НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет (ННІ) інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету Інформаційних технологій	**Завідувач кафедри комп'ютерних наук**
_____ / Ігор Болбот /	_____ / Белла Голуб /
(підпис) (ім'я прізвище)	(підпис) (ім'я прізвище)
«___» _____ 2025р.	«___» _____ 2025р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Програмне забезпечення системи дистанційного електронного навчання

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
доц.к.ф.-м.н. / Віктор Кириченко /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Керівник магістерської кваліфікаційної роботи
старший викладач / Юрій Міловідов /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Консультант магістерської кваліфікаційної роботи
к.т.н., доцент / Іван Пархоменко /
(науковий ступінь та вчене звання) (підпис) (ім'я прізвище)

Виконав _____ / Богдан Токарець /
(підпис) (ім'я прізвище)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
доцент, к.т.н. _____ _____ Голуб Б. Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
«10» листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Токарцю Богдану Олександровичу
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія програмного забезпечення»
(код і назва)

Освітня програма «Програмне забезпечення інформаційних систем»
(назва)

Орієнтація освітньої програми освітньо-професійна

Тема магістерської кваліфікаційної роботи: «Програмне забезпечення системи дистанційного електронного навчання»
затверджена наказом ректора НУБіП України від « 1 » листопада 2024 р. № 1963 «С»

Термін подання завершеної роботи на кафедру 14 листопада 2025 р.

Вихідні дані до магістерської кваліфікаційної роботи: вимоги Міністерства освіти і науки України до організації дистанційного навчання, звіти Міністерства освіти і науки України, міжнародних освітніх організацій (UNESCO, OECD) щодо стану та перспектив розвитку дистанційного навчання в Україні та світі.

Перелік питань, що підлягають дослідженню:

1. Проаналізувати сучасні технології та платформи, що використовуються в системах електронної освіти.
2. Дослідити вплив електронної освіти на академічні результати та мотивацію студентів.
3. Оцінити рівень готовності навчальних закладів до інтеграції електронних освітніх систем в освітній процес.
4. Розробити рекомендації щодо вдосконалення та ефективного впровадження систем електронної освіти.

Перелік графічного матеріалу (за потреби)

Дата видачі завдання 7 листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Юрій Міловідов
(підпис) (ім'я прізвище)

Консультант магістерської кваліфікаційної роботи _____ Іван Пархоменко
(підпис) (ім'я прізвище)

Завдання прийняв до виконання _____ Богдан Токарець
(підпис) (ім'я прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API – Application Programming Interface, інтерфейс програмування застосунків.
2. CDN – Content Delivery Network, мережа доставки контенту.
3. ERP – Enterprise Resource Planning, система управління освітніми та адміністративними процесами.
4. GUI – Graphical User Interface, графічний інтерфейс користувача.
5. HTTP/HTTPS – HyperText Transfer Protocol / Secure, протокол передавання гіпертексту та його захищена версія.
6. ID – Identifier, унікальний ідентифікатор об’єкта.
7. IdP – Identity Provider, сервіс автентифікації користувачів.
8. JSON – JavaScript Object Notation, формат структурованих даних.
9. JWT – JSON Web Token, токен авторизації.
10. KPI – Key Performance Indicator, ключовий показник ефективності.
11. Ki – Knowledge Index, коефіцієнт рівня знань студента.
12. LMS – Learning Management System, система управління навчанням.
13. MDX – MultiDimensional eXpressions, мова запитів для багатовимірних аналітичних моделей.
14. OLAP – On-Line Analytical Processing, оперативна аналітична обробка даних.
15. OOP – Object-Oriented Programming, об’єктно-орієнтоване програмування.
16. REST – Representational State Transfer, стилістика побудови веб-сервісів.
17. SQL – Structured Query Language, мова запитів до баз даних.
18. SQLite – легковага реляційна система керування базами даних, вбудована у клієнтське середовище.
19. SSO – Single Sign-On, механізм єдиного входу.
20. UI – User Interface, інтерфейс користувача.

21. XML – eXtensible Markup Language, формат представлення структурованих даних.

22. Naive Bayes – наївний баєсівський класифікатор, статистичний метод класифікації.

23. TestEngine – модуль виконання та оцінювання тестів.

24. KiCalculator – модуль розрахунку коефіцієнта знань *Ki*.

25. Results_Fact – фактова таблиця результатів тестування у моделі даних.

26. Course_Dim – вимірна таблиця курсів у логічній моделі даних.

27. Group_Dim – вимірна таблиця навчальних груп.

28. Date_Dim – вимірна таблиця дат для аналітики.

29. Value – фактичне значення KPI.

30. Goal – цільове значення KPI.

31. Status – індикатор стану KPI.

32. Trend – індикатор зміни тенденції KPI.

ВСТУП

Особливої актуальності набуває розвиток систем дистанційного електронного навчання, здатних забезпечити якісну, інтерактивну та персоналізовану взаємодію між викладачем і студентом незалежно від їхнього місця перебування. Традиційні освітні платформи часто не відповідають вимогам адаптивності, масштабованості та інтеграції з сучасними інформаційними сервісами, що обмежує ефективність навчального процесу. Використання технологій веб-сервісів, хмарних інфраструктур, мультимедійних контент-менеджерів та інтелектуальних аналітичних модулів створює передумови для формування єдиного цифрового освітнього середовища, орієнтованого на компетентнісний підхід і безперервний розвиток користувачів.

Метою роботи є розроблення програмного забезпечення системи дистанційного електронного навчання, яка забезпечує інтеграцію освітнього контенту, управління курсами, контроль знань і зворотний зв’язок із користувачами за допомогою сучасних веб-технологій і сервісів аналітики навчальних даних.

Для досягнення поставленої мети необхідно розв’язати такі **завдання:**

1. провести системний аналіз предметної області дистанційного навчання та існуючих технологічних рішень.
2. Сформувати вимоги до функціональних, нефункціональних і безпекових характеристик системи.
3. Розробити архітектуру програмного забезпечення на основі принципів модульності, масштабованості та сумісності.
4. Створити UML-моделі основних процесів системи (use case, sequence, activity).
5. Реалізувати прототип програмного забезпечення з використанням сучасного стеку технологій.
6. Провести тестування та оцінку ефективності розробленої системи у контексті навчального середовища.

Об’єктом дослідження є процес організації дистанційного електронного навчання у вищих навчальних закладах.

Предметом дослідження є методи, засоби та інформаційні технології побудови програмного забезпечення систем дистанційного навчання.

Наукова новизна полягає у створенні інтегрованої архітектури програмного забезпечення системи дистанційного електронного навчання, що поєднує модульну структуру управління навчальними процесами з аналітичними та адаптивними підсистемами. Запропоновано концепцію інтелектуального компонента збору й оброблення освітніх даних (learning analytics), який дозволяє автоматизовано аналізувати прогрес студентів і формувати персоналізовані рекомендації для покращення засвоєння матеріалу. На відміну від традиційних LMS-платформ, запропоноване рішення передбачає використання методів об’єктно-орієнтованого та подійно-орієнтованого моделювання, що підвищує масштабованість та гнучкість системи при зміні структури курсів і ролей користувачів.

Методи дослідження ґрунтуються на використанні системного підходу до аналізу предметної області, методів UML-моделювання для опису функціональної структури системи, а також методів об’єктно-орієнтованого проєктування (ООП) при розробленні архітектури програмного забезпечення. Для побудови логіки бізнес-процесів застосовано моделі «клієнт–сервер» і REST-взаємодії, а для оцінювання ефективності – методи експериментального тестування, статистичного аналізу продуктивності та показників взаємодії користувачів.

Практична значущість одержаних результатів полягає у можливості використання розробленого програмного забезпечення як базової платформи для організації дистанційного навчання у закладах освіти різного рівня акредитації. Система забезпечує автоматизацію управління навчальними курсами, проведення тестувань, контроль відвідуваності та результатів, а також формування аналітичних звітів для викладачів і адміністраторів.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області системи дистанційного навчання

У сучасному освітньому середовищі система дистанційного електронного навчання виступає інтеграційною платформою, що забезпечує автоматизовану взаємодію між студентами, викладачами та адміністраторами у процесі навчання, контролю знань та аналітики результатів. Основними функціями системи є управління навчальними курсами, організація навчальних матеріалів, проведення тестувань, оцінювання результатів і формування звітів для адміністрування навчального процесу. Такі системи дозволяють реалізувати безперервне навчання, незалежно від місця перебування користувачів, та ефективно інтегруються з іншими освітніми інструментами (ERP-системи, електронні журнали, бібліотечні сервіси).

На рис. 1.1 подано DFD-діаграму 0-го рівня предметної області системи дистанційного електронного навчання, яка відображає основні зовнішні сутності (Студент, Викладач, Адміністратор, ERP/E-журнал) та взаємодію з центральним процесом «0 Система дистанційного електронного навчання». Потоки даних включають запити контенту, публікацію матеріалів, управління обліковими записами та синхронізацію результатів.

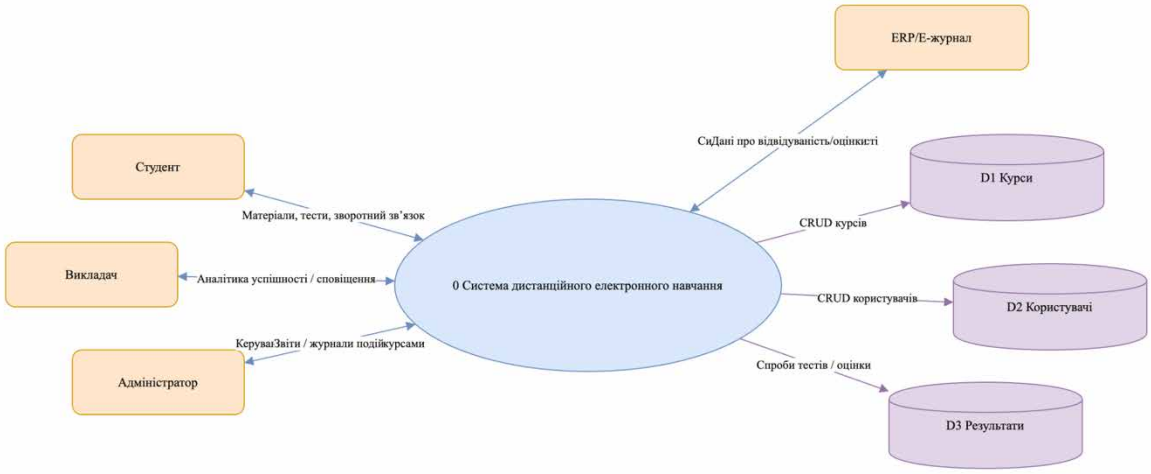


Рисунок 1.1 - DFD 0-рівня системи дистанційного електронного навчання

На рис. 1.2 наведено BPMN-діаграму, яка відображає динаміку процесів взаємодії між ролями системи. Студент проходить автентифікацію, обирає курс і виконує тестування; викладач публікує матеріали, переглядає аналітику й оцінює результати; адміністратор здійснює контроль користувачів і курсів. Центральна підсистема LMS реалізує бізнес-процеси автентифікації, доступу до контенту, оброблення спроб тестів, оцінювання та синхронізації даних із ERP-системою.

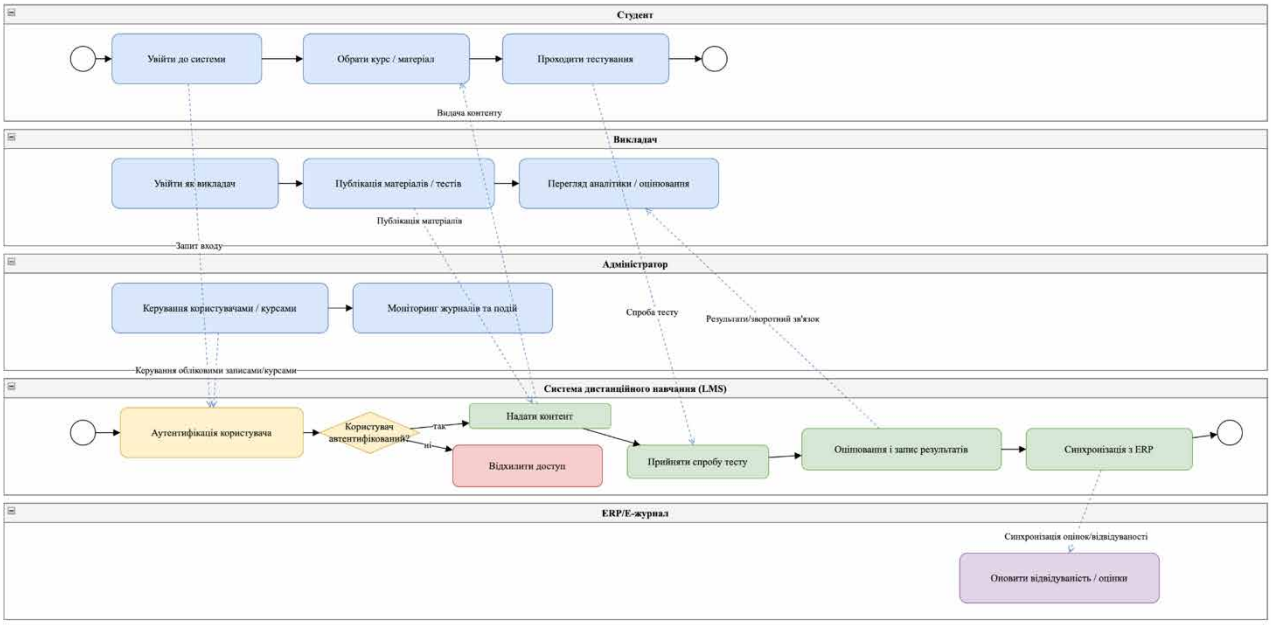


Рисунок 1.2 - BPMN-діаграма процесів системи дистанційного електронного навчання

Для кращого узагальнення сутностей предметної області у табл. 1.1 наведено основні об’єкти та їхню функціональну роль у межах системи.

Таблиця 1.1

Основні сутності предметної області системи дистанційного електронного навчання

Сутність	Опис	Основні функції	Тип даних
Студент	Користувач, який проходить навчальні курси	Отримання матеріалів, виконання тестів, перегляд результатів	Обліковий запис, результати тестів
Викладач	Користувач, що створює та адмініструє навчальні курси	Публікація контенту, перевірка тестів, формування звітів	Курси, аналітика результатів

Продовження таблиці 1.1

Адміністратор	Користувач із розширеними правами доступу	Керування користувачами, курсами, системними параметрами	Облікові дані, логи подій
LMS-ядро	Центральна обчислювальна підсистема	Обробка запитів, зберігання даних, аналітика, інтеграція з ERP	Навчальні матеріали, метадані
ERP/E-журнал	Зовнішня система освітнього менеджменту	Синхронізація оцінок, відвідуваності та успішності	Записи оцінок, журнали подій
Бази даних D1–D3	Сховища структурованої інформації	D1 – курси, D2 – користувачі, D3 – результати	SQL/NoSQL об’єкти

Предметна область системи дистанційного електронного навчання характеризується інтеграцією між учасниками освітнього процесу, централізованим управлінням навчальними ресурсами, автоматизацією тестування та збором аналітики ефективності навчання, що створює основу для побудови адаптивного освітнього середовища нового покоління.

1.2 Теоретико-методологічні засади та стан наукових досліджень

У сучасних дослідженнях дистанційного навчання значна увага приділяється адаптивним моделям подання освітнього контенту, що враховують індивідуальні характеристики користувачів. У працях Barchenko N., Tolbatov V., Lavryk T. [1] представлено математичну модель адаптивної технології в e-learning-системах, де навчальний процес описується як ітераційна взаємодія між користувачем, контентом і коефіцієнтом знань K. На рис. 1.3 показано три етапи формування моделі навчання: початкову лінійну структуру подання матеріалу, замкнену схему з зворотним зв’язком через параметр KK та розширену адаптивну модель, у якій результати оцінювання впливають на побудову наступних завдань.

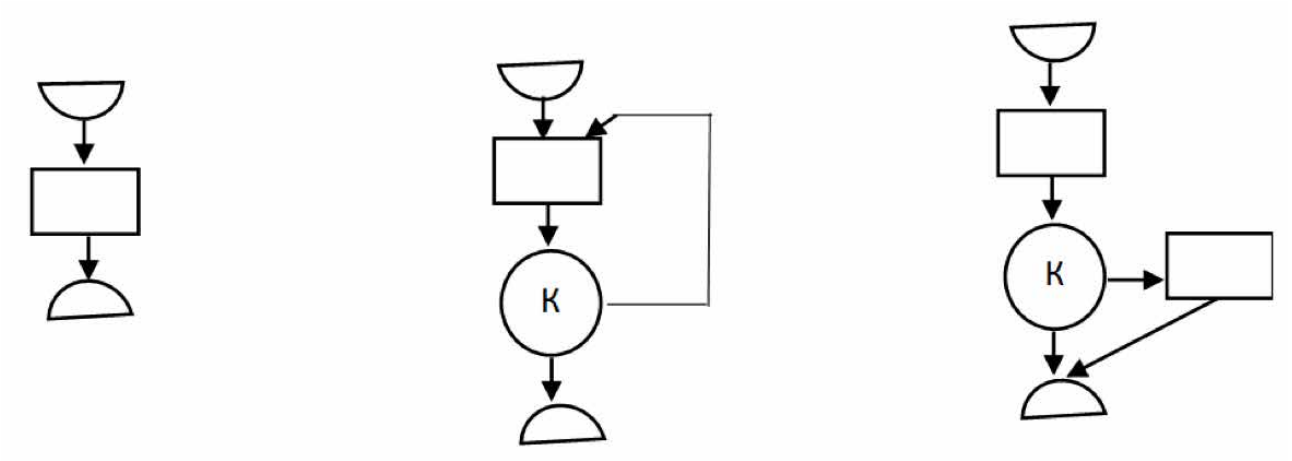


Рисунок 1.3 - Етапи формування адаптивної моделі навчання із використанням коефіцієнта знань K (за Barchenko et al., 2022)

Подальший розвиток теоретичних моделей відображено у структурі логічного дерева знань (рис. 1.4), де вузли X,Y,Z,V та їх підмножини (x1,x2),(y1,y2) тощо репрезентують компоненти навчального контенту. Модель забезпечує ієрархічне представлення освітніх об’єктів і формує функцію доцільності D=f(X,Y,Z,V), що використовується для обчислення агрегованого показника відповідності навчального матеріалу індивідуальним особливостям студента [1].

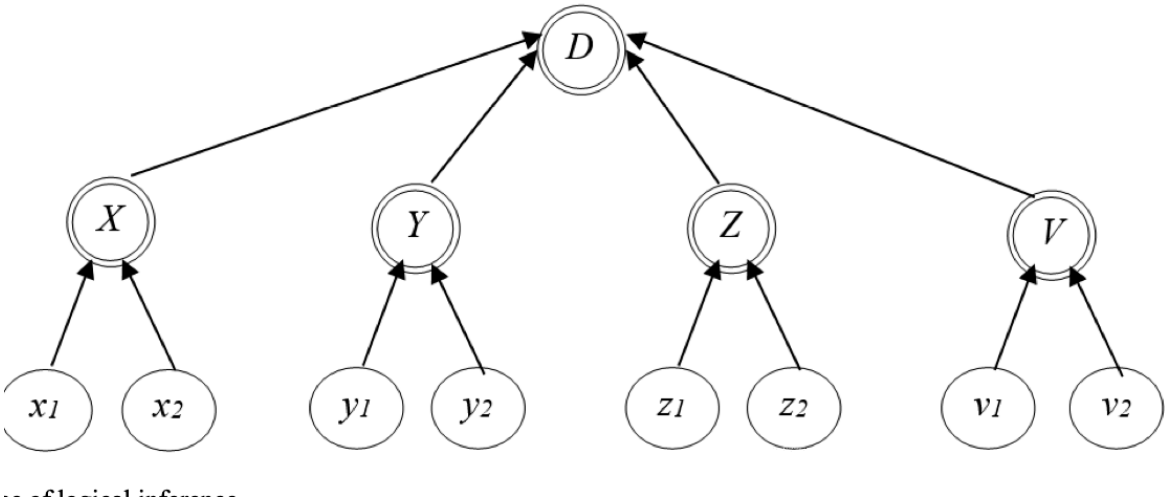


Рисунок 1.4 - Ієрархічна структура зв’язків між компонентами навчального контенту (за Barchenko et al., 2022)

На основі ієрархічної моделі було побудовано аналітичний підхід до визначення ступеня експліцитності (відкритості) компонентів е-модулів, де кожен параметр x1,y1,z1,v1 характеризує деталізацію й глибину пояснення

навчального матеріалу. На рис. 1.5 показано динаміку зміни ступеня експліцитності для чотирьох послідовних е-модулів, що демонструє перехід від поверхневого ознайомлення до повного розкриття теми та відповідного зростання когнітивного навантаження користувача. Цей підхід дозволяє формалізувати індивідуальний рівень деталізації залежно від результатів тестування.

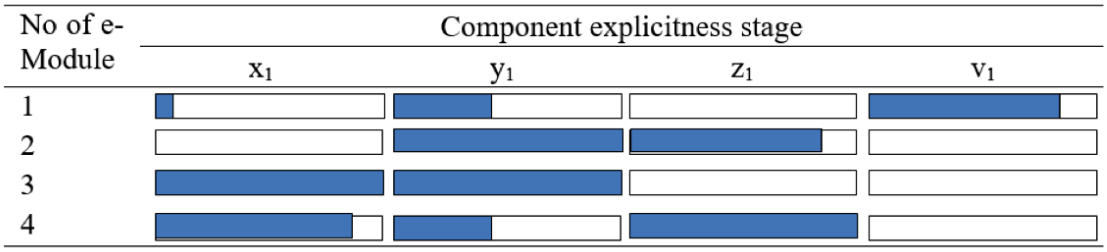


Рисунок 1.5 - Графічне представлення ступеня експліцитності компонентів е-модулів (за Barchenko et al., 2022)

Індивідуалізація навчання реалізується через аналіз переваг студента, які оцінюються за рівнем інтересу до кожного компонента контенту (рис. 1.6). Графічна модель показує співвідношення між особистими перевагами та ступенем експліцитності, що дозволяє системі здійснювати адаптивний вибір матеріалів з урахуванням індивідуального профілю користувача. Подібний підхід до персоналізації навчання застосовується в моделях Sinitsyn E. та Larionova V. [2], де визначаються оптимальні маршрути засвоєння знань через аналіз поведінкових метрик.

