МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
НН Інститут ІТ та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістра

на тему: «**Розробка моделей управління ризиками в проєкті»**

Виконав: студент 2 курсу, групи МУП-21
другого (магістерського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Управління проєктами»
Бондарчука Владислава Олександровича

Керівник: *кандидат економічних наук, доцент*
Новоселецький Олександр Миколайович

Рецензент: *кандидат технічних наук, доцент,*
доцент кафедри прикладної математики Донецького національного університету імені Василя Стуса
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)
Протокол № 5 від 04 грудня 2025 р.

Острог, 2025

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня магістра

Тема: *Розробка моделей управління ризиками в проєкті*……
……...……
…
Автор:……

Науковий керівник:
……...…….
Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *86 с., 23рис., 7 табл., 1 додаток, 11 джерел.*
Ключові слова: *управління ризиками, ISO 31000, GPT-API, Risk-App, матриця «ймовірність × вплив», Fault Tree Analysis, Decision Tree, RACI-матриця, KPI, виробничі проєкти*

Короткий зміст праці:

У кваліфікаційній роботі розроблено комплексну модель управління ризиками для підприємства з виробництва жниварок, що інтегрує методологію ISO 31000, практики PMI PMBOK Guide та сучасні інструменти штучного інтелекту. На основі аналітичного огляду побудовано ієрархію джерел ризику (RBS) і створено узагальнений реєстр з параметрами ймовірності (P) та впливу (I). Реалізовано програмний прототип Risk-App, який формує структуровані запити до GPT-API, одержує відповіді у форматі JSON, автоматично будує матрицю P×I, дерево відмов (FTA) та дерево рішень (Decision Tree), а також підтримує експорт реєстру ризиків у форматах CSV/XLSX.

Модель апробовано на виробничих сценаріях жниварок із використанням даних PLM/ERP/MES/QMS-систем. Проведене тестування підтвердило коректність побудови діаграм, відтворюваність результатів та інтеграційну готовність інструмента. На основі кількісного індексу SRI проведено оцінку ефективності мітигаційних дій, що продемонструвала зниження сукупного ризикового впливу на ≈ 50 % і позитивний економічний ефект за показниками OEE та ROI.

Запропоновані рекомендації визначають організаційний і технічний порядок впровадження Risk-App - розподіл ролей за RACI-матрицею, політику ведення реєстру, набір KPI та процедури моніторингу. Сформовано дорожню карту розвитку системи у напрямках подієво-орієнтованих інтеграцій, використання RAG-підходів, впровадження кількісних методів Монте-Карло та оптимізації портфеля контрзаходів.

ANNOTATION
of the Master's Qualification Thesis

Topic: *Development of Risk Management Models in a Project*

Author:……

Scientific Supervisor:
……...……..
Defended on «……»…… 20__.

Explanatory note to the qualification thesis*: 86 pages, 23 figures, 7 tables, 1 appendix, 11 sources.*
Keywords: risk management, ISO 31000, GPT-API, Risk-App, probability-impact matrix, Fault Tree Analysis, Decision Tree, RACI matrix, KPI, production projects.

Summary:

The qualification thesis presents the development of a comprehensive risk management model for an agricultural machinery manufacturing enterprise, integrating the ISO 31000 methodology, PMI PMBOK Guide practices, and modern artificial intelligence tools. Based on an analytical review, a hierarchical structure of risk sources (RBS) was constructed and a unified risk register was developed with parameters of probability (P) and impact (I). A software prototype Risk-App was implemented, which generates structured prompts to the GPT-API, receives responses in JSON format, automatically builds the P×I matrix, Fault Tree (FTA), and Decision Tree, and supports exporting the risk register in CSV/XLSX formats.

The model was tested on real manufacturing scenarios of combine harvesters using PLM/ERP/MES/QMS system data. Testing confirmed the correctness of diagram generation, reproducibility of results, and the integration readiness of the tool. Based on the quantitative SRI index, an assessment of the effectiveness of mitigation actions was carried out, demonstrating an approximately 50% reduction in total risk exposure and a positive economic effect according to OEE and ROI indicators.

The proposed recommendations define the organizational and technical framework for implementing Risk-App, including role distribution according to the RACI matrix, risk register management policy, KPI set, and monitoring procedures. A development roadmap for the system was also formulated, focusing on event-driven integrations, RAG-based approaches, application of quantitative Monte Carlo methods, and optimization of the mitigation portfolio.

ЗМІСТ

ВСТУП

Актуальність теми зумовлена швидким поширенням інтелектуальних сервісів, що використовують великі мовні моделі для підтримки прийняття рішень у програмній інженерії. Інтеграція таких сервісів через API дає змогу автоматизувати створення технічних концепцій, уточнення вимог і початковий аналіз ризиків, скорочуючи тривалість передпроєктних робіт та підвищуючи їх якість. Методологічне підґрунтя дослідження спирається на міжнародний стандарт ISO/IEC/IEEE 12207 [1], який встановлює узгоджену рамку процесів життєвого циклу програмного забезпечення і забезпечує системність планування, виконання та оцінювання результатів на всіх стадіях розроблення .

Метою дослідження є розробка моделей і методів управління ризиками в інжинірингових проєктах із застосуванням сучасних підходів до ідентифікації, оцінки, аналізу й мінімізації ризиків, а також організація впровадження розробленого інструментарію в інформаційні системи управління проєктами.

Для досягнення мети дослідження необхідно послідовно виконати наступні завдання:

Завдання роботи

- Проаналізувати сучасні стандарти, підходи та інструменти управління ризиками у проєктах.
- Ідентифікувати типові ризики для інжинірингових проєктів з урахуванням специфіки предметної області.
- Сконструювати імітаційні та аналітичні моделі для оцінювання ризиків із використанням різних методик.
- Розробити інформаційну модель інтеграції управління ризиками в програмне забезпечення управління проєктами (UML, BPMN, API).
- Провести дослідження ефективності моделей управління ризиками на типових сценаріях виконання проєктів.
- Надати рекомендації щодо впровадження розроблених інструментів у діючі системи управління проєктами.

Об’єктом дослідження є процеси управління ризиками в інжинірингових, виробничих, IT-проєктах, які впливають на досягнення цілей проекту та ефективність менеджменту.

Предметом дослідження виступають моделі, інструменти, алгоритми та інформаційні технології і процеси, що забезпечують ідентифікацію, оцінку, аналіз і управління ризиками у сучасних проєктах.

У роботі використовуються такі методи дослідження: аналіз і синтез літературних джерел, застосування міжнародних стандартів управління ризиками (ISO 31000, PMI), побудова структур ризиків за допомогою Risk Breakdown Structure (RBS), розробка моделей і оцінка ризиків з використанням Risk Matrix, Bow-tie, Fault Tree Analysis (FTA), Event Tree Analysis (ETA) та Decision Tree, а також імітаційне моделювання на основі методу Монте-Карло. Значна увага приділяється експертним оцінкам, сценарному аналізу, аналізу чутливості, побудові UML- та BPMN-діаграм бізнес-процесів для інтеграції моделей ризиків у інформаційні системи, а також використанню інструментів статистичного аналізу та комп’ютерного моделювання для оцінки ефективності запропонованих підходів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ РИЗИКАМИ В ПРОЄКТАХ

1.1. Концепція та класифікація ризиків у проєктному управлінні

Управління ризиками у проєктах базується на усвідомленні невизначеності як постійного чинника, що здатен як погіршувати, так і поліпшувати досягнення цілей за обмежень часу, бюджету та якості. Міжнародний стандарт ISO 31000 трактує ризик як вплив невизначеності на цілі та пропонує інтегрований, циклічний підхід: встановлення контексту і критеріїв, ідентифікація, аналіз та оцінювання ризиків[13], вибір і реалізація заходів впливу, а також постійні комунікація, моніторинг і перегляд. Як зображено на рисунку 1.1 [2], концепція поєднує три взаємопов'язані частини - принципи, структуру (framework) та процес - що забезпечують системність і безперервне вдосконалення управління ризиками [14].

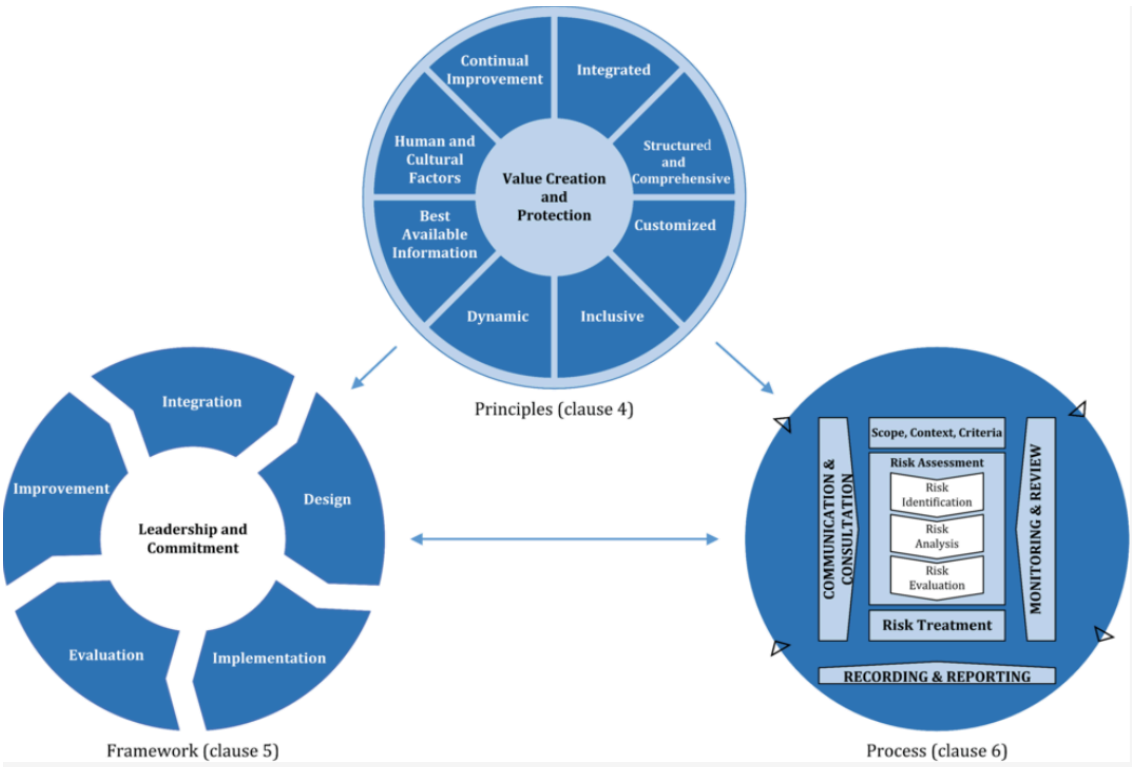

Рис. 1.1 Схема ISO 31000: принципи, структура та процес управління ризиками (адаптовано за ISO 31000:2018)

У проєктному контексті ризик розглядають як подію або умову з невизначеною імовірністю настання та наслідками для цілей проєкту; важливо, що спектр наслідків включає як загрози, так і можливості. Практика PMI (Project Management Institute - Інститут управління проєктами) підкреслює «джерельно-орієнтоване» мислення: опис ризику, оцінювання імовірності та впливу, а також структуроване групування джерел невизначеності для забезпечення повноти виявлення та прозорої пріоритезації в реєстрі ризиків[12]. Це робить процес повторюваним і порівнюваним між різними ініціативами, у тому числі в IT-проєктах, де суттєве місце посідають технічні, правові й організаційні невизначеності.

Ключовою ідеєю сучасної концепції є зсув від реактивної до проактивної поведінки. Стандарт ISO 31000 наголошує на визначенні критеріїв ризику та схильності до ризику, що дають змогу встановлювати порогові значення прийнятності, узгоджувати їх із зацікавленими сторонами та відстежувати ефективність заходів впливу у замкненому циклі «плануй-виконуй-перевіряй-дій». Таким чином ризик-менеджмент інтегрується у стратегічні та операційні рішення, а результати системно документуються й переглядаються.

Класифікація ризиків забезпечує повноту і несуперечливість ідентифікації. Типово її здійснюють за кількома ознаками: за ознакою впливу (загрози проти можливостей), за джерелом виникнення (внутрішні організаційні та технічні проти зовнішніх регуляторних, ринкових і середовищних), а також за фазами життєвого циклу проєкту (ініціювання, планування, виконання, контроль, завершення). Для IT-ініціатив, зокрема інтеграції сервісів ШІ, характерні групи технічних ризиків сумісності й продуктивності, правові та етичні ризики роботи з даними, комерційні ризики залежності від постачальника, а також операційні ризики масштабування та супроводу. Стандартизовані шкали імовірності та впливу, прийняті в організації, уможливлюють порівнюваність профілів і коректне резервування часу й бюджету.

Ефективному структуруванню невизначеності слугує ієрархія джерел ризику - Risk Breakdown Structure (RBS). Вона відображає «дерево» від верхнього рівня до дедалі деталізованіших категорій, полегшуючи призначення власників ризиків та виявлення «сліпих зон». У цій роботі використано узагальнену таблицю RBS для типового проєкту за Hall & Hulett; її подано як таблицю 1.1 [3]. Практична цінність RBS полягає не лише в повноті первинної ідентифікації, а й у можливості порівнювати експозицію між проєктами, уніфікувати терміни та будувати матриці перетину RBS із WBS для пошуку концентрацій ризику.

Таблиця 1.1

Рівень 0	Рівень 1	Рівень 2	Рівень 3 (приклади)
Project risk	Management	Corporate	Організаційна стабільність; Фінанси; тощо
		Customer & stakeholder	Договірні умови; Визначення та стабільність вимог; тощо
	External	Natural environment	Фізичне середовище; Об’єкти та майданчики; Місцеві служби; тощо
		Cultural	Політичні чинники; Правові/регуляторні вимоги; Групи впливу; тощо
		Economic	Ринок праці; Умови праці; Фінансовий ринок; тощо
	Technology	Requirements	Невизначеність обсягу; Умови використання; Складність; тощо
		Performance	Зрілість технологій; Технологічні межі; тощо
		Application	Організаційний досвід; Кваліфікація персоналу; Матеріальні ресурси; тощо

Ієрархія Risk Breakdown Structure (RBS) для типового проєкту

В операційному застосуванні RBS стає «словником» джерел ризику для проєктів певного типу, прискорюючи роботу з реєстром і покращуючи якість планів реагування. На практиці це означає узгоджені шаблони опису, уніфіковані шкали оцінювання та зрозумілі механізми ескалації, що в

сукупності підвищує прогнозованість результатів виконання. Для цифрових рішень і інтеграцій із зовнішніми API RBS допомагає завчасно виділити проблемні кластери - від вимог і архітектури до правових обмежень та підготовки персоналу - і цілеспрямовано інвестувати в превентивні заходи.

Синергія ISO31000 з практиками PMI полягає в поєднанні метапроцесу «принципи-рамка-процес» із конкретними інструментами на кшталт RBS, реєстру ризиків і планів реагування. Для нашого дослідження таке поєднання означає одночасно системне вбудовування ризик-менеджменту у всі рішення та операційний набір рутин, які роблять дії відтворюваними і вимірюваними.

1.2. Методи та моделі аналізу ризиків

Початкове якісне ранжування доцільно виконувати на матриці «ймовірність × вплив», де кожен ризик розміщується у клітинці, що відповідає оціненим шкалам частоти та тяжкості наслідків. Така «теплова карта» забезпечує швидку візуалізацію пріоритетів і створює спільну мову для команди, коли потрібно вирішити, які позиції переходять у глибше опрацювання та план реагування [4]. Як видно на рис. 1.2, типова матриця має «зелені», «жовті» й «червоні» зони, що полегшує комунікацію зі стейкґолдерами, а також дозволяє фіксувати прийнятні пороги для ескалації.

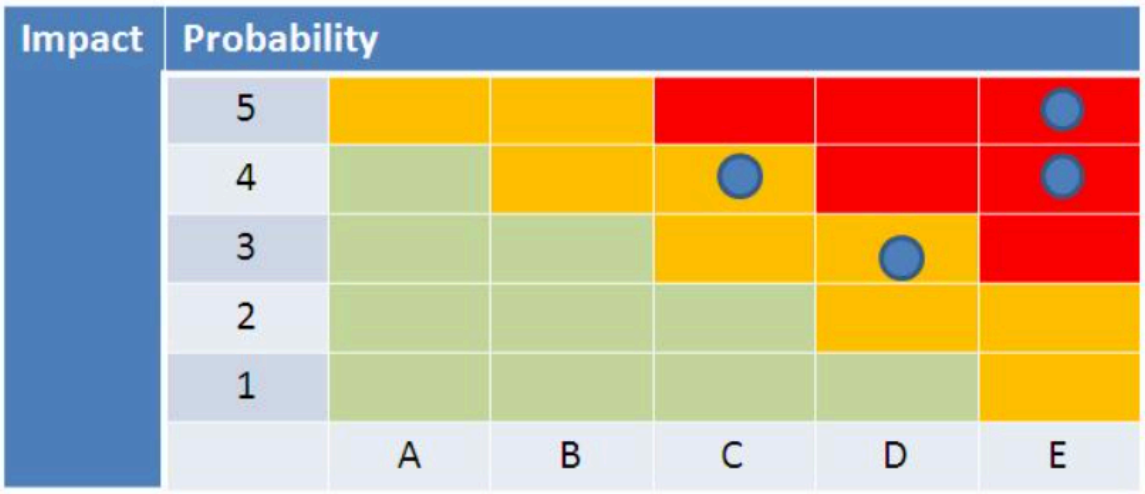

Рис. 1.2 Матриця «ймовірність × вплив» (Risk Matrix).

Разом із перевагами матриця має й обмеження: грубі шкали можуть приховувати нюанси, а добуток P×I не завжди відображає реальну відносну небезпеку. Тому її варто використовувати як інструмент первинного скринінгу з наступним переходом до формальніших технік. У нашому контексті інтеграції GPT API вона допомагає швидко виділити групи ризиків - технічні, правові, операційні - і визначити, які з них потребують моделювання або розкладання на причинно-наслідкові ланцюги.

Для системного зв'язування причин і наслідків навколо однієї «події ризику» зручно застосовувати діаграму «метелик» (bow-tie). Ліва частина моделі концентрує загрози та превентивні бар'єри, права - наслідки та пом'якшувальні заходи, що наочно показує, де саме існують прогалини контролів[17] і які бар'єри потребують підсилення [5]. На рис. 1.3 це зображено як симетричну структуру «причини - подія - наслідки», у центрі якої фокусується увага команди.

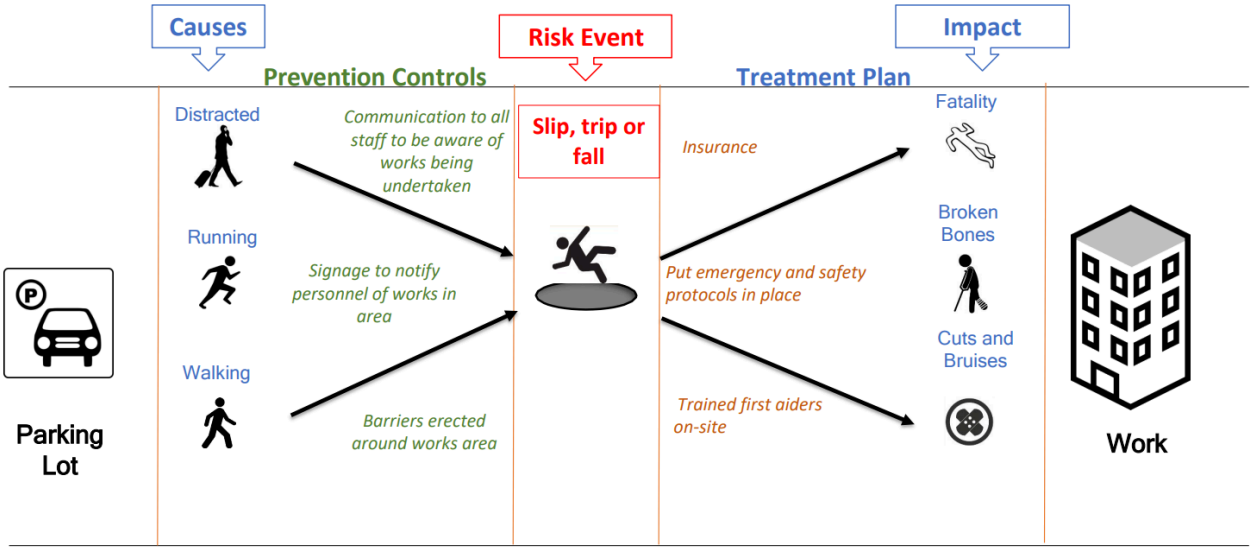

Рисунок 1.3 Діаграма «метелик» (Bow-tie): причини - бар'єри - наслідки.

Практична цінність bow-tie у проєктах із ШІ полягає в балансі між профілактикою та реагуванням: наприклад, для ризику «витік чутливих даних у

запитах» превентивними бар’єрами будуть фільтри й політики анонімізації на вході, а пом’якшувальними - журнали аудиту, механізми відкликання токенів і процедури повідомлення інцидентів. Така карта контролів стає робочим артефактом для узгодження відповідальностей.

Коли необхідно перевести обговорення причин у формальне поле та оцінити внесок кожного чинника у верхню подію [15], застосовують Fault Tree Analysis. FTA починається з інциденту у вершині дерева й через логічні ворота AND/OR розкладає його до базових подій[18], що дозволяє як якісно, так і кількісно оцінити ймовірність інциденту [16] та визначити мінімальні зрізи - найменші комбінації причин, достатні для його настання [6]. На рис. 1.4 подано типову структуру дерева, придатну для декомпозиції технічних відмов.

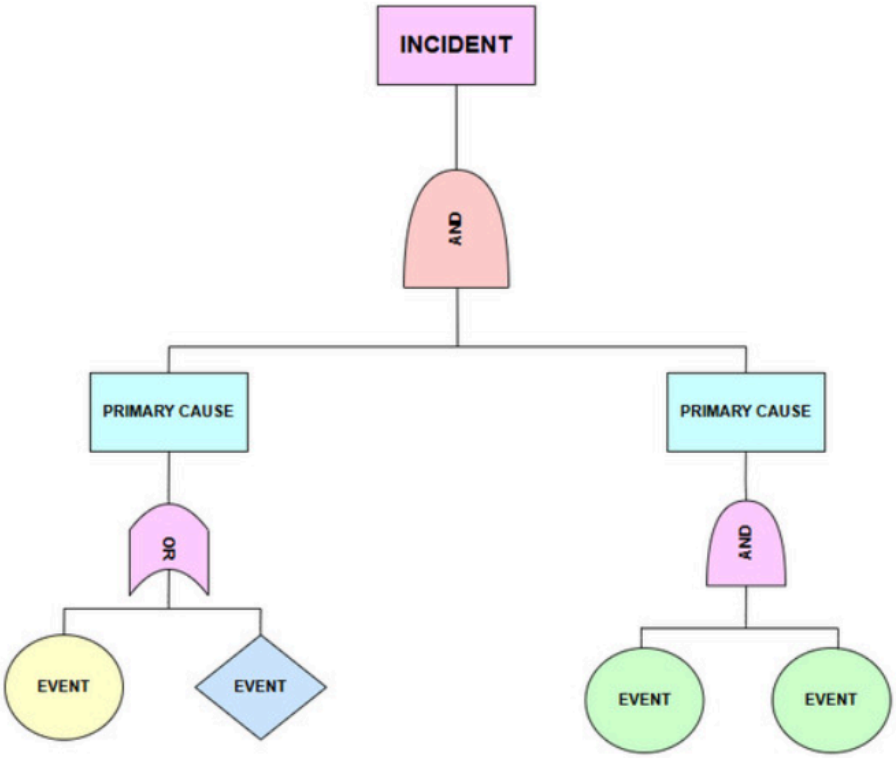

Рисунок 1.4 - Дерево відмов (Fault Tree Analysis)

У нашому дипломному проєкті FTA доречно використовувати для технічних ланцюгів залежностей: наприклад, «помилка автентифікації» може виникати лише за одночасного збою кеша ключів та відмови резервного обміну

ключами (вузол AND), або через будь-яку з альтернативних базових подій, як-от закінчення строку дії токена чи мережевий таймаут (вузол OR). Такий розклад чітко показує, які контролі дають найбільший ефект зниження ризику.

Комплементарним до FTA є Event Tree Analysis, що рухається «від події до наслідків». ETA починається з ініціювальної події та далі розгалужується послідовностями «успіх/відмова» для бар'єрів чи функцій безпеки, формуючи сценарії з відповідними гілковими ймовірностями[21] та підсумковими наслідками [7]. На рис. 1.5 показано, як обчислюються ймовірності фінальних станів як добутки значень уздовж гілок.

