ВСТУП

Сучасний прогрес у сфері інформаційних технологій створює нові шляхи для підвищення ефективності та оптимізації комунікаційних процесів, особливо у сфері освіти. Мобільні додатки значно спрощуючи доступ до інформації та забезпечуючи ефективну взаємодію між користувачами. В умовах швидкого темпу життя студенти часто стикаються з проблемами отримання своєчасних повідомлень про зміни в навчальному процесі або організаційні заходи. Не поточність комунікацій може викликати такі незручності, як пропуск важливих подій або недостатню поінформованість про актуальні зміни.

На сьогоднішній день багато закладів вищої освіти використовують спеціальні платформи для інформування студентів. Проте часто такі рішення мають обмежений функціонал або складний інтерфейс, що ускладнює доступ до необхідних даних. Існує необхідність у єдиному рішенні, яке дозволить студентам зручно та вчасно отримувати важливі повідомлення про організаційні події, зміни в розкладі тощо.

Метою цього проекту є розробка мобільного застосунку, який забезпечить централізоване сповіщення студентів. Ключовою особливістю застосунку стане персоналізація сповіщень залежно від ідентифікації користувача, що враховуватиме специфічні потреби студентів та надаватиме актуальну інформацію. Це сприятиме підвищенню зручності та оперативності комунікації в освітньому середовищі.

Актуальність цієї теми обґрунтована необхідністю забезпечення швидкого та ефективного інформування студентів в умовах стрімкого технологічного прогресу. Впровадження такого застосунку допоможе закладам освіти покращити взаємодію зі студентами, оптимізувати інформаційні потоки та забезпечити високий рівень комфорту для користувачів. Успішна реалізація проекту дозволить створити рішення, яке стане незамінним помічником у студентському житті.

1. ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Загальний огляд задачі

Мобільні телефони, планшети, смартфони та інші портативні пристрої давно стали невід'ємною частиною життя студентів. У світі, де 9 із 10 користувачів Інтернету підключаються через мобільні пристрої, їхня роль у навчальному процесі стрімко зростає. В Україні, за даними досліджень, кількість інтернет-користувачів на початок 2024 року перевищила 32 мільйони осіб, причому понад 70% з них активно використовують мобільні пристрої для доступу до інформації.

Сучасні студенти стикаються з великою кількістю завдань, які потребують швидкого та зручного доступу до організаційної інформації. В умовах динамічного навчального середовища часто виникає потреба в отриманні оперативних сповіщень про зміни в розкладі, організацію заходів чи виконання інших завдань. Пандемія і війна підкреслюють недоліки системи дистанційного та мобільного навчання в усьому світі, зокрема в Україні, де більшість освітніх закладів не мали достатніх цифрових рішень для ефективної комунікації зі студентами.

Розробка мобільного застосунку, що дозволяє централізовано сповіщати студентів про важливі події та зміни у навчальному процесі, є актуальним завданням для підвищення ефективності навчального процесу. Мета цього проекту є створення такого програмного рішення, яке задовольнить потреби сучасного студентства, враховуючи технічний прогрес і зростаючі вимоги до цифровізації освітнього середовища.

Щоб досягти поставленої задачі, потрібно вирішити такі завдання:

- аналіз існуючих рішень, що забезпечують інформування студентів через мобільні застосунки, включаючи переваги та недоліки аналогів.
- обґрунтувати необхідність створення нового рішення на основі аналізу функціоналу сучасних додатків.

- вибрати оптимальні засоби розробки, зокрема мову програмування, платформи для розробки мобільних додатків і архітектурні рішення для забезпечення масштабованості.

- створити функціональний інтерфейс взаємодії з користувачем, що враховує простоту, зручність та адаптивність до потреб різних студентів.

- забезпечити можливість надсилання сповіщень про заходи, зміни в розкладі або важливі дати через інтеграцію з внутрішніми базами даних навчального закладу.

- провести тестування розробленого додатку на основі реальних сценаріїв використання.

Мобільні технології стають необхідними не лише через зростаючу кількість інтернет-користувачів, але й завдяки їхній універсальності. Сьогодні більшість студентів надають перевагу мобільним рішенням, що дозволяють швидко отримувати доступ до необхідної інформації без прив’язки до фізичного місця або стаціонарного комп’ютера.

Дослідження 2024 року показують, що студенти в середньому отримують понад 10 повідомлень на день від різних навчальних платформ. Однак значна частина цих повідомлень або дублюється, або залишається нерелевантною для кінцевого користувача. Це підкреслює важливість персоналізації сповіщень.

1.2 Типи та особливості мобільних додатків

1.2.1 Типи мобільних додатків

Мобільні додатки класифікуються залежно від ряду характеристик, які визначають їх функціональність, спосіб розробки та аудиторію. Основні категорії:

- за типом використання. Цей підхід до класифікації визначає, для чого саме використовується мобільний додаток, і які задачі він виконує:

1) Інформаційні додатки – це додатки орієнтовані на передачу корисної інформації користувачам, наприклад, новини, погодні умови, актуальні події чи аналітику такі як BBC News, AccuWeather.

2) Комунікаційні додатки – це додатки для обміну повідомленнями, голосового або відео зв’язку. Вони включають додаткові функції, такі як спільні чати чи відео конференції, прикладом є Zoom, WhatsApp, Discord.

3) Розважальні додатки – вони зосереджені на задоволенні потреб у розвагах. Це ігри, відео та музичні платформи, також додатки для малювання чи читання, прикладом є Netflix, TikTok.

4) Освітні додатки – це додатки для онлайн-курсів, тестування, самостійного навчання та управління розкладом.

5) Бізнес-додатки – вони призначені для управління проектами, моніторингу продуктивності, фінансових розрахунків тощо.

- за платформою. Класифікація залежить від того, на якій операційній системі та технологічній базі працює додаток:

1) Нативні додатки - створені для конкретної операційної системи (наприклад, iOS чи Android), завдяки чому забезпечується висока продуктивність і доступ до всіх апаратних функцій пристрою (камери, GPS).

2) Кросплатформені додатки - використовують фреймворки (Flutter, React Native), що дозволяє створювати додатки, які працюють одночасно на кількох платформах, знижуючи витрати.

3) Веб-додатки - працюють через браузер без необхідності встановлення. Вони менш інтерактивні, але доступні з будь-якого пристрою.

- за складністю функціоналу. Ця класифікація враховує кількість функцій, рівень інтеграції з іншими системами та технічні вимоги:

1) Прості додатки - вони мають обмежену кількість функцій, але швидкі в розробці.

2) Середньої складності - включають функції інтеграції з базами даних, інтерактивний інтерфейс або геолокацію.

3) Складні додатки - містять великий функціонал, використовують AI, блокчейн чи машинне навчання.

- за цільовою аудиторією. Ця класифікація базується на тому, для кого створено мобільний додаток:

1) Персональні додатки - створені для індивідуального використання, наприклад, трекери звичок чи фінансів.

2) Корпоративні додатки - використовуються в компаніях для внутрішніх бізнес-процесів (наприклад, програми управління персоналом).

3) Масові додатки - соціальні мережі, маркетплейси або онлайн-магазини для широкої аудиторії.

- сфери застосування мобільних додатків:

1) Освіта - мобільні додатки змінили підхід до навчання, дозволяючи дистанційно отримувати знання, виконувати тести, відстежувати успішність і планувати розклад.

2) Бізнес і фінанси - дозволяють підприємствам оптимізувати операції, взаємодіяти з клієнтами та забезпечувати зручний доступ до послуг.

3) Медицина - від моніторингу здоров'я до запису на консультації. Такі додатки особливо актуальні у період пандемій.

4) Розваги - ігри, стримінгові платформи, додатки для створення контенту. Вони зосереджені на підвищенні залученості користувача.

5) Логістика - мобільні додатки полегшують пошук маршрутів, моніторинг вантажів чи замовлення транспорту.

1.2.2 Особливості розробки мобільних додатків

Складності розробки мобільних додатків. Технологічні виклики - вибір платформи. Це нативні додатки складніші у розробці, але забезпечують кращу продуктивність. Сумісність – забезпечення стабільної роботи на різних пристроях, ОС і роздільній здатності екранів. Також впровадження новітніх технологій. Використання AI, AR/VR або інтеграція з «хмарними» сервісами. Ще проектування інтерфейсу - інтуїтивно зрозумілий інтерфейс впливає на залучення користувачів, але його створення потребує досліджень і тестувань. Одне з найголовніших безпека - захист даних користувачів є пріоритетом. Впроваджуються шифрування, багатофакторна автентифікація, захист від витоку даних. Ще ресурси - розробка

складних додатків вимагає команди з програмістів, дизайнерів, тестувальників, що збільшує витрати. І також тестування - необхідно протестувати додаток на різних пристроях і сценаріях використання. Це все потребує часу й ресурсів.

1.3 Огляд існуючих рішень

Сучасний навчальний процес потребує ефективних цифрових рішень для організації та управління інформацією. На сьогодні існує велика кількість платформ і додатків, які спрямовані на оптимізацію навчального процесу, забезпечення доступу до розкладів, журналів успішності та інших ресурсів. Однак кожна із цих систем має свої переваги та недоліки, що впливає на їх ефективність та зручність у використанні.

У цій роботі ми зосередимо увагу на веб-додатку ДУІКТ. Цей додаток розроблений для того, щоб забезпечити студентів і викладачів доступом до важливої інформації в межах навчального закладу. Одним із ключових аспектів розробки подібних систем є дотримання балансу між зручністю користування і безпекою даних. Хоча інформація про розклад занять чи сесій не належить до категорії конфіденційної, відкритий доступ до таких даних також не завжди є доцільним через можливі ризики неправомірного використання.

Функціонал додатка можна описати наступним чином:

- перегляд розкладу занять та сесій. Ця функція дозволяє студентам і викладачам швидко знайти актуальну інформацію про час і місце проведення занять, а також внести зміни у свої плани в разі оновлення розкладу.

- доступ до онлайн-журналу успішності. Користувачі можуть переглядати свої оцінки за певний період, що дозволяє їм аналізувати свій прогрес у навчанні.

- перегляд журналу відвідуваності. Ця функція корисна як для студентів, які можуть контролювати свою присутність на заняттях, так і для викладачів, які відстежують активність студентів.

- список викладачів. Інформація про викладачів, їхні дисципліни та контакти допомагає студентам краще орієнтуватися в освітньому середовищі.

- робочий план. Додаток дозволяє користувачам переглядати основні навчальні завдання, дедлайни та інші важливі аспекти навчального процесу.

- авторизація. Ця функція забезпечує індивідуальний доступ до даних для кожного користувача, що гарантує безпеку та персоналізований досвід.

Проте, як і в багатьох подібних додатках, у ДУІКТ відсутня важлива функція – можливість отримувати автоматичні сповіщення про зміни в навчальному процесі чи інших важливих аспектах діяльності навчального закладу. Це може бути особливо корисним для своєчасного інформування студентів про перенесення занять, зміни в розкладі чи нові події.

На рисунку 1.1 показана початкова сторінка додатка ДУІКТ.

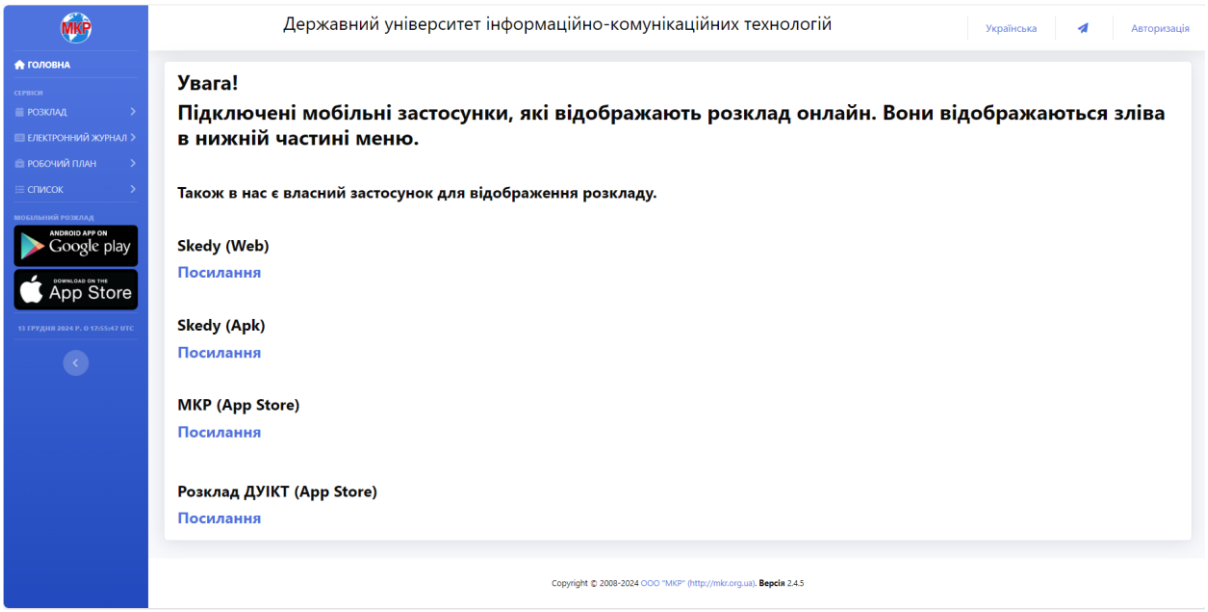


Рисунок 1.1 – Початкова сторінка додатка ДУІКТ

1.4 Постановка задачі

Задача цього проекту це розробка мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача.

Даний застосунок буде призначений для студентів та викладачів, також для адміністраторів додатка.

Розроблений застосунок повинен мати такі основні характеристики:

- централізоване управління інформаційними потоками.
- персоналізація повідомлень на основі групи користувача.
- можливість вчителів зробити будь яке повідомлення.
- для адміністраторів всі основні функції адміністрування додатка.
- доступність для основних мобільних платформ (iOS, Android).

Зручний інтерфейс, орієнтований на студентів із різними технічними навичками.

Впровадження такого рішення в навчальний процес сприятиме підвищенню комунікації між адміністрацією закладу освіти та студентами, а також дозволить знизити рівень інформаційного шуму, що виникає через недоліки традиційних каналів зв'язку. Успішна реалізація проекту відкриє нові перспективи для цифровізації освітнього середовища та підвищить його конкурентоспроможність у глобальному контексті.

Висновок до розділу

Мобільні додатки залишаються важливими інструментами для вирішення завдань у різних сферах. Їх класифікація та застосування демонструють масштаб можливостей, а розуміння складностей розробки дозволяє створювати якісні продукти, які відповідають очікуванням користувачів. Успішна реалізація мобільного додатку потребує балансу між інноваціями, функціональністю та витратами.

Розглянутий функціонал веб-додатка ДУІКТ демонструє потенціал для поліпшення управління навчальною інформацією. Основними перевагами додатка є доступ до розкладу занять, журналів успішності й присутності, а також інформації про викладачів та навчальні плани. Однак, відсутність функції автоматичних сповіщень знижує його корисність у контексті динамічного навчального середовища, де важливо швидко інформувати студентів про зміни.

Розробка мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача дозволить вирішити низку

актуальних проблем. Такий додаток матиме персоналізований підхід до сповіщень, централізоване управління інформаційними потоками та інтуїтивно зрозумілий інтерфейс для користувачів із різними технічними навичками.

Впровадження мобільного застосунку забезпечить:

- покращення комунікації між адміністрацією навчального закладу, викладачами та студентами;
- персоналізований підхід до надання інформації залежно від ролі користувача (студент, викладач, адміністратор);
- оперативність у передачі важливих повідомлень про зміни в розкладі чи організацію заходів;
- зниження інформаційного шуму завдяки чіткій фільтрації повідомлень.

Високий рівень адаптивності, інтеграція з основними платформами (такі як Android і iOS) та підтримка функцій адміністрування роблять цей проект актуальним і перспективним для цифровізації освітнього середовища. Успішна реалізація такого рішення дозволить підвищити конкурентоспроможність навчального закладу, покращити користувацький досвід студентів і викладачів, а також створити нові можливості для оптимізації управлінських процесів.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

Як ми могли переконатися з попередніх пунктів, процес розробки мобільного додатку є складним і вимагає вирішення низки завдань та викликів. Велика кількість доступних технологій, архітектур і підходів до створення програмного забезпечення дає розробникам широкий вибір, але водночас створює складнощі, пов'язані з необхідністю зробити оптимальний вибір для конкретного проекту.

Мобільні додатки сьогодні має мати наступні характеристики, бути зручними у використанні та забезпечувати необхідний функціонал. Для досягнення цього необхідно враховувати низку технічних і користувацьких вимог, які визначають як загальну концепцію, так і деталі реалізації додатку.

У цьому розділі ми детально розглянемо всі вимоги до розроблюваного додатку, враховуючи такі аспекти:

Успішний додаток повинен бути побудований на основі продуманої архітектури, яка забезпечує масштабованість, стабільність та легкість у підтримці. Ми проаналізуємо можливі підходи до організації архітектури додатку, включаючи її структуру та взаємодію між компонентами.

Розробка мобільного додатку потребує вибору відповідної мови програмування, платформ для розробки та інструментів для забезпечення ефективності процесу. Ми розглянемо, які технології найкраще підходять для реалізації нашого проекту, а також порівняємо альтернативи. Також одним із ключових елементів є визначення функціоналу додатку. Ми зосередимося на тому, які завдання він повинен виконувати, як організований інтерфейс користувача та які дані потрібно обробляти. У сучасних умовах особлива увага приділяється безпеці даних користувачів. Ми розглянемо методи захисту персональної інформації, що дозволять зробити додаток надійним і відповідним сучасним стандартам кібербезпеки. Одним з головних є забезпечення зручності використання додатку ми визначимо вимоги до його дизайну, логіки інтерфейсу та адаптивності для різних пристроїв.

Отже ми можемо виділити основні вимоги:

- архітектурні рішення.
- технологічні вимоги.
- функціональні вимоги.
- безпека.

2.1 Теоритичні данні для створення мобільного застосунку.

Створення мобільного застосунку - це складний багатоступеневий процес, що потребує детального планування та міждисциплінарного підходу та постійного тестування. Кожен етап має свої моменти, виклики та потребує професійного підходу.

На початковому етапі необхідно чітко сформулювати ідею додатку, визначити, яку проблему він вирішуватиме, і для кого він створюється. Основні кроки включають у собі. Формулювання ідеї, на цьому етапі можна задавати собі такі питання “Чому користувачам буде цікавий цей додаток?”, “Яку унікальну цінність він принесе?”, “Чи є цей додаток зручним для різних категорій користувачів?”. Також потрібно визначитись з цільовою аудиторією. Визначення профілю користувачів, вік, інтереси, проблеми, які додаток допоможе вирішити. Чітке формулювання цілей: освітні, комерційні, розважальні або інші.

Приклад: Якщо додаток орієнтований на студентів, його мета може полягати в оптимізації доступу до навчальних матеріалів, розкладу або сповіщень.

Також перед розробкою необхідно детально вивчити ринок, щоб оцінити конкурентів. Сюди входять таки етапи як дослідження конкурентів, дослідження конкурентів, огляд схожих додатків, виявлення їхніх сильних і слабких сторін, аналіз відгуків користувачів для розуміння їхніх потреб.

Також потрібно визначити що саме нас буде виділяти серед інших, якась унікальність

Самим простим способом визначення унікальності є задавання собі наступних питань:

- Що виділятиме ваш додаток серед інших?

- Чим він буде корисніший або зручніший?

Особливо потрібно приділити увагу трендам ринку. Вивчення популярних технологій. Аналіз поведінки користувачів можна провести опитування що б дізнатися який інтерфейс чи функціонал вони б хотіли побачити у додатку.

Пройшовши ці пункти, ми отримаємо глибоке розуміння ринку та конкурентів. Це дозволить створити продукт, який відповідатиме запитам аудиторії.

Наступним етапом стане розробка технічного завдання.

Технічне завдання - документ, який описує всі аспекти додатку та слугує дорожньою картою для команди розробників.

Технічне завдання може включати наступні пункти. Функціональні вимоги - основні функції (авторизація, сповіщення, бази даних) та додатковий функціонал (аналітика, інтеграція з іншими системами). Також технічні вимоги - мова програмування. Тип бази даних (SQL або NoSQL залежно від обсягу даних). UI/UX-дизайн - основні принципи дизайну це зручність, мінімалізм, адаптивність. Інтеграції -підключення до API або зовнішніх сервісів (Google Maps, платіжні системи тощо).