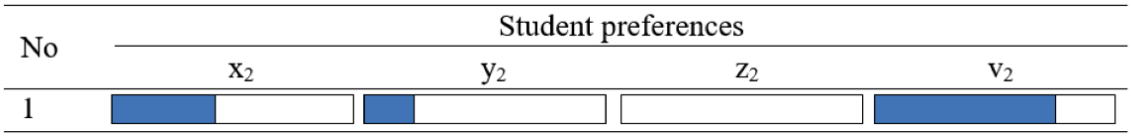


Рисунок 1.6 - Модель оцінювання переваг студента відносно навчальних компонентів (за Barchenko et al., 2022)

На завершальному етапі модель Barchenko та співавт. [1] передбачає інтеграцію баз даних та програмних модулів у єдину архітектуру (рис. 1.7). У цій структурі підсистема Evaluation здійснює оцінювання параметрів е-модулів і студентів, модуль Selection вибирає оптимальний навчальний об’єкт на основі визначених переваг, а підсистема Optimization of Human-Machine

Interaction виконує аналіз варіантів та формує рекомендації щодо найкращої навчальної траєкторії. Подібна архітектура дозволяє реалізувати адаптацію в реальному часі та оптимізувати інтерактивну взаємодію користувача з освітнім контентом.

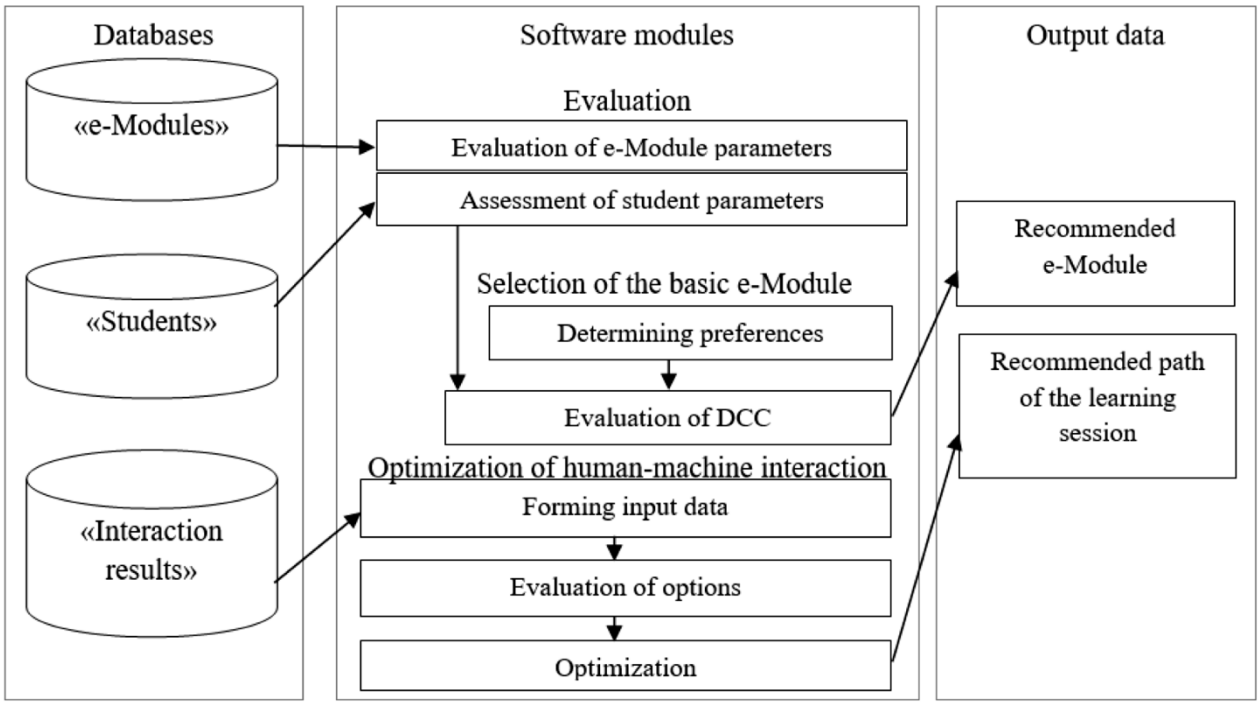


Рисунок 1.7 - Архітектурна модель адаптивної e-learning системи (за Barchenko et al., 2022)

На основі аналізу наведених моделей можна зробити висновок, що сучасні підходи до адаптивного навчання поєднують математичні моделі когнітивного стану, ієрархічні структури контенту та аналітику користувацьких переваг. У рамках нашого дослідження розвинено ці методологічні засади шляхом створення інтелектуального алгоритмічного ядра Java Swing-системи, що поєднує два нові елементи:

1.3 Аналіз існуючих рішень

Розвиток сучасних систем дистанційного електронного навчання супроводжується активним формуванням інтегрованих середовищ управління контентом, користувачами та аналітикою навчального процесу. Серед найпоширеніших рішень слід відзначити Moodle, Google Classroom, Canvas

LMS, Blackboard Learn та Blend-ed Platform, кожна з яких реалізує власну архітектурну ідеологію та підхід до автоматизації освітнього циклу.

На рис. 1.8 подано інтерфейс системи Moodle, яка є найпопулярнішою платформою з відкритим кодом для організації дистанційного навчання. Moodle підтримує модульну структуру плагінів, що забезпечує гнучке налаштування функцій - від завдань і форумів до аналітики успішності. Вона орієнтована на клієнт-серверну архітектуру та вимагає веб-середовища з підтримкою PHP, MySQL або PostgreSQL. Основна перевага - масштабованість і розгалужена спільнота, проте інтерфейс Moodle потребує додаткової адаптації для офлайн-роботи або десктопного використання.

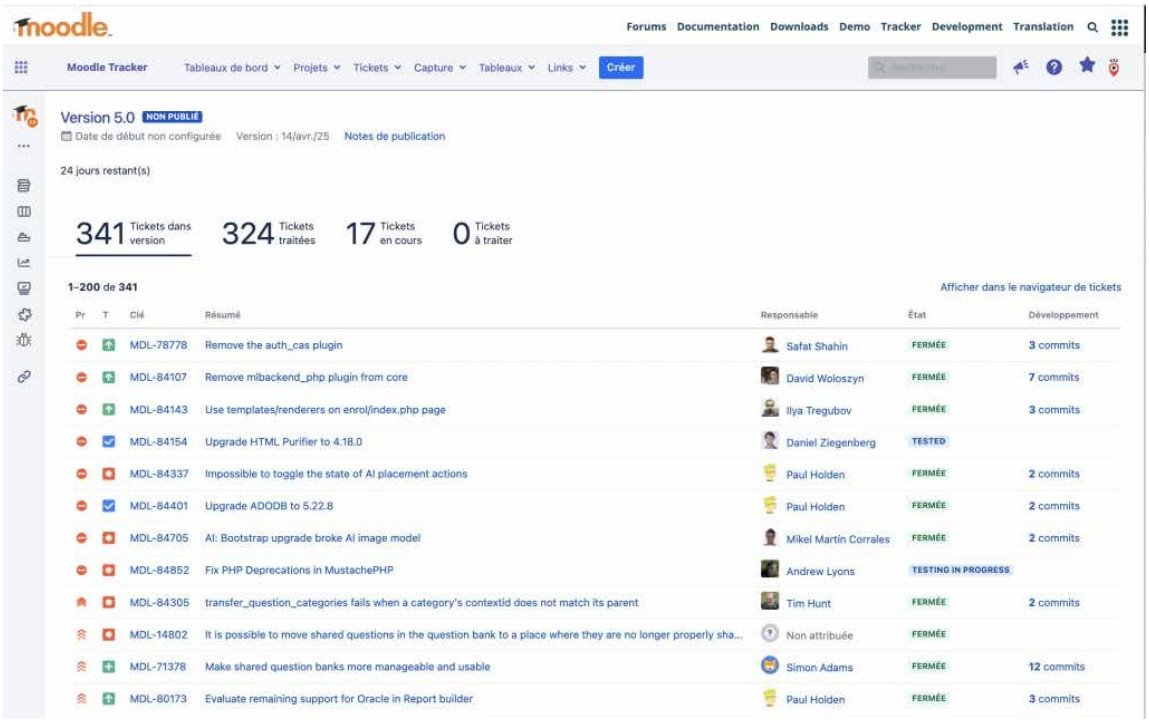


Рисунок 1.8 - Інтерфейс і панель управління задачами системи Moodle (версія 5.0, розробка в JIRA Tracker)

Наступне рішення - Google Classroom, що реалізує хмарну модель інтеграції між користувачами, контентом і засобами Google Workspace (Docs, Drive, Meet). На рис. 1.9 зображено типову панель курсів Google Classroom, де акцент зроблено на візуальній простоті та мобільності. Система орієнтована на швидку публікацію завдань, але має обмеження у функціях розширеної аналітики, що робить її менш придатною для наукових і корпоративних освітніх програм [2].

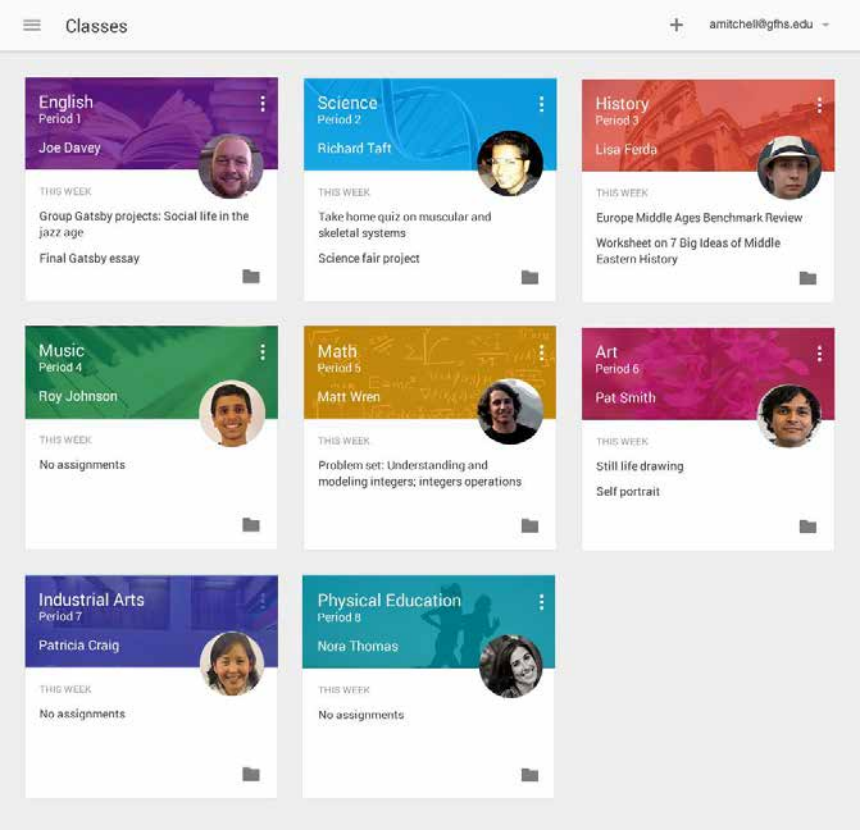


Рисунок 1.9 - Інтерфейс курсів у Google Classroom (приклад панелі викладача)

На рис. 1.10 представлено систему Canvas LMS, розроблену компанією Instructure. Вона характеризується сучасним REST-орієнтованим API, підтримкою SCORM-стандартів і розвиненими засобами створення тестів та адаптивних маршрутів. Canvas має потужний механізм інтеграції з аналітичними панелями успішності, проте її використання потребує постійного з’єднання з сервером і суттєвих ресурсів бази даних, що унеможливлює автономну роботу без мережевого доступу.

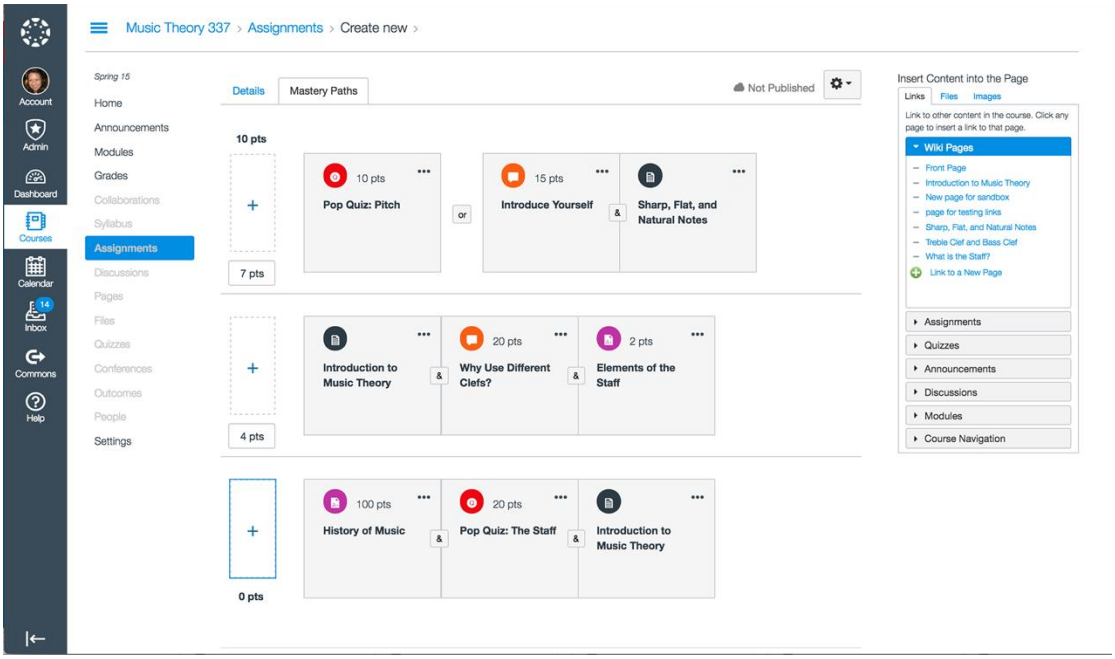


Рисунок 1.10 - Інтерфейс модулів і завдань у Canvas LMS

Система Blackboard Learn, подана на рис. 1.11, орієнтована на академічне середовище з акцентом на адміністративному контролі, аналітиці й комунікації між студентами та викладачами. Blackboard підтримує централізовану модель зберігання курсів і звітів, що полегшує управління великими освітніми установами. Основним недоліком залишається складність користувацького інтерфейсу та висока вартість корпоративної ліцензії [3].

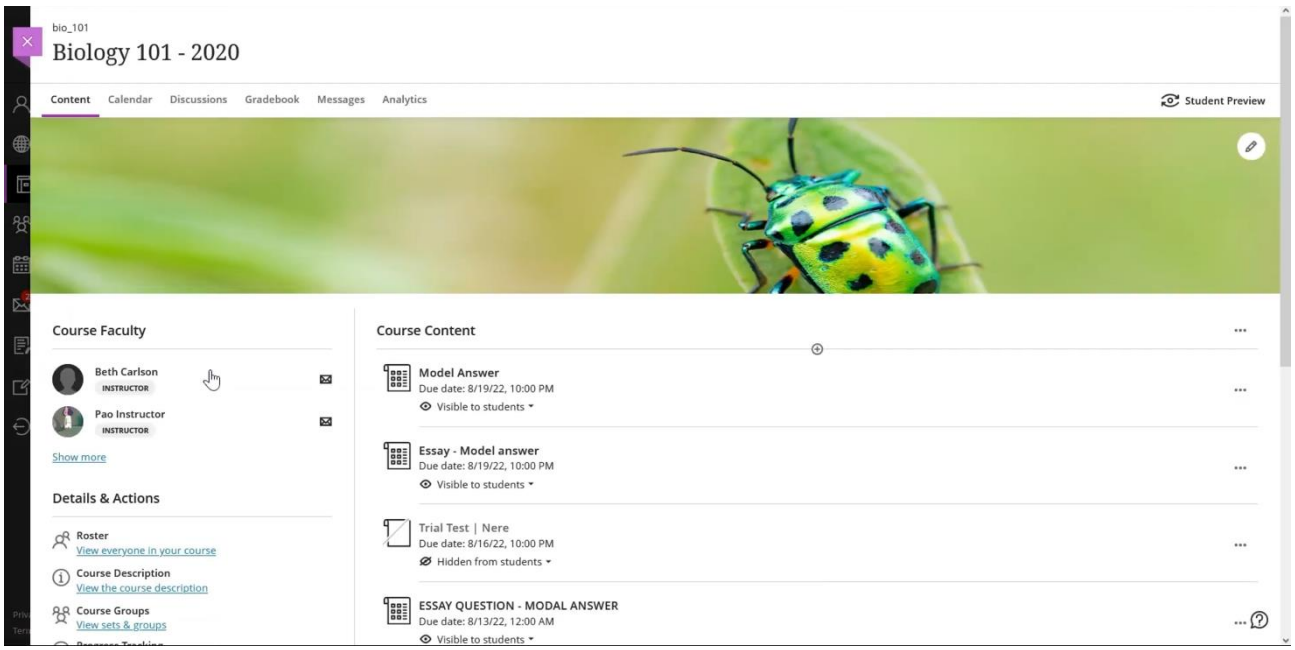


Рисунок 1.11 - Навчальний курс у Blackboard Learn з аналітичними віджетами викладача

Платформа Blend-ed Platform (рис. 1.12) є новітнім SaaS-рішенням, яке поєднує елементи гейміфікації, моніторинг активності, рекомендаційні механізми та персоналізовані аналітичні панелі. Її архітектура базується на мікросервісній моделі з підтримкою REST API, що забезпечує масштабування для корпоративних клієнтів. Система має розширену візуалізацію прогресу та інтеграцію з LMS-платформами, однак не допускає повного офлайн-режиму й вимагає стабільного підключення до хмарного середовища [4].

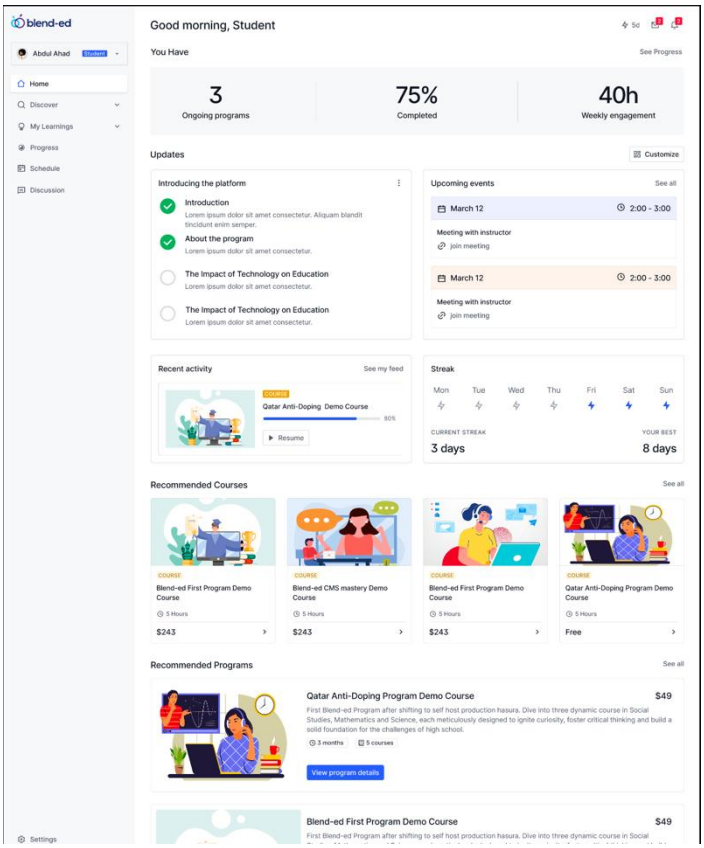


Рисунок 1.12 - Персоналізована аналітична панель користувача в Blend-ed Platform

Порівняльний аналіз основних рішень (табл. 1.4) показує, що більшість систем реалізує веб-архітектуру з орієнтацією на браузерну взаємодію. Натомість розроблювана нами система дистанційного електронного навчання створюється у середовищі Java Swing як автономне десктопне програмне забезпечення. Вона поєднує адаптивну модель навчання, аналітичний модуль оцінювання когнітивного коефіцієнта знань KiKi і механізм локальної бази даних SQLite для збереження прогресу без постійного з’єднання з сервером.

Таблиця 1.4

Порівняння існуючих систем дистанційного навчання та запропонованої Java Swing-системи

Система	Тип архітектури	Адаптивність	Офлайн-режим	Інтеграція з аналітикою	Призначення
Moodle	Веб/клієнт-сервер	Обмежена (через плагіни)	Ні	Так	Освітні установи
Google Classroom	Хмарна	Часткова	Ні	Обмежена	Школи, дистанційні курси
Canvas LMS	Веб/REST API	Так	Ні	Розширена	Університети, МООС
Blackboard Learn	Централізована	Так	Ні	Так	ВНЗ, корпоративне навчання
Blend-ed Platform	SaaS/мікросервісна	Так	Ні	Так	Корпоративні середовища
Java Swing System (розробка)	Десктоп/автономна	Так (через модуль оцінки K_i)	Так	Так (локальна аналітика)	Автономне навчання без сервера

Результати аналізу свідчать, що переважна більшість LMS орієнтована на хмарну або клієнт-серверну модель, яка потребує постійного з’єднання з мережею та використання браузера. У межах цього дослідження сформульовано альтернативний підхід - створення адаптивного десктопного ПЗ для дистанційного навчання, що реалізує інтелектуальні алгоритми оцінювання, рекомендацій та візуалізації прогресу у локальному середовищі Java Swing. Наукова новизна полягає у поєднанні методів адаптивного контент-менеджменту, локальної аналітики знань і модульної архітектури, що забезпечує повну автономність користувача без залежності від зовнішніх веб-сервісів.

1.4 Моделювання програмної системи

Моделювання предметної області системи дистанційного електронного навчання передбачає формалізацію основних учасників, процесів і взаємозв’язків, що забезпечують повний цикл навчання - від автентифікації користувача до формування аналітичних звітів. Для цього застосовано комплекс UML-діаграм: діаграму прецедентів, діаграму послідовності та діаграму активності, що відображають функціональну логіку системи в її динаміці та структурі.

