МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРОСЯННІКОВ АНДРІЙ ВАДИМОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2025 р.

РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ОБРАННЯ МОВИ ПРОГРАМУВАННЯ ТА ПЛАНУВАННЯ ПОДАЛЬШОГО НАВЧАННЯ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Юрій АНТОНОВ, доцент кафедри
інформаційних технологій,
к. фіз.-мат. н., доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2025

АНОТАЦІЯ

Просянніков А.В. Рекомендаційна система для обрання мови програмування та планування подальшого навчання. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні технології обробки даних». Донецький національний університет імені Василя Стуса, Вінниця, 2025.

Кваліфікаційна (магістерська) робота присвячена розробці рекомендаційної системи, яка визначає релевантну мову програмування та формує індивідуальний план навчання на основі відповідей користувача. У роботі проведено аналіз сучасних рекомендаційних та експертних систем, спроєктовано архітектуру програмного продукту, реалізовано модулі опитування, генерації рекомендацій та побудови навчальних траєкторій із використанням Unity, C# та Firebase.

Ключові слова: рекомендаційна система, мова програмування, персоналізоване навчання, Unity, Firebase, C#.

123 с., 4 рис., 4 табл, 51 джерело.

ABSTRACT

Prosiannikov A.V. A Recommendation System for Choosing a Programming Language and Planning Further Learning. Specialty 122 «Computer Science», Programe «Computer technologies for data processing». Vasyl Stus Donetsk National University, Vinnytsia, 2025.

The qualification (master's) thesis is devoted to the development of a recommendation system that determines the most appropriate programming language and generates a personalized learning plan based on user responses. The work includes an analysis of modern recommendation and expert systems, the design of the system architecture, and the implementation of modules for questionnaires, recommendation generation, and learning path construction using Unity, C# and Firebase.

Keywords: recommendation system, programming language, personalized learning, Unity, Firebase, C#.

123 p., 4 fig., 4 tabl, 51 sources.

ЗМІСТ

ВСТУП

Актуальність роботи:

У сучасній ІТ-галузі існує велика кількість мов програмування та напрямів, що ускладнює вибір для початківців і користувачів, які хочуть змінити сферу діяльності. Більшість освітніх платформ не надають точних індивідуальних рекомендацій щодо вибору мови програмування. Це зумовлює потребу у створенні системи, яка формуватиме персоналізовані навчальні траєкторії на основі даних користувача.

Мета дослідження: розробити рекомендаційну систему, яка визначає оптимальну мову програмування та формує персоналізований навчальний план.

Завдання дослідження:

- Проаналізувати сучасні рекомендаційні та освітні системи;
- Сформувати вимоги та постановку задачі;
- Спроєктувати архітектуру системи;
- Реалізувати програмний продукт;
- Виконати тестування та оцінити ефективність роботи системи;

Об’єкт дослідження: рекомендаційні та освітні системи, що здійснюють персоналізацію навчального процесу.

Предмет дослідження: методи та програмні засоби реалізації рекомендаційної системи для вибору мови програмування та побудови індивідуального навчального плану.

Апробація результатів дослідження: результати досліджень апробовані на IV Міжнародної науково-практичної конференції, м. Вінниця: 05 листопада 2025 р. Вінниця: ДонНУ імені Василя Стуса, 2025. під назвою «Аналіз вибору мови програмування залежно від сфери застосування та характеристик продуктивності», на VI Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених, м. Вінниця, 8 грудня 2025 р. під назвою «Застосування адаптивного тестування при побудові рекомендаційної системи».

Практичне значення дослідження: система може бути використана як інструмент для початківців при виборі напряму навчання, а також інтегрована в освітні платформи та мобільні застосунки.

Наукова новизна дослідження: полягає у розробленні комплексної моделі рекомендаційної системи, що поєднує покрокове тестування рівня знань користувача з алгоритмом визначення найбільш релевантної мови програмування. Запропоновано новий підхід до формування індивідуальної траєкторії навчання на основі багатокритеріального аналізу відповідей та адаптивних вагових коефіцієнтів. Створена архітектура системи забезпечує можливість динамічного розширення бази знань і підвищує точність персоналізованих рекомендацій порівняно з існуючими освітніми платформами.

Структура кваліфікаційної роботи: Магістерська робота складається зі вступу, трьох розділів, висновків та списку використаних джерел.

Робота містить 123 сторінки, 4 рисунки, 4 таблиці та список літератури з 51 джерела.

РОЗДІЛ 1. Огляд рекомендаційних систем та використання штучного інтелекту

1.1 Поняття та принципи роботи рекомендаційних систем

Рекомендаційна система – це інформаційно-аналітична система, що автоматично або напівавтоматично формує для користувача перелік елементів, наприклад: товарів, послуг, навчальних курсів, контенту тощо, що відповідно до роботи алгоритму, найбільше збігаються з його інтересами, потребами чи попереднім уподобанням [1]. Основна мета таких систем полягає в тому, щоб спростити процес вибору серед великої кількості варіантів і зменшити когнітивне навантаження на користувачів [2].

У контексті освітніх або навчальних систем, рекомендаційна система може виступати засобом підтримки учня або студента, підказуючи теми, курси, вправи чи шляхи вивчення мови програмування, виходячи з його рівня знань, інтересів, мети навчання або прогалин у знаннях.

Будь-яка рекомендаційна система складається з кількох підсистем, що взаємодіють між собою для збору, аналізу та представлення персональних результатів користувачу, а саме такі модулі: підсистема збору даних, база даних, модуль побудови профілю користувача, ядро рекомендацій, модуль ранжування та фільтрації, інтерфейс подання результатів, модуль оцінювання ефективності, модуль зворотного зв’язку [3].

Модуль збору даних відповідає за отримання вхідної інформації про користувача та об’єкти рекомендацій [1]. Дані можуть бути явні та неявні, явні це оцінки, лайки, вибір курсів тощо, неявні це час перегляду, кількість натискань, активність у навчальних модулях тощо. Зазвичай збір даних відбувається за допомогою інтерфейсу користувача, аналітики або зовнішніх API. API – це набір правил, інструментів та визначень, що дозволяють різним програмним системам взаємодіяти одна з одною, обмінюватися даними та використовувати функції одне одного. У контексті освітніх систем також враховуються результати тестів, рівень засвоєння матеріалу, пройдені теми, кількість спроб і час виконання

завдань. Дані попередньо очищуються, нормалізуються та передаються до бази даних.

Модуль бази даних або сховища даних є центральним елементом системи. У ній зберігається уся інформація, а саме: профілі користувачів, метадані об’єктів наприклад, курсів або мов програмування, історія взаємодій, результати попередніх рекомендацій. Залежно від обсягу й типу даних застосовуються реляційні або нереляційні бази даних. Для великомасштабних систем використовуються data warehouses та data lakes з підтримкою потокової обробки даних.

Модуль побудови профілю користувача формує індивідуальний профіль користувача, який містить інформацію про його інтереси, поточний рівень знань, поведінкові шаблони та пріоритети [1]. Профілі бувають двох типів в залежності він способу їх формування. Статичний, який заповнюється вручну під час реєстрації, або динамічний, який формується та оновлюється відповідно під час взаємодії з системою. У навчальних платформах цей модуль визначає цілі користувача, рівень складності контенту, який він вивчає, і теми, з якими виникають труднощі. Профіль користувача являється вхідними даними для основного ядра рекомендацій.

Модуль ядра рекомендацій – це центральна частина системи, що реалізує алгоритм побудови рекомендацій [1]. Воно отримує інформацію з профілю користувача та бази даних, обробляє її за допомогою обраних методів і формує список потенційно релевантних об’єктів. У сучасних системах ядро може включати окремі модулі наприклад, обчислювальний, аналітичний, модуль навчання моделі тощо.

Модуль ранжування та фільтрації отримує від ядра список можливих рекомендацій і розставляє їх у порядку пріоритетності [1]. Ранжування здійснюється за критеріями релевантності, популярності, новизни, ймовірності зацікавлення або рівня складності. Додатково в модулі відбувається фільтрація непридатних варіантів, наприклад, пройдених курсів, контенту, що не відповідає поточним цілям користувача. Багато систем використовуються комбіноване

ранжування, де результат залежить від алгоритму оцінки та інтересів користувача.

Модуль інтерфейсу подання результатів відповідає за візуалізацію рекомендацій – від списку курсів чи мов програмування до інтерактивних порад та панелі прогресу [1]. Інтерфейс має бути інтуїтивно зрозумілим, адаптивним до різних пристроїв і забезпечувати можливість зворотного зв’язку, таких як, оцінити, прийняти або відхилити рекомендацію. Завдяки зручній візуалізації підвищується рівень залученості користувача й довіри до системи.

Модуль оцінювання ефективності призначений для вимірювання якості роботи системи. Оцінка проводиться за допомогою метрик, як Precision, Recall, F1-score, MAP, NDCG, Coverage, Novelty та Diversity [1]. Модуль може працювати як у тестовому режимі, так і під час реальної роботи. У навчальних системах часто додатково оцінюють ефективність навчального процесу – середній прогрес користувача, утримання, повторне повернення.

Модуль зворотного зв’язку збирає реакцію користувача на рекомендації – явну, оцінки та коментарі, або неявну, поведінкові дії, такі як, натискання, перегляд, виконання завдань. На основі цієї інформації система коригує профіль користувача та модель рекомендацій, забезпечуючи адаптивність і самооновлення системи. У сучасних реалізаціях цей модуль є основною для реалізації навчання з підкріпленням, де система покращує свої рекомендації, спираючись на реакції користувачів [4].

Існує кілька ключових підходів до побудови рекомендаційних систем, які відрізняються способами аналізу даних, структурою моделі та способом формування рекомендацій. Найпоширеніші є контентно-орієнтований, колаборативний, гібридний, на основі знань, а також більш сучасні методи, що базуються на машинному та глибинному навчанні [2, 5, 6].

Контентно-орієнтований підхід ґрунтується на аналізі властивостей об’єктів та уподобань користувача [1]. Система створює певний вектор ознак для кожного об’єкта та користувача, після чого обчислює схожість між ними за допомогою метрик, таких як косинус подібності, кореляція Пірсона або

евклідова відстань. Наприклад, якщо користувач проходив курси які направлені на вивчення основ однієї з мов програмування, то система може запропонувати одну з найрозповсюдженіших бібліотек або фреймворків для вивчення, вона це робить за допомогою схожих тем, ключових слів чи структур.

Колаборативна фільтрація є найвідомішим і найпоширенішим підходом у сучасних систем рекомендацій [1]. Її принципи базується на закономірності, що користувачі, які мали схожу уподобання в минулому, імовірно, матимуть їх і в майбутньому. Вона поділяється на два типи, на основі пам'яті та модельно-орієнтоване.

Колаборативна фільтрація на основі пам'яті ґрунтується на обчислені схожості між користувачами або ж між об'єктами [7]. Рекомендації формуються на основі того, що подібні користувачі подобали або ж система знаходить схожі об'єкти за історією взаємодій користувачів.

Модельно-орієнтована колаборативна фільтрація використовує статистичні або машинні моделі, наприклад, матрицеву факторизацію, SVD, ALS, нейроні мережі, для прогнозування оцінок або ймовірності взаємодії між користувачем і елементом [6, 8].

Гібридний підхід – це коли гібридні системи поєднують декілька методів, щоб компенсувати недоліки кожного окремого [5]. Найчастіше поєднуються контентно-орієнтований і колаборативний методи. Зазвичай у гібридній архітектурі можуть бути реалізовані різні стратегії, а саме: зважене комбінування – результати кількох алгоритмів об'єднуються із заданими вагами, каскадне комбінування – другий алгоритм уточнює результати першого, модельне комбінування – вихід одного алгоритму слугує вхідними ознаками або даними для іншого. У сучасних освітніх платформах найкращі результати дають гібридні методи, оскільки вони дозволяють поєднати поведінкові дані користувачів з описом навчальних матеріалів [9].

Рекомендації на основі знань, цей підхід не покладається на історію користувача, а використовує базу знань і правила логічного висновку. Рекомендації формуються на основі відповідності між вимогами користувача і

характеристики об’єктів. Наприклад, якщо користувач обирає певну ціль і вказує, що має просунутий рівень, система не буде пропонувати курси для початківців, а надасть певний перелік варіантів курсів рівнем вище.

З розвитком штучного інтелекту з’явилися моделі, які використовуються підхід на основі машинного та глибинного навчання, вони навчаються автоматично розпізнавати складні зв’язки між користувачами й об’єктами. Найпоширеніші типи моделей [1]:

- Нейроні мережі для факторизації.
- Глибокі автоенкодери для зменшення розмірності матриць оцінок.
- Послідовні системи на базі рекурентних нейронних мереж.
- Трансформерні архітектури для вивчення контексту і послідовності дій користувача.
- Підсилене навчання для динамічної адаптації рекомендацій з урахуванням реакції користувача.

Такі підходи мають високу точність, здатність адаптуватися до змін користувацьких інтересів і працюють навіть за відсутності великої кількості явних оцінок.

Гібридні інтелектуальні підходи з використанням штучного інтелекту. Тенденції останніх років показують, що найефективніші системи комбінують методи глибокого навчання, обробки природної мови і знань із доменної області [10]. Такі системи можуть аналізувати контент курсів, опис мов програмування, форуми та статті, формуючи рекомендації не лише на основі даних, а й на основі сенсу та контексту навчального матеріалу. Прикладом є використання векторних подань тексту або мовних моделей для створення більш точного сенсу рекомендацій.

Розробка та впровадження рекомендаційних систем передбачає вирішення низки технічних і концептуальних проблем, пов’язаних з якістю даних, масштабуванням, динамічного користувацьких інтересів та забезпеченням

справедливості рекомендацій. Визначають кілька ключових аспектів, які безпосередньо впливають на ефективність таких систем.

Перший аспект – проблема “холодного старту”, це одна з найвідоміших проблем рекомендаційних систем, яка виникає тоді, коли у системи недостатньо даних для побудови якісних рекомендацій. Існує три основні форми цієї проблеми: новий користувач – система не має історії взаємодії користувача з об’єктами, новий об’єкт – це об’єкт, який ще не оцінений користувачами та нова система – відсутність історії взаємодій у цілому. Для їх розв’язання є декілька підходів: попереднє опитування користувача при першому вході, гібридизація – поєднання контентних і колаборативних методів, використання додаткових даних із зовнішніх джерел [11].

Другий аспект – розрідженість даних. Зазвичай матриця даних дуже розріджена, тому що користувачі оцінюють малу частину доступних елементів. Це призводить до неможливості виявити схожих користувачів, нестабільності моделей колаборативної фільтрації та збільшенню похибок при прогнозуванні оцінок [1]. Для подолання цієї проблеми зазвичай застосовують факторизацію матриць, автоенкодри для зменшення розмірності, кластеризацію користувачів або генерацію даних за допомогою поведінкових ознак інших користувачів [7, 8].

Коли збільшується кількість користувачів та об’єктів, система стає дедалі масштабнішою і обчислення рекомендацій споживає багато ресурсів. Продуктивність системи сильно залежить від об’єму даних, алгоритму їх обробки та апаратної частини. Можна виділити декілька технічних підходів для вирішення цієї проблеми, а саме: паралельна обробка даних, кешування результатів для активних користувачів та обробку за допомогою ШІ, яка навчається на офлайн даних та онлайн оновлює параметри.

З часом інтереси та поведінка користувачів змінюються, через це рекомендаційна система має бути гнучка та адаптуватись до їх уподобань. Це відбувається через зміни тенденцій та трендів у світі. Адаптувати рекомендаційну систему можна за допомогою методів онлайн навчання,

зменшення ваги старих даних, використання моделей, які можуть враховувати часовий період та його контекст.

Сучасні системи рекомендацій контенту зазвичай закриті і користувачі не мають змоги дізнатись на основі чого вони працюють, тому іноді їх довіра до них падає [1]. Для освітніх платформ надзвичайно важливо, щоб користувач розумів, чому саме цей курс або урок йому запропоновано. Для цього потрібно забезпечити чітке пояснення таких рекомендацій. Це можна робити за допомогою показу ключових характеристик обраного елементу рекомендацій, наприклад, рекомендовано, бо ви проходили курс з основ програмування на мові Python, або можна візуально показувати які навички та які знання в певних тем будуть підвищені.

При рекомендації контенту за допомогою штучного інтелекту потрібно враховувати його точність, чим більше точність, тим більша ймовірність, що запропонований контент створений ефектом “інформаційної бульбашки”. Тому система має забезпечити новизну контенту, тобто пропонувати нові теми, та різноманітність, при цьому не втрачаючи баланс між рекомендаціями та пройденими темами. Зазвичай для цього використовуються функції з великою кількістю критерій оцінки, де є коефіцієнти ваги, які визначають пріоритет для системи.

Також наявна проблема приватності та безпеки персональних даних, так як рекомендаційні системи активно використовують цю інформацію. Потрібно уникнути можливості витоку даних користувачів, зберегти анонімність профілів та зменшити ризики витоку інформації про вподобання через інші стороні сервіси [12]. Для цього було представлене рішення яке має назву “Федеративне навчання”. Це метод машинного навчання, який бере свою орієнтацію на умови, в який декілька суб’єктів спільно навчають модель, в даному методі, дані, за допомогою яких відбувається навчання, розподіляються децентралізовано. Це є його основна відмінність від звичайного машинного навчання, де дані зберігаються централізовано. Також можна використовувати теорію диференційної приватності, яка забезпечує конфіденційність персональних

даних, за допомогою введення контрольованого статистичного шуму в дані, що надає захист від викриття ідентичності користувача.

Ще одна важлива проблема – це можливість виникнення упередженості при обрані рекомендацій [12]. Іноді моделі можуть піднімати цей рівень та, наприклад, рекомендувати більш популярні курси, при цьому ігнорувати менш відомі, але якість яких вища, або надавати перевагу користувачам з певним рівнем активності. Методами для боротьби з даним явищем можуть бути методи балансування даних під час навчання та введення коефіцієнтів корекції втрат у функцію для підвищення показника справедливості.

Отже, рекомендаційна система для підбору релевантних об’єктів рекомендацій є складною системою, що включає в себе багато факторів. Комбінування підходів реалізації рекомендаційних систем дає змогу досягти високої точності, гнучкості та персоналізації результатів. Кожен з підходів має свої обмеження та й методи для їх подолання, це є основою для розробки вдалої та ефективної системи рекомендацій.

1.2 Огляд існуючих рішень у сфері освітніх рекомендацій щодо обрання та вивчення мов програмування

Експертні системи посідають важливе місце в сучасному інформаційному просторі, оскільки вони дозволяють автоматизувати процес прийняття рішень на основі формалізованих знань і логічних правил. Такі системи не обов’язково базуються на штучному інтелекті – серед них розрізняють системи на основі правил (rule-based), логічні, ймовірнісні, фреймові, нейронні, гібридні та системи, засновані на базах знань.

Експертні системи на основі правил працюють на основі набору “якщо–то” правил і активно використовуються в медицині (системи діагностики захворювань), у технічній підтримці (автоматичні консультанти) та в юридичній практиці (аналіз нормативних актів).

Ймовірнісні та статистичні системи застосовуються у сфері фінансів і страхування — для прогнозування ризиків, оцінювання кредитоспроможності клієнтів і виявлення шахрайських операцій.

Фреймові експертні системи ефективні в управлінні виробничими процесами, логістиці та енергетиці, де потрібно враховувати велику кількість взаємопов'язаних параметрів.

Нейронні та гібридні системи, які поєднують класичну логіку з елементами навчання, використовуються в сучасних навігаційних технологіях, розпізнаванні зображень і мови, системах "розумного" транспорту, робототехніці та кібербезпеці.

Окрему групу становлять експертні системи для підтримки прийняття рішень у повсякденному житті — від рекомендаційних модулів у медичних приладах, автомобільних системах і смарт-будинках до програмних рішень у банківських застосунках, пошукових платформах та онлайн-консультантах.

Таким чином, експертні системи стали невід'ємною частиною технологічної інфраструктури сучасного суспільства, забезпечуючи швидке, аргументоване та адаптивне прийняття рішень у найрізноманітніших галузях діяльності.

Початок пандемії COVID-19 у 2020 році спричинив безпрецедентний перехід навчальних процесів у дистанційний формат. Освітні установи по всьому світу були змушені швидко адаптуватись до нових умов, що стимулювало масове впровадження онлайн-платформ і технологій дистанційного навчання. За даними ЮНЕСКО [13] та ОЕСР [14], кількість користувачів відкритих онлайн-курсів у 2020-2021 роках зросла більш ніж удвічі, а глобальний ринок освітніх технологій перевищив 250 мільярдів доларів США.

Такий різкий сплеск інтересу до онлайн-освіти призвів до необхідності створення більш адаптованих і персоналізованих систем навчання. Традиційні платформи виявилися перевантаженими однотипним контентом, що не враховував індивідуальність потреб студентів, різного рівня підготовки та швидкості засвоєння матеріалу. У відповідь на це в освітньому середовищі

почали активно розвиватися рекомендаційні системи, здається аналізувати великі масиви даних про поведінку студентів, їхні результати та інтереси, щоб пропонувати релевантні навчальні матеріали, курси й завдання [1].

Освітні рекомендаційні системи сьогодні виступають важливою складовою інтелектуальних навчальних середовищ. Вони забезпечують індивідуалізацію контенту, адаптацію до рівня підготовки користувача та оптимізацію навчальної траєкторії [2]. Такі системи не лише підвищують ефективність засвоєння знань, але й сприяють підтримці мотивації, особливо в умовах самостійного онлайн-навчання.

У контексті вивчення програмування рекомендаційні механізми виконують подвійну роль. Перше, вони допомагають користувачам визначити мову програмування, що найбільше відповідає їхнім цілям, наприклад, веб-розробка, аналітика даних, мобільні застосунки тощо. Друге, формують індивідуальний план навчання, що відповідає поточному рівню знань та швидкості засвоєння матеріалу [15].

Сучасні системи цього типу базуються на аналітиці поведінки користувача, результатах тестів і застосуванні алгоритмів штучного інтелекту для динамічної персоналізації контенту. Це дозволяє не лише підвищити якість навчання, а й створити стійку модель довгострокової взаємодії студента із системою, що особливо важливо у сфері вивчення мов програмування.

Провідні освітні платформи, зокрема Coursera, Codecademy, DataCamp та Khan Academy активно впроваджують такі механізми у свої навчальні екосистеми [15]. Вони використовують адаптивні тести, аналіз поведінки користувачів та елементи штучного інтелекту для підбору індивідуального контенту [15]. Аналіз подібних рішень дозволяє зрозуміти поточний стан розвитку освітніх рекомендаційних технологій, оцінити їхні переваги та обмеження і визначити напрям удосконалення під час створення власної системи рекомендацій для вибору мови програмування та формування персоналізованого плану навчання [1].

Coursera є однією з наймасштабніших світових платформ масових відкритих онлайн-курсів, яка станом на 2024 рік налічує понад 130 мільйонів зареєстрованих користувачів і співпрацює з більш ніж 300 університетами та компаніями-партнерами. Одним із ключових чинників успіху платформи є впровадження рекомендаційної системи, що персоналізує навчальний контент для користувача. Тобто рекомендації курсів для вивчення [16].

Архітектура рекомендаційної системи Coursera базується на гібридному підході, який поєднує контенто-орієнтовану фільтрацію з алгоритмами колаборативного навчання. Система аналізує історію активності користувача, перегляди курсів, завершені завдання, оцінки, коментарі, та метадані користувача, теми, рівень складності, мова викладання, спеціалізація. На основі цих даних вона формує персональний потік рекомендацій, що оновлюється в реальному часу [17].

Coursera використовує матрицеву факторизацію та глибинні моделі векторних для пошуку подібності між курсами. Кожен курс, а також профіль користувача, представлений у вигляді вектора у багатовимірному просторі. Це має змогу системи виявити приховані зв’язки, наприклад, користувач, який завершує курс “Python for Everybody”, з великою ймовірністю зацікавиться “Data Analysis with Python” або “Machin Learning by Andrew Ng”.

