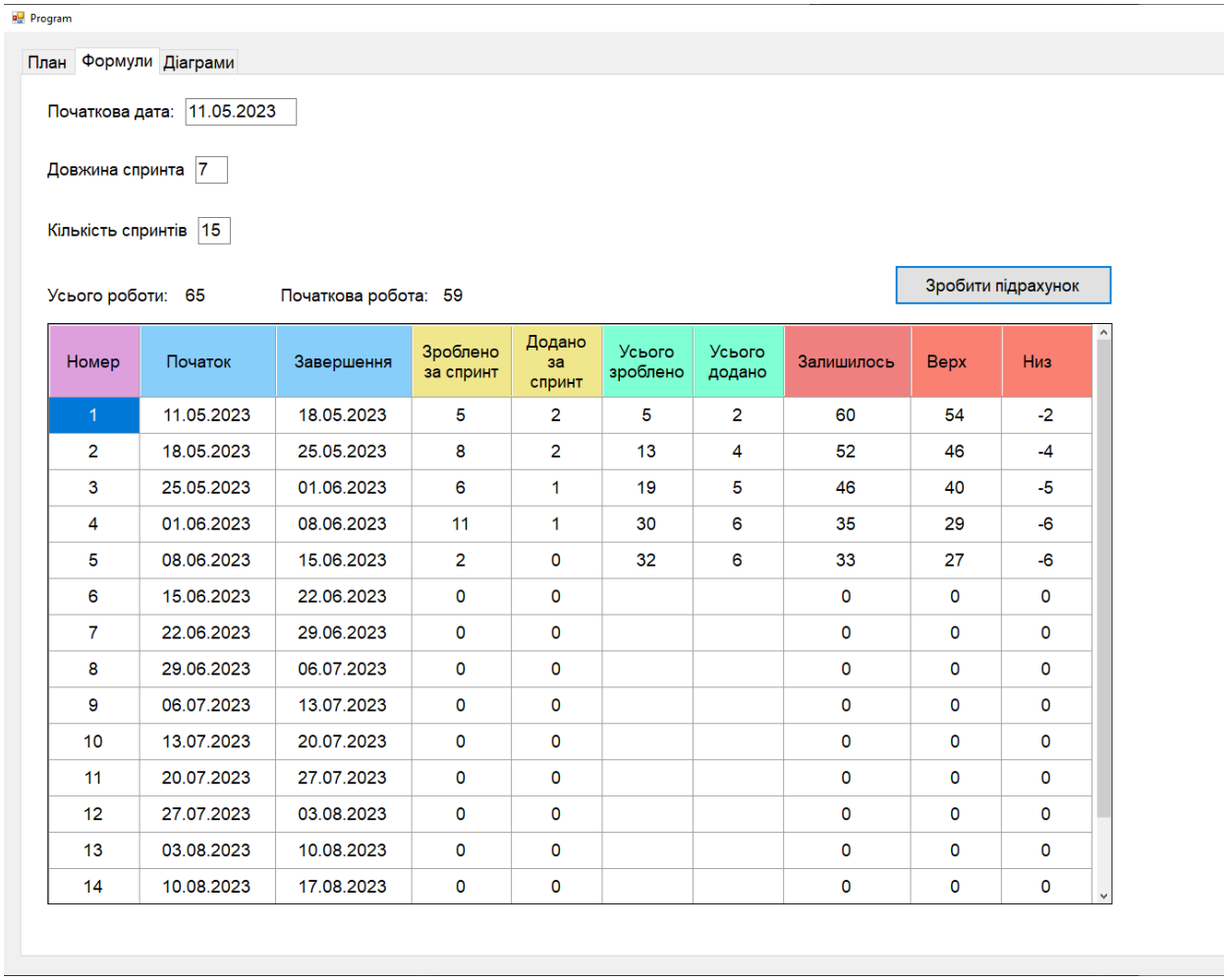
Захист та шифрування: iTextSharp дозволяє захищати PDF-документи за допомогою різних методів шифрування. Ви можете встановлювати паролі для обмеження доступу до вмісту PDF, а також шифрувати документи, щоб забезпечити їх безпеку.

Експорт та імпорт даних: iTextSharp дозволяє експортувати дані з PDF-документів у різні формати, такі як текстові файли, HTML-документи або зображення. Ви також можете імпортувати дані з інших джерел у PDF-документи [17].

Бібліотека iTextSharp також підтримує розширені можливості для обробки тексту в PDF-файлах. Ви можете використовувати різні шрифти, керувати вирівнюванням, розмірами і стилями тексту, а також застосовувати спеціальні ефекти, такі як підкреслення, заштрихування або перекреслення.

Крім того, iTextSharp дозволяє вставляти графічні елементи у PDF-документи. Ви можете додавати зображення, малюнки, графіки і навіть відео, щоб створювати контент у своїх PDF-документах [18].

2. Вкладка Формули (рисунок 3.2).



Номер	Початок	Завершення	Зроблено за спринт	Додано за спринт	Усього зроблено	Усього додано	Залишилось	Верх	Низ
1	11.05.2023	18.05.2023	5	2	5	2	60	54	-2
2	18.05.2023	25.05.2023	8	2	13	4	52	46	-4
3	25.05.2023	01.06.2023	6	1	19	5	46	40	-5
4	01.06.2023	08.06.2023	11	1	30	6	35	29	-6
5	08.06.2023	15.06.2023	2	0	32	6	33	27	-6
6	15.06.2023	22.06.2023	0	0			0	0	0
7	22.06.2023	29.06.2023	0	0			0	0	0
8	29.06.2023	06.07.2023	0	0			0	0	0
9	06.07.2023	13.07.2023	0	0			0	0	0
10	13.07.2023	20.07.2023	0	0			0	0	0
11	20.07.2023	27.07.2023	0	0			0	0	0
12	27.07.2023	03.08.2023	0	0			0	0	0
13	03.08.2023	10.08.2023	0	0			0	0	0
14	10.08.2023	17.08.2023	0	0			0	0	0

Рисунок 3.2 – Вигляд заповненої таблиці

Ця вкладка створена для конфігурації спринтів. Спринт – це одиниця часу, етап, у якому буде виконуватися робота. Користувач задає вхідні параметри, а саме: початкову дату, довжину спринту та кількість спринтів. Програма на основі цих даних вираховує усю інформацію у таблиці, а також показує обсяг поточної та початкової роботи. Більш детально, що означають кожні стовпці буде наведено у пункті 3.3. Далі дана інформація як обчислювалися деякі стовпці.

3. Вкладка Діаграми (рисунок 3.3).

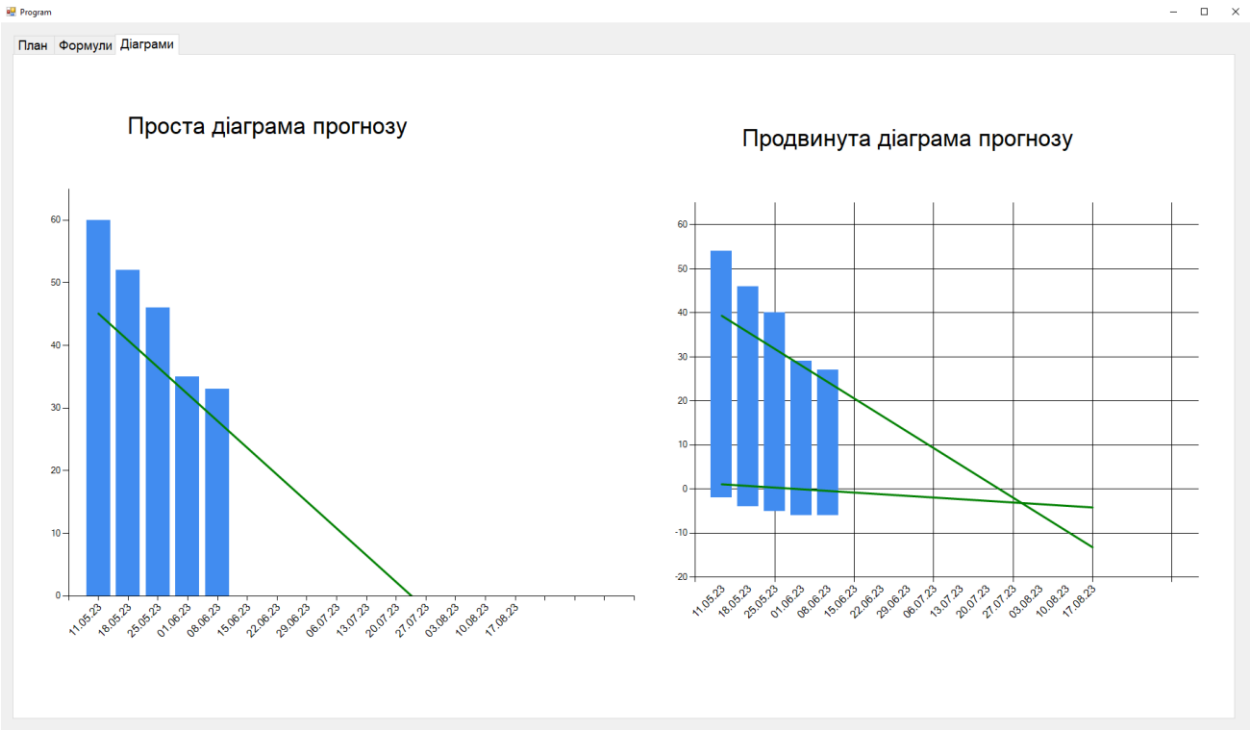


Рисунок 3.3 – Вигляд вкладки Діаграми

На цій вкладці користувач може побачити графіки, які були побудовані за даними з таблиці dgvFormulas, що знаходиться у розділі Формули. Принципи, за якими вони будуються описані у пункті 3.3.

Для наочності показано як будується перший графік, та лінія тренду на ньому.

У початковій частині коду встановлюються параметри для осей графіка. Наприклад, для осі Y (вертикальна вісь) встановлюються максимальне та мінімальне значення, інтервал, а також відключається показ сітки. А для осі X (горизонтальна вісь) встановлюються інтервал, формат міток, шрифт та нахил тексту міток.

Далі, створюється область графіка, яка визначає його положення та розміри. Після цього, використовуючи дані з dgvFormulas, налаштовуються мітки на осі X. Кожна мітка створюється на основі дати з першої комірки рядка та додається до осі X (дивись лістинг у Додатку).

Блок-схема будування графіку зображена на рисунку 3.4.

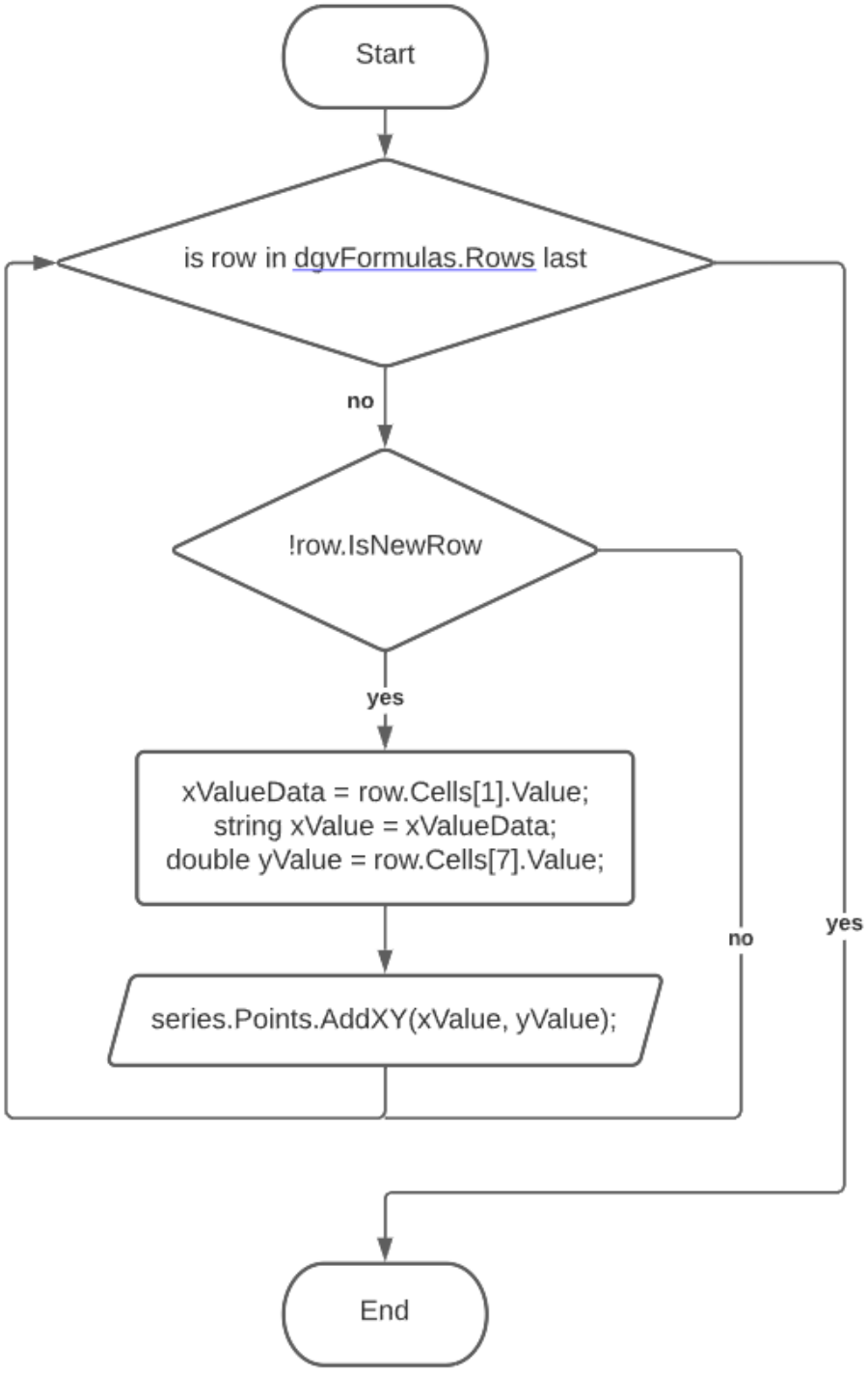


Рисунок 3.4 – Блок-схема будування графіку

Лінійний тренд. Спочатку створюється серія під назвою "TrendLine". Ця серія встановлюється з типом графіка Line, зеленим кольором та товщиною лінії 3.

Далі, встановлюються параметри для осей графіка. Мінімальне значення по осі X (xAxis.Minimum) встановлюється як 0, інтервал (xAxis.Interval) – як 1. Максимальне значення по осі Y (yAxis.Maximum) встановлюється на основі значення lbTotal_Work.Text, а мінімальне значення – як 0 (дивись лістинг у Додатку).

Для побудови лінії тренду використовується список значень values, який отримується зі значень стовпця 7 (row.Cells[7].Value) в dgvFormulas. Цей список містить лише числові значення (дивись лістинг у Додатку).

Після цього обчислюються суми (sumX, sumY, sumXY, sumXX) для подальшого обчислення нахилу (slope) та перетину (intercept) лінії тренду.

Мінімальне та максимальне значення для лінії тренду визначаються на основі мінімального та максимального значень по осі X (minX, maxX) та обчисленого нахилу та перетину.

На останок, серія "TrendLine" додається до графіка chartFirst_diagram, а дві точки з координатами (maxX, maxY) та (minX, minY) додаються до цієї серії для побудови лінії тренду (дивись лістинг у Додатку).

Так виглядає загальна блок-схема програми (рисунок 3.5.):

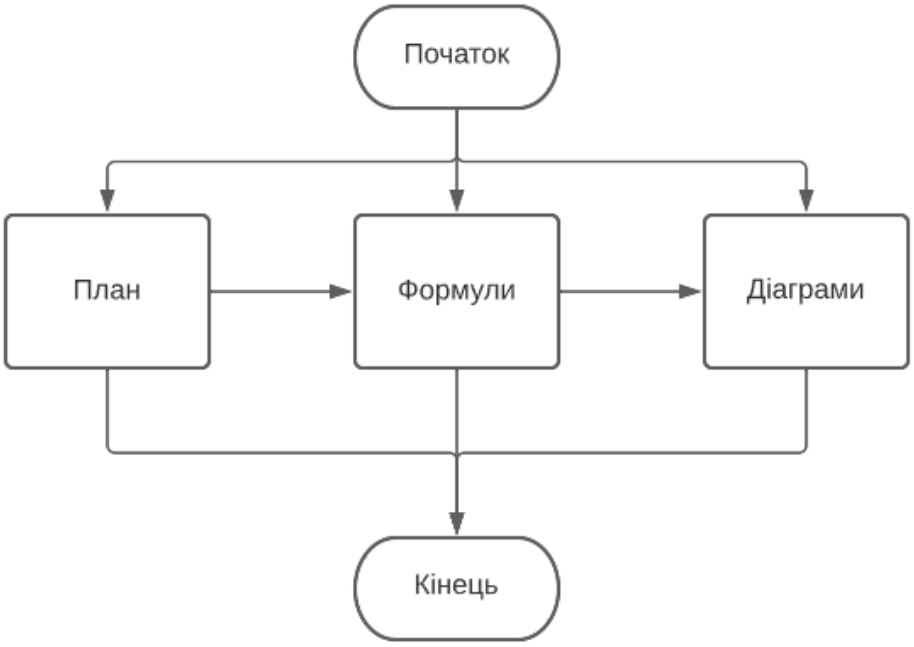


Рисунок 3.5 – Блок-схема всієї програми

Діаграма класів зображена на рисунку 3.6.

