ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Катерина ВДОВИЧЕНКО
(підпис)

Виконала: здобувачка вищої освіти групи ПД-43

Катерина ВДОВИЧЕНКО

Керівник: Юрій ЗАДОНЦЕВ
кандидат технічних наук

Рецензент:

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Вдовиченко Катерині Вячеславівні

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель»
керівник кваліфікаційної роботи: кандидат технічних наук Юрій ЗАДОНЦЕВ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: інформація з організції, підготовки та проведення тендерів з метою з'ясування та вибору найкращих постачальників для закупівлі на основі рейтингової оцінки; наукові матеріали та публікації з обгрунтування та визначення показників та коефіцієнтів оцінювання постачальників; приклади сучасних програмних рішень з проведення тендерів; технічна документація щодо використання технологій та інструментів для розробки програмних засобів для проведення тендерів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз процесу тендерних закупівель і методів оцінювання постачальників.

2. Проєктування веб-застосунку для рейтингової оцінки постачальників.
3. Програмна реалізація та опис функціонування системи рейтингової оцінки постачальників.
4. Тестування програмного забезпечення.

5. Перелік графічного матеріалу: *презентація*
1. Мета, об’єкт та предмет дослідження.
2. Задачі кваліфікаційної роботи.
3. Аналіз аналогів.
4. Вимоги до програмного забезпечення.
5. Програмні засоби реалізації.
6. Діаграма прецедентів.
7. Діаграма послідовності.
8. Діаграма пакетів.
9. Діаграма компонентів.
10. Екранні форми
11. Екранні форми.
12. Апробація результатів дослідження
13. Висновки

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Огляд існуючих алгоритмів обробки тексту та методів і технологій побудови тематичного словника	08.03-17.03.2026	
4	Проєктування застосунку для автоматизованої побудови тематичного словника навчальної дисципліни	18.03-27.03.2026	
5	Програмна реалізація застосунку	28.03-10.04.2026	
6	Тестування застосунку	11.04-21.04.2026	
7	Оформлення роботи: вступ, висновки, реферат	22.04-29.04.2026	
8	Розробка демонстраційних матеріалів	30.04-10.05.2026	

9	Попередній захист роботи	11.05-22.05.2026	

Здобувач(ка) вищої освіти _____ Катерина ВДОВИЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Юрій ЗАДОНЦЕВ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: _58_ стор., __1__ табл., _32_ рис., _20_ джерел.

Мета роботи – підвищення прозорості та спрощення вибору постачальників в умовах тендерних закупівель за рахунок розробки web-застосунку для рейтингової оцінки постачальників.

Об'єкт дослідження – процеси рейтингової оцінки постачальників у системі тендерних закупівель.

Предмет дослідження – методи та технології розробки програмного забезпечення для рейтингової оцінки постачальників.

Короткий зміст роботи: У роботі проведено аналіз процесу тендерних закупівель та існуючих методів оцінювання постачальників. Проаналізовано інструментальні засоби розробки web-застосунків: мову програмування C#, фреймворк ASP.NET Core MVC, технології Entity Framework Core та бібліотеки Identity. Розроблено алгоритм роботи застосунку та програмно реалізовано ключовий функціонал, що включає: введення й обробку даних про постачальників керування показниками та учасниками тендеру, встановлення вагових коефіцієнтів оцінювання, а також автоматизоване формування рейтингу постачальників для визначення переможців. В роботі використано мову програмування C#, фреймворк ASP.NET Core MVC, Entity Framework Core, Identity, а також HTML, CSS і JavaScript для реалізації інтерфейсу.

Сферою використання застосунку є оптимізація процесу тендерних закупівель та підвищення об'єктивності оцінювання постачальників.

КЛЮЧОВІ СЛОВА: ТЕНДЕРНІ ЗАКУПІВЛІ, ПОСТАЧАЛЬНИКИ, РЕЙТИНГОВА ОЦІНКА, АНАЛІЗ ДАНИХ, СИСТЕМА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, WEB-ЗАСТОСУНОК, КОЕФІЦІЄНТИ, ПРОЗОРІСТЬ, ЕФЕКТИВНІСТЬ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI - User Interface

API - Application Programming Interface

KPI - Key Performance Indicators

UML - Unified Modeling Language

ER - Entity-Relationship

SQL - Structured Query Language

DDL - Data Definition Language

ORM - Object-Relational Mapping

HTTP - HyperText Transfer Protocol

SOA - Service-Oriented Architecture

SOAP - Simple Object Access Protocol

REST - Representational State Transfer

MVC - Model-View-Controller

EDA - Event-Driven Architecture

DDD - Domain-Driven Design

CSS - Cascading Style Sheets

HTML - HyperText Markup Language

JS - JavaScript

ВСТУП

Актуальність: ефективнність процесу тендерних закупівель суттєво впливає на діяльність організацій, оскільки результативність роботи, якість отриманих товарів та послуг, а також оптимізація витрат безпосередньо залежать від обґрунтованості вибору постачальників. У сучасних умовах актуальною є проблема недостатньої прозорості процесів та складності об’єктивного оцінювання постачальників, що зумовлює прийняття неефективнних управлінських рішень. Значна частка процесів виконується вручну, що ускладнює аналіз даних та знижує точність оцінювання.

Впровадження спеціалізованого програмного забезпечення для рейтингової оцінки постачальників дозволяє систематизувати дані, підвищити об’єктивність аналізу та забезпечити прозорість процесу відбору учасників тендеру. Розробка web-застосунку для зазначених цілей надає інструментарій для введення, обробки та аналізу даних, а також забезпечує формування об’єктивних рейтингів постачальників на основі чітко визначених критеріїв.

Об’єктом дослідження є процеси рейтингової оцінки постачальників у системі тендерних закупівель.

Предметом дослідження є методи та технології розробки програмного забезпечення для рейтингової оцінки постачальників.

Метою роботи є підвищення прозорості та спрощення вибору постачальників в умовах тендерних закупівель за рахунок розробки web-застосунку для рейтингової оцінки постачальників.

Для досягнення поставленої мети було проведено комплексний аналіз предметної області тендерних закупівель та існуючих методологій оцінювання постачальників. Для розробки програмного забезпечення було обрано наступний технологічний стек: C#, ASP.NET Core MVC, Entity Framework Core, Identity, HTML, CSS та JavaScript. У процесі розробки було забезпечено функціонал системи, що включає введення даних про постачальників, керування показниками

та учасниками тендеру, введення коефіцієнтів, а також автоматизоване формування рейтингу. На завершальному етапі було проведено тестування розробленого програмного забезпечення для підтвердження його належної функціональності.

Наукова новизна роботи полягає у розробці програмного забезпечення, призначеного для рейтингової оцінки постачальників у процесі тендерних закупівель на основі визначених критеріїв.

Практична значущість полягає у створенні програмного забезпечення, призначеного для оптимізації процедури тендерних закупівель, забезпечення об’єктивності оцінювання постачальників, а також підтримки прийняття управлінських рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТЕНДЕРНИХ ЗАКУПІВЕЛЬ

1.1 Опис предметної області оцінювання постачальників у тендерних закупівлях

Комплексне розуміння особливостей модернізації тендерних процедур за допомогою впровадження систем рейтингування постачальників потребує глибокого аналізу цієї сфери. У роботі досліджується специфіка проведення тендерів, виявляються основні недоліки існуючих підходів та обґрунтовуються проблеми та потенційні переваги інтеграції об’єктивних інструментів оцінювання.

Процес тендерних закупівель охоплює низку послідовних етапів: від визначення потреб до вибору постачальника, аналізу пропозицій, укладання угод та подальшої координації співпраці [1]. На цих етапах часто виникають проблеми, зумовлені високим рівнем суб’єктивізму під час оцінювання. На рішення щодо закупівель часто впливають ручний аналіз, упередження або обмежені дані, що призводить до неоптимальних результатів.

Важливим аспектом є управління ризиками та дотримання вимог. Організації повинні гарантувати, що вибрані постачальники відповідають нормативним стандартам, договірним зобов’язанням і етичним нормам, щоб зменшити юридичні та репутаційні ризики.

Незважаючи на ці проблеми, існують можливості для оптимізації процесу тендерних закупівель. Використання автоматизованих систем, що базуються на аналізі даних і застосуванні чітко визначених критеріїв оцінювання, дозволяє підвищити точність та обґрунтованість прийняття рішень, а також скоротити трудомісткість процесів.

Системи рейтингу постачальників є ефективним інструментом для вирішення зазначених проблем. Систематично оцінюючи ефективнність постачальника на основі заздалегідь визначених критеріїв, таких як надійність, економічна ефективність і відповідність, організації можуть приймати

обґрунтовані рішення та зменшувати ризики.

Впровадження таких систем включають покращену прозорість і підзвітність у виборі постачальників, покращене управління ризиками, підвищення операційної ефективності та кращу узгодженість зі стратегічними цілями.

Крім того, критично важливим є проведення аналізу зацікавлених сторін. Врахування перспектив і потреб фахівців із закупівель, постачальників, регуляторних органів та кінцевих користувачів забезпечує, що запропоноване програмне рішення ефективно відповідає їхнім вимогам.

Здійснюючи комплексний системний аналіз, ми отримуємо ґрунтовну інформацію щодо специфіки та складності процесу тендерних закупівель. Цей аналіз є фундаментом для проектування та розробки програмного рішення, яке ефективно вирішує проблеми та забезпечує відчутні переваги організаціям, які беруть участь у тендерних закупівлях.

Сфера створення розробки програмного забезпечення рейтингової оцінки постачальників з метою оптимізації тендерних закупівель лежить на перетині управління закупівлями, аналізу даних та програмної інженерії.

Управління закупівлями передбачає стратегічне забезпечення організації необхідними товарами і послугами. Цей процес охоплює різні заходи, зокрема: ідентифікацію постачальника, оцінку, вибір, переговори та управління контрактами. Ефективне управління закупівлями є критично важливе для організацій, щоб досягти економії коштів, підтримувати стандарти якості та пом’якшити ризики у взаємодії з постачальниками.

У сфері управління закупівлями тендерний процес є ключовим етапом, під час якого організації здійснюють відбір пропозицій від потенційних постачальників для виконання їхніх вимог. Цей процес включає формування запитів на подання пропозицій (RFP) або запрошення до участі в тендері (ITT), оцінку пропозицій постачальників та подальше укладення контрактів з обраними постачальниками. Ефективність процесу тендерних закупівель істотно впливає на ефективність організації та конкурентоспроможність.

Однією з ключових проблем у процесі тендерних закупівель є суб’єктивність під час оцінювання та вибору постачальника. Фахівці із закупівель часто покладаються на ручні оцінювання, минулий досвід або обмежені дані для прийняття рішень, через що виникають неузгодженість та неефективність. Крім того, забезпечення дотримання нормативних вимог і договірних зобов’язань ускладнює управління постачальниками.

Щоб вирішити ці перешкоди та оптимізувати процес тендерних закупівель, запропонований web-застосунок спрямований на розробку рейтингової системи для постачальників. Ця система використовуватиме методи аналізу даних для об’єктивної оцінки ефективності постачальника за різними критеріями, такими як надійність, економічна ефективність і відповідність вимогам. Надаючи професіоналам із закупівель практичну інформацію та можливості підтримки в ухваленні рішень, програмне забезпечення дозволить здійснювати більш обґрунтовані та ефективні рішення при виборі постачальників.

Розробка програмного забезпечення включає кілька ключових етапів, зокрема збір вимог, проектування системи, безпосередню реалізацію програмного забезпечення, тестування та розгортання. Це вимагає досвіду розробки програмного забезпечення, аналізу даних, розробки інтерфейсів користувача, а також інтеграції з наявними системами закупівель.

Кінцевою метою розробки web-застосунку рейтингової оцінки постачальників є підвищення ефективності, прозорості та результативності процесу тендерних закупівель. Надаючи організаціям інструменти для ідентифікації та вибору найефективніших постачальників, програмне забезпечення сприяє економії коштів, посиленню контролю якості та мінімізацію ризиків у відносинах з постачальниками. Крім того, це сприяє підвищенню рівню підзвітності та дотриманню нормативних стандартів, що сприяє зміцненню конкурентоспроможності та загальному успіху організацій на ринку.

1.2 Аналіз вимог до системи рейтингової оцінки постачальників

Розробка програмного забезпечення рейтингової системи постачальників з метою оптимізації тендерних закупівель потребує здійснення комплексного аналізу системних вимог. Даний аналіз передбачає визначення функціональних та нефункціональних вимог, яким повинно відповідати програмне забезпечення для задоволення потреб зацікавлених сторін та ефективнного вирішення проблем у межах процесу закупівель.

Функціональні вимоги:

1. управління даними постачальників: забезпечення функціоналу для введення та збереження даних про постачальників з метою їх подальшого комплексного оцінювання;
2. внесення змін: редагування та видалення коефіцієнтів, що використовуються для розрахунку рейтингу;
3. керування користувачами: система повинна підтримувати механізми автентифікації та авторизації користувачів, щоб лише авторизовані користувачі могли отримати доступ і змінити оцінки постачальників та іншу конфіденційну інформацію;
4. розрахунок рейтингу: програмне забезпечення повинно застосовувати алгоритми для розрахунку рейтингів постачальників на основі зважених балів, присвоєних відповідним критеріям ефективнності;
5. звітування та візуалізація: система повинна передбачати можливість для формування звітів, візуального відображення рейтингів постачальників та ключових показників ефективнності.

Нефункціональні вимоги:

1. продуктивність системи: формування результатів аналізу тендера не перевищує 5 секунд за умови стандартного навантаження;
2. автентифікація користувачів: забезпечення автентифікації користувачів із перевіркою облікових даних під час входу в систему та відображенням повідомлень про помилки у разі некоректного введення;

3. зручність використання: забезпечення доступу до основних функцій у межах двох або трьох кроків від головної сторінки;

4. швидкодія системи: забезпечення обробки даних постачальників, показників і коефіцієнтів у процесах введення, перегляду та формування рейтингу, із часом відгуку системи до 5 секунд без зниження швидкодії інтерфейсу.

На основі проведеного аналізу вимог до програмного забезпечення встановлено, що розроблене рішення повною мірою відповідає потребам зацікавлених сторін, вирішує проблемні аспекти процесу тендерних закупівель, а також сприяє оптимізації управління постачальниками та процесу прийняття управлінських рішень [3].

Розробка ефективнного інтерфейсу для програмного забезпечення оцінювання постачальників є критично важливим чинником забезпечення належного рівня користувацького досвіду та функціональної зручності. Цей аналіз охоплює як інтерфейс користувача (UI), так і інтерфейс прикладного програмування (API) з метою забезпечення безперешкодної інтеграції з іншими системами.

Вимоги до інтерфейсу користувача (UI):

1. інтуїтивно зрозуміла інформаційна панель повинна забезпечувати користувачам можливість отримання чіткого огляду рейтингів постачальників, динаміки показників та ключових індикаторів ефективнності (KPI), а також гарантувати зручну навігацію між програмними розділами;

2. профілі постачальників мають містити інформацію, таку як продуктивність та поточні рейтинги, з доступом і можливостями редагування;

3. можливість налаштовувати критерії рейтингу дають змогу користувачам адаптувати систему рейтингу до конкретних вимог до закупівель, підвищуючи гнучкість і налаштування.

Вимоги до інтерфейсу прикладного програмування (API):

1. інтеграція з наявними системами закупівель має здійснюватися шляхом безперебійної взаємодії через API, що забезпечує обмін даними та їх синхронізацію для оптимізації робочих процесів;

2. функція імпорту та експорту даних через API полегшує ефективнність операцій масивів інформації та формуванню звітності, підтримуючи потреби аналізу та задовільняючи потреби користувачів;

3. механізм автентифікації та авторизації в API забезпечують безпечний доступ до даних постачальників, забезпечуючи конфіденційність та запобігаючи неавторизованому доступу.

Дотримування цих вимог до інтерфейсу гарантує стабільне функціонування програмного забезпечення з одночасним забезпеченням ефективнної інтеграції із системами закупівель, що дозволяє оптимізувати процес участі у тендерах.

Розробка ефективнного інтерфейсу програмного забезпечення має вирішальне значення для оптимізації процесу тендерних закупівель через систему рейтингу постачальників. Аналіз охоплює як інтерфейс користувача (UI), так і інтерфейс програмування додатків (API).

Для інтерфейсу користувача є критично важливим наявність інтуїтивно зрозумілої інформаційної панелі, що забезпечує огляд рейтингів постачальників та їх профілів із можливістю налаштування критеріїв. Інструменти візуалізації даних і функції пошуку та фільтрації сприяє підвищенню зручності роботи, тоді як адаптивний дизайн забезпечує доступність на різних пристроях.

У частині функціонування API обов’язковим є бездоганна інтеграція із системами закупівель, можливості імпорту та експорту даних і надійних заходів безпеки.

У підсумку, дотримання зазначених вимог до інтерфейсу забезпечує безперебійну роботу користувача та ефективнну інтеграцію, надаючи користувачам змогу ефективнно оптимізувати процес тендерних закупівель.

1.3 Моделювання процесу оцінювання постачальників

Уніфікована мова моделювання (UML) — це стандартизована мова візуального моделювання, що використовується в розробці програмного забезпечення та проектуванні систем. Це полегшує представлення, документування та передачу інформації щодо структури, поведінки та взаємодії програмних систем [5].

UML надає стандартизований набір графічних символів та нотацій, о дозволяють розробникам програмного забезпечення, дизайнерам та зацікавленим сторонам створювати візуальні представлення різних аспектів системи. Зазначені візуалізації допомагають зрозуміти, проаналізувати та задокументувати архітектуру системи, компоненти, зв’язки та процеси.

Даний інструментарій можна використовувати для моделювання різних типів систем, зокрема програмні додатки, бази даних, бізнес-процеси та організаційні структури. UML забезпечує уніфікований підхід до створення діаграм і моделей, що оптимізує комунікацію та сприяє ефективнній взаємодії між членами команди, залученими до розробки та обслуговування системи.

Діаграми UML охоплюють кілька типів, включаючи діаграми класів, діаграми варіантів використання, діаграми послідовності, діаграми діяльності, діаграми кінцевих автоматів, діаграми компонентів і діаграми розгортання. Кожен із зазначених типів слугує інструментом для візуалізації специфічних аспектів архітектури та поведінкових характеристик системи.

Ці UML-діаграми служать візуальними представленнями архітектури та поведінки системи, допомагаючи зацікавленим сторонам краще зрозуміти, проаналізувати та повідомити дизайн і функціональність системи. UML забезпечує чітку та лаконічну комунікацію між учасниками процесів розробки програмного забезпечення.

При побудові моделей складних систем в UML можна застосовувати такі принципи, як розділення підсистем. Розробка підсистеми довела ефективнність в управлінні складністю системи. Підсистеми можуть бути структуровані як окремі

функціональні блоки або логічні групи компонентів. В якості альтернативи можна застосовувати рівні абстракції. Різні рівні абстракції дозволяють представити систему на різних рівнях деталізації. Наприклад, використання загальних класів на високому рівні, а потім їх деталізація у відповідних підкласах на нижчих рівнях. Додатково в процесі моделювання застосовуються принципи агрегації та композиції. Використання зв'язків агрегації та композиції додатково описує зв'язки між компонентами системи. Агрегація вказує на те, що один компонент містить інші компоненти або складається з них, тоді як композиція демонструє сильну приналежність одного компонента іншому.

Зазначені принципи сприяють створенню зрозумілих, гнучких і масштабованих моделей складних систем засобами UML. Вони забезпечують належний рівень якості аналізу та проектування, а також оптимізують комунікацію між розробниками, дизайнерами та іншими зацікавленими сторонами проекту.

