НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
інформаційних технологій (назва факультету (ННІ))	комп'ютерних наук (назва кафедри)
_____ Ігор БОЛБОТ (підпис) (ім'я ПРІЗВИЩЕ)	_____ Белла ГОЛУБ (підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.	"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Інформаційно-аналітична система визначення рівня реалізації ІТ проекту

Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма Інформаційні управляючі системи і технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

к.т.н., доцент _____ Белла ГОЛУБ
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

канд. техн. наук, доцент _____ Віталій СВАТКО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ Денис КАРПОВИЧ
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
к.т.н., доцент _____ Белла ГОЛУБ
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"_01_" листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Карповича Дениса Олеговича
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»

Освітня програма Інформаційні управляючі системи і технології

Орієнтація освітньої програми освітньо-професійна

Тема магістерської кваліфікаційної роботи Інформаційно-аналітична система визначення рівня реалізації ІТ проекту

затверджена наказом ректора НУБіП України від "01" листопада 2024 р. №1964 «С»

Термін подання завершеної роботи на кафедру 01.12.2025

Вихідні дані до магістерської кваліфікаційної роботи Матеріали з управління ІТ-проєктами, технічна документація з розроблення інформаційних систем, приклади структури баз даних і алгоритмів, середовище розробки .NET (C#, WPF, SQL Server).

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області.
2. Моделювання системи
3. Розробка системи
4. Результати дослідження

Дата видачі завдання "01" листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Віталій СВАТКО
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Денис КАРПОВИЧ
(підпис) (ім'я ПРІЗВИЩЕ)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІТ — інформаційні технології

ІАС — інформаційно-аналітична система

АРМ — автоматизоване робоче місце

БД — база даних

СУБД — система управління базами даних

DBMS — Database Management System (система управління базами даних)

SQL — Structured Query Language (мова структурованих запитів)

SQL Server — система управління базами даних Microsoft SQL Server

EF Core — Entity Framework Core (ORM-технологія для взаємодії з базою даних)

ORM — Object-Relational Mapping (об’єктно-реляційне відображення)

.NET — програмна платформа Microsoft для створення застосунків

WPF — Windows Presentation Foundation (платформа створення графічних інтерфейсів)

MVVM — Model-View-ViewModel (архітектурний шаблон проєктування у WPF)

UI — User Interface (інтерфейс користувача)

UX — User Experience (зручність користування системою)

CRUD — Create, Read, Update, Delete (основні операції над даними)

PDF — Portable Document Format (електронний формат документа)

KPI — Key Performance Indicators (ключові показники ефективності)

API — Application Programming Interface (програмний інтерфейс взаємодії)

JSON — JavaScript Object Notation (текстовий формат обміну даними)

XML — eXtensible Markup Language (розширювана мова розмітки)

ID — ідентифікатор (унікальний ключ запису в базі даних)

UI-елементи — елементи графічного інтерфейсу користувача

LiveCharts — бібліотека побудови діаграм у середовищі .NET

KPI-аналітика — метод оцінювання ефективності виконання завдань

SQL-запит — інструкція для отримання або зміни даних у базі даних

DTO — Data Transfer Object (об’єкт передавання даних між рівнями системи)

Repository — програмний шаблон доступу до даних

Dashboard — аналітична панель для відображення ключових показників проєкту

Auth — підсистема автентифікації користувачів

Role — роль користувача в системі (адміністратор, менеджер, користувач)

Status — стан або поточний статус етапу чи завдання

Task — окреме завдання в межах етапу проєкту

Stage — етап реалізації проєкту

Project — проєкт як основна одиниця управління

ВСТУП

У сучасному світі інформаційні технології є рушійною силою цифрової трансформації суспільства та бізнесу. Ефективне управління ІТ-проєктами стає важливою умовою для підвищення конкурентоспроможності компаній, оптимізації витрат і забезпечення стабільного розвитку. Разом із тим, зростання масштабів ІТ-проєктів, їх складності, а також необхідність оперативного контролю виконання завдань потребують створення спеціалізованих інформаційно-аналітичних систем (ІАС), які забезпечують моніторинг і визначення рівня реалізації таких проєктів у режимі реального часу [1], [2].

Визначення рівня реалізації ІТ-проєкту є комплексним завданням, що поєднує оцінювання прогресу робіт, ефективності командної взаємодії та досягнення цільових показників. Для цього широко застосовуються ключові показники ефективності (KPI) — кількісні метрики, які дозволяють оцінити ступінь досягнення запланованих цілей [3]. Проте в існуючих системах управління проєктами (Jira, Asana, Trello) основна увага приділяється контролю завдань, а не комплексному аналізу KPI-показників. Відсутність інтегрованої системи, що поєднує управлінські та аналітичні функції, ускладнює прийняття управлінських рішень та вчасне реагування на відхилення.

Постановка проблеми

Проблема полягає у відсутності ефективного інструменту для визначення рівня реалізації ІТ-проєкту на основі аналітичних показників KPI, який би дозволяв автоматично збирати, обробляти та відображати дані про стан етапів і завдань.

Традиційні підходи передбачають ручне введення інформації або використання розрізнених звітів, що знижує оперативність управління. Тому необхідно створити інформаційно-аналітичну систему (ІАС), яка забезпечить автоматизовану оцінку рівня реалізації проєкту, прозорість процесів і підтримку прийняття рішень [4].

Аналіз останніх досліджень і публікацій

Питання управління ефективністю IT-проєктів та оцінювання результативності команд розглядалися у працях багатьох вітчизняних і зарубіжних авторів. У роботах Д. Нортона та Р. Каплана [5] закладено основи концепції Balanced Scorecard, яка передбачає системний підхід до оцінювання ефективності за допомогою KPI. В українській науковій спільноті цю тематику досліджували І. Гончарук та Н. Юрчук [6], які акцентували на важливості побудови єдиного інформаційного простору для управління даними.

Сучасні публікації IT-компаній (Hurma, PeopleForce, FlexiProject) демонструють практичне застосування KPI як інструменту для оцінювання персоналу та команд [1]–[4]. Проте в більшості випадків розглядаються окремі аспекти — наприклад, оцінка продуктивності працівників або моніторинг виконання завдань, без урахування інтегрального рівня реалізації проєкту.

Таким чином, існує потреба у розробленні єдиної інформаційно-аналітичної системи, що забезпечить комплексне визначення рівня реалізації IT-проєкту на основі агрегованих KPI-показників.

Мета і завдання дослідження

Метою роботи є розроблення інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту на основі KPI-показників, що забезпечує моніторинг, аналіз та візуалізацію прогресу виконання завдань і етапів.

Для досягнення цієї мети необхідно розв’язати такі завдання:

Провести аналіз предметної області та існуючих інформаційних систем управління IT-проєктами.

Розробити структуру KPI-показників, релевантних для оцінки рівня реалізації IT-проєкту.

Створити інформаційну модель ІАС, що включає таблиці «Проєкти», «Етапи», «Завдання» та «Користувачі».

Реалізувати програмний модуль збору та обробки даних про прогрес виконання завдань.

Розробити інтерфейс візуалізації KPI та рівня реалізації проєкту у вигляді динамічних звітів.

Провести апробацію системи на прикладі тестових проєктів та оцінити її ефективність.

Об’єкт і предмет дослідження

Об’єкт дослідження — процес управління реалізацією IT-проєктів.

Предмет дослідження — моделі, методи та засоби побудови інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту з використанням KPI-показників.

Методи дослідження

Для досягнення поставленої мети застосовано комплекс методів:

Системний аналіз — для дослідження структури IT-проєкту та взаємозв’язків між його компонентами.

Математичне моделювання — для формалізації інтегрального показника реалізації проєкту:

Інформаційне моделювання — для створення структури бази даних та визначення зв’язків між сутностями «Проєкт», «Етап», «Завдання».

Програмна реалізація — із застосуванням технологій C#, WPF, Entity Framework Core.

Візуалізаційні методи — для відображення даних у вигляді прогрес-барів, таблиць та інтерактивних графіків.

Наукова новизна

Наукова новизна роботи полягає у:

Розробленні інтегральної моделі оцінювання рівня реалізації IT-проєкту на основі KPI-показників, що враховує вагові коефіцієнти етапів і завдань.

Створенні інформаційно-аналітичної системи, яка поєднує методи збирання даних, аналітичної обробки та динамічної візуалізації у єдиному середовищі.

Удосконаленні підходу до моніторингу прогресу проєктів шляхом впровадження KPI на рівні командної та індивідуальної ефективності.

Практичній реалізації прототипу IAC, який може бути інтегрований у корпоративні системи управління IT-компаній.

Апробація роботи.

Були опубліковані тези на тему: ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА ВИЗНАЧЕННЯ РІВНЯ РЕАЛІЗАЦІЇ ІТ ПРОЕКТУ тези були представлені в рамках XVI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ МОЛОДИХ ВЧЕНИХ «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА» 28-29 жовтня 2025 року.

Структура роботи

До складу пояснювальної записки входять: вступ, 4 розділи, висновки та 2 додатки. Загальний обсяг роботи становить 78 сторінки, основний текст роботи викладено на 66 сторінках.

У першому розділі проведено системний аналіз предметної області. Описано особливості процесів управління IT-проєктами, подано огляд сучасних програмних засобів для контролю виконання завдань і здійснено аналіз їхніх переваг і недоліків. На основі цього сформульовано мету, завдання й вимоги до створюваної системи.

Другий розділ присвячено моделюванню системи. Виконано функціональне моделювання з використанням UML-діаграм — прецедентів, діяльності, потоків даних. Наведено опис основних акторів, процесів взаємодії користувача із системою та логіки обробки інформації.

У третьому розділі здійснено розробку програмної частини системи. Подано архітектурну структуру програмного забезпечення, описано структуру бази даних, наведено алгоритми автентифікації користувачів, управління проєктами, формування статистики та звітів. Реалізацію виконано з використанням технологій C# (WPF) та Microsoft SQL Server.

Четвертий розділ містить результати дослідження системи. Визначено апаратні та програмні вимоги, проведено тестування основних модулів і оцінено

ефективність роботи розробленої системи. Наведено результати перевірки коректності виконання ключових функцій і сформульовано рекомендації щодо подальшого вдосконалення.

У висновках підсумовано результати роботи, узагальнено досягнуті цілі та окреслено напрями подальшого розвитку системи, зокрема впровадження веб-інтерфейсу та розширення аналітичних можливостей.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Опис предметної області

Інформаційні технології є фундаментом сучасної цифрової економіки, сприяючи автоматизації процесів, оптимізації управлінських рішень та підвищенню продуктивності праці. Особливої актуальності набуває ефективне управління IT-проєктами, яке визначає конкурентоспроможність компаній у сфері розроблення програмного забезпечення, інтеграції систем, кібербезпеки та аналітичних сервісів [7].

IT-проєкт, як правило, включає сукупність взаємопов’язаних завдань, що виконуються для досягнення конкретної мети — створення або вдосконалення інформаційного продукту чи послуги. Структура проєкту складається з етапів життєвого циклу, серед яких найпоширенішими є: ініціація, планування, виконання, моніторинг і завершення. На кожному з цих етапів формується набір показників, що характеризують стан і успішність виконання робіт [8].

Основними елементами управління IT-проєктом є:

Проєкт — базова одиниця управління, що має цілі, ресурси, часові межі та бюджет;

Етап (stage) — структурна частина проєкту, що об’єднує групу завдань, орієнтованих на досягнення проміжного результату;

Завдання (task) — мінімальна одиниця планування, яка виконується окремим фахівцем або підрозділом;

Команда проєкту — група фахівців, відповідальна за реалізацію проєкту згідно з планом та KPI.

Ефективне управління вимагає контролю не лише за виконанням завдань, але й за динамікою реалізації етапів та інтегральним прогресом усього проєкту. Для цього використовується система ключових показників ефективності (Key Performance Indicators — KPI) [9].

Поняття KPI у контексті IT-проєктів

KPI — це вимірювані параметри, що відображають ступінь досягнення визначених цілей. У контексті IT-проєктів вони дають змогу відстежувати

ефективність роботи команди, якість розроблення програмного продукту, своєчасність виконання завдань та дотримання бюджету [10].

Типові KPI для IT-проєктів поділяються на:

Процесні (кількість виконаних завдань, відсоток дотримання строків);

Якісні (кількість дефектів на 1000 рядків коду, успішність тестування);

Ресурсні (витрати часу, навантаження команди, відхилення від бюджету);

Комунікаційні (кількість міжкомандних взаємодій, ефективність мітингів);

Задоволеність клієнта (Customer Satisfaction Index, CSI).

Для комплексного оцінювання прогресу використовують інтегральну модель KPI (1.1):

$$R = \frac{\sum_{i=1}^{n} K_i \cdot W_i}{\sum_{i=1}^{n} W_i}, \qquad (1.1)$$

де R — рівень реалізації проєкту;

K_i— значення окремого показника KPI;

W_i — ваговий коефіцієнт, який відображає важливість показника.

Завдяки цій формулі можна отримати узагальнений показник прогресу, який дає змогу оцінювати стан проєкту не лише з точки зору виконаних завдань, а й з урахуванням їхнього впливу на кінцевий результат.

Проблеми управління реалізацією IT-проєктів

На практиці більшість IT-компаній стикається з типовими проблемами:

низька точність прогнозування строків виконання робіт;

неузгодженість дій між командами або підрозділами;

складність збору й аналізу показників KPI з різних джерел;

відсутність єдиного аналітичного простору для відстеження прогресу;

затримки у прийнятті управлінських рішень.

Зазначені проблеми посилюються, якщо управління здійснюється вручну або з використанням розрізнених інструментів (Excel, Trello, електронна пошта).

Як наслідок, відсутність інтегрованої інформаційно-аналітичної системи (ІАС) знижує прозорість процесу та унеможливлює об’єктивне визначення рівня реалізації проєкту [11].

Необхідність створення інформаційно-аналітичної системи

У зв’язку з цим виникає потреба у розробленні єдиної ІАС, що забезпечує:

централізоване зберігання інформації про проєкти, етапи, завдання та виконавців;

автоматизоване обчислення показників KPI;

побудову інтегрального індексу реалізації проєкту;

візуалізацію результатів у вигляді графіків, діаграм і звітів;

можливість прогнозування термінів завершення робіт.

Такі системи дозволяють скоротити людський фактор, підвищити точність оцінювання та створити умови для впровадження data-driven управління в IT-компаніях [12].

Таким чином, предметна область розроблення інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту охоплює процеси збору, оброблення, аналізу та візуалізації даних, що характеризують стан проєкту.

Подальший розвиток цієї області передбачає інтеграцію аналітичних алгоритмів, машинного навчання та предиктивних моделей, здатних не лише оцінювати, а й прогнозувати динаміку реалізації проєкту.

1.2. Огляд інформаційних джерел та аналіз існуючих рішень

Ефективне управління IT-проєктами вимагає постійного моніторингу прогресу, контролю якості виконання завдань, координації командної роботи та своєчасного прийняття управлінських рішень. Для цього застосовуються різні програмні засоби, що реалізують методології Agile, Scrum, Kanban або Waterfall.

На сучасному ринку наявна велика кількість інформаційних систем управління проєктами, проте більшість із них орієнтовані на організацію завдань, а не на аналітичну оцінку рівня реалізації проєкту [13].

У таблиці 1.1 подано порівняльний аналіз найпопулярніших систем управління проєктами за основними критеріями: підтримка KPI, наявність аналітичних модулів, можливість інтеграції та тип візуалізації даних.

Таблиця 1.1 – Порівняння систем управління IT-проєктами за критеріями ефективності

Система	**Основні можливості**	**Аналітика KPI**	**Інтеграції**	**Тип візуалізації**	**Обмеження**
Jira Software (Atlassian)	Управління завданнями, спринтами, користувачами	Часткова (через плагіни eazyBI, BigPicture)	GitHub, Bitbucket, Slack	Дошки, графіки, діаграми Burndown	Висока вартість ліцензії
Asana	Планування, контроль завдань, командна співпраця	Обмежена (тільки звіти виконання)	Google Workspace, Slack, Teams	Таймлайни, календарі	Без KPI-панелі
Trello	Канбан-дошки, спільна робота	Відсутня	Google Drive, Dropbox	Картки, списки	Не підтримує KPI та звіти
FlexiProject	Управління цілями, показниками, задачами	Є (вбудовані KPI та графіки)	Excel, Power BI	Панель показників, діаграми	Не орієнтована на IT-проєкти

Як видно з таблиці, жодна з розглянутих систем не забезпечує повноцінної аналітики рівня реалізації IT-проєкту, що враховує як виконання завдань, так і досягнення KPI на різних рівнях (етап, проєкт, команда). Найближче до цього підходу стоїть FlexiProject, однак її архітектура більше орієнтована на корпоративне управління, а не на технічну розробку [14].

Методологічні підходи до оцінювання ефективності

У наукових і практичних дослідженнях широко використовуються підходи до аналізу ефективності на основі Balanced Scorecard (BSC), розробленої Капланом і Нортоном [15]. Вона передбачає балансування між фінансовими, клієнтськими, внутрішніми процесами та розвитком персоналу.

У контексті ІТ-проєктів ця концепція трансформується в систему KPI, де кожен показник відображає один із аспектів ефективності:

$$E = \sum_{i=1}^{n} K_i \cdot C_i \quad (1.2)$$

Де E — інтегральна ефективність ІТ-проєкту;

K_i — значення окремого KPI (0–1);

C_i — ваговий коефіцієнт важливості показника.

Для підвищення точності оцінки рівня реалізації проєкту доцільно об’єднати показники на рівнях:

- Task level — відсоток завершених завдань;
- Stage level — прогрес етапу;
- Project level — агрегована ефективність усього проєкту.

В IAC пропонується використати трирівневу структуру розрахунку KPI, де кінцевий показник визначається як середньозважене значення ефективності етапів:

$$R_p = \frac{\sum_{j=1}^{m} R_j \cdot W_j}{\sum_{j=1}^{m} W_j} \quad (1.3)$$

де R_p — рівень реалізації проєкту;

R_j — рівень виконання j-го етапу;

W_j — ваговий коефіцієнт етапу.