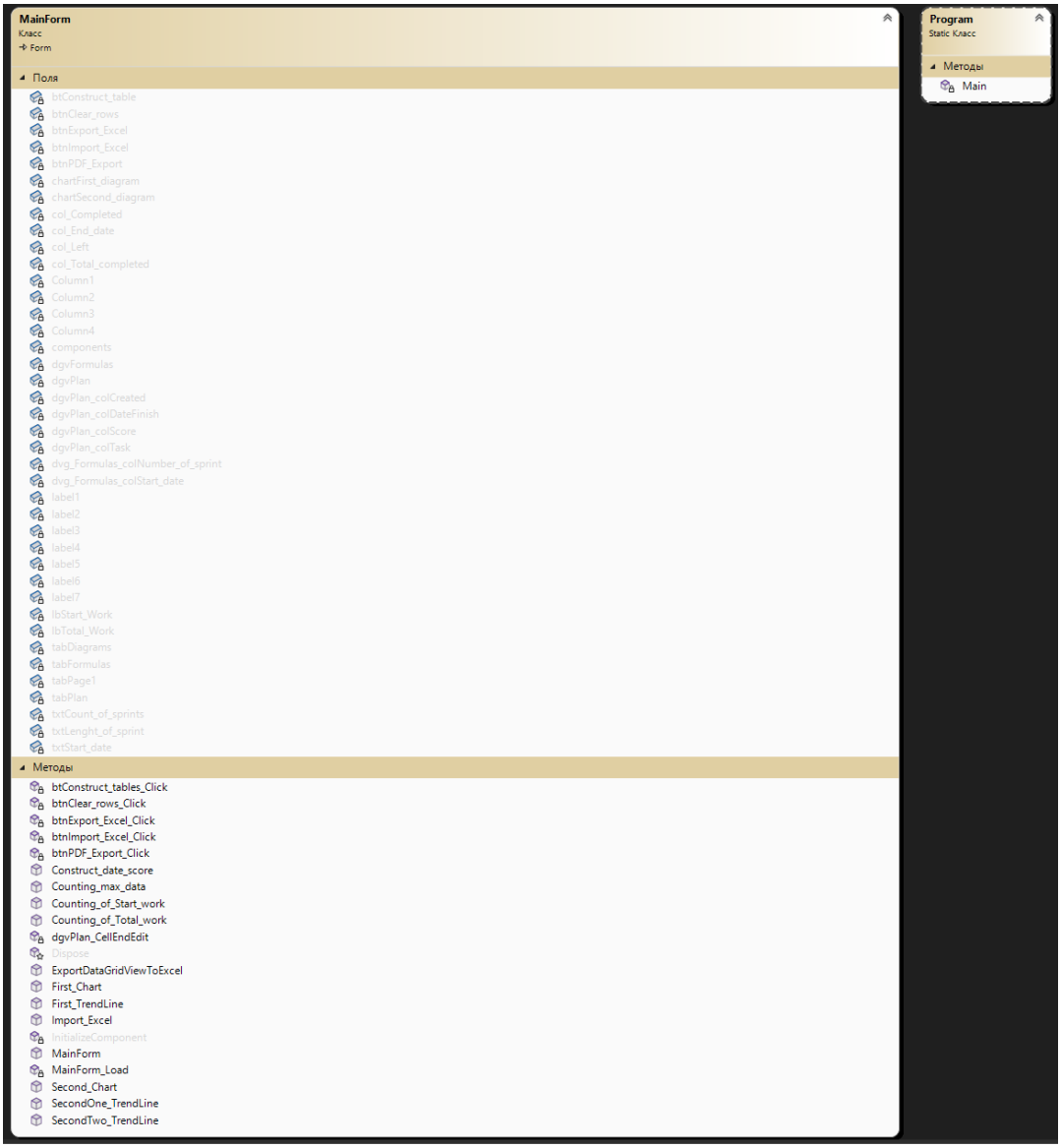


Рисунок 3.6 – Вигляд діаграми класів

3.2 Інтерфейс користувача

Проєктування користувацького інтерфейсу (UI) є критичним етапом розробки будь-якого програмного продукту. Ефективний та зручний інтерфейс забезпечує успішну взаємодію з користувачем та покращує загальний досвід використання продукту. У цьому дописі ми розглянемо основні стилі проєктування, а також принципи та кращі практики для побудови інтерфейсу, що задовольняє потреби користувачів [19].

Основні стилі проєктування:

– мінімалізм. Ставлення «менше – це краще» передбачає спрощення інтерфейсу шляхом видалення зайвих деталей та сконцентрування на основних функціях. Мінімалістичний дизайн створює чистий та елегантний вигляд інтерфейсу;

– матеріальний дизайн. Розроблений компанією Google, цей стиль використовує реалістичні тіні, підсвітки та анімацію для створення привабливого та інтуїтивно зрозумілого інтерфейсу. Він підкреслює важливість простору, ієрархії та контексту;

– плоский дизайн. Цей стиль, розповсюджений після випуску операційної системи Windows 8, характеризується відсутністю тіней, градієнтів та текстур. Він пропонує чистий та сучасний вигляд, спрощує елементи інтерфейсу та полегшує сприйняття інформації;

– типографічний дизайн. Зосереджений на використанні шрифтів, цей стиль створює інтерфейс зрозумілим та привабливим за допомогою правильного вибору шрифтів, їх розміру, кольору та розташування.

Принципи побудови зручного інтерфейсу:

– контекстуальність. Інтерфейс повинен забезпечувати зрозумілість та взаємозв’язок з контекстом використання. Показуйте користувачеві тільки необхідну інформацію, що стосується його поточної дії;

– простота та ясність. Уникайте перевантаження інтерфейсу зайвими елементами. Використовуйте зрозумілу та доступну мову, інтуїтивно зрозумілі піктограми та іконки;

– консистентність. Забезпечте однаковий стиль та поведінку елементів інтерфейсу на всіх сторінках та вікнах. Це дозволяє користувачеві швидше зорієнтуватися та впевнено використовувати програмний продукт;

– відповідність контексту. Призначайте відповідні функціональні кнопки, меню та елементи керування для виконання конкретних завдань у відповідних місцях інтерфейсу.

– зручність навігації. Забезпечте легку навігацію по програмному продукту. Використовуйте ієрархічну структуру, зрозумілі назви розділів та зручні посилання між сторінками [20].

Правильне проєктування користувацького інтерфейсу є ключовим чинником успіху будь-якого програмного продукту. Використання певного стилю дизайну, такого як мінімалізм, матеріальний дизайн або плоский дизайн, може покращити.

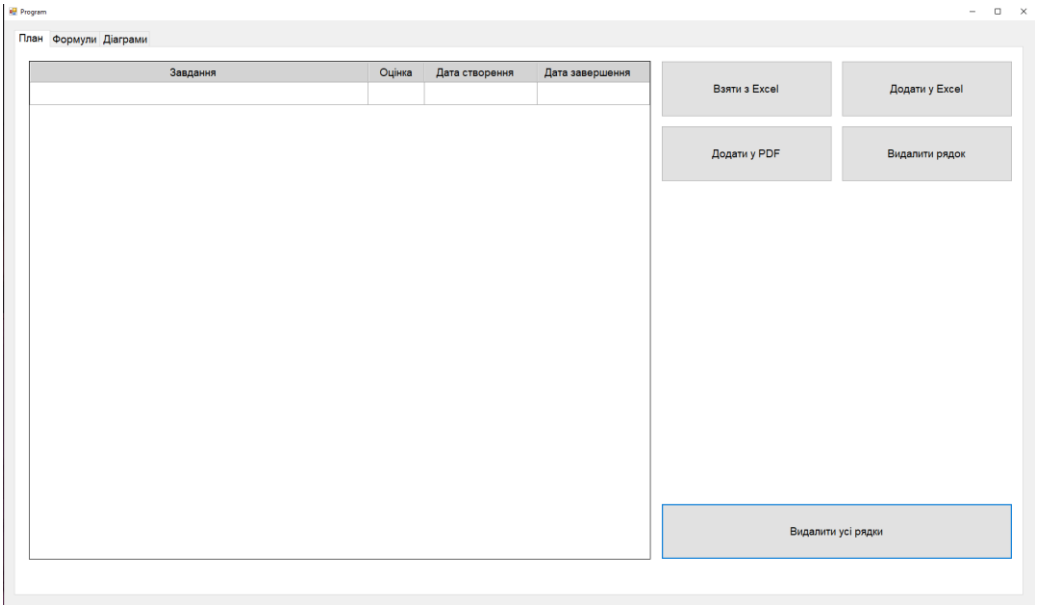


Рисунок 3.7 – Вигляд початкової вкладки

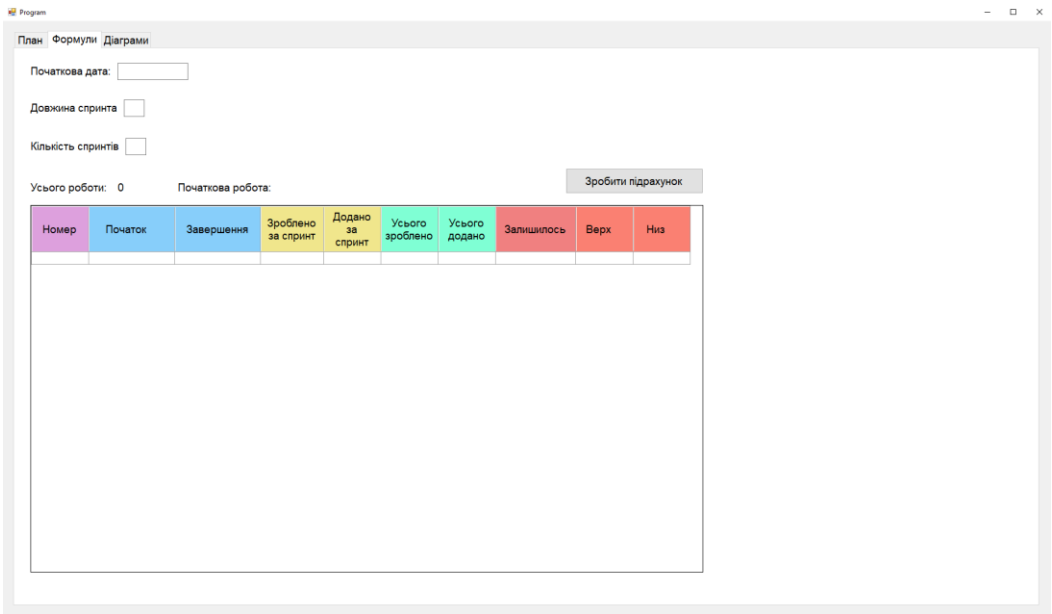


Рисунок 3.8 – Вигляд другої вкладки

На рисунку 3.7 зображений вигляд першої вкладки без заповненої таблиці План. Це початковий екран програми, на якому опиняється користувач після запуску додатку. Рисунок 3.8 показує вид другої вкладки Формули, де будуть заповнятися необхідні поля та показуватися розрахована таблиця Формули.

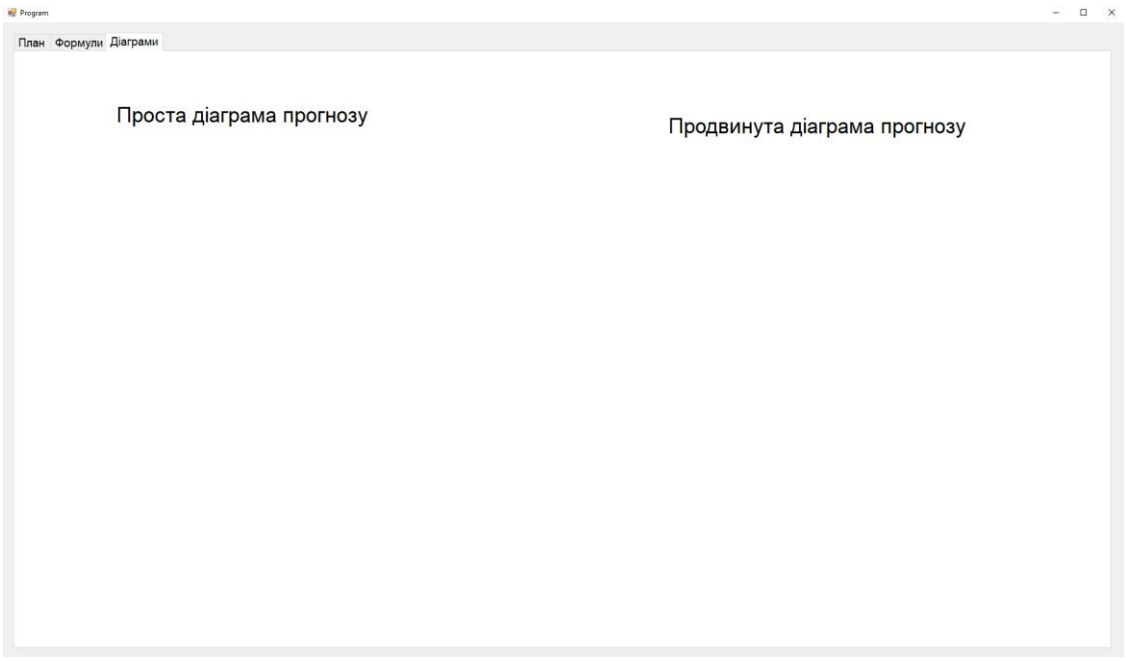


Рисунок 3.9 – Вигляд вкладки з графіками

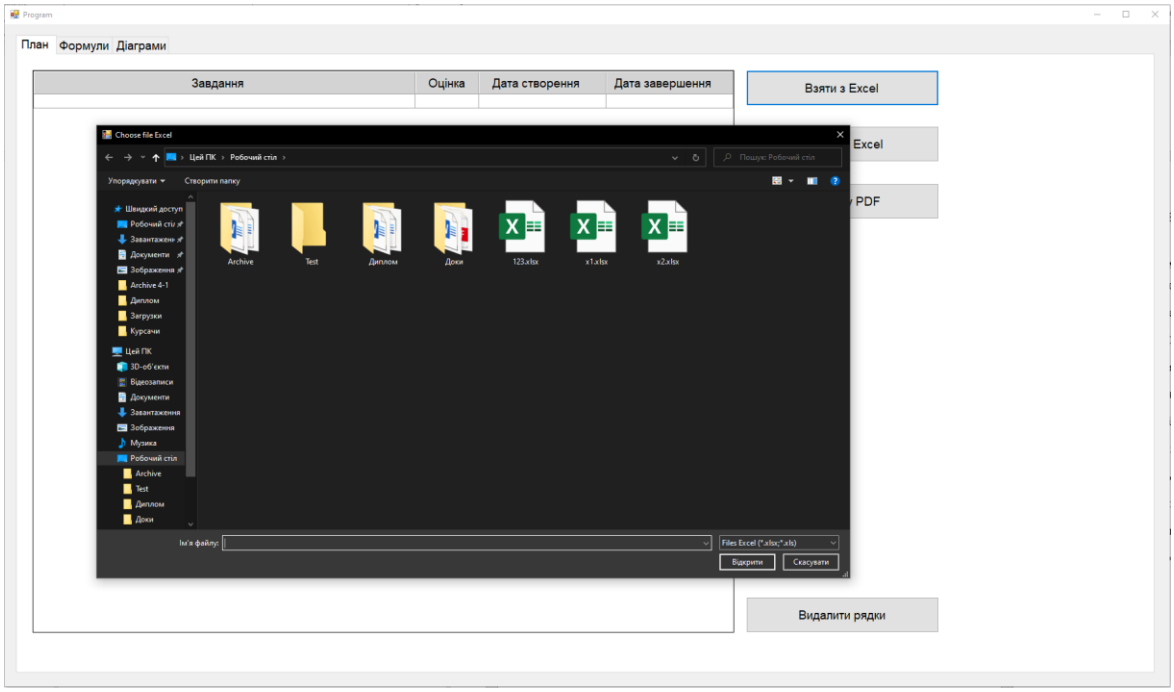


Рисунок 3.10 – Вікно імпорту з Excel

На скріншотах, що зображені на рисунках 3.9 та 3.10, показується вікно третьої вкладки Діаграми з ще пустими графіками та вікно файлового провідника для імпорту даних з Excel. На третій вкладці після заповнення таблиці будуть зображені два графіка прогнозу. Вікно імпорту дозволяє обрати користувачу будь-який файл у форматі xlsx або xls.

