

Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня вищої освіти бакалавра
зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка мобільного застосунку для планування розкладу репетиторів на базі Android

Засвідчую, що в цій
кваліфікаційній роботі
немає запозичень із праць
інших авторів без
відповідних посилань.
Студент гр. ІПЗ–22–1
_____ /Д. А. Дорохін /

Керівник кваліфікаційної роботи	_____	/ <u>Д. В. Швець</u> /
Завідувач кафедри	_____	/ <u>А. М. Стрюк</u> /

Кривий Ріг
2026

Криворізький національний університет
Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: бакалавр
Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«_» _____ 2026р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІІЗ–22–1 Дорохін Денис Андрійович

1. Тема: Розробка мобільного застосунку для планування розкладу репетиторів на базі Android затверджено наказом по КНУ № 22 від «28» листопада 2025 р.
2. Термін подання студентом закінченої роботи: «05» червня 2026 р.
3. Вихідні дані по роботі: розроблюваний мобільний застосунок повинен забезпечувати збереження та обробку даних учнів.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити):
виконати аналіз існуючих мобільних додатків та методів планування розкладу для репетиторів, визначити вимоги до функціональності та інтерфейсу мобільного застосунку, спроектувати архітектуру розкладу репетиторів, здійснити програмну реалізацію на базі Android, реалізувати роботу з базою даних щоб зберегти інформації про учнів та заняття, провести тестування розробленого мобільного додатка.
5. Перелік ілюстративного матеріалу: діаграма архітектури мобільного застосунку , блок–схеми розроблених алгоритмів, схема взаємодії модулів додатка, знімки екранних форм інтерфейсу.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою мобільних календарів та збір даних про інструменти Android	27.02.2026 – 29.02.2026
2	Проведення порівняльного аналізу методів планування розкладу в мобільних додатках для освітян	30.02.2026 – 09.03.2026
3	Проведення порівняльного аналізу інструментів для оплат та записів учнів	10.03.2026– 18.03.2026
4	Підготовка матеріалів першого розділу роботи	19.03.2026 – 23.03.2026
5	Розроблення математичного моделювання планування	24.03.2026– 30.03.2026
6	Підготовка матеріалів другого розділу роботи	1.04.2026 – 04.04.2026
7	Підготовка матеріалів третього розділу роботи	05.04.2026– 13.05.2026
8	Розроблення програмного забезпечення мобільного додатка	14.05.2026 – 20.05.2026
9	Підготовка матеріалів четвертого розділу роботи	21.05.2026 – 26.05.2026
10	Оформлення пояснювальної записки	27.05.2026 – 05.06.2026

Дата видачі завдання:

«26» лютого 2026 р.

Студент

_____ / Д. А. Дорохін /

Керівник роботи

_____ / Д. В. Швець /

РЕФЕРАТ

МОБІЛЬНИЙ ЗАСТОСУНОК, ANDROID, ПЛАНУВАННЯ РОЗКЛАДУ, РЕПЕТИТОР, УЧНІ, ЗАНЯТТЯ, БАЗА ДАНИХ, REACT NATIVE.

Пояснювальна записка: 64 с., 3 табл., 21 рис., 3 дод., 11 джерел.

Метою кваліфікаційної роботи є розробка мобільного додатка для планування розкладу репетиторів на базі операційної системи Android для зручного керування розкладом занять, обліку учнів та відображення навчального процесу.

У роботі проведено аналіз існуючих мобільних додатків у сфері організації розкладу, визначено їхні переваги та недоліки з погляду зручності використання для звичайних користувачів, а також відповідності сучасним адаптивним і функціональним потребам викладачів.

Розроблено архітектуру програмного забезпечення, що включає в себе керування учнями, фінансовим балансом та календарем занять. Реалізовано збереження та обробку даних про учнів і заняття з використанням бази даних Firebase, де забезпечено можливість додавання, редагування та видалення учнів і їхніх занять, автоматичне списання вартості за уроки з балансу учня, а також відображення інформації про заплановані заняття за вибраний період.

Зроблено програмну реалізацію мобільного додатка мовою JavaScript та фреймворком React Native, який перетворює його на нативні компоненти для Android у середовищі Visual Studio Code. Проведено тестування програмного продукту, яке підтвердило правильність та стабільність додатка.

Результатом виконання кваліфікаційної роботи є працездатний мобільний додаток, який може бути використаний репетиторами для ефективного планування розкладу занять, контролю навчального процесу.

ABSTRACT

MOBILE APPLICATION, ANDROID, SCHEDULE PLANNING, TUTOR, STUDENTS, LESSONS, DATABASE, REACT NATIVE.

Thesis in 64 p., 3 tab., 21 fig., 3 app., 11 references.

The purpose of this qualification thesis is to develop a mobile application for tutor schedule planning based on the Android operating system, designed for convenient schedule management, student accounting, and visualization of the educational process.

The work includes an analysis of existing mobile applications in the field of schedule organization, identifying their advantages and disadvantages from the perspective of user experience for regular users, as well as their compliance with the modern adaptive and functional needs of educators.

The software architecture has been developed to include student management, financial balance tracking, and a lesson calendar. Data storage and processing for students and lessons are implemented using the Firebase database. This ensures the ability to add, edit, and delete students and their classes, provides automatic deduction of lesson costs from the student's balance, and displays information about scheduled lessons for a selected period.

The mobile application is programmatically implemented using the JavaScript language and the React Native framework, which transforms the code into native components for Android within the Visual Studio Code environment. Testing of the software product was conducted, confirming the correctness and stability of the application.

The result of this qualification thesis is a functional mobile application that can be used by tutors for effective lesson schedule planning and educational process control.

ЗМІСТ

ВСТУП	7
1 АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ	8
1.1 Критичний аналіз літературних джерел за темою	8
1.2 Актуальність теми кваліфікаційної роботи	13
1.3 Цілі та завдання кваліфікаційної роботи	17
1.3.1 Мета кваліфікаційної роботи	18
1.3.2 Завдання кваліфікаційної роботи	18
1.3.3 Очікувані результати	20
1.4 Завдання кваліфікаційної роботи	21
1.4.1 Етапи виконання завдань та розробки додатка	22
1.4.2 Практична цінність реалізованих завдань	23
2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ	25
2.1. Розробка та докладний опис функціональної схеми програми	25
2.2. Розробка та докладний опис алгоритму роботи програми	27
2.3. Алгоритм процесу програми з автоматизацією ручної праці	28
2.4. Алгоритм завантаження даних при старті.....	30
2.4.1. Алгоритм додавання/редагування учня	30
2.4.2. Алгоритм планування навчального процесу.....	31
2.4.3. Проведення заняття та фінансове списання	32
2.4.4. Аналіз діяльності.....	33
2.4.5. Налаштування	34
2.4.6. Реєстр функціональних вимог	35
2.4.7. Нефункціональні вимоги	36
2.5. Розробка інтерфейсу програми	36
3 РОЗРОБКА БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	40
3.1 Аналіз обраної середовища програмування	40
3.2 Розробка бази даних.....	42
3.3 Програмна реалізація основних функцій	43
3.4 Методика роботи користувача	44
3.5 Інструкція користувача мобільного додатка	46
3.5.1 Запуск та головний екран додатка	47
3.5.2 Робота зі списком учнів	48
3.5.3 Додавання та планування занять	49
3.5.4 Фінансовий облік та контроль балансу	51
3.5.5 Переваги використання додатка	53
ДОДАТОК А	57
ДОДАТОК Б	61
ДОДАТОК В	69

ВСТУП

З розвитком цифрових технологій мобільні додатки стали важливою складовою сучасного життя. Вони активно інтегруються в освітню сферу, бізнес, комунікаційні мережі та організацію особистого часу. Особливо важливу роль мобільні рішення відіграють у сфері приватного викладання та репетиторства, де існує потреба в ефективному управлінні навчальним та, і важніше, своїм процесом, включаючи планування занять, облік учнів та сам контроль цих дій.

Репетитори, які працюють з великою кількістю учнів, часто відчують труднощі з координацією розкладу, відстеженням дат і часу проведення уроків та, найважливіше, обліком отриманих оплат за уроки. Так, використання паперових записів або загальних календарів не завжди є оптимальним та надійним рішенням. Тому створення спеціального мобільного додатка для планування розкладу репетиторів має актуальну та практичну цінність.

З огляду на вищезазначене, виникає потреба у розробці спеціального мобільного рішення. Воно має на меті оптимізувати планування навчальних занять, забезпечити зберігання даних про учнів, ефективно управляти фінансами та надавати користувачу інтуїтивно зрозумілий доступ до розкладу. З огляду на це, можна стверджувати, що цей інструмент повністю відповідає сучасним трендам цифровізації освітнього процесу та сприяє підвищенню результативності індивідуальних навчальних програм.

1 АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Критичний аналіз літературних джерел за темою

Питання створення мобільних додатків для ефективного планування та управління даними є предметом досліджень як наукових, так і прикладних. Тому що особливо актуальними є теми оптимізації робочого часу, автоматизації складання розкладів та інтеграції мобільних технологій у освітній процес.

У праці з розробкою для платформи Android детально розкриваються ключові аспекти, такі як проектування архітектури додатків, застосування шаблонів проектування та взаємодія з віддаленими базами даних [1]. Автори наголошують на необхідності чіткого поділу функціональних блоків логіки програми, інтерфейсу користувача та доступу до даних для забезпечення зручного розвитку програмного продукту.

Значну увагу приділено використанню хмарних баз даних, зокрема Firebase Realtime Database [2]. Ці джерела є перевагою хмарних рішень, такими, як миттєва синхронізація даних, ефективне зберігання структурованої інформації та можливість доступу до неї з різних пристроїв.

Окремий напрямок досліджень охоплює організацію розкладів в освітніх системах. У такій роботі аналізував алгоритми планування, вирішуючи проблеми конфліктів у розкладі, та розглядав методи візуалізації календарних подій [3]. Однак більшість існуючих рішень розроблена для загальних освітніх установ або для групових занять і не враховує специфічні потреби індивідуального репетиторства.

Аналіз сучасних мобільних додатків для планування занять показує їхню в цілому обмежену функціональність. Зазвичай вони не включають інтегрований облік учнів, фінансових операцій чи персоналізованого розкладу [4]. Це підкреслює існування невирішених проблем на ринку і необхідність створення рішення, яке було б орієнтоване на потреби самозайнятих репетиторів.

Приватні викладачі, які проводять заняття віч-на-віч або у малих форматах, наприклад, з математики, іноземних мов, на НМТ або з інших шкільних дисциплін. Ці особи є самозайнятими професіоналами, що одночасно опікуються кількістю учнів від одного до десяти й більше, мають необмежений графік і не належать до великих навчальних закладів.

Які потреби потрібно задовольнити:

- а) Швидке та зручне планування/перенесення занять без конфліктів;
- б) Централізований облік учнів: контакти, ім'я, предмет, баланс та уроки;
- в) Автоматичний фінансовий контроль, баланс кожного учня, передоплата, автосписання вартості уроку після проведення, відстеження боржників;
- г) Миттєвий доступ до розкладу на ходу в мобільному форматі;
- д) Усунення ручних помилок, дублювання, забуті уроки, неточний облік платежів;
- е) Надійне хмарне зберігання даних.

Ключові переваги рішення:

- а) Усе в одному мобільному застосунку – це календар + учні + фінанси, і не потрібно перемикатися між Google Calendar, Notion та Sheets.
- б) Автоматизація фінансових операцій, автоматичне списання після відмітки уроку – знижують ризик суперечок і фінансових втрат.
- в) Простота та інтуїтивність інтерфейсу, адаптованого саме під репетитора без зайвих функцій.
- г) Використання Firebase Realtime Database – миттєва синхронізація, резервне копіювання, відновлення даних.
- д) Повна безкоштовна базова версія, що робить застосунок доступним для початківців-репетиторів.

Особливу увагу в сучасних дослідженнях і практичних оглядах приділено аналізу універсальних цифрових засобів, які часто застосовуються

репетиторами для організації власної діяльності, зокрема Google Календар, Notion, Excel-таблиця та Google Sheets. Зазначені засоби є популярними завдяки доступності, легкості та кросплатформенності, однак їх функціональні властивості мають обмеження у контексті індивідуального репетиторства.

Google Календар ефективно застосовується для візуалізації подій у часовому вимірі, дозволяє генерувати нагадування та переглядати розклад у різних форматах [5]. На рисунку 1.1 бачимо дизайн та відображення елементів, як-от виділених різдвяних або важливих днів для користувача. Однак цей інструмент не підтримує зберігання розширених відомостей про учнів, не має засобів фінансового обліку та не забезпечує логічного зв'язку між учнем, заняттям та оплатою. Як наслідок, репетитор змушений користуватися додатковими засобами для забезпечення фінансових та особистих даних, що ускладнює робочий процес.

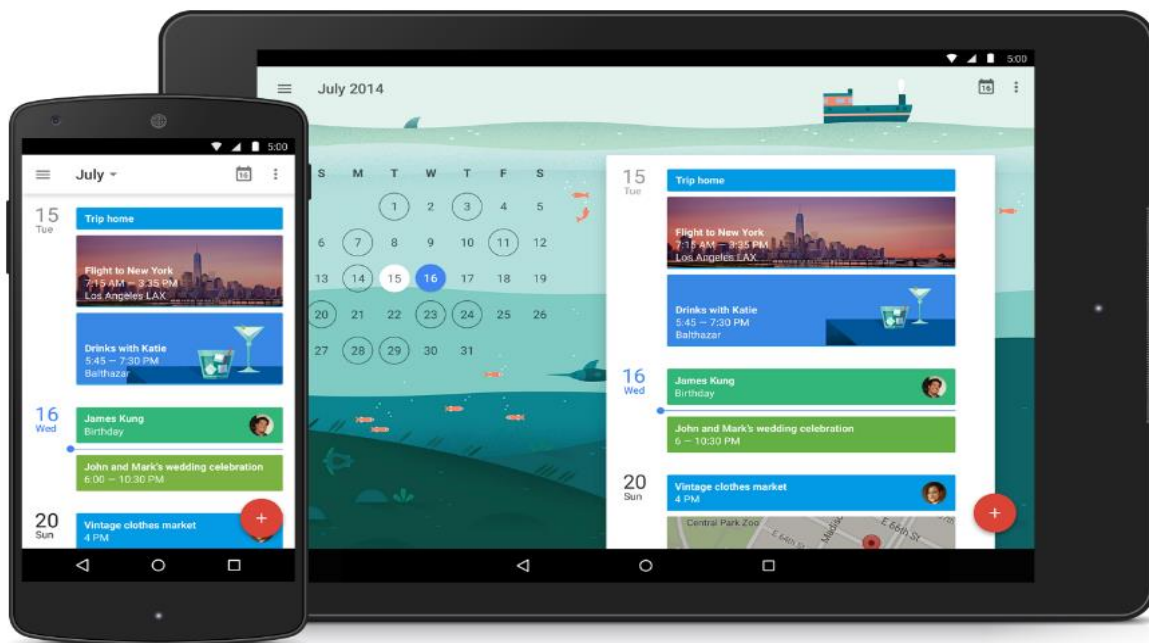


Рисунок 1.1 – Google Календар

Notion позиціонується як універсальна система для впорядкування

інформації, яка дозволяє створювати бази даних, таблиці та записи [6]. Він теж має календарні елементи, день та час, з вибором кольору та активним або виконаним завданням на рисунку 1.2. Хоча цей інструмент надає більшу гнучкість у структуруванні даних, його налаштування потребує значних часових витрат, а відсутність спеціалізованих сценаріїв для репетиторської діяльності знижує ефективність використання у щоденній праці. Крім того, Notion не сфокусований на автоматизації фінансових операцій, таких як списання коштів за проведення заняття.

