МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСИСТЕМИ НАВЧАННЯ З ЕЛЕМЕНТАМИ ГЕЙМІФІКАЦІЇ»**

Виконала: студентка 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
К. І. Большова
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Миронова Н. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.т.н. Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

« 08 » грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Большовій Карині Іванівній
(прізвище, ім'я та по-батькові)

1. Тема роботи Проєктування та розробка вебсистеми навчання з елементами гейміфікації

керівник роботи Миронова Наталя Олексіївна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та постановка задачі.
2. Проєктування вебсистеми навчання.
3. Розробка та тестування вебсистеми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	17.02.2026	
2.	Збір вихідних даних.	04.03.2026	
3.	Обробка методичних та теоретичних	12.03.2026	
	джерел.		
4.	Розробка першого та другого розділу.	20.04.2026	
5.	Розробка третього розділу.	01.05.2026	
6.	Оформлення та нормоконтроль	02.06.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2026	

Студент _____ К. І.Большова
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н. О.Миронова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування та розробка вебсистеми навчання з елементами гейміфікації»: 125 с., 45 рис., 18 табл., 38 джерел, 5 додатків.

АДАПТИВНЕ НАВЧАННЯ, ВЕБСИСТЕМА НАВЧАННЯ, ГЕЙМІФІКАЦІЯ, ДОСВІД (XP), ДОСЯГНЕННЯ, ЕЛЕКТРОННЕ НАВЧАННЯ, ОСВІТНЯ ПЛАТФОРМА, NEXT.JS, REACT.

Об’єктом дослідження є процеси організації електронного навчання в вебсередовищі.

Предметом дослідження є методи та засоби проєктування і розробки вебсистеми навчання з використанням елементів гейміфікації.

Метою роботи є проєктування та розробка вебсистеми навчання з елементами гейміфікації, яка забезпечує підвищення ефективності навчального процесу шляхом інтерактивної взаємодії користувачів та використання мотиваційних механізмів.

Методи дослідження – аналіз літератури, порівняльний аналіз існуючих систем, методи проєктування програмного забезпечення, об’єктно-орієнтоване програмування, тестування (unit, інтеграційне, E2E), оцінка продуктивності вебзастосунків.

У роботі проаналізовано сучасні вебсистеми навчання, досліджено принципи гейміфікації, сформовано вимоги до системи, спроєктовано архітектуру та базу даних. Реалізовано вебсистему з використанням сучасного технологічного стеку, включаючи серверну та клієнтську частини, а також гейміфікаційні модулі. Проведено тестування системи та оцінено її ефективність. Отримані результати підтверджують працездатність та практичну цінність розробленої системи в освітньому процесі.

SUMMARY

Bachelor's Qualifying Thesis «Design and Development of a Web-Based Learning System with Gamification Elements»: 125 pages, 45 figures, 18 tables, 38 references, 5 supplements.

ADAPTIVE LEARNING, WEB-BASED LEARNING SYSTEM, GAMIFICATION, EXPERIENCE (XP), ACHIEVEMENTS, E-LEARNING, EDUCATIONAL PLATFORM, NEXT.JS, REACT.

The object of the research is the processes of organizing e-learning in a web environment.

The subject of the research is the methods and tools for designing and developing a web-based learning system using gamification elements.

The aim of the work is to design and develop a web-based learning system with gamification elements that improves the efficiency of the educational process through interactive user engagement and motivational mechanisms.

Research methods include literature analysis, comparative analysis of existing systems, software design methods, object-oriented programming, testing (unit, integration, and end-to-end testing), and performance evaluation of web applications.

The work analyzes modern web-based learning systems, studies the principles of gamification, defines system requirements, and designs the architecture and database. A web system is implemented using a modern technology stack, including backend and frontend components as well as gamification modules. System testing is conducted and its effectiveness is evaluated. The obtained results confirm the functionality and practical value of the developed system in the educational process.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Аналіз предметної області та постановка задачі 11
1.1 Поняття вебсистеми навчання 11
1.2 Гейміфікація в електронному навчанні 13
1.3 Аналіз наявних рішень у сфері вебсистем навчання та гейміфікації 15
1.3.1 Moodle 15
1.3.2 Canvas LMS..... 16
1.3.3 Blackboard Learn..... 17
1.3.4 TalentLMS 18
1.3.5 Duolingo 18
1.3.6 Khan Academy..... 19
1.3.7 Classcraft..... 20
1.4. Вимоги до вебсистеми: функціональні та нефункціональні 22
2 Проєктування вебсистеми навчання 26
2.1 Проєктування функціональної та інформаційної структури вебсистеми 26
2.2. Проєктування архітектури програмної системи та бази даних..... 32
2.4 Опис потоків даних та взаємодії компонентів системи 41
2.3. Обґрунтування вибору технологічного стеку розробки 47
3 Розробка та тестування вебсистеми 51
3.1 Реалізація серверної частини системи 51
3.2 Реалізація клієнтської частини та користувацького інтерфейсу 55
3.3 Реалізація елементів гейміфікації..... 60
3.4 Тестування та оцінка ефективності системи..... 63

3.5 Інструкція користувача вебсистеми 66
Висновки 70
Перелік посилань 73
Додаток А Лістинг – реалізація серверної частини системи 77
Додаток Б Лістинг – реалізація клієнтської частини та користувацького інтерфейсу 84
Додаток В Інтерфейс розробленої вебсистеми 89
Додаток Г Лістинг – реалізація елементів гейміфікації 105
Додаток Д Лістинг – тестування 114

ВСТУП

У сучасних умовах цифрової трансформації освіти відбувається активне впровадження вебтехнологій у навчальний процес, що суттєво змінює підходи до організації навчання, взаємодії викладача та здобувача освіти, а також оцінювання навчальних досягнень. Вебсистеми навчання стають невід'ємною складовою освітнього середовища закладів освіти різних рівнів, забезпечуючи доступність, гнучкість та індивідуалізацію навчального процесу. Особливої актуальності набуває інтеграція елементів гейміфікації, які дозволяють підвищити мотивацію користувачів, залученість до навчального процесу та ефективність засвоєння матеріалу.

Актуальність теми зумовлена зростаючими вимогами до якості електронного навчання та необхідністю підвищення мотиваційної складової освітніх платформ. Традиційні системи дистанційного навчання часто стикаються з проблемою зниження залученості студентів, відсутності стабільної внутрішньої мотивації та недостатньої інтерактивності. У цьому контексті гейміфікація, як використання ігрових механік у неігрових процесах, стає ефективним інструментом підвищення активності користувачів, формування стійкого інтересу до навчання та покращення результатів засвоєння знань.

Сучасні дослідження та практичний досвід провідних освітніх платформ (зокрема Moodle, Coursera, Duolingo, Khan Academy) свідчать про ефективність впровадження таких елементів гейміфікації, як бали, рівні, досягнення, рейтинги та системи нагород. Водночас аналіз існуючих рішень показує, що значна частина вебсистем або недостатньо адаптує гейміфікаційні механізми до конкретних навчальних сценаріїв, або реалізує їх поверхнево, без глибокої інтеграції в архітектуру платформи. Це обумовлює необхідність розробки сучасної вебсистеми навчання з продуманою архітектурою та ефективною реалізацією елементів гейміфікації.

Метою роботи є проєктування та розробка вебсистеми навчання з елементами гейміфікації, яка забезпечує підвищення ефективності навчального процесу шляхом інтерактивної взаємодії користувачів та використання мотиваційних механізмів.

Для досягнення поставленої мети визначено такі основні завдання:

- проаналізувати сучасні вебсистеми навчання та підходи до їх побудови;
- дослідити принципи гейміфікації та її застосування в електронному навчанні;
- проаналізувати наявні рішення та сформувати вимоги до вебсистеми;
- спроєктувати архітектуру системи та базу даних;
- реалізувати вебсистему навчання з елементами гейміфікації;
- провести тестування системи та оцінити її ефективність.

Об’єктом дослідження є процеси організації електронного навчання в вебсередовищі.

Предметом дослідження є методи та засоби проєктування і розробки вебсистеми навчання з використанням елементів гейміфікації.

Методи дослідження – аналіз літератури, порівняльний аналіз існуючих систем, методи проєктування програмного забезпечення, об’єктно-орієнтоване програмування, тестування (unit, інтеграційне, E2E), оцінка продуктивності вебзастосунків.

Практичне значення отриманих результатів полягає у створенні готової до використання вебсистеми навчання, яка може бути впроваджена у закладах освіти або використана як основа для подальшого розвитку освітніх платформ. Запропоновані рішення можуть бути адаптовані для різних предметних областей, включаючи шкільну, університетську та корпоративну освіту.

Теоретичне значення роботи полягає у систематизації підходів до використання гейміфікації в електронному навчанні та аналізі сучасних методів побудови вебсистем освітнього призначення. Отримані результати можуть бути використані для подальших наукових досліджень у сфері освітніх технологій та розробки інтерактивних навчальних платформ.

Наукова новизна роботи полягає у комплексному поєднанні сучасної вебархітектури з механізмами гейміфікації, що дозволяє не лише забезпечити функціональність системи навчання, але й підвищити мотивацію користувачів шляхом інтегрованих ігрових елементів у структуру навчального процесу.

Отримані результати можуть бути застосовані при розробці сучасних систем дистанційного навчання, корпоративних освітніх платформ, інтерактивних тренінгових систем, а також у сфері EdTech-рішень для підвищення залученості користувачів та покращення якості навчання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття вебсистеми навчання

Вебсистема навчання є ключовим елементом сучасного електронного освітнього середовища та виступає технологічною основою організації дистанційного й змішаного навчання. У загальному розумінні вона визначається як інформаційно-педагогічна система, що базується на вебтехнологіях і забезпечує підтримку, організацію та управління навчальним процесом у мережевому середовищі. Її функціонування здійснюється через вебінтерфейс, що робить доступ до навчальних ресурсів незалежним від місця та часу навчання [1].

Аналіз наукових джерел дозволяє зробити висновок, що вебсистеми навчання розглядаються не лише як інструмент надання навчального контенту, а як комплексне цифрове середовище, яке поєднує навчальні, комунікаційні та управлінські функції. У цьому контексті вони виступають підсистемою ширшого поняття e-learning, що охоплює різні форми електронного навчання та технологічно орієнтованих освітніх рішень [2].

Однією з ключових характеристик вебсистем навчання є їхня інтернет-орієнтованість, яка забезпечує доступ до освітніх ресурсів через глобальну мережу. Це принципово відрізняє їх від автономних навчальних програм і дозволяє реалізувати безперервний доступ до навчання. У дослідженнях підкреслюється, що саме мережевий характер таких систем забезпечує їхню масштабованість та гнучкість у використанні [3].

Іншою важливою ознакою є інтерактивність, яка передбачає активну взаємодію між усіма учасниками навчального процесу. На практиці це реалізується через форуми, чати, тестові завдання, системи зворотного зв’язку та інші інструменти комунікації. Саме інтерактивність дозволяє перейти від

пасивного засвоєння інформації до активного навчання, що підвищує ефективність освітнього процесу.

Третьою базовою характеристикою є керованість навчальним процесом, яка реалізується через функціонал систем управління навчанням (LMS). Вона включає адміністрування курсів, контроль успішності, реєстрацію користувачів, аналітику навчальної діяльності та формування звітності. У наукових дослідженнях підкреслюється, що саме цей компонент забезпечує системність та структурованість освітнього процесу в цифровому середовищі [2].

Тобто узагальнюючи різні підходи, вебсистему навчання можна визначити як інтегроване цифрове середовище, що поєднує навчальні ресурси, інструменти комунікації та засоби управління освітнім процесом для реалізації електронного навчання в Інтернет-середовищі. Це визначення підкреслює її комплексний характер і вихід за межі звичайного розміщення навчальних матеріалів у мережі.

Структурно вебсистема навчання включає кілька взаємопов’язаних компонентів. Першим є навчальний контент, який містить структуровані матеріали курсу: тексти, презентації, відео та інтерактивні завдання. Другим є модуль управління навчальним процесом, який забезпечує організацію курсів, облік студентів, контроль виконання завдань та аналітику результатів. Третім є комунікаційний модуль, що забезпечує взаємодію між користувачами через синхронні та асинхронні канали зв’язку [4].

Аналіз структури таких систем показує, що їх ефективність визначається не лише наявністю окремих модулів, а й рівнем їх інтеграції. У сучасних вебсистемах навчання важливо забезпечити єдину інформаційну архітектуру, де навчальний контент, комунікація та управління даними працюють як цілісна екосистема, а не як окремі незалежні компоненти.

У цілому вебсистеми навчання можна розглядати як базовий інструмент цифрової трансформації освіти, який забезпечує перехід від традиційної моделі навчання до гнучких, персоналізованих та інтерактивних форм освітньої діяльності. Вони створюють основу для подальшого впровадження інноваційних

підходів, зокрема гейміфікації, адаптивного навчання та використання штучного інтелекту в освіті.

1.2 Гейміфікація в електронному навчанні

Гейміфікація в електронному навчанні є однією з ключових сучасних педагогічних технологій, спрямованих на підвищення мотивації та залученості здобувачів освіти в цифровому освітньому середовищі. У загальному розумінні вона передбачає інтеграцію ігрових елементів у неігрові навчальні процеси з метою підвищення інтересу, активності та результативності навчання. При цьому важливо підкреслити, що гейміфікація не передбачає повного перетворення навчального курсу на гру, а лише використання окремих ігрових механік у структурі освітнього процесу [5].

Аналіз наукових джерел показує, що гейміфікація розглядається як стратегія впливу на поведінку та мотивацію користувачів шляхом використання елементів геймдизайну в освітніх системах. У контексті електронного навчання це означає впровадження механізмів прогресу, винагород, цілей та зворотного зв’язку у навчальні платформи, системи управління навчанням (LMS) та онлайн-курси [6].

Важливим теоретичним аспектом є розмежування понять гейміфікації та game-based learning (ігрового навчання). Якщо ігрове навчання передбачає створення повноцінних освітніх ігор з окремою ігровою механікою, то гейміфікація полягає у впровадженні окремих ігрових елементів у вже наявну навчальну структуру без зміни її змістової основи [7]. Тобто гейміфікація змінює не зміст навчання, а спосіб його подачі та організації взаємодії користувача із системою.

У процесі аналізу наукових досліджень встановлено, що найбільш поширеними елементами гейміфікації в електронному навчанні є бали, рівні, бейджі, рейтинги, прогрес-бари та сюжетні (квестові) завдання. Кожен із цих

елементів виконує окрему мотиваційну функцію. Зокрема, система балів забезпечує кількісну оцінку досягнень, рівні формують відчуття поступового розвитку, бейджі виступають символічним підтвердженням досягнень, а рейтинги створюють елемент соціального порівняння результатів [8].

Окремо слід виділити роль прогресивних індикаторів, таких як прогрес-бари, які дозволяють користувачу візуально відстежувати власний рух у навчальному курсі. Дослідження показують, що саме візуалізація прогресу суттєво впливає на рівень залученості та завершення навчальних курсів, оскільки формує чітке уявлення про досягнутий та залишковий обсяг роботи [6].

Аналіз ефективності гейміфікації в електронному навчанні свідчить про її позитивний вплив на когнітивні, поведінкові та мотиваційні результати навчання. Зокрема, дослідження показують зростання рівня залученості студентів, підвищення регулярності виконання завдань та покращення навчальних досягнень. Водночас ефект є помірним і залежить від якості педагогічного проєктування гейміфікованої системи. Емпіричні дослідження також підтверджують, що найбільш ефективними елементами є системи миттєвого зворотного зв’язку, прогрес-індикатори та бали, тоді як бейджі та рейтинги мають допоміжний характер [5]. Це свідчить про те, що ефективність гейміфікації визначається не кількістю ігрових елементів, а їх педагогічною доцільністю та інтеграцією в структуру навчального процесу.

У науковій літературі також підкреслюється, що гейміфікація повинна базуватися на принципах внутрішньої мотивації, зокрема автономії, відчуття компетентності та соціальної взаємодії. Якщо система надмірно орієнтована на зовнішні винагороди (бали, рейтинги), це може призвести до зниження внутрішньої мотивації до навчання та формального виконання завдань без глибокого засвоєння матеріалу [6; 9].

Окрему увагу слід приділити ризикам використання гейміфікації. Серед основних проблем дослідники виділяють можливе підвищення рівня навчального стресу через конкуренцію, нерівномірність залучення користувачів, а також ризик спрощення навчального процесу до механічного накопичення

балів. Це підкреслює необхідність збалансованого підходу до проєктування гейміфікованих систем [9].

Отже, гейміфікація в електронному навчанні є ефективним інструментом підвищення якості освітнього процесу, однак її результативність безпосередньо залежить від правильного педагогічного та системного проєктування. Найбільш успішними є ті рішення, які інтегрують ігрові механіки не як декоративні елементи, а як функціональну частину навчальної моделі, що підтримує досягнення освітніх цілей.

1.3 Аналіз наявних рішень у сфері вебсистем навчання та гейміфікації

Стрімкий розвиток цифрових технологій суттєво трансформував освітній простір, породивши цілий клас платформ, що прагнуть зробити навчання ефективнішим, інтерактивнішим та більш мотивуючим. Серед ключових інструментів цього процесу виокремлюються системи управління навчанням (LMS – Learning Management System) та підходи до гейміфікації навчального контенту. Глобальний ринок LMS у 2024 році оцінювався в діапазоні від 22,9 до 26,2 млрд доларів США і продовжує зростати [10]. Нижче подано детальний аналіз найбільш поширених рішень – як класичних LMS, так і спеціалізованих гейміфікованих платформ.

1.3.1 Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) є найпоширенішою у світі відкритою системою управління навчанням. Серед «великої четвірки» академічних LMS – Canvas, Blackboard, Moodle і D2L Brightspace – кожна з двох останніх платформ утримує по 16% ринку в Північній Америці [10]. Платформа використовується тисячами університетів, шкіл і

корпорацій по всьому світу та відрізняється надзвичайно гнучкою модульною архітектурою.

Основна перевага платформи – наявність понад 1 600 плагінів у каталозі, що охоплюють відстеження відвідуваності, інструменти гейміфікації та розширену аналітику [11]. Завдяки цій екосистемі Moodle здатна інтегруватися із зовнішніми системами: Zoom, Microsoft Teams, HR-платформами та системами студентського обліку. Відкритий код дозволяє спрямувати кошти, зекономлені на ліцензіях, на розробку навчального контенту.

Проте гейміфікація у Moodle реалізована не нативно, а через зовнішні модулі. Дослідження 2024 року підтвердило, що Moodle і Blackboard підтримують гейміфікаційні функції виключно через додаткові плагіни [12]. Впровадження таких модулів потребує значних інфраструктурних змін, а подекуди – повного перегляду архітектури розгортання [13]. Для закладів без потужної IT-команди Moodle може перетворитися на суттєве технічне навантаження: розгортання повноцінного середовища займає тижні або місяці. Інтерфейс платформи критикується за застарілість і недостатню інтуїтивність [11]. Організація несе повну відповідальність за безпеку, хостинг та оновлення системи.

1.3.2 Canvas LMS

Canvas LMS, розроблена компанією Instructure, є беззаперечним лідером академічного ринку з часткою 41% у Північній Америці [10]. Платформа вирізняється чистим інтуїтивним інтерфейсом і завоювала популярність серед університетів та коледжів, які шукають зручне у використанні рішення.

Серед ключових переваг Canvas – відкритий API, що дозволяє розробляти власні інтеграції [14]. Система підтримує стандарт LTI (Learning Tools Interoperability), що забезпечує сумісність з Microsoft Teams, Google Classroom, Zoom і Adobe Creative Suite [14]. Мобільний застосунок та зручний редактор курсів отримують стабільно позитивні відгуки від користувачів.

У питанні гейміфікації Canvas поступається конкурентам. Інструменти гейміфікації – жетони, бали, персоналізовані навчальні шляхи – потребують ручного налаштування або залучення сторонніх рішень: система балів прив’язана до виконання завдань, а не до спеціалізованих гейміфікаційних механізмів. Таблиці лідерів формуються вручну. Платформа є комерційним рішенням; окремі просунуті модулі потребують доплати, а можливості кастомізації в нижніх тарифних планах обмежені [14].

1.3.3 Blackboard Learn

Blackboard Learn, що нині входить до складу компанії Anthology, є однією з найдавніших LMS-платформ. Система обслуговує заклади вищої освіти, бізнес і державні установи більш ніж у 70 країнах, розгорнута на інфраструктурі Amazon Web Services [15].

Blackboard відрізняється широким набором інструментів для управління курсами та оцінювання, орієнтованих на великі установи. Функції штучного інтелекту включені до базових ліцензій, що є конкурентною перевагою [15]. Підтримка доступності (accessibility) виконана на достатньо високому рівні.

Проте Blackboard традиційно критикують за надмірну складність і громіздкість інтерфейсу. Гейміфікація доступна лише через плагіни – аналогічно до Moodle [12].

Це означає, що значні інфраструктурні зміни є необхідною умовою для розгортання повноцінних гейміфікаційних елементів [13]. Ціна платформи не публікується відкрито, а деякі функції – Video Studio та AI Summarize – є платними додатками [15].

Висока вартість ліцензій у поєднанні з необхідністю залучення зовнішніх підрядників для кастомізації робить платформу недоступною для малих і середніх закладів.

1.3.4 TalentLMS

TalentLMS – хмарна LMS-система, орієнтована насамперед на корпоративне навчання. Вона широко застосовується для навчання персоналу, онбордингу та навчання клієнтів у компаніях різного масштабу [11]. На відміну від академічних LMS, TalentLMS розроблена за принципом «розгорнуто й запущено одразу» без потреби у технічній команді.

Ключовою перевагою TalentLMS у контексті гейміфікації є наявність вбудованих механізмів: балів, рівнів, жетонів і таблиць лідерів – без необхідності встановлювати сторонні плагіни [14]. За результатами порівняльного аналізу, TalentLMS перевершує Canvas за простотою та комплексністю гейміфікаційного досвіду саме завдяки нативному набору інструментів [14]. Платформа також пропонує інструмент генерації контенту на базі штучного інтелекту – TalentCraft, а також бібліотеку з понад 1 000 готових курсів. За оцінками користувачів платформи, інтерфейс є інтуїтивним і зручним [11].

Разом із тим TalentLMS має обмеження, що проявляються при масштабуванні. Можливості брендування базові, складна розгалужена логіка навчальних маршрутів не підтримується нативно [11]. Екосистема інтеграцій менша, ніж у Moodle. Платформа не є ідеальною для дуже великих або сильно сегментованих організацій, а можливості розширеної гейміфікації та інтерактивності обмежені порівняно зі спеціалізованими рішеннями.

1.3.5 Duolingo

Duolingo – найбільш показовий приклад повноцінної гейміфікації у сфері освіти. Заснований у 2011 році, сервіс є лідером серед освітніх мобільних застосунків: у 2025 році платформа досягла зростання кількості щоденних активних користувачів на 36% у річному вимірі [16].

Стратегія гейміфікації Duolingo включає серії (streaks), ліги та таблиці лідерів, систему рівнів, значки, квести та персонажа Duo – фірмову зелену сову,

яка нагадує про пропущені уроки. Завдяки цим механізмам відтік користувачів (churn) знизився з 47% у 2020 році до 28% у ключових ринках до 2024 року [16]. Дослідження підтверджують, що гейміфікаційна стратегія Duolingo сприяє кращій абсорбції знань, підвищенню уваги та мотивації, а також формуванню звички регулярного навчання [17].

Проте Duolingo як навчальний інструмент має суттєві обмеження. Платформа спеціалізується виключно на вивченні іноземних мов і не є універсальною освітньою системою. Серед зафіксованих недоліків – нестача матеріалів для читання та розмовної практики, повторюваність завдань, залежність від інтернет-з'єднання та наявність реклами в безплатній версії [18]. Дослідження 2025 року виявило, що гейміфікований підхід Duolingo не дає статистично значущих переваг для самоефективності порівняно з традиційним навчанням [17]. Конкурентні механіки підходять не всім типам учнів, а гейміфікація може дрейфувати до «порожнього гріндингу» – механічного виконання завдань заради балів [16].

1.3.6 Khan Academy

Khan Academy – некомерційна освітня платформа, що пропонує безплатний відкритий доступ до навчальних відеоматеріалів і вправ з математики, природничих наук, гуманітарних дисциплін та комп'ютерних наук. Платформа використовує елементи гейміфікації: смуги прогресу, значки та систему балів [19].

Серед переваг Khan Academy – повна безплатність, широкий охват дисциплін, якісні відеопояснення та адаптивний підхід до практики, що автоматично підлаштовує складність завдань під рівень знань учня. Штучний інтелект Khanmigo забезпечує персоналізовану підтримку у форматі інтелектуального репетитора. Гейміфікаційні елементи допомагають розбивати складний матеріал – зокрема, математику і фізику – на керовані блоки [19].

Однак Khan Academy зазнає критики за поступове послаблення гейміфікаційних механізмів. Один невдалий елемент – наприклад, неможливість скинути результати тестів – здатен нівелювати десятки позитивних мотивуючих факторів [19]. Платформа орієнтована переважно на учнів середньої та старшої школи й не містить достатньо матеріалів для молодших школярів. Повторюваність завдань знижує мотивацію, а платформа не є повноцінною LMS і не підходить для формального академічного середовища з атестацією та управлінням курсами.

1.3.7 Classcraft

Classcraft – унікальна платформа, що перетворює шкільний клас на рольову гру (RPG). Система дозволяє вчителям заохочувати позитивну поведінку та знання, нараховуючи учням досвідні очки й золоті монети, або карати за порушення – відніманням умовного «здоров'я» [19]. Компанія керується принципом: мотивувати через задоволення, а не через страх покарання.

Ключова перевага Classcraft – глибока інтеграція гейміфікації у процес управління класом. Командна гейміфікація стимулює співпрацю: учні об'єднуються в команди для вирішення спільних завдань [19]. Дослідження підтверджують, що підхід, на якому базується Classcraft, розвиває внутрішню мотивацію та когнітивну гнучкість учня ефективніше, ніж традиційна зовнішньо-мотивуюча гейміфікація [20].

Водночас Classcraft орієнтована на формат очного або змішаного навчання в школах K–12 і не призначена для дистанційного навчання дорослих чи корпоративного використання. Платформа не є повноцінною LMS: вона не управляє навчальними матеріалами, не генерує академічну звітність і не інтегрується зі студентськими інформаційними системами без додаткових налаштувань. Конкурентні штрафні механіки можуть негативно впливати на учнів, які не є змагальними за природою. Безплатна версія значно обмежена у

функціональності, а безпека персональних даних учнів потребує ретельного захисту відповідно до нормативних вимог [13].

З метою узагальнення результатів проведеного аналізу та наочного порівняння основних характеристик розглянутих платформ у сфері вебсистем навчання та гейміфікації, основні параметри систем наведено в таблиці 1.1. Порівняння здійснюється за такими критеріями, як тип системи, рівень інтеграції гейміфікації, гнучкість налаштування, складність впровадження, модель доступу (ціна) та цільова аудиторія. Такий підхід дозволяє комплексно оцінити переваги та недоліки кожного рішення та визначити їх місце в сучасному освітньому середовищі.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень

Платформа	**Тип**	**Геймі-фікація**	**Гнучкість**	**Складність**	**Ціна**	**Цільова аудиторія**
Moodle	LMS	Плагіни	Висока	Висока	FREE	Університети, школи
Canvas LMS	LMS	Часткова	Середня	Середня	PAID	Вища освіта
Blackboard	LMS	Плагіни	Низька	Висока	PAID +	Великі ВНЗ
TalentLMS	Корп. LMS	Нативна	Середня	Низька	PAID	Корпоративне навчання
Duolingo	Мобільна	Повна	Низька	Низька	Free mium	Масова аудиторія
Khan Academy	Навчальна	Часткова	Низька	Низька	FREE	Школярі, студенти
Classcraft	Кл. менедж.	Повна (RPG)	Середня	Середня	Free mium	Школи K–12

Наявні вебсистеми навчання суттєво відрізняються за рівнем функціональності та підходами до реалізації гейміфікації. Класичні LMS-платформи (Moodle, Canvas, Blackboard) забезпечують розвинені інструменти управління навчальним процесом, однак мають обмежену або зовнішньо реалізовану гейміфікацію. Це знижує ефективність мотиваційних механізмів та ускладнює їх інтеграцію в основну архітектуру системи. Спеціалізовані гейміфіковані платформи (Duolingo, Classcraft) демонструють високий рівень залучення користувачів завдяки глибокій інтеграції ігрових механік, однак вони

мають вузьку предметну спрямованість і не забезпечують повноцінного функціоналу LMS для формальної освіти. Корпоративні системи (TalentLMS) поєднують зручність використання та вбудовані гейміфікаційні елементи, проте орієнтовані переважно на бізнес-середовище та не повністю враховують специфіку академічного навчання.

Отже, результати порівняльного аналізу підтверджують наявність розриву між повноцінними LMS та гейміфікованими навчальними платформами, що обґрунтовує необхідність розробки інтегрованої вебсистеми навчання з вбудованими механізмами гейміфікації.

1.4. Вимоги до вебсистеми: функціональні та нефункціональні

Визначення вимог є базовим етапом проєктування програмної системи, оскільки саме на цьому рівні формується її функціональна структура, поведінка та якісні характеристики. Коректно сформульовані вимоги забезпечують узгодженість між очікуваннями користувачів та реалізацією системи, а також знижують ризики помилок на етапах розробки.

Вимоги до вебсистеми навчання з елементами гейміфікації сформовано на основі аналізу сучасних LMS, гейміфікованих платформ та потреб цільової аудиторії. Усі вимоги поділено на функціональні та нефункціональні. Функціональні вимоги визначають основні можливості системи та описують її поведінку з погляду користувача (табл. 1.2).

У таблиці подано узагальнений перелік функціональних вимог до вебсистеми навчання та гейміфікації. Вони охоплюють основні підсистеми платформи: автентифікацію та авторизацію користувачів, навчальний модуль, модуль гейміфікації, управління навчальними класами та адміністрування системи. Зокрема, передбачено реалізацію реєстрації та автентифікації користувачів із підтримкою рольової моделі доступу, а також механізмів управління сесіями.

Таблиця 1.2 – Основні функціональні вимоги

ID	Вимога	Пріоритет	Опис
FR-AUTH-01	Реєстрація користувачів	Високий	Система повинна дозволяти створення облікового запису (ім’я, email, пароль)
FR-AUTH-02	Автентифікація	Високий	Вхід у систему за email і паролем
FR-AUTH-03	Рольова модель	Високий	Підтримка ролей: учень, вчитель, адміністратор
FR-AUTH-04	Управління сесіями	Високий	JWT-аутентифікація з терміном дії
FR-LEARN-01	Каталог предметів	Високий	Перегляд доступних предметів і тем
FR-LEARN-02	Виконання завдань	Високий	Виконання інтерактивних завдань різних типів
FR-LEARN-03	Перевірка відповідей	Високий	Автоматична перевірка та зворотний зв’язок
FR-LEARN-04	Адаптивне навчання	Високий	Підлаштування складності завдань
FR-LEARN-05	Прогрес навчання	Високий	Відстеження результатів і прогресу
FR-GAME-01	XP система	Високий	Нарахування очок досвіду
FR-GAME-02	Рівні	Високий	Підвищення рівня користувача
FR-GAME-03	Досягнення	Середній	Система бейджів та нагород
FR-GAME-04	Серії активності	Середній	Відстеження streaks (щоденна активність)
FR-GAME-05	Рейтинг	Середній	Таблиця лідерів учнів
FR-CLASS-01	Створення класів	Високий	Створення навчальних класів вчителем
FR-CLASS-02	Приєднання до класу	Високий	Вхід учня за кодом класу
FR-CLASS-03	Завдання для класу	Високий	Призначення завдань вчителем
FR-CLASS-04	Статистика класу	Середній	Перегляд результатів учнів
FR-ADMIN-01	Користувачі	Високий	Управління акаунтами
FR-ADMIN-02	Контент	Високий	Створення та редагування навчального контенту
FR-ADMIN-03	Система	Середній	Перегляд загальної статистики

Навчальний модуль включає функції перегляду предметів і тем, виконання інтерактивних завдань, автоматичну перевірку відповідей, адаптацію складності навчального матеріалу та відстеження прогресу здобувачів освіти.

Гейміфікаційна складова системи передбачає нарахування очок досвіду (XP), систему рівнів, досягнень, відстеження щоденної активності та формування рейтингу користувачів.

Модуль управління класами забезпечує створення навчальних груп, приєднання до них за унікальним кодом, призначення завдань і перегляд результатів виконання. Адміністративний модуль відповідає за управління користувачами та навчальним контентом, а також за моніторинг загальної статистики системи.

Нефункціональні вимоги визначають якісні характеристики системи (табл. 1.3).