В результаті ми отримуємо повне розуміння, що саме потрібно розробити, які інструменти використовувати і якого результату досягти.

Далі етап прототипування та дизайну. Цей етап здійснює створення перших візуальних моделей додатку, які демонструють, як виглядатимуть екрани та як вони взаємодіятимуть між собою.

На цьому етапі є наступне.

Wireframes. Створення чорнових макетів екранів без деталізації дизайну. Також відображення основних елементів: кнопок, меню, полів вводу. Опрацювання зручності навігації. Визначення логіки переходів між екранами.

Далі іде UI-дизайн. Розробка візуального стилю: кольорова гама, шрифти, іконки. Адаптація для різних розмірів екранів.

Після всіх пунктів можна розпочати розробку мобільного застосунку.

Це ключовий етап, під час якого дизайн перетворюється на функціональний додаток.

С початку потрібно зробити Back-end.

А саме розробити серверну частину. В розробку серверної частини входять бази даних, API, системи автентифікації. Ще потрібно забезпечити стабільність та безпеку.

Після цього можна розробляти Front-end:

В її розробку входять реалізація дизайну та інтерактивності на основі прототипів. Інтеграція з серверною частиною через API. Розробка специфічного функціоналу. Таких як інтеграція push-сповіщень, геолокації, платіжних систем. Також оптимізація швидкості роботи.

Додатковим етапом являється тестування.

Додаток повинен бути перевірений на стабільність, функціональність і зручність перед запуском. Потрібно провести функціональне тестування, перевірити відповідності функціоналу вимогам ТЗ, тестування продуктивності, оцінка швидкодії за різного навантаження, тестування сумісності, перевірка роботи на різних пристроях і ОС, аналіз зручності використання для кінцевих користувачів.

Після успішного тестування додатоку він деплоється на свої платформи.

Для цього потрібно підготуватись до релізу. А саме створення опису додатку, скріншотів, відеоінструкцій. Також перевірка відповідності політикам платформ.

Основним етапом для просування додатку це маркетинг. Що б про ваш додаток дізналися потрібна в першу чергу реклама в соціальних мережах, SEO-оптимізація, залучення користувачів через блог-платформи.

Коли продукт запущено остається підтримка та його оновлення. Для того що б це робити ефективно потрібно виправляти усі помилок. Потрібно оперативна реакція на відгуки користувачів. Також потрібно постійно оновлювати функціонал. Розширення можливостей відповідно до потреб користувачів. Потрібно моніторити поведінку користувачів через спеціальні інструменти.

2.2 Вибір платформи та технологій для розробки

Розробка програмного забезпечення потребує вибору платформи, яка найкраще відповідатиме вимогам проекту, аудиторії та наявним ресурсам. Основні платформи та технології, що використовуються для розробки сучасних додатків, включають Android, iOS та WEB. Кожна з них має свої переваги, недоліки та специфіку використання.

2.2.1 Огляд платформи Android

Android є найпопулярнішою мобільною платформою у світі, яка використовується більшістю виробників смартфонів. Завдяки цьому додатки на Android можуть охопити величезну аудиторію. Крім того, платформа дозволяє гнучко налаштовувати функціонал і працює на різноманітних пристроях, починаючи від телефонів і планшетів та закінчуючи телевізорами чи смарт-годинниками. Однак така різноманітність створює і складності – потрібно враховувати різні версії ОС і характеристики пристроїв, що збільшує час тестування.

Переваги та недоліки показані на таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки платформи Android.

Переваги	Недоліки
Широка аудиторія. Завдяки великій кількості пристроїв, що працюють на Android, додатки на цій платформі охоплюють широку аудиторію.	Фрагментація. Різноманітність пристроїв і версій ОС створює складності у забезпеченні сумісності додатку.
Гнучкість. Android надає можливість розробникам глибоко налаштовувати функціонал додатку,	Час розробки. Через велику кількість пристроїв потрібно більше часу на тестування.

інтегруватися з іншими сервісами та пристроями.	
Різноманітність пристроїв. Android використовується на смартфонах, планшетах, смарт-годинниках, телевізорах тощо.	

Android залишається чудовим вибором для створення мобільних додатків, орієнтованих на велику кількість користувачів, особливо якщо важлива інтеграція з апаратними функціями пристрою.

2.2.2 Огляд платформи IOS

У свою чергу, iOS, операційна система від Apple, характеризується високою стабільністю, безпекою та зручністю розробки через невелику кількість моделей пристроїв, на яких вона працює. Для iOS простіше забезпечити якість роботи додатка, адже розробникам не потрібно враховувати таку кількість змінних, як у випадку з Android. Однак iOS є платформою з більш закритою екосистемою, і її використання потребує додаткових ресурсів, таких як Mac-комп'ютери для розробки та платна підписка на доступ до App Store. Переваги та недоліки показані на таблиці 2.2.

Таблиця 2.2 – Переваги та недоліки платформи IOS.

Переваги	Недоліки
Однорідність пристроїв. Оскільки Apple випускає обмежену кількість моделей, розробка і тестування додатків є простішими.	Закрита екосистема. Обмеження з боку Apple можуть ускладнювати розробку специфічних функцій.
Платоспроможна аудиторія. Користувачі iOS частіше купують платні додатки або роблять покупки	Дорожчий вступ. Розробка для iOS потребує використання Mac-

в додатках, що може бути важливим для монетизації.	комп'ютера та платної підписки для доступу до App Store.
Безпека. iOS забезпечує високий рівень захисту даних і конфіденційності.	

iOS є ідеальною платформою для розробки застосунків, орієнтованих на преміум-сегмент користувачів або якщо важливі стабільність та безпека.

2.2.3 Огляд технології WEB

WEB стає універсальним вибором для багатоплатформних рішень завдяки своїй доступності через браузери на будь-яких пристроях, незалежно від операційної системи. Це дозволяє створювати додатки, які не потребують розробки окремих версій для Android чи iOS, що суттєво зменшує витрати часу та ресурсів. Веб-додатки мають багато переваг, серед яких простота оновлення – всі зміни впроваджуються на сервері, і користувач одразу отримує доступ до нових функцій. Проте вони мають обмежений доступ до апаратних функцій пристрою, що іноді знижує продуктивність. Переваги та недоліки показані на таблиці 2.3.

Таблиця 2.3 – Переваги та недоліки технології WEB.

Переваги	Недоліки
Мультиплатформність. Один додаток доступний для всіх платформ через браузер, що значно економить час і ресурси на розробці.	Обмежений доступ до функцій пристрою. Веб-додатки мають обмежений доступ до апаратних функцій (наприклад, Bluetooth або push-сповіщень).
Економічність. Немає потреби розробляти окремі версії для iOS, Android чи інших платформ.	Продуктивність. Веб-додатки можуть працювати повільніше, ніж нативні рішення.

Легка підтримка. Оновлення додатка відбувається на сервері, і користувачі миттєво отримують доступ до нових функцій.	
Доступність. Користувачі можуть відкрити додаток з будь-якого пристрою, що підтримує браузер.	

Зробивши аналіз вибір пав на WEB і створення PWA .

2.2.4 Огляд технології PWA

Сучасним підходом до розробки веб-додатків є PWA. Це прогресивні веб-додатки, які поєднують переваги і нативні рішення. Вони працюють у браузері, але користувач може встановити їх на свій пристрій, як звичайний додаток. PWA підтримують офлайн-режим завдяки кешуванню даних, що робить їх ідеальними для використання в умовах нестабільного інтернету. Також вони дозволяють заощадити ресурси на розробці, адже замість створення окремих додатків для кожної платформи створюється один універсальний продукт. Проте деякі обмеження все ж залишаються, наприклад, PWA не підтримуються на всіх браузерах, а доступ до апаратних функцій, таких як камера чи Bluetooth, може бути обмежений. Переваги та недоліки показані на таблиці 2.4.

Таблиця 2.4 – Переваги та недоліки PWA.

Переваги	Недоліки
Швидкість і легкість доступу. Користувач може встановити додаток одним натисканням, без необхідності завантажувати його з App Store чи Google Play.	Обмежена функціональність. PWA не може отримати повний доступ до деяких функцій пристрою, як-от камери чи сенсорів.

Офлайн-робота. Завдяки кешуванню PWA може працювати без підключення до Інтернету.	Не підтримується всіма браузерами. Деякі функції PWA недоступні в Safari на iOS.
Мультиплатформність. Один додаток працює на всіх операційних системах через браузер.	
Менші витрати. Розробка PWA обходиться дешевше, ніж створення окремих додатків для Android і iOS.	

Вибір WEB та PWA як платформи та технології для реалізації проекту є оптимальним рішенням у сучасних умовах. Вони дозволяють швидко створювати доступні, економічні та універсальні рішення, які працюють на всіх пристроях. Такий підхід відповідає актуальним тенденціям цифровізації, де важливу роль відіграють швидкість, адаптивність і зручність для користувачів.

2.3 Язик та фреймворк для розробки

Вибір мови програмування та фреймворку є одним із ключових етапів при плануванні розробки додатка. Саме від цього рішення залежить архітектура проекту, швидкість розробки, продуктивність додатка та легкість його підтримки в майбутньому. У разі створення веб-додатка необхідно враховувати сучасні тренди, особливості кожної мови, сумісність з обраним середовищем розробки та зручність використання для команди. Далі ми розглянемо мови програмування та фреймворки, які широко використовуються для побудови веб-додатків.

2.3.1 Огляд та аналіз мови програмування JavaScript та його фреймворків

JavaScript - одна з найбільш затребуваних мов програмування у світі, яка є ключовою для сучасної веб-розробки. Спочатку створена для додавання

інтерактивних елементів на веб-сторінки, сьогодні вона активно використовується не лише у фронтенд-розробці, але й для серверної частини додатків завдяки Node.js.

JavaScript є динамічною мовою програмування, яка дозволяє змінювати структуру та стиль веб-сторінки в real-time режимі, забезпечуючи інтерактивний досвід користувача. Вона підтримується всіма сучасними браузерами, що робить її універсальною для веб-розробки. Він має широкі можливості використання, включаючи створення веб-додатків, мобільних додатків і серверної логіки, а її екосистема пропонує безліч бібліотек і фреймворків, таких як React, Angular, що дозволяє спростити процес розробки. Широка та активна спільнота , це означає що іде постійний розвиток технологій і доступ до навчальних ресурсів.

Але також є недоліки:

- одним із головних це відсутність строгої типізації, це означає, що під час розробки буде набагато більше помилок і часу на їх пошук.

- залежність від середовища виконання (браузера чи Node.js).

- різниця у реалізації деяких функцій у різних браузерах, хоча сучасні стандарти (ES6 та новіші) значно зменшили цю проблему.

JavaScript є універсальним інструментом, без якого не обходиться розробка веб-додатків. Завдяки своїй популярності та широким можливостям, JS залишається лідером у галузі фронтенд розробки.

Оглянемо фреймфорк React.

React - це JavaScript-бібліотека, що використовується для створення користувацьких інтерфейсів. Він дозволяє будувати динамічні веб-додатки за допомогою компонентного підходу, забезпечуючи зручність у розробці, підтримці та масштабуванні додатків. Це одна з найпопулярніших бібліотек серед розробників завдяки її простоті, гнучкості та високій продуктивності.

React базується на ідеї компонентного підходу, де весь додаток розбивається на окремі частини — компоненти. Кожен компонент є ізольованою частиною інтерфейсу, яка може повторно використовуватись у різних частинах додатка. Ця бібліотека дозволяє розробникам будувати складні інтерфейси з простих компонентів.

React застосовує віртуальний DOM, щоб оптимізувати оновлення користувацького інтерфейсу. Замість безпосереднього оновлення DOM у браузері. Він створює віртуальну копію DOM, яка швидко порівнюється зі станом інтерфейсу, що змінюється. Це значно прискорює оновлення інтерфейсу та знижує навантаження на браузер.

Основні особливості React полягають у використанні компонентного підходу, який дозволяє розділяти додаток на незалежні частини, що легко підтримуються і повторно використовуються. Він використовує віртуальний DOM для оптимізації процесу оновлення інтерфейсу, що забезпечує високу продуктивність за рахунок мінімізації операцій із реальним DOM. Бібліотека підтримує JSX, що дозволяє писати HTML-подібний код безпосередньо в JavaScript, роблячи розробку більш інтуїтивною. Також він реалізує принцип одностороннього потоку даних, де зміни передаються зверху вниз від батьківських компонентів до дочірніх, забезпечуючи передбачуваність роботи додатка. Що б управляти станом React використовуються власні інструменти, а також сторонні бібліотеки, такі як Redux або Context API. Також має багату екосистему, яка включає інструменти для маршрутизації, серверного рендерінгу та тестування, це робить його універсальним рішенням для веб-розробки, а переваги та недоліки представлені в таблиці 2.5.

Таблиця 2.5 – Переваги та недоліки фреймворку React.

Переваги	Недоліки
DOM React дає високу швидкість оновлення сторінок, що позитивно впливає на продуктивність.	Часті оновлення та поява нових інструментів можуть ускладнити підтримку старих проектів.
Спрощує повторне використання коду та полегшує його підтримку.	Для повноцінного функціонування складних проектів потрібні зовнішні інструменти, як-от Redux, що збільшує складність для новачків.
React інтегрується з іншими бібліотеками і фреймворками, що	Попри простоту базових концепцій, робота з великими

дозволяє використовувати його в проектах різної складності.	проектами вимагає розуміння додаткових інструментів і патернів.
Завдяки популярності React має велику кількість готових рішень і активну підтримку від розробників по всьому світу.	

Оглянемо фреймворк Angular

Angular - застосовується для створення односторінкових додатків (SPA), надаючи потужні інструменти для побудови складних і масштабованих інтерфейсів користувача. Angular є одним з найпопулярніших фреймворків серед розробників завдяки своїй масштабованості, високій продуктивності та потужній екосистемі.

Angular базується на компонентному підході, де додаток розбивається на незалежні компоненти. Кожен компонент відповідає за частину інтерфейсу і має власну логіку. Така організація коду спрощує розробку, тестування та підтримку додатків. Крім того, завдяки автоматичному двосторонньому зв’язку даних у Angular змінні автоматично синхронізуються між моделлю та інтерфейсом користувача.

Компоненти в Angular є основними одиницями для створення інтерфейсу Вони складаються з шаблону HTML стилів CSS та логіки TypeScript Це дозволяє будувати модульні додатки де кожен компонент відповідає за окрему частину інтерфейсу.

TypeScript Angular написаний на TypeScript що робить типізацію статичною і дозволяє працювати з великими кодовими базами знижуючи ймовірність помилок.

Директиви Angular використовує директиви для маніпулювання DOM Вони дозволяють змінювати поведінку елементів на сторінці такі як умови відображення ngIf цикли ngFor.

Dependency Injection Angular використовує патерн впровадження залежностей що дозволяє легко управляти інтерфейсами та сервісами в додатку.

Маршрутизація Angular надає систему маршрутизації що дозволяє створювати багатосторінкові додатки без перезавантаження сторінки.

Що б робота асинхронних операцій відбувалась в Angular, він активно використовує бібліотеку RxJS, яка дозволяє ефективно обробляти потоки даних та подій. Переваги та недоліки наведено в таблиці 2.6.

Таблиця 2.6 - Переваги та недоліки Angular.

Переваги	Недоліки
Є універсальним під любий розмір проекта.	Розмір: Завдяки великій кількості вбудованих функцій, Angular може бути важким для проектів, де потрібна лише базова функціональність.
Наявність Angular CLI дозволяє швидко створювати проекти, компоненти, сервіси, а також спрощує тестування та деплоймент.	Складність: Angular може бути складним для початківців, оскільки вимагає знань TypeScript, структури додатка та патернів проектування.
Фреймворк забезпечує просту інтеграцію з іншими бібліотеками та інструментами, такими як Redux для керування станом або Angular Material для використання готових UI-компонентів.	Часті оновлення Angular іноді можуть спричиняти проблеми з сумісністю та оновленням старих проектів.

2.3.2 Огляд та аналіз мови програмування Python та його фреймворків

«Python – це високорівнева мова програмування, яку розробив наприкінці 1980-х років Гвідо ван Россум. » [38]

Він є інтерпретованою, динамічно типізованою мовою, яка підтримує об'єктно-орієнтоване, процедурне та функціональне програмування. Завдяки своїй універсальності використовується для всіх масштабів проектів.

Python орієнтований на простоту синтаксису, що робить його доступним для новачків і водночас потужним для досвідчених розробників. Його код легкий для читання та написання завдяки мінімальній кількості синтаксичних конструкцій і використанню відступів для позначення блоків коду.

Мова підтримує кросплатформенність, що дозволяє запускати програми на різних операційних системах без потреби їхньої модифікації.

Python інтерпретована мова, отже виконання коду відтворюється рядок за рядком без необхідності попередньої компіляції, це спрощує тестування і відладку.

Мова також використовує динамічну типізацію, що дає змогу змінним змінювати типи даних в період виконання програми, забезпечуючи гнучкість у розробці.

Об'єктно-орієнтоване програмування Python підтримує об'єктно-орієнтовану парадигму де все є об'єктом Це дозволяє працювати з класами та об'єктами для створення структурованих додатків.

Звичайна версія є надзвичайно багатою та містить модулі для роботи, що значно спрощує розробку програм.

Підтримка багатопарадигмальності, підтримує різні підходи до розв'язання задач включаючи процедурне функціональне та об'єктно-орієнтоване програмування.

Модулі та пакети Python підтримує створення модулів і пакетів що дозволяє розділяти код на логічні блоки та повторно використовувати його в різних проєктах.

Синтаксична простота завдяки мінімальним синтаксичним конструкціям Python є однією з найбільш легких для вивчення мов програмування.

Оглянемо фреймворк Flask

Flask - це мікрофреймворк для веб-розробки на мові програмування Python який створений Арміном Ронахером у 2010 році. Flask є легким мінімалістичним та гнучким рішенням для створення веб-додатків. Його головна особливість — простота у використанні та розширюваність завдяки модульній архітектурі.

Flask надає лише базову функціональність, залишаючи вибір додаткових інструментів на розсуд розробника. Цей фреймворк підтримує маршрутизацію

URL, що дозволяє створювати маршрути для обробки запитів і генерації відповідей. Flask дозволяє легко інтегрувати сторонні засоби і інструменти для розробки.

Однією з ключових особливостей Flask є його простота, адже для старту роботи потрібно лише кілька рядків коду. Водночас, Flask підтримує розширення, які дозволяють додавати потрібну функціональність. Переваги та недоліки показані на таблиці 2.7.

Таблиця 2.7 - Переваги та недоліки фреймворку Flask.

Переваги	Недоліки
Дозволяє швидко розпочати розробку навіть з мінімальними знаннями завдяки простій структурі та мінімуму коду.	Flask не має вбудованих систем для автентифікації, управління користувачами чи адміністрування, що змушує розробника шукати сторонні рішення.
Не накладає жорстких обмежень на структуру проекту, дозволяючи розробникам організовувати його відповідно до власних потреб.	У великих проєктах відсутність стандартної структури може призводити до хаотичної організації коду.
Дозволяє додавати лише потрібні компоненти та бібліотеки, уникаючи надмірності у функціоналі.	Більше підходить для невеликих проектів, оскільки масштабування великих систем може вимагати додаткових інструментів.
Завдяки численним розширенням, таким як Flask-SQLAlchemy для баз даних чи Flask-WTF для форм, можна додавати необхідну функціональність у проєкт.	
Flask часто використовується для створення невеликих сервісів або API через свою легкість і гнучкість.	

Оглянемо фреймворк Django

Django - це високорівневий веб-фреймворк для Python розроблений у 2005 році. Django надає всі інструменти з акцентом на швидкість розробки чистий код та масштабованість. Фреймворк побудований на принципі "Batteries Included" що означає наявність готових рішень для найпоширеніших завдань.

Django забезпечує автоматизоване створення структури проекту, що значно пришвидшує розробку та дає змогу уникати багатьох рутинних завдань. Бази даних можуть працювати через його ORM дозволяє працювати через Python-код, замість написання SQL-запитів, забезпечуючи гнучкість і безпеку. Також включає вбудовану систему управління користувачами, яка забезпечує реєстрацію, вхід і управління правами доступу, що спрощує створення захищених веб-додатків. Система маршрутизації дозволяє легко створювати URL-адреси для сторінок і API, використовуючи регулярні вирази або інтуїтивно зрозумілі маршрути.

