Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Бакалавр
(назва освітнього ступеня)

на тему: Розробка автоматизованої системи порівняльного аналізу програмних рішень на основі техніко-економічних показників

Виконав(ла): студент(ка) 4 курсу, групи СПс-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Чорнописький Б.Т.
(підпис) (прізвище та ініціали)

Керівник Цуприк Г.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *програмної інженерії*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« 6 » 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 Чорнопиському Богдану Тарасовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розробка автоматизованої системи порівняльного аналізу програмних рішень на основі техніко-економічних показників»*

Керівник роботи Цуприк Галина Богданівна к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *6* » *квітня* 2026 року № *4/9-172*

2. Термін подання студентом завершеної роботи *22.06.2026*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз вимог до програмної системи
2. Проєктування та розробка програмної системи
3. Тестування, впровадження та підтримка
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Слайди презентації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання *06.04.2026р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
	Розробка технічного завдання	*06.04 – 12.04*	
	Робота над першим розділом «Аналіз вимог до програмної системи»	*13.04 – 24.04*	
	Робота над другим розділом «Проєктування та розробка програмної системи»	*25.04 – 04.05*	
	Робота над третім розділом «Тестування, впровадження та підтримка»	*05.05 – 18.05*	
	Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»	*19.05 – 24.05*	
	Оформлення пояснювальної записки та графічного матеріалу	*25.05 – 07.06*	
	Перевірка на академічний плагіат, перевірка керівником та консультантами	*08.06 – 14.06*	
	Попередній захист кваліфікаційної роботи бакалавра	*15.06 – 21.06*	
	Захист кваліфікаційної роботи бакалавра		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка автоматизованої системи порівняльного аналізу програмних рішень на основі техніко-економічних показників // Кваліфікаційна робота освітнього рівня «Бакалавр» // Чорнописький Богдан Тарасович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПс-41 // Тернопіль, 2026 // С. – 63, рис. – 15, табл. – 3, додат. – 2, бібліогр. – 44.

Ключові слова: порівняльний аналіз, програмне забезпечення, автоматизована система, вебдодаток, критерії оцінювання.

Кваліфікаційна робота присвячена дослідженню та розробці автоматизованої системи порівняльного аналізу програмних рішень на основі техніко економічних показників, для оптимізації процесу вибору найкращих цифрових рішень.

У першому розділі проведено аналіз предметної області, розглянуто існуючі аналоги систем порівняння та визначено основні критерії оцінювання програмних продуктів.

У другому розділі досліджено вимоги до системи, подано архітектуру вебдодатка, обґрунтовано вибір технологій розробки, спроєктовано структуру бази даних, описано реалізацію системи, проаналізовано її основні модулі, розроблено користувацький інтерфейс.

У третьому розділі описано тестування працездатності і функціоналу розробки, описано верифікацію та вимоги системи до апаратного забезпечення.

У четвертому розділі розглядаються питання забезпечення безпеки життєдіяльності та охорони праці.

Об'єкт дослідження: процес автоматизованого порівняльного аналізу та вибору програмного забезпечення.

Предмет дослідження: методи, технології та програмні засоби розробки інформаційної системи для попарного порівняння програмних продуктів.

ABSTRACT

Development of an Automated System for the Comparative Analysis of Software Solutions Based on Technical and Economic Indicators // Bachelor's Thesis // Bohdan Tarasovych Chornopyskyi // Ivan Pul'uj Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SPs-41 // Ternopil, 2026 // P. – 63, fig. – 15, tab. – 3, append. – 2, bibliogr. – 33.

Keywords: comparative analysis, software, automated system, web application, evaluation criteria.

This thesis is devoted to the research and development of an automated system for the comparative analysis of software solutions based on technical and economic indicators, with the aim of optimizing the process of selecting the best digital solutions.

The first chapter analyzes the subject area, examines existing comparable systems, and defines the main criteria for evaluating software products.

The second chapter examines the system requirements, presents the web application architecture, justifies the choice of development technologies, designs the database structure, describes the system implementation, analyzes its main modules, and develops the user interface.

The third chapter describes the testing of the system's performance and functionality, as well as the verification process and the system's hardware requirements.

The fourth section discusses issues of ensuring life safety and occupational health and safety.

Research object: the process of automated comparative analysis and software selection.

Research subject: methods, technologies, and software tools for developing an information system for pairwise comparison of software products.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД – база даних, впорядкована сукупність структурованих даних.

Клієнт-сервер – архітектура комп'ютерної мережі, в якій навантаження розподіляється між постачальниками послуг та замовниками.

Тест-кейс – набір умов, кроків та очікуваних результатів, розроблений для перевірки виконання конкретної функції програмного забезпечення.

API – (Application Programming Interface) прикладний програмний інтерфейс; набір визначень, протоколів та інструментів для створення програмного забезпечення та взаємодії між різними додатками.

CRUD – (Create, Read, Update, Delete) базові операції керування даними: створення, читання, оновлення та вилучення.

CSS – (Cascading Style Sheets) каскадні таблиці стилів спеціальна мова, що використовується для опису зовнішнього вигляду та оформлення вебсторінок.

DOM – (Document Object Model) об’єктна модель документа; незалежний від платформи та мови програмний інтерфейс, який дозволяє програмам та скриптам динамічно отримувати доступ до вмісту HTML-документів та оновлювати його.

ES6+ – (ECMAScript 6+) сучасна специфікація та стандарт мови програмування JavaScript, що визначає її синтаксис та базові можливості.

HTML – (HyperText Markup Language) мова розмітки гіпертексту;

JSON – (JavaScript Object Notation) Текстовий формат обміну даними, заснований на JavaScript, який легко читається як людиною, так і комп'ютерними програмами.

MySQL – Популярна реляційна система управління базами даних з відкритим вихідним кодом.

PDO – (PHP Data Objects) легковажний та універсальний інтерфейс в мові PHP для безпечного доступу до різноманітних баз даних.

PHP – (Hypertext Preprocessor) скриптова мова програмування загального призначення, яка інтенсивно застосовується для розробки серверної частини вебдодатків.

ЗМІСТ

ВСТУП

В умовах стрімкого цифрового розвитку бізнес-процесів та постійного розширення ринку інформаційних технологій перед інформаційними-відділами, системними адміністраторами та керівниками компаній постає складна проблема вибору оптимальних програмних рішень. На ринку існують десятки аналогів для кожної категорії ПЗ (офісні пакети, інструменти розробки, тощо), які суттєво відрізняються за вимогами до апаратного забезпечення та ціновою політикою. Традиційний "ручний" підхід до порівняння є неефективним, ресурсомістким та часто спирається на суб'єктивні оцінки або маркетингові матеріали розробників. Відповідно, виникає гостра потреба у створенні спеціалізованого програмного інструментарію, здатного автоматизувати процес збирання даних та проведення багатокритеріального порівняння технічних та економічних показників. Це дозволить приймати обґрунтовані рішення при проєктуванні та розширенні технологічної інфраструктури, що й зумовлює актуальність даної кваліфікаційної роботи.

Метою кваліфікаційної роботи є розробка веб-орієнтованої автоматизованої системи для проведення порівняльного аналізу існуючих програмних рішень на основі їх технічних вимог та економічних показників з метою підтримки прийняття рішень щодо вибору програмного забезпечення.

Для досягнення поставленої мети було сформульовано та вирішено такі задачі:

- провести аналіз предметної області та існуючих підходів до оцінювання програмних продуктів;
- спроєктувати архітектуру клієнт-серверного веб-додатка та розробити концептуальну схему реляційної бази даних;
- розробити серверну частину системи з використанням мови PHP та технології PDO для безпечної обробки та зберігання інформації;
- спроєктувати та реалізувати адаптивний інтерфейс користувача з використанням технологій HTML, CSS та JavaScript;

– розробити алгоритми порівняння та реалізувати інтерактивну візуалізацію результатів аналізу;
– провести комплексне тестування та верифікацію розробленої програмної системи.;

Об’єкт дослідження – процес вибору та багатокритеріального порівняльного оцінювання програмного забезпечення.

Предмет дослідження – методи, алгоритми та програмні засоби автоматизації аналізу та порівняння техніко-економічних показників програмних рішень.

Наукова новизна одержаних результатів полягає у вдосконаленні підходу до оцінювання програмного забезпечення шляхом алгоритмічного поєднання інтерактивної візуалізації апаратних вимог із розрахунком економічних показників у єдиному веб-орієнтованому середовищі, що мінімізує вплив суб'єктивного фактору при прийнятті рішень.

Розроблена автоматизована система є готовим до використання програмним продуктом. Її впровадження дозволяє менеджерам скоротити час на проведення аудиту ринку ПЗ, уникнути ризиків несумісності програм з наявною комп'ютерною технікою та оптимізувати бюджетні витрати на закупівлю ліцензій. Програмний код має модульну архітектуру і може бути легко розширений новими категоріями та критеріями оцінки.

Основні положення та результати кваліфікаційної роботи опубліковано в матеріалах «Науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя» (м. Тернопіль, 2026 р.) та наведено в додатку А.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі проводиться комплексне дослідження предметної області, пов’язаної з процесами вибору та оцінювання програмного забезпечення для потреб сучасних підприємств та організацій. На основі аналізу наявних ринкових аналогів окреслено ключові проблеми та технологічні розриви, які зумовлюють необхідність автоматизації техніко-економічного аналізу ПЗЗ метою формалізації вимог до програмного продукту здійснюється ідентифікація майбутніх користувачів, а також моделювання їхньої взаємодії з системою шляхом визначення та детального опису ключових варіантів використання [1].

1.1 Аналіз предметної області

Сучасний етап розвитку інформаційних технологій характеризується надзвичайно високою інтенсивністю розробки та впровадження програмного забезпечення (ПЗ) у всі сфери людської діяльності [2]. Для вирішення будь-якої прикладної чи виробничої задачі на ринку, як правило, існує велика кількість альтернативних програмних продуктів від різних розробників. У зв'язку з цим перед IT-директорами, системними архітекторами, системними адміністраторами та керівниками підприємств постійно постає складне завдання вибору оптимального програмного рішення для корпоративної або індивідуальної інфраструктури [3].

Процес вибору ПЗ є комплексною багатокритеріальною задачею, яка вимагає одночасного врахування двох основних груп показників:

– Технічні критерії. Сюди відносяться мінімальні та рекомендовані системні вимоги програмного продукту: обсяг оперативної пам'яті, необхідне місце на жорсткому диску або твердотільному накопичувачі, архітектура та потужність процесора, сумісність з операційними системами тощо. Ігнорування цих параметрів може призвести до критичного зниження продуктивності або повної неможливості функціонування ПЗ на наявному

комп'ютерному парку компанії, що зумовлює додаткові незаплановані витрати на модернізацію техніки [3, 4].

– Економічні критерії. Ця група включає фінансові аспекти придбання та експлуатації ПЗ, такі як вартість базової ліцензії для індивідуального використання, вартість бізнес- або корпоративної ліцензії, модель ліцензування одноразова покупка чи періодична підписка та сукупна вартість володіння [5].

На сьогоднішній день у практиці управління інформаційною інфраструктурою виділяють кілька основних підходів до проведення порівняльного аналізу ПЗ:

Експертний підхід за допомогою табличних процесорів. За такого підходу відповідальний аналітик самостійно збирає дані з офіційних сайтів розробників, заносить їх у таблицю та намагається зіставити параметри. Основними недоліками цього методу є висока трудомісткість, значний ризик допущення людських помилок при копіюванні даних, швидке моральне застарівання зібраної інформації та відсутність динамічних засобів інтерактивної візуалізації для представлення результатів керівництву [6].

Використання спеціалізованих глобальних платформ-агрегаторів. Дані веб-ресурси містять величезні каталоги програмного забезпечення, розділені за категоріями, та надають базові інструменти порівняння «пліч-о-пліч». Проте аналіз цих платформ виявив низку суттєвих обмежень [6]. По-перше, вони орієнтовані переважно на суб'єктивні текстові відгуки користувачів. По-друге, вони практично не пропонують жорстких алгоритмічних інструментів для кількісного порівняння сухих технічних характеристик та чітких фінансових витрат під конкретну кількість ліцензій. По-третє, такі системи мають закриту архітектуру, що унеможливлює їх адаптацію або інтеграцію у внутрішні процеси конкретного підприємства [3, 7].

Таким чином, проведення аналізу предметної області свідчить про наявність явного технологічного розриву між потребою ринку в швидкому, об'єктивному техніко-економічному оцінюванні ПЗ та існуючими інструментами автоматизації цього процесу.

Вирішенням окресленої проблеми є розробка автономної, веб-орієнтованої автоматизованої системи, яка дозволить централізовано зберігати структуровані техніко-економічні характеристики програмних продуктів різних категорій у реляційній базі даних [8]. Використання клієнт-серверної архітектури на базі мови PHP, СКБД MySQL та сучасних JavaScript-бібліотек візуалізації забезпечить користувачам можливість миттєвого побудови порівняльних матриць та інтерактивних діаграм, що оптимізує та автоматизує процес прийняття управлінських рішень під час проєктування та розширення комп'ютерних мереж та технічної інфраструктур [2, 9].

1.2 Постановка завдання та цілей

На основі проведеного аналізу предметної області виявлено потребу у створенні спеціалізованого програмного інструментарію, який би дозволив автоматизувати процес порівняння різних програмних продуктів за їхніми ключовими параметрами [10, 11].

Головним завданням розробки є створення веб-орієнтованої автоматизованої системи порівняльного аналізу програмного забезпечення, яка забезпечить агрегацію, зберігання та інтерактивну візуалізацію техніко-економічних характеристик IT-рішень різних категорій.

Для успішного виконання поставленого завдання необхідно досягти таких цілей:

1. Створити єдину централізовану базу даних для зберігання актуальної інформації про програмні продукти (їхні назви, категорії, мінімальні та рекомендовані апаратні вимоги, моделі ліцензування та вартість).
2. Забезпечити зручний та інтуїтивно зрозумілий користувацький інтерфейс для швидкого вибору категорії та конкретних додатків для порівняння.
3. Реалізувати алгоритми паралельного зіставлення характеристик обраних програмних рішень.

4. Мінімізувати вплив суб'єктивного фактору під час вибору ПЗ, надаючи користувачеві виключно кількісні та підтверджені дані.

Розроблена система повинна мати клієнт-серверну архітектуру, працювати в усіх сучасних веб-браузерах без необхідності встановлення додаткового програмного забезпечення на комп'ютер клієнта та швидко обробляти запити завдяки оптимізованій взаємодії з базою даних. Досягнення цих цілей дозволить значно скоротити час на проведення IT-аудиту та прийняття обґрунтованих рішень щодо закупівлі програмного забезпечення.

1.3 Пошук акторів та варіантів використання

Проєктування будь-якої програмної системи вимагає чіткого розуміння того, хто саме буде з нею взаємодіяти та які функції повинні бути доступні кожній категорії користувачів під час експлуатації [12]. Для формалізації функціональних вимог до розроблюваної автоматизованої системи порівняльного аналізу програмного забезпечення доцільно використати методологію об’єктно-орієнтованого аналізу, зокрема побудову моделі прецедентів мовою уніфікованого моделювання UML [13]. Першим етапом проєктування такої моделі є ідентифікація акторів – зовнішніх сутностей (користувачів або суміжних систем), що безпосередньо взаємодіють із системою для досягнення певних бізнес-цілей [14].

Аналіз предметної області та формування цілей розробки дає можливість виділити двох основних акторів:

1. Користувач. Це основний споживач послуг веб-додатка, який шукає інформацію для прийняття рішення щодо вибору ПЗ. Цей актор взаємодіє з клієнтською частиною системи, не маючи прямого доступу до зміни даних. Його основна мета – отримати достовірний та наочний порівняльний аналіз.
2. Адміністратор системи. Це авторизований користувач, відповідальний за підтримку актуальності бази даних. Оскільки ринок IT-технологій

динамічно змінюється, системі необхідний суб'єкт, що здійснюватиме управління контентом.

Для визначених акторів було ідентифіковано наступний набір ключових варіантів використання, які описують поведінку системи у відповідь на їхні запити для Користувача:

– Перегляд категорій та ПЗ – ознайомлення з наявними напрямками аналізу.–Вибір програм для порівняння – інтерактивний підбір двох альтернативних рішень.
– Генерація аналітичного звіту – запуск порівняльного аналізу з виведенням зведених таблиць та 4 типів діаграм.

Для Адміністратора включає всі можливості користувача та адміністрування:

– Авторизація в системі – захищений вхід за паролем (admin_login.php).– Управління каталогом (CRUD) – додавання нових продуктів та перегляд усього списку з можливістю швидкої фільтрації.
– Модифікація даних ПЗ – редагування техніко-економічних полів у модальному вікні без перезавантаження сторінки.
– Контроль сесії (Вихід) – безпечне завершення роботи з адмін-панеллю.

Для кращого сприйняття варіантів використання та акторів було розроблено UML – діаграму варіантів використання на рисуноку1.1.

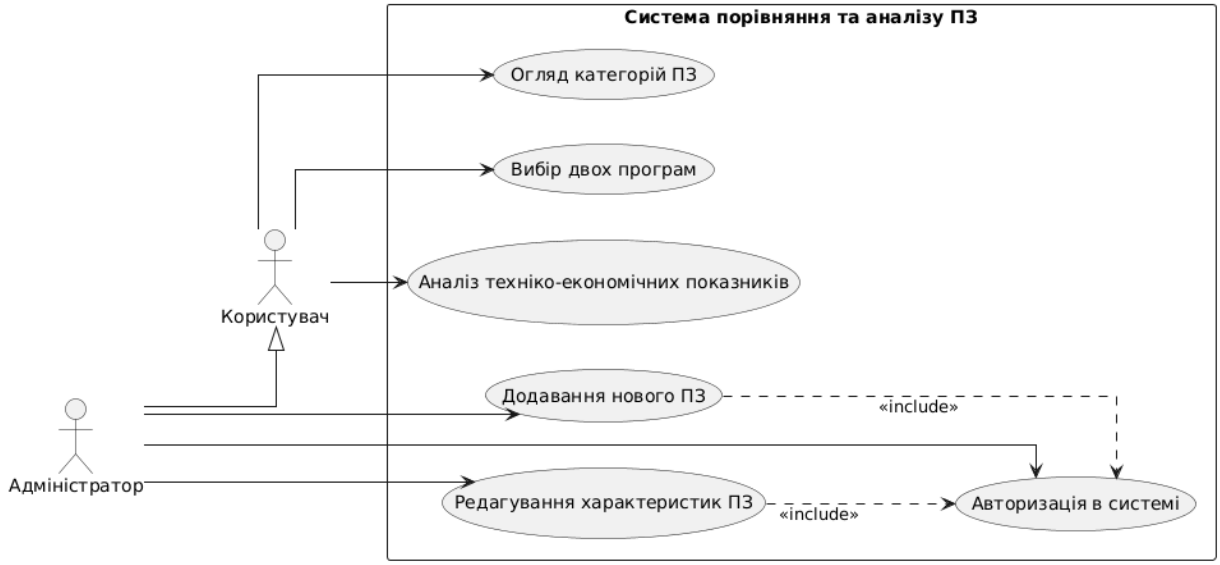

Рисунок 1.1 – Діаграма варіантів використання

Визначення цих акторів та прецедентів дозволяє окреслити чіткі межі розроблюваної системи та формує основу для подальшого детального проєктування логіки взаємодії користувача з веб-інтерфейсом та базою даних.

1.4 Опис ключових варіантів використання

Для детального аналізу внутрішньої динаміки системи та формалізації вимог до алгоритмів обробки даних необхідно виконати текстове та графічне проектування ключових прецедентів [14]. Згідно з методологією розробки складних програмних систем, звичайного переліку функцій недостатньо для побудови архітектури, тому кожен базовий прецедент розгортається у вигляді формалізованої специфікації, що описує взаємодію акторів із системою у часі [15].

Для візуалізації взаємодії об’єктів у межах архітектурного шаблону MVC, який реалізовано в додатку за допомогою розподілу логіки між клієнтським JavaScript та серверним PHP, доцільно застосувати діаграми послідовності [16]. Вони дозволяють чітко зафіксувати асинхронні AJAX-запити та життєвий цикл об'єктів під час сесії.

Специфікація прецеденту «Генерація аналітичного звіту» - цей прецедент є основним сценарієм для актора Користувач, що забезпечує виконання головної функції системи – візуалізації порівняльного аналізу на основі обраних альтернатив.

– Актор: Користувач
– Передумови: Веб-додаток успішно завантажено в браузері, база даних software ініціалізована та доступна для читання.
– Післяумови: Сформовано порівняльну таблицю, обчислено текстовий висновок та побудовано 4 інтерактивні діаграми Chart.js без перезавантаження веб-сторінки.

Основний сценарій виконання:

1. Користувач обирає категорію програмного забезпечення зі списку id="categorySelect".

2. Клієнтський скрипт script.js перехоплює подію change та надсилає асинхронний запит до get_software.
3. Серверний скрипт виконує безпечний запит до БД через PDO, отримує масив об'єктів і повертає його у форматі JSON [17].
4. Скрипт script.js динамічно розблоковує та заповнює елементи вибору software1Select та software2Select, відображаючи відповідні логотипи продуктів.
5. Користувач обирає дві різні програми (виконується прецедент «Вибір програм для порівняння») та натискає кнопку «Порівняти ПЗ».
6. script.js надсилає запит до compare.php.
7. compare.php вилучає з бази даних усі 16 техніко-економічних характеристик для обох програм за допомогою згрупованого запиту.
8. Сервер формує JSON-відповідь. Клієнтська частина обробляє дані, вираховує автоматизований аналітичний висновок (analysisSummary) та рендерить 4 графіки (апаратні вимоги, цінова політика, користувацький рейтинг, час освоєння).

Нижче на рисунку 1.2 наведено діаграму послідовності, яка відображає взаємодію компонентів під час генерації звіту.

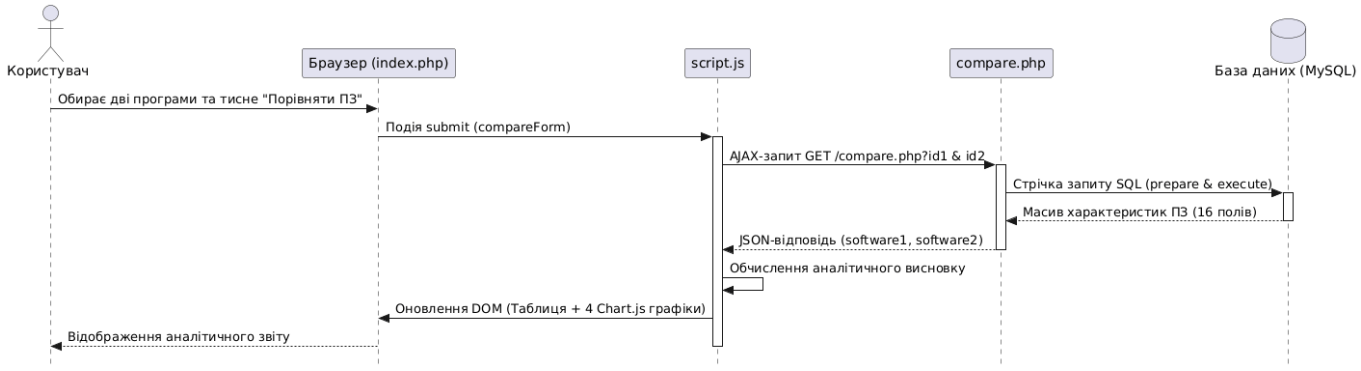

Рисунок 1.2 – Діаграма послідовності користувача

Специфікація прецеденту «Модифікація даних ПЗ» – цей прецедент описує привілейовану адміністративну функцію керування каталогом, реалізовану в

модулях admin.php та admin.js, що викликається з інтерфейсу управління каталогом (CRUD).