Таблиця 1.3 – Нефункціональні вимоги

Категорія	**Вимога**	**Показник**
Продуктивність	Швидкість відповіді API	≤ 500 мс
Продуктивність	Завантаження сторінки	≤ 3 сек
Безпека	Захист даних	JWT + bcrypt
Безпека	Захист від атак	XSS, CSRF, SQL injection
Надійність	Доступність системи	≥ 99%
Масштабованість	Кількість користувачів	до 10 000
Зручність	Адаптивний інтерфейс	Mobile + Desktop

У таблиці 1.3 представлені узагальнені нефункціональні вимоги до вебсистеми, які визначають якісні характеристики її роботи. Вони охоплюють основні аспекти функціонування системи, зокрема продуктивність, безпеку, надійність, масштабованість та зручність використання.

Зокрема, встановлено вимоги до швидкодії системи (час відповіді API та завантаження сторінок), забезпечення захисту даних користувачів за допомогою JWT та bcrypt, а також захисту від основних типів вебзагроз.

Також визначено вимогу до високого рівня доступності системи, можливість підтримки до 10 000 користувачів та адаптивність інтерфейсу для різних типів пристроїв.

Отже, сформовані вимоги визначають повний набір функціональних можливостей та якісних характеристик вебсистеми навчання з елементами

гейміфікації. Функціональні вимоги охоплюють основні процеси роботи системи, зокрема організацію навчального процесу, реалізацію механізмів гейміфікації, управління класами, навчальними матеріалами та користувачами різних ролей. Вони забезпечують логічну структуру взаємодії між учнем, викладачем і системою, а також визначають сценарії виконання ключових операцій у межах платформи.

Нефункціональні вимоги встановлюють параметри якості функціонування системи, включаючи продуктивність, безпеку, надійність, масштабованість та зручність використання. Особлива увага приділяється швидкодії обробки запитів, стабільності роботи при високому навантаженні, захисту користувацьких даних та можливості подальшого розширення функціоналу без суттєвого перероблення архітектури. Сукупність цих вимог формує основу для створення ефективної, стійкої та зручної у використанні навчальної платформи, яка відповідає сучасним стандартам розробки вебсистем.

2 ПРОЄКТУВАННЯ ВЕБСИСТЕМИ НАВЧАННЯ

2.1 Проєктування функціональної та інформаційної структури вебсистеми

На етапі функціонального та інформаційного проєктування вебсистеми EduQuest було визначено її логічну структуру, ключові модулі, основні процеси взаємодії користувачів із системою, а також інформаційні потоки між компонентами. У межах проєктованої вебсистеми навчання визначено три основні актори: учень, вчитель та адміністратор (рис. 2.1). Кожен з них взаємодіє із системою відповідно до своїх ролей та рівня доступу.

Учень є основним користувачем системи, який проходить навчальний процес. Йому доступні функції реєстрації, авторизації та зміни пароля, що реалізуються через модуль автентифікації. У межах навчального модуля учень може переглядати перелік предметів, обирати теми та виконувати завдання. Також передбачено можливість перегляду власного прогресу. Гейміфікаційна частина системи забезпечує отримання очок досвіду (XP), доступ до досягнень, участь у навчальних іграх та перегляд таблиці лідерів. Окрім цього, учень має можливість приєднуватися до навчальних класів.

Вчитель виступає як організатор навчального процесу. Він має доступ до авторизації та зміни пароля через модуль автентифікації. Основні функції включають створення навчальних класів, призначення завдань учням та перегляд результатів їх виконання. Це дозволяє здійснювати контроль та оцінювання навчальної діяльності учнів у межах класу.

Адміністратор відповідає за загальне управління системою. Йому доступна авторизація та функції адміністрування, які включають управління користувачами, керування навчальним контентом та налаштування системи. Це забезпечує підтримку працездатності платформи та контроль за її наповненням.

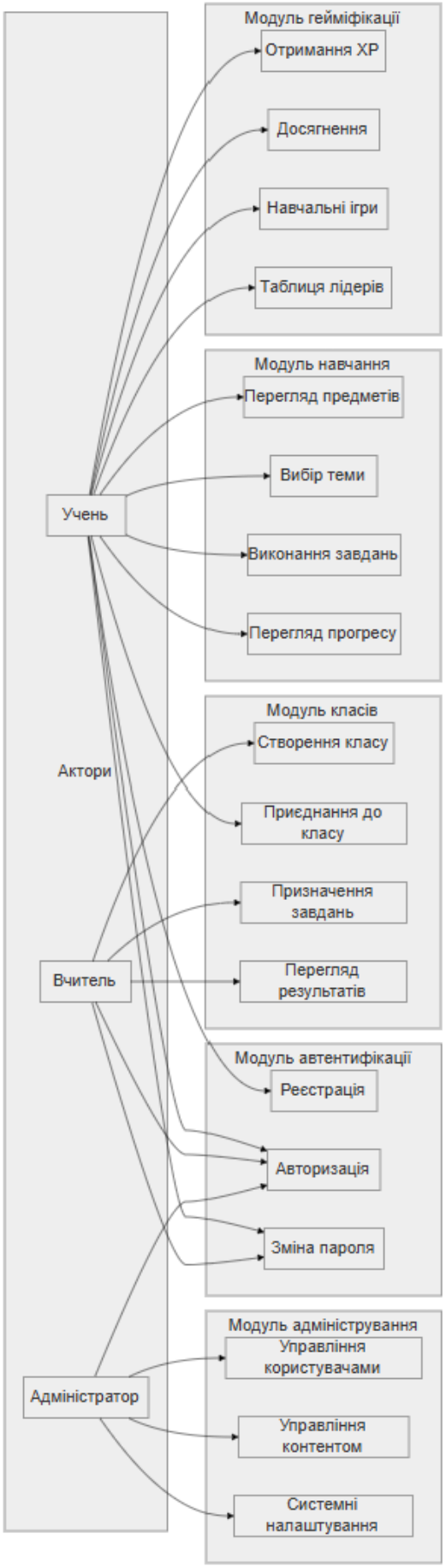


Рисунок 2.1 – Діаграма прецедентів (Use Case Diagram)

На основі визначених акторів системи та їхніх функціональних ролей доцільним є подальше деталізування внутрішньої структури вебсистеми EduQuest. Зокрема, для забезпечення зрозумілої організації програмної логіки та ефективної реалізації основних можливостей системи виконується функціональна декомпозиція, що дозволяє виділити окремі модулі відповідно до їх призначення та відповідальності.

У таблиці 2.1 представлено функціональну декомпозицію системи EduQuest.

Таблиця 2.1 – Функціональні модулі системи EduQuest

Модуль	**Призначення**	**Основні функції**
Автентифікація	Забезпечення безпечного доступу до системи	Реєстрація користувачів, авторизація, перевірка сесій, управління токенами
Навчальний модуль	Реалізація освітнього процесу	Надання навчального контенту, виконання завдань, перевірка відповідей
Гейміфікаційний модуль	Підвищення мотивації користувачів	Нарахування XP, формування рівнів, система досягнень, streak-активність
Модуль класів	Організація групового навчання	Створення класів, приєднання учнів, призначення завдань, контроль результатів
Адміністративний модуль	Загальне управління системою	Керування користувачами, контентом та системною аналітикою

Представлена декомпозиція демонструє поділ системи на логічно завершені модулі, кожен з яких відповідає за окремий аспект функціонування платформи. Такий підхід дозволяє реалізувати принцип модульності, що своєю чергою сприяє спрощенню розробки, тестування та подальшого масштабування системи. Окремо варто відзначити, що чітке розмежування відповідальностей між модулями підвищує підтримуваність коду та зменшує складність внесення змін у систему.

Подальший аналіз доцільно зосередити на деталізації ключових функціональних компонентів. Зокрема, базовим є модуль автентифікації та авторизації (табл. 2.2), який забезпечує контроль доступу до системи та є критично важливим з точки зору безпеки.

Таблиця 2.2 – Функції модуля автентифікації та авторизації

Функція	Вхідні дані	Вихідні дані	Опис
Реєстрація	email, пароль, ім’я	обліковий запис, JWT токен	Створення нового користувача та видача токена доступу
Авторизація	email, пароль	JWT токен, роль користувача	Перевірка облікових даних та визначення прав доступу
Перевірка сесії	JWT токен	дані користувача	Валідація активної сесії та отримання інформації про користувача
Зміна пароля	старий та новий пароль	статус операції	Оновлення пароля з перевіркою автентичності користувача

Модуль охоплює процеси реєстрації та входу користувачів, перевірки облікових даних, валідації активних сесій, а також оновлення пароля. Аналіз вхідних і вихідних даних кожної операції дозволяє формалізувати механізми керування доступом та забезпечити належний рівень захисту інформації в системі.

Наступним ключовим компонентом є навчальний модуль (табл. 2.3), який безпосередньо реалізує освітній процес та формує індивідуальну траєкторію навчання користувача.

Таблиця 2.3 – Функції навчального модуля

Функція	Вхідні дані	Вихідні дані	Опис
Отримання предметів	ідентифікатор користувача	список предметів із прогресом	Формування персоналізованого навчального контенту
Отримання тем	предмет, користувач	список тем з рівнем засвоєння	Визначення структури навчального процесу
Отримання завдання	тема, рівень користувача	адаптивне завдання	Підбір завдань відповідно до рівня складності
Перевірка відповіді	завдання, відповідь користувача	результат, XP, пояснення	Оцінювання виконання завдання та нарахування балів

Як випливає з наведених даних, навчальний модуль охоплює ключові етапи освітнього процесу в системі EduQuest. Зокрема, передбачено отримання персоналізованого списку предметів і тем із урахуванням рівня підготовки користувача, підбір завдань відповідно до складності, а також перевірку відповідей із автоматичним нарахуванням XP та формуванням зворотного

зв’язку. Така структура дозволяє реалізувати адаптивну модель навчання, орієнтовану на індивідуальний прогрес здобувача освіти.

Для подальшого уточнення логіки взаємодії компонентів системи було побудовано діаграму потоків даних (рис. 2.2), яка дозволяє візуалізувати загальну структуру обробки інформації в межах вебсистеми EduQuest.

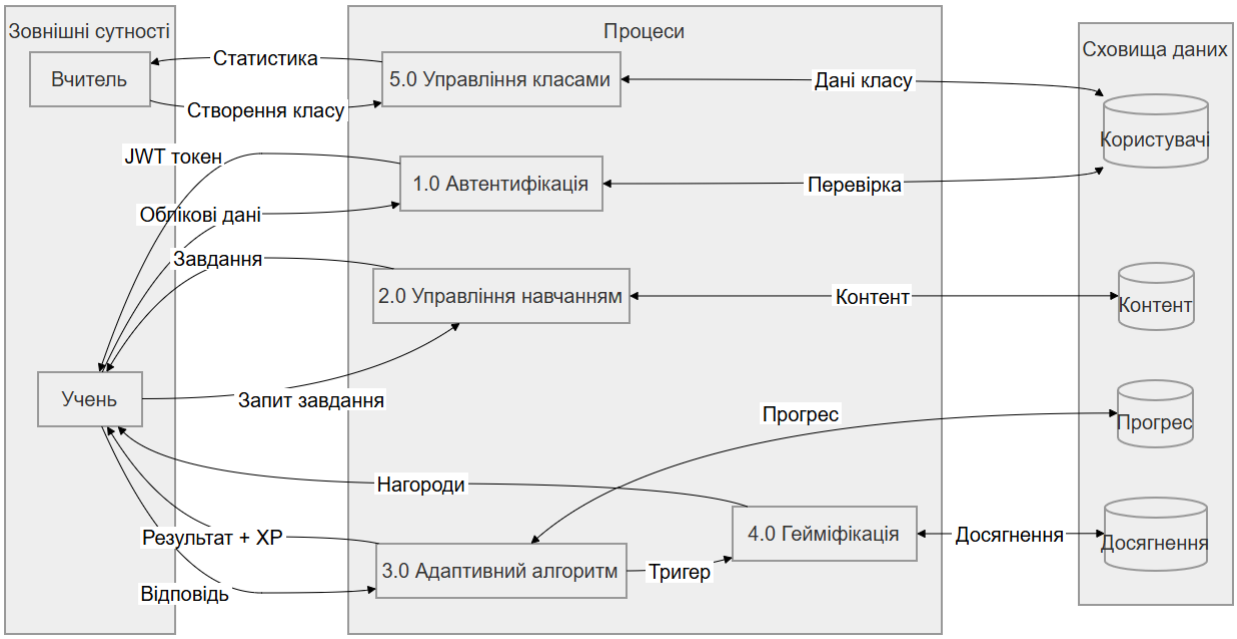


Рисунок 2.2 – Діаграма потоків даних (DFD)

На діаграмі зовнішніми сутностями виступають учень і вчитель, які взаємодіють із системою через відповідні запити. Учень здійснює доступ до навчального контенту та виконує завдання, тоді як вчитель відповідає за організацію навчального процесу та управління класами. Основні процеси системи включають автентифікацію користувачів (1.0), управління навчанням (2.0), адаптивну обробку результатів (3.0), гейміфікацію (4.0) та управління класами (5.0). Кожен із процесів виконує окрему функціональну роль і взаємодіє з відповідними сховищами даних. Система використовує чотири основні сховища: користувачі, навчальний контент, прогрес і досягнення, які забезпечують збереження та обробку інформації.

У межах інформаційного моделювання системи (табл. 2.4) визначено ключові сутності предметної області, які відображають основні об’єкти та їх характеристики.

Таблиця 2.4 – Інформаційна модель системи

Сутність	Атрибути	Опис
User	id, email, name, role, xp, level	Основна сутність користувача системи
Subject	id, name, description	Навчальний предмет
Topic	id, subject_id, name	Структурна одиниця навчального предмета
Task	id, topic_id, question, answer	Навчальне завдання
Progress	user_id, topic_id, mastery_level	Рівень засвоєння матеріалу
Class	id, teacher_id, code	Навчальна група
Achievement	id, name, condition, xp_bonus	Система нагород

Представлена інформаційна модель включає основні сутності, такі як користувач (User), навчальний предмет (Subject), тема (Topic), завдання (Task), прогрес (Progress), клас (Class) та система досягнень (Achievement). Для кожної сутності визначено набір атрибутів, що використовуються для зберігання відповідних даних та забезпечення логіки роботи системи.

Тобто інформаційна модель системи EduQuest є основою для проєктування бази даних і забезпечує цілісне представлення структури зберігання інформації. Вона дозволяє узгодити взаємодію між функціональними модулями та гарантує коректну обробку даних у межах усієї системи.

Узагальнюючи результати проєктування, можна зазначити, що визначено ключові ролі користувачів, їхні повноваження та сценарії взаємодії з платформою. На основі цього сформовано логічну структуру системи, що охоплює основні модулі: автентифікацію, навчання, гейміфікацію, управління класами та адміністрування.

Інформаційна структура системи описує сутності предметної області та їх взаємозв’язки, що забезпечує цілісне уявлення про зберігання та обробку даних і створює основу для подальшого етапу – проєктування архітектури програмного забезпечення та реалізації бази даних.

2.2 Проєктування архітектури програмної системи та бази даних

Система EduQuest реалізована на основі архітектурного підходу MVC (Model–View–Controller), який забезпечує логічне розділення відповідальності між різними рівнями програмної системи. Такий підхід дозволяє відокремити інтерфейс користувача, бізнес-логіку та рівень збереження даних, що підвищує структурованість та масштабованість програмного рішення.

У межах реалізованої архітектури можна виділити три основні рівні, які узагальнено наведено в таблиці 2.5.

Таблиця 2.5 – Архітектурна структура системи

Шар	Компоненти	Призначення
View	React-компоненти, сторінки Next.js	Відображення інтерфейсу користувача
Controller	API Routes, Server Actions	Обробка запитів та бізнес-логіка
Model	SQLite база даних, TypeScript типи	Зберігання та структура даних

Подана структура відображає класичний поділ MVC-моделі. Рівень представлення (View) відповідає за відображення інтерфейсу користувача та реалізується за допомогою React-компонентів і сторінок Next.js. Контролерний рівень (Controller) забезпечує обробку запитів користувача, реалізацію бізнес-логіки та взаємодію між інтерфейсом і даними через API Routes та Server Actions. Модель (Model) відповідає за структуру та збереження даних і реалізується через SQLite-базу даних та TypeScript-типи. Такий багаторівневий розподіл дозволяє мінімізувати зв’язність між компонентами системи та забезпечує незалежність окремих шарів. Завдяки цьому спрощується тестування, модифікація та подальше розширення функціональності платформи.

Загальна компонентна архітектура системи відображає взаємодію між клієнтською частиною, серверною логікою та рівнем збереження даних (рис. 2.3).

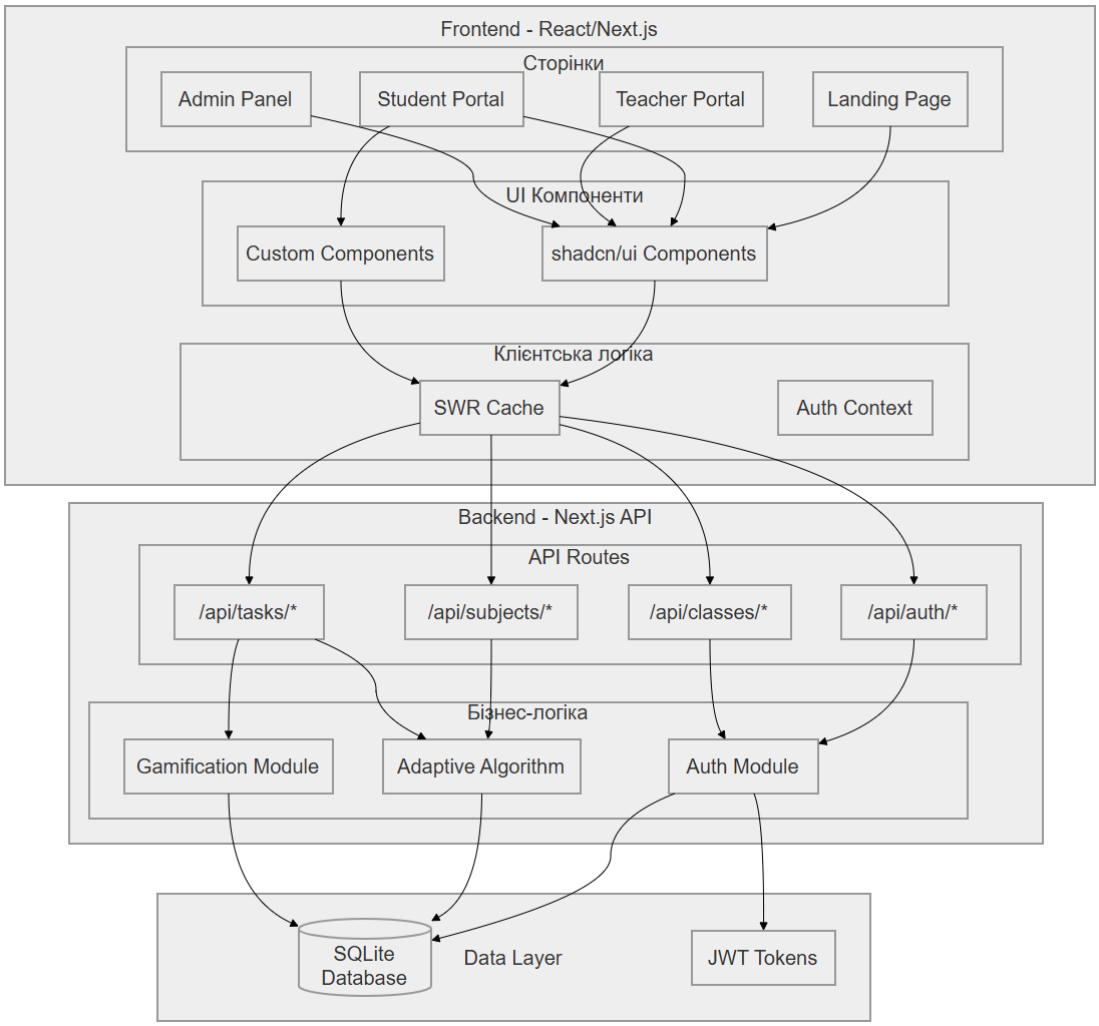


Рисунок 2.3 – Компонентна діаграма архітектури системи EduQuest

На діаграмі показано багаторівневу структуру системи, у якій фронтенд реалізує інтерфейс користувача на основі React і Next.js. Інтерфейс розділено відповідно до ролей користувачів – учня, вчителя та адміністратора, що забезпечує персоналізований доступ до функціоналу системи. Клієнтський рівень взаємодіє із серверною частиною через API-запити, використовуючи механізми керування станом та кешування даних. Це дозволяє зменшити кількість зайвих запитів до сервера та підвищити продуктивність системи. Серверний рівень представлений API Routes, які реалізують основну бізнес-логіку системи, зокрема автентифікацію користувачів, адаптивне навчання та гейміфікаційні механізми. Він забезпечує обробку запитів, координацію між модулями та взаємодію з базою даних.

Рівень даних у системі EduQuest реалізує збереження та захист усієї інформації, що використовується у навчальному процесі. Він базується на використанні бази даних SQLite, яка забезпечує структуроване зберігання даних про користувачів, навчальний контент, результати виконання завдань, прогрес навчання та гейміфікаційні елементи. Додатково для безпечної взаємодії із системою застосовується механізм JWT, який забезпечує автентифікацію та контроль доступу користувачів.

Загальна структура бази даних відображена на ER-діаграмі (рис. 2.4), яка ілюструє взаємозв’язки між основними сутностями системи.

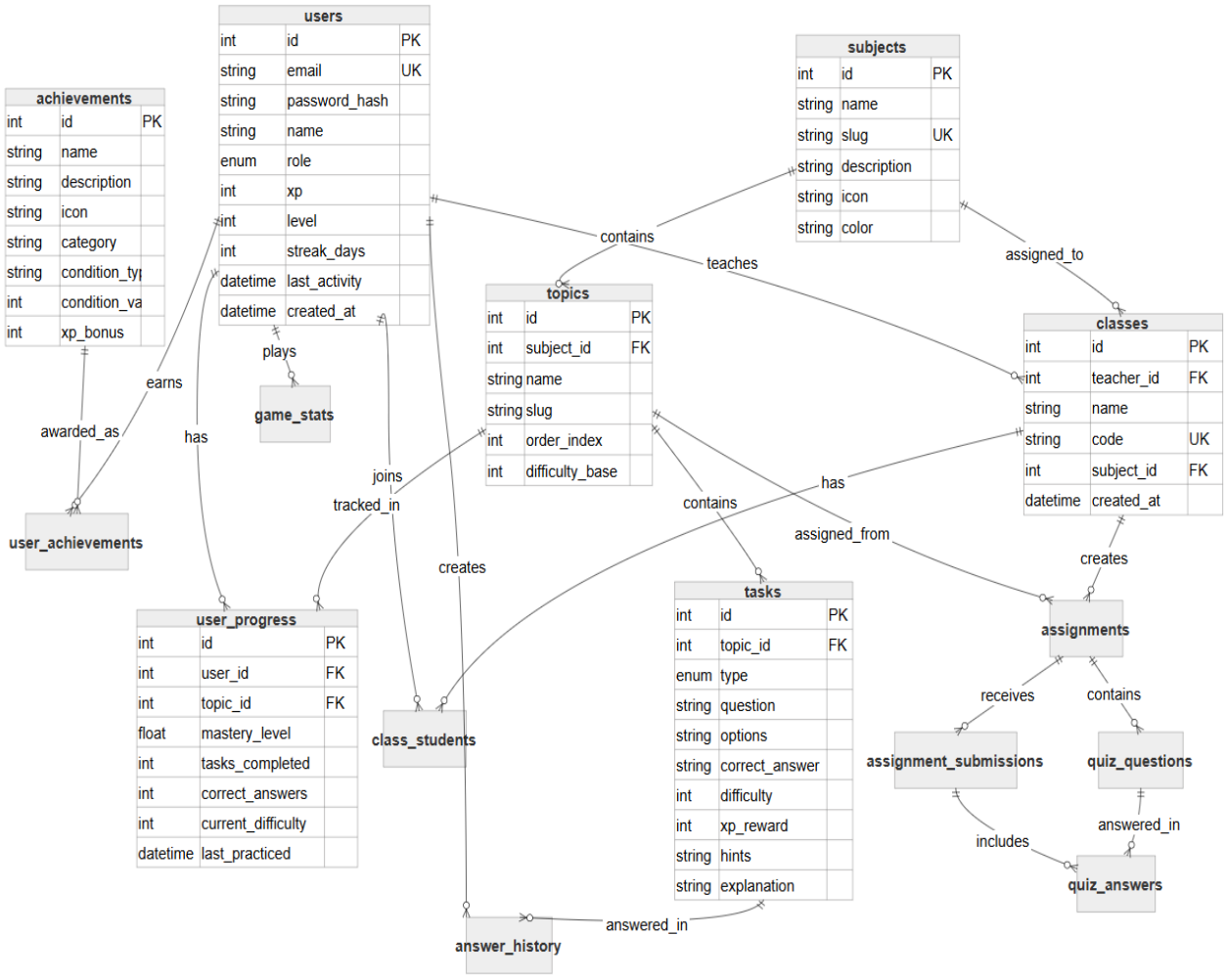


Рисунок 2.4 – ER-діаграма бази даних системи EduQuest

Центральною сутністю виступає користувач, який пов’язаний із навчальним прогресом, класами, виконаними завданнями та системою

досягнень. Така організація даних дозволяє забезпечити цілісність інформації та підтримку персоналізованого навчального процесу. Завдання у системі формують основу навчального контенту та прив’язані до конкретних тем, що забезпечує логічну структуру подання матеріалу. На основі результатів їх виконання формується запис прогресу користувача, який дозволяє відстежувати рівень засвоєння знань і формувати адаптивну траєкторію навчання. Гейміфікаційна складова, що включає XP, рівні та досягнення, інтегрована безпосередньо в модель даних через зв’язки з користувачем, що дозволяє реалізувати механізми мотивації та візуалізації навчального прогресу.

Узагальнена модель дозволяє виділити ключові таблиці бази даних, які реалізують відповідні сутності та забезпечують функціонування всіх основних модулів системи.

Наприклад, структура таблиці користувачів (табл. 2.6) відображає ключові атрибути облікового запису:

Таблиця 2.6 – Таблиця користувачів (users)

Поле	Тип	Обмеження	Опис
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор користувача
email	TEXT	UNIQUE	Електронна пошта користувача
password_hash	TEXT	NOT NULL	Хеш пароля
name	TEXT	NOT NULL	Ім’я користувача
role	TEXT	CHECK (student, teacher, admin)	Роль користувача
xp	INTEGER	DEFAULT 0	Очки досвіду
level	INTEGER	DEFAULT 1	Рівень користувача
streak_days	INTEGER	DEFAULT 0	Серія активності

У таблиці users акумульовано ключові атрибути облікового запису, що дозволяють реалізувати повний цикл взаємодії користувача із системою – від автентифікації до персоналізованого відображення навчального прогресу. Зокрема, поля xp, level та streak_days безпосередньо підтримують механізми гейміфікації, стимулюючи регулярну навчальну активність та підвищення залученості користувача.

Наступною важливою структурною одиницею є таблиця навчальних завдань (табл. 2.7), яка формує змістовне наповнення освітнього процесу та є основою для реалізації адаптивного навчання.

Таблиця 2.7 – Таблиця завдань (tasks)

Поле	Тип	Обмеження	Опис
id	INTEGER	PRIMARY KEY	Унікальний ідентифікатор завдання
topic_id	INTEGER	FOREIGN KEY	Посилання на тему
type	TEXT	NOT NULL	Тип завдання
question	TEXT	NOT NULL	Умова завдання
correct_answer	TEXT	NOT NULL	Правильна відповідь
difficulty	INTEGER	1–10	Рівень складності
xp_reward	INTEGER	DEFAULT 10	Нагорода XP

Таблиця tasks містить структуровані одиниці навчального контенту, які використовуються для формування завдань різних типів і рівнів складності. Завдяки полю difficulty система може реалізовувати адаптивний підбір завдань, а xp_reward забезпечує інтеграцію з гейміфікаційним модулем через нарахування винагороди за виконання.

Для відстеження результативності навчання використовується таблиця прогресу користувачів (табл. 2.8), яка відображає динаміку засвоєння матеріалу в розрізі окремих тем.

Таблиця 2.8 – Таблиця прогресу користувачів (user_progress)

Поле	Тип	Опис
user_id	INTEGER	Ідентифікатор користувача
topic_id	INTEGER	Ідентифікатор теми
mastery_level	REAL	Рівень засвоєння матеріалу
tasks_completed	INTEGER	Кількість виконаних завдань

Таблиця є індикатором навчальної активності користувача та дозволяє формувати адаптивну траєкторію навчання. Показник mastery_level використовується системою для оцінювання рівня засвоєння теми, тоді як

tasks_completed відображає фактичну залученість користувача до виконання навчальних завдань.

Одним із ключових елементів системи є адаптивний алгоритм (табл. 2.9), який забезпечує динамічну зміну складності навчальних завдань відповідно до рівня підготовки користувача. Його параметрична модель визначає чутливість системи до змін успішності та дозволяє підтримувати оптимальний баланс між складністю та доступністю матеріалу.

Таблиця 2.9 – Параметри адаптивного алгоритму підбору складності

Параметр	**Значення**	**Опис**
ADAPTATION_RATE	0.3	Коефіцієнт швидкості зміни складності завдань залежно від результатів учня
WINDOW_SIZE	10	Кількість останніх відповідей, що використовуються для аналізу успішності
TARGET_SUCCESS_RATE	0.7	Цільовий рівень успішності (70%), до якого система прагне адаптувати складність
MIN_DIFFICULTY	1	Мінімально допустимий рівень складності завдання
MAX_DIFFICULTY	10	Максимально допустимий рівень складності завдання

Параметр ADAPTATION_RATE визначає інтенсивність реакції системи на зміну результатів користувача, забезпечуючи поступову адаптацію без різких коливань складності. WINDOW_SIZE задає обсяг історичних даних, що враховуються при аналізі, завдяки чому зменшується вплив випадкових результатів. TARGET_SUCCESS_RATE (70%) виступає цільовим орієнтиром, до якого система автоматично підлаштовує рівень складності завдань. Обмеження MIN_DIFFICULTY та MAX_DIFFICULTY гарантують стабільність алгоритму та запобігають виходу параметра складності за допустимі межі.

Математично процес адаптації можна подати у вигляді узагальненої залежності:

$$D_{new} = D_{current} + (SR - 0.7) * 0.3 * 10 + B_{streak} + B_{time}, \qquad (2.1)$$

де D_{new} – новий рівень складності;

$D_{current}$ – поточний рівень складності;

SR – частка правильних відповідей (success rate);

B_{streak} – бонус за серію правильних відповідей;

B_{time} – бонус за швидкість виконання завдань.

Запропонована модель дозволяє враховувати не лише загальну успішність користувача, але й його поведінкові характеристики, що забезпечує більш точну адаптацію навчального процесу.

Параметри, які визначають роботу алгоритму, наведено в таблиці 2.10.

Таблиця 2.10 – Параметри адаптивного алгоритму

Параметр	**Значення**	**Вплив**
ADAPTATION_RATE	0.3	Плавність зміни складності
WINDOW_SIZE	10	Кількість відповідей для аналізу
TARGET_SUCCESS_RATE	0.7	Оптимальний баланс виклику
Streak Bonus	+0.5	Заохочення серій відповідей
Time Bonus	+0.3	Заохочення швидкості

Параметр ADAPTATION_RATE визначає швидкість зміни складності та забезпечує плавну адаптацію без різких стрибків рівня завдань.

WINDOW_SIZE задає кількість останніх відповідей, які використовуються для аналізу успішності користувача. У системі враховуються останні 10 відповідей, що дозволяє оперативно реагувати на зміни результатів.

TARGET_SUCCESS_RATE встановлено на рівні 70%, оскільки саме такий показник вважається оптимальним балансом між складністю та доступністю навчального матеріалу.

Додатково передбачені бонуси за серії правильних відповідей та швидке виконання завдань, що стимулює активність користувача та підтримує мотивацію до навчання.

Загальна логіка роботи алгоритму відображена на схемі (рис. 2.5). Вона демонструє послідовність обробки навчальних даних та прийняття рішень щодо зміни складності завдань.

