Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

з напряму підготовки 121 «Інженерія програмного забезпечення»

На тему: *Розробка автоматизованої системи підрахунку рейтингу науково-педагогічних працівників*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студентка гр. ІПЗ-24м
_____ *Колтунова Є. Є.*

Керівник кваліфікаційної роботи _____ */ Стрюк А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2025

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *магістр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2025__ р.

ЗАВДАННЯ
на кваліфікаційну роботу
студентки групи ІПЗ-24м Колтунової Єлизавети Євгеніївни

1. Тема: *Розробка автоматизованої системи підрахунку рейтингу науково-педагогічних працівників* затверджено наказом по КНУ №___ від «__» _____ 2025 р.
2. Термін подання студентом закінченого проекту «__» _____ 2025 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 123 сторінки, 5 рисунків, 2 додатоки, 26 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): дослідження рівня автоматизації розрахунку рейтингового оцінювання науково-педагогічних працівників, проектування програмного забезпечення автоматизованої системи розрахунку рейтингу науково-педагогічних працівників, розробка програмного забезпечення рейтингового оцінювання науково-педагогічних працівників, економічне обгрунтування проекту.
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програм. Діаграма варіантів використання. Діаграма компонентів. Діаграма класів. Приклади роботи розробленної програми.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*15.02.2025-20.02.2025*
2	*Аналіз інформаційних джерел*	*21.02.2025-09.03.2025*
3	*Визначення вимог до програмного забезпечення*	*10.03.2025-18.04.2025*
4	*Розробка функціональної схеми*	*19.04.2025-23.05.2025*
5	*Розробка алгоритмів*	*24.05.2025-31.06.2025*
6	*Розробка бази даних*	*01.06.2025-14.07.2025*
7	*Розробка програмного забезпечення*	*15.07.2025-13.09.2025*
8	*Тестування програмного забезпечення*	*14.09.2025-20.10.2025*
9	*Оформлення пояснювальної записки*	*21.10.2025-12.11.2025*
10	*Розробка демонстраційних матеріалів*	*13.11.2025-18.12.2025*

Дата видачі завдання: «_____»_____ 2025 р.
Студентка _____ Колтунова Є. Є.
Керівник роботи _____ Стрюк А. М.

РЕФЕРАТ

РЕЙТИНГОВЕ ОЦІНЮВАННЯ, АВТОМАТИЗАЦІЯ, ЗБІР ДАНИХ, .NET, REACT, POSTGRESQL

Пояснювальна записка: 123 с., 5 рис., 26 використаних джерел.

Метою роботи є проектування, технічне обґрунтування та розрахунок економічної ефективності впровадження автоматизованої системи рейтингового оцінювання науково-педагогічних працівників Криворізького національного університету, яка забезпечить інтеграцію з існуючою IT-інфраструктурою, мінімізує адміністративне навантаження та підвищить прозорість кадрової політики.

Об'єкт дослідження: Процес управління кадровим потенціалом та забезпечення якості освіти у закладі вищої освіти.

Предмет дослідження: Методи, інформаційні технології та інструменти автоматизації рейтингового оцінювання діяльності науково-педагогічних працівників.

В процесі роботи досліджено нормативну базу рейтингування в українських ЗВО та специфіку КНУ. Виявлено недоліки поточної моделі оцінювання та сформульовано функціональні вимоги до нової системи. Розроблено архітектуру системи на основі сучасного технологічного стеку (.NET 8, React, PostgreSQL). Запропоновано концепцію "Widget-First" для безшовної інтеграції рейтингів у веб-ресурси університету. Спроектовано схему бази даних з використанням JSONB для забезпечення гнучкості критеріїв оцінювання. Створено стратегію інтеграції з корпоративними сервісами Microsoft 365 (для єдиної авторизації). Розроблено механізми автоматичного збору даних із зовнішніх наукометричних баз (Scopus API, ORCID) та внутрішніх систем. Розраховано собівартість розробки системи. Проведено порівняльний аналіз вартості власної розробки та аутсорсингу. Визначено економічний ефект від впровадження та термін окупності проєкту за рахунок економії фонду робочого часу.

ABSTRACT

RATING ASSESSMENT, AUTOMATION, DATA COLLECTION, .NET, REACT, POSTGRESQL

Explanatory note: 123 pages, 5 figures, 26 sources used.

The purpose of the work is to design, technically justify, and calculate the economic efficiency of implementing an automated rating system for scientific and pedagogical workers at Kryvyi Rih National University, which will ensure integration with the existing IT infrastructure, minimize the administrative burden, and increase the transparency of personnel policy.

Object of research: The process of managing human resources and ensuring the quality of education in higher education institutions.

Subject of research: Methods, information technologies, and tools for automating the rating assessment of scientific and pedagogical workers.

In the course of the work, the regulatory framework for ranking in Ukrainian higher education institutions and the specifics of KNU were studied. The shortcomings of the current evaluation model were identified and functional requirements for the new system were formulated. The system architecture was developed based on a modern technology stack (.NET 8, React, PostgreSQL). The “Widget-First” concept was proposed for seamless integration of ratings into the university's web resources. A database schema was designed using JSONB to ensure flexibility of evaluation criteria. A strategy for integration with Microsoft 365 corporate services (for single sign-on) was created. Mechanisms for automatic data collection from external scientometric databases (Scopus API, ORCID) and internal systems have been developed. The cost of developing the system has been calculated. A comparative analysis of the cost of in-house development and outsourcing has been carried out. The economic effect of implementation and the payback period of the project due to savings in working time have been determined.

ЗМІСТ

ВСТУП

В умовах глобальної цифровізації та інтеграції українського освітнього простору до Європейського дослідницького простору (ERA), ефективність управління людським капіталом стає визначальним фактором конкурентоспроможності закладу вищої освіти. Для Криворізького національного університету (КНУ), як провідного науково-освітнього центру промислового регіону, питання об'єктивного оцінювання та стимулювання науково-педагогічних працівників (НПП) набуває критичного значення.

Нормативно-правові зміни 2025 року: Зміни у податковому законодавстві (підвищення військового збору до 5%) та нові вимоги МОН щодо співвідношення навчального та наукового навантаження вимагають від керівництва університету жорсткого контролю ефективності використання фонду оплати праці та переходу до доказового менеджменту (evidence-based management).

Існуюча практика збору паперових звітів та заповнення розрізнених електронних таблиць призводить до значних втрат робочого часу висококваліфікованого персоналу. За попередніми оцінками, ручна обробка рейтингів 900 викладачів КНУ коштує університету понад 1,5 млн грн щорічно у вигляді оплаченого, але непродуктивного часу.

Національне агентство із забезпечення якості вищої освіти (НАЗЯВО) вимагає наявності прозорої внутрішньої системи забезпечення якості, де рейтинг викладача є не каральним інструментом, а механізмом планування кар'єри. Без автоматизації забезпечити прозорість та оперативний зворотний зв'язок неможливо.

Необхідність автоматизованого збору наукометричних даних (Scopus, Web of Science, ORCID) для коректного позиціювання університету в міжнародних рейтингах (QS, THE) та приєднання до ініціатив відкритої науки (CoARA).

Метою роботи є проектування, технічне обґрунтування та розрахунок економічної ефективності впровадження автоматизованої системи рейтингового оцінювання науково-педагогічних працівників Криворізького національного університету, яка забезпечить інтеграцію з існуючою IT-інфраструктурою, мінімізує адміністративне навантаження та підвищить прозорість кадрової політики.

Для досягнення поставленої мети було вирішено наступні задачі:

- Аналіз предметної області та бізнес-процесів;
- Архітектурне проектування;
- Розробка інтеграційних сценаріїв;
- Економічне обґрунтування.

Об'єкт дослідження: Процес управління кадровим потенціалом та забезпечення якості освіти у закладі вищої освіти.

Предмет дослідження: Методи, інформаційні технології та інструменти автоматизації рейтингового оцінювання діяльності науково-педагогічних працівників.

1 ДОСЛІДЖЕННЯ РІВНЯ АВТОМАТИЗАЦІЇ РОЗРАХУНКУ РЕЙТИНГОВОГО ОЦІНЮВАННЯ НАУКОВО-ПЕДАГОГІЧНИХ ПРАЦІВНИКІВ

1.1 Актуальність та механізми рейтингового оцінювання науково-педагогічних працівників в українських закладах вищої освіти

Сучасна система вищої освіти України переживає період глибоких тектонічних зрушень, зумовлених як глобальними трендами цифровізації та глобалізації, так і специфічними внутрішніми викликами – повномасштабною війною, демографічною кризою та необхідністю прискореної євроінтеграції. У цьому складному ландшафті питання ефективності використання кадрового потенціалу набуває екзистенційного значення для університетів. Актуальність проведення рейтингового оцінювання науково-педагогічних працівників (НПП) виходить далеко за межі простої адміністративної звітності; це інструмент стратегічного виживання та розвитку інституцій в умовах жорсткої конкуренції за ресурси та абітурієнтів.

Історично українська вища школа спиралася на систему державної атестації та фіксованих тарифних сіток, які мало залежали від поточної продуктивності викладача. Однак перехід до принципів університетської автономії, закріплених у Законі України "Про вищу освіту" , відкрив шлях до впровадження внутрішніх систем забезпечення якості (Internal Quality Assurance), де рейтингування стало ключовим механізмом диференціації та стимулювання. Рейтингове оцінювання сьогодні розглядається не просто як спосіб контролю, а як складний управлінський механізм, що дозволяє узгодити індивідуальні цілі науковця зі стратегічними пріоритетами університету – будь то входження у світові рейтинги QS та Times Higher Education, чи отримання статусу дослідницького університету.

В умовах воєнного стану, коли бюджетне фінансування освіти є критично обмеженим, а вимоги до наукової результативності зростають, рейтингові системи стають основою для "природного відбору" найбільш

ефективних кадрів та справедливого розподілу обмеженого фонду матеріального заохочення. Водночас, цей процес супроводжується гострими дискусіями в академічному середовищі щодо об'єктивності критеріїв, ризиків бюрократизації та психологічного тиску на викладачів.

Правовий фундамент рейтингового оцінювання в Україні формується на перетині імперативних норм державного законодавства та локальних актів університетів, що реалізують своє право на автономію. Основним документом, що визначає рамкові умови, є Закон України "Про вищу освіту", який декларує необхідність створення внутрішніх систем забезпечення якості освіти.

Рейтингове оцінювання: Це інструмент внутрішнього менеджменту, який впроваджується за рішенням Вченої ради ЗВО. На відміну від атестації, рейтингування зазвичай проводиться щорічно і має на меті не просто підтвердження відповідності посаді, а вимірювання динаміки та результативності роботи за звітний період. Законодавство не встановлює єдиної методики рейтингування, залишаючи це на розсуд закладів, що породжує значну варіативність підходів.

Постанова Кабінету Міністрів України №800 "Деякі питання підвищення кваліфікації педагогічних і науково-педагогічних працівників" опосередковано впливає на рейтинги, оскільки встановлює вимоги до професійного розвитку, які часто інкорпоруються в рейтингові показники як обов'язкові умови допуску або як бальні індикатори.

Аналіз внутрішньої нормативної документації провідних українських університетів дозволяє виділити декілька архетипових моделей рейтингування, кожна з яких відображає специфічну корпоративну культуру та стратегічні пріоритети закладу.

Інтегрально-фінансова модель (Національний університет "Львівська політехніка")

Система Львівської політехніки є однією з найбільш комплексних та жорстко пов'язаних з фінансовими результатами діяльності підрозділів. Вона

базується на філософії колективної відповідальності, де індивідуальні досягнення трансформуються в успіх кафедри.

Механізм функціонування:

Рейтингування проводиться на рівні кафедр за чотирма напрямами: кадровий потенціал, освітня діяльність, наукова діяльність та міжнародна співпраця. Показники розраховуються на одну штатну одиницю НПП, що нівелює перевагу великих кафедр і змушує кожного працівника робити внесок.

Фінансова імплікація:

Унікальністю моделі є пряма кореляція між місцем кафедри в рейтингу та розміром матеріального заохочення її керівництва.

Якщо кафедра входить до першої групи (лідери), завідувач може отримати надбавку до 150% посадового окладу.

Кафедри третьої групи (аутсайдери) позбавляються права на встановлення надбавок завідувачам, що створює потужний тиск на керівників для мобілізації колективу.

Критеріальна база:

Особливий акцент зроблено на міжнародній видимості та комерціалізації. Високо оцінюються монографії, видані мовами країн ОЕСР (40 балів) та публікації у виданнях Springer/Elsevier (60 балів). Також враховуються продані ліцензії на інтелектуальну власність вартістю не менше 30 тис. грн.

Квартильно-індивідуалізована модель (ТНПУ ім. В. Гнатюка)

Тернопільський національний педагогічний університет демонструє модель, орієнтовану на персоніфіковану наукову результативність з глибокою диференціацією якості публікацій. Ця система стимулює "полювання" за високорейтинговими журналами.

Структура оцінювання науки:

Система балів за публікації є чітко ієрархізованою та прив'язаною до квартилів (Quartile) журналів у базах Scopus/Web of Science.8

Особливості:

Мультиплікатор впливовості: Індекс Гірша (h-index) не просто додається, а виступає множником. Показник h-index у Scopus/WoS множиться на 20, що робить цитованість найвагомішим фактором довгострокового успіху в рейтингу. Для порівняння, h-index у Google Scholar множиться лише на 5.

Грантовий пріоритет: Керівництво проєктами НФДУ або МОН оцінюється у 30 балів, що еквівалентно публікації в Q4, але вимагає значно більших організаційних зусиль.

Прозорість: Доступ до рейтингів є вільним для всіх викладачів на сайті університету, що створює ефект прозорої конкуренції ("гейміфікація" робочого процесу).

Цифрова екосистема та методичний акцент (КПІ ім. Ігоря Сікорського)

Київський політехнічний інститут імплементував систему рейтингування в автоматизоване цифрове середовище "Intellect". Це вирішує одну з головних проблем — бюрократичне навантаження зі збору паперових підтверджень.

Ключові характеристики:

Автоматизація: Дані про публікації підтягуються з профілів Scopus/ORCID, дані про навантаження — з деканатів. Викладач лише верифікує інформацію.

Фокус на методиці: На відміну від суто наукових моделей, КПІ високо цінує створення якісного навчального контенту. Видання підручника з грифом Вченої ради приносить 75 балів — це більше, ніж стаття в Q1 у багатьох інших вишах.

Інтернаціоналізація освіти: Викладання дисциплін іноземною мовою (англійською) тарифікується вище (18 балів за 50 годин), ніж викладання українською (13 балів), що підтримує стратегію залучення іноземних студентів.

Публічність: Рейтинги НПП є публічно доступними на сайті intellect.kpi.ua, що дозволяє абітурієнтам та партнерам оцінювати науковий рівень викладачів.

Специфічні та гібридні моделі

Військова специфіка (Інститут ВМС НУ "ОМА"): Тут рейтингування враховує специфічні компетентності. Наприклад, володіння іноземною мовою на рівні С2 (STANAG 6001) оцінюється у 50 балів — стільки ж, скільки стаття Q1. Враховуються також спортивні досягнення (Майстер спорту — 30 балів) та польові виходи. Це демонструє адаптивність рейтингових систем до профілю закладу.

Бар'єрна модель (КНУ ім. Тараса Шевченка): У провідному університеті країни рейтингові показники часто виступають не лише інструментом ранжування, а й жорстким фільтром допуску. Для участі в конкурсі на посаду доцента вимагається наявність не менше 5 публікацій у фахових виданнях та мінімум однієї у Scopus/WoS за останні 5 років. Відсутність цих показників автоматично блокує кар'єрне просування.

Порівняльний аналіз індикаторів ефективності

Узагальнюючи дані з проаналізованих ЗВО, можна структурувати універсальну матрицю індикаторів, яка використовується в українській вищій освіті. Вона складається з трьох основних кластерів, вага яких варіюється залежно від типу університету (дослідницький vs педагогічний).

Науково-дослідна діяльність (Scientific Impact)

Цей кластер є найбільш вагомим (40-60% загального рейтингу) і найбільш формалізованим.

Публікації: Безальтернативна домінанта баз Scopus та Web of Science. Українські "фахові видання категорії Б" оцінюються в 3-4 рази нижче, ніж міжнародні. Це відображає державну політику інтеграції в світовий науковий простір, але створює бар'єри для гуманітаріїв та суспільствознавців, чиї національні контексти важче публікувати у глобальних журналах.

Цитування: Впровадження індексів Гірша як множників (ТНПУ) або окремих критеріїв.

Гранти та госпдоговори: Оцінюється здатність залучати кошти ("гроші ходять за викладачем"). Керівництво проєктами оцінюється значно вище за виконавську участь.

Освітньо-методична робота (Teaching Excellence)

Оцінювання цього компоненту є складнішим через проблему вимірюваності якості.

Студентські оцінки: Практично всі проаналізовані ЗВО (ЗНУ, ЛПНУ, КПІ) використовують результати анонімного анкетування студентів. У ТНПУ показник "Якість викладання" розраховується як середнє арифметичне оцінок студентів. У Львівській політехніці успішність студентів також впливає на рейтинг викладача, що може стимулювати інфляцію оцінок.

Методичне забезпечення: Розробка дистанційних курсів на платформах Moodle/Google Classroom стала обов'язковим елементом після пандемії та початку війни. Якість силабусів та наявність власних підручників є вагомими індикаторами.

Організаційна та професійна активність (Community Service)

Експертна діяльність: Участь у спецрадах із захисту дисертацій, рецензування статей, робота в комісіях МОН/НАЗЯВО.

Мовна компетентність: Сертифікати рівня B2/C1 з англійської мови стають стандартом, який приносить додаткові бали або є умовою контракту.

Виховна робота: Кураторство груп, профорієнтація, волонтерство (актуально в умовах війни).

Міжнародний вимір реформи: Виклик CoARA та криза метрик

Найважливішою стратегічною зміною 2024-2025 років є поступове приєднання української наукової спільноти до Коаліції з удосконалення оцінювання дослідницької діяльності (CoARA). Цей процес створює фундаментальну методологічну колізію з існуючими практиками рейтингування.

Сутність конфлікту парадигм

Більшість українських рейтингових систем (як показано на прикладі ТНПУ та ЛПНУ) побудовані на жорсткому сциєнтизмі — вірі в те, що імпакт-фактор журналу (IF) та квартиль (Q1-Q4) є об'єктивними мірилами якості роботи вченого. Натомість, Угода CoARA, до якої вже приєдналися Інститут вищої освіти НАПН, НФДУ, Прикарпатський університет та інші , проголошує відмову від використання метрик журналів як сурогату якості індивідуального дослідження.

Основні принципи CoARA, що суперечать українській практиці:

Відмова від фетишизації індексів: Не можна оцінювати вченого лише за h-індексом, оскільки це стимулює самоцитування та "citation cartels".

Якісна експертиза: Пріоритет має надаватися peer-review (експертному оцінюванню) суті роботи, а не місцю її публікації.

Різноманітність результатів: Визнання цінності даних, програмного забезпечення, протоколів, а не лише статей.

Виклики імплементації в Україні

Імплементація CoARA в Україні наштовхується на серйозні перешкоди:

Відсутність культури експертизи: Перехід на якісне оцінювання вимагає великої кількості неупереджених експертів, яких в обмеженому академічному середовищі України (де "всі всіх знають") знайти важко. Кількісні метрики сприймаються як "менше зло", що захищає від кумівства.

Фінансова залежність: Державне фінансування науки часто прив'язане до формальних показників, тому відмова від них на рівні університету може призвести до втрати державного замовлення.

Ресурсна обмеженість: Якісне оцінювання — це дорого і довго. В умовах війни університети схильні до автоматизованих, швидких рішень (як система "Intellect" в КПІ), а не до створення складних експертних комісій.

Тим не менш, рух у цьому напрямку неминучий. Національний план щодо відкритої науки та дії НФДУ свідчать про те, що критерії фінансування будуть поступово змінюватися в бік принципів відповідального оцінювання.

Соціально-економічні наслідки та критичний дискурс

Впровадження рейтингових систем має глибокий вплив на соціальний клімат в університетах, умови праці та психологічний стан викладачів.

В умовах дефіциту бюджету рейтинги стають головним інструментом розподілу премій та надбавок. У Львівській політехніці, наприклад, від рейтингу залежить до 50% і більше доходу керівника кафедри, що транслюється на підлеглих. Це перетворює рейтингування на "гонитву за виживанням". Виникає явище прекаризації: контракти з викладачами укладаються на короткі терміни (1-3 роки), і їх продовження напряму залежить від рейтингових балів. Це створює постійний стан стресу та невпевненості у майбутньому.

Однією з найгостріших проблем 2024-2025 років є ініціатива щодо збільшення навчального навантаження на ставку з 18 до 22 годин (і потенційно до 36 годин загального робочого часу). Профспілка працівників освіти і науки України виступає категорично проти цього, аргументуючи, що це призведе до масових звільнень (до 70 тис. осіб) та фізичної неможливості виконувати наукову роботу, необхідну для рейтингу. Виникає парадокс: від викладача вимагають високих наукових рейтингів (статті Q1, гранти), але одночасно збільшують аудиторне навантаження, не залишаючи часу на дослідження. Оплата праці при цьому не зростає пропорційно навантаженню.

Тиск рейтингових показників ("Закон Гудхарта": коли показник стає ціллю, він перестає бути хорошим показником) призводить до негативних адаптивних стратегій:

Хижацькі журнали: Публікація у виданнях, які за гроші друкують будь-що, аби формально виконати вимогу Scopus.

Інфляція оцінок: Викладачі можуть завищувати оцінки студентам, щоб отримати кращий рейтинг "якості викладання" в анкетах.

"Салямі-слайсінг": Розбиття одного дослідження на кілька дрібних статей для збільшення кількості публікацій.

Існує значний дисбаланс між оцінкою "ресурсомістких" результатів (гранти, Scopus) та внутрішньої роботи (методика, робота зі студентами). Це призводить до маргіналізації викладачів, які є блискучими педагогами, але не мають високих наукометричних показників. Критики зазначають, що сучасні рейтинги не враховують "бренд" викладача та якість управління знаннями всередині кафедри.

Проведений аналіз дозволяє стверджувати, що рейтингове оцінювання НПП в Україні є незворотним процесом, який трансформує саму сутність академічної професії. Воно перетворило університети з інертних структур на динамічні корпорації, де кожен працівник є капіталом, що має приносити вимірюваний результат. Однак поточна модель, орієнтована на валові кількісні показники, наближається до межі своєї ефективності та потребує якісного перезавантаження.

Майбутнє українських систем оцінювання лежить у площині переходу від кількісного диктату до збалансованої оцінки профілів.

Ключові вектори розвитку:

Диверсифікація кар'єрних треків: Університети мають офіційно запровадити різні профілі НПП (наприклад, Teaching Track, Research Track, Management Track). Рейтингова система повинна мати різні вагові коефіцієнти для кожного треку. Для "дослідника" головним має бути Scopus та гранти, для "викладача" — підручники, методики та відгуки студентів. Це зніме напругу від необхідності "вміти все".

Інтеграція CoARA: Відмова від автоматичного нарахування балів за імпакт-фактор журналу на користь оцінки топ-3 кращих досягнень викладача за рік (наративне резюме), які підтверджуються експертною комісією університету.

Захист академічного часу: Врахування реальних часових витрат на виконання рейтингових показників при плануванні навантаження. Зменшення аудиторних годин для лідерів рейтингу як форма нематеріального заохочення.

Для ректорів та проректорів: Переглянути Положення про рейтингування у світлі стандартів CoARA. Ввести мораторій на збільшення вимог до публікаційної активності без відповідного фінансового забезпечення досліджень. Забезпечити повну цифровізацію збору даних (інтеграція репозитаріїв з ORCID), щоб мінімізувати час викладачів на звітність.

Для HR-відділів та відділів якості: Розробити системи психологічної підтримки та профілактики вигорання для викладачів, які перебувають під тиском рейтингових вимог. Проводити тренінги з академічної доброчесності та написання грантових заявок, надаючи інструменти для досягнення цілей, а не лише вимагаючи результатів.

Для профспілок: Активно лобіювати включення рейтингових показників у колективні договори, чітко визначаючи "ціну" кожного балу та гарантії оплати за додаткове навантаження.