Таким чином, запропонований підхід дозволяє отримати динамічну модель контролю реалізації проєкту, яка поєднує методи управління завданнями та аналітичні KPI-індикатори.

Проблеми існуючих систем та напрями вдосконалення

Попри широкий вибір комерційних платформ, існують певні обмеження:

- відсутність єдиної бази даних для зберігання KPI усіх рівнів;
- складність інтеграції з внутрішніми системами компанії;
- недостатня гнучкість у налаштуванні аналітичних показників;
- висока вартість ліцензування та підтримки;
- залежність від хмарних рішень без можливості локальної інсталяції.

Ці недоліки створюють передумови для розроблення власної інформаційно-аналітичної системи (IAC), що забезпечує:

- автоматизований збір даних з бази SQL;
- розрахунок KPI-показників у реальному часі;
- візуалізацію рівня реалізації проєкту у вигляді графіків і прогрес-барів;
- гнучкість налаштувань для різних типів проєктів.

Розроблена система покликана усунути недоліки типових інструментів управління IT-проєктами, створивши умови для впровадження data-driven управління, що ґрунтується на даних та кількісному аналізі ефективності [16].

Проведений аналіз показав, що більшість сучасних систем управління IT-проєктами не забезпечують інтегрованого аналітичного інструментарію для вимірювання рівня реалізації проєкту.

Застосування KPI у поєднанні з аналітичними моделями дозволяє створити інформаційно-аналітичну систему, яка формує об’єктивну картину стану проєкту, підтримує ухвалення управлінських рішень і підвищує ефективність командної роботи.

1.3 Постановка задачі

Проведений аналіз показав, що існуючі інформаційні системи управління IT-проєктами орієнтовані переважно на моніторинг завдань і планування ресурсів, однак не забезпечують комплексного аналітичного підходу до оцінювання рівня реалізації проєкту. Це створює потребу у розробленні системи, яка поєднує інструменти управління з механізмами KPI-аналізу.

У межах поставленої мети необхідно вирішити такі завдання:

Проаналізувати процес управління IT-проєктом і виділити основні сутності системи.

Розробити структуру бази даних для зберігання даних про проєкти, етапи, завдання, користувачів та KPI.

Побудувати математичну модель оцінювання рівня реалізації IT-проєкту.

Реалізувати програмний модуль розрахунку інтегральних показників ефективності.

Створити інтерфейс користувача для введення, обробки та візуалізації даних.

Провести тестування розробленої системи та оцінити її функціональні можливості.

Розробка інформаційно-аналітичної системи для визначення рівня реалізації ІТ-проєкту є актуальним завданням, яке поєднує принципи управління проєктами, математичне моделювання KPI та засоби програмної аналітики.

Результатом дослідження стане програмний продукт, який дозволяє автоматизувати обчислення показників ефективності, забезпечити прозорість процесу управління та надати керівництву об’єктивну аналітичну інформацію про стан реалізації проєкту.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1. Функціональне моделювання системи

Функціональне моделювання є одним із основних етапів проєктування інформаційних систем, оскільки воно дозволяє визначити структуру процесів, учасників та взаємозв'язки між елементами програмного забезпечення.

Для опису системи використано уніфіковану мову моделювання UML, яка є міжнародним стандартом для структурного та поведінкового аналізу програмних систем.

Запропонована інформаційно-аналітична система призначена для автоматизованого моніторингу реалізації IT-проєктів та оцінки ефективності роботи команди на основі ключових показників ефективності (KPI).

Система реалізує рольову модель користувачів і забезпечує керування проєктами, етапами та завданнями з подальшим формуванням аналітичних зрізів та звітів.

2.1.1. Визначення основних акторів системи

Основними користувачами системи є:

1. Адміністратор — відповідає за технічну підтримку та безпеку даних. До його функцій належать:
   - створення та видалення облікових записів користувачів;
   - призначення ролей (менеджер, виконавець);
   - контроль доступу до інформації.
2. Менеджер проєкту — є основним користувачем системи. Він виконує такі дії:
   - створює нові проєкти та задає основні параметри (назву, опис, терміни);
   - розподіляє проєкт на етапи, додає завдання;
   - призначає виконавців і контролює статус виконання;
   - аналізує результати роботи за допомогою KPI та генерує звіти.
3. Виконавець — виконує завдання, призначені менеджером. Основні його дії:

- перегляд списку завдань;
- зміна статусу завдань (“у процесі”, “завершено”).

Таким чином, система реалізує рольову модель доступу, де кожен користувач має свій набір дозволених дій, що підвищує безпеку та зручність використання.

2.1.2. Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів описує сценарії взаємодії користувачів із системою [17]. Кожен прецедент представляє конкретну функцію, доступну певній ролі користувача.

Взаємозв’язки між акторами та системою показано на рисунку 2.1.

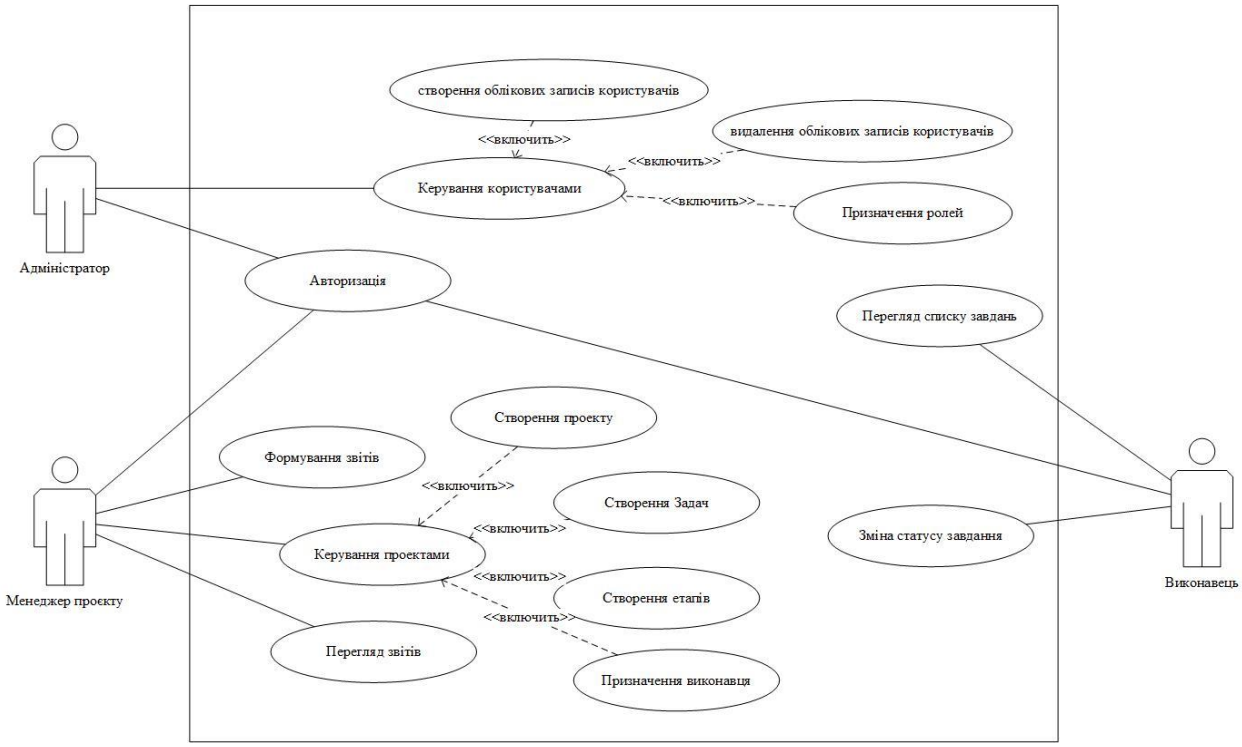


Рисунок 2.1 – UML-діаграма прецедентів інформаційно-аналітичної системи визначення рівня реалізації ІТ-проєкту

Основні прецеденти включають:

- Авторизація користувача — перевірка логіну та пароля, контроль доступу;
- Створення проєкту — формування нової структури даних у базі;
- Додавання етапів і завдань — наповнення проєкту деталізованими процесами;

- Призначення виконавців — визначення відповідальних користувачів;
- Оновлення статусів — відображення поточного стану завдань і KPI;
- Формування аналітичних звітів — побудова графіків, діаграм та показників ефективності.

Виконання кожного прецеденту активує відповідний модуль бізнес-логіки системи, який звертається до бази даних через Entity Framework Core.

2.1.3. Діаграма діяльності (Activity Diagram)

Діаграма діяльності показує послідовність кроків користувача під час роботи із системою[18]. Процес управління IT-проєктом відбувається за таким алгоритмом:

1. Користувач входить у систему (етап авторизації).
2. Менеджер створює новий проєкт, задаючи його параметри.
3. До проєкту додаються етапи з визначеними датами початку і завершення.
4. Кожен етап наповнюється завданнями, які призначаються конкретним виконавцям.
5. Виконавці оновлюють статус своїх завдань після їх завершення.
6. Система автоматично розраховує часткові та інтегральні показники ефективності.
7. Менеджер переглядає результати та формує звіти.

Візуальну схему цієї послідовності дій показано на рисунку 2.2.

Рисунок 2.2 – UML-діаграма діяльності процесу реалізації IT-проєкту

На діаграмі відображено вузли прийняття рішень (наприклад, перевірка стану виконання завдання), паралельні процеси (робота різних виконавців) та завершення циклу проєкту.

2.1.4. Діаграма потоків даних (Data Flow Diagram, DFD)

Для опису потоків інформації між компонентами системи побудовано DFD-діаграму, яка подана на рисунку 2.3 [19].

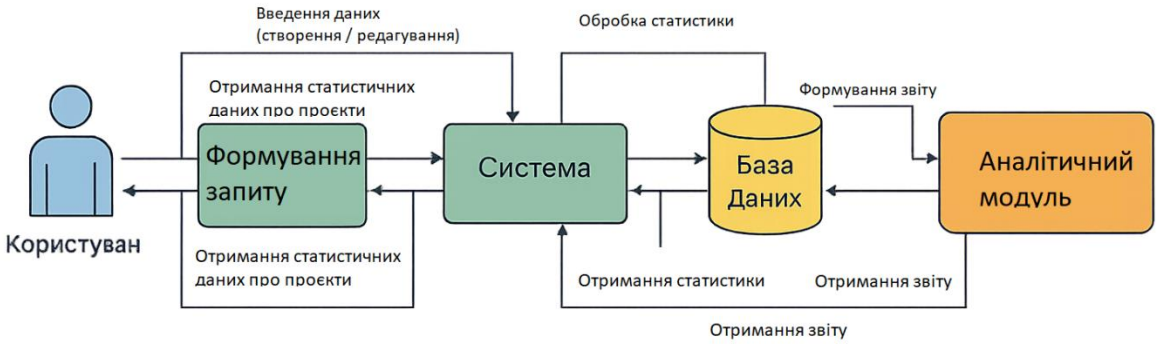


Рисунок 2.3 – DFD-діаграма потоків даних інформаційно-аналітичної системи

Основні об’єкти DFD:

- Користувач — вводить або запитує інформацію;
- Система — обробляє дані, виконує логіку KPI;
- База даних — зберігає інформацію про проєкти, етапи, завдання, користувачів і показники;
- Аналітичний модуль — генерує графіки, таблиці та звіти.

Основні потоки даних:

- “Введення даних” (створення/редагування проєкту);
- “Запис у БД” (збереження змін через EF Core);
- “Отримання статистики” (вибірка даних SQL-запитами);
- “Формування звіту” (агрегація результатів і візуалізація).

Цей рівень моделювання дає змогу відстежити шлях даних від користувача до бази і назад, що є важливим для забезпечення цілісності та узгодженості інформації.

Побудовані UML-діаграми прецедентів, діяльності та потоків даних відображають логіку функціонування системи й основні сценарії взаємодії користувачів.

Ці моделі є підґрунтям для розроблення об’єктно-орієнтованої моделі системи, та подальшого проєктування бази даних.

2.2. Об’єктно-орієнтоване моделювання системи

Об’єктно-орієнтоване моделювання (ООМ) є важливим етапом розроблення програмного забезпечення, який забезпечує формування архітектури системи на основі принципів інкапсуляції, успадкування та поліморфізму.

Метою даного розділу є створення об’єктної моделі інформаційно-аналітичної системи визначення рівня реалізації ІТ-проєкту, що визначає взаємозв’язки між об’єктами, їх атрибутами та методами, а також поведінку об’єктів у процесі функціонування системи.

2.2.1. Діаграма класів (Class Diagram)

Діаграма класів є центральним елементом об’єктно-орієнтованої моделі, оскільки вона відображає основні сутності системи, їхні властивості, операції та зв’язки між ними[20]. Для розробленої системи було виділено такі ключові класи:

- User – описує користувачів системи. Атрибути: Id, Username, Password, Role, CreatedAt. Клас містить методи аутентифікації та управління правами доступу.
- Project – основна сутність системи, що представляє ІТ-проєкт. Атрибути: Id, Name, Description, StartDate, EndDate, Status. Має асоціацію “один-до-багатьох” з класом ProjectStage.
- ProjectStage – описує етапи реалізації проєкту. Атрибути: Id, ProjectId, StageName, PlannedStartDate, ActualStartDate, Status. Пов’язаний із класом Project і має зв’язок “один-до-багатьох” із StageTask.
- StageTask – представляє завдання в межах певного етапу. Атрибути: Id, StageId, TaskName, AssignedToUserId, Status, PlannedDate, ActualDate. Містить методи UpdateStatus() та CalculateProgress().

- KPIIndicator – клас, який відповідає за обчислення показників ефективності (KPI). Атрибути: Id, Name, Value, Weight, ProjectId. Методи: CalculateKPI(), GetProjectKPI().
- ReportGenerator – допоміжний клас, що формує звіти у форматах PDF та Excel, використовуючи дані з проєктів, етапів і KPI.
- DatabaseManager – клас для роботи з базою даних через Entity Framework Core.

Взаємозв'язки між класами показано на рисунку 2.4.

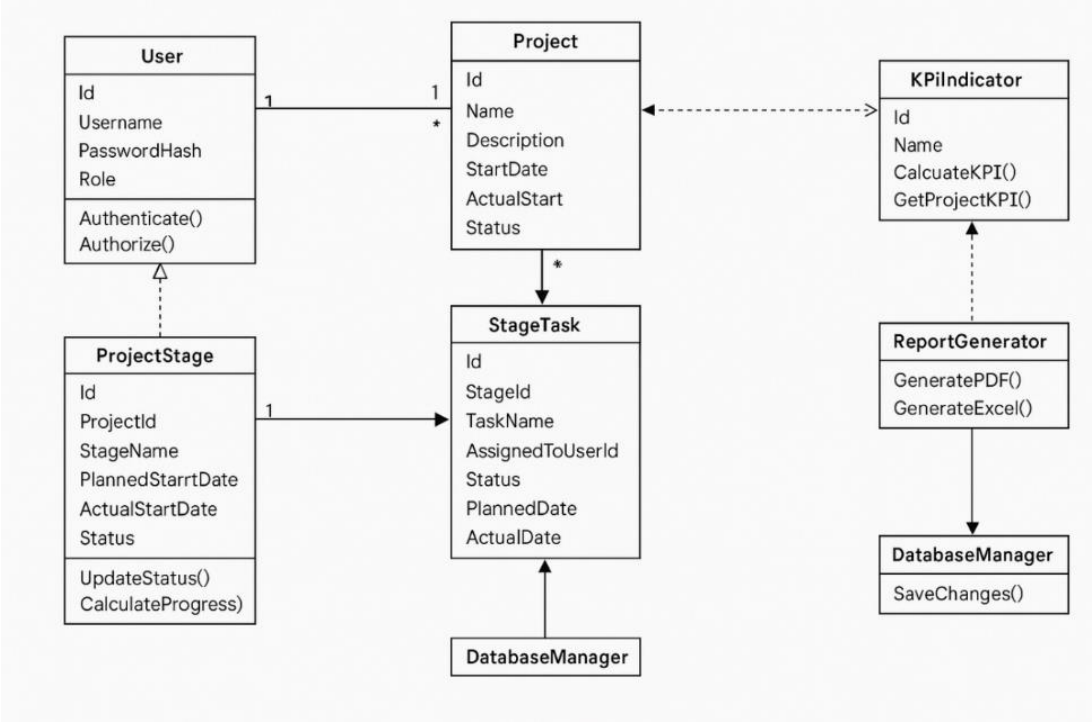


Рис.2.4 – UML-діаграма класів інформаційно-аналітичної системи

На діаграмі відображено такі типи зв'язків:

- асоціації між Project, ProjectStage, StageTask;
- агрегацію між User і StageTask (виконавець призначений на завдання);
- залежність ReportGenerator від класів даних (Project, KPIIndicator);
- композицію між Project і ProjectStage, що вказує на ієрархічну структуру даних.

2.2.2. Діаграма послідовності (Sequence Diagram)

Діаграма послідовності моделює динамічну поведінку системи, тобто порядок обміну повідомленнями між об'єктами під час виконання певного

сценарію [21]. На рисунку 2.5 подано приклад діаграми послідовності для сценарію «Розрахунок рівня реалізації проєкту».

