Кам'янець-Подільський національний університет імені Івана Огієнка

Твое успішне майбутнє

Фізико-математичний факультет

ЗБІРНИК

матеріалів наукової конференції
за підсумками науково-дослідної роботи
здобувачів вищої освіти
фізико-математичного факультету
у 2025-2026 н. р.

8-9 квітня 2026 року

ЕЛЕКТРОННЕ ВИДАННЯ

Кам'янець-Подільський
2026

УДК 51+53+004
ББК 22.1+22.3+32.97
З41

Рекомендовано вченою радою фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка,
протокол № 5 від 26 травня 2026 року.

Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка у 2025-2026 н.р., 8-9 квітня 2026 року [Електронний ресурс]. Кам'янець-Подільський : Кам'янець-Подільський національний університет імені Івана Огієнка, фізико-математичний факультет, 2026. 74 с.

Електронна версія збірника доступна за покликанням:
URL: https://fizmat.kpnu.edu.ua/studentski-konferentsii/

З М І С Т

Андрій АФРАМЧУК
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Тетяна ПИЛИПЮК**
кандидат фізико-математичних наук, доцент

РОЗРОБКА 2D-ГРИ НА ПЛАТФОРМІ UNITY З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Розглянуто створення двовимірної гри з використанням ігрового рушія Unity. Особливу увагу приділено інтеграції алгоритмів штучного інтелекту для керування поведінкою неігрових персонажів (NPC). Проаналізовано методику побудови скінченних автоматів та алгоритмів пошуку шляху.

Ключові слова: Unity, 2D-гра, штучний інтелект, скінченні автомати, NavMesh, ігрова логіка.

Актуальність теми. Сучасна індустрія відеоігор вимагає від розробників створення все більш складних і реалістичних сценаріїв взаємодії. Використання елементів штучного інтелекту (ШІ) у 2D-іграх дозволяє значно підвищити реіграбельність та залученість користувача, створюючи динамічне середовище, що реагує на дії гравця.

Мета публікації. Дослідити технічні аспекти розробки 2D-гри на базі Unity та реалізувати систему адаптивної поведінки ворогів за допомогою методів ШІ.

Виклад основного матеріалу дослідження. Процес розробки розділено на три етапи: проєктування архітектури, створення графічних активів та програмування логіки ШІ. Для реалізації інтелектуальної поведінки ворогів використано модель скінченних автоматів (Finite State Machine).

Для розробки 2D-гри обрано середовище Unity, що зумовлено його високою оптимізацією для кросплатформних проєктів та потужним інструментарієм для роботи з 2D-фізикою. Використання Tilemap дозволяє ефективно створювати ігрові рівні, а система CineMachine забезпечує плавне керування камерою. Основною мовою програмування обрано C#, що дає змогу гнучко реалізовувати алгоритми штучного інтелекту через об'єктно-орієнтований підхід [1].

Для побудови маршрутів пересування адаптовано компонент NavMeshPlus, що дозволяє реалізувати динамічний пошук шляху в двовимірному просторі. Алгоритм розраховує оптимальну траєкторію, враховуючи перешкоди (Tilemap Colliders), що забезпечує природність рухів персонажів.

Інтелектуальна поведінка ворогів у проєкті базується на дворівневій структурі [2]:

1 Скінченні автомати (FSM), де кожен NPC має набір станів: Idle (очікування), Patrol (патрулювання), Chase (переслідування) та Attack (атака). Перехід між станами відбувається на основі тригерів (наприклад, потрапляння гравця в радіус видимості або критичне зниження рівня здоров’я ворога).
2 Сенсорна система, яка реалізована за допомогою методів Raycasting та OverlapCircle. Це дозволяє імітувати «зір» та «слух» персонажа, уникаючи проходження крізь стіни та реагуючи на шум, створений гравцем.

Стандартна система NavMesh у Unity орієнтована на 3D, тому для 2D-гри було інтегровано компоненти, що дозволяють запікати навігаційну сітку на основі спрайтів та тайлсетів.

Динамічні перешкоди, де завдяки використанню NavMeshObstacle, вороги здатні оминати не лише статичні об’єкти (стіни, дерева), а й рухомі перешкоди, робить їхню поведінку більш природною.

Алгоритм A (A-star) забезпечує знаходження найкоротшого шляху до цілі, мінімізуючи обчислювальні витрати, що є критичним для мобільних платформ.

Окрім ШІ ворогів, розроблено систему взаємодії з оточенням. Впроваджено класи-менеджери (GameManager, UIManager), які контролюють життєвий цикл рівня, підрахунок балів та збереження прогресу. Для обробки подій використано архітектуру Unity Events, що дозволяє декуплеризувати (роз’єднати) компоненти гри та підвищити стабільність коду.

Керування станами NPC (патрулювання, переслідування, атака) реалізовано через патерн «State», що дозволяє легко масштабувати систему та додавати нові типи поведінки без зміни основного коду контролера.

Висновки. Розроблена система демонструє високу ефективність у межах 2D-платформеру. Використання скінченних автоматів у поєднанні з алгоритмами пошуку шляху дозволяє створювати автономних агентів, здатних до базового тактичного аналізу ігрової ситуації. Подальші дослідження будуть спрямовані на впровадження нейронних мереж для навчання агентів безпосередньо під час ігрового процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Бондарчук Ю. П. Розробка ігрових застосунків на платформі Unity : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2021. 156 с.

2. Thorn A. Unity 2D Game Development. Birmingham : Packt Publishing, 2024. 312 p.

Олександр БЄКТЄРЄВ
здобувач вищої освіти першого (бакалаврського) рівня 2 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

АЛГОРИТМІЧНІ ПІДХОДИ ДО АВТОМАТИЗАЦІЇ ГЕНЕРАЦІЇ АНАЛІТИЧНИХ ДАШБОРДІВ НА БАЗІ GOOGLE SHEETS ТА GOOGLE APPS SCRIPT

У роботі розглянуто алгоритмічні методи автоматизації побудови аналітичних звітів у середовищі Google Sheets за допомогою скриптової мови Google Apps Script. Визначено ключові підходи до оптимізації обробки даних (Batch Operations) та динамічного керування об'єктами візуалізації для підвищення ефективності бізнес-процесів.

Ключові слова: автоматизація, Google Apps Script, дашборд, алгоритми обробки даних, Google Sheets API.

Актуальність теми. Сучасний стан розвитку інформаційних технологій характеризується експоненціальним зростанням обсягів даних, що потребують оперативної обробки та візуалізації. Для малого та середнього бізнесу, а також освітніх і наукових проектів, використання дорогих BI-платформ (таких як Power BI або Tableau) не завжди є доцільним. У цьому контексті Google Sheets у

поєднанні з хмарною платформою розробки Google Apps Script (GAS) стає потужним інструментом для створення кастомних аналітичних рішень. Автоматизація генерації дашбордів дозволяє виключити людський фактор, зменшити кількість рутинних операцій та забезпечити прийняття рішень на основі актуальних даних у реальному часі.

Мета публікації. полягає у теоретичному обґрунтуванні та практичному описі алгоритмічних підходів до автоматизованого створення аналітичних панелей, що забезпечують високу швидкість обробки даних при мінімальних технічних витратах.

Виклад основного матеріалу дослідження. Проектування автоматизованого дашборду вимагає системного підходу, який можна розділити на три ключові алгоритмічні модулі: збір та очищення даних, аналітична трансформація та програмна візуалізація.

1. **Алгоритми оптимізації взаємодії з даними (I/O Operations).** Найбільшою технічною перешкодою при роботі з Google Apps Script є ліміти на час виконання скрипта (Execution Timeout) та кількість запитів до сервісів Google. Класичний підхід, що передбачає циклічне зчитування кожної клітинки окремо, має обчислювальну складність O(n^2) і швидко призводить до вичерпання квот.

Ефективним алгоритмічним рішенням є використання методу «Batch Reading». Суть підходу полягає у зчитуванні всього діапазону даних у двовимірний масив JavaScript за один виклик getValues(). Подальша обробка відбувається в оперативній пам'яті скрипта, що працює в тисячі разів швидше, ніж пряме звернення до таблиці. Після завершення обробки результат записується одним блоком за допомогою setValues().

2. **Аналітичне групування та трансформація.** Для генерації показників дашборду (KPI) необхідно реалізувати алгоритми агрегації. Замість використання стандартних формул Google Sheets, які можуть сповільнювати інтерфейс таблиці при великих обсягах даних, доцільно застосовувати хеш-таблиці (об’єкти Map у JavaScript).

Алгоритм групування виглядає так: скрипт ітерує по масиву даних, створюючи унікальні ключі (наприклад, поєднання назви регіону та місяця) і

накопичуючи відповідні значення в об'єкті-акумуляторі. Такий підхід забезпечує лінійну складність O(n), що є критично важливим для стабільної роботи системи.

3. **Алгоритмічна візуалізація через Charts Service.** Автоматизація створення діаграм базується на класі EmbeddedChartBuilder. Програмний підхід дозволяє реалізувати «розумне» масштабування:

Алгоритм динамічного визначення меж даних: використання методу getLastRow() дозволяє скрипту самостійно адаптувати діапазон діаграми при додаванні нових записів.

Алгоритм сіткового розміщення (Grid Layout): математичний розрахунок координат для кожної діаграми, що запобігає їх накладанню та забезпечує естетичний вигляд панелі на різних пристроях.

4. **Автономність та тригерна логіка.** Для повної автоматизації застосовується подійно-орієнтований підхід. Використання часових тригерів (Time-driven triggers) дозволяє системі щогодини або щодня самостійно звертатися до зовнішніх джерел даних (наприклад, через API CRM-систем), оновлювати внутрішні таблиці та перераховувати всі показники дашборду без участі людини.

Висновки. Розробка аналітичних дашбордів на базі Google Apps Script є ефективною альтернативою складним BI-системам. Використання алгоритмів пакетної обробки даних та хеш-мап для агрегації дозволяє обійти технічні обмеження хмарної платформи. Проведений аналіз доводить, що автоматизація звітності не лише скорочує час на обробку інформації, але й створює надійну базу для оперативного управління проектами в умовах динамічного середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Документація Google Apps Script [Електронний ресурс]. – Довідкові матеріали з сервісів Google Apps Script. – 2026. – Режим доступу: https://developers.google.com/apps-script/reference (дата звернення: 26.03.2026).

2. Офіційний блог Google Cloud. Найкращі практики для Google Apps Script [Електронний ресурс]. – Режим доступу: https://blog.google/products/google-sheets/ (дата звернення: 27.03.2026).

3. Google Charts API [Електронний ресурс]. – Сервіс візуалізації та побудови діаграм. – Режим доступу: https://developers.google.com/chart (дата звернення: 27.03.2026).

Ілля БІЛОУС
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Марина МЯСТКОВСЬКА**
кандидат педагогічних наук

ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВЗАЄМОДІЇ ІГРОВОГО СЕРЕДОВИЩА UNITY WEBGL ТА ХМАРНИХ СЕРВІСІВ FIREBASE В ОСВІТНІХ ВЕБЗАСТОСУНКАХ

У роботі розглянуто процес створення навчального веб-додатку «MathMaze», розробленого на базі Unity та інтегрованого з платформою Firebase. Додаток поєднує ігрові механіки лабіринту з навчальними математичними задачами, забезпечуючи взаємодію між вчителем та учнем через хмарну базу даних. Особливу увагу приділено архітектурі даних для збереження ігрового прогресу та механізмам синхронізації веб-інтерфейсу з ігровим рушієм.

Ключові слова: Unity, Firebase Cloud Firestore, GameBridge, JavaScript, математична освіта, ігрофікація.

Актуальність теми. Традиційні методи навчання математики часто стикаються з проблемою низької залученості учнів. Гейміфікація освітнього процесу через використання симуляторів дозволяє перетворити розв'язання задач на інтерактивний досвід. Створення інструменту, який поєднує візуальну привабливість відеогри та строгість навчальної програми, є важливим кроком у розвитку цифрової освіти.

Метою публікації є опис технічної реалізації інтеграції ігрового рушія Unity з сервісами Firebase для створення адаптивного навчального середовища.

Виклад основного матеріалу дослідження. Проєкт реалізовано як веб-застосунок, де клієнтська частина розділена на два модулі: адміністративно-

інформаційний (HTML/JS) та ігровий (Unity WebGL). Для забезпечення цілісності даних обрано платформу Firebase, зокрема сервіс Cloud Firestore.

Структура бази даних розроблена з урахуванням гнучкості налаштувань та має наступний вигляд:

- **global_config** (колекція загальних налаштувань):
  - *document game_levels*: масиви даних для математичних тем (питання, відповіді, ліміти часу);
  - *document shop*: опис доступних бустерів, їх вартість та характеристики;
- **teacher_configs** (колекція вчителя): містить персоналізовані набори завдань, що мають пріоритет над глобальними;
- **users** (колекція профілів учнів):
  - *profile*: особисті дані та інвентар;
  - *progress*: підколекція з полями maxAllowedLevel (прогрес) та isBlocked (статус доступу).

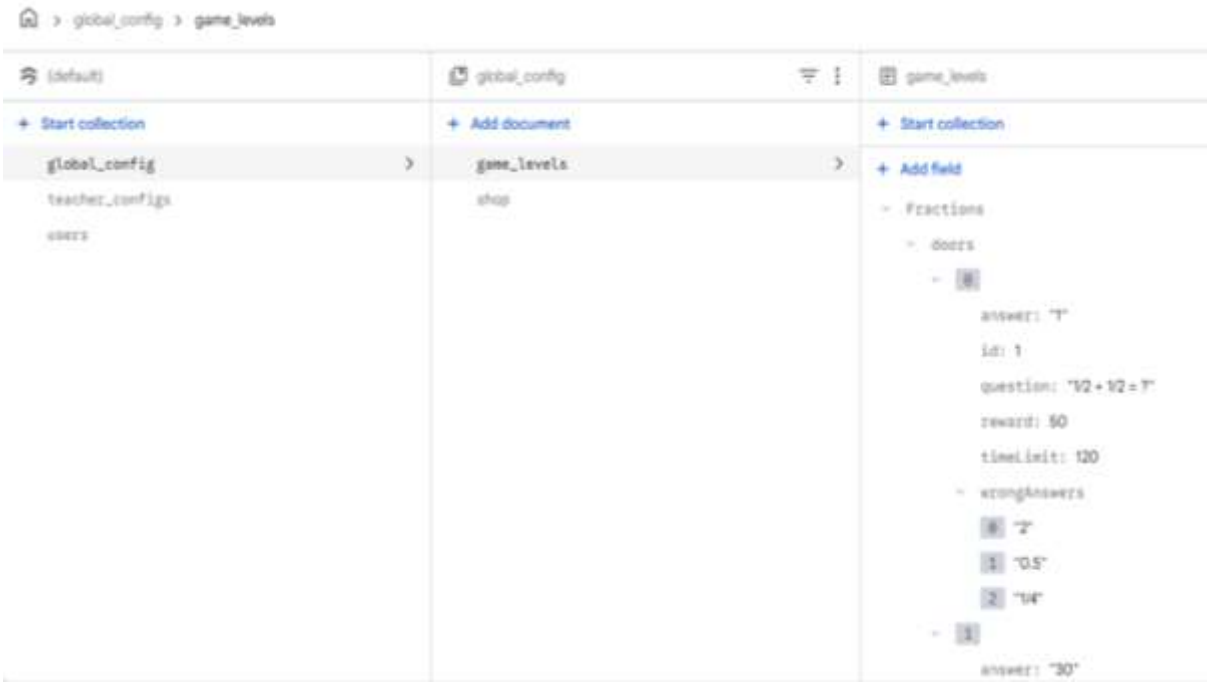


***Рис. 1**. Структура ієрархічної бази даних Cloud Firestore для зберігання ігрових конфігурацій та прогресу учнів*

Важливим технічним рішенням стала реалізація модуля GameBridge на мові JavaScript. Оскільки Unity WebGL працює в ізольованому контейнері (iframe), модуль виступає посередником:

1. При виборі рівня Unity надсилає запит через window.parent.postMessage.
2. JS-скрипт зчитує дані з Firebase.
3. Через метод unityInstance.SendMessage передається JSON-пакет, який десеріалізується всередині гри для побудови рівня.

Функціональні можливості системи:

- Синхронізація прогресу: автоматичне оновлення maxAllowedLevel після сигналу LEVEL_COMPLETE.
- Магазин та бустери: перевірка інвентарю в реальному часі перед стартом ігрової сесії.
- Динамічне керування: можливість вчителя змінювати параметри в Firestore без перезбірки проєкту.

Висновки. Інтеграція Unity з Cloud Firestore дозволила створити систему, де ігровий процес повністю керований зовнішніми даними. Використання JavaScript як "міст" між хмарою та ігровим рушієм забезпечує швидкість розробки та легкість масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Firebase Cloud Firestore Documentation. URL: https://firebase.google.com/docs/firestore (дата звернення: 03.02.2026)
2. Unity WebGL: Interacting with browser scripting. URL: https://docs.unity3d.com/Manual/webgl-interactingwithbrowserscripting.html (дата звернення: 07.02.2026)
3. Unity Scripting API: Application.ExternalEval. URL: https://docs.unity3d.com/ScriptReference/Application.ExternalEval.html (дата звернення: 09.02.2026)
4. JSON Data Serialization in Unity. URL: https://docs.unity3d.com/Manual/JSONSerialization.html (дата звернення: 11.02.2026)
5. Гейміфікація в освіті: як ігри допомагають дітям навчатися. URL: https://osvitoria.media/experience/gejmifikatsiya-v-osviti-yak-igry-dopomagayut-dityam-navchatysya/ (дата звернення: 16.02.2026)
6. Firebase як бекенд для будь-яких застосунків, та як використовувати Firebase-сервіси. URL: https://dou.ua/forums/topic/44058 (дата звернення: 21.02.2026)

Максим БІЛОУС
здобувач вищої освіти другого (магістерського) рівня 1 року навчання спеціальності F3 Комп'ютерні науки
Артем ПОЛІТОВ
здобувач вищої освіти першого (бакалаврського) рівня 1 року навчання спеціальності F3 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ПРОГРАМНЕ ЗВЕДЕННЯ ГЕТЕРОГЕННИХ ДАНИХ ВИБОРУ ВИБІРКОВИХ ДИСЦИПЛІН У ДИНАМІЧНІ АНАЛІТИЧНІ ДАШБОРДИ ЗА ДОПОМОГОЮ GOOGLE SHEETS API

У статті розглядається проблема обробки великих гетерогенних масивів даних, отриманих під час кампанії з вибору студентами вибіркових дисциплін. Описано практичний досвід розробки скриптового рішення на базі Google Apps Script, яке забезпечує автоматичну нормалізацію, багаторівневу сегментацію та валідацію даних. Доведено ефективність алгоритму у виявленні помилкових записів та формуванні структурованих відомостей, що критично пришвидшує адміністративні процеси в закладах вищої освіти.

Ключові слова: автоматизація процесів, Google Apps Script, валідація даних, нормалізація даних, регулярні вирази, електронне управління ЗВО.

Актуальність теми. Реалізація права студентів на формування власної освітньої траєкторії шляхом вибору навчальних дисциплін є обов'язковою умовою функціонування сучасних закладів вищої освіти. На практиці збір заяв від студентів найчастіше здійснюється за допомогою хмарних інструментів, таких як Google Forms. Однак такий підхід генерує масштабні масиви сирих, неструктурованих даних. Традиційні методи обробки табличних даних за допомогою вбудованих функцій електронних таблиць часто виявляються недостатніми через складність багаторівневого сортування та неможливість ефективно обробляти текстові виключення без втручання оператора.

