Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ПРАЦІВНИКІВ ОРГАНІЗАЦІЇ

Виконав: студент 4 курсу, групи 1КН-21б.
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Владислав ВОЛОХ
(прізвище та ініціали)

Керівник: к.т.н, доцент. каф. КН
Руслан БЕЛЗЕЦЬКИЙ
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н. проф. каф. АІІТ
Віктор МІЗЕРНИЙ
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри КН д.т.н. проф. Андрій ЯРОВИЙ
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
"05" травня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Владиславу ВОЛОХУ

1. Тема роботи: Розробка програмного додатку оцінювання ефективності працівників організації.
Керівник роботи: Белзецький Руслан Станіславович, асистент.
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97

2.Термін подання студентом роботи «30» травня 2025 р.

3. Вихідні дані до роботи:
мова програмування C#, середовище розробки Visual Studio, використання реляційних БД.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
Аналіз предметної області;
Розробка програмного додатку оцінювання ефективності працівників організації;
Програмна реалізація додатку оцінювання ефективності працівників організації.

Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
UML-діаграми програмного модуля;
Загальний вигляд інтерфейсу програмного модуля.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Белзецький Р.С., к.т.н., доц. каф.КН		

6. Дата видачі завдання «05» травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області цифрових засобів оцінювання працівників	05.05.25	07.05.25	
2	Розробка програмного модуля оцінювання ефективності працівників організації	08.05.25	11.05.25	
3	Ппрограмна реалізація додатку оцінювання ефективності працівників організації	12.05.25	15.05.25	
4	Висновки	16.05.25	26.05.25	
5	Оформлення додатків	27.05.25	29.05.25	
6	Розробка інструкції користувача	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БДР	02.06.25	04.06.25	

Студент _____ Волох В.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Белзецький Р.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Волох В. С. Розробка програмного додатку оцінювання ефективності працівників організації. Бакалаврська кваліфікаційна робота складається з 83 сторінок формату А4, на яких є 15 рисунків, список використаних джерел містить 20 найменувань.

У роботі проведено аналіз предметної області цифрових засобів оцінювання працівників, розглянуто сучасні дослідження у цій сфері та відомі програмні аналоги. Сформульовано постановку задачі щодо створення програмного модуля для оцінювання ефективності працівників організації. Розроблено структурну схему роботи програмного модуля та побудовано UML-діаграми, що відображають основні процеси. Виконано програмну реалізацію модуля із використанням сучасних засобів розробки, проведено тестування працездатності програмного продукту. Результатом роботи є програмний модуль, який забезпечує об’єктивне оцінювання ефективності працівників на основі визначених критеріїв.

Ключові слова: управління персоналом, критерій оцінювання, ефективність працівників, UML-моделювання.

ABSTRACT

Volokh V. S. Development of a Software Application for Evaluating Employee Performance.The Bachelor's thesis consists of 83 A4 pages and includes 15 figures. The list of references comprises 20 sources.

The work presents an analysis of the subject area of digital tools for employee evaluation, examines current research in this field, and reviews well-known software analogs. The task of creating a software module for evaluating employee performance within an organization is formulated. A structural diagram of the module's operation is developed, along with UML diagrams that reflect the main processes. The software implementation of the module was carried out using modern development tools, followed by functionality testing.

The result of the work is a software module that ensures objective evaluation of employee performance based on predefined criteria.

Keywords: human resource management, evaluation criteria, employee efficiency, UML modeling.

ЗМІСТ

ВСТУП

Актуальність дослідження. В умовах високої конкуренції, динамічних змін ринку праці та необхідності оперативного прийняття обґрунтованих управлінських рішень, ефективне оцінювання персоналу стає одним із ключових чинників успішності організації. Традиційні методи оцінювання часто є суб’єктивними, неструктурованими або надто трудомісткими, що знижує їхню об'єктивність та оперативність. Саме тому виникає нагальна потреба у створенні сучасних програмних рішень, які забезпечують автоматизацію цього процесу, зменшують вплив людського чинника та сприяють прийняттю виважених управлінських рішень.

Використання програмного забезпечення для оцінювання ефективності працівників дозволяє інтегрувати різноманітні критерії оцінки – такі як продуктивність, дотримання термінів, якість виконання завдань, ініціативність, командна робота тощо – у єдину систему, що забезпечує комплексний підхід до аналізу кадрового потенціалу. Застосування програмного додатку дозволяє збирати, зберігати та аналізувати великі обсяги інформації про результати діяльності працівників, що відкриває можливості для впровадження інструментів HR-аналітики, побудови індивідуальних траєкторій розвитку персоналу та підвищення ефективності системи мотивації.

Цифрові інструменти оцінювання є важливим елементом загальної стратегії цифрової трансформації підприємств, оскільки сприяють автоматизації рутинних процесів, підвищують прозорість управління та покращують комунікацію між керівниками та підлеглими. [1,4] Розробка та впровадження такого програмного додатку є важливим кроком у напрямі підвищення рівня організаційної зрілості підприємства та його адаптаційної здатності до сучасних викликів.

Метою дослідження є розширення функціональних можливостей автоматизованого оцінювання ефективності працівників організації з урахуванням сучасних вимог до управління персоналом, що дозволить підвищити об’єктивність, оперативність та результативність процесу оцінювання.

Також можна виділити наступні задачі дослідження:

1. Проаналізувати існуючі підходи, методики та критерії оцінювання ефективності працівників у сучасних організаціях.

2. Дослідити програмні рішення, що вже використовуються в сфері HR-менеджменту, та визначити їх сильні і слабкі сторони.

3. Сформулювати вимоги до функціоналу програмного додатку відповідно до потреб потенційних користувачів.

4. Розробити архітектуру та інтерфейс програмного додатку з урахуванням зручності використання та можливостей масштабування.

5. Реалізувати програмний продукт з використанням відповідних засобів розробки програмного забезпечення.

6. Провести тестування розробленого додатку на предмет функціональності, надійності та зручності користування.

7. Оцінити ефективність використання розробленого рішення в умовах реальної або модельованої організаційної структури.

Об’єкт дослідження – процес створення програмного додатку для підвищення ефективності оцінювання працівників організації.

Предмет дослідження – методи та засоби для оцінювання ефективності працівників організації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЦИФРОВИХ ЗАСОБІВ ОЦІНЮВАННЯ ПРАЦІВНИКІВ

1.1 Аналіз сучасних досліджень в області цифрових засобів оцінювання працівників

У сучасних умовах цифрової трансформації особливу увагу науковців, аналітиків і практиків привертає проблема ефективного оцінювання персоналу за допомогою цифрових засобів. Це зумовлено як зростанням обсягів інформації, що потребує аналізу в процесі прийняття управлінських рішень, так і прагненням підприємств до автоматизації та оптимізації внутрішніх процесів. У центрі досліджень у цій галузі знаходиться пошук оптимальних інструментів, які б забезпечували об’єктивне, швидке та комплексне оцінювання результативності праці співробітників. Різноманітні цифрові рішення, зокрема HR-аналітика, системи керування продуктивністю (Performance Management Systems), платформи з елементами штучного інтелекту, дедалі частіше використовуються для формування об'єктивної картини ефективності працівників.

Огляд наукових джерел засвідчує, що однією з ключових тенденцій є поступове зміщення фокусу від формального оцінювання, яке базується на кількісних показниках, до інтегративного аналізу, що поєднує якісні та поведінкові характеристики. Дослідження, проведені в галузі менеджменту та інформаційних технологій, акцентують увагу на тому, що цифрові інструменти дозволяють враховувати не лише продуктивність у класичному розумінні, а й показники залученості, рівень співпраці, лояльність до організації, ініціативність тощо. Особливого поширення набувають платформи, які інтегруються з іншими системами управління (ERP, CRM), що дозволяє оцінювати ефективність персоналу у взаємозв’язку з фінансовими та операційними показниками підприємства [1,2].

Значна частина сучасних досліджень присвячена застосуванню елементів машинного навчання та штучного інтелекту для аналізу поведінкових патернів працівників та прогнозування їх продуктивності. Такі системи аналізують великі

масиви даних, що збираються з електронної пошти, корпоративних месенджерів, таск-трекерів, систем контролю робочого часу, і на основі цього формують індикатори ефективності, виявляючи тенденції та відхилення від норми. Окремі автори звертають увагу на ризики упередженості алгоритмів, пов’язані з якістю даних та налаштуванням моделей, що вимагає ретельного підходу до проектування подібних систем.

У контексті розвитку цифрових засобів оцінювання активно обговорюється концепція безперервного зворотного зв’язку (continuous feedback), яка реалізується за допомогою спеціалізованих платформ. Такі платформи дозволяють менеджерам та колегам залишати відгуки у реальному часі, формуючи динамічну картину ефективності працівника, що є більш інформативною у порівнянні з традиційними щорічними оцінками. Дослідження підтверджують, що така модель сприяє підвищенню мотивації персоналу, формує культуру відкритого зворотного зв’язку та підтримує розвиток працівників у режимі реального часу [3-5].

Не менш важливим аспектом у наукових дослідженнях є питання етичності та приватності використання цифрових інструментів для моніторингу персоналу. Науковці наголошують на необхідності дотримання балансу між інтересами роботодавця та правами працівника на недоторканість особистого простору [6]. У цьому контексті підкреслюється роль прозорої політики організації щодо збору, обробки та використання персональних даних у межах системи оцінювання.

У цілому, аналіз сучасних досліджень свідчить про високий рівень зацікавленості до теми цифрової трансформації оцінювання працівників, що обумовлено як прагненням до ефективності, так і викликами, що виникають у процесі реалізації таких інструментів. Водночас у науковому середовищі визнається, що успішне впровадження цифрових засобів оцінювання вимагає комплексного підходу: технічна реалізація має бути поєднана з методологічно обґрунтованими критеріями оцінки та чітко визначеними етичними рамками. Саме це відкриває перспективи для подальших досліджень у цій сфері та підкреслює важливість тематики даної дипломної роботи.

1.2 Аналіз відомих програм аналогів

Проаналізуємо відомі програми-аналоги та визначимо їх ключові особливості, переваги та недоліки.

SAP SuccessFactors – це одна з найпотужніших та найпоширеніших систем для управління людськими ресурсами, яка включає модуль Performance & Goals, спеціально розроблений для оцінювання ефективності працівників. [1] Ця система дає змогу створювати персоналізовані цілі, моніторити їх виконання та здійснювати багатовимірну оцінку продуктивності. У SuccessFactors активно використовується підхід "continuous performance management" – постійний моніторинг ефективності з регулярним фідбеком, що сприяє динамічному розвитку працівника.

Перевагами цієї платформи є інтеграція з іншими HR-модулями (підбір, навчання, компенсації), гнучка налаштовуваність метрик, можливість 360-градусної оцінки та інтерфейс з елементами аналітики. Програмне забезпечення дозволяє керівникам відстежувати індивідуальні плани розвитку, що базуються на результатах оцінювання, а також формувати зведені звіти для стратегічного планування. Основним недоліком є складність впровадження в малих організаціях через високу вартість та потребу у спеціалізованих знаннях для налаштування.

SAP SuccessFactors вирізняється глибоким функціональним наповненням, яке дозволяє не лише проводити оцінювання ефективності персоналу, а й формувати стратегічне бачення розвитку людського капіталу в організації. Однією з ключових особливостей є можливість створення індивідуальних карт продуктивності, які враховують як досягнення працівника за встановленими цілями, так і його поведінкові характеристики, участь у командній роботі, рівень професійного розвитку. У системі широко застосовуються автоматизовані рекомендації щодо навчання, що дозволяє пов’язати результати оцінювання із подальшими освітніми траєкторіями.

Ще однією перевагою SAP SuccessFactors є високий рівень аналітичної підтримки. Платформа пропонує інструменти глибокої HR-аналітики, які дають змогу керівництву проводити порівняльний аналіз ефективності співробітників,

виявляти закономірності у динаміці продуктивності, прогнозувати ризики вигорання або плинності кадрів. Усе це робить SAP SuccessFactors не лише засобом оцінювання, а повноцінним інструментом прийняття стратегічних HR-рішень на основі об’єктивних даних, що особливо цінно для великих організацій зі складною структурою.

На рисунку 1.1 наведено інтерфейс даного додатку.

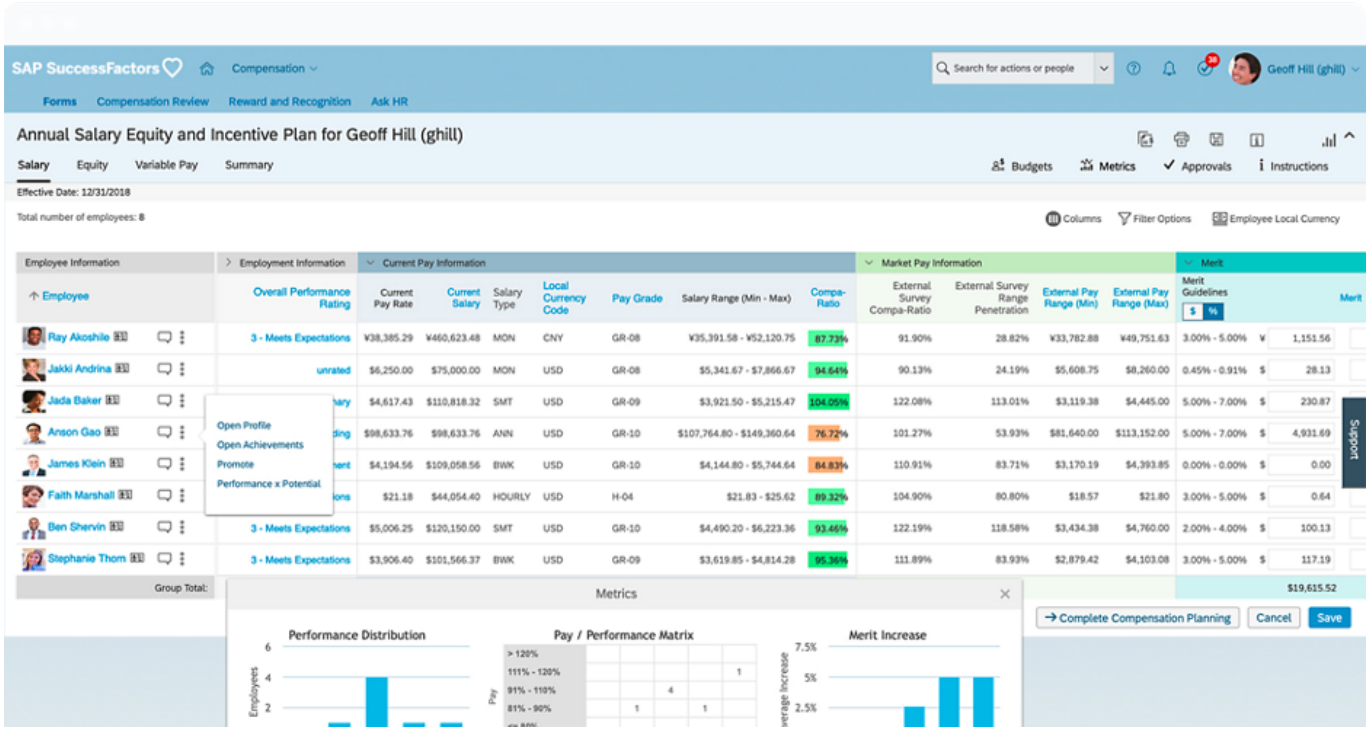

Рисунок 1.1 – Інтерфейс додатку SAP SuccessFactors

BambooHR – це хмарна HR-система, орієнтована переважно на малі та середні компанії. [2] Вона має модуль Performance Management, який підтримує циклічне оцінювання персоналу, функцію збирання зворотного зв’язку від колег, а також щоквартальні або щорічні огляди результатів роботи. BambooHR дозволяє легко налаштовувати шаблони оцінок, визначати ключові компетенції та KPI, а також аналізувати динаміку змін ефективності за обраний період.

Важливою перевагою цього ПЗ є його інтуїтивно зрозумілий інтерфейс, швидке налаштування без необхідності тривалого навчання персоналу, а також

демократична ціна. Програмний продукт пропонує гнучкі варіанти інтеграції з іншими інструментами, такими як Slack, G Suite, Microsoft Teams. Проте в системі відсутня глибока аналітика, характерна для більш складних рішень, що обмежує її використання в масштабних організаціях з великою кількістю працівників (рис.1.2).

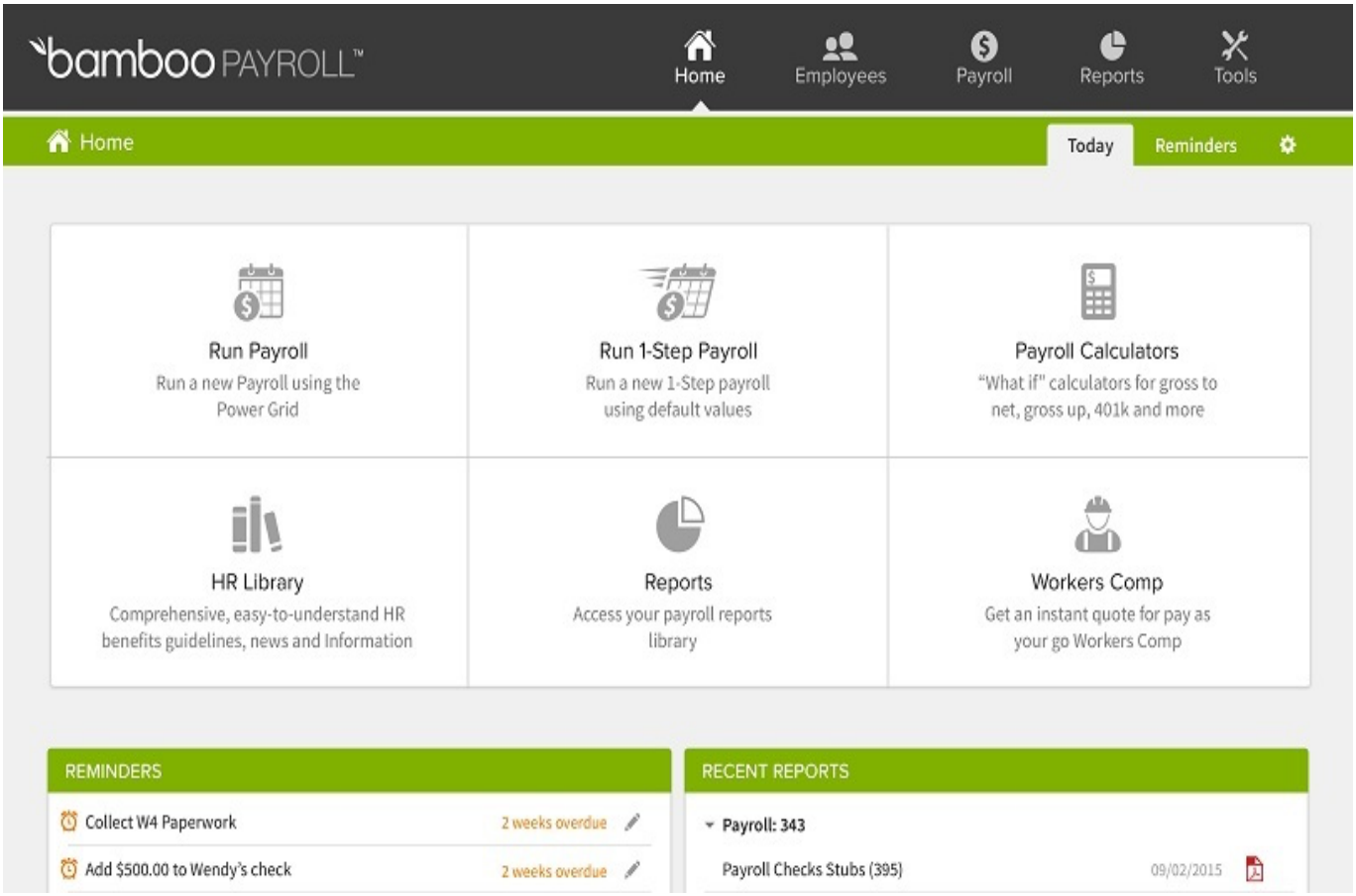

Рисунок 1.2 – Інтерфейс додатку BambooHR

Zoho People – це комплексна платформа для управління людськими ресурсами, яка також включає модуль Performance Appraisal, що дозволяє оцінювати працівників на основі декількох параметрів – результатів виконання завдань, поведінкових характеристик, досягнення цілей тощо. [3] Система підтримує 360-градусну оцінку, де працівник отримує фідбек від колег, підлеглих, керівництва та навіть клієнтів.