UML діаграма є графічним представленням, що відображає структурні, поведінкові або взаємодіючі аспекти системи чи програмного додатка. UML передбачає різні типи діаграм, кожна з яких має специфічне призначення та систему позначень, що дозволяє візуалізувати та передавати складні аспекти архітектури системи.

Діаграми UML слугують інструментом для збору та передачі відомостей про архітектуру системи, її компоненти, зв'язки, поведінку та взаємодію між елементами. Вони використовуються для поглиблення розуміння, аналізу, проектування та документування систем програмного забезпечення [5].

1.3.1 Діаграма прецедентів

Діаграма прецедентів (Use-case diagram) – це графічна модель, що відображає функціональні можливості програмної системи, доступні для кожної категорії користувачів.

Основними компонентами діаграми є актори та варіанти використання.

Актор представляє сукупність логічно пов'язаних ролей, що реалізуються під час взаємодії з варіантами використання або сутностями (системою, підсистемою

або класом). Актором може виступати як користувач, так і зовнішня система, підсистема чи клас, що знаходяться поза межами проектованої сутності. Графічно актор зображений у вигляді фігурки.

Випадок використання описує серію послідовних подій (включаючи варіанти), які виконує система, що призводить до досягнення результату актором. Варіант використання відображає поведінку сутності, описуючи взаємодію між акторами та системою. Він показує не те, як досягається певний результат, а скоріше те, що виконується. Варіанти використання представлені дуже просто - у формі еліпса, всередині якого вказано його назву.

Спроєктована діаграма прецедентів представлена на рисунку 1.1.

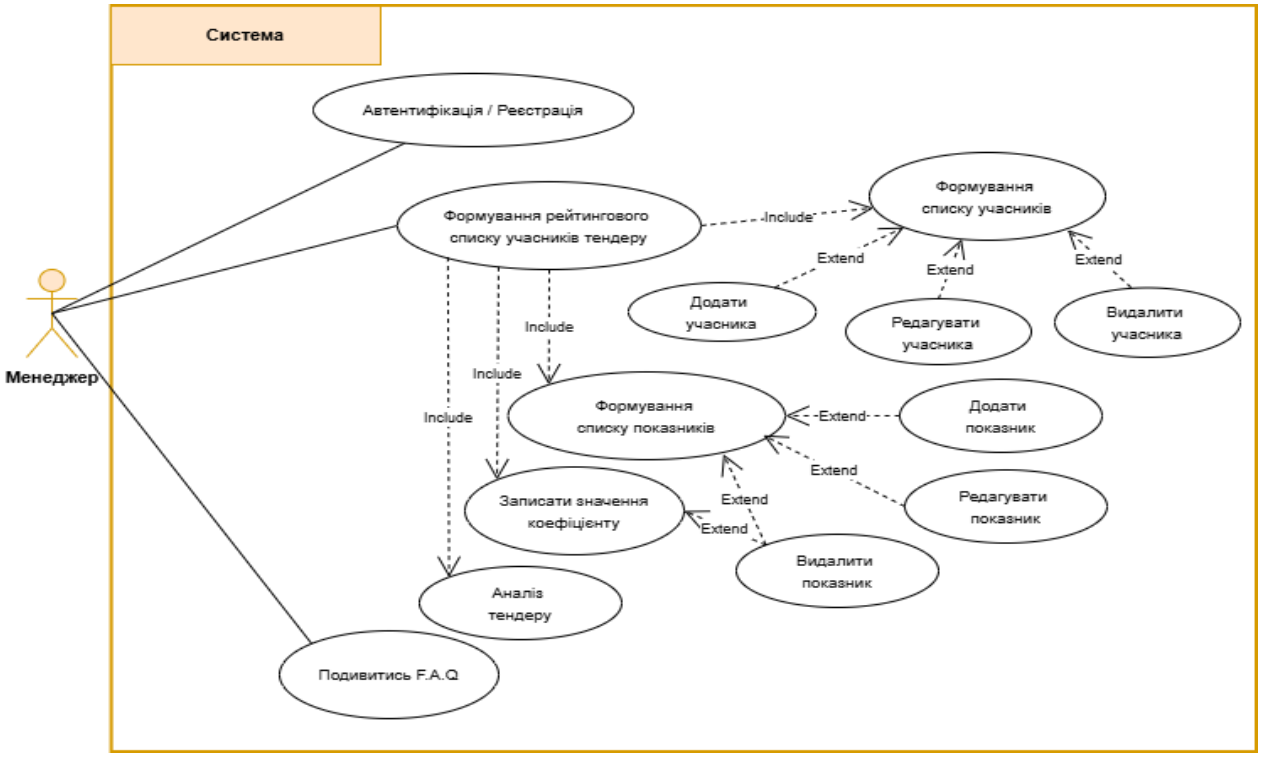


Рис. 1.1 Діаграма прецедентів

Створена діаграма прецедентів містить актора: «Менеджер закупівель».

Актор «Менеджер закупівель» включає такі прецеденти:

- автентифікація / реєстрація;
- формування рейтингового списку учасників тендеру;
- формування списку учасників;

- додати, редагувати та видалити учасника;
- формування списку показників;
- додати, редагувати та видалити показник;
- записати значення коефіцієнту;
- аналіз тендеру;
- перегляд F.A.Q.

Прецеденти певним чином залежать одне від одного.

Прецедент «Формування рейтингового списку учасників тендеру» включає такі прецеденти, як «Формування списку учасників», «Формування списку показників» та «Записати значення коефіцієнту».

Прецеденти «Додати учасника», «Редагувати учасника» та «Видалити учасника» розширюють прецедент «Формування списку учасників».

Прецеденти «Додати показник», «Редагувати показник» та «Видалити показник» розширюють прецедент «Формування списку показників».

Розглянемо детальніше вищеописані прецеденти.

Назва прецеденту використання: «Формування рейтингового списку учасників тендеру».

Варіант використання «Формування рейтингового списку учасників тендеру» передбачає автоматизоване формування ранжованого списку учасників на основі визначених критеріїв оцінювання. Даний інструмент призначений для спрощення процесу аналізу та вибору найбільш ефективних постачальників з метою оптимізації тендерних закупівель.

Актори: менеджер закупівель.

Передумови: менеджер із закупівель має бути автентифікованим і авторизованим для доступу до системи. Дані про учасників тендеру та показники оцінювання мають бути доступними й актуальними в системі.

Основний потік:

1. менеджер закупівель здійснює вхід у систему програмного забезпечення та активує функцію «Формування рейтингового списку учасників тендеру;

2. система надає користувачу перелік критеріїв та показників для оцінювання учасників;

3. менеджер закупівель формує список учасників та список показників;

4. користувач вводить значення коефіцієнтів для відповідних показників;

5. система отримує дані про учасників із бази даних і застосовує визначені критерії оцінювання;

6. на основі отриманих результатів система формує рейтинговий список учасників тендеру;

7. менеджер закупівель переглядає сформований рейтинг та може провести додатковий аналіз тендеру;

8. за потреби користувач може зберегти або експортувати результати оцінювання.

Альтернативні потоки:

1. у разі відсутності або неактуальності даних про учасників система надсилає відповідне повідомлення менеджеру закупівель із пропозицією оновити інформацію;

2. у разі виникнення технічних збоїв або помилок під час формування рейтингу система генерує повідомлення про помилку та пропонує повторити спробу або звернутися до служби підтримки.

Винятки:

Якщо менеджер закупівель не має необхідних прав доступу до функції формування рейтингового списку, система обмежує доступ до зазначеної опції та інформує користувача.

Якщо введені коефіцієнти або показники є некоректними чи неповними, система вимагає введення валідних даних перед продовженням процесу оцінювання.

Назва прецеденту використання: «Аналіз тендеру»

Опис: варіант використання «Аналіз тендеру» передбачає аналіз результатів оцінювання учасників тендеру на основі попередньо визначених критеріїв та коефіцієнтів у межах програмної системи. Даний функціонал забезпечує можливість об’єктивного аналізу результатів тендеру та визначення найбільш ефективного постачальника.

Актори: менеджер звкупівель

Передумови: менеджер закупівель має бути автентифікованим і авторизованим для доступу до функції аналізу тендеру. Дані про учасників, показники та результати оцінювання мають бути доступними в системі.

Основний потік:

1. менеджер закупівель переходить до функції «Аналіз тендеру»;
2. система відображає результати оцінювання та рейтинговий список учасників;
3. користувач переглядає інформацію про учасників і результати розрахунків;
4. система надає можливість аналізу показників та коефіцієнтів оцінювання;
5. менеджер закупівель може переглянути додаткову інформацію щодо окремих учасників;
6. система зберігає результати аналізу та оновлює відповідні записи.

Альтернативні потоки:

1. у разі недостатності даних для проведення аналізу система повідомляє користувача про необхідність оновлення або доповнення інформації;
2. у разі виникнення суперечностей у результатах оцінювання система забезпечує можливість повторного аналізу даних.

Винятки:

Якщо під час аналізу виникають технічні помилки, система повідомляє менеджера закупівель і пропонує повторити спробу;

Якщо показники оцінювання або коефіцієнти є некоректними, система вимагає їх перевірки перед продовженням аналізу.

1.3.2 Діаграма послідовності

Діаграма послідовності графічно демонструє порядок взаємодії певних об'єктів програми у часі. Зазвичай, в цій діаграмі демонструється, як користувачі (актори з діаграми варіантів використання) взаємодіють з іншими компонентами програми під час реалізації тих чи інших варіантів використання програми, та як при цьому взаємодіють інші компоненти програмної системи. Зазвичай одна діаграма послідовності присвячена опису одного з варіантів використання, зазначеного в Use-case діаграмі [6].

Діаграми послідовності є інструментом формалізації сценаріїв використання, пропонуючи такі переваги, як завчасне визначення складу взаємодіючих компонентів та детальний опис потоку повідомлень від одного компонента до іншого. Ці компоненти та потоки повідомлень пізніше перетворюються на конкретні класи (об’єкти) та їхні методи відповідно до термінології Java. Отже, модель системних подій, які класи даних (об’єкти) будуть підтримувати та обробляти, одразу уточнюється.

Діаграма послідовностей, зображена на рисунку 1.2, містить такі об’єкти:

- «Менеджер закупівель»;
- «Інтерфейс»;
- «Система»;
- «База даних».

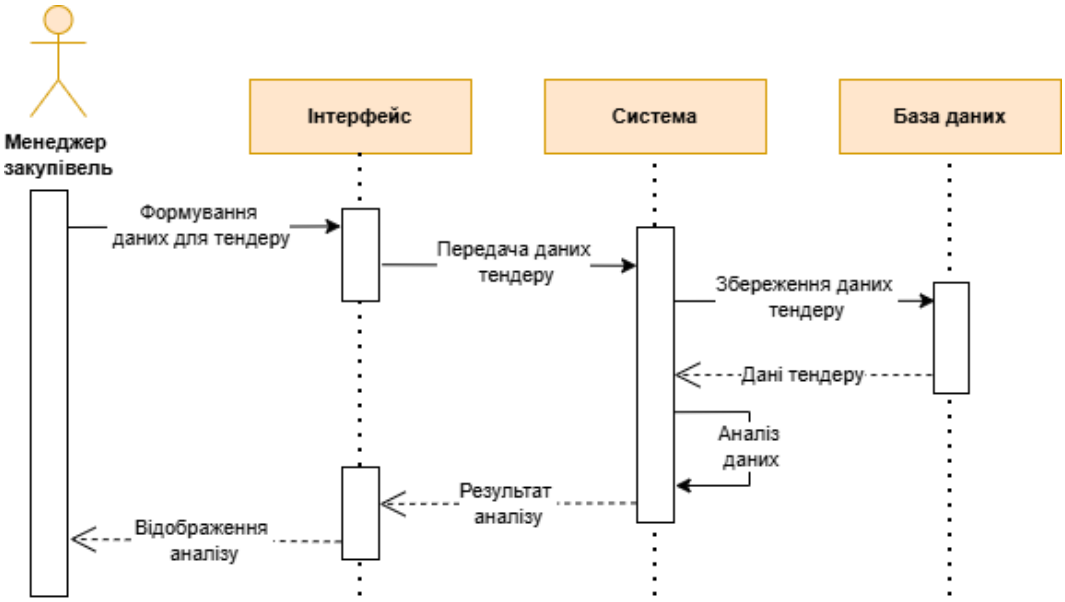


Рис. 1.2 Діаграма послідовності

Об’єкт «Менеджер закупівель» надсилає запит до об’єкта «Інтерфейс» на формування даних для тендеру та отримує у відповідь відображення результатів аналізу. Під час цього «Інтерфейс» передає дані тендеру до об’єкта «Система». Далі «Система» здійснює збереження даних у «Базі даних» та отримує необхідні дані тендеру для подальшої обробки. Після цього система виконує аналіз даних і формує результат аналізу, який передається до об’єкта «Інтерфейс». У результаті інтерфейс відображає менеджеру закупівель результати проведеного аналізу тендеру.

1.4 Огляд інформаційних джерел та існуючих рішень

Проаналізуємо існуючі рішення в зазначеній предметній області.

Система SRM (Supplier Relationship Management) APS SMART — це комплексний інструмент, призначений для оптимізації процесів закупівель і вдосконалення взаємодії з постачальниками. Завдяки застосуванню принципів розширеного планування та планування (APS), система інтегрує передову технологію для оптимізації управління постачальниками та операцій із закупівель [7].

SRM-система APS SMART – це комплексне рішення класу Supplier Relationship Management, що автоматизує управління закупівлями будь-якого бізнесу.

Програмний продукт адаптований для підприємств усіх форм власності України, APS SMART є автоматизованою системою управління бізнес-процесами закупівель повного циклу: від етапу планування закупівель та оптимального вибору постачальників до забезпечення своєчасного постачання на склад.

APS SMART надає всім замовникам доступ до власного електронного тендерного майданчика, який агрегує комерційні тендери всіх замовників та забезпечує оптимізацію та прозорість усіх закупівельних процедур на всіх етапах їх реалізації.

До ключових переваг системи – 100% контроль витрат та зниження витрат на постачання, від 20% фінансової економії завдяки підвищенню оборотності, скороченню складських запасів та ефективнному використанню робочого часу.

Основним завданням APS SMART є формування цілісного інформаційного середовища для всіх етапів та процесів закупівельної діяльності підприємства: від прогнозування потреб до приймання та обліку поставок, як у внутрішніх бізнес-процесах, так і при взаємодії із зовнішніми контрагентами. Автоматизація відділу закупівель є невід'ємною складовою загального процесу операційного управління компанією.

Комплексна інтеграція функціональних модулів системи, що охоплюють управління заявками, тендерами, контрактами та поставками, забезпечує повну автоматизацію та контроль закупівельної діяльності підприємства.

Завдяки функціоналу, система забезпечує взаємодію всіх підрозділів, залучених до закупівельних процесів: бухгалтерії, складу, юридичного, служби безпеки та інших.

Модульна система дозволяє гнучке налаштування моделей робочих бізнес-процесів відповідно до внутрішньої політики та процедур компанії, що робить її універсальною для будь-якого виду діяльності. Система виступає централізованим сховищем даних усіх інформаційних потоків та має можливість

автоматично генерувати аналітику та звітність у різних розрізах.

Кожен модуль стандартизує операційні процедури та спрощує взаємодію та контроль на одному з етапів закупівельного процесу. Додатково, окремі модулі забезпечують роботу в «єдиному вікні» з розмежуванням прав доступу до документації, інтеграцію з обліковими системами підприємства та надає можливість самостійного адміністрування системи.

Ключові особливості

Управління постачальниками: платформа APS SMART надає надійні комплексні інструменти для ефективнного управління відносинами з постачальниками. Система пропонує централізоване зберігання даних про постачальників, моніторинг показників ефективнності та можливості співпраці, що сприяє оперативній ідентифікації та взаємодії з найефективннішими постачальниками.

Оптимізація закупівель: завдяки вискористанню розширених алгоритмів та прогнозної аналітики, APS SMART оптимізує процеси закупівель. Аналізуючи історичні дані, ринкові тенденції та прогнози попиту, система сприяє прийняттю стратегічних рішень щодо вибору постачальника та укладання контрактів, забезпечуючи економію коштів та підвищення загальної операційної ефективнності.

Управління запасами: APS SMART пропонує функціонал для оптимізації запасів, спрямований на досягнення балансу між попитом та пропозицією. Відображення актуальних даних щодо рівня запасів, термінів виконання поставок та статусу замовлень у режимі реального часу дозволяє мінімізувати ризики виникнення дефіциту, усунути надмірні запаси та підвищити рівень своєчасного виконання замовлень.

Планування попиту: система забезпечує точне прогнозування попиту завдяки інтеграції з даними про продажі, ринковими тенденціями та аналітичною інформацією про клієнтів. Це дає змогу організаціям завчасно передбачати коливання попиту, оптимізувати рівень запасів і узгоджувати закупівельну діяльність із стратегічними цілями підприємства.

Платформа для співпраці: APS SMART слугує платформою для співпраці всіх зацікавлених сторін закупівельного процесу. Система сприяє прозорому спілкуванню, ефективнному обміну документації та автоматизації робочого процесу, сприяючи прозорості, ефективнності та підзвітності на кожному етапі закупівель.

Переваги

Розширені відносини з постачальниками: APS SMART дозволяє організаціям будувати міцніші відносини з постачальниками завдяки прозорій комунікації, системному моніторингу ефективнності та тісній співпраці.

Економія: завдяки оптимізації процесів закупівель, мінімізації витрат на утримування запасів та укладанню найбільш вигідних контрактів APS SMART допомагає організаціям досягти значної економії фінансових ресурсів організації.

Покращена операційна ефективнність: спрощені робочі процеси, забезпечення моніторингу даних у режимі реального часу та прогнозована аналітика підвищують операційну ефективнність і гнучкість, що дозволяє організаціям оперативно реагувати на зміни ринкових умов.

Прийняття стратегічних рішень: надаючи практичну інформацію та прогнозну аналітику, APS SMART забезпечує можливість прийняття обґрунтованих управлінських рішень, що ґрунтуються на основі даних, сприяючи стратегічному зростанню та конкурентній перевазі.

Отже, система SRM APS SMART є комплексним рішенням для організацій, які прагнуть оптимізувати процеси закупівель, покращити відносини з постачальниками та досягти операційної досконалості.

Інтерфейс додатку «SRM APS SMART» зображено на рисунку 1.3.

