МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка мобільного застосунку для планування роботи репетитора

Виконав студент групи К-20і
спеціальності 122 Комп'ютерні науки
Говера Ігор Павлович

Керівник Штефан
Наталія Зінов'ївна

Консультант д.т.н., проф.
Казакова Надія Феліксівна

Рецензент д.т.н., професор
Мещеряков Володимир Іванович

Одеса 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

API – Application Programming Interface

APK – Android Package

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

SDK – Software Development Kit

Android – операційна система

iOS – операційна система

Binding – прив'язка уявлень

C++ – мова програмування

C# – мова програмування

IDE – інтегроване середовище розробки

IntelliJ IDEA – Java IDE з вбудованою підтримкою Android

Gradle – системи автоматичного складання

Java – мова програмування

Kivy Python – фреймворк

Kotlin – мова програмування

PostgreSQL – об'єктно-реляційна система баз даних

Python – мова програмування

Objective C – мова програмування

Swift – мова програмування

SQL – Structured Query Language

UML – Unified Modeling Language

Unity – межплатформна середа розробки ігор

Use-Case – варіант використання на UML-діаграмі

UX – User Experience

ВСТУП

Не так давно смартфони вважалися джерелом відволікання студентів. Вчителі та батьки зробили все, щоб смартфони не вплинули на навчання учнів. Усі їхні спроби закінчилися марними. Але, кожна хмара має срібну підкладку. Цього разу розробка мобільних додатків була найкращою.

Мобільні додатки перетворили смартфони на віртуальні класи, де учні з легкістю та зосереджено виконують навчальні завдання. Крім того, додатки також заохочують дітей любити навчання, роблячи їх сходинкою до освіти інтерактивною та веселою.

Мобільні додатки разом із новим і передовим програмним забезпеченням для управління навчанням змінюють освітній ландшафт. Чи то вивчення предметів, організація діяльності в класі чи розвиток навичок у нових галузях, освітні програми зробили все простішим і приємнішим.

Бути вчителем важко. Вони відповідають за виховання нашої молоді. Вони часто не отримують усіх необхідних інструментів для досягнення успіху. На щастя, технології можуть трохи допомогти. Зараз є програми, які можуть трохи полегшити процес. Деякі з них допомагають відстежувати оцінки студентів та такі речі, як відвідування. Інші дозволяють вам краще спілкуватися як з дітьми, так і з батьками.

Об’єктом розробки для дипломної роботи обрано мобільний додаток репетитора. Завдяки йому викладач зможе контролювати розклад учнів з телефону або планшета на операційній системі Android.

1 АНАЛІТИЧНИЙ РОЗДІЛ ПРОЕКТУ

1.1 Опис предметної області

Продаж смартфонів зростає з блискавичною швидкістю, а компанії-виробники вкладають нечувані суми в мобільні технології, їх розвиток та популяризацію на ринку. Мобільна розробка – область програмування, що дуже стрімко зростає, адже кількість мобільних пристроїв значно перевищує кількість персональних комп'ютерів, і ця тенденція буде тільки зростати.

Онлайн-додаток для репетитора та домашнє навчання кидаються як серйозні альтернативи особистому навчанню. За останні кілька місяців навіть ми спостерігали експоненційне збільшення запитів щодо розробки додатків для онлайн репетиторів.

Зростання додатків в індустрії онлайн-освіти було підштовхнуто невпевненістю батьків у тому, що їхні діти пропускають своєчасну освіту. Громадські школи далекі від стандартизації надання послуг онлайн-репетиторства, завдяки чому ринок приватних репетиторів розвивається. Окрему приватну платформу для репетиторів розвивати швидше, ніж програму, орієнтовану на установу.

Але кажучи це, це не відриває від наборів функцій жодної з категорій. Незалежно від того, чи це шкільний вчитель на іншому кінці, чи приватно найнятий репетитор на вимогу, наступні функції програми покращать загальний досвід як для майстра, так і для учня [1].

1.2 Характеристика об’єкту розробки

Існує два різні способи технічної розробки та реалізації проектів для мобільних пристроїв: мобільний веб-сайт та мобільний додаток. На перший погляд, обидва способи виглядають дуже схожими. Але насправді кожен з них пропонує свої переваги і має власні недоліки.

Мобільний сайт працює лише через браузер. На відміну від звичайних веб-сайтів він розроблений спеціально для мобільних пристроїв. Поряд із мобільними, існують сайти з адаптивним дизайном. Адаптивний сайт містить HTML-сторінки, пов'язані разом, які переглядаються у браузерах через інтернет. Адаптивна (гумова) верстка призначена для правильного відображення всіх розмірів екранів.