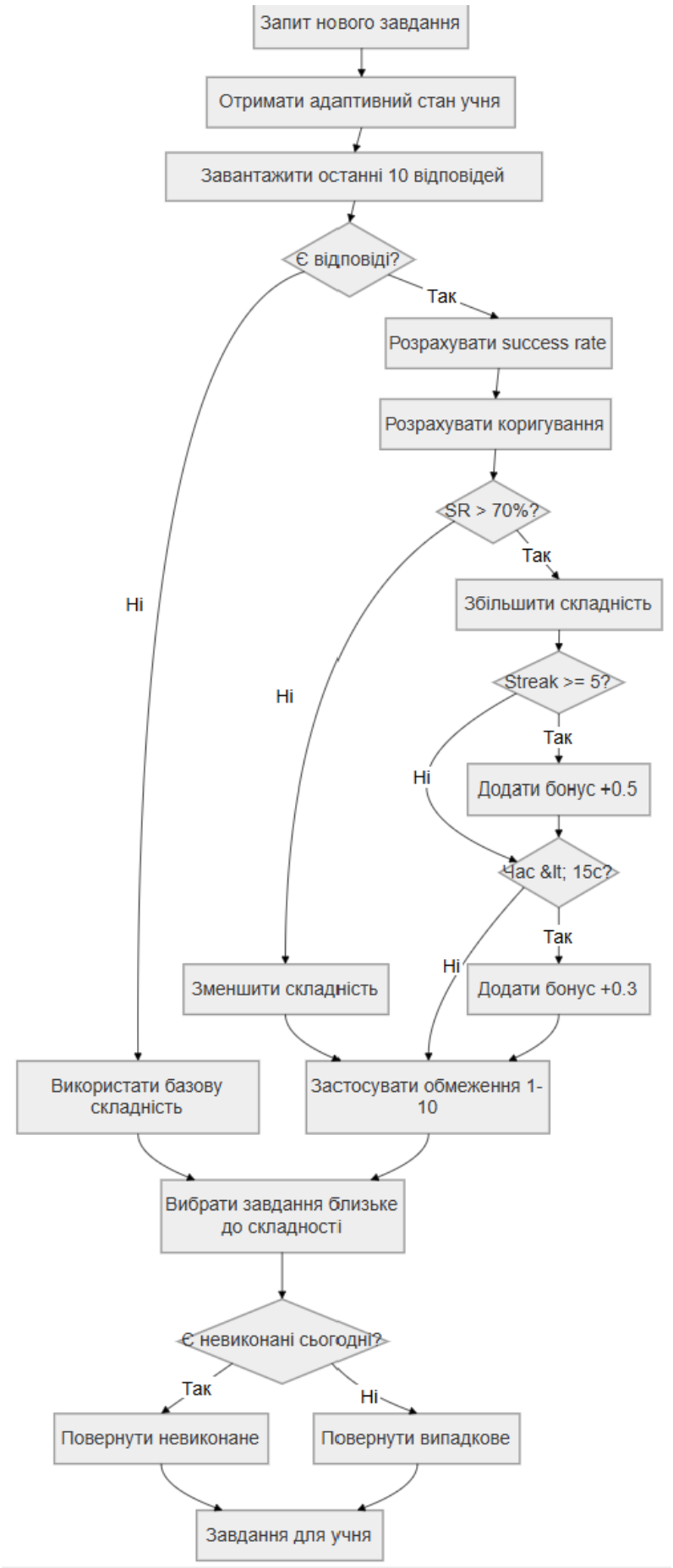


Рисунок 2.5 – Схема роботи адаптивного алгоритму підбору складності

Спочатку система ініціює запит на формування нового навчального завдання та завантажує адаптивний стан користувача, який включає поточний рівень складності, історію відповідей і параметри навчальної активності. Далі здійснюється аналіз останніх результатів навчання – враховуються до 10 попередніх відповідей користувача, що дозволяє оцінити його актуальний рівень підготовки та зменшити вплив випадкових помилок або одиничних успіхів.

У разі наявності даних обчислюється показник success rate, після чого визначається напрямок адаптації. Якщо рівень успішності перевищує 70%, складність завдань поступово підвищується, у протилежному випадку – знижується.

Додатково враховуються поведінкові фактори: серії правильних відповідей (streak) та швидкість виконання завдань. За наявності довгих серій правильних відповідей додається бонус до складності, а швидке виконання завдань додатково підсилює адаптаційний ефект.

Після цього система застосовує обмеження діапазону складності (1–10), що забезпечує стабільність алгоритму та запобігає виходу значень за допустимі межі. Далі виконується підбір завдання, максимально наближеного до розрахованого рівня складності, з урахуванням пріоритету невиконаних завдань поточного дня.

У результаті користувач отримує персоналізоване завдання, яке відповідає його поточному рівню підготовки та забезпечує адаптивну траєкторію навчання.

Отже, у межах роботи здійснено проєктування архітектури вебсистеми навчання та визначено принципи взаємодії її основних компонентів. Сформовано інформаційну модель предметної області з виділенням ключових сутностей і зв’язків між ними. Розроблені структура бази даних та архітектурне рішення забезпечують логічну цілісність системи, підтримку її масштабованості та ефективну організацію даних.

2.4 Опис потоків даних та взаємодії компонентів системи

У системі EduQuest організація потоків даних побудована за принципом багаторівневої архітектури з чітким розмежуванням відповідальності між клієнтською частиною, серверною логікою та рівнем збереження даних. Кожна дія користувача ініціює послідовність запитів, які проходять через визначені шари обробки, трансформуються бізнес-логікою та повертаються у вигляді оновленого стану інтерфейсу. Такий підхід дозволяє забезпечити передбачуваність роботи системи, спростити масштабування та підвищити стабільність взаємодії між компонентами.

Загальна структура потоків даних у системі відображає взаємодію між основними рівнями – клієнтським інтерфейсом, мережею, серверною логікою та базою даних. Важливою особливістю є використання проміжного шару кешування (SWR), який забезпечує швидкий доступ до даних і мінімізує кількість повторних запитів до сервера. Загальна логіка взаємодії компонентів та напрямки передачі даних подані на рисунку 2.6.

Подана схема демонструє односпрямований потік даних у системі, де кожен рівень архітектури виконує чітко визначену функціональну роль. На рівні клієнта формується запит користувача, який ініціює взаємодію із системою. Далі цей запит передається на серверний рівень, де відбувається його первинна обробка та маршрутизація до відповідного компонента бізнес-логіки. Рівень бізнес-логіки виконує основні обчислення, включаючи перевірку умов, обробку даних користувача та реалізацію прикладних алгоритмів системи, зокрема адаптивного навчання та гейміфікаційних механізмів. Після завершення обробки результати передаються до рівня збереження даних, де здійснюється їх запис або оновлення у базі даних. Зворотний потік даних забезпечує повернення результатів обробки назад до клієнтського рівня. Це дозволяє синхронізувати інтерфейс користувача з актуальним станом системи, забезпечуючи відображення оновленого прогресу, результатів виконання завдань та інших змін

у реальному часі. Тобто схема ілюструє цілісність взаємодії між усіма рівнями системи та забезпечення безперервного циклу обробки даних.

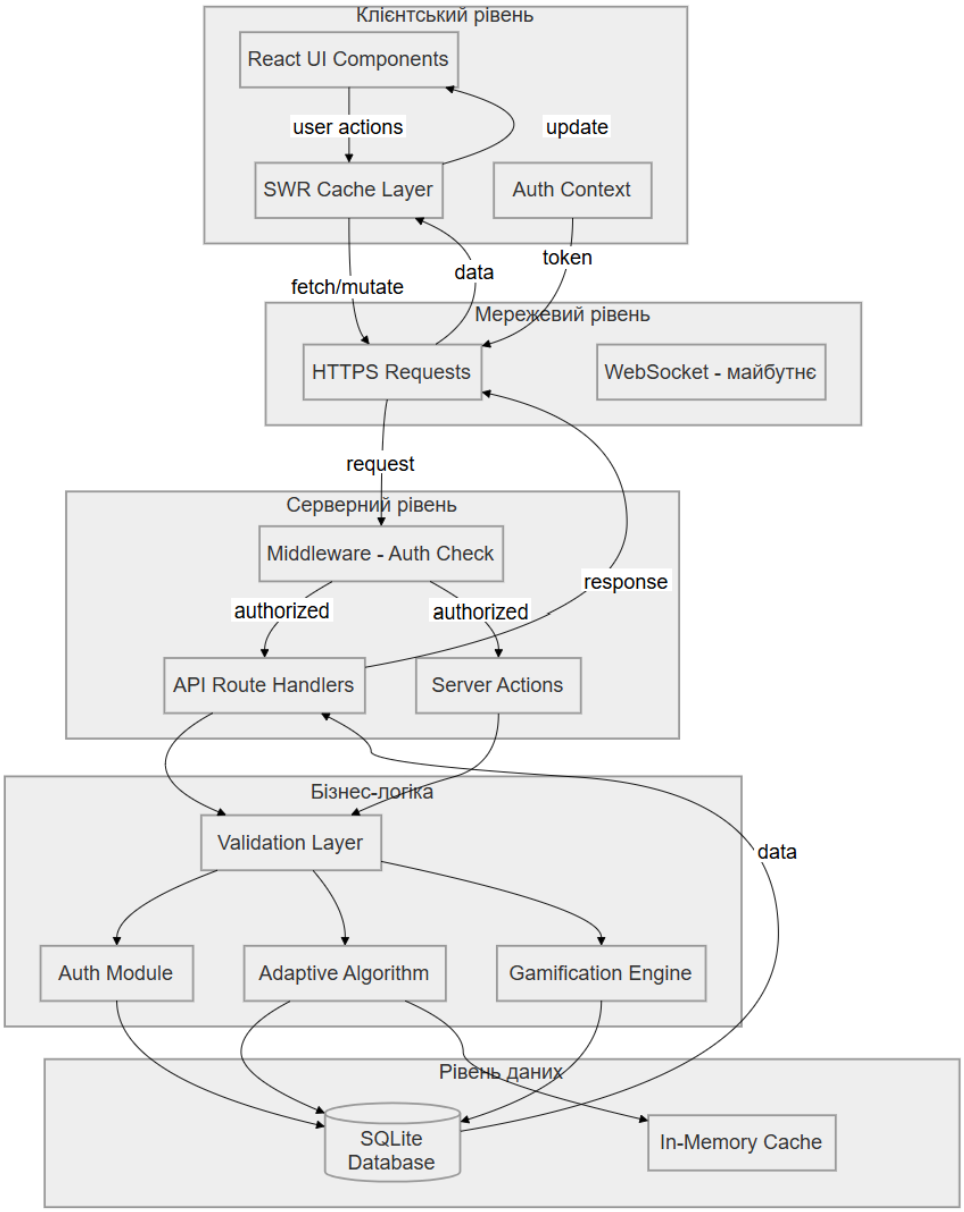


Рисунок 2.6 – Загальна архітектура потоків даних у системі EduQuest

Подальшим важливим процесом у системі є автентифікація користувача, яка забезпечує захист доступу до функціоналу платформи та є базовим етапом будь-якої взаємодії з API. Після введення облікових даних користувачем запит проходить перевірку на клієнтському рівні, після чого передається на сервер, де відбувається валідація та звірка з базою даних. У разі успішної перевірки

генерується JWT-токен, який повертається клієнту та зберігається в захищеному cookie. Детальна послідовність цього процесу наведена на рисунку 2.7.

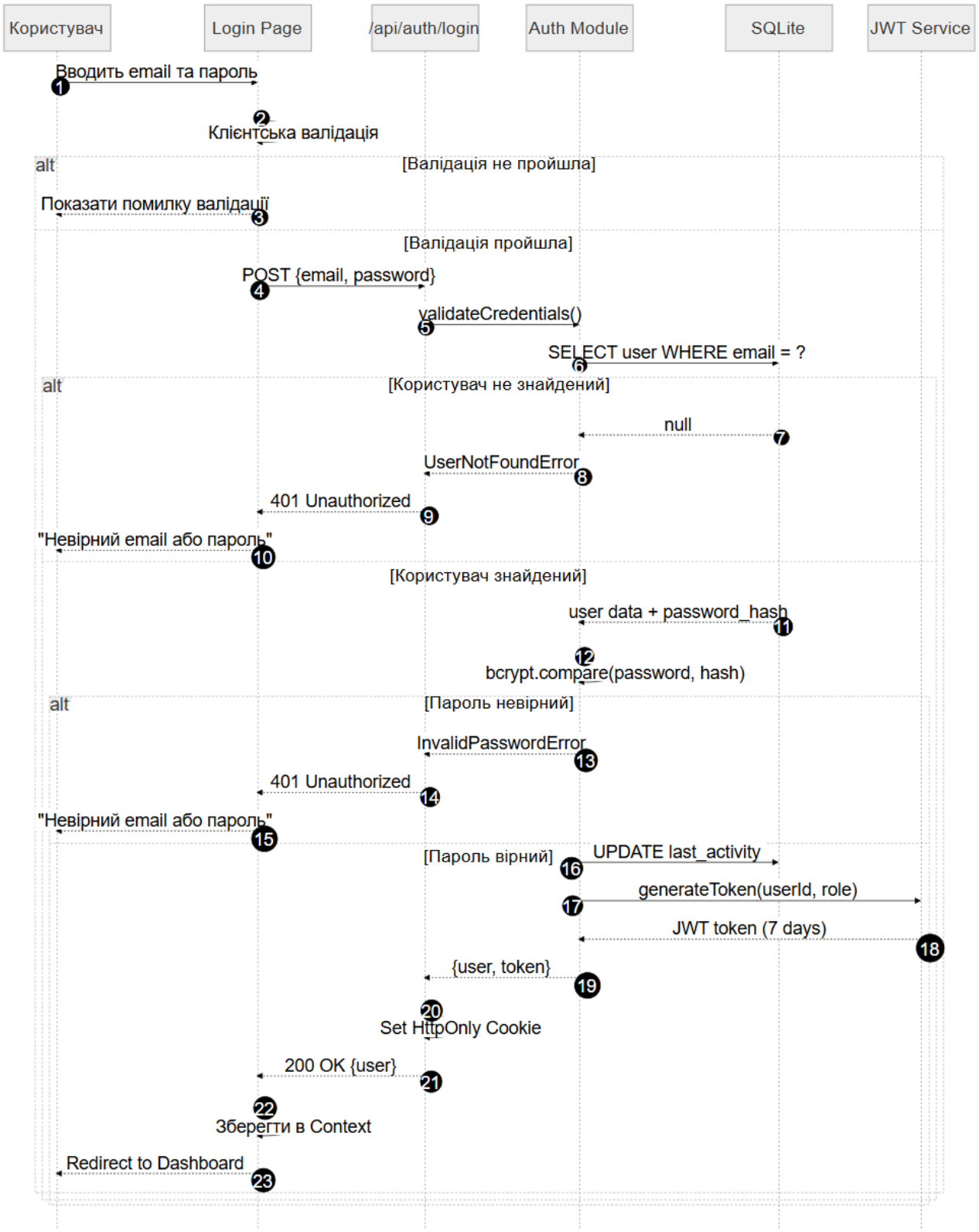


Рисунок 2.7 – Потік автентифікації користувача в системі EduQuest

У результаті автентифікації система створює захищений сеанс користувача, що дозволяє йому взаємодіяти з іншими модулями платформи без повторного введення даних.

Наступним ключовим процесом є адаптивне навчання, яке забезпечує персоналізацію навчального контенту відповідно до рівня знань користувача. При зверненні до певної теми система спочатку перевіряє наявність кешованих даних, після чого, у разі необхідності, виконує запит до сервера. Там відбувається аналіз прогресу користувача, обчислення рівня складності та вибір оптимального завдання. Цей процес дозволяє динамічно адаптувати навчальний контент і підтримувати оптимальний рівень складності. Візуалізація цього процесу представлена на рисунку 2.8.

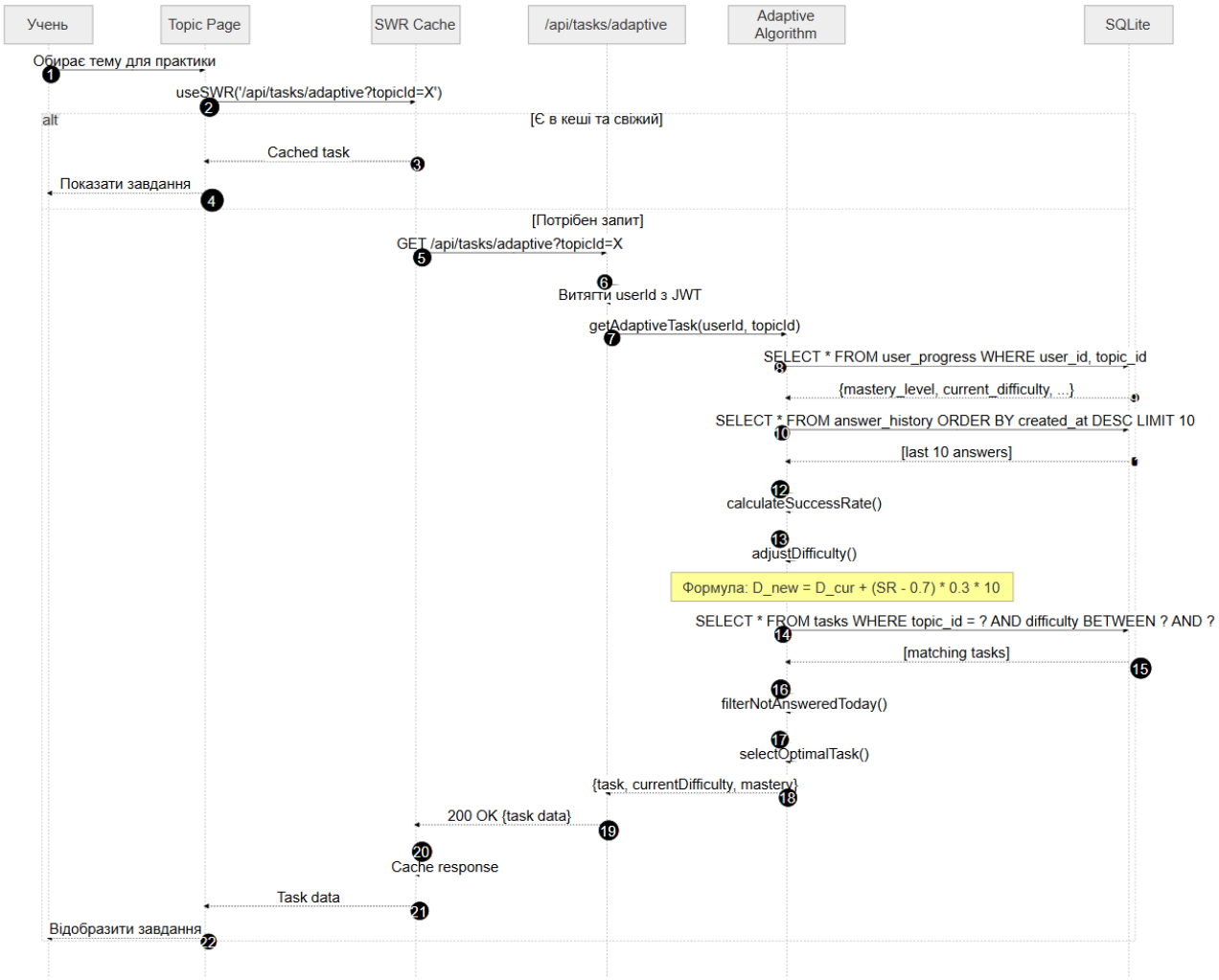


Рисунок 2.8 – Потік адаптивного навчання та підбору завдань

Особливу роль у системі відіграє механізм перевірки відповідей та нарахування результатів, оскільки він об’єднує навчальну логіку з гейміфікацією. Після надсилання відповіді система перевіряє її правильність, оновлює прогрес користувача, нараховує XP та перевіряє можливість отримання нових рівнів або досягнень. Цей процес має комплексний характер і забезпечує миттєвий зворотний зв’язок для користувача. Його структура наведена на рисунку 2.9.

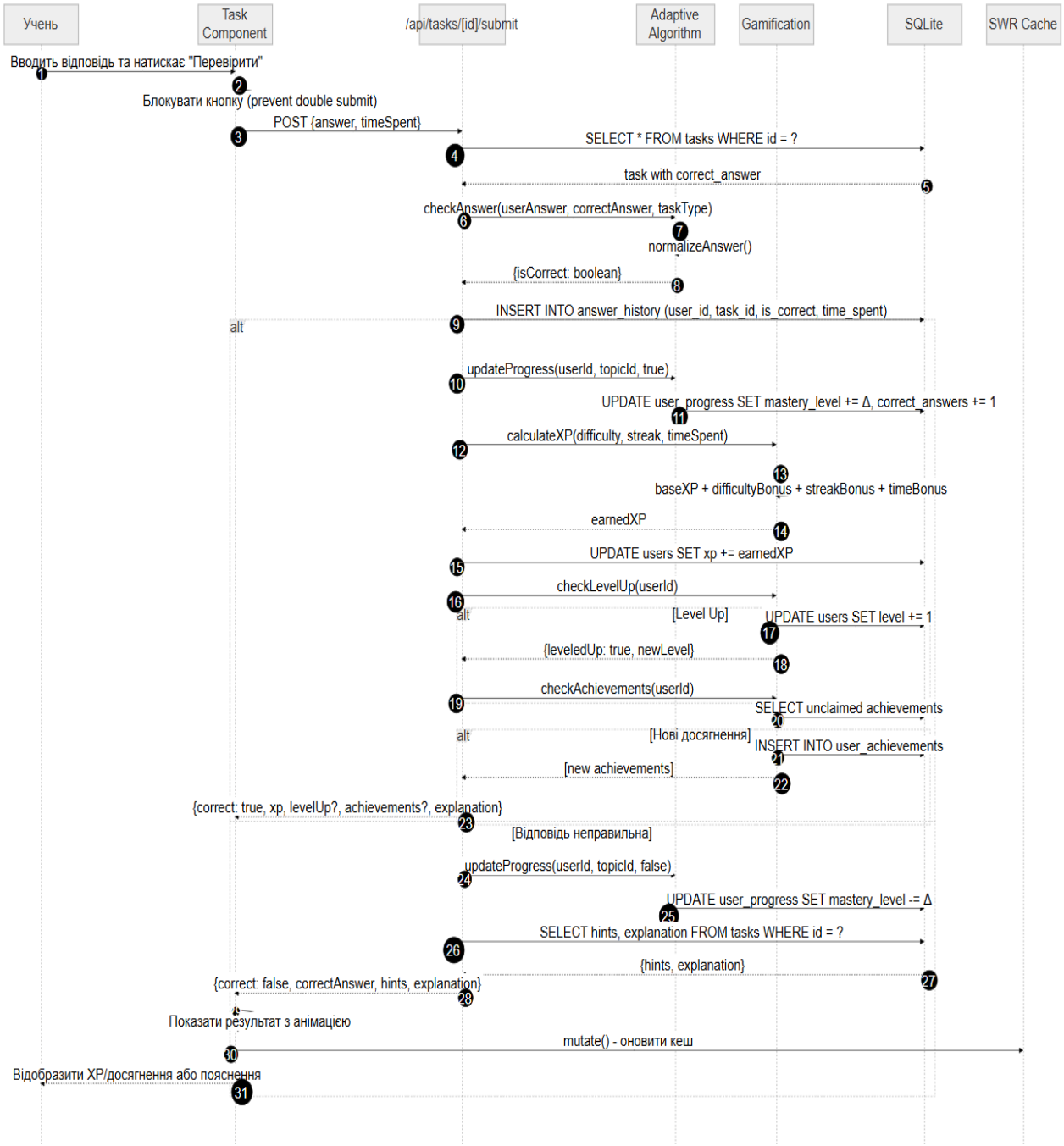


Рисунок 2.9 – Потік перевірки відповіді та нарахування XP

Завершальним аспектом взаємодії компонентів є синхронізація даних між клієнтом і сервером, яка реалізована через SWR-механізм (рис. 2.10).

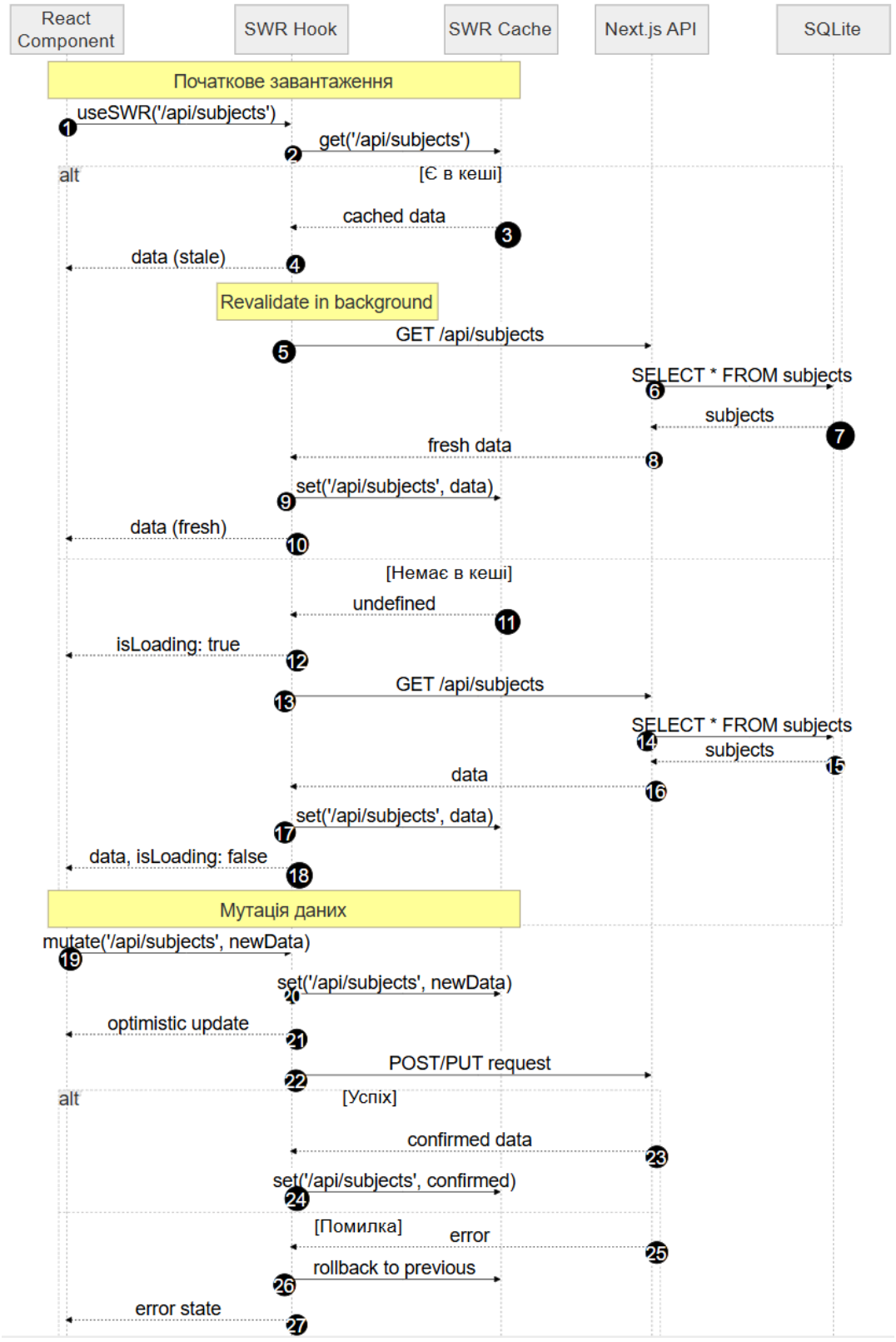


Рисунок 2.10 – Механізм синхронізації даних через SWR

На етапі початкового завантаження React-компонент ініціює запит через useSWR, після чого SWR перевіряє наявність даних у локальному кеші. Якщо дані вже збережені, вони одразу повертаються до інтерфейсу як «stale», забезпечуючи швидке відображення сторінки, паралельно запускаючи фонове оновлення з сервера. У випадку відсутності кешу система переходить у стан завантаження та звертається до API Next.js, який отримує дані з бази даних SQLite і повертає їх назад через SWR із подальшим збереженням у кеші.

Окремо показано механізм мутації даних, який дозволяє виконувати оптимістичне оновлення інтерфейсу. У цьому випадку нові дані спочатку одразу відображаються користувачу та записуються в кеш, після чого надсилається запит на сервер для їх підтвердження. У разі успішного виконання сервер повертає підтверджені дані, які остаточно синхронізуються з кешем. Якщо виникає помилка, система виконує відкат до попереднього стану та повідомляє користувача про збій.

У підсумку, описані потоки даних демонструють цілісну та узгоджену архітектуру системи EduQuest, у якій кожен компонент виконує чітко визначену роль. Завдяки цьому забезпечується ефективна взаємодія між модулями, висока продуктивність системи та можливість її подальшого розширення без порушення існуючої структури.

2.3 Обґрунтування вибору технологічного стеку розробки

При виборі технологічного стеку для системи EduQuest було враховано низку ключових критеріїв, що визначають якість сучасних вебзастосунків: продуктивність, масштабованість, швидкість розробки, підтримуваність коду, зрілість екосистеми, а також витрати на розгортання та супровід системи. Окрему увагу приділено можливості довгострокового розвитку платформи без необхідності кардинальної зміни архітектури.

Загалом розглядалися різні альтернативи для кожного рівня системи. Для фронтенд-частини як основні варіанти порівнювалися React у класичній SPA-

реалізації (наприклад, через Vite), Angular та Next.js. Хоча Angular забезпечує повну «з коробки» архітектуру, він є більш громіздким та складнішим для швидкої розробки [21]. React із Vite забезпечує високу швидкість, однак не вирішує задачі серверного рендерингу та маршрутизації на рівні фреймворку [22]. У результаті обрано Next.js як більш комплексне рішення, що поєднує SSR, SSG, API Routes та сучасну файлову маршрутизацію, що особливо важливо для освітньої платформи з динамічним контентом [23].

Для серверної логіки розглядалися варіанти окремого бекенду (Node.js + Express/NestJS) та інтегрованого підходу в межах Next.js API Routes [24]. Окремий бекенд забезпечує більшу гнучкість у великих системах, однак збільшує складність інфраструктури та розгортання. З урахуванням масштабу EduQuest було обрано інтегрований підхід, що дозволяє зменшити кількість сервісів і спростити підтримку системи.

Як систему управління базою даних порівнювалися SQLite, PostgreSQL та MySQL. PostgreSQL є найбільш функціональним рішенням для великих навантажень, однак потребує окремого серверного розгортання та складнішої конфігурації [25]. MySQL є поширеним рішенням, але поступається PostgreSQL у частині розширених можливостей [26]. SQLite обрано як оптимальний варіант для навчальної та середньонавантаженої системи завдяки відсутності необхідності серверної інфраструктури, простоті інтеграції та достатньому рівню продуктивності [27].

Для побудови інтерфейсу користувача розглядалися Material UI [28], Ant Design [29] та shadcn/ui [30]. Важкі UI-бібліотеки (Material UI, Ant Design) забезпечують велику кількість готових компонентів, однак створюють надлишковий розмір бандлу та обмежують гнучкість кастомізації. Натомість shadcn/ui у поєднанні з Tailwind CSS [31] дозволяє створювати легкий, повністю контрольований інтерфейс із високим рівнем адаптивності та мінімальними залежностями.

Для автентифікації розглядалися сесійна модель (server sessions), OAuth-провайдери та JWT-підхід. Сесійна модель потребує зберігання стану на сервері, що ускладнює масштабування [32], тоді як Oauth [33] залежить від зовнішніх

сервісів. JWT було обрано як компроміс між безпекою, простотою реалізації та масштабованістю, оскільки він дозволяє реалізувати stateless-автентифікацію [32].

У процесі розробки також використовувалися SWR для кешування та синхронізації даних [34], TypeScript для забезпечення типобезпеки [35], а також сучасні інструменти тестування (Jest, React Testing Library, Playwright), що дозволяють реалізувати багаторівневу перевірку якості програмного продукту [36].

Розгортання системи здійснюється на платформі Vercel [37], яка забезпечує оптимізовану підтримку Next.js, автоматичний CI/CD та CDN-доставку контенту, що позитивно впливає на продуктивність і доступність системи. Запуск та написання коду здійснювався в Visual Code [38].

Узагальнені дані аналізу технологій для розробки вебсистеми наведено у таблиці 2.11.

Таблиця 2.11 – Порівняльна таблиця технологічного стеку

Компонент системи	**Обране рішення**	**Альтернативи**	**Причина вибору**
Frontend	Next.js	React SPA (Vite), Angular	SSR/SSG, маршрутизація, API Routes, продуктивність
Backend	Next.js API Routes	Express, NestJS	Менша складність, єдина кодова база
База даних	SQLite	PostgreSQL, MySQL	Простота, відсутність серверу, достатня продуктивність
UI бібліотека	shadcn/ui + Tailwind	Material UI, Ant Design	Легкість, кастомізація, мінімальний бандл
Auth	JWT + bcrypt	Sessions, OAuth	Stateless, масштабованість, незалежність
State/Data fetching	SWR	Redux, React Query	Простота, кешування, авто-рефетч
Мова	TypeScript	JavaScript	Типобезпека, менше помилок
Тестування	Jest + RTL + Playwright	Cypress, Mocha	Покриття unit + integration + E2E
Деплой	Vercel	AWS, VPS	Оптимізація під Next.js, CI/CD

У результаті проведеного аналізу та порівняння альтернативних технологічних рішень було обґрунтовано вибір оптимального технологічного стеку для розробки системи EduQuest. Встановлено, що обрані інструменти забезпечують найкращий баланс між продуктивністю, простотою реалізації, масштабованістю та зручністю подальшого супроводу системи.