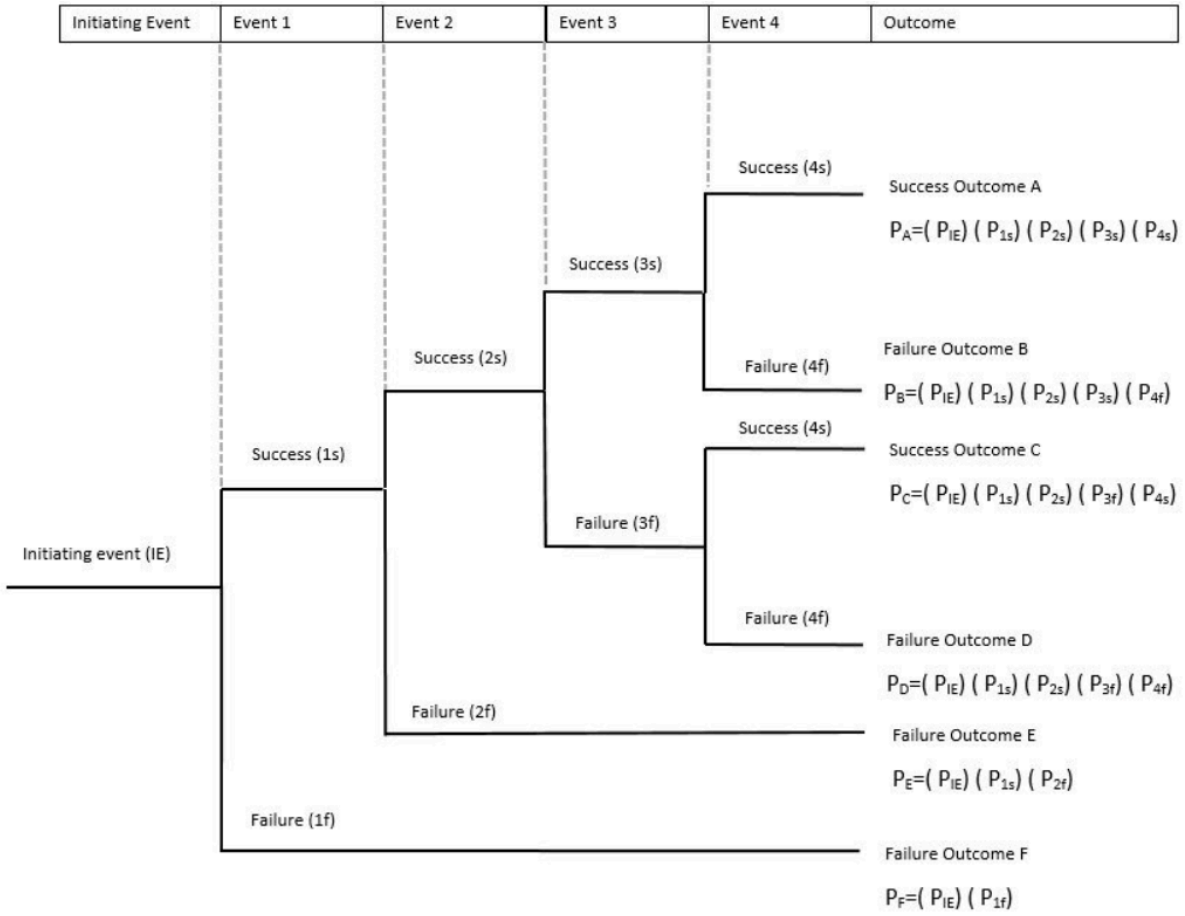

Рисунок 1.5 Дерево подій (Event Tree Analysis)

У застосуванні до GPT-інтеграцій ETA зручна для моделювання експлуатаційних сценаріїв: наприклад, «помилка відповіді моделі» як ініціювальна подія веде до гілок перевірки форматів, повторних спроб,

фолбек-моделі та ручної модерації; множення відповідних імовірностей дає профіль ризику для кожної комбінації дій і дозволяє обґрунтувати інвестиції у найвпливовіші бар’єри.

Коли потрібно обрати між альтернативними стратегіями реагування чи варіантами архітектурних рішень, доцільно використовувати дерево рішень. Воно чергує вузли вибору та випадкових подій і дозволяє розрахувати очікувану цінність кожного маршруту з урахуванням імовірностей і витрат/вигод, що робить вибір прозорим і відтворюваним [8]. Приклад оформлення наведено на рис. 1.6 і показує, як оцінка EV може схилити рішення на користь, скажімо, фолбек-архітектури з вищими операційними витратами, але нижчим профілем ризику.

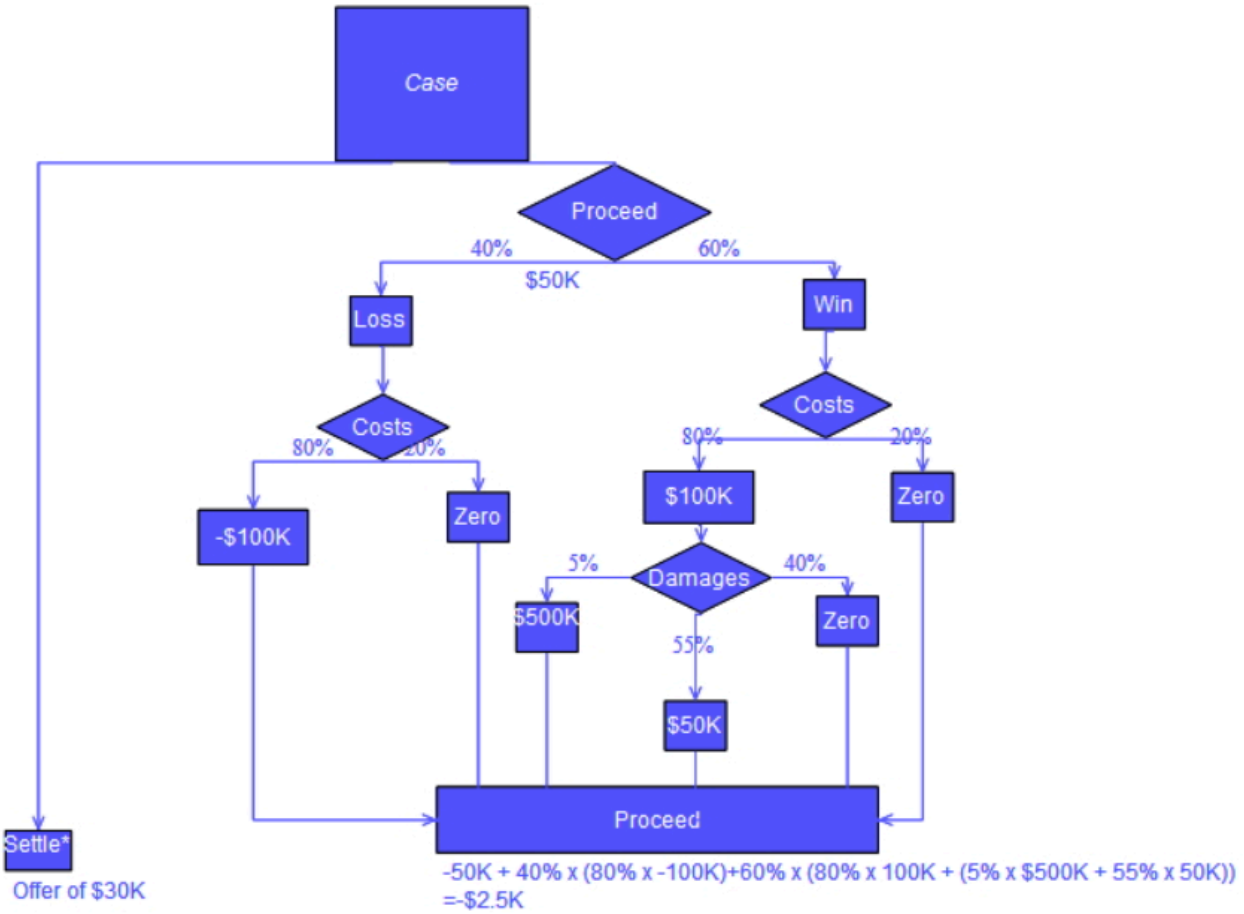

Рисунок 1.6 Дерево рішень (Decision Tree) для порівняння альтернатив.

Для нашого продукту дерево рішень допоможе, наприклад, порівняти три підходи до оброблення помилок API: «повторні спроби», «миттєве переключення на менш точну модель» і «черга ручної перевірки». Фіксуючи на гілках імовірності відмов і очікувані втрати SLA, можна вибрати стратегію, яка мінімізує сумарну очікувану шкоду за заданих обмежень вартості.

Таблиця 1.2

Огляд методів і моделей аналізу ризиків

Метод	Тип оцінювання	Суть підходу	Вхідні дані	Основний результат
Матриця «Ймовірність×Вплив»	Якісна	Ранжування ризиків за зонами теплової карти	Оціночні шкали імовірності та наслідків	Пріоритет ризику, зона P×I
Bow-tie	Напівкількісна	Причини → подія ризику ← наслідки з превентивними/пом’якшувальними бар’єрами	Перелік загроз, контролів і наслідків	Карта контролів, прогалини в захисті
FTA (дерево відмов)	Логіко-кількісна	Дедуктивний розклад верхньої події через AND/OR до базових причин	Ймовірності базових подій	Оцінка ймовірності інциденту, критичні комбінації
ETA (дерево подій)	Кількісна	Індуктивні сценарії «успіх/відмова» після ініціювальної події	Імовірності спрацювання бар’єрів	Ймовірності та наслідки сценаріїв
Decision Tree	Кількісна	Порівняння альтернатив за очікуваною цінністю	Імовірності, витрати/вигоди	Найкраща стратегія з урахуванням EV

У підсумку практичний конвеєр аналізу має вигляд: матриця «ймовірність × вплив» як фільтр для пріоритизації; bow-tie для картування контролів навколо центральної події; FTA для дедуктивного розбору причин; ETA для індуктивного моделювання послідовностей імовірностей; дерево рішень для вибору між альтернативами з урахуванням очікуваної цінності. Зведені

характеристики методів, типи вхідних даних і результати подано в табл. 1.2, що полегшує узгодження інструментарію з фазами життєвого циклу та типами невизначеності в проєкті.

1.3. Особливості ризиків у проєктах з виробництва техніки

Виробничі проєкти у сфері машинобудування відзначаються підвищеною концентрацією ризиків, що безпосередньо пов’язані з безпечністю конструкції та відповідністю нормативним вимогам протягом усього життєвого циклу виробу[23]. На етапах дослідження та проєктування критичним є своєчасне виявлення небезпек, оцінювання їх тяжкості та ймовірності, а також визначення експозиції користувача або оператора. У межах підходу, закріпленого в ISO 12100, зниження ризику відбувається за ієрархією пріоритетів: насамперед через конструктивно безпечні рішення, далі через захисні технічні заходи та лише потім через інформацію для безпечної експлуатації й навчання персоналу [9].

Специфіка виробництва техніки посилюється багатоступеневістю валідації: лабораторні випробування й розрахунки мають підтверджуватися прототипуванням, випробуваннями на міцність і стійкість, контрольними збираннями, а також сертифікаційними процедурами на відповідних ринках збуту [26]. Кожна інженерна зміна породжує повторне оцінювання ризику та актуалізацію технічного файлу виробу. У практиці підприємств це перетворюється на зв’язку методів: D-FMEA на рівні вузлів для запобігання режимам відмов, FTA для доведення шляхів до верхніх подій, P-FMEA у виробництві для контролю стабільності процесів і запобігання дефектам. Значущою групою є також ризики надійності та зношування, що визначають гарантійні зобов’язання, вартість життєвого циклу та потребу в запасі міцності й резервуванні функцій[25].

На межі конструкторських та виробничих рішень з’являються ризики інтеграції: сумісність механічних і електронних модулів, адекватність сенсорики та систем керування, стійкість до електромагнітних завад,

коректність калібрування та алгоритмів безпеки [24]. Постачальницький ланцюг додає власні невизначеності, від коливань термінів і цін до доступності критичних компонентів і ризику їх виведення з виробництва. Усе це змушує заздалегідь закладати альтернативи, проводити кваліфікацію постачальників, передбачати замінники та перевіряти їх вплив на безпеку й надійність. Важливою стає й відповідність вимогам до маркування, інструкцій і навчання, оскільки людський фактор часто перетворює залишкові ризики в інциденти.

У сучасних підприємствах виробництво техніки невіддільне від цифрового ланцюжка даних: CAD/CAE → PLM → ERP → MES/SCADA → сервісні платформи. Тому вагомою складовою профілю ризику стають кіберфізичні ризики операційних технологій [28]. Рекомендації NIST SP 800-82 вимагають «глибинної оборони» для OT-середовища: інвентаризації активів, сегментації мереж за рівнями виробництва, розмежування зон і кондуїтів, створення демілітаризованої зони між IT та OT, багаторівневого моніторингу, керування вразливостями з урахуванням безперервності процесів і особливих сценаріїв реагування на інциденти [10, 31]. Як показано на рисунку 1.7, референс-архітектура включає окремі сервери автентифікації для IT та OT, проміжну DMZ з вузлами віддаленого доступу та «історіками» даних, міжмережеві екрани між усіма рівнями та контрольовані шлюзи для обміну з корпоративними системами [10].

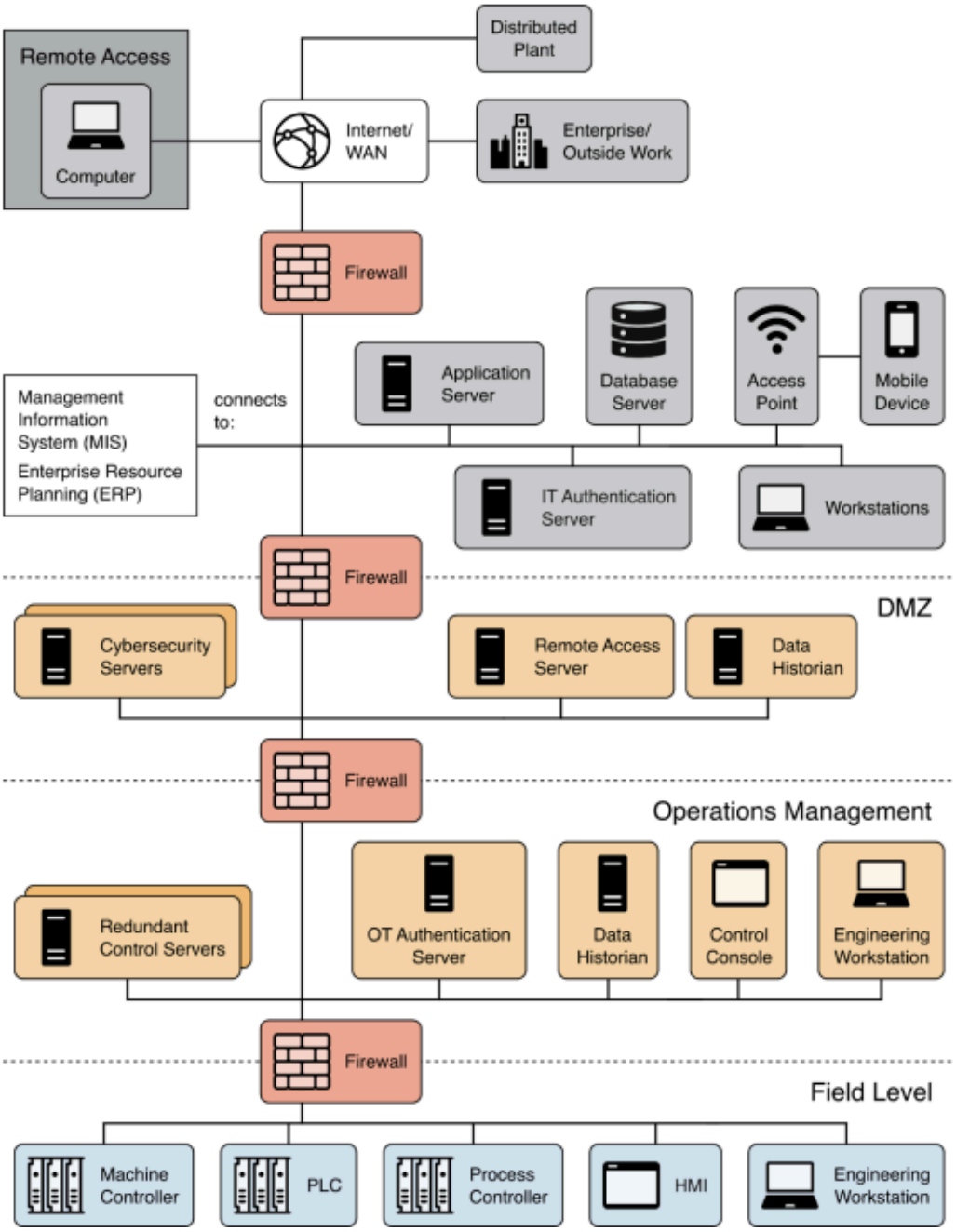

Рис. 1.7. Узагальнена архітектура OT/ICS із сегментацією мереж та DMZ між IT і OT

Таке розшарування не лише зменшує ймовірність компрометації контролерів і робочих станцій інженерів, а й локалізує наслідки можливого інциденту, не допускаючи ескалації у фізичні ушкодження, простій ліній чи небезпечні ситуації для персоналу [28].

Окремо варто розглянути вплив IT-відділу на управління ризиками у таких проєктах. По-перше, IT забезпечує цілісність «цифрового двійника» виробу та процесу: керування версіями CAD-моделей і специфікацій у PLM, простежуваність змін до технічних вимог, узгодженість із ERP та MES. Це мінімізує ризики помилок через розсинхронізовані дані, неправильні варіанти складання, застарілі технологічні карти [42]. По-друге, IT формує керовані ланцюжки постачання програмного забезпечення та прошивок: політики підпису й перевірки цілісності, ведення SBOM для контролерів і вбудованих систем, планування «вікон» оновлення без порушення безперервності. По-третє, IT спільно з виробництвом вибудовує захищений віддалений доступ постачальників, керує ідентичностями та багатофакторною автентифікацією, встановлює правила ведення журналів й кореляції подій між SOC і OT-моніторингом [29,43].

Таким чином, особливості ризиків у проєктах з виробництва техніки полягають у необхідності поєднати інженерно-конструкторську безпеку, надійність і відповідність із кіберфізичною стійкістю виробничого середовища. Ієрархія зниження ризику «від конструкції до інструкцій» задає логіку прийняття інженерних рішень [9], тоді як «глибинна оборона» та сегментація IT/OT забезпечують операційну керованість і відсіч інцидентам на межі цифрового й фізичного світів [10].

Висновки до розділу 1

Проведений у розділі огляд теоретичних засад засвідчив, що сучасне управління ризиками в проєктах спирається на інтегрований, ітеративний підхід ISO 31000, у якому принципи, рамка (framework) та процес утворюють єдину систему для прийняття рішень за умов невизначеності. У цьому підході ризик трактується як вплив невизначеності на цілі, а отже розглядається не лише як загроза, а й як можливість підвищення результативності. Практики PMI доповнюють стандарт операційними інструментами - реєстром, шкалами

ймовірності та впливу, структурованими описами і механізмами ескалації - що робить процес відтворюваним і порівнюваним між ініціативами. Поєднання ISO 31000 з інструментарієм PMI формує методологічну основу роботи: визначення контексту та критеріїв, цілеспрямована ідентифікація, системне оцінювання й контроль виконання заходів у циклі PDCA.

Класифікація ризиків постає ключовою умовою повноти та несуперечливості ідентифікації. Ієрархія Risk Breakdown Structure, використана як «словник» джерел невизначеності, забезпечує єдину термінологію, полегшує призначення власників і дає змогу порівнювати експозицію між проєктами. У проєктах, пов’язаних із IT та інтеграцією ШІ-сервісів, RBS особливо важлива для раннього виявлення технічних, правових та організаційних кластерів ризику, що впливають на архітектуру, дані й комплаєнс. Наявність узгоджених шкал і критеріїв прийнятності створює підґрунтя для коректної пріоритизації та резервування ресурсів.

Методи й моделі аналізу у розділі розглянуто як взаємодоповнювальний набір. Матриця «ймовірність × вплив» слугує швидким фільтром для первинного ранжування, проте її обмеження вимагають переходу до формальніших технік. Діаграма bow-tie наочно моделює баланс превентивних та пом’якшувальних бар’єрів навколо центральної події і дає картину прогалин контролів. Fault Tree Analysis дозволяє дедуктивно розкласти інцидент до базових причин і визначити мінімальні комбінації, достатні для відмови, тоді як Event Tree Analysis індуктивно формує сценарії наслідків з урахуванням успішності або відмови бар’єрів. Дерево рішень забезпечує прозоре порівняння альтернатив за очікуваною цінністю. У сукупності цей конвеєр переходить від якісного скринінгу до кількісного аргументування, що критично для узгодження дій із зацікавленими сторонами й економічного обґрунтування контрзаходів.

Специфіка машинобудівних проєктів, розглянута на прикладі виробництва техніки, посилює вимоги до безпеки, надійності та відповідності нормативам упродовж життєвого циклу виробу. Ієрархія зниження ризику за ISO 12100 задає пріоритет конструктивно безпечних рішень перед захисними

заходами та інформаційними попередженнями, що безпосередньо впливає на логіку проєктних і технологічних рішень. Висока інтегрованість цифрового ланцюжка CAD/PLM/ERP/MES/SCADA висуває на перший план кіберфізичні ризики OT-середовища; рекомендації щодо «глибинної оборони», сегментації IT/OT і демілітаризованих зон зменшують імовірність ескалації кіберінцидентів у фізичні наслідки. Роль IT-функції при цьому виходить за межі підтримки: це управління цілісністю даних та версій, організація безпечних ланцюгів постачання ПЗ і прошивок, керування ідентичностями й аудитом, що безпосередньо впливає на профіль ризику.

Отже, теоретичний фундамент розділу доводить: ефективний ризик-менеджмент вимагає, по-перше, інтегрованої методології (поєднання ISO 31000 і практик PMI) з чіткими критеріями прийнятності та відповідальністю; по-друге, структурованої класифікації джерел невизначеності (RBS) як основи повної ідентифікації та узгодженої комунікації; по-третє, послідовного застосування методів від якісних до кількісних (P×I, bow-tie, FTA, ETA, decision tree) для обґрунтованого вибору дій; по-четверте, урахування галузевої специфіки безпеки та OT/ICS-архітектур, що визначають як технічні, так і організаційні вимоги до процесів. Ці висновки задають рамку для подальших розділів роботи, у яких буде сформовано інтеграційну модель і практичні інструменти, показано їх впровадження у виробничому контексті та оцінено ефективність на основі вимірюваних показників.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ УПРАВЛІННЯ РИЗИКАМИ ДЛЯ ПРОЄКТУ ПІДПРИЄМСТВА

2.1. Аналіз поточного стану управління проєктами на підприємстві

Підприємство спеціалізується на виробництві жниварок та супутніх вузлів до самохідних комбайнів. Тип виробництва - серійно-потоковий із чіткою пооперaційною організацією: розкрій і різання металу, зварювання та механообробка, фарбування, вузлове складання, кінцева збірка й стендові випробування. Інженерний контур побудовано за принципом безперервного цифрового ланцюжка даних: конструкторська підготовка ведеться у CAD/CAE з керуванням конфігураціями в PLM; затверджені специфікації та виробничі маршрути передаються в ERP, далі у MES для диспетчеризації робіт і відстеження виконання; технологічне обладнання[23] інтегроване з SCADA/PLC, що забезпечує збір параметрів процесу та журналювання подій. IT-відділ відповідає за підтримку PLM/ERP/MES, мережеву інфраструктуру, сервіс користувачів і OT-безпеку [28], включно з сегментацією мереж[30], політиками віддаленого доступу та контрольованими «вікнами» оновлення програмного забезпечення контролерів. Такий поділ зон відповідальності дозволяє спиратися на дані з операційних систем для управління програмами розробки й запуску модифікацій виробу, узгоджуючи календарі проєктів із реальними вузькими місцями цехів [31].

Вибір канадського ринку сільськогосподарської техніки як референтного зумовлений кількома факторами. По-перше, саме цей ринок належить до числа найструктурованіших і найвідкритіших у світі в частині аналітичної звітності: асоціація AEM щомісяця публікує повні статистичні дані щодо обсягів реалізації, динаміки, запасів і сезонності, що робить можливим використання цих відомостей як достовірної основи для дослідження. По-друге, структура машинно-тракторного парку Канади є типовою для північних регіонів із подібним кліматом і циклічністю аграрних робіт, що дозволяє екстраполювати

тенденції попиту на техніку до умов українського ринку або аналогічних виробничих підприємств.

Крім того, канадський ринок є орієнтиром для багатьох виробників сільськогосподарської техніки через схожість технологічних вимог, високий рівень механізації та активну участь міжнародних брендів. Це забезпечує валідність використання цих даних у моделюванні сезонного навантаження на виробництво жниварок і плануванні ризиків постачання комплектуючих. Таким чином, аналітика AEM виступає не лише джерелом статистичних показників, а й індикатором тенденцій, релевантних для оцінки поточної та прогнозованої динаміки виробничих проєктів.

Для побудови базової картини попиту й сезонності використано офіційні дані галузевої асоціації AEM щодо канадського ринку сільськогосподарської техніки. На рис. 2.1 наведено фрагмент щомісячного звіту за липень 2025 року, де показані роздрібні продажі техніки та графік сезонної динаміки. За цим джерелом, у липні 2025 року в Канаді реалізовано 166 самохідних комбайнів, що на 11,4% більше, ніж у липні 2024-го; накопичувальний підсумок з початку року становить 1105 одиниць проти 1136 роком раніше (-2,7%). У документі також наведено початкові запаси на кінець місяця - 422 одиниці [11]. Ці показники використані для формування Таблиці 2.1 нашого дослідження; сама таблиця містить тільки релевантний нам рядок «Self-Prop Combines», оскільки він безпосередньо корелює з потребою в жниварках як укомплектуванні машин на ринку. Як зображено на рис. 2.1, графік демонструє чітко виражений сезонний пік восени, що відповідає календарю польових робіт і задає граничні умови для планування виробництва та логістики комплектуючих [11].

Таблиця 2.1.