Підсумок: Рейтингове оцінювання — це потужна зброя в руках університетського менеджменту. Вона може як мобілізувати колектив на досягнення світового рівня, так і зруйнувати внутрішню мотивацію через бюрократичний тиск. Мистецтво управління сучасним українським університетом полягає у знаходженні тонкого балансу між вимогами метрик та збереженням академічної свободи і людського потенціалу в надскладних умовах сьогодення.

1.2 Стан, проблеми та перспективи автоматизації рейтингового оцінювання науково-педагогічних працівників

Рейтингове оцінювання науково-педагогічних працівників (НПП) трансформувалося з рутинної адміністративної процедури в стратегічний інструмент забезпечення якості освіти, стимулювання наукової продуктивності та прозорого розподілу фінансових ресурсів. В умовах переходу до моделі "Університет 4.0", де дані стають новим активом, автоматизація процесів збору, обробки та аналізу показників діяльності викладачів набуває критичного значення.

Актуальність автоматизації рейтингування підсилюється необхідністю інтеграції українських закладів вищої освіти (ЗВО) до Європейського простору вищої освіти (EHEA) та Європейського дослідницького простору (ERA). Вимоги до прозорості, академічної доброчесності та об'єктивності оцінювання, які висуваються міжнародними партнерами та грантодавцями, неможливо задовольнити в рамках застарілих "паперових" технологій. Відтак, впровадження комплексних інформаційно-аналітичних систем (IAC) стає не просто питанням технологічної модернізації, а умовою інституційної виживаності та розвитку.

Глобальний ринок систем управління інформацією про студентів (Student Information Systems - SIS), які часто включають модулі для оцінювання персоналу, демонструє стійку тенденцію до зростання. За аналітичними даними, обсяг цього ринку у 2024 році оцінювався в 11,29 млрд доларів США, а до 2034 року прогнозується його зростання до 70,72 млрд доларів із середньорічним темпом приросту (CAGR) на рівні 20,16%. Таке динамічне зростання свідчить про те, що заклади освіти по всьому світу інвестують значні ресурси в цифрові рішення для оптимізації адміністративних та академічних процесів, прагнучи замінити фрагментарні застарілі інструменти на інтегровані хмарні платформи.

Цей звіт пропонує глибокий аналіз поточного стану автоматизації рейтингового оцінювання НПП, базуючись на порівнянні вітчизняних практик (КПІ ім. Ігоря Сікорського, НУ "Львівська політехніка", СумДУ, ХНУРЕ) та передового зарубіжного досвіду використання спеціалізованих систем класу CRIS (Current Research Information Systems), таких як Elsevier Pure, Symplectic Elements та DSpace-CRIS.

Впровадження автоматизованих систем рейтингування не є самоціллю, а відображає зміну управлінської парадигми. Традиційні методи оцінювання, що базувалися на епізодичних перевірках та суб'єктивних характеристиках, замінюються системами безперервного моніторингу ключових показників

ефективності (KPI). Автоматизація дозволяє реалізувати поліфункціональну модель оцінювання:

Мотиваційна функція: Забезпечення прямого та прозорого зв'язку між результатами діяльності викладача та його матеріальною винагородою. Як свідчить досвід Національного університету «Львівська політехніка», інтегральний рейтинг структурних підрозділів (кафедр, інститутів) безпосередньо впливає на розрахунок фонду матеріального заохочення та встановлення надбавок керівному складу, що створює каскадну систему мотивації зверху донизу.

Стратегічна функція: Використання агрегованих даних для прийняття рішень щодо кадрової політики, відкриття нових напрямів досліджень або оптимізації навантаження. Аналіз ринку показує, що 74,23% доходу ринку SIS припадає на програмне забезпечення, орієнтоване на централізацію академічних та адміністративних операцій, що підтверджує попит на стратегічну аналітику.

Репутаційна функція (Showcase): Трансформація внутрішніх даних про успішність у публічні профілі дослідників. Сучасні системи, такі як Elsevier Pure Portal або Discovery Module у Symplectic Elements, автоматично генерують веб-сторінки вчених, візуалізуючи їхні зв'язки та здобутки для глобальної спільноти, що є критичним для міжнародних рейтингів та грантової активності.

Адміністративна функція (Tenure & Promotion): У системах вищої освіти США, зокрема в університетах Ліхай (Lehigh University) та Іллінойс (University of Illinois), автоматизовані системи забезпечують процеси щорічного огляду (Annual Review) та пост-теньюр огляду (Post-Tenure Review), формуючи доказову базу для продовження контрактів або надання пожиттєвого найму.

Архітектура будь-якої автоматизованої рейтингової системи базується на тріаді академічної діяльності: навчальна, наукова та організаційна робота. Однак, методи збору даних для цих компонентів суттєво відрізняються, що створює технічні виклики для автоматизації.

Таблиця 1.1 – Структура рейтингу

Компонент рейтингу	**Типові індикатори**	**Джерела автоматизованого збору (Harvesting)**	**Проблеми автоматизації**
Наукова діяльність	Публікації (статті, монографії), цитування, h-індекс, участь у конференціях	Scopus, Web of Science, Google Scholar, ORCID, Crossref	Проблеми ідентифікації авторів (Author ID), дублювання профілів, затримка індексації.[5]
Навчально-методична робота	Аудиторне навантаження, розробка силабусів, дистанційні курси, керівництво дипломами	АСУ "Деканат", LMS Moodle, репозитарії університетів	Складність оцінки якості методичних матеріалів, необхідність інтеграції з розкладом та кадровими системами.[10]
Організаційно-виховна робота	Кураторство, профорієнтація, участь у вчених радах, волонтерство	Внутрішні накази, системи електронного документообігу, Self-reporting	Висока частка ручного введення даних через відсутність цифрових слідів для багатьох видів активності.[12]

Особливістю сучасних систем є перехід від "самозвіту" (self-reporting), коли викладач вручну вносить усі дані, до "верифікації", коли система автоматично збирає дані з зовнішніх джерел, а викладач лише підтверджує їх коректність.

Зарубіжні університети, особливо у країнах з розвиненою дослідницькою культурою (Велика Британія, США, Австралія), використовують спеціалізовані системи класу Research Information Management Systems (RIMS) або CRIS. Ринок цих рішень консолідований навколо декількох ключових гравців, які пропонують комплексні екосистеми управління даними.

Система Pure від корпорації Elsevier є одним із лідерів ринку, особливо в Європі та Великій Британії. Її використовують такі провідні інституції, як Оксфордський університет, Університет Абердіна, Університет Бата та багато інших.

Функціональна архітектура:

Автоматизований збір даних: Ключовою перевагою Pure є глибока інтеграція з базою даних Scopus (також продукт Elsevier). Нові публікації автоматично імпортуються до профілю дослідника щотижня, що мінімізує необхідність ручного введення. Це вирішує проблему актуальності даних та знижує адміністративне навантаження на вчених.

Fingerprint Engine™: Унікальна технологія на базі штучного інтелекту та обробки природної мови (NLP), яка аналізує тексти публікацій дослідника та автоматично створює індекс зважених концепцій (Concepts). Це дозволяє сформувати унікальний "відбиток" наукових інтересів, який використовується для пошуку експертів, потенційних колаборatorів та візуалізації компетенцій університету.

Модульність: Система складається з ядра та додаткових модулів, таких як Award Management (управління грантами протягом усього життєвого циклу), Assessment Module (підготовка до національних оцінювань, наприклад, британського REF) та Pure Portal (публічний інтерфейс).

Репутаційний менеджмент: Pure Portal забезпечує візуально привабливе представлення досягнень університету для зовнішньої аудиторії. Дослідники можуть демонструвати свої метрики (включаючи альтернативні метрики Altmetric), мережі співпраці та повні тексти публікацій, що сприяє відкритій науці.

Кейс впровадження: Університет Оксфорда використовує Pure для створення порталу "Research@Oxford", який дозволяє зовнішнім стейкхолдерам легко знаходити експертів за специфічними темами, переглядати географію співпраці та аналізувати вплив досліджень на суспільство.

Symplectic Elements (продукт компанії Digital Science) є головним конкурентом Pure, займаючи сильні позиції в США (Duke University, Indiana University, Carnegie Mellon) та Великій Британії (Cambridge, Imperial College London).

Ключові особливості:

Агрегація з багатьох джерел: На відміну від Pure, яка має преференцію до Scopus, Elements позиціонується як нейтральна платформа, що збирає дані з найширшого спектра джерел: Web of Science, PubMed, Crossref, arXiv, RePEc, SSRN та інших. Це дозволяє створити більш повний профіль дослідника, особливо в гуманітарних та соціальних науках.

Зменшення адміністративного тягаря: Система спроектована навколо принципу "enter once, use many times" (введи один раз, використовуй багатократно). Дані про публікації, гранти та викладання автоматично використовуються для генерації CV, річних звітів (Faculty Activity Reports) та профілів на вебсайтах.

Assessment Module: Потужний інструмент для автоматизації процесів внутрішнього оцінювання та атестації. Він дозволяє налаштовувати гнучкі маршрути погодження (workflows), де викладач подає звіт, а рецензенти та керівники можуть залишати коментарі, виставляти оцінки та затверджувати результати в єдиному інтерфейсі. Це замінює обмін електронними листами та паперовими формами.

Discovery Module: Забезпечує створення сучасних, оптимізованих для пошукових систем (SEO) публічних профілів, які інтегрують дані про публікації з інформацією про доступність обладнання, наукові групи та можливості для менторства.

Кейс впровадження: Університет Індіани обрав Symplectic Elements у 2024 році для заміни попередньої системи (Watermark DMAI), щоб покращити зручність використання (UX) та забезпечити кращу інтеграцію з різноманітними школами та коледжами в межах своєї мульти-кампусної структури.

Компанія Watermark (раніше Digital Measures) займає значну частку ринку в США, обслуговуючи понад 3500 інституцій. Система Faculty Success спеціалізується на процесах, специфічних для американської академічної культури.

Специфіка функціоналу:

Акредитаційна звітність: Система містить вбудовані шаблони для генерації звітів для провідних акредитаційних агентств (ABET, AACSB, CAEP). Це дозволяє університетам автоматично формувати необхідну документацію про кваліфікацію викладачів, що є критичним для збереження ліцензії.

Tenure & Promotion: Автоматизація складних та чутливих процесів розгляду кандидатур на отримання пожиттєвого контракту (tenure). Система дозволяє оцифрувати збір досьє, рецензування комітетами та голосування, забезпечуючи конфіденційність та дотримання процедурних вимог.

Rapid Reports: Інструмент для швидкого створення звітів та CV у різних форматах, необхідних для грантових заявок (наприклад, формат NIH Biosketch), що економить час дослідників.

Для університетів, які прагнуть зберегти повний контроль над даними та уникнути високих ліцензійних платежів, альтернативою є DSpace-CRIS — рішення з відкритим вихідним кодом (Open Source), що розширює можливості популярного репозитарію DSpace.

Технічні переваги:

Гнучка модель даних: DSpace-CRIS дозволяє створювати та налаштовувати будь-які сутності (Дослідники, Організації, Проєкти, Гранти, Обладнання) та зв'язки між ними, що забезпечує високу адаптивність до специфічних потреб університету.

CERIF-сумісність: Система базується на європейському стандарті CERIF (Common European Research Information Format), що гарантує інтероперабельність та можливість обміну даними з іншими європейськими системами.

Інтеграція з ORCID: Реалізована повна двостороння синхронізація (Push/Pull) з реєстрами ORCID. Це дозволяє не лише імпортувати дані з ORCID в локальну систему, а й експортувати локальні дані в глобальний профіль дослідника, підвищуючи його видимість.

Вартість: Як Open Source продукт, DSpace-CRIS не вимагає плати за ліцензії, проте впровадження потребує наявності кваліфікованої технічної команди або залучення партнерів (наприклад, 4Science) для налаштування та підтримки.

Українські заклади вищої освіти демонструють значний прогрес у цифровізації, переходячи від паперових звітів до власних програмних розробок. Однак, на відміну від західних університетів, де домінують готові промислові рішення, в Україні переважає модель in-house розробки ("самописні" системи).

КПІ ім. Ігоря Сікорського реалізував одну з найбільш комплексних систем автоматизації на базі власної платформи «Електронний кампус».

Модуль «Рейтинг НПП»:

Нормативна база: Процес рейтингування регламентується "Положенням про рейтингування НПП", де чітко прописано обов'язок кожного викладача вносити інформацію про свою діяльність. Ця вимога є невід'ємною частиною трудового контракту, а ухилення від неї може мати дисциплінарні наслідки.

Процедура: Викладачі вносять дані через персональний кабінет. Система передбачає механізми самоконтролю та верифікації: після генерації рейтинг-листа він подається на затвердження завідувачу кафедри. Це забезпечує валідацію даних на рівні підрозділу.

Контроль доброчесності: Університет впровадив жорсткі механізми контролю достовірності даних. Виявлення фактів фальсифікації (наприклад, приписування чужих публікацій) є підставою для розгляду справи Комісією з етики та академічної доброчесності Вченої ради, що може призвести до розірвання контракту.

Технічна підтримка: Розробку та супровід системи здійснює внутрішній підрозділ — Конструкторське бюро інформаційних систем (КБ ІС). Це забезпечує незалежність від зовнішніх вендорів та можливість оперативно вносити зміни відповідно до рішень Вченої ради.

Компоненти рейтингу: Рейтинг формується за трьома напрямами: навчально-методична діяльність (якість викладання, силабуси), науково-інноваційна (публікації, гранти) та організаційно-виховна. Для кожного напряму розроблена детальна бальна система, яка регулярно оновлюється.

У Львівській політехніці система рейтингування глибоко інтегрована з фінансово-економічними механізмами управління.

Зв'язок з оплатою праці: Університет застосовує унікальну модель, де результати рейтингування прямо конвертуються у фінансові стимули. Встановлення надбавок та премій керівникам підрозділів (завідувачам кафедр, директорам інститутів) напряму залежить від позиції їхнього підрозділу в загальноуніверситетському рейтингу за попередній рік. Це створює сильну мотивацію для керівників забезпечувати високу продуктивність кожного працівника.

Інструментарій: Університет проводить дослідження та впровадження систем автоматизованого моніторингу робочого часу (Time Tracking) та аналітики активності. Розглядаються рішення, що дозволяють фіксувати часові витрати на виконання різних видів робіт, хоча питання приватності співробітників залишається предметом дискусій. Крім того, активно використовуються дані з міжнародних баз Scopus та Web of Science для формування рейтингів, що дозволяє університету займати високі позиції в міжнародних ранжуваннях (THE, QS).

Сумський державний університет: СумДУ є лідером у впровадженні процесного підходу до автоматизації. Особистий кабінет співробітника інтегрує різноманітні сервіси, від бібліотечних до фінансових. Особливістю є високий рівень автоматизації збору даних про навчальне навантаження безпосередньо з систем планування та розкладу, що зменшує потребу в ручному введенні. Університет розглядає автоматизацію як інструмент підвищення керованості підприємством та прозорості процесів.

Харківський національний університет радіоелектроніки (ХНУРЕ): Використовує інформаційну систему "Рейтинг", яка надає детальну

візуалізацію досягнень викладачів та кафедр. Система дозволяє формувати порівняльні таблиці та діаграми, що сприяє створенню здорової конкуренції всередині колективу.

Важливим кроком до уніфікації рейтингового оцінювання в масштабах країни є створення Національної електронної науково-інформаційної системи URIS (Ukrainian Research Information System).

Мета: Створення єдиного вікна доступу до інформації про наукову діяльність в Україні, інтеграція даних про дослідників, установи, проєкти та публікації.

Інтеграція: URIS розробляється з урахуванням необхідності інтеграції з локальними системами університетів та міжнародними реєстрами. Проте, існують виклики, пов'язані з фрагментацією даних, відсутністю єдиних класифікаторів наукових напрямів та необхідністю розробки модуля дослідницьких інфраструктур.

Перспективи: Повноцінне впровадження URIS дозволить автоматизувати звітність університетів перед міністерством, усунувши дублювання інформації та паперовий документообіг.

На основі проведеного аналізу можна класифікувати системи автоматизації рейтингування на три основні групи, кожна з яких має свої переваги та недоліки (табл. 1.2).

Таблиця 1.2 – Порівняльний аналіз архітектури та функціональності систем

Характеристика	**Власні розробки (In-house)**	**Комерційні CRIS (Pure, Elements)**	**Open Source (DSpace-CRIS)**
Поширення	Україна (КПІ, ЛП, СумДУ)	США, Велика Британія, ЄС	Глобально (бюджетні установи)

Характеристика	Власні розробки (In-house)	Комерційні CRIS (Pure, Elements)	Open Source (DSpace-CRIS)
Модель даних	Специфічна для конкретного ЗВО, часто жорстка	Гнучка, базується на міжнародних стандартах	CERIF-сумісна, повністю налаштовувана
Збір даних	Переважно ручне введення (Self-reporting), частковий імпорт	Автоматичний харвестинг (Harvesting) з глобальних баз	Ручне введення + імпорт через API
Вартість	Високі витрати на розробку та утримання штату програмістів	Висока вартість ліцензії (SaaS), щорічна підписка	Безкоштовна ліцензія, витрати на впровадження
Інтероперабельність	Низька, складність обміну даними	Висока (вбудовані конектори)	Висока (відкриті API, OAI-PMH)
Публічні профілі	Часто відсутні або мають обмежений функціонал	Високоякісні портали з візуалізацією мереж	Інтегровані сторінки дослідників

Таблиця 1.3 – Порівняння підходів до автоматизації процесів оцінювання

Процес	Українська практика	Західна практика (Best Practices)
Ініціація	Наказ ректора, початок навчального року	Автоматичні нагадування системи, циклічність
Наповнення	Ручне заповнення форм викладачем	Автоматичний збір публікацій, грантів; викладач лише верифікує
Верифікація	Завідувач кафедри, кадрова комісія	Peer review, автоматичні алгоритми перевірки

Процес	Українська практика	Західна практика (Best Practices)
Результат	Рейтинг-лист, наказ про преміювання	CV для tenure, публічний профіль, аналітика для стратегії

Автоматизація не усуває людський фактор, а іноді створює нові можливості для маніпуляцій ("гейміфікація" системи).

Накрутка цитувань: Використання "сміттєвих" журналів або взаємне цитування для штучного підвищення показників, які автоматично підтягуються в рейтинг. Це вимагає впровадження додаткових фільтрів та якісної експертизи джерел даних.

Дублювання досягнень: Внесення однієї тези конференції як статті у фаховому виданні. Системи ручного введення особливо вразливі до таких зловживань.

Інституційна реакція: Університети змушені створювати етичні комісії для розгляду таких випадків, що перетворює процес рейтингування на квазі-судову процедуру.

Впровадження автоматизованих систем є дороговартісним процесом.

Вартість розробки: Аналіз тендерів в системі Prozorro та комерційних пропозицій показує, що вартість розробки кастомізованої CRM/ERP системи для університету в Україні може варіюватися від 15 000 до 100 000 доларів США і вище, залежно від складності. Щорічна підтримка також вимагає значних витрат бюджетних коштів.

Технічна інфраструктура: Необхідність у потужних серверах або хмарних сховищах, забезпечення захисту персональних даних (КСЗІ) та безперебійного доступу.

Впровадження рейтингових систем часто наштовхується на опір з боку персоналу.

Недовіра до алгоритмів: Викладачі можуть вважати критерії непрозорими або несправедливими, особливо якщо вони часто змінюються.

Страх контролю: Автоматизація сприймається як інструмент тотального контролю ("Великий брат"), а не допомоги.

Демотивація: Нечіткість критеріїв або орієнтація виключно на кількісні показники може призвести до демотивації та професійного вигорання.

Проведений аналіз дозволяє зробити наступні висновки:

Незворотність автоматизації: Рейтингове оцінювання НПП в Україні перейшло точку неповернення від паперових носіїв до цифрових систем. Лідери ринку (КПІ, Львівська політехніка, СумДУ) вже мають розвинені екосистеми, які інтегрують рейтингування з кадровими та фінансовими процесами.

Розрив у підходах: Існує концептуальний розрив між українськими системами, орієнтованими на внутрішній контроль та "ручне" управління, та західними CRIS-системами, які фокусуються на автоматичному зборі даних, промоції досягнень та стратегічній аналітиці.

Потреба в інтеграції: Відсутність єдиних стандартів обміну даними гальмує створення національного наукового простору. Проєкт URIS є важливим кроком, але його успіх залежить від готовності університетів відкрити свої дані через API.

Рекомендації для закладів вищої освіти:

Перехід до гібридних моделей: Замість розробки систем "з нуля", розглянути можливість використання Open Source рішень (DSpace-CRIS) або інтеграції власних кабінетів з API глобальних платформ (ORCID, Scopus) для автоматизації наповнення профілів.

Фокус на User Experience: Зменшити адміністративне навантаження на викладачів шляхом автоматизації збору даних про навчальне навантаження з LMS (Moodle) та АСУ "Деканат". Принцип "введення даних один раз" має стати ключовим.

Розвиток публічних профілів: Інвестувати в створення публічних порталів дослідників, які автоматично генеруються з рейтингової системи. Це

перетворить рейтингування з інструменту контролю на інструмент маркетингу наукового потенціалу університету.

Прозорість та комунікація: Забезпечити повну прозорість алгоритмів розрахунку рейтингу та залучати академічну спільноту до обговорення критеріїв оцінювання, щоб мінімізувати опір та підвищити довіру до системи.

Впровадження цих заходів дозволить українським університетам не лише оптимізувати внутрішні процеси, а й суттєво підвищити свою конкурентоспроможність на міжнародній арені.

1.3 Аналіз вимог та техніко-економічне обґрунтування впровадження автоматизованої системи рейтингового оцінювання науково-педагогічних працівників у Криворізькому національному університеті

На сьогоднішній день процес рейтингового оцінювання в КНУ, як і в багатьох інших українських університетах, характеризується значною часткою ручної праці, фрагментарністю даних та складністю верифікації показників. Згідно з чинним Положенням про рейтингове оцінювання , кожен науково-педагогічний працівник зобов'язаний особисто подавати інформацію про свою діяльність, що створює ризики суб'єктивізму та механічних помилок. Інформація про досягнення розпорошена між різними носіями: паперовими звітами, локальними файлами кафедр, базою даних автоматизованої системи управління (АСУ) «Деканат» та платформою дистанційного навчання Moodle. Відсутність єдиної інтегрованої екосистеми призводить до того, що адміністрація університету витрачає надмірні ресурси на збір та агрегацію статистики, замість того щоб фокусуватися на стратегічному аналізі та прийнятті управлінських рішень.

Важливість автоматизації підкреслюється також зовнішніми факторами. Позиціювання КНУ в національних рейтингах, таких як «ТОП-200 Україна», та міжнародних наукометричних базах Scopus і Web of Science безпосередньо залежить від активності викладачів. Ректорські звіти свідчать про позитивну

динаміку університету в отриманні патентів та публікаційній активності , проте підтримка та прискорення цього темпу вимагає інструментів оперативного моніторингу. Впровадження програмного забезпечення для автоматизації рейтингу дозволить трансформувати рейтингову систему з інструменту «посмертного» підбиття підсумків року на динамічну панель управління (dashboard), яка стимулюватиме викладачів до безперервного професійного розвитку.

Успіх впровадження будь-якої інформаційної системи залежить від того, наскільки точно вона відображає потреби та очікування всіх зацікавлених сторін (стейкхолдерів). В контексті КНУ можна виділити кілька ключових груп користувачів, кожна з яких має специфічні сценарії взаємодії з Системою.

Науково-педагогічні працівники (НПП)

Це найчисленніша група користувачів, яка включає асистентів, викладачів, старших викладачів, доцентів та професорів. Згідно зі штатним розписом та даними з відкритих джерел, чисельність академічного персоналу КНУ становить близько 900 осіб.[10]

- **Потреби:** Для НПП критично важливим є мінімізація часу, що витрачається на бюрократичні процедури. Викладачі зацікавлені в тому, щоб Система автоматично підтягувала дані про їхні публікації, методичні розробки та навчальне навантаження, позбавляючи необхідності ручного введення. Також важливим є прозорість нарахування балів: викладач повинен бачити, за що саме нараховано або знято бали, та мати можливість подати апеляцію.
- **Сценарії використання:** Вхід у систему через корпоративний акаунт Office 365 [11], перегляд автоматично згенерованого профілю, завантаження скан-копій сертифікатів конференцій, верифікація знайдених системою публікацій у Scopus, формування друкованої форми рейтингового листа для підписання (або накладання КЕП).

