Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повна назва інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної (циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(рівень вищої освіти)
на тему: «Розроблення кросплатформного мобільного застосунку для онлайн-курсів засобами Flutter»

Виконав студент 2 курсу, групи КНС-21
спеціальності:
122 „Комп'ютерні науки"
(шифр і назва напрямку підготовки спеціальності)

Гориславський Р.В.
(прізвище, ініціали)

Керівник: Сало М.Ф.
(прізвище, ініціали)

Керівник: Яцишин С.І.
(прізвище, ініціали)

Рецензент: Флуд Л.О.
(прізвище, ініціали)

Львів-2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти перший(бакалаврський)
Спеціальність 122 "Комп'ютерні науки"

ЗАТВЕРДЖУЮ:
Завідувач кафедри ІТ
_____ Борецька І.Б.
„10" червня 2025 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Гориславський Руслан Юрійович

(прізвище, ім'я, по батькові)

1. Тема бакалаврської роботи: Розробка кросплатформного застосунку для онлайн-курсів засобами Flutter
керівники роботи ст.викл. Сало М. Ф., к.т.н., доц. Яцишин С.І.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом вищого навчального закладу від "15" листопада 2024 року, №С-882
2. Термін подання студентом проекту(роботи) 10 червня 2025р
3. Вихідні дані до проекту (роботи) Розробити мобільний застосунок для роботи з онлайн-курсами. Застосунок з можливістю працювати над завданням певного курсу з допомогою мобільного пристрою та зберігати результат. Користувач здатен працювати над завданнями, незалежно від місця перебування.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
РОЗДІЛ 1. Стан проблемної області
РОЗДІЛ 2. Інформаційне та математичне забезпечення
РОЗДІЛ 3. Програмне та технічне забезпечення
ВИСНОВКИ
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка матеріалу до доповіді .
6. Дата видачі завдання 18 листопада 2024р.

КАЛЕНДАРНИЙ ПЛАН

№. з/п	Етапи бакалаврської роботи	Терміни виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	18.11.24 – 08.12.24	Виконано
2.	Постановка задачі і її формалізація	09.12.24 – 29.01.25	Виконано
3.	Виконання вхідного етапу технології	30.01.25 – 09.02.25	Виконано
4.	Реалізація головних класів проекту	10.02.25 – 18.03.25	Виконано
5.	Виконання етапу відлагодження проекту	19.03.25 – 16.04.25	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	17.04.25 – 15.05.25	Виконано
7.	Оформлення записки до дипломного проекту.	16.05.25 – 08.06.25	Виконано

Студент _____ Гориславський Р.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сало М.Ф.
(підпис) (прізвище та ініціали)

Керівник _____ Яцишин С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота містить 80 сторінок, 28 ілюстрацій, 7 додатків, 17 джерел.

Дипломна робота присвячена розробці мобільного застосунку для користувачів онлайн-курсів з програмування віком від 10 до 18 років. Основною функціональністю є зручний інтерфейс для навчання та створення власного HTML-коду за допомогою інтеграції API, що забезпечує інтерактивну роботу з кодом. Застосунок створено за допомогою мови Dart, що гарантує високий рівень продуктивності та комфорт у використанні. Використання сучасних бібліотек та технологій значно скоротило час розробки, зменшило обсяг зайвого коду та дозволило зосередитися на забезпеченні інтуїтивно зрозумілого інтерфейсу та корисних функцій для молодих користувачів.

Ключові слова: ***Dart, Flutter, API, HTML, SQLite, Render Dashboard***

ABSTRACT

Bachelor thesis (project): explanatory note: 80 pages, 28 illustrations, 7 appendices, 17 sources.

The thesis is devoted to the development of a mobile application for users of online programming courses aged 10 to 18. The main feature is a user-friendly interface for learning and creating custom HTML code through integration with an API, providing an interactive coding experience. The application is built using Dart, ensuring high performance and ease of use. The application of modern libraries and technologies significantly reduced development time, minimized redundant code, and allowed focusing on delivering an intuitive interface and valuable features for young learners.

Keywords : Dart, Flutter, API, HTML, SQLite, Render Dashboard

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити мобільний застосунок для організації навчального процесу між викладачами та учнями віком від 8 до 16 років. Для зберігання даних використати локальну базу даних SQLite, що забезпечить швидкодію і зручність у роботі з даними на мобільних пристроях.

На фронтенді застосунку буде реалізовано інтуїтивний та сучасний інтерфейс за допомогою Flutter та Dart, з використанням бібліотеки Flutter Material Design для створення привабливих та зручних UI-компонентів.

Для управління ролями користувачів (учень, викладач, адміністратор) та їх правами застосовано відповідну логіку аутентифікації та авторизації. Реалізовано можливість входу через OAuth 2.0 для забезпечення безпечної та зручної автентифікації користувачів.

Основні функції застосунку включають створення та участь у курсах, додавання та виконання завдань, оцінювання відповідей, ведення розкладу занять та календаря подій, що відображається у зручному календарі (TableCalendar).

Для підвищення ефективності та автоматизації пошуку навчальних матеріалів і завдань передбачено інтеграцію API, що дозволяє створювати власний HTML-код або виконувати інші дії у межах застосунку.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

SQLite — легка вбудована реляційна база даних, що використовується для зберігання даних у мобільних додатках.

Flutter — відкритий фреймворк від Google для створення кросплатформних мобільних застосунків з єдиним кодом.

Dart — мова програмування, яка використовується для розробки застосунків на Flutter.

UI/UX — принципи розробки зручного і інтуїтивно зрозумілого інтерфейсу користувача та досвіду взаємодії з ним.

OAuth 2.0 — протокол авторизації, що забезпечує безпечний вхід користувачів до застосунку через сторонні сервіси.

Supabase — платформа з відкритим кодом для управління даними та автентифікації, що використовується для зберігання та обробки даних застосунку.

Render.com — хмарна платформа для розгортання та хостингу серверних додатків і баз даних.

Visual Studio Code — популярний легкий редактор коду, що використовується для розробки та тестування застосунків.

API — прикладний програмний інтерфейс, що дозволяє взаємодіяти з зовнішніми сервісами або внутрішніми компонентами застосунку.

ВСТУП

Розробка мобільного додатка для організації навчального процесу між викладачами та учнями є актуальним і затребуваним рішенням у сфері освітніх технологій. Додаток забезпечує інтерактивну взаємодію між користувачами, спрощує управління курсами, завданнями та розкладом занять, що сприяє підвищенню ефективності навчального процесу.

Об'єктом дослідження є використання мобільних технологій для організації навчального процесу.

Метою дипломної роботи є розроблення мобільного додатка, що дозволяє викладачам створювати курси, призначати завдання, керувати розкладом занять, а учням—зручно взаємодіяти з навчальним контентом.

Практичне значення роботи полягає у створенні інтуїтивно зрозумілого та функціонального мобільного додатка, який допоможе оптимізувати освітній процес, спростити комунікацію між викладачами та учнями, а також забезпечити доступність навчального контенту в цифровому форматі.

Предметом дослідження є реалізація мобільного додатка для навчальних курсів із використанням Flutter та SQLite.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області

У сучасному світі цифрових технологій освітній процес потребує все більшої адаптації до потреб користувачів, зокрема учнів та викладачів. Традиційні методи навчання часто супроводжуються викликами, такими як складність організації навчальних курсів, управління завданнями та комунікація між учнями та викладачами.

Попри зростаючий інтерес до онлайн-освіти та цифрових навчальних платформ, навчальні уста’нови стикаються з проблемами, такими як обмеженість існуючих сервісів у підтримці інтерактивного навчального процесу, неструктурованість розкладу занять, а також складність ведення електронного журналу оцінок.

У цьому контексті розробка мобільного додатка для управління навчальним процесом на базі Flutter є актуальним завданням. Така система значно полегшує взаємодію між викладачами та учнями, забезпечуючи можливість організованого навчального процесу.

Запропонований додаток дозволяє створювати курси, призначати завдання, оцінювати виконання та керувати розкладом занять. Використання технологій Flutter, Dart та SQLite забезпечує гнучкість розробки, високу продуктивність та інтеграцію локальної бази даних для ефективного збереження інформації.

Інтелектуальна система управління курсами включає такі функціональні можливості, як адміністрування користувачів (учні, викладачі, адміністратори), формування персоналізованих навчальних курсів, підтримку календаря подій із розкладом занять та дедлайнами. Крім того, додаток забезпечує захист даних та інтеграцію з хмарними сервісами для можливого майбутнього розширення його функціональності.

1.2. Сучасні тенденції та виклики

В епоху цифрових технологій освіта переживає активне трансформування, поступово переходячи до інтерактивних платформ та мобільних додатків. Зростає потреба в ефективних навчальних інструментах, що дозволяють організувати навчальний процес та спростити взаємодію між учнями та викладачами.

Однією з основних проблем є недостатня інтеграція технологій у навчальний процес, що призводить до складнощів у структурованому поданні навчального матеріалу, плануванні занять та управлінні завданнями. Багато учнів та викладачів стикаються з труднощами під час організації навчання, що може знижувати його ефективність та мотивацію учасників.

Традиційні методи організації навчального процесу, такі як друковані матеріали або розклади на папері, часто є незручними та не забезпечують гнучкість у доступі до навчальної інформації. Використання мобільного додатка для навчальних курсів може суттєво покращити взаємодію між користувачами, автоматизувати процеси адміністрування курсів та дозволити більш ефективне управління навчанням.

Розробка мобільного додатка на базі Flutter забезпечує широкий спектр можливостей для викладачів та учнів. Система підтримує створення курсів, призначення завдань, ведення розкладу занять, а також інтеграцію календаря подій. Використання локальної бази даних SQLite дозволяє зберігати всі дані без необхідності постійного з'єднання з сервером, що підвищує автономність додатка.

1.3. Роль мобільних навчальних систем

У сучасному освітньому процесі мобільні навчальні системи відіграють ключову роль, забезпечуючи інтерактивність та доступність навчальних матеріалів. Використання таких технологій дозволяє значно спростити управління курсами, організувати завдання та взаємодію між викладачами й учнями.

Мобільні додатки для освіти не лише покращують комунікацію між

учасниками навчального процесу, а й надають можливість персоналізованого підходу до навчання. Завдяки інтегрованим механізмам, таким як системи розкладу занять, функції адміністрування та інструменти зворотного зв'язку, навчальний процес стає більш структурованим та ефективним.

Впровадження сучасних мобільних навчальних систем дозволяє зменшити адміністративне навантаження, автоматизувати процеси управління курсами та покращити загальну організацію освітнього середовища.

1.4. Переваги використання Flutter

Flutter, як сучасний фреймворк для розробки мобільних додатків від Google, є одним із найпопулярніших інструментів для створення кросплатформних мобільних рішень. Використання Flutter надає розробникам широкий набір переваг, що робить його ідеальним вибором для створення інтуїтивних, продуктивних та функціональних мобільних додатків.

Однією з ключових переваг Flutter є його здатність створювати мобільні додатки, що працюють на різних операційних системах, таких як Android та iOS, використовуючи один базовий код. Це дозволяє значно зменшити витрати на розробку та пришвидшити процес розгортання додатка. Завдяки використанню власного механізму рендерингу, Flutter забезпечує високу швидкість виконання програм, що особливо важливо для навчальних платформ із великим обсягом даних.

Flutter використовує власний набір віджетів для побудови інтерфейсу, що дозволяє створювати сучасний та адаптивний дизайн. Завдяки цьому розробники можуть легко змінювати стилі, анімацію та загальний вигляд додатка без необхідності використання додаткових бібліотек. Це забезпечує високу гнучкість у розробці навчальних платформ, де необхідно враховувати потреби учнів та викладачів.

Одним із важливих аспектів мобільного додатка є збереження даних користувачів, розкладу занять та завдань. Використання SQLite дозволяє

забезпечити ефективне управління інформацією без необхідності постійного підключення до серверної частини. Це покращує автономність додатка та дозволяє учням та викладачам працювати з курсами навіть без доступу до інтернету.

Мобільний додаток для навчання повинен забезпечувати високий рівень безпеки даних користувачів. Flutter підтримує вбудовані механізми шифрування та автентифікації, що допомагають захистити інформацію від несанкціонованого доступу. Крім того, стабільність платформи дозволяє створювати надійні рішення, що працюють без збоїв, навіть при великому навантаженні.

Flutter легко інтегрується з різноманітними сервісами, такими як API для обробки запитів, хмарні бази даних та сервіси для аналітики. Опційно додаток може використовувати Render.com для хостингу серверної частини, що дозволяє додавати нові функції в майбутньому та забезпечувати масштабованість платформи.

Спільнота розробників Flutter постійно розширюється, що забезпечує доступ до численних ресурсів, документації та навчальних матеріалів. Це значно полегшує розробку додатків та допомагає швидко знаходити рішення для технічних питань.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Flutter

Flutter — це фреймворк з відкритим вихідним кодом, розроблений компанією Google для створення нативних мобільних, веб та десктопних застосунків з єдиною базою коду. Основною мовою програмування для Flutter є Dart. Фреймворк дозволяє розробникам створювати високопродуктивні інтерфейси користувача з привабливим дизайном і плавною анімацією.[1]

На відміну від багатьох інших фреймворків, Flutter не покладається на нативні компоненти платформи (наприклад, Android Views чи UIKit для iOS). Натомість, він використовує власний рушій на основі бібліотеки Skia для рендерингу графіки. Це дозволяє точно контролювати вигляд інтерфейсу на всіх платформах і гарантує однакову поведінку елементів інтерфейсу.

