Міністерство освіти і науки України
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Кафедра прикладних інформаційних систем

122 Комп'ютерні науки
Освітня програма «Прикладне програмування»

Кваліфікаційна робота бакалавра

на тему:

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ФАКУЛЬТАТИВНИХ ЗАНЯТЬ В УНІВЕРСИТЕТІ

Виконав студент 4 курсу групи ПП−42

_____
(Підпис)
Попадинець С.Р.
(прізвище, ім'я, по батькові)

_____
Керівник к.т.н., д.Силантьєв С.О.
(прізвище, ім'я, по батькові)

_____
_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л. _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021 року

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	24.05.2021	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	24.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

АНОТАЦІЯ

Дипломна робота: 61 с., 19 рис., 2 табл., 26 джерел, 1 дод.

Тема: Мобільний застосунок для організації факультативних занять в університеті.

Цю дипломну роботу присвячено проектуванню та розробленню мобільного застосунку для організації факультативних занять в університеті та вивченню необхідної для цього теоретичної та практичної бази.

Метою дипломної роботи є ефективна організація факультативних занять в університеті.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Дослідити загально-теоретичні засади організовування факультативних занять в університеті та систем навчання (дистанційного, мобільного тощо); сучасні підходи до розроблення і впровадження мобільних застосунків навчання.
- Здійснити аналіз архітектурних рішень і програмних засобів для реалізації мобільних застосунків для навчання, надати їх переваги та недоліки.
- Спроектувати, реалізувати, впровадити мобільний застосунок для організації факультативних університетських занять.

Об'єкт дослідження: навчальний процес, який стосується факультативних дисциплін.

Предмет дослідження: мобільний застосунок для організації факультативних університетських занять.

Методи дослідження: теорія навчального процесу, системний аналіз і синтез, аналогія, метод порівняння, теорія побудови мобільних застосунків тощо.

Ключові слова: навчальний процес, мобільний застосунок, React Native, факультативна дисципліна.

ABSTRACT

Thesis: 61 p., 19 figs., 2 table, 26 sources, 1 appendix.

Mobile application for organizing elective classes at university.

This thesis is devoted to the design and development of a mobile application for the organization of elective classes at university and the study of the necessary theoretical and practical basis.

The purpose of the thesis is an effective organization of elective classes at the university.

To achieve this purpose it is necessary to solve the following tasks:

• Investigate the general theoretical principles of organizing elective classes at university and learning systems (remote, mobile, etc.); modern approaches to the development and implementation of mobile applications for learning.

• Analyze architectural solutions and software for the implementation of mobile applications for learning, provide their advantages and disadvantages.

• Design and implement, a mobile application for the organization of optional university classes.

Object of research: educational process related to optional disciplines.

Subject of research: mobile application for the organization of elective classes at university.

Research methods: theory of educational process, system analysis and synthesis, analogy, method of comparison, theory of construction of mobile applications, etc.

Keywords: educational process, mobile application, React Native, optional discipline.

ЗМІСТ

ВСТУП

Навчальний процес закладів вищої освіти - це система організаційних, методичних і дидактичних заходів, які спрямовані на реалізацію відповідного освітнього рівня згідно з державними стандартами освіти та Національної рамки кваліфікації, яка структурована за компетентностями опису кваліфікаційних рівнів освіти.

Актуальність дипломної роботи. Бакалаврський рівень - це перший рівень освіти, який характеризується рівнем складності відповідної освітньої професійної програми, у якій розкрито, які саме має досягти студент компетентності загальні та спеціальні й відповідні, пов’язані з ними програмні результати. У кожній освітній програмі її структурним елементом є перелік компонент та їх логічна послідовність. У цьому переліку надають обов’язкові компоненти та вибіркові компоненти згідно її структурно-логічної схеми. У навчальному процесі згідно переліку структурних компонент формують навчальний план підготовки здобувачів вищої освіти, у якому вказують графік навчального процесу, план навчального процесу та факультативні дисципліни з відповідними кредитами і навчальними годинами. Саме для факультативних дисциплін присвячено дослідження, пов’язане з проектуванням і розробленням мобільного застосунку як для викладачів, так і для студентів з метою поглиблення знань, умінь і навичок, у тому числі цифрових.

Взаємодії між студентами і викладачами завдяки мобільному застосунку навчання сприятимуть процесу навчання упродовж життя. Тому дослідження є актуальним і потрібним для громадян цифрової планети на основі використання сучасних інформаційно-комунікаційних технологій.

Метою дипломної роботи є ефективна організація факультативних занять в університеті.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Дослідити загально-теоретичні засади організовування факультативних занять в університеті та систем навчання

(дистанційного, мобільного тощо); сучасні підходи до розроблення і впровадження мобільних застосунків навчання.

- Здійснити аналіз архітектурних рішень і програмних засобів для реалізації мобільних застосунків для навчання, надати їх переваги та недоліки.
- Спроектувати, реалізувати, впровадити мобільний застосунок для організації факультативних університетських занять.

Об’єкт дослідження: навчальний процес, який стосується факультативних дисциплін.

Предмет дослідження: мобільний застосунок для організації факультативних університетських занять.

Методи дослідження: теорія навчального процесу, системний аналіз і синтез, аналогія, метод порівняння, теорія побудови мобільних застосунків тощо.

Структура роботи: Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1. ЗАГАЛЬНО-ТЕОРЕТИЧНІ ЗАСАДИ ТА ПІДХОДИ ДО РОЗРОБЛЕННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ НАВЧАННЯ

1.1 Засади розроблення мобільного застосунку для факультативного навчання

Сьогодні мобільні застосунки використовують в різних предметних сферах, у тому числі у навчальному процесі. Для розроблення таких мобільних застосунків проектні менеджери, програмісти мають в розпорядженні різні платформи, залежно від цілей навчального процесу і його особливостей. Як правило освітній процес базований на принципах науковості, гуманізму, демократичності, безперервності та ступеневості освіти. При цьому навчальний процес орієнтовано на формування освіченої, гармонійно розвинутої особистості, здатної до оновлення знань на постійній основі, професійної мобільності та адаптації в умовах цифрового суспільства.

Коли потрібно прийняти рішення, яка ж інформаційна технологія використовуватиметься для розроблення мобільного застосунку з необхідною функціональною частиною як для викладача, так і студента, то слід ретельно оцінювати всі підходи, які охоплюють передусім веб-технології, наприклад, прогресивні веб-застосунки (Progressive Web Apps), гібридні (Hybrid) рішення, нативні (Native) та крос-платформні (Cross-platform) підходи.

Кожен із вищезазначених має свої переваги та недоліки. Оскільки мобільні застосунки залежать від мети програми, кожен підхід варто розглянути. Кожен крок оцінки всіх факторів є надзвичайно важливим для надання викладачам та студентам дружнього інтерфейсу та гнучкості при користуванні застосунком.

Для викладачів і студентів у більшості випадків веб-мобільні застосунки швидші та менш трудомісткі у сенсі розробки, ніж нативні рішення. Особливо, якщо проект має невеликий набір функцій, який пов'язаний з нативними телефонними модулями.

Прогресивні веб-застосунки (*p*rogressive *w*eb *a*pplication, PWA) - це тип веб-застосунків, який створюється за звичайними технологіями, такими як HTML, CSS та JavaScript тощо (рис.1.1). Вони призначені для роботи на будь-якій платформі, яка використовує стандартний браузер (наприклад, Chrome або Mozilla Firefox) для десктопних та мобільних пристроїв. PWA розширює загальнодоступні веб-застосунки нативними функціями телефону.

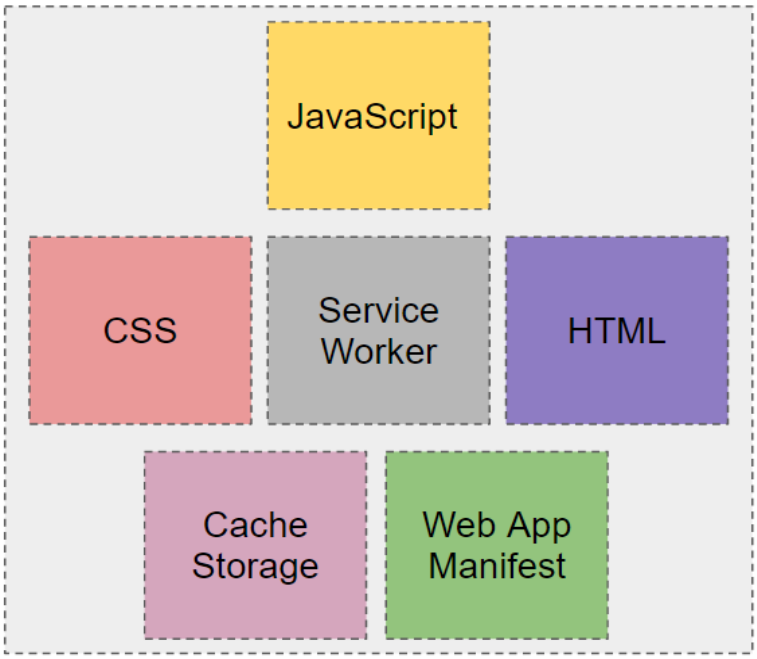

Рисунок 1.1 – Технології PWA

Можливості PWA дозволяють впроваджувати кращі веб-застосунки, забезпечуючи нативний процес користування мобільними пристроями.

Основними функціями PWA є те, що:

- Підтримує push-сповіщення.
- Має менший розмір, ніж у мобільної програми.
- Функціонує в автономному режимі (офлайн).
- Має вищу продуктивність виконання, ніж у традиційних веб-сайтах.
- Має фонове оброблення в окремому потоці.

● Підтримує можливість використання нативних функцій телефону.

● Має значок для завантажування на головному екрані телефону.

Це надає переваги для автоматизації робочих процесів проведення факультативних занять. По-перше, має високу продуктивність, працює в автономному режимі та взаємодіє з користувачами завдяки мобільним застосункам. По-друге, використовуючи такі функції, як push-сповіщення, можна надати ролям викладача та студента більше можливостей. По-третє, технічне обслуговування таких проектів простіше, ніж у мобільних застосунках, оскільки немає необхідності розробляти дві окремі програми для обох платформ (Android / iOS) і обробляти процес розгортання в App Store та Play Market. Студентам і викладачам значно простіше відкрити веб-програму за допомогою значка/лого, ніж шукати її у браузері, який може спрямовувати їх на сторінки програм аналогів.

Однак PWA має недоліки порівняно з мобільними застосунками, з якими наразі можна впоратися. Деякі нативні функції телефону, такі як Touch ID, не підтримуються. Крім того, Safari не дозволяє надсилати push-сповіщення (але в iOS 14 можна змінити браузер за замовчуванням). Отже, PWA зазвичай не підходить для проектів, що мають жорстку обчислювальну бізнес-логіку на стороні клієнта та ускладнену взаємодію з нативними телефонними модулями.

Іноді в проектах, де веб-застосунок вже впроваджено, з’являється важливість нативної мобільної версії. Наприклад, у випадках, коли для проекту потрібна лише програма для iOS / Android або веб-програма не може обробляти нові функції, які потребують нативних мобільних датчиків. Одним з найшвидших рішень може бути гібридний застосунок, розроблений за допомогою React Native. Крім того, можна реалізувати два окремі нативні програми з веб-обгортками.

Для деяких випадків розробки було б простіше створювати застосунки, використовуючи один і той же програмний код, що водночес може використовуватися на різних платформах, таких як Desktop (Browser) та Mobile

(Hybrid App, рис.2.1). Як результат, це рішення дозволяє поєднувати веб-технології та можливості доступу до функцій мобільного пристрою. Гібридні програми розгортаються у власному контейнері, який використовує мобільний об’єкт WebView. Після запуску програми цей об’єкт представляє основний вміст веб-технологій (JavaScript, CSS, HTML).

