Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПІДГОТОВКИ ДО МУЛЬТИПРЕДМЕТНОГО ТЕСТУВАННЯ З АДАПТИВНОЮ ПЕРЕВІРКОЮ ЗНАНЬ»

Виконав: студент IV курсу, групи 1ІСТ-21б
спеціальності 126 – Інформаційні
системи та технології
Міщук М. Д.
Керівник: к.т.н., доцент каф. АІІТ
Севастьянов В. М.
«12» 06 2025 р.
Консультант: к.т.н., доцент кафедри АІІТ
Богач І. В.
«12» 06 2025 р.
Рецензент: к.т.н., доцент кафедри КН
Сілагін О. В.
«13» 06 2025 р.

Допущено до захисту
зав. кафедри АІІТ
«17» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
(шифр і назва)
Спеціальність 126 – Інформаційні системи та технології
(шифр і назва спеціальності)
Освітньо-професійна програма Інтелектуальні інформаційні системи
(назва освітньо-професійної програми)

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., проф. Бісікало О. В.
16. 06. 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Міщуку Максиму Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка застосунку для підготовки до мультипредметного тестування з адаптивною перевіркою знань»
Керівник роботи Севастьянов Володимир Миколайович к.т.н., доцент кафедри АІІТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "20" березня 2025 року № 97
2.Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи: Платформа розробки – Flutter; Мова програмування – Dart; Кількість сутностей у базі даних – не менше 5; Тип бази даних – нереляційна (Firebase Firestore); Кількість основних екранів мобільного застосунку – не менше 5 (вхід,головне меню, тестування, результати, профіль); Система авторизації – через email та пароль (Firebase Authentication); Система ролей – користувач / адміністратор; Візуалізація статистики – графік успішності користувача; Підтримка адаптивного тестування – з урахуванням попередніх результатів; Вивантаження результатів у Firestore – у вигляді структурованих документів;
4. Зміст текстової частини: Аналіз предметної області. Проєктування застосунку для тестування знань. Програмна реалізація застосунку. Висновки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) Блок-схема алгоритм роботи програми; Схема взаємодії ролей користувачів у мобільному застосунку; UML-діаграма активностей

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Богач І.В., доц. каф. АІІТ		

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	вик
1.	Аналіз предметної області	07.03.2025	13.04.2025	вик
2.	Огляд існуючих програмних засобів для реалізації поставленої задачі	13.04.2025	14.04.2025	вик
3.	Архітектура проєкту	15.04.2025	19.04.2025	вик
4.	Розробка технічного завдання (ТЗ)	20.04.2025	27.04.2025	вик
5.	Проєктування архітектури проєкту	28.04.2025	02.04.2025	вик
6.	Програмна реалізація мобільного застосунку	02.04.2025	30.04.2025	вик
7.	Оформлення пояснювальної записки та графічної частини	07.05.2025	17.05.2025	вик.
8.	Попередній захист БКР, доопрацювання, рецензування БКР	08.06.2025	10.06.2025	вик
9.	Захист БКР ЕК	16.06.2025	17.06.2025	вик

Студент _____ Міщук М. Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Севастьянов В. М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 105 сторінок формату А4 включаючи додатки, на яких є 28 рисунок, список використаних джерел містить 22 найменувань.

У бакалаврській кваліфікаційній роботі розглянуто розробку застосунку для мобільних пристроїв на базі Android та iOS для підготовки до мультипредметного тестування з адаптивною перевіркою знань та рекомендаціями. Основною метою є створення інструменту, який забезпечує ефективне, адаптивне та індивідуалізоване навчання з урахуванням рівня знань кожного користувача.

У додатку реалізовано можливість проходження тестів із різних предметів, таких як математика, українська мова, історія України, біологія, фізика та географія. Для кожного тесту передбачено п’ять варіантів відповідей, автоматичну перевірку результатів, збереження статистики та надання рекомендацій з урахуванням попередніх помилок.

Технічна реалізація виконана з використанням фреймворку Flutter, бази даних Firebase та Firestore для збереження користувацьких даних і статистики. Застосунок має сучасний інтерфейс, включає профіль користувача зі статистикою, діаграмами прогресу та системою нагадувань для регулярної підготовки.

Результати тестування продемонстрували стабільність роботи, зручність користування та позитивну динаміку навчальних досягнень серед учасників експериментального впровадження.

Ключові слова: мобільний застосунок, мультипредметне тестування, НМТ, адаптивне тестування, перевірка знань, навчальна система, персоналізація, Firebase, Flutter, освітні технології.

ABSTRACT

Bachelor's qualification work consists of 105 pages of the format A4 including applications with 28 drawings, the list of used sources contains 22 titles.

The thesis discusses the development of an application for mobile devices based on Android and iOS to prepare for multi-subject testing with adaptive knowledge verification and recommendations. The main goal is to create a tool that provides effective, adaptive and individualized training, taking into account the level of knowledge of each user.

The application provides the ability to pass tests in various subjects, such as mathematics, Ukrainian language, history of Ukraine, biology, physics and geography. For each test, there are five answer options, automatic verification of results, saving statistics and providing recommendations taking into account previous errors.

The technical implementation was made using the Flutter framework, Firebase and Firestore databases to store user data and statistics. The application has a modern interface, includes a user profile with statistics, progress charts and a dunning system for regular preparation.

The test results demonstrated the stability of work, ease of use and positive dynamics of educational achievements among the participants of the experimental implementation.

Keywords: mobile application, multi-target testing, NMT, adaptive testing, knowledge check, educational system, personalization, Firebase, Flutter, educational technologies.

ЗМІСТ

ВСТУП

Сучасна система освіти в Україні перебуває в стані активного реформування, що спрямоване на модернізацію змісту освіти, впровадження новітніх технологій та забезпечення рівного доступу до якісної підготовки учнів. Одним із ключових викликів сучасної освіти є забезпечення ефективної підготовки до зовнішнього незалежного оцінювання, яке у 2022 році було трансформовано у формат національного мультипредметного тесту (НМТ). Даний формат передбачає проходження тестування з кількох предметів у межах одного іспиту, що вимагає від учнів нових підходів до навчання та самопідготовки.

Враховуючи це нагальною є потреба у створенні сучасних цифрових інструментів, які не лише забезпечують проходження тестів, а й адаптують складність завдань до рівня знань користувача, зберігають для нього індивідуальну статистику та ритм, формують рекомендації щодо подальшого навчання. Такі інструменти дають можливість підвищити мотивацію до навчання, розвитку навичок самоконтролю та якісної підготовки до іспиту.

У даній роботі розроблено мобільний застосунок, який забезпечує інтерактивну підготовку до НМТ. Він дозволяє учням проходити тести з основних предметів, отримувати результати, аналізувати допущені помилки та слідкувати за своїм прогресом у зручному форматі. Адаптивна перевірка знань та візуалізація статистики допомагають ефективно коригувати навчальний процес.

Метою роботи є удосконалення методів підготовки учнів до національного мультипредметного тесту шляхом впровадження ефективних освітніх стратегій та сучасних технологій навчання.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

Завдання роботи полягає у:

1. Провести аналіз предметної області.
2. Спроєктувати інформаційну систему мобільного застосунку.

3. Реалізації інтерфейсів і функціональних модулів;

4. Створенні механізму адаптивного тестування та генерації рекомендацій;

5. Проведенні тестування програмного продукту.

Об’єктом дослідження є інформаційна система підтримки навчального процесу в умовах підготовки до НМТ.

Предметом дослідження виступає мобільний застосунок для адаптивного тестування та аналізу знань.

Методи дослідження. У процесі виконання бакалаврської кваліфікаційної роботи було використано комплекс теоретичних, емпіричних та прикладних методів дослідження, що забезпечили повне та системне вирішення поставлених завдань:

1. Аналіз літературних джерел та аналогічних програмних рішень – для вивчення сучасних підходів до комп’ютерного тестування, адаптивного навчання та реалізації мобільних застосунків освітнього призначення.

2. Метод системного аналізу – для визначення основних функціональних вимог до мобільного застосунку, постановки задачі та проєктування структури системи.

3. Методи об’єктно-орієнтованого проєктування (ООП) – використано при побудові архітектури програмного забезпечення та створенні UML-діаграм (класів, компонентів, послідовностей).

4. Моделювання та реалізація адаптивної логіки – для формування індивідуальних траєкторій тестування залежно від рівня знань користувача.

5. Експериментальне тестування – для перевірки працездатності програмного продукту в умовах, наближених до реального використання, із залученням тестових користувачів.

6. Методи візуалізації даних – реалізовано побудову графіків статистики результатів користувачів.

Науково-технічний результат роботи у результаті виконання бакалаврської кваліфікаційної роботи розроблено функціональний мобільний

застосунок, який забезпечує адаптивну підготовку до мультипредметного тестування (НМТ) з індивідуальним підходом до кожного користувача. Досягнуто наступних науково-технічних результатів:

1. Розроблено архітектуру застосунку
2. Реалізовано механізм адаптивного тестування
3. Створено систему збереження статистики з можливістю перегляду історії проходження тестів, побудови діаграм результатів та аналізу прогресу користувача.
4. Інтегровано базу запитань із підтримкою кількох предметів, з можливістю розширення вмісту.
5. Застосовано хмарні технології Firebase.
6. Розроблено інтуїтивно зрозумілий інтерфейс, який відповідає принципам UX/UI дизайну і забезпечує комфортне користування застосунком на мобільних пристроях.
7. Проведено функціональне та модульне тестування системи.

Практичне значення результатів роботи Результати бакалаврської кваліфікаційної роботи мають значне практичне значення, оскільки реалізований мобільний застосунок може бути ефективно використаний у сфері освіти для самостійної підготовки учнів до національного мультипредметного тестування (НМТ).

Апробація результатів роботи. Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету [1] та на Міжнародній науково-практичній інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН–2025)» [2].

Впровадження результатів роботи

Результати роботи планується використати в розробках та управлінні проєктами ТОВ «ІТІ» (додаток Г) та подана заявка на отримання свідоцтва на авторське право.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У сучасному світі освіта як і багато інших сфер діджиталізуються. З кожним роком зростає роль інформаційних технологій у забезпеченні якісної підготовки учнів до перевірки знань в формі підсумкових атестацій та вступних випробувань. Особливо це стосується мультипредметного тестування, яке запроваджується як форма оцінювання знань випускників загальноосвітніх навчальних закладів та абітурієнтів в Україні.

Мультипредметне тестування (НМТ) стало важливим етапом у системі вступу до закладів вищої освіти, замінивши традиційне зовнішнє незалежне оцінювання в умовах воєнного стану. Воно охоплює декілька предметів одночасно, зокрема українську мову, математику, історію України, а також на вибір біологію, хімію, фізику, географію (як вибіркові предмети). Такий формат потребує не лише глибоких знань із кожної дисципліни, а й уміння швидко орієнтуватися у змісті завдань, розподіляти час та приймати обґрунтовані рішення.

Разом з тим, учні часто стикаються з низкою труднощів під час підготовки до НМТ. Серед найпоширеніших – відсутність персоналізованого підходу, недостатній контроль за прогресом, низька мотивація та брак якісних електронних ресурсів. Крім того, в умовах обмеженого доступу до репетиторів чи офлайн-занять виникає потреба в зручному, доступному та адаптивному засобі підготовки, який враховує індивідуальний рівень знань учня.

Існуючі рішення на ринку переважно зосереджені на загальних тестах без чіткого поділу за предметами або без системи рекомендацій. Часто подібні застосунки не дозволяють проаналізувати помилки, не мають можливості формувати індивідуальні підбірки завдань та не надають адаптивного навантаження, що враховувало б результати попередніх відповідей. Це ускладнює ефективне засвоєння матеріалу.

У зв’язку з цим виникла ідея створити мобільний застосунок, який не тільки дозволяє проходити тести з кількох предметів, а й адаптується до рівня

знань конкретного користувача. Крім того, у застосунку реалізовано функцію рекомендацій – на основі результатів тестування система підказує, які теми потребують додаткової уваги. Це створює більш індивідуалізований та ефективний підхід до навчання.

Розробка такого застосунку має значне значення для підвищення рівня підготовки школярів. Вона дозволяє зробити процес навчання більш гнучким, доступним і технологічним. Водночас це сприяє впровадженню сучасних IT-рішень у сферу освіти, що є одним із пріоритетів розвитку інформаційного суспільства.

Таким чином, аналіз предметної області показує наявність реальної потреби у створенні мобільного інструменту для адаптивної підготовки до мультипредметного тестування з елементами рекомендаційної системи. Такий підхід сприяє підвищенню ефективності підготовки та розвитку цифрової культури навчання в Україні.

1.1 Загальні відомості про освітні мобільні застосунки

У наш час мобільні застосунки стали звичним та важливим інструментом у сфері освіти. Вони дозволяють учням і студентам отримувати знання в зручному форматі, використовуючи лише смартфон або планшет. Освітні застосунки охоплюють різноманітні напрямки: від вивчення мов і точних наук до проходження тестів і підготовки до іспитів. Такий формат навчання особливо актуальний в умовах дистанційної освіти, обмеженого доступу до репетиторів або необхідності самопідготовки.

Основними перевагами освітніх мобільних застосунків є:

- Доступність – користувачі можуть навчатися у зручний для себе час і в будь-якому місці;
- Інтерактивність – навчання відбувається у формі тестів, ігор, візуалізацій;

- Адаптивність – система може підлаштовуватись під рівень знань конкретного користувача;
- Мотивація до навчання – завдяки гейміфікації, візуальній статистиці, досягненням тощо.

Однією з найважливіших задач таких додатків є персоналізація освітнього процесу. Вони мають враховувати рівень знань, темп навчання, попередні результати та складність матеріалу. Найефективніші програми не лише оцінюють правильність відповідей, а й формують рекомендації для користувача, вказуючи на слабкі місця та допомагаючи покращити розуміння теми.

Особливе місце займають застосунки, орієнтовані на підготовку до стандартизованих тестів – таких як НМТ (Національне мультипредметне тестування). Цей формат передбачає необхідність швидкої перевірки знань з кількох предметів, аналізу типових помилок та відстеження прогресу. Саме тому розробка спеціалізованого мобільного застосунку з адаптивною перевіркою знань та рекомендаціями є актуальною та затребуваною серед учнів старших класів [3].

Таким чином, освітні мобільні додатки є ефективним інструментом у сучасній системі навчання, а створення спеціалізованих застосунків для підготовки до НМТ сприяє покращенню якості підготовки майбутніх абітурієнтів.

1.2 Тенденції розвитку освітніх застосунків

У сфері освітніх технологій останніми роками спостерігається стрімкий розвиток мобільних застосунків, орієнтованих на самостійне навчання, підготовку до іспитів та розвиток персональних навичок. Це зумовлено зростанням доступності смартфонів, розширенням інтернет-покриття та зміною підходів до навчання серед молоді.

Однією з головних тенденцій є перехід від традиційного навчання до гнучких цифрових рішень, що дає змогу учням самостійно планувати графік

підготовки до НМТ. Освітні застосунки стали не лише інструментами для перевірки знань, а й повноцінними навчальними платформами [4].

Основні напрямки розвитку:

- Адаптивне навчання – це підхід, при якому система сама визначає, на якому рівні перебуває користувач, і підлаштовує завдання саме під нього. Завдяки цьому тести перетворюються не просто на інструмент оцінювання, а на реальний засіб покращення знань і поступового розвитку.
- Гейміфікація – це використання ігрових механік, наприклад, нагород, рівнів, балів чи таблиць лідерів, у навчальному процесі. Такий підхід робить навчання цікавішим і підвищує внутрішню мотивацію користувачів продовжувати займатись.
- Аналітика результатів дає змогу користувачам побачити, у чому вони досягли прогресу, а над чим ще варто попрацювати. Детальний зворотний зв’язок допомагає краще розуміти власні успіхи й проблемні теми, а отже – раціонально планувати подальше навчання.
- Рекомендаційні системи – на основі відповідей користувача система підказує, які теми слід повторити або поглибити.
- Підтримка кількох предметів – сучасні застосунки часто об’єднують підготовку з кількох дисциплін в одному інтерфейсі, що актуально саме для мультипредметного тестування.

Ці тенденції свідчать про зростання попиту на персоналізоване, доступне й ефективне навчання. Саме на цих принципах базується концепція створеного застосунку, який пропонує адаптивну перевірку знань, рекомендації, багатопредметний формат тестування та мотиваційні механізми.

Таким чином, аналіз тенденцій розвитку освітніх застосунків підтверджує доцільність і своєчасність розробки мобільного продукту для підготовки до НМТ, який не просто перевіряє знання, а допомагає навчатися ефективніше.

1.3 Функціональні вимоги до освітнього застосунку для підготовки до мультипредметного тестування

Процес розробки будь-якого програмного забезпечення починається з чіткого формулювання функціональних вимог. У контексті створення освітнього мобільного застосунку для підготовки до мультипредметного тестування ці вимоги визначають ключові можливості системи, які мають забезпечити ефективну, зручну й персоналізовану підготовку користувача.

Застосунок має бути не просто платформою для проходження тестів, а й інтелектуальним помічником, що враховує рівень підготовки кожного учня, формує рекомендації для подальшого навчання, а також мотивує до самостійної роботи.

1. Реєстрація та вхід до системи

Кожен користувач має змогу створити свій обліковий запис, використовуючи індивідуальні логін та пароль. Після успішної реєстрації облікові дані зберігаються в базі даних – це може бути як локальне сховище, так і хмарне. Авторизація не лише відкриває доступ до системи, а й забезпечує персоналізований досвід, дозволяючи зберігати історію тестувань. Додатково передбачена функція відновлення або зміни пароля. У системі реалізовано поділ користувачів на ролі: звичайний учень і адміністратор.

2. Вибір предметів та тематики

Після входу користувач отримує доступ до переліку навчальних дисциплін, що відповідають структурі НМТ: українська мова, математика, історія України, біологія, фізика, географія. Доступна фільтрація питань за темами, що дозволяє зосередитись на окремих розділах.

3. Процес проходження тесту

Тести можуть формуватися випадковим чином або з урахуванням попередніх результатів користувача, що дає змогу реалізувати адаптивний формат. Кожне питання супроводжується п’ятьма варіантами відповіді, з яких

лише один є правильним. Для наближення до реального іспиту використовується таймер. Після вибору відповіді дані фіксуються для подальшого аналізу.

4. Результати та аналіз тестування

Після завершення проходження тесту система показує:

– загальну кількість набраних балів;

– відсоток правильних відповідей;

– перелік помилкових відповідей із правильними варіантами.

Уся інформація зберігається в обліковому записі користувача, що дозволяє будувати детальну статистику – як по предметах, так і по темах.

5. Персоналізовані рекомендації

Виходячи з аналізу результатів, система генерує поради:

– які теми слід повторити;

– які тести можуть допомогти закріпити слабкі місця;

– як краще організувати підготовку.

Алгоритми автоматично виявляють типові помилки та співвідносять їх із тематикою питань.

6. Особистий кабінет користувача

У профілі зібрана повна історія проходжень тестів, динаміка змін у вигляді графіків за кожним предметом, а також відображаються середній бал, кількість спроб і загальний рейтинг. Користувач має змогу змінювати персональні дані при потребі.

7. Інструменти адміністратора

Для адміністраторів передбачений окремий розділ застосунку, доступний лише за відповідними правами. Вони мають можливість:

– додавати, редагувати або видаляти питання;

– оновлювати вміст бази даних;

– переглядати активність учнів;

– слідкувати за якістю контенту та його відповідністю освітнім стандартам.

8. Збереження прогресу та офлайн-режим

Навіть у разі втрати інтернет-з’єднання, застосунок зберігає результати локально. Після відновлення доступу до мережі всі зміни автоматично синхронізуються з хмарною базою.

9. Інтерфейс і зручність користування

Дизайн застосунку розроблений з урахуванням потреб сучасних користувачів: логічне розміщення елементів, читабельні шрифти, приємна кольорова палітра. Інтерфейс адаптується до різних розмірів екранів, що забезпечує комфортну роботу з мобільного пристрою будь-якого типу.

1.4 Технологічні вимоги до освітнього застосунку

У процесі розробки мобільного застосунку для підготовки до мультипредметного тестування важливу роль відіграє правильний вибір технологій, які зможуть забезпечити стабільну роботу, масштабованість та зручність у підтримці й розвитку програмного продукту. Нижче подано ключові технологічні вимоги, які висуваються до такого типу застосунків:

- Кросплатформеність

Застосунок має бути доступним на Android та iOS. Оптимальним вибором є використання Flutter – фреймворку з єдиною кодовою базою для обох платформ.

- Легка вага та продуктивність

Застосунок повинен бути швидким у завантаженні, не споживати багато ресурсів пристрою та працювати без затримок навіть на бюджетних смартфонах.

- База даних

Для локального збереження результатів тестувань і користувацьких даних застосовується SQLite або Hive. Для синхронізації з хмарою – Firebase Cloud Firestore.

- Інтерфейс користувача (UI/UX)

Інтерфейс має бути простим, логічним і адаптивним. Всі елементи – кнопки, діаграми, вкладки – мають бути інтуїтивно зрозумілими для учнів.

1.5 Огляд конкурентних методів

Аналіз існуючих освітніх мобільних застосунків, які доступні на українському ринку, дозволяє краще зрозуміти, що саме цінується користувачами. Аналіз конкурентів допомагає не лише побачити їхні переваги, а й виявити недоліки або незадоволені потреби. Це, у свою чергу, дає змогу чітко визначити, які функції та підходи варто врахувати при створенні власного рішення. Далі представлено порівняльну характеристику найпопулярніших мобільних платформ для підготовки до ЗНО та НМТ, які нині активно використовуються учнями в Україні.

1.5.1 ZNO.ua

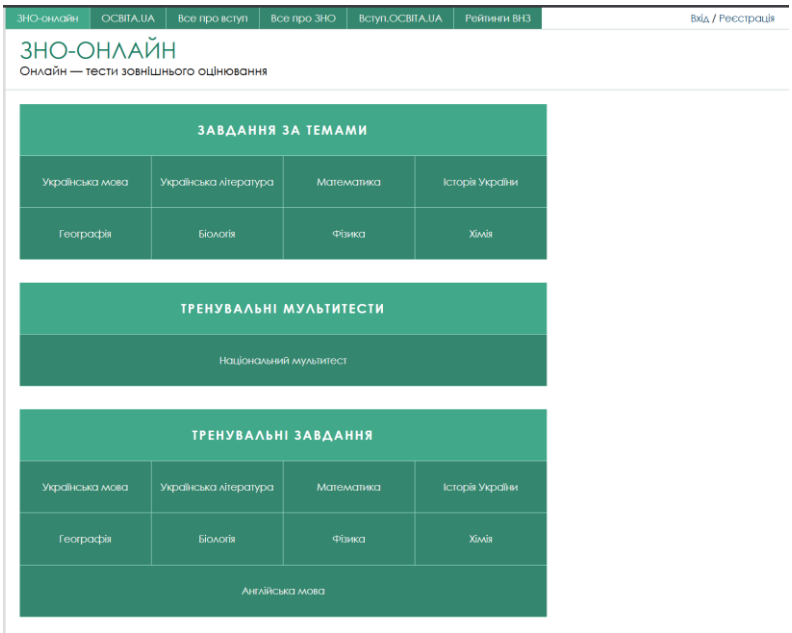


Рисунок 1.1 – головна сторінка ЗНО.ua

ЗНО.ua – один з перших українських мобільних застосунків (рис. 1.1), орієнтованих безпосередньо на підготовку до ЗНО. Даний застосунок надає

доступ до великої бази тестових завдань з основних навчальних предметів, що відповідають навчальним програмам Міністерства освіти і науки України [5].