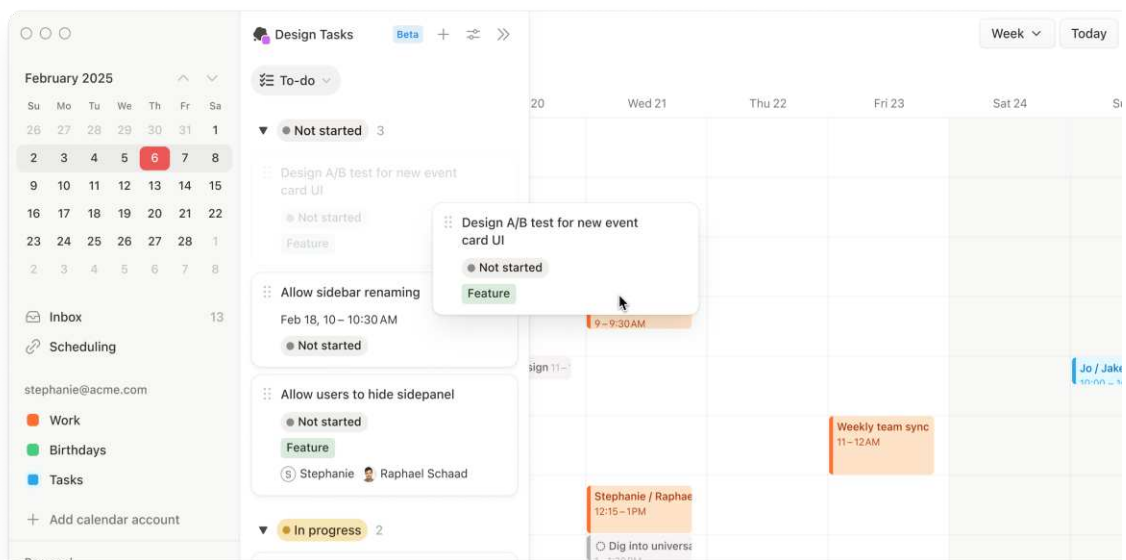


Рисунок 1.2 – Notion

Excel та Google Sheets часто вживаються для ведення переліків учнів, розкладів та грошового обліку. Їхньою перевагою є можливість створення формул та гнучке редагування відомостей [7]. Водночас такі рішення є не надто зручними для мобільного застосування, не надають візуалізації розкладу у вигляді календаря та не борються з людським фактором, зокрема, випадковим вилученням або подвоєнням записів.

Створено мобільний застосунок, що функціонує на React Native й використовує Firebase, котрий має на меті усунути рутинну роботу репетитора, заміщуючи незграбні таблиці на динамічний календар та механізм, що працює у режимі реального часу. Основною фішкою є

реалізована логіка автоматичного зняття оплати: після підтвердження заняття баланс учня оновлюється за декілька секунд, а фінансова звітність генерується без необхідності ручних обчислень.



Рисунок 1.3 – Excel та Google Sheets

Проведено аналіз реальних аналогів, які найчастіше використовують репетитори: Google Календар — універсальний інструмент та Notion — універсальна база даних.

Таблиця 1.1 – Аналіз існуючих аналогів

Критерій	Google Календар	Notion	Мій застосунок
Нативний моб. Застосунок Android	Так	Так але обмежена функціональність	Так оптимізована для репетитора
Календарний вид	Відмінний	Добрий але потрібно налаштувати	Відмінний з відображенням учнів у вибраний період

Критерій	Google Календар	Notion	Мій застосунок
Облік учнів	Ні	Так база даних	Так з пов'язуванням занять та фінансів
Фінансовий облік та баланс	Ні	Ні, Вручну	Автоматичне списування після вибраного дня
Автоматизація списання коштів	Ні	Ні	Так включено
Синхронізація в реальному часі	Google	Хмарна	Firebase Realtime Database
Спеціалізація саме під репетиторів	Ні	Ні	Так адаптовано
Ціна	Безкоштовно	Безкоштовно	Безкоштовно
Недоліки для репетиторів	Нема зв'язку – учень – урок – гроші	Складне налаштування немає автофінансів	Усі недоліки усунуто

Таким чином, розбір наявних цифрових засобів вказує на відсутність комплексного вирішення, яке б водночас поєднувало управління учнями, заняттями, фінансами та календарем у єдиному мобільному додатку, пристосованому до потреб окремого репетитора.

1.2 Актуальність теми кваліфікаційної роботи

Сучасний світ вимагає ефективного управління часом, особливо для

тих, хто працює на себе, наприклад, репетиторів. Зростання популярності індивідуальних занять робить репетиторство більш привабливим, але одночасно ставить перед фахівцями складні задачі. Вони одночасно обробляють багато учнів, працюють з різними темами, мають гнучкі графіки, стикаються з необхідністю перенесення або скасування уроків, ведуть фінансову звітність і контролюють відвідуваність. У таких умовах ручне планування стає малоефективним і може спричинити помилки.

Традиційні методи організації роботи репетиторів – блокноти, нотатки, прості календарі та месенджери – вже не відповідають сучасним вимогам. Ці інструменти не забезпечують збереження інформації про учнів та їхні заняття, не створюють чіткої структури “учень–уроки–фінанси” і ускладнюють швидкий доступ до потрібної інформації про розклад.

Неточність у розкладі може мати серйозні наслідки:

- а) Наприклад уроків це призводить до плутанини та втрати часу;
- б) Забуті уроки або неправильний час, підриває довіру учнів та батьків;
- в) Складність внесення змін, перенесення чи редагування уроків без помилок стає проблематичною.

Такі помилки негативно впливають на репутацію репетитора, можуть призвести до втрати клієнтів та прямих фінансових моментів, які б не хотілося мати.

Фінансовий контроль також є важливим моментом. Багато репетиторів працюють за передоплатою або ведуть баланс для кожного учня. Ситуації, коли оплата надходить частинами, потребують автоматичного списання коштів за урок або швидкого визначення боржників чи тих, хто переплатив. Виникають постійно. Ручний фінансовий облік майже завжди призводить до неточностей і суперечок, адже легко забути списати оплату або зробити помилку в сумі. Тому існує нагальна потреба у розробці додатка, який не тільки зберігатиме інформацію про заняття, а й автоматизуватиме фінансові процеси, наприклад, списання коштів за проведені уроки.

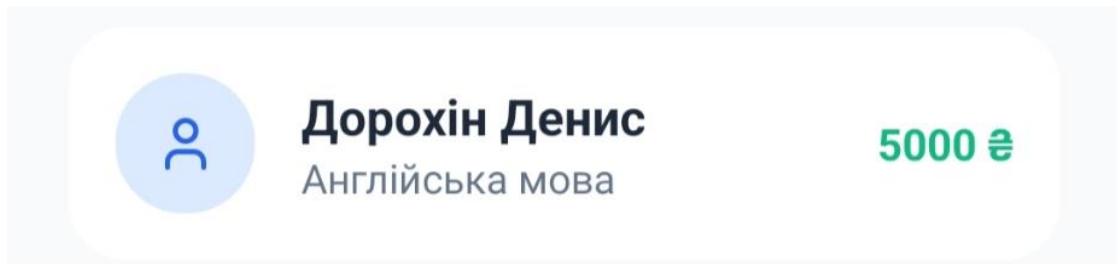


Рисунок 1.2 – Вчасна оплата

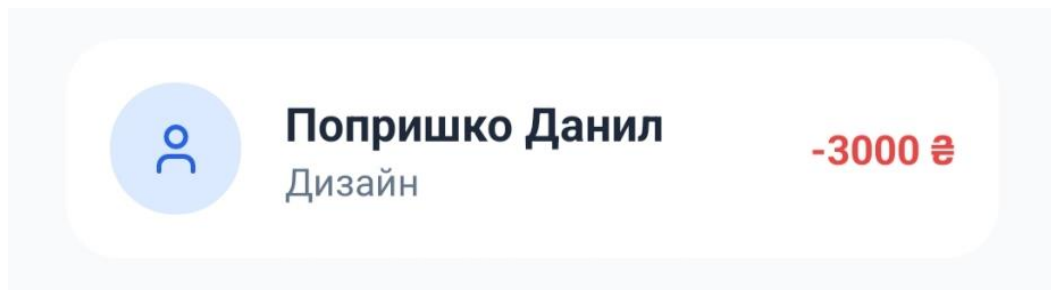


Рисунок 1.3 – Боржник

Мобільний формат є особливо актуальним, адже смартфон завжди під рукою.

Це дозволяє:

- а) Швидко переглядати розклад будь-де;
- б) Легко додавати або переносити уроки;
- в) Перевіряти список учнів;
- г) Миттєво бачити, хто має заняття сьогодні;
- д) Контролювати фінансовий баланс без додаткових зусиль.

Розвиток хмарних технологій та баз даних у реальному часі, таких як Firebase Realtime Database, робить розробку мобільних додатків ще більш привабливою. Ці технології забезпечують надійне зберігання даних, швидку синхронізацію між пристроями та можливість відновлення інформації у разі

втрати або перевстановлення додатка, що є значною перевагою порівняно з локальним зберіганням даних [12].

Наслідки ручного ведення обліку в діяльності репетитора проявляються не лише у зменшенні зручності роботи, а й у появі системних помилок, які з часом накопичуються. Брак автоматизованого інструменту спричиняє неузгодженість даних між розкладом, фінансовим балансом та фактично проведеними уроками. У підпуску репетитор витрачає велику кількість часу на звірку інформації та виправлення помилок замість безпосередньої освітньої діяльності.



Рисунок 1.4 – Проблеми та рішення репетиторської діяльності

Фінансові помилки є одними з найчастіших труднощів при ручному веденні обліку. Тому неправильно врахована оплата, некоректний залишок або витрати можуть спричинити суперечливі моменти з учнями та їхніми батьками [8]. Такі обставини пригнічують професійний імідж викладача та формують додатковий емоційний тягар та відгуки.

Окремою проблемою є пропущені або переплутані заняття. Відсутність єдиного календаря з актуальною інформацією може призвести до того, що урок не буде проведений вчасно або відбудеться накладання кількох занять. Подібні помилки впливають на якість наданих освітніх послуг.

Важливість створення спеціалізованого програмного забезпечення також викликана зростанням потреби у приватних освітніх послугах. Все більше викладачів віддають перевагу індивідуальній формі праці, що вимагає дієвих засобів самоорганізації. За таких обставин мобільний застосунок, сфокусований саме на репетиторах, стає не просто додатковим інструментом, а визначною складовою фахової діяльності.

1.3 Цілі та завдання кваліфікаційної роботи

Сучасні мобільні технології проникають у всі сфери життя, включаючи освіту. Для репетиторів мобільний додаток може стати незамінним помічником, що спрощує організацію навчального процесу. Він дозволяє відмовитися від паперових записів та численних сервісів, об'єднуючи всі необхідні функції в одному місці. Це особливо важливо, адже робота репетитора вимагає не лише проведення занять, а й ефективного планування часу, спілкування з учнями, контролю фінансів та відстеження балансів.

Традиційне планування розкладу часто супроводжується труднощами:

- а) Перенесення уроків;
- б) Дублювання записів;
- в) Забуті заняття;
- г) Складний пошук інформації про майбутні уроки.

Ручне ведення фінансового обліку, особливо при великій кількості учнів, є трудним і схильним до помилок. Тому створення мобільного додатка, який автоматизує ці процеси та забезпечує їхню прозорість, є надзвичайно актуальним.

Отже, виникає потреба у формуванні єдиного програмного рішення, яке надасть репетитору змогу централізовано керувати всіма аспектами своєї професійної діяльності. Мобільний застосунок повинен об'єднувати функції планування графіка та грошових операцій, що сприяє зниженню тягаря на користувача та збільшенню результативності його праці.

У використанні мобільного програмного забезпечення у сфері

індивідуального навчання сприяють зростанню організованості навчального процесу, зменшенню числа помилок та оптимізації втрат часу. Особливо важливою є можливість швидкого доступу до історії проведених уроків, що неможливо гарантувати при застосуванні застарілих засобів планування.

1.3.1 Мета кваліфікаційної роботи

Основна мета цієї кваліфікаційної роботи – розробка мобільного застосунку для Android, який стане незамінним помічником для репетиторів у плануванні та організації їхньої роботи.

Додаток надасть можливість:

- а) Зберігати повну інформацію про кожного учня;
- б) Зручно керувати всіма уроками від створення до видалення;
- в) Вести точний фінансовий облік;
- г) Відстежуючи баланс коштів кожного учня;
- д) Автоматизувати процес списання вартості проведених занять;
- е) Відображати всі заплановані заняття у форматі календаря.

Розробка даного додатка буде відрізнятися простотою використання та надійністю завдяки використанню бази даних для безпечного зберігання всіх даних.

1.3.2 Завдання кваліфікаційної роботи

Щоб досягти поставленої мети, важаю що потрібно виконати наступні кроки:

- а) Вивчити сферу діяльності репетитора та зрозуміти, як працюють репетитори, які завдання виникають при складанні розкладу і чому виникають помилки при ручному веденні записів.
- б) Проаналізувати існуючі рішення, щоб дослідити мобільні додатки та методи планування, які вже використовують репетитори (календарі, таблиці, нотатки, спеціалізовані програми), які ми всі знаємо. Виявити їхні переваги та недоліки для роботи репетитора.

в) Визначити вимоги до нового додатка:

1. Функціонал, що додаток повинен вміти облік учнів, занять, фінансів, календар;
2. Інтерфейс як він мав виглядати зручно, зрозуміло, без зайвих дій;
3. Надійність – це забезпечення безпечного зберігання даних та стабільної роботи.

г) Спроекувати структуру додатка: визначити, як будуть взаємодіяти різні частини програми, як зберігатимуться дані та як працюватимуть основні функції.

д) Розробити базу даних: створити структуру для зберігання інформації про учнів та уроки, забезпечивши правильне додавання, оновлення, видалення та відображення даних, а також цілісність інформації.

е) Розробити мобільний додаток для Android: створити додаток мовою React Native у середовищі Visual Studio Code, використовуючи сучасні інструменти.

ж) Створити зручний інтерфейс користувача:

1. Головний екран з календарем;
2. Перегляд уроків на обраний день;
3. Вікна для додавання занять;
4. Редагування часу та статусу уроку;
5. Зміна теми застосунку.

з) Автоматизувати фінансові операціях:

1. Вести баланс для кожного учня;
2. Дозволяти змінювати баланс, поповнення або повернути власнику;
3. Автоматично списувати вартість уроку після змінення статусу у проведеному занятті.

и) Аналітика:

1. Відображення статистики: за місяць та всього;
2. Рейтинг за доходом із можливістю бачити історію.

к) Протестувати додаток:

1. Перевірити роботу календаря та відображення уроків;
2. Протестувати додавання, редагування та видалення даних;
3. Перевірити збереження даних у Firebase;
4. Переконалися у стабільності роботи та відсутності дублювання уроків.

Здійснення окреслених завдань дає змогу комплексно підійти до створення мобільного додатка та брати до уваги як технічні, так і практичні аспекти його застосування. Послідовне виконання кожного кроку гарантує логічність зведення програмного добутку та його відповідність поставленій цілі.

Також важливу роль відіграє етап розбору вже наявних рішень, адже він дозволяє оминати загальні прорахунки та скористатися найкращими напрямками при розробці додатка. Проектування структури та бази даних є основою стабільної роботи системи та запобігає появі проблем із повторенням чи втратою даних.

Тому тестування розробки є завершальним, але не менш важливим ніж інші етапи, що дозволяє перевірити роботу всіх елементів та переконалися у стабільності програмного рішення.