Показники для самохідних комбайнів (Канада, липень 2025)

Категорія	Липень 2025, шт.	Липень 2024, шт.	% зміни (м/м)	YTD до липня 2025, шт.	YTD до липня 2024, шт.	% зміни (YTD)	Початковий запас, лип. 2025, шт.
Самохідні комбайни	166	149	+11.4%	1 105	1 136	−2.7%	422

AEM Canada Ag Tractor and Combine Report
July 2025

(Report Released 8/11/2025)

	July			YTD - July			Beginning Inventory
	2025	2024	%Chg	2025	2024	%Chg	Jul 2025
2WD Farm Tractors							
< 40 HP	1,211	1,139	6.3	8,192	7,845	4.4	8,699
40 < 100 HP	355	334	6.3	2,362	2,312	2.2	3,516
100+ HP	246	286	-14.0	1,754	1,987	-11.7	2,176
Total 2WD Farm Tractors	**1,812**	**1,759**	**3.0**	**12,308**	**12,144**	**1.4**	**14,391**
4WD Farm Tractors	**24**	**115**	**-79.1**	**625**	**597**	**4.7**	**152**
Total Farm Tractors	**1,836**	**1,874**	**-2.0**	**12,933**	**12,741**	**1.5**	**14,543**
Self-Prop Combines	**166**	**149**	**11.4**	**1,105**	**1,136**	**-2.7**	**422**

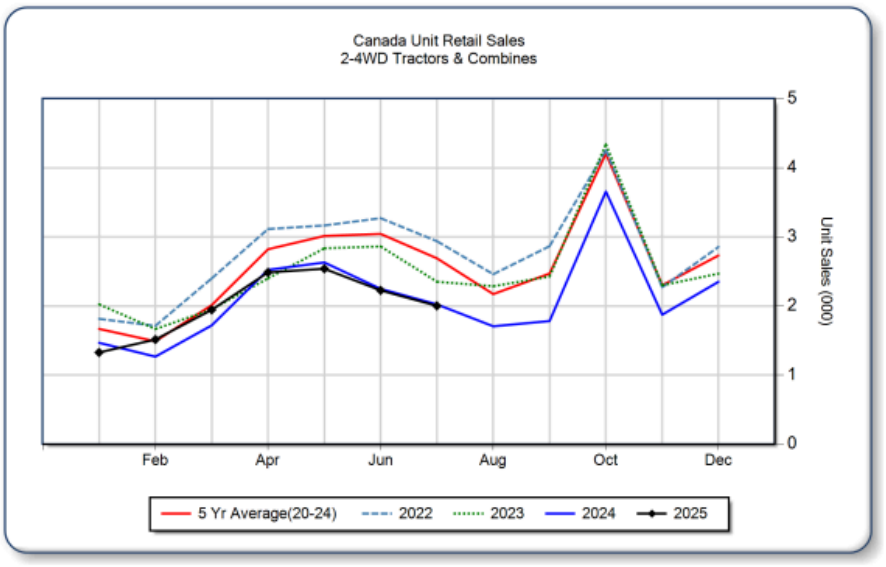

Рисунок 2.1 Фрагмент місячного звіту AEM для Канади (липень 2025): таблиця продажів і сезонний графік [11]

Сезонність попиту накладає специфічні вимоги на план-факт проєктного управління. Для підприємства це означає необхідність фронт-завантаження критичних активностей: своєчасного завершення конструкторських змін (ECN/ECO) і затвердження нових специфікацій, дострокового бронювання потужностей фарбування та випробувальних стендів, а також укладання рамкових договорів із ключовими постачальниками редукторів, гідрокомпонентів і електроніки. Інженерна підготовка виробництва має бути синхронізована з MRP-циклами ERP, щоб випередити осінній пік випуску, тоді як MES має забезпечити короткий час переналагодження та стійкість до

коливань партій. На рівні ІТ-відділу це трансформується в чіткі правила керування версіями специфікацій у PLM, SLA на доступність ERP/MES і політики відмовостійкого оновлення ПЗ стендів і контролерів, аби будь-яке обслуговування не припадало на пікові тижні.

На основі зібраної інформації і з урахуванням виробничої специфіки виконано ідентифікацію ключових ризиків. Для структурованості застосовано Risk Breakdown Structure, що віддзеркалює три великі класи: управлінські (планування, стейкголдери, постачання), зовнішні (регуляторні, ринкові, сезонні), технологічні та виробничі (конструкція, якість, процеси, OT/IT). Такий поділ зручний для прив’язки кожного ризику до конкретної точки процесу й відповідального підрозділу та для подальшого узгодження запобіжних і пом’якшувальних заходів. У нашій роботі RBS реалізовано як просту ієрархічну схему з можливістю розгортання гілок; її макет буде використано як «каркас» реєстру ризиків у наступних підрозділах.

Після первинної інвентаризації сформовано початковий перелік із двадцяти ризиків, що охоплюють вузли «ріжучий апарат/приводи», операційну готовність випробувальних стендів, синхронізацію специфікацій у PLM↔ERP, мережеву надійність OT/SCADA і доступність дефіцитних позицій електроніки. Для швидкого пріоритезування застосовано матрицю «ймовірність × вплив» із п’ятирівневими шкалами. На рис. 2.2 наведено композиційний вигляд цієї матриці з нанесеними ризиками; кожна позначка відповідає ідентифікатору з нашого переліку, а значення P та I визначалися експертно на підставі історії простоїв, результатів випробувань і доступної ринкової інформації щодо постачань.

Рисунок 2.2 Початкова матриця «ймовірність × вплив» для топ-20 ризиків у проєкті виробництва жниварок

Інтерпретація матриці показує три кластери. Перший - «червона» зона високого пріоритету - включає затримки постачання редукторів і дефіцит електронних компонентів, а також ризики відмов PLC/HMI на випробувальній дільниці. Для цих позицій P×I наближається до 16-20, що вимагає негайних превентивних заходів: альтернативних постачальників і страхових запасів для редукторів, специфікації дозволених замінників для електроніки та жорсткого керування версіями прошивок і програмним забезпеченням стендів. Другий кластер - «жовта» зона середнього пріоритету - охоплює коливання FTT на зварюванні рами, нестачу кваліфікованих наладчиків і ризики розсинхронізації BOM між PLM та ERP. Тут доцільні заходи процесної стабілізації: навчання та сертифікація персоналу, контрольні карти якості, додаткові перевірки узгодженості специфікацій і автоматичні гейти в процесі зміни виробу. Третій кластер - «зелена» зона низького пріоритету - містить, зокрема, ризики логістичних коливань фарбоматеріалів і похибок калібрування, для яких достатньо рутинних контролів та періодичного моніторингу.

Отримані пріоритети узгоджуються з ринковим контекстом, відображеним у звіті AEM. Осінній сплеск попиту, який простежується на часовій кривій, підсилює критичність вузлів довгого циклу постачання[38] - редукторів,

гідрокомпонентів та електронних модулів управління. Початкові запаси в 422 комбайни на кінець липня, зафіксовані у звіті, вказують на високу ціну простою виробничих ліній у межсезоння та на потребу в синхронізації графіків із партіями відвантаження комплектуючих [11]. Тому в рамках проєктного плану управління ризиками доцільно поєднати контрактні інструменти з постачальниками (рамкові угоди з порогами запасів, SLA на терміни) і технічні - політики дозволених замінників, повторне використання уніфікованих компонентів та ревізію конструкцій на предмет зниження залежності від окремих позицій.

Роль IT-відділу в цій конфігурації є визначальною. По-перше, саме IT забезпечує цілісність цифрового ланцюжка: коректну передачу специфікацій із PLM у ERP та їх автоматичне використання в MES; від цього безпосередньо залежить ризик розсинхронізації BOM і помилкових комплектувань[42]. По-друге, IT керує змінами в OT-середовищі: версіями прошивок PLC/HMI, політиками оновлень і тестовими стендами, що знижує ймовірність відмов під час приймальних випробувань. По-третє, разом із виробництвом IT забезпечує спостережність процесів через SCADA та телеметрію, надаючи дані для аналізу FTT, простоїв і причин відхилень, а також підживлюючи реєстр ризиків фактичними індикаторами[43].

Висновок цього підрозділу полягає в тому, що підприємство має зрілу цифрову інфраструктуру, достатню для даних-керованого управління ризиками, а ринкова сезонність і структура постачань формують «критичну трійку» ризиків: довгі ланцюги постачання силових вузлів, стійкість електроніки й випробувальних засобів та процесна стабільність зварювально-складальних дільниць. Застосована нами RBS дала зрозумілу рамку класифікації, а первинна матриця P×I окреслила чергу реагування на найближчі спринти.

2.2. Проєктування моделі управління ризиками

У межах цього підрозділу визначено методологічні засади та побудовано модель процесу управління ризиками для нашого програмного продукту. Вибір було зроблено на користь UML як універсальної мови моделювання[44], що дозволяє на різних рівнях деталізації зафіксувати вимоги, поведінку й взаємодію компонентів системи. BPMN розглядався як альтернатива для опису бізнес-процесів, однак у нашому випадку критичною була потреба відобразити не лише послідовність робіт, а й інтерфейсні «взаємодії з актором», внутрішні програмні об’єкти, зовнішні сервіси та сценарії обробки помилок. Тому основні результуючі дані створено у вигляді UML-діаграм прецедентів, послідовностей та діяльності; їхня сукупність задає цільову логіку платформи, яка отримує від GPT структуровану відповідь[32], перетворює її на візуалізаційні матеріали управління ризиками та зберігає в локальній БД для подальшої аналітики.

Концептуальна рамка моделі опирається на три ключові принципи. По-перше, «інтероперабельність» між користувачем, UI-клієнтом і зовнішнім GPT-сервісом має бути формалізована через контракт запиту-відповіді [34], де формат результату суворо визначений; саме це дозволяє автоматизувати побудову матриці «ймовірність × вплив», дерев відмов та рішень. По-друге, «трасованість» від запиту до графічних даних забезпечується збереженням первинних даних і вироблених обчислень у SQLite, що важливо для аудиту та повторного відтворення результатів. По-третє, «розмежування відповідальності» закріплює зони дії: користувач формує параметри предметної області, застосунок збирає і валідує дані, GPT виконує аналітичну генерацію в узгодженому форматі[37], модуль звітності візуалізує та експортує.

На рисунку 2.3 подано діаграму прецедентів системи. Модуль «Користувач» і «Керівник» представляють дві типові ролі доступу. Користувач взаємодіє з системою у межах прецедентів «Увійти/Зареєструватися», «Згенерувати аналіз ризиків (GPT)», «Побудувати графіки (P×I, Fault, Decision)» та «Експортувати звіт». Окремим стереотипом позначено зовнішній сервіс «GPT API»[33], що знаходиться поза межами системи, але входить у сферу інтеграційних взаємодій. Для керівника передбачено можливість перегляду й

експорту результатів; у реальних сценаріях це може означати як спільне використання одних і тих самих допоміжних матеріалів, так і накладання політики ролей для підпису звітів. Головна ідея діаграми полягає в тому, що система з самого початку орієнтована на повторне використання збережених запитів: користувач може у будь-який момент переключитися на попередній запит і побудувати нові візуалізації без повторного звернення до GPT.

У межах цього прецедентного поля важливо домовитися про межі «Системи». Прямокутник на рисунку 2.3 охоплює UI-клієнт, менеджер запитів, адаптер GPT, модуль побудови графіків і підсистему експорту. База даних і кеш відповідей розглядаються як ресурси системи; зовнішнім є лише GPT API. Цей поділ зменшує архітектурну невизначеність під час зміни постачальника LLM-моделі: контракт запиту/відповіді лишається власністю нашої системи, а конкретна реалізація постачальника - змінною.

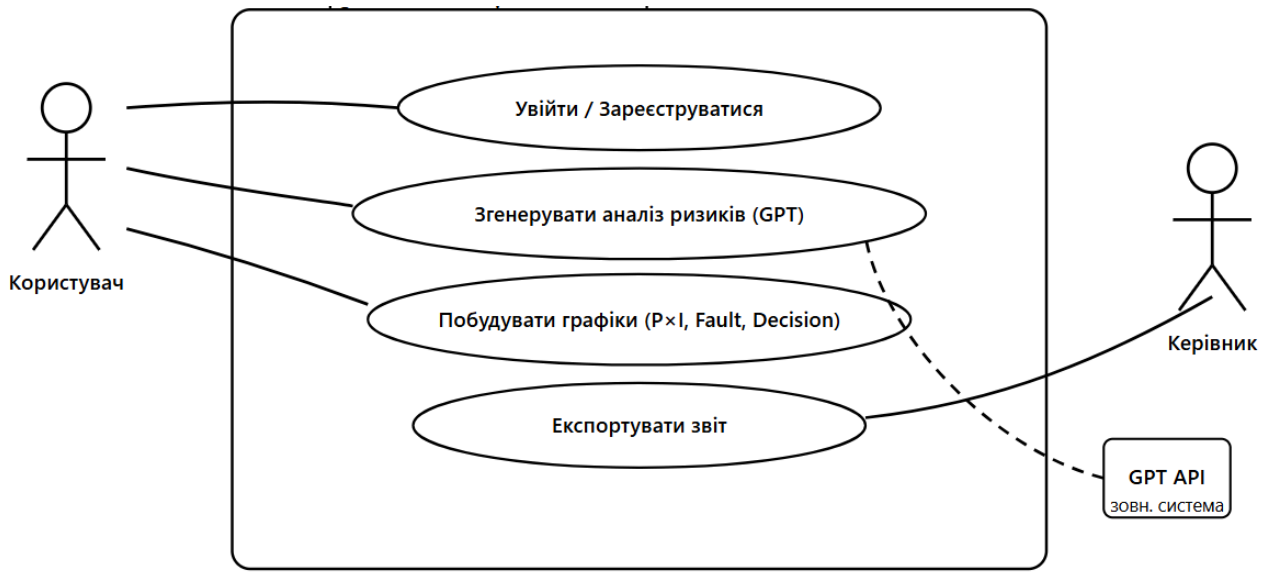

Рисунок 2.3 Діаграма прецедентів системи управління ризиками.

Поведінкова перспектива деталізована на рисунку 2.4, де наведено діаграму послідовностей. Ліфлайни «Користувач», «UI (App)», «GPT API», «SQLite», «Звіти/Графіки» демонструють наскрізний сценарій від аутентифікації до експорту. На стартовому етапі користувач ініціює вхід, а

застосунок верифікує облікові дані через БД. Відповідь «OK» відкриває форму параметризації: користувач обирає шаблон запиту та заповнює поля, які автоматично підставляються в промпт. Далі UI формує запит до GPT згідно з протоколом, що вимагає суворої JSON-схеми відповіді [32]. Після отримання результату система зберігає і запит, і відповідь у SQLite, фіксуючи службову інформацію для подальшого аудиту. Візуалізаційний модуль на базі matplotlib і pandas перетворює дані на візуалізаційні матеріали: матриця P×I, дерево відмов і дерево рішень. Користувач може сформувати і завантажити звіт у форматах XLSX/PNG. Діаграма підкреслює ідемпотентність стадій: якщо запит уже існує, користувач може повторно відтворити графіки без нового звернення до GPT, що зменшує вартість і час обробки.

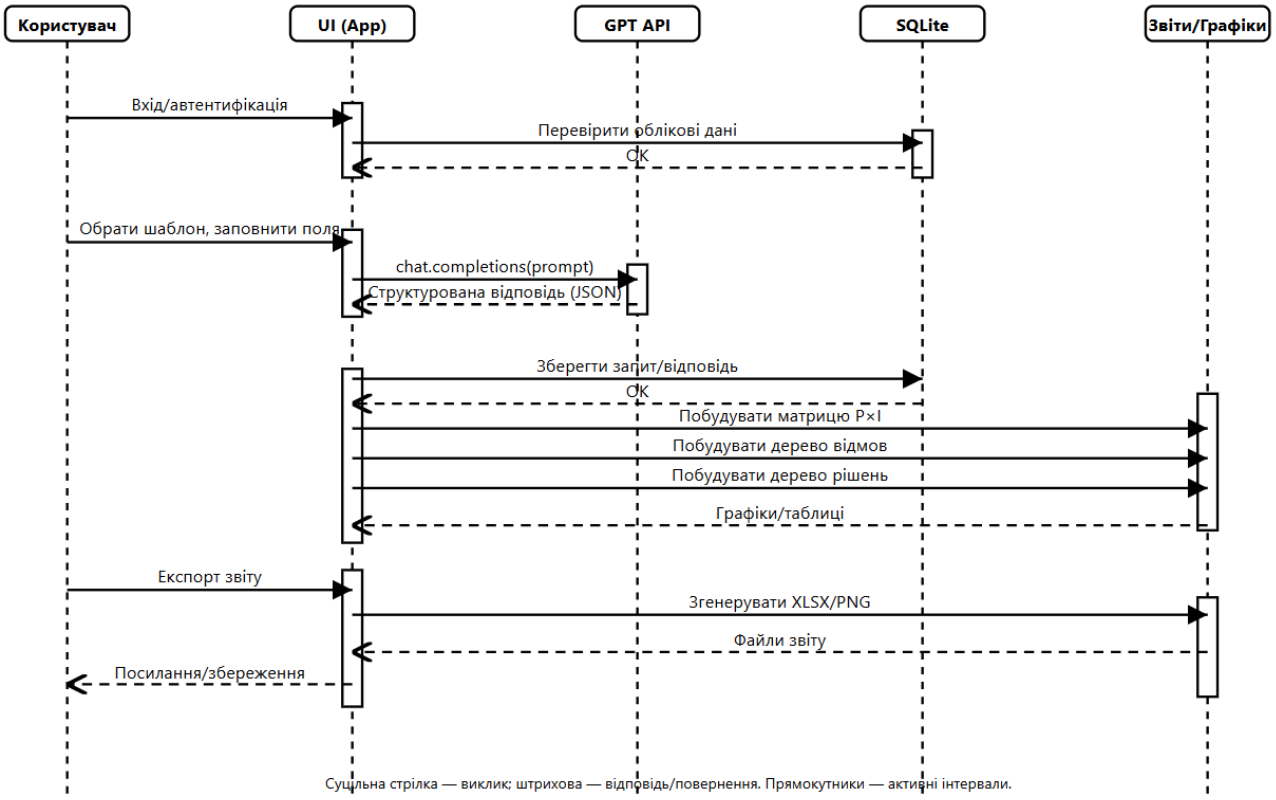

Рисунок 2.4 Діаграма послідовностей основного сценарію.

Особливу увагу приділено синхронної/асинхронній взаємодії. Всі стрілки в діаграмі послідовностей виконуються синхронно в межах одного сеансу

користувача, однак у реалізації закладено неблокуючий виклик GPT з індикатором прогресу. Це дозволяє зберегти чуйність інтерфейсу, а також коректно обробляти помилки мережі, тайм-аути[42] або перевищення квоти. Повернення до попередніх запитів моделюється як звернення UI до БД, після чого повторно активуються lifeline «Звіти/Графіки», не торкаючись «GPT API».

На рисунку 2.5 наведено діаграму діяльності з вертикальними доріжками відповідальності. Вона описує алгоритм керування ризиками в розрізі чотирьох зон: «Користувач», «UI (застосунок)», «GPT API», «БД і Звіти». Процес починається зі стартового вузла, переходить до перевірки облікових даних і має розгалуження «ОК?». Негативна гілка повертає користувача до повторного введення, позитивна - веде до вибору шаблону й формування промпта. Далі здійснюється виклик GPT і отримання відповіді з наступною перевіркою «Валідна?». У разі невідповідності схемі ініціюється цикл повідомлення/редагування; у разі відповідності дані зберігаються й активуються підпроцеси побудови візуалізацій. Завершальна фаза передбачає експорт аналітичних виходів системи і повернення результатів користувачеві. Діаграма діяльності відображає «логіку прийняття рішень» на рівні застосунку: будь-яка нестандартна ситуація (відсутність зв’язку, порожня відповідь, невалідний JSON) переводить процес у режим корекції з явним повідомленням. Таким чином, користувацький досвід лишається передбачуваним, а дані - консистентними.

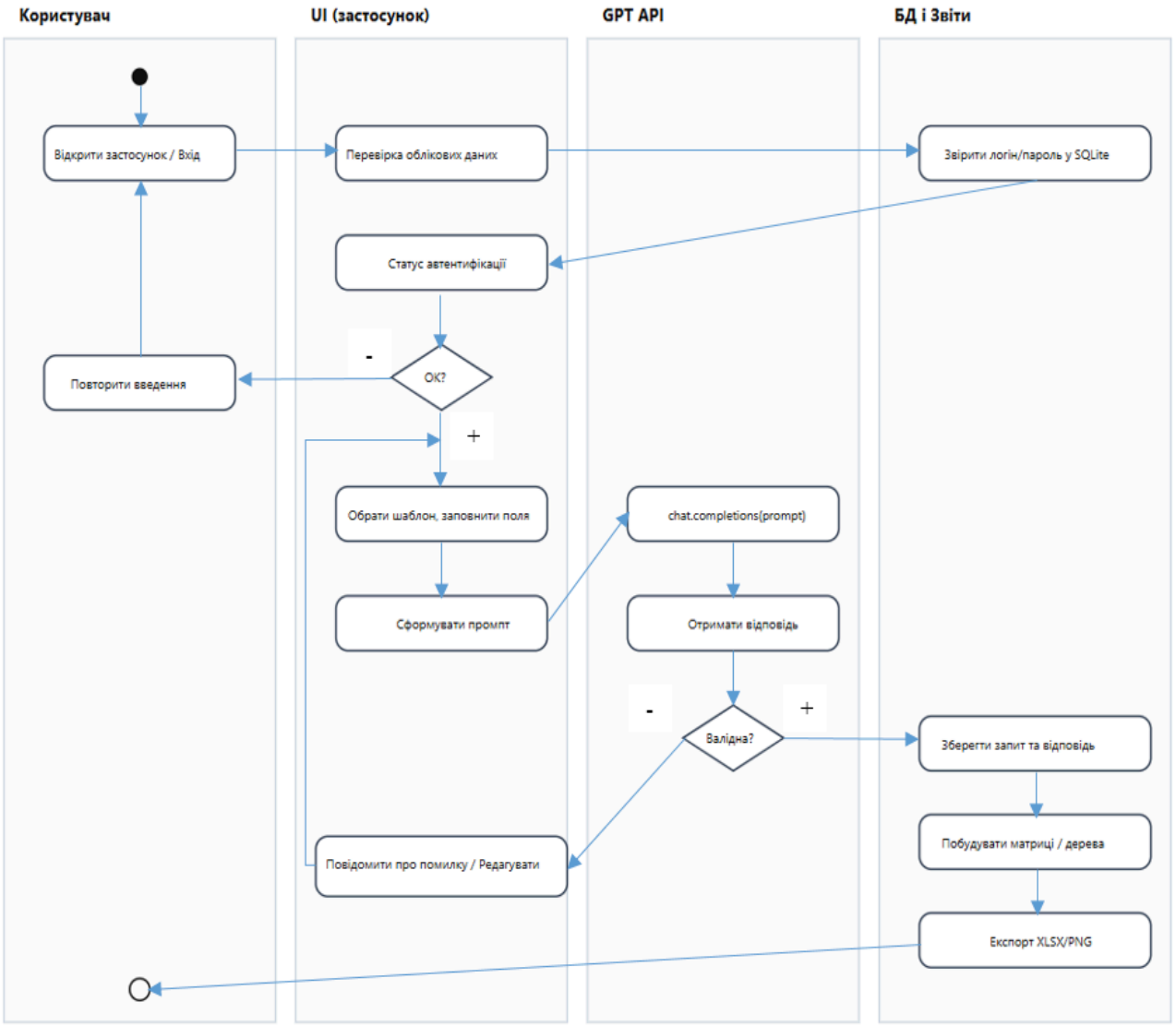

Рисунок 2.5 Діаграма діяльності з доріжками відповідальності

Алгоритм управління ризиками, зафіксований у цих діаграмах, передбачає стандартизовану структуру даних. Вхідні параметри прецизійно описують предметну область: назву проєкту, категорію продукту, стадію життєвого циклу, часовий горизонт і пороги класифікації ризиків. Шаблон промпта вимагає від GPT повернути об’єкти у вигляді JSON з чіткими ключами для списку ризиків, шкали ймовірностей та впливів, зв’язків для дерева відмов і вузлів для дерева рішень. Завдяки цьому побудова матриці P×I виконується механічно: вузли розміщуються на координатній сітці відповідно до значень P і I; колірні коди відповідають зонам пріоритету, а підписи - ідентифікаторам ризиків. Дерево відмов використовується для аналізу причинно-наслідкових зв’язків інциденту,

а дерево рішень - для порівняння стратегій мінімізації ризиків з урахуванням ймовірностей і очікуваних витрат/вигод.

З погляду вимог до якості реалізації UML-модель закладає кілька нефункціональних властивостей. Продуктивність забезпечується кешуванням відповідей і локальним рендерингом графіків. Безпека зосереджена на захисті облікових даних і контрольованому зберіганні токенів доступу; інтеграція з GPT ізолюється адаптером зі статичними обмеженнями на довжину промпта й максимальні витрати токенів. Аудит і відтворюваність гарантуються збереженням «сирих» відповідей і параметрів генерації. Практичність підтримується наявністю заготовлених шаблонів запитів, які дають можливість миттєво побудувати візуалізаційні матеріали навіть без звернення до зовнішнього сервісу.

Варто підкреслити роль «вузла валідації» в діяльнісній діаграмі. Розроблений алгоритм вимагає, щоб GPT повертала поля у строго заданому форматі; при найменшій невідповідності застосунок виводить повідомлення із зазначенням, яка саме секція не пройшла перевірку, і пропонує два шляхи: відредагувати параметри або застосувати демо-шаблон із коректною структурою даних. Це дозволяє не блокувати роботу користувача у випадках тимчасової недоступності сервісу або перевищення квоти. Така поведінка прямо відображена на діаграмі послідовностей: після невдалої взаємодії з GPT відбувається локальний перехід до формування графіків на тестових даних, що дає змогу продовжити моделювання ризиків без затримок.

Сформована UML-модель також враховує сценарії повторного використання. Коли користувач відкриває вкладку візуалізацій, список попередніх запитів підтягується з БД і доступний у випадачі. Вибір елемента ініціює відновлення контексту: параметри шаблону, первинна відповідь GPT, набори точок для матриці та структурні описання дерев. Жодної повторної взаємодії із зовнішнім сервісом не потрібно, доки користувач не вимагає «оновити» аналітику для змінених вихідних умов. Цей механізм знижує витрати

і створює прозору історію прийняття рішень - кожен графік та звіт однозначно пов’язаний з конкретним запитом.

З архітектурного погляду діаграми разом задають чіткий «скелет» підсистем. Адаптер GPT концентрує всю логіку зовнішнього виклику, включаючи побудову промпта, інтерпретацію відповіді і перетворення в об’єкти предметної області. Підсистема візуалізації оперує лише внутрішнім доменним форматам - координати для P×I, ребра і ворота для дерева відмов, вузли, гілки та ймовірності для дерева рішень; завдяки цьому вона незалежна від деталей зовнішньої моделі. Модуль звітності працює з «матрицею даних» pandas, куди зведені агрегати, та експортує як табличні, так і растрові представлення. БД SQLite зберігає користувачів, сеанси, запити, відповіді й метадані про побудовані графіки; типова транзакція охоплює запис запиту, збереження відповіді та індексацію ключових полів для швидкого пошуку.

