ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
__Олена ОЛЬХОВСЬКА
(підпис)

«___»__202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Розробка адаптивного тренажера для вивчення основ мови програмування Java»

зі спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня магістра

Виконавець роботи Кіліян Дмитро Володимирович

_____«___»_____202_ р.
(підпис)

Науковий керівник к. ф.-м. н., Олексійчук Юрій Федорович

_____«___»_____202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2025

РЕФЕРАТ

Записка: 49 с., 13 рис., 1 таблиця, 1 додаток, 18 джерел.

АДАПТИВНИЙ ТРЕНАЖЕР, JAVA, НАВЧАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, WPF, ПРОГРАМУВАННЯ, АДАПТИВНЕ ТЕСТУВАННЯ.

Об’єкт розробки – процес навчання студентів основам мови програмування Java в умовах індивідуалізованого цифрового освітнього середовища.

Мета роботи – проєктування та програмна реалізація адаптивного навчального програмного забезпечення для вивчення основ мови Java з використанням автоматичного добору рівня складності тестових завдань.

Методи дослідження – аналіз підходів до адаптивного навчання, інструменти побудови графічних інтерфейсів у середовищі WPF, технології платформи .NET, локальна база даних SQLite, мова програмування C Sharp, бібліотека LiveCharts, система формування PDF-звітів QuestPDF.

Виконано аналіз сучасних навчальних платформ та систем тестування, визначено їх сильні та слабкі сторони. Розроблено архітектуру адаптивного тренажера для вивчення основ Java, створено блок схеми його основних модулів та алгоритм адаптивного добору тестових завдань.

Здійснено повну програмну реалізацію адаптивного тренажера: розроблено модулі управління користувачами, тестовими сесіями, адаптивним добором завдань, статистикою та формуванням PDF-звіту. Забезпечено зручний графічний інтерфейс, побудовано механізми збереження результатів у локальній базі SQLite та візуалізації навчального прогресу.

Зміст

ПЕРЕЛІК УМОВИХ ПОЗНАЧЕНЬ....4

ВСТУП....5

РОЗДІЛ 1: ПОСТАНОВКА ЗАДАЧІ....8

1.1. Мета та завдання роботи.... 8
1.2. Вимоги до реалізації програмного забезпечення.... 9

РОЗДІЛ 2: ІНФОРМАЦІЙНА ЧАСТИНА....11

2.1. Аналіз існуючих навчальних платформ.... 11
2.2. Огляд навчальних тренажерів.... 12
2.3. Позитивні аспекти розглянутих робіт.... 13
2.4. Вади розробок з оглянутих робіт.... 14

РОЗДІЛ 3: ТЕОРЕТИЧНА ЧАСТИНА....15

3.1. Проєктування архітектури програмного забезпечення.... 15
3.2. Графічне представлення архітектури.... 17
3.3. Обґрунтування вибору програмних засобів для реалізації завдання роботи.... 23

РОЗДІЛ 4: ПРАКТИЧНА ЧАСТИНА....25

4.1. Опис програми.... 25
4.2. Опис програмної реалізації.... 28
4.3. Необхідна користувачу програми інструкція.... 40

ВИСНОВКИ....45

СПИСОК ЛІТЕРАТУРИ....48

ДОДАТОК А.КОД ПРОГРАМИ....50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
Адаптивний тренажер	Комп’ютерне програмне забезпечення, що автоматично добирає рівень складності завдань залежно від результатів користувача
C Sharp	Об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET
.NET	Кероване програмне забезпечення з відкритим кодом для операційних систем Windows, Linux, macOS
WPF (Windows Presentation Foundation)	Фреймворк для створення графічних інтерфейсів у настільних застосунках
SQLite	Локальна реляційна база даних, яка використовується для зберігання питань, відповідей, сесій та статистики
Java	Об’єктно-орієнтована мова програмування, що використовується для навчального курсу
LiveCharts	Бібліотека для побудови графіків у WPF
PdfSharp	Бібліотека для генерації PDF-файлів у C Sharp

Вступ

Підготовка майбутніх спеціалістів у сфері програмування сьогодні потребує використання інструментів, які забезпечують індивідуалізацію навчального процесу та підвищують його результативність. Класичні методики викладання, засновані на однакових для всіх практичних завданнях, що не завжди враховують різний рівень підготовки студентів та їх індивідуальний темп опанування матеріалу. Це знижує ефективність навчального процесу.

Мова програмування Java має значну популярність у різних сферах індустрії завдяки своїй універсальності та широким можливостям застосування. Вивчення основ мови програмування Java вимагає глибокого розуміння синтаксису, типів даних, алгоритмів та дає можливість практично застосовувати отримані знання.

Системи адаптивного навчання функціонують за рахунок обробки результатів тестувань, що формують історію попередніх відповідей та прогрес у темах, це дозволяє автоматично підбирати завдання відповідного рівня складності. Така методологія забезпечує оптимальне навантаження, підвищує мотивацію та значно покращує результати навчання завдяки персоналізації.

Незважаючи на наявність численних онлайн платформ для вивчення Java, більшість із них не забезпечують офлайн роботу, не дозволяють викладачу контролювати локальну базу знань, не містять адаптивних алгоритмів і не дають можливості візуально аналізувати індивідуальний прогрес студента. Це обмежує можливість їх використання у сучасності, де часто потрібна автономність і контрольованість.

У зв’язку з цим актуальним є створення локального адаптивного програмного забезпечення, яке не залежить від зовнішніх серверів, підтримує зберігання даних у локальній базі, дозволяє викладачу або студенту працювати в автономному режимі та забезпечує гнучкий механізм добору завдань залежно від результатів тестування.

Метою роботи є реалізація адаптивного тренажера для вивчення основ мови програмування Java.

Об’єктом розробки роботи є адаптивне навчальне програмне забезпечення для основ мови програмування Java.

Предмет розробки – безпосередньо програмний продукт, який реалізує адаптивне програмне забезпечення для вивчення основ мови програмування Java, реалізований на мові C Sharp з застосуванням технологій платформи .NET, фреймворку WPF та бази даних SQLite.

Головне завдання – розробити алгоритм роботи та виконати програмну реалізацію адаптивного навчального програмного забезпечення.

Методи розробки – програмні засоби мови C Sharp, технології платформи .NET, фреймворку WPF та бази даних SQLite.

При реалізації навчальної програми використано мову програмування C Sharp та середовище розробки Microsoft Visual Studio 2022.

Структура магістерської роботи складається з чотирьох розділів, висновків, списку джерел і додатків. Перший розділ розглядає постановку задачі та основних вимог до змісту. Другий розділ описує аналіз існуючих програм для адаптивного навчання. Третій розділ розглядає процес проєктування архітектури програмного забезпечення, також описано алгоритм роботи та блок схему програмного забезпечення. У четвертому розділі описано процес програмної реалізації та інструкцію по використанню

адаптивного навчального програмного забезпечення.

Обсяг пояснювальної записки: 111 стор., основна частина – 49 стор., джерела – 18 назв.

1. Постановка задачі

1.1. Мета та завдання роботи

Метою створення адаптивного програмного забезпечення є розробка системи, яка забезпечує індивідуалізацію процесу навчання мови програмування Java шляхом автоматичного добору навчальних завдань залежно від рівня знань студента, його прогресу та результатів тестування.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- розробити модель представлення студента, яка зберігатиме інформацію про пройдений матеріал, рівень засвоєння тем і статистику відповідей;
- створити адаптивний механізм добору завдань, що коригує рівень складності на основі результатів користувача;
- спроєктувати структуру локальної бази даних для зберігання користувачів, тем, запитань, відповідей і результатів тестування;
- реалізувати програмний інтерфейс з використанням WPF для забезпечення зручної взаємодії зі студентом;
- розробити модуль статистики, який відображатиме прогрес студента, результати тестів і аналітичні графіки;
- забезпечити можливість формування підсумкового PDF звіту за результатами тестування.

1.2. Вимоги до реалізації програмного забезпечення

Адаптивний тренажер має забезпечувати такі функціональні можливості:

- робота в автономному режимі без підключення до Інтернету;
- вибір або створення користувача з локальної бази даних;
- вибір теми для проходження тестування;

- наявність банку тестових запитань різного рівня складності;
- адаптивний вибір наступного питання залежно від попередніх відповідей;
- зміна рівня складності питань відповідно до успішності студента;
- збереження результатів кожної тестової сесії;
- накопичення статистики за темами, збереження загального прогресу користувача;
- перегляд профілю студента з графіками та детальною статистикою;
- формування PDF звіту зі статистикою та результатами тестування.

Оскільки адаптивний тренажер розробляється для вивчення основ мови програмування Java, пропонується сформувати такі змістові блоки, що відповідають початковому рівню вивчення мови:

- вступ до Java;
- змінні та типи даних;
- масиви;
- методи;
- оператори;
- основи ООП;
- умовні конструкції;
- цикли.

Розроблене програмне навчальне забезпечення повинно мати зрозумілий для користувача інтерфейс, тому слід зберегти таку структуру:

- **Головне вікно:**
  - вибір користувача;
  - перехід до вибору тем і параметрів тестування.
- **Вікно вибору тем:**

- перелік доступних тем;
- вибір кількості запитань.

- **Вікно тестування:**

  - формулювання питання;
  - варіанти відповідей;
  - індикатор номера поточного запитання;
  - перехід до наступного адаптивно підібраного завдання.

- **Профіль користувача:**

  - історія тестових сесій;
  - графіки успішності за темами;
  - кнопка формування PDF-звіту.

2. ІНФОРМАЦІЙНА ЧАСТИНА

2.1. Аналіз існуючих навчальних платформ

У сучасному освітньому середовищі для вивчення програмування активно застосовуються цифрові платформи, що частково виконують функції тренажерів. Їх аналіз дозволяє визначити можливості та обмеження наявних рішень, а також окреслити вимоги до розробки власної адаптивної системи.

Інтерактивна платформа Codecademy забезпечує навчання Java у режимі покрокових практичних вправ зі швидким зворотним зв’язком. Проте структура курсу є фіксованою та не адаптується під рівень знань студента [1]. Подібною за підходом є мобільна платформа SoloLearn, яка пропонує короткі модулі та тести, але також не формує індивідуальної траєкторії навчання [2].

Платформи HackerRank та LeetCode містять великі банки задач. Проте користувач самостійно обирає рівень складності, а система не оцінює поточний рівень знань і не коригує завдання адаптивно [3; 4].

Ресурс W3Schools надає довідкові матеріали з Java, проте не має механізмів тестування чи адаптивності [5]. Академічні платформи Coursera та edX вимагають постійного доступу до мережі й мають фіксовану структуру проходження матеріалу [6; 7].

У науковій літературі адаптивність розглядається як ключовий принцип побудови сучасних навчальних систем. Спірін підкреслює важливість моделі студента та моніторингу прогресу в реальному часі [8]. Биков визначає адаптивність як одну з центральних властивостей відкритих освітніх систем, що забезпечує індивідуалізацію навчання [9].

Моделі Item Response Theory (IRT), описані Лордом та Новіком, дозволяють кількісно оцінювати рівень знань студента та будувати адаптивні тестові траєкторії [10]. Модель Bayesian Knowledge Tracing, представлена Корбеттом і Андерсоном, пояснює механізм оцінювання ймовірності засвоєння навички після кожної відповіді [11].

Таким чином, аналіз платформ і наукових досліджень свідчить, що наявні рішення не забезпечують автономності, локальної бази даних і повноцінної адаптивності, що підтверджує потребу у створенні власного тренажера.

2.2. Огляд навчальних тренажерів

У межах спеціальності «Комп'ютерні науки» виконано низку робіт, що стосуються розробки навчальних тренажерів для окремих дисциплін. Аналіз цих робіт дає можливість окреслити існуючі підходи та визначити їхні функціональні обмеження.

У роботі Мандрика В. М. [12] розроблено тренажер для теми «1-R алгоритм». Основний акцент зроблено на покроковому контролі виконання обчислювальних операцій, що включають визначення частот, формування правил та оцінювання помилок. Програмне забезпечення реалізовано у вигляді статичного набору завдань без механізмів адаптації до рівня підготовки користувача.

Ярмоленко А. В. у роботі [13] представив тренажер з теми «Асимптотичні оцінки функцій». Тренажер забезпечує перевірку відповідей та відображення допоміжних підказок, проте структура взаємодії користувача залишається фіксованою, а добір завдань не здійснюється на основі попередніх результатів.

Робота Кривошея О. С. [14 розглядає реалізацію окремих елементів тренажера з дисципліни «Аналіз даних». Автор використовує традиційний

підхід до формування навчальних модулів, який передбачає послідовне виконання теоретичних і практичних завдань без динамічного коригування складності.

У досліджених роботах відсутні такі елементи, як адаптивне коригування рівня завдань, накопичення індивідуальних даних користувача, аналіз результатів попередніх тестувань та автоматичне регулювання навчальної траєкторії. Таким чином, проаналізовані програмні засоби реалізують лише базову функціональність навчальних тренажерів і не підтримують адаптивний режим роботи

2.3. Позитивні аспекти оглянутих робіт

На основі аналізу можна виділити такі позитивні аспекти сучасних навчальних платформ:

1. Швидкий зворотний зв’язок під час виконання завдань (Codecademy, HackerRank, LeetCode) [1; 3; 4].
2. Чітка організація навчального процесу за темами (SoloLearn, Coursera) [2; 6].
3. Механізми підтримки мотивації через поступове ускладнення завдань (SoloLearn) [2].
4. Наявність зручних інструментів аналізу результатів (Coursera, LeetCode) [4; 6].
5. Можливість працювати в індивідуальному темпі.

Ці аспекти визначають сильні сторони існуючих систем і частково визначають орієнтири для створення нового інструмента.