- Актор: Адміністратор системи.
- Передумови: Адміністратор успішно пройшов автентифікацію (виконано прецедент «Авторизація в системі»), сесія $_SESSION['admin_logged_in'] є активною, час бездіяльності не перевищує 2 годин [18].
- Післяумови: Запис у таблиці software оновлено, дані у вікні каталогу актуалізовано через динамічний DOM без перезавантаження всієї сторінки.

Основний сценарій виконання:

1. Адміністратор у таблиці каталогу знаходить потрібний програмний продукт і натискає кнопку «Редагувати».
2. Функція openEditModal(sw) в admin.js зчитує data-* атрибути обраного рядка (де зберігаються всі 16 параметрів об'єкта) та миттєво ініціалізує поля форми всередині модального вікна id=”editModal”.
3. Система блокує прокручування основної сторінки.
4. Адміністратор вносить зміни в будь-яке з 16 техніко-економічних полів форми і натискає кнопку «Зберегти зміни».
5. Дані форми відправляються на сервер через метод POST, де компонент admin.php виконує валідацію типів даних та оновлює інформацію в БД за допомогою SQL-оператора UPDATE з використанням Prepared Statements.

Для моделювання умов перевірки прав доступу, валідації сесії та процесу модифікації даних ПЗ використовується діаграма діяльності (рисунок 1.3). Вона найкраще демонструє розгалуження процесів [16].

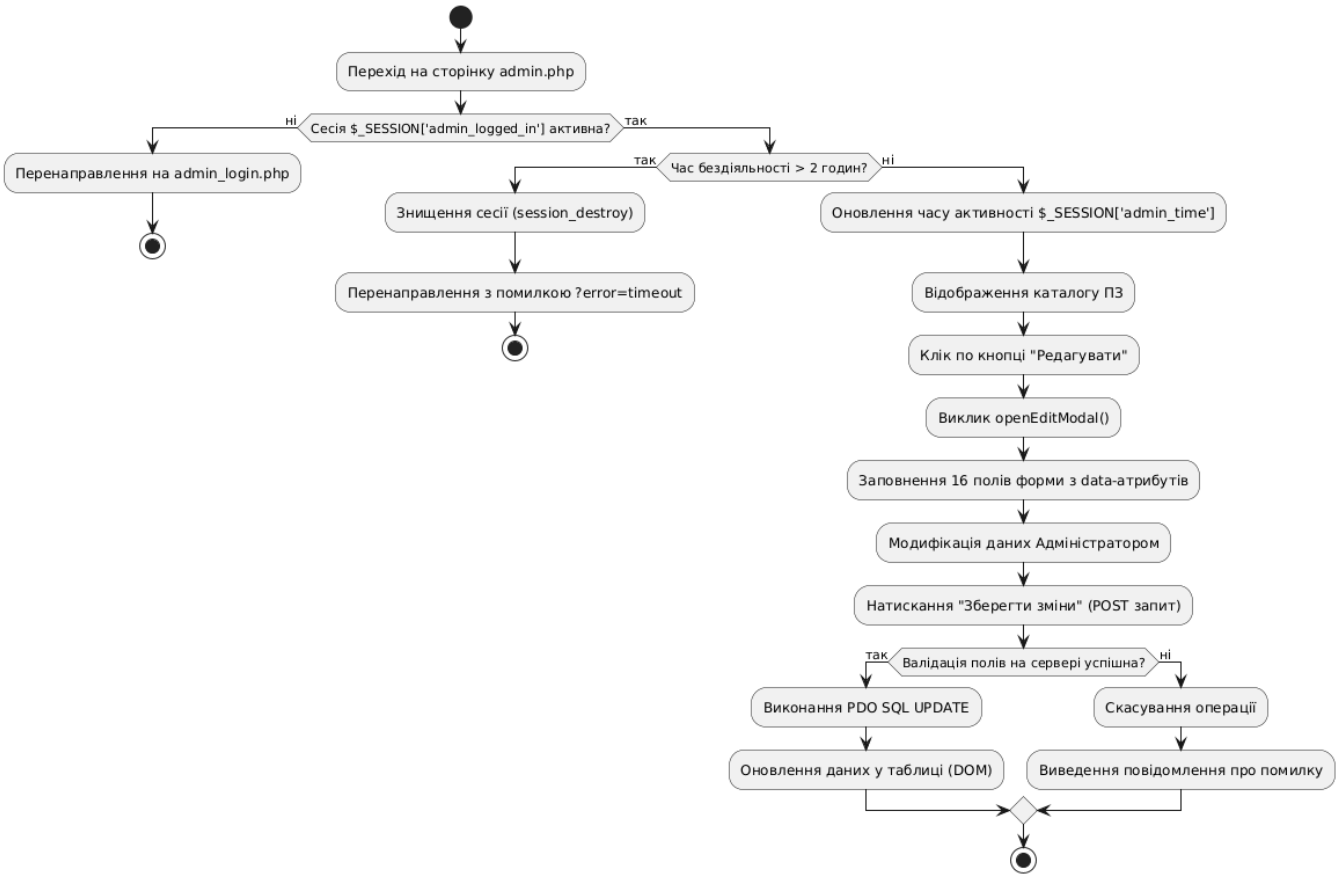

Рисунок 1.3 – Діаграма діяльності адміністратора

У результаті проведено комплексний аналіз предметної області та обґрунтовано доцільність розробки системи. Визначено мету, ключові завдання системи та ідентифіковано її основних акторів. Сформовано чіткі вимоги і побудовано UML-модель прецедентів, формалізовано логіку взаємодії із системою, та обробки техніко-економічних характеристик ПЗ, що відкриває простір для подальшого проєктування бази даних, архітектури та клієнт-серверних модулів додатку в наступному розділі.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

Розділ дозволяє детально розглянути практичні аспекти створення розробки. Описано вибір методології розробки, проєктування архітектури системи, побудову логічної та фізичної структури бази даних, а також процес програмної реалізації серверної та клієнтської частин вебдодатка. Крім того, наведено обґрунтування вибору базового технологічного стека та продемонстровано результати розробки графічного інтерфейсу користувача.

2.1 Вибір процесу розробки

Успішна та своєчасна реалізація будь-якого програмного забезпечення вимагає застосування адекватної моделі життєвого циклу розробки. Зважаючи на специфіку розроблюваної автоматизованої системи порівняльного аналізу програмного забезпечення, яка передбачає тісну інтеграцію клієнтських технологій із серверною бізнес-логікою та реляційною базою даних MySQL, класична каскадна модель Waterfall була визнана неефективною через її негнучкість до змін вимог на етапах реалізації та тестування [19].

Для реалізації даного проєкту було обрано ітеративно-інкрементну модель розробки, яка базується на принципах гнучкої методології [14]. Цей підхід дозволяє розбивати процес створення системи на невеликі керовані цикли (ітерації). У результаті кожної ітерації створюється працездатний і протестований фрагмент програми (інкремент), що поступово розширює загальний функціонал системи. Вибір цієї моделі обґрунтовується такими факторами:

- Зниження архітектурних ризиків: Дозволяє на ранніх етапах протестувати зв'язок бази даних із сервером ще до створення повноцінного графічного інтерфейсу.
- Гнучкість до змін: Надає можливість додавати нові техніко-економічні критерії для порівняння ПЗ без необхідності глобальної перебудови ядра системи.

– Безперервне тестування: Кожен розроблений компонент тестується ізольовано, що значно спрощує відлагодження асинхронних клієнт-серверних запитів.

У контексті розробки даної системи ЖЦ був розбитий на чотири основні ітерації:

1. Проєктування та розгортання бази даних: Формування структури таблиць software, categories та безпечного коннектора для роботи з об'єктами доступу до даних.
2. Розробка серверного API: Створення маршрутизаторів, які обробляють HTTP-запити, виконують SQL-вибірки та формують відповіді у стандартизованому форматі JSON [20].
3. Реалізація модуля адміністрування: Створення захищеної зони для управління каталогом, впровадження механізмів контролю сесій адміністратора та модальних вікон для CRUD-операцій.
4. Розробка клієнтського інтерфейсу та візуалізації: Створення головної сторінки, написання логіки обробки подій та інтеграція бібліотеки Chart.js для побудови аналітичних гістограм [21].

Використання ітеративно-інкрементного підходу дозволило ефективно розподілити час, забезпечити високу стабільність коду та створити зручний користувацький інтерфейс, який повною мірою відповідає всім заявленим у першому розділі вимогам.

2.2 Проєктування архітектури системи

Проєктування архітектури є одним із найважливіших етапів створення програмної системи, оскільки воно визначає загальну структуру додатка, взаємодію між його компонентами та забезпечує масштабованість і простоту підтримки коду. Належним чином побудована архітектурна модель дозволяє розмежувати зони відповідальності різних модулів, мінімізуючи зв'язність між ними. Для реалізації автоматизованої системи порівняльного аналізу було обрано класичний

трирівневий архітектурний патерн MVC (Model-View-Controller), який є галузевим стандартом для розробки сучасних веб додатків [22].

Суть патерну MVC полягає в розділенні логіки застосунку на три незалежні пласти, кожен з яких виконує суворо визначену функцію:

1. Модель (Model): Відповідає за бізнес-дані, правила їх збереження, валідації та безпосередню взаємодію з реляційним сховищем даних. Модель не володіє інформацією про те, як дані будуть відображатися користувачеві.
2. Представлення (View): Відповідає за формування графічного інтерфейсу користувача (UI) та візуалізацію даних, отриманих від моделі. Компоненти цього рівня лише відображають інформацію та передають дії користувача далі за ланцюжком.
3. Контролер (Controller): Виступає в ролі посередника між Моделлю та Представленням. Він приймає вхідні запити від клієнтської частини, трансформує їх у команди для Моделі, отримує результат та передає його назад у Представлення для фінального рендерингу [23].

Взаємодія між компонентами в межах розробленої системи відбувається за асинхронним принципом. Клієнтська частина ініціює запит до сервера, контролер перехоплює цей запит, звертається до сховища через інтерфейс доступу до даних, отримує структуровану інформацію, перетворює її на універсальний текстовий формат обміну даними та повертає клієнту для динамічного оновлення інтерфейсу без перезавантаження сторінки.

У структурі даного програмного забезпечення архітектурні шари розподілені між файлами проєкту таким чином:

- Рівень Моделі (Model Layer):
  - Модуль конфігурації та з'єднання зі сховищем даних – забезпечує безпечне підключення до бази даних.
  - Реляційна база даних – безпосередньо таблиці сховища, що зберігають структуровану інформацію про категорії, програмні продукти, та їх критерії.

- Рівень Контролера (Controller Layer):
  - Серверний маршрутизатор вибірки даних – приймає вхідні параметри, виконує безпечні SQL-запити до моделі та повертає відфільтрований перелік програмних продуктів.
  - Серверний модуль порівняльного аналізу – аналізує ідентифікатори обраних об'єктів, здійснює повну вибірку характеристик і формує масив даних для аналізу.
  - Підсистема автентифікації та контролю доступу – група серверних скриптів які керують сесіями адміністратора.
- Рівень Представлення (View Layer):
  - Основний користувацький інтерфейс – каркас головної сторінки додатка, що містить форми вибору та блоки для виведення результатів.
  - Адміністративний інтерфейс – панель управління каталогом, що містить таблиці даних, форми авторизації та вікна для модифікації записів.
  - Модулі стилізації – зовнішні каскадні таблиці стилів, які забезпечують адаптивність, колірну схему та дизайн інтерфейсу.
  - Клієнтські сценарії динамічної логіки — сценарії, що виконуються на стороні веббраузера. [24].

Для формалізації та наочного відображення архітектурної структури системи, зв’язків між її модулями та розподілу за шарами MVC було розроблено UML-діаграму компонентів [25]. Ця діаграма демонструє організацію фізичних компонентів коду та залежності між ними. Графічне представлення розробленої архітектурної схеми та взаємозв'язків між компонентами наведено на рисунку 2.1.

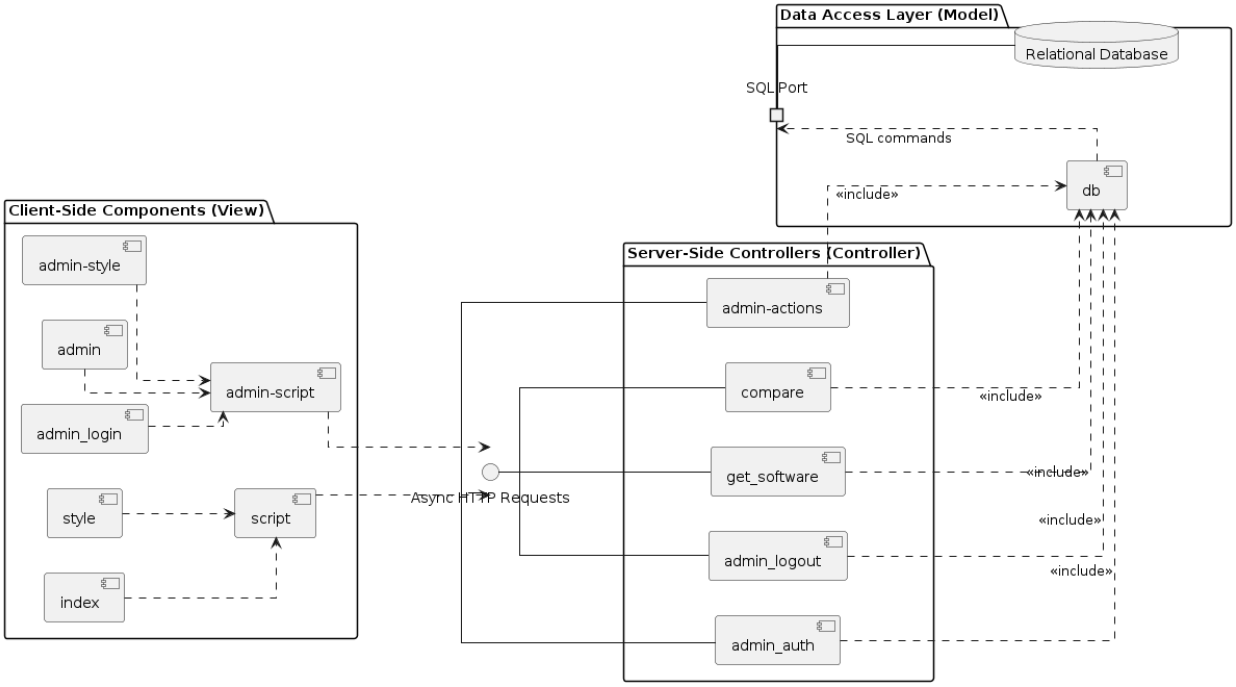

Рисунок 2.1 – Діаграма компонентів розробки

На діаграмі компонентів виділено три основні підсистеми.

Пакет «Client-Side Components (View)»: містить компоненти користувацького інтерфейсу, адміністративного інтерфейсу та клієнтських сценаріїв. Компоненти інтерфейсу залежать від клієнтських сценаріїв, які через інтерфейс асинхронних HTTP-запитів взаємодіють із сервером.

Пакет «Server-Side Controllers (Controller)»: об'єднує модулі обробки бізнес-логіки. Компоненти авторизації та обробки даних приймають запити від клієнта. Усі вони мають дескриптор залежності від компонента підключення до бази даних.

Пакет «Data Access Layer (Model)»: містить модуль конфігурації з'єднання та безпосередньо вузол реляційної бази даних, який через стандартизований порт обробляє SQL-команди, що надходять від контролерів.

2.3 Побудова схеми бази даних

Надійне та ефективне функціонування розробки безпосередньо залежить від раціонального проєктування її інформаційного забезпечення. На цьому етапі здійснюється фізична реалізація сховища даних за допомогою обраної системи

керування базами даних. Процес розробки структури даних охоплює побудову ER-діаграми та її подальшу трансформацію у логічні та фізичні схеми реляційних таблиць у середовищі реляційної СУБД MySQL за допомогою інтерактивного інструменту адміністрування phpMyAdmin [26].

Для забезпечення базового функціоналу порівняльного аналізу було виділено дві стрижневі сутності: "Категорія ПЗ" (categories) та "Програмне забезпечення" (software). Між цими сутностями встановлено ідентифікуючий зв’язок типу "один-до-багатьох", оскільки кожна категорія може містити безліч програмних продуктів, але один продукт не може належати до кількох категорій одночасно. Зв'язок реалізується шляхом міграції первинного ключа таблиці категорій у таблицю програмного забезпечення як зовнішнього ключа [27].

Графічне представлення концептуальної моделі даних, що відображає взаємозв'язки сутностей предметної області та їхні ключові атрибути, наведено у вигляді ER-діаграми на рисунку 2.2.

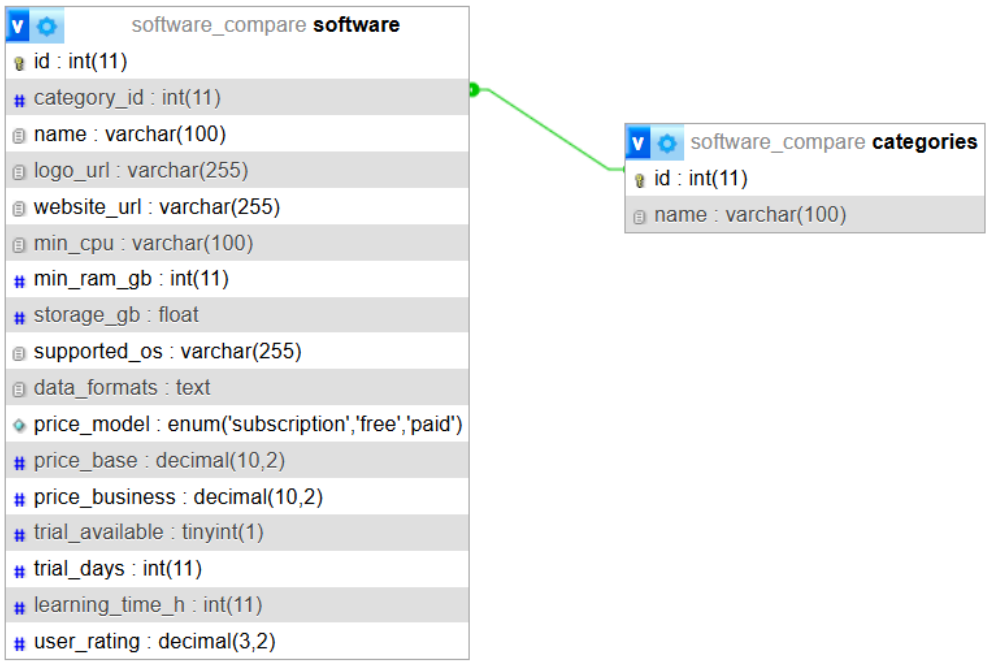

Рисунок 2.2 – ER-Діаграма бази даних

Логічну структуру першої таблиці – categories, яка слугує довідником категорій систем, наведено у таблиці 2.1.

Таблиця 2.1 – Структура таблиці категорій ПЗ

Назва поля	**Тип даних MySQL**	**Ключ / Індекс**	**Опис поля**
Id	INT (11)	Primary Key	Унікальний ідентифікатор категорії
Name	VARCHAR (255)	-	Назва категорії ПЗ

Таблиця categories є максимально компактною та нормалізованою. Поле id має прапорець AUTO_INCREMENT, що забезпечує автоматичну генерацію унікальних цілих чисел при додаванні нових записів адміністратором через інтерфейс phpMyAdmin.

Таблиця 2.2 – Структура таблиці порівняльних характеристик ПЗ

Назва поля	**Тип даних MySQL**	**Ключ / Індекс**	**Опис поля**
id	INT (11)	Primary Key	Унікальний ідентифікатор програмного продукту
name	VARCHAR (255)	-	Комерційна назва ПЗ
category_id	INT (11)	Foreign Key	Ідентифікатор категорії (зв'язок з categories.id)
logo_url	VARCHAR (255)	-	Шлях до файлу логотипа або URL-адреса зображення
website_url	VARCHAR (255)	-	Посилання на офіційний вебсайт розробника
min_cpu	VARCHAR (255)	-	Мінімальні вимоги до процесора (CPU)
min_ram_gb	INT (11)	-	Мінімальний обсяг оперативної пам'яті (у Гб)
storage_gb	INT (11)	-	Необхідний вільний простір на накопичувачі (у Гб)
supported_os	VARCHAR (255)	-	Перелік підтримуваних операційних систем
data_formats	VARCHAR (255)	-	Формати файлів, з якими здатне працювати ПЗ
price_model	VARCHAR (50)	-	Модель поширення
price_base	DECIMAL (10, 2)	-	Базова вартість індивідуальної ліцензії ($)
price_business	DECIMAL (10, 2)	-	Вартість бізнес- або корпоративної ліцензії ($)
trial_available	TINYINT (1)	-	Прапорець наявності пробного періоду (0 – ні, 1 – так)
trial_days	INT (11)	-	Тривалість безкоштовного періоду (в днях)
learning_time_h	INT (11)	-	Середній час на освоєння базового функціоналу (год)
user_rating	DECIMAL (3, 2)	-	Інтегральний рейтинг користувачів (від 1.00 до 5.00)

Для таблиці software дані було сформовано і нормалізовано за наступними критеріями:

- Тип INT використано для полів, що оперують виключно цілими дискретними значеннями.
- Тип VARCHAR із різною довжиною виділено під текстові описи, конфігурації заліза та рядки уніфікованих покажчиків ресурсів (URL), що оптимізує використання пам'яті на сервері MySQL.
- Для фінансових показників (price_base, price_business) та оцінок користувачів застосовано точний тип DECIMAL замість типів із плаваючою комою (FLOAT/DOUBLE). Це запобігає виникнення помилок округлення під час виконання аналітичних розрахунків у системі [28].
- Логічний прапорець trial_available реалізовано за допомогою типу TINYINT(1), де значення 1 інтерпретується системою як істина, а 0 – як хибність.

Для забезпечення цілісності даних на рівні СУБД для поля category_id налаштовано обмеження зовнішнього ключа FOREIGN KEY із правилами ON DELETE RESTRICT та ON UPDATE CASCADE. Це гарантує, що адміністратор не зможе випадково видалити з phpMyAdmin категорію, якщо до неї прив'язані діючі програмні продукти, запобігаючи появі "осиротілих" записів у системі.

2.4 Побудова UML-діаграм класів

Для формалізації структури автоматизованої системи, визначення внутрішнього складу її модулів та логічних взаємозв'язків між ними розроблено UML-діаграму класів. Ця діаграма дозволяє зафіксувати архітектурні шаблони, типи даних, зони відповідальності окремих компонентів, а також правила їхньої взаємодії [29].Структура системи була спроєктована відповідно до обраного патерна MVC, розподіливши класи на три функціональні групи: класи керування

даними та сховищем, класи обробки бізнес-логіки та маршрутизації запитів та класи забезпечення взаємодії з користувачем через графічний інтерфейс [30].

У межах даної архітектури виділено такі основні класи:

1. Database – клас конфігурації та з’єднання:
   - –Атрибути: host, db, user, pass, charset, pdo.
   - Методи: connect– ініціалізує захищене з'єднання з базою даних MySQL, налаштовує параметри обробки виключних ситуацій та режимів вибірки асоціативних масивів; getConnection– повертає активний екземпляр об'єкта підключення.
2. SoftwareModel – клас моделі даних програмних продуктів:
   - –Атрибути: id, name, categoryId, logoUrl, websiteUrl, minCpu, minRamGb, storageGb, supportedOs, dataFormats, priceModel, priceBase, priceBusiness, trialAvailable, trialDays, learningTimeH, userRating.
   - Методи: getByCategory – здійснює вибірку скороченого масиву продуктів для фільтрації; getPairForComparison – вилучає повний спектр техніко-економічних характеристик для двох обраних об'єктів; insert()/update()/delete() – методи адміністративного керування записами каталогу.
3. CategoryModel – клас моделі категорій:
   - –Атрибути: id, name.
   - Методи: getAll– повертає повний перелік категорій для ініціалізації навігаційних форм.
4. SoftwareController – клас серверного контролера обробки запитів:
   - Методи: actionGetSoftware– перехоплює асинхронні HTTP-запити від клієнта, перевіряє вхідні параметри, звертається до SoftwareModel та трансформує результат в універсальний формат JSON; actionCompare– контролює процес вилучення даних для аналізу двох систем та передає структуровану JSON-відповідь на сторону клієнта.
5. AdminController – клас підсистеми адміністрування та безпеки:
   - –Атрибути: adminPassword, sessionTime.