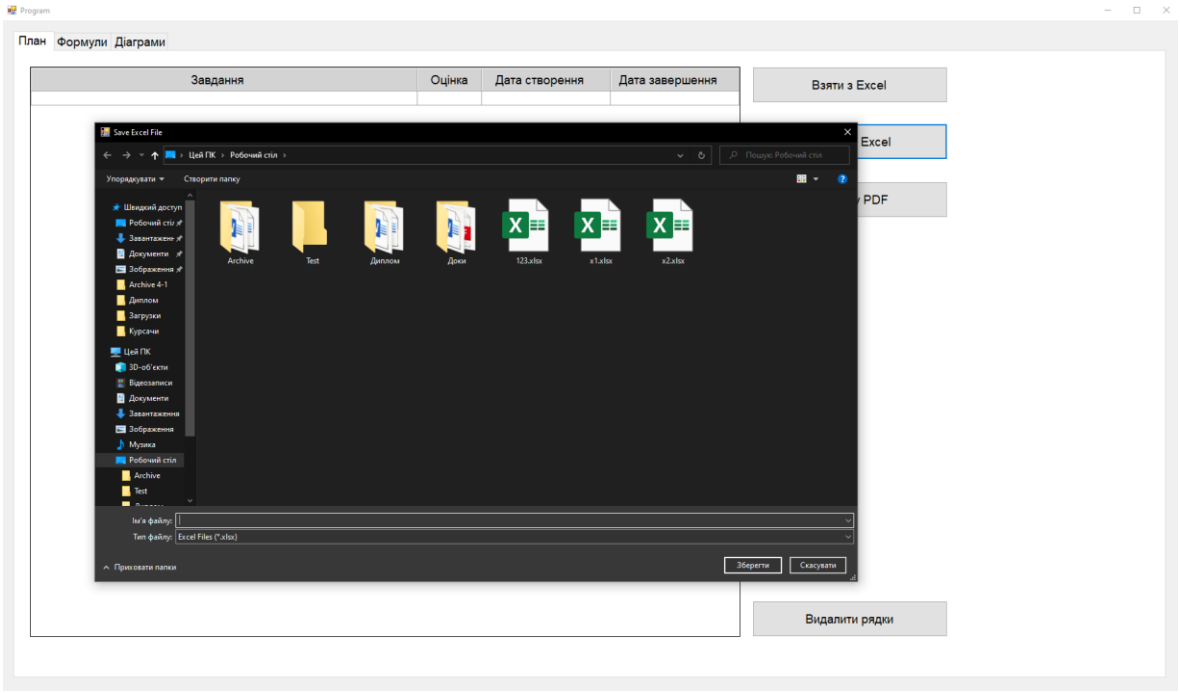


Рисунок 3.11 – Вікно експорту у Excel

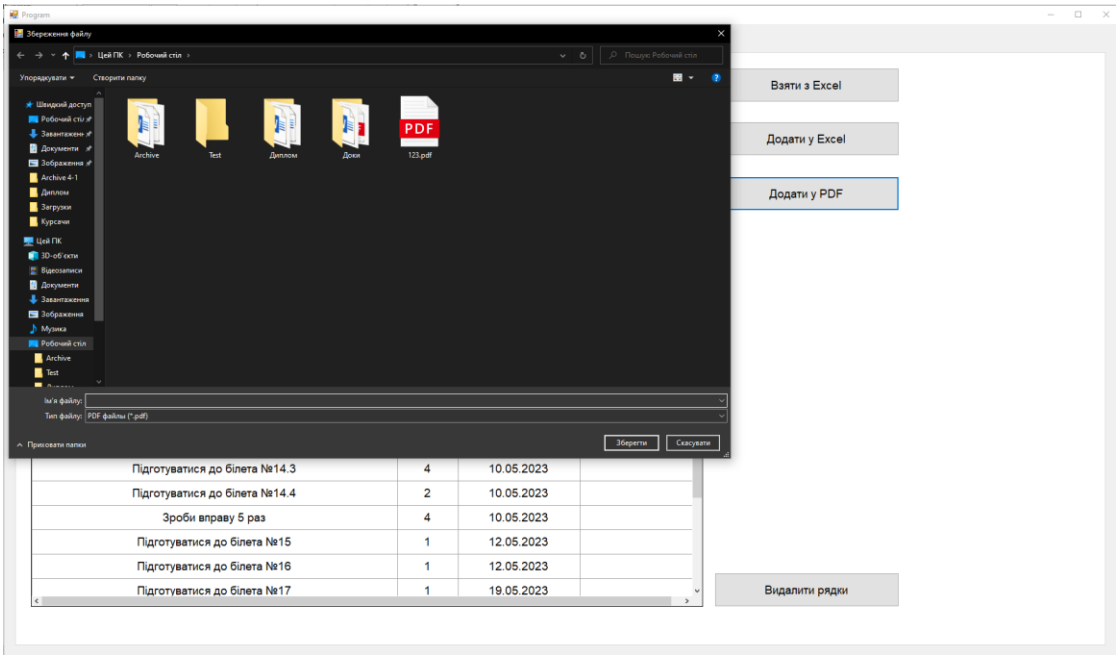
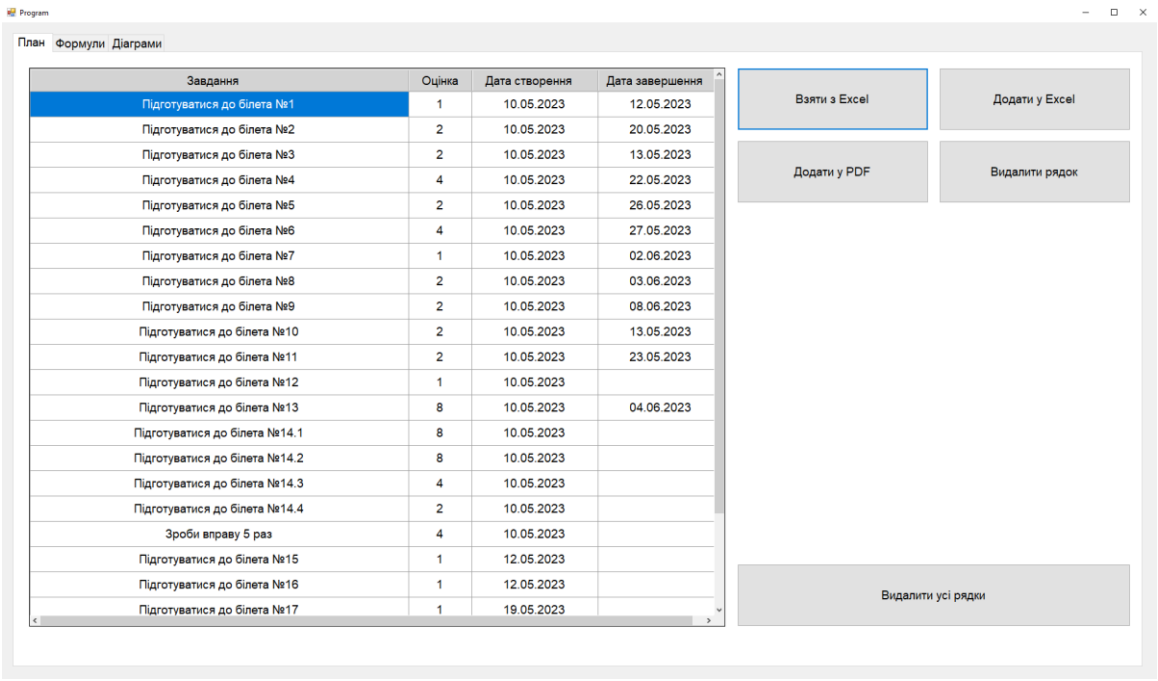


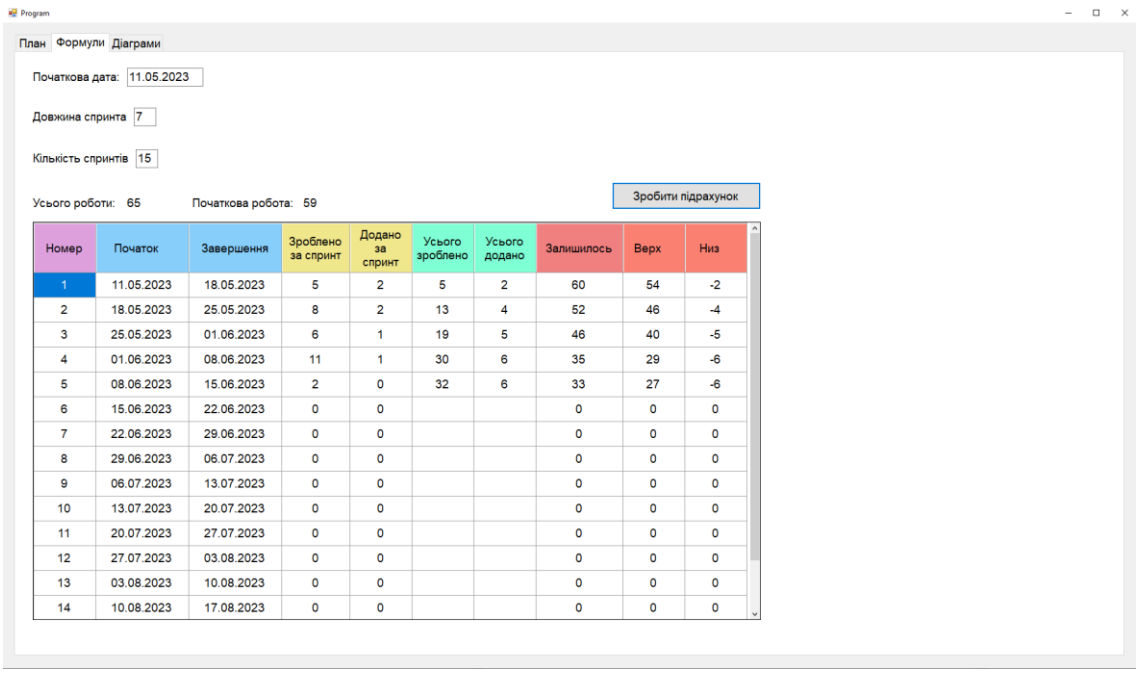
Рисунок 3.12 – Вікно експорту у PDF

Рисунки 3.11. та 3.12. показують вікна експорту даних таблиць План та Формули у файли Ексель та PDF. Користувач може обрати існуючий файл для перепису або створити новий. Передбачений фільтр формату файлу для коректного завантаження.



Завдання	Оцінка	Дата створення	Дата завершення
Підготуватися до білета №1	1	10.05.2023	12.05.2023
Підготуватися до білета №2	2	10.05.2023	20.05.2023
Підготуватися до білета №3	2	10.05.2023	13.05.2023
Підготуватися до білета №4	4	10.05.2023	22.05.2023
Підготуватися до білета №5	2	10.05.2023	26.05.2023
Підготуватися до білета №6	4	10.05.2023	27.05.2023
Підготуватися до білета №7	1	10.05.2023	02.06.2023
Підготуватися до білета №8	2	10.05.2023	03.06.2023
Підготуватися до білета №9	2	10.05.2023	08.06.2023
Підготуватися до білета №10	2	10.05.2023	13.05.2023
Підготуватися до білета №11	2	10.05.2023	23.05.2023
Підготуватися до білета №12	1	10.05.2023	
Підготуватися до білета №13	8	10.05.2023	04.06.2023
Підготуватися до білета №14.1	8	10.05.2023	
Підготуватися до білета №14.2	8	10.05.2023	
Підготуватися до білета №14.3	4	10.05.2023	
Підготуватися до білета №14.4	2	10.05.2023	
Зроби вправу 5 раз	4	10.05.2023	
Підготуватися до білета №15	1	12.05.2023	
Підготуватися до білета №16	1	12.05.2023	
Підготуватися до білета №17	1	19.05.2023	

Рисунок 3.13 – Заповнена таблиця План



Номер	Початок	Завершення	Зроблено за спринт	Додано за спринт	Усього зроблено	Усього додано	Залишилось	Верх	Низ
1	11.05.2023	18.05.2023	5	2	5	2	60	54	-2
2	18.05.2023	25.05.2023	8	2	13	4	52	46	-4
3	25.05.2023	01.06.2023	6	1	19	5	46	40	-5
4	01.06.2023	08.06.2023	11	1	30	6	35	29	-6
5	08.06.2023	15.06.2023	2	0	32	6	33	27	-6
6	15.06.2023	22.06.2023	0	0			0	0	0
7	22.06.2023	29.06.2023	0	0			0	0	0
8	29.06.2023	06.07.2023	0	0			0	0	0
9	06.07.2023	13.07.2023	0	0			0	0	0
10	13.07.2023	20.07.2023	0	0			0	0	0
11	20.07.2023	27.07.2023	0	0			0	0	0
12	27.07.2023	03.08.2023	0	0			0	0	0
13	03.08.2023	10.08.2023	0	0			0	0	0
14	10.08.2023	17.08.2023	0	0			0	0	0

Рисунок 3.14 – Заповнена таблиця Формули

На рисунках 3.13. та 3.14. зображені заповнені таблиці План та Формули. У таблиці План записана інформація про завдання, оцінку його складності, дату створення та дату завершення. Цей повний обсяг даних буде використовуватися для розрахунків у таблиці Формули. Вкладка Формули також заповнена, вказані змінні початкової дати у форматі «дд.мм.рррр», довжина спринту цілого числа більше нуля та кількість спринтів також у форматі цілого числа більше нуля.

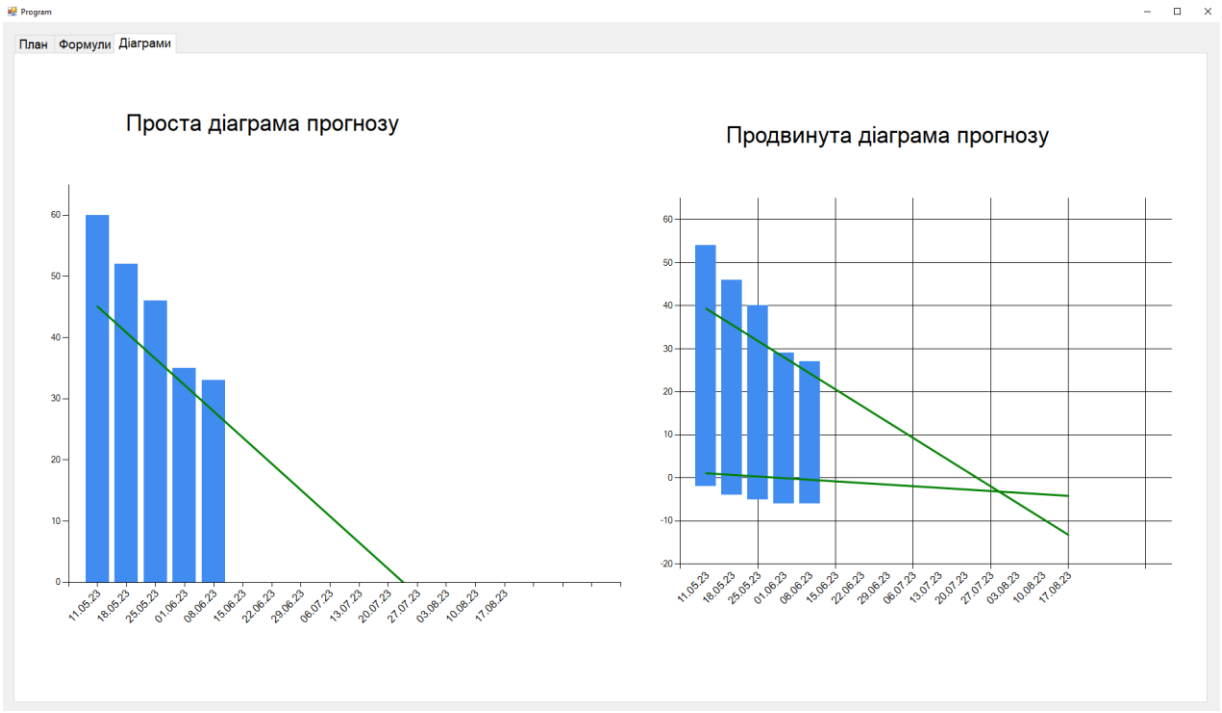


Рисунок 3.15 – Побудовані графіки

На рисунку 3.15 показаний вигляд побудованих графіків Простої діаграми прогнозу та Продвинутої діаграми прогнозу.