2.4. Вади розробок з оглянутих робіт

Незважаючи на переваги, сучасні платформи мають низку суттєвих недоліків щодо використання у ролі адаптивних тренажерів:

1. Відсутній механізм адаптивної зміни складності завдань (Codecademy, SoloLearn) [1; 2].
2. Робота лише з підключенням до мережі (Coursera, edX) [6; 7].
3. Відсутність локального збереження прогресу.
4. Неможливе розширення банку завдань викладачем.
5. Немає використання моделей IRT або BKT, які є сучасним стандартом адаптивних систем [10; 11].

Ці вади підкреслюють, що наявні платформи не можуть повністю замінити автономний адаптивний тренажер, побудований під конкретну освітню програму.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Проєктування архітектури програмного забезпечення

Архітектура адаптивного тренажера з мови Java побудована за принципами модульності та трирівневої організації, що забезпечує чітке розділення відповідальності, масштабованість і зручність у супроводженні. Система включає три основні рівні: рівень даних, рівень бізнес логіки та рівень представлення.

Рівень даних

Рівень даних реалізовано на основі реляційної бази даних SQLite. Вона використовується як основне локальне сховище інформації, яке містить таблиці Users, Topics, Questions, Answers, TestSessions, QuestionResults і UserTopicProgress.

Взаємодія з базою даних здійснюється за допомогою бібліотеки System.Data.SQLite. На цьому рівні реалізовані:

- моделі сутностей, які відображають структуру таблиць;
- класи доступу до даних, що виконують SQL запити;
- репозиторії, які інкапсулюють логіку роботи з даними (отримання, збереження, оновлення).

Рівень даних повністю ізольований від інтерфейсу користувача та взаємодіє виключно з рівнем бізнес логіки.

Рівень бізнес логіки

Рівень бізнес логіки є центральним компонентом архітектури і відповідає за:

- адаптивний вибір запитань;
- обробку відповідей користувача;
- формування тестових сесій;
- обчислення статистики;
- оновлення прогресу по темах;
- підготовку даних для графічного представлення і PDF звітів.

У межах цього рівня реалізовані такі модулі:

Модуль управління тестовими сесіями

Керує створенням, веденням та завершенням тестових сесій, зберігає результати у БД та забезпечує послідовність процесу тестування.

Адаптивний модуль

Реалізує алгоритм адаптивного добору складності. Система коригує рівень складності на основі правильності або неправильності відповіді. Це дозволяє поступово підвищувати складність завдань або знижувати її у разі помилок.

Модуль статистики та аналітики

Збирає інформацію про всі сесії, виконує агрегацію результатів, визначає успішність по кожній темі, формує числові показники, які пізніше відображаються у вигляді графіків та таблиць.

Рівень представлення

Рівень представлення реалізований засобами Windows Presentation Foundation (WPF). Він містить усі графічні компоненти програми:

- головне вікно;

- вікно вибору користувача;
- вікно вибору тем;
- інтерфейс тестування;
- інтерфейс результатів;
- профіль студента з графічними діаграмами;
- модуль експорту PDF звіту.

Рівень представлення взаємодіє з бізнес логікою через чітко визначені методи та не містить власних алгоритмів обробки даних.

3.2. Графічне представлення архітектури

Для опису структури системи використовуються графічні схеми, що демонструють взаємодію її компонентів. Архітектура складається з трьох основних діаграм: загальної структури системи, діаграми даних і діаграми послідовності тестування.

Загальна структурна схема

У структурному представленні система поділена на три рівні:

1. **Рівень представлення (WPF)**

    - Головне вікно
    - Вікно тестування
    - Вікно профілю
    - Вікно статистики
    - Модуль формування PDF

2. **Рівень бізнес логіки (C Sharp)**

- Модуль сесій
- Адаптивний модуль
- Модуль статистики
- Модуль обробки результатів

3. **Рівень даних (SQLite + System.Data.SQLite)**

- Таблиці Users, Topics, Questions, Answers, TestSessions, QuestionResults, UserTopicProgress
- Класи доступу до даних
- Репозиторії

Структурна схема чітко демонструє, що потік даних між рівнями односпрямований, див. рисунок 3.1.

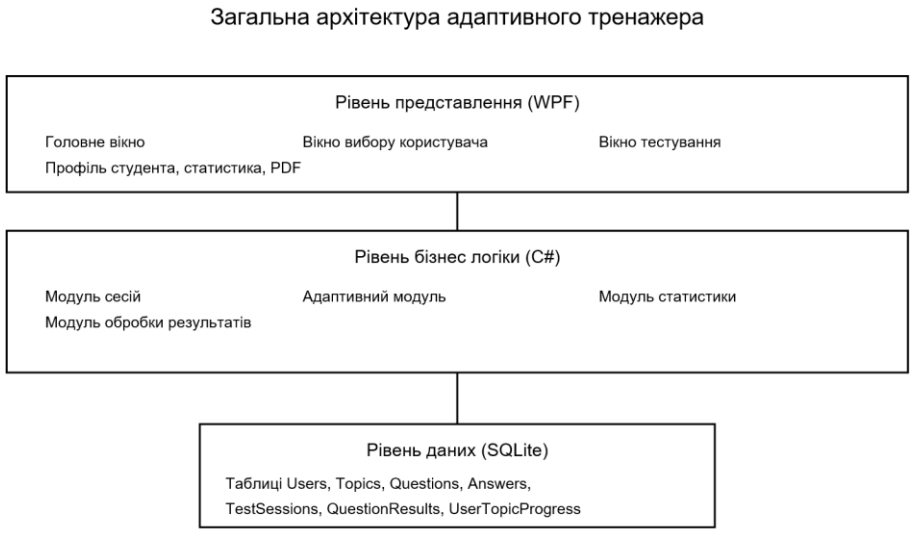

Рисунок 3.1 - Загальна архітектура адаптивного тренажера

Логічна діаграма процесу тестування

Послідовність взаємодії модулів під час тестування див. рисунок 3.2.

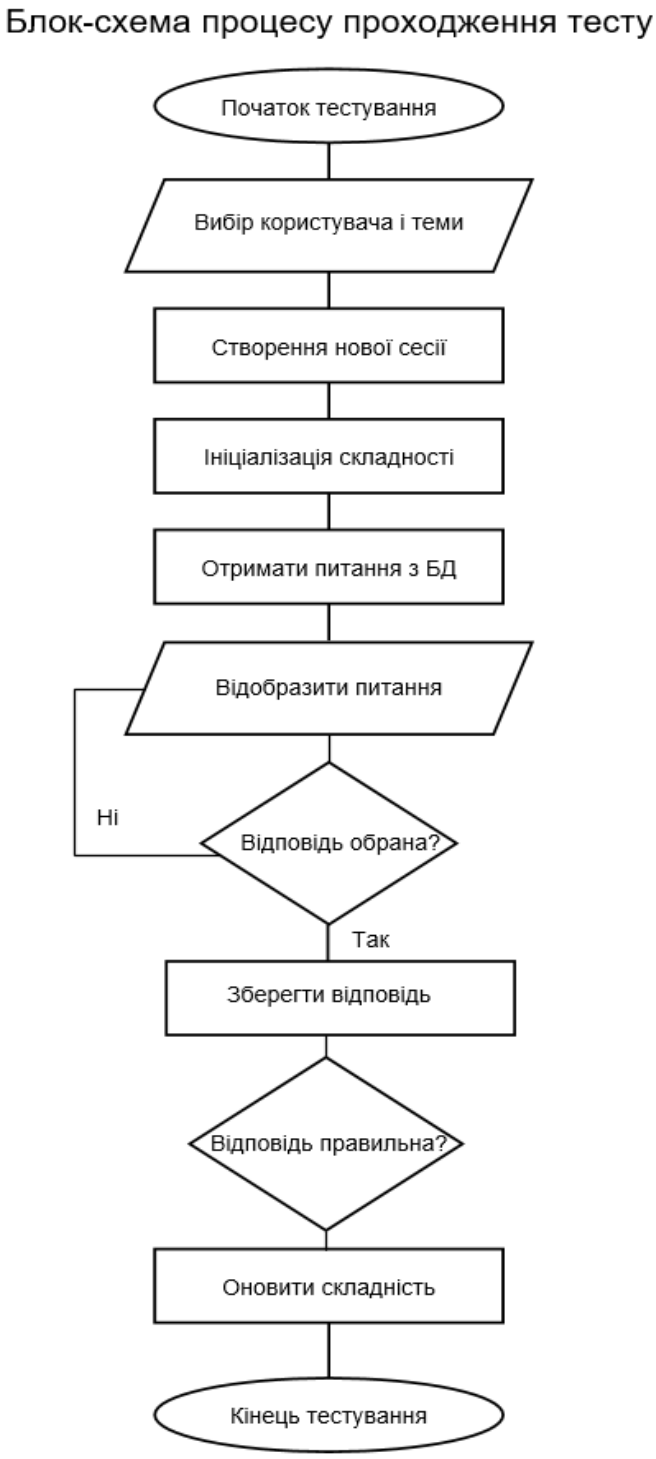

Рисунок 3.2 - Взаємодія модулів під час проходження тесту

Користувач обирає тему та починає нову тестову сесію, далі, інтерфейс надсилає команду до бізнес логіки, яка створює новий запис TestSession. Адаптивний модуль визначає початковий рівень складності див. рисунок 3.3.

Рисунок 3.3 - Робота адаптивного модуля

Бізнес логіка отримує з репозиторію питання відповідної складності. Питання передається на рівень представлення. Користувач дає відповідь, яка повертається назад у бізнес логіку. Бізнес логіка фіксує результат, оновлює складність наступного питання і зберігає інформацію в БД. Цикл повторюється, доки сесія не буде завершена. Після завершення модуль статистики агрегує результати і передає їх у інтерфейс. Користувач може переглянути графіки прогресу або сформувати PDF звіт див. рисунок 3.4.

Рисунок 3.4 - Блок формування статистики

Структура бази даних

Для відображення логічної організації бази даних, побудовано схему моделі даних, що демонструє взаємозв’язки між основними таблицями локальної бази SQLite. На ER-діаграмі подано таблиці Users, Topics, Questions, Answers, TestSessions, QuestionResults та UserTopicProgress, а також типи зв’язків між ними (див. рисунок 3.5).

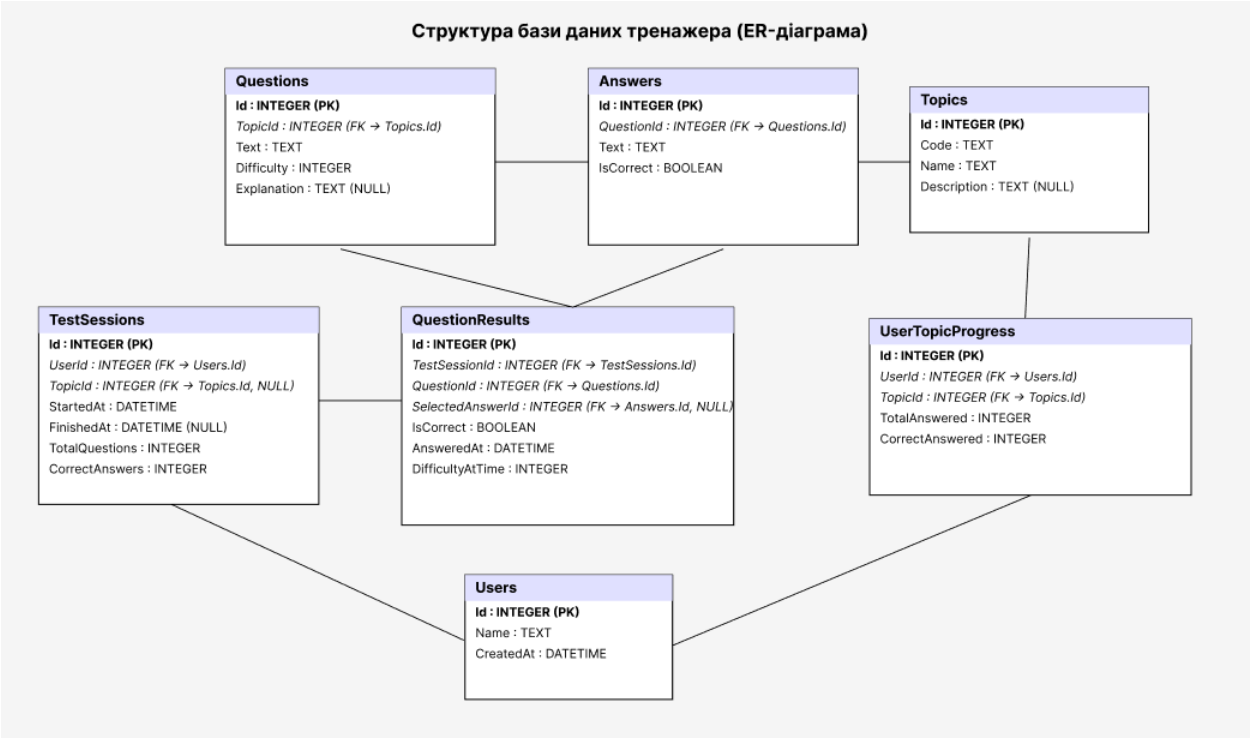

Рисунок 3.5 – Схема структури бази даних

Співвідношення між основними класами

Демонстрація взаємодії програмних модулів візуалізовано діаграмою класів, яка відображає основні компоненти системи: модуль доступу до даних (Database), репозиторій (QuestionRepository), моделі сутностей та інтерфейсні вікна (MainWindow, StudentProfileWindow). На схемі показано композиції, залежності та використання колекцій об’єктів (див. рисунок 3.6).

Рисунок 3.6 – Схема співвідношення між основними класами програми

3.3. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Вибір інструментів розробки був продиктований вимогами до системи: автономність, швидкодія, можливість локального зберігання даних, графічними інтерфейсами і підтримкою аналітичних засобів.