Основний функціонал:

- доступ до тестових завдань попередніх років;
- можливість проходження тематичних тестів;
- облік правильних і неправильних відповідей;
- підрахунок загального результату у форматі 200-бальної шкали.

Переваги:

– Широкий вибір різноманітних тем: користувачу надається можливість обирати як повні тести, так і окремі тематичні блоки, що дозволяє краще готуватись до розділів, які гірше засвоєні і при тестуванні дали нижчий результат.

– Статистика результатів: кожен тест супроводжується короткою аналітикою, що дозволяє учню відстежувати свій прогрес, планувати свій темп навчання.

Недоліки:

– Відсутність адаптивного навчання: застосунок не враховує рівень знань користувача при формуванні завдань.

– Немає індивідуальних рекомендацій: програма не пропонує учню повторити конкретні теми на основі результатів.

– Морально застарілий інтерфейс: частина візуальних елементів та навігації виглядає застарілою на фоні сучасних дизайнерських рішень.

– Низька ергономіка застосунку.

ЗНО.ua є надійним інструментом для перевірки знань, однак він орієнтований здебільшого на повторення, а не на розвиток. Його функціонал не передбачає гнучкого, адаптивного підходу до навчання, що обмежує його ефективність у порівнянні з сучасними вимогами до навчання.

Відгуки про сайт znoua переважно позитивні, зокрема стосовно якості навчання, доступності завдань. Багато учнів відзначають, що курси значно

покращили їхню підготовку до ЗНО та НМТ, допомогли виявити та усунути прогалини в знаннях, а також створили позитивну атмосферу навчання.

Багато учнів вказують на зростання їхніх результатів після відвідування даного застосунку про що свідчать позивні відгуки присутні на сайті.

Відгуки переважно позитивні, але зрідка можна зустріти думки про потребу у більшій кількості часу для опанування матеріалу або про потребу у більш детальному поясненні певних тем.

1.5.2 ILearn

Одним із найпомітніших українських проєктів у сфері безкоштовної освіти є освітня онлайн-платформа iLearn, створена громадською організацією «Освіторія». Проєкт був запущений з метою забезпечення учнів рівним доступом до якісної підготовки до зовнішнього оцінювання знань. З часом платформа набула широкого визнання, особливо серед учнів з регіонів, де існує обмежений доступ до репетиторських послуг або якісного навчального контенту.

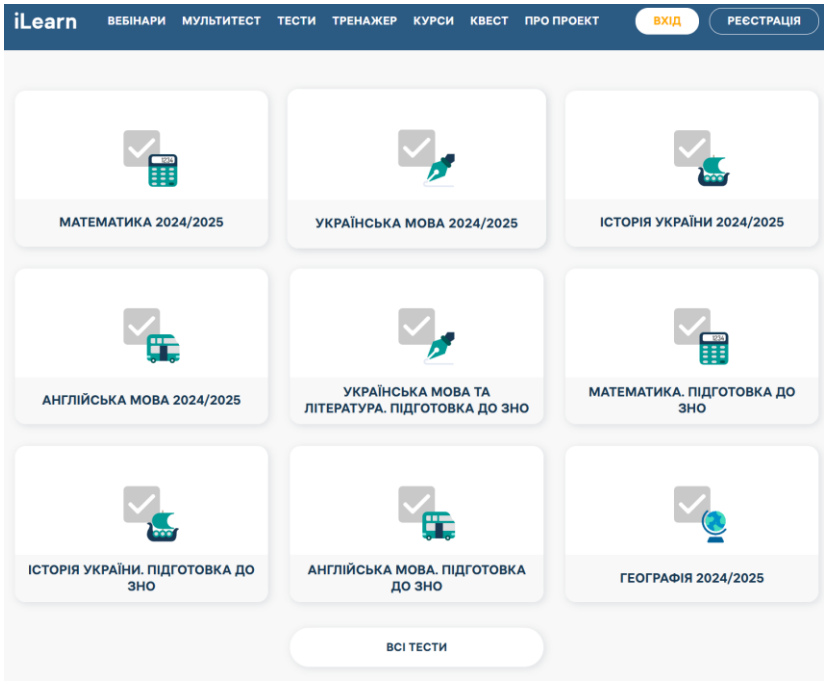


Рисунок 1.2 – Головне меню навчальної платформи ILearn

Платформа iLearn (рис. 1.2) орієнтована насамперед на підготовку до Національного мультипредметного тесту (НМТ). Платформа пропонує повноцінну навчальну підтримку, поєднуючи відеолекції, інтерактивні завдання, пояснювальні матеріали та мотиваційні елементи, що дозволяє охопити різні стилі сприйняття інформації [6].

Основний функціонал платформи:

- Відеолекції, які допомагають глибше зрозуміти складні теми;
- Тестові тренажери, що дозволяють учням перевіряти рівень знань у режимі реального часу;
- Структуровані навчальні матеріали, згруповані за тематиками відповідно до чинної шкільної програми;
- Механізми мотивації, серед яких – система досягнень, рейтингів, заохочень, що сприяють підвищенню зацікавленості учнів у навчанні.

Переваги використання iLearn:

Однією з найвагоміших переваг платформи є наявність повноцінного мультимедійного контенту, що значно підвищує якість засвоєння знань. Матеріали, представлені у формі відеолекцій, дозволяють учням краще зрозуміти складні теми, а також самостійно повторювати їх у зручний для себе час. Візуальний і слуховий типи сприйняття активно задіяні, що робить процес підготовки більш ефективним, ніж традиційне заучування.

Крім того, платформа акцентує увагу на розвитку критичного мислення. У багатьох завданнях, що пропонуються в рамках тестів, користувачу потрібно не просто обрати правильну відповідь, а й аргументувати свій вибір. Такий підхід відповідає сучасним педагогічним принципам та сприяє формуванню аналітичного мислення.

Ще однією важливою перевагою є абсолютна безкоштовність ресурсу. Користувачі мають доступ до повного обсягу матеріалів без додаткових оплат, реєстраційних зборів або реклами. Це робить платформу доступною для широкої аудиторії, включно з учнями з малозабезпечених родин.

Недоліки та обмеження:

Попри численні переваги, iLearn має й певні недоліки, які обмежують його ефективність у контексті персоналізованого мобільного навчання. Зокрема, платформа не передбачає глибокого аналізу помилок користувача. Після завершення тесту учень отримує лише загальний результат, без пояснення допущених помилок або рекомендацій щодо подальшого опрацювання слабких тем. Це унеможливлює повноцінну адаптацію освітнього процесу під конкретного користувача.

Ще одним суттєвим недоліком є обмежена функціональність мобільного застосунку. У порівнянні з вебверсією, мобільна версія має суттєво спрощений інтерфейс і не підтримує повний набір функцій, зокрема аналітику, перегляд частини матеріалів, роботу з відео у фоновому режимі тощо. У зв’язку з цим використання платформи на смартфонах менш зручне і не дозволяє повноцінно навчатися «на ходу».

Отже, можна підбити наступні висновки, а саме, що iLearn – це досить не погана інноваційна освітня ініціатива, яка відіграє важливу роль у забезпеченні рівного доступу учнів для якісної підготовки до НМТ. Її сильними сторонами є безкоштовність, мультиформатність контенту, мотиваційні інструменти та залучення кваліфікованих педагогів. Проте, з точки зору мобільного, адаптивного навчання застосунок не реалізує повною мірою можливості сучасних інформаційних технологій. Відсутність глибокого аналізу результатів, персональних рекомендацій та обмежений мобільний функціонал вказують на потенціал для подальшого вдосконалення застосунку.

У розроблюваному в межах даної бакалаврської кваліфакаційної роботи застосунку пропонується врахувати згадані недоліки, реалізувавши інструменти адаптивної перевірки знань, індивідуального аналізу помилок та рекомендаційних підказок, що дозволить користувачу отримати більш гнучкий, ефективний та персоналізований досвід підготовки.

1.5.3 Math Corporation:

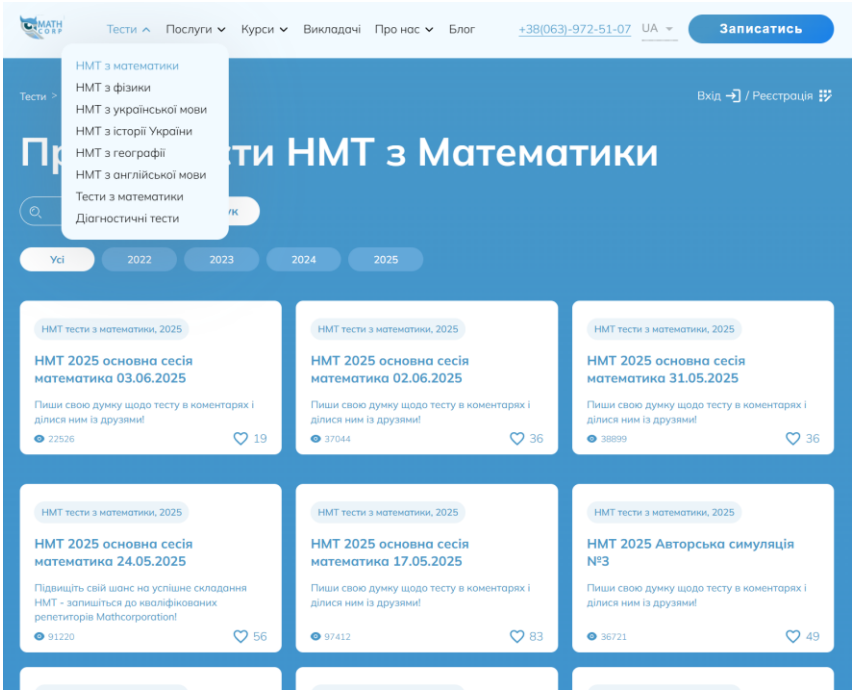


Рисунок 1.3 – Головне меню навчальної платформи Math Corporation

Платформа Math Corporation (рис 1.3) є прикладом сучасного спеціалізованого вебзастосунку, що орієнтується на підготовку до Національного мультипредметного тесту (НМТ) з математики. Сервіс надає можливість проходження завдань з попередніх років у форматі, наближеному до офіційного. Основна цільова аудиторія – випускники шкіл, абітурієнти, учні старших класів, які готуються до складання тесту з математики [7].

Основний функціонал платформи:

- База тестів НМТ за всі роки проведення, з розподілом за роками;
- Інтерактивний інтерфейс тестування, наближений до реального екзаменаційного середовища;
- Автоматична перевірка відповідей та обчислення результату;
- Підтримка математичних символів і формул, що важливо для коректного відображення завдань;

- Можливість анонімного використання – проходження тестів без реєстрації.

Переваги:

- Вузька спеціалізація. Платформа зосереджена саме на математичних завданнях, що дозволяє глибоко опрацювати цей предмет.
- Максимальна наближеність до реального формату НМТ. Структура, оформлення та логіка подання питань відповідають офіційним тестам.
- Оперативність оновлення. Нові тести з'являються практично одразу після проведення реального НМТ.
- Висока якість візуалізації математичних формул завдяки використанню спеціальних HTML-розміток і рендерінгу.

Недоліки:

- Відсутність загального кабінету користувача. Платформа не надає особистий профіль, історію проходження тестів чи можливість відстежувати прогрес.
- Відсутність тематичного поділу. Тести представлені лише за роками, без можливості вибору конкретної теми або розділу математики.
- Обмеженість до одного предмету. Платформа не підтримує інші дисципліни, що не дозволяє її використовувати як універсальний інструмент для повної підготовки до НМТ.

Math Corporation – це якісний, спеціалізований веб-ресурс для підготовки саме до математичного блоку НМТ. Однією з сильних сторін є точність у відтворенні формату справжнього тесту, а також чистота реалізації математичних елементів. Проте відсутність індивідуалізації, загального обліку результатів, рекомендацій і охоплення інших предметів обмежує його функціональність у порівнянні з багатопредметними, адаптивними мобільними застосунками. Такий ресурс може бути корисним як додатковий інструмент, але не замінює повноцінної системи з аналітикою, персоналізацією та мотиваційною складовою.

1.5.4 Освіта.ua

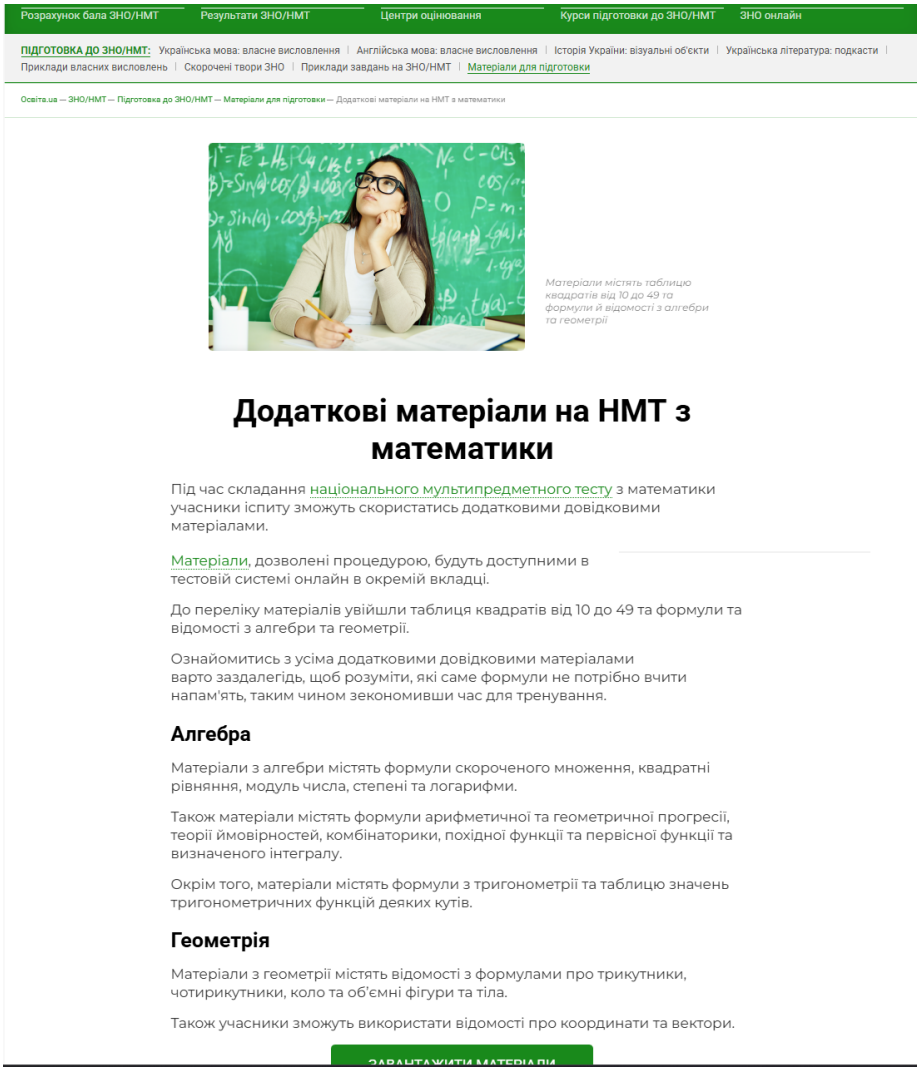


Рисунок 1.4 – Інтерфейс сайту Освіта.ua

Платформа Освіта.ua (рис. 1.4) є одним із найбільш відвідуваних освітніх порталів в Україні, що містить велику кількість корисних матеріалів для учнів, абітурієнтів та вчителів. У межах підготовки до ЗНО та НМТ сайт надає безкоштовний доступ до тематичних та повних тестів, що базуються на матеріалах Українського центру оцінювання якості освіти. Тести охоплюють усі основні предмети, які включено до структури НМТ, зокрема українську мову, математику, історію України, біологію, фізику, географію. Користувач має змогу проходити тести онлайн з моментальною перевіркою результатів, а також бачити правильні відповіді після завершення кожного блоку завдань [8].

Переваги:

- Платформа є повністю безкоштовною, не потребує реєстрації, що дозволяє швидко розпочати тестування з будь-якого пристрою.
- Великий вибір тестів за темами та роками дає змогу системно опрацьовувати потрібний матеріал.
- Завдяки простому інтерфейсу та миттєвій перевірці відповідей учень отримує швидкий зворотний зв’язок, що дозволяє самостійно оцінити рівень підготовки.
- Окремою перевагою є наявність архіву тестів попередніх років, що допомагає краще зрозуміти структуру та рівень складності реальних екзаменаційних завдань

Недоліки:

- Платформа не підтримує індивідуальний профіль користувача, не зберігає історію результатів і не пропонує системи статистичного аналізу успішності.
- Відсутній механізм адаптації складності завдань до рівня учня, що унеможливлює створення персоналізованої навчальної траєкторії. Крім того, система не генерує рекомендацій щодо тем для повторення на основі попередніх помилок, що знижує ефективність використання ресурсу як інструменту тривалої підготовки.

Отже, платформа Освіта.ua є зручним та доступним джерелом тестових матеріалів, проте вона більше виконує роль відкритої бази даних, а не повноцінного інтерактивного навчального середовища. Для глибшої та персоналізованої підготовки потрібне рішення, яке об’єднує адаптивне навчання, статистичну аналітику, рекомендаційні підказки та інтерфейс, орієнтований на постійне відстеження прогресу користувача

1.5.5 Studinfo.org

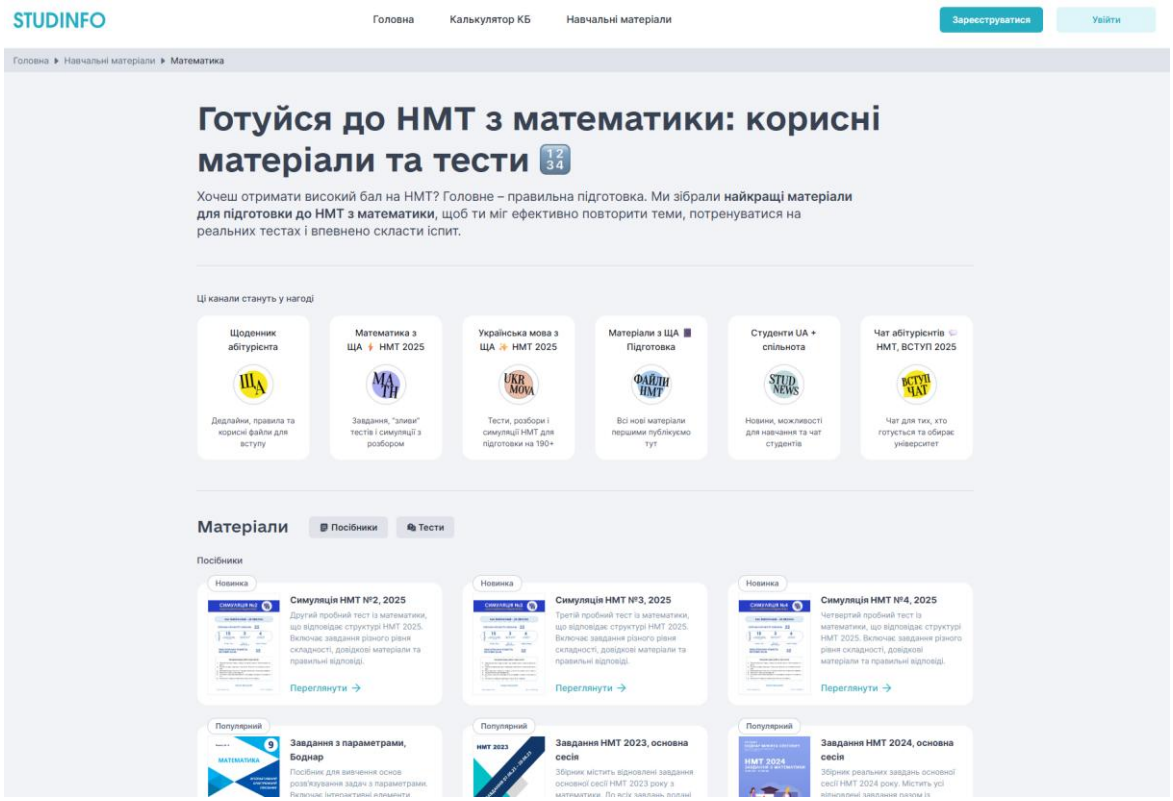


Рисунок 1.5 - Головне меню навчальної платформи Studinfo.org

Studinfo.org – це онлайн-ресурс (рис. 1.5) , що пропонує доступ до великої кількості навчальних матеріалів з різних предметів, зокрема й з математики. Сайт містить достатньо теоретичного матеріалу, формули, довідкові таблиці та приклади розв’язання типових завдань, які можуть бути корисними під час самостійної підготовки до тестування. Хоча платформа не має окремого мобільного додатку, її інтерфейс добре адаптований для використання на різноманітних ґаджетах через браузер.

Сайт орієнтований насамперед на надання доступу до текстових навчальних ресурсів, які можуть бути використані для підготовки до тестування. На відміну від інших платформ, тут немає інтерактивного тестування чи адаптивної системи перевірки знань, однак для користувачів, які шукають компактні й перевірені матеріали для повторення тем, цей ресурс є досить зручним [9].

Переваги:

- Сайт не вимагає реєстрації, має зручну навігацію за темами, охоплює як шкільний курс, так і завдання підвищеної складності.
- Доступні таблиці формул, логічно структуровані підрозділи та приклади розв’язань роблять його корисним джерелом для швидкого повторення матеріалу.

Недоліки:

- Не є повноцінним інтерактивним інструментом – відсутня система тестування, немає персонального кабінету, аналітики чи адаптації матеріалів до рівня учня.
- Вміст більше нагадує електронний підручник, ніж сучасну навчальну платформу.

Отже, Studinfo.org може розглядатися як допоміжне джерело для теоретичного закріплення знань, однак не є самодостатнім інструментом для повної підготовки до мультипредметного тестування. У рамках розробки власного застосунку доцільно брати з таких ресурсів структурність і стислий формат теорії, але доповнювати їх аналітикою, адаптивністю та інтерактивністю.

1.6 Перспективні напрями розвитку мобільних освітніх застосунків для підготовки до мультипредметного тестування

Сфера інтерактивної освіти із застосуванням сучасних цифрових інформаційних технологій останніми роками зазнає суттєвих змін, особливо після переходу шкільної системи на дистанційну та дуальну форму навчання. Саме тому розробка мобільних застосунків, орієнтованих на підготовку до іспитів, зокрема до Національного мультипредметного тестування, стає дедалі актуальнішою. Користувачі більше не очікують простої бази тестів – вони шукають зручний, зрозумілий, а головне – ефективний інструмент, який дозволить готуватися у власному ритмі, бачити свій прогрес і вчасно отримувати

підказки про те, що потрібно повторити. Це формує чітке уявлення про те, у якому напрямку мають розвиватися подібні продукти.