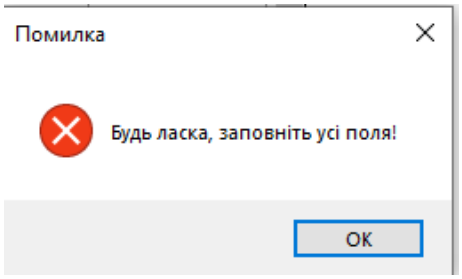


Рисунок 3.16 – Помилка незаповнених полів

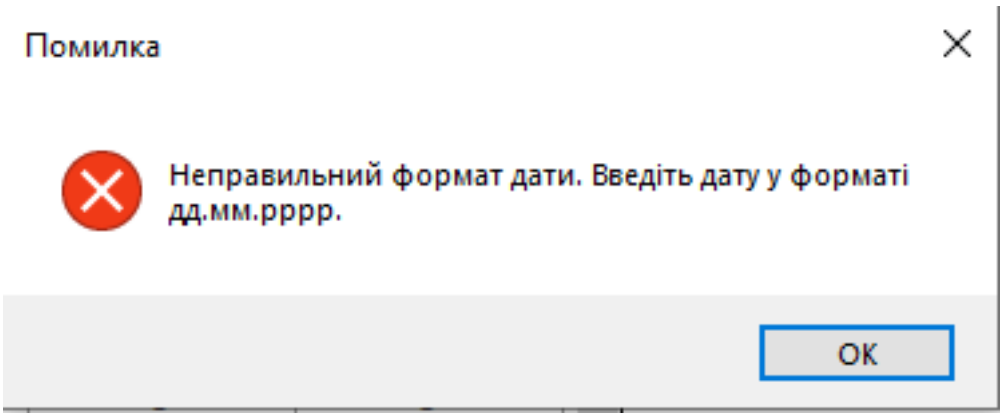


Рисунок 3.17 – Помилка невірного формату

Рисунки 3.16 та 3.17 відображають повідомлення на екрані користувача про помилки під час введення некоректних даних, як у випадку з полем початкової дати, або під час спроби зробити розрахунки коли незаповнене якесь поле. Це допомагає програмі працювати коректно та є підказкою користувачу.

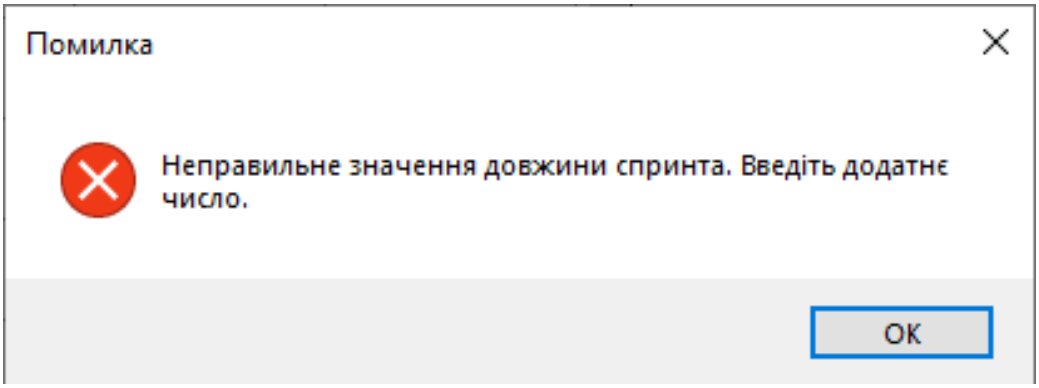


Рисунок 3.18 – Помилка невірного значення довжини

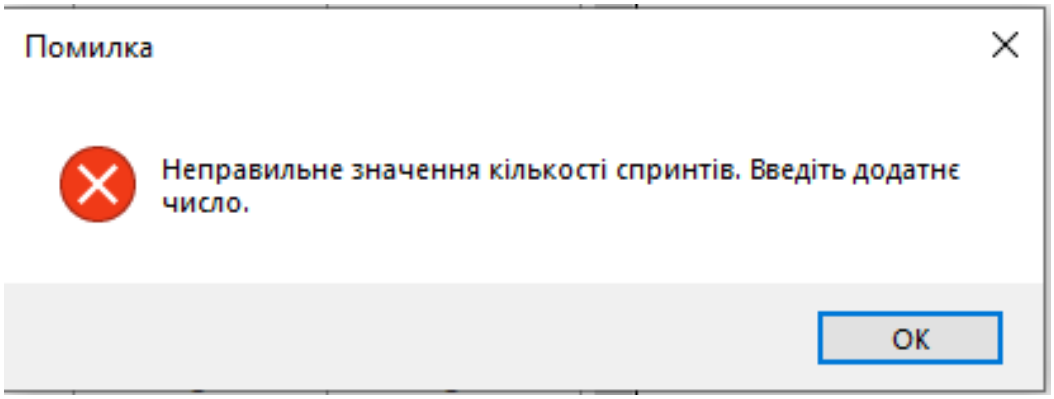


Рисунок 3.19 – Помилка невірного значення кількості

Рисунки 3.18. та 3.19. показують віка сповіщення про невірне введення даних у полях Довжина спринту та Кількість спринтів.

3.3 Сценарії роботи користувача

Коли користувач відкриває програму, йому пропонується заповнити таблицю План для подальшого аналізу проєкта. Нехай, наш користувач це студент, далі описаний його порядок дій.

1) Потрібно написати чіткі завдання.

Це важливий етап, тому що правильно поставлене завдання – це вже половина його виконання. Користувач хоче розв’язати 20 екзаменаційних білетів. Замість запису «Білет 1» краще написати «Підготовка до білета 1». На перший огляд, різниці немає, але якщо задача буде складніше, а самих завдань більше, то можна буде розгубитися. Зараз чітко вказано, що поточна задача – саме підготовка.

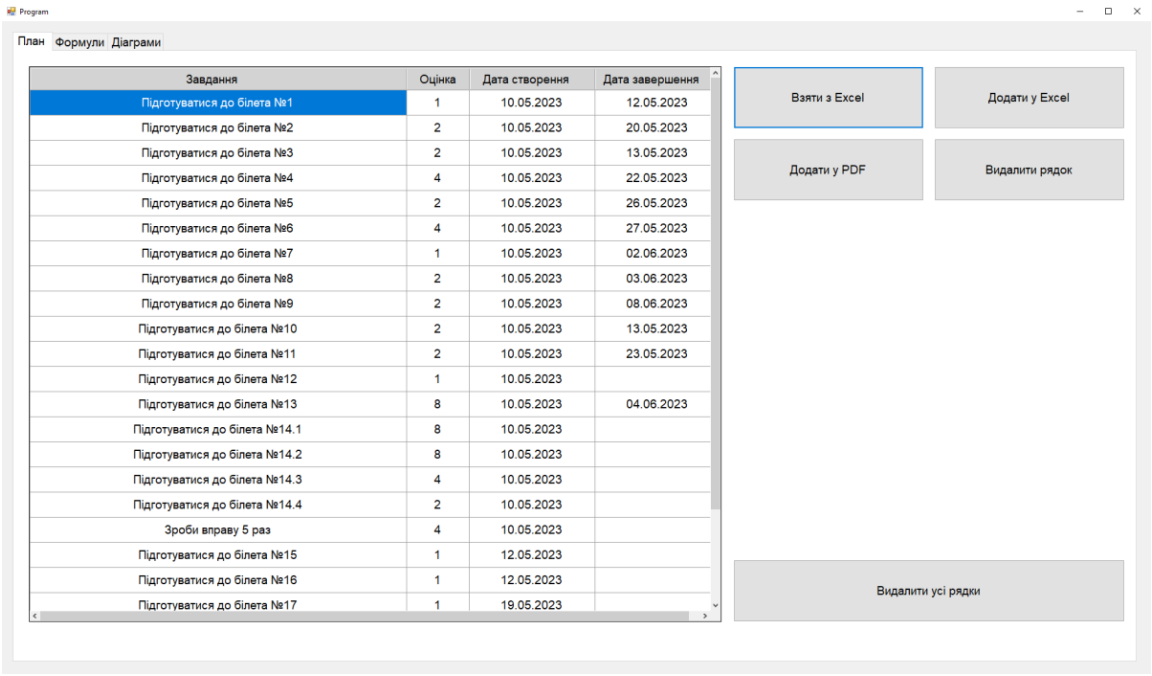
2) Правила оцінки.

Далі починається огляд перших білетів, щоб зрозуміти наскільки студент підготовлений. Якщо раптом студент натрапляє на складе завдання, то краще подумати про майбутнє та записати собі тренування щодо цієї задачі.

Під час оцінювання складності треба чесно та прозоро розставити поінти. Якщо якийсь білет здався складним, тоді краще розбити його на декілька частин. Також слід мати на увазі, що велика амплітуда оцінок не є добре, і це може бути натяком на розподілення завдань.

Після того, як користувач записав задачі, оцінив їх, далі заповнюються дати, коли задача була додана та коли завершена. Під час роботи над проєктом усі завдання поступово будуть виконуватися, так само будуть заповнятися дати завершення.

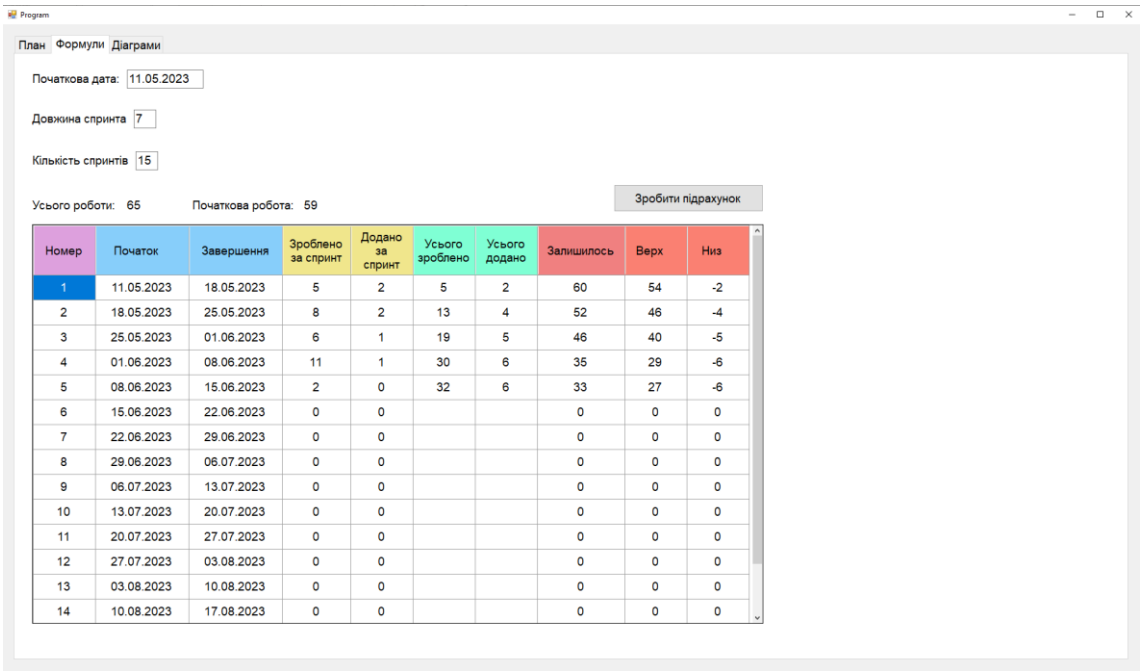
Заповнена таблиця План зображена на рис 3.20.



Завдання	Оцінка	Дата створення	Дата завершення
Підготуватися до білета №1	1	10.05.2023	12.05.2023
Підготуватися до білета №2	2	10.05.2023	20.05.2023
Підготуватися до білета №3	2	10.05.2023	13.05.2023
Підготуватися до білета №4	4	10.05.2023	22.05.2023
Підготуватися до білета №5	2	10.05.2023	26.05.2023
Підготуватися до білета №6	4	10.05.2023	27.05.2023
Підготуватися до білета №7	1	10.05.2023	02.06.2023
Підготуватися до білета №8	2	10.05.2023	03.06.2023
Підготуватися до білета №9	2	10.05.2023	08.06.2023
Підготуватися до білета №10	2	10.05.2023	13.05.2023
Підготуватися до білета №11	2	10.05.2023	23.05.2023
Підготуватися до білета №12	1	10.05.2023	
Підготуватися до білета №13	8	10.05.2023	04.06.2023
Підготуватися до білета №14.1	8	10.05.2023	
Підготуватися до білета №14.2	8	10.05.2023	
Підготуватися до білета №14.3	4	10.05.2023	
Підготуватися до білета №14.4	2	10.05.2023	
Зроби вправу 5 раз	4	10.05.2023	
Підготуватися до білета №15	1	12.05.2023	
Підготуватися до білета №16	1	12.05.2023	
Підготуватися до білета №17	1	19.05.2023	

Рисунок 3.20 – Готова вкладка План

Далі, у вкладці Формули потрібно вписати початкові параметри для створення таблиці етапів, які називаються спринти. Вводиться дата початку, довжину спринту та їх кількість. Нехай, етапи розпочнуться наступного дня, довжина етапу буде тиждень, і їх 15 штук. Натискається кнопка «Зробити підрахунок», йде перевірка на правильне заповнення полів і показується заповнена таблиця Формули (рисунок 3.21).



Номер	Початок	Завершення	Зроблено за спринт	Додано за спринт	Усього зроблено	Усього додано	Залишилось	Верх	Низ
1	11.05.2023	18.05.2023	5	2	5	2	60	54	-2
2	18.05.2023	25.05.2023	8	2	13	4	52	46	-4
3	25.05.2023	01.06.2023	6	1	19	5	46	40	-5
4	01.06.2023	08.06.2023	11	1	30	6	35	29	-6
5	08.06.2023	15.06.2023	2	0	32	6	33	27	-6
6	15.06.2023	22.06.2023	0	0			0	0	0
7	22.06.2023	29.06.2023	0	0			0	0	0
8	29.06.2023	06.07.2023	0	0			0	0	0
9	06.07.2023	13.07.2023	0	0			0	0	0
10	13.07.2023	20.07.2023	0	0			0	0	0
11	20.07.2023	27.07.2023	0	0			0	0	0
12	27.07.2023	03.08.2023	0	0			0	0	0
13	03.08.2023	10.08.2023	0	0			0	0	0
14	10.08.2023	17.08.2023	0	0			0	0	0

Рисунок 3.21 – Готова вкладка Формули

Усього роботи – загальна кількість роботи на даний момент.

Початкова робота: загальна кількість роботи до першого спринту.

Опис полів:

1) Номер. Звичайний номер спринту для нумерації.

2) Початок. Тут вписуються начальні дати спринтів. За формулою «попередній + довжина», у нас довжина 7.

2) Завершення. У цьому стовпці пишуться дати завершення спринту, також за формулою «+довжина», але різниця полягає у тому, що дата завершення строга, тобто по факту завершення першого спринту буде 17.05 у 23:59.

3) Зроблено за спринт. У цю колонку записується сума оцінок тих задач, які були виконані у проміжок спринту.

4) Додано за спринт. У цю колонку записується сума оцінок тих задач, які були додані у проміжок спринту.

5) Усього зроблено. В ці клітинки вписується сума усієї виконаної роботи за всі спринти.

6) Усього додано. В ці клітинки вписується сума усієї доданої роботи за всі спринти.

7) Залишилось. Даними для цього стовпця є інформація, який обсяг роботи залишилося зробити.

8) Верх. Початкова робота – зроблена за спринт, тобто 59 – 5.

9) Низ. Від’ємне значення доданої, тобто -2.

Далі користувач йде до вкладки Діаграми, вона зображена на рисунку 3.22.

Можна побачити два графіка. У обох з них є підписи на осі X, які дозволяють побачити приблизну дату закінчення усіх робіт.

1) Проста діаграма прогнозу.

Ця діаграма будується з масиву даних роботи, яка залишилась, а потім будується лінія тренду. Згідно з цим графіком, усі завдання будуть виконані приблизно 25.07.23.

Недоліком цього прогнозу є те, що він не враховує можливість додання нової роботи.

2) Продвинута діаграма прогнозу.

