МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук, доцент

_____ Євген ДАВИДЕНКО

«___» _____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ЗАСТОСУНОК АНАЛІЗУ УСПІШНОСТІ ЗДОБУВАЧІВ ОСВІТИ З ПОДАЛЬШИМ ФОРМУВАННЯМ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Марія ДАНИЛЕНКО
«__» _____ 2025 р.

Керівник канд. техн. наук, доцент _____Євген ДАВИДЕНКО
«__» _____ 2025 р.

Миколаїв 2025

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступінь	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«____« _____ 2025 р.

ЗАВДАННЯ

на кваліфікаційну магістерську роботу здобувача вищої освіти

Даниленко Марії

1. Тема кваліфікаційної роботи «Застосунок аналізу успішності здобувачів освіти з подальшим формуванням персоналізованих рекомендацій» затверджена наказом ректора ЧНУ ім. Петра Могили № 182 від «2» липня 2025 р.
2. Строк представлення кваліфікаційної роботи «22» грудня 2025 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом роботи є вебзастосунок для аналізу успішності здобувачів освіти та формування персоналізованих рекомендацій, який забезпечує створення й проходження тестів, збирання та обробку результатів

навчання, а також автоматизовану генерацію індивідуальних порад на основі алгоритмів машинного навчання.

4. Перелік питань, що підлягають розробці:

− проведення аналізу предметної області та існуючих систем оцінювання успішності;

− дослідження методів аналізу навчальних даних та формування персоналізованих рекомендацій;

− визначення критеріїв успішності здобувачів освіти та ключових факторів для аналізу їхніх тестових результатів;

− обрання та обґрунтування методів обробки даних для персоналізації освітньої траєкторії;

− розробка математичної моделі аналізу успішності та алгоритмів формування персоналізованих рекомендацій;

− інтеграція розробленої моделі в існуючий вебзастосунок тестування та аналізу навчальних результатів.

5. Перелік графічних матеріалів:

презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Застосунок аналізу успішності здобувачів освіти з подальшим формуванням персоналізованих рекомендацій

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	02.07.2025	02.07.2025	Виконано
2.	Огляд літератури за темою роботи	01.09.2025	05.09.2025	Виконано
3.	Складання календарного плану КМР	08.09.2025	09.09.2025	Виконано
4.	Аналіз предметної області	10.09.2025	12.09.2025	Виконано
5.	Розробка проєктних рішень	15.09.2025	19.09.2025	Виконано
6.	Моделювання та конструювання ПЗ	22.09.2025	17.10.2025	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	20.10.2025	14.11.2025	Виконано
8.	Відгук керівника КМР	17.11.2025	19.11.2025	Виконано
9.	Оформлення КМР та презентації	19.11.2025	21.11.2025	Виконано
10.	Попередній захист	24.11.2025	24.11.2025	Виконано
11.	Рецензування	25.11.2025	02.12.2025	Виконано
12.	Завершення оформлення КМР та презентації	05.12.2025	12.12.2025	Виконано
13.	Захист кваліфікаційної роботи	22.12.2025	22.12.2025	Виконано

Здобувач _____ **Марія ДАНИЛЕНКО**

«__» _____ 20__ р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

Застосунок аналізу успішності здобувачів освіти з подальшим формуванням персоналізованих рекомендацій

Здобувачка 608М гр.: Даниленко Марія

Керівник: канд. техн. наук, доцент Євген Давиденко

У сучасній системі освіти зростає потреба в інструментах, які здатні не лише оцінювати навчальні досягнення здобувачів освіти, а й аналізувати індивідуальні результати для формування персоналізованих рекомендацій. Традиційні вебплатформи для тестування обмежені статичною перевіркою знань та не забезпечують глибокої аналітики, що знижує ефективність навчального процесу. Розробка інтелектуальних систем підтримки навчання із застосуванням алгоритмів машинного навчання дає змогу автоматизувати аналіз успішності та надавати здобувачам освіти персоналізовані поради для покращення результатів.

Об’єкт дослідження − процес аналізу успішності здобувачів освіти та визначення їх індивідуальної навчальної траєкторії.

Предмет дослідження − метод та алгоритм аналізу успішності здобувачів освіти для формування персоналізованих навчальних рекомендацій.

Мета дослідження −формування персоналізованих рекомендацій з метою визначення індивідуального навчального вектора шляхом розробки та впровадження системи аналізу успішності здобувачів освіти.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність теми, визначено об’єкт, предмет, мету та завдання дослідження.

У першому розділі проведено аналіз сучасних вебплатформ для оцінювання навчальних досягнень і персоналізованого навчання, визначено їх

переваги та недоліки, а також виконано аналіз системи, що розробляється, із формуванням вимог до її функціональності та архітектури.

Другий розділ присвячено проєктуванню системи: виконано моделювання функцій та інформаційних потоків за допомогою діаграм IDEF0 та DFD, а також розроблено математичне та алгоритмічне забезпечення. Детально обґрунтовано вибір методу машинного навчання − дерева рішень, наведено його математичну модель та описано логіку побудови алгоритму для аналізу навчальної успішності й формування рекомендацій.

Третій розділ присвячено проєктуванню архітектури та вибору технологічних засобів реалізації вебзастосунку аналізу успішності здобувачів освіти. Описано загальну структуру системи та взаємодію її основних компонентів на основі UML-діаграм, зокрема діаграми компонентів, варіантів використання та послідовності. Обґрунтовано вибір стеку технологій для серверної та клієнтської частин, а також інструментів реалізації модуля машинного навчання, що забезпечують стабільну роботу системи, масштабованість і ефективну обробку навчальних даних.

Четвертий розділ присвячено практичній реалізації вебзастосунку та модуля аналізу навчальних результатів. Розглянуто особливості впровадження алгоритму дерева рішень у процес обробки даних, описано логіку його інтеграції з інформаційною системою та процес формування персоналізованих рекомендацій. Проаналізовано результати роботи розробленого модуля, наведено приклади його застосування та оцінено ефективність використання машинного навчання для підтримки індивідуальної навчальної траєкторії здобувачів освіти.

У висновках підбито підсумки проведеного дослідження та підтверджено досягнення поставленої мети роботи.

Кваліфікаційна робота викладена на 105 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 30 найменувань та 1 додаток. Праця містить 7 таблиць та 19 рисунків.

Ключові слова: оцінювання навчальних досягнень, персоналізовані рекомендації, машинне навчання, дерево рішень, освітня аналітика, вебзастосунок.

ABSTRACT

to the qualifying master's thesis

Web Application for the Analysis of Students' Academic Performance and the Generation of Personalized Recommendations

Student of 608M group: Danylenko Mariia

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor Davydenko Yevhen

In the modern educational system, there is an increasing demand for tools that can not only assess students' academic achievements but also analyze individual results to provide personalized recommendations. Traditional web platforms for testing are limited to static knowledge assessment and do not offer deep analytics, which reduces the effectiveness of the learning process. The development of intelligent learning support systems using machine learning algorithms makes it possible to automate performance analysis and provide students with personalized advice to improve their results.

Object of the research – the process of analyzing students' academic performance and determining their individual learning trajectory.

Subject of the research – the method and algorithm for analyzing students' academic performance to generate personalized learning recommendations.

Purpose of the research – to develop and implement a system for analyzing students' academic performance in order to generate personalized recommendations and define individual learning paths.

The qualification work consists of an introduction, 4 chapters, conclusions and a list of references.

The introduction substantiates the relevance of the topic, defines the object, subject, aim, and research tasks.

The first chapter analyzes modern web platforms for assessing academic performance and supporting personalized learning, identifies their strengths and

weaknesses, and presents the analysis of the system under development with the formulation of requirements for its functionality and architecture.

The second chapter is dedicated to system design: it includes functional and data flow modeling using IDEF0 and DFD diagrams, as well as the development of mathematical and algorithmic support. The choice of the decision tree method for machine learning is justified in detail, its mathematical model is presented, and the logic of building the algorithm for academic performance analysis and recommendation generation is described.

The third chapter is devoted to designing the architecture and selecting technological tools for implementing a web application for analysing the academic performance of learners. The overall structure of the system and the interaction of its main components are described based on UML diagrams, in particular component, use case, and sequence diagrams. The choice of the technology stack for the server and client parts, as well as the tools for implementing the machine learning module, is substantiated, ensuring stable system operation, scalability, and efficient processing of educational data.

The fourth chapter focuses on the practical implementation of the web application and the module for analysing learning outcomes. The features of implementing the decision tree algorithm in the data processing workflow are considered, the logic of its integration with the information system is described, and the process of generating personalised recommendations is outlined. The results of the developed module are analysed, examples of its application are provided, and the effectiveness of using machine learning to support the individual learning trajectory of learners is evaluated.

The conclusions summarize the conducted research and confirm the achievement of the stated goal.

The qualification work is presented on 105 pages of typewritten text, consists of an introduction, 4 chapters, general conclusions, a list of references with 30 titles and 1 appendix. The work contains 7 tables and 19 figures.

Keywords: academic performance assessment, personalized recommendations, machine learning, decision tree, educational analytics, web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
ІТ	−	інформаційні технології
ОС	–	операційна система
ПЗ	–	програмне забезпечення

AI	—	artificial intelligence
API	–	application programming interface
CART	—	classification and regression trees
CRUD	–	create, read, update, delete
DOM	−	document object model
LINQ	−	language-integrated query
LMS	—	learning management system
LTI	—	learning tools interoperability
ML	−	machine learning
MS	−	Microsoft
REST	−	representational state transfer
WCAG	—	web content accessibility guidelines
xAPI	—	experience API

ВСТУП

Сучасний розвиток інформаційних технологій і цифровізація освіти кардинально змінюють підходи до навчання. Пандемія COVID-19 разом із поточними соціально-політичними викликами, зокрема повномасштабним вторгненням в Україну, спричинили масовий перехід до дистанційних форматів навчання. Особливо гостро ця проблема проявляється у зв’язку з воєнними подіями, коли здобувачі освіти знаходяться не лише по всій території України, а й у різних куточках світу, об'єднуючись в єдині навчальні групи. У таких умовах вчитель, який змушений одночасно працювати з 90 і більше здобувачами освіти, фізично не може відслідковувати індивідуальні особливості кожного учня, що негативно впливає на якість навчального процесу.

Нестача персоналізованого підходу створює труднощі при формуванні оптимальної навчальної траєкторії, оскільки кожен здобувач освіти має унікальні здібності, потреби та мотиваційні особливості. Саме тому дослідження, спрямоване на аналіз успішності здобувачів освіти та формування персоналізованих рекомендацій для визначення індивідуального навчального вектора, є надзвичайно актуальним. Результати цього дослідження можуть стати базою для модернізації сучасних освітніх систем, сприяти розвитку адаптивних методик навчання та підтримувати процес цифрової трансформації освіти як в Україні, так і за її межами.

Об’єкт дослідження: процес аналізу успішності здобувачів освіти та визначення їх індивідуальної навчальної траєкторії.

Предмет дослідження: метод та алгоритм аналізу успішності здобувачів освіти для формування персоналізованих навчальних рекомендацій.

Мета дослідження: формування персоналізованих рекомендацій з метою визначення індивідуального навчального вектора шляхом розробки та впровадження системи аналізу успішності здобувачів освіти.

Для досягнення зазначеної мети необхідно виконати наступні **завдання**:

− провести аналіз предметної області та існуючих систем оцінювання успішності;

− дослідити метод аналізу навчальних даних та формування персоналізованих рекомендацій;

− визначити критерії успішності здобувачів освіти та ключові фактори аналізу їхніх тестових результатів;

− обрати методи обробки даних для персоналізації освітньої траєкторії;

− розробити програмну модель аналізу успішності здобувачів освіти та алгоритми формування персоналізованих рекомендацій;

− інтегрувати розроблену модель в наявну систему тестування.

Апробація результатів КМР відбулась під час XXVIII Всеукраїнської щорічної науково–практичної конференції «Могилянські читання – 2025», Миколаїв, 10–14 листопада, 2025 року (Додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні тенденції розвитку аналогічних інформаційних систем

Сучасна цифровізація освіти потребує інструментів, здатних не лише фіксувати результати навчання, а й здійснювати їх комплексний аналіз для підтримки індивідуального розвитку кожного здобувача освіти. В умовах зростання обсягів навчального контенту, ускладнення освітніх програм та варіативності форм навчання особливої ваги набуває ефективне використання освітніх даних для об’єктивного оцінювання рівня підготовки студентів, виявлення прогалин у знаннях і прогнозування подальших результатів навчання. У центрі цієї роботи перебуває процес аналізу успішності здобувачів освіти та визначення їхньої індивідуальної навчальної траєкторії, а також методи й алгоритми, що дозволяють на основі накопичених даних формувати персоналізовані рекомендації, спрямовані на оптимізацію навчального процесу та підвищення його результативності. Усвідомлення того, як еволюціонують сучасні освітні платформи, їх функціональні можливості та підходи до обробки навчальних даних, є необхідним для обґрунтування вимог до майбутнього рішення, формування його концепції та визначення напрямів подальшого удосконалення.

У відповідь на зростаючі вимоги до якості освітнього процесу та необхідність більш гнучкого підходу до навчання відбувається поступовий перехід від традиційних інструментів тестування до комплексних вебзастосунків, що поєднують функції оцінювання, навчальну аналітику та засоби підтримки індивідуальних освітніх траєкторій. Такі системи орієнтуються не лише на фіксацію результатів, а й на їх глибоку інтерпретацію, що дозволяє враховувати особливості навчальної поведінки кожного здобувача освіти. Головним трендом розвитку стає персоналізація навчання, за якої освітні платформи

використовують алгоритмічні методи для формування цифрових профілів студентів, аналізу їх навчальної активності та прогнозування рівня засвоєння матеріалу. На основі цих даних системи здатні автоматично адаптувати зміст навчання, пропонувати індивідуальні рекомендації щодо повторення складних тем, коригувати складність завдань і визначати оптимальний темп опрацювання матеріалу. Такий підхід сприяє підвищенню мотивації здобувачів освіти, більш ефективному засвоєнню знань і формуванню умов для побудови персоналізованого освітнього маршруту відповідно до реального рівня підготовки та навчальних потреб кожного користувача.

Важливим напрямом є навчальна аналітика та раннє виявлення ризиків. Сучасні платформи збирають події взаємодії користувачів із контентом, результати оцінювань і історію спроб, що дозволяє будувати прогностичні моделі «at-risk» і вчасно пропонувати коригувальні дії. Паралельно поширюються компетентнісний підхід та майстерне навчання (mastery learning): системи відстежують досягнення за окремими результатами, формують індивідуальні «дорожні карти» закриття прогалин і підтримують адаптивне тестування зі зміною складності завдань у реальному часі [1].

З погляду користувацького досвіду домінують mobile-first-парадигма, мікронавчання, офлайн-режим, push-сповіщення та гейміфікація (бейджі, рівні, челенджі). Все більшої ваги набувають інтероперабельність (інтеграції через стандарти LTI та xAPI), безпека персональних даних та доступність інтерфейсів згідно з WCAG. Архітектурно переважає модульний підхід: фронтенд і бекенд відокремлюються, аналітика винесена в окремі сервіси з можливістю оновлення моделей без зупинки основної системи [2].

Узагальнюючи, освітні вебплатформи рухаються від статичного оцінювання до інтелектуальних систем підтримки навчання, де персоналізація, аналітика й інтеграція з іншими сервісами стають базовими вимогами. Це

обґрунтовує доцільність розробки рішення, здатного автоматично аналізувати успішність та надавати персоналізовані рекомендації здобувачам освіти.

1.2 Особливості аналогічних інформаційних систем

Аналіз предметної області показує, що сучасні інформаційні системи для підтримки навчального процесу вже давно вийшли за межі простих платформ тестування. Вони поєднують збір, зберігання та аналіз навчальних даних із можливістю персоналізації освітньої траєкторії користувачів. Типовими компонентами таких систем є вебінтерфейс для викладачів та учнів, серверна частина з бізнес-логікою, база даних для накопичення результатів і аналітичний модуль, який опрацьовує статистику взаємодії студентів із навчальним контентом.

У більшості рішень застосовується рольова модель доступу:

− викладачі створюють навчальні матеріали, тести та контролюють оцінювання;

− учні виконують завдання, переглядають свої результати та отримують зворотний зв’язок;

− адміністратори керують користувацькими правами та інтеграціями з іншими освітніми середовищами.

Важливим елементом таких платформ є навчальна аналітика (learning analytics), яка виступає основою для прийняття обґрунтованих управлінських і педагогічних рішень. Системи накопичують і систематизують різнорідні дані про результати тестувань, рівень активності користувачів, кількість спроб виконання завдань, середні бали, часові показники, історію навчальної взаємодії та динаміку успішності. На основі цих даних формується аналітичне середовище, що дозволяє відстежувати прогрес здобувачів освіти, виявляти проблемні теми, визначати зони ризику та оцінювати ефективність навчальних матеріалів і

методик. Отримана інформація використовується не лише для підготовки зведених звітів і контролю загальної динаміки навчання групи або окремого студента, а й для підтримки персоналізованого підходу, зокрема шляхом адаптації навчального контенту, рекомендацій щодо повторення матеріалу та корекції індивідуальної освітньої траєкторії.

Все більшого поширення набувають адаптивні механізми, які трансформують освітні платформи з пасивних засобів фіксації результатів у активні інструменти супроводу навчального процесу. Такі системи здатні динамічно змінювати логіку подання навчального матеріалу, коригувати послідовність завдань, регулювати рівень їх складності та пропонувати індивідуальні траєкторії опанування тем залежно від поведінки й поточних досягнень користувача. Окрім цього, адаптивні алгоритми можуть автоматично визначати студентів, які демонструють нестабільну або знижену успішність, та ініціювати додаткові навчальні впливи у вигляді підказок, повторних пояснень або альтернативних матеріалів. Водночас у багатьох існуючих рішеннях такі можливості реалізовані на основі шаблонних правил і спрощених логічних умов, що обмежує глибину аналізу та не дозволяє повною мірою враховувати індивідуальні особливості навчальної поведінки здобувачів освіти.

Архітектурно сучасні освітні платформи орієнтуються на модульний принцип побудови, за якого окремі функціональні компоненти системи розвиваються незалежно один від одного. Клієнтська та серверна частини реалізуються як автономні підсистеми, що взаємодіють через стандартизовані інтерфейси та API, а аналітичні сервіси, включаючи модулі машинного навчання, виділяються в окремі логічні або фізичні компоненти з власним циклом оновлення моделей. Така структура сприяє гнучкому масштабуванню обчислювальних ресурсів відповідно до навантаження, спрощує внесення змін до

окремих підсистем та забезпечує стабільну роботу платформи навіть у разі часткових збоїв.

Водночас важливе значення має здатність до взаємодії систем (interoperability), яка забезпечує інтеграцію з іншими освітніми середовищами, системами управління навчанням та цифровими платформами. Підтримка стандартів LTI та xAPI дозволяє обмінюватися навчальними даними, результатами оцінювання та подіями користувачів між різними системами, формуючи єдиний інформаційний простір. Використання уніфікованих механізмів автентифікації та авторизації сприяє централізованому управлінню доступом, підвищує безпеку та забезпечує цілісність даних в умовах розподіленого середовища функціонування.

Попри наявність функцій персоналізації та елементів аналітики, значна частина сучасних освітніх платформ обмежується поверхневим підходом до аналізу навчальних досягнень і не забезпечує достатньої глибини інтерпретації індивідуальних результатів користувачів. У більшості випадків рекомендаційні механізми базуються на простих правилах або статистичних залежностях, що не враховують комплексний характер навчального процесу, динаміку змін показників успішності, рівень засвоєння окремих тем та індивідуальні особливості здобувачів освіти. Відсутність можливості інтеграції власних моделей аналізу даних або адаптації алгоритмів під конкретні освітні цілі значно знижує ефективність таких систем у контексті підтримки персоналізованого навчання.

Це зумовлює потребу у створенні та впровадженні більш гнучких програмних рішень, орієнтованих на використання сучасних методів машинного навчання та інтелектуального аналізу даних. Такі системи повинні забезпечувати комплексну обробку навчальної інформації, враховувати багатовимірні параметри успішності та формувати рекомендації, адаптовані до поточної

освітньої траєкторії здобувача освіти. Реалізація подібного підходу сприяє підвищенню обґрунтованості педагогічних рішень, оптимізації навчального процесу та створенню умов для більш ефективного розвитку кожного учня відповідно до його індивідуальних навчальних потреб.

1.3 Огляд сучасних підходів та аналогів

Для визначення вимог до розроблюваної системи та формулювання напрямів її вдосконалення було проведено аналіз сучасних вебплатформ, що підтримують оцінювання навчальних досягнень та персоналізоване навчання. Серед них обрано три найбільш показові приклади – Docebo [3], Fishtree [4] та Edmodo [5], які мають широке розповсюдження та демонструють ключові тенденції у цій сфері.

Система Docebo є комерційною платформою для управління навчанням (LMS), що підтримує персоналізацію контенту та базові рекомендаційні механізми. Основний фокус платформи − корпоративне навчання та підвищення кваліфікації працівників. Вона активно використовується в міжнародних компаніях для організації навчальних програм, адаптації нових співробітників, оцінювання рівня їхніх знань і відстеження професійного розвитку.

Платформа поєднує інструменти створення курсів, тестування та аналітики з елементами штучного інтелекту, що дозволяють автоматизувати процес підбору навчальних матеріалів відповідно до поведінки користувача. Docebo підтримує побудову індивідуальних траєкторій навчання, однак ці механізми орієнтовані переважно на оптимізацію корпоративного навчального процесу, а не на комплексний педагогічний аналіз навчальної успішності в контексті формальної освіти. Архітектура платформи є закритою, що обмежує можливості впровадження власних алгоритмів аналізу даних і гнучкої адаптації системи до специфічних освітніх потреб, зокрема інтеграції альтернативних моделей

машинного навчання. Попри це, Docebo демонструє високий рівень стабільності, масштабованості та функціональної зрілості, що робить її показовим прикладом сучасної LMS-платформи, орієнтованої на автоматизацію та персоналізацію навчання.

Таблиця 1.1 – Характеристики платформи «Docebo»

Назва	Docebo
Виробник	Docebo S.p.A.
Архітектура	Web application
Мова реалізації	PHP, JavaScript
Функції	1) управління навчальними курсами; 2) призначення навчальних модулів користувачам; 3) базова персоналізація контенту; 4) аналітика відвідуваності та прогресу; 5) інтеграції з іншими LMS через API;
Переваги	1) сучасний інтерфейс; 2) наявність інтегрованого AI для рекомендацій; 3) можливість масштабування для великих організацій;
Недоліки	1) орієнтація на корпоративний сегмент, а не на формальну освіту; 2) обмежена гнучкість алгоритмів рекомендацій; 3) закритий код та платна ліцензія;
Джерело інформації	https://www.docebo.com

Fishtree позиціонується як платформа адаптивного навчання, яка використовує аналітику та машинне навчання для підбору навчального матеріалу відповідно до рівня підготовки користувача. Основний акцент зроблено на

автоматизації процесу персоналізації для шкіл та університетів, що дозволяє викладачам ефективніше організовувати навчальний процес і враховувати індивідуальні потреби здобувачів освіти.

Платформа орієнтована на формування індивідуальних освітніх траєкторій шляхом аналізу результатів тестування, навчальної активності та поведінкових характеристик студентів. На основі зібраних даних Fishtree динамічно адаптує навчальний контент, підбираючи матеріали відповідно до поточного рівня знань, темпу засвоєння та навчальних прогалин користувача. Це створює умови для більш гнучкого й персоналізованого навчання порівняно з традиційними системами. Водночас архітектура платформи залишається закритою, що ускладнює інтеграцію сторонніх алгоритмів аналізу даних і обмежує можливості впровадження власних моделей машинного навчання. Наявні рекомендаційні механізми мають обмежену гнучкість налаштування та не завжди дозволяють враховувати специфіку локальних освітніх стандартів і навчальних програм. Це знижує універсальність платформи в контексті її використання для глибокого аналізу успішності та побудови індивідуальної навчальної стратегії.