Перш за все, зростає потреба у персоналізації. Якщо раніше більшості учнів було достатньо проходити стандартні блоки тестів, то зараз дедалі більше користувачів очікують, що застосунок буде підлаштовуватися під їх рівень. Тобто не просто показувати правильні й неправильні відповіді, а аналізувати ці помилки, робити висновки й пропонувати щось на кшталт індивідуального плану дій. Це може бути просте повідомлення: «Ви часто помиляєтесь у темах з тригонометрії» або «Варто повторити частину з правопису», – і вже такий функціонал дає користувачу відчуття живої взаємодії.

Ще одна важлива тенденція – це візуалізація прогресу. Людям, особливо підліткам, важливо бачити результат. Не просто відсоток правильних відповідей, а й зміну цього показника з часом. Наприклад, якщо учень постійно бачить, що показники рівнів знань з предметів зростає з умовних 40% до 75%, то, авжеж, це позитивно впливає на подальшу мотивацію до підготовки. Або ж, навпаки, якщо динаміка йде вниз – це привід замислитися й повернутись до складних тем. Усе це можна реалізувати через прості графіки, діаграми, кольорові індикатори – без зайвого ускладнення інтерфейсу.

Окрім цього, важливо, щоб сам застосунок не відчувався як обов’язок та тягар. І тут на допомогу приходять елементи гейміфікації. Це не означає, що потрібно перетворювати підготовку до НМТ у гру, але навіть найпростіші речі – система балів, відкриття нових рівнів, невеликі винагороди у вигляді візуальних значків – можуть сильно впливати на мотивацію. У багатьох учнів така механіка працює краще, ніж будь-які нагадування від вчителів.

Ще одним напрямком, який не варто ігнорувати, є можливість використовувати застосунок в офлайн-режимі. В Україні досі багато регіонів, де доступ до стабільного інтернету обмежений. Якщо програма дозволяє проходити тести без підключення до мережі, а потім синхронізує результати, коли інтернет з’явиться, – це суттєва перевага. Така автономність, з одного боку, підвищує

зручність, а з іншого – свідчить про продуманий підхід до користувача, який може опинитися в будь-яких умовах.

І, нарешті, не менш важливою є можливість інтеграції з іншими ресурсами. Наприклад, багатьом учням було б зручно експортувати результати в Google Classroom, або мати можливість поділитися успіхами з батьками чи вчителем. Усе це – перспективні, хоча й не завжди очевидні, напрямки, які можуть суттєво підвищити якість та конкурентоздатність освітнього мобільного застосунку.

Загалом, ринок освітніх цифрових продуктів стрімко змінюється, і ті рішення, які ще кілька років тому здавались новітніми, нині вже сприймаються як мінімальна необхідність. Створення застосунку, що не просто дозволяє проходити тести, а й вміє аналізувати результати, давати поради, зберігати статистику й мотивувати – це вже не побажання, а вимога часу. Саме на таких принципах і базується ідея розробки системи, що представлена у межах цієї бакалаврської роботи.

1.7 Об’єкт автоматизації

У сучасних умовах, коли самостійна підготовка до іспитів стає все більш актуальною через нестачу часу, обмежений доступ до репетиторів або дистанційний формат навчання, постає потреба у якісних цифрових рішеннях, які можуть частково або повністю замінити традиційні методи навчання. Саме тому об’єктом автоматизації у межах даної бакалаврської кваліфікаційної роботи виступає процес підготовки учнів до національного мультипредметного тесту (НМТ) за допомогою спеціалізованого мобільного застосунку.

Іншими словами, мова йде про автоматизацію ключових етапів навчання, які раніше вимагали участі викладача або великого обсягу ручної роботи з боку самого учня. Застосунок, розроблений у межах цього проєкту, бере на себе частину завдань, які традиційно виконувались у форматі «людина-людина», і переводить їх у формат «людина-система». Зокрема, автоматизовано такі процеси:

- Формування тестів з урахуванням попередніх результатів користувача — тобто не випадковий набір завдань, а адаптивне тестування, що підлаштовується під рівень знань;
- Оцінювання результатів у режимі реального часу, без необхідності перевірки з боку викладача;
- Фіксація та збереження статистики успішності — кожне тестування автоматично зберігається, що дозволяє відстежувати прогрес користувача;
- Генерація індивідуальних рекомендацій — система самостійно аналізує, де користувач найчастіше помиляється, і пропонує теми для повторення;
- Візуалізація результатів у зручному вигляді — графіки, середні бали, динаміка успішності;
- Адміністрування бази знань — адміністратор може легко додавати, редагувати чи видаляти питання без змін у програмному коді.

Автоматизація саме цих елементів навчального процесу дозволяє не лише заощадити час учня, а й зробити сам процес підготовки ефективнішим, зручнішим і менш залежним від людського фактору. Завдяки інтерактивності, доступності з мобільного пристрою та адаптивності, застосунок виконує роль не лише інструменту перевірки знань, а й повноцінного «цифрового тьютора», що супроводжує користувача протягом усього періоду підготовки до НМТ.

Таким чином, об’єкт автоматизації — це не окрема технічна функція, а цілісний освітній процес, який із допомогою інформаційних технологій трансформується у більш ефективну, персоналізовану і керовану форму навчання.

1.8 Висновки

У даному розділі було здійснено глибокий аналіз специфіки наявних освітніх мобільних застосунків, орієнтованих на підготовку до тестування в тому

числі НМТ. Основна увага приділялась визначенню ключових можливостей, які мають бути реалізовані в сучасному навчальному сервісі. Серед них – проведення тестувань, фіксація результатів, формування персоналізованих рекомендацій та адаптація під рівень користувача.

Поряд із цим було розглянуто декілька популярних рішень, що вже представлені на українських навчальних інтернет платформах, зокрема: ЗНО.ua, iLearn, EdEra, Math Corporation, Освіта.ua та Studinfo.org. Кожен із цих ресурсів має свої переваги – в основному зручний інтерфейс, доступність матеріалів, якісна база знань. Проте жодна з платформ не забезпечує повноцінної адаптивності, де система самостійно аналізує відповіді, накопичує статистику і пропонує персональні поради на основі дій користувача.

Також було відзначено сучасні тенденції в освітніх технологіях. Користувач очікує не просто набір питань, а розумну, гнучку систему, яка вміє "слухати" його поведінку, реагувати на помилки, показувати прогрес та будувати індивідуальний шлях навчання. Усе це підкреслює необхідність створення нового підходу – рішення, яке не повторює вже наявні продукти, а формує якісно новий рівень взаємодії з учнем.

У результаті проведеного аналізу були чітко визначені функціональні та технічні вимоги до майбутнього застосунку. Саме вони стали базисом для подальшого проєктування архітектури, підбору технологій і формування логіки системи. Таким чином, сформувалось бачення того, яким повинен бути сучасний мобільний інструмент для підготовки до НМТ – інтуїтивним, розумним, адаптивним і справді корисним для кожного користувача.

2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ

Розробка програмного продукту, зокрема мобільного застосунку, розпочинається не з написання коду, а з продуманого проєктування архітектури. Від коректності та простоти побудови внутрішньої логіки, визначення взаємозв’язків між компонентами, організації зберігання даних та принципів обміну інформацією залежить зручність використання системи, її стабільність, гнучкість до змін та можливість масштабування у подальшому.

Під час створення мобільного застосунку для підготовки до мультипредметного тестування з адаптивною перевіркою знань було реалізовано підхід, який поєднує зручність реалізації, доступність сучасних технологій і практичність у подальшому використанні. В основу проєкту покладено фреймворк Flutter, що дає змогу реалізовувати застосунок для Android та iOS одночасно з єдиної кодової бази. Такий підхід забезпечує простоту технічної підтримки й дозволяє зосередитися на функціональній складовій.

Для організації зберігання даних обрано хмарну платформу Firebase, а саме компонент Cloud Firestore – нереляційну базу даних, що відзначається високою гнучкістю та інтеграцією з Flutter-застосунками. Структура даних реалізується у вигляді колекцій та документів, що відповідає логіці моделювання тестових завдань, результатів проходження, профілів користувачів та індивідуальної статистики. Додатково, за допомогою Firebase Authentication, забезпечено надійну автентифікацію без необхідності реалізації окремого серверного модуля.

На етапі архітектурного проєктування важливим аспектом було не лише поділ застосунку на окремі екрани, а й формування чітких взаємозв’язків між основними функціональними блоками: модулем тестування, профілем користувача, підсистемою аналітики, базою питань та інтерфейсом адміністратора. Це забезпечило створення логічно зв'язаної архітектури, орієнтованої на стабільну роботу та зручність у користуванні.

Наступні підрозділи детально описують архітектурні рішення, що були реалізовані під час розробки, принципи взаємодії між компонентами, структуру

обробки даних та обґрунтування вибору відповідних технологій, що дозволили досягти основної мети – створення адаптивного, доступного та функціонального освітнього мобільного застосунку.

2.1 Вибір платформи для виконання поставленої задачі

Мобільний застосунок, реалізований у межах даної бакалаврської роботи, створено з використанням фреймворку Flutter. Це сучасне кросплатформене рішення, яке дозволяє розробляти повноцінні інтерфейси для мобільних операційних систем Android та iOS на основі єдиної кодової бази. Такий підхід істотно спрощує процес підтримки та розширення функціональності застосунку, оскільки всі зміни стають одночасно доступними на обох цільових платформах.

Архітектура застосунку включає кілька ключових елементів інтерфейсу: головне меню, екран вибору предмету, екран тестування, екран результатів та профіль користувача з візуалізацією статистики. Окремо реалізовано адміністративний модуль, доступ до якого мають лише користувачі з відповідними правами доступу. У цьому модулі передбачено можливість додавання нових тестових запитань до бази даних.

Усі дані, включаючи облікові записи користувачів, відповіді на тестові завдання, а також результати проходження тестів, зберігаються у хмарній базі даних Firebase Firestore. Структура зберігання побудована за принципом вкладених колекцій: для кожного користувача створено окремий документ, що містить підколекцію stats, у якій фіксуються такі параметри, як дата проходження, назва предмета, кількість правильних відповідей тощо. Така модель забезпечує зручність доступу до інформації та її подальшої обробки, що є основою для формування статистичних звітів та графічних візуалізацій у профільному розділі застосунку. У цьому модулі передбачено можливість додавання нових тестових запитань до бази даних.

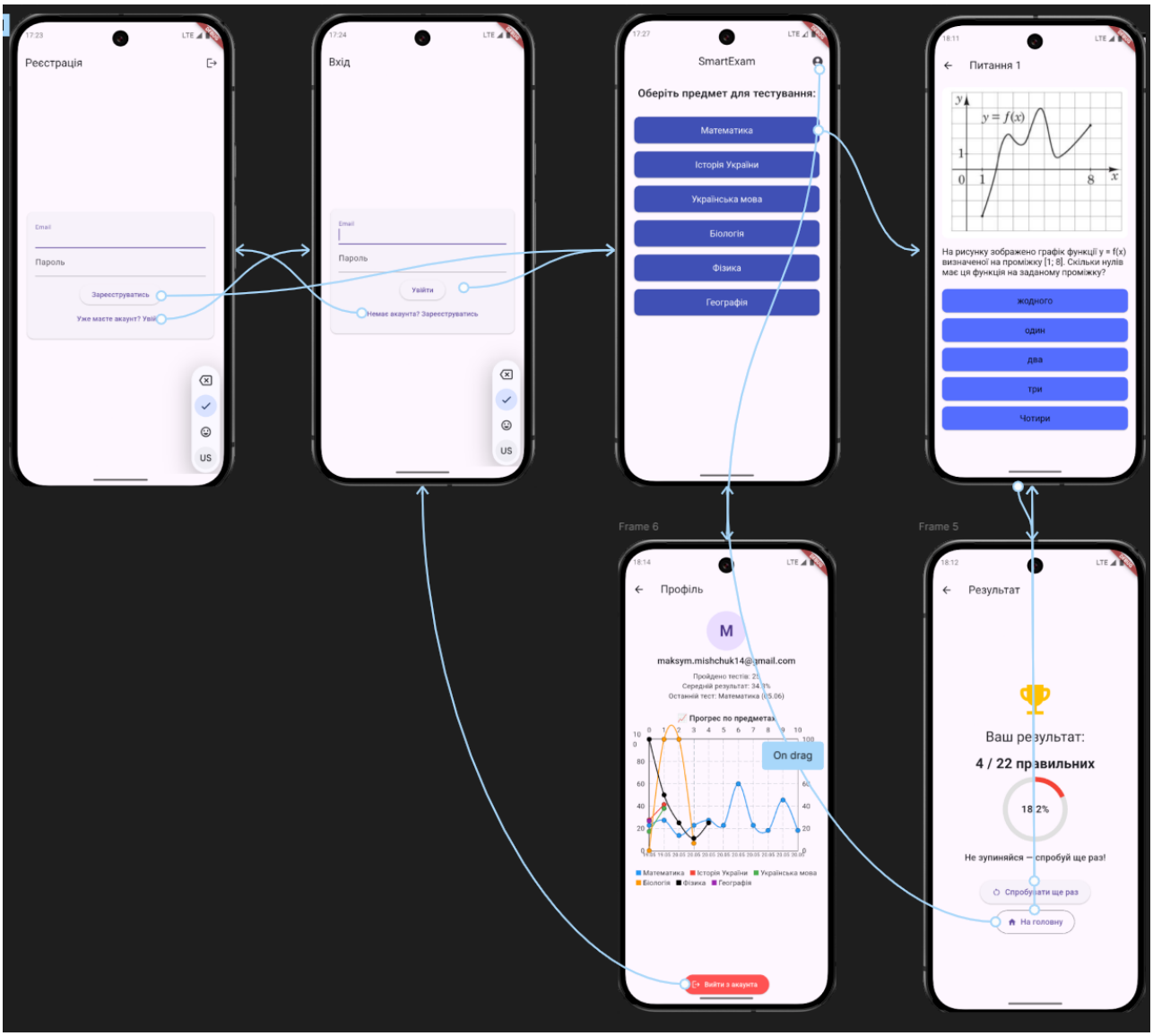


Рисунок 2.1 – Загальна схема екранів застосунку та зв’язків між ними

Загальну логіку переходів між основними екранами застосунку наведено на рисунку 2.1.

У межах реалізації взаємодії з Firebase у застосунку не використовується окремий серверний API. Усі операції, пов’язані з автентифікацією користувачів, зчитуванням та записом результатів тестування, здійснюються безпосередньо за допомогою SDK Firebase for Flutter. Такий підхід суттєво знижує обсяг необхідного коду, підвищує безпеку обробки даних і мінімізує ризики їх втрати або некоректного зберігання.

Окремим функціональним елементом застосунку є реалізація адаптивної перевірки знань. Логіка адаптації ґрунтується на обліку попередніх результатів тестування: у разі, якщо користувач неодноразово припускається помилок у завданнях певної тематики, система пропонує користувачу на омснові аналізу графічної динаміки власних результатів зробити вибір, щодо подальших дій.

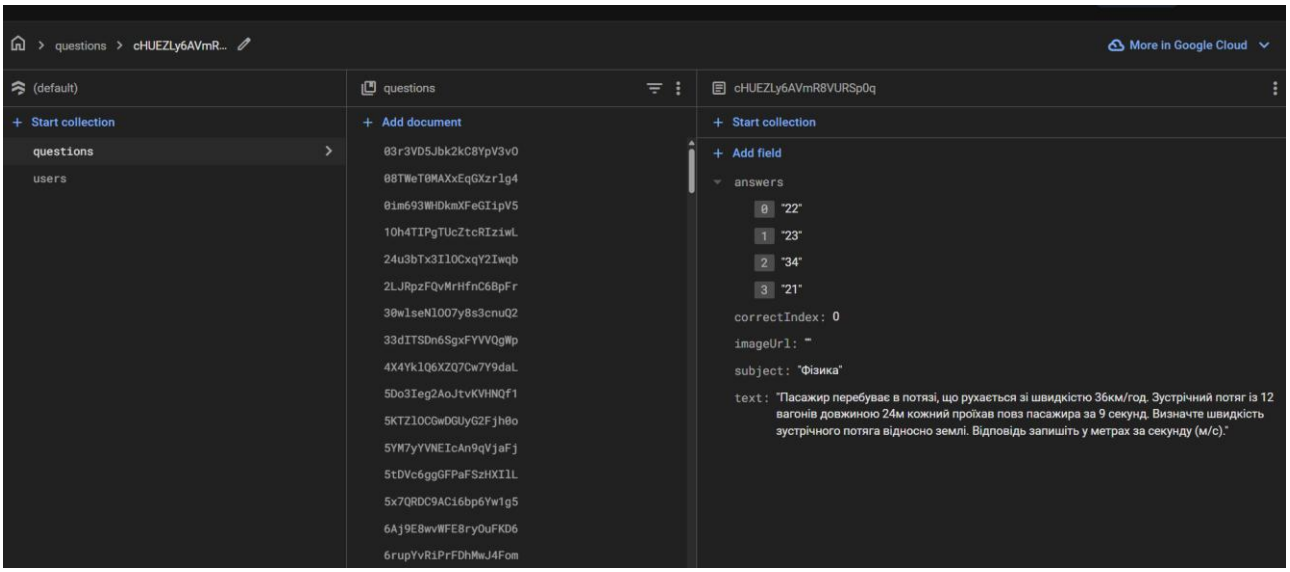


Рисунок 2.2 – Структура бази даних у Firestore (користувачі, питання, результати)

Для реалізації зворотнього зв’язку з рекомендаціями в подальшому можна використати механізм перевірки категорій завдань у базі Firestore, що дозволяє забезпечити персоналізований підхід без залучення складних алгоритмів штучного інтелекту. Структура колекцій та полів, що зберігаються у Firestore, зокрема питань, відповідей і тематик, наведена на рисунку 2.2.

Таким чином, архітектура застосунку побудована просто, але ефективно: чітко розділені елементи інтерфейсу, логіка користувача і логіка збереження даних. Застосунок не перевантажений функціями, але водночас має усе необхідне для індивідуального й результативного навчання. Такий підхід дозволяє підтримувати й масштабувати програму у майбутньому без повної перебудови структури.

2.1.1 Використання Flutter для кросплатформенного інтерфейсу

Фреймворк Flutter був обраний як основна технологія для реалізації мобільного застосунку у межах даного проєкту. Це сучасне рішення, яке забезпечує створення мобільних застосунків одночасно для платформ Android та iOS з використанням єдиної кодової бази. Такий підхід дозволяє знизити витрати ресурсів на розробку, спрощує подальшу підтримку та модернізацію програмного забезпечення.

Flutter, розроблений компанією Google, відзначається високою продуктивністю, широкими можливостями налаштування інтерфейсу та активною спільнотою розробників. Основною мовою програмування є Dart, що спеціально адаптована під використання у межах цього фреймворку. Однією з ключових функцій є підтримка механізму "hot reload", який дозволяє в реальному часі оновлювати інтерфейс без перезапуску додатку. Це значно підвищує ефективність розробки та пришвидшує внесення змін до структури екранів [10].

Використання Flutter забезпечило реалізацію всіх компонентів користувацького інтерфейсу – екранів, кнопок, стилів, анімацій – без необхідності залучення сторонніх бібліотек. Завдяки вбудованій системі навігації вдалося організувати логічні переходи між основними частинами застосунку: з головного меню – до тестування, з екрану результатів – до перегляду статистики тощо.

Фреймворк також демонструє високу сумісність з платформою Firebase, що було використано для зберігання даних та реалізації механізму автентифікації. Така інтеграція дозволила уникнути створення окремого серверного бекенду, зосередившись на функціональності клієнтської частини.

Окрім технічних переваг, Flutter надає широкі можливості для візуального налаштування дизайну. Завдяки зручності зміни стилів, кольорових схем та розташування елементів, вдалося створити адаптивний і зрозумілий інтерфейс, який однаково коректно відображається як на смартфонах, так і на планшетах.

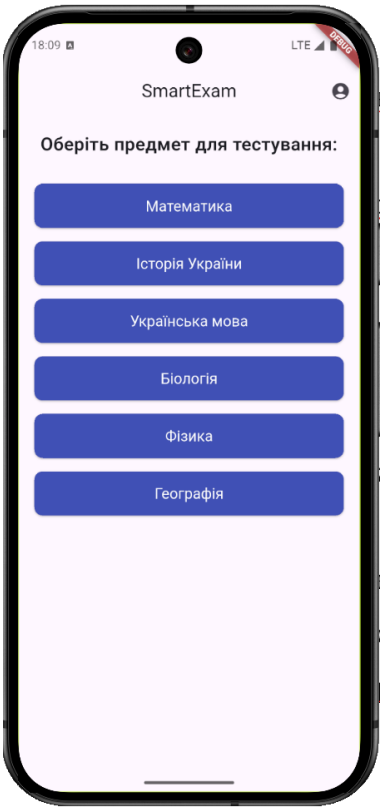


Рисунок 2.3 – Приклад розробленого інтерфейсу «Головний екран»

На рисунку 2.3 представлено приклад головного екрана застосунку, де користувач обирає предмет для проходження тестування.

Загалом, Flutter показав себе як дуже зручний інструмент, який дозволяє швидко розробити повноцінний застосунок із красивим інтерфейсом, не втрачаючи при цьому гнучкості або функціональності. В умовах обмеженого часу й ресурсів саме ця технологія дозволила реалізувати все заплановане в межах бакалаврського проєкту, не жертвуючи якістю чи зручністю для користувача.

2.1.2 Firebase як основа для зберігання даних і аутентифікації

Після визначення Flutter як основного фреймворку для розробки мобільного застосунку, постала необхідність у виборі відповідного інструменту для зберігання даних, пов’язаних з користувачами, тестовими завданнями,

результатами та статистикою. Було обрано рішення, яке не вимагало створення складної серверної інфраструктури, але при цьому забезпечувало надійність, масштабованість і зручну інтеграцію з Flutter. Такими характеристиками володіє Firebase – платформа від Google, яка поєднує кілька ключових сервісів для підтримки мобільного застосунку.

Для організації бази даних застосовано компонент Cloud Firestore, який є хмарною нереляційною базою, структурованою за принципом колекцій та документів. Така модель виявилася ефективною у контексті мобільної архітектури, де важливо забезпечити гнучкість, швидкий доступ до даних та адаптивність структури. Наприклад, кожному користувачу відповідає окрема підколекція зі статистикою проходження тестів, яка автоматично оновлюється після завершення кожної сесії. Загальна база тестових питань зберігається у відокремленій колекції, доступ до якої обмежено лише адміністративною роллю [11].

Однією з визначальних переваг Firestore є підтримка реактивної моделі оновлення даних. Будь-які зміни, що вносяться до бази, миттєво відображаються в інтерфейсі користувача без необхідності перезапуску застосунку. Це дозволяє досягти високого рівня зручності: результати тестування стають доступними одразу після завершення, а актуальна статистика оновлюється в режимі реального часу у відповідному розділі профілю.