Описані схеми UML виконують не лише роль документації, а й роль інженерного «контракту», на підставі якого реалізовано моноліт програми. Кожен крок на діаграмі діяльності має відповідний обробник у коді; кожне повідомлення на діаграмі послідовностей відповідає виклику або колбеку між модулями; кожен прецедент у діаграмі прецедентів має відбиток у меню чи кнопках інтерфейсу. Така відповідність «діаграма ↔ код» істотно спрощує супровід системи, адже будь-яка зміна вимог одразу відображається у відповідній діаграмі і лише потім переноситься в кодову базу.

2.3. Інтеграція IT-рішень у процес управління ризиками

Інтеграція інформаційних систем у ризик-менеджмент потрібна не лише для збирання сигналів про можливі відхилення, а й для швидкого прийняття рішень на основі узгоджених даних. Для підприємства, що виготовляє жниварки, до критичних джерел відносяться PLM (конструкторські дані та специфікації), ERP (постачання, виробниче планування, фінанси), MES/SCADA (показники виконання виробництва), QMS (якість і CAPA), CMMS/EAM [41]

(обслуговування та надійність). У межах проведеного дослідження роль «шлюзу ризиків» відіграє застосунок Risk-App, який формує шаблонізовані промпти до GPT-API, зберігає структуровані відповіді (SQLite), будує матрицю «ймовірність×вплив», дерево відмов і дерево рішень, а також віддає агрегати в дашборди чи менеджмент. Схему інтеграційного ландшафту наведено на рис. 2.6. Як зображено на рисунку, Risk-App розміщено в центрі потоку даних[37]: ліворуч - системи-джерела, праворуч - споживачі результатів (BI/DWH, управлінська звітність, канали нотифікацій). Горизонтально виділено служби автентифікації (IAM/AD) та транспорт (ETL/Message Bus), а нижче - GPT-API як зовнішній інтелектуальний сервіс. Суцільні стрілки позначають синхронний обмін через API, пунктир - пакетні/подійні інтеграції.

Конструкторські та технологічні зміни зберігаються в PLM і визначають склад виробу (BOM), вимоги до матеріалів і маршрутів. ERP перетворює ці вимоги на план закупівель та виробництва, відстежує ETA постачань, рівні запасів, дебіторку/кеш-флоу й відхилення собівартості. MES/SCADA забезпечує оперативні вимірювання: продуктивність, OEE, відмови обладнання, скрап та переробки; QMS - невідповідності, претензії, статуси CAPA; CMMS/EAM - WO, MTBF/MTTR і планово-попереджувальні роботи. Risk-App поєднує ці джерела у спільній моделі реєстру ризиків: кожен запис має атрибути джерела, категорії RBS, рівні PPP та III, власника, статус і посилання на первинні дані. Така структура дає можливість одночасно бачити «звідки прийшов сигнал» і «який вплив він створює на виробничу програму жниварок».

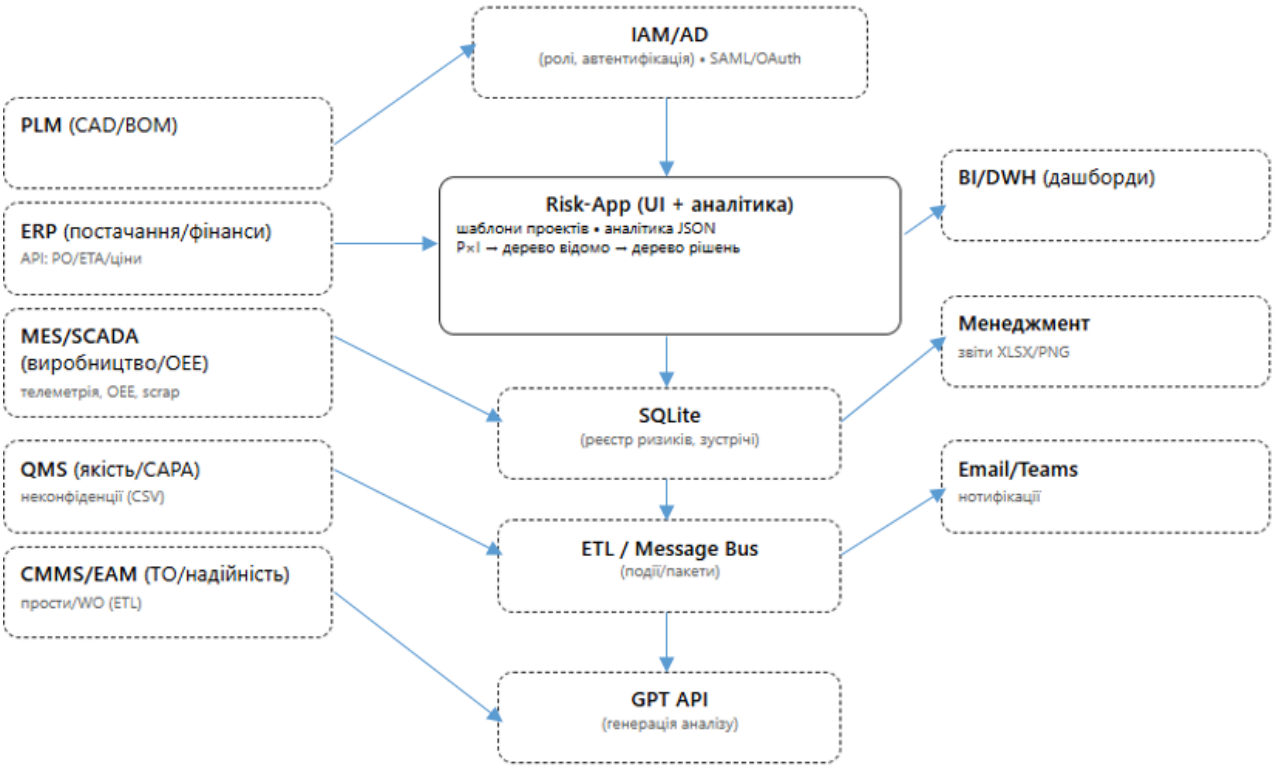

Рисунок 2.6 Інтеграційна схема «Risk-App» у виробничому ландшафті підприємства

З технічного боку використовуються два режими. Перший - синхронний API для подій, що потребують негайної реакції (наприклад, різка зміна ETA критичного постачання, збій у ланцюжку PLM→ERP→MES, аварійний сигнал SCADA)[28]. Другий - пакетний/подієвий обмін для масових даних (щодобові зведення закупівель, агрегати OEE/скрап, реєстри невідповідностей, журнали WO). У разі відсутності прямого API застосовується ETL або повідомлення в шину (Message Bus) з проміжним зберіганням у staging-зонах. Risk-App читає агрегати, формує промпт за затвердженим шаблоном, отримує від GPT-API структуровану відповідь (строго визначений JSON/CSV), валідує її, після чого оновлює реєстр ризиків і автоматично будує візуалізації.

Щоб забезпечити відтворюваність аналізу, схему реєстру спроєктовано мінімалістичною, але придатною для інтеграції. Ключові поля: RiskID, Title, RBS_Category, SourceSystem, EvidenceLink, P, I, Owner, Status, CreatedAt, NextReviewAt, MitigationPlan. Відповідники з джерел подано у табл. 2.2. У стовпці «Тригери» описано бізнес-правила, за якими автоматично

створюється/оновлюється запис ризику (наприклад, «ETA > план +7 днів» або «Scrap rate > 3σ протягом трьох діб»). Стовпець «Інтеграція» вказує засіб (API/ETL/CSV/подія), а «Періодичність» - частоту оновлення.

Таблиця 2.2

Мапінг даних до реєстру ризиків і правила створення записів

Система-джерело	Об’єкт/метрика	Приклад тригера	Інтеграція	Періодичність	Відповідальний
PLM	Зміна версії BOM, ECN	Новий ECN для вузла жатки, критична зміна матеріалу	API/ETL	Подієво (on change)	Інженер-конструктор
ERP	Замовлення постачання (PO), ETA, ціни	ETA критичного комплекту > план +7 днів; ↑ ціни > 8%	API	Щогодини (критичні позиції) / щодень (інші)	Планувальник MRP
MES/SCADA	OEE, Scrap rate, аварійні події	Scrap rate > 3σ протягом 3 діб; OEE < 60%	ETL/події	15 хв (OEE/alarms) / щодень (агрегати)	Майстер зміни
QMS	Невідповідність, CAPA	Нова NCR класу «Major» без CAPA > 48 год	CSV/API	Щодень	Інженер з якості
CMMS/EAM	WO, MTBF/MTTR	Позаплановий простій > 2 год на «вузлі-бутлнеку»	API/ETL	30 хв (критичні лінії) / щодень (агрегати)	Інженер-механік
Фінанси/ERP	Бюджет/вартість	Перевищення кошторису етапу > 10%	API	Щотижня	Фінансовий контролер
Зовн. джерела	Ринкові ціни, логістика	Індекс фрахту > порогу	ETL/API	Щотижня	Аналітик закупівель

Мапінг дозволяє одразу визначити «що, де і як підключати», а також хто відповідає за якість первинних даних. Це особливо важливо для подальшої автоматизації: якщо поріг-тригер перевищено, Risk-App створює або оновлює ризик, проставляє початкові P та I, ініціює нотифікації й формує рекомендації GPT-моделі.

Для забезпечення чіткого розподілу відповідальності під час ідентифікації, аналізу та реагування на ризики застосовується матриця RACI (Responsible, Accountable, Consulted, Informed). Вона дає змогу формалізувати

ролі учасників процесу ризик-менеджменту та уникнути дублювання функцій або «сірих зон» відповідальності.

На рисунку 2.7 зображено узагальнену RACI-матрицю відповідальності за управління ризиками у межах інтегрованої системи Risk-App. Вона демонструє, що безпосередню відповідальність (R) за створення й актуалізацію записів несуть інженери-конструктори, планувальники MRP та інженери з якості. Функція погодження (A) зосереджена у керівника виробництва, який затверджує контрзаходи та контролює виконання. Експертну (консультативну) роль (C) відіграють IT/OT-адміністратори та планувальники, які надають технічні дані для аналізу. Ролі, позначені (I), отримують сповіщення про події й оновлення станів ризиків.

Такий розподіл дозволяє забезпечити баланс між технічною, аналітичною та управлінською складовими. Кожна категорія користувачів має чітко визначений набір дій у Risk-App - від формування сигналу до підтвердження результатів виконання CAPA/WO. Завдяки інтеграції з IAM/AD ролі синхронізуються автоматично, що спрощує масштабування системи на інші виробничі дільниці або нові продуктові лінійки.

Роль / Завдання	Ідентифікація	Аналіз	Пріоритезація	Вибір контрзаходів	Затвердження	Моніторинг
Інженер-конструктор	R	C	—	—	—	I
Планувальник MRP	C	R	R	C	—	I
Інженер якості	C	R	C	R	C	A
Керівник виробництва	—	—	C	A	A	R
IT / OT адміністратор	I	C	—	C	—	R

R **Responsible** — виконує роботу / активність

A **Accountable** — несе відповідальність за результат та затвердження

C **Consulted** — надає експертизу, двосторонній зв'язок

I **Informed** — отримує сповіщення, односторонній зв'язок

Позначка «—» означає, що роль не залучена до конкретного завдання.

Рисунок 2.7 RACI-матриця відповідальності за управління ризиками

Інтеграційні потоки підтримують циклічний процес управління ризиками:

Detect - сигнал ідентифіковано за тригером (ERP/MES/QMS/CMMS);

Assess - Risk-App збирає контекст, формує промпт, отримує структуровану оцінку від GPT-API та розраховує R=P×IR=P\times IR=P×I;

Decide - автоматична пріоритизація, присвоєння власника, формування варіантів контрзаходів;

Act - створення задач у ERP/QMS/CMMS, оновлення статусів, нотифікації менеджменту;

Learn - зворотний зв'язок (фактичні результати, зниження PPP чи III), накопичення історії у SQLite і покращення шаблонів промптів.

Завдяки цьому Risk-App не лише реєструє подію, а й переводить її в керований процес, де кожній дії відповідає системний «тіньовий» об'єкт (заявка, CAPA, WO, зміна плану).

Щоб візуалізації будувались без ручного втручання, відповідь моделі повинна бути строго структурованою. У шаблоні промпта закріплено формат JSON/CSV з такими блоками:

а) risks [] - список ризиків з унікальними id, title, category, p, i, rationale;

б) fta - опис дерева відмов (вузли, зв'язки, типи AND/OR);

в) decision_tree - варіанти рішень із ймовірностями/наслідками;

г) actions[] - короткі контрзаходи з відповідальними й плановими датами.

Risk-App перевіряє схему (валідацію), а в разі відхилення - повертає кероване повідомлення користувачу з підсвіткою помилкового елементу. Схвалені дані миттєво потрапляють у реєстр і стають доступними для дашбордів.

Підприємство отримує єдиний інформаційний простір: інтерактивна матриця P×I із фільтрами (категорія RBS, цех, етап ЖЦ), дерево відмов для вузлів жниварки, дерево рішень для вибору контрзаходів, а також тренди R за періодами. Для менеджменту формується зведений звіт (XLSX/PNG) з топ-ризиками за очікуваними втратами, статусом реалізації планів (CAPA/WO),

таблицею змін рівнів P та I. Вивантаження можна надсилати e-mail або публікувати в BI. Завдяки збереженню структурованого JSON джерелом правди виступає SQLite, а BI-моделі читають ті самі агрегати, що й UI.

Інтеграція з IAM/AD забезпечує єдині ролі: інженери бачать свої RBS-категорії; планувальники - ризики постачання; керівники - агреговану картину. Всі події журналюються (audit trail у SQLite). Транспорт між системами шифровано (TLS), архіви даних резервуються. Для каналів OT/SCADA застосовується демілітаризована зона (DMZ) з односпрямованими конекторами, щоб уникнути впливу з IT-сегмента на технологічну мережу. Це узгоджується з вимогами виробничої безпеки і мінімізує операційні ризики від інтеграцій.

Пілотний етап доцільно виконувати на одній виробничій лінії й для 3-5 ключових метрик (ETA критичних комплектуючих з ERP, OEE та scrap із MES/SCADA, NCR із QMS). Далі додаються решта сигналів і інтеграційні сценарії: автогенерація CAPA/WO, зв’язок із бюджетами впливу (ERP), розширення набору візуалізацій. Важливо підтримувати регулярні ретроспективи - перегляд фактів і корекція правил тригерів та шаблонів промптів.

Запропонована інтеграційна архітектура перетворює розподілені дані PLM/ERP/MES/SCADA/QMS/CMMS на цілісний процес ризик-менеджменту, де кожен сигнал автоматично потрапляє в реєстр, отримує оцінку P та I, візуалізується і веде до конкретних дій у виробництві жниварок. Центральний застосунок Risk-App зменшує час від виявлення до реагування, знімає ручні рутинні кроки завдяки GPT-підказкам і забезпечує прозорість прийняття рішень завдяки узгодженим форматам даних, веденню журналу і єдиній моделі доступу. Схема на рис. 2.6 та мапінг у табл. 2.2 задають практичну «дорожню карту» для поетапного підключення систем і масштабування рішення на весь портфель виробів.

Висновки до розділу 2

У розділі 2 розроблено й обґрунтовано прикладну модель управління ризиками для підприємства з виробництва жниварок, яка спирається на реальні дані операційних систем і ринкові індикатори та інтегрує їх у єдиний цикл прийняття рішень. Аналітика поточного стану показала достатню зрілість цифрового ланцюжка CAD/PLM/ERP/MES/SCADA та наявність організаційних передумов для даних-керованого ризик-менеджменту. Використання статистики AEM щодо канадського ринку дозволило сформувати обґрунтовану сезонну рамку для планування: осінній пік попиту підсилює вразливість довгих ланцюгів постачання й випробувальних потужностей, що зумовило фокус моделі на вузлах «редуктори - гідрокомпоненти - електроніка», а також на стійкості стендів і процесній стабільності зварювально-складальних дільниць. Первинна матриця «ймовірність × вплив» окреслила чергу реагування, а RBS забезпечила повноту ідентифікації та прив’язку ризиків до відповідальних підрозділів.

Проєктування моделі у форматі UML сформувало чіткий інженерний контракт між користувачем, застосунком і зовнішнім GPT-сервісом: у прецедентах закріплено ролі й функції, у діаграмах послідовностей - наскрізний сценарій від параметризації до експорту, у діаграмі діяльності - правила обробки помилок і валідації. Ключовим рішенням стало застосування шаблонізованих промптів і суворої JSON-схеми відповіді, що дозволяє механічно будувати матрицю P×I, дерево відмов і дерево рішень, а також зберігати повний слід даних у локальній БД для аудиту та відтворюваності. Такий підхід мінімізує суб’єктивність описів, знижує витрати часу на підготовку звітів і забезпечує повторне використання вже отриманих результатів без зайвих звернень до зовнішнього сервісу.

Інтеграційна частина моделі обґрунтувала двошаровий режим обміну даними: синхронні виклики API для подій, що потребують негайної реакції, та пакетні/подієві інтеграції для масивів історичних та агрегованих показників. Сформовано мінімалістичну, але повну схему реєстру ризиків з атрибутами

джерела, категорії RBS, оцінок P та I, власника, статусу й посилань на первинні свідчення; мапінг із PLM/ERP/MES/QMS/CMMS переведено у набір практичних тригерів, що автоматично створюють або оновлюють записи. Впровадження RACI-матриці закріпило однозначний розподіл відповідальностей між виробництвом, якістю, постачанням та IT/OT, а інтеграція з IAM/AD гарантує узгоджені ролі, трасованість змін і відповідність політикам доступу. Передбачено вимоги безпеки для IT/OT-границі - DMZ, шифрування трафіку, журналювання - що мінімізує технологічні ризики інтеграцій і підтримує комплаєнс.

Сукупний ефект запропонованої архітектури полягає в перетворенні розрізнених сигналів на керований процес із повним циклом Detect - Assess - Decide - Act - Learn: події з операційних систем через тригери конвертуються у стандартизовані записи ризиків, отримують кількісну оцінку, пріоритизуються з урахуванням очікуваних втрат, переводяться у CAPA/WO та супроводжуються до підтвердженого результату. Очікувані вигоди для підприємства - скорочення часу від виявлення до дії, зменшення простоїв і браку за рахунок проактивних контрзаходів, підвищення прозорості прийняття рішень і якості комунікації зі стейкголдерами. Пілотна стратегія на одній лінії та обмеженому переліку метрик дає змогу швидко продемонструвати результат і підготувати безболісне масштабування: перехід від локального SQLite до серверного сховища, розширення набору інтеграцій і підключення BI-вітрин без змін предметної моделі.

РОЗДІЛ 3.
ОЦІНКА ЕФЕКТИВНОСТІ ТА РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ МОДЕЛІ

3.1. Верифікація та тестування моделі управління ризиками

Мета верифікації - підтвердити, що розроблена модель та програмний інструмент забезпечують відтворюваний алгоритм виявлення, оцінювання та пріоритизації ризиків, а також дозволяють оперативно будувати візуальні подання[34] (матриця «ймовірність × вплив», дерево відмов, дерево рішень) і формувати матеріали аудиту у вигляді таблиць та експортів. Тестування проводилося на сценаріях виробництва жниварок, що відповідають предметній області кваліфікаційної роботи, з використанням підготовлених шаблонів запитів та уніфікованого формату відповіді GPT-API [33] (JSON-структура з полями risk_items, pxi, fault_tree, decision_tree, registry).

Першим кроком перевірено коректність створення користувачів, входу та доступу до історії запитів. На рисунку 3.1 показано успішну реєстрацію користувача та подальшу автентифікацію в застосунку[42]. Механізм авторизації є передумовою персоналізованого трекінгу кейсів: кожен користувач бачить власні запити, їхні результати, графіки та експорти.

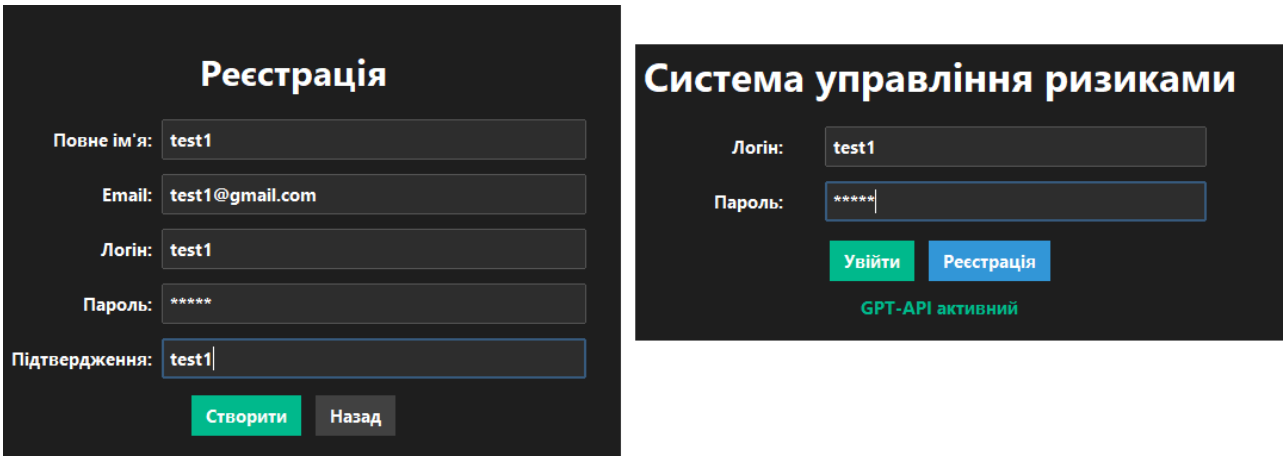

Рисунок 3.1 Реєстрація користувача та вхід до системи.

Далі валідували роботу «конструктора промптів». Він забезпечує введення виробничого контексту через поля «Продукт», «Тип виробництва», «Потужність», «Дільниці/цехи», «Цифровий ланцюжок», «Критичні вузли», «Постачання/LT», «Вузькі місця» та інші. На рисунку 3.2 показано приклад сформованого структурованого запиту та отриманої від GPT відповіді у форматі JSON, що відразу проходить синтаксичну перевірку. Важливо, що обраний формат не допускає довільних текстових вставок[33]: поля та типи жорстко фіксовані, завдяки чому подальша обробка для графіків та таблиць відбувається автоматично без ручних правок.

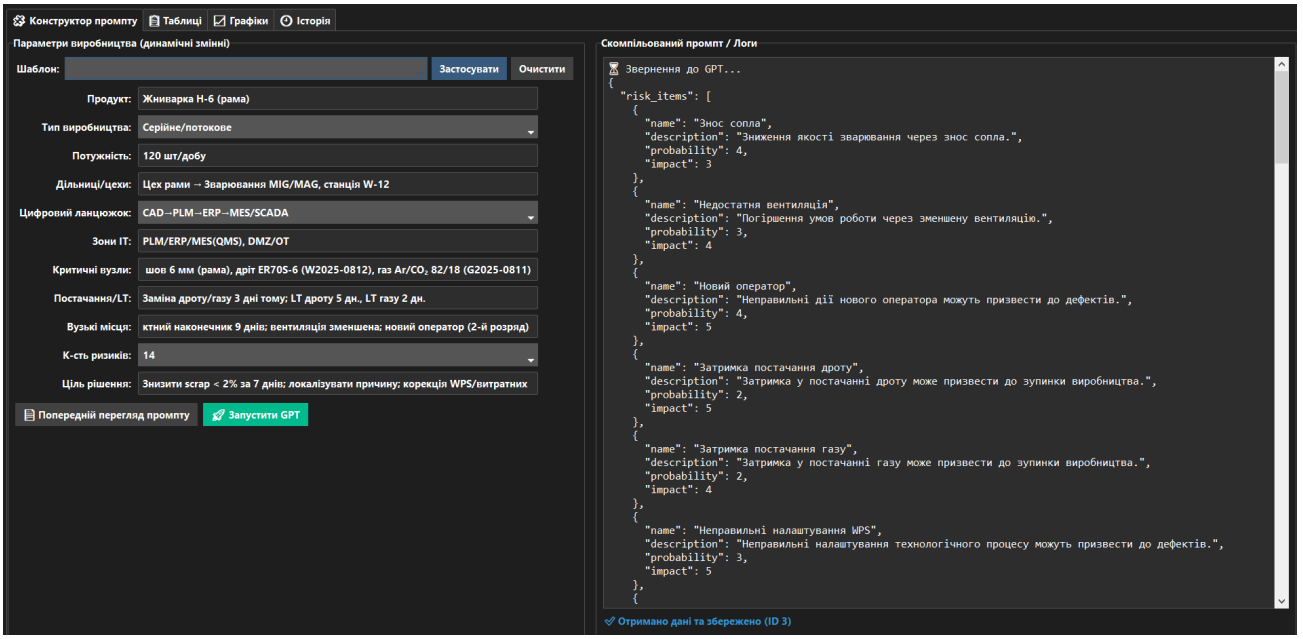

Рисунок 3.2 Конструктор промптів і приклад структурованої відповіді GPT у форматі JSON.

Якісну частину верифікації виконано на базі побудованих діаграм. На рисунку 3.3 наведено дерево відмов (Fault Tree) для сценарію різкого зростання скрапу під час зварювання; центральна подія «Дефекти зварювання» розкладається на причинні гілки, що відображають типові технічні та процесні відмови (керування WPS, знос сопла, проблеми з контролем якості, вентиляція, помилки налаштування, людський фактор). Діаграма підтвердила, що модель

коректно використовує структуру «AND/OR»[25] для причинних зв’язків і виводить перелік базових подій, які можна трансформувати в дії з мітигації.