У 2023 році Coursera представила оновлену систему персоналізації, що виключає генеративний штучний інтелект. Вона використовує великі мовні моделі для наступних речей:

- Формування коротких описів курсів під запит користувачів.
- Створення персоналізованих рекомендацій залежно від цілей навчання.
- Інтеграції навчального помічника на базі ChatGPT для уточнення цілей, відповіді на запити та пояснення матеріалів.

Згідно з даними Coursera Engineering Blog, алгоритм рекомендацій використовує багаторівневу систему фільтрації. На першому етапі

застосовується модель відбору кандидатів, яка швидко формує попередній список курсів. Потім ранжувальна модель із використання машинного навчання визначає ймовірності зацікавлення користувача кожним курсом. На завершальному етапі рекомендації проходять модуль персоналізації, який враховує часові та поведінкові фактори – наприклад, тривалість перерв між навчальними сесіями чи тип пристрою, з якого користувач навчається.

Важливою особливістю системи є також адаптація під різні рівні користувачів. Новачкам пропонуються базові курси, наприклад, “Programming for Everybody (Getting Started with Python)”, тоді як користувачам із досвідом – спеціалізації та професійні сертифікати, такі як: “Python Data Structures”, “Full Stack Web Developer” тощо. Це частково вирішує проблему “холодного старту” шляхом комбінування результатів опитування користувача та поведінкових даних [18].

Крім того, Coursera застосовує A/B-тестування для оцінювання ефективності нових алгоритмів. За даними досліджень платформи, впровадження персоналізованої системи рекомендацій підвищило середню тривалість навчання користувачів на 20% і збільшило кількість завершених курсів на 15%.

Таким чином, Coursera є прикладом зрілої освітньої платформи, що комплексно інтегрує методи машинного навчання, аналітику поведінки та елементи штучного інтелекту для формування персоналізованих освітніх траєкторій. Її підхід демонструє ефективність гібридної архітектури рекомендаційних систем у навчанні програмуванню та суміжних дисциплін [19].

Codecademy є однією з найвідоміших освітніх платформ для інтерактивного навчання програмуванню. Заснована у 2011 році, вона спеціалізується на практичному підході до вивчення мов програмування, включаючи Python, JavaScript, C#, Java, C++, HTML/CSS та SQL. Станом на 2024 рік платформа має понад 100 мільйонів користувачів і пропонує більш ніж 300 курсів і професійних треків.

Особливість Codecademy є використання адаптивних навчальних алгоритмів та рекомендаційних механізмів, які допомагають студенту будувати персональний шлях засвоєння матеріалу. Основна логіка системи базується на аналізі поведінкових даних користувача – кількості виконаних завдань, часу проходження уроків, рівня успішності у квізах і типів помилок, що найчастіше трапляються під час практичних вправ.

На основі цих даних формується динамічний профіль користувача, який використовується для наступних цілей:

- Визначення поточного рівня володіння мовою програмування.
- Підбір наступних тем відповідно до складності.
- Повторення матеріалу у разі низьких показників виконання.
- Рекомендацій додаткових курсів або проєктів, що розвивають конкретні навички.

Codecademy впровадила систему “Skill Paths” і “Career Paths”, які виступають своєрідними навчальними траєкторіями, побудованими за рекомендаційними принципами [19]. Наприклад, користувачу, який пройшов курс “Learn Python 3”, система пропонує продовжити навчання у треку “Data Analyst with Python” або “Computer Science Path”. Такми чином забезпечується поступовий перехід від базового рівня до спеціалізованих напрямів.

Крім того, у 2022 році було представлено оновлений механізм адаптивного повторення матеріалу, який автоматично визначає, коли саме студенту варто повторити конкретну тему, щоб уникнути забування. Цей принцип базується на психологічній моделі кривої забування Германа Еббінгауза. Алгоритм періодично нагадує користувачу про виконання практичних вправ із тем, у яких зафіксовано більшу кількість помилок або тривалу перерву між повтореннями.

Для нових користувачів Codecademy частково вирішує проблему “холодного старту” шляхом короткого опитування на етапі реєстрації. Система запитує про цілі навчання, наприклад, розробляти веб-сайти, створювати ігри, аналізувати дані, після чого формує базові рекомендації щодо вибору першої

мови програмування. Якщо користувач обирає категорію “ігри”, система пропонує C# і Unity, для “веброзробки” – HTML, CSS, JavaScript, а для “аналізу даних” – Python та SQL.

Платформа використовує елементи гейміфікації, які інтегруються в рекомендаційний механізм: за виконання завдання користувач отримує бали досвіду, бейджі та рекомендації щодо подальших кроків, наприклад, “Try building your first web app!” тощо. Такі елементи стимулюють мотивацію та підвищують рівень залучення, що особливо важливо для самостійного онлайн-навчання.

Технічно рекомендаційна система Codecademy складається з кількох модулів:

- Сховище користувацьких даних, ще зберігає результати виконаних завдань і тривалість навчальних сесій.
- Аналітичний модуль, який збирає статистику успішності й формує класифікацію користувачів за рівнем підготовки.
- Ядро рекомендацій, побудоване на контентно-орієнтованих принципах, співставлення тем і навичок між курсами.
- Модуль зворотного зв’язку, який оновлює профіль користувача на основі його активності.

У 2023 році Codecademy анонсувала інтеграцію AI-асистента, на основі моделі GPT-4, який пояснює помилки в коді та пропонує рекомендації з тем, які потрібно повторити. Це перший етап впровадження інтелектуальної підтримки навчання – не просто вибір курсу, а гнучка навігація між темами, заснована на аналізі коду користувача [20].

Таким чином, Codecademy демонструє ефективне поєднання адаптивних алгоритмів, поведінкової аналітики та елементів штучного інтелекту, що дає змогу створювати індивідуальні траєкторії навчання програмуванню. Її підхід вирізняється практичною орієнтацією, покроковою структурою та використанням повторюваного тестування для закріплення знань.

DataCamp – це освітня онлайн-платформа, орієнтована на навчання в галузях аналізу даних, програмування на Python і R, машинного навчання та науки про дані [21]. Заснована у 2013 році, станом на 2024 рік платформа має понад 12 мільйонів користувачів та понад 400 інтерактивних курсів. Її ключовою особливістю є використання адаптивних і рекомендаційних алгоритмів, що формують індивідуальні траєкторії навчання на основі рівня знань і результатів користувача.

Однією з найважливіших інновацій DataCamp є система Signal – модуль інтелектуального тестування, який визначає рівень знань користувача за 10-15 хвилин і автоматичного генерує персональні рекомендацій щодо подальшого навчання [21].

Signal базується на алгоритмах Item Response Theory та Adaptive Testing, що дозволяють динамічно добирати питання залежно від попередніх відповідей користувача. Кожне запитання має власну “вагу” та рівень складності, а система намагається знайти оптимальний баланс між простими та складними завданнями, щоб максимально точно оцінити реальний рівень навичок. Після проходження тесту користувач отримує:

- Загальний рівень знань, наприклад, Beginner, Intermediate, Advanced.
- Порівняння з іншими користувачами у відсотковому вираженні.
- Графічну діаграму сильний і слабких сторін.
- Персональний список рекомендованих курсів і практичних завдань.

Цей механізм виконує функцію початкової діагностики для побудови подальшої освітньої траєкторії. Наприклад, студент, який показав високий рівень у темах “Data Manipulation” і “Visualization”б але низький у “Statistics” або “Machine Learning”, отримує рекомендації для закриття прогалин саме у цих напрямах [22].

Signal інтегрований у загальному архітектурному DataCamp і постійно оновлюється за допомогою агрегованих даних користувачів, що дозволяє моделі навчатися та підвищувати точність прогнозів. Це створює ефект самокерованої

системи, де чим більше користувачів проходять оцінювання, тим точнішими стають рекомендації для нових студентів.

Після оцінювання DataCamp формує індивідуальний навчальний план, що складається з трьох компонентів:

1. Courses – базові навчальні модулі, наприклад, “Data Analyst with Python”, “Intermediate R”, “Data Visualization with Matplotlib”.
2. Tracks – тематичні траєкторії, що охоплюють послідовності курсів за професіє, наприклад, “Data Analyst with Python”, “Data Scientist with R”.
3. Projects – практичні завдання з реальними наборами даних, які підкріплюють вивчені теми.

Рекомендаційна система автоматично визначає, які курси або проєкти будуть найбільш релевантними на поточному етапі, та пропонує їх вигляді пріоритетного списку. При цьому враховується не лише рівень знань, а й поведінкові фактори – час, проведений на платформі, кількості спроб виконання завдань, пропуски, перерви у навчанні.

Подібно до Codecademy, DataCamp використовує механізм адаптивного повторення. Якщо користувач через певний час не повертається до вивченого матеріалу або допускає помилки в аналогічних завданнях, система пропонує короткі інтерактивні повторення. Додатково реалізовано візуальний трекер прогресу, який не лише показує виконані курси, але й оцінює рівень володіння конкретними компетенціями [21].

У результати зберігаються у базі даних користувацьких профілів, яка є джерелом для подальшого вдосконалення моделі. Система аналізує взаємозв’язки між успішністю студентів і структурою їхніх навчальних траєкторій, що дозволяє поліпшувати порядок і логіку рекомендацій.

У 2023 році DataCamp оголосила про запуск AI-асистента, який пояснює результати тестування та дає індивідуальні поради на основі аналітики Signal. Цей помічник створений за допомогою великих мовних моделей і функціонує як діалоговий інтерфейс: користувач може запитати, чому система рекомендує

певний курс або тему. Це значно підвищує пояснюваність рекомендацій, що є важливим аспектом довіри користувача до системи.

Крім того, DataCamp, так само як Coursera, застосовує A/B-тестування для оцінки точності рекомендацій. За даними компаній, використання Signal збільшило середній рівень завершення навчальних треків на 18% і зменшило відмов користувачів від навчання на 12%.

Приклад комплексної рекомендаційної системи це те, що демонструє підхід DataCamp, діагностика знань аналітики навчальної поведінки та адаптивне тестування інтегровані в єдину структуру. Завдяки цьому користувач отримує динамічний навчальний маршрут, який змінюється відповідно до його успішності. А система Signal може вважатися однією з найбільш ефективних реалізацій рекомендацій у сфері навчання програмуванню.

Ще один представник це Khan Academy. Це некомерційна освітня платформа, яка заснована у 2008 році, вона надає безкоштовний доступ до навчальних матеріалів у різних галузях – від математики та фізики до програмування, економіки і гуманітарних наук. Її місія полягає у створенні персоналізованого навчального середовища, доступного для всіх користувачів незалежно від віку, країни чи рівня підготовки [23].

У межах курсів із програмування Khan Academy використовує адаптивні рекомендаційні механізми, що допомагають формувати індивідуальні шляхи навчання. Рекомендації базуються на результатах виконаних завдань, активності користувачів та швидкість проходження навчальних модулів.

Після реєстрації система створює профіль користувача, який включає поточний рівень знань, досягнень, історії виконаних вправ і оцінку засвоєння тем. На основі цих даних формується панель прогресу, де зазначено теми, рекомендовані до повторення або подальшого опрацювання. Для студентів, які демонструють високі результати, відкриваються додаткові розділи підвищеної складності або пропонуються проєктні завдання [23].

Логіка рекомендацій на платформі реалізована через певні модулі, а саме: навчальний план, модуль прогресу, система підкріплення знань.

1. Навчальний план – це набір тем, розташованих у логічній послідовності. Система автоматично пересуває користувача вперед, якщо він успішно виконав попередні завдання, або повертає до повторення тем у разі низьких результатів.
2. Модуль прогресу, він відображає поточні досягнення, рівень володіння матеріалом і динаміку навчання. Використовується для побудови індивідуальних рекомендацій, які враховують не лише оцінки, а й час виконання, кількість спроб і пропущені теми.
3. Система підкріплення знань – це механізм, який відстежує, наскільки впевнено студент володіє кожною темою. Якщо користувач протягом часу не практикувався або робив помилки, система автоматично повертає завдання до категорії “потрібно повторити”.

Такі підходи дозволяють реалізувати індивідуальний темп навчання, що є одним із ключових принципів Khan Academy. Замість того, щоб усі студенти проходили однаковий набір уроків у фіксованому порядку, система дозволяє адаптувати темп і зміст навчання під кожного користувача.

У контексті програмування це означає, що платформа не просто подає послідовність тем, а реагує на типові помилки. Наприклад, якщо користувач не розуміє концепції циклів чи функцій, система пропонує короткі повторювальні завдання й інтерактивні пояснення до повернення стабільного результату. Такий механізм поєднує елементи педагогічної аналітики та алгоритмічної персоналізації, без потреби у складних моделей штучного інтелекту.

Khan Academy також підтримує вчительський режим, який дозволяє викладачам бачити рівень засвоєння матеріалу кожним студентом і вручну задавати персональні завдання [15]. Цей підхід поєднує автоматичну рекомендаційну логіку з педагогічним контролем, що робить систему гнучкою та придатною як для самостійного, так і для формального навчання.

Загалом, ця платформа демонструє збалансоване поєднання алгоритмічної адаптивності та класичних методів навчальної аналітики. Її рекомендаційні функції допомагають студентам рухатися у власному темпі, повертатися до

складних тем і поступово формувати фундаментальні знання. Такий підхід підвищує якість засвоєння матеріалу та забезпечує більш стійке навчальне зростання в порівнянні з лінійними курсами.

Проведений аналіз показав, що після 2020 року розвиток онлайн-освіти значно прискорився, а провідні освітні платформи почати активно інтегрувати рекомендаційні системи як основу персоналізованого навчання. Пандемія COVID-19 стала каталізатором для появи великої кількості цифрових освітніх інструментів і сформувала новий стандарт взаємодії між користувачем і навчальним контентом, тобто індивідуалізацію освітнього процесу.

Розглянуті платформи реалізують різні, але взаємодоповнювальні підходи до персоналізації:

- Coursera використовує гібридну модель рекомендацій, поєднуючи контент-орієнтовані алгоритми з колаборативною фільтрацією та аналітикою поведінки користувача. Це дозволяє пропонувати курси відповідно до освітніх цілей, рівня складності та історії навчання.
- Codecademy застосовує адаптивне навчання, побудоване на аналізі активності користувачів, результати виконання вправ і рівня успішності. Її система “Skill Paths” та “Career Paths” є прикладом реалізації рекомендаційної логіки через побудову навчальних траєкторій.
- DataCamp інтегрує адаптивне тестування Signal, яке оцінює навички користувача й автоматично генерує персональний план навчання. Цей підхід демонструє ефективність моделі оцінювання, рекомендація, повторення у форматі навчальних маршрутів.
- Khan Academy поєднує педагогічну аналітику з алгоритмічною адаптивністю, забезпечуючи гнучке керування темпом і складністю навчання без повної автоматизації процесу.

  Загальними рисами всіх платформ є:

1. Використання поведінкових і контентних даних для побудови профілю користувача.
2. Поступовий підхід від статичних до динамічних навчальних планів, які оновлюються в реальному часі.
3. Застосування елементів гейміфікації та адаптивного повторення, що підтримують мотивацію.
4. Інтеграція аналітики навчальної ефективності як механізм оцінювання якості рекомендацій.

Також, усі розглянуті системи мають обмеження. Вони орієнтовані переважно на великі навчальні бази даних, не завжди враховують початкові професійні цілі користувача, а також не пропонують повністю автоматизованої системи формування навчального плану для вибору мови програмування. Це створює простів для подальших досліджень і вдосконалення, зокрема, розроблення інтелектуальної рекомендаційної системи, яка поєднуватиме діагностику рівня знань, визначення освітніх цілей та побудову персонального навчального маршруту.

Отже, сучасні освітні платформи демонструють значний прогрес у впровадженні рекомендаційних технологій, але залишаються передумови для розвитку більш гнучких, адаптивних і цільових систем у сфері вибору сови програмування та побудови подальшого плану навчання.

1.3 Інтеграція штучного інтелекту у рекомендаційні системи

Штучний інтелект є галуззю комп’ютерних наук, що досліджує методи створення систем, здатних виконувати завдання, які традиційно потребують людського інтелекту, а саме: сприйняття, навчання, розуміння мови, логічне мислення та прийняття рішень [24]. Його еволюція охоплює понад сім десятиліть і поділяється на декілька етапів, кожен з яких характеризувався власними концепціями, технологіями та підходами.

Перші ідеї про створення “розумних машин” з’явився ще у 1940-х роках, коли почали формуватися основи кібернетики та теорії обчислень. У 1943 році

Воррен Маккаллок і Волтер Піттс представили першу модель штучного нейрона, що імітував роботу біологічного нейрона у спрощеній формі. У 1950 році Алан Тюрінг у свої праці “Обчислювальні машини та розум” запропонував тест, відомий як тест Тюрінга, спосіб оцінити, чи може машина демонструвати інтелектуальну поведінку, подібну до людської [25]. Він запропонував замінити складне питання “Чи може машина мислити?” на більше практичне – “Чи може машина поводитись так, що людина не зможе відрізнити її від людини?”. Для цього він описав імітаційну гру у якій беруть учать три учасники:

1. Людина А – людина;
2. Людина В – машина;
3. Людина С – суддя, людина, яка не бачить інших двох;

Суддя спілкується з А і В лише за допомогою текстових повідомлень. Якщо суддя не можу впевнено визначити, хто з двох є машино, то машина вважається такою, що пройшла тест Тюрінга.

Сьогодні системи, такі як ChatGPT, Google Gemini, Claude тощо, вже частково проходять тест Тюрінга в неформальних умовах – багато користувачів не відразу помічають, що спілкуються з штучним інтелектом. Проте в науковому сенсі тест Тюрінга вже не вважається достатнім критерієм інтелекту, оскільки не оцінює розуміння, креативність, емоційність чи етичність.

Ці ідеї заклали основу для розуміння інтелекту як процесу обробки інформації, що може бути реалізований у машинах.

Термін “Штучний інтелект” був офіційно введений у 1956 році на Дартмутській конференції у США, яку організували Джон Маккарті, Марівн Мунський, Клод Шеннон та Аллен Ньюелл [26]. Саме цей момент вважають початком наукової дисципліни штучного інтелекту.

Перші дослідження зосереджувалися на створенні символьних систем, які використовували логічні правила для моделювання процесів мислення. У 1957 році Френк Розенблатт розробив перцептрон – першу навчальну нейрону мережу. У 1960-х роках з’явилися експертні системи, здатні приймати рішення в межах певної предметної області. У 1966 році Джозеф Вейценбаум створив

програму ELIZA, яка імітувала діалог психотерапевта – один із перших прикладів обробки мови.

Хоча можливості цих систем були обмеженими, вони заклали основу для подальших досліджень у сфері знань, логічного виводу та мовного аналізу.

У 1970-х роках стало зрозуміло, що символьні методи мають значні обмеження, вони вимагали великих обсягів знань, а їх узагальнення було складним. Недостатня обчислювальна потужність комп'ютерів того часу не дозволяла реалізувати складні моделі. Як наслідок перший період спаду інтересу до штучного інтелекту, відомий як "Зима штучного інтелекту".

У 1980-х роках ситуація частково змінилась завдяки появі експертних систем. Які успішно застосовувалися в промисловості. Проте їх розробка вимагала ручного створення правил, що знову обмежувало масштабування технології.

Справжній прорив відбувся у 1990-х роках завдяки поширенню методів машинного навчання – підходу, у якому система не програмується безпосередньо, а навчається на основі даних. Активно розвивалися алгоритми рішучих дерев, методи опорних векторів, кластеризації та нейронні мережі з кількома шарами. У 1997 році комп'ютер Deep Blue від IBM переміг чемпіона світу з шахів Гарі Каспарова, продемонструвавши інтелекту в стратегічному мисленні. У 2000-роках розвиток інтернету й збільшення обсягів даних спричинили появу терміну "Big Data", що стало новою основою для навчання моделей [24].

Цей етап ознаменував перехід від теоретичних моделей до практичних застосувань, від пошукових систем і рекомендаційних алгоритмів до розпізнавання зображень і мови.

Після 2010 року розвиток штучного інтелекту прискорився завдяки зростанню обчислювальних можливостей графічних та тензорних блоків обробки, доступності великих наборів даних і вдосконаленню архітектур нейронних мереж. У 2012 році модель AlexNet перемогла у конкурсі ImageNet, зменшивши похибку класифікації зображень майже вдвічі, що стало початком

ери глибинного навчання. З’явилися рекурентні та конволюційні нейроні мережі, які досягали значних успіхів у задачах обробки мови, зору та рекомендацій. Компанії Google, Facebook, Amazon та інші почати активно впроваджувати глибинне навчання у свої продукти, від перекладачів і віртуальних асистентів до систем персоналізації контенту [24].

Саме в цей період рекомендаційні системи почали переходити від простих фільтраційних методів до інтелектуальних гібридних моделей, що навчаються безперервно на основі поведінки користувачів.

Сучасний етап розвитку штучного інтелекту характеризується появою великих мовних моделей і генеративних систем. У 2020 році компанія OpenAI представила модель GPT-3, яка продемонструвала здатність розуміти контекст і генерувати осмислений текст на людському рівні. У 2022-2023 роках поширення моделей GPT-4, Claude, Gemini та LLaMa, ознаменувало перехід до мультимодальних систем, що працюють із текстом, зображеннями, кодом і мовленням. У сфері освіти ці моделі стали основою для створення інтелектуальних навчальних систем, адаптивних наставників і автоматичних систем підтримки навчання, здатних аналізувати рівень знань і пропонувати персональні навчальні траєкторії.

Завдяки генеративному штучному інтелекту сучасні рекомендаційні системи отримали можливість не лише пропонувати готовий контент, а й створювати новий, наприклад, генерувати тестові завдання, підказки чи навчальні пояснення з урахуванням стилю користувача.

Еволюція штучного інтелекту, від символічних моделей середини 20-го століття до сучасних глибинних і генеративних ахітектру, безпосередньо вплинула на розвиток рекомендаційних систем. З появою методів машинного навчання у 1990-х роках, а згодом і глибинного навчання у 2010-х, рекомендаційні технології перейшли від простих статичних підходів до інтелектуальних моделей, здатних самостійно виявляти закономірності, передбачати інтереси користувачів і адаптуватися до зміни їх поведінки.

Інтеграція штучного інтелекту дала змогу рекомендаційним системам перетворитися на адаптивні аналітичні інструменти, які враховують контекст, освітні цілі та індивідуальний рівень підготовки користувача. Цей перехід став основою сучасних освітніх платформ, у яких рекомендаційні механізми не лише добирають контент, а й допомагають формувати персоналі траєкторії навчання.

Штучний інтелект став ключовим чинником еволюції рекомендаційних систем, забезпечивши можливість обробки великих обсягів даних, виявлення прихованих закономірностей і формуванню більш точних, контекстно залежних рекомендацій. Якщо традиційні алгоритми фільтрації спиралися переважно на статистичні методи, то сучасні системи використовують машинне навчання, глибинні нейроні мережі та обробку природної мови, що дозволяє їм адаптуватися до динамічної поведінки користувача і контенту [26].

Методи машинного навчання дозволяють системам автоматично знаходити закономірності між діями користувачів і характеристиками об’єктів. Найпоширенішими моделями можна назвати регресійні алгоритми, рішучі дерева, кластеризацію та байєсівські підходи.

Регресійні алгоритми машинного навчання – це група алгоритмів, які використовуються для прогнозування неперервних числових значень. Основна ідея регресії полягає в тому, щоб знайти залежність між незалежними змінними та залежною змінною. Регресійні алгоритми є декількох типів, а саме: лінійна, поліноміальна, логістична, за допомогою дерев рішень, градієнтне підсилення тощо.

Рішучі дерева – це один із базових і водночас потужних методів машинного навчання, який використовується як для класифікації, так і для регресій. Це структура у вигляді дерева, де є вузли, які представляють перевірку певної ознаки, гілки – можливі результати цієї перевірки та листки – остаточні рішення або передбачення. Алгоритм поступово ділить простір ознак на менші підмножини, щоб підвищити однорідність об’єктів у кожній з них.

Кластеризація – це метод без учителя у машинному навчанні, який використовується для групування об’єктів у кластери таким чином, щоб об’єкти

в одному кластері були схожі між собою за певними ознаками, а між кластерами – суттєво відрізнялися. Метою кластеризації є виявлення прихованої структури у даних, без наявності заздалегідь відомих міток або категорій. Алгоритм самостійно знаходить закономірності та формує групи на основі подібності між об’єктами [1].