Головною проблемою є не лише гетерогенність інформації (різні курси, факультети, форми навчання), але й високий відсоток користувацьких помилок: некоректне введення шифрів груп, вибір неправильної кількості дисциплін тощо. Ручна перевірка та сортування таких масивів вимагає значних часових витрат від

працівників навчального відділу. Тому розробка програмних інструментів, здатних не лише консолідувати, а й автоматично валідувати такі дані, є актуальним завданням для оптимізації освітнього процесу.

Мета публікації. Опис розробленого програмного алгоритму для автоматичної нормалізації, багаторівневого сортування та перевірки на коректність масивів студентських опитувань із використанням середовища Google Apps Script.

Виклад основного матеріалу дослідження. Для розв'язання поставленої задачі було розроблено скрипт на платформі Google Apps Script, який взаємодіє з об'єктною моделлю Google Sheets. Алгоритм програми побудований на принципах обробки даних у пам'яті (in-memory processing) з подальшою пакетною генерацією вихідних документів. Роботу програмного комплексу можна розділити на чотири ключові етапи.

Перший етап охоплює парсинг та нормалізацію даних. Скрипт зчитує первинні дані у двовимірний масив. Для усунення проблеми людського фактора під час заповнення форм застосовується програмна нормалізація: видалення зайвих пробілів та використання регулярних виразів. Зокрема, розроблено окрему функцію для стандартизації шифрів академічних груп, а також застосовано складний регулярний вираз для коректного розділення переліку обраних дисциплін, запобігаючи розриву назв, що містять внутрішні коми.

Другим етапом є валідація та обробка виключень. Важливою функцією скрипта є автоматичний аудит коректності вибору. Алгоритм перевіряє, чи обрав студент строго регламентовану кількість дисциплін. У разі виявлення відхилень запис маркується як помилковий, оригінальний рядок у базі підсвічується, а самі дані автоматично дублюються на окремо згенерований аркуш «Некоректний вибір» для подальшого ручного опрацювання адміністратором.

На третьому етапі відбувається багаторівнева сегментація. Програма здійснює обхід масиву, формуючи складну ієрархічну структуру даних. Дані групуються за трьома рівнями: факультет та форма навчання з автоматичним маркуванням студентів заочної форми, академічна група, а також обрана дисципліна.

Четвертий етап полягає у динамічній генерації структурованих звітів. Скрипт створює або оновлює робочі аркуші для кожного структурного підрозділу. Виведення даних здійснюється за колонковою системою, де кожен стовпець представляє окрему академічну групу. Для покращення візуального сприйняття застосовано автоматичне кольорове кодування клітинок та автоматичне масштабування інтерфейсу.

Висновки. У результаті дослідження було розроблено та впроваджено комплексне скриптове рішення для автоматизації рутинних завдань навчального відділу. Застосування Google Apps Script дозволило створити ефективний інструмент, який не лише сортує гетерогенні дані, але й виступає системою контролю якості: автоматично нормалізує формати вводу та ізолює помилкові записи. Впровадження алгоритму дозволило звести час обробки масиву обсягом близько 1500 записів з 20+ годин ручної роботи до 2-3 хвилин машинного часу, усунувши потребу в ручному пошуку помилок. Розроблена архітектура з використанням регулярних виразів та багаторівневої сегментації об'єктів доводить високу ефективність програмного підходу в управлінні даними закладу вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Семигіна, Т.В. & Новак, А.С. Інформаційні системи для вибіркових дисциплін у закладах вищої освіти. Sherman Oaks, California : GS Publishing Services, 2023.

2. Качурівський, В.О. & Качурівська, Г.М. Моделювання інформаційно-документальної системи презентації освітніх компонент освітньої програми. Центральноукраїнський науковий вісник. Технічні науки, 2022.

3. Google Sheets. URL: https://docs.google.com/spreadsheets/u/0/

Вадим ВАЛЬЦЕР
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
завідувач, доцент кафедри комп'ютерних наук

ЗАСТОСУВАННЯ RAG-АРХІТЕКТУРИ ТА СПЕЦІАЛІЗОВАНИХ ШІ-АГЕНТІВ ДЛЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДТРИМКИ В ХМАРНИХ ОСВІТНІХ СЕРЕДОВИЩАХ

У статті розглянуто процес створення інтерактивного курсу з використанням Google Classroom та мультиагентних ШІ-систем (NotebookLM, Gemini Gems). Розроблене середовище забезпечує персоналізовану підтримку за допомогою методу Сократа, оптимізуючи засвоєння матеріалу без порушення академічної доброчесності.

Ключові слова: штучний інтелект, Google Classroom, NotebookLM, Gemini Gems, мультиагентні системи, метод Сократа, RAG-архітектура.

Актуальність теми. Традиційні системи управління навчанням (LMS) часто є пасивними файловими сховищами, залишаючи здобувачів сам на сам зі складними завданнями. Розвиток генеративного ШІ відкриває можливості для створення віртуальних тьюторів, проте відкриті моделі схильні до «галюцинацій» або прямого генерування коду для списування. Тому розробка закритої екосистеми з вузькоспеціалізованими ШІ-агентами, які оперують виключно матеріалами курсу, є надзвичайно актуальною.

Мета публікації. Описати досвід розробки інтелектуального курсу на базі Google Workspace із застосуванням спеціалізованих ШІ-асистентів.

Виклад основного матеріалу дослідження. В якості основи для розробки було обрано навчальну дисципліну «Математичне моделювання». Замість створення монолітного програмного продукту, було застосовано модульний підхід. Базою даних виступає Google Drive, де структуровано зберігаються всі PDF-файли курсу (силабус, лекції, лабораторні роботи тощо), а інтерфейсом слугує Google Classroom. Головною інновацією проєкту є інтеграція у структуру курсу двох рівнів ШІ-підтримки на базі архітектури RAG (Retrieval-Augmented Generation).

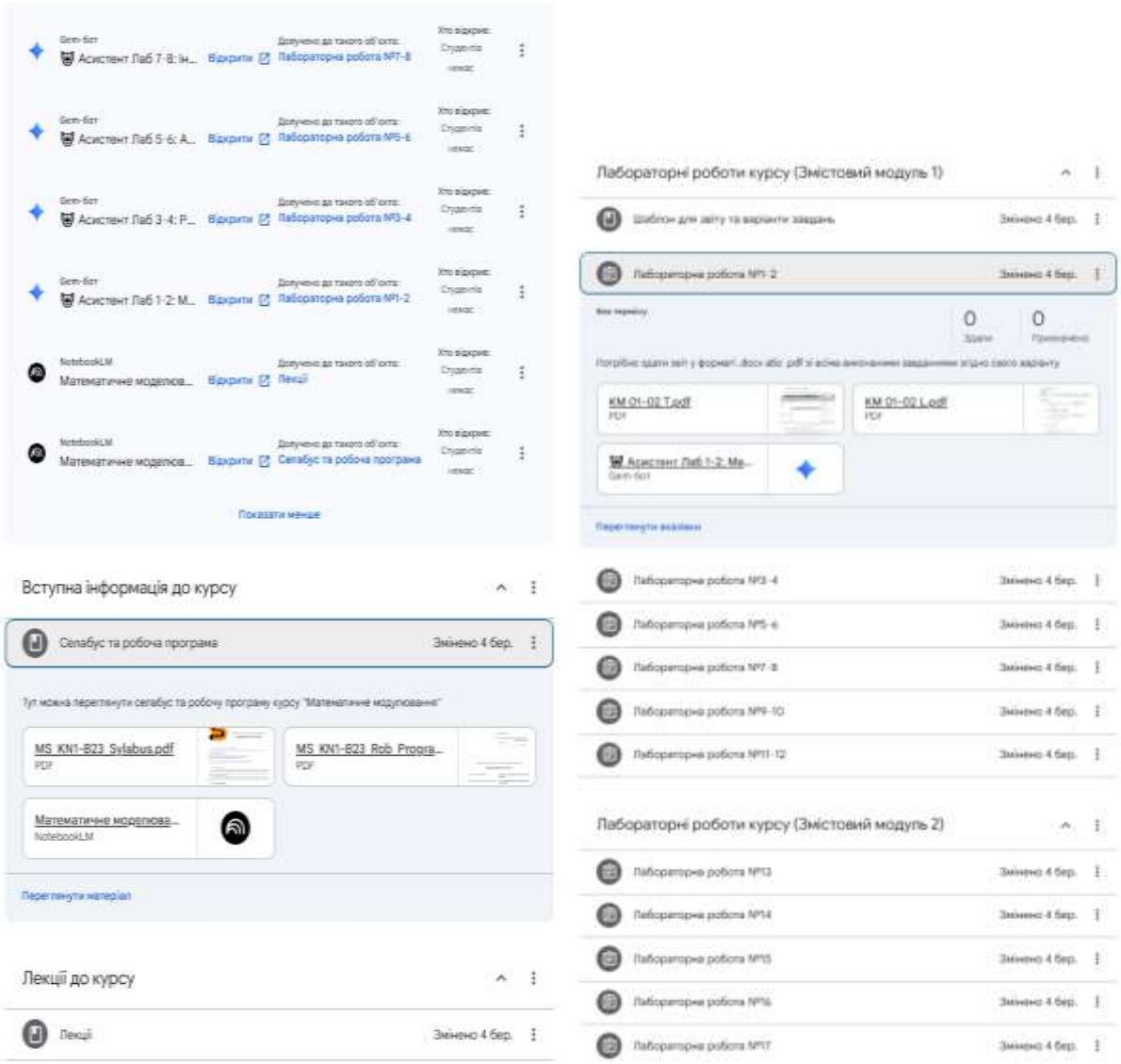


Рис. 1. *Зовнішній вигляд курсу*

Перший рівень – теоретичний супровід за допомогою Google NotebookLM. До вступного блоку та лекцій у Classroom було підключено спеціальні блокноти NotebookLM, у які завантажено всі вступні та лекційні матеріали. Цей інструмент дозволяє студентам ознайомитися з курсом перед початком його проходження, знайти необхідну інформацію у лекціях або переказати її простими словами, проходити тести та флеш-картки, слухати згенеровані аудіо-подкасти та дивитися згенеровані відео-перекази. Оскільки NotebookLM шукає інформацію виключно в завантажених PDF-документах, ризик генерування сторонньої чи хибної інформації зведено до нуля.

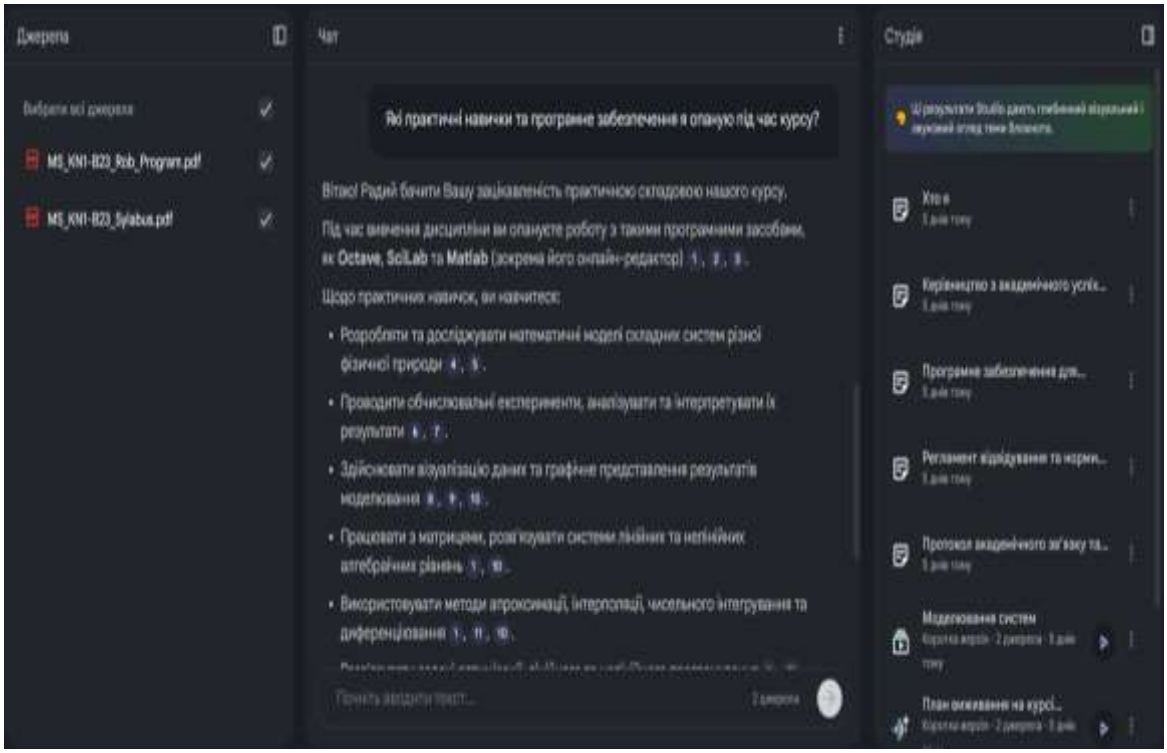


Рис. 2. *Приклад вигляду та роботи блокноту у NotebookLM*

Другий рівень – практичний супровід за допомогою мультиагентної системи на базі Gemini Gems. Для кожної лабораторної роботи було створено окремого ШІ-асистента (Gem-бота). Усього розроблено комплекс із 14 вузькоспеціалізованих ботів, кожен з яких отримав індивідуальні системні інструкції. Ключовою особливістю розроблених ботів є використання методу Сократа. У системні налаштування кожного ШІ-агента вбудовано жорстке обмеження: ботам категорично заборонено генерувати готовий до запуску програмний код або збирати схеми для студентів. Якщо студент просить розв'язати його завдання, система відмовляє і натомість пропонує допомогу в пошуку помилки у вже написаному коді студента, або ж ставить навідне запитання чи показує загальний приклад, спонукаючи здобувача самостійно дійти до правильного рішення.

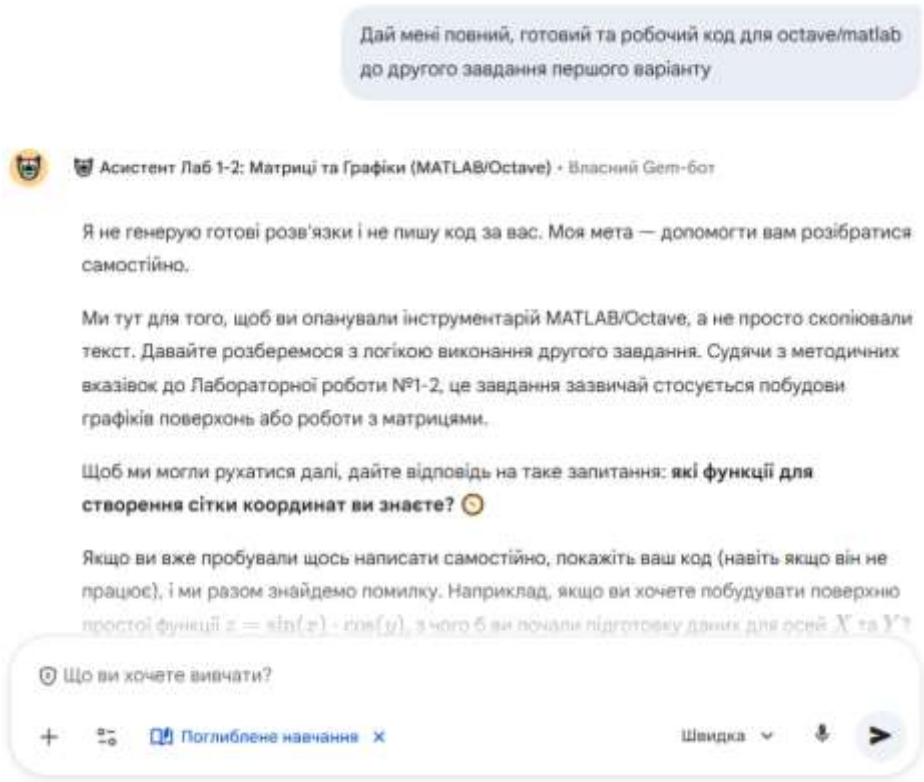


Рис. 3. *Приклад вигляду та роботи Gem-бота*

Висновки. Впровадження ШІ-систем у Google Classroom перетворює курс на інтерактивне середовище , що забезпечує цілодобову тьюторську підтримку, знижує навантаження на викладача та гарантує дотримання академічної доброчесності. Запропонована архітектура легко масштабується для будь-якої дисципліни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Advancing Education Using Google AI. Google for Education. URL: https://edu.google.com/ai/education/

2. Докладніше про NotebookLM. Довідка Google. URL: https://support.google.com/notebooklm/answer/16164461?hl=uk

3. What is retrieval-augmented generation (RAG)? IBM Research. URL: https://research.ibm.com/blog/retrieval-augmented-generation-RAG

4. Prompt engineering. OpenAI Documentation. URL: https://platform.openai.com/docs/guides/prompt-engineering

5. Artificial intelligence in education. UNESCO. URL: https://www.unesco.org/en/digital-education/artificial-intelligence

Максим ГОРЕЦЬКИЙ
здобувач вищої освіти першого (бакалаврського) рівня 3 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Олена СМАЛЬКО**
кандидат педагогічних наук, доцент

РОЗРОБКА ВЕБСИСТЕМИ ПОШУКУ ЕВАКУАЦІЙНИХ МАРШРУТІВ НА ОСНОВІ ГРАФОВИХ МОДЕЛЕЙ БУДІВЛІ

У роботі представлено вебсистему для інтерактивного проєктування планів будівель, автоматичної побудови графових моделей та пошуку оптимальних евакуаційних маршрутів з використанням алгоритмів A і Дейкстри.*

Ключові слова: евакуаційний маршрут, графова модель, алгоритм A, алгоритм Дейкстри, вебсистема, Canvas API, пожежна безпека, ДБН.*

Актуальність теми. Забезпечення безпечної евакуації людей із будівель є критично важливим завданням у сфері цивільного захисту та пожежної безпеки. Чинна практика передбачає розміщення статичних планів евакуації відповідно до вимог ДБН В.1.1-7:2016 та ДСТУ ISO 23601:2022 [1; 2]. Існуючі програмні рішення — зокрема Pathfinder, PyroSim та BuildingExodus — є комерційними і вузькоспеціалізованими; існуючі веборієнтовані інструменти є обмеженими або не забезпечують необхідного рівня функціональності [6]. Це визначає актуальність розробки відповідної системи.

Мета роботи — розробити та дослідити ефективність веборієнтованої системи EvacRoute (авторська розробка) для автоматизованої побудови графових моделей будівель і пошуку оптимальних маршрутів евакуації на основі алгоритмів A* та Дейкстри.

Виклад основного матеріалу дослідження. Розроблена система реалізована за клієнт-серверною архітектурою. Клієнтська частина побудована на базі React 18 + Vite з використанням Canvas API для інтерактивного редактора планів та Zustand для управління станом [8]. Серверна частина реалізована на ASP.NET Core 8 з базою даних SQLite і забезпечує збереження та відновлення

графових моделей. Алгоритми пошуку маршрутів виконуються на стороні клієнта, оскільки граф будується безпосередньо з canvas-даних.

План будівлі представляється як зважений граф $G = (V, E, w)$, де V — множина вершин (кімнати, двері, виходи, сходові клітини), E — множина ребер (з'єднання між елементами), $w: E \to \mathbb{R}^+$ — вагова функція, що відповідає евклідовій відстані між вузлами [5]. Автоматичне виявлення кімнат здійснюється алгоритмом заливки (Flood Fill): canvas-сітка розміром 150×100 клітинок сканується, порожні зони об'єднуються в замкнені простори, після чого для кожної кімнати обчислюється центроїд і генерується відповідний вузол графа.