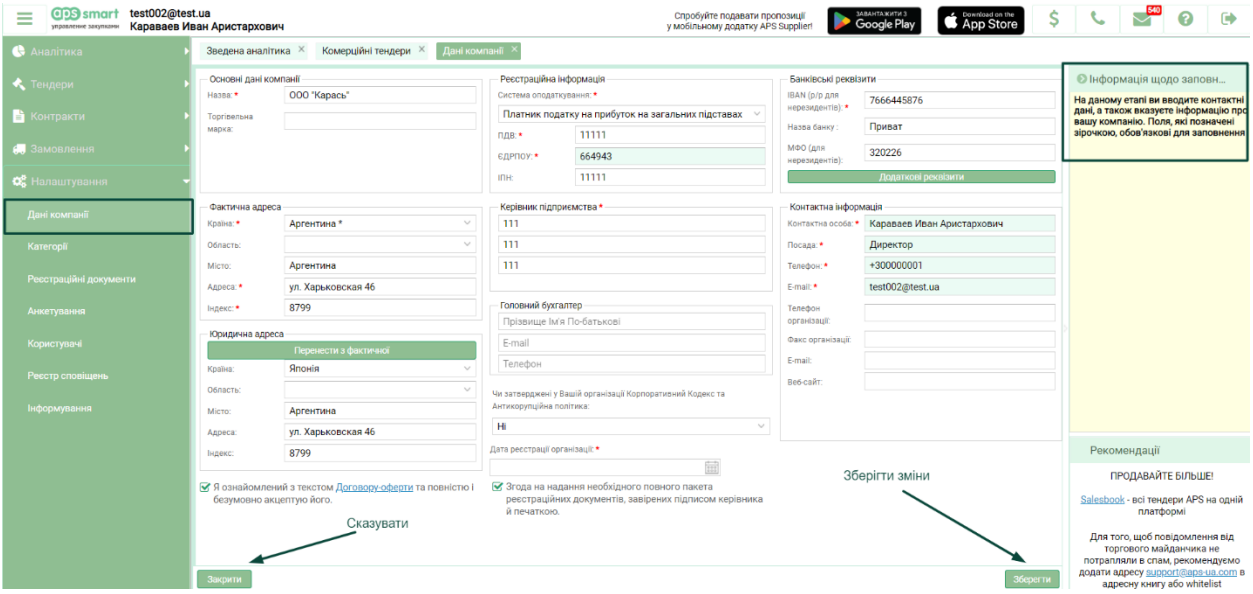


Рис. 1.3 Програмне забезпечення «SRM APS SMART»

E-Tender – це унікальне рішення для оптимізації закупівельної діяльності комерційних підприємств. Автоматизація тендерних процесів охоплює всі складові закупівель в організації: від етапу бюджетування та формування внутрішньої потреби у закупівлі до виконання договірних зобов'язань з контрагентом та електронним обміном юридично значимих документів. Платформа для комерційних закупівель E-Tender пропонує гнучкі налаштування для формування оптимального робочого середовища відповідно до індивідуальних потреб закупівельника. Закупникам надається доступ до широкого спектра процедур, різних моделей аукціонів з можливістю їхнього індивідуального налаштування [8].

Електронні тендери, також відомі як електронні тендери або онлайн-тендери, революціонізують процеси закупівельної діяльності завдяки використанню цифрових технологій. Використання спеціалізованих онлайн-платформ та електронних систем значно оптимізує електронну публікацію повідомлень про тендер, подання конкурсних пропозицій та їхньої подальшої оцінки.

Публікація повідомлень про тендери: електронні закупівельні системи дають змогу організаціям розміщувати інформацію про закупівлі в цифровому форматі, забезпечуючи висвітлення вимог до закупівель, термінів виконання, критеріїв кваліфікації та інструкцій щодо процедури подання заявок.

Подання тендерних пропозицій: постачальники здійснюють подання тендерних пропозицій в електронному форматі, використовуючи стандартизовані шаблони або форми, надані закупівельною організацією. Документація повинна містити інформацію щодо цінових показників, технічних специфікацій та інших супровідних матеріалів.

Оцінка тендерних пропозицій: закупівельні організації проводять оцінку отриманих тендерних пропозицій в електронному вигляді, керуючись заздалегідь встановленими критеріями з метою визначення найбільш прийнятного постачальника(-ів) для укладення договору.

Присудження контракту: успішні учасники торгів отримують контракти в електронному форматі. Умови договорів передаються через платформу електронних закупівель та засвідчуються за допомогою електронних цифрових підписів.

Переваги електронних тендерів:

Ефективнність: впроваджені електронні тендери спрощують процес закупівель, знижуючи адміністративне навантаження, використання паперу та час обробки.

Прозорість: сприяє дотриманню принципів справедливості, надаючи рівний доступ до тендерних можливостей і забезпечуючи об’єктивну оцінку пропозицій на основі заздалегідь визначених критеріїв.

Економія бюджетних коштів: впроваджені електронні торги зменшують витрати на проведення закупівель, усуваючи використання паперової документації, поштові витрати та ручну обробку.

Доступність: використання електронних систем забезпечує можливість участі в торгах постачальників із різних регіонів, що сприяє посиленню конкурентного середовища та розширенню кола потенційних постачальників.

Аудиторський слід: формування електронних записів створюють комплексний контрольний слід, документуючи всі етапи закупівельного процесу.

Цифровий розрив: недостатній рівень доступу до технологій інфраструктури та низький рівень цифрової грамотності можуть створювати проблеми для деяких постачальників.

Проблеми безпеки: платформи для проведення електронних тендерів потребують впровадження ефективнних механізмів захисту інформації з метою забезпечення конфіденційності даних.

Відповідність правовим і нормативним вимогам. Організації, що здійснюють закупівлі, повинні забезпечити дотримання чинного законодавства, нормативних актів і внутрішні політики у сфері закупівельної діяльності.

Сумісність системи. Проблеми сумісності між різними платформами або застарілими системами можуть перешкоджати сумісності й обміну даними.

Попри зазначені труднощі, електронні торги забезпечують значні переваги в ефективнності, прозорості та економічній ефективнності, що робить його все більш популярним для процесів закупівель у різних галузях і секторах.

Інтерфейс додатку «E-Tender» представлено на рисунку 1.4.

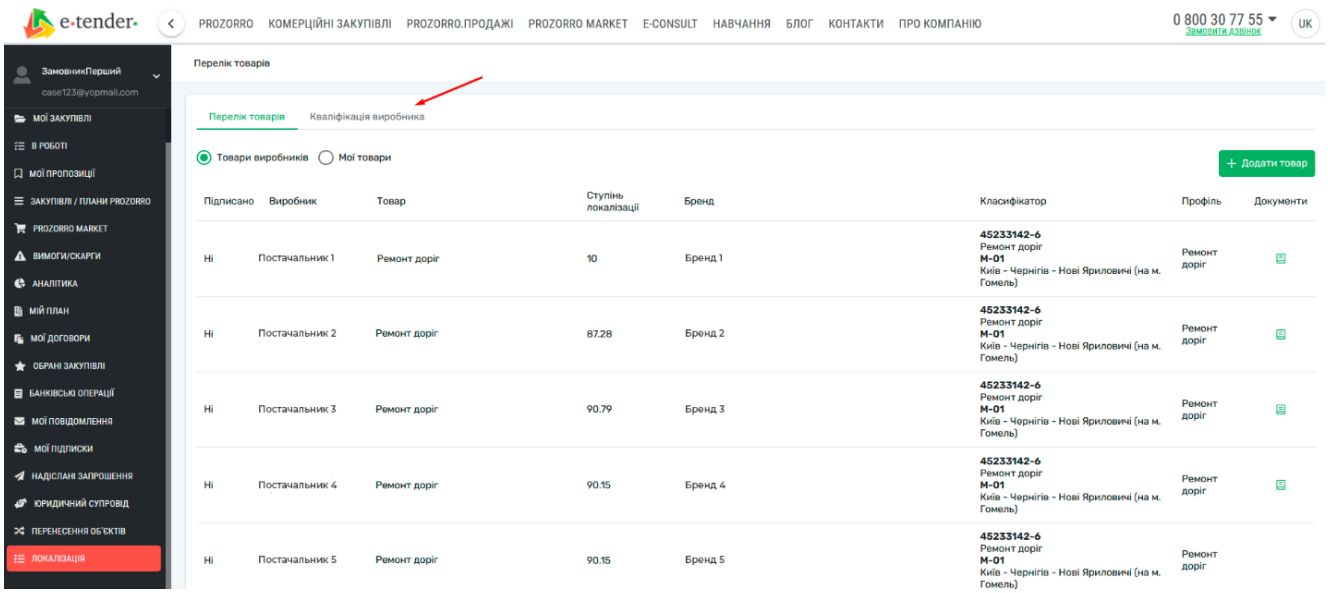


Рис. 1.4 Програмне забезпечення “E-Tender”

Компанія E-Tender активно розробляє власну унікальну систему для комерційних замовників. Постачальники мають можливість обирати закупівлі як приватних, так і державних підприємств.

На практиці клієнти цього майданчика часто поєднують роль замовників у системі комерційних закупівель із роллю учасників у тендерних процедурах системи «Prozorro». Компанія дбає про удосконалення не лише технічних якостей цієї платформи, а й реалізує персональні вимоги окремих клієнтів. Це дозволяє користувачам застосовувати персоналізовані налаштування систем, що значно підвищує ефективнність їх закупівель та продажів. Важливим у роботі є захист інформації компаній, цьому підтвердження є Атестат Комплексної системи захисту інформації [8].

З 2019 року компанія E-Tender є частиною групи E-Tech, до складу якої також входять e-Docs та Health24. E-Docs - система автоматизації документообігу тих бізнес-процесів компаній. Health24 - сучасна медична інформаційна платформа для лікарів та пацієнтів. Завдяки цьому клієнти отримують доступ до комплексних електронних рішень, спрямованих на ефективнний розвиток бізнесу.

У воєнний час, а також у часи пандемії, послуги E-Тендер приносять особливу користь клієнтам, адже забезпечують безперебійність процесів у роботі закупівельників публічної та приватної сфери. Високий рівень автоматизації всіх етапів закупівельних процедур, що здійснюються в електронному форматі, надає користувачам можливість безперешкодного доступу до системи у режимі 24/7 з будь-якої точки світу [8].

Зведені результати аналізу характеристик розглянутих додатків та розробленого web-застосунку наведено у таблиці 1.1.

Таблиця 1.1

Зведені результати аналізу характеристик додатків та розробленого web-застосунку

Критерій	**E-Tender**	**APS SMART**	**TenderAll** (розроблений web-застосунок)
Основна мета	Майданчик для участі у тендерах	Автоматизація закупівель та управління постачальниками	Рейтингове оцінювання постачальників
Автоматизований рейтинг постачальників	Відсутній	Частково реалізований	Повністю реалізований (на основі критеріїв)
Механізм формування рейтингу	Відсутній	Частково реалізований	Автоматичний після введення показників і коефіцієнтів
Робота з показниками оцінки	Відсутній	Корпоративні	Додавання, перегляд, видалення показників
Введення коефіцієнтів	Відсутній	Частково	Реалізовано через форму введення
Формування результату (топ постачальників)	Відсутній	Частково	Автоматичне формування рейтингу учасників

1.5 Постановка завдання

При постановці задачі основна увага приділяється визначенню та вирішенню основних завдань і завдань, пов’язаних із розробкою та впровадженням програмного рішення.

Основна проблема полягає в неефективності та відсутності прозорості поточного процесу тендерних закупівель. Це охоплює такі проблеми, як складність об’єктивної оцінки та порівняння ефективності роботи постачальника,

трудомісткі ручні процедури та відсутність прийняття рішень на основі даних. Крім того, відсутність прозорості підриває підзвітність і довіру як всередині, так і ззовні.

Щоб подолати ці проблеми, запропоноване програмне рішення спрямоване на розробку надійної рейтингової системи для для рейтингової оцінки постачальників, таким чином оптимізуючи процес тендерних закупівель. Це передбачає розробку зручного інтерфейсу для спеціалістів із закупівель для введення, відстеження та аналізу даних про продуктивність постачальника. Алгоритми та методології будуть розроблені для об’єктивної оцінки та ранжирування постачальників на основі попередньо визначених критеріїв, таких як ціноутворення та ефективнність доставки. Інтеграція з існуючими системами управління закупівлями та базами даних забезпечить безперебійний обмін даними, а можливості аналітики та звітності в реальному часі полегшать прийняття рішень на основі даних.

Прозорість і підзвітність буде покращено завдяки таким функціям, як журнал аудиту та контроль доступу користувачів. Для забезпечення надійності, точності та безпеки програмного рішення буде проведено ретельне тестування та перевірку. Крім того, буде надано комплексне навчання та підтримка для сприяння адаптації та ефективнного використання.

Вирішуючи ці цілі, програмне забезпечення для рейтингової системи постачальників має на меті оптимізувати процес тендерних закупівель, покращити результати відбору постачальників і підвищити прозорість і підзвітність у практиці закупівель.

2 ПРОЄКТУВАННЯ СИСТЕМИ РЕЙТИНГОВОЇ ОЦІНКИ ПОСТАЧАЛЬНИКІВ

2.1 Логічна модель даних у вигляді ER-діаграми

ERwin — це потужний інструмент моделювання даних, який широко використовується в корпоративних середовищах для проектування та керування базами даних. Його зручний графічний інтерфейс дозволяє користувачам створювати, візуалізувати та документувати моделі баз даних на основі підходу Entity-Relationship (ER) [9].

Щоб розпочати роботу з ERwin, вам необхідно встановити та налаштувати програмне забезпечення на вашому комп’ютері. Після встановлення ви можете створити нову модель даних. В інтерфейсі ERwin ви можете визначати сутності та їхні атрибути, що представляють об’єкти реального світу та їхні характеристики.

Однією з ключових особливостей ERwin є можливість встановлювати зв’язки між сутностями. Ці зв’язки представляють асоціації або залежності між різними об’єктами. Ви можете вказати характер зв’язків, включаючи обмеження участі та кількість.

Працюючи над моделлю даних, ви можете вдосконалити її, додавши додаткові деталі, такі як первинні ключі, зовнішні ключі, індекси та обмеження. ERwin надає інструменти перевірки та аналізу для забезпечення структурної цілісності вашої моделі та дотримання найкращих практик. Крім того, ви можете створити сценарії SQL або оператори мови визначення даних (DDL), щоб створити фізичну схему бази даних на основі вашої логічної моделі.

Документація є важливою частиною процесу моделювання, і ERwin пропонує можливості для створення повної документації для вашої моделі даних. Це включає діаграми сутності та зв’язку, словники даних і різні звіти. Належна документація допомагає зрозуміти та підтримувати базу даних з часом.

Логічна модель системи представлена на рисунку 2.1.

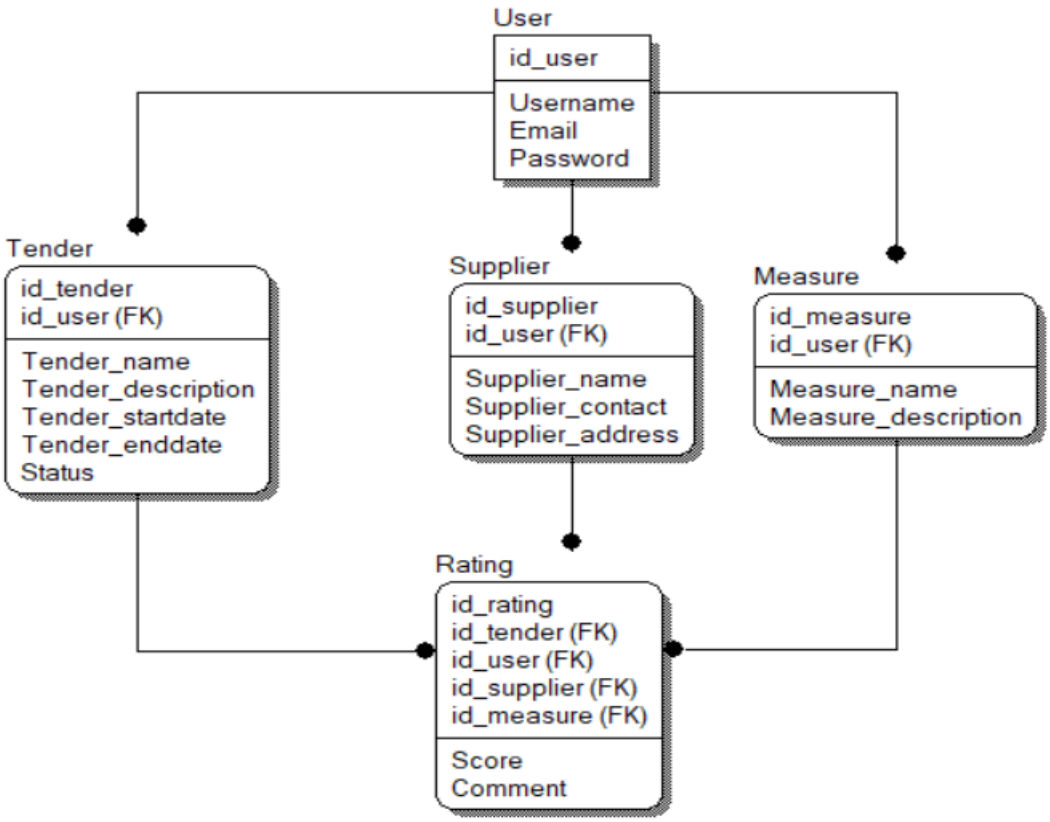


Рис. 2.1 ER-діаграма

Логічна модель складається з таких сутностей:

- «User» містить інформацію про користувача;
- «Tender» містить інформацію про тендер;
- «Supplier» містить інформацію про постачальників;
- «Measure» містить інформацію про показники оцінки;
- «Rating» містить інформацію про оцінку тендеру.

Сутність «User» має такі атрибути:

- id_user;
- password;
- username;
- email.

Сутність «Tender» має такі атрибути:

- id_tender;
- id_user;
- tender_name;
- tender_description;
- tender_startdate;
- tender_enddate;
- status.

Сутність «Supplier» має такі атрибути:

- id_supplier;
- id_user;
- supplier_name;
- supplier_contact;
- supplier_address.

Сутність «Measure» має такі атрибути:

- id_measure;
- id_user;
- measure_name;
- measure_description.

Сутність «Rating» має такі атрибути:

- id_rating;
- id_user;
- id_tender;
- id_supplier;
- id_measure;
- score;
- comment.

Сутність «User» пов’язана з сутностями «Tender», «Measure» та «Supplier». Також сутність «Rating» зв’язана з сутностями «Tender», «Measure» та «Supplier».

2.2 Діаграма класів та кооперацій

Діаграма класів визначає типи класів у системі та встановлює різноманітні статичні взаємозв'язки між ними. Вона описує атрибути класу, операції та обмеження, накладені на зв’язки між класами.

Зовнішній вигляд і інтерпретація діаграми класів суттєво залежать від перспективи (рівня абстракції): класи можуть представляти сутності в аналізі домену або елементи програмної системи в процесах проектування та впровадження [10].

В основі діаграми – класи та зв’язки між ними. Класи характеризуються атрибутами та операціями. Атрибути описують характеристики об'єктів класу. Більшість об’єктів у класі набувають індивідуальності через відмінності в їхніх атрибутах і зв’язках з іншими об’єктами.

Однак можливі об’єкти з ідентичними значеннями атрибутів і зв’язками. Імена атрибутів мають бути унікальними в межах класу, і вони можуть містити свій тип і значення за замовчуванням.

Операції — це функції або перетворення. Операція може мати параметри та значення, що повертаються.

Типи зв’язків включають асоціацію, агрегацію та успадкування.

Спеціальна діаграма класів була створена з використанням об’єктно-орієнтованого підходу для розробленого програмного забезпечення (рисунок 2.2).