Мова програмування C Sharp і платформа .NET

C Sharp обрана як основна мова реалізації, оскільки вона:

- сумісна з WPF;
- підтримує об'єктно орієнтовану модель;
- має доступ до широкого набору бібліотек для роботи з файлами, базами даних і графічним інтерфейсом;
- забезпечує високу продуктивність і стабільність.

.NET надає єдине середовище виконання, що спрощує розгортання програми на різних системах Windows.

Windows Presentation Foundation (WPF)

WPF використовується як **єдина технологія побудови GUI для тренажера**. Вона забезпечує:

- підтримку XAML для опису інтерфейсу;
- зв’язування даних (Data Binding);
- шаблонізацію елементів;
- можливість створення складних візуальних компонентів.

WPF ідеально підходить для настільних навчальних систем, де важлива наочність і простота використання.

SQLite як система керування базами даних

SQLite є **єдиною системою зберігання даних, що використовується у тренажері**. Причини вибору:

- автономність (не потребує серверу);
- компактність (один файл .db);
- висока швидкість виконання операцій;
- простота розповсюдження разом із застосунком.

Взаємодія з SQLite здійснюється через бібліотеку System.Data.SQLite.

LiveCharts для побудови графіків

Для графічного відображення статистики використовується **бібліотека LiveCharts.** Вона дозволяє:

- будувати лінійні, стовпчикові та кругові діаграми;
- відображати динаміку прогресу;
- інтегрувати графіки у WPF інтерфейс без додаткових адаптерів.

LiveCharts є придатною для навчальних програм, де потрібно представити результати в наочній формі.

Модуль генерації PDF

Для експорту звітів використовується **бібліотека** PdfSharp. Функціональність включає:

- формування PDF документа;
- вставлення текстових блоків;
- вставлення таблиць і можливих графіків;
- збереження звіту у файл.

Це дозволяє студентам та викладачам отримувати формалізовані документи з результатами тестування.

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програми

Призначення та загальна структура

Програма Adaptive Java Trainer призначена для проведення адаптивного тестування з основ мови Java. Вона працює локально, використовує базу даних SQLite і дозволяє:

- зберігати профілі користувачів;
- проводити тестування за вибраними темами;
- адаптивно підбирати питання;
- вести історію сесій;
- візуалізувати прогрес;
- експортувати PDF-звіт.

Стартова точка програми — клас App:

```
namespace AdaptiveJavaTrainer
{
    public partial class App : Application
    {
    }
}
```

Головна форма MainWindow є центром взаємодії користувача з системою та містить посилання на репозиторій QuestionRepository:

```
public partial class MainWindow : Window
{
    private readonly QuestionRepository _repo;

    private List<Topic> _topics = new();
    private List<Question> _allQuestions = new();
    private HashSet<int> _asked = new();

    private Question? _currentQuestion;
    private User? _currentUser;
```

```
    private int? _currentTopicId;
    private int? _sessionId;
    private int _difficulty = 1;
    private int _questionLimit = 0;
    private int _askedCount = 0;
    private int _answeredCount = 0;
    private int _correctCount = 0;
    private List<UserTopicProgress> _userTopicStats = new();
    private readonly Random _rand = new();
    // ...
}
```

Така структура дозволяє зберігати поточний стан тестування (користувач, тема, складність, ліміти, статистика) безпосередньо в головному вікні.

Робота з користувачами та темами

Список користувачів завантажується з таблиці Users за допомогою методу GetAllUsers репозиторію, а потім прив’язується до елемента UserComboBox:

```
private void LoadUsers()
{
    try
    {
        var users = _repo.GetAllUsers();
        UserComboBox.ItemsSource = users;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Помилка завантаження користувачів: " + ex.Message);
    }
}
```

Аналогічно завантажуються теми і відображаються у відповідному списку:

```
private void LoadTopics()
{
    try
    {
        _topics = _repo.GetTopics();
        TopicsListBox.ItemsSource = _topics;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Помилка завантаження тем: " + ex.Message);
    }
}
```

Проведення тестування

Після вибору користувача, тем та параметрів тесту (кількість питань) натискання кнопки старт запускає сесію:

```
private void StartButton_Click(object sender, RoutedEventArgs e)
{
    // перевірка вибору користувача, тем і ліміту
    // ...

    _sessionId = _repo.CreateTestSession(_currentUser.Id, _currentTopicId,
_questionLimit);
    _asked.Clear();
    _askedCount = 0;
    _answeredCount = 0;
    _correctCount = 0;

    SelectNextQuestion();
    ShowQuestion();
    UpdateStatus();
}
```

Далі для кожного кроку викликається SelectNextQuestion, яка використовує згаданий адаптивний алгоритм, та ShowQuestion, що заповнює елементи інтерфейсу текстом питання і варіантами відповідей.

Результати відповідей фіксуються у таблиці QuestionResults через метод LogQuestionResult в QuestionRepository (код методу є в проєкті й тут не спотворюється), а підсумок по сесії оновлюється методом UpdateTestSessionSummary.

Статистика та профіль студента

При переході до профілю студента відкривається вікно StudentProfileWindow, якому передається обраний користувач. Там із репозиторію завантажуються всі сесії, обчислюється статистика, а дані під’єднуються до графіків LiveCharts, зокрема до діаграми точності:

```
AccuracyChart.Series = new[]
{
    new LineSeries<double>
```

```
    {
        Values = ordered.Select(s => s.Accuracy * 100).ToArray(),
        Fill = null
    }
};
```

Це дозволяє студенту та викладачу візуально оцінити динаміку прогресу.

4.2. Опис програмної реалізації

Процес розробки адаптивного тренажера з мови Java складався з кількох послідовних етапів: проєктування архітектури, створення бази даних, реалізації бізнес-логіки, розробки графічного інтерфейсу та інтеграційного тестування.

Структура та доступ до бази даних

Одним із ключових етапів розробки адаптивного тренажера стало проєктування структури бази даних та реалізація модулів для взаємодії з локальною SQLite. На цьому етапі було сформовано архітектурне рішення, яке забезпечує автономність роботи програмного забезпечення, мінімальну затримку при зверненні до таблиць та відсутність необхідності у зовнішньому сервері.

Для створення та підключення до бази було використано модуль Database. Такий підхід відповідає принципу Single Responsibility (SRP) та забезпечує централізацію логіки роботи з підключенням. Це важливо з кількох причин:

- уникаємо дублювання коду у кожному репозиторії;
- єдине місце для зміни параметрів підключення;
- легше тестувати компоненти, що працюють із БД.

```
namespace AdaptiveJavaTrainer.Data
{
    public class Database
    {
        private readonly string _connectionString =
            "Data Source=trainer.db;Cache=Shared";
```

```
        public SqliteConnection CreateConnection()
        {
            var c = new SqliteConnection(_connectionString);
            c.Open();
            return c;
        }
    }
}
```

Тут було використано Cache=Shared, це дозволяє декільком з'єднанням одночасно працювати з однією й тією ж файлу бази SQLite, що критично для застосунків, де послідовно виконуються десятки запитів під час тестування. Та прийнято рішення відкривати з'єднання всередині методу, що відкидає ситуацію, коли довго відкрите підключення блокує файл БД, та не потрібно вручну контролювати стан з'єднань.

Розглянемо структуру репозиторію QuestionRepository через нього здійснюється подальша робота з базою даних. Він отримує екземпляр Database через конструктор і виконує конкретні SQL-запити. Репозиторій створюється за принципами:

- інкапсульована взаємодія з базою;
- чітке розмежування даних і бізнес-логіки;
- спрощення читання коду в класах інтерфейсу.

```
public QuestionRepository(Database database)
{
    _database = database;
}
```

Завдяки цьому MainWindow не виконує жодних SQL запитів напряму - він тільки викликає методи репозиторію.

Далі розглянемо метод отримання списку тем — він демонструє загальний підхід, за яким побудовані всі операції читання БД:

```
public List<Topic> GetTopics()
{
    var topics = new List<Topic>();
```

```
    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();
    cmd.CommandText = @"
        SELECT Id, Code, Name, Description
        FROM Topics
        ORDER BY Name;
    ";

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        topics.Add(new Topic
        {
            Id = reader.GetInt32(0),
            Code = reader.GetString(1),
            Name = reader.GetString(2),
            Description = reader.IsDBNull(3) ? null : reader.GetString(3)
        });
    }

    return topics;
}
using var reader = cmd.ExecuteReader();
```

1) Створення з’єднання

```
using var connection = _database.CreateConnection();
```

"using var" гарантує автоматичне закриття з’єднання після завершення блока.

2) Створення SQL-команди

```
using var cmd = connection.CreateCommand();
```

cmd.CommandText зберігає SQL-запит. ORDER BY Name потрібен для стабільності відображення у UI. Виконання запиту

3) Виконання запиту

```
using var reader = cmd.ExecuteReader();
```

SQLite повертає результати через DataReader — найбільш ефективний спосіб читання множини рядків.

4) Мапінг записів у модель Topic

Метод reader.Read() читає один рядок за раз. Важливо:

- GetInt32(0), GetString(1)… прив’язані до порядку стовпців;
- Description може бути NULL → тому IsDBNull(3);

Цей патерн забезпечує:

- максимальну продуктивність;
- низьке споживання пам’яті;
- однозначне відображення структури таблиці.

Аналогічний підхід застосовано до завантаження питань, варіантів відповідей, сесій тестування, прогресу за темами тощо.

Завантаження питань

```
public List<Question> GetAllQuestions(int? topicId)
{
    var questions = new List<Question>();

    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    if (topicId.HasValue)
    {
        cmd.CommandText = @"
            SELECT Id, TopicId, Text, Difficulty, QuestionType, CodeSnippet
            FROM Questions
            WHERE TopicId = $topicId
            ORDER BY Difficulty;
        ";
        cmd.Parameters.AddWithValue("$topicId", topicId.Value);
    }
    else
    {
        cmd.CommandText = @"
            SELECT Id, TopicId, Text, Difficulty, QuestionType, CodeSnippet
            FROM Questions
            ORDER BY Difficulty;
        ";
    }

    using (var reader = cmd.ExecuteReader())
 {
        while (reader.Read())
        {
            var q = new Question
```

```
            {
                Id = reader.GetInt32(0),
                TopicId = reader.GetInt32(1),
                Text = reader.GetString(2),
                Difficulty = reader.GetInt32(3),
                QuestionType = reader.GetString(4)
                CodeSnippet = reader.IsDBNull(5) ? null : reader.GetString(5)
            };
            q.Text = q.Text
                .Replace("\\n", "\n")
                .Replace("\\t", "\t");

            if (!string.IsNullOrEmpty(q.CodeSnippet))
            }
                q.CodeSnippet = q.CodeSnippet
                    .Replace("\\n", "\n")
                    .Replace("\\t", "\t");
            }

            questions.Add(q);

        }

    foreach (var q in questions)
    {
        q.Answers = GetAnswersForQuestion(q.Id);
    }

    return questions;

}
```

Завантаження варіантів відповідей

```
public List<Answer> GetAnswersForQuestion(int questionId)
{
    var list = new List<Answer>();

    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    cmd.CommandText = @"
        SELECT Id, QuestionId, Text, IsCorrect
        FROM Answers
        WHERE QuestionId = $q";

    cmd.Parameters.AddWithValue("$q", questionId);

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        list.Add(new Answer
        {
            Id = reader.GetInt32(0),
```

```
            QuestionId = reader.GetInt32(1),
            Text = reader.GetString(2),
            IsCorrect = reader.GetBoolean(3)
        });
    }

    return list;
}
```

Сесії тестування

```
public int CreateTestSession(int userId, int? topicId, int questionLimit)
{
    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    cmd.CommandText = @"
        INSERT INTO TestSessions (UserId, TopicId, StartedAt, QuestionLimit,
AnsweredCount, CorrectCount)
        VALUES ($userId, $topicId, $startedAt, $limit, 0, 0);
        SELECT last_insert_rowid();
    ";

    cmd.Parameters.AddWithValue("$userId", userId);
    cmd.Parameters.AddWithValue("$topicId",
        topicId.HasValue ? topicId.Value : (object)DBNull.Value);
    cmd.Parameters.AddWithValue("$startedAt", DateTime.Now);
    cmd.Parameters.AddWithValue("$limit", questionLimit);

    var id = (long)cmd.ExecuteScalar();
    return (int)id;

}
```

Завантаження сесій в профіль

```
public List<TestSession> GetSessions(int userId)
{
    var list = new List<TestSession>();

    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();
    cmd.CommandText = @"
        SELECT Id, UserId, TopicId, StartedAt, FinishedAt,
               QuestionLimit, AnsweredCount, CorrectCount
        FROM TestSessions
        WHERE UserId = $userId
        ORDER BY StartedAt;
    ";
    cmd.Parameters.AddWithValue("$userId", userId);

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
```

```
    {
        list.Add(new TestSession
        {
            Id = reader.GetInt32(0),
            UserId = reader.GetInt32(1),
            TopicId = reader.IsDBNull(2) ? (int?)null : reader.GetInt32(2),
            StartedAt = DateTime.Parse(reader.GetString(3)),
            FinishedAt = reader.IsDBNull(4) ? null :
DateTime.Parse(reader.GetString(4)),
            QuestionLimit = reader.GetInt32(5),
            AnsweredCount = reader.GetInt32(6),
            CorrectCount = reader.GetInt32(7)
        });
    }

    return list;

}
```

Така структура дозволяє чітко розділити логіку зберігання даних та бізнес-логіку системи.

Реалізація бізнес логіки та адаптивного механізму

Бізнес-логіка тренажера зосереджена у вікні MainWindow та в класі репозиторію. Після завантаження всіх питань обирається поточний користувач, регулюється складність, створюється сесія тестування та запускається адаптивний механізм вибору наступного питання.