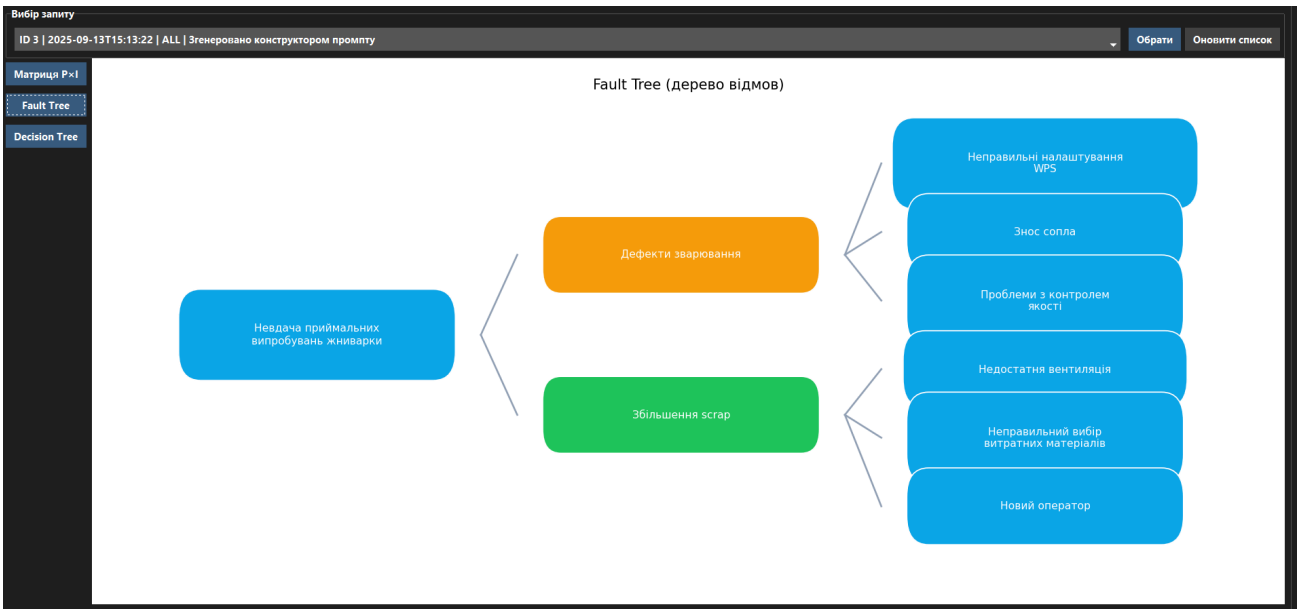

Рисунок 3.3 Fault Tree (дерево відмов) для сценарію зростання скрапу під час зварювання.

Наступним етапом перевірено дерево рішень (Decision Tree) - рисунок 3.4. Воно показує альтернативи зниження скрапу, включно з корекцією технологічного процесу, підвищенням контролю якості та усуненням першопричин дефектів. Гілки містять імовірності та очікувані ефекти (EV), що дозволяє порівнювати варіанти втручання не лише якісно, а й кількісно. Таким чином, інструмент підтримує аналітику «що-якщо»[37], необхідну для обґрунтованих управлінських рішень.

Правильність формування контексту підтверджено окремо (рисунок 3.5), де видно, що поля введення відображають виробничу специфіку (SPOF-обладнання, критичний ЗІП, відставання ППР тощо)[14]. Це важливо для відтворюваності: при однаковому контексті програма стабільно повертає однотипну структуру відповідей і порівнювані результати.

Рисунок 3.4 Decision Tree (дерево рішень) з альтернативами зниження скрапу та очікуваними ефектами.

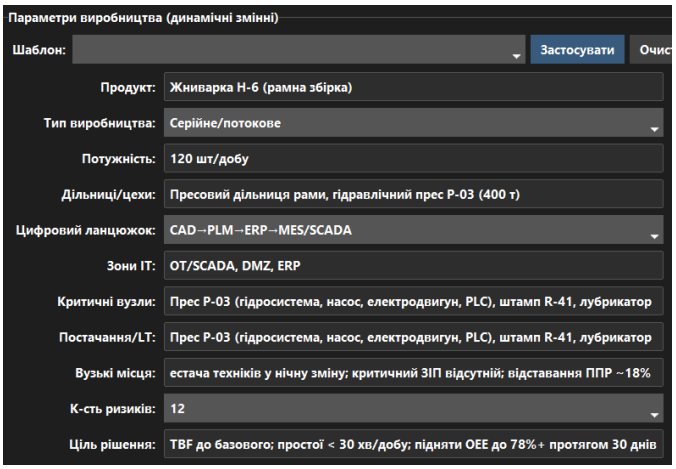

Рисунок 3.5 Приклад заповнення виробничого контексту та параметрів тесту.

Для перевірки критеріїв аудиту виконано експорт реєстру ризиків у формат XLSX/CSV (рисунок 3.6). Експортована таблиця містить ключові колонки id, title, probability (P), impact (I), category, owner, notes і може бути імпортована в ERP/BI-системи[41] або використана для проведення перехресних перевірок у підрозділах. Це підтверджує інтеграційну готовність рішення до звичних офісних і корпоративних інструментів.

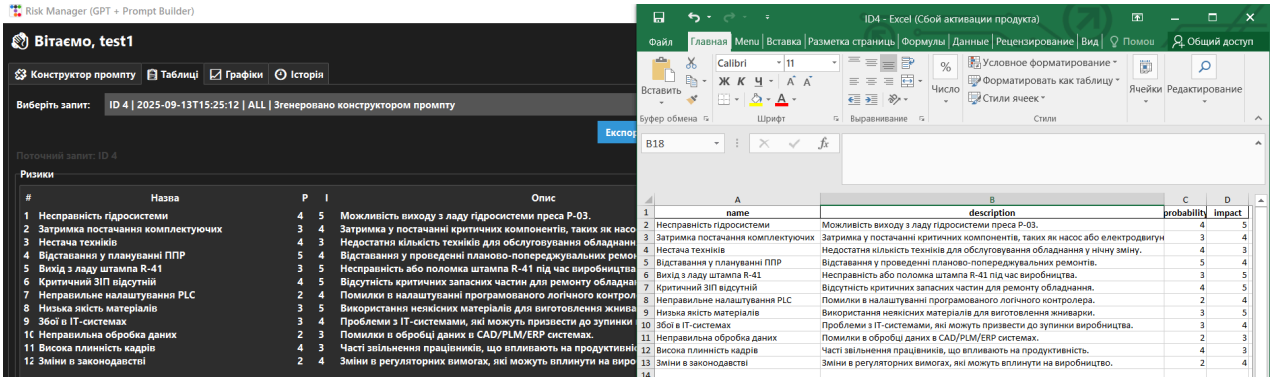

Рисунок 3.6 Експорт реєстру ризиків у форматі XLSX/CSV (pandas-таблиця).

Окремо верифіковано модуль побудови матриці «ймовірність × вплив»[4] (рисунок 3.7). Точки на полі матриці відповідають окремим ризикам із реєстру; правий верхній сектор позначає високий пріоритет мітигації. Візуалізація супроводжується нумерованою легендою, що унеможливлює неоднозначність інтерпретації. За результатами тесту переконливо видно, що ризики, пов’язані з технологічною надійністю та якістю зварювання, мають вищий вплив, тоді як постачання ЗІП за наявності буферів - нижчий.

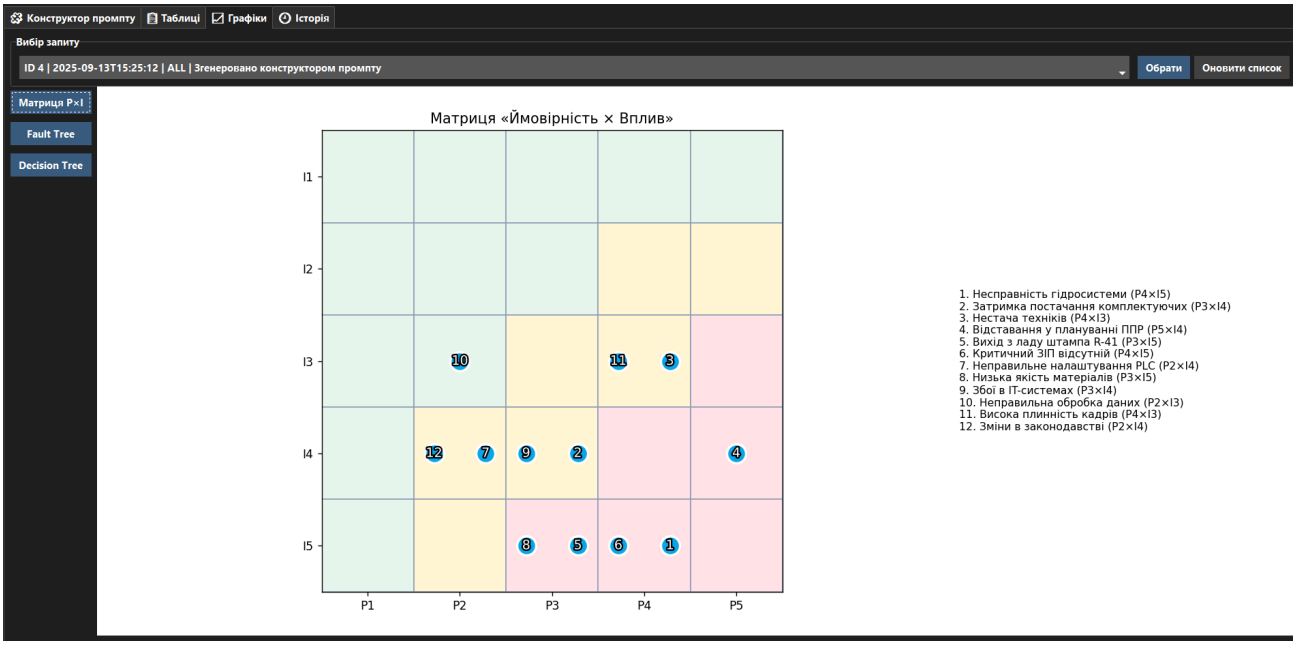

Рисунок 3.7 Матриця «Ймовірність × Вплив» з пріоритетами мітигації.

Стійкість до збоїв показано на прикладі недоступності або обмеження квоти GPT-API. На рисунку 3.8 зафіксовано повідомлення про помилку з

переходом до DEMO-режиму. Така поведінка дозволяє продовжити демонстрацію і тест кейсів, не блокуючи роботу користувача, а також дає змогу регламентувати дії при відмові зовнішнього сервісу (повтор, кеш, ручна обробка).

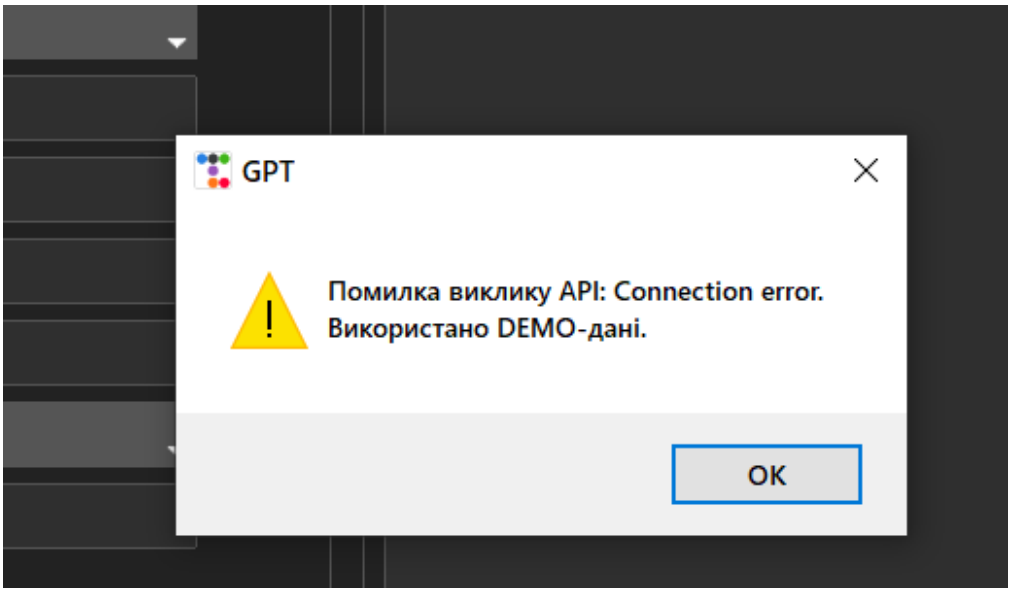

Рисунок 3.8 Повідомлення про помилку зовнішнього сервісу та перехід у DEMO-режим.

Завершальним кроком було підтвердження відстежуваності операцій у журналі. На рисунку 3.9 показано історію запитів із часовими мітками та режимами. Наявність історії забезпечує прозорість змін, відповідальність власників ризиків і відтворюваність аналітики.

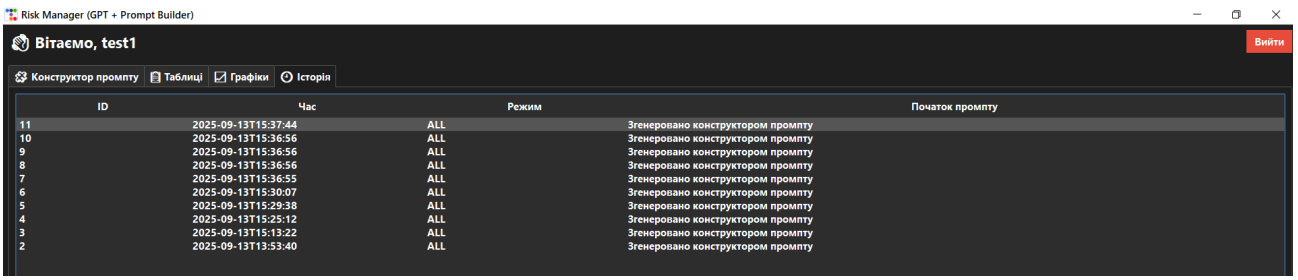

Рисунок 3.9 Журнал (історія) запитів користувача з часовими мітками.

Для кількісної оцінки ефективності використано три показники. По-перше, сукупний ризиковий вплив SRI, який розраховується як сума зважених добутків

$$SRI=\sum w_k \cdot P_k \cdot I_k, \quad (3.1)$$

де w_k - ваги категорій ризиків, визначені політикою підприємства, P_{k-}, I_{k-}.
По-друге, операційний ефект за часом - зменшення тривалості циклу аналізу завдяки автоматичній генерації структури ризиків і графіків (оцінюється за витраченим часом від запуску до отримання візуалізаційних матеріалів). По-третє, фінансовий ефект - очікуване зниження втрат через пріоритизовані дії з мітигації, що обчислюється на основі EV із дерева рішень та питомої вартості простоїв/бракованої продукції.

За матеріалами наданих тестів отримано повний цикл: від введення контексту та збереження кейсу до побудови трьох ключових візуалізацій і формування реєстру. На практичних прикладах зварювальної дільниці та критичного пресового обладнання підтверджено, що модель коректно відокремлює технічні, процесні та постачальні ризики, розміщує їх у матриці за очікуваним профілем і пропонує логічні гілки причин та рішень. Експорт у таблиці та лог подій формують доказову базу для аудиту, а обробка помилок - необхідний рівень стійкості.

З урахуванням результатів верифікації можна вважати, що розроблена модель і програмний інструмент задовольняють вимоги кваліфікаційної роботи: забезпечують структуроване виявлення ризиків, наочну пріоритизацію, підтримку ухвалення рішень та інтеграцію з процесами управління на підприємстві. Подальші експерименти можуть бути спрямовані на калібрування ваг w_k під конкретні KPI (OEE, MTBF/MTTR, Scrap Rate), а також на автоматичне порівняння «до/після» мітигаційних заходів для розрахунку фактичної дельти SRISRISRI і фінансового ефекту.

3.2. Рекомендації щодо впровадження моделі в діяльність підприємства

Впровадження моделі управління ризиками доцільно організовувати як безперервний виробничо-організаційний цикл, у якому ідентифікація, оцінювання, пріоритизація, планування, виконання та моніторинг формують єдину «колію» даних і рішень. Запропонований прикладний інструмент Risk-App виконує функцію центру цього циклу: він дисциплінує введення виробничого контексту у стандартизовані поля, перетворює його на уніфікований запит до GPT-API, одержує відповідь у суворому JSON-форматі[33], автоматично будує матрицю «ймовірність × вплив», дерево відмов і дерево рішень, а також підтримує локальний реєстр із повною історією змін і експортом для аудиту. Така побудова дає два ключові ефекти. По-перше, вона зменшує управлінське тертя між підрозділами: виробництво, якість, постачання та обслуговування обладнання працюють з одним і тим самим матеріалом - записом ризику, який має назву, власника, термін та міру пріоритету. По-друге, вона робить результати вимірюваними: кожна зміна параметрів P та I негайно відображається у матриці й підсумковому індексі, а графи дерева рішень дозволяють зіставляти очікувану вартість альтернатив, фіксуючи економічний сенс дій.

Практичне розгортання починається з чіткої ролі власника процесу, координатора ризиків, аналітика Risk-App, відповідальних осіб цехів і служби IT/ІБ. Легітимований розподіл ролей знімає типові суперечки про «хто за що відповідає» і забезпечує стабільну рутину. Методика оцінювання, шкали і ваги затверджуються власником процесу; координатор проводить сесії виявлення ризиків і стежить за строками оновлення; аналітик підтримує шаблони промптів, перевіряє валідність JSON-відповідей і формує звіти; виробництво, якість і постачання є власниками конкретних записів і виконують мітигаційні дії; IT/ІБ забезпечують доступи, резервування і інтеграції. Саме у такому логічному каркасі Risk-App розкриває свою цінність: уніфікований інтерфейс зменшує помилки введення, а жорстка структура відповіді унеможливлює «вільні художні» тексти, що руйнують обробку.

Технічно пропонується зберегти для пілоту просту архітектуру з локальним SQLite, щоб мінімізувати інерцію і швидко продемонструвати результат, а під час масштабування перейти на серверне PostgreSQL без змін предметної моделі. Вхід даних організовується змішаним способом: щоденні агрегати та історичні зрізи з ERP/MES/QMS/CMMS надходять у вигляді CSV/XLSX-пакетів, а подієві сигнали (аномалія scrap, відмова вузла, простій понад поріг) доставляються через легку шину повідомлень і автоматично створюють чернетки ризиків у реєстрі. Зворотні інтеграції реалізуються як експорт таблиць і знімків діаграм у BI/DWH для порівняння KPI у динаміці, а канали сповіщень (ел. пошта чи командні месенджери) інформують власників ризиків про перетин порогів IP×I, прострочені дії або появу нових записів. Наявність єдиного джерела правди і контрольованої «вітрини» у BI дисциплінують процес і спрощують захист рішень на виробничих нарадах.

Питання якості даних є центральним для надійності моделі. Канонічний запис має містити ідентифікатор, назву, категорію, підсистему або вузол, власника, оцінки P і I у шкалі 1-5, інтегральний добуток, статус і строк, опис плану мітигації та джерело появи (PLM, ERP, MES, QMS, CMMS, GPT). Обов’язковість полів «власник» і «строк» усуває «нічийні» ризики, а узгоджені шкали гарантують коректне відображення на матриці. У Risk-App доцільно ввімкнути ввічливу автоматичну валідацію: при неповних або двозначних даних запис не приймається, натомість пояснюється помилка і пропонується виправлення. З перших днів визначаються політика зберігання історії, режим резервного копіювання і процедура відновлення, щоб аудит і розслідування інцидентів не залежали від людської пам’яті.

Експлуатаційна рутина вибудовується як безперервний PDCA-цикл. Виявлення ризиків відбувається на регулярних сесіях, де майстри змін, технологи і фахівці з якості вносять факти і припущення у параметризовані форми; аналітик формує одноманітні запити до GPT-API і перевіряє структуру відповіді; система автоматично будує матрицю «ймовірність × вплив», дерево відмов і дерево рішень, що одразу робить видимими вузли підвищеної уваги.

Плани мітигації затверджуються у вигляді коротких, але конкретних дій, система стежить за строками і статусами, а підсумкові наради розглядають агрегати реєстру і зміну індексу. Важливо, що всі кроки фіксуються в одному місці; відсутні «паралельні правди» у вигляді приватних таблиць або листувань, які руйнують відтворюваність. Коли в MES/SCADA з'являється сигнал аномального простою або у QMS формується сплеск відходів, тригер створює новий запис, призначає чергового власника та відсилає сповіщення; середовище переходить від реактивної до проактивної поведінки.

Пілот доцільно розпочинати на вузлах із найбільшим операційним впливом - зварювальна і пресова дільниці рами. Перші тижні потрібні для формалізації шкал і довідників та для навчання користувачів працювати з шаблонами; середина циклу присвячується налаштуванню обмінів і відпрацюванню рутини; завершення - приймальним випробуванням із зафіксованими критеріями успіху. Результат має бути очевидним і вимірюваним: час підготовки звіту скорочується з днів до годин, частка браку зменшується після мітигації, простої критичного обладнання скорочуються через пріоритизацію за критерієм впливу. Після доведення ефекту система масштабується на інші дільниці з міграцією бази на сервер, а словники ризиків уніфікуються по підприємству. Усі інтеграції закріплюються формальними SLA, щоб цикл не руйнувався при зміні людей або технологічних ініціатив.

Безпека і відповідність вимогам ІБ не повинні перетворюватися на бюрократичний бар'єр, проте мають залишатися обов'язковими для виконання. Єдина автентифікація з багатофакторним захистом, рольова модель доступу з принципом мінімальних привілеїв, сегментація мереж і шлюзи між IT та OT, шифрування даних у русі та на зберіганні, ведення записів в журналі CRUD-операцій і переглядів формують повноцінний аудиторський слід. Доцільно передбачити деградаційні сценарії зовнішнього сервісу: у випадку недоступності GPT-API або перевищення квот Risk-App переходить у DEMO-режим із кешованими відповідями, не блокуючи процес, але фіксуючи інцидент у журналі та повідомляючи користувача про обмеження.

Економічний сенс і ефективність моделі зручно демонструвати на стислих числових прикладах «до/після», без відриву від практики. Інтегральний ризиковий індекс визначається як SRI=100·∑wi·(Pi/5)·(Ii/5) де Pi та Ii - оцінки за шкалою 1-5, а wi - ваги за політикою підприємства. Якщо для трьох ризиків до мітигації маємо (P,I)=(4,4),(3,5),(2,4) за рівних ваг, то доданки становлять 0,64; 0,60; 0,32, SRIдо=156. Після виконання дій (3,3),(2,4),(1,3) одержуємо 0,36; 0,32; 0,12 і SRIпісля=80, тобто скорочення на 48,7 відсотка. Для постачання ключового вузла очікувані втратидоцільно рахувати як EL=p·C. Якщо ймовірність зриву 30 %, а вплив 30 тис. дол., то ELбаз=9 000; резервний постачальник за фіксовану плату 3 тис. дол. знижує ймовірність до 10 %, тоді очікувана вартість альтернативи з мітигацією EVміт=3 000+0,10·30 000=6 000дол. економія 3 тис. дол., а ROI=(9 000−6 000)/3 000=100%.

У виробничих одиницях приклад зі зварюванням виглядає так: за випуску 2 400 виробів на місяць і scrap 4,5 % утворюється 108 відходів; після коригувальних дій 2,5 % - 60, різниця 48 виробів, що за прямої собівартості 400 дол. дає 19 200 дол. економії; фактор «Quality» в OEE зростає з 95,5 % до 97,5 % і підтягує загальний показник. Для надійності преса скорочення простою з 2,0 до 0,8 години щодоби за 20 робочих днів дає 24 години; за продуктивності 5 виробів на годину і маржинальній віддачі 150 дол. одержуємо 18 000 дол. місячного ефекту; разові витрати 6 000 дол. повертаються із запасом, а «Availability» разом з «Quality» піднімають OEE на кілька пунктів. Часовий ефект зручно подавати як ΔT=Tдо−Tпісля\Delta і переводити у грошовий еквівалент через ставку вузького місця; за 750 дол. годинної маржі (5 виробів × 150 дол.) зекономлені 24 години дорівнюють зазначеним 18 000 дол., що повністю узгоджується з попередніми обчисленнями.

Організаційна зрілість процесу фіксується не лише описами, а й ролями, які щоденно виконують конкретні дії та несуть відповідальність за артефакти. Для зручності наведено узагальнену RACI-матрицю, яка легко імпортується у корпоративні регламенти і не потребує спеціальних адаптацій інструмента.

Перший блок рішень стосується організаційної моделі. Процес потребує легітимованого власника, координатора ризиків, аналітика, який працює безпосередньо з Risk-App, та представників виробництва, якості й постачання - власників окремих ризиків. Формальна матриця ролей RACI забезпечує однозначність відповідальності за методику, дані і строки. У табл. 3.1 наведено базовий розподіл функцій: власник процесу затверджує методику та шкали, координатор модерує сесії і відповідає за регулярність циклу, аналітик підтримує шаблони промптів, перевіряє валідність JSON-відповідей GPT та формує звіти, а IT-адміністратор відповідає за єдину автентифікацію, резервне копіювання, контроль доступу та інтеграційні адаптери. Важливо, щоб кожен запис реєстру ризиків мав визначеного власника, цільову дату і статус; саме ця простота дисциплінує процес не гірше за складні регламенти.

У процесі впровадження інтегрованої моделі управління ризиками можливе виникнення низки бар'єрів, пов'язаних із технологічними, організаційними та поведінковими аспектами. До технологічних належать відсутність уніфікованих форматів даних, різнорідність інформаційних систем (ERP, MES, QMS, CMMS), а також обмежений досвід роботи з GPT-API у промислових середовищах. Для їх подолання слід передбачити етап узгодження довідників і структур даних, створення конекторів із проміжним буфером (staging), а також поступове навчання аналітиків Risk-App із демонстраційними сценаріями.

Організаційні бар'єри пов'язані з розподілом відповідальності та культурою взаємодії між підрозділами. Типовою є ситуація, коли ризики розглядаються у «власних» таблицях або на рівні цехів, що призводить до втрати узгодженості. Для подолання цієї проблеми пропонується закріпити власника процесу наказом по підприємству, утворити координаційну групу ризик-менеджменту, а на початковому етапі - проводити спільні воркшопи із заповнення реєстру та інтерпретації матриці P×I.

Поведінкові бар'єри полягають у недовірі до автоматизованих рекомендацій або у страху втрати автономності прийняття рішень. Тут важливо

демонструвати, що GPT-модель не замінює експерта, а навпаки, підсилює його аналітично, скорочуючи рутину. Для підвищення прийняття системи варто публікувати приклади успішних кейсів («до/після») та забезпечити прозору методику перевірки точності результатів.