Завідувачі кафедр

Завідувачі кафедр виступають першою ланкою верифікації даних. У КНУ функціонують десятки кафедр, наприклад, кафедра автоматизації, комп'ютерних наук і технологій [12] або кафедри гірничо-металургійного факультету.

- **Потреби:** Інструменти для швидкої перевірки достовірності даних, поданих підлеглими. Завідувач має бачити зведену таблицю по кафедрі, динаміку виконання планових показників та мати важелі впливу (затвердження або відхилення записів).
- **Сценарії використання:** Отримання повідомлень про нові звіти, перегляд деталізації балів кожного співробітника, формування рейтингу кафедри, аналіз слабких місць (наприклад, недостатня кількість публікацій Q1/Q2) для коригування роботи кафедри.

Деканати факультетів

Декани та співробітники деканатів (факультети: інформаційних технологій, електротехнічний, гірничо-металургійний, механічної інженерії та транспорту тощо) відповідають за організацію навчального процесу та моніторинг показників на рівні структурного підрозділу.[7]

- **Потреби:** Агрегована аналітика по кафедрах, порівняння ефективності різних кафедр, контроль дотримання термінів подачі рейтингових форм (до 17 грудня для наукової роботи та до 17 червня для методичної).[13]
- **Сценарії використання:** Генерація звітів для вченої ради факультету, експорт даних для розподілу преміального фонду або надбавок, моніторинг кадрового потенціалу факультету.

Адміністрація університету (Ректорат, Центр забезпечення якості освіти)

Вище керівництво університету (Ректор, проректори) та спеціалізовані підрозділи, такі як Центр забезпечення якості вищої освіти [14], є основними споживачами аналітичних даних.

- **Потреби:** Стратегічне бачення розвитку університету, формування рейтингів для МОН, контроль виконання контрактів викладачами,

аналіз кореляції між рейтинговими показниками та позиціями університету в глобальних рейтингах.

- **Сценарії використання:** Перегляд KPI-дашбордів, налаштування вагових коефіцієнтів рейтингу (наприклад, збільшення ваги за статті в Scopus для стимулювання науки), прийняття кадрових рішень.

Технічний персонал (Адміністратори АСУ, ІОЦ)

Співробітники Інформаційно-обчислювального центру [15] відповідають за підтримку працездатності Системи.

- **Потреби:** Надійність, безпека, зручність адміністрування, можливості інтеграції з наявним зоопарком систем (Dekanat, Moodle, Office 365).

Аналіз бізнес-процесів та функціональна архітектура

Модель даних та інтеграційне середовище

КНУ вже має розгорнуту IT-інфраструктуру, яка включає кілька критично важливих компонентів. Створення ізольованої системи рейтингування буде помилкою; натомість пропонується архітектура, що базується на глибокій інтеграції.

Інтеграція з АСУ «Деканат»

Система управління навчальним процесом «Деканат» від компанії «Політек-СОФТ» широко використовується в КНУ для обліку студентів, навантаження та кадрів.[16]

- **Дані для імпорту:** Структура університету (факультети, кафедри), перелік НПП, їхні посади, наукові ступені, вчені звання, частка ставки (FTE), навчальне навантаження (виконані години).
- **Технічна реалізація:** Оскільки «Деканат» часто використовує бази даних типу Firebird або Interbase, необхідна розробка проміжного шлюзу (ETL-процесу), який щоночі синхронізуватиме кадрові дані з базою даних Системи рейтингування. Це забезпечить актуальність інформації про те, хто має право подавати рейтинг і в якій ролі.

Інтеграція з Moodle (LMS)

Платформа Moodle є основним середовищем для змішаного та дистанційного навчання в КНУ.[4]

- **Дані для автоматичного збору:**
  - Активність викладача: кількість створених курсів, оновлення матеріалів, активність на форумах, перевірка завдань.
  - Якість контенту: наявність силабусів, методичних вказівок, відеоматеріалів (через аналіз типів файлів у курсах).
- **Методика:** Розробка плагіна для Moodle або використання Moodle Web Services API для витягування статистики активності ("Logs") та її конвертації в рейтингові бали згідно з затвердженими критеріями методичної роботи.

Інтеграція з Microsoft 365

КНУ активно використовує сервіси Microsoft для корпоративної пошти та комунікації.[11]

- **Функція:** Single Sign-On (SSO).
- **Реалізація:** Використання протоколу OAuth 2.0 або SAML через Azure Active Directory. Це дозволить викладачам входити в Систему, використовуючи свої звичні облікові дані (email @knu.edu.ua), що значно підвищить рівень безпеки та зручність користування.

Функціональні модулі системи

Система повинна складатися з наступних функціональних модулів, що покривають повний цикл рейтингового оцінювання.

Модуль 1: Особистий кабінет та Портфоліо НПП

Центральний вузол взаємодії користувача з системою.

- **Профіль:** Відображення особистих даних (імпортованих з «Деканату»), ідентифікаторів ORCID, Scopus Author ID, ResearcherID. Можливість редагування контактної інформації та наукових інтересів.
- **Дашборд:** Візуалізація поточного рейтингу, порівняння з середнім по кафедрі/факультету, прогрес виконання індивідуального плану.

Модуль 2: Наукова діяльність (Research Activity)

Цей модуль вимагає найвищого рівня автоматизації, оскільки наукові результати є найбільш структурованими даними.

- **Публікації:** Автоматичний пошук статей у Scopus та Web of Science за допомогою API (Elsevier Scopus API, Clarivate API). Система повинна автоматично визначати квартиль журналу (Q1-Q4), кількість цитувань та розраховувати бали.
    - *Проблема:* Не всі журнали індексуються.
    - *Рішення:* Можливість ручного додавання статей у фахових виданнях України (категорія Б) з обов'язковим додаванням DOI або посилання на репозитарій бібліотеки.
- **Конференції:** Форма для внесення даних про участь у конференціях (назва, дата, рівень: міжнародна/всеукраїнська). Врахування ролі (доповідач/слухач).
- **Наукове керівництво:** Облік підготовки аспірантів та здобувачів (дані можуть частково братися з наказів).
- **НДР та Гранти:** Внесення інформації про участь у держбюджетних або госпрозрахункових темах, міжнародних грантах (Erasmus+, Horizon Europe).[19]

Модуль 3: Навчально-методична робота (Teaching & Methodology)

- **Навчальне навантаження:** Автоматичний імпорт виконання навантаження з АСУ «Деканат».
- **Методичне забезпечення:** Інтеграція з Moodle для автоматичного зарахування балів за електронні курси. Ручне внесення даних про видані підручники, посібники (з завантаженням PDF для підтвердження або посиланням на репозитарій).
- **Якість викладання:** Можливість інтеграції результатів студентського анкетування (якщо таке проводиться в цифровому вигляді) як поправочного коефіцієнта.[21]

Модуль 4: Організаційно-виховна робота (Organizational Work)

- **Кураторство:** Автоматичне підтягування даних про кураторство груп з «Деканату».
- **Профорієнтація:** Звітування про відвідування шкіл, проведення днів відкритих дверей.
- **Громадська активність:** Участь у вчених радах, експертних комісіях, редколегіях.

Модуль 5: Адміністрування та Аналітика

- **Конструктор формул:** Гнучкий інструмент для налаштування вагових коефіцієнтів та балів за різні види діяльності без необхідності перепрограмування системи (адже Положення може змінюватися щорічно).
- **Звітність:** Генерація зведених звітів по кафедрі, факультету, університету. Експорт у Excel, PDF. Аналіз KPI (наприклад, % викладачів з h-index > 0).

Детальний опис алгоритмів розрахунку рейтингу

На основі аналізу доступних фрагментів методик оцінювання (зокрема [22]), пропонується наступний підхід до алгоритмізації розрахунків, який має бути закладений в ядро Системи.

Базова формула рейтингу

Індивідуальний рейтинговий показник (R_i) науково-педагогічного працівника формується як сума зважених балів за різними напрямами діяльності, нормована на частку ставки або штатну одиницю.

Загальна формула може бути представлена як:

$$R_i = \frac{\alpha \cdot B_{edu} + \beta \cdot B_{meth} + \gamma \cdot B_{sci} + \delta \cdot B_{org} + \epsilon \cdot B_{qual}}{K_{fte}},$$

де:

- Bedu – бали за навчальну роботу (навантаження).
- Bmeth – бали за методичну роботу (підручники, Moodle, методички).
- Bsci – бали за наукову роботу (статті, монографії, конференції, патенти).
- Borg – бали за організаційну роботу (кураторство, адміністрування).

- Bqual – бали за підвищення кваліфікації та стажування.
- α, β, γ, δ, ε – вагові коефіцієнти пріоритетності (наприклад, для дослідницького університету γ може бути вищим).
- Kfte – коефіцієнт частки ставки (Full-Time Equivalent). Наприклад, для 1.0 ставки K=1, для 0.5 ставки K=0.5 (або інша логіка, якщо певні види робіт не залежать від ставки).

Деталізація показників (Критерії оцінювання)

Система повинна підтримувати довідник критеріїв з налаштовуваною "вартістю" в балах. Нижче наведено орієнтовну структуру довідника на основі аналізу типових положень українських ЗВО та специфіки КНУ.

Таблиця 1.3 – Деталізація показників

Категорія	**Показник (Вид діяльності)**	**Одиниця виміру**	**Орієнтовні бали**	**Примітка до автоматизації**
Наукова	Стаття у виданні, що індексується в Scopus/WoS (Q1-Q2)	1 стаття	100-150	Автоматично через API (перевірка ISSN/DOI)
Наукова	Стаття у виданні, що індексується в Scopus/WoS (Q3-Q4)	1 стаття	80-100	Автоматично через API
Наукова	Стаття у фаховому виданні України (Категорія Б)	1 стаття	20-40	Ручне введення + верифікація
Наукова	Індекс Гірша (h-index) Scopus > 0	За одиницю	10 * h-index	Автоматично [24]
Наукова	Отримання патенту на винахід	1 патент	50	Ручне введення (номер патенту)
Наукова	Участь у міжнародній конференції (за кордоном)	1 доповідь	30	Ручне введення + скан програми
Методична	Видання підручника з грифом МОН / Вченої ради	1 друк. арк.	40	Ручне введення

Категорія	Показник (Вид діяльності)	Одиниця виміру	Орієнтовні бали	Примітка до автоматизації
Методична	Розробка дистанційного курсу в Moodle (повний комплекс)	1 курс	50	Інтеграція з Moodle (перевірка структури)
Методична	Оновлення методичного забезпечення (силабус, РПНД)	1 дисципліна	10	Підтвердження зав. кафедрою
Організац.	Виконання обов'язків куратора академічної групи	1 рік	20	Імпорт з «Деканату»
Організац.	Робота у приймальній комісії / Профорієнтація	10 годин	5	Звіт про відрядження / табель

Особливості розрахунку для співавторства

Система повинна коректно обробляти роботи, виконані у співавторстві.

- **Алгоритм:** Загальна кількість балів за публікацію або патент ділиться на кількість співавторів.
- **Опція:** Можливість налаштування, чи враховуються тільки співавтори-співробітники КНУ, чи всі співавтори. Зазвичай, ділення відбувається на загальну кількість авторів, незалежно від їх афіліації.

Рейтинг кафедр та факультетів

Рейтинг структурного підрозділу розраховується як середнє арифметичне рейтингів його штатних працівників або як сума балів, нормована на штатну чисельність.[13]

$$R_{dept} = \frac{\sum R_{i}}{N_{staff}}$$

Де N_{staff} — кількість штатних одиниць кафедри. Це дозволяє порівнювати великі та малі кафедри.

Вимоги до інтерфейсу та юзабіліті (UX)

- **Адаптивність:** Інтерфейс повинен коректно відображатися на десктопних моніторах (для роботи адміністраторів та зав. кафедр) та мобільних пристроях (для перегляду рейтингу викладачами).
- **Локалізація:** Основна мова інтерфейсу — українська.
- **Доступність:** Відповідність стандартам WCAG 2.1 (контрастність, підтримка скрінрідерів) для забезпечення інклюзивності.

Вимоги до безпеки та захисту даних

Система оброблятиме персональні дані (ПІБ, посади) та інформацію, що може впливати на фінансове забезпечення (премії), тому безпека є пріоритетом.

- **Аутентифікація:** Інтеграція з Azure Active Directory (Microsoft 365) для централізованого управління доступом. Заборона використання слабких локальних паролів.
- **Рольова модель доступу (RBAC):** Чітке розмежування прав:
  - *НПП:* Тільки читання/редагування власних даних.
  - *Зав. кафедри:* Читання/редагування даних кафедри.
  - *Декан:* Читання даних факультету.
  - *Адміністратор:* Повний доступ.
- **Захист каналів зв'язку:** Використання протоколу HTTPS (TLS 1.2/1.3) для шифрування трафіку.
- **Журналювання (Audit Logging):** Фіксація всіх змін у системі (хто, коли, яку оцінку змінив) для запобігання фальсифікаціям.

Рекомендований технологічний стек

Враховуючи необхідність інтеграції з існуючими системами та вимоги до масштабованості, пропонується наступний стек:

- **Backend:**
  - Мова: PHP (фреймворк Laravel або Symfony) — забезпечує легку інтеграцію з Moodle (написаний на PHP) та має потужні бібліотеки для роботи з Enterprise-системами. Альтернатива: Python (Django) для зручності роботи з аналітикою даних.

- API: RESTful API або GraphQL для взаємодії фронтенду з бекендом.

- **Frontend:**
  - JavaScript фреймворк: React.js або Vue.js. Забезпечують створення швидкого та інтерактивного інтерфейсу (Single Page Application).
- **База даних:**
  - Основна БД: PostgreSQL або MySQL. Реляційна модель ідеально підходить для структурованих даних рейтингу.
  - Кешування: Redis для прискорення завантаження звітів.
- **Хостинг та інфраструктура:**
  - Сервери КНУ (On-Premise) або хмарне середовище (наприклад, Azure, враховуючи партнерство з Microsoft).
  - Контейнеризація (Docker) для спрощення розгортання та оновлення.

Специфіка інтеграції з АСУ «Деканат» (Politek-SOFT)

Оскільки «Деканат» є пропрієтарним ПЗ, прямий доступ до його бази даних може бути обмеженим. Рекомендується:

1. Узгодити з розробником («Політек-СОФТ») можливість налаштування "View" (представлень) у базі даних для читання кадрової інформації.
2. Якщо прямий доступ неможливий, налаштувати регулярний експорт XML/CSV файлів з «Деканату» в захищену папку, звідки Система рейтингування буде їх імпортувати (файловий обмін).

План реалізації та дорожня карта (Roadmap)

Впровадження системи такого масштабу вимагає поетапного підходу.

Етап 1: Підготовка та проєктування (1-2 місяці)

- Створення робочої групи (представники ректорату, ІТ-відділу, відділу кадрів).
- Формалізація методики рейтингування (узгодження всіх коефіцієнтів та формул).
- Технічний аудит баз даних «Деканату» та Moodle.

- Розробка детального технічного завдання (ТЗ) та прототипів інтерфейсу (UI/UX).

Етап 2: Розробка MVP (3-4 місяці)

- Розробка ядра системи (БД, бекенд).
- Реалізація модуля «Особистий кабінет» та ручного введення даних.
- Налаштування інтеграції з Microsoft 365 (SSO).
- Реалізація базових звітів.

Етап 3: Інтеграція та розширення функціоналу (3-4 місяці)

- Розробка конекторів до Scopus/WoS API.
- Розробка модуля інтеграції з Moodle (отримання статистики активності).
- Налаштування імпорту з АСУ «Деканат».
- Реалізація модуля аналітики та дашбордів для керівництва.

Етап 4: Пілотне впровадження (2 місяці)

- Запуск системи на базі одного факультету (наприклад, Факультету інформаційних технологій).
- Збір зворотного зв'язку від користувачів, виправлення помилок.
- Навантажувальне тестування.

Етап 5: Повномасштабний запуск та підтримка

- Розгортання на весь університет.
- Навчання користувачів (вебінари, інструкції).
- Технічна підтримка та подальший розвиток (додавання нових функцій).

Таблиця 1.4 – Ризики та стратегії їх мінімізації

Ризик	**Ймовірність**	**Вплив**	**Стратегія мінімізації**
Опір персоналу (небажання користуватися новою системою)	Висока	Високий	Проведення роз'яснювальної роботи, акцент на спрощенні роботи (автоматизація замість паперів), зручний інтерфейс.
Технічні проблеми інтеграції (зміни в	Середня	Високий	Гнучка архітектура, використання проміжних

Ризик	Ймовірність	Вплив	Стратегія мінімізації
API Scopus, закритість «Деканату»)			форматів обміну даними, тісна співпраця з вендорами.
Некоректність вхідних даних (помилки в назвах кафедр, дублі користувачів)	Висока	Середній	Реалізація процедур очищення даних (Data Cleansing) на етапі міграції, інструменти для об'єднання дублікатів.
Захист персональних даних (витік інформації)	Низька	Критичний	Дотримання стандартів КСЗІ, регулярні аудити безпеки, шифрування, мінімізація збережених даних.

Розробка та впровадження автоматизованої системи рейтингового оцінювання НПП у Криворізькому національному університеті є своєчасним та необхідним кроком. Це дозволить:

1. **Підвищити прозорість та об'єктивність** оцінювання персоналу.
2. **Зменшити адміністративне навантаження** на викладачів та завідувачів кафедр шляхом автоматизації рутинних процесів збору даних.
3. **Покращити позиції університету в рейтингах** завдяки оперативному моніторингу ключових показників ефективності (публікації, цитування, патенти).
4. **Створити єдиний інформаційний простір**, інтегрувавши розрізнені системи (Moodle, Dekanat, Office 365) в єдину екосистему управління якістю освіти.

Рекомендується розпочати проєкт з детального аудиту наявних даних та затвердження оновленого «Положення про рейтингове оцінювання», адаптованого під можливості автоматизації (наприклад, перехід до чітких кількісних метрик там, де це можливо). Технічну реалізацію доцільно виконувати з використанням сучасних веб-технологій та хмарних сервісів, що забезпечить масштабованість та надійність системи на роки вперед.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ РОЗРАХУНКУ РЕЙТИНГУ НАУКОВО-ПЕДАГОГІЧНИХ ПРАЦІВНИКІВ

2.1 Аналіз існуючої інфраструктури та обмежень

Критичним фактором успішного впровадження АСРО є її безшовна інтеграція з наявними системами університету. Аналіз інфраструктури КНУ виявляє наступні ключові компоненти, що впливають на архітектурні рішення:

По-перше, університет активно використовує хмарні сервіси Microsoft. Викладачам та студентам надається доступ до пакетів Office 365 A3 та A5. Це означає наявність розгорнутого Azure Active Directory (Azure AD), що є фундаментом для побудови єдиної системи автентифікації (Single Sign-On). Ліцензія A3 включає інструменти для керування й захисту, а A5 додає розширену аналітику та відповідність вимогам безпеки, що дозволяє використовувати ці можливості для захисту даних рейтингу. Ігнорування цього ресурсу призвело б до необхідності створення дублюючої системи облікових записів, що є архітектурною помилкою.

По-друге, управління контингентом студентів та навчальним навантаженням здійснюється через АСУ «Деканат» (розробник ПП «Політек-СОФТ»). Ця система є джерелом майстер-даних про кадрову структуру (кафедри, факультети, посади) та навчальну роботу. Однак, доступність API цієї системи часто є обмеженою або специфічною (базується на закритих протоколах або прямій взаємодії з БД Firebird/Interbase), що вимагає розробки спеціалізованого адаптера або використання файлового обміну для інтеграції.

По-третє, навчальний процес підтримується платформою Moodle, яка містить дані про розроблені дистанційні курси, активність викладачів та, потенційно, результати анкетування студентів. Інтеграція Moodle з іншими системами, такими як SIS (Student Information Systems), є стандартною практикою, але вимагає правильного налаштування токенів та веб-сервісів.

По-четверте, наукова діяльність оцінюється за показниками у міжнародних базах. Доступ до API Scopus для установ можливий, але регулюється політикою Elsevier щодо підписок та інституційних токенів. Водночас, Google Scholar не надає офіційного API, що створює значний технічний виклик для автоматизованого збору цитувань та вимагає використання технік скрапінгу (scraping) або сторонніх сервісів.

2.2 Визначення вимог та архітектурних драйверів

Проектування архітектури починається з чіткої формалізації вимог, які поділяються на функціональні (що система робить) та нефункціональні (як система це робить), або атрибути якості.

2.2.1 Функціональні вимоги

Система повинна забезпечувати повний цикл роботи з рейтинговими показниками: від їх виникнення (публікація статті, проведення лекції) до врахування у підсумковому балі.

Управління профілем та ідентифікація: Система повинна автоматично створювати профілі користувачів на основі даних з Azure AD та АСУ «Деканат». Користувач повинен мати можливість прив'язати свої ідентифікатори у наукометричних базах (Scopus Author ID, ORCID iD, ResearcherID, посилання на профіль Google Scholar). Критично важливою є валідація приналежності цих ідентифікаторів конкретній особі для уникнення приписування чужих заслуг.

Автоматизований збір даних (Data Harvesting): АСРО повинна періодично опитувати зовнішні API (Scopus, WoS) для виявлення нових публікацій афілійованих з університетом авторів. Для Google Scholar необхідно реалізувати механізм парсингу профілів, що здатен обходити захист від ботів (CAPTCHA) та коректно інтерпретувати неструктуровані дані. Система має розрізняти типи публікацій (стаття, тези конференції, монографія) та квартилі журналів, оскільки від цього залежить кількість балів.

Введення та верифікація ручних даних: Значна частина роботи НПП (виховна робота, профорієнтація, видання внутрішніх методичних вказівок) не відображається у глобальних базах даних. Система повинна надавати зручний інтерфейс для ручного введення таких досягнень із обов'язковим додаванням підтверджуючих документів (скан-копії, посилання). Ці записи повинні проходити процес верифікації відповідальною особою (завідувачем кафедри, заступником декана з наукової роботи).

Модуль анкетування «Очі студента»: Відповідно до сучасних стандартів забезпечення якості освіти, думка студентів є вагомим компонентом оцінки викладача. Система повинна дозволяти студентам анонімно оцінювати якість викладання дисциплін. Цей модуль має бути інтегрований з даними про запис студентів на курси, щоб студент міг оцінити лише тих викладачів, у яких він реально навчався.

Розрахунок рейтингу (Rating Engine): Ядром системи є модуль розрахунку, який застосовує затверджену Вченою радою формулу до зібраних даних. Враховуючи, що методика оцінювання може змінюватися щороку , модуль розрахунку повинен бути реалізований як гнучкий конструктор формул, а не жорстко закодований алгоритм. Це дозволить адміністраторам змінювати вагові коефіцієнти та додавати нові критерії без залучення розробників.

2.2.2 Атрибути якості (Non-Functional Requirements)

Модифікованість (Modifiability): Здатність системи адаптуватися до змін у регламенті рейтингування є ключовою. Архітектура повинна підтримувати версійність методик оцінювання, щоб зберігати історичні дані розрахованими за правилами, що діяли на момент оцінювання, і паралельно проводити розрахунки за новими правилами.

Інтероперабельність (Interoperability): Система повинна безшовно обмінюватися даними з різнорідними системами (Legacy "Dekanat", Cloud Moodle, External REST APIs). Це вимагає використання стандартних протоколів (HTTP/REST, SOAP, LTI) та форматів даних (JSON, XML).

Надійність та точність даних (Data Integrity): Помилка в рейтингу може вплинути на фінансове стимулювання або продовження контракту викладача. Система повинна мати механізми захисту від дублювання даних (наприклад, коли одна стаття індексується і в Scopus, і в WoS), а також журнал аудиту всіх змін, внесених користувачами або адміністраторами.