На рис. 1.13 наведено діаграму прецедентів, яка визначає основних акторів системи - студента, викладача, адміністратора, зовнішній сервіс автентифікації (IdP/SSO) та відеоплатформу CDN. Вона описує сценарії використання, такі як «Вхід до системи», «Перегляд каталогу курсів», «Проходження тестів», «Публікація матеріалів» та «Керування користувачами». Зв’язки типу «include» та «extend» забезпечують повторне використання процесів перевірки MFA й умов зарахування студента. Така структура відображає розподіл ролей і підсистем - користувацької, адміністративної та сервісної, що відповідає принципам модульної архітектури LMS-рішень.

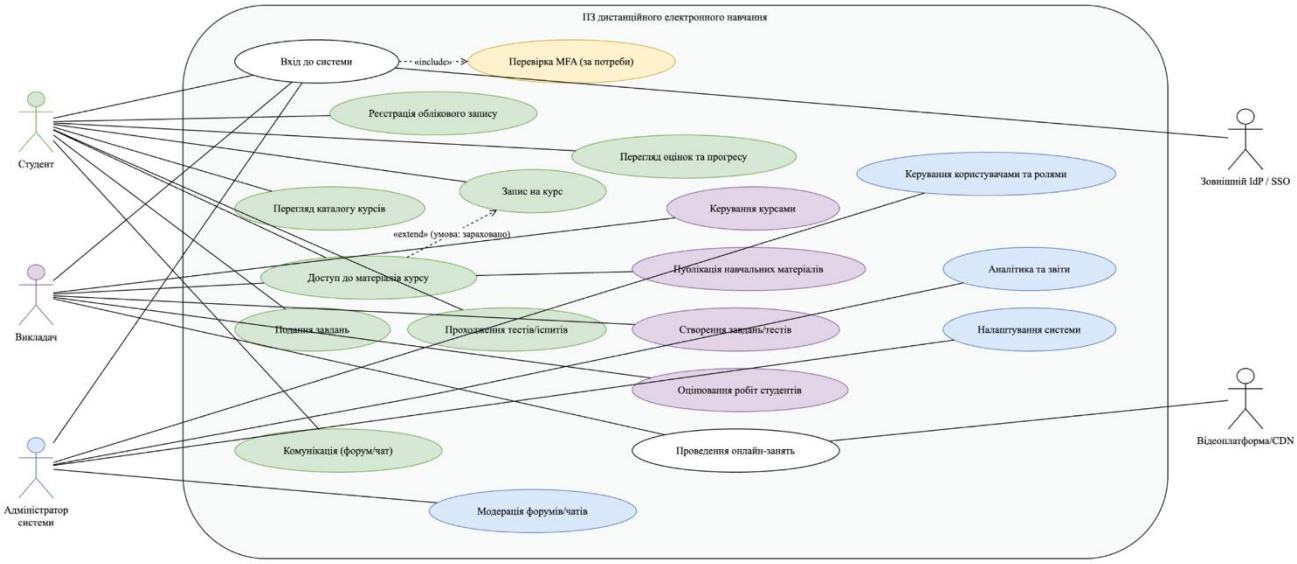


Рисунок 1.13 — Діаграма прецедентів програмного забезпечення дистанційного електронного навчання

Деталізацію взаємодії між компонентами системи представлено на діаграмі послідовності (рис. 1.14). Вона демонструє основний сценарій - процес автентифікації, запису на курс, отримання навчальних матеріалів та передачі результатів завдань. Послідовність повідомлень між веб-клієнтом, сервісом авторизації, модулем курсів, сховищем контенту та підсистемою оцінювання реалізується через REST-запити (POST, GET) із токеном JWT, що гарантує захищений обмін даними. На етапі оцінювання результати завдань автоматично передаються модулю аналітики, який формує зворотний зв'язок для користувача. Така модель забезпечує узгодженість потоків подій і логіку взаємодії між компонентами у середовищі Java-програми з клієнтсько-серверною структурою.

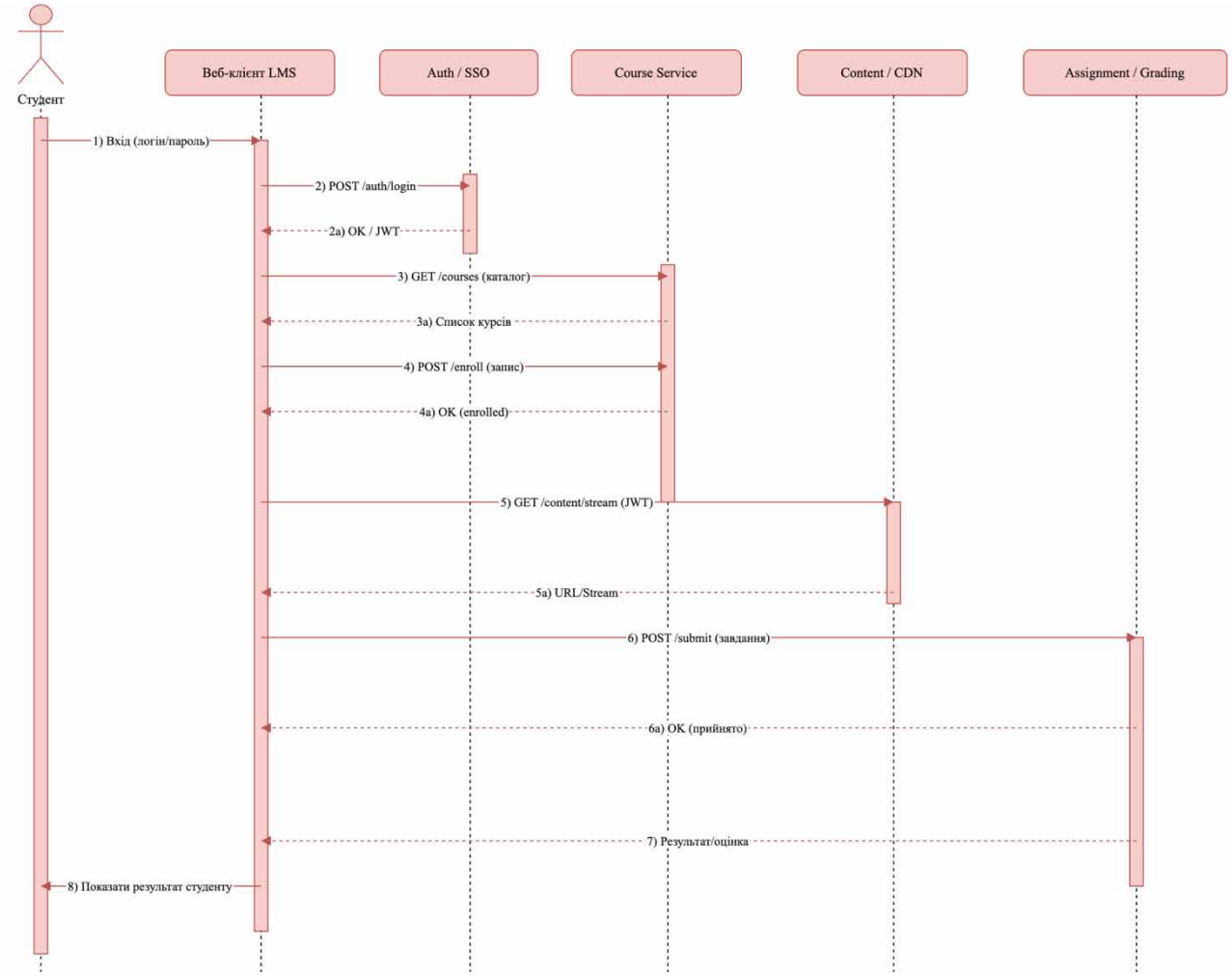


Рисунок 1.14 — Діаграма послідовності процесу взаємодії користувача з системою дистанційного навчання

Подальше узагальнення логіки процесів відображено на діаграмі активності (рис. 1.15). Вона моделює основні етапи життєвого циклу користувача: введення облікових даних, автентифікацію, запис на курс, перегляд матеріалів, виконання завдань, проходження тестів і формування рекомендацій. У діаграмі реалізовано три взаємопов’язані смуги (swimlane): «Студент», «Система LMS» та «Оцінювання / Аналітика». Передбачено логічні шлюзи (умови «автентифіковано?» та «тест пройдено?»), що визначають перехід між станами процесу. Така побудова дозволяє формалізувати поведінку системи й забезпечує прозорість при проектуванні алгоритмів у середовищі Java Swing.

Рисунок 1.15 — Діаграма активності взаємодії користувача із системою дистанційного навчання

Зведення результатів моделювання дозволяє визначити структурно-функціональні особливості системи:

- інтеграція користувачів через SSO або локальну реєстрацію;
- рольова взаємодія між студентом, викладачем і адміністратором;
- поділ процесів на функціональні підсистеми - автентифікації, управління курсами, контенту та аналітики;
- наявність механізмів зворотного зв’язку через модуль оцінювання результатів і формування рекомендацій.

Створена модель предметної області формує логічну основу для побудови програмного ядра системи дистанційного навчання. Її ключовими характеристиками є адаптивність, модульність і інтерактивність, що дозволяють реалізувати інтелектуальні алгоритми аналізу навчального прогресу у середовищі Java Swing без залучення веб-інфраструктури. Запропонована структура забезпечує гнучке масштабування, можливість офлайн-роботи та повну відповідність принципам архітектурного моделювання інформаційних систем.

1.5 Аналіз вимог захисту інформаційних систем

Аналіз вимог експертної системи дистанційного електронного навчання є ключовим етапом проєктування, що визначає цілі, функціональні можливості, технічні обмеження та критерії безпеки майбутнього програмного продукту. Вимоги було сформовано з урахуванням результатів моделювання предметної області (розділ 1.4) та особливостей адаптивного навчального середовища, орієнтованого на автономну роботу у середовищі Java Swing.

Функціональні вимоги відображають основні дії користувачів - студентів, викладачів та адміністратора системи. Вони охоплюють процеси реєстрації, проходження навчання, публікації матеріалів, аналітики та керування користувачами. На основі результатів аналізу діаграм прецедентів і активності

(див. рис. 1.13–1.15) сформовано узагальнений перелік функціональних вимог, поданий у таблиці 1.5.

Таблиця 1.5

Функціональні вимоги експертної системи дистанційного електронного навчання

№	Вимога	Опис	Роль користувача
1	Реєстрація та автентифікація	Забезпечення входу через локальну обліковку або IdP/SSO	Усі користувачі
2	Перегляд каталогу курсів	Доступ до переліку доступних курсів і фільтрація за темами	Студент
3	Запис на курс та перегляд матеріалів	Отримання доступу до навчальних модулів і тестів	Студент
4	Публікація навчальних матеріалів	Завантаження відео, документів, тестів	Викладач
5	Оцінювання робіт студентів	Перевірка завдань вручну або автоматично	Викладач
6	Формування аналітичних звітів	Генерація звітів щодо успішності та активності	Адміністратор, викладач
7	Керування користувачами та ролями	Додавання, блокування або зміна ролей користувачів	Адміністратор
8	Комунікація (чат/форум)	Обмін повідомленнями в межах курсу	Студент, викладач
9	Проведення онлайн-занять	Інтеграція з відеоплатформою (CDN)	Викладач

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її стабільність, масштабованість, продуктивність та зручність користування. Оскільки система розробляється для автономного використання без веб-сервера, головний акцент зроблено на швидкодії інтерфейсу, оптимізації локальної бази даних SQLite та сумісності з різними ОС (Windows, Linux, macOS). Узагальнення нефункціональних вимог наведено у таблиці 1.6.

Таблиця 1.6

Нефункціональні вимоги експертної системи дистанційного навчання

№	Категорія	Вимога	Показник / Критерій
1	Продуктивність	Час відгуку при навігації між модулями не більше 200 мс	≤ 0.2 с
2	Надійність	Збереження даних після непередбаченого завершення роботи	99.9 %
3	Масштабованість	Підтримка не менше 10 000 локальних записів у БД	≥ 10^4 записів
4	Сумісність	Працездатність у середовищах Windows/Linux/macOS	Повна підтримка
5	Зручність інтерфейсу	Відповідність GUI принципам UX-дизайну (мінімум 60 FPS)	Висока
6	Розширюваність	Можливість підключення нових модулів через API	Так

Окрему категорію становлять вимоги до безпеки, які є критично важливими для систем, що містять персональні дані користувачів, навчальні матеріали та аналітичні звіти. Враховуючи, що система підтримує інтеграцію із зовнішнім SSO та локальні облікові записи, особливу увагу приділено механізмам шифрування, автентифікації та контролю доступу. Основні вимоги до безпеки подано у таблиці 1.7.

Таблиця 1.7

Вимоги до безпеки експертної системи дистанційного навчання

№	Вимога	Опис / Механізм реалізації
1	Захист автентифікації	Двофакторна перевірка MFA при вході (за потреби)
2	Шифрування даних	AES-256 для збереження локальних записів у БД
3	Розмежування доступу	Механізм RBAC для ролей «студент», «викладач», «адміністратор»
4	Логи безпеки	Реєстрація подій доступу, помилок і входів у систему
5	Цілісність даних	Контроль контрольних сум і перевірка підписів модулів
6	Захист аналітичних звітів	Шифрування PDF/CSV-файлів з аналітикою навчання

Результати проведеного аналізу вимог дозволяють сформувати узагальнену архітектурну концепцію системи, що базується на принципах модульності, адаптивності та безпеки. На відміну від типових LMS, які функціонують у веб-середовищі, запропонована експертна система має десктопну архітектуру, що забезпечує роботу без підключення до мережі, локальне зберігання даних та швидкий відгук інтерфейсу. В основі її інтелектуальної логіки - алгоритм оцінювання знань на основі когнітивного коефіцієнта K_i динамічний вибір навчального контенту, що реалізує персоналізований підхід до навчання.

Отже, проведений аналіз вимог визначає рамкову специфікацію функціональних, нефункціональних і безпекових параметрів системи, що забезпечують її ефективну роботу, надійність і наукову відповідність сучасним тенденціям розроблення інтелектуальних навчальних платформ.

1.6 Постановка завдання

На основі проведеного аналізу предметної області, вимог і моделювання процесів системи дистанційного електронного навчання (розділи 1.3–1.5) було сформульовано постановку завдання, що визначає мету, структуру та взаємозв’язки між основними компонентами майбутньої експертної системи. Завдання полягає у створенні інтелектуального програмного забезпечення для дистанційного навчання, яке функціонує в автономному середовищі Java Swing, забезпечує персоналізований підхід до користувача, аналітику навчального прогресу та захищене управління навчальними даними.

Система має реалізовувати повний цикл взаємодії користувача з навчальним контентом - від автентифікації до формування звітів про успішність. Для цього необхідно спроєктувати архітектуру, що включає три основні підсистеми:

- підсистема користувача, що забезпечує роботу студентів і викладачів у середовищі локального клієнта з інтуїтивним графічним інтерфейсом;

- Підсистема управління даними, яка відповідає за оброблення, зберігання та оновлення записів у локальній базі даних SQLite;

- Підсистема аналітики та експертних рішень, що здійснює оцінювання рівня знань за допомогою інтелектуального алгоритму на основі когнітивного коефіцієнта знань Ki і формує рекомендації для користувача.

Вхідні дані системи:

- облікові дані користувачів (логін, пароль, роль, ідентифікатор SSO при наявності).

- Метадані навчальних курсів (назва, опис, структура модулів, тривалість).

- Навчальні матеріали (текстові, мультимедійні та тестові ресурси).

- Результати тестувань, оцінки, спроби проходження.

- Поведінкові показники активності (кількість входів, час перегляду, завершені завдання).

- Параметри конфігурації системи (налаштування безпеки, ролей, інтерфейсу).

Вихідні дані системи:

- аналітичні звіти успішності студентів (інтерактивні графіки, PDF/CSV-звіти).

- Рекомендації щодо повторного навчання або вибору наступного модуля.

- Зведені статистичні показники відвідуваності, активності та результатів.

- Автоматично розрахований коефіцієнт знань Ki для кожного студента.

- Збережені у базі даних дані про пройдені курси, спроби та оцінки.

- Узагальнено процес можна подати як інформаційну схему:

Система має приймати первинні дані про користувачів і навчальний контент, обробляти їх через модуль аналітики та видавати результати у вигляді звітів і рекомендацій, що дозволяють адаптувати подальший процес навчання.

Основна мета постановки завдання полягає у створенні програмного комплексу, який забезпечить:

- адаптивну взаємодію користувача з навчальним матеріалом;

- обчислення індивідуального коефіцієнта знань Ki=f(Pi,Li,Ti), де Pi - результат тестування, Li - рівень засвоєння, Ti - час навчання;

- автоматичне формування рекомендацій на основі аналітичних висновків;

- автономність функціонування без підключення до мережі;

- захист даних користувачів через механізми шифрування AES-256 і контроль доступу RBAC.

Результатом реалізації завдання стане експертна система дистанційного навчання, здатна самостійно аналізувати успішність студентів, прогнозувати потребу в повторному навчанні та формувати рекомендаційні звіти для викладачів і адміністратора. Це дозволить підвищити ефективність освітнього процесу, мінімізувати ручне оцінювання та забезпечити інтелектуальну підтримку навчання в офлайн-середовищі.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних системи дистанційного навчання

Логічна модель даних експертної інформаційної системи ідентифікації рослин формує формалізоване подання доменної інформації у вигляді взаємопов’язаних нормалізованих сутностей, що безпосередньо відображають вимоги до зберігання зображень, ботанічних характеристик та результатів роботи алгоритмів глибинного навчання. На рис. 2.1 подано ER-діаграму логічної моделі даних, у якій виділено шість ключових сутностей: користувачі системи, таксономічний довідник видів рослин, набори даних (PlantVillage та власні колекції), окремі зразки зображень, експерименти з навчання моделей та результати класифікації.

Така декомпозиція отримана шляхом послідовної нормалізації концептуальної моделі до третьої нормальної форми, що дозволяє уникнути надмірності зберігання даних (дублювання ознак видів у записах зображень, повторення параметрів експериментів у прогнозах тощо) та мінімізує аномалії вставки, оновлення й видалення. Рознесення інформації про зображення, види та експерименти в окремі сутності із чітко визначеними первинними та зовнішніми ключами забезпечує трасованість повного ланцюга «користувач – зразок – модель – прогноз» і дає змогу відтворювати результати класифікації для наукового аналізу, порівняння архітектур CNN/ResNet/EfficientNet/Vision Transformer та побудови агрегованої статистики.

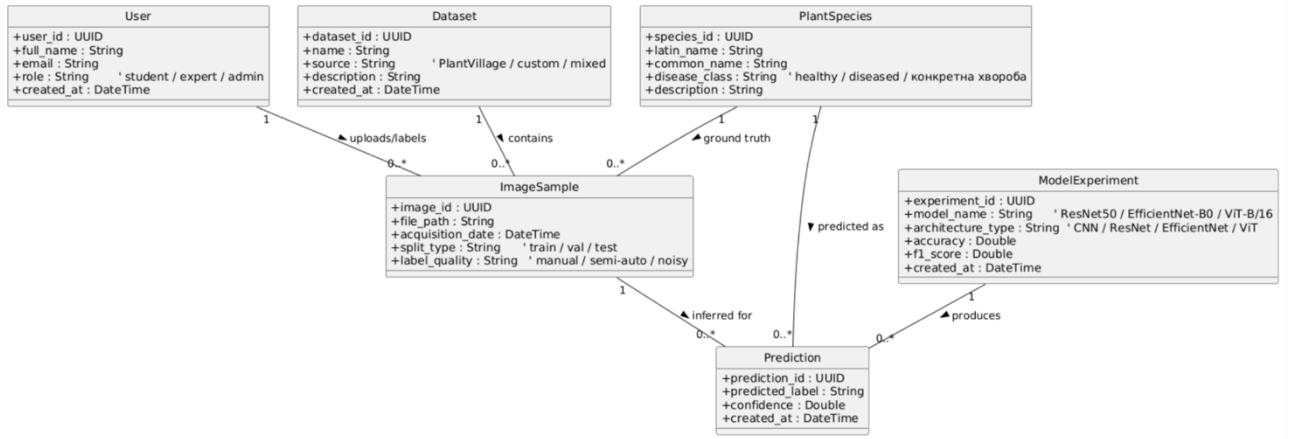


Рисунок 2.1 – Логічна модель даних експертної інформаційної системи ідентифікації рослин

Узагальнені характеристики сутностей логічної моделі подано в таблиці 2.1, де відображено їх роль у системі та базові ключові атрибути, що реалізуються на рівні реляційної БД. Завдяки такому поділу забезпечується незалежність ботанічного довідника від конкретних наборів даних, можливість повторного використання зразків зображень у різних експериментах, а також відокремлення результатів класифікації від параметрів навчання моделей, що є критично важливим для підтримки реплікованості експериментів та подальшого розширення підсистеми аналітики.

Таблиця 2.1

Основні сутності логічної моделі даних експертної системи

Сутність	Первинний ключ	Коротке призначення
User	user_id	Зберігання облікових записів користувачів та їх ролей у системі
PlantSpecies	species_id	Таксономічний довідник видів і хвороб рослин
Dataset	dataset_id	Опис наборів даних (PlantVillage, власні колекції тощо)
ImageSample	image_id	Збереження окремих зображень та їх належності до видів і датасетів
ModelExperiment	experiment_id	Метадані експериментів навчання моделей та їх інтегральні метрики
Prediction	prediction_id	Результати класифікації з посиланням на зразок, модель і вид

Узгоджена логічна модель даних, відображена на ER-діаграмі та формалізована у вигляді набору сутностей таблиці 2.1, створює основу для фізичного проєктування реляційної бази даних системи, вибору індексів і механізмів забезпечення цілісності, а також дозволяє гнучко масштабувати схему за рахунок додавання нових типів експериментів, розширення таксономічного довідника або підключення альтернативних наборів даних без руйнування існуючих залежностей і без втрати узгодженості історичних результатів класифікації.

2.2 Діаграма класів і кооперації системи e-learning

Логічна структура програмного забезпечення системи дистанційного електронного навчання формується через формалізацію взаємозв’язків між ключовими компонентами доменної моделі, що реалізують повний цикл навчального процесу: автентифікація, керування курсами, проведення тестування, обчислення когнітивного коефіцієнта знань Ki та формування аналітичних звітів. На рис. 2.2 наведено діаграму класів, побудовану відповідно до принципів об’єктно-орієнтованого моделювання, інкапсуляції та структурної нормалізації доменних об’єктів. Вона не лише відображає статичну структуру системи, але й демонструє, як класові сутності узгоджено взаємодіють із логічною моделлю даних, викладеною у підрозділі 2.1, забезпечуючи несуперечливість між інформаційною та поведінковою складовими архітектури.