В Flutter усе є віджетами: кнопки, поля вводу, контейнери, навіть розмітка і анімації. Цей підхід дозволяє створювати складні інтерфейси за допомогою комбінації простих компонентів. Існує два типи віджетів: **Stateful** (змінні) та **Stateless** (незмінні), що дозволяє ефективно управляти станом застосунку.[2]

За допомогою Flutter розробник може створити застосунок, який працює на Android, iOS, Windows, macOS, Linux та у браузері, використовуючи одну й ту саму кодову базу. Це значно скорочує час та витрати на розробку, а також спрощує підтримку продукту.

Flutter має великий каталог відкритих бібліотек (пакетів), доступних через pub.dev. Ці пакети спрощують роботу з HTTP-запитами, базами даних, авторизацією, навігацією, анімаціями тощо. Вони дозволяють розширити функціональність застосунку без необхідності писати власні реалізації з нуля.

2.2. Dart

Dart – це об'єктно-орієнтована мова програмування з відкритим вихідним кодом, яка була розроблена компанією Google у 2011 році. Основною метою створення Dart є забезпечення високої продуктивності та зручності розробки сучасних клієнтських застосунків, зокрема веб-, мобільних та десктопних[3]. Dart є основною мовою для створення застосунків у фреймворку Flutter.

До основних характеристик входять:

1. **Об’єктно-орієнтована структура.** Усі дані в Dart є об'єктами, навіть примітивні типи, такі як числа чи рядки. Dart підтримує класи, наслідування, абстракцію, інтерфейси, міксини та інші парадигми об'єктно-орієнтованого програмування.
2. **Сильна та динамічна типізація.** Dart дозволяє розробнику явно вказувати тип змінної, але також підтримує ключове слово var, яке визначає тип автоматично. Починаючи з Dart 2, типізація стала строгою, що підвищує безпеку та передбачуваність коду.
3. **Асинхронне програмування.** Dart має вбудовану підтримку асинхронного коду за допомогою ключових слів async, await та Future. Це дозволяє зручно працювати з мережевими запитами, таймерами або іншими операціями, які займають певний час.[4]
4. **JIT та AOT компіляція.** Dart підтримує два типи компіляції:

- JIT (Just-in-Time) — компіляція під час розробки, що забезпечує швидкий запуск і гаряче перезавантаження (Hot Reload) для зручності тестування.
- AOT (Ahead-of-Time) — компіляція у нативний код перед запуском, що забезпечує високу продуктивність застосунків у продакшн-версіях

5. **Збірка сміття (Garbage Collection)**. Dart має автоматичну систему керування пам’яттю, яка очищає непотрібні об’єкти, що більше не використовуються в програмі. Це знижує ризик витоку пам’яті та спрощує роботу розробника.

6. **Підтримка функціонального стилю**. Dart дозволяє використовувати функціональні конструкції, такі як замикання (closures), анонімні функції, стрілкові функції (=>) та вищі функції. Це робить код більш компактним та зручним для роботи з колекціями.
7. **Багата стандартна бібліотека**. Мова Dart включає в себе потужну стандартну бібліотеку для роботи з колекціями, файлами, HTTP-запитами, регулярними виразами, JSON, часом та датами, математикою тощо.
8. **Кросплатформеність**. За допомогою Dart можна створювати програми для Android, iOS, вебу, настільних платформ (Windows, macOS, Linux), а також backend-сервери. Це робить Dart універсальною мовою для створення повноцінних застосунків на всіх основних платформах.
9. **Підтримка пакетів та плагінів**. Dart має власний менеджер пакетів pub.dev, що дозволяє легко підключати сторонні бібліотеки для розширення функціональності. Це спрощує реалізацію мережевих запитів, збереження даних, навігації, інтеграції з базами даних тощо.

2.3. Visual Studio Code

Flutter має потужні інструменти для розробки, включаючи інтеграцію з популярними IDE, такими як Visual Studio Code та Android Studio.

Visual Studio Code (VS Code) — це безкоштовний, відкритий та кросплатформений редактор коду, розроблений корпорацією Microsoft. Вперше випущений у 2015 році, VS Code швидко здобув популярність серед розробників завдяки поєднанню легкості редактора тексту з потужністю інтегрованого середовища розробки (IDE). VS Code підтримує розробку на багатьох мовах програмування, серед яких Dart, JavaScript, Python, C++, Java та інші.

Visual Studio Code доступний на операційних системах Windows, macOS та Linux, що дозволяє розробникам використовувати один інструмент незалежно від платформи.

VS Code спроектований як легкий редактор з швидким стартом, що не

навантажує систему, при цьому має функції, характерні для повноцінних IDE. Це робить його зручним як для невеликих скриптів, так і для великих проєктів.

Однією з головних сильних сторін VS Code є його екосистема розширень. Вони дозволяють додавати підтримку нових мов програмування, інтегрувати системи контролю версій (наприклад, Git), засоби для налагодження, форматування коду, а також працювати з Docker, Kubernetes, базами даних та іншими технологіями.[5] Для розробки на Flutter і Dart існують спеціальні плагіни, що значно спрощують процес. За допомогою IntelliSense користувачі отримують підказки, автоматичне доповнення коду, виявлення помилок синтаксису та інші інтелектуальні функції, що підвищують продуктивність і зменшують кількість помилок.

Вбудована підтримка Git дозволяє відслідковувати зміни, створювати коміти, гілки та злиття безпосередньо у редакторі, що робить його зручним інструментом для командної роботи. Вбудований дебагер дає змогу ставити точки зупину, слідкувати за значеннями змінних і контролювати виконання коду. Для різних мов доступні додаткові налаштування налагоджування через розширення.

Саме в контексті мобільної розробки на Flutter VS Code надає функцію Hot Reload, що дозволяє миттєво застосовувати зміни без перезапуску застосунку і значно прискорює процес створення додатків. Інтерфейс забезпечує легку навігацію між файлами, дозволяє швидко відкривати документацію та перетягувати компоненти для швидкого створення UI.[6]

Інструменти тестування Flutter інтегровані з VS Code, що дозволяє запускати юніт-тести і інтеграційні тести безпосередньо з редактора. У VS Code можна встановлювати додаткові плагіни для роботи з базами даних, сервісами бекенду, тестування тощо.

2.4. Async та Await

Одним з ключових підходів у розробці мобільного додатку стало використання асинхронного програмування, що дозволяє виконувати тривалі операції без блокування головного потоку програми. Це особливо важливо у випадках, коли необхідно звертатися до бази даних, надсилати запити до сервера або завантажувати великі обсяги інформації[7]. У середовищі Flutter/Dart асинхронне програмування реалізується за допомогою ключових слів async та await. Найпоширенішими асинхронними операціями можна назвати отримання даних через мережу, запис інформації у базу даних, зчитування з бази даних. Асинхронні функції дозволяють:

- Виконувати запити до SQLite-бази без заморожування інтерфейсу.
- Отримати відповідь від API.
- Послідовно обробляти кілька джерел даних.
- Динамічно оновлювати UI після завершення завантаження

До переваг використання асинхронного підходу входять: плавна взаємодія з інтерфейсом, де користувач не відчуває значних затримок під час завантаження; оптимізація продуктивності, при якій основний потік залишається вільним; гнучкість, як дозволяє взаємодіяти з паралельними та послідовними асинхронними запитами; безпека через логічну конструкцію try-catch.

2.5. SQLite

SQLite — це легка, вбудована система управління базами даних (СУБД), яка базується на реляційній моделі. Це означає, що дані зберігаються у вигляді таблиць, і можна виконувати SQL-запити для їх обробки[8]. На відміну від традиційних серверних баз даних (MySQL, PostgreSQL, SQL Server), SQLite не потребує окремого сервера або процесу для роботи. Вона є вбудованою бібліотекою, яку можна інтегрувати безпосередньо у додатки.

Завдяки своїй зручності, простоті та легкості, SQLite широко використовується в мобільних додатках, браузерах, вбудованих системах та інших програмних рішеннях.

SQLite займає всього кілька сотень кілобайт, що робить її ідеальною для вбудованих систем. Вихідний код дуже малий (менше ніж кілька сотень кілобайт), тому її легко додавати до мобільних, десктопних систем.

Вся база даних зберігається у одному файлі, що полегшує її копіювання, резервне копіювання і поширення. SQLite підтримує більшу частину стандартного SQL, включаючи транзакції, підзапити, тригери, індекси і т.п.[9]

Додаток використовує SQL-запити для створення таблиць, вставки, оновлення, видалення і читання даних, зокрема звичні ELECT, INSERT, UPDATE, DELETE.

SQLite складається з ядра бази даних, яке обробляє SQL-запити, та файлової системи, що зберігає дані. Вона використовує B-дерева для організації таблиць та індексів, що забезпечує швидкий доступ до інформації[10].

2.6. HTTP(API)

Для забезпечення обміну даними між клієнтською частиною мобільного додатку та сервером використовується протокол HTTP (HyperText Transfer Protocol).

HTTP є стандартом прикладного рівня для передавання повідомлень у мережі Інтернет[11]. У контексті мобільного застосунку на Flutter, він забезпечує надсилання запитів до серверної частини (бекенду) та отримання відповідей.

Обмін даними відбувається через RESTful API (Application Programming Interface), що надає структурований інтерфейс для взаємодії між додатком та базою даних, яка розміщується на сервері. API дозволяє додатку виконувати операції створення, читання, оновлення та видалення (CRUD) даних.

У проєкті використовується бібліотека http, яка дозволяє реалізовувати мережеві запити типу GET, POST, PUT та DELETE.

Взаємодія застосунку з сервером через HTTP працює наступним чином: Застосунок формує HTTP-запит(GET, POST, PUT, DELETE.). Далі, застосунок використовує HTTP-клієнт через модуль або бібліотеку для відправки запиту на сервер. Сервер отримує запит, обробляючи його і повертає HTTP-відповідь. Застосунок аналізує JSON або інший формат відповіді, оновлює UI або внутрішній стан додатку.

2.7. Render.com

Для хостингу серверної частини програмного забезпечення було використано хмарну платформу Render.com. Це сучасна хмарна платформа для розгортання, хостингу та управління веб-додатками та сервісами. Вона позиціює себе як альтернативу традиційним хмарним провайдерам (AWS, GCP, Azure), пропонуючи простий, зручний та доступний інтерфейс для розробників і команд.

Render.com забезпечує автоматичне розгортання застосунків безпосередньо з репозиторіїв GitHub або GitLab. Завдяки цьому розробник може організувати CI/CD-процеси (безперервну інтеграцію та доставку), що значно підвищує ефективність роботи з серверною логікою[12].

У межах розробки мобільного додатку Render.com використовується для хостингу REST API, яке обробляє запити з Flutter-додатку. Серверна частина реалізована з використанням відповідного стеку технологій (наприклад, Node.js, Express, PostgreSQL або інші), і надає зовнішній інтерфейс для роботи з даними: завданнями, курсами, відповідями учнів, розкладом занять тощо.

Перевагами Render.com є підтримка автоматичного SSL (HTTPS) для безпечної передачі даних, просте масштабування у разі збільшення навантаження, зручна система логування для відстеження помилок, підтримка серверів із постійним або тимчасовим живленням (free/dedicated instances).

До недоліків Render.com можна віднести:

- обмеження безплатного тарифу, яке автоматично вимкнеться через 15хв без активності.
- обмеження щодо обсягу пам’яті та CPU.
- ліміт на вихідний трафік і зберігання, що призводить до уповільнення запуску додатка після тимчасового сну.

2.8. WebView Flutter

WebView Flutter — це офіційний Flutter-плагін, що дозволяє вбудовувати вебсторінки безпосередньо в інтерфейс мобільного застосунку. По суті, WebView — це контейнер, у якому рендериться веббраузер на базі WebKit (для iOS) або Chromium (для Android)[13]. Зосередимось на Chromium.

Chromium — це проєкт із відкритим вихідним кодом, на базі якого створено браузери, зокрема Google Chrome, Microsoft Edge та інші. Він надає рушій рендерингу вебсторінок (Blink), механізм обробки JavaScript (V8), а також мережевий стек. При використанні WebView у Android, система використовує вбудований компонент Chromium, що дозволяє обробляти та відображати сторінки через WebView так само, як у звичайному браузері Chrome.

Використання цієї компоненти має декілька переваг:

- Повна підтримка HTML5, CSS3, JavaScript — усі сучасні технології веброзробки працюють «із коробки»;
- Безпека — Chromium регулярно оновлюється Google, тому вразливості закриваються швидко;
- Підтримка медіаконтенту, відео, аудіо тощо;
- Сумісність з реальними сайтами — сторінки відображаються так само, як у Chrome.

Звичайно, є і мінуси використання Chromium у WebView:

- Chromium — важка технологія, тому запуск WebView потребує більше оперативної пам’яті;
- Оновлення не залежить від додатку, а від версії Android WebView у системі. Якщо на пристрої не оновлений системний WebView, деякі функції можуть працювати некоректно;
- Обмежена взаємодія з Flutter — хоча можна використовувати JavaScript-канали, інтеграція з нативними елементами обмежена.

2.9. Android Studio

Android Studio — це офіційне інтегроване середовище розробки (IDE), створене компанією Google для розробки мобільних застосунків на платформі Android. Воно базується на IntelliJ IDEA від JetBrains і надає повноцінне середовище для написання, тестування, налагодження та розгортання застосунків. Починаючи з 2018 року Android Studio отримало повноцінну підтримку фреймворку Flutter, завдяки чому стало одним з основних інструментів для Flutter-розробників.