Завантаження всіх питань для обраної (або всіх) тем:

```
_allQuestions = _repo.GetAllQuestions(_currentTopicId);
if (_allQuestions.Count == 0)
{
    MessageBox.Show("Для обраної теми немає питань.");
    return;
}

_questionLimit = Math.Min(limit, _allQuestions.Count);

_asked.Clear();
_askedCount   = 0;
_answeredCount = 0;

_correctCount  = 0;


}
```

Вибір / створення поточного користувача:

```
_currentUser = _repo.GetOrCreateUserByName(userName);

LoadUsers();

  UserComboBox.Text = _currentUser.Name;
```

Тут репозиторій або знаходить User, або додає нового в таблицю Users.

Розрахунок стартової складності _difficulty на основі минулої статистики користувача по темі:

```
var stat = _repo.GetUserTopicProgress(_currentUser.Id, _currentTopicId);
if (stat != null && stat.TotalAnswered >= 5)
{
    double acc = (double)stat.CorrectAnswered / stat.TotalAnswered;
    _difficulty = acc switch
    {
        < 0.4 => 1,
        < 0.7 => 2,
        _     => 3
    };
}
else
{
    _difficulty = 1;
}


_userTopicStats = _repo.GetUserTopicsProgress(_currentUser.Id);

}
```

Тут же заповнюється _userTopicStats, який потім використовує адаптивний алгоритм.

Створення запису про сесію у таблиці TestSessions:

У MainWindow.xaml.cs. QuestionRepository

```
_sessionId = _repo.CreateTestSession(
    _currentUser.Id,
    _currentTopicId,
    _questionLimit

);
```

У QuestionRepository

```
public int CreateTestSession(int userId, int? topicId, int questionLimit)
{
    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    cmd.CommandText = @"
        INSERT INTO TestSessions (UserId, TopicId, StartedAt, QuestionLimit,
AnsweredCount, CorrectCount)
        VALUES ($userId, $topicId, $startedAt, $limit, 0, 0);
        SELECT last_insert_rowid();
    ";

    cmd.Parameters.AddWithValue("$userId", userId);
    cmd.Parameters.AddWithValue("$topicId",
        topicId.HasValue ? topicId.Value : (object)DBNull.Value);
    cmd.Parameters.AddWithValue("$startedAt", DateTime.Now);
    cmd.Parameters.AddWithValue("$limit", questionLimit);

    var id = (long)cmd.ExecuteScalar();
    return (int)id;
}
```

Запуск адаптивного механізму вибору наступного питання

Старт механізму — виклик SelectNextQuestion() після підготовки всіх параметрів та виклик UpdateStatus() у кінці StartButton_Click. Сам механізм вибору: SelectNextQuestion() + SelectAdaptiveAcrossTopics()

```
private void SelectNextQuestion()
{
    if (_askedCount >= _questionLimit)
    {
        FinishTest();
        return;
    }

    var remaining = _allQuestions
        .Where(q => !_asked.Contains(q.Id))
        .ToList();

    if (!remaining.Any())
    {
        FinishTest();
        return;
    }

    Question next;

    if (_currentTopicId != null)
    {
        // фіксована тема: підбір за складністю
        var candidates = remaining
            .Where(q => q.Difficulty == _difficulty)
            .ToList();

        if (!candidates.Any())
            candidates = remaining;

        next = candidates[_rand.Next(candidates.Count)];
    }
    else
    {
        // кілька тем: повністю адаптивний вибір
        next = SelectAdaptiveAcrossTopics(remaining);
    }

    _currentQuestion = next;
    _asked.Add(next.Id);
    _askedCount++;

    ShowQuestion();
    UpdateStatus();
}
```

```
private Question SelectAdaptiveAcrossTopics(List<Question> remaining)
{
    var topicsInRemaining = remaining
        .Select(q => q.TopicId)
        .Distinct()
        .ToList();

    double GetAccuracyForTopic(int topicId)
    {
        var s = _userTopicStats.FirstOrDefault(p => p.TopicId == topicId);
        if (s == null || s.TotalAnswered == 0)
        {
            return 0.5;
        }

        return (double)s.CorrectAnswered / s.TotalAnswered;
    }

    int weakestTopicId = topicsInRemaining
        .OrderBy(tid => GetAccuracyForTopic(tid))
        .First();

    var topicCandidates = remaining
        .Where(q => q.TopicId == weakestTopicId && q.Difficulty ==
_difficulty)
        .ToList();

    if (!topicCandidates.Any())
    {
        topicCandidates = remaining
            .Where(q => q.TopicId == weakestTopicId)
            .ToList();
    }

    if (!topicCandidates.Any())
    {
        topicCandidates = remaining;
    }

    return topicCandidates[_rand.Next(topicCandidates.Count)];

}
```

Ключовою частиною є метод SelectAdaptiveAcrossTopics, який працює із залишком питань та статистикою користувача по темах _userTopicStats:

Логіка методу така:

- обчислюється перелік тем, для яких ще залишилися питання;
- для кожної теми оцінюється точність користувача (CorrectAnswered / TotalAnswered);

- обирається «найслабша» тема (з найменшою точністю);
- з цієї теми беруться питання потрібної складності _difficulty;
- якщо таких немає – з цієї теми беруться будь-які питання;
- якщо і їх немає – вибір здійснюється з усіх доступних питань.

Таким чином, система не тільки враховує рівень складності, а й свідомо підсилює саме ті теми, з якими користувач має найбільші проблеми.

Формування статистики та PDF-звітів

Статистику та звіти реалізовано у вікні StudentProfileWindow. На основі сесій, отриманих з репозиторію, обчислюються показники точності по кожній темі та загальна успішність, а також готуються дані для графіків LiveCharts. Фрагмент побудови серії для діаграми точності виглядає так:

```
AccuracyChart.Series = new[]
{
    new LineSeries<double>
    {
        Values = ordered.Select(s => s.Accuracy * 100).ToArray(),
        Fill = null
    }
};
```

Для експорту результатів у формат PDF використовується бібліотека QuestPDF. В обробнику ExportPdf_Click створюється документ із заголовком, даними про студента, узагальненими показниками та таблицею сесій. Фрагмент опису сторінки:

```
var document = Document.Create(container =>
{
    container.Page(page =>
    {
        page.Size(PageSizes.A4);
        page.Margin(2, Unit.Centimetre);
        page.PageColor(Colors.White);
        page.DefaultTextStyle(x => x.FontSize(12));

        page.Content().Column(col =>
        {
            col.Spacing(6);
```

```
            col.Item().Text("Звіт про прогрес студента")
                .FontSize(18).Bold().AlignCenter();

            col.Item().Text($"Студент: {_user.Name}")
                .FontSize(14);

            col.Item().Text($"Дата генерації: {DateTime.Now:dd.MM.yyyy}")
                .FontSize(12);

            col.Item()
                .PaddingTop(10)
                .LineHorizontal(1);

            col.Item().Text($"Кількість сесій: {orderedSessions.Count}");
            col.Item().Text($"Усього відповідей: {totalAnswered}");
            col.Item().Text($"Правильних відповідей: {totalCorrect}");
            col.Item().Text($"Загальна точність: {totalAcc:0.0}%");

            col.Item().Text("Сесії тестування")
                .FontSize(14).Bold();
            // ... подальший опис таблиці
        });
    });
});
```

Таким чином, процес програмної реалізації включає проєктування та створення бази даних, кодування логіки адаптивного добору запитань, реалізацію інтерфейсу користувача, побудову графіків і формування звітів у форматі PDF на основі реальних даних.

4.3. Необхідна користувачу програми інструкція

Нижче наведено інструкцію для кінцевого користувача програми Adaptive Java Trainer.

Створення та вибір користувача

1. У головному вікні програми, див. рисунок 4.1. У полі вибору користувача відкрити список, див. рисунок 4.2.

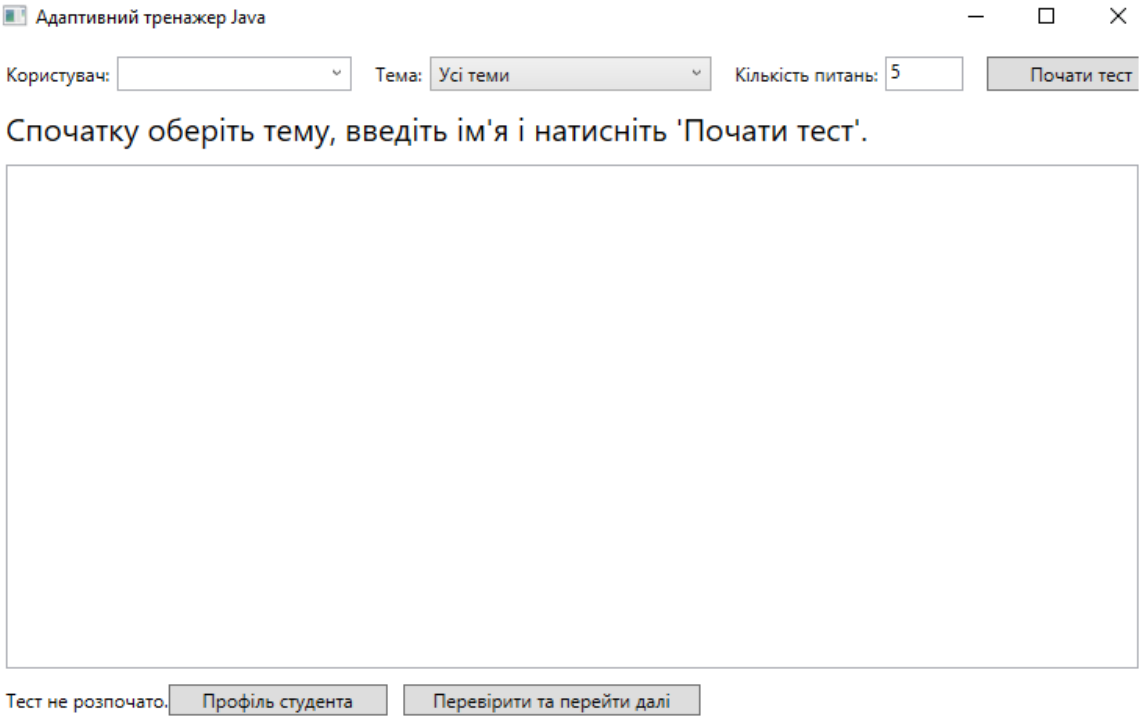

Рисунок 4.1 - Головне вікно програми

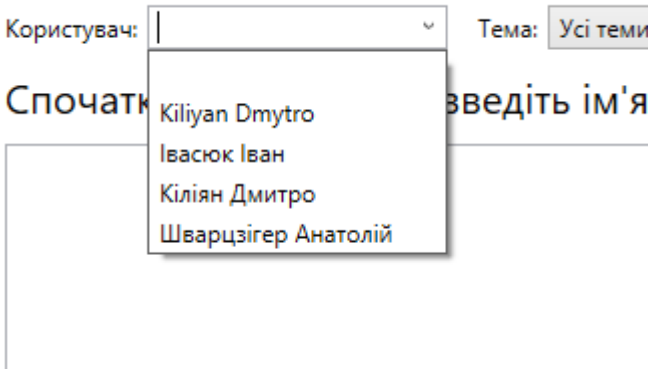

Рисунок 4.2 - Список користувачів

2. Якщо користувач уже існує - вибрати його зі списку або якщо користувач відсутній - вибрати опцію створення нового, ввести ім’я та зберегти.

Вибір тем і параметрів тесту

1. У списку тем, див. рисунок 4.3, відмітити одну або кілька тем, з яких потрібно пройти тестування.

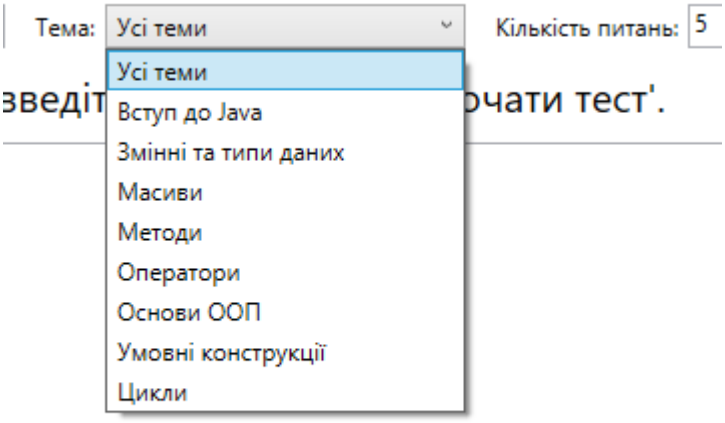

Рисунок 4.3 - Список тем

2. У полі ліміту питань, див. рисунок 4.4, вказати кількість завдань, які будуть задані в межах сесії.

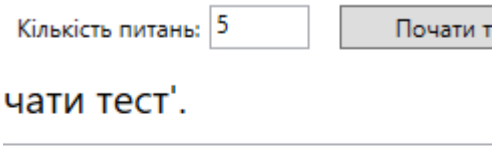

Рисунок 4.4 - Рядок Кількості завдань

3. Натиснути кнопку «Почати тест».

Проходження тесту

1. У вікні тестування, див. рисунок 4.5, уважно прочитати текст питання.