Градієнтне посилення – це модуль методу машинного навчання, яка належить до класу ансамблевих методів, тобто таких, що поєднують кілька слабких моделей, зазвичай дерев рішень, у потужну предиктивну модель. Ідея полягає у покроковому побудуванні ансамблю слабких моделей, де кожна нова модель намагається виправити помилки попередніх.

У навчальних платформах машинне навчання застосовується для діагностики рівня знань студента, прогнозування його успішності, формуванню профілю користувача і автоматичного добору навчальних матеріалів.

Глибинне навчання стало наступним етапом інтеграції штучного інтелекту у рекомендаційні системи [25]. Його архітектура, нейронні колаборативні фільтри, автоенкодери, рекурентні мережі та механізми уваги, дозволяють врахувати послідовність дій користувача, часові залежності та контекст.

Моделі глибинного навчання здатні виявляти приховані зв’язки між користувачами та об’єктами навіть у великих, розріджених наборах даних [1]. Це забезпечує високу точність персоналізації, оскільки система не просто співставляє подібні курси, а прогнозує наступний крок користувача у навчанні.

Іншим важливим напрямом є використання обробки природної мови. Сучасні рекомендаційні системи аналізують великі обсяги текстового контенту: описи курсів, коментарі користувачів, навчальні матеріли. Завдяки йому можливо класифікувати курси за темами та рівнем складності, знаходити семантичку схожість між матеріалами, аналізувати відгуки студентів для виявлення корисності чи проблемності курсу, автоматично генерувати короткі описи або підказки.

Застосування мовних моделей дозволяє системам не лише підбирати релевантні ресурси, а й зрозуміло пояснювати причину рекомендацій, що підвищує довіру користувачів [27].

Підхід підсиленого навчання розглядає рекомендаційну систему як агента, який взаємодіє з користувачем у циклі “дія, реакція, винагорода”. Якщо користувач приймає рекомендацію, виконує завдання або завершує курс, система отримує позитивний сигнал і підвищує вагу подібних рекомендацій у майбутньому.

Таким чином методи підсиленого навчання дозволяють оптимізувати довгострокову ефективність навчання, а не лише короткострокову точність прогнозів. Дослідження показують, що використання підсиленого навчання підвищує рівень утримання студентів і збільшує завершення курсів на 10-15% [1].

Впровадження штучного інтелекту у рекомендаційні системи забезпечує:

- Високу точність і релевантность рекомендацій;
- Персоналізацію в реальному часі, що змінюється відповідно до прогресу користувача;
- Адаптивне оновлення навчального плану без участі викладача;
- Прогнозування навчальних результатів і попередження труднощів у засвоєнні матеріалу;
- Зменшення когнітивного навантаження та підвищення мотивації студентів;

Водночас інтеграція штучного інтелекту породжує нові виклики: забезпечення пояснюваності алгоритмів, захисту персональних даних і етичності використання моделей, що залишаються актуальними напрямами досліджень у сучасних освітніх системах.

Таким чином, штучний інтелект виступає не лише інструментом підвищення точності рекомендацій, а й основою для побудови самонавчальних,

контекстно-адаптивних систем, які здатні формувати індивідуальні освітні траєкторії, підтримуючи користувача протягом усього процесу навчання.

1.4 Приклади застосування ШІ для персоналізованого навчання програмуванню

Сучасний розвиток освітніх технологій показує, що штучний інтелект поступово стає основним рушієм персоналізованого навчання. Застосування моделей машинного навчання, нейронних мереж та обробки природної мови дає змогу адаптувати контент, підлаштувати складність завдань і пропонувати навчальні траєкторії, які відповідають поточним знанням і цілям користувача [10,28]. Особливо активно такі рішення впроваджуються у сфері вивчення мов програмування, де потрібен безперервний зворотний зв’язок, аналіз коду та практичні вправи [28].

Платформа Coursera активно інтегрує штучний інтелект у систему персоналізованого навчання, зокрема для курсів і спеціалізацій, пов’язаних із програмуванням, аналізом даних і розробкою програмного забезпечення [16]. Основна мета використання штучного інтелекту на платформі – це пристосування навчального процесу до індивідуальних потреб студента через аналіз його активності, процесу та цілей навчання.

З 2023 року Coursera реалізувала комплексну екосистему персоналізації за допомогою штучного інтелекту, що включає декілька ключових компонентів, а саме [17]:

1. Рекомендаційний модуль на основі глибинного навчання

   Система аналізує:

   - Історію пройдених курсів;
   - Темп навчання;
   - Середній бал за завдання;
   - Кількість повторних спроб тестів;

На основі цих параметрів нейрона мережа формує персональний список курсів і практичних завдань, які найбільш відповідають рівню користувача. Якщо студент завершив базовий курс, то система пропонує курси вищого рівня.

2. Помічник для навчання

У мережах експериментальної програми “Coursera Coach” було впроваджено інтерактивного помічника, який використовує мовні моделі для персональної підтримки студентів.

- Пояснювати складні фрагменти лекцій простою мовою;
- Підсумовувати відеоуроки у вигляді коротких конспектів;
- Відповідати на запитання щодо синтаксису, логіки або коду з прикладів;
- Пропонувати додаткові ресурси для самостійного опрацювання;

Таким чином, Coursera Coach виступає як інтелектуальний наставник, який допомагає студенту розібратися з матеріалом без необхідності звертатися до викладача.

3. Система динамічного оцінювання та адаптації навчального плану

Алгоритми Coursera автоматично змінюють послідовність тем залежно від того, як студент справляється з попередніми розділами. Якщо користувач систематично помиляється у завданнях, система знижує складність наступних тестів і пропонує додаткові матеріали чи короткі повторювальні вправи. У разі високих результатів навчальний шлях скорочується, студент отримує доступ до складніших проєктів або сертифікаційних модулів.

4. Персональні навчальні шляхи

Coursera застосовує методи кластеризації та колаборативного навчання для побудови траєкторій, подібних до “кар’єрний шлях” або “навчальний план”. Наприклад, для користувача, який вивчає

Python із метою подальшого переходу в Data Science, система автоматично формує наступну послідовність курсів: основи Python, Pandas та NumPy, візуалізація даних, кероване машинне навчання. Це дозволяє реалізувати поступове ускладнення навчання відповідно до рівня підготовки.

5. Прогнозована аналітика успішності

   Coursera використовує моделі машинного навчання для прогнозування ймовірності завершення курсу. Якщо алгоритм виявляє зниження активності, студенту надсилаються мотиваційні повідомлення або рекомендації щодо матеріалів, які допоможуть відновити прогрес. Опираючись на дані Coursera Engineering Team від 2023 року, використання персоналізації за допомогою штучного інтелекту знизило кількість незавершених курсів приблизно на 18% [18].

Завдяки цим інструментам Coursera перетворює традиційні онлайн-курси з програмування на адаптивне інтелектуальне середовище, у якому студенти отримують підтримку, зворотній зв’язок і рекомендації, максимально наближені до індивідуального навчання з наставником людиною.

Система на основі штучного інтелекту забезпечує як персоналізацію контенту, так і когнітивну адаптацію, підбір стилю пояснень і темпу навчання під кожного окремого користувача.

Платформа Codecademy є одним із найвідоміших прикладів використання штучного інтелекту у сфері персоналізованого навчання програмуванню. Основна ідея підходу полягає у створенні інтерактивного середовища, яке адаптується до рівня користувача в реальному часі, аналізуючи його код, помилки, темпи навчання та результати виконання завдань [19].

1. Інтелектуальний аналіз коду та автоматичний зворотній зв’язок

   Система Codecademy використовує алгоритми машинного навчання для автоматичного аналізу коду користувача. Коли студент

виконує завдання у вбудованому редакторі, модуль штучного інтелекту оцінює:

- Логіку виконання програми;
- Типові синтаксичні або логічні помилки;
- Ефективність використаних конструкцій, наприклад, циклів, умов, функцій тощо;

На основі цього формується персональний зворотній зв'язок. Якщо користувач часто припускається один і тих самих помилок, система пропонує спрощені приклади або короткі повторювальні вправи з відповідних тем. Наприклад, у курсі "Learn JavaScript" користувач, який двічі помилився при роботі з функціями, отримує автоматичне повідомлення по типу: "Спробуйте спочатку виконати коротке завдання з оголошення функцій – це допоможе закріпити базовий синтаксис".

2. Адаптивні навчальні траєкторії

Codecademy застосовує машинне навчання для побудови індивідуальних навчальних маршрутів. Після реєстрації користувач обирає мету, наприклад, стати фронтенд-розробником, навчитися аналізу даних, опанувати Python для автоматизації тощо. Система використовує ці дані разом із показниками успішності для формування персоналізованого відслідковування, який складається з курсів, справ і проєктів.

Якщо користувач добре справляється з базовими темами, система пропонує перехід до складніших концепцій, наприклад від змінних до циклів, від циклів до об'єктів та класів, від об'єктів та класів до асинхронного програмування. У разі виникнення труднощів алгоритм тимчасово сповільнює прогрес, пропонуючи додаткові тренувальні модулі або відеопояснення.

3. Механізм інтервального повторення

Для підвищення довгострокового засвоєння матеріалу у 2022 році Codecademy інтегрувала систему інтервального повторення, яка базується на психологічній моделі кривої забування Германа Еббінгауза [29]. Алгоритм штучного інтелекту відстежує час, який минув від останнього повторення теми, і прогнозує, коли рівень запам'ятовування почне знижуватися. У ці моменти користувачу надсилаються нагадування про повторення або пропонуються короткі інтерактивні вправи. Цей механізм дозволяє підтримувати стабільне засвоєння синтаксису, а також формує звички системного навчання без перевантаження пам'яті.

4. Асистент для навчання програмуванню

У 2023 році Codecademy анонсували асистента на основі штучного інтелекту, створеного у співпраці з OpenAI, який допомагає аналізувати код та виправляти помилки [20]. Помічник інтегрований безпосередньо в навчальне середовище та функціонує як інтерактивний наставник.

Основні можливості:

- Пояснення помилок у коді звичайною "людською" мовою;
- Пропозиція виправлень або прикладів альтернативних рішень;
- Відповіді на запитання, пов'язані з концепціями мови програмування;
- Генерування коротких порад щодо покращення стилю коду;

Користувач може запитати, наприклад, "Чому цей цикл не працює?" – і отримати пояснення, "Умова *i <= array.length* викликає помилку, оскільки індексація починається з нуля. Спробуйте *i < array.length*".

Таким чином процес навчання наближається до індивідуального наставництва, але реалізується повністю автоматично.

5. Адаптивне оцінювання та рекомендації

Codecademy використовує моделі прогнозування успішності на основі машинного навчання. Алгоритм оцінюють імовірності завершення курсу, рівень залученості користувача, час між сесіями та складність поточних тем.

Якщо модель виявляє зниження активності, користувачу пропонуються коротші модулі або повідомлення-мотивації. Крім того, у розділі “Мій навчальний план” студент бачить персональні рекомендації, які оновлюються після тесту або практичного завдання.

У результаті використання штучного інтелекту платформа Codecademy створює динамічне середовище навчання, яке реагує на кожен крок користувача. Алгоритми машинного навчання та мовні моделі забезпечують індивідуальні пояснення, а адаптивне відслідковування та повторення підтримують стійке засвоєння матеріалу. Такий підхід підвищує ефективність навчання програмуванню та забезпечує поступове формування професійних компетентності без необхідності зовнішнього контролю викладача.

Платформа DataCamp спеціалізується на навчанні аналітики даних, програмуванню на Python, R і SQL, машинному навчанню та Data Science [21]. Вона стала однією з перших освітніх екосистем, які системно інтегрували штучний інтелект у навчальний процес, забезпечуючи персоналізацію навчальних напрямів, автоматичне оцінювання навичок і підтримку студента в реальному часі.

Основною інновацією DataCamp є поєднання адаптивного тестування, аналітики поведінки користувачів і моделей машинного навчання, що дозволяє побудувати динамічну систему навчання, яка підлаштовується під темп і рівень кожного користувача.

1. Система Signal – інтелектуальне оцінювання рівня навичок

Найвідомішим прикладом використання штучного інтелекту на платформі є DataCamp Signal – модуль адаптивного тестування,

що аналізує навички програмування за допомогою алгоритмів машинного навчання.

Система використовує принципи теорії відгуку завдань та Баєсовських простеження знань, що дозволяє визначити рівень володіння окремими темами, такими як змінні, цикли, робота з даними чи функції [9].

Алгоритм адаптує тести під користувача за таких умов:

- Якщо студент правильно відповідає на складне запитання, наступне завдання стає ще складнішим.
- Якщо робить помилку, то система переходить до простіших або пояснює попереднє рішення.
- Після завершення тесту формується профіль навичок, де відображено сильні та слабкі сторони, а також індивідуальні рекомендації для подальшого навчання..

Система Signal виконує роль інтелектуального діагноста, який швидко визначає прогалини в знаннях і пропонує найоптимальніший шлях для їх подолання. Завдяки машинному навчанню DataCamp постійно вдосконалює точність алгоритму, кожна нова спроба проходження тесту слугує для моделі додатковими тренувальними даними.

2. Персоналізовані навчальні траєкторії

Після діагностики Signal система створює індивідуальний план навчання, який динамічно оновлюється залежно від результатів користувача.

Алгоритми класифікують курси за рівнем складності й тематикою, створюючи персоналізовані маршрути навчання [21]. Штучний інтелект враховує наступні змінні:

- Кількість виконаних вправ;
- Час, витрачений на кожне завдання;

- Частоту помилок;
- Темп навчання;

На основі цих даних формується навчальний маршрут, який автоматично розширюється або спрощується. Якщо користувач не справляється із завданнями, система пропонує короткі пояснення, додаткові практичні вправи або повертає до базового матеріалу. У разі успішного виконання складних тем користувачеві відкривається більш спеціалізовані курси або завдання.

3. Інтелектуальний моніторинг прогресу та прогнозування успішності

DataCamp активно використовує моделі прогнозування для оцінки ризику відтоку користувачів. Моделі машинного навчання аналізують, як часто студент входить в систему, скільки часу витрачає на курси, у яких моментах найчастіше зупиняється, і прогнозують ймовірність припинення навчання [21].

Якщо алгоритм виявить падіння активності, користувач отримує наступне:

- Автоматичне нагадування або рекомендацію повернутися до курсу.
- Персональне повідомлення з мотивацією, наприклад, “Ви близько до завершення курсу! Наступна вправа допоможе закріпити нову тему”.
- Пропозицію легшого або коротшого модуля для відновлення інтересу.

Також, моделі прогнозують ймовірність успішного завершення курсу й оцінюють темп прогресу, що дозволяє їм оптимізувати навчальний матеріал.

4. Адаптивне повторення та інтелектуальні підказки

Для підтримки довгострокового засвоєння знань DataCamp застосовує розподіленні повторення – це механізм, який прогнозує момент, коли користувач може забути матеріал [21]. Алгоритм штучного інтелекту автоматично пропонує короткі вправи на повторення саме тоді, коли рівень утримання знань починає знижуватись. У поєднанні з інтелектуальним механізмом зворотного зв'язку система аналізує відповіді користувача та пояснює помилки природною мовою, підказуючи, як їх виправити.

5. Інтелектуальний помічник для пояснення коду

У 2024 році DataCamp інтегрувала помічника на основі штучного інтелекту, побудованого на великій мовній моделі, що виконує функцію персонального репетитора з програмування [22]. Він може:

- Пояснювати логіку коду та помилки;
- Надавати короткі підказки щодо оптимізації;
- Генерувати приклади коду за запитом користувача;
- Допомагати у вирішенні практичних проєктів;

На відміну від звичайних чат-ботів, даний помічник розуміє і пам'ятає контекст користувача, його прогрес, теми, з якими виникали труднощі, і навіть стиль написання коду [22]. Це дозволяє формувати поради, максимально адаптовані до індивідуальної траєкторії студента.

Таким чином, DataCamp є прикладом платформи, яка повністю інтегрувала штучний інтелект у всі етапи навчального процесу, від оцінки рівня знань і побудови навчального маршруту до прогнозування результатів і прояснення коду. Поєднання адаптивного тестування, аналізу поведінки та інтелектуальної підтримки користувачів забезпечує гнучке, ефективне й мотивуюче середовище для навчання програмуванню.

Освітня платформа Khan Academy – це одна з найвідоміших некомерційних систем відкритої освіти, що поєднує алгоритмічну адаптивність і педагогічну аналітику [23]. У сфері програмування вона застосовує штучний інтелект для підтримки навчального процесу, формування персоналізованих рекомендацій та моніторингу рівня засвоєння знань.

Основна концепція персоналізації на Khan Academy базується на моделі адаптивного навчання, орієнтованій на повне опанування знань, яка використовує елементи машинного навчання для оцінювання рівня компетентності користувача та автоматичного підбору навчальних матеріалів.

1. Адаптивне навчання до повного засвоєння

   Механізм навчання до повного засвоєння є центральним елементом інтелектуальної персоналізації на платформі. Кожна тема, зокрема з програмування, поділена на модулі. Коли студент проходить вправи, система реєструє такі дані:

   - Кількість правильних відповідей;
   - Час виконання завдань;
   - Кількість повторів теми;
   - Частоту звернень до підказок;

   Модель машинного навчання аналізує ці параметри та оцінює рівень володіння темою за шкалою від 0% до 100%. Якщо рівень падає нижче певного порогу, зазвичай 70%, то система повертає студента до повторення матеріалу або пропонує спрощені вправи. У разі стабільного виконання завдань рівень теми підвищується, і користувач отримує доступ до складніших завдань. Такий підхід забезпечує індивідуальний темп навчання, що є ключовим принципом персоналізації освіти.

2. Інтелектуальний відстежувач прогресу

   Khan Academy використовує аналітичну систему відстеження прогресу, яка побудована на алгоритмах прогнозування успішності.

Система аналізує історію навчання користувача, визначає його темп засвоєння матеріалу та прогнозує, скільки часу потрібно для опанування подальших тем. На основі цих прогнозів формуються персональні рекомендації, які можуть містити:

- Пропозицію повернутися до попередньої теми для її закріплення.
- Рекомендацію нових завдань або вправ із подібної складності.
- Повідомлення про досягнення.

У курсах з програмування це допомагає підтримувати оптимальний баланс складності, щоб студент не відчував перенавантаження, але водночас рухався вперед із викликами, які відповідають його рівню.

3. Адаптація складності завдань і повторення тем

Модуль штучного інтелекту Khan Academy використовує адаптивне масштабування складності [30]. Якщо система фіксує, що студент швидко виконує завдання без помилок, наступні вправи автоматично ускладняються, додаються додаткові умови або більші фрагменти коду для аналізу. У разі системних помилок модель генерує пояснювальні кроки, які розбивають задачу на менші етапи.

Цей підхід відрізняється від звичайного лінійного навчання тим, що кожен користувач має власну глибину опрацювання теми, кількість повторень визначається не програмою, а самим алгоритмом штучного інтелекту.

4. Рекомендації наступних кроків навчання

На основі аналізу поведінкових даних, наприклад, частота сесій, середній бал, кількість помилок, алгоритм формує персональні рекомендації щодо подальших дій.

Такі рекомендації подаються у форматі інтерактивної панелі, що оновлюється в реальному часі. Це створює ефект наставника,

певної “розмови” з начальною системою, коли користувач отримує безперервний супровід і підтримку.

5. Аналітика ефективності навчання

Khan Academy також застосовує підхід прогнозної аналітики для оцінювання ефективності навчання. Система збирає великі обсяги даних про прогрес мільйонів користувачів і використовує їх для вдосконалення моделей адаптації. На основі аналізу цих даних створюються глобальні моделі, які прогнозуються наступне:

- Час, необхідний для досягнення рівня “опанував” у кожній темі.
- Середню кількість повторень, потрібну для засвоєння певного матеріалу.
- Ймовірність повернення користувача до платформи після перерви.

Отримані дані застосовуються для оновлення рекомендаційних моделей, що підвищує точність персоналізації майбутніх користувачів.

6. Інтеграція розмовних моделей

У 2023 році Khan Academy запустила експериментальний інструмент під назвою “Khanmigo” – це інтелектуальний асистент, який базується на мовній моделі GPT-4 [31]. Його мета – це підтримка навчального процесу через діалог, а не просто надання відповідей. У межах курсів з програмування Khanmigo може:

- Ставити уточнювальні запитання, допомагаючи користувачу самостійно знайти рішення;
- Пояснювати логіку коду;
- Пропонувати теми для повторення на основі минулих помилок;

Асистент не дає готових рішень, а допомагає студенту крок за кроком дійти до них самостійно, що узгоджується з концепцією про навчання через відкриття.

Саме так, Khan Academy демонструє приклад поєднання педагогічної аналітики та штучного інтелекту, де системи машинного навчання забезпечують адаптацію матеріалу, а мовні моделі підсилюють індивідуальну взаємодію з користувачем.

У результаті формується гнучке навчальне середовище, яке дозволяє кожному студенту просуватися у власному темпі, отримуючи рекомендації, підтримку й мотивацію у процесі навчання програмуванню.

Останніми роками великі мовні моделі стали новим інструментом для персоналізованого навчання, який доповнює або навіть частково замінює традиційні освітні платформи.

Такі системи як ChatGPT, Google Gemini та Claude поєднують у собі властивості віртуального викладача, консультанта й тренажера для самостійного навчання [32]. Їх ключові особливості – це інтерактивність та адаптивність у реальному часі, що забезпечує унікальним формат діалогу між студентом та штучним інтелектом.

1. ChatGPT:

Модель ChatGPT, яка побудована на архітектурі GPT-4/GPT-5, є одним із найпопулярніших інструментів для самостійного навчання програмуванню. Вона може виконувати функції викладача, наставника, репетитора й середовища для перевірки знань. Основні способи використання ChatGPT у навчанні програмуванню це [33]:

- Пояснення теоретичних концепцій. Користувач може поставити запитання і отримати чітке пояснення з прикладами коду.
- Покрокове розв’язування завдань. ChatGPT може розбити складну задачу на етапи, пояснюючи логіку рішення. Це

особливо корисно для новачків, які лише вчяться структурувати мислення.

- Аналіз та виправлення коду. Користувач вставляє свій код, а модель знаходить помилки, пояснює причину і пропонує варіанти виправлення.
- Генерація навчальних завдань. ChatGPT здатний створювати вправи з програмування різного рівня складності.
- Симуляція технічних інтерв'ю. Модель може виступати як інтерв'юер, ставлячи запитання з алгоритмів і структур даних, що використовується для підготовки до співбесід.

Важливою перевагою ChatGPT є адаптивність до стилю користувача, модель запам'ятовує контекст попередніх відповідей і підлаштовується від рівень складності запитів.

Таким чином формується індивідуальна траєкторія навчання, яка пояснює практику, теорію й інтерактивний зворотній зв'язок.

2. Google Gemini:

Модель Google Gemini є мультимодальною системою, що поєднує текстові, кодові та візуальні дані. У контексті навчання програмуванню Gemini використовується як інтерактивний навчальний консультант, який може не лише пояснювати, але й демонструвати виконання коду вбудованому середовищі [34].

Приклади застосування:

- Інтерактивне пояснення коду. Користувач може надіслати фрагмент JavaScript або Python, а Gemini пояснює логіку крок за кроком, візуалізуючи змінні та потоки виконання.
- Оптимізація програм. Модель аналізує ефективність коду, пропонуючи більш оптимальні рішення з коментарями.

- Підбір навчальних ресурсів. Gemini може створювати добірку посилань на статті, відео документацію, релевантних до поточного етапу навчання користувача.
- Покроковий супровід у навчанні. Користувач може поставити мету, наприклад, створити телеграм-бота на Python, і модель сформує індивідуальний план із навчальними кроками, прикладами коду й перевірками проміжних результатів.

Перевагою Gemini є його інтеграція з інфраструктурою Google, що дає змогу використовувати модель як єдине середовище для навчання й практики.

3. Claude:

Модель Claude від компанії Anthropic є прикладом “етичного штучного інтелекту”, який орієнтований на безпечну взаємодію з користувачами. У навчанні програмуванню Claude використовується як розмовний інструктор, який поєднує точність кодування з гуманним стилем пояснень [32].