Zoho People дозволяє формувати матрицю компетенцій для кожної посади, створювати опитування, формувати графіки проведення оцінок, а також автоматизувати процес узгодження результатів із відповідальними особами. Завдяки можливості інтеграції з інструментами бізнес-аналітики від Zoho, користувачі мають змогу глибше аналізувати продуктивність команд і підрозділів. Програма має гнучку модель ліцензування, але інтерфейс може бути дещо перевантаженим для нових користувачів, і для повноцінного впровадження потрібна попередня адаптація під потреби організації.

На рисунку 1.3 можна побачити інтерфейс даного додатку.

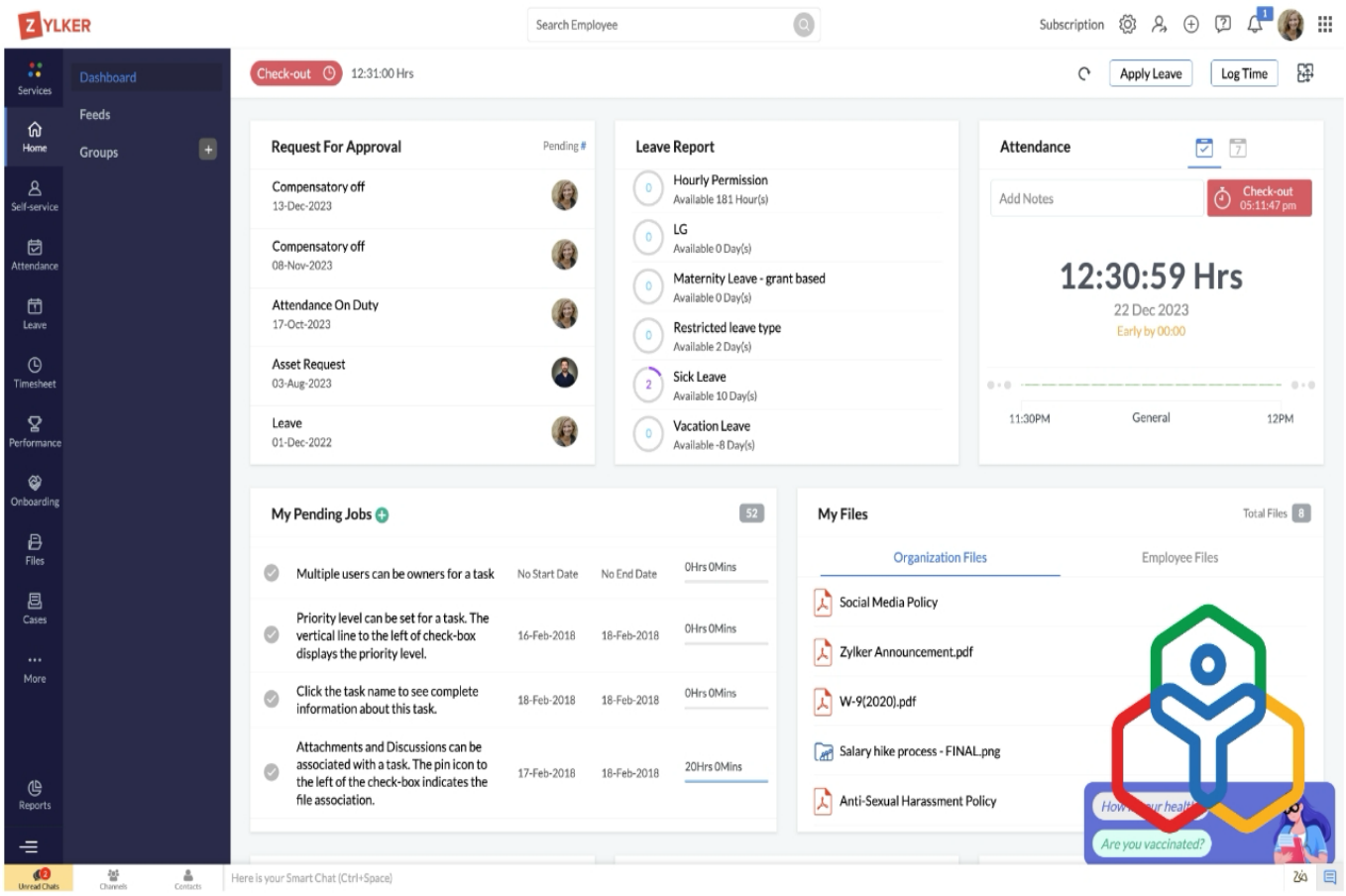

Рисунок 1.3 – Інтерфейс додатку Zoho People

Lattice – це сучасний інструмент, створений спеціально для управління ефективністю та залученістю працівників. На відміну від багатьох інших систем,

Lattice фокусується на створенні культури постійного розвитку та взаємного зворотного зв’язку. Система дозволяє керівникам задавати цілі, формувати огляди продуктивності, ініціювати щотижневі 1:1 зустрічі та використовувати інструменти для опитувань про залученість персоналу (рис.1.4).

Lattice підтримує OKR-підхід, що дозволяє працівникам бачити, як їх особисті цілі узгоджуються з цілями компанії. Також реалізована система похвали («praise»), яка дозволяє колегам визнавати досягнення одне одного в публічному або приватному форматі. Платформа надає вбудовані дашборди з візуалізацією ключових показників ефективності, а також автоматизоване нагадування про необхідність оновлення цілей або проходження чергового етапу оцінки. З мінусів – програмне забезпечення більше орієнтоване на англомовну аудиторію.

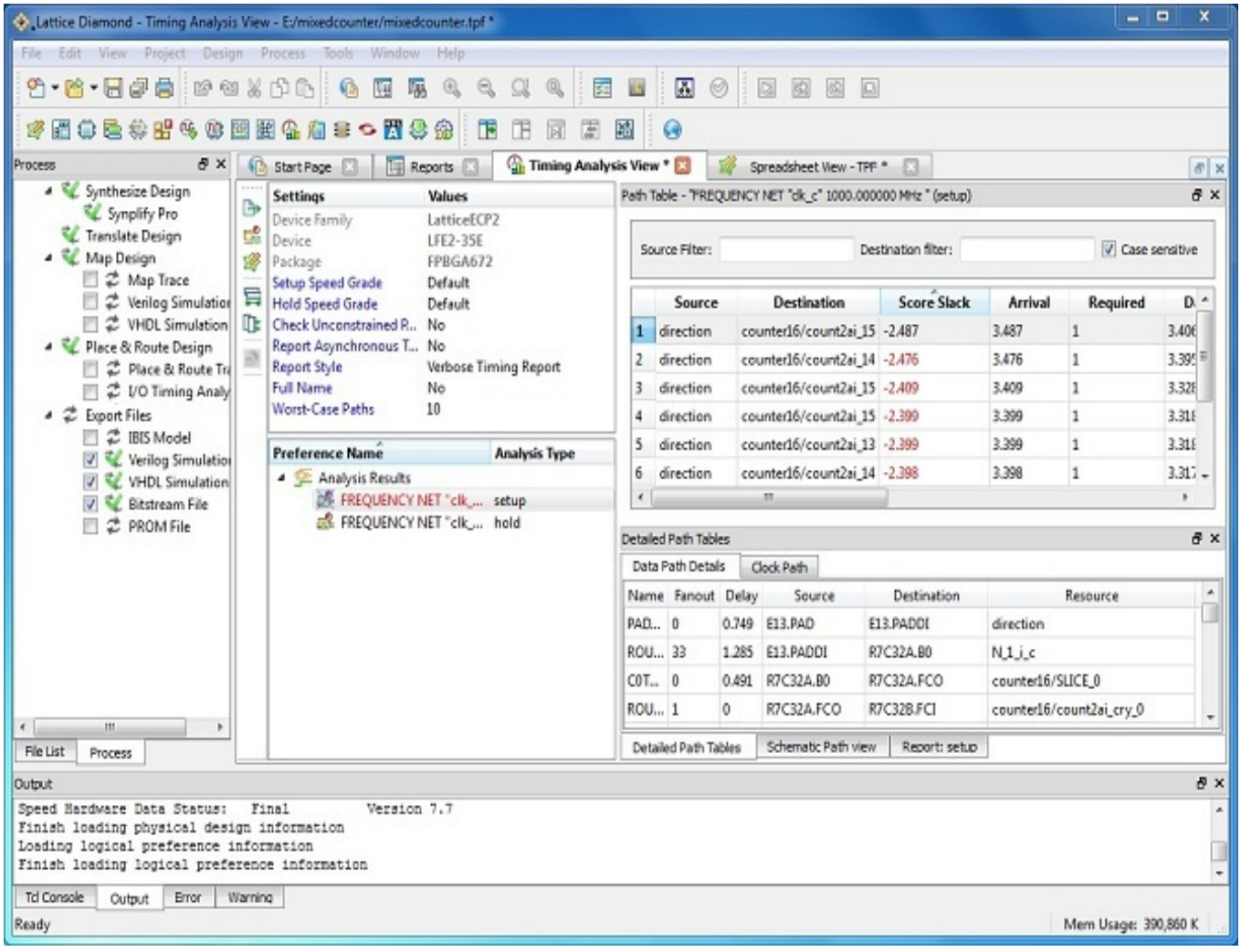

Рисунок 1.4 – Інтерфейс додатку Lattice

В таблиці 1.1 можна побачити порівняльну характеристику усіх вище перерахованих додатків.

Таблиця 1.1 – Порівняльна характеристика програм аналогів

Назва додатку	Переваги	Недоліки
SAP SuccessFactors	– Потужна аналітика та глибока HR-інтеграція – Підтримка 360° оцінювання – Гнучка система цілей і KPI – Автоматизовані освітні рекомендації	– Висока вартість впровадження – Потреба у спеціальній підготовці персоналу
BambooHR	– Простий та інтуїтивний інтерфейс – Швидке впровадження – Ідеально підходить для малих і середніх компаній – Інтеграція з популярними сервісами	– Обмежені аналітичні можливості – Відсутність складної персоналізації оцінювання
Zoho People	– 360° оцінка з широким налаштуванням параметрів – Модуль компетенцій – Можливість інтеграції з BI – Гнучка система налаштувань	– Перевантажений інтерфейс – Потрібен час на адаптацію під специфіку організації
Lattice	– Сильний фокус на залученість і розвиток – OKR-модель – Інтерактивні дашборди та система "Praise" – Постійний фідбек і опитування	– Орієнтованість на англомовну аудиторію – Відсутність локалізації українською мовою

Таким чином, за результатами порівняльного аналізу додаток SAP SuccessFactors має найкращу ефективність в плані можливостей аналітики, тоді як Zoho People є найменш зручним з точки зору користувача. Виходячи з цього, необхідно розробити додаток, який містить переваги SAP SuccessFactors, але є безкоштовним для користувачів.

1.3 Постановка задачі

Опишемо функціональні вимоги на розробку програмного додатку оцінювання ефективності працівників організації.

1. Загальний опис системи

Система призначена для оцінювання ефективності роботи працівників за різними параметрами, управління структурою департаментів та відстеження прогресу співробітників. Система повинна забезпечувати ієрархічну структуру управління з різними рівнями доступу.

2. Аутентифікація та авторизація

- Система повинна забезпечувати вхід користувачів через логін/пароль
- Підтримка різних ролей користувачів: адміністратор, керівник (Head), керівник команди (TeamLead), звичайний користувач (user)
- Забезпечення захищеного доступу до різних розділів системи залежно від ролі
- Можливість виходу з системи

3. Управління параметрами оцінювання

- Можливість створення нових параметрів оцінювання з визначенням:
  - Назви параметра
  - Коефіцієнта важливості
  - Опису для кожної з п'яти оцінок за п’ятибальною шкалою Лайкерта (від "жахливо" до "чудово")
- Перегляд списку всіх параметрів з їх характеристиками
- Редагування існуючих параметрів

- Перегляд детальної інформації про параметр
- Видалення параметрів з підтвердженням

4. Управління департаментами

- Створення нових департаментів
- Призначення керівника департаменту
- Перегляд списку всіх департаментів
- Редагування інформації про департамент
- Видалення департаментів з підтвердженням
- Перегляд структури департаментів

5. Управління працівниками

- Створення облікових записів працівників з наступними даними:
  - Ім'я
  - Прізвище
  - Email
  - Пароль
  - Роль в системі
  - Статус
  - Керівник
  - Департамент
- Перегляд списку всіх працівників
- Редагування інформації про працівника
- Видалення працівника з підтвердженням
- Фільтрація працівників за департаментами

6. Оцінювання працівників

- Можливість оцінювати працівників за визначеними параметрами (шкала від 1 до 5)
- Збереження історії оцінювань
- Розрахунок загальної ефективності на основі оцінок та коефіцієнтів параметрів
- Визначення прогресу працівника ("Stagnation", "Progress up" тощо)

- Виділення працівників з прогресом ("Progressing workers")
- Виділення працівників з проблемами ("Looping employees")

7. Сторінка департаменту

- Відображення інформації про департамент
- Відображення керівника департаменту
- Перегляд списку працівників департаменту з їх ефективністю та прогресом
- Перемикання між переглядом працівників, параметрів та груп департаменту

8. Звітність

- Формування звітів про ефективність працівників
- Відображення прогресу працівників
- Можливість порівняння ефективності різних департаментів

9. Інтерфейс користувача

- Інтуїтивно зрозумілий інтерфейс
- Навігаційне меню з доступом до основних розділів: параметри, департаменти, працівники, структура департаментів
- Відображення інформації в табличному вигляді
- Кнопки для основних дій: створення, редагування, видалення, перегляд деталей
- Форми для введення та редагування даних
- Підтвердження критичних дій (наприклад, видалення)
- Кнопки повернення на попередні сторінки

10. Системні вимоги

- Веб-інтерфейс з адаптивним дизайном
- Підтримка сучасних браузерів
- Захищене з'єднання через HTTPS
- Локальне розгортання (localhost)
- Швидкий відгук інтерфейсу

11. Масштабованість та розширюваність

- Можливість додавання нових параметрів оцінювання
- Підтримка зростаючої кількості працівників та департаментів
- Можливість розширення функціоналу без значних змін архітектури

12. Безпека

- Шифрування паролів
- Контроль доступу на рівні ролей
- Захист від несанкціонованого доступу
- Валідація введених даних

1.4 Висновки

У першому розділі бакалаврської кваліфікаційної роботи було здійснено всебічний аналіз сучасного стану досліджень і практичного використання цифрових засобів оцінювання персоналу. Встановлено, що в умовах цифрової трансформації підприємств особливого значення набуває впровадження інтегрованих інструментів, які поєднують аналітичні можливості, автоматизацію та етичні підходи до моніторингу ефективності працівників. Дослідження свідчать про зростання ролі таких елементів, як машинне навчання, безперервний зворотний зв’язок та повна інтеграція з ERP/CRM-системами.

Огляд програм-аналогів (SAP SuccessFactors, BambooHR, Zoho People, Lattice) дозволив виокремити основні функціональні особливості та переваги сучасних систем управління персоналом: гнучкість налаштування, підтримка 360-градусного оцінювання, наявність аналітичних модулів, а також орієнтація на підвищення залученості та розвитку персоналу. Водночас виявлено обмеження окремих рішень, пов’язані з складністю впровадження, високою вартістю або недостатньою глибиною аналітики.

Отже, результати порівняльної характеристики програм-аналогів підтвердили актуальність теми дослідження, виявили ключові напрями розвитку цифрових інструментів оцінювання персоналу та окреслили вимоги до створення ефективної системи оцінювання, адаптованої до потреб конкретної організації.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ПРАЦІВНИКІВ ОРГАНІЗАЦІЇ

2.1 Проектування структурної схеми роботи програмного модуля

Структурна схема системи оцінювання ефективності працівників відображає комплексну багаторівневу архітектуру програмного забезпечення, що дозволяє ефективно управляти процесом оцінювання персоналу. Архітектура системи складається з декількох логічно відокремлених рівнів, які взаємодіють між собою для забезпечення повноцінного функціонування платформи.

На верхньому рівні знаходиться клієнтська частина (Frontend), яка забезпечує інтерфейс взаємодії користувачів із системою. Ця частина включає п'ять основних модулів: модуль аутентифікації, модуль параметрів, модуль департаментів, модуль працівників та модуль оцінювання. Модуль аутентифікації відповідає за вхід користувачів в систему, перевірку облікових даних та керування сесіями, що було видно на першому знімку екрану із формою входу. Модуль параметрів надає інтерфейс для управління параметрами оцінювання, включаючи їх створення, редагування та видалення. Модуль департаментів забезпечує функціональність для роботи зі структурою компанії, зокрема для перегляду, додавання та модифікації департаментів. Модуль працівників дозволяє керувати обліковими записами персоналу - додавати нових співробітників, редагувати інформацію про існуючих та видаляти їх за потреби. Модуль оцінювання забезпечує можливість оцінювати ефективність працівників та візуалізувати результати оцінювання.

Для забезпечення взаємодії між клієнтською та серверною частинами використовується API Gateway, який виступає єдиною точкою входу для всіх запитів. API Gateway виконує важливі функції маршрутизації запитів до відповідних контролерів, валідації вхідних даних, перевірки автентифікації та авторизації користувачів, а також обробки помилок і логування запитів [8]. Така централізована точка входу дозволяє підвищити безпеку системи та спростити її адміністрування.

Серверна частина (Backend) системи складається з набору контролерів, кожен з яких відповідає за певний функціональний аспект. Контролер користувачів обробляє запити щодо аутентифікації та керування обліковими записами. Контролер параметрів відповідає за операції створення, читання, оновлення та видалення параметрів оцінювання. Контролер департаментів керує операціями з департаментами – створенням, редагуванням і видаленням. Контролер працівників займається обробкою запитів стосовно персоналу компанії. Контролер оцінювання забезпечує логіку процесу оцінювання ефективності працівників. Усі контролери взаємодіють із сервісним шаром для виконання бізнес-логіки та отримання або оновлення даних.

Сервісний шар містить основну бізнес-логіку системи та забезпечує обробку даних відповідно до бізнес-правил. Сервіс аутентифікації реалізує логіку входу користувачів, керування сесіями та ролями. Сервіс параметрів забезпечує логіку роботи з параметрами оцінювання, включаючи валідацію даних та розрахунок коефіцієнтів. Сервіс департаментів реалізує бізнес-логіку щодо структури компанії. Сервіс працівників забезпечує функціональність для управління персоналом, включаючи призначення ролей, керівників та департаментів. Сервіс оцінювання реалізує алгоритми оцінки ефективності, розрахунку прогресу та аналізу показників. Усі сервіси взаємодіють із шаром доступу до даних для отримання та збереження інформації.

Шар доступу до даних (Репозиторії) відповідає за взаємодію з базою даних та забезпечує абстракцію для верхніх шарів системи. Кожна ключова сутність системи (користувачі, параметри, департаменти, працівники, оцінки) має свій репозиторій, який надає методи для виконання операцій читання, запису, оновлення та видалення даних. Використання шаблону репозиторій дозволяє відокремити бізнес-логіку від механізмів доступу до даних та спрощує тестування системи.

На найнижчому рівні архітектури знаходиться база даних, яка забезпечує постійне зберігання всіх даних системи. В базі даних зберігається інформація про користувачів, параметри оцінювання, структуру департаментів, дані працівників,

історія оцінювань та розрахункові показники ефективності. База даних є фундаментом системи, забезпечуючи цілісність та надійність зберігання даних.

Взаємодія між компонентами системи відбувається послідовно від верхніх рівнів до нижніх. Клієнтська частина формує запити, які передаються через API Gateway до відповідних контролерів серверної частини. Контролери делегують виконання бізнес-логіки сервісам, які при необхідності звертаються до репозиторіїв для роботи з даними. Репозиторії взаємодіють з базою даних для виконання операцій зберігання та отримання інформації. Результати операцій передаються у зворотному порядку – від репозиторіїв до сервісів, від сервісів до контролерів, від контролерів через API Gateway до клієнтської частини, де вони відображаються користувачу. Така послідовна взаємодія забезпечує чітке розділення відповідальності між компонентами та спрощує розробку і підтримку системи.