Попри зазначені обмеження, Fishtree є показовим прикладом сучасної адаптивної освітньої платформи, що демонструє практичне застосування аналітики навчальних даних та алгоритмічних підходів до персоналізації навчання в умовах цифрового освітнього середовища.

Таблиця 1.2 – Характеристики платформи «Fishtree»

Назва	Fishtree
Виробник	Fishtree Inc.
Архітектура	Web application
Мова реалізації	JavaScript, Node.js

Кінець таблиці 1.2

Функції	1) адаптивний вибір навчального контенту; 2) відстеження прогресу студентів; 3) рекомендації матеріалів на основі поведінки; 4) інструменти для викладачів для управління класами;
Переваги	1) інтеграція адаптивного навчання та аналітики; 2) простий інтерфейс для викладачів; 3) готові освітні ресурси;
Недоліки	1) закрита архітектура, складно інтегрувати власні ML-моделі; 2) обмежені можливості кастомізації рекомендацій; 3) слабка підтримка локалізації та роботи з даними вітчизняних освітніх систем;
Джерело інформації	https://www.fishtree.com

Edmodo − це соціально орієнтована платформа для навчання, яка надає можливість викладачам створювати онлайн-класи, ділитися матеріалами та відстежувати прогрес учнів. Платформа поєднує елементи навчального середовища та соціальної мережі, акцентуючи увагу на комунікації, взаємодії та підтримці спільної роботи між учасниками освітнього процесу.

Основною перевагою Edmodo є простота використання та зрозумілий інтерфейс, що робить її доступною для широкого кола користувачів, зокрема в шкільному середовищі. Платформа активно використовується для організації дистанційного навчання, обміну навчальними матеріалами, проведення тестувань і забезпечення зворотного зв’язку між викладачем і учнями. Вона дозволяє створювати віртуальні класи, керувати завданнями та відстежувати

базові показники успішності. Разом з тим, можливості аналітики в Edmodo залишаються обмеженими: система не підтримує складні алгоритми аналізу навчальних даних і не надає інструментів для глибокого прогнозування успішності або формування індивідуальних навчальних траєкторій. Персоналізація реалізована переважно на основі поведінкової активності користувачів, без використання повноцінних моделей машинного навчання. Закритість архітектури та обмежені можливості інтеграції сторонніх ML-рішень ускладнюють адаптацію платформи до специфічних потреб освітніх установ, особливо в контексті впровадження власних інтелектуальних модулів аналізу успішності. Крім того, система має недостатню гнучкість щодо налаштування під локальні стандарти освіти та навчальні програми.

Таким чином, Edmodo є прикладом платформи, орієнтованої насамперед на організацію освітньої взаємодії та комунікації, однак її функціональні можливості в частині глибокої аналітики та персоналізації залишаються обмеженими, що знижує її ефективність у задачах інтелектуального аналізу навчальних результатів.

Таблиця 1.3 – Характеристики платформи «Edmodo»

Назва	Edmodo
Виробник	Edmodo LLC
Архітектура	Web application
Мова реалізації	PHP, JavaScript
Функції	1) створення віртуальних класів; 2) розміщення навчальних матеріалів і завдань; 3) спілкування між викладачами та учнями; 4) відстеження успішності;

Кінець таблиці 1.3

	5) базові рекомендації контенту на основі активності користувачів;
Переваги	1) простота використання; 2) орієнтація на шкільний сегмент; 3) наявність інструментів для спілкування та обміну матеріалами;
Недоліки	1) відсутність розширеної персоналізації та глибокої аналітики; 2) закрита архітектура та обмежені можливості інтеграції ML-рішень; 3) недостатня адаптація під локальні стандарти освіти;
Джерело інформації	https://www.educationworld.com/a_curr/rubin/edmodo-platform-transforms-learning.shtml

Розглянуті рішення демонструють загальний поступ у напрямі персоналізованого навчання та використання навчальної аналітики, однак мають низку суттєвих спільних обмежень. Передусім це відсутність гнучких механізмів інтеграції власних моделей аналізу даних, що унеможливлює впровадження індивідуально налаштованих алгоритмів машинного навчання відповідно до конкретних потреб освітнього закладу. Більшість платформ використовує закриті або фіксовані рекомендаційні механізми, які не допускають модифікації чи розширення логіки обробки даних.

Другим важливим недоліком є обмеженість у глибокій аналітиці результатів навчання та прогнозуванні успішності. Наявні системи здебільшого зосереджені на відображенні статистичних показників (середній бал, кількість спроб, рівень активності), але не забезпечують комплексного аналізу динаміки

навчання, виявлення прихованих закономірностей чи побудови прогнозів щодо подальших освітніх результатів здобувачів освіти.

Крім того, спостерігається недостатня адаптація до локальних освітніх стандартів, мовного середовища та специфіки навчальних програм, що особливо актуально для вітчизняної системи освіти. Низький рівень локалізації, обмежена підтримка національних нормативів оцінювання та негнучкість у налаштуванні структури навчального контенту знижують ефективність використання таких платформ у реальному освітньому процесі.

Сукупність зазначених недоліків підтверджує актуальність розроблення спеціалізованого програмного рішення, яке поєднуватиме функції тестування з поглибленим аналізом навчальних даних, використанням сучасних алгоритмів машинного навчання та формуванням персоналізованих рекомендацій для кожного здобувача освіти. Такий підхід дозволяє перейти від формального оцінювання знань до інтелектуальної підтримки навчального процесу та побудови індивідуальних освітніх траєкторій.

1.4 Аналіз інформаційної системи, що розробляється

Проведений аналіз сучасних тенденцій розвитку освітніх інформаційних систем, їх архітектурних особливостей та наявних рішень показав, що більшість платформ обмежуються базовим тестуванням та спрощеними механізмами персоналізації, не забезпечуючи глибокого аналізу результатів навчання. З урахуванням виявлених переваг і недоліків існуючих аналогів було визначено потребу в удосконаленні вже наявної системи тестування, розробленої раніше в рамках бакалаврського проєкту.

На основі вже наявної системи тестування, створеної в рамках бакалаврського проєкту, розробляється вебзастосунок аналізу успішності здобувачів освіти та формування персоналізованих рекомендацій. Його нова

версія не лише зберігає базовий функціонал організації та проходження тестування, а й суттєво розширює його за рахунок інтеграції модуля машинного навчання, який забезпечує автоматизований аналіз результатів навчальної діяльності. Це дозволяє переходити від простого фіксування оцінок до комплексної інтерпретації даних, враховуючи індивідуальні показники успішності, динаміку результатів, частоту спроб, темп виконання завдань та інші навчальні характеристики.

У межах такого підходу система формує індивідуальні освітні траєкторії для кожного здобувача освіти, що передбачає надання рекомендацій щодо повторення тем, опрацювання додаткових матеріалів або корекції темпу навчання. Для викладача вебзастосунок виступає ефективним аналітичним інструментом, який забезпечує оперативну оцінку рівня успішності як окремого студента, так і навчальної групи загалом, дозволяє виявляти проблемні теми та приймати обґрунтовані рішення щодо корекції навчального процесу. Водночас здобувач освіти отримує персоналізований зворотний зв’язок, який сприяє усвідомленому плануванню власної навчальної діяльності та підвищенню мотивації до навчання.

Система орієнтована на використання в умовах змішаного та дистанційного навчання, де освітній процес відбувається у цифровому середовищі та супроводжується значними обсягами навчальних даних. У таких умовах викладачі часто стикаються з труднощами в оперативному відстеженні індивідуального прогресу кожного здобувача освіти, особливо при роботі з великими групами. Запропонований вебзастосунок дозволяє структурувати цей процес, автоматизуючи обробку результатів тестування та забезпечуючи систематичний моніторинг навчальної діяльності.

Поєднання інструментів тестування з елементами освітньої аналітики створює умови для переходу від фрагментарного контролю знань до цілісної

системи підтримки навчання, у якій результати оцінювання використовуються не лише для формальної перевірки, а й для аналізу тенденцій успішності. Це особливо важливо в умовах цифровізації освіти та розподіленого освітнього процесу, де ефективність навчання значною мірою залежить від здатності системи забезпечити узгодження між навчальним контентом, результатами оцінювання та поточним рівнем підготовки здобувача освіти.

Таблиця 1.4 – Аналіз системи, що розробляється

Основні задачі	1) Реєстрація користувачів; 2) авторизація існуючих користувачів; 3) додавання учнів до групи/класу викладача; 4) створення та редагування тестів; 5) призначення тестів для учнів або груп; 6) проходження тестів учнями; автоматична перевірка закритих типів питань; 8) перевірка та оцінювання відкритих запитань викладачем; 9) збереження та перегляд результатів тестування в особистих профілях; 10) сповіщення про необхідність проходження або перевірки тесту; 11) автоматизований аналіз успішності учнів на основі накопичених даних; 12) генерація персоналізованих рекомендацій для індивідуального навчального вектора; 13) візуалізація аналітики успішності для викладачів (зведені показники, проблемні теми).

Кінець таблиці 1.4

Користувачі системи	1) учень; 2) вчитель;
Сценарії роботи системи	1) викладач створює тест та призначає його певній групі або індивідуальним учням; 2) учень проходить тест у зручний час; після завершення система автоматично перевіряє закриті питання, а відкриті − відображаються викладачеві для ручного оцінювання;
Сценарії роботи системи	3) після перевірки всі результати зберігаються у профілі учня та стають доступними для подальшого аналізу; 4) модуль машинного навчання аналізує накопичені результати (бал, час виконання, повторні спроби тощо), формує індивідуальні рекомендації для учня (наприклад, теми для повторення, додаткові матеріали); 5) викладач отримує аналітичні звіти про успішність групи та може скоригувати навчальний процес на основі даних.
Засоби апаратної та програмної реалізації	1) back-end: C#, .NET, AspNetCore, EntityFrameworkCore. 2) front-end: JavaScript, React; 3) ML-модуль: Python (scikit-learn); 4) database: Microsoft SQL Server;

Описані характеристики, задачі та сценарії функціонування системи дозволяють сформувати цілісне уявлення про логіку її роботи, структурну організацію та основні напрями використання у навчальному процесі. Чітке визначення ролей користувачів, функціональних можливостей і послідовності виконання процесів свідчить про системний підхід до побудови вебзастосунку та його орієнтацію на практичні потреби освітнього середовища. Представлена

модель відображає узгодженість між етапами збору даних, їх обробкою, аналізом та подальшим використанням результатів для підтримки навчальної діяльності.

Запропонована структура вебзастосунку забезпечує комплексний підхід до аналізу навчальної успішності, поєднуючи інструменти тестування з аналітичними механізмами та засобами підтримки персоналізованого навчання. Це дозволяє не лише оцінювати рівень засвоєння матеріалу, але й формувати індивідуальні рекомендації, спрямовані на корекцію навчальної траєкторії здобувача освіти та підвищення ефективності освітнього процесу в цілому.

Такий підхід створює функціональне підґрунтя для подальшого проєктування архітектури системи та обґрунтування вибору технологічних рішень, що будуть розглянуті в наступних розділах роботи, а також визначає основу для реалізації алгоритмічного модуля аналізу й інтеграції сучасних цифрових інструментів у процес оцінювання навчальних досягнень.

Висновки до розділу 1

У першому розділі магістерської кваліфікаційної роботи здійснено комплексний аналіз предметної галузі та сучасних інформаційних систем, що застосовуються для підтримки навчального процесу й оцінювання навчальних досягнень здобувачів освіти. Розглянуто основні тенденції розвитку таких систем, серед яких провідне місце займають персоналізація навчання на основі аналізу даних, впровадження навчальної аналітики, адаптивне тестування, використання модульної архітектури та інтеграція із зовнішніми освітніми платформами через стандарти LTI та xAPI. Показано, що сучасні рішення еволюціонують у напрямі інтелектуальних систем підтримки навчання, однак у більшості випадків рівень реальної аналітичної глибини залишається обмеженим, а рекомендаційні механізми не враховують складну динаміку індивідуальних освітніх траєкторій.

У розділі проаналізовано особливості функціонування аналогічних інформаційних систем, їх типову архітектуру, склад функціональних модулів та рольову модель доступу. Визначено, що такі платформи характеризуються наявністю навчальної аналітики, накопиченням великого обсягу освітніх даних і використанням адаптивних елементів, проте здебільшого застосовують спрощені алгоритмічні підходи без можливості гнучкої інтеграції складних моделей аналізу даних. Проведено порівняльний огляд платформ Docebo, Fishtree та Edmodo, що дозволило виявити їх функціональні переваги, а також спільні обмеження, до яких належать закритість архітектури, низька адаптивність до локальних освітніх потреб, обмежені можливості кастомізації та недостатній рівень персоналізації навчального процесу.

На основі здійсненого аналізу обґрунтовано доцільність удосконалення раніше створеної системи тестування шляхом інтеграції інтелектуального модуля аналізу результатів навчання із застосуванням методів машинного навчання. Визначено ключові задачі розроблюваної системи, її функціональні ролі, користувачів та сценарії взаємодії, а також описано програмно-технічне середовище реалізації. Сформульовані вимоги та структурні характеристики створюють теоретичне й методологічне підґрунтя для подальшого проєктування архітектури вебзастосунку, спрямованого на автоматизований аналіз успішності та формування персоналізованих рекомендацій для підтримки індивідуальних освітніх траєкторій здобувачів освіти.

2 ПРОЄКТНІ РІШЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальна концепція проєктування системи

Система аналізу успішності та формування персоналізованих навчальних рекомендацій розробляється як розширення вже існуючого вебзастосунку для створення та проходження тестів. У попередній бакалаврській роботі було реалізовано платформу, що забезпечує реєстрацію користувачів, створення тестових завдань викладачами, проходження тестів здобувачами освіти та збереження результатів у їхніх профілях. Проте ця система не мала інструментів для глибокого аналізу навчальних досягнень та надання персоналізованих порад учням.

У межах магістерської роботи вебзастосунок розширюється модулем машинного навчання, що здатний на основі накопичених результатів тестування виконувати прогнозування успішності учнів та генерувати персональні рекомендації щодо подальшого навчання. Такий підхід дозволяє не лише автоматизувати процес оцінювання, а й перетворити систему на інтелектуального асистента викладача, який допомагає визначити проблемні теми та адаптувати навчальний процес до потреб кожного здобувача освіти.

Проєктування системи базується на двох ключових складових:

− моделюванні функцій та інформаційних потоків, що забезпечує структуроване представлення логіки роботи всієї системи;

− розробці математичного забезпечення модуля машинного навчання, яке дозволяє виконувати прогнозування та формувати рекомендації.

Поєднання цих складових дає змогу створити цілісну архітектуру системи, де традиційні вебзастосунки для тестування доповнюються інтелектуальним модулем аналізу даних. Такий підхід забезпечує узгодженість усіх компонентів,

підвищує ефективність процесу оцінювання та створює основу для персоналізації навчання.

2.2 Моделювання функцій та інформаційних потоків системи

Одним з ключових етапів проєктування програмного забезпечення є моделювання функцій та інформаційних потоків. Це дозволяє формалізувати логіку роботи системи, відобразити взаємодію між її складовими та визначити шляхи передачі даних. Для побудови функціональної моделі використано нотацію IDEF0, яка дозволяє відобразити функції системи у вигляді блоків із зазначенням вхідних та вихідних даних, механізмів та керуючих впливів. Для уточнення руху інформації застосовано діаграми потоків даних (DFD), що демонструють, як дані передаються між користувачами, базою даних, серверною логікою та модулем машинного навчання [6].

Для опису функціональної структури вебзастосунку та визначення основних потоків інформації використано нотацію IDEF0, яка дозволяє формалізувати процес аналізу успішності у вигляді ієрархії взаємопов’язаних функцій. Ця нотація забезпечує чіткий поділ елементів процесу на вхідні та вихідні дані, керуючі впливи та механізми реалізації, що сприяє кращому розумінню логіки роботи системи, ролі кожного її компонента та взаємозв’язків між ними. Використання IDEF0 є доцільним у контексті даної роботи, оскільки дозволяє структуровано представити складний процес обробки навчальних даних і формування персоналізованих рекомендацій.

На першому рівні абстракції побудовано контекстну діаграму (A-0), яка відображає систему як єдиний узагальнений процес аналізу навчальних досягнень та генерації рекомендацій для здобувачів освіти (рисунок 2.1). Вона демонструє основні інформаційні потоки між системою та зовнішнім

середовищем, окреслює роль користувачів і допоміжних компонентів, а також слугує базою для подальшої деталізації функцій на нижчих рівнях моделювання.

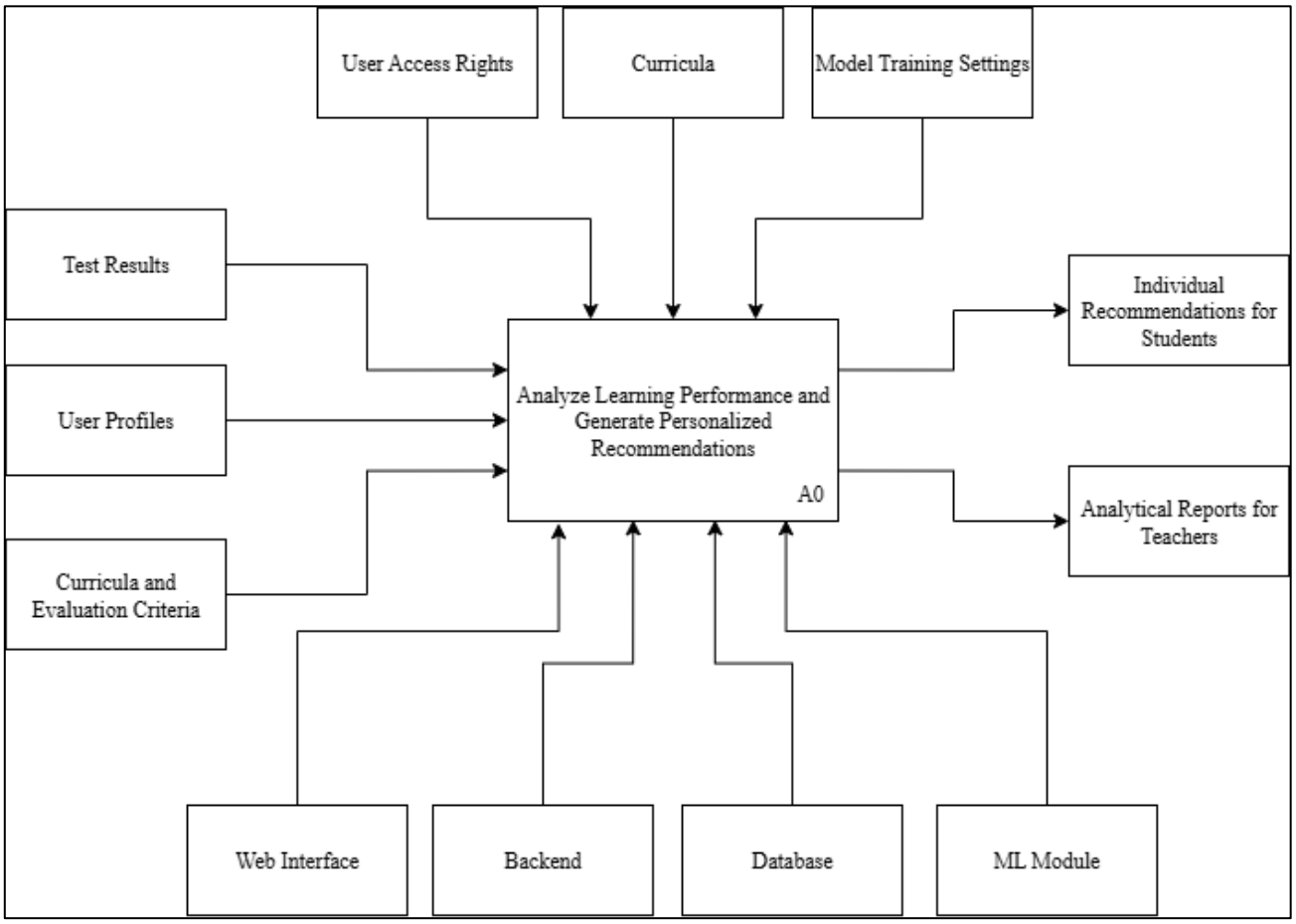


Рисунок 2.1 – Контекстна діаграма IDEF0 системи

Контекстна діаграма IDEF0 системи відображає узагальнену структуру процесу аналізу успішності здобувачів освіти та формування персоналізованих рекомендацій. Центральний процес «Analyze Learning Performance and Generate Personalized Recommendations» отримує на вході результати тестів, профілі користувачів та навчальні плани з критеріями оцінювання. Керуючими впливами виступають права доступу користувачів, навчальні плани та налаштування навчання моделей машинного навчання. Вихідними даними є індивідуальні рекомендації для учнів та аналітичні звіти для викладачів. Як механізми, що забезпечують функціонування даного процесу, використовуються вебінтерфейс,

серверна частина, база даних та модуль машинного навчання, які спільно реалізують повний цикл обробки освітніх даних − від їх збору до генерації аналітичних результатів.

Для деталізації роботи системи було виконано декомпозицію контекстної діаграми (A-0) на перший рівень. На цьому рівні функціональність системи поділено на чотири основні блоки, що описують ключові процеси: управління користувачами, управління тестами, проведення тестування та аналітику з формуванням персоналізованих рекомендацій. Діаграма A0 відображає внутрішню структуру системи та потоки даних між її складовими (рисунок 2.2).

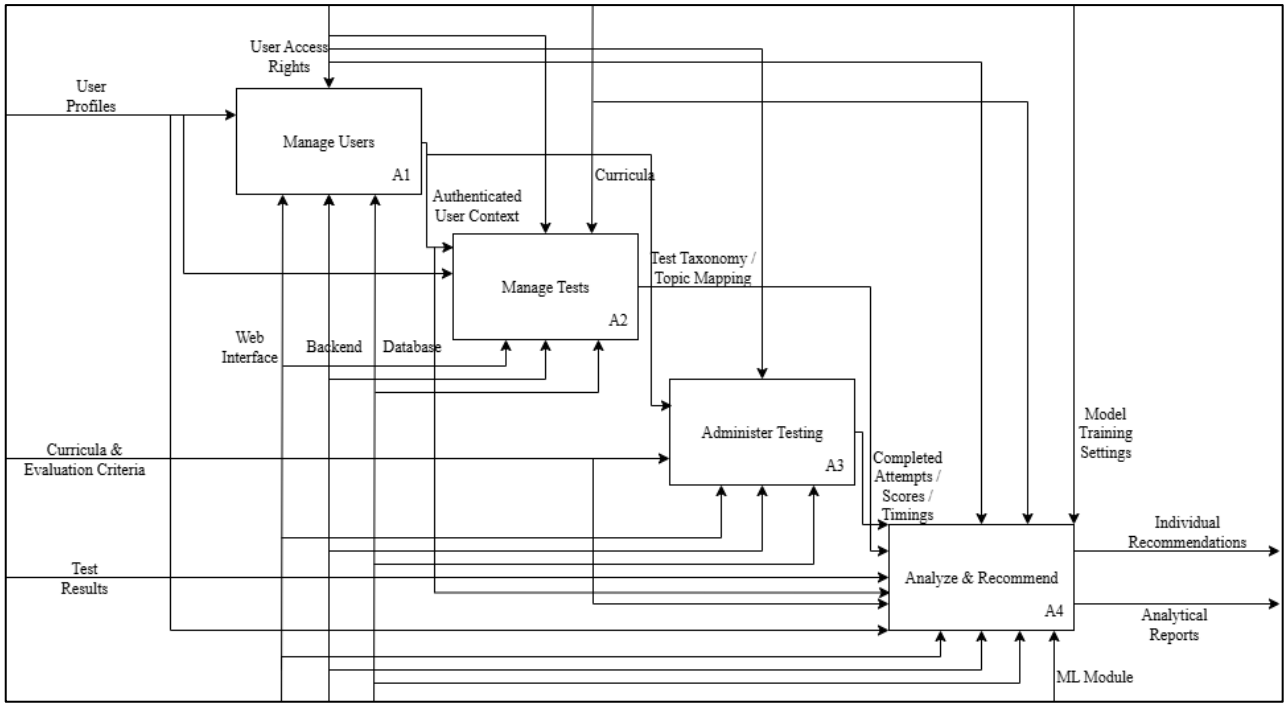


Рисунок 2.2 − Декомпозиція контекстної діаграми IDEF0 системи

Система аналізу навчальних досягнень і формування персоналізованих рекомендацій складається з чотирьох основних функціональних блоків, які забезпечують повний цикл роботи із навчальними даними. Перший блок Manage Users відповідає за реєстрацію, автентифікацію та авторизацію користувачів, визначає їхні ролі (учень, викладач, адміністратор) та формує контекст доступу,

який використовується іншими складовими. Другий блок Manage Tests надає викладачам інструменти для створення та редагування тестів, використовуючи навчальні плани та критерії оцінювання для визначення тем, складності завдань і зв’язку з навчальними цілями; він формує структуровану інформацію про тести, яка надалі застосовується при проведенні оцінювання та аналітиці. Третій блок Administer Testing забезпечує проведення тестувань: учні проходять призначені їм завдання, система автоматично перевіряє закриті питання, фіксує результати спроб, час виконання та відповіді на відкриті запитання, що потребують ручного оцінювання викладачем. Четвертий блок Analyze & Recommend виконує обробку зібраних результатів, аналізує дані про користувачів та структуру тестів, застосовує алгоритми машинного навчання для прогнозування успішності та формування персоналізованих рекомендацій для учнів, а також генерує аналітичні звіти для викладачів. Така структура дозволяє інтегрувати управління користувачами, створення тестів, проведення оцінювання та інтелектуальний аналіз результатів у єдину узгоджену систему.