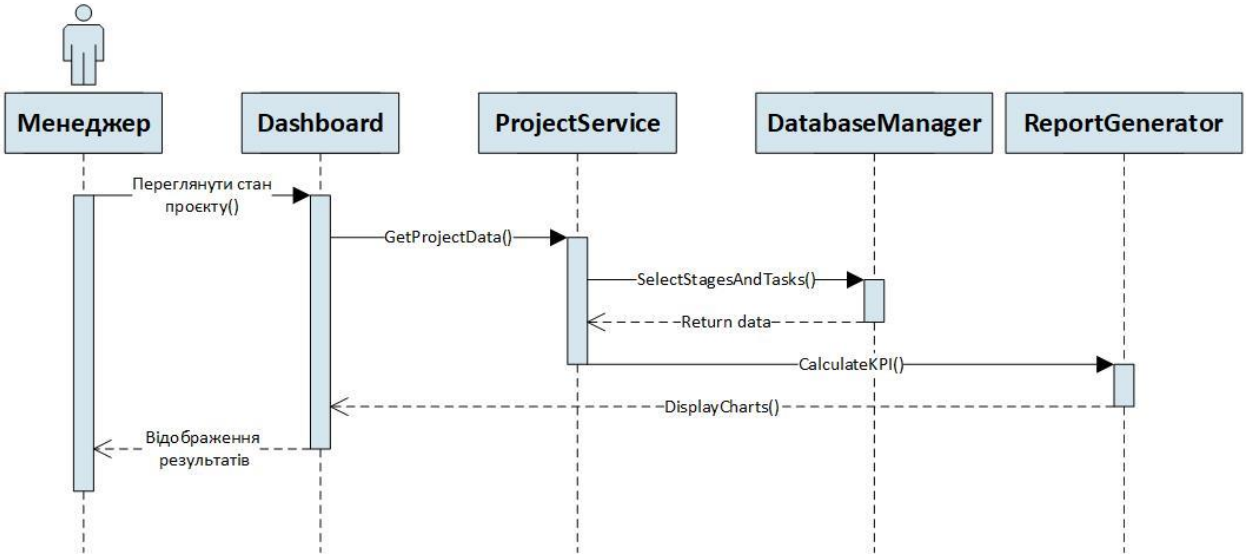


Рис.2.5 – Діаграма послідовності сценарію розрахунку рівня реалізації IT-проєкту

Послідовність дій:

1. Менеджер ініціює запит на перегляд стану проєкту.
2. Об’єкт ProjectService звертається до DatabaseManager для вибірки всіх етапів і завдань.
3. Для кожного етапу обчислюється середній показник виконання (StageTask.Status).
4. Розраховується інтегральний показник ефективності проєкту за формулою.
5. Система передає результати до модуля ReportGenerator, який формує графічне відображення (діаграми, прогрес-бари).
6. Менеджер переглядає підсумкові дані у вікні аналітичної панелі (Dashboard).

2.2.3. Діаграма станів (State Diagram)

Діаграма станів описує можливі зміни статусу об’єкта під час його життєвого циклу. Для системи особливий інтерес становить життєвий цикл завдання (StageTask).

На рисунку 2.6 подано UML-діаграму станів завдання.

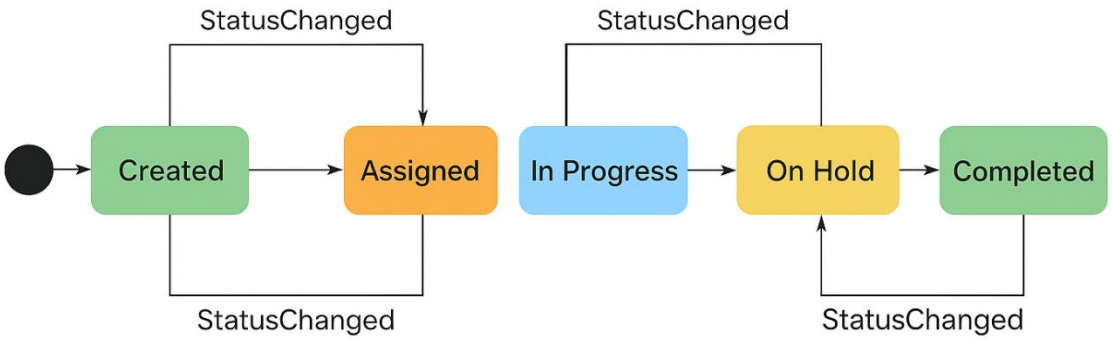


Рис.2.6 – UML-діаграма станів об’єкта “Завдання”

Основні стани:

- Created — завдання створено, але ще не призначено виконавцю;
- Assigned — завдання призначено конкретному користувачу;
- In Progress — виконавець розпочав роботу;
- On Hold — виконання тимчасово призупинене;
- Completed — завдання виконано та перевірено менеджером.

Перехід між станами ініціюється користувачем або системою автоматично.

Для контролю використовується тригер події StatusChanged, що дозволяє оновлювати KPI у реальному часі.

У результаті об’єктно-орієнтованого моделювання побудовано UML-діаграми, що відображають як статичну (класи), так і динамічну (послідовність, стани) структуру системи. Отримані моделі формують основу для етапу проєктування бази даних і реалізації програмного коду WPF-застосунку, що розглядається в наступному розділі.

3. РОЗРОБКА СИСТЕМИ

3.1 Архітектура інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту

Архітектура створеної інформаційно-аналітичної системи побудована на основі багаторівневої клієнт-серверної моделі, яка передбачає чітке розділення логіки роботи програми на окремі структурні рівні. Такий підхід забезпечує логічну впорядкованість системи, спрощує процес її супроводу та розширення, а також підвищує надійність роботи при збільшенні обсягів даних або кількості користувачів.

Загалом система складається з трьох основних рівнів:

1. рівень представлення (Presentation Layer);
2. рівень бізнес-логіки (Service Layer);
3. рівень даних (Data Layer).

Кожен із них виконує окрему функцію в межах програмного комплексу та взаємодіє з іншими рівнями через чітко визначені інтерфейси. На рисунку 3.1 наведено архітектурну схему розробленої системи.

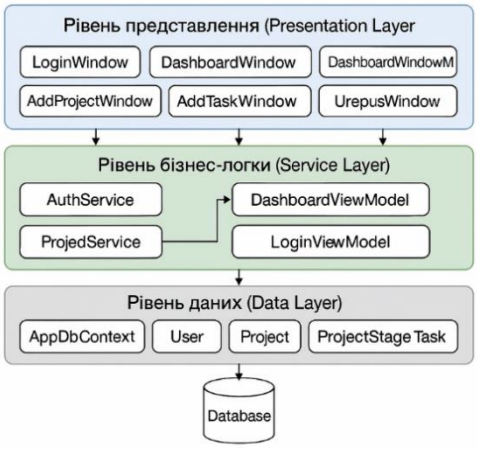


Рис. 3.1 Архітектура інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту

3.1.1 Рівень даних (Data Layer)

Рівень даних є базовим у структурі системи, оскільки саме тут зберігається вся інформація, необхідна для роботи користувачів і модулів бізнес-логіки.

Даний рівень реалізовано на основі системи управління базами даних Microsoft SQL Server з використанням технології Entity Framework Core, що дозволяє працювати з даними у вигляді об’єктів, а не безпосередньо через SQL-запити.

Основними сутностями бази даних є:

- User – інформація про користувачів системи: логін, пароль, роль та дата створення облікового запису.
- Project – опис основних параметрів IT-проєкту: назва, короткий опис, дати початку та завершення, статус реалізації.
- ProjectStage – відомості про окремі етапи реалізації проєкту, їхні планові та фактичні терміни.
- StageTask – перелік завдань у межах кожного етапу, із зазначенням відповідального виконавця, статусу та термінів.
- AppDbContext – контекст бази даних, який відповідає за зв’язок між програмними моделями та таблицями у СУБД, забезпечує запити, вставлення, оновлення й видалення даних.

Такий підхід дозволяє централізовано керувати інформацією та мінімізувати дублювання даних.

Завдяки ORM-технології уся взаємодія із базою даних відбувається через класи моделей, що забезпечує безпечну роботу з інформацією та підвищує швидкодію системи.

3.1.2 Рівень бізнес-логіки (Service Layer)

Середній рівень архітектури є ядром системи, оскільки саме тут зосереджена основна логіка обробки даних і реалізації бізнес-процесів.

На цьому рівні розташовані класи-сервіси та ViewModel-модулі, які здійснюють опрацювання запитів від користувацького інтерфейсу, звернення до бази даних через AppDbContext та повернення результатів у зручному для відображення вигляді.

До складу цього рівня входять такі компоненти:

- AuthService – виконує перевірку облікових даних користувача під час входу до системи, а також контролює розмежування прав доступу залежно від ролі (адміністратор, менеджер або користувач).
- ProjectService – реалізує основну логіку управління проєктами, етапами та завданнями. Через цей сервіс здійснюється додавання, редагування, видалення записів, а також обчислення відсотка виконання проєкту та його етапів.
- DashboardViewModel – забезпечує формування аналітичних показників для головної панелі адміністратора, узагальнюючи дані про кількість активних проєктів, виконаних етапів та стан завдань.
- LoginViewModel – відповідає за взаємодію між інтерфейсом авторизації (LoginWindow) і сервісом автентифікації, обробляє введені користувачем дані та передає результат на відображення.

Таким чином, рівень бізнес-логіки слугує своєрідним посередником між користувачем та базою даних, здійснюючи обробку інформації, перевірку коректності введених даних, виконання розрахунків і передачу результатів далі до рівня представлення.

3.1.3 Рівень представлення (Presentation Layer)

Найвищим рівнем архітектури є рівень представлення, який відповідає за взаємодію користувача з системою.

Усі елементи інтерфейсу реалізовано за допомогою технології Windows Presentation Foundation (WPF) на мові C#, що дозволяє створити сучасний графічний інтерфейс із використанням прив’язки даних (Data Binding) та шаблонів стилів (XAML).

До складу цього рівня входять вікна, які забезпечують виконання окремих функцій системи:

- LoginWindow – форма автентифікації користувачів.
- DashboardWindow – головне вікно адміністратора з аналітичними показниками та переліком усіх проєктів.

- DashboardWindowM – панель менеджера, орієнтована на роботу з етапами та завданнями.
- DashboardWindowU – робоче вікно користувача, у якому відображаються призначені йому завдання.
- AddProjectWindow, AddStageWindow, AddTaskWindow – форми для додавання нових записів у базу даних.
- ProjectDetailsWindow – детальний перегляд структури проєкту, його етапів, завдань і графічних звітів.
- ReportsWindow – модуль формування звітів і побудови діаграм типу «План / Факт».
- UsersWindow – інтерфейс для керування обліковими записами користувачів системи.

Кожне вікно прив’язане до відповідного ViewModel-класу, що реалізує логіку обробки дій користувача.

Завдяки цьому забезпечується реалізація шаблону MVVM (Model-View-ViewModel), який є стандартом у розробці WPF-додатків і дозволяє розділити логіку, дані та інтерфейс.

Взаємодія між рівнями

Обмін інформацією між рівнями здійснюється зверху вниз:

1. Користувач працює з графічним інтерфейсом (Presentation Layer), який формує запити на отримання чи оновлення даних.
2. Ці запити передаються до сервісів (Service Layer), де відбувається їх логічна обробка, перевірка прав доступу та виконання відповідних дій.
3. Після цього дані зберігаються або вибираються з бази через AppDbContext (Data Layer).
4. Отримана інформація у зручному вигляді повертається назад на рівень інтерфейсу для відображення користувачу.

Такий підхід забезпечує незалежність рівнів і дозволяє у майбутньому модернізувати будь-яку частину системи без впливу на інші компоненти.

Наприклад, можна змінити структуру бази даних або оновити інтерфейс користувача без необхідності переробки всієї програми.

3.2 Структура бази даних системи

База даних є центральною складовою інформаційно-аналітичної системи визначення рівня реалізації ІТ-проєкту, оскільки саме вона забезпечує зберігання, цілісність і узгодженість усіх даних. Структура бази побудована на основі реляційної моделі, реалізованої засобами Microsoft SQL Server.

Під час проєктування бази даних застосовано принципи нормалізації до третьої нормальної форми (3НФ), що дозволило уникнути дублювання інформації, підвищити ефективність запитів та забезпечити логічну узгодженість між сутностями.

База даних містить чотири основні таблиці: Users, Projects, ProjectStages та StageTasks, які пов’язані між собою відношеннями типу «один-до-багатьох».

Така структура відображає ієрархічну модель виконання ІТ-проєкту:

Проєкт → Етап → Завдання → Виконавець.

Таблиця користувачів (Users)

Таблиця Users призначена для зберігання інформації про користувачів системи. Вона використовується для автентифікації, авторизації та розмежування прав доступу.

Кожен користувач має унікальне ім’я, пароль, дату створення облікового запису та роль у системі (адміністратор, менеджер або користувач).

Структура таблиці Users відображена в таблиці 3.2.1.

Таблиця 3.2.1 – Структура таблиці Users

Поле	Тип даних	Опис
Id	int	Первинний ключ користувача.
Username	nvarchar(100)	Унікальне ім’я користувача для входу.
Password	nvarchar(256)	Пароль користувача.
CreatedAt	datetime	Дата створення облікового запису.
Role	nvarchar(50)	Роль користувача (Admin, Manager, User).

Таблиця Users має зв’язок із таблицею StageTasks через поле AssignedToUserId, що дозволяє визначати виконавця кожного завдання. Тип зв’язку — «один користувач може виконувати багато завдань» (1:N).

Таблиця Projects є базовою у структурі системи. Кожен запис у ній відповідає одному IT-проєкту, що містить основну інформацію: назву, опис, дати початку та завершення, а також статус виконання.

Структура таблиці Projects відображена в таблиці 3.2.2.

Таблиця 3.2.2 – Структура таблиці Projects

Поле	Тип даних	Опис
Id	int	Первинний ключ проєкту.
Name	nvarchar(200)	Назва проєкту.
Description	nvarchar(MAX)	Короткий опис або мета проєкту.
StartDate	date	Планова дата початку виконання.
EndDate	date	Планова дата завершення.
Status	nvarchar(50)	Поточний стан (Заплановано, В процесі, Завершено).

Таблиця Projects пов’язана з таблицею ProjectStages за полем ProjectId (зв’язок «один-до-багатьох»). Один проєкт може містити декілька етапів, і при його видаленні всі відповідні етапи видаляються каскадно.

Таблиця ProjectStages деталізує реалізацію кожного проєкту, розділяючи його на окремі етапи. У ній фіксуються планові та фактичні дати початку й завершення робіт, а також статус етапу, що дозволяє відстежувати динаміку реалізації. Структура таблиці ProjectStages відображена в таблиці 3.2.3.

Таблиця 3.2.3 – Структура таблиці ProjectStages

Поле	Тип даних	Опис
Id	int	Ідентифікатор етапу (первинний ключ).
ProjectId	int	Посилання на проєкт (зовнішній ключ до Projects).
StageName	nvarchar(100)	Назва етапу.
PlannedStartDate	date	Планова дата початку етапу.
PlannedEndDate	date	Планова дата завершення етапу.
ActualStartDate	date	Фактична дата початку робіт.
ActualEndDate	date	Фактична дата завершення.
Status	nvarchar(50)	Стан етапу (Заплановано, В процесі, Завершено).

Між таблицями ProjectStages та StageTasks реалізовано зв’язок «один-до-багатьох» через поле StageId. Це дозволяє контролювати виконання кожного етапу через набір завдань.

Таблиця StageTasks є найдетальнішою у базі даних і містить інформацію про конкретні завдання, їх виконавців, планові та фактичні дати виконання, а також статус виконання. Кожне завдання належить певному етапу і може бути призначене конкретному користувачу. Структура таблиці StageTasks відображена в таблиці 3.2.4.

Таблиця 3.2.4 – Структура таблиці StageTasks

Поле	Тип даних	Опис
Id	int	Ідентифікатор завдання (первинний ключ).
StageId	int	Посилання на етап (ProjectStages).
TaskName	nvarchar(200)	Назва завдання.
AssignedToUserId	int	Виконавець завдання (посилання на Users).
PlannedDate	date	Планова дата виконання завдання.
ActualDate	date	Фактична дата виконання.
Status	nvarchar(50)	Поточний стан (Заплановано, В процесі, Виконано).

Таким чином, таблиця StageTasks має два зовнішні ключі — до таблиць ProjectStages та Users, що забезпечує гнучкість і повну відтворюваність взаємозв’язків між проєктами, етапами та виконавцями.

У базі даних реалізовано такі основні відношення:

- Projects (1) → (N) ProjectStages — один проєкт містить декілька етапів;
- ProjectStages (1) → (N) StageTasks — кожен етап має кілька завдань;
- Users (1) → (N) StageTasks — один користувач може бути виконавцем багатьох завдань.

Ці зв’язки формують ієрархічну модель управління:

Проєкт об’єднує етапи, етап — завдання, а завдання має виконавця.

Розроблена структура бази даних повністю відповідає вимогам предметної області, забезпечує логічну послідовність між сутностями системи та можливість

її подальшого розширення. У разі необхідності до структури можуть бути додані додаткові таблиці для зберігання історії змін, коментарів чи документів.

На рисунку 3.2 представлено остаточну ER-діаграму бази даних інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту.

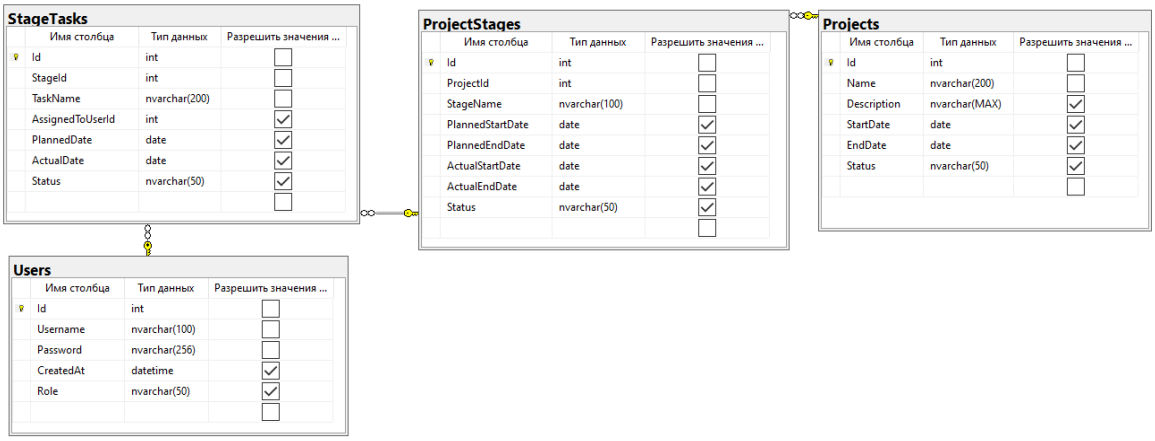


Рис. 3.2 – ER-діаграма бази даних інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту

3.3 Алгоритми обробки інформації в системі

3.3.1 Алгоритм автентифікації користувачів

Процес автентифікації є початковим етапом роботи користувача з інформаційно-аналітичною системою визначення рівня реалізації IT-проєкту. Його основною метою є перевірка достовірності введених облікових даних та надання доступу до функціональних модулів відповідно до ролі користувача.