Запропонована архітектура забезпечує високу модульність, що дозволяє розробляти та модифікувати компоненти незалежно один від одного. Крім того, вона сприяє масштабованості, оскільки окремі компоненти можуть бути розгорнуті на різних серверах для збільшення продуктивності системи. Важливою перевагою є також висока тестованість – кожен компонент можна тестувати окремо, що спрощує забезпечення якості програмного забезпечення. Гнучкість архітектури дозволяє змінювати реалізації окремих компонентів без впливу на інші частини системи. Нарешті, багаторівнева архітектура з API Gateway та сервісним шаром забезпечує високий рівень безпеки завдяки строгому контролю доступу. Представлена структурна схема є міцним фундаментом для розробки ефективної системи оцінювання персоналу, яка відповідає сучасним вимогам до програмного забезпечення та дозволяє досягти бізнес-цілей щодо управління ефективністю працівників. На рисунку 2.1 можна побачити структурну схему розробленого додатку.

Рисунок 2.1 – Структурна схема програмного додатку оцінювання ефективності працівників організації

2.2 Розробка UML-діаграм оцінювання ефективності працівників організації

UML (Unified Modeling Language) – це стандартизована мова моделювання, яка пропонує широкий набір засобів для аналізу, проектування та документування програмних систем. Вона поділяється на кілька груп діаграм, що дозволяють описувати різні аспекти системи.

Структурні діаграми UML показують статичну будову системи. Діаграма класів демонструє класи разом з їх атрибутами, методами та зв'язками, що дозволяє відобразити об'єктну модель системи. Діаграма компонентів ілюструє фізичну структуру – взаємозв'язки компонентів, що допомагає краще розуміти архітектуру. Діаграма пакетів дозволяє логічно групувати елементи системи в пакети та відображати залежності між ними, що особливо корисно для великих проектів.

Поведінкові діаграми UML зосереджуються на динаміці системи [9-10]. Діаграма послідовності відображає взаємодію об'єктів у вигляді послідовних повідомлень, акцентуючи увагу на часовому аспекті. Діаграма станів описує, як об'єкти змінюють свої стани у відповідь на події в процесі життєвого циклу.

UML також включає діаграми взаємодії, такі як діаграми комунікації та співпраці, що деталізують складні взаємодії об'єктів з різних точок зору. Діаграма розгортання моделює фізичне розміщення апаратних та програмних ресурсів, показуючи, як елементи системи будуть розгорнуті у реальному середовищі.

Усі типи UML-діаграм є важливими інструментами на різних етапах створення програмного забезпечення. Вони допомагають краще зрозуміти структуру та поведінку системи, а також забезпечують ефективну комунікацію між розробниками, аналітиками, дизайнерами та іншими учасниками проекту. Стандартизований підхід UML полегшує виявлення проблем і ризиків на ранніх стадіях розробки та формує міцну основу для подальшої реалізації системи.

Основна мета використання UML – полегшення спільного розуміння складних систем та забезпечення якісної комунікації в команді [7]. Візуалізація складності допомагає на ранніх етапах виявляти слабкі місця, планувати розвиток системи та оптимізувати процес розробки.

Для створення та аналізу UML-діаграм існує багато інструментів. Наприклад, Enterprise Architect – потужний засіб, що підтримує всі типи UML-діаграм, надає можливості для відстеження залежностей і генерації коду. Visual Paradigm дозволяє створювати різні діаграми, працювати колективно над моделями та повторно використовувати елементи. IBM Rational Rose, один із перших професійних

інструментів для UML, забезпечує моделювання діаграм, імпорт та експорт моделей, а також генерацію вихідного коду.

Розробка UML-діаграм базується на використанні чітких стандартів, правил та умовностей, що забезпечують однозначність і спрощують спільну роботу. Стандартизована нотація UML допомагає точно передавати задум розробників. Принцип модульності дозволяє розділяти систему на окремі компоненти для полегшення проектування та тестування, а кожна діаграма може акцентуватися на певному аспекті системи.

UML об'єднує структурне та поведінкове моделювання, дозволяючи повноцінно описувати різні сторони системи. Завдяки принципу розширюваності, можна створювати власні розширення стандартної нотації за допомогою стереотипів, тегів та обмежень, адаптуючи UML до специфіки конкретного проекту.

UML підтримує увесь життєвий цикл розробки програмного забезпечення: від аналізу вимог і проектування архітектури до реалізації, тестування та супроводу, що дозволяє зберігати єдину модель системи протягом усього часу її існування. Інструменти UML також підтримують колективну роботу над моделями – це спрощує обмін інформацією, коментування змін і управління версіями.

Завдяки своїй гнучкості та розширюваності UML підходить для моделювання різноманітних систем – не лише програмних, а й бізнес-процесів та архітектурних рішень. Він успішно використовується в різних галузях і може бути адаптований до конкретних потреб будь-якого проекту.

На рисунку 2.2 наведено UML-діаграму прецедентів програмного додатку оцінювання ефективності працівників організації.

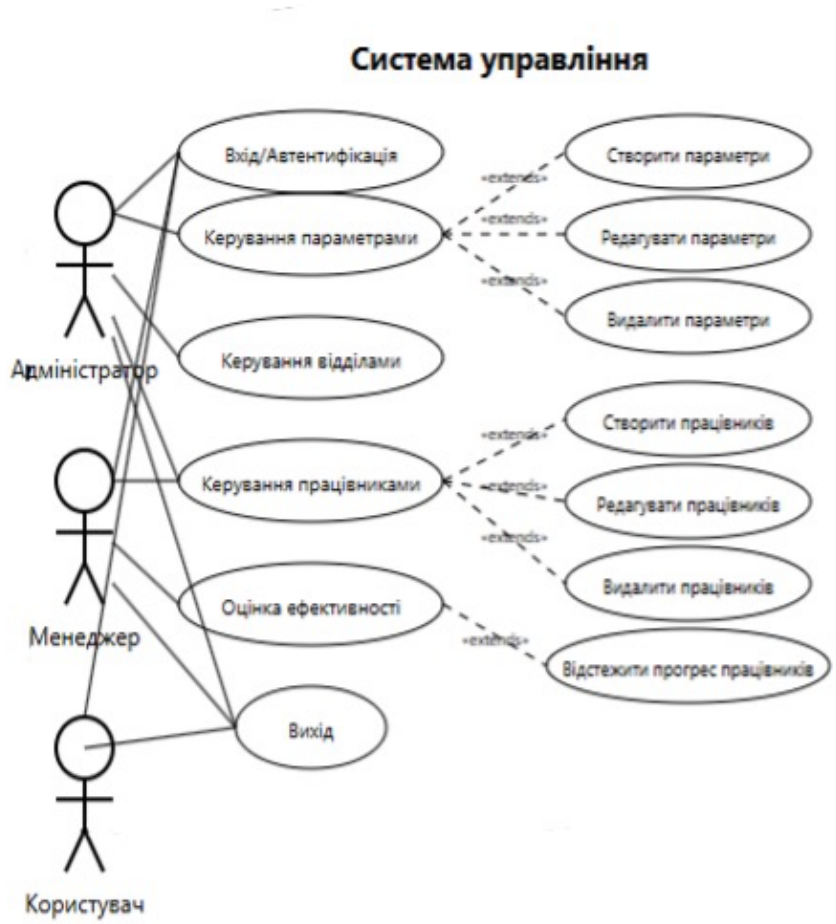

Рисунок 2.2 – UML-діаграма прецедентів програмного додатку оцінювання ефективності працівників організації

Представлена UML діаграма прецедентів відображає функціональність системи оцінки ефективності працівників, яка була зображена на наданих знімках екрану. Система є комплексним програмним рішенням для управління продуктивністю співробітників в рамках організаційної структури підприємства. Основними акторами в цій системі виступають Адміністратор, Керівник та звичайний Користувач, кожен з яких має різні рівні доступу та можливості взаємодії з системою.

Адміністратор має найвищий рівень доступу та може виконувати всі функції в системі, включаючи управління параметрами оцінювання, відділами, працівниками, а також налаштування системи в цілому. Керівник має дещо обмежені права порівняно з адміністратором, але може керувати відділами,

працівниками та проводити оцінювання продуктивності підлеглих. Звичайний користувач має найбільш обмежені права, які дозволяють йому лише входити в систему, переглядати власні оцінки та показники продуктивності.

Усі користувачі розпочинають взаємодію з системою через прецедент "Вхід/Автентифікація", де вони повинні надати правильні облікові дані (логін/електронну пошту та пароль) для отримання доступу до відповідних функцій системи. Після успішної автентифікації користувачі отримують доступ до функцій, які відповідають їхній ролі.

Управління параметрами є ключовою функцією системи, яка доступна адміністратору. Цей базовий прецедент розширюється трьома іншими: "Створення параметрів", "Редагування параметрів" та "Видалення параметрів". При створенні параметрів адміністратор задає назву, коефіцієнт важливості та опис для кожного рівня оцінки (від 1 до 5, що відповідає градаціям "жахливо", "погано", "нормально", "добре", "чудово"). Шкалу побудовано на основі п'ятибальної шкали Лайкерта.У системі вже визначено такі параметри оцінювання, як рівень відповідальності, рівень незалежності, командна комунікація, якість роботи, лідерські навички, швидкість роботи та рівень навичок.

Управління відділами дозволяє адміністратору та керівникам створювати, редагувати та видаляти відділи в структурі організації. Кожен відділ має свою назву, керівника та пов'язаних з ним працівників. Система підтримує ієрархічну структуру відділів, що дозволяє відображати організаційну структуру підприємства.

Прецедент "Управління працівниками" є базовим для роботи з даними співробітників і розширюється прецедентами "Створення працівників", "Редагування працівників" та "Видалення працівників". При створенні працівника в систему вносяться такі дані, як ім'я, прізвище, електронна пошта, пароль, роль, стан, безпосередній керівник та відділ. Система підтримує ієрархічні відносини між працівниками, де кожен працівник, крім найвищого рівня, має свого керівника.

Оцінка продуктивності є одним із головних функціональних аспектів системи, який дозволяє керівникам оцінювати своїх підлеглих за раніше визначеними параметрами. Цей прецедент розширюється прецедентом

"Відстеження прогресу", який дозволяє моніторити зміни в продуктивності працівників з часом. Система класифікує працівників на "Прогресуючих" та "Відстаючих" залежно від їхніх показників продуктивності. Кожен параметр має свій ваговий коефіцієнт, наприклад, рівень відповідальності має коефіцієнт 10, що підкреслює його важливість у загальній оцінці працівника.

Прецедент "Вихід" дозволяє користувачам будь-якого рівня безпечно завершити роботу в системі та повернутися до екрану входу.

Також для зручності подальшої розробки системи побудуємо діаграму класів (рис. 2.3).

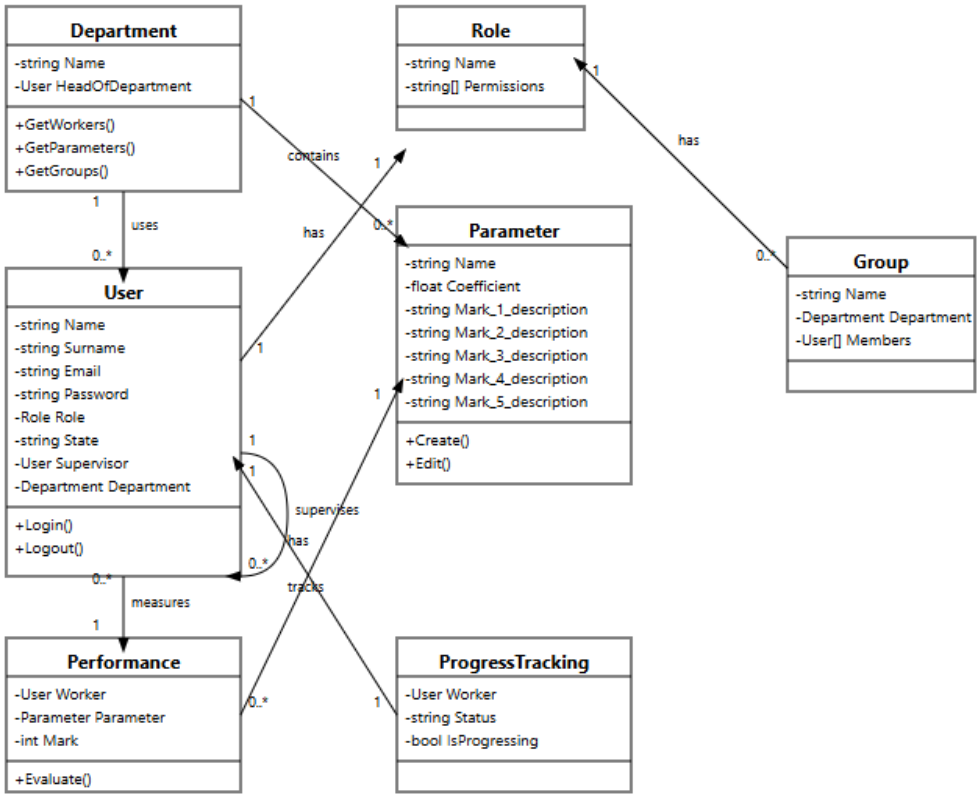

Рисунок 2.3 – UML-діаграма класів програмного додатку оцінювання ефективності працівників організації

Діаграма класів представляє архітектуру веб-додатку, який, судячи з функціональності, є системою оцінки продуктивності співробітників. Діаграма ілюструє логічну структуру додатку та зв'язки між його основними класами.

Центральними класами системи є User, Department, Parameter, Performance, Role, Group та ProgressTracking. Клас User представляє користувачів системи з атрибутами, такими як ім'я, прізвище, електронна пошта, пароль, роль, стан, керівник та відділ. Цей клас містить методи для входу та виходу з системи.

Клас Department відображає відділи компанії. Кожен відділ має назву та керівника (HeadOfDepartment). Відділ може містити працівників, параметри оцінки та групи. Клас також має методи для отримання списку працівників, параметрів, груп та відстеження прогресу співробітників.

Клас Parameter представляє параметри оцінки продуктивності працівників. Кожен параметр має назву, коефіцієнт та описи для п'яти можливих оцінок (від "horrible" до "great"). Клас містить методи для створення, редагування, видалення та перегляду деталей параметра оцінки працівника.

Клас Performance відображає оцінку працівника за певним параметром. Він містить посилання на працівника, параметр та значення оцінки. Клас має метод для проведення оцінювання.

Клас Role представляє ролі користувачів у системі, такі як адміністратор, керівник (Head), керівник команди (Teamlead) та звичайний користувач. Кожна роль має назву та набір дозволів.

Клас Group представляє групи працівників у межах відділу. Кожна група має назву, належить до відділу та містить список членів групи.

Клас ProgressTracking відстежує прогрес працівників. Він містить посилання на працівника, статус прогресу (наприклад, "Stagnation", "Progress up") та прапорець, що вказує, чи прогресує працівник.

Зв'язки між класами показують, що один користувач може керувати багатьма іншими користувачами (зв'язок User-User), відділ містить багатьох користувачів (зв'язок Department-User), користувач має багато оцінок продуктивності (зв'язок User-Performance), параметр використовується для багатьох оцінок (зв'язок Parameter-Performance), відділ використовує багато параметрів (зв'язок Department-Parameter), відділ має багато груп (зв'язок Department-Group), а для кожного користувача відстежується прогрес (зв'язок User-ProgressTracking).

Система дозволяє адміністратору керувати параметрами оцінки, створювати відділи, призначати керівників та працівників. Керівники можуть оцінювати продуктивність підлеглих за різними параметрами, а система відстежує прогрес працівників. Додаток має ієрархічну структуру управління, де кожен працівник має керівника, а відділи мають керівників відділів.

2.3 Висновки

У даному розділі було детально розглянуто проектування та розробку програмного модуля для оцінювання ефективності працівників організації. Розроблено структурну схему системи, яка демонструє багаторівневу архітектуру програмного забезпечення з чітким розподілом на клієнтську частину (Frontend), API Gateway, серверну частину (Backend), сервісний шар, шар доступу до даних (Репозиторії) та базу даних. Така багаторівнева архітектура забезпечує високу модульність, масштабованість, тестованість, гнучкість та безпеку розробленої системи.

Клієнтська частина включає наступні основні класи: User, Department, Parameter, Performance, Role, Group та ProgressTracking. API Gateway виступає єдиною точкою входу для всіх запитів, виконуючи функції маршрутизації, валідації даних та перевірки автентифікації. Серверна частина складається з набору контролерів для обробки специфічних функціональних аспектів, а сервісний шар містить основну бізнес-логіку системи. Шар доступу до даних забезпечує абстракцію для взаємодії з базою даних, а сама база даних є фундаментом системи, забезпечуючи постійне зберігання всіх необхідних даних.

В рамках проектування системи було розроблено UML-діаграму прецедентів та діаграму класів роботи системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ПРАЦІВНИКІВ ОРГАНІЗАЦІЇ

3.1 Обґрунтування вибору мови програмування і середовища розробки

Для початку реалізації системи необхідно проаналізувати відомі мови програмування і обрати одну з них. В таблиці 3.1 наведено порівняльну характеристику мов програмування для проектування програмного додатку.

Таблиця 3.1 – Порівняльна характеристика мов програмування

Мова програмування	**Призначення**	**Переваги**	**Недоліки**
C#	Застосовується для розробки операційних систем і офісних продуктів на базі Microsoft, створення веб-додатків, настільних програм (Windows Forms, WPF), серверних рішень (ASP.NET) та мобільних застосунків для W*indows* Phone.	Швидкість створення невеликих проектів, легкий та зрозумілий код, потужна IDE, підтримка спадкування і узагальнень, оптимізація виконання через CLR, зручність розробки веб-служб.	Важко адаптується для інших платформ, залежить від технологій Microsoft, потребує наявності .NET Framework, обмеження у використанні деструкторів і шаблонів.
C++	Використовується для розробки	Висока сумісність із C, підтримка різних	Складний синтаксис для

Таблиця 3.1 (продовження)

	операційних систем, драйверів, прикладного та ігрового програмного забезпечення, а також програм для вбудованих систем та серверів.	парадигм програмування, можливість кросплатформеної розробки, доступність та популярність серед розробників, гнучкість при створенні контейнерів та алгоритмів.	деяких задач, потреба у додатковому середовищі виконання для автономності програм, знижена продуктивність розробників через складність мови, що впливає на якість продуктів.
Java	Підходить для створення десктопних, мобільних та серверних додатків, а також для ігрової розробки.	Висока універсальність і багатофункціональність, підтримка різних платформ, відкритий код, велика кількість готових бібліотек, розвинена система безпеки, стабільна робота в багатозадачному середовищі.	Відсутність можливості створення власних типів даних і множинного спадкування, нестача інструментів автодоповнення методів у деяких середовищах розробки.

Для створення програмного додатку оцінювання ефективності працівників організації було вирішено обрати C# як мову розробки з кількох причин.

C# є однією з найпотужніших і найзручніших мов програмування для створення комплексних додатків, особливо коли йдеться про інтеграцію з базами даних і веб-системами. Його основні переваги в такому проєкті – це швидкість розробки, простота синтаксису, підтримка об'єктно-орієнтованого підходу та потужна інтеграція з платформою .NET. Завдяки автоматичному керуванню пам’яттю, вбудованій безпеці типів і широкій підтримці мережевих протоколів, C# дозволяє створювати стабільні, масштабовані та легко підтримувані рішення [13].

Особливо важливу роль у процесі розробки відіграє Visual Studio – інтегроване середовище розробки, яке надає все необхідне для повного циклу створення додатку. Visual Studio пропонує гнучкий редактор коду з підсвічуванням синтаксису, автоматичним доповненням коду (IntelliSense), вбудованими інструментами для роботи з базами даних (наприклад, SQL Server Management), зручним дизайнером інтерфейсів користувача (Windows Forms, WPF) і потужною системою налагодження.

Для задачі створення додатку з оцінювання ефективності працівників організації, що інтегрується з базою даних і веб-системою, поєднання C# і Visual Studio є ідеальним вибором. Середовище дозволяє легко підключатися до баз даних за допомогою ADO.NET, Entity Framework або Dapper, що спрощує як запити до БД, так і обробку даних. Для створення веб-частини можна використовувати ASP.NET або Blazor, що забезпечує плавну інтеграцію між клієнтською та серверною логікою.

Visual Studio також підтримує модульне тестування, безперервну інтеграцію (CI/CD) та має розширені інструменти для профілювання продуктивності додатків, що особливо важливо для оцінювання ефективності системи в реальному використанні [14-16]. Наявність шаблонів проєктів, генераторів коду та вбудованої підтримки сучасних стандартів розробки значно скорочує час розробки та зменшує кількість помилок.