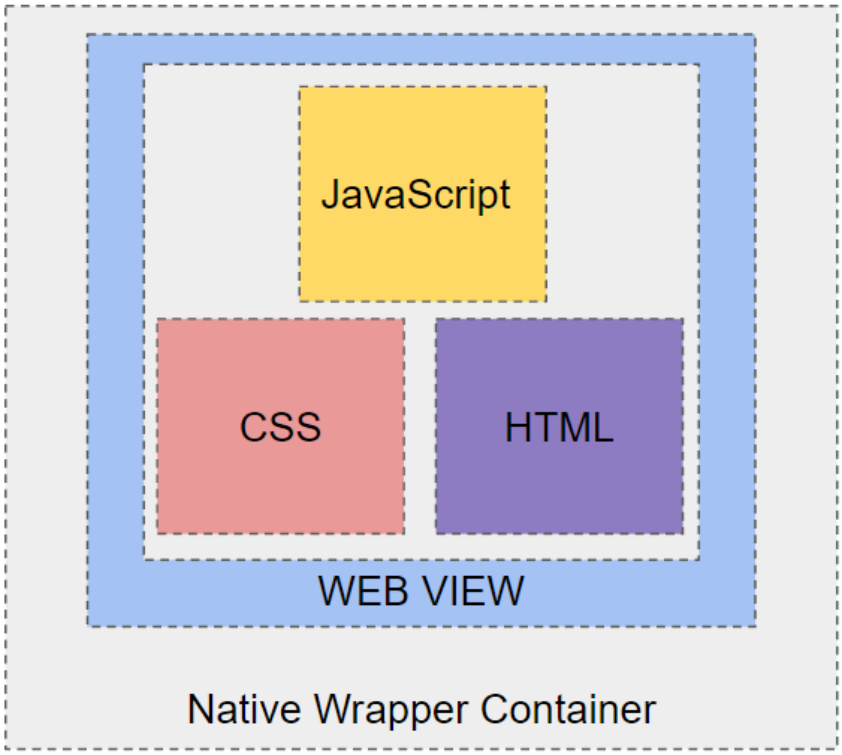

Рисунок 1.2 – Технології гібридних рішень

Це рішення дозволяє впровадити мобільний застосунок із веб-версії, спочатку “загорнутий” у власний контейнер, і під час наступних випусків змінювати функції браузера нативними. Як результат, перша версія мобільного застосунку може бути розроблена дуже швидко.

Гібридні програми з невеликою кількістю нестандартних функцій, як правило, набагато швидше створюються. Ось чому їх створення дешевше, ніж нативних застосунків.

Це рішення, як і будь-яке інше, теж має деякі проблеми. Оскільки в браузері працює рівень представлення, продуктивність програми може бути повільнішою, ніж нативна. Іноді App Store не підтримує розгортання мобільних застосунків,

реалізованих таким чином, оскільки їх кодова база на перший погляд виглядає однаковою. Крім того, веб-версія повинна мати адаптивний дизайн для мобільних екранів.

Мета крос-платформної розробки полягає в реалізації застосунків для декількох операційних систем мобільних платформ водночас, як правило, для iOS та Android (рис.1.3). Цього можна досягти, використовуючи одну кодову базу на більш абстрактному рівні, яка, у свою чергу, взаємодіє з нативними модулями кожної платформи. Наступне зображення демонструє кросплатформну архітектуру на прикладі React Native.

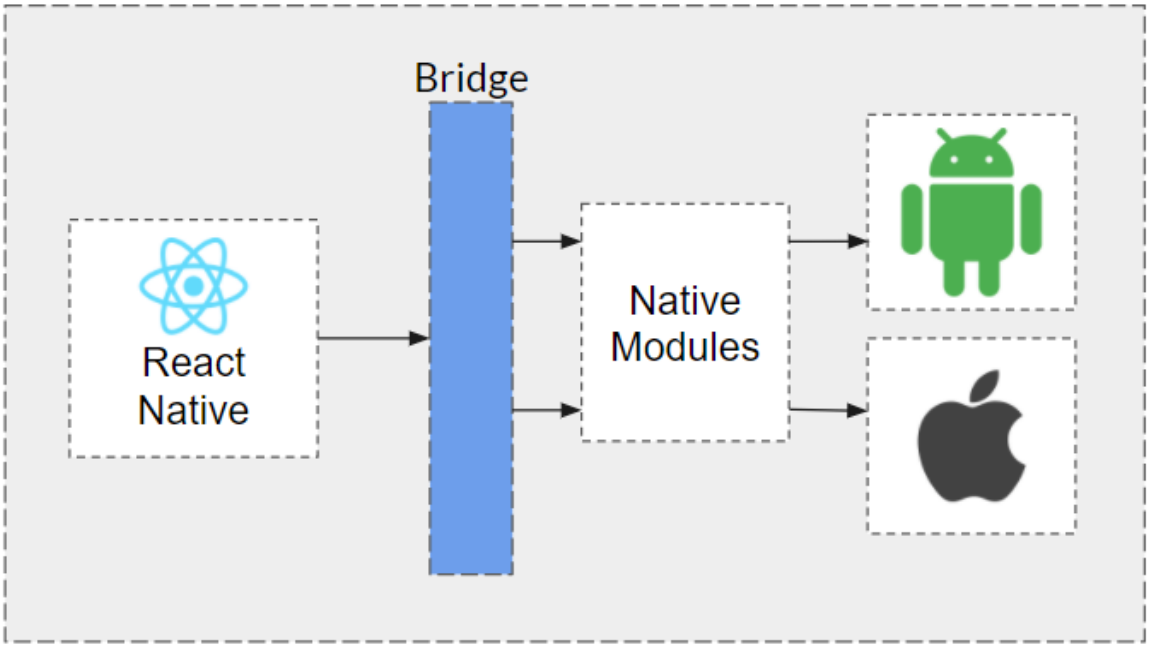

Рисунок 1.3 - Технології кросплатформних рішень

У кросплатформному застосунку достатньо мати один проект та одну команду розробників для обох платформ. Управління проектами стає простішим. Як результат, витрати на розробку можуть бути зменшені.

Є можливість повторного використання деяких частин коду між частинами iOS та Android. Причиною цього, звичайно, є одна кодова база.

Проект реалізується для кожної платформи в один і той же момент. Немає необхідності вибрати, яку платформу потрібно розробити першою. Мобільні

програми приблизно одночасно можуть бути інтегровані для обох ринків (App Store, Play Market).

З недоліків слід зазначити, що додатковий рівень абстракції, який використовується для кросплатформності, має нижчу продуктивність, ніж той самий застосунок, написаний за допомогою нативної розробки. Крос-платформні програми не можуть забезпечити повноцінних переваг нативного користувацького досвіду.

Скласти дизайн коду важче, тому що відразу потрібно обробляти особливі випадки певних пристроїв для кожної платформи і навіть винятки самих платформ.

Всі нові функції телефону спочатку створені для нативних рішень. Кожному кростплатформному фреймворку потрібен час для реалізації його підтримки. Це може сповільнити оновлення проекту.

Нарешті, важко отримати доступ до деяких функцій датчиків телефону і до всіх інтерфейсів(API) пристрою цим методом. Дуже особлива функціональність програми, яка потребує їх, іноді не може бути реалізована.

Розробка нативних застосунків - це процес створення мобільного застосунку лише для однієї платформи. Застосунок створюють мовами програмування та інструментами, які використовуються для цієї конкретної платформи.

Цей підхід передбачає, що застосунок створено та оптимізовано для певної платформи. Отже, застосунок демонструє надзвичайно високий рівень продуктивності. Нативні програми швидкі, оскільки вони створені для цієї конкретної платформи і компілюються з використанням її основної мови програмування та API.

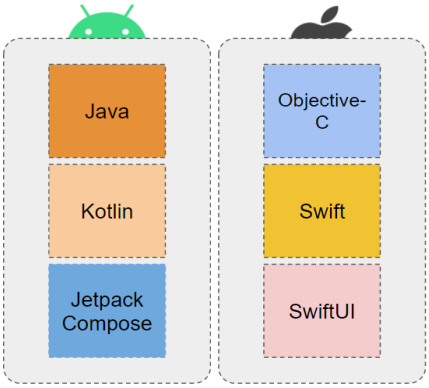

Рисунок 1.4 – Технології нативних рішень

Пристрій зберігає застосунок, що дозволяє програмному забезпеченню збільшити швидкість обробки пристрою. Поки користувачі переглядають нативну мобільну програму, вміст та візуальні елементи вже зберігаються на їхніх телефонах, а це означає, що час завантаження є швидким.

Розробник має доступ до кожного API та інструменту, що постачається платформою, над якою він працює. Взаємодія можлива з різними апаратними частинами, такими камера, мікрофон тощо, що призводить до розширення сфери можливостей функціональності застосунків.

Крім того, “рідні” програми мають менше залежностей, отже, менше простору для появи помилок. Більше того, потрібно менше зусиль, якщо виникає потреба в налаштуванні, коли виходять нові версії Android та iOS. Розробники нативних програм мають доступ до нових наборів для розробки програмного забезпечення (SDK – Software Development Kit) для створення застосунків із найновішими функціями.

З іншого боку, все ще є деякі недоліки, про які, безумовно, варто згадати. Розробка власного застоунку коштує дорого, якщо програма повинна бути доступна як в App Store, так і в Google Play. Це часто означає залучення більшої кількості розробників до роботи над різними платформами. Багато часу потрібно витратити, оскільки для кожної платформи потрібно виконати окрему роботу.

Нативні програми для Android в основному написані на Java або Kotlin. Вони обидва офіційно підтримуються Google та Android Studio. Але все ж вони мають деякі відмінності.

Свого часу Java була єдиною альтернативою, яка могла дати розробникові повний спектр можливостей у розробці Android. З часом підтримка Java зросла. Як результат, Java використовується як основний інструмент у багатьох великих проектах та корпоративних рішеннях.

Оскільки інтерфейс операційної системи Android розроблений на Java, ця мова має широкий SDK та багато готових бібліотек, які можуть стати в нагоді в процесі розробки.

1.2 Аналіз існуючих програм для факультативних занять

Для аналізу наявних програм в предметній області цієї роботи, таких майданчиках як App Store та Google Play слід відібрати мобільні застосунки, які вирішують завдання організації факультативних занять як з боку викладача, так і збоку студента. Для зважування їхніх переваг та недоліків слід визначити, за чим буде проводитися порівняння і постановка завдання.

My Study Life. Автор цієї програми вирішив створити щось, що він міг би використовувати щодня для відстеження своїх занять та завдань, що врешті-решт замінило потребу в паперовому планувальнику. Він знав, що рішенням має бути проста у використанні програма, яка дозволить йому ефективніше відстежувати свої заняття та завдання. Застосувавши програму вперше стало зрозуміло, що вона запропонує кращий досвід використання, якщо він зможе користуватися нею де завгодно і коли завгодно; наприклад, можливість додавати завдання в класі та мати його на своєму комп’ютері, повернувшись додому. Отож, через рік після першої розробки програми My Study Life (рис.1.5), веб-програма була запущена, а незабаром і програми для Windows 8 та Android.

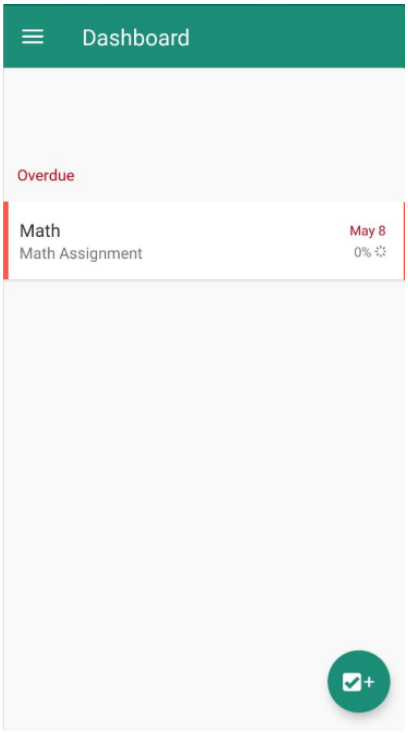

Рисунок 1.5 – Відображення заняття в застосунку “My Study Life”

Сьогодні застосунок працює на всіх популярних платформах, і команда «My Study Life» залишається зосередженою на забезпеченні того, щоб усі студенти мали просте у використанні рішення для організації всіх своїх занять, завдань та іспитів.

Chegg Prep пропонує простіше поставитися до факультативних занять за допомогою безкоштовної бібліотеки з понад п’ятистами мільйонами високоякісних карт. Chegg Prep (рис. 1.6) - це найпростіший спосіб вивчити та засвоїти знання незалежно від того, що студент вивчає.

Можливості студента:

• Створювати картки та вивчати їхній зміст безкоштовно.

• Налаштовувати навчальні матеріали за допомогою зображень та тексту.

• Знаходити актуальні картки, які мають значення для стдентів

Можливості т’юторів:

• Складати картки для свого курсу та надавати їх студентам

• Незалежно від того, що викладається, Chegg Prep - це Інтернет-бібліотека для зберігання та обміну матеріалами курсу

• Залишатися на зв’язку з іншими викладачами та дивитися, які картки доступні для курсу.