Переваги мобільного/адаптивного сайту:

1. Сумісність. Зручний під час роботи з різними типами смартфонів/планшетів. Не вимагає розробки окремої версії з урахуванням різних операційних систем. Може підтримувати легку інтеграцію з такими функціями як QR-коди, текстові повідомлення.
2. Більше широке охоплення цільової аудиторії. Завдяки підтримці кількох пристроїв, які забезпечує адаптивний веб-дизайн на різних платформах, задіяна ширша аудиторія користувачів.
3. Підтримка обслуговування. В порівнянні з нативними програмами, які вимагають завантаження кожного оновлення, адаптивні веб-сайти дозволяють гнучко вносити зміни. Внесені зміни стають активними та видимими одразу на всіх типах пристроїв.

Обмеження:

1. Зручність. На відміну від програми адаптивний веб-сайт не може ефективно використовувати всі функції смартфона. Камери, GPS, телефонний набір та інші функції, вбудовані у пристрої, не завжди добре розроблені для адаптивних веб-сайтів.
2. Розмір екрана пристрою. Через невеликі розміри екрани смартфонів/планшетів відображають набагато менше контенту, ніж монітор настільного ПК або екран ноутбука. Незважаючи на те, що адаптивний веб-дизайн динамічно підлаштовується під потрібний розмір екрана, він фактично зменшує та перебудовує вміст, доступний на робочому столі [1].

3. Автономний доступ. Автономний режим функціонування мобільного сайту можливий лише за умови використання кешованих сторінок. Повноцінна робота потребує гарного підключення до інтернету.

Мобільний додаток на відміну від адаптивних/мобільних веб-сайтів, що працюють через браузери, нативні програми повинні бути завантажені з певних порталів, таких як Google Play Market, App Store та інші. Мобільні програми розробляються окремо для кожної операційної системи, вимагають установки, забезпечують швидший доступ до вмісту.

Плюси:

1. Зручність. Аналіз показує, що програми популярніші, ніж аналогічні веб-сайти, оскільки зручніші. Вони забезпечують кращу взаємодію користувача, швидше завантажують контент, простіше у використанні. Крім цього, мають push-сповіщення та дизайн, який більш гнучко сумісний із різними розмірами екрану.
2. Персоналізація. Мобільні програми є чудовим рішенням для служб, які потребують регулярного використання. Вони дозволяють користувачам створювати особисті облікові записи та зберігати важливу інформацію під рукою.

Мінуси:

1. Робота в автономному режимі Оскільки програми вимагають встановлення, вони можуть надавати доступ до своїх функцій та Сумісність. Забезпечення належного функціонування нативної програми залежить від вимог конкретної операційної системи. Це означає, що для кожної платформи (iOS, Android, Windows) потрібна окрема робоча версія програми.
2. Підтримка обслуговування. Коли програма розробляється для декількох різних платформ, його підтримка потребує більше часу та грошей. Необхідно регулярно надавати оновлення, виправляти проблеми сумісності з кожним типом пристроїв. Крім того, завжди

потрібно нагадувати користувачам про необхідність встановлення нових оновлень [2].

3. контенту навіть без підключення до Інтернету.

Об’єктом розробки для дипломної роботи обрано мобільний додаток. Завдяки йому викладач зможе контролювати учнів з телефону або планшета на операційній системі Android. Викладач легко, швидко і будь-коли може подивитися свої заняття, хто на них записаний, відзначити відвідування та прогули учнів, поставити оцінки, написати домашнє завдання та розіслати його на e-mail учням.

1.3 Огляд аналогів

Першим аналогом було розглянуто популярний мобільний додаток «Smart Tutor» (рис. 1)