Використання Next.js дозволяє поєднати клієнтську та серверну логіку в межах єдиної архітектури, SQLite забезпечує просте та ефективне зберігання даних без складної інфраструктури, а TypeScript підвищує надійність коду завдяки статичній типізації. Додатково SWR забезпечує ефективне кешування та синхронізацію даних, що покращує користувацький досвід.

Отже, обраний технологічний стек повністю відповідає вимогам системи EduQuest, забезпечує її стабільну роботу та створює умови для подальшого розвитку й масштабування без суттєвих змін у базовій архітектурі.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБСИСТЕМИ

3.1 Реалізація серверної частини системи

Серверна частина системи EduQuest реалізована з використанням Next.js API Routes, що дозволяє поєднати клієнтську та серверну логіку в межах одного проєкту. Такий підхід забезпечує спрощення розробки, централізовану структуру коду та ефективну взаємодію між усіма компонентами системи. Архітектура побудована за принципом модульності, де кожен функціональний компонент відповідає за окрему область логіки та може розвиватися незалежно від інших модулів (табл. 3.1).

Таблиця 3.1 – Структура серверних модулів системи

Модуль	Файл	Призначення
База даних	lib/db.ts	Ініціалізація SQLite, схема таблиць
Автентифікація	lib/auth.ts	JWT токени, хешування паролів, сесії
Адаптивний алгоритм	lib/adaptive-algorithm.ts	Підбір складності завдань
Досягнення	lib/achievements.ts	Перевірка та нагородження

Модуль бази даних забезпечує створення локального сховища даних, підключення до SQLite та ініціалізацію структури таблиць. У ньому також реалізуються оптимізації продуктивності та механізми взаємодії з даними.

Модуль автентифікації відповідає за безпечну роботу з обліковими записами користувачів. Він реалізує механізми хешування паролів, генерації JWT-токенів, створення сесій та перевірки прав доступу.

Модуль адаптивного алгоритму виконує аналіз результатів користувача та визначає оптимальний рівень складності наступних завдань. Алгоритм враховує успішність виконання, швидкість відповідей та серії правильних результатів.

Модуль досягнень реалізує логіку гейміфікації, що включає перевірку виконання умов отримання нагород, відкриття нових досягнень та оновлення прогресу користувача.

Модуль бази даних реалізовано на основі SQLite з використанням бібліотеки better-sqlite3. Такий підхід забезпечує швидкий доступ до даних, простоту розгортання та відсутність необхідності у виділеному сервері бази даних. Код ініціалізації бази даних наведено у додатку А.1.

У процесі запуску система автоматично створює файл бази даних eduquest.db, після чого виконується ініціалізація структури таблиць та допоміжних індексів. Для підвищення продуктивності активується режим WAL (Write-Ahead Logging), який дозволяє одночасно виконувати операції читання та запису без блокування всієї бази даних (табл. 3.2).

Таблиця 3.2 – Ключові особливості реалізації бази даних

Особливість	**Опис**	**Перевага**
WAL режим	Write-Ahead Logging	Паралельне читання та запис
Prepared statements	Підготовлені запити	Захист від SQL-ін’єкцій
Індекси	На часто використовуваних полях	Прискорення запитів
FOREIGN KEY	Зв’язки між таблицями	Цілісність даних

У системі використовується WAL-режим журналювання, який дозволяє виконувати операції запису без блокування операцій читання. Це особливо важливо для навчальної платформи, де велика кількість користувачів може одночасно проходити завдання та переглядати статистику. Prepared statements застосовуються для підготовлених SQL-запитів, що дозволяє запобігти SQL-ін’єкціям та забезпечити безпечну роботу з вхідними даними користувача. Для пришвидшення доступу до інформації створено індекси на полях, які найчастіше використовуються під час пошуку та фільтрації даних. Зокрема, індексуються теми завдань, рівні складності, прогрес користувача та історія відповідей.

Також використовуються зовнішні ключі FOREIGN KEY, які забезпечують зв’язки між таблицями та підтримують цілісність даних під час оновлення або видалення записів.

Модуль автентифікації забезпечує безпечну роботу з користувацькими даними та контроль доступу до функціональних можливостей системи. Основні механізми реалізовані у файлі lib/auth.ts, код якого наведено у додатку А.2.

Під час реєстрації користувача пароль не зберігається у відкритому вигляді. Перед записом у базу даних він проходить процедуру хешування за допомогою алгоритму bcrypt із 12 раундами сольового шифрування. Це суттєво підвищує захищеність облікових даних.

Після успішної авторизації система генерує JWT-токен, який містить ідентифікатор користувача та його роль у системі. Токен використовується для перевірки прав доступу при кожному зверненні до API. Алгоритм bcrypt із 12 salt rounds забезпечує складне та ресурсоємне хешування паролів, що значно ускладнює можливість їх підбору або дешифрування. JWT-токени реалізовані за допомогою бібліотеки jose та алгоритму HS256. Stateless-підхід дозволяє серверу не зберігати окремі сесії користувачів у базі даних, що зменшує навантаження на систему.

Для збереження токенів використовуються HttpOnly cookie, які недоступні для JavaScript-коду на стороні браузера. Додатково використовується параметр sameSite: lax, що знижує ризик CSRF-атак. Термін дії токена становить 7 днів, після чого користувач повинен пройти повторну авторизацію.

Адаптивний алгоритм є одним із ключових компонентів системи EduQuest, оскільки саме він забезпечує персоналізацію навчального процесу та автоматичне налаштування складності завдань відповідно до рівня підготовки користувача. Реалізація алгоритму наведена у додатку А.3.

Алгоритм аналізує останні відповіді користувача, визначає показник успішності та виконує коригування рівня складності. Додатково враховуються швидкість виконання завдань та наявність серії правильних відповідей.

Система EduQuest містить набір REST API endpoint-ів, які забезпечують взаємодію між клієнтською частиною та серверною логікою (табл. 3.3). Маршрути організовані за функціональними модулями відповідно до їх призначення.

Таблиця 3.3 – Основні API endpoints системи

Група	Endpoint	Метод	Призначення
Auth	/api/auth/login	POST	Вхід користувача
Auth	/api/auth/register	POST	Реєстрація
Auth	/api/auth/logout	POST	Вихід
Auth	/api/auth/me	GET	Поточний користувач
Tasks	/api/tasks/next	GET	Наступне адаптивне завдання
Tasks	/api/tasks/submit	POST	Відправка відповіді
Progress	/api/progress	GET	Прогрес учня
Games	/api/games/submit	POST	Збереження результату гри
Classes	/api/teacher/classes	GET/POST	Управління класами
Admin	/api/admin/users	GET/PUT/DELETE	Управління користувачами

Маршрути групи Auth забезпечують реєстрацію користувачів, авторизацію, завершення сесії та отримання інформації про поточного користувача.

Модуль Tasks реалізує логіку адаптивного навчання. Один endpoint відповідає за отримання наступного завдання відповідно до поточного рівня користувача, а інший – за перевірку відповідей та оновлення прогресу.

Модуль Progress використовується для отримання статистики навчання, кількості XP, рівня користувача та інших показників прогресу.

Окремо реалізовано маршрути для навчальних ігор, управління класами вчителями та адміністративного управління користувачами системи.

Логіка одного з API Route для обробки відповідей користувача наведена у додатку А.4.

Отже, у процесі реалізації серверної частини системи EduQuest було сформовано модульну архітектуру, яка забезпечує стабільну взаємодію між базою даних, механізмами автентифікації, адаптивним алгоритмом та API-

маршрутами. Реалізовані серверні компоненти забезпечують безпечну обробку даних, персоналізацію навчального процесу та підтримку основної бізнес-логіки системи. Обрані технологічні рішення сприяють продуктивності, масштабованості та подальшому розширенню функціональних можливостей платформи.

3.2 Реалізація клієнтської частини та користувацького інтерфейсу

Клієнтська частина системи EduQuest реалізована на основі React 19 та Next.js App Router, що дозволяє створити сучасний вебінтерфейс із підтримкою компонентного підходу, серверного рендерингу та гнучкої маршрутизації. Використання Next.js забезпечує високу швидкість завантаження сторінок, оптимізацію взаємодії з серверною частиною та ефективну організацію структури проєкту.

Архітектура клієнтської частини побудована за принципом поділу функціональності відповідно до ролей користувачів системи. У структурі проєкту передбачено окремі модулі для учня, вчителя та адміністратора, що дозволяє ізолювати функціональні можливості та реалізувати різні сценарії роботи в межах єдиної платформи. Такий підхід підвищує масштабованість системи та спрощує подальше розширення функціоналу.

Кореневий каталог app/ містить глобальні елементи застосунку, зокрема головний layout та базові сторінки системи. Файл layout.tsx відповідає за формування спільної структури інтерфейсу, підключення глобальних стилів, теми оформлення та загальних компонентів навігації. Файл page.tsx реалізує головну сторінку платформи, яка виконує роль landing page та містить основну інформацію про систему.

Для реалізації процесів авторизації створено окремі сторінки входу та реєстрації (login/page.tsx і register/page.tsx). Вони забезпечують взаємодію користувача із серверними API для створення облікового запису та входу в

систему. Дані сторінки інтегровані з механізмом JWT-автентифікації та використовують форми з валідацією введених даних.

Окремий модуль student/ реалізує портал учня. У його структурі передбачено власний layout із боковою панеллю навігації, що забезпечує швидкий доступ до основних функцій системи. У межах цього модуля реалізовано сторінку дашборду, сторінки навчання, розділ навчальних ігор, систему досягнень та профіль користувача. Дашборд відображає статистику навчання, прогрес та поточні результати учня. Модуль навчання забезпечує проходження тем та адаптивний підбір завдань, а розділ досягнень реалізує гейміфікаційні механізми системи.

Для вчителів передбачено окремий портал teacher/, який містить функції управління навчальними класами, створення завдань та моніторингу успішності учнів. У цьому модулі реалізовано інструменти організації освітнього процесу, перегляду статистики класів та контролю навчального прогресу.

Адміністративний модуль admin/ забезпечує централізоване керування системою. Він містить сторінки управління користувачами, навчальними предметами та статистикою платформи. Завдяки цьому адміністратор може контролювати структуру навчального контенту, налаштовувати доступ користувачів та аналізувати загальну активність системи.

Важливою особливістю архітектури є використання вкладених layout-компонентів. Такий підхід дозволяє створювати єдину структуру навігації для кожної підсистеми окремо. Наприклад, усі сторінки учня використовують спільний sidebar, верхню панель та однакову структуру інтерфейсу. Це забезпечує консистентність дизайну, спрощує навігацію та покращує користувацький досвід.

Для реалізації навчального процесу використовується механізм динамічної маршрутизації Next.js. Зокрема, маршрут learn/[topicId] дозволяє автоматично формувати сторінки для різних навчальних тем на основі параметра URL. Завдяки цьому система може динамічно завантажувати контент відповідно до обраної теми без необхідності створення окремих сторінок вручну. Такий підхід

забезпечує гнучкість структури навчального контенту та значно спрощує підтримку системи.

Архітектура клієнтської частини також орієнтована на повторне використання компонентів. Базові елементи інтерфейсу, такі як кнопки, картки, модальні вікна та елементи навігації, реалізовані у вигляді окремих React-компонентів. Це дозволяє підтримувати єдиний стиль інтерфейсу та зменшує дублювання коду в різних частинах системи.

Загалом обрана архітектура клієнтської частини забезпечує модульність, масштабованість та зручність супроводу системи EduQuest. Поєднання React 19, Next.js App Router та компонентного підходу дозволяє створити швидкий, адаптивний та інтерактивний інтерфейс для всіх категорій користувачів платформи.

Для побудови інтерфейсу системи використано бібліотеку shadcn/ui, яка забезпечує набір готових React-компонентів із підтримкою Tailwind CSS та сучасних принципів доступності (табл. 3.4). Використання цієї бібліотеки дозволило створити уніфікований інтерфейс із можливістю глибокої кастомізації зовнішнього вигляду.

Таблиця 3.4 – Основні UI-компоненти системи

Компонент	**Бібліотека**	**Кастомізація**
Button	shadcn/ui	Варіанти для XP, success
Card	shadcn/ui	Стилі для статистики
Progress	shadcn/ui	Анімація, кольори
Dialog	shadcn/ui	Модальні вікна досягнень
Skeleton	shadcn/ui	Стани завантаження

Наведені компоненти використовуються для побудови всіх основних елементів інтерфейсу. Компонент Button реалізує інтерактивні елементи керування та підтримує різні стилі для навчальних дій, підтверджень або гейміфікаційних елементів. Компонент Card використовується для візуального групування статистики, прогресу та інформаційних блоків. Progress забезпечує

відображення рівня проходження тем або заповнення XP-шкали. Dialog використовується для модальних вікон із повідомленнями про нові досягнення чи підтвердження дій. Компонент Skeleton забезпечує відображення placeholder-станів під час завантаження даних із сервера.

Приклад реалізації дашборду учня наведений у додатку Б.1.

Компонент дашборду реалізує завантаження статистики користувача через API-запити та відображає її у вигляді карток із ключовими показниками. Для отримання даних використовується React hook useEffect, який виконує запит до endpoint /api/progress після завантаження сторінки. Поки дані не отримані, інтерфейс відображає skeleton-компоненти, що створюють плавний ефект завантаження та покращують сприйняття швидкодії системи.

Після отримання даних компонент формує адаптивну сітку статистики, яка містить кількість виконаних завдань, рівень користувача, XP та інші показники. Для покращення візуального сприйняття використовуються SVG-іконки бібліотеки Lucide React. Додатково реалізовано блок прогресу за предметами, де для кожного предмета відображається кількість пройдених тем, середній рівень засвоєння та прогрес-бар.

Особливістю реалізації є використання умовного рендерингу та динамічних стилів Tailwind CSS. Наприклад, колір предмета формується динамічно через inline-style, що дозволяє створювати унікальне оформлення для різних дисциплін. Також застосовується hover-анімація для інтерактивних карток, що покращує взаємодію користувача з інтерфейсом.

Для управління станом клієнтської частини та синхронізації даних із сервером у системі використовується бібліотека SWR. Даний підхід забезпечує автоматичне кешування даних, фонове оновлення та мінімізацію кількості повторних HTTP-запитів.

Приклад використання SWR наведений у додатку Б.2.

У реалізації використовується функція useSWR, яка автоматично виконує запит до API та зберігає отримані результати в локальному кеші. Функція fetcher забезпечує універсальний механізм отримання JSON-даних із сервера. Hook

useUserStats інкапсулює логіку отримання статистики користувача та повертає дані, статус завантаження і помилки.

Однією з важливих особливостей SWR є механізм revalidateOnFocus, який автоматично оновлює дані після повернення користувача до вкладки браузера. Додатково використовується параметр refreshInterval, що дозволяє періодично синхронізувати статистику кожні 30 секунд.

Механізм кешування дозволяє повторно використовувати вже отримані дані без повторного звернення до сервера, що суттєво зменшує навантаження на API. Фонове оновлення забезпечує актуальність інформації без необхідності перезавантаження сторінки. Optimistic Updates дозволяє миттєво змінювати стан інтерфейсу ще до завершення серверного запиту, що покращує користувацький досвід. Вбудована система обробки помилок дозволяє централізовано реагувати на проблеми мережі або серверної частини.

Після надсилання відповіді на завдання використовується функція mutate, яка інвалідує кеш і запускає повторне завантаження статистики та прогресу користувача. Завдяки цьому інтерфейс миттєво відображає нові значення XP, рівня та навчального прогресу.

Інтерфейс системи EduQuest реалізований із використанням адаптивного підходу, що забезпечує коректне відображення на мобільних пристроях, планшетах, ноутбуках та великих моніторах. Для цього використовується Tailwind CSS із системою responsive breakpoints.

Для мобільних пристроїв також реалізовано адаптивне меню навігації, збільшені інтерактивні області кнопок та оптимізовані відступи між елементами. Окрему увагу приділено продуктивності інтерфейсу: анімації та ефекти виконуються через CSS-трансформації, що мінімізує навантаження на браузер та забезпечує плавність роботи навіть на слабких пристроях.

Отже, у процесі реалізації клієнтської частини системи EduQuest було створено сучасний користувацький інтерфейс на основі React 19 та Next.js App Router. Реалізована архітектура забезпечує модульність, адаптивність та зручну навігацію для різних категорій користувачів. Використання компонентного

підходу, системи кешування SWR та адаптивного дизайну дозволило забезпечити високу швидкодію інтерфейсу, ефективну взаємодію із серверною частиною та комфортну роботу системи на різних типах пристроїв.

3.3 Реалізація елементів гейміфікації

Гейміфікація в системі EduQuest реалізована як комплекс взаємопов'язаних механізмів мотивації, що впливають на поведінку користувача в процесі навчання. Основна ідея полягає у перетворенні навчальної діяльності на послідовність досяжних цілей із постійним зворотним зв'язком, винагородами та візуалізацією прогресу. До ключових елементів гейміфікації належать система досвіду (XP), рівні, досягнення, серії активності, навчальні ігри та система візуального прогресу.

Система досвіду (XP) є базовим механізмом мотивації користувача, оскільки кожна навчальна дія (виконання завдання, участь у грі або проходження теми) супроводжується нарахуванням певної кількості очок досвіду. Накопичення XP безпосередньо впливає на рівень користувача, що формує відчуття поступового зростання компетентності.

Залежність між кількістю досвіду та рівнем має нелінійний характер і описується кореневою функцією. Це дозволяє зробити перші рівні доступними для швидкого проходження, тоді як кожен наступний рівень потребує дедалі більшого обсягу зусиль, що підтримує довгострокову мотивацію.

Рівень користувача обчислюється за формулою: $level = \left[\sqrt{\frac{XP}{100}}\right] + 1$.

Для реалізації цієї логіки використано функції обчислення рівня та визначення меж XP для кожного рівня (Додаток Г.1). Вони дозволяють системі динамічно визначати поточний рівень користувача, прогрес до наступного рівня та необхідну кількість досвіду для підвищення.

Для оцінки прогресії використовується також зворотна функція, яка дозволяє визначити, скільки XP необхідно для досягнення конкретного рівня. Це

забезпечує узгодженість між відображенням прогресу в інтерфейсі та внутрішньою логікою системи.

Загалом, така модель прогресії створює баланс між доступністю початкового етапу навчання та поступовим ускладненням вимог на вищих рівнях, що є важливим фактором довготривалої залученості користувача.

Система досягнень реалізує додатковий мотиваційний рівень, який стимулює користувача до виконання різноманітних навчальних дій поза межами базового проходження завдань. Досягнення виступають як індикатори конкретних поведінкових патернів: регулярність навчання, швидкість виконання завдань, кількість правильних відповідей та рівень засвоєння тем.

Логіка перевірки досягнень побудована на аналізі контексту дій користувача та його статистики. При кожній значущій події система виконує перевірку умов для всіх доступних досягнень, порівнюючи їх з поточними показниками користувача (Додаток Г.2).

Досягнення поділяються на кілька типів залежно від умов їх отримання: кількість виконаних завдань, серії правильних відповідей, кількість днів поспіль активності, швидкість виконання завдань та рівень опанування тем. Кожне досягнення має власний поріг активації, що дозволяє формувати багаторівневу систему мотивації.

Важливою складовою є механізм винагороди, який передбачає нарахування додаткового XP за кожне отримане досягнення. Це підсилює позитивне підкріплення та стимулює користувача до повторної взаємодії з системою. Завдяки такій структурі досягнення виконують не лише декоративну функцію, а й активно впливають на навчальну поведінку користувача.

Ігровий компонент системи реалізований через набір навчальних мініігор, які спрямовані на закріплення знань у форматі коротких інтерактивних сесій. Основна мета ігор полягає у підвищенні залученості користувача та тренуванні навичок у швидкісному режимі.

У системі реалізовано кілька типів ігор, кожна з яких орієнтована на певний когнітивний процес: швидкий рахунок, розв’язування математичних

прикладів, запам’ятовування пар та логічні головоломки. Кожна гра має власну механіку взаємодії, але всі вони об’єднані спільною системою нарахування балів і XP.

Наприклад, гра «Швидкий рахунок» побудована на генерації випадкових арифметичних виразів та виборі правильної відповіді з кількох варіантів (Додаток Г.3). Під час гри система фіксує кількість правильних відповідей, швидкість реакції та формує комбінований результат, який впливає на підсумковий XP.

Особливістю ігрового процесу є наявність таймера та системи серій правильних відповідей, що дозволяє формувати додаткові бонуси. Тобто користувач мотивується не лише до правильності, а й до швидкості виконання завдань.

Система серій активності спрямована на формування регулярної навчальної поведінки. Вона фіксує кількість послідовних днів, у які користувач виконував навчальні дії в системі.

Алгоритм перевіряє дату останньої активності користувача та порівнює її з поточною датою. У випадку щоденної активності серія збільшується, а при пропуску днів – скидається або починається заново (Додаток Г.4).

Цей механізм доповнюється бонусною системою, яка нараховує додатковий XP за тривалі серії. Зокрема, короткі серії заохочують початкову регулярність, тоді як довготривалі серії стимулюють сталість навчання протягом тижнів і місяців.

Окремим елементом гейміфікації є візуалізація прогресу користувача, яка реалізується через інтерфейсні компоненти клієнтської частини. Вона включає відображення рівня, прогресу до наступного рівня, статистики виконаних завдань, серій активності та отриманих досягнень.

Прогрес подається у вигляді індикаторів, які дозволяють користувачу швидко оцінити власний стан у системі та зрозуміти, скільки зусиль необхідно для досягнення наступної мети. Візуальні елементи доповнюються анімаціями та інтерактивними ефектами, що підсилюють емоційне сприйняття досягнень.

Тобто гейміфікаційна складова EduQuest формує цілісну систему мотивації, яка поєднує математично обґрунтовану модель прогресії, поведінкові механізми заохочення та візуальне підкріплення результатів навчання

Отже, реалізовані елементи гейміфікації забезпечують підвищення мотивації та залученості користувачів до навчального процесу. Система XP, рівнів, досягнень, серій активності та навчальних ігор формує регулярну навчальну поведінку та надає постійний зворотний зв'язок щодо прогресу. Завдяки цьому навчання набуває більш інтерактивного та динамічного характеру, що позитивно впливає на ефективність засвоєння матеріалу.

3.4 Тестування та оцінка ефективності системи

Для забезпечення якості та стабільності роботи системи EduQuest була застосована багаторівнева стратегія тестування, яка охоплює всі ключові компоненти програмного забезпечення – від окремих функцій до повного користувацького сценарію.

Загальна структура тестування побудована за принципом тестової піраміди, де найбільша частка припадає на модульні (unit) тести, менша – на інтеграційні, і найменша – на end-to-end тестування. Такий підхід дозволяє поєднати швидкість перевірки окремих модулів із перевіркою реальної роботи системи в умовах, наближених до користувацьких.

На нижньому рівні реалізовано 32 unit-тести, які перевіряють окремі функції бізнес-логіки, зокрема розрахунок XP, рівнів, перевірку валідації даних та роботу UI-компонентів. Середній рівень включає 8 інтеграційних тестів, що перевіряють взаємодію між модулями системи, наприклад, обмін даними між клієнтською частиною та API. Верхній рівень представлений 12 E2E тестами, реалізованими за допомогою Playwright, які моделюють повний сценарій роботи користувача – від входу в систему до виконання навчальних дій.

Такий розподіл забезпечує баланс між швидкістю виконання тестів і глибиною перевірки функціональності, що є важливим для підтримки стабільності системи під час розвитку. Unit тестування було реалізовано з використанням фреймворку Jest і спрямоване на перевірку окремих функцій та компонентів системи в ізольованому середовищі.

Основна увага приділялась перевірці ключових алгоритмів, зокрема розрахунку досвіду (XP), визначення рівнів користувача, системи бонусів за серії правильних відповідей, а також функцій валідації даних. Окремо тестувалися UI-компоненти, що забезпечують коректне відображення інтерфейсу.

У межах тестування було перевірено коректність нарахування XP залежно від складності завдання, правильності відповіді та наявності серії правильних результатів. Також перевірено, що при неправильній відповіді досвід не нараховується. Окремі тести підтверджують зростання XP при підвищенні складності завдань, що відповідає логіці гейміфікації системи. Крім того, виконано перевірку функцій валідації електронної пошти та пароля, де тестувались як валідні, так і невалідні вхідні дані. Це дозволило гарантувати базовий рівень захисту від некоректного введення даних користувачем.

Загалом unit-тестування забезпечує перевірку найбільш критичних функцій системи на рівні коду, що дозволяє швидко виявляти помилки ще на етапі розробки. Результати тестування наведені на рисунку 3.1.

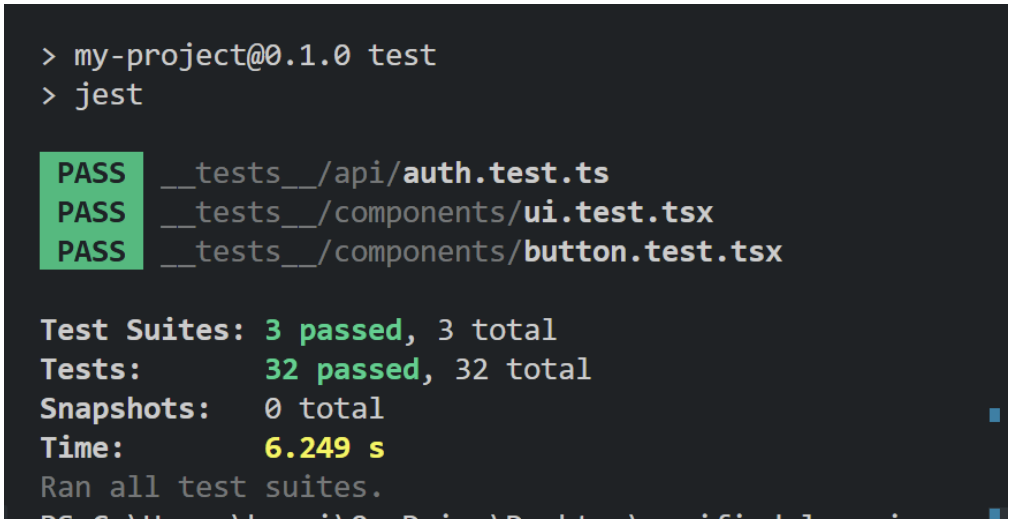


Рисунок 3.1 – Результати проходження кейсів unit-тестування

End-to-End тестування було реалізовано за допомогою Playwright і спрямоване на перевірку повного користувацького сценарію в системі EduQuest.

Ці тести моделюють поведінку реального користувача та перевіряють коректність роботи інтерфейсу, навігації та взаємодії з формами. Основні сценарії включають відкриття головної сторінки, процес входу в систему, реєстрацію нового користувача та перевірку валідації форм.

Особлива увага приділялась перевірці обробки помилок, наприклад, неправильних облікових даних або невідповідності паролів під час реєстрації. Також тестувалась наявність ключових елементів інтерфейсу, таких як поля введення, кнопки та повідомлення про помилки.

E2E тести виконуються у різних браузерах (Chromium, Firefox, WebKit), що дозволяє перевірити кросбраузерну сумісність системи. Результати виконання тестів підтвердили стабільну роботу основних користувацьких сценаріїв без критичних помилок (рис. 3.2).

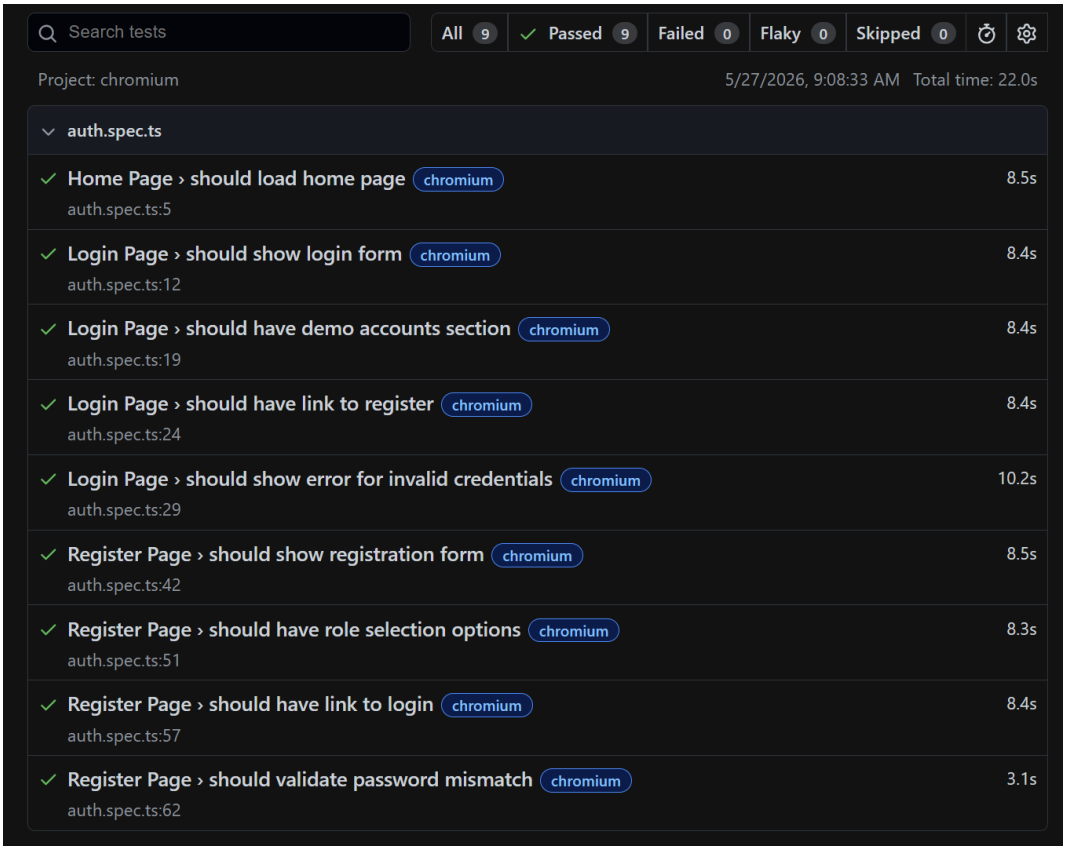


Рисунок 3.2 – Результати проходження e2e тестування

У результаті проведеного тестування було підтверджено стабільність роботи системи EduQuest, коректність реалізації основних функцій та відповідність вимогам до навчальної платформи. Система демонструє високу продуктивність, достатній рівень тестового покриття та стабільну роботу ключових користувацьких сценаріїв.

3.5 Інструкція користувача вебсистеми

Для коректної роботи вебсистеми EduQuest користувачеві необхідно забезпечити відповідність базовим технічним вимогам. Система працює у сучасних браузерах, зокрема Chrome (версія 90 і вище), Firefox (88+), Safari (14+) та Microsoft Edge (90+). Обов’язковою умовою є наявність стабільного підключення до Інтернету зі швидкістю не менше 1 Мбіт/с, що забезпечує коректне завантаження навчальних матеріалів та інтерактивних елементів.

Інтерфейс системи є адаптивним, тому підтримуються як мобільні пристрої з мінімальною роздільною здатністю 320×568 пікселів, так і десктопні пристрої до 2560×1440 пікселів. Для роботи системи обов’язково має бути увімкнено виконання JavaScript у браузері, оскільки саме на ньому базується логіка клієнтської частини. Також необхідно, щоб у браузері були дозволені cookies, які використовуються для автентифікації користувачів та збереження сесії.

Робота з системою починається з процесу реєстрації або входу в обліковий запис. Під час реєстрації користувач переходить на сторінку створення акаунта, де необхідно заповнити базову інформацію: ім’я, електронну пошту, пароль та підтвердження пароля. Ім’я повинно містити щонайменше два символи, email має бути унікальним, а пароль – не менше шести символів. Після введення даних користувач додатково обирає свою роль у системі – учень або вчитель. Після підтвердження форми створюється новий акаунт і користувач автоматично отримує доступ до відповідного інтерфейсу.

Процес входу є спрощеним і передбачає введення електронної пошти та пароля. Після успішної автентифікації система автоматично перенаправляє користувача до персонального кабінету відповідно до його ролі.

Для тестування системи передбачені демонстраційні акаунти для кожної ролі: учня, вчителя та адміністратора, що дозволяє швидко ознайомитися з функціоналом без необхідності створення нового облікового запису.

Після входу учень потрапляє до головного дашборду, який виконує функцію центральної панелі управління навчальним процесом. Тут відображається основна інформація про прогрес: кількість виконаних завдань, рівень досвіду (XP), серія активних днів, а також загальна статистика успішності.