3.2 Створення програмного модуля оцінювання ефективності працівників організації

Розглянемо детально створення коду для програмного модуля оцінювання ефективності працівників організації. У процесі розробки додатку для оцінювання ефективності працівників організації було використано сучасні підходи до побудови архітектури на основі C#, платформи .NET та ORM-бібліотеки Entity Framework Core. Основним класом доступу до бази даних є ApplicationContext, який успадковує DbContext і забезпечує налаштування сутностей через перевизначення методу OnModelCreating. У цьому методі реалізовано конфігурацію складних зв'язків "багато до багатьох" через проміжні таблиці (DepartmentParameters і ParametersGroup), а також початкове заповнення бази даних даними (HasData), включаючи створення стандартних ролей користувачів, станів співробітників та адміністративного користувача.

У класі DepartmentsRepository реалізовано логіку роботи з департаментами та співробітниками. Основні методи включають AllDepartments() для отримання списку всіх департаментів з пов'язаними сутностями, CountTop() для визначення кращих співробітників за певними параметрами на основі останніх оцінок, Delete() для видалення департаментів, Insert() і Update() для додавання та оновлення даних відповідно. Важливою частиною є методи GetLastEvaluations() і GetOldEvaluations(), які розділяють оцінки користувачів на актуальні та застарілі, що дозволяє правильно розраховувати прогрес працівників через метод GetUserProgress().

Методи GetUserLastEvaluationAvg() і GetUsersEvaluationAvg() виконують обчислення середніх оцінок користувача за останні і попередні періоди відповідно, що дає змогу визначити динаміку змін ефективності працівників. Логіка визначення найкращих співробітників (CountTop) базується на сумуванні балів за оціночними параметрами, враховуючи коефіцієнти важливості кожного параметра.

Окремо варто виділити метод WorkbookCreate(), який за допомогою бібліотеки ClosedXML генерує Excel-файл зі списком кращих співробітників, де виводяться їх імена, прізвища, результати оцінювання та список оцінюваних

параметрів. Це значно підвищує зручність роботи з даними для подальшого аналізу та звітності.

Перед запуском застосунку ініціалізуються основні сутності, що беруть участь у процесі оцінювання ефективності працівників. На рисунку 3.1 зображено загальну структурну схему, яка демонструє послідовність етапів запуску, формування контексту застосунку та взаємодію з ключовими сутностями. Всі ці компоненти об’єднуються в рамках системи оцінювання, що забезпечує логіку збирання, збереження та аналізу результатів діяльності працівників на основі заданих параметрів.

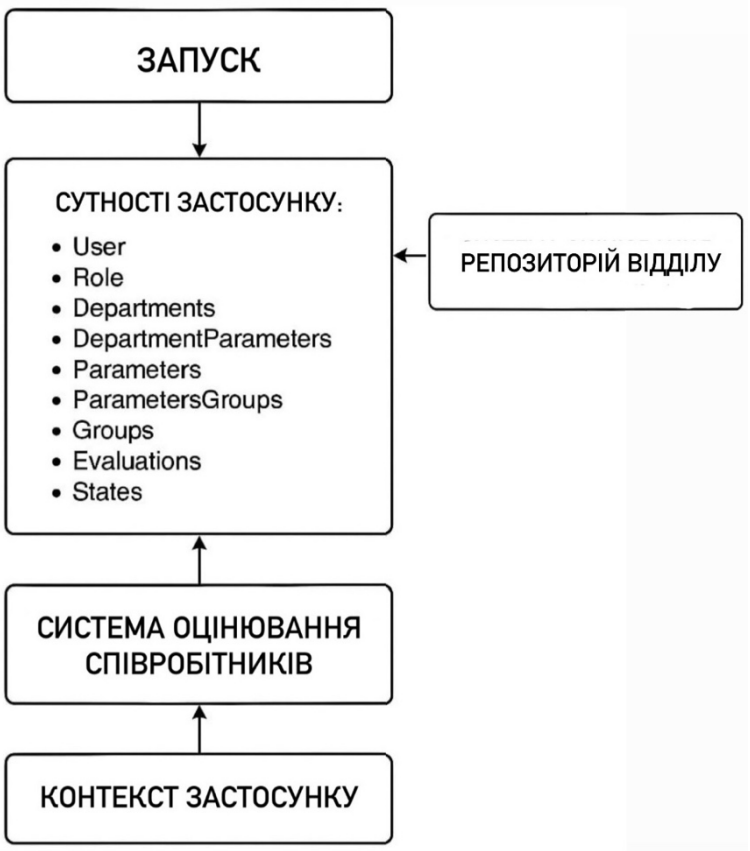

Рисунок 3.1 – Структурна діаграма розроблюваного модуля

Серед використовуваних сутностей у базі даних можна виділити:

- User (Користувач) – зберігає інформацію про співробітників.
- Role (Роль) – відповідає за рівень доступу.

- Departments (Департаменти) – об'єднують працівників у структурні підрозділи.
- Groups (Групи) – групують співробітників у межах департаменту.
- Parameters, DepartmentParameters, ParametersGroup – забезпечують гнучку систему оцінювання через різні критерії.
- Evaluations (Оцінки) – зберігають результати оцінювання працівників за параметрами.
- State (Статус працівника) v визначає, чи працівник активний, чи звільнений.

3.3 Написання коду програмного модуля оцінювання ефективності працівників організації

Під час розробки додатку було використано комплексний підхід до створення інформаційної системи управління ефективністю персоналу з інтеграцією сучасних технологій. Архітектура додатку передбачає використання принципів інверсії керування та впровадження залежностей (Dependency Injection), реалізованих у класі Startup. Через контейнер залежностей реєструються сервіси і репозиторії, які виконують чітко визначені функції: репозиторії забезпечують прямий доступ до бази даних, тоді як сервіси містять бізнес-логіку та забезпечують зв'язок між репозиторіями та іншими частинами додатку. Також реалізовано механізм аутентифікації за допомогою файлів cookie, що забезпечує безпеку доступу до інформації та функцій додатку.

```
public async Task<IActionResult> Login(LoginModel model)
{
if (ModelState.IsValid)
 {
var user = _userService.Authenticate(model.Email, model.Password);
if (user != null)
{
```

```
await Authenticate(user);
return RedirectToAction("Index", "Home");
}
}
return View(model);
}
```

Цей метод у AccountController реалізує автентифікацію користувача. Після успішного введення логіна й пароля, створюється сесія з відповідними правами доступу. Така реалізація гарантує безпечне керування доступом у системі.

У класі `Startup` у методі `ConfigureServices` реєструються всі необхідні сервіси, репозиторії, а також контекст бази даних. Крім цього, налаштовується схема автентифікації:

```
services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
.AddCookie(options => {
options.LoginPath = new PathString("/Account/Login");
 });
services.AddDbContext<ApplicationContext>(options =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
```

В модулі DepartmentsRepository реалізовано широкий спектр функціоналу для аналізу продуктивності працівників. Зокрема, метод отримання всіх департаментів використовує eager loading для ефективної вибірки даних із бази. Методи для обчислення рейтингу працівників (CountTop) дозволяють сортувати їх за результатами оцінок, що розраховуються як сума оцінок, зважених коефіцієнтами параметрів. Вибірка топових працівників здійснюється за вказаною кількістю з максимальними значеннями рейтингу. Крім того, передбачено методику визначення прогресу працівників на основі порівняння поточних та попередніх оцінок (DepEmployeesProgreses), яка дозволяє об'єктивно оцінювати динаміку їхньої продуктивності. Допоміжні методи, такі як визначення останніх і старих оцінок (GetLastEvaluations, GetOldEvaluations), а також обчислення середніх значень

(GetUserLastEvaluationAvg, GetUsersEvaluationAvg), дозволяють точно і глибоко аналізувати отримані дані.

Клас також включає реалізацію CRUD-операцій, які відповідають за додавання, оновлення та видалення інформації про відділи в системі. Важливим компонентом розробки є метод генерації Excel-звітів (WorkbookCreate), який забезпечує можливість експорту результатів оцінювання персоналу для подальшого аналітичного опрацювання.

Метод `WorkbookCreate()` реалізує логіку створення Excel-документа з використанням бібліотеки ClosedXML. У методі формується новий аркуш з назвою "Top_Emplyee", у якому по рядках записуються імена, прізвища, оцінки за параметрами та обчислені результати для кожного користувача з топ-списку.

```
var workbook = new XLWorkbook();
var worksheet = workbook.Worksheets.Add("Top_Emplyee");
int currentRow = 1;
worksheet.Cell(currentRow, 1).Value = "Name";
worksheet.Cell(currentRow, 2).Value = "SourName";
worksheet.Cell(currentRow, 3).Value = "Parameters";
worksheet.Cell(currentRow, 4).Value = "Result";

foreach (var user in UsersforExport)
{
currentRow++;
worksheet.Cell(currentRow, 1).Value = user.Name;
 worksheet.Cell(currentRow, 2).Value = user.SourName;
string Parameters = "";
 foreach (var paramGroup in ParametersGroupsForExport)
{
foreach (var eval in GetLastEvaluations())
{
if (eval.UserId == user.Id && eval.ParameterId == paramGroup.ParameterId)
```

```
Parameters += eval.Parameter.Name + ": " + eval.Mark + "\\n";
}
}
worksheet.Cell(currentRow, 3).Value = Parameters;
worksheet.Cell(currentRow, 4).Value = user.result;
}
```

Форматування файлу Excel реалізовано за допомогою бібліотеки ClosedXML. Метод WorkbookCreate створює таблицю з колонками Name, SourName, Parameters, Result і заповнює їх згідно з останніми оцінками:

```
var worksheet = workbook.Worksheets.Add("Top_Emplyee");
worksheet.Cell(currentRow, 1).Value = user.Name;
worksheet.Cell(currentRow, 2).Value = user.SourName;
worksheet.Cell(currentRow, 3).Value = Parameters;
worksheet.Cell(currentRow, 4).Value = user.result;
```

У структурі клієнтської частини розробленого програмного забезпечення чітко виділяються п’ять основних функціональних модулів: аутентифікації, параметрів, департаментів, працівників та оцінювання. Кожен із них реалізований засобами архітектури ASP.NET MVC у поєднанні з технологіями Entity Framework Core, що дозволяє забезпечити гнучку та надійну взаємодію з базою даних, а також чітко розмежувати зони відповідальності. Нижче наведено короткі фрагменти реалізації та обґрунтовано їхнє значення.

Проаналізуємо фрагмент коду по модулю аутентифікації:

```
services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
   .AddCookie(options => {
      options.LoginPath = new PathString("/Account/Login");
   });
```

Даний фрагмент коду відповідає за налаштування механізму автентифікації користувачів за допомогою cookie-файлів. Такий підхід дозволяє безпечно зберігати сеанс користувача та обмежувати доступ до функціоналу системи залежно від його ролі.

Проаналізуємо фрагмент коду по модулю параметрів:

```
public DbSet<Parameters> Parameters { get; set; }
public DbSet<DepartmentParameters> DepartmentParameters { get; set; }
public DbSet<ParametersGroup> ParametersGroups { get; set; }
```

У ApplicationContext цей блок визначає основні таблиці для збереження параметрів оцінки. Кожен параметр має назву, коефіцієнт важливості та градації від 1 до 5, які описуються в інтерфейсі (наприклад, "жахливо" до "чудово"). Завдяки сутностям DepartmentParameters і ParametersGroup система дозволяє адаптувати набір параметрів до різних підрозділів і груп працівників.

Проаналізуємо фрагмент коду по модулю департаментів:

```
public List<Departments> AllDepartments()
{
return _context.Departments
.Include(u => u.User)
.Include(u => u.Groups)
.Include(u => u.DepartmentParameters)
.ToList();
}
```

Даний модуль реалізує механізм завантаження всіх департаментів із пов'язаними користувачами, групами та параметрами. Через використання .Include() забезпечується "жадібне завантаження", що покращує ефективність доступу до даних і зменшує кількість SQL-запитів.

Окрім отримання департаментів, у DepartmentsRepository реалізовано метод CountTop, який обчислює інтегральний рейтинг кожного працівника шляхом множення оцінок на коефіцієнти параметрів. Потім обираються топові працівники:

```
user.result += evaluations.Parameter.Coefficient * evaluations.Mark;
```

Проаналізуємо фрагмент коду по модулю працівників:

```
public int GetUserProgress(User user)
{
if (GetUsersEvaluationAvg(user) < GetUserLastEvaluationAvg(user))
```

```
user.progress = 1;
else if (GetUsersEvaluationAvg(user) > GetUserLastEvaluationAvg(user))
user.progress = -1;
else
user.progress = 0;
return user.progress;
}
```

Даний модуль визначає прогрес працівника на основі порівняння середньої оцінки попередніх періодів із останньою. Це дає змогу не лише бачити оцінки в статичному розрізі, а й фіксувати динаміку – чи працівник покращує свої результати, стагнує або деградує.

Допоміжні методи GetUserLastEvaluationAvg та GetUsersEvaluationAvg використовуються для аналізу динаміки оцінок:

```
Avg = sum / count;
```

Проаналізуємо фрагмент коду по модулю оцінювання:

```
public List<User> CountTop(int DepId, int GroupId, int EmployeeCount)
{
foreach (User user in departments.User)
{
user.result = 0.0;
foreach (ParametersGroup parametersGroup in _context.ParametersGroups.Where(p
=> p.GroupId == GroupId))
{
foreach (Evaluations eval in GetLastEvaluations())
{
if (eval.UserId == user.Id && eval.ParameterId == parametersGroup.ParameterId)
user.result += eval.Parameter.Coefficient * eval.Mark;
}
}
}
```

```
return departments.User.OrderByDescending(u =>
u.result).Take(EmployeeCount).ToList();
```

Даний метод виконує головну функцію системи – обчислення інтегрального показника ефективності працівника. Розрахунок ведеться за принципом зваженої оцінки, де кожен параметр має свій коефіцієнт. Після цього працівники сортуються у рейтингу. Це дозволяє керівництву швидко визначити найбільш і найменш ефективних співробітників, стимулювати кращих і вчасно реагувати на відставання.

Усі п’ять модулів тісно інтегровані між собою та реалізовані відповідно до принципів модульності, масштабованості та розділення відповідальностей (SoC) [11-12]. Кожен із них не лише виконує свою окрему функцію, а й забезпечує узгоджену взаємодію з іншими частинами системи.

Контролери системи реалізовані відповідно до принципів MVC. Наприклад, DepartmentsController надає доступ до списку департаментів та рейтингу працівників:

```
public IActionResult TopEmployees(int depId, int groupId)
{
var top = _departmentsService.CountTop(depId, groupId, 5);
return View(top);
}
```

3.4 Висновки

У цьому розділі було обґрунтовано вибір мови програмування, технологій та інструментів, які стали основою для реалізації програмного забезпечення з оцінювання ефективності працівників організації. На підставі порівняльного аналізу сучасних мов програмування та відповідного інструментарію для розробки інформаційних систем було прийнято рішення про використання саме C# у поєднанні з середовищем Visual Studio 2022. Такий вибір обумовлений низкою переваг, зокрема – високим рівнем інтеграції інструментів розробки, багатим набором бібліотек, зручністю побудови веб-застосунків та сучасною підтримкою шаблонів архітектури MVC.

В основі побудови архітектури системи лежить використання фреймворку ASP.NET Core та бібліотеки Entity Framework Core як ORM-рішення для взаємодії з базою даних. Це дозволило реалізувати багаторівневу архітектуру з чітким розмежуванням відповідальностей між контролерами, сервісами та репозиторіями. Завдяки цьому система зберігає гнучкість, масштабованість і підтримуваність.

Було реалізовано основні модулі для управління користувачами, департаментами, параметрами оцінювання, оцінками та формування звітів. Архітектура застосунку побудована з урахуванням принципів інверсії керування (IoC) та впровадження залежностей (DI), що зменшило зв’язність між компонентами та спростило модульне тестування.

Особливу увагу приділено аутентифікації на основі cookie, що забезпечує доступ до функцій згідно з роллю користувача, а також реалізовано обробку помилок відповідно до середовища виконання для стабільної роботи та зручного тестування.

У межах розділу було проведено детальний аналіз ключових фрагментів програмного коду, що ілюструють реалізацію основної логіки системи — зокрема, розрахунку показників ефективності працівників із використанням вагових коефіцієнтів, формування топ-списків співробітників, визначення динаміки продуктивності, експорту результатів у формат Excel тощо.

Кожен модуль був розглянутий із позицій не лише технічної реалізації, а й його ролі у загальному функціонуванні додатку. Контролери, які опрацьовують запити, тісно взаємодіють із сервісами, де зосереджена бізнес-логіка, що в свою чергу використовує репозиторії для доступу до даних. Такий підхід відповідає принципам розділення відповідальностей (SoC) і забезпечує легкість модифікацій у майбутньому.

Отже, реалізована програмна частина повністю відповідає поставленим вимогам та забезпечує функціональність, необхідну для об’єктивного, масштабованого та зручного оцінювання ефективності працівників у рамках організаційної структури підприємства.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ПРАЦІВНИКІВ ОРГАНІЗАЦІЇ

4.1 Методи тестування програмного забезпечення

Юніт-тестування. Тестування окремих модулів або компонентів коду.

Інтеграційне тестування. Перевірка взаємодії між різними модулями чи компонентами.

Системне тестування. Тестування всієї системи як єдиного цілого.

Функціональне тестування. Перевірка, чи відповідає система вимогам користувача або бізнесу.

Тестування регресії. Перевірка системи після внесення змін для забезпечення того, що новий код не вплинув негативно на існуючий функціонал.

Тестування надійності. Визначення здатності системи працювати без збоїв протягом певного часу.

Тестування безпеки. Оцінка системи на вразливості, загрози та ризики безпеки.

Альфа- та бета-тестування. Перший етап тестування, що виконується розробниками або спеціалізованою групою тестувальників в рамках організації розробника. Тестування системи обмеженою групою користувачів в реальних умовах перед остаточним випуском продукту.

Для тестування розроблених модуль проведемо лише функціональне тестування. Оскільки цей тип дозволить впевнитися у коректності роботи розроблених модулів.

4.2 Функціональне тестування системи

Протестуємо розроблений додаток на працездатність та коректність роботи. Для цього необхідно зайти з боку адміністратора на ресурс за допомогою реалізованої форми автентифікації, яка доступна після запуску додатку. Зображено інтерфейс форми входу до системи (рис.4.1).

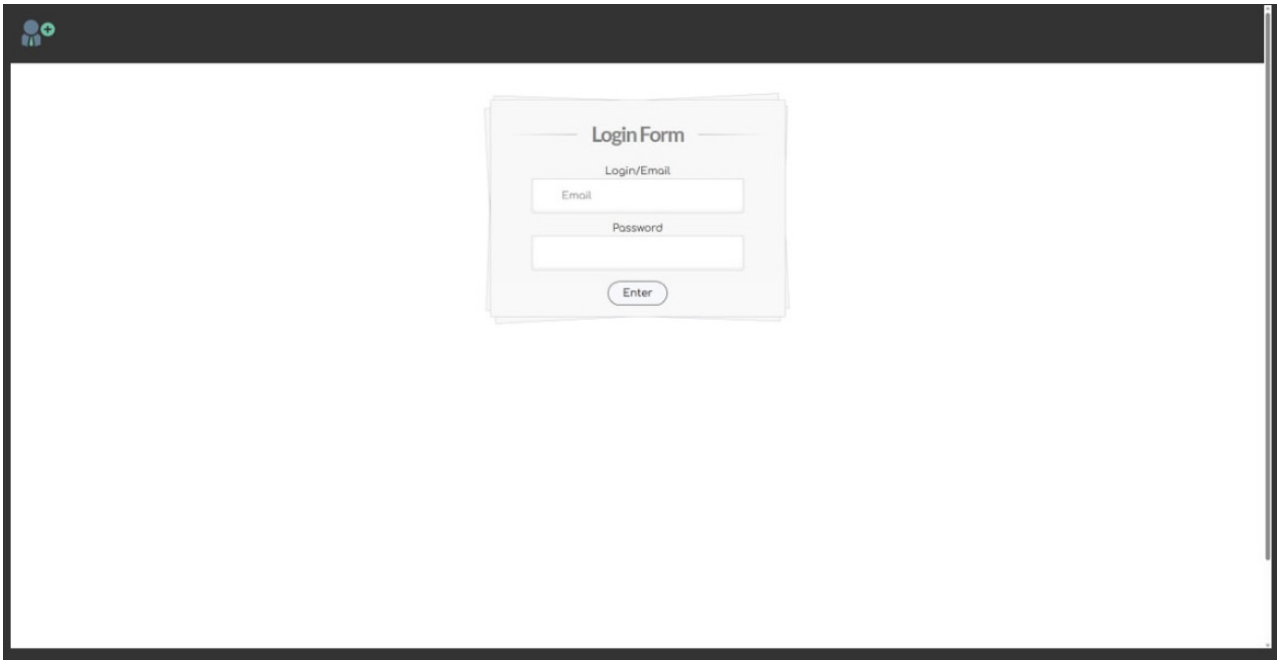

Рисунок 4.1 – Інтерфейс форми входу до системи програмного додатку

Користувач повинен ввести електронну пошту (логін) та пароль у відповідні поля, після чого натиснути кнопку **Enter**. У разі правильного введення облікових даних система ініціалізує сесію користувача та перенаправляє його до головної сторінки.