Django підтримує автоматичну генерацію адміністративної панелі, яка дозволяє зручно керувати базами даних і контентом без необхідності створення окремого інтерфейсу. Вбудована підтримка кешування, захисту від CSRF-атак і XSS-загроз робить додатки, створені на ньому, більш захищеними. Переваги та недоліки показані на таблиці 2.8.

Таблиця 2.8 - Переваги та недоліки фреймворку Django.

Переваги	Недоліки
Дозволяє швидко створювати веб-додатки завдяки вбудованим інструментам, таким як генерація структури проекту, ORM та адміністративна панель.	Для простих проектів Django може бути занадто складним через надлишок вбудованого функціоналу.
Є вбудовані рішення для автентифікації, роботи з формами, маршрутизації, кешування, захисту	Початківцям може бути складно освоїти Django через необхідність дотримання суворої структури проекту та вивчення додаткових

від CSRF-атак та багатьох інших завдань.	концепцій.
Захищає додаток від найпоширеніших загроз.	Швидкість розвитку: Часті оновлення Angular іноді можуть спричиняти проблеми з сумісністю та оновленням старих проектів.
Python-код спрощує написання SQL-запитів та забезпечує гнучкість.	Деякі аспекти, такі як робота з кешем чи масштабування, можуть потребувати додаткового часу та зусиль.
Підходить для створення проектів різного розміру — від невеликих сайтів до масштабних систем із високим навантаженням.	Стандартний підхід до організації проекту може не підійти для розробників, які хочуть мати більше контролю над архітектурою.

2.3.3 Огляд та аналіз мови програмування Dart та його фреймворка

Dart - це сучасна мова програмування, розроблена компанією Google у 2011 році. Вона створена спеціально для швидкої та зручної розробки багатоплатформних додатків і є основною мовою для фреймворку Flutter. Має багато особливостей, що роблять її потужним інструментом для будь якої платформи.

Dart вирізняється своїм сучасним синтаксисом, який поєднує найкращі елементи мов програмування, таких як JavaScript, Java та C++, що робить його зрозумілим і легким для вивчення. Мова забезпечує багатоплатформність, дозволяючи створювати додатки для IOS, Android, WEB та десктопів з використанням єдиного базового коду. Підтримує два типи компіляції - Ahead-of-Time для швидкої роботи на пристроях і Just-in-Time що робить швидке виконання змін у коді. Завдяки вбудованій підтримці асинхронності через async і await спрощує обробку потоків даних і роботу із серверними запитами. Мова автоматично управляє пам’яттю за допомогою Garbage Collection, що забезпечує стабільність роботи додатків. Також

підтримує як статичну, так і динамічну типізацію, що дозволяє розробникам знижувати ризик помилок у коді. Великий набір бібліотек і пакетів доступний через репозиторій pub.dev, це дає змогу інтегрувати його у різні функції, такі як мережеві запити, обробка графіки.

Оглянемо фреймворк Flutter

Flutter - фреймворк із відкритим кодом, створений у 2017 році для розробки кросплатформних додатків. Використовуючи Dart, Flutter дозволяє створювати додатки, які працюють на Android, iOS, веб-браузерах та десктопах, з використанням єдиного базового коду.

Flutter працює за принципом «одного коду для всіх платформ», що дозволяє значно зменшити час і витрати на розробку. Основною унікальною рисою є його власний графічний рушій, який рендерить UI елементи. Він не використовує вбудовані елементи системи (наприклад, кнопки Android чи iOS), а створює їх самостійно.

Цей підхід забезпечує однаковий вигляд і поведінку додатків на всіх платформах. Розробники мають повний контроль над зовнішнім виглядом і можуть створювати кастомізовані інтерфейси.

Flutter також підтримує Hot Reload, що дозволяє миттєво вносити зміни в код і бачити результат в real-time режимі, це дає можливість не перезапускати додаток. Переваги та недоліки показані на таблиці 2.9.

Таблиця 2.9 - Переваги та недоліки фреймворку Flutter.

Переваги	Недоліки
Один код базується на Dart і працює на Android, iOS, WEB та десктопах, що значно зменшує витрати на розробку.	Додатки на Flutter займають більше місця порівняно з нативними через включення графічного рушія в кожен проєкт.
Завдяки компіляції Dart у машинний код і власному	Для деяких специфічних функцій пристроїв (Bluetooth, датчики, AR) може знадобитися використання

графічному рушію Flutter забезпечує високу швидкість роботи додатків.	додаткових бібліотек чи платформо залежного коду.
Дозволяє створювати привабливі, кастомізовані інтерфейси з багатими анімаціями та інтерактивними елементами.	У деяких випадках інтеграція додатків із наявними нативними проектами може бути складною та вимагати більше часу.
Пропонує велику бібліотеку щоб легко створювати компоненти інтерфейсу, які адаптуються до потреб розробника.	Деякі аспекти, такі як робота з кешем чи масштабування, можуть потребувати додаткового часу та зусиль.

Серед представлених мов програмування та фреймворків кожна має свої сильні сторони та обмеження. Обираючи відповідний інструмент для розробки, слід враховувати специфіку завдання, вимоги до продуктивності, масштабованість, зручність у розробці та підтримку проекту. Для нашої задачі найбільш оптимальним рішенням є використання мови програмування Dart у поєднанні з Flutter.

Вибір Dart і Flutter має стратегічне значення, адже ці технології дозволяють не тільки створити продуктивний і естетичний веб-додаток, а й забезпечити можливість масштабування проекту в майбутньому. Фреймворк Flutter, будучи нативно мультиплатформеним, відкриває можливості для швидкого перенесення розробки на інші платформи, такі як мобільні операційні системи Android та iOS, а також десктопні середовища. Завдяки єдиній базі коду час на адаптацію додатку для інших платформ є мінімальним, а процес перенесення не потребує значних зусиль.

Однією з важливих переваг Dart і Flutter є їхня гнучкість у виборі технологій і можливість масштабування проекту відповідно до потреб. Починаючи з веб-додатку, можна легко інтегрувати новий функціонал, адаптувати дизайн або навіть повністю перенести систему на нову платформу без потреби створювати код із

нуля. Це дає змогу швидко реагувати на зміну вимог ринку чи розширення аудиторії користувачів.

Таким чином, обрані технології ідеально відповідають потребам поточного проекту, забезпечуючи баланс між функціональністю, швидкістю розробки та перспективами майбутнього розвитку. Це не лише задовольняє сучасні вимоги, а й дає основу для довготривалого успіху проекту в умовах швидко змінюваного технологічного середовища.

2.4 Середовище розробки

Під час вибору середовища розробки перед нами стояло багато варіантів, кожен із яких має свої переваги, недоліки та особливості використання. Це рішення залежить від цілей проекту, технологій, які планується використовувати, та специфіки продукту, який буде створюватись. Далі ми детальніше розглянемо основні варіанти середовищ розробки, які можуть бути найбільш ефективними в нашому проекті та легко інтегруються з Flutter.

2.4.1 Огляд та аналіз середовища розробки Android Studio

«Android Studio — це офіційне інтегроване середовище розробки застосунків (IDE) для Android від Google, а також потужний інструмент для ручних тестувальників.» [39]

Воно пропонує великий набір інструментів.

Надає функції автозаповнення, перевірки помилок у реальному часі та зручного рефакторингу. Візуальний редактор інтерфейсу дозволяє швидко створювати UI за допомогою перетягування елементів, що спрощує дизайн додатків.

Також оснащене інструментами для відстеження використання пам’яті, частоту кадрів (FPS) та інші важливі параметри роботи додатку. Завдяки вбудованій системі збірки Gradle розробники можуть автоматизувати компіляцію проєктів і

легко керувати залежностями. Переваги та недоліки середовища наведено в таблиці 2.10.

Таблиця 2.10 - Переваги та недоліки середи розробки Android Studio.

Переваги	Недоліки
Офіційна підтримка Google, регулярні оновлення та інтеграція з Android SDK.	Працює повільно на комп'ютерах із обмеженими ресурсами.
Великий набір інструментів.	Зосередженість лише на одній платформі, що робить IDE менш гнучкою для мультиплатформних проектів.
Інтеграція з Firebase, що спрощує впровадження бекенд-сервісів.	

2.4.2 Огляд та аналіз середовища розробки IntelliJ IDEA

Підтримує всі основні мови програмування. IntelliJ IDEA отримала велику полярність бо має здатність ефективно працювати зі складними та масштабними проектами.

Пропонує інтелектуальне автозаповнення коду, аналіз продуктивності та всі необхідні інструменти для рефакторінгу. IDE автоматично аналізує код у реальному часі, виявляючи помилки та пропонуючи варіанти їх виправлення. Android Studio підтримує інтеграцію з інструментами для взаємодії баз даних і серверною частиною, є універсальним вибором для реалізації масштабних проектів. Масштабованість дозволяє ефективно працювати з проектами, що включають тисячі файлів, а вбудовані репозиторії полегшують управління гілками та комітами. Переваги та недоліки показані на таблиці 2.11.

Таблиця 2.11 - Переваги та недоліки IntelliJ IDEA.

Переваги	Недоліки

Інтеграція з популярними мовами програмування та платформами.	Важка IDE, яка вимагає значних системних ресурсів.
Містить засоби для оптимізації та вдосконалення коду.	Платна версія для розширеного функціоналу.
Чудово підходить для реалізації масштабних корпоративних проєктів.	

2.4.3 Огляд та аналіз середовища розробки Visual Studio Code

Visual Studio Code (VS Code) - це легке, багатофункціональне текстове середовище розробки, створене компанією Microsoft. Завдяки розширенням VS Code підтримує більшість мов програмування та інструментів, є універсальним рішенням для багатьох задач.

IDE забезпечує швидкий запуск і невелике споживання ресурсів, це робить оптимальним вибором навіть для комп'ютерів із середньою продуктивністю. Розробники можуть інтегрувати у VS Code розширення для підтримки мов програмування JavaScript, Python, Dart, C++ та інших, що робить середовище універсальним. Вбудований термінал дає можливості виконувати команди у самому середовищі. Гнучка система налаштувань дозволяє кастомізувати інтерфейс, змінювати теми, налаштовувати гарячі клавіші та створювати зручне робоче середовище. Переваги та недоліки показані на таблиці 2.12.

Таблиця 2.12 - Переваги та недоліки Visual Studio Code.

Переваги	Недоліки
Легка у використанні і працює швидко навіть на комп'ютерах із середніми характеристиками.	З базової версії немає повного функціоналу для специфічних платформ (наприклад, Android).
Забезпечує високу гнучкість завдяки широкому вибору розширень та плагінів.	Більша залежність від сторонніх розширень, які іноді не стабільні.

Підтримка мультиплатформенності, що дозволяє працювати з Android, iOS, вебом і десктопом.	

Для нашого проекту більше всього підійде Visual Studio Code який є ідеальним вибором завдяки своїй універсальності, легкості та багатофункціональності. На відміну від Android Studio чи IntelliJ IDEA, VS Code не обмежується однією платформою або мовою програмування, це робить її чудовим рішенням для реалізації різноманітних проектів.

Visual Studio Code запускається швидше, займає менше місця і не перевантажує систему, що робить його зручним навіть на комп'ютерах із середніми характеристиками. Його гнучка система розширень дозволяє додати все необхідне для проекту, а вбудований термінал і інтеграція з Git прискорюють робочі процеси.

Особливо для створення веб-додатків Visual Studio Code має чудову підтримку HTML, CSS, JavaScript і Dart, це робить його чудовим вибором для роботи з Flutter, оскільки IDE підтримує всі необхідні засоби для мультиплатформеної розробки. Крім того, гнучкість налаштувань і легкий інтерфейс дозволяють зосередитися на розробці, не витрачаючи час на вивчення складного середовища.

Висновок до розділу

Був зроблений великий аналіз, в ході якого ми дізналися наступне.

Вибір WEB і PWA як платформи для розробки проекту є оптимальним завдяки їх універсальності, економічності та доступності. WEB-додатки забезпечують мультиплатформність, дозволяючи працювати на будь-яких пристроях через браузер без необхідності створення окремих версій для Android чи iOS. PWA поєднує в собі переваги веба та нативних рішень, забезпечуючи швидкий доступ, офлайн-роботу та можливість встановлення додатка на пристрій. Такий підхід

дозволяє значно зменшити витрати на розробку, забезпечити швидке оновлення та адаптивність, що відповідає сучасним вимогам до цифрових рішень.

Dart і Flutter разом створюють потужну платформу для розробки кросплатформних додатків. Dart забезпечує високу продуктивність і легкий синтаксис, а Flutter дає змогу створювати додатки з єдиним кодом для різних платформ, що зменшує витрати на розробку. Flutter стає дедалі популярнішим серед розробників, особливо в проектах, де важлива кросплатформність, привабливий інтерфейс і швидкість розробки.

Отже у цьому розділі ми систематизували всі ключові вимоги, що є основою для розробки мобільного додатку. Це дозволить побудувати рішення, яке буде ефективно вирішувати поставлені завдання, відповідати сучасним тенденціям і задовольняти потреби цільової аудиторії.

Також кожен етап розробки мобільного додатку є взаємозалежним і важливим для досягнення успіху. Дотримання чіткої структури процесу, використання сучасних технологій і врахування потреб аудиторії дозволяють створити якісний продукт, який відповідатиме вимогам ринку і потребам користувачів.

3 ЗАСОБИ РОЗРОБКИ

У цьому розділі детально розглянуто ключові бібліотеки, сервіси та технології, які використовуються для розробки проекту. Особливу увагу приділено їхнім перевагам, специфіці застосування та причинам вибору саме цих інструментів, що найкраще відповідають вимогам проекту й забезпечують ефективну реалізацію поставлених завдань.

3.1 Порівняння та вибрів бази даних

Вибір бази даних є надзвичайно важливим етапом, адже саме від нього залежатимуть швидкість роботи, стабільність, безпека, масштабованість і вартість розробки та підтримки нашого додатку.

С початку потрібно розібратися який тип бази даних нам підходять. Є два основних типа - SQL та NoSQL. Розглянемо їх більш детально.

3.1.1 Огляд і аналіз типу NoSQL бази даних

NoSQL- це нереляційні бази даних, які пропонують більшу гнучкість для зберігання даних. Вони зазвичай використовуються для великих обсягів неструктурованої інформації та в режимі реального часу.

NoSQL бази даних не мають жорсткої схеми. Дані можуть зберігатися у форматах JSON, BSON, колонках, графах чи ключ-значення. Вони забезпечують гнучке горизонтальне масштабування, що дозволяє додавати сервери для збільшення продуктивності. NoSQL бази добре підходять для взаємодії з великими обсягами неструктурованих даних. Переваги та недоліки показані на таблиці 3.1.

Таблиця 3.1 - Переваги та недоліки NoSQL типу бази даних.

Переваги	Недоліки

Добре працюють із динамічними чи неструктурованими даними.	Відсутність транзакційного контролю ACID, що призводити до неконсистентних даних та втрату ресурсу.
Легко додають нові сервери для збільшення продуктивності.	Менш зріла технологія, що вимагає більше часу для інтеграції та налаштування.
Відмінно підходять для великих обсягів даних, які змінюються в реальному часі.	Не підходять для систем із чітко визначеними зв’язками між даними.

3.1.2 Огляд і аналіз типу SQL бази даних

SQL- це реляційні бази даних, які зберігають інформацію у вигляді таблиць із чітко визначеною структурою. Ці бази даних використовують SQL як стандартну мову запитів.

SQL бази даних використовують чітку структуру збереження даних у вигляді таблиць. Дані організовані відповідно до схем, які визначають типи даних і їх зв’язки. Вони підтримують мову запитів, яка забезпечує простоту роботи з інформацією та можливість виконання складних операцій. Також відповідають принципам ACID, що гарантує надійність та коректність обробки даних. Переваги та недоліки показані на таблиці 3.2.

Таблиця 3.2 - Переваги та недоліки SQL типу бази даних.

Переваги	Недоліки
Вони гарантують точність та узгодженість інформації, що є критично важливим для фінансових, освітніх та інших систем.	Менш гнучкі для динамічних чи неструктурованих даних.

Давно використовуються, мають розвинену документацію та підтримку.	Масштабування зазвичай вертикальне (збільшення потужності одного сервера), що може бути дорожчим.
Ідеально підходять для додатків із чітко структурованими даними, де пріоритетом є надійність транзакцій і забезпечення цілісності інформації.	

Зробивши аналіз цих типів можна читко сказати що враховуючи особливості нашого проекту, SQL база даних є найкращим вибором.

У нашому додатку дані мають чітку структуру (наприклад, розклад, користувачі, сповіщення), яка добре відповідає моделі реляційних баз даних. Використання ACID гарантує коректність та надійність обробки інформації, що критично для освітнього середовища. Для нашого проекту вертикальне масштабування SQL бази буде достатнім на початкових етапах. У разі зростання навантаження можна розглянути оптимізацію або міграцію.

3.1.3 Огляд платформи Supabase

Supabase - це платформа для створення бекенд-інфраструктури, яка базується на PostgreSQL. Вона позиціонується як «відкритий аналог Firebase», але з акцентом на гнучкості та розширюваності. Supabase забезпечує розробників усіма необхідними інструментами для роботи з базами даних, автентифікацією, хмарним зберіганням і реальним часом, що дозволяє швидко створювати та запускати повноцінні додатки.

Платформа автоматично генерує REST API для таблиць, цу дає змогу швидко інтегрувати базу даних із фронтендом. Supabase має real-time режим через WebSockets, що ідеально підходить для додатків із функцією миттєвих оновлень. Платформа пропонує вбудовану систему автентифікації користувачів із

підтримкою соціальних мереж, електронної пошти та інших методів входу. Supabase включає хмарне сховище файлів із можливістю керувати доступом до даних через рольові політики. Інтуїтивно зрозуміла консоль дозволяє легко налаштовувати базу даних, створювати таблиці та працювати з тригерами. Платформа є відкритою, що дає можливість розгорнути її локально або на власних серверах для забезпечення повного контролю над інфраструктурою.

Для нашого проекту Supabase є найкращим вибором з таких причин:

- оскільки структура даних у нашому додатку є чітко визначеною, використання PostgreSQL через Supabase забезпечить стабільність, консистентність і простоту роботи з даними.

- автоматичне створення API значно пришвидшить розробку, оскільки дозволяє взаємодіяти з базою даних через REST запити без потреби писати окремий бекенд.

- підтримка реального часу забеспечить сповіщень чи оновлень у реальному часі є важливим функціоналом нашого додатку.

- у разі потреби ми можемо локально розгорнути базу даних, що забезпечить повний контроль над даними та конфіденційністю.

- також легко працює з Dart і Flutter, це робить його кращим вибором для створення сучасного кросплатформного додатку.

Supabase розробило бібліотеку для Flutter яка називається supabase_flutter.

3.2 Огляд технології Push Notification в Firebase

Push notifications - це короткі повідомлення, які надсилаються користувачам додатків у реальному часі на їхні пристрої. Вони можуть відображатися у вигляді банера, спливаючого вікна або текстового повідомлення на екрані навіть тоді, коли додаток не активний або закритий. Пуш-сповіщення дозволяють доставляти важливу інформацію прямо до користувача без необхідності відкривати додаток.

Для нашого проекту пуш-сповіщення є ключовим елементом, адже вони забезпечують швидку та ефективну комунікацію між системою і користувачами.

Сповіщення дозволяють миттєво інформувати студентів про зміни в розкладі, скасування або перенесення занять, а також про важливі події чи заходи. Пуш-сповіщення допомагають утримувати користувачів, нагадувати їм про важливі дедлайни чи інші дії, які потрібно виконати. Щоб користувачеві не потрібно було постійно перевіряти розклад чи оновлення вручну, система автоматично надсилає відповідні повідомлення. Завдяки сповіщенням наш додаток стане більш зручним і корисним, створюючи позитивний досвід для студентів і викладачів.

Firebase - це багатофункціональна платформа, яка забезпечує розробників інструментами для інтеграції пуш-сповіщень за допомогою Firebase Cloud Messaging (FCM).

Firebase пропонує готові бібліотеки для Flutter, такі як firebase_messaging і firebase_core, які значно спрощують процес інтеграції пуш-сповіщень у додаток. Завдяки добре документованим бібліотекам і прикладам реалізації, розробка пуш-сповіщень з Firebase є швидкою і зрозумілою. Також дозволяє працювати з пуш-сповіщеннями на WEB, забезпечуючи єдине рішення для усіх платформ. Підтримує як одиничні сповіщення, так і масові кампанії, а також дозволяє налаштовувати сповіщення для конкретних груп користувачів або подій. Одним із головних є забезпечення миттєвої доставки повідомлень, що особливо важливо для критично важливої інформації. Firebase є безкоштовним для базових функцій, включаючи пуш-сповіщення, що робить його доступним навіть для проектів із невеликим бюджетом.