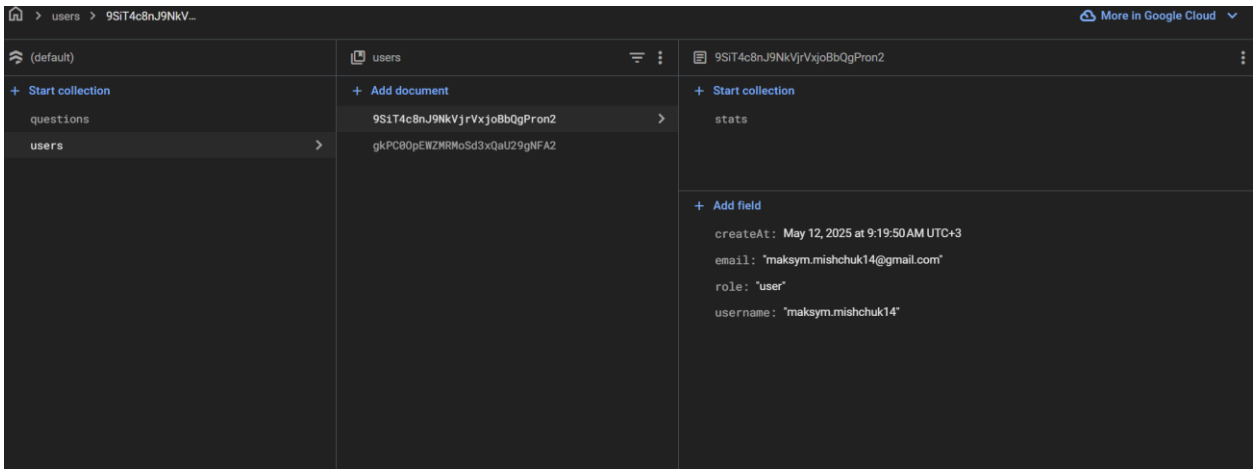


Рисунок 2.4 – Структура бази даних Firestore у застосунку

На рисунку 2.4 наведено структуру зберігання користувачів у базі даних Firestore із вкладеною підколекцією stats, де зберігаються результати тестів.

Крім організації зберігання даних, важливою складовою інформаційної архітектури стало впровадження системи аутентифікації користувачів. Для цього використано сервіс Firebase Authentication, який забезпечує створення облікових записів, вхід до системи та обмеження доступу до окремих функціональних модулів залежно від ролі користувача. Впроваджено базову схему автентифікації з використанням електронної пошти та пароля, що є найбільш зручним і поширеним варіантом у навчальних мобільних застосунках.

Firebase забезпечує високий рівень безпеки, оскільки обробка й зберігання паролів реалізується автоматично на стороні сервісу. Усі паролі хешуються, і жодна конфіденційна інформація не передається чи не зберігається у відкритому вигляді. Завдяки цьому суттєво знижується ризик витоку персональних даних, а також відпадає потреба в реалізації окремих модулів шифрування на стороні клієнтського застосунку.

Для розмежування прав доступу реалізовано просту, але ефективну рольову модель, засновану на правилах безпеки в Firestore. Зокрема, лише користувачу з логіном «admin» надано право додавання нових тестових завдань, тоді як звичайні користувачі мають доступ виключно до проходження тестів і перегляду власної статистики. Такий підхід дозволяє забезпечити гнучке управління функціональністю, не ускладнюючи логіку програми.

Загалом, використання Firebase як базового рішення для аутентифікації та управління ролями дозволило не лише спростити архітектуру, а й забезпечити високу інтегрованість усіх складових застосунку. Система побудована таким чином, щоб бути гнучкою до подальших змін: у разі необхідності можливо додати нові модулі, розширити перелік предметів або масштабувати функціональність на інші платформи без зміни базової логіки.

2.1.3 Структура взаємодії між компонентами застосунку

Для забезпечення зручності використання мобільного застосунку, а також його стабільної роботи, необхідно ретельно продумати не лише зовнішній інтерфейс, але й внутрішню логіку функціонування. Особливу увагу слід приділити взаємодії між екранами, розмежуванню їх функцій, а також організації зберігання та передачі даних між компонентами системи.

Структура реалізованого мобільного застосунку є результатом поетапного вдосконалення – від початкових ескізів до повноцінної моделі з чітко визначеними функціональними модулями. Кожен екран виконує конкретну роль у загальній архітектурі та не дублює функціональність інших.

Алгоритм взаємодії користувача із застосунком починається з екрана авторизації або реєстрації. Після успішного входу до системи користувач потрапляє на головний екран, де відображено список доступних навчальних предметів (наприклад, українська мова, математика, історія тощо). Натискання на назву предмета ініціює перехід до екрану тестування, де користувачеві пропонуються завдання з кількома варіантами відповідей. Після завершення тестування формується результат – кількість правильних відповідей, відсоток успішності та сумарний бал [12].

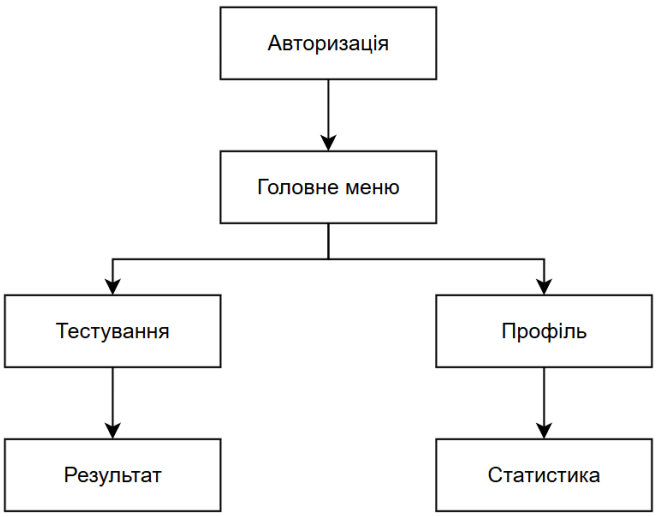


Рисунок 2.5 – Схема переходів між екранами мобільного застосунку

Дані про проходження тестів зберігаються в хмарній базі Firestore, зокрема у вкладеній колекції stats, прив’язаній до облікового запису кожного користувача. Таким чином, забезпечується персоналізоване збереження результатів, які згодом можна переглянути у відповідному розділі профілю. У вкладці статистики представлено історію проходжень у форматі списку або графічної візуалізації, що дозволяє аналізувати динаміку прогресу користувача у навчанні. Схематичну структуру переходів між основними екранами мобільного застосунку наведено на рисунку 2.5.

Усі переходи між екранами мобільного застосунку реалізовано за допомогою стандартної навігаційної системи Flutter – з використанням методів Navigator.push та Navigator.pop. Такий підхід дозволяє ефективно управляти стеком екранів, забезпечуючи логічну послідовність дій та зручне повернення до попередніх станів без необхідності перевантаження всього інтерфейсу.

Тестові завдання не зберігаються локально на пристрої. Замість цього, при відкритті екрана тестування, застосунок надсилає запит до колекції questions, розміщеної в базі Firestore. Завдяки цьому з’являється можливість динамічного оновлення навчального контенту – додавання нових питань або коригування наявних – без потреби оновлення застосунку через маркетплейси. Окрема частина інтерфейсу, доступна лише для авторизованих користувачів з адміністративними правами, дозволяє керувати тестовою базою безпосередньо з мобільного пристрою.

Усі дії, що передбачають взаємодію з Firebase, супроводжуються перевіркою наявності інтернет-з’єднання. У випадках відсутності стабільного підключення реалізовано базову логіку обробки помилок, що дозволяє запобігти втраті даних. У подальшій розробці передбачено впровадження механізмів тимчасового локального збереження даних (з використанням SQLite або Hive) з автоматичною синхронізацією при відновленні з’єднання.

Структура застосунку побудована з урахуванням чіткої логіки: кожен екран виконує визначену функцію, кожна дія призводить до очікуваного результату, а всі дані, зокрема результати тестування, фіксуються та підлягають

подальшому аналізу. Така архітектура забезпечує зручність супроводу й масштабування застосунку, відкриває можливості для розширення функціоналу – додавання нових предметів, типів тестів, ролей користувачів, а також вдосконалення аналітичних інструментів.

2.1.4 Особливості роботи з Firestore у Flutter

Однією з ключових задач у процесі розробки мобільного застосунку стало впровадження ефективного механізму збереження та обробки даних, який би забезпечував стабільність роботи, високу швидкість доступу та не вимагав складної серверної інфраструктури. З цією метою було обрано хмарну нереляційну базу даних Cloud Firestore від платформи Firebase, яка має гнучку структуру та надійну інтеграцію з середовищем Flutter.

Завдяки готовим бібліотекам, зокрема cloud_firestore, інтеграція Firestore у мобільний застосунок здійснюється напряму без необхідності створення додаткових API або окремого бекенд-сервера. Структура бази побудована на основі вкладених колекцій: колекція users містить документи, що ідентифікують окремих користувачів за унікальним UID, а всередині кожного з них розміщується підколекція stats, де зберігаються результати проходження тестів – назви предметів, кількість правильних відповідей, відсоток успішності та дати [13].

Додавання нових документів до бази даних реалізовано через формування структурованого об’єкта типу Map<String, dynamic>, який передається до методів add() або set(). Такий механізм дозволяє гнучко зберігати різноманітні типи даних без потреби створення попередніх схем чи фіксованих таблиць.

Особливістю Firestore є його реактивність – інтерфейс застосунку оновлюється автоматично при зміні відповідних даних у базі. Такий підхід реалізовано через використання потоків (наприклад, StreamBuilder), які слухають зміни в колекціях та оновлюють вміст екранів без необхідності перезапуску чи ручного оновлення.

Безпека взаємодії з базою забезпечується за допомогою Firebase Security Rules, які дозволяють обмежити доступ до окремих документів і колекцій. Зокрема, користувач має доступ лише до власних результатів, тоді як адміністратор отримує розширені можливості, зокрема доступ до редагування контенту тестів. Реалізація таких обмежень здійснюється шляхом перевірки UID користувача в умовах доступу.

Зміни в базі можуть вноситися динамічно через Firebase Console без потреби модифікувати код застосунку. Це відкриває широкі можливості для оновлення контенту: додавання нових запитань, створення нових предметів, редагування вже існуючих елементів тощо.

Таким чином, Firestore забезпечує повноцінну екосистему для зберігання, обробки та оновлення навчальних даних. Його інтеграція з Flutter створює ефективну, масштабовану та безпечну архітектуру, що повністю задовольняє потреби освітнього мобільного застосунку в рамках студентських або MVP-проєктів.

2.2 Вибір технологій та програмного середовища для виконання поставленої задачі

Під час реалізації мобільного застосунку було застосовано набір сучасних інструментів і технологій, що забезпечили ефективне виконання усіх етапів розробки – від проєктування інтерфейсу до зберігання даних та тестування функціональності. Особливістю підходу стала можливість побудови повноцінного технологічного стеку без потреби у складній інфраструктурі або важкому програмному забезпеченні. Обрана комбінація інструментів дозволила створити зручне, гнучке й масштабоване рішення.

Основним середовищем розробки виступив Visual Studio Code – легкий та гнучкий редактор коду, який завдяки численним розширенням, зокрема Flutter, Dart, Firebase Tools, забезпечив зручне створення інтерфейсів, швидкий пошук

та виправлення помилок, реалізацію механізму hot reload, а також оптимальну навігацію між екранами [14].

Головною технологією став Flutter – кросплатформенний фреймворк від компанії Google, що дозволяє розробляти застосунки для Android та iOS на основі єдиної кодової бази. Такий підхід значно спрощує підтримку застосунку, знижує витрати часу на окрему розробку для кожної платформи та забезпечує послідовність дизайну й функціональності. Flutter надає розробникам широкі можливості завдяки готовим віджетам, розвиненій документації та активній спільноті.

Для зберігання даних, реалізації автентифікації користувачів та підключення до хмарної інфраструктури було використано Firebase. У межах цього проєкту застосовано Firestore як основну базу даних, Firebase Authentication для створення та перевірки облікових записів, а також консоль керування, що надала можливість адмініструвати контент та розмежовувати ролі користувачів. Інтеграція Firebase через Flutter SDK забезпечила просту та стабільну реалізацію всіх ключових функцій без потреби у власному серверному рішенні.

На етапі візуального проєктування інтерфейсу використовувалась платформа Figma. Вона надала змогу створити макети екранів і наочно продемонструвати логіку переходів між ними ще до початку написання коду. Також Figma використовувалась для підготовки схем і графічного супроводу технічної документації.

Для перевірки працездатності застосунку, адаптивності інтерфейсу та швидкості роботи було використано Android-емулятор та фізичний мобільний пристрій. Це дозволило провести базове тестування зовнішнього вигляду інтерфейсу на різних розмірах екранів і оцінити реальний користувацький досвід.

Загалом, використання вищезазначених інструментів забезпечило комфортний процес розробки, дозволило уникнути складних налаштувань і зосередитися на функціональній реалізації програмного продукту.

2.2.1 Мова програмування

У процесі створення мобільного застосунку основною мовою програмування було обрано Dart. Такий вибір зумовлений тим, що ця мова є нативною для роботи з фреймворком Flutter, який використовувався для розробки інтерфейсу. Dart є сучасною та гнучкою мовою з інтуїтивно зрозумілим синтаксисом, що поєднує риси JavaScript і Java. Завдяки цьому вона добре підходить для розробників різного рівня підготовки, включаючи тих, хто лише починає роботу з мобільною розробкою.

Однією з ключових переваг Dart є її орієнтація на створення інтерфейсних застосунків із високою швидкістю роботи та привабливим візуальним оформленням. Убудована підтримка асинхронного програмування (через конструкції async/await) забезпечує ефективну взаємодію з віддаленими сервісами, зокрема з базою даних Firebase. Це дає змогу реалізовувати зчитування та запис даних без затримок чи блокувань головного потоку виконання.

Мова Dart також підтримує об’єктно-орієнтовану модель програмування, що дозволяє організовувати програмну логіку у вигляді класів, сервісів та окремих компонентів інтерфейсу. Такий підхід сприяє високій структурованості коду, спрощує його підтримку та забезпечує можливість масштабування – зокрема, через легке додавання нових типів тестів, предметів або функцій [15].

Ще однією важливою особливістю є підтримка механізму "гарячого перезапуску" (hot reload). Цей функціонал надає змогу миттєво переглядати зміни в інтерфейсі без повного перезавантаження застосунку, що істотно підвищує швидкість і зручність процесу розробки, а також дозволяє оперативно тестувати нові рішення.

Таким чином, мова Dart продемонструвала свою ефективність у реалізації мобільного застосунку освітнього призначення. Поєднання простоти, продуктивності та сумісності з Flutter зробило її оптимальним вибором для

розробки швидкого, гнучкого та зручного у використанні застосунку для підготовки до мультипредметного тестування.

2.2.2 Інтегроване середовище розробки

Для написання та тестування коду мобільного застосунку було обрано Visual Studio Code (VS Code) як основне інтегроване середовище розробки. Цей редактор відзначається високою швидкістю запуску, невеликим споживанням системних ресурсів і гнучкістю налаштувань, що робить його придатним для створення застосунків на основі Flutter.

У порівнянні з повноцінними IDE, такими як Android Studio, VS Code демонструє кращу продуктивність на слабших пристроях і дозволяє зосередитися виключно на процесі програмування. Наявність необхідних розширень (Flutter, Dart, Firebase Tools, Bracket Pair Colorizer, Material Icon Theme тощо) забезпечує комфортну роботу з інтерфейсом, логікою застосунку та інтеграцією з хмарними сервісами.

Інтерфейс середовища дозволяє швидко перемикатися між файлами, переглядати структуру проєкту, запускати емулятор, відстежувати помилки за допомогою інтегрованої консолі. Особливо ефективною виявилася підтримка механізму "гарячого перезапуску" (hot reload), що дозволяє миттєво переглядати зміни інтерфейсу без потреби повного перезапуску програми. Це значно пришвидшило процес налагодження та вдосконалення дизайну інтерфейсу.

Додатковою перевагою є вбудована підтримка системи контролю версій Git, яка дозволяє здійснювати коміти, створювати гілки, переглядати історію змін безпосередньо в межах редактора. Такий функціонал є вкрай корисним при довготривалій роботі над застосунком і сприяє структурованому підходу до управління кодовою базою.

Загалом, Visual Studio Code забезпечив стабільне та комфортне середовище розробки, що повністю відповідало вимогам проєкту. Його можливості сприяли

ефективній реалізації усіх етапів – від початкового прототипування до фінального тестування функціоналу мобільного застосунку.

2.2.3 Інструмент управління проєктами

Для організації процесу розробки застосунку було обрано спрощену, але ефективну модель управління завданнями, яка не вимагала використання повноцінних корпоративних рішень типу Jira або Asana. У якості основного інструменту було використано Trello – вебзастосунок для візуального планування, побудований за принципом канбан-дошки.

Завдання групувалися в категорії на кшталт To Do, In Progress, Testing, Done, що дозволяло структуровано відстежувати хід виконання проєкту. Кожне завдання оформлювалося у вигляді картки з відповідним описом, дедлайнами, посиланнями або додатковими файлами. Це забезпечувало централізоване управління інформацією, сприяло системному підходу до виконання задач та зменшувало ризик пропущених етапів або забутих елементів функціональності.

Окремі картки відповідали ключовим аспектам проєкту, зокрема підключенню Firebase, реалізації авторизації, створенню статистики, адаптації інтерфейсу під різні розміри екранів тощо. Ідеї, що виникали в процесі, але не входили до початкового плану, також фіксувалися у Trello для можливого впровадження в майбутньому.

Крім Trello, додатково використовувався табличний редактор Google Sheets для контролю дедлайнів, фіксації виконаних функцій, обліку знайдених помилок та ведення загального прогресу. Такий інструмент сприяв кращому плануванню часу й дозволив систематизувати технічні нотатки.

Використані засоби управління забезпечили високий рівень організованості при мінімальних витратах часу на їх обслуговування. Такий підхід виявився ефективним для індивідуальної розробки, забезпечив прозорість процесу та сприяв стабільному просуванню проєкту до завершення.

2.3 Вибір шаблонів проєктування

У процесі розробки мобільного застосунку для підготовки до мультипредметного тестування було прийнято рішення використовувати кілька поширених шаблонів проєктування, які забезпечують гнучкість архітектури, розділення обов'язків між модулями та спрощення подальшої підтримки коду.

Одним із ключових шаблонів, що був застосований, є MVVM (Model-View-ViewModel). Цей шаблон дозволяє логічно відокремити бізнес-логіку застосунку від його інтерфейсу, що значно спрощує тестування окремих компонентів. У межах цього підходу:

- Model відповідає за доступ до даних – зокрема, взаємодію з Firebase Firestore, зчитування та збереження результатів, структурування статистики користувача;
- View – це інтерфейс користувача, побудований із використанням Flutter-виджетів;
- ViewModel – компонент, який опрацьовує дані з Model, готує їх до відображення та керує станами, що відображаються у View.

Для управління станом застосунку обрано Provider – офіційно підтримуваний Flutter-пакет для реалізації патерну Dependency Injection. Provider забезпечив зручний і простий спосіб організації обміну даними між віджетами без потреби в надмірному наслідуванні чи ручному передаванні параметрів [16].

Крім того, в межах архітектури застосунку були використані такі шаблони:

- Singleton – для реалізації класів, які мають бути доступні глобально, зокрема сервіси роботи з Firebase або сховища даних;
- Repository – для абстрагування доступу до даних і створення єдиного інтерфейсу взаємодії з Firebase (наприклад, UserRepository, TestRepository);
- Factory Method – у випадках, коли потрібно створювати об'єкти в залежності від певного параметра (наприклад, генерація екрану залежно від ролі користувача: учень чи адміністратор).

Застосування цих шаблонів дало змогу розділити логіку на незалежні частини, спростити відлагодження, покращити читабельність коду і створити основу для подальшого масштабування. У майбутньому це дає змогу, наприклад, легко додати вебінтерфейс або розширити систему аналітики без повного переписування логіки застосунку.

2.4 Підключення додаткових сторонніх бібліотек

Для реалізації функціоналу мобільного застосунку з підготовки до мультипредметного тестування було використано набір сторонніх бібліотек, які забезпечили інтеграцію з хмарними сервісами, керування станом, побудову графіків, обробку інтерфейсу та підвищення зручності розробки.

Основні підключені бібліотеки:

1. firebase_core – основна бібліотека для ініціалізації Firebase у Flutter-застосунках. Необхідна для доступу до інших Firebase-сервісів.
2. cloud_firestore – забезпечує взаємодію з Cloud Firestore. Дозволяє зчитувати, додавати, редагувати й видаляти документи та колекції у базі даних Firebase.
3. firebase_auth – бібліотека для реалізації аутентифікації користувачів за допомогою електронної пошти та пароля. Також надає функціонал для реєстрації та скидання паролю.
4. provider – бібліотека для керування станом застосунку, що дозволяє будувати гнучку, реактивну архітектуру з розділенням відповідальностей.
5. charts_flutter – використовується для побудови графіків у вкладці статистики профілю користувача. Завдяки цій бібліотеці реалізовано візуалізацію прогресу у вигляді стовпчикових або лінійних діаграм.
6. intl – забезпечує форматування дати, часу та чисел згідно з регіональними стандартами. Застосовується для відображення дати проходження тесту у профілі користувача.

7. flutter_spinkit – використовується для реалізації анімацій завантаження, які покращують користувацький досвід у момент очікування відповіді від Firebase.

8. shared_preferences (опціонально) – може використовуватись для локального зберігання незначних даних, зокрема стану авторизації або останнього вибраного предмета.

9. fluttertoast – для виведення коротких повідомлень (toast) у разі помилок або успішних дій, наприклад: "Результат збережено", "Неправильний пароль", тощо.

Завдяки використанню зазначених бібліотек вдалося значно скоротити час розробки, підвищити стабільність роботи застосунку та реалізувати розширений функціонал без необхідності створювати все "з нуля". Усі бібліотеки були офіційно підтримані спільнотою, добре задокументовані та легко інтегруються з архітектурою Flutter.

2.5 Опис структури бази даних

Для зберігання даних у мобільному застосунку було використано хмарну базу даних Cloud Firestore, що є частиною платформи Firebase. Вибір саме цього рішення обумовлений його гнучкістю, масштабованістю та високою швидкістю роботи. База даних Firestore побудована на основі документо-орієнтованої моделі, де основними сутностями є колекції та документи. Кожна колекція може містити довільну кількість документів, які у свою чергу можуть містити вкладені підколекції.

У структурі проєкту реалізовано кілька основних колекцій. Колекція users зберігає інформацію про кожного зареєстрованого користувача, де ключем є унікальний ідентифікатор користувача, створений Firebase Authentication. У документі користувача зберігається електронна пошта, роль (наприклад, звичайний користувач або адміністратор), а також дата створення облікового запису. В межах кожного користувача створюється підколекція stats, яка містить

інформацію про проходження тестів, зокрема предмет, кількість правильних відповідей, загальну кількість запитань, відсоток успішності та дату тестування.

Колекція questions використовується для зберігання тестових завдань. Кожен документ у цій колекції містить текст запитання, масив варіантів відповідей, індекс правильної відповіді, а також позначення предмета, до якого належить це запитання. Завдяки такому підходу адміністратор може редагувати або додавати нові питання без необхідності оновлення коду застосунку.

Дані в базі є реактивними, тобто у разі зміни інформації в Firestore інтерфейс користувача оновлюється автоматично без перезапуску застосунку. Це досягається за рахунок використання потоків даних і віджетів StreamBuilder у Flutter. Таким чином, користувач завжди бачить актуальні результати у своєму профілі після проходження тесту.