Окремо відображається прогрес по предметах, що дозволяє учневі бачити рівень засвоєння кожної дисципліни та визначати теми, які потребують повторення. Також система формує рекомендації на основі попередньої активності, пропонуючи найбільш релевантні навчальні матеріали.

Навчальний процес відбувається через вибір предмета та конкретної теми. Після відкриття теми учневі пропонуються завдання, які необхідно виконати шляхом вибору або введення відповіді. Після кожної відповіді система надає миттєвий зворотний зв’язок: у разі правильної відповіді нараховується досвід та відображається позитивне повідомлення, у разі помилки – надається пояснення правильної відповіді.

Окремим елементом навчання є ігрові механіки, які дозволяють закріплювати знання у форматі мініігор, таких як швидке вирішення прикладів, пошук пар або логічні головоломки. Це підвищує залученість користувача та робить процес навчання більш інтерактивним.

Портал вчителя призначений для організації навчального процесу та управління учнями. Основним елементом є система класів, де вчитель може створити новий клас, вказати його назву та обрати предмет. Після створення класу система генерує унікальний код, який використовується учнями для приєднання.

Учні можуть приєднатися до класу, ввівши отриманий код у відповідному розділі, після чого вони автоматично з’являються у списку класу. Це дозволяє швидко формувати навчальні групи без складної адміністративної процедури.

Вчитель також має можливість призначати навчальні завдання. Для цього необхідно обрати клас, тему, встановити дедлайн та визначити вимоги до виконання. Після цього завдання стає доступним для учнів відповідного класу.

Окремий функціонал порталу дозволяє здійснювати моніторинг успішності учнів. Вчитель може переглядати загальну статистику класу, індивідуальні показники учнів, рівень активності та результати виконання завдань. Це забезпечує можливість оперативного контролю навчального процесу.

Система досягнень призначена для підвищення мотивації учнів шляхом винагородження за виконання певних навчальних цілей. Досягнення поділяються на кілька категорій, серед яких перші кроки, майстерність, серії правильних відповідей, активність та швидкість виконання завдань.

Кожне досягнення має певні умови отримання, наприклад виконання певної кількості завдань, досягнення високого рівня точності або підтримання активності протягом визначеного періоду. За кожне отримане досягнення користувач додатково отримує бонусний досвід.

У системі також передбачено перегляд як отриманих, так і ще заблокованих досягнень, що дозволяє користувачеві розуміти подальші цілі та мотивує до активнішого навчання.

Користувач має можливість керувати своїм профілем через розділ налаштувань. Доступні функції включають зміну імені, оновлення пароля та керування сповіщеннями.

Зміна пароля здійснюється через введення поточного та нового пароля, що забезпечує безпеку облікового запису. Також користувач може вмикати або вимикати сповіщення, які використовуються для нагадувань про навчальну активність.

Для підвищення зручності використання системи реалізовано набір гарячих клавіш. Клавіша Enter використовується для підтвердження відповіді, цифрові клавіші від 1 до 4 – для вибору варіантів відповідей у тестових завданнях, Esc – для закриття модальних вікон, а Tab – для переходу між полями введення. Це дозволяє значно пришвидшити взаємодію з інтерфейсом, особливо під час проходження навчальних завдань та ігор.

У системі передбачено розділ із відповідями на найпоширеніші запитання. Зокрема, користувач може відновити пароль через функцію «Забули пароль?», яка передбачає надсилання інструкцій на електронну пошту.

XP нараховується лише за правильні відповіді, тому у випадку відсутності нарахувань слід перевірити коректність виконання завдання. Вихід із класу здійснюється через відповідні налаштування, а зміна ролі користувача можлива лише через адміністратора системи.

Для запуску системи на локальному середовищі необхідно встановити Node.js (версія 18 або вище) та редактор коду VS Code. Далі потрібно клонувати репозиторій проєкту, встановити залежності за допомогою менеджера пакетів та створити конфігураційний файл середовища.

Після цього запуск системи здійснюється командою запуску dev-сервера, після чого вебсистема стає доступною у браузері за локальною адресою.

Для тестування передбачені окремі команди запуску unit-тестів, end-to-end тестів та генерації звіту про покриття коду. Це дозволяє розробникам перевіряти стабільність системи та контролювати якість коду на всіх етапах розробки.

У результаті інструкція користувача охоплює всі ключові сценарії роботи з вебсистемою EduQuest, включаючи реєстрацію, навчальний процес, управління класами, систему досягнень та налаштування профілю, що забезпечує повноцінне використання платформи різними категоріями користувачів.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено вебсистему навчання з елементами гейміфікації, яка спрямована на підвищення мотивації учнів, покращення залученості до навчального процесу та підвищення ефективності засвоєння навчального матеріалу. Реалізована система поєднує освітні функції з інтерактивними механізмами, що дозволяє забезпечити більш динамічний і практикоорієнтований підхід до навчання.

Під час виконання роботи було послідовно реалізовано всі поставлені завдання. На першому етапі проведено ґрунтовний аналіз сучасних вебсистем навчання, зокрема Moodle, Canvas LMS, Blackboard Learn, TalentLMS, Duolingo, Khan Academy та Classcraft. Дослідження цих платформ дозволило виявити їхні ключові архітектурні та функціональні особливості, підходи до організації навчального контенту, а також методи взаємодії з користувачами. Окрему увагу було приділено перевагам і недолікам кожного рішення, що дало змогу сформувати цілісне уявлення про сучасний стан розвитку електронного навчання та визначити напрями його вдосконалення.

Наступним етапом стало дослідження принципів гейміфікації в електронному навчанні. Було розглянуто основні мотиваційні механізми, які використовуються для підвищення активності користувачів, зокрема системи нарахування досвіду (XP), рівнів прогресу, досягнень, серій активності (streaks), а також інтерактивні навчальні ігрові елементи. Аналіз цих підходів дозволив обґрунтувати ефективність гейміфікації як інструменту підвищення внутрішньої мотивації учнів та формування регулярної навчальної поведінки.

На основі отриманих результатів були сформовані функціональні та нефункціональні вимоги до вебсистеми. Визначено ключові ролі користувачів (учень, вчитель, адміністратор), їхні права та основні сценарії взаємодії з

системою. Це стало базою для подальшого проєктування архітектури програмного забезпечення.

На етапі проєктування було розроблено загальну архітектуру вебсистеми, визначено структуру бази даних, а також описано взаємодію між основними компонентами системи. Обґрунтовано вибір сучасного технологічного стека, що забезпечує високу продуктивність, гнучкість, масштабованість та можливість подальшого розвитку системи без суттєвих змін у її архітектурі.

Практична реалізація роботи включала розробку серверної та клієнтської частин вебсистеми, а також впровадження ключових елементів гейміфікації. Зокрема, було реалізовано систему досвіду та рівнів, механізм досягнень, систему серій активності користувачів, а також навчальні мініігри, що забезпечують інтерактивну форму закріплення знань. Це дозволило створити більш залучене навчальне середовище, яке стимулює користувачів до регулярного навчання та саморозвитку.

Проведене багаторівневе тестування системи, що включало unit, інтеграційне та end-to-end тестування, підтвердило коректність роботи основних функціональних модулів і стабільність системи в цілому. Отримані показники покриття коду та результати тестування свідчать про достатній рівень якості програмного забезпечення. Додатково проведена оцінка продуктивності показала відповідність системи сучасним вимогам до вебзастосунків, зокрема щодо швидкодії та користувацького досвіду.

Водночас слід зазначити, що система має певні обмеження, зумовлені навчальним характером розробки. Зокрема, це стосується рівня масштабованості, який наразі розрахований на обмежену кількість користувачів, а також часткової реалізації функціональності окремих модулів.

Подальший розвиток вебсистеми може бути спрямований на розширення її функціональних можливостей та підвищення рівня персоналізації навчання. Зокрема, перспективним є впровадження адаптивних алгоритмів навчання на основі штучного інтелекту, які дозволятимуть автоматично підбирати складність завдань відповідно до рівня знань кожного учня.

Окремим напрямом розвитку є інтеграція складніших механік гейміфікації, таких як командні змагання, рейтингові сезони, соціальна взаємодія між користувачами та розширені системи винагород.

Також доцільним є розширення предметної області системи шляхом додавання нових навчальних дисциплін та мультимедійного контенту, включаючи інтерактивні симуляції та відеоматеріали. У межах подальшого розвитку також передбачено реалізацію додаткових навчальних модулів у форматі пісочниць, зокрема «Науковий калькулятор» (розширений інструмент для виконання математичних обчислень з історією операцій) та «Математичні ігри» (інтерактивний модуль для навчання через логічні та арифметичні завдання).

З технічного погляду перспективним є оптимізація продуктивності системи, впровадження мобільного застосунку, а також масштабування серверної інфраструктури для підтримки більшої кількості користувачів.

Отже, розроблена система має значний потенціал для подальшого вдосконалення та розширення у сфері цифрової освіти.

ПЕРЕЛІК ПОСИЛАНЬ

1. Герасименко І. В. Методика використання технологій дистанційного навчання в підготовці бакалаврів комп'ютерних наук : дис. … канд. пед. наук : 13.00.10. Черкаси, 2015. 302 с.

2. Глибока С. Електронне навчання (E-learning): сутність, походження та етапи розвитку, особливості застосування у ЗВО. *Актуальні питання гуманітарних наук*. 2024. Вип. 76. Т. 1. С. 242–248.

3. Bakia M. Internet-based education. In: Elsevier eBooks. 2010. P. 102–108. DOI : https://doi.org/10.1016/b978-0-08-044894-7.00755-7.

4. Chrysafiadi K., Virvou M. Student modeling approaches : A literature review for the last decade. *Expert Systems with Applications*. 2013. Vol. 40. No. 11. P. 4715–4729. DOI : https://doi.org/10.1016/j.eswa.2013.02.007.

5. Sailer M., Homner L. The gamification of learning : a meta-analysis. *Educational Psychology Review*. 2019. Vol. 32. No. 1. P. 77–112. DOI : https://doi.org/10.1007/s10648-019-09498-w.

6. Галицька-Дідух Т. В., Гречановська О. В., Батарейна І. О., Гурська В. А., Бабік І. В. Гейміфікація як засіб індивідуалізації навчання XXI століття. *Futurity Research Publishing*, 2025. 204 с.

7. Chugai O. The power of games: what, why and how to employ it in education. *European Science*. 2023. Vol. 2. No. sge18-02. P. 52–82. DOI : https://doi.org/10.30890/2709-2313.2023-18-02-024

8. Reiter M. Gamification in education: effectiveness and impact on student motivation. *Acta Educationis Generalis*. 2025. Vol. 15. No. s1. P. 38–51. DOI : https://doi.org/10.2478/atd-2025-0013.

9. Araya R., Arias Ortiz E., Bottan N. L., Cristia J. P. Does gamification in education work? : experimental evidence from Chile. 2019. DOI : https://doi.org/10.18235/0001777.

10. Baytech Consulting. The Learning Management System 2025. URL : https://www.baytechconsulting.com/blog/the-learning-management-system-2025 (дата звернення : 12.03.2026).

11. TalentLMS. Top 7 Moodle alternatives for 2025. URL : https://www.talentlms.com/blog/moodle-alternatives/ (дата звернення : 12.03.2026).

12. Niazi S., Iqbal R. An evaluation framework and comparative analysis of the widely used learning management systems. *PLOS ONE*. 2024. URL : https://pmc.ncbi.nlm.nih.gov/articles/PMC11658596/ (дата звернення : 12.03.2026).

13. Selleo. What are the challenges of implementing gamification in LMS platforms? 2024. URL : https://selleo.com/blog/what-are-the-challenges-of-implementing-gamification (дата звернення : 12.03.2026).

14. SoftwareFinder. TalentLMS vs. *Canvas : in-depth comparison*. 2025. URL : https://softwarefinder.com/resources/talentlms-vs-canvas (дата звернення : 12.03.2026).

15. D2L. Moodle alternatives 2026 : top LMS competitor overview. URL : https://www.d2l.com/blog/moodle-alternatives/ (дата звернення : 12.03.2026).

16. StriveCloud. *Duolingo gamification explained*. 2026. URL : https://www.strivecloud.io/blog/gamification-examples-boost-user-retention-duolingo (дата звернення : 12.03.2026).

17. Zandvakili R., et al. Exploring the potential of gamified reading: the effects of Duolingo on L2 reading. PMC. 2025. URL : https://pmc.ncbi.nlm.nih.gov/articles/PMC12781743/ (дата звернення : 12.03.2026).

18. Daniele F. Decoding Duolingo : A Case Study on the Impact of Gamification on User Experience. *Medium*. 2024. URL : https://medium.com/@flordaniele/duolingo-case-study-research-on-gamification-90b5bac3ada0 (дата звернення : 12.03.2026).

19. Mambo.io. What is eLearning Gamification : A Comprehensive Guide. 2025. URL : https://mambo.io/blog/elearning-gamification (дата звернення : 12.03.2026).

20. Sanchez E., Young S., Jouneau-Sion C. Classcraft : from gamification to ludicization of classroom management. ResearchGate. 2017. URL : https://www.researchgate.net/publication/301831246 (дата звернення : 12.03.2026).

21. Angular. What is Angular? URL : https://angular.dev/overview (дата звернення : 12.03.2026).

22. React. Build a React app from Scratc. URL : https://react.dev/learn/build-a-react-app-from-scratch (дата звернення : 12.03.2026).

23. Next.js. Documentation. URL : https://nextjs.org/docs (дата звернення : 12.03.2026).

24. Next.js. Routing : API routes. URL : https://nextjs.org/docs/pages/building-your-application/routing/api-routes (дата звернення : 12.03.2026).

25. PostgreSQL. Documentation. URL : https://www.postgresql.org/docs/ (дата звернення : 12.03.2026).

26. MySQL. Documentation. URL : https://dev.mysql.com/doc/ (дата звернення : 12.03.2026).

27. SQLite. Documentation. URL : https://sqlite.org/docs.html (дата звернення : 12.03.2026).

28. Material UI. React components that implement Material Design. URL : https://mui.com/material-ui/ (дата звернення : 12.03.2026).

29. Ant Design. The world’s second most popular React UI framework. URL : https://ant.design/ (дата звернення : 12.03.2026).

30. shadcn/ui. The Foundation for your Design System. URL : https://ui.shadcn.com/ (дата звернення : 12.03.2026).

31. Tailwind CSS. Rapidly build modern websites without ever leaving your HTML. URL : https://tailwindcss.com/ (дата звернення : 12.03.2026).

32. Харовюк В. Аутентифікація та управління станом : коли обирати сесії, а коли – JWT і Local Storage. *WebCraft*. 2026. 20 берез. URL : https://webscraft.org/blog/autentifikatsiya-ta-upravlinnya-stanom-koli-obirati-sesiyi-a-koli-jwt-i-local-storage (дата звернення : 12.03.2026).

33. OAuth 2.0. Офіційна документація. URL : https://oauth.net/2/ (дата звернення : 12.03.2026).

34. SWR. Офіційна документація. URL : https://swr.vercel.app/ (дата звернення : 12.03.2026).

35. TypeScript. JavaScript with syntax for types. URL : https://www.typescriptlang.org/ (дата звернення : 12.03.2026).

36. Athimoolam P. Comparing Jest, React Testing Library, and Playwright : Testing Approaches for React Applications. *DEV Community*. 2024. 28 жовт. URL : https://dev.to/padmajothi_athimoolam_23d/comparing-jest-react-testing-library-and-playwright-testing-approaches-for-react-applications-15ic (дата звернення : 12.03.2026).

37. Vercel. Official website. URL : https://vercel.com/ (дата звернення : 12.03.2026).

38. Visual Studio Code. Official website. URL : https://code.visualstudio.com/ (дата звернення : 12.03.2026).

ДОДАТОК А

Лістинг – реалізація серверної частини системи

Лістинг А.1 – Ініціалізація бази даних (lib/db.ts)

```
import Database from 'better-sqlite3'
import path from 'path'

const dbPath = path.join(process.cwd(), 'eduquest.db')
const db = new Database(dbPath)

// Включення WAL режиму для кращої продуктивності
db.pragma('journal_mode = WAL')

export function initializeDatabase() {
  db.exec(`
    -- Користувачі
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      email TEXT UNIQUE NOT NULL,
      password_hash TEXT NOT NULL,
      name TEXT NOT NULL,
      role TEXT CHECK(role IN ('student', 'teacher', 'admin')) DEFAULT
'student',
      xp INTEGER DEFAULT 0,
      level INTEGER DEFAULT 1,
      streak_days INTEGER DEFAULT 0,
```

```
    last_activity TEXT,
    created_at TEXT DEFAULT (datetime('now'))
  );

  -- Індекси для швидкого пошуку
  CREATE INDEX IF NOT EXISTS idx_tasks_topic ON tasks(topic_id);
  CREATE INDEX IF NOT EXISTS idx_tasks_difficulty ON tasks(difficulty);
  CREATE INDEX IF NOT EXISTS idx_progress_user ON user_progress(user_id);
  CREATE INDEX IF NOT EXISTS idx_history_user ON answer_history(user_id);
  `)
}

export default db
```

Лістинг А.2 – Модуль автентифікації (lib/auth.ts)

```
import { hash, compare } from 'bcryptjs'
import { SignJWT, jwtVerify } from 'jose'
import { cookies } from 'next/headers'

const JWT_SECRET = new TextEncoder().encode(
  process.env.JWT_SECRET || 'eduquest-secret-key'
)
const COOKIE_NAME = 'eduquest-session'

// Хешування паролів з bcrypt (12 раундів)
export async function hashPassword(password: string): Promise<string> {
  return hash(password, 12)
```

```
}

export async function verifyPassword(password: string, hash: string): Promise<boolean> {
  return compare(password, hash)
}

// Генерація JWT токену з терміном дії 7 днів
export async function createToken(userId: number, role: UserRole): Promise<string> {
  return new SignJWT({ userId, role })
    .setProtectedHeader({ alg: 'HS256' })
    .setExpirationTime('7d')
    .sign(JWT_SECRET)
}

// Створення сесії з HttpOnly cookie
export async function createSession(userId: number, role: UserRole): Promise<void> {
  const token = await createToken(userId, role)
  const cookieStore = await cookies()

  cookieStore.set(COOKIE_NAME, token, {
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    sameSite: 'lax',
    maxAge: 60 * 60 * 24 * 7, // 7 днів
    path: '/',
  })
}
```

Лістинг А.3 – Адаптивний алгоритм (lib/adaptive-algorithm.ts)

```
// Параметри алгоритму
const ADAPTATION_RATE = 0.3    // Швидкість адаптації
const WINDOW_SIZE = 10         // Вікно аналізу відповідей
const TARGET_SUCCESS_RATE = 0.7 // Цільова успішність 70%
const MIN_DIFFICULTY = 1
const MAX_DIFFICULTY = 10

export function calculateTargetDifficulty(state: AdaptiveState): number {
  if (state.recentAnswers.length === 0) {
    return state.currentDifficulty
  }

  // Розрахунок успішності
  const successRate = state.recentAnswers.filter(Boolean).length / state.recentAnswers.length

  // Базове коригування складності
  let adjustment = (successRate - TARGET_SUCCESS_RATE) * ADAPTATION_RATE * 10

  // Бонус за серію правильних відповідей
  if (state.streak >= 5) {
    adjustment += 0.5
  }

  // Бонус за швидкі правильні відповіді
  if (state.averageTime < 15 && successRate > 0.8) {
    adjustment += 0.3
```

```
  }

  const newDifficulty = state.currentDifficulty + adjustment

  return Math.max(MIN_DIFFICULTY, Math.min(MAX_DIFFICULTY,
Math.round(newDifficulty)))
}
```

Лістинг А.4 – Приклад API Route для відправки відповіді

```
// app/api/tasks/submit/route.ts
import { NextRequest, NextResponse } from 'next/server'
import { z } from 'zod'
import db from '@/lib/db'
import { getSession, updateUserXP, updateUserStreak } from '@/lib/auth'
import { recordAnswer } from '@/lib/adaptive-algorithm'
import { checkAndAwardAchievements } from '@/lib/achievements'

const submitSchema = z.object({
  taskId: z.number(),
  answer: z.string(),
  timeSpent: z.number().min(0)
})

export async function POST(request: NextRequest) {
  try {
    // Перевірка автентифікації
    const session = await getSession()
    if (!session) {
```

```
  return NextResponse.json(
    { success: false, error: 'Не авторизовано' },
    { status: 401 }
  )
}

// Валідація вхідних даних
const body = await request.json()
const data = submitSchema.parse(body)

// Отримання завдання
const task = db.prepare('SELECT * FROM tasks WHERE id = ?')
  .get(data.taskId) as Task

// Перевірка відповіді
const isCorrect = data.answer.trim().toLowerCase() ===
                  task.correct_answer.trim().toLowerCase()

// Запис відповіді та оновлення прогресу
recordAnswer(session.userId, data.taskId, isCorrect, data.timeSpent)
updateUserStreak(session.userId)

// Нарахування XP при правильній відповіді
let xpEarned = 0
if (isCorrect) {
  xpEarned = task.xp_reward
  updateUserXP(session.userId, xpEarned)
}

// Перевірка досягнень
```

```
        const newAchievements = await
checkAndAwardAchievements(session.userId, {
          isCorrect,
          timeSpent: data.timeSpent
        })

        return NextResponse.json({
          success: true,
          data: {
            is_correct: isCorrect,
            correct_answer: task.correct_answer,
            xp_earned: xpEarned,
            explanation: task.explanation,
            new_achievements: newAchievements
          }
        })
      } catch (error) {
        if (error instanceof z.ZodError) {
          return NextResponse.json(
            { success: false, error: error.errors[0].message },
            { status: 400 }
          )
        }
        return NextResponse.json(
          { success: false, error: 'Помилка сервера' },
          { status: 500 }
        )
      }
    }
```

ДОДАТОК Б

Лістинг – реалізація клієнтської частини та користувацького інтерфейсу

Лістинг Б.1 – Компонент дашборду учня

```
// app/student/dashboard/page.tsx
"use client"

import { useEffect, useState } from "react"
import Link from "next/link"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Progress } from "@/components/ui/progress"
import { Button } from "@/components/ui/button"
import { Skeleton } from "@/components/ui/skeleton"
import { BookOpen, Target, Trophy, Zap, TrendingUp } from "lucide-react"

export default function StudentDashboard() {
  const [data, setData] = useState<DashboardData | null>(null)
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    fetch("/api/progress")
      .then(res => res.json())
      .then(result => {
        if (result.success) setData(result.data)
      })
```

```
      .finally(() => setLoading(false))
  }, [])

  if (loading) {
   return (
    <div className="space-y-6">
     <Skeleton className="h-8 w-48" />
     <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
      {[1, 2, 3, 4].map(i => <Skeleton key={i} className="h-32" />)}
     </div>
    </div>
   )
  }

  return (
   <div className="space-y-6">
    {/* Stats Grid */}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
     <Card>
      <CardContent className="pt-6">
       <div className="flex items-center gap-4">
        <div className="w-12 h-12 rounded-xl bg-primary/20 flex items-
center justify-center">
         <Target className="w-6 h-6 text-primary" />
        </div>
        <div>
         <p className="text-2xl font-bold">{data.stats.totalTasks}</p>
         <p          className="text-sm          text-muted-foreground">Завдань
виконано</p>
        </div>
```

```
      </div>
    </CardContent>
  </Card>

  {/* Інші картки статистики */}
</div>

{/* Прогрес по предметах */}
<Card>
  <CardHeader>
    <CardTitle className="flex items-center gap-2">
      <BookOpen className="w-5 h-5" />
      Прогрес по предметах
    </CardTitle>
  </CardHeader>
  <CardContent className="space-y-4">
    {data.subjects_progress.map(subject => (
      <Link key={subject.id} href={`/student/subjects/${subject.slug}`}>
        <div className="p-4 rounded-xl border hover:border-primary/30">
          <div className="flex items-center gap-3 mb-3">
            <div className="w-10 h-10 rounded-lg"
                 style={{ backgroundColor: `${subject.color}20` }}>
              <BookOpen style={{ color: subject.color }} />
            </div>
            <div className="flex-1">
              <p className="font-medium">{subject.name}</p>
              <p className="text-xs text-muted-foreground">
                {subject.topics_practiced} з {subject.total_topics} тем
              </p>
            </div>
```

```
                <span className="text-sm font-medium">
                  {Math.round(subject.avg_mastery)}%
                </span>
              </div>
              <Progress value={subject.avg_mastery} className="h-2" />
            </div>
          </Link>
        ))}
      </CardContent>
    </Card>
  </div>
 )
}
```

Лістинг Б.2 – Приклад використання SWR:

```
import useSWR, { mutate } from 'swr'

const fetcher = (url: string) => fetch(url).then(res => res.json())

function useUserStats() {
 const { data, error, isLoading } = useSWR('/api/user/stats', fetcher, {
  revalidateOnFocus: true,
  refreshInterval: 30000  // Оновлення кожні 30 секунд
 })

 return {
  stats: data?.data,
  isLoading,
```

```
    isError: error
  }
}

// Оновлення кешу після дії
async function submitAnswer(taskId: number, answer: string) {
  const response = await fetch('/api/tasks/submit', {
    method: 'POST',
    body: JSON.stringify({ taskId, answer })
  })

  // Інвалідація кешу для оновлення статистики
  mutate('/api/user/stats')
  mutate('/api/progress')

  return response.json()
}
```