Використання Firebase і push notifications для нашого проекту є найкращим рішенням завдяки вище перерахованим фактам.

Також, наше система стане краще за аналогічні тому що:

- користувачі, які отримують пуш-сповіщення, відкривають додаток на 60-80% частіше.

- ймовірність втрати активних користувачів зменшується на 30-40%.

- Push notifications збільшують середній час взаємодії з додатком на 88%.

У нашому додатку Push Notifications виконують такі ключові функції:

- забезпечення оперативного сповіщення студентів про зміни в розкладі, події чи новини.

- підтримка комунікації між студентами та викладачами.

- зменшення ризику пропуску важливої інформації, що підвищує загальну ефективність навчального процесу.

Без пуш-сповіщень користувачам доведеться постійно перевіряти оновлення в додатку вручну, що знижує зручність і залученість. З пуш-сповіщеннями додаток стає інтерактивним і більш цінним для кінцевого користувача, що робить його невід'ємною частиною навчального середовища.

3.3 Огляд платформи Vercel

Vercel - це хмарна платформа, що спрощує розробку, розгортання та управління сучасними веб-застосунками. Її головна мета — забезпечити безшовний досвід розробки, швидке тестування та доставку контенту користувачам. Розгортання проектів на Vercel є максимально простим: достатньо підключити репозиторій, і всі зміни автоматично будуть застосовуватися завдяки інтеграції з CI/CD (безперервна інтеграція та доставка). Це дозволяє миттєво впроваджувати оновлення та зосередитися на написанні коду, а не на налаштуванні серверів.

Vercel пропонує унікальні можливості для створення швидких і оптимізованих застосунків. Наприклад, завдяки підтримці серверного рендерінгу (SSR) та статичної генерації сторінок (SSG), платформа дозволяє досягти високої продуктивності веб-проектів. Глобальна мережа CDN забезпечує швидку доставку контенту незалежно від місця розташування кінцевого користувача. Додатково Vercel підтримує серверні функції без необхідності налаштовувати сервери, що робить її ідеальною для створення API або виконання інших бекенд задач. Функція Edge Middleware дозволяє реалізовувати кастомізовану обробку запитів на рівні серверів, що забезпечує персоналізований досвід для кожного користувача.

Vercel може взаємодіяти від простих особистих веб-сайтів до масштабованих корпоративних рішень. Завдяки інтуїтивно зрозумілому інтерфейсу та інтеграції з

популярними інструментами розробників, платформа є універсальним рішенням для створення застосунків, які легко масштабуються.

3.4 Огляд на стандарт JWT

JWT - це стандарт для передачі інформації у форматі токенів. Унікальність JWT полягає у його структурі, яка дозволяє підписувати дані, гарантувати їх достовірність і захист від модифікації. Токен складається з трьох частин: заголовка, тіла і підпису, які об'єднуються в один рядок і передаються через HTTP-запити, URL або cookies.

JWT зазвичай використовується для авторизації та аутентифікації. Після успішного входу користувача у систему сервер генерує токен та робить передачу клієнту. Цей токен додається до всіх запитів користувача для підтвердження його особи та надання доступу до захищених ресурсів. Оскільки підпис токена гарантує цілісність даних, серверу не потрібно кожного разу звертатися до бази даних, це дає змогу швидше обробляти запити .

JWT ідеально підходить для використання у мобільних застосунках, особливо для централізованого сповіщення. Наприклад, токен використовується для ідентифікації користувача та визначення його прав доступу до навчальних заходів. Дані, які зберігаються в токені, не шифруються, але вони підписуються, тому їх не можна змінити без виявлення. Це забезпечує надійну аутентифікацію і авторизацію навіть у розподілених системах, таких як сучасні вебзастосунки або мобільні сервіси.

3.5 Бібліотеки для розробки

На Flutter є безліч бібліотек які допомагають в розробці майже всіх задач. Далі я наведу всі бібліотеки які були корисні при розробці проекту.

3.5.1 Огляд бібліотеки flutter_modular

Flutter Modular - це бібліотека для організації модульної архітектури у Flutter застосунках, яка забезпечує управління залежностями, маршрутизацію та модульність компонентів. Вона створена щоб можливо було розбивати проект на логічно ізольовані частини, які легко масштабувати, підтримувати й тестувати.

Основою роботи бібліотеки є модулі, що слугують контейнерами для залежностей і маршрутів, які використовуються у межах певного функціонального блоку застосунку. Залежності автоматично додаються у потрібних місцях, забезпечуючи динамічне управління станом і ресурсами. Крім того, дозволяє створювати складні системи навігації завдяки підтримці динамічної маршрутизації.

Ця бібліотека актуальна для великих мобільних застосунків, які потребують масштабованості та структурованості. Наприклад, у багаторівневих додатках з численними екранами або функціями, розділення на модулі дозволяє уникати дублювання коду та покращує управління залежностями.

Flutter Modular вирішує проблему перевантаженості коду в великих проектах, а також складнощі із підтримкою складної архітектури. Вона спрощує процеси оновлення, додаючи гнучкість у розробку та підтримку застосунку.

3.5.2 Огляд бібліотеки flutter_screenutil

Flutter ScreenUtil - це бібліотека для адаптивного дизайну у Flutter, яка дозволяє створювати екрани, що коректно відображаються на пристроях із різними розмірами дисплея та роздільною здатністю. Вона допомагає досягти узгодженості у вигляді застосунку на різних платформах.

Основний принцип роботи бібліотеки полягає у масштабуванні розмірів елементів інтерфейсу (ширина, висота, текст, відступи) відповідно до розмірів екрана. Розробники можуть використовувати гнучкі одиниці вимірювання, які автоматично адаптуються до конкретного пристрою, зберігаючи пропорції і зручність для користувача.

Бібліотека вирішує проблему ручної адаптації дизайну під різні розміри екранів. Вона позбавляє необхідності прописувати індивідуальні стилі для кожного пристрою, роблячи процес створення адаптивного дизайну простішим і швидшим.

У мобільних додатках надзвичайно висока, особливо для створення застосунків, які повинні коректно працювати на телефонах, планшетах та інших пристроях із різною роздільною здатністю. Це дозволяє забезпечити однаковий користувацький досвід незалежно від типу пристрою.

3.5.3 Огляд бібліотеки flutter_dotenv

Flutter DotEnv - це бібліотека для управління змінними середовища у Flutter-застосунках, яка забезпечує зручне і безпечне зберігання конфіденційних даних, таких як API-ключі, URL серверів чи інші параметри конфігурації.

Основною концепцією є використання .env-файлів, де зберігаються всі змінні середовища. Бібліотека дозволяє завантажувати ці файли в застосунок під час його запуску, зчитувати значення змінних та забезпечувати різні конфігурації для середовищ розробки, тестування і продакшену.

Flutter DotEnv є актуальною для застосунків, де важлива безпека і можливість динамічного налаштування без змін у коді. Наприклад, для мобільних додатків, що працюють із зовнішніми сервісами чи API, конфігурація середовищ стає важливою частиною процесу розробки.

Ця бібліотека вирішує проблему зберігання чутливих даних у коді та спрощує процес переключення між середовищами. Завдяки їй розробники можуть забезпечити безпеку та легкість налаштування застосунку.

3.5.4 Огляд бібліотеки pointycastle

PointyCastle - це потужна бібліотека для роботи з криптографією у Dart/Flutter, яка надає інструменти для виконання шифрування, хешування, підпису і генерації

ключів. Вона підтримує широкий спектр алгоритмів, таких як AES, RSA, SHA та інші.

Основною концепцією бібліотеки є забезпечення криптографічних операцій високого рівня без необхідності складного налаштування. Розробники можуть використовувати готові алгоритми для захисту даних і забезпечення їхньої цілісності.

PointyCastle є актуальною для мобільних застосунків, що працюють із конфіденційною інформацією, такою як паролі, токени чи особисті дані користувачів. У контексті безпеки вона є важливим інструментом для реалізації безпечного зберігання і передачі інформації.

Бібліотека вирішує проблему створення власних алгоритмів шифрування або використання ненадійних сторонніх рішень. Вона забезпечує стандартизований підхід до криптографії, що значно спрощує впровадження безпеки в мобільні додатки.

3.5.5 Огляд бібліотеки basic_utils

Basic Utils - це універсальна бібліотека для виконання повсякденних завдань у Flutter, таких як форматування строк, перевірка сертифікатів, валідація даних та інші утилітарні функції. Вона значно полегшує реалізацію багатьох загальних функціональностей.

Основою бібліотеки є набір готових утиліт для роботи з криптографією, мережевими з’єднаннями та базовими маніпуляціями з даними. Це дозволяє розробникам зосередитися на логіці застосунку, уникаючи дублювання коду для стандартних операцій.

Basic Utils актуальна для застосунків, які потребують виконання стандартних, але різноманітних завдань без використання додаткових залежностей. Вона корисна для зменшення кількості коду, особливо у складних проектах.

Ця бібліотека вирішує проблему дублювання функціональності та зменшує обсяг роботи, пов'язаної з написанням базових методів. Вона також забезпечує стандартизацію у використанні утилітарних функцій.

3.5.6 Огляд бібліотеки intl

Intl — це бібліотека для роботи з локалізацією та інтернаціоналізацією у Flutter. Вона дозволяє форматувати дати, числа, валюти та текст відповідно до регіональних стандартів.

Основна концепція бібліотеки полягає у створенні перекладів і форматів, що відповідають культурним особливостям користувачів. Вона включає підтримку шаблонів для перекладу і генерацію локальних ресурсів із файлів .arb.

Бібліотека вирішує проблему створення локалізованих інтерфейсів і забезпечує правильне відображення даних, таких як дати та валюти, відповідно до регіональних стандартів. Вона спрощує процес масштабування застосунків для міжнародного ринку.

Як ми могли побачити, всі бібліотеки є актуальними, вони вирішують багато задач, тому їх в присутність у проекті значно спрощує розробку, а без деяких розробка була би не можлива.

3.6 Опис структурної схеми

Структурна схема включає наступні компоненти.

Користувач:

- ідентифікується у системі через ведення приватного ключа.
- отримує або відправляє сповіщення.

Мобільний застосунок:

- відображає інформацію (сповіщення, заходи).
- запитує сповіщення у бази даних
- обробляє і відправляє в відповідному форматі данні на базу даних

- взаємодіє з API

База даних:

- містить всі необхідні дані (користувачі, події, сповіщення).

- взаємодіє з додатком для запису та зчитування даних.

Адміністратор:

- через веб-додаток керує всією базою даних.

Додаток зможе запускатися в 3 режимах. Режим студента, де студент сможе переглядати список повідомлень, режим вчителя, де вчитель зможе відправляти ці повідомлення, а також адміністратор, де можна буде управляти базою даних.

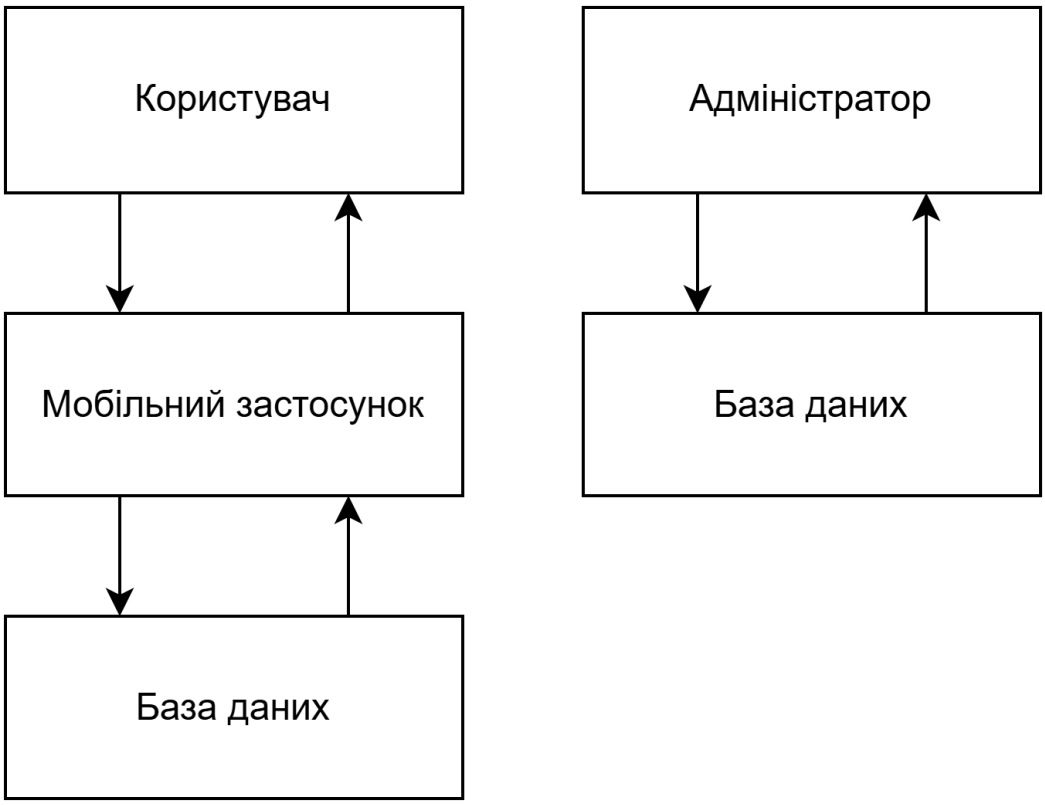


Рисунок 3.1 – Структурна схема.

Висновок до розділу

Я обрав SQL бази даних, завдяки їх структурованості, надійності та зручності роботи з транзакціями, є ідеальним рішенням для нашого проекту. Вони

забезпечують необхідну консистентність і стабільність, що особливо важливо для освітнього додатку з великою кількістю організованих даних і чіткими взаємозв'язками. Хоча NoSQL має свої переваги для інших задач, у нашому випадку SQL надає більше переваг і гарантує стабільну основу для розвитку системи.

Supabase - це ідеальне рішення для проектів, які потребують потужної реляційної бази даних із підтримкою реального часу, автоматизації створення API та надійного захисту даних. У порівнянні з аналогами, забезпечує більшу гнучкість, відкритість і простоту інтеграції, що робить її найкращим вибором для нашого додатку.

Push Notifications є важливим елементом нашого додатку, оскільки вони забезпечують оперативне сповіщення користувачів і підвищують зручність використання. Вибір Firebase Cloud Messaging як основного інструменту для реалізації цієї функції є найбільш доцільним завдяки простоті інтеграції, готовим бібліотекам для Flutter та можливостям масштабування. Це рішення дозволяє швидко та ефективно впровадити пуш-сповіщення, що відповідатимуть потребам нашого проекту та забезпечать позитивний користувацький досвід.

Кожна з бібліотек відіграє важливу роль у розробці сучасного, безпечного та адаптивного мобільного застосунку. Вони забезпечують модульність архітектури, безпеку даних, зручність роботи з інтерфейсом і підтримку користувачів із різними регіональними налаштуваннями.

4 ОПИС РЕАЛІЗАЦІЇ

Розробка мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача буде складатися з 3 частин:

1. Веб-додаток
2. База даних
3. Налаштування push notification

Основний сценарій роботи системи полягає у наступному.

Студент вводить свій приватний ключ для авторизації. Після успішного входу він отримує доступ до перегляду всіх повідомлень, які були йому надіслані.

Викладач, у свою чергу, також вводить свій приватний ключ для авторизації. Після входу він отримує можливість створювати повідомлення для студентів.

Адміністратори авторизуються в системі за допомогою приватного ключа. Після авторизації вони можуть додавати нових студентів і викладачів до бази даних, а також створювати пари ключів (приватний і публічний) для користувачів.

4.1 Веб-додаток

Почати потрібно з описа архітектури яку ми використовуємо.

4.1.1 Архітектура MVC

MVC - це архітектурний шаблон, який використовується для розділення логіки додатка на три компоненти. Такий підхід дозволяє організувати код додатка, зробити його зрозумілим, підтримуваним та зручним для розширення.

Для нашого додатку MVC є ідеальним вибором через такі причини.

Розділення додатка на модель, представлення та контролер забезпечує зрозумілу архітектуру, яка легко масштабується.Завдяки незалежності компонентів ми можемо легко оновлювати дизайн (View), змінювати логіку роботи з даними

(Model) або додавати нові функції (Controller). У майбутньому, якщо додаток розширюватиметься, MVC дозволить легко додавати нові функції без суттєвого перероблення існуючого коду. Багато популярних фреймворків (наприклад, Angular, React у поєднанні з Redux, або Flutter з відповідними патернами) підтримують принципи MVC, що робить його стандартом для сучасної розробки.

На рисунку 4.1 можна побачити скриншот архітектури с середи розробки VSCode.

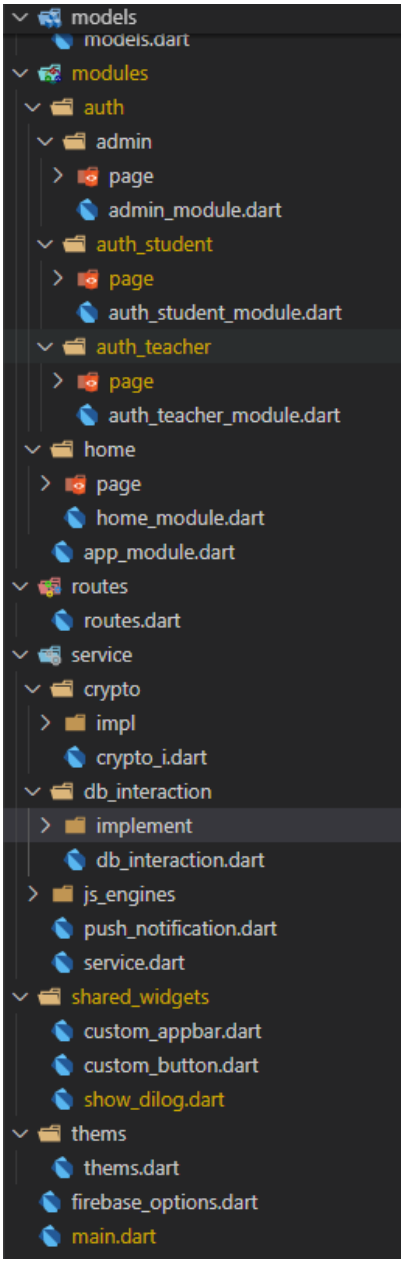


Рисунок 4.1 – Скриншот архітектури с середи розробки VSCode

У нашій архітектурі кожен екран додатку структуровано у вигляді окремих модулів, кожен із яких має свою власну сторінку. Це забезпечує логічний поділ компонентів і сприяє їх зручному управлінню. Ми також використовуємо повторно використовувані віджети, такі як app bar, кнопки та modal dialog, що допомагає зменшити дублювання коду та спрощує його підтримку.

Особливу увагу приділено маршрутизації (роутінгу), яка винесена в окрему структуру для більшої зручності та прозорості. Завдяки цьому ми легко можемо контролювати та змінювати навігацію між екранами.

Для роботи з даними впроваджені моделі, які чітко визначають структуру інформації: як тієї, що надходить до додатку, так і тієї, що передається далі. Це дозволяє впорядкувати роботу з даними та уникнути помилок у їхньому форматі.

Вся бізнес-логіка розміщена у спеціально створених сервісах. Вони відповідають за взаємодію з пуш-сповіщеннями, обробку запитів до бази даних та інтеграцію з криптографічними алгоритмами. Кожен сервіс має власний інтерфейс, що реалізує принципи ООП. Це гарантує, що ми не залежимо від конкретної реалізації сервісів і можемо легко їх замінювати або модифікувати без впливу на решту додатку.

Така структура забезпечує модульність, легкість у підтримці та масштабуванні, а також відповідає принципам чистої архітектури, що є основою для створення надійних і гнучких додатків.

4.1.2 Авторизація

Для авторизації користувача спочатку необхідно виконати аутентифікацію. Цей процес починається з того, що користувач вводить свій приватний ключ, який є унікальним ідентифікатором. На основі цього ключа створюється публічний ключ, що слугує посередником для передачі даних.

Публічний ключ передається в базу даних, де з нього формується хешований публічний ключ за допомогою JWT. Цей хешований ключ стає основою для ідентифікації користувача в системі. Після цього система перевіряє, чи існує такий користувач у базі даних, і визначає його права доступу.