У цій діаграмі нижньою точкою стовпця є значення з колонки «Низ», а верхньою – відповідно «Верх», і будуються дві лінії тренду. Це допомагає враховувати, що під час проєкту буде додаватися робота, завдяки цьому другий графік більш точний для складних та довгих проєктів. Його прогноз – закінчення приблизно 29.07.

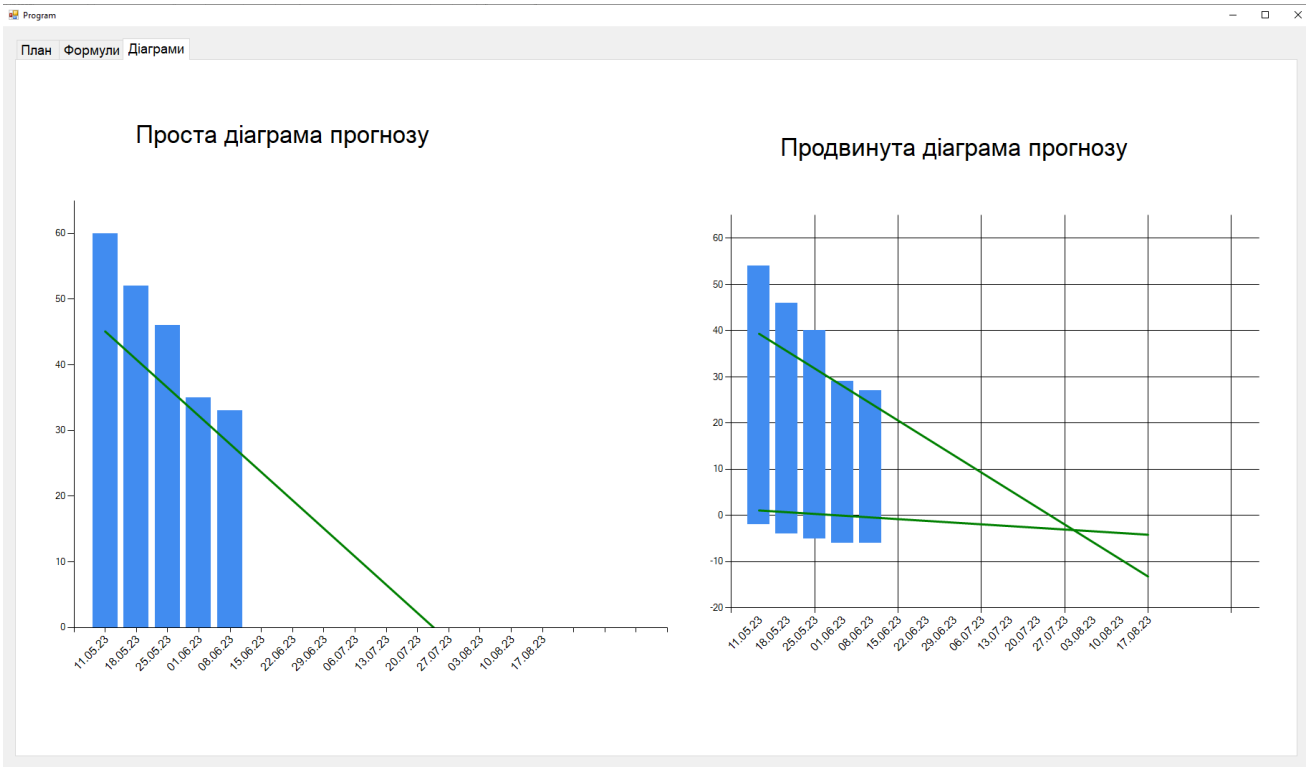
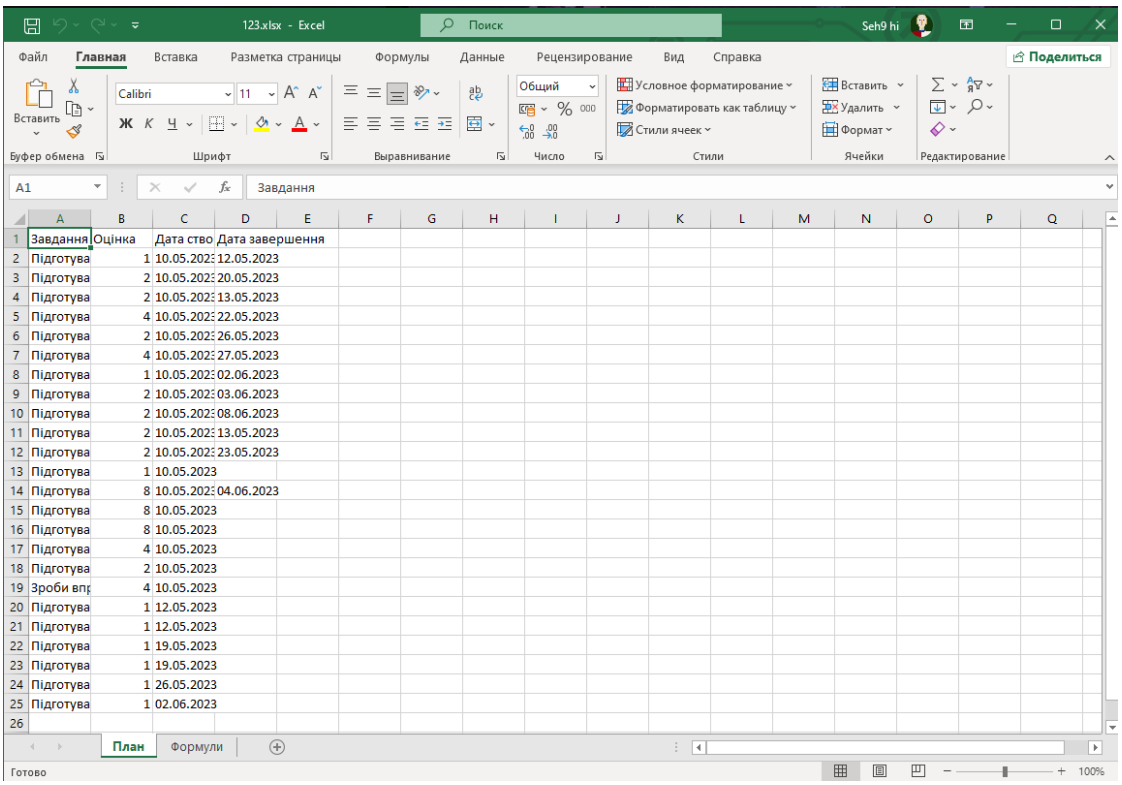


Рисунок 3.22 – Готова вкладка Діаграми

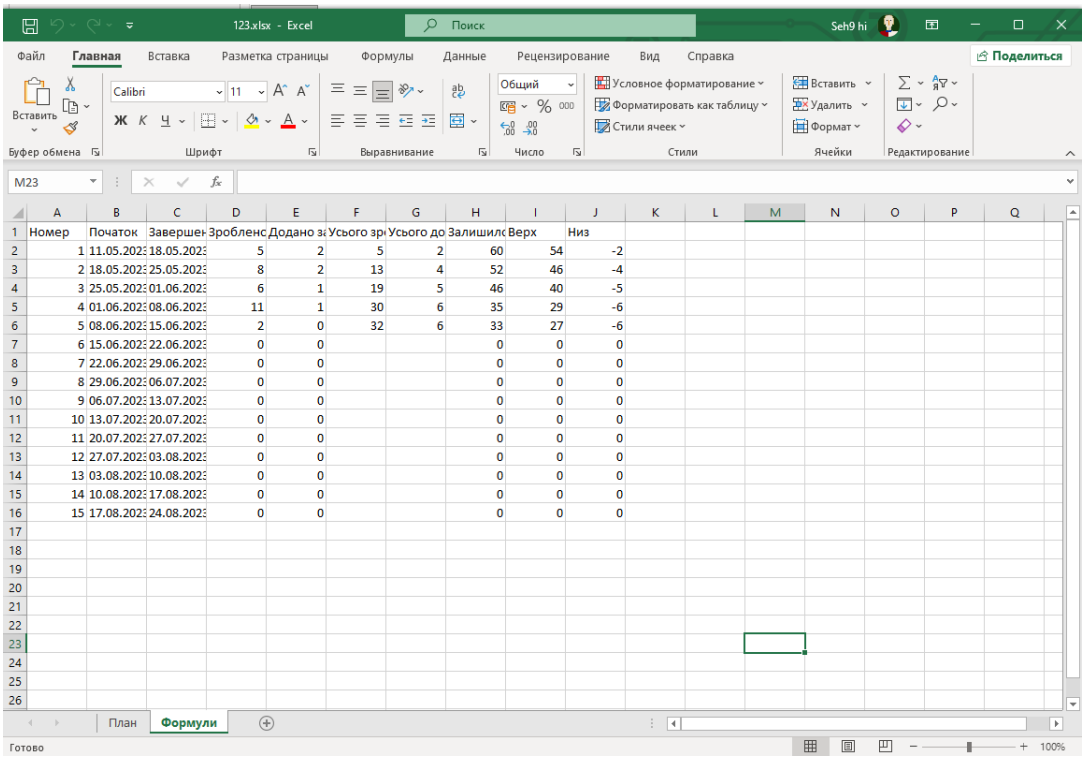
Далі користувач може експортувати дані перших двох вкладок для особистого користування.

На рисунках 3.23. та 3.24. показаний Excel файл з двома листами, на яких відповідно розміщені таблиці План та Формули.



Завдання	Оцінка	Дата ство	Дата завершення
Підготува	1	10.05.2023	12.05.2023
Підготува	2	10.05.2023	20.05.2023
Підготува	2	10.05.2023	13.05.2023
Підготува	4	10.05.2023	22.05.2023
Підготува	2	10.05.2023	26.05.2023
Підготува	4	10.05.2023	27.05.2023
Підготува	1	10.05.2023	02.06.2023
Підготува	2	10.05.2023	03.06.2023
Підготува	2	10.05.2023	08.06.2023
Підготува	2	10.05.2023	13.05.2023
Підготува	2	10.05.2023	23.05.2023
Підготува	1	10.05.2023	
Підготува	8	10.05.2023	04.06.2023
Підготува	8	10.05.2023	
Підготува	8	10.05.2023	
Підготува	4	10.05.2023	
Підготува	2	10.05.2023	
Зроби вп	4	10.05.2023	
Підготува	1	12.05.2023	
Підготува	1	12.05.2023	
Підготува	1	19.05.2023	
Підготува	1	19.05.2023	
Підготува	1	26.05.2023	
Підготува	1	02.06.2023	

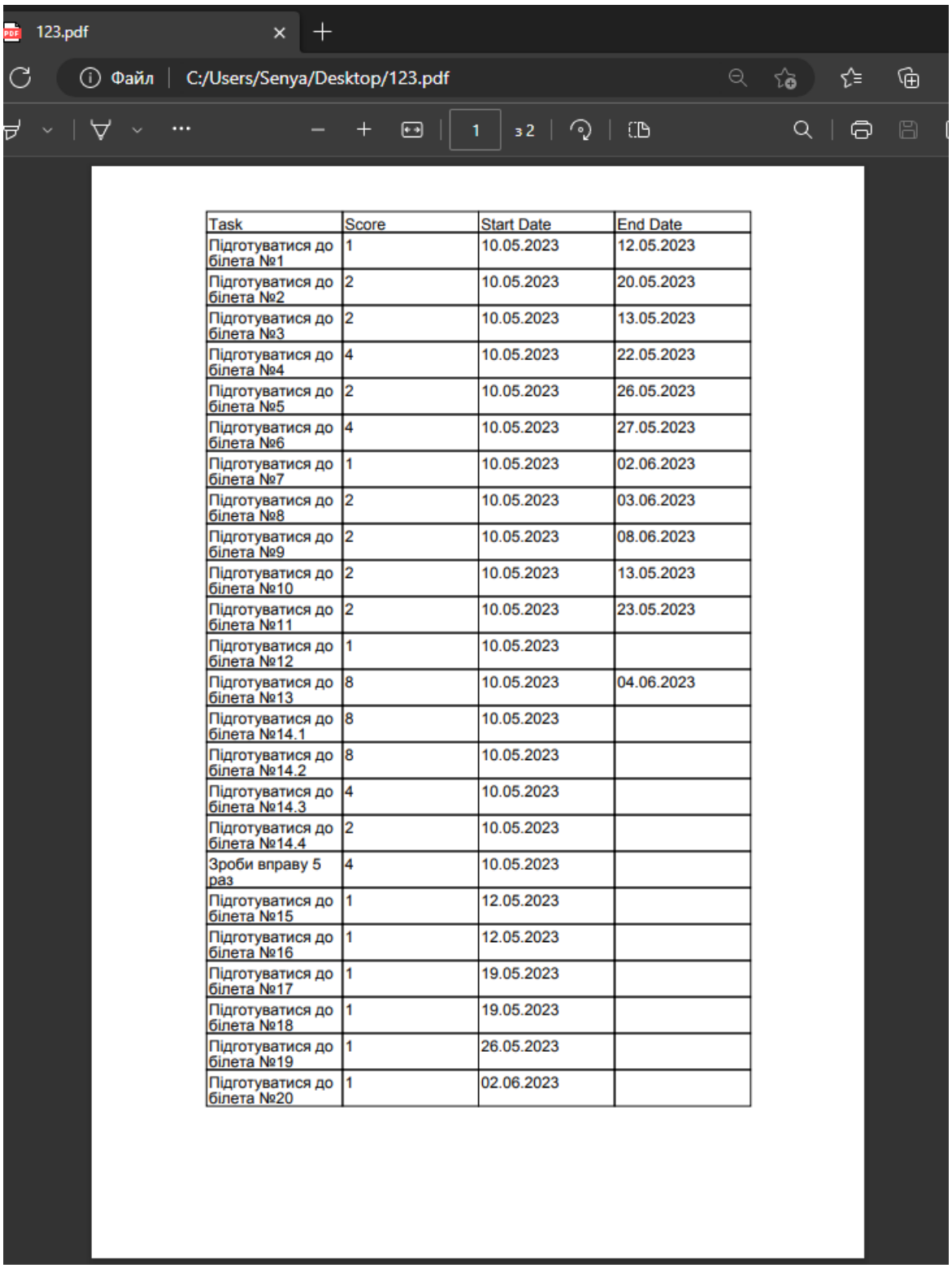
Рисунок 3.23 – Експортовані дані ч.1



Номер	Початок	Завершен	Зроблено	Додано за	Усього зр	Усього до	Залишило	Верх	Низ
1	11.05.2023	18.05.2023	5	2	5	2	60	54	-2
2	18.05.2023	25.05.2023	8	2	13	4	52	46	-4
3	25.05.2023	01.06.2023	6	1	19	5	46	40	-5
4	01.06.2023	08.06.2023	11	1	30	6	35	29	-6
5	08.06.2023	15.06.2023	2	0	32	6	33	27	-6
6	15.06.2023	22.06.2023	0	0			0	0	0
7	22.06.2023	29.06.2023	0	0			0	0	0
8	29.06.2023	06.07.2023	0	0			0	0	0
9	06.07.2023	13.07.2023	0	0			0	0	0
10	13.07.2023	20.07.2023	0	0			0	0	0
11	20.07.2023	27.07.2023	0	0			0	0	0
12	27.07.2023	03.08.2023	0	0			0	0	0
13	03.08.2023	10.08.2023	0	0			0	0	0
14	10.08.2023	17.08.2023	0	0			0	0	0
15	17.08.2023	24.08.2023	0	0			0	0	0