- Методи: login – перевіряє автентифікаційні дані, ініціалізує захищену сесію адміністратора; logout – знищує поточну сесію та виконує перенаправлення користувача; checkSession — запобігає несанкціонованому доступу до панелі керування.

6. UserInterfaceManager – клас клієнтської логіки представлення:
   - Методи: loadCategories() – заповнює списки вибору; handleSelectionChange() – керує доступністю елементів інтерфейсу; fetchComparisonData() – ініціює асинхронний запит до серверного контролера; renderComparisonTable() – будує адаптивну матрицю характеристик; renderCharts(hardwareData, priceData, ratingData, learningData – ініціює роботу графічних компонентів Canvas для візуалізації аналітики.
7. AdminInterfaceManager – клас клієнтського адміністрування:
   - Методи: toggleSection(id) — керує відображенням блоків інтерфейсу панелі керування; filterTableByCategory() — виконує локальну фільтрацію записів; openEditModal(softwareObject) — активує модальне вікно та заповнює його поля даними обраного продукту для редагування; closeEditModal() — закриває форми.

Між даними класами було встановлено наступні типи відношень.

Асоціація: зв’язок між SoftwareModel та CategoryModel, що відображає приналежність програмних продуктів до певної категорії.

Контролери SoftwareController та AdminController мають зв'язок залежності з Database, оскільки класи використовують відповідні методи Database для отримання та маніпуляції даними.

Клієнтські класи UserInterfaceManagerта AdminInterfaceManager взаємодіють із серверними контролерами за допомогою протоколу передачі гіпертексту, надсилаючи запити та приймаючи об'єкти конфігурації JSON [31].

Графічне представлення даної об’єктно-орієнтованої структури системи, її класи з їхніми атрибутами, методами та їхніми зв’язками наведено в вигляді UML-діаграми класів на рисунку 2.3.

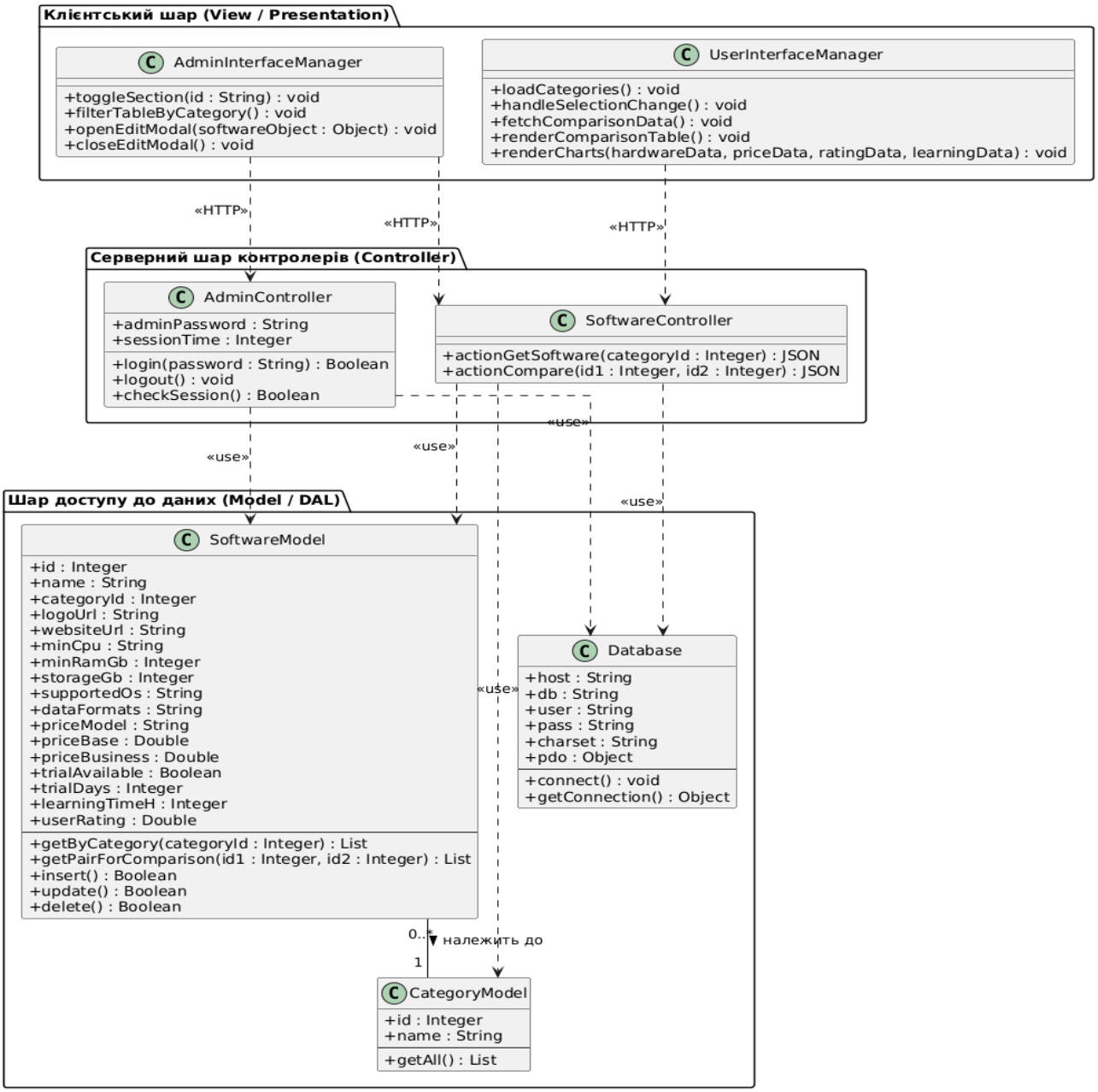

Рисунок 2.3 – Діаграма класів

Описаний розподіл системи забезпечує високий рівень модульності коду. Зміни у внутрішній логіці роботи моделей, або модифікація дизайну інтерфейсів у класах, не призведе до порушення роботи всієї системи що значно спрощує модернізацію та подальший супровід даної системи.

Додавання нового функціоналу відбувається шляхом розширення існуючих класів або додавання нових модулів за готовим шаблоном, без потреби переписувати базову архітектуру системи.

2.5 Вибір мови та середовища розробки

Ефективність роботи, рівень безпеки та можливість масштабування програмної системи на пряму залежить від раціонального вибору технології та середовища розробки. Для реалізації даної розробки було обрано клієнт-серверний підхід з застосуванням сучасних стандартів веброзробки. Вибір кожної технології ґрунтується на критеріях продуктивності, безпеки та доцільності для вирішення поставлених завдань та цілей розробки.

В якості серверної платформи для розробки бізнес-логіки було обрано мову програмування PHP. Це потужна сценарна мова загального призначення, яка історично й архітектурно оптимізована для створення динамічних вебдодатків. Основними чинниками цього вибору стали висока швидкість обробки вхідних HTTP-запитів, низькі накладні витрати пам'яті під час виконання скриптів та має відмінний рівень інтеграції з реляційними базами даних, що забезпечує стабільну роботу системи за умов інтенсивного обміну даними [32].

Для організації взаємодії між серверними контролерами та шаром моделей застосовано об'єктно-орієнтоване розширення PDO. На відміну від застарілих процедурних функцій, PDO є сучасним галузевим стандартом, який забезпечує кілька критично важливих переваг для архітектури проєкту:

- Гарантований захист від SQL-ін'єкцій: PDO на рівні драйвера реалізує механізм підготовлених виразів. Під час виконання запиту відбувається суворе розділення логічної структури SQL-команди та користувацьких даних. Будь-які вхідні параметри, що надходять від клієнта, автоматично екрануються й інтерпретуються СКБД виключно як значення, що повністю унеможливлює виконання впровадженого зловмисного коду.

- Абстракція доступу до даних: PDO надає уніфікований інтерфейс для роботи з різними СКБД. Якщо у процесі масштабування проєкту виникне потреба мігрувати з MySQL на більш потужну систему, розробнику не доведеться переписувати логіку запитів у класах моделей. Достатньо буде лише змінити параметри підключення – DataSource Name – у конфігураційному класі Database [33].
- Ефективне керування даними та помилками: Розширення дозволяє легко налаштувати режими обробки виключних ситуацій та автоматично повертати результати вибірки у вигляді асоціативних масивів. Це значно спрощує подальшу конвертацію даних у формат JSON для відправки на користувацький інтерфейс.

Розробка інтерактивної та динамічної складової користувацького інтерфейсу реалізована за допомогою мови програмування JavaScript із суворим дотриманням сучасних специфікацій стандарту ECMAScript. Інтеграція новітніх синтаксичних конструкцій – зокрема стрілкових функцій, концепції блокової області видимості та деструктуризації об'єктів – дозволила оптимізувати обсяг вихідного коду. Це забезпечило високу читабельність архітектури клієнтських скриптів, підвищило безпеку роботи зі змінними за рахунок усунення ризиків їхнього неконтрольованого перевизначення та суттєво спростило подальший супровід вебдодатка.

Для організації асинхронного обміну даними із сервером було прийнято рішення відмовитися від морально застарілої технології AJAX на базі низькорівневого об'єкта XMLHttpRequest, а також від використання додаткових громіздких бібліотек типу jQuery. Натомість основним інструментом взаємодії став сучасний вбудований інтерфейс Fetch API. Використання Fetch API зумовлене такими перевагами:

- Забезпечення коректної роботи в різних браузерах: Fetch API вбудований безпосередньо у всіх сучасних веббраузерах. Відмова від сторонніх бібліотек дозволила уникнути завантаження зайвих файлів, що значно знизило загальну вагу клієнтських скриптів, зменшило час завантаження сторінок та оптимізувало використання мережевого трафіку.

– Об'єктна модель на базі промісів: Інтерфейс функціонує на основі механізму Promise, що дозволяє будувати асинхронну логіку у лінійному та прозорому стилі. Такий підхід повністю ліквідує проблему каскадного вкладення функцій зворотного виклику, спрощує централізовану обробку виключних ситуацій і робить код максимально адаптивним для використання сучасних конструкцій [34].

Оскільки ключовим функціональним ядром розробленої системи є порівняльна аналітика програмного забезпечення, завдання наочного та інтуїтивно зрозумілого представлення метрик потребувало інтеграції надійного та високопродуктивної технології графічного представлення. Для вирішення цієї задачі було обрано спеціалізовану JavaScript-бібліотеку Chart.js. Вибір даної бібліотеки було зроблено спираючись на її наступні переваги:

– Висока обчислювальна продуктивність: Бібліотека здійснює відображення графічних елементів за допомогою низькорівневого API елемента <canvas>. На противагу SVG-орієнтованих аналогів, які створюють окремі вузли для кожного стовпчика чи лінії діаграм, Chart.js працює з єдиним растровим полем. Що в свою чергу мінімізує навантаження на центральний процесор клієнта, усуває затримки при маніпуляціях з DOM-деревом браузера та забезпечує плавну анімацію під час первинного відображення та оновлення аналітичних даних.
– Повна адаптивність та реактивність: Інтегровані графіки мають вбудовану підтримку динамічного масштабування. Рушій автоматично відстежує зміни розмірів батьківського вікна та коректно перераховує пропорції осей, координати точок, розміри шрифтів і маркерів підказок. Це гарантує високу ефективність інтерфейсу та коректне відображення аналітики як на моніторах робочих станцій адміністратора, так і на мобільних пристроях користувачів.
– Стабільна та звична інтеграція з JSON-структурами: Архітектура даних Chart.js повністю синхронізована зі структурами масивів JavaScript. Це дозволяє методу renderCharts() класу UserInterfaceManager приймати

масиви характеристик, які асинхронно надходять у форматі JSON від PHP-бекенду, та безпосередньо передавати їх у конфігураційні об'єкти діаграм без необхідності складної попередньої трансформації чи проміжного збору даних [35].

Як середовище основи для збереження, структурування та маніпуляції інформацією в системі було інтегровано реляційну систему керування базами даних MySQL. Вибір цієї технології для реалізації шару доступу до даних обумовлений її високою надійністю, відповідністю концепції ACID та оптимальним балансом обчислювальної продуктивності. Основними перевагами СКБД MySQL у межах розробленої архітектури є:

– Оптимізація під інтенсивні операції читання: За профілем навантаження системи, понад 90% усіх транзакцій до бази даних складають операції вибірки – наприклад, під час динамічного огляду категорій користувачами, вилучення техніко-економічних характеристик двох програм чи формування масивів аналітики. MySQL демонструє виняткову швидкодію та ефективність індексації саме при таких операціях, забезпечуючи мінімальний час відгуку сервера.
– Забезпечення суворої цілісності та зв'язності даних: Завдяки повній підтримці механізму зовнішніх ключів, система гарантує каскадну цілісність даних безпосередньо на рівні сховища. Це повністю виключає появу логічних збоїв чи помилок, що критично важливо для стабільної роботи бізнес-логіки моделей [36].

Для розгортання локального середовища розробки, проектування бази даних та тестування серверних скриптів було використано кросплатформний інструментальний стек XAMPP. Його інтеграція дозволила швидко налаштувати необхідне середовище розробки, мінімізуючи часові витрати на ручну конфігурацію та уникнення конфліктів версій окремих сервісів. До складу розгорнутого локального контуру увійшли такі ключові компоненти:

– Вебсервер Apache: Виступає в ролі локального HTTP-сервера, який керує потоками запитів. Він забезпечує низькорівневу маршрутизацію, обробляє

статичний контент, перехоплює асинхронні запити від клієнтського Fetch API, керує їхньою обробкою відповідним PHP-контролерам та повертає згенеровані структури JSON назад на сторону клієнта, гарантуючи високу стабільність з'єднання [37].

– Утиліта phpMyAdmin: Надає повнофункціональний веб-інтерфейс для візуального адміністрування бази даних, що нівелює потребу роботи через консольний клієнт. Вона ефективно використовувалася на етапах проектування реляційної схеми, швидкої генерації таблиць, ручного заповнення каталогу тестовими даними ПЗ, а також для оперативного моніторингу стану сховища, керування правами доступу та інструментального налагодження складних SQL-запитів [38].

Проєктування, безпосереднє написання програмного коду, створення програмної архітектури, інтеграція клієнтських скриптів та налаштування розмітки інтерфейсу здійснювалось в середовищі сучасного редактора вихідного коду Visual Studio Code від компанії Microsoft. Завдяки своїй гнучкості, підтримці багатьох платформ та високій швидкості роботи, цей інструмент став фактичним галузевим стандартом у сфері веброзробки. VS Code як середовище основного робочого простору було обрано через наступні переваги:

– Висока продуктивність та легковажність: На відміну від повнофункціональних, але досить важких інтегрованих середовищ розробки, таких як JetBrains PHPStorm або NetBeans, VS Code має модульну архітектуру. Він споживає мінімальний обсяг оперативної пам'яті та забезпечує миттєвий відгук інтерфейсу. Це дозволило розробнику підтримувати високу плавність роботи системи навіть за умов одночасного запуску локального сервера Apache та СКБД MySQL у межах стеку XAMPP.

– Потужна екосистема розширень: Базовий функціонал редактора легко масштабується під специфіку конкретного технологічного стеку. Зокрема, інтеграція спеціалізованого плагіна PHP Intelephense трансформує редактор у повноцінне середовище для бекенд-розробки, додаючи

інтелектуальне автодоповнення коду, контекстне підсвічування синтаксису, інструменти швидкого оптимізацію коду та верифікацію помилок у режимі реального часу. Додатково використання розширень для JavaScript та HTML/CSS забезпечило контроль якості коду на всіх шарах архітектури.

– Інтегрований інструментарій та вбудована консоль: VS Code надає розробнику єдине згруповане робоче вікно. Наявність вбудованого терміналу усуває потребу постійного перемикання між різними вікнами операційної системи для виконання консольних команд чи налаштування локального контуру. Крім того, глибока нативна інтеграція з розподіленою системою контролю версій Git дозволяє виконувати фіксацію змін, керувати гілками розробки та синхронізувати репозиторії безпосередньо з графічного інтерфейсу редактора [39].

Обрання даного технологічного стеку забезпечує оптимальний баланс між високою обчислювальною швидкодією, надійним захистом даних та простотою реалізації клієнт-серверної архітектури. Сукупність сучасного інструментарію Fetch API, графічного рушія Chart.js та СКБД MySQL дозволяє ефективно обробляти й візуалізувати складну аналітику без використання громіздких сторонніх фреймворків. Застосування сучасних стандартів (ES6+, PDO) у поєднанні з гнучким середовищем VS Code та XAMPP гарантує чистоту програмного коду, високу архітектурну гнучкість системи та низьку вартість її подальшого супроводу й масштабування.

2.6 Реалізація основних класів та методів

Практична реалізація логічної архітектури автоматизованої системи порівняльного аналізу ПЗ базується на принципах модульності, безпеки обробки даних та асинхронності клієнт-серверної взаємодії. Нижче наведено лістинги ключових програмних компонентів системи, що відповідають спроєктованим логічним класам, із детальним аналізом методів їхньої реалізації.Ініціалізація

зв'язку із реляційною СУБД MySQL та налаштування безпечного з'єднання реалізовано у файлі db.php. Програмна логіка базується на створенні екземпляра об'єкта PDO із застосуванням захищених параметрів конфігурації. Реалізацію цього підходу наведено в лістингу 2.1.

Лістинг 2.1 – Код db.php підключення до бази даних

```
<?php
     $host = 'localhost';
     $db   = 'software_compare';
     $user = 'root';
     $pass = '';
     $charset = 'utf8mb4';
     $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
     $options = [
         PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
         PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
         PDO::ATTR_EMULATE_PREPARES   => false,
     ];

     try {
         $pdo = new PDO($dsn, $user, $pass, $options);
     } catch (\PDOException $e) {
         die("Помилка підключення до БД: " . $e->getMessage());
     }
?>
```

У даному коді масив конфігурацій $options відіграє ключову роль у безпеці додатку. Параметр PDO::ATTR_EMULATE_PREPARES=>false вимикає штучну емуляцію підготовлених виразів силами PHP, змушуючи систему виконувати реальне екранування на рівні сервера MySQL, що запобігає загрозі SQL-ін'єкцій. Режим PDO::FETCH_ASSOC оптимізує споживання пам'яті, повертаючи чисті асоціативні масиви без дублювання даних.

Серверна логіка обробки вхідних запитів реалізовано за допомогою окремих функціональних скриптів-контролерів. Файл get_software.php відповідає за динамічне вилучення списку програмних продуктів за обраною категорією.

Лістинг 2.2 – Метод для вибірки ПЗ за категорією.

```
<?php
    require_once 'db.php';

    header('Content-Type: application/json; charset=utf-8');

    if                (isset($_GET['category_id'])                &&
    is_numeric($_GET['category_id'])) {
        $category_id = (int) $_GET['category_id'];

        $stmt  =  $pdo->prepare("SELECT  id,  name,  logo_url  FROM
    software WHERE category_id = :category_id ORDER BY name ASC");
        $stmt->execute(['category_id' => $category_id]);

        $software = $stmt->fetchAll();

        echo json_encode($software);
    } else {

        echo json_encode([]);
    }
?>
```

Через даний код реалізована конструкція перевірки через if та функція is_numeric, вони відіграють ключову роль у обробці запиту, відсікаючи потенційно некоректні параметри ще на етапі їхнього надходження. Примусове приведення типу (int) забезпечує сувору типізацію змінної $category_id, гарантуючи передачу в базу даних виключно цілого числа. Метод $pdo->prepareспільно з іменованим маркером :category_id реалізує механізм підготовлених виразів, який надійно відокремлює структуру SQL-команди від користувацьких даних, повністю ліквідуючи загрозу SQL-ін'єкцій. Функція json_encode забезпечує фінальну трансформацію отриманого масиву продуктів в універсальний текстовий формат JSON, а блок else виступає захисним інструментом, який у разі будь-якої помилки валідації повертає безпечну порожню структуру замість системних повідомлень про збій.

Для попарного порівняльного аналізу характеристик двох існуючих систем розроблено модуль compare.php див. лістинг 2.3. Цей модуль вилучає повні записи з бази даних та розподіляє їх для передачі на клієнтську частину.

Лістинг 2.3 – Код модуля compare.php

```
<?php
    require_once 'db.php';
    header('Content-Type: application/json; charset=utf-8');

    if (isset($_GET['id1']) && isset($_GET['id2'])) {
        $id1 = (int)$_GET['id1'];
        $id2 = (int)$_GET['id2'];

        $stmt = $pdo->prepare("SELECT * FROM software WHERE id IN (?,
    ?)");
        $stmt->execute([$id1, $id2]);

        $results = $stmt->fetchAll();

        $data = [];

        foreach ($results as $row) {
            if ($row['id'] == $id1) {
                $data['software1'] = $row;

            } elseif ($row['id'] == $id2) {
                $data['software2'] = $row;
            }
        }

        echo json_encode($data);
    } else {
        echo json_encode(['error' => 'Не передано ID програм для
    порівняння']);}
?>
```

В даному лістингу цикл foreach відіграє ключову роль в архітектурі модуля, розподіляючи отримані рядки за чіткими ключами software1 та software2 для правильної ідентифікації систем на фронтенді незалежно від порядку їхнього вилучення з БД. Функція json_encode забезпечує фінальну структуризацію відповіді, повертаючи або готовий для візуалізації об'єкт порівняння, або явне повідомлення про помилку у текстовому форматі JSON у разі збою обробки запиту.Для захисту панелі управління застосовано механізм сесій, який забезпечує контроль доступу лише для авторизованих адміністраторів. Обробка форми логіну та створення сесії реалізовані у контролері admin_auth.php див. лістинг 2.4.

Лістинг 2.4 – Механізм автентифікації адміністратора

```
<?php
session_start();
$ip = $_SERVER['REMOTE_ADDR'];
if (!checkLoginAttempts($ip) || !hash_equals($_SESSION['csrf_token'],
$_POST['csrf_token'] ?? '')) {
    logSecurityEvent('SECURITY_VIOLATION', $ip);
    header('Location: admin_login.php?error=secure_block');
    exit;
}
if ($_POST['password'] === $adminPassword) {
    resetLoginAttempts($ip);
    session_regenerate_id(true);
    $_SESSION['admin_logged_in'] = true;
    $_SESSION['admin_ip'] = $ip;
    $_SESSION['admin_user_agent']=$_SERVER['HTTP_USER_AGENT'] ?? '';
    logSecurityEvent('ADMIN_LOGIN_SUCCESS', $ip);
    header('Location: admin.php');
} else {
    incrementLoginAttempts($ip);
    header('Location: admin_login.php?error=invalid');
}
?>
```

Після введення пароля система перевіряє його відповідність встановленому значенню. У разі успішної автентифікації створюється сесія користувача, у якій зберігається інформація про факт входу адміністратора та час авторизації. Якщо введений пароль є неправильним, система формує повідомлення про помилку та перенаправляє користувача назад на сторінку входу. Для цього використовується механізм перенаправлення HTTP (header()), а інформація про помилку зберігається у змінній сесії. Такий підхід забезпечує базовий рівень безпеки та обмежує доступ до функцій адміністрування стороннім користувачам.