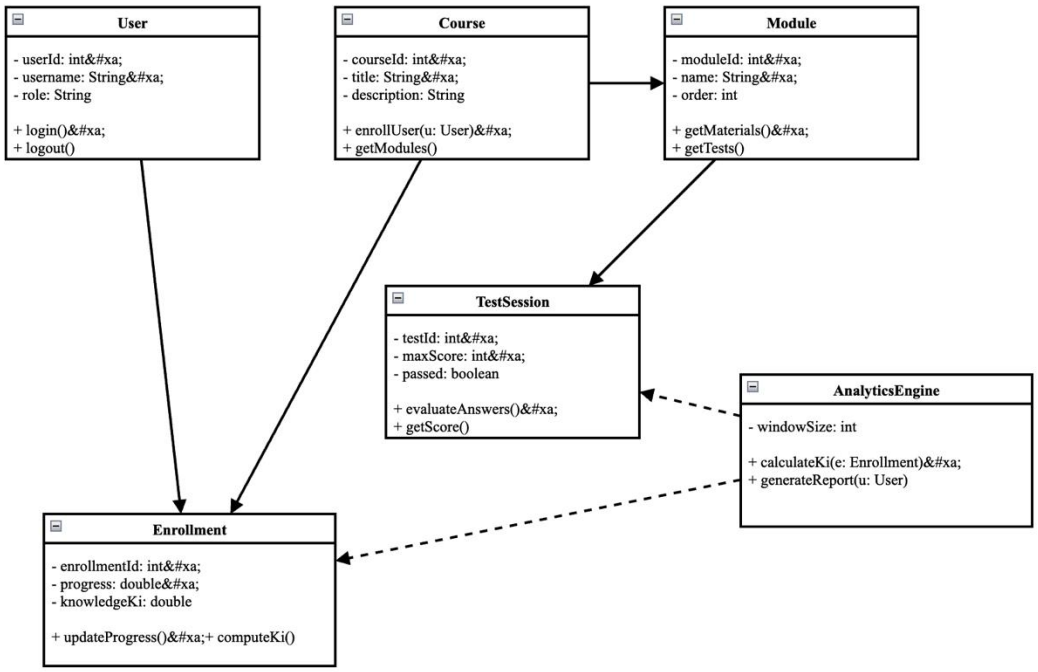


Рисунок 2.2 – UML-діаграма класів підсистеми дистанційного електронного навчання

Клас User реалізує уніфіковану модель користувача (студента або викладача), зберігаючи атрибути ролі та автентифікаційні методи. Клас Course інкапсулює параметри навчальних курсів і є центральною сутністю, що агрегує модулі (Module) та дозволяє виконувати операції запису користувачів. Елемент Enrollment забезпечує нормалізований зв’язок «багато-до-багатьох» між користувачами та курсами, усуваючи дублювання атрибутів і дозволяючи зберігати параметри прогресу, включаючи когнітивний показник Ki. Компонент TestSessionінкапсулює логіку оброблення тестових спроб, а AnalyticsEngine виконує алгоритмічний аналіз даних, забезпечуючи адаптивність навчання на рівні окремого студента.

Для відображення поведінкової взаємодії класів застосовано кооперації, які визначають послідовність обміну повідомленнями між об’єктами в конкретних сценаріях. На рис. 2.3 подано кооперацію для процесу автентифікації та відображення каталогу курсів, де LoginForm комунікує з AuthService, отримує підтвердження автентичності та ініціює завантаження навчальних матеріалів через CourseCatalog.

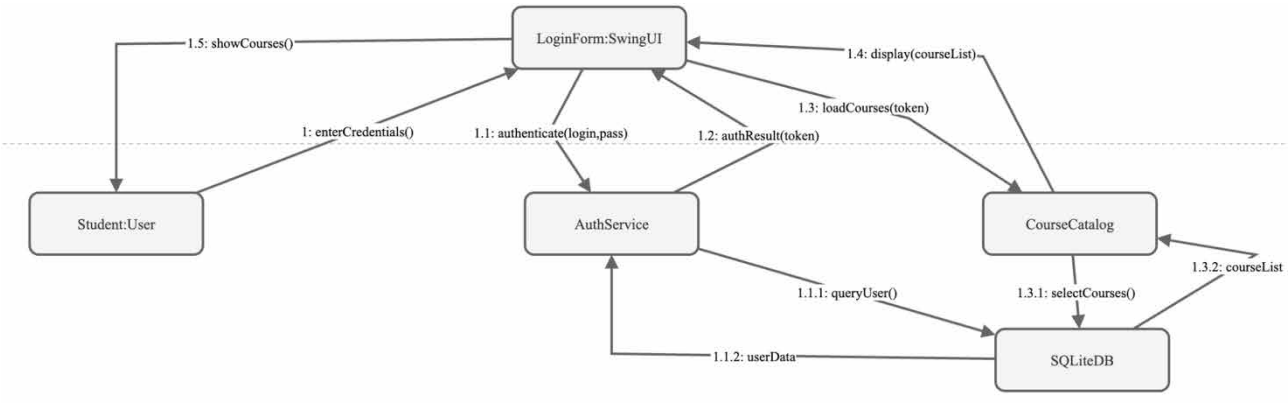


Рисунок 2.3 – Кооперація «Автентифікація та завантаження курсу»

На рис. 2.4 відтворено кооперацію проведення тестування, яка демонструє взаємодію між CourseUI, TestEngine, AnalyticsEngine та SQLiteDB. Така модель дозволяє чітко відокремити бізнес-логіку тестування, механізми оцінювання відповідей та модуль аналітики, що обчислює значення Ki на основі результатів студента. Важливо, що структура кооперації відповідає вимогам до модульності та забезпечує можливість незалежного удосконалення аналітичного блоку без зміни інших компонентів системи.

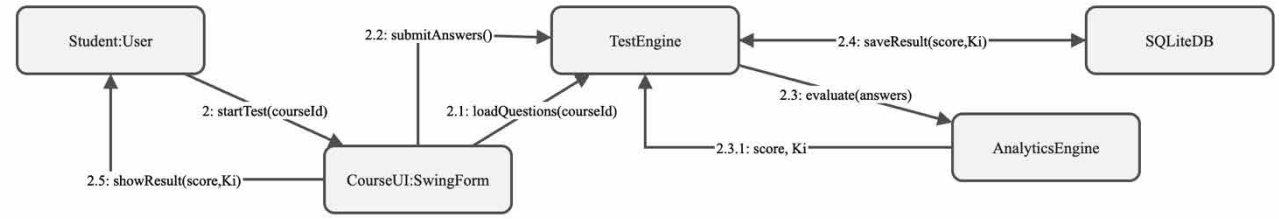


Рисунок 2.4 – Кооперація «Проведення тестування та розрахунок коефіцієнта знань Ki»

На рис. 2.5 наведено кооперацію роботи викладача з аналітичним модулем, де Dashboard, AnalyticsEngine та ReportGenerator реалізують циклічну взаємодію для формування звітів про успішність студентів. Така поведінкова модель повністю відповідає вимогам до експертної системи, орієнтованої на автономну роботу у середовищі Java Swing, та забезпечує формування PDF/CSV-звітів на основі локально збережених даних.

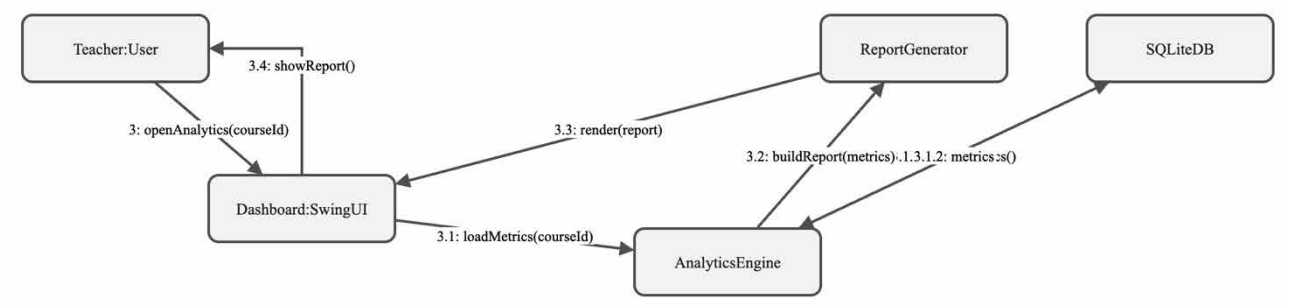


Рисунок 2.5 – Кооперація «Аналітична панель викладача та формування звітів»

У таблиці 2.2 подано узагальнену характеристику ключових класів, що входять до програмної архітектури. Представлені атрибути та функції відображають нормалізовану структуру інформаційних одиниць, узгоджену з логічною моделлю даних, та забезпечують мінімальну зв’язаність класів при максимальній функціональній декомпозиції.

Таблиця 2.2

Основні класи та їх функціональне призначення

Клас	Основні атрибути	Призначення
User	userId, username, role	Зберігання облікових даних, автентифікація
Course	courseId, title, description	Логічний контейнер навчальних модулів
Module	moduleId, name, order	Структурна одиниця курсу, містить матеріали і тести
Enrollment	enrollmentId, progress, knowledgeKi	Відстеження навчального прогресу студента
TestSession	testId, maxScore, passed	Реалізація тестування й оброблення відповідей
AnalyticsEngine	windowSize	Обчислення когнітивного коефіцієнта Ki та генерація звітів

У результаті узагальнення можна стверджувати, що діаграма класів і кооперацій формує основу поведінкової логіки системи й забезпечує структурну узгодженість усіх компонентів. Вона демонструє модульність, низьку зв’язаність і високу внутрішню когерентність підсистем, що критично важливо для адаптивного навчального середовища, орієнтованого на локальну роботу без серверної інфраструктури.

2.3 Архітектурне подання системи у вигляді діаграми компонентів

Архітектурне подання експертної системи дистанційного електронного навчання базується на модульному принципі побудови, який забезпечує незалежність функціональних блоків, спрощує модернізацію та підтримує можливість автономного виконання без серверної інфраструктури. На рис. 2.6 наведено діаграму компонентів, яка відображає структурну взаємодію між клієнтським інтерфейсом, підсистемою автентифікації, менеджером курсів і модулів, аналітичним ядром з обчислення коефіцієнта знань Ki, а також підсистемою збереження даних на основі локальної бази SQLite. Обрана архітектурна декомпозиція повністю відповідає принципам інкапсуляції й низької зв’язаності, що дозволяє розподілити відповідальності між компонентами та мінімізує міжмодульні залежності.

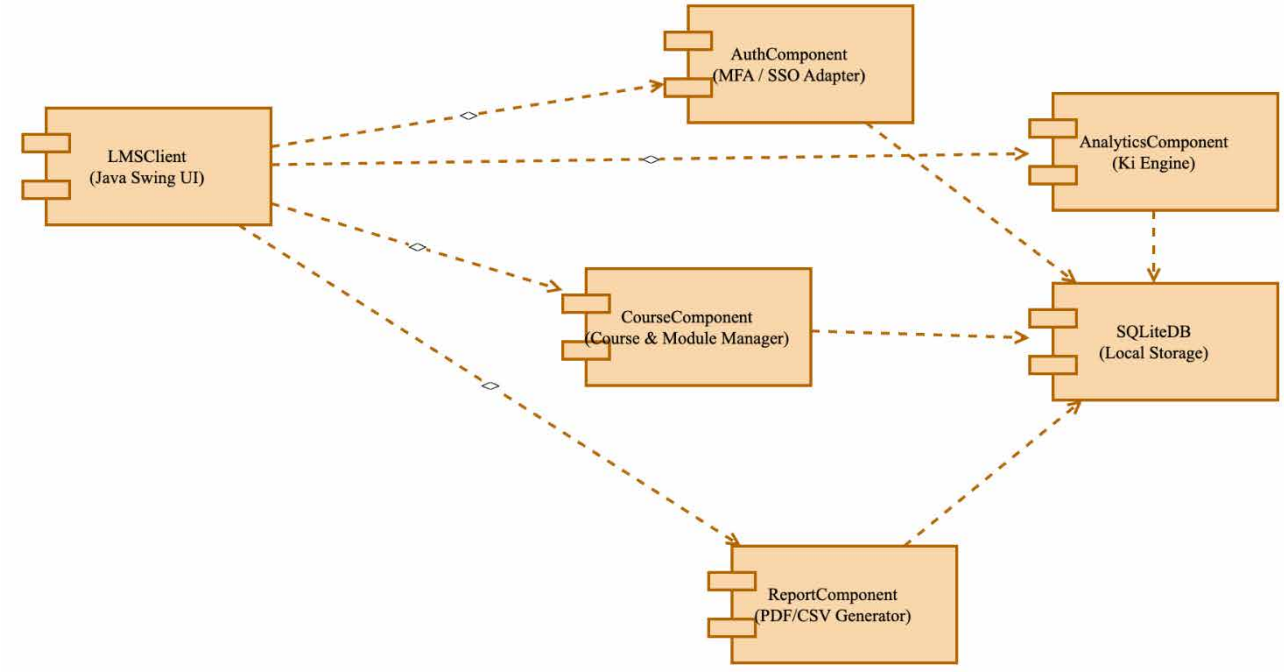


Рисунок 2.6 – Діаграма компонентів експертної системи дистанційного електронного навчання

Як показано на рис. 2.6, основним елементом є компонент LMSClient, реалізований засобами Java Swing, який формує графічний інтерфейс користувача та ініціює виклики до інших підсистем. Компонент AuthComponentзабезпечує підтримку локальної автентифікації, адаптера MFA/SSO та логіку керування доступом згідно з ролями користувачів. Модуль

CourseComponent відповідає за завантаження, структурування та керування навчальними курсами й модулями, а також реалізує стандартизований API для інших частин системи. Аналітичний компонент AnalyticsComponent виконує обчислення коефіцієнта знань Ki, аналіз навчальних траєкторій та формування агрегованих метрик успішності, що є ядром інтелектуальної складової системи. Компонент ReportComponentреалізуЄ генерацію звітів у форматах PDF/CSV, використовуючи агреговані аналітичні дані. Локальна база SQLiteDB виступає універсальним сховищем структурованої інформації – від облікових записів і курсів до результатів тестування та значень індивідуальних коефіцієнтів знань.

У таблиці 2.3 подано узагальнену характеристику компонентів системи, їх функціональні зони відповідальності та ключові дані, з якими вони працюють. Такий поділ забезпечує незалежне оновлення аналітичних і адміністративних функцій, відповідність принципам модульної архітектури та можливість масштабування системи шляхом додавання нових підсистем (наприклад, чат-сервісу, адаптивної рекомендаторної моделі, інтеграції з ERP або зовнішніми освітніми платформами).

Таблиця 2.3

Основні компоненти системи та їх функціональні ролі

Компонент	Функціональне призначення	Основні дані / API
LMSClient	Взаємодія з користувачем, навігація, відображення контенту	UI-події, запити до менеджерів
AuthComponent	Автентифікація, SSO/MFA, контроль доступу	Облікові записи, токени доступу
CourseComponent	Управління курсами, модулями, тестами	Метадані курсів і модулів
AnalyticsComponent	Обчислення *Ki*, аналіз навчального прогресу	Дані спроб тестування, історія навчання
ReportComponent	Генерація PDF/CSV звітів	Аналітичні метрики, шаблони звітів
SQLiteDB	Локальне зберігання результатів, курсів, користувачів	SQL-таблиці, CRUD-операції

Узагальнюючи, діаграма компонентів дозволяє оцінити архітектурну зрілість системи та виокремити ключові механізми забезпечення її адаптивності: чітку модульність, стандартизовані інтерфейси між компонентами, автономне локальне зберігання даних і незалежний аналітичний модуль. Такий підхід сприяє гнучкому масштабуванню, спрощує тестування та забезпечує основу для безпечного розширення функціональності системи в рамках загальної архітектурної моделі дистанційного електронного навчання.

2.4 Пакетна структуризація програмної архітектури системи

Пакетна декомпозиція експертної системи дистанційного електронного навчання забезпечує формалізоване розділення програмної логіки на ізольовані підсистеми, що спрощує підтримку коду, модульне тестування та поступове масштабування функціональності. На рис. 2.7 подано діаграму пакетів, у якій структуру системи поділено на логічні домени відповідно до функціональних зон - ядро програми, інтерфейс користувача, модулі курсового менеджменту, підсистему аналітики Ki, модуль автентифікації та пакет збереження даних. Такий поділ є необхідним для побудови автономної Java Swing-системи, яка об’єднує офлайн-взаємодію, аналітичні алгоритми та локальну персистентність у єдиний структурований простір.

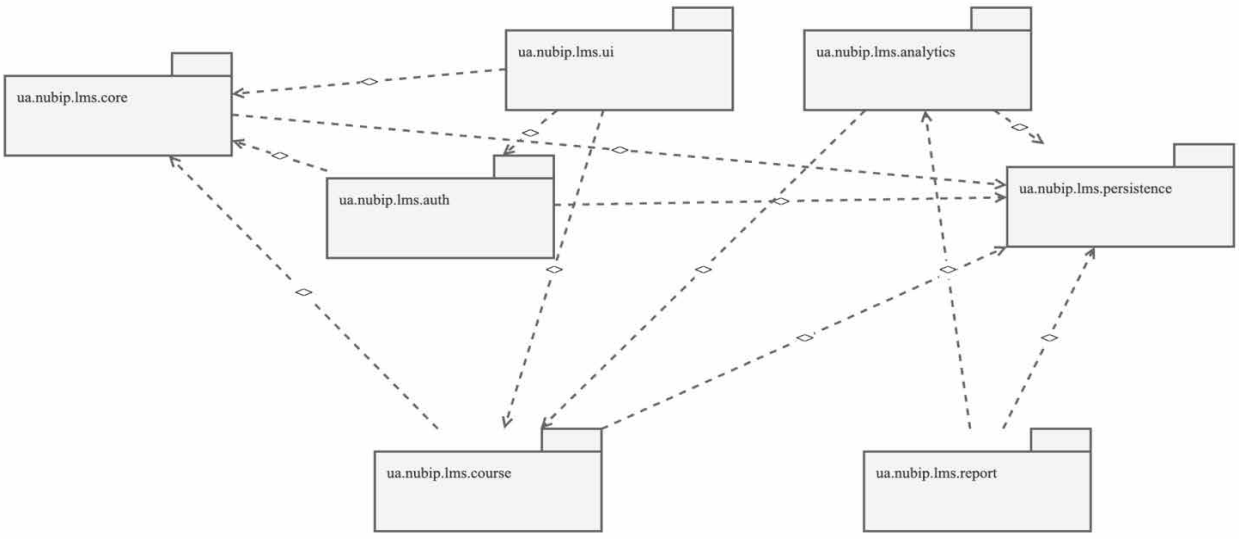


Рисунок 2.7 – Пакетна структура експертної системи дистанційного електронного навчання

Структура пакетів узгоджується з вимогами модульності, визначеними на етапах аналізу вимог та моделювання. Зокрема, пакет ua.nubip.lms.core виконує роль системного ядра, що координує загальні механізми роботи, тоді як ua.nubip.lms.ui ізольовано відповідає за графічний інтерфейс. Пакет ua.nubip.lms.course містить засоби управління курсами та навчальними модулями, а ua.nubip.lms.analytics реалізує аналітичні алгоритми та обчислення когнітивного коефіцієнта знань Ki, інтегруючи дані про прогрес навчання. Блок ua.nubip.lms.auth забезпечує незалежність механізмів автентифікації, що важливо для безпеки та підтримки SSO/MFA-розширень, а ua.nubip.lms.persistence відповідає за роботу з локальною базою SQLite. Пакет ua.nubip.lms.report забезпечує формування та експорт звітів успішності. Така структура пакетів дозволяє мінімізувати перехресні залежності, відокремити бізнес-логіку від інтерфейсу та забезпечити передбачувану еволюцію системи за рахунок можливості розширення окремих модулів без рефакторингу всієї програми.

У таблиці 2.4 подано узагальнення функціонального призначення пакетів, що входять до архітектури системи. Представлена структуризація чітко відображає розподіл відповідальностей між підсистемами, підвищує внутрішню когерентність і забезпечує відповідність принципам побудови складних автономних навчальних платформ.

Таблиця 2.4

Пакети системи та їх функціональні області

Пакет	Функціональна роль
ua.nubip.lms.core	Базова логіка системи, координація роботи модулів
ua.nubip.lms.ui	Графічний інтерфейс користувача, навігація, взаємодія з компонентами
ua.nubip.lms.auth	Автентифікація, MFA/SSO, механізми контролю доступу
ua.nubip.lms.course	Керування курсами, навчальними модулями та тестовими елементами
ua.nubip.lms.analytics	Алгоритми аналізу, обчислення коефіцієнта знань *Ki*, агрегування метрик
ua.nubip.lms.persistence	Локальне зберігання даних, робота з SQLite, транзакційний доступ
ua.nubip.lms.report	Генерація PDF/CSV-звітів, формування аналітичних представлень

Узагальнення результатів свідчить, що пакетна структура не лише впорядковує програмний код, але й формує архітектурну основу для стійкої роботи системи в автономному режимі, забезпечуючи незалежність модулів, структурованість взаємозв’язків і можливість інтегрування нових функціональних блоків зі збереженням цілісності програмного середовища.