Рисунок 3.24 – Експортовані дані ч.2

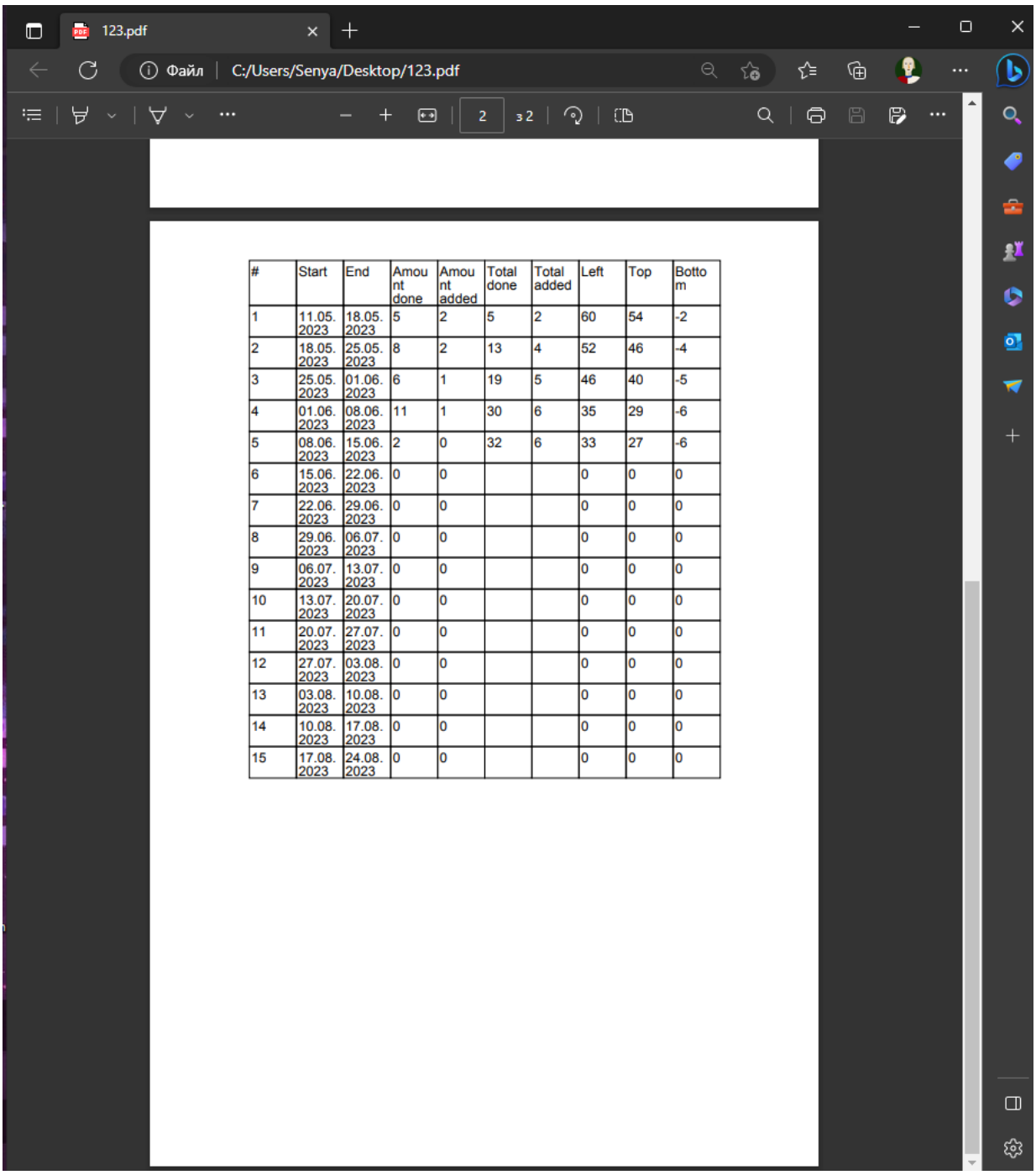
Вигляд експортованої таблиці План у файл PDF зображений на рисунку 3.25.



Task	Score	Start Date	End Date
Підготуватися до білета №1	1	10.05.2023	12.05.2023
Підготуватися до білета №2	2	10.05.2023	20.05.2023
Підготуватися до білета №3	2	10.05.2023	13.05.2023
Підготуватися до білета №4	4	10.05.2023	22.05.2023
Підготуватися до білета №5	2	10.05.2023	26.05.2023
Підготуватися до білета №6	4	10.05.2023	27.05.2023
Підготуватися до білета №7	1	10.05.2023	02.06.2023
Підготуватися до білета №8	2	10.05.2023	03.06.2023
Підготуватися до білета №9	2	10.05.2023	08.06.2023
Підготуватися до білета №10	2	10.05.2023	13.05.2023
Підготуватися до білета №11	2	10.05.2023	23.05.2023
Підготуватися до білета №12	1	10.05.2023	
Підготуватися до білета №13	8	10.05.2023	04.06.2023
Підготуватися до білета №14.1	8	10.05.2023	
Підготуватися до білета №14.2	8	10.05.2023	
Підготуватися до білета №14.3	4	10.05.2023	
Підготуватися до білета №14.4	2	10.05.2023	
Зроби вправу 5 раз	4	10.05.2023	
Підготуватися до білета №15	1	12.05.2023	
Підготуватися до білета №16	1	12.05.2023	
Підготуватися до білета №17	1	19.05.2023	
Підготуватися до білета №18	1	19.05.2023	
Підготуватися до білета №19	1	26.05.2023	
Підготуватися до білета №20	1	02.06.2023	

Рисунок 3.25 – Експортовані дані ч.3

Перенесені усі дані з додатку (рисунок 3.25): стовпці завдання, оцінки, дати створення та дати кінця. Це перша з двох сторінок, яка містить інформацію з вкладки План. Друга таблиця буде надрукована на наступній сторінці.



#	Start	End	Amount done	Amount added	Total done	Total added	Left	Top	Bottom
1	11.05.2023	18.05.2023	5	2	5	2	60	54	-2
2	18.05.2023	25.05.2023	8	2	13	4	52	46	-4
3	25.05.2023	01.06.2023	6	1	19	5	46	40	-5
4	01.06.2023	08.06.2023	11	1	30	6	35	29	-6
5	08.06.2023	15.06.2023	2	0	32	6	33	27	-6
6	15.06.2023	22.06.2023	0	0			0	0	0
7	22.06.2023	29.06.2023	0	0			0	0	0
8	29.06.2023	06.07.2023	0	0			0	0	0
9	06.07.2023	13.07.2023	0	0			0	0	0
10	13.07.2023	20.07.2023	0	0			0	0	0
11	20.07.2023	27.07.2023	0	0			0	0	0
12	27.07.2023	03.08.2023	0	0			0	0	0
13	03.08.2023	10.08.2023	0	0			0	0	0
14	10.08.2023	17.08.2023	0	0			0	0	0
15	17.08.2023	24.08.2023	0	0			0	0	0

Рисунок 3.26 – Експортовані дані ч.4

На рисунку 3.26 зображена таблиця Формули на другій сторінці експортованого файлу PDF. Збережено усю інформацію про спринти, а саме: номер спринту, дату початку, дату кінця, кількість зробленої роботи, кількість доданої роботи, скільки роботи зроблено та додано усього, скільки залишилось зробити та дані про верх та низ, які знадобляться для будування другої діаграми.

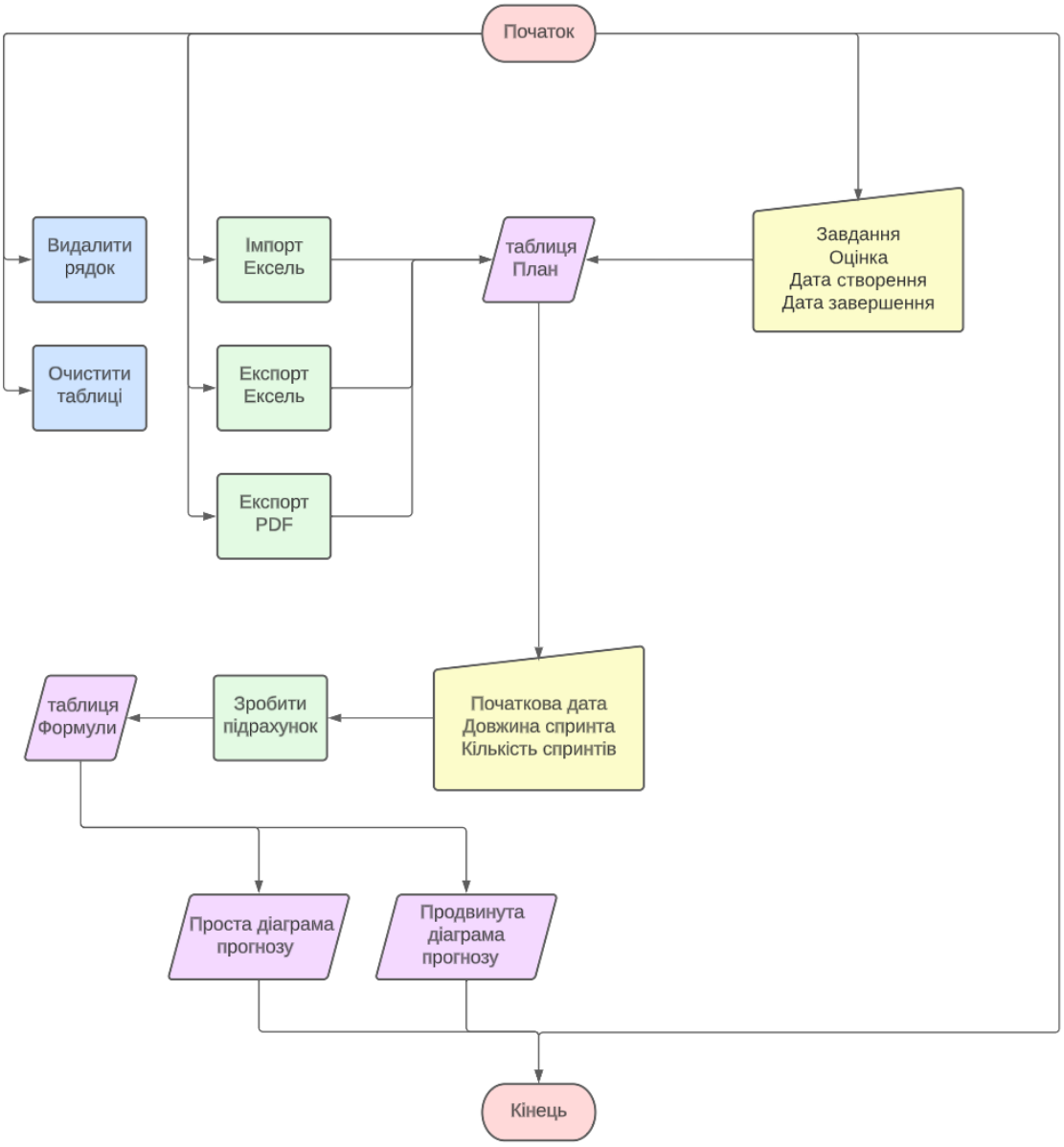


Рисунок 3.27 – Сценарій дій користувача

На рисунку 3.27 зображено карту дій користувача. При використанні програми, користувач може вибрати дію з декількох груп. Наприклад, видалити один рядок з таблиці План, або повністю очистити таблиці, ці події відзначені блакитним кольором. Процеси та взаємодія із сторонніми форматами розфарбовані зеленим кольором: користувачу дається можливість введення даних через Excel, або виведення даних через Excel та PDF. Ручне введення має жовтий колір, за допомогою нього можна ввести початкові дані, а також необхідні дані про дату старту, кількість і довжину спринтів.

Фіолетовим кольором позначені елементи, які містять у собі інформацію – це дві таблиці та дві діаграми.

3.4 Висновки до третього розділу

У третьому розділі була описана архітектура програми, наведені зображення інтерфейсу програми (пуста таблиця План, заповнена таблиця план, пуста таблиця Формули, заповнена таблиця Формули, пусті та намальовані області графіків), також була представлена інструкція користувача, яка дає детальний огляд на можливі дії, описує порядок вчинків для коректної та корисної роботи. Програма буде складатися з трьох вкладок, кожна з яких має свої функції та структуру взаємодії з іншими вкладками. На блок-схемі (рисунок 3.27) можна побачити інформацію про устрій додатку та шаблон дій, зв яким може діяти користувач.

Показані екрани виведення повідомлення про помилки, які допоможуть юзеру правильно та доцільно скористатися додатком. Передбачені ситуації, коли користувач забув ввести інформацію в одне з полів, або намагається вводити дані, формат яких непередбачений у програмі, наприклад введення строкових даних у вікно дати чи введення від’ємного числа у поле для вводу кількості або тривалості спринту.

Описані принципи, за якими будувався інтерфейс програми, а саме: простота, мінімальне навантаження ока, кольорове розділення стовпців у таблиці Формули, ненавантаженість кнопками та вікнами.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи була розроблена програма для контролю виконання завдань, спрямована на особисте або корпоративне використання. Головною метою проєкту було створення зручного, зрозумілого і простого додатку, що допомагає користувачам вести облік своїх проєктів та задач.

Під час виконання кваліфікаційної роботи була проаналізована проблема щодо розробки ПЗ для оцінювання та прогнозу проєктів. Визначена актуальність галузі через загальне зростання кількості проєктів та популяризацію розробки. Описані причини необхідності наявності плану та мапи етапів.

Було сформульоване завдання роботи, описаний перелік задач, які повинна виконувати програма. Обґрунтований вибір функцій та елементів додатку. Вказані необхідні можливості програми та їх взаємозв’язок.

Сформульовано технічні вимоги до додатку, їх профіль та пояснення щодо вибору. Вказані системні потреби, щоб чітко розуміти, на яку технічну базу користувачів можна орієнтуватися.

Проведений аналіз оцінювання багатоетапної роботи. Зазначені психологічні аспекти людини, що перешкоджають аналізуванню проєкта та менеджерські засоби, які дозволять корисно описувати план та його завдання.

Описано математичні засоби, за допомогою який буде проходити оцінка стану проєкту та прогноз щодо його закінчення. Вказано традиційні методи та обґрунтований вибір стосовно додатку, що розробляється.

Проведено аналіз технологій розробки програм на сьогоднішній час. Розглянуто засоби та інструменти, за допомогою яких створюється програма. Визначено їх переваги та недоліки і поєднання з іншими платформами.

Розроблено архітектуру програмної системи: спроєктовані класи, функції та інтерфейс програми. Намальовані блок-схеми для кращого

пояснення роботи програми, зазначені важливі моменти та описані випадки різної поведінки користувача.

Спроєктований та реалізований інтерфейс програми згідно з сучасними стандартами та моделями. Акцент був зроблений на простоті, доступності та розумінні потреб користувача. Була спроба зробити мінімалістичний стиль для зручності та приємного користування.

Проведено тестування програми для різних випадків користування, передбачені підказки для користувача. Описані сценарії поведінки, спроєктована мапа подій для наочного зображення можливостей при користуванні програмою. Архітектура програми спроектована так, щоб у користувача було менше варіантів помилитися і шлях використання програми був інтуїтивно зрозумілий.

Програма була розроблена з використанням мови програмування C# і середовища розробки Visual Studio. Для створення графічного інтерфейсу було використано API Windows Forms, що дозволило забезпечити зручну взаємодію користувача з програмою. Це створило основу для ефективного та приємного користувацького досвіду. Основними елементами програми стали таблиці та діаграми.

В процесі розробки програми було досягнуто поставлені цілі та вирішено ряд завдань, що допомагають контролювати процес виконання проєктів. Користувач може зберігати задачі, оцінювати їх складність, встановлювати дати початку та закінчення окремих задач або етапів, а також контролювати прогрес виконання. Це дозволяє користувачам бути більш організованими та ефективними у своїй роботі.

Однією з ключових переваг програми є візуальне відображення даних на графіках. Це дозволяє користувачам швидко оцінити стан проєкту та прогнозувати дату його завершення. Графічне представлення даних надає зручну візуалізацію інформації, що допомагає користувачам приймати обґрунтовані рішення та планувати свої дії. В програмі зроблено дві діаграми. На першій є графік та лінія тренду, друга має графік та дві лінії тренду.

Графіки будуються з масива даних, у першому видку з кількості роботи, що залишилась, у другому з даних про кількість доданої та зробленої роботи. Лінія тренду будується за методом найменших квадратів, тому що він пропонує оптимальну лінію апроксимації, яка мінімізує суму квадратів відхилень між спостережуваними значеннями і передбаченими значеннями. Це означає, що метод шукає найкращий підхід до даних, що дозволяє отримати більш точні та надійні результати.

Гнучкість вхідних даних є ще однією з переваг програми. Користувачі можуть налаштовувати програму під свої потреби та вимоги. Вони можуть встановлювати пріоритети завдань шляхом оцінювання їх складності, додавати завдання, визначати дедлайни та початкові строки. Це робить програму універсальним інструментом для керування завданнями, який може бути придатним для різних типів робіт та проєктів.

Програма також надає можливість експорту та імпорту даних з Excel. Це спрощує обмін інформацією та співпрацю з іншими користувачами. Користувачі можуть експортувати дані для подальшого аналізу та обробки в інших програмах, а також імпортувати дані з інших джерел. Такий обмін даними забезпечує високу гнучкість та взаємодію з іншими інструментами та системами.

Крім того, програма має можливість експорту даних до PDF-файлу для зручного зберігання та надсилання звітів. Користувачі можуть створювати докладні звіти про прогрес виконання завдань та поділитися ними з колегами, клієнтами або керівництвом. Це сприяє покращенню комунікації та співпраці в команді чи залишити у себе для майбутніх аналізів, а також забезпечує прозорість і контроль над виконанням завдань.