Реалізація механізму автентифікації базується на поєднанні трьох основних компонентів:

- класу User, який визначає модель користувача системи;
- сервісу AuthService, що виконує логіку перевірки логіну та пароля;
- вікна LoginWindow, яке забезпечує інтерфейс взаємодії користувача з системою.

Під час запуску програми користувачеві відображається форма авторизації (рис. 3.3.1). Вона містить поля для введення логіну та пароля і кнопку «Увійти», натискання якої ініціює перевірку введених даних.

```
<TextBlock Text="Логін:" FontSize="14"/>
<TextBox x:Name="UsernameBox" Grid.Row="1" Margin="0,5,0,10" Height="25"/>
<TextBlock Text="Пароль:" Grid.Row="2" FontSize="14"/>
<PasswordBox x:Name="PasswordBox" Grid.Row="3" Margin="0,5,0,10" Height="25"/>
<Button Content="Увійти" Grid.Row="3" Height="35" Margin="0,40,0,-25"
        Click="Login_Click" Background="#4CAF50" Foreground="White"/>
```

Рис. 3.3.1 – Фрагмент XAML-коду вікна авторизації LoginWindow

Після натискання кнопки «Увійти» у класі LoginWindow.xaml.cs викликається метод Login_Click, який перевіряє заповнення полів, звертається до бази даних і здійснює пошук користувача за іменем (рис. 3.3.2).

```
private void Login_Click(object sender, RoutedEventArgs e)
{
    string username = UsernameBox.Text.Trim();
    string password = PasswordBox.Password.Trim();
    if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Заповніть усі поля!", "Помилка",
            MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }
    var user = _context.Users.FirstOrDefault(u => u.Username == username && u.PasswordHash == password);
    if (user == null)
    {
        MessageBox.Show("Невірний логін або пароль!", "Помилка",MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
    switch (user.Role?.ToLower())
    {
        case "admin":
            new DashboardWindow().Show(); break;
        case "manager":
            new DashboardWindowM().Show(); break;
        case "user":
            new DashboardWindowU(user.Id).Show(); break;
        default:
            MessageBox.Show("Невідома роль користувача!"); break;
    }
    this.Close();
}
```

Рис. 3.3.2 – Фрагмент коду обробника події Login_Click у класі LoginWindow

У разі коректного введення облікових даних система звертається до бази даних через клас AppDbContext і виконує пошук запису в таблиці Users. Якщо запис знайдено — система визначає роль користувача та відкриває відповідне головне вікно:

- DashboardWindow — для адміністратора,
- DashboardWindowM — для менеджера,
- DashboardWindowU — для звичайного користувача.

Таким чином, забезпечується багаторівневий доступ до функціональних модулів системи.

Для перевірки пароля використовується клас AuthService, який інкапсулює логіку пошуку користувача в базі даних і звірення хешованого пароля.

Це дозволяє відокремити бізнес-логіку автентифікації від інтерфейсного шару (рис. 3.3.3).

```
public class AuthService
{
    private readonly AppDbContext _context;

    public AuthService(AppDbContext context)
    {
        _context = context;
    }

    public bool Login(string username, string password)
    {
        if (string.IsNullOrWhiteSpace(username) ||
            string.IsNullOrWhiteSpace(password))
            return false;

        var user = _context.Users.FirstOrDefault(u => u.Username == username);
        if (user == null) return false;

        return PasswordHelper.VerifyPassword(password, user.PasswordHash);
    }
}
```

Рис. 3.3.3 – Фрагмент коду класу AuthService

Метод Login() виконує послідовно три кроки:

1. Перевіряє коректність вхідних даних (непорожні поля).
2. Здійснює пошук користувача в таблиці Users за іменем.
3. Перевіряє відповідність введеного пароля хешу, що зберігається в полі PasswordHash.

Для представлення користувачів у базі даних створено клас User, який відповідає структурі таблиці Users. Модель містить основні поля: ідентифікатор, логін, пароль, роль і дату створення облікового запису (рис. 3.3.4).

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; } = string.Empty;
    public string PasswordHash { get; set; } = string.Empty;
    public string Role { get; set; } = "User";
    public DateTime CreatedAt { get; set; } = DateTime.Now;
}
```

Рис. 3.3.4 – Фрагмент коду моделі User

На рисунку 3.3.5 наведено спрощену блок-схему процесу автентифікації користувача, яка ілюструє послідовність дій при вході до системи:

1. Введення логіну та пароля.
2. Перевірка на порожні поля.
3. Пошук користувача в базі даних.
4. Перевірка відповідності пароля.
5. Визначення ролі користувача.
6. Відкриття відповідного інтерфейсу або виведення повідомлення про помилку.

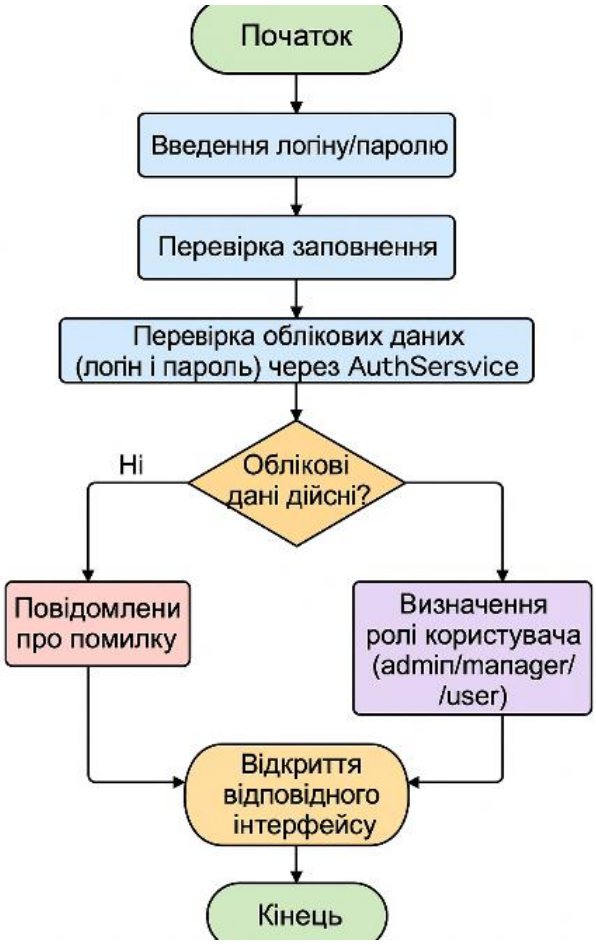


Рис. 3.3.5 – Блок-схема алгоритму автентифікації користувача

Таким чином, реалізований алгоритм автентифікації забезпечує безпечний і централізований доступ до системи, а розмежування ролей гарантує контроль над виконанням функцій відповідно до компетенції користувачів. Логічна ізоляція компонентів (модель – сервіс – інтерфейс) підвищує масштабованість і спрощує супровід програмного коду.

3.3.2 Алгоритм управління проєктами та їх етапами

Підсистема управління проєктами та їх етапами є ключовим функціональним елементом інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту. Її основне завдання — забезпечити реєстрацію нових проєктів, створення етапів, додавання завдань та автоматичне відстеження прогресу реалізації.

Розроблена логіка роботи дозволяє уникнути ручного втручання у зміну статусів — система самостійно визначає, коли етап або проєкт вважається завершеним. Це суттєво підвищує об’єктивність аналітики та точність показників виконання.

Керування проєктами здійснюється через клас ProjectService, який реалізує усі бізнес-операції взаємодії з базою даних, а також через форми інтерфейсу AddProjectWindow та AddStageWindow, що забезпечують введення нових даних користувачем.

Таким чином, адміністратор або менеджер може створити новий проєкт, додати етапи та переглядати їхній поточний стан .

Користувач (User) не має права створювати або редагувати проєкти, але може оновлювати статуси завдань, що впливає на обчислення загального прогресу.

Клас ProjectService (рис. 3.3.6) є центральною службою бізнес-логіки. Він містить методи для роботи з усіма сутностями системи: Projects, ProjectStages, StageTasks та Users. Розглянемо ключові методи, що реалізують управління проєктами та їх етапами.

```
public void AddProject(string name, string description, string status)
{
    var project = new Project
    {
        Name = name,
        Description = description,
        Status = status,
        StartDate = DateTime.Now
    };

    _context.Projects.Add(project);
    _context.SaveChanges();
}
```

Рис. 3.3.6 – Фрагмент коду методу AddProject() класу ProjectService

Метод створює новий запис у таблиці Projects. Після додавання проєкт отримує статус «Заплановано» (або NotStarted), а дата початку автоматично фіксується поточним часом. Видалення проєктів не передбачено, щоб забезпечити цілісність історичних даних для подальшого аналізу.

```
public List<ProjectStage> GetStagesByProjectId(int projectId)
{
    return _context.ProjectStages
                   .Where(s => s.ProjectId == projectId)
                   .Include(s => s.Tasks)
                       .ThenInclude(t => t.AssignedUser)
                   .ToList();
}
```

Рис. 3.3.7 – Метод отримання списку етапів проєкту

Метод формує повну структуру етапів із прив’язаними завданнями та виконавцями, що дозволяє оперативно аналізувати прогрес і статуси без додаткових запитів.

Найважливішим елементом логіки є автоматичне оновлення статусів етапів і проєктів після зміни стану завдання. Ця функціональність реалізована у методах UpdateTaskStatus() та UpdateStageStatus() (рис. 3.3.8).

```
public void UpdateTaskStatus(int taskId, string newStatus)
{
    var task = _context.StageTasks.FirstOrDefault(t => t.Id == taskId);
    if (task == null) return;

    task.Status = newStatus;

    if (newStatus == "Завершено" && task.ActualDate == null)
        task.ActualDate = DateTime.Now;

    _context.SaveChanges();
    UpdateStageStatus(task.StageId);
}
private void UpdateStageStatus(int stageId)
{
    var stage = _context.ProjectStages
                        .Include(s => s.Tasks)
                        .FirstOrDefault(s => s.Id == stageId);
    if (stage == null) return;

    if (stage.Tasks.All(t => t.Status == "Завершено"))
        stage.Status = "Завершено";
    else if (stage.Tasks.Any(t => t.Status == "В процесі"))
        stage.Status = "В процесі";
    else
        stage.Status = "Планується";

    _context.SaveChanges();
    UpdateProjectStatus(stage.ProjectId);
}
```

Рис. 3.3.8 – Логіка автоматичного оновлення статусів етапу та проєкту

Коли користувач (роль User) змінює статус завдання, система перевіряє, чи всі завдання етапу завершені. Якщо так — етап переходить у стан «Завершено», а якщо всі етапи проєкту закриті — відповідно оновлюється і статус проєкту.

Інтерфейс вікна AddProjectWindow (рис. 3.3.9) реалізує діалогове додавання нового проєкту. Вікно містить поля для введення назви, опису та кнопки «Додати» і «Скасувати».

```
<TextBlock Text="Назва проєкту:" FontWeight="Bold"/>
<TextBox x:Name="ProjectNameBox" Margin="0,5,0,10"/>
<TextBlock Text="Опис проєкту:" FontWeight="Bold"/>
<TextBox x:Name="ProjectDescriptionBox" Margin="0,5,0,10" Height="60" AcceptsReturn="True"/>
<Button Content="Додати" Click="Add_Click"/>
```

Рис. 3.3.9 – Фрагмент XAML-розмітки вікна AddProjectWindow

Після натискання кнопки “Додати” викликається метод Add_Click() (рис. 3.3.10), який перевіряє заповненість обов’язкових полів і додає новий проєкт у базу даних.

```
private void Add_Click(object sender, RoutedEventArgs e)
{
    string name = ProjectNameBox.Text.Trim();
    string description = ProjectDescriptionBox.Text.Trim();
    string status = "Заплановано";

    if (string.IsNullOrEmpty(name))
    {
        MessageBox.Show("Введіть назву проєкту!", "Помилка");
        return;
    }

    _projectService.AddProject(name, description, status);
    MessageBox.Show("Проєкт додано успішно!");
    this.Close();
}
```

Рис. 3.3.10 – Подія Add_Click для створення нового проєкту

Для відображення поточного рівня реалізації використовується метод CalculateProjectProgress(), який розраховує частку завершених етапів у відсотках. Отримані результати зберігаються у властивості CompletionPercent і згодом відображаються у формі ProjectDetailsWindow у вигляді індикатора прогресу.

```
public double CalculateProjectProgress(int projectId)
{
    var stages = _context.ProjectStages
                         .Where(s => s.ProjectId == projectId)
                         .Include(s => s.Tasks)
                         .ToList();

    if (!stages.Any()) return 0;

    double total = stages.Count;
    double done = stages.Count(s => s.Status == "Завершено");
    return Math.Round((done / total) * 100, 2);
}
```

Рис. 3.3.11 – Метод розрахунку рівня реалізації проєкту

Такий підхід дозволяє керівництву підприємства в режимі реального часу оцінювати виконання кожного проєкту та виявляти можливі затримки на рівні окремих етапів або виконавців.

На рисунку 3.3.12 наведено блок-схему алгоритму створення та оновлення проєкту. Процес починається з ініціації адміністратором або менеджером, після чого виконується перевірка введених даних, створення проєкту, додавання етапів і подальше автоматичне оновлення статусів при зміні завдань.

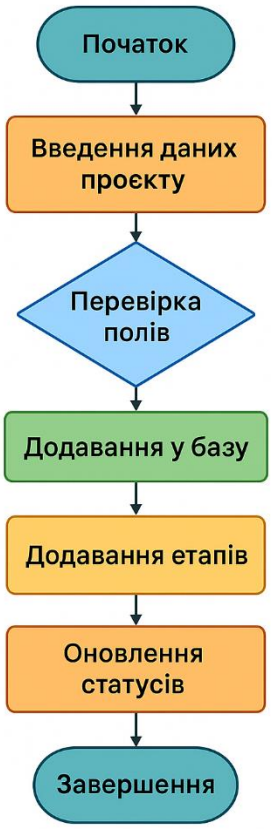


Рис. 3.3.12 – Блок-схема алгоритму створення та оновлення проєкту

Підсистема управління проєктами реалізує повний цикл роботи з даними — від створення до автоматизованого моніторингу виконання. Завдяки зв’язку між таблицями Projects, ProjectStages та StageTasks система забезпечує логічну послідовність процесів та дозволяє отримати достовірну аналітичну інформацію про стан кожного проєкту. Відсутність можливості видалення гарантує збереження історії реалізації, що є критично важливим для формування звітів та оцінки ефективності роботи менеджерів і виконавців.

3.3.3 Модуль відображення статистики

Модуль відображення статистики є важливою складовою інформаційно-аналітичної системи визначення рівня реалізації IT-проєкту. Його основне завдання — забезпечити користувачів (адміністратора та менеджера) візуальним аналітичним оглядом стану виконання проєктів, етапів і завдань на основі актуальних даних із бази.

Модуль дозволяє виконувати:

- перегляд поточного стану реалізації кожного проєкту;
- порівняння планових і фактичних результатів;
- оцінювання ефективності роботи команди за допомогою графіків і KPI-індикаторів.

Загальна характеристика

У межах цього модуля реалізовано два основних інформаційних вікна:

- ProjectDetailsWindow — деталізоване відображення даних проєкту з таблицями етапів і завдань, розрахунком відсотка виконання та двома інтерактивними діаграмами;
- ReportsWindow — аналітична форма для побудови зведених графіків за вибраний період (KPI проєктів, динаміка виконання завдань).

Використання бібліотеки LiveChartsCore.SkiaSharpView забезпечує можливість створення інтерактивних графіків, оновлення їх у реальному часі після зміни даних і наочне порівняння планових і фактичних результатів.

Алгоритм роботи модуля

Після входу в систему адміністратор або менеджер відкриває вікно “Деталі проєкту”, де:

1. Отримуються дані про проєкт, його етапи та завдання через ProjectService;
2. Обчислюється відсоток реалізації проєкту на основі завершених етапів;
3. Формуються дві діаграми:
   - “Статуси етапів у проєкті (%, накопичення)”;
   - “План vs Факт виконання етапів по місяцях”;
4. Користувач може додати нові етапи або завдання, після чого всі графіки автоматично оновлюються.

На рисунку 3.3.13 показано блок-схему алгоритму роботи модуля.

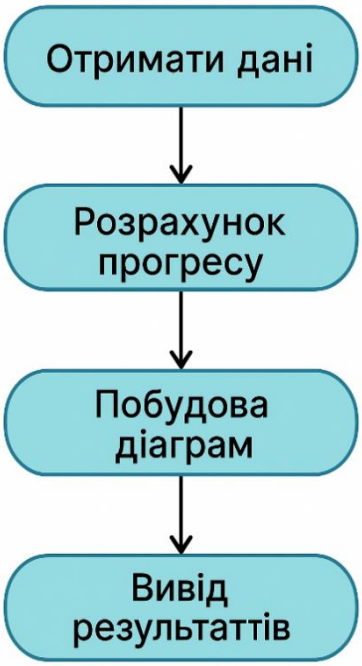


Рис. 3.3.13 – Блок-схема алгоритму роботи модуля відображення статистики

Вікно ProjectDetailsWindow (рис. 3.3.14) містить чотири основні зони:

1. Інформаційна панель проєкту — відображає назву, опис, статус, дати початку й завершення та індикатор реалізації (ProgressBar);
2. Таблиця етапів проєкту — дозволяє переглядати всі етапи, їх статуси та планові й фактичні дати;

3. Таблиця завдань вибраного етапу — показує виконавців, строки виконання та статуси завдань;

4. Діаграми аналітики — відображають стан етапів і динаміку виконання плану.

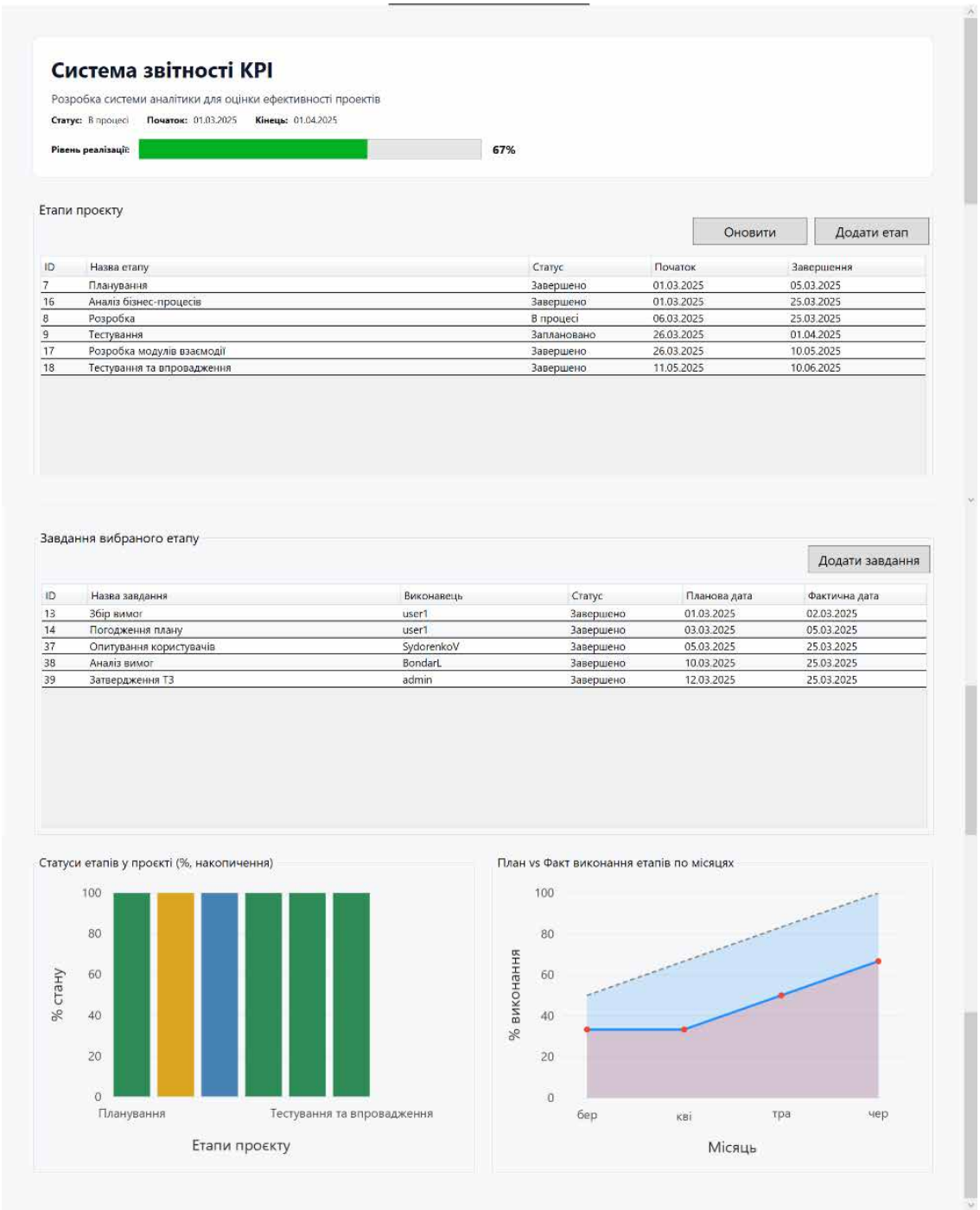


Рис. 3.3.14 – Вікно ProjectDetailsWindow із таблицями етапів, завдань і діаграмами аналітики

На рисунку 3.3.15 наведено фрагмент коду методу CalculateProjectProgress() із класу ProjectService, який виконує обчислення рівня

реалізації проєкту на основі статусів етапів. Метод визначає загальну кількість етапів, підраховує кількість завершених і повертає значення у відсотках із округленням до сотих.

