Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

на тему: «Проєктування та створення веб-застосунку експертного оцінювання якості програмного забезпечення»

Виконав: студент IV курсу, групи СП-43
спеціальності 121 «Інженерія програмного забезпечення»

	(підпис)	Шарун М.О. (прізвище та ініціали)
Керівник	(підпис)	Бревус Г.Б. (прізвище та ініціали)
Консультант	(підпис)	Петрик М.Р. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Загородна Н.В. (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
Кафедра *програмної інженерії*

ЗАТВЕРДЖУЮ
Завідувач кафедри *Петрик М.Р.*

«*06*» *квітня* 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *121 «Інженерія програмного забезпечення»*
(шифр і назва спеціальності)

студенту *Шаруну Максиму Олександровичу*
(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) *Проєктування та створення веб-застосунку експертного оцінювання якості програмного забезпечення*

Керівник проекту (роботи) *Бревус Галина Богданівна*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року №_____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *Моделі якості програмного забезпечення, ролі користувачів, пехнології розробки програмного забезпечення, C#, ASP.NET, MS SQL Server*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та загальних вимог до веб-застосунку експертного оцінювання якості програмного забезпечення. 2. Проєктування та розробка веб-застосунку експертного оцінювання якості програмного забезпечення. 3. Реалізація та тестування веб-застосунку експертного оцінювання якості програмного забезпечення 4. Безпека життє-діяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета роботи. 2. Характеристики якості програмного забезпечення. 3. Функціональні та нефункціональні вимоги до веб-застосунку. 4. Діаграми прецедентів. 5. Архітектура веб-застосунку. 6. ER-діаграма бази даних. 7. Ролі користувачів у веб-додатку 8. Інтерфейс користувачів в залежності від ролі. 7.Результати оцінювання якості ПЗ при використанні експертних технологій. 9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Гурик О.Я., доц. каф. МТ*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*6.04 – 12.04*	
2.	*Робота над першим розділом «Аналіз предметної області та загальних вимог до веб-застосунку експертного оцінювання якості програмного забезпечення»*	*13.04 – 26.04*	
3.	*Робота над другим розділом «Проєктування та розробка веб-застосунку експертного оцінювання якості програмного забезпечення»*	*27.04 – 03.05*	
4.	*Робота над третім розділом «Реалізація та тестування веб-застосунку експертного оцінювання якості програмного забезпечення»*	*04.05 – 17.05*	
5.	*Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»*	*18.05 – 24.05*	
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*25.05 – 7.06*	
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*8.06 – 14.06*	
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	
9	*Захист кваліфікаційної роботи бакалавра*		

Студент _____ *Шарун М.О.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Бревус Г.Б.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Проєктування та створення веб-застосунку експертного оцінювання якості програмного забезпечення // Кваліфікаційна робота освітнього рівня «Бакалавр» // Шарун Максим Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-43 // Тернопіль, 2026 // С. 79, рис. – 13, табл. – 10, ліст. – 14, бібліогр. – 34.

Ключові слова: програмне забезпечення; якість програмного забезпечення; експертне оцінювання; веб-застосунок; критерії оцінювання; ASP.NET MVC; C#; MS SQL Server.

Кваліфікаційна робота присвячена проєктуванню та створенню веб-застосунку експертного оцінювання якості програмного забезпечення.

У першому розділі розглянуто предметну область оцінювання якості програмного забезпечення, проаналізовано основні моделі якості ПЗ та обґрунтовано використання ISO/IEC 25010:2023.

У другому розділі сформовано функціональні та нефункціональні вимоги до веб-застосунку, визначено ролі користувачів, розроблено діаграми варіантів використання, діаграми потоків даних, архітектуру системи та структуру бази даних.

У третьому розділі описано реалізацію веб-застосунку з використанням ASP.NET MVC, мови програмування C# та бази даних MS SQL Server.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці, зокрема заходи захисту населення від впливу радіації та засоби зменшення шкідливого впливу ультразвуку на організм людини.

ABSTRACT

Design and Development of a Web Application for Expert Evaluation of Software Quality // Bachelor's Qualification Work// Sharun Maksym Oleksandrovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-43 // Ternopil, 2026 // P. 79, fig. – 13, tabl. – 10, listings – 14, references – 34.

Keywords: software; software quality; expert evaluation; web application; evaluation criteria; ASP.NET MVC; C#; MS SQL Server.

The qualification work is devoted to the design and development of a web application for expert evaluation of software quality.

The first chapter considers the subject area of software quality evaluation, analyzes the main software quality models, and substantiates the use of ISO/IEC 25010:2023.

The second chapter formulates the functional and non-functional requirements for the web application, defines user roles, and develops use case diagrams, data flow diagrams, the system architecture, and the database structure.

The third chapter describes the implementation of the web application using ASP.NET MVC, the C# programming language, and the MS SQL Server database.

The fourth chapter considers issues of life safety and occupational safety, in particular measures for protecting the population from radiation exposure and means of reducing the harmful effect of ultrasound on the human body.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
ПЗ	Програмне забезпечення
BE	Back End
CASE	Computer Aided Software Engineering
ER	Entity Relations
FE	Front End
UML	Unified Modeling Language

ВСТУП

Сучасна індустрія програмного забезпечення характеризується високою динамікою розвитку, зростанням складності програмних систем та підвищенням вимог до їхньої якості. Програмні продукти застосовуються в управлінні бізнес-процесами, освіті, медицині, фінансовій сфері, електронній комерції, державних сервісах та інших напрямах, де помилки, нестабільність роботи або низька зручність використання можуть призводити до суттєвих організаційних, економічних і репутаційних втрат. У зв’язку з цим питання оцінювання якості програмного забезпечення є одним із важливих напрямів інженерії програмного забезпечення.

Якість програмного забезпечення не обмежується лише правильністю виконання окремих функцій. Вона охоплює функціональну придатність, продуктивність, надійність, зручність використання, захищеність, супроводжуваність, переносимість та інші характеристики. На практиці оцінювання таких характеристик є складним завданням, оскільки частина з них має кількісний характер, а частина потребує експертного аналізу. Саме тому актуальним є створення програмних засобів, які дають змогу формалізувати процес експертного оцінювання, зменшити вплив суб’єктивності, зберігати результати оцінок і надавати користувачам узагальнені висновки щодо якості програмного продукту.

Одним із практичних шляхів вирішення цієї задачі є розроблення веб-застосунку для експертного оцінювання якості програмного забезпечення. Такий застосунок може забезпечувати створення об’єктів оцінювання, формування набору критеріїв якості, залучення експертів, введення оцінок, автоматизоване обчислення підсумкових показників та формування результатів для подальшого аналізу. Використання веб-технологій є доцільним, оскільки вони забезпечують доступність системи для різних користувачів, централізоване збереження даних, можливість розмежування ролей та зручну організацію взаємодії між учасниками процесу оцінювання.

Актуальність теми кваліфікаційної роботи полягає у необхідності створення програмного засобу, який підтримує процес експертного оцінювання якості програмного забезпечення та дозволяє організувати його у вигляді зрозумілого, повторюваного й контрольованого процесу.

Метою кваліфікаційної роботи є проєктування та створення веб-застосунку експертного оцінювання якості програмного забезпечення, який забезпечує формування критеріїв оцінювання якості, введення експертних оцінок, обчислення узагальненого показника та представлення результатів користувачам.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАГАЛЬНИХ ВИМОГ ДО ВЕБ-ЗАСТОСУНКУ ЕКСПЕРТНОГО ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У першому розділі розглядається предметна область експертного оцінювання якості програмного забезпечення. Проаналізовано поняття якості ПЗ, основні характеристики програмного продукту та роль експертного підходу в процесі оцінювання. У роботі обґрунтовано вибір моделі ISO/IEC 25010:2023 як основи для формування характеристик і критеріїв оцінювання. Також визначено загальні вимоги до веб-застосунку, який повинен забезпечувати роботу з програмними продуктами, критеріями якості, експертними оцінками та підсумковими результатами.

1.1. Аналіз предметної області оцінювання якості програмного забезпечення

Програмне забезпечення є одним із ключових результатів діяльності у сфері інформаційних технологій. У межах інженерії програмного забезпечення якість ПЗ розглядається як сукупність властивостей програмного продукту, які визначають його здатність задовольняти встановлені та очікувані потреби користувачів [1]. Це означає, що якісне програмне забезпечення повинно не лише виконувати передбачені функції, а й робити це надійно, безпечно, ефективно та зрозуміло для користувача. Тому оцінювання якості ПЗ є важливою складовою процесу розроблення, впровадження та супроводу програмних систем [2].

Оцінювання якості програмного забезпечення може виконуватися на різних етапах життєвого циклу. На етапі аналізу вимог воно дозволяє визначити, які характеристики майбутньої системи є найважливішими для замовника та користувачів. На стадії проєктування визначення якості допомагає перевірити, чи відповідають архітектурні рішення очікуваним властивостям програмного продукту. Під час реалізації та тестування оцінювання якості використовується для

виявлення дефектів, перевірки коректності функцій, аналізу зручності інтерфейсу та відповідності системи нефункціональним вимогам. Після впровадження програмного продукту оцінювання може застосовуватися для визначення напрямів його подальшого вдосконалення.

До основних характеристик якості програмного забезпечення належать [1-3]:

– функціональна придатність;
– ефективність продуктивності;
– сумісність;
– здатність до взаємодії;
– надійність;
– захищеність;
– супроводжуваність;
– гнучкість;
– безпечність.

Кожна з цих характеристик може бути деталізована через окремі критерії. Наприклад, зручність використання може оцінюватися через зрозумілість інтерфейсу, простоту навігації та логічність розміщення елементів керування. Надійність може визначатися стабільністю роботи програми, коректною обробкою помилок і здатністю відновлюватися після збоїв.

У процесі оцінювання якості програмного забезпечення важливу роль відіграють експерти [3]. Експертами можуть бути розробники, тестувальники, аналітики, викладачі, користувачі або інші фахівці, які мають досвід роботи з програмними системами. Вони аналізують програмний продукт за визначеними критеріями та виставляють оцінки за обраною шкалою.

Експертне оцінювання є корисним тоді, коли певну властивість важко виміряти автоматично. Наприклад, час відповіді системи можна визначити за допомогою тестів, а зручність інтерфейсу або зрозумілість повідомлень про помилки часто потребують саме експертної думки. Водночас експертне оцінювання може бути суб’єктивним, оскільки різні фахівці можуть по-різному сприймати один і той самий програмний продукт.

Для зменшення суб’єктивності доцільно використовувати формалізований процес оцінювання. Він передбачає визначення єдиних критеріїв, однакової шкали оцінювання, вагових коефіцієнтів і правил обчислення підсумкового результату. Саме для цього в роботі пропонується створити веб-застосунок експертного оцінювання якості програмного забезпечення.

Загальну схему процесу експертного оцінювання якості програмного забезпечення подано на рис. 1.1.

Рисунок 1.1 – Процес експертного оцінювання програмного забезпечення

Як видно з рис. 1.1, процес оцінювання починається з вибору програмного продукту. Далі визначаються характеристики та критерії якості, за якими буде проводитися оцінювання. Після цього експерти вводять оцінки, а система автоматично обчислює підсумкові показники. Завершальним етапом є формування результатів, які можуть бути використані для аналізу якості програмного забезпечення та визначення напрямів його покращення.

Веб-застосунок, який розробляється в межах цієї кваліфікаційної роботи, має автоматизувати основні етапи цього процесу. Він повинен забезпечувати створення об’єктів оцінювання, додавання критеріїв якості, роботу з експертами, збереження оцінок, обчислення результатів і перегляд підсумкової інформації.

Основними користувачами такого веб-застосунку можуть бути адміністратор, експерт і користувач, який переглядає результати оцінювання. Адміністратор керує обліковими записами, критеріями та об’єктами оцінювання. Експерт виставляє оцінки за заданими критеріями. Користувач переглядає результати та аналізує підсумковий рівень якості програмного продукту.

Отже, предметна область оцінювання якості програмного забезпечення охоплює критерії якості, експертів, оцінки, вагові коефіцієнти та підсумкові результати. Розроблення веб-застосунку для підтримки цього процесу дозволить зробити оцінювання більш впорядкованим, зрозумілим і зручним для практичного використання.

1.2. Аналіз моделей якості програмного забезпечення та критеріїв оцінювання

Для оцінювання якості програмного забезпечення потрібно мати чіткий набір характеристик і критеріїв. Без цього процес оцінювання буде неточним, оскільки кожен експерт може по-різному розуміти поняття якості. Саме тому в інженерії програмного забезпечення використовують моделі якості ПЗ.

Модель якості програмного забезпечення – це структурований набір характеристик, за якими можна оцінити програмний продукт [3-5]. Вона допомагає визначити, що саме потрібно перевіряти: правильність роботи функцій, швидкодію, зручність інтерфейсу, безпеку, надійність, можливість супроводу та інші властивості.

У різні періоди розвитку інженерії програмного забезпечення було запропоновано декілька моделей якості. До найбільш відомих належать модель МакКола, модель Боема, FURPS та ISO/IEC 25010. Кожна з них має власний набір

характеристик, але всі вони спрямовані на те, щоб зробити оцінювання якості ПЗ більш зрозумілим і впорядкованим [4].

Модель МакКола є однією з перших моделей якості програмного забезпечення. Вона розглядає якість через такі властивості, як коректність, надійність, ефективність, зручність використання, супроводжуваність, гнучкість, тестованість і переносимість. Перевагою цієї моделі є спроба систематизувати характеристики якості ПЗ. Проте для сучасних програмних продуктів вона є досить загальною.

Модель Боема також орієнтована на оцінювання якості програмного забезпечення. Вона враховує зручність використання, надійність, ефективність, зрозумілість, модифікованість і переносимість. Ця модель корисна для аналізу програмного продукту з погляду розробника, але її складно безпосередньо застосувати в реальних умовах розробки [1, 4].

Модель FURPS описує якість програмного забезпечення через п’ять основних груп характеристик: функціональність, зручність використання, надійність, продуктивність і супроводжуваність. Ця модель є досить зрозумілою і може бути використана для попереднього формування критеріїв оцінювання.

Найбільш доцільною для цієї роботи є модель ISO/IEC 25010, оскільки вона є сучасною та широко використовується для опису якості програмних продуктів. Вона містить набір характеристик, які добре підходять для оцінювання веб-застосунків та інших програмних систем.

Порівняння основних моделей якості програмного забезпечення подано в табл. 1.1.

Таблиця 1.1 – Порівняння моделей якості програмного забезпечення

Модель якості	**Основна ідея**	**Можливість використання**
1	**2**	**3**
Модель МакКола	Оцінювання якості через набір факторів, критеріїв і метрик	Може бути використана для загального аналізу підходів до якості ПЗ

Продовження таблиці 1.1

1	2	3
Модель Боема	Оцінювання якості з погляду зручності, надійності, ефективності та супроводу	Корисна для розуміння зв’язку між якістю та супроводом ПЗ
FURPS	Поділ якості на функціональність, зручність, надійність, продуктивність і супровід	Може бути використана для простого групування критеріїв
ISO/IEC 25010:2023	Сучасна модель якості продукту для ICT-продуктів і програмного забезпечення	Доцільна як основа для критеріїв оцінювання у веб-застосунку

У межах кваліфікаційної роботи доцільно використати саме ISO/IEC 25010:2023, оскільки ця модель дозволяє оцінювати програмне забезпечення за кількома важливими характеристиками. Основні характеристики моделі якості продукту згідно ISO/IEC 25010:2023 подано на рис. 1.2.

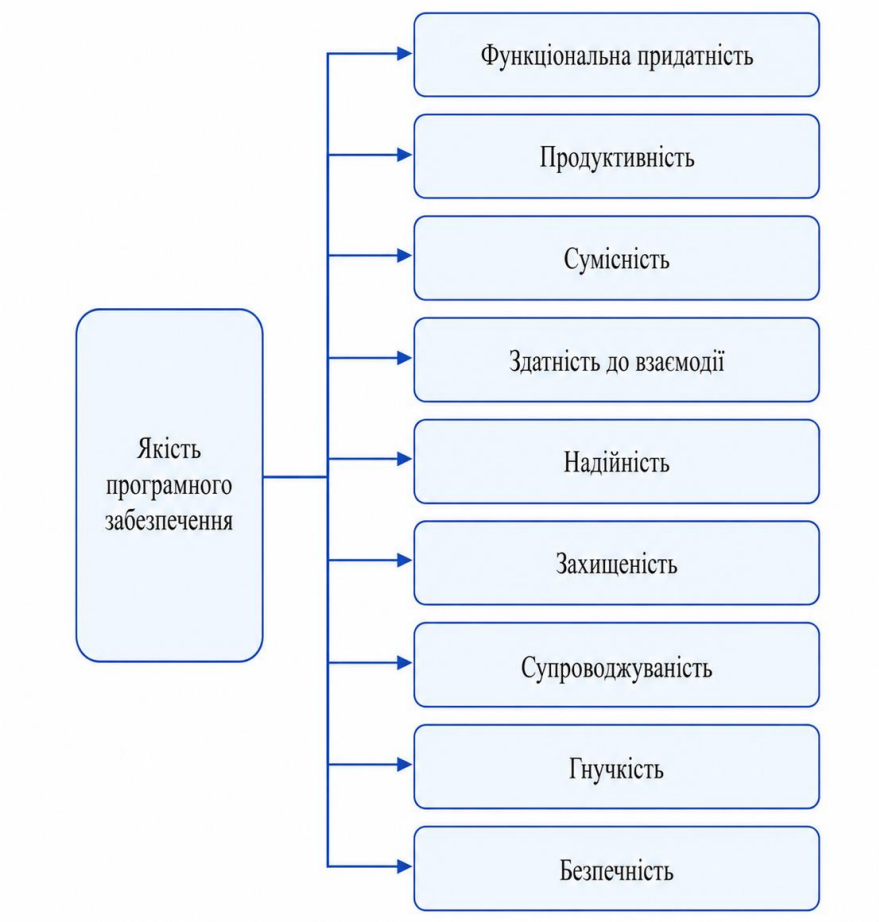

Рисунок 1.2 – Якість програмного забезпечення в термінах моделі ISO/IEC 25010:2023

Як видно з рис. 1.2, модель якості продукту ISO/IEC 25010:2023 містить дев’ять характеристик. Функціональна придатність показує, наскільки програмний продукт виконує потрібні користувачу функції. Для веб-застосунку експертного оцінювання це означає можливість створювати об’єкти оцінювання, додавати критерії, вводити оцінки та переглядати результати.