Особливу увагу було приділено системі безпеки. У Firestore реалізовано механізм обмеження доступу до окремих колекцій і документів за допомогою правил безпеки. Вони визначають, хто має право читати або змінювати дані. Звичайні користувачі мають доступ лише до власної інформації, тоді як адміністратор отримує розширені можливості, зокрема право на редагування тестових завдань і перегляд загальної статистики.

Такий підхід до структури бази даних дозволяє забезпечити гнучкість у розширенні функціоналу, додаванні нових предметів, типів запитань або статистичних звітів без порушення цілісності даних. Firestore забезпечує високу швидкодію, стабільність і простоту підтримки застосунку на всіх етапах його використання.

2.6 UML-діграми

UML (Unified Modeling Language) – це стандартизована мова моделювання, яка широко використовується для візуалізації, опису та документування компонентів програмного забезпечення. Вона дозволяє чітко структурувати архітектурні рішення та продемонструвати логіку

функціонування інформаційної системи [17]. Завдяки UML можна відобразити як загальні функціональні можливості застосунку, так і взаємодію між окремими об'єктами, модулями, користувачами та зовнішніми сервісами.

Використання UML у межах проєктування даного мобільного застосунку є доцільним і обґрунтованим, оскільки дозволяє:

- представити основні сценарії використання системи (через діаграми варіантів використання);
- описати структуру внутрішніх компонентів (через діаграми класів);
- візуалізувати алгоритми взаємодії між об'єктами (через діаграми послідовності);
- спростити аналіз логіки виконання дій користувача (через діаграми активностей) [18].

Розробка UML-діаграм дозволяє не лише краще спланувати архітектуру, а й зробити систему більш прозорою для подальшої підтримки та розвитку. У наступних пунктах буде наведено основні типи діаграм, що використовувались під час проєктування мобільного застосунку для підготовки до мультипредметного тестування [19].

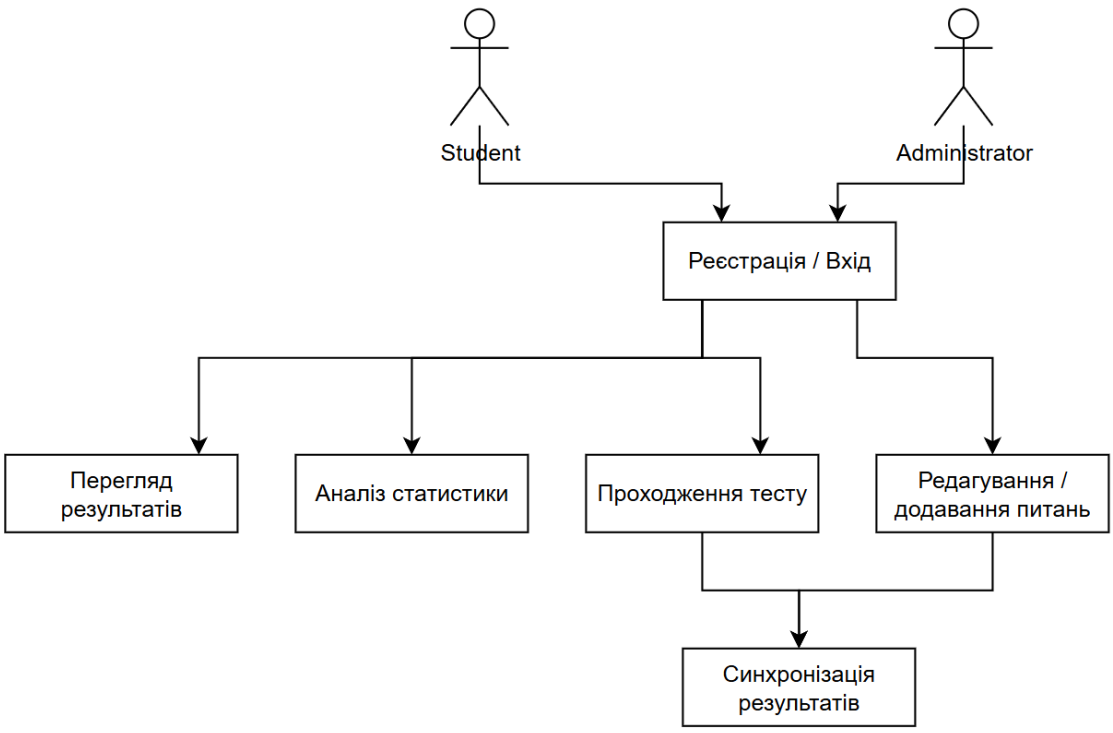


Рисунок 2.6 – Діаграма випадків використання мобільного застосунку

На рисунку 2.6 наведено діаграму випадків використання, яка відображає основні сценарії взаємодії користувачів із мобільним застосунком для підготовки до мультипредметного тестування.

Діаграма ілюструє двох акторів – учня (Student) та адміністратора (Administrator), які взаємодіють із системою. Обидва користувачі мають доступ до реєстрації та входу в застосунок. Після автентифікації учень може проходити тестування, переглядати результати, а також аналізувати власну статистику. У свою чергу, адміністратор, крім базових функцій, має доступ до редагування та додавання нових тестових запитань.

Усі результати тестування, а також статистика проходжень синхронізуються з базою даних автоматично. Цей процес взаємодії реалізовано через інтеграцію з Firebase, що виступає в ролі фонової системи зберігання та обробки даних.

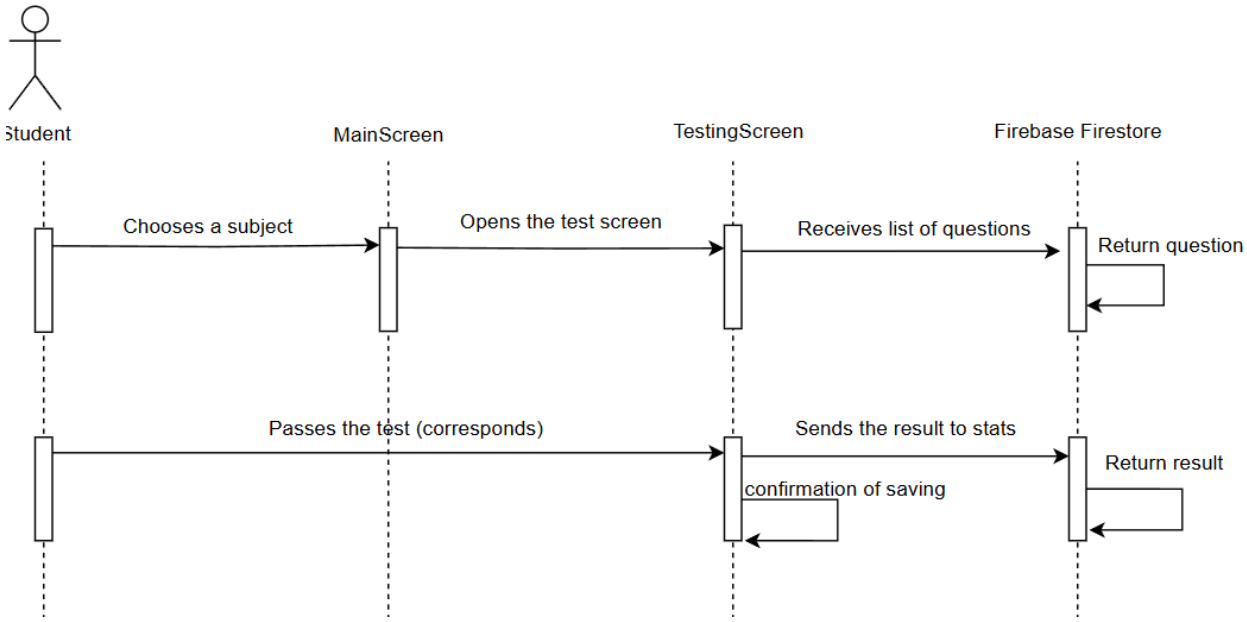


Рисунок 2.7 – Діаграма послідовності: взаємодії між учнем та інтерфейсом

На рисунку 2.7 представлено діаграму послідовності, яка ілюструє повний цикл взаємодії між учнем, інтерфейсом мобільного застосунку та базою даних під час проходження тесту й подальшого збереження результату. Цей сценарій є одним із основних у роботі системи, оскільки відображає ключову функцію –

перевірку знань користувача за допомогою тестових завдань і збереження його прогресу для подальшого аналізу.

Процес починається з того, що учень, перебуваючи на головному екрані застосунку, обирає предмет, з якого бажає пройти тестування. Ця дія ініціює перехід до екрану тестування, де система формує відповідний запит до хмарної бази даних Firestore з метою отримання актуального списку питань. Firestore обробляє запит і повертає набір питань, відсортованих за категорією або обраним предметом.

Після завантаження питань учень починає послідовно відповідати на них. На цьому етапі екран тестування накопичує відповіді користувача, підраховує правильні й неправильні варіанти та формує фінальний результат. Після завершення тесту дані автоматично структуруються у вигляді об’єкта та передаються назад у Firestore, у вкладену підколекцію stats, яка зберігається в межах документа відповідного користувача.

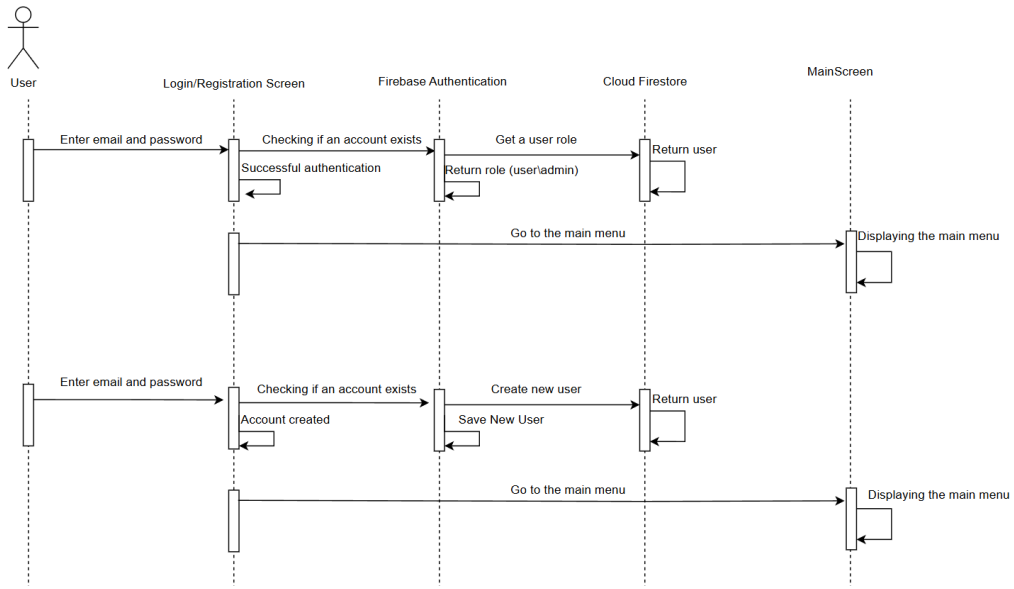


Рисунок 2.8 – Діаграма послідовності: реєстрація та вхід користувача із визначенням ролі

Firestore приймає й обробляє запит, після чого надсилає підтвердження про успішне збереження. Після цього застосунок переходить до етапу візуалізації

результату, відображаючи учневі кількість правильних відповідей, відсоток успішності та короткий підсумок. Уся ця взаємодія відбувається без потреби у перезапуску застосунку або ручному оновленні – завдяки реактивному підходу та використанню потокових механізмів.

Дана послідовність демонструє ефективну взаємодію між клієнтською частиною застосунку й базою даних, що є фундаментальним елементом функціональності освітньої системи. Завдяки автоматичному збереженню результатів та їх синхронізації користувач має змогу переглядати історію своїх проходжень і відстежувати власний прогрес у реальному часі.

На рисунку 2.8 подано діаграму послідовності, яка описує процес реєстрації або входу користувача до мобільного застосунку, а також визначення його ролі в системі. Даний сценарій є базовим для будь-якого користувача, оскільки саме з автентифікації починається взаємодія із застосунком.

Після запуску застосунку користувач переходить до екрана входу або реєстрації, де вводить свою електронну пошту та пароль. Застосунок надсилає запит на перевірку до модуля Firebase Authentication. Якщо обліковий запис уже існує, система здійснює автентифікацію користувача та ініціює запит до бази даних Firestore для зчитування ролі, закріпленої за цим користувачем. Отримана роль (наприклад, "user" або "admin") передається назад на клієнтську частину, після чого користувач перенаправляється до відповідного головного меню.

У разі, якщо обліковий запис не існує, застосунок ініціює процес реєстрації: Firebase створює нового користувача та додає запис у базу Firestore, зазвичай із роллю "user" за замовчуванням. Після успішного створення акаунту виконується перехід до основного меню.

Ця логіка дозволяє ефективно розділяти повноваження в системі. Наприклад, адміністратору доступні функції редагування запитань і керування базою, тоді як звичайний користувач має можливість лише проходити тести й переглядати результати. Таким чином, діаграма відображає не лише процес входу, а й перший крок до реалізації рольової моделі в межах архітектури застосунку.

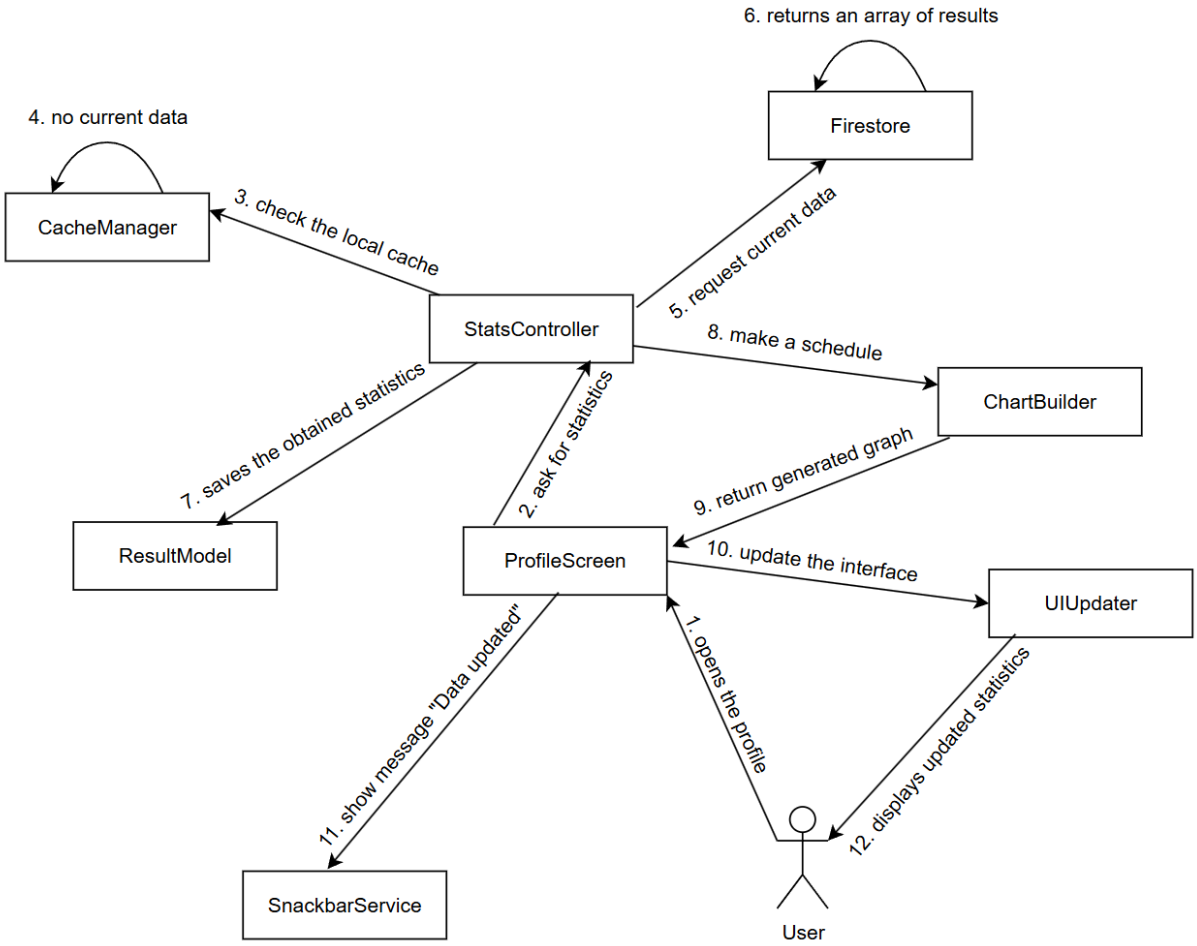


Рисунок 2.9 – Діаграма кооперації: перегляд статистики користувача

На рисунку 2.9 зображено діаграму кооперації, яка відображає процес формування статистики користувача в профілі мобільного застосунку. Дана діаграма демонструє взаємодію між різними логічними об'єктами системи під час виконання запиту на перегляд попередніх результатів тестування та їх візуалізації у вигляді графіка.

Процес розпочинається з ініціативи користувача, який відкриває вкладку профілю в застосунку. Це запускає екран ProfileScreen, який надсилає запит на отримання статистичних даних до логічного модуля StatsController. Контролер спочатку перевіряє наявність актуальних даних у локальному сховищі за допомогою об'єкта CacheManager. Якщо свіжі дані відсутні, здійснюється запит до хмарної бази Firestore для отримання повного масиву результатів.

Після успішного отримання даних контролер передає їх до ResultModel, де вони зберігаються у внутрішній структурі. Паралельно статистичні показники передаються об'єкту ChartBuilder, який формує візуалізацію у вигляді графіка. Згенерований графік надсилається назад до ProfileScreen, після чого ініціюється оновлення інтерфейсу за допомогою UIUpdater.

Щоб користувач був поінформований про оновлення даних, екран ProfileScreen взаємодіє з SnackbarService, який виводить коротке повідомлення "Дані оновлено". У фінальному етапі користувач бачить оновлену статистику, представлену у вигляді графіків або списків, що дозволяє відстежити динаміку особистих результатів.

Уся система працює реактивно, забезпечуючи швидке завантаження, відображення та оновлення даних, що підвищує зручність користування та інформативність інтерфейсу.

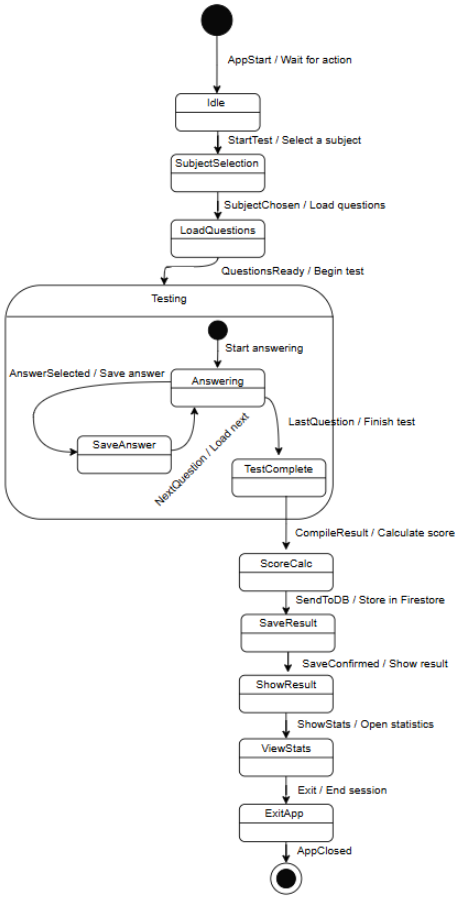


Рисунок 2.10 – Діаграма станів: логіка проходження тесту користувачем

На рисунку 2.10 зображено діаграму станів, яка описує динамічну поведінку мобільного застосунку в процесі проходження тестування. Дана діаграма ілюструє послідовну зміну станів, через які проходить система – від початкового запуску до завершення сеансу після перегляду результатів та статистики.

Сценарій починається зі стану очікування дії користувача (Idle), де система перебуває після запуску застосунку. При натисканні на кнопку початку тесту ініціюється вибір предмета (SubjectSelection), після чого система переходить у стан завантаження відповідних тестових завдань (LoadQuestions). У разі успішного отримання запитань застосунок переходить до вкладеного стану Testing, де реалізовано основну логіку тестування.

У межах Testing система перебуває у стані Answering, де користувач поетапно відповідає на питання. Після кожної відповіді здійснюється перехід до стану SaveAnswer, який відповідає за збереження даних, а потім – повернення до наступного запитання або, якщо тест завершено, перехід до стану TestComplete.

Після цього активується модуль обчислення результату (ScoreCalc), який виконує підрахунок кількості правильних відповідей та відсоткового результату. Далі інформація передається у Firestore у стані SaveResult, і після підтвердження збереження користувачу відображається екран результатів (ShowResult). Користувач має змогу перейти до перегляду статистики (ViewStats), а завершення сеансу відбувається у стані ExitApp, після чого система повертається у фінальний стан (AppClosed).

Дана діаграма дозволяє чітко відслідкувати логіку внутрішніх переходів між ключовими етапами застосунку, забезпечуючи повну прозорість функціонування механізму тестування.

2.7 Висновки

У межах другого розділу було здійснено комплексне проєктування архітектури мобільного застосунку для підготовки до мультипредметного

тестування. Розглянуто та обґрунтовано вибір архітектурних рішень, шаблонів проєктування, сторонніх бібліотек, а також структури бази даних.

Використання архітектурного підходу, що базується на розділенні логіки, представлення та зберігання даних, забезпечило зручність підтримки коду, гнучкість у розширенні функціоналу та стабільність роботи. Розроблена база даних у Firebase Firestore чітко структурує інформацію за користувачами, результатами, питаннями та предметами, що дозволяє ефективно реалізувати адаптивність тестування та зберігати детальну статистику.

Окрему увагу було приділено побудові UML-діаграм, серед яких: діаграма випадків використання, послідовності, кооперації та діаграма станів. Вони відображають ключові сценарії роботи системи, взаємодію компонентів та внутрішню логіку роботи застосунку.

Таким чином, результати проєктування архітектури стали надійною основою для подальшої реалізації, тестування та використання застосунку, а також забезпечили високий рівень структурованості, модульності та адаптивності системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

У цьому розділі розглянуто особливості практичної реалізації мобільного застосунку, призначеного для підготовки до мультипредметного тестування з адаптивною перевіркою знань. Основна увага приділена структуруванню логіки програми, взаємодії між її компонентами, підключенню до бази даних, реалізації автентифікації користувачів, механізмам адаптації запитань на основі результатів, а також прикладам реальної роботи інтерфейсу.

Розробка здійснювалась із використанням фреймворку Flutter, що дозволило створити кросплатформенний застосунок, придатний для Android та iOS. У якості хмарного сховища та сервісу автентифікації використовувалась платформа Firebase, яка надала функціональні можливості для зберігання результатів, структурування тестових даних та керування обліковими записами користувачів без необхідності окремої серверної частини.

Архітектура застосунку побудована відповідно до принципів модульності: кожен екран або функціональний блок має власну логіку, що підвищує зручність підтримки та масштабування проєкту. Особлива увага приділялась побудові адаптивного алгоритму, який дозволяє системі реагувати на успішність користувача під час проходження тесту та підбирати завдання відповідної складності.