1.3.3 Очікувані результати

У результаті виконання кваліфікаційної роботи планую отримати мобільний застосунок який:

- а) Забезпечує зручне планування уроків репетитора;
- б) Надає можливості вести базу учнів із ключовими даними;
- в) Дозволяє швидко переглядати хто має заняття у певний день;
- г) Автоматизує облік фінансів;
- д) Використовує базу даних для надійного збереження інформації.

Застосування створеного застосунку дозволить відчутно збільшити дієвість організації навчального процесу та скоротити затрати часу на буденні операції.

Завдяки автоматизації головних процесів зменшується вплив людського чинника, який часто є винуватцем помилок у розкладі або фінансовому обліку. Ясна структура даних та використання бази даних гарантують цілісність інформації та її сучасність.

Створений застосунок має потенціал до подальшого розширення та масштабування, зокрема шляхом додавання нових функцій, аналітики прибутків або синхронізації з іншими сервісами. Це робить програмний продукт гнучким та придатним для вжитку в умовах зростання кількості учнів або зміни формату роботи репетитора.

Отримані наслідки засвідчують доцільність використання мобільних технологій у сфері індивідуального навчання та виявляють практичну важливість створеного програмного забезпечення.

1.4 Завдання кваліфікаційної роботи

Всі етапи розробки мобільного додатка для планування розкладу. Це включає аналіз, проектування, створення та тестування. Оскільки такі додатки є затребуваними в освітній сфері, моя тема – це створити не просто календар, а ефективний інструмент для репетиторів.

Завдання роботи враховують як практичні потреби репетиторів, так і сучасні стандарти розробки Android-додатків, де є зручність, продуктивність, безпека та стабільність. Кожен етап дозволяє перейти від дослідження до готового продукту.

Такі завдання як:

- а) Вивчення специфіки роботи репетиторів;
- б) Визначення вимог до функціоналу та інтерфейсу;
- в) Проектування бази даних учні, заняття та фінанси ;
- г) Реалізацію зберігання даних та синхронізації через Firebase;

- д) Розробку інтуїтивно зрозумілого інтерфейсу для управління учнями та заняттями;
- е) Впровадження функцій додавання, редагування та видалення занять та учнів;
- ж) Створення системи фінансового обліку де є баланс, автоматичні списання та від'ємний баланс;
- з) Проведення тестування для забезпечення стабільності.

Таким чином, робота охоплює повний цикл розробки, від аналізу до функціонального продукту. Варто підкреслити, що поставлені завдання кваліфікаційної роботи спрямовані не лише на технічне втілення мобільного додатка, а й на формування зручного засобу, пристосованого до дійсних умов праці репетитора. Під час виконання роботи беруться до уваги риси індивідуального викладання, нерівномірне завантаження, потребу швидкого внесення правок до графіка та неперервний нагляд за фінансовими операціями.

Особливу увагу приділено скороченню обсягу рутинних кроків, які викладач мусить здійснювати щодня. Автоматизація обліку уроків та платежів дає змогу зменшити ймовірність хиб, пов'язаних із людським чинником, та збільшити загальну дієвість налагодження навчального процесу.

Отже, окреслені завдання формують логічний та вибудований ґрунт для переходу від аналізу сфери застосування до впровадження цілісного програмного виробу, здатного розв'язувати практичні питання репетитора у щоденній діяльності.

1.4.1 Етапи виконання завдань та розробки додатка

Процес розробки мобільного додатка для репетиторів починається з глибокого аналізу їхньої поточної роботи. Досліджував, як саме вони керують розкладом, яку інформацію про учнів та заняття зберігають і з якими труднощами стикаються. Часто це ручне ведення записів у блокнотах чи

простих таблицях, що призводить до втрати даних, складнощів з пошуком інформації, відсутності автоматичного обліку оплат та помилок при плануванні через велике навантаження.

На основі цього аналізу формую чіткий перелік вимог до майбутнього додатка. Ключові моменти, найчастіші дії користувача та бажаний вигляд інтерфейсу: приділив особливу увагу зручності та швидкості взаємодії, де репетитор повинен мати змогу вносити зміни мінімальною кількістю натискань. Далі відбувається проектування архітектури додатка. Це включає визначення основних сутностей, їхніх взаємозв'язків та логіки відображення даних. Одночасно розробляється механізм роботи календаря та списків занять, що дозволить користувачеві легко переглядати заплановані уроки за різними датами.

Наступним етапом є вибір технологій та інструментів розробки. Для створення мобільного додатка обрано платформу Android, що є однією з найпоширеніших мобільних операційних систем. Використання мови програмування JavaScript з фреймворком React Native дозволяє забезпечити надійність коду, зручність підтримки та відповідність сучасним стандартам Android-розробки. Паралельно з реалізацією функціональної логіки виконується розробка користувацького інтерфейсу. Інтерфейс проектується з урахуванням принципів зручності та інтуїтивної зрозумілості, оскільки додаток орієнтований на щоденне використання. Основна увага приділяється швидкому доступу до розкладу, списку учнів та фінансової інформації [13].

Завершальним етапом є тестування розробленого додатка. Проводиться перевірка коректності збереження даних, відображення занять у календарі, роботи фінансового обліку та стабільності застосунку при тривалому використанні. Це дозволяє виявити та усунути можливі недоліки ще до завершення розробки.

1.4.2 Практична цінність реалізованих завдань

Цей мобільний додаток приносить реальні переваги репетиторам та

викладачам, які працюють індивідуально. Він допомагає впорядкувати всю інформацію про учнів: від контактів та предметів до балансу оплат та історії занять. Особливо зручним є планування уроків через календар. Викладач одразу бачить розклад на день, може легко додавати чи редагувати заняття, а всі зміни автоматично зберігаються. Додаток також спрощує фінансовий облік: автоматичне списання вартості уроків з балансу мінімізує помилки та робить оплати прозорішими.

Отже, головна користь цього розроблення – його практичне застосування у щоденній роботі. Додаток економить час, зменшує організаційні помилки та допомагає тримати в порядку розклад і фінанси, що особливо цінно при роботі з багатьма учнями.

Практична вартість втіленого програмного продукту полягає також у його універсальності. Додаток може застосовуватися репетиторами з різних дисциплін незалежно від формату занять, числа учнів чи тривалості уроків. Гнучка структура зберігання відомостей дозволяє легко підлаштовувати застосунок до змін у робочому процесі користувача.

Використання хмарного сховища забезпечує збереження інформації та її своєчасність навіть у разі зміни апарату або повторного встановлення додатка. Це збільшує стійкість програмного забезпечення та довіру споживачів до результатів його функціонування.

Ключова перевага впровадженої системи полягає в автоматизованому генеруванні аналітичних звітів, завдяки чому наставник може вмиль проаналізувати результативність своєї роботи за обраний відрізок часу. Завдяки застосуванню фреймворку React Native, застосунок вирізняється чутливою реакцією інтерфейсу й надійною роботою вбудованих елементів системи Android. Отже, створений помічник перетворює звичну рутину на високотехнологічний процес, даруючи викладачеві змогу цілковито присвятити себе підвищенню рівня навчання.

2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1. Розробка та докладний опис функціональної схеми програми

Цей мобільний застосунок працює як персональний асистент для репетиторської діяльності, що поєднує функції ведення бази учнів, керування розкладом занять та обліку фінансових операцій як балансу учнів, так і оплат. Основна ідея полягає у тому, що користувач має змогу швидко додавати учнів, призначати заняття на календарі на якусь дату та одразу бачити, хто саме має заняття у вибраний день.

З технічної точки зору, додаток побудований за принципом розділення обов'язків. Це те, що бачимо на екрані: логіка управління даними та взаємодія з базою даних, які працюють незалежно одна від одної.

Тому функціонально додаток працює як набір модулів, де кожен з них виконує окремі операції, але всі вони пов'язані між собою:

- а) Модуль керування учнями: створення, редагування, видалення учнів, предмету, балансу та вартості уроку;
- б) Модуль календаря: перегляд місяця, вибір дати, візуальне позначення днів із заняття;
- в) Модуль уроків: створення уроку який може бути разовим, редагування існуючого, видалення, а також відображення уроків у списку;
- г) Фінансовий модуль: де автоматичне списання вартості уроку з балансу учня, можливість балансу в мінусову як боржник з виділенням кольору;
- д) Модуль збереження даних: синхронізація з Firebase Realtime Database з збереженням всіх записів.

Таким чином, система спрощує типові задачі репетитора: замість ручних нотаток у блокноті або в соціальних мережах користувач отримує єдину платформу для організації навчального процесу.

Функціональна схема програми демонструє логіку взаємодії між користувачем та головними компонентами системи. Усі операції репетитора, як-от додання учнів, створення занять чи зміна фінансових відомостей, запускаються через інтерфейс користувача та надсилаються до логічного рівня застосунку, де провадиться перевірка правильності даних та виконання відповідних дій.

Ключову роль у функціональній структурі відіграє календарний елемент, котрий є центральною складовою навігації. Саме завдяки календарю користувач здобуває загальне уявлення про власний графік, бачить зайняті й вільні дні, а також може оперативно переходити до детального перегляду занять на обрану дату. Такий метод суттєво підвищує зрозумілість та комфортність взаємодії з розкладом.

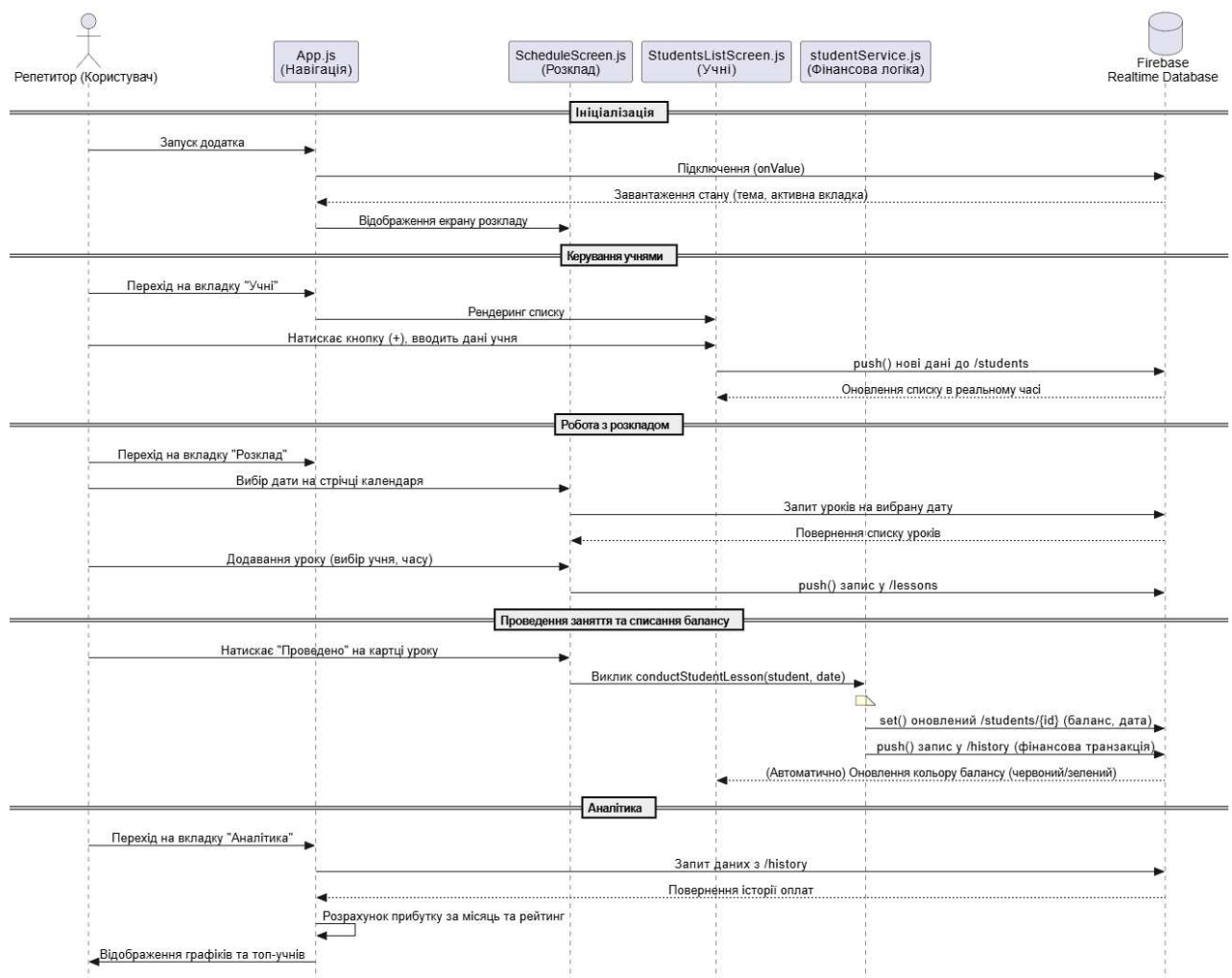


Рисунок 2.1 – Сценарій взаємодії додатка

Фінансові транзакції інтегровані у загальну логіку роботи додатка. Після створення заняття система автоматично розпочинає процес списання вартості уроку з балансу відповідного учня. Це усуває потребу ручного контролю оплат і зменшує вірогідність помилок. У разі недостатнього балансу система дозволяє фіксувати борг, що надає репетитору повну картину фінансового стану кожного учня.

2.2. Розробка та докладний опис алгоритму роботи програми

Основу роботи програми складає календар, який служить центральним інструментом для планування. При старті додаток завантажує дані про учнів та уроки з бази, готуючи їх для відображення. Коли користувач обирає дату в календарі, програма фільтрує заняття, показуючи лише тих учнів, у яких заплановано уроки на цей день.

При створенні уроку також активується фінансова частина: для платних занять сума списується з балансу учня. Якщо коштів бракує, баланс стає від'ємним, що допомагає репетитору відстежувати заборгованість.

Окрім основних дій із перегляду розкладу, алгоритм функціонування програми передбачає сталу взаємодію між інтерфейсом користувача та сховищем даних. Кожна дія користувача, як-от додавання нового учня, формування або зміна заняття, миттєво опрацьовується програмною логікою та фіксується у віддаленому сховищі даних Firebase. Це дає змогу гарантувати свіжість даних та запобігти втраті відомостей у разі перезавантаження додатка чи зміни пристрою.

Алгоритм роботи програми побудований таким чином, щоб зменшити потребу у ручних кроках з боку користувача. Наприклад, після створення заняття система автоматично звіряє наявність коштів на рахунок відповідного учня та проводить фінансову операцію без потреби додаткового схвалення. Усі розрахунки виконуються у тлі, що збільшує швидкість додатка та знижує ймовірність хиб, пов'язаних із людським чинником.

Важливою рисою алгоритму є опрацювання періодичних занять. Під

час вибору проміжку дат та днів тижня програма утворює набір окремих занять, які логічно сполучені між собою. Це дозволяє репетитору миттєво створювати розклад на тиждень чи місяць, не витрачаючи час на власноручне додавання кожної лекції окремо.

Також алгоритм містить перевірку правильності внесених даних. Програма стежить за точністю дат, часу початку заняття та його тривалості, унеможлиблюючи накладання уроків чи формування логічно некоректного розкладу.

2.3. Алгоритм процесу програми з автоматизацією ручної праці

Основною метою розробленого алгоритму є мінімізація рутинних дій репетитора та виключення помилок. Програма перетворює лінійний процес ведення записів у блокноті на інтерактивну систему, де кожна дія користувача викликає ланцюжок автоматичних підрахунків.

Автоматизація у цій системі організована шляхом розмежування функціональних обов'язків між викладачем та мобільним додатком. Запропонований алгоритм охоплює увесь ланцюжок взаємодії – від подачі заявки на урок і аж до фінансової звітності. Важливою складовою є інтелектуальна маніпуляція вхідною інформацією, яка включає автоматизовану верифікацію розкладу на предмет збігів у часі. Це запобігає ситуаціям накладання, як-от призначенню двох різних учнів на одну й ту саму годину, що є вагомою перевагою порівняно зі стандартними паперовими методами.