Android Studio пропонує вбудовану підтримку Dart і Flutter через офіційні плагіни. IDE автоматично підтягує всі залежності, формує структуру проєкту, дозволяє запускати Hot Reload, переглядати структуру віджетів та інтегрувати Flutter Inspector[14].

Android Studio містить потужні засоби для візуального редагування макетів інтерфейсу, включаючи Layout Inspector, Device Preview, UI Debugging та інші. Це особливо корисно при створенні складних графічних інтерфейсів у Flutter.

Android Studio надає вбудований емулятор Android, який дозволяє швидко тестувати застосунок без потреби у фізичному пристрої. Емулятор підтримує різні версії Android, налаштування екрана, GPS, камеру тощо.

Android Studio інтегровано з системою збирання Gradle, яка дозволяє точно налаштовувати залежності, створювати різні варіанти збірки, підключати зовнішні бібліотеки та забезпечувати CI/CD.

IDE має вбудовані інструменти профілювання, які дозволяють детально аналізувати продуктивність застосунку: використання процесора, пам’яті, графічного інтерфейсу, потоків і батареї.

Android Studio спрощує інтеграцію з хмарними сервісами Google, такими як **Firebase**, **Google Maps**, **Analytics**, **Crashlytics** тощо, що дозволяє розширювати функціональність Flutter-застосунків.

Чому ж було обрано середовище розробки Visual Studio Code? Як і будь-яке середовище розробки, Android Studio має свої недоліки відносно іншого середовища Visual Studio Code.

В Android Studio швидкість запуску є повільнішим, оскільки відбувається більша витрата ресурсів. Середовище потребує багато оперативної пам'яті. Інтерфейс користувача буде складним для новачка, оскільки є перевантаженим – елементи, необхідні для розробки, не завжди розміщені в інтуїтивно-зрозумілих місцях. Більшість налаштувань відбуваються через графічні меню. Щодо системи плагінів, Visual Studio Code показує кращий результат, оскільки має легку доступність до них та широкий вибір у MarketPlace.

2.10. Supabase

Supabase — це сучасна, відкрита хмарна платформа, яка надає повноцінний функціонал для створення серверної частини застосунків. Вона позиціонується як відкрита альтернатива Firebase, заснована на PostgreSQL.[15] Платформа дозволяє розробникам швидко створювати бекенд без потреби у глибоких знаннях систем адміністрування баз даних, автентифікації чи REST API.

У контексті розробки Flutter-додатків Supabase забезпечує простий спосіб зберігання, обробки та доступу до даних, реалізації автентифікації користувачів, хостингу файлів тощо.

Supabase використовує потужну реляційну базу даних **PostgreSQL**, яка підтримує складні SQL-запити, зв’язки між таблицями, тригери, представлення (views), збережені процедури (stored procedures) та інші професійні можливості.

Після створення таблиці в базі даних Supabase автоматично генерує **REST API** та **WebSocket API** для взаємодії з даними. Це дозволяє миттєво підключати фронтенд-застосунок до серверної частини без написання серверного коду.

Supabase дозволяє реалізовувати функціональність у реальному часі: будь-

які зміни в таблицях можуть миттєво передаватися клієнтському застосунку через WebSocket-з'єднання. Це корисно, наприклад, для чатів, дошок завдань або спільного редагування.[16]

Supabase має вбудований модуль автентифікації, який підтримує:

- Email/Password логін
- OAuth2 (Google, GitHub, Apple тощо)
- Magic Links (посилання для входу без паролю)
- JSON Web Tokens (JWT) для авторизації

Це дозволяє легко керувати реєстрацією, входом, скиданням паролю та авторизацією запитів на сервері.

Supabase також надає можливість зберігати медіа-файли (зображення, відео, документи). Система дозволяє контролювати доступ до файлів на основі авторизації.

Supabase дозволяє розміщувати функції (написані на JavaScript/TypeScript) на сервері для обробки запитів або тригерів. Це аналог Firebase Cloud Functions, але базується на Deno.

Веб-інтерфейс Supabase дозволяє керувати базою даних, таблицями, користувачами, API, файлами та іншими ресурсами в зручному візуальному середовищі, без потреби писати SQL вручну (але з можливістю це робити).[17]

Переваги:

1 - Розробники мають повний контроль над структурою бази даних та запитами, на відміну від Firebase, де запити мають обмежений формат.

2 - Увесь код Supabase є відкритим, що дозволяє використовувати платформу локально або хостити власний екземпляр.

3 - Завдяки PostgreSQL можна будувати складні взаємозв’язки між

таблицями, що важливо для структурованих систем (курси, учні, викладачі, завдання тощо).

4 - REST та Realtime API створюються автоматично, що дозволяє скоротити час на розробку.

Недоліки:

1 - Supabase порівняно молода платформа (вперше представлена у 2020 році), тому вона ще не така стабільна як Firebase.

2 - На відміну від Firebase, Supabase не має вбудованої підтримки для роботи офлайн з автоматичною синхронізацією.

3 - Хоча Supabase активно розвивається, на даний момент вона має менше готових інтеграцій із сторонніми сервісами, ніж Firebase.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Проектування бази даних

Під час проєктування бази даних для мобільного додатку, основна увага була зосереджена на визначенні ключових об'єктів системи, які відповідають за навчальний процес. Метою розробки було створити гнучку та масштабовану структуру, яка б забезпечувала зберігання інформації про користувачів, курси, завдання, відповіді учнів, а також розклад занять.

На початковому етапі було визначено основні логічні сутності: **користувачі**, **курси**, **завдання**, **відповіді на завдання**, **розклад**, **календар**. Кожна з цих сутностей відображала окрему область функціональності додатку та мала свої атрибути й зв'язки.

Користувачі додатку поділяються на три ролі: **студент**, **викладач** та **адміністратор**. Реалізована система реєстрації з логіном та паролем дозволяє ідентифікувати користувача та обмежувати його доступ до певної частини функціоналу залежно від ролі.

У процесі проєктування таблиці було пов’язані між собою за допомогою **зовнішніх ключів (foreign keys)**, що забезпечило цілісність даних і спростило навігацію між об'єктами в додатку.

Структуру бази даних показано на рисунку 1. (Рис 3.1)

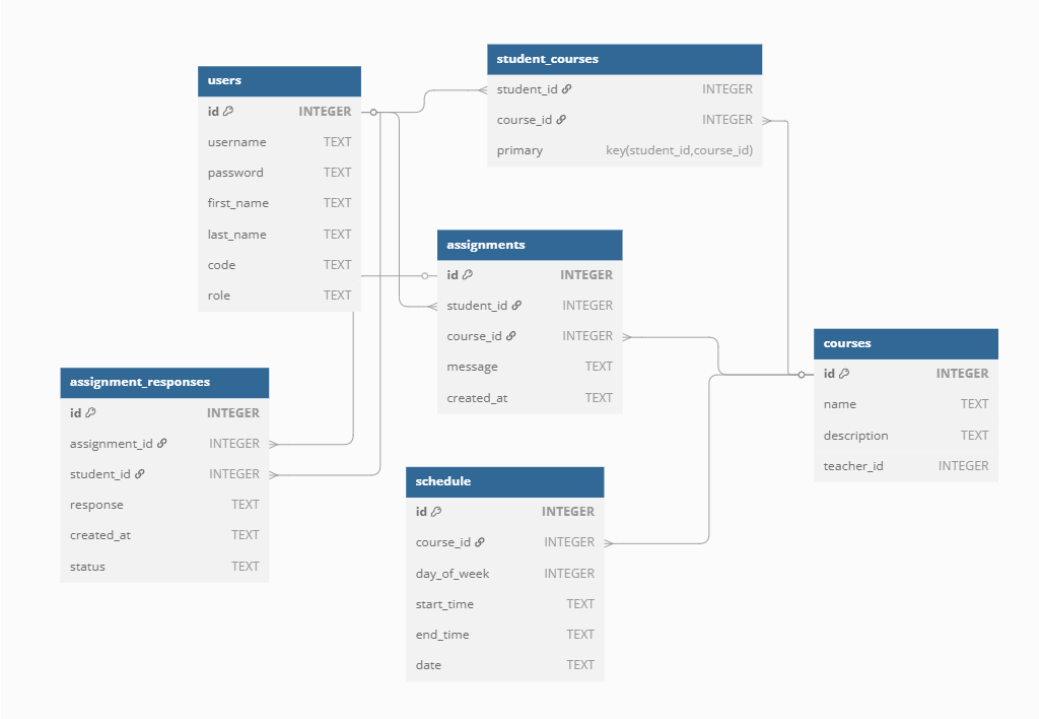

Рисунок 3.1 – Структура бази даних.

Аналізуючи структуру та використання своєї бази даних, я дійшов висновку, що однією з найбільш активно використовуваних таблиць буде таблиця завдань (assignments). Це зумовлено тим, що саме через неї реалізується основна взаємодія між викладачами та учнями: призначення завдань, передача повідомлень і подальше оцінювання відповіді на них.

Зважаючи на це, було прийнято рішення оптимізувати продуктивність бази даних шляхом індексування ключових полів у таблицях, які відповідають за зв’язки між користувачами, курсами та завданнями. Водночас я врахував і потенційні ризики індексації — наприклад, перевантаження пам’яті або уповільнення операцій вставки при надмірному використанні індексів. Тому рішення про індексацію було вибірковим: воно стосувалося лише тих полів, які критично впливають на швидкість виконання запитів і є найбільш часто використовуваними під час фільтрації та зв’язування даних між таблицями.

3.2 Серверна логіка та обробка даних

Засобами мови Dart та середовища розробки Visual Studio Code було розпочато роботу над створенням самого вигляду додатку. Головна сторінка мобільного додатку виступає початковим інтерфейсом, з яким взаємодіє користувач одразу після запуску застосунку. Її метою є ознайомлення користувача з концепцією додатку, надання візуальної інформації та можливості перейти до авторизації.

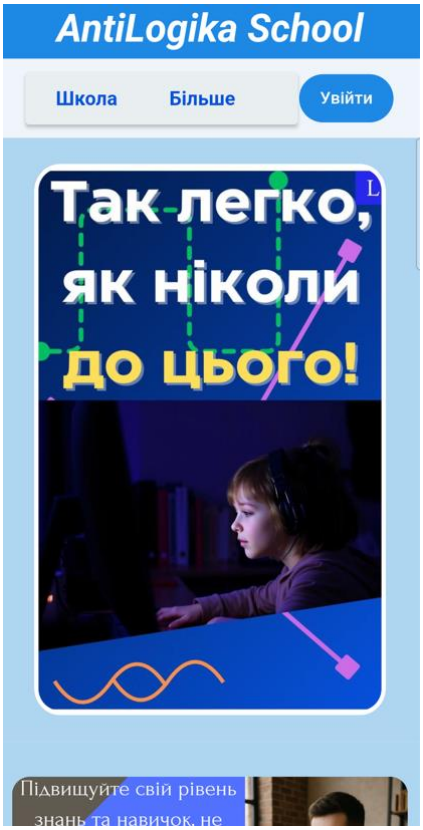

Рисунок 3.2 – Головне інформаційне вікно main.dart.

Інтерфейс головної сторінки реалізовано у файлі main.dart. Він складається з наступних структурних компонентів: AppBar із назвою проєкту "AntiLogika School"; MainMenu, яке являє собою меню навігації з переходами на інші допоміжні сторінки; MyHeader з логотипом школи та слайдером зображень; MyBody із трьома ілюстраціями, що підсилюють контент додатку, та кнопкою "Увійти", яка веде користувача на екран авторизації; MyFooter що завершує сторінку та виводить контактну

інформацію.

З технічної точки зору, головна сторінка працює як контейнер для демонстраційного контенту, який не потребує автентифікації. Перехід до подальшої взаємодії із системою можливий лише після натискання кнопки "Увійти", або після натискання на підпункт меню з тією ж назвою об’єкта, що здійснює навігацію до екрану логіну з допомогою метода Navigator.push.

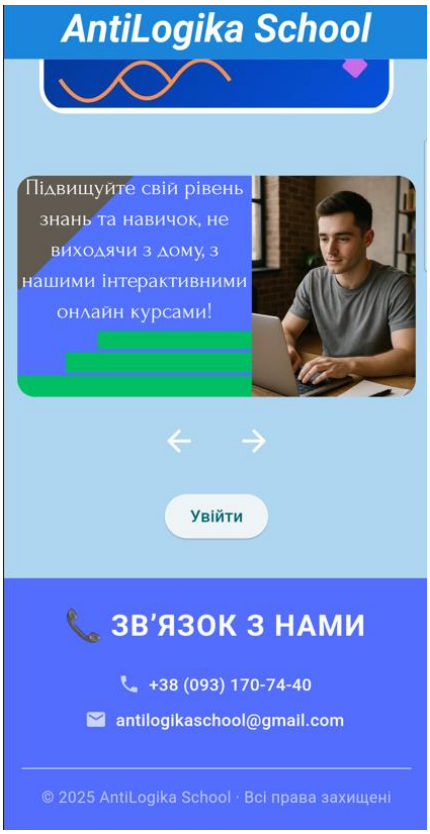

Рисунок 3.3 – Візуальне представлення кнопки “Увійти”.

Додатково при запуску застосунку ініціалізується Supabase-клієнт, а також встановлюється українська локалізація дати, що забезпечує правильне відображення календарів та дат у додатку.

Для забезпечення швидкої навігації між основними розділами мобільного додатку використовується спеціальне меню навігації, реалізоване у файлі menu_bar.dart. Цей компонент є візуальним елементом

верхньої частини інтерфейсу та виконує роль панелі керування, яка дає змогу користувачу переходити до ключових сторінок системи.