Продуктивність характеризує швидкість роботи програмної системи та раціональність використання ресурсів. Для веб-застосунку це може проявлятися у швидкому завантаженні сторінок, оперативному збереженні оцінок і швидкому формуванні підсумкових результатів [2].

Сумісність визначає здатність програмного продукту працювати разом з іншими системами, сервісами або компонентами без конфліктів. Для веб-застосунку це може стосуватися сумісності з різними браузерами, базою даних та зовнішніми сервісами [2].

Здатність до взаємодії описує, наскільки зручно й ефективно користувач може взаємодіяти із програмним продуктом. Для розроблюваної системи важливо, щоб експерт міг швидко знайти потрібний об’єкт оцінювання, зрозуміти критерії та внести оцінки без зайвих дій [2].

Надійність означає здатність програмного забезпечення стабільно працювати в заданих умовах. У межах цієї роботи надійність може оцінюватися через коректне збереження введених даних, обробку помилок і відсутність втрати результатів оцінювання.

Захищеність показує, наскільки система захищає дані користувачів і результати оцінювання [3]. Для веб-застосунку це означає необхідність авторизації користувачів, розмежування прав доступу та захисту від несанкціонованих дій.

Супроводжуваність характеризує зручність внесення змін у програмну систему. Якщо веб-застосунок має зрозумілу структуру коду, поділ на модулі та чітку логіку роботи, його простіше розвивати й підтримувати.

Гнучкість визначає здатність програмного продукту адаптуватися до різних або змінних умов використання [3]. Для розроблюваного веб-застосунку це означає

можливість змінювати критерії оцінювання, додавати нові характеристики якості та адаптувати систему до різних типів програмних продуктів.

Безпечність показує, наскільки програмний продукт не створює ризиків для користувачів, даних, бізнес-процесів або середовища використання [3]. У межах цієї роботи ця характеристика може розглядатися через коректність роботи системи, недопущення втрати важливих результатів оцінювання та зменшення ризику помилкових управлінських рішень.

Для розроблюваного веб-застосунку доцільно сформувати критерії оцінювання на основі наведених характеристик. Приклад такого набору критеріїв подано в табл. 1.2.

Таблиця 1.2 – Приклад критеріїв оцінювання якості ПЗ

Характеристика якості	**Приклади критеріїв оцінювання**
Функціональна придатність	Повнота функцій, правильність виконання операцій, відповідність вимогам
Продуктивність	Швидкість завантаження сторінок, час обробки запиту, швидкість формування результатів
Сумісність	Коректна робота в різних браузерах, взаємодія з базою даних, можливість інтеграції з іншими сервісами
Здатність до взаємодії	Зрозумілість інтерфейсу, простота навігації, зручність введення оцінок
Надійність	Стабільність роботи, обробка помилок, збереження даних
Захищеність	Авторизація користувачів, розмежування ролей, захист результатів оцінювання
Супроводжуваність	Модульність коду, зрозумілість структури, можливість додавання нових функцій
Гнучкість	Можливість змінювати критерії, додавати нові шкали оцінювання, адаптувати систему до різних типів ПЗ
Безпечність	Запобігання втраті важливих даних, коректність підсумкових розрахунків, зменшення ризику помилкових висновків

У веб-застосунку експертного оцінювання кожна характеристика може мати декілька критеріїв. Для кожного критерію експерт виставляє оцінку, наприклад за п’ятибальною шкалою. Після цього система може обчислити середнє значення або зважений підсумковий показник якості.

Такий підхід є зручним, оскільки дозволяє не просто виставити загальну оцінку програмному продукту, а побачити, які саме характеристики є сильними, а які потребують покращення. Отже, моделі якості програмного забезпечення є основою для побудови критеріїв експертного оцінювання. У цій роботі доцільно орієнтуватися на характеристики ISO/IEC 25010:2023, оскільки вони охоплюють основні властивості сучасних програмних продуктів.

1.3. Загальні вимоги до веб-застосунку експертного оцінювання якості програмного забезпечення

Після аналізу предметної області та моделей якості програмного забезпечення можна визначити загальні вимоги до веб-застосунку, який розробляється у кваліфікаційній роботі. Такий застосунок призначений для організації процесу експертного оцінювання якості програмних продуктів за вибраними характеристиками та критеріями.

Основне призначення веб-застосунку полягає в тому, щоб надати користувачам зручний інструмент для створення об’єктів оцінювання, формування критеріїв якості, введення експертних оцінок, обчислення підсумкових результатів та перегляду отриманих висновків. На відміну від звичайних таблиць або текстових форм, веб-застосунок повинен забезпечувати централізоване збереження даних, розмежування ролей користувачів і автоматичне опрацювання результатів.

Об’єктом оцінювання у системі є програмний продукт. Це може бути веб-застосунок, мобільний застосунок, інформаційна система, програмний модуль або інший вид програмного забезпечення. Для кожного об’єкта оцінювання потрібно зберігати його назву, короткий опис, дату створення запису, перелік критеріїв та результати оцінювання.

У системі доцільно передбачити три основні ролі: адміністратора, експерта та користувача результатів. Адміністратор керує об’єктами оцінювання, критеріями та обліковими записами. Експерт вводить оцінки за визначеними

критеріями. Користувач результатів переглядає підсумкові показники та аналізує якість програмного продукту.

Система повинна підтримувати роботу з характеристиками та критеріями якості. За основу доцільно взяти характеристики моделі ISO/IEC 25010:2023: функціональну придатність, ефективність продуктивності, сумісність, здатність до взаємодії, надійність, захищеність, супроводжуваність, гнучкість і безпечність. Для кожної характеристики можуть визначатися окремі критерії, за якими експерти будуть виконувати оцінювання.

Для зручності роботи експертів веб-застосунок повинен мати простий і зрозумілий інтерфейс. Експерт має швидко знаходити потрібний програмний продукт, бачити перелік критеріїв і вводити оцінки без зайвих дій. Доцільно використовувати однакову шкалу оцінювання, наприклад від 1 до 5, де мінімальне значення означає низький рівень відповідності критерію, а максимальне — високий рівень відповідності.

Окремою вимогою є автоматичне обчислення підсумкових результатів. Після введення оцінок система повинна визначати середні або зважені значення за окремими критеріями, характеристиками та програмним продуктом загалом. Це дозволить уникнути ручних розрахунків і зменшити ризик помилок під час підбиття підсумків.

Результати оцінювання мають подаватися у зрозумілому вигляді. Користувач повинен бачити загальний рівень якості програмного продукту, оцінки за окремими характеристиками та критерії, які отримали найнижчі значення. Така інформація потрібна для визначення сильних і слабких сторін програмного продукту та формування рекомендацій щодо його покращення.

Веб-застосунок також повинен забезпечувати збереження історії оцінювання. Це дасть змогу порівнювати результати для різних програмних продуктів або аналізувати зміни якості одного продукту після його доопрацювання. Збереження історії є важливим для контролю якості та прийняття рішень щодо подальшого розвитку програмного забезпечення.

До системи висуваються загальні вимоги щодо безпеки. Оскільки веб-застосунок працює з користувачами, оцінками та результатами аналізу, необхідно передбачити авторизацію, розмежування доступу та захист від несанкціонованої зміни даних. Кожен користувач повинен мати доступ лише до тих функцій, які відповідають його ролі.

Важливою є також вимога до надійності роботи системи. Дані, введені експертами, не повинні втрачатися під час збереження або обчислення результатів. Система має коректно обробляти помилкові дії користувача, наприклад незаповнені поля, некоректні значення або спробу виконати дію без відповідних прав доступу.

Застосунок повинен бути придатним до подальшого розвитку. У майбутньому до нього можна буде додати нові шкали оцінювання, розширені звіти, експорт результатів, графічне представлення даних або порівняння декількох програмних продуктів. Тому під час розроблення необхідно передбачити логічну структуру модулів і зрозумілу організацію програмного коду.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ЕКСПЕРТНОГО ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У другому розділі виконується проєктування веб-застосунку експертного оцінювання якості програмного забезпечення. На основі результатів аналізу предметної області визначаються функціональні та нефункціональні вимоги до системи, розробляються діаграми варіантів використання, діаграми потоків даних, архітектура програмної системи та структура бази даних.

Основна увага в розділі приділяється інженерним аспектам створення програмного забезпечення. Це включає опис ролей користувачів, визначення основних функцій веб-застосунку, проєктування логіки обробки експертних оцінок, організацію збереження даних і підготовку основи для подальшої програмної реалізації.

2.1. Формування функціональних вимог до веб-застосунку

Функціональні вимоги визначають, які дії повинна виконувати програмна система та які можливості вона має надавати користувачам [5]. Для веб-застосунку експертного оцінювання якості програмного забезпечення такі вимоги формуються з урахуванням основного призначення системи: організувати процес оцінювання програмних продуктів за визначеними характеристиками та критеріями.

У розроблюваному веб-застосунку доцільно передбачити три основні ролі користувачів: адміністратора, експерта та користувача результатів. Кожна роль має власний набір функцій і рівень доступу до даних системи.

Адміністратор відповідає за налаштування системи та керування основними даними. Він повинен мати можливість створювати та редагувати облікові записи користувачів, додавати об’єкти оцінювання, формувати характеристики якості, задавати критерії оцінювання та переглядати результати роботи експертів.

Експерт є користувачем, який безпосередньо виконує оцінювання програмного продукту. Він повинен мати доступ до призначених йому об’єктів

оцінювання, переглядати критерії якості, вводити оцінки за встановленою шкалою та, за потреби, додавати короткий коментар до оцінки.

Користувач результатів переглядає підсумкову інформацію про якість програмного продукту. Він не змінює критерії та не вводить оцінки, але може переглядати загальний результат, оцінки за окремими характеристиками, слабкі місця програмного продукту та сформовані висновки.

До основних функціональних вимог веб-застосунку належать:

– реєстрація та авторизація користувачів;
– розмежування прав доступу відповідно до ролей користувачів;
– створення, редагування та видалення об’єктів оцінювання;
– створення характеристик якості програмного забезпечення;
– створення та редагування критеріїв оцінювання;
– призначення експертів до об’єктів оцінювання;
– введення експертних оцінок за визначеною шкалою;
– збереження оцінок і коментарів експертів;
– автоматичне обчислення підсумкових показників якості;
– перегляд результатів оцінювання;
– формування узагальненого висновку щодо якості програмного продукту;
– збереження історії оцінювання;
– пошук і фільтрація об’єктів оцінювання.

Загальну характеристику функціональних вимог до веб-застосунку подано в табл. 2.1.

Таблиця 2.1 – Функціональні вимоги до веб-застосунку

Код вимоги	**Назва вимоги**	**Короткий опис**
1	**2**	**3**
FR-01	Авторизація користувачів	Система повинна забезпечувати вхід користувачів за логіном і паролем
FR-02	Розмежування ролей	Система повинна надавати різні права адміністратору, експерту та користувачу результатів

Продовження таблиці 2.1

1	2	3
FR-03	Керування об’єктами оцінювання	Адміністратор повинен мати можливість створювати, редагувати та видаляти програмні продукти для оцінювання
FR-04	Керування характеристиками якості	Система повинна дозволяти створювати характеристики якості відповідно до вибраної моделі оцінювання
FR-05	Керування критеріями	Адміністратор повинен мати можливість додавати критерії до відповідних характеристик якості
FR-06	Призначення експертів	Система повинна дозволяти призначати експертів до конкретних об’єктів оцінювання
FR-07	Введення оцінок	Експерт повинен мати можливість виставляти оцінки за визначеними критеріями
FR-08	Збереження коментарів	Експерт повинен мати можливість додавати короткі пояснення до виставлених оцінок
FR-09	Розрахунок результатів	Система повинна автоматично обчислювати підсумкові показники якості
FR-10	Перегляд результатів	Користувач повинен мати можливість переглядати результати оцінювання у зручному вигляді
FR-11	Історія оцінювання	Система повинна зберігати попередні результати оцінювання програмних продуктів
FR-12	Пошук і фільтрація	Система повинна підтримувати пошук програмних продуктів та фільтрацію результатів

Функціональні вимоги формують основу для подальшого проєктування веб-застосунку. На їх основі можна визначити основні сценарії роботи користувачів, побудувати діаграму варіантів використання, спроєктувати структуру бази даних і визначити склад програмних модулів [6-8].

Отже, функціональні вимоги до веб-застосунку визначають набір основних можливостей, необхідних для організації експертного оцінювання якості програмного забезпечення. Детальніше взаємодію користувачів із системою буде подано за допомогою діаграми варіантів використання.

2.2. Формування нефункціональних вимог до веб-застосунку

Нефункціональні вимоги визначають якісні умови роботи програмної системи. Якщо функціональні вимоги описують, які дії повинен виконувати веб-

застосунок, то нефункціональні вимоги визначають, наскільки зручно, стабільно, безпечно та ефективно ці дії мають виконуватися.

Для веб-застосунку експертного оцінювання якості програмного забезпечення нефункціональні вимоги мають важливе значення, оскільки система працює з обліковими записами користувачів, критеріями оцінювання, експертними оцінками та підсумковими результатами. Тому під час її проєктування необхідно врахувати вимоги до інтерфейсу, захисту даних, швидкодії, надійності збереження інформації та можливості подальшого розвитку.

Передусім веб-застосунок повинен мати зрозумілий інтерфейс. Адміністратор має швидко переходити до керування користувачами, програмними продуктами та критеріями оцінювання. Експерт повинен без зайвих дій знаходити призначені йому об’єкти та вводити оцінки. Користувач результатів повинен мати можливість легко переглядати підсумкові показники та оцінки за окремими характеристиками.

Важливою вимогою є коректна організація доступу до системи. Кожен користувач повинен входити до веб-застосунку за допомогою облікового запису. Після авторизації система має визначати роль користувача та надавати доступ лише до тих функцій, які відповідають цій ролі. Це дозволить уникнути несанкціонованої зміни критеріїв, оцінок або результатів.

Система повинна забезпечувати надійне збереження даних. Інформація про користувачів, програмні продукти, критерії, оцінки та результати повинна зберігатися в базі даних і не втрачатися під час роботи веб-застосунку. У разі помилкового введення даних система повинна повідомляти користувача про проблему та не допускати збереження некоректної інформації.

До веб-застосунку також висуваються вимоги щодо швидкодії. Основні сторінки системи повинні відкриватися без помітних затримок, а операції збереження оцінок і формування підсумкових результатів мають виконуватися достатньо швидко для комфортної роботи користувачів. У межах бакалаврської роботи система орієнтована на роботу з невеликою або середньою кількістю користувачів і об’єктів оцінювання.

Окрему увагу потрібно приділити коректності обчислення результатів. Після введення експертних оцінок система повинна правильно виконувати розрахунок підсумкових показників і відображати результат у зрозумілому вигляді. Користувач повинен бачити, з яких оцінок сформовано загальний результат, щоб мати можливість перевірити логіку оцінювання.

Веб-застосунок має бути придатним до подальшого супроводу. Його програмна структура повинна бути логічною та поділеною на окремі модулі. Це спростить внесення змін, виправлення помилок і додавання нових можливостей, наприклад експорту результатів, побудови графіків або використання інших шкал оцінювання.

Також необхідно врахувати сумісність веб-застосунку з сучасними браузерами. Система повинна коректно відображатися на персональному комп’ютері або ноутбуці та забезпечувати стабільну роботу основних сторінок. Для користувача це означає можливість працювати із системою без встановлення додаткового програмного забезпечення. Узагальнення нефункціональних вимог до веб-застосунку подано в табл. 2.2.