Для більш детального відображення структури інформаційних потоків та взаємозв’язків між основними процесами вебзастосунку була побудована діаграма потоків даних (DFD) першого рівня. Вона дозволяє структурувати рух інформації між зовнішніми користувачами, функціональними підсистемами та сховищем даних, фокусуючись на передачі результатів тестування, управлінні користувачами та формуванні аналітичних даних.

Застосування цієї діаграми дає змогу графічно проілюструвати логіку циркуляції навчальної інформації в системі, окреслити ключові процеси обробки даних та показати їхню взаємодію з модулем аналітики успішності. Такий підхід забезпечує цілісне розуміння того, як формуються, трансформуються та використовуються дані на різних етапах функціонування вебзастосунку.

Діаграма потоків даних першого рівня відображає повний цикл роботи системи аналізу навчальних досягнень та формування персоналізованих рекомендацій, починаючи від взаємодії з користувачами й завершуючи збереженням результатів та рекомендацій (рисунок 2.3).

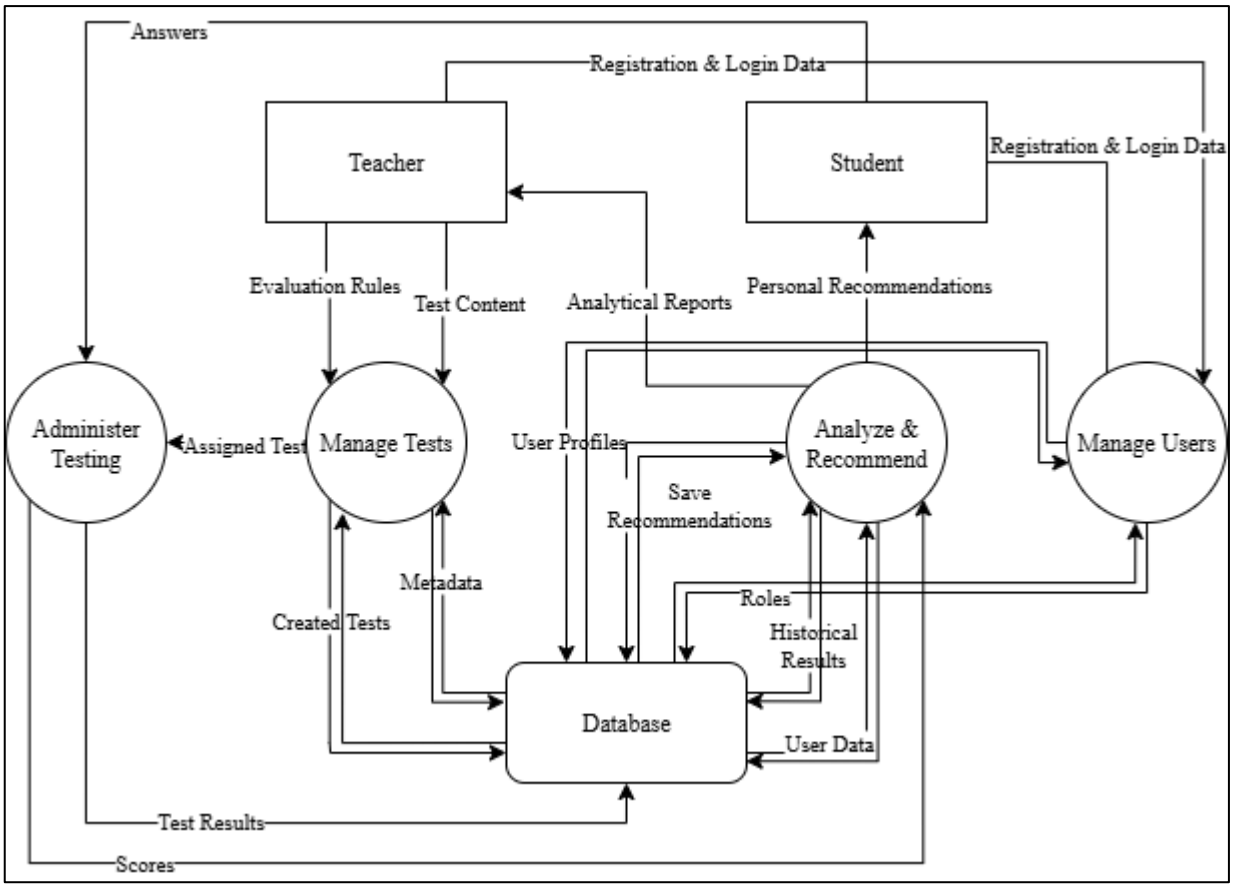


Рисунок 2.3 − Діаграма потоків даних першого рівня для системи

Викладачі та учні передають у процес Manage Users дані реєстрації та авторизації, на основі яких формується профіль користувача, визначаються його роль і права доступу, а відомості зберігаються у центральній базі даних. Процес Manage Tests дає викладачам змогу створювати та редагувати тести, налаштовувати правила оцінювання, передаючи створені завдання та метадані до бази даних і далі до процесу Administer Testing. Учні отримують призначені тести, виконують їх і надсилають відповіді, після чого результати проходжень −

оцінки, час виконання та спроби − зберігаються в базі даних та можуть бути передані викладачеві для перевірки відкритих питань. Процес Analyze & Recommend використовує дані про користувачів, результати тестів та історію навчання для аналітики й прогнозування успішності за допомогою алгоритмів машинного навчання; на основі аналізу формуються персоналізовані рекомендації для учнів і аналітичні звіти для викладачів. Сформовані рекомендації додатково зберігаються у базі даних, що забезпечує їх подальше використання та підтримує цілісність інформаційних потоків у системі.

Таким чином, побудовані функціональні та інформаційні моделі відображають основні процеси роботи системи та шляхи руху даних між її складовими. Щоб забезпечити коректність подальшої реалізації модуля аналізу навчальних досягнень та формування рекомендацій, окремі функції системи було формалізовано математично.

2.3 Математичне та алгоритмічне забезпечення

У межах поставленого дослідження було сформульовано задачу бінарної класифікації освітніх результатів здобувачів освіти на основі вхідного вектору навчальних ознак, що включають як кількісні (наприклад, кількість виконаних завдань, середній бал, частота взаємодії з навчальною платформою), так і якісні характеристики (тип дисципліни, форма навчання тощо). Такий тип задачі належить до категорії навчання з учителем (supervised learning), де модель будується на основі навчальної вибірки, що містить приклади з відомими мітками класів, і повинна узагальнювати виявлені закономірності для застосування до нових даних [7].

Серед відомих підходів до побудови моделей класифікації (логістична регресія, метод опорних векторів, наївний баєсівський класифікатор, нейронні мережі тощо) було обрано метод дерева рішень − як найбільш придатний для

інтерпретованого, структурованого й адаптивного аналізу в освітньому середовищі. Цей метод реалізує жадібний алгоритм побудови ієрархічної структури, де кожна внутрішня вершина (вузол) відповідає перевірці деякої умови на одній із ознак, а кожна гілка − результату цієї перевірки, що веде до подальшого розщеплення [8].

Першою перевагою є відсутність жорстких вимог до структури даних: дерево рішень здатне ефективно працювати з гетерогенними наборами ознак (числовими та категоріальними), не потребує попереднього масштабування чи нормалізації, а також демонструє стійкість до наявності пропущених значень. Це знижує загальну складність етапу попередньої обробки даних і робить метод зручним для прикладних досліджень.

По-друге, дерево рішень має високу інтерпретованість результатів, що особливо важливо для освітньої сфери, де пояснюваність моделі має пріоритет над її складністю. Побудоване дерево можна подати у вигляді наочної діаграми з послідовністю логічних умов, що дозволяє відстежити причини класифікації кожного об'єкта та сформулювати рекомендації на основі структури дерева.

По-третє, метод забезпечує оцінювання вагомості ознак (feature importance), що дає змогу ідентифікувати ключові фактори, які найбільше впливають на навчальні результати. Це відкриває простір для подальшого статистичного та педагогічного аналізу, спрямованого на виявлення глибинних причин успішності чи проблемних зон у навчанні.

Окрім того, алгоритм CART (Classification and Regression Trees) [9] включає механізми обрізання дерева (pruning) та налаштування максимальної глибини, що дає змогу зменшити ризик перенавчання (overfitting) і досягти балансу між узагальненням і точністю моделі. У навчанні модель оптимізує функціонал якості, заснований на зменшенні критерію Джині або ентропії, що формалізує вибір найінформативніших ознак на кожному етапі побудови дерева.

У процесі побудови дерева рішень кожен вузол містить умову, яка дозволяє поділити вибірку на підмножини з нижчою неоднорідністю. Рішення приймаються послідовно − від кореневого вузла до одного з листів дерева. Такий механізм забезпечує зрозумілу логіку класифікації об’єкта за набором ознак (рисунок 2.4).

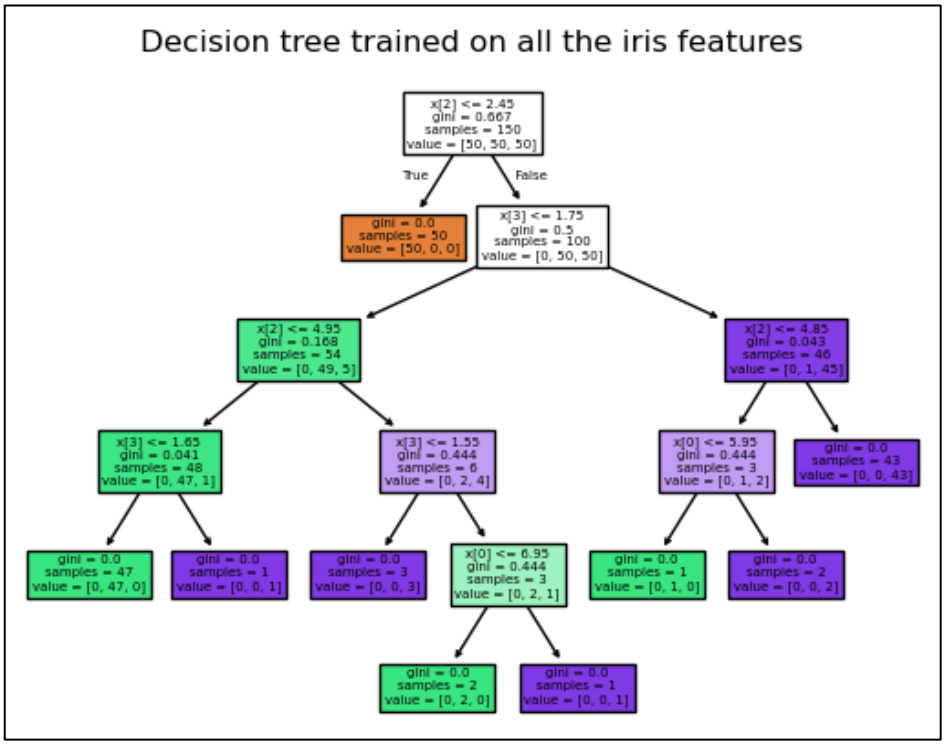


Рисунок 2.4 − Повна структура дерева рішень, навчена на даних Iris [10]

Таким чином, метод дерева рішень було обрано з огляду на його методологічну прозорість, гнучкість у роботі з реальними освітніми даними, простоту реалізації та відповідність принципам пояснюваної аналітики, що є особливо актуальним у контексті задач підтримки прийняття рішень у сфері освіти.

Враховуючи обґрунтованість вибору методу дерева рішень як основного інструменту для задачі класифікації, наступним кроком дослідження стала побудова моделі на основі реального освітнього датасету. Для цього необхідно

попередньо формалізувати вхідні дані − тобто перетворити їх у впорядковану структуру, яка б відповідала вимогам алгоритмів машинного навчання. Кожен навчальний приклад у такій структурі представлений як пара, що включає вектор ознак здобувача освіти та відповідне значення цільової змінної. Цей підхід відображається у вигляді математичної моделі:

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}, \quad (1)$$

де $x^{(i)} \in R^m$ − це числовий вектор, що описує навчальний приклад, тобто одного здобувача освіти, зокрема його середній бал, кількість пропущених занять, активність у системі дистанційного навчання, результати тестування тощо [11]. Відповідно, $y^{(i)} \in \{0,1\}$ − цільове значення, що вказує на рівень його успішності: умовно, клас 0 відповідає низькій успішності, а клас 1 − високій.

У даній кваліфікаційній роботі така формалізація використовується як вихідна основа для навчання моделі дерева рішень. Зокрема, під час навчання алгоритм буде аналізувати ці пари «ознаки − мітка класу», щоб побудувати узагальнену структуру правил, які дозволяють передбачити рівень успішності нових здобувачів освіти. Модель буде навчена на реальних або змодельованих освітніх даних, де кожен рядок відповідає одному студенту, а навчання полягає у пошуку ознак, що найсильніше впливають на кінцеве рішення.

Формула $D=\{(x^{(i)},y^{(i)})\}$ лежить в основі всіх подальших математичних процедур, пов’язаних із розрахунком критерію поділу, побудовою дерева та оцінюванням точності моделі. Саме вона задає структуру даних, яку використовує алгоритм машинного навчання під час етапу тренування.

Алгоритм побудови дерева полягає у рекурсивному розбитті простору ознак на підмножини. На кожному кроці обирається така ознака x_j і порогове значення t, які максимізують інформаційний приріст (Information Gain) або мінімізують неоднорідність (Impurity).

Побудова дерева рішень відбувається шляхом рекурсивного розбиття простору вхідних ознак на підмножини, у яких значення цільової змінної є більш однорідними. На кожному етапі алгоритм аналізує можливі варіанти розщеплення вибірки за ознаками і пороговими значеннями та обирає таке розбиття, яке найкраще зменшує «неоднорідність» у підмножинах, що утворилися. Кількісною мірою цієї неоднорідності виступає функція неоднорідності (impurity), яка оцінює, наскільки змішаними є класи у підмножині даних.

У цьому дослідженні як критерій неоднорідності використовується індекс Джині. Його значення для деякої множини DDD обчислюється за формулою:

$$G(D) = 1 - \sum_{k=1}^{K} p_k^2,\ p_k = \frac{n_k}{|D|}, \quad (2)$$

де p_k − це частка елементів класу k у підмножині D, а n_k − відповідна кількість об’єктів цього класу. Індекс Джині досягає нуля у випадку повної однорідності (усі приклади належать до одного класу), а його зростання вказує на змішування класів у вибірці.

Під час навчання дерева, для кожної ознаки та кожного можливого порогового значення алгоритм обчислює потенційне зниження неоднорідності після поділу. Для цього використовується формула критерію розщеплення:

$$\Delta G = G(D) - \left(\frac{|D_L|}{|D|} G(D_L) + \frac{|D_R|}{|D|} G(D_R)\right), \quad (3)$$

де D_L і D_R − відповідно ліва та права підмножини, що виникають після розщеплення, а $G(D_L)$ і $G(D_R)$ − їхні індекси Джині. Алгоритм обирає те розщеплення, яке забезпечує найбільше зниження ΔG, тобто найкраще підвищує однорідність кожної з отриманих частин.

У контексті даної роботи цей механізм дозволяє побудувати дерево, яке відображає логіку ухвалення рішень щодо класифікації здобувачів освіти на основі їхніх навчальних характеристик. Результатом є модель, здатна автоматично визначати рівень успішності нового здобувача, використовуючи ті самі ознаки,

що застосовувалися для побудови дерева. Таким чином, побудова дерева рішень є центральним етапом у створенні персоналізованої системи рекомендацій, яка враховує індивідуальні особливості освітнього профілю кожного користувача.

У процесі побудови дерева рішень ключовим кроком є вибір оптимального розщеплення простору ознак. Для цього алгоритм оцінює, наскільки кожен можливий поділ покращує структуру даних, тобто зменшує їх неоднорідність. Така оцінка здійснюється за допомогою функції зниження неоднорідності (impurity decrease), або ж інформаційного приросту, що обчислюється за формулою:

$$\Delta I = I(D) - \left(\frac{|D_{left}|}{|D|} I\left(D_{left}\right) + \frac{|D_{right}|}{|D|} I(D_{right})\right). \quad (4)$$

У цій формулі I(D) позначає значення обраної міри неоднорідності (наприклад, індексу Джині або ентропії) для початкової множини даних D, тоді як $I(D_{left})$ та $I(D_{right})$ – це значення неоднорідності у підмножинах, отриманих після розщеплення. Множини D_{left} та D_{right} утворюються шляхом поділу вихідної вибірки за певною ознакою та пороговим значенням.

Формула визначає, наскільки покращилась однорідність даних після розщеплення: чим більше значення ΔI, тим більший виграш від поділу, отже, тим ефективніше розщеплення. Саме це значення використовується як критерій для вибору оптимального вузла дерева. Алгоритм перебирає усі можливі ознаки та порогові значення, обчислює відповідні значення ΔI і обирає той варіант, для якого цей показник максимальний.

У контексті реалізації моделі аналізу успішності здобувачів освіти, дана формула дозволяє автоматично знаходити найбільш релевантні ознаки, що впливають на рівень успішності. Таким чином, зниження неоднорідності лежить в основі процесу навчання дерева, забезпечуючи ефективну побудову ієрархії рішень на основі освітніх даних.

Після завершення побудови дерева рішень кожен листовий вузол зберігає статистичну інформацію про ті приклади навчальної вибірки, що потрапили до нього. Зокрема, на основі цих прикладів визначається ймовірність належності об'єкта до кожного з можливих класів. Під час прогнозування для нового прикладу модель проходить шляхом гілок від кореня дерева до певного листа, в якому і виконується остаточне рішення − присвоєння класу.

Прогноз здійснюється за правилом більшості. Тобто модель обирає той клас, який має найбільшу емпіричну ймовірність серед об'єктів, що потрапили у відповідний лист під час навчання. Це формалізується наступним виразом:

$$\hat{y} = \arg\max_{k} p_k, \tag{5}$$

де p_k − це частка об’єктів класу k у даному листі. Функція arg max повертає значення індексу k, для якого ймовірність p_k є максимальною. Таким чином, новий об’єкт отримає мітку класу, який є найбільш типовим для навчальних прикладів у цьому вузлі.

У контексті поставленого дослідження це дозволяє автоматично віднести нового здобувача освіти до категорії «низька» або «висока» успішність на основі тієї ж структури дерева, яка була сформована під час навчання моделі. Вибір класу в листовому вузлі забезпечує завершення логіки класифікації і є фінальним етапом прийняття рішення у моделі дерева рішень.

Оскільки дерево рішень виконує поділ простору ознак на області, у межах яких приймається те саме рішення, такий поділ можна представити у вигляді поверхні класифікації, що демонструє залежність прогнозованого класу від значень ознак у різних комбінаціях (рисунок 2.5). У задачах регресії модель дерева рішень також формує розгалужену структуру рішень шляхом послідовного розщеплення вибірки за ознаками. Проте цільова змінна в цьому випадку є не дискретною, а неперервною числовою величиною.

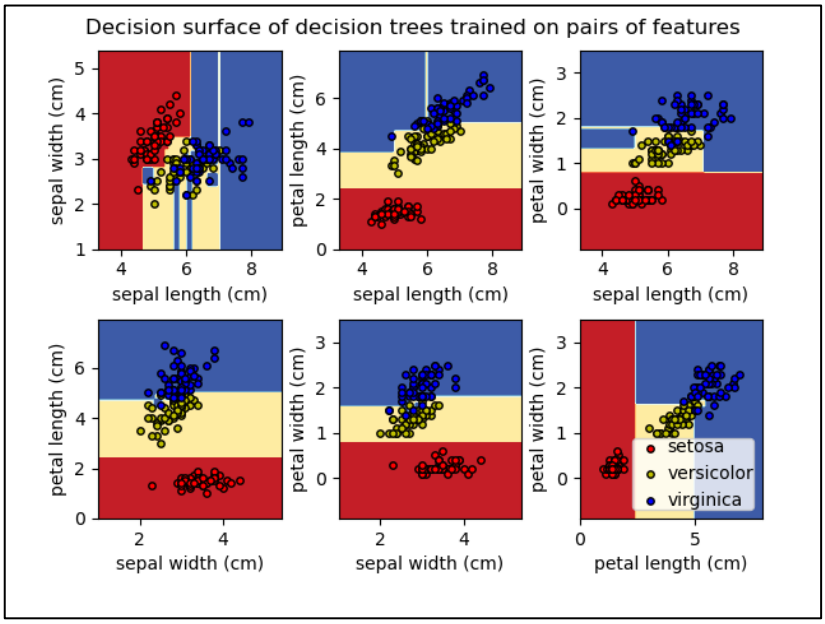


Рисунок 2.5 – Поверхні прийняття рішень дерева класифікації на парних комбінаціях ознак (набір Iris) [12]

Після завершення побудови дерева, кожен листовий вузол містить підмножину навчальних прикладів, що мають певні числові значення цільової змінної. Прогноз для нового прикладу визначається як середнє значення цільової змінної серед усіх об’єктів, які потрапили до відповідного листа під час навчання. Це дозволяє отримати наближене числове передбачення для нових даних.

Формально це записується як:

$$\hat{y} = \frac{1}{|D_{leaf}|} \sum_{(x^{(i)}, y^{(i)}) \in D_{leaf}} y^{(i)}, \quad (6)$$

де D_{leaf} – множина всіх навчальних прикладів, що потрапили до листового вузла, а $y^{(i)}$ – фактичне значення цільової змінної для кожного з них.

Таким чином, прогнозом моделі є середнє значення по всіх відомих прикладах, які, згідно з логікою дерева, мають схожий профіль ознак. У контексті освітньої аналітики така модель може бути застосована, наприклад, для передбачення майбутнього середнього балу здобувача освіти на основі наявних характеристик його успішності.

2.4 Специфікації вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням вебзастосунку є автоматизований аналіз успішності здобувачів освіти та формування персоналізованих рекомендацій для індивідуального навчального вектора.

Погодження, що ухвалені в програмній документації

Було погоджено використання ASP.NET Core, Entity Framework Core, React, Microsoft SQL та Python зі scikit-learn.