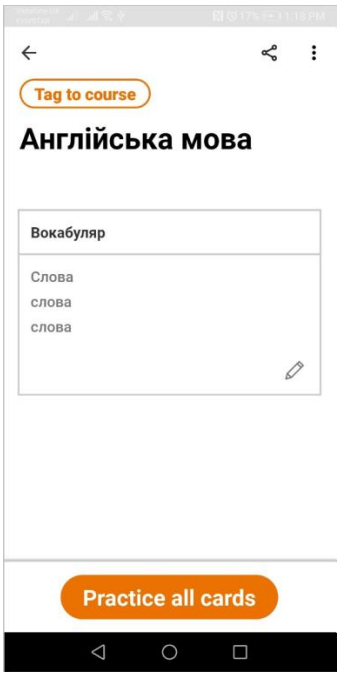

Рисунок 1.6 – Відображення карточки з вмістом для заняття в застосунку “Chegg Prep”

Байдуже, чи потрібні користувачам швидкі відповіді, чи вони хочуть забрати весь невикористаний час додаткового навчання. За допомого Chegg Prep, можна організовувати свої картки та провести наприклад вікторину або інший факультатив. З одногрупниками з Chegg Prep можна налагодити зв’язок, навчитися та засвоїти навчання.

Conqr дозволяє створити свою власну навчальну групу та ділитися знаннями та ресурсами з друзями. За допомогою Conqr користувач може:

• Створити свою власну навчальну групу або приєднатися до існуючої, щоб навчатися з друзями.

• Приєднуватись до будь-якої громадської навчальної групи або до приватної..

• Поділитися знаннями з колегами.

• Задавати запитання та вирішувати свої питання у кількох друзів, які вивчають один предмет або готуються до тих самих іспитів.

• Отримати допомогу щодо завдання або проекту в класі, розмістивши його у відповідній навчальній групі..

• Знайти кількох друзів по всьому світу зі своєї освітньої галузі.

• Створювати навчальні нотатки та діліться ними у своїй групі.

• Співпрацювати з друзями, щоб створювати спільні нотатки..

• Отримувати доступ до великої кількості навчальних ресурсів, якими ділиться ваша навчальна група, та завантажувати їх в автономному режимі у різних форматах, таких як ppt, pdf, doc, xls, txt тощо.

• Додавати у закладки важливі запитання, примітки, навчальний матеріал для подальшого використання (рис.1.7).

Є можливість створювати групи для свого особистого зростання, наприклад, вдосконалення англійської мови, логічних міркувань, словникового запасу, вивчення навичок роботи з комп'ютером чи програмування та багато іншого. Однак в цій програмі спостерігається певний функціональний надлишок який не є обов’язковим для організації факультативних занять.

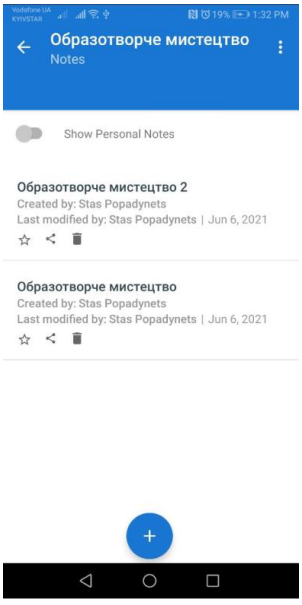

Рисунок 1.7 – Створення нотаток в застосунку “Conqr”

Brainscape - це мобільна освітня платформа, яка дозволяє студентам вивчати картки наповнені інформацією про заняття. Мобільний застосунок дозволяє студентам, викладачам створювати електронні картки та знаходити картки, створені іншими користувачами.

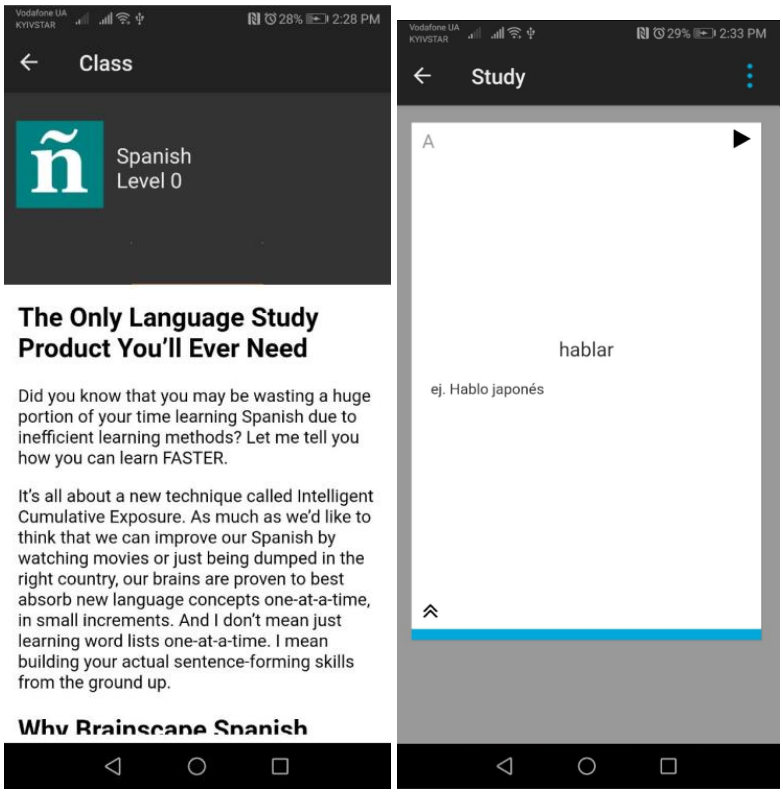

Рисунок 1.8 – Відображення інформації про заняття та приклад карточки для навчання

У студента є можливість переглядати опис та основні засади мотивації створення заняття. Окрім того тренуватися у вивченні матеріалів цього заняття.

З усіх наведених вище аналогів для порівняння можна виснувати наступне:

Таблиця 1.1 – Порівняння параметрів застосунків аналогів

	My Study Life	Chegg Prep	Conqr	Brainscape
Розділення на ролі викладача та студента	-	-	+	+
Створення занять викладачем та інформації про них	-	+	+	-

Задання тестування(контрольних запитань) викладачем	+-	+	+	-
Розділення занять і студентів на групи	-	-	+	-
Українська локалізація	-	-	-	-
Безкоштовні частини функціоналу	+	+	+	-

Огляд аналогів дозволив побачити в яких критичних критеріях є простір для вдосконалення і які хороші сторони можна взяти за взірець.

1.3 Оцінка вимог до мобільного застосунку та постановка завдання

Мобільний застосунок для організації факультативних занять має передбачати всі процеси, які полегшать взаємодію студента та викладача. Викладач має прикладати мінімальну кількість зусиль для того щоб забезпечити інформацією студента. А студент в свою чергу матиме матеріал який стосується факультативного заняття. Користувацький інтерфейс має буте дружнім і не навантажений зайвою логікою та інформацією. Палітра кольорів має бути приємною для ока. Оскільки те що виглядає гарно назовні має бути зрозумілим за своєю суттю. Увесь процес розробки слід побудувати таким чином, щоб в першу чергу займатися розробкою функціоналу застосунку, а не такими речама як налаштування середовища розробки чи адаптація до специфіки платформ, для яких розробляється застосунок. Для цього буде використовуватися кросплатформна розробка, а саме фреймворк React Native, який дозволить належним чином сфокусуватися на механізмах реалізації визначеного функціоналу без урахувань надлишкових деталей які притаманні іншим методам розробки.

Серед недоліків у вищезазначених існуючих застосунках спостерігається неспроможність поєднання критично важливих можливості для факультативних занять для викладачів та студентів до яких належить:

- Місце проведення факультативної дисципліни.
- Авторство(керівник заняття).
- Учасники(студенти).
- Проходження тестування студентами за підсумками заняття.

Функціонал застосунку має чітко окремі рамки, які дозволяють забезпечити користувачів усіма автоматизованими процесами для організації факультативних занять. Такі функції як додавання факультативних занять, інформація про них, про місце проведення та визначення питань, які дозволяють контролювати, не залишають для учасника простору для невизначеності та неправильного використання застосунку.

Підтримуючи всі базові сторони наявних мобільних застосунків, наприклад, створюючи інформацію про заняття, завдання груп, студентів, які можуть оцінювати заняття, можна розробити запрошення, яке підходить для студентів та викладачів.

Беручи до уваги всіх користувачів, які проводять організацію факультативних занять, завдання для реалізації рішення будуть наступні питання:

- Реалізувати процес організації факультативних занять з урахуванням критеріїв, необхідних для забезпечення інформації та опитування.
- Визначити функціональні та технічні вимоги для проектування та розробки мобільного застосунку.
- Передбачити та визначити сценарії навігації по застосунку та його функціональної частини та реалізувати доступність застосунку \для ролей т’ютора та студента.

Висновки до розділу 1

Упродовж дослідження загально-теоретичних засад і підходів щодо розроблення мобільних застосунків навчання можна зазначити що у еру цифрової освіти мобільні застосунки для цілей навчання, у тому числі факультативних дисциплін відіграють особливо важливу роль. Було наведено технології PWA, гібридних, кросплатформних, нативних рішень тощо.

Також було наведено мобільні застосунки MyStudyLife, Chegg Prep, Brainscape та Conqr з відповідною таблицею порівняння їх параметрів за функціями, зокрема розподілу на ролі викладача та студента, створення занять викладачем, тестовими питаннями, розподілу занять та студентів за групами, української локалізації тощо. Також було здійснено оцінку вимог до застосунку, пов’язаного з організацією факультативних занять та запропоновано дружність інтерфейсу з використанням фреймворку React Native.

РОЗДІЛ 2. АНАЛІЗ АРХІТЕТУРНИХ РІШЕНЬ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ МОБІЛЬНИХ ЗАСТОСУНКІВ НАВЧАННЯ

2.1Огляд та вибір архітектурних рішень для розробки мобільного застосунку для організації факультативних занять

Розробка програм для Android та iOS ніколи не була простою, коли справа доходить до розроблення потрібного застосунку кінцевому користувачеві, то виникають різні проблемні ситуації в контексті цільової аудиторії, її потреб.

Фреймворк React Native повністю якраз є платформою яка вирішує зазначені проблеми. В екосистемі React Native спосіб розробки програм завжди був простішим, оскільки завдяки великій підтримці спільноти можна легко обрати частину роботи, виконаної іншими розробниками. У цьому розділі будуть визначені шаблони проектування та побудови структури проекту в React-Native, які допоможуть написати застосунок, функціональна частина якого підпадатиме під вимоги проведення факультативних дисциплін.

React Native взаємодіє з нативним кодом, щоб адаптувати його. Для початку архітектура React Native спирається на «міст», який слугує комунікатором між JavaScript кодом, який охоплює використання компонентів React Native та нативним кодом. Це діє як перетворення React DOM із компонентів React в API DOM, але у випадку React Native він надсилає сигнал на повернення єдиного формату JSON асинхронно, щоб він комунікував зі своїми нативними аналогами коду.

Варто зазначити, що, незважаючи на те, що його архітектура має "міст", щоб подолати розрив між «світами» нативного коду та JavaScript, вони все одно працюють паралельно. Спілкування між ними може бути проблемою, особливо якщо починаєте свій проект у нативному коді.

React Native має контейнер управління станом. Не можна заперечувати, що екосистема React має кращий контейнер управління станом, ніж будь-які інші

фреймворки, і Redux є доказом цього, оскільки він використовується в інших фреймворках з іншою парадигмою, включаючи Angular.

У випадку для мобільної розробки для організації факультативних занять завдяки React Native Redux є гарним вибором інструментарію для управління станом застосунку, оскільки, базуючись на даних, які може зберігати Redux, залежатиме навігація по застосунку та користування ним. Такими даними можуть бути ролі користувачів (викладач чи студент), група, до якої належать студенти, дані про поточне заняття – місце проведення, опис назва. Управління станом React Native значною мірою працює подібно до того, як функціонує React Web (рис.2.1). Орієнтир ефективності з часом зростатиме з кожним роком завдяки підтримці великих компаній.

Продуктивність (performance) врешті-решт покращуватиметься. Є велика підтримка спільноти та багато підтримки великими компаніями, і це гарантує, що спостерігатиметься значний приріст продуктивності в найближчі роки, оскільки отримує необхідну увагу.