Коли планування проходить успішно, система автоматично прив'язує заплановану подію в календарі до персональної картки студента. Найбільш суттєвий елемент автоматизації проявляється у фінальній точці – після закінчення заняття. Замість того, щоб власноруч перераховувати залишки коштів у відомостях, репетиторові достатньо лише змінити статус заняття на «Завершено».

Це ініціює програмний тригер, який негайно виконує такі дії:

- а) списання фіксованої вартості заняття з електронного рахунку учня;
- б) актуалізацію поточного балансу безпосередньо в момент здійснення операції;
- в) моніторинг фінансового стану для виявлення потенційних боргів.

Такий підхід дає змогу педагогу цілком сфокусуватися на викладанні, передаючи складні математичні розрахунки та управління графіками програмному забезпеченню. Рисунок 2.3 наводить візуальне представлення алгоритму, демонструє логіку послідовності дій та точок прийняття рішень.

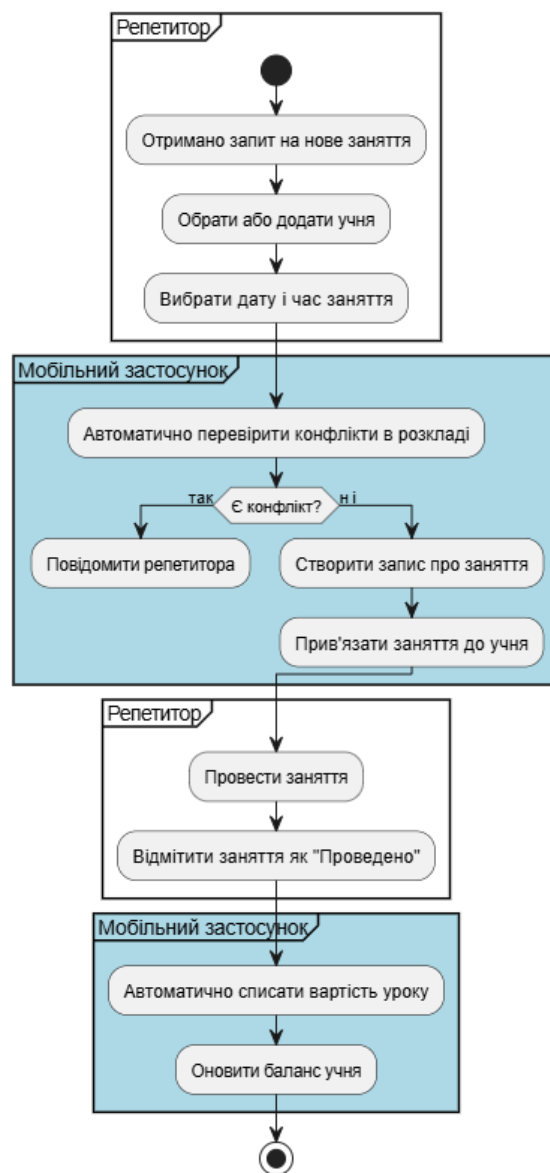


Рисунок 2.4 – Діаграма процесів автоматизації

2.4. Алгоритм завантаження даних при старті

При запуску програми важливо якнайкраще відобразити користувачеві базовий інтерфейс, одночасно виконуючи завантаження даних у фоні. Для цього використовується наступний підхід:

- а) Користувач запускає застосунок;
- б) Потім система завантажує основний контейнер (App.js), ініціалізує зв'язок з Firebase Realtime Database;
- в) Відкривається вкладка за замовчуванням – «Розклад» (ScheduleScreen.js). Користувач бачить календарну стрічку та список занять на сьогодні.



Рисунок 2.5 – Алгоритм запуску та завантаження даних

2.4.1. Алгоритм додавання/редагування учня

Оптимізація роботи з даними учнів досягається шляхом застосування єдиного алгоритму для операцій додавання та редагування.

Цей універсальний процес складається з:

- а) Користувач натискає на іконку «Учні» у нижній панелі навігації та

- натискає кнопку (+);
- б) Відкривається модальне вікно для введення імені, предмету та початкового балансу;
 - в) Після підтвердження дані відправляються у Firebase де в системі оновлюється список учнів у реальному часі через слухач onValue;
 - г) Сценарій перегляду боржників: Користувач гортає список; учні з від'ємним балансом підсвічуються червоним кольором в StudentsListScreen.js.

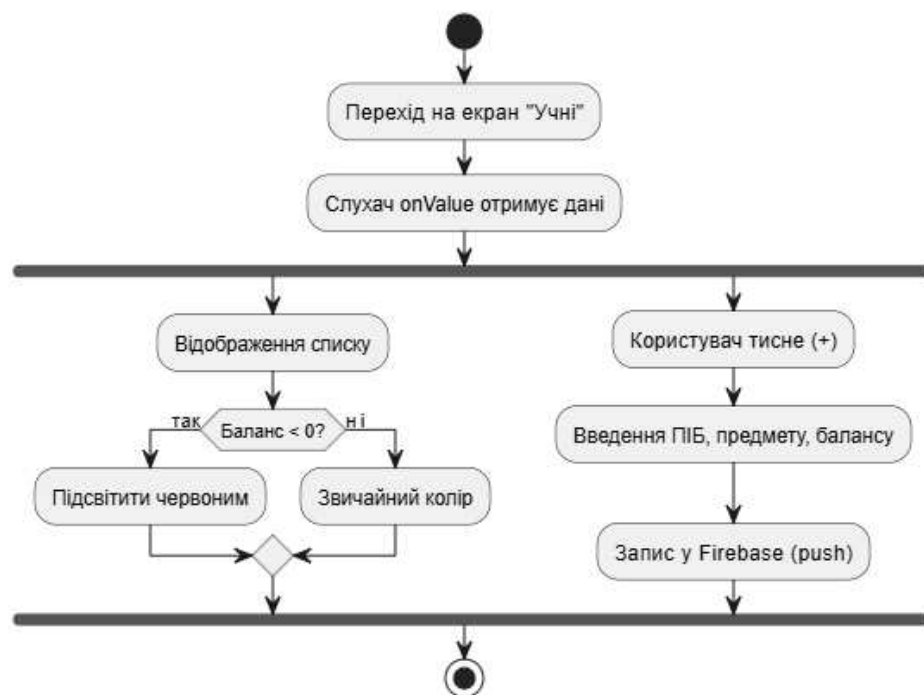


Рисунок 2.5.1. – Алгоритм керування учнями

2.4.2. Алгоритм планування навчального процесу

Основна сутність програми – це заняття, яке характеризується датою та часом проведення, прив'язкою до конкретного учня, тривалістю та вартістю. Після збереження інформації про заняття система починає керуватися бізнес-правилом, тобто може ініціювати автоматичну зміну балансу учня.

- а) Повернення на вкладку «Розклад»;
- б) Користувач вибирає дату на календарній стрічці;
- в) Натискає кнопку додавання уроку;

- г) Вибирає учня зі списку що підтягується з бази та вказує час;
- д) Система вже записує об'єкт уроку в гілку lessons у Firebase. На календарі з'являється маркер, що свідчить про наявність занять у цей день.



Рисунок 2.5.2. – Алгоритм створення уроку

2.4.3. Проведення заняття та фінансове списання

Дії для взаємодії з проведенням та фінансовими функціями:

- а) Користувач знаходить потрібний урок у списку на обраний день;
- б) Користувач натискає на урок або спеціальну кнопку виконання;
- в) Викликається функція з `studentService.js`;
- г) Система віднімає встановлену вартість уроку від поточного балансу

учня в базі;

- д) Оновлює дату останнього заняття (lastLesson);
- е) Додає запис про успішну операцію в гілку history для аналітики;
- ж) Урок позначається як «Проведено», баланс учня миттєво оновлюється на всіх екранах.



Рисунок 2.5.3. – Алгоритм фінансового списання

2.4.4. Аналіз діяльності

Аналітика вже існуючих операції за опрацьовані уроки:

- а) Натискання на вкладку «Аналітика» (AnalyticsScreen.js);
- б) Система зчитує дані з history;
- в) Групує дохід за поточний місяць та за весь час;
- г) Сортує учнів за кількістю принесеного прибутку (рейтинг);
- д) Користувач бачить фінансовий звіт та динаміку своєї роботи.



Рисунок 2.5.4. – Алгоритм генерації аналітики

2.4.5. Налаштування

Зміна теми застосунку для комфортного використання:

- Користувач натискає кнопку перемикавання теми (Сонце/Місяць);
- Система змінює стан `isDarkMode` у `App.js`;
- Глобальний об'єкт `theme` змінює кольори фону, тексту, картки у всіх модулях одночасно.

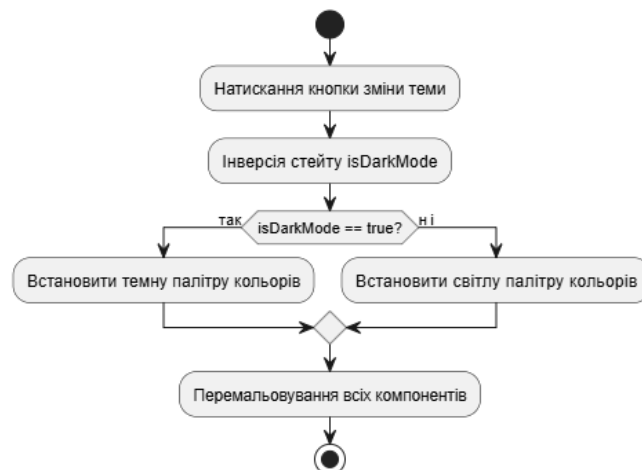


Рисунок 2.5.5. – Алгоритм зміни теми

2.4.6. Реєстр функціональних вимог

Таблиця 2.5.6 – Вимоги до застосунку

ID Вимоги	Опис вимоги	Критерії приймання
US.01	Як репетитор, я хочу додавати нових учнів, щоб вести облік їхніх контактів та предметів.	1. Наявність форми з полями: ПІБ, предмет, ціна, баланс. 2. Дані успішно зберігаються у Firebase-вузлі students/.
US.02	Як репетитор, я хочу планувати заняття в календарі, щоб бачити свій графік на день.	1. Можливість вибору дати на горизонтальній стрічці. 2. Відображення списку уроків саме для обраної дати.
US.03	Як репетитор, я хочу відмічати урок як проведений, щоб автоматично списувати оплату.	1. Натискання кнопки "Проведено" викликає функцію списання. 2. Баланс учня зменшується.
US.04	Як репетитор, я хочу бачити фінансову аналітику, щоб розуміти свій заробіток.	1. Розрахунок загального доходу та доходу за поточний місяць. 2. Відображення рейтингу учнів за сумою оплат.
US.05	Як репетитор, я хочу змінювати тему інтерфейсу, щоб комфортно працювати ввечері.	1. Перемикач Dark/Light Mode змінює кольори всього додатка. 2. Стан теми зберігається під час навігації.

2.4.7. Нефункціональні вимоги

Якість роботи цієї системи:

а) Продуктивність:

1. Де вимогою є час синхронізації даних з хмарною базою при стабільному інтернеті;
2. Метрика оновлення інтерфейсу після зміни балансу в базі даних Firebase має відбуватися не довше ніж за півтори секунди.

б) Надійність:

1. Цілісність даних при проведенні транзакцій тобто списання коштів;
2. Успішні фінансові операції, які складають 100%, якщо є зв'язок з сервером.

в) Зручність використання:

1. Зрозумілість інтерфейсу для нових користувачів;
2. Додавання нового учня або уроку має займати не більше 4 кліків: екран – кнопка (+) – введення – зберегти.

г) Адаптивність:

1. Коректне відображення на різних пристроях;
2. Інтерфейс має коректно масштабуватися на екранах стандартних Android-смартфонів.

2.5. Розробка інтерфейсу програми

Щодо інтерфейсу, не так вразить дизайнерське рішення, а максимально зручним та швидким у використанні. Репетитор, відкривши додаток, повинен бачити свої заняття та розуміти, хто наступний у розкладі.

Зробивши центральний екран головним в якому все що потрібно є та реалізував концепцію головного екрану, який об'єднує найважливіші функції:

а) Календар: для зручного вибору дати;

- б) Список учнів: для швидкого доступу до інформації про кожного;
- в) Кнопки швидких дій: дозволяють миттєво додати нового учня, запланувати заняття або змінити час уроку.

Таким підходом мінімізував кількість переходів по меню, тому що репетитори часто працюють в умовах обмеженого часу; можливість виконати потрібну дію за 2–4 натискання є важливою

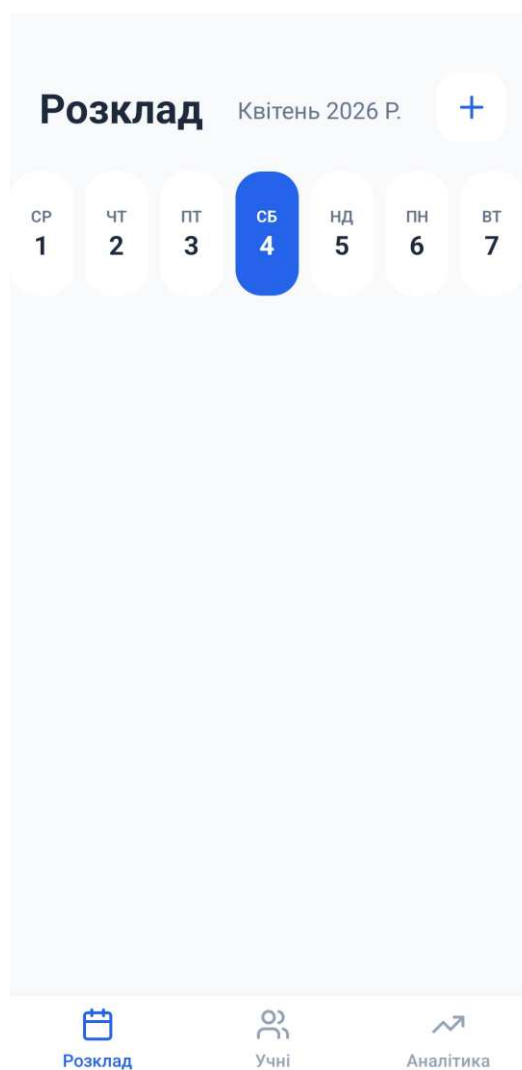


Рисунок 2.6 – Головна сторінка додатка

Рисунок 2.6 відображає головний екран, який буде зустрічати користувача. Відразу буде вибраний поточний день, в якому вказано число, день неділі та місяць, з можливістю додати урок.