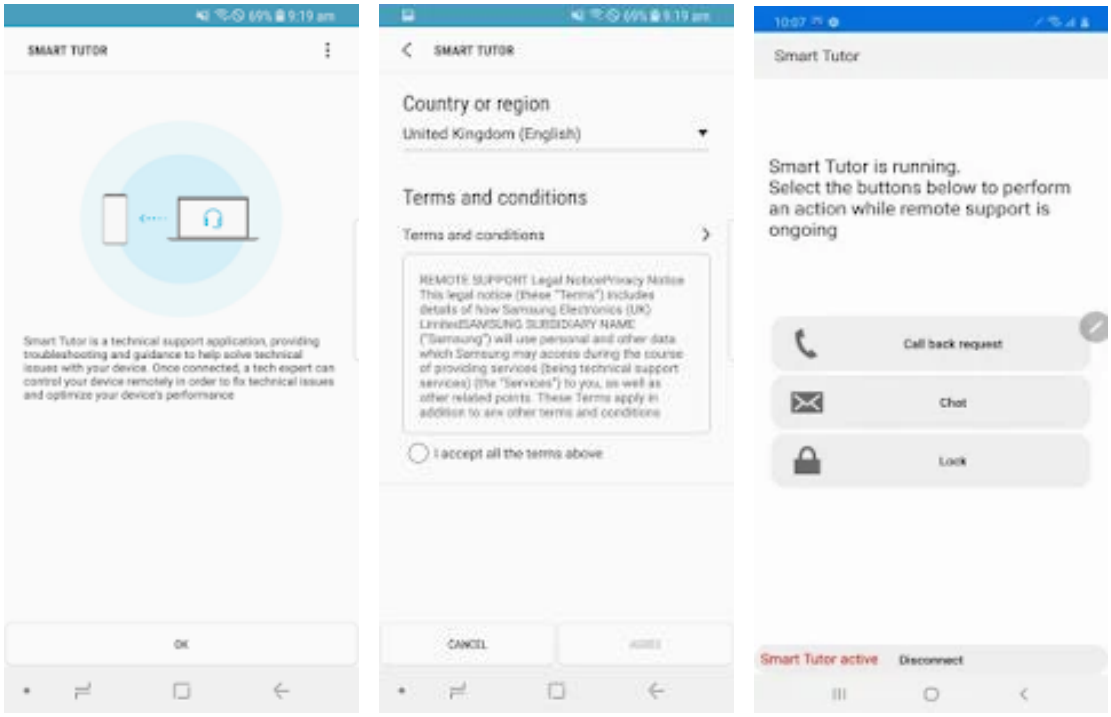

Рисунок 1 – мобільний додаток «Smart Tutor»

«Smart Tutor» є простим, швидким та безпечним засобом для здійснення консультацій щодо смартфонів та планшетів Android. За допомогою цієї програми можна зробити віддалену діагностику пристрою з метою оптимізації його продуктивності та надання наступних конструктивних рекомендацій.

Діагностика може проводитися в таких цілях:

- передача даних, резервне копіювання та відновлення;
- рекомендації щодо нових функцій;
- налаштування облікового запису (samsung/google/електронна пошта/ тощо);
- перевірка оновлень програмного забезпечення.

Переваги:

1. Безпека та надійність.

Можна не турбуватися про розкриття особистих даних. «Smart Tutor» обмежує доступ технічного консультанта до програм, що містять приватну інформацію клієнта: Галерея, Повідомлення, E-mail електронна пошта та іншим – за допомогою спеціальних функцій.

2. Зручність та простота.

Забезпечення віддаленої підтримки вашого пристрою Android відбувається швидко та легко завдяки 3G/4G або Wi-Fi.

3. Особливості – спільне використання екрана/Чат/Блокування екрана / Блокування програми.

Вимоги та примітки

1. Smart Tutor працює з ОС Android (Android 2.3.7 вище).
2. Не підтримуються пристрої Google Experience Device (наприклад, Galaxy Nexus).
3. За підключення в 3G/4G мережі стягуватиметься плата відповідно до тарифу вашого оператора. Перед підключенням слід перевірити наявність Wi-Fi-з'єднання для отримання безкоштовної підтримки.

«Remind» (раніше Remind101) – це безкоштовна програма для керування класом та спілкування. Вчителі можуть запланувати нагадування, які

надсилатимуться кожному учню та батькам. «Remind» обробляє багато основних речей, включаючи індивідуальний зворотній зв’язок, ведення записів, комунікації та функції організації.

Такий додаток може допомогти вчителю нагадати учням та батькам, щоб вони підписали ці листки дозволу на екскурсію, здали завдання та інші заходи та події. Крім цього, через додаток є можливість безпосередньо зв’язатися з учнями або батьками, якщо потрібно [2].