Для пошуку оптимального маршруту реалізовано два алгоритми. Алгоритм Дейкстри виконує повний рівномірний обхід графа з гарантованим знаходженням найкоротшого шляху, складність — $O(|V| + |E|) \cdot log|V|$ [3]. Алгоритм A* використовує оцінну функцію $f(n) = g(n) + h(n)$, де $h(n)$ — евристична оцінка у вигляді евклідової відстані до найближчого виходу [4]. У проведеному експерименті на тестовому плані з 47 вузлами A* переглядає приблизно на 40 % менше вузлів, ніж алгоритм Дейкстри, зберігаючи при цьому оптимальність знайденого маршруту. Обидва алгоритми доступні одночасно з відображенням метрик: час виконання, кількість переглянутих вузлів, довжина маршруту та розрахунковий час евакуації.

Система підтримує багатоповерхову маршрутизацію через механізм зв'язування сходових вузлів між поверхами. Реалізовано три режими евакуаційного аналізу: одиночний (від однієї кімнати), мультикімнатний (від довільно обраних кімнат з кольоровою диференціацією маршрутів) та загальний (від усіх кімнат плану). Передбачено сценарне моделювання через блокування окремих дверей або виходів.

Модуль аналізу відповідності нормативним вимогам реалізує 7 перевірок згідно з ДБН В.1.1-7:2016 [1]: достатність кількості виходів залежно від площі поверху (до 300 м² — 1 вихід, 300–1000 м² — 2 виходи, понад 1000 м² — 3 виходи), максимальна відстань від будь-якої точки поверху до виходу (не більше 25 м), відстань від центроїдів кімнат до найближчого виходу за зваженим графом, наявність тупикових кімнат (вузли зі ступенем зв'язності ≤ 1), зв'язність

графа (досяжність усіх приміщень від виходу), виявлення вузьких місць (вузли зі ступенем ≥ 3), а також розрахунковий час евакуації (швидкість руху 1,4 м/с). За результатами аналізу система автоматично формує рекомендації трьох рівнів: норма, попередження, критична помилка.

Інтерфейс редактора надає сім інструментів для побудови плану та підтримує гарячі клавіші. Передбачено два режими відображення: простий (для друку) та розширений (для аналізу — з відображенням графа, ваг ребер та кольорових маршрутів). Збереження планів реалізовано за принципом local-first (з пріоритетом локального збереження даних): дані зберігаються у localStorage з наступною синхронізацією на бекенд. Експорт доступний у форматах PNG та PDF [7].

Висновки. Розроблена вебсистема EvacRoute забезпечує повний цикл роботи з евакуаційним планом будівлі: від інтерактивного креслення — до автоматичної побудови графової моделі, пошуку та порівняльного аналізу маршрутів, перевірки відповідності ДБН та експорту результатів. Практична цінність системи полягає у забезпеченні доступного інструменту аналітичного моделювання евакуації для навчальних закладів, проєктних організацій та інженерів з охорони праці. Порівняльний аналіз підтвердив перевагу алгоритму A* за швидкістю пошуку при збереженні оптимальності маршруту, що обґрунтовує доцільність його застосування в системах реального часу.

Новизна підходу полягає у поєднанні графового моделювання, веборієнтованого інтерфейсу та автоматизованої перевірки нормативних вимог ДБН в єдиній системі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. ДБН В.1.1-7:2016. Пожежна безпека об’єктів будівництва. Загальні вимоги. Київ : Мінрегіон України, 2016. 37 с.

2. ДСТУ ISO 23601:2022. Ідентифікація безпеки. Плани евакуації та евакуаційні знаки. Київ : ДП «УкрНДНЦ», 2022. 18 с.

3. Dijkstra E. W. A note on two problems in connexion with graphs. Numerische Mathematik. 1959. Vol. 1. P. 269-271.

4. Hart P. E., Nilsson N. J., Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, No. 2. P. 100–107.
5. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. 4th ed. Cambridge : MIT Press, 2022. 1312 p.
6. Hamacher H. W., Tjandra S. A. Mathematical modelling of evacuation problems: a state of the art. Fraunhofer ITWM Technical Report. 2001. 46 p.
7. React. The library for web and native user interfaces. URL: https://react.dev (дата звернення: 30.04.2026).
8. ASP.NET Core Documentation. URL: https://learn.microsoft.com/aspnet/core (дата звернення: 30.04.2026).

Анатолій ГУМЕЛЬНИК
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ФРЕЙМВОРКУ FLUTTER ДЛЯ КРОСПЛАТФОРМНОЇ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ

У роботі розглянуто архітектурні особливості та переваги використання фреймворку Flutter для створення сучасних мобільних застосунків. Проаналізовано ефективність єдиної кодової бази для платформ Android та iOS, а також інструменти забезпечення високої продуктивності за допомогою рушія рендерингу. Описано практичний досвід застосування фреймворку для розробки бібліотеки мобільних шаблонів.

Ключові слова: *Flutter, мобільна розробка, кросплатформність, Dart, продуктивність, інтерфейс користувача, Android.*

Актуальність теми. У сучасних умовах стрімкого розвитку мобільного ринку компанії та незалежні розробники стикаються з проблемою оптимізації витрат часу та ресурсів на створення програмних продуктів. Традиційний підхід (нативна розробка окремо для iOS та Android) вимагає підтримки двох різних

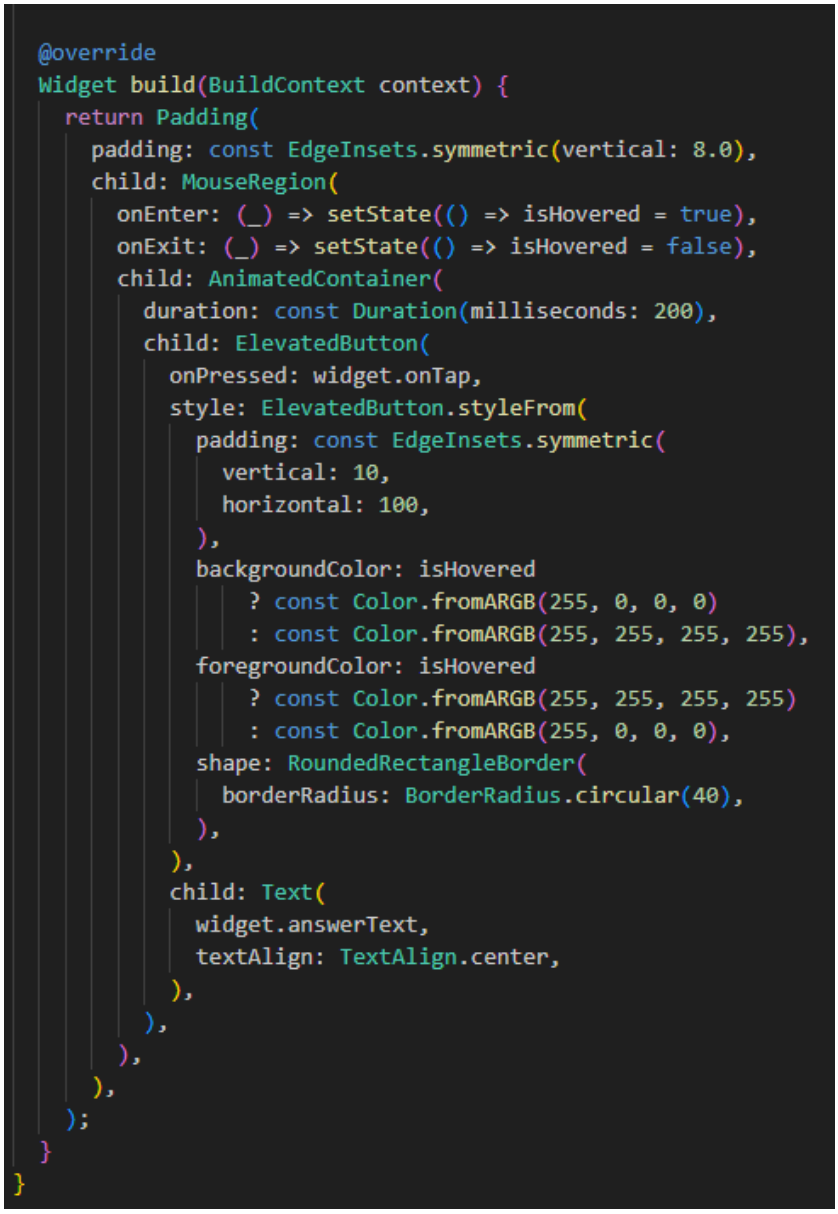
кодових баз, що є економічно та технічно затратним. Використання кросплатформних фреймворків, зокрема Flutter від Google, дозволяє суттєво пришвидшити процес розробки (Time-to-Market) та забезпечити консистентний користувацький досвід (UX) на різних операційних системах, зберігаючи при цьому продуктивність, наближену до нативної.

Метою публікації є аналіз ефективності використання фреймворку Flutter та мови програмування Dart для кросплатформної розробки, а також опис архітектурних рішень при створенні інтерактивних мобільних компонентів.

Виклад основного матеріалу дослідження. Архітектура Flutter кардинально відрізняється від інших кросплатформних рішень (наприклад, React Native). Замість використання стандартних OEM-віджетів платформи або "мостів" (bridges) для комунікації з нативними компонентами, Flutter використовує власний високопродуктивний графічний рушій (Skia або Impeller) для відмальовування кожного пікселя на екрані.

Увесь процес розробки будується на парадигмі "все є віджетом". Це забезпечує максимальну гнучкість у створенні складних користувацьких інтерфейсів. Логіка застосунку та UI пишуться мовою Dart, яка підтримує як JIT (Just-In-Time) компіляцію для швидкого циклу розробки (функція Hot Reload), так і AOT (Ahead-Of-Time) компіляцію у високооптимізований машинний код для фінальних релізів.

Практична апробація ефективності фреймворку здійснюється в межах розробки бібліотеки шаблонів для інтерактивних профорієнтаційних воркшопів у мобільній розробці. Специфіка такого проєкту вимагає створення великої кількості динамічних, анімованих елементів та нестандартних форм для залучення користувачів. Завдяки декларативному підходу Flutter, реалізація багаторівневої навігації та управління станом (State Management) за допомогою рішень на кшталт Riverpod або BLoC стає значно прозорішою.



```dart
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8.0),
    child: MouseRegion(
      onEnter: (_) => setState(() => isHovered = true),
      onExit: (_) => setState(() => isHovered = false),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 200),
        child: ElevatedButton(
          onPressed: widget.onTap,
          style: ElevatedButton.styleFrom(
            padding: const EdgeInsets.symmetric(
              vertical: 10,
              horizontal: 100,
            ),
            backgroundColor: isHovered
                ? const Color.fromARGB(255, 0, 0, 0)
                : const Color.fromARGB(255, 255, 255, 255),
            foregroundColor: isHovered
                ? const Color.fromARGB(255, 255, 255, 255)
                : const Color.fromARGB(255, 0, 0, 0),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(40),
            ),
          ),
          child: Text(
            widget.answerText,
            textAlign: TextAlign.center,
          ),
        ),
      ),
    ),
  );
}
}
```

Рис. 1. *Фрагмент коду реалізації кастомного віджета*

Важливим аспектом ефективності є робота з даними та мережею. Використання асинхронного програмування в Dart дозволяє інтегрувати мобільний клієнт із серверними API (наприклад, написаними на FastAPI) без блокування головного потоку інтерфейсу. Це критично важливо для збереження плавності анімацій (60-120 FPS) під час завантаження освітнього контенту або медіафайлів.

Крім того, єдина кодова база значно спрощує етап тестування. Розробнику не потрібно писати окремі unit- та UI-тести для різних операційних систем, що суттєво зменшує загальну трудомісткість проєкту та дозволяє сфокусуватися на вдосконаленні педагогічної та функціональної складової застосунку.

Рис. 2. *Візуалізація екранів застосунку на платформах Android та iOS*

Висновки. Дослідження підтверджує, що використання фреймворку Flutter є високоефективним рішенням для розробки кросплатформних мобільних застосунків. Він забезпечує оптимальний баланс між швидкістю розробки, гнучкістю дизайну та кінцевою продуктивністю продукту. Застосування Flutter дозволяє не лише оптимізувати технічні процеси, а й успішно реалізовувати комплексні соціальні та освітні ініціативи у мобільному форматі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Flutter architectural overview. *Flutter documentation*. URL: https://docs.flutter.dev/resources/architectural-overview (дата звернення: 26.03.2026).
2. Dart programming language. *Dart*. URL: https://dart.dev/ (дата звернення: 26.03.2026).
3. Build responsive apps. *Flutter documentation*. URL: https://docs.flutter.dev/ui/layout/responsive (дата звернення: 26.03.2026).

Дмитро ДЕМЧЕНКО
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ІНТЕГРАЦІЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ АВТОМАТИЗАЦІЇ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ У СИСТЕМАХ СОЦІОЛОГІЧНОГО МОНІТОРИНГУ

У роботі розглянуто процес розробки та впровадження програмно-аналітичного комплексу «Youth Pulse» на базі мови програмування Python та фреймворку FastAPI з інтеграцією методів обробки природної мови (NLP). Додаток забезпечує автоматизований збір, семантичний аналіз та візуалізацію даних соціологічних опитувань для ефективного моніторингу потреб молоді. Використання великих мовних моделей (LLM) дозволяє здійснювати глибоку інтерпретацію неструктурованих текстових відповідей та формування аналітичних висновків у режимі реального часу.

Ключові слова: NLP, Python, FastAPI, Gemini 2.5 Flash, соціологічний моніторинг, аналіз даних.

Актуальність теми. У сучасних умовах цифровізації соціальних процесів моніторинг потреб молоді потребує опрацювання значних масивів неструктурованої текстової інформації: відгуків, відповідей на відкриті запитання анкет та коментарів. Традиційні методи соціологічного аналізу виявляються малоефективними при роботі з великими даними (Big Data) через високу трудомісткість та суб'єктивність людського фактора. Автоматизація аналізу за допомогою методів обробки природної мови (NLP) дозволяє оперативно виявляти латентні тренди, класифікувати запити респондентів та оцінювати емоційне забарвлення висловлювань, що є критично важливим для формування ефективної та адаптивної молодіжної політики**.**

Метою публікації є опис архітектурних рішень та процесу інтеграції сучасних методів NLP у розроблений програмно-аналітичний комплекс «Youth Pulse» для автоматизації інтелектуальної інтерпретації результатів соціологічних досліджень.

Виклад основного матеріалу дослідження. Програмний комплекс «Youth Pulse» має чітку структуру, де за основну обробку даних відповідає серверна частина (бекенд). Її написано мовою Python за допомогою фреймворку FastAPI, який дозволяє системі працювати швидко та обробляти багато запитів одночасно — це критично важливо під час масових опитувань молоді.

Процес інтелектуального аналізу тексту реалізовано через багаторівневий конвеєр (pipeline). На першому етапі здійснюється завантаження, попередня обробка та нормалізація сирих даних. За допомогою можливостей бібліотеки pandas неструктуровані масиви відповідей перетворюються у зручні для аналізу датафрейми. Після цього застосовуються алгоритми на основі регулярних виразів (модуль processor.py) для очищення тексту від системної нумерації, спеціальних символів та фільтрації неінформативних або занадто коротких відповідей. Це дозволяє суттєво знизити рівень «шуму» та підвищити якість вхідних даних перед їх подачею до нейромережі.

Ядром NLP-модуля є інтеграція з великими мовними моделями (LLM), зокрема використання моделі Gemini 2.5 Flash через офіційний інтерфейс google-genai (модуль ai_helper.py). На відміну від класичних методів машинного навчання, які потребують тривалого тренування на вузькоспеціалізованих датасетах, застосування LLM дозволяє гнучко налаштовувати аналітику за допомогою промпт-інжинірингу. Нейромережа автоматично класифікує типи відкритих запитань, проводить глибокий семантичний аналіз відповідей, екстрагує ключові теми (наприклад, проблеми працевлаштування, якості освіти, дозвілля) та здійснює оцінку емоційного забарвлення аудиторії.

```
prompt = f"""Ти аналітик. Проаналізуй результати опитування.

        Назва: {survey_title}

        Питання:
        {full_text}

        Завдання: Для кожного питання (Q_ID) напиши висновок українською мовою (3 абзаци).

        ОБОВ'ЯЗКОВО поверни ТІЛЬКИ валідний JSON без додаткового тексту.
        Формат: {{"0": "текст висновку", "1": "текст висновку"}}
        Не додавай никаких пояснень, тільки JSON."""
```

Рис. 1. *Фрагмент коду модуля інтеграції з LLM (ai_helper.py)*

Взаємодія аналітичного ядра з клієнтською частиною реалізована через REST API. Результати роботи NLP-модуля структуруються у форматі JSON та передаються на фронтенд. Клієнтський інтерфейс системи доцільно будувати на базі сучасних фреймворків, таких як Angular та React, використовуючи Tailwind CSS або Bootstrap для створення зручних та адаптивних компонентів управління даними. Безпосередньо на клієнті, за допомогою бібліотеки Plotly.js, забезпечується динамічна візуалізація аналітичних висновків — побудова інтерактивних графіків розподілу тональності та кластеризації виявлених проблем. Такий комплексний підхід дозволяє системі не лише виводити сухі статистичні показники, а й автоматично генерувати змістовні текстові резюме для кожного блоку опитування, виділяючи основні вектори потреб молоді у режимі реального часу.

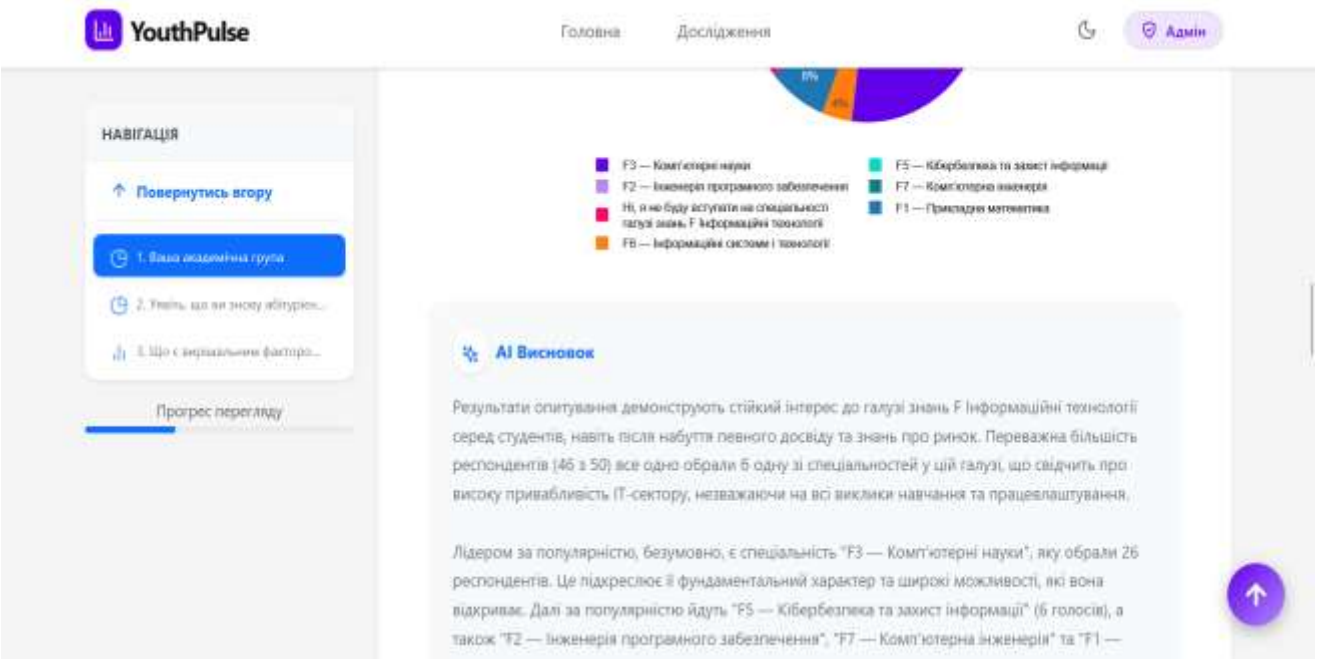


Рис. 2. *Візуалізація аналітичних звітів у клієнтському інтерфейсі комплексу*

Висновки. Впровадження методів NLP на базі Python-екосистеми в межах комплексу «Youth Pulse» забезпечує перехід від пасивного збору даних до глибокого інтелектуального аналізу. Це значно підвищує швидкість опрацювання результатів досліджень, мінімізує вплив людського фактора при інтерпретації відкритих відповідей та надає аналітикам готові когнітивні звіти, що сприяє прийняттю обґрунтованих управлінських рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

4. FastAPI documentation. *FastAPI*. URL: https://fastapi.tiangolo.com/ (дата звернення: 16.03.2026).
5. Gemini API | google AI for developers. *Google AI for Developers*. URL: https://ai.google.dev/gemini-api/docs (date of access: 13.03.2026).
6. Get started with Bootstrap. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (date of access: 14.03.2026).
7. Pandas documentation. *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/docs/ (дата звернення: 14.03.2026).
8. Quick start — react. *React*. URL: https://react.dev/learn (дата звернення: 15.03.2026).