Таблиця 3.1

RACI та основні артефакти процесу

Етап/Артефакт	Власник процесу	Координатор ризиків	Власники ризиків (виробництво/якість/постачання)	Аналітик Risk-App	IT/ІБ адміністратор
Політика і шкали P,IP,I, ваги	A/R	C	C	C	C
Шаблони промптів, довідники вузлів/категорій	C	A/R	C	R	C
Сесії виявлення ризиків	A	R	R	C	-
Оцінка і пріоритизація P×IP\times I	A	R	R	C	-
Дерева відмов/рішень, сценарії	C	R	R	R	-
Реєстр, звітність, аудит	A	R	R	R	-
Інтеграції PLM/ERP/MES/QMS/CMMS, BI/DWH	C	C	C	C	A/R
Доступи, SSO, резерви, журналювання	C	-	-	-	A/R

Інформаційна безпека й відповідність вимогам політик ІБ також можуть бути фактором обережності. Вирішити це можна шляхом поетапного підключення GPT-сервісів через демілітаризовану зону (DMZ), впровадження TLS-шифрування, автентифікації SSO та ролей доступу, що вже закладено в архітектурі Risk-App. Комбінація цих заходів створює контрольоване середовище і дає змогу масштабувати рішення без втрати довіри з боку служби безпеки.

Для контролю зрілості процесу впровадження та підтвердження його результативності необхідно запровадити систему кількісних показників. Вони мають не лише відображати технічний стан Risk-App чи швидкість реакції на

події, а й демонструвати реальну управлінську користь - наскільки швидше виявляються ризики, як зменшується час ухвалення рішень і який економічний ефект дає мітигація. Задля цього формується набір ключових показників ефективності (KPI), які дають змогу перетворити процес управління ризиками на вимірювану дисципліну.

У таблиці 3.2 наведено приклад базових метрик, що рекомендовані для використання на етапі пілотного впровадження системи Risk-App та подальшого масштабування на рівень підприємства.

Таблиця 3.2

Ключові показники ефективності впровадження Risk-App

KPI	Формула або джерело	Цільове значення	Періодичність контролю	Відповідальний
Середній час реагування на ризик (TTR)	$t_{реакц.} - t_{виявл.}$	≤ 24 год	Щотижня	Координатор ризиків
Частка записів без власника	#без Owner / заг. #	0 %	Щомісяця	Власник процесу
Зниження інтегрального індексу ризику (ΔSRI)	$(SRI_{до} - SRI_{після})/SRI_{до}$	≥ 30 %	Щокварталу	Аналітик Risk-App
Виконання CAPA/WO в строк	#вчасно / заг. #	≥ 90 %	Щомісяця	Керівник виробництва
Доступність системи Risk-App	uptime за журналом	≥ 99,5 %	Постійно	IT-адміністратор

Узагальнюючи, впровадження моделі управління ризиками через Risk-App забезпечує не лише технічну інтеграцію даних, а й організаційне вирівнювання процесів. Поступове масштабування - від пілотної лінії до всього підприємства - формує культуру проактивного прийняття рішень, у якій ризик розглядається не як загроза, а як керований параметр ефективності.

3.3. Напрямки розвитку системи управління ризиками

Запропонована в роботі модель - це перша ітерація платформи, яка поєднує шаблонізований збір контексту, автоматизоване формування запитів до

GPT-API, нормовані вихідні структури (JSON для побудови матриці P×I, дерева відмов і дерева рішень), а також збереження допоміжних та візуалізаційних матеріалів у локальній БД з подальшим формуванням звітів і візуалізацій. На наступному етапі розвиток системи доцільно спрямувати в трьох взаємопов'язаних площинах: технологічній (архітектура, дані, інтеграції), аналітичній (методи і моделі) та організаційній (процеси, компетенції, комплаєнс). Нижче наведено бачення середньострокової еволюції, орієнтоване на реалії машинобудівного підприємства з виробництва жниварок.

Поточна реалізація доводить життєздатність шаблонного промптингу та структурованих відповідей. Подальший розвиток передбачає перехід від «точкових» обмінів до подієво-орієнтованої архітектури даних. У практичному вимірі це означає використання шини повідомлень або легкого ETL-шару для синхронізації подій і довідників між PLM (BOM/версії), ERP (PO/ETA/ціни/склади), MES/SCADA (OEE, телеметрія, скрап, алерти), QMS (невідповідності/CAPA), CMMS/EAM (WO/PM/MTBF/MTTR). Дані, що впливають на ризики, повинні бути доступні близько до реального часу й проходити нормалізацію в корпоративному сховищі (DWH/датамарт «Risk»), де формується єдина сутність «ризик» з атрибутами джерела, власника, категорії, ймовірності/впливу, поточних контрзаходів і статусу. На рівні безпеки інтеграцій - централізація автентифікації (SAML/OAuth з IAM/AD), ролей і аудит-трейлів, а також зонування доступу (корпоративна DMZ, сегментація OT-мережі відповідно до ISA/IEC 62443).

Перший реліз використовує структуровані підказки й суворий парсинг JSON. Подальші кроки включають використання «function calling» та «JSON-schema mode», що мінімізує ризик невалідних відповідей і зменшує пост-обробку. Доцільним є поєднання GPT з корпоративним знаннявим прошарком (RAG): при формуванні аналізу модель отримує релевантні фрагменти з шаблонів CAPA, бібліотеки FMEA, стандартних операційних карт (SOP/WI), специфікацій WPS та історичних інцидентів. У виробничих сценаріях це суттєво підвищує відтворюваність рекомендацій, а також спрощує

аудит рішень (джерела, на базі яких згенеровано поради, зберігаються разом із відповіддю моделі). Для чутливих даних варто передбачити локальний проксі та політики анонімізації: до зовнішньої моделі не вивозяться персональні дані, точні назви виробів або внутрішні коди дефектів - вони замінюються псевдонімами, а зіставлення виконується на внутрішньому шарі.

Система вже будує матрицю P×I, дерево відмов (FTA) та дерево рішень. Наступною сходинкою є поява кількісних методів для сценарного моделювання: Монте-Карло[37] для варіації ймовірностей/впливів; байєсівські мережі для умовних залежностей між подіями (наприклад, вплив якості витратних матеріалів і параметрів процесу на дефектність зварювання); аналіз чутливості (tornado-діаграми) для ранжування ключових драйверів ризику; розрахунок економічної цінності інформації (VOI/EVPI) - чи варто інвестувати в додаткову діагностику перед прийняттям рішення; оптимізація портфеля контрзаходів (integer programming) під обмеження бюджету та планових простоїв. Для виробничих ланцюгів із вузькими місцями (SPOF-обладнання) актуальними стають моделі надійності (RBD, Markov Chains) та поєднання з CMMS-даними (MTBF/MTTR, резервування, політика ЗІП).

Базова тріада P×I поступово трансформується в інтегральні індикатори, зрозумілі менеджменту виробництва: Risk-Adjusted OEE (коригування доступності/продуктивності/якості з урахуванням ризиків), Risk-Adjusted Cost per Unit (додає очікувані втрати через простій/переробку/скрап), Risk Burn-down (швидкість зменшення сукупного ризику після впровадження CAPA), Early-Warning Index (індикатори раннього попередження, що збираються з MES/SCADA/QMS). Такі метрики дозволяють інтегрувати ризик-менеджмент у щоденні виробничі наради (tier meetings) та квартальне планування інвестицій і техобслуговування.

Для критичних операцій - пресування, зварювання, фарбування - перспективним є створення «легких» цифрових двійників, що поєднують параметри рецепта/оснастки, телеметрію сенсорів, історію відхилень і результати контролю якості. На такому двійнику можна проганяти «що-якщо»

сценарії (зміна режимів, альтернативних постачальників витратних матеріалів, варіантів черговості замовлень) і отримувати оцінку ризиків ще до фактичного впровадження змін. Паралельно предиктивні моделі на базі CMMS/MES (градієнтні бустинги, ізоляційні ліси для аномалій, рекурентні мережі для сезонності) дають оцінку ймовірності відмов вузлів і дозволяють планувати ППР з мінімальним впливом на такт.

Система еволюціонує від інструмента аналітика до платформи, що ініціює й супроводжує виконання контрзаходів. У практиці це - інтеграція з QMS/CMMS: з реєстру ризиків формуються CAPA-карти або WO з автоматичним вибором маршруту узгодження (RACI-матриця, SLA, ескалації), контроль дедлайнів і підрахунок ефективності заходів «до/після». Нотифікації через корпоративні канали (Email/Teams/Telegram-бот) повинні мати глибокі посилання на відповідні сутності в ERP/QMS/CMMS. Важливим елементом є auditability: будь-яке автоматично згенероване рішення зберігає «причину та джерело» (вхідні дані, уривки документів, версія промпта/моделі), що критично для внутрішніх і зовнішніх аудитів.

З точки зору користувача подальший розвиток - це бібліотека «профілів» ризик-аналізу для типових кейсів машинобудування: дефектність зварювання, переналагодження штампів, нестабільність фарбувальних ліній, логістичні ризики «довгих» комплектуючих, сезонність попиту[11] на жниварки, кіберризики OT-сегмента. Кожен профіль - це набори полів, довідників, приклади заповнення, чек-листи даних, рекомендовані діаграми й алгоритми. Для досвідчених інженерів - режим «low-code»: можливість створювати власні шаблони промптів і правила парсингу без змін у коді програми.

В умовах інтеграції з OT-мережами пріоритетом стає безпека: сегментація, міжмережеві екрани, контрольних шлюзів для доступу в DMZ, «журнали довіри» для кожного конектора. На рівні даних - політики якості (data quality): контроль повноти та своєчасності потоків (freshness), сповіщення про збої ETL, правила узгодження довідників (BOM, коди дефектів). З комплаєнс-вимог релевантні ISO 31000 (загальні принципи

ризик-менеджменту), IATF 16949 (автопром, управління невідповідностями/PPAP/APQP), IEC 61508/61511 (функціональна безпека), ISA/IEC 62443 (OT-кібербезпека), NIST CSF (зрілість процесів кіберзахисту). Запровадження регулярних аудитів промптів і моделі (prompt governance, model cards) зменшує ризики помилкових рекомендацій і забезпечує відтворюваність.

Коли кількість користувачів і запитів зростає, монофайлова реалізація перетворюється на вузьке місце. Еволюція передбачає виділення окремих сервісів: шар доступу до даних (API для реєстру ризиків), сервіс генерації та валідації відповідей GPT (із чергами задач і ретраями), сервіс звітності/візуалізацій. На клієнті зберігаємо «легкий» застосунок, що працює офлайн на рівні локальної БД й синхронізується, коли є мережа. Для великих графіків і таблиць - серверний рендерінг (headless-рушій), щоб розвантажити робочі станції.

Ключова перевага системи - накопичення корпусу «запит → відповідь → виконані дії → ефект». Цей зворотний зв’язок відкриває шлях до тонкого налаштування моделей (fine-tuning) на доменному контенті, побудови мета-моделей для авто-калібрування P×I і наватренування детекторів високого ризику на ранніх ознаках (поєднання невеликих відхилень у кількох джерелах). Практично це реалізується через розширення схеми БД (лог результативності CAPA, cost-to-benefit, час впровадження, вплив на OEE), а також через «ML-ops-петлю» - збереження версій промптів, контрольних датасетів і періодичний A/B-аналіз.

Виробництво жниварок має виражену сезонність і залежність від зернового ринку та логістики. Система повинна підтримувати портфельне планування ризиків для варіантів продуктів (моделі/комплектації) і сценаріїв попиту: коливання постачання сталі, електроніки, гідравліки; коливання валют і фрахту; регіональні обмеження. У блоці дерева рішень корисно автоматизувати критерії вибору: коли вигідніше акумулювати запас і нести витрати на зберігання, а коли - прийняти ризик затримки або часткової комплектації. Для нових проєктів (модернізація рами, додавання опцій) система має працювати як

«консультант із ризиків» на етапі передпроєктних робіт: оцінювати вплив на технологічний маршрут, такт, потребу в оснастці/штампах, навантаження на вузькі місця та потребу в додаткових компетенціях.

Короткостроково (0-3 місяці) - стабілізація інтеграцій, уніфікація шаблонів, збагачення бібліотеки профілів, контроль якості даних, запуск рольової моделі доступу. Середньостроково (3-9 місяців) - RAG зі знаннявими базами (SOP/CAPA/FMEA), «function calling», автоматизований ланцюг CAPA/WO, перші кількісні моделі (Монте-Карло, VOI), основні KPI та дашборди. Довгостроково (9-18 місяців) - цифрові двійники критичних процесів, предиктивна надійність, оптимізація портфеля контрзаходів, часткова автономія прийняття рішень у межах встановленого «апетиту до ризику», масштабування на суміжні дивізіони або партнерські майданчики.

Запропоновані напрямки розвитку роблять систему не лише інструментом аналізу, а й платоформою постійного зменшення невизначеності у виробництві. У вимірюваних показниках це проявляється як скорочення непродуктивних втрат (скрап/переробки/простої), зменшення розкиду якості, пришвидшення циклу «виявив ризик → погодив дію → реалізував → зафіксував ефект», а також підвищення прозорості й відтворюваності управлінських рішень. Водночас архітектура лишається гнучкою: при зміні ринку, постачальників або технологій можна додати нові джерела даних, шаблони і метрики без переробки ядра.

Подальша еволюція системи управління ризиками повинна відбуватися не лише у технологічному вимірі, а й у стратегічному контексті всієї організації. Risk-App доцільно інтегрувати у корпоративну систему управління якістю (QMS) та процес постійного вдосконалення (Continuous Improvement), який реалізується через цикли PDCA. Це забезпечує реалізацію принципу «risk-based thinking», передбаченого стандартами ISO 9001 та IATF 16949, а також створює цифровий слід усіх рішень - від виявлення проблеми до перевірки результату дії. У рамках концепції Lean-виробництва система може виступати ядром реалізації підходу Jidoka, коли процес самостійно ідентифікує відхилення та

ініціює корекційні дії. Інтеграція з інструментами проєктного управління (MS Project, Jira, SAP PP) дозволить забезпечити трасування ризиків по всьому життєвому циклу виробу та синхронізацію між цехами, підрозділами і портфелем проєктів.

Сучасна практика управління підприємствами орієнтується також на ESG-підходи[38] (Environmental, Social & Governance). У цьому контексті Risk-App може бути розширена функціоналом для консолідації екологічних, соціальних і корпоративних ризиків. Доцільно передбачити додаткові категорії RBS, які охоплюють вплив на довкілля, охорону праці, енергоспоживання, поводження з відходами, а також кібербезпеку і відповідність етичним принципам постачання. Такий розвиток дає змогу підприємству не лише контролювати виробничі ризики, а й готувати інтегровані ESG-звіти відповідно до GRI та UN SDG Framework[45], що підвищує його репутаційну стійкість і привабливість для інвесторів.

Узагальнено напрямки розвитку системи подано в таблиці 3.3, де наведено орієнтовні горизонти, цілі та очікувані результати кожного етапу еволюції Risk-App.

Таблиця 3.3

Етапи розвитку системи Risk-App

Горизонт	Основні цілі	Ключові результати
0-3 міс. (пілот)	Уніфікація шаблонів, навчання користувачів, стабілізація обмінів	Заповнення реєстру, матриця P×I, базові дашборди
3-9 міс. (масштабування)	Підключення RAG-бази, function calling, автоматизація CAPA/WO	Повноцінний цикл PDCA, кількісні моделі VOI, KPI-панелі
9-18 міс. (зріла фаза)	Цифрові двійники, предиктивна надійність, портфельна оптимізація	Часткова автономія прийняття рішень, інтеграція ESG та QMS

У підсумку, запропонована стратегія розвитку Risk-App демонструє перехід від локального аналітичного інструмента до корпоративної платформи керування ризиками в реальному часі. Комбінація GPT-API, RAG, ETL-процесів, DWH-архітектури та цифрових двійників формує основу для самонавчальної екосистеми, у якій дані виробництва автоматично перетворюються на рішення, а досвід - на знання. Така трансформація зміцнює стійкість підприємства до зовнішніх і внутрішніх шоків, сприяє формуванню культури proactive risk management та підвищує конкурентоспроможність у середовищі Industry 4.0 / 5.0.

Висновки до розділу 3

Розділ 3 підтвердив практичну спроможність запропонованої моделі та інструмента Risk-App забезпечувати відтворюваний цикл управління ризиками - від ідентифікації та структурованої оцінки до пріоритизації, вибору дій і аудиту результатів. Верифікаційні сценарії для виробництва жниварок показали, що використання шаблонізованих промптів і суворої JSON-схеми стабільно приводить до коректного формування трьох ключових артефактів - матриці «ймовірність × вплив», дерева відмов і дерева рішень; дані автоматично зберігаються в реєстрі, а експорт у табличні формати гарантує інтеграційну готовність до ERP/BI. Закладені механізми стійкості - кеш, DEMO-режим, контроль формату відповіді - доводять безперервність роботи навіть за недоступності зовнішнього GPT-сервісу, що критично для виробничого середовища.

Рекомендації щодо впровадження окреслили керований шлях від пілота до масштабування. Організаційна модель із призначеним власником процесу, координатором ризиків, аналітиком Risk-App і чітко визначеними власниками записів у цехах забезпечує єдине «джерело правди» та дисципліну виконання. Технічно виправданим є старт на локальному SQLite з подальшою міграцією до серверного PostgreSQL без зміни предметної моделі; змішаний підхід до

інтеграцій дозволяє швидко підключити PLM/ERP/MES/QMS/CMMS і розгортати нотифікації для перетину порогів ризику чи прострочених дій. Уніфіковані шкали P та I, обов'язковість полів «власник» і «строк», а також політика зберігання історії створюють аудиторський слід і запобігають «нічийним» ризикам. Пілот на дільницях зі значним операційним впливом (зварювання, пресування) дозволяє швидко показати вимірюваний результат у часі реакції, якості та простоях, після чого модель масштабують на інші дільниці із закріпленням SLA інтеграцій.

Економічна доцільність моделі проявляється у скороченні сукупного ризикового впливу та прямій фінансовій віддачі від пріоритизованих контрзаходів. Введення інтегрального показника SRI для моніторингу «до/після» дозволяє прозоро демонструвати ефект у відсотках і грошовому еквіваленті, а дерева рішень - обґрунтовувати вибір альтернатив через очікувану цінність. Для виробничих вузлів із вузькими місцями економічна аргументація підсилюється часовим ефектом (ΔT) і ризик-скоригованими операційними, що напряму конвертуються у зменшення браку й простоїв.

Потенційні бар'єри впровадження - різнорідність даних, «острівна» культура ведення реєстрів, обережність служби ІБ і скепсис щодо автоматизованих рекомендацій - адресовані у проєкті через уніфікацію довідників і форматів, формалізацію ролей за RACI, поетапну безпечну інтеграцію через DMZ та TLS, а також через демонстрацію кейсів «до/після» і прозорість джерел рекомендацій. Запропонований набір KPI для пілоту та подальшого масштабу (час реагування, частка записів без власника, ΔSRI, своєчасність CAPA/WO, доступність системи) переводить впровадження з площини намірів у вимірювану дисципліну і створює підстави для регулярного управлінського огляду результатів.

Напрямки розвитку моделі визначають еволюцію від локального інструмента до корпоративної платформи керування ризиками в реальному часі. Перехід до подієво-орієнтованої архітектури з шиною повідомлень і датамартом «Risk», застосування RAG над доменними знаннями (SOP, FMEA, CAPA),

function calling/JSON-schema mode для ще суворішої валідації, а також кількісні методи (Монте-Карло, VOI/EVPI, оптимізація портфеля контрзаходів) підвищують аналітичну глибину. Інтеграція з процесами QMS/Lean і розширення на ESG-ризики підсилюють стратегічну роль системи, тоді як сервісна декомпозиція забезпечує масштабованість і продуктивність.

Отже, результати верифікації, рекомендації щодо впровадження і окреслена дорожня карта розвитку узгоджено демонструють, що Risk-App є життєздатним і економічно обґрунтованим рішенням для серійно-потокового виробництва жниварок. Модель переводить управління ризиками з простору дискусій у простір фактів, чисел і відповідальностей, скорочує час від сигналу до дії, підвищує прозорість рішень і створює стійку основу для подальшого кількісного вдосконалення процесів у логіці Industry 4.0/5.0.

ВИСНОВКИ

У роботі розроблено й експериментально апробовано модель управління ризиками для серійно-потокового виробництва жниварок, а також прикладний інструмент Risk-App, що реалізує повний цикл PDCA: ідентифікацію, оцінювання, пріоритизацію, планування дій, моніторинг і аудит. Теоретичну основу склали принципи ISO 31000 та практики PMI з використанням RBS як «словника» джерел невизначеності; інструментарій аналізу поєднав матрицю «ймовірність×вплив» як фільтр пріоритизації з причинно-наслідковими та сценарними моделями (bow-tie, FTA, ETA) і деревом рішень для порівняння альтернатив за очікуваною цінністю. Для виробничих проєктів обґрунтовано особливості ризиків на перетині інженерної безпеки, надійності й OT-кіберстійкості та показано роль IT-підрозділу у забезпеченні цілісності цифрового ланцюжка CAD/PLM→ERP→MES/SCADA→QMS/CMMS.

Спроєктована UML-модель (прецеденти, послідовності, діяльність) задала інженерний «контракт» між користувачем, застосунком і GPT-API: уніфікований промпт, сувора JSON-схема відповіді, локальне збереження елементів моделі у БД та автоматичний рендер матриці P×I, дерева відмов і дерева рішень. Інтеграційна архітектура визначила місце Risk-App у виробничому ландшафті: зліва - джерела даних (PLM, ERP, MES/SCADA, QMS, CMMS), справа - споживачі результатів (BI/DWH, управлінська звітність, канали нотифікацій), по горизонталі - IAM/AD і транспорт ETL/Message Bus, з урахуванням вимог безпеки (TLS, DMZ, журналювання). Мінімалістична схема реєстру ризиків забезпечила відтворюваність та імпорт/експорт у звичних офісних і корпоративних інструментах.

Верифікація показала, що Risk-App відтворювано формує артефакти аналізу та аудиту і зберігає їх у реєстрі; введення виробничого контексту через шаблони гарантує стабільність структури даних, а DEMO-режим і кешування - безперервність роботи за недоступності зовнішнього сервісу. На прикладах дільниць зварювання та пресування підтверджено адекватність розміщення

ризиків на матриці й логіку гілок дерев відмов/рішень. Запроваджений інтегральний індекс SRI та приклади розрахунків «до/після» продемонстрували вимірюваний ефект: скорочення SRI майже удвічі на тестових наборах і конвертацію часової дельти в грошовий еквівалент, що корелює з поліпшенням компонент OEE. Рекомендований набір KPI (час реагування, частка записів без власника, ΔSRI, своєчасність CAPA/WO, доступність системи) переводить впровадження з площини намірів у вимірювану управлінську практику.

Організаційна модель з RACI-розподілом відповідальності та «єдиним джерелом правди» усуває паралельні неформальні реєстри, зменшує тертя між підрозділами та прискорює рух від сигналу до дії. Технічно обґрунтовано шлях впровадження: пілот на локальному SQLite з подальшою міграцією до PostgreSQL і поетапним підключенням інтеграцій; змішаний режим обміну (подієвий і пакетний) дозволяє швидко здобути перші результати без складних перебудов інфраструктури. Ідентифіковані бар’єри - різнорідність даних, культурні звички «островів» Excel, обережність служби ІБ і скепсис до автоматизованих рекомендацій - подолано у роботі через уніфікацію довідників і форматів, формальне закріплення ролей, DMZ-підхід та прозорий аудит походження рекомендацій.

Наукова новизна полягає у поєднанні шаблонізованого доменного промптингу, суворої JSON-валідації та класичних інженерних методів аналізу ризиків у єдиному виробничому конвеєрі, що працює на реальних потоках даних. Практична цінність - у зниженні часу підготовки аналітики з днів до годин, у прозорій пріоритизації дій за критерієм очікуваної цінності та у можливості прямого підключення до управлінських циклів підприємства. Запропонована дорожня карта розвитку (RAG над SOP/FMEA/CAPA, function calling/JSON-schema mode, кількісні моделі Монте-Карло/VOI, цифрові двійники критичних процесів, портфельна оптимізація контрзаходів) окреслює перехід від локального інструмента до корпоративної платформи реального часу з ризик-скоригованими виробничими метриками та інтеграцією в QMS/ESG.

Обмеження роботи пов’язані з використанням тестових даних пілоту та припущень щодо ваг у SRI; їх слід уточнювати під конкретні KPI підприємства, а профілі ризиків - калібрувати на історії інцидентів і фактичних ефектах CAPA/WO. Подальші дослідження доцільно спрямувати на автоматизоване калібрування шкал P та I, побудову мета-моделей раннього попередження за мультиджерельними ознаками, розширення бібліотеки доменних «профілів» і впровадження ML-ops-петлі для версіонування промптів, контрольних датасетів і періодичного A/B-порівняння.

У підсумку сформована модель і реалізований Risk-App зміщують управління ризиками з площини дискусій у площину фактів, чисел і відповідальностей, забезпечують керовану, вимірювану та аудиторськи підтверджувану практику прийняття рішень і створюють стійку основу для еволюції до індустрії 4.0/5.0. Показана відтворюваність, інтеграційна готовність і економічна доцільність рішення є достатньою підставою для масштабування на інші дільниці, продуктові лінійки та споріднені підприємства машинобудівного профілю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC/IEEE 12207:2017 Systems and software engineering - Software life cycle processes. - Geneva: International Organization for Standardization, 2017. - URL: https://www.iso.org/standard/63712.html (дата звернення: 15.09.2025).
2. ISO 31000:2018. Risk management - Guidelines. - Женева: International Organization for Standardization, 2018. URL: https://www.iso.org/standard/65694.html (дата звернення: 14.09.2025).
3. Hillson D. Use a Risk Breakdown Structure (RBS) to Understand Your Risks. Proceedings of the PMI Annual Seminars & Symposium, 2002 URL: https://www.pmi.org/learning/library/risk-breakdown-structure-understand-risks-1042 (дата звернення: 16.09.2025).
4. Risk matrix. - Wikipedia, The Free Encyclopedia. - URL: https://en.wikipedia.org/wiki/Risk_matrix (дата звернення: 10.10.2025).
5. Health Service Executive (HSE) Ireland. Illustrative example of a Bow-tie analysis . - PDF. - URL: https://www.hse.ie/eng/about/who/riskmanagement/risk-management-documentation/hse-enterprise-risk-management-supporting-tools/illustrative-example-of-a-bowtie-analysis.pdf (дата звернення: 25.09.2025).
6. MindOnMap. Fault Tree Analysis: The Ultimate Guide. - URL: https://www.mindonmap.com/uk/blog/fault-tree-analysis/ (дата звернення: 28.09.2025).
7. Event tree analysis . - Wikipedia, The Free Encyclopedia. - URL: https://en.wikipedia.org/wiki/Event_tree_analysis (дата звернення: 21.09.2025).
8. Decision tree. - Wikipedia, The Free Encyclopedia. - URL: https://en.wikipedia.org/wiki/Decision_tree (дата звернення: 24.09.2025).
9. ISO 12100:2010. Safety of machinery - General principles for design - Risk assessment and risk reduction. - Женева: International Organization for

Standardization, 2010. - URL: https://www.iso.org/standard/51528.html (дата звернення: 30.09.2025).