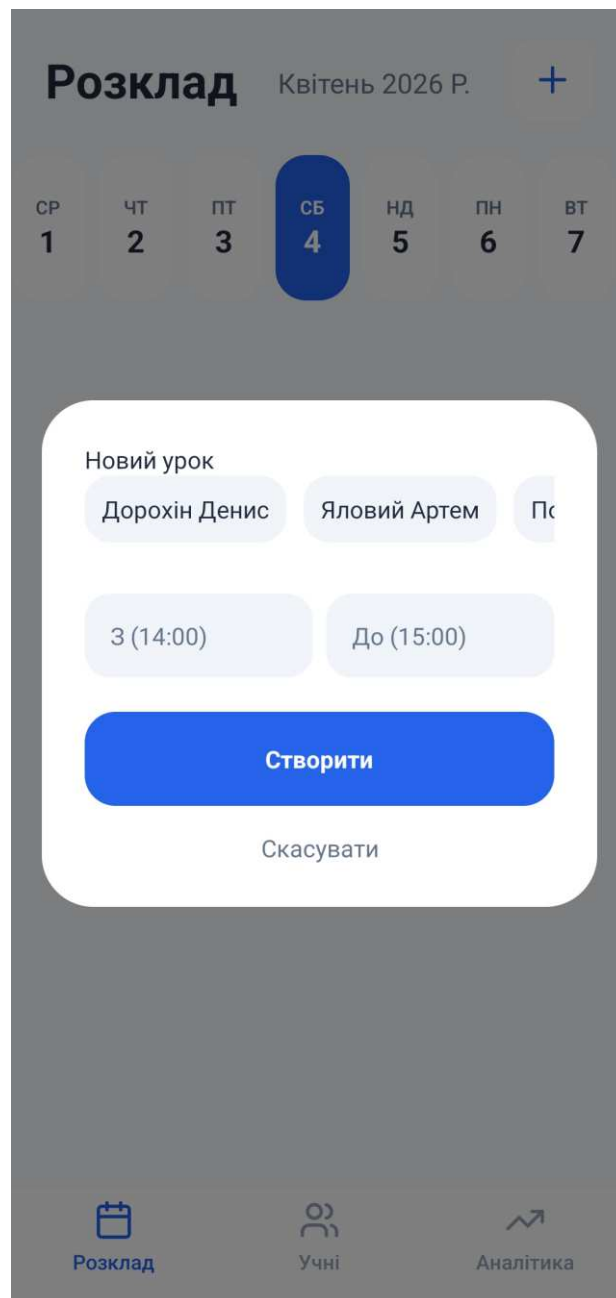


Рисунок 2.7 – Сторінка додавання студентам уроки

На рисунку 2.7 після спроби додати студента відображається спливаюче вікно з вибором учня з бази даних та на який час — початок та кінець уроку.

Із ключових елементів інтерфейсу:

- а) Колендарний компонент: для вибору конкретного дня чи місяця, а також для зручної навігації між датами;
- б) Список учнів та занять: реалізовано RecyclerView, який потрібен для списків в Android, що забезпечує ефективне відображення даних.

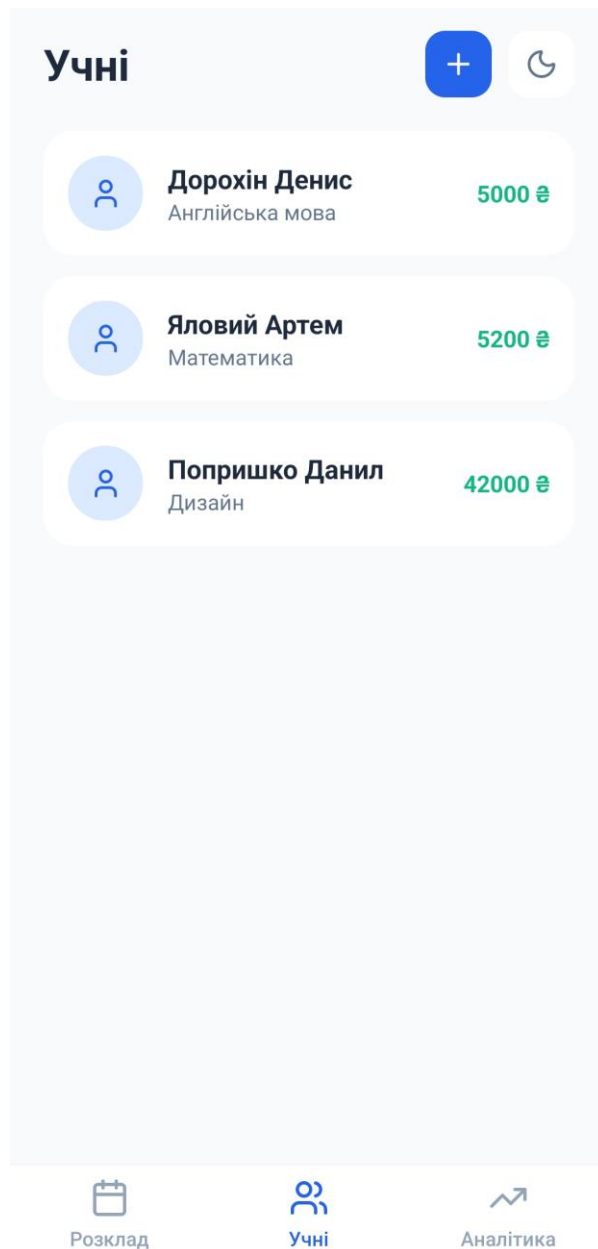


Рисунок 2.8 – Область даних студента

Як дані відображаються в інтерфейсі:

- Розділення відповідальності: інтерфейс (UI) не взаємодіє безпосередньо з базою даних (Firebase);
- Централізоване оновлення: UI підписаний на зміни стану в ViewModel. Коли дані оновлюються, ViewModel повідомляє про це адаптер списку, який, у свою чергу, перемальовує відповідні елементи.

3 РОЗРОБКА БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз обраної середи програмування

Розробив мобільний додаток для планування розкладу репетиторів на платформі Android. Цей вибір забезпечує максимальну доступність для користувачів, адже Android є найпоширенішою мобільною системою у світі, а також надає потужні можливості для створення сучасних програм.

Із основних інструментів для розробки — Visual Studio Code, офіційне середовище, яке робить процес створення додатка ефективним. Воно дозволяє легко створювати привабливий інтерфейс, швидко виправляти помилки, тестувати додаток на різних пристроях та забезпечувати його плавну роботу. Для написання коду обрав React Native — сучасний фреймворк, рекомендований для Android. JavaScript робить код більш читабельним, зменшує ймовірність помилок, легко інтегрується з JavaScript та дозволяє ефективно обробляти складні завдання.

Дані репетиторів будуть надійно зберігатися та синхронізуватися за допомогою Firebase Realtime Database. Це означає, що інформація буде доступна в будь-який час, на будь-якому пристрої, в режимі реального часу, а також значно спрощує технічну сторону роботи додатка.

Таблиця 3.1 – Аналіз вибраних інструментів

Фреймворк	React Native який дозволяє використовувати єдину базу коду для Android, забезпечуючи продуктивність
Мова	JavaScript через використання асинхронних функцій (async/await) дозволяє взаємодіяти з базою даних без блокування інтерфейсу
Середовище	Visual Studio Code
База даних	Firebase Realtime Database це хмарне рішення, що працює за

протоколом WebSocket, забезпечуючи оновлення даних у додатку в реальному часі без перезавантаження сторінок

Таким чином, ми створюємо мобільний додаток, який буде не тільки функціональним та стабільним, але й зручним для репетиторів, допомагаючи їм ефективно керувати своїм розкладом.

Окрім вибору платформи та середовища програмування, важливим складником стало дотримання актуальних методів творення мобільних додатків. Під час розробки використовувалися рекомендовані Google архітектурні принципи, які передбачають розмежування обов'язків між відображенням, бізнес-логікою та опрацюванням. Такий спосіб збільшує надійність програми, спрощує її обслуговування та дає змогу надалі просто доповнювати функціональні властивості.

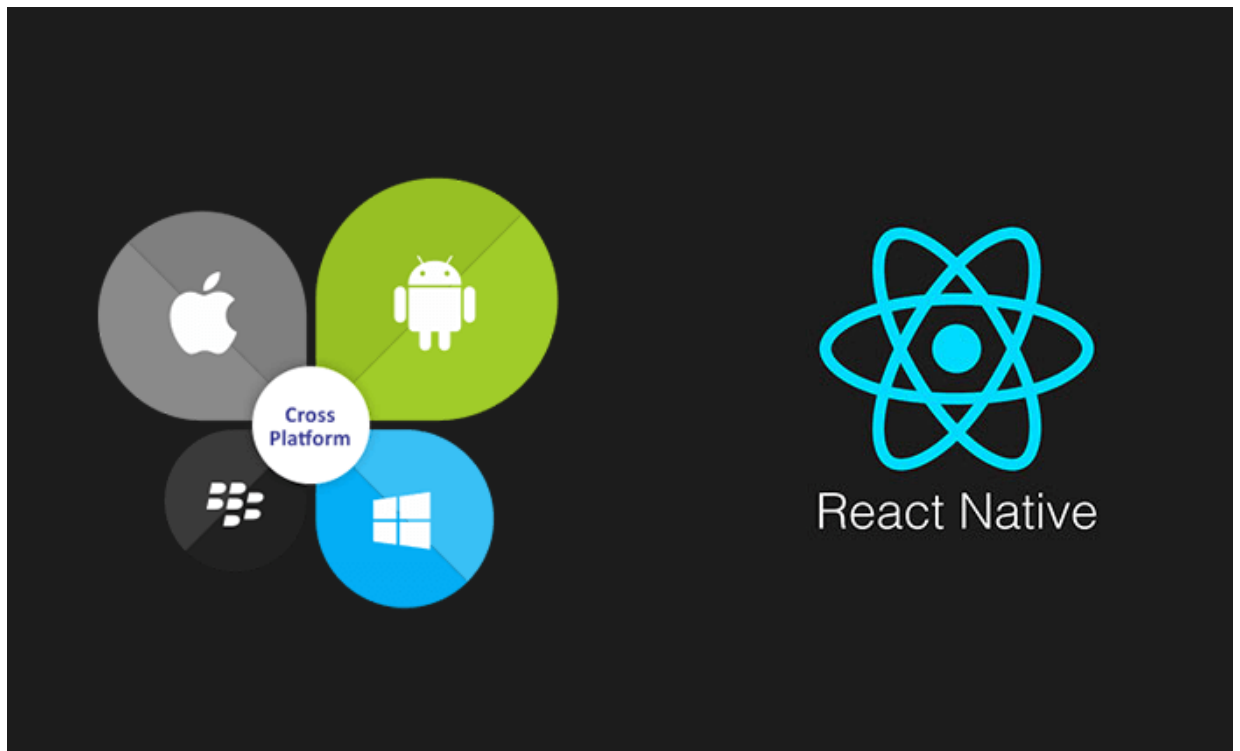


Рисунок 3.1 – React Native

Неабияку увагу приділено комфорту споживача та пристосованістю інтерфейсу. React Native дарує арсенал засобів для формування інтерфейсів,

які правильно показуються на гаджетах з різними габаритами дисплеїв та версіями системи. Це дає змогу гарантувати єдиний досвід користувача незалежно від виду телефону чи планшета [9].

Застосування мови програмування JavaScript позитивно вплинуло на гатунок програмного коду. Завдяки сучасним мовним конструкціям зменшується обсяг коду, підвищується його помітність та знижується вірогідність виникнення критичних помилок під час виконання програми. Також JavaScript гарно пасує для втілення асинхронних операцій, що є важливим при роботі з віддаленою базою даних та оновленні відомостей у реальному часі[10].

Використання Firebase Realtime Database дозволило реалізувати централізоване збереження даних без потреби у створенні власного серверного вирішення. Це значно полегшує процес розвитку та зменшує витрати часу на облаштування інфраструктури. Окрім того, Firebase гарантує автоматичну синхронізацію даних між приладами, що є особливо важливим у випадку застосування додатка з різних смартфонів або при зміні пристрою користувачем [11].

Отже, обране середовище програмування та набір технологій цілком відповідають вимогам до сучасного мобільного додатка. Вони забезпечують сталість, продуктивність, надійність збереження даних та створюють підґрунтя для подальшого розвитку програмного забезпечення без значних архітектурних змін.

3.2 Розробка бази даних

База даних реалізована за моделлю NoSQL. Дані зберігаються у вигляді єдиного JSON-дерева. Така структура дозволяє легко масштабувати додаток та швидко отримувати доступ до вкладених об'єктів.

Структура бази даних, розроблена для зберігання даних про учнів, їхні заняття та системні налаштування, базується на Firebase Realtime Database.

Дані організовані у три основні гілки:

- а) students – тут зберігається список усіх учнів. Кожен запис містить: ім'я, предмет, поточний баланс та колір картки.
- б) lessons – календарний план занять. Кожне заняття прив'язане до ID учня і містить дату та час проведення.
- в) history – журнал усіх проведених уроків. Сюди записується інформація про кожне списання коштів для подальшого аналізу доходів.

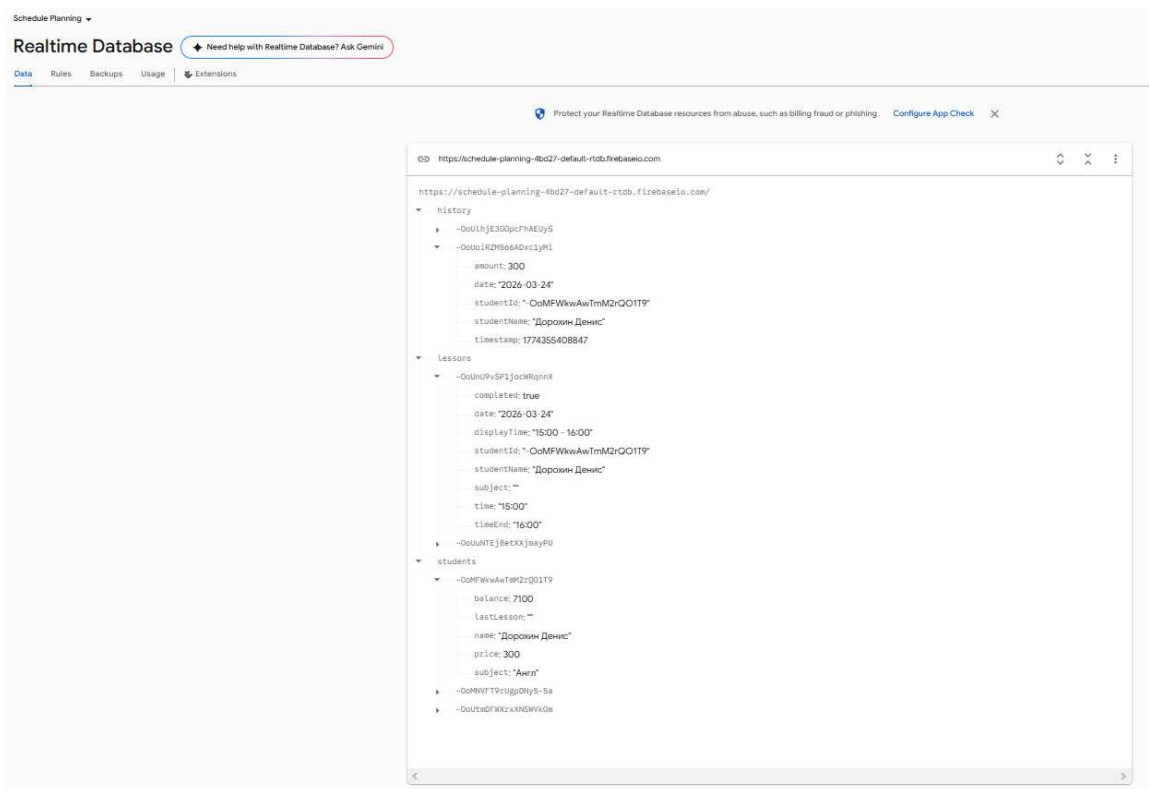


Рисунок 3.2 – Відображення учнів в базі

3.3 Програмна реалізація основних функцій

Додаток побудований на основі компонентного підходу React Native. Програмний код розділений на функціональні модулі (Екрани) та сервіси. Кожен компонент відповідає за конкретну частину бізнес-логіки.