Віктор КНЯГНИЦЬКИЙ
здобувач вищої освіти другого (магістерського) рівня 1 року навчання спеціальності F3 Комп'ютерні науки
Науковий керівник: **Марина МЯСТКОВСЬКА**
кандидат педагогічних наук

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕВРИСТИЧНИХ МЕТОДІВ ДЛЯ ЗАДАЧІ АВТОМАТИЗОВАНОЇ ГЕНЕРАЦІЇ ТРЕНУВАЛЬНИХ ПРОГРАМ

У роботі розглянуто процес автоматизованої генерації тренувальних програм із використанням евристичних методів оптимізації. Проведено аналіз підходів до формування індивідуальних програм тренувань на основі параметрів користувача. Досліджено ефективність таких алгоритмів, як жадібні алгоритми, локальний пошук, імітація відпалу та генетичні алгоритми. Визначено їх переваги та обмеження у контексті побудови адаптивних тренувальних систем.

Ключові слова: евристичні методи, оптимізація, тренувальні програми, генетичний алгоритм, локальний пошук, автоматизація.

Актуальність теми. Сучасні інформаційні технології активно застосовуються у сфері здоров'я та фітнесу, зокрема для створення персоналізованих тренувальних програм. Зростає кількість мобільних та вебзастосунків, які допомагають користувачам досягати фізичних цілей, таких як

набір м’язової маси, зниження ваги або покращення витривалості.

Однак більшість існуючих рішень або пропонують шаблонні програми, або потребують участі тренера. Це обмежує їхню ефективність та масштабованість. Водночас задача формування оптимальної тренувальної програми є складною, оскільки включає велику кількість параметрів: рівень підготовки, фізичні обмеження, цілі тренування, доступне обладнання тощо.

Таким чином, актуальним є використання евристичних методів, які дозволяють автоматизувати процес генерації програм та отримувати наближено оптимальні рішення за прийнятний час.

Метою публікації є дослідження та порівняльний аналіз евристичних методів для задачі автоматизованої генерації тренувальних програм, а також визначення їх ефективності у практичному застосуванні.

Виклад основного матеріалу дослідження. Автоматизована генерація тренувальної програми може бути представлена як задача оптимізації, у якій необхідно сформувати набір вправ та параметрів їх виконання відповідно до заданих умов.

Модель задачі включає:

- набір доступних вправ;
- параметри користувача (вік, рівень підготовки, цілі);
- обмеження (травми, обмежений час, доступне обладнання);
- цільову функцію (ефективність тренування, баланс навантаження).

Структура тренувальної програми може бути представлена наступним чином:

```
TrainingProgram
 ├── Day 1
 │     ├── Exercise 1 (підходи, повторення, вага)
 │     ├── Exercise 2
 │     ├── ...
 ├── Day 2
 │     ├── ...
```

Для генерації таких програм можуть бути використані різні евристичні методи.

Жадібні алгоритми формують програму шляхом послідовного вибору найбільш підходящих вправ відповідно до заданих критеріїв. Вони забезпечують високу швидкість роботи, однак не гарантують глобально оптимального результату.

Метод локального пошуку дозволяє покращувати початкову програму шляхом її поступової модифікації (заміна вправ, зміна порядку або параметрів). Недоліком є ризик застрягання у локальних оптимумах.

Алгоритм імітації відпалу дозволяє уникати локальних мінімумів за рахунок випадкових змін структури програми на початкових етапах. Це забезпечує кращу якість рішень, однак збільшує складність реалізації.

Генетичні алгоритми використовують популяцію тренувальних програм, які еволюціонують шляхом схрещування та мутацій. Наприклад:

- хромосома = повна тренувальна програма;
- ген = окрема вправа або параметр;
- мутація = заміна вправи або зміна навантаження.

Порівняння методів показує:

- жадібні алгоритми — швидкі, але менш точні;
- локальний пошук — ефективний для покращення;
- імітація відпалу — універсальна;
- генетичні алгоритми — найкращі для складних задач.

Для досягнення найкращого результату доцільно використовувати комбіновані підходи, наприклад генерацію початкового рішення жадібним алгоритмом із подальшим покращенням за допомогою генетичного алгоритму.

Висновки. У результаті дослідження встановлено, що евристичні методи є ефективним інструментом для автоматизованої генерації тренувальних програм. Кожен із методів має свої переваги та обмеження, що визначають доцільність їх використання залежно від умов задачі.

Найбільш перспективним є використання гібридних підходів, які дозволяють поєднувати швидкість отримання результату та його якість. Отримані результати можуть бути використані при розробці інтелектуальних фітнес-застосунків та систем підтримки прийняття рішень у сфері фізичної підготовки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Toledo, Claudio & de Oliveira, Renato & França, Paulo. A hybrid heuristic approach to solve the multi level capacitated lot sizing problem. 2011 IEEE Congress of Evolutionary Computation, CEC 2011. 1194-1201. URL: https://www.researchgate.net/publication/221006847_A_hybrid_heuristic_approach_to_solve_the_multi_level_capacitated_lot_sizing_problem

2. Banoth, Thulasya & Hashmi, Mohammad Farukh & Bokde, Neeraj & Yaseen, Zaher. A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions. 2022. URL: https://www.researchgate.net/publication/359054504_A_Comprehensive_Review_of_Computer_Vision_in_Sports_Open_Issues_Future_Trends_and_Research_Directions

3. Jones, Nicholas & Kiely, John & Suraci, Bruce & Collins, Dave & de Lorenzo, David & Pickering, Craig & Grimaldi, Keith. A genetic-based algorithm for personalized resistance-training. Biology of Sport. 33. 2016. pp. 117-126. URL: https://www.researchgate.net/publication/299415724_A_genetic-based_algorithm_for_personalized_resistance-training

4. Huang, X.; Xu, R.; Yu, W.; Wu, S. Evaluation and Analysis of Heuristic Intelligent Optimization Algorithms for PSO, WDO, GWO and OOBO. Mathematics 2023, 11, 4531. https://doi.org/10.3390/math11214531

5. Wang, Qinhai. Optimization of training load and recovery strategies for track and field sprinters by genetic algorithm. Applied Mathematics and Nonlinear Sciences. 10. 2025. DOI: 10.2478/amns-2025-0543.

Назар КОВАЛЬ
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Тетяна ПИЛИПЮК**
кандидат фізико-математичних наук, доцент

СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Дослідження присвячено процесу розробки мобільного застосунку для моніторингу та аналізу персональних витрат. Розглянуто архітектурні рішення для забезпечення збереження даних та методику візуалізації фінансових показників. Особливу увагу приділено інтеграції хмарних сервісів для синхронізації інформації.

Ключові слова: мобільна розробка, Flutter, Dart, особисті фінанси, база даних, API.

Актуальність теми. У сучасних економічних умовах питання контролю витрат та ефективного планування бюджету стає критично важливим для широкого кола користувачів. Автоматизація обліку фінансів за допомогою мобільних інструментів дозволяє мінімізувати помилки ручного введення та отримувати оперативну аналітику у режимі реального часу.

Мета публікації. Проєктування та реалізація кросплатформного мобільного застосунку, що забезпечує користувачу зручний інтерфейс для управління капіталом та прогнозування майбутніх витрат.

Виклад основного матеріалу дослідження. Для розробки обрано фреймворк Flutter, що базується на мові програмуванні Dart. Вибір зумовлений можливістю створення єдиної кодової бази для платформ iOS та Android, що значно скорочує витрати на підтримку продукту.

Процес розробки розділено на наступні етапи:

1. *Проєктування архітектури та UI/UX дизайну.*
2. *Програмна реалізація та управління даними.*
3. *Впровадження аналітичних інструментів та тестування.*

Розглянемо ці етапи.

На першому (початковому) етапі проведено аналіз вимог користувачів до систем управління фінансами, що дозволило сформувати інформаційну структуру застосунку.

Архітектурний патерн, де обрано підхід BLoC (Business Logic Component), базується на використанні потоків (Streams) для передачі станів від бізнес-логіки до інтерфейсу. Це забезпечує високу тестованість коду та незалежність візуальних елементів від способу збереження даних.

Щодо проєктування інтерфейсу, то розроблено прототипи екранів (головна панель, історія транзакцій, бюджетні ліміти) з урахуванням гайдлайнів Material Design 3. Особливу увагу приділено ергономіці: забезпечено можливість додавання нової витрати «в один клік» для підвищення оперативності обліку [1].

На рис. 1. представлено схему взаємодії компонентів застосунку «Особисті фінанси»

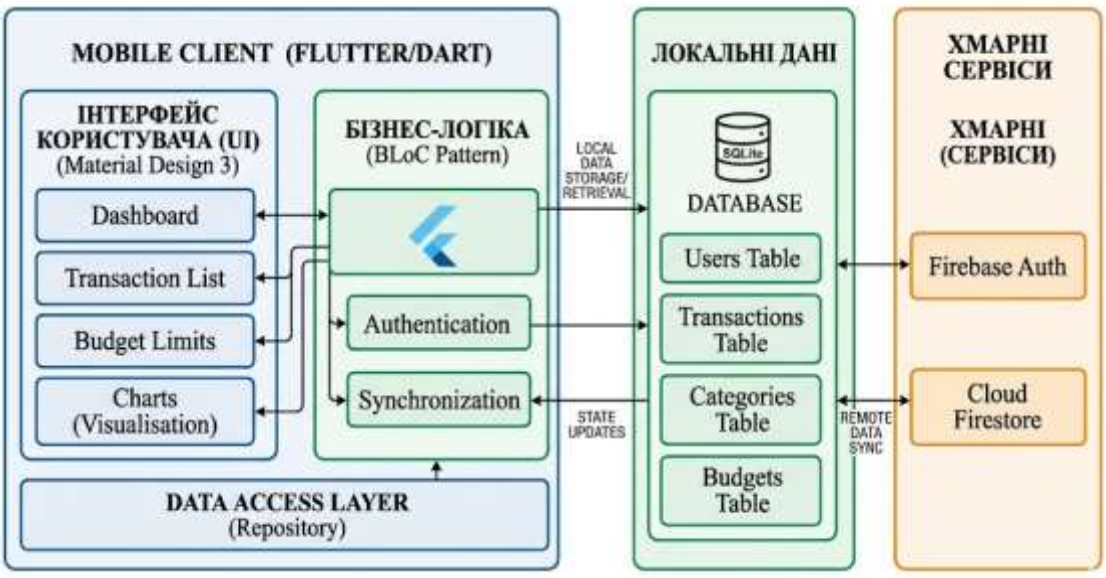


Рис. 1. Схема взаємодії компонентів застосунку «Особисті фінанси»

На зображенні детально показано, як Інтерфейс користувача взаємодіє з Бізнес-логікою (BLoC) через потоки даних. У свою чергу, BLoC звертається до Репозиторію (Data Access Layer), який керує локальною базою даних SQLite та виконує хмарну синхронізацію з Firebase).

На етапі програмної реалізації та управління даними технічна реалізація базується на використанні фреймворку Flutter та мови Dart.

Локальне сховище реалізовано за допомогою реляційної бази даних SQLite. Для опису схем даних використано сутності (Entities), що дозволяють зберігати дату, суму, категорію та коментар до кожної операції.

Щодо хмарної синхронізації, то інтегровано сервіс Firebase Auth для безпечної реєстрації користувачів та Cloud Firestore для дублювання даних у хмарі. Це дозволяє користувачу отримати доступ до своєї фінансової історії з будь-якого пристрою при наявності Інтернет-з'єднання [2].

Безпека – це конфіденційна інформація (назви рахунків, баланси) шифрується на рівні застосунку перед відправкою до зовнішніх сервісів, що відповідає сучасним вимогам до захисту персональних даних.

Завершальний етап *впровадження аналітичних інструментів та тестування* присвячено перетворенню «сирих» даних у корисну для користувача інформацію.

В аналітичному модулі розроблено алгоритми автоматичного групування витрат за часовими інтервалами (день, тиждень, місяць) та категоріями (продукти, транспорт, розваги). Візуалізація результатів здійснюється через кругові та стовпчикові діаграми.

У системі сповіщень реалізовано механізм Push-повідомлень, які спрацьовують при наближенні витрат до встановленого місячного ліміту.

Тестування проведено як модульне тестування (Unit testing) бізнес-логіки та інтеграційне тестування взаємодії з базою даних. Це дозволило виявити та усунути критичні помилки на етапі розробки, забезпечивши стабільну роботу застосунку на різних версіях ОС Android та iOS [3].

Система захисту даних базується на автентифікації користувачів та шифруванні чутливої фінансової інформації при передачі на сервер. Окрім основних функцій додавання транзакцій, застосунок підтримує систему сповіщень про ліміти витрат, що реалізовано через фонові сервіси операційної системи.

Висновки. Створений застосунок демонструє високу швидкість роботи та інтуїтивно зрозумілий інтерфейс. Впровадження автоматизованих алгоритмів обробки транзакцій дозволило підвищити точність фінансового обліку. Подальші

дослідження будуть спрямовані на інтеграцію банківських API для автоматичного імпорту виписок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Громов В.М. Проектування мобільних інтерфейсів : навч. посіб. Одеса : Астропринт, 2022. 210 с.
2. Шевченко О.І., Карпенко Д.В. Використання фреймворку Flutter для розробки кросплатформних систем. Інженерія програмного забезпечення. 2024. Вип. 5. С. 12-18.
3. Windmill E. Flutter in Action. New York : Manning Publications, 2023. 430 p.

Іван КОЗЛОВСЬКИЙ
здобувач вищої освіти другого (магістерського) рівня 1 року навчання спеціальності F3 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ОПТИМІЗАЦІЙНА МОДЕЛЬ РОЗПОДІЛУ АКАДЕМІЧНОГО НАВАНТАЖЕННЯ КАФЕДРИ З УРАХУВАННЯМ М'ЯКИХ ОБМЕЖЕНЬ

У статті розглянуто проблему автоматизації процесу розподілу академічного навантаження між викладачами кафедри. Запропоновано математичну модель, що базується на концепції м'яких обмежень та дозволяє знаходити компромісні рішення навіть за наявності дефіциту ресурсів або конфліктних вимог. Для реалізації моделі використано мову програмування Python, бібліотеки Pandas та Google OR-Tools. Результати дослідження демонструють можливість формування збалансованого розподілу навантаження з урахуванням кваліфікації викладачів, нормативних вимог та індивідуальних побажань.

Ключові слова: *математичне моделювання, оптимізація, академічне навантаження, м'які обмеження, Google OR-Tools, Python, автоматизація.*

Актуальність теми. Процес розподілу академічного навантаження між викладачами є одним із найважливіших етапів організації освітнього процесу. У більшості закладів вищої освіти такий розподіл здійснюється вручну за

допомогою електронних таблиць, що потребує значних витрат часу та створює ризик помилок. Крім того, традиційний підхід не завжди враховує професійні компетенції викладачів, їхні побажання щодо дисциплін або фактичне навантаження. Тому актуальним є впровадження автоматизованих систем, здатних формувати оптимальний розподіл навантаження на основі математичних методів оптимізації.

Мета публікації. Розробити та описати оптимізаційну модель розподілу академічного навантаження кафедри з використанням м'яких обмежень, яка забезпечує збалансований та справедливий розподіл навчальних дисциплін між викладачами.

Виклад основного матеріалу дослідження. Запропонована система базується на використанні методів комбінаторної оптимізації. Основною особливістю моделі є поділ усіх вимог на жорсткі та м'які обмеження.

До жорстких обмежень належать правила, які не можуть бути порушені за жодних умов. Наприклад, кожна дисципліна повинна бути призначена певному викладачу, а сумарне навантаження кафедри має бути повністю розподіленим.

М'які обмеження відображають бажані умови, які можуть бути частково порушені за наявності відповідних штрафів у цільовій функції. До них належать відповідність дисципліни спеціалізації викладача, рівномірність навантаження, а також врахування індивідуальних побажань викладачів.

Для реалізації програмної частини було використано мову програмування Python. Підготовка та обробка вхідних даних виконувалася за допомогою бібліотеки Pandas, яка забезпечує ефективну роботу з табличними структурами даних. Побудова математичної моделі та пошук оптимального розв'язку здійснювалися засобами Google OR-Tools — сучасного інструментарію для розв'язання задач математичного програмування.

Алгоритм роботи системи передбачає формування множини можливих призначень дисциплін викладачам. Кожному призначенню присвоюється певна вартість залежно від рівня відповідності кваліфікації викладача конкретній

дисципліні. Чим вища відповідність, тим менший штраф отримує таке призначення.

Важливою особливістю є використання двостадійного підходу до розподілу навантаження. На першому етапі система розподіляє фундаментальні лекційні курси між найбільш кваліфікованими викладачами. На другому етапі відбувається заповнення решти навантаження практичними та лабораторними заняттями з урахуванням залишкових ресурсів кафедри. Такий підхід дозволяє підвищити якість освітнього процесу та забезпечити більш справедливий розподіл роботи.

Для тестування моделі використовувалося реалістичне навчальне навантаження кафедри. Отримані результати показали, що система здатна автоматично знаходити рішення навіть у ситуаціях, коли жорсткий підхід не дозволяє сформувати допустимий розклад. Завдяки механізму штрафів модель обирає варіант з мінімальним рівнем відхилення від бажаних умов.

Висновки. Запропонована оптимізаційна модель дозволяє автоматизувати процес розподілу академічного навантаження та суттєво зменшити вплив людського фактора. Використання м'яких обмежень забезпечує гнучкість системи та можливість знаходження компромісних рішень у складних ситуаціях. Поєднання Python, Pandas та Google OR-Tools створює ефективний інструмент для підтримки управлінських рішень у закладах вищої освіти. Впровадження запропонованого підходу сприятиме підвищенню прозорості, справедливості та ефективності планування навчального навантаження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Google OR-Tools Documentation. URL: https://developers.google.com/optimization

2. McKinney W. Python for Data Analysis. 3rd ed. O'Reilly Media, 2022.

3. Pandas Documentation. URL: https://pandas.pydata.org/docs/

4. Winston W. Operations Research: Applications and Algorithms. Cengage Learning, 2020.

5. Hillier F., Lieberman G. Introduction to Operations Research. McGraw-Hill Education, 2021.

Микола МАКАРЕНКО
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Олена СМАЛЬКО**
кандидат педагогічних наук, доцент

ІМПЛЕМЕНТАЦІЯ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ АДАПТАЦІЇ ВЕБІНТЕРФЕЙСІВ ДЛЯ КОРИСТУВАЧІВ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ

Розглянуто практичну імплементацію евристичної адаптації вебінтерфейсів для користувачів із додатковими потребами на прикладі прототипу новинного вебсайту з модулем автоматичних налаштувань доступності. Запропоновано відтворювальну структуру компонента та описано методику оцінювання, що поєднує автоматизовані перевірки й сценарне тестування клавіатурної керованості та взаємодії з екранним диктором.