10. Stouffer K., Pillitteri V., et al. NIST SP 800-82, Rev. 3: Guide to Operational Technology (OT) Security. - Gaithersburg, MD: NIST, 2023. - 316 с. - URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r3.pdf (дата звернення: 03.10.2025).
11. Association of Equipment Manufacturers (AEM) - «Canada Ag Tractor and Combine Report - July 2025 - URL: https://www.farmequip.org/wp-content/uploads/2025/08/July-2025-Canada.pdf (дата звернення: 11.10.2025).
12. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7th ed. - Newtown Square: PMI, 2021.
13. Project Management Institute. Practice Standard for Project Risk Management. - Newtown Square: PMI, 2019.
14. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 13th ed. - Hoboken: Wiley, 2022.
15. Hulett, D. T. Integrated Cost-Schedule Risk Analysis. - Gower Publishing, 2011.
16. Modarres, M. Risk Analysis in Engineering: Techniques, Tools, and Trends. - CRC Press, 2006.
17. Ericson, C. A. Hazard Analysis Techniques for System Safety. - Wiley, 2005.
18. Vesely, W. E. Fault Tree Handbook. - U.S. Nuclear Regulatory Commission, 1981.
19. Center for Chemical Process Safety. Guidelines for Hazard Evaluation Procedures. 3rd ed. - Wiley-AIChE, 2008.
20. CCPS. Bow-Tie Risk Analysis Methodology. - URL: https://www.aiche.org/ccps (дата звернення: 15.10.2025).
21. Kumamoto, H., Henley, E. Probabilistic Risk Assessment and Management for Engineers and Scientists. - IEEE Press, 1996.
22. Smith, C., Reinertsen D. Engineering Decision Making and Risk Management. - Wiley, 2015.

23. Shigley, J. Mechanical Engineering Design. 11th ed. - McGraw-Hill, 2020.
24. SAE International. J1939 Standards Collection. - URL: https://www.sae.org (дата звернення: 17.10.2025).
25. Rao, S. S. Reliability Engineering. - Prentice Hall, 2014.
26. Montgomery, D. Introduction to Statistical Quality Control. 8th ed. - Wiley, 2020.
27. FAO. Agricultural Machinery Safety Guidelines. - URL: https://www.fao.org (дата звернення: 20.10.2025).
28. International Society of Automation. ISA/IEC 62443 Cybersecurity Standards. - URL: https://www.isa.org (дата звернення: 24.10.2025).
29. Dragos. ICS/OT Cybersecurity Year in Review 2024. - URL: https://www.dragos.com
30. Siemens. Industrial Security - Defense in Depth Concept. - URL: https://www.siemens.com/industrialsecurity (дата звернення: 25.10.2025).
31. Honeywell. OT Cybersecurity Best Practices. - URL: https://www.honeywell.com (дата звернення: 27.10.2025).
32. OpenAI. GPT-4 Technical Report. 2023. - URL: https://openai.com/research/gpt-4 (дата звернення: 27.10.2025).
33. OpenAI. GPT-4o System Card. 2024. - URL: https://openai.com (дата звернення: 28.10.2025).
34. Lewis, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. - Meta AI Research, 2023.
35. Bubeck, S. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. - Microsoft Research, 2023.
36. Gartner. AI in Risk Management Outlook 2024.
37. IBM. Applying AI to Enterprise Risk Management. - Режим доступу: https://www.ibm.com (дата звернення: 25.10.2025).
38. McKinsey. The Future of AI in Manufacturing and Supply Chains. 2024. - URL: https://www.mckinsey.com/capabilities/operations/our-insights/beyond-automation-how-gen-ai-is-reshaping-supply-chains#/ (дата звернення: 28.10.2025).

39. Zhao, Q. Modern Fault Detection Using Machine Learning and AI. - Springer, 2021.
40. Zhang, Y. Decision Trees for Engineering Risk Assessment. - Elsevier, 2020.
41. Ericsson. AI & Automation in Smart Manufacturing - IndustryLab Report 2024.
42. ISO/IEC 27001:2022 Information Security Management Systems. - Geneva: ISO, 2022.
43. NIST. Guide for Conducting Risk Assessments (SP 800-30 Rev.1). - Gaithersburg: NIST, 2022.
44. Doran, T. (2006). IEEE 1220: for practical systems engineering. Computer, 39(5), 92-94.
45. IEC 61508:2010 Functional Safety of Electrical/Electronic/Programmable Systems. - International Electrotechnical Commission.