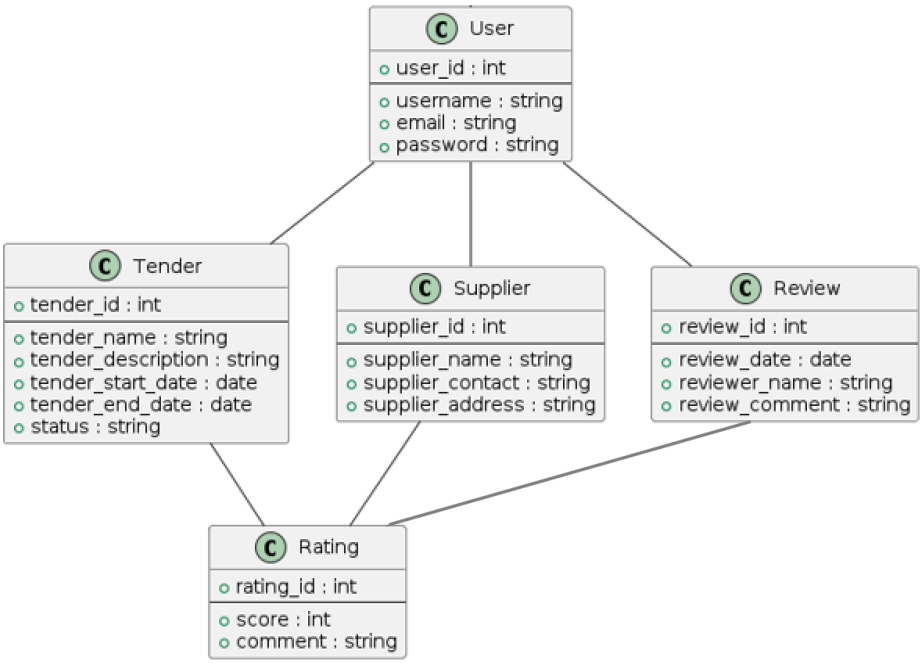


Рис. 2.2 Діаграма класів

Діаграма класів ілюструє фундаментальні сутності та їхні взаємозв'язки в межах «Програмного забезпечення для рейтингової оцінки постачальників», з метою оптимізації процесу тендерних закупівель.

Тендери є ключовими сутностями, що характеризуються такими атрибути, як ідентифікатор, ім'я, опис, дати початку та завершення, а також їх поточний статус. Кожен тендер може мати кілька пов'язаних рейтингів.

Постачальники представляють підприємства, що беруть участь у тендері. До їх атрибутів належить ідентифікатор, ім'я, контактна інформація та адреса. Постачальники можуть отримати кілька рейтингів сформованих на основі своєї продуктивності.

Рейтинги означають оцінки, надані користувачами щодо роботи постачальника на тендерах. Вони складаються з таких атрибутів, як ID, оцінка та коментарі. Рейтинги пов'язані як з тендерами, так і з постачальниками.

Користувачі системи охоплюють різних учасників, включаючи адміністраторів, покупців або рецензентів. До складу атрибутів користувача

включають ідентифікатор, ім’я користувача, адресу електронної пошти та пароль. Користувачі взаємодіють з тендерами, постачальниками та оглядами.

Відділи представляють організаційні одиниці та характеризуються такими атрибутами, як ID та назва. Вони пов’язані з користувачами, вказуючи на приналежність користувачів до відповідних відділів.

Відгуки містять оцінки користувачів щодо діяльності постачальників або загального тендерного процесу. Атрибути включають ідентифікатор, дату перевірки, ім’я рецензента та коментарі. Відгуки пов’язані з користувачами, відображаючи особу користувача, який пише відгук.

Дана діаграма забезпечує чіткий огляд сутностей системи, їхніх атрибутів і зв’язків між ними. Вона слугує основним джерелом для розуміння структури даних системи та взаємодії.

Діаграми кооперацій UML надають візуальне представлення процесу взаємодії об’єктів для виконання системних функцій. Вони є засобом для розуміння динаміки системи під час виконання. Об’єкти, зображені у вигляді прямокутників, взаємодіють шляхом обміну повідомленнями, представленими у вигляді посилань. Кожен об’єкт може взяти на себе різні ролі та відповідальність у рамках співпраці, сприяючи загальній функціональності системи.

Потік повідомлень між об'єктами ілюструє послідовність взаємодій. Ця візуалізація дозволяє розробникам зрозуміти зв’язки між складними об’єктами та моделі зв’язку, спрощуючи проектування та аналіз складних систем.

Діаграми кооперацій є особливо цінними на етапі проектування, допомагаючи виявити потенційні проблеми та оптимізувати продуктивність системи. Забезпечуючи чітке та стисле представлення об’єктної співпраці, ці діаграми покращують спілкування між зацікавленими сторонами та спрощують процес розробки.

Для даного програмного забезпечення було створено власні діаграми кооперацій (рисунок 2.3).

Рис. 2.3 Діаграма кооперацій

2.3 Діаграма пакетів

Діаграми пакетів UML служать для зображення організації та залежностей між різними пакетами в системі. Пакет діє як контейнер, що групує пов’язані елементи, такі як класи, інтерфейси, компоненти та інші пакети, щоб забезпечити організацію та керованість. На діаграмі пакетів пакети представлені у вигляді прямокутників із символом вкладки вгорі, що містить назву пакета. Залежності між пакетами показано стрілками.

Основна мета діаграми пакетів — надати огляд архітектури системи, демонструючи, як різні компоненти та підсистеми організовані в пакети. Це допомагає зрозуміти структуру високого рівня системи, визначити залежності між пакетами та полегшити модульне проектування та розробку.

Діаграми пакетів можна використовувати для візуалізації системних рівнів або рівнів, таких як рівень презентації, рівень бізнес-логіки та рівень доступу до даних у програмному забезпеченні. Вони також можуть представляти залежності між підсистемами, модулями або компонентами в межах більшої системи.

На додаток до відображення структурної організації пакетів, діаграми пакетів можуть містити додаткову інформацію, таку як зв’язки між пакетами (наприклад, узагальнення, реалізація або залежність), вміст пакетів (наприклад, класи або інтерфейси в кожному пакеті) і видимість пакетів (наприклад, загальнодоступних, приватних або захищених).

Діаграми пакетів UML забезпечують високорівневе уявлення про структуру системи, допомагаючи зацікавленим сторонам зрозуміти організацію та залежності між різними пакетами в системі. Вони є цінними інструментами для системних архітекторів, дизайнерів і розробників протягом життєвого циклу розробки програмного забезпечення.

Діаграма пакета (рисунок 2.5) для цього програмного забезпечення використовується для візуального представлення архітектури та організації системи. У ньому зображено різні пакети, що становлять програмне забезпечення, і показано зв’язки та залежності між ними.

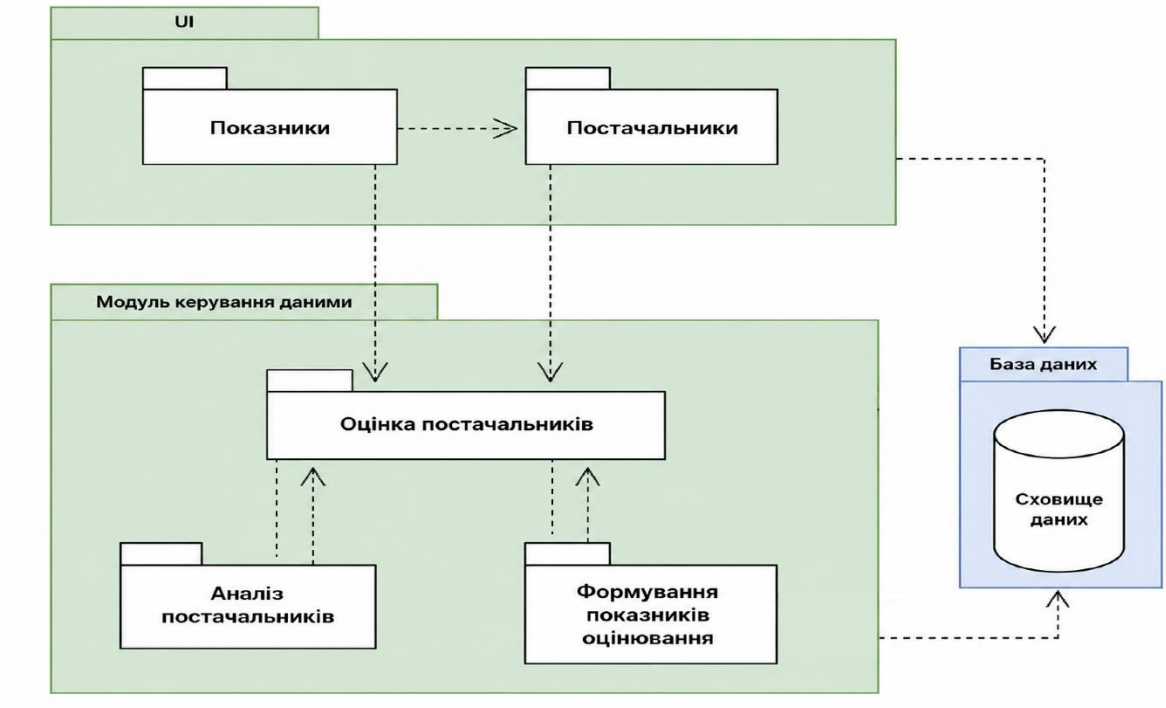


Рис. 2.5 Діаграма пакетів

Діаграма пакета ілюструє різні функціональні та структурні компоненти програмного забезпечення, згруповані в пакети. Кожен пакет представляє собою логічне групування пов’язаних класів, інтерфейсів та інших елементів, які сприяють певним функціям або модулям у системі.

На діаграмі пакетів пакети зображені у вигляді прямокутних коробок, усередині яких відображаються їхні назви. Стрілки або лінії використовуються для представлення зв’язків і залежностей між пакетами. Відносини можуть включати залежності, узагальнення або реалізації, демонструючи, як один пакет покладається на інший або розширює його.

2.4 Діаграма компонентів

Діаграма компонентів UML ілюструє організацію та залежності компонентів у системі на високому рівні. Він демонструє, як різні компоненти, такі як бібліотеки, виконувані файли та модулі, взаємодіють, щоб виконати системні вимоги або забезпечити певні функції.

До основних елементів діаграми компонентів належать:

- компоненти:

  це фундаментальні блоки системи, що характеризують собою модульні одиниці з чітко визначеними інтерфейсами. Вони забезпечують інкапсуляцію функціональності та можуть охоплювати бібліотеки, виконувані файли, файли вихідного коду або інші модулі, які можна розгортати;

- інтерфейси:

  визначають контракти чи угоди, яких компоненти дотримуються під час взаємодії. Вони визначають перелік методів, операцій або послуг, які надаються компонентом, а також ті, які потрібні іншим;

- залежності:

вказують на відносини або зв’язки між компонентами. Залежності класифікуються, як залежності використання, коли один компонент використовує послуги, надані іншим, або залежності реалізації, коли компонент реалізує або реалізує інтерфейс, наданий іншим;

- з’єднувачі:

виконують роль каналів зв’язку або механізмів, за допомогою яких компоненти взаємодіють. Вони включають різні типи протоколів зв’язку, зокрема виклики методів, повідомлення або сигнали;

- порти:

це точки взаємодії між компонентами та їх середовищем. Вони визначають інтерфейси, через які компоненти надсилають і отримують повідомлення або дані;

- артефакти:

вони представляють фізичні файли або модулі, створені або розгорнуті під час процесу розробки. Вони можуть включати файли вихідного коду, бібліотеки, виконувані файли, файли конфігурації або будь-який інший матеріальний продукт процесу розробки.

Діаграми компонентів є цінними для розуміння архітектури системи та її модульної структури. Вони допомагають ідентифікувати повторно використовувані компоненти, визначати чіткі інтерфейси та керувати залежностями між різними частинами системи.

Крім того, вони сприяють спілкуванню між зацікавленими сторонами, надаючи візуальне представлення архітектури системи та взаємодії компонентів.

Діаграма компонентів для розробленої програми представлена на рисунку 2.6.

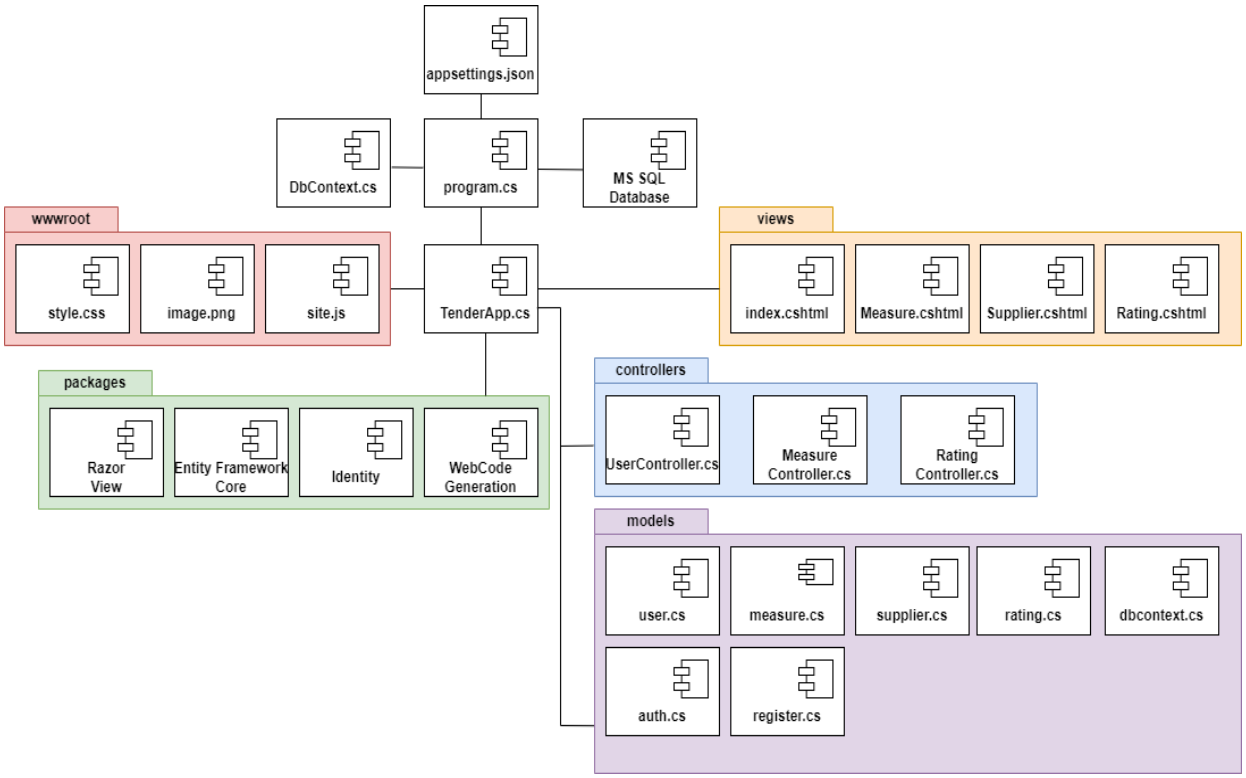


Рис. 2.6 Діаграма компонентів

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління базою даних

Реляційна база даних — це тип системи керування базами даних (СУБД), яка організовує та зберігає дані структурованим чином. Він дотримується реляційної моделі, де дані зберігаються в таблицях, що складаються з рядків і стовпців. Кожна таблиця представляє певну сутність, а стовпці представляють атрибути цієї сутності [13].

У реляційній базі даних таблиці є первинними структурами. Вони складаються з рядків (записів або кортежів) і стовпців (полів або атрибутів). Рядки представляють окремі екземпляри змодельованої сутності, тоді як стовпці представляють конкретні характеристики сутності.

Зв’язки встановлюють зв’язки між таблицями, визначаючи, як дані в одній таблиці співвідносяться з даними в іншій. Ці зв’язки базуються на загальних атрибутах або ключах, спільних між таблицями.

Ключі використовуються для унікальної ідентифікації рядків у таблиці. Первинний ключ — це стовпець або комбінація стовпців, які унікально ідентифікують кожен запис. Зовнішні ключі встановлюють зв’язки між таблицями, посилаючись на первинний ключ іншої таблиці.

Нормалізація - це процес організації таблиць для усунення надмірності та підвищення цілісності даних. Це передбачає ефективнне структурування даних, зменшення дублювання та забезпечення того, що інформація зберігається лише в одному місці.

Структурована мова запитів (SQL) — це стандартизована мова для взаємодії з реляційними базами даних [15]. Це дозволяє користувачам запитувати, змінювати та керувати базою даних. SQL підтримує пошук даних, вставку, оновлення, видалення та визначення схеми.

Реляційні бази даних пропонують такі переваги, як цілісність даних, гнучкість, масштабованість, підтримка складних операцій і дотримання властивостей ACID (атомність, узгодженість, ізоляція, довговічність). Вони широко використовуються в різних галузях завдяки здатності обробляти великі обсяги даних, забезпечувати незалежність даних і підтримувати надійну обробку транзакцій.

Реляційні бази даних пропонують структурований та ефективнний підхід до зберігання та керування даними. Вони використовують таблиці, зв’язки, ключі та SQL для забезпечення цілісності даних, гнучкості, масштабованості та надійної обробки транзакцій. Завдяки своїй надійності та здатності ефективнно обробляти структуровані дані, ці бази даних широко застосовуються в різних програмах і галузях.

Вибір правильної системи керування базами даних (СУБД) має вирішальне значення для розробки та функціонування системи. Вибір має відповідати вимогам і цілям програмного забезпечення.

Процес відбору розпочався з оцінювання конкретних вимог до програмного забезпечення оптимізації процесу тендерних закупівель, враховуючи такі фактори, як обсяг даних, робоче навантаження користувача, типи даних, потреби безпеки та масштабованість. Цей аналіз допоміг визначити бажані функції та можливості СУБД.

Згодом було досліджено та оцінено різні параметри СУБД, які підходять для програмного забезпечення оптимізації процесу тендерних закупівель, включаючи MySQL, PostgreSQL, Oracle, MongoDB або SQL Server. Були враховані такі фактори, як функції, продуктивність, масштабованість, безпека, підтримка спільноти та вартість ліцензування. Було також забезпечено, що обрана СУБД підтримує необхідні моделі даних і мови запитів.

Потреби в інтеграції програмного забезпечення для відстеження стану здоров’я та підтримки фізичної активності було оцінено, щоб визначити, чи може СУБД бездоганно інтегруватися з іншими використовуваними технологіями чи

фреймворками. Сумісність і легкість інтеграції є критично важливими для плавного процесу розробки.

Оцінено тести продуктивності опцій СУБД. Такі функції, як індексування, кешування та оптимізація запитів, були визначені для підвищення продуктивності. Варіанти масштабованості, пропоновані СУБД, розглядалися для адаптації до майбутнього зростання та збільшення навантаження на користувачів.

Безпека даних має вирішальне значення для програмного забезпечення оптимізації процесу тендерних закупівель, яке обробляє конфіденційну інформацію користувачів. Було оцінено функції безпеки кожної СУБД, включаючи шифрування, контроль доступу, можливості аудиту та дотримання правил захисту даних.

Було враховано простоту розробки, адміністрування та обслуговування СУБД. Було визначено надійну екосистему, документацію та підтримку спільноти, щоб допомогти у вирішенні проблеми та надати своєчасну допомогу. Було оцінено наявність інструментів і фреймворків, що оптимізують операції з базами даних і міграції.

Були розраховані витрати на ліцензування, плата за підтримку та потенційні витрати, пов’язані з інфраструктурою, пов’язані з параметрами СУБД.

На підставі оцінки цих факторів було прийнято рішення про вибір СУБД для розробки програмного забезпечення відстеження здоров’я та підтримки фізичної активності: MS SQL Server.

MS SQL розроблено для роботи з великими обсягами даних і забезпечує високу масштабованість, обслуговуючи тисячі користувачів, що одночасно працюють, і масштабується відповідно до змінних вимог компанії. Це високопродуктивна база даних, оптимізована для швидкого доступу та пошуку даних. Він підтримує розширені методи індексування та оптимізації запитів, що робить його ідеальним для програм, які потребують швидкої обробки даних.