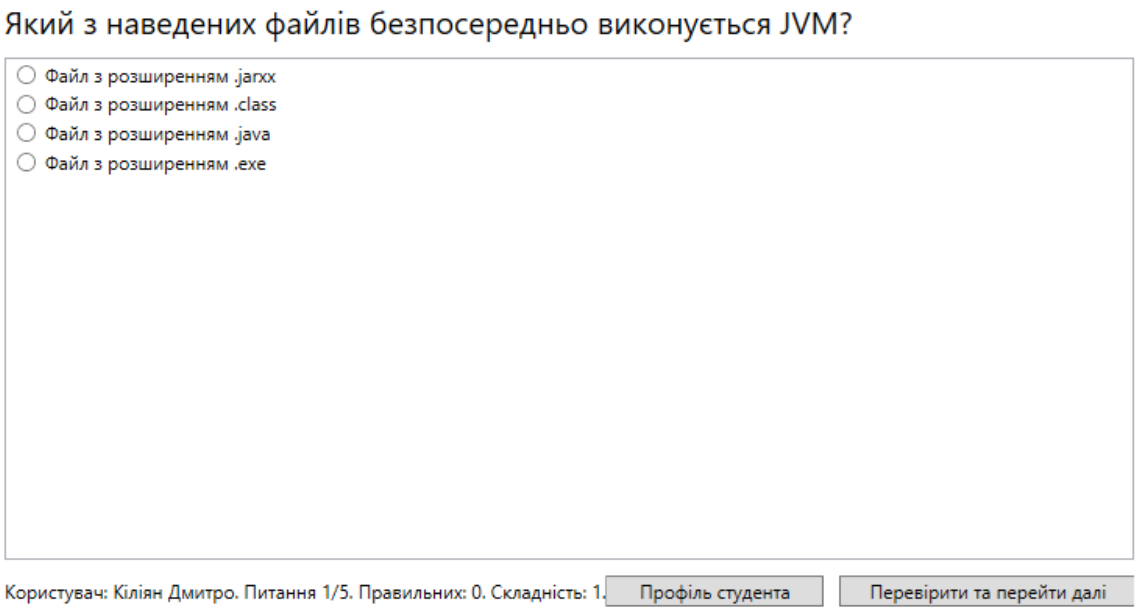

Рисунок 4.5 – Вікно тестування

2. Обрати один із варіантів відповіді у списку.
3. Натиснути кнопку підтвердження відповіді.
4. Після обробки відповіді система автоматично покаже наступне питання.
5. Після досягнення ліміту питань буде виведено підсумковий результат сесії.

Протягом тесту користувач не бачить, як змінюється внутрішній рівень складності, але саме він визначає підбір наступних питань.

Перегляд статистики та формування звіту

1. З головного вікна натиснути кнопку переходу до профілю студента, див. рисунок 4.6.

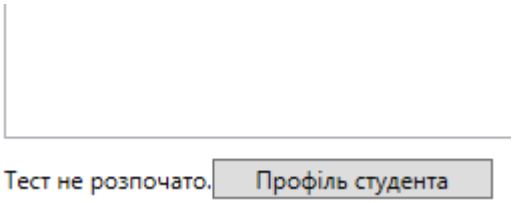

Рисунок 4.6 - Кнопка переходу до профілю студента

2. У вікні профілю, див. рисунок 4.7, переглянути графіки точності та таблицю сесій.

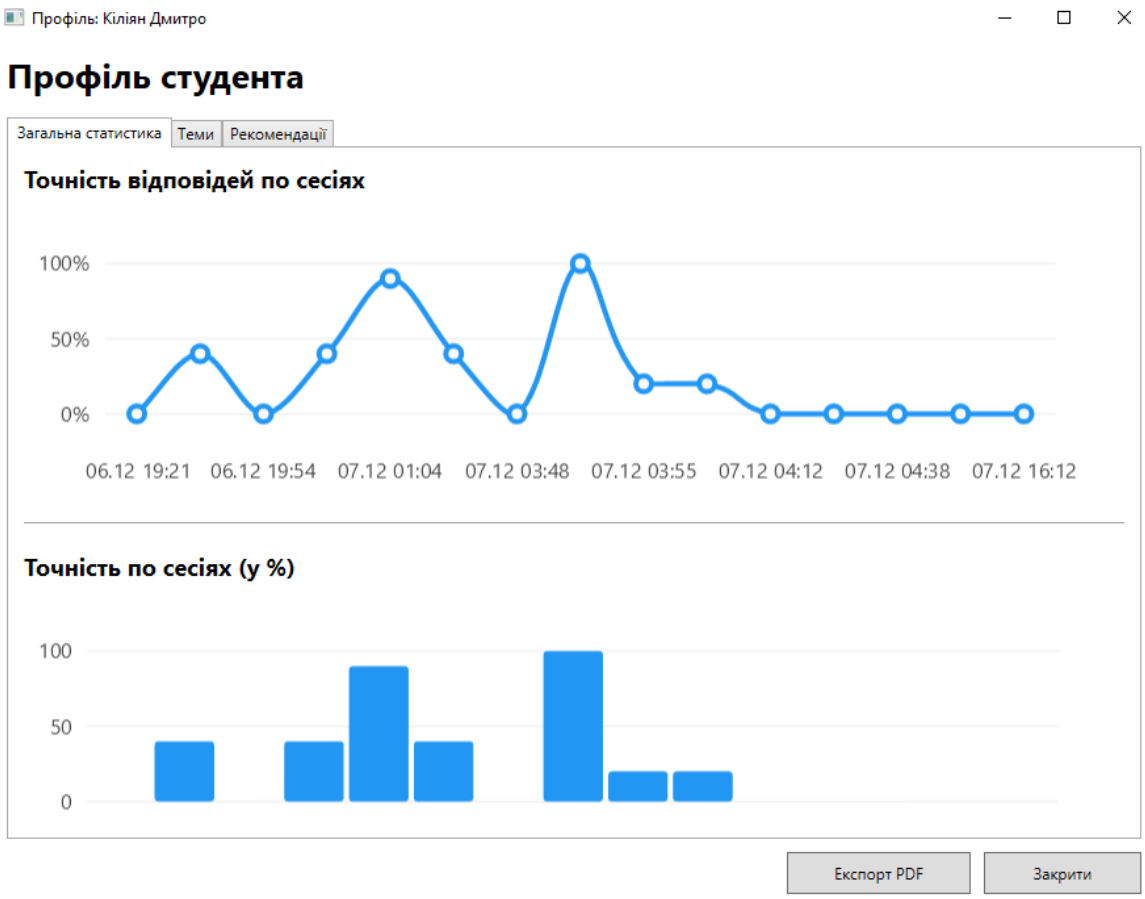

Рисунок 4.7 - Вікно профілю студента

3. Для створення PDF-звіту натиснути кнопку експорту.
4. У діалоговому вікні вибрати папку та ім’я файлу, підтвердити збереження.
5. Відкрити створений PDF-файл у будь-якому переглядачі документів.

Завершення роботи

Після завершення роботи з тренажером натиснути кнопку «Вихід» у головному вікні або закрити вікно стандартним способом.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено навчальне програмне забезпечення — адаптивний тренажер для вивчення основ мови програмування Java. Створення такого інструменту є важливим та актуальним завданням у умовах розвитку сучасного освітнього середовища, де дедалі частіше застосовуються індивідуалізовані та дистанційні форми навчання.

Розроблений тренажер забезпечує студентам можливість поетапного, адаптивного та систематичного опанування базових понять мови Java, а також формує навички програмування за допомогою режиму автоматичного підбору завдань відповідно до рівня підготовки користувача. Завдяки використанню локальної бази даних та алгоритмів аналізу відповідей, система створює персоналізовані траєкторії навчання, що сприяє ефективнішому засвоєнню матеріалу та підвищує мотивацію студентів.

Функціональні можливості програмного забезпечення — адаптивний добір тестів, збереження результатів, формування статистики, візуалізація прогресу та генерація PDF-звіту — дозволяють використовувати тренажер як для самостійної підготовки, так і як допоміжний інструмент у навчальному процесі.

У процесі розробки адаптивного навчального програмного забезпечення для вивчення «основ мови програмування Java» дисципліни «Програмування» було досягнуто ряд основних цілей проєкту:

- Розроблено модель представлення студента, яка зберігає інформацію про пройдений матеріал, рівень засвоєння тем і статистику відповідей.

- Створено адаптивний механізм добору завдань, що коригує рівень складності на основі результатів користувача.
- Спроєктовано структуру локальної бази даних, що зберігає користувачів, тем, запитань, відповідей і результатів тестування.
- Реалізовано програмний інтерфейс з використанням WPF для забезпечення зручної взаємодії зі студентом.
- Розроблено модуль статистики, який відображає прогрес студента, результати тестів і аналітичні графіки.
- Забезпечено можливість формування підсумкового PDF звіту за результатами тестування.

Сформовані такі змістові блоки, що відповідають початковому рівню вивчення мови:

- вступ до Java;
- змінні та типи даних;
- масиви;
- методи;
- оператори;
- основи ООП;
- умовні конструкції;
- цикли.

Розроблене програмне адаптивне навчальне забезпечення має зрозумілий для користувача інтерфейс, що має таку структуру:

- **Головне вікно:**
  - вибір користувача;
  - перехід до вибору тем і параметрів тестування.
- **Вікно вибору тем:**

- перелік доступних тем;
- вибір кількості запитань.

- **Вікно тестування:**

  - формулювання питання;
  - варіанти відповідей;
  - індикатор номера поточного запитання;
  - перехід до наступного адаптивно підібраного завдання.

- **Профіль користувача:**

  - історія тестових сесій;
  - графіки успішності за темами;
  - кнопка формування PDF-звіту.

Всі вимоги, описані в постановці задачі, були виконані.

СПИСОК ЛІТЕРАТУРИ

1. Codecademy. Learn Java [Електронний ресурс]. – Режим доступу: https://www.codecademy.com/learn/learn-java (дата звернення: 08.12.2025).
2. SoloLearn. Java Course [Електронний ресурс]. – Режим доступу: https://www.sololearn.com (дата звернення: 08.12.2025).
3. HackerRank. Java Practice [Електронний ресурс]. – Режим доступу: https://www.hackerrank.com (дата звернення: 08.12.2025).
4. LeetCode. Java Problems [Електронний ресурс]. – Режим доступу: https://leetcode.com (дата звернення: 08.12.2025).
5. W3Schools. Java Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/java (дата звернення: 08.12.2025).
6. Coursera. Java Programming Courses [Електронний ресурс]. – Режим доступу: https://www.coursera.org (дата звернення: 08.12.2025).
7. edX. Java Courses [Електронний ресурс]. – Режим доступу: https://www.edx.org (дата звернення: 08.12.2025).
8. Спірін О. М. Комп’ютерно орієнтовані системи навчання. – К.: НПУ ім. М. П. Драгоманова, 2009. – 216 с.
9. Биков В. Ю. Моделі організаційних систем відкритої освіти. – К.: Атіка, 2009. – 364 с.
10. Лорд Ф., Новік М. Статистичні теорії результатів психометричних тестів. – К.: Addison-Wesley (укр. вид.), 1968. – 585 с.
11. Корбетт А., Андерсон Дж. Відстеження знань у навчальних системах // User Modeling and User Adapted Interaction. – 1995. – 27 с.
12. Мандрика В. М., Олексійчук Ю. Ф. Тренажер з теми «1-R алгоритм» дисципліни «Комп’ютерний аналіз статистичних даних» // КНіПМ-2018. – Полтава: ПУЕТ, 2018. – С. 27–30.
13. Ярмоленко А. В., Олексійчук Ю. Ф. Алгоритм роботи тренажеру з теми «Асимптотичні оцінки функцій» дисципліни «Аналіз алгоритмів» // КНіПМ-2018, вип. 2. – Полтава: ПУЕТ, 2018. – С. 14–16.
14. Кривошей О. С., Олексійчук Ю. Ф. Програмна реалізація елементів тренажера з теми «Виявлення аномальних спостережень за допомогою критерію Томпсона» дисципліни «Аналіз алгоритмів і прикладні пакети статистичної обробки» // Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті: тези доповідей XLIII Міжнародної наукової студентської конференції за підсумками

науково-дослідних робіт студентів за 2019 рік (м. Полтава, 07–08 квітня 2020 року). Частина 2 – Полтава: ПУЕТ, 2020. – С. 123–125.
15. С.В. Гаркуша, О. В. Ольховська, О. О. Черненко. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня магістра / С.В. Гаркуша, О. В. Ольховська, О. О. Черненко. – Полтава : ПУЕТ, 2024. – 59 с.
16. Олексійчук Ю. Ф. Розробка та впровадження дистанційного курсу з дисципліни «Програмування» // Дистанційна освіта: забезпечення доступності та неперервної освіти впродовж життя (e-learning and university education-2017): Матеріали XLII МНМК. – Полтава: ПУЕТ, 2017. – С. 167–169.
17. Кіліян Д. В. Розробка навчального програмного забезпечення з теми «Collections. LinkedList» з дисципліни «Програмування» – Полтава: ПУЕТ, 2024. – 70 с.
18. Кіліян Д. В., Олексійчук Ю. Ф. Роль адаптивних навчальних програм у потенціалі дистанційної освіти // КНІТ-2025. – Полтава, 2025. – С. 30–32.

ДОДАТОК А.