Таблиця 2.2 – Нефункціональні вимоги до веб-застосунку

Код вимоги	**Назва вимоги**	**Опис вимоги**
1	**2**	**3**
NFR-01	Зручність інтерфейсу	Інтерфейс системи повинен бути зрозумілим для адміністратора, експерта та користувача результатів
NFR-02	Авторизований доступ	Користувачі повинні входити до системи за допомогою облікового запису
NFR-03	Розмежування прав	Доступ до функцій системи повинен залежати від ролі користувача
NFR-04	Збереження даних	Дані про користувачів, критерії, оцінки та результати повинні зберігатися в базі даних
NFR-05	Перевірка введених даних	Система повинна повідомляти користувача про незаповнені поля або некоректні значення
NFR-06	Швидкодія	Основні сторінки системи повинні відкриватися без помітних затримок
NFR-07	Коректність розрахунків	Підсумкові показники якості повинні обчислюватися відповідно до заданої логіки оцінювання
NFR-08	Зрозуміле подання результатів	Результати оцінювання повинні відображатися у вигляді, зручному для аналізу

Продовження таблиці 2.2

1	2	3
NFR-09	Супроводжуваність	Програмний код повинен мати логічну структуру та поділ на окремі модулі
NFR-10	Можливість розвитку	Система повинна дозволяти подальше додавання нових функцій і способів подання результатів
NFR-11	Сумісність із браузерами	Веб-застосунок повинен коректно працювати в сучасних веб-браузерах
NFR-12	Стабільність роботи	Система повинна коректно обробляти помилкові дії користувача без втрати даних

Наведені нефункціональні вимоги доповнюють функціональні вимоги та визначають умови якісної роботи веб-застосунку. Їх урахування є необхідним для подальшого проєктування архітектури та відповідних компонентів веб-застосунку.

Отже, нефункціональні вимоги до веб-застосунку експертного оцінювання якості програмного забезпечення спрямовані на забезпечення зручної, стабільної, захищеної та зрозумілої роботи системи. Вони деталізують не те, які саме функції реалізує система, а те, якими властивостями повинна володіти її програмна реалізація.

2.3. Розробка діаграм варіантів використання веб-застосунку

Після визначення функціональних і нефункціональних вимог доцільно перейти до моделювання взаємодії користувачів із веб-застосунком. Для цього використано діаграми варіантів використання UML, які дозволяють показати, які дії може виконувати кожна роль у системі [9].

Діаграма варіантів використання відображає зовнішню поведінку програмної системи з погляду користувача. Вона не описує внутрішню реалізацію функцій, структуру бази даних або програмний код, а показує, які можливості система надає своїм користувачам. Такий підхід є зручним на етапі проєктування, оскільки дозволяє уточнити межі системи та визначити основні сценарії її використання.

Для веб-застосунку експертного оцінювання якості програмного забезпечення виділено три основні ролі користувачів: адміністратора, експерта та

користувача результатів. Кожна роль має власний набір функцій і різний рівень доступу до даних системи.

Адміністратор є користувачем із найширшими правами доступу. Він відповідає за підготовку системи до проведення оцінювання, керує обліковими записами користувачів, створює об’єкти оцінювання, формує характеристики та критерії якості, призначає експертів і переглядає результати. Діаграму варіантів використання для ролі адміністратора подано на рис. 2.1.

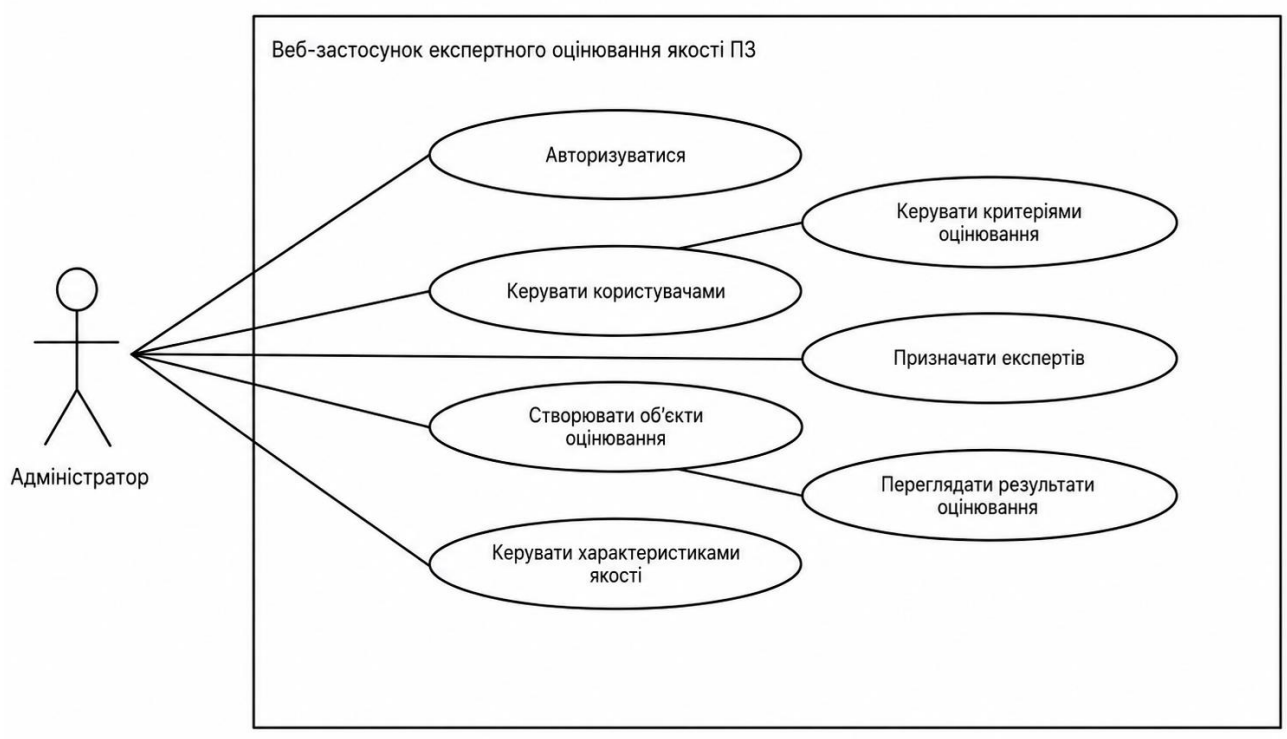

Рисунок 2.1 – Діаграма варіантів використання для ролі адміністратора

На діаграмі (рис. 2.1) для адміністратора передбачено такі основні варіанти використання, які забезпечують налаштування системи та організацію процесу експертного оцінювання.

Експерт є користувачем, який безпосередньо бере участь у процесі оцінювання програмного продукту. Він не керує системними налаштуваннями, а працює лише з тими об’єктами, які були йому призначені адміністратором. Діаграму варіантів використання для ролі експерта подано на рис. 2.2.

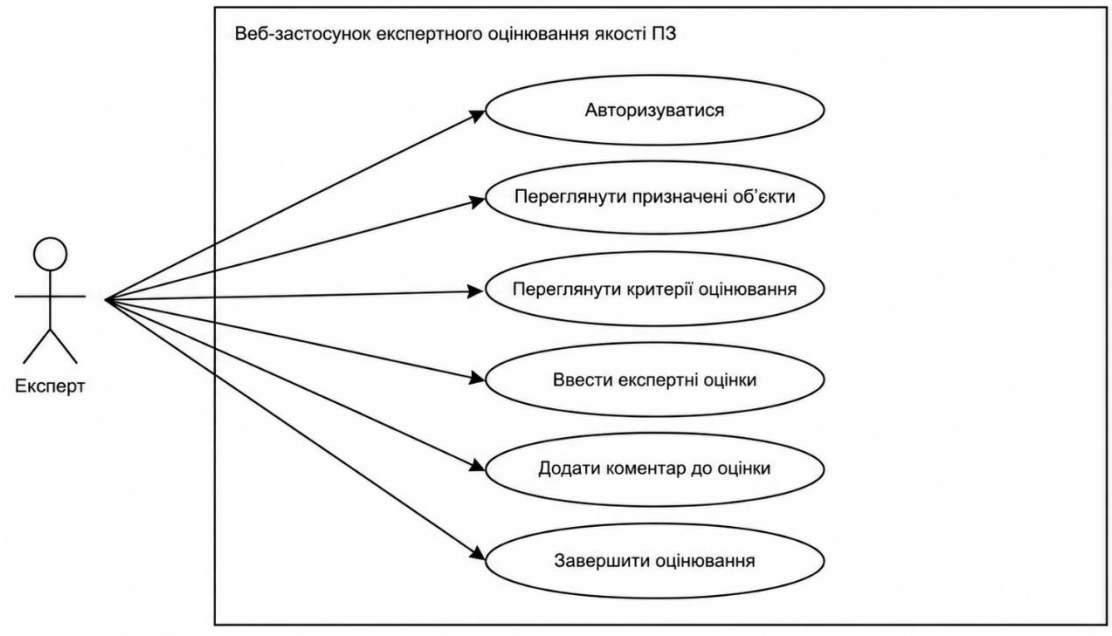

Рисунок 2.2 – Діаграма варіантів використання для ролі експерта

Основним сценарієм роботи експерта є вибір призначеного програмного продукту, ознайомлення з критеріями та введення оцінок за встановленою шкалою.

Користувач результатів працює з підсумковою інформацією. Він не змінює об'єкти оцінювання, критерії або оцінки, а лише переглядає сформовані системою результати. Така роль може бути корисною для керівника проєкту, замовника або іншої особи, яка аналізує якість програмного продукту. Діаграму варіантів використання для ролі користувача результатів подано на рис. 2.3.

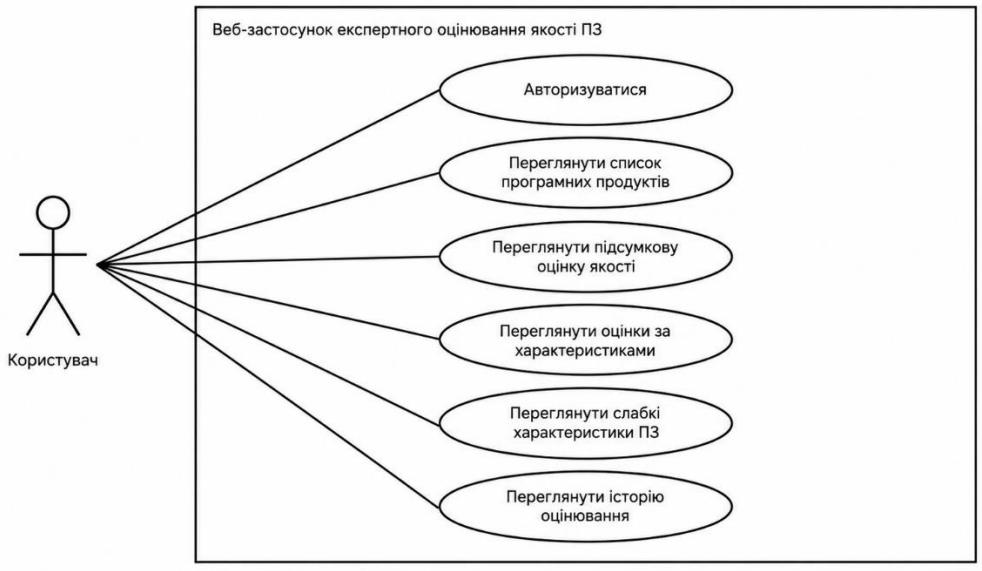

Рисунок 2.3 – Діаграма варіантів використання для ролі користувач

Основна мета ролі користувача полягає в отриманні зрозумілої інформації про рівень якості програмного забезпечення.

Розроблені діаграми варіантів використання дозволяють уточнити функціональні межі веб-застосунку. Вони показують, які дії доступні кожній ролі, і допомагають уникнути змішування прав доступу між різними користувачами. Наприклад, експерт може вводити оцінки, але не повинен змінювати критерії оцінювання, а користувач результатів може переглядати підсумки, але не має права редагувати дані.

Таким чином, use case діаграми є основою для подальшого проєктування веб-застосунку. На їх основі можна визначити сторінки веб-застосунку, маршрути переходів між ними, права доступу до функцій і логіку обробки дій користувачів.

2.4. Розробка діаграм потоків даних веб-застосунку

Після визначення функціональних вимог і побудови діаграм варіантів використання доцільно розглянути, як саме дані переміщуються всередині веб-застосунку експертного оцінювання якості програмного забезпечення. Для цього використовують діаграми потоків даних, які дозволяють показати основні інформаційні потоки між користувачами, процесами системи та сховищами даних.

Діаграма потоків даних відображає не послідовність виконання команд, а рух інформації в системі [10-12]. Вона показує, які дані надходять до веб-застосунку, як вони опрацьовуються, де зберігаються та які результати повертаються користувачам. Такий підхід є зручним для проєктування програмної системи, оскільки допомагає краще зрозуміти взаємозв’язок між основними модулями.

Для розроблюваного веб-застосунку основними зовнішніми сутностями є адміністратор, експерт і користувач результатів.

Основними сховищами даних у системі є бази даних користувачів, об’єктів оцінювання, критеріїв якості, експертних оцінок і результатів оцінювання. Їх можна реалізувати як окремі таблиці реляційної бази даних. Поділ даних на такі

групи дозволяє впорядкувати збереження інформації та спростити подальше проєктування структури бази даних.

На контекстному рівні веб-застосунок можна подати як єдиний процес, який взаємодіє із зовнішніми користувачами. Адміністратор передає системі дані для налаштування оцінювання, експерт вводить оцінки, а користувач результатів отримує сформовані підсумки. Контекстну діаграму потоків даних веб-застосунку показано на рис. 2.4.

Рисунок 2.4 – Контекстна діаграма потоків даних веб-застосунку експертного оцінювання якості програмного забезпечення

На контекстній діаграмі центральним процесом є веб-застосунок експертного оцінювання якості програмного забезпечення. До нього надходять дані від адміністратора та експерта, а результатом роботи є підсумкова інформація про якість програмного продукту. Така діаграма показує загальні межі системи та основні напрями обміну даними.

Для детальнішого опису роботи системи доцільно побудувати DFD-діаграму першого рівня. На цьому рівні веб-застосунок поділяється на декілька основних процесів: керування користувачами, керування об’єктами оцінювання, керування критеріями якості, введення експертних оцінок, розрахунок результатів і перегляд підсумкової інформації.

Процес керування користувачами забезпечує створення облікових записів, збереження даних користувачів і визначення їхніх ролей. Процес керування об’єктами оцінювання відповідає за створення та редагування програмних продуктів, які будуть оцінюватися. До таких даних належать назва програмного продукту, короткий опис, дата створення запису та інша службова інформація.

Процес керування критеріями якості забезпечує формування характеристик і критеріїв, за якими експерти оцінюватимуть програмні продукти.

Процес введення експертних оцінок забезпечує отримання оцінок від експертів за визначеними критеріями. Процес розрахунку результатів використовує збережені експертні оцінки, критерії та, за потреби, вагові коефіцієнти. На основі цих даних система обчислює показники якості за окремими характеристиками та загальну оцінку програмного продукту.

Процес перегляду результатів забезпечує подання підсумкової інформації користувачу. Користувач результатів може переглянути загальний показник якості, оцінки за характеристиками, слабкі сторони програмного продукту та історію попередніх оцінювань.

DFD-діаграму [11-13] першого рівня для веб-застосунку експертного оцінювання якості програмного забезпечення подано на рис. 2.5.

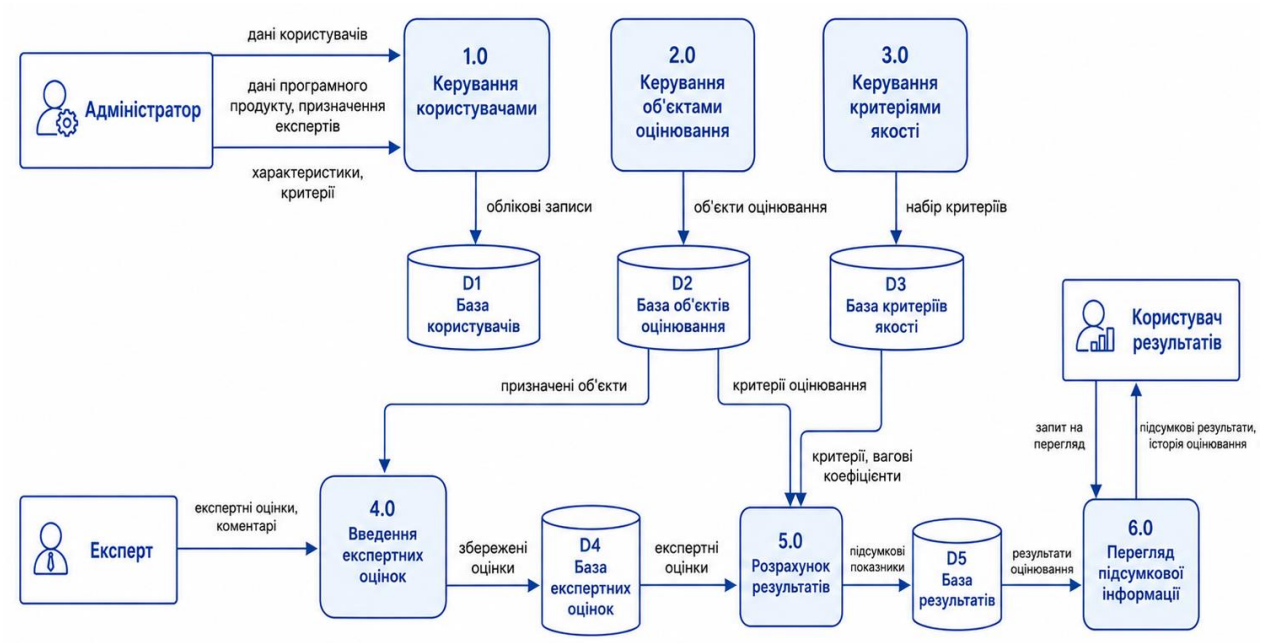

Рисунок 2.5 – DFD-діаграма

Для кращого розуміння інформаційних потоків у системі доцільно узагальнити їх у таблиці. Основні потоки даних веб-застосунку подано в табл. 2.3.