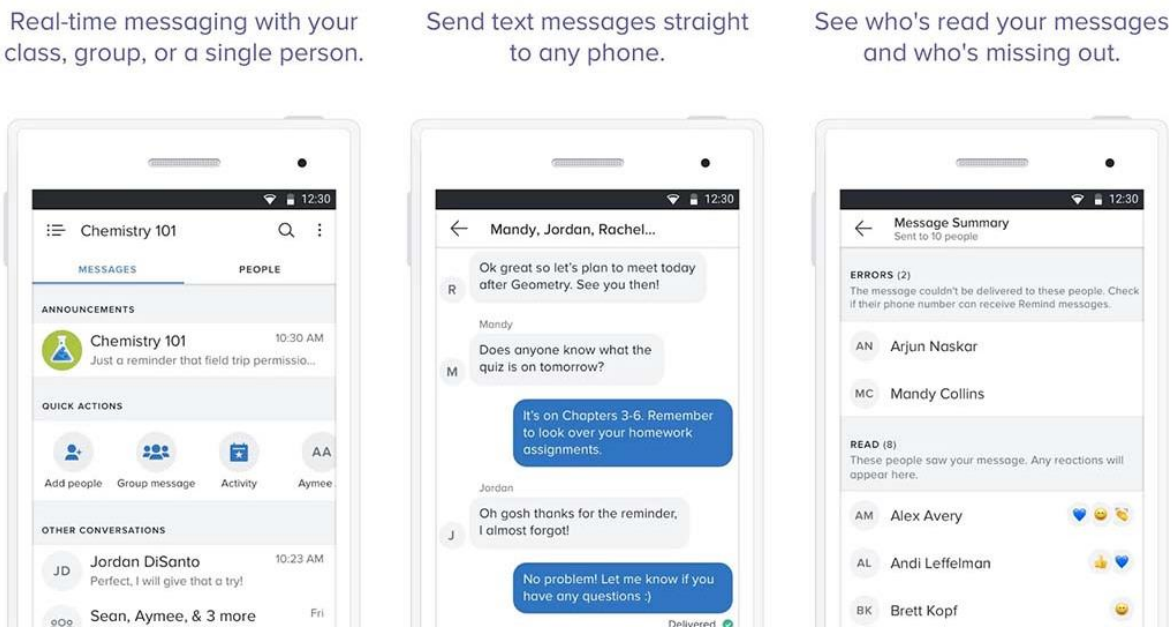

Рисунок 2 – Скрін мобільного додатку «Remind»

«Remind» підтримує 70 мов. Цей додаток також дуже простий у використанні. Це чудова альтернатива для вчителів, які хочуть спілкуватися, але не хочуть відвідувати віртуальний клас.

«TickTick» – одна з найкращих доступних програм для Android (рис. 3), особливо для вчителів. Це простий, але потужний додаток зі списком справ. В ньому є можливість запланувати всі види подій і нагадувань.

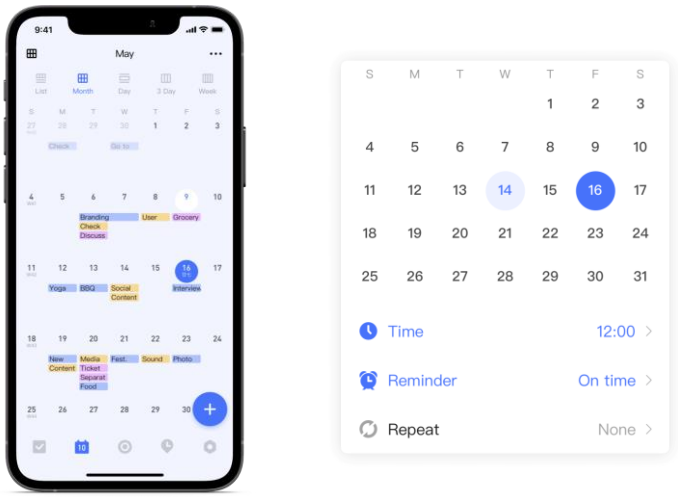

Рисунок 3 – Приклад планувальника «TickTick»

Додаток дає настроювані нагадування про майбутні події. Крім того, це корисно для списків покупок, якщо вчителю потрібні речі для арт-проекту чи чогось подібного. Безкоштовна версія програми повинна працювати для більшості людей. Профі-версія становить 27,99 доларів США на рік. У «TickTick» реалізована синхронізація між нескольками платформами (рис. 4).

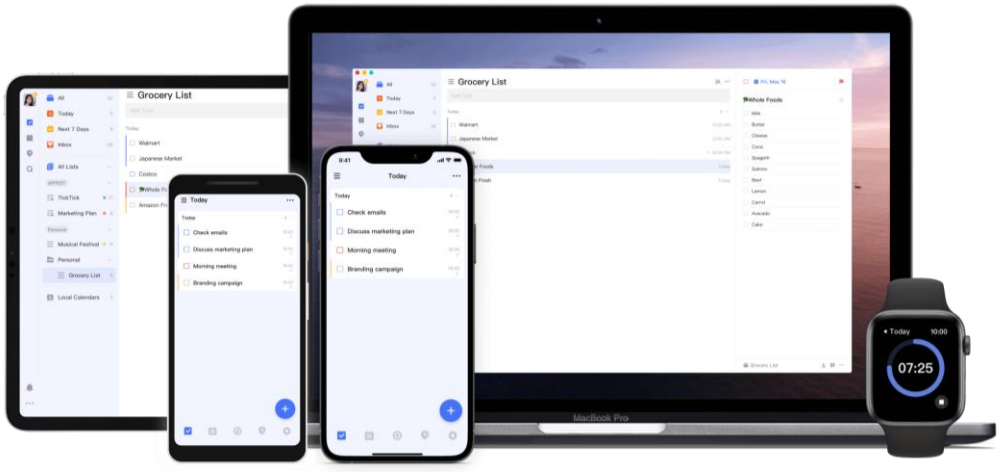

Рисунок 4 – Синхронізація «TickTick»