Межі проєкту ПЗ

Крайня дата завершення роботи над програмним забезпеченням – 31.11.2025 р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Програмне забезпечення використовується у закладах загальної середньої та вищої освіти для оцінювання навчальних досягнень, збору результатів тестування та надання персоналізованих рекомендацій студентам.

Характеристики користувачів

Основні характеристики користувачів: наявність ПК та підключення до мережі Інтернет.

Загальна структура і склад системи

Програмне забезпечення складається з клієнтського інтерфейсу на React, серверної частини на ASP.NET Core для обробки запитів і логіки, модуля машинного навчання на Python, MS SQL Server для зберігання даних через Entity Framework Core, REST API для зв’язку між клієнтською, серверною частинами та ML-модулем.

ФУНКЦІЇ СИСТЕМИ АНАЛІЗУ УСПІШНОСТІ ТА ФОРМУВАННЯ ПЕРСОНАЛІЗОВАНИХ НАВЧАЛЬНИХ РЕКОМЕНДАЦІЙ

Функція створення тестів

Опис функції

Функція дозволяє викладачам формувати індивідуальні тестові завдання для оцінювання навчальних досягнень учнів.

Вхідна і вихідна інформація

Вхідні дані: назва та опис тесту, перелік питань, типи відповідей, бали за питання, час виконання.

Вихідні дані: збережений структурований тест, готовий для призначення учням.

Функціональні вимоги

Доступ до бази даних зі створеними тестами, наявність інтерфейсу редагування та стабільне підключення до мережі Інтернет.

Функція проходження тестів

Опис функції

Функція дає змогу учням виконувати призначені тести онлайн.

Вхідна і вихідна інформація

Вхідні дані: відповіді учня на тестові завдання.

Вихідні дані: автоматично розраховані результати закритих питань, збережені відповіді на відкриті запитання, підсумкова оцінка після перевірки викладачем.

Функціональні вимоги

Доступ до бази даних із тестами та результатами, перевірка прав користувача (учень), робота через Інтернет.

Функція редагування тестів

Опис функції

Функція дозволяє викладачам оновлювати створені раніше тести, змінюючи назву, опис, питання, варіанти відповідей, часові обмеження та критерії оцінювання.

Вхідна і вихідна інформація

Вхідні дані: ідентифікатор тесту, нові або змінені параметри тесту (питання, відповіді, час).

Вихідні дані: оновлений тест із збереженими змінами.

Функціональні вимоги

Доступ до таблиць бази даних із тестами, можливість авторизованого редагування лише викладачем, підключення до Інтернету.

Функція оцінення відкритих питань

Опис функції

Функція надає викладачам можливість перевірки відповідей на відкриті питання, які неможливо перевірити автоматично.

Вхідна і вихідна інформація

Вхідні дані: відповіді учнів на відкриті питання.

Вихідні дані: оцінки за відкриті питання, внесені викладачем, оновлений підсумковий бал учня.

Функціональні вимоги

Доступ до бази даних з результатами тестування, перевірка прав доступу користувача (викладач), підключення до Інтернету.

Функція зберігання результатів тестування

Опис функції

Система зберігає всі результати виконаних тестів у профілях учнів для подальшого аналізу, перегляду та використання модулем машинного навчання.

Вхідна і вихідна інформація

Вхідні дані: результати тестів, відповіді, оцінки викладача за відкриті питання.

Вихідні дані: історія тестувань кожного учня з результатами та балами.

Функціональні вимоги

Доступ до бази даних із результатами тестування, захищене зберігання персональних даних, робота через мережу Інтернет.

Функція аналітики успішності та генерації рекомендацій (нова)

Опис функції

Модуль машинного навчання аналізує накопичені результати тестувань (бали, теми, час виконання, повторні спроби) та формує персоналізовані рекомендації для кожного учня. Викладач отримує аналітичні звіти про успішність групи та проблемні теми.

Вхідна і вихідна інформація

Вхідні дані: результати тестів (бал, тема, дата проходження, час виконання, кількість спроб).

Вихідні дані: для учнів − персональні рекомендації щодо тем для повторення або додаткових матеріалів; для викладачів − зведена аналітика успішності групи та окремих учнів.

Функціональні вимоги

Доступ до бази даних результатів, взаємодія із ML-модулем через REST API, асинхронне виконання аналізу, наявність підключення до Інтернету.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації

Джерелами інформації є викладачі, які створюють та редагують тести, перевіряють відкриті питання та переглядають аналітику успішності, а також учні, які проходять тести та отримують результати й персоналізовані

рекомендації. Система збирає та аналізує результати тестування, формує індивідуальні навчальні рекомендації та генерує аналітичні звіти.

Нормативно-довідкова інформація

До нормативно-довідкової інформації належать перелік навчальних дисциплін та тем, критерії оцінювання й шкали оцінок, а також навчальні плани та рекомендовані матеріали, що можуть використовуватися при формуванні рекомендацій.

Вимоги до способів організації, збереження та ведення інформації

Уся інформація зберігається у базі даних Microsoft SQL Server. Обмін даними між клієнтською частиною, сервером та модулем машинного навчання здійснюється через REST API. Доступ до даних розмежований відповідно до ролей користувачів, передбачений захист персональних даних та авторизований вхід.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Жорстких вимог до технічного забезпечення немає. Користувачеві потрібен персональний комп’ютер або ноутбук із сучасним веббраузером та підключенням до мережі Інтернет.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура системи складається з клієнтської частини на React, серверної частини на ASP.NET Core, бази даних Microsoft SQL Server та модуля машинного навчання, реалізованого на Python (scikit-learn), що взаємодіє із сервером через REST API.

Системне програмне забезпечення

Для роботи застосунку використовується Windows 10, а клієнтська частина підтримується сучасними веббраузерами (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).

Мережне програмне забезпечення

Передача даних між клієнтом, сервером та модулем машинного навчання здійснюється за протоколами HTTP та HTTPS поверх мережевої моделі TCP/IP.

Програмне забезпечення ведення інформаційної бази

Усі операції зі збереження, редагування та вибірки даних виконуються через ORM-фреймворк Entity Framework Core для бази даних Microsoft SQL Server.

Мова і технологія розробки ПЗ

Серверна частина реалізована мовою C# із використанням ASP.NET Core та Entity Framework Core, клієнтська частина − мовою JavaScript із застосуванням фреймворку React, модуль машинного навчання розроблено на Python.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс є веборієнтованим та адаптивним, забезпечує зручну навігацію, передбачає роздільний доступ для учнів, викладачів та адміністратора та візуалізацію аналітики й персональних рекомендацій.

Апаратний інтерфейс

Робота із системою передбачає використання персонального комп’ютера або ноутбука з доступом до Інтернету та екраном із роздільною здатністю не нижче 1366x768 пікселів.

Програмний інтерфейс

Для інтеграції клієнтської частини, серверної частини та модуля машинного навчання використовується REST API. Передбачено можливість розширення API для підключення зовнішніх освітніх платформ.

Комунікаційний протокол

Передача даних здійснюється за допомогою протоколів HTTP та HTTPS із використанням TLS для шифрування інформації.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Система працює у сучасних веббраузерах та не потребує встановлення додаткового клієнтського програмного забезпечення.

Супроводжуваність

Програмне забезпечення розроблене з можливістю оновлення моделей машинного навчання та розширення функціональності без зміни основної архітектури.

Переносимість

Застосунок може бути розгорнутий як на серверах з Windows, так і на Linux.

Продуктивність

Час виконання запитів не перевищує трьох секунд при середньому навантаженні. Аналітичні операції виконуються асинхронно та не блокують основний функціонал.

Надійність

Реалізовано контроль доступу за ролями, резервне копіювання даних та захист від несанкціонованого доступу.

Безпека

Система використовує авторизацію та автентифікацію користувачів із захищеними токенами, передає дані лише по протоколу HTTPS, а паролі зберігає у хешованому вигляді, забезпечуючи конфіденційність персональної інформації.

Висновки до розділу 2

У другому розділі магістерської кваліфікаційної роботи виконано комплексне проєктування системи аналізу навчальних досягнень та формування персоналізованих рекомендацій, що забезпечує цілісне бачення її функціональної організації та логіки роботи. Розроблено функціональну модель системи з

використанням нотації IDEF0, яка дозволила формалізувати основні процеси та їх взаємозв’язки. Побудовано контекстну діаграму та її декомпозицію на ключові підсистеми: управління користувачами, управління тестами, проведення тестування та аналітичну обробку результатів із формуванням рекомендацій. Такий підхід сприяв чіткому визначенню ролі кожного процесу в загальній структурі системи та забезпечив логічну послідовність обробки навчальних даних.

Для деталізації інформаційних потоків була розроблена діаграма потоків даних (DFD) першого рівня, яка відображає взаємодію між користувачами, серверною частиною, базою даних і модулем машинного навчання. Це дозволило наочно представити шляхи передачі даних, моменти їх обробки, збереження та використання в процесі прийняття рішень, а також забезпечити узгодженість функціональних компонентів системи.

У межах розділу також сформульовано задачу бінарної класифікації рівня успішності здобувачів освіти та обґрунтовано вибір методу дерева рішень як базового інструменту аналітичного модуля. Перевагою даного методу визначено його інтерпретованість, здатність працювати з різнотипними ознаками та відповідність вимогам пояснюваної аналітики в освітньому середовищі. Подано математичні основи побудови дерева рішень, описано критерії розщеплення даних, механізми оцінювання неоднорідності та принцип формування остаточного прогнозу. Це дозволило закріпити теоретичне підґрунтя для подальшої реалізації алгоритмічного модуля.

Окрему увагу приділено формуванню специфікації вимог до програмного забезпечення, що охоплює архітектуру системи, вимоги до інформаційного, технічного та програмного забезпечення, а також характеристики зовнішніх інтерфейсів.

3 АРХІТЕКТУРА ТА АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Загальна архітектура вебзастосунку

Вебзастосунок функціонує як багаторівнева система, у якій кожен компонент відповідає за окремий етап обробки інформації. Користувачі взаємодіють із системою через вебінтерфейс, який забезпечує доступ до тестів, результатів проходження, аналітичних звітів і сформованих рекомендацій. Серверна частина координує всі процеси, відповідає за обробку запитів, збереження результатів та ініціює аналітичну обробку після завершення тестування.

3.1.1 Діаграма компонентів вебзастосунку

Ключовою відмінністю досліджуваного вебзастосунку є наявність інтегрованого аналітичного модуля, який виконує функції обробки накопичених навчальних даних. Цей модуль працює як окремий логічний компонент, що взаємодіє із серверною частиною через структуровані запити. Після завершення тестування результати зберігаються у базі даних, звідки передаються до модуля аналізу для подальшої обробки. На основі вхідних параметрів − балів, кількості спроб, часу виконання, динаміки успішності − аналітичний модуль застосовує алгоритм дерева рішень для формування прогнозу рівня успішності та відповідних навчальних рекомендацій.

Інтеграція ML-модуля, відображена на діаграмі компонентів (рисунок 3.1), реалізує принцип відокремленості функцій, за якого сервер відповідає за управління процесами, а аналітичний компонент − за інтерпретацію даних та генерацію рекомендацій. Такий підхід забезпечує гнучкість архітектури, можливість оновлення або заміни алгоритмів аналізу без суттєвого впливу на інші складові системи, а також підвищує адаптивність програмного рішення до

змінних вимог освітнього середовища. Окрім цього, взаємодія між серверною частиною, базою даних і аналітичним модулем формує замкнений цикл обробки інформації: від фіксації результатів навчання до формування рекомендацій та їх подальшого використання здобувачем освіти у процесі корекції власної навчальної траєкторії.

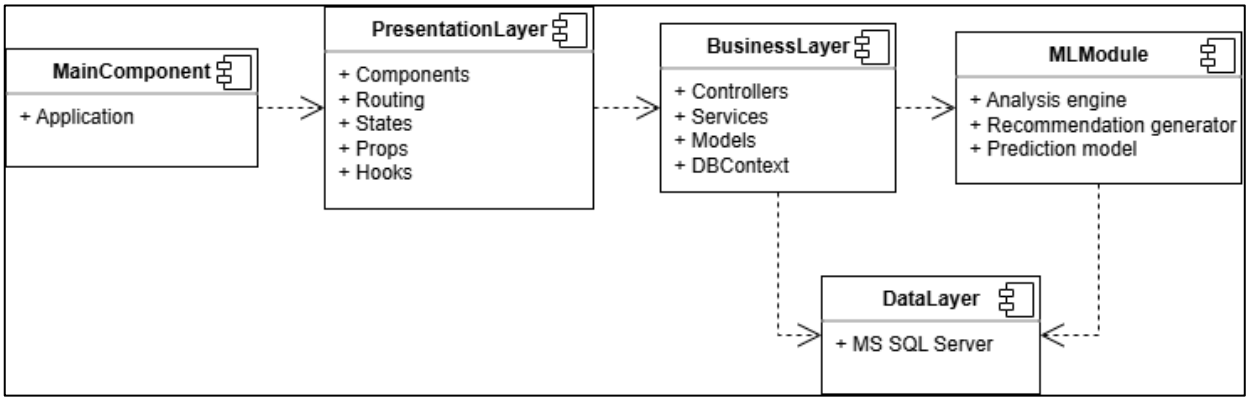


Рисунок 3.1 − Діаграмі компонентів системи

На діаграмі компонентів відображено логічну структуру вебзастосунку аналізу успішності здобувачів освіти та взаємодію між його основними підсистемами. Архітектура побудована за принципом багатошарової організації, де кожен компонент виконує чітко визначені функції та взаємодіє з іншими через регламентовані інтерфейси [13].

Компонент MainComponent представляє точку входу до застосунку та відповідає за ініціалізацію всієї системи. Від нього залежить запуск клієнтського інтерфейсу та маршрутизація до відповідних функціональних частин. Presentation Layer реалізує рівень представлення та забезпечує взаємодію користувачів із системою. До його складу входять візуальні компоненти, механізми маршрутизації, управління станами інтерфейсу, передавання параметрів та обробка користувацьких подій. Саме цей рівень забезпечує відображення тестів, результатів, аналітичних даних і рекомендацій. Business Layer відповідає за серверну логіку системи. У даному компоненті зосереджено контролери, сервіси

та моделі, які забезпечують обробку запитів, перевірку бізнес-правил, формування відповідей клієнтській частині та взаємодію з іншими підсистемами. Також через цей рівень здійснюється підготовка даних для аналітичного модуля та управління процесом генерації рекомендацій. Data Layer представлений базою даних MS SQL Server і забезпечує збереження всієї інформації, включаючи дані користувачів, результати тестування, історію спроб та сформовані рекомендації. Доступ до цього шару здійснюється через бізнес-рівень, що дозволяє зберігати контроль над цілісністю та безпекою даних. Окремим компонентом виділено ML Module, який реалізує інтелектуальну частину системи. До його складу входять механізми аналізу даних, модель прогнозування та генератор рекомендацій. Модуль отримує необхідні дані з бази даних через бізнес-рівень, виконує обчислювальну обробку та повертає результати у вигляді рекомендацій і прогнозів, які зберігаються у базі даних для подальшого відображення користувачам.

Таким чином, діаграма компонентів демонструє чітке розділення відповідальностей між рівнями системи та показує, що модуль машинного навчання інтегрований як автономна аналітична підсистема, яка розширює функціональність вебзастосунку, забезпечуючи інтелектуальний аналіз успішності та формування персоналізованих рекомендацій.

3.1.2 Діаграма варіантів використання системи

Окрім структурної побудови системи, важливим аспектом є відображення функціональної взаємодії користувачів із вебзастосунком. Саме через сценарії використання розкривається практична сутність системи, її орієнтація на потреби викладача та здобувача освіти, а також логіка реалізації процесів створення, проходження та аналізу тестування.

У межах даної роботи вебзастосунок розглядається не лише як технічна сукупність модулів, а як інструмент підтримки навчального процесу, в якому кожна роль має чітко визначені функції. Викладач взаємодіє із системою з метою організації тестування, контролю знань та аналізу академічної успішності, тоді як здобувач освіти використовує систему для виконання навчальних завдань, перегляду результатів та отримання персоналізованих рекомендацій. Відображення цих ролей і їхніх дій у вигляді формалізованої моделі дозволяє узагальнити функціональні можливості системи та визначити межі відповідальності кожного учасника процесу.

Саме тому доцільним є використання діаграми варіантів використання (Use Case) (рисунок 3.2), яка наочно демонструє основні сценарії взаємодії користувачів із вебзастосунком. Така діаграма дозволяє систематизувати функціонал системи, виявити ключові процеси та підкреслити роль модуля аналізу успішності у формуванні персоналізованих рекомендацій.

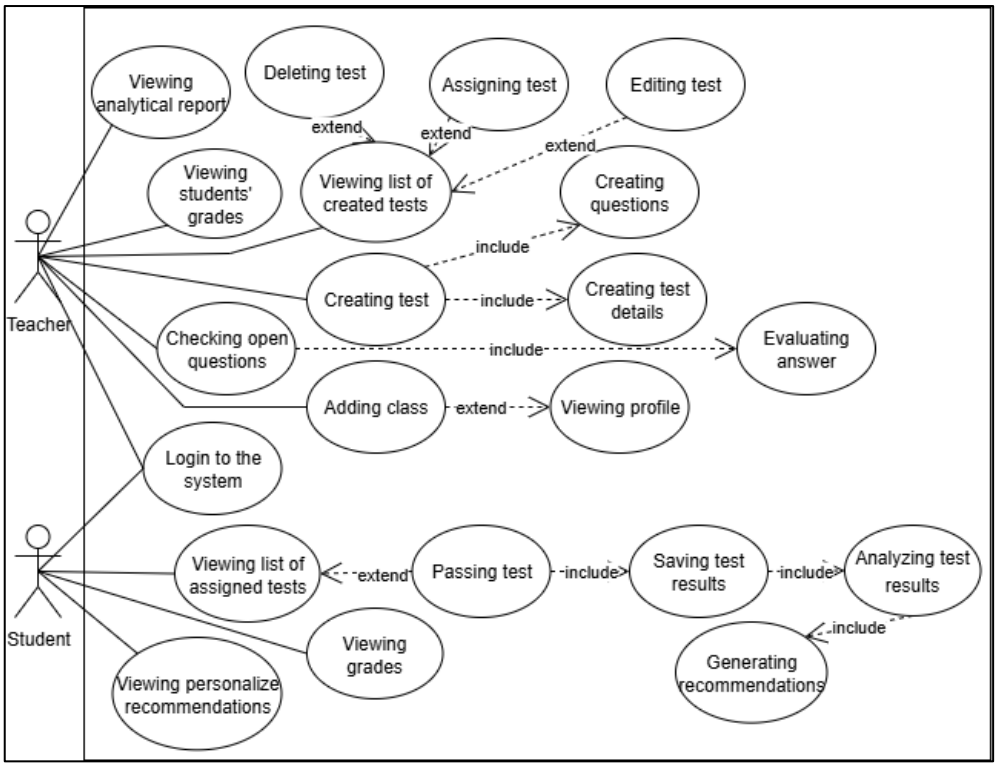


Рисунок 3.2 − Діаграма варіантів використання

Діаграма варіантів використання вебзастосунку аналізу успішності здобувачів освіти та формування персоналізованих рекомендацій відображає взаємодію користувачів із системою та окреслює основні функціональні можливості, що реалізуються в межах розробленого програмного продукту [14].

У системі виділено дві ключові ролі: викладач і здобувач освіти**.** Кожна з ролей має власний набір сценаріїв використання, які відповідають їх функціональним обов'язкам у навчальному процесі. Викладач здійснює управління навчальним контентом і контролює результати студентів. До його функцій належать створення тестів, редагування та видалення вже створених тестових матеріалів, призначення тестів окремим студентам або групам, перевірка відкритих відповідей, перегляд оцінок здобувачів освіти та отримання аналітичних звітів. Також передбачено можливість додавання класу та перегляду власного профілю. Всі ці дії формують сукупність управлінських функцій, що забезпечують організацію та контроль навчального процесу. Студент взаємодіє із системою з позиції користувача, який проходить тестування та отримує результати. Він переглядає список призначених тестів, обирає потрібний тест для проходження, виконує завдання та після завершення отримує оцінки й персоналізовані рекомендації. Окремо передбачена функція перегляду результатів навчання, що дає змогу аналізувати власну успішність і динаміку прогресу.

Важливою особливістю діаграми є відображення процесу аналізу результатів тестування та генерації рекомендацій, який інтегрований у загальний цикл взаємодії користувача із системою. Після збереження результатів тесту система ініціює процедуру аналізу, що завершується формуванням індивідуальних порад для студента. Таким чином, діаграма демонструє не лише стандартну логіку тестування, а й додатковий інтелектуальний рівень обробки даних. Зв'язки типу include та extend дозволяють деталізувати залежності між

окремими сценаріями, показуючи, які дії є обов’язковими складовими основного процесу, а які виконуються за потреби.

Отже, діаграма варіантів використання дає цілісне уявлення про функціональну структуру вебзастосунку, ролі користувачів та логіку їх взаємодії із системою, що створює підґрунтя для більш детального аналізу процесів обміну даними, відображених на діаграмі послідовності.

3.1.3 Діаграма послідовності взаємодії компонентів

Наведені раніше діаграми компонентів та варіантів використання дозволяють визначити структурну побудову вебзастосунку та функціональні можливості, доступні різним категоріям користувачів. Проте вони не відображають детально сам процес взаємодії між окремими елементами системи в динаміці, тобто не показують, у якому порядку та яким чином відбувається обмін повідомленнями між користувацьким інтерфейсом, серверною частиною, базою даних і модулем аналізу.

Саме для демонстрації логіки виконання конкретних сценаріїв у часовій послідовності використовується діаграма послідовності (рисунок 3.3). Вона дає змогу наочно простежити повний шлях обробки запиту − від ініціювання дії користувачем до отримання кінцевого результату, а також відобразити участь кожного компонента у цьому процесі. У контексті даної системи така діаграма є особливо важливою, оскільки дозволяє показати механізм інтеграції модуля аналізу успішності в загальну логіку роботи вебзастосунку: момент передачі результатів тестування на обробку, виконання алгоритмічного аналізу та повернення персоналізованих рекомендацій користувачеві.