У підрозділах цього розділу наведено реалізацію ключових модулів застосунку, зокрема системи автентифікації, логіки тестування, обробки результатів, адаптивного аналізу та формування рекомендацій. Також наведено приклади роботи інтерфейсу, що ілюструють функціональність готового застосунку.

3.1 Структура та логіка взаємодії компонентів інтерфейсу

Мобільний застосунок побудований на базі Flutter із застосуванням компонентного підходу, де кожен функціональний елемент або екран реалізовано у вигляді окремого віджета з власною відповідальністю. Така архітектура забезпечує зручність розробки, повторне використання коду, простоту тестування та легкість масштабування функціоналу в майбутньому.

Інтерфейс поділяється на низку ключових компонентів, які взаємодіють між собою за допомогою навігаційного стеку та потоків даних. Основними складовими є:

— Екран входу/реєстрації – забезпечує автентифікацію користувача через Firebase Authentication. Після успішного входу здійснюється перенаправлення до головного меню.

— Головне меню – надає вибір предмета тестування, доступ до профілю, перегляду результатів та інших функцій.

— Екран тестування – реалізує логіку проходження тесту: поетапне виведення запитань, обробку відповідей, підрахунок правильних результатів.

— Модуль адаптації – на основі відповідей користувача автоматично підбирає запитання зі схожими помилками з попередніх тестів, забезпечуючи персоналізований підхід до навчання.

— Профіль користувача – відображає загальну статистику успішності та надає доступ до результатів попередніх тестувань.

— Інтерфейс адміністратора (лише для облікового запису з логіном "admin") – дозволяє додавати, редагувати або видаляти тестові завдання.

Обмін даними відбувається через Firestore – усі ключові події (збереження відповідей, результатів, історії проходження) синхронізуються з хмарною базою в реальному часі. Це дозволяє забезпечити збереження прогресу користувача та миттєве оновлення інтерфейсу при зміні даних.

Також передбачено кешування проміжних даних, що дозволяє зберігати відповіді при нестабільному інтернет-з’єднанні, з подальшою синхронізацією при його відновленні.

У наступних підрозділах буде детально описано реалізацію кожного з основних функціональних модулів, включаючи систему автентифікації, тестування та адаптивну перевірку знань.

3.1.1 Реалізація автентифікації користувачів

Система автентифікації є обов’язковим елементом інформаційної безпеки у будь-якому освітньому застосунку, що передбачає збереження особистих результатів, прогресу користувача та доступ до персоніфікованих функцій. У межах розробленого мобільного застосунку реалізовано базову, але надійну автентифікацію з використанням платформи Firebase Authentication.

Основною перевагою використання Firebase є те, що даний сервіс забезпечує повний цикл реєстрації, входу та управління обліковими записами без потреби в реалізації власного серверного бекенду або зберігання паролів на клієнтській стороні. Усі паролі зберігаються в зашифрованому вигляді на сервері Google, що значно підвищує рівень безпеки.

Процес автентифікації передбачає два базових сценарії:

- Реєстрація нового користувача – введення електронної пошти та паролю з подальшим створенням облікового запису.
- Авторизація – вхід до системи з уже зареєстрованими обліковими даними.

У випадку успішної автентифікації здійснюється перевірка ролі користувача (звичайний учень або адміністратор). Це реалізовано через зчитування відповідного поля role у базі Firestore. Якщо користувач має роль admin, йому відкривається додатковий функціонал для управління базою тестових запитань [20].

Для реалізації автентифікації у Flutter-застосунку було використано офіційний SDK firebase_auth, що дозволяє виконувати такі дії:

- Створення нового користувача (createUserWithEmailAndPassword)
- Вхід до системи (signInWithEmailAndPassword)
- Вихід із системи (signOut)
- Відстеження змін статусу входу користувача (authStateChanges)

Після успішної авторизації UID користувача використовується як унікальний ідентифікатор для доступу до його даних у Firestore, зокрема статистики, результатів тестів та особистих налаштувань.

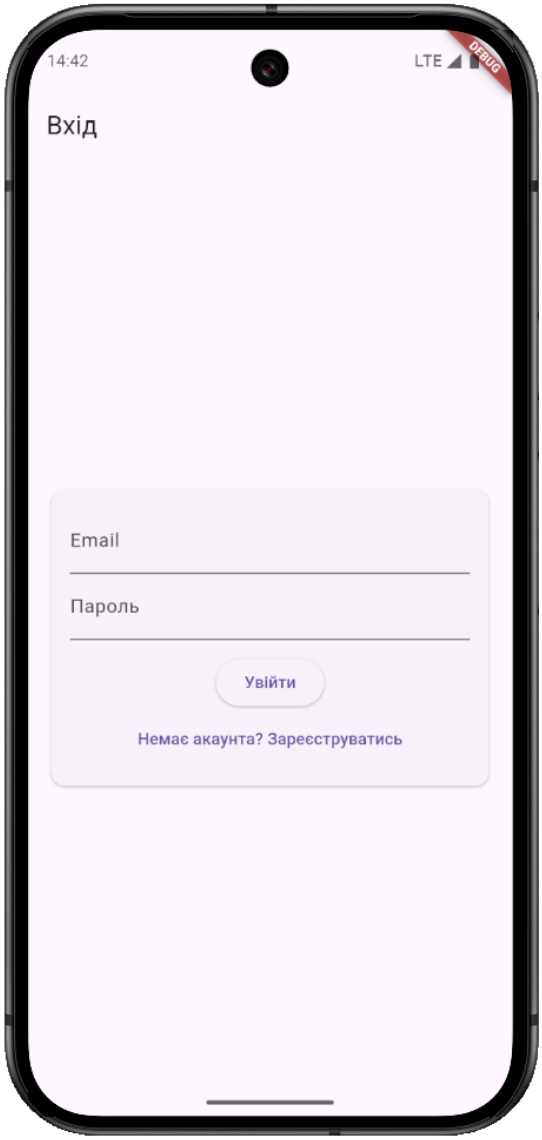


Рисунок 3.1 - Екран авторизації у мобільному застосунку

Для захисту доступу до різних розділів застосунку використовуються перевірки прав доступу, які реалізовано як у коді застосунку, так і в правилах безпеки Firestore. Таким чином, доступ до певних функцій обмежено залежно від типу користувача, що дозволяє гнучко розділяти ролі та запобігати несанкціонованому редагуванню даних. На рисунку 3.1 представлено екран авторизації, який є початковою точкою входу до системи та визначає подальший доступ користувача до функціоналу відповідно до його ролі.

```
if (_isLogin) {
  userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
    email: _emailController.text.trim(),
    password: _passwordController.text.trim(),
  );
```

Рисунок 3.2 – Фрагмент коду автентифікації з використанням Firebase Authentication

На рисунку 3.2 зображено фрагмент коду, що реалізує процес входу користувача до мобільного застосунку за допомогою сервісу Firebase Authentication. Метод signInWithEmailAndPassword викликається з параметрами електронної пошти та пароля, які попередньо введено користувачем у формі. У разі успішної автентифікації повертається об'єкт UserCredential, який використовується для подальших дій – зокрема, отримання ролі користувача та навігації до відповідного інтерфейсу.

3.1.2 Реалізація логіки проходження тестів

Функціонал проходження тестів є ключовою складовою мобільного застосунку для підготовки до мультипредметного тестування. Реалізація цього модуля охоплює послідовне відображення питань, обробку відповідей користувача, підрахунок результатів, збереження статистики та подальший перехід до екрана результатів.

У структурі застосунку виділено окремий екран TestScreen, який відповідає за виведення одного питання за раз. При переході на цей екран з головного меню, система отримує список питань з Firebase на основі обраного предмета. Запитання мають структуру, яка включає текст питання, п’ять варіантів відповідей та індекс правильної.

При відповіді на кожне запитання система:

1) перевіряє, чи відповідь правильна;
2) збільшує лічильник правильних відповідей;
3) фіксує вибір користувача для подальшого аналізу;
4) оновлює стан для переходу до наступного запитання.

Після останнього запитання система переходить у стан обробки результату: підраховується загальна кількість правильних відповідей, визначається відсоток успішності, а також оцінка у зручній для сприйняття формі (наприклад, кількість балів із загального числа) [21].

Отримані дані зберігаються у Firestore у вигляді окремого документу у вкладеній колекції stats, прив’язаній до ідентифікатора користувача. При збереженні фіксуються:

- дата проходження тесту;
- назва предмета;
- кількість правильних відповідей;
- загальна кількість питань;
- відсоток успішності.

Це дозволяє будувати історію результатів для кожного користувача та використовувати її у вкладці статистики профілю.

Для візуального переходу між запитаннями використовується механізм setState, який оновлює інтерфейс без повного перезавантаження сторінки. По завершенню тесту користувач автоматично перенаправляється до екрана ResultScreen, де бачить свій результат, а також має можливість перейти до повної статистики або повторно пройти тест.

Крім цього, у логіку проходження тесту включено базову адаптивність: якщо в історії тестувань виявлено систематичні помилки в певних темах, під час наступного запуску тесту система з більшою ймовірністю запропонує подібні запитання. Ця логіка реалізована через фільтрацію питань за тегами та пріоритетне вибіркове завантаження з бази.

Таким чином, реалізація модуля проходження тестів не лише виконує функцію контролю знань, але й дозволяє підлаштовуватись під користувача, фіксувати прогрес та сприяти ефективному повторенню матеріалу.

Рисунок 3.3 – Екран проходження тестового завдання

На рисунку 3.3 показано інтерфейс проходження тесту. Користувачу надається питання з варіантами відповідей (у цьому прикладі – з теми математики, що містить зображення графіка функції). Питання відображається по одному, що дозволяє сфокусуватись на конкретному завданні. Вибір варіанту автоматично зберігається та ініціює перехід до наступного питання.

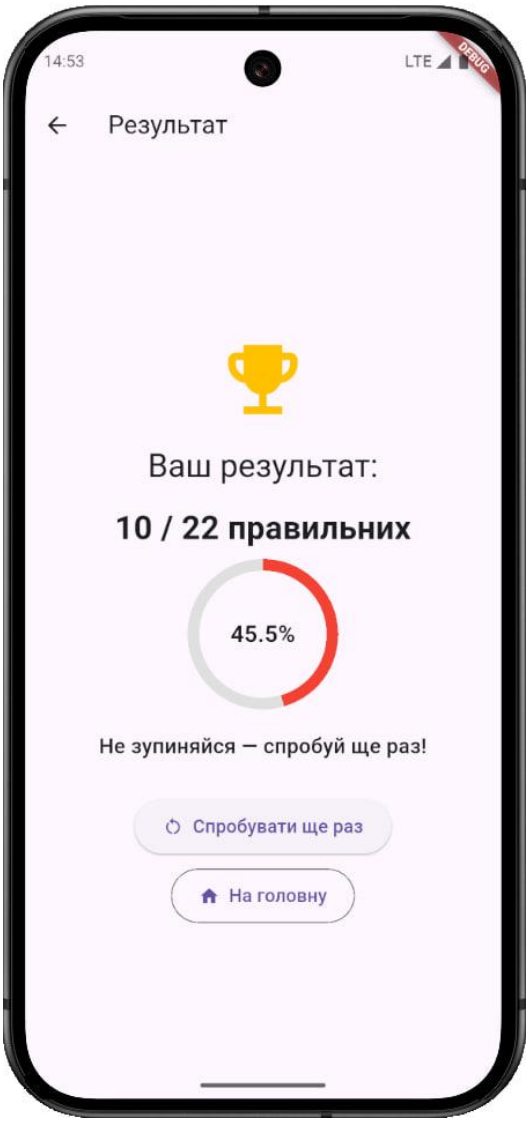


Рисунок 3.4 – Екран результату після завершення тесту

Рисунок 3.4 демонструє результат проходження тесту. Користувач бачить кількість правильних відповідей та відсоток успішності, що обчислюється автоматично на основі збережених відповідей. Також передбачено зворотний

зв’язок у вигляді мотиваційного повідомлення та можливість пройти тест ще раз або повернутись на головний екран.

3.1.3 Адаптивна перевірка знань та рекомендації

Однією з ключових особливостей розробленого мобільного застосунку є реалізація елементарної адаптивності при формуванні набору тестових завдань, а також можливість аналізу результатів для надання користувачу рекомендацій щодо повторення тем.

Адаптивна перевірка знань реалізована не як повноцінна система з використанням штучного інтелекту чи машинного навчання, а як логічна модель аналізу минулих результатів. Після завершення кожного тестування система зберігає дані про відповіді користувача, включаючи:

- дату проходження;
- назву предмета;
- індекси питань, у яких були допущені помилки;
- відсоток правильних відповідей.

Ці дані зберігаються у Firestore у вигляді вкладеної колекції stats, що дозволяє надалі аналізувати історію проходження тестів для кожного користувача.

Під час формування нового тесту система перевіряє, чи є в історії користувача помилки по запитаннях певного типу або тематики (ідентифікованої, наприклад, через поле tag або category у базі). Якщо такі знайдено, то система з певною пріоритетністю додає подібні питання до нового набору. Таким чином, користувач частіше стикається з темами, які викликали у нього труднощі раніше, що сприяє повторенню та кращому засвоєнню матеріалу.

Крім того, у профілі користувача реалізована вкладка Статистика, де відображаються графіки прогресу по предметах. Це дозволяє самостійно відслідковувати успішність, бачити покращення або виявляти тенденції до

зниження результатів. Візуалізація створюється на основі збережених об'єктів stats, і будується динамічна діаграма (наприклад, кругова або гістограма), яка відображає частку правильних відповідей у кожному тесті.

Загалом, навіть у простій формі така адаптивна система дає змогу персоналізувати навчання та зробити підготовку більш ефективною, що особливо важливо в умовах обмеженого часу перед іспитами.

3.1.4 Збереження результатів і побудова статистики

Після завершення проходження тесту застосунок автоматично зберігає результати користувача у хмарну базу даних Firebase Firestore. Такий підхід дозволяє не лише забезпечити збереження прогресу, але й формувати динамічну статистику результатів тестування.

Збереження результатів

Кожен користувач має у Firestore окремий документ у колекції users, що містить персональні дані. До цього документа належить підколекція stats, де кожен документ відповідає одному тестуванню. Структура запису включає такі поля:

- subject – назва предмета (наприклад, "Математика");
- score – кількість правильних відповідей;
- total – загальна кількість запитань у тесті;
- date – дата й час проходження тесту.

Збереження результатів відбувається одразу після завершення тесту, без додаткових дій з боку користувача.

Побудова статистики

Для відображення успішності користувача реалізовано окрему вкладку "Статистика", у якій показується:

- список усіх пройдених тестів із зазначенням дати, предмета та кількості правильних відповідей;
- візуалізований графік прогресу користувача.

Графік будується на основі даних з підколекції stats. Було використано бібліотеку charts_flutter, яка дозволяє формувати інтерактивну стовпчикову діаграму. Кожен стовпчик відповідає одному тесту, а його висота – кількості правильних відповідей.

Цей механізм допомагає користувачеві бачити, які предмети вимагають додаткової підготовки, а також відстежувати власний прогрес у динаміці.

3.1.5 Обробка винятків та UX-оптимізація

Для забезпечення надійної роботи застосунку особливу увагу було приділено обробці виняткових ситуацій, а також покращенню зручності користування (UX) у процесі взаємодії з інтерфейсом.

Обробка виняткових ситуацій

Застосунок передбачає декілька типових ситуацій, які можуть виникнути під час роботи:

- Відсутність вибору відповіді: якщо користувач намагається перейти до наступного питання, не обравши жодного варіанту, система виводить повідомлення з попередженням і не дозволяє рухатися далі до моменту вибору.
- Втрачений інтернет-зв’язок: у разі розриву з’єднання під час збереження результатів, застосунок повідомляє про помилку і пропонує повторити спробу після відновлення доступу до мережі.
- Вихід з облікового запису або зупинка тесту: реалізовано підтвердження дій, які можуть призвести до втрати прогресу (наприклад, повернення на головний екран під час проходження тесту).

Усі винятки опрацьовуються з використанням конструкцій try-catch, а користувач отримує відповідні повідомлення через SnackBar або діалогові вікна.

UX-оптимізація

Для покращення користувацького досвіду було реалізовано низку функціональних і візуальних рішень:

- Автоматичне прокручування: у випадку довгих питань або малих екранів реалізовано автоматичну прокрутку до активного блоку.
- Візуальні підказки: активна відповідь підсвічується кольором, що підвищує наочність вибору.
- Підтвердження важливих дій: перехід до результатів, вихід із тесту або обнулення статистики супроводжуються підтвердженням.
- Тематика інтерфейсу: використано стримані кольори, чітку типографіку, великі клікабельні елементи для зручності взаємодії на сенсорних пристроях.

Ці заходи забезпечують не лише стабільну роботу застосунку, а й підвищують довіру користувача до системи та комфорт під час навчання.

3.2 Тестування застосунку та порівняння системи з аналогами

Тестування розробленого застосунку підтвердило його стабільність, зручність використання та відповідність поставленим вимогам. Функціональні тести засвідчили коректну роботу проходження тестів, збереження результатів у Firestore та формування статистики користувача. Перевірка механізму адаптивного підбору запитань показала, що складність коригується залежно від рівня знань користувача.

Тестування інтерфейсу підтвердило його зручність як для новачків, так і для досвідчених користувачів — усі основні дії (вибір предмета, відповіді, перегляд результатів) виконуються інтуїтивно зрозуміло. Було проведено перевірку на різних розмірах екранів, і застосунок коректно адаптується до смартфонів із різними параметрами.

Збереження статистики проходження тестів у Firestore відбулося без втрат, а побудова графіків на основі зібраних даних продемонструвала стабільну інтеграцію з бібліотеками візуалізації. Регресійне тестування після оновлення бази запитань показало збереження працездатності основного функціоналу.

Виявлені дрібні недоліки (затримка виведення результату при великій кількості запитань, некоректне відображення кнопки «Далі» у рідкісних випадках) були усунені шляхом оптимізації логіки інтерфейсу.

Порівняння з аналогами:

Порівняємо розроблений мобільний застосунок із поширеними онлайн-ресурсами для підготовки до НМТ. У таблиці 3.1 наведено порівняння функціональних можливостей розробленого мобільного застосунку з існуючими платформами для підготовки до мультипредметного тестування.

Таблиця 3.1 – Порівняння мобільних застосунків для підготовки до мультипредметного тестування

Система	Адаптивність	Статистика користувача	Аналіз помилок	Мобільний доступ	Предмети
Розроблений застосунок	✓	✓	×	✓	6
ZNO.ua	×	×	×	✓	3
iLearn	×	×	×	✓	4
Освіта.ua	×	×	×	✓	3
Studinfo.org	×	✓(частково)	×	×	3

Розроблена система демонструє перевагу завдяки адаптивному підходу до перевірки знань, зручному мобільному інтерфейсу, підтримці збереження особистої статистики та можливості аналізу результатів із подальшими рекомендаціями.

3.3 Приклад роботи застосунку

У цьому підрозділі представлено приклади роботи основних функціональних модулів мобільного застосунку, реалізованого у межах бакалаврського кваліфікаційного проєкту. Застосунок дозволяє користувачам проходити тестування з різних предметів, отримувати миттєвий результат, переглядати статистику та працювати над помилками.

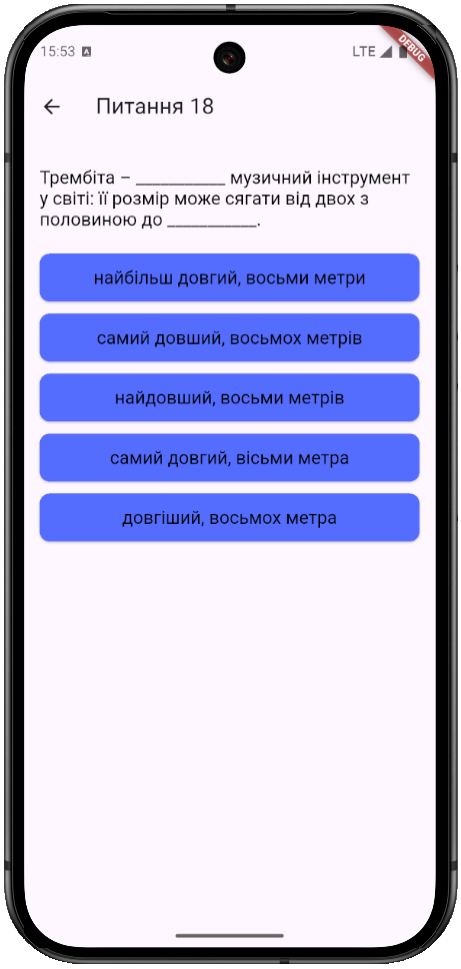


Рисунок 3.5 – Екран проходження тестового запитання

На рисунку 3.5 зображено приклад проходження тесту. Користувачу подається запитання з варіантами відповідей, серед яких потрібно обрати правильний. Інтерфейс розроблений у простому та зрозумілому стилі, з урахуванням адаптивності для різних розмірів екранів.

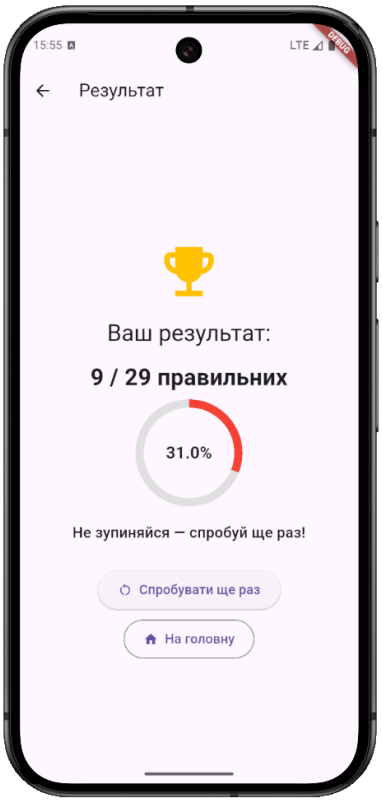


Рисунок 3.6 – Відображення результатів після завершення тесту

Після завершення тестування відображається екран з результатами, як показано на рисунку 3.6. Користувач бачить кількість правильних відповідей, відсоток успішності та можливість повторити тест або повернутись на головну сторінку.

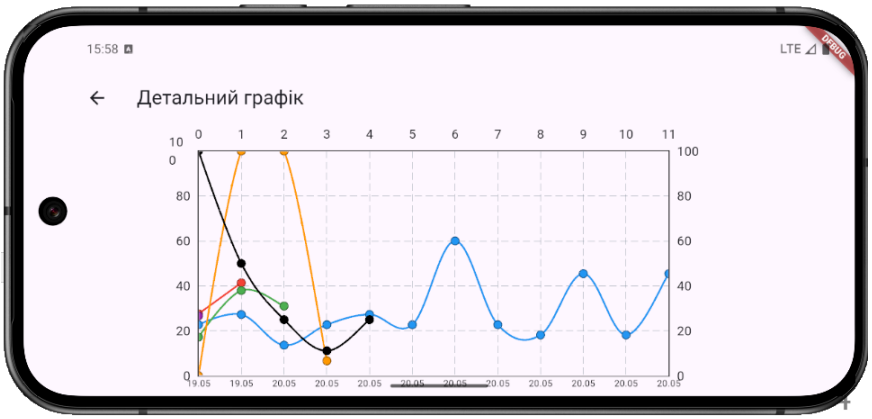


Рисунок 3.7 – Інтерактивна візуалізація прогресу користувача у повноекранному режимі

У вкладці “Профіль” користувач має доступ до розширеної статистики проходжень. Графік динаміки успішності побудований на основі Firebase Firestore і виводиться у вигляді лінійної діаграми, як показано на рисунку 3.7. Графік інтерактивний: натискання на нього відкриває повноекранний режим, у якому можна гортати результат вліво та вправо.