Якщо логін або пароль введено некоректно, система не дозволяє вхід і виводить відповідне повідомлення про помилку. (рис. 4.2).

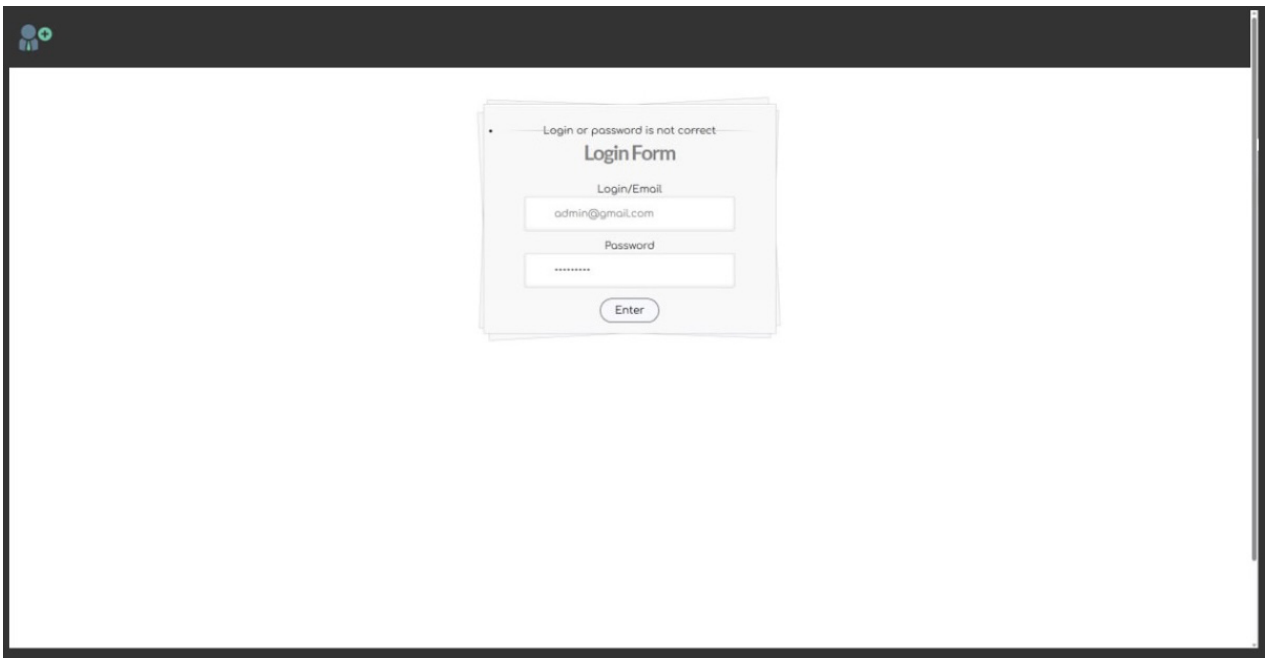

Рисунок 4.2 – Інтерфейс форми входу до системи програмного додатку з невірними даними

Після коректного входу можна побачити основні параметри оцінювання ефективності працівників організації (рис.4.3).

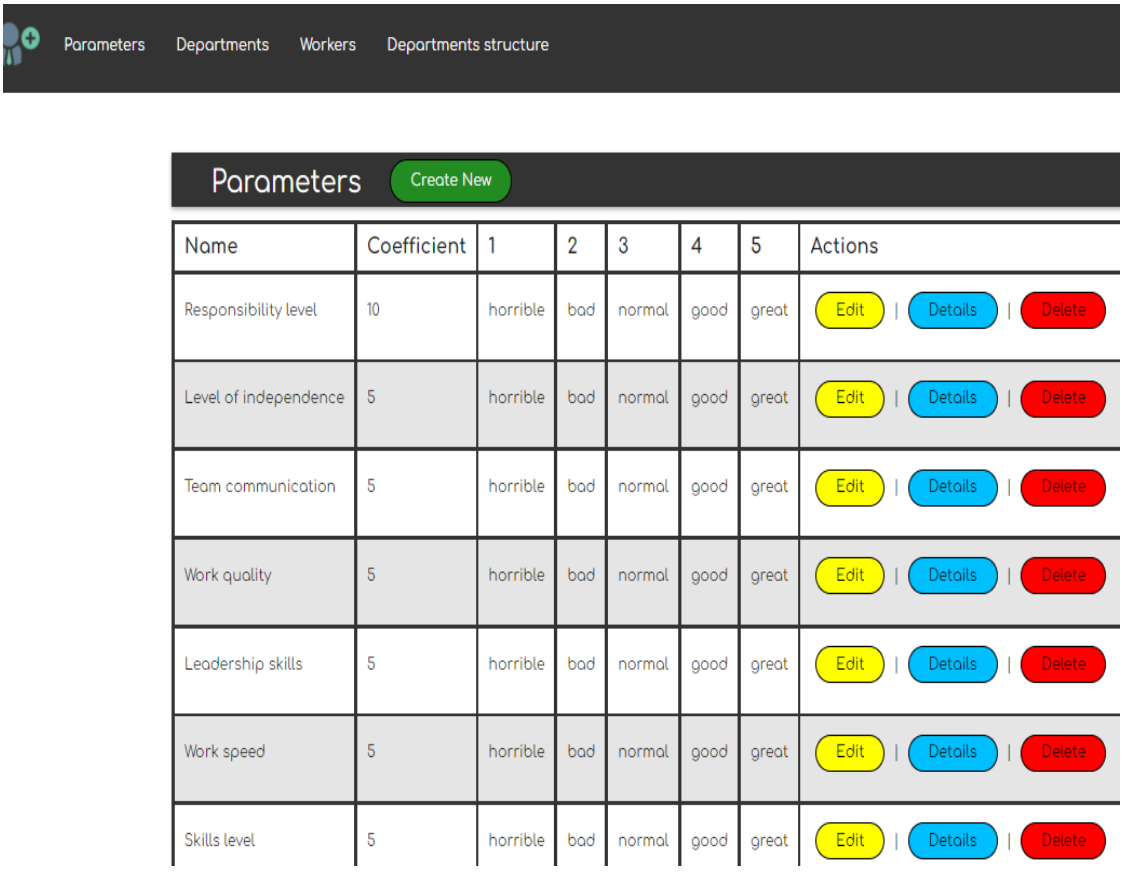

Рисунок 4.3 – Основні параметри оцінювання ефективності працівників

На наступному скріншоті відображена панель редагування параметра для оцінювання ефективності працівників (рис.4.4). Інтерфейс має назву "Edit parameter - EPA" і містить такі елементи:

1. Основні поля:
   - Name (Назва): "Responsibility level" (Рівень відповідальності)
   - Coefficient (Коефіцієнт): "10"
2. Шкала оцінювання з 5 рівнів:
   - Description for mark "1": "horrible" (жахливо)
   - Description for mark "2": "bad" (погано)
   - Description for mark "3": "normal" (нормально)
   - Description for mark "4": "good" (добре)
   - Description for mark "5": "great" (чудово)
3. Функціональні кнопки:
   - "Save" (Зберегти) - зелена кнопка для збереження змін
   - Back" (Назад) - для повернення на попередню сторінку.

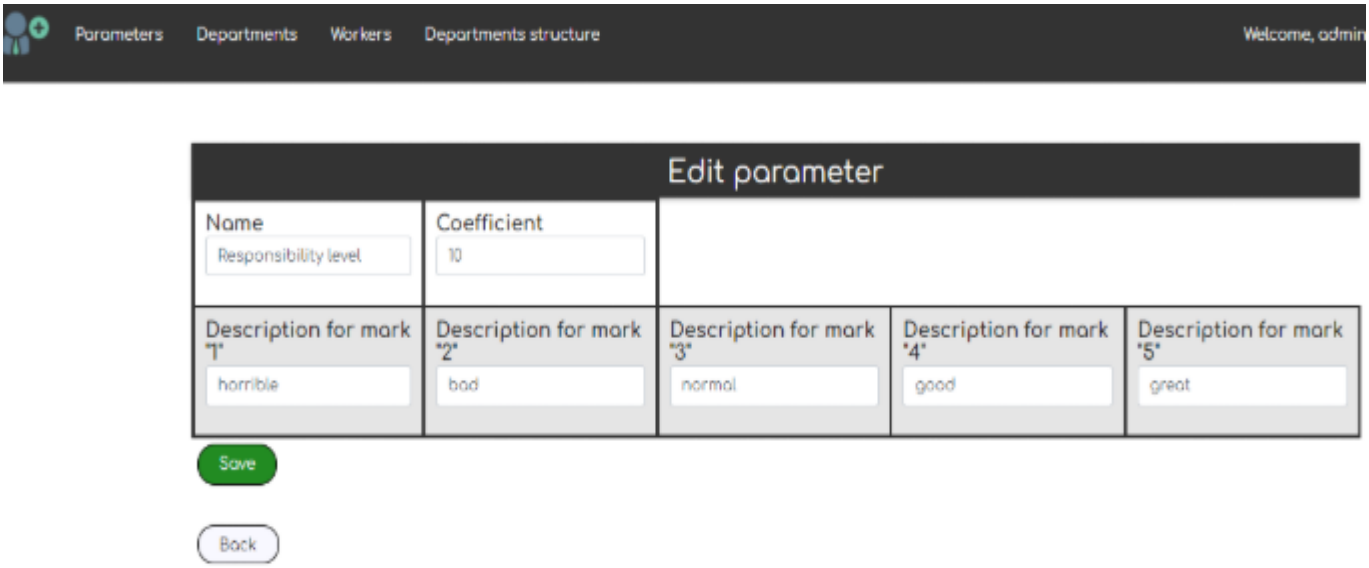

Рисунок 4.4 – Редагування параметрів оцінювання ефективності працівників організації

На наступному скріншоті представлена сторінка департаменту в системі оцінювання ефективності. На сторінці є кнопки "Edit" (жовтого кольору) та червона кнопка (ймовірно для видалення), а також кнопка "Back" для повернення на попередню сторінку. У верхній частині розташовані вкладки "Workers" (Працівники), "Parameters" (Параметри) та "Groups" (Групи), причому активною є вкладка "Workers". Також є кнопки "Progressing workers" (зелена) та "Lagging employees" (червона) для фільтрації працівників. Основну частину сторінки займає таблиця працівників з колонками: "Name" (Ім'я), "Sourname" (Прізвище), "Perfomances" (Показники) та "Progress" (Прогрес). У таблиці відображено список співробітників та їх статуси. Ця сторінка дозволяє керівництву відстежувати ефективність працівників відділу та їхній прогрес (рис.4.5).

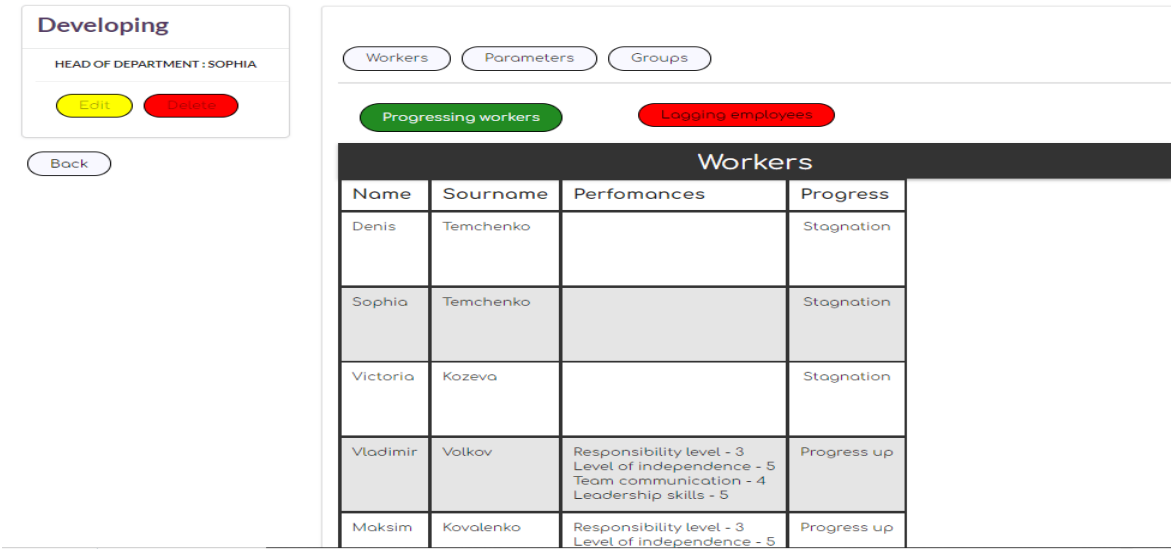

Рисунок 4.5 – Результати проведеного оцінювання працівників

Тестування основних функцій розробленого додатку повністю підтвердило коректність його роботи.

Розроблений додаток показує високий рівень ефективності у порівнянні з провідними аналогами. Його перевага забезпечується завдяки інтегрованому підходу до оцінювання продуктивності персоналу, зручному інтерфейсу та розширеній функціональності, яка включає налаштування параметрів з деталізованими описами, гнучку шкалу оцінювання, а також можливості для

детального аналізу та експорт результатів оцінки в Excel.

В таблиці 4.1 наведемо порівняння характеристик ефективності розробленого додатку та аналогів шляхом внесення інтегральних оцінок результатів опитування десяти користувачів щодо чотирьох програм-аналогів та розробленого додатку оцінювання ефективності працівників організації, а на рисунку 4.5 – відповідний порівняльний графік.

Таблиця 4.1 – Порівняльний аналіз ефективності розробленого додатку та його аналогів

Назва системи	Загальна ефективність (%)	Інтерфейс користувача (%)	Аналітичні можливості (%)	Гнучкість налаштувань (%)
Розроблений додаток	92	95	94	93
SAP SuccessFactors	88	93	90	85
BambooHR	85	88	83	90
Zoho People	80	85	80	83
Lattice	82	90	84	86

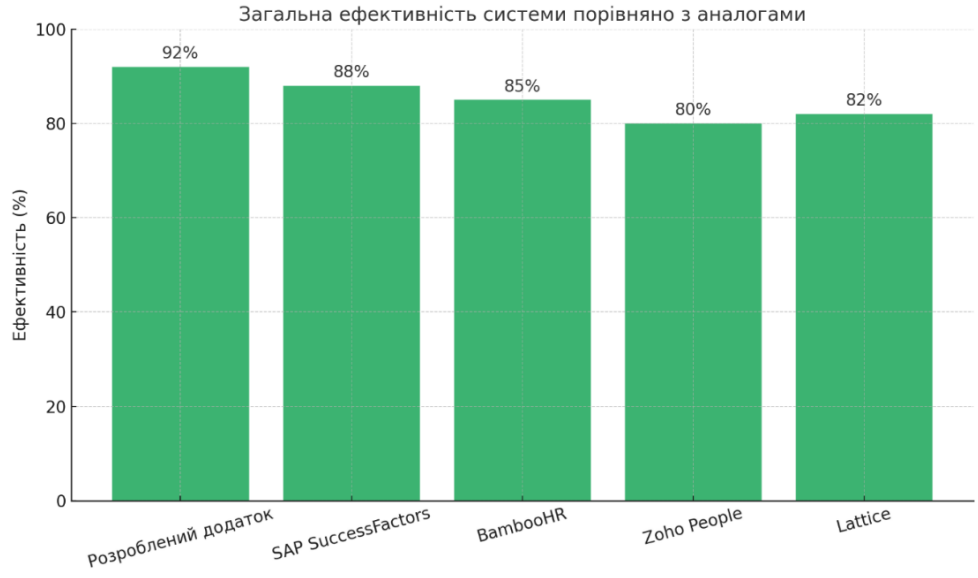

Рисунок 4.6 – Графік порівняння ефективності розробленого додатку з аналогами

Для перевірки коректності розрахунків ефективності працівників у розробленій системі було змодельовано ситуацію з десятьма умовними співробітниками, кожен з яких отримав оцінки за п’ятьма параметрами. Кожен параметр має власний коефіцієнт важливості, що визначає його вплив на загальний результат. Підсумкова ефективність обчислюється як зважена сума оцінок відповідно до заданих коефіцієнтів.

Нижче наведено таблицю, яка була автоматично згенерована за результатами обробки введених даних (рис 4.7).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Ім’я**	**Прізвище**	**Група**	**Коефіцієнт**	**Оцінка 1**	**Коефіцієнт**	**Оцінка 2**	**Коефіцієнт**	**Оцінка 3**	**Коефіцієнт**	**Оцінка 4**	**Коефіцієнт**	**Оцінка 5**	**Результат**
2	Denis	Temchenko	Group1	0,8	5	0,6	3	0,7	4	0,5	2	0,9	3	12,3
3	Sophia	Temchenko	Group1	0,8	4	0,6	4	0,7	5	0,5	3	0,9	4	14,2
4	Viktoria	Сидоренко	Group1	0,8	3	0,6	5	0,7	4	0,5	4	0,9	5	14,7
5	Volodimir	Volkov	Group1	0,8	5	0,6	3	0,7	5	0,5	5	0,9	3	14,5
6	Maksim	Kovalenko	Group1	0,8	4	0,6	4	0,7	4	0,5	2	0,9	4	13
7	Tetiana	Boyko	Group1	0,8	3	0,6	5	0,7	5	0,5	3	0,9	5	14,9
8	Oleg	Melnik	Group1	0,8	5	0,6	3	0,7	4	0,5	4	0,9	3	13,3
9	Yulia	Gavrilyuk	Group1	0,8	4	0,6	4	0,7	5	0,5	5	0,9	4	15,2
10	Bogdan	Shevchenko	Group1	0,8	3	0,6	5	0,7	4	0,5	2	0,9	5	13,7
11	Irina	Kozak	Group1	0,8	5	0,6	3	0,7	5	0,5	3	0,9	3	13,5

Рисунок 4.7 – Таблиця результатів оцінювання ефективності працівників організації

4.3 Висновки

У четвертому розділі було проведено тестування розробленого програмного забезпечення з метою перевірки його працездатності, коректності виконання основного функціоналу та відповідності поставленим вимогам. У процесі тестування перевірялись ключові сценарії взаємодії користувача із системою, зокрема авторизація, перегляд та оцінювання працівників, генерація звітів і робота з параметрами ефективності.

Результати тестування засвідчили, що всі основні модулі — автентифікації, департаментів, параметрів, працівників та оцінювання — функціонують стабільно та без критичних помилок. Зокрема, була успішно перевірена авторизація з обмеженням доступу до захищених розділів, коректне виведення результатів оцінювання працівників, а також правильність сортування за ефективністю. Також

протестовано експорт даних у формат Excel, що підтвердив відповідність структури звіту очікуваному формату.

Тестування графічного інтерфейсу виявило, що користувацький досвід є інтуїтивно зрозумілим, а навігація по додатку не викликає труднощів. Проведено вхід у систему за допомогою тестового облікового запису, після чого були успішно доступні усі дозволені функції згідно з роллю користувача.

Отже, результати тестування підтвердили працездатність програмного продукту, правильність реалованої логіки та готовність системи до використання у реальних умовах для оцінювання ефективності персоналу організації.

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було досягнуто основної мети – розроблено й протестовано програмний модуль, який забезпечує об'єктивне та ефективне оцінювання працівників організації, що може бути адаптовано для потреб конкретних компаній і подальшого вдосконалення.