Продуктивність та масштабованість: Збір даних з зовнішніх джерел є ресурсоємною операцією. Синхронізація профілів тисяч викладачів не повинна блокувати роботу користувацького інтерфейсу. Це диктує необхідність використання асинхронної обробки даних та черг повідомлень.

2.3 Концептуальна архітектура системи

Для вирішення поставлених завдань пропонується використати **сервіс-орієнтовану архітектуру (SOA)** з елементами мікросервісів для найбільш навантажених компонентів. Така гібридна модель дозволяє зберегти цілісність бізнес-логіки (характерну для моноліту, що спрощує розробку на початкових етапах) та забезпечити гнучкість і масштабованість окремих модулів (характерну для мікросервісів).

2.3.1. Технологічний стек

Враховуючи інтеграцію КНУ з продуктами Microsoft, доцільно обрати технологічний стек на базі.NET, що забезпечить нативну сумісність з Azure AD та високу продуктивність.

Таблиця 2.1 – Технологічний стек

Рівень	**Технологія**	**Обґрунтування**
Backend	.NET 7/8 (C#)	Висока продуктивність, типізація, чудова інтеграція з MS SQL та Azure Services.
Frontend	React.js / TypeScript	Забезпечує створення реактивного інтерфейсу (SPA), що важливо для складних форм введення даних та дашбордів.
Database	PostgreSQL	Потужна об'єктно-реляційна СКБД, безкоштовна, відмінна підтримка JSONB для зберігання неструктурованих метаданих публікацій.

Рівень	Технологія	Обґрунтування
Identity	Azure Active Directory	Використання корпоративних акаунтів Office 365 для SSO.[17]
Scraping	Python (Playwright)	Python має найкращі бібліотеки для аналізу даних та скрапінгу (scholarly, beautifulsoup), тому модуль збору даних доцільно винести в окремий сервіс на Python.[10]
API Docs	Swagger / OpenAPI	Стандарт для документування REST API.

2.3.2. Архітектурні шари

Система розбивається на чотири логічні рівні:

1. **Рівень представлення (Presentation Layer):** Веб-портал з адаптивним дизайном. Включає особистий кабінет викладача, кабінет завідувача кафедри (для верифікації), панель адміністратора (для налаштування формул) та публічний розділ (рейтингові списки).
2. **Рівень API (Application Gateway):** Єдина точка входу для фронтенду. Відповідає за маршрутизацію запитів, автентифікацію (валідацію JWT токенів), rate limiting та логування.
3. **Рівень бізнес-логіки (Domain Layer):**
    - *Core Service:* Управління користувачами, структурою університету, довідниками.
    - *Rating Calculation Engine:* Реалізація алгоритмів розрахунку балів.
    - *Validation Service:* Логіка перевірки та затвердження досягнень.
4. **Рівень інфраструктури та інтеграції (Infrastructure Layer):**
    - *Harvester Service:* Асинхронний сервіс для взаємодії зі Scopus, WoS, Scholar.
    - *Dekanat Sync Adapter:* Модуль для імпорту даних з АСУ «Деканат».
    - *Notification Service:* Відправка сповіщень через SMTP або Graph API.

2.4 Детальне проектування компонентів та інтеграцій

Найбільш складною частиною системи є її інтеграційні шлюзи. Розглянемо їх детальніше.

2.4.1 Підсистема збору наукометричних даних

Цей модуль функціонує автономно від основного веб-додатку. Він запускається за розкладом (наприклад, щотижня) або за запитом користувача ("Оновити дані зараз").

Інтеграція зі Scopus: Scopus надає API, який дозволяє отримувати метадані статей, кількість цитувань та h-індекс. Для роботи необхідний API Key, який генерується на порталі розробників Elsevier. Оскільки КНУ має підписку, слід використовувати інституційний токен (Institutional Token), щоб отримати повний доступ до даних, а не обмежену версію для публічного використання. Бібліотека pybliometrics (Python) є рекомендованим інструментом, оскільки вона автоматично кешує результати запитів, зменшуючи навантаження на API та прискорюючи повторні звернення. Важливо враховувати ліміти запитів (Quotas) і реалізувати стратегію "Exponential Backoff" у випадку отримання помилки HTTP 429 (Too Many Requests).

Інтеграція з Google Scholar: На відміну від Scopus, Google Scholar не має офіційного API і активно протидіє автоматизованому збору даних. Використання простих HTTP-запитів призведе до швидкого блокуванні IP-адреси університету. Архітектура повинна передбачати використання headless-браузерів (наприклад, Playwright або Selenium), які імітують поведінку реального користувача. Для забезпечення стабільності роботи критично важливо використовувати проксі-сервери з ротацією IP-адрес. Альтернативним, більш надійним (але платним) варіантом є використання спеціалізованих API-сервісів, таких як SerpApi або ScrapingBee, які беруть на себе складність обходу CAPTCHA та підтримки актуальності парсерів при зміні верстки Google Scholar. В архітектурі слід закласти інтерфейс

IScholarProvider, що дозволить легко перемикатися між власним парсером та платним сервісом залежно від бюджету та технічних можливостей.

Інтеграція з ORCID: ORCID надає Public API для читання даних профілю. Це дозволяє отримати підтверджений список публікацій, який сам дослідник додав до свого профілю. Це слугує додатковим джерелом верифікації та дозволяє об'єднати публікації з різних баз під одним ідентифікатором. Використання OAuth аутентифікації дозволяє системі отримати права на читання навіть обмежених даних (Trusted Party).

2.4.2 Інтеграція з АСУ «Деканат»

Система «Деканат» є джерелом правди про штатний розклад. Оскільки пряма інтеграція з базою даних стороннього розробника може порушувати ліцензійні умови або створювати ризики цілісності даних, рекомендується використовувати шаблон Integration Database або File Transfer. Адміністратори «Деканату» налаштовують регулярний експорт даних (про викладачів, кафедри, дисципліни, навантаження) у формат XML або CSV у захищену мережеву папку або FTP-сервер. АСРО має модуль синхронізації, який парсить ці файли, порівнює з поточною базою даних і вносить зміни (створює нових викладачів, деактивує звільнених, оновлює назви кафедр).

2.4.3 Інтеграція з Moodle

Для отримання даних про методичну роботу (наявність курсу, його наповнення, активність викладача) використовується Moodle Web Services API. Необхідно створити в Moodle спеціального користувача-бота з роллю, що має права moodle/course:view, moodle/user:viewdetails. АСРО використовує згенерований токен для виконання REST-запитів до Moodle, зіставляючи викладачів за адресою корпоративної пошти (яка є ідентичною в Office 365, Moodle та АСРО).

2.5 Проектування бази даних

База даних повинна підтримувати як сувору реляційну структуру для кадрових даних, так і гнучку структуру для наукових метаданих.

Основні сутності:

Users (Користувачі): Зберігає AzureOid (незмінний ID з Azure AD), ПІБ, email, ролі.

Profiles (Профілі): Розширені дані про науковця – науковий ступінь, вчене звання, ідентифікатори (ScopusID, ORCID), посилання на профілі.

Departments (Підрозділи): Ієрархічна структура університету (Університет -> Факультет -> Кафедра).

RatingCriteria (Довідник критеріїв): Містить опис показника, кількість балів за одиницю, категорію (наукова, навчальна, організаційна), формулу розрахунку (наприклад, чи діляться бали на кількість авторів).

Achievements (Досягнення): Таблиця фактів. Зберігає зв'язок UserID, CriteriaID, значення (кількість), дату, статус верифікації, посилання на докази.

Publications (Публікації): Окрема таблиця для детальної інформації про статті. Містить DOI (як унікальний ключ), назву, рік, журнал, квартиль, кількість цитувань.

PublicationAuthors (Автори публікацій): Таблиця зв'язку багато-до-багатьох, що визначає, які користувачі системи є авторами конкретної публікації. Це критично для уникнення дублювання нарахування балів за одну й ту ж статтю декільком співробітникам КНУ (бали мають ділитися або нараховуватися кожному згідно з правилами).

Для зберігання історії рейтингів доцільно використовувати підхід Snapshotting: після затвердження рейтингу за рік, розраховані бали зберігаються в окрему таблицю RatingHistory, яка є незмінною (immutable). Це захищає від ситуації, коли зміна формули в новому році заднім числом змінює результати минулих років.

2.6. UML-діаграми

Для візуалізації архітектурних рішень розроблено комплект UML-діаграм, що описують різні аспекти системи.

2.6.1. Діаграма прецедентів (Use Case Diagram)

Ця діаграма (рис. 2.1) окреслює межі системи та взаємодію акторів з функціональними можливостями.

Актори:

- НПП (Викладач): Основний користувач, зацікавлений у підвищенні свого рейтингу.
- Завідувач кафедри: Відповідальний менеджер, який контролює достовірність даних.
- Адміністратор системи (Секретар Вченої ради): Керує правилами гри (критеріями).
- Студент: Зовнішній актор, що надає зворотний зв'язок.
- Таймер (System Scheduler): Ініціатор фонових процесів.

Основні прецеденти:

- Вхід через Office 365: Усі користувачі автентифікуються через єдиний корпоративний акаунт.
- Синхронізація профілю: Користувач прив'язує свої зовнішні ID (Scopus, ORCID).
- Автоматичний імпорт публікацій: Система періодично завантажує нові статті.
- Ручне додавання досягнень: Викладач вносить дані про видані методички або участь у профорієнтації.
- Верифікація досягнень: Зав. кафедри перевіряє внесені вручну дані та підтверджує або відхиляє їх.
- Розрахунок рейтингу: Система перераховує бали на основі активних формул.

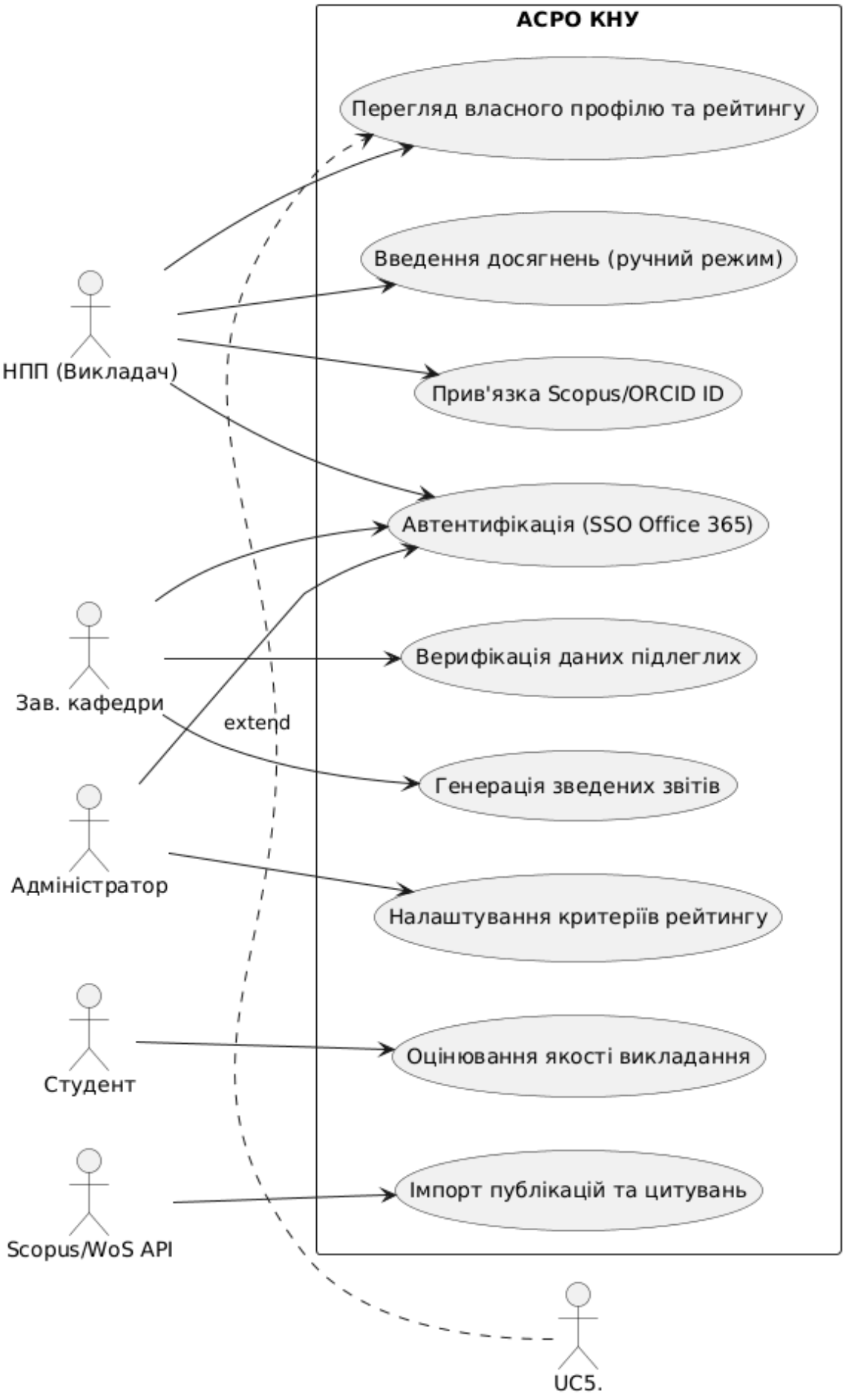


Рисунок 2.1 – Діаграма прецедентів

- Формування звіту: Генерація PDF/Excel файлу з результатами по кафедрі/факультету.
- Оцінювання викладача: Студент заповнює анкету по дисципліні.

2.6.2. Діаграма класів (Class Diagram)

Рисунок 2.2 Відображає статичну структуру доменних об'єктів.

Ключові класи:

- User: Базовий клас з даними авторизації.
- ResearcherProfile: Містить наукові метрики (h-index, загальна кількість цитувань).
- ActivityRecord: Узагальнений клас для будь-якої активності, що оцінюється. Має поліморфні зв'язки з Publication, TeachingLoad, MethodicalWork.
- ScoreConfiguration: Клас, що містить логіку нарахування балів (вага, ліміти).

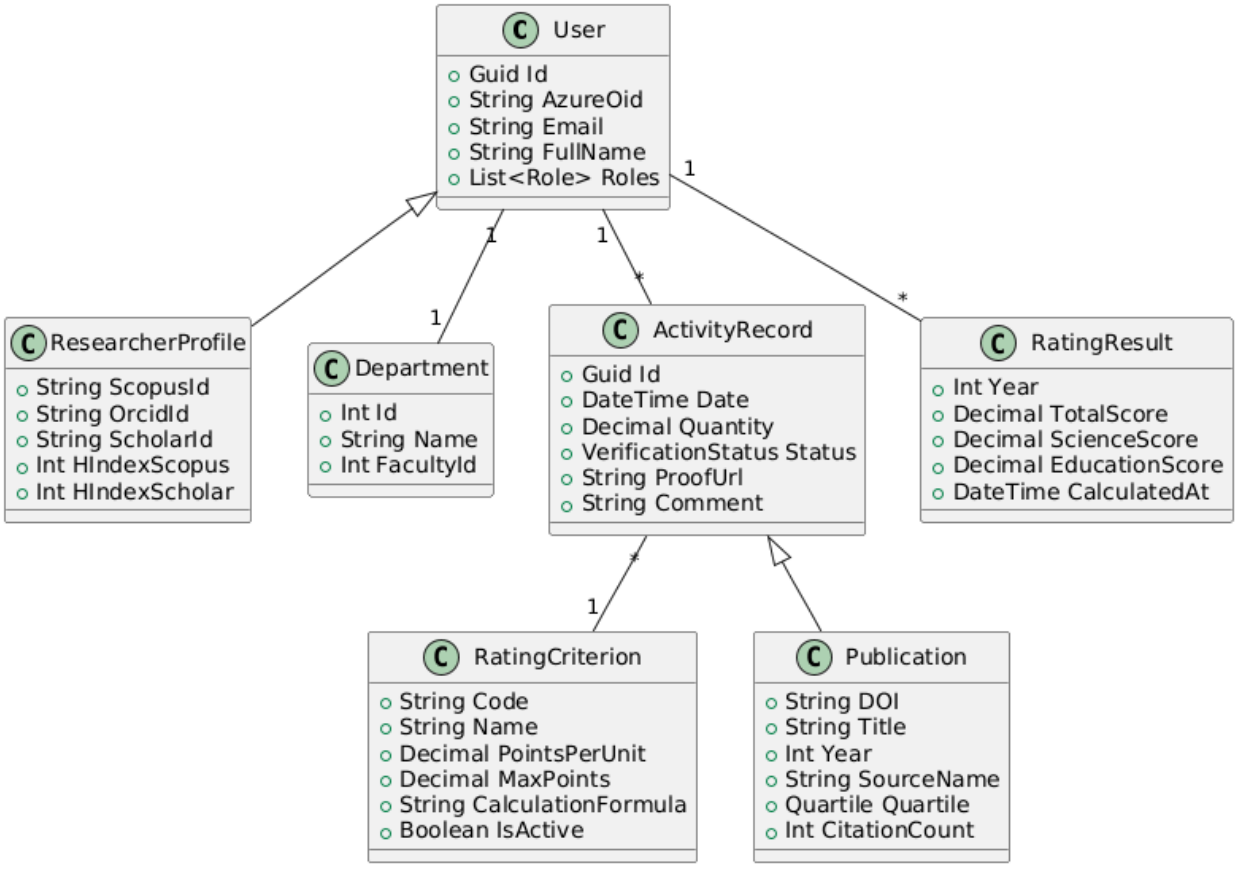


Рисунок 2.2 – Діаграма класів

2.6.3. Діаграма компонентів (Component Diagram)

Рисунок 2.3 ілюструє модульну структуру програмного забезпечення.

Компоненти:

- Web Portal (SPA): Фронтенд додаток.
- Identity Service: Обгортка над Azure AD.
- Rating Core API: Основний бекенд сервіс.
- Data Harvester: Сервіс-воркер для збору зовнішніх даних.
- Dekanat Adapter: Компонент інтеграції з SIS.
- Report Generator: Сервіс для створення документів.

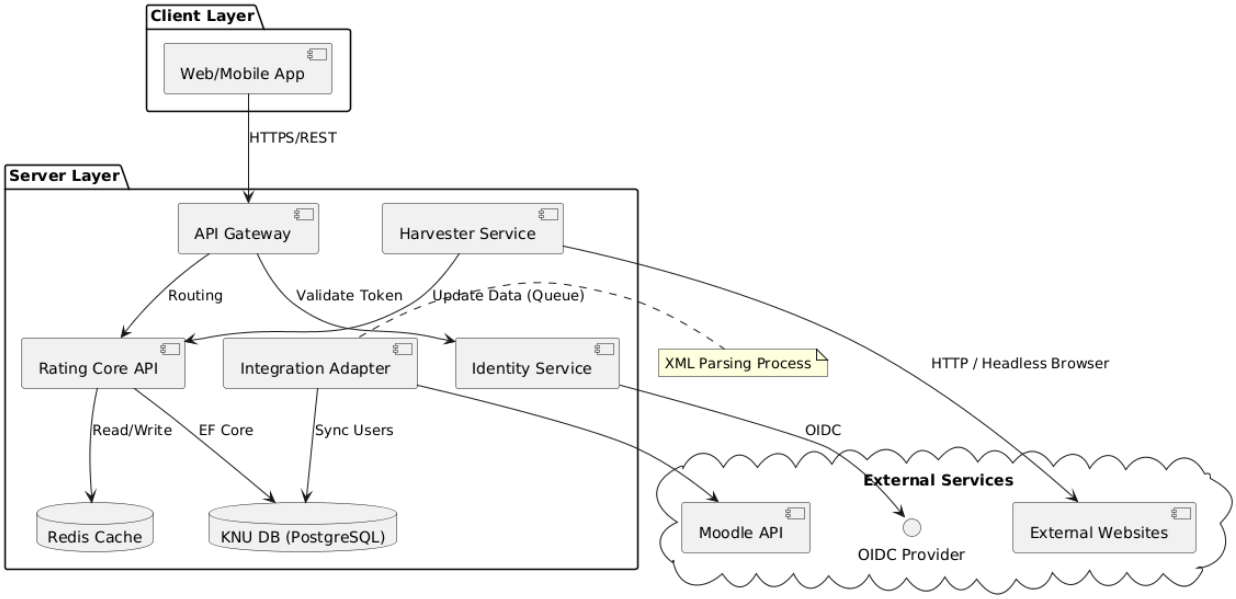


Рисунок 2.3 Діаграма компонентів

2.6.4. Діаграма послідовності (Sequence Diagram)

Діаграма послідовності (рис. 2.4) деталізує процес автоматичного оновлення рейтингу викладача при появі нової публікації в Scopus.

Сценарій:

- Планувальник запускає задачу оновлення.
- Harvester отримує список активних Scopus ID.
- Для кожного ID виконується запит до API Elsevier.
- Отримані дані порівнюються з БД.
- Якщо знайдено нову статтю, вона додається в БД.

- Запускається перерахунок рейтингу.
- Викладач отримує сповіщення.

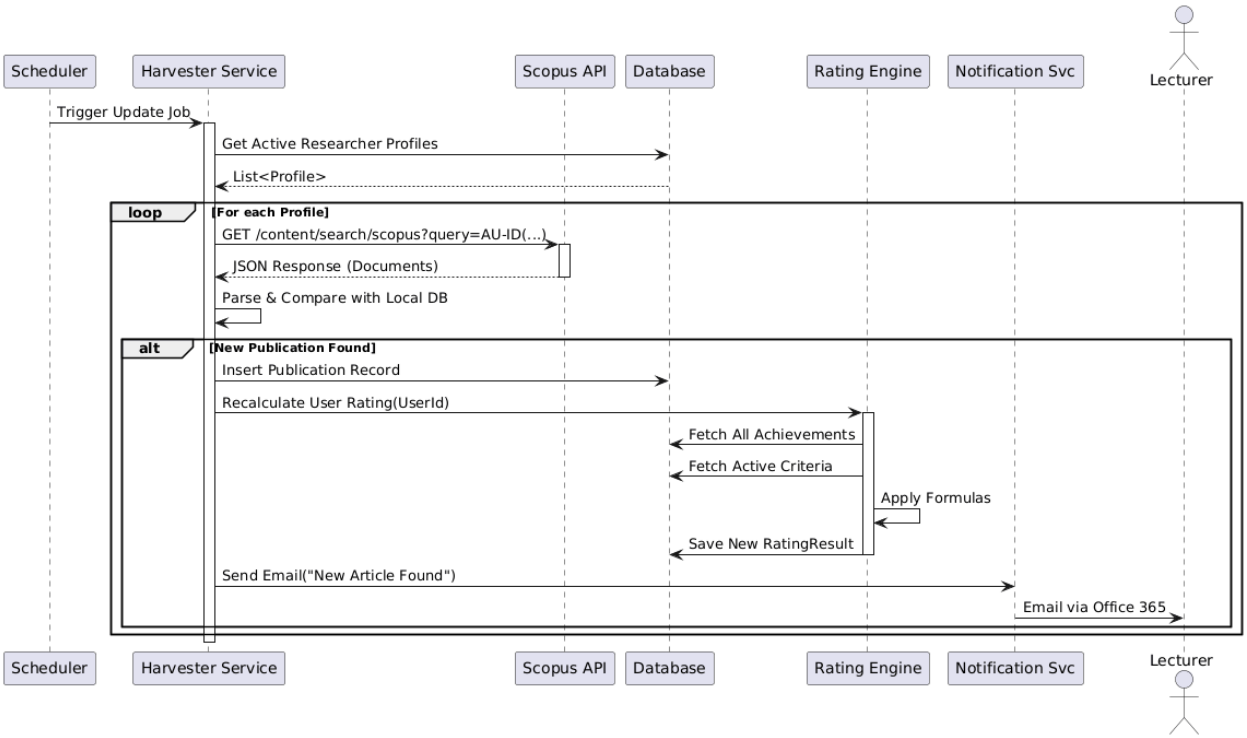


Рисунок 2.4 – Діаграма послідовності

2.6.5. Діаграма розгортання (Deployment Diagram)

Діаграма розгортання (рис. 2.5) описує фізичне розміщення компонентів на серверах.