Таблиця 2.3 – Основні потоки даних веб-застосунку

Джерело даних	**Дані**	**Процес-одержувач**	**Результат опрацювання**
Адміністратор	Дані користувачів	Керування користувачами	Створення або оновлення облікових записів
	Дані програмного продукту	Керування об’єктами оцінювання	Створення об’єкта оцінювання
	Характеристики та критерії якості	Керування критеріями якості	Формування набору критеріїв оцінювання
	Дані про призначення експертів	Керування об’єктами оцінювання	Закріплення експертів за програмним продуктом
Експерт	Експертні оцінки	Введення експертних оцінок	Збереження оцінок у базі даних
	Коментарі до оцінок	Введення експертних оцінок	Збереження пояснень до виставлених оцінок
База критеріїв	Перелік критеріїв	Введення експертних оцінок	Відображення критеріїв для експерта
База експертних оцінок	Збережені оцінки	Розрахунок результатів	Обчислення підсумкових показників
Користувач	Запит на перегляд результатів	Перегляд підсумкової інформації	Відображення результатів оцінювання
База результатів	Підсумкові показники	Перегляд підсумкової інформації	Формування звіту або сторінки результатів

Побудова діаграм потоків даних дозволяє визначити, які дані необхідно зберігати в системі та які процеси мають їх опрацьовувати [13]. Крім того, DFD-діаграми допомагають уточнити межі відповідальності програмних модулів. Наприклад, модуль керування критеріями не повинен безпосередньо виконувати розрахунок результатів, а модуль введення оцінок не повинен змінювати облікові записи користувачів. Такий поділ сприяє кращій структурованості програмної системи та полегшує її подальший супровід.

Отже, діаграми потоків даних дають змогу описати інформаційну логіку веб-застосунку експертного оцінювання якості програмного забезпечення.

2.5. Проєктування архітектури веб-застосунку

Для веб-застосунку експертного оцінювання якості програмного забезпечення доцільно використати клієнт-серверну архітектуру. Такий підхід є поширеним для веб-систем, оскільки дозволяє відокремити інтерфейс користувача від логіки обробки даних і збереження інформації [14]. Користувач працює із системою через веб-браузер, а основна обробка запитів виконується на серверній частині.

У межах розроблюваного веб-застосунку можна виділити три основні рівні архітектури: клієнтський рівень, серверний рівень та рівень бази даних.

Клієнтський рівень відповідає за взаємодію користувача із системою. До нього належать сторінки веб-застосунку, форми введення даних, елементи навігації, таблиці результатів і повідомлення про помилки. Через клієнтський рівень адміністратор керує об’єктами оцінювання та критеріями, експерт вводить оцінки, а користувач результатів переглядає підсумкову інформацію.

Серверний рівень виконує основну логіку роботи веб-застосунку. Він приймає запити від клієнтської частини, перевіряє права доступу користувача, опрацьовує введені дані, виконує розрахунок підсумкових показників і передає результати назад до інтерфейсу. Саме на цьому рівні реалізуються правила роботи системи та алгоритм експертного оцінювання.

Рівень бази даних призначений для збереження інформації, яка використовується у веб-застосунку. У базі даних зберігаються облікові записи користувачів, ролі, програмні продукти, характеристики якості, критерії оцінювання, експертні оцінки, коментарі та підсумкові результати. Такий підхід забезпечує впорядковане зберігання даних і можливість їх подальшого використання для перегляду історії оцінювання. Загальну архітектуру веб-застосунку подано на рис. 2.6.

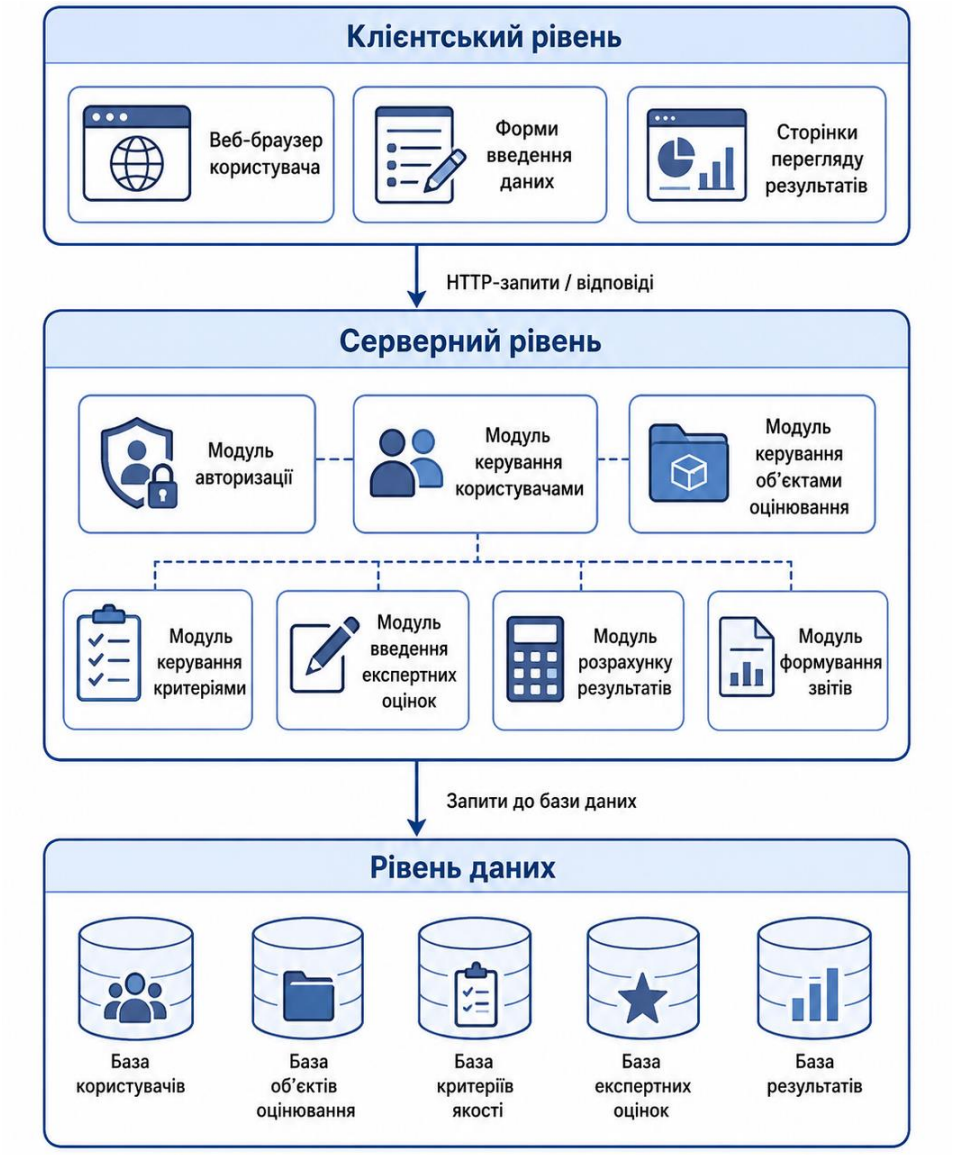

Рисунок 2.6 – Архітектура веб-застосунку експертного оцінювання якості програмного забезпечення

Клієнтська частина повинна бути зрозумілою для користувачів різних ролей. Після входу до системи користувач бачить лише ті функції, які відповідають його ролі. Адміністратор отримує доступ до сторінок керування системою, експерт – до сторінок оцінювання, а користувач результатів – до перегляду підсумкової інформації.

Серверна частина повинна забезпечувати перевірку запитів і контроль доступу. Наприклад, якщо експерт намагається відкрити об’єкт оцінювання, який йому не призначено, система повинна заборонити таку дію. Так само користувач

результатів не повинен мати можливості редагувати критерії або змінювати експертні оцінки.

Окремим складником серверного рівня є модуль розрахунку результатів. Він отримує з бази даних оцінки експертів, критерії та вагові коефіцієнти, після чого обчислює підсумкові показники якості. Результати розрахунку зберігаються в базі даних і можуть бути відображені користувачу у вигляді таблиці або короткого звіту.

Рівень бази даних повинен забезпечувати цілісність і узгодженість інформації. Для цього потрібно правильно визначити зв’язки між користувачами, програмними продуктами, критеріями та оцінками. Наприклад, кожна експертна оцінка має бути пов’язана з конкретним експертом, конкретним програмним продуктом і конкретним критерієм.

Основні програмні модулі веб-застосунку подано в табл. 2.4.

Таблиця 2.4 – Основні програмні модулі веб-застосунку

Модуль	**Призначення**
Модуль авторизації	Забезпечує вхід користувачів до системи та перевірку їхніх ролей
Модуль керування користувачами	Дозволяє адміністратору створювати та редагувати облікові записи
Модуль керування об’єктами оцінювання	Забезпечує створення та редагування програмних продуктів, які оцінюються
Модуль керування критеріями	Дозволяє створювати характеристики якості та критерії оцінювання
Модуль експертного оцінювання	Забезпечує введення оцінок і коментарів експертами
Модуль розрахунку результатів	Обчислює підсумкові показники якості програмного продукту
Модуль перегляду результатів	Відображає загальний результат, оцінки за характеристиками та історію оцінювання
Модуль роботи з базою даних	Забезпечує збереження, оновлення та отримання даних

Запропонована архітектура дозволяє відокремити інтерфейс користувача, логіку роботи системи та збереження даних. Це відповідає загальним принципам проєктування веб-застосунків і забезпечує основу для подальшої реалізації програмної системи.

Отже, для веб-застосунку експертного оцінювання якості програмного забезпечення обрано клієнт-серверну архітектуру з поділом на клієнтський рівень, серверний рівень і рівень бази даних. Такий підхід є зрозумілим, придатним для бакалаврського проєкту та достатнім для реалізації основних функцій системи.

2.6. Проєктування структури бази даних

Для веб-застосунку експертного оцінювання якості програмного забезпечення доцільно використати реляційну базу даних. Такий підхід є зручним, оскільки дані в системі мають чітку структуру та можуть бути подані у вигляді взаємопов’язаних таблиць. Наприклад, кожна експертна оцінка повинна бути пов’язана з конкретним експертом, конкретним програмним продуктом і конкретним критерієм оцінювання.

Під час проєктування бази даних необхідно врахувати основні сутності предметної області. До них належать користувачі, ролі, об’єкти оцінювання, характеристики якості, критерії, призначення експертів, експертні оцінки та результати оцінювання.

Сутність «Користувач» призначена для збереження інформації про осіб, які працюють із веб-застосунком. До таких даних належать ім’я користувача, електронна адреса, пароль, роль і статус облікового запису. Роль користувача визначає, які функції системи йому доступні.

Сутність «Роль» використовується для розмежування прав доступу. У системі передбачено три основні ролі: адміністратор, експерт і користувач результатів. Такий поділ дозволяє обмежити доступ до функцій відповідно до призначення кожного користувача.

Сутність «Об’єкт оцінювання» описує програмний продукт, якість якого потрібно оцінити. Для такого об’єкта зберігається назва, опис, дата створення запису та поточний статус оцінювання. Об’єктом оцінювання може бути веб-застосунок, мобільний застосунок, інформаційна система або окремий програмний модуль.

Сутність «Характеристика якості» містить назви характеристик, за якими виконується оцінювання програмного продукту.

Сутність «Критерій оцінювання» деталізує характеристику якості. Один критерій належить до однієї характеристики, а кожна характеристика може мати декілька критеріїв. Наприклад, для характеристики «Здатність до взаємодії» можуть бути визначені такі критерії, як зрозумілість інтерфейсу, простота навігації та зручність введення даних.

Сутність «Призначення експертів» потрібна для того, щоб визначити, які експерти беруть участь в оцінюванні конкретного програмного продукту. Вона пов’язує користувача з роллю експерта та об’єкт оцінювання. Завдяки цьому експерт отримує доступ лише до тих програмних продуктів, які йому призначені.

Сутність «Експертна оцінка» зберігає значення оцінки, яку експерт виставив за конкретним критерієм. Крім числового значення, доцільно зберігати коментар експерта та дату внесення оцінки. Це дає змогу не лише отримати підсумковий бал, а й пояснити причини виставленої оцінки.

Сутність «Результат оцінювання» використовується для збереження підсумкових показників якості. У ній можуть зберігатися значення за окремими характеристиками, загальний показник якості програмного продукту та дата формування результату. Основні таблиці БД веб-застосунку подано в табл. 2.5.

Таблиця 2.5 – Основні таблиці бази даних веб-застосунку

Назва таблиці	**Призначення**
Roles	Зберігає ролі користувачів системи
Users	Зберігає дані користувачів веб-застосунку
SoftwareProducts	Зберігає інформацію про програмні продукти, які оцінюються
QualityCharacteristics	Зберігає характеристики якості програмного забезпечення
EvaluationCriteria	Зберігає критерії оцінювання, пов’язані з характеристиками якості
ExpertAssignments	Зберігає інформацію про призначення експертів до об’єктів оцінювання
ExpertScores	Зберігає експертні оцінки за критеріями
EvaluationResults	Зберігає підсумкові результати оцінювання
EvaluationHistory	Зберігає історію змін і попередніх оцінювань

Для кожної таблиці визначено набір основних полів. Структура таблиць бази даних подана в табл. 2.6.

Таблиця 2.6 – Структура основних таблиць бази даних

Назва таблиці	**Основні поля**
Roles	role_id, role_name, description
Users	user_id, role_id, full_name, email, password_hash, is_active, created_at
SoftwareProducts	product_id, name, description, created_by, status, created_at
QualityCharacteristics	characteristic_id, name, description
EvaluationCriteria	criterion_id, characteristic_id, name, description, weight
ExpertAssignments	assignment_id, product_id, expert_id, assigned_at, status
ExpertScores	score_id, product_id, criterion_id, expert_id, score_value, comment, created_at
EvaluationResults	result_id, product_id, total_score, quality_level, calculated_at
EvaluationHistory	history_id, product_id, result_id, action_description, created_at

Зв’язки між таблицями визначають логіку збереження та обробки даних. Таблиця Users пов’язана з таблицею Roles через поле role_id. Це дозволяє визначити роль кожного користувача. Таблиця SoftwareProducts пов’язана з користувачем, який створив об’єкт оцінювання, через поле created_by.

Таблиця EvaluationCriteria пов’язана з таблицею QualityCharacteristics через поле characteristic_id. Це означає, що кожен критерій належить до певної характеристики якості. Такий зв’язок дозволяє групувати оцінки за характеристиками та формувати підсумкові результати не лише для окремих критеріїв, а й для всієї характеристики.

Таблиця ExpertAssignments пов’язує експертів із програмними продуктами. Один програмний продукт може оцінюватися кількома експертами, і один експерт може бути призначений до кількох програмних продуктів. Для реалізації такого зв’язку використовується окрема проміжна таблиця.

Таблиця ExpertScores є однією з основних у системі, оскільки саме в ній зберігаються експертні оцінки. Вона пов’язана з програмним продуктом, критерієм

і експертом. Завдяки цьому можна визначити, хто саме виставив оцінку, за яким критерієм і для якого програмного продукту.

Таблиця EvaluationResults зберігає підсумкові результати оцінювання. Вона пов’язана з таблицею SoftwareProducts, оскільки кожен результат належить до конкретного програмного продукту. Якщо оцінювання виконується повторно, новий результат може бути збережений окремим записом, що дозволяє аналізувати історію змін. Загальну структуру бази даних веб-застосунку у вигляді ER-діаграми подано на рис. 2.7.