Загалом, розроблена програма відповідає поставленим завданням. Вона забезпечує користувачам зручний і ефективний інструмент для контролю виконання завдань і проєктів. При використанні програми можна отримати ретельний огляд стану проєктів, оцінити їх результативність та вчасно реагувати на можливі ризики та проблеми.

Розробка даної програми відкрила шлях до подальшого вдосконалення та розширення функціональності. Запровадження додаткових можливостей, таких як автоматичне сповіщення про терміни виконання, спільна робота над проєктами в режимі реального часу та інші, можуть покращити користувацький досвід та розширити сферу застосування програми.

В першу чергу, пріоритетними завданнями для вдосконалення програми є:

– розробка більш глибокого способу аналізу даних для ймовірного прогнозу. Це дозволить ще точніше оцінювати строки завершення проєкту та дасть користувачам більше впевненості у прогнозі.

– додавання можливості експорту графіків у Excel та PDF. Таке покращення дозволить експортувати не тільки текстову інформацію, а ще й графічну, що допоможе у комунікації між колегами.

– впровадження можливості завантаження програми до хмарного сховища та створення сторінок для кожного проєкта. Хмарний сервіс є більш гнучким через можливість працювати з ним у будь-якому місці, а створення окремих сторінок для кожного проєкту надасть можливість користувачу проєктувати більш об’ємні та складні за структурою завдання.

У цілому, розробка даної програми з використанням графічних елементів та візуального відображення даних є актуальним та перспективним напрямом у галузі менеджменту та контролю виконання завдань. Вона допомагає користувачам бути більш організованими, ефективними та успішними у досягненні своїх цілей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лінії тренду. URL: https://www.instaforex.com/support/ua/article/127.
2. Столяренко О. Б. Психологія особистості. 2005. 320 с.
3. Як правильно ставити задачі. URL: https://worksection.com/ua /blog/kak-stavyt-zadachy.html.
4. Mathematical model assessment and forecasting reliability software URL: https://jrnl.nau.edu.ua/index.php/ZI/article/view/10594.
5. Методи та моделі економічного прогнозування URL: https://library.if.ua/book/72/5247.html.
6. Мотивація і оцінка персоналу URL: https://library.if.ua/book/ 144/9603.html.
7. Visual Studio 2022 URL: https://itpro.ua/product/visual-studio-2022/?tab=requirements.
8. Мармоза А. Економічна статистика. 2010. 423 с.
9. Jamie Chan, C#: Learn C# in One Day and Learn It Well. URL: https://itpro.ua/product/visual-studio-2021/?tab=requirements.
10. Visual Studio documentation. URL: https://learn.microsoft.com/en-us/docs.
11. Шилдт Г. Книга C# 4.0: повний посібник. 2017. 563 с.
12. Windows Forms documentation URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-7.0.
13. Basic WinForms & Web Forms Tutorial URL: https://docs.devexpress.com/eXpressAppFramework/113496/getting-started/basic-tutorial-winforms-webforms?v=21.2.
14. Microsoft.Office.Interop.Excel Namespace. URL: https://learn.microsoft. com/en-us/dotnet/api/microsoft.office.interop.excel?view=excel-pia.
15. Джепікс Ф., Троєлсен Е. Мова програмування C# 7.0 та платформи .NET та .NET Core. 2017. 365 с.

16. Itextsharp In C# Itextsharp documentation. URL: https://www.c-sharpcorner.com/article/itextsharp-in-C-Sharp/.

17. Itextsharp documentation. URL: https://itextpdf.com/products/itextsharp.

18. 10 принципів створення хороших інтерфейсів. URL: https://www.gen.tech/post/desyat-principiv-stvorennya-horoshih-interfeisiv.

19. Мартін Р. Чистий Код. 2021. 765 с.

20. Visual Studio 2022 Product Family System Requirements. URL: https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements.

21. Create a Windows Forms app in Visual Studio with C#. URL: https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022.

22. Tutorial: Working with Windows Forms #. URL: https://www.c-sharpcorner.com/article/tutorial-working-with-windows-forms-part-i/.

23. Introduction to C# Windows Forms Applications. URL: https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/.

24. C# Tutorial. URL: https://www.w3schools.com/cs/index.php.

25. Introduction to C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/tutorials/.

26. 100 Smart Interface Design Patterns & Live Examples. URL: https://smart-interface-design-patterns.com/?gclid=CjwKCAjwsvujBhAXEiwA_UXnAHe2oMUSDmcBRTw_UXHlbarf6M-jW3LXG3C9qdIGtOBB_HRM6gzkBxoCqpUQAvD_BwE.

27. Product Management and UX – Listen and Learn. URL: https://productsthatcount.com/rent-the-runway-sr-vp-of-consumer-product-on-creating-a-seamless-user-experience/?utm_source=google&utm_medium=ads&utm_campaign=220927-PopPods&gad=1&gclid=CjwKCAjwsvujBhAXEiwA_UXnAIcAY0Ithhed5mP4iSe_kEKkaHe1fnsRXgwVpDgWocqOyXn0biVzzBoCvq4QAvD_BwE.

28. Documentation in UX Design. URL: https://aelaschool.com/en/strategy /documentation-ux-design-track-information-communicate-effortlessly/.

29. Don Norman, The Design of Everyday Things. 2016. 384 p.

30. Nir Eyal, Hooked: How to Build Habit-Forming Products. 2014. 256 p.

31. Steve Krug, Don’t Make Me Think. 2020. 216 p.

32. Susan Weinschenk, 100 Things Every Designer Needs to Know About People. 2017. 256 p.

33. Leah Buley, The User Experience Team of One: A Research and Design Survival Guide by Leah Buley. 2017. 284 p.

34. Steve Krug, Rocket Surgery Made Easy. 2020. 368 p.

35. William Lidwell, Universal Principles of Design. 2019. 272 p.

36. Carolyn Chandler, A Project Guide to UX Design: For user experience designers in the field or in the making by Russ Unger. 2018. 360 p.

37. Laura Klein, Build Better Products: A Modern Approach to Building Successful User-Centered Products. 2016. 338 p.

38. Kinneret Yifrah, Microcopy: The Complete Guide. 2019. 292 p.

39. Mohammad Rahman, C# Deconstructed – How C# Works On .NET Framework. 2020. 268 p.

40. Nathan Clark, C#: Programming Basics for Absolute Beginners. 2017. 120 p.

41. Tony Gaddis, Starting out with Visual C#. 2018. 936 p.

42. Vaskaran Sarcar, Getting Started With Advanced C#. 2018. 274 p.

43. Jon Skeet, C# in Depth: Fourth Edition. 2019. 528 p.

44. Gary Mclean, Adaptive Code via C#: Agile coding with design patterns and SOLID principles. 2021. 448 p.

45. Stephen Cleary, Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. 2018. 304 p.

ДОДАТОК

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using Application = Microsoft.Office.Interop.Excel.Application;
using Color = System.Drawing.Color;
using Excel = Microsoft.Office.Interop.Excel;
using Series = System.Windows.Forms.DataVisualization.Charting.Series;
using Workbook = Microsoft.Office.Interop.Excel.Workbook;
using Worksheet = Microsoft.Office.Interop.Excel.Worksheet;
using ChartArea = System.Windows.Forms.DataVisualization.Charting.ChartArea;
using Range = Microsoft.Office.Interop.Excel.Range;
using Axis = System.Windows.Forms.DataVisualization.Charting.Axis;
using Font = System.Drawing.Font;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
using Document = iTextSharp.text.Document;
using System.Runtime.InteropServices;

namespace Diplom
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void MainForm_Load(object sender, EventArgs e)
        {
            dgvPlan.EnableHeadersVisualStyles = false;
            dgvPlan.ColumnHeadersDefaultCellStyle.BackColor = Color.LightGray;
            dgvPlan.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None;
            dgvPlan.RowsDefaultCellStyle.WrapMode = DataGridViewTriState.True;
            dgvPlan.RowTemplate.Height = 40;

            dgvFormulas.EnableHeadersVisualStyles = false;
            dgvFormulas.ColumnHeadersDefaultCellStyle.BackColor = Color.LightGray;
            dgvFormulas.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None;
```

```
            dgvFormulas.RowsDefaultCellStyle.WrapMode = DataGridViewTriState.True;
            dgvFormulas.RowTemplate.Height = 40;

            dgvFormulas.ReadOnly = true;

            dgvFormulas.Columns[0].HeaderCell.Style.BackColor = Color.FromArgb(221,
160, 221);
            dgvFormulas.Columns[1].HeaderCell.Style.BackColor = Color.FromArgb(135,
206, 250);
            dgvFormulas.Columns[2].HeaderCell.Style.BackColor = Color.FromArgb(135,
206, 250);
            dgvFormulas.Columns[3].HeaderCell.Style.BackColor = Color.FromArgb(240,
230, 140);
            dgvFormulas.Columns[4].HeaderCell.Style.BackColor = Color.FromArgb(240,
230, 140);
            dgvFormulas.Columns[5].HeaderCell.Style.BackColor = Color.FromArgb(127,
255, 212);
            dgvFormulas.Columns[6].HeaderCell.Style.BackColor = Color.FromArgb(127,
255, 212);
            dgvFormulas.Columns[7].HeaderCell.Style.BackColor = Color.FromArgb(240,
128, 128);
            dgvFormulas.Columns[8].HeaderCell.Style.BackColor = Color.FromArgb(250,
128, 114);
            dgvFormulas.Columns[9].HeaderCell.Style.BackColor = Color.FromArgb(250,
128, 114);



            lbStart_Work.Text = "";
            lbTotal_Work.Text = "";
            Counting_of_Total_work();
            Counting_max_data();

        }

        public void Counting_of_Total_work()
        {
            int sum = 0;
            foreach (DataGridViewRow row in dgvPlan.Rows)
            {
                if (row.Cells[1].Value != null)
                {
                    int cellValue;
                    if (int.TryParse(row.Cells[1].Value.ToString(), out cellValue))
                        sum += cellValue;
```

```
                }
            }

            lbTotal_Work.Text = sum.ToString();
        }

        public int Counting_of_Start_work()
        {
            int sum = 0;
            foreach (DataGridViewRow row in dgvPlan.Rows)
            {
                if (row.Cells[1].Value != null && row.Cells[2].Value != "" &&
dgvFormulas.Rows[1].Cells[1].Value != "")
                {
                    DateTime dateAdding = Convert.ToDateTime(row.Cells[2].Value);
                    DateTime firstSprint =
Convert.ToDateTime(dgvFormulas.Rows[0].Cells[1].Value);

                    if (dateAdding < firstSprint)
                    {
                        int cellValue;
                        if (int.TryParse(row.Cells[1].Value.ToString(), out
cellValue))
                            sum += cellValue;
                    }
                }
            }

            lbStart_Work.Text = sum.ToString();
            return sum;
        }

        public void Counting_max_data()
        {
            DateTime maxDate = DateTime.MinValue;

            foreach (DataGridViewRow row in dgvPlan.Rows)
            {
                if (row.Cells[3].Value != null)
                {
                    string dateString = row.Cells[3].Value.ToString();
                    DateTime currentDate;
```

```
                    if (DateTime.TryParseExact(dateString, "dd.MM.yyyy",
CultureInfo.InvariantCulture, DateTimeStyles.None, out currentDate))
                        if (currentDate > maxDate)
                            maxDate = currentDate;
                }
            }
        }

        public void Import_Excel()
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            openFileDialog.Filter = "Files Excel|*.xlsx;*.xls";
            openFileDialog.Title = "Choose file Excel";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                dgvPlan.Rows.Clear();

                Application excelApp = new Application();

                Workbook workbook =
excelApp.Workbooks.Open(openFileDialog.FileName);

                Worksheet worksheet = workbook.Sheets[1];

                Range range = worksheet.UsedRange;

                for (int row = 2; row <= range.Rows.Count; row++)
                {
                    object cellValue1 = (range.Cells[row, 1] as Range).Value2;
                    object cellValue2 = (range.Cells[row, 2] as Range).Value2;
                    object cellValue3 = (range.Cells[row, 3] as Range).Value2;
                    object cellValue4 = (range.Cells[row, 4] as Range).Value2;


                    string value1 = cellValue1 != null ? cellValue1.ToString() : "";
                    string value2 = cellValue2 != null ? cellValue2.ToString() : "";

                    string value3 = "";
                    if (cellValue3 != null)
                        if (cellValue3.ToString() != "")
```

```
                            if (DateTime.TryParseExact(cellValue3.ToString(),
"dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out var
dateValue3))
                                value3 = dateValue3.ToString("dd.MM.yyyy");
                            else
                                value3 =
DateTime.FromOADate((double)cellValue3).ToString("dd.MM.yyyy");

                    string value4 = "";
                    if (cellValue4 != null)
                        if (cellValue4.ToString() != "")
                            if (DateTime.TryParseExact(cellValue4.ToString(),
"dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out var
dateValue4))
                                value4 = dateValue4.ToString("dd.MM.yyyy");
                            else
                                value4 =
DateTime.FromOADate((double)cellValue4).ToString("dd.MM.yyyy");

                    dgvPlan.Rows.Add(value1, value2, value3, value4);
                }

                workbook.Close();
                excelApp.Quit();

                Marshal.ReleaseComObject(range);
                Marshal.ReleaseComObject(worksheet);
                Marshal.ReleaseComObject(workbook);
                Marshal.ReleaseComObject(excelApp);
            }
        }

        private void btnExport_Excel_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Excel Files|*.xlsx";
            saveFileDialog.Title = "Save Excel File";
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                string filePath = saveFileDialog.FileName;

                Excel.Application excelApp = new Excel.Application();
                Excel.Workbook workbook = excelApp.Workbooks.Add();
```

```
                try
                {
                    Excel.Worksheet worksheet2 = workbook.Sheets.Add();
                    worksheet2.Name = tabPlan.TabPages[1].Text;
                    ExportDataGridViewToExcel(dgvFormulas, worksheet2);

                    Excel.Worksheet worksheet1 = workbook.Sheets.Add();
                    worksheet1.Name = tabPlan.TabPages[0].Text;
                    ExportDataGridViewToExcel(dgvPlan, worksheet1);

                    if (workbook.Sheets.Count >= 3)
                    {
                        Excel.Worksheet defaultSheet =
(Excel.Worksheet)workbook.Sheets[3];
                        defaultSheet.Delete();
                    }

                    workbook.SaveAs(filePath);
                }
                finally
                {
                    workbook.Close(false);
                    excelApp.Quit();
                    Marshal.ReleaseComObject(workbook);
                    Marshal.ReleaseComObject(excelApp);
                }
            }
        }

        public void ExportDataGridViewToExcel(DataGridView dataGridView,
Excel.Worksheet worksheet)
        {
            for (int j = 0; j < dataGridView.Columns.Count; j++)
                worksheet.Cells[1, j + 1] = dataGridView.Columns[j].HeaderText;

            for (int i = 0; i < dataGridView.Rows.Count; i++)
                for (int j = 0; j < dataGridView.Columns.Count; j++)
                    worksheet.Cells[i + 2, j + 1] =
dataGridView.Rows[i].Cells[j].Value;
        }

        private void btnImport_Excel_Click(object sender, EventArgs e)
        {
            Import_Excel();
```

```
            Counting_of_Total_work();

            Counting_max_data();
        }

        private void dgvPlan_CellEndEdit(object sender, DataGridViewCellEventArgs e)
        {
            Counting_of_Total_work();
        }

        public void Construct_date_score()
        {
            int rowCount = Convert.ToInt32(txtCount_of_sprints.Text);
            string startDate = txtStart_date.Text;
            int sprintLength = Convert.ToInt32(txtLenght_of_sprint.Text);

            DateTime startDateValue = DateTime.ParseExact(startDate, "dd.MM.yyyy",
CultureInfo.InvariantCulture);

            for (int i = 0; i < rowCount; i++)
            {
                DateTime currentDate = startDateValue.AddDays(i * sprintLength);

                DateTime endDate = currentDate.AddDays(sprintLength);

                dgvFormulas.Rows.Add(i + 1, currentDate.ToString("dd.MM.yyyy"),
endDate.ToString("dd.MM.yyyy"));
            }

            int rowCount1 = dgvFormulas.Rows.Count;

            for (int i = 0; i < rowCount1 - 1; i++)
            {
                DateTime dateStart =
Convert.ToDateTime(dgvFormulas.Rows[i].Cells[1].Value);

                DateTime dateEnd =
Convert.ToDateTime(dgvFormulas.Rows[i].Cells[2].Value);

                int amount_of_done = 0;
                int amount_added = 0;

                foreach (DataGridViewRow row in dgvPlan.Rows)
```

```
                {

                    int complexity_of_task = Convert.ToInt32(row.Cells[1].Value);

                    if (row.Cells[3].Value != "")
                    {
                        DateTime date1 = Convert.ToDateTime(row.Cells[3].Value);