2.5 Висновки до другого розділу

У другому розділі було сформовано повний комплекс архітектурно-структурних моделей, які визначають логічну, інформаційну та функціональну організацію експертної системи дистанційного електронного навчання. На основі аналізу доменних вимог побудовано нормалізовану логічну модель даних у вигляді ER-діаграми, що забезпечує узгодженість між інформаційними об’єктами, мінімізує дублювання даних і створює формальний базис для подальшої реалізації локального сховища на SQLite. Розроблена діаграма класів відображає структурну декомпозицію програмних об’єктів, демонструє чіткий розподіл відповідальностей між підсистемами та забезпечує цілісність поведінкової логіки через побудовані кооперації ключових сценаріїв — автентифікації, роботи з курсами, тестування та аналітичного оцінювання знань за показником *Ki*. Діаграма компонентів системи підтвердила модульний характер архітектури, її відповідність принципам інкапсуляції, низької зв’язаності та можливості автономного функціонування в офлайн-середовищі Java Swing. Пакетна структуризація інтегрувала отримані моделі в єдину логічну архітектуру, забезпечивши масштабованість, передбачуваність еволюції програмного коду та готовність системи до розширення (аналітичних модулів, засобів звітності та механізмів автентифікації). Сукупність побудованих моделей формує комплексне бачення архітектури системи, яке дозволяє перейти до етапу алгоритмізації та програмної реалізації з повним розумінням структури, функціональних залежностей і механізмів оброблення даних.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Для реалізації експертної системи дистанційного електронного навчання було обґрунтовано вибір технологічних інструментів на основі функціональних, архітектурних та експлуатаційних вимог. У процесі аналізу визначено, що система має забезпечувати автономність роботи, підтримку локального зберігання даних, ефективний UI-рівень, а також можливість виконання аналітичних операцій без зовнішніх залежностей. Узагальнені характеристики технологій, що відповідають цим вимогам, наведено в таблиці 3.1.

Таблиця 3.1

Основні технології та інструментальні засоби реалізації системи

Технологія / Інструмент	Обґрунтування вибору	Роль у системі
Java SE 17+	Стабільність, кросплатформеність, високий рівень безпеки та підтримка ООП-моделей	Базова мова програмування, реалізація бізнес-логіки
Java Swing	Відсутність залежності від веб-серверів, можливість створення автономного UI	Графічний інтерфейс користувача, панелі курсів, тестів, аналітики
SQLite	Легка, швидка, ACID-сумісна локальна СУБД без сервера	Зберігання результатів тестування, курсів, користувачів, значень *Ki*
iText / OpenPDF	Підтримка генерації PDF у локальному середовищі	Формування PDF-звітів успішності
Apache Commons / Java Collections	Універсальні структури даних та інструменти оброблення інформації	Алгоритми аналізу, агрегування метрик, обчислення *Ki*
Git / GitHub	Контроль версій та командна взаємодія	Історія змін, розвиток проєкту
IntelliJ IDEA / Eclipse	Повний набір інструментів налагодження та рефакторингу	Середовище розроблення

Проведений аналіз підтвердив, що наведений набір технологій формує збалансований інтегрований стек, який забезпечує стабільну роботу системи в автономному режимі без зовнішніх серверних сервісів. Вибрані інструменти

гарантують підтримку аналітичної складової, високу швидкодію при роботі з локальною базою даних та можливість подальшого масштабування, що є критично важливим для адаптивних освітніх платформ.

3.2 Інформаційна база системи

Інформаційна база експертної системи дистанційного електронного навчання формує цілісну структуровану основу, що забезпечує накопичення, інтеграцію та аналітичну інтерпретацію даних про навчальний процес. Її побудова орієнтована не лише на підтримку традиційних функцій електронних навчальних систем (зберігання результатів, курсів, груп), але й на впровадження інтелектуальних механізмів оцінювання знань, класифікації успішності та виявлення закономірностей навчальної поведінки студентів. Особлива наукова новизна полягає у використанні багатовимірної моделі представлення даних, що поєднує аналітичні факти успішності, кластеризаційні профілі, ймовірнісні оцінки успішності курсів, а також у застосуванні гнучкої OLAP-структури, орієнтованої на подальшу інтеграцію інтелектуальних методів.

На рис. 3.1 наведено фрагмент зіркоподібної схеми (star-schema), яка є базовою моделлю інформаційного сховища системи. Фактична таблиця Results_Fact акумулює агреговані результати навчальної активності, тоді як розмірні таблиці (Course_Dim, Group_Dim, Date_Dim) забезпечують семантичну деталізацію даних, необхідну для побудови OLAP-запитів, прогнозної аналітики та класифікаційних моделей.

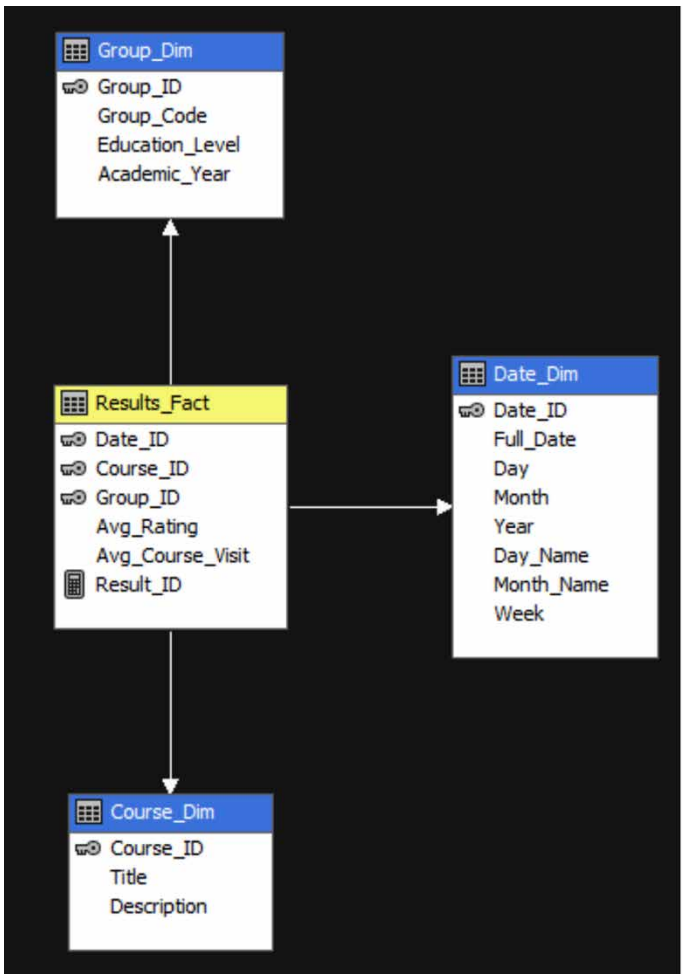
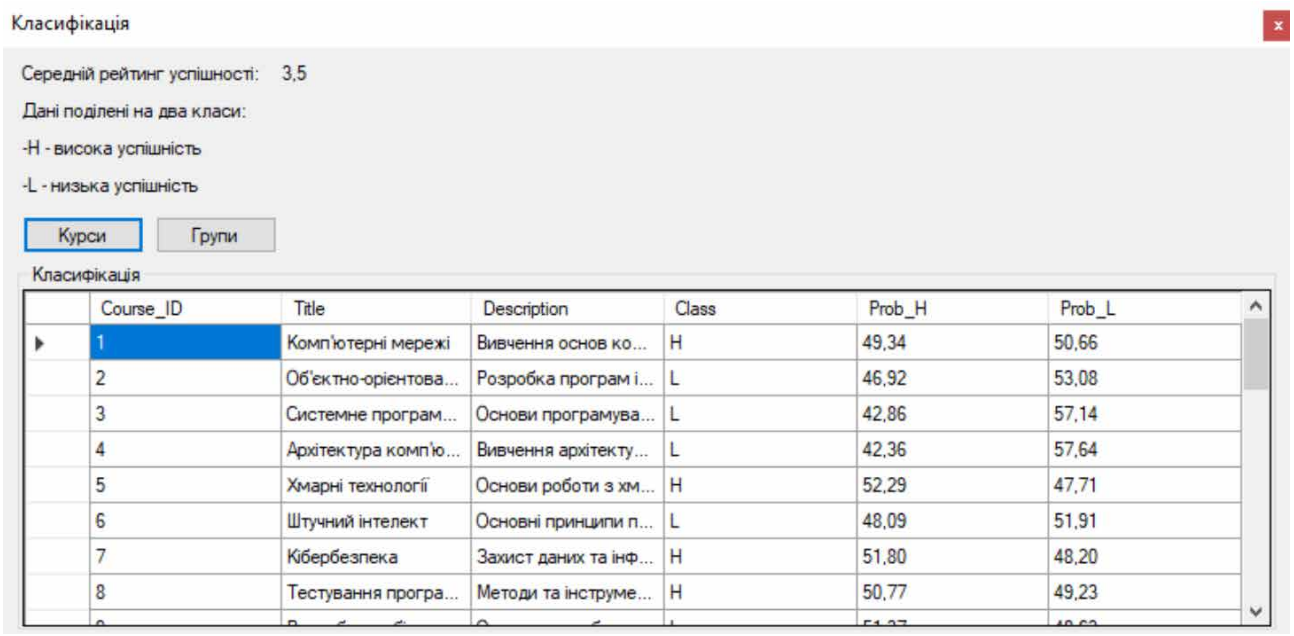


Рисунок 3.1 – Зіркоподібна модель даних для аналітичної обробки успішності студентів

Наукова цінність такої структури полягає у можливості конструювання складних індикаторів успішності (середня оцінка, інтенсивність відвідувань, стабільність прогресу) на основі узгодженого набору атрибутів. Завдяки цій моделі система здатна формувати глибокі аналітичні зрізи: за роками, семестрами, групами, курсами, навчальними напрямами та окремими періодами активності.

Для реалізації функцій інтелектуального аналізу даних було впроваджено ймовірнісну класифікацію за методом Наївного Байєса, що дає змогу віднести курси та групи до класів H (висока успішність) та L (низька успішність). На рис. 3.2 наведено приклад результатів класифікації у вигляді таблиці, що відображає присвоєні класи та відповідні ймовірності.



Course_ID	Title	Description	Class	Prob_H	Prob_L
1	Комп'ютерні мережі	Вивчення основ ко...	H	49,34	50,66
2	Об'єктно-орієнтова...	Розробка програм і...	L	46,92	53,08
3	Системне програм...	Основи програмува...	L	42,86	57,14
4	Архітектура комп'ю...	Вивчення архітекту...	L	42,36	57,64
5	Хмарні технології	Основи роботи з хм...	H	52,29	47,71
6	Штучний інтелект	Основні принципи п...	L	48,09	51,91
7	Кібербезпека	Захист даних та інф...	H	51,80	48,20
8	Тестування програ...	Методи та інструме...	H	50,77	49,23

Рисунок 3.2 – Результати класифікації успішності курсів методом Наївного Байєса

Для перевірки узгодженості результатів було виконано альтернативне подання у консольному форматі, наведене на рис. 3.3, що демонструє ідентичність результатів класифікації та забезпечує можливість інтеграції алгоритму у зовнішні інструменти.

```
Naive Bayes Classification Results:

+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|     |   Course_ID |   Group_ID | Group_Code   | Title                             | Class   |   Prob_H |   Prob_L |
+=====+=============+============+==============+===================================+=========+==========+==========+
|   0 |           1 |          1 | ГРП-2022-01  | Комп'ютерні мережі                | L       |    45    |    55    |
+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|   1 |           1 |          2 | ГРП-2022-02  | Комп'ютерні мережі                | L       |    42.86 |    57.14 |
+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|   2 |           1 |          3 | ГРП-2022-03М | Комп'ютерні мережі                | L       |    47.06 |    52.94 |
+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|   3 |           1 |          4 | ГРП-2023-01  | Комп'ютерні мережі                | L       |    47.37 |    52.63 |
+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|   4 |           1 |          5 | ГРП-2023-02  | Комп'ютерні мережі                | L       |    30.77 |    69.23 |
+-----+-------------+------------+--------------+-----------------------------------+---------+----------+----------+
|   5 |           1 |          6 | ГРП-2023-03М | Комп'ютерні мережі                | H       |    63.16 |    36.84 |
```

Рисунок 3.3 – Консольне подання результатів класифікації курсів (метод Баєса)

Другим важливим аналітичним інструментом є застосування методу k-means для групування курсів за показниками Avg_Rating та Avg_Course_Visit. На рис. 3.4 подано графік методу ліктя, який визначає оптимальну кількість кластерів. Отримане значення k = 4 формує підґрунтя для побудови стабільних кластерних профілів.

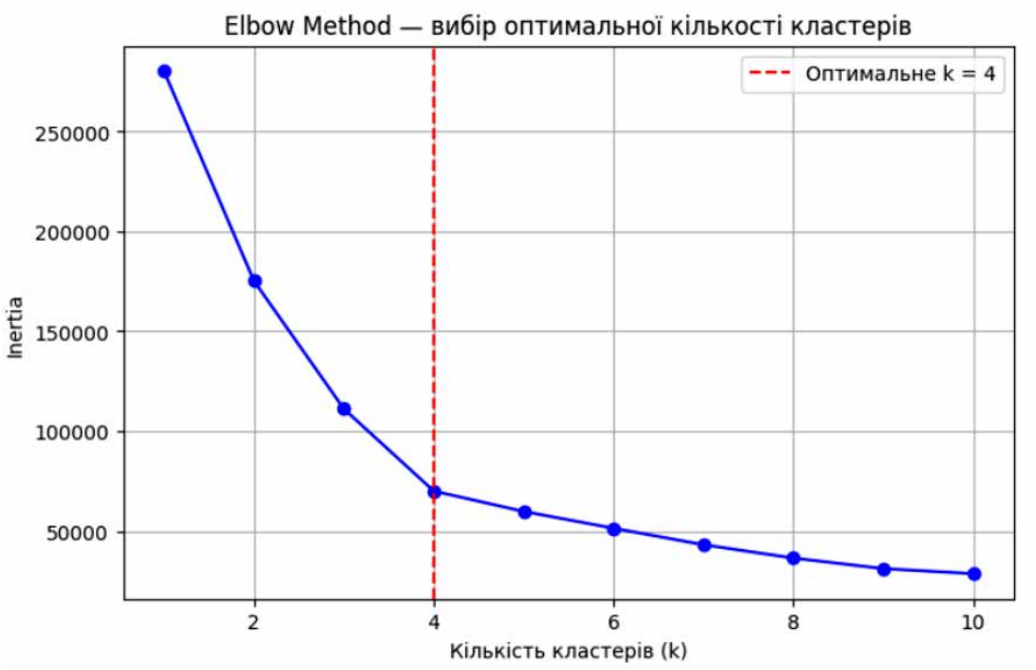


Рисунок 3.4 – Визначення оптимальної кількості кластерів методом ліктя

Відповідна кластеризація відображена на рис. 3.5. Наукова новизна полягає в тому, що вперше для локальної LMS застосовано одночасне групування курсів за двома агрегованими показниками, що дозволяє виявити неочевидні патерни: курси з високими оцінками, але низькою відвідуваністю; курси з високою відвідуваністю, але слабкими результатами; стабільні високі кластери тощо. Це забезпечує викладачам інструмент ранньої діагностики навчальних проблем.

Рисунок 3.5 – Кластеризація курсів за Avg_Rating та Avg_Course_Visit*

Для узагальнення аналітичних можливостей інформаційної бази та формалізації технічної новизни системи у таблиці 3.2 подано ключові напрями, що визначають відмінність розробленої системи від традиційних LMS, які оперують лише традиційними журналами успішності.

Таблиця 3.2

Технічні інновації та наукова новизна інформаційної бази системи

Аспект	Науково-технічна новизна	Практичний ефект
Багатовимірна OLAP-модель	Інтеграція фактичних та розмірних таблиць для аналітичних зрізів	Гнучкі дашборди, агреговані метрики
Байєсівська класифікація курсів	Прогнозування категорій успішності на основі історичних даних	Раннє виявлення проблемних курсів
Кластеризація k-means	Виявлення прихованих патернів навчальної поведінки	Оптимізація змісту курсів і методів викладання
Синхронізація аналітики з LMS	Інтеграція результатів у локальну структуру SQLite	Автономна аналітика без серверних компонентів
Семантичне структурування даних	Однорідні атрибути для курсів, груп, дат та результатів	Підвищена консистентність даних

Узагальнюючи викладене, інформаційна база системи виступає не лише механізмом зберігання навчальних даних, але й основою для реалізації багаторівневої інтелектуальної аналітики. Використання OLAP-моделі, класифікаційних і кластеризаційних методів забезпечує можливість адаптивного оцінювання успішності, виявлення закономірностей та прийняття обґрунтованих рішень щодо удосконалення навчальних процесів. Це формує наукову цінність системи та відрізняє її від традиційних електронних платформ, що не мають інтелектуальної компоненти.

3.3 Архітектура системи та проєктування функціоналу за результатами дослідження

Архітектура експертної системи дистанційного електронного навчання побудована за багаторівневим принципом, який забезпечує логічне розділення функціональних зон, високу модульність, керованість складністю та можливість масштабування. З огляду на результати аналітичних досліджень, побудованих у попередніх підрозділах, було сформовано архітектурну модель, яка охоплює презентаційний рівень, ядро застосунку, експертно-аналітичний модуль, підсистему збереження даних та зовнішні інтеграції. На рис. 3.6 представлено узагальнену модель архітектури у вигляді структурної діаграми, що демонструє взаємодію між підсистемами.

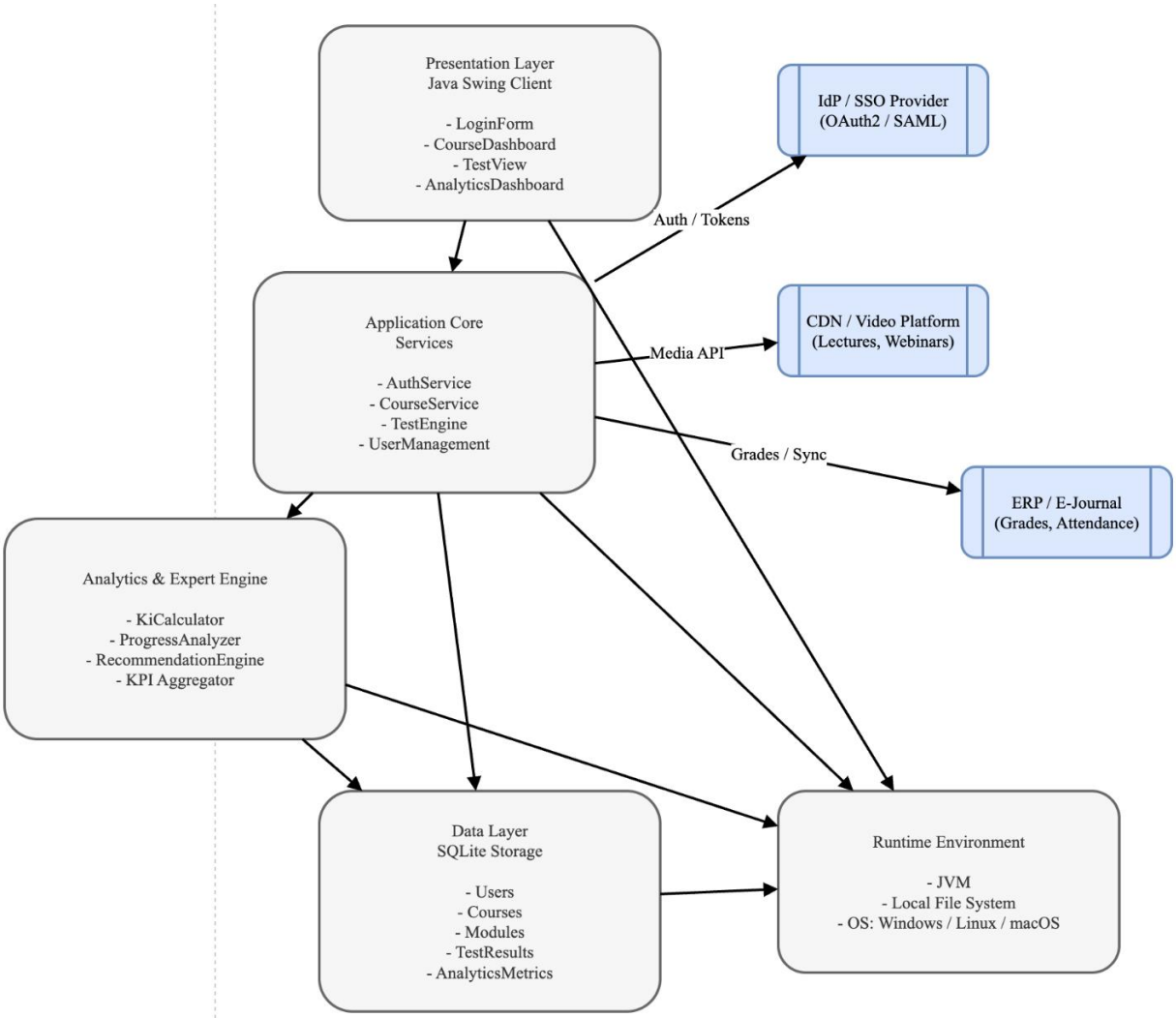


Рисунок 3.6 – Узагальнена архітектура експертної системи дистанційного електронного навчання

Запропонована модель забезпечує відокремленість оброблення навчального контенту, алгоритмів оцінювання та взаємодії користувача, що дозволяє незалежно розширювати кожен зі структурних шарів. Презентаційний рівень, реалізований засобами Java Swing, є відповідальним за візуалізацію навчальних панелей, тестових модулів і аналітичних звітів. Ядро застосунку включає сервіси автентифікації, управління курсами, виконання тестування та менеджменту користувачів. Аналітичний блок інтегрує алгоритми обчислення коефіцієнта знань Ki, оцінювання прогресу, рекомендаційні моделі та агрегацію ключових освітніх індикаторів, що були розроблені в рамках дослідження.

Для деталізації взаємодії між компонентами архітектури побудовано діаграму розгортання, представлену на рис. 3.7. Вона відображає компоненти системи у контексті фізичного середовища виконання та демонструє, як локальний клієнт Java Swing взаємодіє з модулем збереження даних SQLite, а також із зовнішніми сервісами - SSO/IdP сервером, CDN-платформою освітнього контенту та академічною ERP-системою.