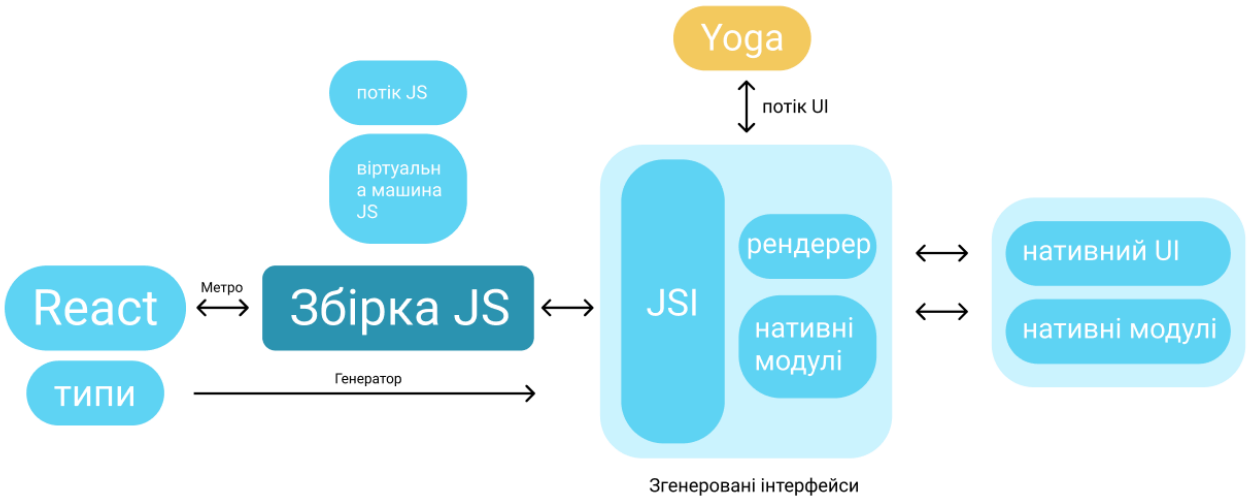

Рисунок 2.1 – Архітектура кросплатформної розробки на React Native

Компонент - це все, що React використовує в переважній більшості застосунків, і правильне написання компонентів є дуже важливим, перед нами є завдання написати застосунок, який в перспективі буде користуватися попитом

при організації факультативних занять. Ці практики використовуються професійними розробниками для створення програм на React Native,

Що стосується компонентів, дуже важливо зробити їх багаторазовими, оскільки застосунок може розвиватися з часом, коли буде важко створювати нові компоненти щоразу, щоб використовувати їх десь в іншому місці, і в кінцевому підсумку дотримуватися відомого принципу програміста не повторюватися (don’t repeat yourself – DRY). Саме презентаційні компоненти призначені для цього.

Композиція. Ключовою особливістю React є склад компонентів. Компоненти, написані різними людьми, повинні працювати разом. Важливо, що є можливість додати функціональність компонента, не викликаючи незрозумілих змін у всій кодовій базі.

Наприклад, повинна бути можливість ввести в компонент якийсь локальний стан, не змінюючи жодного з компонентів, використовуючи його. Подібним чином, повинна бути можливість додати деякий код ініціалізації та відключення для будь-якого компонента, коли це необхідно.

У використанні методів стану чи життєвого циклу в компонентах немає нічого “поганого”. Як і будь-яку потужну функцію, їх слід використовувати в помірних кількостях. Навпаки, є причини вважати, що вони є невід’ємними частинами того, що робить React корисним. В майбутньому можуть впроваджуватися більш функціональні моделі, але як локальний стан, так і методи життєвого циклу будуть частиною цієї моделі.

У React компоненти описують будь-яку поведінку, яку можна скласти, і це включає візуалізацію, життєвий цикл та стан. Деякі зовнішні бібліотеки, такі як Relay, розширюють компоненти, наділяючи їх іншими обов'язками, такими як опис залежностей даних. Цілком можливо, що ці ідеї також можуть повернутись у React.

Загальна абстракція. Загалом, є схильність протидіяти додаванню особливостей, які можуть бути реалізовані на боці користувача. Не слід розширяти програми зайвим кодом бібліотеки. Компоненти можуть бути

використані для створення компонентів, що містять стани. Коли ж компонент не має бути наділений станом, то його можна перетворити у функціональний компонент. Слід визначити, як можна перетворити його на компонент без стану чи рівня бізнес-логіки та логіки даних.

Найпоширеніший підхід - використання синтаксису функцій (arrow function) ES6, оскільки такі компоненти називаються функціональними компонентами.

Оскільки компонент не містить жодного внутрішнього стану, це означає, що в ньому не зберігаються ніякі приватні дані, все, що компонент відображає сам, забезпечується зовнішнім впливом, і компонент продовжує виконувати функцію відображення що слугуватиме користувачу за інтуїтивний та прямолінійний інтерфейс.

Прикладами застосування такого підходу до написання компонентів може бути побудова елементів, які відображають інформацію про факультативне заняття або ж поля для введення даних студента чи викладача при реєструванні. За допомогою передавання даних, властивостей, атрибутів та звичайного тексту функціональним компонентам можна підлаштовувати їхню поведінку під різні сценарії користування застосунком. Таким чином можна зекономити час для побудови інтерфейсу користувача використовуючи написані компоненти в декількох місцях.

Бекенд.

Для розгортання backend повинен мати виділений сервер. Для цього існує два доступних рішення: хмарний або локальний сервери. Кожен із них має певні переваги, які залежать від різних бізнес-потреб, таких як: спеціальні функції застосунків, його щоденне завантаження користувачами або навіть регіону, де ці користувачі знаходяться.

Хмара - це тип віддаленого сервера, який зазвичай знаходиться в центрах обробки даних. Доступ до нього можливий за допомогою Інтернету. Ці сервери, в більшості випадків, орендуються, і клієнт не є їх власником. З іншого боку,

локальний сервер - це сервер, який хтось купує і належить йому у фізичному плані для подальших цілей.

Бекенд як сервіс забезпечує основну функціональну можливість для застосунку для факультативних занять. Це зокрема є аутентифікація студентів та викладачів, управління базами даних, які зберігатимуть дані про авторів факультативних занять, їхніх учасників, локацію проведення та дані, необхідні для контролю. Це рішення дозволяє зосередитись на розробці на стороні клієнта, не турбуючись про логіку сервера. Незважаючи на те, що послуги бекенду як сервісу не є гнучкими за своєю суттю чи адаптованими до кастомізації під особливі потреби, для нашого дослідження особливих підходів не потрібно.

Найкращі рішення – це Firebase та AWS Mobile Hub (рис.2.2).

Рисунок 2.2 – Зв’язок застосунку з Firebase(BaaS)

З іншого боку, розгортання локального сервера з довільною кількістю налаштованих під власні потреби властивостей та особливостей може здатися доволі доцільним варіантом. Основними проблемами локальних серверів є їх установка та обслуговування.

2.2 Огляд способів покращення недоліків наявних рішень

Факультативні заняття - це перш за все однакова залученість як студентів так і викладачів, це вибіркове заняття обране студентом з рядку факультативних предметів чи курсів у навчальній програмі. Факультативні заняття, як правило, є більш спеціалізованими. На факультативних заняттях зазвичай менше студентів, ніж на необхідних курсах.

Це підштовхує нас до самої місії, мотивації виникнення факультативних занять. Студенти з мотивів власної волі вирішують долучитися до необов'язкової форми навчання. Для того аби полегшити їм цю ініціативу , мобільний застосунок пропонуватиме автоматизацію різноманітних процесів, що стосуються факультативних занять. Також вони мають мати можливість висловити свою задоволеність проведенням того чи іншого заняття у якійсь формі. Зручним способом це можна представити у формі опитування

Такі речі як створення факультативного заняття, інформації про нього, призначення заняття студентам певної групи і тому подібне є необхідним, щоб гарантувати зрозумілий процес автоматизації всього що стосується занять. Також, не завжди користувач зможе моментально зорієнтуватися в інтерфейсі та навігації по застосунку. Проте це є критично важливим коли користувач ні в чому іншому не є зацікавлений окрім як перегляду інформації про факультативне заняття або ж створення його у формі зрозумілій для викладача. Насамперед користувач має мати вибір в якості якої ролі кого він користуватиметься застосунком. При реєстрації йому представлена можливість вибрати викладача або ж студента, базуючись на чому залежати навігація по застосунку та взаємодія з заняттями. Викладачу буде представлено в першу чергу переглянути ті факультативні заняття які він вже створив. Створити заняття можна безпосередньо перейшовши до екрану з формою для заповнення всієї інформації що стосується заняття. Серед даних про заняття буде фігурувати інформація про контрольні запитання по тематиці факультативу. Від цього кроку викладача залежатиме те, що студент

матиме в своєму розпорядженні при взаємодії з заняттям. Класифікація заняття на теми чи підтеми буде зручним способом структурувати їх як з точки хору викладача так і з точки зору студента.

Важливим аспектом при реалізації задумки є механізм збору даних та розділення логіки збору цих даних оскільки користувач може набувати двох ролей при використанні застосунку.

До організації факультативних занять входить певна форма контролю, тестування засвоєних знань. При аналізі наявних рішень не було помічено можливості тестування для студентів. Саме ця складова є хорошим доповненням до процесу організації факультативних занять. Викладачу має бути представлена можливість задання тих запитань і варіантів відповідей на них які відповідатимуть тематиці факультативної дисципліни. До таких дисциплін можуть належати такі предмети як “Тарас Шевченко та київський університет”, “Фізична культура” та “Анлійська мова”. При цьому вся відповідальність за створення тестування лежить на викладачеві. Як для викладача так і для студента важливо бути отримувати звітність для проходження тестування, яке можна реалізувати відправленням даних про нього на електронні скриньки викладача та студента. Це дасть змогу користувачам одразу бути поінформованим стосовно всього що стосується пройденого тестування. У викладача це матиме вигляд детальної звітності по тому, на які запитання студент дав відповіді правильно, а на які – ні. Студенту ж будуть представлені загальні підсумки по проходженню контролю.

Для підбору необхідних відтінків кольорів було використано інструмент для вибору кольорів від Material Design. Це передбачає первинний колір. Первинний колір - це колір, який найчастіше відображається на екранах і компонентах програми. Основний колір можна використовувати для створення кольорової теми для програми, включаючи темні та світлі варіанти основного кольору. Використання вторинного кольору забезпечує більше способів акцентувати увагу та відрізняти те яким потрібно захопити користувача.

Наявність вторинного кольору є необов’язковою, тому в цій роботі його помірно застосовано до акцентування виділених частин інтерфейсу.

Навігація по застосунку. Її може забезпечити React Navigation. Цей інструмент входить до списку найкращих інструментів для розробки React Native, оскільки забезпечує чудовий досвід користування та навігацію в застосунку.

React Navigation - не єдине вирішення проблеми навігації для RN, але воно довело що може слугувати для розробки інтуїтивної навігації в мобільному застосунку.

За допомогою React Navigation можна реалізувати всі види переходів нестандартно. Звичайно, інструмент підтримує такі елементи навігації, як панель вкладок, навігація по стеку (ковзання поперек), навігація по шухляді (ковзання вгору) та глибокі зв'язки.

Функціонал мобільного застосунку має бути спрямований на забезпечення всього що стосується організації факультативних занять користувачеві як в ролі викладача так і в ролі студента та їхню взаємодію з сутністю факультативного заняття. З урахуванням недоліків пункти функціоналу будуть наступні:

- Реєстрація викладача та студента.
- Перегляд факультативних занять.
- Приналежність по групах.
- Задання інформації про заняття.
- Створення заняття і відображення в студента.
- Створення голосування для студентів для підрахунку підсумків заняття.
- Вивантаження файлів викладачем та завантаження студентом.

Зазначені пункти забезпечать користувача всім необхідним в процесі організації факультативних занять. В програмах аналогах частково спостерігалися окремі деякі частини функціоналу проте не всі вони були підточені під університетські вимоги з якими стикаються студенти та викладачі.

2.3 Вибір та проектування бази даних

База даних для зберігання всієї необхідної інформації про всі сутності застосунку для організацій факультативних занять є важливою та неодмінною складовою. Вона буде використовуватися для зберігання таких даних як: найменування навчальної дисципліни, її анотацію, цільова аудиторія, автор навчальної дисципліни, місце та час проведення навчання тощо. Документо-орієнтована база даних цілком справляється з таким обсягом даних для зберігання інформаційних ресурсів.

Cloud Firestore - це гнучка масштабована база даних для розробки мобільних веб-застосунків і серверів з Firebase і Google Cloud. Як і Firebase Realtime Database, він підтримує синхронізацію даних між клієнтськими застосунками за допомогою слухачів в режимі реального часу і пропонує автономну підтримку для мобільних пристроїв і Інтернет, щоб користувачі могли створювати адаптивні застосунки, які працюють незалежно від затримок в мережі. Cloud Firestore також пропонує безшовну інтеграцію з іншими продуктами Firebase і Google Cloud, включаючи Cloud Functions.