Вузли:

- Web Server: Хостить фронтенд (static files) та API (Docker containers).
- Worker Server: Окремий сервер для Harvester, щоб ізолювати навантаження від парсингу.
- Database Server: Кластер PostgreSQL.
- CI/CD Runner: Сервер для автоматичного розгортання оновлень.

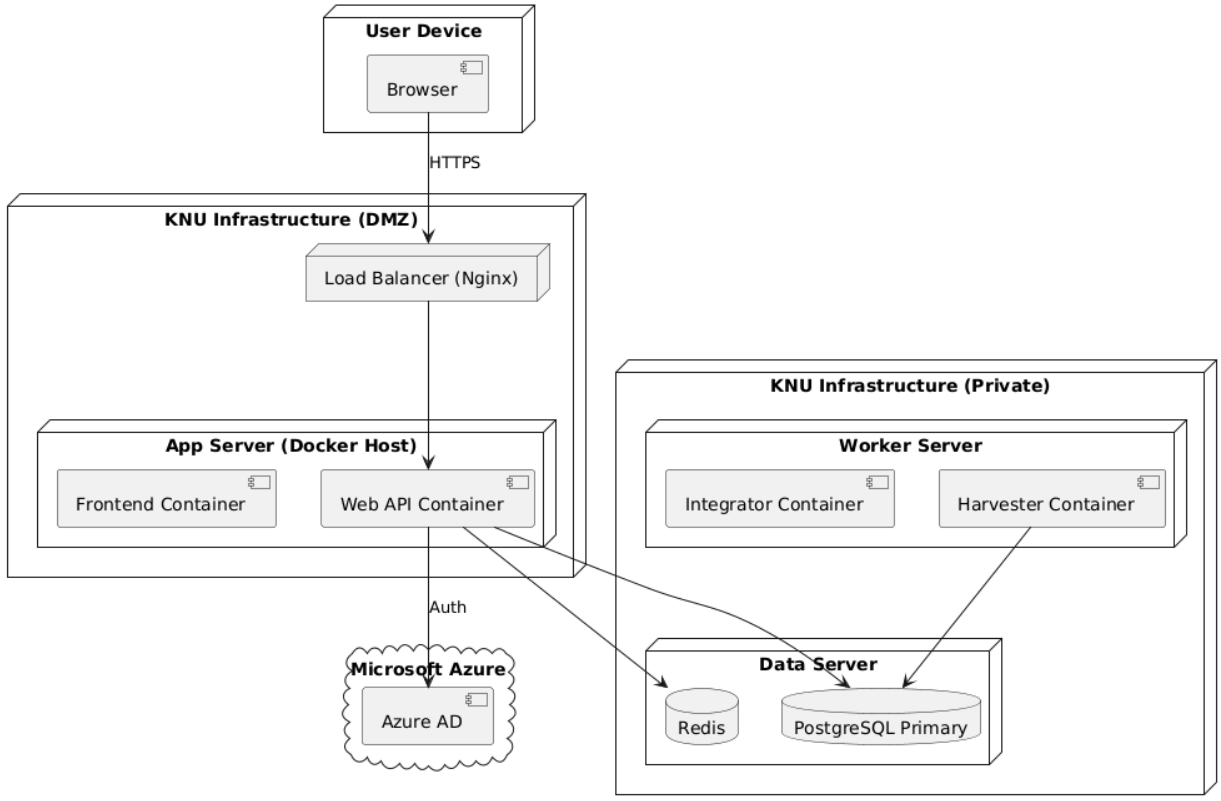


Рисунок 2.5 – Діаграма розгортання

Розроблена архітектура Автоматизованої системи рейтингового оцінювання для Криворізького національного університету є надійним, масштабованим та безпечним рішенням, яке враховує специфіку існуючої інфраструктури закладу. Використання сучасних підходів (SOA, інтеграція з Azure AD, автоматичний харвестинг наукометричних даних) дозволить перетворити рейтингове оцінювання з бюрократичної процедури на ефективний інструмент управління якістю освіти.

Ключові переваги рішення:

- Єдиний вхід: Використання Office 365 акаунтів спрощує доступ та підвищує безпеку.
- Об'єктивність: Мінімізація людського фактору при підрахунку балів за публікації.
- Гнучкість: Можливість налаштування формул дозволяє адаптувати систему до змін у стратегії університету.

Економія часу: Автоматизація збору даних вивільняє час викладачів для наукової та навчальної роботи.

Рекомендований план впровадження:

Етап 1: Розгортання БД та Core API, інтеграція з Azure AD, імпорт користувачів з «Деканату», реалізація ручного введення та верифікації досягнень.

Етап 2: Реалізація модуля Harvester (Scopus, ORCID), інтеграція з Moodle.

Етап 3: Впровадження скрапінгу Google Scholar, розробка аналітичних дашбордів, модуль студентського оцінювання.

Реалізація цього проекту стане вагомим внеском у цифрову трансформацію КНУ, забезпечуючи прозорість, конкурентоспроможність та інноваційний розвиток університету.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РЕЙТИНГОВОГО ОЦІНЮВАННЯ НАУКОВО-ПЕДАГОГІЧНИХ ПРАЦІВНИКІВ

3.1 Загальна реалізація архітектури системи

Для забезпечення масштабованості, надійності та можливості подальшого розширення, пропонується використання багаторівневої архітектури (Clean Architecture) з чітким розділенням відповідальності.

3.1.1 Загальна топологія компонентів

Система складатиметься з трьох основних логічних блоків:

1. **Backend API (Core System):** Центральний вузол, реалізований на ASP.NET Core, що відповідає за бізнес-логіку, доступ до даних, розрахунки рейтингів та надання API для клієнтів.
2. **Web Dashboard (Admin/User Portal):** Повнофункціональний веб-інтерфейс для викладачів та адміністраторів, реалізований як React SPA.
3. **Embeddable Widgets (Micro-frontends):** Набір легковагових компонентів (Web Components або React-to-WebComponent обгорток), призначених для вбудовування на сторонні ресурси.

Таблиця 3.2 – Технологічний стек

Компонент	**Технологія**	**Обґрунтування вибору**
Мова програмування (Backend)	C# /.NET 8	Висока продуктивність, строга типізація, розвинена екосистема корпоративних бібліотек, нативна підтримка асинхронності.[8]
Фреймворк (API)	ASP.NET Core Web API	Вбудована підтримка Dependency Injection, потужний конвеєр Middleware, ефективна робота з CORS.[9]
База даних	PostgreSQL 15+	Підтримка складних реляційних запитів та JSONB для зберігання динамічних атрибутів рейтингових критеріїв.[10]
ORM	Entity	Прискорення розробки, зручність

Компонент	Технологія	Обґрунтування вибору
	Framework Core	міграцій схеми БД.
Frontend / Widgets	React 18 + TypeScript	Компонентний підхід ідеальний для створення віджетів. TypeScript забезпечує надійність коду.[2]
Сбірка (Build Tool)	Vite	Забезпечує швидку збірку та оптимізацію бандлів для віджетів.[2]
Стилізація віджетів	Tailwind CSS (scoped) / CSS Modules	Запобігання конфліктам стилів з хост-сайтом.

3.2 Проектування Бази Даних: Гнучкість та Нормалізація

Структура даних повинна вирішувати головну проблему систем рейтингування — часту зміну критеріїв. Жорстко закодовані колонки (наприклад, points_for_article) призведуть до необхідності змінювати схему БД при кожному новому наказі ректора.

3.2.1 Схема даних (ERD) та використання JSONB

Використання PostgreSQL дозволяє застосувати гібридний підхід: реляційна цілісність для основних сутностей та NoSQL-гнучкість (JSONB) для атрибутів досягнень.[10]

Таблиця Criteria (Метадані системи)

Замість того, щоб програмувати логіку в коді, ми описуємо її в базі даних.

```
CREATE TABLE criteria (
    id SERIAL PRIMARY KEY,
    code VARCHAR(50) UNIQUE NOT NULL, -- Наприклад,
'SC_SCOPUS_ARTICLE'
    category VARCHAR(50) NOT NULL, -- 'Scientific',
'Methodological'
    name_uk TEXT NOT NULL,
    base_points DECIMAL(10, 2) NOT NULL,
    calculation_strategy VARCHAR(100), -- Посилання
```

```
на клас стратегії в коді, напр. 'ScopusCalculationStrategy'
        validation_schema JSONB, -- JSON Schema для валідації полів форми на фронтенді
        is_active BOOLEAN DEFAULT TRUE,
        effective_date DATE NOT NULL
    );
```

Поле validation_schema дозволяє динамічно генерувати форми на клієнті. Якщо завтра додасться вимога вказувати DOI, адміністратор оновить JSON-схему, і поле з'явиться автоматично без перезбірки фронтенду.

Таблиця Submissions (Фактичні дані)

Тут зберігаються записи про активність викладачів.

```
    CREATE TABLE submissions (
        id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
        user_id UUID NOT NULL REFERENCES users(id),
        criterion_id INT NOT NULL REFERENCES criteria(id),
        status VARCHAR(20) DEFAULT 'Pending', -- Pending, Approved, Rejected
        data JSONB NOT NULL, -- Всі специфічні поля: { "journal": "Nature", "quartile": "Q1", "authors": 3 }
        evidence_url TEXT,
        calculated_points DECIMAL(10, 2),
        verifier_id UUID REFERENCES users(id),
        created_at TIMESTAMP DEFAULT NOW()
    );
```

Індекс для прискорення пошуку по JSON атрибутам (наприклад, знайти всі статті Q1)

```
    CREATE INDEX idx_submissions_data ON submissions
```

```
USING GIN (data);
```

Використання GIN-індексу є критичним для продуктивності аналітичних звітів, дозволяючи фільтрувати записи за внутрішніми полями JSON з швидкістю, порівнянною зі звичайними колонками.[11]

3.2.2 Аудит та Історичність

Рейтинги можуть впливати на матеріальне заохочення, тому будь-яка зміна балів повинна бути залогована. Необхідно впровадити таблицю audit_logs, яка фіксуватиме «знімок» (snapshot) запису до і після зміни, а також ID користувача, що здійснив зміну. Це вимога безпеки та прозорості академічного процесу.

3.3 Серверна реалізація (backend): asp.net core architecture

3.3.1. Організація Проекту та Патерни

Застосування Strategy Pattern є ключовим для реалізації варіативної логіки нарахування балів.

Інтерфейс Стратегії:

```
public interface IScoringStrategy
{
    string StrategyName { get; }
    decimal Calculate(JsonElement data, decimal basePoints);
    IEnumerable<string> Validate(JsonElement data);
}
```

Реалізація для Публікацій (Приклад):

```
public class ScopusPublicationStrategy : IScoringStrategy
{
    public string StrategyName => "ScopusCalculationStrategy";
```

```
        public decimal Calculate(JsonElement data,
decimal basePoints)
        {
            var quartile =
data.GetProperty("quartile").GetString();
            var authorsCount =
data.GetProperty("authorsCount").GetInt32();
            var isStudentCoAuthor =
data.GetProperty("hasStudentCoAuthor").GetBoolean();

            decimal multiplier = quartile switch
            {
                "Q1" => 2.0m,
                "Q2" => 1.5m,
                "Q3" => 1.0m,
                _ => 0.5m
            };

            if (isStudentCoAuthor) multiplier += 0.2m;
// Бонус за роботу зі студентами

            return (basePoints * multiplier) /
authorsCount; // Розподіл балів між авторами
        }
        //... validation logic
    }
```

Такий підхід дозволяє додавати нові формули розрахунку (наприклад, для патентів або художніх виставок), створюючи нові класи стратегій, не змінюючи основний код сервісу.[8]

3.3.2 API Design та оптимізація для віджетів

API повинно надавати спеціалізовані ендпоінти для віджетів, оптимізовані за розміром відповіді та кешуванням.

```
GET /api/public/widgets/profile/{userId}
```

Цей ендпоінт повинен повертати DTO (Data Transfer Object), що містить тільки публічні дані, дозволені для показу. Використання Response Caching Middleware є обов'язковим, оскільки віджет може завантажуватися тисячі разів на день при відвідуванні сайту університету.

```
 [HttpGet("widgets/profile/{userId}")]
// Кеш на 1 годину
public async Task<IActionResult> GetWidgetProfile(Guid userId)
{
    var profile = await _ratingService.GetPublicProfileAsync(userId);
    return Ok(profile);
}
```

3.3.3 Налаштування CORS (Cross-Origin Resource Sharing)

Це критичний аспект безпеки для віджета. Браузер заблокує запит віджета з домену knu.edu.ua до API api.rating-system.com, якщо сервер не надішле правильні заголовки.[9]

Конфігурація в Program.cs:

```
var builder = WebApplication.CreateBuilder(args);

// Визначення політики CORS для довірених хостів
builder.Services.AddCors(options =>
{
    options.AddPolicy("WidgetHostPolicy", policy =>
    {
        policy.WithOrigins(
```

```
                    "https://knu.edu.ua",
                    "https://www.knu.edu.ua",
                    "https://moodle.knu.edu.ua"
                )
                .AllowAnyMethod()
                .AllowAnyHeader()
                .AllowCredentials(); // Необхідно, якщо віджет передає авторизаційні куки
        });
    });

    var app = builder.Build();
    app.UseCors("WidgetHostPolicy"); // Middleware має бути перед UseAuthorization
```

3.4. Клієнтська частина (frontend): технології вбудовування

Розробка віджета, який можна інтегрувати в сторонній сайт, вимагає вирішення проблеми ізоляції. Стилі сайту-хоста (наприклад, глобальний CSS для h1 або button) можуть "поламати" верстку віджета, і навпаки.

Таблиця 3.2 – Порівняння методів інтеграції

Характеристика	**Iframe**	**Web Components (Custom Elements)**	**JavaScript Widget (Direct DOM)**
Ізоляція стилів	Повна (Браузерна ізоляція)	Висока (Shadow DOM)	Низька (Потребує унікальних класів)
Продуктивність	Низька (Завантаження повного документа)	Висока (Легковаговий)	Висока
SEO	Контент не видно пошуковикам	Видно (якщо рендеринг на клієнті налаштовано)	Видно

Характеристика	Iframe	Web Components (Custom Elements)	JavaScript Widget (Direct DOM)
Складність взаємодії	Тільки через postMessage	Нативна JS взаємодія	Нативна JS взаємодія
Адаптивність висоти	Складно (потрібні скрипти resize)	Автоматично (частина DOM)	Автоматично

Для даного проекту оптимальним є використання React загорнутого в Web Component (Custom Element) з використанням Shadow DOM для ізоляції стилів. Це поєднує зручність розробки на React з нативною сумісністю стандарту HTML.

Для перетворення React-компонента на Custom Element використовуємо бібліотеку react-to-webcomponent або власну обгортку. Важливо правильно ін'єктувати стилі всередину Shadow Root.

```
// WidgetEntry.tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import WidgetApp from './WidgetApp';
import styles from './index.css?inline'; // Vite: імпорт CSS як стрічки

class KnuRatingWidget extends HTMLElement {
  connectedCallback() {
    const mountPoint = document.createElement('div');
    const shadow = this.attachShadow({ mode: 'open' }); // Відкритий Shadow DOM

    // Ін'єкція стилів (Tailwind або власні) всередину ізольованого контейнера
    const styleTag = document.createElement('style');
```

```
        styleTag.textContent = styles;
        shadow.appendChild(styleTag);
        shadow.appendChild(mountPoint);

        // Отримання параметрів з атрибутів HTML тегу
        const userId = this.getAttribute('user-id');
        const theme = this.getAttribute('theme') |

    | 'light';

        const root = ReactDOM.createRoot(mountPoint);
        root.render(<WidgetApp          userId={userId}
theme={theme} />);
      }
    }

    // Реєстрація кастомного елемента
    customElements.define('knu-academic-rating',
KnuRatingWidget);
```

- Тепер інтеграція на сайт університету виглядає максимально просто:

```
    HTML
    <script      src="https://rating-system.com/widget-
bundle.js"></script>
    <knu-academic-rating   user-id="550e8400-e29b-41d4-
a716-446655440000"></knu-academic-rating>
```

Віджет повинен вміти приймати конфігурацію через атрибути (props).

- data-theme: "light", "dark" або "university-brand".
- data-lang: "uk", "en".

- `data-view-mode: "full", "compact", "badge-only".`

3.5 Детальна специфікація бізнес-логіки оцінювання

На основі аналізу та загальних практик, система повинна реалізувати таку логіку:

3.5.1 Модуль "Наукові публікації"

- **Введення:** DOI (Digital Object Identifier).
- **Автоматизація:** Система робить запит до API CrossRef/Scopus за DOI.
- **Дані:** Отримує назву, журнал, рік, список авторів.
- **Розрахунок:**
    1. Перевірка журналу в базі SJR (Scimago Journal Rank) для визначення квартиля на рік публікації.
    2. Базовий бал (наприклад, Q1 = 100).
    3. Перевірка співавторів. Якщо серед співавторів є користувачі системи зі статусом "Студент", застосувати коефіцієнт $K_{stud} = 1.2$.
    4. Ділення на кількість авторів, якщо це передбачено налаштуваннями кафедри.

3.5.2 Модуль "Академічна мобільність"

- **Типи:** Віртуальна, фізична, змішана.[5]
- **Валідація:** Завантаження скан-копії угоди Learning Agreement або сертифікату.
- **Бали:** Залежать від тривалості (кредитів ECTS). Наприклад, 1 кредит ECTS = 5 балів рейтингу.

3.6 Стратегія розгортання та експлуатації (devops)

3.6.1 Контейнеризація

Використання Docker забезпечує ідентичність середовищ розробки та продакшену.

- **Backend:** Docker image на базі mcr.microsoft.com/dotnet/aspnet:8.0.

- **Frontend (Admin):** Nginx container для статики.
- **Widget:** Розміщення JS-бандла та CSS на CDN (або через Nginx) з налаштованим кешуванням та Gzip/Brotli компресією.

3.6.2 CI/CD Pipeline

Автоматизація процесу збірки та деплою (GitHub Actions або GitLab CI):

1. **Build:** Компіляція.NET рішення, збірка React додатків.
2. **Test:** Запуск Unit-тестів (xUnit) та Integration тестів.
3. **Publish:** Створення Docker образів і пуш в Registry.
4. **Deploy:** Оновлення сервісів на сервері (Docker Swarm або Kubernetes).

3.6.3 Моніторинг

Впровадження Health Checks для API. Віджет повинен вміти "тихо" відключатися (graceful degradation), якщо API недоступне, не ламаючи при цьому верстку хост-сайту (наприклад, просто не відображаючись або показуючи кешовану заглушку).

Запропоноване архітектурне рішення на базі ASP.NET Core 8, PostgreSQL та React Web Components повністю задовольняє вимоги щодо створення гнучкого, інтегрованого веб-застосунку для рейтингування НПП.

Використання підходу Widget-First дозволяє університету трансформувати внутрішню систему звітності у потужний інструмент публічної комунікації, демонструючи науковий потенціал на офіційних ресурсах.

Гнучка схема бази даних на основі JSONB та патерн Strategy на бекенді гарантують довговічність системи та її здатність адаптуватися до змінних вимог Міністерства освіти і науки України без необхідності дороговартісної переробки коду.

Реалізація цього проекту стане каталізатором цифровізації управлінських процесів КНУ, забезпечуючи прозорість, об'єктивність та доступність даних у реальному часі.

4 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ПРОЕКТУ

4.1 Аналіз поточного стану та недоліки ручної моделі

На даний момент процедура рейтингового оцінювання в багатьох українських ЗВО, включаючи КНУ, часто характеризується фрагментарністю та високою трудомісткістю. Процес зазвичай виглядає наступним чином:

Збір первинних даних: НПП самостійно відстежує свої досягнення (статті, тези, участь у конференціях, методична робота) та вносить їх у локальні файли або паперові анкети.

Верифікація: Завідувач кафедри або відповідальна особа перевіряє достовірність даних, часто вимагаючи фізичні копії підтверджуючих документів (сертифікати, титульні сторінки публікацій).

Агрегація: Дані з кафедр передаються до деканатів, де зводяться у факультетські звіти, а згодом — до ректорату або навчально-методичного відділу для формування загальноуніверситетського рейтингу.

Така модель має низку критичних недоліків:

- Висока ймовірність помилок: Ручне введення та перенесення даних неминуче призводить до механічних помилок, дублювання записів або втрати інформації.
- Часова затримка: Результуючий рейтинг стає доступним через значний проміжок часу після завершення звітного періоду, що унеможливлює оперативне реагування на низькі показники.
- Відсутність прозорості: Викладачі часто не мають доступу до динаміки свого рейтингу протягом року і бачать лише кінцевий результат, що знижує мотиваційний ефект.
- Неможливість аналітики: Дані, що зберігаються в паперовому вигляді або в статичних файлах Excel, важко використовувати для глибокого аналізу (наприклад, кореляції між науковим ступенем та публікаційною активністю, або між віком та використанням новітніх методик).

2025 рік вносить суттєві корективи у функціонування закладів вищої освіти України. По-перше, зміни в податковому законодавстві, зокрема підвищення військового збору до 5% , вимагають від керівництва університетів більш ретельного планування бюджетів та пошуку шляхів оптимізації витрат. По-друге, нові вимоги МОН щодо зменшення навчального навантаження до 400-500 годин на рік при одночасному підвищенні вимог до наукової роботи створюють потребу в точному обліку саме наукової складової роботи викладача.

Автоматизована система повинна не лише фіксувати досягнення, але й слугувати інструментом доказової бази виконання індивідуального плану в частині наукової роботи, що стає підставою для збереження ставок та нарахування стимулюючих виплат. Впровадження такої системи є прямим виконанням стратегії цифрової трансформації освіти і науки.

4.2 Розрахунок витрат на інфраструктуру

Враховуючи обмежені ресурси (один розробник), архітектура системи повинна бути монолітною (Modular Monolith), але з чітким розділенням шарів. Мікросервісна архітектура в даному випадку буде надлишковою і призведе до ускладнення підтримки ("DevOps overhead").

Пропонований стек технологій:

- Backend: ASP.NET Core 8/9. Забезпечує високу продуктивність та вбудовані механізми безпеки.
- Frontend: React.js або Angular. React є більш популярним і дозволяє швидше знаходити готові компоненти інтерфейсу.
- Database: PostgreSQL. Це потужна об'єктно-реляційна СУБД з відкритим кодом. Вибір PostgreSQL замість MS SQL Server дозволяє університету заощадити значні кошти на ліцензуванні, не втрачаючи при цьому в продуктивності та надійності.

- Containerization: Docker. Дозволяє легко розгортати систему на будь-якому сервері (Linux VPS), ізолювати середовища розробки та продуктиву.

Для розміщення веб-додатку та бази даних необхідний віртуальний приватний сервер (VPS). Розглянемо варіанти, доступні на ринку у 2025 році, порівнюючи ціну та технічні характеристики.

Таблиця 4.1 – Порівняльний аналіз VPS провайдерів (Конфігурація: 2-4 vCPU, 4-8 GB RAM)

Провайдер	**Тарифний план**	**Вартість (міс)**	**Валюта**	**Еквівалент (грн)**	**Переваги/Недоліки**
Hetzner (DE)	CPX21 / CX33	€8.00 - €11.00	EUR	~380 - 520 грн	Найкраще співвідношення ціна/якість. Висока надійність. Дата-центри в Німеччині (GDPR).[24]
DigitalOcean (US)	Basic Droplet (4GB)	$24.00	USD	~1,080 грн	Зручний інтерфейс, але значно дорожче за аналогічні ресурси.[26]
HostPro (UA)	Turbo NVMe	~800 - 1200	UAH	800 - 1200 грн	Локалізація в Україні (швидкість доступу). Підтримка українською. Дорожче за Hetzner.[28]
FreeHost (UA)	Cloud VPS	~600 - 900	UAH	600 - 900 грн	Бюджетний варіант, але технічні показники I/O часто гірші.[29]

Для бюджетної установи оптимальним вибором є Hetzner Cloud або український HostPro, якщо є жорстка вимога щодо розміщення серверів фізично в Україні. Враховуючи євроінтеграційний курс та відповідність GDPR, сервери в Німеччині (Hetzner) є прийнятними. Однак, для розрахунку

бюджету з запасом візьмемо середню вартість якісного українського хостингу або вищого тарифу Hetzner + резервне копіювання.