Відповідно до ролі користувача відбувається авторизація в додатку. Якщо це студент, система перенаправляє його на інтерфейс студента, якщо викладач - на відповідну сторінку викладача. У випадку адміністратора користувач переходить на сторінку адміністратора.

4.1.3 Студент

Студент має право отримувати пуш-сповіщення, які автоматично надсилаються на його пристрій. Усі отримані повідомлення зберігаються в базі даних, тому користувач може в будь-який момент переглядати історію сповіщень, адресованих саме йому. Крім того, студент має доступ до перегляду інформації свого профілю, що включає особисті дані, права доступу та іншу релевантну інформацію.

4.1.4 Викладач

Викладач отримує право переглядати свій профіль, що дозволяє йому ознайомлюватися з даними, пов’язаними з його обліковим записом. Крім того, викладач може надсилати повідомлення конкретним студентським групам. Для цього він заповнює відповідні поля, зазначаючи текст повідомлення, вибір групи. Цей функціонал полегшує комунікацію між викладачами та студентами, дозволяючи швидко передавати важливу інформацію.

4.1.5 Адміністратор

Адміністратор, отримавши відповідні права доступу, має можливість повного керування базою даних. У його розпорядженні знаходяться всі операції, які відповідають принципу CRUD.

CRUD — це набір базових операцій для роботи з даними:

- create - операція додавання нових записів до бази даних, наприклад, створення нового облікового запису студента чи викладача.

- read - операція доступу до існуючих даних, наприклад, перегляд інформації профілю або отриманих повідомлень.

- update - операція зміни існуючих записів, наприклад, оновлення даних профілю або прав доступу користувача.

- delete - операція видалення записів із бази даних, наприклад, видалення користувача або видалення застарілих повідомлень.

Адміністратор може:

- видаляти, додавати та оновлювати дані студентів і викладачів, включаючи їхні облікові записи, профілі та ролі.

- керувати правами доступу, налаштовуючи ролі користувачів відповідно до їхніх обов’язків.

- генерувати пари ключів для нових користувачів.

- вносити публічні ключі до бази даних, забезпечуючи інтеграцію користувачів із системою.

Це дає адміністратору повний контроль над системою, забезпечуючи її коректну роботу та підтримку. Ключі генерувати може тільки адміністратор. Це зробленно для того щоб в додаток могли попасти тільки студенти цього закладу. Деплой як раніше було наведено проводився на Vercel.

4.2 Реалізація бази даних

База даних була розроблена під всі теперішні вимоги, а також із можливостю майбутнього масштабування.

На рисунку 4.2 зображена ER діаграма бази даних.

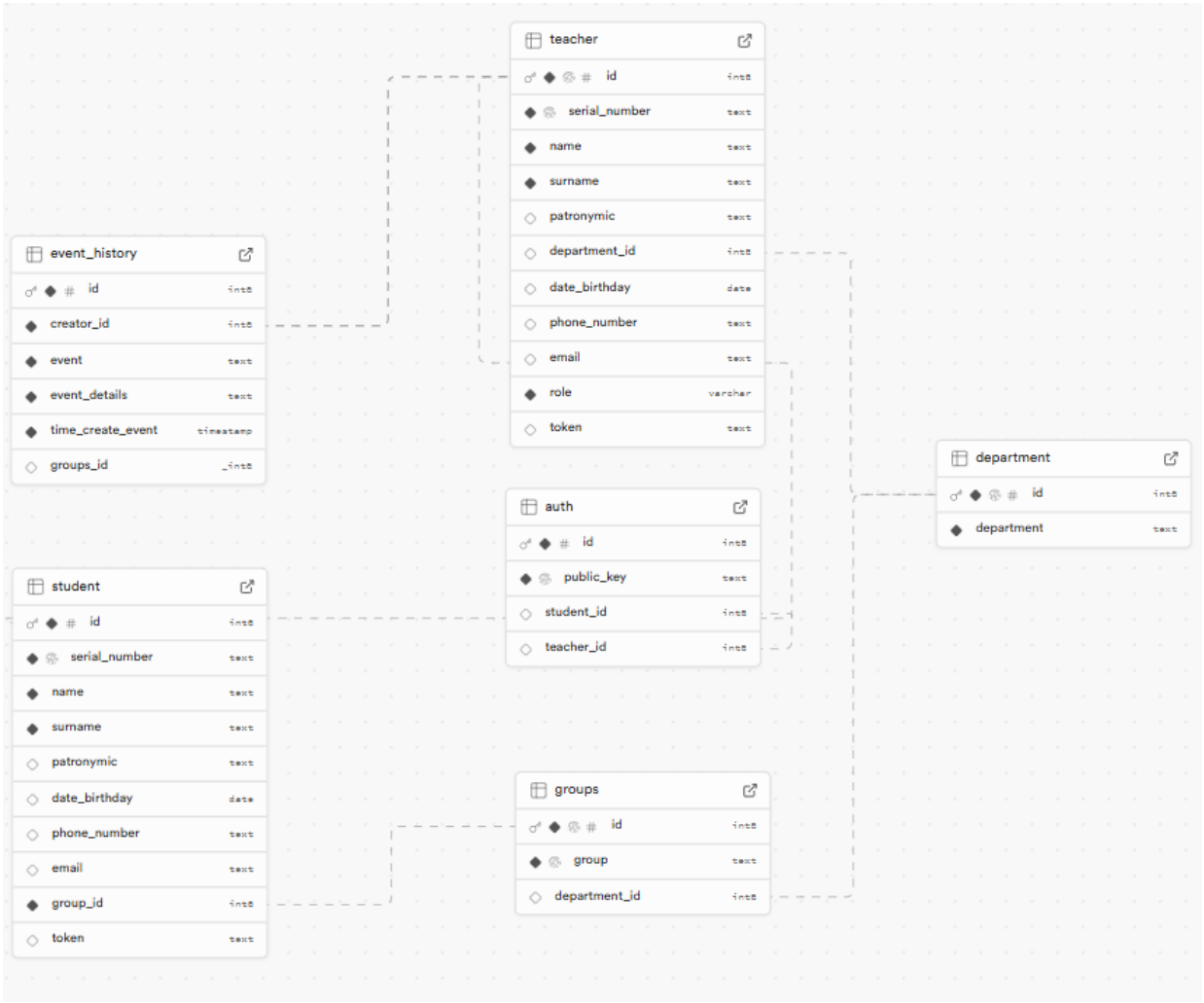


Рисунок 4.2 - ER діаграма бази даних

Таблиця teacher

Ця таблиця містить дані про викладачів. Основні поля включають:

- id (унікальний ідентифікатор викладача),
- serial_number (реєстраційний номер викладача),
- name, surname, patronymic (ім'я, прізвище та по батькові),
- department_id (ідентифікатор кафедри, з якою пов'язаний викладач),
- date_birthday (дата народження),
- phone_number та email (контактні дані),
- role (роль у системі, наприклад, викладач чи адміністратор),
- token (токен для аутентифікації).

Таблиця student

Таблиця для зберігання інформації про студентів. Вона містить:

- id (унікальний ідентифікатор студента),
- serial_number (унікальний номер студента),
- name, surname, patronymic (особисті дані студента),
- date_birthday (дата народження студента),
- phone_number та email (контактні дані),
- group_id (ідентифікатор групи, до якої належить студент).

Таблиця department

Містить інформацію про відділи або кафедри. Поля:

- id (унікальний ідентифікатор кафедри),
- department (назва кафедри або відділу).

Таблиця groups

Відповідає за групування студентів у певні академічні групи. Містить:

- id (унікальний ідентифікатор групи),
- group (назва групи),
- department_id (ідентифікатор кафедри, до якої належить група).

Таблиця auth

Призначена для зберігання автентифікаційних даних користувачів (студентів та викладачів). Поля включають:

- id (унікальний ідентифікатор запису),
- public_key (публічний ключ користувача для шифрування),
- student_id (ідентифікатор студента, якщо це студент),
- teacher_id (ідентифікатор викладача, якщо це викладач).

Таблиця event_history

Відповідає за історію подій у системі. Зберігає інформацію про створені події:

- id (унікальний ідентифікатор події),
- creator_id (ідентифікатор користувача, який створив подію),
- event (назва події),
- event_details (деталі події),
- time_create_event (час створення події),
- groups_id (ідентифікатор групи, для якої створена подія).

База даних побудована для забезпечення централізованого управління студентами, викладачами та їхньою взаємодією у системі. У таблицях передбачено поля для автентифікації через ключі (auth), управління подіями (event_history), а також організації користувачів за кафедрами (department) та групами (groups). Система дозволяє чітко розмежувати ролі користувачів (викладачі, студенти, адміністратори) та забезпечує гнучке керування сповіщеннями.

4.3 Налаштування push notification

У своєму дипломному проекті я реалізував підтримку Push Notifications за допомогою Firebase Cloud Messaging (FCM). У цьому розділі я детально опишу, як саме виконав налаштування та інтеграцію цієї функціональності.

С початку потрібно налаштувати проект та додати потрібні бібліотеки.

Я додав необхідні бібліотеки до проекту Flutter для реалізації Push Notifications. Для цього я відкрив файл pubspec.yaml та додав наступні залежності:

- firebase_core
- firebase_messaging

Потім потрібно налаштувати Firebase.

Спочатку я створив проєкт у Firebase Console та підключив свій додаток Flutter.

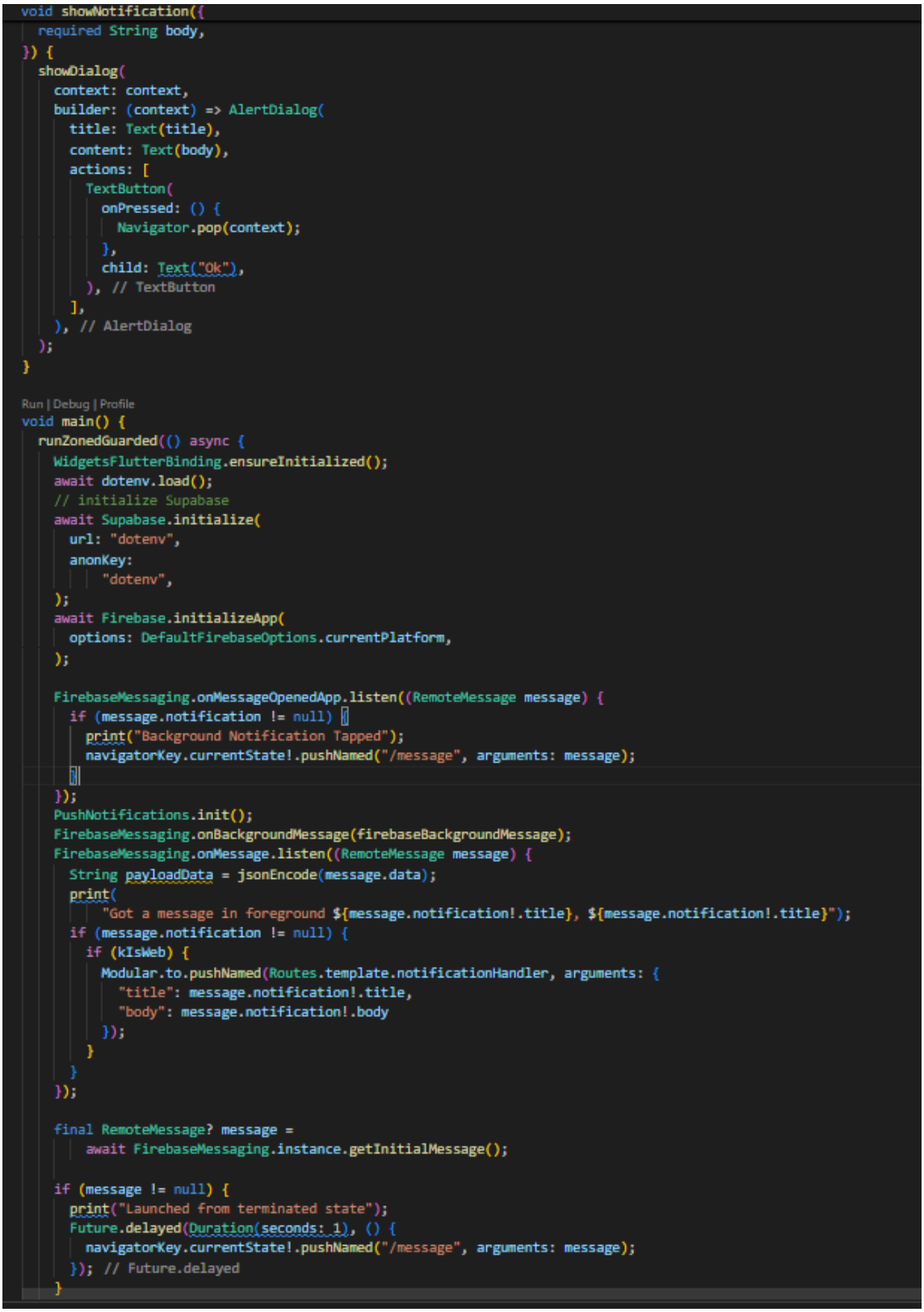
Потім потрібно ініціалізувати його і запитатий дозвіл на прийнятя повідомлень.

Функція генерації токена FCM виконується у разі авторизації користувача. Як тільки ми отримали його, в базі даних перевіється згенерований токен і токен який збережено в базі даних, я якщо вони різні, то оновлюється токен на новий.

Після цього потрібно включити обробник для повідомлень у різних режимах. Є 3 основі режима:

- background (коли додаток звернутий)
- foreground (коли додаток відкритий і на головному екрані)
- terminated (коли додаток повністю закритий)

Також для кожного обробника режима потрібно зробити свій кастомний візуальний вигляд кожного з цих повідомлень. На рисунку 4.3 зображено код ініціалізації повідомлень.



```
void showNotification({
  required String body,
}) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text(title),
      content: Text(body),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text("Ok"),
        ), // TextButton
      ],
    ), // AlertDialog
  );
}

Run | Debug | Profile
void main() {
  runZonedGuarded(() async {
    WidgetsFlutterBinding.ensureInitialized();
    await dotenv.load();
    // initialize Supabase
    await Supabase.initialize(
      url: "dotenv",
      anonKey:
          "dotenv",
    );
    await Firebase.initializeApp(
      options: DefaultFirebaseOptions.currentPlatform,
    );

    FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {
      if (message.notification != null) {
        print("Background Notification Tapped");
        navigatorKey.currentState!.pushNamed("/message", arguments: message);
      }
    });
    PushNotifications.init();
    FirebaseMessaging.onBackgroundMessage(firebaseBackgroundMessage);
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      String payloadData = jsonEncode(message.data);
      print(
          "Got a message in foreground ${message.notification!.title}, ${message.notification!.title}");
      if (message.notification != null) {
        if (kIsWeb) {
          Modular.to.pushNamed(Routes.template.notificationHandler, arguments: {
            "title": message.notification!.title,
            "body": message.notification!.body
          });
        }
      }
    });

    final RemoteMessage? message =
        await FirebaseMessaging.instance.getInitialMessage();

    if (message != null) {
      print("Launched from terminated state");
      Future.delayed(Duration(seconds: 1), () {
        navigatorKey.currentState!.pushNamed("/message", arguments: message);
      }); // Future.delayed
    }
```

Рисунок 4.3 - Код ініціалізації повідомлень

Також код повідомлення в режимі foreground зображено на рисунку 4.4.

```
unknown, last week | 1 author (unknown)
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
      unknown, last week • add: push notification
unknown, last week | 1 author (unknown)
class Message extends StatefulWidget {
  const Message({super.key});

  @override
  State<Message> createState() => _MessageState();
}

unknown, last week | 1 author (unknown)
class _MessageState extends State<Message> {
  Map payload = {};
  @override
  Widget build(BuildContext context) {
    final data = ModalRoute.of(context)!.settings.arguments;
    if (data is RemoteMessage) {
      payload = data.data;
    }

    return Scaffold(
      appBar: AppBar(title: Text("Your Message")),
      body: Center(child: Text(payload.toString())),
    ); // Scaffold
  }
}
```

Рисунок 4.4 - Код повідомлення в режимі foreground

4.4 Опис діаграми прецедентів

На рисунку 4.5 представлена діаграма прецедентів для веб-додатку, де показані основні функціональні можливості системи. У діаграмі зображені актор (користувач системи) та пов'язані з ним прецеденти (функціональність системи).

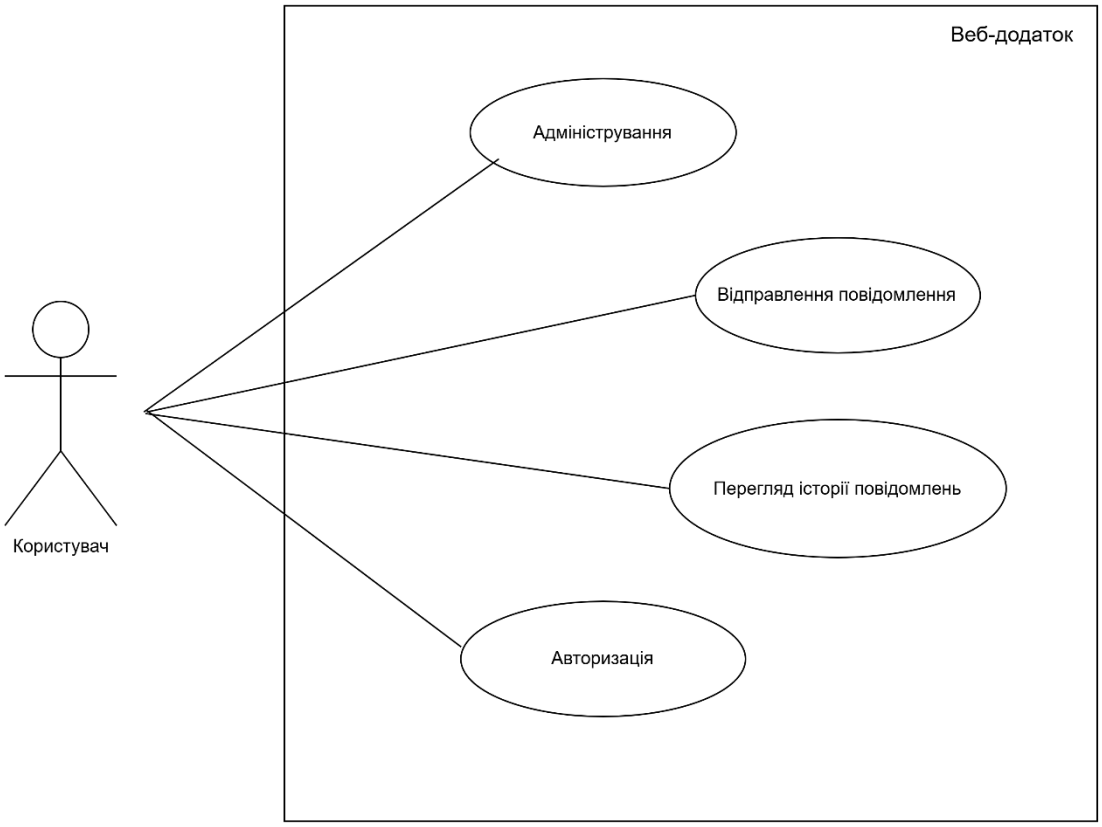


Рисунок 4.5 – Діаграма прецедентів

Актор:

Користувач (зліва) — представлений у вигляді кола, яке символізує користувача системи. Це суб'єкт, який взаємодіє з веб-додатком.

Прецеденти:

Адміністрування

Цей прецедент вказує на можливість керування системою. Наприклад, це може включати додавання або видалення користувачів, налаштування параметрів веб-додатку тощо.

Відправлення повідомлення

Цей прецедент представляє функцію надсилання повідомлень. Користувач може відправляти повідомлення через веб-додаток, наприклад, іншим користувачам чи системі.

Перегляд історії повідомлень

Ця функція дозволяє користувачу переглядати історію раніше відправлених або отриманих повідомлень.

Авторизація

Прецедент "Авторизація" вказує на можливість входу в систему за допомогою логіна та пароля або іншого механізму автентифікації.

Ця діаграма прецедентів наочно демонструє основний функціонал веб-додатку, де користувач має можливість адмініструвати систему, надсилати повідомлення, переглядати їхню історію та проходити процес авторизації для доступу до системи.