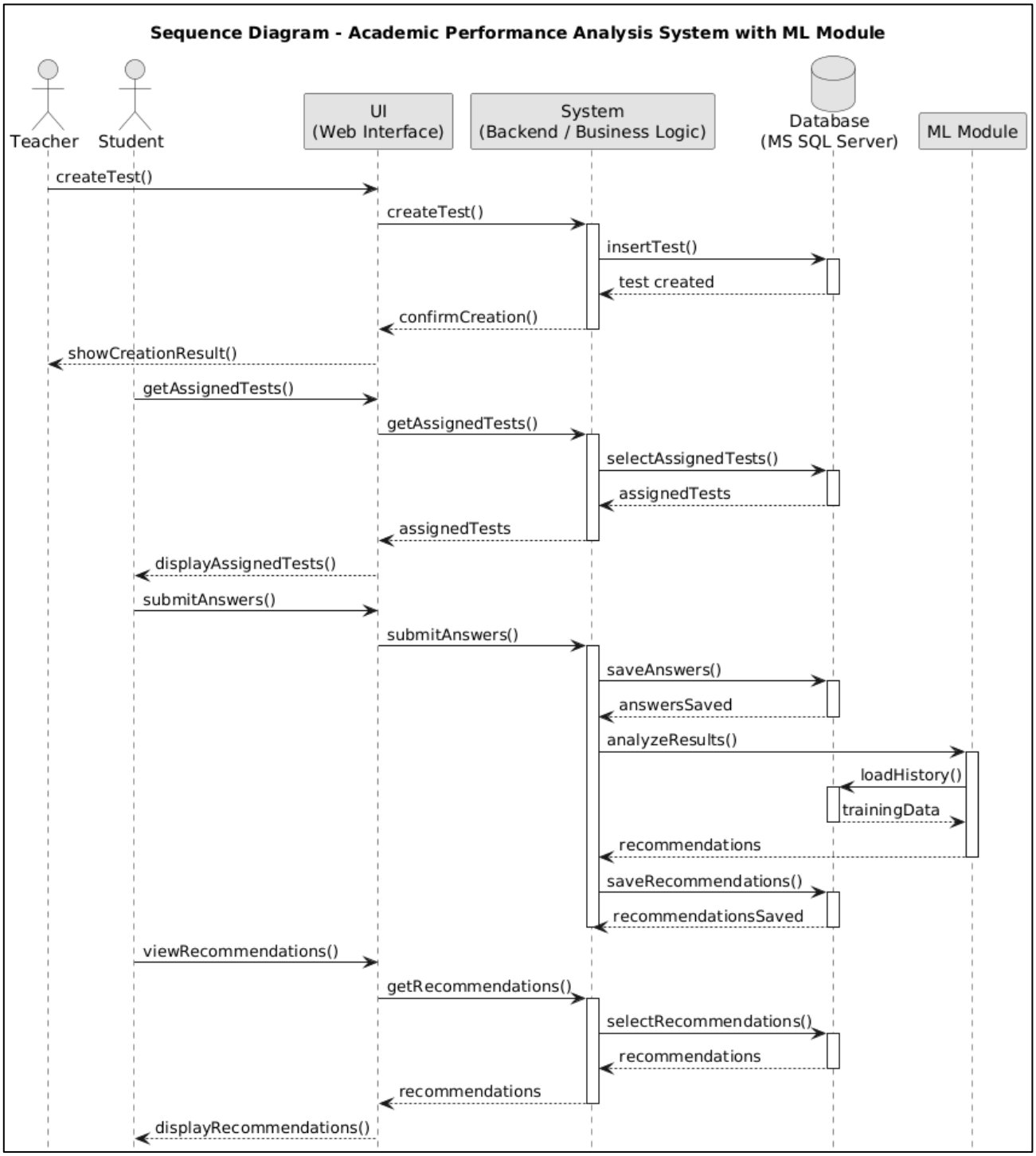


Рисунок 3.3 − Діаграма послідовності взаємодії компонентів

Діаграма послідовності відображає процес взаємодії основних учасників та компонентів вебзастосунку під час проходження тестування та формування персоналізованих рекомендацій. Діаграма демонструє часову послідовність

повідомлень між користувачами системи (викладачем і здобувачем освіти), користувацьким інтерфейсом, серверною частиною, базою даних та модулем машинного навчання [14].

Процес починається з дій викладача, який через вебінтерфейс ініціює створення тесту. Запит передається до серверної частини, де виконується обробка логіки створення та збереження тесту в базі даних. Після успішного виконання операції система повертає підтвердження користувачу про створення тесту. Далі здобувач освіти звертається до системи для отримання списку призначених йому тестів. Серверна частина виконує вибірку відповідних даних із бази даних та передає їх у вебінтерфейс для відображення користувачу. Після перегляду списку студент обирає тест і переходить до його проходження, вводячи відповіді на запитання. Введені відповіді передаються на сервер, де здійснюється їх збереження в базі даних. Після цього ініціюється процес аналізу результатів тестування, який виконується за допомогою модуля машинного навчання. Сервер передає до ML-модуля необхідні дані, включаючи історію результатів та показники успішності, на основі яких формується набір рекомендацій. Отримані рекомендації повертаються до серверної частини, зберігаються в базі даних і стають доступними для користувача. Здобувач освіти може переглянути персоналізовані поради через відповідну функцію вебінтерфейсу, після чого результати аналізу відображаються у зручному вигляді.

Таким чином, діаграма послідовності наочно ілюструє логіку обміну повідомленнями між компонентами системи, демонструючи інтеграцію модуля машинного навчання в основний робочий процес та підкреслюючи його роль у формуванні індивідуальних рекомендацій на основі результатів навчальної діяльності здобувача освіти.

Створені діаграми наочно подають структуру та логіку функціонування вебзастосунку, дозволяючи комплексно представити як його архітектурну

побудову, так і особливості взаємодії користувачів із системою. Діаграма компонентів відображає розподіл системи на основні функціональні модулі та зв’язки між ними, діаграма варіантів використання демонструє ключові сценарії роботи викладача і здобувача освіти, а діаграма послідовності деталізує процес обміну даними й часову логіку виконання операцій під час аналізу результатів тестування та формування персоналізованих рекомендацій. Такий підхід забезпечує цілісне уявлення про роботу вебзастосунку та підкреслює взаємозв’язок між його функціональними і аналітичними компонентами.

3.2 Огляд стеку технологій

Важливим етапом реалізації вебзастосунку аналізу успішності здобувачів освіти є вибір та обґрунтування технологічних засобів, які забезпечують стабільну роботу системи, її продуктивність і можливість подальшого розвитку. Використаний стек технологій охоплює інструменти для побудови клієнтської та серверної частини, організації зберігання даних, а також інтеграції модуля аналізу результатів навчання на основі алгоритмічних методів.

До основних технологій, що застосовуються у системі, належать:

- HTML, CSS, JavaScript, React − для реалізації клієнтської частини та формування інтерактивного вебінтерфейсу;
- C#, ASP.NET Core, Entity Framework Core − для побудови серверної логіки та обробки запитів;
- Microsoft SQL Server − для збереження, структуризації та обробки навчальних даних;
- Python та бібліотека scikit-learn − для реалізації модуля аналізу результатів навчання та формування персоналізованих рекомендацій.

Обрані технології відповідають вимогам до масштабованості, безпеки та ефективної обробки навчальних даних, що є необхідним для реалізації

функціоналу персоналізованих рекомендацій і забезпечення стабільної роботи вебзастосунку.

3.2.1 Технології реалізації вебзастосунку

Реалізація вебзастосунку аналізу успішності здобувачів освіти потребує використання сучасних технологічних засобів, здатних забезпечити стабільну роботу системи, швидку обробку запитів та надійну взаємодію між її основними компонентами. Вибір мов програмування здійснювався з урахуванням архітектури клієнт–серверної системи, необхідності ефективної обробки навчальних даних і можливості подальшого розширення функціоналу.

Для реалізації серверної частини вебзастосунку аналізу успішності здобувачів освіти обрано мову програмування **C#**, яка є доцільним рішенням для систем, що поєднують інтенсивну роботу з базою даних, складну бізнес-логіку та інтеграцію з аналітичними модулями. Особливістю даного застосунку є необхідність обробки великого обсягу навчальних даних, результатів тестування, формування аналітичних зрізів та підготовки структурованої інформації для подальшого аналізу, що потребує стабільного та надійного серверного середовища.

C# забезпечує високу продуктивність при обробці запитів та підтримує ефективну роботу з багатопоточністю й асинхронними операціями, що є важливим для системи з одночасним доступом великої кількості користувачів [15]. Це особливо актуально в умовах дистанційного навчання, коли сервер повинен обробляти паралельні звернення під час проходження тестів, збереження результатів та генерації рекомендацій. Строга типізація мови сприяє підвищенню надійності програмного коду, зменшує ймовірність логічних помилок та полегшує масштабування системи. Чітка структура об’єктно-орієнтованої моделі дозволяє логічно розділяти функціональні компоненти серверної частини, що є

важливим для подальшого розширення системи, зокрема вдосконалення алгоритмів аналізу успішності. Окрім цього, C# характеризується широкими можливостями інтеграції із зовнішніми сервісами та окремими модулями обробки даних, що дозволяє ефективно організувати взаємодію з ML-модулем, який здійснює аналітичну обробку результатів навчання. Завдяки цьому мова виступає не лише засобом реалізації серверної логіки, а й інструментом, що забезпечує узгоджену роботу всіх компонентів системи.

Для розроблення серверної частини вебзастосунку разом із мовою C# було обрано фреймворки ASP.NET Core та Entity Framework Core, які органічно доповнюють один одного і відповідають вимогам до продуктивності, масштабованості та надійної роботи з даними в освітній системі.

ASP.NET Core використовується як основа для побудови вебзастосунку та REST-орієнтованих сервісів [16]. Його перевагою є модульна архітектура, що дозволяє підключати лише необхідні компоненти та таким чином оптимізувати продуктивність. Для даної системи це особливо важливо, оскільки сервер повинен обробляти типові для навчального середовища сценарії: авторизацію користувачів, отримання списку тестів, збереження результатів, запити до аналітичного модуля тощо. ASP.NET Core надає зручні засоби маршрутизації, фільтрації запитів, обробки помилок та реалізації політик безпеки, що дає змогу чітко структурувати бізнес-логіку, відокремити її від інфраструктурних аспектів і забезпечити зрозумілий життєвий цикл обробки кожного запиту. Підтримка асинхронних операцій (async/await) дозволяє ефективно працювати з великою кількістю одночасних звернень, що є актуальним при масовому проходженні тестів або перегляді результатів у пікові періоди навчального процесу.

Entity Framework Core обрано як ORM-технологію для роботи з базою даних Microsoft SQL Server [17]. Для системи, яка оперує значними обсягами навчальних даних (результати тестів, історія спроб, профілі користувачів,

згенеровані рекомендації), важливо мати інструмент, що дозволяє поєднати зручність розробки з контролем над структурою даних. EF Core забезпечує роботу в парадигмі «об'єктно-реляційного відображення», де сутності доменної області описуються у вигляді C#-класів, а доступ до таблиць і зв'язків здійснюється через запити LINQ. Це зменшує кількість «ручного» SQL-коду, знижує ризик помилок при формуванні запитів і покращує читабельність коду, що важливо для підтримки та розвитку проєкту в довгостроковій перспективі. Механізм міграцій дозволяє керовано змінювати структуру бази даних у процесі еволюції системи, синхронізуючи модель даних із фізичною схемою БД без порушення цілісності інформації. У контексті даного вебзастосунку це дає змогу поступово розширювати модель (додавати нові показники успішності, типи рекомендацій, аналітичні сутності) без суттєвих змін низькорівневого коду і без ризику втрати даних.

У поєднанні ASP.NET Core та Entity Framework Core забезпечують цілісний технологічний стек для побудови серверної частини: перший відповідає за обробку HTTP-запитів, реалізацію бізнес-логіки та інтеграцію з клієнтським інтерфейсом і ML-модулем, другий − за надійне збереження, вибірку й оновлення навчальних даних. Це дозволяє зосередитися на реалізації прикладних задач − аналізі успішності та формуванні персоналізованих рекомендацій − без перевантаження проєкту надмірними інфраструктурними деталями.

Для зберігання та обробки даних у вебзастосунку аналізу успішності здобувачів освіти обрано Microsoft SQL Server, який забезпечує надійну роботу з великими обсягами структурованої інформації та відповідає вимогам до стабільності, продуктивності й аналітичної обробки, що є критично важливими саме для даного проєкту.

Специфіка системи передбачає накопичення різнорідних даних: результатів тестування, історії спроб, часу виконання завдань, рівнів успішності,

параметрів для аналізу та згенерованих рекомендацій. Microsoft SQL Server ефективно підтримує роботу з такими даними завдяки чіткій реляційній моделі, можливості створення складних зв’язків між таблицями та забезпеченню цілісності інформації [18]. Використання первинних і зовнішніх ключів, обмежень цілісності та тригерів дозволяє гарантувати коректність збережених результатів, що особливо важливо для системи, яка використовується в освітньому процесі і є джерелом об’єктивних показників успішності. Однією з ключових переваг Microsoft SQL Server для даного застосунку є висока продуктивність при обробці аналітичних запитів. Система підтримує оптимізацію виконання SQL-запитів, індексацію, кешування та планування запитів, що дозволяє швидко формувати зведені звіти, статистичні показники успішності та вибірки даних для подальшого аналізу в ML-модулі. Це особливо актуально у сценаріях, коли викладач одночасно переглядає аналітику для цілої групи, або коли система виконує масовий аналіз результатів після проходження тестів великою кількістю користувачів. Значущою перевагою SQL Server є підтримка механізмів безпеки, які дозволяють реалізувати розмежування доступу до даних, шифрування, аудит операцій та захист конфіденційної інформації користувачів. Оскільки система зберігає персональні дані здобувачів освіти та результати їх навчання, використання СУБД із розвиненими засобами захисту є важливою умовою відповідності етичним і нормативним вимогам до обробки освітньої інформації. Крім того, Microsoft SQL Server підтримує масштабування та стабільну роботу в умовах зростання обсягу даних. У процесі експлуатації вебзастосунку база даних поступово накопичуватиме історичні результати тестування, що у перспективі може використовуватися для побудови більш точних моделей аналізу успішності. Завдяки можливості оптимізації структури зберігання та використанню інструментів адміністрування SQL Server, система зберігає продуктивність навіть при значному збільшенні навантаження.

Microsoft SQL Server також добре підходить для інтеграції з серверною частиною, побудованою на ASP.NET Core та Entity Framework Core. Така зв’язка забезпечує узгодженість між логічною моделлю даних і фізичною структурою бази, спрощує доступ до інформації та зменшує кількість низькорівневого коду. Це важливо для поступового розширення функціоналу системи, наприклад, додавання нових показників успішності або типів рекомендацій без необхідності радикальної перебудови схеми даних.

Для реалізації клієнтської частини вебзастосунку використовується JavaScript як основна мова забезпечення динамічної та інтерактивної взаємодії користувача з інтерфейсом [19]. Її застосування є доцільним з огляду на необхідність оперативного відображення результатів тестування, аналітичних показників і персоналізованих рекомендацій без перезавантаження сторінки.

JavaScript забезпечує обробку подій, керування елементами інтерфейсу та асинхронну взаємодію із серверною частиною. У межах даного проєкту це дозволяє реалізувати плавну навігацію між тестами, миттєве оновлення результатів, відображення прогресу здобувача освіти та інтерактивну візуалізацію навчальних даних. Важливою перевагою JavaScript є його тісна інтеграція з сучасними бібліотеками, зокрема React, що дає змогу будувати компонентну структуру інтерфейсу, спрощувати підтримку клієнтської частини та забезпечувати її масштабованість при розширенні функціоналу системи.

Для формування структури сторінок вебзастосунку використовується HTML (HyperText Markup Language), який забезпечує логічну розмітку інтерфейсу, розміщення елементів керування, текстової інформації, форм тестування та блоків відображення результатів. HTML виступає основою побудови користувацького інтерфейсу та визначає ієрархію його компонентів [20].

Оформлення та візуальне подання інтерфейсу реалізується за допомогою CSS (Cascading Style Sheets), що дозволяє налаштовувати кольорову палітру, типографіку, розміри, відступи та адаптивну поведінку елементів під різні розміри екранів [20]. Для спрощення створення уніфікованого та адаптивного дизайну використовується фреймворк Bootstrap, який надає готову сіткову систему, стилізовані компоненти та інструменти для швидкої розробки зручного інтерфейсу без необхідності повністю проєктувати стилі з нуля [21].

Для реалізації клієнтської частини вебзастосунку обрано бібліотеку React, яка є одним із найпоширеніших інструментів для створення динамічних користувацьких інтерфейсів. Її застосування дозволяє забезпечити високу інтерактивність, гнучкість та масштабованість інтерфейсу системи аналізу успішності здобувачів освіти [22].

Ключовою особливістю React є побудова інтерфейсу на основі компонентного підходу, згідно з яким кожен елемент інтерфейсу (форма проходження тесту, таблиця результатів, панель рекомендацій, графіки успішності тощо) реалізується як окремий незалежний компонент. Такий підхід спрощує структурування коду, підвищує його повторне використання та полегшує підтримку й розширення функціоналу системи. Завдяки використанню віртуального DOM React забезпечує високу продуктивність при оновленні інтерфейсу [23]. Це є особливо важливим для даного вебзастосунку, оскільки система передбачає часті динамічні зміни даних: оновлення результатів тестування, виведення аналітичних показників, відображення рекомендацій, формування статистичних графіків тощо. Віртуальний DOM мінімізує кількість операцій перемальовування сторінки, що позитивно впливає на швидкодію. React також забезпечує ефективну реалізацію односторінкової архітектури (SPA), у межах якої зміна вмісту сторінки відбувається без повного перезавантаження браузера. Це значно покращує користувацький досвід: навігація між тестами,

профілем учня, аналітичними звітами та рекомендаціями є плавною і швидкою, що підвищує зручність взаємодії з системою. У контексті проєкту React дозволяє зручно працювати з асинхронними запитами до серверної частини через REST API. Це забезпечує оперативну передачу даних між клієнтською частиною та сервером, зокрема при отриманні списків тестів, завантаженні результатів, виведенні рекомендацій та обновленні статистичних показників.

Важливою перевагою React є також можливість інтеграції з бібліотеками візуалізації даних, що актуально для відображення результатів аналізу успішності у вигляді графіків, діаграм та аналітичних панелей. Це сприяє кращій інтерпретації результатів як для викладачів, так і для здобувачів освіти.

Для організації обміну даними між клієнтською частиною та сервером у проєкті використовується бібліотека Axios, яка є універсальним інструментом для виконання HTTP-запитів у JavaScript-середовищі [24]. Вона забезпечує зручну взаємодію з REST API, підтримує налаштування заголовків авторизації, обробку відповідей у форматі JSON та централізоване керування помилками, що є важливим для стабільної роботи вебзастосунку.

Оптимізація процесу отримання та кешування даних реалізується за допомогою бібліотеки React Query, яка дозволяє ефективно керувати асинхронними запитами, автоматично оновлювати застарілі дані та зменшувати кількість звернень до сервера [25]. Це позитивно впливає на швидкодію інтерфейсу та покращує користувацький досвід під час перегляду результатів тестування й аналітичних звітів.

Для обробки даних автентифікації застосовується бібліотека jwtDecode, яка використовується для розшифровування токенів доступу (JSON Web Token) на стороні клієнта [26]. Це дозволяє отримувати інформацію про користувача, його роль і права доступу без додаткових запитів до сервера, забезпечуючи коректне керування сесією та контроль доступу до функціональних можливостей системи.

3.2.2 Технології реалізації ML-модуля

Окремим функціональним компонентом вебзастосунку є модуль аналізу навчальних результатів, який забезпечує обробку накопичених даних, виявлення закономірностей успішності та формування персоналізованих рекомендацій для здобувачів освіти. Для реалізації такого модуля необхідні технології, що дозволяють здійснювати ефективну обробку даних, реалізовувати алгоритмічні моделі аналізу та забезпечувати їх інтеграцію з основною серверною частиною системи. Вибір інструментів для побудови ML-модуля орієнтований на гнучкість, точність обчислень і можливість подальшого розширення алгоритмічного функціоналу відповідно до потреб освітньої аналітики.

Для реалізації модуля аналізу навчальної успішності та формування персоналізованих рекомендацій у вебзастосунку було обрано мову програмування Python, яка є де-факто стандартом у сфері аналізу даних та машинного навчання [27]. Її використання зумовлене високою ефективністю при роботі з великими масивами структурованих даних, можливістю швидкої реалізації алгоритмів та широкою підтримкою інструментів для побудови моделей прогнозування.

Python відзначається простим, лаконічним і читабельним синтаксисом, що суттєво спрощує процес реалізації алгоритмічної логіки та експериментування з моделями. Це особливо важливо в межах даного проєкту, де необхідно не лише реалізувати модель дерева рішень, а й мати можливість коригувати її параметри, адаптувати до специфіки навчальних даних і аналізувати результати її роботи. Читабельність коду також сприяє підвищенню прозорості реалізованих рішень та полегшує їх подальшу підтримку. Важливою характеристикою Python є його орієнтованість на наукові та аналітичні задачі. Мова активно застосовується у сфері освітньої аналітики, прогнозування успішності, обробки статистичних даних та побудови інтелектуальних рекомендаційних систем. Саме тому її

використання є доцільним для задач інтерпретації навчальних результатів та моделювання індивідуальних навчальних траєкторій здобувачів освіти.

Окремою перевагою Python є здатність легко інтегруватися з іншими програмними платформами та мовами, зокрема з C# та ASP.NET Core, які використовуються для реалізації серверної частини вебзастосунку. Python-модуль може бути розгорнутий як окремий сервіс або фоновий аналітичний компонент, що взаємодіє з основною системою через HTTP-запити, API або доступ до спільної бази даних. Такий підхід забезпечує модульність архітектури, дозволяє ізолювати процеси навчання моделі від основної бізнес-логіки та мінімізує вплив аналітичних обчислень на продуктивність вебінтерфейсу. Крім того, Python забезпечує високу гнучкість у процесі розширення функціональності ML-модуля. За потреби можливе підключення додаткових алгоритмів, зміна структури навчального датасету або впровадження більш складних методів аналізу без кардинальної перебудови всієї системи. Це робить Python перспективним рішенням із точки зору масштабування та подальшого розвитку проєкту.

Для реалізації модуля машинного аналізу в даному проєкті як основну бібліотеку було обрано scikit-learn, яка є однією з найбільш усталених і зрілих бібліотек для класичного машинного навчання в екосистемі Python [28]. Вона надає широкий набір інструментів для розв’язання задач класифікації, регресії, кластеризації, відбору ознак та оцінювання моделей, що повністю відповідає потребам системи аналізу успішності здобувачів освіти. Вибір scikit-learn обумовлений насамперед її орієнтацією на прикладні задачі з табличними даними, де кожен об’єкт описується вектором ознак, а цільова змінна задає клас або числовий показник − саме такий формат мають навчальні дані в розроблюваній системі.

Однією з ключових причин використання scikit-learn є наявність у ній стабільної та добре реалізованої моделі дерев рішень, що виступає базовим алгоритмом у даній кваліфікаційній роботі. Бібліотека забезпечує готову реалізацію алгоритму CART із підтримкою налаштувань глибини дерева, мінімальної кількості об’єктів у листі, критеріїв розщеплення (індекс Джині, ентропія), механізмів боротьби з перенавчанням (обмеження складності дерева, обрізання) [29]. Це дозволяє сконцентруватися на коректному формуванні навчальної вибірки, підборі інформативних ознак та інтерпретації результатів, а не на ручній реалізації низькорівневих процедур побудови дерева. Крім того, бібліотека надає засоби оцінювання важливості ознак, що directly підтримує завдання виявлення факторів, які найбільше впливають на успішність здобувачів освіти. Важливою перевагою scikit-learn є уніфікований підхід до побудови й використання моделей: усі алгоритми мають єдиний інтерфейс fit/predict, підтримують однаковий формат вхідних даних і можуть комбінуватися в конвеєри (pipeline). Це спрощує експериментування з різними конфігураціями моделі (наприклад, зміна набору ознак, попередньої обробки, параметрів дерева) та забезпечує відтворюваність експериментів. У контексті проєкту це дозволяє на одному й тому самому наборі освітніх даних будувати кілька варіантів моделей, порівнювати їх якість за метриками точності, повноти, F1-міри тощо, і обґрунтовано обирати найкращу конфігурацію для використання у виробничому середовищі. З точки зору інтеграції з іншими компонентами системи scikit-learn є практичним рішенням, оскільки підтримує серіалізацію навчених моделей (збереження у файл і подальше завантаження для використання без повторного навчання). Це дає змогу розділити процеси навчання й експлуатації: модель може бути навчена офлайн на історичних даних, а потім використовуватися у вигляді окремого аналітичного модуля, який отримує з бази даних структуровані записи про здобувачів освіти, обчислює прогноз успішності та формує рекомендації.

Такий підхід органічно поєднується з архітектурою вебзастосунку, де ML-модуль функціонує як автономний компонент, але працює з тими самими даними, що й основна система. Додатковою перевагою scikit-learn є її зрілість, документація та стійкість до змін, що важливо для академічного й навчального проєкту. Бібліотека має детальну документацію, численні приклади використання, а її алгоритми широко описані в літературі та дослідницьких роботах. Це полегшує як обґрунтування обраних методів у тексті кваліфікаційної роботи, так і подальше розширення чи модифікацію моделі. Використання поширеного, добре задокументованого інструменту також підвищує прозорість та відтворюваність отриманих результатів, що є важливою вимогою до систем, які застосовуються в освітній сфері.

У підсумку вибір scikit-learn як основної бібліотеки для реалізації модуля машинного аналізу є обґрунтованим як з точки зору алгоритмічних можливостей (підтримка дерев рішень та засобів оцінювання якості моделей), так і з огляду на інтеграцію в загальну архітектуру системи, відтворюваність експериментів, зручність розробки та наявність широкої методичної бази для обґрунтування прийнятих рішень.