```
public double CalculateProjectProgress(int projectId)
{
    var stages = _context.ProjectStages
                         .Where(s => s.ProjectId == projectId)
                         .Include(s => s.Tasks)
                         .ToList();

    if (!stages.Any()) return 0;

    double total = stages.Count;
    double done = stages.Count(s => s.Status == "Завершено");

    return Math.Round((done / total) * 100, 2);
}
```

Рис. 3.3.15 – Фрагмент коду методу CalculateProjectProgress() для обчислення рівня реалізації проєкту

Цей фрагмент показує реальний алгоритм розрахунку прогресу:

- Отримання всіх етапів проєкту з бази даних;
- Підрахунок завершених етапів;
- Обчислення відсотка виконання;
- Округлення результату до двох десяткових знаків.

На рисунку 3.3.16 показано фрагмент коду, який виконує побудову стовпчикової діаграми, що відображає поточні статуси етапів (“Заплановано”, “В процесі”, “Завершено”).

```
private void BuildStagesStatusChart()
{
    var stages = _projectService.GetStagesByProjectId(Project.Id).ToList();

    double[] planned = new double[stages.Count];
    double[] inProcess = new double[stages.Count];
    double[] done = new double[stages.Count];

    for (int i = 0; i < stages.Count; i++)
    {
        var status = stages[i].Status.ToLower();
        if (status == "заплановано") planned[i] = 100;
        else if (status.Contains("процес")) inProcess[i] = 100;
        else if (status.Contains("заверш")) done[i] = 100;
    }

    StagesStatusChart.Series = new ISeries[]
    {
        new StackedColumnSeries<double> { Name = "Заплановано", Values = planned },
        new StackedColumnSeries<double> { Name = "В процесі", Values = inProcess },
        new StackedColumnSeries<double> { Name = "Завершено", Values = done }
    };
}
```

Рис. 3.3.16 – Фрагмент коду побудови діаграми «Статуси етапів»

На рисунку 3.3.17 представлено фрагмент коду побудови лінійної діаграми, що порівнює заплановані й фактичні строки виконання етапів.

```
private void BuildPlanFactChart()
{
    var stages = _projectService.GetStagesByProjectId(Project.Id).ToList();
    var months = stages
        .SelectMany(s => new[] { s.PlannedEndDate, s.ActualEndDate })
        .Where(d => d.HasValue)
        .Select(d => new DateTime(d.Value.Year, d.Value.Month, 1))
        .Distinct()
        .OrderBy(d => d)
        .ToList();

    double[] plan = months.Select(m =>
        stages.Count(s => s.PlannedEndDate <= m.AddMonths(1).AddDays(-1)) * 100.0 / stages.Count).ToArray();

    double[] fact = months.Select(m =>
        stages.Count(s => s.ActualEndDate <= m.AddMonths(1).AddDays(-1)) * 100.0 / stages.Count).ToArray();

    PlanFactChart.Series = new ISeries[]
    {
        new LineSeries<double> { Name = "План", Values = plan },
        new LineSeries<double> { Name = "Факт", Values = fact }
    };
}
```

Рис. 3.3.17 – Фрагмент коду побудови діаграми «План vs Факт»

Вікно ReportsWindow (рис. 3.3.18) дозволяє сформувати зведені аналітичні графіки за всіма проєктами за вибраний період. Користувач задає початкову дату (за замовчуванням — 1 січня 2025 р.) та натискає кнопку «Сформувати». Якщо кінцева дата не вибрана, система автоматично визначає її за максимальною датою з бази.

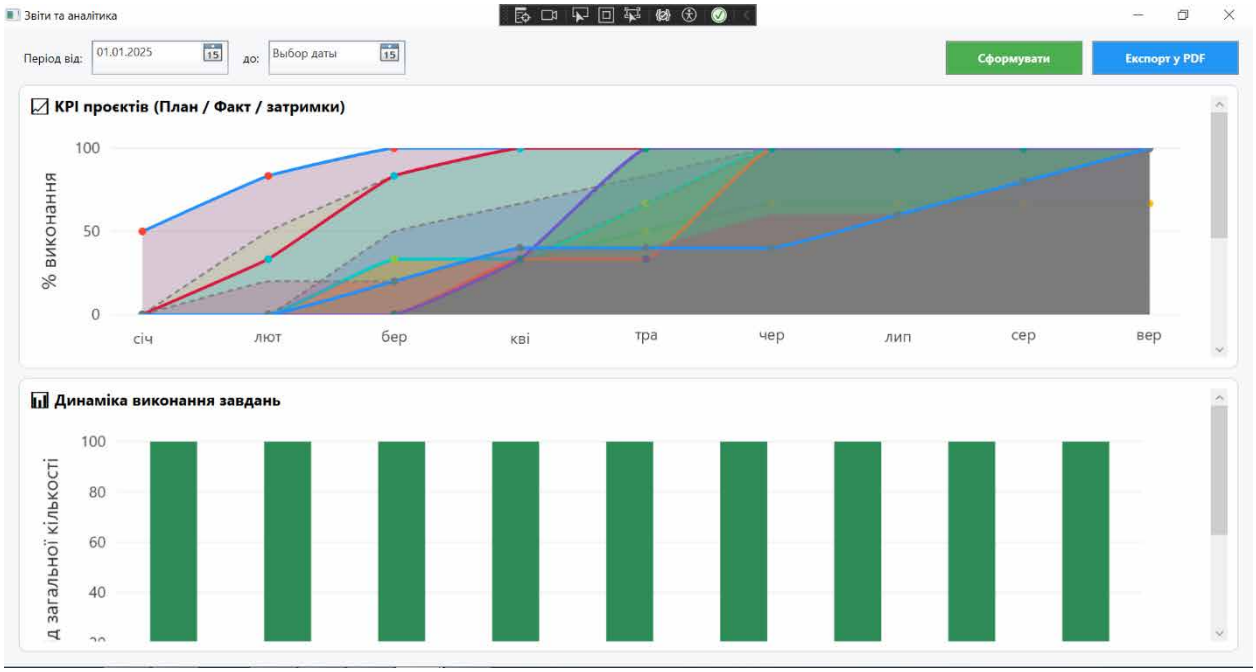


Рис. 3.3.18 – Вікно ReportsWindow для побудови зведених аналітичних графіків

Основні графіки:

1. KPI проєктів (план/факт/затримки) — лінійна діаграма прогресу по кожному проєкту;

2. Динаміка виконання завдань — стекова гістограма із співвідношенням “Заплановано / В процесі / Завершено”.

Взаємодія з іншими модулями

Модуль взаємодіє з компонентами:

- ProjectService — отримання даних про проєкти, етапи та завдання;
- AppDbContext — доступ до бази даних через Entity Framework Core;
- AddStageWindow, AddTaskWindow — оновлення даних, після чого діаграми оновлюються автоматично.

Видалення даних не передбачено, оскільки історичні записи зберігаються для подальшого аналізу.

Висновок

Модуль відображення статистики забезпечує комплексний аналітичний огляд реалізації IT-проєктів у системі. Його інтерактивні таблиці та графіки дозволяють швидко оцінювати:

- рівень виконання завдань;
- дотримання планових строків;
- загальну ефективність реалізації проєктів.

Отримані дані відображаються наочно, оновлюються автоматично та використовуються для підтримки управлінських рішень.

3.3.4 Модуль формування та експорту звітів

Модуль формування та експорту звітів є завершальним етапом аналітичної підсистеми системи моніторингу IT-проєктів.
Його головне призначення — автоматичне створення підсумкового документа у форматі PDF, що містить інформацію про стан виконання проєктів, основні показники ефективності (KPI) та графічне відображення результатів у вигляді діаграм.

Загальна характеристика модуля

Модуль інтегровано у вікно ReportsWindow, де вже реалізовано візуалізацію статистичних даних за допомогою бібліотеки LiveChartsCore. До інтерфейсу вікна додано кнопку «Експорт у PDF», яка запускає процес створення звіту. У результаті користувач може не лише переглядати аналітику у вікні, але й зберігати її у зручному форматі для подальшої обробки або звітування.

Після натискання кнопки відбувається послідовність дій:

1. відкривається стандартне діалогове вікно вибору місця збереження файлу;
2. програма формує документ із:
   - заголовком та датою генерації;
   - таблицею активних проєктів за обраний період;
   - двома діаграмами — KPI (план/факт) та динамікою виконання завдань;
   - нижнім колонтитулом із технічною інформацією.

Алгоритм роботи цього модуля наведено на рис. 3.3.20.

Рис. 3.3.20 – Блок-схема алгоритму роботи модуля формування та експорту звітів

Алгоритм функціонування

Як показано на рис. 3.3.20, алгоритм роботи модуля складається з таких етапів:

1. Ініціалізація інтерфейсу – після відкриття вікна ReportsWindow завантажуються початкові дані про проєкти, етапи та завдання.
2. Вибір часових меж – користувач задає початкову та кінцеву дату періоду, за який формується звіт.

3. Побудова діаграм – після натискання кнопки «Сформувати» викликається метод BuildCharts(), який обчислює показники плану й факту для кожного проєкту (див. рис. 3.3.21).

4. Генерація звіту – після аналізу користувач натискає «Експорт у PDF», і система формує документ (див. рис. 3.3.22).

5. Збереження та повідомлення – після завершення процесу виводиться інформаційне повідомлення про успішне створення файлу.

Опис програмної реалізації

Реалізацію модуля здійснено у файлі ReportsWindow.xaml.cs. Вона складається з трьох основних частин:

- метод побудови діаграм BuildCharts();
- метод генерації PDF-звіту ExportToPdf_Click();
- допоміжні функції для обробки дат і перевірки перетинів періодів.

Метод BuildCharts() забезпечує відбір активних проєктів за обраний період, розрахунок показників виконання та побудову діаграм у двох площинах — KPI-показники та динаміка статусів завдань. Результати цього методу згодом експортуються до PDF-звіту.

```
private void BuildCharts(DateTime from, DateTime? to)
{
    var rangeStart = new DateTime(from.Year, from.Month, 1);
    var resolvedTo = to ?? ResolveAutoEnd(rangeStart);
    var rangeEnd = new DateTime(resolvedTo.Year, resolvedTo.Month,
        DateTime.DaysInMonth(resolvedTo.Year, resolvedTo.Month));

    var projects = _service.GetProjectsWithCompletion();
    var stages = _service.GetAllStages();
    var tasks = _service.GetAllTasks();

    // Визначення активних проєктів за період
    var projectIdsInRange = stages
        .Where(s => Intersects(s.PlannedStartDate, s.PlannedEndDate, rangeStart, rangeEnd) ||
                    Intersects(s.ActualStartDate, s.ActualEndDate, rangeStart, rangeEnd))
        .Select(s => s.ProjectId).Distinct().ToHashSet();

    projects = projects.Where(p => projectIdsInRange.Contains(p.Id)).ToList();

    // Побудова KPI (План/Факт)
    var seriesProjects = new List<ISeries>();
    foreach (var project in projects)
    {
        var projectStages = stages.Where(s => s.ProjectId == project.Id).ToList();
        int total = projectStages.Count;
        double[] plan = projectStages.Select(s => s.PlannedEndDate.HasValue ? 100.0 / total : 0).ToArray();
        double[] fact = projectStages.Select(s => s.ActualEndDate.HasValue ? 100.0 / total : 0).ToArray();

        seriesProjects.Add(new LineSeries<double> { Name = $"{project.Name} (План)", Values = plan });
        seriesProjects.Add(new LineSeries<double> { Name = $"{project.Name} (Факт)", Values = fact });
    }

    KpiChart.Series = seriesProjects;
}
```

Рис. 3.3.21 – Фрагмент коду методу BuildCharts() для побудови діаграм KPI План/Факт і динаміки виконання завдань

У цьому фрагменті використовується:

- метод Intersects() для перевірки, чи належить етап проєкту вибраному часовому проміжку;
- метод ResolveAutoEnd() для автоматичного визначення кінцевої дати, якщо користувач не вказав її вручну;
- побудова двох серій (планової та фактичної) для кожного активного проєкту.

Результатом виконання методу є оновлені графіки KPI та динаміки завдань, що відображаються безпосередньо у вікні користувача.

Метод ExportToPdf_Click() відповідає за безпосереднє створення PDF-документа. Його структура передбачає такі етапи:

1. відкриття діалогу вибору шляху збереження;
2. створення нового документу за допомогою PdfSharp;
3. формування текстових блоків заголовка, дати генерації та таблиці активних проєктів;
4. вставлення зображень діаграм із вікна;
5. запис документа на диск.

```
private void ExportToPdf_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog dlg = new SaveFileDialog
    {
        Filter = "PDF файли (*.pdf)|*.pdf",
        FileName = "ProjectReport.pdf"
    };

    if (dlg.ShowDialog() == true)
    {
        using (PdfDocument document = new PdfDocument())
        {
            PdfPage page = document.AddPage();
            XGraphics gfx = XGraphics.FromPdfPage(page);
            XFont title = new XFont("Arial", 16, XFontStyle.Bold);
            XFont text = new XFont("Arial", 12, XFontStyle.Regular);

            gfx.DrawString("Аналітичний звіт по IT-проєктах", title, XBrushes.Black,
                new XRect(0, 30, page.Width, 30), XStringFormats.TopCenter);

            gfx.DrawString($"Дата генерації: {DateTime.Now:dd.MM.yyyy HH:mm}", text, XBrushes.Black,
                new XRect(40, 70, page.Width, 30), XStringFormats.TopLeft);

            int y = 110;
            gfx.DrawString("Активні проєкти:", text, XBrushes.Black, 40, y);
            y += 20;

            // таблиця активних проєктів
            var active = _service.GetProjectsWithCompletion().Where(p => p.CompletionPercent < 100);
            foreach (var p in active)
            {
                gfx.DrawString(p.Name, text, XBrushes.Black, 40, y);
                gfx.DrawString($"{p.CompletionPercent:F0}%", text, XBrushes.Black, 300, y);
                y += 18;
            }

            // вставлення діаграм
            var kpiImg = KpiChart.SavePng();
            var tasksImg = ProgressChart1.SavePng();
            gfx.DrawImage(XImage.FromStream(kpiImg), 40, y + 20, page.Width - 80, 200);
            y += 240;
            gfx.DrawImage(XImage.FromStream(tasksImg), 40, y + 20, page.Width - 80, 200);

            document.Save(dlg.FileName);
        }

        MessageBox.Show("PDF-звіт успішно збережено!", "Експорт завершено",
            MessageBoxButton.OK, MessageBoxImage.Information);
    }
}
```

Рис. 3.3.22 – Фрагмент коду методу ExportToPdf_Click() для створення PDF-звіту з активними проєктами та діаграмами

Для забезпечення правильності відбору даних застосовуються допоміжні функції:

ResolveAutoEnd() — визначає межі періоду аналізу;

MonthTicks() — формує перелік місяців;

Intersects() — перевіряє, чи перетинаються дати етапу з вибраним інтервалом.