Логіка поведінки користувацького інтерфейсу панелі адміністратора, динамічне відображення модальних вікон редагування, а також локальна фільтрація каталогу реалізовані мовою JavaScriptна лістингу 2.5. Використання клієнтських сценаріїв дозволяє забезпечити інтерактивність вебзастосунку та підвищити швидкість взаємодії користувача з системою без необхідності постійного оновлення сторінки.

Лістинг 2.5 – Методи контролю вікна та таблиці admin.js

```
function filterTable() {
    const val  = document.getElementById('filterCategory').value;
    const rows = document.querySelectorAll('#catalogTable tbody tr');
    rows.forEach(row => {
        row.style.display = (!val || row.dataset.category === val) ?
'' : 'none';
    });
}
function openEditModal(sw) {
    document.getElementById('editModal').classList.remove('hidden');
    document.body.style.overflow = 'hidden';

    const fields = [
        'id', 'name', 'category_id', 'logo_url', 'website_url',
        'min_cpu', 'min_ram_gb', 'storage_gb', 'supported_os',
        'data_formats',          'price_model',          'price_base',
'price_business',
        'trial_days', 'learning_time_h', 'user_rating'
    ];

    fields.forEach(f => {
        const el = document.getElementById('edit_' + f);
        if (el) el.value = (sw[f] !== null && sw[f] !== undefined) ?
sw[f] : '';
    });
    const cb = document.getElementById('edit_trial_available');
    if (cb) cb.checked = (sw['trial_available'] == 1);

    const catSel = document.getElementById('edit_category_id');
    if (catSel && sw['category_id']) catSel.value = sw['category_id'];

    const pmSel = document.getElementById('edit_price_model');
    if (pmSel && sw['price_model']) pmSel.value = sw['price_model'];
}
```

Функція filterTable() відповідає за фільтрацію записів каталогу за вибраною категорією. Для цього використовується метод querySelectorAll(), який отримує всі рядки таблиці, після чого кожен рядок перевіряється на відповідність обраній категорії через атрибут data-category. Якщо категорія збігається або фільтр не встановлено, запис залишається видимим, інакше — приховується шляхом зміни CSS-властивості display. Такий підхід дозволяє виконувати фільтрацію миттєво на стороні клієнта без додаткових HTTP-запитів до сервера, що позитивно впливає на продуктивність та зручність використання системи.

Функція openEditModal() реалізує механізм відкриття модального вікна для редагування інформації про програмне забезпечення. Після виклику функції модальне вікно стає видимим, а прокручування сторінки блокується для покращення взаємодії користувача з формою. Функція приймає JSON-об’єкт із даними вибраного запису та автоматично заповнює поля форми відповідними значеннями. Для цього використовується масив назв полів таблиці software, який містить 16 атрибутів, включаючи текстові поля, списки вибору та числові параметри. Всі лістинги файлів проєкту знаходяться в додатку Б.

2.7 Розробка інтерфейсу користувача

Ефективність функціонування автоматизованої системи порівняльного аналізу ПЗ безпосередньо залежить від якості проєктування користувацького інтерфейсу та забезпечення позитивного досвіду взаємодії. Головними критеріями під час розробки графічного інтерфейсу системи було обрано інтуїтивність, адаптивність, швидкість відгуку та мінімізацію кількості дій користувача для отримання кінцевого результату.Стилістичне оформлення системи базується на сучасних концепціях вебдизайну із використанням напівпрозорих елементів, м'яких градієнтів та чіткої візуальної ієрархії. Спираючись на це було визначено наступні параметри інтерфейсу:

- Шрифтова композиція: Як основний шрифт системи інтегровано гарнітуру Nunito через сервіс Google Fonts. Це забезпечує високу чіткість читання техніко-економічних показників як на моніторах ПК, так і на екранах мобільних пристроїв.
- Колірна палітра: Основний фон базується на глибоких сучасних відтінках із використанням адаптивного розмиття, що зменшує навантаження на зір користувача під час тривалого аналізу табличних даних.
- Адаптивність: Компоновка контенту реалізована за допомогою технологій CSS Flexbox та Grid Layout. Інтерфейс автоматично перебудовується під будь-яку ширину екрана.

Головна сторінка системи index.php виконує роль інтерактивного інструменту для кінцевого користувача. Вона функціонально розділена на дві основні зони: панель вибору та панель виведення аналітичних результатів. Процес взаємодії з інтерфейсом побудовано за лінійною логікою:

1. Користувач обирає категорію програмного забезпечення з першого випадаючого списку.
2. На основі обраної категорії система асинхронно активує та заповнює два наступні списки для вибору конкретних програмних продуктів для попарного порівняння.
3. Кнопка «Порівняти ПЗ» динамічно розблоковується тільки тоді коли вибрано обидві програми, що запобігає відправці некоректних запитів.
   – Після активації порівняння інтерфейс плавно відображає приховану секцію аналітичних результатів яка містить текстове резюме аналізу, детальну таблицю порівняння характеристик, сітку інтерактивних діаграм.

Зовнішній вигляд головної сторінки системи представлено на рисунку 2.4, а результати аналізу порівняння двох продуктів відображено на рисунку 2.5.

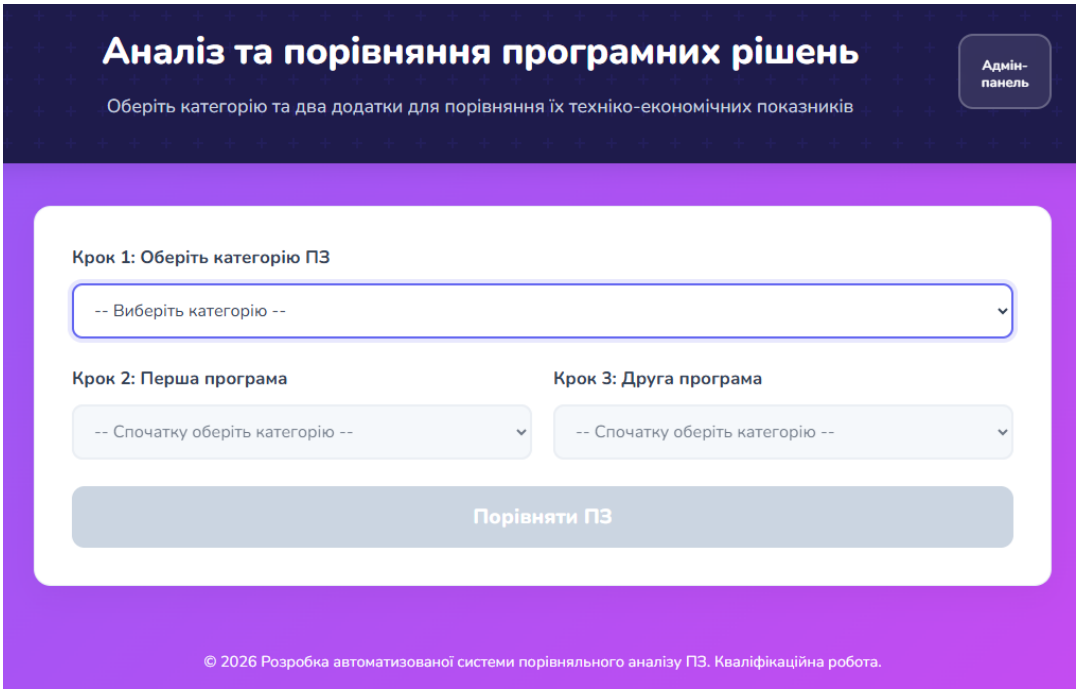

Рисунок 2.4 – Головна сторінка автоматизованої системи порівняння ПЗ

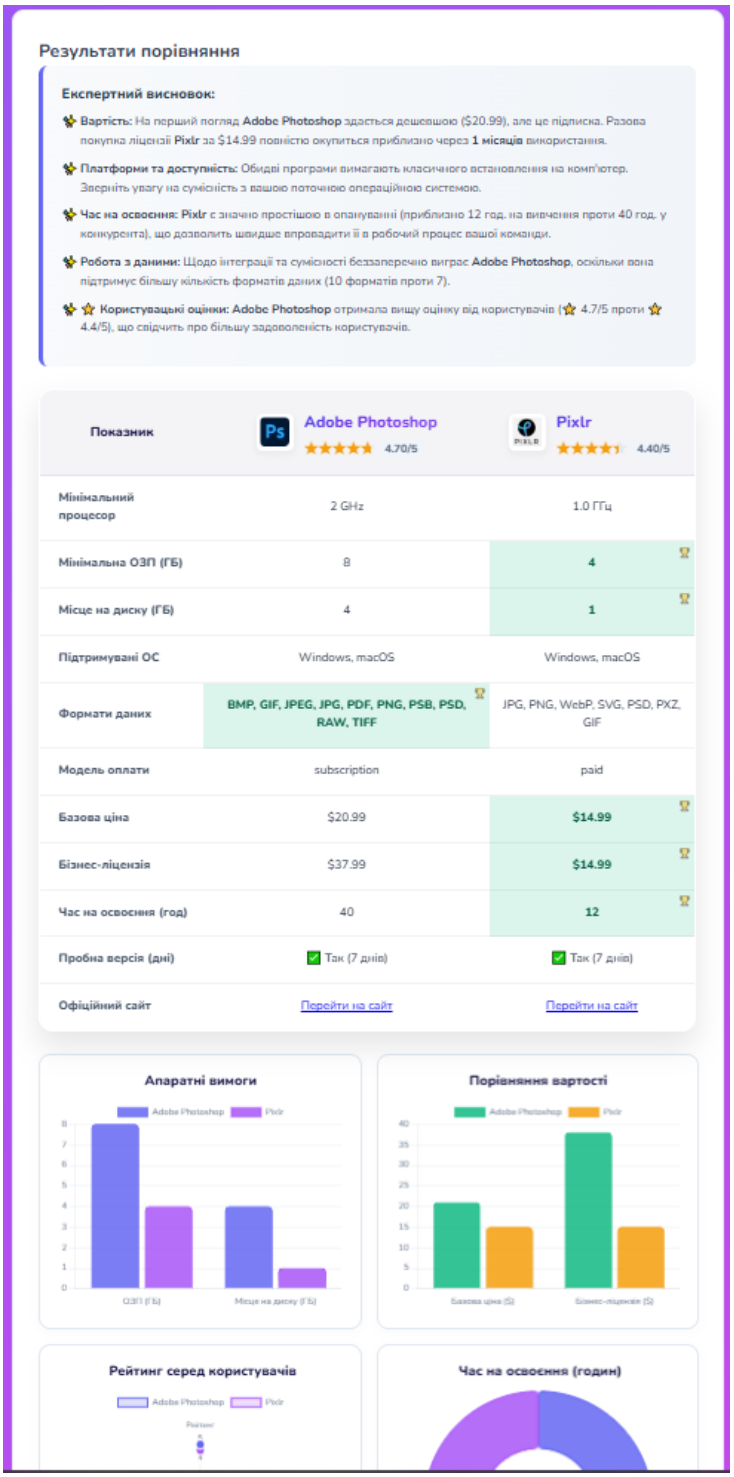

Рисунок 2.5 – Результати порівняльного аналізу та діаграм Chart.js

Доступ до адміністративних функцій суворо обмежений і вимагає обов’язкової авторизації. Інтерфейс входу реалізовано у файлі у вигляді захищеної вебформи (рис. 2.5). Для зручності користувача в полі введення пароля інтегровано кнопку перемикання його видимості. Візуальне оформлення та адаптивність форми забезпечується стилями у файлі.

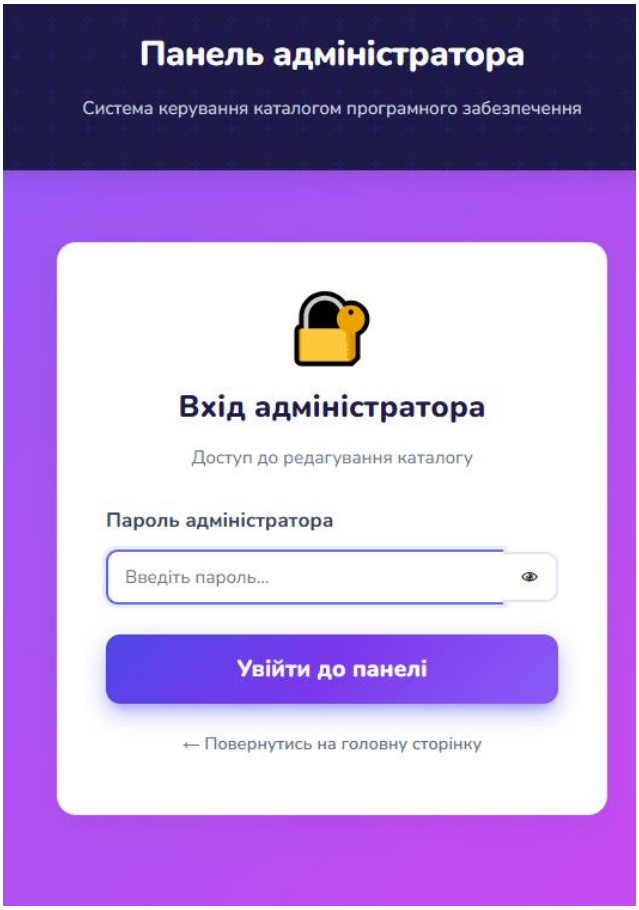

Рисунок 2.5 – Інтерфейс входу в панель адміністратора

Серверна логіка перевірки даних у файлі admin_auth_4.php реалізує комплексний захист системи, обмежуючи спроби входу до 3 із блокуванням IP-адреси на 5 хвилин та виконуючи обов'язкову серверну перевірку сесії. Для запобігання перехопленню сесії у момент автентифікації регенерується її ідентифікатор, а надалі контролюється незмінність IP-адреси, User-Agent та тривалість сесії з автовиходом після 2 годин бездіяльності адміністратора. Усі спроби авторизації, помилки та процедура виходу через форму фіксуються у захищених серверних журналахsecurity.log та admin_actions.log.

Панель управління каталогом надає повний набір CRUD-інструментів для менеджменту бази даних програмного забезпечення. Для оптимізації робочого простору інтерфейс адмін-панелі містить такі елементи:

– Секція додавання нових програм: реалізована у вигляді блоку, що згортається/розгортається за допомогою функції toggleSection(). Це дозволяє приховати масивну форму, коли адміністратор просто переглядає список.
– Швидка фільтрація таблиці: елемент select із викликом функції filterTable() миттєво приховує рядки таблиці каталогу, які не належать до обраної категорії, спрощуючи користувачу пошук потрібної програми.
– Модальне вікно редагування: Замість переходу на іншу сторінку для зміни даних, клієнтський JS-скрипт відкриває спливаюче вікно поверх поточної таблиці, блокуючи прокрутку основного екрана. Форма автоматично заповнюється всіма поточними параметрами обраного продукту.

На рисунку 2.6 зображено панель адміністратора каталогу, а на рисунку 2.7 відображено модальне вікно редагування програми яка вже є в базі даних.

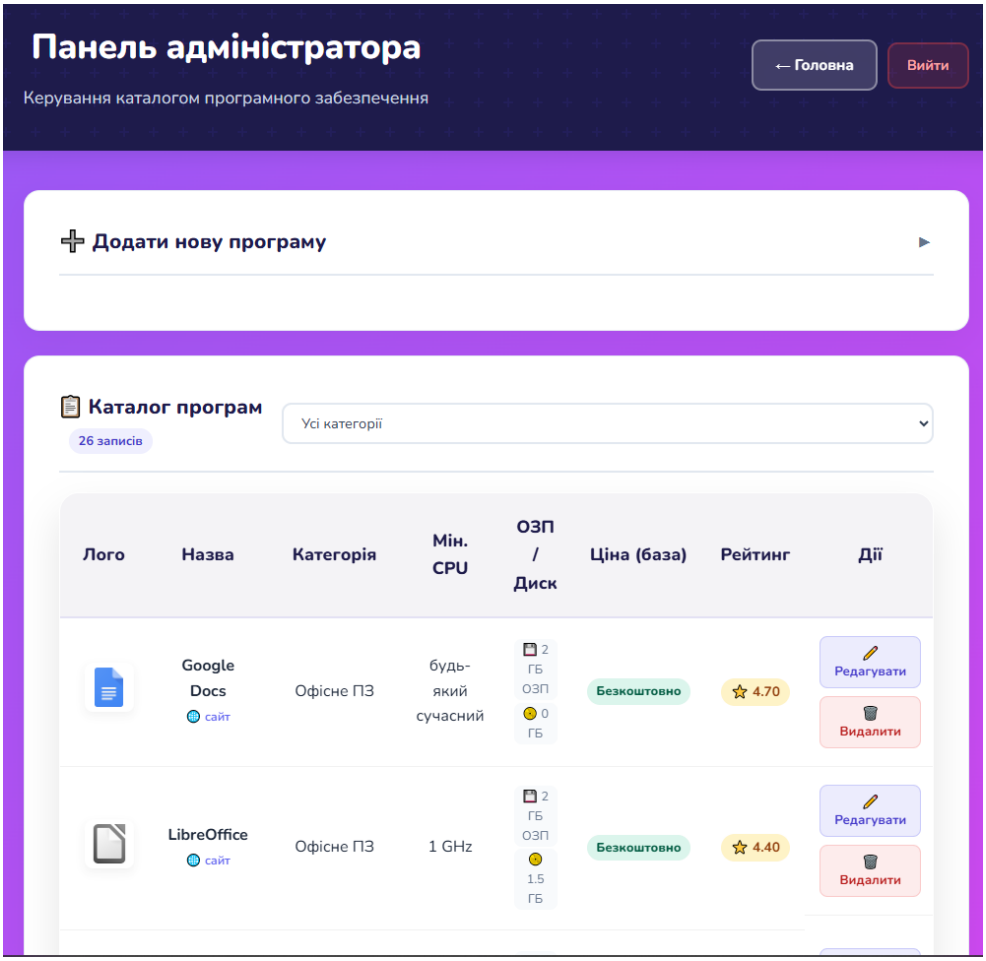

Рисунок 2.6 – Панель адміністратора

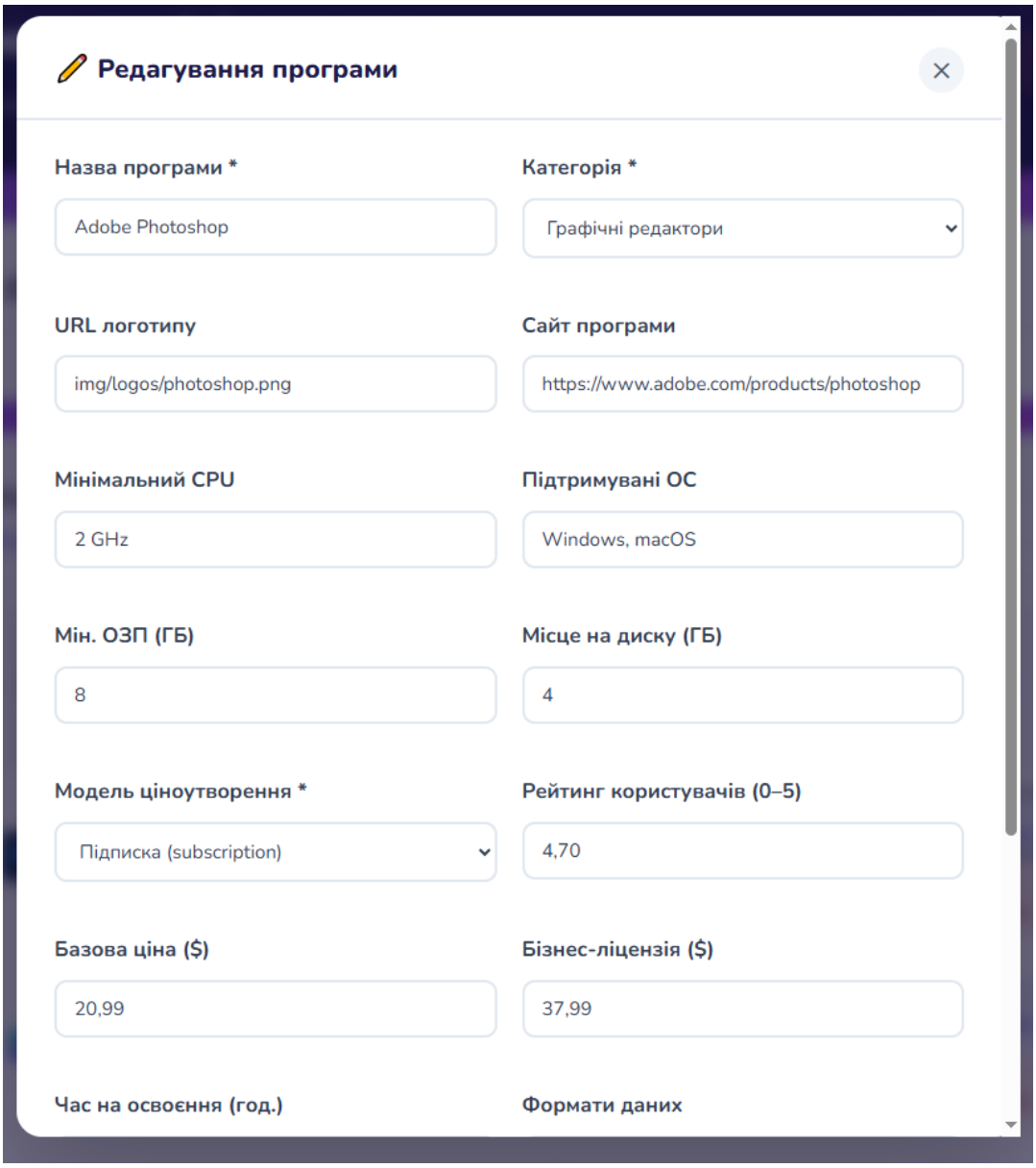

Рисунок 2.7 – Модальне вікно редагування програми

Таким чином, розроблений користувацький інтерфейс забезпечує повну відповідність функціональним вимогам системи, мінімізує час відгуку на дії користувача та створює комфортне середовище як для проведення аналізу, так і для адміністрування сховища даних.

Зважаючи на різноманітність пристроїв, з яких користувачі можуть взаємодіяти із системою, однією з ключових вимог до клієнтської частини стало забезпечення повноцінної адаптивності. Архітектура фронтенду побудована таким чином, щоб динамічно підлаштовувати відображення контенту під будь-яку

роздільну здатність екрана – від смартфонів до широкоформатних комп'ютерних моніторів.

Адаптивна поведінка системи реалізована за допомогою Flexbox, CSS Grid та медіа-запитів, що забезпечує коректне відображення інтерфейсу на різних пристроях. Елементи сторінки автоматично змінюють структуру залежно від ширини екрана, а для великих таблиць реалізовано горизонтальне прокручування. Також передбачено адаптивне масштабування шрифтів і збільшені області натискання для зручної роботи на сенсорних екранах. Вигляд адаптивного інтерфейсу відображено на рисунку 2.8.