Особливості у неї такі:

- Модель даних Cloud Firestore підтримує гнучкі ієрархічні структури даних. Можна зберігати свої дані в документах, організованих в колекції. Документи можуть містити складні вкладені об'єкти на додаток до вкладених колекцій.
- У Cloud Firestore можна використовувати запити для отримання окремих конкретних документів або для отримання всіх документів в колекції, які відповідають параметрам запиту. Ваші запити можуть включати кілька пов'язаних фільтрів і комбінувати фільтрацію і сортування. Вони також індексуються

за замовчуванням, тому продуктивність запиту пропорційна розміру набору результатів, а не набору даних.

- Як і база даних в реальному часі, Cloud Firestore використовує синхронізацію даних для поновлення даних на будь-якому підключеному пристрої. Однак вона також призначена для ефективного виконання простих одноразових запитів на вибірку.
- Cloud Firestore кешує дані, які активно використовує цей застосунок, тому застосунок може записувати, читати, запитувати дані, навіть якщо пристрій знаходиться в автономному режимі. Коли пристрій повертається в мережу, Cloud Firestore синхронізує будь-які локальні зміни з Cloud Firestore.
- Cloud Firestore пропонує таке з інфраструктури Google Cloud: автоматичну реплікацію даних в декількох регіонах, надійні гарантії узгодженості, атомарні пакетні операції і підтримку реальних транзакцій. Cloud Firestore був розроблений для обробки найскладніших робочих навантажень баз даних з найбільших застосунків світу.

Нормалізація бази даних - це процес, який використовується для організації бази даних у таблиці та стовпці у нормальних формах Кодда. Основна ідея полягає в тому, що кожна таблиця повинна стосуватися певного довідника або таблиці відношення.

Обмежуючи таблицю однією метою, зменшується кількість повторюваних даних, що містяться у базі даних факультативних занять. Це усуває деякі проблеми, пов'язані з модифікацією бази даних.

Для досягнення цих цілей у цій роботі будуть використовуватися деякі встановлені правила. Перехід до оптимізованої проходить через кілька нормальних форм: першу, другу і третю нормальну форми.

Оскільки таблиці задовольняють кожній послідовній формі нормалізації бази даних, вони стають менш схильними до аномалій модифікації бази даних і більше зосереджуються на єдиній меті або темі.

У базі даних створена колекція під назвою заняття, оскільки завжди виникає потреба якимось способом ідентифікувати один екземпляр цієї сутності, документу при створення присвоюється згенерований ай-ді, що передбачено Firestore. До всіх властивостей та атрибутів цієї сутності належить такий перелік:

- title – збрігає назву заняття.
- description – зберігає опис заняття.
- teacherId – ідентифікатор викладача.
- group – зберігає назву групи, студентам якої можна відвідувати заняття.
- quiz – об’єкт, який містить в собі всі дані що стосуються опитування на тему задоволеності студентом цих занять.

Разом з тим присутня колекція з сутністю під назвою “користувач”, аналогічно документам цієї сутності присвоюється ідентифікатор згенерований Firestore. До всіх властивостей та атрибутів цієї сутності належить наступний перелік:

- role – відповідає за роль користувача – студент або ж викладач.
- email – зберігає пошту користувача.
- group – зберігає групу користувача (необов’язкове поле).

У базі даних (рис.2.3), що відповідає за аутентифікацію користувача та його сесію в застосунку, зберігаються тільки ті дані, які необхідні для забезпечення послідовності авторизації, а саме:

- ідентифікатор – електронна скринька;
- дата створення;

- дата останнього входу;
- ідентифікатор.

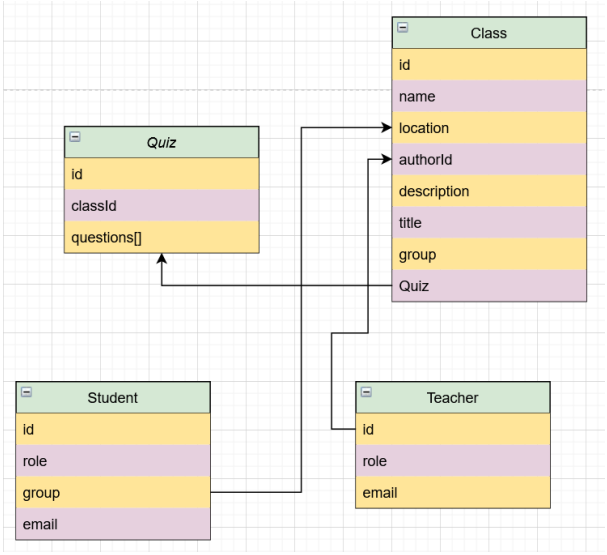

Рисунок 2.3 – Структура бази даних

Зв’язки – відношення сутностей в документо-орієнтованій базі даних подано на рис.2.3.

Висновки до розділу 2

У цьому розділі наведено огляд і вибір архітектурних рішень для розроблення мобільного застосунку для організації факультативних дисциплін.

Архітектура кросплатформної розробки на React Native є гарним рішенням. Cloud Firestore - це гнучка масштабована база даних для розробки мобільних веб-застосунків і серверів з Firebase і Google Cloud, тому підходить для розроблення мобільних застосунків навчання. Дотримання принципів функціонального програмування та паттернів разом із іншими архітектурними рішеннями дозволили належним чином структурувати написання програмного коду. Кросплатформний фреймворк React Native з бібліотекою керування станом під назвою Redux, є інструментарієм для побудови застосунку із визначеними деталями функціональної частини програми.

Визначено підхід до проектування документо-орієнтованої бази даних, а саме Cloud Firestore, що дозволяє зберігати, синхронізувати та запитувати дані для мобільного застосунку навчання.

РОЗДІЛ 3. ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ФАКУЛЬТАТИВНИХ УНІВЕРСИТЕТСЬКИХ ЗАНЯТЬ

3.1 Інформаційне забезпечення проектованої системи

Все вище вказане підводить нас до того, яка інфраструктура проекту мобільного застосунку для організації факультативних занять. Оскільки в інфраструктуру проекту та стек технологій залучено React Native, Redux, Firebase Auth та Firebase Firestore, то вона має такий вигляд інфраструктури:

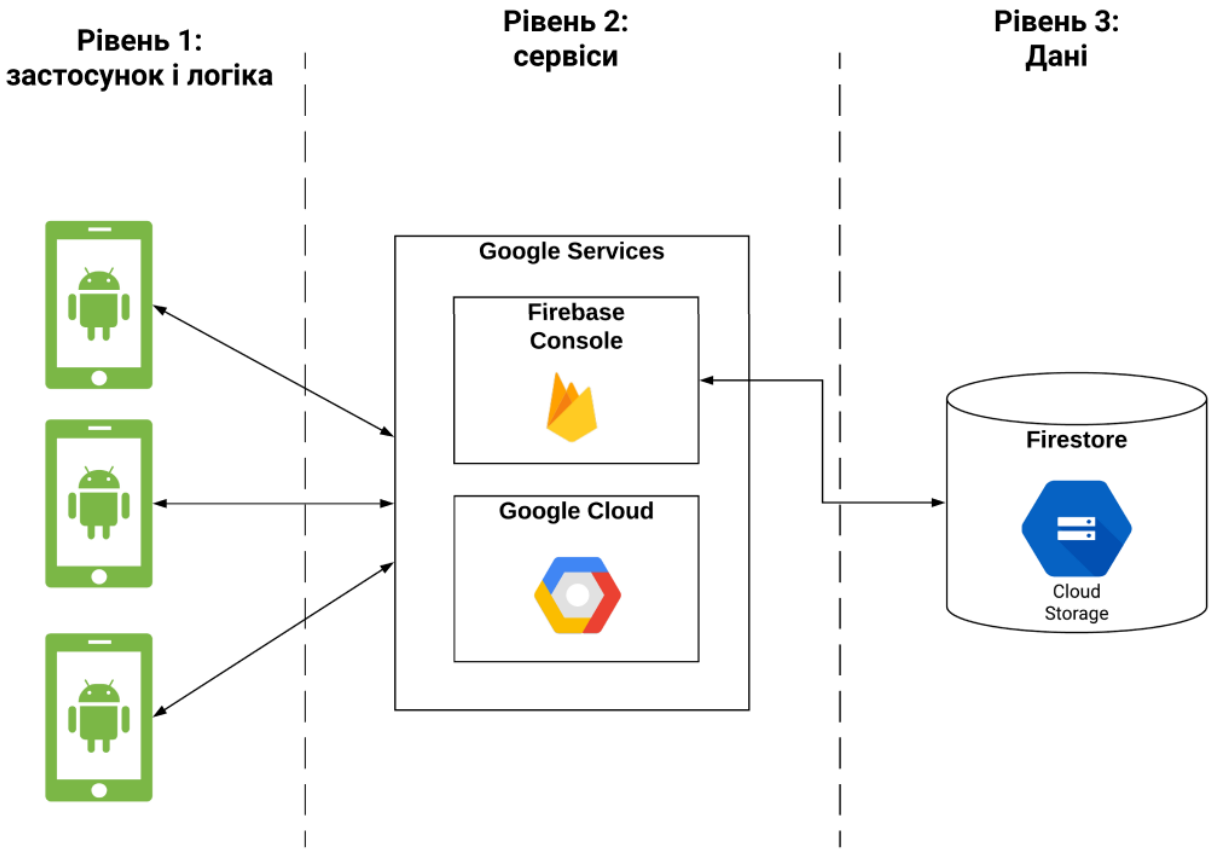

Рисунок 3.1 – Інфраструктура застосунку

Для реалізації поставлених завдань були використані різноманітні бібліотеки, набори інструментів та розширення для інтегрованого середовища розробки. Деякі з них дозволяли структурувати написання коду належним чином, інші – заощаджувати час та зусилля для написання базових частин функціоналу.

При написання програмного коду були використані розширення до середовища розробки Visual Studio Code, зокрема ES7 React/ Redux/ GraphQL/ React-Native snippets допомогло швидко генерувати шаблони для написання функціональних компонентів, в яких була написана логіка для відображення елементів користувацького інтерфейсу взаємодії з користувачем. До прикладів таких компонентів належать компонент реєстрації студента чи викладача, компонент-форма для заповнення даних про факультативне заняття, компонент-форма для створення контрольних запитань користувачем в викладача для тестування студенів за тематикою факультативної дисципліни тощо. Також Bracket Pair Colorizer був у нагоді коли справа доходила до написання функцій для реалізації обробки даних введених користувачем. Оскільки вкладеність функцій набувала високих рівнів коли доводилося фільтрувати такі речі, як інформація про місце проведення заняття, опис факультативної дисципліни. Це розширення допомогло швидко орієнтуватися де починалася область видимості однієї функції і де закінчувалася область іншої, це саме стосувалося таких блоків виконання коду як цикли та умовні вказівки. Розширення Prettier - Code formatter забезпечило чистоту і охайність описаного коду, яке передбачало його форматування при кожному збереженні файлу при умові що у поточному файлі не було синтаксичних помилок.

При процесі розроблення застосунку використовувалися були додані декілька залежностей щоб забезпечити його необхідним функціоналом для організації факультативних занять. Для цього був вибраний менеджер пакетів і залежностей npm. Він дозволяв додавати до проекту ті складові та бібліотеки, які є необхідні для реалізації застосунку з розділенням ролей на студента та викладача. Попри те що спільнота розробників пропонує безліч різновидів додаткових пакетів для даної роботи було вибрано самі ті які відповідають вимогам до мобільного застосунку для організації факультативних занять Мотивація використання тих чи інших залежностей чи бібліотек наведена у таблиці 3.1.

Таблиця 3.1 Залежності та бібліотеки мобільного застосунку для організації факультативних занять

Назва залежності	Опис
@react-native-community/async-storage redux-persist	Відповідає за збереження таких поточних даних студента чи викладача, як електронна скринька та група. Ця залежні дозволяє не втрачати дані при закритті застосунку.
@react-native-firebase/app @react-native-firebase/auth @react-native-firebase/firestore @react-native-firebase/storage	Дозволяють зв’язати мобільний застосунок з бекендом як сервісом Firebase та забезпечити студента та викладча реєстрацією, збереженням даних про факультативну дисципліну, тестування та можливістю вивантаження й скачування матеріалів для заняття.
@react-navigation/bottom-tabs @react-navigation/drawer @react-navigation/native @react-navigation/stack	Відповідають за навігацію по застосунку щоб забезпечити студенту й викладачу належні переходи між екранами які відповідають за різні частини функціоналу..
axios	Відповідає за здійснення запитів до бази даних для збереження інформації про факультативну дисципліну, тестування тощо.
react-native-document-picker	Відповідає за вибір та вивантаження матеріалів для факультативної дисципліни.

uuidv4	Генерує ідентифікатори для запитань та відповідей в тестуванні.
redux react-redux	Бібліотека управління станом мобільного застосунку. Використовується для керування і редагування даними студента чи викладача при його взаємодії з застосунком.
nodemailer	Надсилає звітність після проходження тестування на пошту студенту та викладачу.