Кожна точка на графіку відповідає окремому сеансу тестування з певного предмета. Це дозволяє наочно оцінити тенденцію покращення або спадання рівня підготовки та, відповідно, скоригувати навчальний процес.

3.4 Інструкція користувача

Мобільний застосунок розроблено таким чином, щоб забезпечити інтуїтивно зрозумілу взаємодію з користувачем. Навіть новачок, який не має попереднього досвіду роботи з подібними програмами, зможе легко пройти тестування, переглянути результати та отримати рекомендації.

Реєстрація та вхід

Після встановлення застосунку користувач має можливість створити обліковий запис або увійти до існуючого через email та пароль. У разі першого входу система зберігає профіль користувача у базі даних Firebase.

Головне меню

Після авторизації користувач потрапляє до головного меню, яке складається з кількох вкладок:

- "Тести" – для вибору предмета і проходження тестування;
- "Статистика" – перегляд результатів попередніх тестів у вигляді списку і графіка;
- "Профіль" – дані користувача та можливість виходу з облікового запису.

Проходження тесту

1. Користувач переходить у вкладку "Тести".

2. Обирає предмет із запропонованого списку (наприклад: математика, історія, біологія тощо).

3. Починається тестування – на кожному екрані відображається запитання та 5 варіантів відповіді.

4. Користувач обирає відповідь і натискає кнопку "Далі" для переходу до наступного питання.

5. Після завершення тесту на екрані з’являється результат: кількість правильних відповідей, загальна кількість питань та коротка рекомендація.

6. Дані автоматично зберігаються у хмарній базі та використовуються для формування загальної статистики.

Перегляд статистики

У вкладці "Статистика" користувач може побачити:

- список пройдених тестів із датами;
- відсоток правильних відповідей;
- діаграму прогресу за предметами.

Це дозволяє відстежувати покращення результатів і виявляти слабкі місця в підготовці.

Вихід із системи

Для виходу із застосунку потрібно перейти до вкладки "Профіль" і натиснути кнопку "Вийти". При повторному вході потрібно знову ввести логін і пароль.

3.5 Висновки

У четвертому розділі було представлено реалізацію мобільного застосунку, що повністю відповідає раніше сформульованим вимогам та обраній архітектурній моделі. Розробка здійснювалась із використанням сучасного інструментарію: для створення інтерфейсу було використано Flutter, що дозволяє забезпечити стабільну роботу на різних платформах, а для організації серверної

частини та зберігання даних застосовано можливості Firebase — зокрема, для автентифікації, роботи з базою та синхронізації користувацьких даних.

Одним із пріоритетів під час реалізації стала зручність інтерфейсу. Була врахована потреба в адаптації елементів до різних розмірів екранів, а також реалізовано логіку тестування з урахуванням гнучкості й простоти для користувача. Користувач може авторизуватись, обирати предмети для тестування, переглядати свої результати та аналізувати прогрес у власному профілі. Окремо було реалізовано інтерактивну статистику у вигляді графіка, який можна переглядати на весь екран, масштабувати та гортати, що значно підвищує зручність вивчення результатів.

Додатковою перевагою є реалізація системи ролей: користувачі мають статус або звичайного учня, або адміністратора. Останній має можливість керувати базою запитань — додавати, редагувати або видаляти вміст. Архітектура коду побудована за принципами модульності, що забезпечує гнучкість, повторне використання елементів та простоту подальшого масштабування.

Таким чином, реалізований мобільний застосунок підтверджує ефективність обраного технологічного підходу. Він працює стабільно, має зрозумілий дизайн і виконує всі ключові функції, необхідні для ефективної підготовки до мультипредметного тестування, поєднуючи зручність користування з реальним навчальним результатом.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було спроєктовано, реалізовано та протестовано мобільний застосунок для підготовки учнів до мультипредметного тестування, що відповідає сучасним вимогам до освітніх програмних засобів. Основна мета проєкту – створення зручного, адаптивного та функціонального інструменту для самостійної підготовки – була повністю досягнута.

У ході дослідження проведено аналіз предметної області, розглянуто існуючі рішення та виявлено їх недоліки, зокрема відсутність персоналізації та гнучкого аналізу помилок. Ці аспекти стали базисом для формування вимог до власного програмного продукту. Результатом стала реалізація системи, яка враховує попередні результати користувача та адаптує подальші запитання відповідно до виявлених прогалин у знаннях.

Для реалізації інтерфейсу було обрано фреймворк Flutter, що забезпечив кросплатформенну сумісність, високу швидкодію та зручну візуалізацію. Firebase виступив як основа для зберігання даних та аутентифікації, що дозволило створити надійне та масштабоване серверне середовище без додаткових витрат на інфраструктуру.

Особливу увагу приділено візуалізації статистики користувача. Було реалізовано інтерактивний графік прогресу, який відображає результати за всіма предметами у динаміці. Це дозволяє користувачеві наочно спостерігати за успішністю та визначати теми, які потребують повторення.

У результаті виконаної роботи:

- розроблено архітектуру та функціональні модулі мобільного застосунку;
- реалізовано систему реєстрації, проходження тестів, перегляду результатів та аналізу помилок;
- здійснено тестування функціональних можливостей та стабільності роботи застосунку.

Запропонований застосунок може бути рекомендований для індивідуального користування старшокласниками, а також у якості допоміжного інструменту для вчителів при організації підготовки до національного мультипредметного тесту. Крім того, звіт про виконану роботу було складено відповідно до вимог методичних вказівок, що гарантує його структурованість, логічність складання матеріалу та відповідність академічним стандартам [22].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Міщук М.Д., Богач І.В. Аналіз труднощів у підготовці до мультипредметного тестування та шляхи їх подолання. Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025). URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24452/20213 (дата звернення: 17.05.2025)

2. Цифрові рішення для індивідуалізації навчального процесу в умовах масового тестування /М.Д.Міщук, І.В.Богач// Матеріали міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». Збірник доповідей [Електронний ресурс], Вінниця: ВНТУ, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25426 (дата звернення: 06.06.2025)

3. Постанова Кабінету Міністрів України №1392 від 27.12.2023 «Про затвердження Порядку проведення національного мультипредметного тесту у 2024 році».URL: https://zakon.rada.gov.ua/laws/show/1392-2023-%D0%BF#Text

4. Міністерство освіти і науки України. Інформація про національний мультипредметний тест (НМТ). URL: https://testportal.gov.ua/nmt/. (дата звернення: 06.06.2025)

5. ZNO.ua. URL: https://zno.osvita.ua/ (дата звернення: 06.06.2025)

6. ILearn. URL: https://ilearn.org.ua/ (дата звернення: 06.06.2025)

7. Math Corporation. URL: https://www.mathcorporation.com/ (дата звернення: 06.06.2025)

8. Osvita.ua. URL: https://osvita.ua/test/ (дата звернення: 06.06.2025)

9. Studinfo. URL: https://studinfo.org/materials (дата звернення: 06.06.2025)

10. Flutter documentation URL: https://flutter.dev/ (дата звернення: 06.06.2025)

11. Firebase documentation URL: https://firebase.google.com/ (дата звернення: 06.06.2025)

12. Firestore Data Model: An Easy Guide URL: https://hevodata.com/learn/firestore-data-model/ (дата звернення: 07.06.2025)

13. Getting Started with Firebase Firestore in Flutter. URL: https://medium.com/@samra.sajjad0001/getting-started-with-firebase-firestore-in-flutter-a-comprehensive-guide-with-crud-operations-ec75f2188355 (дата звернення: 07.06.2025)

14. Setting Up Flutter in Visual Studio Code: A Step-by-Step Guide URL: https://medium.com/@hanansani2002/setting-up-flutter-in-visual-studio-code-a-step-by-step-guide-1c64450728a7 (дата звернення: 07.06.2025)

15. Dart Programming Language Guide URL: https://www.geeksforgeeks.org/dart-tutorial/ (дата звернення: 07.06.2025)

16. Design Patterns: Elements of Reusable Object-Oriented Software: монографія / Gamma E., Helm R., Johnson R., Vlissides J. Boston : Addison-Wesley, 1994. 395 с. (дата звернення: 07.06.2025)

17. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти URL: https://dou.ua/forums/topic/40575/ (дата звернення: 07.06.2025)

18. Для чого потрібні UML діаграми? UML: https://foxminded.ua/uml-diagramy/ (дата звернення: 09.06.2025)

19. UML Distilled: a brief guide to the standard object modeling language: монографія / Martin Fowler. Boston : Addison-Wesley, 2004. 208 с.

20. Firebase Authentication for Flutter URL: https://firebase.google.com/docs/auth/flutter/start (дата звернення: 09.06.2025)

21. Запис даних у Firestore із server Google Tag Manager https://stape.io/ua/blog/zapis-dannih-v-firestore-s-server-google-tag-manager-ua (дата звернення: 09.06.2025)

22. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проєктів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп'ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с.

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання на бакалаврську кваліфікаційну роботу

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)

«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

«Розробка застосунку для підготовки до мультипредметного тестування з адаптивною перевіркою знань»

08-31.БКР.014.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Володимир СЕВАСТЬЯНОВ

_____
(підпис)

«_23_» _травня__ 2025 р.

Розробив студент гр. 1ІСТ-21б
_____Максим МІЩУК_____

_____
(підпис)

«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розробка застосунку для підготовки до мультипредметного тестування з адаптивною перевіркою знань» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є вдосконалення процесу підготовки до мультипредметного тестування шляхом створення інтелектуального мобільного засобу з адаптивною перевіркою знань, персоналізованими рекомендаціями та інтерактивною візуалізацією навчального прогресу для підвищення ефективності самостійного навчання.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1ІСТ-21б Міщук Максим Дмитрович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз предметної області	07.03 – 19.03
Е2	Проєктування застосунку для тестування знань	20.04 – 26.04
Е3	Програмна реалізація мобільного застосунку	10.05 – 01.06
Е4	Висновки	02.06 – 10.06

4 Призначення і галузь застосування

Мобільний застосунок для підготовки до мультипредметного тестування призначений для проведення інтерактивного тестування учнів з основних навчальних предметів, аналізу отриманих результатів, виявлення прогалин у знаннях та надання персоналізованих рекомендацій щодо подальшої підготовки. Застосунок дозволяє адаптувати складність запитань відповідно до рівня знань користувача, зберігає історію проходження тестів та візуалізує статистику у вигляді графіків.

Розроблена система застосовується у сфері освіти для самостійної підготовки учнів до національного мультипредметного тесту (НМТ), а також може бути використана у навчальних закладах, репетиторських курсах, підготовчих центрах та для індивідуальної роботи учнів. Вона особливо корисна в умовах дистанційного або змішаного навчання як допоміжний інструмент для підвищення ефективності освітнього процесу.

5 Технічні дані

5.1 Платформа розробки – Flutter 3.10;
5.2 Операційна система – Android 8.0 або новіша;
5.3 Підтримка iOS;
5.4 Тип бази даних – Firestore;
5.5 Формат збереження статистики – JSON (Firestore);
5.6 Підтримка автентифікації – Firebase Authentication (email/пароль);

6 Джерела розробки

6.1 Dart Programming Language Guide URL: https://www.geeksforgeeks.org/dart-tutorial/
6.2 Setting Up Flutter in Visual Studio Code: A Step-by-Step Guide URL: https://medium.com/@hanansani2002/setting-up-flutter-in-visual-studio-code-a-step-by-step-guide-1c64450728a7
6.3 Firestore Data Model: An Easy Guide URL: https://hevodata.com/learn/firestore-data-model/
6.4 Запис даних у Firestore із server Google Tag Manager https://stape.io/ua/blog/zapis-dannih-v-firestore-s-server-google-tag-manager-ua

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПІДГОТОВКИ ДО МУЛЬТИПРЕДМЕТНОГО ТЕСТУВАННЯ З АДАПТИВНОЮ ПЕРЕВІРКОЮ ЗНАНЬ

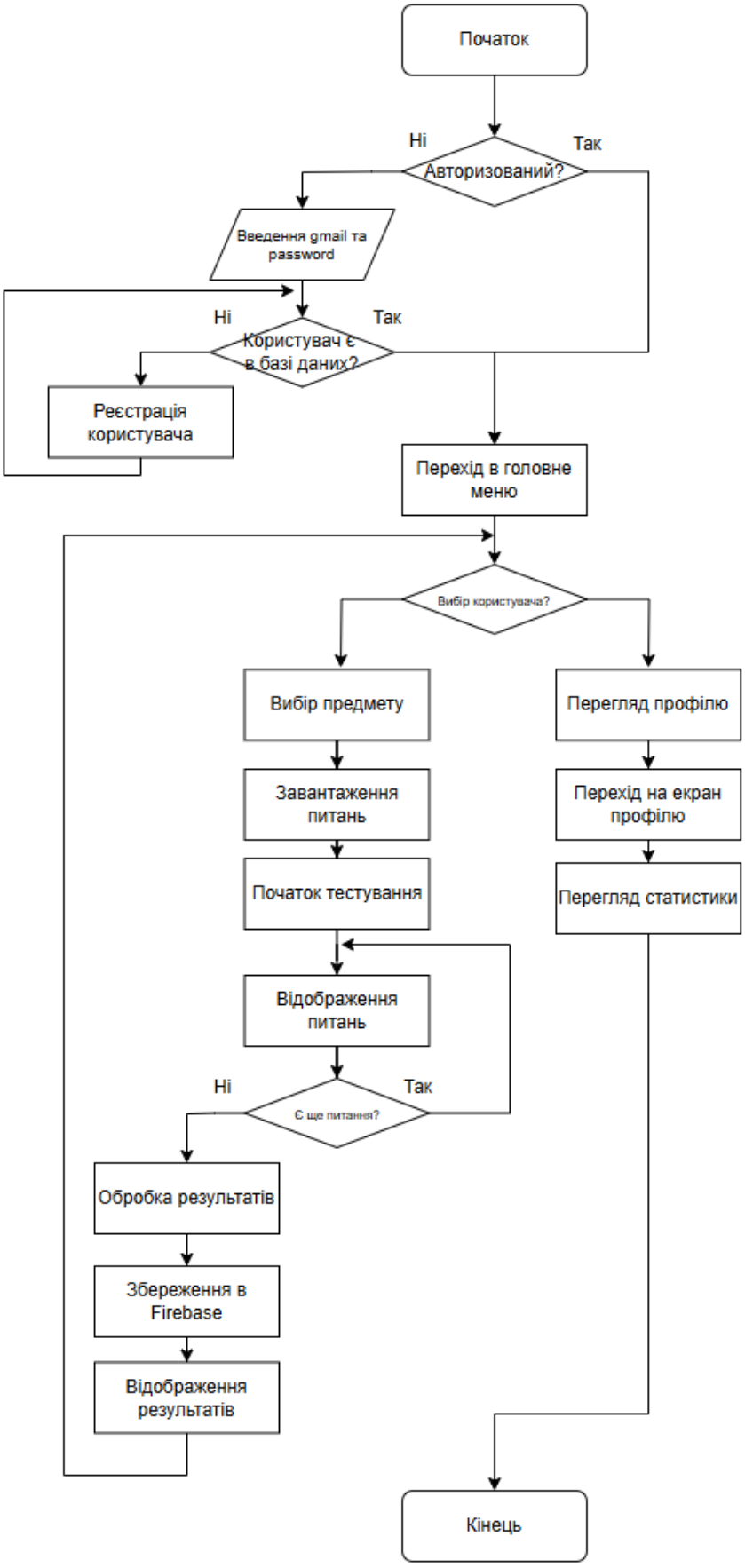


Рисунок А.1 – Блок-схема алгоритм роботи програми

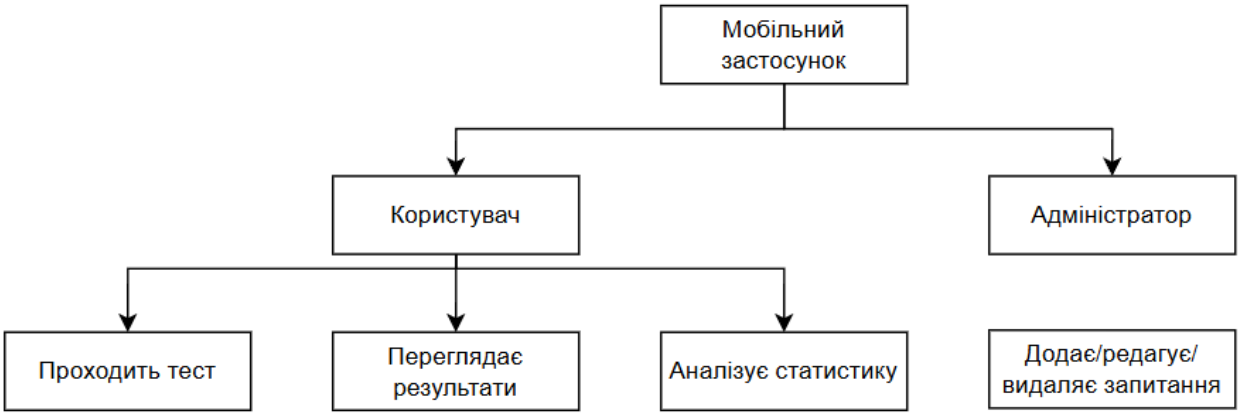


Рисунок А.2 – Схема взаємодії ролей користувачів у мобільному застосунку

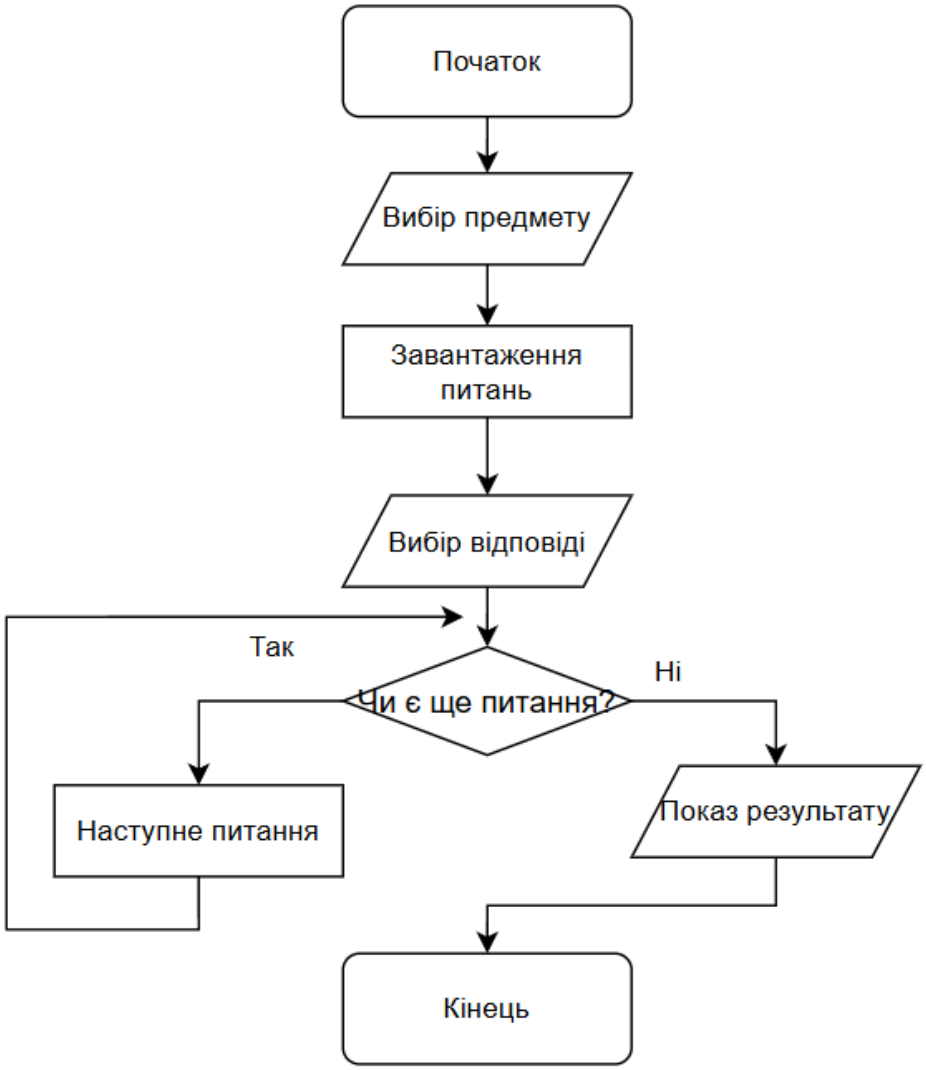


Рисунок А.3 – UML-діаграма активностей

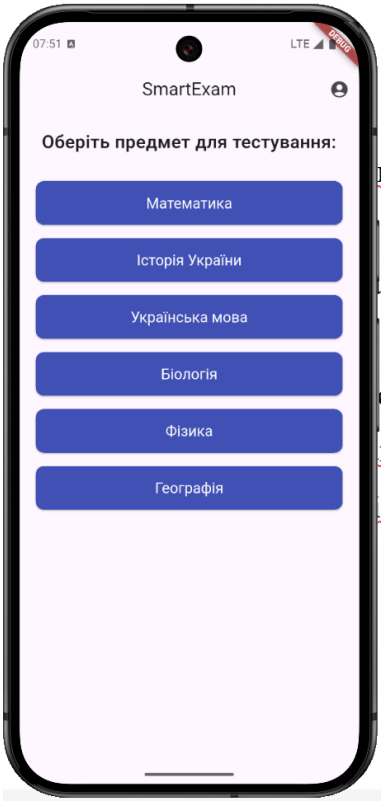


Рисунок А.4 – Загальний вигляд головного екрана застосунку

Рисунок А.5 – Екран проходження тесту

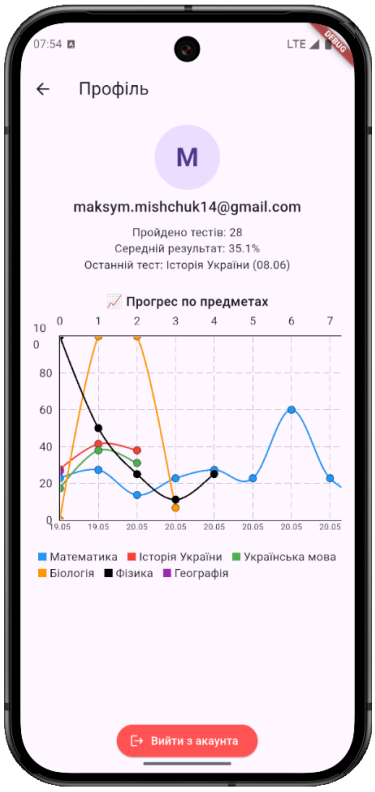


Рисунок А.6 – Вікно профілю користувача з графіком успішності

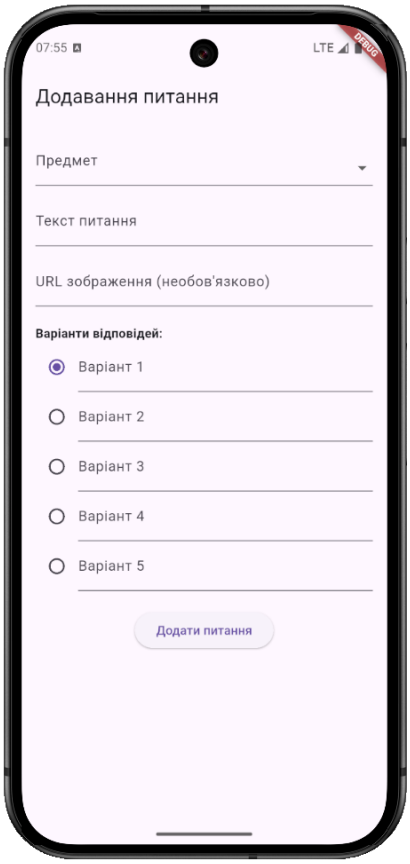


Рисунок А.7 – Інтерфейс адміністратора для додавання запитань

Додаток В

(обов'язковий)