                        if (dateStart <= date1 && date1 < dateEnd)
                            amount_of_done += complexity_of_task;
                    }

                    if (row.Cells[2].Value != "")
                    {
                        DateTime date2 = Convert.ToDateTime(row.Cells[2].Value);

                        if (dateStart <= date2 && date2 < dateEnd)
                            amount_added += complexity_of_task;
                    }
                }

                dgvFormulas.Rows[i].Cells[3].Value = amount_of_done;
                dgvFormulas.Rows[i].Cells[4].Value = amount_added;
            }

            for (int i = 0; i < rowCount1 - 1; i++)
            {
                if (Convert.ToInt32(dgvFormulas.Rows[i].Cells[3].Value) != 0)
                {
                    if (i == 0)
                        dgvFormulas.Rows[i].Cells[5].Value =
dgvFormulas.Rows[i].Cells[3].Value;
                    else
                        dgvFormulas.Rows[i].Cells[5].Value =
Convert.ToInt32(dgvFormulas.Rows[i].Cells[3].Value) +
Convert.ToInt32(dgvFormulas.Rows[i - 1].Cells[5].Value);

                    if (i == 0)
                        dgvFormulas.Rows[i].Cells[6].Value =
dgvFormulas.Rows[i].Cells[4].Value;
                    else
                        dgvFormulas.Rows[i].Cells[6].Value =
Convert.ToInt32(dgvFormulas.Rows[i].Cells[4].Value) +
Convert.ToInt32(dgvFormulas.Rows[i - 1].Cells[6].Value);
```

```
                    dgvFormulas.Rows[i].Cells[7].Value =
Convert.ToInt32(lbTotal_Work.Text) -
Convert.ToInt32(dgvFormulas.Rows[i].Cells[5].Value);
                }
                else
                    dgvFormulas.Rows[i].Cells[7].Value = 0;



                if (Convert.ToInt32(dgvFormulas.Rows[i].Cells[3].Value) != 0)
                {
                    if (i == 0)
                        dgvFormulas.Rows[i].Cells[8].Value =
Counting_of_Start_work() - Convert.ToInt32(dgvFormulas.Rows[i].Cells[3].Value);
                    else
                        dgvFormulas.Rows[i].Cells[8].Value =
Convert.ToInt32(dgvFormulas.Rows[i - 1].Cells[8].Value) -
Convert.ToInt32(dgvFormulas.Rows[i].Cells[3].Value);

                    dgvFormulas.Rows[i].Cells[9].Value = -
Convert.ToInt32(dgvFormulas.Rows[i].Cells[6].Value);
                }
                else
                {
                    dgvFormulas.Rows[i].Cells[8].Value = 0;
                    dgvFormulas.Rows[i].Cells[9].Value = 0;
                }
            }
        }

        private void btConstruct_tables_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(txtStart_date.Text) ||
string.IsNullOrEmpty(txtLenght_of_sprint.Text) ||
string.IsNullOrEmpty(txtCount_of_sprints.Text))
            {
                MessageBox.Show("Будь ласка, заповніть усі поля!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (!DateTime.TryParseExact(txtStart_date.Text, "dd.MM.yyyy",
CultureInfo.InvariantCulture, DateTimeStyles.None, out _))
```

```
            {
                MessageBox.Show("Неправильний формат дати. Введіть дату у форматі
дд.мм.рррр.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (!int.TryParse(txtLenght_of_sprint.Text, out int value2) || value2 <=
0)
            {
                MessageBox.Show("Неправильне значення довжини спринту. Введіть
додатнє число.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (!int.TryParse(txtCount_of_sprints.Text, out int value3) || value3 <=
0)
            {
                MessageBox.Show("Неправильне значення кількості спринтів. Введіть
додатнє число.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            dgvFormulas.Rows.Clear();

            Construct_date_score();

            Counting_of_Start_work();

            chartFirst_diagram.Series.Clear();

            chartSecond_diagram.Series.Clear();

            First_Chart();

            First_TrendLine();

            Second_Chart();

            SecondOne_TrendLine();

            SecondTwo_TrendLine();
        }

        public void First_Chart()
```

```
{
    ChartArea MainGraph = chartFirst_diagram.ChartAreas[0];
    Series series = new Series("Data");

    Axis xAxis = chartFirst_diagram.ChartAreas[0].AxisX;
    Axis yAxis = chartFirst_diagram.ChartAreas[0].AxisY;

    yAxis.MajorGrid.Enabled = false;
    xAxis.MajorGrid.Enabled = false;

    yAxis.Maximum = Convert.ToInt32(lbTotal_Work.Text);
    yAxis.Minimum = -20;
    yAxis.Interval = 10;



    xAxis.Interval = 0;
    xAxis.Maximum = dgvFormulas.Rows.Count + 3;
    xAxis.Interval = 1;
    xAxis.LabelStyle.Format = "dd/MM/yy";
    xAxis.LabelStyle.Font = new Font("Arial", 12f);
    xAxis.LabelStyle.Angle = -45;

    MainGraph.Position.Width = 100;
    MainGraph.Position.Height = 100;

    MainGraph.Position.X = 0;
    MainGraph.Position.Y = 0;

    foreach (DataGridViewRow row in dgvFormulas.Rows)
    {
        if (!row.IsNewRow)
        {
            DateTime xValueData = Convert.ToDateTime(row.Cells[1].Value);

            string xValue = xValueData.ToString("dd.MM");
            double yValue = Convert.ToDouble(row.Cells[7].Value);

            series.Points.AddXY(xValue, yValue);
        }
    }

    xAxis.CustomLabels.Clear();

    for (int i = 0; i < series.Points.Count; i++)
```

```
            {
                DateTime date =
Convert.ToDateTime(dgvFormulas.Rows[i].Cells[1].Value);

                CustomLabel customLabel = new CustomLabel(i + 0.5, i + 1.5,
date.ToString("dd/MM/yy"), 0, LabelMarkStyle.None);
                xAxis.CustomLabels.Add(customLabel);
            }

            chartFirst_diagram.Series.Add(series);
        }

        public void First_TrendLine()
        {
            Series trendLineSeries = new Series("TrendLine");

            trendLineSeries.ChartType = SeriesChartType.Line;
            trendLineSeries.Color = Color.Green;
            trendLineSeries.BorderWidth = 3;

            Axis xAxis = chartFirst_diagram.ChartAreas[0].AxisX;
            Axis yAxis = chartFirst_diagram.ChartAreas[0].AxisY;

            xAxis.Minimum = 0;
            xAxis.Interval = 1;
            yAxis.Maximum = Convert.ToInt32(lbTotal_Work.Text);
            yAxis.Minimum = 0;

            List<double> values = new List<double>();
            foreach (DataGridViewRow row in dgvFormulas.Rows)
                if (row.Cells[7].Value != null &&
double.TryParse(row.Cells[7].Value.ToString(), out double value))
                    values.Add(value);

            double[] xValues = new double[values.Count];

            for (int i = 0; i < xValues.Length; i++)
                xValues[i] = i + 1;

            double sumX = 0;
            double sumY = 0;
            double sumXY = 0;
            double sumXX = 0;
            int n = values.Count;
```

```
    for (int i = 0; i < n; i++)
    {
        double x = xValues[i];
        double y = values[i];

        sumX += x;
        sumY += y;
        sumXY += x * y;
        sumXX += x * x;
    }

    double slope = (n * sumXY - sumX * sumY) / (n * sumXX - sumX * sumX);
    double intercept = (sumY - slope * sumX) / n;

    double minX = xValues[0];
    double maxX = xValues[xValues.Length - 1];
    double minY = minX * slope + intercept;
    double maxY = maxX * slope + intercept;

    chartFirst_diagram.Series.Add(trendLineSeries);

    chartFirst_diagram.Series["TrendLine"].Points.AddXY(maxX, maxY);
    chartFirst_diagram.Series["TrendLine"].Points.AddXY(minX, minY);
}

public void Second_Chart()
{
    ChartArea chartArea = new ChartArea();

    Series series1 = new Series("FirstLine");
    series1.ChartType = SeriesChartType.Line;
    series1.BorderWidth = 2;

    Series series2 = new Series("SecondLine");
    series2.ChartType = SeriesChartType.Line;
    series2.BorderWidth = 2;

    foreach (DataGridViewRow row in dgvFormulas.Rows)
    {
        double value1 = Convert.ToDouble(row.Cells[8].Value);
        double value2 = Convert.ToDouble(row.Cells[9].Value);

        series1.Points.Add(value1);
```

```
                series2.Points.Add(value2);
            }

            chartSecond_diagram.Series.Add(series1);
            chartSecond_diagram.Series.Add(series2);

            Series series = chartSecond_diagram.Series.Add("Columns");

            Axis xAxis = chartSecond_diagram.ChartAreas[0].AxisX;
            Axis yAxis = chartSecond_diagram.ChartAreas[0].AxisY;

            yAxis.Maximum = Convert.ToInt32(lbTotal_Work.Text);
            yAxis.Interval = 10;

            xAxis.Maximum = dgvFormulas.Rows.Count + 3;
            xAxis.Minimum = 0;
            xAxis.Interval = 3;
            xAxis.LabelStyle.Format = "dd/MM/yy";
            xAxis.LabelStyle.Font = new Font("Arial", 12f);
            xAxis.LabelStyle.Angle = -45;



            chartArea.Position.X = 0;
            chartArea.Position.Y = 0;

            foreach (DataGridViewRow row in dgvFormulas.Rows)
            {
                double upperValue = Convert.ToDouble(row.Cells[8].Value);
                double lowerValue = Convert.ToDouble(row.Cells[9].Value);

                series.Points.AddXY(row.Index+1, upperValue);
                series.Points.AddXY(row.Index+1, lowerValue);

                if (upperValue >= lowerValue)
                    series.Points[series.Points.Count - 2].Color =
Color.FromArgb(65, 140, 240);
                else
                    series.Points[series.Points.Count - 2].Color =
Color.FromArgb(65, 140, 240);

                series.Points[series.Points.Count - 1].Color = Color.FromArgb(65,
140, 240);
            }
```

```
            xAxis.CustomLabels.Clear();

            for (int i = 0; i < series1.Points.Count - 1; i++)
            {
                DateTime date =
Convert.ToDateTime(dgvFormulas.Rows[i].Cells[1].Value);

                CustomLabel customLabel = new CustomLabel(i + 0.5, i + 1.5,
date.ToString("dd/MM/yy"), 0, LabelMarkStyle.None);
                xAxis.CustomLabels.Add(customLabel);
            }

            chartSecond_diagram.Series["FirstLine"].Enabled = false;
            chartSecond_diagram.Series["SecondLine"].Enabled = false;
        }

        public void SecondOne_TrendLine()
        {
            Series trendLineSeries = new Series("TrendLine");

            trendLineSeries.ChartType = SeriesChartType.Line;
            trendLineSeries.Color = Color.Green;
            trendLineSeries.BorderWidth = 3;

            List<double> values = new List<double>();
            foreach (DataGridViewRow row in dgvFormulas.Rows)
                if (row.Cells[8].Value != null &&
double.TryParse(row.Cells[8].Value.ToString(), out double value))
                    values.Add(value);

            double[] xValues = new double[values.Count];

            for (int i = 0; i < xValues.Length; i++)
                xValues[i] = i + 1;

            double sumX = 0;
            double sumY = 0;
            double sumXY = 0;
            double sumXX = 0;
            int n = values.Count;

            for (int i = 0; i < n; i++)
            {
                double x = xValues[i];
```

```
                double y = values[i];

                sumX += x;
                sumY += y;
                sumXY += x * y;
                sumXX += x * x;
            }

            double slope = (n * sumXY - sumX * sumY) / (n * sumXX - sumX * sumX);
            double intercept = (sumY - slope * sumX) / n;

            double minX = xValues[0];
            double maxX = xValues[xValues.Length - 1];
            double minY = minX * slope + intercept;
            double maxY = maxX * slope + intercept;

            chartSecond_diagram.Series.Add(trendLineSeries);

            chartSecond_diagram.Series["TrendLine"].Points.AddXY(maxX, maxY);
            chartSecond_diagram.Series["TrendLine"].Points.AddXY(minX, minY);
        }

        public void SecondTwo_TrendLine()
        {
            Series trendLineSeries = new Series("TrendLineTwo");

            trendLineSeries.ChartType = SeriesChartType.Line;
            trendLineSeries.Color = Color.Green;
            trendLineSeries.BorderWidth = 3;

            List<double> values = new List<double>();
            foreach (DataGridViewRow row in dgvFormulas.Rows)
                if (row.Cells[9].Value != null &&
double.TryParse(row.Cells[9].Value.ToString(), out double value))
                    values.Add(value);

            double[] xValues = new double[values.Count];

            for (int i = 0; i < xValues.Length; i++)
                xValues[i] = i + 1;

            double sumX = 0;
            double sumY = 0;
            double sumXY = 0;
```

```
    double sumXX = 0;
    int n = values.Count;

    for (int i = 0; i < n; i++)
    {
        double x = xValues[i];
        double y = values[i];

        sumX += x;
        sumY += y;
        sumXY += x * y;
        sumXX += x * x;
    }

    double slope = (n * sumXY - sumX * sumY) / (n * sumXX - sumX * sumX);
    double intercept = (sumY - slope * sumX) / n;

    double minX = xValues[0];
    double maxX = xValues[xValues.Length - 1];
    double minY = maxX * slope + intercept;
    double maxY = minX * slope + intercept;

    chartSecond_diagram.Series.Add(trendLineSeries);

    chartSecond_diagram.Series["TrendLineTwo"].Points.AddXY(maxX, maxY);
    chartSecond_diagram.Series["TrendLineTwo"].Points.AddXY(minX, minY);
}

private void btnClear_rows_Click(object sender, EventArgs e)
{
    dgvPlan.Rows.Clear();
    dgvFormulas.Rows.Clear();
}

private void btnPDF_Export_Click(object sender, EventArgs e)
{
    Document document = new Document();

    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "PDF файлы (*.pdf)|*.pdf";
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = saveFileDialog.FileName;
```

```
                using (FileStream fs = new FileStream(filePath, FileMode.Create))
                {
                    PdfWriter writer = PdfWriter.GetInstance(document, fs);

                    document.Open();

                    string fontFilePath =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Fonts", "Arialcyr.ttf");
                    BaseFont baseFont = BaseFont.CreateFont(fontFilePath,
BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
                    iTextSharp.text.Font font = new iTextSharp.text.Font(baseFont,
12, 0);

                    PdfPTable table1 = new PdfPTable(4);

                    string[] headers1 = { "Task", "Score", "Start Date", "End Date"
};

                    foreach (string header in headers1)
                    {
                        PdfPCell headerCell = new PdfPCell(new Phrase(header,
font));
                        table1.AddCell(headerCell);
                    }

                    for (int i = 0; i < dgvPlan.Rows.Count - 1; i++)
                        for (int j = 0; j < dgvPlan.Columns.Count; j++)
                        {
                            string cellValue = dgvPlan[j, i].Value != null ?
dgvPlan[j, i].Value.ToString() : "";
                            PdfPCell cell = new PdfPCell(new Phrase(cellValue,
font));
                            table1.AddCell(cell);
                        }

                    document.Add(table1);

                    document.NewPage();

                    PdfPTable table2 = new PdfPTable(10);

                    string[] headers2 = { "#", "Start", "End", "Amount done",
"Amount added", "Total done", "Total added", "Left", "Top", "Bottom" };
```

```
                    foreach (string header in headers2)
                    {
                        PdfPCell headerCell = new PdfPCell(new Phrase(header,
font));
                        table2.AddCell(headerCell);
                    }

                    for (int i = 0; i < dgvFormulas.Rows.Count - 1; i++)
                        for (int j = 0; j < dgvFormulas.Columns.Count; j++)
                        {
                            string cellValue = dgvFormulas[j, i].Value != null ?
dgvFormulas[j, i].Value.ToString() : "";
                            PdfPCell cell = new PdfPCell(new Phrase(cellValue,
font));
                            table2.AddCell(cell);
                        }

                    document.Add(table2);

                    document.Close();
                }

                System.Diagnostics.Process.Start(filePath);
            }
        }

        private void btDelete_row_Click(object sender, EventArgs e)
        {
            if (dgvPlan.SelectedCells.Count > 0)
            {
                int selectedRowIndex = dgvPlan.SelectedCells[0].RowIndex;
                if (selectedRowIndex >= 0 && selectedRowIndex < dgvPlan.Rows.Count
&& !dgvPlan.Rows[selectedRowIndex].IsNewRow)
                    dgvPlan.Rows.RemoveAt(selectedRowIndex);
            }
        }
    }
}
```