Меню містить у собі 2 пункти, які містять у собі підменю з двома елементами у кожного. Меню адаптовано до стилю додатку та виконано в єдиному кольоровому рішенні з використанням напівпрозорого фону.

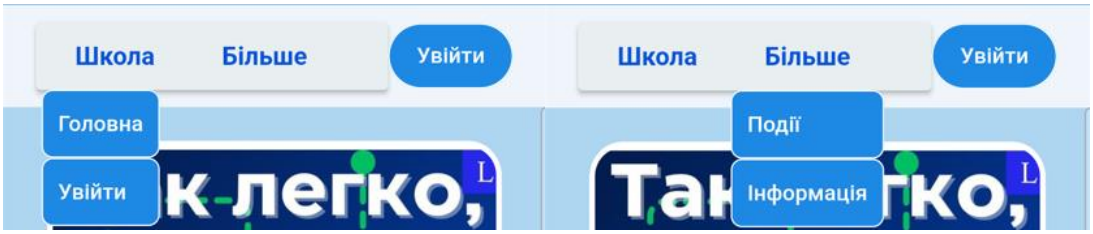

Рисунок 3.4 – Навігаційне меню з пунктами та піделементами.

Таким чином, навігаційне меню виконує функцію доступу до ключових сторінок та формує логічну структуру взаємодії користувача з додатком. Його використання спрощує переміщення між екранами та підвищує зручність роботи з інтерфейсом.

Сторінка подій реалізована у файлі events_screen.dart і призначена для інформування користувачів про майбутні або поточні події, що стосуються навчального процесу, діяльності школи або позашкільних заходів. Користувач, відкривши цю сторінку через навігаційне меню, потрапляє до розділу, в якому можуть бути розміщені оголошення про заняття, вебінари чи змагання, новини школи, розклад тематичних заходів. Зображення. Представлені в даному вікні, є інтерактивними елементами, при натисканні на які відкриватимуться тестові веб-сторінки, створені засобами Google Sites.

Основною метою даного вікна є забезпечення актуальної інформації для учнів та викладачів, сприяючи організації та залученню до шкільного життя.

Рисунок 3.5 –Інформаційна сторінка з елемента підменю «Події»

Інформаційна сторінка, реалізована у файлі info_page.dart, виконує роль довідкового або презентаційного розділу, де користувач може ознайомитися з основною інформацією про застосунок або освітню платформу. Дана сторінка розроблена у простому та зрозумілому стилі, з рекламними елементами. Відкривши дану сторінку, користувач може ознайомитись з основними даними про навчальний заклад.

Рисунок 3.6 –Інформаційна сторінка з елемента підменю «Інформація»

Сторінка авторизації, реалізована у файлі login_screen.dart, забезпечує процес автентифікації користувачів перед доступом до функціоналу додатку. Вона є точкою входу до особистих кабінетів учнів, викладачів та адміністраторів. Текстові поля введення логіну та паролю, реалізовані за допомогою TextField, дозволяють користувачу ввести облікові дані. Кнопка "Увійти", при натисканні якої викликається метод _login, що виконує перевірку введених даних. Для перевірки введених даних викликається звернення до бази даних. При успішному вході витягуються логін, ім’я, прізвище, роль.

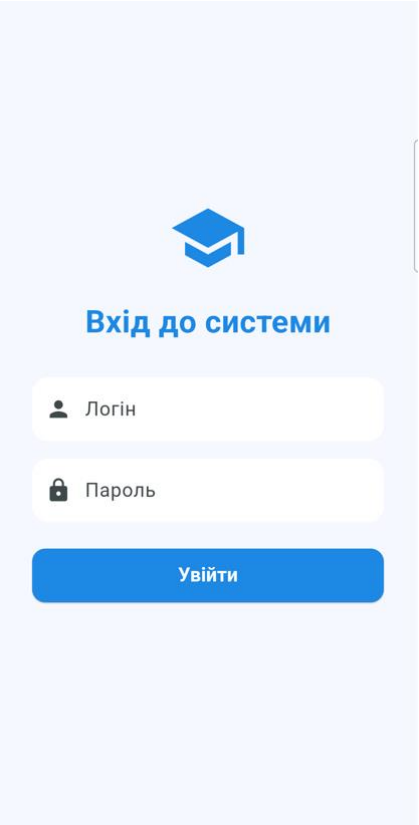

Рисунок 3.7 – Головна сторінка авторизації користувача

В залежності від введених даних, користувача перенаправить на відповідну сторінку – сторінку користувача, викладача або ж адміністратора.

Сторінка адміністратора, реалізована у файлі admin_screen.dart, є центральним інтерфейсом керування користувачами та навчальним процесом. Вона доступна виключно для користувачів із роллю admin після успішної автентифікації.

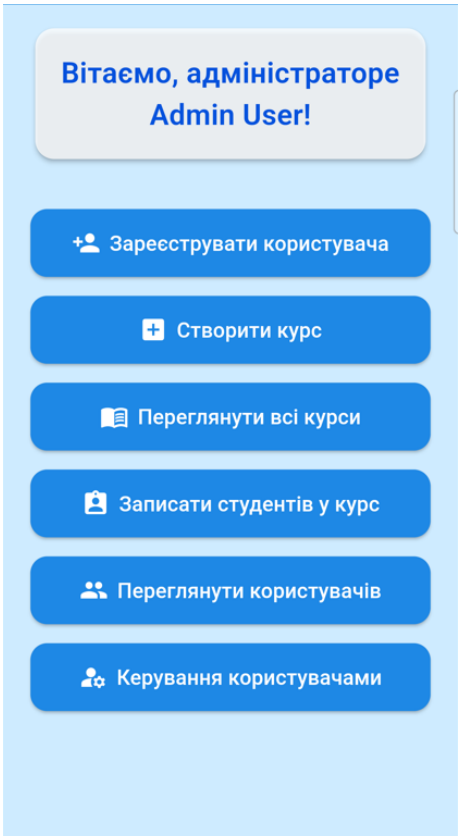

Рисунок 3.8 – Головна сторінка користувача в ролі «Адміністратора»

Після входу адміністратор бачить привітальне повідомлення з іменем та прізвищем, а також перелік функціональних кнопок для навігації до відповідних сторінок:

- Зареєструвати користувача
- Створити курс
- Переглянути всі курси
- Записати студентів у курс
- Переглянути користувачів
- Керування користувачами

Цей екран є основним засобом керування платформою для адміністратора. Він дозволяє здійснювати повний контроль над життєвим циклом користувачів і навчальними курсами, тим самим забезпечуючи гнучкість та адаптивність додатку до потреб навчального процесу.

Сторінка реєстрації, реалізована у файлі register_screen.dart, використовується адміністратором для створення нових облікових записів користувачів у системі. Вона забезпечує введення основної інформації про користувача та зберігає її в локальну базу даних.

На сторінці передбачено п’ять полів введення та один випадаючий список**:** Логін, Пароль, Ім’я, Прізвище, Роль користувача. Після натискання кнопки "Зареєструвати" викликається відповідний метод, який отримує дані, обробляє їх, перевіряє поля, вводить в базу даних SQLite та виводить відповідне повідомлення.

Ця сторінка дозволяє гнучко створювати нових учасників системи і тим самим керувати доступом до функціональності додатку відповідно до ролі. Завдяки спрощеному інтерфейсу адміністратор може швидко вводити нові облікові записи без складних дій.

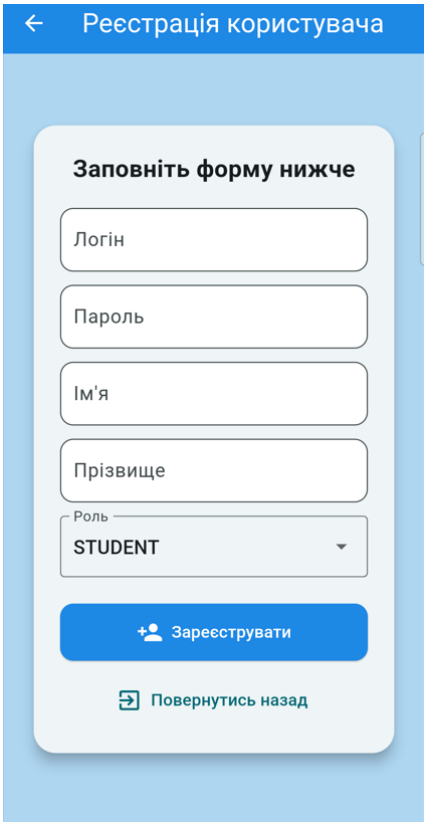

Рисунок 3.9 – Створення нового користувача адміністратором

Сторінка UserListScreen, реалізована у файлі userlist.dart, надає адміністратору можливість переглядати список зареєстрованих користувачів

(за винятком інших адміністраторів). Вона дозволяє візуалізувати базову інформацію про кожного користувача в системі.

Інтерфейс представлено у вигляді прокручуваного списку де кожен користувач відображається в окремому картковому віджеті. Для кожного користувача відображається ім’я, прізвище, логін, роль. Відображення паролів є конфіденційним правом адміністратора, для перегляду паролів використовується відповідна кнопка. Цей функціонал розроблений для відновлення даних про користувачів головним адміністратором при випадковій втраті певної частини інформації.

Ця сторінка служить для загального моніторингу користувачів у системі — переважно студентів і викладачів. Вона дозволяє адміністратору оперативно отримати інформацію про облікові записи, що особливо корисно у процесі супроводу освітнього середовища.

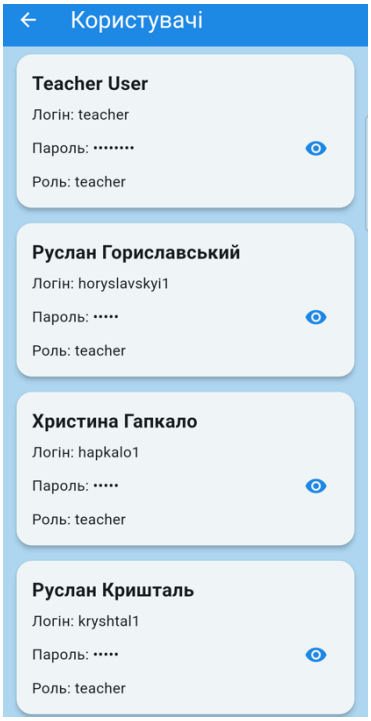

Рисунок 3.10 – Вікно з інформацією про усіх користувачів

Сторінка CoursesListScreen реалізована для перегляду наявних курсів у системі як з боку адміністратора, так і викладача. Вона також забезпечує можливість надсилання завдань викладачем та керування списком студентів,

закріплених за курсом. Дана сторінка забезпечує універсальний доступ до даних курсів, в залежності від ролі. Дане вікно викликається як у адміністратора, так і в викладача та виконує різні модифікації.

Кожен курс відображається у вигляді картки, яка містить назву курсу, опис, ім’я закріпленого викладача, кнопку надсилання завдання, список закріплених до курсу студентів з кнопкою можливості видалення учня з групи(IconButton).

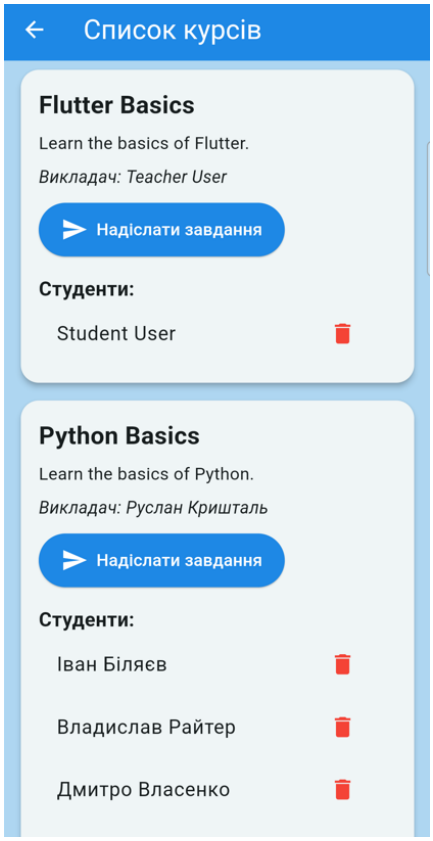

Рисунок 3.11 – Вікно з інформацією про усі курси

Сторінка CreateCourcesScreen реалізована для створення нового навчального курсу з можливістю призначення відповідального викладача. Це ще один функціонал, доступний тільки адміністратору системи.

Дане вікно надає можливість створити новий курс та прикріпити до нього існуючого викладача. Цей функціонал здійснюється з допомогою двох текстових полів – назви й опису курсу, а також випадаючого списку DropdownButtonFormField з існуючими викладачами. Ця сторінка дозволяє

адміністратору оперативно створювати нові навчальні курси в системі та призначати викладача, що забезпечує централізоване управління освітнім процесом. Вона є важливою частиною адміністративного функціоналу. Після натискання на кнопку «Зберегти курс», дані перевіряться на валідацію, сформуються та відправляться у базу даних.