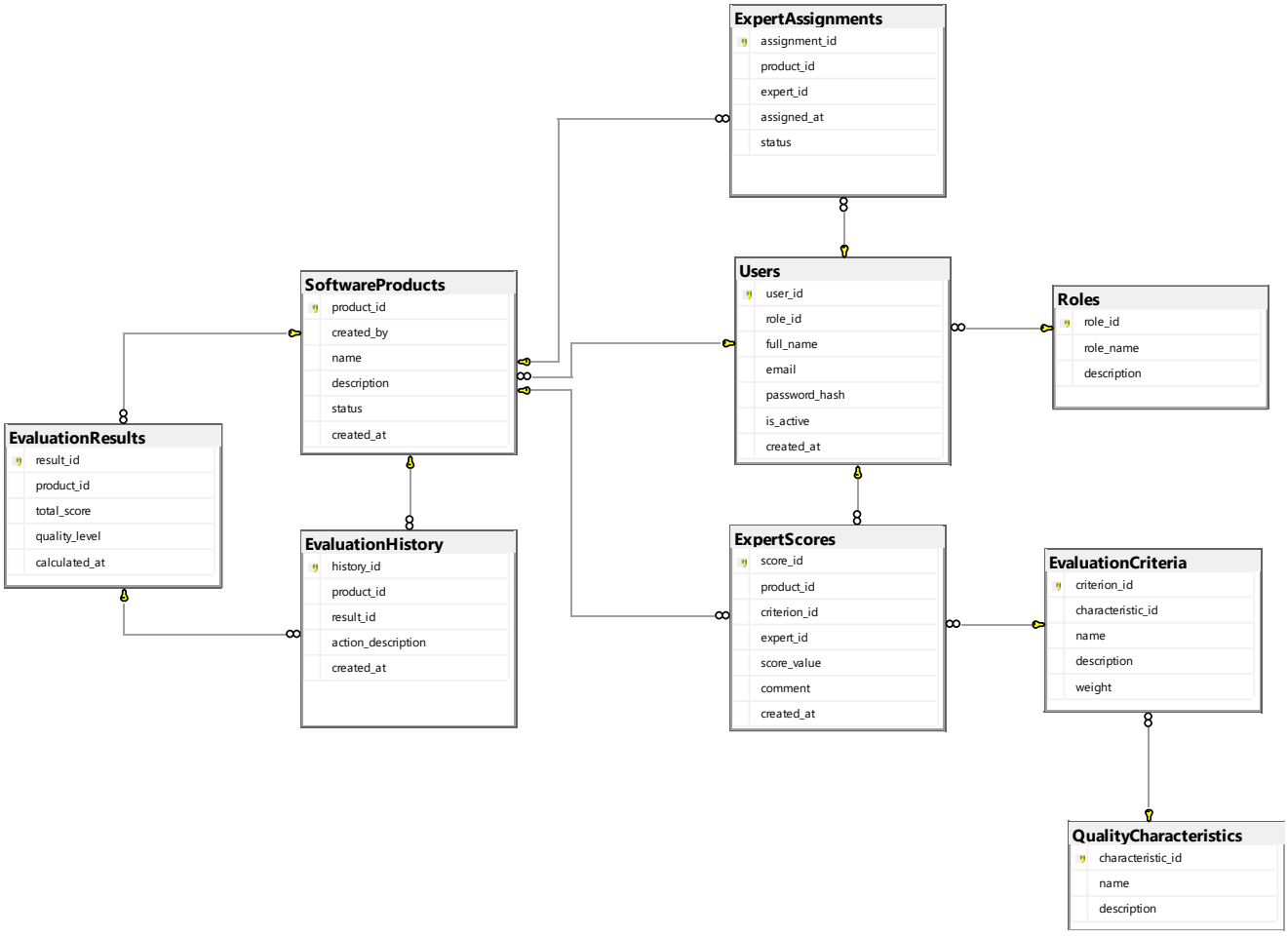

Рисунок .2.7 – ER-діаграма бази даних

Запропонована структура бази даних дозволяє зберігати всі дані, необхідні для роботи веб-застосунку. Вона підтримує розмежування користувачів за ролями, створення об’єктів оцінювання, формування критеріїв, призначення експертів, введення оцінок і збереження підсумкових результатів.

Під час реалізації бази даних важливо було забезпечити цілісність зв’язків між таблицями. Для цього первинні ключі використовуються для однозначної ідентифікації записів, а зовнішні ключі – для встановлення зв’язків між сутностями. Такий підхід дозволяє уникнути дублювання даних і забезпечити коректність роботи системи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ ЕКСПЕРТНОГО ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У третьому розділі розглядається реалізація веб-застосунку експертного оцінювання якості програмного забезпечення. Розробка веб-застосунку виконується з використанням технології ASP.NET MVC та мови програмування C# [16-18]. Такий підхід дозволяє розділити програмну систему на модель, контролери та представлення. Модель відповідає за опис сутностей предметної області, контролери обробляють запити користувачів і викликають потрібні методи, а представлення забезпечують відображення сторінок веб-застосунку. Для збереження даних використовується MS SQL Server. Доступ до цих даних реалізується через класи предметної області та репозиторії.

3.1. Реалізація структури класів предметної області та доступу до даних

Реалізацію веб-застосунку експертного оцінювання якості програмного забезпечення виконано з використанням технології ASP.NET MVC та мови програмування C#. Для збереження даних застосовано MS SQL Server, а доступ до таблиць бази даних організовано через класи предметної області, контекст бази даних і репозиторії.

У другому розділі було спроєктовано основні сутності системи та визначено їх зв’язки. На етапі реалізації ці сутності подано у вигляді класів C#, які відповідають таблицям бази даних. Такий підхід дозволяє працювати з даними не безпосередньо через SQL-запити в контролерах, а через об’єкти предметної області.

Для доступу до бази даних використовується контекст ApplicationDbContext [19]. Він містить набори даних для основних таблиць системи та забезпечує взаємодію між програмним кодом і MS SQL Server. Програмний код класу ApplicationDbContext проілюстровано на рис. 3.1

Лістинг 3.1 – Програмний код класу ApplicationDbContext

```
public class ApplicationDbContext : DbContext
{
    public DbSet<Role> Roles { get; set; }
    public DbSet<User> Users { get; set;}
    public DbSet<SoftwareProduct> SoftwareProducts { get; set;}
    public DbSet<QualityCharacteristic> QualityCharacteristics
{ get; set; }
    public DbSet<EvaluationCriterion> EvaluationCriteria {get;
set;}
    public DbSet<ExpertAssignment> ExpertAssignments { get;
set; }
    public DbSet<ExpertScore> ExpertScores { get; set; }
    public DbSet<EvaluationResult> EvaluationResults { get;
set; }
    public DbSet<EvaluationHistory> EvaluationHistory { get;
set; }
}
```

У цьому фрагменті кожна властивість типу DbSet відповідає окремій таблиці бази даних. Наприклад, Users використовується для роботи з користувачами, SoftwareProducts – з програмними продуктами, ExpertScores – з експертними оцінками, а EvaluationResults — з підсумковими результатами оцінювання. Реалізація класу експертної оцінки подано у лістингу 3.2.

Лістинг 3.2 – Реалізація класу EvaluationResults

```
public class ExpertScore
{
    public int ScoreId { get; set; }
    public int ProductId { get; set; }
    public int CriterionId { get; set; }
    public int ExpertId { get; set; }
    public int ScoreValue { get; set; }
    public string Comment { get; set; }
    public DateTime CreatedAt { get; set; }
    public SoftwareProduct Product { get; set; }
    public EvaluationCriterion Criterion { get; set; }
    public User Expert { get; set; }
}
```

Клас ExpertScore містить поля для збереження значення експертної оцінки, коментаря експерта та дати внесення запису. Також у ньому визначено зв’язки з програмним продуктом, критерієм оцінювання та користувачем-експертом. Це дає

змогу однозначно визначити, хто виставив оцінку, для якого об’єкта та за яким критерієм.

Для того щоб не розміщувати логіку доступу до даних безпосередньо в контролерах, у системі використано репозиторії [20-21]. Репозиторій є окремим класом, який містить методи для отримання, додавання, редагування та видалення даних. Такий підхід спрощує структуру програми та зменшує залежність контролерів від конкретної реалізації бази даних.

Програмний код інтерфейсу репозиторію для роботи з програмними продуктами, які необхідно оцінювати, представлено у лістингу 3.3.

Лістинг 3.3 – Інтерфейс ISoftwareProductRepository

```
public interface ISoftwareProductRepository
{
    IEnumerable<SoftwareProduct> GetAll();
    SoftwareProduct GetById(int productId);
    void Add(SoftwareProduct product);
    void Update(SoftwareProduct product);
    void Delete(int productId);
    void SaveChanges();
}
```

Інтерфейс ISoftwareProductRepository визначає набір основних операцій для роботи з об’єктами оцінювання. Метод GetAll() повертає список програмних продуктів, GetById() використовується для отримання одного запису, Add() додає новий об’єкт, Update() оновлює дані, Delete() видаляє запис, а SaveChanges() зберігає зміни в базі даних. Реалізацію цього інтерфейсу наведено у лістингу 3.4.

Лістинг 3.4 – Інтерфейс ISoftwareProductRepository

```
public class SoftwareProductRepository :
ISoftwareProductRepository
{
    private readonly ApplicationDbContext context;
    public SoftwareProductRepository(ApplicationDbContext
context)
    {
        this.context = context;
    }
```

```
    public IEnumerable<SoftwareProduct> GetAll()
    {
        return context.SoftwareProducts.ToList();
    }
    public SoftwareProduct GetById(int productId)
    {
        return context.SoftwareProducts
            .FirstOrDefault(p => p.ProductId == productId);
    }
    public void Add(SoftwareProduct product)
    {
        context.SoftwareProducts.Add(product);
    }
    public void Update(SoftwareProduct product)
    {
        context.SoftwareProducts.Update(product);
    }
    public void Delete(int productId)
    {
        var product = GetById(productId);
        if (product != null)
        {
            context.SoftwareProducts.Remove(product);
        }
    }
    public void SaveChanges()
    {
        context.SaveChanges();
    }
}
```

У наведеному у лістингу 3.4 фрагменті клас SoftwareProductRepository працює з таблицею програмних продуктів через ApplicationDbContext. Контролер, який використовує цей репозиторій, не звертається напряму до бази даних, а викликає готові методи репозиторію. Це робить програмний код більш зрозумілим і зручним для супроводу.

Аналогічно реалізовані репозиторії для користувачів, критеріїв оцінювання, експертних оцінок і результатів [22-24]. Наприклад, репозиторій експертних оцінок повинен містити методи для збереження оцінки, отримання оцінок за конкретним програмним продуктом і отримання оцінок окремого експерта. Програмний код репозиторію експертних оцінок показано у лістингу 3.5.

Лістинг 3.5 – Інтерфейс IExpertScoreRepository

```
public interface IExpertScoreRepository
{
    IEnumerable<ExpertScore> GetScoresByProduct(int productId);
    IEnumerable<ExpertScore> GetScoresByExpert(int expertId);
    void AddScore(ExpertScore score);
    void UpdateScore(ExpertScore score);
    void SaveChanges();
}
```

Для збереження нової експертної оцінки використовується метод AddScore(). Перед збереженням значення оцінки може бути перевірене на відповідність установленій шкалі, наприклад від 1 до 5. Це дозволяє запобігти внесенню некоректних даних до бази. Програмний код функції збереження оцінок продемонстровано у лістингу 3.6.

Лістинг 3.6 – Метод AddScore

```
public void AddScore(ExpertScore score)
{
    if (score.ScoreValue < 1 || score.ScoreValue > 5)
    {
        throw new ArgumentException("Оцінка повинна бути в
межах від 1 до 5.");
    }
    score.CreatedAt = DateTime.Now;
    context.ExpertScores.Add(score);
    context.SaveChanges();
}
```

Такий підхід дозволяє перенести частину перевірки даних у рівень доступу до даних або сервісної логіки. У результаті система не лише зберігає інформацію, а й контролює її коректність.

Для підключення до MS SQL Server у конфігураційному файлі веб-застосунку задається рядок з’єднання [25-27]. Він містить назву сервера, назву бази даних і параметри доступу. Програмний код конфігураційного файлу показаний у лістингу 3.7.

Лістинг 3.7 – Конфігураційний файл

```
<connectionStrings>
    <add name="ExpertQualityAssessmentDB"
         connectionString="Data Source=.;Initial
Catalog=ExpertQualityAssessmentDB;Integrated Security=True"
         providerName="System.Data.SqlClient" />
</connectionStrings>
```

Завдяки використанню конфігураційного файлу параметри підключення до бази даних не потрібно жорстко задавати в програмному коді. Це спрощує перенесення веб-застосунку на інший комп’ютер або сервер.

Отже, реалізація доступу до даних у веб-застосунку базується на використанні класів предметної області, контексту бази даних і репозиторіїв. Така структура забезпечує відокремлення логіки роботи з даними від контролерів, підвищує зручність супроводу програмного коду та створює основу для подальшої реалізації серверної логіки веб-застосунку.

3.2. Реалізація серверної логіки та контролерів веб-застосунку

Серверна логіка веб-застосунку експертного оцінювання якості програмного забезпечення реалізується за допомогою контролерів ASP.NET MVC. Контролери приймають HTTP-запити від клієнтської частини, викликають методи репозиторіїв або сервісних класів, виконують перевірку даних і повертають користувачу відповідне представлення.

У структурі MVC контролер є проміжною ланкою між моделлю даних і сторінкою відображення [28]. Він не повинен містити надмірної логіки доступу до бази даних, оскільки ця частина винесена до репозиторіїв. Основне завдання контролера полягає в тому, щоб отримати запит, підготувати потрібні дані, перевірити права користувача та передати результат у представлення.

Для розроблюваного веб-застосунку доцільно реалізувати такі основні контролери:

– AccountController;
– UsersController;

- SoftwareProductsController;
- CriteriaController;
- ExpertScoresController;
- ResultsController.

Контролер AccountController відповідає за вхід користувача до системи та завершення сеансу роботи. Він обробляє форму авторизації, перевіряє введені облікові дані та встановлює поточну роль користувача. Саме цей контролер забезпечує початковий доступ користувача до веб-застосунку.

Контролер UsersController використовується для керування обліковими записами. Його функції доступні адміністратору. Через цей контролер можна переглядати список користувачів, створювати нові записи, редагувати дані користувачів і змінювати їхній статус.

Контролер SoftwareProductsController відповідає за роботу з програмними продуктами, які підлягають оцінюванню. Він забезпечує створення нового об’єкта оцінювання, редагування його опису, перегляд списку програмних продуктів і зміну статусу оцінювання.

Контролер CriteriaController призначений для керування характеристиками якості та критеріями оцінювання. За його допомогою адміністратор може створювати критерії, задавати вагові коефіцієнти та прив’язувати критерії до відповідних характеристик якості.

Контролер ExpertScoresController використовується експертом для введення оцінок. Він отримує список призначених експерту програмних продуктів, відображає критерії оцінювання, приймає введені оцінки та передає їх до репозиторію для збереження.

Контролер ResultsController відповідає за перегляд підсумкових результатів. Він отримує з бази даних оцінки та результати розрахунків, формує дані для відображення і передає їх у відповідне представлення.

Загальну характеристику контролерів веб-застосунку подано в табл. 3.1.

Таблиця 3.1 – Основні контролери веб-застосунку

Контролер	**Призначення**
AccountController	Авторизація користувача та завершення сеансу роботи
UsersController	Керування обліковими записами користувачів
SoftwareProductsController	Створення та редагування об’єктів оцінювання
CriteriaController	Керування характеристиками якості та критеріями оцінювання
ExpertScoresController	Введення та збереження експертних оцінок
ResultsController	Формування та перегляд результатів оцінювання

Реалізація контролера для роботи з програмними продуктами наведена у лістингу 3.8.

Лістинг 3.8 – Контролер для роботи з програмними продуктами

```
public class SoftwareProductsController : Controller
{
    private readonly ISoftwareProductRepository
productRepository;
    public
SoftwareProductsController(ISoftwareProductRepository
productRepository)
    {
        this.productRepository = productRepository;
    }
    public ActionResult Index()
    {
        var products = productRepository.GetAll();
        return View(products);
    }
    public ActionResult Details(int id)
    {
        var product = productRepository.GetById(id);
        if (product == null)
        {
            return HttpNotFound();
        }
        return View(product);
    }
    [HttpGet]
    public ActionResult Create()
    {
        return View();
    }
    [HttpPost]
    public ActionResult Create(SoftwareProduct product)
    {
```

```
            if (!ModelState.IsValid)
            {
                return View(product);
            }
            product.CreatedAt = DateTime.Now;
            product.Status = "Створено";
            productRepository.Add(product);
            productRepository.SaveChanges();
            return RedirectToAction("Index");
        }
    }
```

У наведеному лістингу 3.8 контролер SoftwareProductsController отримує репозиторій через конструктор. Метод Index() формує список програмних продуктів і передає його у представлення. Метод Details() використовується для перегляду інформації про конкретний програмний продукт. Методи Create() обробляють відкриття форми створення нового об’єкта оцінювання та збереження введених даних.

Для обробки даних форми використовується два варіанти методу Create(). Метод з атрибутом [HttpGet] відкриває сторінку з формою, а метод з атрибутом [HttpPost] приймає дані, введені користувачем [29-31]. Перед збереженням перевіряється стан моделі за допомогою ModelState.IsValid. Якщо дані введено некоректно, користувач повертається до форми для виправлення помилок.

Аналогічний підхід використовується під час реалізації контролера критеріїв оцінювання. Адміністратор може переглядати список критеріїв, створювати нові критерії, редагувати їх і задавати вагові коефіцієнти. У лістингу 3.9 представлено програмний код контролера критеріїв оцінювання.

Лістинг 3.9 – Контролер критеріїв експертного оцінювання

```
public class CriteriaController : Controller
{
    private readonly ICriteriaRepository criteriaRepository;
    public CriteriaController(ICriteriaRepository
criteriaRepository)
    {
        this.criteriaRepository = criteriaRepository;
    }
    public ActionResult Index()
    {
```

```
        var criteria = criteriaRepository.GetAll();
        return View(criteria);
    }
    [HttpGet]
    public ActionResult Edit(int id)
    {
        var criterion = criteriaRepository.GetById(id);
        if (criterion == null)
        {
            return HttpNotFound();
        }
        return View(criterion);
    }
    [HttpPost]
    public ActionResult Edit(EvaluationCriterion criterion)
    {
        if (!ModelState.IsValid)
        {
            return View(criterion);
        }
        criteriaRepository.Update(criterion);
        criteriaRepository.SaveChanges();
        return RedirectToAction("Index");
    }
}
```

У контролері CriteriaController метод Index() виводить перелік критеріїв оцінювання, а методи Edit() забезпечують редагування вибраного критерію. Такий підхід дозволяє відокремити сторінку перегляду даних від сторінки зміни даних.

Для введення експертних оцінок використовується окремий контролер ExpertScoresController, програмний код якого представлений у лістингу 3.10. Він повинен враховувати роль користувача та показувати експерту лише ті програмні продукти, які йому призначені.