App.xaml

```
<Application x:Class="AdaptiveJavaTrainer.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:AdaptiveJavaTrainer"
             StartupUri="MainWindow.xaml">
    <Application.Resources>


    </Application.Resources>
</Application>
```

AssemblyInfo.cs

```
using System.Windows;


[assembly: ThemeInfo(
    ResourceDictionaryLocation.None,
    ResourceDictionaryLocation.SourceAssembly
)]
```

MainWindow.xaml

```
<Window x:Class="AdaptiveJavaTrainer.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Адаптивний тренажер Java" Height="803" Width="770"
        WindowStartupLocation="CenterScreen">
```

```
    <Grid Margin="10">

        <Grid.RowDefinitions>

            <RowDefinition Height="Auto"/>

            <RowDefinition Height="Auto"/>

            <RowDefinition Height="*"/>

            <RowDefinition Height="Auto"/>

        </Grid.RowDefinitions>


        <StackPanel Orientation="Horizontal" Grid.Row="0" Margin="0 0 0 10"
VerticalAlignment="Center">


            <TextBlock Text="Користувач:" Margin="0 0 5 0"
VerticalAlignment="Center"/>


            <ComboBox x:Name="UserComboBox"

                Width="150"

                Margin="0 0 15 0"

                IsEditable="True"

                IsTextSearchEnabled="True"

                DisplayMemberPath="Name"

                StaysOpenOnEdit="True"/>


            <TextBlock Text="Тема:" Margin="0 0 5 0"
VerticalAlignment="Center"/>

            <ComboBox x:Name="TopicComboBox"

                      Width="180"

                      DisplayMemberPath="Name"

                      SelectedValuePath="Id"
```

```
                    Margin="0 0 15 0"/>


            <TextBlock Text="Кількість питань:" Margin="0 0 5 0"
VerticalAlignment="Center"/>

            <TextBox x:Name="QuestionCountTextBox" Width="50" Text="5"
Margin="0 0 15 0"/>



            <Button Content="Почати тест"

                    x:Name="StartButton"

                    Width="120"

                    Click="StartButton_Click"/>

        </StackPanel>



        <StackPanel Grid.Row="1"

            Margin="0 0 0 10">



            <TextBlock x:Name="QuestionTextBlock"

               FontSize="20"

               TextWrapping="Wrap"

               Text="Спочатку оберіть тему і натисніть 'Почати тест'." />



            <Border x:Name="CodeBorder"

            Margin="0,8,0,0"

            Padding="8"

            Background="#111111"

            CornerRadius="4"

            Visibility="Collapsed">

                <ScrollViewer HorizontalScrollBarVisibility="Auto">
```

```
                    <TextBlock x:Name="CodeTextBlock"
                        FontFamily="Consolas"
                        Foreground="White"
                        TextWrapping="NoWrap"
                        xml:space="preserve" />
                </ScrollViewer>
            </Border>
        </StackPanel>

        <ListBox x:Name="AnswersListBox"
                 Grid.Row="2"
                 Margin="0 0 0 10"
                 SelectionMode="Single">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <RadioButton Content="{Binding Text}"
                                 GroupName="Answers"
                                 IsChecked="{Binding
RelativeSource={RelativeSource AncestorType=ListBoxItem},
                                                     Path=IsSelected,
Mode=TwoWay}" />
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>

        <DockPanel Grid.Row="3">

            <TextBlock x:Name="StatusTextBlock"
```

```
                    Text="Тест не розпочато."
                    VerticalAlignment="Center"
                    DockPanel.Dock="Left"/>

            <StackPanel Orientation="Horizontal" DockPanel.Dock="Right">

                <Button Content="Профіль студента"
                        x:Name="ProfileButton"
                        Width="140"
                        Margin="0 0 10 0"
                        Click="ProfileButton_Click"/>

                <Button Content="Перевірити та перейти далі"
                        x:Name="NextButton"
                        Width="190"
                        Click="NextButton_Click"/>
            </StackPanel>

        </DockPanel>

    </Grid>
</Window>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
```

```
using AdaptiveJavaTrainer.Data;
using AdaptiveJavaTrainer.Models;

namespace AdaptiveJavaTrainer
{
    public partial class MainWindow : Window
    {
        private readonly QuestionRepository _repo;

        private List<Topic> _topics = new();
        private List<Question> _allQuestions = new();
        private HashSet<int> _asked = new();
        private Question? _currentQuestion;

        private User? _currentUser;
        private int? _currentTopicId;
        private int? _sessionId;

        private int _difficulty = 1;
        private int _questionLimit = 0;
        private int _askedCount = 0;
        private int _answeredCount = 0;
        private int _correctCount = 0;

        private List<UserTopicProgress> _userTopicStats = new();

        private readonly Random _rand = new();
```

```
public MainWindow()
{
    InitializeComponent();
    _repo = new QuestionRepository(new Database());

    LoadTopics();
    LoadUsers();
    ResetUI();
}

private void LoadTopics()
{
    try
    {
        _topics = _repo.GetTopics();

        TopicComboBox.Items.Clear();

        TopicComboBox.Items.Add(new Topic
        {
            Id = -1,
            Name = "Усі теми",
            Code = "all"
        });

        foreach (var t in _topics)
```

```
                    TopicComboBox.Items.Add(t);


                TopicComboBox.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка завантаження тем: " + ex.Message);
            }
        }

        private void LoadUsers()
        {
            try
            {
                var users = _repo.GetAllUsers();
                UserComboBox.ItemsSource = users;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка завантаження користувачів: " +
ex.Message);
            }
        }

        private void StartButton_Click(object sender, RoutedEventArgs e)
        {
            string userName = UserComboBox.Text?.Trim() ?? "";
            if (string.IsNullOrWhiteSpace(userName))
```

```
            {
                MessageBox.Show("Введіть ім'я користувача.");
                return;
            }

            if (!int.TryParse(QuestionCountTextBox.Text, out int limit) ||
limit <= 0)
            {
                MessageBox.Show("Введіть коректну кількість питань.");
                return;
            }

            var topic = TopicComboBox.SelectedItem as Topic;
            _currentTopicId = (topic != null && topic.Id != -1) ? topic.Id :
null;

            try
            {
                _currentUser = _repo.GetOrCreateUserByName(userName);

                LoadUsers();
                UserComboBox.Text = _currentUser.Name;

                _allQuestions = _repo.GetAllQuestions(_currentTopicId);
                if (_allQuestions.Count == 0)
                {
                    MessageBox.Show("Для обраної теми немає питань.");
                    return;
```

```
                }

                _questionLimit = Math.Min(limit, _allQuestions.Count);

                _asked.Clear();
                _askedCount = 0;
                _answeredCount = 0;
                _correctCount = 0;

                var stat = _repo.GetUserTopicProgress(_currentUser.Id,
_currentTopicId);
                if (stat != null && stat.TotalAnswered >= 5)
                {
                    double acc = (double)stat.CorrectAnswered /
stat.TotalAnswered;
                    _difficulty = acc switch
                    {
                        < 0.4 => 1,
                        < 0.7 => 2,
                        _ => 3
                    };
                }
                else
                {
                    _difficulty = 1;
                }

                _userTopicStats =
_repo.GetUserTopicsProgress(_currentUser.Id);
```

```
                _sessionId = _repo.CreateTestSession(_currentUser.Id,
_currentTopicId, _questionLimit);



                SelectNextQuestion();

                UpdateStatus();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Помилка підготовки тесту: " + ex.Message);

            }

        }



        private void SelectNextQuestion()

        {

            if (_askedCount >= _questionLimit)

            {

                FinishTest();

                return;

            }



            var remaining = _allQuestions

                .Where(q => !_asked.Contains(q.Id))

                .ToList();



            if (!remaining.Any())

            {

                FinishTest();
```

```
    return;
}

Question next;

if (_currentTopicId != null)
{
    var candidates = remaining
        .Where(q => q.Difficulty == _difficulty)
        .ToList();

    if (!candidates.Any())
        candidates = remaining;

    next = candidates[_rand.Next(candidates.Count)];
}
else
{
    next = SelectAdaptiveAcrossTopics(remaining);
}

_currentQuestion = next;
_asked.Add(next.Id);
_askedCount++;

ShowQuestion();
UpdateStatus();
```

```
        }

        private Question SelectAdaptiveAcrossTopics(List<Question> remaining)
        {
            var topicsInRemaining = remaining
                .Select(q => q.TopicId)
                .Distinct()
                .ToList();

            double GetAccuracyForTopic(int topicId)
            {
                var s = _userTopicStats.FirstOrDefault(p => p.TopicId ==
topicId);
                if (s == null || s.TotalAnswered == 0)
                {
                    return 0.5;
                }

                return (double)s.CorrectAnswered / s.TotalAnswered;
            }

            int weakestTopicId = topicsInRemaining
                .OrderBy(tid => GetAccuracyForTopic(tid))
                .First();

            var topicCandidates = remaining
                .Where(q => q.TopicId == weakestTopicId && q.Difficulty ==
_difficulty)
```

```
            .ToList();

        if (!topicCandidates.Any())
        {
            topicCandidates = remaining
                .Where(q => q.TopicId == weakestTopicId)
                .ToList();
        }

        if (!topicCandidates.Any())
        {
            topicCandidates = remaining;
        }

        return topicCandidates[_rand.Next(topicCandidates.Count)];
    }

    private static void Shuffle<T>(IList<T> list)
    {
        var rnd = new Random();
        for (int i = list.Count - 1; i > 0; i--)
        {
            int j = rnd.Next(i + 1);
            (list[i], list[j]) = (list[j], list[i]);
        }
    }
```

```
private void ShowQuestion()
{
    if (_currentQuestion == null)
    {
        QuestionTextBlock.Text = "Питання відсутнє.";
        CodeTextBlock.Text = string.Empty;
        CodeBorder.Visibility = Visibility.Collapsed;
        AnswersListBox.ItemsSource = null;
        return;
    }

    QuestionTextBlock.Text = _currentQuestion.Text;

    if (!string.IsNullOrWhiteSpace(_currentQuestion.CodeSnippet))
    {
        CodeTextBlock.Text = _currentQuestion.CodeSnippet;
        CodeBorder.Visibility = Visibility.Visible;
    }
    else
    {
        CodeTextBlock.Text = string.Empty;
        CodeBorder.Visibility = Visibility.Collapsed;
    }

    var shuffledAnswers = _currentQuestion.Answers
        .Select(a => new Answer
        {
```

```
            Id = a.Id,
            QuestionId = a.QuestionId,
            Text = a.Text.Replace("\\n", "\n"),
            IsCorrect = a.IsCorrect
        })
        .OrderBy(a => _rand.Next())
        .ToList();

    AnswersListBox.ItemsSource = shuffledAnswers;
    AnswersListBox.SelectedItem = null;
}

private void NextButton_Click(object sender, RoutedEventArgs e)
{
    if (_currentQuestion == null)
    {
        MessageBox.Show("Спочатку почніть тест.");
        return;
    }

    var ans = AnswersListBox.SelectedItem as Answer;
    if (ans == null)
    {
        MessageBox.Show("Оберіть відповідь.");
        return;
    }
```

```
bool correct = ans.IsCorrect;
int difficultyBefore = _difficulty;

_answeredCount++;

try
{
    if (correct)
    {
        _correctCount++;
        if (_difficulty < 3) _difficulty++;
    }
    else
    {
        if (_difficulty > 1) _difficulty--;
    }
}
catch (Exception ex)
{
    MessageBox.Show("Помилка логіки: " + ex.Message);
}

if (_sessionId.HasValue)
{
    try
    {
        _repo.LogQuestionResult(
```

```
                _sessionId.Value,
                _currentQuestion.Id,
                difficultyBefore,
                correct,
                null
            );
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка запису в базу: " + ex.Message);
        }
    }

    UpdateStatus();

    if (_answeredCount >= _questionLimit)
    {
        FinishTest();
        return;
    }

    SelectNextQuestion();
}

private void FinishTest()
{
    try
```

```
            {
                if (_currentUser != null)
                {
                    _repo.SaveOrUpdateUserTopicProgress(
                        _currentUser.Id,
                        _currentTopicId,
                        _answeredCount,
                        _correctCount,
                        _difficulty
                    );
                }

                if (_sessionId.HasValue)
                {
                    _repo.UpdateTestSessionSummary(
                        _sessionId.Value,
                        _answeredCount,
                        _correctCount
                    );
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Не вдалося зберегти прогрес: " +
ex.Message);
            }

            MessageBox.Show(
```

```
                $"Тест завершено!\n" +
                $"Користувач: {_currentUser?.Name}\n" +
                $"Питань: {_askedCount}\n" +
                $"Правильних: {_correctCount}",
                "Результат");

            ResetUI();
        }

        private void ResetUI()
        {
            QuestionTextBlock.Text =
                "Спочатку оберіть тему, введіть ім'я і натисніть 'Почати тест'.";

            CodeTextBlock.Text = string.Empty;
            CodeBorder.Visibility = Visibility.Collapsed;

            AnswersListBox.ItemsSource = null;

            _currentQuestion = null;
            _asked.Clear();
            _sessionId = null;

            _difficulty = 1;
            _askedCount = 0;
            _answeredCount = 0;
            _correctCount = 0;
```

```
    UpdateStatus();
}

private void UpdateStatus()
{
    if (_currentUser == null)
    {
        StatusTextBlock.Text = "Тест не розпочато.";
        return;
    }

    StatusTextBlock.Text =
        $"Користувач: {_currentUser.Name}. " +
        $"Питання {_askedCount}/{_questionLimit}. " +
        $"Правильних: {_correctCount}. " +
        $"Складність: {_difficulty}.";
}

private void ProfileButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        string name = UserComboBox.Text?.Trim() ?? "";

        if (string.IsNullOrWhiteSpace(name))
        {
```

```
                MessageBox.Show("Введіть або оберіть ім'я користувача.");
                return;
            }

            User user = _currentUser ??
_repo.GetOrCreateUserByName(name);

            LoadUsers();
            UserComboBox.Text = user.Name;

            var window = new StudentProfileWindow(_repo, user);
            window.Owner = this;
            window.ShowDialog();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Не вдалося відкрити профіль: " +
ex.Message);
        }
    }
}
```