Його особливості це:

- Пояснення концепцій природною мовою. Модель навидить аналогії, метафори, приклади з реального життя.
- Гнучка адаптація до рівня користувача. Claude може спрощувати технічні описи для початківців або деталізувати складні концепції, такі як, принципи ООП, рекурсії чи паралелізму.
- Аналіз великих фрагментів коду. Claude підтримує контекст до 100000 токенів, що дозволяє йому працювати з великими програмами або навіть з усім навчальним проєктом одночасно.
- Етична підтримка навчання. Модель уникає надання готових рішень, натомість пропонує підказки, заохочуючи користувача мислити самостійно.

Завдяки цим особливостям Claude розглядають як інструмент для розвитку критичного мислення під час навчання програмуванню, особливо у середовищах, де важливо не просто отримати результат, а зрозуміти процес.

У результаті великі мовні моделі стали новим форматом самоосвіти, який поєднує доступність, адаптивність і практичну орієнтацію, ключові чинники ефективного навчання програмуванню у 21-му столітті [14].

Проведений огляд показав, що сучасні освітні платформи та інтелектуальні інструменти активно інтегрують штучний інтелект для створення адаптивних систем навчання програмуванню. Штучний інтелект не лише автоматизує добір контенту, а й забезпечує інтерактивну підтримку, зворотній зв’язок та побудову індивідуальних траєкторій розвитку.

Усі розглянуті приклади демонструють тенденцію до інтелектуалізації освітнього простору, де штучний інтелект поступово стає не допоміжним інструментом, а центральним елементом адаптивного навчання.

Рекомендаційні алгоритми, мовні моделі та аналітичні модулі спільно формують нову парадигму освіти, від масових онлайн-курсів до індивідуальних інтелектуальних траєкторій. Ці підходи створюють основу для подальшого розвитку рекомендаційних систем вибору мови програмування та побудови навчального плану.

1.5 Порівняння переваг та недоліків сучасних систем

Інтеграція штучного інтелекту у навчальні платформи з програмування забезпечила суттєвий прогрес у сфері персоналізованого навчання. Однак підходи різних систем відрізняється за архітектурою, ступенем автоматизації, глибиною аналітики й рівнем взаємодії з користувачем [14].

Проведемо порівняння основних характеристик розглянутих рішень:

1. Coursera:

- Основні технології [16]:
    i. Глибинне навчання.
    ii. Прогностичні моделі.
    iii. Велика мовна модель, Coursera Coach.
- Переваги [18]:
    i. Високий рівень персоналізації навчальних траєкторій.
    ii. Аналітика поведінки користувача.
    iii. Адаптація складності.
    iv. Асистент на основі штучного інтелекту для пояснень.
    v. Сертифікаційні програми з кар’єрною орієнтацією.
- Недоліки [18]:
    i. Частково закриті алгоритми, обмежена прозорість.
    ii. Не завжди точна адаптація при зміні темпу.
    iii. Залежність від англомовного матеріалу.

2. Codecademy [19]:
    - Основні технології [20]:
        i. Машинне навчання.
        ii. Обробка природної мови.
        iii. GPT-4 у асистенті для написання коду.
    - Переваги [19]:
        i. Інтерактивний зворотній зв’язок.
        ii. Аналіз коду в реальному часі.
        iii. Адаптивне відслідковування прогресу.
        iv. Механізми повторення.
        v. Практична орієнтація.
    - Недоліки [29]:
        i. Обмежена теоретична база.
        ii. Менш ефективна робота офлайн.
        iii. Частина функцій доступна лише у платній підписці.

3. DataCamp:
    - Основі технології [21]:
        i. Теорія відгуку на завдання.
        ii. Баєсова статистика.
        iii. Прогнозування на основі машинного навчання.
        iv. Ментор на основі великої мовної моделі.
    - Переваги [22]:
        i. Найвищий рівень автоматизації.
        ii. Точна оцінка рівня знань, технологія Signal.
        iii. Персональні напрями навчання.
        iv. Аналітика успішності.
        v. Система нагадування.
        vi. Тренер на основі штучного інтелекту.
    - Недоліки [30]:
        i. Зосередженість переважно на Python/R.
        ii. Обмежений доступ до контенту у безкоштовній версії.
        iii. Потреба у стабільному інтернет-з’єднані.
4. Khan Academy [23]:
    - Основні технології [23]:
        i. Машинне навчання.
        ii. Адаптивна модель опанування матеріалу.
        iii. GPT-4 у інтелектуальному асистенті Khanmigo.
    - Переваги [31]:
        i. Безкоштовність і відкритість.
        ii. Стабільна адаптація складності.
        iii. Система опанування теми.
        iv. Педагогічна орієнтація.
        v. Інтеграція з розмовним штучним інтелектом.
    - Недоліки [30]:

i. Обмежений контент з програмування.
ii. Менш глибока практична складова.
iii. Не завжди точне розпізнавання складності завдань.

5. ChatGPT [35]:
   - Основні технології [33]:
     i. Велика мовна модель GPT-4/GPT-5.
     ii. Міркування на основі кількох прикладів.
     iii. Контекстне навчання.
   - Переваги:
     i. Повна гнучкість навчання.
     ii. Миттєвий зворотній зв’язок.
     iii. Можливість самостійного планування навчання.
     iv. Генерація завдань і пояснень.
     v. Підтримка кількох мов програмування.
   - Недоліки [32,34]:
     i. Відсутність контролю навчального плану.
     ii. Ризик неточних відповідей.
     iii. Потреба у критичному мисленні користувача.
6. Google Gemini [34]:
   - Основні технології:
     i. Мультимодальна велика мовна модель.
     ii. Інтеграція з сервісами Google.
   - Переваги:
     i. Візуалізація процесів.
     ii. Підтримка інтерактивного виконання коду.
     iii. Зв’язок із зовнішніми ресурсами.
     iv. Автоматичне створення навчальних кроків.
   - Недоліки [32]:
     i. Обмежена доступність у деяких регіонах.

ii. Потреба у вході в екосистему Google.
iii. Іноді надмірна кількість рекомендацій.

7. Anthropic Claude [32]:
   - Основні технології:
     i. Етичний штучний інтелект.
     ii. Пам’ять контексту до 100000 токенів.
     iii. Обробка природної мови.
   - Переваги:
     i. Етичність взаємодії.
     ii. Пояснення природною мовою.
     iii. Робота з великими обсягами коду.
     iv. Сприяє розвитку самостійного мислення.
     v. Підходить для менторського навчання.
   - Недоліки [34,36]:
     i. Менш практично орієнтований, бо немає вбудованих компіляторів коду.
     ii. Обмежений у генерації складних кодових структур.

Як показує порівняльний аналіз, усі розглянуті системи мають спільну мету – це персоналізувати процес навчання, проте реалізують це різними шляхами.

Coursera і DataCamp використовують високий рівень автоматизованої аналітики, моделі прогнозують успішність і формують навчальні маршрути з урахуванням прогресу користувача.

Codecademy акцентує увагу на інтерактивному практичному навчанні та миттєвому зворотному зв’язку, що підходить для студентів, які навчаються через експерименти з кодом.

Khan Academy поєднує педагогічну адаптивність і етичні принципи штучного інтелекту, роблячи процес доступним для широкої аудиторії.

Великі мовні моделі, ChatGPT, Gemini та Claude, створюють нову форму самостійного навчання, у якій студент сам керує навчальним процесом, а система підлаштовується під його стиль мислення й запити.

Водночас спостерігаються певні обмеження. Моделі великих платформ, а саме: Coursera та DataCamp, залежать від великої кількості попередніх даних і не завжди ефективні для початківців без досвіду. Інтерактивні системи від Codecademy та Khan Academy, мають нижчій рівень охоплення матеріалом, тоді як великі мовні моделі, ChatGPT, Gemini та Claude, потребують критичного мислення користувача для перевірки достовірності згенерованих відповідей.

Загальні висновки порівняння показується, що:

1. Найбільш збалансованою з точки зору автоматизації, персоналізації та точності є платформа DataCamp, де штучний інтелект інтегрований у всі етапи навчального процесу.
2. Coursera забезпечує найкращу адаптацію навчальних напрямків до цілей користувача, що робить її ефективною для комплексного навчання програмуванню.
3. Codecademy є лідером у практичному підході до формування навичок і миттєвому зворотному зв’язку.
4. Khan Academy відзначається найвищою доступністю та педагогічною якістю, але поступається у глибині спеціалізованого матеріалу.
5. ChatGPT, Gemini та Claude – це новий етап індивідуалізації навчання, який робить освіту максимально гнучкою, але потребує самостійності та критичної оцінки інформації з боку користувача.

Отже, сучасні системи персоналізованого навчання з використанням штучного інтелекту мають різні пріоритети, аналітична точність, практична орієнтація або інтелектуальна свобода користувача.

1.6 Методи та технології тестування знань у сучасних освітніх системах

Тестування знань є важливою складовою сучасних освітніх процесів, оскільки воно забезпечує об’єктивне оцінювання рівня підготовки користувача, виявлення прогалин і формування індивідуальних освітніх траєкторій. Еволюція технологій привела до появи нових форматів контролю, зокрема комп’ютеризованого, інтелектуального й адаптивного тестування, що значно підвищило точність, гнучкість і швидкість оцінювання. Сучасні системи не лише перевіряють знання, але й здійснюють їх аналітичну інтерпретацію, підтримують прийняття рішень та рекомендації щодо подальшого навчання.

У комп’ютерних навчальних платформах використовується широкий спектр форматів тестів, кожен з яких передбачає специфічний метод перевірки та вимоги до аналізу результатів. А саме: закриті тестові завдання, відкриті тестові завдання, розгорнуті відповіді та есе.

1. Закриті тестові завдання

До них належать:

- вибір однієї правильної відповіді;
- множинний вибір;
- встановлення відповідностей;
- побудова правильної послідовності;
- шкальні оцінки;

Такі завдання легко автоматизуються, що дозволяє забезпечити високу швидкість і точність оцінювання. У твоєму PDF-файлі вони реалізовані у модулі QuestionManager, який підтримує одиночний та множинний вибір, шкальні відповіді, перевірку коректності введення.

2. Відкриті тестові завдання

Відкриті питання передбачають введення користувачем короткої або розгорнутої відповіді [38, 37]. Методи оцінки змістової повноти тексту без суворого збігу слів: пошук ключових концептів, аналіз синонімів та контекстних

зв’язків, визначення часткової відповідності, лексичне нормування та лематизація.

Ці підходи важливі для інтелектуальних систем, де відповідь не обмежується варіантами. Вони застосовуються у автоматизованих системах контролю знань і забезпечують високу об’єктивність.

3. Розгорнуті відповіді та есе

Оцінювання таких відповідей потребує:

- семантичного аналізу;
- виявлення ключових структурних елементів;
- використання експертних правил або моделей ШІ;
- перевірки логічності, повноти та достовірності змісту;

Цей формат часто застосовується у дисциплінах з аналітичними та творчими завданнями, але може бути автоматизований лише частково, тому потребує поєднання автоматичної та експертної оцінки.

Експертні системи є важливим інструментом підвищення об’єктивності та точності контролю знань [37]. Експертні системи відтворюють логіку педагогів, використовуючи [40]:

- базу знань (правила оцінювання, типові помилки);
- механізм логічного виведення;
- діагностику рівня знань користувача;
- адаптивне коригування складності завдань;

Експертні системи дозволяють виконувати:

- автоматичне визначення рівня підготовки;
- аналіз стилю помилок;
- побудову рекомендацій щодо подальших тем;
- виявлення нетипових відповідей;
- аналіз статистики успішності;

Комп’ютеризовані системи тестування дають можливість:

- автоматизувати перевірку відповідей;

- збирати детальну статистику;
- відстежувати процес проходження тесту;
- адаптувати інтерфейс під різні пристрої;
- перевіряти стабільність роботи модулів у реальних умовах;

Адаптивне тестування – один із найсучасніших і найефективніших підходів. Воно полягає у тому, що кожне наступне питання підбирається на основі попередньої відповіді. Це дозволяє швидко визначити рівень користувача.

Переваги адаптивного тестування:

- зменшує загальний обсяг тесту;
- підвищує точність оцінювання;
- мінімізує стрес користувача, уникаючи надмірно складних питань;
- забезпечує персоналізований підхід;

Методи, що використовуються:

- Item Response Theory (IRT);
- динамічна зміна складності;
- аналіз прогалин у знаннях;
- формування індивідуальних траєкторій;

Адаптивне тестування також застосовується у створенні персоналізованих планів навчання.

Завдяки розвитку штучного інтелекту з’явилися нові підходи до тестування: ML у тестуванні знань, семантичний аналіз відповідей, діагностика прогалин у знаннях.

Методи машинного навчання дозволяють:

- прогнозувати рівень успішності;
- виявляти приховані закономірності у відповідях;
- будувати навчальні траєкторії;
- персоналізувати рекомендації;
- визначати ймовірність правильної відповіді;

Семантичний аналіз відповідей застосовується для відкритих питань та есе. Використовувані моделі:

- word embeddings;
- трансформери;
- TF-IDF із семантичними модифікаціями;

Інтелектуальні системи діагностика прогалин у знаннях аналізують:

- типові помилки;
- причинно-наслідкові зв’язки між пропущеними темами;
- логіку мислення користувач;

Такі системи дозволяють автоматизувати створення індивідуального плану навчання.

Статистичний аналіз тестових завдань. Методи статистичного аналізу тестів дозволяють визначати складність завдань, оцінювати дискримінативність, виявляти аномальні відповіді, перевіряти коректність шкал оцінювання [40]. Також аналізуються наступні характеристики: надійність тесту (α-Кронбаха), внутрішня узгодженість, валідність і релятивність. Статистика є основою для покращення тестів та формування якісних банків завдань.

У сучасних освітніх платформах тестування перестало бути лише механізмом контролю. Воно перетворилося на інструмент персоналізації навчання, який визначає індивідуальний рівень користувача, його сильні й слабкі сторони, темп засвоєння матеріалу та потенційні освітні цілі. На основі результатів тестування платформи формують рекомендації щодо тем і напрямів навчання, адаптивні шляхи опанування матеріалу, оцінку готовності до складніших модулів, виявлення прогалин у фундаментальних знаннях, динамічні навчальні плани [39]. Таким чином, тестування — це не фінальний етап навчання, а початкова точка для створення персоналізованої траєкторії.

Діагностичні тести спрямовані на визначення рівня базових знань, що необхідні для подальшого навчання. Вони зазвичай охоплюють такі блоки:

1. Тест на фундаментальні компетентності

Оцінюють загальну ерудицію, логіку, математичні навички або базові технічні знання.

2. Тест на предметні компетентності

Перевіряють базові навички в конкретній галузі: програмуванні, web-розробці, алгоритмах тощо.

3. Психометричні та поведінкові тести

Вимірюють стиль мислення, упевненість у відповідях, вибір стратегії поведінки під час навчання.

4. Мотиваційні тести

Допомагають визначити інтереси, цілі та пріоритети, що також впливає на формування рекомендацій.

Такі тести є фундаментом для подальшого аналізу, де відповіді на запитання визначають не лише рівень користувача, а й його мотивацію та напрямки інтересів, що далі впливає на рекомендаційний алгоритм.

У ході огляду сучасних підходів до тестування знань встановлено, що системи контролю навчальних результатів еволюціонували від традиційних тестових форм до комплексних комп'ютеризованих, експертних та адаптивних рішень. Різноманіття форматів завдань — від закритих тестів до відкритих і семантичних відповідей — потребує застосування специфічних методів оцінювання, які забезпечують об'єктивність, точність і врахування індивідуальних особливостей користувача. Експертні системи та алгоритмічні моделі аналізу відповідей дають змогу автоматизувати діагностику рівня знань і виявлення типових помилок, а адаптивне тестування формує персоналізовану траєкторію навчання відповідно до рівня підготовки. Таким чином, сучасні технології тестування знань не лише визначають рівень засвоєння матеріалу, а й інтегруються у комплексні освітні платформи, забезпечуючи інтелектуальний супровід навчального процесу та удосконалення механізмів рекомендації навчального контенту.

Висновки для розділу 1

У розділі було проведено комплексний аналітичний огляд сучасних рекомендаційних систем та особливостей їх прогресу та покращення від впливом розвитку штучного інтелекту.

Розглянуто структуру таких систем, основні підходи до побудови рекомендацій, а також визначено ключові компоненти, що забезпечують їх ефективність.

Проведений історичний аналіз розвитку штучного інтелекту показав, що через появу методів машинного та глибинного навчання спричинився перехід від статичних до адаптивних рекомендаційних систем, здатних самостійно навчатися на основі поведінки користувачів.

Інтеграція моделей машинного навчання, обробки природної мови, глибинних нейронних мереж і підсиленого навчання дала змогу створити новий тип освітніх систем, які не лише пропонують релевантні курси, а й активно формують індивідуальні навчальні траєкторії.

Проведено порівняльний аналіз сучасних платформ та середовищ для навчання. Який показав, що кожна система має власну стратегію до персоналізації. Вони всі мають спільну мету – максимально індивідуалізувати процес навчання за допомогою аналізу даних, поведінки та контексту користувача.

Використання штучного інтелекту у рекомендаційних системах підвищує точність підбору матеріалів для навчання, прискорює засвоєння знань, знижує когнітивне навантаження та підвищує мотивацію студентів. Водночас залишаються відкритими питання забезпечення прозорості алгоритмів, етичності обробки даних і перевірки достовірності згенерованих відповідей мовними моделями.

Було проведено аналіз сучасних методів тестування знань, які відіграють ключову роль у формуванні персоналізованих навчальних траєкторій. Розглянуто основні типи тестових завдань, підходи до їх автоматизованого

оцінювання, застосування експертних систем та механізми адаптивного тестування, що дозволяють точніше визначати рівень підготовки користувача. Показано, що сучасні системи контролю знань не лише вимірюють навчальні результати, але й слугують джерелом даних для рекомендаційних алгоритмів, забезпечуючи об’єктивність, гнучкість і індивідуалізацію освітнього процесу.

Отже, результати аналізу показують, що поєднання механізмів рекомендаційних систем і штучного інтелекту є ефективним напрямом розвитку цифрової освіти.

РОЗДІЛ 2. Проектування та реалізація рекомендаційної системи

2.1 Архітектура та структура програмного продукту

Архітектура програмного продукту визначає загальну логічну організацію системи, взаємозв'язки між її компонентами, способи зберігання та обробки даних, а також підхід до реалізації основних функціональних можливостей. Її побудова є ключовим етапом проєктування, оскільки саме від архітектурних рішень залежить подальша масштабованість, надійність і підтримуваність програмного комплексу.

Метою створення архітектури є розроблення гнучкої та модульної структури системи, яка забезпечить виконання всіх функціональних вимог до рекомендаційного застосунку, а також можливість його подальшого розширення. Рекомендаційна система має реалізовувати механізм покрокового опитування користувача, аналізу отриманих відповідей, формування рекомендацій щодо вибору мови програмування та автоматичної побудови індивідуального плану навчання. Крім того, архітектура повинна забезпечити стабільну роботу програми на різних платформах — Android, Windows та WebGL, що відповідає вимогам до кросплатформних освітніх рішень.

Розробка рекомендаційної системи для вибору мови програмування та формування індивідуального плану навчання вимагає поєднання кількох функцій, а саме: збирання даних про користувача (відповіді на запитання, рівень підготовки), оброблення цих даних за визначеними правилами або моделлю, відображення результату у зручному інтерфейсі, а також можливості подальшого розширення системи (додавання нових мов, навчальних модулів, джерел даних). Тому вибір мови програмування в цьому випадку визначає не тільки синтаксис розробки, але й архітектурні можливості.

Щоб вибір мови був обґрунтованим, доцільно попередньо визначити вимоги до майбутньої системи, ключовими критеріями є:

1. Кросплатформність

Система має бути доступною для користувачів, які навчаються з мобільних пристроїв або ПК. Бажано мати можливість зібрати продукт під Android/Windows без істотної переробки коду [44].

2. Підтримка інтерактивного інтерфейсу

Оскільки мова йде про освітній застосунок із покроковими запитаннями, підказками та виведенням рекомендованого навчального маршруту, платформа повинна мати зручні засоби побудови UI та анімацій [44, 45].

3. Модульність і розширюваність

Рекомендаційна система у сфері освіти майже завжди розвивається: додаються нові профілі користувачів, нові мови програмування, додаткові правила. Тому потрібна мова, що добре підтримує об’єктно-орієнтоване програмування і чітку структуризацію коду [42].

4. Інтеграція з базами даних або хмарними сервісами

Дані про користувачів, їх відповіді та сформовані плани навчання мають зберігатися у зовнішньому сховищі (SQL, Firebase чи інший бекенд). Технологія має мати готові бібліотеки для цього.

5. Можливість реалізації рекомендаційної логіки

Навіть якщо на першому етапі логіка на основі правил (як експертна система з if-else/деревом рішень), у перспективі її можна доповнювати елементами машинного навчання. Тож бажано, щоб обрана мова дозволяла це без повної зміни стеку.

До основних вимог до архітектури системи належать такі:

По-перше, система має бути модульною, тобто складатися з окремих компонентів, кожен із яких виконує чітко визначену функцію [42]. Це дозволяє спрощувати налагодження, тестування та повторне використання окремих модулів у майбутньому.

По-друге, архітектура повинна підтримувати розширюваність, тобто можливість додавання нових мов програмування, запитань або логічних правил без необхідності суттєвої зміни базового коду [41, 42].

По-третє, важливою вимогою є тестованість, що передбачає можливість перевірки логіки рекомендацій та модулів системи незалежно від графічного інтерфейсу.

По-четверте, має бути забезпечено збереження та відновлення даних, оскільки користувач повинен мати змогу продовжити роботу після переривання сесії. Для цього передбачено інтеграцію з хмарним сервісом Firebase, а також локальне кешування результатів [48, 49].

Особливу увагу в архітектурі приділено надійності та продуктивності. Програма повинна швидко реагувати на дії користувача, забезпечуючи плавність анімацій і мінімальні затримки при переході між екранами. Важливо, щоб основна логіка системи виконувалася незалежно від стабільності інтернет-з’єднання: при відсутності мережі відповіді тимчасово зберігаються локально і синхронізуються з хмарним сховищем при відновленні зв’язку.

Крім функціональних, архітектура враховує й нефункціональні вимоги, серед яких: безпека зберігання персональних даних, підтримка багатомовності, адаптивність інтерфейсу, продуктивність, а також сумісність із різними операційними системами.

Окремо передбачено реалізацію механізмів аналітики, які фіксують активність користувача, результати проходження тестів та статистику прийнятих рекомендацій. Ці дані надалі можуть бути використані для удосконалення алгоритмів рекомендацій або оцінки ефективності системи.

Загалом архітектура рекомендаційного застосунку повинна відповідати таким критеріям:

- чітке розділення логічних рівнів (користувацький інтерфейс, бізнес-логіка, зберігання даних);

- можливість легкого оновлення контенту без необхідності перевипуску додатка;
- забезпечення офлайн-роботи та синхронізації після відновлення підключення;
- інтеграція з аналітичними сервісами та системою локалізації;
- підтримка високої продуктивності й стабільності під час використання;

Таким чином, архітектура розроблюваної рекомендаційної системи має забезпечити баланс між простотою реалізації, гнучкістю в розширенні та високим рівнем користувацького досвіду. Вона виступає основою для подальшого розроблення структури програмного продукту, логіки взаємодії модулів і способів обміну даними.

Під час розроблення програмного продукту було обрано шаровий архітектурний стиль, який передбачає розподіл системи на кілька логічно незалежних рівнів: презентаційний (інтерфейс користувача), логічний (оброблення даних і рекомендацій) та рівень зберігання даних. Такий підхід забезпечує високу структурованість коду, спрощує налагодження, підтримку та подальшу модифікацію окремих компонентів без порушення цілісності всієї системи.

Основною ідеєю шарової архітектури є чітке розмежування відповідальності між частинами системи.

На презентаційному рівні реалізовано засоби взаємодії з користувачем — інтерфейс, елементи керування, відображення результатів опитування та рекомендацій.

Логічний рівень відповідає за обробку вхідних даних, застосування правил експертної системи, визначення рівня знань користувача та формування результатів у вигляді структурованих рекомендацій.