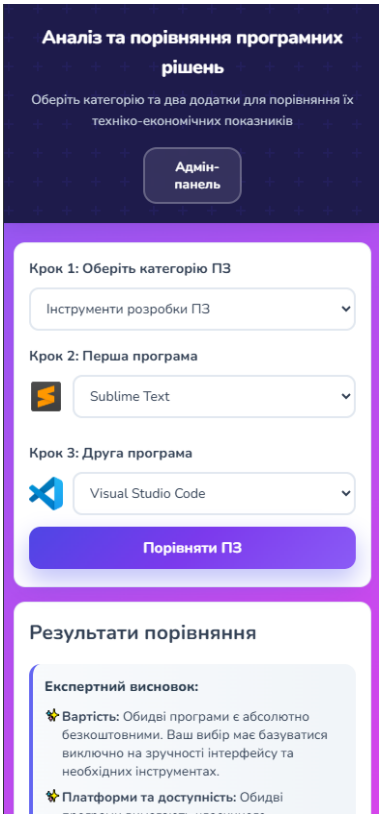

Рисунок 2.8 – Вигляд сторінки на смартфоні

У другому розділі кваліфікаційної роботи було здійснено повний цикл проєктування та практичної реалізації автоматизованої системи порівняльного аналізу програмного забезпечення. На основі проведеного аналізу предметної області було прийнято низку інженерних та архітектурних рішень, що дозволило створити повноцінний програмний продукт.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

Після завершення етапів архітектурного проєктування та програмної реалізації автоматизованої системи, критично важливим кроком є перевірка її працездатності та стабільності. У даному розділі проведено процеси комплексного тестування розробленого програмного продукту, спрямовані на виявлення та усунення можливих дефектів алгоритмів і користувацького інтерфейсу. Крім того, визначено системні вимоги та порядок розгортання вебдодатка на робочому сервері, а також здійснено його остаточну верифікацію на відповідність початковому технічному завданню.

3.1 Тестування програмної системи

Процес тестування є невід’ємним етапом життєвого циклу розробки. Його головна мета – виявлення та усунення дефектів, перевірка коректності обробки даних та забезпечення стабільної роботи як клієнтської, так і адміністративної частин вебдодатка.

3.1.1 Види та план тестування

Для всебічної перевірки програмного продукту було обрано стратегію автоматизованого тестування з використанням середовища Node.js та тестового фреймворку Jest. Для виконання HTTP-запитів до локального сервера використовувався модуль node-fetch.

Автоматизоване тестування охопило три ключові напрямки, для кожного з яких розроблено окремий набір тестів:

– Тестування API: Перевірка коректності роботи серверних PHP-обробників, валідація формату JSON, перевірка статус-кодів HTTP та швидкості обробки запитів.

– Тестування безпеки: Перевірка стійкості панелі адміністратора до базових вразливостей, таких як SQL-ін'єкції, міжсайтовий скриптинг, а також тестування механізмів авторизації та захисту від підбору паролів.
– Тестування користувацького інтерфейсу та статики: Аналіз доступності клієнтських файлів, перевірка наявності ключових DOM-елементів на сторінках та коректності завантаження стилів.

3.1.2 Розробка тестових сценаріїв

Для систематизації процесу автоматизованої перевірки було розроблено загалом 28 незалежних тестових сценаріїв. Усі тести запускалися комплексно за допомогою команди пакетного менеджера npm. Опис ключових груп автоматизованих тестів та їхні фактичні результати зведено до таблиці 3.1.

Таблиця 3.1 – Набори тестових сценаріїв та результати

Назва	**Опис тестових сценаріїв**	**Очікуваний результат**	**Фактичний результат**
Тестування API api.test.js	ТС-API-1 – ТС-API-3: Запити до get_software.php. ТС-API-12 – ТС-API-14: Обробка некоректних параметрів (невірний ID, неіснуючі файли). ТС-API-17: Тест продуктивності API.	Сервер повертає HTTP 200 та коректний JSON-масив із даними ПЗ. При помилках повертаються відповідні коди. Час відповіді в межах норми.	Відповіді сервера коректні, масиви об'єктів сформовані правильно. Помилкові запити оброблені безпечно.
Тестування безпеки security.test.js	ТС-БЗ-01 – ТС-БЗ-03: Захист сторінки логіну та обробка POST-запитів в admin_auth.php. ТС-ІН-01 – ТС-ІН-02: Симуляція SQL-ін'єкцій та XSS-атак у параметрах запиту.	Некоректні спроби авторизації блокуються. Спроби впровадження SQL-коду або JS-скриптів нейтралізуються сервером.	Вразливостей не виявлено. Перенаправлення неавторизованих користувачів працює миттєво.
Тестування UI/UX ui.test.js	ТС-ГС-01 – ТС-ГС-03: Завантаження index.php, перевірка наявності полів вибору та кнопок. ТС-АДП-01 – ТС-АДП-03: Доступність файлів style.css, admin.css та script.js.	Сторінки віддають статус 200., скрипти та таблиці стилів успішно завантажуються.	Усі необхідні DOM-елементи та статичні файли доступні для клієнта.

Для остаточного контролю якості було активовано комплексний сценарій тестування всього проєкту за допомогою команди npm run test:all. На рисунку 3.1 зображено результат виконання набору тестів для серверних endpoint системи, який підтверджує правильність обміну даними у форматі JSON.

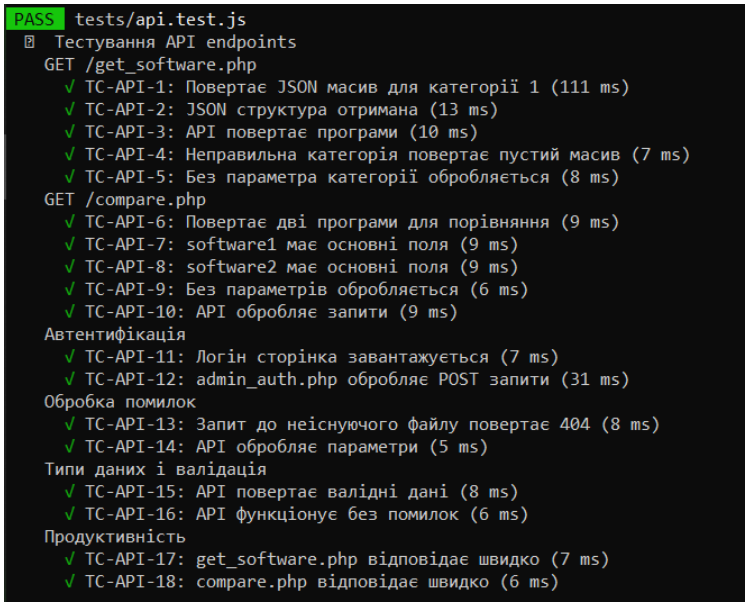

Рисунок 3.1 – виконання тестів для API

Для перевірки стійкості панелі адміністратора до кібератак та вразливостей було запущено тести безпеки, результати виконання якого наведено на рисунку 3.2.

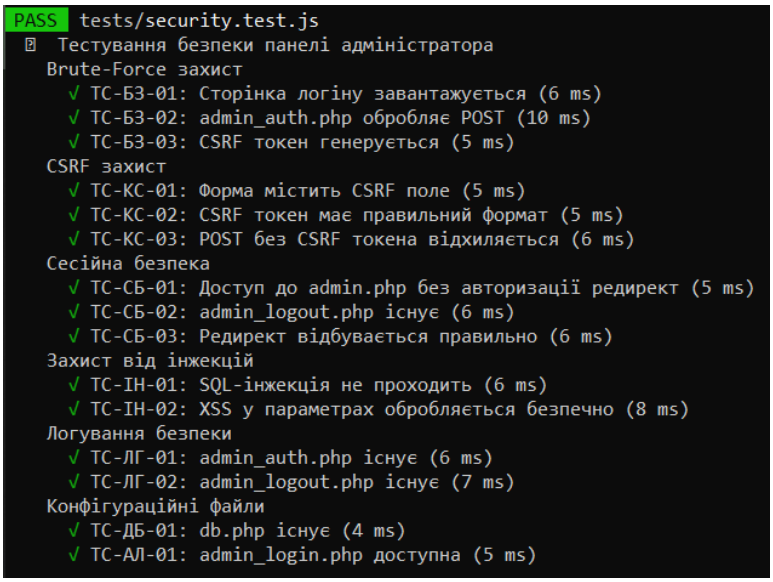

Рисунок 3.2 – Виконання тестів безпеки.

Перевірка доступності статичних ресурсів, оформлення системи та наявності обов'язкових інтерактивних елементів інтерфейсу відображена на рисунку 3.3.

```
PASS tests/ui.test.js
  Тестування UI/UX функціоналу
    Головна сторінка (index.php)
      √ ТС-ГС-01: Сторінка завантажується успішно (13 ms)
      √ ТС-ГС-02: Наявні всі обов'язкові елементи (44 ms)
      √ ТС-ГС-03: Наявна кнопка входу до адмін-панелі (4 ms)
      √ ТС-ГС-04: Селект категорій містить всі 5 категорій (5 ms)
    Сторінка входу (admin_login.php)
      √ ТС-ЛГН-01: Сторінка логіну завантажується (5 ms)
      √ ТС-ЛГН-02: Наявна форма входу з полем пароля (5 ms)
      √ ТС-ЛГН-03: Наявна кнопка для показу/приховування пароля (5 ms)
      √ ТС-ЛГН-04: Показується повідомлення про помилку при error=invalid (5 ms)
      √ ТС-ЛГН-05: Форма входу має CSRF захист (69 ms)
    Адмін-панель (admin.php)
      √ ТС-АДМ-01: Без авторизації - редирект на login (5 ms)
      √ ТС-АДМ-02: Адмін-панель вимагає авторизацію (9 ms)
      √ ТС-АДМ-03: Адмін-панель має захист доступу (4 ms)
    API endpoints
      √ ТС-АПІ-01: GET /get_software.php повертає дані (48 ms)
      √ ТС-АПІ-02: GET /compare.php доступний (8 ms)
    Адаптивність верстки
      √ ТС-АДП-01: CSS файл style.css завантажується (3 ms)
      √ ТС-АДП-02: CSS файл admin.css завантажується (3 ms)
      √ ТС-АДП-03: JavaScript файл script.js завантажується (5 ms)
    Валідація форм
      √ ТС-ФРМ-01: Форма логіну вимагає пароль (5 ms)
```

Рисунок 3.3 – Виконання тестів інтерфейсу

За результатами програмного тестування було проведено 28 розроблених автоматизованих тестів, 100% з яких були успішними. Загальний час виконання всього пакету тестів склав менш ніж 2 секунди, що підтверджує високу швидкодію системи. Таким чином, розробка програмно підтвердила свою надійність, захищеність та повну готовність до розгортання в реальному середовищі.

3.2 Розгортання програмної системи та системні вимоги

Програмна система розроблена з урахуванням сучасних стандартів вебтехнологій, що робить її невимогливою до апаратних ресурсів сервера та дозволяє легко інтегрувати в існуючі інфраструктури.

Системні вимоги до серверного середовища:

– Операційна система: Linux або Windows.
– Вебсервер: Apache 2.4+ або Nginx.
– Інтерпретатор: PHP версії 7.4 або вище з розширенням PDO_MySQL.
– СКБД: MySQL версії 5.7 або MariaDB 10.3.

Вимоги до клієнтського середовища:

– Пристрій: ПК, ноутбук, планшет або смартфон з доступом до мережі інтернет.
– Браузер: Будь-який сучасний браузер із підтримкою HTML5, CSS3, Flexbox/Grid та увімкненим виконанням JavaScript.

Процес розгортання системи на робочому сервері включає наступні кроки:

1. Створення порожньої БД software_compare у СКБД сервера та імпорт структури.
2. Копіювання директорії проєкту у кореневу директорію вебсервера.
3. Налаштування параметрів авторизації для доступу до БД у файлі db.php.
4. Забезпечення правильних прав доступу на читання та виконання файлів вебсервером.

3.3 Верифікація програмної системи

Процес верифікації системи являє собою комплексне підтвердження того, що розроблений програмний продукт повністю задовольняє всім технічним специфікаціям, функціональним критеріям та експлуатаційним вимогам, які були задекларовані на етапі проєктування у першому розділі. За результатами проведеного всебічного аналізу, якісного налагодження та тестування було експериментально доведено повне виконання інженерно-технічних вимог до архітектури. Зокрема, розроблена реляційна база даних гарантує сувору каскадну цілісність інформаційних зв’язків, використання підготовлених виразів на рівні драйвера PDO дозволило повністю ліквідувати ризики впровадження SQL-ін'єкцій, а впроваджений механізм сесій забезпечив надійний захист та ізоляцію адміністративної панелі керування від несанкціонованого доступу.

Паралельно з цим було підтверджено стовідсоткове забезпечення функціонального компоненту системи. Програмний комплекс успішно реалізує бізнес-логіку збору, структурування, зіставлення та інтерактивної візуалізації техніко-економічних показників програмного забезпечення. При цьому всі розрахункові алгоритми та модулі побудови аналітичних графіків функціонують без збоїв у чіткій відповідності до заданих сценаріїв розробки. Особливу увагу під час верифікації було приділено інтерфейсу користувача, який продемонстрував повну відповідність сучасним критеріям ергономіки. Завдяки адаптивній верстці графічна оболонка безпомилково підлаштовується під будь-які роздільні здатності екранів мобільних чи комп'ютерів, а інтеграція асинхронного обміну даними через Fetch API дозволила досягти високої швидкості відгуку та інтуїтивно зрозумілої взаємодії без потреби повного перезавантаження вебсторінок.

За результатами контрольних випробувань, можна стверджувати, що розроблена автоматизована система успішно пройшла всі етапи верифікації, підтвердила свою стабільність та є повністю готовою до розгортання в реальному операційному середовищі для подальшої практичної експлуатації кінцевими користувачами.

У третьому розділі було проведено тестування розробленої автоматизованої системи за допомогою сформованих тестових сценаріїв, що підтвердило її стабільність та безпеку. Обґрунтовано системні вимоги та процес розгортання вебдодатка на базі серверного стеку PHP/MySQL. У результаті верифікації доведено відповідність програмного продукту початковим вимогам, що повністю підтверджує готовність системи до реального впровадження та дослідної експлуатації.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Створення сучасних програмних продуктів нерозривно пов'язане з тривалим перебуванням фахівця перед монітором комп'ютера у межах офісу чи домашнього кабінету. Цей тип професійної діяльності супроводжується специфічним комплексом навантажень, які здатні негативно вплинути на загальне самопочуття та продуктивність програміста. У цьому розділі проаналізовано ключові ергономічні ризики в системі взаємодії людини з обчислювальною технікою, а також сформульовано чіткі інженерно-технічні вимоги до організації робочого простору з метою мінімізації виробничих небезпек.

4.1 Безпека життєдіяльності

Наукове проектування життєдіяльності людини базується на концепції безвідмовної взаємодії в системі «людина – машина – середовище існування», де головним критерієм ефективності виступає ергономічна відповідність технічних засобів біологічним особливостям користувача [40]. В умовах інтенсивної цифровізації праці ергономіка трансформувалася з чисто технічної дисципліни у фундаментальний елемент безпеки, оскільки невідповідність параметрів техніки фізіологічним нормам людини веде до стрімкого зниження працездатності, зростання кількості помилок у програмному коді та розвитку професійних патологій [41]. Для розробника програмних систем найбільшу загрозу становлять три взаємопов’язані ергономічні проблеми: критичне навантаження на зоровий аналізатор, статичне перенапруження опорно-рухового апарату та нервово-психічне (когнітивне) виснаження [41].

Зорова втомлюваність є первинним деструктивним фактором під час взаємодії з відео дисплейними терміналами. Фізіологічний механізм розвитку комп'ютерного зорового синдрому зумовлений постійним перенапруженням акомодаційного м'яза ока, який змушений утримувати фокус на дискретному піксельному зображенні, що постійно мерехтить [40]. На відміну від читання

паперових носіїв, сприйняття інформації з монітора супроводжується значним зниженням частоти кліпання повік, що призводить до висихання слізної плівки та розвитку синдрому «сухого ока». Цей процес викликає гіперемію кон'юнктиви, відчуття стороннього тіла в очах та прогресуюче зниження гостроти зору [41].

З метою усунення цієї проблеми безпосередньо в архітектурі розроблюваного програмного продукту реалізовано ергономічні принципи проектування користувацьких інтерфейсів (UI/UX). У системі застосовано адаптивну колірну схему з оптимальним рівнем яскравості та колірного контрасту між фоном і текстовими блоками, що виключає появу сліпучого ефекту. Вибір шрифтових гарнітур та міжрядкових інтервалів здійснювався з урахуванням кутового розміру знаків, що забезпечує легке зчитування метрик аналізу програмного забезпечення без потреби наближення обличчя до екрана. Такий інженерний підхід дозволяє перенести частину захисних функцій безпосередньо на рівень програмного забезпечення, знижуючи екзогенне навантаження на органи зору користувачів системи [42].

Статичне навантаження на кістково-м’язову систему розробника є наслідком тривалого перебування у сидячому положенні, яке є біомеханічно нефізіологічним для хребта. За тривалої підтримки фіксованої пози відбувається постійне ізометричне напруження м'язів шиї, плечового поясу та попереку, що призводить до стиснення кровоносних судин та хронічної ішемії м'язових тканин [41]. Наслідком цього стає деформація між хребцевих дисків та розвиток остеохондрозу. Окремою ергономічною проблемою є карпальний тунельний синдром, який виникає через регулярне статико-динамічне навантаження на кисть руки під час роботи з клавіатурою та неклавіатурними пристроями введення [42]. Постійне згинання зап'ястя викликає набряк сухожиль, які затискають серединний нерв у вузькому кістковому каналі, викликаючи стійкий больовий синдром та втрату чутливості пальців.

Когнітивне та нервово-емоційне напруження в роботі програміста викликане необхідністю одночасного утримання в оперативній пам'яті великої кількості абстрактних логічних зв'язків та тривалого пошуку помилок в архітектурі коду.

Інформаційне перевантаження викликає гальмування в корі головного мозку, що проявляється розсіяністю уваги, сповільненням реакції та психоемоційним вигоранням [41]. Для нейтралізації цих явищ розроблено раціональний алгоритм організації життєдіяльності на робочому місці.

Основним інструментом зниження кумулятивного фізіологічного навантаження є впровадження жорсткого тайм-менеджменту праці. Загальний час безпосередньої роботи за екраном монітора обмежується шістьма годинами за зміну, а тривалість безперервного сеансу не перевищує двох годин [40]. Через кожні шістдесят хвилин інтенсивної праці встановлюються регламентовані регламентом безпеки паузи тривалістю п'ятнадцять хвилин. Протягом цих пауз розробник повністю ізолюється від монітора та виконує спеціально розроблений комплекс вправ для акомодаційного м'яза ока та динамічну гімнастику для відновлення лімфотоку та кровообігу у нижніх кінцівках і хребті.

4.2 Основи охорони праці

Організація робочого місця оператора персонального комп'ютера є одним з ключових факторів, що визначають продуктивність праці та збереження здоров'я фахівця в довгостроковій перспективі [41]. Ергономічний підхід до формування робочого простору дозволяє мінімізувати фізичне та психологічне навантаження на розробника і повністю виключити ризик розвитку професійних захворювань. Вимоги до робочих місць, обладнаних відеодисплейними терміналами, чітко регламентуються нормативно-правовими актами України з охорони праці.

Правову та нормативну основу побудови безпечного робочого місця складають ДСанПін 3.3.2.007-98 та НПАОП 0.00-7.15-18. Відповідно до цих документів, категорично заборонено розміщувати комп'ютерні робочі місця у підвальних приміщеннях. На кожне окреме робоче місце, обладнане сучасним персональним комп'ютером, виділяється площа не менше 6 квадратних метрів, а загальний повітряний об'єм приміщення на одного працівника становить не менше 20 кубічних метрів, що попереджає накопичення вуглекислого газу та

антропогенних токсинів. Просторова компоновка робочих місць здійснюється з урахуванням вимог ДСТУ 7299:2013 [43].

Архітектурний базис робочого місця складають ергономічний стіл та крісло, параметри яких обрані відповідно до ДСТУ 8604:2015 [44]. Конструкція столу забезпечує стійкість та повну відсутність вібрацій від зовнішніх джерел. Висота робочої поверхні столу жорстко зафіксована на рівні 750 мм, що відповідає середнім антропометричним характеристикам дорослої людини. Поверхня столу має матове покриття з коефіцієнтом відбиття світла до 0,4, що повністю унеможливлює появу світлових відблисків, які засліплюють користувача. Підпільний простір для ніг має вільні інженерні габарити: висота становить 650 мм, ширина – 550 мм, а глибина на рівні колін – 500 мм, що гарантує безперешкодну зміну положення ніг протягом дня.

Робоче крісло обирається підйомно-поворотного типу, обладнане п'ятипроменевою опорою з роликами, що ковзають без зусиль, відповідно до вимог ДСТУ 7951:2015. Конструкція крісла забезпечує динамічну підтримку постави завдяки регулюванню висоти сидіння в діапазоні 400–520 мм та кута нахилу спинки відносно сидіння від 90 до 110 градусів. Спинка крісла має анатомічний вигин у поперековій зоні, що забезпечує рівномірний розподіл маси тіла та знімає напругу з нижніх відділів хребта, запобігаючи виникненню застійних явищ у малому тазі.

Оптичне та просторове розміщення відеодисплейного монітора безпосередньо впливає на стан шийно-грудного відділу. Монітор встановлюється на відстані 650 мм від очей розробника [44]. Його взаємне розташування обирається таким чином, щоб центральна точка екрана знаходилася нижче рівня очей на 15–20 градусів, а верхня кромка корпусу монітора була розташована точно на лінії зору [44]. Кут нахилу площини екрана відносно вертикалі становить 12 градусів. Для виключення утворення дзеркальних бліків монітор розміщують боком до віконних прорізів, що виключає падіння прямих сонячних променів на матрицю дисплея.

Клавіатура та миша розташовуються безпосередньо на робочому столі на відстані 200 мм від краю, зверненого до розробника. Це забезпечує надійну опору

для передплічь та виключає провисання ліктів у повітрі. Кут у ліктьовому суглобі під час введення даних утримується в діапазоні 95–100 градусів, що мінімізує статичне напруження м'язів плечового поясу.

Світлотехнічне середовище в офісному приміщенні організовано за комбінованою системою, де природне інсоляційне освітлення доповнюється штучним. Відповідно до ДБН В.2.5-28:2018, рівень загальної штучної освітленості на горизонтальній поверхні столу становить 450 люкс . Як джерела штучного світла застосовано сучасні світлодіодні матриці зі світловою температурою 4000 К та індексом кольоропередачі не менше 80. Світильники загального освітлення встановлюються у вигляді суцільних або переривчастих ліній, розташованих паралельно до вікон та лінії погляду програміста, що виключає появу стробоскопічного ефекту та засліпленості.

Мікрокліматичний баланс робочої зони підтримується автоматизованими системами кондиціонування та припливно-витяжної вентиляції згідно з параметрами ДСН 3.3.6.042–99. У приміщенні, де виконується розробка програмного продукту, підтримуються оптимальні параметри мікроклімату для робіт легкої фізичної праці. Температура повітря в зимовий період року становить 22–23 °С, а в літній період – 23–24 °С. Відносна вологість повітря становить 50%, що попереджає накопичення зарядів статичної електрики на корпусах техніки, а швидкість руху повітряних мас обмежена величиною 0,1 м/с для виключення виникнення протягів. Повне та систематичне виконання описаних інженерно-технічних вимог з охорони праці дозволяє створити максимально безпечне ергономічне середовище, що гарантує збереження здоров'я розробника та високу продуктивність його праці.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра успішно розв’язано актуальну інженерно-технічну задачу – розробку та впровадження автоматизованої системи порівняльного аналізу програмних рішень. Створений програмний продукт спрямований на оптимізацію процесів прийняття рішень під час вибору інформаційних систем та цифрових рішень у корпоративному й індивідуальному середовищах, що дозволяє знизити часові та фінансові витрати на етапі аналізу IT-інфраструктури.

Проведено аналіз предметної області та існуючих методологій порівняння програмних продуктів. На основі цього сформовано комплексну матрицю критеріїв оцінювання, яка включає як технічні параметри, так і економічні показники.