У результаті дослідження було досягнено таких основних результатів:

- проаналізовано сучасні підходи, методики та критерії оцінювання ефективності працівників, що дозволило виокремити ключові параметри оцінювання, які лягли в основу розробленої системи;
- досліджено існуючі HR-програми (зокрема SAP SuccessFactors, BambooHR, Zoho People), виявлено їхні переваги та недоліки, що дозволило сформувати конкурентні функціональні характеристики майбутнього додатку;
- сформульовано вимоги до функціоналу програмного забезпечення, визначено основні ролі користувачів, механізм оцінювання, вимоги до інтерфейсу, безпеки та масштабованості;
- розроблено архітектуру додатку, що включає багаторівневу структуру з фронтендом, API-шлюзом, backend-логікою, сервісним шаром, репозиторіями та базою даних. Також спроєктовано інтерфейс з урахуванням зручності використання.
- реалізовано програмний продукт мовою C# з використанням платформи .NET та Entity Framework Core. Реалізовані функції оцінювання, управління працівниками, генерації звітів та аналізу прогресу;
- проведено тестування додатку, яке підтвердило коректну роботу всіх основних функцій, зручність інтерфейсу та стабільність системи.
- оцінено ефективність застосування системи на прикладі модельованої організаційної структури, підтверджено її потенційну користь для автоматизації процесів оцінювання персоналу.

Мету розширення функціональних можливостей додатку досягнуто за рахунок опції створення власноруч відповідних параметрів оцінювання адміністратором,

внаслідок чого можна відбирати списки кращих працівників за певними критеріями і формувати робочі групи за призначенням.

У межах даної роботи було розроблено програмний додаток, результат якої повністю відповідає меті бакалаврської роботи і охоплює всі необхідні інструменти для оцінювання ефективності працівників організації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Акаткін Ю. М. Технології штучного інтелекту у сфері управління персоналом: тенденції впровадження / Ю. М. Акаткін // Менеджмент і економіка. – 2021. – № 6(23). – С. 15–24.

2. Дубина М. В., Лобко О. М. Особливості цифрової трансформації: від автоматизації до цифрових платформ у HR / М. В. Дубина, О. М. Лобко // Проблеми і перспективи економіки та управління. – 2024. – № 2(38). – С. 7–25.

3. Вишневська О. П. Аналітика даних як інструмент підвищення продуктивності HR-системи / О. П. Вишневська // Журнал сучасної економіки. – 2021. – № 9. – С. 56–68.

4. Симоненко Т. В. Цифровізація управлінських процесів: досвід ЄС і перспективи для України / Т. В. Симоненко // Економіка та держава. – 2023. – № 2. – С. 43–50.

5. Савчук В. Л. Стратегічне управління персоналом в умовах цифрової трансформації / В. Л. Савчук // Бізнес і технології. – 2020. – № 12. – С. 77–85.

6. Медведєва Л. П. Цифрові платформи як інструмент розвитку кадрового потенціалу / Л. П. Медведєва // Журнал управління персоналом. – 2023. – № 1. – С. 35–44.

7. Горбунова М. С. Впровадження цифрової культури як умова успішної трансформації HR-системи / М. С. Горбунова // HR-технології. – 2022. – № 11. – С. 60–72.

8. Чорна С. І. Адаптація персоналу до цифрових змін: практика впровадження / С. І. Чорна // Управлінські інновації. – 2021. – № 9. – С. 29–36.

9. Шевченко О. В. Дослідження ефективності цифрових рішень у системі управління персоналом / О. В. Шевченко // Економічний аналіз. – 2023. – № 8. – С. 112–120.

10. Yankovoi R., Stadniichuk R., Zhosan H., Garafonova O., Biriukov I. Innovative transformation of a financial institution in the context of digitalisation and its impact on social conflict management // Financial and Credit Activity: Problems of Theory and Practice. – 2024. – № 2(55). – Pp. 75–88. DOI: 10.55643/fcaptp.2.55.2024.4386.

11. Балуєва О. В. Методи оцінки ефективності персоналу: еволюція під впливом розвитку технологій / О. В. Балуєва, Г. В. Снопенко // Інвестиції: практика та досвід. – 2021. – № 21. – С. 30–36.

12. Гуцуляк Н. П. Застосування сучасних технологій оцінювання та діагностики персоналу / Н. П. Гуцуляк // Збірник наукових праць Таврійського державного агротехнологічного університету імені Дмитра Моторного (економічні науки). – 2019. – Вип. 2 (40). – С. 29–38.

13. Цимбалюк С. О. Оцінювання персоналу: навч. посіб. / С. О. Цимбалюк, О. М. Білик. – Київ: КНЕУ, 2021. – 328 с.

14. Варіс І. Цифровізація бізнес-процесів менеджменту персоналу: можливості HRM-систем / І. Варіс, О. Кравчук, Є. Паращук // Галицький економічний вісник. – 2022. – № 1 (74). – С. 90–102. – DOI: https://doi.org/10.33108/galicianvisnyk_tntu2022.01.

15. Федорова Ю. Інноваційні інформаційні технології в підготовці та управлінні персоналом / Ю. Федорова, Г. Єльникова // Адаптивне управління: теорія і практика. Серія «Економіка». – 2021. – Вип. 11 (22). – DOI: https://doi.org/10.33296/2707-0654-11(22)-11.

16. Малієва Ю. А. Інформаційне та програмне забезпечення менеджера з персоналу ІТ-компанії / Ю. А. Малієва, О. Ю. Персіянова, В. В. Косенко // Сучасний стан наукових досліджень та технологій у промисловості. – 2018. – № 1 (3). – С. 22–32.

17. Шевченко В. М. Інформаційні технології в управлінні персоналом: теорія і практика. – К.: Центр учбової літератури, 2021. – 208 с.

18. Ніколаєнко С. М. Управління людськими ресурсами в умовах цифрової трансформації. – Львів: Вид-во ЛНУ, 2022. – 156 с.

19. Гнатюк О. Г. Системи підтримки прийняття управлінських рішень. – Вінниця: ВНТУ, 2020. – 164 с.

20. Яровий А. А., Сілагін О. В., Богач І. В. Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - «Комп`ютерні науки» – Вінниця: ВНТУ, 2023. – 53 с.

ДОДАТКИ

ДОДАТОК А (обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмного додатку оцінювання ефективності працівників організації

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,95 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Белзецький Р.С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Волох В.С.
(підпис) (прізвище, ініціали)

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧА

1. Встановлення через архів

1. Завантажте Visual Studio з офіційного сайту: https://visualstudio.microsoft.com/ru/downloads

2. Завантажте ZIP-архів проєкту (натисніть "Download ZIP").

3. Перейдіть до папки "Завантаження", знайдіть архів і розпакуйте його у будь-яке зручне місце на комп’ютері.

4. Відкрийте файл Workers_performance_analisys.sln.

5. Збережіть відкритий проєкт комбінацією клавіш Ctrl + Shift + S.

2. Вхід у систему

1. Запустіть застосунок.

2. Авторизуйтесь за допомогою свого логіна/електронної пошти та пароля.

3. Залежно від вашої ролі в системі (адміністратор, керівник відділу, тімлід, працівник), вам буде доступний відповідний інтерфейс та функціональність.

3. Роль: Адміністратор

3.1 Панель керування параметрами

- Перейдіть у вкладку "Parameters" в панелі навігації.
- У вікні буде показано таблицю з існуючими параметрами оцінювання.
- Щоб видалити параметр – натисніть кнопку "Delete" навпроти потрібного параметра та підтвердіть дію.
- Щоб додати новий параметр – натисніть "Create new", заповніть поля у формі, натисніть "Create".
- Щоб редагувати параметр – натисніть "Edit" або "Details", внесіть зміни у поля та збережіть.

3.2 Керування працівниками

- Відкрийте вкладку "Workers".
- У таблиці ви побачите список працівників компанії.
- Щоб видалити працівника (рекомендується змінити статус на "fired", а не видаляти) – натисніть "Delete" та підтвердіть.
- Щоб додати нового працівника – натисніть "Create new", заповніть дані і натисніть "Create".
- Щоб редагувати інформацію – натисніть "Edit", внесіть зміни та натисніть "Save".

3.3 Структура відділів

- Перейдіть у вкладку "Departments structure".
- Відобразиться ієрархія: керівник відділу → тімлід → працівники.

3.4 Керування відділами

- У вкладці "Departments" можна:
  - створювати нові відділи ("Create"),
  - редагувати назви або видаляти відділи (кнопки "Edit" / "Delete" у картці відділу).
- У розділі "Workers" в межах відділу:
  - переглядати імена, прізвища, оцінки, динаміку прогресу,
  - фільтрувати та відстежувати зміни оцінок.
- У розділі "Parameters":
  - переглядати важливі для цього відділу параметри,
  - додавати або видаляти параметри.
- У розділі "Groups":
  - керувати групами працівників,
  - експортувати дані у .xls, створювати рейтинги кращих працівників.

3.5 Вихід із системи

- Натисніть "Logout" (у верхньому правому куті).
- Вас буде перенаправлено на сторінку авторизації.

4. Роль: Тімлід

4.1 Оцінювання працівників

- Після входу відкриється таблиця з оцінками працівників.
- Можна створювати або видаляти оцінки за аналогією з адміністративним функціоналом

4.2 Структура відділів

- Доступна для перегляду у вкладці "Departments structure".

4.3 Вихід із системи

- Аналогічно натисніть "Logout" для виходу.

5. Роль: Керівник відділу

5.1 Оцінювання працівників і тімлідів

- Доступні функції аналогічні тімліду, але також є можливість оцінювати тімлідів.

5.2 Структура відділу

- Перегляд ієрархії аналогічно до інших ролей.

5.3 Вихід із системи

- Натисніть "Logout", щоб завершити сесію.

6. Роль: Працівник

- Працівникам надається обмежений доступ для перегляду власних оцінок і структури відділу.
- Взаємодія з іншими модулями обмежена, тільки перегляд інформації.