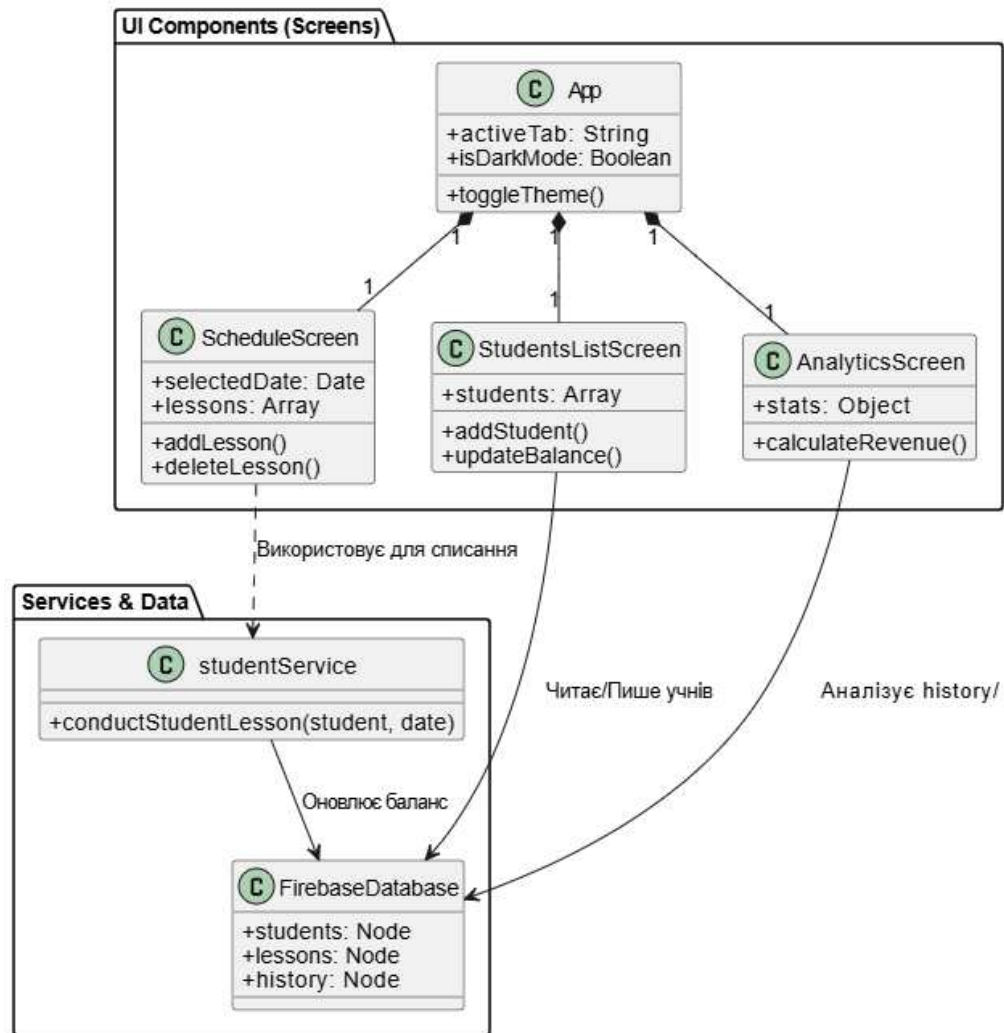


Рисунок 3.3 – Діаграма класів мобільного застосунку планування занять

Реалізовано наступний функціонал:

- Управління списком студентів це додавання та видалення;
- Можливість редагування балансу кожного студента;
- Створення уроків;
- Візуалізація розкладу занять у форматі календаря;
- Автоматична синхронізація даних в реальному часі.

3.4 Методика роботи користувача

Для автоматизації ручної праці репетитора додаток використовує алгоритм автоматичного розрахунку. Користувачу не потрібно пам'ятати, хто і скільки винен – система робить це сама.



Рисунок 3.4 – Діаграма взаємодії користувача з мобільним застосунком

Одразу після запуску програми він потрапляє на головний екран, де зручно розташовані календар із усіма запланованими заняттями та повний список його студентів. Зроблено інтуїтивно зрозумілий інтерфейс, щоб мінімізувати час, необхідний для навчання та ефективної роботи з системою.

Подальша праця фахівця із застосунком спирається на послідовне здійснення типових операцій, характерних для щоденної діяльності викладача. Хід програми збудований таким чином, щоб усі ключові дії виконувалися швидко та без потреби переходу між великою кількістю вікон. Це знижує тягар та зростає загальна дієвість застосування мобільного рішення.

Користувач має змогу оперувати базою учнів, оглядати відомості про кожного з них, а також додавати нових учнів чи коригувати вже наявні

записи. Усі дані зберігаються у впорядкованому вигляді, що забезпечує зручний пошук та швидкий доступ до необхідної інформації. Особливу увагу надано логіці зв'язку між учнем, його заняттями та фінансовим станом, що дозволяє запобігти неузгодженостям у даних.

Планування занять реалізоване через календарний механізм, який показує усі заплановані уроки з прив'язкою до дати та часу. Це надає змогу користувачеві швидко оцінювати завантаженість робочих днів, знаходити вільні інтервали часу та вчасно робити зміни до розкладу. У випадку редагування чи скасування заняття відповідні зміни автоматично поновлюються у системі, що гарантує актуальність розкладу.

Бухгалтерський облік вбудований безпосередньо у процес праці із заняттями. Після формування уроку система автоматично стягує встановлену ціну із залишку відповідного учня. Такий метод зменшує ймовірність хиб, пов'язаних із ручним обчисленням платежів, та дає змогу репетиторові постійно стежити за фінансовим становищем без додаткових розрахунків.

Отак, методика роботи апарату користувача із мобільним застосунком орієнтована на практичні потреби репетитора та передбачає логічну, послідовну взаємодію з усіма головними функціями системи, етапами створення бази даних та програмного забезпечення, призначеного для оптимізації розкладу занять. Вибір конкретного середовища розробки, архітектурного рішення та структури бази даних був прагненням забезпечити надійність функціонування застосунку та його потенціал для майбутніх удосконалень.

3.5 Інструкція користувача мобільного додатка

Тут зосереджено детальний опис практичного застосування для кінцевого користувача, а саме викладача. Вказівки щодо головних функцій програми, а також принципи взаємодії особи з програмним оформленням. У формі поетапного огляду головних робочих ситуацій, починаючи від старту програми й закінчуючи грошовим обліком та застосування системи.

Інтерфейс програми спроектований з урахуванням засад інтуїтивної зрозумілості, зменшення кількості кроків для виконання звичайних операцій та швидкого перегляду ключової інформації. Користувачу не потрібна спеціальна підготовка чи додаткове навчання для початку роботи із програмою, оскільки всі головні функції згруповані логічно і мають ясну візуальну структуру.

Також описується, як застосунок сприяє автоматизуванню буденних процесів, пов'язаних з плануванням уроків, керуванням переліком учнів та грошовими операціями, взаємодією з календарем, формуванням та зміною занять, а також способом автоматичного зняття коштів з рахунку учня.

Далі міститься покроковий опис основних функцій додатка, що формує цілісне уявлення про його практичне використання та демонструє зручність застосування розробленого програмного забезпечення у щоденній роботі репетитора.

3.5.1 Запуск та головний екран додатка

Після запуску додатка ініціалізується головний компонент App.js, який керує станом теми та навігацією. Користувач потрапляє на екран розкладу (ScheduleScreen), що містить горизонтальний календар для вибору дати та список занять [14].

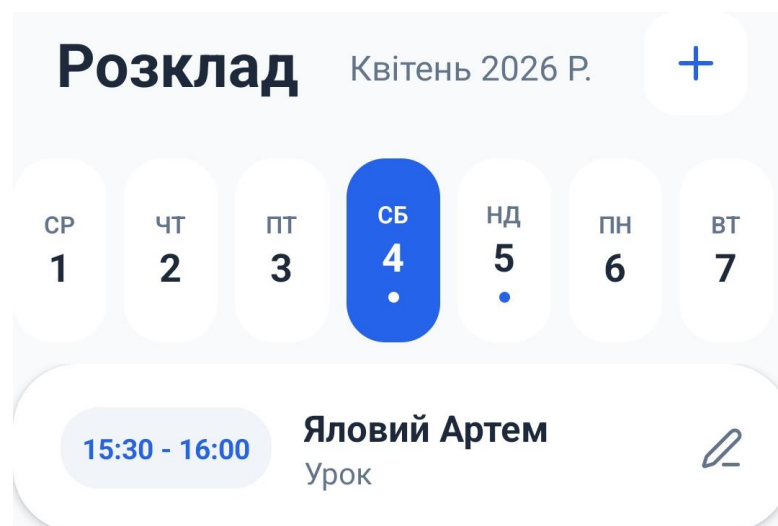


Рисунок 3.5.1. – Відображення колендаря на головному екрані

Ініціалізація головного екрану:

```
export default function App() {
  const [activeTab, setActiveTab] = useState("schedule");
  const [isDarkMode, setIsDarkMode] = useState(false);

  // Налаштування теми та відображення відповідного екрану
  return (
    <SafeAreaView style={[styles.container, { backgroundColor:
theme.bg }]}>
      <View style={{ flex: 1 }}>
        {activeTab === "schedule" && <ScheduleScreen
isDarkMode={isDarkMode} />}
        {activeTab === "students" && <StudentsListScreen
isDarkMode={isDarkMode} />}
        {activeTab === "analytics" && <AnalyticsScreen
isDarkMode={isDarkMode} />}
      </View>
      { /* TabBar для навігації */ }
    </SafeAreaView>
  );
}
```

3.5.2 Робота зі списком учнів

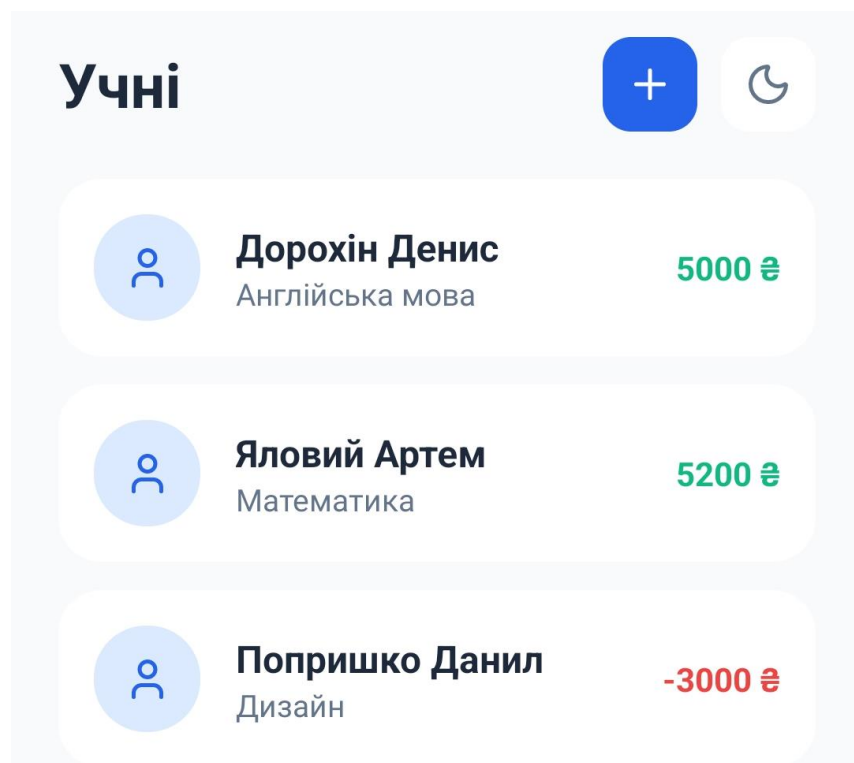


Рисунок 3.5.2. – Відображення учнів з балансом

На екрані `StudentsListScreen` відображається повний перелік учнів, завантажений з `Firebase` у реальному часі. Для кожного учня відображаються його ім'я, предмет та баланс

Система автоматично змінює колір суми: зелений для позитивного балансу та червоний для заборгованості:

- а) включаючи ім'я;
- б) предмет;
- в) поточний фінансовий баланс.

Додаток дозволяє додавати нових учнів, редагувати вже існуючі записи або видаляти учнів у разі завершення співпраці. Усі зміни автоматично зберігаються у базі даних `Firebase`, що гарантує надійність та збереження даних.

Відображення списку учнів:

```
const renderStudentItem = ({ item }) => (
  <View style={[styles.card, { backgroundColor: theme.card }]}>
    <Text style={[styles.name, { color: theme.text
  ]]}>{item.name}</Text>
    <Text style={{ color: theme.subText }}>{item.subject}</Text>
    <Text style={{
      fontWeight: 'bold',
      color: item.balance < 0 ? '#EF4444' : '#10B981'
    }}>
      Баланс: {item.balance} ₴
    </Text>
  </View>
);
```

3.5.3 Додавання та планування занять

Додавання занять відбувається через інтерфейс `ScheduleScreen`. Користувач обирає дату на календарній панелі та натискає кнопку додавання. У модальному вікні обирається учень зі списку `allStudents`. Після підтвердження дані зберігаються у вузлі `lessons` бази даних `Firebase`.

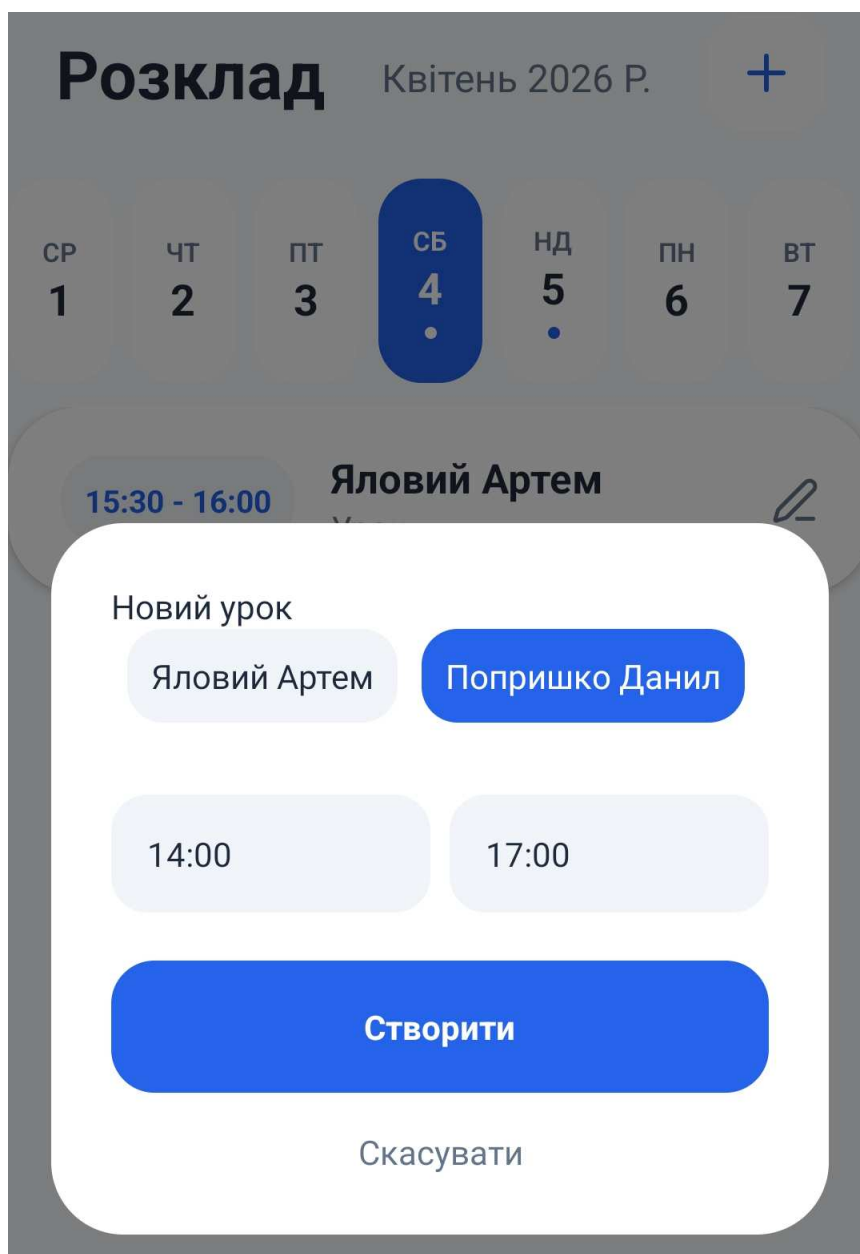


Рисунок 3.5.3. – Створення уроку

Генерація занять:

```
const handleAddLesson = () => {
  const lessonRef = ref(db, "lessons/");
  const newLesson = {
    studentId: selectedStudent.id,
    studentName: selectedStudent.name,
    date: selectedDate,
    time: selectedTime,
    createdAt: serverTimestamp(),
  };
  push(lessonRef, newLesson);
  setIsModalVisible(false);
};
```

3.5.4 Фінансовий облік та контроль балансу

Ключовою функцією додатка є автоматизація списання коштів. При відмітці уроку як проведеного викликається сервіс `studentService.js`, який автоматично віднімає вартість заняття від поточного балансу учня та оновлює дату останнього заняття.