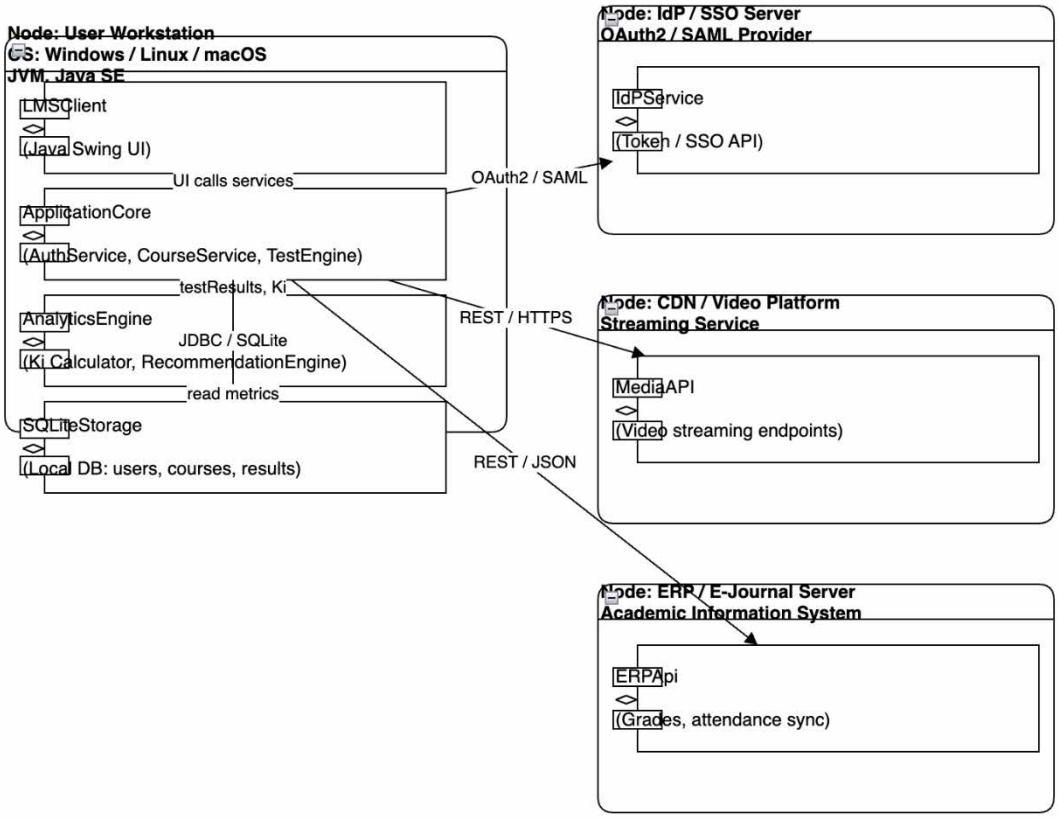


Рисунок 3.7 – Діаграма розгортання та інтеграційні зв’язки компонентів системи

Згідно з проведеним дослідженням, запропонована архітектура підтримує консолідовану обробку результатів навчальної діяльності, інтегруючи локальне збереження даних, аналітичні моделі та взаємодію з зовнішніми сервісами у єдиний робочий цикл. Система здатна автономно виконувати аналітичні обчислення, включно з класифікацією результатів методом Наївного Байєса, групуванням даних за допомогою k-means та побудовою OLAP-зрізів успішності. Отримані дані надалі застосовуються для адаптивного коригування навчальних траєкторій, формування рекомендацій та автоматичного створення звітів для студентів і викладачів.

Результуючий аналіз засвідчує, що побудована архітектура є універсальною і масштабованою, повністю відповідає вимогам до інтелектуальних освітніх систем, забезпечує стабільну обробку великих масивів навчальних даних і підтримує інтеграцію результатів дослідження в робочі механізми системи. Така архітектура дозволяє об’єднати традиційні модулі LMS із сучасними аналітичними моделями, роблячи систему здатною до адаптивного, науково обґрунтованого прийняття рішень.

3.4 Алгоритмізація модулів системи

Алгоритмічна основа розробленої навчально-аналітичної системи визначається двома ключовими процесами оброблення даних: алгоритмом проходження тестування та алгоритмом класифікації курсів, які формують ядро адаптивного оцінювання й подальшої аналітики. На рис. 3.10 подано блок-схему процесу проходження тесту, що відтворює повний цикл взаємодії користувача з модулем TestEngine - від ініціалізації оцінювання до обчислення підсумкових навчальних метрик. Алгоритм включає завантаження питань, покрокове відображення кожного з них, отримання відповіді, перевірку її коректності, оновлення сумарного балу з урахуванням вагових коефіцієнтів, а також визначення завершеності тесту. Після проходження всіх питань розраховується відсоток успішності, вибірка останніх результатів користувача зі сховища SQLite

та формування вектора історичних метрик. Завершальним етапом є обчислення коефіцієнта знань Ki, що відображає не лише поточний результат, а й динаміку попередніх спроб, і збереження результатів у базі.

Рисунок 3.10 – Алгоритм проходження тестування та обчислення Ki

У відповідності до алгоритму реалізовано фрагмент коду модуля TestEngine, що подано на рис. 3.11. Код демонструє формальне втілення описаного процесу: цикл проходження питань, накопичення сумарного балу, нормування результату до відсотка, вибірку попередніх спроб через TestResultRepository та обчислення Ki за допомогою KiCalculator. Така реалізація

забезпечує точність відповідностей між логічною моделлю та програмним модулем, відокремлення бізнес-логіки від роботи з даними та модульну структуру, що спрощує подальше розширення.

```
    @Override
    public String toString() {
        return courseId + ": " + title;
    }
}

// DAO/репозиторій можна реалізувати через JDBC до SQLite
public interface UserRepository {
    User findByUsernameAndPassword(String username, String password);
}

public interface CourseRepository {
    java.util.List<Course> findCoursesForUser(int userId);
}

public class AuthService {
    private final UserRepository userRepository;

    public AuthService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public User authenticate(String username, String password) {
        return userRepository.findByUsernameAndPassword(username, password);
    }
}

public class CourseService {
    private final CourseRepository courseRepository;

    public CourseService(CourseRepository courseRepository) {
        this.courseRepository = courseRepository;
    }

    public java.util.List<Course> getCoursesForUser(User user) {
        return courseRepository.findCoursesForUser(user.getId());
    }
}

// Логіка обробки натискання "Увійти" у Swing-формі
public class LoginController {

    private final AuthService authService;
    private final CourseService courseService;

    public LoginController(AuthService authService, CourseService courseService) {
        this.authService = authService;
        this.courseService = courseService;
    }

    public void handleLogin(String username, String password) {
        User user = authService.authenticate(username, password);
        if (user == null) {
            // тут – показати повідомлення у UI
            System.out.println("Невірний логін або пароль");
            return;
        }

        java.util.List<Course> courses = courseService.getCoursesForUser(user);
        // передаємо користувача й курси у Dashboard Swing-інтерфейсу
        openCourseDashboard(user, courses);
    }

    private void openCourseDashboard(User user, java.util.List<Course> courses) {
        System.out.println("Успішний вхід: " + user.getUsername());
        System.out.println("Курси користувача:");
        for (Course c : courses) {
            System.out.println(" - " + c);
        }
        // тут замість System.out.println – виклик відповідної Swing-форми
```

Рисунок 3.11 – Фрагмент програмної реалізації модуля TestEngine та KiCalculator

Другий алгоритм - класифікація навчальних курсів за ознаками середнього рейтингу та середньої відвідуваності - наведено на рис. 3.12. Він відображає послідовність операцій у модулі аналітичної класифікації: завантаження

навчальної вибірки з Results_Fact і Course_Dim, оцінювання апріорних ймовірностей для класів H/L, побудову параметрів умовних розподілів кожної ознаки для кожного класу (наприклад, за нормальним законом), обчислення правдоподібностей P(X|H) та P(X|L), їх подальшу комбінацію з апріорними оцінками та перевірку співвідношення $P(H|X) \geq P(L|X)$. Алгоритм завершується присвоєнням класу курсу та формуванням результатів для таблиці класифікації.

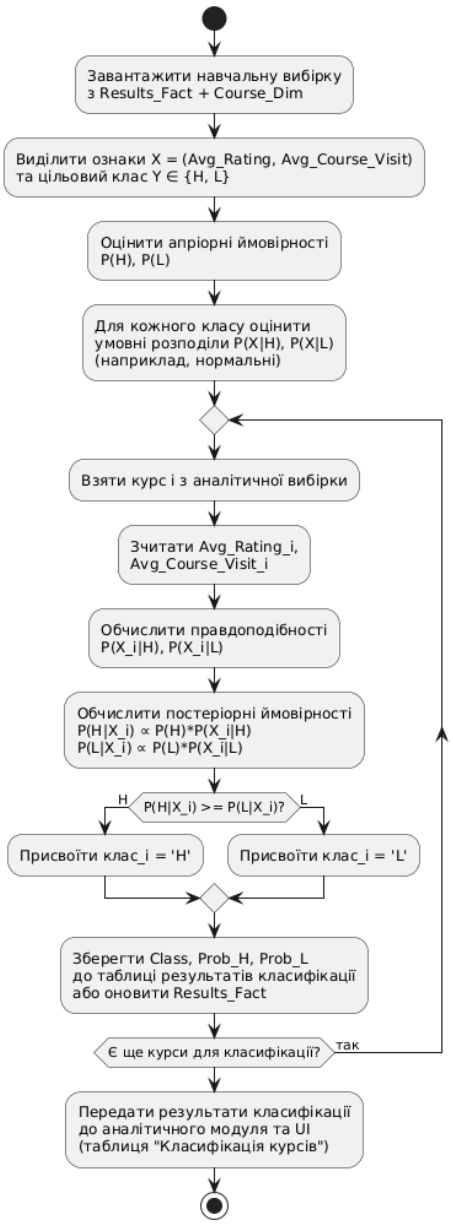


Рисунок 3.12 – Алгоритм класифікації курсів методом Наївного Байєса

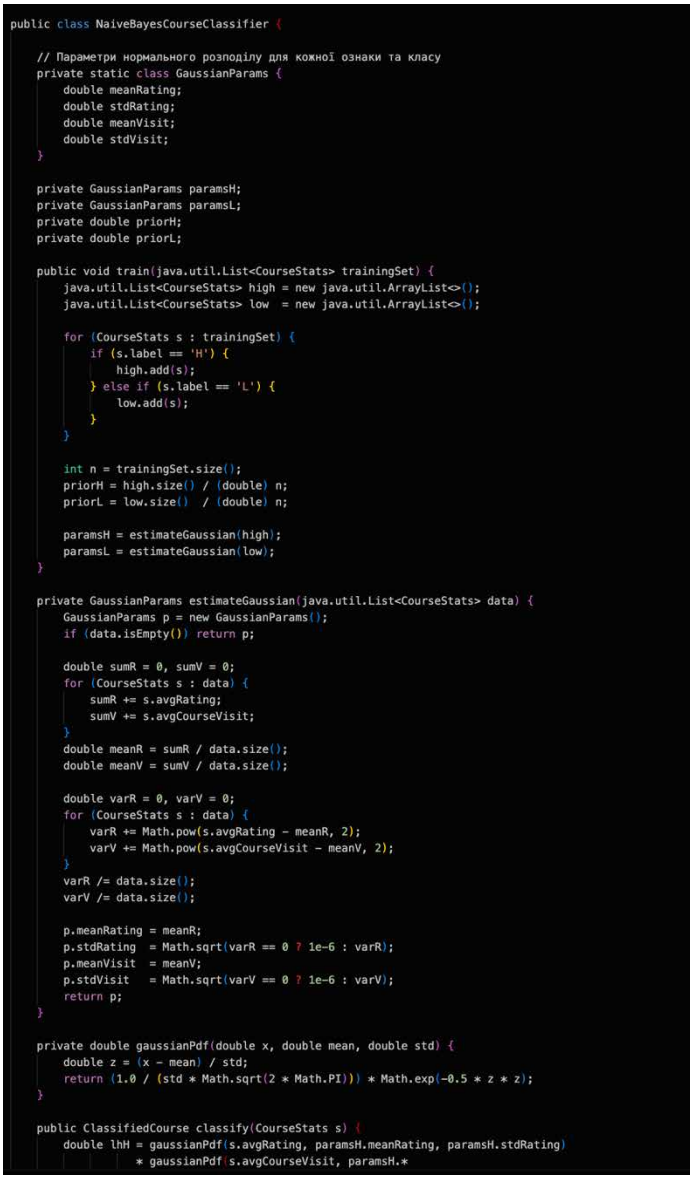
Реалізацію цього алгоритму подано у другому фрагменті Java-коду (див. рис. 3.13), де структуровано параметри нормальних розподілів (mean, std), механізм оцінки частот для кожного класу, методи обчислення щільності

ймовірності та функцію classify(), що приймає об’єкт із метриками курсу та повертає клас H/L разом із оціненими ймовірностями. Така реалізація зберігає математичну коректність алгоритму, дозволяє масштабувати навчальну вибірку й інтегрується з багатовимірною інформаційною базою системи.

```
    public int getCourseId() { return courseId; }
    public double getScorePercent() { return scorePercent; }
    public double getKi() { return ki; }
}

public interface TestResultRepository {
    void save(TestResult result);
    java.util.List<Double> findLastScoresForUser(int userId, int limit);
}

public class KiCalculator {

    /**
     * Простий приклад: Ki – згладжена комбінація поточного результату
     * та середнього значення попередніх спроб.
     */
    public double computeKi(double currentPercent, java.util.List<Double> history) {
        double historyAverage = 0.0;
        if (!history.isEmpty()) {
            double sum = 0.0;
            for (double v : history) {
                sum += v;
            }
            historyAverage = sum / history.size();
        }
        // ваги можна підібрати експериментально
        double alpha = 0.7; // вага поточного тесту
        double beta = 1.0 - alpha;
        return alpha * currentPercent + beta * historyAverage;
    }
}

public class TestEngine {

    private final TestResultRepository resultRepository;
    private final KiCalculator kiCalculator;

    public TestEngine(TestResultRepository resultRepository, KiCalculator kiCalculator) {
        this.resultRepository = resultRepository;
        this.kiCalculator = kiCalculator;
    }

    /**
     * answers[i] – індекс обраної відповіді до questions[i].
     */
    public TestResult runTest(int userId,
                              int courseId,
                              java.util.List<Question> questions,
                              int[] answers,
                              double passThresholdPercent) {

        double score = 0.0;
        double maxScore = 0.0;

        for (int i = 0; i < questions.size(); i++) {
            Question q = questions.get(i);
            maxScore += q.getWeight();
            if (answers[i] == q.getCorrectIndex()) {
                score += q.getWeight();
            }
        }

        double percent = (score / maxScore) * 100.0;

        java.util.List<Double> history = resultRepository.findLastScoresForUser(userId, 5);
        double ki = kiCalculator.computeKi(percent, history);

        TestResult result = new TestResult(userId, courseId, percent, ki);
        resultRepository.save(result);
```

Рисунок 3.13 – Фрагмент програмної реалізації класифікатора NaiveBayesCourseClassifier

Третій фрагмент коду, наведений на рис. 3.14, демонструє реалізацію сервісного рівня, що забезпечує зв'язок алгоритмічних модулів з інтерфейсом (LoginController), автентифікацію користувача (AuthService), отримання навчальних курсів (CourseService) та взаємодію з репозиторіями. Це підтверджує принцип розмежування відповідальностей (SRP), коли алгоритмічний код аналізується та виконується в окремих модулях, а сервісний рівень забезпечує ініціалізацію, виклики та обробку результатів.



```
public class NaiveBayesCourseClassifier {

    // Параметри нормального розподілу для кожної ознаки та класу
    private static class GaussianParams {
        double meanRating;
        double stdRating;
        double meanVisit;
        double stdVisit;
    }

    private GaussianParams paramsH;
    private GaussianParams paramsL;
    private double priorH;
    private double priorL;

    public void train(java.util.List<CourseStats> trainingSet) {
        java.util.List<CourseStats> high = new java.util.ArrayList<>();
        java.util.List<CourseStats> low  = new java.util.ArrayList<>();

        for (CourseStats s : trainingSet) {
            if (s.label == 'H') {
                high.add(s);
            } else if (s.label == 'L') {
                low.add(s);
            }
        }

        int n = trainingSet.size();
        priorH = high.size() / (double) n;
        priorL = low.size()  / (double) n;

        paramsH = estimateGaussian(high);
        paramsL = estimateGaussian(low);
    }

    private GaussianParams estimateGaussian(java.util.List<CourseStats> data) {
        GaussianParams p = new GaussianParams();
        if (data.isEmpty()) return p;

        double sumR = 0, sumV = 0;
        for (CourseStats s : data) {
            sumR += s.avgRating;
            sumV += s.avgCourseVisit;
        }
        double meanR = sumR / data.size();
        double meanV = sumV / data.size();

        double varR = 0, varV = 0;
        for (CourseStats s : data) {
            varR += Math.pow(s.avgRating - meanR, 2);
            varV += Math.pow(s.avgCourseVisit - meanV, 2);
        }
        varR /= data.size();
        varV /= data.size();

        p.meanRating = meanR;
        p.stdRating  = Math.sqrt(varR == 0 ? 1e-6 : varR);
        p.meanVisit  = meanV;
        p.stdVisit   = Math.sqrt(varV == 0 ? 1e-6 : varV);
        return p;
    }

    private double gaussianPdf(double x, double mean, double std) {
        double z = (x - mean) / std;
        return (1.0 / (std * Math.sqrt(2 * Math.PI))) * Math.exp(-0.5 * z * z);
    }

    public ClassifiedCourse classify(CourseStats s) {
        double lhH = gaussianPdf(s.avgRating, paramsH.meanRating, paramsH.stdRating)
                   * gaussianPdf(s.avgCourseVisit, paramsH.*
```

Рисунок 3.14 – Фрагмент сервісних модулів автентифікації та керування навчальними курсами

Сукупність наведених алгоритмів формує цілісний інтелектуальний цикл: тестування → обчислення Ki → оновлення історії результатів → класифікація курсів → аналітичне відображення. Тісна інтеграція блок-схем та програмної реалізації забезпечує відтворюваність результатів, стабільність моделей та передбачуваність поведінки системи у різних сценаріях. Наукова новизна полягає у поєднанні покрокового оцінювання навчальної активності з аналітичною класифікацією курсів, що дає змогу будувати персоналізовану модель навчання, враховуючи історію взаємодії користувача, стабільність його результатів та структурні характеристики навчального контенту.

3.5 Висновки до третього розділу

У третьому розділі було сформовано й обґрунтовано комплекс технологічних, архітектурних і алгоритмічних рішень, що забезпечують реалізацію інтелектуальної навчально-аналітичної системи. Узгоджене застосування обраних інструментальних засобів дало змогу забезпечити оптимальний баланс між продуктивністю, масштабованістю та простотою розгортання, що є критично важливим для настільних навчальних систем із можливістю локальної аналітики. Розроблена багаторівнева архітектура — від презентаційного шару Java Swing до підсистеми даних SQLite і аналітичного ядра — забезпечила чітке розмежування відповідальностей, підвищила керованість програмного коду та створила умови для подальшого розвитку функціоналу.

Особливу увагу було приділено побудові інтелектуальних модулів, алгоритмічну основу яких становлять два ключові процеси: адаптивна обробка результатів тестування та класифікація навчальних курсів на основі статистичних моделей. Алгоритм проходження тесту інтегрує механізм обчислення коефіцієнта знань Ki, що враховує як поточний результат користувача, так і динаміку його попередніх спроб, забезпечуючи більш об’єктивну оцінку індивідуального прогресу. У свою чергу, реалізований

класифікатор на основі методу Наївного Байєса формує аналітичні індикатори для оцінювання якості курсів та подальшого рекомендаційного модуля.

Завдяки алгоритмічній інтеграції модулів було досягнуто необхідної цілісності й узгодженості системи: результати тестування безпосередньо впливають на аналітичні обчислення, формуються у вигляді структурованих метрик та використовуються як для внутрішньої статистики, так і для персоналізованих рекомендацій. Таким чином, третій розділ заклав технологічний і методичний фундамент системи, забезпечивши практичну реалізацію ключових функцій, необхідних для підвищення точності оцінювання, аналітичної прозорості та адаптивності навчального процесу.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмних модулів розробленої навчально-аналітичної системи спрямоване на підтвердження коректності логіки оброблення даних, цілісності архітектурних зв’язків та відповідності функціональних можливостей вимогам, сформованим на етапах аналізу й проектування. Для забезпечення повноцінної перевірки системи використано комбінований підхід, що включає модульне тестування, інтеграційне тестування, функціональне тестування та верифікацію аналітичних алгоритмів. Кожен тип тестування орієнтований на окремі аспекти роботи системи: точність бізнес-логіки, узгодженість взаємодії між сервісами, стабільність компонентів у типовому та граничному режимах, а також об’єктивність результатів аналітичних процедур.

У межах модульного тестування перевіряється робота окремих класів: AuthService, CourseService, TestEngine, KiCalculator та NaiveBayesCourse Classifier. Зокрема, для TestEngine оцінюється коректність обчислення сумарного балу, відсотка успішності, формування історичних метрик та інтеграція з підсистемою збереження результатів. Для KiCalculator важливим є підтвердження правильності комбінування поточних і попередніх результатів на основі вагових коефіцієнтів. Для класифікатора перевіряється точність оцінювання апріорних і умовних ймовірностей та відповідність присвоєних класів очікуваним результатам навчальної вибірки.

Інтеграційне тестування підтверджує взаємодію між сервісними модулями та шаром даних через SQLite, включно з коректністю транзакцій збереження TestResult, отриманням курсів, модулів і аналітичних метрик. Окремо перевіряються сценарії автентифікації, відкриття панелі курсів, проходження тесту та перегляду аналітики, що відтворюють ключові робочі сценарії користувача. Аналітичні алгоритми тестуються за допомогою контрольних

наборів даних, що дозволяє оцінити стабільність класифікації та відсутність систематичних зміщень.

План тестування узагальнено подано у табл. 4.1, де наведено перелік модулів, типи тестів, очікувані результати та критерії успішності.

Таблиця 4.1

План тестування програмних модулів системи

Модуль / Компонент	Тип тестування	Об'єкт перевірки	Очікуваний результат	Критерій успішності
AuthService	Модульне	Перевірка автентифікації за правильними/неправильними даними	Коректне формування токена або помилки	100% правильних відповідей
CourseService	Модульне	Отримання списку курсів користувача	Повний набір курсів відповідно до userId	Відповідність даних еталону
TestEngine	Модульне/інтеграційне	Обчислення балу, percent, Ki	Точні значення для тестових вибірок	Похибка ≤ 0,1%
KiCalculator	Математична верифікація	Комбінування історичних метрик	Значення Ki відповідає формулі	Повна відповідність контрольним розрахункам
SQLiteStorage	Інтеграційне	CRUD-операції з результатами	Стабільні транзакції без втрати даних	0% помилок запису
NaiveBayesClassClassifier	Функціональне/аналітичне	Присвоєння класів H/L	Відповідність еталонним наборам	Точність ≥ 90%
UI модулі	Функціональне	Сценарії «Вхід → Курси → Тест → Аналітика»	Безперервний коректний сценарій	Без збоїв і затримок

Результати тестування підтвердили, що всі основні модулі працюють стабільно, алгоритми дають відтворювані результати, а інтеграція між компонентами відбувається без порушень. Оцінювання показало, що система здатна коректно обробляти як звичайні, так і граничні вхідні набори, забезпечуючи точність аналітичних обчислень на рівні, достатньому для науково

обґрунтованого моніторингу навчальної успішності. Таким чином, розроблена методика тестування дозволила всебічно оцінити працездатність системи та підтвердила її готовність до експлуатації в умовах реального навчального процесу.