ДОДАТОК В
ЛІСТИНГ ПРОГРАМИ

```
using AngleSharp.Dom;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Perfomans.Models
{
   public class ApplicationContext : DbContext
   {
      public DbSet<User> User { get; set; }
      public DbSet<Role> Roles { get; set; }
      public DbSet<Departments> Departments { get; set; }
      public DbSet<DepartmentParameters> DepartmentParameters { get; set; }
      public DbSet<Parameters> Parameters { get; set; }
      public DbSet<ParametersGroup> ParametersGroups { get; set; }
      public DbSet<Groups> Groups { get; set; }
      public DbSet<Evaluations> Evaluations { get; set; }
      public DbSet<State> States { get; set; }


      protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
         modelBuilder.Entity<DepartmentParameters>().HasKey(pi => new { pi.DepartmentId, pi.ParameterId });
         modelBuilder.Entity<DepartmentParameters>().
            HasOne(pi => pi.Department).WithMany(pi => pi.DepartmentParameters).HasForeignKey(p => p.DepartmentId);
         modelBuilder.Entity<DepartmentParameters>().
            HasOne(pi => pi.parameter).WithMany(pi => pi.Departments).HasForeignKey(p => p.ParameterId);

         modelBuilder.Entity<ParametersGroup>().HasKey(pi => new { pi.GroupId, pi.ParameterId });
         modelBuilder.Entity<ParametersGroup>().
```

```
    HasOne(pi => pi.Parameters).WithMany(pi => pi.ParametersGroups).HasForeignKey(p => p.ParameterId);
modelBuilder.Entity<ParametersGroup>().
    HasOne(pi => pi.Groups).WithMany(pi => pi.ParametersGroups).HasForeignKey(p => p.GroupId);



string adminRoleName = "Admin";
string userRoleName = "User";
string TLRoleName = "Teamlead";
string HeadRoleName = "Head";

string adminEmail = "admin@gmail.com";
string adminPassword = "123456";
string adminName = "Vlad";
string adminSourName = "Volokh";

string devdep = "Developing";
string GroupName = "Group1";

string DefState = "Default";
string FireState = "Fired";

Role adminRole = new Role { Id = 1, Name = adminRoleName };
Role userRole = new Role { Id = 2, Name = userRoleName };
Role TLRole = new Role { Id = 3, Name = TLRoleName };
Role HeadRole = new Role { Id = 4, Name = HeadRoleName };

Departments departments = new Departments { Id = 1, Name = devdep };
Groups groups = new Groups { id = 1, Name = GroupName, DepartmentId = departments.Id };

State state = new State { Id = 1, Name = DefState };
State state1 = new State { Id = 2, Name = FireState };



modelBuilder.Entity<Departments>().HasData(new Departments[] { departments });
modelBuilder.Entity<Groups>().HasData(new Groups[] { groups });
```

```
            modelBuilder.Entity<Role>().HasData(new Role[] { adminRole, userRole, TLRole, HeadRole });
            modelBuilder.Entity<State>().HasData(new State[] { state, state1 });

            User adminUser = new User {
                Id = 1,
                Name = adminName,
                SourName = adminSourName,
                Email = adminEmail,
                Password = adminPassword,
                RoleId = adminRole.Id,
                DepartmentId = departments.Id,
                StateId = state.Id,
                SupervisorId = 1
            };
            modelBuilder.Entity<User>().HasData(new User[] { adminUser });

            base.OnModelCreating(modelBuilder);
        }
        public ApplicationContext(DbContextOptions<ApplicationContext> options)
            : base(options)
        {
            Database.EnsureCreated();
        }
    }
}
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Perfomans.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Perfomans.Repository;
using Perfomans.Service;

namespace Perfomans
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      public void ConfigureServices(IServiceCollection services)
      {
         services.AddTransient<IUserRepository, UserRepository>();
         services.AddTransient<IUserService, UserService>();

         services.AddTransient<IParametersRepository, ParametersRepository>();
         services.AddTransient<IParametersService, Parametersservice>();

         services.AddTransient<IGroupsRepository, GroupsRepository>();
         services.AddTransient<IGroupService, GroupService>();

         services.AddTransient<IEvalRepository, EvalRepository>();
         services.AddTransient<IEvalService, EvalService>();

         services.AddTransient<IDepParamRepository, DepParamRepository>();
         services.AddTransient<IDepParamService, DepParamService>();

         services.AddTransient<IDepartmentsRepository, DepartmentsRepository>();
         services.AddTransient<IDepartmentsService, DepartmentsService>();


         string connection = Configuration.GetConnectionString("DefaultConnection");
         services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(connection));
```

```
        services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
            .AddCookie(options =>
            {
                options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
            });
        services.AddControllersWithViews();
    }

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseAuthentication();
        app.UseAuthorization();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
  }
}
```

```
using ClosedXML.Excel;
using Microsoft.EntityFrameworkCore;
using Perfomans.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Perfomans.Repository
{
    public class DepartmentsRepository : IDepartmentsRepository
    {
        public static List<User> UsersforExport = new List<User>();
        public static List<ParametersGroup> ParametersGroupsForExport = new List<ParametersGroup>();
        public static int GroupIdForExport;
        private readonly ApplicationContext _context;

        public DepartmentsRepository(ApplicationContext context)
        {
            _context = context;
        }
        public List<Departments> AllDepartments()
        {
            var deparments = _context.Departments.Include(u => u.User).Include(u => u.Groups).Include(u =>
u.DepartmentParameters);
            return deparments.ToList();
        }

        public List<User> CountTop(int DepId, int GroupId, int EmployeeCount)
        {
            Departments departments = GetById(DepId);
            foreach (User user in departments.User)
            {
                user.result = 0.0;
                foreach (ParametersGroup parametersGroup in _context.ParametersGroups.Where(p => p.GroupId ==
GroupId).ToList())
```

```
            {
                foreach (Evaluations evaluations in GetLastEvaluations())
                {
                    if ((parametersGroup.GroupId == GroupId) & (parametersGroup.ParameterId == evaluations.ParameterId) &
(evaluations.UserId == user.Id))
                    {
                        user.result += (evaluations.Parameter.Coefficient * evaluations.Mark);
                    }
                }

            }
        }
        var topsort = from user in departments.User orderby user.result descending select user;
        List<User> CountTop = new List<User>();

        foreach (User u in topsort)