Спроєктовано оптимальну реляційну структуру бази даних у середовищі СУБД MySQL. Створено нормалізовані таблиці з підтримкою зовнішніх ключів для забезпечення цілісності даних на рівні СКБД. Архітектура системи побудована за модульним принципом на основі клієнт-серверної технології, що забезпечує високу швидкість обробки транзакцій та гнучкість масштабування каталогу.

Розроблено легкий та адаптивний вебдодаток без використання важких фреймворків, що забезпечує мінімальний час завантаження сторінок. Серверну логіку реалізовано за допомогою мови PHP з використанням технології безпечного підключення PDO. Клієнтську частину побудовано на базі ванільного JavaScript, HTML5 та CSS3. Інтегровано сучасну бібліотеку Chart.js, що дозволило реалізувати чотири графічні компоненти для інтерактивного візуального порівняння.

Розроблено захищений модуль адміністратора для динамічного керування каталогом програмного забезпечення. Впроваджено комплекс інженерно-технічних рішень для захисту системи, який включає: механізми захисту від підробки міжсайтових запитів, інтелектуальне обмеження спроб авторизації для блокування Brute-Force атак, верифікацію сесій за IP-адресою та User-Agent для запобігання викраденню сесій.

Розроблена система забезпечує адаптивність інтерфейсу під будь-які типи пристроїв та роздільну здатність екранів. Використання асинхронних JavaScript-запитів дозволило досягти безперервного користувацького досвіду під час фільтрації каталогу та виведення аналітики без повного перезавантаження сторінок сайту. Впроваджено суворі ергономічні стандарти оформлення інтерфейсу, що знижують когнітивне та зорове навантаження на оператора.

Серверна частина демонструє високу продуктивність – час формування JSON-відповіді на запити порівняння становить менше 50 мілісекунд у локальному середовищі. Захисні механізми блокують доступ до адмін-панелі після 3 невдалих спроб введення пароля на фіксований час. Сесія адміністратора автоматично анулюється після 7200 секунд бездіяльності.

Працездатність та стійкість розробленого програмного забезпечення повністю підтверджені впровадженням сучасної методології автоматизованого тестування. За допомогою фреймворку Jest та середовища виконання Node.js було створено та успішно виконано три комплексні групи тестів.

Розроблена автоматизована система є повністю завершеним програмним продуктом і готова до розгортання на хостинг-майданчиках із підтримкою вебсервера та СКБД MySQL. Вона рекомендується до практичного використання:

1. Малими та середніми підприємствами для проведення внутрішнього техніко-економічного аудиту перед закупівлею корпоративного ПЗ.
2. IT-консалтинговими компаніями та системними інтеграторами як інструмент швидкого підбору альтернативних програмних рішень для клієнтів.
3. Освітніми закладами як інформаційна платформа для порівняння навчального та спеціалізованого програмного забезпечення.

Перспективи подальшого розвитку проєкту полягають у розширенні архітектури бази даних для підтримки користувацьких відгуків, впровадженні експорту звітів порівняння у формати PDF/XLSX, а також інтеграції алгоритмів машинного навчання для автоматичного формування рекомендацій на основі індивідуальних обмежень користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 45 с. (https://elartu.tntu.edu.ua/handle/lib/50317).

2. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). Version 4.0 / ed. H. Washizaki. IEEE Computer Society, 2024. 411 p.

3. Грицюк Ю. І. Якість програмного забезпечення та методи її оцінювання: навч. посіб. Львів : Вид-во ЛДУ БЖД, 2018. 216 с.

4. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/

5. Лавренюк С. В. Моделі та методи оцінювання сукупної вартості володіння IT-інфраструктурою підприємства // Сучасні інформаційні системи. 2019. Т. 3, № 2. с. 45–52.

6. Олійник О. О., Ткаченко В. В. Аналіз сучасних платформ вибору програмного забезпечення: переваги та недоліки // Інформаційні технології та комп'ютерна інженерія. 2021. № 1. с. 34–41.

7. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI: https://doi.org/10.32782/IT/2025-2-13.

8. Ковальчук А. М. Проєктування веб-орієнтованих систем підтримки прийняття рішень на основі відкритих технологій. Київ : Наук. думка, 2020. 185 с.

9. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing

devices in technological processes, 82(2), 360–364. https://doi.org/10.31891/2219-9365-2025-82-52

10. Tsupryk H. LLM-based Extraction from Resumes / D. Olianin, H. Tsupryk // Advanced Technologies in Scientific Research: collection of scientific papers with proceedings of the 1st International Scientific and Practical Conference, Rotterdam, Netherlands, 20–22 August 2025. – International Scientific Unity, 2025. – 72-76.

11. Yaroslav Kotov, Evhenia Yavorska, Halyna Tsupryk, Róża Dzierżak 1 , Oleksandr Reshetnik, Viktoriia Bokovets (2025) Evaluating interoperability and data quality in FHIR-based AI assessment pipelines. Proc. SPIE 14009, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2025, 140091F (30 December 2025) https://doi.org/10.1117/12.3100561

12. Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.

13. Лавріщева К. М. Програмна інженерія предметних областей: Підручник / К. М. Лавріщева. – К.: Академперіодика, 2016. – 354 с.

14. Грицюк Ю. І. Програмна інженерія: технологія розробки програмного забезпечення : навч. посіб./ Ю. І. Грицюк.– Львів : Вид-во ЛДУ БЖД, 2018. – 487 с.

15. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development / Craig Larman. – 3rd ed. – London : Pearson, 2015. – 736 p.

16. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language / Martin Fowler. – 3rd ed. – Boston : Addison-Wesley, 2004. – 208 p.

17. MDN Web Docs. Working with JSON data [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (date of access: 22.05.2026).

18. PHP Manual. Session Management and Security [Electronic resource]. – Access mode: https://www.php.net/manual/en/book.session.php (date of access: 22.05.2026)

19. Sommerville I. Software Engineering / Ian Sommerville. – 10th ed. – London : Pearson, 2015. – 816 p.

20. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin. – Boston : Prentice Hall, 2017. – 432 p.

21. Flanagan D. JavaScript: The Definitive Guide. Master the World's Most-Used Programming Language / David Flanagan. – 7th ed. – Sebastopol : O'Reilly Media, 2020. – 706 p.

22. Fowler M. Patterns of Enterprise Application Architecture / Martin Fowler. – Boston : Addison-Wesley, 2002. – 576 p.

23. Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. – Boston : Addison-Wesley, 1994. – 395 p.

24. Краковський В. В. Веб-технології та проектування веб-систем : навч. посіб. / В. В. Краковський. – Чернівці : Техно, 2021. – 264 с.

25. Miles R. Learning UML 2.0 / R. Miles, K. Hamilton. – Sebastopol : O'Reilly Media, 2006. – 269 p.

26. ConnoLy T. Database Systems: A Practical Approach to Design, Implementation, and Management / T. Connolly, C. Begg. – 6th ed. – Boston : Pearson, 2014. – 1440 p.

27. Пасічник В. В. Організація баз даних та знань : підручник / В. В. Пасічник, В. А. Резніченко. – К. : Видавнича група BHV, 2011. – 448 с.

28. Beighley L. Head First SQL: Your Brain on SQL -- A Learner's Guide / Lynn Beighley. – Sebastopol : O'Reilly Media, 2007. – 528 p.

29. Booch G. The Unified Modeling Language User Guide/ G. Booch, J. Rumbaugh, I. Jacobson. – 2nd ed. – Upper Saddle River: Addison-Wesley, 2005. – 496 p.

30. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development / Craig Larman. – 3rd ed. – Upper Saddle River : Prentice Hall, 2004. – 736 p.

31. Грудочка О. В. Об'єктно-орієнтоване програмування та моделювання програмних систем : навч. посіб. / О. В. Грудочка. – Дніпро : ДНУ, 2022. – 188 с.

32. The PHP Group. PHP: Hypertext Preprocessor [Електронний ресурс] // Офіційна документація PHP. – Режим доступу: https://www.php.net/manual/en/

33. Zandstra M. PHP 8 Objects, Patterns, and Practice / Matt Zandstra. – 6th ed. – Berkeley : Apress, – 594, 2021. p.

34. MDN Web Docs. Using Fetch API [Електронний ресурс] // Mozilla Developer Network – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API

35. Chart.js Contributors. Chart.js Documentation: Simple HTML5 Charts using the canvas tag [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/docs/latest/

36. Schwartz B. High Performance MySQL: Optimization, Backups, and Replication / B. Schwartz, P. Zaitsev, V. Tkachenko. – 3rd ed. – Sebastopol : O'Reilly Media, 2012 p.

37. Apache Software Foundation. Apache HTTP Server Version 2.4 Documentation: Getting Started [Електронний ресурс]. – Режим доступу: https://httpd.apache.org/docs/current/en/getting-started.html

38. phpMyAdmin Contributors. phpMyAdmin: Bringing MySQL to the web [Електронний ресурс]. – Режим доступу: https://www.phpmyadmin.net/

39. Microsoft. Visual Studio Code Documentation [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs

40. Желібо Є.П. Безпека життєдіяльності : підручник / В. В. Зацарний. Київ : Каравела, 2023. 344 с.

41. Запорожець О.І. Безпека життєдіяльності. Підручник, 2-е видання, Центр учбової літератури, 2020. 448 с.

42. Яремко З.М. Безпека життєдіяльності : навч. посіб. Львів: Видавничий центр ЛНУ ім. Ів. Франка, 2017. 301 с.

43. ДСТУ ISO 9241-9:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 9. Вимоги до неклавіатурних пристроїв уведення.

44. Жидецький В.Ц. Охорона праці користувачів комп’ютерів : підручник. Львів : Афіша, 2020. 176 с.

ДОДАТКИ

ДОДАТОК А

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет
Чернівецький національний університет
імені Юрія Федьковича
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

IX МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2026 р.

(збірник тез конференції)

Тернопіль 2026

УДК 004.41
Чорнописький Б. -ст. гр. СПс-41
Тернопільський національний технічний університет імені Івана Пулюя

ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТА ВИБОРУ ІТ-РІШЕНЬ

Науковий керівник: к.т.н., доцент Цуприк Г. Б.

Chornopyskyi B.
Ternopil Ivan Puluj National Technical University

SOFTWARE IMPLEMENTATION OF AN INFORMATION SYSTEM FOR ANALYSIS AND SELECTION OF IT SOLUTIONS

Supervisor: PhD, Associate Professor Tsupryk H. B.

Ключові слова: комплексний аналіз, програмне забезпечення, веб-система.
Keywords: comprehensive analysis, software, web system.

Актуальність теми дослідження зумовлена стрімкими темпами постійного розширенням ринку програмного забезпечення (ПЗ). Перед ІТ-менеджерами та керівниками підприємств постає проблема вибору оптимальних програмних рішень серед десятків аналогів. Відповідно, виникає потреба у розробці спеціалізованого інструментарію, який дозволив би автоматизувати процес порівняння існуючого ПЗ на основі об'єктивних техніко-економічних метрик.

Метою роботи є розробка та автоматизованої веб-орієнтованої системи, здатної здійснювати попарний порівняльний аналіз програмних рішень, а також формувати комплексний аналітичний висновок для підтримки прийняття управлінських рішень.

Для досягнення поставленої мети було спроєктовано архітектуру системи типу клієнт-сервер. Серверна частина реалізована з використанням мови програмування PHP та реляційної бази даних MySQL. Для забезпечення максимального рівня безпеки та захисту від SQL-ін'єкцій взаємодія з базою даних побудована на основі об'єкта доступу до даних PDO (PHP Data Objects) із вимкненою емуляцією підготовлених запитів (EMULATE_PREPARES = false). Це дозволяє системі швидко та безпечно обробляти великі масиви інформації про характеристики програмних продуктів.

Клієнтська частина розроблена з використанням сучасних стандартів HTML5, CSS3 (з використанням технології Flexbox для адаптивного дизайну) та мови JavaScript. Особливістю інтерфейсу є реалізація динамічної взаємодії користувача з системою без перезавантаження веб-сторінок за рахунок використання асинхронних AJAX-запитів (через Fetch API). Процес вибору ПЗ розбито на інтуїтивно зрозумілі кроки: вибір категорії, після чого система миттєво підвантажує відповідні списки доступних додатків із їхніми графічними ідентифікаторами (логотипами).

Основним науково-практичним завданням системи є проведення комплексного аналізу. Технічний блок аналізу включає зіставлення мінімальних та рекомендованих вимог до апаратного забезпечення Економічний блок дозволяє комплексно оцінити фінансове навантаження при впровадженні ПЗ, зіставляючи базову вартість ліцензії, а також ціну спеціалізованих бізнес-версій для корпоративного сектору.

Для підвищення сприйняття інформації особою, що приймає рішення, у системі реалізовано модуль інтерактивної візуалізації даних. За допомогою JavaScript-

бібліотеки Chart.js математичні показники програм перетворюються на наочні гістограми та діаграми.

Важливим елементом розробленої системи є модуль автоматичної генерації висновків. Система використовує закладений алгоритм зважування показників. Аналізуючи різницю в ціні, апаратних, алгоритм формує текстовий рекомендаційний висновок, вказуючи на найбільш збалансований продукт.

Висновки. У результаті виконання роботи створено повнофункціональну автоматизовану систему аналізу та порівняння наявного ПЗ. Впровадження даної системи на підприємствах малого та середнього бізнесу дозволить скоротити час на проведення IT-аудиту ринку, мінімізувати ризики несумісності ПЗ з існуючою технічною базою та оптимізувати бюджетні витрати на закупівлю ліцензій. Розроблений програмний продукт має високий потенціал до масштабування шляхом додавання нових критеріїв оцінки.

Посилання на літературу:

1. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/

2. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI: https://doi.org/10.32782/IT/2025-2-13

3. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes, 82(2), 360–364. https://doi.org/10.31891/2219-9365-2025-82-52

ДОДАТОК Б

Код програми

Код файлу index.php.

```
 <!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Система порівняльного аналізу ПЗ</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700;800&display
=swap" rel="stylesheet">

    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="admin.css">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</head>
<body>
    <header>
        <div class="container header-main">
            <div class="header-main-text">
                <h1>Аналіз та порівняння програмних рішень</h1>
                <p>Оберіть категорію та два додатки для порівняння їх техніко-
економічних показників</p>
            </div>
            <a href="admin_login.php" class="btn-admin-header">
                <span class="btn-admin-header-icon"></span>
                <span>Адмін-<br>панель</span>
            </a>
        </div>
    </header>
    <main class="container">
        <section class="selection-panel">
            <form id="compareForm">
                <div class="form-group">
                    <label   for="categorySelect">Крок   1:   Оберіть   категорію
ПЗ</label>
                    <select id="categorySelect" name="category_id" required>
                        <option value="" disabled selected>-- Виберіть категорію -
-</option>
                        <option value="1">Офісне ПЗ</option>
                        <option value="2">Інструменти розробки ПЗ</option>
                        <option value="3">Бухгалтерські системи</option>
                        <option value="4">Графічні редактори</option>
                        <option value="5">CAD / Інженерне ПЗ</option>
                    </select>
                </div>
                <div class="form-row">
                    <div class="form-group half-width">
                        <label for="software1Select">Крок 2: Перша програма</label>
                        <div class="select-with-logo">
                            <img  id="logo1"  src=""  alt=""  class="dropdown-logo
hidden">
                            <select    id="software1Select"    name="software1_id"
disabled required>
```

```
                                <option value="" disabled selected>-- Спочатку оберіть категорію --</option>
                            </select>
                        </div>
                    </div>
                    <div class="form-group half-width">
                        <label for="software2Select">Крок 3: Друга програма</label>
                        <div class="select-with-logo">
                        <img id="logo2" src="" alt="" class="dropdown-logo hidden">
                <select id="software2Select" name="software2_id" disabled required>
      <option value="" disabled selected>-- Спочатку оберіть категорію --</option>
                            </select>
                        </div>
                    </div>
                </div>

                <button type="submit" id="compareBtn" class="btn-submit" disabled>Порівняти ПЗ</button>
            </form>
        </section>

        <section id="resultsPanel" class="results-panel hidden">
            <h2>Результати порівняння</h2>

            <div id="analysisSummary" class="analysis-summary"></div>

            <div class="table-responsive">
                <table id="comparisonTable" class="compare-table"></table>
            </div>

            <div class="charts-grid">
                <div class="chart-container">
                    <canvas id="hardwareChart" height="250"></canvas>
                </div>
                <div class="chart-container">
                    <canvas id="priceChart" height="250"></canvas>
                </div>
                <div class="chart-container">
                    <canvas id="ratingChart" height="250"></canvas>
                </div>
                <div class="chart-container">
                    <canvas id="learningChart" height="250"></canvas>
                </div>
            </div>
        </section>
    </main>

    <footer>
        <div class="container">
            <p style="color: #fff;">&copy; 2026 Розробка автоматизованої системи порівняльного аналізу ПЗ. Кваліфікаційна робота.</p>
        </div>
    </footer>
</body>
</html>
```