MS SQL має розширені функції безпеки для захисту даних від несанкціонованого доступу, включаючи захист на рівні рядків і завжди зашифровані дані. Крім того, MS SQL розроблено для бездоганної інтеграції з

іншими продуктами та технологіями Microsoft, полегшуючи використання з іншими компонентами програмного забезпечення в стеку технологій Microsoft. Він пропонує такі функції, як дзеркальне відображення бази даних, автоматичне перемикання після відмови, а також можливості резервного копіювання та відновлення для забезпечення постійної доступності даних у разі потреби.

Після створення логічної моделі була побудована фізична структура реляційної бази даних (РБД) для планованої системи. Як згадувалося раніше, MS SQL Server був обраний як система управління базами даних (СУБД) для системи.

3.2 Розробка бази даних системи рейтингової оцінки постачальників

Під час проєктування таблиць у фізичній моделі основою слугували сутності з логічної моделі. Атрибути цих сутностей були використані для розробки структури таблиці.

Отриману схему бази даних зображено на рисунку 3.1.

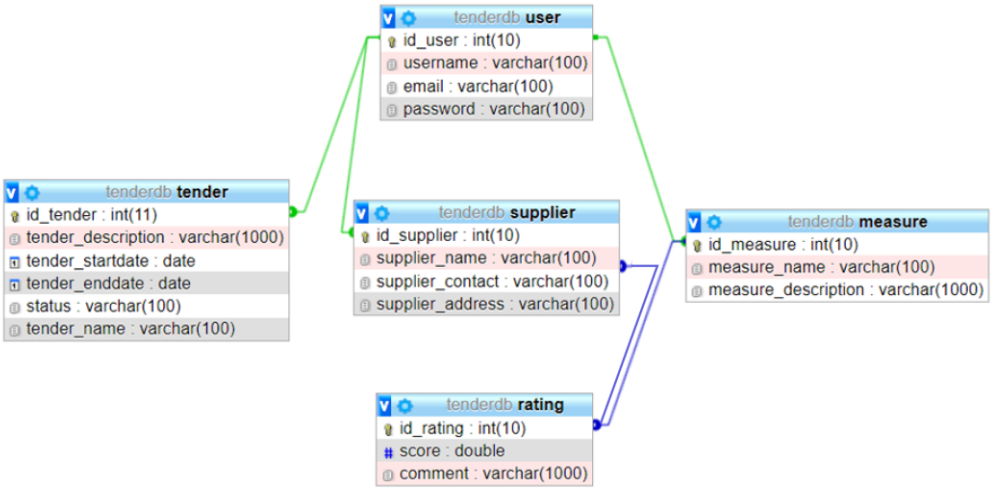


Рис. 3.1 База даних системи

Розглянемо детальніше створення кожної з таблиць і властивості визначені для них. Опишемо назву поля та тип даних усіх атрибутів.

Таблиця «User», зображена на рисунку 3.2, зберігає в БД користувачів, має такі поля:

- id – int(10). Ключове поле таблиці «User»;
- username – varchar(100). Поле, в якому зберігається ім’я користувача;
- email – varchar(100). Поле, в якому зберігається електронна пошта користувача;
- password – varchar(100). Поле, в якому зберігається зашифрований пароль користувача;

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id_user	int(10)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	username	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	3	email	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	4	password	varchar(100)	utf8_general_ci		No	None			Change Drop More

Рис. 3.2 Створена таблиця «User»

Таблиця «Tender», зображена на рисунку 3.3, зберігає в БД тендери, має такі поля:

- id_tender – int(11). Ключове поле таблиці «Tender»;
- tender_name – varchar(100). Поле, в якому зберігається назва тендеру;
- tender_description – varchar(100). Поле, в якому зберігається опис тендеру;
- tender_startdate – date. Поле, в якому зберігається дата початку тендеру;

- tender_enddate -– date. Поле, в якому зберігається дата завершення тендеру;
- status – varchar(255). Поле, в якому зберігається поточний статус тендеру;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_tender	int(11)			No	None		AUTO_INCREMENT	Change Drop
2	tender_description	varchar(1000)	utf8_general_ci		No	None			Change Drop
3	tender_startdate	date			No	None			Change Drop
4	tender_enddate	date			No	None			Change Drop
5	status	varchar(100)	utf8_general_ci		No	None			Change Drop
6	tender_name	varchar(100)	utf8_general_ci		No	None			Change Drop

Рис. 3.3 Створена таблиця «Tender»

Таблиця «Supplier», зображена на рисунку 3.4, зберігає в БД записи постачальників води, має такі поля:

- id_supplier – int(11). Ключове поле таблиці «Supplier»;
- supplier_name – varchar(100). Поле, в якому зберігається назва постачальника;
- supplier_contact - varchar(100). Поле, в якому зберігається інформація контаків постачальника;
- supplier_address - varchar(100). Поле, в якому зберігається адреса постачальника.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_supplier	int(10)			No	None		AUTO_INCREMENT	Change Drop
2	supplier_name	varchar(100)	utf8_general_ci		No	None			Change Drop
3	supplier_contact	varchar(100)	utf8_general_ci		No	None			Change Drop
4	supplier_address	varchar(100)	utf8_general_ci		No	None			Change Drop

Рис. 3.4 Створена таблиця «Supplier»

Таблиця «Measure», зображена на рисунку 3.5, зберігає в БД записи показників оцінки тендерів, має такі поля:

- id_measure – int(11). Ключове поле таблиці «Measure»;
- measure_name – varchar(100). Поле, в якому зберігається назва показника;
- measure_description - varchar(1000). Поле, в якому зберігається опис показника.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_measure	int(10)			No	None		AUTO_INCREMENT	Change Dr
2	measure_name	varchar(100)	utf8_general_ci		No	None			Change Dr
3	measure_description	varchar(1000)	utf8_general_ci		No	None			Change Dr

Рис. 3.5 Створена таблиця «Measure»

Таблиця «Rating», зображена на рисунку 3.6, зберігає в БД записи рейтингу тендерних учасників, має такі поля:

- id_rating – int(10). Ключове поле таблиці «Rating»;
- score – double. Поле, в якому зберігається кількість балів;
- comment - varchar(1000). Поле, в якому зберігається додатковий опис до оцінки.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_rating	int(10)			No	None		AUTO_INCREMENT	Change Drop Mo
2	score	double			No	None			Change Drop Mo
3	comment	varchar(1000)	utf8_general_ci		No	None			Change Drop Mo

Рис. 3.6 Створена таблиця «Rating»

3.3 Вибір інструментальних засобів розробки

Дипломний проект спрямований на розробку web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель. Для реалізації програмного забезпечення було обрано такі технології: C#, ASP.NET Core MVC, Identity, EF Core, Razor Pages, HTML, CSS, JavaScript та Photoshop CS6. Ці інструменти дозволяють створити повноцінне програмне забезпечення із можливістю обробки даних, взаємодії з користувачем та формування результатів оцінювання постачальників.

Мова програмування C# є основною мовою програмування, пропонуючи універсальний і ефективнний синтаксис для побудови серверної логіки програмного забезпечення. Її використання дозволяє ефективнно організувати роботу з даними в межах системи рейтингової оцінки.

Фреймворк ASP.NET Core MVC використовується, як архітектурна основа для побудови структури web-застосунку. Він дозволяє розділити логіку обробки даних, інтерфейс користувача та керування запитами, що спрощує розробку та підтримку програмного забезпечення.

Identity пропонує можливості автентифікації та авторизації, забезпечуючи безпечний доступ до функцій і даних програми. EF Core служить платформою ORM (Object-Relational Mapping), спрощуючи взаємодію з базою даних і забезпечуючи повну інтеграцію з моделлю даних програми.

Razor Pages забезпечує спрощений підхід до створення web-сторінок, дозволяючи розробникам створювати динамічний вміст з мінімальними витратами. HTML, CSS і JavaScript необхідні для розробки та реалізації інтерфейсу користувача програми, забезпечуючи візуально привабливу та інтуїтивно зрозумілу взаємодію з користувачем.

А також, для розробки графічних елементів використовується Photoshop CS6, який дозволяє створювати візуальні компоненти інтерфейсу.

Використовуючи цей комплексний набір інструментів, маємо змогу створити складне та ефективнне програмне рішення, адаптоване до конкретних вимог рейтингової оцінки постачальників з метою оптимізації тендерних закупівель.

3.4 Архітектура програмного забезпечення

Існує кілька програмних архітектур, які зазвичай використовуються при проектуванні та розробці програм.

Рівнева архітектура організовує програму на логічні рівні, такі як рівні презентації, бізнес-логіки та рівня доступу до даних. Кожен рівень несе певну відповідальність і взаємодіє з сусідніми рівнями. Ця архітектура сприяє поділу завдань і модульності.

Клієнт-серверна архітектура включає дві окремі сутності: клієнт, який надсилає запити, і сервер, який обробляє ці запити та надає відповіді. Він зазвичай використовується у веб-додатках, де клієнтом є веб-браузер, а сервер розміщує логіку додатків і дані.

Архітектура мікросервісів розбиває програму на невеликі служби, які можна розгортати незалежно, кожна з яких відповідає за певну бізнес-функцію. Ці служби спілкуються за допомогою спрощених протоколів, таких як HTTP або черги обміну повідомленнями. Архітектура мікросервісів пропонує масштабованість, гнучкість і дозволяє командам працювати над різними сервісами незалежно.

Сервісно-орієнтована архітектура (SOA) схожа на мікросервіси, але сервіси зазвичай більші й обмінюються даними за допомогою стандартизованих протоколів, таких як SOAP або REST. SOA зосереджується на повторному використанні та сумісності між службами.

Model-View-Controller (MVC) розділяє програму на три взаємопов’язані компоненти: модель (дані), представлення (інтерфейс користувача) і контролер (бізнес-логіка). MVC сприяє відокремленню завдань і полегшує обслуговування та розширення програми.

Архітектура, керована подіями (EDA), наголошує на виробництві, виявленні, споживанні та реакції на події. Компоненти системи спілкуються через події, які запускають дії або процеси. EDA корисний для створення систем, які повинні реагувати на зміни в реальному часі.

Компонентна архітектура зосереджена на розкладанні системи на автономні програмні компоненти, які можна використовувати багаторазово. Ці компоненти інкапсулюють функціональність і можуть бути зібрані для створення складних систем. Компонентна архітектура сприяє повторному використанню коду та зручності обслуговування.

Шестикутна архітектура (порти та адаптери) відокремлює основну логіку програми від зовнішніх проблем, таких як бази даних, фреймворки інтерфейсу користувача та сторонні служби. Він використовує порти (інтерфейси) і адаптери (реалізації), щоб відокремити логіку ядра від зовнішніх залежностей, роблячи систему більш тестованою та гнучкою.

Domain-Driven Design (DDD) зосереджується на моделі домену та співпраці між експертами домену та розробниками програмного забезпечення для створення насиченої, виразної моделі проблемного домену. DDD наголошує на розбитті складних доменів на менші, більш керовані компоненти та узгодженні дизайну програмного забезпечення з концепціями домену.

Кожна архітектура програмного забезпечення має свої сильні та слабкі сторони, і вибір архітектури залежить від таких факторів, як вимоги до проекту, потреби в масштабованості, досвід команди та характер програми, що розробляється.

Для дипломного проекту, зосередженого на розробці програмного забезпечення для рейтингової системи підприємства-постачальника для оптимізації процесу тендерних закупівель, вибір відповідної архітектури програмного забезпечення має вирішальне значення для забезпечення масштабованості, зручності обслуговування та ефективнності. Враховуючи складність системи та необхідність повної інтеграції різних компонентів, модель багаторівневої архітектури буде добре підходящою.

Модель багаторівневої архітектури організовує програму на окремі рівні, кожен з яких відповідає за певну функціональність. Такий підхід сприяє модульності, полегшуючи керування та обслуговування системи в міру її зростання.

Рівень презентації:

- відповідає за обробку взаємодії з користувачем і представлення інформації користувачам;
- реалізує інтерфейс користувача за допомогою таких технологій, як Razor Pages, HTML, CSS і JavaScript;
- використовує ASP.NET Core MVC для маршрутизації запитів і обробки введених користувачем даних;
- інтегрується з Identity для автентифікації та авторизації користувачів.

Рівень бізнес-логіки:

- містить основну бізнес-логіку та правила програми;
- впроваджує рейтингові алгоритми, процеси закупівель і критерії оцінки постачальників;
- використовує C# для реалізації бізнес-логіки;
- використовує EF Core для взаємодії з базою даних і доступу до об’єктів даних.

Рівень доступу до даних:

- керується взаємодією з базою даних і збереженням даних;
- використовує EF Core для зіставлення сутностей бази даних з об’єктами домену та виконання операцій з базою даних;
- забезпечує цілісність і безпеку даних шляхом належної перевірки та керування транзакціями.

Рівень інфраструктури:

- надає послуги інфраструктури та утиліти, що використовуються на різних рівнях;
- включає журналювання, кешування, обробку помилок і керування конфігурацією;
- реалізує наскрізні проблеми, такі як журналювання та обробка винятків.

Рівень зовнішніх служб (необов’язково):

- інтерфейси із зовнішніми службами або API для додаткових функцій;
- інтегрується зі сторонніми службами для збагачення даних, перевірки або інтеграції із зовнішніми системами.

Прийнявши модель багаторівневої архітектури, ми можемо досягти поділу проблем, полегшуючи розуміння, підтримку та розширення системи з часом. Цей підхід також сприяє масштабуванню, дозволяючи кожному шару масштабуватися незалежно залежно від конкретних вимог програми.

Крім того, це дає змогу краще тестувати та налагоджувати, оскільки кожен рівень можна тестувати окремо, що призводить до покращення якості та надійності програмного забезпечення.

Нижче буде продемонстровано багатошарова архітектура даного програмного забезпечення (рисунок 3.7).

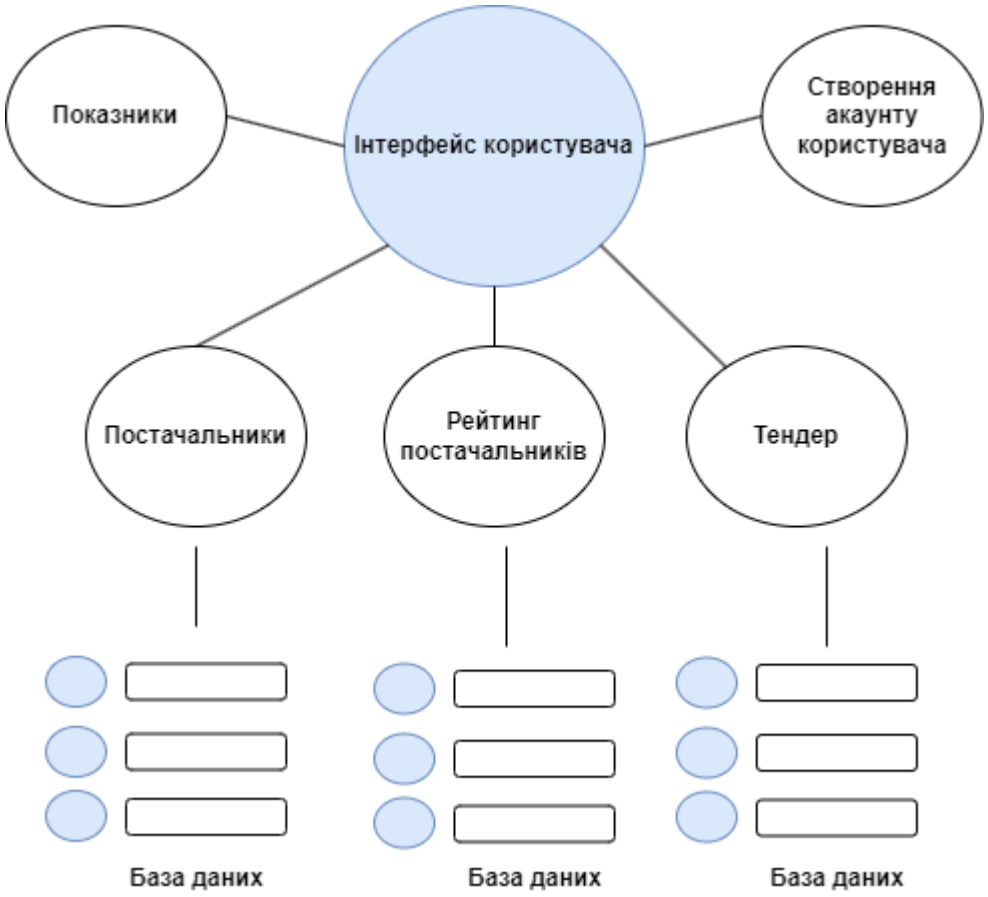


Рис. 3.7 Багатошарова архітектура програмного забезпечення

З рисунка вище ми бачимо, що наша система побудована на багаторівневій архітектурі, що складається з презентаційного рівня, бізнес-рівня та рівня бази даних.

Інформація в системі поступово переходить від рівня до наступного рівня по черзі.

Перший рівень - це інтерфейс користувача, тобто екран комп'ютера або телефона.

Другий рівень - це основний функціонал програми, з яким працює користувач. На цьому рівні він вносить свої дані, може використовувати тендери та формувати рейтинги.

І третій рівень, безпосередньо, база даних, в який зберігається вся інформація користувача.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ РЕЙТИНГОВОЇ ОЦІНКИ ПОСТАЧАЛЬНИКІВ

4.1 Тестування системи

Після запуску системи відкривається головна сторінка програмного забезпечення (рисунок 4.1, рисунок 4.2).

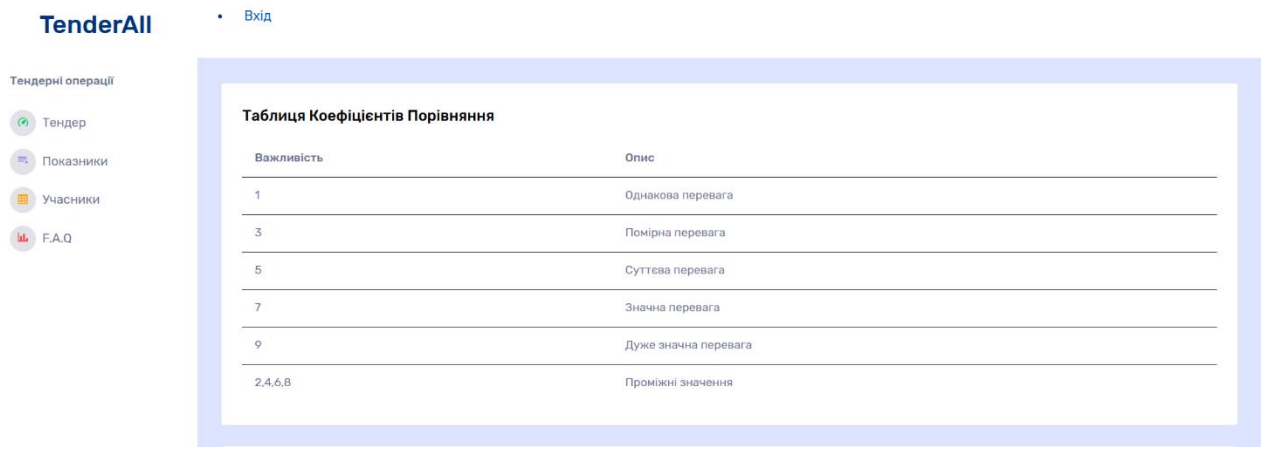


Рис. 4.1 Головна сторінка

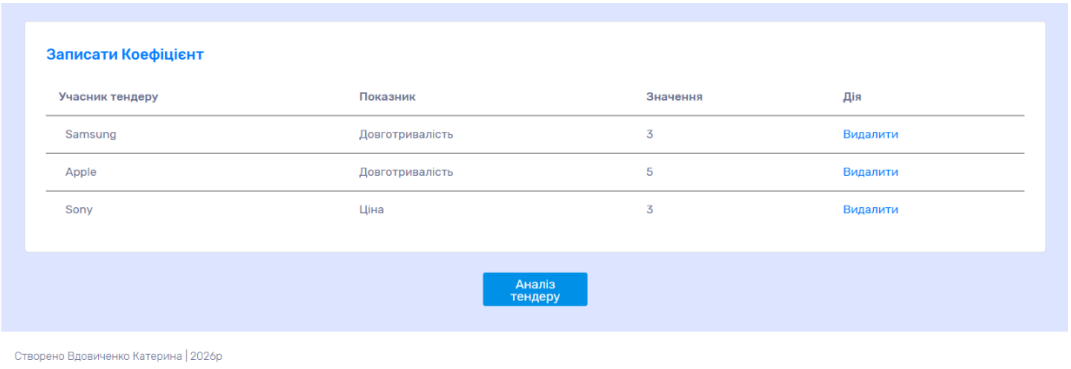


Рис. 4.2 Головна сторінка

На головній сторінці користувач одразу бачить сторінку з тендерами. На ній відображається таблиця з рівнями оцінки показників, а також безпосередньо таблицю коефіцієнтів учасників тендеру.