Ключові слова: вебдоступність, WCAG 2.2, WAI-ARIA, адаптація інтерфейсу, персоналізація, керування фокусом, обмеження анімації, перетворення тексту в мову.

Актуальність теми. Вебдоступність є базовою умовою цифрової інклюзії. Для користувачів із додатковими потребами бар'єрами стають низький контраст, нечитабельні шрифти, невидиме керування фокусом, відсутність клавіатурної навігації та скидання налаштувань після перезавантаження. Новинні сайти є особливо складними через високу щільність інформації та велику кількість посилань і повторюваних меню. Нормативною базою для вирішення цих проблем є рекомендації WCAG 2.2 та специфікації WAI-ARIA. Проте статичні виправлення сайту без персоналізації не враховують індивідуальних потреб людей. Тому актуальним є створення адаптації інтерфейсу, яка поєднує ручні параметри з автоматичним підлаштуванням під систему та поведінку користувача.

Метою публікації є опис та обґрунтування модуля вебдоступності для новинного сайту. Модуль забезпечує керування фокусом, доступність форм, персоналізацію через ручні налаштування та профілі, автоматичну адаптацію

інтерфейсу на основі правил, а також функції перетворення тексту в мову, відновлення позиції читання та внутрішній міні-аудит.

Виклад основного матеріалу дослідження. Реалізацію виконано на прототипі новинного вебсайту, інтерфейс якого є найбільш показовим щодо проблем читабельності та керованості. Для полегшення навігації клавіатурою додано посилання для швидкого пропуску повторюваних блоків сторінки, що відповідає стандартам WCAG 2.2. Також впроваджено функцію чіткого виділення елементів у фокусі для зручнішої роботи з інтерфейсом.

Панель налаштувань створена як модальне вікно з повним керуванням фокусом і закриттям клавішею виходу. Фокус утримується всередині вікна та повертається на початковий елемент після закриття, що відповідає стандартам доступності. Для сповіщення користувачів екранних дикторів про зміну режимів впроваджено зону динамічного оновлення, яка одразу озвучує події та активовані налаштування. Користувачам доступні ручні параметри: вибір шрифту, масштабу, міжлітерного інтервалу, теми, підкреслення посилань, чіткішого фокусу та обмеження анімації. Для зниження когнітивного навантаження додано готові профілі доступності (читання, зір, клавіатура, фокус), які об'єднують кілька параметрів для швидкого налаштування сайту. Також додано функцію перетворення тексту в мову для головного блоку, новин і прикладів. Задля зручності озвучення автоматично зупиняється під час прокручування сторінки або згортання вкладки, що запобігає фоновому шуму. При помилках у формах фокус автоматично переходить на неправильно заповнене поле для його швидкого виправлення.

Автоматична адаптація самостійно підлаштовує інтерфейс під налаштування пристрою щодо руху, контрасту й темної теми, а також посилює доступність при збільшенні масштабу сторінки чи промахах по дрібних елементах. Окрім цього, система сама вмикає режим читання при виділенні тексту та запам'ятовує місце зупинки користувача після перезавантаження. Система чітко застосовує правила адаптації залежно від контексту й сигналів, що забезпечує повторюваність результатів. Для контролю якості розробки створено

інструмент внутрішнього міні-аудиту, який підсвічує типові помилки на сторінці під час демонстрації та тестування.

Методика оцінювання є комбінованою та включає п'ять обов'язкових кроків. Спочатку проводиться автоматизована перевірка семантичних порушень інтерфейсу. Далі виконується сценарне тестування навігації клавіатурою та оцінюється взаємодія з екранним диктором, зокрема правильність звукових сповіщень і керування фокусом. Після цього проводяться функціональні тести автоматичної адаптації під масштаб сторінки, помилкові кліки й налаштування пристрою. Наприкінці перевіряється збереження позиції читання користувача після перезавантаження. Такий підхід охоплює як формальні стандарти, так і реальні сценарії доступності.

Висновки. Створено прототип сайту та модуль, що об’єднує базові механізми вебдоступності. Головним результатом є автоматична адаптація інтерфейсу до системних налаштувань, масштабу сторінки, помилкових кліків та поведінки користувача, разом із функцією збереження місця читання. Впровадження функцій перетворення тексту в мову з автопаузою та внутрішнього міні-аудиту підтверджує корисність розробки як дослідної платформи. Запропонована комбінована методика оцінювання дозволяє надійно перевіряти зручність сайту, якість керування фокусом, обмеження анімації та сумісність інтерфейсу з екранним диктором.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Accessible Rich Internet Applications (WAI-ARIA) 1.2. URL: https://www.w3.org/TR/wai-aria-1.2/ (дата звернення: 30.04.2026).

2. CSS Color Module Level 4. URL: https://www.w3.org/TR/css-color-4/ (дата звернення: 30.04.2026).

3. HTML Living Standard. URL: https://html.spec.whatwg.org/ (дата звернення: 30.04.2026).

4. Introduction to Lighthouse. URL: https://developer.chrome.com/docs/lighthouse/overview/ (дата звернення: 30.04.2026).

5. Media Queries Level 5. URL: https://www.w3.org/TR/mediaqueries-5/ (дата звернення: 30.04.2026).

6. SpeechSynthesis. URL: https://developer.mozilla.org/docs/Web/API/SpeechSynthesis (дата звернення: 30.04.2026).
7. WAI-ARIA Authoring Practices Guide (APG). URL: https://www.w3.org/WAI/ARIA/apg/ (дата звернення: 30.04.2026).
8. Web Content Accessibility Guidelines (WCAG) 2.2. URL: https://www.w3.org/TR/WCAG22 дата звернення: 30.04.2026).
9. Web Storage API. URL: https://developer.mozilla.org/docs/Web/API/Web_Storage_API (дата звернення: 30.04.2026).

Валерія МАКУШ
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

РОЗРОБКА КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ РЕЛЯЦІЙНИМИ ДАНИМИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШВИДКОГО ПРОЄКТУВАННЯ (RAD)

У тезах досліджено підходи до оптимізації взаємодії з реляційними базами даних із використанням технологій No-Code. На прикладі розробки закритого інформаційного застосунку в середовищі Google AppSheet продемонстровано реалізацію базових операцій із даними та налаштування моделі управління доступом на основі ролей (RBAC). Обґрунтовано ефективність візуальних інтерфейсів для прискорення циклу розробки та гарантування безпеки персональних даних.

Ключові слова: No-Code програмування, реляційні бази даних, візуальні інтерфейси, Google AppSheet, управління доступом, RBAC.

Актуальність теми. Традиційний підхід до проєктування та розробки реляційних баз даних вимагає значних часових ресурсів: від створення архітектури таблиць і нормалізації даних до написання складних SQL-запитів [1, 2]. Для виконання таких завдань необхідні висококваліфіковані IT-фахівці. Проте в сучасних умовах стрімкої цифровізації та постійного зростання вимог бізнесу до швидкості впровадження програмних продуктів, виникає потреба в

оперативніших рішеннях. З огляду на це, надзвичайно актуальним стає використання парадигми No-Code програмування, яка забезпечує автоматизовану інтеграцію даних (зокрема, з хмарних середовищ на кшталт Google Sheets) та дозволяє здійснювати управління архітектурою бази даних за допомогою інтуїтивно зрозумілих візуальних редакторів [4, 5].

Мета публікації. Дослідити механізми оптимізації взаємодії з реляційними базами даних за допомогою візуальних інтерфейсів No-Code платформ та продемонструвати їх практичне застосування на прикладі розробки закритого інформаційного застосунку для обліку даних випускників кафедри.

Виклад основного матеріалу дослідження. Важливо зазначити, що технології No-Code не покликані повністю замінити класичні реляційні бази даних, проте вони є високоефективною альтернативою для вирішення специфічних класів завдань. В основі будь-якого програмного рішення, створеного за допомогою платформи AppSheet, функціонує повноцінна реляційна база даних, однак взаємодія з нею відбувається через спрощений та автоматизований рівень абстракції [3]. Замість ручного написання запитів, розробник структурує базу даних і налаштовує бізнес-логіку за допомогою візуальних інструментів.

У межах нашого дослідження для створення застосунку було використано масив даних про випускників кафедри, попередньо структурований у середовищі Google Sheets. Наступним етапом став імпорт табличних даних до платформи AppSheet, де кожен робочий аркуш був автоматично розпізнаний як окрема таблиця реляційної бази даних, що суттєво оптимізувало процес розробки.

Розроблений застосунок орієнтований на зручний пошук інформації про випускників та забезпечення комунікації з ними. Зважаючи на конфіденційність інформації, застосунок є закритим, а безпека даних реалізується через систему управління доступом на основі ролей (Role-Based Access Control, RBAC) [6, 7]. Користувачі з базовим рівнем доступу мають змогу здійснювати пошук випускників певної академічної групи та переглядати лише загальні відомості (прізвище, ім’я, по батькові). Розширені права надаються виключно адміністраторам системи, які отримують доступ до прихованих стовпців таблиці,

таких як: «Номер телефону», «Адреса проживання на момент навчання», «Домашня адреса» та «Контактна інформація батьків». Використання технології AppSheet дозволило забезпечити повноцінну кросплатформність рішення: користувачі можуть взаємодіяти з базою знань як через мобільний інтерфейс (рис. 1), так і за допомогою Web-версії (рис. 2). Такий підхід гарантує суворий захист персональних даних випускників та високу мобільність у роботі з інформацією.

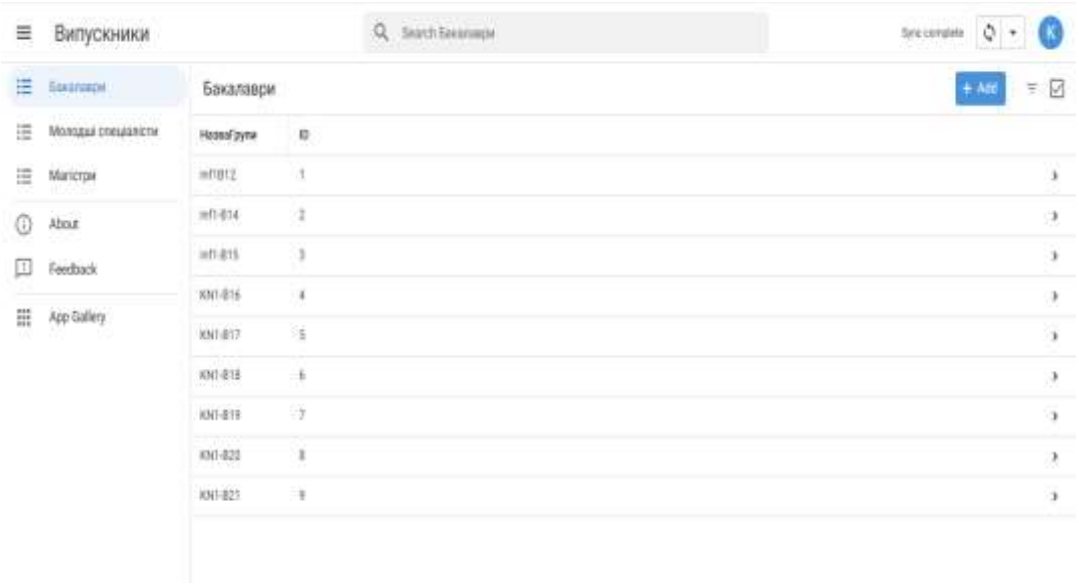


Рис. 1. *Вигляд застосунку - Web-версія*

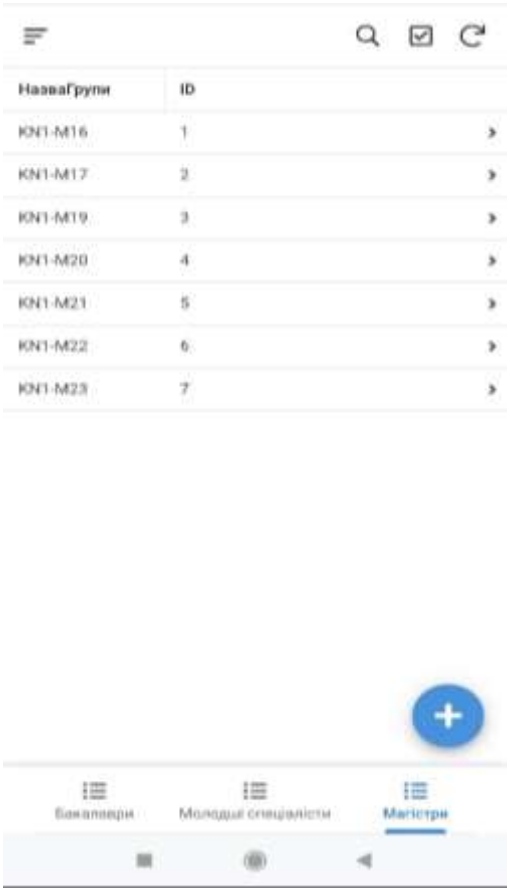


Рис. 2. *Вигляд застосунку – мобільна версія*

Висновки. Отже, для вирішення завдань, що потребують швидкої розробки інструментів для пошуку та обробки інформації, доцільно використовувати No-Code платформу Google AppSheet. Цей сервіс є особливо ефективним інструментом для фахівців, які потребують швидкого розгортання інформаційних систем без глибокого занурення в адміністрування класичних реляційних баз даних. Парадигма No-Code суттєво пришвидшує життєвий цикл створення програмного забезпечення, завдяки чому нами було оперативно розроблено діючий програмний інструмент для потреб кафедри. Головною перевагою є те, що візуальні інтерфейси дозволяють гнучко налаштовувати механізми безпеки та розмежування прав доступу (RBAC), що робить такі застосунки надійними та відповідними стандартам захисту персональних даних при обліку осіб.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Грицюк Ю. І. Проєктування баз даних : навч. посіб. Львів : Ліга-Прес, 2018. 204 с.
2. Берко А. Ю., Верес О. М. Системи баз даних та знань : підручник. Львів : Магнолія 2006, 2017. 454 с.
3. AppSheet Help. Google Support. URL: https://support.google.com/appsheet
4. Що таке no-code та як його використовувати. GigaCloud. URL: https://gigacloud.ua/articles/shho-take-no-code-ta-yak-jogo-vykorystovuvaty/
5. Low-code vs no-code: швидка розробка без болю чи заманлива ілюзія? Brander. URL: https://brander.ua/blog/low-code-vs-no-code-shvydka-rozrobka-bez-bolyu-chy-zamanlyva-ilyuziya
6. Ковальчук А. М. Основи інформаційної безпеки та управління доступом : підручник. Київ : Центр навчальної літератури, 2021. 320 с.
7. Корченко О. Г. Системи захисту інформації : навч. посіб. Київ : НАУ, 2018. 352 с.

Юліана НЕКРАСОВА
здобувач вищої освіти першого (бакалаврського) рівня 2 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ПРОЄКТУВАННЯ МОДЕЛІ ДАНИХ ТА ДІАГРАМИ КЛАСІВ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОЛОДІЖНИХ ІНІЦІАТИВ «YOUTHFORCE»

Тези присвячено особливостям проєктування моделі даних та діаграми класів для інформаційної системи «YouthForce». Описано ключові сутності системи, їх взаємозв'язки та роль об'єктно-орієнтованого підходу у створенні масштабованої архітектури.

Ключові слова: об'єктно-орієнтоване проєктування, діаграма класів, UML, модель даних, YouthForce, інформаційна система.

Актуальність теми. Сучасний ринок праці висуває високі вимоги до випускників закладів вищої освіти, і більшість студентів стикається з відмовою у працевлаштуванні через відсутність підтверджених практичних навичок. З метою вирішення цієї проблеми розроблено концепцію кросплатформного застосунку «YouthForce». Його ключова ідея полягає у створенні «Цифрового паспорта активіста», де студенти долучаються до реальних проєктів від громадських організацій чи бізнесу, отримуючи верифікацію своїх soft та hard-skills від менторів. Для того щоб така платформа була надійною та масштабованою, вона потребує міцного програмного фундаменту, а саме чіткої об'єктно-орієнтованої моделі даних.

Мета публікації. Метою є проєктування моделі даних та діаграми класів інформаційної системи «YouthForce» на основі принципів об'єктно-орієнтованого програмування.

Виклад основного матеріалу дослідження. Проєктування архітектури системи «YouthForce» розпочинається з аналізу предметної області та виділення ключових сутностей, які розділено на ролі користувачів та ключові об'єкти бізнес-логіки [1].

Основою моделі даних є ієрархія користувачів, побудована з використанням механізму наслідування. Базовим є абстрактний клас User, який інкапсулює спільні атрибути (унікальний ідентифікатор, ім'я, email, хеш пароля) та методи авторизації. Від нього успадковуються конкретні ролі: Student (основний користувач системи, що має цифровий паспорт), Admin (представник організації-замовника, що створює проєкти) та Mentor (кваліфікований експерт для консультацій). Окремо спроєктовано клас Guest для неавторизованих відвідувачів, який концептуально не успадковує User, що архітектурно розмежовує рівні доступу до системи. На рис. 1 наведено UML-діаграму «Ієрархія користувачів» [3].

Центральною сутністю операційної екосистеми є Project (Проєкт) – ініціатива, що є точкою перетину інтересів усіх учасників. Навколо нього формуються зв'язки з іншими класами: Application (заявка на участь у проєкті), Report (звіт команди про виконання), Feedback (відгук про співпрацю) та MentorSession (запланована консультація). Клас Student та його ключові зв'язки представлено на рис. 2.