StudentProfileWindow.xaml

```
<Window x:Class="AdaptiveJavaTrainer.StudentProfileWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:lvc="http://livecharts.com"
```

```
        xmlns:skia="clr-
namespace:LiveChartsCore.SkiaSharpView.WPF;assembly=LiveChartsCore.SkiaSharpV
iew.WPF"

        Title="Профіль студента"

        Width="900" Height="700"

        WindowStartupLocation="CenterOwner">



    <Grid Margin="10">

        <Grid.RowDefinitions>

            <RowDefinition Height="Auto"/>

            <RowDefinition Height="*"/>

            <RowDefinition Height="Auto"/>

        </Grid.RowDefinitions>



        <TextBlock Text="Профіль студента"

                   FontSize="26"

                   FontWeight="Bold"

                   Margin="0 0 0 15"/>



        <TabControl Grid.Row="1">



            <TabItem Header="Загальна статистика">

                <StackPanel Margin="10">



                    <TextBlock Text="Точність відповідей по сесіях"

                               FontSize="18" FontWeight="Bold" Margin="0 0 0
10"/>



                    <skia:CartesianChart x:Name="AccuracyChart"
```

```
                                    Height="220" />

                    <Separator Margin="0 20 0 20"/>

                    <TextBlock Text="Точність по сесіях (у %)"
                               FontSize="18" FontWeight="Bold" Margin="0 0 0
10"/>

                    <skia:CartesianChart x:Name="DifficultyChart"
                                         Height="220" />

                </StackPanel>
            </TabItem>

            <TabItem Header="Теми">
                <StackPanel Margin="10">

                    <TextBlock Text="Правильність відповідей по темах"
                               FontSize="18" FontWeight="Bold" Margin="0 0 0
10"/>

                    <skia:PieChart x:Name="TopicsPieChart"
                                   Height="260"/>

                    <TextBlock x:Name="TopicsTextSummary"
                               FontSize="14" Margin="0 20 0 0"
                               TextWrapping="Wrap" />
```

```
            </StackPanel>
        </TabItem>

        <TabItem Header="Рекомендації">
            <ScrollViewer Margin="10">
                <TextBlock x:Name="RecommendationsText"
                           FontSize="16"
                           TextWrapping="Wrap"/>
            </ScrollViewer>
        </TabItem>

    </TabControl>

    <StackPanel Grid.Row="2"
                Orientation="Horizontal"
                HorizontalAlignment="Right"
                Margin="0 10 0 0">

        <Button Content="Експорт PDF"
                Width="140" Height="32"
                Margin="0 0 10 0"
                Click="ExportPdf_Click"/>

        <Button Content="Закрити"
                Width="120" Height="32"
                Click="Close_Click"/>
    </StackPanel>
```

```
    </Grid>
</Window>
```

StudentProfileWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using AdaptiveJavaTrainer.Data;
using AdaptiveJavaTrainer.Models;

using LiveChartsCore;
using LiveChartsCore.SkiaSharpView;

// QuestPDF
using QuestPDF.Fluent;
using QuestPDF.Helpers;
using QuestPDF.Infrastructure;

// SaveFileDialog
using Microsoft.Win32;

namespace AdaptiveJavaTrainer
{
    public partial class StudentProfileWindow : Window
    {
        private readonly QuestionRepository _repo;
```

```
private readonly User _user;

private List<TestSession> _sessions = new();
private List<TopicSummary> _topicStats = new();
private string _recommendations = "";

public StudentProfileWindow(QuestionRepository repo, User user)
{
    QuestPDF.Settings.License = LicenseType.Community;

    InitializeComponent();

    _repo = repo;
    _user = user;

    Title = $"Профіль: {_user.Name}";

    LoadCharts();
}

private void LoadCharts()
{
    var sessions = _repo.GetSessions(_user.Id);
    var labels = sessions
        .Select(s => s.StartedAt.ToString("dd.MM HH:mm"))
        .ToList();
```

```
            _sessions = sessions;

            int step = Math.Max(1, labels.Count / 5);

            if (sessions.Count == 0)
            {
                RecommendationsText.Text = "Дані відсутні. Пройдіть хоча б
один тест.";
                return;
            }

            var ordered = sessions.OrderBy(s => s.StartedAt).ToList();

            AccuracyChart.Series = new[]
            {
                new LineSeries<double>
                {
                    Values = ordered.Select(s => s.Accuracy * 100).ToArray(),
                    Fill = null
                }
            };

            AccuracyChart.XAxes = new[]
            {
                new Axis
                {
                    Labels = labels,
                    MinStep = step,
```

```
        LabelsRotation = 0
    }
};

AccuracyChart.YAxes = new[]
{
    new Axis
    {
        Labeler = v => v + "%"
    }
};

DifficultyChart.Series = new[]
{
    new ColumnSeries<double>
    {
        Values = ordered
            .Select(s => s.AnsweredCount == 0
                ? 0
                : (double)s.CorrectCount / s.AnsweredCount * 100)
            .ToArray()
    }
};

DifficultyChart.XAxes = new[]
{
    new Axis
```

```
                {
                    Labels = ordered.Select(s =>
s.StartedAt.ToShortDateString()).ToArray()
                }
            };

            var topicStats = CalculateTopicStats(sessions);
            _topicStats = topicStats;

            TopicsPieChart.Series = topicStats
                .Select(t => new PieSeries<double>
                {
                    Name = t.TopicName,
                    Values = new[] { t.Accuracy }
                })
                .ToArray();

            TopicsTextSummary.Text = string.Join("\n",
                topicStats.Select(t => $"{t.TopicName}: {t.Accuracy:0}%
правильних"));

            _recommendations = BuildRecommendations(topicStats);
            RecommendationsText.Text = _recommendations;
        }

        private class TopicSummary
        {
            public int? TopicId { get; set; }
```

```
        public string TopicName { get; set; } = "";
        public double Accuracy { get; set; }
    }

    private List<TopicSummary> CalculateTopicStats(List<TestSession>
sessions)
    {
        var result = new List<TopicSummary>();
        var topics = _repo.GetTopics();

        var grouped = sessions.GroupBy(s => s.TopicId);

        foreach (var g in grouped)
        {
            int? topicId = g.Key;

            string topicName =
                topicId == null
                    ? "Усі теми"
                    : topics.FirstOrDefault(t => t.Id == topicId)?.Name
                      ?? $"Тема #{topicId}";

            int totalCorrect = g.Sum(x => x.CorrectCount);
            int totalAnswered = g.Sum(x => x.AnsweredCount);

            double acc = totalAnswered == 0
                ? 0
                : (double)totalCorrect / totalAnswered * 100;
```

```
            result.Add(new TopicSummary
            {
                TopicId = topicId,
                TopicName = topicName,
                Accuracy = acc
            });
        }

        return result;
    }

    private static string BuildRecommendations(List<TopicSummary> stats)
    {
        var weak = stats.Where(s => s.Accuracy < 60).ToList();
        var strong = stats.Where(s => s.Accuracy > 85).ToList();

        string text = "";

        if (weak.Count > 0)
        {
            text += "❗ Рекомендуємо попрацювати над темами:\n";
            foreach (var w in weak)
                text += $" • {w.TopicName} (точність {w.Accuracy:0}%)\n";
            text += "\n";
        }
```

```
            if (strong.Count > 0)
            {
                text += "✓ Ви добре засвоїли:\n";
                foreach (var s in strong)
                    text += $" • {s.TopicName} ({s.Accuracy:0}%)\n";
            }

            if (text == "")
                text = "Недостатньо даних для рекомендацій.";

            return text;
        }

        private void ExportPdf_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (_sessions == null || _sessions.Count == 0)
                {
                    MessageBox.Show("Немає даних для експорту. Спочатку
пройдіть хоча б одну сесію.",
                        "Експорт PDF", MessageBoxButton.OK,
MessageBoxImage.Information);
                    return;
                }

                var dialog = new SaveFileDialog
                {
```

```
                FileName = $"Звіт_{_user.Name}_{DateTime.Now:yyyy-MM-
dd}.pdf",
                Filter = "PDF файл (*.pdf)|*.pdf"
            };

            if (dialog.ShowDialog() != true)
                return;

            string filePath = dialog.FileName;

            int totalAnswered = _sessions.Sum(s => s.AnsweredCount);
            int totalCorrect = _sessions.Sum(s => s.CorrectCount);
            double totalAcc = totalAnswered == 0
                ? 0
                : (double)totalCorrect / totalAnswered * 100;

            var orderedSessions = _sessions.OrderBy(s =>
s.StartedAt).ToList();

            var document = Document.Create(container =>
            {
                container.Page(page =>
                {
                    page.Size(PageSizes.A4);
                    page.Margin(2, Unit.Centimetre);
                    page.PageColor(Colors.White);
                    page.DefaultTextStyle(x => x.FontSize(12));
```

```
                        page.Content().Column(col =>
                        {
                            col.Spacing(6);

                            col.Item().Text("Звіт про прогрес студента")
                                .FontSize(18).Bold().AlignCenter();

                            col.Item().Text($"Студент: {_user.Name}")
                                .FontSize(14);

                            col.Item().Text($"Дата генерації:
{DateTime.Now:dd.MM.yyyy}")
                                .FontSize(12);

                            col.Item()
                                .PaddingTop(10)
                                .LineHorizontal(1);

                            col.Item().Text($"Кількість сесій:
{orderedSessions.Count}");
                            col.Item().Text($"Усього відповідей:
{totalAnswered}");
                            col.Item().Text($"Правильних відповідей:
{totalCorrect}");
                            col.Item().Text($"Загальна точність:
{totalAcc:0.0}%");

                            col.Item().Text("Сесії тестування")
                                .FontSize(14).Bold();
```

```
                col.Item().Table(table =>
                {
                    table.ColumnsDefinition(columns =>
                    {
                        columns.ConstantColumn(25);
                        columns.RelativeColumn();
                        columns.ConstantColumn(60);
                        columns.ConstantColumn(60);
                        columns.ConstantColumn(60);
                    });

                    table.Header(header =>
                    {
header.Cell().Element(HeaderCell).Text("#");
header.Cell().Element(HeaderCell).Text("Дата");
header.Cell().Element(HeaderCell).Text("Ліміт");
header.Cell().Element(HeaderCell).Text("Відповіли");
header.Cell().Element(HeaderCell).Text("Точність");
                    });

                    int i = 1;
                    foreach (var s in orderedSessions)
                    {
                        double acc = s.AnsweredCount == 0
                            ? 0
```

```
                                        : (double)s.CorrectCount /
s.AnsweredCount * 100;


table.Cell().Element(BodyCell).Text(i.ToString());
                                    table.Cell().Element(BodyCell)
.Text(s.StartedAt.ToString("dd.MM.yyyy HH:mm"));
table.Cell().Element(BodyCell).Text(s.QuestionLimit.ToString());
table.Cell().Element(BodyCell).Text(s.AnsweredCount.ToString());
table.Cell().Element(BodyCell).Text($"{acc:0.0}%");

                                    i++;
                                }

                                static IContainer HeaderCell(IContainer
container) =>
                                    container.DefaultTextStyle(x =>
x.SemiBold())
                                             .PaddingVertical(2)
                                             .BorderBottom(1)
.BorderColor(Colors.Grey.Medium);

                                static IContainer BodyCell(IContainer
container) =>
                                    container.PaddingVertical(2);
                            });
```

```
                                if (_topicStats.Count > 0)
                                {
                                    col.Item().Height(10);

                                    col.Item().Text("Точність по темах")
                                        .FontSize(14).Bold();

                                    col.Item().Table(table =>
                                    {
                                        table.ColumnsDefinition(columns =>
                                        {
                                            columns.RelativeColumn();
                                            columns.ConstantColumn(70);
                                        });

                                        table.Header(header =>
                                        {
header.Cell().Element(HeaderCell).Text("Тема");
header.Cell().Element(HeaderCell).Text("Точність");
                                        });

                                        foreach (var t in _topicStats)
                                        {
table.Cell().Element(BodyCell).Text(t.TopicName);
table.Cell().Element(BodyCell).Text($"{t.Accuracy:0.0}%");
```

```
                                    }

                                    static IContainer HeaderCell(IContainer
container) =>
                                        container.DefaultTextStyle(x =>
x.SemiBold())
                                                .PaddingVertical(2)
                                                .BorderBottom(1)

.BorderColor(Colors.Grey.Medium);

                                    static IContainer BodyCell(IContainer
container) =>
                                        container.PaddingVertical(2);
                                });
                            }

                            if (!string.IsNullOrWhiteSpace(_recommendations))
                            {
                                col.Item().Height(10);

                                col.Item().Text("Рекомендації")
                                    .FontSize(14).Bold();

                                col.Item().Text(_recommendations);
                            }
                        });
                    });
                });
```

```
                document.GeneratePdf(filePath);


                MessageBox.Show("PDF-звіт успішно збережено.",
                    "Експорт PDF", MessageBoxButton.OK,
MessageBoxImage.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка експорту PDF: " + ex.Message,
                    "Експорт PDF", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }
    }
}
```

Database.cs

```
using Microsoft.Data.Sqlite;

namespace AdaptiveJavaTrainer.Data
{
    public class Database
    {
```

```
        private readonly string _connectionString =
            "Data Source=trainer.db;Cache=Shared";

        public SqliteConnection CreateConnection()
        {
            var c = new SqliteConnection(_connectionString);
            c.Open();
            return c;
        }
    }
}
```

QuestionRepository.cs

```
using System;
using System.Collections.Generic;
using Microsoft.Data.Sqlite;
using AdaptiveJavaTrainer.Models;

namespace AdaptiveJavaTrainer.Data
{
    public class QuestionRepository
    {
        private readonly Database _database;

        public QuestionRepository(Database database)
        {
            _database = database;
        }
```

```
public List<User> GetAllUsers()
{
    var list = new List<User>();

    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    cmd.CommandText = "SELECT Id, Name FROM Users ORDER BY Name;";

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        list.Add(new User
        {
            Id = reader.GetInt32(0),
            Name = reader.GetString(1)
        });
    }

    return list;
}

public List<Topic> GetTopics()
{
    var topics = new List<Topic>();
```

```
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();
            cmd.CommandText = @"
                SELECT Id, Code, Name, Description
                FROM Topics
                ORDER BY Name;
            ";

            using var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                topics.Add(new Topic
                {
                    Id = reader.GetInt32(0),
                    Code = reader.GetString(1),
                    Name = reader.GetString(2),
                    Description = reader.IsDBNull(3) ? null :
reader.GetString(3)
                });
            }

            return topics;
        }

        public List<Question> GetAllQuestions(int? topicId)
        {
            var questions = new List<Question>();
```