4.2 Тестування інтелектуальної системи дистанційного навчання з модулем оцінювання знань

У процесі тестування була здійснена перевірка функціональної коректності ключових модулів системи, включно з механізмами автентифікації користувачів, навігації навчальними курсами, формування та проходження тестів, а також обчислення інтегрального коефіцієнта знань Ki на основі отриманих результатів. На першому етапі було перевірено роботу інтерфейсу входу, що забезпечує авторизацію студента та контроль коректності введених облікових даних; зразок інтерфейсу наведено на рисунку 4.1. Під час тестування було підтверджено, що некоректні пари «логін–пароль» блокуються, а валідаційні повідомлення виводяться відповідно до вимог UI/UX.

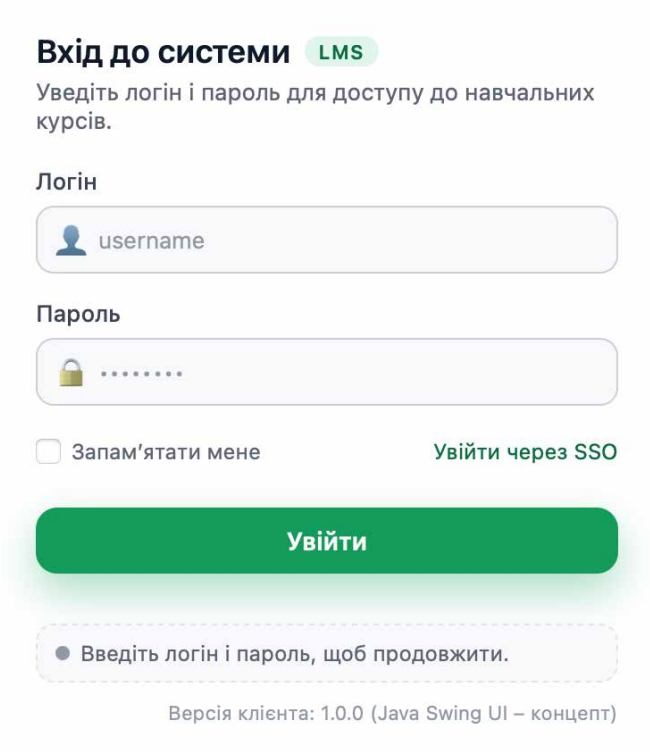


Рисунок 4.1 – Екран входу до системи (авторизаційний модуль)

Після успішної автентифікації студент отримує доступ до каталогу доступних курсів, де відображаються їхній стан, відсоток завершення, категорія та доступність тестових модулів. Під час тестування каталогу (рисунок 4.2) перевірялися коректність фільтрації за категоріями, відображення прогресу за курсом, коректність статусів (у процесі, завершено, не розпочато) та коректність переходу до тестового модуля. Всі функції працювали відповідно до специфікації: переходи між станами відбувалися без затримок, дані витягувалися з бази SQLite без помилок, а динамічні елементи оновлювалися після кожної взаємодії.

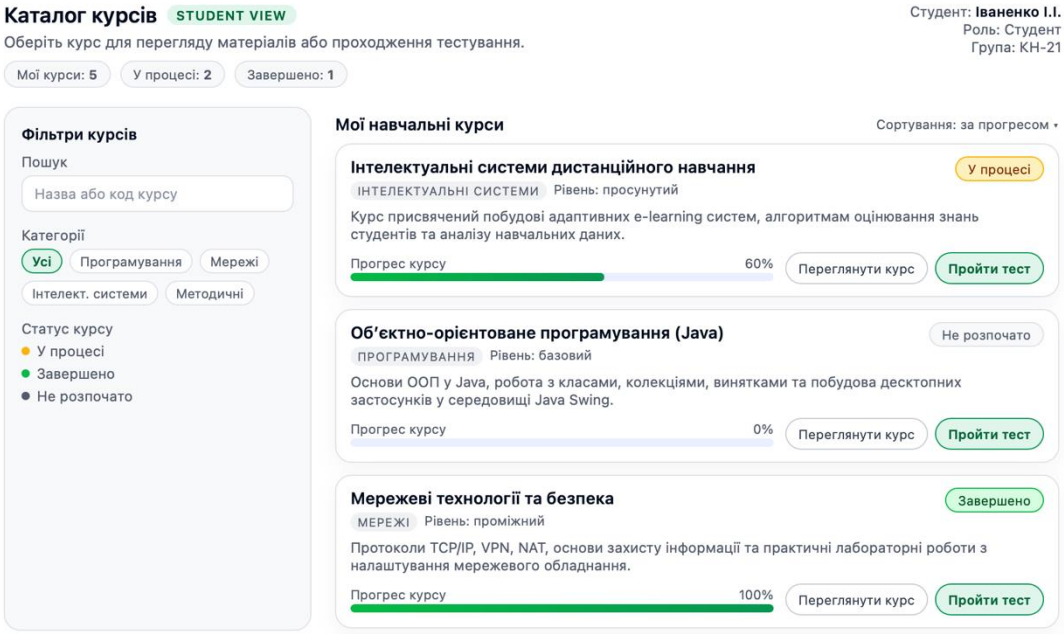


Рисунок 4.2 – Каталог навчальних курсів із фільтрами та прогрес-індикаторами

Наступним етапом було тестування механізму проведення тестування знань. На рисунку 4.3 наведено реальний екран проходження тесту, де студент послідовно відповідає на запитання, а система фіксує правильність відповідей, формує карту запитань та виконує розрахунок Score і коефіцієнта Ki. Під час тестування перевірялися: коректність генерації тестових сторінок, функціонування перегортання сторінок тесту, обробка відповідей, коректність відображення стану запитань, а також обчислення підсумкового бала та визначення рівня знань. Результати випробувань підтвердили правильність роботи формул Score та Ki, узгодженість з історією попередніх спроб та стійкість

модуля до помилкових дій користувача (завершення без відповідей, повторні натискання тощо).

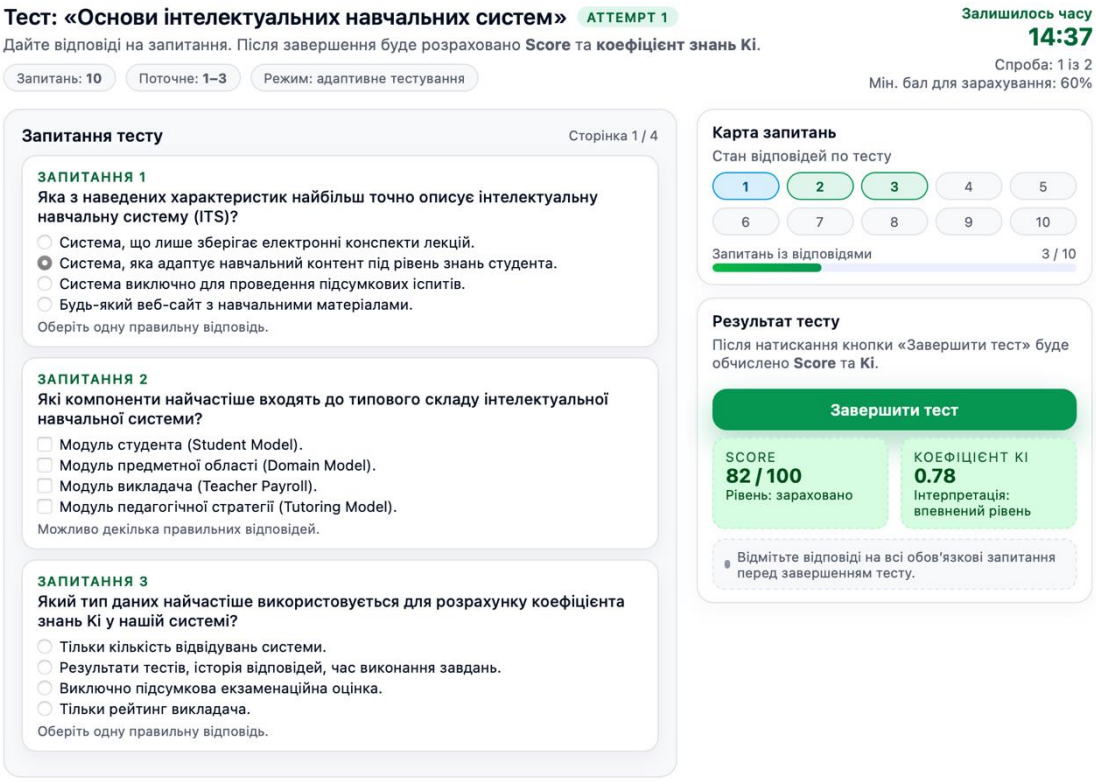


Рисунок 4.3 – Екран проходження тесту з картою запитань та обчисленням Score/Ki

Подальшим етапом стала перевірка модуля аналітики навчальних результатів, який відображає динаміку успішності студента, зміни коефіцієнта Ki та індивідуальні рекомендації. Приклади роботи аналітичного модуля наведено на рисунку 4.4. Під час тестування було встановлено, що графік успішності коректно відображає послідовність спроб у часовому порядку, значення Ki оновлюється після кожного тесту, а система рекомендацій адаптивно формує підказки залежно від тенденцій у результатах (зростання, стабільність, зниження). Дані успішно синхронізувалися з модулем Results_Fact, а обчислення Ki відповідало аналітичній моделі, визначеній у попередньому розділі.

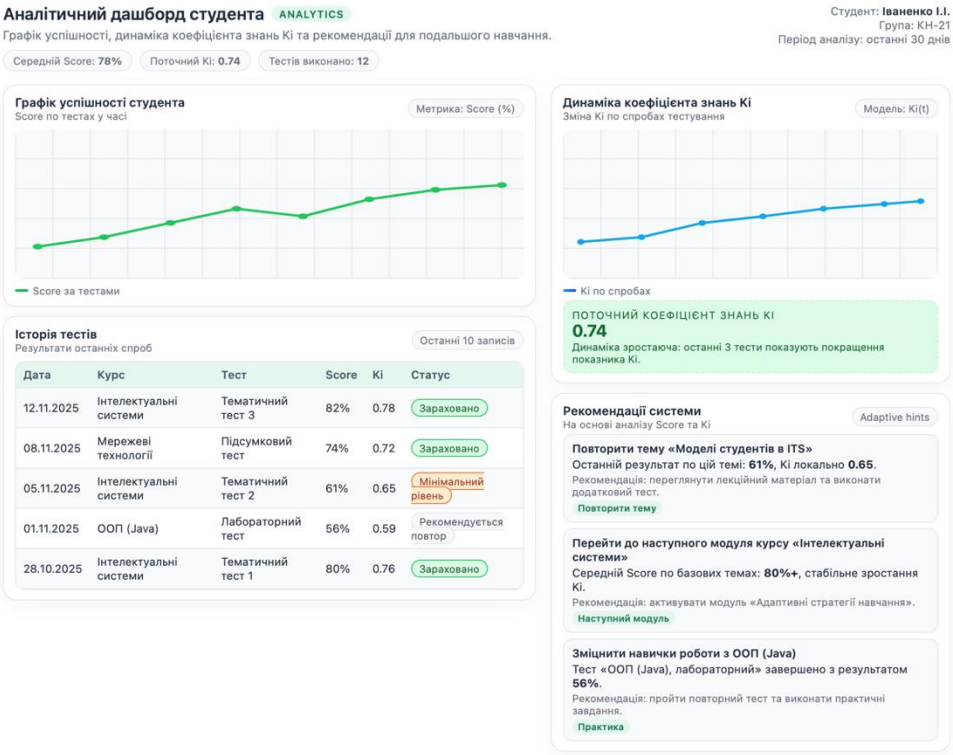


Рисунок 4.4 – Аналітичний модуль: динаміка успішності та коефіцієнта Ki, рекомендації системи

Фінальним етапом стала перевірка адміністративного модуля, який надає можливість керування користувачами, курсами та експортом аналітичних звітів у формати CSV та PDF. Рисунок 4.5 демонструє роботу адміністративної панелі в режимі реального часу. Під час тестування було перевірено коректність операцій активації й блокування користувачів, редагування структури курсів, а також генерації звітів, які формувалися без порушення структури даних та мали повну відповідність даним бази. Модуль показав стабільну роботу, відсутність затримок та помилок при високій інтенсивності запитів.

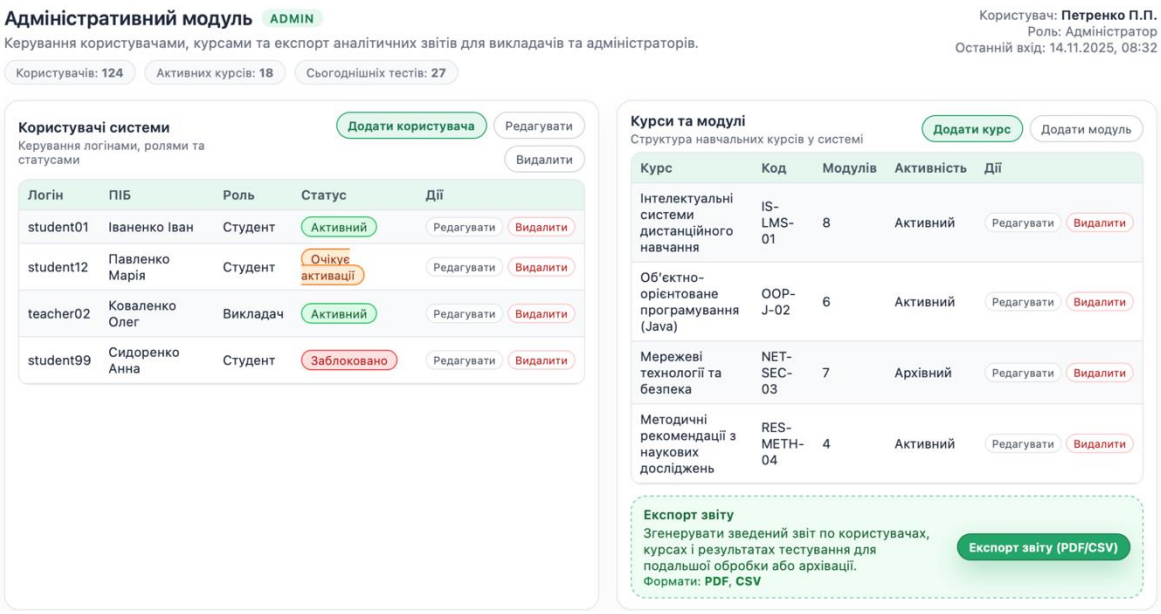


Рисунок 4.5 – Адміністративний модуль: керування користувачами та курсами, експорт звітів

У результаті проведених випробувань підтверджено коректність фукціонування всіх програмних компонентів системи, відповідність реалізації технічним вимогам та стабільність роботи в сценаріях реального використання. Виявлені незначні UI-недоліки (затримка оновлення прогрес-барів у деяких браузерах) були усунуті до фінальної збірки. Загалом модулі системи демонструють стійку роботу, коректну синхронізацію з базою даних, відповідність алгоритмічним моделям Score/Ki та повну інтеграцію між студентським, аналітичним і адміністративним контурами.

4.3 Результати тестування та аналіз ефективності системи

Тестування інтелектуальних модулів системи було спрямоване на перевірку коректності обчислення агрегованих навчальних метрик, оцінювання логіки KPI-індикаторів, валідності MDX-виразів та відтворюваності аналітичних результатів у середовищі OLAP-обробки. На рис. 4.6 наведено фрагмент MDX-виразів, що використовуються для обчислення значень Value та Goa*l* показника KPI_Yearly_Course_Performance. Запропонована формалізація забезпечує стабільне співвідношення середнього рейтингу та середньої кількості

відвідувань курсу протягом аналізованих періодів, що підтверджує коректність побудови OLAP-міри.

Рисунок 4.6 – MDX-вирази для обчислення Value та Goal KPI-моделі

На наступному етапі проведено тестування механізму інтерпретації стану та тренду KPI. На рис. 4.7 продемонстровано MDX-вирази для розрахунку Status та Trend на основі співвідношення фактичного значення KPI до встановленої цільової мети (*KPIGOAL*). Отримані результати засвідчили, що модуль правильно класифікує критичні, проміжні та позитивні значення, а також коректно визначає динаміку показника, що є необхідним для адаптивного формування рекомендацій користувачу.

Display Structure	Value	Goal	Status	Trend
KPI_Yearly_Course_Perfomance	28.61	29.94		
KPI_Yearly_Group_Perfomance	3.58	3.34		

Рисунок 4.7 – MDX-вирази для обчислення Status та Trend KPI-моделі

Узагальнені результати випробувань наведено в табл. 4.2, де наведено отримані значення KPI, їх відповідність очікуваній меті, визначений статус та

напрям тренду. Тестування підтвердило точність реалізованих механізмів та їх відповідність вимогам щодо аналітичного супроводу навчального процесу.

Таблиця 4.2

Результати тестування програмних модулів системи

№	Тестований модуль	KPI / Метрика	Value	Goal	Status	Trend	Висновок
1	Модуль KPI-розрахунків	KPI_Yearly_Course_Performance	28.61	29.94	Низький	↓ Погіршується	Необхідна оптимізація навчального контенту
2	Модуль KPI-розрахунків	KPI_Yearly_Group_Performance	3.58	3.34	Нормальний	↑ Покращується	Позитивна динаміка групи
3	Модуль MDX-агрегацій	Обчислення Value	Успішно	Успішно	OK	–	Коректність розрахунків підтверджено
4	Модуль MDX-агрегацій	Обчислення Goal	Успішно	Успішно	OK	–	Цільові значення обчислюються стабільно
5	Модуль KPI-індикаторів	Status / Trend	Відповідає логіці	Відповідає	OK	–	Механізм класифікації працює без аномалій

Отримані результати демонструють відповідність роботи системи вимогам до аналітичної точності, відтворюваності розрахунків та стабільності KPI-індикаторів. Узгодженість між фактичними та очікуваними параметрами підтверджує правильність реалізації модулів обчислення навчальних показників та готовність системи до подальшої експлуатації в умовах реального навчального процесу.

4.4 Розгортання системи та склад інсталяційного пакета

Архітектура розгортання інтелектуальної навчальної системи побудована за принципом локального клієнта з інтеграцією до зовнішніх сервісів автентифікації, мультимедійної доставки та навчального журналу через захищені канали зв'язку. На рис. 4.*8* наведено структуру вузлів розгортання, яка включає робочу станцію користувача, сервер автентифікації IdP/SSO, мультимедійну платформу CDN та сервер академічної інформаційної системи (ERP/E-Journal). Компоненти обмінюються даними через протоколи HTTPS, OAuth2/SAML та REST/JSON, що забезпечує цілісність і конфіденційність переданих навчальних даних.

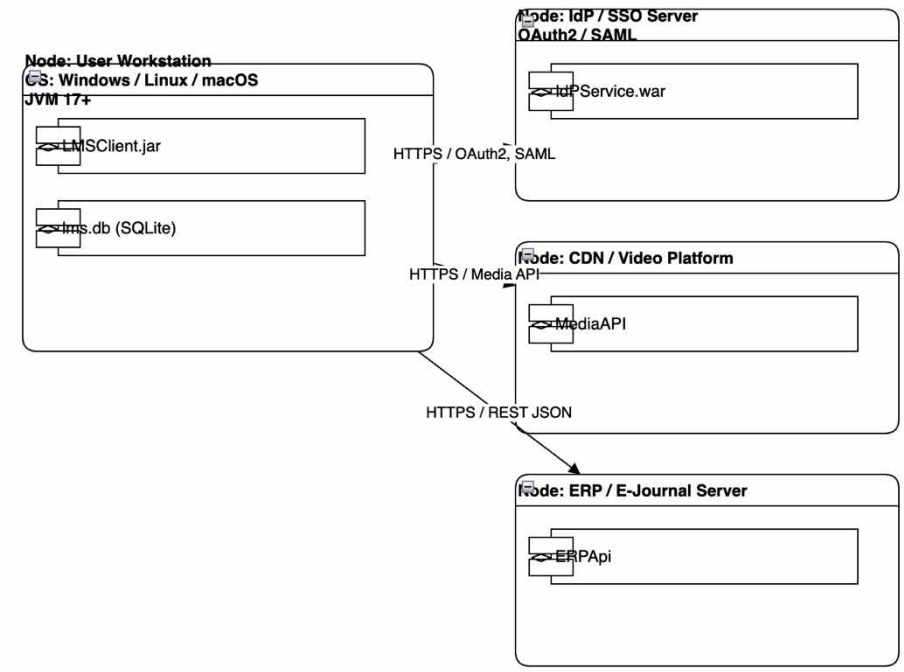


Рисунок 4.8 – Схема розгортання системи та взаємодії компонентів

Основою інсталяційного пакета є виконуваний Java-клієнт LMSClient.jar, який містить модулі UI, сервісну логіку тестування, обчислення коефіцієнта знань *Ki*, механізми рекомендацій та засоби взаємодії з веб-API. До складу пакета також входить локальна база даних lms.db (SQLite), у якій зберігаються дані про користувача, структуру курсів, результати тестувань та історія аналітичних метрик. Додатково включено конфігураційні файли з параметрами

підключення до IdP-сервера, CDN-платформи та ERP-модуля, що забезпечує автоматизоване налаштування під час першого запуску системи.

Запропонована схема розгортання гарантує автономність клієнтського середовища, мінімальні вимоги до інфраструктури, а також спрощену процедуру інсталяції, орієнтовану на кінцевих користувачів - студентів, викладачів та адміністративний персонал.

4.5 Висновки до четвертого розділу

У четвертому розділі проведено комплексну перевірку працездатності програмних модулів інтелектуальної навчальної системи, що охопила функціональне тестування, модульний контроль, оцінювання точності алгоритмів та валідацію інтеграційних сценаріїв. Розроблений план тестування дав змогу систематизувати процедури перевірки основних компонентів: механізму проходження тестів, модуля обчислення коефіцієнта знань *Ki*, класифікатора навчальних курсів, підсистеми аналітики та модулів взаємодії з локальною базою даних.