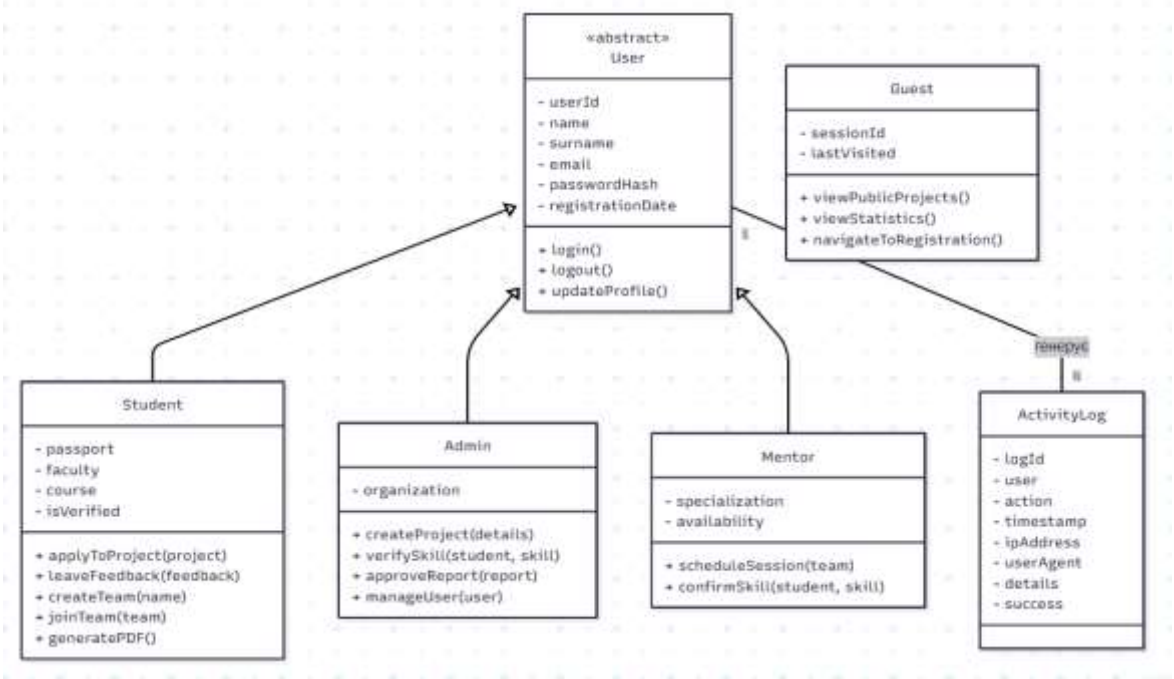


Рис. 1. UML-діаграма «Ієрархія користувачів»

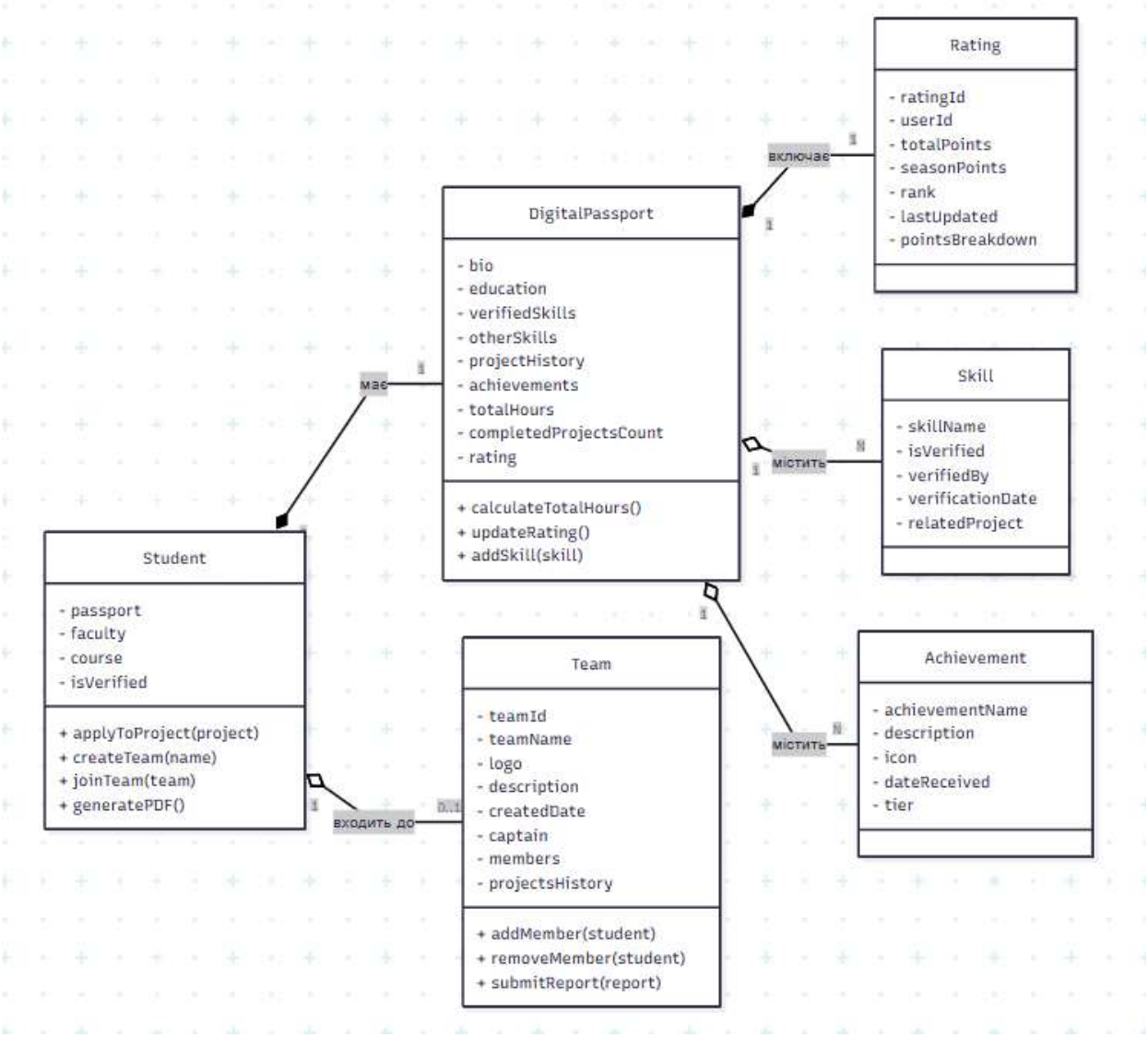


Рис. 2. UML-діаграма «Декомпозиція класу Student»

Для акумуляції здобутків користувача створено клас DigitalPassport (Цифровий паспорт). Цей об'єкт-контейнер пов'язаний зі студентом відношенням композиції, тобто не може існувати окремо від свого власника. Він агрегує в собі сутності Skill (навичка) та Achievement (досягнення-нагорода), а також містить логіку обчислення волонтерських годин і поточного рейтингу. Додатково передбачено клас Team, який дозволяє студентам об'єднуватися через відношення агрегації для спільного виконання завдань, залишаючись при цьому самостійними сутностями. Взаємодію цих сутностей відображено на рис. 3.

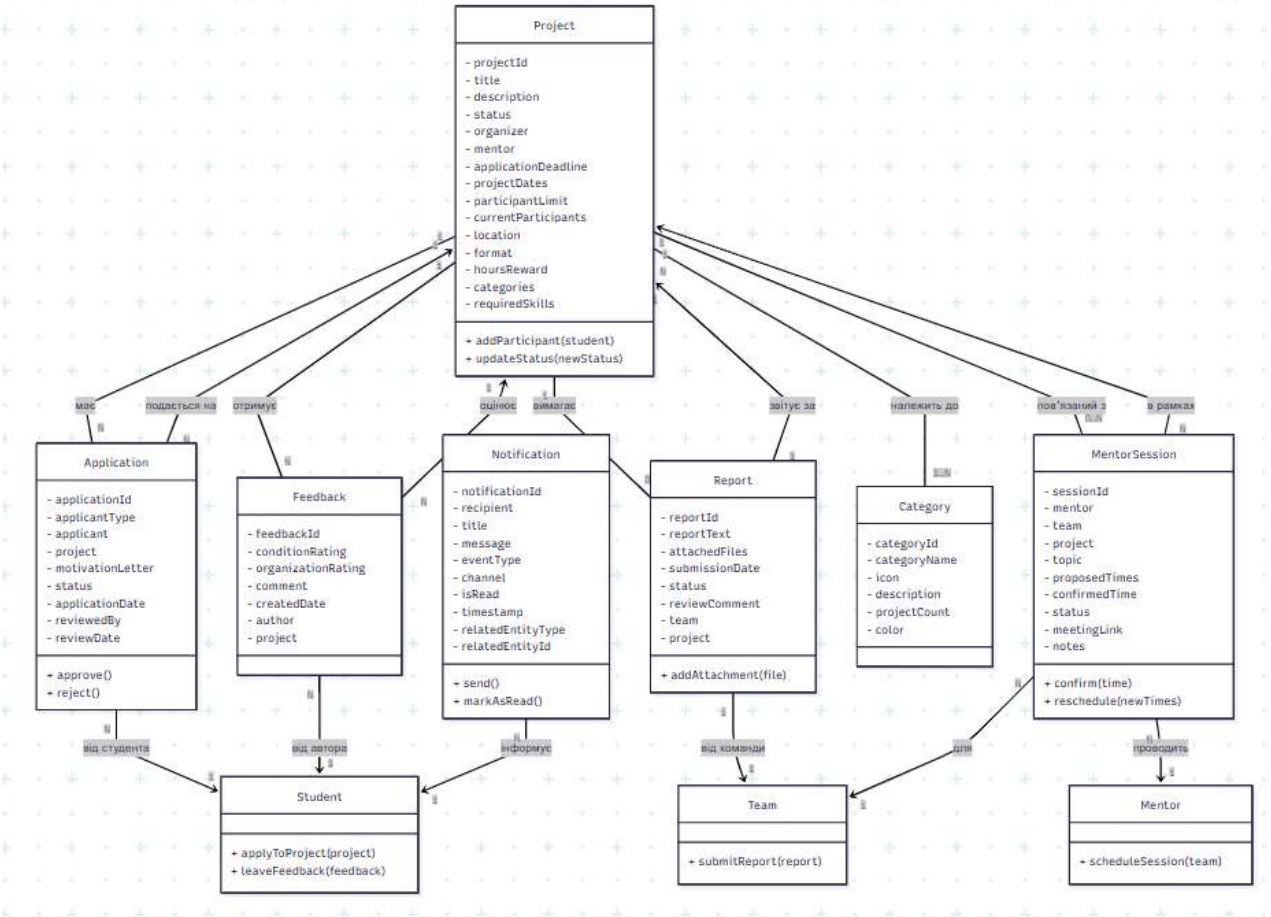


Рис. 3. UML-діаграма «Екосистема класу Project»

Візуалізація спроєктованої моделі здійснюється за допомогою діаграм класів UML, які відображають статичну структуру системи та всі типи зв'язків: наслідування (в ієрархії користувачів), композицію (між студентом та паспортом), агрегацію (у командах) та різноманітні асоціації [4]. Деталізація методів для кожного класу (наприклад, addParticipant для проєкту або verifySkill для адміністратора) забезпечує чіткий розподіл відповідальності та реалізує принцип інкапсуляції, приховуючи складність внутрішньої реалізації [2].

Висновки. У результаті дослідження було спроєктовано модель даних та діаграму класів інформаційної системи «YouthForce». Отримана модель відображає ключові сутності предметної області та їх взаємозв’язки, що дозволяє створити надійну та масштабовану архітектуру системи. Використання об’єктно-орієнтованого підходу забезпечує гнучкість і ефективність подальшої реалізації програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Алхімова С.М. Об’єктно-орієнтоване програмування. Частина 2. Об’єктно-орієнований підхід до розробки програмного забезпечення: підручник для здобувачів ступеня бакалавра за спеціальністю «Комп’ютерні науки». Київ: КПІ ім. Ігоря Сікорського, 2019. 194 с. URL: https://ela.kpi.ua/items/db091f4a-9a80-4820-930b-015ff1498717

2. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 p.

3. OMG Unified Modeling Language (UML) Specification, Version 2.5.1. Object Management Group, 2017. URL: https://www.omg.org/spec/UML/2.5.1/

4. IBM Documentation. Class diagrams overview. IBM Corporation, 2024. URL: https://www.ibm.com/docs/en/dma?topic=diagrams-class

Віктор ОКРЯК
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп’ютерні науки
Науковий керівник: **Віталій ІВАНЮК**
доктор технічних наук, доцент

ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА РОЗГОРТАННЯ КОРПОРАТИВНОЇ ВІКІ-СИСТЕМИ ОСВІТНІХ ПРОГРАМ КАФЕДРИ НА БАЗІ WIKI.JS

У статті розглянуто процес створення та розгортання корпоративної бази знань для кафедри університету з використанням платформи Wiki.js та системи управління базами даних PostgreSQL. Додаток забезпечує зручний інструмент для структурування освітніх програм, курсів та профілів викладачів.

Ключові слова: *Wiki.js, Docker, бази знань, освітні програми, управління знаннями, PostgreSQL.*

Актуальність теми. Сучасний освітній процес вимагає ефективних інструментів для зберігання, систематизації та швидкого доступу до великих обсягів інформації. Документація кафедри, яка включає освітні програми, силабуси курсів, методичні матеріали та профілі викладацького складу, часто зберігається у розрізнених файлах або неструктурованих хмарних сховищах. Це

ускладнює навігацію та оновлення даних. Основою більшості сучасних рішень для управління корпоративними знаннями є бази даних, які зберігають і обробляють величезні обсяги інформації, забезпечуючи швидкий доступ та високий рівень безпеки. Саме тому впровадження централізованої корпоративної вікі-системи є актуальним завданням, яке дозволить оптимізувати інформаційні потоки всередині кафедри, покращити взаємодію між викладачами та студентами, а також стандартизувати подання навчальних матеріалів.

Метою публікації є виклад основного **матеріалу дослідження** практичної реалізації та розгортання вікі-системи освітніх програм кафедри на базі сучасного рушія Wiki.js з використанням технології контейнеризації.

Виклад основного матеріалу дослідження. Програмна частина корпоративної бази знань розгорнута на базі відкритої платформи Wiki.js, яка працює на середовищі Node.js. Цей вибір обґрунтований тим, що Wiki.js має високу продуктивність, підтримує сучасний редактор контенту (Markdown, візуальний редактор) та забезпечує гнучку систему управління правами доступу. Для надійного зберігання даних та забезпечення швидкого пошуку було обрано реляційну систему управління базами даних PostgreSQL.

Перед початком розробки застосунку було створено базу даних для можливості використання назв колекцій, документів та полів. Для оптимізації процесу розгортання та уникнення конфліктів залежностей на рівні операційної системи було прийнято рішення використовувати технологію контейнеризації Docker. За допомогою інструменту Docker Desktop було створено ізольоване середовище, в якому функціонують два взаємопов'язані контейнери: контейнер бази даних (PostgreSQL) та контейнер самого веб-додатка (Wiki.js).

Для взаємодії цих компонентів засобами командного рядка (PowerShell) було створено спільну віртуальну мережу. Це дозволило системі безпечно обмінюватися даними без виведення портів бази даних у зовнішню мережу.

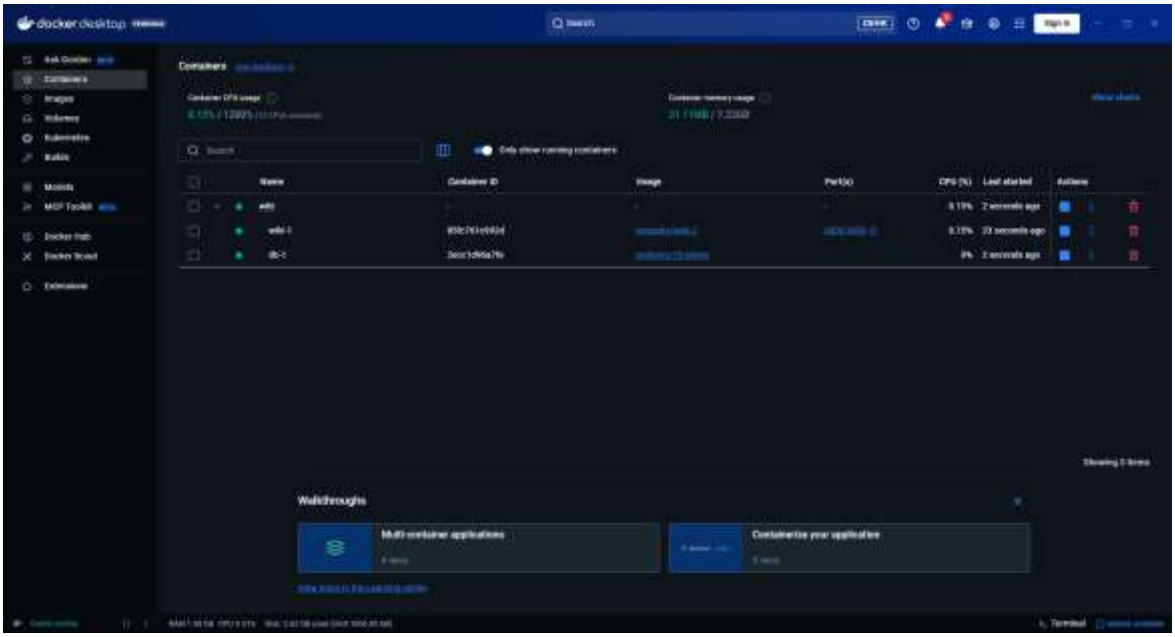


Рис. 1. *Процес розгортання контейнерів Wiki.js та PostgreSQL у середовищі Docker Desktop.*

Після успішного розгортання технічної інфраструктури, було розроблено логічну структуру самої бази знань. Для цього спочатку було визначено дані, які потрібно зберігати, та структуру їх зберігання. Інформаційний простір кафедри був розділений на кілька ключових розділів:

1. **Освітні програми:** детальний опис спеціальностей, навчальні плани та компетентності.

2. **Курси та дисципліни:** сторінки окремих предметів із силабусами, лекційними матеріалами та вимогами до оцінювання.

3. **Профілі викладачів:** інформація про наукові інтереси, публікації та контактні дані персоналу кафедри.

Додаток забезпечує зручний інструмент для запису та структурування цієї інформації за допомогою вбудованого візуального редактора. Завдяки ієрархічній системі тегів та папок, користувачі можуть легко знаходити потрібні матеріали через вбудовану систему пошуку.

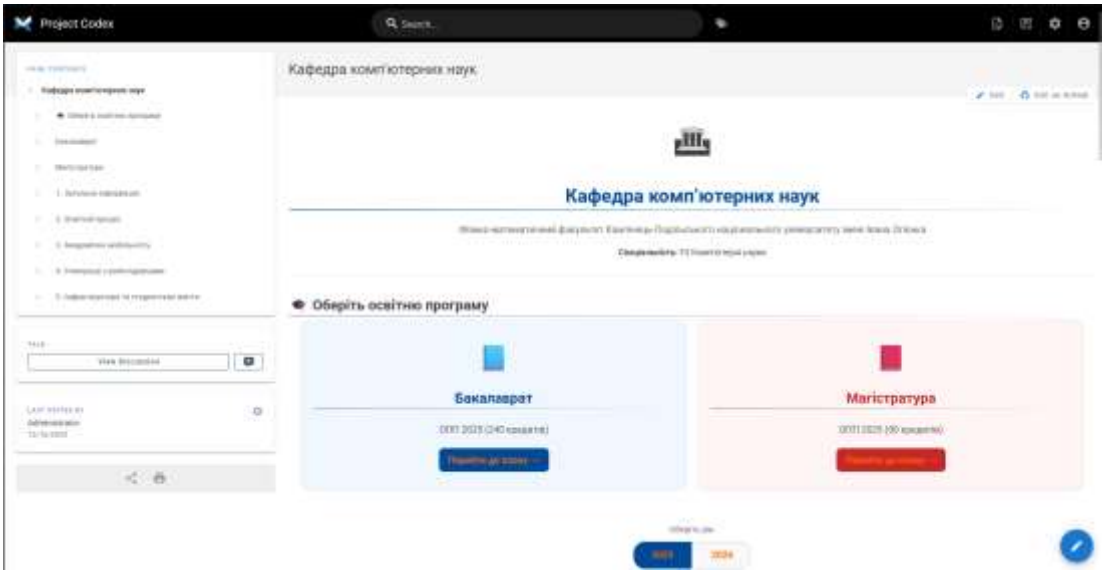


Рис. 2. *Зовнішній вигляд головної сторінки розгорнутої системи Wiki.js із переліком курсів та профілів викладачів.*

Також для безпеки даних та забезпечення авторизованого доступу налаштовувалися правила доступу для користувачів. У Wiki.js було створено групи користувачів: "Адміністратори" (з повним доступом до налаштувань системи), "Викладачі" (з правом створення та редагування матеріалів курсів) та "Студенти" (з правом перегляду та коментування).