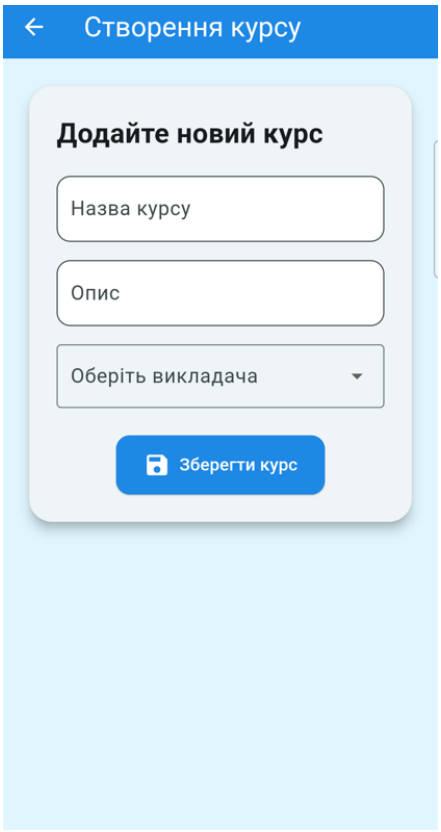

Рисунок 3.12 – Вікно з можливістю створення нового курсу

В результаті, буде виведено повідомлення про успішне створення нового курсу. Якщо перейти на вікно із випадаючим списком усіх курсів, можна знайти новостворений курс з відповідними даними.

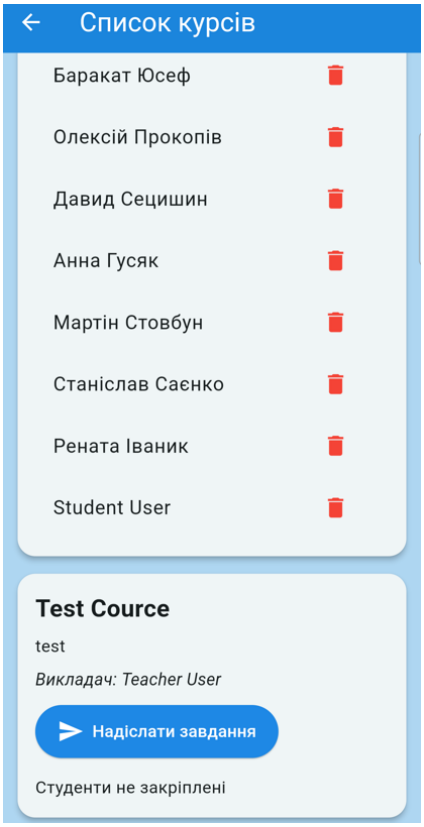

Рисунок 3.13 – Вікно з інформацією про усі курси, включно з новоствореним

Сторінка AssignStudentScreen реалізує інтерфейс для призначення студентів до певного навчального курсу. Цей функціонал є частиною адміністративного модуля та забезпечує повну керованість освітнього процесу з боку адміністратора.

На даному вікні розміщені основні компоненти функціоналу відповідного методу, а саме:

- Випадаючий список курсів
- Список студентів
- Кнопка відправлення та збереження даних

При натисканні на кнопку «Записати студентів», виконується перевірка на обраний курс та наявність вибраних студентів. Наступним кроком є запис кожного студента до курсу.

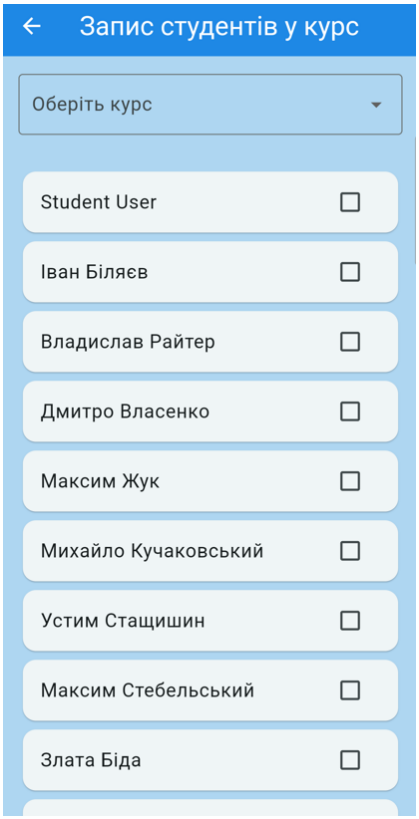

Рисунок 3.14 – Вікно з можливістю запису студента у курс

Результат буде виведено у вікні cources_list.dart біля вказаного попередньо курсу.

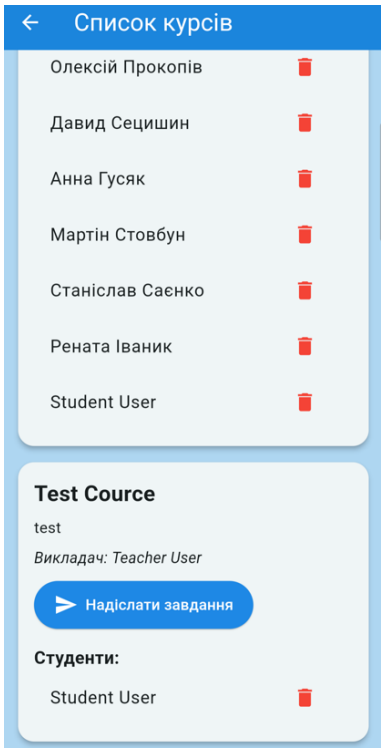

Рисунок 3.15 – Вікно з курсами та закріпленими студентами

Сторінка TeacherScreen виконує роль центрального інтерфейсу викладача, де реалізовані ключові функціональні можливості для організації освітнього процесу, керування курсами, моніторингу завдань і редагування розкладу.

При завантаженні інтерфейсу, викладач бачить вітальне повідомлення, яке динамічно відображає його ім’я та прізвище**,** передані через конструктор як аргументи firstName, lastName, а також teacherId— унікальний ідентифікатор у базі даних.

На головній сторінці з основних функцій ми можемо побачити:

1. Можливість перегляду курсів – кнопка, яка веде на сторінку CourcesListScreen, де відображаються лише курси, закріплені за викладачем;
2. Оцінювання завдань – невелике поле. У якому з’являтимуться виконані студентами завдання;
3. Календар занять – елемент календаря CalendarTable, де візуалізується розклад занять на тиждень;
4. Кнопка з можливістю редагування розкладу занять.

Усі операції виконуються асинхронно з використанням Future, що дозволяє не блокувати UI. Інтерфейс реалізовано у формі StatefulWidget з адаптацією під різні сценарії взаємодії.

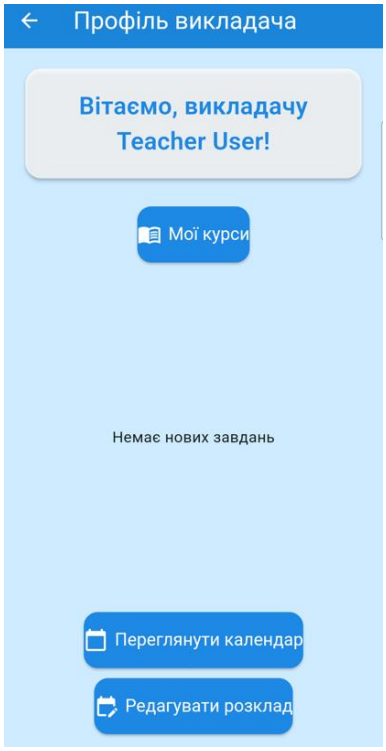

Рисунок 3.16 – Головна сторінка користувача в ролі «Викладача»

Після того, як натиснути кнопку “Мої курси”, для викладача відображатиметься вікно cources_list.dart, але, в даному випадку, викладачеві буде доступним функціонал надсилання завдання.

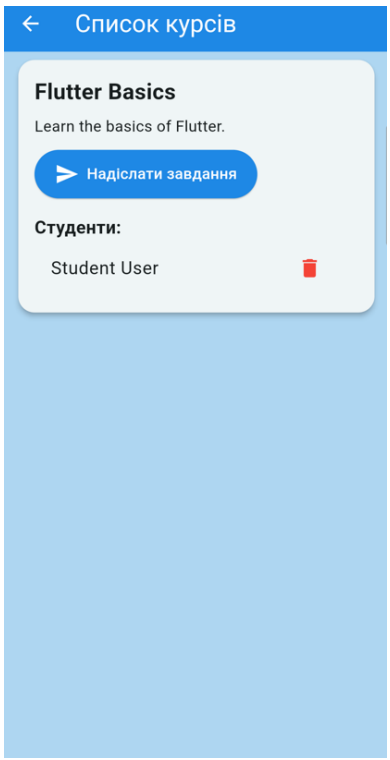

Рисунок 3.17 – Вікно з курсами (Викладач)

Після натискання на кнопку “Надіслати завдання”, для викладача відкриється вікно send_assignment.dart

Сторінка SendAssignmentScreen реалізує функцію створення та відправлення завдань викладачем студентам, які записані на певний курс. Вона дозволяє викладачу ввести повідомлення-завдання та вибрати студентів, яким це завдання буде надіслано.

Компонент приймає два обов’язкових аргументи: ідентифікатор курсу, на який записані студенти; ідентифікатор поточного викладача. Ці значення передаються через конструктор від зовнішнього виклику.

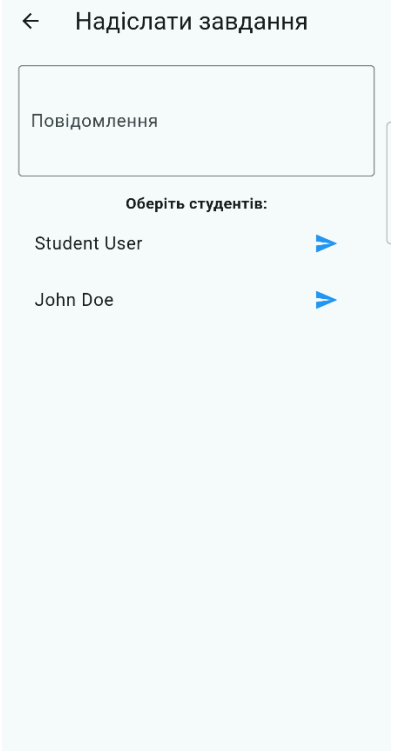

Рисунок 3.18 – Вікно надсилання завдання студентам

Компонент ScheduleEditorScreen надає інтерфейс для створення, редагування та видалення розкладу занять, які прив’язані до курсів викладача. Це ключова функціональність для підтримки актуального розкладу в навчальному середовищі мобільного додатку. Для створення запису, потрібно натиснути відповідну кнопку, яка закине розклад у масив та об’єднає його з назвою курсу.

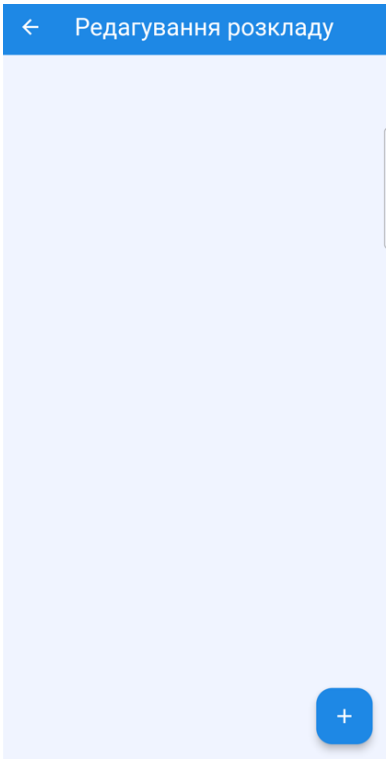

Рисунок 3.19 – Вікно створення, редагування або видалення розкладу занять певного курсу

У новій формі потрібно обрати назву курсу із списку, обрати дату першого заняття(здійснюється елементом DateTable), вписати дату початку та завершення заняття.

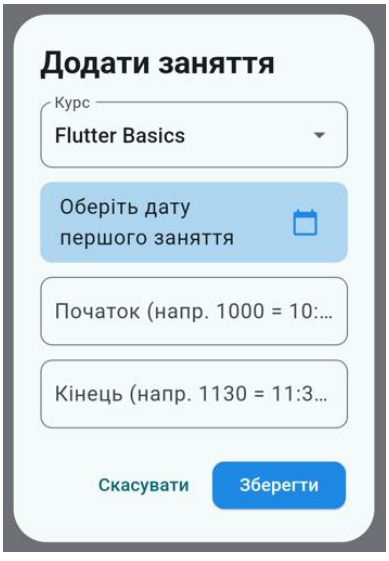

Рисунок 3.20 – Вікно створення розкладу

Після створення розкладу, автоматично створяться 10 наступних занять, що відділені один від одного днем тижня.

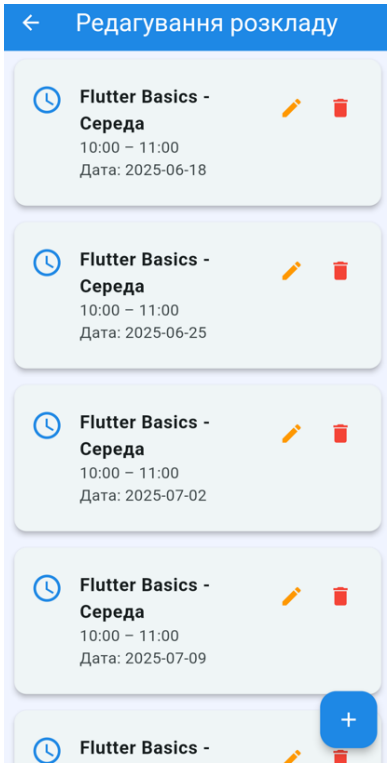

Рисунок 3.21 – Вікно зі створеними розкладами

Було створено два окремих файла, які мають схожу будову та функціонал – table_calendar.dart та teacher_calendar.dart. Дані файли реалізують інтерфейсний компонент календаря на основі сторонньої бібліотеки table_calendar, який використовується для візуалізації дат і занять. Дана бібліотека дозволяє переглядати дні і тижні у вигляді інтерактивного календаря, обробляти вибір дати у зручному вигляді. По базовим налаштуванням, до календарів підключений файл holidays.dart, який містить у собі визначні дати для України, а також кастомні дати навчального закладу. Після створення розкладу, у викладача та студента, закріплених за цим курсом, з’являться дати проведення занять у вигляді івент-подій починаючи від першого створеного елемента до останнього 10-го системного.