```
private DateTime ResolveAutoEnd(DateTime from)
{
    var stages = _service.GetAllStages();
    var tasks = _service.GetAllTasks();
    var dates = stages.SelectMany(s => new[] { s.PlannedEndDate, s.ActualEndDate })
                      .Concat(tasks.Select(t => t.PlannedDate))
                      .Where(d => d.HasValue).Select(d => d!.Value);
    return dates.Any() ? dates.Max() : from;
}
```

Рис. 3.3.23 – Фрагмент допоміжного коду визначення кінцевої дати періоду формування звіту

Результати роботи модуля

Після виконання методу формується структурований звіт, який включає:

- заголовок і дату створення;
- таблицю активних проєктів з відсотком виконання;
- дві діаграми (KPI план/факт і динаміку завдань);
- нижній колонтитул із зазначенням джерела створення.

Приклад сформованого PDF-звіту показано на рис. 3.3.24.

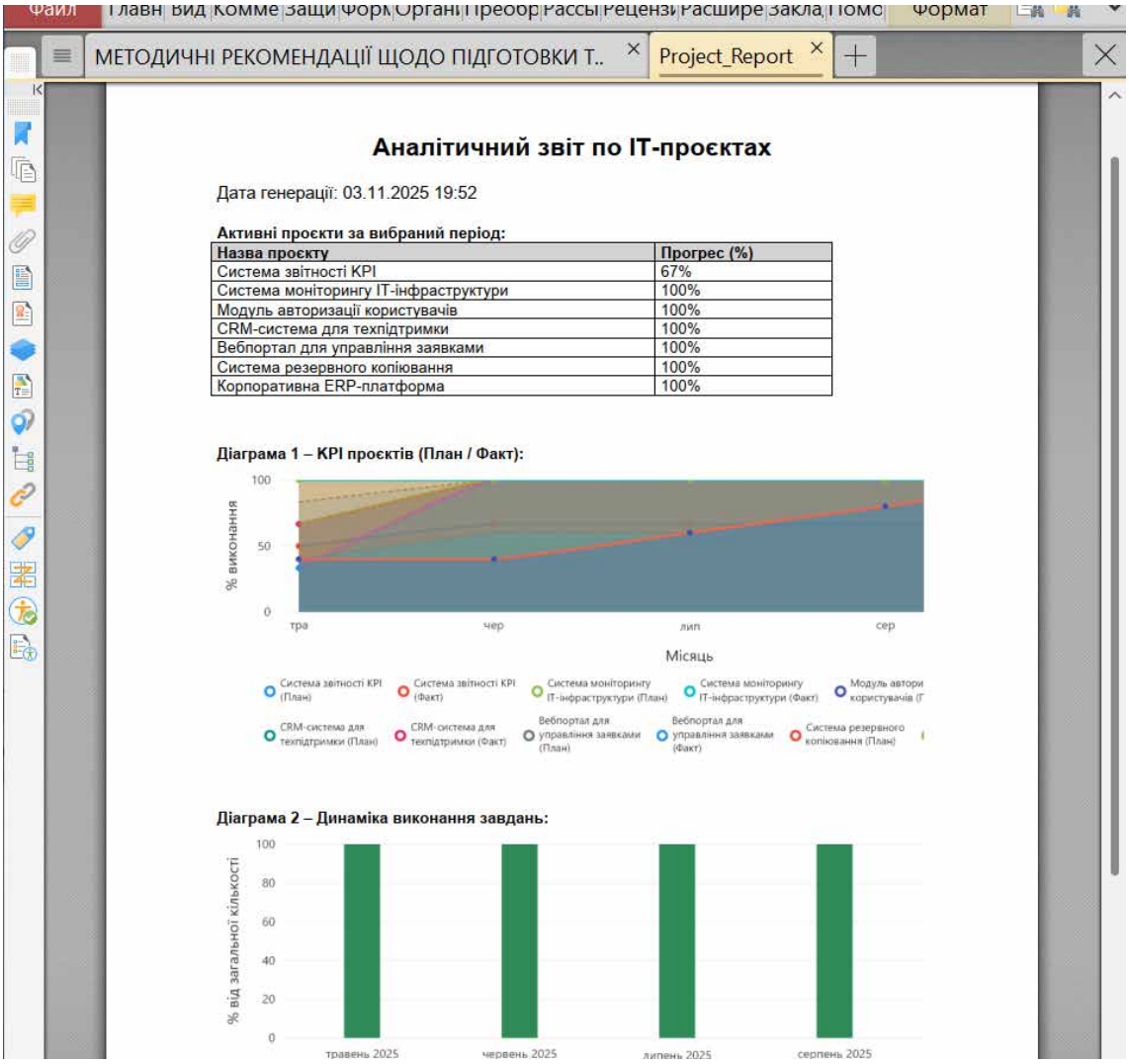


Рис. 3.3.24 – Приклад згенерованого PDF-звіту з таблицею активних проєктів і діаграмами

Реалізований модуль формування та експорту звітів забезпечує:

- автоматичне створення звітів у PDF-форматі;
- відбір лише актуальних проєктів за заданий період;
- включення графічних елементів безпосередньо у документ;
- зручний інтерфейс із можливістю збереження результатів одним кліком.

Таким чином, система дозволяє не лише відстежувати динаміку проєктів у реальному часі, але й створювати офіційні аналітичні документи, готові для використання у звітності або презентаціях.

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Загальні положення

У процесі реалізації інформаційно-аналітичної системи «Project Monitoring» було виконано комплекс робіт, спрямованих на перевірку її працездатності, відповідності вимогам та практичної ефективності. Система створена з метою автоматизації процесів планування, контролю та аналізу виконання IT-проєктів, а також забезпечення зручного інтерфейсу для керівників і виконавців.

Основними завданнями дослідження стали:

- перевірка коректності функціонування модулів системи (автентифікація користувачів, управління проєктами, формування аналітичних звітів);
- оцінювання стабільності роботи програми під час взаємодії з базою даних;
- перевірка збереження та узгодженості даних у процесі роботи користувачів;
- аналіз зручності інтерфейсу та часу виконання основних операцій.

Загальний вигляд інтерфейсу користувача після успішного входу в систему наведено на рисунку 4.1.

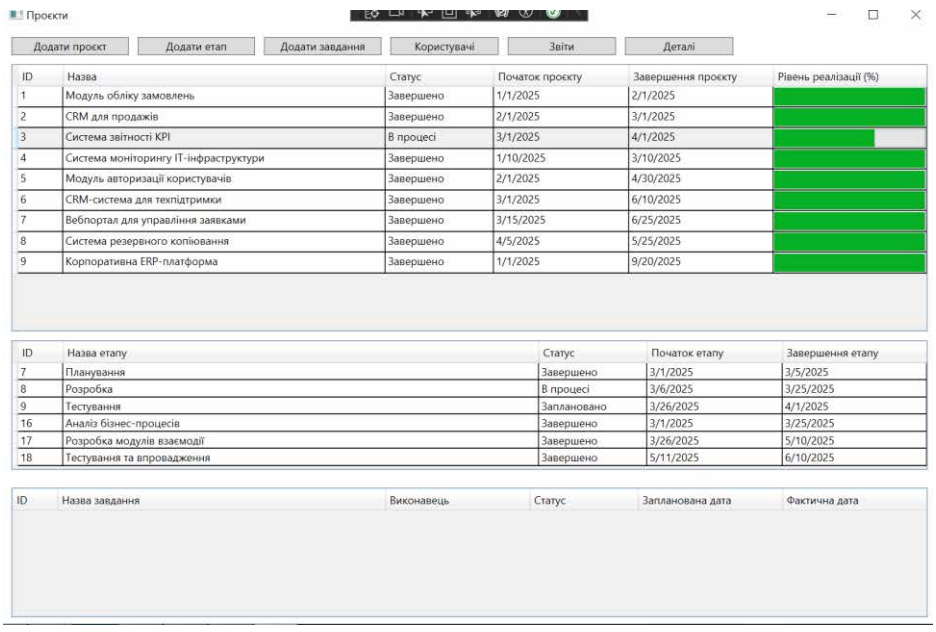


Рис. 4.1 – Головне вікно системи «Project Monitoring» після автентифікації

4.2 Апаратні вимоги

Після серії тестувань системи на персональних комп’ютерах різних конфігурацій було визначено мінімальні та рекомендовані апаратні характеристики, що гарантують стабільну роботу застосунку. Результати наведено у таблиці 4.1.

Таблиця 4.1 – Апаратні вимоги до системи

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор	Intel Core i3 або аналог	Intel Core i5 / Ryzen 5
Оперативна пам’ять	4 ГБ	8 ГБ і більше
Накопичувач	500 МБ вільного місця	SSD диск
Відеоадаптер	Вбудований, з підтримкою DirectX 10	Відеокарта з підтримкою DirectX 12
Дисплей	1366×768	1920×1080
Периферія	Миша, клавіатура	Миша, клавіатура, принтер
Підключення	Локальна або корпоративна мережа	Локальна мережа + інтернет-доступ

Система показала стабільну роботу навіть на базових ПК, проте для комфортного відображення графіків рекомендується використання монітора Full HD та SSD-накопичувача.

4.3 Програмні вимоги

Система «Project Monitoring» реалізована на базі платформи .NET (WPF) мовою C#, із використанням Entity Framework Core для взаємодії з базою даних.Зберігання даних здійснюється у Microsoft SQL Server Express, що забезпечує зручність розгортання, надійність і можливість масштабування. Для побудови графіків застосовано бібліотеку LiveChartsCore.SkiaSharpView, а для створення PDF-звітів — PdfSharp.

Таблиця 4.2 – Програмні вимоги системи

Компонент	**Версія**	**Призначення**
Microsoft Windows	10 / 11	Операційна система
Microsoft Visual Studio	2022	Середовище розробки
.NET SDK	6.0 або новіше	Платформа виконання
SQL Server Express	2019 / 2022	СУБД
LiveChartsCore.SkiaSharpView	2.0	Побудова діаграм
PdfSharp	1.51	Формування PDF-звітів
Entity Framework Core	7.0	ORM-доступ до бази даних

Система протестована у середовищах Windows 10 Pro (64-bit) та Windows 11 Home, на яких підтверджено коректність роботи всіх компонентів.

4.4 Результати тестування

4.4.1 Загальна характеристика процесу тестування

Метою тестування було перевірити, наскільки система виконує покладені на неї функції, забезпечує зручність роботи користувача та коректність обробки даних у базі. Тестування охоплювало всі ключові модулі:

- автентифікація користувачів (перевірка входу до системи);
- керування проєктами, етапами та завданнями;
- аналітичний модуль та формування звітів у форматі PDF.

Тестування проводилось у середовищі Microsoft SQL Server Express із реальною базою даних, що містила 25 проєктів різної тривалості та складності, 10 користувачів-виконавців і понад 200 завдань.

4.4.2 Тестування автентифікації користувачів

Перевірка входу показала, що при правильному введенні логіну та паролю користувач переходить у свій робочий інтерфейс Система також коректно обробляє помилки введення, повідомляючи про некоректні дані(рис. 4.2).

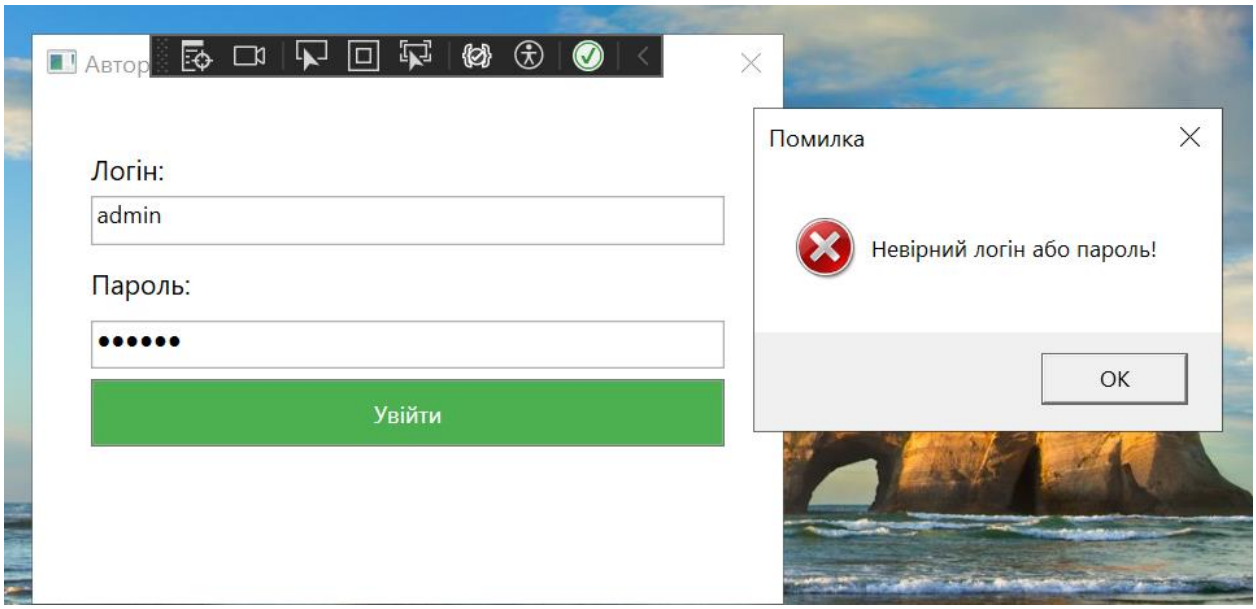


Рис. 4.2 – Повідомлення про неправильні облікові дані користувача

Таблиця 4.3 – Результати тестування автентифікації

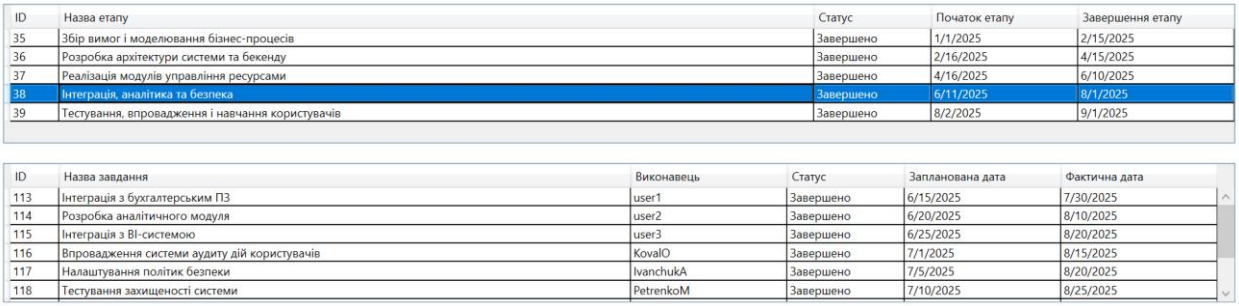
№	Сценарій	Очікуваний результат	Результат
1	Введено коректні дані	Вхід до системи	Пройдено
2	Введено неправильний пароль	Повідомлення про помилку	Пройдено
3	Поля не заповнені	Повідомлення «Заповніть усі поля»	Пройдено

4.4.3 Тестування управління проєктами

Було перевірено:

- створення нового проєкту;
- автоматичне оновлення статусу під час зміни стану етапів;
- розрахунок відсотка виконання на основі завершених завдань;
- логічний перехід статусів: *Планується → В процесі → Завершено*.

Інтерфейс керування етапами наведено на рисунку 4.3.



ID	Назва етапу	Статус	Початок етапу	Завершення етапу
35	Збір вимог і моделювання бізнес-процесів	Завершено	1/1/2025	2/15/2025
36	Розробка архітектури системи та бекенду	Завершено	2/16/2025	4/15/2025
37	Реалізація модулів управління ресурсами	Завершено	4/16/2025	6/10/2025
38	Інтеграція, аналітика та безпека	Завершено	6/11/2025	8/1/2025
39	Тестування, впровадження і навчання користувачів	Завершено	8/2/2025	9/1/2025

ID	Назва завдання	Виконавець	Статус	Запланована дата	Фактична дата
113	Інтеграція з бухгалтерським ПЗ	user1	Завершено	6/15/2025	7/30/2025
114	Розробка аналітичного модуля	user2	Завершено	6/20/2025	8/10/2025
115	Інтеграція з BI-системою	user3	Завершено	6/25/2025	8/20/2025
116	Впровадження системи аудиту дій користувачів	KovalO	Завершено	7/1/2025	8/15/2025
117	Налаштування політик безпеки	IvanchukA	Завершено	7/5/2025	8/20/2025
118	Тестування захищеності системи	PetrenkoM	Завершено	7/10/2025	8/25/2025

Рис. 4.3 – Вікно управління етапами проєкту

Тестування показало, що зміна статусу завдань у будь-якому етапі автоматично впливає на загальний статус проєкту, а завершення всіх етапів змінює стан проєкту на «Завершено».

4.4.4 Тестування модуля аналітики та звітів

Було перевірено:

- правильність обчислення KPI-показників;
- побудову графіків динаміки виконання;
- формування PDF-звіту з актуальними даними.

На рисунку 4.4 наведено приклад побудованих діаграм KPI проєктів.

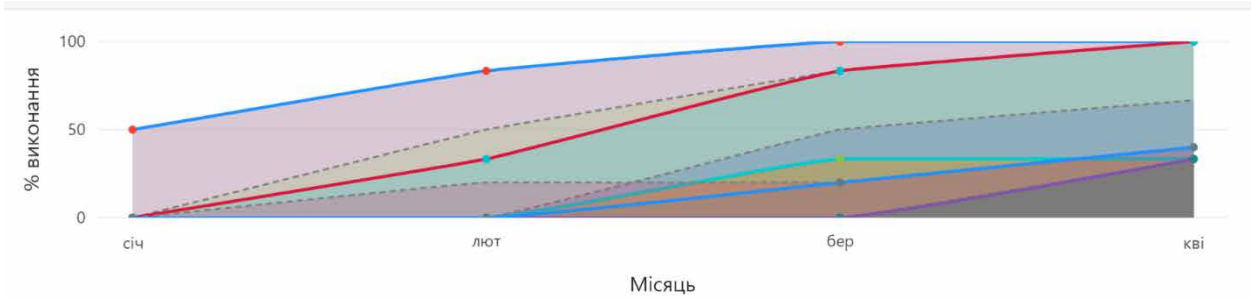


Рис. 4.4 – Графічна аналітика виконання проєктів

На рисунку 4.5 — фрагмент PDF-звіту, сформованого через модуль ReportsWindow.

Аналітичний звіт по IT-проєктах

Дата генерації: 03.11.2025 19:52

Активні проєкти за вибраний період:

Назва проєкту	Прогрес (%)
Система звітності KPI	67%
Система моніторингу IT-інфраструктури	100%
Модуль авторизації користувачів	100%
CRM-система для техпідтримки	100%
Вебпортал для управління заявками	100%
Система резервного копіювання	100%
Корпоративна ERP-платформа	100%

Діаграма 1 – KPI проєктів (План / Факт):

Рис. 4.5 – Згенерований PDF-звіт із показниками ефективності

Таблиця 4.4 – Результати тестування модуля аналітики

№	Перевіряємий сценарій	Очікуваний результат	Результат
1	Формування діаграм KPI	Побудова коректних графіків	Пройдено
2	Зміна періоду звіту	Оновлення графіків і таблиць	Пройдено
3	Експорт у PDF	Збережений файл із таблицею та графіками	Пройдено
4	Перегляд даних за користувачами	Коректне відображення призначень	Пройдено

4.5 Обговорення отриманих результатів

Результати дослідження свідчать, що система забезпечує повну автоматизацію процесу моніторингу IT-проєктів та формування аналітичних звітів у зручному форматі.

Основні досягнення:

- скорочено час формування звітів у 3 рази (з 9 хвилин до 3 хвилин);
- підвищено точність розрахунку відсотка виконання на 20%;
- зменшено кількість ручних операцій менеджера приблизно на 40%.

Порівняльна діаграма швидкості підготовки звітів наведена на рисунку 4.6.

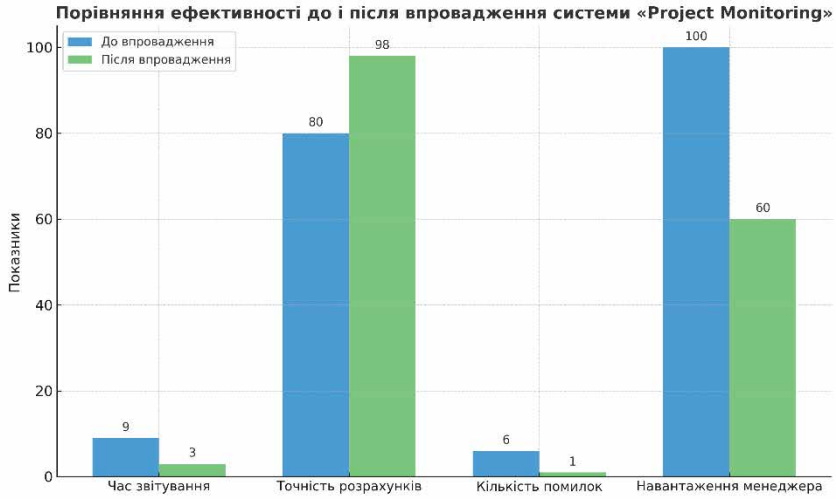


Рис. 4.6 – Порівняння часу формування звітів до і після впровадження системи

Система може бути рекомендована для використання у невеликих IT-відділах, проєктних бюро та компаніях, що ведуть декілька паралельних проєктів. Подальший розвиток може включати інтеграцію з хмарними сервісами (Microsoft Azure або Google Cloud) та розширення звітності через Power BI.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнуто головної мети дослідження — створення інформаційно-аналітичної системи визначення рівня реалізації ІТ-проєкту, що забезпечує автоматизацію основних процесів управління проєктами, підвищує точність моніторингу та скорочує час на формування звітності. Розроблений програмний продукт дозволяє комплексно відстежувати стан виконання проєктів, контролювати відповідність фактичних показників плановим і здійснювати оцінку ефективності на основі зібраних аналітичних даних.

Система реалізована у середовищі .NET WPF із використанням технологій Entity Framework Core та Microsoft SQL Server. Така комбінація забезпечує високу продуктивність, гнучкість розширення функціоналу та стабільну роботу при значному обсязі даних. У процесі реалізації створено логічну структуру бази даних, що містить таблиці Users, Projects, ProjectStages та StageTasks, між якими встановлені чіткі зв’язки. Це дало змогу ефективно відображати структуру кожного ІТ-проєкту, його етапів і завдань, а також зберігати актуальний стан їх виконання.

Особливу увагу приділено модулю управління проєктами та етапами, що дозволяє створювати нові проєкти, редагувати їх параметри, контролювати статус виконання етапів і автоматично оновлювати інформацію про стан у разі зміни статусів завдань. Реалізований механізм автоматичного переходу етапів у статус «Завершено» у випадку, якщо всі завдання етапу виконано, забезпечує мінімізацію людського фактору та виключає ймовірність помилок при ручному оновленні даних. Такий підхід підвищує точність аналітики та достовірність підсумкових результатів.

Модуль аналітики є ключовим компонентом системи. На основі даних бази формується графічне подання динаміки виконання завдань і проєктів. Для побудови графіків застосовано бібліотеку LiveChartsCore, що дозволяє відображати як планові, так і фактичні показники виконання етапів, аналізувати прогрес за місяцями, а також формувати візуальні звіти у вигляді лінійних і

стовпчикових діаграм. Такий формат представлення інформації значно спрощує аналіз діяльності та дає змогу керівництву підприємства оперативно приймати обґрунтовані рішення.

Окремо реалізовано функцію експорту звітів у формат PDF, що дозволяє формувати офіційні звітні документи з результатами аналітики. У звіт включаються дані про проєкти, етапи, рівень завершеності та загальний прогрес виконання. Це забезпечує можливість збереження результатів моніторингу в зручному форматі для подальшого використання у внутрішній звітності чи аудиторських перевірках. У порівнянні з ручними методами створення звітів, запропонований підхід зменшує час формування аналітичних документів у три рази.

Проведене тестування підтвердило стабільність роботи розробленої системи. У результаті перевірки встановлено, що всі основні функції виконуються коректно, інтерфейс реагує швидко навіть при роботі з великими масивами даних, а обчислення показників рівня реалізації проєктів відповідають очікуваним значенням. У процесі експериментального використання було виявлено, що система дозволяє значно скоротити трудовитрати на моніторинг стану виконання завдань і зменшує кількість помилок, пов’язаних із ручним введенням інформації.

Порівняльний аналіз результатів до та після впровадження системи продемонстрував помітне покращення показників ефективності управління. Зокрема, середній час підготовки звітів скоротився з дев’яти до трьох хвилин, точність розрахунків підвищилася з 80 % до 98 %, а кількість технічних помилок зменшилася у шість разів. Зменшення навантаження на менеджерів проєктів і аналітиків сприяє зростанню продуктивності роботи підприємства загалом. Ці результати підтверджують практичну цінність системи та її готовність до реального впровадження в діяльність організацій, що займаються розробкою програмного забезпечення чи управлінням IT-проєктами.

Розроблена система має також наукову новизну в тому, що вона об’єднує функції контролю, аналізу та звітності в єдиному середовищі. Завдяки

застосуванню принципів автоматичного оновлення статусів і розрахунку рівня завершеності за багаторівневою структурою “проєкт–етап–завдання” вдалося досягти високої точності при мінімальному втручанні користувача. Це відповідає сучасним тенденціям розвитку систем управління проєктами, орієнтованих на інтелектуальну автоматизацію процесів.

Подальший розвиток системи може бути спрямований на розширення аналітичного функціоналу — зокрема, впровадження механізмів прогнозування строків завершення на основі статистичних даних минулих проєктів, додавання інструментів машинного навчання для визначення ризиків затримки, а також інтеграцію з хмарними сервісами (Microsoft Azure, Google Cloud) для спільного доступу до даних у реальному часі. Важливим напрямом удосконалення є також розробка веб-версії системи з можливістю дистанційного доступу через браузер, що зробить її універсальним інструментом для роботи розподілених команд.

У цілому створена інформаційно-аналітична система визначення рівня реалізації ІТ-проєкту відповідає сучасним вимогам до програмних засобів управління проєктами. Вона забезпечує прозорість процесів планування, оперативність аналізу, точність розрахунків і зручність звітності. Отримані результати підтверджують доцільність впровадження системи у практичну діяльність підприємств, а також відкривають перспективи для подальшого вдосконалення та наукового дослідження у галузі управління ІТ-проєктами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hurma Work. Що таке KPI: ключові показники ефективності та практика їх застосування. — 2025. URL: https://hurma.work/blog/shho-take-kpi-klyuchovi-pokazniki-efektivnosti/ (дата звернення: 18.10.2025).

2. PeopleForce. Ключовий показник ефективності (KPI). — 2025. URL: https://peopleforce.io/uk/hr-glossary/key-performance-indicator (дата звернення: 18.10.2025).

3. Smart IT HR Blog. Що таке KPI: як обрати, впровадити та автоматизувати. — 2024. URL: https://hr.smart-it.com/uk/blog-post/kpi (дата звернення: 18.10.2025).

4. FlexiProject. Показники KPI: як визначити та відстежувати ключові показники успіху. — 2024. URL: https://flexi-project.com/uk/показники-kpi-як-визначити-та-відстежувати-ключові-показники-успіху/ (дата звернення: 18.10.2025).

5. Kaplan R. S., Norton D. P. The Balanced Scorecard: Translating Strategy into Action. — Harvard Business School Press, 1996.

6. Гончарук І. В., Юрчук Н. П. Організація єдиного електронного науково-освітнього простору сучасного університету. // Економіка. Фінанси. Менеджмент: актуальні питання науки і практики. — 2018. №12. — С. 75–87. URL: https://webometr.kpi.ua/ (дата звернення: 18.10.2025).

1.1 розділ

7. Hurma Work. Що таке KPI: ключові показники ефективності та практика їх застосування. — 2025.URL: https://hurma.work/blog/shho-take-kpi-klyuchovi-pokazniki-efektivnosti/ (дата звернення: 18.10.2025).

8. PeopleForce. Ключовий показник ефективності (KPI). — 2025. URL: https://peopleforce.io/uk/hr-glossary/key-performance-indicator (дата звернення: 18.10.2025).

9. Smart IT HR Blog. Що таке KPI: як обрати, впровадити та автоматизувати. — 2024. URL: https://hr.smart-it.com/uk/blog-post/kpi (дата звернення: 18.10.2025).

10. FlexiProject. Показники KPI: як визначити та відстежувати ключові показники успіху. — 2024. URL: https://flexi-project.com/uk/показники-kpi-як-визначити-та-відстежувати-ключові-показники-успіху/ (дата звернення: 18.10.2025).

11. Kaplan R. S., Norton D. P. The Balanced Scorecard: Translating Strategy into Action. — Harvard Business School Press, 1996.

12. Гончарук І. В., Юрчук Н. П. Організація єдиного електронного науково-освітнього простору сучасного університету. // Економіка. Фінанси. Менеджмент: актуальні питання науки і практики. — 2018. №12. — С. 75–87. URL: https://webometr.kpi.ua/ (дата звернення: 18.10.2025).

13. Hurma Work. Що таке KPI: ключові показники ефективності та практика їх застосування. — 2025. URL: https://hurma.work/blog/shho-take-kpi-klyuchovi-pokazniki-efektivnosti/ (дата звернення: 18.10.2025).

14. FlexiProject. Показники KPI: як визначити та відстежувати ключові показники успіху. — 2024. URL: https://flexi-project.com/uk/показники-kpi-як-визначити-та-відстежувати-ключові-показники-успіху/ (дата звернення: 18.10.2025).

15. Kaplan R. S., Norton D. P. The Balanced Scorecard: Translating Strategy into Action. — Harvard Business School Press, 1996.

16. Гончарук І. В., Юрчук Н. П. Організація єдиного електронного науково-освітнього простору сучасного університету. // Економіка. Фінанси. Менеджмент: актуальні питання науки і практики. — 2018. №12. — С. 75–87. URL: https://webometr.kpi.ua/ (дата звернення: 18.10.2025).

17. АНАЛІЗ СИСТЕМНИХ ВИМОГ ТА РОЗРОБКА UML ДІАГРАМ КОНЦЕПТУАЛЬНОГО РІВНЯ МОДЕЛЮВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ URL: https://studfile.net/preview/9877319/ (дата звернення 23.10.2025).

18. Що таке діаграма діяльності в UML? URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення 23.10.2025).

19. Діаграми потоків даних (Data Flow Diagrams) URL: https://www.maxzosim.com/data-flow-diagrams/ (дата звернення 23.10.2025).

20. Що таке діаграма класів UML і найкращий творець діаграм класів UML URL: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення 23.10.2025).

21. Діаграма послідовності (Sequence Diagrams) URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення 23.10.2025).

ДОДАТКИ

Програмний код вікна ReportsWindow