Закладений бюджет на інфраструктуру: 1,500 грн/міс (включаючи Prod сервер, Test сервер та Backups).

4.3 Розрахунок собівартості розробки

4.1. Розрахунок фонду оплати праці розробника

Цільова зарплата Middle Full Stack розробника становить $2,000.

Курс: 45.00 грн/$.

Цільова сума: 90,000 грн.

Для виплати такої суми працівнику в штаті, необхідно нарахувати "брудну" зарплату (Gross), з якої будуть утримані податки працівника (ПДФО 18% + ВЗ 5%).

$$\text{Gross} = \frac{\text{Net}}{1 - (\text{ПДФО} + \text{ВЗ})} = \frac{90,000}{1 - (0.18 + 0.05)} = \frac{90,000}{0.77} \approx 116,883 \text{ грн.}$$

Витрати роботодавця (Університету):

На суму Gross нараховується ЄСВ (22%). Оскільки 116,883 грн менше за максимальну базу (160,000 грн), ЄСВ нараховується на всю суму.

$$\text{ЄСВ} = 116,883 \times 0.22 = 25,714 \text{ грн.}$$

Таблиця 4.2. – Щомісячні витрати на утримання одного розробника

Стаття	**Сума (грн)**	**Коментар**
Зарплата "на руки" (Net)	90,000	Еквівалент $2,000
ПДФО (18%)	21,039	Утримується з зарплати
Військовий збір (5%)	5,844	Утримується з зарплати (підвищена ставка)
Нарахована зарплата (Gross)	116,883	Фонд зарплати
Нарахування ЄСВ (22%)	25,714	Сплачує університет понад фонд ЗП

Стаття	Сума (грн)	Коментар
ПОВНА ВАРТІСТЬ (Total Cost)	142,597	Реальні витрати університету на місяць

4.4. Кошторис проєкту

Крім зарплати, необхідно врахувати витрати на обладнання робочого місця (ноутбук, монітор), ліцензії на програмне забезпечення (IDE, підписки) та серверну інфраструктуру на період розробки.

Таблиця 4.3. –Зведений кошторис розробки

Категорія витрат	Найменування	Од. виміру	К-сть	Ціна за од. (грн)	Всього (грн)
Персонал	Full Stack Розробник (6 міс.)	міс.	6	142,597	**855,582**
Обладнання	Ноутбук (рівня MacBook Pro / Dell XPS)	шт.	1	80,000	80,000
	Монітор 27" 4K	шт.	1	15,000	15,000
ПЗ та Ліцензії	JetBrains All Products Pack (рік)	шт.	1	12,000	12,000
Інфраструктура	Оренда VPS (Dev/Test)	міс.	6	1,500	9,000
	Доменне ім'я (.edu.ua /.com.ua)	рік	1	1,000	1,000
Непередбачувані	Резервний фонд (5%)	-	-	-	48,629
РАЗОМ					**1,021,211**

Отже, собівартість розробки системи складає 1,021,211 грн (трохи більше 1 мільйона гривень). У доларовому еквіваленті це приблизно $22,700.

Для порівняння: замовлення аналогічної системи в аутсорсинговій компанії за ставками $35-40/год 11 обійшлося б у суму:

$$6 \text{ міс} \times 160 \text{ год} \times \$35 = \$33,600$$

До цієї суми аутсорсери зазвичай додають витрати на PM (20%), QA (15%) та ризики. Реальний кошторис "під ключ" склав би $50,000 - $60,000 (2.2

- 2.7 млн грн). Таким чином, власна розробка економить університету понад 50% бюджету.

4.5 Аналіз економічної ефективності

Економічна вигода від впровадження інформаційної системи в бюджетній установі виражається не в прямому прибутку, а в економії ресурсів (часу персоналу) та підвищенні ефективності їх використання.

4.5.1 Монетизація часу науково-педагогічних працівників

Щоб розрахувати економію, необхідно визначити вартість однієї години роботи НПП. Використаємо дані про посадові оклади за Єдиною тарифною сіткою (ЄТС) на 2025 рік та типові надбавки.

Профіль типового НПП (Доцент, кандидат наук):

- Тарифний розряд: 19.
- Базовий оклад: ~11,000 грн.
- Надбавка за стаж (понад 10 років): 20%.
- Надбавка за науковий ступінь: 15%.
- Надбавка за вчене звання: 25%.
- Сумарна місячна зарплата (Gross): ~$18,000 - $19,000 грн.

З урахуванням професорів (які отримують вищі надбавки) та асистентів, приймемо середню зважену вартість місячної зарплати НПП на рівні 20,000 грн.

Повні витрати університету на 1 НПП (з ЄСВ 22%):

$$20,000 \times 1.22 = 24,400 \text{ грн/міс.}$$

При середній нормі робочого часу 160 годин/місяць, вартість 1 години роботи становить:

$$\frac{24,400}{160} = 152.5 \text{ грн/год.}$$

Для спрощення розрахунків округлимо до 153 грн/год.

4.5.2 Розрахунок витрат часу на процес рейтингування

Процес рейтингування охоплює 900 викладачів КНУ.

Сценарій «Ручний режим»:

НПП: Збір даних, ксерокопіювання, заповнення таблиць, візити на кафедру. Експертна оцінка: 10 годин на рік.

Зав. кафедри: Перевірка анкет кафедри (15-20 осіб), зведення звіту. Витрати: 1 година на кожного викладача.

Всього людино-годин:

$$(900 \text{ НПП} \times 10 \text{ год}) + (900 \text{ анкет} \times 1 \text{ год}) = 9,900 \text{ годин.}$$

Вартість процесу:

$$9,900 \text{ год} \times 153 \text{ грн/год} = 1,514,700 \text{ грн.}$$

Сценарій «Автоматизована система»:

НПП: Перевірка автоматично підтягнутих даних (з Scopus, ЄДЕБО, Деканату), завантаження PDF-файлів для неавтоматизованих пунктів. Оцінка: 2 години на рік.

Зав. кафедри: Верифікація в адмін-панелі (натискання кнопки "Затвердити" або коментар). Оцінка: 0.2 години (12 хвилин) на анкету.

Всього людино-годин:

$$(900 \times 2) + (900 \times 0.2) = 1,800 + 180 = 1,980 \text{ годин.}$$

Вартість процесу:

$$1,980 \text{ год} \times 153 \text{ грн/год} = 302,940 \text{ грн.}$$

4.5.3 Розрахунок річної економії та терміну окупності

Річна економія:

$$1,514,700 - 302,940 = 1,211,760 \text{ грн.}$$

Це сума "віртуальних" коштів, які університет вивільняє. Фактично, це означає, що викладачі витратять на 8,000 годин більше на написання статей,

грантових заявок або методичну роботу, що принесе університету додаткове фінансування в майбутньому.

Операційні витрати системи на рік:

Після розробки систему потрібно підтримувати. Пропонується перевести розробника на 0.25 ставки або укласти договір підтримки.

Підтримка (0.25 ставки): 142,597/4 = 35,650 грн/міс*12 = 427,800 грн/рік.

Хостинг та домен: 1,500 *12 + 1,000 = 19,000 грн/рік.

Всього OPEX: 446,800 грн/рік.

Чистий річний економічний ефект:

$$\text{Економія} - \text{OPEX} = 1,211,760 - 446,800 = 764,960 \text{ грн.}$$

Термін окупності:

$$PP = \frac{\text{CAPEX}}{\text{Net Benefit}} = \frac{1,021,211}{764,960} \approx 1.33 \text{ року.}$$

Інвестиція у власну розробку окупиться через 1 рік і 4 місяці експлуатації. Це надзвичайно високий показник ефективності для IT-проєктів у державному секторі.

Реалізація проєкту створення автоматизованої системи рейтингового оцінювання НПП КНУ є економічно обґрунтованою та стратегічно необхідною. При початкових інвестиціях у розмірі ~1.02 млн грн, система забезпечує річну економію ресурсів університету в еквіваленті ~1.2 млн грн та окупається менш ніж за півтора року.

Вибір моделі власної розробки (in-house) на стеку.NET дозволяє отримати продукт, максимально адаптований до специфіки університету, за ціною, що є на 50-60% нижчою за ринкові пропозиції аутсорсингу. Окрім прямого економічного ефекту, впровадження системи забезпечить прозорість кадрової політики, підвищить мотивацію персоналу та покращить позиції університету в міжнародних рейтингах завдяки якісній аналітиці даних. Проєкт рекомендовано до включення в план стратегічного розвитку КНУ на 2025 рік.

ВИСНОВКИ

Проєкт розробки Автоматизованої системи рейтингового оцінювання для Криворізького національного університету пройшов повний цикл попереднього аналізу: від визначення актуальності ідеї до детального технічного проєктування та економічного розрахунку.

У ході нашої роботи ми сформували повний пакет проєктної документації, необхідної для старту розробки. Визначено, що ручний збір даних є критично неефективним (витрачається понад 9000 годин персоналу на рік). Обґрунтовано необхідність системи в контексті нових вимог законодавства 2025 року (зміни податків, вимоги до наукової звітності).

Досліджено досвід інших ЗВО (Львівська політехніка, КПІ, закордонні CRIS-системи) та виявлено, що готові рішення або занадто дорогі, або не гнучкі.

Розроблено специфікацію вимог, адаптовану під інфраструктуру КНУ (інтеграція з Office 365, Moodle, АСУ «Деканат»). Визначено ролі користувачів (НПП, Зав. кафедри, Адміністратор) та сценарії їх взаємодії.

Спроєктовано модульну архітектуру на базі.NET 8 (Backend) та React (Frontend). Запропоновано інноваційний підхід «Widget-First», що дозволяє вбудовувати рейтинги викладачів на будь-які сайти університету.

Розроблено схему бази даних (PostgreSQL) з використанням JSONB для гнучкого налаштування критеріїв без зміни коду.

Створено комплект UML-діаграм (прецедентів, класів, компонентів, послідовності), що є готовим гайдом для розробника.

Обрано модель розробки In-house (один штатний Full Stack розробник) як найбільш економічно вигідну. Розраховано бюджет розробки (~1.02 млн грн) та термін окупності (1.3 року).

Доведено, що система заощадить університету ресурсів на суму понад 1.2 млн грн щорічно за рахунок вивільнення часу викладачів.

Проєкт є класичним прикладом цифрової трансформації, яка має чіткий економічний та репутаційний ефект. Система перетворить рейтинг з "інструменту покарання" на "інструмент розвитку", показуючи викладачу його прогрес у реальному часі та прозоро пов'язуючи результати з преміюванням.

Обраний стек (.NET + React + PostgreSQL) та архітектурні рішення (Strategy Pattern, JSONB fields) гарантують, що система зможе адаптуватися до змін у законодавстві чи критеріях оцінювання в найближчі 5-7 років без переписування коду.

Власна розробка коштує вдвічі дешевше за аутсорсинг і дозволяє залишити експертизу всередині університету.

ПЕРЕЛІК ПОСИЛАНЬ

1. Про вищу освіту [Електронний ресурс] : Закон України від 01.07.2014 р. № 1556-VII / Верховна Рада України. — Режим доступу: https://zakon.rada.gov.ua/laws/show/2145-19.
2. Деякі питання підвищення кваліфікації педагогічних і науково-педагогічних працівників [Електронний ресурс] : постанова Кабінету Міністрів України від 21.08.2019 р. № 800. — Режим доступу: https://zakon.rada.gov.ua/laws/show/800-2019-%D0%BF.
3. Про затвердження Положення про атестацію педагогічних працівників [Електронний ресурс] : наказ М-ва освіти і науки України від 09.09.2022 р. № 805. — Режим доступу: https://zakon.rada.gov.ua/laws/show/z1649-22.
4. Положення про рейтингування кафедр Національного університету «Львівська політехніка» [Електронний ресурс] : затв. наказом ректора від 23.03.2020 р. № 153-1-10 (зі змінами від 14.10.2024 р.). — Режим доступу: https://lpnu.ua/polozhennia-pro-reitynguvannia-kafedr.
5. Положення про матеріальне заохочення та інші виплати працівникам Національного університету «Львівська політехніка» [Електронний ресурс] : затв. наказом ректора від 15.11.2022 р. № 566-1-10. — Режим доступу: https://lpnu.ua/polozhennia-pro-materialne-zaokhochennia.
6. Положення про рейтингове оцінювання професійної діяльності науково-педагогічних працівників ТНПУ ім. В. Гнатюка [Електронний ресурс]. — Тернопіль : ТНПУ, 2024. — Режим доступу: https://tnpu.edu.ua/about/public_inform/upload/2024/Polozhennia_pro_reitynhove_otsiniuvannia_profesiinoi_diialnosti_naukovo_pedahohichnykh_pratsivnykiv.pdf.
7. Норми бального оцінювання діяльності науково-педагогічних працівників КПІ ім. Ігоря Сікорського на 2023/2024, 2024/2025 навчальні роки [Електронний ресурс] : додаток 2 до наказу. — Київ : КПІ, 2023. — Режим доступу:

https://osvita.kpi.ua/sites/default/files/downloads/dodatok_2_normi_balnogo_ocinyuvannya_2023-1.pdf.

8. Положення про рейтингове оцінювання діяльності науково-педагогічних працівників Інституту Військово-Морських Сил [Електронний ресурс]. — Одеса, 2025. — Режим доступу: https://ivms.mil.gov.ua/wp-content/uploads/2025/06/polozhennya-pro-rejtyngove-oczinyuvannya-npp.pdf.
9. Порядок конкурсного відбору на посади науково-педагогічних працівників у Київському національному університеті імені Тараса Шевченка [Електронний ресурс]. — Київ : КНУ, 2021. — Режим доступу: https://qft.knu.ua/wp-content/uploads/2021/09/15-poryadok-konkursnogo-vidboru-na-posady-naukovo-pedagogichnyh-praczivnykiv-u-kyyivskomu-naczionalnomu-universyteti-imeni-tarasa-shevchenka.pdf.
10. Рекомендації для закладів вищої освіти щодо розбудови внутрішньої системи забезпечення якості освітньої діяльності та якості вищої освіти [Електронний ресурс] / Нац. агентство із забезпечення якості вищої освіти. — Київ, 2024. — Режим доступу: https://naqa.gov.ua/wp-content/uploads/2024/12/Rozjasnennja-shhodo-zastosuvannja-Kriteriiv-ocinjuvannja-jakosti-osvitnoi-programi.pdf.
11. Скиба Ю. Оцінювання діяльності науково-педагогічних працівників: вітчизняний досвід і вектори розвитку в умовах автономії університетів [Електронний ресурс] / Ю. Скиба // Інститут вищої освіти НАПН України. — Режим доступу: https://lib.iitta.gov.ua/id/eprint/744336/1/Оцінювання%20діяльності%20науково-педагогічних%20працівників.pdf.
12. Профспілка освітян виступає проти збільшення навантаження на вчителів [Електронний ресурс] // Укрінформ. — 2025. — 14 листопада. — Режим доступу: https://www.ukrinform.ua/rubric-society/4058868-profspilka-osvitan-vistupae-proti-zbilsenna-navantazenna-na-vciteliv.html.

13. Agreement on Reforming Research Assessment [Electronic resource] / Coalition for Advancing Research Assessment (CoARA). — 2022. — Mode of access: https://coara.eu/agreement/signatories/.
14. Положення про рейтингування науково-педагогічних працівників КПІ ім. Ігоря Сікорського [Електронний ресурс] : затв. наказом ректора від 30.12.2021 р. — Київ : КПІ ім. Ігоря Сікорського, 2021. — Режим доступу: https://osvita.kpi.ua/sites/default/files/downloads/Pol_reityng_NPP_30-12-2021.pdf.
15. Положення про рейтингування кафедр Національного університету «Львівська політехніка» [Електронний ресурс] : затв. наказом ректора від 23.03.2020 р. № 153-1-10 (зі змінами від 14.10.2024 р.). — Львів : НУ «Львівська політехніка», 2024. — Режим доступу: https://lpnu.ua/polozhennia-pro-reitynguvannia-kafedr.
16. Положення про рейтингове оцінювання діяльності науково-педагогічних працівників Інституту Військово-Морських Сил [Електронний ресурс]. — Одеса, 2025. — Режим доступу: https://ivms.mil.gov.ua/wp-content/uploads/2025/06/polozhennya-pro-rejtyngove-oczinyuvannya-npp.pdf.
17. Положення про матеріальне заохочення та інші виплати працівникам Національного університету «Львівська політехніка» [Електронний ресурс]. — Львів, 2022. — Режим доступу: https://lpnu.ua/sites/default/files/2020/pages/2041/polozhennya-pro-materialne-zaokhochennya-ta-inshi-viplati-pracivnikam.pdf.
18. Pure: The world’s leading Research Information Management System [Electronic resource] / Elsevier. — Mode of access: https://www.elsevier.com/products/pure.
19. Symplectic Elements: Powering the Research Management Ecosystem [Electronic resource] / Digital Science. — Mode of access: https://www.symplectic.co.uk/theelementsplatform/.

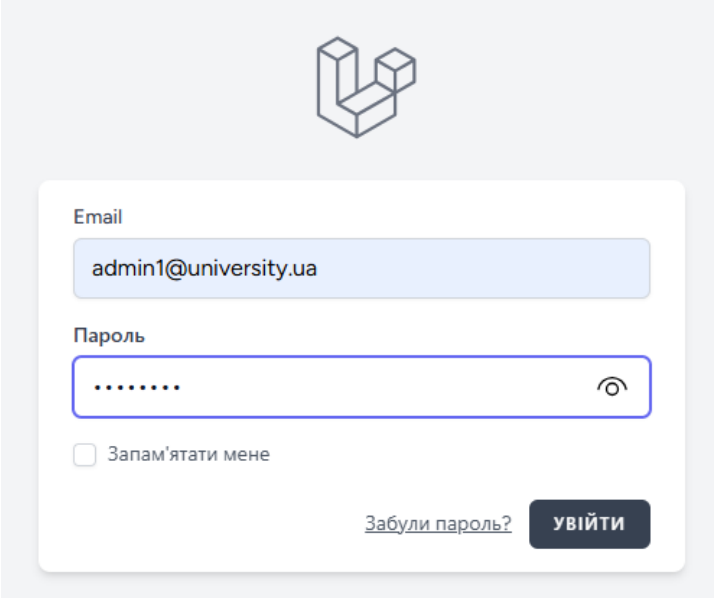
20. DSpace-CRIS: Introduction and Features [Electronic resource] / Lyrasis Wiki. — Mode of access:(https://wiki.lyrasis.org/display/DSPACECRIS/).
21. Faculty Success: Capture faculty information once — use it infinitely [Electronic resource] / Watermark Insights. — Mode of access: https://www.watermarkinsights.com/solutions/faculty-success/.
22. Indiana University selects Symplectic Elements as faculty activity reporting system [Electronic resource] / Digital Science Blog. — 2024. — Mode of access: https://www.digital-science.com/blog/2024/09/indiana-university-selects-symplectic-elements/.
23. Student Information System Market Share, Size, Trends, Industry Analysis Report [Electronic resource] / Polaris Market Research. — 2024. — Mode of access: https://www.polarismarketresearch.com/industry-analysis/student-information-system-market.
24. Національна електронна науково-інформаційна система «URIS» [Електронний ресурс] / ДНТБ України. — Київ. — Режим доступу: https://dntb.gov.ua/completed-projects/urisinfo.
25. Driving National R&D: Methodological Insights into Developing a Classifier for the Ukrainian National CRIS System [Electronic resource] / I. Tsybenko, S. Zherebchuk, A. Fedchuk // University Library at a New Stage of Social Communications Development. — 2024. — Mode of access: https://unilibnsd.ust.edu.ua/article/view/315157.
26. Рейтинги викладачів на intellect.kpi.ua [Електронний ресурс] / КБ ІС КПІ ім. Ігоря Сікорського. — 2024. — Режим доступу: https://kbis.kpi.ua/news/2024-10-11.

Додаток А

Інструкція користувача

1) Як зайти в систему (для всіх ролей)

1. Відкрийте сайт у браузері (Chrome/Edge).
2. Ви одразу потрапите на сторінку входу (Login).
3. У поле Email введіть вашу пошту.
4. У поле Пароль введіть пароль.
5. Натисніть кнопку “Увійти”.



Після входу зверху буде меню:

- Головна
- Активності
- Рейтинг
- (для Завідувача/Адміна) Модерація
- (для Адміна) Адміністрування
- справа зверху ваше ім’я → Профіль / Вийти

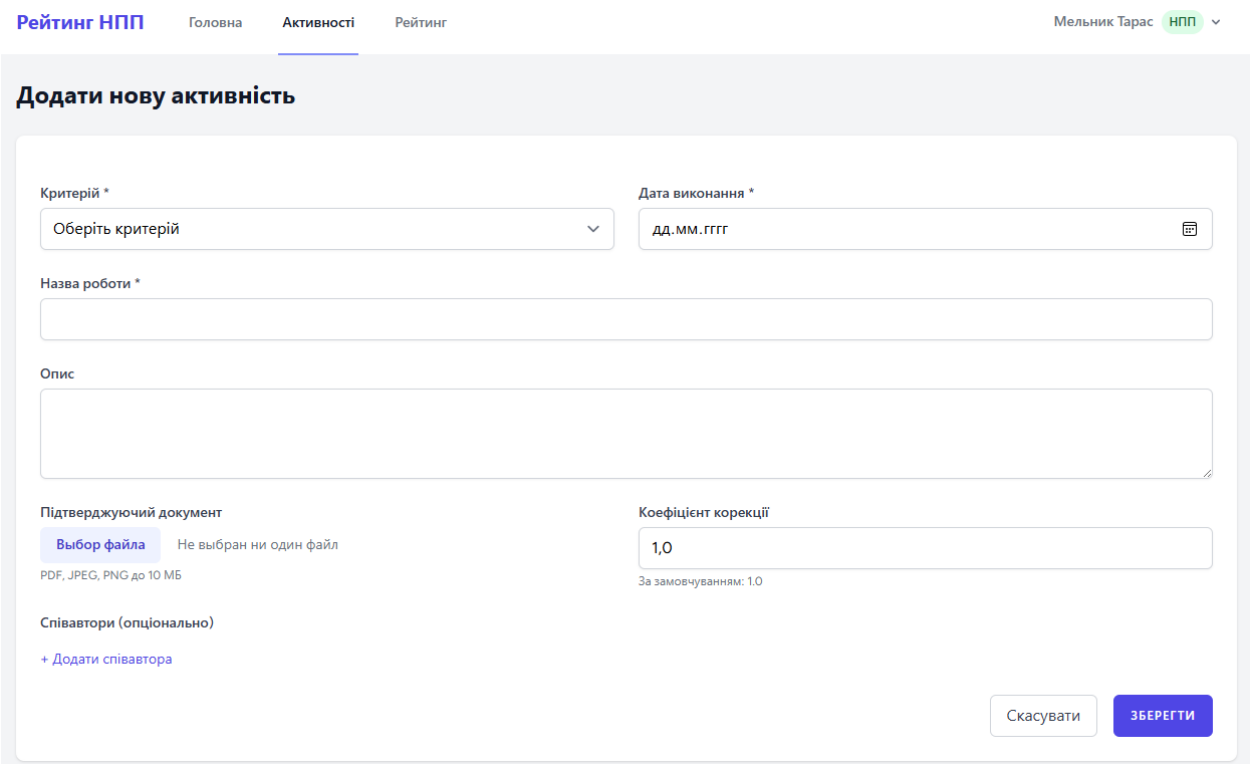
Якщо не пускає

- *Якщо бачите повідомлення "Ваш акаунт деактивовано" — зверніться до адміністратора.*
- *Якщо після входу бачите "Профіль не створено. Зверніться до адміністратора системи." — це означає, що вам створили акаунт, але не створили профіль працівника (Employee).*

2) Інструкція для НПП (викладач) (Email - npp1@university.ua, пароль – password)

2.1. Додати нову активність (досягнення)

1. Натисніть у меню "Активності".
2. Справа зверху натисніть "Додати активність".
3. Заповніть форму:
    - Критерій → оберіть зі списку (показує і назву, і бали).
    - Дата виконання → виберіть у календарі.
    - Назва роботи → коротка назва (обов'язково).
    - Опис → за потреби.
    - Підтверджуючий документ → натисніть "Вибрати файл" і додайте файл (PDF/JPG/PNG, до 10 МБ).
    - Коефіцієнт корекції → зазвичай лишайте 1.0 (міняти тільки якщо вам сказали).
5. Якщо є співавтори:
    - Натисніть "+ Додати співавтора"
    - Оберіть людину зі списку
    - Вкажіть % внеску цієї людини
    - (можна додати кількох)
6. Натисніть "Зберегти".