Лістинг 3.10 – Контролер ExpertScoresController

```
public class ExpertScoresController : Controller
{
    private readonly IExpertScoreRepository scoreRepository;
    private readonly ISoftwareProductRepository
productRepository;
    public ExpertScoresController(
        IExpertScoreRepository scoreRepository,
        ISoftwareProductRepository productRepository)
    {
        this.scoreRepository = scoreRepository;
```

```
        this.productRepository = productRepository;
    }
    [HttpGet]
    public ActionResult Evaluate(int productId)
    {
        var product = productRepository.GetById(productId);
        if (product == null)
        {
            return HttpNotFound();
        }
        return View(product);
    }

    [HttpPost]
    public ActionResult SaveScore(ExpertScore score)
    {
        if (!ModelState.IsValid)
        {
            return View("Evaluate", score);
        }
        score.CreatedAt = DateTime.Now;
        scoreRepository.AddScore(score);
        scoreRepository.SaveChanges();
        return RedirectToAction("Evaluate", new { productId =
score.ProductId });
    }
}
```

Метод Evaluate() відкриває сторінку оцінювання конкретного програмного продукту. На цій сторінці експерт бачить перелік критеріїв і поля для введення оцінок. Метод SaveScore() приймає введену оцінку, перевіряє її коректність і зберігає у базі даних.

Для перегляду підсумкової інформації використовується контролер ResultsController. Він отримує результати оцінювання з бази даних і передає їх у представлення. Програмний код контролера ResultsController наведений у лістингу 3.11.

Лістинг 3.11 – Контролер ResultsController

```
public class ResultsController : Controller
{
    private readonly IResultRepository resultRepository;
    public ResultsController(IResultRepository
resultRepository)
    {
        this.resultRepository = resultRepository;
```

```
    }
    public ActionResult ProductResults(int productId)
    {
        var result =
resultRepository.GetResultByProduct(productId);
        if (result == null)
        {
            return View("NoResults");
        }
        return View(result);
    }
}
```

Якщо для програмного продукту ще не сформовано результат, користувачу може бути показано окреме представлення NoResults. Це дозволяє уникнути помилок і зробити роботу системи зрозумілішою для користувача.

Важливою частиною серверної логіки є перевірка прав доступу [32]. Для цього окремі контролери або методи контролерів можуть бути обмежені відповідно до ролі користувача. Наприклад, керування користувачами та критеріями повинно бути доступним лише адміністратору, а введення оцінок – лише експерту. Програмний код перевірки прав доступу користувачів наведений у лістингу 3.12.

Лістинг 3.12 – Перевірка прав доступу користувачів

```
[Authorize(Roles = "Адміністратор")]
public class UsersController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Атрибут [Authorize] дозволяє обмежити доступ до контролера або окремого методу. Якщо користувач не має відповідної ролі, система не повинна дозволяти виконання цієї дії.

Отже, серверна логіка веб-застосунку реалізується за допомогою контролерів ASP.NET MVC, які обробляють запити користувачів, взаємодіють із репозиторіями та передають дані у представлення. Така структура дозволяє розділити

відповідальність між компонентами системи, зробити код більш зрозумілим і забезпечити подальший розвиток веб-застосунку.

3.3. Реалізація алгоритму розрахунку результатів експертного оцінювання

Однією з основних функцій веб-застосунку є автоматичний розрахунок результатів експертного оцінювання якості програмного забезпечення. Після того як експерти вводять оцінки за визначеними критеріями, система повинна опрацювати ці дані, обчислити показники за окремими характеристиками якості та сформувати загальний результат для програмного продукту.

У розроблюваному веб-застосунку використовується шкала оцінювання від 1 до 5. Значення 1 відповідає низькому рівню відповідності критерію, а значення 5 — високому рівню відповідності. Для кожного критерію може задаватися ваговий коефіцієнт, який показує важливість цього критерію під час формування підсумкової оцінки.

Алгоритм розрахунку результатів складається з таких основних етапів:

– отримання списку критеріїв для програмного продукту;

– отримання експертних оцінок за кожним критерієм;

– перевірка наявності достатньої кількості оцінок;

– обчислення середньої оцінки за кожним критерієм;

– обчислення зважених показників за характеристиками якості;

– формування загального показника якості програмного продукту;

– визначення рівня якості;

– збереження результату в базі даних.

Спочатку система отримує з бази даних усі оцінки, які були виставлені експертами для вибраного програмного продукту. Кожна оцінка пов’язана з конкретним критерієм, експертом і програмним продуктом. Це дозволяє групувати оцінки за критеріями та характеристиками якості.

Для кожного критерію обчислюється середнє значення оцінок, виставлених експертами. Якщо один критерій оцінювали кілька експертів, система враховує всі значення та визначає узагальнену оцінку. Це дозволяє зменшити вплив суб’єктивної думки окремого експерта.

Для реалізації цього алгоритму створено окремий сервісний клас EvaluationService. Він відповідає за отримання оцінок, виконання розрахунків і формування результату оцінювання. Лістинг 3.13 реалізує клас EvaluationService.

Лістинг 3.13 – Клас EvaluationService

```
public class EvaluationService
{
    private readonly IExpertScoreRepository scoreRepository;
    private readonly ICriteriaRepository criteriaRepository;
    private readonly IResultRepository resultRepository;
    public EvaluationService(
        IExpertScoreRepository scoreRepository,
        ICriteriaRepository criteriaRepository,
        IResultRepository resultRepository)
    {
        this.scoreRepository = scoreRepository;
        this.criteriaRepository = criteriaRepository;
        this.resultRepository = resultRepository;
    }
    public EvaluationResult CalculateResult(int productId)
    {
        var scores = scoreRepository
            .GetScoresByProduct(productId)
            .ToList();
        var criteria = criteriaRepository
            .GetAll()
            .ToList();
        decimal totalWeightedScore = 0;
        decimal totalWeight = 0;
        foreach (var criterion in criteria)
        {
            var criterionScores = scores
                .Where(s => s.CriterionId ==
criterion.CriterionId)
                .ToList();
            if (!criterionScores.Any())
            {
                continue;
            }
            decimal averageScore = criterionScores
                .Average(s => s.ScoreValue);
```

```
            totalWeightedScore += averageScore *
criterion.Weight;
            totalWeight += criterion.Weight;
        }

        decimal totalScore = totalWeight > 0
            ? totalWeightedScore / totalWeight
            : 0;
        var result = new EvaluationResult
        {
            ProductId = productId,
            TotalScore = Math.Round(totalScore, 2),
            QualityLevel = DefineQualityLevel(totalScore),
            CalculatedAt = DateTime.Now
        };
        resultRepository.Add(result);
        resultRepository.SaveChanges();
        return result;
    }
    private string DefineQualityLevel(decimal totalScore)
    {
        if (totalScore >= 4.5m)
        {
            return "Високий рівень якості";
        }
        if (totalScore >= 3.5m)
        {
            return "Достатній рівень якості";
        }
        if (totalScore >= 2.5m)
        {
            return "Середній рівень якості";
        }
        return "Низький рівень якості";
    }
}
```

У лістингу 3.13 метод CalculateResult() отримує всі експертні оцінки для вибраного програмного продукту та список критеріїв оцінювання. Далі для кожного критерію визначається середня оцінка. Якщо для певного критерію оцінок ще немає, такий критерій не враховується в розрахунку.

Змінні totalWeightedScore і totalWeight використовуються для обчислення зваженого середнього значення. Після завершення циклу система визначає загальний показник якості totalScore. Якщо оцінок немає або сума ваг дорівнює нулю, результат встановлюється рівним 0.

Метод DefineQualityLevel() використовується для інтерпретації числового результату. Якщо підсумковий показник наближається до максимального значення, програмний продукт отримує високий рівень якості. Якщо значення є нижчим, система визначає достатній, середній або низький рівень якості.

Для запуску розрахунку використовується метод контролера ResultsController. Він викликає сервіс розрахунку та передає сформований результат у представлення.

Метод Calculate() приймає ідентифікатор програмного продукту, викликає метод розрахунку та після цього перенаправляє користувача на сторінку перегляду результату. Такий підхід дозволяє відокремити логіку обчислень від контролера і зосередити її в окремому сервісному класі.

Окремо реалізується розрахунок показників за кожною характеристикою якості. Для цього оцінки групуються не лише за критеріями, а й за характеристиками. Це дозволяє показати користувачу не тільки загальний рівень якості, а й сильні та слабкі сторони програмного продукту. Лістинг 3.14 демонструє реалізацію розрахунку показників за кожною характеристикою якості.

Лістинг 3.14 – Реалізація групування оцінок якості за характеристиками

```
public IEnumerable<CharacteristicResult>
CalculateCharacteristicResults(int productId)
{
    var scores = scoreRepository
        .GetScoresByProduct(productId)
        .ToList();
    var criteria = criteriaRepository
        .GetAll()
        .ToList();
    var results = criteria
        .GroupBy(c => c.CharacteristicId)
        .Select(group =>
        {
            decimal weightedSum = 0;
            decimal weightSum = 0;
            foreach (var criterion in group)
            {
                var criterionScores = scores
                    .Where(s => s.CriterionId ==
criterion.CriterionId)
                    .ToList();
```