На клієнтській частині застосунку оперування даними відбувалося за допомогою типу даних JavaScript під назвою “object”. За допомогою API мови програмування дані надсилаються до Firebase Firestore. Оскільки Firestore підтримує формат даних JSON, доволі легко передавати, зберігати та отримувати дані використовуючи цю базу даних. Наприклад, сформовані дані про створену факультативну дисципліну у об’єкт вважаються готовими для відправлення до бази даних і збереження без додаткових маніпуляцій зі змінами формату даних.

3.2 Організація факультативних занять

Викладач і студент мають в своєму розпорядженні певні можливості які надає розроблений застосунок, що дозволять здійснити процес організації факультативних занять. Для того аби розпочати цей процес користувачі мають мати доступ до мережі Інтернет та мобільний пристрій. Сам застосунок який встановлюється на телефон буде в першу чергу давати можливість зареєструватися. Після цього йому будуть представлені варіації в тому, як він

буде взаємодіяти з застосунком. У вигляді діаграми навігації по застосунку це виглядає наступним чином(рис. 3.2).

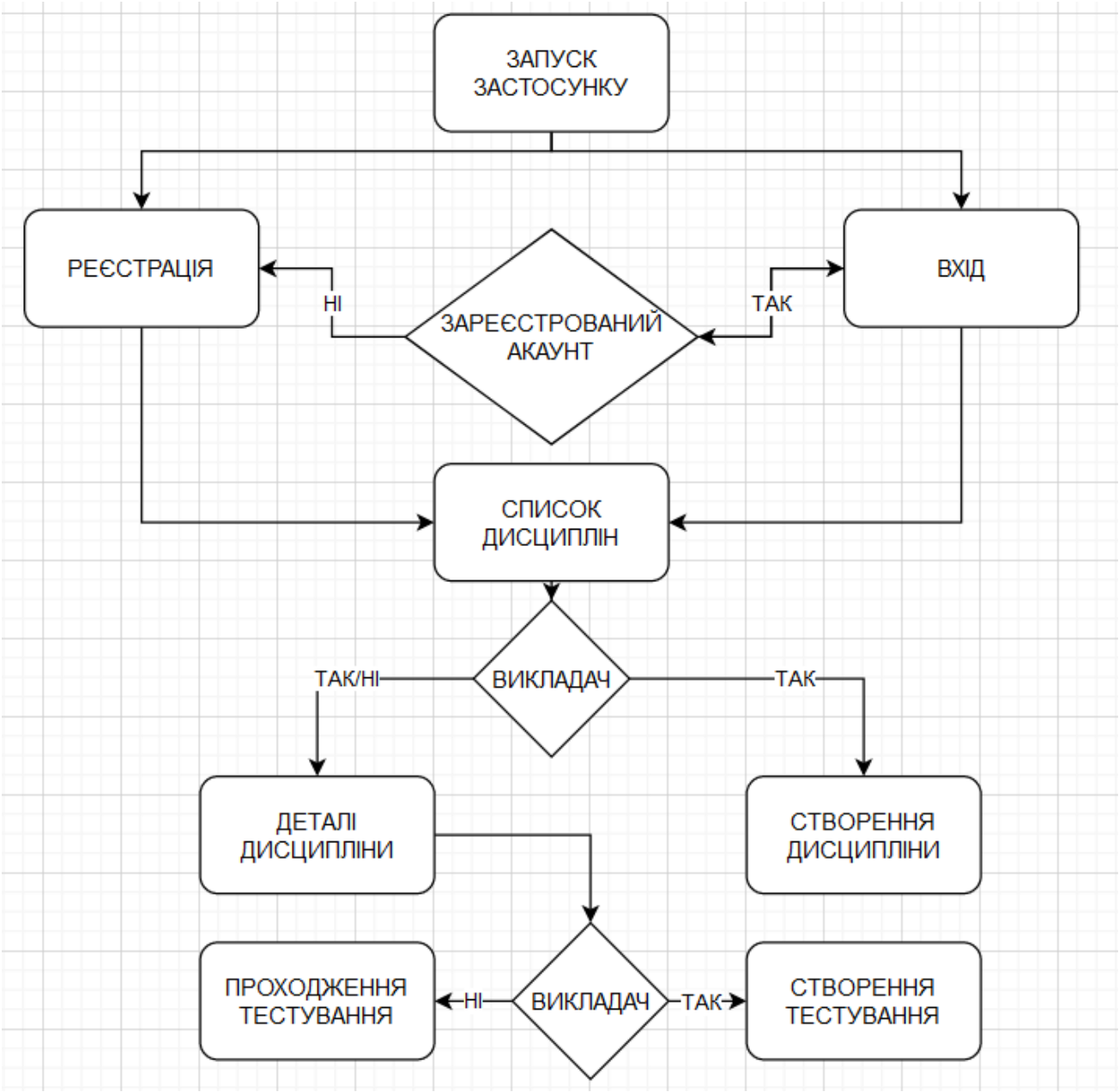

Рисунок 3.2 – Навігація по застосунку

Організація факультативних занять передбачає наповнення кожного з них певною інформацією, яка включає назву, опис, місце проведення тощо. Це саме ті речі які потрібні студенту для того щоб орієнтуватися коли в нього на меті стоїть знайомство з факультативною дисципліною та отримання базового розуміння для чого вона була створена, при умові звісно що сам студент проявив цікавість в факультативній дисципліні як такій.

Після того як ця потреба студента була адресована наступним кроком в його пізнанні факультативної дисципліни є ознайомлення з матеріалами по її тематиці, які викладач вважав за потрібне опублікувати. Ці матеріали можуть бути представлені у багатьох форматах до яких належать такі:

- csv;
- doc;
- docx;
- image;
- pdf;
- txt;
- ppt;
- pptx;
- video;
- xls;
- xlsx;
- zip;

Матеріали вивантажені викладачем дадуть студенту змогу глибше зануритися в сутність факультативної дисципліни та збагнути концепти та поняття про які в ній йдеться. Доступ до файлу, який можна завантажити можна отримати за посиланням представленим на екрані деталей дисципліни. Посилання генерується кожного разу коли викладач вивантажує файл.

Викладач, як автор факультативної дисципліни має змогу організувати своєрідний контроль який відслідковуватиме те, наскільки його аудиторія, тобто студенти, які проявили інтерес у вивченні факультативної дисципліни засвоюють її основи. Контроль створюється у формі опитування з варіантами відповідей. Викладач вирішує які запитання виноситимуться на контроль та з-поміж яких варіантів відповідей студенту доведеться вибирати правильний. Серед усіх

можливих варіантів відповідей правильною є тільки одна, та яку викладач увів першою.

Підсумки тестування підводяться у формі звітності які відправляються як тому хто проходив тестування, так і тому хто його створював. Результати надсилаються на електронну скриньку, вказану при реєстрації викладача і студента. Викладачу, тобто автору тестування відправляються два листи: один містить в собі узагальнену інформацію про проходження тестування певним студентом, другий – детальніший, який містить в собі інформацію про те, які на які відповіді студент відповідав правильно, а на які – ні. Студенту ж на електронну скриньку відправляється тільки узагальнений результат проходження тестування – відсоток правильних відповідей.

3.3 Функціональна частина застосунку

Перш ніж користувач запустить застосунок, він бачитиме його значок – перший спосіб взаємодії із застосунком. Зі зростанням кількості мобільних застосунків майданчик для застосунків перетворився на поле боротьби, де кожен мобільний застосунок бореться за свій простір.

У цьому середовищі слід переконатись, що не тільки значок цього мобільного застосунку виділяється серед інших, але і є достатньо переконливим для користувачів, щоб завантажити програму. Щонайменше це повинно зацікавити їх.

Значок виглядає наступним чином:

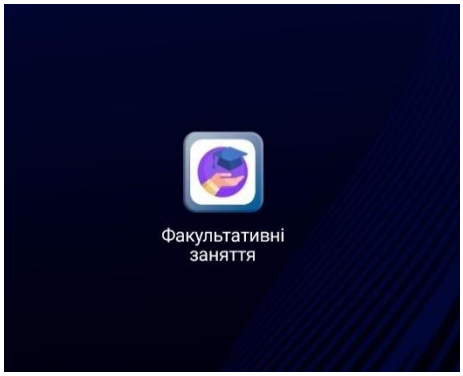

Рисунок 3.3 - значок застосунку

Користувач має змогу зареєструватися як в якості викладача так і в якості студента, йому представлений інтуїтивний інтерфейс, в якому легко орієнтуватся, після введення необхідних даних користувач переходитиме на наступну сторінку:

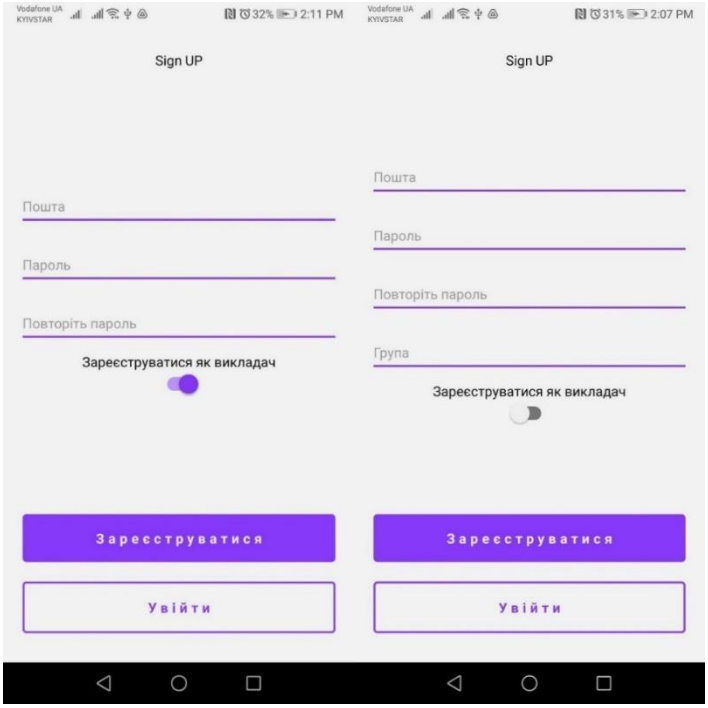

Рисунок 3.4 - Реєстрація т’ютора та студента

У випадку якщо користувач вже зареєстрований він може перейти на екран входу. Що є передбачуваним. Це допоможе користувачу не збитися з пантелику. Всі механізми навігації, зокрема при процесі автентифікації, розроблені за допомогою React Navigation.

Після реєстрації дані користувача одразу ж зберігаються в базі за допомогою сервісу Firebase Authentication, що має свої переваги оскільки для авторизації й автентифікації відповідає окремий сервіс.

При затвердженні форми входу яка містить в собі поля для введення електронної скриньки та паролю виконується функція auth().signInWithEmailAndPassword яка здійснює вхід користувача в систему, себто створює нову сесію для нього.

В методі-гачку – еквіваленті методу життєвого циклу в класових компонентах під назвою componentDidMount – виконується функція для запиту даних про списки факультативних занять які відображаються при успішному вході в систему. У випадку вчителя це ті заняття які були ним створені. У випадку студента – доступні йому.

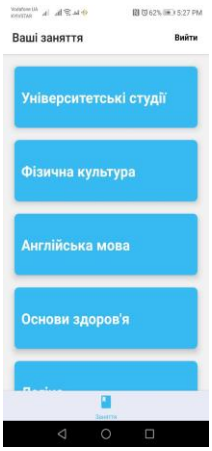

Рисунок 3.5 - Список найменувань занять

У викладача є можливість створити заняття - його назву, опис, та вказати студенти якої групи матимуть до нього доступ. Ці всі дані зберігатимуться і відображатимуться належним чином тим студентам які проявлятимуть бажання їх відвідати.