Лістинг програмного забезпечення

main.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:smart_exam/screens/login_screen.dart';
import 'package:smart_exam/screens/home_screen.dart';
import 'package:smart_exam/screens/profile_screen.dart';
import 'firebase_options.dart';
import 'providers/quiz_provider.dart';
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => QuizProvider()),
      ],
      child: MyApp(),
    ),
  );
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Exam',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => LoginScreen(),
        '/home': (context) => HomeScreen(),
        '/profile': (context) => ProfileScreen(),
      },
    );
  }
}
```

home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'quiz_screen.dart';
import 'login_screen.dart';
```

```
import 'profile_screen.dart';
class HomeScreen extends StatelessWidget {
  final List<String> subjects = [
    'Математика',
    'Історія України',
    'Українська мова',
    'Біологія',
    'Фізика',
    'Географія'
  ];
  void _logout(BuildContext context) async {
    await FirebaseAuth.instance.signOut();
    Navigator.of(context).pushReplacement(
      MaterialPageRoute(builder: (_) => LoginScreen()),
    );
  }
  void _openProfile(BuildContext context) {
    Navigator.of(context).push(
      MaterialPageRoute(builder: (_) => ProfileScreen()),
    );
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SmartExam'),
        centerTitle: true,
        actions: [
          IconButton(
            icon: const Icon(Icons.account_circle),
            onPressed: () => _openProfile(context),
          ),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            const Text(
              'Оберіть предмет для тестування:',
              style: TextStyle(fontSize: 22, fontWeight: FontWeight.w600),
              textAlign: TextAlign.center,
            ),
            const SizedBox(height: 24),
            ...subjects.map(
              (subject) => Padding(
                padding: const EdgeInsets.symmetric(vertical: 8.0),
                child: ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.indigo,
                    foregroundColor: Colors.white,
```

```
                padding: const EdgeInsets.symmetric(vertical: 16),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                textStyle: const TextStyle(fontSize: 18),
              ),
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) => QuizScreen(subject: subject),
                  ),
                );
              },
              child: Text(subject),
            ),
          ),
        )
      ],
    ),
  ),
 );
 }
}
```

profile_screen.dart

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:intl/intl.dart';
import 'package:fl_chart/fl_chart.dart';
import 'login_screen.dart';
import 'dart:math';
class ProfileScreen extends StatefulWidget {
 @override
 _ProfileScreenState createState() => _ProfileScreenState();
}
class _ProfileScreenState extends State<ProfileScreen> {
 final currentUser = FirebaseAuth.instance.currentUser;
 List<Map<String, dynamic>> stats = [];
 @override
 void initState() {
  super.initState();
  fetchStats();
 }
 Future<void> fetchStats() async {
  final snapshot = await FirebaseFirestore.instance
      .collection('users')
      .doc(currentUser!.uid)
      .collection('stats')
      .orderBy('date')
      .get();
```

```
    setState(() {
      stats = snapshot.docs.map((doc) => doc.data()).toList();
    });
  }
  void _openChartFullScreen(Map<String, List<FlSpot>> spotsMap, List<DateTime> sortedDates,
Map<String, Color> subjectColors) {
    final dateFormat = DateFormat('dd.MM');
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (_) => Scaffold(
          appBar: AppBar(title: Text('Детальний графік')),
          body: Center(
            child: SingleChildScrollView(
              scrollDirection: Axis.horizontal,
              child: SizedBox(
                width: max(spotsMap.values.fold<double>(300, (prev, list) => list.length * 50.0 > prev ?
list.length * 50.0 : prev), MediaQuery.of(context).size.width),
                height: MediaQuery.of(context).size.height * 0.6,
                child: LineChart(
                  LineChartData(
                    minY: 0,
                    maxY: 100,
                    lineTouchData: LineTouchData(enabled: true),
                    clipData: FlClipData.all(),
                    titlesData: FlTitlesData(
                      leftTitles: AxisTitles(
                        sideTitles: SideTitles(showTitles: true, reservedSize: 32),
                      ),
                      bottomTitles: AxisTitles(
                        sideTitles: SideTitles(
                          showTitles: true,
                          getTitlesWidget: (value, _) {
                            int index = value.toInt();
                            if (index < sortedDates.length) {
                                  return Text(dateFormat.format(sortedDates[index]), style: TextStyle(fontSize:
10));
                            }
                            return Text('');
                          },
                        ),
                      ),
                    ),
                    lineBarsData: spotsMap.entries.map((entry) {
                      final color = subjectColors[entry.key] ?? Colors.purple;
                      return LineChartBarData(
                        spots: entry.value,
                        isCurved: true,
                        barWidth: 2,
                        color: color,
                        dotData: FlDotData(show: true),
                      );
                    }).toList(),
```

```
              ),
            ),
          ),
        ),
      ),
    ),
  ),
 );
}
@override
Widget build(BuildContext context) {
 final email = currentUser?.email ?? '';
 final subjectGroups = <String, List<Map<String, dynamic>>>{};
 for (final stat in stats) {
  final subject = stat['subject'] ?? 'Інше';
  subjectGroups.putIfAbsent(subject, () => []).add(stat);
 }
 final subjectColors = {
  'Математика': Colors.blue,
  'Історія України': Colors.red,
  'Українська мова': Colors.green,
  'Біологія': Colors.orange,
  'Фізика': Colors.black,
  'Географія': Colors.purple,
 };

 final dateFormat = DateFormat('dd.MM');
 final spotsMap = <String, List<FlSpot>>{};
 final dates = <DateTime>[];
 for (final entry in subjectGroups.entries) {
  final subject = entry.key;
  final list = entry.value;
  final subjectSpots = <FlSpot>[];
  for (int i = 0; i < list.length; i++) {
   final stat = list[i];
   final date = (stat['date'] as Timestamp).toDate();
   final percent = (stat['score'] / stat['total']) * 100;
   dates.add(date);
   subjectSpots.add(FlSpot(i.toDouble(), percent));
  }
  spotsMap[subject] = subjectSpots;
 }
 final sortedDates = dates.toSet().toList()..sort();
 final averagePercent = stats.isNotEmpty
    ? stats.map((s) => s['score'] / s['total']).reduce((a, b) => a + b) / stats.length * 100
    : 0;
 final last = stats.isNotEmpty ? stats.last : null;
 return Scaffold(
  appBar: AppBar(title: Text('Профіль')),
  body: Padding(
   padding: const EdgeInsets.all(16.0),
   child: Column(
```

```
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          CircleAvatar(
            radius: 40,
            child: Text(
              email.isNotEmpty ? email[0].toUpperCase() : '?',
              style: TextStyle(fontSize: 32, fontWeight: FontWeight.bold),
            ),
          ),
          SizedBox(height: 8),
          Text(email, style: TextStyle(fontSize: 18, fontWeight: FontWeight.w500)),
          SizedBox(height: 8),
          Text('Пройдено тестів: ${stats.length}'),
          Text('Середній результат: ${averagePercent.toStringAsFixed(1)}%'),
          if (last != null)
                    Text('Останній тест: ${last['subject']} (${dateFormat.format((last['date'] as
Timestamp).toDate())})'),
          SizedBox(height: 24),
                  Text('📈 Прогрес по предметах', style: TextStyle(fontSize: 16, fontWeight:
FontWeight.w600)),
          GestureDetector(
            onTap: () => _openChartFullScreen(spotsMap, sortedDates, subjectColors),
            child: Container(
              color: Colors.transparent,
              height: 280,
              child: SingleChildScrollView(
                scrollDirection: Axis.horizontal,
                child: Builder(
                  builder: (context) {
                    final chartWidth = max(
                      spotsMap.values.map((list) => list.length * 50.0).fold<double>(300, (a, b) => a > b ?
a : b),
                      MediaQuery.of(context).size.width,
                    );
                    return SizedBox(
                      width: chartWidth,
                      child: LineChart(
                        LineChartData(
                          minY: 0,
                          maxY: 100,
                          lineTouchData: LineTouchData(enabled: true),
                          clipData: FlClipData.all(),
                          titlesData: FlTitlesData(
                            leftTitles: AxisTitles(
                              sideTitles: SideTitles(showTitles: true, reservedSize: 32),
                            ),
                            bottomTitles: AxisTitles(
                              sideTitles: SideTitles(
                                showTitles: true,
                                getTitlesWidget: (value, _) {
                                  int index = value.toInt();
```

```
                        if (index < sortedDates.length) {
                        return Text(dateFormat.format(sortedDates[index]), style: TextStyle(fontSize:
10));
                        }
                        return Text('');
                      },
                    ),
                  ),
                ),
                lineBarsData: spotsMap.entries.map((entry) {
                  final color = subjectColors[entry.key] ?? Colors.purple;
                  return LineChartBarData(
                    spots: entry.value,
                    isCurved: true,
                    barWidth: 2,
                    color: color,
                    dotData: FlDotData(show: true),
                  );
                }).toList(),
              ),
            ),
          );
        },
      ),
    ),
  ),
),
const SizedBox(height: 12),
Wrap(
  spacing: 12,
  children: subjectGroups.keys.map((subject) {
    final color = subjectColors[subject] ?? Colors.purple;
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        Container(width: 10, height: 10, color: color),
        SizedBox(width: 4),
        Text(subject),
      ],
    );
  }).toList(),
),
const Spacer(),
ElevatedButton.icon(
  onPressed: () async {
    await FirebaseAuth.instance.signOut();
    Navigator.of(context).pushAndRemoveUntil(
      MaterialPageRoute(builder: (context) => LoginScreen()),
      (route) => false,
    );
  },
  icon: Icon(Icons.logout),
```

```
            label: Text('Вийти з акаунта'),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.redAccent,
              foregroundColor: Colors.white,
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

quiz_screen.dart

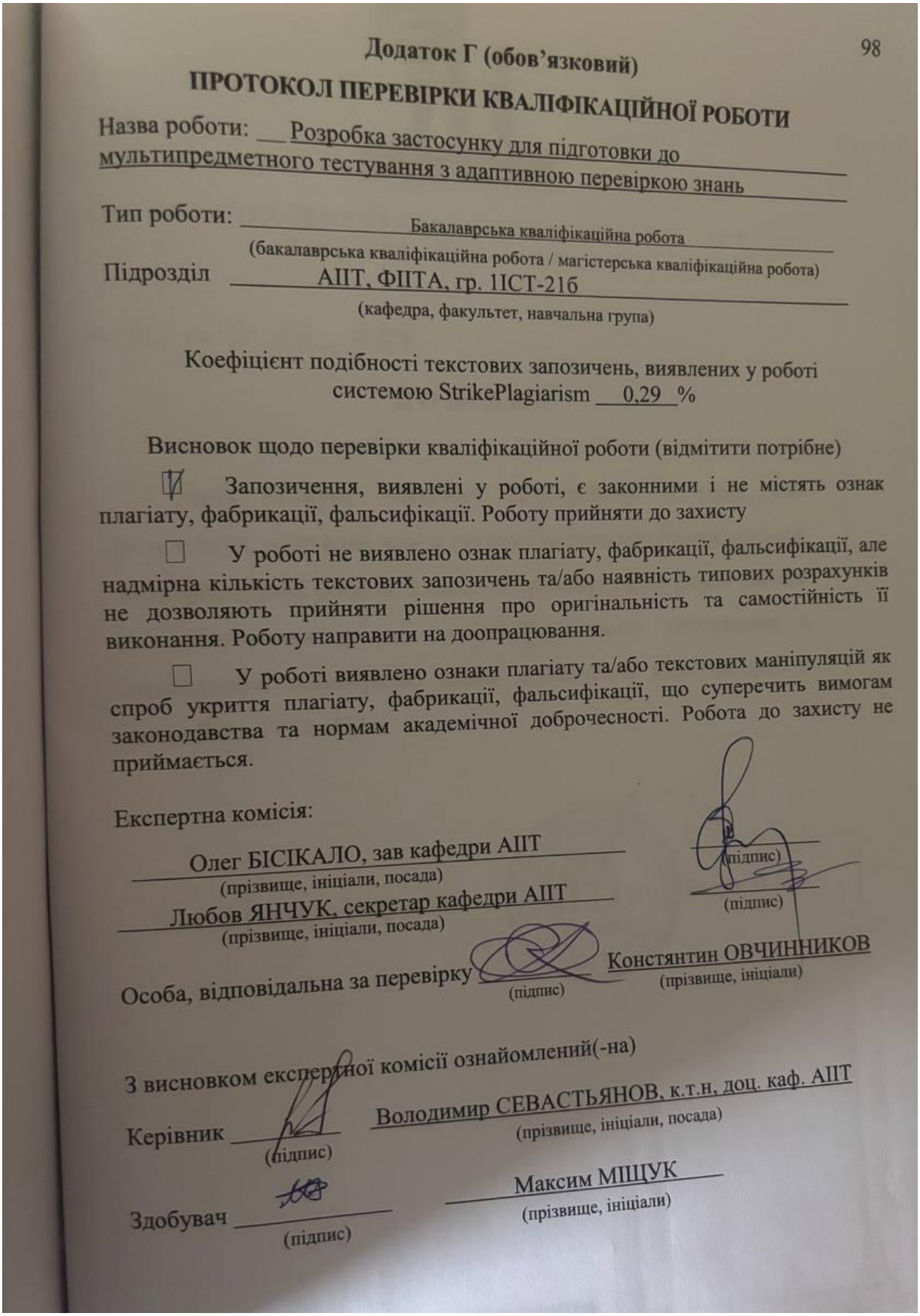
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../models/question.dart';
import '../providers/quiz_provider.dart';
import 'result_screen.dart';
class QuizScreen extends StatefulWidget {
 final String subject;
 const QuizScreen({Key? key, required this.subject}) : super(key: key);
 @override
 _QuizScreenState createState() => _QuizScreenState();
}
class _QuizScreenState extends State<QuizScreen> {
 bool isLoading = true;
 @override
 void initState() {
  super.initState();
  loadQuestions();
 }
 Future<void> loadQuestions() async {
  await Provider.of<QuizProvider>(context, listen: false)
     .fetchQuestions(widget.subject);
  setState(() {
   isLoading = false;
  });
 }
 void answerQuestion(int selectedIndex) {
  final quizProvider = Provider.of<QuizProvider>(context, listen: false);
  quizProvider.answerCurrentQuestion(selectedIndex);
  if (quizProvider.isLastQuestion) {
   Navigator.of(context).pushReplacement(
    MaterialPageRoute(
     builder: (_) => ResultScreen(subject: widget.subject),
    ),
   );
  } else {
   quizProvider.nextQuestion();
  }
 }
```

```
TextSpan _buildTextSpan(String text) {
  final spans = <TextSpan>[];
  final regex = RegExp(r'\*\*(.+?)\*\*');
  int start = 0;
  for (final match in regex.allMatches(text)) {
    if (match.start > start) {
      spans.add(TextSpan(text: text.substring(start, match.start)));
    }
    spans.add(TextSpan(
      text: match.group(1),
      style: const TextStyle(fontWeight: FontWeight.bold),
    ));
    start = match.end;
  }
  if (start < text.length) {
    spans.add(TextSpan(text: text.substring(start)));
  }
  return TextSpan(
    style: const TextStyle(fontSize: 18, color: Colors.black),
    children: spans,
  );
}
@override
Widget build(BuildContext context) {
  final quizProvider = Provider.of<QuizProvider>(context);
  if (isLoading) {
    return Scaffold(
      appBar: AppBar(title: const Text('Завантаження')),
      body: const Center(child: CircularProgressIndicator()),
    );
  }
  if (quizProvider.questions.isEmpty) {
    return Scaffold(
      appBar: AppBar(title: const Text('Питання')),
      body: Center(
        child: Text('Немає питань для предмету "${widget.subject}"'),
      ),
    );
  }
  final question = quizProvider.currentQuestion;
  return Scaffold(
    appBar: AppBar(
      title: Text('Питання ${quizProvider.currentIndex + 1}'),
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          if (question.imageUrl != null && question.imageUrl!.isNotEmpty)
            ClipRRect(
              borderRadius: BorderRadius.circular(12),
```

```
            child: Image.network(
              question.imageUrl!,
              fit: BoxFit.contain,
            ),
          ),
        const SizedBox(height: 16),
        RichText(
          text: _buildTextSpan(question.text),
        ),
        const SizedBox(height: 24),
        ...List.generate(question.answers.length, (index) {
          final answer = question.answers[index];
          final isImageAnswer = answer.startsWith('http') && answer.contains('.png');
          return Padding(
            padding: const EdgeInsets.only(bottom: 12.0),
            child: ElevatedButton(
              onPressed: () => answerQuestion(index),
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.indigoAccent,
                foregroundColor: Colors.white,
                minimumSize: const Size(double.infinity, 48),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                textStyle: const TextStyle(fontSize: 16),
              ),
              child: isImageAnswer
                  ? Image.network(answer, height: 120)
                  : RichText(text: _buildTextSpan(answer)),
            ),
          );
        }),
      ],
    ),
  ),
 );
 }
}
```

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ



Додаток Г (обов'язковий) 98

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка застосунку для підготовки до мультипредметного тестування з адаптивною перевіркою знань

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ АІІТ, ФІІТА, гр. 1ІСТ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,29 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав кафедри АІІТ (підпис)
(прізвище, ініціали, посада)

Любов ЯНЧУК, секретар кафедри АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Констянтин ОВЧИННИКОВ
(прізвище, ініціали)

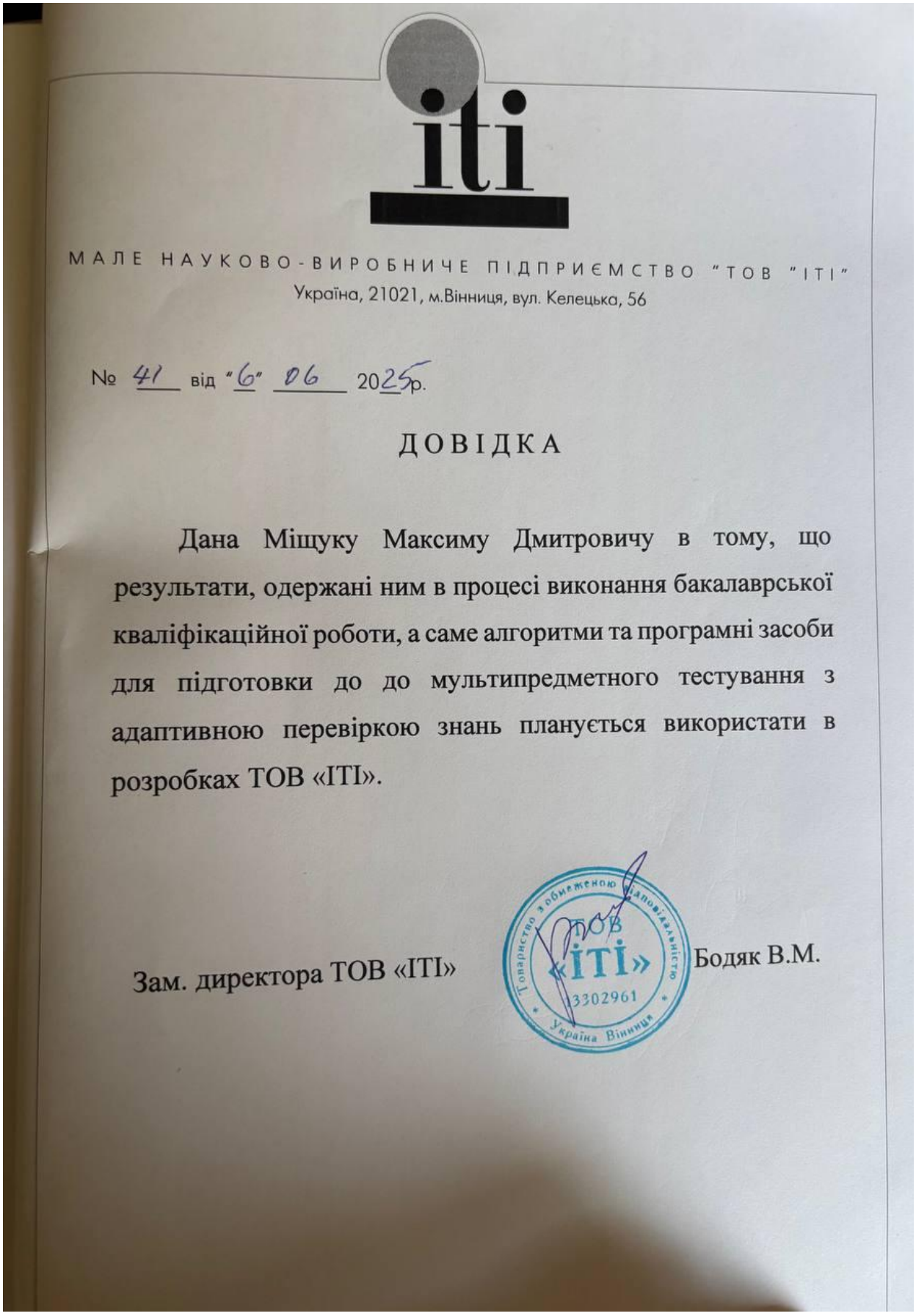
З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Володимир СЕВАСТЬЯНОВ, к.т.н, доц. каф. АІІТ
(прізвище, ініціали, посада)

Здобувач (підпис) Максим МІЩУК
(прізвище, ініціали)

Додаток Д (довідниковий)

Акт впровадження результатів роботи в ТОВ «СПІЛЬНА СПРАВА»



МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ 41 від "6" 06 2025р.

ДОВІДКА

Дана Міщуку Максиму Дмитровичу в тому, що результати, одержані ним в процесі виконання бакалаврської кваліфікаційної роботи, а саме алгоритми та програмні засоби для підготовки до до мультипредметного тестування з адаптивною перевіркою знань планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» Бодяк В.М.

Товариство з обмеженою відповідальністю ТОВ «ІТІ» 3302961 Україна Вінниця