        {
            CountTop.Add(u);
            if (CountTop.Count == EmployeeCount) { break; }
        }
        GroupIdForExport = GroupId;
        ParametersGroupsForExport = _context.ParametersGroups.Where(p => p.GroupId == GroupId).ToList();
        UsersforExport = CountTop;

        return CountTop;
    }

    public void Delete(int? id)
    {
        Departments departments = _context.Departments.Find(id);
        _context.Departments.Remove(departments);
        _context.SaveChanges();
    }

    public Departments DepEmployeesProgreses(int? Id)
    {
```

```
        Departments departments = GetById(Id);
        foreach (User u in departments.User)
        {
            u.progress = GetUserProgress(u);
        }
        return departments;
    }

    public Departments GetById(int? id)
    {
        return _context.Departments.Include(d => d.DepartmentParameters).Include(d => d.Groups).Include(d => d.User).FirstOrDefault(d => d.Id == id);
    }

    public List<Evaluations> GetLastEvaluations()
    {
        List<Evaluations> lastevaluations = new List<Evaluations>();
        foreach (Evaluations evaluations in _context.Evaluations.ToList())
        {
            lastevaluations.Add(evaluations);
        }
        foreach (Evaluations laste in lastevaluations.ToList())
        {
            foreach (Evaluations e in _context.Evaluations.ToList())
            {
                if ((e.UserId == laste.UserId) & (e.ParameterId == laste.ParameterId) & (e.Id < laste.Id))
                {
                    lastevaluations.Remove(e);
                }
            }
        }
        return (lastevaluations);
    }

    public List<Evaluations> GetOldEvaluations()
    {
```

```
    List<Evaluations> oldevaluations = new List<Evaluations>();
    foreach (Evaluations evaluations in _context.Evaluations.ToList())
    {
        oldevaluations.Add(evaluations);
    }
    foreach (Evaluations e in GetLastEvaluations())
    {
        oldevaluations.Remove(e);
    }
    return (oldevaluations);
}

public double GetUserLastEvaluationAvg(User user)
{
    double Avg = 0.0;
    double count = 0.0;
    double sum = 0.0;
    foreach (Evaluations e in GetLastEvaluations())
    {
        if (e.UserId == user.Id)
        {
            sum += e.Mark;
            count += 1;
        }
    }
    Avg = sum / count;
    return Avg;
}

public int GetUserProgress(User user)
{
    user.progress = 0;

    if (GetUsersEvaluationAvg(user) < GetUserLastEvaluationAvg(user))
    {
        user.progress += 1;
```

```
    }
    else if (GetUsersEvaluationAvg(user) > GetUserLastEvaluationAvg(user))
    {
        user.progress -= 1;
    }
    else
    {
        user.progress -= 0;
    }

    return user.progress;
}

public double GetUsersEvaluationAvg(User user)
{
    double Avg = 0.0;
    double count = 0.0;
    double sum = 0.0;
    foreach (Evaluations e in GetOldEvaluations())
    {
        if (e.UserId == user.Id)
        {
            sum += e.Mark;
            count += 1;
        }
    }
    Avg = sum / count;
    return Avg;
}

public void Insert(Departments departments)
{
    _context.Add(departments);
    _context.SaveChanges();
}
```

```
public void Update(Departments departments)
{
    _context.Departments.Update(departments);
    _context.SaveChanges();
}

public void WorkbookCreate(XLWorkbook workbook)
{
    var worksheet = workbook.Worksheets.Add("Top_Emplyee");
    var currentRow = 1;
    worksheet.Cell(currentRow, 1).Value = "Name";
    worksheet.Cell(currentRow, 2).Value = "SourName";
    worksheet.Cell(currentRow, 3).Value = "Parameters";
    worksheet.Cell(currentRow, 4).Value = "Result";
    foreach (User user in UsersforExport)
    {
        currentRow++;
        worksheet.Cell(currentRow, 1).Value = user.Name;
        worksheet.Cell(currentRow, 2).Value = user.SourName;
        string Parameters = " ";

        foreach (ParametersGroup parametersGroup in ParametersGroupsForExport)
        {
            foreach (Evaluations evaluations in GetLastEvaluations())
            {
                if ((parametersGroup.GroupId == GroupIdForExport) & (parametersGroup.ParameterId ==
evaluations.ParameterId) & (evaluations.UserId == user.Id))
                {
                    Parameters = Parameters + evaluations.Parameter.Name + "- " + evaluations.Mark + "\n";

                }
            }
        }
        worksheet.Cell(currentRow, 3).Value = Parameters;
        worksheet.Cell(currentRow, 4).Value = user.result;
    }
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)
ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ПРАЦІВНИКІВ ОРГАНІЗАЦІЇ

Виконав: студент 4 курсу, групи 1КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Волох В. С.
(прізвище та ініціали)

Керівник: к.т.н. доцент. каф. КН
Белзецький Р.С.
(прізвище та ініціали)

«03» червня 2025 р.

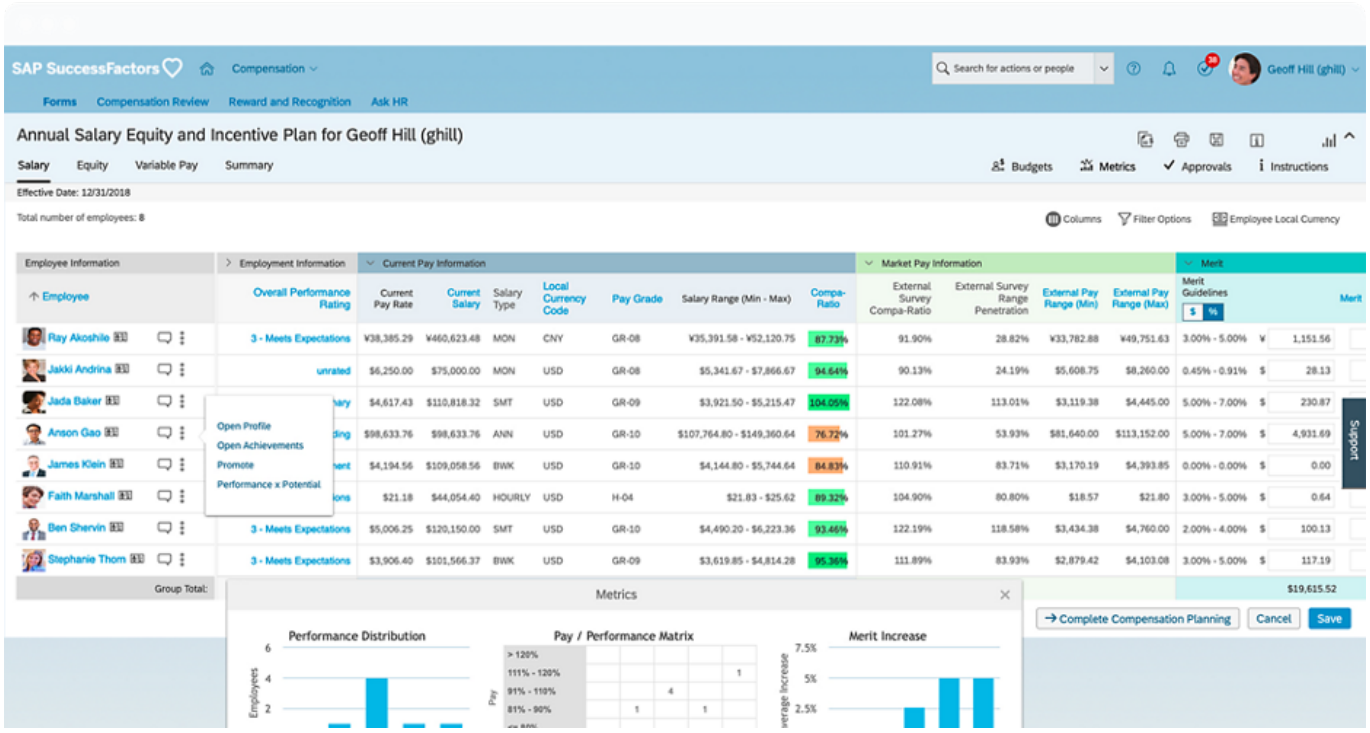

Рисунок Г.1 – Інтерфейс додатку SAP SuccessFactors

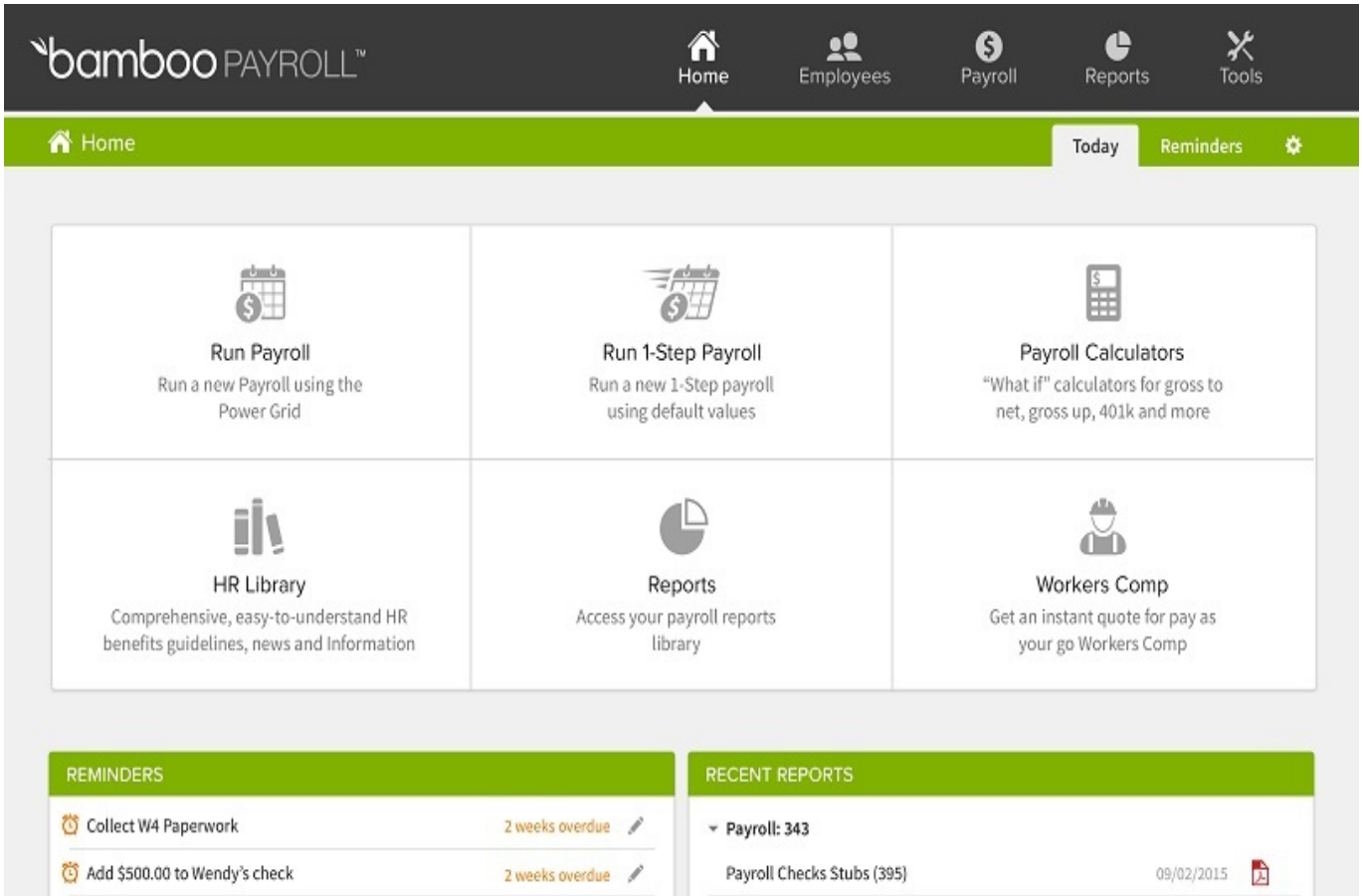

Рисунок Г.2 – Інтерфейс додатку BambooHR

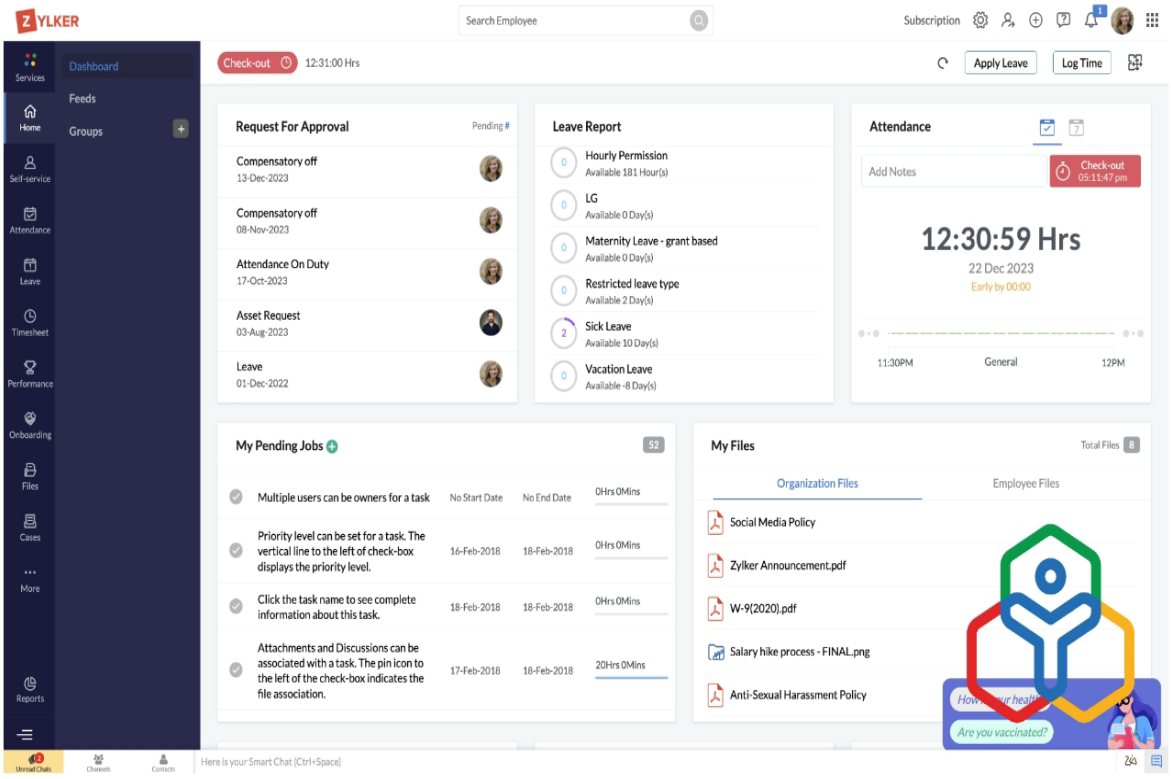

Рисунок Г.3 – Інтерфейс додатку Zoho People

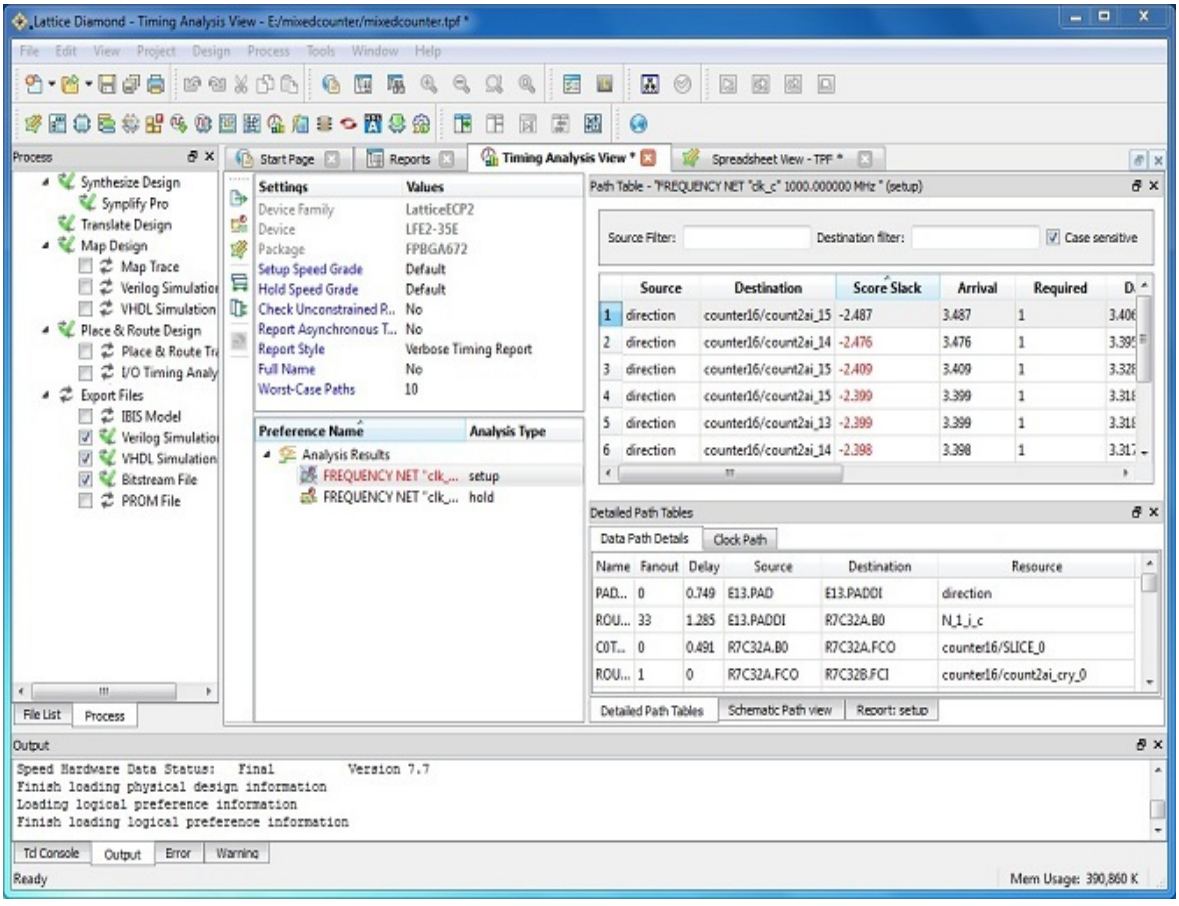

Рисунок Г.4 – Інтерфейс додатку Lattice

Рисунок Г.5 – Структурна схема програмного додатку

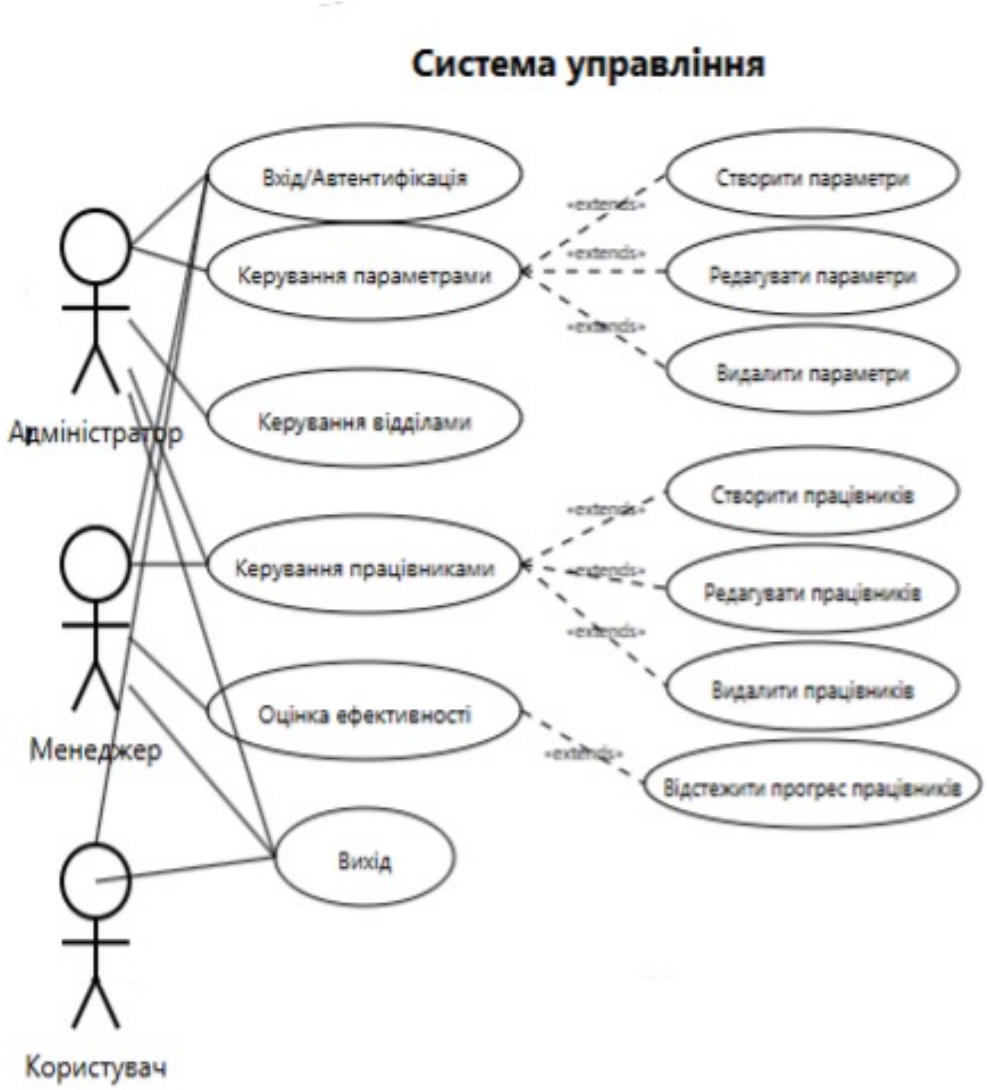

Рисунок Г.6 – Діаграма прецедентів

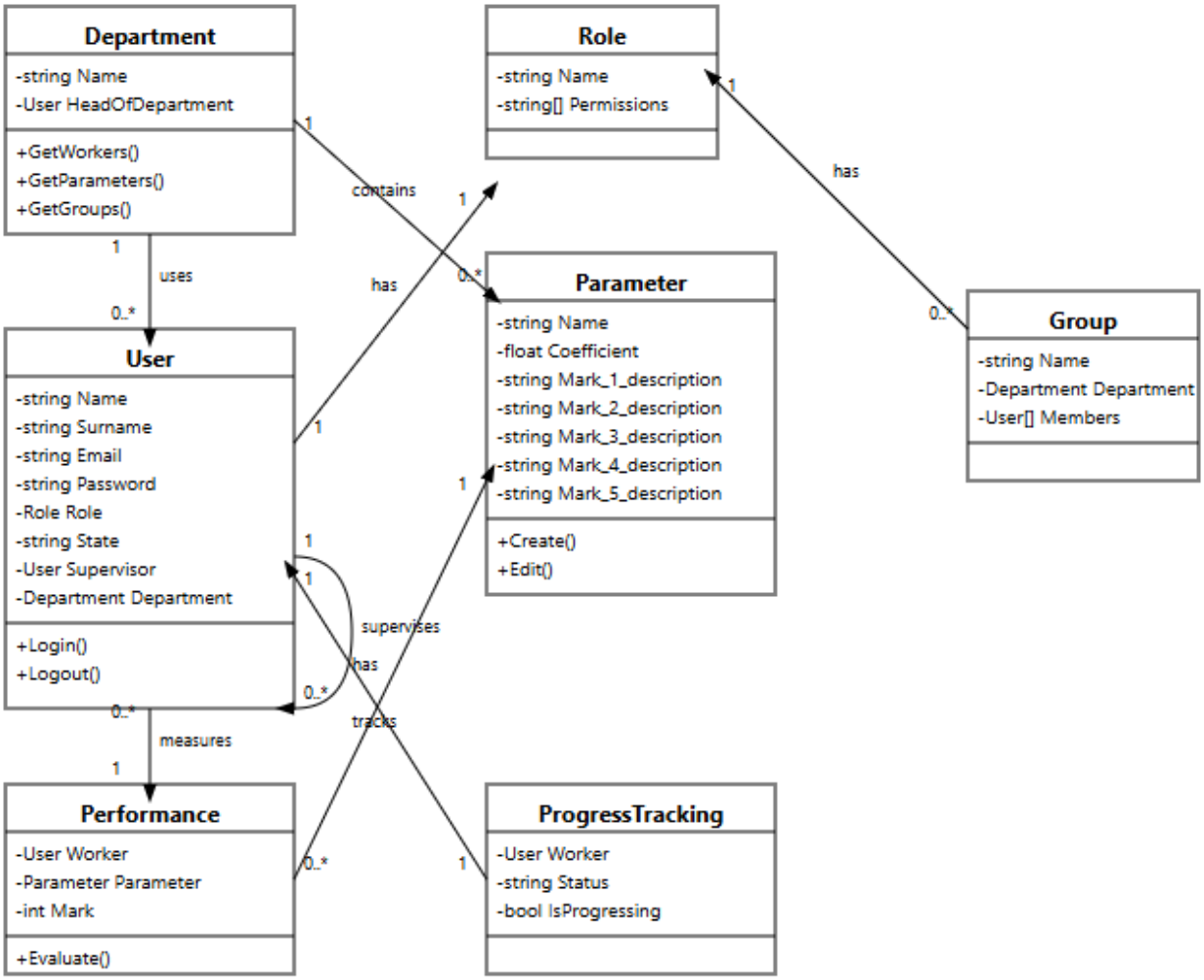

Рисунок Г.7 – Структурна діаграма розроблюваного модуля

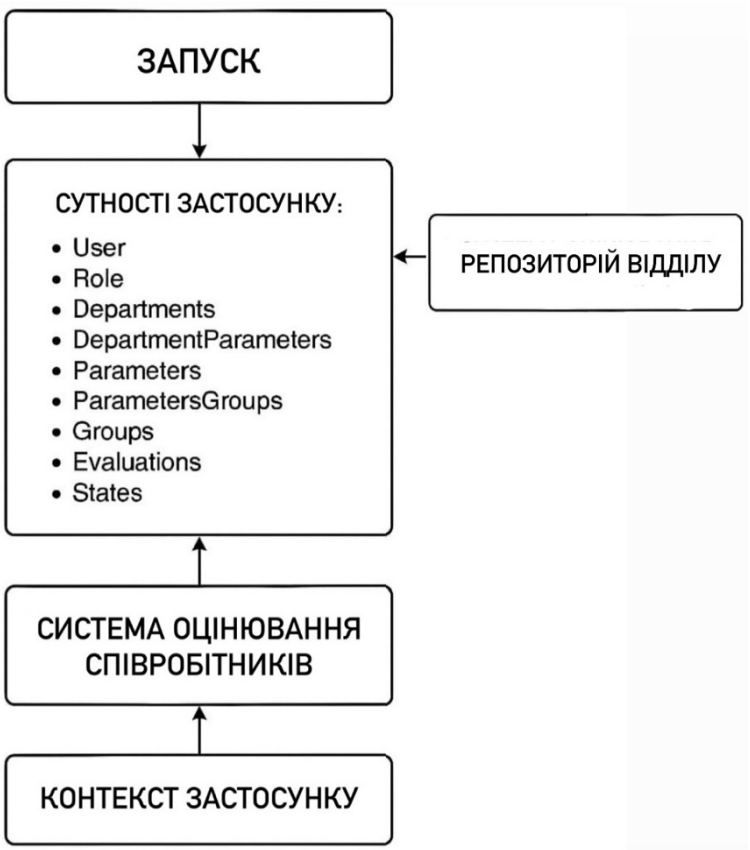

Рисунок Г.8 – Структурна діаграма розроблюваного модуля

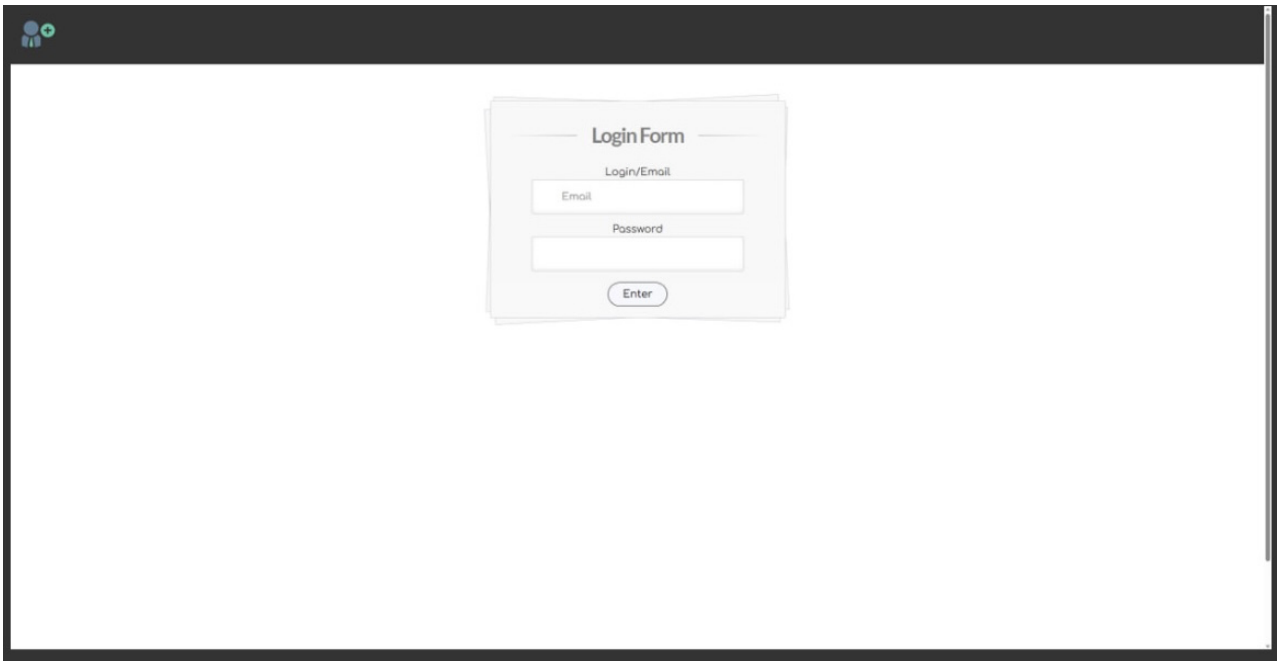

Рисунок Г.9 – Структурна діаграма розроблюваного модуля

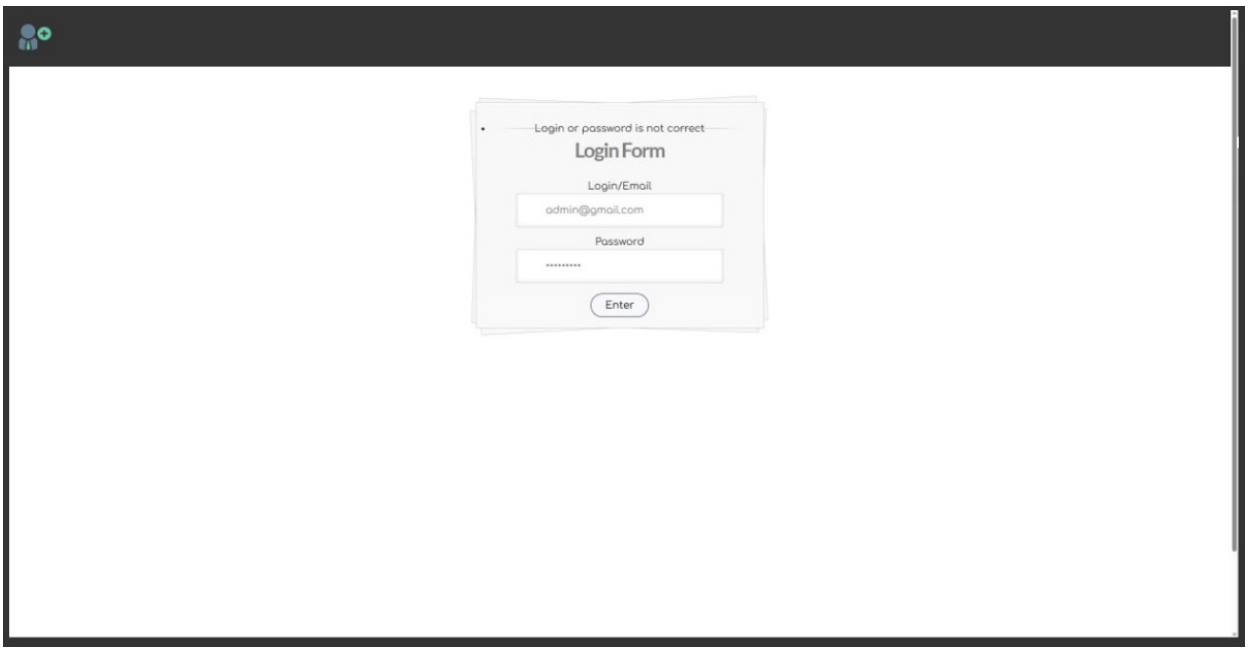

Рисунок Г.10 – Інтерфейс форми входу до системи програмного додатку з невірними даними

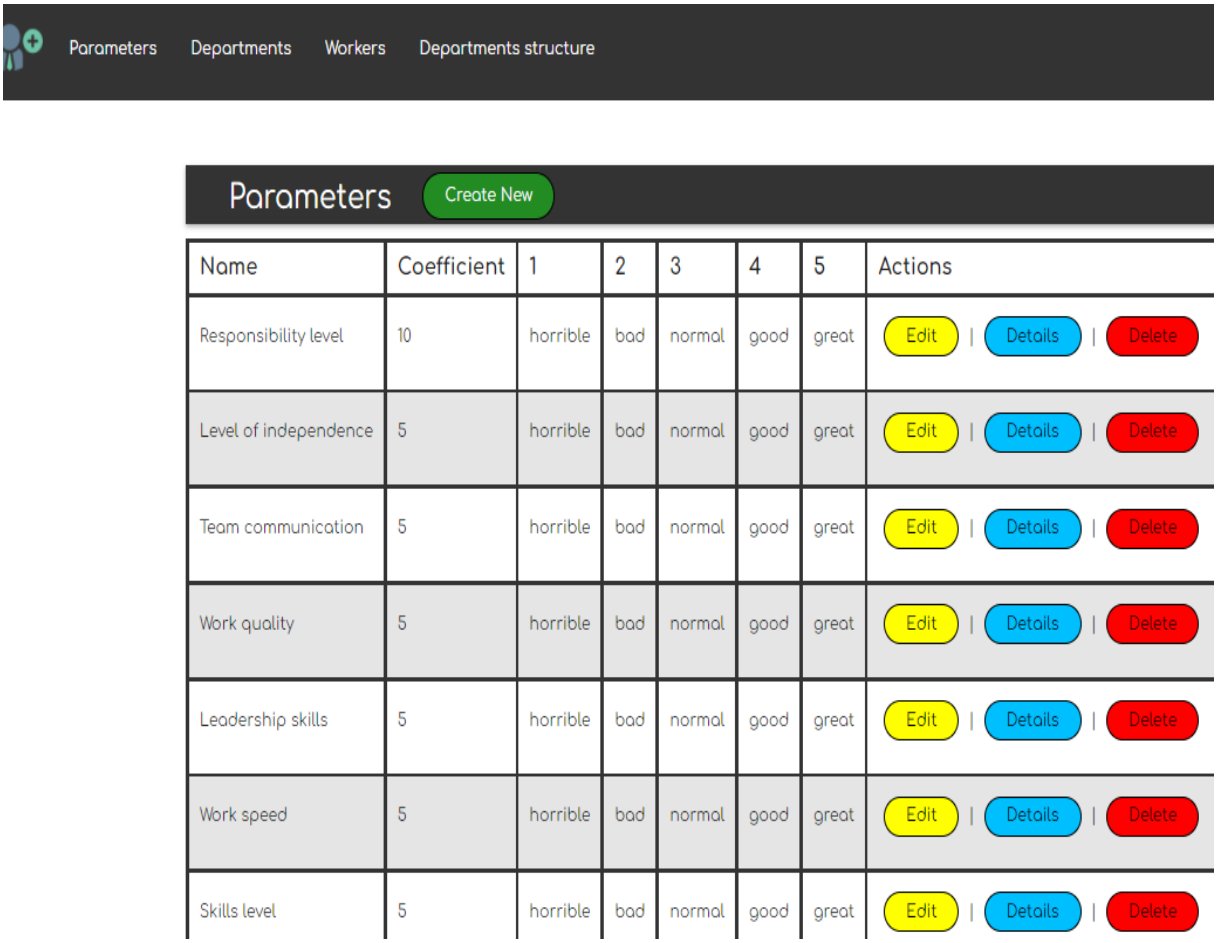

Рисунок Г.11 – Головне вікно програмного модуля

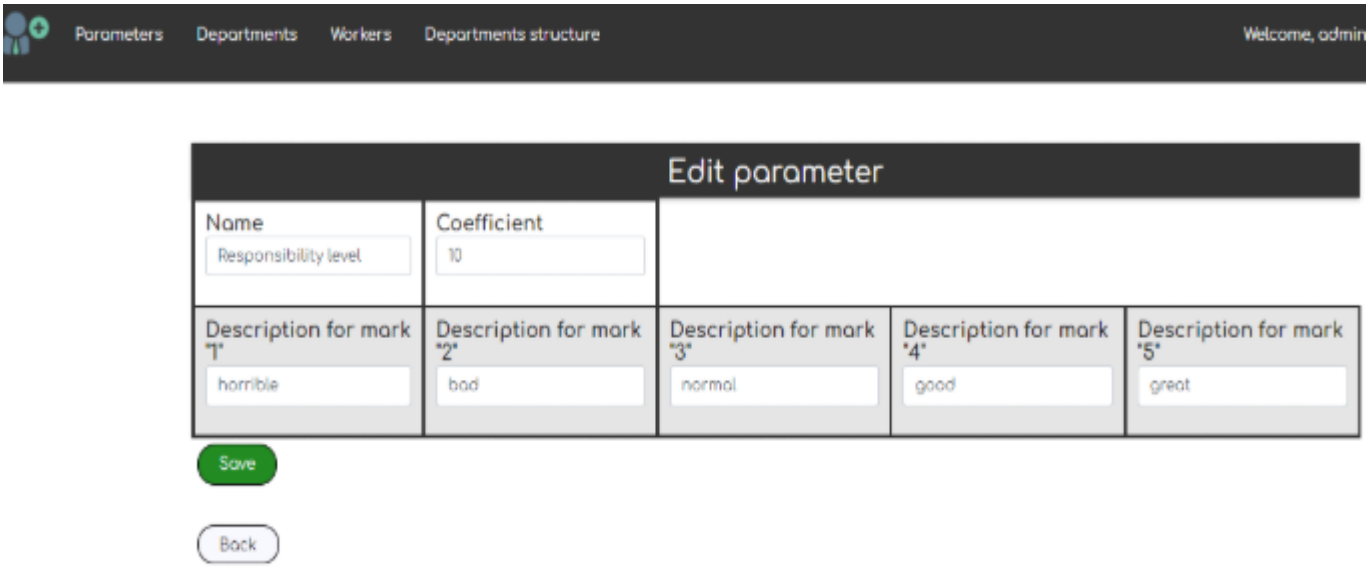

Рисунок Г.12 – Редагування параметрів оцінювання ефективності працівників організації

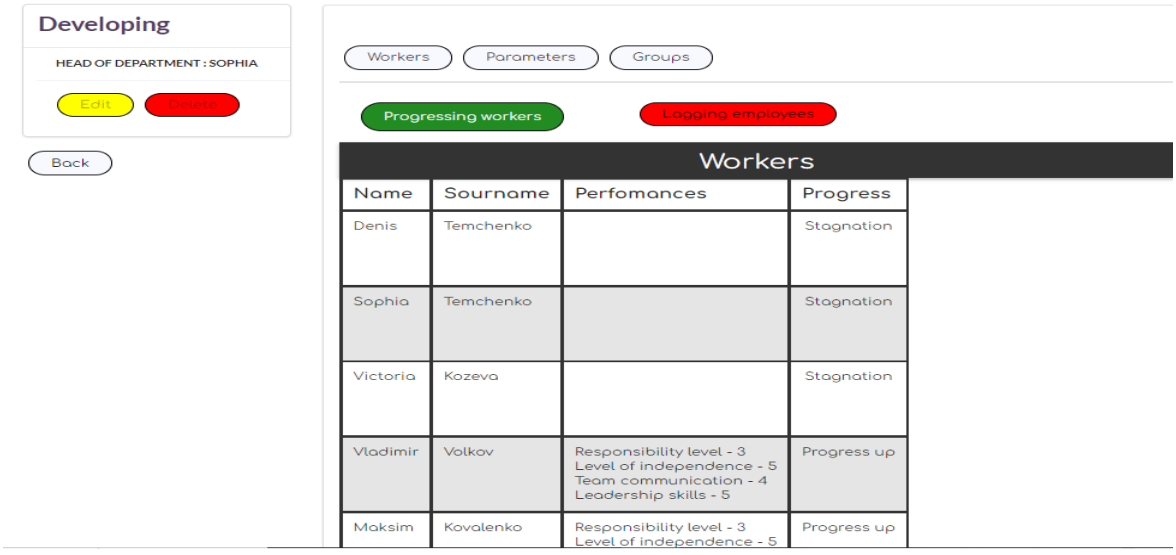

Рисунок Г.13 – Результати проведеного оцінювання працівників

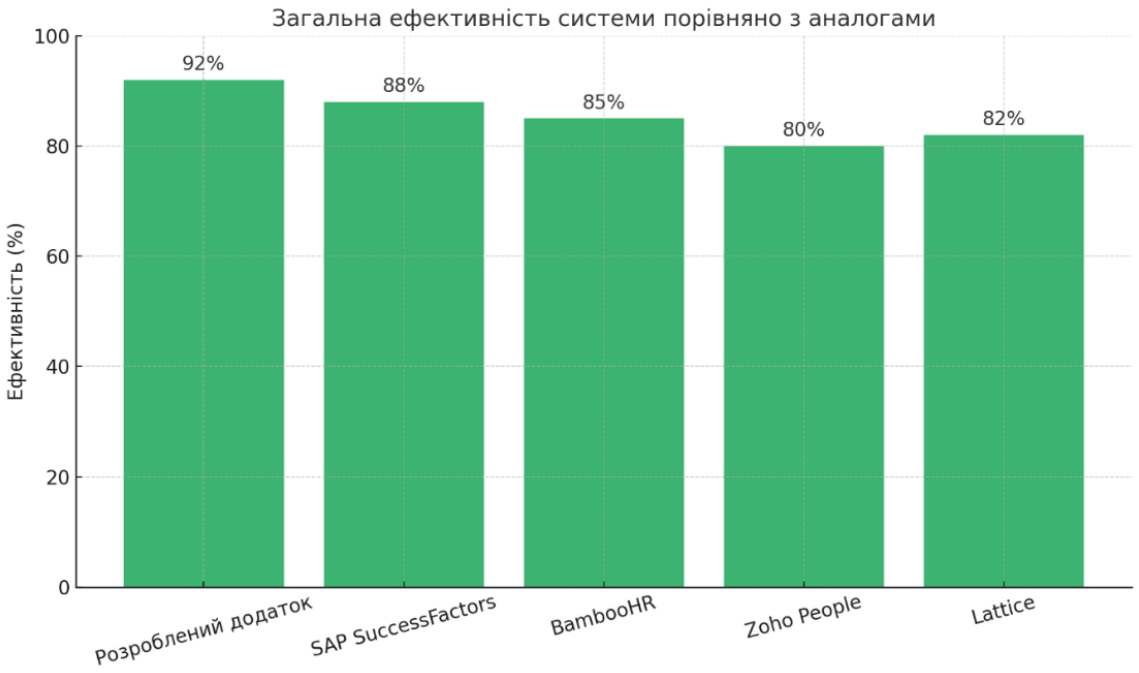

Рисунок Г.14 – Графік порівняння ефективності розробленого додатку з аналогами

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Ім'я**	**Прізвище**	**Група**	**Коефіцієнт**	**Оцінка 1**	**Коефіцієнт**	**Оцінка 2**	**Коефіцієнт**	**Оцінка 3**	**Коефіцієнт**	**Оцінка 4**	**Коефіцієнт**	**Оцінка 5**	**Результат**
2	Denis	Temchenko	Group1	0,8	5	0,6	3	0,7	4	0,5	2	0,9	3	12,3
3	Sophia	Temchenko	Group1	0,8	4	0,6	4	0,7	5	0,5	3	0,9	4	14,2
4	Viktoria	Сидоренко	Group1	0,8	3	0,6	5	0,7	4	0,5	4	0,9	5	14,7
5	Volodimir	Volkov	Group1	0,8	5	0,6	3	0,7	5	0,5	5	0,9	3	14,5
6	Maksim	Kovalenko	Group1	0,8	4	0,6	4	0,7	4	0,5	2	0,9	4	13
7	Tetiana	Boyko	Group1	0,8	3	0,6	5	0,7	5	0,5	3	0,9	5	14,9
8	Oleg	Melnik	Group1	0,8	5	0,6	3	0,7	4	0,5	4	0,9	3	13,3
9	Yulia	Gavrilyuk	Group1	0,8	4	0,6	4	0,7	5	0,5	5	0,9	4	15,2
10	Bogdan	Shevchenko	Group1	0,8	3	0,6	5	0,7	4	0,5	2	0,9	5	13,7
11	Irina	Kozak	Group1	0,8	5	0,6	3	0,7	5	0,5	3	0,9	3	13,5

Рисунок Г.15 – Таблиця результатів оцінювання ефективності працівників організації