Після цього активність буде зі статусом “На розгляді”.

2.2. Перевірити статус / знайти потрібну активність

1. Меню “Активності”.
2. У фільтрах зверху можна вибрати:
    - Всі статуси / На розгляді / Затверджено / Відхилено
    - або конкретний критерій
4. Натисніть “Фільтрувати”.

2.3. Переглянути або відредагувати активність

1. У списку натисніть “Переглянути”.
2. Якщо активність ще “На розгляді”, з’явиться кнопка “Редагувати” — натисніть і виправте.
3. Якщо активність вже затверджена/відхилена, редагування недоступне.

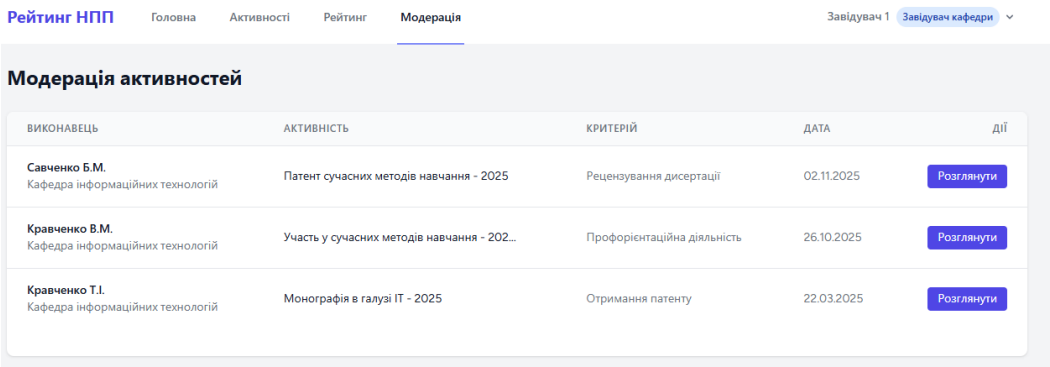
2.4. Завантажити свій підтверджуючий документ

1. Відкрийте активність (“Переглянути”).
2. Натисніть “Завантажити підтверджуючий документ”.

3) Інструкція для Завідувача кафедри (Head) (Email - head1@university.ua, пароль – password)

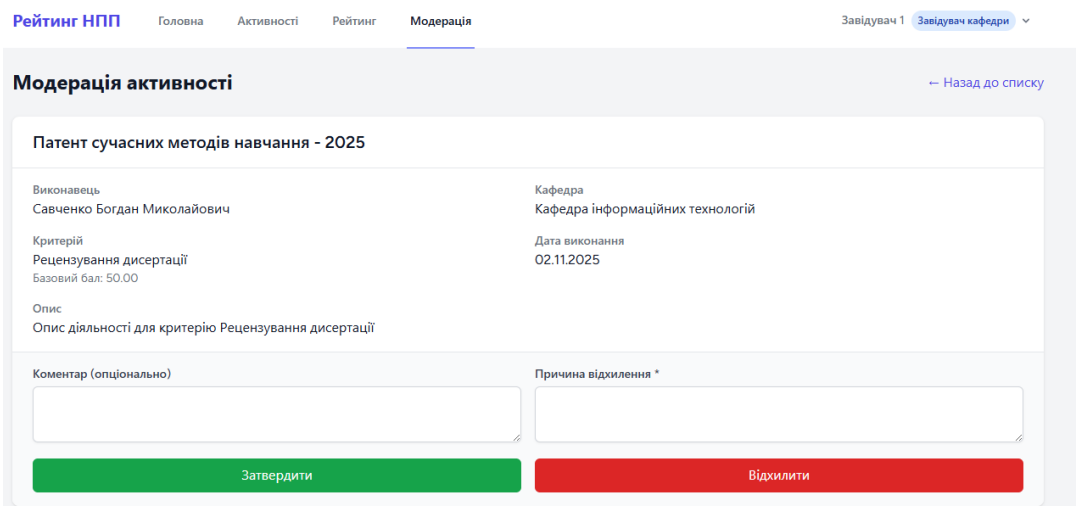
3.1. Подивитися, що треба промодерувати

1. Натисніть “Модерація” в меню.
2. Ви побачите список активностей, які очікують рішення.
3. Натисніть кнопку “Розглянути” біля потрібного запису.



3.2. Як затвердити активність

1. На сторінці модерації перевірте: хто виконавець, критерій і базовий бал, дату, опис, співавторів, файл (за потреби натисніть “Завантажити підтверджуючий документ”)
2. У лівому блоці:
    - (опціонально) напишіть коментар
    - натисніть “Затвердити”



3.3. Як відхилити активність

1. У правому блоці обов’язково впишіть “Причина відхилення”
2. Натисніть “Відхилити”

4) Інструкція для Адміністратора (Admin) (Email - admin1@university.ua, пароль – password)

4.1. “Перший запуск” системи (що треба налаштувати один раз)

Щоб система працювала “з нуля”, адміністратор робить так:

Крок 1 — Створити кафедри

1. Адміністрування → Кафедри
2. Створити кафедру
3. Заповнити:
    - Код кафедри
    - Назва
    - (необов’язково) Завідувач (можна призначити пізніше)
4. Створити

Крок 2 — Додати працівників (це одразу створює їм логін/пароль)

1. Адміністрування → Працівники
2. Додати працівника
3. Заповнити:
    - ПІБ
    - Email

- Пароль
- Кафедра
- Роль (НПП / Завідувач кафедри / Адміністратор)
- Посада та інше

4. Натиснути “Створити”

Якщо у НПП після входу пише “Профіль не створено” — значить працівника додали не через “Працівники”, або профіль не прив’язався. Виправляється створенням/прив’язкою в адмінці.

Крок 3 — Додати критерії

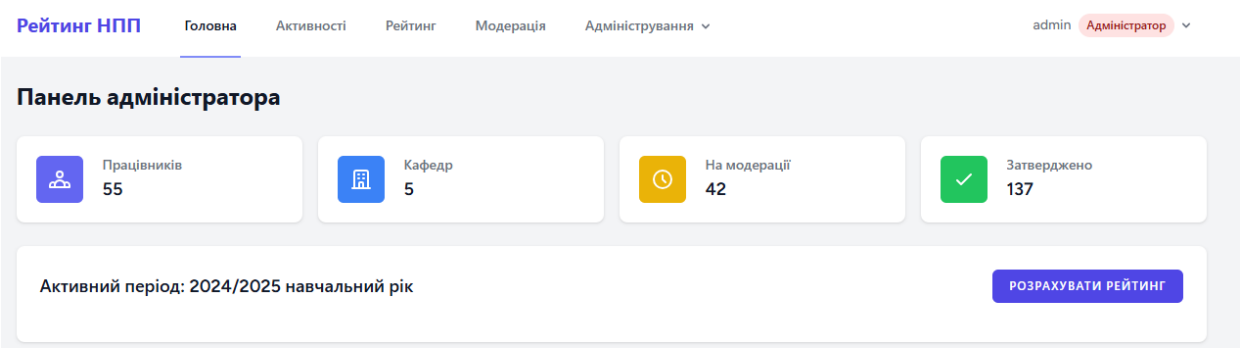
1. Адміністрування → Критерії
2. Додати
3. Заповнити:
   - назва/опис
   - категорія (science/teaching/organization)
   - base_score
   - ваги по напрямах
4. Зберегти

Крок 4 — Створити період рейтингування і зробити його активним

1. Адміністрування → Періоди
2. Додати період
3. Заповнити назву та дати, ваги періоду
4. Після створення натиснути “Активувати” (щоб був активний період)

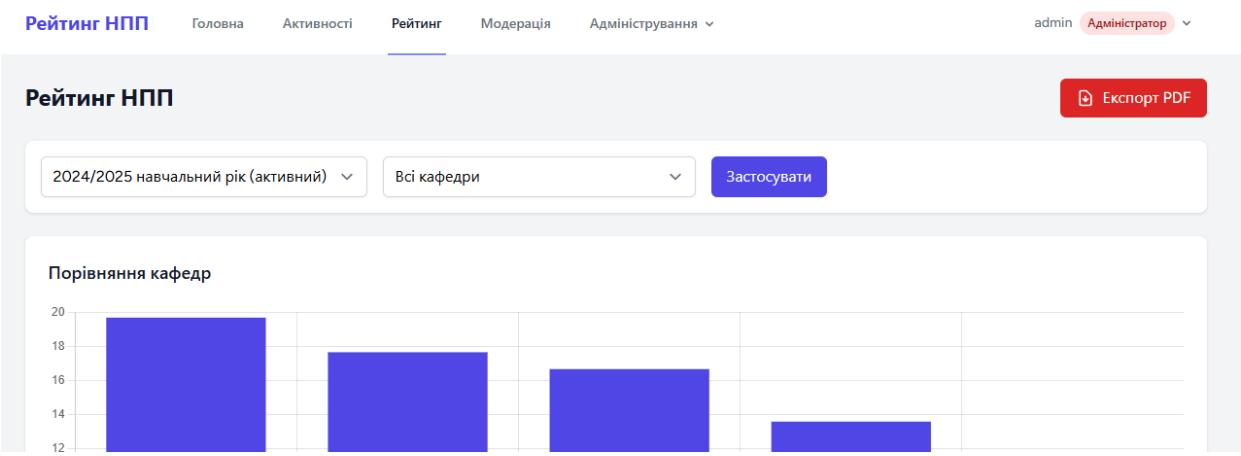
4.2. Як запустити розрахунок рейтингу

1. Перейдіть на Головна (панель адміністратора)
2. Якщо активний період є — буде блок “Активний період: …”
3. Натисніть “Розрахувати рейтинг”
4. Потім зайдіть у “Рейтинг” і перевірте таблицю.



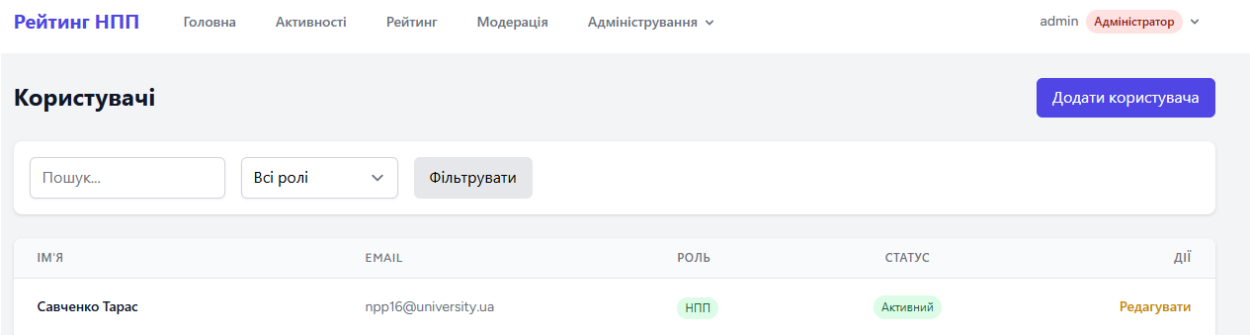
4.3. Як подивитися рейтинг і зробити PDF

1. Меню “Рейтинг”
2. Виберіть:
    - Період
    - (за потреби) Кафедра
3. Натисніть “Застосувати”
4. Натисніть “Експорт PDF”



4.4. Як заблокувати/розблокувати користувача

1. Адміністрування → Користувачі
2. У колонці Статус натисніть кнопку “Активний/Неактивний”
3. Якщо зробили “Неактивний” — при наступній спробі входу користувача система покаже, що акаунт деактивовано.



5) Найчастіші питання (дуже коротко)

- Де додати досягнення? → Активності → Додати активність
- Де подивитися рейтинг? → Рейтинг
- Чому рейтингу немає? → адмін не натиснув “Розрахувати рейтинг” або нема активного періоду
- Де модерація? → у меню “Модерація” (тільки Head/Admin)
- Не дає завантажити файл → у вас немає прав або файл не прикріплений

Додаток Б

Текст програми

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     */
    public function create(): View
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     */
    public function store(LoginRequest $request): RedirectResponse
    {
        $request->authenticate();

        $request->session()->regenerate();

        return    redirect()->intended(route('dashboard',    absolute:
false));
    }

    /**
     * Destroy an authenticated session.
     */
    public function destroy(Request $request): RedirectResponse
    {
        Auth::guard('web')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\ValidationException;
use Illuminate\View\View;

class ConfirmablePasswordController extends Controller
{
    /**
     * Show the confirm password view.
     */
    public function show(): View
    {
        return view('auth.confirm-password');
    }

    /**
     * Confirm the user's password.
     */
    public function store(Request $request): RedirectResponse
    {
        if (! Auth::guard('web')->validate([
            'email' => $request->user()->email,
            'password' => $request->password,
        ])) {
            throw ValidationException::withMessages([
                'password' => __('auth.password'),
            ]);
        }

        $request->session()->put('auth.password_confirmed_at', time());

        return    redirect()->intended(route('dashboard',    absolute:
false));
    }
}


<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class EmailVerificationNotificationController extends Controller
{
    /**
     * Send a new email verification notification.
     */
    public function store(Request $request): RedirectResponse
    {
```

```
        if ($request->user()->hasVerifiedEmail()) {
            return  redirect()->intended(route('dashboard',  absolute:
false));
        }

        $request->user()->sendEmailVerificationNotification();

        return back()->with('status', 'verification-link-sent');
    }
}


<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\View\View;

class EmailVerificationPromptController extends Controller
{
    /**
     * Display the email verification prompt.
     */
    public function __invoke(Request $request): RedirectResponse|View
    {
        return $request->user()->hasVerifiedEmail()
                    ? redirect()->intended(route('dashboard', absolute:
false))
                    : view('auth.verify-email');
    }
}
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\PasswordReset;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Password;
use Illuminate\Support\Str;
use Illuminate\Validation\Rules;
use Illuminate\View\View;

class NewPasswordController extends Controller
{
    /**
     * Display the password reset view.
     */
    public function create(Request $request): View
    {
        return view('auth.reset-password', ['request' => $request]);
    }
```

```
    /**
     * Handle an incoming new password request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'token' => ['required'],
            'email' => ['required', 'email'],
            'password'       =>       ['required',       'confirmed',
Rules\Password::defaults()],
        ]);

        // Here we will attempt to reset the user's password. If it is
successful we
        // will update the password on an actual user model and persist
it to the
        // database. Otherwise we will parse the error and return the
response.
        $status = Password::reset(
            $request->only('email',                          'password',
'password_confirmation', 'token'),
            function (User $user) use ($request) {
                $user->forceFill([
                    'password' => Hash::make($request->password),
                    'remember_token' => Str::random(60),
                ])->save();

                event(new PasswordReset($user));
            }
        );

        // If the password was successfully reset, we will redirect the
user back to
        // the application's home authenticated view. If there is an
error we can
        // redirect them back to where they came from with their error
message.
        return $status == Password::PASSWORD_RESET
                    ?       redirect()->route('login')->with('status',
__($status))
                    : back()->withInput($request->only('email'))
                        ->withErrors(['email' => __($status)]);
    }
}
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;

class PasswordController extends Controller
```

```
{
    /**
     * Update the user's password.
     */
    public function update(Request $request): RedirectResponse
    {
        $validated = $request->validateWithBag('updatePassword', [
            'current_password' => ['required', 'current_password'],
            'password'     =>     ['required',     Password::defaults(),
'confirmed'],
        ]);

        $request->user()->update([
            'password' => Hash::make($validated['password']),
        ]);

        return back()->with('status', 'password-updated');
    }
}
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Password;
use Illuminate\View\View;

class PasswordResetLinkController extends Controller
{
    /**
     * Display the password reset link request view.
     */
    public function create(): View
    {
        return view('auth.forgot-password');
    }

    /**
     * Handle an incoming password reset link request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'email' => ['required', 'email'],
        ]);

        // We will send the password reset link to this user. Once we
have attempted
        // to send the link, we will examine the response then see the
message we
        // need to show to the user. Finally, we'll send out a proper
response.
        $status = Password::sendResetLink(
            $request->only('email')
```

```
        );

        return $status == Password::RESET_LINK_SENT
                    ? back()->with('status', __($status))
                    : back()->withInput($request->only('email'))
                        ->withErrors(['email' => __($status)]);
    }
}
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use Illuminate\View\View;

class RegisteredUserController extends Controller
{
    /**
     * Display the registration view.
     */
    public function create(): View
    {
        return view('auth.register');
    }

    /**
     * Handle an incoming registration request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email',
'max:255', 'unique:'.User::class],
            'password'       =>       ['required',       'confirmed',
Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        event(new Registered($user));

        Auth::login($user);

        return redirect(route('dashboard', absolute: false));
```

```
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\Activity;
use App\Models\Criterion;
use App\Models\Employee;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class ActivityController extends Controller
{
    public function index(Request $request)
    {
        $user = $request->user();
        $query      =      Activity::with(['employee',      'criterion',
'moderator']);

        if ($user->isNpp()) {
            $employee = $user->employee;
            if (!$employee) {
                return    redirect()->route('dashboard')->with('error',
'Профіль не знайдено');
            }
            $query->where('employee_id', $employee->id);
        } elseif ($user->isHead()) {
            $department = $user->headOfDepartment;
            if ($department) {
                $query->whereHas('employee',      fn($q)      =>      $q-
>where('department_id', $department->id));
            }
        }

        if ($request->status) {
            $query->where('status', $request->status);
        }

        if ($request->criterion_id) {
            $query->where('criterion_id', $request->criterion_id);
        }

        $activities = $query->latest()->paginate(15);
        $criterions = Criterion::where('is_active', true)->get();

        return      view('activities.index',      compact('activities',
'criterions'));
    }

    public function create()
    {
        $user = auth()->user();

        if (!$user->isNpp()) {
```

```
            return       redirect()->route('dashboard')->with('error',
'Тільки НПП можуть додавати активності');
        }

        $employee = $user->employee;

        if (!$employee) {
            return       redirect()->route('dashboard')->with('error',
'Профіль не знайдено');
        }

        $criterions     =     Criterion::where('is_active',     true)-
>orderBy('category')->get();
        $employees = Employee::where('id', '!=', $employee->id)->get();

        return     view('activities.create',     compact('criterions',
'employees', 'employee'));
    }

    public function store(Request $request)
    {
        $user = $request->user();

        if (!$user->isNpp()) {
            return       redirect()->route('dashboard')->with('error',
'Тільки НПП можуть додавати активності');
        }

        $employee = $user->employee;

        if (!$employee) {
            return       redirect()->route('dashboard')->with('error',
'Профіль не знайдено');
        }

        $validated = $request->validate([
            'criterion_id' => 'required|exists:criterions,id',
            'title' => 'required|string|max:500',
            'description' => 'nullable|string',
            'date_of_activity' => 'required|date',
            'file' => 'nullable|file|mimes:pdf,jpeg,jpg,png|max:10240',
            'correction_factor' => 'nullable|numeric|min:0.1|max:2',
            'co_authors' => 'nullable|array',
            'co_authors.*.employee_id' => 'exists:employees,id',
            'co_authors.*.contribution_share'                          =>
'numeric|min:0|max:100',
        ], [
            'criterion_id.required' => 'Виберіть критерій',
            'title.required' => 'Назва роботи обов\'язкова',
            'date_of_activity.required' => 'Дата обов\'язкова',
            'file.max' => 'Файл не повинен перевищувати 10 МБ',
            'file.mimes' => 'Дозволені формати: PDF, JPEG, PNG',
        ]);

        $filePath = null;
        if ($request->hasFile('file')) {
            $filePath   =   $request->file('file')->store('activities',
'private');
```

```
        }

        $activity = Activity::create([
            'employee_id' => $employee->id,
            'criterion_id' => $validated['criterion_id'],
            'title' => $validated['title'],
            'description' => $validated['description'] ?? null,
            'date_of_activity' => $validated['date_of_activity'],
            'file_path' => $filePath,
            'correction_factor' => $validated['correction_factor'] ??
1.0,
            'status' => 'pending',
        ]);

        if (!empty($validated['co_authors'])) {
            foreach ($validated['co_authors'] as $coAuthor) {
                if (!empty($coAuthor['employee_id']) &&
!empty($coAuthor['contribution_share'])) {
                    $activity->coAuthors()-
>attach($coAuthor['employee_id'], [
                        'contribution_share' =>
$coAuthor['contribution_share'],
                    ]);
                }
            }
        }

        return redirect()->route('activities.index')
            ->with('success', 'Активність успішно додано. Очікує на
модерацію.');
    }

    public function show(Activity $activity)
    {
        $this->authorizeView($activity);
        $activity->load(['employee', 'criterion', 'moderator',
'coAuthors']);

        return view('activities.show', compact('activity'));
    }

    public function edit(Activity $activity)
    {
        $this->authorizeEdit($activity);

        $criterions = Criterion::where('is_active', true)-
>orderBy('category')->get();
        $employees = Employee::where('id', '!=', $activity-
>employee_id)->get();

        return view('activities.edit', compact('activity',
'criterions', 'employees'));
    }

    public function update(Request $request, Activity $activity)
    {
        $this->authorizeEdit($activity);
```

```
        $validated = $request->validate([
            'criterion_id' => 'required|exists:criterions,id',
            'title' => 'required|string|max:500',
            'description' => 'nullable|string',
            'date_of_activity' => 'required|date',
            'file' => 'nullable|file|mimes:pdf,jpeg,jpg,png|max:10240',
            'correction_factor' => 'nullable|numeric|min:0.1|max:2',
            'co_authors' => 'nullable|array',
            'co_authors.*.employee_id' => 'exists:employees,id',
            'co_authors.*.contribution_share'                          =>
'numeric|min:0|max:100',
        ]);

        if ($request->hasFile('file')) {
            if ($activity->file_path) {
                Storage::disk('private')->delete($activity-
>file_path);
            }
            $validated['file_path']      =      $request->file('file')-
>store('activities', 'private');
        }

        $activity->update([
            'criterion_id' => $validated['criterion_id'],
            'title' => $validated['title'],
            'description' => $validated['description'] ?? null,
            'date_of_activity' => $validated['date_of_activity'],
            'file_path'   =>   $validated['file_path']   ??   $activity-
>file_path,
            'correction_factor'  =>  $validated['correction_factor']  ??
1.0,
            'status' => 'pending',
            'moderator_id' => null,
            'moderation_comment' => null,
            'moderation_date' => null,
        ]);

        $activity->coAuthors()->detach();
        if (!empty($validated['co_authors'])) {
            foreach ($validated['co_authors'] as $coAuthor) {
                if         (!empty($coAuthor['employee_id'])         &&
!empty($coAuthor['contribution_share'])) {
                    $activity->coAuthors()-
>attach($coAuthor['employee_id'], [
                        'contribution_share'                           =>
$coAuthor['contribution_share'],
                    ]);
                }
            }
        }

        return redirect()->route('activities.index')
            ->with('success', 'Активність успішно оновлено');
    }

    public function destroy(Activity $activity)
    {
        $this->authorizeEdit($activity);
```

```
        if ($activity->file_path) {
            Storage::disk('private')->delete($activity->file_path);
        }

        $activity->delete();

        return redirect()->route('activities.index')
            ->with('success', 'Активність успішно видалено');
    }

    public function downloadFile(Activity $activity)
    {
        $this->authorizeView($activity);

        if    (!$activity->file_path    ||    !Storage::disk('private')-
>exists($activity->file_path)) {
            abort(404, 'Файл не знайдено');
        }

        return            Storage::disk('private')->download($activity-
>file_path);
    }

    protected function authorizeView(Activity $activity): void
    {
        $user = auth()->user();

        if ($user->isAdmin()) {
            return;
        }

        if ($user->isHead()) {
            $department = $user->headOfDepartment;
            if  ($department  &&  $activity->employee->department_id  ===
$department->id) {
                return;
            }
        }

        if ($activity->employee_id === $user->employee?->id) {
            return;
        }

        abort(403, 'Доступ заборонено');
    }

    protected function authorizeEdit(Activity $activity): void
    {
        $user = auth()->user();

        if ($user->isAdmin()) {
            return;
        }

        if   ($activity->employee_id   ===   $user->employee?->id   &&
$activity->status === 'pending') {
            return;
```