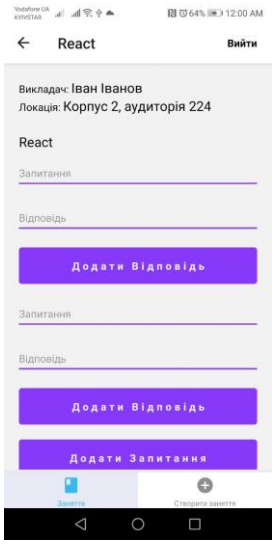

Рисунок 3.6 – Тестові запитання факультативної дисципліни

У документо-орієнтованій базі даних (Firestore) дані про заняття зберігатимуться в базі окремій від тієї яка відповідає авторизацію/ автентифікацію. Це є прямим дотриманням принципу нормалізації баз даних.

До кожного заняття у викладача є можливість прикріпити файл для освоєння матеріалу. Щоб завантажити файл безпосередньо з пристрою користувача, метод putFile у приймає рядок з уніфкованим ідентифікатором ресурсу для файлу на пристрої користувача. Це реалізовано, за допомогою допоміжної бібліотеки react-native-file-picker, яка дозволяє вибрати файл та отримати посилання(URI) на нього, яке необхідне для збереження у Firebase Storage (рис. 3.7).

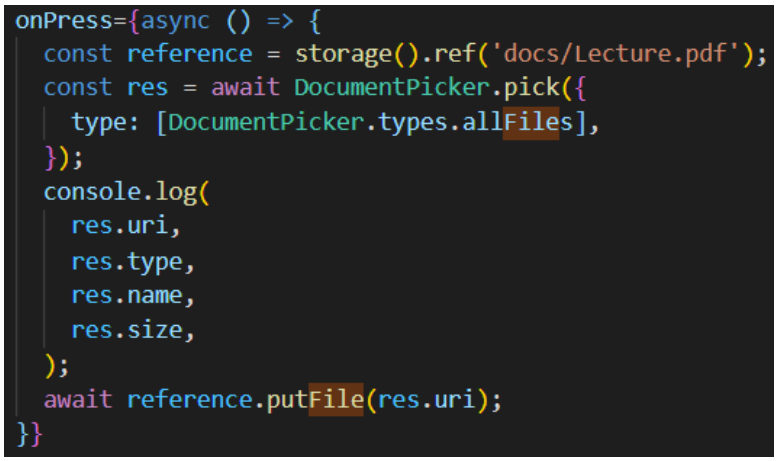

Рисунок 3.7 – Програмний код який відповідає за вивантаження фалів

Не менш важливою частиною функціоналу є процес створення контрольних запитань або ж так званого голосування по тематиці факультативного заняття після його завершення, для зберігання інформації використовується “двоповерхова” вкладеність таких типів даних як масив та об’єкт, оскільки треба вмістити багато інформації яка стосується запитань, варіантів відповідей на них та правильних відповідей. Такими можливостями наділений лише керівник заняття. Студенту надано можливість пройти тестування на основі питань викладача (т’ютора). Для студента це матиме вигляд вибору однозначної правильної відповіді з наведеного переліку варіантів відповідей на певне питання(рис 3.7).

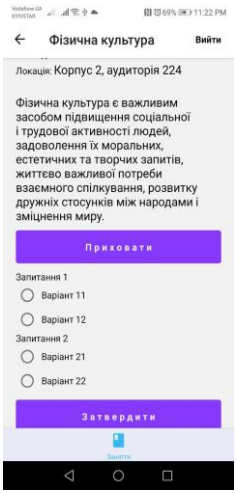

Рисунок 3.8 – Проходження тесту студентом

Після проходження тестування студенту представлений результат одразу ж після затвердження відповідей у вигляді повідомлення, яке мстить у собі інформацію про те скільки відповідей з усіх наявних були правильні. Одразу після цього звітність про проходження тестування відправляється на електронну скриньку студента, який його проходив. Окрім цього викладачу приходить два листа, один з яких містить в собі загальну інформацію про проходження тестування, а другий – детальні підсумки разом з тим на які конкретно відповіді була дана правильна відповідь, а на які – ні.

Для автоматизації процесу проходження тестування і отримання звітності за підсумками тестування було використано бібліотеку nodemailer, які потребує запуску NodeJS серверу з імплементацією логіки відправлення електронних листів. З клієнтського застосунку відправлявся запит на цей сервер який ніс в собі інформацію про те, кому слід відправляти листа і який його вміст. Ці дані оброблялися на клієнтський частині за допомогою обробника подій та форм заповнення з якими взаємодіяв користувач.

Висновки до розділу 3

До основних складових функціональної частини застосунку, який безпосередньо стосується організації факультативних занять, належить створення

заняття та інформації про нього, задання опитування за підсумками заняття, вивантаження та зберігання файлів та розподіл на роль студента та викладача.

Сформований опис функціональної частини застосунку, детально описано яким чином було реалізовано поставлені завдання та за допомогою яких програмних методів.

Авторизація студента та викладача здійснена за допомогою Firebase Auth, та дані студента які стосуються авторизації збережено в окремій базі даних.

Вивантаження файлів реалізована зо допомогою Firebase Storage. Викладачі мають змогу вивантажувати матеріали за тематикою факультативної дисципліни, а студенти – скачувати вивантажені файли.

Процес організації, проходження тестування за тематикою факультативної дисципліни та звітування реалізовані за допомогою залежності nodemailer. Дані про тестування відправляються на зберігання в Firebase Firestore.

Роботу над застосунком було розподілено на розробку клієнтської частини та інтеграцію з бекендом як сервісом, а потім структуровано їх до єдиного інфраструктурного вигляду, детально описавши процес програмування кожної частини. Після розробки застосунку проведено тестування застосунку з метою виявлення показників відповідності до поставленої задачі та функціональних вимог. За результатами тестування виконано порівняння продуктивності розроблюваного застосунку в розрізі поставлених завдань та мотивації розробки застосунку, способів організації факультативних занять, інтуїтивності використання застосунку та доступності.

Загальні висновки

У дослідженні дипломної роботи було досягнуто мету та вирішено такі завдання:

- Висвітлено загально-теоретичні засади організовування факультативних занять в університеті та систем навчання (дистанційного, мобільного тощо); сучасні підходи до розроблення і впровадження мобільних застосунків навчання.
- Здійснено аналіз архітектурних рішень і програмних засобів для реалізації мобільних застосунків для навчання, надати їх переваги та недоліки.
- Спроектовано, реалізовано, впроваджено мобільний застосунок для організації факультативних університетських занять.

СПИСОК ЛІТЕРАТУРИ

1. Ha Van. Building a universal application with React and React Native

2. Що таке мобільний застосунок? URL : https://www.techopedia.com/definition/2953/mobile-application-mobile-app

3. React Native URL: https://en.wikipedia.org/wiki/React_Native

4. Akshat Paul. React Native for Mobile Development: Harness the Power of React Native to Create Stunning IOS and Android Applications

5. Nader Dabit. React Native in Action

6. Google Firestore URL: https://firebase.google.com/docs/firestore

7. Firebase Authentication URL: https://firebase.google.com/docs/auth

8. А. Бэнкс; Е. Порселло. React и Redux. Функциональная веб – разработка

9. Хмарні обчислення URL: http://integritysys.com.ua/solutions/pricatecloud-solution/

10. Застосунки для навчальних процесів URL: https://blog.steppingblocks.com/20-essential-apps-for-college-student-survival

11.Нововведення в стандарті ECMAScript 12 URL: https://javascript.plainenglish.io/es12-is-going-to-make-your-life-easier-6be8d131e117

12. Прогресивні мобільні застосунки URL: https://web.dev/progressive-web-apps/

13. Що таке докумено-орієнтована база даних? URL : https://aws.amazon.com/nosql/document/

14. Що краще для кросплатформної розробки? URL: https://www.spec-india.com/blog/kotlin-vs-react-native

15. Організація курсів для полегшення навчання. URL: https://poorvucenter.yale.edu/OrganizingYourCourse

16. Вичерпний посібник до Redux Persist. URL: https://blog.reactnativecoach.com/the-definitive-guide-to-redux-persist-84738167975

17. Принципи проектування React Native URL: https://uk.reactjs.org/docs/design-principles.html

18. Налаштування середовища розробки. URL: https://reactnative.dev/docs/environment-setup

19. Патерни проектування на JavaScript URL: https://shivamethical.medium.com/design-patterns-in-javascript-node-2020-96c19e157428

20. Архітектура React Native. URL: https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html

21. Інткграція Firebase з проєктом React Native. URL: https://rnfirebase.io/

22. Управління станом глобальним застосунку. URL: https://redux.js.org/style-guide/style-guide

23. Факультативне заняття та його аналіз URL: https://osvita.ua/school/method/technol/726/

24. Наталія Л. Факультативи як форма організації диференціації та індивідуалізації навчання старшокласників.

25. Навігація між екранами. URL: https://reactnative.dev/docs/navigation

26. Нормалізація баз даних. URL: https://ukr.4meahc.com/database-normalization-basics-60451