```
                if (!criterionScores.Any())
                {
                    continue;
                }
                decimal averageScore = criterionScores
                    .Average(s => s.ScoreValue);
                weightedSum += averageScore * criterion.Weight;
                weightSum += criterion.Weight;
            }
            return new CharacteristicResult
            {
                CharacteristicId = group.Key,
                Score = weightSum > 0
                    ? Math.Round(weightedSum / weightSum, 2)
                    : 0
            };
        });
    return results;
}
```

Результати за характеристиками використовуються для побудови таблиці або діаграми на сторінці результатів. Наприклад, якщо система показує, що найнижчу оцінку отримала характеристика «Здатність до взаємодії», це означає, що програмний продукт потребує покращення інтерфейсу або сценаріїв взаємодії з користувачем.

Для підвищення коректності роботи алгоритму необхідно враховувати декілька обмежень. Значення експертної оцінки повинно бути в межах установленої шкали, наприклад від 1 до 5. Ваговий коефіцієнт критерію повинен бути додатним числом. Розрахунок результату доцільно виконувати лише тоді, коли для програмного продукту внесено хоча б частину оцінок.

Під час збереження результату в базі даних фіксується дата розрахунку. Це дає змогу зберігати історію оцінювання та порівнювати результати після повторного аналізу програмного продукту. Якщо програмний продукт було доопрацьовано, новий результат можна порівняти з попереднім і визначити, чи покращилась його якість.

Отже, реалізований алгоритм розрахунку результатів експертного оцінювання дозволяє автоматично опрацьовувати введені оцінки, враховувати вагові коефіцієнти критеріїв, визначати загальний рівень якості та зберігати

підсумкові результати. Це зменшує кількість ручних розрахунків, підвищує узгодженість оцінювання та забезпечує користувачу зрозуміле представлення якості програмного продукту.

3.4. Реалізація користувацьких інтерфейсів веб-застосунку

Користувацький інтерфейс веб-застосунку експертного оцінювання якості програмного забезпечення реалізовано з урахуванням рольової моделі системи. Основна ідея полягає в тому, що після авторизації користувач отримує доступ лише до тих сторінок і функцій, які відповідають його ролі. Це спрощує роботу із системою, зменшує кількість зайвих елементів на екрані та знижує ризик помилкових дій. У веб-застосунку передбачено окремі інтерфейсні сценарії для адміністратора, експерта та користувача результатів. Для всіх ролей використано єдину візуальну структуру: верхню панель із назвою системи, блоком поточного користувача, ліве навігаційне меню та основну робочу область. Такий підхід забезпечує цілісність інтерфейсу та робить взаємодію з веб-застосунком зрозумілою для користувачів.

Початковим екраном системи є сторінка авторизації, яка показана на рис. 3.1.

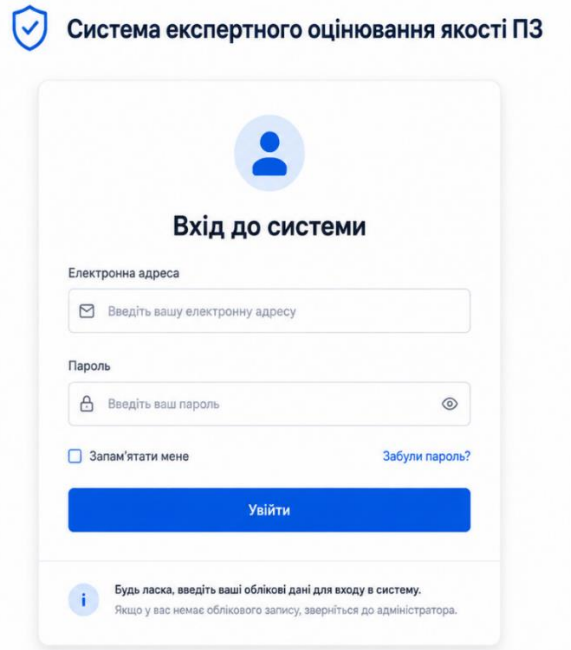

Рисунок 3.1 – Інтерфейс авторизації користувача у веб-застосунку

Форма авторизації містить два основні поля введення, прапорець запам’ятовування користувача та кнопку входу до системи. У нижній частині форми подано коротке службове повідомлення.

Як видно з рис. 3.1, інтерфейс авторизації має просту структуру та не містить зайвих елементів. Основна увага користувача зосереджена на введенні облікових даних. Використання єдиного стилю, піктограми користувача, підписів до полів і основної кнопки входу робить форму зрозумілою навіть для користувача, який працює із системою вперше.

Після успішної авторизації адміністратор переходить до робочої панелі керування. Для адміністратора передбачено доступ до основних довідників і функцій системи: користувачів, програмних продуктів, характеристик якості, критеріїв оцінювання, призначення експертів і результатів. Один із основних екранів адміністратора – сторінка керування програмними продуктами.

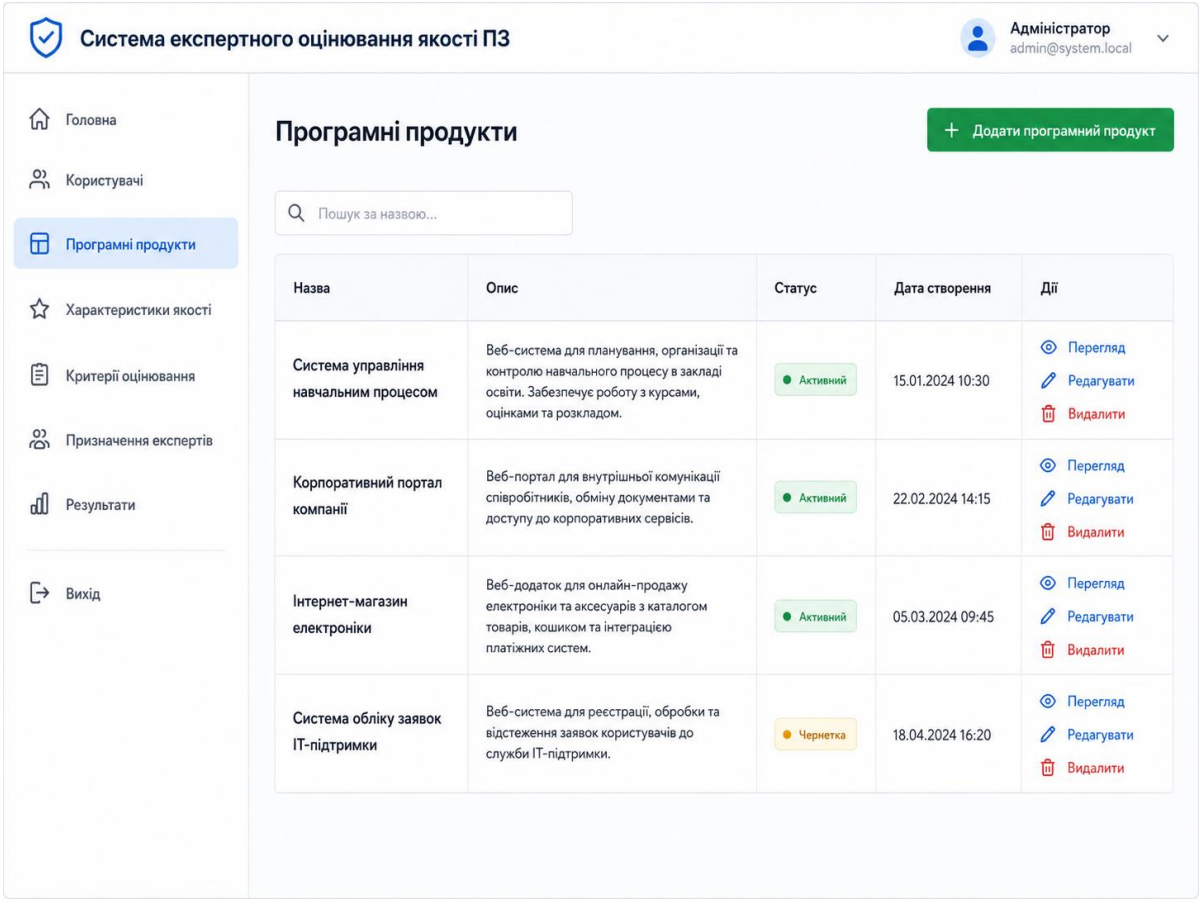

Рисунок 3.2 – Інтерфейс адміністратора для керування програмними продуктами

На рис. 3.2 подано екран адміністратора, на якому відображається список програмних продуктів, що можуть бути оцінені експертами. У лівій частині розміщено навігаційне меню, активним пунктом якого є «Програмні продукти». У центральній частині сторінки подано таблицю з назвами програмних продуктів, описами, статусами, датами створення та доступними діями.

Для кожного програмного продукту адміністратор може виконати перегляд, редагування або видалення запису. Також на сторінці передбачено поле пошуку за назвою та кнопку додавання нового програмного продукту. Такий інтерфейс є зручним для роботи з переліком об’єктів оцінювання, оскільки вся основна інформація подана в табличній формі.

Окремий інтерфейс, який представлено на рис. 3.3, передбачено для експерта.

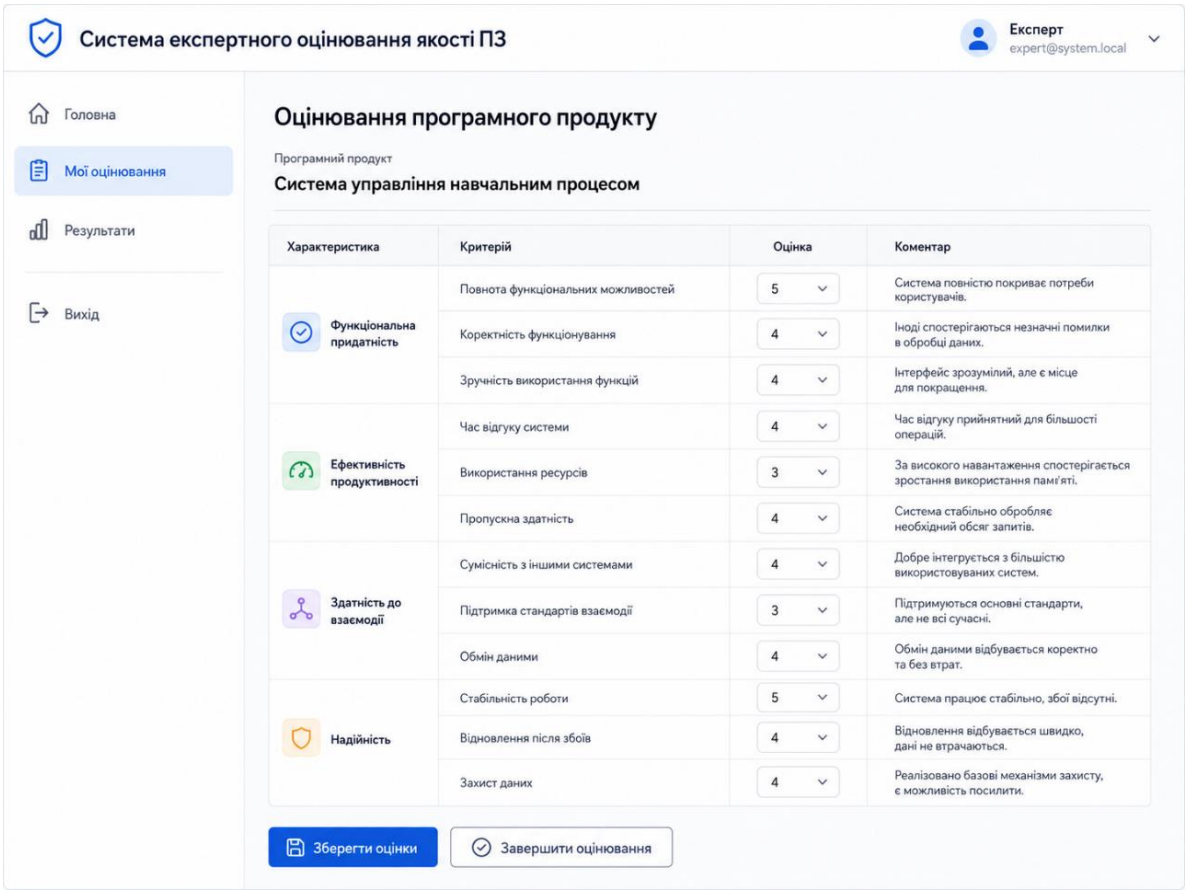

Рисунок 3.3 – Інтерфейс експерта для введення оцінок програмного продукту

Після входу до системи експерт бачить лише ті розділи, які пов’язані з його роботою: головну сторінку, призначені оцінювання, результати та вихід із системи.

Основним екраном експерта є сторінка введення оцінок для конкретного програмного продукту. У верхній частині робочої області вказано назву об’єкта оцінювання. Нижче розміщено таблицю, у якій критерії згруповано за характеристиками якості. Для кожного критерію експерт може вибрати числову оцінку та додати короткий коментар. Така структура сторінки дозволяє експерту послідовно пройти всі критерії оцінювання та не пропустити важливі характеристики програмного продукту. У таблиці передбачено окремі колонки для характеристики, критерію, оцінки та коментаря. У нижній частині сторінки розміщено кнопки збереження оцінок і завершення оцінювання. Це відповідає логіці роботи експерта: спочатку внести або змінити оцінки, а після завершення – підтвердити готовність результатів до подальшої обробки.

Для користувача результатів реалізовано сторінку перегляду підсумкової інформації, яка показана на рис. 3.4. Вона призначена для аналізу якості програмного продукту після завершення експертного оцінювання та розрахунку підсумкових показників.

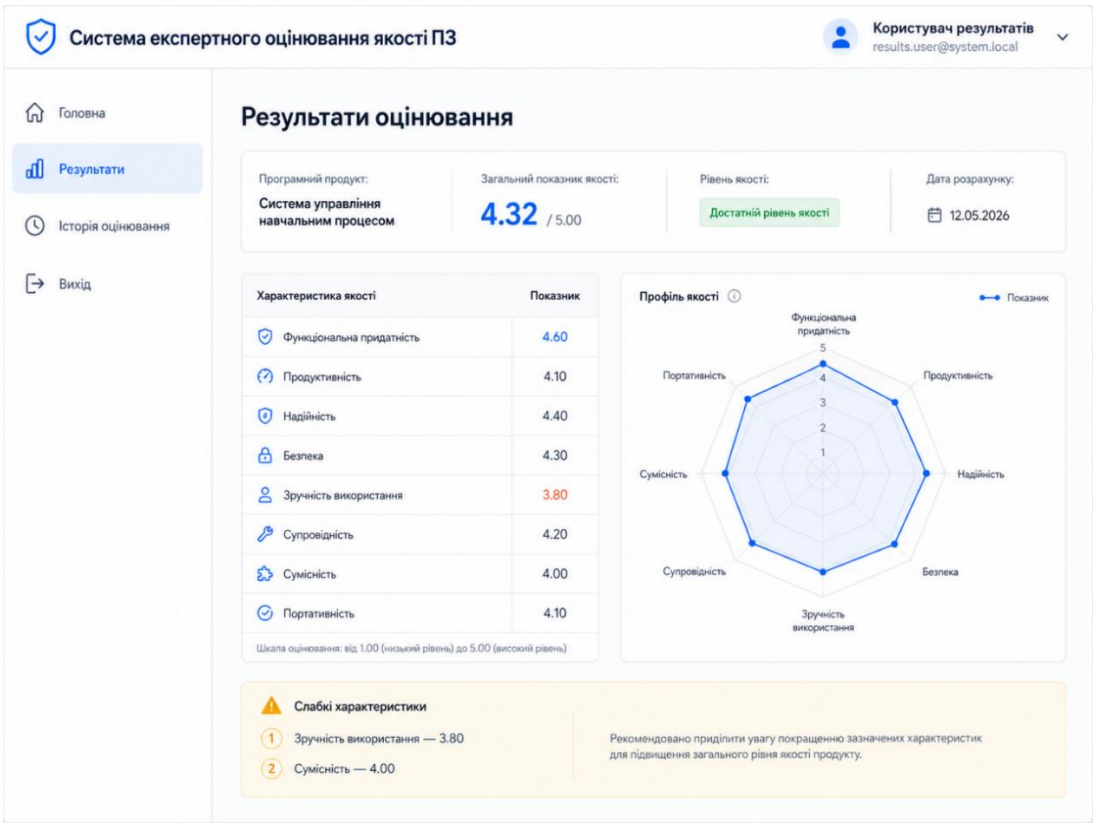

Рисунок 3.4 – Інтерфейс перегляду результатів експертного оцінювання

У верхній частині сторінки відображається короткий підсумок: назва програмного продукту, загальний показник якості, рівень якості та дата розрахунку. Такий блок дає змогу швидко оцінити загальний стан програмного продукту без детального аналізу всіх критеріїв.

Нижче подано таблицю з показниками за окремими характеристиками якості. Це дозволяє користувачу побачити, які властивості програмного продукту оцінені найвище, а які потребують покращення. Для кращого сприйняття результатів на сторінці також передбачено графічне подання профілю якості. Діаграма дає змогу візуально порівняти показники за характеристиками та швидко визначити слабкі місця програмного продукту.

Окремо на сторінці виділено блок «Слабкі характеристики». У ньому система показує характеристики, які отримали найнижчі значення. Такий елемент інтерфейсу є корисним для прийняття рішень щодо подальшого вдосконалення програмного продукту, оскільки одразу вказує на проблемні напрями.

Усі інтерфейси веб-застосунку виконано в єдиному стилі. Для навігації використовується ліва бічна панель, а у верхній частині сторінки відображається назва системи та дані поточного користувача. Активний пункт меню виділяється іншим кольором, що допомагає користувачу розуміти, у якому розділі системи він перебуває.

Важливим елементом інтерфейсу є використання таблиць, статусів, піктограм і кольорових позначень. Наприклад, активні програмні продукти позначаються зеленим статусом, а слабкі характеристики у результатах виділяються окремим попереджувальним блоком. Це підвищує інформативність сторінок і робить результати оцінювання простішими для сприйняття.

Таким чином, реалізовані користувацькі інтерфейси забезпечують підтримку основних сценаріїв роботи веб-застосунку. Адміністратор отримує засоби керування програмними продуктами та критеріями, експерт – зручну форму введення оцінок, а користувач результатів – інструменти перегляду та аналізу підсумкових показників якості. Така організація інтерфейсу відповідає рольовій моделі системи та забезпечує зручну взаємодію користувачів із веб-застосунком.

3.5. Тестування функціональних можливостей веб-застосунку

Після реалізації основних модулів веб-застосунку виконано тестування його функціональних можливостей. Метою тестування було перевірити коректність роботи авторизації, рольового доступу, керування програмними продуктами, введення експертних оцінок, розрахунку підсумкових результатів та відображення інформації користувачу.

Тестування проводилося за основними сценаріями роботи користувачів. Для адміністратора перевірено можливість входу до системи, створення програмного продукту, редагування його даних, додавання критеріїв оцінювання та призначення експертів. Для експерта перевірено доступ до призначених об’єктів, введення оцінок за критеріями, додавання коментарів і збереження результатів. Для користувача перевірено перегляд підсумкової оцінки, показників за характеристиками якості та історії оцінювання. Основні результати тестування подано в табл. 3.2.

Таблиця 3.2 – Результати тестування функціональних вимог

№	Тестовий сценарій	Очікуваний результат	Результат
1	Вхід користувача з правильними даними	Користувач переходить до системи	Виконано
2	Вхід з неправильним паролем	Система виводить повідомлення про помилку	Виконано
3	Додавання нового програмного продукту	Новий об’єкт зберігається в базі даних	Виконано
4	Додавання критерію оцінювання	Критерій відображається у списку	Виконано
5	Призначення експерта до об’єкта	Експерт бачить призначене оцінювання	Виконано
6	Введення експертної оцінки	Оцінка зберігається в базі даних	Виконано
7	Введення оцінки поза шкалою	Система не приймає некоректне значення	Виконано
8	Розрахунок підсумкового результату	Система формує загальний показник якості	Виконано
9	Перегляд результатів	Користувач бачить підсумкові показники	Виконано
10	Спроба доступу до чужої ролі	Система обмежує доступ	Виконано

Під час тестування також перевірено коректність повідомлень про помилки. Якщо користувач залишає обов’язкове поле порожнім або вводить некоректне значення, система не зберігає такі дані та виводить відповідне повідомлення. Це дозволяє зменшити кількість помилок під час роботи із системою.

Окремо перевірено правильність розрахунку підсумкового показника якості. Для цього було введено тестові експертні оцінки за кількома критеріями та порівняно результат, сформований системою, із ручним розрахунком. Отримані значення збіглися, що підтверджує коректність реалізованого алгоритму.

Отже, результати тестування показали, що основні функціональні можливості веб-застосунку працюють коректно. Система забезпечує авторизацію користувачів, розмежування доступу, введення експертних оцінок, розрахунок підсумкових показників і перегляд результатів оцінювання.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У даному розділі проаналізовано питання охорони праці, де розкрито питання необхідності і способів дотримання вимог охорони праці та техніки безпеки для розробників програмних систем. Окрім цього, у розділі визначено фактори, що впливають на функціональний стан користувачів комп'ютера. Для зменшення негативного впливу таких факторів наведено заходи. Яких необхідно дотримуватися при використанні комп'ютерів та оргтехніки.

4.1. Аварії з викидом радіоактивних речовин та захист населення від впливу радіації

При виникненні надзвичайної ситуації, зокрема, при аварійному викиданні в атмосферу радіоактивних речовин можливі такі види радіоактивного впливу на населення [32, 33]:

– зовнішнє опромінення при проходженні радіоактивної хмари;

– внутрішнє опромінення при вдиханні радіоактивних аерозолів (інгаляційна небезпека);

– контактне опромінення внаслідок радіоактивного забруднення шкіри і одягу;

– зовнішнє опромінення, зумовлене радіоактивним забрудненням поверхні землі, будівель, споруд та ін.;

– внутрішнє опромінення при використанні забруднених продуктів харчування і води.

Розрахункові дані та результати прямих вимірювань рівня радіації і дози опромінення мають бути основою для вжиття заходів захисту населення від зовнішнього і внутрішнього опромінення, в тому числі й профілактичне застосування стабільного йоду [32].

Основою розробки заходів захисту населення в умовах радіоактивного забруднення при ядерній аварії є рекомендації Міжнародного агентства з атомної енергії (МАГАТЕ) 1988 р., а також норми радіаційної безпеки України/

Враховуючи рівень радіації, а також прогноз можливих аварійних викидів радіоактивних речовин та метеорологічні дані, приймається рішення про проведення таких термінових і невідкладних заходів захисту в умовах ранньої фази радіаційної аварії:

– укриття населення;

– обмеження перебування населення на відкритій місцевості;

– евакуація у разі загрози здоров'ю;

– проведення йодової профілактики;

– тимчасова заборона вживання продуктів харчування і води із зони радіоактивного забруднення.

Крім цих заходів у період ранньої і пізньої фази проводяться довгострокові заходи:

– тимчасове відселення;

– евакуація — переселення на постійне місце проживання;

– обмеження вживання води і продуктів харчування забруднених радіоактивними речовинами;

– заходи захисту при виробництві продукції тваринництва, рослинництва і лісогосподарської діяльності;

– дезактивація території і будівель;

– інші заходи: гідрологічні, протиповіневі, обмеження лісокористування, полювання, рибної ловлі, перебування у полі при проведенні сільськогосподарських робіт.

Критерієм для прийняття рішення про заходи захисту населення на ранній і середніх фазах після аварії є дози зовнішнього і внутрішнього опромінення (табл. 4.1) з установленими двома рівнями радіаційного впливу — нижнім і верхнім — згідно з рекомендацією МАГАТЕ і 97Д-2000.

При прогнозованому опроміненні, що не перевершує нижнього рівня, заходи, перелічені в табл. 4.1 не проводяться. Якщо прогнозоване опромінення перевищує нижній рівень, але не досягає верхнього рівня, то проведення вказаних заходів може бути відкладене [33].

Таблиця 4.1 – Критерії для прийняття рішень на ранній фазі розвитку аварії

Захисні заходи	**Дозові критерії (прогнозована доза за перші 10 діб), мЗв**			
	Все тіло		**Окремі органи (легені, щитовидна залоза, шкіра)**	
	Нижній рівень	**Верхній рівень**	**Нижній рівень**	**Верхній рівень**
Укриття, захист органів дихання і шкіри	5	50	50	500
Йодова профілактика:				
дорослі	—	—	50*	500*
діти, вагітні жінки	—	—	50*	250*
Евакуація:				
дорослі діти, вагітні жінки	50 10	500 50	500 200*	5000 500*

Радіаційний захист населення включає в себе:

– організацію безперервного контролю, виявлення та оцінку радіаційної та хімічної обстановки в районах розміщення радіаційно-небезпечних об'єктів;

– завчасне накопичення, підтримання в готовності і використання при необхідності засобів індивідуального захисту, приладів радіаційної розвідки і контролю;

– створення, виробництво та застосування уніфікованих засобів захисту, приладів і комплектів радіаційної розвідки і дозиметричного контролю;

– придбання населенням у встановленому порядку в особисте користування засобів індивідуального захисту та контролю за використанням їх за призначенням;

– своєчасне впровадження і застосування засобів і методів виявлення та оцінки масштабів і наслідків аварій на радіаційно-небезпечних об'єктах;

– створення і використання на радіаційно-небезпечних об'єктах систем (переважно автоматизованих) контролю обстановки і локальних систем оповіщення;

– розробку і застосування, за необхідності, режимів радіаційного захисту населення і функціонування об'єктів економіки та інфраструктури в умовах забрудненості (зараженості) місцевості;

– завчасне пристосування об'єктів комунально-побутового обслуговування і транспортних підприємств для проведення спеціальної обробки одягу, майна і транспорту, проведенням цієї обробки в умовах аварій;

– навчання населення використання засобів індивідуального захисту і правилам поведінки на забрудненій (зараженої) території.

Радіація на сьогодні є чи не найнебезпечнішим фактором впливу не тільки на людину, але й на усі живі організми на планеті. Підтвердженням цього є аварія на Чорнобильській АЕС, наслідки якої відчувають до сьогодні як в Україні, так і за її межами. Дотримання рекомендацій щодо захисту населення від впливу радіації, які проаналізовано вище, дає змогу мінімізувати ризики, пов’язані із загибеллю великої кількості людей, а також зберегти їхнє здоров’я.

4.2. Заходи щодо боротьби з шкідливою дією ультразвуку на організм людини

ДСН 3.3.6.037 – 99 „Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку” визначає допустимі рівні впливу різних видів шуму на організм людини.

Ультразвук представляє собою механічні коливання пружного середовища і відрізняються від звукових хвиль більш високою частотою, що перевищує верхній поріг чутності. Ультразвукові хвилі поширюються в будь-якому пружному середовищі (рідкому, твердому, газоподібному), краще в металах, воді, гірше в повітрі. При проходженні в різних середовищах ультразвукові хвилі по різному поглинаються ними. Абсорбційні властивості м’язової тканини вище жирової, в

сірій мозковій речовині поглинання майже в два рази вище, ніж в білій. Найбільше поглинання спостерігається в кістковій тканині, найменше – в спинномозковій речовині.

Поглинання ультразвуку супроводжується нагріванням середовища, причому термічний ефект посилюється з підвищенням частоти коливань. Також при проходженні ультразвуку в рідині виникає ефект кавітації (пароутворення та наступного схлопування бульбашок пари з одночасною конденсацією пара в струмі рідини, що супроводжується шумом та гідравлічними ударами, утворення в рідині порожнин, які заповнюються паром самої рідини). З цим явищем пов'язана механічна дія ультразвуку. Утворення кавітаційних порожнин супроводжується появою на поверхнях електричних зарядів, що викликають люмінесцентне світіння, іонізацію молекул води. З цими явищами пов'язані хімічні ефекти – окислювальна дія ультразвуку, прискорення хімічних реакцій, руйнування органічних сполук.

Прояви дії ультразвуку широко використовується в багатьох галузях промисловості для інтенсифікації процесів хімічного травлення, нанесення металевого покриття, очищення, змивання та знежирення деталей і виробів, дефектоскопії (оцінка якості зварних швів, структури сплаву).

За способом передачі від джерела до людини ультразвук поділяють на: повітряний (передається через повітря) та контактний (передається на руки людини, що працює через тверде чи рідинне середовище).

За спектром ультразвук поділяють на: низькочастотний (коливання частотою від 1,2 х 104 до 1,0 х 106 Гц, що передаються людині повітряним чи контактним шляхом) та високочастотний (коливання частотою від 1,0 х105 до 1,0 х109 Гц , що передаються людині тільки контактним шляхом).

Параметрами повітряного ультразвуку, що нормуються у робочій зоні, є рівні звукового тиску в третиноктавних смугах з середньогеометричними частотами 12,5; 16,0; 20,0; 25,0; 31,5; 40,0; 63,0; 80,0;100,0 кГц. Для контактного ультразвуку параметром, що нормується, є пікове значення віброшвидкості в частотному діапазоні від 0,1 МГц до 10,0 МГц або його логарифмічний рівень. Допускається, також, застосовувати як параметр інтенсивність ультразвуку [33].

Ультразвук, так само як і інфразвук, орган слуху людини не сприймає, однак він може спричиняти біль голови, загальну втому, розлади серцево-судинної та нервової систем. При клінічному обстеженні може бути виявлений астенічний синдром. У осіб, що тривалий час зайняті експериментальною роботою на ультразвукових установках, іноді спостерігаються діенцефальні порушення (зниження ваги, різкий підйом вмісту цукру в крові з повільним зниженням до вихідного рівня, підвищення механічного збудження м'язів). Можливі порушення периферичної нервової системи (оніміння, зниження чутливості, гіпергідроз) порушення вестибулярного апарата. Периферичні порушення обумовлені переважно контактним впливом ультразвукових коливань [33].

Заходи щодо зниження шкідливої дії ультразвуку направлені на обмеження впливу шуму та ультразвуку, що передається через повітря, а також контактним засобом.

Для унеможливлення впливу контактного ультразвуку роботи з коливними рідинними середовищами (завантаження, вивантаження) необхідно проводити при вимкненому джерелі ультразвуку або використовувати для цього спеціальні інструменти, що мають ручки з еластичним покриттям, наприклад, гумовим. Як засоби індивідуального захисту використовують протишумові навушники (дія через повітря) та двошарові рукавички із зовнішнім гумовим шаром (контактна дія) [34].

Робітники, які працюють в умовах впливу ультразвуку, підлягають щорічному періодичному медичному огляду з обов'язковим залученням до складу лікарняної комісії невропатолога, офтальмолога, хірурга та проведенням досліджень вібраційної чутливості (за показанням) [34].

ВИСНОВКИ

У кваліфікаційній роботі виконано проєктування та створення веб-застосунку експертного оцінювання якості програмного забезпечення. У процесі виконання роботи розглянуто предметну область оцінювання якості ПЗ, визначено роль експертного підходу та обґрунтовано потребу у створенні програмного засобу, який дозволяє впорядкувати процес оцінювання, зберігати результати та автоматизувати розрахунок підсумкових показників.

У першому розділі проаналізовано основні підходи до оцінювання якості програмного забезпечення та розглянуто моделі якості ПЗ. Як основу для формування критеріїв оцінювання обрано модель ISO/IEC 25010:2023, яка охоплює характеристики якості програмного продукту. На її основі визначено загальні вимоги до веб-застосунку, зокрема підтримку роботи з користувачами, програмними продуктами, характеристиками якості, критеріями, експертними оцінками та підсумковими результатами.

У другому розділі сформовано функціональні та нефункціональні вимоги до веб-застосунку. Визначено три основні ролі користувачів: адміністратора, експерта та користувача результатів. Для кожної ролі побудовано діаграми варіантів використання, що дозволило уточнити доступні функції системи. Також розроблено діаграми потоків даних, клієнт-серверну архітектуру веб-застосунку та структуру бази даних. Спроєктована база даних забезпечує збереження інформації про користувачів, ролі, програмні продукти, критерії, експертні оцінки, результати та історію оцінювання.

У третьому розділі описано реалізацію веб-застосунку з використанням технологій ASP.NET MVC, мови програмування C# та бази даних MS SQL Server. Реалізовано структуру класів предметної області, репозиторії для доступу до даних, контролери серверної частини, алгоритм розрахунку результатів експертного оцінювання та користувацькі інтерфейси для різних ролей.

Розроблений алгоритм дозволяє опрацьовувати експертні оцінки, враховувати вагові коефіцієнти критеріїв, обчислювати загальний показник якості

та визначати рівень якості програмного продукту. Додатково передбачено формування результатів за окремими характеристиками, що дає змогу виявляти сильні та слабкі сторони програмного забезпечення.

Проведене тестування підтвердило коректність роботи основних функціональних можливостей веб-застосунку. Перевірено авторизацію користувачів, розмежування доступу за ролями, додавання програмних продуктів, формування критеріїв, призначення експертів, введення оцінок, розрахунок підсумкових результатів і перегляд результатів оцінювання. Отримані результати тестування показали, що система працює відповідно до визначених вимог.

Отже, мету кваліфікаційної роботи досягнуто. Розроблений веб-застосунок забезпечує організацію процесу експертного оцінювання якості програмного забезпечення, автоматизує обробку оцінок, підтримує рольовий доступ користувачів і надає результати у зручному для аналізу вигляді. Подальший розвиток системи може передбачати додавання експорту звітів, розширення набору аналітичних діаграм, підтримку різних шкал оцінювання та інтеграцію з іншими програмними засобами управління якістю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering. 10th ed. Boston: Pearson Education. 2016. 816 p.

2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York: McGraw-Hill Education. 2020. 672 p.

3. Bourque P., Fairley R. E. Guide to the Software Engineering Body of Knowledge: SWEBOK Guide, Version 3.0. Los Alamitos: IEEE Computer Society. 2014. 335 p.

4. ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model.

5. ISO/IEC 25023:2016 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.

6. ISO/IEC/IEEE 12207:2017 Systems and software engineering — Software life cycle processes.

7. ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering.

8. IEEE Std 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation.

9. Miguel J. P., Mauricio D., Rodriguez G. A Review of Software Quality Models for the Evaluation of Software Products. International Journal of Software Engineering & Applications. 2014. Vol. 5, No. 6. PP. 31–53.

10. Microsoft. Entity Framework Documentation. Microsoft Learn. 2024.

11. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення: навчальний посібник. Київ: Центр навчальної літератури. 2021. 204 с.

12. Левус Є. В., Мельник Н. Б. Вступ до інженерії програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки. 2018. 248 с.

13. Петрик М. Р. Лабораторний практикум з розділу «Шаблони проєктування» дисципліни «Архітектура та проектування програмного

забезпечення»: навчальний посібник. Тернопіль: ТНТУ імені Івана Пулюя. 2016. 36 с.

14. Pastukh O., Yatsyshyn V., Kukharska V., Palamar A., Kulikov S. Method and tool of detecting software architecture patterns in the process of computer systems development. CEUR Workshop Proceedings. 2024. Vol. 3896. PP. 12–24.

- Microsoft. Overview of ASP.NET Core MVC. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-10.0 (дата звернення: 08.06.2026 р.).

15. Microsoft. Get started with ASP.NET Core MVC. URL: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-10.0 (дата звернення: 08.06.2026 р.).

16. Microsoft. Entity Framework documentation. URL: https://learn.microsoft.com/en-us/ef/ (дата звернення: 09.06.2026 р.).

17. Microsoft. Overview of Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 09.06.2026 р.).

18. Microsoft. SQL Server technical documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17 (дата звернення: 10.06.2026 р.).

19. Microsoft. SQL Server documentation. URL: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver17 (дата звернення: 11.06.2026 р.).

20. Microsoft. C# documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 12.06.2026 р.).

21. Microsoft. .NET documentation. URL: https://learn.microsoft.com/en-us/dotnet/ (дата звернення: 13.06.2026 р.).

22. OWASP Foundation. Application Security Verification Standard. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 14.06.2026 р.).

23. ISO. ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. URL: https://www.iso.org/standard/78176.html (дата звернення: 08.06.2026 р.).

24. OWASP Foundation. OWASP Top 10:2025. URL: https://owasp.org/Top10/2025/en/ (дата звернення: 09.06.2026 р.).

25. OWASP Foundation. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/ (дата звернення: 10.06.2026 р.).

26. NIST. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. URL: https://csrc.nist.gov/pubs/sp/800/218/final (дата звернення: 11.06.2026 р.).

27. Microsoft. Authentication and authorization in ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/ (дата звернення: 12.06.2026 р.).

28. Microsoft. Authorization in ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authorization/introduction (дата звернення: 13.06.2026 р.).

29. Microsoft. Test ASP.NET Core MVC apps. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/testing (дата звернення: 14.06.2026 р.).

30. Microsoft. Razor syntax reference for ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor (дата звернення: 15.06.2026 р.).

31. Bootstrap Team. Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 15.06.2026 р.).

32. Бедрій Я. Основи охорони праці користувачів персональних комп'ютерів: навчальний посібник для студентів ВНЗ та інженерів-практиків. Навчальна книга-Богдан. 2014. 144 с.

33. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Київ. 2018.

34. Методичні вказівки для написання розділу «Безпека життєдіяльності, основи охорони праці» в кваліфікаційних роботах здобувачів освітнього рівня «бакалавр». Для студентів всіх форм навчання, рівень вищої освіти перший (бакалаврський) / укл.: О.Я. Гурик, І.Б. Окіпний. Тернопіль: ТНТУ імені Івана Пулюя, 2021. 20 с.

ДОДАТКИ

Додаток А – Скрипт генерації бази даних