ДОДАТКИ

ДОДАТОК А

Лістинг коду

```
import os, json, sqlite3, hashlib, datetime, threading, traceback, textwrap,
random
import tkinter as tk
from tkinter import messagebox, filedialog
from tkinter import scrolledtext
import ttkbootstrap as ttk
from ttkbootstrap.constants import *

import pandas as pd
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.patches as patches

APP_TITLE = "Risk Manager (GPT + Prompt Builder)"
DB_FILE   = "risk_manager2.db"
THEME     = "darkly"
GPT_MODEL = "gpt-4o-mini"

OPENAI_API_KEY = "TEST"

def sha256(s: str) -> str:
    return hashlib.sha256(s.encode("utf-8")).hexdigest()

def now_iso() -> str:
    return datetime.datetime.utcnow().isoformat()

def clamp_1_5(x, default=3):
    try:
        v = int(x)
        return 1 if v < 1 else 5 if v > 5 else v
    except:
        return default

def safe_json_extract(text: str):
    try:
        s = text.strip()
        if s.startswith("```"):
            lines = [ln for ln in s.splitlines() if not
ln.strip().startswith("```")]
            s = "\n".join(lines)
        l = s.find("{"); r = s.rfind("}")
        if l != -1 and r != -1 and r > l:
            return json.loads(s[l:r+1])
        return json.loads(s)
    except Exception:
        raise ValueError("Неможливо розібрати JSON у відповіді моделі.")

SCHEMA_SQL = """
PRAGMA foreign_keys = ON;

CREATE TABLE IF NOT EXISTS users(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    full_name TEXT,
```

```
    created_at TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS queries(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    created_at TEXT NOT NULL,
    analysis_type TEXT NOT NULL,
    prompt TEXT NOT NULL,
    compiled_prompt TEXT NOT NULL,
    raw_json TEXT,
    FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS risk_items(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    query_id INTEGER NOT NULL,
    name TEXT NOT NULL,
    description TEXT,
    probability INTEGER NOT NULL,
    impact INTEGER NOT NULL,
    FOREIGN KEY(query_id) REFERENCES queries(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS fault_tree_nodes(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    query_id INTEGER NOT NULL,
    node_id TEXT NOT NULL,
    label TEXT NOT NULL,
    node_type TEXT NOT NULL,
    parent_id TEXT,
    FOREIGN KEY(query_id) REFERENCES queries(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS decision_tree_nodes(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    query_id INTEGER NOT NULL,
    node_id TEXT NOT NULL,
    label TEXT NOT NULL,
    node_type TEXT NOT NULL,
    parent_id TEXT,
    edge_prob REAL,
    payoff REAL,
    FOREIGN KEY(query_id) REFERENCES queries(id) ON DELETE CASCADE
);
"""

class Database:
    def __init__(self, path=DB_FILE):
        self.path = path
        self._first_time = not os.path.exists(path)
        self.lock = threading.Lock()
        self.con = sqlite3.connect(path, check_same_thread=False)
        self.con.execute("PRAGMA foreign_keys = ON;")
        self.con.executescript(SCHEMA_SQL)
        self.con.commit()
        if self._first_time:
            self.seed()

    def seed(self):
        with self.lock:
            cur = self.con.cursor()
            cur.execute("""INSERT INTO
```

```
users(username,email,password_hash,full_name,created_at)
                          VALUES(?,?,?,?,?)""",
                        ("admin", "admin@local", sha256("admin123"), "Системний
адміністратор", now_iso()))
            uid = cur.lastrowid
            demo_prompt = "DEMO: Жниварки. Згенерувати ризики, Fault Tree,
Decision Tree (строгий JSON)."
            cur.execute("""INSERT INTO
queries(user_id,created_at,analysis_type,prompt,compiled_prompt,raw_json)
                          VALUES(?,?,?,?,?,?)""",
                        (uid, now_iso(), "ALL", demo_prompt, demo_prompt, "{}"))
            qid = cur.lastrowid
            demo_risks = [
                ("Затримка постачання редукторів","Lead-time 10-12 тижнів, 1-2
постачальники",4,4),
                ("Відмова PLC на стенді","Несумісність прошивки/мережеві
збої",3,5),
                ("Низький FTT зварювання","Розкид параметрів та оснастки",4,3),
                ("Дефіцит електроніки","Нестача MCU/драйверів",4,4),
                ("Розсинхронізація BOM PLM↔ERP","Помилки інтеграції/процесів",3,4)
            ]
            for n,d,p,i in demo_risks:
                cur.execute("""INSERT INTO
risk_items(query_id,name,description,probability,impact)
                               VALUES(?,?,?,?,?)""", (qid,n,d,p,i))
            self.con.commit()

    # --- Users
    def create_user(self, username, email, password, full_name):
        with self.lock:
            try:
                self.con.execute("""INSERT INTO
users(username,email,password_hash,full_name,created_at)
                                    VALUES(?,?,?,?,?)""",

(username,email,sha256(password),full_name,now_iso()))
                self.con.commit()
                return True
            except sqlite3.IntegrityError:
                return False

    def auth_user(self, username, password):
        with self.lock:
            row = self.con.execute("""SELECT id,username,full_name FROM users
                                      WHERE username=? AND password_hash=?""",
                                   (username,sha256(password))).fetchone()
        if row:
            return {"id":row[0],"username":row[1],"full_name":row[2] or row[1]}
        return None

    # --- Queries + data
    def insert_query_payload(self, user_id, analysis_type, prompt,
compiled_prompt, payload):
        with self.lock:
            cur = self.con.cursor()
            cur.execute("""INSERT INTO
queries(user_id,created_at,analysis_type,prompt,compiled_prompt,raw_json)
                          VALUES(?,?,?,?,?,?)""",
                        (user_id, now_iso(), analysis_type, prompt,
compiled_prompt,
                         json.dumps(payload, ensure_ascii=False)))
            qid = cur.lastrowid
            for it in payload.get("risk_items", []):
```

```
                self.con.execute("""INSERT INTO
risk_items(query_id,name,description,probability,impact)
                                    VALUES(?,?,?,?,?)""",
                                  (qid, it.get("name","Без назви"),
it.get("description",""),
                                   clamp_1_5(it.get("probability",3)),
clamp_1_5(it.get("impact",3))))
            for n in payload.get("fault_tree", []):
                self.con.execute("""INSERT INTO
fault_tree_nodes(query_id,node_id,label,node_type,parent_id)
                                    VALUES(?,?,?,?,?)""",
                                  (qid, n.get("id"), n.get("label",""),
n.get("type","EVENT"), n.get("parent")))
            for n in payload.get("decision_tree", []):
                self.con.execute("""INSERT INTO
decision_tree_nodes(query_id,node_id,label,node_type,parent_id,edge_prob,payoff)
                                    VALUES(?,?,?,?,?,?,?)""",
                                  (qid, n.get("id"), n.get("label",""),
n.get("type","DECISION"),
                                   n.get("parent"), n.get("prob"),
n.get("payoff")))
            self.con.commit()
            return qid

    def user_queries(self, user_id):
        with self.lock:
            return self.con.execute("""SELECT id,created_at,analysis_type,prompt
                                       FROM queries WHERE user_id=? ORDER BY id
DESC""",
                                    (user_id,)).fetchall()

    def risks_for_query(self, qid):
        with self.lock:
            return self.con.execute("""SELECT name,description,probability,impact
                                       FROM risk_items WHERE
query_id=?""",(qid,)).fetchall()

    def fault_nodes_for_query(self, qid):
        with self.lock:
            return self.con.execute("""SELECT node_id,label,node_type,parent_id
                                       FROM fault_tree_nodes WHERE
query_id=?""",(qid,)).fetchall()

    def decision_nodes_for_query(self, qid):
        with self.lock:
            return self.con.execute("""SELECT
node_id,label,node_type,parent_id,edge_prob,payoff
                                       FROM decision_tree_nodes WHERE
query_id=?""",(qid,)).fetchall()

class GPTClient:
    def __init__(self, api_key: str):
        self.api_key = api_key.strip()

    def call(self, messages, temperature=0.2):
        if not self.api_key:
            return '{"risk_items":[{"name":"DEMO ризик","description":"демо
опис","probability":3,"impact":3}],"fault_tree":[{"id":"TOP","label":"DEMO
TOP","type":"EVENT","parent":null}],"decision_tree":[{"id":"D0","label":"DEMO
DECISION","type":"DECISION","parent":null,"prob":null,"payoff":null},{"id":"T1","l
abel":"Результат","type":"TERMINAL","parent":"D0","prob":null,"payoff":0.0}]}'
        try:
            import openai
```

```
            if hasattr(openai, "OpenAI"):
                client = openai.OpenAI(api_key=self.api_key)
                res = client.chat.completions.create(model=GPT_MODEL,
temperature=temperature, messages=messages)
                return res.choices[0].message.content
            else:
                openai.api_key = self.api_key
                res = openai.ChatCompletion.create(model=GPT_MODEL,
temperature=temperature, messages=messages)
                return res["choices"][0]["message"]["content"]
        except Exception as e:
            msg = str(e)
            if "insufficient_quota" in msg or "You exceeded your current quota" in
msg:
                messagebox.showwarning("GPT", "429 insufficient_quota: поповніть
баланс або збільшіть ліміти. Використано DEMO.")
            else:
                messagebox.showwarning("GPT", f"Помилка виклику API:
{e}\nВикористано DEMO-дані.")
            return '{"risk_items":[{"name":"DEMO ризик
(quota)","description":"демо","probability":3,"impact":4}],"fault_tree":[{"id":"TO
P","label":"DEMO
TOP","type":"EVENT","parent":null}],"decision_tree":[{"id":"D0","label":"DEMO
DECISION","type":"DECISION","parent":null,"prob":null,"payoff":null},{"id":"T1","l
abel":"Результат","type":"TERMINAL","parent":"D0","prob":null,"payoff":0.0}]}'

def draw_risk_matrix(fig: Figure, risks_df: pd.DataFrame):
    import math
    import matplotlib.patheffects as pe
    from matplotlib import patches

    fig.clear()
    ax = fig.add_subplot(111)
    fig.subplots_adjust(left=0.06, right=0.70, top=0.92, bottom=0.08)

    # Фон матриці 5×5
    for x in range(5):
        for y in range(5):
            score = (x + 1) * (y + 1)
            color = "#e8f7ee" if score <= 6 else "#fff7d6" if score <= 12 else
"#ffe4e6"
            ax.add_patch(
                patches.Rectangle(
                    (x, y), 1, 1, facecolor=color, edgecolor="#94a3b8",
linewidth=1
                )
            )

    buckets = {}
    df = risks_df.reset_index(drop=True)
    for i, row in df.iterrows():
        px = clamp_1_5(row["probability"])
        py = clamp_1_5(row["impact"])
        buckets.setdefault((px, py), []).append((i, row["name"]))

    def offsets(n: int):
        if n <= 1:
            return [(0.0, 0.0)]
        pts = []
        first = min(n, 6)              # до 6 точок на зовнішньому кільці
        for k in range(first):
            ang = 2 * math.pi * k / first
            pts.append((0.28 * math.cos(ang), 0.28 * math.sin(ang)))
```

```
            if n > 6:                       # решта - на внутрішньому кільці
                rest = n - 6
                for k in range(rest):
                    ang = 2 * math.pi * k / max(rest, 1)
                    pts.append((0.14 * math.cos(ang), 0.14 * math.sin(ang)))
            return pts[:n]

        # Малюємо точки + номера без перекриттів
        idx_to_num = {}
        for (px, py), items in buckets.items():
            for (dx, dy), (i, name) in zip(offsets(len(items)), items):
                x = px - 0.5 + dx
                y = py - 0.5 + dy
                ax.scatter([x], [y], s=260, c="#0ea5e9", edgecolors="white",
                           linewidths=2.0, zorder=3)
                num = i + 1
                idx_to_num[i] = num
                t = ax.text(x, y, str(num), color="white", fontsize=11,
                            ha="center", va="center", zorder=4)
                # біле "хало" навколо цифри для контрасту
                t.set_path_effects([pe.withStroke(linewidth=2.6, foreground="black")])


        legend_lines = []
        for i, row in df.iterrows():
            nm = textwrap.shorten(str(row["name"]), 60)
            p = clamp_1_5(row["probability"])
            im = clamp_1_5(row["impact"])
            legend_lines.append(f"{idx_to_num.get(i, i+1)}. {nm} (P{p}×I{im})")

        ax.set_xlim(0, 5); ax.set_ylim(0, 5)
        ax.set_xticks([i + 0.5 for i in range(5)], [f"P{i}" for i in range(1, 6)])
        ax.set_yticks([i + 0.5 for i in range(5)], [f"I{i}" for i in range(1, 6)])
        ax.set_title("Матриця «Ймовірність × Вплив»")
        ax.set_aspect("equal")
        ax.invert_yaxis()
        ax.grid(False)

        legend_text = "\n".join(legend_lines) if legend_lines else "-"
        fig.text(0.72, 0.5, legend_text, va="center", ha="left", fontsize=9)

    def _tree_levels(edges_map, root_id):
        levels = {}; frontier = [root_id]; lvl=0
        while frontier:
            levels[lvl] = frontier[:]
            nxt = []
            for u in frontier: nxt.extend(edges_map.get(u, []))
            frontier = nxt; lvl += 1
        return levels

    def _wrap_label(label: str, width_chars: int = 26, max_lines: int = 4):
        lines = textwrap.wrap(label, width=width_chars) or [label]
        if len(lines) > max_lines:
            lines = lines[:max_lines]
            # додамо "…" в окремий рядок лише якщо обрізали (але не всередині рядка)
            lines[-1] = lines[-1] + "…"
        return "\n".join(lines), len(lines), max((len(l) for l in lines), default=1)

    def render_fault_tree(fig: Figure, nodes: list):
        fig.clear(); ax = fig.add_subplot(111)
        fig.subplots_adjust(left=0.03, right=0.97, top=0.93, bottom=0.07)

        if not nodes:
```

```
        ax.text(0.5,0.5,"Немає даних Fault Tree", ha="center", va="center")
        ax.axis("off"); fig.tight_layout(); return

    by_id = {n["node_id"]:n for n in nodes}
    childs = {}; root=None
    for n in nodes:
        p = n["parent_id"]
        if p is None: root = n["node_id"]
        childs.setdefault(p, []).append(n["node_id"])
    if not root: root = nodes[0]["node_id"]
    levels = _tree_levels(childs, root); pos={}
    for lvl, ids in levels.items():
        for i, nid in enumerate(ids):
            pos[nid]=(lvl, 1-(i+1)/(len(ids)+1))

    # edges
    for n in nodes:
        p = n["parent_id"]
        if p is None or p not in pos: continue
        x1,y1 = pos[p]; x2,y2 = pos[n["node_id"]]
        ax.plot([x1+0.45,x2-0.45],[y1,y2], color="#94a3b8")

    # nodes (перенос підписів)
    for n in nodes:
        x,y = pos[n["node_id"]]
        wrapped, n_lines, maxlen = _wrap_label(n["label"], width_chars=24)
        node_w = 0.72 + 0.02*max(0, maxlen-20)  # ширше для довгих рядків
        node_h = 0.12 + 0.03*(n_lines-1)       # вища коробка під кілька рядків

        if n["node_type"]=="EVENT":
            rect = patches.FancyBboxPatch((x-node_w/2,y-node_h/2),node_w,node_h,

boxstyle="round,pad=0.02,rounding_size=0.06",
                                          facecolor="#0ea5e9", edgecolor="white")
        else:
            color = "#22c55e" if n["node_type"]=="GATE_AND" else "#f59e0b"
            rect = patches.FancyBboxPatch((x-node_w/2,y-node_h/2),node_w,node_h,

boxstyle="round,pad=0.02,rounding_size=0.05",
                                          facecolor=color, edgecolor="white")
        ax.add_patch(rect)
        ax.text(x,y, wrapped, ha="center", va="center", color="white", fontsize=9)

    ax.set_title("Fault Tree (дерево відмов)"); ax.axis("off")

def render_decision_tree(fig: Figure, nodes: list):
    fig.clear(); ax = fig.add_subplot(111)
    fig.subplots_adjust(left=0.03, right=0.97, top=0.93, bottom=0.07)

    if not nodes:
        ax.text(0.5,0.5,"Немає даних Decision Tree", ha="center", va="center")
        ax.axis("off"); return

    childs = {}; root=None
    for n in nodes:
        p = n["parent_id"]
        if p is None: root = n["node_id"]
        childs.setdefault(p, []).append(n["node_id"])
    if not root: root = nodes[0]["node_id"]

    levels = _tree_levels(childs, root); pos={}
    for lvl, ids in levels.items():
        for i, nid in enumerate(ids):
```

```
            pos[nid]=(lvl, 1-(i+1)/(len(ids)+1))

    # edges
    for n in nodes:
        p = n["parent_id"]
        if p is None or p not in pos: continue
        x1,y1 = pos[p]; x2,y2 = pos[n["node_id"]]
        ax.plot([x1+0.45,x2-0.45],[y1,y2], color="#94a3b8")
        if n["node_type"]!="DECISION" and n.get("edge_prob") is not None:
            ax.text((x1+x2)/2,(y1+y2)/2, f"p={float(n['edge_prob']):.2f}",
                    color="#94a3b8", fontsize=8)

    # nodes - перенос, адаптивні розміри
    for n in nodes:
        x,y = pos[n["node_id"]]
        wrapped, n_lines, maxlen = _wrap_label(n["label"], width_chars=28)
        node_w = 0.80 + 0.03*max(0, maxlen-24)
        node_w = min(node_w, 1.40)
        node_h = 0.12 + 0.035*(n_lines-1)

        t = n["node_type"]
        face = "#0284c7" if t=="DECISION" else "#22c55e" if t=="CHANCE" else
"#f59e0b"
        rect = patches.FancyBboxPatch((x-node_w/2,y-node_h/2),node_w,node_h,

boxstyle="round,pad=0.02,rounding_size=0.06",
                                      facecolor=face, edgecolor="white")
        ax.add_patch(rect)
        lbl = wrapped
        if t=="TERMINAL" and n.get("payoff") is not None:
            lbl += f"\nEV={float(n['payoff']):.1f}"
        ax.text(x,y, lbl, ha="center", va="center", color="white", fontsize=9)

    ax.set_title("Decision Tree (дерево рішень)"); ax.axis("off")

class App:
    def __init__(self, root: ttk.Window):
        self.root = root
        self.root.title(APP_TITLE)
        self.root.geometry("1280x860")
        self.root.minsize(1140, 780)

        self.db = Database(DB_FILE)
        self.api_key = OPENAI_API_KEY

        self.user = None
        self.current_qid = None

        self._loading = False
        self._spin_idx = 0

        self.build_login()

    def _make_entry(self, parent, var, width=64):
        ent = ttk.Entry(parent, textvariable=var, width=width)
        def show_end(*_):
            try:
                ent.icursor(tk.END)
                ent.xview_moveto(1)
            except:
                pass
        var.trace_add("write", lambda *_: show_end())
        ent.bind("<Map>", show_end)
```

```
        ent.bind("<FocusIn>", show_end)
        self.root.after(0, show_end)
        return ent

    def _start_spinner(self, message="Виконується..."):
        self._loading = True
        self._spin_idx = 0
        chars = ["⠋","⠙","⠹","⠸","⠼","⠴","⠦","⠧","⠇","⠏"]
        def tick():
            if not self._loading: return
            self._spin_idx = (self._spin_idx + 1) % len(chars)
            self.status_var.set(f"{message} {chars[self._spin_idx]}")
            self.root.after(120, tick)
        tick()

    def _stop_spinner(self, message="Готово"):
        self._loading = False
        self.status_var.set(message)

    def build_login(self):
        for w in self.root.winfo_children(): w.destroy()
        frm = ttk.Frame(self.root, padding=28); frm.pack(expand=True)

        ttk.Label(frm, text="Система управління ризиками", font=("Segoe UI", 20,
"bold")).grid(row=0,column=0,columnspan=2,pady=(0,12))

        ttk.Label(frm,
text="Логін:").grid(row=1,column=0,sticky="e",padx=6,pady=6)
        self.ent_user = ttk.Entry(frm, width=34);
self.ent_user.grid(row=1,column=1,pady=6)
        ttk.Label(frm,
text="Пароль:").grid(row=2,column=0,sticky="e",padx=6,pady=6)
        self.ent_pwd  = ttk.Entry(frm, show="*", width=34);
self.ent_pwd.grid(row=2,column=1,pady=6)

        row = ttk.Frame(frm); row.grid(row=3,column=0,columnspan=2,pady=10)
        ttk.Button(row, text="Увійти", bootstyle=SUCCESS,
command=self.do_login).pack(side=LEFT,padx=6)
        ttk.Button(row, text="Реєстрація", bootstyle=INFO,
command=self.build_register).pack(side=LEFT,padx=6)

        status = "GPT-API активний" if self.api_key else "DEMO MODE (без ключа)"
        ttk.Label(frm, text=status, bootstyle=(SUCCESS if self.api_key else
WARNING)).grid(row=4,column=0,columnspan=2)

        self.ent_user.focus_set()
        self.root.bind("<Return>", lambda e: self.do_login())

    def build_register(self):
        for w in self.root.winfo_children(): w.destroy()
        frm = ttk.Frame(self.root, padding=28); frm.pack(expand=True)
        ttk.Label(frm, text="Реєстрація", font=("Segoe UI", 18,
"bold")).grid(row=0,column=0,columnspan=2,pady=(0,12))
        self.reg = {}
        for i,(lbl,key) in enumerate([("Повне
ім'я","full"),("Email","mail"),("Логін","user"),("Пароль","pass"),("Підтвердження"
,"conf")], start=1):
            ttk.Label(frm,
text=lbl+":").grid(row=i,column=0,sticky="e",padx=6,pady=6)
            show = "*" if "Пароль" in lbl else None
            self.reg[key] = ttk.Entry(frm, width=38, show=show)
            self.reg[key].grid(row=i,column=1,pady=6)
        row = ttk.Frame(frm); row.grid(row=6,column=0,columnspan=2,pady=8)
```

```
        ttk.Button(row, text="Створити", bootstyle=SUCCESS,
command=self.do_register).pack(side=LEFT,padx=6)
        ttk.Button(row, text="Назад", bootstyle=SECONDARY,
command=self.build_login).pack(side=LEFT,padx=6)

    def do_register(self):
        full = self.reg["full"].get().strip()
        mail = self.reg["mail"].get().strip()
        user = self.reg["user"].get().strip()
        pasw = self.reg["pass"].get().strip()
        conf = self.reg["conf"].get().strip()
        if not (mail and user and pasw and conf):
            messagebox.showerror("Помилка","Заповніть усі поля."); return
        if pasw != conf:
            messagebox.showerror("Помилка","Паролі не співпадають."); return
        if self.db.create_user(user, mail, pasw, full):
            messagebox.showinfo("OK","Користувача створено."); self.build_login()
        else:
            messagebox.showerror("Помилка","Логін або email вже існує.")

    def do_login(self):
        info = self.db.auth_user(self.ent_user.get().strip(),
self.ent_pwd.get().strip())
        if info:
            self.user = info
            self.build_main()
        else:
            messagebox.showerror("Помилка","Невірний логін або пароль.")


    def build_main(self):
        for w in self.root.winfo_children(): w.destroy()
        top = ttk.Frame(self.root, padding=8); top.pack(fill=X)
        ttk.Label(top, text=f"👋 Вітаємо, {self.user['full_name']}", font=("Segoe
UI", 14, "bold")).pack(side=LEFT)
        ttk.Button(top, text="Вийти", bootstyle=DANGER,
command=self.build_login).pack(side=RIGHT)

        self.nb = ttk.Notebook(self.root, padding=8); self.nb.pack(fill=BOTH,
expand=True)
        self.tab_builder = ttk.Frame(self.nb); self.nb.add(self.tab_builder,
text="🧩 Конструктор промпту")
        self.tab_tables  = ttk.Frame(self.nb); self.nb.add(self.tab_tables,
text="📋 Таблиці")
        self.tab_charts  = ttk.Frame(self.nb); self.nb.add(self.tab_charts,
text="📈 Графіки")
        self.tab_hist    = ttk.Frame(self.nb); self.nb.add(self.tab_hist,
text="🕘 Історія")

        self.build_builder()
        self.build_tables()
        self.build_charts()
        self.build_history()

    # ---------- TAB: PROMPT BUILDER ----------
    def build_builder(self):
        frm = ttk.Frame(self.tab_builder); frm.pack(fill=BOTH, expand=True)

        # пресети
        self.PRESETS = {
            "Жниварки - серійне (120/міс)": dict(
                product="Жниварки для зернокомбайнів",
prod_type="Серійне/потокове",
```

```
                capacity="120 од./міс",
                sections="різання; мехобробка; зварювання рами; фарбування;
субзбірки; фінальна збірка; стендові випробування; ОТК",
                chain="CAD→PLM→ERP→MES/SCADA",
                it="PLM/ERP/MES; інтеграції; мережа/OT-безпека; сервіс PLC/HMI",
                critical="рама; редуктор приводу; ріжучий апарат; гідравліка;
електроніка",
                suppliers="редуктори: EU-Reduct GmbH (LT 10-12 тижнів);
електроніка (MCU/драйвери); листовий метал; фарби",
                bottlenecks="зварювання рам; своєчасність постачань; синхронізація
інженерних змін; стендові відмови PLC/HMI; фарбування репроц",
                rc="14", objective="мінімізація ризику постачання редукторів і
відмов на стендах"
            ),
            "Жниварки - дрібносерійні (40/міс)": dict(
                product="Жниварки для нішевих культур", prod_type="Дрібносерійне",
                capacity="40 од./міс",
                sections="різання; мехобробка; зварювання; фарбування; збірка;
тестування",
                chain="CAD→PDM→ERP",
                it="PDM/ERP; інтеграції; мережа; сервіс",
                critical="рама; різальний брус; привід; електроніка",
                suppliers="редуктори: локальні (LT 6-8 тижнів); електроніка -
імпорт",
                bottlenecks="налаштування оснастки; нестабільність ланцюга
постачань",
                rc="12", objective="скорочення циклу від замовлення до
відвантаження"
            ),
            "Стендові випробування - фокус на PLC/HMI": dict(
                product="Стендові комплекси для жниварок", prod_type="Одиничне",
                capacity="N/A",
                sections="електромонтаж; ПЗ; стендові тести",
                chain="PLM→ERP→MES",
                it="MES/SCADA; мережа/OT-безпека; сервіс PLC/HMI",
                critical="PLC; датчики; гідравліка; прошивки",
                suppliers="PLC/модулі I/O; кабелі; датчики тиску/потоку",
                bottlenecks="несумісність прошивок; мережеві колізії; доступ до
репозиторіїв",
                rc="16", objective="підвищення успішності стендових випробувань"
            )
        }

        # Ліва колонка - параметри
        left = ttk.Labelframe(frm, text="Параметри виробництва (динамічні
змінні)", padding=10)
        left.pack(side=LEFT, fill=Y, padx=(0,8), pady=6)

        # Порожні значення за замовчуванням
        self.var_product      = tk.StringVar(value="")
        self.var_prod_type    = tk.StringVar(value="")
        self.var_capacity     = tk.StringVar(value="")
        self.var_sections     = tk.StringVar(value="")
        self.var_chain        = tk.StringVar(value="")
        self.var_it_areas     = tk.StringVar(value="")
        self.var_critical     = tk.StringVar(value="")
        self.var_suppliers    = tk.StringVar(value="")
        self.var_bottlenecks  = tk.StringVar(value="")
        self.var_risk_count   = tk.StringVar(value="14")
        self.var_objective    = tk.StringVar(value="")

        # Лінія пресетів
        preset_line = ttk.Frame(left); preset_line.pack(fill=X, pady=(0,6))
```

```
        ttk.Label(preset_line, text="Шаблон:").pack(side=LEFT)
        self.cmb_preset = ttk.Combobox(preset_line, state="readonly", width=56,
                                       values=list(self.PRESETS.keys()))
        self.cmb_preset.pack(side=LEFT, padx=6)
        ttk.Button(preset_line, text="Застосувати", bootstyle=PRIMARY,
command=self.apply_preset).pack(side=LEFT)
        ttk.Button(preset_line, text="Очистити", bootstyle=SECONDARY,
command=self.clear_fields).pack(side=LEFT, padx=6)

        grid = ttk.Frame(left); grid.pack(fill=X)

        def row(i, label, kind="entry", var=None, values=None):
            ttk.Label(grid, text=label, anchor="e").grid(row=i, column=0,
sticky="e", padx=4, pady=4)
            if kind=="combo":
                cb = ttk.Combobox(grid, textvariable=var, values=values,
state="readonly", width=62)
                cb.grid(row=i, column=1, sticky="we", padx=4, pady=4)
            else:
                ent = self._make_entry(grid, var, width=64)
                ent.grid(row=i,column=1,sticky="we",padx=4,pady=4)

        row(0, "Продукт:", "entry", self.var_product)
        row(1, "Тип виробництва:", "combo", self.var_prod_type,
["Серійне/потокове","Дрібносерійне","Одиничне"])
        row(2, "Потужність:", "entry", self.var_capacity)
        row(3, "Дільниці/цехи:", "entry", self.var_sections)
        row(4, "Цифровий ланцюжок:", "combo", self.var_chain,
["CAD→PLM→ERP→MES/SCADA","CAD→PDM→ERP","PLM→ERP→MES"])
        row(5, "Зони IT:", "entry", self.var_it_areas)
        row(6, "Критичні вузли:", "entry", self.var_critical)
        row(7, "Постачання/LT:", "entry", self.var_suppliers)
        row(8, "Вузькі місця:", "entry", self.var_bottlenecks)
        row(9, "К-сть ризиків:", "combo", self.var_risk_count, [str(i) for i in
range(12,19)])
        row(10,"Ціль рішення:", "entry", self.var_objective)

        btns = ttk.Frame(left); btns.pack(fill=X, pady=(8,0))
        ttk.Button(btns, text="📄 Попередній перегляд промпту",
bootstyle=SECONDARY, command=self.preview_prompt).pack(side=LEFT)
        ttk.Button(btns, text="🚀 Запустити GPT", bootstyle=SUCCESS,
command=self.start_gpt).pack(side=LEFT, padx=8)

        # Права колонка - текст/лог + статус (спінер)
        right = ttk.Labelframe(frm, text="Скомпільований промпт / Логи",
padding=10)
        right.pack(side=RIGHT, fill=BOTH, expand=True, padx=(8,0), pady=6)
        self.txt_preview = scrolledtext.ScrolledText(right, height=30,
font=("Consolas", 10), wrap=tk.WORD)
        self.txt_preview.pack(fill=BOTH, expand=True)
        self.status_var = tk.StringVar(value="")
        self.lbl_status = ttk.Label(right, textvariable=self.status_var,
bootstyle=INFO)
        self.lbl_status.pack(anchor="w", pady=(6,0))

    def apply_preset(self):
        key = self.cmb_preset.get().strip()
        if not key: return
        p = self.PRESETS[key]
        self.var_product.set(p["product"])
        self.var_prod_type.set(p["prod_type"])
        self.var_capacity.set(p["capacity"])
        self.var_sections.set(p["sections"])
```

```
        self.var_chain.set(p["chain"])
        self.var_it_areas.set(p["it"])
        self.var_critical.set(p["critical"])
        self.var_suppliers.set(p["suppliers"])
        self.var_bottlenecks.set(p["bottlenecks"])
        self.var_risk_count.set(p["rc"])
        self.var_objective.set(p["objective"])

    def clear_fields(self):
        for v in
[self.var_product,self.var_prod_type,self.var_capacity,self.var_sections,

self.var_chain,self.var_it_areas,self.var_critical,self.var_suppliers,
                  self.var_bottlenecks,self.var_objective]:
            v.set("")
        self.var_risk_count.set("14")

    def compile_prompt(self):
        ctx = f"""
ПІДПРИЄМСТВО
• Продукт: {self.var_product.get().strip()}.
• Тип виробництва: {self.var_prod_type.get().strip()}.
• Потужність: {self.var_capacity.get().strip()}.
• Дільниці/цехи: {self.var_sections.get().strip()}.
• Цифровий ланцюжок: {self.var_chain.get().strip()} (трасування BOM/маршрутів,
WIP, FTT, PPM).
• Зони IT-відповідальності: {self.var_it_areas.get().strip()}.
• Критичні вузли виробу: {self.var_critical.get().strip()}.
• Постачання/Lead Time: {self.var_suppliers.get().strip()}.
• Вузькі місця: {self.var_bottlenecks.get().strip()}.
"""
        want_risks = self.var_risk_count.get().strip() or "14"
        schema = {

"risk_items":[{"name":"string","description":"string","probability":"1..5","impact
":"1..5"}],

"fault_tree":[{"id":"string","label":"string","type":"EVENT|GATE_AND|GATE_OR","par
ent":"string|null"}],

"decision_tree":[{"id":"string","label":"string","type":"DECISION|CHANCE|TERMINAL"
,"parent":"string|null","prob":"0..1|null","payoff":"float|null"}]
        }

        system = (
            "Ти - експерт з управління ризиками у серійному виробництві
агротехніки. "
            "Проаналізуй наданий контекст і ПОВЕРНИ СТРОГО JSON (без markdown і
пояснень) "
            f"зі структурою: {json.dumps(schema, ensure_ascii=False)}. "
            "Валідація: probability/impact ∈ [1..5]; fault_tree.type ∈
{EVENT,GATE_AND,GATE_OR}; "
            "decision_tree: один DECISION-корінь; CHANCE має 'prob'; TERMINAL має
'payoff'. "
            "Якщо бракує даних - зроби реалістичні припущення."
        )

        user = f"""
КОНТЕКСТ:
{ctx}

ЗАВДАННЯ:
1) Згенеруй {want_risks} унікальних ризиків (risk_items) українською.
```

```
2) Fault Tree (мінімум 8 вузлів), корінь:
   {{"id":"TOP","label":"Невдача приймальних випробувань
жниварки","type":"EVENT","parent":null}},
   включи хоча б один GATE_AND і один GATE_OR.
3) Decision Tree: кореневий DECISION із ціллю
   "{self.var_objective.get().strip() or 'мінімізація ризику постачання та відмов
на стендах'}".
   Під ним 2-3 CHANCE-гілки (ймовірності у сумі = 1.0±0.01),
   кожна має 1-2 TERMINAL листки з 'payoff' (float, очікуваний ефект у тис. USD).

ПОВЕРНИ ЛИШЕ ОДИН ВАЛІДНИЙ JSON-ОБ'ЄКТ БЕЗ ДОДАТКОВОГО ТЕКСТУ.
"""
        return system, user, ctx

    def preview_prompt(self):
        system, user, ctx = self.compile_prompt()
        self.txt_preview.delete("1.0", tk.END)
        self.txt_preview.insert(tk.END, "- SYSTEM -\n")
        self.txt_preview.insert(tk.END, system + "\n\n")
        self.txt_preview.insert(tk.END, "- USER -\n")
        self.txt_preview.insert(tk.END, user.strip())

    def start_gpt(self):
        system, user, ctx = self.compile_prompt()
        self.txt_preview.delete("1.0", tk.END)
        self.txt_preview.insert(tk.END, "⏳ Звернення до GPT...\n")
        self._start_spinner("Генерація відповіді")

        def worker():
            try:
                client = GPTClient(self.api_key)
                text = client.call(
                    messages=[{"role":"system","content":system},
                              {"role":"user","content":user}]
                )
                try:
                    data = safe_json_extract(text)
                    for it in data.get("risk_items", []):
                        it["probability"] = clamp_1_5(it.get("probability",3))
                        it["impact"]      = clamp_1_5(it.get("impact",3))
                except Exception:
                    data = {"risk_items":[{"name":"DEMO ризик","description":"не
вдалося розібрати JSON","probability":3,"impact":3}],
                            "fault_tree":[{"id":"TOP","label":"DEMO
TOP","type":"EVENT","parent":None}],
                            "decision_tree":[{"id":"D0","label":"DEMO
DECISION","type":"DECISION","parent":None,"prob":None,"payoff":None},

{"id":"T1","label":"Результат","type":"TERMINAL","parent":"D0","prob":None,"payoff
":0.0}]}
                self.root.after(0, lambda: self._commit_result(user, data))
            except Exception as e:
                tb = traceback.format_exc()
                self.root.after(0, lambda: self._gpt_fail(e, tb))

        threading.Thread(target=worker, daemon=True).start()

    def _commit_result(self, compiled_prompt, data):
        try:
            qid = self.db.insert_query_payload(
                self.user["id"], "ALL",
                prompt="Згенеровано конструктором промпту",
                compiled_prompt=compiled_prompt,
```

```
                payload=data
            )
            self.current_qid = qid
            self._stop_spinner(f"✅ Отримано дані та збережено (ID {qid})")
            self.txt_preview.insert(tk.END, json.dumps(data, ensure_ascii=False,
indent=2))
            self.refresh_history()
            self.populate_combos()
            self.refresh_table()
        except Exception as e:
            self._gpt_fail(e, traceback.format_exc())

    def _gpt_fail(self, e, tb):
        self._stop_spinner("❌ Помилка")
        self.txt_preview.insert(tk.END, f"❌ Помилка: {e}\n{tb}")
        messagebox.showerror("GPT", f"Не вдалося виконати аналіз:\n{e}")

    # ---------- TAB: TABLES ----------
    def build_tables(self):
        frm = ttk.Frame(self.tab_tables, padding=8); frm.pack(fill=BOTH,
expand=True)

        top = ttk.Frame(frm); top.pack(fill=X)
        ttk.Label(top, text="Виберіть запит:", width=14).pack(side=LEFT)
        self.cmb_table = ttk.Combobox(top, state="readonly", width=110)
        self.cmb_table.pack(side=LEFT, fill=X, expand=True, padx=6, pady=4)
        ttk.Button(top, text="Обрати", bootstyle=PRIMARY,
command=self.pick_from_table_combo).pack(side=LEFT, padx=6)

        act = ttk.Frame(frm); act.pack(fill=X, pady=4)
        ttk.Button(act, text="Експорт CSV", bootstyle=INFO, command=lambda:
self.export_table("csv")).pack(side=RIGHT)
        ttk.Button(act, text="Експорт XLSX", bootstyle=INFO, command=lambda:
self.export_table("xlsx")).pack(side=RIGHT, padx=6)

        self.lbl_cur = ttk.Label(frm, text="Поточний запит: (не вибрано)",
bootstyle=SECONDARY)
        self.lbl_cur.pack(anchor="w", pady=(2,6))

        box = ttk.Labelframe(frm, text="Ризики", padding=6); box.pack(fill=BOTH,
expand=True)
        cols = ("#", "Назва", "P", "I", "Опис")
        self.tree = ttk.Treeview(box, columns=cols, show="headings", height=16)
        for c,w in zip(cols,(50,420,60,60,650)):
            self.tree.heading(c,text=c); self.tree.column(c,width=w,anchor="w")
        self.tree.pack(fill=BOTH, expand=True)

    def populate_table_combo(self):
        if not self.user: return
        items = self.db.user_queries(self.user["id"])
        self._combo_map_table = []
        labels=[]
        for (qid,ts,mode,prompt) in items:
            s = textwrap.shorten((prompt or "").replace("\n"," "), 80,
placeholder="…")
            labels.append(f"ID {qid} | {ts[:19]} | {mode} | {s}")
            self._combo_map_table.append(qid)
        self.cmb_table["values"] = labels
        if self.current_qid:
            try:
                idx = [i for i,q in enumerate(self._combo_map_table) if
q==self.current_qid][0]
                self.cmb_table.current(idx)
```

```
            except: pass

    def pick_from_table_combo(self):
        idx = self.cmb_table.current()
        if idx < 0:
            messagebox.showwarning("Увага","Оберіть запит."); return
        self.current_qid = self._combo_map_table[idx]
        self.refresh_table()
        messagebox.showinfo("OK", f"Обрано запит ID {self.current_qid}")

    def export_table(self, fmt="csv"):
        if not self.current_qid:
            messagebox.showwarning("Увага","Оберіть запит."); return
        rows = self.db.risks_for_query(self.current_qid)
        if not rows:
            messagebox.showwarning("Увага","Немає даних."); return
        df = pd.DataFrame(rows,
columns=["name","description","probability","impact"])
        fname = filedialog.asksaveasfilename(defaultextension=f".{fmt}",
                    filetypes=[("CSV","*.csv"),("Excel","*.xlsx"),("All","*.*")])
        if not fname: return
        (df.to_csv if fmt=="csv" else df.to_excel)(fname, index=False)
        messagebox.showinfo("OK", f"Експортовано: {fname}")

    def refresh_table(self):
        for i in self.tree.get_children(): self.tree.delete(i)
        if not self.current_qid:
            self.lbl_cur.configure(text="Поточний запит: (не вибрано)"); return
        rows = self.db.risks_for_query(self.current_qid)
        for idx,(name,desc,p,i) in enumerate(rows, start=1):
            self.tree.insert("", "end", values=(idx,name,p,i,desc))
        self.lbl_cur.configure(text=f"Поточний запит: ID {self.current_qid}")
        self.populate_table_combo()

    # ---------- TAB: CHARTS ----------
    def build_charts(self):
        frm = ttk.Frame(self.tab_charts, padding=8); frm.pack(fill=BOTH,
expand=True)
        sel = ttk.Labelframe(frm, text="Вибір запиту", padding=6);
sel.pack(fill=X)
        self.cmb_charts = ttk.Combobox(sel, state="readonly", width=110)
        self.cmb_charts.pack(side=LEFT, fill=X, expand=True, padx=6, pady=4)
        ttk.Button(sel, text="Обрати", bootstyle=PRIMARY,
command=self.pick_from_charts_combo).pack(side=LEFT, padx=6)
        ttk.Button(sel, text="Оновити список", bootstyle=SECONDARY,
command=self.populate_combos).pack(side=LEFT)

        left = ttk.Frame(frm); left.pack(side=LEFT, fill=Y, padx=(0,8))
        ttk.Button(left, text="Матриця P×I", bootstyle=PRIMARY,
command=self.show_matrix).pack(fill=X, pady=6)
        ttk.Button(left, text="Fault Tree",  bootstyle=PRIMARY,
command=self.show_fault).pack(fill=X, pady=6)
        ttk.Button(left, text="Decision Tree", bootstyle=PRIMARY,
command=self.show_decision).pack(fill=X, pady=6)

        # Правий блок з динамічним ресайзом
        self.chart_area = ttk.Frame(frm); self.chart_area.pack(side=RIGHT,
fill=BOTH, expand=True)
        self.fig = Figure(figsize=(7,5), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.chart_area)
        self.canvas.get_tk_widget().pack(fill=BOTH, expand=True)
        # підганяти фігуру під розмір фрейму
        self.chart_area.bind("<Configure>", self._on_chart_resize)
```

```
        self.populate_combos()

    def _on_chart_resize(self, event):
        if event.width < 50 or event.height < 50: return
        dpi = self.fig.get_dpi()
        self.fig.set_size_inches(event.width/dpi, event.height/dpi, forward=True)
        self.canvas.draw_idle()

    def populate_charts_combo(self):
        if not self.user: return
        items = self.db.user_queries(self.user["id"])
        self._combo_map_charts = []
        labels=[]
        for (qid,ts,mode,prompt) in items:
            s = textwrap.shorten((prompt or "").replace("\n"," "), 80,
placeholder="…")
            labels.append(f"ID {qid} | {ts[:19]} | {mode} | {s}")
            self._combo_map_charts.append(qid)
        self.cmb_charts["values"] = labels
        if self.current_qid:
            try:
                idx = [i for i,q in enumerate(self._combo_map_charts) if
q==self.current_qid][0]
                self.cmb_charts.current(idx)
            except: pass

    def populate_combos(self):
        self.populate_table_combo()
        self.populate_charts_combo()

    def pick_from_charts_combo(self):
        idx = self.cmb_charts.current()
        if idx < 0:
            messagebox.showwarning("Увага","Оберіть запит."); return
        self.current_qid = self._combo_map_charts[idx]
        self.refresh_table()
        messagebox.showinfo("OK", f"Обрано запит ID {self.current_qid}")

    def show_matrix(self):
        if not self.current_qid: messagebox.showwarning("Увага","Оберіть запит.");
return
        rows = self.db.risks_for_query(self.current_qid)
        if not rows: messagebox.showwarning("Увага","Немає ризиків."); return
        df = pd.DataFrame(rows,
columns=["name","description","probability","impact"])
        draw_risk_matrix(self.fig, df); self.canvas.draw_idle()

    def show_fault(self):
        if not self.current_qid: messagebox.showwarning("Увага","Оберіть запит.");
return
        rows = self.db.fault_nodes_for_query(self.current_qid)
        nodes = [{"node_id":r[0],"label":r[1],"node_type":r[2],"parent_id":r[3]}
for r in rows]
        render_fault_tree(self.fig, nodes); self.canvas.draw_idle()

    def show_decision(self):
        if not self.current_qid: messagebox.showwarning("Увага","Оберіть запит.");
return
        rows = self.db.decision_nodes_for_query(self.current_qid)
        nodes = [{"node_id":r[0],"label":r[1],"node_type":r[2],"parent_id":r[3],
                  "edge_prob":r[4], "payoff":r[5]} for r in rows]
        render_decision_tree(self.fig, nodes); self.canvas.draw_idle()
```

```
    # ---------- TAB: HISTORY ----------
    def build_history(self):
        frm = ttk.Frame(self.tab_hist, padding=8); frm.pack(fill=BOTH,
expand=True)
        cols = ("ID","Час","Режим","Початок промпту")
        self.hist = ttk.Treeview(frm, columns=cols, show="headings", height=16)
        for c,w in zip(cols,(80,170,120,800)):
            self.hist.heading(c,text=c); self.hist.column(c,width=w,anchor="w")
        self.hist.pack(fill=BOTH, expand=True)
        self.hist.bind("<<TreeviewSelect>>", self.on_hist_select)
        self.refresh_history()

    def refresh_history(self):
        for i in self.hist.get_children(): self.hist.delete(i)
        items = self.db.user_queries(self.user["id"])
        for (qid,ts,mode,prompt) in items:
            s = textwrap.shorten((prompt or "").replace("\n"," "), 120,
placeholder="…")
            self.hist.insert("", "end", values=(qid,ts[:19],mode,s))

    def on_hist_select(self, _evt=None):
        sel = self.hist.selection()
        if not sel: return
        vals = self.hist.item(sel[0])["values"]
        self.current_qid = int(vals[0])
        self.refresh_table(); self.populate_combos()


if __name__ == "__main__":
    _ = Database(DB_FILE)
    root = ttk.Window(themename=THEME)
    try:
        root.iconbitmap("icon.ico")
    except:
        pass
    app = App(root)
    root.mainloop()
```