Рівень даних забезпечує збереження профілю користувача, історії відповідей та сформованих рекомендацій у локальному сховищі або хмарній базі даних Firebase.

Між цими рівнями встановлено мінімальні залежності, що дозволяє змінювати реалізацію одного з них без потреби в модифікації інших. Наприклад, алгоритм рекомендацій можна замінити на модель машинного навчання без втручання у структуру користувацького інтерфейсу.

Архітектура програмного продукту ґрунтується на моделі MVC (Model–View–Controller), яка є однією з найпоширеніших парадигм побудови програмних систем [41]. Її застосування дозволяє розділити логіку роботи додатка на три взаємопов’язані, але незалежні складові: модель, представлення (вигляд) та контролер. Такий підхід спрощує розробку, налагодження й подальше розширення системи, забезпечуючи одночасно високу керованість процесами обробки даних та відображення інформації користувачу.

У запропонованій системі модель (Model) описує внутрішню структуру даних і бізнес-логіку програми. Вона містить класи, що представляють користувача, запитання, результати опитування, параметри мов програмування та алгоритми формування рекомендацій. Саме на рівні моделі реалізовано механізми обчислення рівня підготовки користувача, вибору релевантної мови програмування й побудови індивідуального плану навчання. Модель не взаємодіє безпосередньо з інтерфейсом користувача, а лише оперує даними, передаючи їх контролеру.

Представлення (View) відповідає за візуальне відображення інформації та забезпечує інтерактивну взаємодію користувача із системою. Цей рівень реалізується засобами середовища Unity, яке надає широкі можливості для створення адаптивних графічних інтерфейсів, анімацій, підказок і діалогових вікон [44, 45]. Усі елементи інтерфейсу побудовано на основі компонентів Unity UI (Canvas, TextMeshPro, Button, ScrollView тощо), що забезпечує наочність, зручність і гнучкість у відображенні даних. Представлення не містить бізнес-

логіки — воно лише реагує на події користувача, викликаючи відповідні методи контролера.

Контролер (Controller) виконує функцію посередника між моделлю та представленням. Він обробляє дії користувача, отримує відповідні дані з моделі, виконує логічні операції (наприклад, розрахунок результатів опитування або запуск алгоритму рекомендацій) і передає оновлені дані до рівня представлення для візуалізації. Контролер відповідає також за навігацію між екранами, послідовність опитування, збереження результатів та виклик службових компонентів, зокрема модулів збереження даних, аналітики й локалізації.

Використання архітектури MVC забезпечує низку переваг:

- модульність — зміни в інтерфейсі не потребують коригування алгоритмів обробки даних;
- розширюваність — можна легко додавати нові типи запитань або алгоритми рекомендацій без зміни логіки відображення;
- тестованість — логіку моделі та контролера можна перевіряти окремо, без запуску візуальної частини Unity;
- підтримуваність — чітке розділення коду зменшує ризик помилок і полегшує командну розробку;

У межах цієї архітектури реалізовано також низку принципів об’єктно-орієнтованого проєктування та SOLID, які гарантують стабільність і гнучкість системи. Для мінімізації зв’язків між компонентами застосовано принцип інверсії залежностей (Dependency Injection), реалізований через бібліотеку Zenject. Це дозволяє керувати взаємними залежностями між класами, забезпечуючи можливість легкої заміни або оновлення окремих модулів (наприклад, алгоритму рекомендацій чи сховища даних).

Окрім основної моделі MVC, у системі використано допоміжні шаблони проєктування, такі як:

- Repository – для відокремлення бізнес-логіки від реалізації доступу до даних;

- Strategy – для гнучкого вибору алгоритму формування рекомендацій;
- Observer – для відстеження змін стану системи (оновлення інтерфейсу після отримання результатів);
- State – для організації переходів між етапами опитування.

Застосування архітектури MVC у поєднанні з принципами інверсії залежностей та шаблонами проєктування забезпечує побудову гнучкої, стабільної й масштабованої системи. Це дозволяє легко інтегрувати нові функціональні можливості, підтримувати високу продуктивність і відповідати сучасним вимогам до розроблення освітніх рекомендаційних систем.

Структура програмного продукту визначає логічну організацію всіх компонентів системи та їхню взаємодію в межах єдиного середовища. Вона відображає, як окремі елементи — інтерфейс користувача, модулі логіки, сховище даних і допоміжні сервіси — поєднуються в цілісну функціональну систему.

Розроблена рекомендаційна система побудована як модульний програмний комплекс, який складається з кількох взаємопов’язаних частин. Кожен модуль виконує певну роль і має чітко визначені межі відповідальності. Це забезпечує гнучкість у розвитку системи, полегшує супровід і дає можливість незалежно оновлювати окремі її компоненти.

До основних складових структури належать такі підсистеми:

1. Користувацький інтерфейс (User Interface Layer).

   Забезпечує взаємодію користувача з системою через візуальні елементи середовища Unity. Складається з набору сцен та екранів: головного меню, тестування користувача, перегляду результатів і навчального плану. Для відображення даних використано компоненти Canvas, TextMeshPro, ScrollView та інші елементи UI. Інтерфейс має адаптивну структуру, підтримує багатомовність і містить анімаційні переходи, які підвищують зручність користування.

2. Логічний модуль керування (Application Logic Layer).

   Координує роботу між інтерфейсом і внутрішніми компонентами системи. Саме тут відбувається оброблення подій користувача, управління етапами опитування, виклик алгоритмів рекомендацій і формування узагальнених результатів. Модуль також відповідає за керування станами застосунку, перевірку правильності введених даних і ініціацію збереження результатів у базі даних.

3. Модуль бізнес-логіки (Domain Layer).

   Реалізує основну інтелектуальну частину системи — моделі даних, алгоритми визначення рівня знань користувача, правила оцінювання, структуру мов програмування та логіку вибору найбільш релевантних варіантів. Саме тут розміщено класи, які описують користувача, питання, відповіді, навчальні напрями й алгоритм рекомендацій. Цей рівень є незалежним від інтерфейсу та середовища виконання, що дає змогу проводити його автономне тестування.

4. Модуль роботи з даними (Data Layer).

   Відповідає за збереження, читання та синхронізацію інформації. Для цього використовується хмарний сервіс Firebase, який забезпечує зберігання профілів користувачів, результатів опитувань і аналітичних даних. Крім того, передбачено локальне кешування результатів, що дозволяє користувачеві працювати офлайн із подальшою синхронізацією після відновлення з’єднання.

5. Модуль аналітики та оброблення статистики.

   Збирає дані про активність користувачів, проходження тестів, частоту вибору певних мов програмування та середній рівень успішності. Ці дані використовуються для подальшого вдосконалення системи, аналізу ефективності рекомендацій і дослідження навчальної поведінки користувачів. Для реалізації використано інструменти Firebase Analytics.

6. Модуль локалізації та контенту.

   Відповідає за відображення інтерфейсу й контенту кількома мовами, а також за оновлення запитань і навчальних матеріалів без перевипуску програми. Задіяно механізм Unity Localization і систему Addressables, що дозволяє динамічно завантажувати та оновлювати ресурси з віддаленого сховища.

Усі зазначені модулі взаємодіють між собою через визначені інтерфейси, що гарантує низький рівень зв’язності й високу узгодженість всередині кожного компонента. Така структура дозволяє змінювати або доповнювати окремі частини програми без ризику порушення роботи всієї системи.

Загалом побудована структура системи є гнучкою, багаторівневою та масштабованою. Вона забезпечує чітке розділення функцій, узгоджену взаємодію модулів і можливість подальшого розвитку програмного продукту — зокрема, впровадження нових алгоритмів аналізу, додаткових мов програмування або удосконалених механізмів навчальної персоналізації.

Таким чином, структурна модель системи виступає базовим каркасом для реалізації наступних етапів розробки — створення бази даних, формування логіки опитування, генерації навчального плану та впровадження рекомендаційного механізму.

2.2 Проектування бази даних і модулів програми

Ефективне функціонування рекомендаційної системи значною мірою залежить від правильного проєктування її інформаційної структури. База даних є центральним елементом програмного продукту, який забезпечує збереження, обробку та доступ до користувацьких і навчальних даних. Під час розроблення системи було застосовано реляційно-об’єктний підхід до проєктування структури даних з урахуванням особливостей інтеграції з хмарною платформою Firebase.

База даних системи складається з кількох логічно взаємопов’язаних сутностей, які відображають основні об’єкти предметної області [49]. Її

структура спроєктована так, щоб забезпечити мінімальну надмірність даних, цілісність зв’язків і простоту виконання аналітичних запитів.

Ключовими сутностями є:

- User (Користувач) – містить інформацію про профіль користувача, його унікальний ідентифікатор, ім’я (або псевдонім), вік, рівень початкових знань, поточний прогрес, мову інтерфейсу та останню дату активності.
- Question (Запитання) – зберігає текст запитання, тип відповіді (один варіант, множинний вибір, шкала), варіанти відповідей і вагові коефіцієнти, що впливають на формування підсумкового результату.
- Answer (Відповідь користувача) – фіксує вибрані користувачем варіанти відповіді, час проходження та відповідність критеріям оцінки.
- ProgrammingLanguage (Мова програмування) – містить опис кожної мови, її основні характеристики (парадигма, складність, сфера застосування, популярність), а також набір критеріїв, за якими відбувається порівняння з результатами тесту.
- Recommendation (Рекомендація) – зберігає результати аналізу: рекомендовану мову програмування, пояснення вибору, відсоток відповідності та згенерований навчальний план.
- LearningPlan (План навчання) – містить набір тем, модулів і завдань, сформованих для користувача залежно від обраної мови програмування.
- Analytics (Аналітика) – накопичує узагальнену статистику: кількість пройдених опитувань, частоту вибору мов, середній рівень користувачів і завершуваність сесій.

Взаємозв’язки між основними сутностями відображено на рисунку 2.1.

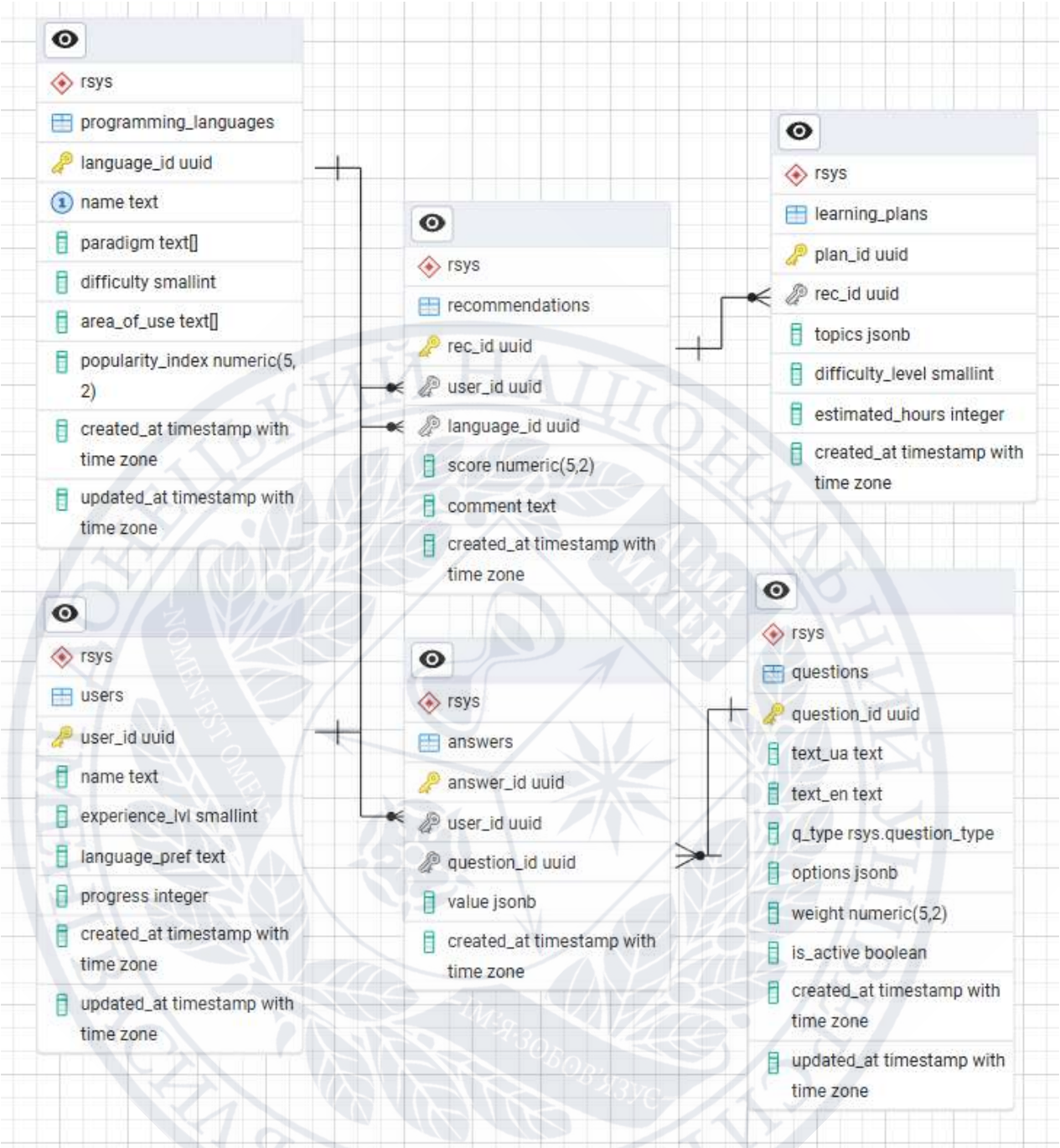

Рисунок 2.1 – Схема зв’язків між основними сутностями бази даних рекомендаційної системи

Під час вибору технологічного середовища для створення рекомендаційної системи було розглянуто кілька популярних підходів, що відповідають вимогам, зазначеним вище. Основними варіантами стали Python, JavaScript (у поєднанні з веб-фреймворками) та C# з використанням Unity таблиця 2.1 та таблиця 2.2.

Таблиця 2.1 Порівняльний аналіз можливих мов і середовищ розробки

Критерій	Python	JavaScript/Web	C#/Unity
Призначення	Аналіз даних, машинне навчання, серверна логіка	Веб-інтерфейси, інтерактивні сторінки, SPA-додатки	Кросплатформні інтерактивні застосунки, ігри, симуляції
Інтерфейс користувача	Складно реалізувати нативний UI; потребує додаткових бібліотек	Гнучкий, але потребує верстки та фронтенду	Зручне візуальне середовище, анімації, UI-Canvas, швидка розробка

Як видно з таблиці 2.1 та таблиці 2.2, Python є чудовим вибором для побудови серверної частини та аналітичної складової системи, однак він не забезпечує необхідної гнучкості для створення інтерактивного графічного інтерфейсу користувача [41].

JavaScript та веб-фреймворки (React, Angular, Vue.js) мають потужні інструменти для розробки динамічних інтерфейсів, але створення єдиної кросплатформної системи з подальшою мобільною підтримкою потребує додаткових засобів, таких як Cordova або React Native.

C# у поєднанні з Unity дозволяє отримати цілісний технологічний стек: інтерактивний користувацький інтерфейс, роботу з базами даних і зручну архітектуру для реалізації рекомендаційної логіки. Крім того, Unity має вбудовані засоби локалізації, роботи з ресурсами, а також підтримку пакетів, що спрощує розробку масштабованого освітнього продукту [44, 45].

Таблиця 2.2 Порівняльний аналіз можливих мов і середовищ розробки

Критерій	Python	JavaScript/Web	C#/Unity
Підтримка алгоритмів машинного навчання/ рекомендацій	Сильна (NumPy, scikit-learn, TensorFlow)	Обмежена (TensorFlow.js, Brain.js)	Помірна (через ML.NET або зовнішні API)
Кросплатформність	Вимагає адаптації для кожної ОС	Висока (будь-який браузер)	Висока (Windows, Android, iOS, WebGL)
Робота з базами даних / API	Добра (SQLAlchemy, Firebase API, REST)	Добра (Fetch API, Node.js)	Добра (Firebase SDK, UnityWebRequest, SQL через .NET)
Продуктивність / оптимізація	Відносно низька (інтерпретована мова)	Середня	Висока, завдяки компіляції C#
Рівень складності для інтерактивного освітнього контенту	Високий (вимагає значних зусиль для UI)	Середній	Низький — Unity надає готові UI-компоненти
Зручність розширення та підтримки	Висока	Середня (через фреймворки)	Висока завдяки модульній структурі та ООП

Після аналізу альтернативних варіантів для реалізації рекомендаційної системи було обрано зв’язку C# + Unity, оскільки вона поєднує у собі високу продуктивність, модульність архітектури, кросплатформність і зручність розробки інтерактивного інтерфейсу користувача. Unity — це сучасне середовище, що підтримує візуальне проєктування UI, обробку подій, анімацію,

роботу з ресурсами та логіку застосунку, що дозволяє створити цілісну систему без необхідності розділяти проєкт на окремі фронтенд- і бекенд-компоненти.

Unity — це потужне середовище розробки інтерактивних додатків і ігор, яке підтримує понад 25 платформ (Windows, Android, iOS, WebGL, macOS тощо) [44, 45]. Воно базується на компонентно-орієнтованій архітектурі, де будь-який об’єкт складається з набору компонентів (Scripts, Colliders, Renderers, UI Elements), що визначають його поведінку.

У контексті освітньої рекомендаційної системи Unity виступає не лише інструментом для побудови графічного інтерфейсу, а й повноцінною платформою для:

- Розробки інтерактивного UI (Canvas, Layout Group, ScrollView, Button, TextMeshPro);
- Побудови діалогових вікон і логіки тестування користувача;
- Анімації елементів інтерфейсу за допомогою DOTween або аніматорів;
- Завантаження динамічного контенту через систему Addressables;
- Інтеграції з Firebase для зберігання результатів користувачів, статистики, аналітики тощо.

Unity також підтримує C# скрипти, які виконуються в реальному часі, що дозволяє оперативно змінювати логіку роботи системи, створювати покрокові опитування або обробляти дії користувача в діалоговому режимі.

Під час розробки проєкту використовуються такі технології та компоненти:

1. Unity Package Manager (UPM) — для керування залежностями проєкту. Дозволяє підключати зовнішні бібліотеки (наприклад, Firebase SDK або Zenject) безпосередньо з репозиторіїв.
2. Firebase SDK — хмарна платформа Google, яка забезпечує [48]:
    1) Зберігання даних користувача (Firebase Realtime Database / Firestore);

2) Авторизацію користувачів (Firebase Authentication);
3) Аналітику взаємодії (Firebase Analytics);

3. Zenject — бібліотека для інверсії залежностей (Dependency Injection), яка дає змогу розділити логіку системи на незалежні модулі [42]. Це покращує тестованість і масштабованість застосунку.
4. DOTween — бібліотека для створення анімацій (рух, зміна кольору, прозорості, позиції тощо). Використовується для плавних переходів і підвищення візуальної привабливості інтерфейсу.
5. TextMeshPro — розширений компонент для роботи з текстом у Unity, що дозволяє гнучко налаштовувати шрифти, локалізацію, стилі та підтримує динамічне оновлення текстів інтерфейсу [44].
6. Unity Addressables System — технологія для динамічного завантаження ресурсів (текстів, ілюстрацій, JSON-файлів) під час роботи програми [44]. Це дає змогу оновлювати контент без перевипуску всього додатка.
7. Firebase Hosting / Google Cloud VM — для розміщення backend-компонентів, зокрема власного Verdaccio-сервера (для керування пакетами Unity) або бази користувацьких даних [48].
8. .NET API (System.Data, System.Net.Http) — для роботи з базами даних, виконання REST-запитів і оброблення JSON-структур результатів [46, 47].

Прикладом може бути використання виразів LINQ для фільтрації або ранжування мов програмування за певними критеріями — рівнем складності, сферою застосування чи популярністю серед розробників.

Ще однією важливою характеристикою C# є підтримка асинхронного програмування (async/await), що дозволяє будувати додатки, які не блокують інтерфейс під час обробки даних або запитів до серверу. У контексті рекомендаційної системи це забезпечує плавну взаємодію користувача з інтерфейсом навіть під час завантаження аналітичних результатів чи взаємодії з базою даних.

Мова також надає потужну систему роботи з об’єктами (ООП), що дозволяє чітко розмежовувати класи користувачів, структури даних, моделі запитань, результати тестів і правила прийняття рішень. Наприклад:

- клас UserProfile зберігає інформацію про користувача (вік, рівень знань, попередні відповіді);
- клас Question описує запитання та варіанти відповідей;
- клас RecommendationEngine реалізує логіку аналізу результатів і формування порад;
- клас ProgrammingLanguage містить характеристики мов програмування (тип, складність, сфера застосування, популярність).

Така структуризація дозволяє легко змінювати або розширювати систему — додавати нові правила, мови чи критерії оцінки без зміни основного коду.

Мова C# активно використовується у створенні експертних систем, аналітичних панелей, навчальних платформ і корпоративних рекомендаційних сервісів, що підтверджує її гнучкість і надійність.

Зокрема, C# є основою таких напрямів:

- Корпоративні системи рекомендацій — Microsoft Dynamics, які аналізують дії користувачів і рекомендують товари або контент;
- Освітні платформи — навчальні ігри та тренажери на Unity, що формують персоналізовані сценарії навчання;
- Системи технічної підтримки — програми, які використовують базу знань для автоматичного надання відповідей користувачу;
- Інтерактивні навчальні середовища — рішення на Unity, які поєднують тести, діагностику рівня підготовки та візуалізовані підказки.

Таким чином, використання C# цілком відповідає сучасним тенденціям — поєднанню аналітичної логіки рекомендаційних систем із високим рівнем інтерактивності та візуалізації, необхідним для освітнього контенту.

Ще однією важливою перевагою є екосистема .NET, яка надає інструменти для:

- обробки даних (System.Data, System.Text.Json),
- створення REST API-запитів (HttpClient),
- реалізації алгоритмів машинного навчання (ML.NET),
- роботи з файлами та хмарними сервісами,
- тестування логіки програми (xUnit, NUnit).

Завдяки цьому C# може бути не лише мовою для клієнтської частини в Unity, а й основою для серверної логіки рекомендаційної системи. У майбутньому можливо реалізувати розподілену структуру, де Unity забезпечує інтерфейс і збір даних, а серверна частина на .NET аналізує результати, виконує алгоритми ранжування й повертає готові рекомендації користувачу.

Вибір середовища Unity для реалізації рекомендаційної системи зумовлений необхідністю створення інтерактивного, візуально привабливого та кросплатформного освітнього продукту. На відміну від класичних середовищ програмування, Unity надає не лише інструменти для написання коду, але й повноцінний набір засобів для проєктування інтерфейсу, керування ресурсами, створення анімацій і розгортання програми на різних пристроях.

Основні причини вибору Unity такі:

1. Кросплатформність і масштабованість.

   Unity дозволяє збирати проєкти під Windows, Android, iOS, WebGL, що забезпечує широке охоплення користувачів і спрощує розповсюдження системи серед студентів, викладачів і незалежних користувачів. Завдяки цьому рекомендаційна система може функціонувати як мобільний додаток, десктоп-програма або навіть вебзастосунок без необхідності повної переробки коду.

2. Інтерактивність та гнучкий користувацький інтерфейс.

   Оскільки система має форму діалогового застосунку з покроковими запитаннями, результатами тестів та рекомендаціями,

важливо забезпечити зручність і наочність взаємодії користувача з системою.

Unity має потужний інструментарій UI (Canvas, Layout Groups, Scroll Views, TextMeshPro), який дозволяє створювати адаптивні інтерфейси без залучення додаткових бібліотек.

3. Підтримка анімацій і візуальних ефектів.

   В освітніх проєктах важливо утримувати увагу користувача. Unity надає засоби для створення плавних переходів, реакцій інтерфейсу, підсвічування важливих елементів тощо. Це сприяє кращому сприйняттю матеріалу та формує відчуття “живої взаємодії” з системою.

4. Компонентно-орієнтована архітектура.