4.5 Опис схеми алгоритма авторизації

На рисунку 4.6 наочно продемонстровано процес авторизації користувача в системі. Вона складається з послідовних кроків, які включають введення ключів, перевірку на відповідність у базі даних та отримання доступу або відмови. Далі детально опишемо кожен етап.

Процес починається з того, що користувач вводить свій приватний ключ у відповідному полі додатку. Це є початковим етапом авторизації.

З приватного ключа система генерує публічний ключ. Це відбувається за допомогою криптографічних методів, де приватний ключ використовується як основа для створення публічного ключа.

Згенерований публічний ключ передається до бази даних для подальшої перевірки.

У базі даних публічний ключ хешується за допомогою JWT. JWT використовується для безпечного збереження ключів та автентифікації користувача.

Хешований публічний ключ звіряється з ключами у таблиці auth, яка містить дані зареєстрованих користувачів. Це необхідно для перевірки, чи існує введений ключ у системі.

Відбувається перевірка результату звірки:

- так. Якщо користувач знайдений у системі, виконуються наступні кроки.

- ні. Якщо користувач не знайдений, надсилається відповідь про помилку, що користувач не існує.

Якщо користувач знайдений, з відповідних таблиць у базі даних вибираються дані, що стосуються цього користувача. Ці дані включають права доступу, роль користувача, налаштування тощо.

Додаток отримує дані користувача та на їх основі надає відповідні права доступу. Користувач отримує дозвіл увійти в систему.

Якщо користувача не знайдено, додаток виводить повідомлення про те, що користувач не знайдений, і процес авторизації завершується.

4.6 Опис схеми алгоритма роботи додатка

Нижче буде приведений короткий опис даної схеми і її візуалізація.

На рисунку 4.7 відображений основна схема алгоритма роботи мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача. Також нижче буде доданий стислий опис даної схеми.

Алгоритм починається з авторизації користувача за допомогою приватного ключа. Після успішного входу користувач автоматично потрапляє на відповідний екран, залежно від його ролі в системі. Кожна роль має свої функції:

Студент:

Можливість переглядати інформацію про повідомлення.

Вчитель:

Можливість надсилати повідомлення відповідним групам.

Адміністратор:

Можливість редагувати дані в базі даних.

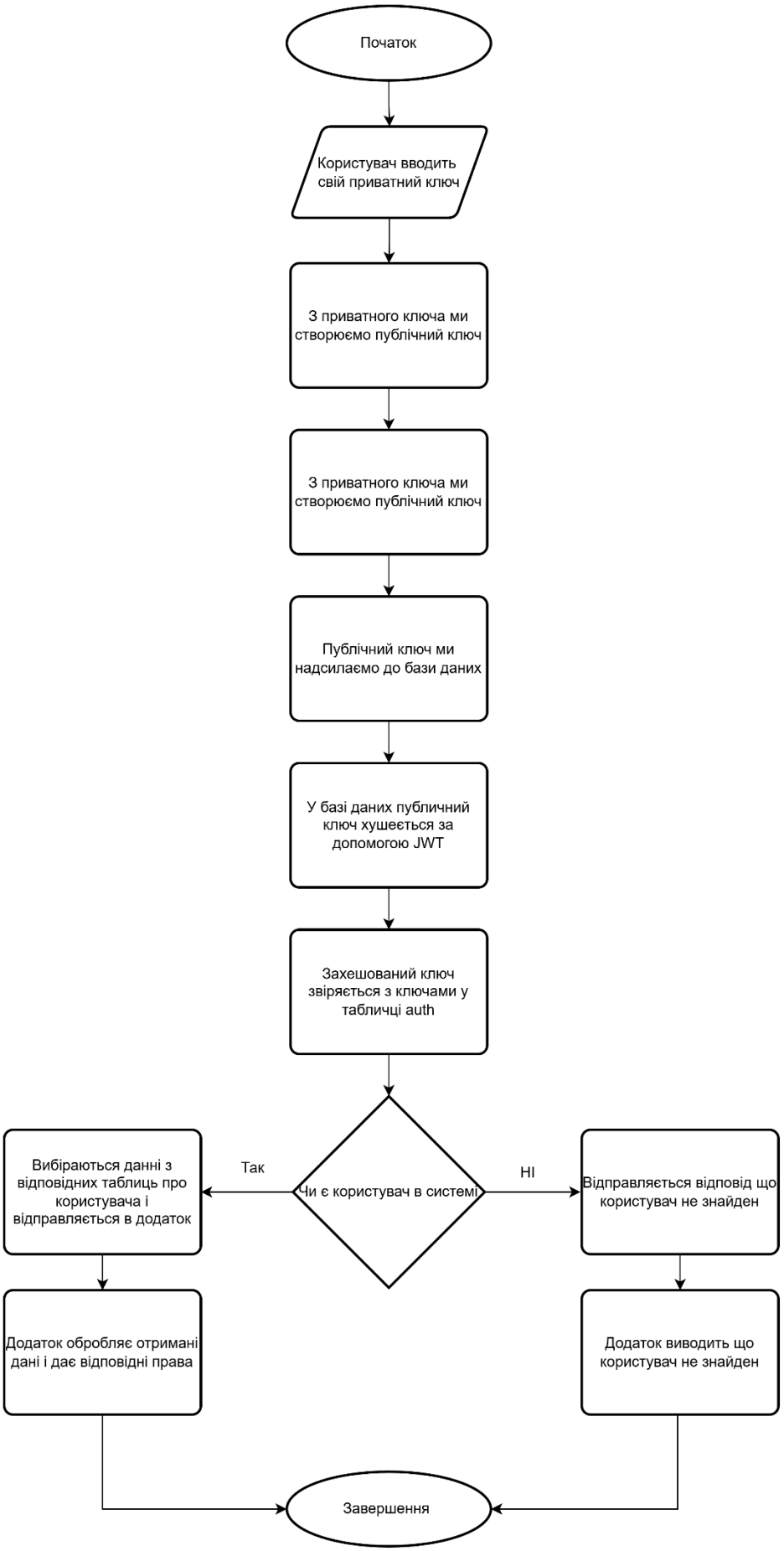


Рисунок 4.6 – Схема алгоритму авторизації

Рисунок 4.7 - Схема алгоритму роботи мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача

4.7 Тестування

З початку потрібно розповісти про головний інструмент який ми використовуємо під час тестування - ngrok.

ngrok - це потужний інструмент, який створює тунель між вашим локальним сервером і публічно доступним URL-адресом. Завдяки цьому можна протестувати локальний додаток на різних пристроях без необхідності налаштовувати хостинг або публічний сервер.

Першим етапом ми створили WEB-версію додатка, використовуючи Flutter.

Щоб протестувати додаток на смартфоні в режимі реального часу, ми використали ngrok.

За допомогою згенерованого ngrok-URL ми відкривали веб-додаток на смартфоні. Це дозволило перевірити всі доступні функції додатка на мобільному пристрої, оцінити продуктивність, адаптивність інтерфейсу та зручність користування.

Загалом як показало себе тестування усі розробляємі функції працюють так як і було розраховано в самому початку роботи.

Висновок до розділу

У даному розділі було детально оглянуто весь процес розробки.

Додатково було представлено обрану архітектуру і її реалізацію, також як і базу даних, так процес отримання та відправлення повідомлень.

Також було оглянуто и описано наступні схеми:

- діаграми прецедентів.
- схема алгоритма авторизації.
- схема алгоритма роботи мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача

Було оглянуто процес тестування у ході розробки.

5 ПРОБЛЕМИ ТА ПОКРАЩЕННЯ СИСТЕМИ

5.1 Проблеми

На початку розробки планувалося, що користувачі будуть здійснювати авторизацію в додатку за допомогою індивідуальних карток (наприклад, студентських або інших карт, які підтверджують належність до навчального закладу). Ідея полягала в тому, що користувач зможе відсканувати NFC-мітку з картки за допомогою Web NFC API прямо через веб-додаток на своєму смартфоні.

5.1.1 Огляд технології NFC

NFC - це технологія бездротового зв'язку ближнього радіусу дії, яка дозволяє обмінюватися даними між пристроями на відстані до 10 см. NFC є підвидом RFID (радіочастотна ідентифікація), але працює у дуже обмеженому радіусі, що робить її безпечнішою для передачі конфіденційних даних.

Простота у використанні, оскільки для обміну даними достатньо просто наблизити два пристрої або пристрій і мітку один до одного. Технологія є енергоефективною, оскільки пасивні NFC-мітки не потребують власного живлення й активуються електромагнітним полем зчитувача. При цьому NFC забезпечує швидкість передачі даних до 424 кбіт/с, чого цілком достатньо для обміну невеликими обсягами інформації. Обмежена дальність дії (до 10 см) робить технологію безпечнішою, знижуючи ризики несанкціонованого зчитування, а також дозволяє використовувати її для конфіденційних операцій, таких як платежі чи авторизація.

Типи NFC-взаємодії:

- Peer-to-Peer (P2P)
- зчитування або запис міток.
- емуляція картки.

5.1.2 Огляд веб-інтерфейса Web NFC API

Web NFC API – це сучасний веб-інтерфейс, який дозволяє веб-додаткам взаємодіяти з NFC-мітками через браузер на мобільному пристрої. Web NFC API надає можливість зчитувати та записувати дані на NFC-мітки без необхідності встановлення нативних додатків.

Web NFC API дозволяє веб-додаткам працювати з NFC-мітками без необхідності встановлення додаткових додатків чи використання нативних функцій платформи. Основною перевагою є те, що користувач може зчитувати й записувати дані на NFC-мітки безпосередньо через веб-браузер, що відкриває нові можливості для веб-застосунків, таких як швидка авторизація, перевірка ідентифікації, обмін інформацією та інтеграція з IoT-пристроями.

Однак Web NFC API має значні обмеження: технологія наразі підтримується лише на Android пристроях у браузері Chrome, що суттєво звужує охоплення користувачів, оскільки iOS та інші браузери не забезпечують необхідної підтримки. Однак API підтримує лише певні типи NFC-міток, зокрема NDEF (NFC Data Exchange Format), що звужує можливості взаємодії з іншими форматами. До того ж, через обмежену дальність дії NFC (до 10 см) та безпекові вимоги, Web NFC API має обмежений сценарій використання, особливо для масштабованих чи високонавантажених застосунків.

Отже на етапі розробки була ідея інтегрувати можливість авторизації користувача за допомогою NFC-карток, що дозволило б швидко ідентифікувати його причетність до навчального закладу. Ця функція планувалася на основі Web NFC API, який дозволяє зчитувати й записувати дані на NFC-мітки безпосередньо через веб-браузер. Однак під час реалізації з'ясувалося, що підтримка NFC у веб-середовищі є досить обмеженою.

Web NFC API наразі працює виключно на Android-пристроях у браузері Chrome, що значно обмежує його застосування, оскільки iOS та інші популярні браузери не підтримують цей стандарт. До того ж, технологія сумісна лише з певними типами

NFC-міток, такими як NDEF (NFC Data Exchange Format), тоді як більшість сучасних NFC-міток використовують інші формати. У той же час, було проведено тестування функціоналу в нативному середовищі для Android та iOS, де NFC працював стабільно й без обмежень.

Але тестуваня в браузері через WebView показало погані що бразуери ще не мають змоги працювати зі всіма функціоналами NFC.

У результаті постало важливе рішення: обрати нативний підхід для Android та iOS з повною підтримкою NFC, або ж віддати перевагу веб-підходу, який забезпечує кросплатформенність і охоплює ширшу аудиторію користувачів. Для ухвалення оптимального рішення було проведено детальний аналіз, щоб визначити, що важливіше для користувачів: швидка авторизація через NFC або ж доступність веб-додатку на різних пристроях.

Зрештою, перевага була надана веб-версії додатку завдяки її універсальності та можливості охопити більше користувачів незалежно від пристрою. Хоча підтримка NFC у вебі наразі відсутня, функціонал взаємодії з NFC-мітками залишився на рівні концепції. У майбутньому, коли Web NFC API отримає повноцінну підтримку й вирішить свої обмеження, цей функціонал буде додано до веб-додатку.

5.2 Покращення

Якщо цей продукт виявиться корисним для користувачів та зможе залучити широку аудиторію, у майбутньому планується значне розширення його функціоналу. Серед потенційних можливостей — індивідуальний чат з викладачем, що дозволить студентам оперативно отримувати відповіді на свої питання, повний розклад занять для зручного планування навчального процесу, а також перегляд успішності студента, що надасть доступ до оцінок та аналітики навчальних досягнень. Окрім цього, буде додано багато інших корисних функцій, які підвищать ефективність та комфорт використання продукту.

Особлива увага приділятиметься візуальній частині проекту. Планується створення сучасного та зрозумілого інтерфейсу, що забезпечить зручну навігацію

та приємний користувацький досвід. Для покращення загальної естетики будуть додані анімації та інші елементи візуального дизайну, що зроблять роботу з додатком більш привабливою та динамічною.

В репозиторій можна перейти за QR кодом який зображено на рисунку 5.1.

Рисунок 5.1 – Посилання на репозиторій

Висновок до розділу

NFC (Near Field Communication) - це сучасна технологія, яка надає широкі можливості для авторизації, безконтактних платежів і зчитування даних із пристроїв. Веб-версія цієї технології, Web NFC API, перебуває на ранньому етапі розвитку та наразі має певні обмеження у функціональності та підтримці платформ. Проте її потенціал у сфері веб-додатків є значним. Очікується, що з розвитком браузерів і стандартів функціонал Web NFC API стане ширшим, а його сумісність із різними платформами значно покращиться, що відкриє нові можливості для інтеграції цієї технології в майбутньому.

Окрім цього, у майбутньому планується розширення функціоналу додатку для покращення комунікації та зручності. Одним із важливих доповнень стане інтеграція чату між студентами та викладачами, що створить більш тісний зв'язок і спростить обмін інформацією.

Також можливим є впровадження повноцінного доступу до академічних таблиць, що відображатимуть оцінки та відвідуваність студентів. Це дозволить студентам детально аналізувати свій прогрес у навчанні. Додатково планується додати функціонал перегляду списку дзвінків, пар і сесій, що ще більше спростить організацію навчального процесу.

Важливим аспектом є оновлення візуальної частини додатку. Планується модернізація дизайну до сучасних стандартів із використанням анімацій та плавних переходів між сторінками. Це покращить користувацький досвід і зробить взаємодію з додатком більш приємною.

Поточний дизайн уже оптимізований під усі можливі розширення екранів, що дозволяє запускати додаток на будь-якому пристрої. Незалежно від розміру чи роздільної здатності екрану, інтерфейс залишається коректним і стабільним, без зміщення чи інших візуальних проблем. Це робить додаток універсальним і доступним для користувачів із різними пристроями.

6 ВЗАЄМОДІЯ КОРИСТУВАЧА З PWA ДОДАТКОМ

Для того щоб почати працювати з додатком вам потрібен будь який прилад який може працювати з браузерами. Також потрібен безперебійний інтернет шо б мати змогу завантажувати та відкривати додаток.

У самому початку у вас з’явиться запит на дозвіл відправляти вам повідомлення, ви повинні прийняти його. Це стандартна процедура дозволу користування браузером та додатком функціоналу вашого приладу. На рисунку 6.1 зображений запит на дозвіл відправки повідомлень.

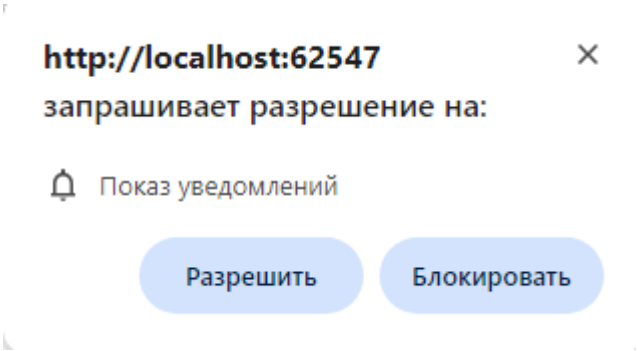


Рисунок 6.1 - Запит на дозвіл відправки повідомлень

6.1 Головний екран

Після цього буде доступна головна сторінка додатку. На ній потрібно нажати на кнопку Login. На рисунку 6.2 зображена головна сторінка додатку.

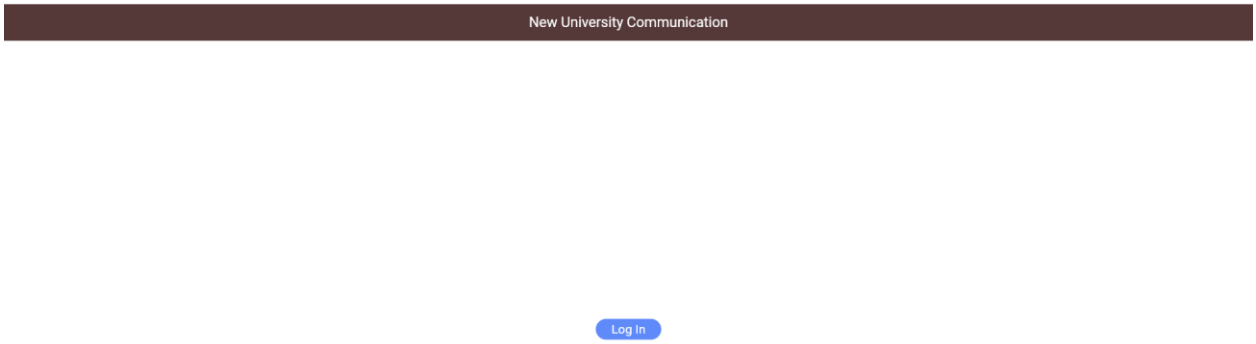
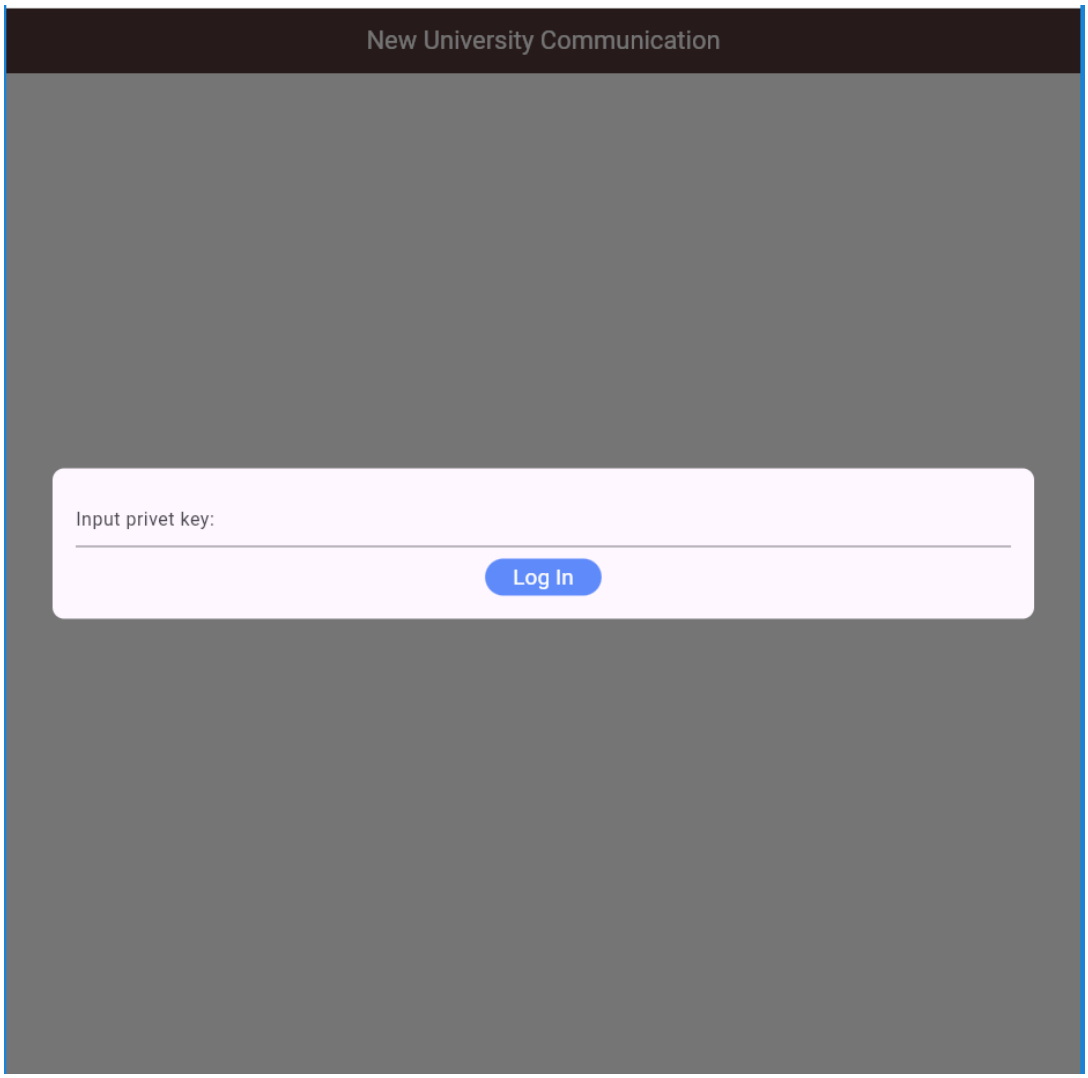


Рисунок 6.2 - Головна сторінка додатку



Рисунку 6.3 - Модальне вікно Log In

Після того як ви натиснули на кнопку, у вас з’явиться модальне вікно, на якому буде поле вводу. Туди потрібно вставити ваш приватний ключ, після чого натиснути на кнопку Log In. На рисунку 6.3 зображено модальне вікно Log In.