Натиснувши кнопку «Аналіз тендера» користувач переходить на сторінку з результатами порівняння учасників тендеру.

На рисунку 4.3 представлена форма авторизації в системі, яку необхідно заповнити при першому вході в систему.

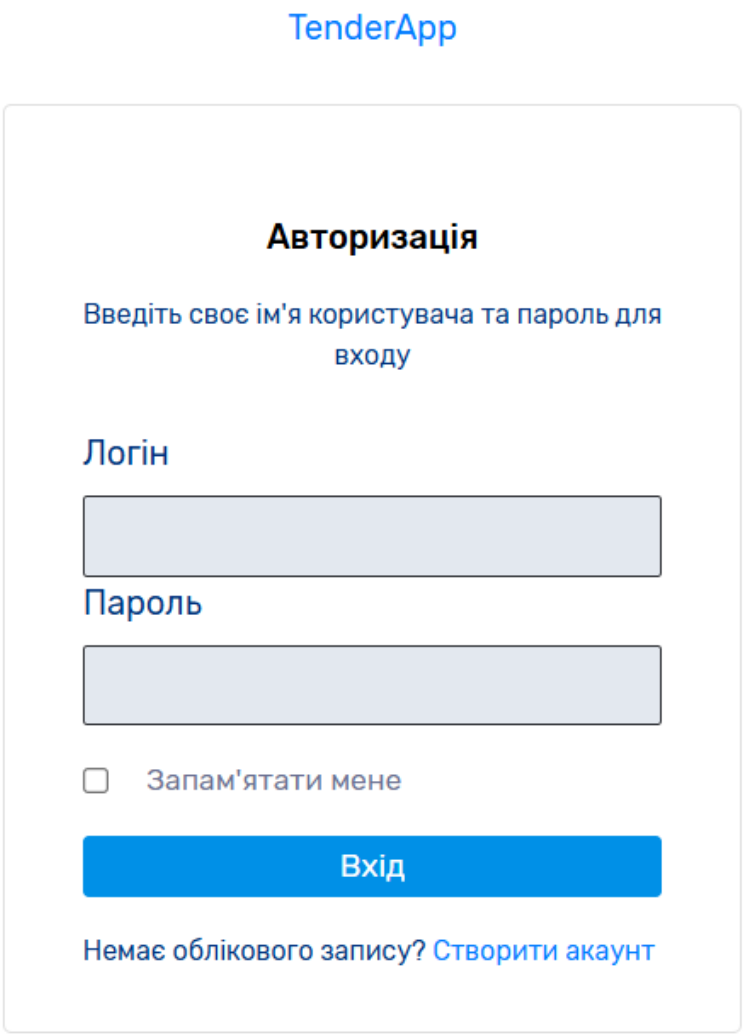


Рис. 4.3 Форма авторизації

Якщо користувач не має облікового запису, то наступним кроком є реєстрації в системі - створення профілю користувача та заповнення його особистими даними. Ця форма представлена на рисунку 4.4.

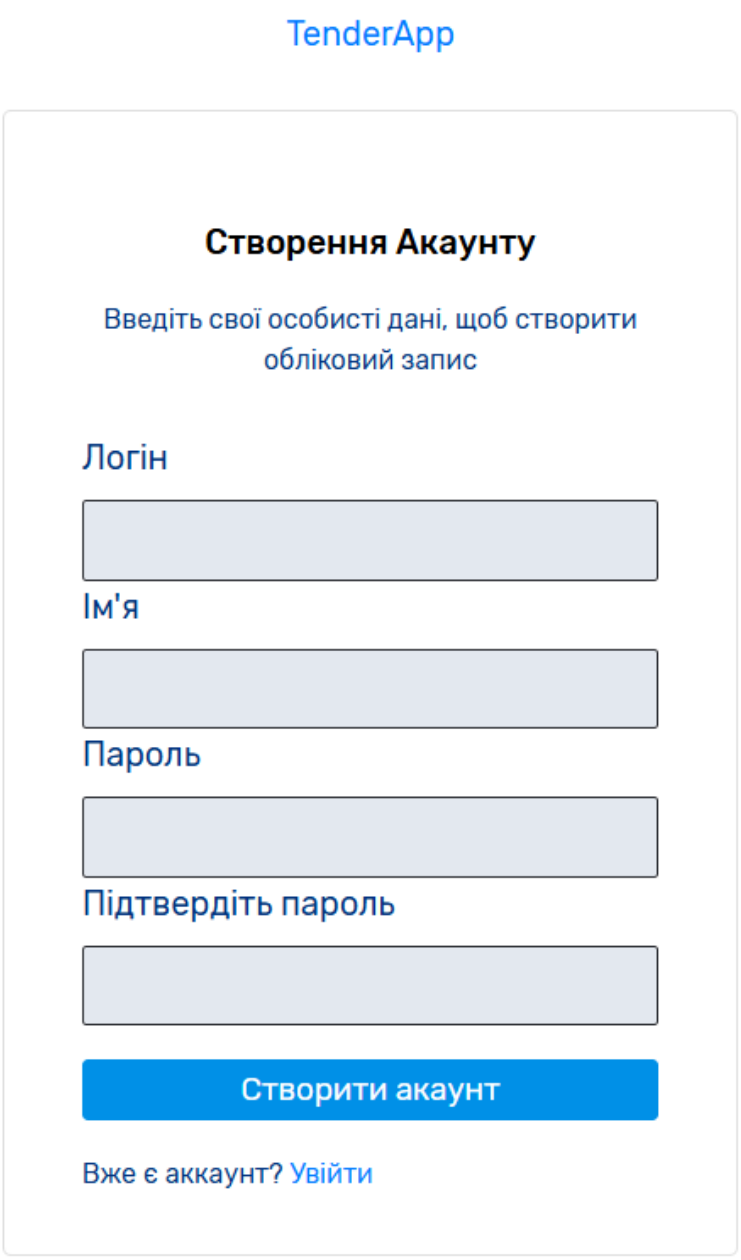


Рис. 4.4 Форма реєстрації

При вході систему ввівши неправильно дані буде виводитись помилка входу та показуватиметься повідомленню користувачеві (рисунок 4.5).

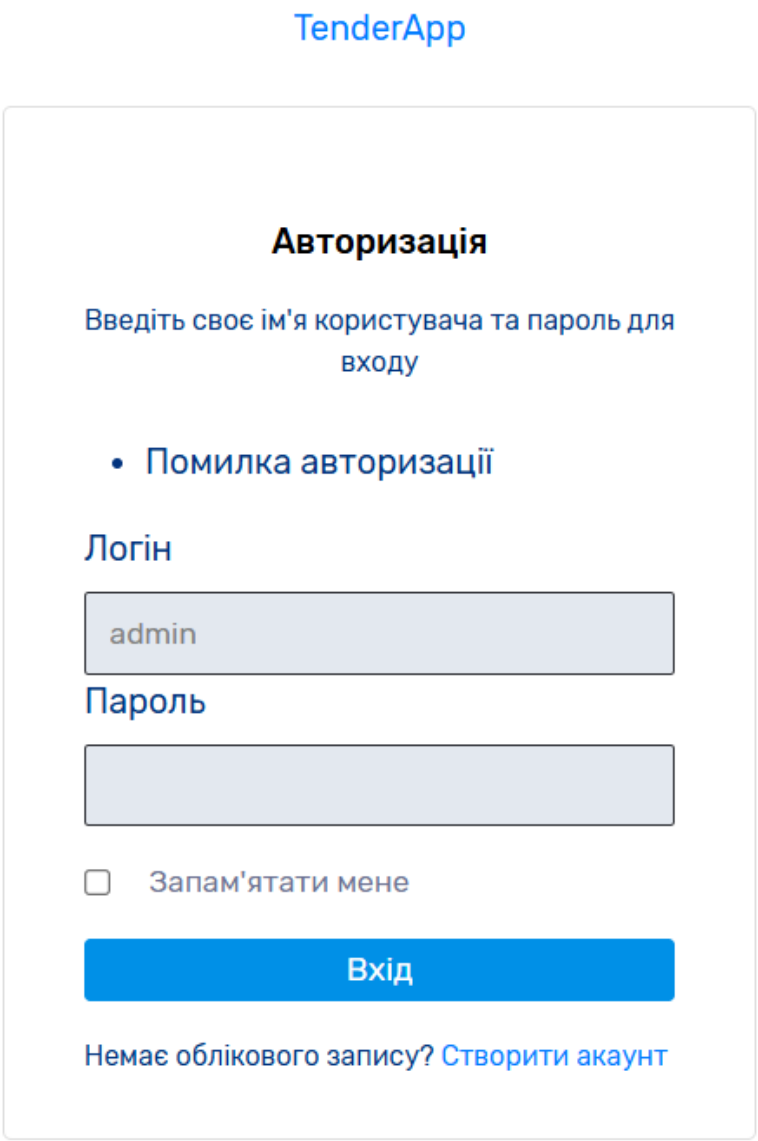


Рис. 4.5 Помилка при спробі авторизації

Після успішного входу в систему можна перейти на вкладку «Показники» та переглянути введені показники (рисунок 4.6).

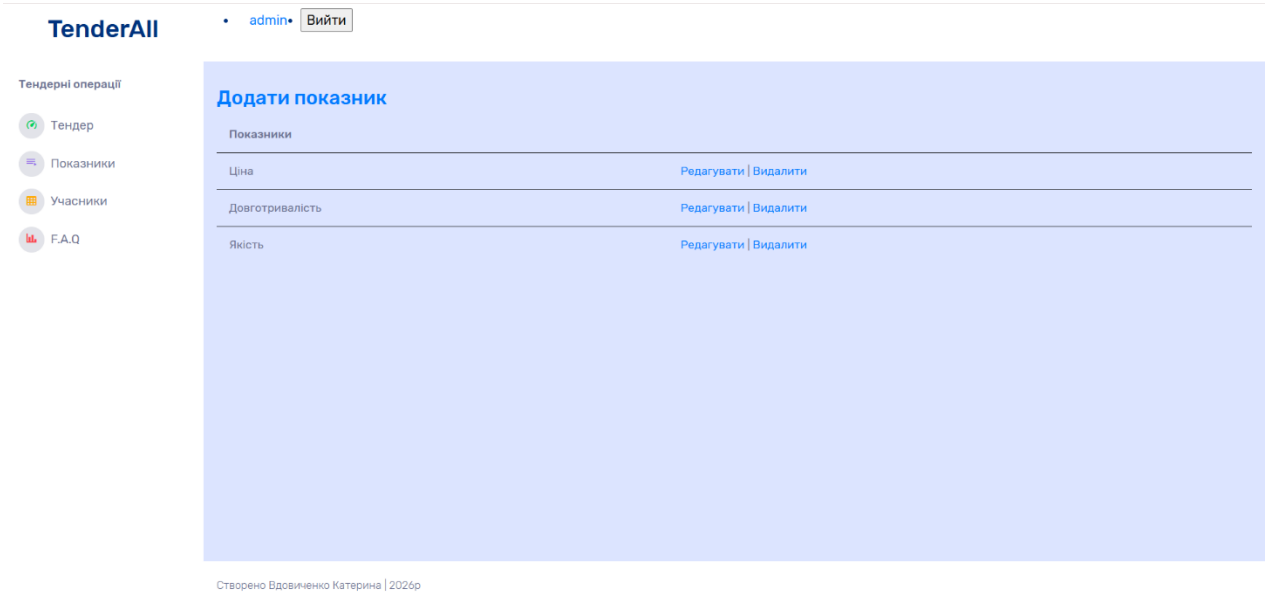


Рис. 4.6 Сторінка показників

Можна додати новий показник, для цього натискаємо на кнопку «Додати показник», після чого відкривається відповідне вікно з формами заповнення даних, а також показник можна редагувати в подальшій роботі (рисунок 4.7).

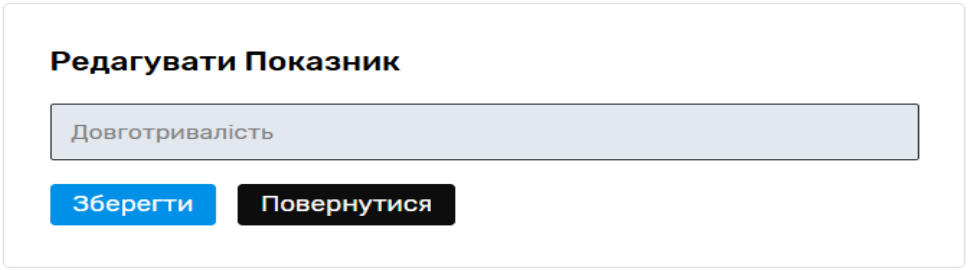


Рис. 4.7 Форма редагування показників

Крім цього, розроблена система містить можливість видалення показників, це вікно представлено на рисунку 4.8.

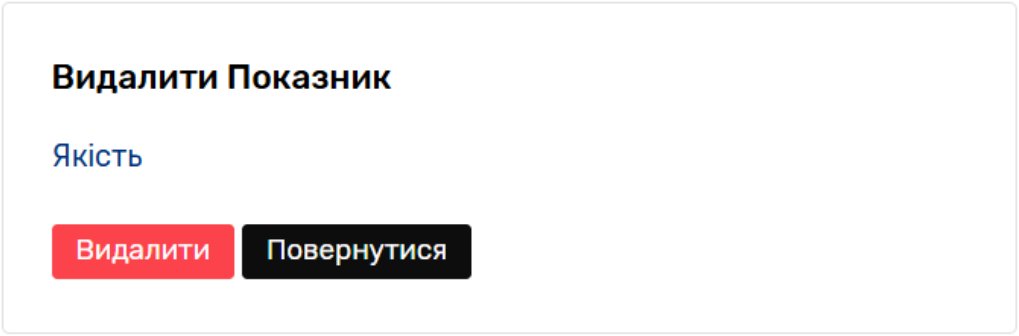


Рис. 4.8 Підтвердження видалення показника

Наступною сторінкою в меню є «Учасники», ця сторінка представлена на рисунку 4.9.

Рис. 4.9 Сторінка учасників

Система містить можливість додавання учасників. При натисканні на кнопку «Додати учасника тендеру» відкривається відповідне вікно з формами заповнення даних (рисунок 4.10).

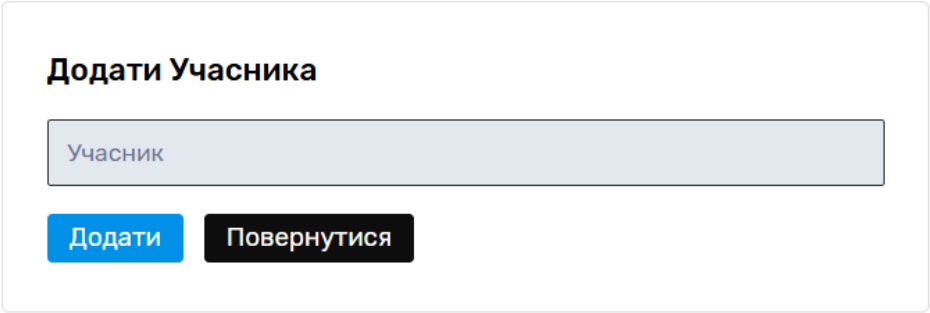


Рис. 4.10 Форма додавання учасників

Учасників також можна редагувати та видаляти (рисунок 4.11, рисунок 4.12)

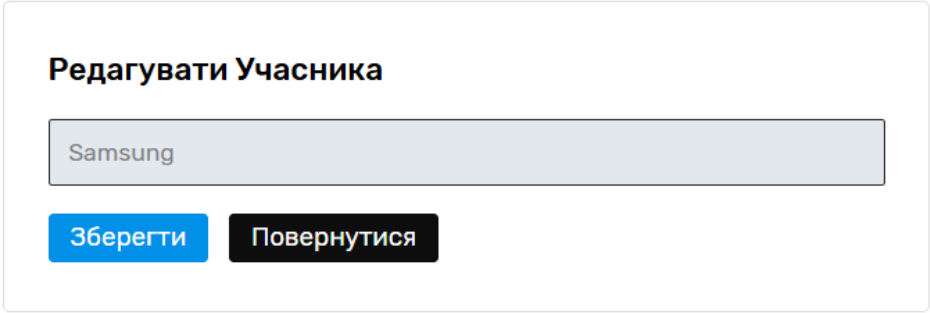


Рис. 4.11 Форма редагування учасників

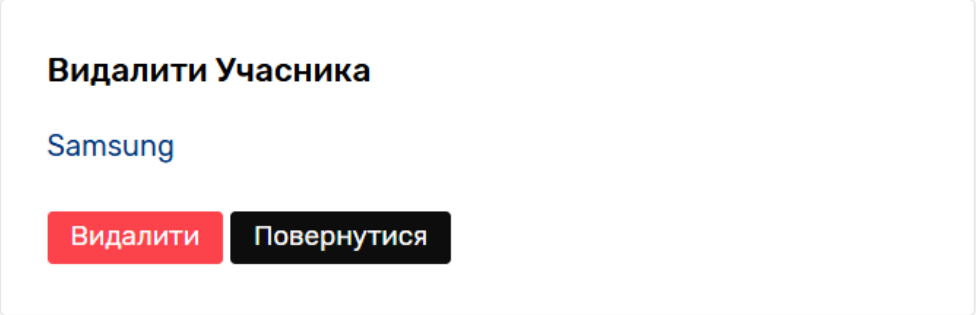


Рис. 4.12 Форма видалення учасників

Програмне забезпечення також має сторінку з відповідями питання «F.A.Q». На ній представлені відповіді на можливі питання користувача до системи, а також контактні дані власника (рисунок 4.13).