```
CREATE DATABASE ExpertQualityAssessmentDB;
GO

USE ExpertQualityAssessmentDB;
GO

CREATE TABLE Roles (
    role_id INT IDENTITY(1,1) PRIMARY KEY,
    role_name NVARCHAR(50) NOT NULL UNIQUE,
    description NVARCHAR(255) NULL
);
GO

CREATE TABLE Users (
    user_id INT IDENTITY(1,1) PRIMARY KEY,
    role_id INT NOT NULL,
    full_name NVARCHAR(150) NOT NULL,
    email NVARCHAR(150) NOT NULL UNIQUE,
    password_hash NVARCHAR(255) NOT NULL,
    is_active BIT NOT NULL DEFAULT 1,
    created_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),

    CONSTRAINT FK_Users_Roles
        FOREIGN KEY (role_id) REFERENCES Roles(role_id)
);
GO

CREATE TABLE SoftwareProducts (
    product_id INT IDENTITY(1,1) PRIMARY KEY,
    created_by INT NOT NULL,
    name NVARCHAR(200) NOT NULL,
    description NVARCHAR(MAX) NULL,
    status NVARCHAR(50) NOT NULL DEFAULT N'створено',
```

```
    created_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),

    CONSTRAINT FK_SoftwareProducts_Users
        FOREIGN KEY (created_by) REFERENCES Users(user_id)
);
GO

CREATE TABLE QualityCharacteristics (
    characteristic_id INT IDENTITY(1,1) PRIMARY KEY,
    name NVARCHAR(150) NOT NULL UNIQUE,
    description NVARCHAR(MAX) NULL
);
GO

CREATE TABLE EvaluationCriteria (
    criterion_id INT IDENTITY(1,1) PRIMARY KEY,
    characteristic_id INT NOT NULL,
    name NVARCHAR(200) NOT NULL,
    description NVARCHAR(MAX) NULL,
    weight DECIMAL(5,2) NOT NULL DEFAULT 1.00,

    CONSTRAINT FK_EvaluationCriteria_QualityCharacteristics
        FOREIGN KEY (characteristic_id)
        REFERENCES QualityCharacteristics(characteristic_id),

    CONSTRAINT CHK_EvaluationCriteria_Weight
        CHECK (weight > 0)
);
GO

CREATE TABLE ExpertAssignments (
    assignment_id INT IDENTITY(1,1) PRIMARY KEY,
    product_id INT NOT NULL,
    expert_id INT NOT NULL,
    assigned_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),
    status NVARCHAR(50) NOT NULL DEFAULT N'призначено',
```

```
    CONSTRAINT FK_ExpertAssignments_Products
        FOREIGN KEY (product_id)
        REFERENCES SoftwareProducts(product_id),

    CONSTRAINT FK_ExpertAssignments_Experts
        FOREIGN KEY (expert_id)
        REFERENCES Users(user_id),

    CONSTRAINT UQ_ExpertAssignments_Product_Expert
        UNIQUE (product_id, expert_id)
);
GO

CREATE TABLE ExpertScores (
    score_id INT IDENTITY(1,1) PRIMARY KEY,
    product_id INT NOT NULL,
    criterion_id INT NOT NULL,
    expert_id INT NOT NULL,
    score_value INT NOT NULL,
    comment NVARCHAR(MAX) NULL,
    created_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),

    CONSTRAINT FK_ExpertScores_Products
        FOREIGN KEY (product_id)
        REFERENCES SoftwareProducts(product_id),

    CONSTRAINT FK_ExpertScores_Criteria
        FOREIGN KEY (criterion_id)
        REFERENCES EvaluationCriteria(criterion_id),

    CONSTRAINT FK_ExpertScores_Experts
        FOREIGN KEY (expert_id)
        REFERENCES Users(user_id),

    CONSTRAINT CHK_ExpertScores_Value
```

```
        CHECK (score_value BETWEEN 1 AND 5),

    CONSTRAINT UQ_ExpertScores_Product_Criterion_Expert
        UNIQUE (product_id, criterion_id, expert_id)
);
GO

CREATE TABLE EvaluationResults (
    result_id INT IDENTITY(1,1) PRIMARY KEY,
    product_id INT NOT NULL,
    total_score DECIMAL(5,2) NOT NULL,
    quality_level NVARCHAR(100) NOT NULL,
    calculated_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),

    CONSTRAINT FK_EvaluationResults_Products
        FOREIGN KEY (product_id)
        REFERENCES SoftwareProducts(product_id)
);
GO

CREATE TABLE EvaluationHistory (
    history_id INT IDENTITY(1,1) PRIMARY KEY,
    product_id INT NOT NULL,
    result_id INT NULL,
    action_description NVARCHAR(MAX) NOT NULL,
    created_at DATETIME2 NOT NULL DEFAULT SYSDATETIME(),

    CONSTRAINT FK_EvaluationHistory_Products
        FOREIGN KEY (product_id)
        REFERENCES SoftwareProducts(product_id),

    CONSTRAINT FK_EvaluationHistory_Results
        FOREIGN KEY (result_id)
        REFERENCES EvaluationResults(result_id)
);
GO
```