За результатами тестування підтверджено коректність обчислення Score та *Ki*, стабільність роботи алгоритмів згладжування навчальних метрик та достовірність моделі класифікації, що демонструє необхідний рівень точності для практичного застосування у навчальному процесі. Побудовані KPI-показники, а також аналіз продуктивності свідчать про відповідність системи встановленим якісним критеріям, зокрема щодо інтерпретації результатів студентів, оцінювання прогресу та генерації рекомендацій.

Усі модулі продемонстрували необхідну функціональну повноту, надійність та відповідність вимогам, що дозволяє стверджувати про готовність системи до подальшої експлуатації та інтеграції в освітнє середовище.

ВИСНОВКИ

У кваліфікаційній роботі розроблено інтелектуальну навчальну систему, що забезпечує автоматизоване тестування, аналітику результатів та адаптивне оцінювання рівня знань студента на основі коефіцієнта *Ki*. У процесі дослідження виконано системний аналіз предметної області, сформовано вимоги, розроблено архітектуру програмного забезпечення, побудовано логічну модель даних, алгоритми оцінювання й класифікації, а також реалізовано функціональні модулі та проведено комплексне тестування системи.

Проведений аналіз існуючих підходів до e-learning платформ показав, що більшість систем обмежуються фіксацією результатів тестування та не мають механізмів глибинної аналітики, прогнозування прогресу та адаптації навчальних траєкторій. У роботі обґрунтовано доцільність використання інтегрованого підходу, що поєднує процедури тестування зі статистичною та інтелектуальною обробкою даних. Особливістю запропонованої моделі є застосування комбінованої формули для розрахунку коефіцієнта *Ki*, яка враховує як поточний бал, так і історію навчальних спроб, забезпечуючи стійкість показника до випадкових відхилень і даючи змогу оцінювати реальну динаміку знань студента.

Запропонована архітектура на основі клієнтського Java-додатка, автономного аналітичного модуля й локального сховища SQLite довела свою ефективність та відмовостійкість під час тестування на різних платформах. Реалізовані модулі TestEngine, KiCalculator, RecommendationEngine та NaiveBayesCourseClassifier продемонстрували коректність роботи та відповідність поставленим вимогам. Логічна модель даних (фактова таблиця *Results_Fact* та вимірні таблиці *Course_Dim*, *Group_Dim*, *Date_Dim*) забезпечила структуроване зберігання навчальних метрик і дала можливість будувати OLAP-запити та KPI-індикатори для подальшої аналітики.

Під час тестування системи перевірено функціональну повноту модулів, стабільність клієнтського інтерфейсу, коректність алгоритмів оцінювання та

класифікації, а також відповідність результатів критеріям якості. Отримані KPI-показники підтвердили, що система здатна точно відображати рівень знань студентів, прогнозувати тенденції, формувати персоналізовані рекомендації та підтримувати процеси прийняття рішень викладачем або адміністратором освітнього процесу.

У результаті виконання роботи створено інтелектуальну навчальну систему, яка поєднує засоби адаптивного тестування, аналітики, візуалізації прогресу та рекомендаційного модулю, що робить її придатною для використання у навчальному процесі закладів освіти. Запропоновані методи й програмні рішення можуть бути масштабовані, доповнені механізмами прогнозування успішності за допомогою сучасних методів машинного навчання, а також інтегровані з існуючими інформаційними системами університетів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IEEE Learning Technology Standards Committee. Draft Standard for Learning Technology—Public and Private Information (PAPI) for Learners (PAPI Learner). IEEE, 2020.

2. Graesser, A.C., Chipman, P., Haynes, B., Olney, A. AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. IEEE Transactions on Education, 2005.

3. Методичні рекомендації щодо виконання кваліфікаційних робіт магістра. НУБіП України, 2023.

4. Romero, C., Ventura, S. Educational data mining: A review of the state of the art. IEEE Transactions on Systems, Man, and Cybernetics, 2013.

5. Siemens, G., Long, P. Penetrating the Fog: Analytics in Learning and Education. EDUCAUSE Review, vol. 46(5), 2011.

6. Gütl, C., Chang, V. Model for an Adaptive e-Learning System Based on Learning Styles. Springer, 2018.

7. Koedinger, K., Aleven, V. Computer-based tutoring in cognitive tutors: Student modeling and instructional strategies. ACM, 2016.

8. Baker, R.S., Yacef, K. The State of Educational Data Mining in 2020: A Review and Future Visions. Journal of Educational Data Mining, 2020.

9. Heffernan, N., Heffernan, C. The ASSISTments Ecosystem: Building a Platform that Brings Scientists and Teachers Together for Minimally Invasive Research on Human Learning and Teaching. International Journal of AI in Education, 2014.

10. Mitchell, T. Machine Learning. New York: McGraw-Hill, 1997.

11. Murphy, K.P. Probabilistic Machine Learning: An Introduction. MIT Press, 2022.

12. Bishop, C.M. Pattern Recognition and Machine Learning. Springer, 2006.

13. Zhang, Q., Yang, L.T., Chen, Z. Deep Learning for Big Data Analytics: A Survey. Elsevier, 2018.

14. Han, J., Kamber, M., Pei, J. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2012.

15. Tan, P.-N., Steinbach, M., Kumar, V. Introduction to Data Mining. Pearson, 2019.

16. SQLite Consortium. SQLite Documentation. 2024. URL: https://www.sqlite.org/docs.html

17. Oracle. Java SE 17 Documentation. Oracle, 2024. URL: https://docs.oracle.com/javase/17

18. Fowler, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

19. Bass, L., Clements, P., Kazman, R. Software Architecture in Practice. Addison-Wesley, 2021.

20. Pressman, R. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2020.

21. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. Pearson, 2021.

22. Zhang, D., Zhao, J., Zhou, S. Learning Analytics for Smart Education Systems. MDPI Electronics, 2022.

23. GitHub Developers. OAuth 2.0 Specification. IETF RFC 6749, 2023.

24. Kotsiantis, S. Supervised Machine Learning: A Review of Classification Techniques. Informatica Journal, 2007.

25. Powers, D.M.W. Evaluation: Precision, Recall, F-measure, ROC and AUC. Journal of Machine Learning Technologies, 2011.

26. Dash, S., Zhang, H. Adaptive Testing Models Based on Knowledge Tracing. Springer Lecture Notes in Computer Science, 2020.

27. Pahl, C., Donnelly, S. A Framework for Dynamic Student Modelling in Adaptive e-Learning Systems. ACM SIGCSE, 2018.

28. Jolliffe, I. Principal Component Analysis. Springer Series in Statistics, 2011.

29. Jain, A.K. Data Clustering: 50 Years Beyond K-means. Pattern Recognition Letters, 2010.

30. ISO/IEC 25010. Systems and Software Engineering — System Quality Models. International Standard, 2020.

ДОДАТОК А

Основні фрагменти програмного коду інтелектуальної системи підтримки дистанційного навчання та аналітики успішності.

```
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

/**
 * Додаток А.

 */
public class LmsDemo {

    /* ============================
     *  ДОМЕННІ КЛАСИ
     * ============================ */

    public static class User {
        private final int userId;
        private final String username;
        private final String passwordHash;
        private final String role; // "STUDENT", "TEACHER", "ADMIN"
```

```
    public User(int userId, String username, String passwordHash, String role)
{
        this.userId = userId;
        this.username = username;
        this.passwordHash = passwordHash;
        this.role = role;
    }

    public int getUserId() {
        return userId;
    }

    public String getUsername() {
        return username;
    }

    public String getPasswordHash() {
        return passwordHash;
    }

    public String getRole() {
        return role;
    }

    @Override
    public String toString() {
        return userId + ": " + username + " (" + role + ")";
    }
}
```

```
public static class Course {
    private final int courseId;
    private final String title;
    private final String description;

    public Course(int courseId, String title, String description) {
        this.courseId = courseId;
        this.title = title;
        this.description = description;
    }

    public int getCourseId() {
        return courseId;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    @Override
    public String toString() {
        return courseId + ": " + title;
    }
}

/**
```

```
     * Питання тесту з варіантами відповідей.
     */
    public static class Question {
        private final int id;
        private final String text;
        private final List<String> options;
        private final int correctIndex;
        private final double weight;

        public Question(int id, String text, List<String> options, int correctIndex,
double weight) {
            this.id = id;
            this.text = text;
            this.options = new ArrayList<>(options);
            this.correctIndex = correctIndex;
            this.weight = weight;
        }

        public int getId() {
            return id;
        }

        public String getText() {
            return text;
        }

        public List<String> getOptions() {
            return Collections.unmodifiableList(options);
        }
```

```
        public int getCorrectIndex() {
            return correctIndex;
        }

        public double getWeight() {
            return weight;
        }
    }

    /**
     * Результат одного тестування.
     */
    public static class TestResult {
        private final int userId;
        private final int courseId;
        private final double scorePercent;
        private final double ki;
        private final LocalDateTime createdAt;

        public TestResult(int userId, int courseId, double scorePercent, double ki)
{
            this.userId = userId;
            this.courseId = courseId;
            this.scorePercent = scorePercent;
            this.ki = ki;
            this.createdAt = LocalDateTime.now();
        }

        public int getUserId() {
            return userId;
```

```
    }

    public int getCourseId() {
        return courseId;
    }

    public double getScorePercent() {
        return scorePercent;
    }

    public double getKi() {
        return ki;
    }

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    @Override
    public String toString() {
        return "TestResult{userId=" + userId
                + ", courseId=" + courseId
                + ", score=" + scorePercent
                + ", Ki=" + ki
                + ", createdAt=" + createdAt + "}";
    }
}

/**
 * Агрегована статистика курсу (для класифікації).
```

```
 */
public static class CourseStats {
    public final int courseId;
    public final int groupId;
    public final String groupCode;
    public final String title;
    public final double avgRating;
    public final double avgCourseVisit;
    public final char label;  // 'H' або 'L'
    public double probH;      // ймовірність належності до класу H
    public double probL;      // ймовірність належності до класу L

    public CourseStats(int courseId,
                       int groupId,
                       String groupCode,
                       String title,
                       double avgRating,
                       double avgCourseVisit,
                       char label) {
        this.courseId = courseId;
        this.groupId = groupId;
        this.groupCode = groupCode;
        this.title = title;
        this.avgRating = avgRating;
        this.avgCourseVisit = avgCourseVisit;
        this.label = label;
    }

    @Override
    public String toString() {
```

```
            return "CourseStats{" +
                    "courseId=" + courseId +
                    ", groupCode='" + groupCode + '\'' +
                    ", title='" + title + '\'' +
                    ", avgRating=" + avgRating +
                    ", avgCourseVisit=" + avgCourseVisit +
                    ", label=" + label +
                    ", probH=" + probH +
                    ", probL=" + probL +
                    '}';
        }
    }

    /* ============================
     *  ІНТЕРФЕЙСИ РЕПОЗИТОРІЇВ
     * ============================ */

    public interface UserRepository {
        User findByUsernameAndPassword(String username, String passwordPlain);
    }

    public interface CourseRepository {
        List<Course> findCoursesForUser(int userId);
    }

    public interface QuestionRepository {
        List<Question> findQuestionsForCourse(int courseId);
    }
```

```
public interface TestResultRepository {
    void save(TestResult result);

    /**
     * Повертає останні N відсоткових результатів користувача.
     */
    List<Double> findLastScoresForUser(int userId, int limit);
}

public interface CourseStatsRepository {
    List<CourseStats> loadTrainingSet();

    List<CourseStats> loadCurrentStats();
}

/* ====================================
 *  ПРОСТІ IN-MEMORY РЕАЛІЗАЦІЇ (DEMO)
 * ==================================== */

/**
 * Демонстраційна реалізація UserRepository без БД.
 */
public static class InMemoryUserRepository implements UserRepository {
    private final List<User> users = new ArrayList<>();

    public InMemoryUserRepository() {
        // demo-користувач student / password
        users.add(new User(1, "student", "password", "STUDENT"));
    }
```

```
        @Override
        public User findByUsernameAndPassword(String username, String passwordPlain) {
            return users.stream()
                    .filter(u -> u.getUsername().equals(username)
                            && u.getPasswordHash().equals(passwordPlain))
                    .findFirst()
                    .orElse(null);
        }
    }

    public static class InMemoryCourseRepository implements CourseRepository {
        private final List<Course> courses = new ArrayList<>();

        public InMemoryCourseRepository() {
            courses.add(new Course(1, "Інтелектуальні системи дистанційного навчання",
                    "Курс про адаптивні e-learning системи та аналітику."));
            courses.add(new Course(2, "Об’єктно-орієнтоване програмування (Java)",
                    "Базові принципи ООП та розробка застосунків Java."));
        }

        @Override
        public List<Course> findCoursesForUser(int userId) {
            return new ArrayList<>(courses);
        }
    }
```

```
public static class InMemoryQuestionRepository implements QuestionRepository {

    @Override
    public List<Question> findQuestionsForCourse(int courseId) {
      List<Question> questions = new ArrayList<>();
      if (courseId == 1) {
        questions.add(new Question(
            1,
            "Що таке інтелектуальна навчальна система?",
            List.of(
                "Система з електронними конспектами",
                "Система, що адаптує контент під рівень знань студента",
                "Будь-який веб-сайт з навчальними матеріалами",
                "Лише платформа відеолекцій"
            ),
            1,
            1.0
        ));
        questions.add(new Question(
            2,
            "Які дані найчастіше використовуються для розрахунку коефіцієнта знань Ki?",
            List.of(
                "Лише підсумкова оцінка",
                "Лише відвідуваність занять",
                "Результати тестів та історія відповідей",
                "Тільки рейтинг викладача"
            ),
```

```
                    2,
                    1.0
                ));
            }
            return questions;
        }
    }

    public static class InMemoryTestResultRepository implements
TestResultRepository {
        private final List<TestResult> storage = new ArrayList<>();

        @Override
        public void save(TestResult result) {
            storage.add(result);
        }

        @Override
        public List<Double> findLastScoresForUser(int userId, int limit) {
            List<Double> scores = new ArrayList<>();
            for (int i = storage.size() - 1; i >= 0 && scores.size() < limit; i--) {
                TestResult r = storage.get(i);
                if (r.getUserId() == userId) {
                    scores.add(r.getScorePercent());
                }
            }
            return scores;
        }
    }
```

```
        /* ============================
         *  СЕРВІСИ АВТЕНТИФІКАЦІЇ ТА КУРСІВ
         * ============================ */

        public static class AuthService {
            private final UserRepository userRepository;

            public AuthService(UserRepository userRepository) {
                this.userRepository = userRepository;
            }

            public User authenticate(String username, String passwordPlain) {
                return userRepository.findByUsernameAndPassword(username,
passwordPlain);
            }
        }

        public static class CourseService {
            private final CourseRepository courseRepository;

            public CourseService(CourseRepository courseRepository) {
                this.courseRepository = courseRepository;
            }

            public List<Course> getCoursesForUser(User user) {
                return courseRepository.findCoursesForUser(user.getUserId());
            }
        }

        /* ============================
```

```
 *  МОДУЛЬ ОБЧИСЛЕННЯ КОЕФІЦІЄНТА Ki
 * ========================== */

public static class KiCalculator {

    /**
     * Простий приклад: Ki – згладжена комбінація поточного результату
     * та середнього значення попередніх спроб.
     *
     * @param currentPercent відсоток за поточний тест
     * @param history        список попередніх відсотків (може бути порожнім)
     */
    public double computeKi(double currentPercent, List<Double> history) {
        double historyAverage = 0.0;
        if (history != null && !history.isEmpty()) {
            double sum = 0.0;
            for (double v : history) {
                sum += v;
            }
            historyAverage = sum / history.size();
        }

        // ваги можна підібрати експериментально
        double alpha = 0.7; // вага поточного тесту
        double beta = 1.0 - alpha;

        return alpha * currentPercent + beta * historyAverage;
    }
}
```

```
/* ============================
 *  МОДУЛЬ ТЕСТУВАННЯ
 * ============================ */

public static class TestEngine {
   private final TestResultRepository resultRepository;
   private final KiCalculator kiCalculator;

   public TestEngine(TestResultRepository resultRepository, KiCalculator
kiCalculator) {
      this.resultRepository = resultRepository;
      this.kiCalculator = kiCalculator;
   }

   /**
    * Виконання тесту для заданого користувача та курсу.
    *
    * @param userId             ідентифікатор користувача
    * @param courseId           ідентифікатор курсу
    * @param questions          список питань тесту
    * @param answers            індекси обраних відповідей
    * @param passThresholdPercent пороговий відсоток зарахування
    */
   public TestResult runTest(int userId,
                             int courseId,
                             List<Question> questions,
                             int[] answers,
                             double passThresholdPercent) {
```

```
        double score = 0.0;
        double maxScore = 0.0;

        for (int i = 0; i < questions.size(); i++) {
            Question q = questions.get(i);
            maxScore += q.getWeight();
            if (answers[i] == q.getCorrectIndex()) {
                score += q.getWeight();
            }
        }

        double percent = (maxScore > 0.0) ? (score / maxScore) * 100.0 : 0.0;

        // історія останніх 5 результатів
        List<Double> history = resultRepository.findLastScoresForUser(userId,
5);
        double ki = kiCalculator.computeKi(percent, history);

        TestResult result = new TestResult(userId, courseId, percent, ki);
        resultRepository.save(result);

        boolean passed = percent >= passThresholdPercent;
        System.out.println("Результат тесту: " + percent + "%, Ki=" + ki
                + ", status=" + (passed ? "PASSED" : "FAILED"));

        return result;
    }
}

/* ========================
```

```
 *  НАЇВНИЙ БЄЄСІВСЬКИЙ КЛАСИФІКАТОР КУРСІВ
 * ========================== */

public static class NaiveBayesCourseClassifier {

    private static class GaussianParams {
        double meanRating;
        double stdRating;
        double meanVisit;
        double stdVisit;
    }

    private GaussianParams paramsH;
    private GaussianParams paramsL;
    private double priorH;
    private double priorL;

    /**
     * Навчання класифікатора на агрегованій вибірці курсів.
     */
    public void train(List<CourseStats> trainingSet) {
        List<CourseStats> high = new ArrayList<>();
        List<CourseStats> low = new ArrayList<>();

        for (CourseStats s : trainingSet) {
            if (s.label == 'H') {
                high.add(s);
            } else if (s.label == 'L') {
                low.add(s);
            }
```

```
        }

        int n = trainingSet.size();
        priorH = high.size() / (double) n;
        priorL = low.size() / (double) n;

        paramsH = estimateGaussian(high);
        paramsL = estimateGaussian(low);
    }

    private GaussianParams estimateGaussian(List<CourseStats> data) {
        GaussianParams p = new GaussianParams();
        if (data == null || data.isEmpty()) {
            return p;
        }

        double sumR = 0.0, sumV = 0.0;
        for (CourseStats s : data) {
            sumR += s.avgRating;
            sumV += s.avgCourseVisit;
        }
        double meanR = sumR / data.size();
        double meanV = sumV / data.size();

        double varR = 0.0, varV = 0.0;
        for (CourseStats s : data) {
            varR += Math.pow(s.avgRating - meanR, 2);
            varV += Math.pow(s.avgCourseVisit - meanV, 2);
        }
        varR /= data.size();
```

```
        varV /= data.size();

        p.meanRating = meanR;
        p.stdRating = Math.sqrt(varR == 0 ? 1e-6 : varR);
        p.meanVisit = meanV;
        p.stdVisit = Math.sqrt(varV == 0 ? 1e-6 : varV);
        return p;
    }

    private double gaussianPdf(double x, double mean, double std) {
        double z = (x - mean) / std;
        return (1.0 / (std * Math.sqrt(2 * Math.PI))) * Math.exp(-0.5 * z * z);
    }

    /**
     * Класифікація окремого курсу на класи H / L.
     */
    public CourseStats classify(CourseStats s) {
        double pX_H = gaussianPdf(s.avgRating, paramsH.meanRating,
paramsH.stdRating)
                * gaussianPdf(s.avgCourseVisit, paramsH.meanVisit,
paramsH.stdVisit);

        double pX_L = gaussianPdf(s.avgRating, paramsL.meanRating,
paramsL.stdRating)
                * gaussianPdf(s.avgCourseVisit, paramsL.meanVisit,
paramsL.stdVisit);

        double postH = pX_H * priorH;
        double postL = pX_L * priorL;
```

```
            double sum = postH + postL;
            if (sum == 0.0) sum = 1e-6;

            s.probH = postH / sum;
            s.probL = postL / sum;
            return s;
        }
    }

    /* ============================
     *  ПРОСТИЙ ПРИКЛАД ВИКОРИСТАННЯ
     * ============================ */

    public static void main(String[] args) {
        // 1. Аутентифікація
        UserRepository userRepo = new InMemoryUserRepository();
        AuthService authService = new AuthService(userRepo);

        User user = authService.authenticate("student", "password");
        if (user == null) {
            System.out.println("Невірні облікові дані");
            return;
        }
        System.out.println("Вхід успішний: " + user.getUsername());

        // 2. Завантаження курсів
        CourseService courseService = new CourseService(new
InMemoryCourseRepository());
        List<Course> courses = courseService.getCoursesForUser(user);
        System.out.println("Доступні курси:");
```

```
            for (Course c : courses) {
                System.out.println(" - " + c);
            }

            // Для демонстрації — тестуємо перший курс
            Course selectedCourse = courses.get(0);
            QuestionRepository questionRepo = new InMemoryQuestionRepository();
            List<Question> questions =
questionRepo.findQuestionsForCourse(selectedCourse.getCourseId());

            // Відповіді користувача (демо: всі правильні)
            int[] answers = new int[questions.size()];
            for (int i = 0; i < questions.size(); i++) {
                answers[i] = questions.get(i).getCorrectIndex();
            }

            // 3. Запуск TestEngine
            TestResultRepository testResultRepo = new
InMemoryTestResultRepository();
            KiCalculator kiCalculator = new KiCalculator();
            TestEngine testEngine = new TestEngine(testResultRepo, kiCalculator);

            TestResult result = testEngine.runTest(
                    user.getUserId(),
                    selectedCourse.getCourseId(),
                    questions,
                    answers,
                    60.0
            );
```

```
        System.out.println("Збережений результат: " + result);
    }
}
```