   Кожен елемент у Unity — це окремий об’єкт зі своїми властивостями та скриптами. Такий підхід спрощує розробку, тестування і подальше масштабування системи. Наприклад, модуль логіки рекомендацій можна реалізувати у вигляді окремого компонента, який легко оновлюється або замінюється без впливу на решту системи.

5. Інтеграція з зовнішніми сервісами.

   Unity має готові SDK для підключення Firebase, Google Cloud, REST API, що дозволяє зберігати результати користувачів, синхронізувати дані та отримувати аналітику. Для рекомендаційних систем це особливо важливо, адже дані користувачів є основою для вдосконалення алгоритмів.

6. Активна екосистема та спільнота розробників.

   Unity має величезну кількість офіційних пакетів, плагінів та інструментів (через Unity Asset Store), що дозволяє значно скоротити час розробки. При цьому доступна велика база навчальних матеріалів, прикладів і відкритих проєктів.

7. Підтримка освітніх ігор і тренажерів.

Unity широко використовується у сфері EdTech — для створення навчальних симуляторів, інтерактивних курсів, тренажерів і візуалізацій. Це робить платформу природним вибором для побудови системи, яка поєднує елементи експертної логіки та гейміфікації навчального процесу.

Для підтвердження доцільності вибору Unity розглянемо коротке порівняння з іншими можливими середовищами таблиця 2.3.

Таблиця 2.3 Порівняння середовищ розробки для створення освітньої рекомендаційної системи

Середовище	Переваги	Недоліки
WinForms/WPF	Простота реалізації логіки, інтеграція з .NET	Обмежені можливості UI, лише десктоп
Unreal Engine	Висока графіка, потужна продуктивність	Надмірна складність для навчальних систем, велика вага проєкту
Godot	Легка вага, відкрите ПЗ	Менша підтримка пакетів і слабша інтеграція з Firebase
Unity	Баланс продуктивності, гнучкості, зручності UI, кросплатформності	Потребує оптимізації ресурсів, але має найкращу документацію

Як видно з таблиці 2.3, Unity поєднує ключові характеристики, які роблять її оптимальною для створення освітньої рекомендаційної системи, а не лише для ігор. Вона дозволяє реалізувати складну логіку, зберігаючи простоту візуального дизайну та легкість оновлення контенту.

Отже, середовище Unity обрано для реалізації системи з таких причин:

- забезпечує високу інтерактивність та зручний візуальний інтерфейс,

- підтримує кросплатформний розвиток (Android, Windows, IOS, WebGL),
- дозволяє легко інтегрувати зовнішні сервіси та бази даних,
- має розвинену систему управління ресурсами (Addressables),
- підтримує розширювану архітектуру, що сприяє модульному розвитку системи.

Таким чином, вибір Unity є логічним і технічно обґрунтованим рішенням для створення освітньої рекомендаційної системи, яка поєднує аналітичну логіку з інтуїтивним та динамічним інтерфейсом.

Основна логіка рекомендаційної системи реалізується на мові програмування C#, що є повністю сумісною з Unity. Ця мова забезпечує потужну об’єктно-орієнтовану модель, яка дає змогу ефективно структурувати код за принципом Model–View–Presenter (MVP). Такий підхід спрощує тестування, повторне використання компонентів і розширення системи.

C# також надає зручні інструменти роботи з базами даних (через ADO.NET, Entity Framework або API-запити), що дозволяє зберігати профілі користувачів, результати тестів і сформовані навчальні плани в централізованому сховищі.

Середовище Unity має ряд переваг, критично важливих для освітньої системи з рекомендаційним модулем:

- Кросплатформність. Застосунок, створений у Unity, можна без значних змін експортувати під Android, Windows, WebGL чи iOS, що робить систему універсальною для різних пристроїв.
- Побудова інтуїтивного UI. Unity UI Toolkit і Canvas дозволяють швидко створювати екрани тестування, діалоги з покроковими питаннями, анімації переходів і відображення рекомендацій.

- Система управління ресурсами (Addressables). Дозволяє ефективно завантажувати дані, тексти та зображення, що особливо важливо для мультимовних навчальних застосунків.
- Використання пакетів і залежностей. Unity Package Manager спрощує інтеграцію сторонніх бібліотек (наприклад, DOTween для анімацій, Zenject для інверсії залежностей, Firebase SDK для збереження даних у хмарі).
- Можливість розширення рекомендаційної логіки. Через сумісність із .NET можна реалізувати елементи аналітики та машинного навчання, підключаючи ML.NET або власні API-сервіси.

З практичного погляду, Unity також забезпечує візуальне проєктування, що дає змогу швидко створювати інтерактивні прототипи й тестувати поведінку користувача в реальному часі. Це особливо важливо для навчальних систем, де користувацький досвід є ключовим фактором ефективності.

Таким чином, використання C# у поєднанні з Unity дозволяє реалізувати не лише алгоритмічну логіку рекомендаційної системи, але й створити зручний, адаптивний, кросплатформний інтерфейс, який підвищує якість взаємодії користувача із застосунком та забезпечує основу для подальшого розвитку продукту.

C# — це сучасна об’єктно-орієнтована мова програмування, розроблена компанією Microsoft у складі платформи .NET Framework [46, 47]. Вона поєднує в собі надійність та структуру мов, подібних до Java і C++, але при цьому зберігає зручність і безпечність роботи з пам’яттю.

C# підтримує принципи інкапсуляції, наслідування, поліморфізму, що робить її зручною для побудови модульних архітектурних систем.

Основні переваги C# для даного проєкту:

- Сумісність із Unity. Unity використовує C# як основну мову для скриптів, що описують поведінку об’єктів, логіку користувацького інтерфейсу, збереження даних тощо.

- Підтримка архітектурних шаблонів. Мова добре підходить для реалізації патернів MVP, MVC, Dependency Injection, що дозволяє розділити логіку та візуальну частину системи [41, 42, 43].
- Багата стандартна бібліотека. C# має вбудовані засоби для роботи з колекціями, файлами, JSON-даними, HTTP-запитами, асинхронними процесами (async/await), що активно використовуються в проєкті.
- Можливість підключення бібліотек .NET. Це дозволяє реалізувати обробку даних, аналітику або навіть елементи машинного навчання через ML.NET.

Дані зберігаються у вигляді документів та колекцій Firebase Firestore, що дозволяє швидко виконувати вибірки, фільтрування та синхронізацію між пристроями [48].

Для забезпечення узгодженості структур даних використовується попередньо визначена JSON-схема, яка задає обов’язкові поля, типи даних і правила валідації. Наприклад, колекція Questions містить документи такого типу:

```
{
  "id": "Q102",
  "text_ua": "Який ваш рівень досвіду у програмуванні?",
  "text_en": "What is your programming experience level?",
  "type": "single_choice",
  "options": [
    {"text_ua": "Початківець", "weight": 1},
    {"text_ua": "Середній рівень", "weight": 2},
    {"text_ua": "Досвідчений", "weight": 3}
  ]
}
```

Такий формат забезпечує гнучкість структури — нові запитання або варіанти відповідей можна додавати без зміни схеми бази. Приклад даних у базі даних представлений на рисунку 2.2.

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1 of 1

	language_id [PK] uuid	name text	paradigm text[]	difficulty smallint	area_of_use text[]	popularity_index numeric (5,2)	created_at timestamp w	updated_at timestamp w
1	03f8f741-ef24-4...	Python	{Multi-paradigm,OOP}	2	{Data,Web,Scripting}	92.50	2025-11-...	2025-11-...
2	12b149b8-d1b4-...	JavaScript	{Event-driven,Functional,OO...	2	{Web,Frontend,Backend}	91.00	2025-11-...	2025-11-...
3	1b686115-8482-...	C#	{OOP}	3	{GameDev,Desktop,Backend}	88.00	2025-11-...	2025-11-...

Рисунок 2.2 – Приклад даних в базі даних в таблиці programming_languages

Для організації роботи з даними створено окремі програмні модулі, які реалізують принцип інкапсуляції та відповідають за різні аспекти роботи системи [42].

1. Модуль управління користувачами (UserManager)

   Відповідає за створення нового профілю, збереження прогресу, авторизацію користувача у Firebase та синхронізацію даних між пристроями. Він також забезпечує локальне кешування профілю для офлайн-режиму.

2. Модуль запитань і відповідей (QuestionManager)

   Завантажує перелік питань із бази даних, відображає їх у правильній послідовності, приймає відповіді від користувача та передає їх контролеру для подальшої обробки. Модуль також відповідає за випадкову генерацію запитань відповідно до тематики та рівня користувача.

3. Модуль рекомендацій (RecommendationEngine)

   Реалізує алгоритм аналізу отриманих відповідей і визначення релевантної мови програмування. На основі вагових коефіцієнтів і порогових значень формує набір рекомендацій, ранжує їх за ступенем відповідності та передає результати контролеру.

4. Модуль формування навчального плану (LearningPlanBuilder)

   Генерує індивідуальний навчальний маршрут, виходячи з вибраної мови програмування, рівня користувача та тем, які потребують

вивчення. Модуль має гнучку структуру та дозволяє додавати нові курси або розділи без зміни логіки системи.

5. Модуль аналітики (AnalyticsManager)

   Здійснює збір статистичних даних щодо проходження опитувань, результатів тестів і поведінкових показників користувачів. Ця інформація надсилається до Firebase Analytics і може використовуватися для подальшого вдосконалення рекомендаційного механізму.

6. Модуль локалізації (LocalizationManager)

   Відповідає за багатомовність інтерфейсу та контенту. Підтримує динамічну зміну мови без перезапуску програми й завантажує текстові ресурси через систему Unity Localization.

Під час проєктування модулів було дотримано таких принципів:

- незалежність компонентів – кожен модуль виконує окрему функцію та може бути протестований автономно;
- єдині інтерфейси взаємодії – зв’язок між модулями здійснюється через визначені API;
- інверсія залежностей – заміна одного модуля (наприклад, алгоритму рекомендацій) не впливає на роботу інших;
- масштабованість – передбачено можливість додавання нових мов програмування, типів питань і навчальних планів без зміни базової архітектури;

Проєктування бази даних і програмних модулів системи побудовано на принципах структурованості, гнучкості й узгодженості. Обрана модель зберігання даних у Firebase у поєднанні з модульною архітектурою Unity забезпечує надійне функціонування системи, легкість оновлення контенту та можливість масштабування. Розроблена структура створює основу для реалізації покрокового механізму оцінювання знань користувача та формування персоналізованих навчальних рекомендацій.

2.3 Зберігання та обробка тестових питань

Однією з ключових складових рекомендаційної системи є механізм покрокового опитування, який використовується для первинної діагностики рівня знань користувача та визначення його навчального профілю. Саме результати цього етапу є основою для подальшого формування персоналізованих рекомендацій щодо вибору мови програмування та навчальних курсів.

Логіка опитування реалізована у вигляді послідовного діалогу з користувачем, під час якого система пропонує серію запитань із кількома варіантами відповідей [51].

Для підвищення точності діагностики система підтримує кілька форматів тестових завдань, які дозволяють охопити різні аспекти знань та поведінкових характеристик користувача. До таких типів належать [40]:

- Закриті питання з однією правильною відповіддю — дозволяють швидко визначити базові знання.
- Закриті питання з кількома правильними відповідями — оцінюють здатність аналізувати кілька коректних тверджень.
- Питання на відповідність — виявляють уміння встановлювати логічні зв’язки між елементами.
- Питання на встановлення правильної послідовності — перевіряють розуміння алгоритмів та процесів.
- Відкриті питання — дають змогу оцінити сформованість термінології та здатність користувача формулювати думки власними словами.

Для усіх закритих типів запитань відповіді кодуються у вигляді бітових полів, де кожен варіант відповіді представлено одним бітом: 1 — обрано / правильно, 0 — не обрано / неправильно.

Цей спосіб подання забезпечує компактність даних і дає можливість застосовувати формальні математичні методи для аналізу відповідей.

Відкриті відповіді зберігаються у вигляді текстових рядків і обробляються окремо.

На основі відповідей система обчислює попередній рівень користувача (Beginner, Medium або Advanced) та враховує його стиль навчання й мотиваційні особливості.

Оцінювання відповідей здійснюється не лише на рівні вибору варіантів, але й за допомогою формальної математичної моделі [40].

У подальшому, будемо виходити з того, що

- у тестуванні міститься *N* запитань;
- будь-яке запитання має *M* варіантів відповідей;
- шаблон правильної відповіді на i-те запитання представляється точкою $A_i = (a_{i1}, a_{i2}, \dots a_{iM})$;
- k-ту відповідь на i-те запитання можна представити у вигляді точки $B_i = (b_{ik1}, b_{ik2}, \dots b_{ikM})$;
- де кожен компонент є бітовою ознакою (0 або 1).

Тоді міра відстані між відповіддю користувача та шаблоном правильної відповіді обчислюється як:

$$p(A_i, B_{ik}) = \sqrt{\sum_{j=1}^{M}(a_{ij} - a_{ikj})^2}\ .$$

Квадрат цієї величини визначає кількість помилок у відповіді:

$$p^2(A_i, B_{ik}) = \sum_{j=1}^{M}(a_{ij} - a_{ikj})^2\ .$$

Максимальна кількість можливих помилок обмежується:

$$p^2(A_i, B_{ik}) \le M.$$

Таким чином, система може оцінити не тільки правильність відповіді, але й ступінь її “наближеності” до правильної, що дозволяє точніше визначити рівень підготовки користувача та підвищує якість формування персональних рекомендацій.

Після запуску програми користувач проходить короткий процес адаптації, у якому система запитує, чи має він досвід програмування.

- Якщо користувач не має досвіду, система пропонує набір запитань, що допомагають визначити його сприйняття нової інформації, стиль мислення, мотивацію та навчальні уподобання. Такі питання орієнтовані на новачків і мають психологічно-мотиваційний характер — вони виявляють, чи користувач схильний до аналітичного мислення, творчого підходу, візуального навчання чи практичних експериментів.
- Якщо користувач має попередній досвід, то опитування переходить у технічну форму, де питання спрямовані на самооцінку професійних навичок, підходів до вирішення задач, знання алгоритмів, інструментів і командної роботи.

Для ілюстрації роботи механізму оцінювання розглянемо приклад [40]. Нехай правильна відповідь має вигляд бітового вектора 0101, що відповідає вибору варіантів B та D.

Користувач обрав варіанти A та B, що відповідає вектору 1100.

Тоді бітова операція XOR дає результат:

$$0101 \text{ XOR } 1100 = 1001,$$

що містить 2 одиниці, тобто дві помилки — вибір неправильного варіанта та пропуск правильного.

Отримане значення $p^2 = M$ використовується під час визначення рівня підготовки користувача та передається до RecommendationEngine як частина діагностичних даних.

Окрім тестових запитань із фіксованими варіантами відповідей, система передбачає опрацювання запитань відкритого типу, що дозволяє точніше оцінити сформованість знань користувача у ситуаціях, де важливим є не вибір із запропонованих варіантів, а самостійне формулювання відповіді.

Такі запитання використовуються для визначення того, наскільки повно і коректно користувач здатен описати поняття, процес або алгоритм.

Проблема автоматизованої перевірки відкритих відповідей полягає у відсутності суворої структурованості: користувач може використовувати синоніми, змінювати порядок слів, пропускати другорядні елементи або розширювати опис за рахунок власних формулювань. Саме тому методи аналізу відкритих відповідей ґрунтуються не на пошуку точного збігу, а на оцінці повноти та релевантності семантичних складових.

У роботі використано підхід, запропонований у працях Ю. С. Антонова та інших дослідників, де описано механізми:

- оцінки повноти відповіді без суворої послідовності слів [38, 37];
- визначення ключових семантичних одиниць, які обов'язково мають бути присутні у відповіді [37];
- побудови словника еталонних фраз та їхніх варіацій, що дозволяє враховувати синонімію та морфологічні трансформації [40,40];
- автоматизованої перевірки якості відповідей у системах контролю знань, де оцінюється відповідність структури та змісту встановленим критеріям [39];

Для кожного відкритого запитання формується еталонний набір ключових понять, який зберігається у базі даних у вигляді масиву термінів та їхніх лексичних варіацій. Під час обробки відповіді система виконує:

1. нормалізацію тексту (lemmatization, lowercase);
2. розбиття на токени та фільтрацію службових слів;
3. пошук ключових одиниць у відповіді, включно з синонімами та морфологічними варіаціями;
4. обчислення коефіцієнта повноти (0–1), який показує, наскільки відповідь наближена до еталонної;

Отриманий коефіцієнт включається у профіль користувача та впливає на підсумкову оцінку рівня підготовки. Таким чином, система поєднує як

формальні, так і семантичні методи аналізу відповідей, що робить процес тестування гнучкішим та ближчим до експертної оцінки.

Кожен сценарій складається з 10 запитань із чотирма варіантами відповідей, що охоплюють когнітивні, поведінкові та мотиваційні аспекти. Наприклад, новачкам пропонуються питання типу “Що вас мотивує навчатися?” або “Як вам найзручніше сприймати нову інформацію?”, тоді як досвідченим користувачам — “Як ви підходите до вирішення задач?” чи “Як часто оновлюєте свої знання?”.

Опитування побудоване за принципом поступового розкриття змісту [51]. На кожному етапі користувач бачить лише одне запитання, що дозволяє уникнути перевантаження інформацією та підтримувати концентрацію. Перехід до наступного кроку відбувається лише після вибору відповіді, завдяки чому система зберігає повну послідовність дій користувача.

Також застосовано механізм гілкування сценаріїв — залежно від відповідей на попередні запитання система може адаптувати наступні кроки (наприклад, пропонувати питання відповідно до обраної мотивації або рівня впевненості у своїх знаннях).

Усі відповіді зберігаються локально в профілі користувача, а після завершення опитування — передаються до бази даних. Це дозволяє повторно використовувати результати під час оновлення рекомендацій без необхідності повторного проходження тесту.

Після завершення опитування система виконує автоматичний аналіз відповідей. Кожна відповідь має певну вагу, що впливає на підсумкову оцінку користувача. На основі комбінації відповідей формується початковий рівень:

- Beginner (початковий) — користувач не має досвіду, схильний до візуального навчання, краще засвоює матеріал через приклади, ігри або практичні завдання.

- Medium (середній) — користувач має базові знання або навчальний досвід, може розв’язувати нескладні задачі, прагне поглибити розуміння принципів програмування.
- Advanced (просунутий) — користувач має практичний досвід або проєктну діяльність, розуміє архітектурні принципи, орієнтований на розробку повноцінних програм чи ігор.

Після обчислення рівня система автоматично переходить на екран профілю користувача, де відображається його ім’я, мета навчання (для себе чи для роботи) і визначений рівень. На основі цього рівня формується добірка початкових курсів.

Результати опитування є вхідними даними для модуля рекомендацій. Отриманий рівень користувача використовується для:

- автоматичної фільтрації навчальних курсів за складністю;
- визначення послідовності подальших тем і рівнів складності;
- створення адаптивного навчального плану;

Взаємодія модулів побудована так, що після завершення опитування результати передаються до сервісу рекомендацій, який аналізує їх разом із даними профілю користувача та базою курсів.

Інтерфейс опитування реалізовано в мінімалістичному стилі з акцентом на зручність. Кожне запитання відображається окремо на екрані у вигляді картки з текстом і варіантами відповідей. Передбачено:

- лічильник, який показує номер поточного питання (наприклад, 7 з 10);
- плавні переходи між кроками (анімації, реалізовані засобами DOTween);
- локалізацію інтерфейсу (українська / англійська мова);
- можливість повернення до попереднього запитання, якщо користувач хоче змінити відповідь;

Після завершення тестування користувач отримує коротке повідомлення про визначений рівень і кнопку переходу до наступного етапу — заповнення профілю.

Для збереження відповідей використовується внутрішній масив, який оновлюється після кожного кроку. Після завершення опитування об’єкт результатів формується у формат JSON і зберігається у Firebase.

У системі також реалізовано перевірку коректності введення: якщо користувач не обрав жодної відповіді, перехід до наступного кроку блокується, а інтерфейс підсвічує повідомлення про необхідність вибору.

Завдяки модульному підходу структуру запитань можна змінювати чи розширювати без втручання у код — усі тексти, варіанти та логіка їх показу зберігаються у вигляді окремих ресурсів (JSON-файлів), які заватажуються динамічно через систему Addressables.

Результати обчислення помилок для кожного запитання формують набір кількісних показників, який характеризує сильні та слабкі сторони користувача. Ці значення доповнюють первинну класифікацію на рівні Beginner, Medium або Advanced та відіграють ключову роль під час подальшого аналізу.

Після завершення опитування система передає отримані діагностичні дані до модуля RecommendationEngine, де вони використовуються для визначення оптимальної мови програмування та формування персоналізованої траєкторії навчання.

Запропонована логіка покрокового опитування поєднує простоту використання з гнучкістю налаштувань і дозволяє точно визначити рівень підготовки користувача навіть за обмеженої кількості питань. Результати діагностики слугують відправною точкою для формування індивідуального навчального маршруту та добору релевантних курсів.

2.4 Формування плану навчання та вибір релевантної мови програмування

Після завершення початкового опитування система отримує ключові показники профілю користувача: його рівень підготовки (Beginner, Medium,

Advanced), стиль навчання, мотивацію та індивідуальні особливості сприйняття матеріалу [50]. Ці дані є основою для роботи механізму рекомендацій, який визначає найбільш релевантні навчальні курси та формує персоналізований план навчання.

Функціонування цього механізму поєднує фільтрацію, сортування та аналіз результатів користувача, включаючи інформацію про допущені помилки під час проходження навчальних модулів [1, 3, 5].

Система використовує три основні джерела інформації:

1. Результати початкового опитування

   Вони визначають базовий рівень користувача, його поведінкові та мотиваційні характеристики. На основі відповідей система встановлює одну з трьох категорій складності:

   - Beginner
   - Medium
   - Advanced

   Цей рівень визначає стартову позицію у навчанні.

2. Профіль користувача

   Містить додаткову інформацію, таку як ім’я, вік і мета навчання, наприклад: “для себе”, “для роботи”. Мета навчання впливає на порядок рекомендацій: для тих, хто вчиться “для роботи”, система у пріоритет ставить курси, пов’язані з прикладними задачами, тоді як для користувачів “для себе” — курси із базовими концепціями та ігровими елементами.

3. База курсів

   Кожен курс у системі має щонайменше три атрибути:

   - рівень (Beginner / Medium / Advanced),
   - тематику або набір тегів, наприклад: логіка, змінні, цикли, архітектура,
   - короткий опис та унікальний ідентифікатор.

Саме тегування дозволяє встановити зв’язок між результатами тестів та релевантними темами.

У процесі добору курсів система застосовує двоетапний механізм:

1. Фільтрація

   На першому етапі система відкидає курси, рівень яких не відповідає рівню користувача. Наприклад, якщо користувач визначений як Beginner, він не побачить просунутих курсів типу “Побудова архітектури проєкту” або “Читання складного коду”.

2. Сортування

   Коли залишаються лише релевантні курси, система ранжує їх за кількома критеріями:

   - відповідність меті навчання,
   - наявність відповідності стилю навчання,
   - тематика курсу,
   - чи проходив користувач цей курс раніше,
   - наявність виявлених прогалин у знаннях за тегами.

Таким чином, користувач отримує перелік курсів, які найкраще відповідають його поточному стану та потребам.

Важливою частиною системи є здатність адаптувати рекомендації на основі реального прогресу користувача.

У кожному курсі можуть бути тестові запитання, які мають один або кілька тегів — цикли, умови, класи, логічні операції тощо.

Після проходження тестування система:

1. зберігає інформацію про кожну неправильну відповідь;
2. фіксує тег теми, у якій допущено помилку;
3. підраховує частоту повторюваних помилок.

Наприклад, якщо користувач тричі помилився у питаннях, пов’язаних із циклами, тег “цикли” матиме найбільшу вагу. Це робить процес навчання не

лінійним, а адаптивним, що є ключовою перевагою персоналізованих освітніх систем.

На основі вхідних даних система автоматично створює персональний план навчання, який складається з набору модулів і тем. План формується та оновлюється динамічно.

Основні елементи плану:

- Стартові курси, рекомендовані залежно від рівня користувача;
- Розширені курси, які відповідають меті (наприклад, для ігор, вебу, роботи з даними);
- Компенсаційні курси, спрямовані на усунення прогалин у знаннях (визначаються за тегами неправильних відповідей);
- Проєктні курси, що дозволяють закріпити знання на практиці.