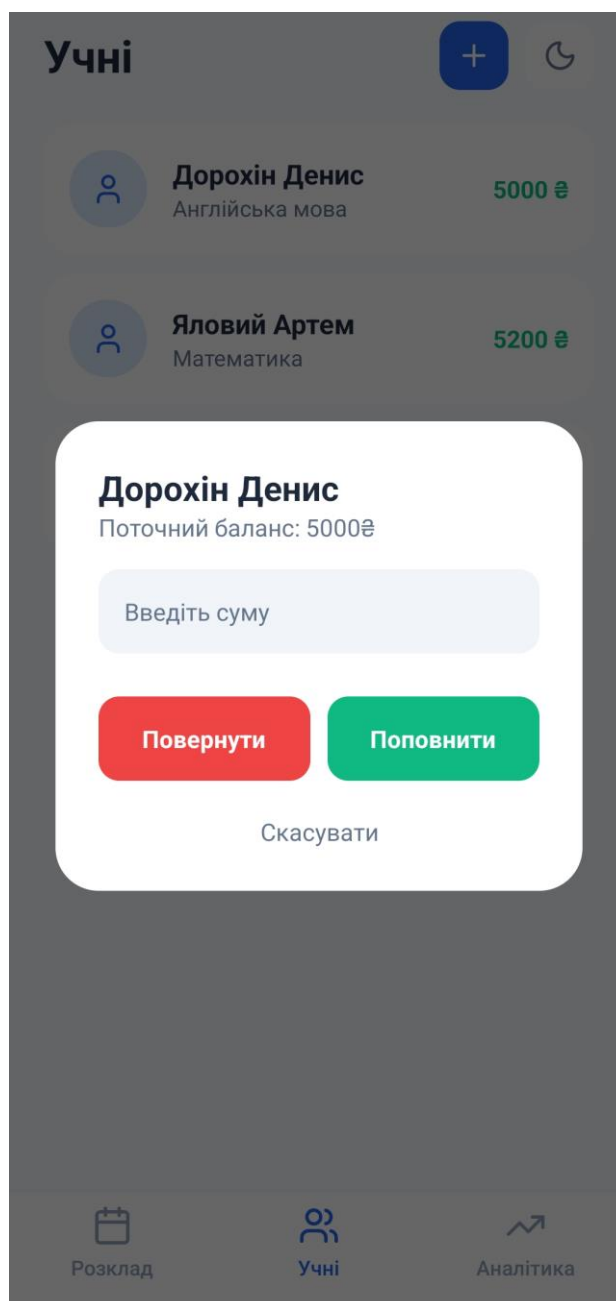


Рисунок 3.5.4.1. – Взаємодія з балансом учня

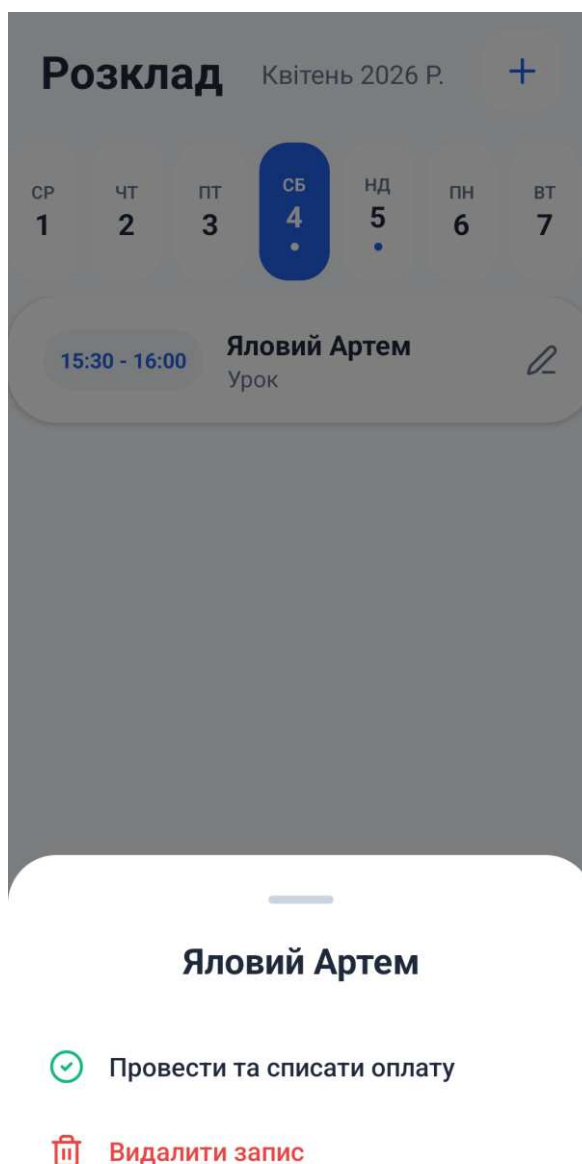


Рисунок 3.5.4.2. – Функціонал діючого уроку

Автоматизація фінансових операцій зменшує ризик помилок та підвищує прозорість взаєморозрахунків між репетитором та учнями.

Списання балансу:

```
export const conductStudentLesson = (student, selectedDate) => {
  const studentRef = ref(db, `students/${student.id}`);

  // Автоматичне зменшення балансу на 300 грн
  return update(studentRef, {
    ...student,
    balance: student.balance - 300,
    lastLesson: selectedDate
  });
};
```


3.5.5 Переваги використання додатка

Завдяки екрану AnalyticsScreen репетитор може бачити загальний дохід та рейтинг учнів за прибутковістю. Це реалізовано шляхом агрегації даних із вузла history у Firebase.

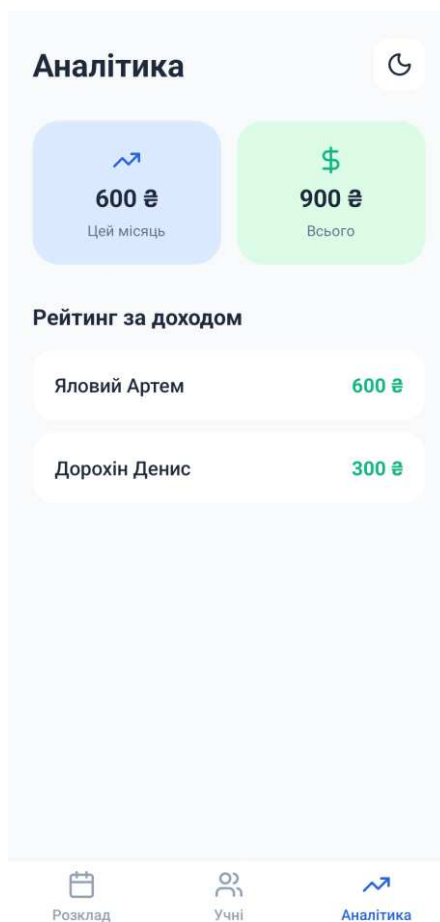


Рисунок 3.5.5. – Екран аналітики доходів

```
Object.values(data).forEach(entry => {
  total += Number(entry.amount);
  // Групування за доходом для рейтингу
  rankingMap[entry.studentId] = (rankingMap[entry.studentId] || 0) +
  Number(entry.amount);
});
```

Використання додатка на React Native забезпечує високу швидкість роботи та можливість використання темної теми для комфортної роботи в будь-який час доби.

ВИСНОВОК

Було успішно створено мобільний застосунок для Android-юзерів, який стане незамінним помічником для репетиторів. Мета – спросити та автоматизувати планування занять і організацію роботи. Дослідження показало, що більшість існуючих програм орієнтована на широку аудиторію, але не враховує унікальних потреб викладачів, таких як одночасне відстеження учнів, фінансових операцій та персоналізованих розкладів. Саме тому додаток є таким актуальним та необхідним

Розроблено додаток з використанням сучасної архітектури, роблячи його гнучким, легким у підтримці. Всі дані надійно зберігатимуться у хмарній базі Firebase Realtime Database, забезпечуючи миттєвий доступ та синхронізацію.

З цим додатком можливо легко керувати інформацією про кожного учня: зберігати дані, предмети, які ви викладаєте, та відстежувати їхній фінансовий баланс. Планування занять стане простим і корисним – є можливість створювати уроки, вказуючи дату, час, тривалість та тему, а додаток автоматично відобразить їх у зручному календарі, усуваючи ризики помилки.

Особливо зручною є функція автоматизації фінансових розрахунків. Після створення заняття вартість уроку автоматично списуватиметься з балансу учня, що позбавить вас необхідності ручних підрахунків та забезпечить точний фінансовий облік.

За інтерфейс додатка розроблено так, щоб бути максимально простим та зрозумілим. Головний екран з календарем дозволить миттєво бачити свій розклад, а окремі розділи дадуть зручний доступ до списку учнів, деталей занять та фінансової інформації. Можливість користуватися додатком без жодних труднощів.

Тестування підтвердило, що додаток працює стабільно та виконує всі поставлені завдання, від управління даними учнів до фінансових операцій та відображення розкладу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Android Developers. Сторінка про роботу з мережею та базами даних у архітектурі застосунків [Електронний ресурс]. – Режим доступу: <https://developer.android.com/topic/libraries/architecture/paging/v3-network-db>
2. Firebase. Firebase Realtime Database: зберігання та синхронізація даних у режимі реального часу [Електронний ресурс]. – Режим доступу: <https://firebase.google.com/products/realtime-database>
3. Applying Visualization for Teaching Scheduling Algorithms in Computer Science. Візуалізація для викладання алгоритмів планування в комп'ютерних науках [Електронний ресурс]. – Режим доступу: <https://scholarworks.calstate.edu/concern/projects/sx61dp21c>
4. SplashLearn. 10 найкращих застосунків для планування уроків для вчителів у 2025 році [Електронний ресурс]. – Режим доступу: <https://www.splashlearn.com/blog/best-lesson-planning-apps-for-teachers/>
5. Google Календар як інструмент для ефективної роботи [Електронний ресурс]. – Режим доступу: <https://wedex.com.ua/blog/google-kalendar-instrument-dlya-efektyvnoyi-roboty-i-osobystyh-sprav/>
6. Notion можливості інструменту для організації інформації [Електронний ресурс]. – Режим доступу: <https://maxnet.ua/blog/notion-oglyad-mozhливостей-universalnogo-instrumentu-dlya-organizaciyi-informaciyi/>
7. Фінансова система в Google Sheets [Електронний ресурс]. – Режим доступу: <https://www.youtube.com/watch?v=JVJc4ymOOMw>
8. Фінансові помилки які можуть дорого обійтись [Електронний ресурс]. – Режим доступу: <https://firstaccounting.com.ua/10-financial-mistakes-that-can-cost-your-business-dearly-uk/>
9. Android-розробник особливості професії, перспективи, необхідні навички та вміння [Електронний ресурс]. – Режим доступу: <https://javarush.com/ua/groups/posts/uk.2419.android-rozrobnik-osoblivost-profes-perspektivi-neobkhdn-navichki-ta-vmnnja>
10. Курс JavaScript [Електронний ресурс]. – Режим доступу:

<https://www.youtube.com/watch?v=oXuOWiaR9Sg>

11. Розробка додатків з Firebase [Електронний ресурс]. – Режим доступу: <https://avada-media.ua/blog/firebase/>

12. Is Firebase Realtime Database the Right Choice for Real-time Data Sync [Електронний ресурс]. – Режим доступу: <https://appmaster.io/blog/firebase-realtime-database-for-real-time-data-sync>

13. Розробка мобільних додатків від А до Я [Електронний ресурс]. – Режим доступу: <https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkov-vid-a-do-ja-povnij-gajd/>

14. ScrollView React Native [Електронний ресурс]. – Режим доступу: <https://reactnative.dev/docs/scrollview>

Список опублікованих праць:

1. Швець. Д. В., Котов І. А., Дорохін Д. А. Розробка мобільного застосунку для автоматизації розкладу репетиторів на базі Android та Firebase:

ДОДАТОК А

Клас логіку списання коштів та оновлення статусу учня – studentService.js

```
import { ref, set } from 'firebase/database';
import { db } from '../../firebaseConfig';

/**
 * Клас-сервіс для обробки логіки занять.
 * Метод conductStudentLesson:
 * - Зменшує баланс на 300 грн.
 * - Оновлює дату останнього уроку в Firebase.
 */
export const conductStudentLesson = (student, selectedDate) => {
  const studentRef = ref(db, `students/${student.id}`);

  return set(studentRef, {
    ...student,
    balance: student.balance - 300,
    lastLesson: selectedDate
  });
};
```

Компонент-клас картки StudentCard.js – це логіка відображення окремого елемента списку учнів.

```
import { BookOpen, Clock, User } from "lucide-native";
import { StyleSheet, Text, TouchableOpacity, View } from "react-native";

export const StudentCard = ({ item, onConduct }) => (
  <View style={styles.card}>
    <View style={styles.header}>
      <View style={styles.userInfo}>
        <View style={styles.iconContainer}>
          <User color="#2563EB" size={20} />
        </View>
        <Text style={styles.name}>{item.name}</Text>
      </View>
      <View style={[[styles.badge, { backgroundColor: item.balance < 0
? "#FEE2E2" : "#DCFCE7" }]]}>
        <Text style={[[styles.balance, { color: item.balance < 0 ?
"#EF4444" : "#16A34A" }]]}>
          {item.balance} ₪
        </Text>
      </View>
    </View>
    <View style={styles.details}>
```

```

    <View style={styles.infoRow}>
      <BookOpen size={16} color="#6B7280" />
      <Text style={styles.infoText}>{item.subject}</Text>
    </View>
    <View style={styles.infoRow}>
      <Clock size={16} color="#6B7280" />
      <Text style={styles.infoText}>Останній урок: {item.lastLesson
|| "немає"}</Text>
    </View>
  </View>

  <TouchableOpacity style={styles.button} onPress={() =>
onConduct(item.id, item.balance)}>
    <Text style={styles.buttonText}>Провести урок (-300 ₪)</Text>
  </TouchableOpacity>
</View>
);

const styles = StyleSheet.create({
  card: { backgroundColor: "#fff", borderRadius: 16, padding: 16,
marginBottom: 12, elevation: 3 },
  header: { flexDirection: "row", justifyContent: "space-between",
alignItems: "center", marginBottom: 12 },
  userInfo: { flexDirection: "row", alignItems: "center" },
  iconContainer: { backgroundColor: "#DBEAFE", padding: 8,
borderRadius: 10 },
  name: { fontSize: 17, fontWeight: "bold", marginLeft: 10 },
  badge: { paddingHorizontal: 10, paddingVertical: 4, borderRadius: 8
},
  balance: { fontWeight: "700" },
  details: { borderTopWidth: 1, borderTopColor: "#F3F4F6", paddingTop:
12, marginBottom: 12 },
  infoRow: { flexDirection: "row", alignItems: "center", marginBottom:
6 },
  infoText: { marginLeft: 8, color: "#4B5563", fontSize: 14 },
  button: { backgroundColor: "#2563EB", padding: 12, borderRadius: 12,
alignItems: "center" },
  buttonText: { color: "#fff", fontWeight: "600" }
});