ДОДАТОК В

Інтерфейс розробленої вебсистеми

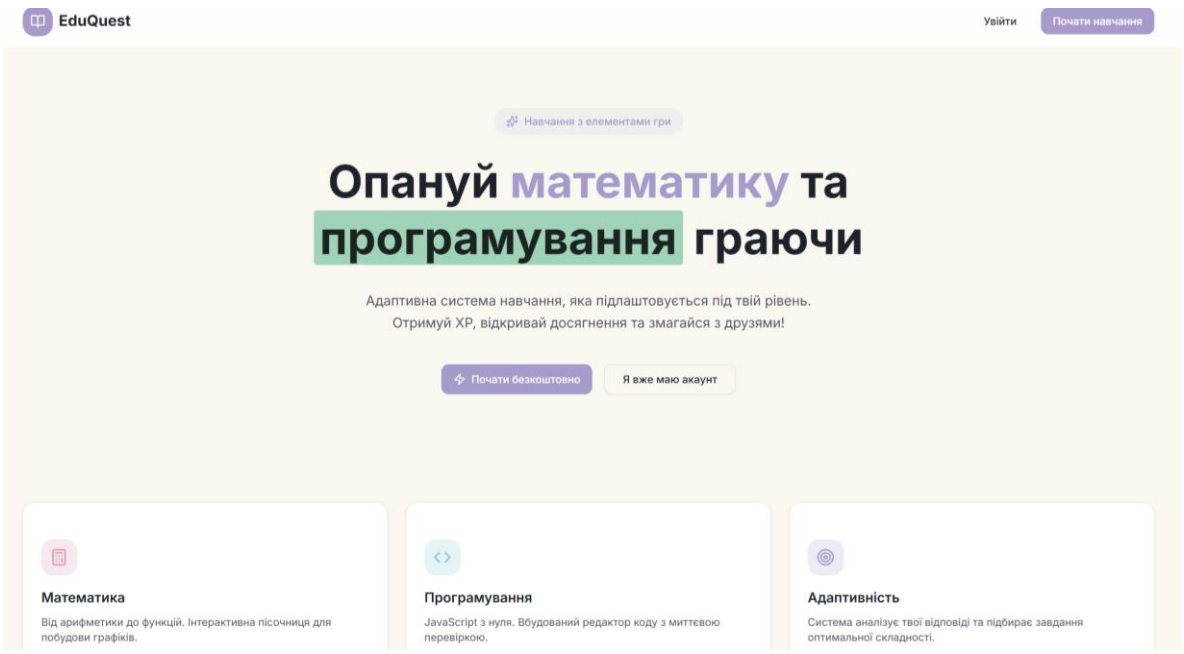


Рисунок В.1 – Головна сторінка

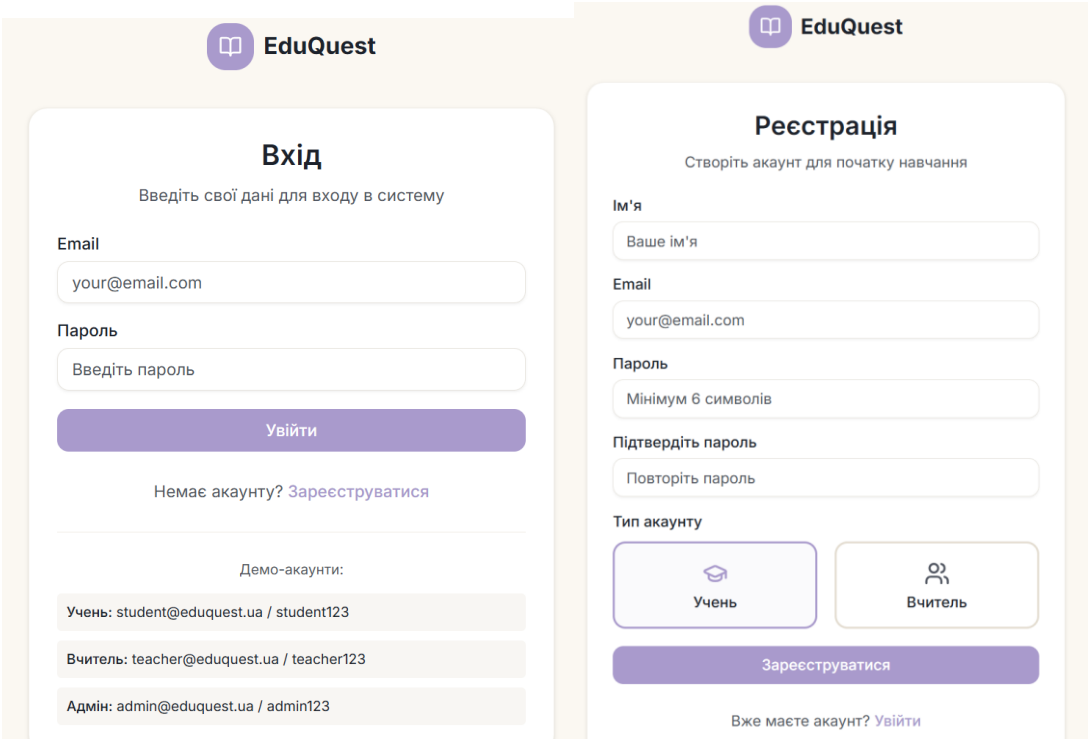


Рисунок В.2 – Сторінка входу та реєстрації

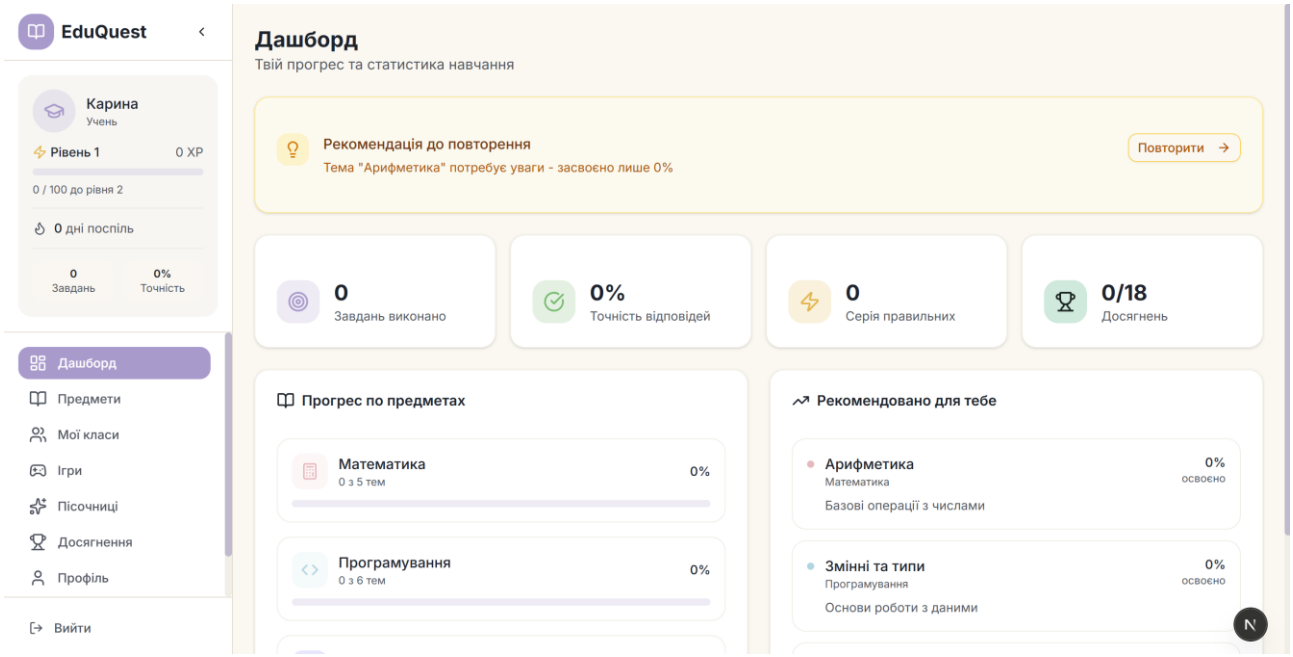


Рисунок В.3 – Дашборд учня

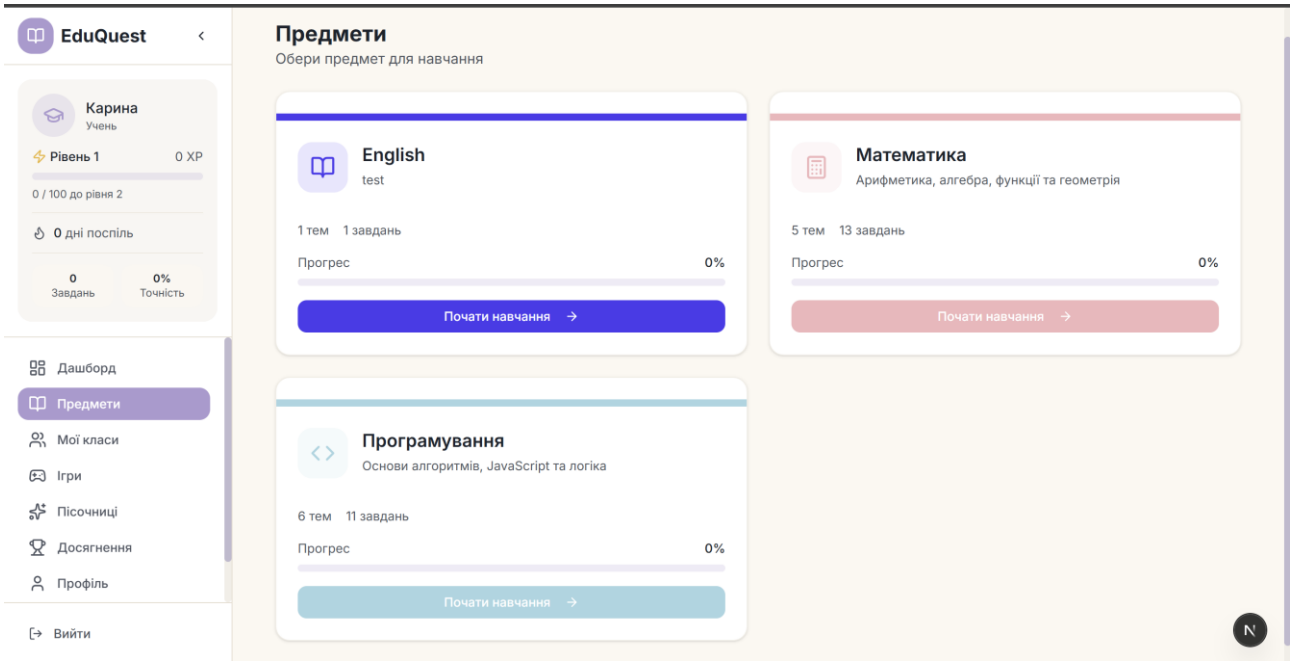


Рисунок В.4 – Сторінка предметів в учня

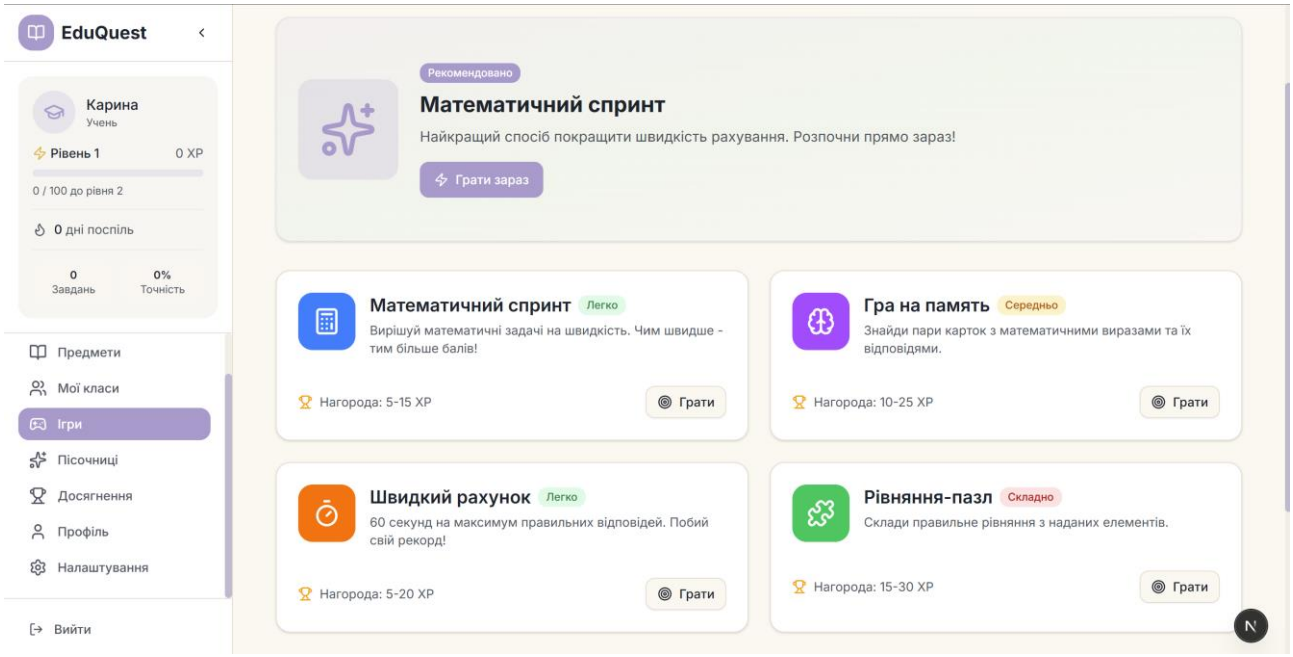


Рисунок В.5 – Сторінка ігор в учня

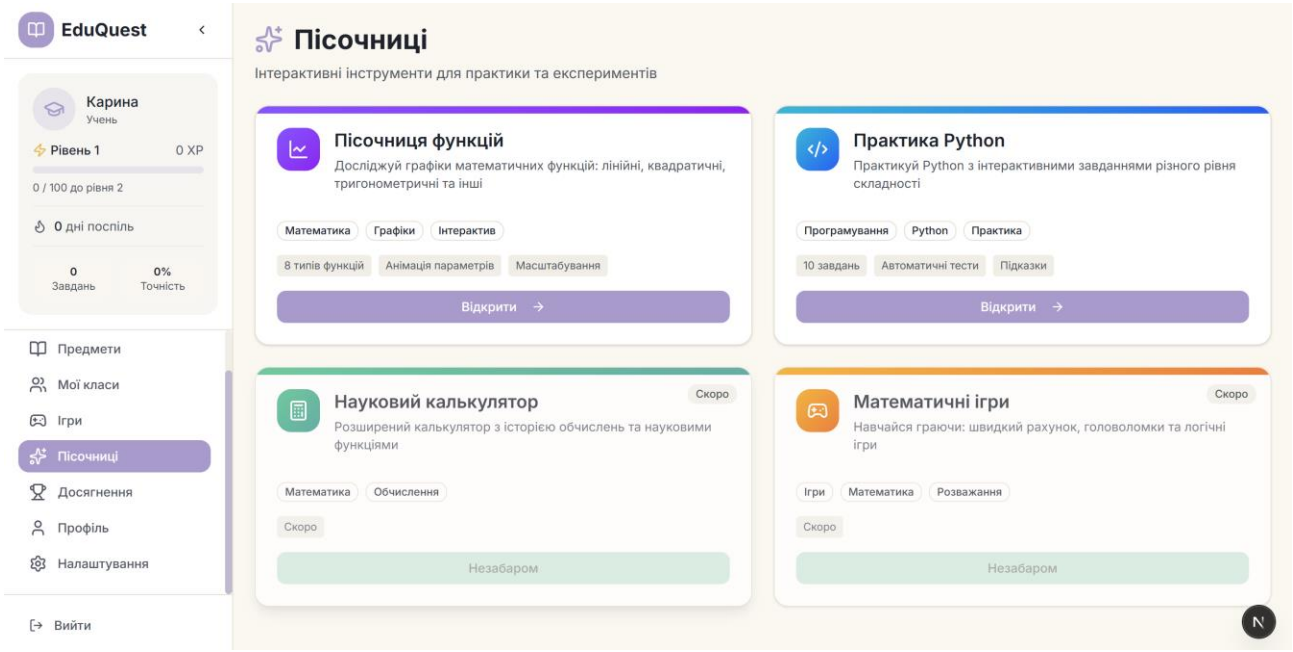


Рисунок В.6 – Сторінка пісочниця в учня

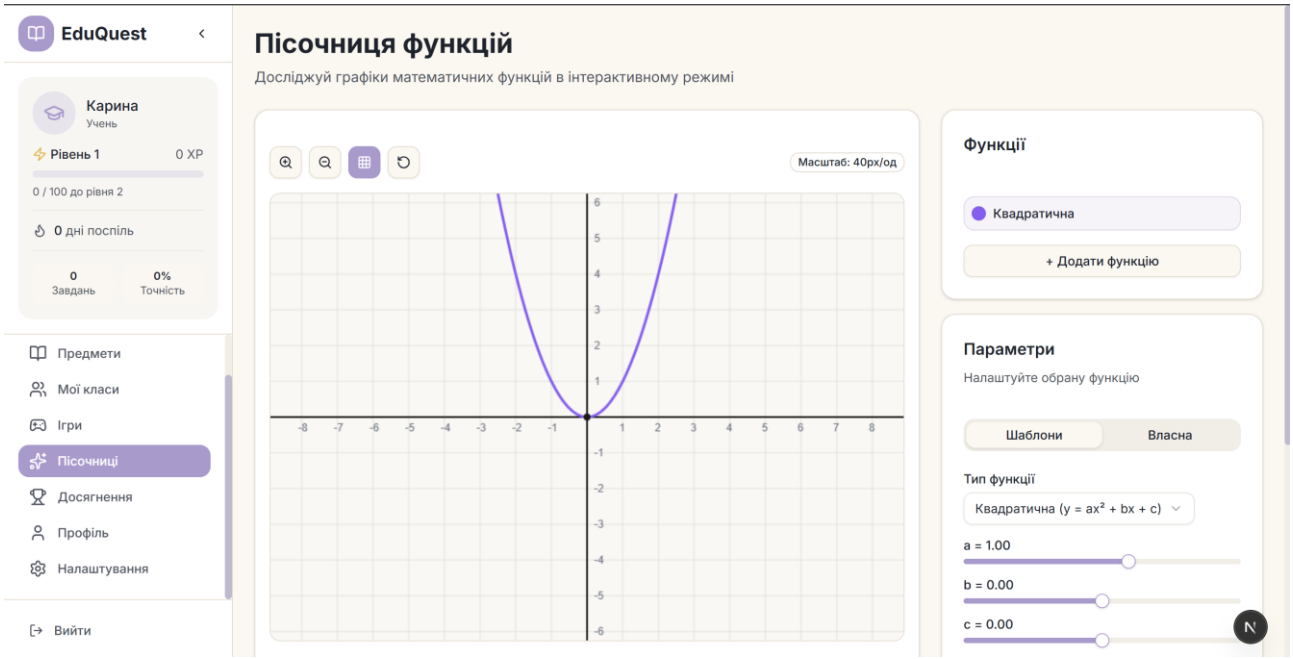


Рисунок В.7 – Сторінка пісочниці функцій

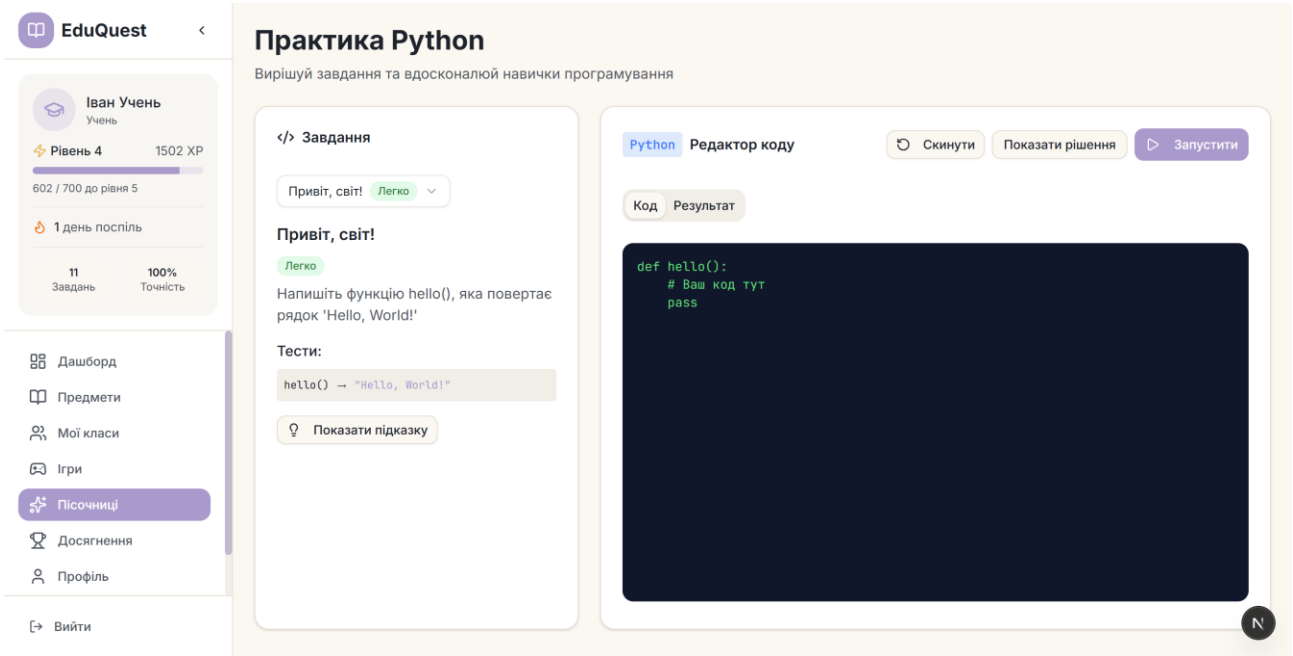


Рисунок В.8 – Сторінка пісочниці з програмування

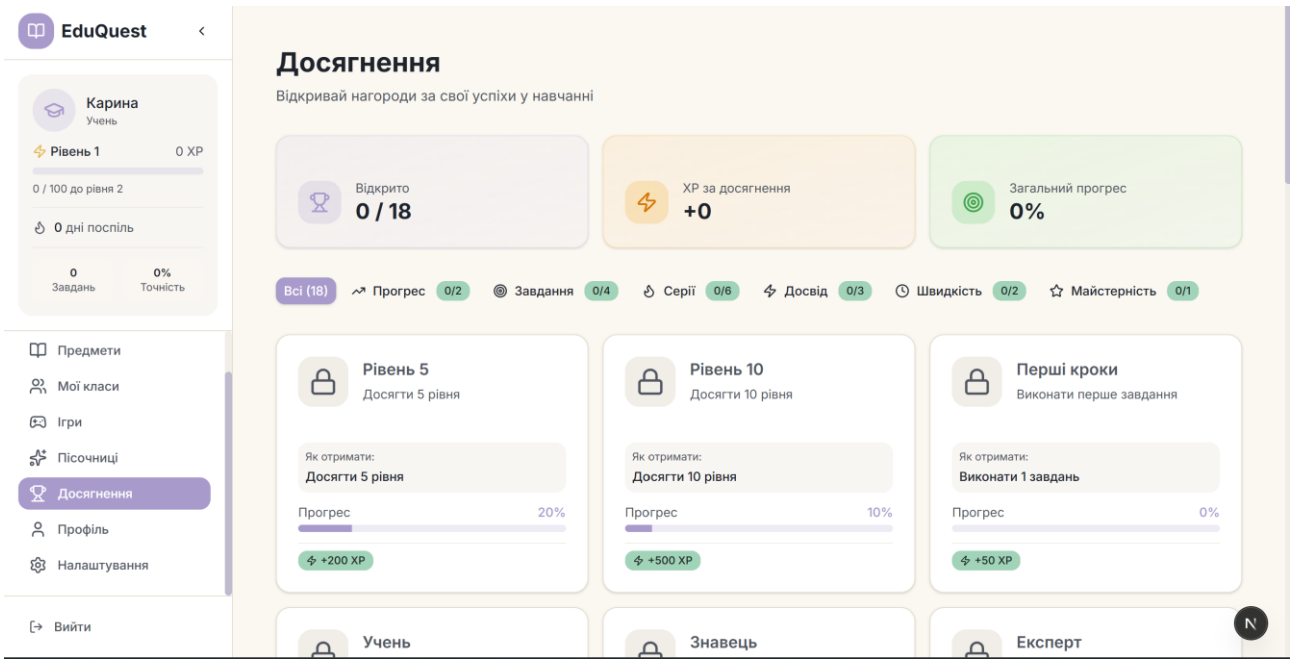


Рисунок В.9 – Сторінка досягнень учня

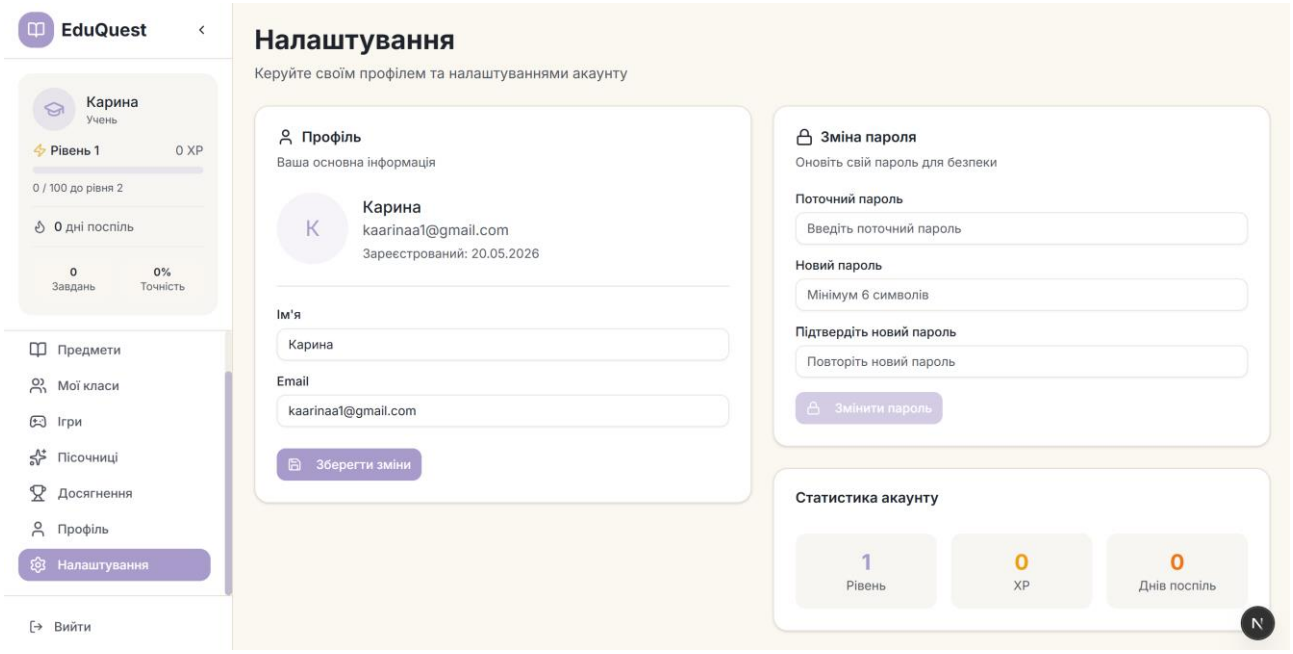


Рисунок В.10 – Сторінка налаштувань

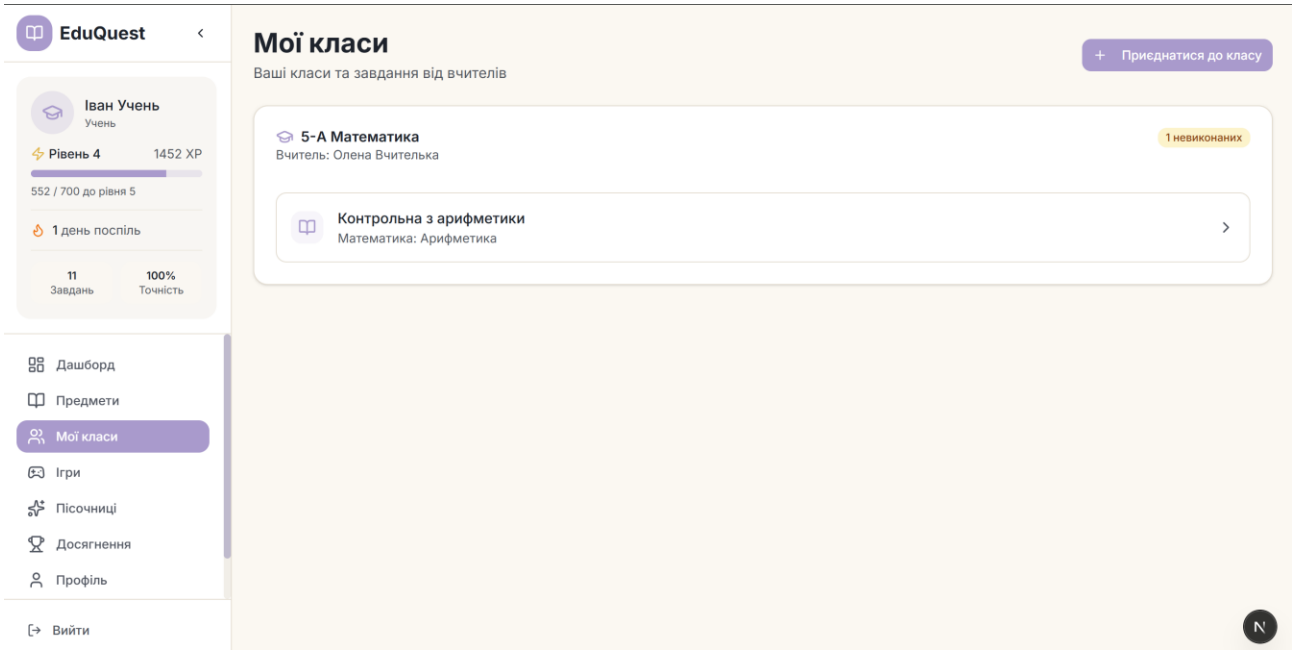


Рисунок В.11 – Сторінка класів в учня

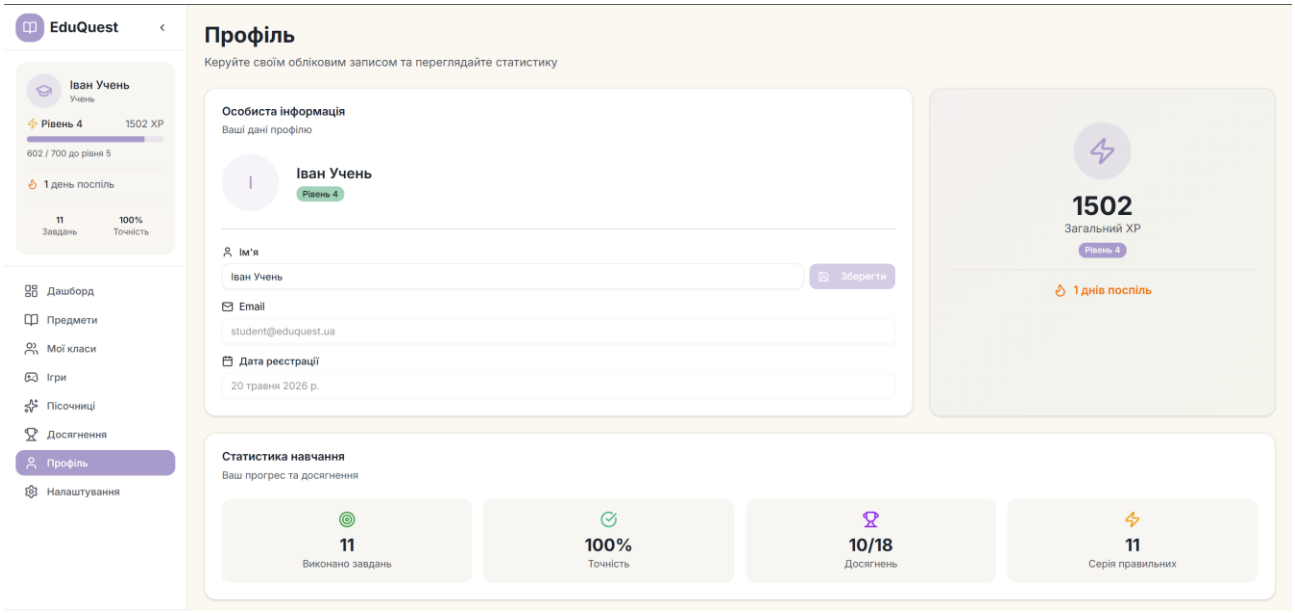


Рисунок В.12 – Сторінка профілю в учня

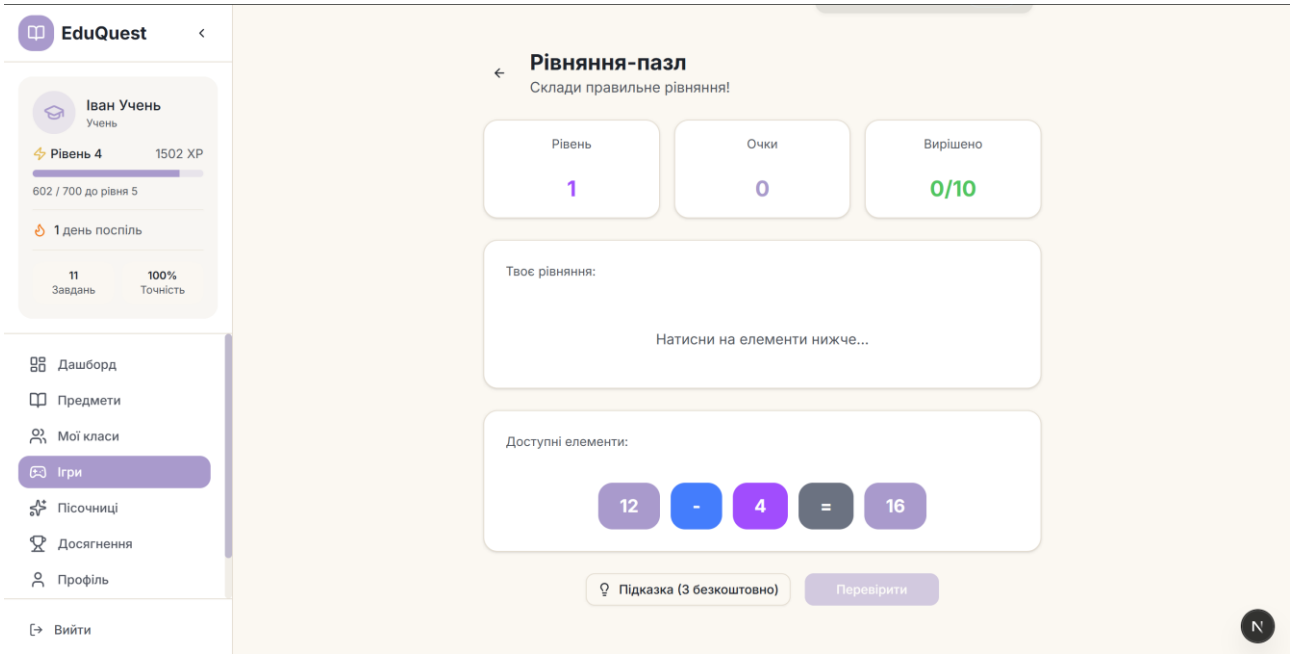


Рисунок В.13 – Гра «Рівняння-пазл»

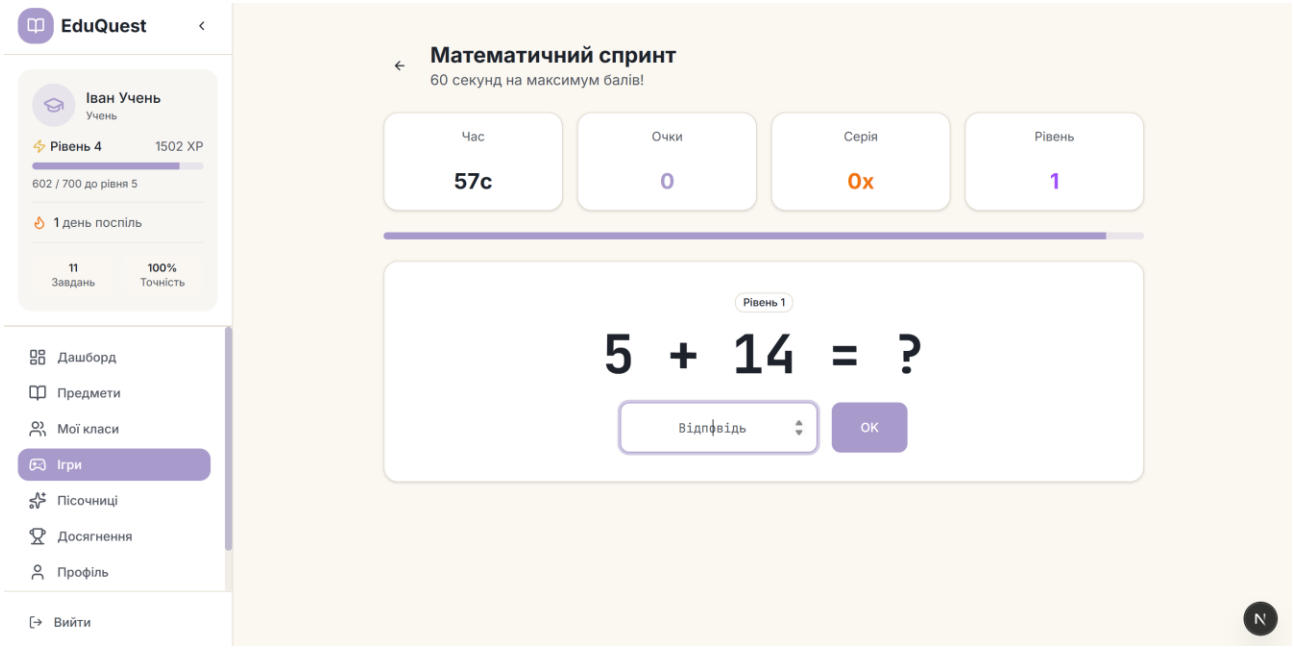


Рисунок В.14 – Гра «Математичний спринт»

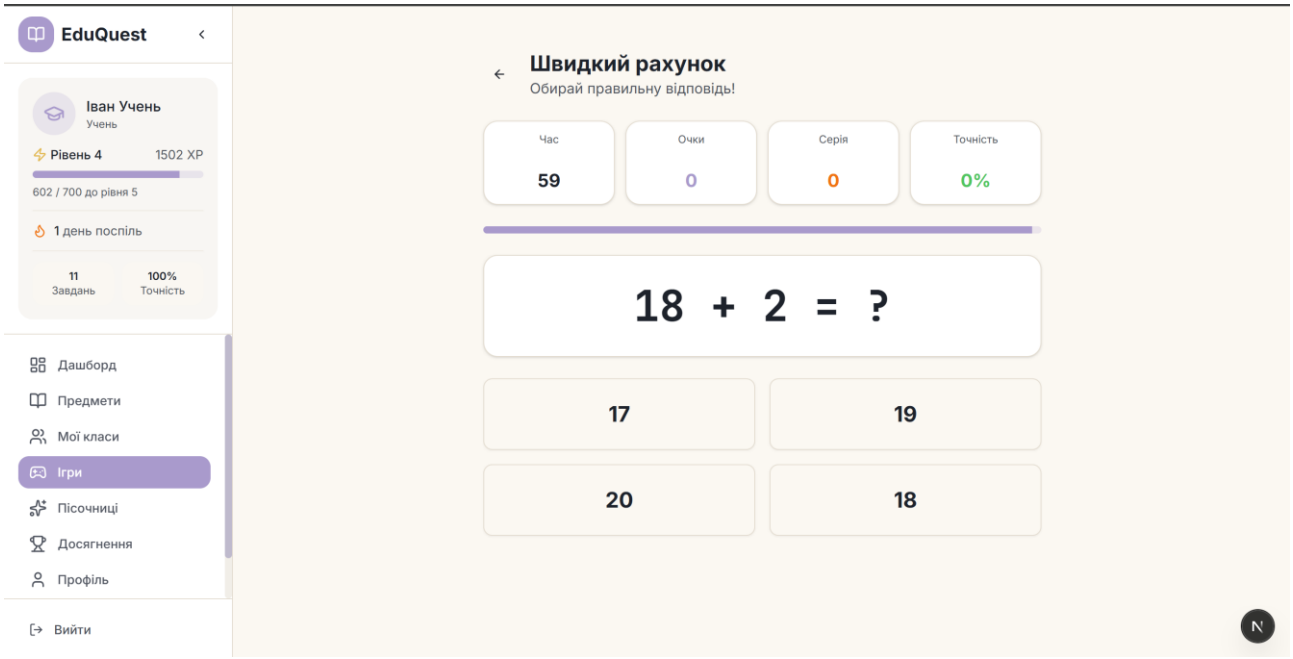


Рисунок В.15 – Гра «Швидкий рахунок»