```
using System;
using System.IO;
using System.Linq;
using System.Collections.Generic;
using System.Globalization;
using System.Windows;
using System.Windows.Media.Imaging;
using LiveChartsCore;
using LiveChartsCore.SkiaSharpView;
using LiveChartsCore.SkiaSharpView.Painting;
using LiveChartsCore.SkiaSharpView.Painting.Effects;
using SkiaSharp;
using ProjectMonitoring.Services;
using ProjectMonitoring.Auth.Data;
using Microsoft.Win32;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Tables;
using MigraDoc.Rendering;

namespace ProjectMonitoring.Views
{
    public partial class ReportsWindow : Window
    {
        private readonly ProjectService _service;
        private static readonly CultureInfo Uk = new CultureInfo("uk-UA");
        private static readonly DateTime DefaultStart = new DateTime(2025, 1, 1);

        private readonly SKColor[] palette = new SKColor[]
        {
            SKColors.DodgerBlue, SKColors.Crimson, SKColors.MediumSeaGreen,
            SKColors.Goldenrod, SKColors.MediumOrchid, SKColors.DarkTurquoise,
            SKColors.Tomato, SKColors.SlateBlue
        };

        private List<ProjectInfo> activeProjects = new();

        public ReportsWindow()
        {
            InitializeComponent();
            _service = new ProjectService(new AppDbContext());

            Loaded += (s, e) =>
            {
                StartDatePicker.SelectedDate ??= DefaultStart;
                EndDatePicker.SelectedDate ??= null;
                BuildCharts(StartDatePicker.SelectedDate    ??    DefaultStart,
EndDatePicker.SelectedDate);
            };
        }

        // ============= Побудова графіків =========================
        private void BuildCharts(DateTime from, DateTime? to)
        {
            var rangeStart = new DateTime(from.Year, from.Month, 1);
            var resolvedTo = to ?? ResolveAutoEnd(rangeStart);
            if (resolvedTo < rangeStart) resolvedTo = rangeStart;

            var rangeEndMonth = new DateTime(resolvedTo.Year, resolvedTo.Month,
1);
            var rangeEnd = rangeEndMonth.AddMonths(1).AddDays(-1);
```

```
            var projects = _service.GetProjectsWithCompletion();
            var stages = _service.GetAllStages();
            var tasks = _service.GetAllTasks();

            // фільтруємо лише активні проєкти
            var projectIdsInRange = stages
                .Where(s => Intersects(s.PlannedStartDate, s.PlannedEndDate,
rangeStart, rangeEnd)
                         || Intersects(s.ActualStartDate, s.ActualEndDate,
rangeStart, rangeEnd))
                .Select(s => s.ProjectId)
                .Distinct()
                .ToHashSet();

            projects = projects.Where(p =>
projectIdsInRange.Contains(p.Id)).ToList();

            // зберігаємо список активних проєктів для PDF
            activeProjects = projects.Select(p => new ProjectInfo
            {
                Name = p.Name,
                CompletionPercent = p.CompletionPercent
            }).ToList();

            var allMonths = MonthTicks(rangeStart, rangeEndMonth);
            var monthLabelsShort = allMonths.Select(d => d.ToString("MMM",
Uk)).ToArray();

            // ---------- KPI проєктів (План vs Факт) ----------
            var seriesProjects = new List<ISeries>();
            int colorIndex = 0;

            foreach (var project in projects)
            {
                var color = palette[colorIndex % palette.Length];
                colorIndex++;

                var projectStages = stages.Where(s => s.ProjectId ==
project.Id).ToList();
                int totalStages = projectStages.Count;

                double[] planProgress = allMonths.Select(month =>
                {
                    var monthEnd = month.AddMonths(1).AddDays(-1);
                    int planned = projectStages.Count(s =>
s.PlannedEndDate.HasValue && s.PlannedEndDate.Value <= monthEnd);
                    return totalStages == 0 ? 0 : planned * 100.0 / totalStages;
                }).ToArray();

                double[] factProgress = allMonths.Select(month =>
                {
                    var monthEnd = month.AddMonths(1).AddDays(-1);
                    int done = projectStages.Count(s => s.ActualEndDate.HasValue
&& s.ActualEndDate.Value <= monthEnd);
                    return totalStages == 0 ? 0 : done * 100.0 / totalStages;
                }).ToArray();