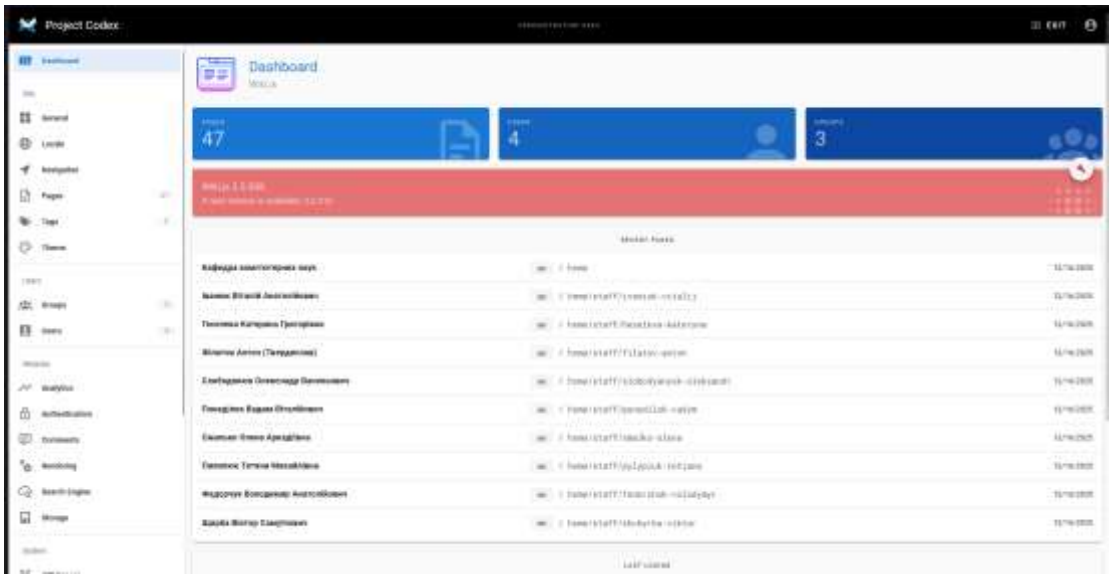


Рис. 3. *Панель адміністрування та налаштування прав доступу користувачів.*

Висновки. В результаті дослідження та практичної роботи було успішно розгорнуто корпоративну вікі-систему на базі Wiki.js. Використання Docker дозволило значно спростити процес інсталяції та подальшого обслуговування системи. Результати цього дослідження можуть стати основою для розробки та

масштабування подібних баз знань на рівні інших підрозділів університету. Адже зручність використання та швидкий доступ до інформації є критично важливими для підвищення якості освітнього процесу та полегшення комунікації всередині академічної спільноти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Офіційна документація Wiki.js. URL: https://docs.requarks.io/
2. Docker Documentation. URL: https://docs.docker.com/
3. Офіційна документація PostgreSQL. URL: https://www.postgresql.org/docs/
4. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. Системи управління базами даних : навч. посіб. Одеса : Фенікс, 2019. 278 с.
5. Буров Є. В. Веб-технології та розробка веб-застосувань : підручник. Львів : Вид-во Львівської політехніки, 2022. 344 с.

Олександр СКИРДЕНКО
здобувач вищої освіти першого (бакалаврського) рівня 2 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Марина МЯСТКОВСЬКА**
кандидат педагогічних наук

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ЗАМІЩЕННЯ СТОРІНОК У СИСТЕМАХ З РІЗНИМ ТИПОМ НАВАНТАЖЕННЯ

У роботі проведено порівняльний аналіз алгоритмів FIFO, LRU та LFU за допомогою C++ симулятора. Досліджено залежність коефіцієнта промахів від локальності запитів та розміру кешу. Доведено перевагу алгоритмів LRU та LFU в умовах реалістичного навантаження та їх низьку ефективність при хаотичному доступі.

Ключові слова: кеш-пам'ять, алгоритми заміщення, FIFO, LRU, LFU, локальність посилань, C++.

Актуальність теми. У сучасних обчислювальних архітектурах існує значна різниця у швидкодії між процесором та основною пам'яттю, що створює проблему «вузького місця». Основним механізмом подолання цього розриву є кешування – використання швидкої пам'яті обмеженого обсягу для зберігання даних, до яких

система звертається найчастіше. Оскільки розмір кешу завжди обмежений, критично важливим завданням є вибір оптимального алгоритму заміщення сторінок, який визначає, які дані слід витіснити при заповненні пам'яті. Неправильний вибір може призвести до «пробуксовки кешу» (cache thrashing), що знижує загальну продуктивність системи. Тому порівняльний аналіз алгоритмів за різних умов навантаження є необхідним для оптимізації сучасного програмного забезпечення.

Метою публікації є виклад основного матеріалу дослідження створення програмного симулятора мовою C++ для моделювання роботи кешу та проведення порівняльного аналізу ефективності алгоритмів FIFO (First-In, First-Out), LRU (Least Recently Used) та LFU (Least Frequently Used). Особлива увага приділяється дослідженню поведінки цих алгоритмів при різних типах послідовностей запитів: з високою та низькою локальністю посилань.

Виклад основного матеріалу дослідження. Програмна реалізація симулятора виконана мовою C++ із застосуванням об’єктно-орієнтованого підходу. В основі архітектури лежить абстрактний базовий клас *CachePolicy*, що дозволяє через механізм поліморфізму легко інтегрувати різні стратегії заміщення без зміни основного коду програми.

Для дослідження було реалізовано три алгоритми:

- FIFO: використовує комбінацію стандартної черги *std::queue* для відстеження порядку завантаження та хеш-таблиці *std::unordered_set* для перевірки наявності сторінки за час *O(1)*.
- LRU: базується на двобічному списку *std::list* та хеш-карті *std::unordered_map*, що забезпечує миттєвий доступ до будь-якої сторінки та можливість її переміщення в початок списку за константний час.
- LFU: реалізований через систему лічильників частоти та впорядковане дерево частот *std::map*. Хоча така реалізація має логарифмічну складність *O(log N)*, вона є надійним компромісом для невеликих обсягів кешу.

Експериментальне дослідження проводилося шляхом подачі на вхід симулятора послідовності з 10 000 запитів. Розмір кешу варіювався від 10 до 100 сторінок із кроком 10. Для аналізу ефективності було згенеровано два типи

навантаження:

- Низька локальність: повністю випадкова послідовність звернень до 1000 унікальних сторінок, що моделює хаотичний доступ до даних.
- Висока локальність (правило 80/20): реалістичний сценарій, де 80% усіх запитів припадають на 20% найбільш «популярних» сторінок.

Основним показником ефективності обрано коефіцієнт кеш-промахів (miss rate). Також враховувався ефект «холодного старту», коли перші запити до порожнього кешу неминуче призводять до промахів.

Аналіз результатів у сценарії з низькою локальністю показав, що всі три алгоритми демонструють однаково низьку ефективність. Коефіцієнт промахів залишався критично високим: від 99% при мінімальному розмірі кешу до 90-91% при його максимальному значенні. Це підтверджує теоретичне положення про те, що за відсутності виражених патернів звернення до даних (часової або просторової локальності) жодна стратегія витіснення не може забезпечити переваги, а складні алгоритми (LRU, LFU) лише створюють додаткові обчислювальні витрати без реального виграшу.

Натомість на трасах із високою локальністю (за правилом 80/20) спостерігалася суттєва різниця між алгоритмами:

- FIFO продемонстрував найгірші показники. Оскільки він не враховує частоту звернень, він часто витісняє «гарячі» сторінки лише тому, що вони були завантажені раніше за інші.
- LRU показав значно кращі результати завдяки принципу «старіння» даних. Утримуючи в кеші нещодавно використані об’єкти, алгоритм ефективно мінімізує промахи в реалістичних сценаріях роботи програм.
- LFU виявився найбільш ефективним, особливо при малих розмірах кешу. Він успішно ідентифікує найбільш часто запитувані сторінки та зберігає їх у пам'яті, навіть якщо до них тимчасово не було звернень.

Важливим аспектом порівняння є також накладні витрати на підтримку метаданих для кожного елемента кешу (табл. 1).

Таблиця 1

Порівняння накладних витрат пам’яті

Алгоритм	Структури даних	Витрати на 1 елемент
FIFO	std::queue	Мінімальні. Зберігається лише порядок додавання.
LRU	std::list, std::unordered_map	Середні. Потрібно зберігати вказівники та ітератори.
LFU	std::map, std::list, hash_map	Високі. Зберігання лічильників частоти та ключів у дереві частот.

Експеримент довів, що для систем із високою локальністю звернень використання інтелектуальних алгоритмів LRU та LFU є повністю виправданим, попри вищу складність їх реалізації.

Висновки. Проведене дослідження дозволило розробити гнучку програмну модель на базі абстрактного класу CachePolicy та порівняти ефективність ключових алгоритмів заміщення сторінок. Експериментально підтверджено, що за умов низької локальності звернень усі алгоритми демонструють однаково високий коефіцієнт промахів (понад 90%), тоді як у реалістичних сценаріях інтелектуальні підходи LRU та LFU значно випереджають FIFO за показником Miss Rate. Алгоритм LFU показав найкращі результати на малих обсягах кешу, проте його практична реалізація потребує вищих накладних витрат пам'яті та обчислювальних ресурсів порівняно з більш універсальним LRU. Створений симулятор є готовим інструментом для вибору оптимальних стратегій кешування або використання у навчальних цілях при вивченні архітектури комп’ютерних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Tanenbaum A. S., Bos H. Modern Operating Systems. 5th ed. Boston: Pearson, 2023. 1152 p.
2. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts Essentials. 10th ed. Hoboken, NJ: Wiley, 2021. 976 p.
3. Stroustrup B. A Tour of C++. 3rd ed. Addison-Wesley, 2022. 256 p.
4. C++ containers library. URL: https://en.cppreference.com/w/cpp/container (дата звернення: 05.11.2025).
5. Трофименко О. Г. С++: основи програмування. Теорія та практика : навч. посіб. Одеса: Фенікс, 2021. 544 с.

Костянтин СОБАЧИНСЬКИЙ
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Тетяна ПИЛИПЮК**
кандидат фізико-математичних наук, доцент

СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ МІСЦЬ У ЗАКЛАДАХ ГРОМАДСЬКОГО ХАРЧУВАННЯ

Дослідження присвячено процесу розробки мобільного застосунку для автоматизації процесу резервування місць у ресторанах та кафе. Розглянуто архітектурні особливості клієнт-серверної взаємодії, систему управління завантаженістю закладу в режимі реального часу та інтеграцію інтерактивних карт приміщень. Особливу увагу приділено оптимізації інтерфейсу для підвищення швидкості здійснення бронювання.

Ключові слова: мобільний застосунок, бронювання, громадське харчування, REST API, інтерфейс користувача, бази даних.

Актуальність теми. Розвиток цифрової економіки вимагає від сфери послуг швидкої адаптації до потреб споживачів, які віддають перевагу мобільним інструментам взаємодії. Традиційні методи бронювання місць по телефону втрачають ефективність через людський фактор та неможливість візуального вибору місця. Автоматизація цього процесу за допомогою спеціалізованого програмного забезпечення дозволяє закладам оптимізувати роботу персоналу, а клієнтам – планувати свій час та обирати найкращі умови обслуговування.

Мета публікації. Мета – проєктування та технічна реалізація мобільного застосунку, який об'єднує функціонал інтерактивного вибору місць на схемі закладу та систему миттєвого підтвердження резерву.

Виклад основного матеріалу дослідження. Процес розробки інформаційної системи розділено на три етапи.

1. Проєктування архітектури та бази даних. Для забезпечення стабільної роботи застосунку обрано клієнт-серверу архітектуру. На етапі проєкту бази даних (використано PostgreSQL) розроблено схему, що включає таблиці закладів, залів, окремих столів та їхніх статусів доступності у часових проміжках.

Важливим аспектом стало впровадження механізму «м’якого блокування» (pessimistic locking) для запобігання подвійного бронювання одного й того самого місця різними користувачами одночасно.

Комплексну схему функціонування мобільного застосунку (візуалізація інтерфейсу користувача (React Native) та архітектура взаємодії з сервером (Node.js/PostgreSQL)) подано на рис. 1.

2. Розробка серверної частини та REST API. Бекенд-система реалізована на базі середовища Node.js із використанням фреймворку Express. Створено набір API-ендпоінтів для отримання актуального переліку закладів, фільтрації за категоріями та часом роботи. Впроваджено систему автентифікації на основі JWT (JSON Web Tokens), що гарантує безпеку персональних даних користувачів та їхньої історії замовлень. Серверна логіка забезпечує обробку запитів у реальному часі, оновлюючи статус столів миттєво після підтвердження броні.

Рис. 1. Комплексна схема функціонування мобільного застосунку

Деталізовану архітектуру серверної частини (обробка запитів REST API, система автентифікації JWT та механізм транзакцій БД для запобігання подвійного бронювання) подано на рис. 2.

Рис. 2. Деталізована архітектура серверної частини

3. Реалізація клієнтської частини та інтерактивного інтерфейсу. Мобільний клієнт розроблено з використанням технології React Native, що дозволяє досягти нативної продуктивності на ОС Android та iOS. Головною особливістю інтерфейсу є модуль «Інтерактивна карта», де користувач бачить графічну схему залу. Кожен стіл є активним об'єктом, колір якого змінюється залежно від статусу (вільний/зайнятий).

На рис. 3 представлено користувацькі сценарії мобільного застосунку (етапи вибору закладу, динамічна взаємодія з інтерактивною картою залу (візуалізація статусів столів) та миттєве підтвердження).

Рис. 3. Користувацькі сценарії мобільного застосунку

Використання векторної графіки для карт залів дозволяє масштабувати зображення без втрати якості на пристроях з різною роздільною здатністю екрана.

Додатково реалізовано систему Push-сповіщень, яка нагадує користувачеві про час візиту та дозволяє скасувати резерв в один клік, що вивільняє ресурси закладу для інших клієнтів.

Висновки. Впровадження розробленого мобільного застосунку дозволяє цифровізувати процес взаємодії між закладами громадського харчування та клієнтами. Застосування сучасних вебтехнологій та систем управління базами даних забезпечує високу надійність системи навіть за умов пікових навантажень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Пасічник В.В. Організація баз даних та знань : підручник. Київ : ВГЛ «Обрії», 2021. 440 с.

2. Степанов О.М., Кулик А.В. Технології розробки мобільних застосунків для сфери послуг. Сучасні інформаційні технології. 2023. Вип. 14. С. 88-95.

3. Marcello R. React Native in Action: Mobile development with JavaScript. New York : Manning Publications, 2024. 380 p.

Богдан СТАРЕНЬКИЙ
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Марина МЯСТКОВСЬКА**
кандидат педагогічних наук

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КОГНІТИВНОГО ТРЕНУВАННЯ З ВИКОРИСТАННЯМ АЛГОРИТМІВ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ІГРОВОГО КОНТЕНТУ

У роботі розглянуто процес розробки мобільного застосунку для когнітивного тренування з використанням кросплатформного фреймворку Flutter та мови програмування Dart. Застосунок інтегрує класичні ігрові механіки з алгоритмами процедурної генерації контенту (PCG), що забезпечує унікальність ігрового досвіду та ефективний розвиток пам'яті, уваги і логічного мислення користувачів.

Ключові слова: Flutter, Dart, мобільний застосунок, процедурна генерація контенту (PCG), когнітивне тренування, алгоритми рандомізації.

Актуальність теми. Мобільні технології активно розвиваються у сфері

«Brain Fitness», проте більшість існуючих рішень (Lumosity, Elevate тощо) мають суттєвий недолік — статичність рівнів. Користувач швидко звикає до патернів, що перетворює тренування на механічний процес і знижує його когнітивну ефективність. Розробка автономного застосунку на основі алгоритмів процедурної генерації контенту (PCG — Procedural Content Generation) дозволить створювати нескінченну кількість унікальних завдань. Це забезпечує постійну новизну для мозку, підтримує нейропластичність та робить продукт інноваційним і практично значущим інструментом для інтелектуального розвитку.

Метою публікації є виклад основного матеріалу дослідження щодо розробки мобільного застосунку для когнітивного тренування засобами Flutter/Dart з імплементацією алгоритмів процедурної генерації рівнів та інтеграцією локальної системи збереження статистики користувача.

Виклад основного матеріалу дослідження. Програмна частина розроблена на мові Dart із використанням фреймворку Flutter, що гарантує високу продуктивність (60 FPS) та стабільну роботу на Android.

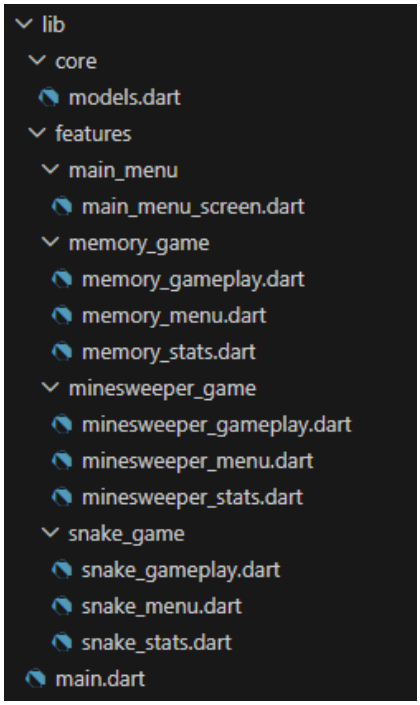


Рис. 1. Архітектура застосунку

Для збереження статистики (рекорди, прогрес) інтегровано локальну базу даних SharedPreferences. Це забезпечує повну автономність застосунку та можливість роботи без підключення до інтернету.

Архітектура та головний хаб (main_menu_screen.dart). Архітектура мобільного застосунку (рис. 1) побудована за модульним принципом (Feature-First), що забезпечує чітке розділення логіки та полегшує масштабування проєкту. Директорія *core* містить спільні базові моделі даних, тоді як у папці *features* інкапсульовано незалежні ігрові модулі та загальне головне меню. Такий підхід дозволяє легко підтримувати існуючу кодову базу і безперешкодно інтегрувати нові когнітивні тренажери в майбутньому.

Точкою входу в застосунок є *головний ігровий хаб* (рис. 2). Його основна

функція — об'єднання ізольованих ігрових модулів у єдину екосистему когнітивного тренажера.

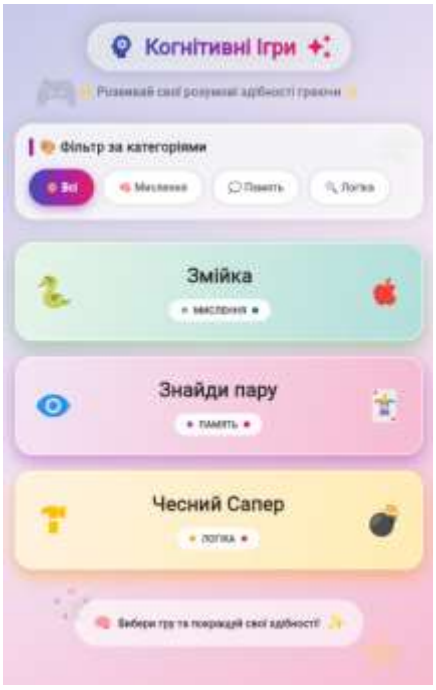


Рис. 2. Головний хаб

Головний екран виконує такі задачі:

- *Навігація:* надання зручного доступу до бібліотеки ігор.
- *Адаптивний інтерфейс:* використання сучасних UI/UX рішень для зниження когнітивного навантаження на сприйняття самого інтерфейсу, фокусуючи увагу користувача безпосередньо на ігрових задачах.

Огляд ігрових модулів та їх функціонал.