Код файлу script.js.

```
document.addEventListener('DOMContentLoaded', function() {
    const categorySelect = document.getElementById('categorySelect');
    const software1Select = document.getElementById('software1Select');
    const software2Select = document.getElementById('software2Select');
    const compareBtn = document.getElementById('compareBtn');
```

```
    function checkSelections() {
        const val1 = software1Select.value;
        const val2 = software2Select.value;

        if (val1 && val2 && val1 !== val2) {
            compareBtn.disabled = false;
            compareBtn.title = '';
        } else if (val1 && val2 && val1 === val2) {
            compareBtn.disabled = true;
            compareBtn.title = '⚠ Оберіть дві різні програми';
        } else {
            compareBtn.disabled = true;
            compareBtn.title = '⚠ Виберіть обидві програми';
        }
    }
    categorySelect.addEventListener('change', function() {
        const categoryId = this.value;

        software1Select.disabled = true;
        software2Select.disabled = true;
        compareBtn.disabled = true;
                  software1Select.innerHTML    =    '<option    value=""    disabled
selected>Завантаження...</option>';
                  software2Select.innerHTML    =    '<option    value=""    disabled
selected>Завантаження...</option>';
        fetch(`get_software.php?category_id=${categoryId}`)
            .then(response => response.json())
            .then(data => {
                 let optionsHtml = '<option value="" disabled selected>-- Оберіть
програму --</option>';

                data.forEach(item => {
                              optionsHtml  +=  `<option  value="${item.id}"  data-
logo="${item.logo_url}">${item.name}</option>`;
                });

                software1Select.innerHTML = optionsHtml;
                software2Select.innerHTML = optionsHtml;

                software1Select.disabled = false;
                software2Select.disabled = false;
            })
            .catch(error => {
                console.error('Помилка завантаження даних:', error);
                   software1Select.innerHTML = '<option value="" disabled>Помилка
завантаження</option>';
                  software2Select.innerHTML = '<option value="" disabled>Помилка
завантаження</option>';
            });
    });

    function updateDropdownLogo(selectElement, logoImgId) {
        const selectedOption = selectElement.options[selectElement.selectedIndex];
        const logoUrl = selectedOption.getAttribute('data-logo');
        const logoImg = document.getElementById(logoImgId);

        if (logoUrl && logoUrl !== 'null' && logoUrl !== '') {
            logoImg.src = logoUrl;
            logoImg.classList.remove('hidden');
        } else {
            logoImg.classList.add('hidden');
        }
```

```
}

software1Select.addEventListener('change', function() {
    checkSelections();
    updateDropdownLogo(this, 'logo1');
});

software2Select.addEventListener('change', function() {
    checkSelections();
    updateDropdownLogo(this, 'logo2');
});

const compareForm = document.getElementById('compareForm');
const resultsPanel = document.getElementById('resultsPanel');
const comparisonTable = document.getElementById('comparisonTable');

let hardwareChart = null;
let priceChart = null;
let ratingChart = null;
let learningChart = null;

// ── ОБРОБКА ЗМІНИ РОЗМІРУ ВІКНА ──────────────────────
let resizeTimeout;
window.addEventListener('resize', function() {
    clearTimeout(resizeTimeout);
    resizeTimeout = setTimeout(function() {
        if (hardwareChart || priceChart || ratingChart || learningChart) {
            // Перебудовуємо діаграми при зміні розміру
            if (hardwareChart) hardwareChart.resize();
            if (priceChart) priceChart.resize();
            if (ratingChart) ratingChart.resize();
            if (learningChart) learningChart.resize();
        }
    }, 250);
});

compareForm.addEventListener('submit', function(e) {
    e.preventDefault();

    const id1 = software1Select.value;
    const id2 = software2Select.value;

    const originalBtnText = compareBtn.innerText;
    compareBtn.innerText = 'Аналізую...';
    compareBtn.disabled = true;

    fetch(`compare.php?id1=${id1}&id2=${id2}`)
        .then(response => response.json())
        .then(data => {
            if (data.error) {
                alert('Помилка: ' + data.error);
                return;
            }

            renderResults(data.software1, data.software2);

            resultsPanel.classList.remove('hidden');
            compareBtn.innerText = originalBtnText;
            checkSelections();
            resultsPanel.scrollIntoView({ behavior: 'smooth' });
        })
        .catch(error => {
            console.error('Помилка порівняння:', error);
```

```
                compareBtn.innerText = originalBtnText;
                checkSelections();
            });
    });

    function getChartLayout() {
        const width = window.innerWidth;
        if (width >= 1200) {
            return { columns: 2, maxHeight: 400 };
        } else if (width >= 768) {
            return { columns: 2, maxHeight: 350 };
        } else if (width >= 480) {
            return { columns: 1, maxHeight: 300 };
        } else {
            return { columns: 1, maxHeight: 250 };
        }
    }

    function renderResults(sw1, sw2) {
        // Форматування даних
         const trial1 = sw1.trial_available ? `✅ Так (${sw1.trial_days} днів)` :
'❌ Ні';
         const trial2 = sw2.trial_available ? `✅ Так (${sw2.trial_days} днів)` :
'❌ Ні';

            const priceBase1 = sw1.price_model === 'free' ? 'Безкоштовно' :
`$${sw1.price_base}`;
            const priceBase2 = sw2.price_model === 'free' ? 'Безкоштовно' :
`$${sw2.price_base}`;
        const priceBus1 = (sw1.price_business > 0) ? `$${sw1.price_business}` : '-
';
        const priceBus2 = (sw2.price_business > 0) ? `$${sw2.price_business}` : '-
';

        const ram1 = parseFloat(sw1.min_ram_gb) || 0;
        const ram2 = parseFloat(sw2.min_ram_gb) || 0;
        const storage1 = parseFloat(sw1.storage_gb) || 0;
        const storage2 = parseFloat(sw2.storage_gb) || 0;
        const learn1 = parseFloat(sw1.learning_time_h) || 0;
        const learn2 = parseFloat(sw2.learning_time_h) || 0;

               const numPriceBase1 = sw1.price_model === 'free' ? 0 :
parseFloat(sw1.price_base) || 0;
               const numPriceBase2 = sw2.price_model === 'free' ? 0 :
parseFloat(sw2.price_base) || 0;
        const numPriceBus1 = parseFloat(sw1.price_business) || 0;
        const numPriceBus2 = parseFloat(sw2.price_business) || 0;

        const trialDays1 = sw1.trial_available ? parseInt(sw1.trial_days) || 0 : 0;
        const trialDays2 = sw2.trial_available ? parseInt(sw2.trial_days) || 0 : 0;

        const formatsCount1 = sw1.data_formats ? sw1.data_formats.split(',').length
: 0;
        const formatsCount2 = sw2.data_formats ? sw2.data_formats.split(',').length
: 0;

        const rating1 = parseFloat(sw1.user_rating) || 0;
        const rating2 = parseFloat(sw2.user_rating) || 0;

        function getWinClass(val1, val2, lowerIsBetter = true) {
            if (val1 === val2) return '';
            if (lowerIsBetter) {
```

```
                return val1 < val2 ? 'winner-cell' : '';
            } else {
                return val1 > val2 ? 'winner-cell' : '';
            }
        }

        let classBus1 = '', classBus2 = '';
        if (numPriceBus1 > 0 && numPriceBus2 > 0) {
            classBus1 = getWinClass(numPriceBus1, numPriceBus2, true);
            classBus2 = getWinClass(numPriceBus2, numPriceBus1, true);
        }

        const tableHTML = `
            <tr>
                <th>Показник</th>
                <th>
                    <div class="table-header-flex">
                                <img  src="${sw1.logo_url}"  alt="Лого  ${sw1.name}"
class="table-logo" onerror="this.style.display='none'">
                        <div class="table-header-info">
                            <span class="software-name">${sw1.name}</span>
                            <div class="rating-container">
                              <div class="stars" style="--rating: ${sw1.user_rating
|| 0};"></div>
                                  <span class="rating-number">${sw1.user_rating ||
0}/5</span>
                            </div>
                        </div>
                    </div>
                </th>
                <th>
                    <div class="table-header-flex">
                                <img  src="${sw2.logo_url}"  alt="Лого  ${sw2.name}"
class="table-logo" onerror="this.style.display='none'">
                        <div class="table-header-info">
                            <span class="software-name">${sw2.name}</span>
                            <div class="rating-container">
                              <div class="stars" style="--rating: ${sw2.user_rating
|| 0};"></div>
                                  <span class="rating-number">${sw2.user_rating ||
0}/5</span>
                            </div>
                        </div>
                    </div>
                </th>
            </tr>
            <tr>
                <td><strong>Мінімальний процесор</strong></td>
                <td>${sw1.min_cpu}</td>
                <td>${sw2.min_cpu}</td>
            </tr>
            <tr>
                <td><strong>Мінімальна ОЗП (ГБ)</strong></td>
                <td class="${getWinClass(ram1, ram2, true)}">${sw1.min_ram_gb}</td>
                <td class="${getWinClass(ram2, ram1, true)}">${sw2.min_ram_gb}</td>
            </tr>
            <tr>
                <td><strong>Місце на диску (ГБ)</strong></td>
                                <td   class="${getWinClass(storage1,   storage2,
true)}">${sw1.storage_gb}</td>
                                <td   class="${getWinClass(storage2,   storage1,
true)}">${sw2.storage_gb}</td>
            </tr>
```

```
            <tr>
                <td><strong>Підтримувані ОС</strong></td>
                <td>${sw1.supported_os}</td>
                <td>${sw2.supported_os}</td>
            </tr>
            <tr>
                <td><strong>Формати даних</strong></td>
                            <td  class="${getWinClass(formatsCount1,  formatsCount2,
false)}">${sw1.data_formats}</td>
                            <td  class="${getWinClass(formatsCount2,  formatsCount1,
false)}">${sw2.data_formats}</td>
            </tr>
            <tr>
                <td><strong>Модель оплати</strong></td>
                <td>${sw1.price_model}</td>
                <td>${sw2.price_model}</td>
            </tr>
            <tr>
                <td><strong>Базова ціна</strong></td>
                            <td  class="${getWinClass(numPriceBase1,  numPriceBase2,
true)}">${priceBase1}</td>
                            <td  class="${getWinClass(numPriceBase2,  numPriceBase1,
true)}">${priceBase2}</td>
            </tr>
            <tr>
                <td><strong>Бізнес-ліцензія</strong></td>
                <td class="${classBus1}">${priceBus1}</td>
                <td class="${classBus2}">${priceBus2}</td>
            </tr>
            <tr>
                <td><strong>Час на освоєння (год)</strong></td>
                                    <td    class="${getWinClass(learn1,    learn2,
true)}">${sw1.learning_time_h}</td>
                                    <td    class="${getWinClass(learn2,    learn1,
true)}">${sw2.learning_time_h}</td>
            </tr>
            <tr>
                <td><strong>Пробна версія (дні)</strong></td>
                                <td   class="${getWinClass(trialDays1,   trialDays2,
false)}">${trial1}</td>
                                <td   class="${getWinClass(trialDays2,   trialDays1,
false)}">${trial2}</td>
            </tr>
            <tr>
                <td><strong>Офіційний сайт</strong></td>
                      <td><a  href="${sw1.website_url}"  target="_blank">Перейти  на
сайт</a></td>
                      <td><a  href="${sw2.website_url}"  target="_blank">Перейти  на
сайт</a></td>
            </tr>
        `;
        comparisonTable.innerHTML = tableHTML;

        // === ГЕНЕРУЄМО РОЗУМНИЙ ТЕКСТОВИЙ ВИСНОВОК ===
        const analysisSummary = document.getElementById('analysisSummary');
          let analysisText = "<h3 style='margin-bottom: 10px; color: #2c3e50;'>
Експертний висновок:</h3><ul class='analysis-list'>";

        // 1. АНАЛІЗ ЦІНИ
        let priceText = "";
        if (numPriceBase1 === 0 && numPriceBase2 === 0) {
             priceText = "Обидві програми є абсолютно безкоштовними. Ваш вибір має
базуватися виключно на зручності інтерфейсу та необхідних інструментах.";
```

```
        } else if (numPriceBase1 === 0) {
            priceText = `<strong>${sw1.name}</strong> є кращим вибором для економії бюджету, оскільки програма безкоштовна. Натомість за <strong>${sw2.name}</strong> доведеться заплатити $${numPriceBase2}.`;
        } else if (numPriceBase2 === 0) {
            priceText = `<strong>${sw2.name}</strong> є кращим вибором для економії бюджету, оскільки програма безкоштовна. Натомість за <strong>${sw1.name}</strong> доведеться заплатити $${numPriceBase1}.`;
        } else {
            const model1 = sw1.price_model.toLowerCase();
            const model2 = sw2.price_model.toLowerCase();

                        const isSub1 = model1.includes('subscription') || model1.includes('підписк') || model1.includes('місяц');
             const isOneTime1 = model1.includes('paid') || model1.includes('разов') || model1.includes('назавжди');
                        const isSub2 = model2.includes('subscription') || model2.includes('підписк') || model2.includes('місяц');
             const isOneTime2 = model2.includes('paid') || model2.includes('разов') || model2.includes('назавжди');

            if (isSub1 && isOneTime2 && numPriceBase1 > 0) {
                let months = Math.ceil(numPriceBase2 / numPriceBase1);
                priceText = `На перший погляд <strong>${sw1.name}</strong> здається дешевшою ($${numPriceBase1}), але це підписка. Разова покупка ліцензії <strong>${sw2.name}</strong> за $${numPriceBase2} повністю окупиться приблизно через <strong>${months} місяців</strong> використання.`;
            } else if (isSub2 && isOneTime1 && numPriceBase2 > 0) {
                let months = Math.ceil(numPriceBase1 / numPriceBase2);
                priceText = `На перший погляд <strong>${sw2.name}</strong> здається дешевшою ($${numPriceBase2}), але це підписка. Разова покупка ліцензії <strong>${sw1.name}</strong> за $${numPriceBase1} повністю окупиться приблизно через <strong>${months} місяців</strong> використання.`;
            } else {
                if (numPriceBase1 < numPriceBase2) {
                        priceText = `<strong>${sw1.name}</strong> є вигіднішою ($${numPriceBase1}) порівняно з <strong>${sw2.name}</strong> ($${numPriceBase2}), що дозволить зекономити кошти.`;
                } else if (numPriceBase2 < numPriceBase1) {
                        priceText = `<strong>${sw2.name}</strong> є вигіднішою ($${numPriceBase2}) порівняно з <strong>${sw1.name}</strong> ($${numPriceBase1}), що дозволить зекономити кошти.`;
                } else {
                     priceText = `Обидві програми знаходяться в однаковій ціновій категорії ($${numPriceBase1}).`;
                }
            }
        }
        analysisText += `<li><strong> Вартість:</strong> ${priceText}</li>`;

        // 2. АНАЛІЗ ОС ТА УНІВЕРСАЛЬНОСТІ
        const os1 = (sw1.supported_os || '').toLowerCase();
        const os2 = (sw2.supported_os || '').toLowerCase();
             const isWeb1 = os1.includes('web') || os1.includes('веб') || os1.includes('браузер');
             const isWeb2 = os2.includes('web') || os2.includes('веб') || os2.includes('браузер');

        let osText = "";
        if (isWeb1 && !isWeb2) {
             osText = `<strong>${sw1.name}</strong> є більш універсальним рішенням щодо платформ, оскільки має Web-версію (дозволяє працювати через браузер на будь-якій ОС без прив'язки до комп'ютера).`;
```

```
        } else if (!isWeb1 && isWeb2) {
            osText = `<strong>${sw2.name}</strong> є більш універсальним рішенням щодо платформ, оскільки має Web-версію (дозволяє працювати через браузер на будь-якій ОС без прив'язки до комп'ютера).`;
        } else if (isWeb1 && isWeb2) {
            osText = `Обидві програми є максимально мобільними та універсальними, оскільки підтримують роботу через Web-браузер.`;
        } else {
           osText = `Обидві програми вимагають класичного встановлення на комп'ютер. Зверніть увагу на сумісність з вашою поточною операційною системою.`;
        }
            analysisText += `<li><strong> Платформи та доступність:</strong> ${osText}</li>`;

        // 3. АНАЛІЗ ЧАСУ НА ОСВОЄННЯ
        let learnText = "";
        if (learn1 < learn2) {
           learnText = `<strong>${sw1.name}</strong> є значно простішою в опануванні (приблизно ${learn1} год. на вивчення проти ${learn2} год. у конкурента), що дозволить швидше впровадити її в робочий процес вашої команди.`;
        } else if (learn2 < learn1) {
           learnText = `<strong>${sw2.name}</strong> є значно простішою в опануванні (приблизно ${learn2} год. на вивчення проти ${learn1} год. у конкурента), що дозволить швидше впровадити її в робочий процес вашої команди.`;
        } else {
            learnText = `Обидві програми мають приблизно однаковий поріг входження та час на освоєння (${learn1} год.).`;
        }
        analysisText += `<li><strong> Час на освоєння:</strong> ${learnText}</li>`;

        // 4. АНАЛІЗ ФОРМАТІВ ДАНИХ
        let formatsText = "";
        if (formatsCount1 > formatsCount2) {
              formatsText = `Щодо інтеграції та сумісності беззаперечно виграє <strong>${sw1.name}</strong>, оскільки вона підтримує більшу кількість форматів даних (${formatsCount1} форматів проти ${formatsCount2}).`;
        } else if (formatsCount2 > formatsCount1) {
              formatsText = `Щодо інтеграції та сумісності беззаперечно виграє <strong>${sw2.name}</strong>, оскільки вона підтримує більшу кількість форматів даних (${formatsCount2} форматів проти ${formatsCount1}).`;
        } else {
            formatsText = `Обидві програми підтримують однакову кількість форматів даних (${formatsCount1}), тому мають схожі можливості для інтеграції з іншими системами.`;
        }
       analysisText += `<li><strong> Робота з даними:</strong> ${formatsText}</li>`;

        // 5. АНАЛІЗ РЕЙТИНГУ
        let ratingText = "";
        if (rating1 > rating2) {
             ratingText = `<strong>${sw1.name}</strong> отримала вищу оцінку від користувачів (⭐ ${rating1}/5 проти ⭐ ${rating2}/5), що свідчить про більшу задоволеність користувачів.`;
        } else if (rating2 > rating1) {
             ratingText = `<strong>${sw2.name}</strong> отримала вищу оцінку від користувачів (⭐ ${rating2}/5 проти ⭐ ${rating1}/5), що свідчить про більшу задоволеність користувачів.`;
        } else {
             ratingText = `Обидві програми мають однакові оцінки від користувачів (⭐ ${rating1}/5).`;
        }
```

```
            analysisText += `<li><strong>⭐ Користувацькі оцінки:</strong> ${ratingText}</li>`;

        analysisText += "</ul>";
        analysisSummary.innerHTML = analysisText;

        // Глобальне налаштування шрифту для графіків
        Chart.defaults.font.family = "'Nunito', sans-serif";
        Chart.defaults.color = '#64748b';
        Chart.defaults.responsive = true;
        Chart.defaults.maintainAspectRatio = true;

        // Отримуємо налаштування макету залежно від розміру екрану
        const layout = getChartLayout();

        // Оновлюємо сітку графіків
        const chartsGrid = document.querySelector('.charts-grid');
         chartsGrid.style.gridTemplateColumns = layout.columns === 2 ? 'repeat(2, 1fr)' : '1fr';

        // === ГРАФІК 1: АПАРАТНІ ВИМОГИ ===
        const ctxHW = document.getElementById('hardwareChart').getContext('2d');
        if (hardwareChart) hardwareChart.destroy();

        hardwareChart = new Chart(ctxHW, {
            type: 'bar',
            data: {
                labels: ['ОЗП (ГБ)', 'Місце на диску (ГБ)'],
                datasets: [
                    {
                        label: sw1.name,
                        data: [sw1.min_ram_gb, sw1.storage_gb],
                        backgroundColor: 'rgba(99, 102, 241, 0.85)',
                        borderColor: 'rgba(99, 102, 241, 1)',
                        borderWidth: 1,
                        borderRadius: 6
                    },
                    {
                        label: sw2.name,
                        data: [sw2.min_ram_gb, sw2.storage_gb],
                        backgroundColor: 'rgba(168, 85, 247, 0.85)',
                        borderColor: 'rgba(168, 85, 247, 1)',
                        borderWidth: 1,
                        borderRadius: 6
                    }
                ]
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
                indexAxis: window.innerWidth < 480 ? 'y' : 'x',
                scales: {
                      y: { beginAtZero: true, grid: { color: '#e2e8f0'}, ticks: { font: { size: window.innerWidth < 480 ? 10 : 12 }}},
                       x: { grid: { display: false}, ticks: { font: { size: window.innerWidth < 480 ? 10 : 12 }}}
                },
                plugins: {
                    legend: { labels: { font: { size: window.innerWidth < 480 ? 10 : 12 }, padding: 10 }},
                    title: {
                        display: true,
                        text: 'Апаратні вимоги',
```

```
                        font: { size: window.innerWidth < 480 ? 14 : 18, weight:
'800' },
                        color: '#1e1b4b',
                        padding: { bottom: 15 }
                    }
                }
            }
        });

        // === ГРАФІК 2: ЦІНОВА ПОЛІТИКА ===
        const ctxPrice = document.getElementById('priceChart').getContext('2d');
        if (priceChart) priceChart.destroy();

        priceChart = new Chart(ctxPrice, {
            type: 'bar',
            data: {
                labels: ['Базова ціна ($)', 'Бізнес-ліцензія ($)'],
                datasets: [
                    {
                        label: sw1.name,
                        data: [numPriceBase1, numPriceBus1],
                        backgroundColor: 'rgba(16, 185, 129, 0.85)',
                        borderColor: 'rgba(16, 185, 129, 1)',
                        borderWidth: 1,
                        borderRadius: 6
                    },
                    {
                        label: sw2.name,
                        data: [numPriceBase2, numPriceBus2],
                        backgroundColor: 'rgba(245, 158, 11, 0.85)',
                        borderColor: 'rgba(245, 158, 11, 1)',
                        borderWidth: 1,
                        borderRadius: 6
                    }
                ]
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
                indexAxis: window.innerWidth < 480 ? 'y' : 'x',
                scales: {
                      y: { beginAtZero: true, grid: { color: '#e2e8f0'}, ticks: {
font: { size: window.innerWidth < 480 ? 10 : 12 }}},
                       x: { grid: { display: false}, ticks: { font: { size:
window.innerWidth < 480 ? 10 : 12 }}}
                },
                plugins: {
                    legend: { labels: { font: { size: window.innerWidth < 480 ? 10
: 12 }, padding: 10 }},
                    title: {
                        display: true,
                        text: 'Порівняння вартості',
                         font: { size: window.innerWidth < 480 ? 14 : 18, weight:
'800' },
                        color: '#1e1b4b',
                        padding: { bottom: 15 }
                    }
                }
            }
        });

        // === ГРАФІК 3: РЕЙТИНГ КОРИСТУВАЧІВ (РАДАРНА ДІАГРАМА) ===
        const ctxRating = document.getElementById('ratingChart').getContext('2d');
```

```
        if (ratingChart) ratingChart.destroy();

        ratingChart = new Chart(ctxRating, {
            type: 'radar',
            data: {
                labels: ['Рейтинг'],
                datasets: [
                    {
                        label: sw1.name,
                        data: [rating1],
                        borderColor: 'rgba(99, 102, 241, 1)',
                        backgroundColor: 'rgba(99, 102, 241, 0.2)',
                        borderWidth: 2,
                        pointBackgroundColor: 'rgba(99, 102, 241, 1)',
                        pointBorderColor: '#fff',
                        pointBorderWidth: 2,
                        pointRadius: 6
                    },
                    {
                        label: sw2.name,
                        data: [rating2],
                        borderColor: 'rgba(168, 85, 247, 1)',
                        backgroundColor: 'rgba(168, 85, 247, 0.2)',
                        borderWidth: 2,
                        pointBackgroundColor: 'rgba(168, 85, 247, 1)',
                        pointBorderColor: '#fff',
                        pointBorderWidth: 2,
                        pointRadius: 6
                    }
                ]
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
                scales: {
                    r: {
                        beginAtZero: true,
                        max: 5,
                        ticks: { stepSize: 1, font: { size: window.innerWidth < 480
? 9 : 11 }},
                        grid: { color: '#e2e8f0' }
                    }
                },
                plugins: {
                     legend: { labels: { font: { size: window.innerWidth < 480 ? 10
: 12 }, padding: 10 }},
                    title: {
                        display: true,
                        text: ' Рейтинг серед користувачів',
                          font: { size: window.innerWidth < 480 ? 14 : 18, weight:
'800' },
                        color: '#1e1b4b',
                        padding: { bottom: 15 }
                    }
                }
            }
        });

        // === ГРАФІК 4: ЧАС НА ОСВОЄННЯ (КІЛЬЦЕВА ДІАГРАМА) ===
                                            const          ctxLearning          =
document.getElementById('learningChart').getContext('2d');
        if (learningChart) learningChart.destroy();
```

```
        learningChart = new Chart(ctxLearning, {
            type: 'doughnut',
            data: {
                labels: [sw1.name, sw2.name],
                datasets: [
                    {
                        data: [learn1, learn2],
                        backgroundColor: [
                            'rgba(99, 102, 241, 0.85)',
                            'rgba(168, 85, 247, 0.85)'
                        ],
                        borderColor: [
                            'rgba(99, 102, 241, 1)',
                            'rgba(168, 85, 247, 1)'
                        ],
                        borderWidth: 2,
                        borderRadius: 6
                    }
                ]
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
                plugins: {
                    legend: {
                        position: 'bottom',
                         labels: { font: { size: window.innerWidth < 480 ? 10 : 12
}, padding: 15 }
                    },
                    title: {
                        display: true,
                        text: ' Час на освоєння (годин)',
                          font: { size: window.innerWidth < 480 ? 14 : 18, weight:
'800' },
                        color: '#1e1b4b',
                        padding: { bottom: 15 }
                    },
                    tooltip: {
                        callbacks: {
                            label: function(context) {
                                  return context.label + ': ' + context.parsed + '
год.';
                            }
                        }
                    }
                }
            }
        });
    }
});
```

Код файлу get_software.php.

```
<?php
require_once 'db.php';

header('Content-Type: application/json; charset=utf-8');

// Перевіряє, чи передала БД ID категорії
if (isset($_GET['category_id']) && is_numeric($_GET['category_id'])) {
    $category_id = (int) $_GET['category_id'];

    // Готує безпечний SQL-запит (захист від SQL-ін'єкцій)
```

```
    $stmt = $pdo->prepare("SELECT id, name, logo_url FROM software WHERE category_id
= :category_id ORDER BY name ASC");
    $stmt->execute(['category_id' => $category_id]);

    // Отримує всі знайдені програми
    $software = $stmt->fetchAll();

    // Віддає результат у форматі JSON
    echo json_encode($software);
} else {
    // Якщо ID не передали або він некоректний, повертаємо порожній масив
    echo json_encode([]);
}
?>
```

Код файлу db.php.

```
<?php
$host = 'localhost';
$db   = 'software_compare';
$user = 'root';
$pass = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
     PDO::ATTR_ERRMODE              => PDO::ERRMODE_EXCEPTION, // Увімкнути показ
помилок
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,       // Дані повертатимуться
як асоціативний масив
    PDO::ATTR_EMULATE_PREPARES   => false,                  // Підвищує безпеку
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    die("Помилка підключення до БД: " . $e->getMessage());
}
?>
```

Код файлу compare.php.

```
<?php
// Підключаємо файл бази даних
require_once 'db.php';
header('Content-Type: application/json; charset=utf-8');

// Перевіряємо, чи отримали ми ID обох програм
if (isset($_GET['id1']) && isset($_GET['id2'])) {
    $id1 = (int)$_GET['id1'];
    $id2 = (int)$_GET['id2'];

    // Робимо запит до БД: шукаємо програми, де ID дорівнює id1 або id2
    $stmt = $pdo->prepare("SELECT * FROM software WHERE id IN (?, ?)");
    $stmt->execute([$id1, $id2]);
    $results = $stmt->fetchAll();

    $data = [];
    // Розподіляємо результати, щоб знати, де перша програма, а де друга
    foreach ($results as $row) {
        if ($row['id'] == $id1) {
            $data['software1'] = $row;
        } elseif ($row['id'] == $id2) {
```

```
            $data['software2'] = $row;
        }
    }

    // Повертаємо дані для JS
    echo json_encode($data);
} else {
    // Якщо щось пішло не так
    echo json_encode(['error' => 'Не передано ID програм для порівняння']);
}
?>
```