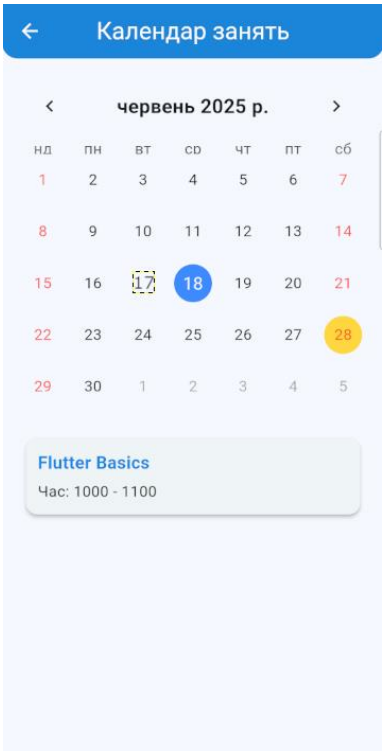

Рисунок 3.22 – Вікно календаря з визначеною датою заняття

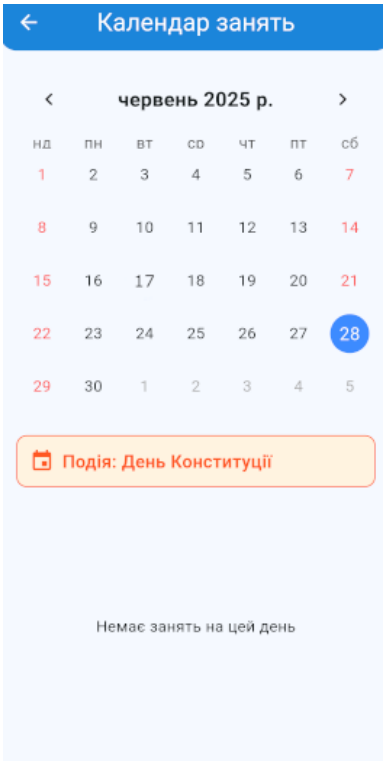

Рисунок 3.23 – Вікно календаря з відзначеною датою події

Файл student_screen.dart реалізує інтерфейс користувача для ролі "студент", забезпечуючи доступ до курсів, завдань, розкладу занять та подій у вигляді календаря.

За допомогою DatabaseHelper відбувається отримання з бази даних:

- Даних про поточного користувача;
- Курси, на які він записаний;
- Завдання, які були йому надіслані;
- Згенерований розклад занять.

На своїй сторінці, студент отримає Card з інформацією про завдання(текст завдання, назва курсу, дата надсилання, статус), поле з закріпленими курсами у вигляді ListView, розклад занять зі списком поточних та майбутніх занять, прохідний(вступний) календар без явного функціоналу.

У користувача є дві основні кнопки переходу до вікна розробки завдання середовищем HTML-редагування та переходу до функціонального вікна з календарем.

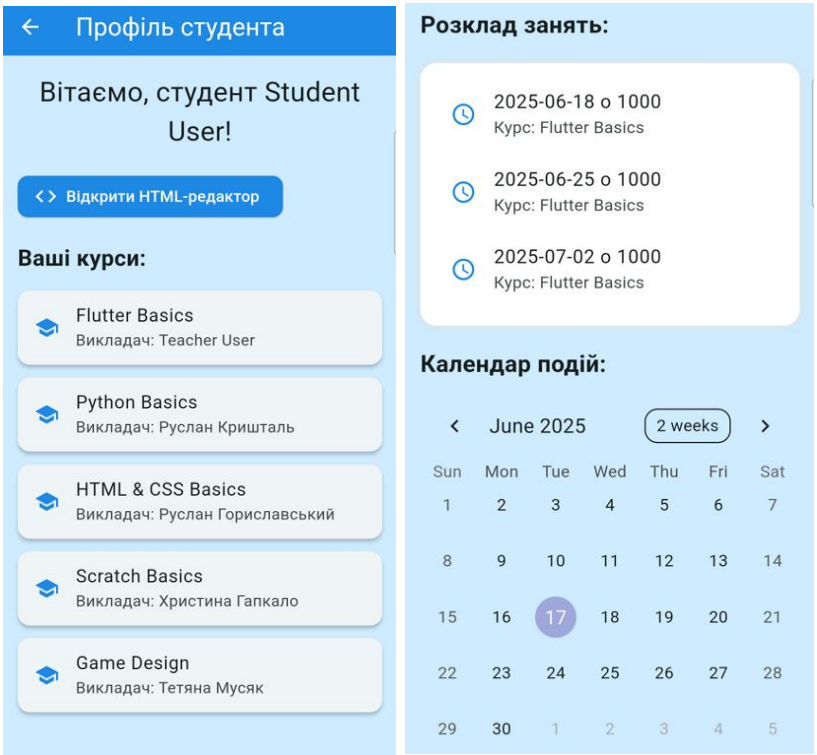

Рисунок 3.24 – Головна сторінка користувача в ролі «Студента»

Після отримання завдання від викладача, студент здатен відправити

відповідь, прикріпивши до текстового елемента відповідь. Після надсилання відповіді, викладач отримує відповідь від студента. До завдання передбачені дві дії – можливість позначити завдання як «виконано» або «не виконано».

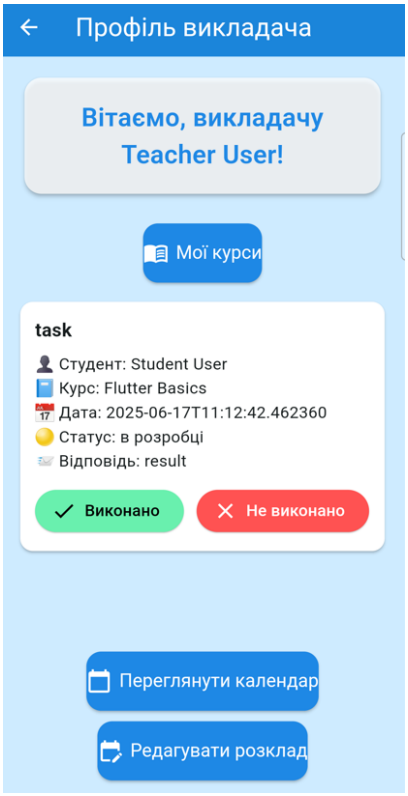

Рисунок 3.25 – Головна сторінка користувача в ролі «Викладача»(завдання від студента отримано)

При виборі викладача «Виконано», студент не матиме змоги змінити дані у відправленому завданні. Статус завдання змінюватиметься відносно вибору викладача.

3.3 Реалізація API

Для забезпечення інтерактивної перевірки HTML-коду у мобільному додатку було реалізовано окремий серверний застосунок (API), розміщений на платформі Render. Репозиторій з API доступний на GitHub за назвою html-api-flutter і містить основні конфігураційні файли: Dockerfile, main.py, render.yaml, requirements.txt.

Для візуалізації даного застосунка, було розроблено кнопку для студента можливості генерації коду мовою HTML з виведенням результату введеного коду.

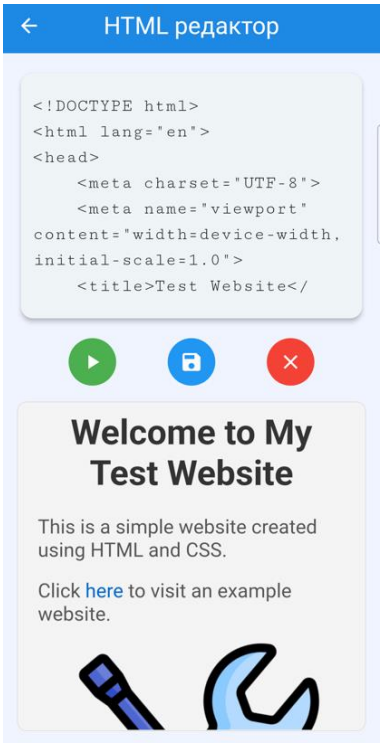

Рисунок 3.26 – Сторінка створення HTML-коду студентом

Серверна частина API написана мовою Python із використанням бібліотеки FastAPI. Основне завдання API — приймати HTML-код у форматі JSON-запиту, обробляти його та повертати повноцінний HTML-документ у відповідь. Це дозволяє виконувати та переглядати результат HTML-коду безпосередньо в додатку, не використовуючи сторонні сервіси.

Окрім цього, користувач може зберегти HTML-код у локальний файл на пристрої, якщо надано відповідні дозволи. Цей функціонал реалізовано з використанням бібліотек path_provider і permission_handler.

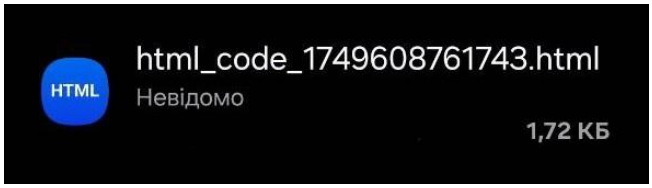

Рисунок 3.27 – Результат збереження коду на пристрій.

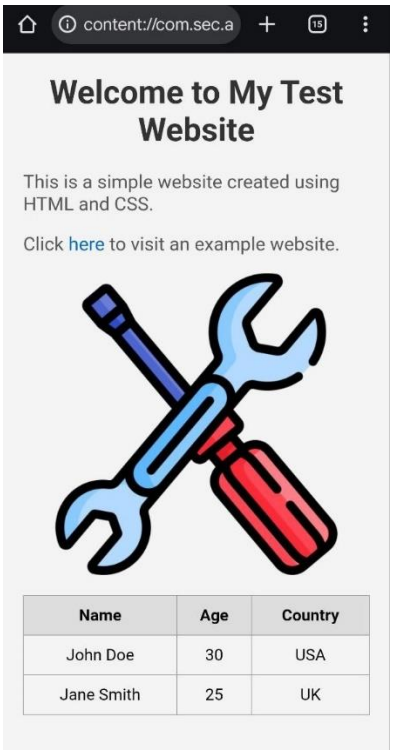

Рисунок 3.28 – Запуск збереженого файлу з допомогою браузера.

ВИСНОВКИ

Розроблено повнофункціональний мобільний застосунок на базі фреймворку Flutter, який призначений для організації навчального процесу у сфері програмування серед учнів віком від 8 до 16 років. Застосунок реалізує комплексний підхід до навчання, поєднуючи функціонал управління курсами, розкладом занять, виконанням завдань і перевіркою результатів.

Інтеграція редактора коду HTML дозволяє студенту писати код, не використовуючи сторонні додатки та дозволяє зручно переглядати та зберігати результат на свій пристрій. Також, студент має змогу надсилати завдання викладачеві, що створює гнучкість та зручність

Інтерфейс програми створено з урахуванням сучасних принципів UI/UX-дизайну: застосовано анімації, адаптивні віджети, зручну навігацію та візуально привабливі компоненти. Це забезпечує позитивний досвід для всіх категорій користувачів – як учнів, так і викладачів та адміністраторів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Napoli M. Beginning Flutter. Apress, 2019.
2. Vashisht A. Flutter Architectures. 2020.
3. Dart Language Guide. URL: https://dart.dev/guides (дата звернення: 09.04.2025).
4. Buckett C. Dart in Action. Manning, 2013.
5. Denton B. Mastering Visual Studio Code. Packt Publishing, 2020.
6. Gackenheimer C. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. O'Reilly Media, 2021.
7. Freeman A. Asynchronous Programming with Dart and Flutter. Apress, 2021.
8. Sqflite Plugin for Flutter. URL: https://pub.dev/packages/sqflite (дата звернення: 20.06.2025).
9. Vinayak. Mastering SQLite in Flutter: A Comprehensive Guide to Sqflite. Medium. URL: https://medium.com/ (дата звернення: 20.04.2025).
10. Owens M. The Definitive Guide to SQLite. Apress, 2010.
11. Relan K. Building REST APIs with Flask: Create Python Web Services with MySQL. Apress, 2019.
12. Render.com. URL: https://render.com (дата звернення: 26.04.2025).
13. WebView in Flutter, official documentation. URL: https://pub.dev/packages/webview_flutter (дата звернення: 14.04.2025).
14. Rose R. Flutter and Dart Cookbook. 2022.
15. Supabase Flutter Docs. URL: https://supabase.com/docs/guides/with-flutter (дата звернення: 20.06.2025).
16. Powell S. Building Backend Services with Supabase. Leanpub, 2023.
17. Supabase Handbook — Supabase team. Supabase.io, 2022. URL: https://supabase.com/docs (дата звернення: 29.04.2025).