```
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();

            if (topicId.HasValue)
            {
                cmd.CommandText = @"
                    SELECT Id, TopicId, Text, Difficulty, QuestionType,
CodeSnippet
                    FROM Questions
                    WHERE TopicId = $topicId
                    ORDER BY Difficulty;
                ";
                cmd.Parameters.AddWithValue("$topicId", topicId.Value);
            }
            else
            {
                cmd.CommandText = @"
                    SELECT Id, TopicId, Text, Difficulty, QuestionType,
CodeSnippet
                    FROM Questions
                    ORDER BY Difficulty;
                ";
            }

            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
```

```
                {
                    var q = new Question
                    {
                        Id = reader.GetInt32(0),
                        TopicId = reader.GetInt32(1),
                        Text = reader.GetString(2),
                        Difficulty = reader.GetInt32(3),
                        QuestionType = reader.GetString(4),
                        CodeSnippet = reader.IsDBNull(5) ? null :
reader.GetString(5)
                    };

                    q.Text = q.Text
                        .Replace("\\n", "\n")
                        .Replace("\\t", "\t");

                    if (!string.IsNullOrEmpty(q.CodeSnippet))
                    {
                        q.CodeSnippet = q.CodeSnippet
                            .Replace("\\n", "\n")
                            .Replace("\\t", "\t");
                    }

                    questions.Add(q);
                }
            }

            foreach (var q in questions)
```

```
    {
        q.Answers = GetAnswersForQuestion(q.Id);
    }

    return questions;
}

public List<Answer> GetAnswersForQuestion(int questionId)
{
    var list = new List<Answer>();

    using var connection = _database.CreateConnection();
    using var cmd = connection.CreateCommand();

    cmd.CommandText = @"
SELECT Id, QuestionId, Text, IsCorrect
FROM Answers
WHERE QuestionId = $q";

    cmd.Parameters.AddWithValue("$q", questionId);

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        list.Add(new Answer
        {
            Id = reader.GetInt32(0),
```

```
            QuestionId = reader.GetInt32(1),
            Text = reader.GetString(2),
            IsCorrect = reader.GetInt32(3) == 1
        });
    }

    return list;
}

public User GetOrCreateUserByName(string name)
{
    using var connection = _database.CreateConnection();

    using (var select = connection.CreateCommand())
    {
        select.CommandText = @"
            SELECT Id, Name
            FROM Users
            WHERE Name = $name
            LIMIT 1;
        ";
        select.Parameters.AddWithValue("$name", name);

        using var reader = select.ExecuteReader();
        if (reader.Read())
        {
            return new User
```

```
                {
                    Id = reader.GetInt32(0),
                    Name = reader.GetString(1)
                };
            }
        }

        using (var insert = connection.CreateCommand())
        {
            insert.CommandText = @"
                INSERT INTO Users (Name)
                VALUES ($name);
                SELECT last_insert_rowid();
            ";
            insert.Parameters.AddWithValue("$name", name);
            var newId = (long)insert.ExecuteScalar();

            return new User
            {
                Id = (int)newId,
                Name = name
            };
        }
    }

    public UserTopicProgress? GetUserTopicProgress(int userId, int?
topicId)
    {
```

```
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();

            cmd.CommandText = @"
                SELECT Id, UserId, TopicId, TotalAnswered, CorrectAnswered,
LastDifficulty
                FROM UserTopicProgress
                WHERE UserId = $userId
                  AND (
                      ($topicId IS NULL AND TopicId IS NULL)
                      OR
                      ($topicId IS NOT NULL AND TopicId = $topicId)
                  )
                LIMIT 1;
            ";

            cmd.Parameters.AddWithValue("$userId", userId);
            cmd.Parameters.AddWithValue("$topicId",
                topicId.HasValue ? (object)topicId.Value : DBNull.Value);

            using var reader = cmd.ExecuteReader();
            if (!reader.Read())
                return null;

            return new UserTopicProgress
            {
                Id = reader.GetInt32(0),
                UserId = reader.GetInt32(1),
```

```
                TopicId = reader.IsDBNull(2) ? (int?)null :
reader.GetInt32(2),

                TotalAnswered = reader.GetInt32(3),

                CorrectAnswered = reader.GetInt32(4),

                LastDifficulty = reader.GetInt32(5)

            };

        }


        public List<UserTopicProgress> GetUserTopicsProgress(int userId)

        {

            var list = new List<UserTopicProgress>();


            using var connection = _database.CreateConnection();

            using var cmd = connection.CreateCommand();

            cmd.CommandText = @"

                SELECT Id, UserId, TopicId, TotalAnswered, CorrectAnswered,
LastDifficulty

                FROM UserTopicProgress

                WHERE UserId = $userId;

            ";

            cmd.Parameters.AddWithValue("$userId", userId);


            using var reader = cmd.ExecuteReader();

            while (reader.Read())

            {

                list.Add(new UserTopicProgress

                {

                    Id = reader.GetInt32(0),
```

```
                    UserId = reader.GetInt32(1),
                    TopicId = reader.IsDBNull(2) ? (int?)null :
reader.GetInt32(2),
                    TotalAnswered = reader.GetInt32(3),
                    CorrectAnswered = reader.GetInt32(4),
                    LastDifficulty = reader.GetInt32(5)
                });
            }

            return list;
        }

        public void SaveOrUpdateUserTopicProgress(
            int userId,
            int? topicId,
            int answered,
            int correct,
            int lastDifficulty)
        {
            if (answered <= 0)
                return;

            using var connection = _database.CreateConnection();

            UserTopicProgress? existing = null;

            using (var selectCmd = connection.CreateCommand())
            {
```

```
                selectCmd.CommandText = @"
                    SELECT Id, UserId, TopicId, TotalAnswered,
CorrectAnswered, LastDifficulty
                    FROM UserTopicProgress
                    WHERE UserId = $userId
                      AND (
                           ($topicId IS NULL AND TopicId IS NULL)
                           OR
                           ($topicId IS NOT NULL AND TopicId = $topicId)
                      )
                    LIMIT 1;
                ";

                selectCmd.Parameters.AddWithValue("$userId", userId);
                selectCmd.Parameters.AddWithValue("$topicId",
                    topicId.HasValue ? (object)topicId.Value : DBNull.Value);

                using var reader = selectCmd.ExecuteReader();
                if (reader.Read())
                {
                    existing = new UserTopicProgress
                    {
                        Id = reader.GetInt32(0),
                        UserId = reader.GetInt32(1),
                        TopicId = reader.IsDBNull(2) ? (int?)null :
reader.GetInt32(2),
                        TotalAnswered = reader.GetInt32(3),
                        CorrectAnswered = reader.GetInt32(4),
```

```
                    LastDifficulty = reader.GetInt32(5)
                };
            }
        }

        if (existing == null)
        {
            using var insertCmd = connection.CreateCommand();
            insertCmd.CommandText = @"
                INSERT INTO UserTopicProgress (UserId, TopicId,
TotalAnswered, CorrectAnswered, LastDifficulty)
                VALUES ($userId, $topicId, $totalAnswered,
$correctAnswered, $lastDifficulty);
            ";

            insertCmd.Parameters.AddWithValue("$userId", userId);
            insertCmd.Parameters.AddWithValue("$topicId",
                topicId.HasValue ? (object)topicId.Value : DBNull.Value);
            insertCmd.Parameters.AddWithValue("$totalAnswered",
answered);
            insertCmd.Parameters.AddWithValue("$correctAnswered",
correct);
            insertCmd.Parameters.AddWithValue("$lastDifficulty",
lastDifficulty);

            insertCmd.ExecuteNonQuery();
        }
        else
        {
            using var updateCmd = connection.CreateCommand();
```

```
                updateCmd.CommandText = @"
                    UPDATE UserTopicProgress
                    SET TotalAnswered   = TotalAnswered   + $answeredDelta,
                        CorrectAnswered = CorrectAnswered + $correctDelta,
                        LastDifficulty  = $lastDifficulty
                    WHERE Id = $id;
                ";

                updateCmd.Parameters.AddWithValue("$answeredDelta",
answered);
                updateCmd.Parameters.AddWithValue("$correctDelta", correct);
                updateCmd.Parameters.AddWithValue("$lastDifficulty",
lastDifficulty);
                updateCmd.Parameters.AddWithValue("$id", existing.Id);

                updateCmd.ExecuteNonQuery();
            }
        }

        public int CreateTestSession(int userId, int? topicId, int
questionLimit)
        {
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();

            cmd.CommandText = @"
                INSERT INTO TestSessions (UserId, TopicId, StartedAt,
QuestionLimit, AnsweredCount, CorrectCount)
                VALUES ($userId, $topicId, $startedAt, $limit, 0, 0);
```

```
                SELECT last_insert_rowid();
            ";

            cmd.Parameters.AddWithValue("$userId", userId);
            cmd.Parameters.AddWithValue("$topicId",
                topicId.HasValue ? (object)topicId.Value : DBNull.Value);
            cmd.Parameters.AddWithValue("$startedAt",
DateTime.Now.ToString("s"));
            cmd.Parameters.AddWithValue("$limit", questionLimit);

            var id = (long)cmd.ExecuteScalar();
            return (int)id;
        }

        public void UpdateTestSessionSummary(int sessionId, int answered, int
correct)
        {
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();

            cmd.CommandText = @"
                UPDATE TestSessions
                SET AnsweredCount = $answered,
                    CorrectCount  = $correct,
                    FinishedAt    = $finishedAt
                WHERE Id = $id;
            ";
```

```
            cmd.Parameters.AddWithValue("$answered", answered);
            cmd.Parameters.AddWithValue("$correct", correct);
            cmd.Parameters.AddWithValue("$finishedAt",
DateTime.Now.ToString("s"));
            cmd.Parameters.AddWithValue("$id", sessionId);

            cmd.ExecuteNonQuery();
        }

        public void LogQuestionResult(
            int sessionId,
            int questionId,
            int difficultyBefore,
            bool isCorrect,
            int? responseTimeMs)
        {
            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();

            cmd.CommandText = @"
        INSERT INTO QuestionResults
            (SessionId, QuestionId, Difficulty, IsCorrect, AnswerTimeMs)
        VALUES
            ($s, $q, $d, $c, $t);";

            cmd.Parameters.AddWithValue("$s", sessionId);
            cmd.Parameters.AddWithValue("$q", questionId);
            cmd.Parameters.AddWithValue("$d", difficultyBefore);
```

```
            cmd.Parameters.AddWithValue("$c", isCorrect ? 1 : 0);
            cmd.Parameters.AddWithValue("$t",
                responseTimeMs.HasValue ? (object)responseTimeMs.Value :
DBNull.Value);

            cmd.ExecuteNonQuery();
        }

        public List<TestSession> GetSessions(int userId)
        {
            var list = new List<TestSession>();

            using var connection = _database.CreateConnection();
            using var cmd = connection.CreateCommand();
            cmd.CommandText = @"
                SELECT Id, UserId, TopicId, StartedAt, FinishedAt,
                       QuestionLimit, AnsweredCount, CorrectCount
                FROM TestSessions
                WHERE UserId = $userId
                ORDER BY StartedAt;
            ";
            cmd.Parameters.AddWithValue("$userId", userId);

            using var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                list.Add(new TestSession
                {
```

```
                    Id = reader.GetInt32(0),

                    UserId = reader.GetInt32(1),

                    TopicId = reader.IsDBNull(2) ? (int?)null :
reader.GetInt32(2),

                    StartedAt = DateTime.Parse(reader.GetString(3)),

                    FinishedAt = reader.IsDBNull(4) ? null :
DateTime.Parse(reader.GetString(4)),

                    QuestionLimit = reader.GetInt32(5),

                    AnsweredCount = reader.GetInt32(6),

                    CorrectCount = reader.GetInt32(7)

                });

            }



            return list;

        }

    }

}
```

Answer.cs

```
namespace AdaptiveJavaTrainer.Models

{

    public class Answer

    {

        public int Id { get; set; }

        public int QuestionId { get; set; }

        public string Text { get; set; } = "";

        public bool IsCorrect { get; set; }

    }

}
```

Question.cs

```
using System.Collections.Generic;
```

```
namespace AdaptiveJavaTrainer.Models
{
    public class Question
    {
        public int Id { get; set; }
        public int TopicId { get; set; }
        public string Text { get; set; } = "";
        public int Difficulty { get; set; }
        public string QuestionType { get; set; } = "MCQ";
        public string? CodeSnippet { get; set; }
        public List<Answer> Answers { get; set; } = new();
    }
}
```

QuestionResult.cs

```
using System;

namespace AdaptiveJavaTrainer.Models
{
    public class QuestionResult
    {
        public int Id { get; set; }
        public int SessionId { get; set; }
        public int QuestionId { get; set; }
        public int Difficulty { get; set; }
        public bool IsCorrect { get; set; }
        public int AnswerTimeMs { get; set; }
```

```
    }
}
```

TestSession.cs

```
using System;

namespace AdaptiveJavaTrainer.Models
{
    public class TestSession
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int? TopicId { get; set; }

        public DateTime StartedAt { get; set; }
        public DateTime? FinishedAt { get; set; }

        public int QuestionLimit { get; set; }
        public int AnsweredCount { get; set; }
        public int CorrectCount { get; set; }

        public double Accuracy =>
            AnsweredCount == 0 ? 0.0 : (double)CorrectCount / AnsweredCount;
    }
}
```

Topic.cs

```
namespace AdaptiveJavaTrainer.Models
{
```

```
    public class Topic
    {
        public int Id { get; set; }
        public string Code { get; set; } = "";
        public string Name { get; set; } = "";
        public string? Description { get; set; }
    }
}
```

User.cs

```
namespace AdaptiveJavaTrainer.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Name { get; set; } = "";
    }
}
```

UserTopicProgress.cs

```
namespace AdaptiveJavaTrainer.Models
{
    public class UserTopicProgress
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int? TopicId { get; set; }
        public int TotalAnswered { get; set; }
        public int CorrectAnswered { get; set; }
```

```
        public double LastDifficulty { get; set; }


        public string? TopicName { get; set; }
    }
}
```