Додаток

```
/**
 * @format
 */

import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

import Config from "react-native-config";

import firebase from '@react-native-firebase/app';
import 'react-native-get-random-values';

const firebaseConfig = {
  apiKey: Config.REACT_NATIVE_APP_FIREBASE_API_KEY,
  authDomain: Config.REACT_NATIVE_APP_FIREBASE_AUTH_DOMAIN,
  projectId: Config.REACT_NATIVE_APP_FIREBASE_POJECT_ID,
  storageBucket: Config.REACT_NATIVE_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: Config.REACT_NATIVE_APP_FIREBASE_MESSAGING_SENDER_ID,
  appId: Config.REACT_NATIVE_APP_FIREBASE_APP_ID,
  databaseURL: Config.REACT_NATIVE_APP_FIREBASE_DATABASE_URL,
};

if (!firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
}

AppRegistry.registerComponent(appName, () => App);
```

```
import React from 'react';
import {StyleSheet, View, Text, StatusBar} from 'react-native';
import {Provider} from 'react-redux';
import {NavigationContainer} from '@react-navigation/native';
import {PersistGate} from 'redux-persist/integration/react';

import configureStore from './Redux/store';
import AppStyles from './Styles/AppStyles';
import {SplashScreen} from './Components';
import Routes from './Routes';

const {reduxStore, persistor} = configureStore();

const App = () => {
  const [isGateLifted, setIsGateLifted] = React.useState(false);

  const onBeforeLift = () => {
```

```
    setTimeout(() => {
      setIsGateLifted(true);
    }, 1000);
  };

  return (
    <Provider store={reduxStore}>
      <PersistGate onBeforeLift={onBeforeLift} persistor={persistor}>
        {!isGateLifted ? (
          <SplashScreen />
        ) : (
          <NavigationContainer>
            <View style={AppStyles.appView}>
              <StatusBar
                translucent
                barStyle="dark-content"
                backgroundColor="transparent"
              />
              <Routes />
            </View>
          </NavigationContainer>
        )}
      </PersistGate>
    </Provider>
  );
};

const styles = StyleSheet.create({});

export default App;
```

```
import React from 'react';
import {useSelector} from 'react-redux';
import {createStackNavigator} from '@react-navigation/stack';
import {createBottomTabNavigator} from '@react-navigation/bottom-tabs';
import auth from '@react-native-firebase/auth';
import {SignUp, SignIn, HomeScreen, Details, CreateClass} from './Screens';
import {useDispatch} from 'react-redux';
import Icon from 'react-native-vector-icons/MaterialIcons';
import {Text, TouchableOpacity} from 'react-native';
import {logout} from './Redux/actions/user';
import colors from './Styles/Colors';

const Stack = createStackNavigator();
const HomeStack = createStackNavigator();

const HomeStackScreen = () => {
  const dispatch = useDispatch();
```

```
  return (
    <HomeStack.Navigator>
      <HomeStack.Screen
        name="Home"
        options={{
          title: 'Ваші заняття',
          headerRight: () => (
            <TouchableOpacity
              onPress={async () => {
                await auth().signOut();
                dispatch(logout());
              }}>
              <Text
                style={{marginRight: 20, color: 'black', fontWeight: 'bold'}}>
                Вийти
              </Text>
            </TouchableOpacity>
          ),
        }}
        component={HomeScreen}
      />
      <HomeStack.Screen
        name="Details"
        options={({route}) => ({
          title: route.params?.subject?.title,
          headerRight: () => (
            <TouchableOpacity
              onPress={async () => {
                await auth().signOut();
                dispatch(logout());
              }}>
              <Text
                style={{marginRight: 20, color: 'black', fontWeight: 'bold'}}>
                Вийти
              </Text>
            </TouchableOpacity>
          ),
        })}
        component={Details}
      />
    </HomeStack.Navigator>
  );
};

const Tab = createBottomTabNavigator();

const Routes = () => {
  const {currentUser} = useSelector((state) => state.user);
  return (
    <>
      {currentUser ? (
        <Tab.Navigator
```

```
          screenOptions={({route}) => ({
            tabBarIcon: ({color, size}) => {
              let iconName;
              if (route.name == 'Заняття') {
                iconName = 'class';
              } else if (route.name == 'Створити заняття') {
                iconName = 'add-circle';
              }
              return <Icon name={iconName} size={size} color={color} />;
            },
          })}
          tabBarOptions={{
            activeBackgroundColor: '#E8EAF6',
            activeTintColor: colors.mainBlue,
          }}>
          <Tab.Screen name="Заняття" component={HomeStackScreen} />
          {currentUser.isTeacher && (
            <Tab.Screen name="Створити заняття" component={CreateClass} />
          )}
        </Tab.Navigator>
      ) : (
        <Stack.Navigator headerMode="none">
          <Stack.Screen name="SignUp" component={SignUp} />
          <Stack.Screen name="SignIn" component={SignIn} />
        </Stack.Navigator>
      )}
    </>
  );
};

export default Routes;
```

```
import { combineReducers } from 'redux';
import { userReducer } from './userReducer';

const rootReducer = combineReducers({
  user: userReducer
});

export default rootReducer;
```

```
const initialState = {
  currentUser: null,
  errorMessage: '',
};

export const userReducer = (state = initialState, action) => {
  switch (action.type) {
```

```
    case 'SET_CURRENT_USER':
      return {
        ...state,
        currentUser: action.payload,
        errorMessage: '',
      };
    case 'LOGOUT':
      return initialState;
    case 'SET_LOGIN_ERROR':
      return {
        ...state,
        errorMessage: action.payload,
      };
    default:
      return state;
  }
};
```

```
import React from 'react';
import {useState, useEffect} from 'react';
import {Button, StyleSheet, Text, View, Linking} from 'react-native';
import {useSelector} from 'react-redux';
import {Input} from '../Components';
import OpacityButton from '../Components/OpacityButton';
import commonStyles from '../Styles/common';
import {RadioButton} from 'react-native-paper';
import {ScrollView} from 'react-native-gesture-handler';
import DocumentPicker from 'react-native-document-picker';
import storage from '@react-native-firebase/storage';
import {v4 as uuidv4} from 'uuid';
import firestore from '@react-native-firebase/firestore';
import axios from 'axios';

export const Details = (navigation) => {
  const {
    description,
    teacherName,
    location,
    group,
    title,
    teacherId,
  } = navigation.route.params.subject;

  const [questions, setQuestions] = useState([]);
  const [studentQuestions, setStudentQuestions] = useState();
  const [currentQuestion, setCurrentQuestion] = useState();
  const [currentAnswer1, setCurrentAnswer1] = useState();
```

```
const [currentAnswer2, setCurrentAnswer2] = useState();
const [isPassingTest, setPassingTest] = useState(false);
const [chosenByStudent, setChosenByStudent] = useState([]);

useEffect(() => {
  const fetchQuiz = async () => {
    if (true) {
      const ref = await firestore()
        .collection('classes')
        .where('title', '==', title)
        .get();

      const quizObj = JSON.parse(ref.docs[0].data().quiz);
      if (quizObj) {
        const reversed = JSON.parse(JSON.stringify(quizObj));
        reversed[1].answers.reverse();
        setStudentQuestions(reversed);
        setQuestions(quizObj);
      }
    }
  };
  fetchQuiz();
}, []);

const {currentUser} = useSelector((state) => state.user);
return (
  <ScrollView style={{...commonStyles.container, marginVertical: 0}}>
    <View>
      <Text>
        Локація: <Text style={styles.name}>{location}</Text>
      </Text>
    </View>
    <Text style={styles.description}>{description}</Text>
    {currentUser.isTeacher ? (
      <OpacityButton
        title="Вивантажити файл"
        onPress={async () => {
          const reference = storage().ref('docs/Lecture.pdf');
          const res = await DocumentPicker.pick({
            type: [DocumentPicker.types.allFiles],
          });
          console.log(
            res.uri,
            res.type, // mime type
            res.name,
            res.size,
          );
          await reference.putFile(res.uri);
```

```
        }}
      />
    ) : (
      <OpacityButton
        title="Завантажити файл"
        onPress={async () => {
          const url = await storage()
            .ref('docs/Lecture.pdf')
            .getDownloadURL();
          Linking.openURL(url);
        }}
      />
    )}
    {currentUser.isTeacher ? (
      <View style={{marginBottom: 80}}>
        <Input
          placeholder="Запитання"
          value={currentQuestion}
          onChangeText={(text) => {
            setCurrentQuestion(text);
          }}
        />
        <Input
          placeholder="Правильний варіант"
          onChangeText={(text) => {
            setCurrentAnswer1(text);
          }}
          value={currentAnswer1}
        />
        <Input
          placeholder="Неправильний варіант"
          value={currentAnswer2}
          onChangeText={(text) => {
            setCurrentAnswer2(text);
          }}
        />
        <OpacityButton
          title={'Додати питання'}
          onPress={() => {
            setQuestions([
              ...questions,
              {
                question: currentQuestion,
                id: uuidv4(),
                answers: [
                  {answer: currentAnswer1, isCorrect: true},
                  {answer: currentAnswer2, isCorrect: false},
                ],
```

```
            },
          ]);
          setCurrentAnswer1('');
          setCurrentAnswer2('');
          setCurrentQuestion('');
        }}
      />

      {questions.map((question) => {
        return (
          <View key={uuidv4()}>
            <Text>{question.question}</Text>
            <View>
              {question.answers.map((answer) => {
                return <Text key={uuidv4()}>---{answer.answer}</Text>;
              })}
            </View>
          </View>
        );
      })}
      <OpacityButton
        title={'Зберегти'}
        onPress={async () => {
          const ref = await firestore()
            .collection('classes')
            .where('title', '==', title)
            .get();

          await firestore()
            .collection('classes')
            .doc(ref.docs[0].id)
            .update({quiz: JSON.stringify(questions)});
        }}
      />
    </View>
  ) : (
    <View>
      <OpacityButton
        title={isPassingTest ? 'Приховати' : 'Пройти тест'}
        onPress={() => setPassingTest(!isPassingTest)}
      />
      {isPassingTest && (
        <View>
          {studentQuestions.map((element) => (
            <View key={element.id}>
              <Text>{element.question}</Text>
              {element.answers.map((answer) => {
                return (
```

```
                <View key={answer.id}>
                  <RadioButton
                    value="first"
                    status={
                      chosenByStudent.includes(answer.answer)
                        ? 'checked'
                        : 'unchecked'
                    }
                    onPress={() => {
                      if (chosenByStudent.includes(answer.answer)) {
                        setChosenByStudent([
                          ...chosenByStudent.filter(
                            (el) => el !== answer.answer,
                          ),
                        ]);
                      } else {
                        setChosenByStudent([
                          ...chosenByStudent,
                          answer.answer,
                        ]);
                      }
                    }}></RadioButton>
                  <Text style={{position: 'absolute', top: 8, left: 40}}>
                    {answer.answer}
                  </Text>
                </View>
              );
            })}
          </View>
        ))}
        <OpacityButton
          title={'Затвердити'}
          onPress={async () => {
            let correctOccurances = 0;
            const correct = [];
            const answered = [];
            console.log(studentQuestions);
            studentQuestions.forEach((el) => {
              el.answers.forEach((answr) => {
                if (answr.isCorrect) {
                  correct.push(answr.answer);
                }
              });
            });
            studentQuestions.forEach((el) => {
              el.answers.forEach((answr) => {
                if (
                  answr.isCorrect &&
```

```
                    chosenByStudent.includes(answr.answer)
                  ) {
                    answered.push({question: el.question, isCorrect: true});
                  }
                  if (
                    answr.isCorrect &&
                    !chosenByStudent.includes(answr.answer)
                  ) {
                    answered.push({
                      question: el.question,
                      isCorrect: false,
                    });
                  }
                });
              });
              console.log(answered);
              chosenByStudent.forEach((el) => {
                if (correct.includes(el)) {
                  correctOccurances++;
                }
              });
              alert(`Правильно ${correctOccurances} з ${questions.length}`);
              const teacherData = await firestore()
                .collection('users')
                .doc(teacherId)
                .get();
              await fetch('http://d4e617f71377.ngrok.io', {
                method: 'POST',
                headers: {'content-type': 'application/json'},
                body: JSON.stringify({
                  to: teacherData.data().email,
                  // to: 'popadynetss@gmail.com',
                  percentage: correctOccurances / questions.length,
                  testee: currentUser.email,
                  // testee: 'popadyentss@knu.ua',
                  subjTitle: title,
                  answered,
                }),
              });
            }}
          />
        </View>
      )}
    </View>
  )}
</ScrollView>
);
};
```

```
const styles = StyleSheet.create({
  teacherName: {
    fontSize: 14,
  },
  name: {
    fontSize: 18,
  },
  description: {
    marginTop: 24,
    fontSize: 18,
  },
});
```

```
{
  "name": "FacultyClasses",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "start": "react-native start",
    "test": "jest",
    "lint": "eslint ."
  },
  "dependencies": {
    "@react-native-community/async-storage": "^1.12.1",
    "@react-native-community/masked-view": "^0.1.10",
    "@react-native-firebase/app": "^12.1.0",
    "@react-native-firebase/auth": "^10.8.1",
    "@react-native-firebase/firestore": "^10.8.1",
    "@react-native-firebase/storage": "^12.1.0",
    "@react-navigation/bottom-tabs": "^5.11.8",
    "@react-navigation/drawer": "^5.12.5",
    "@react-navigation/native": "^5.9.2",
    "@react-navigation/stack": "^5.14.2",
    "axios": "^0.21.1",
    "react": "16.13.1",
    "react-native": "0.63.4",
    "react-native-config": "^1.4.2",
    "react-native-document-picker": "^5.2.0",
    "react-native-flash-message": "^0.1.22",
    "react-native-gesture-handler": "^1.9.0",
    "react-native-get-random-values": "^1.7.0",
    "react-native-paper": "^4.9.1",
    "react-native-safe-area-context": "^3.1.9",
    "react-native-screens": "^2.17.1",
```

```
    "react-native-vector-icons": "^8.0.0",
    "react-redux": "^7.2.2",
    "redux": "^4.0.5",
    "redux-devtools-extension": "^2.13.8",
    "redux-persist": "^6.0.0",
    "uuid": "^8.3.2",
    "uuidv4": "^5.0.1"
  },
  "devDependencies": {
    "@babel/core": "^7.8.4",
    "@babel/runtime": "^7.8.4",
    "@react-native-community/eslint-config": "^1.1.0",
    "babel-jest": "^25.1.0",
    "eslint": "^6.5.1",
    "jest": "^25.1.0",
    "metro-react-native-babel-preset": "^0.59.0",
    "react-test-renderer": "16.13.1"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

```
REACT_NATIVE_APP_FIREBASE_API_KEY=AIzaSyC1G96K5-yKYQxBp1GYa2X7sW4Mdb8JWpk
REACT_NATIVE_APP_FIREBASE_AUTH_DOMAIN=reactnativenew-64873.firebaseapp.com
REACT_NATIVE_APP_FIREBASE_POJECT_ID=reactnativenew-64873
REACT_NATIVE_APP_FIREBASE_STORAGE_BUCKET=reactnativenew-64873.appspot.com
REACT_NATIVE_APP_FIREBASE_MESSAGING_SENDER_ID=358755183059
REACT_NATIVE_APP_FIREBASE_APP_ID=1:358755183059:web:6cdbae67f17aa7d014e726
REACT_NATIVE_APP_FIREBASE_DATABASE_URL=https://reactnativenew-64873-default-rtdb.europe-west1.firebasedatabase.app/
```

```
export const areCredentialsInvalid = (email, password) => {
  // if (__DEV__) return "";

  if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email))
    return 'Please enter a correct email address.';
  if (password.length < 3)
    return 'Password must be at least eight characters long.';
  if (password.length < 8)
    return 'Password must be at least eight characters long.';
  if (!/^(?=.*?[a-z]).{8,}$/.test(password))
    return 'Password should contain at least one lowercase letter.';
  if (!/^(?=.*?[A-Z]).{8,}$/.test(password))
    return 'Password should contain at least one uppercase letter.';
  if (!/^(?=.*?[#?!@$%^&*_-]).{8,}$/.test(password))
    return 'Password should contain at least one special character.';
  return '';
};
```