Висновки до розділу 3

У третьому розділі кваліфікаційної магістерської роботи розглянуто ключові аспекти проєктування та технічної реалізації вебзастосунку аналізу успішності здобувачів освіти з формуванням персоналізованих рекомендацій. Описано загальну архітектуру системи, побудовану на клієнт–серверному підході, яка забезпечує чіткий розподіл функцій між користувацьким інтерфейсом, серверною логікою, базою даних та модулем аналічної обробки результатів навчання. Особливу увагу приділено інтеграції модуля машинного

навчання в загальну структуру вебзастосунку та його ролі у процесі аналізу навчальних даних.

Виконано моделювання структури та функціонування системи засобами UML. Побудовано та проаналізовано діаграму компонентів, яка відображає взаємодію основних складових системи, діаграму варіантів використання, що формалізує сценарії взаємодії користувачів із вебзастосунком, а також діаграму послідовності, яка демонструє покрокову логіку обробки результатів тестування та передачі даних до модуля аналізу з подальшим отриманням рекомендацій. Сукупність цих діаграм дозволяє комплексно представити архітектуру, функціональні зв’язки та процеси системи.

Крім того, здійснено обґрунтований вибір стеку технологій для реалізації як вебзастосунку, так і ML-модуля. Розглянуто використання мови програмування C# для серверної частини, JavaScript для клієнтського інтерфейсу, фреймворків ASP.NET Core, Entity Framework Core та React, а також СУБД Microsoft SQL Server як надійної платформи для зберігання та обробки даних. Для реалізації алгоритмічного аналізу успішності обрано мову Python та бібліотеку scikit-learn, що забезпечують зручну інтеграцію з основною системою, стабільну роботу моделей та ефективне формування індивідуальних навчальних рекомендацій.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система аналізу успішності здобувачів освіти та формування персоналізованих навчальних рекомендацій реалізується як функціональне розширення вже існуючого вебзастосунку для створення та проходження тестів. Початкова версія платформи забезпечувала повний цикл тестування: реєстрацію користувачів, створення тестових завдань викладачами, проходження тестів здобувачами освіти та збереження результатів у персональних профілях. Однак отримані дані використовувалися переважно для фіксації досягнень і не передбачали глибокого аналітичного опрацювання.

Розширення функціональних можливостей системи орієнтоване на перетворення накопичених результатів тестування у джерело для аналітичних висновків і рекомендацій. Інтеграція модуля машинного навчання дозволяє здійснювати інтерпретацію навчальних даних, оцінювати динаміку успішності та формувати індивідуальні навчальні поради з урахуванням рівня підготовки здобувача освіти, його слабких і сильних сторін. Такий підхід змінює роль вебзастосунку з інструмента контролю знань на інтелектуальну систему підтримки навчального процесу, що сприяє підвищенню ефективності навчання та забезпечує більш обґрунтоване управління освітніми результатами.

4.1 Реалізація модуля машинного навчання

Під час проєктування системи було визначено, що модуль машинного навчання має функціонувати як окремий компонент, відповідальний за аналітичну обробку результатів тестування та формування персоналізованих рекомендацій. На концептуальному рівні (розглянутому в попередньому розділі) модуль виконує замкнений цикл обробки даних: отримує з бази даних вебзастосунку актуальні результати навчання, проводить їх алгоритмічний аналіз

і повертає у систему сформовані рекомендації для подальшого відображення користувачеві.

Така організація забезпечує чітке розмежування відповідальності між компонентами: вебзастосунок відповідає за збір та збереження первинних даних, тоді як ML-модуль здійснює інтелектуальну інтерпретацію цих даних, використовуючи методи машинного навчання. Завдяки цьому аналітичний блок може оновлюватися або масштабуватися незалежно від основної логіки системи, що відповідає принципам модульності, описаним у проєктній частині роботи.

Для інтеграції модуля машинного навчання існуючу базу даних вебзастосунку було доповнено низкою аналітичних таблиць (рисунок 4.1). Базові сутності (учні, вчителі, тести, запитання, результати проходжень) залишаються джерелом «сирих» даних, на основі яких формується навчальна вибірка. Нові таблиці зберігають уже агреговані результати роботи ML-модуля.

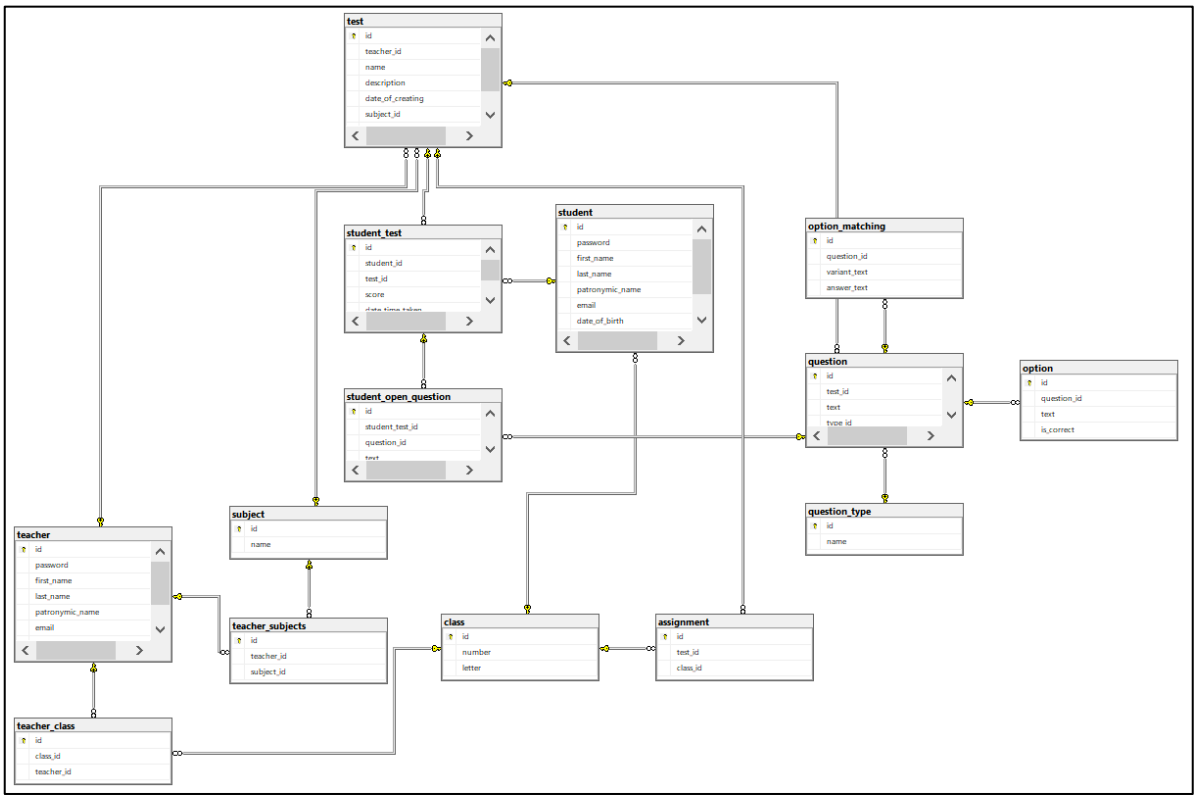


Рисунок 4.1 – Діаграма бази даних

Для коректного врахування навчального контексту та можливості аналізу успішності в динаміці було введено таблицю student_class_history, у якій зберігається історія навчання учня у різних класах. Кожний запис містить ідентифікатор учня, ідентифікатор класу, а також проміжок часу, протягом якого учень навчався у відповідному класі. Це дозволяє прив’язувати результати аналізу до конкретного етапу навчання, коректно формувати річні зрізи успішності та відстежувати прогрес при переході між класами.

Центральною аналітичною сутністю є таблиця student_analysis, у якій для кожного здобувача освіти фіксується підсумковий результат роботи модуля аналізу. Запис у цій таблиці містить ідентифікатор учня, параметр scope, що визначає, чи стосується аналіз усього періоду навчання, чи окремого навчального року або класу, ідентифікатор класу на момент аналізу, дату та час його формування, а також кілька текстових полів. Поле main_profile_text містить опис освітнього профілю учня, career_text – сформовані кар’єрні рекомендації, weak_directions_text – узагальнений опис найбільш відстаючих напрямів, а worsening_subjects_text – інформацію про предмети, за якими спостерігається погіршення результатів. Таким чином, таблиця student_analysis поєднує структуровані ідентифікатори з текстовими інтерпретаціями, придатними для безпосереднього відображення у вебінтерфейсі.

Деталізація аналізу за освітніми напрямами зберігається у таблиці student_analysis_direction, записи якої пов’язані з конкретним аналізом через поле analysis_id. Для кожного освітнього напряму фіксується середній бал, історичний рівень успішності, а також прогнозований бал та відповідний прогнозований рівень. Додатково зберігається кількість врахованих тестів, що дозволяє оцінити надійність оцінки й прогнозу. Ця таблиця забезпечує можливість відстежувати динаміку успішності не лише на рівні окремих предметів, а й у розрізі ширших

освітніх напрямів (наприклад, природничий, гуманітарний тощо), що є важливим для формування цілісного навчального профілю.

Інформація про конкретні теми, за якими учень демонструє найнижчі результати, зберігається в таблиці student_analysis_weak_topics. Кожен запис містить посилання на відповідний аналіз, назву напряму, предмета та теми, а також числову оцінку за цю тему. Така структура дозволяє ML-модулю не лише виділити загальні «слабкі місця», а й надати точкові рекомендації на рівні окремих тем, що безпосередньо використовується при формуванні тексту для блоку «найбільш відстаючі теми».

У сукупності нові таблиці утворюють окремий аналітичний контур у структурі бази даних: student_analysis зберігає зведені результати для конкретного учня й періоду навчання, student_analysis_direction деталізує їх у розрізі освітніх напрямів, student_analysis_weak_topics фіксує проблемні теми, а student_class_history забезпечує часову й організаційну прив’язку аналізу до конкретних класів. Такий підхід дозволяє ML-модулю працювати з історичними даними, зберігати результати кожного запуску та забезпечувати швидкий доступ до сформованих рекомендацій без необхідності повторного повного перерахунку при кожному запиті з боку вебзастосунку. Нові аналітичні таблиці інтегруються в уже побудовану схему БД через систему зовнішніх ключів, що забезпечує узгодженість даних і логічну неперервність усіх інформаційних потоків у системі. Таблиця student_analysis пов’язана з базовою таблицею student за полем student_id, що дозволяє зберігати результати аналізу для кожного конкретного учня, та з таблицею class через поле class_id, що дає можливість формувати рекомендації з урахуванням того, у якому класі перебував учень на момент аналізу. Таблиця student_analysis_direction наслідує свій зв’язок від student_analysis через зовнішній ключ analysis_id, створюючи ієрархію показників успішності в розрізі освітніх напрямів. Аналогічно таблиця

student_analysis_weak_topics також пов’язана з student_analysis, що дозволяє деталізувати результати аналізу до рівня окремих предметів та тем. Таблиця student_class_history логічно інтегрується з таблицями student та class, фіксуючи всі зміни академічних груп і забезпечуючи коректність ретроспективного аналізу успішності. Така структура дозволяє ML-модулю напряму працювати з даними, узгодженими з системною моделлю, і забезпечує повну простежуваність між результатами тестування, навчальною історією та аналітичними висновками. Структура аналітичних таблиць побудована з урахуванням принципів нормалізації: дані про учнів, класи, предмети та тести не дублюються, а пов’язуються через зовнішні ключі з уже наявними довідковими таблицями. Це зменшує надлишковість, спрощує оновлення інформації, запобігає виникненню аномалій вставки й видалення та забезпечує цілісність даних при масштабуванні функціональності модуля машинного навчання.

Сформована структура бази даних разом із встановленими зв’язками між новими та наявними сутностями забезпечує повноцінний цикл зберігання аналітичної інформації та створює основу для подальшої роботи модуля машинного навчання. Коли дані набувають впорядкованого вигляду й стають доступними для обробки, постає завдання реалізувати програмні механізми, здатні отримувати ці дані, виконувати алгоритмічний аналіз і перетворювати результати в структуровані рекомендації. На цьому етапі акцент зміщується з проєктування структури даних на безпосередню імплементацію аналітичного функціоналу та його інтеграцію з вебзастосунком.

Модуль машинного навчання розроблено як окрему програмну компоненту на мові Python, що обумовлено її зручністю для роботи з даними та наявністю потужних інструментів для побудови моделей. Для реалізації алгоритмічної частини використано бібліотеку scikit-learn, у межах якої застосовано метод

дерева рішень як основний підхід до класифікації рівня успішності здобувачів освіти.

Модуль машинного навчання працює безпосередньо з тією ж базою даних, що й вебзастосунок, підключаючись до MS SQL Server через Python-бібліотеки pyodbc та SQLAlchemy. На рівні коду формується рядок підключення з параметрами сервера, бази даних та автентифікації, після чого створюється об’єкт «двигуна» (engine), який використовується для виконання SQL-запитів і завантаження даних у вигляді таблиць.

Для навчання та застосування моделі модуль використовує дані з кількох логічних груп таблиць. Основним джерелом є таблиця з результатами проходження тестів, у якій зберігаються відомості про кожну спробу: ідентифікатор учня, тесту, питання чи теми, набраний бал, дата й час виконання. Додатково залучаються довідникові таблиці користувачів (учні), предметів і тем, що дозволяє однозначно пов’язати числові ідентифікатори з конкретними навчальними дисциплінами та змістовими модулями. Окрема таблиця історії навчання student_class_history використовується для фіксації того, у якому класі й у який період навчався здобувач освіти, що важливо для коректної інтерпретації результатів у динаміці. Під час формування вибірки для аналізу з бази даних вибираються лише ті поля, які є суттєвими для побудови моделі: student_id (ідентифікатор учня), subject_id (ідентифікатор предмета), topic_id (ідентифікатор теми або розділу), score (отриманий бал), дата й час проходження тесту, а також, за потреби, інформація про клас або навчальний рік. На основі цих атрибутів згодом формується як цільова змінна (рівень успішності), так і ознаки, що подаються на вхід моделі. Таким чином, модуль працює не з усім масивом даних вебзастосунку, а з попередньо відібраною та структурованою підмножиною, яка безпосередньо стосується аналізу навчальних досягнень.

Для керування обсягом даних, які потрапляють до моделі машинного навчання, використовується параметр ANALYSIS_SCOPE, що визначає режим аналізу. Його обробка реалізована безпосередньо у функції filter_student_scope, яка отримує таблицю з усіма результатами тестування конкретного учня та повертає вибірку, що відповідає обраному сценарію (рисунок 4.2).

```python
def filter_student_scope(df_student: pd.DataFrame) -> pd.DataFrame:
    df = df_student.copy()

    if ANALYSIS_SCOPE == "all":
        return df

    if ANALYSIS_SCOPE == "current_class":
        class_id, date_from, date_to = get_current_class_period(TARGET_STUDENT_ID)

        if class_id is None or date_from is None:
            return df

        mask = (
            df["date_time_taken"] >= pd.to_datetime(date_from)
        ) & (
            df["date_time_taken"] <= pd.to_datetime(date_to)
        )

        df_filtered = df[mask]
        return df_filtered

    return df
```

Рисунок 4.2 − Фрагмент коду ML модуля фільтрування даних

У режимі «all» фільтрація не застосовується: функція просто повертає копію вихідного DataFrame, що дозволяє модулю працювати на повній історії навчальних даних здобувача освіти за весь час його навчання. У режимі «current_class» виконується додаткова логіка. Спочатку викликається функція get_current_class_period, яка визначає, в якому класі учень навчається зараз, та повертає межі відповідного періоду (date_from, date_to). Якщо такі дані знайдені, формується булевий масив mask, що відбирає лише ті записи, де дата проходження тесту (date_time_taken) належить визначеному часовому інтервалу. Таким чином, модуль аналізує лише актуальні результати − ті, що стосуються поточного навчального року або класу. Якщо ж історія класів не містить

достатньої інформації, функція повертає повну вибірку без обмежень. Завдяки такій структурі filter_student_scope забезпечує універсальність модуля: залежно від обраного сценарію, він може працювати або з повною історією успішності, або лише з поточного року навчання.

Після завантаження та попередньої фільтрації даних модуль машинного навчання переходить до етапу формування ознак (features) і побудови цільової змінної (target), яка використовується під час навчання моделі дерева рішень. Оскільки вихідні дані містять лише «сирі» оцінки за виконані учнем завдання, необхідно створити узагальнену шкалу рівнів навчальних досягнень, яка дозволяє моделі інтерпретувати оцінки в більш структурованому вигляді. Для перетворення числової оцінки (1-12) у відповідний рівень (0-3) у модулі використовується функція score_to_level(). Вона реалізує шкалу, визначену у другому розділі під час опису математичної моделі дерева рішень, де кожен рівень відповідає певній групі оцінок: початковий, середній, достатній та високий (рисунок 4.3).

```
def score_to_level(score: float) -> int:
    if score <= 3:
        return 0
    elif score <= 6:
        return 1
    elif score <= 9:
        return 2
    else:
        return 3
```

Рисунок 4.3 − Фрагмент коду ML модуля формування цільової змінної

Ця функція викликається під час формування набору даних: для кожного рядка вибірки обчислюється нове поле level, яке надалі слугує цільовою змінною для алгоритму класифікації. Таким чином, модель навчається не на окремих оцінках, а на узагальнених рівнях, що підвищує стійкість до коливань у межах одного діапазону оцінювання.

Дерево рішень, яке застосовується в ML-модулі, працює з трьома ключовими ознаками:

1) score – фактична оцінка учня за завдання чи тему (поточний рівень сформованості знань);

2) subject_id – предмет, до якого належить тема завдання (дає змогу виявляти міжпредметні закономірності);

3) topic_id – конкретна тема в межах предмета (дозволяє моделі робити точніший аналіз слабких місць).

Таке поєднання ознак є оптимальним для задачі прогнозування рівня успішності, оскільки охоплює як числовий показник, так і контекст предметної структури. Таким чином, кожен рядок вибірки представляє собою трійку параметрів, на основі яких модель будує дерево рішень і визначає гілки поділу (splits). Перед передачею матриці ознак у модель здійснюється її нормалізація за допомогою засобу MinMaxScaler із бібліотеки scikit-learn. Масштабування приводить кожну ознаку до інтервалу [0, 1] шляхом лінійного перетворення, при якому мінімальне значення ознаки стає 0, максимальне − 1, а всі проміжні значення пропорційно розподіляються між ними. Це забезпечує узгодженість масштабів ознак (score, subject_id, topic_id), оскільки вони мають різні діапазони та семантику. Незважаючи на те, що дерево рішень не є чутливим до абсолютного масштабу значень, нормалізація стабілізує процес навчання, спрощує візуальний аналіз даних та створює універсальний підхід до підготовки вибірки, який може бути повторно використаний у разі зміни архітектури моделі або додавання нових алгоритмів. А метод fit_transform() спочатку обчислює мінімальні та максимальні значення кожної ознаки, а потім застосовує нормалізуюче перетворення до всіх рядків матриці.

У модулі машинного навчання застосовується підхід побудови єдиної глобальної моделі, яка навчається на даних усіх учнів, а не окремої моделі для

кожного здобувача. Така стратегія дає змогу узагальнювати закономірності успішності на рівні всієї вибірки та формувати стабільніші правила класифікації, що враховують як предметну структуру, так і типові патерни помилок і сильних сторін учнів. Для класифікації рівнів навчальних досягнень використано алгоритм DecisionTreeClassifier з бібліотеки scikit-learn. Модель навчається прогнозувати значення цільової змінної level, яке відповідає одному з чотирьох рівнів успішності, описаних у попередньому розділі. Таке кодування відповідає педагогічній шкалі оцінювання і дозволяє алгоритмічно інтерпретувати успішність учнів у цифровому вигляді.

При створенні моделі явно задано такі параметри:

1) criterion="gini" − використовується критерій Джині, який вимірює «нечистоту» вузлів. На кожному кроці дерево вибирає ту ознаку і той поріг, які найсильніше зменшують нечистоту, тобто найбільш чітко розділяють вибірку за рівнями успішності.

2) max_depth=6 − глибина дерева обмежена шістьма рівнями, що дозволяє уникнути перенавчання. Без цього обмеження модель могла б надто точно підлаштуватися під індивідуальні випадкові коливання в успішності окремих учнів, погіршивши узагальнюючу здатність.

3) random_state=42 − фіксація початкового стану генератора випадкових чисел забезпечує відтворюваність структури дерева між запусками.

На кожному етапі побудови дерева алгоритм обирає таку ознаку та такий поріг розділення, які найефективніше відокремлюють приклади з різними рівнями успішності level. Це досягається шляхом обчислення зниження критерію Джині, який показує, наскільки «чистими» стають підвибірки після розщеплення. У ході цього процесу дерево поступово формує ієрархію вкладених правил, кожне з яких описує певний характерний сценарій навчальної успішності. Наприклад, спостереження з низькими оцінками у межах конкретної теми часто

опиняються у гілках, що ведуть до класів 0 або 1, що відповідає низькому та середньому рівням сформованості знань. Водночас стабільно високі оцінки з кількох предметів приводять модель до вибору класів 2 або 3, які відповідають достатньому й високому рівням.

Завдяки включенню таких ознак, як subject_id та topic_id, модель здатна враховувати неоднорідність навчальних дисциплін і складність окремих тем: різні предмети мають різний розподіл оцінок, а окремі теми можуть бути типовими джерелами труднощів для більшості учнів. Ці відмінності відображаються у структурі дерева, оскільки на різних вузлах можуть з’являтися розщеплення саме за предметною або тематичною ознакою − там, де це найбільше покращує розділення даних.

У результаті побудови створюється повноцінна структура дерева рішень, де листові вузли містять остаточний прогнозований рівень успішності. Саме ці значення використовуються як кінцеве навчальне рішення моделі. Усі описані елементи − вибір ознак, створення правил розщеплення, формування листових вузлів та присвоєння їм відповідних рівнів − реалізовані у межах єдиної функції навчання моделі, що забезпечує цілісність, послідовність і відтворюваність усього процесу (рисунок 4.4).

```python
def train_global_model(df_all: pd.DataFrame):
    df = df_all.copy()
    df["level"] = df["score"].apply(score_to_level)

    X = df[["score", "subject_id", "topic_id"]]
    y = df["level"]

    scaler = MinMaxScaler()
    X_scaled = scaler.fit_transform(X)

    model = DecisionTreeClassifier(
        criterion="gini",
        max_depth=6,
        random_state=42
    )
    model.fit(X_scaled, y)

    return model, scaler
```

Рисунок 4.4 – Фрагмент коду ML модуля навчання моделі

Після навчання глобальної моделі дерево рішень використовується для прогнозування рівня успішності конкретного здобувача освіти. На цьому етапі модуль переходить від загальної статистичної моделі до індивідуального аналізу, застосовуючи побудовані правила класифікації до персональної історії результатів учня. Вся логіка реалізована у функції apply_model_to_student, яка приймає дані одного учня, виконує їх попередню обробку та формує прогнозовані рівні засвоєння (рисунок 4.5).

```
def apply_model_to_student(df_student: pd.DataFrame, model, scaler) -> pd.DataFrame:
    df = df_student.copy()
    X = df[["score", "subject_id", "topic_id"]]
    X_scaled = scaler.transform(X)
    df["predicted_level"] = model.predict(X_scaled)
    return df
```

Рисунок 4.5 – Фрагмент коду ML модуля застосування моделі до даних окремого учня

На вхід функція отримує підмножину даних, що містить результати тестування обраного учня. Вона повторно масштабує ознаки score, subject_id і topic_id за допомогою вже навченого нормалізатора MinMaxScaler, який був підготовлений на глобальній вибірці. Це є критично важливим, оскільки масштабування має залишатися однаковим як під час тренування, так і під час застосування моделі; у противному разі дерево рішень отримало б несумісні значення ознак та некоректно класифікувало б приклади. Після масштабування функція викликає метод model.predict(), який повертає для кожного запису значення predicted_level − прогнозований рівень навчальних досягнень. Цей показник відображає, до якої категорії успішності (низький, середній, достатній чи високий рівень) модель відносить конкретну відповідь учня на основі структури дерева рішень, побудованої на глобальних даних.