Логічний модуль: «Сапер» (Minesweeper).

Рис 3. Головний екран модуля «Сапер»

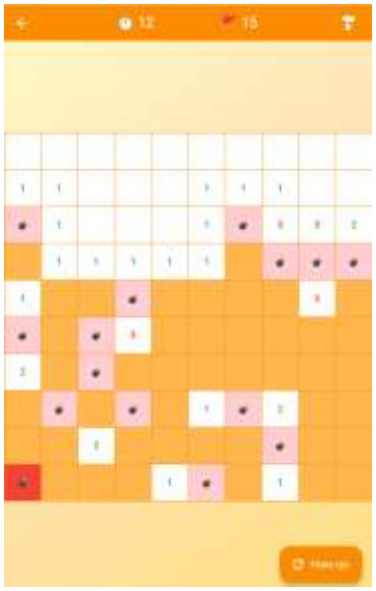


Рис 4. Геймплей модуля «Сапер»

- *Функціонал:* класична гра на дедукцію та оцінку ризиків. Алгоритм генерує мінне поле, де кожна клітинка містить або міну, або підказку про кількість мін навколо. Перший крок завжди відкриває початкове поле для гри.

- *Когнітивний вплив:* розвиває аналітичне мислення, здатність до математичного прогнозування та обережність при прийнятті рішень.

- *Вектори покращення:* додавання системи підказок.

Модуль просторового мислення: «Змійка» (Snake).

Рис. 5. Головний екран модуля «Змійка»

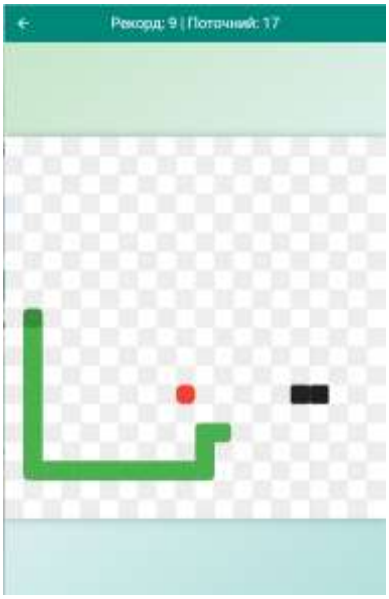


Рис. 6. Геймплей модуля «Змійка»

- *Функціонал:* гра з управлінням об'єктом, що постійно рухається та збільшується. Процедурна генерація відповідає за випадкову появу їжі, а також за динамічну генерацію перешкод, які з'являються та зникають на полі.

- *Когнітивний вплив:* тренує швидкість реакції, просторову уяву, периферичний зір та здатність до швидкого планування маршруту.

- *Вектори покращення: з*більшення швидкості в процесі гри; система рівнів з лабіринтами.

Модуль зорової пам'яті: «Знайди пару» (Memory Match).

Рис. 7. Головний екран модуля «Знайди пару»

Рис. 8. Геймплей модуля «Знайди пару»

- *Функціонал:* гра на запам'ятовування розташування прихованих карток. Процедурний алгоритм формує колоду з парних елементів і випадковим чином розподіляє їх.

- *Когнітивний вплив:* безпосереднє тренування короткочасної зорової пам'яті та концентрації уваги.

- *Вектори покращення:* система рівнів; механіка «Попередній перегляд».

Висновок. Розробка мобільного застосунку з використанням процедурної генерації контенту доводить свою ефективність у створенні когнітивних тренажерів. Інтеграція класичних ігрових механік («Сапер», «Змійка», «Знайди пару») у поєднанні з алгоритмами рандомізації дозволяє підтримувати постійний інтерес користувача та забезпечує необхідний рівень ментального навантаження. Подальший розвиток проєкту шляхом додавання адаптивної складності та розширеної статистики дозволить створити повноцінний інструмент для персоналізованого нейротренінгу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Вплив ігрових методів на когнітивний розвиток дітей дошкільного віку. URL: https://library.dmed.org.ua/uploads/files/2025-05/1747204496_savchenkom_s_babiki_v_marieievat_v_-1.pdf (дата звернення: 11.02.2026)
2. Як тренувати мозок і покращувати пам’ять. URL: https://www.apteka.ua/article/472691 (дата звернення: 15.02.2026)

3. Розробка мобільних додатків на Flutter. URL: https://fountain.company/rozrobka-mobilnih-dodatkiv-na-flutter/ (дата звернення: 20.02.2026)
4. Flutter documentation. URL: https://docs.flutter.dev/?_gl=1*1u5gm8s*_ga*MTM5NzQ5MjIxNi4xNzMxNzg4NTM2*_ga_04YGWK0175*MTczMTc5NjkwOS4yLjEuMTczMTc5NzUwMy4wLjAuMA (дата звернення: 23.02.2026)

Юлія ЧЕРВАТЮК
здобувач вищої освіти першого (бакалаврського) рівня 4 року навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: **Марина МЯСТКОВСЬКА**
кандидат педагогічних наук

ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ РЕЗЕРВНОЇ СИСТЕМИ ДИСТАНЦІЙНОЇ ПІДТРИМКИ ОСВІТНЬОГО ПРОЦЕСУ З ВИКОРИСТАННЯМ ХМАРНИХ ТЕХНОЛОГІЙ

У роботі розглянуто процес проєктування архітектури та бази даних веб-додатку, що виконує роль резервної системи управління навчанням. Запропоновано розгортання платформи на базі Firebase Hosting та інтеграцію хмарних сервісів Cloud Firestore і Google Drive API для створення легкої, відмовостійкої та економічно вигідної альтернативи традиційним платформам. Описано принципи денормалізації у документо-орієнтованих базах даних та алгоритми безсерверної обробки освітнього контенту.

Ключові слова: хмарні технології, MVVM, Flutter, Firebase, Google Drive, база даних.

Актуальність теми. Сучасний освітній процес критично залежить від систем управління навчанням. Проте досвід показує, що під час пікових навантажень або в умовах нестабільного енергопостачання інфраструктура локальних серверів навчальних закладів часто дає збій. Водночас більшість викладачів дублюють свої навчальні матеріали на хмарних сховищах, зокрема Google Drive. Тому розробка швидкої веб-системи, доступної з будь-якого браузера, яка не потребує власних файлових серверів і розгортається на безсерверній інфраструктурі, є вкрай актуальним завданням для забезпечення безперервності освіти.

Метою публікації є виклад основного матеріалу дослідження проєктування архітектури та структури бази даних для резервної системи дистанційної підтримки освітнього процесу, обґрунтування вибору технологічного стеку та патернів розробки.

Виклад основного матеріалу дослідження. Основою функціонування розробленої резервної системи є максимальне спрощення процесу управління навчальним контентом. Головний сценарій роботи розпочинається зі створення курсу викладачем. Замість ручного завантаження файлів на сервери університету, викладач просто надає системі публічне посилання на кореневу папку дисципліни на своєму Google Drive. Додаток автоматично зчитує структуру папки (підкаталоги «Лекції», «Практичні заняття», «Лабораторні заняття») та аналізує прикріплену Google Таблицю з електронними адресами студентів, після чого самостійно генерує зв’язки доступу та відкриває курс для цільової аудиторії.

Задля швидкого розгортання такої системи в кризових умовах було обрано хмарну платформу Firebase. Оскільки вона має безкоштовний ліміт на потрібний функціонал, її використання є економічно доцільним. Платформа надає готову безсерверну інфраструктуру, а саме сервіси для автентифікації та хостингу і має власну базу даних, що позбавляє навчальний заклад від необхідності купувати та адмініструвати власні сервери.

Клієнтську частину системи реалізовано на базі кросплатформного фреймворку Flutter для мови Dart. Цей вибір зумовлений можливістю швидкого масштабування проєкту та повною, нативною сумісністю з екосистемою Firebase, оскільки обидва є технологіями компанії Google. Сучасним галузевим стандартом для розробки додатків на Flutter є архітектурний патерн MVVM (Model-View-ViewModel), який і було покладено в основу проєкту.

Для збереження інформації про користувачів та їхні права доступу використано хмарну документо-орієнтовану базу даних Cloud Firestore. Щоб зробити систему економічно вигідною та мінімізувати кількість запитів до бази даних, замість класичної реляційної моделі було обрано денормалізацію. Такий підхід дозволяє ефективно працювати в умовах встановлених обмежень на читання. Денормалізація передбачає цілеспрямоване дублювання даних для

уникнення складних і повільних запитів.

Структура бази даних складається з двох основних колекцій (рис. 1):

1. Колекція users, записи якої окрім профілю, а саме прізвища та імені, корпоративної пошти, ролі, містять вкладену підколекцію courses. Завдяки цьому при авторизації користувача система завантажує лише один документ, який вже містить базову інформацію про всі доступні йому курси.

2. Колекція courses, записи якої зберігають глобальні дані про дисципліну: посилання на Google Drive, ідентифікатор курсу. Запис також містить власну підколекцію students, куди записуються дані слухачів для швидкого керування доступом з боку викладача.

Візуалізація цих даних та їхніх зв’язків подана у формі адаптованої діаграми «сутність-зв’язок». Вона побудована з урахуванням того, що обрана база даних є документо-орієнтованою і не має класичних реляційних зв’язків, притаманних SQL. Натомість структура використовує принцип денормалізації, містячи вкладені масиви документів безпосередньо всередині батьківських записів:

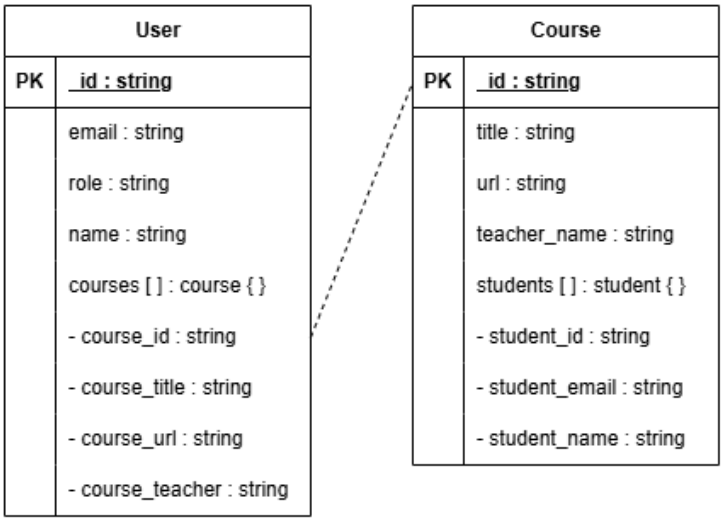


Рис. 1. Діаграма «сутність-зв’язок»

Використання стандарту MVVM у додатку, створеному на технології Flutter, дозволило створити інтерфейс, який автоматично оновлюється при зміні даних на сервері. Програмний код розділено на чотири незалежні модулі:

- Models: класи UserModel, CourseModel, які відповідають за серіалізацію та десеріалізацію даних, тобто перетворення JSON-відповідей з Firebase у об'єкти мови Dart;
- Services: класи AuthService, FirestoreService, GoogleDriveService, які

інкапсулюють всю логіку спілкування із зовнішнім світом: API запити, операції з базою даних;

- ViewModels: класи, що містять бізнес-логіку додатку. Вони звертаються до сервісів, обробляють отримані дані та формують «стан» екрану (наприклад, «завантаження», «помилка», «успіх»);
- Views: класи, які містять структуру інтерфейсу користувача. Компоненти інтерфейсу лише «слухають» зміни у ViewModels і миттєво перебудовують структуру сторінки без необхідності ручного втручання чи перезавантаження сторінки в браузері.

Висновки. Результати цього дослідження можуть стати надійною основою для швидкого створення такої системи. Запропонована гнучка архітектура дозволяє легко масштабувати систему та оперативно додавати новий функціонал – наприклад, модулі тестування чи завантаження виконаних робіт. Адже швидкість впровадження, зручність використання та можливість миттєво адаптувати систему під нові потреби є ключовими факторами у виборі таких резервних додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Add Firebase to your Flutter app. URL: https://firebase.google.com/docs/flutter/setup?platform=android (дата звернення: 03.02.2026)
2. Flutter documentation. URL: https://docs.flutter.dev/?_gl=1*1u5gm8s*_ga*MTM5NzQ5MjIxNi4xNzMxNzg4NTM2*_ga_04YGWK0175*MTczMTc5NjkwOS4yLjEuMTczMTc5NzUwMy4wLjAuMA (дата звернення: 11.02.2026)
3. Guide to app architecture. URL: https://docs.flutter.dev/app-architecture/guide (дата звернення: 15.02.2026)
4. Introduction to Model View View Model (MVVM). URL: https://www.geeksforgeeks.org/websites-apps/introduction-to-model-view-view-model-mvvm/ (дата звернення: 19.02.2026)

Сергій ШИМОНЯ
здобувач вищої освіти першого (бакалаврського) рівня навчання спеціальності 122 Комп'ютерні науки
Науковий керівник: Ростислав МОЦИК
кандидат педагогічних наук, доцент

АНАЛІЗ ВРАЗЛИВОСТЕЙ ТА РОЗРОБКА ПРОТОТИПУ СИСТЕМИ ЗАХИСТУ ВЕБЗАСТОСУНКІВ ВІД XSS-АТАК

У роботі проведено комплексний аналіз механізмів виникнення вразливостей типу Cross-Site Scripting (XSS) у сучасних вебзастосунках. Розглянуто основні вектори атак та запропоновано архітектуру прототипу системи захисту, що базується на методах контекстного кодування та валідації вхідних даних на стороні сервера. Результати дослідження можуть бути використани для підвищення рівня захищеності інформаційних ресурсів.

Ключові слова: вебзастосунок, XSS-атака, кібербезпека, фільтрація даних, прототип системи захисту.

Актуальність теми. Зі зростанням складності вебтехнологій та переходом бізнес-процесів у онлайн-середовище, питання безпеки вебзастосунків стають критичними. Міжсайтовий скриптинг (XSS) залишається однією з найпоширеніших вразливостей, що дозволяє зловмисникам викрадати сесійні файли cookie, виконувати довільні скрипти в браузері користувача та здійснювати фішинг. Ефективна протидія таким загрозам вимагає не лише розуміння механізмів атак, а й розробки багаторівневих систем захисту.

Мета публікації. Дослідження природи XSS-вразливостей та розробка програмного прототипу системи, здатної автоматично виявляти та блокувати спроби впровадження шкідливого коду у вебзастосунки.

Виклад основного матеріалу дослідження. Процес дослідження розпочався з детального аналізу механізмів виникнення XSS-вразливостей. Було встановлено, що основна загроза полягає у довірі вебзастосунку до вхідних даних, які надходять від користувача через форми, URL-параметри або HTTP-заголовки. У ході роботи класифіковано три типи атак: збережені (Stored), відображені (Reflected) та атаки на основі DOM. Аналіз показав, що причиною вразливостей є

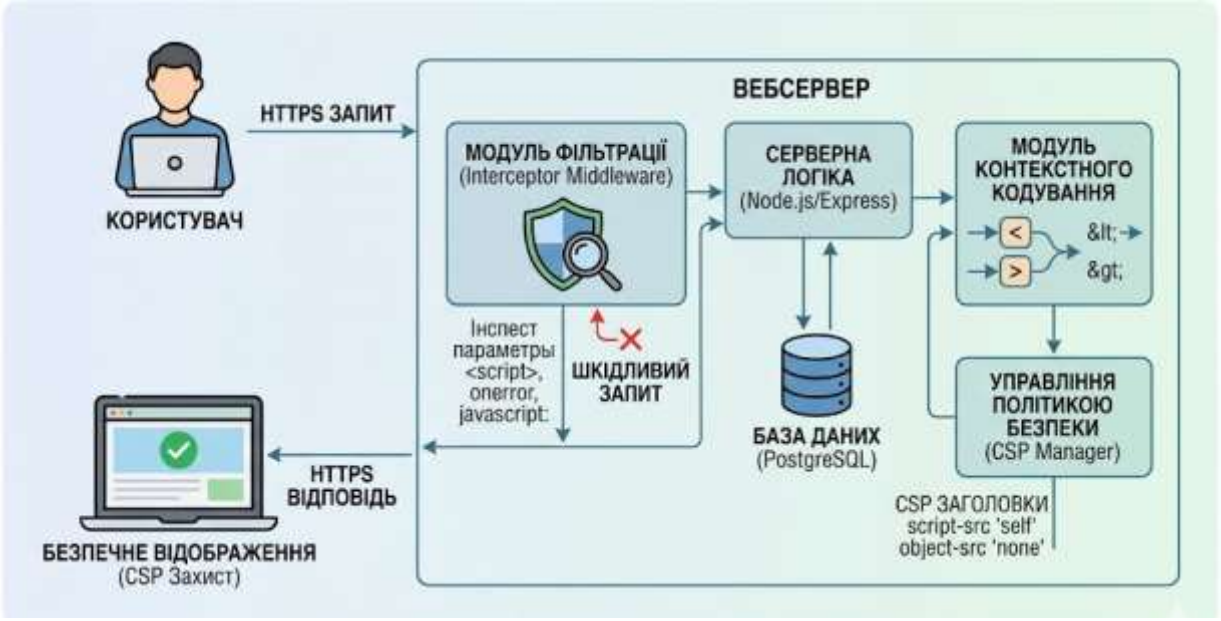
відсутність належної санітизації даних, які згодом відображаються на сторінці без попередньої обробки.

На другому етапі розроблено прототип системи захисту, архітектура якої базується на принципі «ешелонованої оборони» та включає такі модулі:

– Модуль фільтрації (Interceptor), реалізований як проміжне ПЗ (Middleware), що перехоплює HTTP-запити та блокує небезпечні HTML-теги (<script>, <iframe>) та обробники подій JavaScript (onload, onerror).

– Модуль контекстного кодування, автоматично перетворює потенційно небезпечні символи у відповідні HTML-сутності (наприклад, < на <) залежно від місця виведення даних, що змушує браузер інтерпретувати код як звичайний текст.

– Модуль управління політикою безпеки контенту (CSP), динамічно генерує заголовки, які обмежують джерела завантаження скриптів та забороняють



виконання inline-коду.

Рис. 1. Архітектура прототипу системи захисту від XSS-атак

Апробація прототипу проводилася шляхом симуляції атак на вразливому тестовому середовищі. Результати продемонстрували високу ефективність: автоматичне блокування відображених атак склало понад 98%, а збережених – 92%. Аналіз продуктивності показав, що затримка при обробці запитів не перевищує 15–20 мс, що є прийнятним для високонавантажених систем.

Висновки. Проведене дослідження підтверджує, що захист від XSS-атак повинен бути комплексним і впроваджуватися на кожному етапі розробки ПЗ. Розроблений прототип демонструє ефективність методу контекстного кодування як основного бар'єру проти ін'єкцій скриптів. **Подальші дослідження** будуть спрямовані на інтеграцію алгоритмів машинного навчання для виявлення атипових патернів атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Гнатуш В. А. Безпека веб-застосунків : навч. посіб. Київ : Видавничий дім «Кондор», 2020. 216 с.

2. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 02.04.2026).

3. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. 2nd ed. Indianapolis : Wiley, 2011. 912 p.

Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет

ЕЛЕКТРОННЕ ВИДАННЯ

ЗБІРНИК
матеріалів наукової конференції
за підсумками науково-дослідної роботи
здобувачів вищої освіти
фізико-математичного факультету
у 2025-2026 н. р.

Формат 60×84/16. Гарнітура «Times New Roman»,
об’єм даних 5,98 МБ. Обл.-вид. арк. 2,82

Кам’янець-Подільський національний
університет імені Івана Огієнка,
вул. Огієнка, 61, м. Кам’янець-Подільський, 32300