ДОДАТКИ

ДОДАТОК А

login_screen.dart

```
import 'package:flutter/material.dart';
import 'database_holder/database_helper.dart';
import 'users_screen/student_screen.dart';
import 'users_screen/teacher_screen.dart';
import 'users_screen/admin_screen.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final DatabaseHelper _dbHelper = DatabaseHelper();

  void _login() async {
    String username = _usernameController.text.trim();
    String password = _passwordController.text.trim();

    if (username.isNotEmpty && password.isNotEmpty) {
      var userData = await _dbHelper.getUserData(username, password);

      if (userData != null) {
        String role = userData['role'];
        String firstName = userData['first_name'];
        String lastName = userData['last_name'];
        int userId = userData['id'];

        Widget targetScreen;

        switch (role) {
          case 'student':
            targetScreen = StudentScreen(
              firstName: firstName,
              lastName: lastName,
              username: username,
            );
```

```
          break;
        case 'teacher':
          targetScreen = TeacherScreen(
            firstName: firstName,
            lastName: lastName,
            teacherId: userId,
          );
          break;
        case 'admin':
          targetScreen = AdminScreen(
            firstName: firstName,
            lastName: lastName,
          );
          break;
        default:
          targetScreen = Scaffold(
            body: Center(child: Text('Невідома роль: $role')),
          );
      }

      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => targetScreen),
      );
    } else {
      _showSnackBar('Невірний логін або пароль!');
    }
  } else {
    _showSnackBar('Заповніть всі поля!');
  }
}

void _showSnackBar(String message) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      backgroundColor: Colors.red[400],
      behavior: SnackBarBehavior.floating,
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
```

```
backgroundColor: const Color.fromARGB(255, 245, 249, 255),
body: Center(
  child: SingleChildScrollView(
    padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 24),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        const Icon(Icons.school,
            size: 72, color: Color.fromARGB(255, 30, 136, 229)),
        const SizedBox(height: 20),
        const Text(
          'Вхід до системи',
          style: TextStyle(
            fontSize: 26,
            fontWeight: FontWeight.bold,
            color: Color.fromARGB(255, 30, 136, 229),
          ),
        ),
        const SizedBox(height: 30),
        TextField(
          controller: _usernameController,
          decoration: InputDecoration(
            labelText: 'Логін',
            prefixIcon: const Icon(Icons.person),
            filled: true,
            fillColor: Colors.white,
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(14),
              borderSide: BorderSide.none,
            ),
          ),
        ),
        const SizedBox(height: 16),
        TextField(
          controller: _passwordController,
          decoration: InputDecoration(
            labelText: 'Пароль',
            prefixIcon: const Icon(Icons.lock),
            filled: true,
            fillColor: Colors.white,
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(14),
              borderSide: BorderSide.none,
            ),
          ),
```

```
              obscureText: true,
            ),
            const SizedBox(height: 24),
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _login,
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color.fromARGB(255, 30, 136, 229),
                  foregroundColor: Colors.white,
                  padding: const EdgeInsets.symmetric(vertical: 14),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                  textStyle: const TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                child: const Text('Увійти'),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

ДОДАТОК Б

student_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_main/flutter_html_api/html_code_editor_screen.dart';
import 'package:flutter_main/database_holder/database_helper.dart';
import 'package:table_calendar/table_calendar.dart';
import 'package:flutter_main/table_calendar.dart';

class StudentScreen extends StatefulWidget {
  final String firstName;
  final String lastName;
  final String username;
```

```
  const StudentScreen({
    super.key,
    required this.firstName,
    required this.lastName,
    required this.username,
  });

  @override
  State<StudentScreen> createState() => _StudentScreenState();
}

class _StudentScreenState extends State<StudentScreen> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _courses = [];
  List<Map<String, dynamic>> _assignments = [];
  List<Map<String, dynamic>> _schedule = [];
  String dayOfWeekToText(int day) {
    switch (day) {
      case 1:
        return 'Понеділок';
      case 2:
        return 'Вівторок';
      case 3:
        return 'Середа';
      case 4:
        return 'Четвер';
      case 5:
        return 'П’ятниця';
      case 6:
        return 'Субота';
      case 7:
        return 'Неділя';
      default:
        return 'Невідомий день';
    }
  }

  Map<DateTime, List<String>> _events = {};
  DateTime _focusedDay = DateTime.now();
  DateTime? _selectedDay;

  @override
  void initState() {
    super.initState();
    _loadCourses();
```

```
  _loadData();
  _loadSchedule();
  _loadEvents();
}

Future<void> _loadCourses() async {
  final user = await dbHelper.getUser(widget.username);
  if (user != null) {
    int studentId = user['id'];
    final courses = await dbHelper.getCoursesForStudent(studentId);
    setState(() {
      _courses = courses;
    });
  }
}

Future<void> _loadData() async {
  final user = await dbHelper.getUser(widget.username);
  if (user != null) {
    int studentId = user['id'];
    final courses = await dbHelper.getCoursesForStudent(studentId);
    final assignments = await dbHelper.getAssignmentsForStudent(studentId);
    setState(() {
      _courses = courses;
      _assignments = assignments;
    });
  }
}

Future<void> _loadSchedule() async {
  final user = await dbHelper.getUser(widget.username);
  if (user != null) {
    final schedule =
        await dbHelper.getGeneratedScheduleForStudent(user['id']);
    setState(() {
      _schedule = schedule;
    });
  }
}

Future<void> _loadEvents() async {
  final user = await dbHelper.getUser(widget.username);
  if (user == null) return;
  int studentId = user['id'];
```

```
  var schedule = await dbHelper.getGeneratedScheduleForStudent(studentId);
  Map<DateTime, List<String>> events = {};

  for (var item in schedule) {
    final date = DateTime.parse(item['date']);
    final courseName = item['course_name'];
    final startTime = item['start_time'];

    events
        .putIfAbsent(date, () => [])
        .add('Заняття: $courseName о $startTime');
  }

  setState(() {
    _events = events;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 206, 235, 255),
    appBar: AppBar(
      backgroundColor: const Color.fromARGB(255, 30, 136, 229),
      foregroundColor: Colors.white,
      title: const Text('Профіль студента'),
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Center(
            child: Text(
              'Вітаємо, студент ${widget.firstName} ${widget.lastName}!',
              style: const TextStyle(fontSize: 24),
              textAlign: TextAlign.center,
            ),
          ),
          const SizedBox(height: 20),
          ElevatedButton.icon(
            icon: const Icon(Icons.code),
            label: const Text('Відкрити HTML-редактор'),
            onPressed: () {
              Navigator.push(
```

```
      context,
      MaterialPageRoute(
        builder: (context) => const HtmlCodeEditorScreen()),
    );
  },
  style: ElevatedButton.styleFrom(
    backgroundColor: const Color.fromARGB(255, 30, 136, 229),
    foregroundColor: Colors.white,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(10)),
  ),
),
const SizedBox(height: 20),
const Text('Ваші курси:',
    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
const SizedBox(height: 10),
ListView.builder(
  shrinkWrap: true,
  physics: const NeverScrollableScrollPhysics(),
  itemCount: _courses.length,
  itemBuilder: (context, index) {
    final course = _courses[index];
    return Card(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12)),
      elevation: 3,
      margin: const EdgeInsets.symmetric(vertical: 6),
      child: ListTile(
        leading: const Icon(Icons.school,
          color: Color.fromARGB(255, 30, 136, 229)),
        title: Text(course['name']),
        subtitle: Text('Викладач: ${course['teacher_name']}'),
      ),
    );
  },
),
const SizedBox(height: 30),
const Text('Ваші завдання:',
    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
const SizedBox(height: 10),
ListView.builder(
  shrinkWrap: true,
  physics: const NeverScrollableScrollPhysics(),
  itemCount: _assignments.length,
  itemBuilder: (context, index) {
```

```
            final assignment = _assignments[index];
            return Card(
              elevation: 2,
              margin: const EdgeInsets.symmetric(vertical: 6),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  ListTile(
                    title: Text(assignment['message']),
                    subtitle: Text(
                      'Курс: ${assignment['course_name']} \nНадіслано: ${assignment['created_at']} \nСтатус:
${assignment['status'] ?? 'в розробці'}'),
                  ),
                  Padding(
                    padding: const EdgeInsets.only(left: 16.0, bottom: 8),
                    child: ElevatedButton(
                      onPressed: (assignment['status'] == 'виконано')
                          ? null
                          : () async {
                              final user =
                                  await dbHelper.getUser(widget.username);
                              final studentId = user?['id'];
                              if (studentId == null) {
                                ScaffoldMessenger.of(context).showSnackBar(
                                  const SnackBar(
                                      content: Text(
                                          'Не вдалося визначити ID студента')),
                                );
                                return;
                              }

                              final controller = TextEditingController();

                              showDialog(
                                context: context,
                                builder: (context) {
                                  return AlertDialog(
                                    title:
                                        const Text('Відповідь на завдання'),
                                    content: TextField(
                                      controller: controller,
                                      maxLines: 4,
                                      decoration: const InputDecoration(
                                        hintText: 'Введіть відповідь',
                                      ),
```

```
),
actions: [
  TextButton(
    onPressed: () async {
      final responseText =
          controller.text.trim();
      if (responseText.isEmpty) {
        Navigator.pop(context);
        ScaffoldMessenger.of(context)
            .showSnackBar(
          const SnackBar(
              content: Text(
                  'Відповідь не може бути порожньою')),
        );
        return;
      }

      try {
        final result = await dbHelper
            .submitAssignmentResponse(
          assignmentId: assignment[
              'assignment_id'],
          studentId: studentId,
          response: responseText,
        );

        Navigator.pop(context);

        if (result > 0) {
          ScaffoldMessenger.of(context)
              .showSnackBar(
            const SnackBar(
                content: Text(
                    'Відповідь надіслана успішно')),
          );
        } else {
          ScaffoldMessenger.of(context)
              .showSnackBar(
            const SnackBar(
                content: Text(
                    'Помилка при надсиланні відповіді')),
          );
        }

        _loadData();
```

```
                        } catch (e) {
                          Navigator.pop(context);
                          ScaffoldMessenger.of(context)
                              .showSnackBar(
                            SnackBar(
                                content: Text(
                                    'Виникла помилка: $e')),
                          );
                        }
                      },
                      child: const Text('Надіслати'),
                    ),
                    TextButton(
                      onPressed: () {
                        Navigator.pop(context);
                      },
                      child: const Text('Скасувати'),
                    ),
                  ],
                );
              },
            );
          },
      child: Text(
        (assignment['status'] == 'виконано')
            ? 'Завдання виконано'
            : 'Надіслати відповідь',
      ),
    ),
  ),
 ],
 ),
 );
 },
),
const Text('Розклад занять:',
    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
const SizedBox(height: 20),
Container(
  padding: const EdgeInsets.all(12),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(15),
    boxShadow: [
      BoxShadow(color: Colors.grey.shade300, blurRadius: 6)
```

```
    ],
  ),
  child: SizedBox(
    height: 220,
    child: ListView.builder(
      itemCount: _schedule.length,
      itemBuilder: (context, index) {
        final item = _schedule[index];
        return ListTile(
          leading: const Icon(Icons.access_time,
              color: Color.fromARGB(255, 30, 136, 229)),
          title: Text('${item['date']} o ${item['start_time']}'),
          subtitle: Text('Курс: ${item['course_name']}'),
        );
      },
    ),
  ),
),
const SizedBox(height: 20),
const Text('Календар подій:',
    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
const SizedBox(height: 10),
TableCalendar<String>(
  calendarStyle: CalendarStyle(
    selectedDecoration: BoxDecoration(
      color: Color.fromARGB(255, 30, 136, 229),
      shape: BoxShape.circle,
    ),
    markerDecoration: BoxDecoration(
      color: Color.fromARGB(190, 30, 136, 229),
      shape: BoxShape.circle,
    ),
  ),
  firstDay: DateTime.utc(2020, 1, 1),
  lastDay: DateTime.utc(2030, 12, 31),
  focusedDay: _focusedDay,
  selectedDayPredicate: (day) => isSameDay(_selectedDay, day),
  eventLoader: (day) => _events[day] ?? [],
  onDaySelected: (selectedDay, focusedDay) {
    setState(() {
      _selectedDay = selectedDay;
      _focusedDay = focusedDay;
    });
  },
  onPageChanged: (focusedDay) => _focusedDay = focusedDay,
```

```
          ),
          const SizedBox(height: 10),
          ElevatedButton(
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) =>
                        TableCalendarPage(schedule: _schedule)),
              );
            },
            child: const Text('Відкрити повний календар'),
          ),
          if (_selectedDay != null &&
              (_events[_selectedDay] ?? []).isNotEmpty)
            ..._events[_selectedDay]!.map(
              (e) => ListTile(
                leading: const Icon(Icons.event),
                title: Text(e),
              ),
            ),
        ],
      ),
    ),
  );
 }
}
```