6.2 Екран вчителя

Далі якщо ви вчитель, у вас зайде на екран вчителя. На рисунку 6.4 зображено початкову сторінку вчителя.

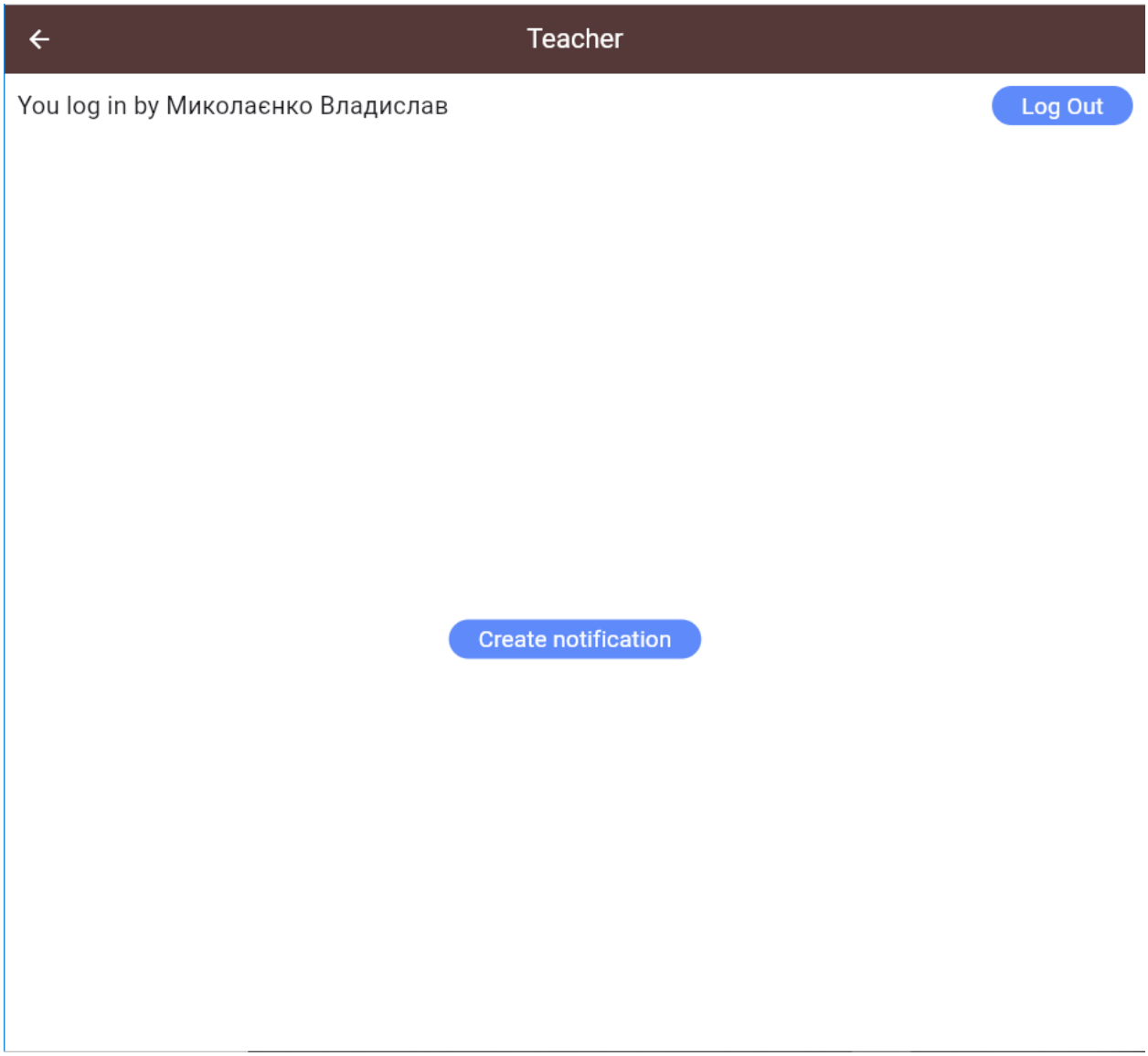


Рисунок 6.4 - Початкова сторінку вчителя

Як ми можем побачити, на цій сторінці представлена мінімальна інформація про профіль вчителя, також з права з верху розташована кнопка Log Out яка дає можливість вийти з профіля. А також кнопку створення повідомлення, яка називається Create notification. Якщо натиснути на цю кнопку, з’явиться мадальне вікно, в якому ми зможемо створити повідомлення. На рисунку 6.5 зображено модальне вікно для створення повідомлення.

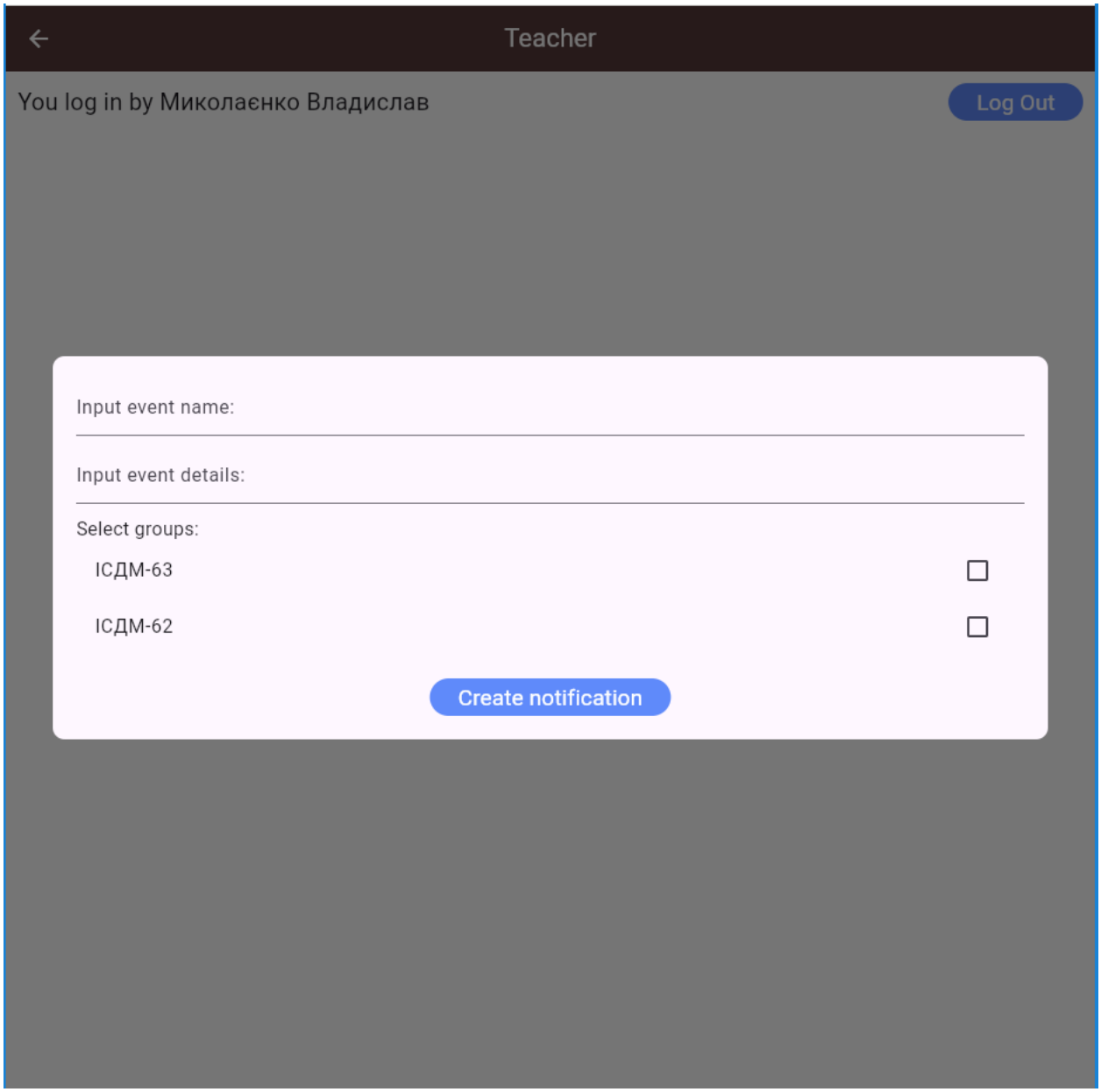


Рисунок 6.5 - Модальне вікно для створення повідомлення

Як ми можемо побачити тут є 2 текстових поля, так один чекбокс.

Потрібно заповнити всі представлені поля. Поле event name повинно містити заголовок повідомлення, головну частину, щоб студент одразу розумів про що буде іти мова, наприклад назва заняття, або назва заходу. Потім іде частина event details, тут ви можете прописати детальну інформацію про захід, наприклад, де і коли буде приходити якійсь захід, що для нього потрібно и т.д. Далі потрібно вибрати групи які отримають дане повідомлення. Приклад повністю заповнених полів відображан на рисунку 6.6. Потім потрібно натиснути кнопку Create notification, після чого всі студенти які навчаються в обраних групах отримують повідомлення.

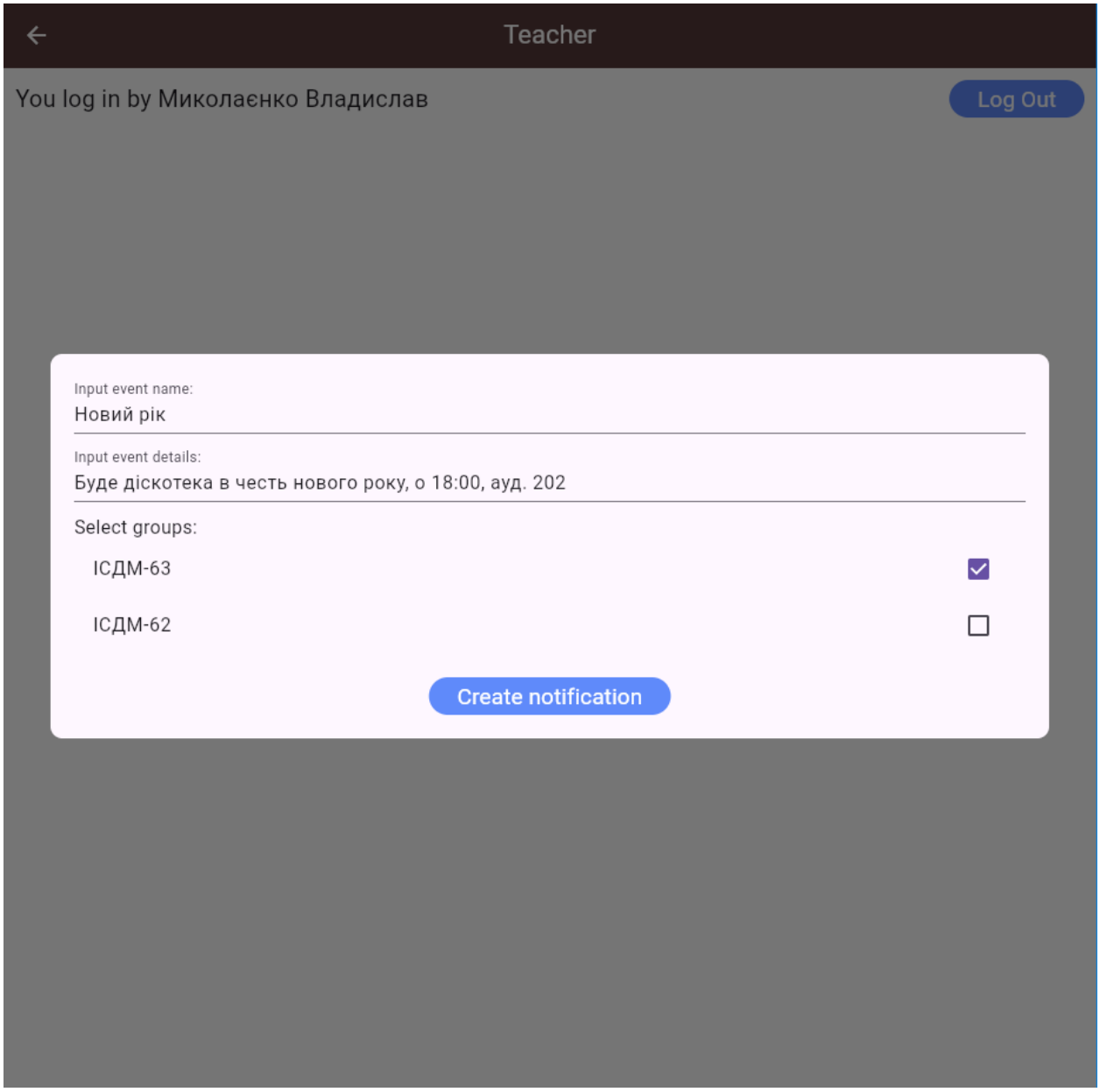


Рисунок 6.6 - Приклад повністю заповнених полів для створення повідомлення

Якщо повідомлення було відправлено успішно, ми отримаємо повідомлення яке зображено на рисунку 6.7.

Рисунку 6.7 – Приклад повідомлення про успішну відправку повідомлення

6.3 Екран студента

Далі якщо ми вводим приватний ключ який належить студенту, нас перекидає на екран студента. Головний екран студента зображено на рисунку 6.8.

Як ми можемо побачити, на головному екрані у нас представлена, мінімальна інформація про профіль студента, також з права з верху розташована кнопка Log Out яка дає можливість вийти з профіля. По центру розташований список повідомлень які були відправленні студенту. З цього списку ми можемо побачити наступну інформацію, назву, детальний опис, час створення, викладача і кафедру якій створив повідомлення. Отже всю основну інформацію яка потрібна для розуміння що саме хотіли сказати в цьому повідомлені.

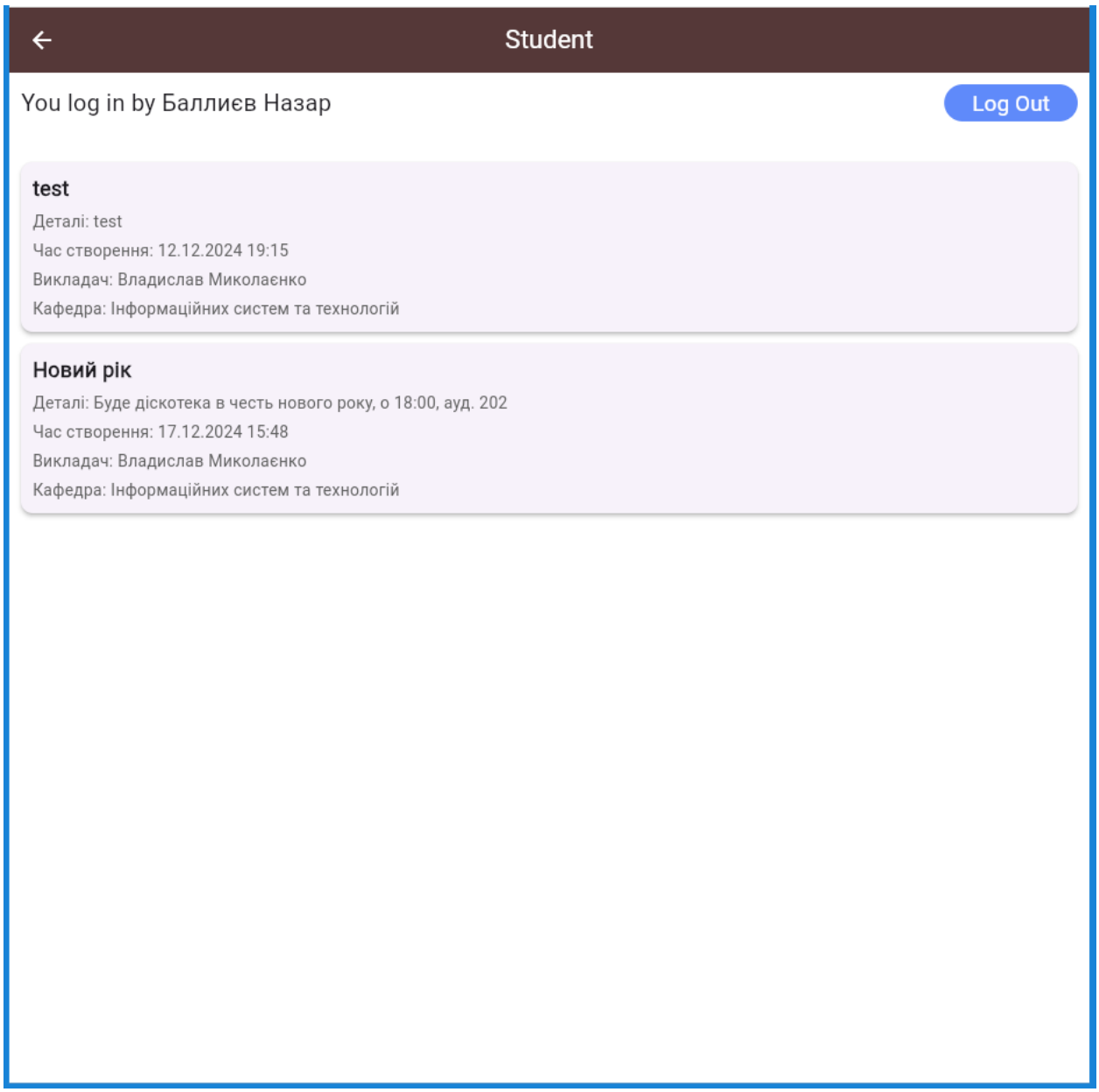


Рисунок 6.8 - Головний екран студента

6.4 Екран адміністратора

Також якщо ми ввели приватний ключ який належить до адміністратора нас перекине на екран адміністратора. Головна сторінка адміністратора зображена на рисунку 6.9.