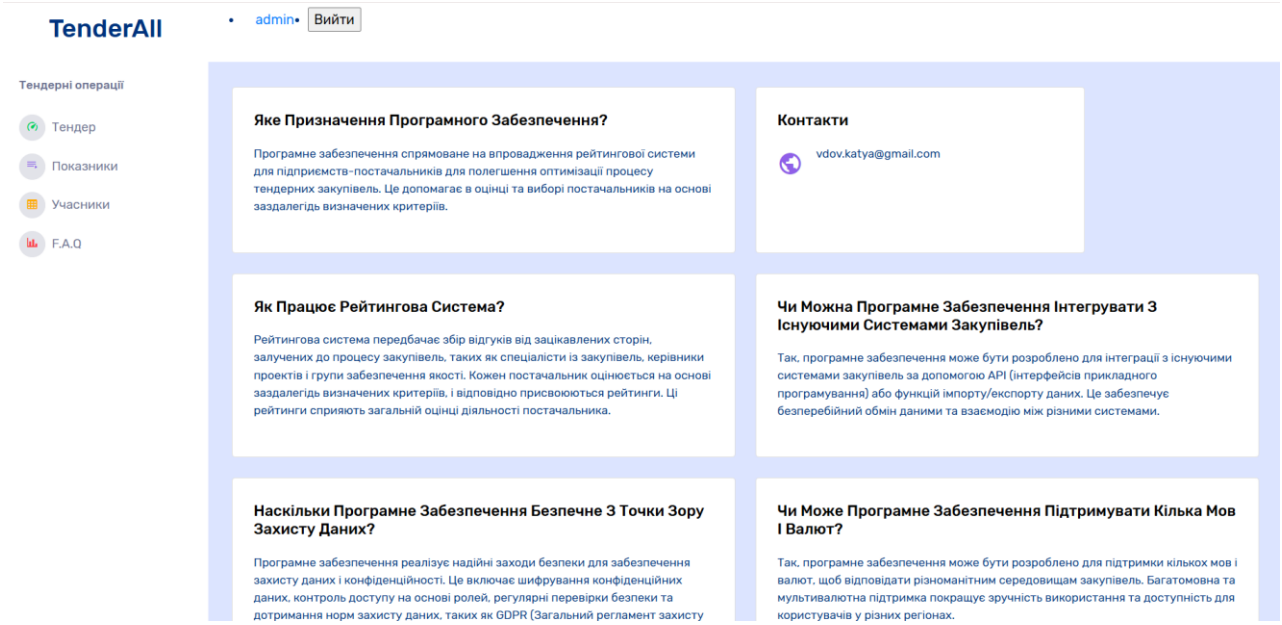


Рис. 4.13 Сторінка F.A.Q

Тепер після внесення показників та учасників ми можемо заповнити таблицю значеннями для складання рейтингу (рисунок 4.14).

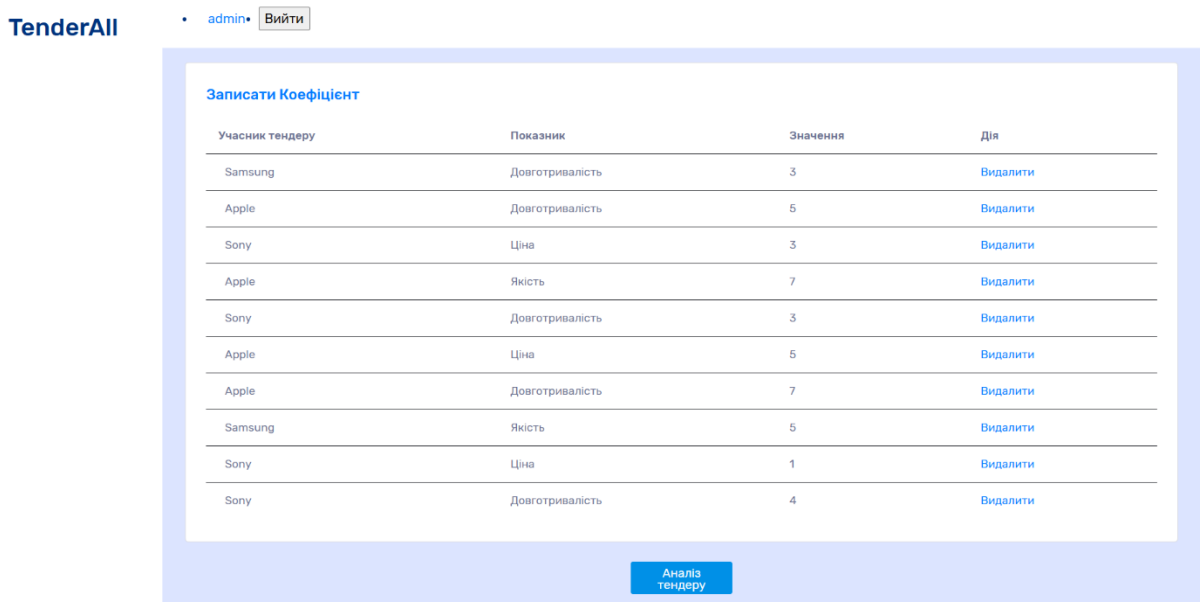


Рис. 4.14 Таблиця коефіцієнтів

Можна додати новий коефіцієнт, для цього натискаємо на кнопку «Записати коефіцієнт», далі відкривається відповідне вікно з формами заповнення даних (рисунок 4.15).

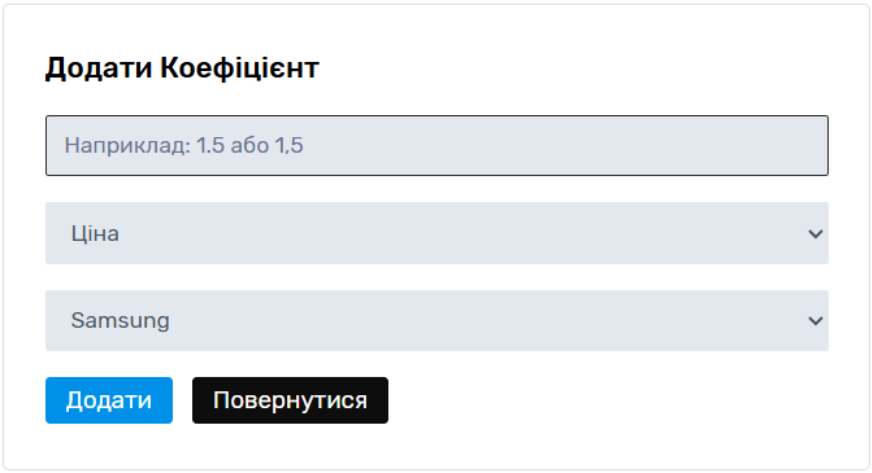


Рис. 4.15 Форма заповнення коефіцієнту

І після натискання «Аналіз тендеру» нам формується топ учасників тендеру та їх оцінка (рисунок 4.16).

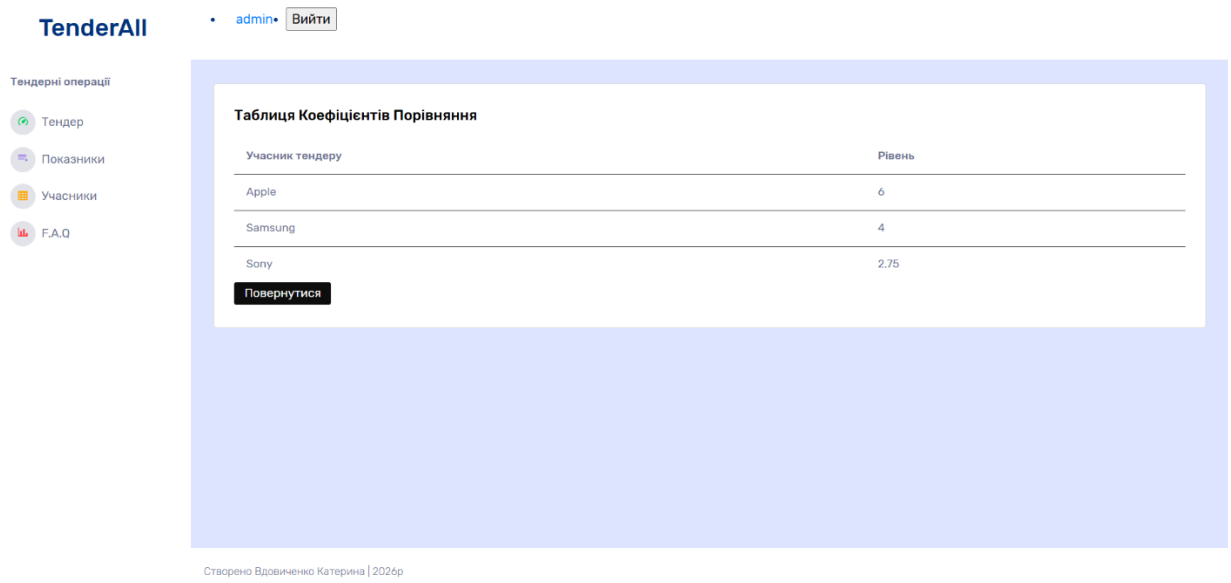


Рис. 4.16 Таблиця коефіцієнтів порівняння

4.2 Вимоги до апаратного та програмного забезпечення системи рейтингової оцінки постачальників

Апаратні вимоги які необхідні бути у комп’ютера для функціонування програмного забезпечення:

- процесор із принаймні двома ядрами, наприклад Intel Core i3 або AMD Ryzen 3l
- оперативна пам’ять 4 ГБ;
- SSD 10 ГБ або більше.

Діаграма розгортання служить для ілюстрації загальної топології розподіленої системи та демонструє представлення різних артефактів на окремих вузлах системи. Його призначення - візуалізувати фізичну структуру системи, зображуючи, як програмні компоненти розгортаються на різних вузлах.

На схемі вузли зображені у вигляді прямокутників або піктограм, що уособлюють фізичне обладнання чи середовища виконання. Артефакти програмного забезпечення, такі як програми, модулі чи служби, зображуються всередині цих вузлів, вказуючи на місце їхнього розгортання.

Відносини розгортання, такі як асоціації, залежності або залежності розгортання, демонструють зв’язки та взаємодію між компонентами програмного забезпечення та апаратними вузлами. Ці зв’язки допомагають зрозуміти, як компоненти спілкуються та співпрацюють у системній інфраструктурі.

Серверна сторона обробляє запити, взаємодіє з базою даних, спілкується з іншими серверами, обробляє дані користувача та структурує веб-додатки.

Клієнт — це користувач, який взаємодіє з програмою за допомогою веб-браузера.

Сервер бази даних — це програмний ресурс, який використовується для зберігання даних у системі.

На рисунку 4.17 зображено схему розгортання цієї системи. Архітектура клієнт-сервер реалізована на двох окремих серверах.

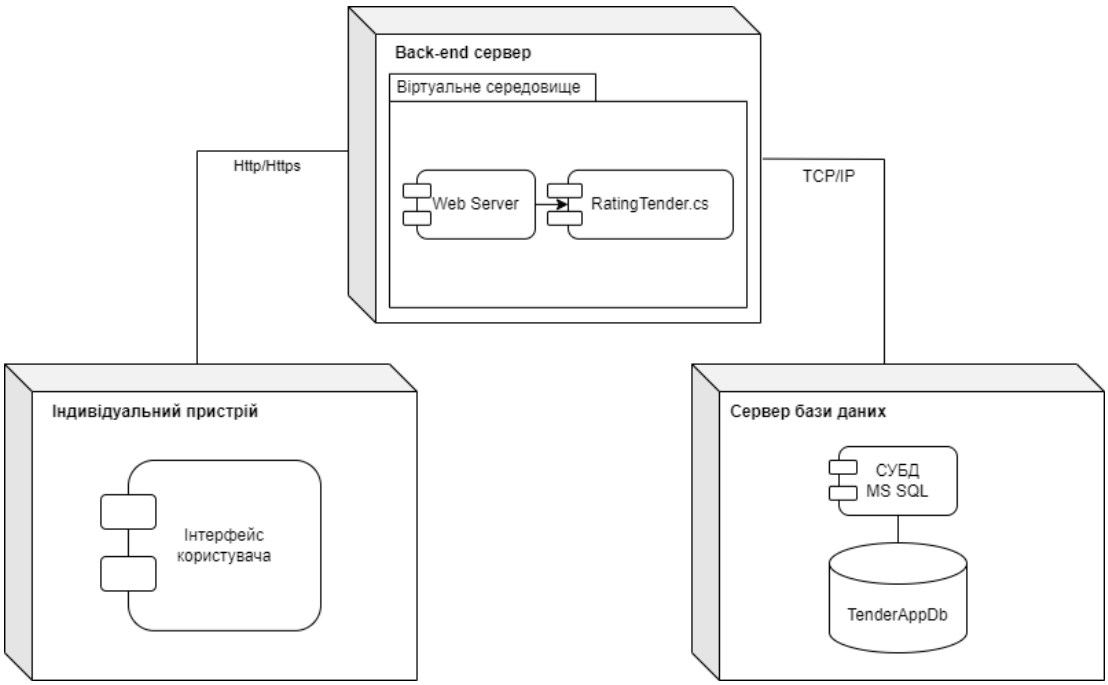


Рис. 4.17 Діаграма розгортання

ВИСНОВКИ

Бакалаврська кваліфікаційна робота спрямована на розробку програмного забезпечення системи рейтингової оцінки підприємств-постачальників з метою оптимізації процесу тендерних закупівель має значний потенціал для підвищення ефективнності та прозорості процесів закупівель в організаціях. Завдяки розробці комплексного програмного рішення, адаптованого до конкретних потреб рейтингових підприємств-постачальників, проект спрямований на вирішення ключових проблем, пов’язаних із тендерними закупівлями.

Робота розпочалася з вивчення предметної області та формулювання проблеми дослідження. Існуючі рішення були переглянуті, щоб виявити прогалини та можливості для вдосконалення програмного забезпечення для оптимізації процесу тендерних закупівель. Це послужило основою для наступних етапів роботи.

Було створено логічну модель даних, що представляє вимоги до інформації та зв’язки в програмному забезпеченні для оптимізації процесу тендерних закупівель. Ця модель лягла в основу проєктування програмної системи. Розроблено фізичну модель даних, що забезпечує ефективнне зберігання та пошук даних за допомогою вибраної системи керування базами даних.

Завдяки використанню сучасних технологій і методологій, таких як C#, ASP.NET Core MVC, Identity, EF Core, Razor Pages, HTML, CSS, JS і Photoshop CS6, проект намагається створити надійну та зручну платформу. Ця платформа сприятиме оцінці та відбору постачальників на основі заздалегідь визначених критеріїв, оптимізуючи процес закупівель і забезпечуючи дотримання стандартів якості.

Етап впровадження включав переведення дизайну в фактичний код та інтеграцію різних програмних компонентів. Для забезпечення функціональності та надійності програмного забезпечення було проведено ретельне тестування та налагодження. Відгуки користувачів і повторювані вдосконалення відіграли

важливу роль у підвищенні продуктивності та зручності використання програмного забезпечення.

Успішне впровадження цього програмного рішення може революціонізувати практику закупівель, що призведе до покращення управління постачальниками, економії коштів і, зрештою, підвищення ефективнності організації. Крім того, фокус проекту на оптимізації процесу тендерних закупівель узгоджується з галузевими тенденціями до цифровізації та автоматизації, позиціонуючи його як цінний актив як для державних, так і для приватних організацій.

Таким чином, дипломний проект є значним кроком у напрямку модернізації практики закупівель і сприяння прозорості та ефективнності оцінки постачальників. Завдяки своєму інноваційному підходу та використанню передових технологій проект має потенціал для значного впливу на сферу управління закупівлями.

Апробація результатів дослідження - тези доповідей:

Вдовиченко К.В., Задонцев Ю.В. Розробка Web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. стор. 316-317.

Вдовиченко К.В., Задонцев Ю.В. Забезпечення безпеки у Web-застосунку рейтингової оцінки постачальника для тендерних закупівель. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. стор. 343-344.

ПЕРЕЛІК ПОСИЛАНЬ

1. Правила проведення тендера в Україні. URL: https://tender.uub.com.ua/pro-prozorro/pravila-provedennya-tendera-v-ukraini-scho-potribno-znaty (дата звернення: 04.04.2026).

2. Infobox Prozorro: інформаційний портал про публічні закупівлі. Інформаційний ресурс - Інфобокс Прозорро. URL: https://infobox.prozorro.org/ (дата звернення: 04.04.2026).

3. Procurement Software Requirements in 2026 (+ Criteria with Examples). URL: https://www.spendflo.com/blog/procurement-software-requirements (date of access: 12.04.2026).

4. Dryden. Procurement & Purchasing Consulting Firm | Dryden Group. URL: https://www.drydengroup.com/blog/6-procurement-methods (date of access: 09.04.2026).

5. What is Unified Modeling Language (UML). Visual Paradigm - AI-Powered Visual Modeling Platform. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ (date of access: 13.04.2026).

6. Sequence Diagram Tutorial - Complete Guide with Examples | Creately. Creately. URL: https://creately.com/guides/sequence-diagram-tutorial/ (date of access: 09.04.2026).

7. SRM система APS SMART | Управління закупівлями підприємства. APS SMART. URL: https://www.aps-smart.com/ (дата звернення: 08.04.2026).

8. E-Tender | Офіційний майданчик для державних та комерційних закупівель. E-Tender. URL: https://e-tender.ua/ (дата звернення: 08.04.2026).

9. Comparative Analysis of Data Modeling Design Tools / G. Carvalho et al. IEEE Access. 2022. Vol. 10. P. 3351–3365. URL: https://doi.org/10.1109/access.2021.3139071 (date of access: 10.04.2026).

10. Dennis A., Tegarden D., Wixom B. Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley & Sons, Limited, John, 2021.

11. ResearchGate. URL: https://www.researchgate.net (date of access: 15.04.2026).

12. Ozkaya M., Erata F. A survey on the practical use of UML for different software architecture viewpoints. Information and Software Technology. 2020. Vol. 121. P. 106275. URL: https://doi.org/10.1016/j.infsof.2020.106275 (date of access: 17.04.2026).

13. What Is a Relational Database | Oracle. Oracle. URL: https://www.oracle.com/database/what-is-a-relational-database/ (date of access: 20.04.2026).

14. Мікула М., Коцюк Ю., Мікула О. Організація баз даних та знань. Вид-во Нац. ун-ту «Острозь-ка акад.», 2021. URL: https://doi.org/10.25264/978-617-8041-08-3 (дата звернення: 21.04.2026).

15. Курс SQL – Мова запитів до баз даних з нуля | SkillsUp. SkillsUp. URL: https://skillsup.ua/blog/articles/2025/06/learning-sql-together-the-basics-of-the-database-query-language/ (дата звернення: 20.04.2026).

16. W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/ (date of access: 22.04.2026).

17. Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (date of access: 22.04.2026).

18. Tu Z. Research on the Application of Layered Architecture in Computer Software Development 2023. Vol. 11, no. 3. P. 34–38. URL: https://doi.org/10.54097/jceim.v11i3.08 (date of access: 29.04.2026).

19. Web-програмування. Частина 1 (frontend) / Б. В. В. та ін. Кропивницький : ЦНТУ, 2022. 208 с. URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/2bfb5a3c-54d9-4283-89c1-5e190e8aa151/content

20. Database Server. ScienceDirect Topics. 2022. URL: https://www.sciencedirect.com/topics/computer-science/database-server (дата звернення: 02.05.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель

Виконала студентка 4 курсу
групи ПД-43
Вдовиченко Катерина Вячеславівна
Керівник роботи канд. техн. наук
Задонцев Юрій Вікторович

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення прозорості, спрощення вибору постачальників та оптимізація процесу тендерних закупівель за рахунок створення web-застосунку для рейтингової оцінки постачальників.

Об'єкт дослідження: процеси тендерних закупівель та рейтингової оцінки постачальників.