Застосування моделі до одного учня є частиною загальної послідовності роботи ML-модуля. Спочатку формується вибірка всіх учнів, виконується

фільтрація за вибраним режимом аналізу та проводиться навчання моделі. Після цього обирається конкретний учень і, залежно від встановленого режиму, виконується фільтрація його результатів за періодом навчання у певному класі. Лише після цього модель застосовується. У результаті формується таблиця, що містить як фактичні оцінки учня, так і прогнозовані модельні рівні, які надалі використовуються для визначення тенденцій успішності, підсилення чи погіршення результатів та формування персоналізованих рекомендацій.

Після отримання прогнозів рівнів успішності для кожного окремого завдання модель машинного навчання передає дані в аналітичний блок, де вони узагальнюються та перетворюються на педагогічно осмислену інформацію. Логіка цього етапу реалізована у функції generate_direction_and_topic_recommendations, яка виконує повний цикл перетворень: від неструктурованих записів до комплексного освітнього профілю учня. Спершу результати групуються за навчальними напрямками, що визначаються на основі предметної належності. Для цього використовується функція detect_direction(subject_id), яка за допомогою мапи SUBJECT_DIRECTION_MAP повертає відповідний напрям, після чого формується зведена таблиця forecast_df. У цій структурі обчислюються середні фактичні бали, кількість виконаних тестів, історичні рівні успішності та прогнозовані рівні, отримані після застосування моделі. Саме ці агреговані показники дозволяють побачити загальну картину розвитку учня в кожному напрямку й визначити, який із них є домінуючим.

Побудована таблиця слугує основою для формування освітнього профілю. Напрямок, у якому учень демонструє найвищий прогнозований рівень та найбільш стабільні середні бали, вважається провідним. Такий підхід дозволяє уникнути хибних висновків у ситуаціях, коли прогноз високий, але фактичні оцінки нестабільні. Відповідно, освітній профіль учня є поєднанням об’єктивної

успішності та машинного прогнозування, а його текстовий опис надалі зберігається у базі даних і відображається в інтерфейсі.

Особливу увагу у функції приділено пошуку слабких тем − саме тих, які потребують повторного опрацювання. На відміну від первинної версії модуля, де визначалися лише найбільш проблемні теми в одному обраному напрямку, оновлена логіка охоплює всі предмети, що дозволяє сформувати більш точні рекомендації. Алгоритм послідовно аналізує кожний предмет, обчислюючи середні значення оцінок для кожної теми, після чого визначає мінімальний середній бал (рисунок 4.6). Якщо найменше значення належить високому рівню (відповідає рівню 3 за шкалою моделі), предмет повністю виключається з переліку проблемних, оскільки навіть найслабша тема у цьому випадку демонструє високий результат. У протилежному випадку модуль відбирає всі теми, оцінки яких дорівнюють мінімальному середньому значенню, і включає їх до переліку слабких тем для подальшого відображення в рекомендаціях.

```
weak_topics_all_subjects = []

for subject, part in df.groupby("subject_name"):
    topic_means = (
        part.groupby("topic")["score"]
            .mean()
            .reset_index()
    )
    topic_means["score"] = topic_means["score"].round(2)

    min_score = topic_means["score"].min()
    worst_topics = topic_means[topic_means["score"] == min_score]

    if score_to_level(min_score) == 3:
        continue

    for _, trow in worst_topics.iterrows():
        weak_topics_all_subjects.append({
            "direction": detect_direction(
                df[df["topic"] == trow["topic"]]["subject_id"].iloc[0]
            ),
            "subject": subject,
            "topic": trow["topic"],
            "score": float(trow["score"]),
        })
```

Рисунок 4.6 – Фрагмент коду ML модуля визначення слабких тем

Таке рішення дає змогу побудувати реалістичну картину індивідуальних проблемних зон учня, не обмежуючись лише одним напрямком чи предметом. Замість поверхневого визначення «найгіршої теми» модуль формує комплекс слабких елементів підготовки, які справді потребують додаткового опрацювання. Проте оцінювання поточного стану знань є лише частиною аналітики: не менш важливо визначити, чи не спостерігається у здобувача освіти негативна динаміка у вивченні окремих предметів. Саме для цього використовується механізм аналізу погіршення результатів, реалізований у функції find_worsening_subjects (рисунок 4.7).

```python
def find_worsening_subjects(df: pd.DataFrame) -> list[str]:
    worsening: list[str] = []

    for subject, part in df.groupby("subject_name"):
        part = part.sort_values("date_time_taken")
        scores = part["score"].to_numpy(dtype=float)
        n = len(scores)

        if n < 5:
            continue

        tail = min(4, n // 2)
        prev_scores = scores[:-tail]
        last_scores = scores[-tail:]

        if len(prev_scores) < 3:
            continue

        prev_mean = float(np.mean(prev_scores))
        last_mean = float(np.mean(last_scores))

        if last_mean <= prev_mean - 0.5:
            worsening.append(subject)

    return sorted(set(worsening))
```

Рисунок 4.7 – Фрагмент коду ML модуля визначення тем з погіршенням оцінок

На відміну від вибірки слабких тем, яка фіксує статичний зріз успішності, функція виявляє саме тенденцію зниження балів у часовій перспективі. Для кожного предмета послідовно аналізується хронологія проходжень тестів, обчислюється середній бал за попередній період і порівнюється з результатами

останніх спроб. Погіршення фіксується лише у випадку, коли зниження є статистично значущим, а не випадковим коливанням. Такий підхід дозволяє точно виявити ті навчальні дисципліни, де учень демонструє втрату стабільності або знань, що може бути критичним сигналом для корекції подальшої навчальної траєкторії.

У результаті комплексна логіка визначення слабких тем та предметів із погіршенням успішності створює багатовимірний аналітичний портрет здобувача освіти. Вона поєднує статичний та динамічний аналіз, що забезпечує високу точність рекомендацій і робить модуль машинного навчання по-справжньому корисним інструментом освітньої аналітики.

Після завершення алгоритмічної обробки та формування прогнозів модуль машинного навчання має повернути результати у вебзастосунок, щоб вони могли бути використані як у профілі здобувача освіти, так і в інтерфейсі викладача. Саме на цьому етапі відбувається передача сформованих даних у структури бази даних, створені спеціально для зберігання результатів аналітики. Уся взаємодія з СУБД Microsoft SQL Server здійснюється через бібліотеку pyodbc, яка забезпечує роботу з параметризованими SQL-запитами та явне підтвердження транзакцій (commit), що гарантує цілісність операцій.

Ключову роль відіграє функція upsert_student_analysis, яка відповідає за створення або оновлення запису в таблиці student_analysis − основній таблиці, що містить результати аналітичного профілю учня (рисунок 4.8). Саме тут зберігаються текстові інтерпретації моделі (освітній профіль, кар’єрні рекомендації, слабкі напрямки, теми зі зниженням успішності), а також службові параметри: student_id, analysis_scope та class_id. Для цього використовується перевірка наявності попереднього запису та виконання відповідного SQL-запиту, що або вставляє новий рядок, або оновлює існуючий.

```
def upsert_student_analysis(student_id, scope, class_id,
                            main_profile_text, career_text,
                            weak_directions_text, worsening_subjects_text):
    conn = pyodbc.connect(RAW_CONNECTION_STRING)
    cursor = conn.cursor()

    if scope == "all":
        cursor.execute("""
            SELECT id FROM dbo.student_analysis
            WHERE student_id = ? AND scope = N'all'
        """, (student_id,))
    else:
        cursor.execute("""
            SELECT id FROM dbo.student_analysis
            WHERE student_id = ? AND scope = N'current_class' AND class_id = ?
        """, (student_id, class_id))

    row = cursor.fetchone()

    if row:
        analysis_id = row.id
        cursor.execute("""
            UPDATE dbo.student_analysis
            SET
                class_id = ?,
                generated_at = SYSDATETIME(),
                main_profile_text = ?,
                career_text = ?,
                weak_directions_text = ?,
```

Рисунок 4.8 – Фрагмент коду ML модуля створення/оновлення записів в БД

Після збереження головного запису модуль переходить до заповнення підлеглих таблиць. Дані щодо оцінювання напрямків − середні бали, історичний рівень, прогнозований рівень − записуються через функцію replace_analysis_directions, яка повністю перезаписує вміст таблиці student_analysis_direction для конкретного analysis_id. Спочатку видаляються старі записи, після чого додаються оновлені значення, що гарантує узгодженість структури даних. Аналогічним чином функція replace_analysis_weak_topics здійснює оновлення таблиці student_analysis_weak_topics, у якій зберігаються всі слабкі теми, виявлені під час аналізу. Механізм роботи такий самий: повне очищення старих даних і вставлення нового набору записів, який відображає актуальний стан навчальних досягнень учня. Такий підхід забезпечує відсутність

дублювання та коректну відповідність між тематичною структурою аналізу та збереженими результатами.

Усі операції запису виконуються в межах транзакцій, що завершуються явним викликом conn.commit(). Це дозволяє гарантувати, що результати аналізу будуть збережені в БД лише у разі повного успішного виконання всіх пов’язаних операцій, що особливо важливо з огляду на атомарність даних студентського профілю.

Таким чином, механізм збереження аналітичних результатів забезпечує цілісну та структуровану інтеграцію ML-модуля з основною інформаційною системою. Дані, сформовані алгоритмом дерева рішень, потрапляють у відповідні таблиці, стають доступними в реальному часі на рівні інтерфейсу користувача та лягають в основу формування персоналізованої навчальної траєкторії.

Робота модуля машинного навчання організована як послідовний конвеєр обробки даних, у межах якого всі попередньо описані етапи формують єдину логічно завершену процедуру аналізу навчальних досягнень здобувача освіти. Після запуску основної функції main() модуль спочатку отримує з бази даних повну вибірку результатів тестування всіх учнів, включно з інформацією про предмети, теми, дату проходження тестів та додатковими атрибутами, які необхідні для інтерпретації успішності. На основі цієї вибірки формується підготовлений датасет, після чого здійснюється навчання глобальної моделі дерева рішень та відповідного масштабувальника ознак. Такий етап навчання є одноразовим для кожного циклу запуску і створює універсальну модель, здатну класифікувати рівень успішності будь-якого учня.

Після цього модуль переходить до індивідуального аналізу. Вибирається конкретний учень, щодо якого потрібно виконати обчислення, і завантажені результати тестування фільтруються відповідно до обраного режиму аналізу

(повна історія або лише поточний навчальний рік згідно з даними таблиці student_class_history). Отримана підмножина даних проходить через раніше навчений масштабувальник, після чого модель дерева рішень прогнозує рівень успішності для кожного запису. На основі цих прогнозів формується агрегована статистика: визначаються середні бали, домінантні рівні сформованості знань, освітній профіль учня, а також проводиться детальний тематичний аналіз для пошуку слабких тем по кожному предмету. Додатково модуль оцінює динаміку змін успішності, виокремлюючи предмети з можливим погіршенням результатів.

Після завершення аналітичної частини створюються текстові інтерпретації результатів − опис освітнього профілю, кар'єрні рекомендації, перелік проблемних напрямків і тем, а також пояснення можливих негативних тенденцій у навчанні. У фінальному кроці модуль здійснює запис сформованих даних у базу: основний підсумковий запис заноситься до таблиці student_analysis, агреговані показники за напрямками − до student_analysis_direction, а список слабких тем − до student_analysis_weak_topics. Усі операції виконуються через параметризовані SQL-запити, що гарантує коректність і захищеність оновлення даних. Таким чином, уся логіка − від збору даних до їх інтерпретації та збереження − реалізується в межах цілісного конвеєра, який забезпечує відтворюваність і структурованість процесу аналізу.

4.2 Інтеграція ML модуля у вебзастосунок

Інтеграція розробленого модуля машинного навчання у вже існуючий вебзастосунок здійснюється з урахуванням того, що базова система створювалася як масштабована та модульна платформа, придатна для подальшого розширення функціональності. Завдяки чітко визначеній архітектурі клієнт–сервер та структурованому доступу до даних, додавання аналітичного компонента не потребувало суттєвої перебудови основного програмного коду. ML-модуль

органічно вбудовується в логіку роботи застосунку: він отримує дані з наявних таблиць, виконує обробку результатів тестування, формує персоналізовані рекомендації та повертає їх назад у систему для подальшого використання. Така інтеграція демонструє гнучкість та адаптивність програмного рішення, дозволяючи розширити можливості платформи без порушення її загальної структури та стабільності роботи.

Інтеграція модуля машинного навчання у вже існуючий вебзастосунок потребувала розширення та адаптації серверної частини, реалізованої мовою C# у середовищі ASP.NET Core з використанням Entity Framework Core для роботи з базою даних. Оскільки новий функціонал передбачає збереження результатів аналітики та обробку додаткових навчальних даних, у проєкт було додано нові моделі даних, що відповідають створеним таблицям student_analysis, student_analysis_direction, student_analysis_weak_topics та student_class_history. Ці моделі інтегровано в клас AppDbContext, що забезпечує їх автоматичне відображення у вигляді ORM-сутностей і підтримує повноцінну взаємодію з Microsoft SQL Server за допомогою механізмів EF Core.

Для інтеграції модуля машинного навчання у вже існуючий вебзастосунок було розроблено спеціалізований контролер MLStudentAnalysisController, який забезпечує повний цикл взаємодії між серверною частиною системи, базою даних і Python-модулем аналізу успішності. Цей контролер виступає центральною ланкою, через яку вебзастосунок отримує доступ до обчислювальних можливостей ML-модуля, виконуючи при цьому роль координатора між усіма технологічними компонентами.

Першим етапом роботи контролера є звернення до бази даних, де для кожного учня зберігається попередньо сформований аналіз успішності. Контролер зчитує запис із таблиці student_analysis, перевіряючи, чи існує вже згенерований результат для даного учня та обраного режиму аналізу. Якщо такий

запис є, додатково виконується перевірка його актуальності: враховується поле generated_at, яке фіксує дату й час попереднього аналізу. Оскільки навчальні дані оновлюються нечасто, система не потребує щоразу запускати ML-модуль − достатньо впевнитися, що з моменту останнього аналізу для цього учня не відбулося змін у результатах тестування чи в історії переходів між класами. У випадку, коли дані залишаються чинними, контролер одразу повертає їх клієнтові, що суттєво пришвидшує роботу системи та зменшує навантаження на сервер.

Якщо ж результатів аналізу немає або вони втратили актуальність, контролер запускає Python-модуль. Передача параметрів − ідентифікатора учня та режиму аналізу − здійснюється через аргументи командного рядка, а сам виклик відбувається через запуск нового процесу (рисунок 4.9).

```
var psi = new ProcessStartInfo
{
    FileName = pythonExePath,
    Arguments = $"\"{scriptPath}\" {studentId} {scope}",
    UseShellExecute = false,
    RedirectStandardOutput = true,
    RedirectStandardError = true,
    CreateNoWindow = true,
    StandardOutputEncoding = Encoding.UTF8,
    StandardErrorEncoding = Encoding.UTF8
};

psi.Environment["PYTHONIOENCODING"] = "utf-8";
psi.Environment["PYTHONUTF8"] = "1";
```

Рисунок 4.9 – Фрагмент коду контролеру виклику ML модуля

Після завершення роботи модуля контролер повторно звертається до бази даних і вибірково зчитує всі створені або оновлені записи: основний результат з таблиці student_analysis, деталізовані показники рівнів успішності за напрямами з student_analysis_direction та повну структуру слабких тем із student_analysis_weak_topics. Усі ці дані об’єднуються у цілісну, структуровану

відповідь, яка включає індивідуальний освітній профіль, текстові рекомендації, прогнозовану динаміку успішності та теми, що потребують доопрацювання.

Контролер не просто викликає ML-модуль, а самостійно визначає, коли аналіз потрібно проводити повторно, а коли достатньо повернути наявні дані. Такий підхід підвищує продуктивність системи, усуває зайві обчислення та робить інтеграцію моделі машинного навчання прозорою для зовнішніх компонентів вебзастосунку. Крім того, винесення ML-модуля в окремий процес дає змогу оновлювати або повністю змінювати алгоритмічну частину без втручання в основну серверну логіку, що повністю відповідає принципам модульності, масштабованості та підтримуваності сучасних освітніх платформ.

У підсумку MLStudentAnalysisController забезпечує єдину точку входу для отримання актуального аналітичного профілю учня, синхронізує дані між двома різними технологічними стеком і забезпечує безперервну роботу всього механізму перетворення навчальних результатів у персоналізовані рекомендації. Такий механізм взаємодії дозволяє динамічно поєднувати два технологічно різних середовища − бекенд на C# та аналітичний модуль на Python − без необхідності змінювати основну архітектуру застосунку. Усі операції з базою даних виконуються через EF Core, що забезпечує типобезпечність, параметризовані запити та контроль транзакційності. У результаті серверна частина набуває ролі координатора, який збирає та надає дані модулю машинного навчання, забезпечує своєчасне оновлення результатів та передає структуровані результати клієнтській частині вебзастосунку.

Інтеграція модуля машинного навчання у клієнтську частину вебзастосунку була спрямована на те, щоб результати аналітичної обробки стали безпосередньо доступними учню та викладачу в інтерфейсі системи. Основним завданням фронтенду стало створення інтуїтивного механізму отримання даних аналізу з бази даних через бекенд-API та їх подальшого наочного відображення.

Реалізація виконана засобами JavaScript і бібліотеки React, що дозволило безболісно вбудувати нові компоненти у вже існуючу архітектуру застосунку.

Для викладача було розроблено окрему сторінку аналізу StudentPerformanceAnalysisPage. Вона під'єднується до відповідного API-методу через React-хуки й автоматично завантажує перелік класів, у яких викладає користувач. Після вибору класу компонент отримує список учнів і звертається до бекенду для отримання результатів ML-аналізу. Запит формується через Axios, а структура компонента побудована таким чином, щоб підтримувати оновлення даних після повторної генерації аналізу. Результат запиту потрапляє у стан компонента та відображається у вигляді аналітичних блоків: освітнього профілю, прогнозованих рівнів знань, слабких тем і предметів із погіршенням успішності. Викладацький інтерфейс став ключовим місцем інтеграції, оскільки дозволяє швидко переглядати аналітику щодо учнів різних класів та миттєво оцінювати загальну картину успішності групи (рисунок 4.10).

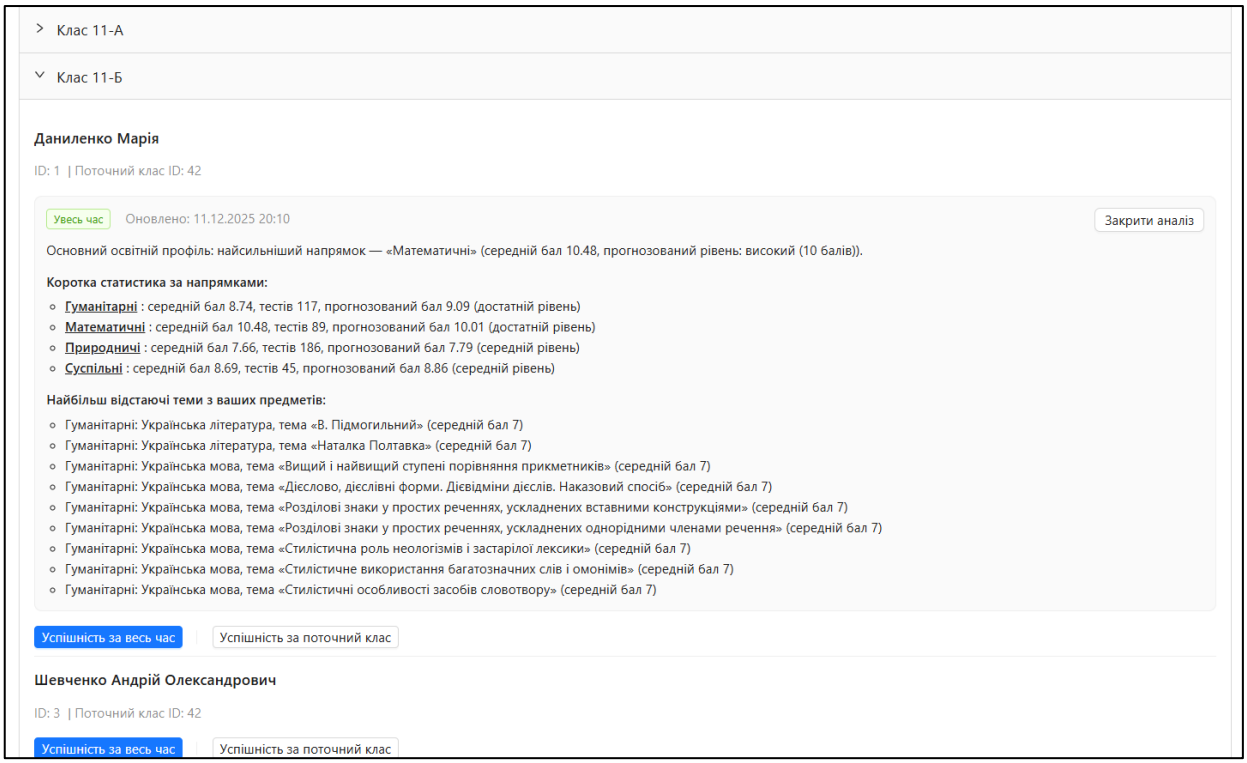


Рисунок 4.10 – Вигляд персоналізованого аналізу на сторінці аналізів вчителя

Паралельно оновлено інтерфейс особистого кабінету учня. У компонент StudentProfilePage було додано новий розділ, який містить індивідуальний аналіз, сформований ML-модулем. На відміну від викладача, учень не взаємодіє з переліком класів або списком інших користувачів − інтерфейс автоматично звертається до API лише з його власним ідентифікатором. Запит до бекенду здійснюється за тією ж схемою, що й у викладацькій частині, однак результати візуалізуються у форматі, орієнтованому на індивідуальне сприйняття: блок із рекомендаціями, короткий опис освітнього профілю, теми, які потребують повторення, а також прогноз на майбутні тестування. Для цього використано JSX-структуру, у якій кожен розділ відображається умовно, залежно від того, чи повернув модуль відповідний блок аналізу. Після отримання даних від бекенду застосунок формує розмічені секції, де зберігається текст освітнього профілю, кар'єрні поради, групування слабких тем за напрямками та предметами, а також блок погіршення результатів, який генерується ML-модулем (рисунок 4.11).