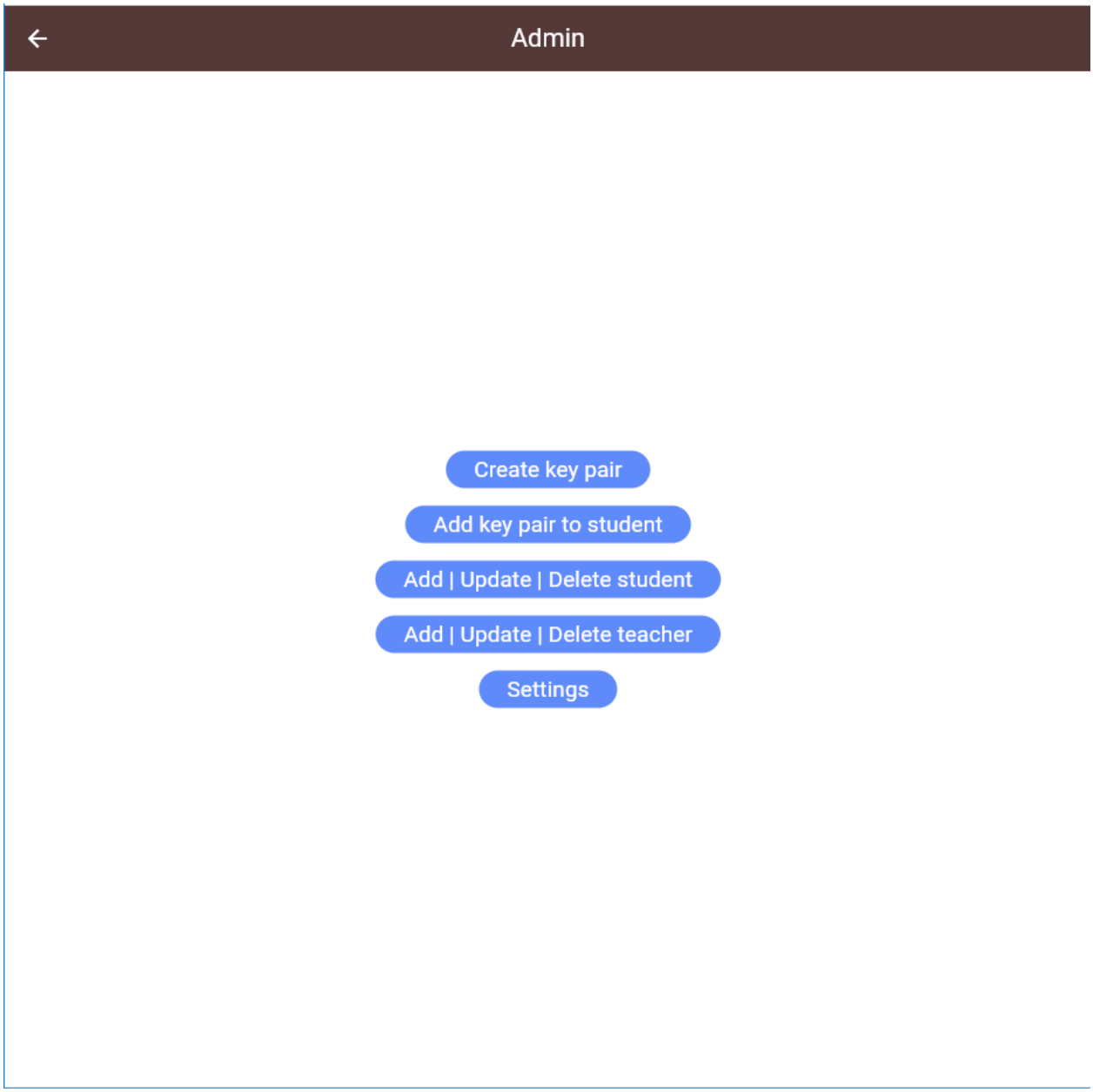


Рисунок 6.9 - Головна сторінка адміністратора

На екрані адміністратора, ми можемо побачити що в нього є основні 5 функцій і відповідні кнопки до них. Далі я розповім про них більш детально.

Кнопка Create key pair створює пару ключів, приватний та публічний, створення ключів може зайняти деякий час, тому що це доволі складний обчислювальний процес.

Також є кнопка Add key pair to student. За допомогою неї можна створити і одразу присвоїти пари ключів до вибраного користувача.

Ще є кнопка Add | Update | Delete student. За допомогою цієї кнопки, можна додавати, видаляти або оновлювати данні про студентів.

Кнопка Add | Update | Delete teacher працює так само как із студентами тільки з вчителями.

Кнопка Setting може змінити будь які налаштування в базі даних.

6.5 Отримання повідомлення

Далі я продемонструю в якому вигляді ми отримаємо повідомлення при різних режимах роботи додатка. Демонструвати роботу я буду з комп'ютера.

Для першого тесту у нас буде таке повідомлення, яке зображено на рисунку 6.10.

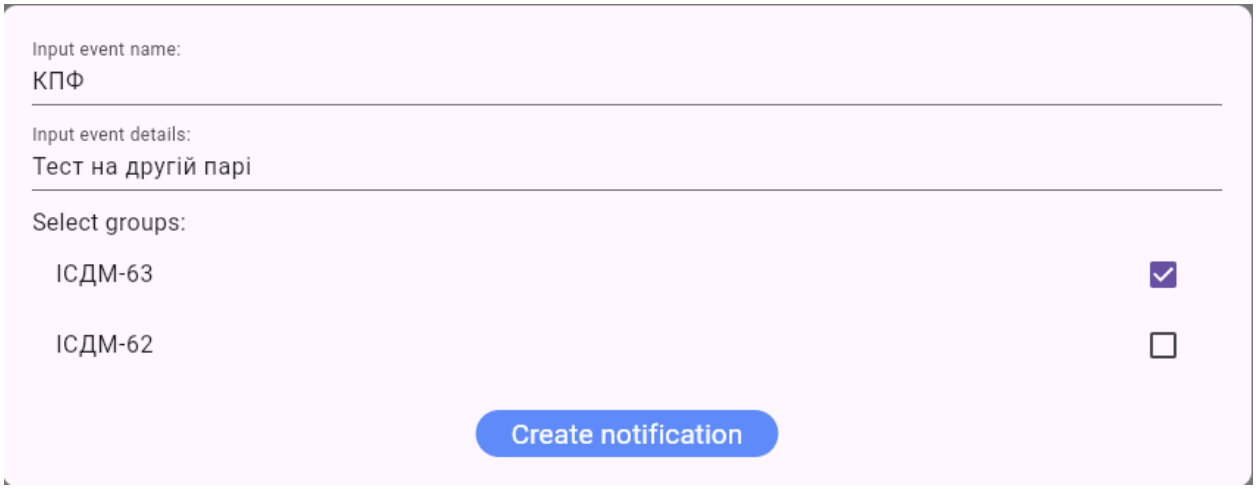


Рисунок 6.10 – Приклад першого тестового повідомлення

І так, якщо додаток в нас знаходиться в відкритому режимі, ми отримаємо повідомлення яке зображено на рисунку 6.11.

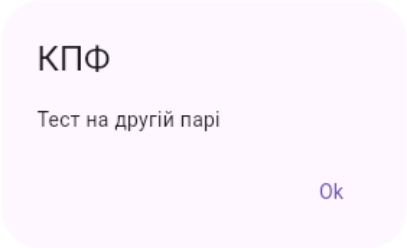


Рисунок 6.12 – Відображення повідомлення в розгорнутому режимі

Після натискання на ОК, на поверне на ту сторінку де ми знаходились. Список з повідомленнями також оновлюється. Побачити це можна на рисунку 6.12.

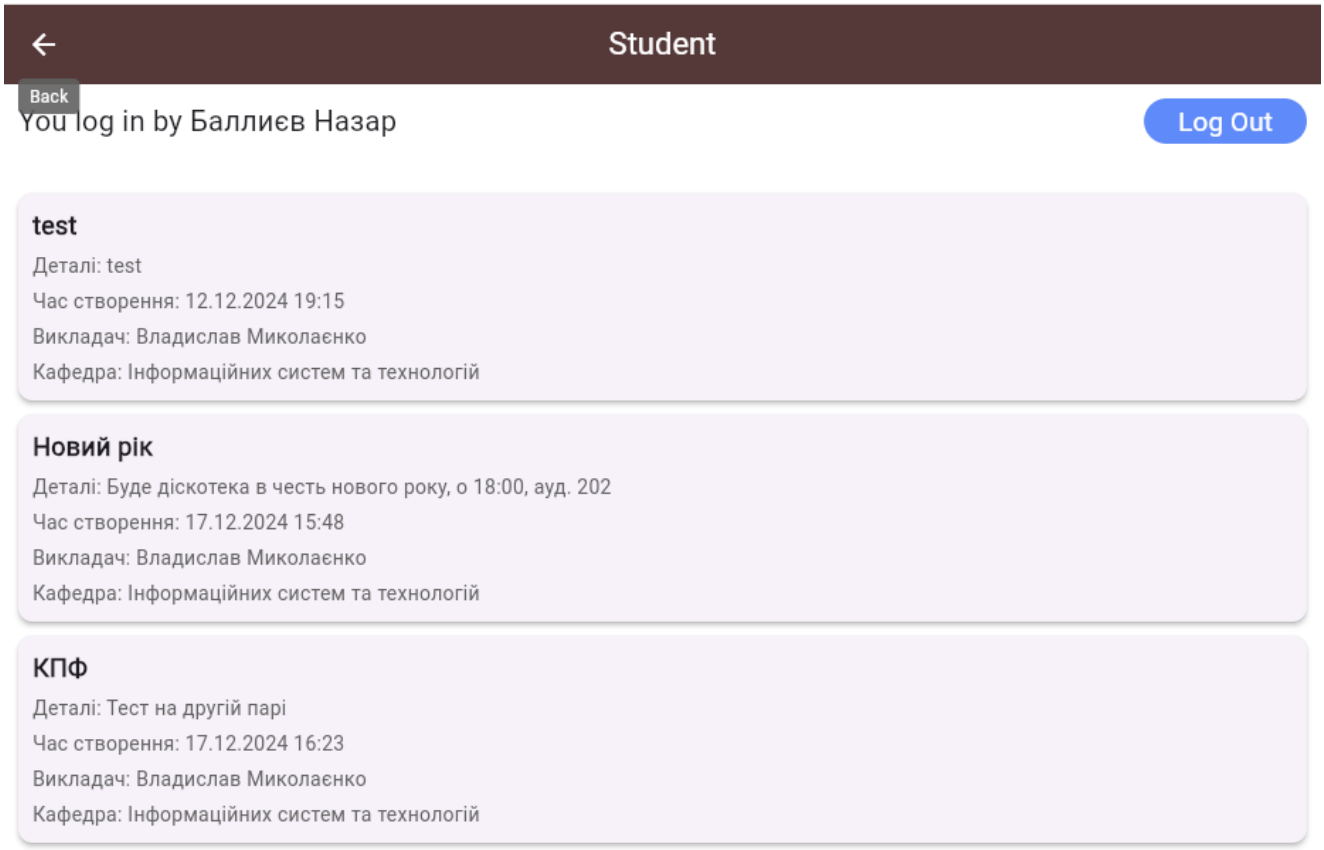


Рисунок 6.12 – Оновлений список після повідомлення

Також повідомлення можна отримати якщо додаток знаходиться в звернутому режимі. Продемонстровано на рисунку 6.13. Він відображається в правому нижньому кутку екрана. Так само він відображається якщо додаток повністю закритий.

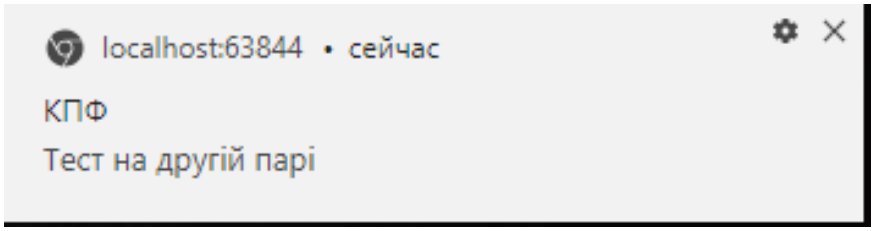


Рисунок 6.13 – Відображення повідомлення в звернутому або закритому режимі

Висновок до розділу

У цьому розділі було показано детально основу частину роботи додатка.

А саме взаємодію з :

- авторизація
- вчителем
- адміністратором
- студентом

ВИСНОВОК

Головна мета магістерського проекту є розробка мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача.

У першому розділі розповідається про важливість мобільних додатків у вирішенні освітніх завдань, їх потенціал для покращення управління навчальними процесами та створення ефективної комунікації між студентами, викладачами й адміністрацією. Також висвітлено перспективи розробки централізованого додатка для сповіщень, який забезпечить персоналізований підхід, оперативність та зручність у використанні.

У другому розділі розповідається про вибір платформи для розробки проекту, зокрема, переваги використання WEB і PWA як оптимального рішення для створення універсального, доступного та економічного продукту. Також висвітлюються можливості Dart і Flutter як інструментів для кросплатформної розробки, що дозволяють зменшити витрати, прискорити процес створення додатка та забезпечити високоякісний користувацький досвід. Розділ систематизує ключові вимоги до розробки мобільного додатка та наголошує на важливості дотримання чіткої структури й використання сучасних технологій для створення продукту, який задовольняє потреби ринку й користувачів.

У третьому розділі розглянуто вибір SQL баз даних для забезпечення структурованості, консистентності та надійності, що є критично важливим для освітнього додатку з великою кількістю взаємозалежних даних. Пояснюється, чому Supabase став оптимальним рішенням завдяки підтримці реального часу, автоматизації API та легкій інтеграції. Також висвітлено важливість Push Notifications для оперативного інформування користувачів і обґрунтовано вибір Firebase Cloud Messaging як ефективного, масштабованого інструменту. Розділ акцентує увагу на ролі бібліотек, які сприяють створенню безпечного, адаптивного та функціонального мобільного додатку, орієнтованого на сучасні потреби користувачів.

У четвертому розділі детально описано весь процес розробки додатку, включаючи обрану архітектуру, її реалізацію та базу даних, а також механізми отримання та відправлення повідомлень. Представлено та проаналізовано ключові схеми, такі як діаграма прецедентів, алгоритм авторизації та алгоритм роботи мобільного застосунку для централізованого сповіщення студентів. Окрему увагу приділено процесу тестування, який проводився на різних етапах розробки, забезпечуючи якість та відповідність функціоналу поставленим вимогам.

У п’ятому розділі описано перспективи розвитку проекту, зокрема впровадження нових функцій і покращення існуючого функціоналу. Розглянуто можливості NFC для авторизації та передачі даних, а також обмеження Web NFC API, який наразі має обмежену підтримку, але в майбутньому може стати потужним інструментом для веб-додатків. Планується додати такі функції, як чат між студентами та викладачами, розширені таблиці оцінок і відвідуваності, список пар і сесій, що значно покращить користувацький досвід і функціональність додатку. Окрема увага приділяється оновленню візуальної частини, зокрема створенню сучасного інтерфейсу з анімаціями та плавними переходами, адаптованого для будь-яких розширень екранів. Завдяки цьому додаток буде зручним для використання на будь-якому пристрої, забезпечуючи високу якість візуального відображення без зміщень чи помилок.

У шостому розділі детально розглянуто основну частину роботи додатку, зокрема взаємодію користувачів із системою. Описано ключові аспекти авторизації, а також особливості взаємодії з різними ролями в додатку: вчителем, який може надсилати повідомлення групам; адміністратором, який керує даними та додає нових користувачів; та студентом, який отримує повідомлення й має доступ до навчальної інформації.

Поставлену задачу виконано, мобільного застосунку для централізованого сповіщення студентів про навчальні заходи на основі ідентифікації користувача був повністю побудовано.

Один з основних аспектів чому даний додаток краще за аналоги, це являється присутність Push notification. Його наявність вирішує наступні питання:

- забезпечення оперативного сповіщення студентів про зміни в розкладі, події чи новини.

- підтримка комунікації між студентами та викладачами.

- зменшення ризику пропуску важливої інформації, що підвищує загальну ефективність навчального процесу.

Наявність пуш-сповіщень значно збільшує охоплення користувачів. Статистично, додатки з пуш-сповіщеннями мають у 3-5 разів вищу залученість, ніж ті, які їх не використовують.

СПИСОК ЛІТЕРАТУРИ

1. Vercel Documentation URL: https://vercel.com/docs (дата звернення: 17.12.2024).
2. Open source SQL Database Supabase URL: https://supabase.com/database (дата звернення: 17.12.2024).
3. Send and receive notifications for a Flutter app using Firebase URL: https://firebase.google.com/codelabs/firebase-fcm-flutter (дата звернення: 17.12.2024).
4. Кращі IDE для Python в 2023 році URL: https://mate.academy/blog/python/ide-for-python-2023 (дата звернення: 17.12.2024).
5. Eclipse, NetBeans чи IntelliJ IDEA? URL: https://javarush.com/ua/groups/posts/4027-eclipse-netbeans-ili-intellij-idea-vihbiraem-ide-dlja-java-razrabotki (дата звернення: 17.12.2024).
6. IntelliJ IDEA – the Leading Java and Kotlin IDE URL: https://www.jetbrains.com/idea (дата звернення: 17.12.2024).
7. Впровадження Flutter у Tide URL: https://fwdays.com/event/cto-fwdays-2023/review/bringing-flutter-in-the-world-of-finances (дата звернення: 17.12.2024).
8. Найпопулярніші інструменти для тестування мобільних додатків URL: https://aw.club/global/uk/blog/most-popular-tools-for-mobile-testing (дата звернення: 17.12.2024).
9. Web NFC API - Web APIs - MDN Web Docs URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_NFC_API (дата звернення: 17.12.2024).
10. NFC Data Exchange Format (NDEF) - Technical Documentation URL: https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/libraries/nfc/ndef/index.html (дата звернення: 17.12.2024).
11. Модель-вид-контролер URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80 (дата звернення: 17.12.2024).
12. ngrok documentation URL: https://ngrok.com/docs/ (дата звернення: 17.12.2024).

13. Flutter documentation URL: https://docs.flutter.dev/ (дата звернення: 17.12.2024).
14. Visual Studio Code documentation URL: https://code.visualstudio.com/docs (дата звернення: 17.12.2024).
15. Python documentation URL: https://docs.python.org/3/ (дата звернення: 17.12.2024).
16. Flask documentation URL: ‘https://flask.palletsprojects.com/en/stable/ (дата звернення: 17.12.2024).
17. Django documentation URL: https://docs.djangoproject.com/en/5.1/ (дата звернення: 17.12.2024).
18. React Getting Started URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 17.12.2024).
19. What is Angular? URL: https://angular.dev/overview (дата звернення: 17.12.2024).
20. Android URL: https://uk.wikipedia.org/wiki/Android (дата звернення: 17.12.2024).
21. IOS URL: https://uk.wikipedia.org/wiki/IOS (дата звернення: 17.12.2024).
22. WEB URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1 (дата звернення: 17.12.2024).
23. WEB 2.0 URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1 (дата звернення: 17.12.2024).
24. Push API URL: https://developer.mozilla.org/en-US/docs/Web/API/Push_API (дата звернення: 17.12.2024).
25. Flutter Modular URL: https://modular.flutterando.com.br/docs/flutter_modular/start/ (дата звернення: 17.12.2024).
26. Flutter screenutil URL: https://pub.dev/packages/flutter_screenutil (дата звернення: 17.12.2024).
27. Flutter dotenv URL: https://pub.dev/packages/flutter_dotenv (дата звернення: 17.12.2024).
28. DotEnv In Flutter URL: https://medium.com/@omartaamallah4/dotenv-in-flutter-e27c45a2f7ed (дата звернення: 17.12.2024).
29. Pointycastle URL: https://pub.dev/packages/pointycastle (дата звернення: 17.12.2024).

30. Basic utils URL: https://pub.dev/packages/basic_utils (дата звернення: 17.12.2024).
31. Intl URL: https://pub.dev/packages/intl (дата звернення: 17.12.2024).
32. Що таке JavaScript? URL: https://landinglist.com.ua/yak-navchytysya-programuvaty-na-javascript/ (дата звернення: 17.12.2024).
33. Як розробити медичний застосунок для лікарів: Ключові особливості, які варто врахувати URL: https://stfalcon.com/uk/blog/post/medical-app-for-doctors (дата звернення: 17.12.2024).
34. Вступ до Python URL: https://epkmoodle.znu.edu.ua/pluginfile.php/117292/mod_resource/content/1/Python%20%231.pdf (дата звернення: 17.12.2024).
35. Рішення IoT для Розумного Міста URL: https://pandateam.net.ua/razrabotka-iot-smart-city/ (дата звернення: 17.12.2024).
36. Після цього можна розпочати розрЗбірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької. – Одеса: "Гельветика", 2023. – 213 с.обку мобільного додатку. URL: https://csc.knu.ua/media/filer_public/cb/02/cb02ab8d-4eff-42c9-b2cc-2335b42fecd7/zbirnik_2023.pdf (дата звернення: 17.12.2024).
37. PyCharm як найкраща IDE для розробки ПЗ на Python URL: https://foxminded.ua/pycharm-tse/ (дата звернення: 17.12.2024).
38. Мова програмування Python та її застосування в різних галузях URL: https://foxminded.ua/de-vykorystovuietsia-python/#:~:text=Python%20%E2%80%93%20%D1%86%D0%B5%20%D0%B2%D0%B8%D1%81%D0%BE%D0%BA%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0%20%D0%BC%D0%BE%D0%B2%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F,%D1%80%D1%96%D1%88%D0%B5%D0%BD%D1%8C%20%D1%80%D1%96%D0%B7%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D1%83%20%D1%82%D0%B0%20%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F. (дата звернення: 17.12.2024).

39. Найпопулярніші інструменти для тестування мобільних застосунків: функції та випадки використання URL: https://aw.club/global/uk/blog/most-popular-tools-for-mobile-testing (дата звернення: 17.12.2024).