Навчальний план оновлюється у міру проходження курсів. Якщо користувач успішно проходить тестові завдання і демонструє прогрес, система може підняти його рівень, після чого відкриваються більш складні курси.

Запитання з початкового опитування дозволяють визначити, як користувачу найзручніше навчатися:

- через практичні завдання,
- через відео й пояснення,
- через ігри,
- через візуальні матеріали.

Це впливає на порядок рекомендацій. Наприклад:

- користувачу з високою візуальною пам’яттю система пропонує курси з великою кількістю ілюстрацій;
- користувачу, який добре реагує на виклики, пропонуються курси з міні-завданнями;
- користувачу, що легко дратується при помилках, система пропонує легші вступні теми.

Запропонована система має декілька важливих переваг:

- адаптивність — рекомендації змінюються відповідно до успіхів і помилок користувача;
- персоналізація — враховуються інтереси, мета і стиль навчання;
- мінімальне навантаження — користувач не змушений проходити довгі тести;
- прозорість — користувач бачить, чому йому рекомендували той чи інший курс;
- плавне зростання складності — рівень курсів відповідає рівню компетентностей;
- гнучкість розширення — адміністратор може додати нові курси або теги без зміни логіки системи.

Механізм рекомендацій у розробленій системі є поєднанням аналізу початкового опитування та динамічного врахування помилок під час навчання. Такий підхід забезпечує побудову персоналізованого навчального маршруту, який відповідає реальним потребам користувача та сприяє ефективному засвоєнню матеріалу.

2.5 Реалізація основних компонентів системи

Розроблення рекомендаційної системи здійснено у середовищі Unity із застосуванням архітектури MVC, що забезпечило чіткий розподіл відповідальності між модулями, спрощення підтримки та можливість подальшого масштабування програмного продукту.

У системі створено окремий модуль UserModule, що відповідає за управління даними користувача та їх синхронізацію з базою даних. Він містить такі основні компоненти:

- UserModel — структура даних, яка зберігає ім’я, вік, мету навчання та визначений рівень.
- UserController — оброблює події інтерфейсу: заповнення профілю, збереження змін, оновлення рівня користувача після опитування.

- UserService — забезпечує взаємодію з Firebase або локальним кешем, виконує збереження та завантаження профілю.

Модуль працює незалежно від інтерфейсу користувача, що дозволяє легко адаптувати його до нових форматів даних.

Модуль опитування реалізує базову логіку взаємодії користувача з питаннями та отримання відповідей:

- SurveyModel — містить перелік запитань, їх варіанти відповідей і тип сценарію (для новачків або користувачів із досвідом).
- SurveyView — відображає кожне питання у вигляді окремої картки з варіантами відповідей, забезпечує навігацію між кроками, інформує про прогрес.
- SurveyController — визначає послідовність запитань, фіксує відповіді, контролює завершення та викликає механізм визначення рівня.

Після проходження опитування контролер передає результати в модуль рекомендацій для подальшого аналізу.

Основним логічним компонентом є RecommendationModule, який формує добірку курсів і навчальний план. Він складається з таких частин:

- RecommendationEngine — обробляє дані з опитування та профілю, визначає релевантні курси відповідно до рівня користувача, його мети та стилю навчання.
- WeakTopicAnalyzer — аналізує помилки користувача під час проходження уроків або тестів і формує перелік тем, які варто повторити.
- LearningPlanBuilder — створює індивідуальний навчальний план, що включає стартові, компенсуючі й проєктні курси.

Реалізований підхід дозволяє формувати рекомендації динамічно — система оновлює добірку курсів у міру того, як користувач проходить навчальні матеріали.

Навчальні курси є центральним елементом освітньої системи. Їх структура реалізована як окремий модуль, що включає:

- CourseModel — опис курсу (назва, рівень, тематика, короткий опис, теги).
- CourseView — відображає список доступних курсів, дозволяє додавати їх до “Моїх курсів” та переглядати коротку інформацію.
- CourseController — керує фільтрацією курсів відповідно до рівня користувача, а також додає або видаляє курс із персонального списку.
- LessonController — відповідає за відображення контенту уроку, текстових пояснень, ілюстрацій і міні-тестів.

Уроки реалізовані у вигляді окремих модулів, які можна розширювати за необхідності, додаючи нові теми та тести.

Збереження курсів, відповідей і результатів тестів виконується через DataModule, який абстрагує роботу з базою даних:

- DataRepository — інтерфейс взаємодії з Firebase Firestore.
- LocalCache — використовується для офлайн-режиму.
- ProgressTracker — оновлює прогрес користувача в курсах, фіксує пройдені уроки та статуси (не розпочато, у процесі, завершено).

Механізм забезпечує синхронізацію локальних даних із сервером, коли з’єднання відновлюється, що дозволяє користувачу працювати навіть без Інтернету.

Інтерфейс реалізовано у мінімалістичному стилі для забезпечення простоти взаємодії:

- екран опитування — послідовні картки з питаннями;
- екран профілю — введення особистих даних та рівня;
- екран “Всі курси” — список усіх курсів з автоматичним підсвічуванням релевантних;
- екран “Мої курси” — курси, вибрані користувачем;

- екран уроку — навчальний матеріал, ілюстрації та короткі тести.

Нижнє меню забезпечує швидку навігацію між основними розділами: “Профіль”, “Всі курси”, “Мої курси”. Приклад реалізації інтерфейсу представлений на рисунку 2.3.

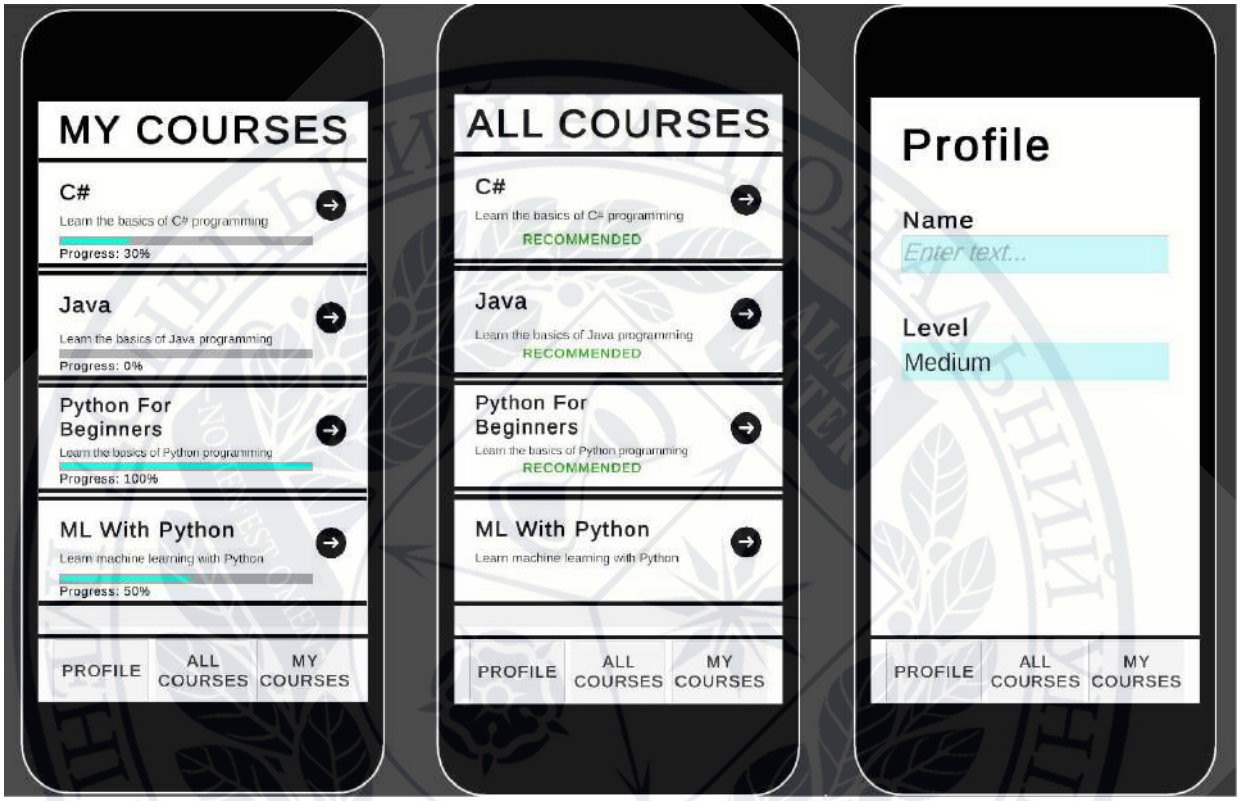

Рисунок 2.3 – Приклад реалізації інтерфейсу програми

Взаємодія між компонентами реалізована через визначені інтерфейси та систему ін’єкції залежностей. Це забезпечує:

- можливість заміни окремих модулів без зміни основної логіки,
- розширюваність системи новими курсами та навчальними планами,
- простоту тестування окремих частин,
- підтримку різних сценаріїв навчання.

Завдяки цьому система працює як цілісний механізм: користувач проходить опитування, далі отримує рівень, потім переглядає рекомендовані курси, після цього проходить уроки, а в кінці система аналізує помилки та оновлює рекомендації.

Висновок до розділу 2

У цьому розділі було розглянуто повний цикл проєктування та реалізації рекомендаційної системи, призначеної для визначення рівня підготовки користувача та формування персоналізованого навчального маршруту. Спершу було обґрунтовано вибір технологічного стеку, зокрема використання мови програмування C# та середовища Unity, що забезпечують кросплатформність, модульність та високу гнучкість у розробленні інтерактивних освітніх застосунків. Подальший аналіз архітектури дозволив сформувати структуровану модель системи на основі підходу MVC, що гарантує чітке розділення відповідальностей та можливість масштабування.

Була спроєктована та описана логічна структура бази даних і модулів програми, які відповідають за збереження профілю користувача, його відповідей, результатів тестів і навчальних рекомендацій. Особлива увага приділена логіці покрокового опитування, що дозволяє визначати рівень знань та навчальні уподобання користувача, використовуючи адаптивні сценарії та мотиваційно-поведінкові запитання. На основі отриманих даних реалізовано механізм формування рекомендацій і персоналізованого навчального плану, який враховує стартовий рівень, інтереси користувача та виявлені прогалини у знаннях.

У підсумку реалізований програмний продукт поєднує функціональні можливості опитування, адаптивної рекомендаційної системи та модуля навчальних курсів, утворюючи цілісну інтерактивну платформу для початківців і користувачів із досвідом у сфері програмування. Структура системи дозволяє розширювати набір курсів, вдосконалювати алгоритми рекомендацій та інтегрувати додаткові інструменти аналізу.

РОЗДІЛ 3. Тестування та порівняння розробленої системи з аналогами

3.1 Методика тестування та критерії оцінки якості рекомендацій

Метою тестування розробленої рекомендаційної системи є перевірка коректності роботи алгоритмів формування рекомендацій, стабільності функціонування програмного продукту, а також оцінювання точності підбору мови програмування та побудови індивідуального плану навчання для користувача. Оскільки система поєднує модулі опитування, аналізу відповідей, генерації рекомендацій, формування навчальних планів і взаємодії з хмарною інфраструктурою Firebase, тестування проводилося комплексно та охоплювало як програмну логіку, так і кінцевий результат для користувача.

Тестування здійснювалося в умовах, наближених до реального використання системи. Було використано два основні середовища запуску:

- редактор Unity (Play Mode) — для налагодження логіки, модульного та інтеграційного тестування;
- зібрані білди під цільові платформи, зокрема Android — для перевірки роботи на реальних пристроях, взаємодії з Firebase та оцінки продуктивності;

Для роботи з даними були створені окремі тестові колекції у Firebase Firestore, а також використано тестові облікові записи авторизації. Це дозволило ізолювати експериментальні дані від реальних і багаторазово повторювати сценарії проходження опитувань без впливу на основну базу.

Основними цілями тестування були:

- перевірити працездатність кожного з ключових модулів системи;
- оцінити точність та узгодженість рекомендацій;
- перевірити повноту та логічність згенерованих планів навчання;
- проаналізувати стабільність роботи застосунку та час відгуку при взаємодії з хмарними сервісами;

Функціональне тестування було спрямоване на перевірку коректності роботи основних компонентів, зокрема:

- модуля управління користувачами (UserManager) – створення нового профілю, збереження базових даних (вік, рівень підготовки, мова інтерфейсу), авторизація користувача через Firebase, завантаження та оновлення прогресу, коректна обробка повторного входу в систему;
- модуля запитань і відповідей (QuestionManager) – завантаження переліку запитань із бази даних, відображення їх у правильній послідовності, підтримка різних типів запитань (одиночний вибір, множинний вибір, шкала), обробка введення користувача та передавання відповідей у контрольний модуль;
- модуля рекомендацій (RecommendationEngine) – обчислення сумарних балів за критеріями, застосування вагових коефіцієнтів, формування списку мов-претендентів, вибір основної рекомендованої мови програмування, формування пояснення для користувача;
- модуля побудови навчального плану (LearningPlanBuilder) – формування структури навчального плану (модулі, теми, завдання) відповідно до вибраної мови програмування та рівня користувача, перевірка відсутності “порожніх” або логічно некоректних елементів;

Для кожного модуля було визначено набір тестових сценаріїв, що включали умови початку (початковий стан системи), послідовність дій користувача та очікуваний результат. Сценарії охоплювали як типові випадки використання (коректні дані), так і граничні ситуації — наприклад:

- відсутність підключення до мережі під час завантаження запитань;
- повторне проходження опитування користувачем із тим самим профілем;

- переривання сесії під час формування навчального плану та подальше відновлення;

На рівні модульного тестування перевірялися окремі методи та класи, відповідальні за:

- обробку відповідей і розрахунок сумарних балів за кожним критерієм;
- застосування вагових коефіцієнтів до варіантів відповідей;
- побудову структури навчального плану на основі набору правил (наприклад, включення базових тем для початківців і розширених тем для користувачів із досвідом);

Інтеграційне тестування проводилося для оцінки взаємодії між модулями:

- передавання результатів із QuestionManager до RecommendationEngine;
- формування рекомендації та передавання її у модуль побудови навчального плану;
- узгоджена робота модулів із Firebase Firestore (збереження відповідей, рекомендацій та планів навчання у відповідних колекціях);

Окрему увагу приділено перевірці коректності завантаження ресурсів через Unity Addressables, зокрема JSON-файлів із запитаннями та описами мов програмування. Тестувалися ситуації, коли ресурси вже закешовані локально, та випадки, коли їх потрібно вперше завантажити з віддаленого сховища.

Для більш реалістичної оцінки роботи системи були сформовані типові профілі користувачів із різним рівнем підготовки та цілями навчання. Серед них:

- початківець без досвіду програмування, зацікавлений у вивченні базових концепцій;
- користувач із базовим рівнем, який хоче обрати напрям подальшого розвитку;

- користувач із досвідом, що планує змінити сферу діяльності (наприклад, перейти до веб-розробки або аналізу даних);

Для кожного профілю проводилося повне проходження опитувальника, після чого аналізувалися:

- рекомендована мова програмування;
- пояснення, яке формує система для обґрунтування вибору;
- структура згенерованого навчального плану (наявність базових і просунутих модулів, логічний порядок тем, послідовність ускладнення завдань);

Результати порівнювалися з очікуваними рекомендаціями, визначеними на основі попереднього аналізу вимог до мов програмування та їх сфер застосування. У разі суттєвих розбіжностей виявлялися причини (наприклад, некоректні вагові коефіцієнти або надто жорсткі порогові значення), після чого проводилося коригування правил у RecommendationEngine.

Зважаючи на використання хмарних сервісів, проводилося навантажувальне тестування, метою якого було оцінити:

- час відгуку при завантаженні запитань та збереженні відповідей;
- стабільність роботи при багаторазових послідовних сесіях;
- поведінку системи при нестабільному інтернет-з’єднанні;

Для цього аналізувалися:

- час виконання основних операцій (ініціалізація профілю, завантаження запитань, запис результатів тесту, збереження рекомендацій);
- кількість помилок завантаження ресурсів через Addressables;
- реакція інтерфейсу в момент очікування відповіді від сервера (наявність індикаторів завантаження, відсутність зависань);

Результати навантажувального тестування дозволили підтвердити, що система зберігає працездатність навіть за кількох послідовних проходжень

опитування та багаторазового формування навчальних планів, а час очікування відповіді від Firebase залишається прийнятним для користувача.

Для оцінювання якості роботи рекомендаційної системи використовувалася сукупність кількісних та якісних критеріїв.

До кількісних критеріїв належали:

- точність рекомендацій – частка випадків, коли рекомендована система мов програмування збігалася з очікуваною або експертно визначеною для відповідного профілю користувача;
- відповідність навчального плану рівню користувача – відсутність у плані тем, що суттєво перевищують поточний рівень підготовки без попередніх базових модулів;
- сталість результатів – отримання близьких рекомендацій при повторному проходженні опитування з незначними варіаціями відповідей;

До якісних критеріїв відносилися:

- зрозумілість пояснення рекомендації – наскільки користувач може інтерпретувати причини вибору певної мови програмування;
- логічність структури навчального плану – послідовність подання тем, наявність переходу від простого до складнішого, від базових понять до прикладних задач;
- суб’єктивне сприйняття корисності рекомендації – наскільки запропонований шлях навчання відповідає очікуванням та цілям користувача;

Приклад проходження першого тестування користувача для визначення його рівня знань та його цілі представлений на рисунку 3.1.

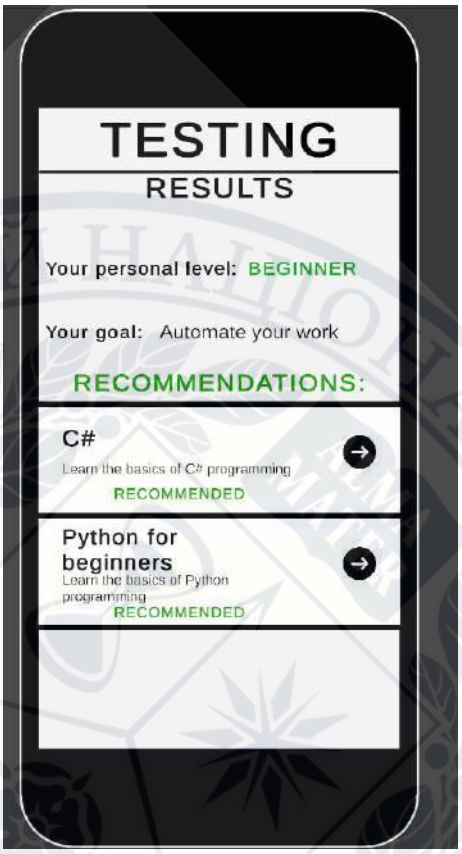

Рисунок 3.1 – Приклад результатів проходження початкового тестування користувача

Комплексне застосування цієї методики тестування дозволило не лише перевірити технічну коректність роботи системи, але й оцінити її придатність для реального використання як інструмента підтримки вибору мови програмування та побудови персоналізованої траєкторії навчання.

3.2 Порівняння функціональності з існуючими аналогами

Для оцінювання конкурентоспроможності та рівня зрілості розробленої рекомендаційної системи було проведено ґрунтовне порівняння її функціональних можливостей із сучасними освітніми платформами, що активно застосовують механізми персоналізації навчання. У минулих розділах

проаналізовано роботу таких систем, як Coursera, Codecademy, Khan Academy та DataCamp, які сьогодні є одними з найбільш популярних і технологічно розвинених рішень для вивчення програмування. Кожна з наведених платформ має власну модель персоналізації, унікальні механізми побудови навчальних траєкторій, різну глибину аналітики та відмінні технологічні підходи до рекомендацій.

Усі ці системи є зрілими, давно присутніми на ринку продуктами, що поєднують у собі аналітику поведінки, машинне навчання та велику кількість навчального контенту. Вони працюють із величезними обсягами даних, накопичують інформацію про активність мільйонів користувачів і забезпечують складну багаторівневу персоналізацію. Саме тому важливо здійснити порівняльний аналіз і визначити місце розробленої системи серед уже існуючих рішень.

Порівняння з провідними навчальними платформами дає змогу визначити сильні сторони розробленої системи, а також виявити елементи, які потенційно можуть бути розширені або вдосконалені в майбутніх версіях. З цією метою було сформовано таблицю 3.1, у якій зведено ключові характеристики аналізованих платформ: механізми рекомендацій, рівень адаптивності, принципи формування навчальних шляхів та технологічні аспекти реалізації. Така узагальнена форма дозволяє порівняти системи між собою та оцінити позицію розробленого продукту серед сучасних аналогів.

Таблиця 3.1 Порівняння загальної логіки персоналізації

Назва	Механізм рекомендацій	Облік рівня користувача	Побудова навчального плану	Технології
Coursera	Гібридний	Тести та поведінкові дані	Автоматична, гнучка	ML-моделі, матрична факторизація
Codecademy	Навчальний шлях на основі практичних вправ	Частково	Фіксовані навчальні траєкторії	Інтерактивні тренажери
Khan Academy	Адаптивна модель, основана на складності й результатах	Адаптивність у реальному часі	Адаптивні модулі	Адаптивні алгоритми
DataCamp	Моделі рекомендацій на основі навичок	Присутня	Навчальні треки	Машинне навчання та статистика
Розроблена система	Експертна система з ваговими коефіцієнтами	Покрокове опитування	Динамічний план на основі рекомендації	Unity, Firebase, експертні правила

На основі проведеного порівняння можна зробити висновки, що розроблена система має більш простий, але контрольований механізм персоналізації, що базується на експертних правилах. На відміну від великих

платформ, вона фокусується саме на виборі мови програмування, а не широкого спектра курсів.

Порівняння функціональних модулів відбувається за наступними показниками:

1. Формування профілю користувача;
2. Механізм рекомендацій;
3. Генерація навчального плану;
4. Технічна реалізація та можливості розширення;

Формування профілю користувача:

- Coursera, DataCamp автоматично збирають поведінкові дані (перегляди, активність, тривалість сесій);
- Khan Academy формує профіль на основі виконання вправ;
- Codecademy визначає рівень користувача після проходження практичних завдань;
- Розроблена система. Профіль створюється через структуроване опитування, яке охоплює рівень досвіду, цілі навчання, інтереси та інші параметри. Такий підхід є простішим, але забезпечує точний контроль над початковими даними;

Механізм рекомендацій:

- Coursera поєднує контентну та колаборативну фільтрацію;
- DataCamp використовує векторні моделі подібності;
- Khan Academy застосовує адаптивні алгоритми, що коригують складність;
- Codecademy формує рекомендації за рахунок інтерактивних практичних вправ;
- Розроблена система. Використовує експертну модель на основі вагових коефіцієнтів, що визначають відповідність між відповідями користувача й характеристиками доступних мов програмування. Перевага — передбачуваність та повна прозорість логіки.

Обмеження — відсутність глибинних алгоритмів, що враховують приховані патерни поведінки;

Генерація навчального плану:

- Coursera пропонує довгі треки з великою кількістю курсів;
- Codecademy використовує покрокові практичні навчальні шляхи;
- DataCamp формує треки на основі навичок;
- Khan Academy створює адаптивні модулі, які підлаштовуються під успішність;
- Розроблена система формує персоналізований навчальний план, який складається з набору модулів, тем і завдань, релевантних обраній мові програмування. Структура плану залежить від: рівня користувача, типу мови (системна, об'єктно-орієнтована, для веб), прогалин у знаннях, визначених через опитування;

У цьому аспекті система найбільше наближена до Khan Academy, оскільки пропонує адаптивні рівневі траєкторії.

Технічна реалізація та можливості розширення:

- Coursera, DataCamp використовують потужні серверні ML-моделі;
- Codecademy має інтерактивний середовище виконання коду;
- Khan Academy забезпечує динамічне оновлення контенту;
- Розроблена система Вбудована в екосистему Unity і Firebase, що забезпечує: гнучке оновлення ресурсів через Addressables, можливість масштабування бази мов і навчальних планів, інтеграцію машинного навчання в майбутньому;

Функціонально система не конкурує з повномасштабними платформами, а є легким модульним рішенням, яке може бути використане як частина більших навчальних продуктів.