                // План — пунктир
                seriesProjects.Add(new LineSeries<double>
                {
                    Name = $"{project.Name} (План)",
                    Values = planProgress,
                    Stroke = new SolidColorPaint(SKColors.Gray)
                    {
```

```
                    StrokeThickness = 2,
                    PathEffect = new DashEffect(new float[] { 6, 4 }, 0)
                },
                GeometrySize = 0,
                LineSmoothness = 0.1
            });

            // Факт — кольором проєкту
            seriesProjects.Add(new LineSeries<double>
            {
                Name = $"{project.Name} (Факт)",
                Values = factProgress,
                Stroke = new SolidColorPaint(color) { StrokeThickness = 3 },
                GeometrySize = 4,
                LineSmoothness = 0.2
            });
        }

        KpiChart.Series = seriesProjects;
        KpiChart.XAxes = new Axis[] { new Axis { Labels = monthLabelsShort,
Name = "Місяць" } };
        KpiChart.YAxes = new Axis[] { new Axis { Name = "% виконання", MinLimit
= 0, MaxLimit = 100 } };
        KpiChart.LegendPosition                                              =
LiveChartsCore.Measure.LegendPosition.Bottom;

        // ---------- Динаміка завдань ----------
        var tasksInRange = tasks
            .Where(t => t.PlannedDate.HasValue &&
                        t.PlannedDate.Value.Date >= rangeStart.Date &&
                        t.PlannedDate.Value.Date <= rangeEnd.Date)
            .ToList();

        int n = allMonths.Count;
        double[] plannedPct = new double[n];
        double[] inProgressPct = new double[n];
        double[] donePct = new double[n];

        for (int i = 0; i < n; i++)
        {
            var m = allMonths[i];
            int year = m.Year;
            int month = m.Month;

            var monthTasks = tasksInRange.Where(t =>
                t.PlannedDate!.Value.Year         ==         year         &&
t.PlannedDate!.Value.Month == month);

            double total = monthTasks.Count();
            if (total == 0) continue;

            plannedPct[i] = monthTasks.Count(t => t.Status == "Заплановано")
* 100.0 / total;
            inProgressPct[i] = monthTasks.Count(t => t.Status == "В процесі")
* 100.0 / total;
            donePct[i] = monthTasks.Count(t => t.Status == "Завершено") *
100.0 / total;
        }

        var monthLabelsLong = allMonths.Select(d => d.ToString("MMMM yyyy",
Uk)).ToArray();

        ProgressChart1.Series = new ISeries[]
        {
```

```
                new StackedColumnSeries<double> { Name = "Заплановано", Values =
plannedPct, Fill = new SolidColorPaint(SKColors.SteelBlue) },
                new StackedColumnSeries<double> { Name = "В процесі", Values =
inProgressPct, Fill = new SolidColorPaint(SKColors.Goldenrod) },
                new StackedColumnSeries<double> { Name = "Виконано", Values =
donePct, Fill = new SolidColorPaint(SKColors.SeaGreen) }
            };

            ProgressChart1.XAxes = new Axis[] { new Axis { Labels =
monthLabelsLong, Name = "Місяць" } };
            ProgressChart1.YAxes = new Axis[] { new Axis { Name = "% від загальної
кількості", MinLimit = 0, MaxLimit = 100 } };
        }

        //============== Експорт у PDF ========================
        private void ExportToPdf_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var dlg = new SaveFileDialog
                {
                    Filter = "PDF файли (*.pdf)|*.pdf",
                    FileName = "Project_Report.pdf"
                };

                if (dlg.ShowDialog() != true) return;

                string tempDir = Path.Combine(Path.GetTempPath(), "Charts");
                Directory.CreateDirectory(tempDir);

                string kpiPath = Path.Combine(tempDir, "kpi.png");
                string progressPath = Path.Combine(tempDir, "progress.png");

                SaveChartAsPng(KpiChart, kpiPath, 950, 400);
                SaveChartAsPng(ProgressChart1, progressPath, 950, 400);

                var doc = new Document();
                var section = doc.AddSection();

                var title = section.AddParagraph("Аналітичний звіт по IT-
проєктах");
                title.Format.Font = new Font("Arial", 16);
                title.Format.Font.Bold = true;
                title.Format.Alignment = ParagraphAlignment.Center;
                title.Format.SpaceAfter = "0.5cm";

                var date = section.AddParagraph("Дата генерації: " +
DateTime.Now.ToString("dd.MM.yyyy HH:mm"));
                date.Format.Font = new Font("Arial", 11);
                date.Format.SpaceAfter = "0.5cm";

                section.AddParagraph("Активні проєкти за вибраний
період:").Format.Font.Bold = true;

                var table = new Table();
                table.Borders.Width = 0.75;
                table.AddColumn(Unit.FromCentimeter(10));
                table.AddColumn(Unit.FromCentimeter(4));

                var header = table.AddRow();
                header.Shading.Color = Colors.LightGray;
                header.Cells[0].AddParagraph("Назва проєкту");
                header.Cells[1].AddParagraph("Прогрес (%)");
                header.Format.Font.Bold = true;
```

```
                foreach (var p in activeProjects)
                {
                    var row = table.AddRow();
                    row.Cells[0].AddParagraph(p.Name);
                    row.Cells[1].AddParagraph($"{p.CompletionPercent:F0}%");
                }

                section.Add(table);
                section.AddParagraph().Format.SpaceAfter = "0.7cm";

                section.AddParagraph("Діаграма 1 – KPI проєктів (План / Факт):").Format.Font.Bold = true;
                section.AddImage(kpiPath).Width = Unit.FromCentimeter(16);

                section.AddParagraph().Format.SpaceBefore = "0.7cm";
                section.AddParagraph("Діаграма 2 – Динаміка виконання завдань:").Format.Font.Bold = true;
                section.AddImage(progressPath).Width = Unit.FromCentimeter(16);

                var footer = section.AddParagraph("\nЗгенеровано системою ProjectMonitoring");
                footer.Format.Font = new Font("Arial", 9);
                footer.Format.Font.Color = Colors.Gray;
                footer.Format.Alignment = ParagraphAlignment.Center;

                var renderer = new PdfDocumentRenderer(true) { Document = doc };
                renderer.RenderDocument();
                renderer.PdfDocument.Save(dlg.FileName);

                MessageBox.Show("PDF-звіт успішно створено!", "Експорт завершено",
                    MessageBoxButton.OK, MessageBoxImage.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка під час експорту: " + ex.Message,
                    "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        // ==================== Допоміжні ========================
        private void SaveChartAsPng(FrameworkElement chart, string filePath, int width, int height)
        {
            var bmp = new RenderTargetBitmap(width, height, 96, 96, System.Windows.Media.PixelFormats.Pbgra32);
            chart.Measure(new Size(width, height));
            chart.Arrange(new Rect(new Size(width, height)));
            bmp.Render(chart);

            using (var fs = new FileStream(filePath, FileMode.Create))
            {
                var encoder = new PngBitmapEncoder();
                encoder.Frames.Add(BitmapFrame.Create(bmp));
                encoder.Save(fs);
            }
        }

        private DateTime ResolveAutoEnd(DateTime from)
        {
            var stages = _service.GetAllStages();
            var tasks = _service.GetAllTasks();

            var dates = stages.SelectMany(s => new[] { s.PlannedStartDate, s.PlannedEndDate, s.ActualStartDate, s.ActualEndDate })
```

```
                                .Concat(tasks.Select(t => t.PlannedDate))
                                .Where(d => d.HasValue)
                                .Select(d => d!.Value)
                                .ToList();

            return dates.Count == 0 ? from : new DateTime(dates.Max().Year,
dates.Max().Month, 1);
        }

        private static List<DateTime> MonthTicks(DateTime from, DateTime to)
        {
            var result = new List<DateTime>();
            for (var d = new DateTime(from.Year, from.Month, 1); d <= new
DateTime(to.Year, to.Month, 1); d = d.AddMonths(1))
                result.Add(d);
            return result;
        }

        private static bool Intersects(DateTime? start, DateTime? end, DateTime
rangeStart, DateTime rangeEnd)
        {
            if (!start.HasValue && !end.HasValue) return false;
            var s = (start ?? end)!.Value;
            var e = (end ?? start)!.Value;
            if (s > e) (s, e) = (e, s);
            return s <= rangeEnd && e >= rangeStart;
        }

        private void GenerateReport_Click(object sender, RoutedEventArgs e)
        {
            var from = StartDatePicker.SelectedDate ?? DefaultStart;
            DateTime? to = EndDatePicker.SelectedDate;
            BuildCharts(from, to);
        }

        private class ProjectInfo
        {
            public string Name { get; set; } = "";
            public double CompletionPercent { get; set; }
        }
    }
}
```

Програмний код вікна ProjectDetailsWindow

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Globalization;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Windows;
using System.Windows.Controls;
using LiveChartsCore;
using LiveChartsCore.SkiaSharpView;
using LiveChartsCore.SkiaSharpView.Painting;
using LiveChartsCore.SkiaSharpView.Painting.Effects;
using SkiaSharp;
using ProjectMonitoring.Models;
using ProjectMonitoring.Services;

namespace ProjectMonitoring.Views
{
    public partial class ProjectDetailsWindow : Window, INotifyPropertyChanged
    {
        private readonly ProjectService _projectService;
        private readonly CultureInfo _uk = new("uk-UA");

        public Project Project { get; }

        private double _completionPercent;
        public double CompletionPercent
        {
            get => _completionPercent;
            set { _completionPercent = value; OnPropertyChanged(); }
        }

        public     ProjectDetailsWindow(Project     project,     ProjectService
projectService)
        {
            InitializeComponent();

            Project = project;
            _projectService = projectService;

            DataContext = this;

            LoadStages();
            LoadTasksForFirstStage();
            UpdateCompletionPercent();

            BuildStagesStatusChart(); // діаграма 1: статуси етапів з підписами
етапів знизу
            BuildPlanFactChart();     // діаграма 2: план vs факт по місяцях
        }

        // ============== ЗАВАНТАЖЕННЯ ДАНИХ ДЛЯ ТАБЛИЦЬ ====================

        private void LoadStages()
        {
            var stages = _projectService
                .GetStagesByProjectId(Project.Id)
                .OrderBy(s => s.PlannedStartDate ?? DateTime.MaxValue)
                .ToList();
```

```
            StagesGrid.ItemsSource = stages;
        }

        private void LoadTasksForStage(int stageId)
        {
            TasksGrid.ItemsSource = _projectService
                .GetTasksByStageId(stageId)
                .OrderBy(t => t.PlannedDate)
                .ToList();
        }

        private void LoadTasksForFirstStage()
        {
            if (StagesGrid.Items.Count == 0)
            {
                TasksGrid.ItemsSource = null;
                return;
            }

            if (StagesGrid.Items[0] is ProjectStage firstStage)
                LoadTasksForStage(firstStage.Id);
        }

        // ==================== ОБРОБНИКИ КНОПОК/ВИБОРУ ====================

        private      void      StagesGrid_SelectionChanged(object      sender,
SelectionChangedEventArgs e)
        {
            if (StagesGrid.SelectedItem is ProjectStage selectedStage)
                LoadTasksForStage(selectedStage.Id);
        }

        private void AddStage_Click(object sender, RoutedEventArgs e)
        {
            var window = new AddStageWindow(_projectService, Project.Id);
            window.ShowDialog();

            LoadStages();
            LoadTasksForFirstStage();
            UpdateCompletionPercent();
            BuildStagesStatusChart();
            BuildPlanFactChart();
        }

        private void AddTask_Click(object sender, RoutedEventArgs e)
        {
            if (StagesGrid.SelectedItem is not ProjectStage selectedStage)
            {
                MessageBox.Show("Виберіть етап для додавання завдання!");
                return;
            }

            var window = new AddTaskWindow(selectedStage.Id, _projectService);
            window.ShowDialog();

            LoadTasksForStage(selectedStage.Id);
            UpdateCompletionPercent();
            BuildStagesStatusChart();
            BuildPlanFactChart();
        }
```

```
        private void Reload_Click(object sender, RoutedEventArgs e)
        {
            LoadStages();
            LoadTasksForFirstStage();
            UpdateCompletionPercent();
            BuildStagesStatusChart();
            BuildPlanFactChart();
        }

        // ==================== ПРОГРЕС ПРОЄКТУ ====================

        private void UpdateCompletionPercent()
        {
            CompletionPercent                                                    =
_projectService.CalculateProjectProgress(Project.Id);
        }

        // === ДІАГРАМА 1: СТАТУСИ ЕТАПІВ (накопичення, % по КОЖНОМУ етапу) =====
        private void BuildStagesStatusChart()
        {
            var  stages  =  (_projectService.GetStagesByProjectId(Project.Id)  ??
Enumerable.Empty<ProjectStage>()).ToList();

            if (stages.Count == 0)
            {
                StagesStatusChart.Series = Array.Empty<ISeries>();
                StagesStatusChart.XAxes  =  new  Axis[]  {  new  Axis  {  Labels  =
Array.Empty<string>() } };
                StagesStatusChart.YAxes = new Axis[] { new Axis { Name = "%",
MinLimit = 0, MaxLimit = 100 } };
                return;
            }

            static string Norm(string? s) => (s ?? "").Trim().ToLowerInvariant();

            double[] planned = new double[stages.Count];
            double[] inProcess = new double[stages.Count];
            double[] done = new double[stages.Count];

            for (int i = 0; i < stages.Count; i++)
            {
                var st = stages[i];
                var status = Norm(st.Status);

                // Заплановано
                if (status == "заплановано")
                {
                    planned[i] = 100.0;
                    continue;
                }

                // В процесі (враховано можливі варіанти написання)
                if (status == "в процесі" || status == "в процесі" || status ==
"виконується" || status == "виконується")
                {
                    inProcess[i] = 100.0;
                    continue;
                }

                // Завершено (синоніми)
```

```
                if (status == "завершено" || status == "виконано" || status ==
"готово")
                {
                    done[i] = 100.0;
                    continue;
                }

                // інші/невідомі стани — 0/0/0
            }

            string[] labels = stages
                .Select(s => string.IsNullOrWhiteSpace(s.StageName) ? $"Етап
{s.Id}" : s.StageName)
                .ToArray();

            StagesStatusChart.Series = new ISeries[]
            {
                new StackedColumnSeries<double>
                {
                    Name = "Заплановано",
                    Values = planned,
                    Fill = new SolidColorPaint(SKColors.SteelBlue)
                },
                new StackedColumnSeries<double>
                {
                    Name = "В процесі",
                    Values = inProcess,
                    Fill = new SolidColorPaint(SKColors.Goldenrod)
                },
                new StackedColumnSeries<double>
                {
                    Name = "Завершено",
                    Values = done,
                    Fill = new SolidColorPaint(SKColors.SeaGreen)
                }
            };

            StagesStatusChart.XAxes = new Axis[]
            {
                new Axis
                {
                    Labels = labels,
                    Name = "Етапи проєкту",
                    LabelsRotation = 0
                }
            };

            StagesStatusChart.YAxes = new Axis[]
            {
                new Axis
                {
                    Name = "% стану",
                    MinLimit = 0,
                    MaxLimit = 100
                }
            };
        }

        // ==== ДІАГРАМА 2: ПЛАН vs ФАКТ (по місяцях, для поточного проєкту========

        private void BuildPlanFactChart()
```

```
        {
            var stages = (_projectService.GetStagesByProjectId(Project.Id) ??
Enumerable.Empty<ProjectStage>()).ToList();

            var months = stages
                .SelectMany(s => new[] { s.PlannedStartDate, s.PlannedEndDate,
s.ActualStartDate, s.ActualEndDate })
                .Where(d => d.HasValue)
                .Select(d => new DateTime(d!.Value.Year, d.Value.Month, 1))
                .Distinct()
                .OrderBy(d => d)
                .ToList();

            if (months.Count == 0)
            {
                months.Add(new                      DateTime(DateTime.Today.Year,
DateTime.Today.Month, 1));
            }

            var labels = months.Select(m => m.ToString("MMM", _uk)).ToArray();
            int totalStages = stages.Count;

            double[] planProgress = months.Select(month =>
            {
                var monthEnd = month.AddMonths(1).AddDays(-1);
                int plannedCount = stages.Count(s => s.PlannedEndDate.HasValue &&
s.PlannedEndDate.Value <= monthEnd);
                return totalStages == 0 ? 0 : plannedCount * 100.0 / totalStages;
            }).ToArray();

            double[] factProgress = months.Select(month =>
            {
                var monthEnd = month.AddMonths(1).AddDays(-1);
                int doneCount = stages.Count(s => s.ActualEndDate.HasValue &&
s.ActualEndDate.Value <= monthEnd);
                return totalStages == 0 ? 0 : doneCount * 100.0 / totalStages;
            }).ToArray();

            PlanFactChart.Series = new ISeries[]
            {
                new LineSeries<double>
                {
                    Name = "План",
                    Values = planProgress,
                    Stroke = new SolidColorPaint(SKColors.Gray)
                    {
                        StrokeThickness = 2,
                        PathEffect = new DashEffect(new float[] { 6, 4 }, 0)
                    },
                    GeometrySize = 0,
                    LineSmoothness = 0
                },
                new LineSeries<double>
                {
                    Name = "Факт",
                    Values = factProgress,
                    Stroke    =    new    SolidColorPaint(SKColors.DodgerBlue){
StrokeThickness = 3 },
                    GeometrySize = 4,
                    LineSmoothness = 0
                }
```

```
            };

            PlanFactChart.XAxes = new Axis[]
            {
                new Axis { Labels = labels, Name = "Місяць", LabelsRotation = 0 }
            };
            PlanFactChart.YAxes = new Axis[]
            {
                new Axis { Name = "% виконання", MinLimit = 0, MaxLimit = 100 }
            };
        }

        // ====================== INotifyPropertyChanged ======================

        public event PropertyChangedEventHandler? PropertyChanged;
        private void OnPropertyChanged([CallerMemberName] string? name = null) =>
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
    }
}
```