Код файлу admin.php.

```
<?php
session_start();

// Перевіряє авторизацію
if (empty($_SESSION['admin_logged_in'])) {
    header('Location: admin_login.php');
    exit;
}

// Верифікуємо сесію
$current_ip = $_SERVER['REMOTE_ADDR'];
$current_user_agent = $_SERVER['HTTP_USER_AGENT'] ?? '';

// Перевіряємо IP
if (isset($_SESSION['admin_ip']) && $_SESSION['admin_ip'] !== $current_ip) {
    session_destroy();
    header('Location: admin_login.php?error=session_hijack');
    exit;
}

if  (isset($_SESSION['admin_user_agent'])  &&  $_SESSION['admin_user_agent']  !==
$current_user_agent) {
    session_destroy();
    header('Location: admin_login.php?error=session_hijack');
    exit;
}

if (isset($_SESSION['admin_time']) && (time() - $_SESSION['admin_time']) > 7200) {
    session_destroy();
    header('Location: admin_login.php?error=timeout');
    exit;
}

$_SESSION['admin_time'] = time();

// Логування дій адміна
function logAdminAction($action, $details = '') {
    $log_file = __DIR__ . '/logs/admin_actions.log';
    if (!is_dir(__DIR__ . '/logs')) {
        mkdir(__DIR__ . '/logs', 0700, true);
    }

    $message = date('[Y-m-d H:i:s]') . " [$action] $details" . PHP_EOL;
    file_put_contents($log_file, $message, FILE_APPEND);
}

// Логуємо вхід в панель
if (!isset($_SESSION['action_logged'])) {
    logAdminAction('PANEL_ACCESS', "IP: {$_SERVER['REMOTE_ADDR']}");
```

```
    $_SESSION['action_logged'] = true;
}
?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Панель адміністратора — Каталог ПЗ</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
                                                                    <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700;800&display
=swap" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="admin.css">
</head>
<body>

<header>
    <div class="container header-admin-flex">
        <div>
            <h1>Панель адміністратора</h1>
            <p>Керування каталогом програмного забезпечення</p>
        </div>
        <div class="header-admin-actions">
            <a href="index.php" class="btn-back">← Головна</a>
            <a href="admin_logout.php" class="btn-logout">Вийти</a>
        </div>
    </div>
</header>

<main class="container">

<?php
require_once 'db.php';

$message     = '';
$messageType = '';

$DB_FIELDS = [
    'category_id', 'name', 'logo_url', 'website_url', 'min_cpu',
    'min_ram_gb', 'storage_gb', 'supported_os', 'data_formats',
    'price_model', 'price_base', 'price_business',
    'trial_available', 'trial_days', 'learning_time_h', 'user_rating'
];

// ВИДАЛЕННЯ
if (isset($_POST['action']) && $_POST['action'] === 'delete') {
    $id = (int)$_POST['id'];
    try {
        $pdo->prepare("DELETE FROM software WHERE id = ?")->execute([$id]);
        $message     = 'Програму успішно видалено з каталогу.';
        $messageType = 'success';
    } catch (PDOException $e) {
        $message     = 'Помилка видалення: ' . $e->getMessage();
        $messageType = 'error';
    }
}

// ДОДАВАННЯ
if (isset($_POST['action']) && $_POST['action'] === 'add') {
    $values = collectFormValues($DB_FIELDS);
```

```
    try {
        $cols = implode(', ', $DB_FIELDS);
        $placeholders = implode(', ', array_map(fn($f) => ":$f", $DB_FIELDS));
          $pdo->prepare("INSERT INTO software ($cols) VALUES ($placeholders)")-
>execute($values);
        $message     = "Програму «{$values['name']}» успішно додано до каталогу.";
        $messageType = 'success';
    } catch (PDOException $e) {
        $message     = 'Помилка додавання: ' . $e->getMessage();
        $messageType = 'error';
    }
}

// РЕДАГУВАННЯ
if (isset($_POST['action']) && $_POST['action'] === 'edit') {
    $id     = (int)$_POST['id'];
    $values = collectFormValues($DB_FIELDS);
    $values['id'] = $id;
    try {
        $sets = implode(', ', array_map(fn($f) => "$f = :$f", $DB_FIELDS));
       $pdo->prepare("UPDATE software SET $sets WHERE id = :id")->execute($values);
        $message     = 'Дані програми успішно оновлено.';
        $messageType = 'success';
    } catch (PDOException $e) {
        $message     = 'Помилка оновлення: ' . $e->getMessage();
        $messageType = 'error';
    }
}

// Збирає значення полів з POST
function collectFormValues(array $fields): array {
    $v = [];
    foreach ($fields as $f) {
        $v[$f] = isset($_POST[$f]) && $_POST[$f] !== '' ? trim($_POST[$f]) : null;
    }
    $v['trial_available'] = !empty($_POST['trial_available']) ? 1 : 0;
    return $v;
}

if ($message): ?>
    <div class="admin-message admin-message--<?= $messageType ?>">
        <?= htmlspecialchars($message) ?>
    </div>
<?php endif;

$allSoftware = $pdo->query("SELECT * FROM software ORDER BY category_id, name ASC")-
>fetchAll();

$categoryNames = [
    1 => 'Офісне ПЗ',
    2 => 'Інструменти розробки',
    3 => 'Бухгалтерські системи',
    4 => 'Графічні редактори',
    5 => 'CAD / Інженерне ПЗ',
];
?>

<!-- ФОРМА ДОДАВАННЯ -->
<section class="selection-panel admin-section">
    <div class="admin-section-header" onclick="toggleSection('addForm')">
        <h2>➕ Додати нову програму</h2>
        <span class="toggle-icon rotated" id="addForm-icon">▼</span>
    </div>
```

```
    <div id="addForm" class="admin-section-body collapsed">
        <form method="POST" action="admin.php">
            <input type="hidden" name="action" value="add">
            <?php renderSoftwareForm('', $categoryNames); ?>
              <button type="submit" class="btn-submit" style="margin-top:10px;">✅
Додати програму</button>
        </form>
    </div>
</section>

<!-- КАТАЛОГ -->
<section class="selection-panel admin-section">
    <div class="admin-section-header" style="cursor:default;">
        <h2>📋 Каталог програм
            <span class="catalog-count"><?= count($allSoftware) ?> записів</span>
        </h2>
         <select id="filterCategory" onchange="filterTable()" class="admin-filter-
select">
            <option value="">Усі категорії</option>
            <?php foreach ($categoryNames as $id => $name): ?>
                <option value="<?= $id ?>"><?= $name ?></option>
            <?php endforeach; ?>
        </select>
    </div>

    <div class="table-responsive">
        <table class="compare-table admin-catalog-table" id="catalogTable">
            <thead>
                <tr>
                    <th>Лого</th>
                    <th>Назва</th>
                    <th>Категорія</th>
                    <th>Мін. CPU</th>
                    <th>ОЗП / Диск</th>
                    <th>Ціна (база)</th>
                    <th>Рейтинг</th>
                    <th>Дії</th>
                </tr>
            </thead>
            <tbody>
            <?php foreach ($allSoftware as $sw): ?>
                <tr data-category="<?= $sw['category_id'] ?>">
                    <td>
                        <?php if (!empty($sw['logo_url'])): ?>
                              <img src="<?= htmlspecialchars($sw['logo_url']) ?>"
alt="logo" class="table-logo">
                        <?php else: ?>
                            <span class="no-logo">—</span>
                        <?php endif; ?>
                    </td>
                    <td>
                        <strong><?= htmlspecialchars($sw['name']) ?></strong>
                        <?php if (!empty($sw['website_url'])): ?>
                             <br><a href="<?= htmlspecialchars($sw['website_url'])
?>" target="_blank" class="table-link">🌐 сайт</a>
                        <?php endif; ?>
                    </td>
                     <td><?= htmlspecialchars($categoryNames[$sw['category_id']] ??
'—') ?></td>
                    <td><?= htmlspecialchars($sw['min_cpu'] ?? '—') ?></td>
                    <td>
```

```
                        <span class="hw-badge"><?= $sw['min_ram_gb'] ?? '?' ?> ГБ
ОЗП</span><br>
                        <span class="hw-badge"><?= $sw['storage_gb'] ?? '?' ?>
ГБ</span>
                    </td>
                    <td>
                        <?php if ($sw['price_model'] === 'free'): ?>
                            <span class="badge-free">Безкоштовно</span>
                        <?php else: ?>
                            $<?= htmlspecialchars($sw['price_base'] ?? '0') ?>
                                <br><small style="color:var(--text-muted)"><?=
htmlspecialchars($sw['price_model']) ?></small>
                        <?php endif; ?>
                    </td>
                    <td>
                        <?php if (!empty($sw['user_rating']) && $sw['user_rating']
> 0): ?>
                                                <span class="rating-badge">⭐ <?=
htmlspecialchars($sw['user_rating']) ?></span>
                        <?php else: ?>—<?php endif; ?>
                    </td>
                    <td class="actions-cell">
                        <button class="btn-action btn-edit"
                          onclick='openEditModal(<?= json_encode($sw, JSON_HEX_APOS
| JSON_HEX_QUOT) ?>)'>
                            ✏️ Редагувати
                        </button>
                        <form method="POST" action="admin.php" style="display:inline;"
                                onsubmit="return confirm('Видалити «<?=
htmlspecialchars($sw['name'], ENT_QUOTES) ?>»?')">
                            <input type="hidden" name="action" value="delete">
                            <input type="hidden" name="id" value="<?= $sw['id'] ?>">
                            <button type="submit" class="btn-action btn-delete">🗑
Видалити</button>
                        </form>
                    </td>
                </tr>
            <?php endforeach; ?>
            </tbody>
        </table>
    </div>
</section>

</main>

<!-- МОДАЛЬНЕ ВІКНО РЕДАГУВАННЯ -->
<div id="editModal" class="modal-overlay hidden">
    <div class="modal-window">
        <div class="modal-header">
            <h2>✏️ Редагування програми</h2>
            <button class="modal-close" onclick="closeEditModal()">✕</button>
        </div>
        <div class="modal-body">
            <form method="POST" action="admin.php">
                <input type="hidden" name="action" value="edit">
                <input type="hidden" name="id" id="edit_id">
                <?php renderSoftwareForm('edit_', $categoryNames); ?>
                <div class="modal-footer">
                    <button type="button" class="btn-action btn-cancel"
onclick="closeEditModal()">Скасувати</button>
                    <button type="submit" class="btn-submit"
style="width:auto;padding:12px 30px;">💾 Зберегти зміни</button>
```

```
                </div>
            </form>
        </div>
    </div>
</div>

<footer>
    <div class="container">
            <p  style="color:#fff;">&copy;  2026  Розробка  автоматизованої  системи
порівняльного аналізу ПЗ. Кваліфікаційна робота.</p>
    </div>
</footer>

<script src="admin.js"></script>
</body>
</html>

<?php
// Функція рендерингу полів форми
function renderSoftwareForm(string $prefix, array $categoryNames): void { ?>
<div class="form-grid-2">

    <div class="form-group">
        <label>Назва програми *</label>
        <input type="text" name="name" id="<?= $prefix ?>name" class="admin-input"
required placeholder="Напр. Microsoft Word">
    </div>

    <div class="form-group">
        <label>Категорія *</label>
          <select  name="category_id"  id="<?=  $prefix  ?>category_id"  class="admin-
input" required>
            <option value="" disabled selected>-- Оберіть --</option>
            <?php foreach ($categoryNames as $id => $catName): ?>
                <option value="<?= $id ?>"><?= $catName ?></option>
            <?php endforeach; ?>
        </select>
    </div>

    <div class="form-group">
        <label>URL логотипу</label>
        <input type="url" name="logo_url" id="<?= $prefix ?>logo_url" class="admin-
input" placeholder="https://...">
    </div>

    <div class="form-group">
        <label>Сайт програми</label>
           <input  type="url"  name="website_url"  id="<?=  $prefix  ?>website_url"
class="admin-input" placeholder="https://...">
    </div>

    <div class="form-group">
        <label>Мінімальний CPU</label>
        <input type="text" name="min_cpu" id="<?= $prefix ?>min_cpu" class="admin-
input" placeholder="Напр. Intel Core i3">
    </div>

    <div class="form-group">
        <label>Підтримувані ОС</label>
          <input  type="text"  name="supported_os"  id="<?=  $prefix  ?>supported_os"
class="admin-input" placeholder="Windows, macOS, Linux">
    </div>
```

```
    <div class="form-group">
        <label>Мін. ОЗП (ГБ)</label>
           <input type="number" name="min_ram_gb" id="<?= $prefix ?>min_ram_gb"
class="admin-input" step="1" min="0" placeholder="4">
    </div>

    <div class="form-group">
        <label>Місце на диску (ГБ)</label>
           <input type="number" name="storage_gb" id="<?= $prefix ?>storage_gb"
class="admin-input" step="0.1" min="0" placeholder="2">
    </div>

    <div class="form-group">
        <label>Модель ціноутворення *</label>
          <select name="price_model" id="<?= $prefix ?>price_model" class="admin-
input" required>
            <option value="" disabled selected>-- Оберіть --</option>
            <option value="free">Безкоштовно (free)</option>
            <option value="paid">Разова покупка (paid)</option>
            <option value="subscription">Підписка (subscription)</option>
        </select>
    </div>

    <div class="form-group">
        <label>Рейтинг користувачів (0–5)</label>
          <input type="number" name="user_rating" id="<?= $prefix ?>user_rating"
class="admin-input" step="0.1" min="0" max="5" placeholder="4.5">
    </div>

    <div class="form-group">
        <label>Базова ціна ($)</label>
           <input type="number" name="price_base" id="<?= $prefix ?>price_base"
class="admin-input" step="0.01" min="0" placeholder="0.00">
    </div>

    <div class="form-group">
        <label>Бізнес-ліцензія ($)</label>
       <input type="number" name="price_business" id="<?= $prefix ?>price_business"
class="admin-input" step="0.01" min="0" placeholder="0.00">
    </div>

    <div class="form-group">
        <label>Час на освоєння (год.)</label>
                <input type="number" name="learning_time_h" id="<?= $prefix
?>learning_time_h" class="admin-input" step="1" min="0" placeholder="20">
    </div>

    <div class="form-group">
        <label>Формати даних</label>
          <input type="text" name="data_formats" id="<?= $prefix ?>data_formats"
class="admin-input" placeholder="docx, pdf, odt">
    </div>

    <div class="form-group" style="grid-column: span 2">
        <label>Пробний період</label>
        <div class="trial-row">
            <label class="admin-checkbox-label">
                    <input type="checkbox" name="trial_available" id="<?= $prefix
?>trial_available" value="1">
                Є пробний період
            </label>
            <div class="trial-days-wrap">
               <input type="number" name="trial_days" id="<?= $prefix ?>trial_days"
```

```
                    class="admin-input trial-days-input" min="0" placeholder="30">
                <span class="trial-days-label">днів</span>
            </div>
        </div>
    </div>

</div>
<?php }
?>
```

Код файлу admin.js.

```
//  admin.js — логіка панелі адміністратора
// ── Розкриття / згортання секції "Додати" ────
function toggleSection(id) {
    const body = document.getElementById(id);
    const icon = document.getElementById(id + '-icon');
    if (!body) return;
    body.classList.toggle('collapsed');
    if (icon) icon.classList.toggle('rotated');
}

// ── Фільтрація таблиці за категорією ──────────
function filterTable() {
    const val  = document.getElementById('filterCategory').value;
    const rows = document.querySelectorAll('#catalogTable tbody tr');
    rows.forEach(row => {
        row.style.display = (!val || row.dataset.category === val) ? '' : 'none';
    });
}

// ── Відкриття модального вікна редагування ────
function openEditModal(sw) {
    document.getElementById('editModal').classList.remove('hidden');
    document.body.style.overflow = 'hidden';

    // Усі реальні колонки таблиці software
    const fields = [
        'id', 'name', 'category_id', 'logo_url', 'website_url',
        'min_cpu', 'min_ram_gb', 'storage_gb', 'supported_os',
        'data_formats', 'price_model', 'price_base', 'price_business',
        'trial_days', 'learning_time_h', 'user_rating'
    ];

    fields.forEach(f => {
        const el = document.getElementById('edit_' + f);
        if (el) el.value = (sw[f] !== null && sw[f] !== undefined) ? sw[f] : '';
    });

    // Чекбокс пробного періоду
    const cb = document.getElementById('edit_trial_available');
    if (cb) cb.checked = (sw['trial_available'] == 1);

    // Вибір категорії — встановлюємо правильний option
    const catSel = document.getElementById('edit_category_id');
    if (catSel && sw['category_id']) catSel.value = sw['category_id'];

    // Вибір price_model
    const pmSel = document.getElementById('edit_price_model');
    if (pmSel && sw['price_model']) pmSel.value = sw['price_model'];
}

// ── Закриття модального вікна ─────────────────
```

```
function closeEditModal() {
    document.getElementById('editModal').classList.add('hidden');
    document.body.style.overflow = '';
}

// Закриття по кліку на затемнений фон
document.addEventListener('DOMContentLoaded', () => {
    const overlay = document.getElementById('editModal');
    if (overlay) overlay.addEventListener('click', e => { if (e.target === overlay)
closeEditModal(); });
});

// Закриття по Escape
document.addEventListener('keydown',   e   =>   {   if   (e.key   ===   'Escape')
closeEditModal(); });
```

Код файлу admin_login.php.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Вхід — Адміністратор</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
                                                                    <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700;800&display
=swap" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="admin.css">
</head>
<body>

<header>
    <div class="container">
        <h1>Панель адміністратора</h1>
        <p>Система керування каталогом програмного забезпечення</p>
    </div>
</header>

<main class="container">
    <div class="login-wrapper">
        <div class="selection-panel login-panel">

            <div class="login-icon-wrap">🔒</div>
            <h2 class="login-title">Вхід адміністратора</h2>
            <p class="login-subtitle">Доступ до редагування каталогу</p>

            <?php
            session_start();

            // Генеруємо CSRF токен
            if (empty($_SESSION['csrf_token'])) {
                $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
            }

            // Перевіряємо чи не заблоковано IP
            $ip = $_SERVER['REMOTE_ADDR'];
            $lockout_key = 'lockout_' . md5($ip);
                    $is_locked   =   isset($_SESSION[$lockout_key])   &&   time()   <
$_SESSION[$lockout_key];
```

```
            ?>

            <?php if (isset($_GET['error']) && $_GET['error'] === 'invalid'): ?>
                <div class="admin-message admin-message--error">
                    Невірний пароль. Спробуйте ще раз.
                </div>
            <?php endif; ?>

            <?php if ($is_locked): ?>
                 <div class="admin-message admin-message--error" style="background:
rgba(239, 68, 68, 0.15);">
                    ✖ Занадто багато невдалих спроб.<br>
                    <small>Спробуйте увійти через 15 хвилин.</small>
                </div>
            <?php endif; ?>

            <?php if (isset($_GET['error']) && $_GET['error'] === 'timeout'): ?>
                <div class="admin-message admin-message--error">
                    Сесія закінчилася. Будь ласка, введіть пароль заново.
                </div>
            <?php endif; ?>

               <form method="POST" action="admin_auth.php" class="login-form" <?=
$is_locked ? 'disabled' : '' ?>>
                <div class="form-group">
                    <label for="password">Пароль адміністратора</label>
                    <div class="password-wrap">
                        <input type="password" id="password" name="password"
                               class="admin-input" placeholder="Введіть пароль..."
                                required autofocus <?= $is_locked ? 'disabled' : ''
?>>
                                        <button  type="button"  class="toggle-pass"
onclick="togglePass()"
                                title="Показати/приховати" <?= $is_locked ? 'disabled'
: '' ?>>
                            <span id="eyeIcon">👁</span>
                        </button>
                    </div>
                </div>

                <!-- CSRF Токен -->
                             <input  type="hidden"  name="csrf_token"  value="<?=
htmlspecialchars($_SESSION['csrf_token']) ?>">

               <button type="submit" class="btn-submit" <?= $is_locked ? 'disabled'
: '' ?>>
                    Увійти до панелі
                </button>
            </form>

              <a href="index.php" class="login-back-link">← Повернутись на головну
сторінку</a>
        </div>
    </div>
</main>
<footer>
    <div class="container">
           <p  style="color:#fff;">&copy;  2026  Розробка  автоматизованої  системи
порівняльного аналізу ПЗ. Кваліфікаційна робота.</p>
    </div>
</footer>
<script>
function togglePass() {
```

```
    const inp = document.getElementById('password');
    const icon = document.getElementById('eyeIcon');

    if (inp.type === 'password') {
        inp.type = 'text';
        icon.textContent = '👁';
    } else {
        inp.type = 'password';
        icon.textContent = '👁';
    }
}
</script>
</body>
</html>
```

Код файлу admin_logout.php.

```
<?php
session_start();
// Логує вихід
$log_file = __DIR__ . '/logs/security.log';
if (!is_dir(__DIR__ . '/logs')) {
    mkdir(__DIR__ . '/logs', 0700, true);
}
$ip = $_SERVER['REMOTE_ADDR'];
$message = date('[Y-m-d H:i:s]') . " [ADMIN_LOGOUT] IP: $ip" . PHP_EOL;
file_put_contents($log_file, $message, FILE_APPEND);
// Знищує сесію
session_destroy();

header('Location: index.php?logout=true');
exit;
?>
```

Код файлу admin_auth.php.

```
<?php
//  БЕЗПЕЧНА ОБРОБКА ЛОГІНУ АДМІНІСТРАТОРА

session_start();
define('MAX_LOGIN_ATTEMPTS', 3);
define('LOCKOUT_TIME', 300); // 5 хвилин
define('USE_PLAIN_PASSWORD', true);

// Захист від брутфорсу
function checkLoginAttempts() {
    $ip = $_SERVER['REMOTE_ADDR'];
    $attempt_key = 'login_attempts_' . md5($ip);
    $lockout_key = 'lockout_' . md5($ip);

    if (isset($_SESSION[$lockout_key]) && time() < $_SESSION[$lockout_key]) {
        return false;
    }
    $attempts = isset($_SESSION[$attempt_key]) ? $_SESSION[$attempt_key] : 0;
    return $attempts < MAX_LOGIN_ATTEMPTS;
}
function incrementLoginAttempts() {
    $ip = $_SERVER['REMOTE_ADDR'];
    $attempt_key = 'login_attempts_' . md5($ip);
    $lockout_key = 'lockout_' . md5($ip);
```

```
            $_SESSION[$attempt_key]     =     (isset($_SESSION[$attempt_key])     ?
$_SESSION[$attempt_key] : 0) + 1;
    if ($_SESSION[$attempt_key] >= MAX_LOGIN_ATTEMPTS) {
        $_SESSION[$lockout_key] = time() + LOCKOUT_TIME;
        logSecurityEvent('BRUTE_FORCE_ATTEMPT', $_SERVER['REMOTE_ADDR']);
    }
}
function resetLoginAttempts() {
    $ip = $_SERVER['REMOTE_ADDR'];
    $attempt_key = 'login_attempts_' . md5($ip);
    $lockout_key = 'lockout_' . md5($ip);

    unset($_SESSION[$attempt_key]);
    unset($_SESSION[$lockout_key]);
}

function logSecurityEvent($event, $ip) {
    $log_file = __DIR__ . '/logs/security.log';
    if (!is_dir(__DIR__ . '/logs')) {
        mkdir(__DIR__ . '/logs', 0700, true);
    }

    $message = date('[Y-m-d H:i:s]') . " [$event] IP: $ip" . PHP_EOL;
    file_put_contents($log_file, $message, FILE_APPEND);
}

// Перевірка CSRF токену
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_POST['csrf_token']) || !hash_equals($_SESSION['csrf_token'] ?? '',
$_POST['csrf_token'])) {
        http_response_code(403);
        logSecurityEvent('CSRF_ATTACK_ATTEMPT', $_SERVER['REMOTE_ADDR']);
        exit('Невалідний запит');
    }

    if (!checkLoginAttempts()) {
       $remaining_time = ceil(($_SESSION['lockout_' . md5($_SERVER['REMOTE_ADDR'])]
- time()) / 60);
        http_response_code(429);
        logSecurityEvent('LOCKOUT_ATTEMPT', $_SERVER['REMOTE_ADDR']);
        $_SESSION['login_error'] = "Занадто багато невдалих спроб. Спробуйте через
$remaining_time хвилин.";
        header('Location: admin_login.php?error=locked');
        exit;
    }

    $password = $_POST['password'] ?? '';

    // Налаштування паролей
    $plain_password = 'admin123';
                                            $hashed_password                    =
'$2y$10$V5dxnEKRKKgZ.YdG8qQc5O9WVTc8Zqf.lR8ZZXLY0QZVJ8Yf6zrZK';

    // Перевіряємо пароль
    if (USE_PLAIN_PASSWORD) {
        $password_correct = ($password === $plain_password);
    } else {
        $password_correct = password_verify($password, $hashed_password);
    }

    if ($password_correct) {
        resetLoginAttempts();
        session_regenerate_id(true);
```

```
        $_SESSION['admin_logged_in'] = true;
        $_SESSION['admin_time'] = time();
        $_SESSION['admin_ip'] = $_SERVER['REMOTE_ADDR'];
        $_SESSION['admin_user_agent'] = $_SERVER['HTTP_USER_AGENT'] ?? '';

        logSecurityEvent('ADMIN_LOGIN_SUCCESS', $_SERVER['REMOTE_ADDR']);

        header('Location: admin.php');
        exit;
    } else {
        incrementLoginAttempts();
        logSecurityEvent('FAILED_LOGIN_ATTEMPT', $_SERVER['REMOTE_ADDR']);

        $_SESSION['login_error'] = 'Невірний пароль';
        header('Location: admin_login.php?error=invalid');
        exit;
    }
}
header('Location: admin_login.php');
exit;
?>
```