```
        }

        abort(403, 'Редагування заборонено');
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\Criterion;
use Illuminate\Http\Request;

class CriterionController extends Controller
{
    public function index(Request $request)
    {
        $query = Criterion::query();

        if ($request->category) {
            $query->where('category', $request->category);
        }

        if ($request->search) {
            $query->where('name', 'ilike', "%{$request->search}%");
        }

        $criterions   =   $query->orderBy('category')->orderBy('name')-
>paginate(15);

        return view('criterions.index', compact('criterions'));
    }

    public function create()
    {
        return view('criterions.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
            'category' => 'required|in:science,teaching,organization',
            'base_score' => 'required|numeric|min:0',
            'weight_science' => 'required|numeric|min:0|max:1',
            'weight_teaching' => 'required|numeric|min:0|max:1',
            'weight_organization' => 'required|numeric|min:0|max:1',
            'is_active' => 'boolean',
        ], [
            'name.required' => 'Назва критерію обов\'язкова',
            'category.required' => 'Категорія обов\'язкова',
            'base_score.required' => 'Базовий бал обов\'язковий',
        ]);

        $validated['is_active'] = $request->has('is_active');
```

```
        Criterion::create($validated);

        return redirect()->route('criterions.index')
            ->with('success', 'Критерій успішно створено');
    }

    public function show(Criterion $criterion)
    {
        return view('criterions.show', compact('criterion'));
    }

    public function edit(Criterion $criterion)
    {
        return view('criterions.edit', compact('criterion'));
    }

    public function update(Request $request, Criterion $criterion)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
            'category' => 'required|in:science,teaching,organization',
            'base_score' => 'required|numeric|min:0',
            'weight_science' => 'required|numeric|min:0|max:1',
            'weight_teaching' => 'required|numeric|min:0|max:1',
            'weight_organization' => 'required|numeric|min:0|max:1',
            'is_active' => 'boolean',
        ]);

        $validated['is_active'] = $request->has('is_active');

        $criterion->update($validated);

        return redirect()->route('criterions.index')
            ->with('success', 'Критерій успішно оновлено');
    }

    public function destroy(Criterion $criterion)
    {
        if ($criterion->activities()->exists()) {
            return back()->with('error', 'Неможливо видалити критерій з
активностями');
        }

        $criterion->delete();

        return redirect()->route('criterions.index')
            ->with('success', 'Критерій успішно видалено');
    }
}



<?php

namespace App\Http\Controllers;

use App\Models\Activity;
```

```
use App\Models\Department;
use App\Models\Employee;
use App\Models\RatingPeriod;
use App\Models\RatingScore;
use Illuminate\Http\Request;

class DashboardController extends Controller
{
    public function index(Request $request)
    {
        $user = $request->user();

        if ($user->isAdmin()) {
            return $this->adminDashboard();
        }

        if ($user->isHead()) {
            return $this->headDashboard($user);
        }

        return $this->nppDashboard($user);
    }

    protected function adminDashboard()
    {
        $activePeriod = RatingPeriod::getActive();

        $stats = [
            'total_employees' => Employee::count(),
            'total_departments' => Department::count(),
            'pending_activities' => Activity::pending()->count(),
            'approved_activities' => Activity::approved()->count(),
        ];

        $topRatings = [];
        $departmentStats = [];

        if ($activePeriod) {
            $topRatings        =        RatingScore::where('period_id',
$activePeriod->id)
                ->with('employee.department')
                ->orderByDesc('score_final')
                ->limit(10)
                ->get();

            $departmentStats                                          =
Department::with(['employees.ratingScores'    =>    function($q)    use
($activePeriod) {
                $q->where('period_id', $activePeriod->id);
            }])->get()->map(function($dept) {
                $avgScore   =   $dept->employees->flatMap->ratingScores-
>avg('score_final') ?? 0;
                return [
                    'name' => $dept->name,
                    'avg_score' => round($avgScore, 2),
                    'employee_count' => $dept->employees->count(),
                ];
            });
```

```
        }

        return view('dashboard.admin', compact('stats', 'topRatings',
'departmentStats', 'activePeriod'));
    }

    protected function headDashboard($user)
    {
        $department = $user->headOfDepartment;
        $activePeriod = RatingPeriod::getActive();

        if (!$department) {
            return $this->nppDashboard($user);
        }

        $pendingActivities       =       Activity::whereHas('employee',
function($q) use ($department) {
            $q->where('department_id', $department->id);
        })->pending()->with(['employee',       'criterion'])->latest()-
>limit(10)->get();

        $departmentRatings = [];
        if ($activePeriod) {
            $departmentRatings    =    RatingScore::where('period_id',
$activePeriod->id)
                ->whereHas('employee', function($q) use ($department) {
                    $q->where('department_id', $department->id);
                })
                ->with('employee')
                ->orderByDesc('score_final')
                ->get();
        }

        $stats = [
            'total_employees' => $department->employees()->count(),
            'pending_activities'    =>    Activity::whereHas('employee',
function($q) use ($department) {
                $q->where('department_id', $department->id);
            })->pending()->count(),
        ];

        return       view('dashboard.head',       compact('department',
'pendingActivities', 'departmentRatings', 'stats', 'activePeriod'));
    }

    protected function nppDashboard($user)
    {
        $employee = $user->employee;
        $activePeriod = RatingPeriod::getActive();

        if (!$employee) {
            return view('dashboard.no-profile');
        }

        $myActivities = Activity::where('employee_id', $employee->id)
            ->with('criterion')
            ->latest()
            ->limit(10)
```

```
            ->get();

        $myRating = null;
        $chartData = null;

        if ($activePeriod) {
            $myRating = RatingScore::where('employee_id', $employee-
>id)
                ->where('period_id', $activePeriod->id)
                ->first();

            if ($myRating) {
                $chartData = [
                    'labels' => ['Наукова діяльність', 'Навчально-
методична', 'Організаційно-виховна'],
                    'data' => [
                        round($myRating->score_normalized_science, 2),
                        round($myRating->score_normalized_teaching, 2),
                        round($myRating-
>score_normalized_organization, 2),
                    ],
                ];
            }
        }

        $activityStats = [
            'pending' => $employee->activities()->pending()->count(),
            'approved' => $employee->activities()->approved()->count(),
            'rejected' => $employee->activities()->rejected()->count(),
        ];

        return view('dashboard.npp', compact('employee',
'myActivities', 'myRating', 'chartData', 'activityStats',
'activePeriod'));
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\Department;
use App\Models\User;
use Illuminate\Http\Request;

class DepartmentController extends Controller
{
    public function index()
    {
        $departments = Department::with('head', 'employees')-
>paginate(15);
        return view('departments.index', compact('departments'));
    }

    public function create()
    {
```

```
        $heads = User::where('role', 'head')->orWhere('role', 'admin')-
>get();
        return view('departments.create', compact('heads'));
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'code' => 'required|string|max:50|unique:departments',
            'head_user_id' => 'nullable|exists:users,id',
        ], [
            'name.required' => 'Назва кафедри обов\'язкова',
            'code.required' => 'Код кафедри обов\'язковий',
            'code.unique' => 'Такий код вже існує',
        ]);

        Department::create($validated);

        return redirect()->route('departments.index')
            ->with('success', 'Кафедру успішно створено');
    }

    public function show(Department $department)
    {
        $department->load(['head', 'employees.user']);
        return view('departments.show', compact('department'));
    }

    public function edit(Department $department)
    {
        $heads = User::where('role', 'head')->orWhere('role', 'admin')-
>get();
        return      view('departments.edit',      compact('department',
'heads'));
    }

    public function update(Request $request, Department $department)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'code' => 'required|string|max:50|unique:departments,code,'
. $department->id,
            'head_user_id' => 'nullable|exists:users,id',
        ]);

        $department->update($validated);

        return redirect()->route('departments.index')
            ->with('success', 'Кафедру успішно оновлено');
    }

    public function destroy(Department $department)
    {
        if ($department->employees()->exists()) {
            return back()->with('error', 'Неможливо видалити кафедру з
працівниками');
        }
```

```
        $department->delete();

        return redirect()->route('departments.index')
            ->with('success', 'Кафедру успішно видалено');
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Department;
use App\Models\Employee;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class EmployeeController extends Controller
{
    public function index(Request $request)
    {
        $query = Employee::with(['user', 'department']);

        if ($request->department_id) {
            $query->where('department_id', $request->department_id);
        }

        if ($request->search) {
            $search = $request->search;
            $query->where(function($q) use ($search) {
                $q->where('surname', 'ilike', "%{$search}%")
                  ->orWhere('name', 'ilike', "%{$search}%")
                  ->orWhere('patronymic', 'ilike', "%{$search}%");
            });
        }

        $employees = $query->orderBy('surname')->paginate(15);
        $departments = Department::all();

        return       view('employees.index',       compact('employees',
'departments'));
    }

    public function create()
    {
        $departments = Department::all();
        return view('employees.create', compact('departments'));
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'email' => 'required|email|unique:users,email',
            'password' => 'required|min:8',
            'role' => 'required|in:npp,head,admin',
            'department_id' => 'required|exists:departments,id',
            'surname' => 'required|string|max:255',
```

```
            'name' => 'required|string|max:255',
            'patronymic' => 'nullable|string|max:255',
            'position' => 'required|string|max:255',
            'academic_degree' => 'nullable|string|max:255',
            'academic_title' => 'nullable|string|max:255',
            'hire_date' => 'nullable|date',
            'phone' => 'nullable|string|max:50',
            'office_number' => 'nullable|string|max:50',
        ], [
            'email.required' => 'Email обов\'язковий',
            'email.unique' => 'Цей email вже використовується',
            'password.required' => 'Пароль обов\'язковий',
            'password.min' => 'Пароль має бути не менше 8 символів',
            'surname.required' => 'Прізвище обов\'язкове',
            'name.required' => 'Ім\'я обов\'язкове',
            'position.required' => 'Посада обов\'язкова',
            'department_id.required' => 'Кафедра обов\'язкова',
        ]);

        $user = User::create([
            'name' => "{$validated['surname']} {$validated['name']}",
            'email' => $validated['email'],
            'password' => Hash::make($validated['password']),
            'role' => $validated['role'],
        ]);

        Employee::create([
            'user_id' => $user->id,
            'department_id' => $validated['department_id'],
            'surname' => $validated['surname'],
            'name' => $validated['name'],
            'patronymic' => $validated['patronymic'] ?? null,
            'position' => $validated['position'],
            'academic_degree' => $validated['academic_degree'] ?? null,
            'academic_title' => $validated['academic_title'] ?? null,
            'hire_date' => $validated['hire_date'] ?? null,
            'phone' => $validated['phone'] ?? null,
            'office_number' => $validated['office_number'] ?? null,
        ]);

        return redirect()->route('employees.index')
            ->with('success', 'Працівника успішно створено');
    }

    public function show(Employee $employee)
    {
        $employee->load(['user', 'department', 'activities.criterion',
'ratingScores.period']);
        return view('employees.show', compact('employee'));
    }

    public function edit(Employee $employee)
    {
        $departments = Department::all();
        return       view('employees.edit',       compact('employee',
'departments'));
    }
```

```
    public function update(Request $request, Employee $employee)
    {
        $validated = $request->validate([
            'email'    =>    'required|email|unique:users,email,'    .
$employee->user_id,
            'role' => 'required|in:npp,head,admin',
            'department_id' => 'required|exists:departments,id',
            'surname' => 'required|string|max:255',
            'name' => 'required|string|max:255',
            'patronymic' => 'nullable|string|max:255',
            'position' => 'required|string|max:255',
            'academic_degree' => 'nullable|string|max:255',
            'academic_title' => 'nullable|string|max:255',
            'hire_date' => 'nullable|date',
            'phone' => 'nullable|string|max:50',
            'office_number' => 'nullable|string|max:50',
            'is_active' => 'boolean',
        ]);

        $employee->user->update([
            'name' => "{$validated['surname']} {$validated['name']}",
            'email' => $validated['email'],
            'role' => $validated['role'],
            'is_active' => $request->has('is_active'),
        ]);

        $employee->update([
            'department_id' => $validated['department_id'],
            'surname' => $validated['surname'],
            'name' => $validated['name'],
            'patronymic' => $validated['patronymic'] ?? null,
            'position' => $validated['position'],
            'academic_degree' => $validated['academic_degree'] ?? null,
            'academic_title' => $validated['academic_title'] ?? null,
            'hire_date' => $validated['hire_date'] ?? null,
            'phone' => $validated['phone'] ?? null,
            'office_number' => $validated['office_number'] ?? null,
        ]);

        return redirect()->route('employees.index')
            ->with('success', 'Дані працівника успішно оновлено');
    }

    public function destroy(Employee $employee)
    {
        $user = $employee->user;
        $employee->delete();
        $user->delete();

        return redirect()->route('employees.index')
            ->with('success', 'Працівника успішно видалено');
    }
}


<?php

namespace App\Http\Controllers;
```

```
use App\Models\Activity;
use App\Models\Department;
use Illuminate\Http\Request;

class ModerationController extends Controller
{
    public function index(Request $request)
    {
        $user = $request->user();
        $query = Activity::with(['employee.department', 'criterion'])-
>pending();

        if ($user->isHead()) {
            $department = $user->headOfDepartment;
            if ($department) {
                $query->whereHas('employee',     fn($q)     =>     $q-
>where('department_id', $department->id));
            } else {
                $query->whereRaw('1 = 0');
            }
        }

        if ($request->department_id && $user->isAdmin()) {
            $query->whereHas('employee',       fn($q)       =>       $q-
>where('department_id', $request->department_id));
        }

        $activities = $query->latest()->paginate(15);
        $departments = $user->isAdmin() ? Department::all() : collect();

        return      view('moderation.index',      compact('activities',
'departments'));
    }

    public function show(Activity $activity)
    {
        $this->authorizeModeration($activity);
        $activity->load(['employee.department',            'criterion',
'coAuthors']);

        return view('moderation.show', compact('activity'));
    }

    public function approve(Request $request, Activity $activity)
    {
        $this->authorizeModeration($activity);

        $validated = $request->validate([
            'moderation_comment' => 'nullable|string|max:1000',
        ]);

        $activity->update([
            'status' => 'approved',
            'moderator_id' => $request->user()->id,
            'moderation_comment' => $validated['moderation_comment'] ??
null,
            'moderation_date' => now(),
```

```
        ]);

        return redirect()->route('moderation.index')
            ->with('success', 'Активність затверджено');
    }

    public function reject(Request $request, Activity $activity)
    {
        $this->authorizeModeration($activity);

        $validated = $request->validate([
            'moderation_comment' => 'required|string|max:1000',
        ], [
            'moderation_comment.required'    =>    'Вкажіть    причину
відхилення',
        ]);

        $activity->update([
            'status' => 'rejected',
            'moderator_id' => $request->user()->id,
            'moderation_comment' => $validated['moderation_comment'],
            'moderation_date' => now(),
        ]);

        return redirect()->route('moderation.index')
            ->with('success', 'Активність відхилено');
    }

    protected function authorizeModeration(Activity $activity): void
    {
        $user = auth()->user();

        if ($user->isAdmin()) {
            return;
        }

        if ($user->isHead()) {
            $department = $user->headOfDepartment;
            if ($department && $activity->employee->department_id ===
$department->id) {
                return;
            }
        }

        abort(403, 'Доступ до модерації заборонено');
    }
}


<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
```

```
use Illuminate\View\View;

class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    public function edit(Request $request): View
    {
        return view('profile.edit', [
            'user' => $request->user(),
        ]);
    }

    /**
     * Update the user's profile information.
     */
    public     function     update(ProfileUpdateRequest     $request):
RedirectResponse
    {
        $request->user()->fill($request->validated());

        if ($request->user()->isDirty('email')) {
            $request->user()->email_verified_at = null;
        }

        $request->user()->save();

        return          Redirect::route('profile.edit')->with('status',
'profile-updated');
    }

    /**
     * Delete the user's account.
     */
    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

        Auth::logout();

        $user->delete();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return Redirect::to('/');
    }
}


<?php

namespace App\Http\Controllers;
```

```
use App\Models\Department;
use App\Models\RatingPeriod;
use App\Models\RatingScore;
use App\Services\RatingCalculationService;
use Barryvdh\DomPDF\Facade\Pdf;
use Illuminate\Http\Request;

class RatingController extends Controller
{
    public function index(Request $request)
    {
        $periods = RatingPeriod::orderByDesc('start_date')->get();
        $selectedPeriod = $request->period_id
            ? RatingPeriod::find($request->period_id)
            : RatingPeriod::getActive();

        $ratings = collect();
        $departmentStats = collect();

        if ($selectedPeriod) {
            $query = RatingScore::where('period_id', $selectedPeriod-
>id)
                ->with('employee.department');

            if ($request->department_id) {
                $query->whereHas('employee', fn($q) => $q-
>where('department_id', $request->department_id));
            }

            $ratings = $query->orderByDesc('score_final')-
>paginate(20);

            $departmentStats =
Department::with(['employees.ratingScores' => function($q) use
($selectedPeriod) {
                $q->where('period_id', $selectedPeriod->id);
            }])->get()->map(function($dept) {
                $scores = $dept->employees->flatMap->ratingScores;
                return [
                    'id' => $dept->id,
                    'name' => $dept->name,
                    'avg_score' => round($scores->avg('score_final') ??
0, 2),
                    'max_score' => round($scores->max('score_final') ??
0, 2),
                    'min_score' => round($scores->min('score_final') ??
0, 2),
                    'count' => $dept->employees->count(),
                ];
            })->sortByDesc('avg_score');
        }

        $departments = Department::all();

        return view('ratings.index', compact('ratings', 'periods',
'selectedPeriod', 'departments', 'departmentStats'));
    }
```

```
    public function calculate(Request $request, RatingCalculationService $service)
    {
        $period = RatingPeriod::findOrFail($request->period_id);

        $service->calculateForPeriod($period);

        return redirect()->route('ratings.index', ['period_id' => $period->id])
            ->with('success', 'Рейтинг успішно розраховано');
    }

    public function exportPdf(Request $request)
    {
        $period = $request->period_id
            ? RatingPeriod::findOrFail($request->period_id)
            : RatingPeriod::getActive();

        if (!$period) {
            return back()->with('error', 'Період не знайдено');
        }

        $department = null;
        $query = RatingScore::where('period_id', $period->id)
            ->with('employee.department')
            ->orderByDesc('score_final');

        if ($request->department_id) {
            $department = Department::findOrFail($request->department_id);
            $query->whereHas('employee', fn($q) => $q->where('department_id', $department->id));
        }

        $ratings = $query->get();

        $pdf = Pdf::loadView('ratings.pdf', compact('ratings', 'period', 'department'));
        $pdf->setPaper('a4', 'landscape');

        $filename = $department
            ? "rating_{$department->code}_{$period->id}.pdf"
            : "rating_university_{$period->id}.pdf";

        return $pdf->download($filename);
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\RatingPeriod;
use Illuminate\Http\Request;

class RatingPeriodController extends Controller
```

```
{
    public function index()
    {
        $periods       =       RatingPeriod::orderByDesc('start_date')-
>paginate(15);
        return view('periods.index', compact('periods'));
    }

    public function create()
    {
        return view('periods.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'start_date' => 'required|date',
            'end_date' => 'required|date|after:start_date',
            'weight_science' => 'required|numeric|min:0|max:1',
            'weight_teaching' => 'required|numeric|min:0|max:1',
            'weight_organization' => 'required|numeric|min:0|max:1',
            'is_active' => 'boolean',
        ], [
            'name.required' => 'Назва періоду обов\'язкова',
            'start_date.required' => 'Дата початку обов\'язкова',
            'end_date.required' => 'Дата закінчення обов\'язкова',
            'end_date.after' => 'Дата закінчення має бути пізніше дати
початку',
        ]);

        $validated['is_active'] = $request->has('is_active');

        if ($validated['is_active']) {
            RatingPeriod::where('is_active', true)->update(['is_active'
=> false]);
        }

        RatingPeriod::create($validated);

        return redirect()->route('periods.index')
            ->with('success', 'Період успішно створено');
    }

    public function edit(RatingPeriod $period)
    {
        return view('periods.edit', compact('period'));
    }

    public function update(Request $request, RatingPeriod $period)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'start_date' => 'required|date',
            'end_date' => 'required|date|after:start_date',
            'weight_science' => 'required|numeric|min:0|max:1',
            'weight_teaching' => 'required|numeric|min:0|max:1',
            'weight_organization' => 'required|numeric|min:0|max:1',
```

```
            'is_active' => 'boolean',
        ]);

        $validated['is_active'] = $request->has('is_active');

        if ($validated['is_active'] && !$period->is_active) {
            RatingPeriod::where('is_active', true)->update(['is_active'
=> false]);
        }

        $period->update($validated);

        return redirect()->route('periods.index')
            ->with('success', 'Період успішно оновлено');
    }

    public function destroy(RatingPeriod $period)
    {
        if ($period->ratingScores()->exists()) {
            return back()->with('error', 'Неможливо видалити період з
розрахованими рейтингами');
        }

        $period->delete();

        return redirect()->route('periods.index')
            ->with('success', 'Період успішно видалено');
    }

    public function setActive(RatingPeriod $period)
    {
        RatingPeriod::where('is_active', true)->update(['is_active' =>
false]);
        $period->update(['is_active' => true]);

        return back()->with('success', 'Період активовано');
    }
}


<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index(Request $request)
    {
        $query = User::query();

        if ($request->role) {
            $query->where('role', $request->role);
        }
```

```
        if ($request->search) {
            $query->where(function($q) use ($request) {
                $q->where('name', 'ilike', "%{$request->search}%")
                  ->orWhere('email', 'ilike', "%{$request->search}%");
            });
        }

        $users = $query->orderBy('name')->paginate(15);

        return view('users.index', compact('users'));
    }

    public function create()
    {
        return view('users.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|unique:users,email',
            'password' => 'required|min:8|confirmed',
            'role' => 'required|in:npp,head,admin',
        ], [
            'name.required' => 'Ім\'я обов\'язкове',
            'email.required' => 'Email обов\'язковий',
            'email.unique' => 'Цей email вже використовується',
            'password.required' => 'Пароль обов\'язковий',
            'password.min' => 'Пароль має бути не менше 8 символів',
            'password.confirmed' => 'Паролі не співпадають',
        ]);

        User::create([
            'name' => $validated['name'],
            'email' => $validated['email'],
            'password' => Hash::make($validated['password']),
            'role' => $validated['role'],
        ]);

        return redirect()->route('users.index')
            ->with('success', 'Користувача успішно створено');
    }

    public function edit(User $user)
    {
        return view('users.edit', compact('user'));
    }

    public function update(Request $request, User $user)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|unique:users,email,' . $user-
>id,
            'password' => 'nullable|min:8|confirmed',
            'role' => 'required|in:npp,head,admin',
            'is_active' => 'boolean',
```

```
        ]);

        $data = [
            'name' => $validated['name'],
            'email' => $validated['email'],
            'role' => $validated['role'],
            'is_active' => $request->has('is_active'),
        ];

        if (!empty($validated['password'])) {
            $data['password'] = Hash::make($validated['password']);
        }

        $user->update($data);

        return redirect()->route('users.index')
            ->with('success', 'Користувача успішно оновлено');
    }

    public function destroy(User $user)
    {
        if ($user->id === auth()->id()) {
            return back()->with('error', 'Неможливо видалити власний
акаунт');
        }

        $user->delete();

        return redirect()->route('users.index')
            ->with('success', 'Користувача успішно видалено');
    }

    public function toggleActive(User $user)
    {
        if ($user->id === auth()->id()) {
            return back()->with('error', 'Неможливо деактивувати
власний акаунт');
        }

        $user->update(['is_active' => !$user->is_active]);

        $status = $user->is_active ? 'активовано' : 'деактивовано';
        return back()->with('success', "Користувача {$status}");
    }
}
```