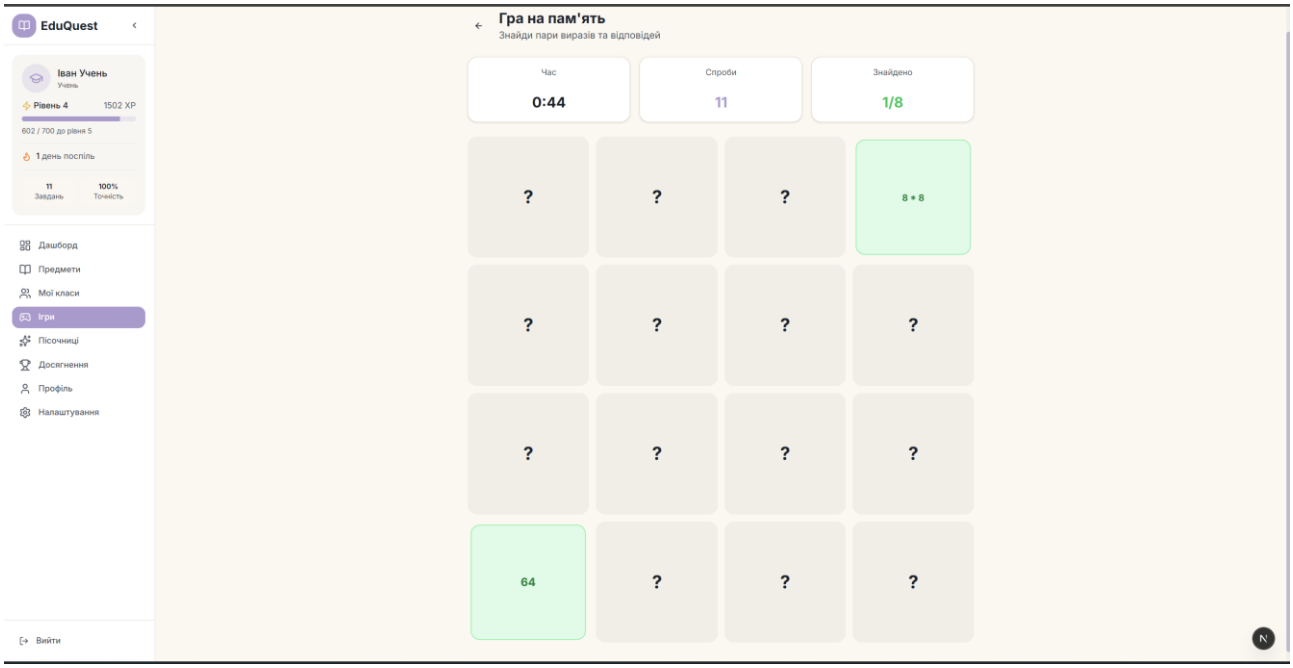


Рисунок В.16 – Гра «Гра на пам’ять»