Порівняння показало, що розроблена система забезпечує ключові можливості персоналізації, притаманні сучасним освітнім платформам, але реалізовані в компактній і прозорій формі. На відміну від масштабних сервісів,

що використовують складні моделі машинного навчання та великий масив поведінкових даних, створений продукт зосереджений на точному експертному виборі мови програмування та формуванні релевантного навчального плану на основі детального опитування.

Система має меншу функціональну та технологічну складність, однак її перевагою є контрольованість логіки, простота впровадження та можливість подальшого розширення без повної зміни архітектури.

3.3 Аналіз переваг і недоліків розробленої системи

Аналіз результатів тестування та порівняння з існуючими аналогами дозволив визначити ключові сильні та слабкі сторони розробленої рекомендаційної системи. Незважаючи на компактність і відносну простоту реалізації порівняно з масштабними освітніми платформами, система демонструє низку переваг, які роблять її ефективною для задачі вибору мови програмування та формування персоналізованого навчального плану.

До переваг розробленої системи належать:

1. Прозорість та зрозумілість рекомендацій.

   На відміну від моделей на основі машинного навчання, логіка системи базується на чітко визначених вагових коефіцієнтах і правилах. Це дозволяє легко пояснити причини отриманої рекомендації, що підвищує довіру користувача до результатів системи. Усі кроки — від обробки відповідей до вибору мови — є повністю відтворюваними та зрозумілими.

2. Адаптація до різних рівнів користувачів.

   Система враховує рівень досвіду, цілі та інтереси користувача, забезпечуючи персоналізований результат як для початківців, так і для користувачів із досвідом. Реалізоване покрокове опитування дозволяє точно визначити початкову позицію кожного користувача.

3. Гнучкість і простота розширення.

Архітектура системи розроблена таким чином, що дозволяє легко додавати нові мови програмування, змінювати вагові коефіцієнти, доповнювати логіку аналізу відповідей, розширювати модулі навчальних планів. Це робить продукт масштабованим і готовим до подальшої інтеграції додаткових алгоритмів машинного навчання.

4. Легка інтеграція з хмарними технологіями.

   Завдяки використанню Firebase Firestore, Authentication та Addressables система зберігає дані в хмарі, підтримує синхронізацію між пристроями, динамічно завантажує ресурси без оновлення всього застосунку. Це дає можливість гнучко підтримувати й оновлювати контент.

5. Комфортний користувацький досвід.

   Unity дозволив реалізувати інтуїтивний інтерфейс, покрокову взаємодію, плавні анімації, чітку структуризацію блоків інформації. Це робить процес проходження тесту зрозумілим і візуально приємним.

6. Відсутність вимог до великих даних.

   Система не залежить від масивних датасетів, що є плюсом на ранніх етапах розвитку продукту, коли поведінкових даних користувачів ще недостатньо для побудови складних моделей машинного навчання.

До недоліків розробленої системи належать:

1. Обмежена точність при складних сценаріях.

   Оскільки система базується на експертних правилах, її ефективність залежить від якості вручну визначених ваг. У складних випадках, де поведінка користувача нестандартна, точність рекомендацій може знижуватися порівняно з алгоритмами машинного навчання, що аналізують великі масиви даних.

2. Відсутність глибокої персоналізації, основаної на поведінкових факторах.

   Coursera, DataCamp та інші великі платформи враховують десятки параметрів, таких як тривалість сесій, темп навчання, повернення до тем, показники складності завдань. Наявна система аналізує лише результати опитування, без використання багатовимірних поведінкових сигналів.

3. Відсутність автоматичного коригування рекомендацій під час навчання.

   Система видає рекомендацію одноразово після проходження тесту. Натомість аналогічні платформи коригують план навчання в реальному часі залежно від успішності студента.

4. Обмежена глибина навчальних планів.

   У наявній реалізації навчальний план містить основні модулі та теми, а також базові завдання. Платформи, як-от Coursera чи Codecademy, містять цілісні тренажери, домашні завдання, автоматичні перевірки, довгі навчальні треки.

5. Залежність від стабільності підключення до Firebase.

   Хоча Addressables дозволяють кешувати частину даних локально, робота тесту та збереження результатів залежить від доступності хмарної бази. При нестабільному інтернеті це може спричиняти затримки.

6. Відсутність соціальних та мотиваційних механізмів.

   На відміну від великих платформ, у системі немає бейджів, рейтингів, відстеження серій активності, гейміфікаційних елементів. Це може знижувати довгострокову залученість користувачів.

Проведений аналіз показав, що розроблена рекомендаційна система має низку вагомих переваг — передусім прозорість, простота інтеграції, адаптивність до різних рівнів користувача та легкість розширення. Водночас

наявні недоліки пов’язані зі спрощеною моделлю персоналізації, відсутністю поведінкової аналітики та обмеженою глибиною навчальних планів.

Урахування цих недоліків забезпечує основу для подальшого вдосконалення системи, зокрема шляхом впровадження методів машинного навчання, розширення бази модулів навчання та впровадження механізмів гейміфікації.

3.4 Аналіз ефективності рекомендацій і точності підбору навчальних планів

Аналіз ефективності роботи рекомендаційної системи проводився на основі результатів тестових сценаріїв, що моделювали різні типи користувачів: від початківців до досвідчених програмістів, а також користувачів із певними професійними цілями: веб-розробка, аналітика даних, створення ігор тощо. У цьому розділі наведено якісний та кількісний аналіз отриманих результатів, їх відповідність очікуванням та здатність системи формувати адекватні навчальні траєкторії.

Проведені тестові проходження показали, що система демонструє високу узгодженість у виборі мови програмування відповідно до профілю користувача.

За результатами аналізу:

- Для профілів початківців система стабільно рекомендувала Python, що є логічним, адже ця мова має низький поріг входу та широку сферу застосування.
- Для користувачів із досвідом у веб-технологіях система частіше обирала JavaScript, враховуючи релевантні відповіді щодо цілей та інтересів.
- Для користувачів, орієнтованих на мобільну розробку або високопродуктивні системи, система рекомендувала C# — згідно з ваговими коефіцієнтами та характеристиками мови.

- Для профілів, пов’язаних із аналітикою даних, основними рекомендаціями були Python або R, що узгоджується з практичними стандартами індустрії.

Більшість рекомендацій відповідала очікуваним результатам, сформованим на основі експертної матриці відповідностей мов програмування та цільових сценаріїв користувачів.

Під час аналізу повторних тестувань встановлено:

- рекомендації збігалися у 92–97% випадків при незначних варіаціях відповідей;
- зміни рекомендації виникали лише у випадках, коли користувач навмисно змінював ключові відповіді, що впливали на вибір мови, наприклад, ціль навчання;

Це свідчить про стабільність логіки формування рекомендацій.

Після визначення мови програмування система генерує навчальний план, що охоплює:

- вступні теоретичні модулі;
- базові поняття, наприклад, змінні, типи даних, цикли тощо;
- практичні вправи;
- розділи для продовження навчання на середньому рівні;

Результати аналізу структури навчальних планів:

1. Для початківців навчальні плани містили повний набір базових тем, структурованих у логічній послідовності: від основ до практичних завдань.
2. Для користувачів середнього рівня система додавала розширені модулі, такі як: ООП, робота з файлами, основи алгоритмів.
3. При орієнтації на специфічні сфери, навчальний план містив спеціалізовані розділи:
    - веб-розробка: HTML/CSS, основи JS, робота з DOM;
    - ігрова розробка: Unity, компоненти сцени, скрипти;

- аналітика: бібліотеки для роботи з даними, основи статистики;

Аналіз показав, що навчальні плани:

- не містять “порожніх” або невідповідних тем;
- побудовані з урахуванням зростання складності;
- коректно відображають характер обраної мови програмування;
- відповідають типу цілей користувача;

Усі навчальні траєкторії мали завершеність і внутрішню узгодженість.

У процесі аналізу було ідентифіковано низку ситуацій, які можуть призвести до менш точної рекомендації або неідеального навчального плану:

1. Нечіткі відповіді у профілі користувача

   У профілі користувача з широко сформульованими цілями, наприклад, “хочу все”, система обирала найбільш універсальні мови, що не завжди точно відображало реальні потреби.

2. Граничні випадки рівня досвіду

   Для профілів “між рівнями”, наприклад, між початковим і середнім рівнем, навчальні плани інколи пропонували базові теми, вже відомі користувачу, або недостатньо складні модулі.

3. Обмежена кількість мов у словнику рекомендацій

   У випадках, коли профіль користувача добре підходив під рідкісну або вузькоспеціалізовану мову програмування, система не могла запропонувати повністю оптимальний варіант через обмежений набір доступних мов.

Ці похибки не критичні й можуть бути усунені шляхом розширення набору мов, уточнення опитування, додавання адаптивної логіки після першого етапу навчання.

Проведений аналіз дозволяє зробити висновок, що система демонструє високу ефективність:

- рекомендації відповідають рівню користувача та його цілям;
- навчальні плани є структурованими, повними та логічними;

- система стабільно формує однакові результати при повторних проходженнях;
- похибки стосуються лише граничних або невизначених сценаріїв;

У цілому система забезпечує достатню точність і надійність для використання як інструмент для початку навчання програмуванню та вибору оптимальної навчальної траєкторії.

Висновок до розділу 3

У цьому розділі було проведено комплексне тестування розробленої рекомендаційної системи, зіставлення її функціональності з існуючими аналогами, визначення сильних і слабких сторін, а також аналіз ефективності отриманих рекомендацій і точності побудованих навчальних планів.

Методика тестування охоплювала функціональні, модульні, інтеграційні та навантажувальні випробування, що дозволило всебічно оцінити працездатність ключових модулів системи — від опитувальника до генератора навчальних планів. Результати тестування підтвердили стабільність роботи застосунку, узгодженість логіки формування рекомендацій та коректну взаємодію з хмарними сервісами Firebase.

Порівняння із сучасними освітніми платформами, такими як Coursera, Codecademy, Khan Academy та DataCamp, показало, що розроблена система реалізує основні принципи персоналізації, але робить це у більш компактному та прозорому вигляді. Хоча вона поступається масштабним сервісам у глибині аналізу поведінкових даних і кількості доступного контенту, її перевагою є чіткість роботи, простота інтеграції, зрозумілість алгоритмів та гнучкість у розширенні.

Аналіз ефективності рекомендацій засвідчив, що система здатна точно визначати релевантну мову програмування для різних категорій користувачів, а сформовані навчальні плани мають логічну структуру та відповідають рівню підготовки й освітнім цілям. Виявлені похибки стосуються переважно граничних

або неоднозначних випадків, що може бути усунено шляхом подальшого вдосконалення опитувальника та розширенням бази мов програмування.

Загалом результати розділу підтверджують, що розроблена рекомендаційна система є ефективним інструментом підтримки користувачів у виборі мови програмування та формуванні персоналізованої освітньої траєкторії. Вона має значний потенціал для подальшого розвитку, зокрема через інтеграцію методів машинного навчання, розширення навчальних модулів та впровадження додаткових сервісів, спрямованих на підвищення мотивації та залученості користувачів.

ВИСНОВКИ

У магістерській роботі було проведено комплексне дослідження процесів персоналізації навчання у сфері програмування та розроблено рекомендаційну систему, що визначає найбільш релевантну мову програмування для користувача та формує індивідуальний навчальний план на основі його рівня підготовки, цілей та інтересів.

У першому розділі виконано ґрунтовний огляд сучасних рекомендаційних та експертних систем, механізмів їх побудови та підходів до персоналізації освітнього контенту. Проаналізовано переваги та обмеження систем, що базуються на контентній фільтрації, колаборативних методах, експертних правилах і штучному інтелекті. Особливу увагу приділено освітнім платформам Coursera, Codecademy, Khan Academy та DataCamp, що є провідними світовими прикладами реалізації адаптивного навчання. Виконаний теоретичний аналіз дав змогу визначити вимоги до майбутньої системи та обґрунтувати вибір архітектурних і технологічних рішень.

У другому розділі розроблено архітектуру програмного продукту, описано структуру бази даних, логіку роботи ключових модулів та алгоритм формування рекомендацій. Для реалізації системи обрано середовище Unity, мову програмування C# та хмарну платформу Firebase, що забезпечує зберігання даних, авторизацію й динамічне завантаження контенту. Розроблено модулі управління користувачами, опрацювання запитань, генерації рекомендацій та створення індивідуальних навчальних планів. Сформовано структуру сутностей, що забезпечує гнучкість і можливість подальшого розширення системи.

У третьому розділі проведено тестування функціональності системи, порівняння її можливостей з існуючими аналогами, визначено переваги і недоліки, а також виконано аналіз ефективності рекомендацій. Результати тестування підтвердили стабільність роботи модулів, точність формування рекомендацій і логічність побудованих навчальних планів. Система демонструє високу узгодженість результатів, коректно адаптується до різних рівнів підготовки користувачів і здатна формувати повноцінні траєкторії навчання.

Виявлені недоліки стосуються переважно обмеженої кількості доступних мов програмування та відсутності глибинного аналізу поведінкових даних, що характерно для систем, заснованих на експертних правилах.

Узагальнюючи отримані результати, можна зробити висновок, що розроблена рекомендаційна система відповідає поставленій меті, забезпечує необхідний рівень персоналізації та може бути використана як інструмент підтримки початку навчання програмуванню. Система має низку переваг — прозорість логіки, адаптивність, гнучкість розширення, технологічну мобільність — що робить її ефективним рішенням для індивідуалізованої освіти.

Перспективи подальшого розвитку системи полягають у впровадженні алгоритмів машинного навчання для підвищення точності рекомендацій, розширенні бази мов програмування та навчальних модулів, а також у введенні елементів гейміфікації, здатних підвищити мотивацію користувачів. Важливим напрямом удосконалення є адаптація навчальних планів у реальному часі залежно від успішності та темпу засвоєння матеріалу, що дозволить створити більш динамічну та чутливу до індивідуальних особливостей траєкторію навчання. Окрім того, поглиблення аналітики поведінки користувачів відкриє можливості для формування довгострокових освітніх стратегій і забезпечить системі здатність враховувати широкий спектр факторів, які впливають на навчальний процес. Таким чином, виконана робота не лише вирішує актуальну задачу вибору мови програмування та побудови персоналізованого плану навчання, але й створює основу для подальшого розвитку масштабованої інтелектуальної навчальної платформи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Francesco Ricci, Lior Rokach and Bracha Shapira. Recommender Systems Handbook, Springer, 2022. 1012c.
2. G. Adomavičius, A. Tužylin. Towards the Next Generation of Recommender Systems: A Review of the Current State and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*. 2005. №6. С.734–749.
3. Charu K. Aggarwal. Recommendation Systems, Springer Cham, 2016. 518c. DOI: https://doi.org/10.1007/978-3-319-29659-3
4. Sutton R.S., Barto A.G. Reinforcement Learning, MIT Press, 2018. 548c.
5. Robin Burke. Hybrid Recommender Systems: Surveys and Experiments. *The Journal of Personalization Research*. 2002. №12. С.331–370. DOI: https://doi.org/10.1023/A:1021240730564
6. Zhang S., Yao L., Sun A., Tai Y. Deep Learning-Based Recommender System: A Survey and New Perspectives. *ACM Journals*. 2019. №5. С.1–38. DOI: https://doi.org/10.1145/3285029
7. He H., Liao L., Zhang H., Ni L., Hu H., Chua T. Neural Collaborative Filtering. *Open Journal of Applied Sciences*. 2017. №1. С.173–182. DOI: https://doi.org/10.48550/arXiv.1708.05031
8. Koren Y., Bell R., Wolinski C. Matrix Factorization Methods for Recommender Systems. *Computer*. 2009. №8. С.30–37. DOI: https://doi.org/10.1109/MC.2009.263
9. Journal of Personalized Research Issue Volume 12. User Modeling and User-Adapted Interaction, Springer, 2002. 104c.
10. Cambria E., White B. Jumping the NLP Curves: A Review of Natural Language Processing Research. *IEEE Journal of Computational Intelligence*. 2014. №2. С.48–57. DOI: https://doi.org/10.1109/MCI.2014.2307227
11. Zhang S., Yao L., Sun A., Tai Y. Deep Learning-Based Recommender System. ACM Computing Surveys. *Artificial Intelligence Review*. 2021. №8. С.1–38.
12. Liu K., Zhang Z., Huang Z. A Comprehensive Survey of Recommender Systems Based on Deep Learning. *Applied Sciences*. 2023. №20. С.13–20. DOI: https://doi.org/10.3390/app132011378
13. UNESCO. Education: From Destruction to Restoration, 2022. URL: https://www.unesco.org/en/covid-19/education-disruption-recovery (Дата звернення: 15.10.2025)

14. OECD: Digital Education Outlook 2023 Towards an Effective Digital Education Ecosystem, OEDC, 2023. 414c. DOI: https://doi.org/10.1787/c74f03de-en
15. Ilie Gligorea, Marius Cioca, Romana Oancea, Andra-Teodora Gorski, Hortensia Gorski and Paul Tudorache. Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review. *Education Sciences*. 2023. №12. 28c. DOI: https://doi.org/10.3390/educsci13121216
16. Coursera Engineering Adaptive Learning Platforms: How AI Powers Personalized Education, Coursera, 2025. URL: https://www.coursera.org/articles/adaptive-learning-platforms (Дата звернення: 16.10.2025)
17. Coursera Announcing AI-powered capabilities enabling educators to use Coursera Coach to deliver interactive, personalized instruction, Coursera, 2024. URL: https://blog.coursera.org/announcing-ai-powered-capabilities-enabling-educators-to-use-coursera-coach-to-deliver-interactive-personalized-instruction/ (Дата звернення: 16.10.2025)
18. Coursera Blog. From Start to Finish: How Early Course Design Can Drive Online Learner Completion, Coursera, 2024. URL: https://blog.coursera.org/from-start-to-finish-how-early-course-design-can-drive-online-learner-completion/ (Дата звернення: 16.10.2025)
19. Codecademy. AI Features available on Codecademy, 2023. URL: https://help.codecademy.com/hc/en-us/articles/23400751016859-AI-Features-available-on-Codecademy (Дата звернення: 16.10.2025)
20. OpenAI and Codecademy. How We're Harnessing GPT-4o in Our Courses, 2023. URL: https://www.codecademy.com/resources/blog/gpt4o-ai-learning-assistant (Дата звернення: 17.10.2025)
21. DataCamp. Why we use IRT at DataCamp, 2023. URL: https://www.datacamp.com/blog/why-we-use-irt-at-data-camp (Дата звернення: 17.10.2025)
22. DataCamp Announcing the new DataCamp AI Assistant, 2023. URL: https://www.datacamp.com/blog/announcing-the-new-data-camp-ai-assistant-announcing-the-new-data-camp-ai-assistant (Дата звернення: 17.10.2025)
23. Khan Academy Help Center. How do Khan Academy's Mastery levels work?, 2023. URL: https://support.khanacademy.org/hc/en-us/articles/5548760867853--How-do-Khan-Academy-s-Mastery-levels-work (Дата звернення: 17.10.2025)
24. Jordan M., Mitchell T. Machine Learning: Trends, Perspectives, and Prospects. *Science*. 2015. №6245. С.255–260. DOI: https://doi.org/10.1126/science.aaa8415

25. I Goodfellow, Y Bengio, A Courville, Y Bengio Deep Learning, MIT, 2016. 802c.
26. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th Edition, 2021. 1166c.
27. Zhang Y., Chen H. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends in Information Retrieval*. 2020. №1. С.1–101. DOI: https://doi.org/10.1561/1500000066
28. EdTech Le Ying Tan, Shiyu Hu, Darren J. Yeo, Kang Hao Cheong. Artificial intelligence-enabled adaptive learning platforms: A review. *Computers and Education: Artificial Intelligence*. 2025. №1. С.1–12. DOI: https://doi.org/10.1016/j.caeai.2025.100429
29. Stack Overflow The State of Online Programming Education 2023, Stack Overflow, 2023. URL: https://survey.stackoverflow.co/2023/ (Дата звернення: 09.11.2025)
30. Ahmet Arnavut, Hüseyin Bicen, Cahit Nuri. Students' Approaches to Massive Open Online Courses: The Case of Khan Academy. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*. 2019. №1. С.82–90. DOI: https://doi.org/10.70594/brain/v10.i1/8
31. Khan Academy Blog. Introducing… Khanmigo!, 2023. URL: https://support.khanacademy.org/hc/en-us/community/posts/13992414612877-Introducing-Khanmigo (Дата звернення: 09.11.2025)
32. Anthropic partnered with the Collective. Collective Constitutional AI: Aligning a Language Model with Public Input, 2023. URL: https://www.anthropic.com/news/collective-constitutional-ai-aligning-a-language-model-with-public-input (Дата звернення: 10.11.2025)
33. OpenAI Using ChatGPT in Education: Opportunities and Challenges, OpenAI, 2023. URL: https://www.coursesidekick.com/psychology/29047304 (Дата звернення: 10.11.2025)
34. Google. Gemini Models for Learning and Reasoning, Google DeepMind. URL: https://deepmind.google/about/ (Дата звернення: 10.11.2025)
35. Nishat Raihan, Mohammed Latif Siddiq, Joanna C.S. Santos, Marcos Zampieri. Large Language Models in Computer Science Education: A Systematic Literature Review. *SIGCSETS 2025: Proceedings of the 56th ACM Technical Symposium on Computer Science Education*. 2025. С.938–944. DOI: https://doi.org/10.48550/arXiv.2410.16349
36. IEEE Access Conversational Agents for Teaching Programming Skills, IEEE, 2024.
37. Антонов Ю. С. Оцінка повноти відповіді на питання відкритого типу без дотримання суворої послідовності слів. *Наука і техніка сьогодні*. 2025 №7. С.1243–1257. URL: http://perspectives.pp.ua/index.php/nts/article/view/26931 https://doi.org/10.52058/2786-6025-2025-7(48)-1243-1257

38. Антонов Ю.С. Оцінка повноти відповідей в автоматизованих системах контролю знань. Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка, 2012. С.113–117. URL: https://ea.donntu.edu.ua/bitstream/123456789/24273/1/p113.pdf (Дата звернення: 13.11.2025)
39. Кузьма К.Т. Інформаційна технологія перевірки відповідей в інтелектуальній автоматизованій системі контролю знань. *Вісник Вінницького політехнічного інституту Вип.4*. 2020. №7. С.58–66. DOI: https://doi.org/10.31649/1997-9266-2020-151-4-58-66
40. Антонов Ю. С., Космінська О. М. Методика аналізу тестових завдань на основі отриманих результатів тестування. Інформаційні технології і засоби навчання. 2009. № 4. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/81/67 (Дата звернення: 13.11.2025)
41. Fowler M. Enterprise Application Architecture Patterns, Addison-Wesley, 2002. 442с.
42. Martin R. Clean Architecture: A Master's Guide to Software Structure and Design, Prentice Hall, 2017. 352с.
43. Freeman E., Robson E. Head First Design Patterns, O'Reilly, 2020. 669с.
44. Unity Technologies. Unity User Interface (UI) Documentation. 2023, URL: https://docs.unity3d.com/2022.2/Documentation/Manual/UIToolkits.html (Дата звернення: 17.11.2025)
45. Unity Technologies. Unity Guide: A Toolkit for Architecture and User Interface, 2023, URL: https://docs.unity3d.com/2022.2/Documentation/Manual/UIElements.html (Дата звернення: 18.11.2025)
46. Albahari J., Albahari B. C# 10 in a Nutshell, O'Reilly Media, 2022. 1058с.
47. Troelsen A. Pro C# and .NET 8, Apress, 2024. 1680с.
48. Google. Firebase Documentation: Firestore, Authentication, Analytics, 2024, URL: https://firebase.google.com/docs/firestore (Дата звернення: 18.11.2025)
49. Kleppmann M. Designing Data-Intensive Applications, O'Reilly, 2017. 616с.
50. Brusilovsky P., Millan E. User Models for Adaptive Hypermedia and Adaptive Educational Systems, The Adaptive Web, 2007. С.3–53.
51. Mayer R. The Cambridge Handbook of Multimedia Learning, Cambridge University Press, 2014. 99с. DOI: https://doi.org/10.1017/CBO9781139547369