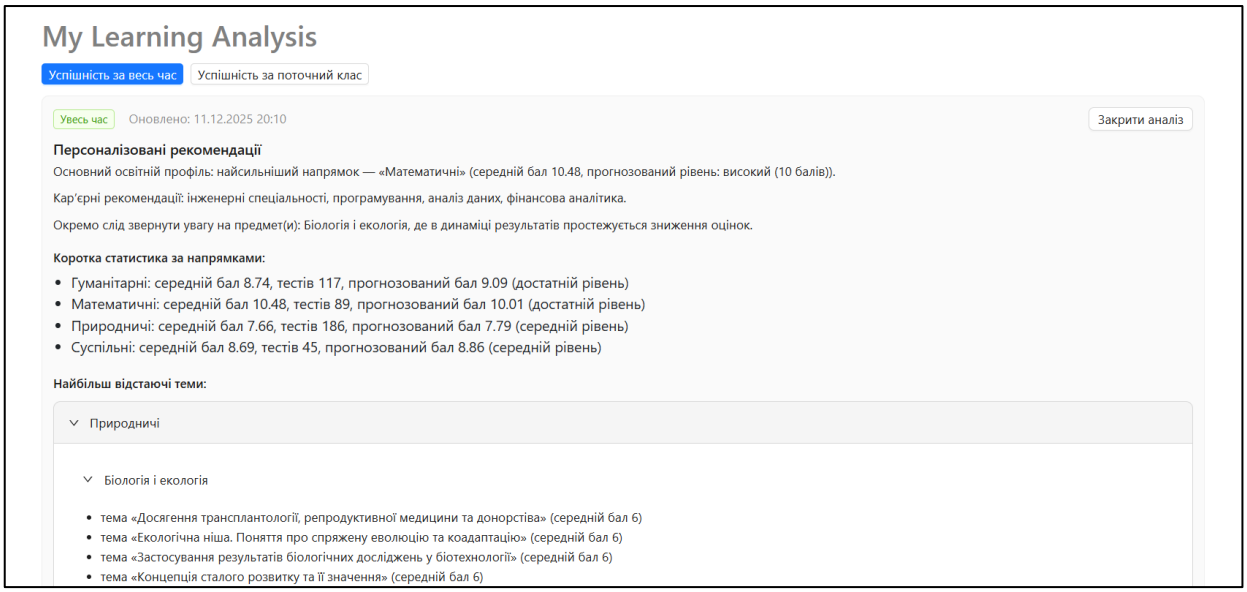


Рисунок 4.11 – Вигляд персоналізованого аналізу на сторінці профілю учня

Таке розширення фронтенду дозволило перетворити аналітичний модуль із суто серверного компоненту на повноцінний елемент користувацької взаємодії.

Учень отримує цілеспрямовані рекомендації, які допомагають вибудувати власну навчальну стратегію, тоді як викладач отримує інструмент швидкого моніторингу успішності класу. Всі зміни гармонійно інтегруються в існуючу архітектуру React-застосунку, демонструючи гнучкість як самого інтерфейсу, так і розробленого ML-модуля.

4.3 Тестування розробленого ПЗ

Тестування розробленого програмного забезпечення є важливим етапом перевірки його працездатності, надійності та відповідності функціональним вимогам, сформульованим у проєктній частині роботи [30]. Оскільки створена система включає як вебзастосунок, так і інтегрований модуль машинного навчання, процес тестування охоплює взаємодію між клієнтською та серверною частинами, коректність обробки даних, а також правильність формування аналітичних результатів і рекомендацій для здобувачів освіти.

Метою тестування є перевірка того, що всі ключові сценарії роботи системи виконуються коректно: дані про результати тестування учнів правильно зчитуються з бази даних, ML-модуль формує прогнози та рекомендації відповідно до закладеної логіки алгоритму, а вебінтерфейс коректно відображає оброблену інформацію для різних категорій користувачів. Особлива увага приділяється точності й узгодженості аналітичних даних, оскільки саме вони лежать в основі персоналізованих рекомендацій та визначення освітніх траєкторій.

Для підтвердження функціональної коректності системи проведено тестування низки базових сценаріїв, що відображають найбільш критичні етапи обробки й відображення даних.

Таблиця 4.1 – Тест-кейс «Перевірка формування та відображення аналізу успішності учня («Успішність за весь час»)»

ID	TC-SPA-001
Estimated Time	10 хвилин
Назва	Перевірка формування та відображення аналізу успішності учня («Успішність за весь час»)
Мета	Перевірити коректність формування, збереження та повторного відображення результатів аналізу успішності учня.
Передумови	Користувач має роль вчителя та валідні облікові дані
	Учень, якого буде обрано, ще не має проведеного аналізу.

№	**Крок / дія**	**Очікуваний результат**	**Результат**
1	Користувач авторизується в системі з валідними даними вчителя.	1) Авторизація успішна. 2) Відкривається сторінка профілю викладача.	Passed
2	Користувач переходить до сторінки Student Performance Analysis.	Вебзастосунок відкриває сторінку зі списками класів, у яких викладає вчитель.	Passed
3	1) Користувач обирає клас та учня, для якого раніше не проводився аналіз. 2) Натискає кнопку «Успішність за весь час».	1) Ім’я учня з’являється у відкритому списку класу з двома кнопками для аналізу під ним. 2) Під іменем учня відображається блок аналізу, що містить: – основний освітній профіль; – коротку статистику за напрямками; – найбільш відстаючі теми з предметів, які викладає користувач.	Passed
4	Користувач перевіряє дату аналізу.	Дата та час аналізу відповідають моменту натискання кнопки «Успішність за весь час».	Passed
5	Користувач оновлює сторінку.	Сторінка Student Performance Analysis знову відображає список класів викладача.	Passed

Кінець таблиці 4.1

6	1) Користувач обирає того самого учня. 2) Натискає кнопку «Успішність за весь час».	1) Учень знову успішно обраний. 2) Відображений аналіз повністю збігається з попереднім. Аналіз не перераховується повторно.	Passed
7	Користувач перевіряє дату аналізу.	Дата аналізу не змінюється та збігається з первинною датою, відображеною під час першого запуску аналізу.	Passed

Таблиця 4.2 – Тест-кейс «Перевірка відображення помилки при запуску аналізу «Успішність за поточний клас» для учня без пройдених тестів»

ID	TC-SPA-002		
Estimated Time	5 хвилин		
Назва	Перевірка відображення помилки при запуску аналізу «Успішність за поточний клас» для учня без пройдених тестів		
Мета	Перевірити, що система коректно обробляє ситуацію, коли аналіз успішності за поточний клас запускається для учня, який не проходив жодного тесту.		
Передумови	Користувач має роль вчителя та валідні облікові дані		
	Учень, якого буде обрано, ще не проходив жодного тесту.		
№	**Крок / дія**	**Очікуваний результат**	**Результат**
1	Користувач авторизується в системі з валідними даними вчителя.	1) Авторизація успішна. 2) Відкривається сторінка профілю викладача.	Passed
2	Користувач переходить до сторінки Student Performance Analysis.	Вебзастосунок відкриває сторінку зі списками класів, у яких викладає викладач.	Passed
3	1) Користувач обирає клас та учня, який не пройшов жодного тесту. 2) Натискає кнопку «Успішність за поточний клас».	1) Ім’я учня з’являється у відкритому списку класу з доступними кнопками аналізу. 2) В правому верхньому куті з’являється повідомлення про помилку: «Не вдалося отримати аналіз».	Passed

Таблиця 4.3 – Тест-кейс «Перевірка формування та оновлення аналізу успішності учня («Успішність за весь час») у профілі учня»

ID	TC-MLA-003		
Estimated Time	12 хвилин		
Мета	Перевірити коректність первинного та повторного формування аналізу «Успішність за весь час» для учня, а також оновлення даних аналізу після проходження тесту.		
Передумови	Користувач має роль учня та валідні облікові дані		
	Користувачу призначено хоча б один новий не пройдений тест.		
№	**Крок / дія**	**Очікуваний результат**	**Результат**
1	Користувач авторизується в системі з валідними даними учня.	1) Авторизація успішна. 2) Відкривається сторінка профілю учня.	Passed
2	Користувач оглядає сторінку профілю.	Профіль містить: – ім’я учня та його роль; – деталі облікового запису; – блок My Learning Analysis з кнопками «Успішність за весь час» та «Успішність за поточний клас».	Passed
3	Учень натискає кнопку «Успішність за весь час».	З’являється блок аналізу, що містить: – основний освітній профіль; – кар’єрні рекомендації; – предмети з помітним погіршенням оцінок (якщо такі є); – коротку статистику за напрямками; – найбільш відстаючі теми, згруповані за напрямками та предметами.	Passed
4	Користувач перевіряє дату аналізу	Дата відповідає даті та часу останнього аналізу.	Passed

Кінець таблиці 4.3

5	Користувач проходить призначений тест на сторінці Tests.	Виконано − тест завершено.	Passed
6	1) Користувач повертається в свій профіль. 2) Натискає кнопку «Успішність за весь час».	1) Переходи виконано. 2) Відображений аналіз оновився, відрізняється від попереднього.	Passed
7	Користувач перевіряє дату аналізу.	Дата аналізу відповідає часу повторного натискання кнопки «Успішність за весь час».	Passed

За результатами проведеного тестування було перевірено ключові сценарії використання розробленого вебзастосунку з позиції основних ролей системи − викладача та здобувача освіти. Зокрема, перший тест-кейс охоплює повний позитивний сценарій роботи викладача з аналізом успішності учня за весь період навчання, включаючи первинне формування результатів, їх збереження в базі даних та повторне відображення без повторного запуску ML-модуля. Отримані результати підтвердили коректність логіки кешування аналізу та перевірки його актуальності.

Другий тест-кейс спрямований на перевірку обробки негативної ситуації, коли аналіз успішності за поточний клас ініціюється для учня без жодного результату тестування. Система коректно ідентифікує відсутність вхідних даних для аналізу та інформує користувача про неможливість його виконання без виникнення помилок або некоректної поведінки інтерфейсу, що свідчить про надійність реалізованих механізмів обробки виняткових випадків.

Третій тест-кейс демонструє взаємодію здобувача освіти з модулем аналізу. Перевірка підтвердила, що система коректно реагує на зміну навчальних даних,

повторно виконує аналіз та відображає актуалізовані рекомендації разом з оновленою датою генерації.

Таким чином, усі розроблені тест-кейси були успішно пройдені. Це підтверджує коректність реалізації основних функціональних сценаріїв, стійкість системи до граничних ситуацій та узгоджену роботу ML-модуля з серверною і клієнтською частинами вебзастосунку.

Висновки до розділу 4

У четвертому розділі виконано практичну реалізацію вебзастосунку аналізу успішності здобувачів освіти з інтегрованим модулем машинного навчання та здійснено перевірку його коректної роботи. Реалізація охоплює повний цикл обробки навчальних даних − від отримання результатів тестування з бази даних до формування персоналізованих аналітичних висновків і рекомендацій та їх подальшого відображення користувачам системи.

Розроблено окремий ML-модуль мовою Python із використанням бібліотеки scikit-learn, у межах якого реалізовано алгоритм дерева рішень для класифікації рівня навчальної успішності. У розділі детально описано логіку підготовки вибірки, формування ознак, навчання глобальної моделі, застосування її до даних окремого здобувача освіти, а також агрегацію результатів за навчальними напрямками, визначення слабких тем і виявлення погіршення динаміки успішності. Ключові етапи реалізації проілюстровано фрагментами програмного коду, що дозволяє наочно простежити відповідність теоретичної моделі практичній реалізації.

Інтеграцію модуля машинного навчання в серверну частину вебзастосунку виконано на основі платформи ASP.NET Core з використанням Entity Framework Core для доступу до бази даних Microsoft SQL Server. У процесі реалізації було розширено модель даних шляхом додавання спеціалізованих таблиць для

збереження результатів аналітичної обробки, а також створено контролер, який відповідає за перевірку актуальності результатів аналізу, ініціацію повторного запуску ML-модуля у разі потреби та повторне використання вже згенерованих даних. Реалізовані механізми забезпечують узгоджену взаємодію між компонентами системи та оптимізують обчислювальні витрати.

На клієнтському рівні виконано інтеграцію аналітичного функціоналу у користувацький інтерфейс вебзастосунку з використанням бібліотеки React. Було реалізовано окрему сторінку для викладачів із відображенням аналітики успішності учнів у класах, які вони викладають, а також розширено профіль здобувача освіти персональним блоком аналізу навчальних досягнень. У розділі продемонстровано приклади готового користувацького інтерфейсу, що підтверджує практичну реалізацію персоналізованих рекомендацій та їх доступність для різних ролей користувачів.

Для оцінювання працездатності розробленого програмного забезпечення проведено функціональне тестування із застосуванням тест-кейсів, які охоплюють основні сценарії роботи системи для викладача й учня, а також негативний сценарій за відсутності даних для проведення аналізу. Усі наведені тест-кейси було успішно пройдено, що підтверджує коректність роботи модуля машинного навчання, серверної логіки та клієнтського інтерфейсу, а також стійкість системи до граничних ситуацій.

Таким чином, у межах розділу реалізовано, проілюстровано фрагментами коду та продемонстровано у вигляді готового вебінтерфейсу повноцінну інтеграцію модуля машинного навчання у вебзастосунок. Отримане рішення забезпечує автоматизований аналіз навчальних досягнень, підтримку персоналізованих освітніх траєкторій і підвищення ефективності управління навчальним процесом в умовах цифровізації освіти.

ВИСНОВКИ

Проведено аналіз предметної галузі та сучасних інформаційних систем, що застосовуються для підтримки навчального процесу, показав стійку тенденцію переходу від простих інструментів тестування до інтелектуальних вебзастосунків, орієнтованих на навчальну аналітику, персоналізацію та підтримку індивідуальних освітніх траєкторій. Водночас встановлено, що більшість існуючих платформ обмежується базовими механізмами персоналізації та не забезпечує глибокого аналізу динаміки успішності здобувачів освіти.

На основі аналізу аналогічних рішень обґрунтовано доцільність удосконалення наявної системи тестування шляхом інтеграції модуля машинного навчання, здатного автоматично аналізувати результати навчання та формувати персоналізовані рекомендації. Визначено основні функціональні вимоги, користувачів системи, сценарії її використання та програмно-технічне середовище реалізації.

Виконано проєктування архітектури вебзастосунку з використанням клієнт–серверного підходу та модульної структури. Засобами нотацій IDEF0, DFD та UML змодельовано функціональну структуру системи, інформаційні потоки, взаємодію компонентів і сценарії роботи користувачів, що забезпечило формалізоване та цілісне представлення логіки функціонування розроблюваного програмного забезпечення.

Обґрунтовано вибір методу дерева рішень для аналізу навчальної успішності здобувачів освіти завдяки його інтерпретованості, гнучкості та можливості педагогічного пояснення результатів. Розроблено математичну модель та алгоритмічне забезпечення для класифікації рівнів успішності й формування аналітичних висновків, що відповідають практичним потребам освітнього процесу.

Реалізовано модуль машинного навчання мовою програмування Python із використанням бібліотеки scikit-learn, який забезпечує повний цикл обробки навчальних даних: підключення до бази даних, підготовку вибірки, навчання глобальної моделі дерева рішень, застосування моделі до окремого здобувача освіти, агрегацію результатів за напрямками та темами, а також формування персоналізованих рекомендацій. Продемонстровано ключові фрагменти програмного коду, що підтверджують практичну реалізацію запропонованого підходу.

Забезпечено інтеграцію розробленого ML-модуля в наявний вебзастосунок, реалізований із використанням ASP.NET Core та React. Розширено структуру бази даних, додано серверні моделі та контролери для керування результатами аналізу, а також реалізовано механізм перевірки актуальності аналітичних даних із повторним запуском модуля машинного навчання за необхідності.

На клієнтській частині вебзастосунку реалізовано відображення результатів аналізу для різних ролей користувачів: викладач отримує узагальнену аналітику успішності учнів у своїх класах, а здобувач освіти − персоналізований аналіз власних навчальних досягнень і рекомендації щодо подальшого навчання. Це підтверджує практичну цінність розробленого рішення та його орієнтацію на реальні потреби освітнього процесу.

Коректність функціонування розробленої системи підтверджено шляхом тестування на основі підготовлених тест-кейсів, які охоплюють основні позитивні та негативні сценарії використання. Усі тест-кейси були успішно пройдені, що свідчить про стабільність роботи вебзастосунку, коректну інтеграцію модуля машинного навчання та правильну обробку граничних ситуацій.

У цілому поставлену мету магістерської кваліфікаційної роботи досягнуто: розроблено та впроваджено вебзастосунок аналізу успішності здобувачів освіти

з використанням методів машинного навчання, який забезпечує автоматизований аналіз навчальних результатів і формування персоналізованих рекомендацій. Отримані результати створюють основу для подальшого розвитку системи, зокрема в напрямі розширення набору алгоритмів аналізу, підвищення точності прогнозування та адаптації до різних освітніх контекстів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Reflections on Adaptive Learning Analytics / A. D. Sarıyalçınkaya et al. *Advancing the Power of Learning Analytics and Big Data in Education*. 2021. P. 61–84. URL: https://doi.org/10.4018/978-1-7998-7103-3.ch003 (date of access: 23.11.2025).

2) Adaptive Learning Supported by Learning Analytics for Student Teachers' Personalized Training during in-School Practices / C. Fernández-Morante et al. *Sustainability*. 2021. Vol. 14, no. 1. P. 124. URL: https://doi.org/10.3390/su14010124 (date of access: 23.11.2025).

3) Docebo − Learning Management System (LMS). URL: https://www.docebo.com (date of access: 23.11.2025).

4) EdTech Company Fishtree Selected as NextGen Most Disruptive and Overall Mobile Award Winner. Online Press Release Distribution Service | PRWeb. URL: https://www.prweb.com/releases/edtech_company_fishtree_selected_as_nextgen_most_disruptive_and_overall_mobile_award_winner/prweb11598189.htm (date of access: 23.11.2025).

5) Edmodo platform transforms learning − Education World. URL: https://www.educationworld.com/a_curr/rubin/edmodo-platform-transforms-learning.shtml (date of access: 23.11.2025).

6) Pressman R. S. Software Engineering Software Engineering: A Practitioner's Approach 6th International Edition. 8th ed. MCGRAW HILL, 2015.

7) Abana E. C. A decision tree approach for predicting student grades in Research Project using Weka. International Journal of Advanced Computer Science and Applications. 2019. Vol. 10, No. 7. P. 1–7.

8) Charbuty B., Abdulazeez A. Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends. 2021. Vol. 2, No. 1. P. 20–28.

9) Harviainen J., Lajunen E., Berg J., Koivisto M. Optimal Decision Tree Pruning Revisited: Algorithms and Complexity. 2025. 12 c. (arXiv preprint; arXiv:2503.03576). DOI: https:.doi.org/10.48550/arXiv.2503.03576.

10) Decision Trees. scikit-learn. URL: https://scikit-learn.org/stable/modules/tree.html (date of access: 23.11.2025).

11) Raschka S., Liu Y. H., Mirjalili V. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Birmingham : Packt Publishing, 2022. 719 c.

12) Plot the decision surface of decision trees trained on the iris dataset. scikit-learn. URL: https://scikit-learn.org/1.2/auto_examples/tree/plot_iris_dtc.html (date of access: 23.11.2025).

13) Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Pearson Education, Limited, 2018.

14) The unified modeling language user guide − ed. by R. James, J. Ivar. 2nd ed. Upper Saddle River, NJ : Addison-Wesley, 2005. 475 p.

15) C# 10 and . NET 6 - Modern Cross-Platform Development - Sixth Edition: Build Apps, Websites, and Services with ASP. NET Core 6, Blazor, and EF Core 6 Using Visual Studio 2022 and Visual Studio Code. Packt Publishing, Limited, 2021. 824 p.

16) ASP. NET Core in Action, Third Edition. Manning Publications Co. LLC, 2023. 1003 p.

17) Smith J. P. Entity Framework Core in Action, Second Edition. Manning Publications Co. LLC, 2021. 624 p.

18) Introducing Microsoft SQL Server 2019 − K. Gorman et al. Packt Publishing Ltd., 2019. 489 p.

19) Svekis L. L., Putten M. v., Percival R. JavaScript from Beginner to Professional: Learn JavaScript Quickly by Building Fun, Interactive, and Dynamic Web Apps, Games, and Pages. Packt Publishing, Limited, 2021.

20) Duckett J. HTML and CSS: Design and Build Websites. Wiley & Sons, Incorporated, John, 2011. 512 p.

21) Get started with Bootstrap. Bootstrap. URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/ (date of access: 23.11.2025).

22) Quick Start – React. React. URL: https://react.dev/learn (дата звернення: 23.11.2025).

23) Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p.

24) Getting Started | Axios Docs. Axios. URL: https://axios-http.com/docs/intro (date of access: 23.11.2025).

25) Overview | TanStack Query React Docs. TanStack. URL: https://tanstack.com/query/latest/docs/framework/react/overview (date of access: 23.11.2025).

26) jwt-decode. npm. URL: https://www.npmjs.com/package/jwt-decode (date of access: 23.11.2025).

27) Machine Learning with Pytorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python. Packt Publishing, Limited, 2022. 771 p.

28) Test J. Python for Beginners: A Crash Course Guide for Machine Learning and Web Programming. Learn a Computer Language in Easy Steps with Coding Exercises. Independently Published, 2020. 138 p.

29) Hao J., Ho T. K. Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral

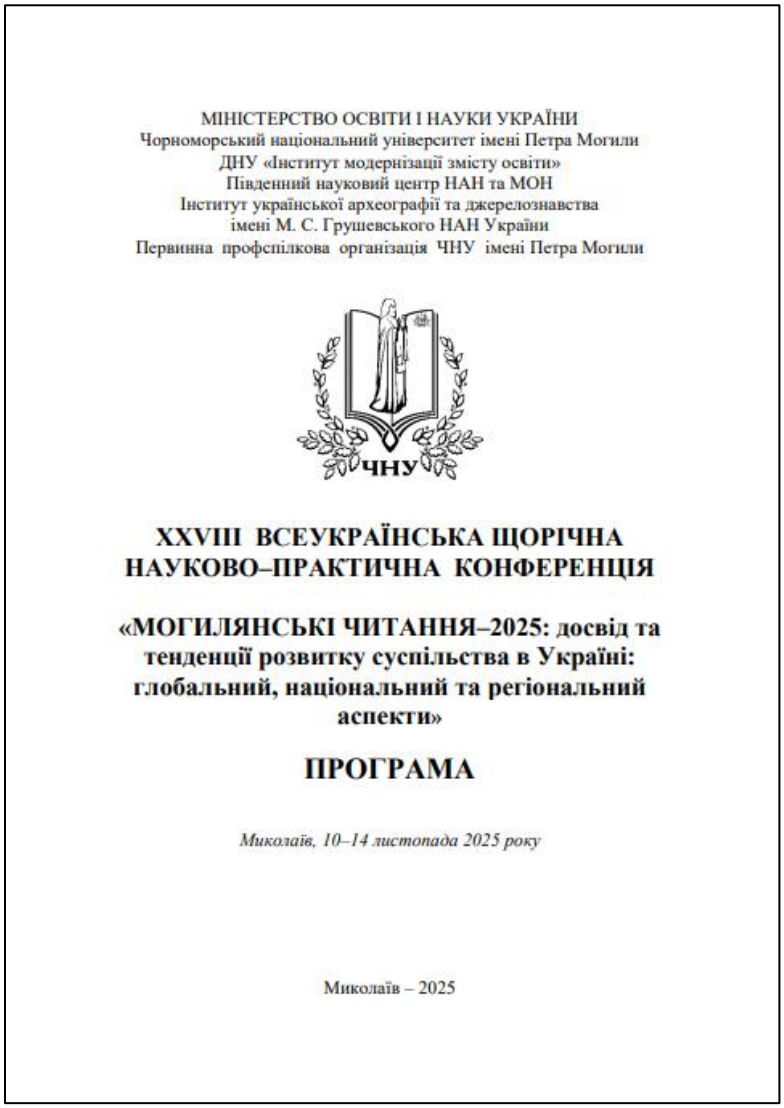
Statistics. 2019. Vol. 44, no. 3. P. 348–361. URL: https://doi.org/10.3102/1076998619832248 (date of access: 23.11.2025).

30) ISO/IEC/IEEE 29119-1:2022. Software and systems engineering — Software testing. Part 1: General concepts. — 2nd ed. — Geneva : ISO, 2022.

ДОДАТОК А

Апробація кваліфікаційної роботи

Результати досліджень були представлені на конференції Могилянські читання–2025 : досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти : XXVIII Всеукр. щоріч. наук.–практ. конф. 10–14 листоп. 2025 р., м. Миколаїв : програма / М-во освіти і науки України ; ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2025.



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ імені Петра Могили

ЧНУ

XXVIII ВСЕУКРАЇНСЬКА ЩОРІЧНА НАУКОВО–ПРАКТИЧНА КОНФЕРЕНЦІЯ

«МОГИЛЯНСЬКІ ЧИТАННЯ–2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти»

ПРОГРАМА

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

Рисунок А.1 – Обкладинка збірника тез конференції Могилянські читання – 2025