```

Головний файл входу App.js – відповідає за навігацію, перемикання між екранами Розклад, Учні та Аналітика, перемиканням Dark/Light mode для всього застосунку, визначає нижню панель TabBar з іконками

```

import React, { useState } from "react";
import { SafeAreaView, StyleSheet, View, StatusBar, TouchableOpacity, Text } from
"react-native";
import ScheduleScreen from "../components/screens/ScheduleScreen";
import StudentsListScreen from "../components/screens/StudentsListScreen";
import AnalyticsScreen from "../components/screens/AnalyticsScreen";

```

```

import { Calendar as CalendarIcon, Users, TrendingUp } from "lucide-react-native";

export default function App() {
  const [activeTab, setActiveTab] = useState("schedule");
  const [isDarkMode, setIsDarkMode] = useState(false);

  const theme = {
    bg: isDarkMode ? "#0F172A" : "FFFFFF",
    tabBar: isDarkMode ? "#1E293B" : "FFFFFF",
    border: isDarkMode ? "#334155" : "F1F5F9",
    accent: "#2563EB"
  };

  const toggleTheme = () => setIsDarkMode(!isDarkMode);

  return (
    <SafeAreaView style={[styles.container, { backgroundColor: theme.bg }]}>
      <StatusBar barStyle={isDarkMode ? "light-content" : "dark-content"} />

      <View style={{ flex: 1 }}>
        {activeTab === "schedule" && <ScheduleScreen isDarkMode={isDarkMode}
toggleTheme={toggleTheme} />
        {activeTab === "students" && <StudentsListScreen isDarkMode={isDarkMode}
toggleTheme={toggleTheme} />
        {activeTab === "analytics" && <AnalyticsScreen isDarkMode={isDarkMode}
toggleTheme={toggleTheme} />
      </View>

      <View style={[styles.tabBar, { backgroundColor: theme.tabBar,
borderTopColor: theme.border }]}>
        <TouchableOpacity style={styles.tabItem} onPress={() =>
setActiveTab("schedule")}>
          <CalendarIcon color={activeTab === "schedule" ? theme.accent :
"#94A3B8"} size={24} />
          <Text style={[styles.tabLabel, { color: activeTab === "schedule" ?
theme.accent : "#94A3B8" }]}>Розклад</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.tabItem} onPress={() =>
setActiveTab("students")}>
          <Users color={activeTab === "students" ? theme.accent : "#94A3B8"}
size={24} />
          <Text style={[styles.tabLabel, { color: activeTab === "students" ?
theme.accent : "#94A3B8" }]}>Учні</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.tabItem} onPress={() =>
setActiveTab("analytics")}>
          <TrendingUp color={activeTab === "analytics" ? theme.accent :
"#94A3B8"} size={24} />

```

```

        <Text style={ [styles.tabLabel, { color: activeTab === "analytics" ?
theme.accent : "#94A3B8" }]}>Аналітика</Text>
      </TouchableOpacity>
    </View>
  </SafeAreaView>
);
}

const styles = StyleSheet.create({
  container: { flex: 1 },
  tabBar: { flexDirection: "row", height: 70, borderTopWidth: 1, paddingBottom:
10 },
  tabItem: { flex: 1, justifyContent: "center", alignItems: "center" },
  tabLabel: { fontSize: 12, marginTop: 4, fontWeight: "500" }
});

```


ДОДАТОК Б

Головний екран розкладу ScheduleScreen.js

```
import React, { useState } from "react";
import { View, Text, FlatList, TouchableOpacity, StyleSheet } from "react-native";
import { Plus } from "lucide-react-native";

export default function ScheduleScreen({ isDarkMode }) {
  const theme = {
    bg: isDarkMode ? "#0F172A" : "#F8FAFC",
    card: isDarkMode ? "#1E293B" : "#FFFFFF",
    text: isDarkMode ? "#F8FAFC" : "#1E293B",
  };

  return (
    <View style={[styles.container, { backgroundColor: theme.bg }]}>
      <View style={styles.headerRow}>
        <Text style={[styles.title, { color: theme.text }]}>Розклад</Text>
        <Text style={{ fontSize: 16, color: theme.subText, textTransform:
'capitalize', marginTop: 4 }}>
          {formatMonthYear(selectedDate)}
        </Text>
        <TouchableOpacity
          style={[styles.iconBtn, { backgroundColor: theme.card }]}
          onPress={() => setIsAddModalVisible(true)}
        >
          <Plus color={theme.accent} size={24} />
        </TouchableOpacity>
      </View>

      <View style={{ height: 95 }}>
        <FlatList
          ref={flatListRef}
          data={dates}
          horizontal
          showsHorizontalScrollIndicator={false}
          keyExtractor={(item) => item}
          getItemLayout={(data, index) => ({
            length: ITEM_WIDTH,
            offset: ITEM_WIDTH * index,
            index,
          })}
          renderItem={({ item }) => {
            const isSelected = selectedDate === item;
            return (
              <TouchableOpacity
                onPress={() => setSelectedDate(item)}
              >
```

```

        style=[
          styles.dateCard,
          { backgroundColor: isSelected ? theme.accent : theme.card },
        ]
      >
      <Text
        style=[
          styles.dateDay,
          { color: isSelected ? "#fff" : theme.subText },
        ]
      >
        {new Date(item).toLocaleDateString("uk", {
          weekday: "short",
        })}
      </Text>
      <Text
        style=[
          styles.dateNum,
          { color: isSelected ? "#fff" : theme.text },
        ]
      >
        {new Date(item).getDate()}
      </Text>
      {markedDates[item] && (
        <View
          style=[
            styles.dot,
            { backgroundColor: isSelected ? "#fff" : theme.accent },
          ]
        />
      )}
    </TouchableOpacity>
  );
}
/>
</View>

<FlatList
  data={lessons}
  keyExtractor={(item) => item.id}
  renderItem={({ item }) => (
    <TouchableOpacity
      style=[
        styles.lessonCard,
        {
          backgroundColor: item.completed
            ? isDarkMode
              ? "#064E3B"
              : "#DCFCE7"
            : theme.card,
        },
      ],
    )
  )
}

```

```

    ]]
    onPress={() => {
      setSelectedLesson(item);
      setIsEditMenuVisible(true);
    }}
  >
  <View
    style={[
      styles.timeBox,
      { backgroundColor: isDarkMode ? "#334155" : "#F1F5F9" },
    ]}
  >
    <Text style={[styles.timeText, { color: theme.accent }]}>
      {item.displayTime || item.time}
    </Text>
  </View>
  <View style={{ flex: 1, marginLeft: 15 }}>
    <Text style={[styles.name, { color: theme.text }]}>
      {item.studentName}
    </Text>
    <Text style={{ color: theme.subText }}>
      {item.completed ? "Завершено" : item.subject || "Урок"}
    </Text>
  </View>
  {item.completed ? (
    <CheckCircle2 size={20} color="#10B981" />
  ) : (
    <Edit3 size={20} color={theme.subText} />
  )}
</TouchableOpacity>
  )}
/>

{/* МОДАЛЬНЕ ВІКНО РЕДАГУВАННЯ / ВИДАЛЕННЯ */}
<Modal visible={isEditMenuVisible} transparent animationType="slide">
  <TouchableOpacity
    style={styles.modalOverlay}
    onPress={() => setIsEditMenuVisible(false)}
  >
    <View style={[styles.menuContent, { backgroundColor: theme.card }]}>
      <View style={styles.menuHandle} />
      <Text style={[styles.menuTitle, { color: theme.text }]}>
        {selectedLesson?.studentName}
      </Text>

      {!selectedLesson?.completed && (
        <TouchableOpacity
          style={styles.menuItem}
          onPress={() => completeLesson(selectedLesson)}
        >
          <CheckCircle2 color="#10B981" size={22} />

```

```

        <Text style={[styles.menuItemText, { color: theme.text }]}>
            Провести та списати оплату
        </Text>
    </TouchableOpacity>
)}

<TouchableOpacity
    style={styles.menuItem}
    onPress={() => deleteLesson(selectedLesson?.id)}
>
    <Trash2 color="#EF4444" size={22} />
    <Text style={[styles.menuItemText, { color: "#EF4444" }]}>
        Видалити запис
    </Text>
</TouchableOpacity>
</View>
</TouchableOpacity>
</Modal>

{/* МОДАЛЬНЕ ВІКНО ДОДАВАННЯ */}
<Modal visible={isAddModalVisible} animationType="fade" transparent>
    <View style={styles.modalOverlayCenter}>
        <View style={[styles.modalBox, { backgroundColor: theme.card }]}>
            <Text style={[styles.modalTitle, { color: theme.text }]}>
                Новий урок
            </Text>
            <ScrollView
                horizontal
                showsHorizontalScrollIndicator={false}
                style={{ marginBottom: 20 }}
            >
                {allStudents.map((s) => (
                    <TouchableOpacity
                        key={s.id}
                        onPress={() =>
                            setNewLesson({ ...newLesson, studentId: s.id })
                        }
                        style={[
                            styles.studentBtn,
                            {
                                backgroundColor:
                                    newLesson.studentId === s.id
                                    ? theme.accent
                                    : theme.input,
                            },
                        ]}
                    >
                        <Text
                            style={{
                                color: newLesson.studentId === s.id ? "#fff" : theme.text,
                            }}

```

```

        >
        {s.name}
      </Text>
    </TouchableOpacity>
  )))
</ScrollView>
<View
  style={{
    flexDirection: "row",
    justifyContent: "space-between",
    marginBottom: 20,
  }}
>
  <TextInput
    placeholder="3 (14:00)"
    placeholderTextColor={theme.subText}
    style={[
      styles.input,
      {
        backgroundColor: theme.input,
        color: theme.text,
        flex: 0.48,
      },
    ]}
    onChangeText={(t) => setNewLesson({ ...newLesson, time: t })}
  />
  <TextInput
    placeholder="до (15:00)"
    placeholderTextColor={theme.subText}
    style={[
      styles.input,
      {
        backgroundColor: theme.input,
        color: theme.text,
        flex: 0.48,
      },
    ]}
    onChangeText={(t) => setNewLesson({ ...newLesson, timeEnd: t })}
  />
</View>
<TouchableOpacity
  style={[styles.saveBtn, { backgroundColor: theme.accent }]}
  onPress={saveLesson}
>
  <Text style={{ color: "#fff", fontWeight: "bold" }}>
    Створити
  </Text>
</TouchableOpacity>
<TouchableOpacity
  onPress={() => setIsAddModalVisible(false)}
  style={{ marginTop: 15 }}

```

```

        >
        <Text style={{ color: theme.subText, textAlign: "center" }}>
            Скасувати
        </Text>
    </TouchableOpacity>
</View>
</View>
</Modal>
</View>
);
}

const styles = StyleSheet.create({
  container: { flex: 1, paddingTop: 20, marginTop: 20 },
  headerRow: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
    paddingHorizontal: 20,
    marginBottom: 20,
    marginTop: 20,
  },
  title: { fontSize: 28, fontWeight: "bold" },
  iconBtn: { padding: 12, borderRadius: 15 },
  dateCard: {
    width: ITEM_WIDTH - 8,
    height: 85,
    borderRadius: 18,
    justifyContent: "center",
    alignItems: "center",
    marginRight: 8,
  },
  dateDay: { fontSize: 11, fontWeight: "600", textTransform: "uppercase" },
  dateNum: { fontSize: 18, fontWeight: "bold" },
  dot: { width: 5, height: 5, borderRadius: 3, marginTop: 4 },
  lessonCard: {
    paddingVertical: 18,
    paddingHorizontal: 22,
    borderRadius: 30,
    marginBottom: 12,
    elevation: 3,
    flexDirection: "row",
    alignItems: "center",
  },
  timeBox: {
    padding: 10,
    borderRadius: 18,
    minWidth: 70,
    alignItems: "center",
  },
  timeText: { fontSize: 13, fontWeight: "bold" },

```

```

name: { fontSize: 17, fontWeight: "bold" },
modalOverlay: {
  flex: 1,
  backgroundColor: "rgba(0,0,0,0.5)",
  justifyContent: "flex-end",
},
modalOverlayCenter: {
  flex: 1,
  backgroundColor: "rgba(0,0,0,0.5)",
  justifyContent: "center",
  alignItems: "center",
},
menuContent: {
  padding: 25,
  borderTopLeftRadius: 30,
  borderTopRightRadius: 30,
},
menuHandle: {
  width: 40,
  height: 5,
  backgroundColor: "#cbd5e1",
  borderRadius: 10,
  alignSelf: "center",
  marginBottom: 20,
},
menuTitle: {
  fontSize: 22,
  fontWeight: "bold",
  marginBottom: 25,
  textAlign: "center",
},
menuItem: { flexDirection: "row", alignItems: "center", paddingVertical: 15 },
menuItemText: { marginLeft: 15, fontSize: 16, fontWeight: "500" },
modalBox: { width: "90%", padding: 25, borderRadius: 30 },
input: { padding: 15, borderRadius: 15 },
studentBtn: {
  padding: 10,
  borderRadius: 15,
  marginRight: 10,
  marginBottom: 10,
},
saveBtn: { padding: 18, borderRadius: 18, alignItems: "center" },
));

```

Екрана аналітики доходів AnalyticsScreen.js

```

import React from "react";
import { ScrollView, View, Text, StyleSheet } from "react-native";
import { DollarSign, TrendingUp } from "lucide-react-native";

export default function AnalyticsScreen({ isDarkMode }) {

```

```

const theme = {
  bg: isDarkMode ? "#0F172A" : "#F8FAFC",
  card: isDarkMode ? "#1E293B" : "#FFFFFF",
  text: isDarkMode ? "#F8FAFC" : "#1E293B",
};

return (
  <ScrollView style={[styles.container, { backgroundColor: theme.bg }]}>
    <Text style={[styles.title, { color: theme.text }]}>Аналітика</Text>

    <View style={styles.row}>
      <View style={[styles.statCard, { backgroundColor: theme.card }]}>
        <TrendingUp color="#10B981" size={24} />
        <Text style={[styles.val, { color: theme.text }]}>12,500 €</Text>
        <Text style={{ color: "#64748B" }}>За місяць</Text>
      </View>

      <View style={[styles.statCard, { backgroundColor: theme.card }]}>
        <DollarSign color="#2563EB" size={24} />
        <Text style={[styles.val, { color: theme.text }]}>48,000 €</Text>
        <Text style={{ color: "#64748B" }}>Всього</Text>
      </View>
    </View>
  </ScrollView>
);
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 20 },
  title: { fontSize: 26, fontWeight: "bold", marginBottom: 25 },
  row: { flexDirection: 'row', justifyContent: 'space-between' },
  statCard: { width: '48%', padding: 20, borderRadius: 22, alignItems: 'center',
  elevation: 2 },
  val: { fontSize: 20, fontWeight: "bold", marginTop: 10 }
});

```


ДОДАТОК В

Посилання на застосунок:

https://mega.nz/file/OJ8imaSA#3gInlp2SIJUPq_8a4C0M7mhexlHdL8OOudkjCFI9QmY