ДОДАТОК В

send_assignment.dart

```
import 'package:flutter/material.dart';
import '../database_holder/database_helper.dart';

class SendAssignmentScreen extends StatefulWidget {
 final int courseId;
 final int teacherId;

 const SendAssignmentScreen({
  super.key,
  required this.courseId,
  required this.teacherId,
 });

 @override
```

```
  State<SendAssignmentScreen> createState() => _SendAssignmentScreenState();
}

class _SendAssignmentScreenState extends State<SendAssignmentScreen> {
  final DatabaseHelper _dbHelper = DatabaseHelper();
  final TextEditingController _messageController = TextEditingController();
  List<Map<String, dynamic>> _students = [];

  @override
  void initState() {
    super.initState();
    _loadStudents();
  }

  Future<void> _loadStudents() async {
    final students = await _dbHelper.getStudentsForCourse(widget.courseId);
    setState(() {
      _students = students;
    });
  }

  Future<void> _sendAssignment(int studentId) async {
    final message = _messageController.text.trim();
    if (message.isEmpty) return;

    final isOwner =
        await _dbHelper.isTeacherOfCourse(widget.teacherId, widget.courseId);
    if (!isOwner) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content:
                Text('У вас немає прав надсилати завдання для цього курсу')),
      );
      return;
    }

    await _dbHelper.sendAssignment(
      studentId: studentId,
      courseId: widget.courseId,
      message: message,
    );
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Завдання надіслано!')),
    );
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Надіслати завдання')),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            TextField(
              controller: _messageController,
              maxLines: 3,
              decoration: const InputDecoration(
                labelText: 'Повідомлення',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 16),
            const Text('Оберіть студентів:',
                style: TextStyle(fontWeight: FontWeight.bold)),
            Expanded(
              child: ListView.builder(
                itemCount: _students.length,
                itemBuilder: (context, index) {
                  final student = _students[index];
                  final fullName =
                      '${student['first_name']} ${student['last_name']}';
                  return ListTile(
                    title: Text(fullName),
                    trailing: IconButton(
                      icon: const Icon(Icons.send, color: Colors.blue),
                      onPressed: () => _sendAssignment(student['id']),
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

ДОДАТОК Ґ

teacher_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_main/cources&groups/courses_list.dart';
import 'package:flutter_main/database_holder/database_helper.dart';
import 'package:flutter_main/schedule_editor.dart';
import 'package:flutter_main/teacher_calendar.dart';

class TeacherScreen extends StatefulWidget {
  final String firstName;
  final String lastName;
  final int teacherId;

  const TeacherScreen({
    super.key,
    required this.firstName,
    required this.lastName,
    required this.teacherId,
  });

  @override
  State<TeacherScreen> createState() => _TeacherScreenState();
}

class _TeacherScreenState extends State<TeacherScreen> {
  final DatabaseHelper dbHelper = DatabaseHelper();
  List<Map<String, dynamic>> _assignments = [];

  @override
  void initState() {
    super.initState();
    _loadAssignments();
  }

  Future<void> _loadAssignments() async {
    final assignments =
        await dbHelper.getAssignmentResponsesForTeacher(widget.teacherId);
    setState(() {
      _assignments = assignments;
    });
  }

  Future<void> _updateStatus(
```

```
    int assignmentId, int studentId, String status) async {
  await dbHelper.updateAssignmentStatus(
    assignmentId: assignmentId,
    studentId: studentId,
    newStatus: status,
  );
  await _loadAssignments();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 206, 235, 255),
    appBar: AppBar(
      title: const Text('Профіль викладача'),
      backgroundColor: const Color.fromARGB(255, 30, 136, 229),
      foregroundColor: Colors.white,
      elevation: 4,
    ),
    body: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 16),
      child: Column(
        children: [
          Card(
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(16)),
            color: const Color.fromARGB(200, 255, 255, 255),
            elevation: 3,
            child: Padding(
              padding: const EdgeInsets.all(20),
              child: Text(
                'Вітаємо, викладачу ${widget.firstName} ${widget.lastName}!',
                style: const TextStyle(
                  fontSize: 22,
                  fontWeight: FontWeight.w600,
                  color: Color.fromARGB(255, 30, 136, 229),
                ),
                textAlign: TextAlign.center,
              ),
            ),
          ),
          const SizedBox(height: 24),
          _buildMainButton(
            label: 'Мої курси',
            icon: Icons.menu_book,
```

```
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) =>
            CoursesListScreen(teacherId: widget.teacherId),
      ),
    );
  },
),
const SizedBox(height: 10),
Expanded(
  child: _assignments.isEmpty
      ? const Center(child: Text('Немає нових завдань'))
      : ListView.builder(
          itemCount: _assignments.length,
          itemBuilder: (context, index) {
            final assignment = _assignments[index];
            return Card(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(12),
              ),
              elevation: 2,
              margin: const EdgeInsets.symmetric(vertical: 8),
              color: Colors.white,
              child: Padding(
                padding: const EdgeInsets.all(14.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      assignment['message'] ?? 'Немає повідомлення',
                      style: const TextStyle(
                        fontWeight: FontWeight.bold,
                        fontSize: 16,
                      ),
                    ),
                    const SizedBox(height: 8),
                    Text(
                      ' Студент: ${assignment['student_name'] ?? 'Невідомо'}\n'
                      ' Курс: ${assignment['course_name'] ?? 'Без назви'}\n'
                      ' Дата: ${assignment['created_at'] ?? '---'}\n'
                      ' Статус: ${assignment['status'] ?? 'в розробці'}\n'
                      ' Відповідь: ${assignment['response'] ?? 'Немає'}',
                      style: const TextStyle(height: 1.5),
```

```
),
const SizedBox(height: 12),
Wrap(
 spacing: 12,
 runSpacing: 8,
 children: [
  ElevatedButton.icon(
   onPressed: () {
    _updateStatus(
     assignment['assignment_id'],
     assignment['student_id'],
     'виконано',
    );
   },
   icon: const Icon(Icons.check),
   label: const Text('Виконано'),
   style: ElevatedButton.styleFrom(
    backgroundColor: Colors.greenAccent,
    foregroundColor: Colors.black,
   ),
  ),
  ElevatedButton.icon(
   onPressed: () {
    _updateStatus(
     assignment['assignment_id'],
     assignment['student_id'],
     'не виконано',
    );
   },
   icon: const Icon(Icons.close),
   label: const Text('Не виконано'),
   style: ElevatedButton.styleFrom(
    backgroundColor: Colors.redAccent,
    foregroundColor: Colors.white,
   ),
  ),
  if (assignment['status'] == 'виконано')
   ElevatedButton.icon(
    onPressed: () async {
     final confirmed =
       await showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
       title:
         const Text('Підтвердження'),
```

```
                        content: const Text(
                            'Ви впевнені, що хочете видалити це завдання для всіх студентів?'),
                        actions: [
                          TextButton(
                            onPressed: () =>
                                Navigator.of(context)
                                    .pop(false),
                            child:
                                const Text('Скасувати'),
                          ),
                          TextButton(
                            onPressed: () =>
                                Navigator.of(context)
                                    .pop(true),
                            child: const Text('Так'),
                          ),
                        ],
                      ),
                    );

                    if (confirmed == true) {
                      await dbHelper
                          .deleteAssignmentForAll(
                              assignment[
                                  'assignment_id']);
                      await _loadAssignments();
                    }
                  },
                  icon: const Icon(Icons.delete),
                  label: const Text('Видалити'),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.grey[400],
                    foregroundColor: Colors.black,
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  ),
  const SizedBox(height: 10),
```

```
        _buildMainButton(
          label: 'Переглянути календар',
          icon: Icons.calendar_today,
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
                    TeacherCalendarPage(teacherId: widget.teacherId),
              ),
            );
          },
        ),
        const SizedBox(height: 10),
        _buildMainButton(
          label: 'Редагувати розклад',
          icon: Icons.edit_calendar,
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
                    ScheduleEditorScreen(teacherId: widget.teacherId),
              ),
            );
          },
        ),
      ],
    ),
  ),
 );
}

Widget _buildMainButton(
   {required String label,
   required IconData icon,
   required VoidCallback onPressed}) {
 return ElevatedButton.icon(
   onPressed: onPressed,
   icon: Icon(icon),
   label: Text(label),
   style: ElevatedButton.styleFrom(
     padding: const EdgeInsets.symmetric(vertical: 16),
     backgroundColor: const Color.fromARGB(255, 30, 136, 229),
     foregroundColor: Colors.white,
```

```
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(14),
      ),
      elevation: 2,
      textStyle: const TextStyle(fontSize: 16),
    ),
  );
 }
}
```

ДОДАТОК Д

admin_screen.dart

```
import 'package:flutter/material.dart';
import '../register_screen.dart';
import 'package:flutter_main/cources&groups/create_cources.dart';
import 'package:flutter_main/cources&groups/courses_list.dart';
import 'package:flutter_main/cources&groups/asign_student.dart';
import 'package:flutter_main/userlist.dart';
import 'package:flutter_main/delete_users.dart';

class AdminScreen extends StatelessWidget {
 final String firstName;
 final String lastName;

 const AdminScreen({
  super.key,
  required this.firstName,
  required this.lastName,
 });

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: const Color.fromARGB(255, 206, 235, 255),
   appBar: AppBar(
    title: const Text('Профіль адміністратора'),
    backgroundColor: const Color.fromARGB(255, 30, 136, 229),
    foregroundColor: Colors.white,
    elevation: 4,
   ),
   body: Padding(
```

```
padding: const EdgeInsets.symmetric(horizontal: 24.0, vertical: 16),
child: Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    Card(
      elevation: 3,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(16),
      ),
      color: const Color.fromARGB(200, 255, 255, 255),
      child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Text(
          'Вітаємо, адміністраторе $firstName $lastName!',
          textAlign: TextAlign.center,
          style: const TextStyle(
            fontSize: 22,
            fontWeight: FontWeight.w600,
            color: Color.fromARGB(255, 6, 82, 221),
          ),
        ),
      ),
    ),
    const SizedBox(height: 30),
    _buildAdminButton(
      context,
      label: 'Зареєструвати користувача',
      icon: Icons.person_add,
      screen: const RegisterScreen(),
    ),
    _buildAdminButton(
      context,
      label: 'Створити курс',
      icon: Icons.add_box,
      screen: const CreateCourseScreen(),
    ),
    _buildAdminButton(
      context,
      label: 'Переглянути всі курси',
      icon: Icons.menu_book,
      screen: const CoursesListScreen(),
    ),
    _buildAdminButton(
      context,
      label: 'Записати студентів у курс',
```

```
            icon: Icons.assignment_ind,
            screen: const AssignStudentsScreen(),
          ),
          _buildAdminButton(
            context,
            label: 'Переглянути користувачів',
            icon: Icons.group,
            screen: const UsersListScreen(),
          ),
          _buildAdminButton(
            context,
            label: 'Керування користувачами',
            icon: Icons.manage_accounts,
            screen: const DeleteUsersScreen(),
          ),
        ],
      ),
    ),
  );
}

Widget _buildAdminButton(BuildContext context,
    {required String label, required IconData icon, required Widget screen}) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8.0),
    child: ElevatedButton.icon(
      style: ElevatedButton.styleFrom(
        backgroundColor: const Color.fromARGB(255, 30, 136, 229),
        foregroundColor: Colors.white,
        padding: const EdgeInsets.symmetric(vertical: 16),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(14),
        ),
        elevation: 2,
      ),
      icon: Icon(icon, size: 24),
      label: Text(
        label,
        style: const TextStyle(fontSize: 16, fontWeight: FontWeight.w500),
      ),
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => screen),
        );
```

```
        },
      ),
    );
  }
}
```

ДОДАТОК Е

create_cource.dart

```
import 'package:flutter/material.dart';
import '../database_holder/database_helper.dart';

class CreateCourseScreen extends StatefulWidget {
  const CreateCourseScreen({super.key});

  @override
  State<CreateCourseScreen> createState() => _CreateCourseScreenState();
}

class _CreateCourseScreenState extends State<CreateCourseScreen> {
  final TextEditingController _titleController = TextEditingController();
  final TextEditingController _descriptionController = TextEditingController();
  final DatabaseHelper _dbHelper = DatabaseHelper();

  List<Map<String, dynamic>> _teachers = [];
  int? _selectedTeacherId;

  @override
  void initState() {
    super.initState();
    _loadTeachers();
  }

  Future<void> _loadTeachers() async {
    final teachers = await _dbHelper.getAllTeachers();
    setState(() {
      _teachers = teachers;
    });
  }

  void _saveCourse() async {
    String title = _titleController.text.trim();
    String description = _descriptionController.text.trim();
```

```
  if (title.isNotEmpty &&
      description.isNotEmpty &&
      _selectedTeacherId != null) {
    await _dbHelper.insertCourse(title, description, _selectedTeacherId!);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Курс створено!')),
    );
    Navigator.pop(context);
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
          content: Text('Заповніть усі поля та оберіть викладача!')),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFFe1f5fe),
    appBar: AppBar(
      backgroundColor: const Color.fromARGB(255, 30, 136, 229),
      foregroundColor: Colors.white,
      title: const Text('Створення курсу'),
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(20),
      child: Card(
        elevation: 8,
        shape:
            RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
        child: Padding(
          padding: const EdgeInsets.all(24.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Text(
                'Додайте новий курс',
                style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
              ),
              const SizedBox(height: 20),
              _buildTextField(_titleController, 'Назва курсу'),
              const SizedBox(height: 16),
              _buildTextField(_descriptionController, 'Опис'),
              const SizedBox(height: 16),
```

```
            DropdownButtonFormField<int>(
              value: _selectedTeacherId,
              decoration: const InputDecoration(
                labelText: 'Оберіть викладача',
                border: OutlineInputBorder(),
              ),
              items: _teachers.map((teacher) {
                final fullName =
                    "${teacher['first_name']} ${teacher['last_name']}";
                return DropdownMenuItem<int>(
                  value: teacher['id'] as int,
                  child: Text(fullName),
                );
              }).toList(),
              onChanged: (value) {
                setState(() {
                  _selectedTeacherId = value;
                });
              },
            ),
            const SizedBox(height: 24),
            Center(
              child: ElevatedButton.icon(
                onPressed: _saveCourse,
                icon: const Icon(Icons.save),
                label: const Text('Зберегти курс'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color.fromARGB(255, 30, 136, 229),
                  foregroundColor: Colors.white,
                  padding: const EdgeInsets.symmetric(
                      vertical: 14, horizontal: 24),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
  Widget _buildTextField(TextEditingController controller, String label) {
    return TextField(
      controller: controller,
      decoration: InputDecoration(
        labelText: label,
        border: OutlineInputBorder(borderRadius: BorderRadius.circular(12)),
        fillColor: Colors.white,
        filled: true,
      ),
    );
  }
}
```