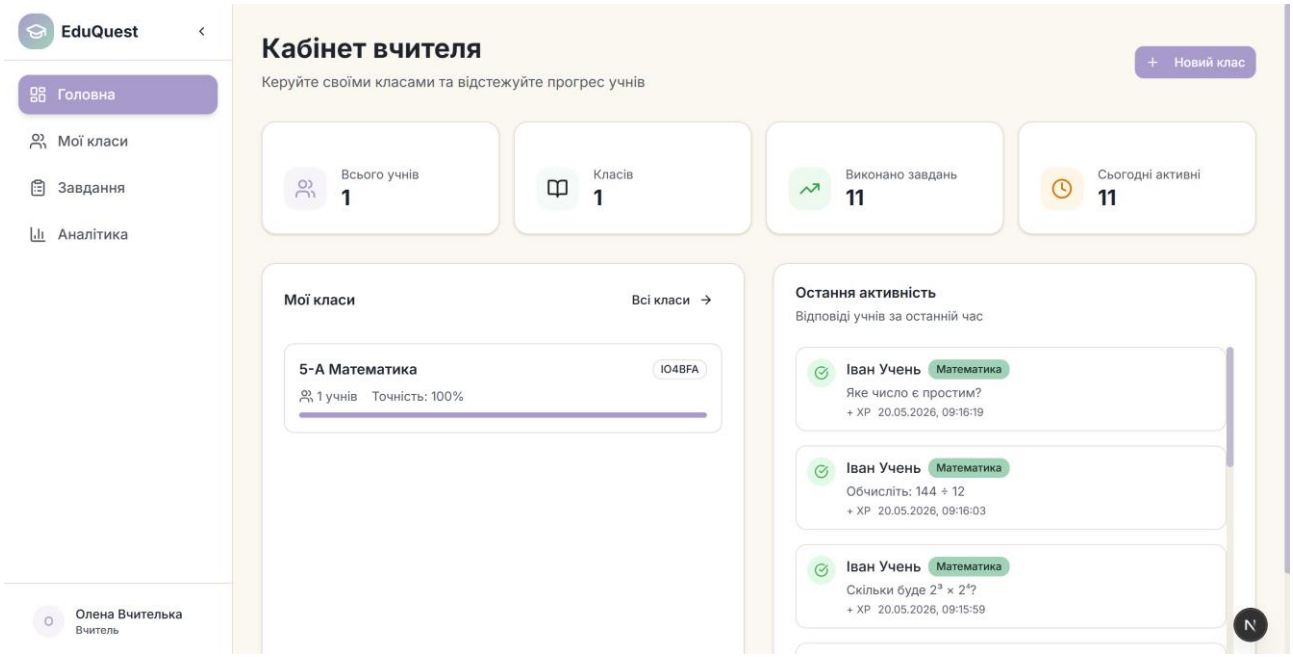


Рисунок В.17 – Кабінет вчителя

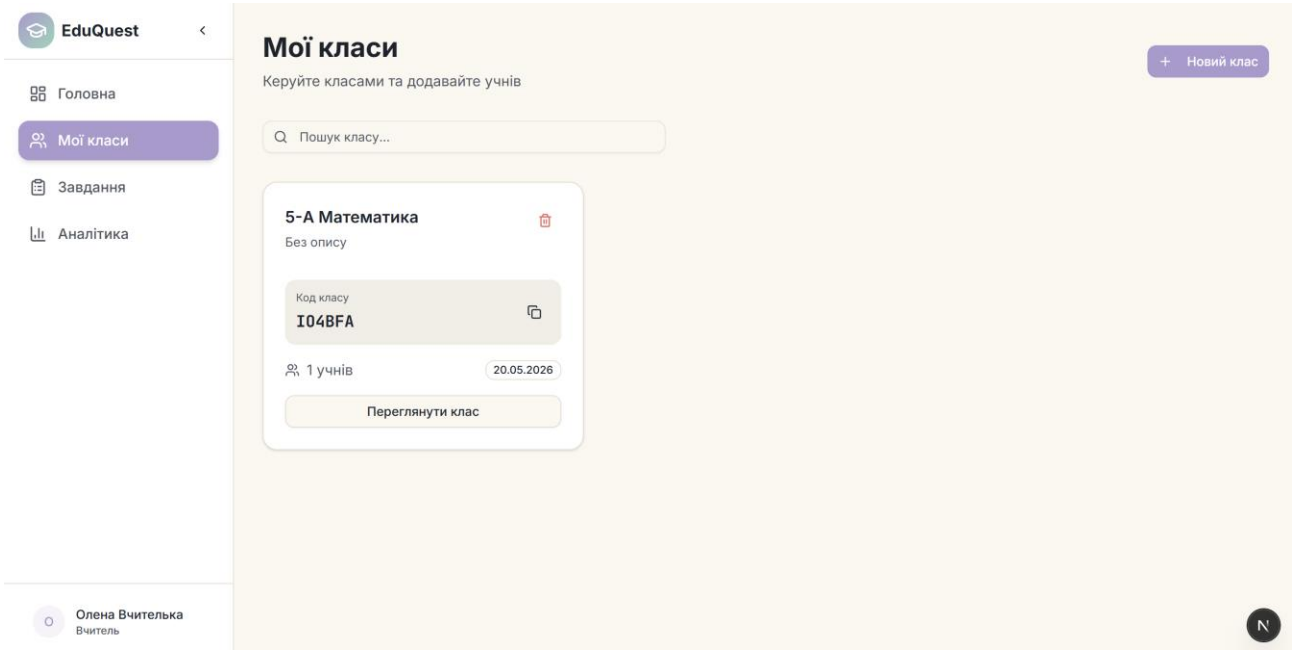


Рисунок В.18 – Сторінка класів у вчителя

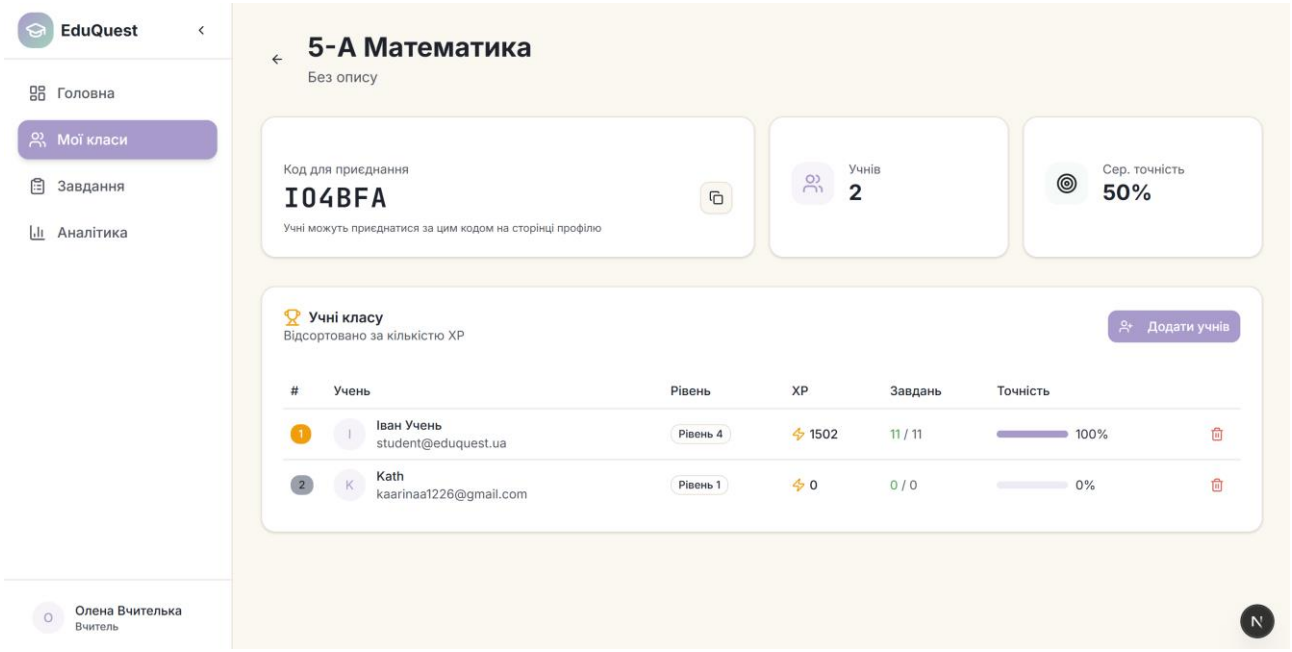


Рисунок В.19 – Сторінка створення завдання

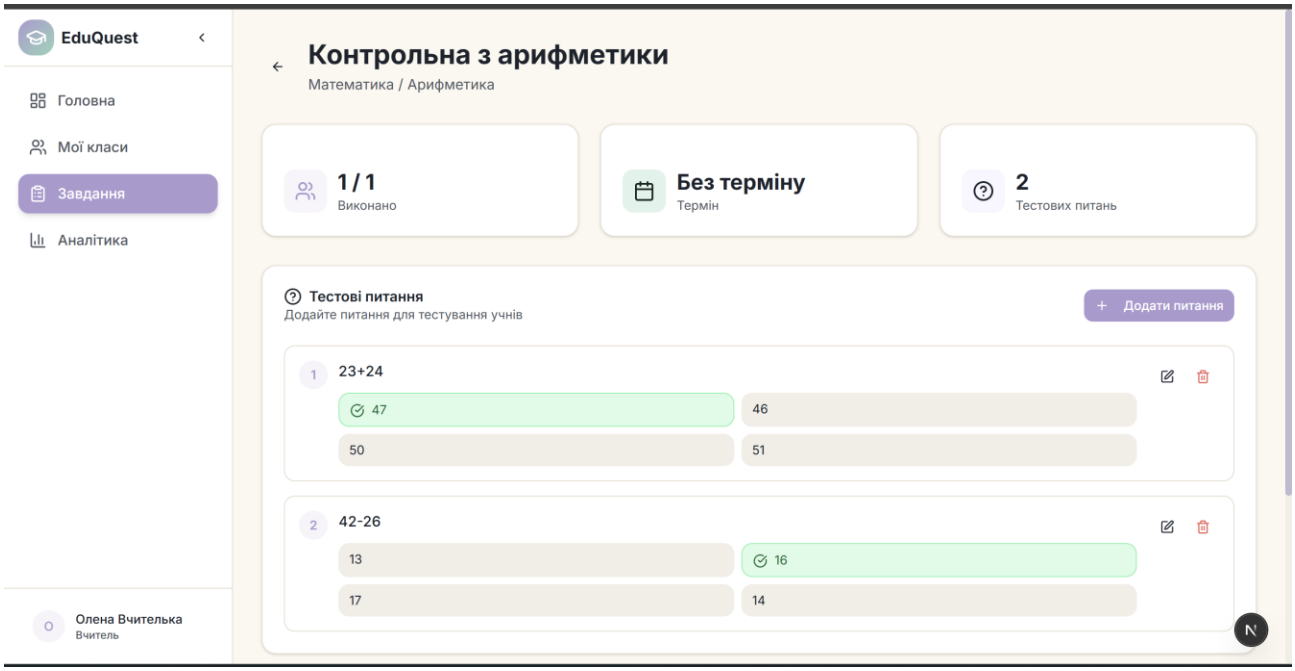


Рисунок В.20 – Сторінка додавання вправ до завдання

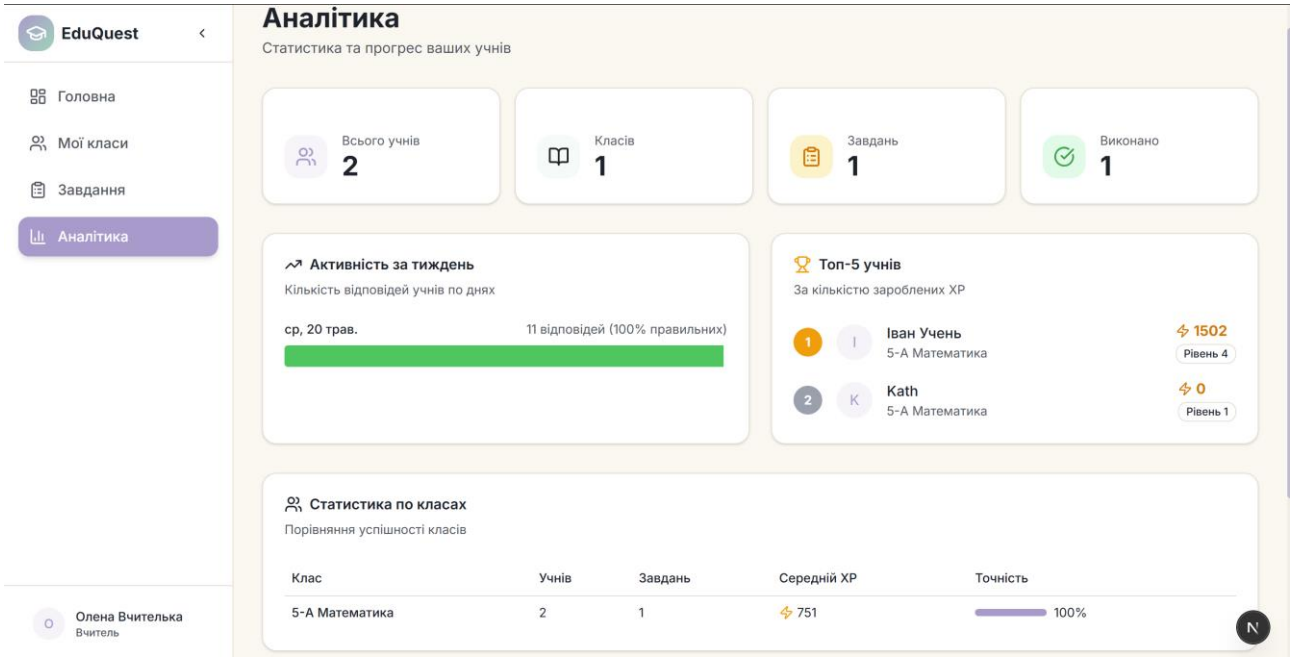


Рисунок В.21 – Сторінка аналітики у вчителя

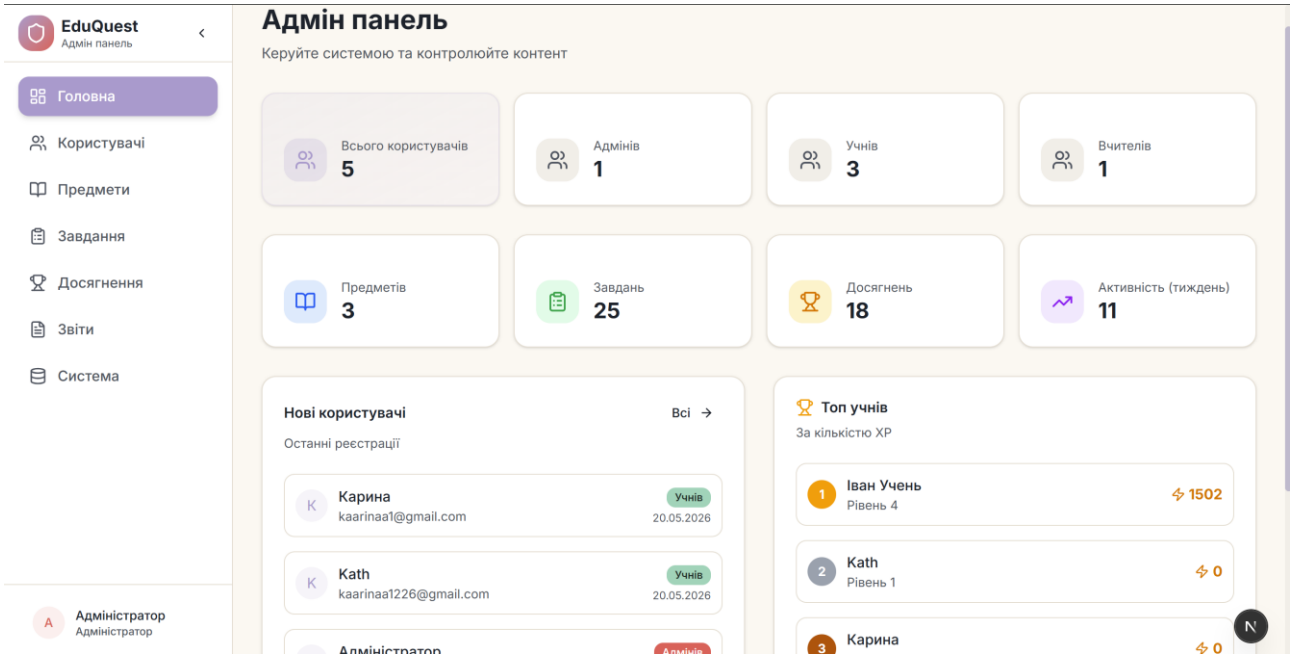


Рисунок В.22 – Адмін-панель

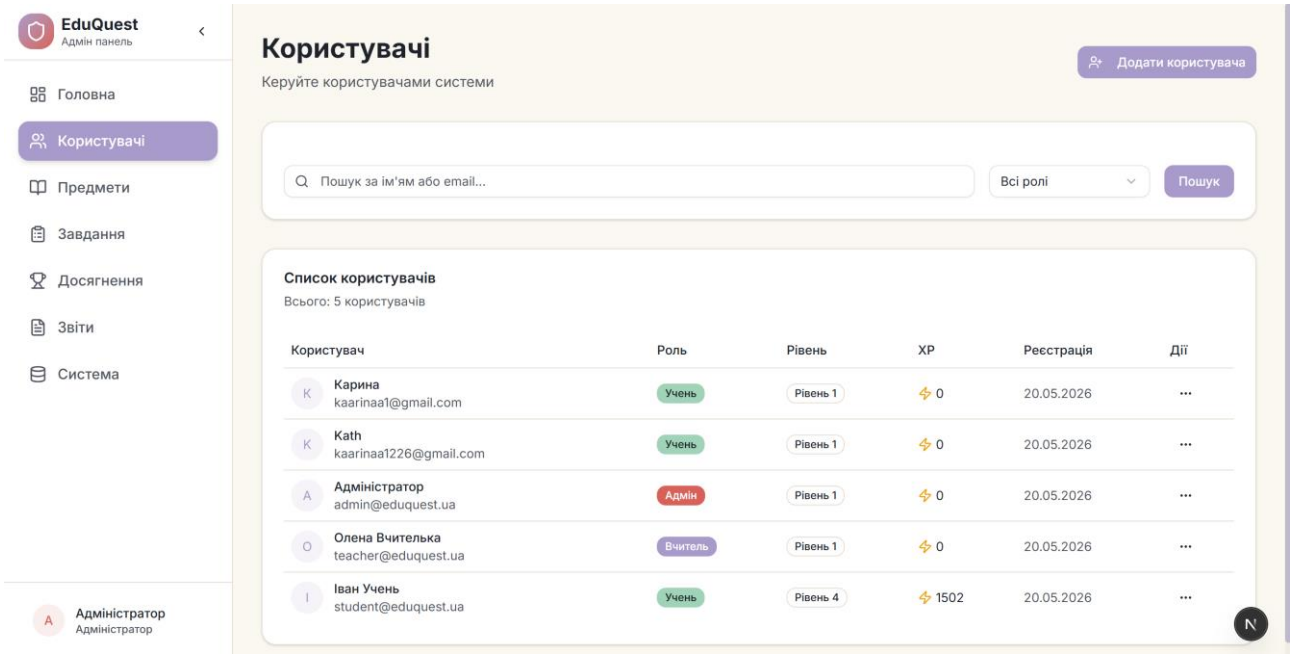


Рисунок В.23 – Сторінка користувачів в адміністратора

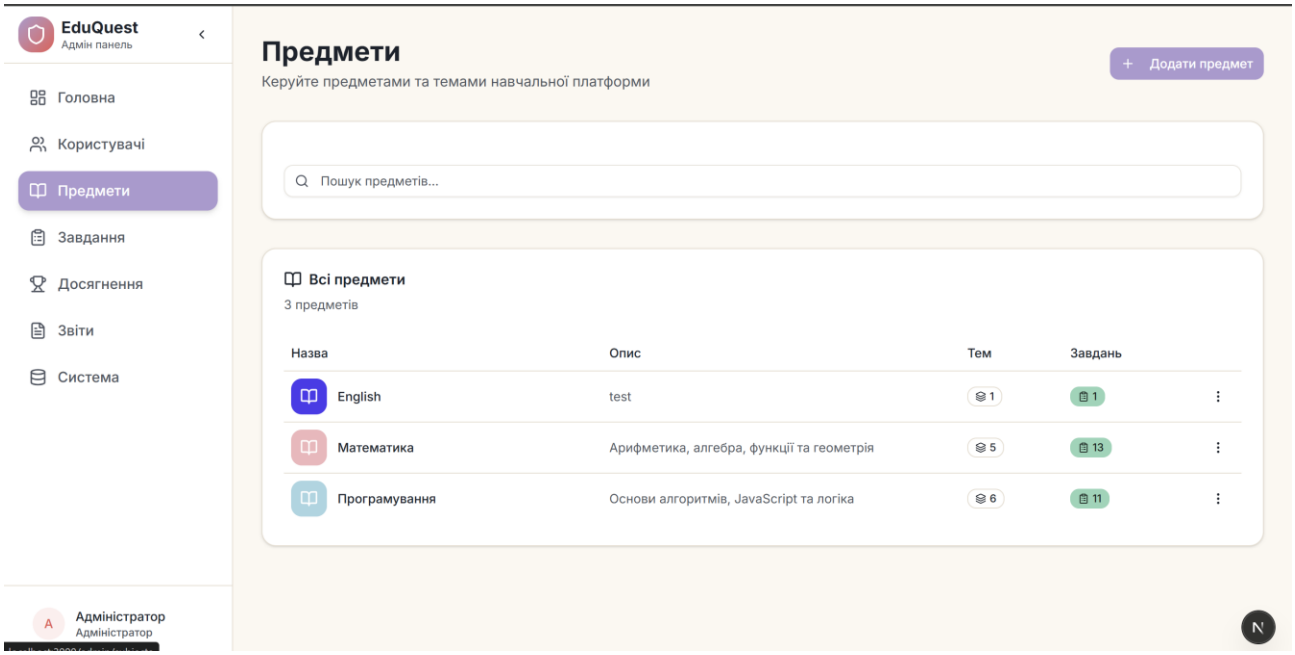


Рисунок В.24 – Сторінка предметів в адміністратора

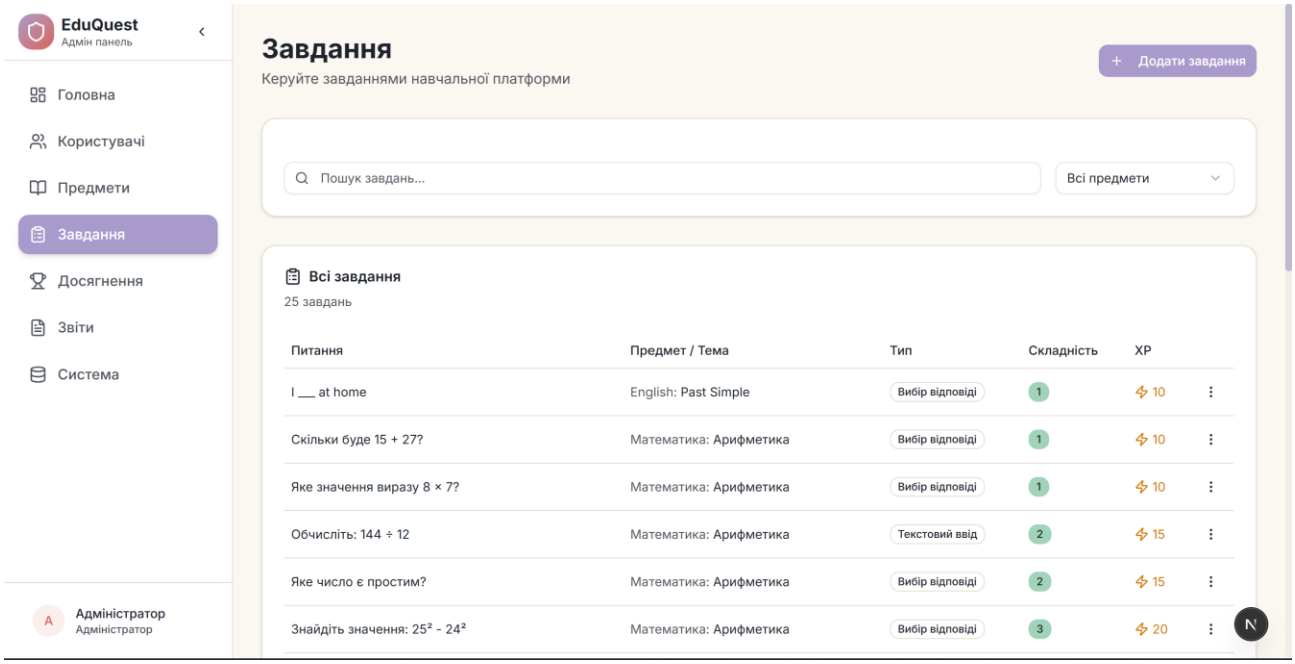


Рисунок В.25 – Сторінка завдань в адміністратора

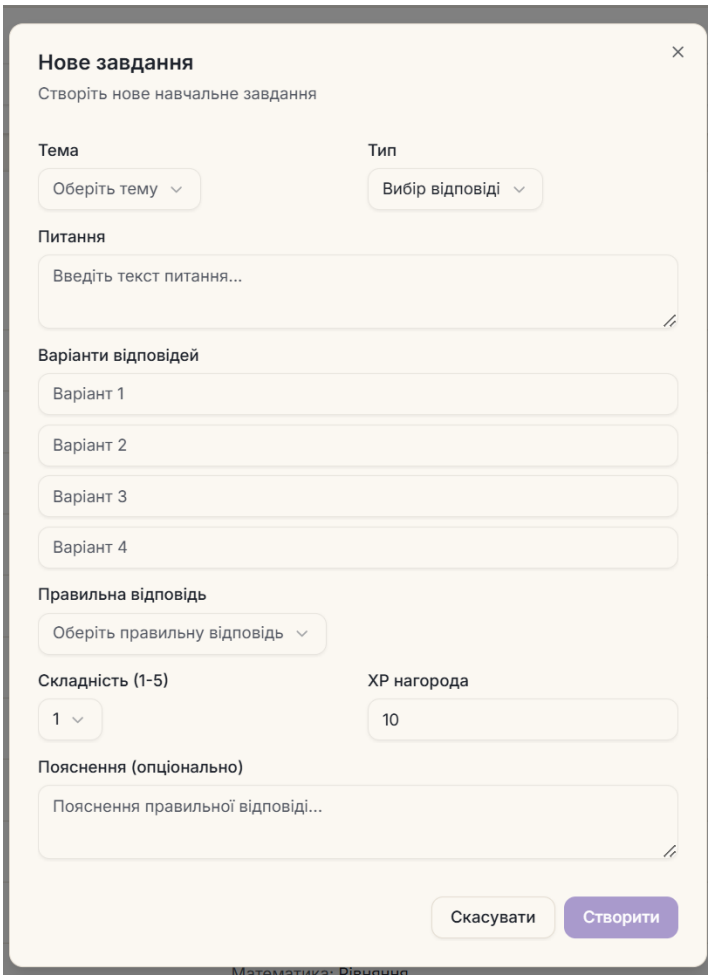


Рисунок В.26 – Вікно додавання завдання

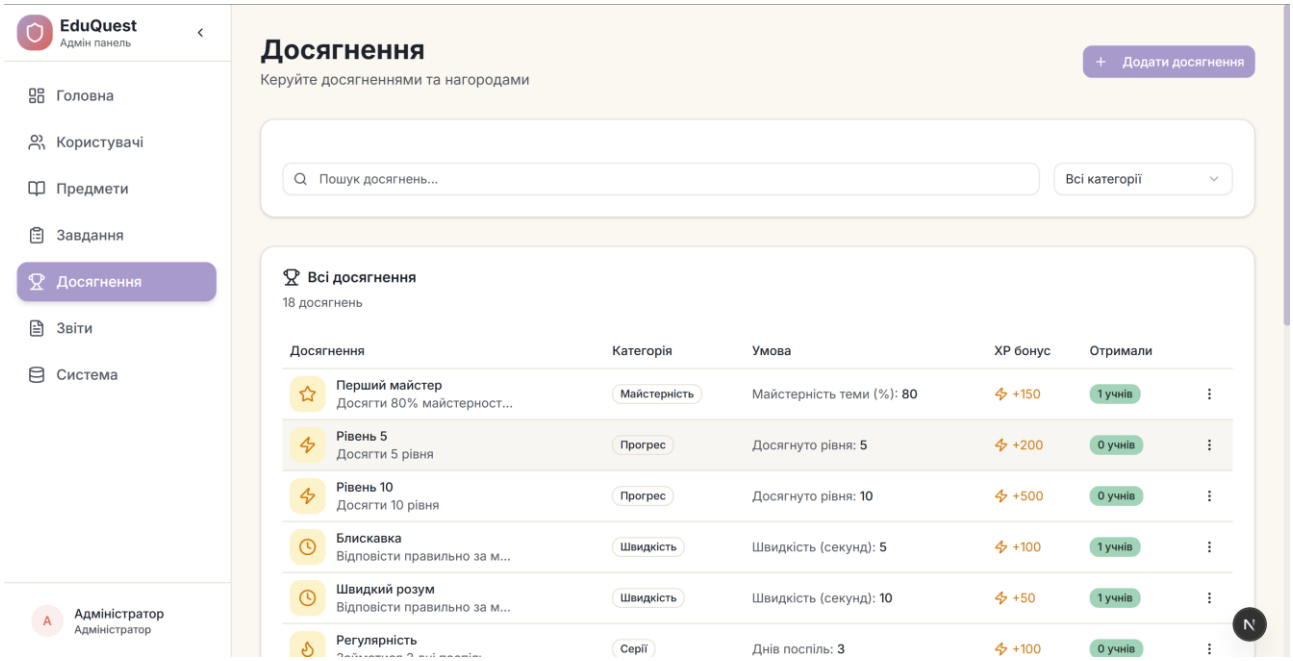


Рисунок В.27 – Сторінка досягнень в адміністратора

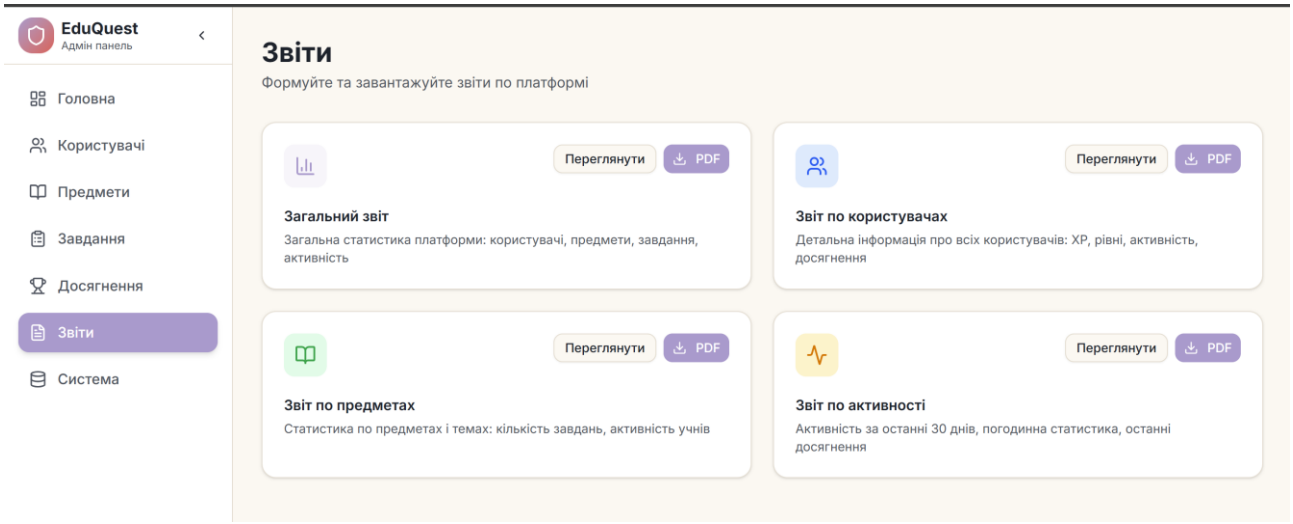


Рисунок В.28 – Сторінка звітів в адміністратора

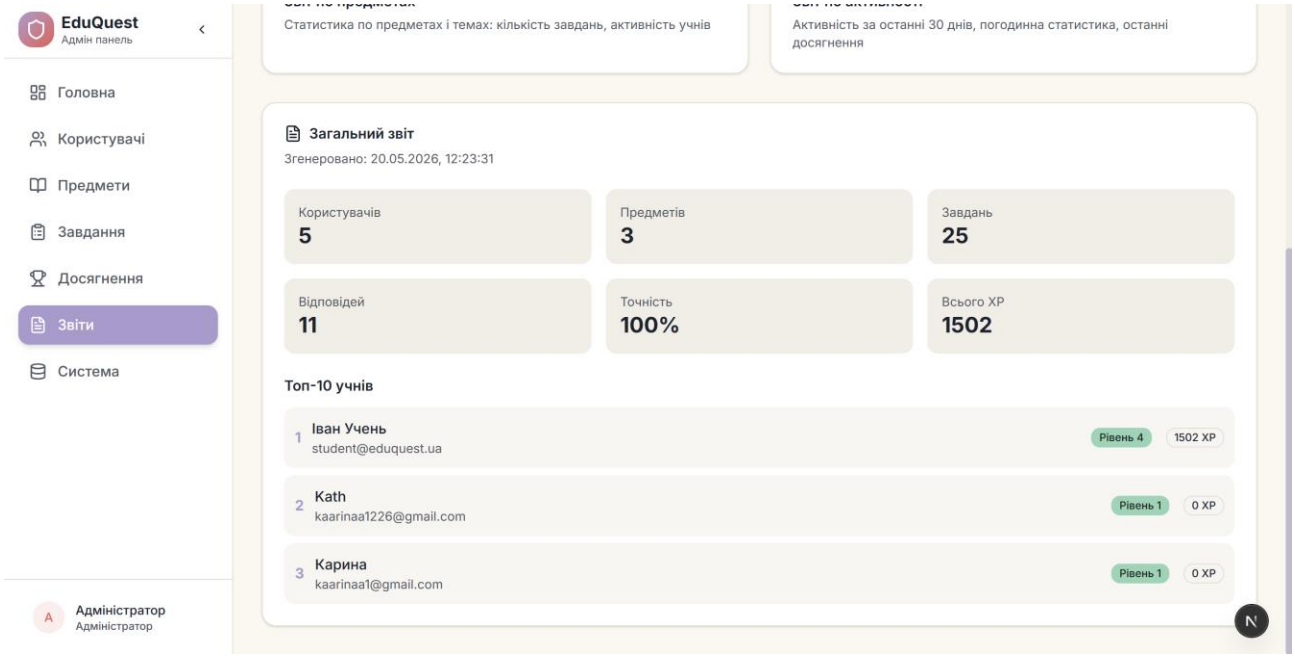


Рисунок В.29 – Приклад відображення звіту на сторінці

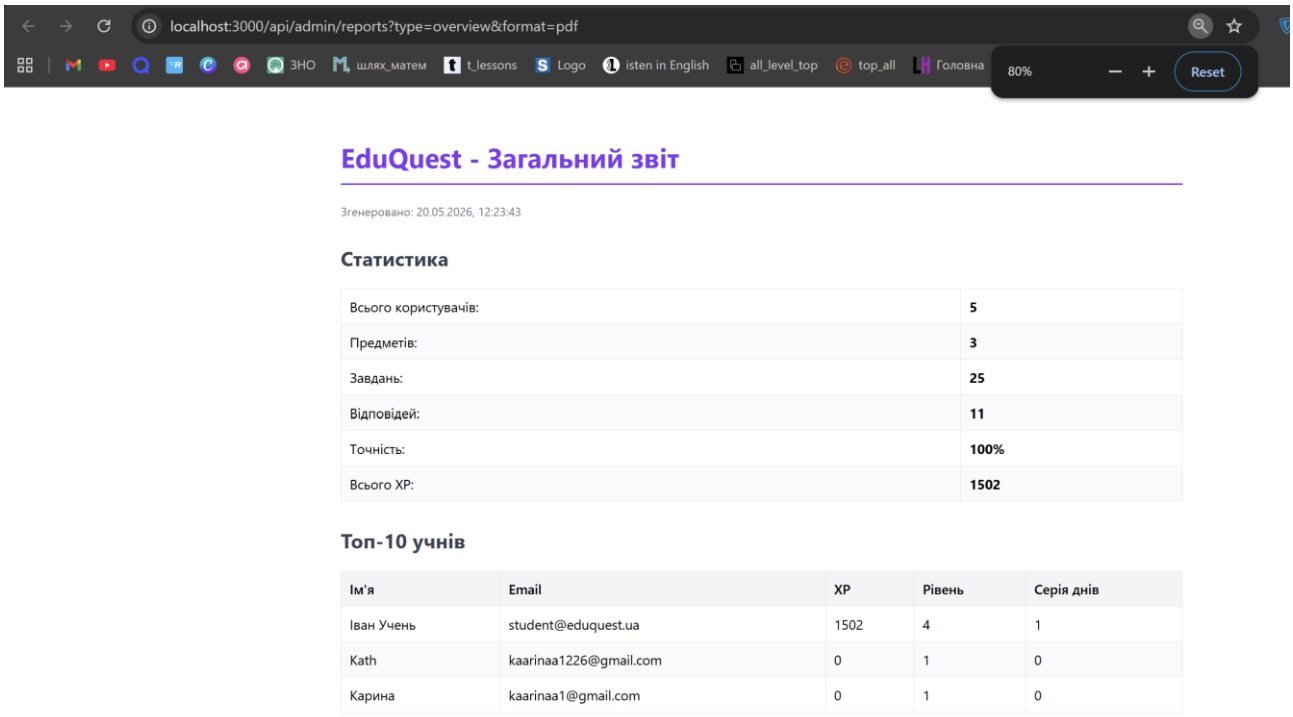


Рисунок В.30 – Приклад згенерованого звіту в PDF форматі

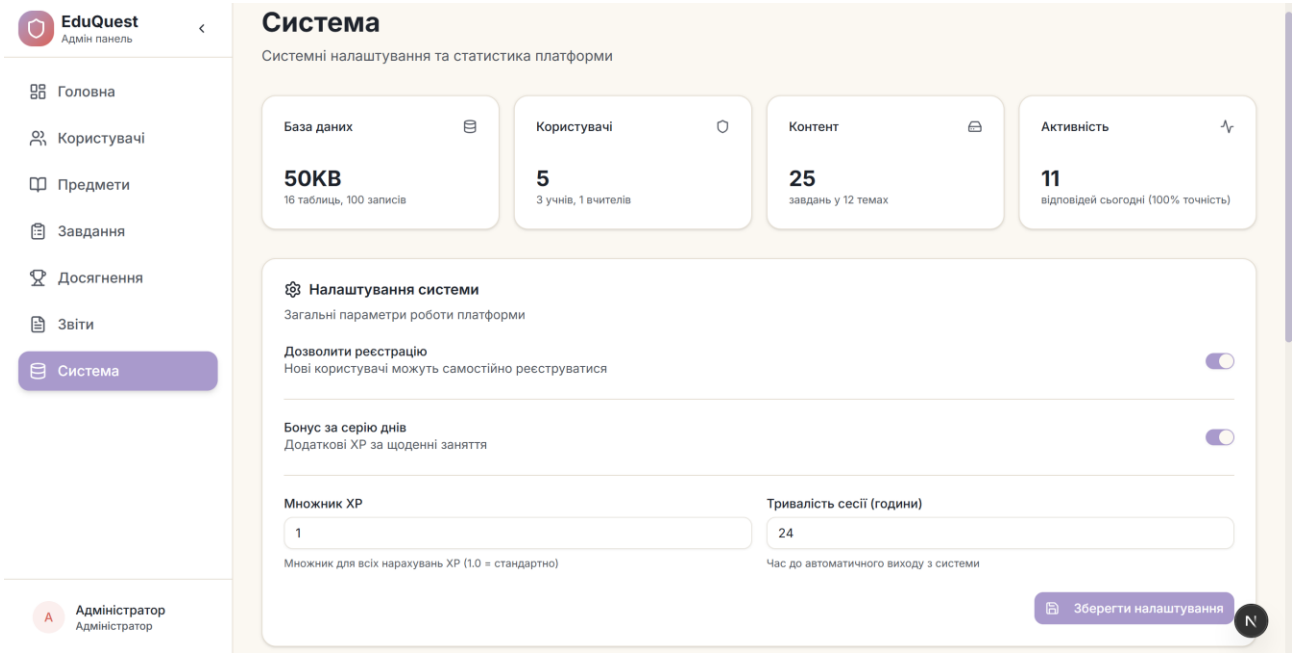


Рисунок В.31 – Сторінка системи в адміністратора (1)

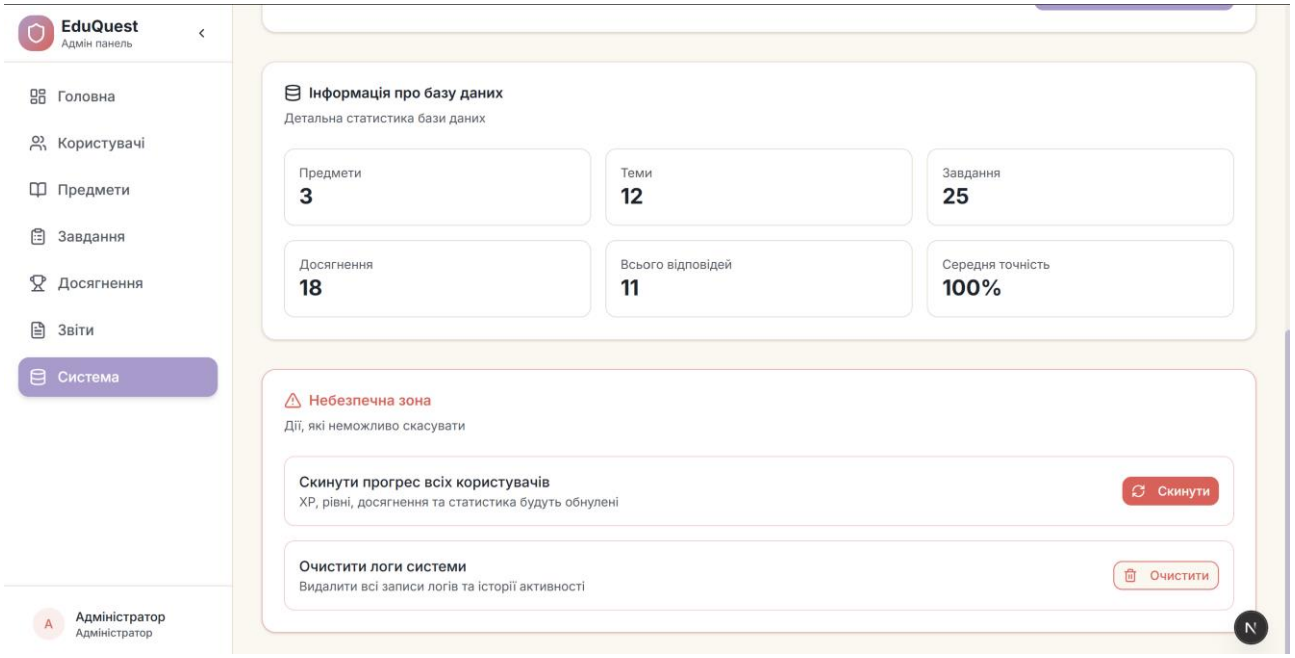


Рисунок В.32 – Сторінка системи в адміністратора (2)

ДОДАТОК Г

Лістинг – реалізація елементів гейміфікації

Лістинг Г.1 – Функції розрахунку рівнів

```
// lib/auth.ts

export function calculateLevel(xp: number): number {
  return Math.floor(Math.sqrt(xp / 100)) + 1
}

export function xpForLevel(level: number): number {
  return Math.pow(level - 1, 2) * 100
}

export function xpForNextLevel(currentXp: number): {
  current: number
  needed: number
  progress: number
} {
  const currentLevel = calculateLevel(currentXp)
  const currentLevelXp = xpForLevel(currentLevel)
  const nextLevelXp = xpForLevel(currentLevel + 1)

  return {
    current: currentXp - currentLevelXp,
    needed: nextLevelXp - currentLevelXp,
```

```
    progress: ((currentXp - currentLevelXp) / (nextLevelXp - currentLevelXp)) *
100
  }
}
```

Лістинг Г.2 – Перевірка та нагородження досягненнями

```
// lib/achievements.ts

export async function checkAndAwardAchievements(
  userId: number,
  context: { isCorrect?: boolean; timeSpent?: number } = {}
): Promise<Achievement[]> {
  const newAchievements: Achievement[] = []

  // Отримуємо всі досягнення та статистику
  const achievements = db.prepare('SELECT * FROM achievements').all()
  const earnedIds = db.prepare(
    'SELECT achievement_id FROM user_achievements WHERE user_id = ?'
  ).all(userId).map(r => r.achievement_id)
  const stats = getStudentStats(userId)
  const user = getUserById(userId)

  for (const achievement of achievements) {
    if (earnedIds.includes(achievement.id)) continue

    let earned = false

    switch (achievement.condition_type) {
```

```
  case 'tasks_completed':
    earned = stats.totalTasks >= achievement.condition_value
    break

  case 'correct_streak':
    earned = stats.currentStreak >= achievement.condition_value
    break

  case 'days_streak':
    earned = (user?.streak_days ?? 0) >= achievement.condition_value
    break

  case 'speed_demon':
    if (context.isCorrect && context.timeSpent !== undefined) {
      earned = context.timeSpent <= achievement.condition_value
    }
    break

  case 'topic_mastery':
    const mastered = db.prepare(`
      SELECT COUNT(*) as count FROM user_progress
      WHERE user_id = ? AND mastery_level >= ?
    `).get(userId, achievement.condition_value)
    earned = mastered.count > 0
    break
}

if (earned) {
  db.prepare(`
    INSERT INTO user_achievements (user_id, achievement_id)
```

```
        VALUES (?, ?)
      `).run(userId, achievement.id)

      if (achievement.xp_bonus > 0) {
        updateUserXP(userId, achievement.xp_bonus)
      }

      newAchievements.push(achievement)
    }
  }

  return newAchievements
}
```

Лістинг Г.3 – Реалізація гри «Швидкий рахунок»

```
// app/student/games/quick-calc/page.tsx
"use client"

import { useState, useEffect, useCallback } from "react"
import { Card, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Progress } from "@/components/ui/progress"

interface Question {
  num1: number
  num2: number
  operation: "+" | "-"
  answer: number
```

```
  options: number[]
}

function generateQuestion(): Question {
  const operation = Math.random() > 0.5 ? "+" : "-"
  let num1: number, num2: number, answer: number

  if (operation === "+") {
    num1 = Math.floor(Math.random() * 20) + 1
    num2 = Math.floor(Math.random() * 20) + 1
    answer = num1 + num2
  } else {
    num1 = Math.floor(Math.random() * 30) + 10
    num2 = Math.floor(Math.random() * num1) + 1
    answer = num1 - num2
  }

  // Генерація неправильних варіантів
  const options = new Set<number>([answer])
  while (options.size < 4) {
    const offset = Math.floor(Math.random() * 10) - 5
    const wrongAnswer = answer + offset
    if (wrongAnswer !== answer && wrongAnswer > 0) {
      options.add(wrongAnswer)
    }
  }

  return {
    num1, num2, operation, answer,
    options: Array.from(options).sort(() => Math.random() - 0.5)
```

```
  }
 }

 export default function QuickCalcGame() {
  const [gameState, setGameState] = useState<"idle" | "playing" | "finished">("idle")
  const [score, setScore] = useState(0)
  const [streak, setStreak] = useState(0)
  const [timeLeft, setTimeLeft] = useState(60)
  const [currentQuestion, setCurrentQuestion] = useState<Question | null>(null)

  // Таймер гри
  useEffect(() => {
   if (gameState !== "playing" || timeLeft <= 0) return

   const timer = setInterval(() => {
    setTimeLeft(t => t - 1)
   }, 1000)

   return () => clearInterval(timer)
  }, [gameState, timeLeft])

  const selectAnswer = (selected: number) => {
   if (!currentQuestion) return

   const isCorrect = selected === currentQuestion.answer

   if (isCorrect) {
    const newStreak = streak + 1
    setStreak(newStreak)
```

```
    // Бонусні очки за серію
    const points = 10 + Math.floor(newStreak / 3) * 5
    setScore(s => s + points)
  } else {
    setStreak(0)
  }

  setCurrentQuestion(generateQuestion())
}

// Збереження результату
const endGame = async () => {
  setGameState("finished")

  const baseXp = Math.floor(score / 10)
  const streakBonus = Math.floor(streak / 5) * 3
  const totalXp = Math.max(5, baseXp + streakBonus)

  await fetch("/api/games/submit", {
    method: "POST",
    body: JSON.stringify({ game: "quick-calc", score, xp: totalXp })
  })
}

return (
  <div className="space-y-6">
    {/* UI гри */}
    {gameState === "playing" && currentQuestion && (
      <Card className="py-8 text-center">
        <p className="text-5xl font-bold font-mono">
```

```
                {currentQuestion.num1}                    {currentQuestion.operation}
{currentQuestion.num2} = ?
                </p>
              </Card>
            )}

            {/* Варіанти відповідей */}
            <div className="grid grid-cols-2 gap-4">
              {currentQuestion?.options.map((option, i) => (
                <Button
                  key={i}
                  variant="outline"
                  className="h-20 text-2xl font-bold"
                  onClick={() => selectAnswer(option)}
                >
                  {option}
                </Button>
              ))}
            </div>
          </div>
        )
      }
```

Лістинг Г.4 – Оновлення серії днів

```
// lib/auth.ts

export function updateUserStreak(userId: number): number {
  const user = getUserById(userId)
```

```
  if (!user) throw new Error('Користувача не знайдено')

  const today = new Date().toISOString().split('T')[0]
  const lastActivity = user.last_activity?.split('T')[0]

  let newStreak = user.streak_days

  if (lastActivity !== today) {
    const yesterday = new Date()
    yesterday.setDate(yesterday.getDate() - 1)
    const yesterdayStr = yesterday.toISOString().split('T')[0]

    if (lastActivity === yesterdayStr) {
      // Продовження серії
      newStreak += 1
    } else if (lastActivity !== today) {
      // Перезапуск серії
      newStreak = 1
    }

    db.prepare(`
      UPDATE users
      SET streak_days = ?, last_activity = datetime('now')
      WHERE id = ?
    `).run(newStreak, userId)
  }

  return newStreak
}
```

ДОДАТОК Д

Лістинг – тестування

Лістинг Д.1 – Jest тестування

```
/**
 * Unit Tests for Authentication Logic
 * Тестуємо бізнес-логіку без залежностей від ESM модулів
 */

describe('XP and Level Calculations', () => {
  // Utility functions for XP calculation (match app logic)
  const calculateXP = (isCorrect: boolean, difficulty: number, streak: number): number => {
    if (!isCorrect) return 0
    const baseXP = 10
    const difficultyBonus = difficulty * 2
    const streakBonus = Math.floor(streak / 3) * 5
    return baseXP + difficultyBonus + streakBonus
  }

  const calculateLevel = (xp: number): number => {
    if (xp < 100) return 1
    return Math.floor(Math.sqrt(xp / 100)) + 1
  }

  const calculateXPForNextLevel = (level: number): number => {
```

```
  return level * level * 100
}

describe('calculateXP', () => {
  it('should return base XP for correct answer with difficulty 1', () => {
    const xp = calculateXP(true, 1, 0)
    expect(xp).toBe(12) // 10 base + 2 difficulty bonus
  })

  it('should return 0 XP for incorrect answer', () => {
    const xp = calculateXP(false, 1, 1)
    expect(xp).toBe(0)
  })

  it('should give more XP for harder difficulty', () => {
    const easyXP = calculateXP(true, 1, 0)
    const mediumXP = calculateXP(true, 2, 0)
    const hardXP = calculateXP(true, 3, 0)

    expect(easyXP).toBe(12)  // 10 + 1*2
    expect(mediumXP).toBe(14) // 10 + 2*2
    expect(hardXP).toBe(16)   // 10 + 3*2
    expect(hardXP).toBeGreaterThan(mediumXP)
    expect(mediumXP).toBeGreaterThan(easyXP)
  })

  it('should give bonus XP for streak of 3+', () => {
    const noStreakXP = calculateXP(true, 1, 0)
    const streak3XP = calculateXP(true, 1, 3)
    const streak6XP = calculateXP(true, 1, 6)
```

```
    expect(noStreakXP).toBe(12)  // 10 + 2 + 0
    expect(streak3XP).toBe(17)   // 10 + 2 + 5
    expect(streak6XP).toBe(22)   // 10 + 2 + 10
  })
})

describe('calculateLevel', () => {
  it('should return level 1 for 0 XP', () => {
    expect(calculateLevel(0)).toBe(1)
  })

  it('should return level 1 for 99 XP', () => {
    expect(calculateLevel(99)).toBe(1)
  })

  it('should return level 2 for 100 XP', () => {
    expect(calculateLevel(100)).toBe(2)
  })

  it('should return level 3 for 400 XP', () => {
    expect(calculateLevel(400)).toBe(3)
  })

  it('should calculate higher levels correctly', () => {
    expect(calculateLevel(900)).toBe(4)
    expect(calculateLevel(1600)).toBe(5)
  })
})
```

```
  describe('calculateXPForNextLevel', () => {
    it('should return 100 XP for level 1', () => {
      expect(calculateXPForNextLevel(1)).toBe(100)
    })

    it('should return 400 XP for level 2', () => {
      expect(calculateXPForNextLevel(2)).toBe(400)
    })

    it('should return 900 XP for level 3', () => {
      expect(calculateXPForNextLevel(3)).toBe(900)
    })
  })
})

describe('Mastery Level Calculations', () => {
  const calculateMastery = (correct: number, total: number): number => {
    if (total === 0) return 0
    return Math.round((correct / total) * 100)
  }

  const updateMastery = (
    currentMastery: number,
    isCorrect: boolean,
    weight: number = 0.1
  ): number => {
    const newValue = isCorrect ? 100 : 0
    const updatedMastery = currentMastery * (1 - weight) + newValue * weight
    return Math.round(updatedMastery)
  }
```

```
describe('calculateMastery', () => {
  it('should return 0 for no attempts', () => {
    expect(calculateMastery(0, 0)).toBe(0)
  })

  it('should return 100 for all correct', () => {
    expect(calculateMastery(10, 10)).toBe(100)
  })

  it('should return 50 for half correct', () => {
    expect(calculateMastery(5, 10)).toBe(50)
  })

  it('should round correctly', () => {
    expect(calculateMastery(1, 3)).toBe(33)
    expect(calculateMastery(2, 3)).toBe(67)
  })
})

describe('updateMastery', () => {
  it('should increase mastery for correct answer', () => {
    const updated = updateMastery(50, true)
    expect(updated).toBeGreaterThan(50)
  })

  it('should decrease mastery for incorrect answer', () => {
    const updated = updateMastery(50, false)
    expect(updated).toBeLessThan(50)
  })
```

```
    it('should use exponential moving average', () => {
      // Starting at 50%, correct answer with weight 0.1
      // 50 * 0.9 + 100 * 0.1 = 45 + 10 = 55
      expect(updateMastery(50, true, 0.1)).toBe(55)

      // Starting at 50%, incorrect answer with weight 0.1
      // 50 * 0.9 + 0 * 0.1 = 45 + 0 = 45
      expect(updateMastery(50, false, 0.1)).toBe(45)
    })
  })
})

describe('Validation Functions', () => {
  const isValidEmail = (email: string): boolean => {
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/
    return emailRegex.test(email)
  }

  const isValidPassword = (password: string): boolean => {
    return password.length >= 6
  }

  const isValidName = (name: string): boolean => {
    return name.trim().length >= 2
  }

  describe('isValidEmail', () => {
    it('should accept valid emails', () => {
      expect(isValidEmail('test@example.com')).toBe(true)
```

```
    expect(isValidEmail('user.name@domain.co.uk')).toBe(true)
    expect(isValidEmail('user+tag@example.org')).toBe(true)
  })

  it('should reject invalid emails', () => {
    expect(isValidEmail('')).toBe(false)
    expect(isValidEmail('invalid')).toBe(false)
    expect(isValidEmail('invalid@')).toBe(false)
    expect(isValidEmail('@domain.com')).toBe(false)
    expect(isValidEmail('test@.com')).toBe(false)
  })
})

describe('isValidPassword', () => {
  it('should accept passwords with 6+ characters', () => {
    expect(isValidPassword('123456')).toBe(true)
    expect(isValidPassword('password123')).toBe(true)
  })

  it('should reject passwords with less than 6 characters', () => {
    expect(isValidPassword('')).toBe(false)
    expect(isValidPassword('12345')).toBe(false)
  })
})

describe('isValidName', () => {
  it('should accept names with 2+ characters', () => {
    expect(isValidName('Jo')).toBe(true)
    expect(isValidName('John Doe')).toBe(true)
  })
```

```
    it('should reject names with less than 2 characters', () => {
      expect(isValidName('')).toBe(false)
      expect(isValidName('J')).toBe(false)
      expect(isValidName('  ')).toBe(false)
    })
  })
})
```

Button.test.tsx

```
import { render, screen } from '@testing-library/react'
import { Button } from '@/components/ui/button'

describe('Button Component', () => {
  it('renders button with text', () => {
    render(<Button>Click me</Button>)
    expect(screen.getByText('Click me')).toBeInTheDocument()
  })

  it('renders button with correct variant class', () => {
    render(<Button variant="destructive">Delete</Button>)
    const button = screen.getByText('Delete')
    expect(button).toHaveClass('bg-destructive')
  })

  it('renders disabled button', () => {
    render(<Button disabled>Disabled</Button>)
    expect(screen.getByText('Disabled')).toBeDisabled()
  })
})
```

Ui.test.tsx

```
import { render, screen, fireEvent } from '@testing-library/react'
import { Card, CardHeader, CardTitle, CardContent } from '@/components/ui/card'
import { Input } from '@/components/ui/input'

describe('Card Component', () => {
  it('renders card with title and content', () => {
    render(
      <Card>
        <CardHeader>
          <CardTitle>Test Card</CardTitle>
        </CardHeader>
        <CardContent>Card content here</CardContent>
      </Card>
    )

    expect(screen.getByText('Test Card')).toBeInTheDocument()
    expect(screen.getByText('Card content here')).toBeInTheDocument()
  })
})

describe('Input Component', () => {
  it('renders input with placeholder', () => {
    render(<Input placeholder="Enter text" />)
    expect(screen.getByPlaceholderText('Enter text')).toBeInTheDocument()
  })

  it('handles value change', () => {
```

```
    render(<Input data-testid="test-input" />)
    const input = screen.getByTestId('test-input')

    fireEvent.change(input, { target: { value: 'test value' } })
    expect(input).toHaveValue('test value')
  })

  it('renders disabled input', () => {
    render(<Input disabled placeholder="Disabled" />)
    expect(screen.getByPlaceholderText('Disabled')).toBeDisabled()
  })
})
```

Лістинг Д.2 – E2E тестування

```
import { test, expect } from '@playwright/test'

// Basic UI tests that don't require database/authentication
test.describe('Home Page', () => {
  test('should load home page', async ({ page }) => {
    await page.goto('/')
    await expect(page).toHaveTitle(/EduQuest/)
  })
})

test.describe('Login Page', () => {
  test('should show login form', async ({ page }) => {
    await page.goto('/login')
    await expect(page.locator('input[type="email"]')).toBeVisible()
```

```
    await expect(page.locator('input[type="password"]')).toBeVisible()
    await expect(page.locator('button[type="submit"]')).toBeVisible()
  })

  test('should have demo accounts section', async ({ page }) => {
    await page.goto('/login')
    await expect(page.locator('text=Демо-акаунти')).toBeVisible()
  })

  test('should have link to register', async ({ page }) => {
    await page.goto('/login')
    await expect(page.locator('a[href="/register"]')).toBeVisible()
  })

  test('should show error for invalid credentials', async ({ page }) => {
    await page.goto('/login')
    await page.fill('input[type="email"]', 'invalid@test.com')
    await page.fill('input[type="password"]', 'wrongpassword')
    await page.click('button[type="submit"]')

    // Wait for any error message (server responds with error)
    const errorElement = page.locator('.bg-destructive\\/10')
    await expect(errorElement).toBeVisible({ timeout: 10000 })
  })
})

test.describe('Register Page', () => {
  test('should show registration form', async ({ page }) => {
    await page.goto('/register')
    // Use id selector since the input has id="name" not name="name"
```

```
    await expect(page.locator('input#name')).toBeVisible()
    await expect(page.locator('input[type="email"]')).toBeVisible()
    await expect(page.locator('input#password')).toBeVisible()
    await expect(page.locator('input#confirmPassword')).toBeVisible()
  })

  test('should have role selection options', async ({ page }) => {
    await page.goto('/register')
    await expect(page.locator('text=Учень')).toBeVisible()
    await expect(page.locator('text=Вчитель')).toBeVisible()
  })

  test('should have link to login', async ({ page }) => {
    await page.goto('/register')
    await expect(page.locator('a[href="/login"]')).toBeVisible()
  })

  test('should validate password mismatch', async ({ page }) => {
    await page.goto('/register')
    await page.fill('input#name', 'Test User')
    await page.fill('input[type="email"]', 'newuser@test.com')
    await page.fill('input#password', 'password123')
    await page.fill('input#confirmPassword', 'differentpassword')
    await page.click('button[type="submit"]')

    // Should show password mismatch error
    await expect(page.locator('text=Паролі не співпадають')).toBeVisible({
timeout: 5000 })
  })
})
```