Предмет дослідження: методи, технології та програмні засоби реалізації web-застосунку для рейтингової оцінки постачальників у системі тендерних закупівель.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної галузі організації та проведення тендерних закупівель для визначення актуальних проблем та потреб у програмних засобах для проведення тендерів.
2. Дослідити методи рейтингової оцінки постачальників для формування підходу до оцінювання учасників тендеру.
3. Провести аналіз існуючих аналогів програмних засобів для рейтингової оцінки постачальників у тендерних закупівлях з метою виявлення їх переваг, недоліків і слабких сторін.
4. Визначити функціональні та нефункціональні вимоги до web-застосунку.
5. Провести аналіз та обґрунтувати вибір технологій, інструментів та програмних засобів для реалізації web-застосунку.
6. Виконати проєктування та розробку web-застосунку для рейтингової оцінки постачальників у системі тендерних закупівель.
7. Провести тестування web-застосунку для перевірки коректності роботи його основних функцій.

3

АНАЛІЗ АНАЛОГІВ

Показник	**E-Tender**	**APS SMART**	**TenderAll**
Основна мета	Майданчик для участі у тендерах	Автоматизація закупівель та управління постачальниками	Рейтингове оцінювання постачальників
Автоматизований рейтинг постачальників	Відсутній	Частково реалізований	Повністю реалізований (на основі критеріїв)
Механізм формування рейтингу	Відсутній	Частково реалізований	Автоматичний після введення показників і коефіцієнтів
Робота з показниками оцінки	Відсутня	Обмежена	Додавання, перегляд, видалення показників
Введення коефіцієнтів	Відсутнє	Частково	Реалізовано через форму введення
Формування результату (топ постачальників)	Відсутнє	Частково	Автоматичне формування рейтингу учасників

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Забезпечення введення та збереження даних про постачальників для їх подальшого оцінювання.
2. Редагування та видалення коефіцієнтів, що використовуються для розрахунку рейтингу.
3. Підтримка механізмів автентифікації та авторизації для захисту конфіденційної інформації.
4. Застосування алгоритмів для оцінювання постачальників на основі зважених критеріїв ефективності.
5. Формування звітів і візуальне відображення рейтингів та ключових показників ефективності.

Нефункціональні вимоги:

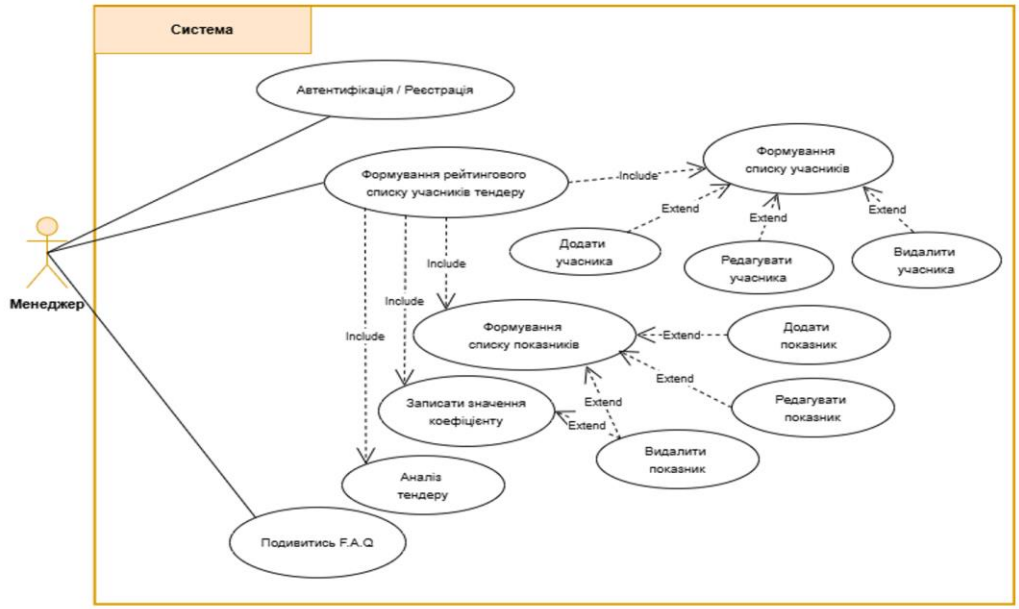
1. Формування результатів аналізу тендера не перевищує 5 секунд за умови стандартного навантаження.
2. Забезпечення автентифікації користувачів із перевіркою облікових даних під час входу в систему та відображанням повідомлень про помилки у разі некоректного введення.
3. Забезпечення доступу до основних функцій у межах двох або трьох кроків від головної сторінки.
4. Забезпечення обробки даних постачальників, показників і коефіцієнтів у процесах введення, перегляду та формування рейтингу без зниження швидкодії інтерфейсу.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

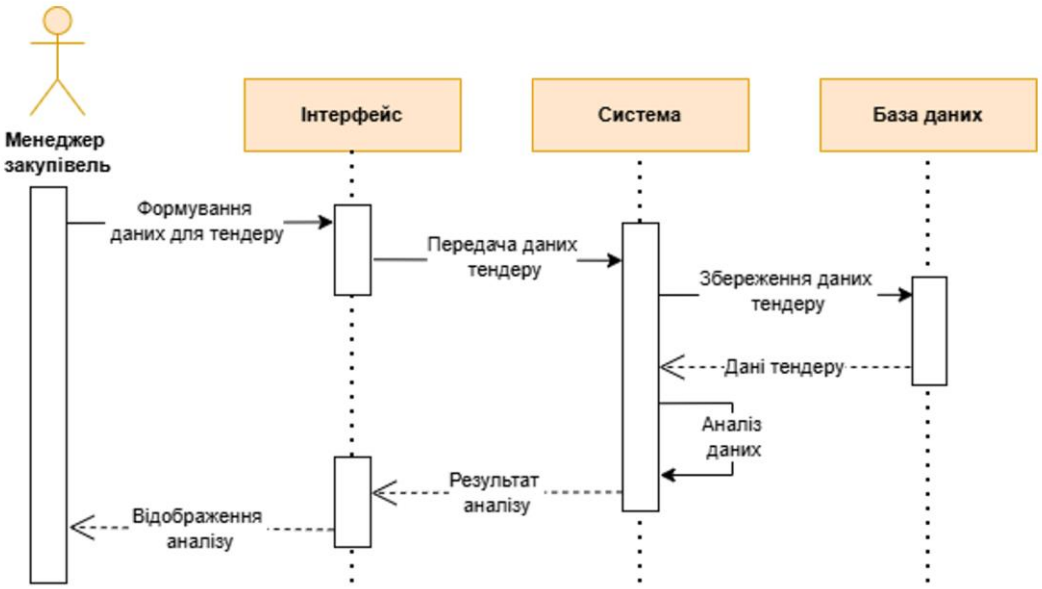
6

ДІАГРАМА ПРЕЦЕДЕНТІВ



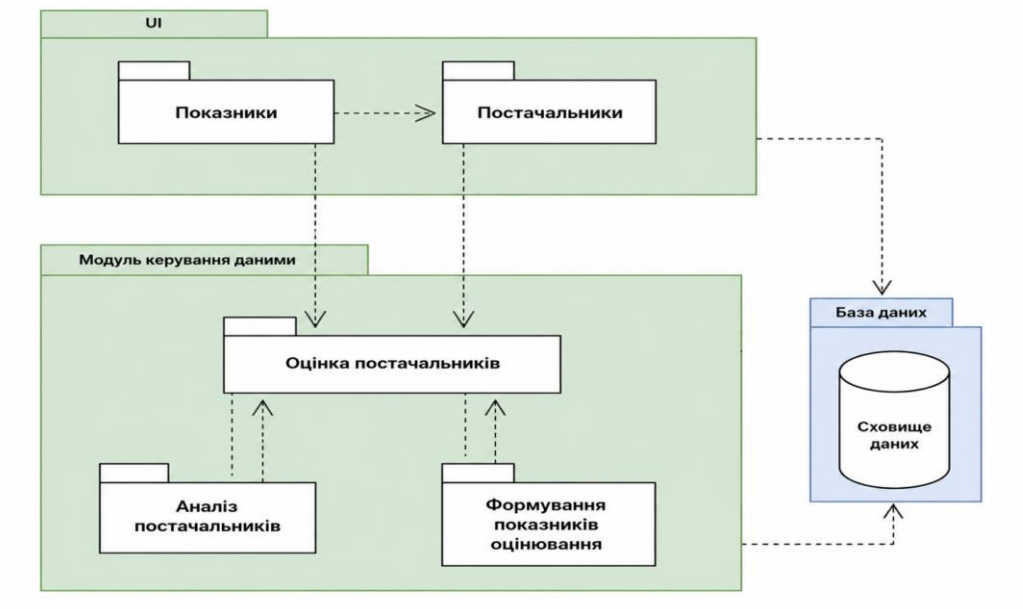
7

ДІАГРАМА ПОСЛІДОВНОСТІ



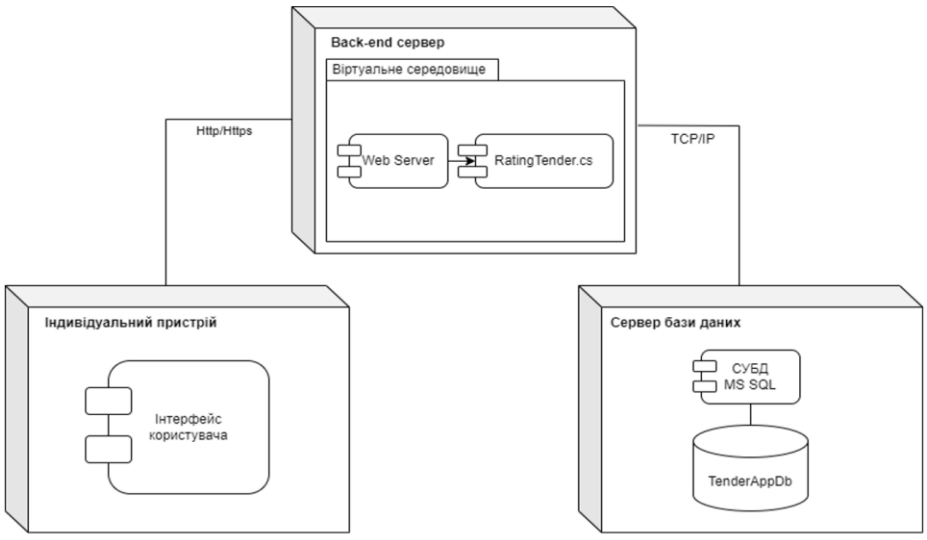
8

ДІАГРАМА ПАКЕТІВ



9

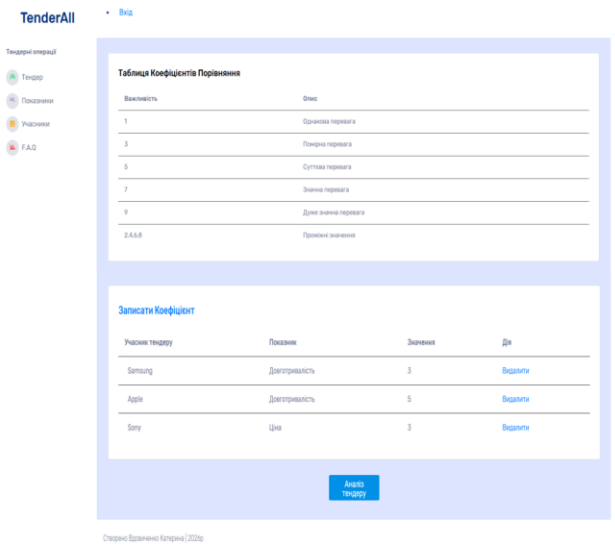
ДІАГРАМА РОЗГОРТАННЯ



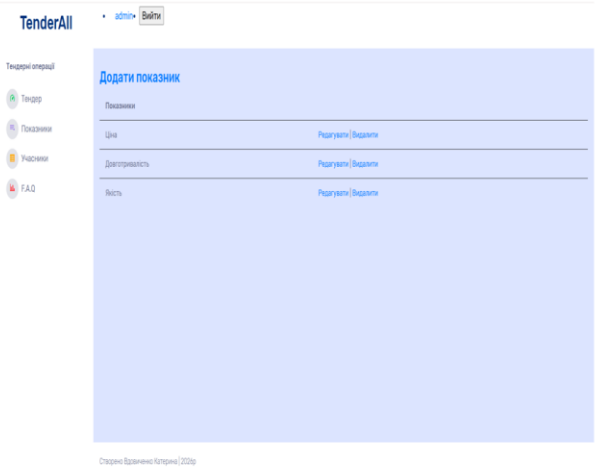
10

ЕКРАННІ ФОРМИ

Головна сторінка

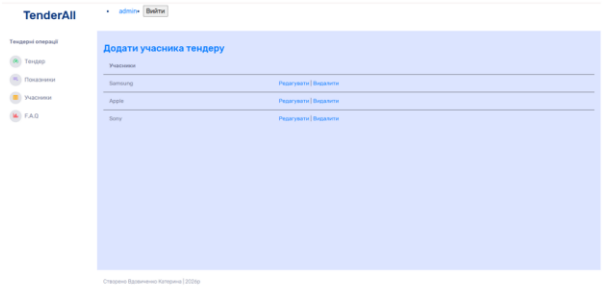


Сторінка показників

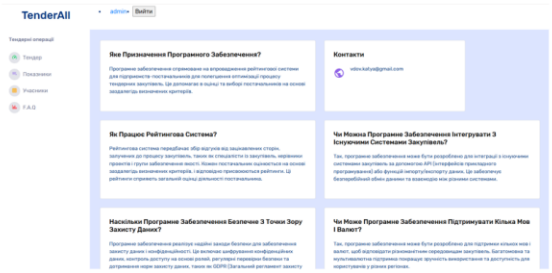


ЕКРАННІ ФОРМИ

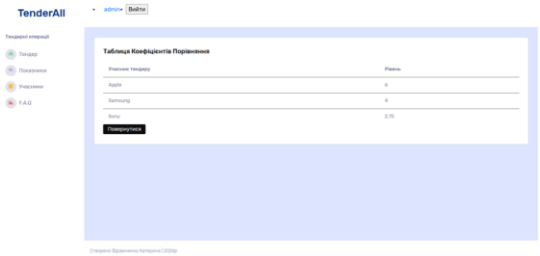
Сторінка учасників



Сторінка питання - відповідь



Таблиця коефіцієнтів порівняння



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Вдовиченко К.В., Задонцев Ю.В. Розробка Web-застосунку для рейтингової оцінки постачальників з метою оптимізації тендерних закупівель. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. стор. 316-317.

2. Вдовиченко К.В., Задонцев Ю.В. Забезпечення безпеки у Web-застосунку рейтингової оцінки постачальника для тендерних закупівель. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. стор. 343-344.

ВИСНОВКИ

1. Проведено аналіз предметної галузі тендерних закупівель та були визначені актуальні проблеми та потреби у програмних засобах для проведення тендерів.
2. Досліджено методи рейтингової оцінки постачальників, завдяки чому сформовано підхід до оцінювання учасників тендеру.
3. Проведено аналіз існуючих аналогів, який показав їх переваги та недоліки, що дозволило враховати це під час розробки.
4. Визначено функціональні та нефункціональні вимоги web-застосунку.
5. Проаналізовані та обрані технології та інструменти для реалізації web-застосунку.
6. Розроблено web-застосунок для рейтингової оцінки постачальників, тому автоматизовано процес аналізу результатів тендеру та оптимізовано вибір постачальників.
7. Реалізовано функції управління постачальниками, показниками та формування рейтингу, тому підвищено прозорість і зручність проведення тендерних закупівель.
8. Проведено тестування системи, де підтверджено коректність роботи основних функцій web-застосунку.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Функція створення показника.

```
namespace TenderApplication.Controllers
{
   public class MeasuresController : Controller
   {
      private readonly ApplicationDbContext _context;

      public MeasuresController(ApplicationDbContext
context)
      {
         _context = context;
      }

      // GET: Measures
      public async Task<IActionResult> Index()
      {
         return View(await
_context.Measures.ToListAsync());
      }

      // GET: Measures/Details/5
      public async Task<IActionResult> Details(int? id)
      {
         if (id == null)
         {
            return NotFound();
         }

         var measure = await _context.Measures
            .FirstOrDefaultAsync(m => m.MeasureId ==
id);
         if (measure == null)
         {
            return NotFound();
         }

         return View(measure);
      }

      // GET: Measures/Create
      public IActionResult Create()
      {
         return View();
      }

      // POST: Measures/Create
      // To protect from overposting attacks, enable the
specific properties you want to bind to.
      // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
      [HttpPost]
      [ValidateAntiForgeryToken]
      public async Task<IActionResult>
Create([Bind("MeasureId,MeasureName")] Measure
measure)
      {
         if (ModelState.IsValid)
         {
            _context.Add(measure);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
         }
         return View(measure);
      }

      // GET: Measures/Edit/5
      public async Task<IActionResult> Edit(int? id)
      {
         if (id == null)
         {
            return NotFound();
         }

         var measure = await
_context.Measures.FindAsync(id);
         if (measure == null)
         {
            return NotFound();
         }
         return View(measure);
      }

      // POST: Measures/Edit/5
      // To protect from overposting attacks, enable the
specific properties you want to bind to.
      // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
      [HttpPost]
      [ValidateAntiForgeryToken]
      public async Task<IActionResult> Edit(int id,
[Bind("MeasureId,MeasureName")] Measure measure)
      {
         if (id != measure.MeasureId)
         {
            return NotFound();
         }

         if (ModelState.IsValid)
         {
            try
            {
               _context.Update(measure);
               await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
               if (!MeasureExists(measure.MeasureId))
               {
```

```
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(measure);
}

}
}
```

Функція запису коефіцієнта у тендері.

```
namespace TenderApplication.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController>
_logger;
        private readonly ApplicationDbContext _context;

        public HomeController(ILogger<HomeController>
logger, ApplicationDbContext context)
        {
            _logger = logger;
            _context = context;
        }

        // GET
        public async Task<IActionResult> Index()
        {
            var applicationDbContext =
_context.Ratings.Include(r => r.Measure).Include(r =>
r.Member);
            return View(await
applicationDbContext.ToListAsync());
        }

        // GET: Create
        public IActionResult Create()
        {
            ViewData["MeasureId"] = new
SelectList(_context.Measures, "MeasureId",
"MeasureName");
            ViewData["MemberId"] = new
SelectList(_context.Members, "MemberId",
"MemberName");
            return View();
        }

        // POST: Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("RatingId,Value,MeasureId,MemberId")]
Rating rating)
        {
            if (ModelState.IsValid)
            {
                _context.Add(rating);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["MeasureId"] = new
SelectList(_context.Measures, "MeasureId",
"MeasureName", rating.MeasureId);
            ViewData["MemberId"] = new
SelectList(_context.Members, "MemberId",
"MemberName", rating.MemberId);
            return View(rating);
        }

        // GET: Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var rating = await _context.Ratings
                .Include(r => r.Measure)
                .Include(r => r.Member)
                .FirstOrDefaultAsync(m => m.RatingId == id);
            if (rating == null)
            {
                return NotFound();
            }

            return View(rating);
        }

        // POST: Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
DeleteConfirmed(int id)
        {
            var rating = await
_context.Ratings.FindAsync(id);
            if (rating != null)
            {
                _context.Ratings.Remove(rating);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool RatingExists(int id)
        {
            return _context.Ratings.Any(e => e.RatingId ==
id);
        }

        public IActionResult Privacy()
        {
            return View();
        }
```

```
    [ResponseCache(Duration = 0, Location =
ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
    public IActionResult TopMembers()
    {
        var topMembers = _context.Ratings
            .Include(r => r.Member)
            .GroupBy(r => r.Member.MemberName)
            .Select(g => new
            {
                MemberName = g.Key,
                AverageValue = g.Average(r => r.Value)
            })
            .OrderByDescending(g => g.AverageValue)
            .ToList();

        return View(topMembers);
    }
}
      }
```