Мобільний додаток «Light-app» (рис. 5) – це календар уроків репетитора, де ви можете відстежувати платежі та прогнозувати свій дохід. Репетитор може додати свої заняття зі студентами до календаря, відстежувати платежі, редагувати баланс клієнтів – усе в одній програмі.

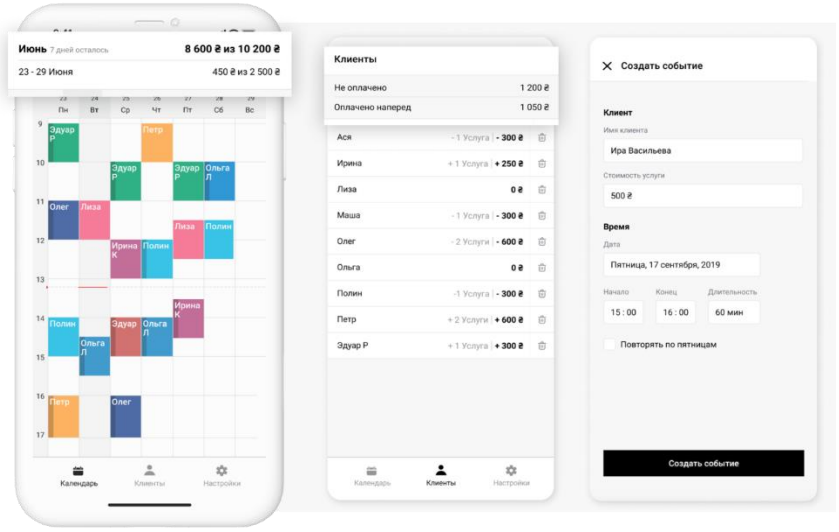

Рисунок 5 – Скріншот мобільного додаток «Light-app»

«Light» призначений для репетиторів, тренерів, психотерапевтів і всіх, хто працює з клієнтами за графіком. Найбільше він орієнтований на репетиторів, але «Light» чудово підходить і для багатьох інших професій.

До основних функцій додатку слід віднести:

1. Планування уроків.

Викладач може вести свій розклад у календарі, створеному спеціально для репетиторів. Щоб запланувати урок з клієнтом, вказується його ім’я, ціну та тривалість уроку. Повторювані світлові події автоматично переносяться на кожен наступний тиждень. У календарі є можливість побачити весь свій графік на тиждень і швидко знайти вільний час для нових клієнтів.

2. Прогнозування доходів.

На основі вашого графіка «Light» підрахує, скільки грошей репетитор заробить за тиждень і місяць. Коли студенти внесли оплату за заняття, «Light» покаже , скільки вже зароблено за минулу частину тижня та місяця.

3. Облік платежів.

Коли заняття з клієнтом закінчується, додаток автоматично «віднімає» вартість уроку з балансу клієнта (рис. 6). Також у календарі можна вказати, платне чи ні той чи інший урок. Якщо клієнт заплатить наперед, то викладач може «поповнити» його баланс, і програма автоматично розподілить ці гроші на його неоплачувану діяльність [3].

Рисунок 6 – Функції обліку платежів «Light»

4. Залишки клієнтів.

Для кожного «Light» показує його баланс: скільки студент заплатив авансом, чи скільки він повинен заплатити. Щоб змінити баланс клієнта, слід позначити платіж у календарі або поповнити його вручну. Таким чином репетитор завжди побачить, скільки винен студент, або скільки уроків він заплатив наперед.

Також на балансі клієнта відображається кількість уроків, оплачених наперед або пройдених, але ще не оплачених.

На даний час є декілька функцій, які ще не реалізовані у мобільному додатку:

- можливість перетягування подій у календарі;
- аналітика про те, скільки ви заробили в різні місяці;
- більше інформації про клієнта;
- історія платежів клієнта та інше [4].

1.4 Вимоги до проекту

Об’єктом розробки буде мобільний додаток для репетитора. За його допомогою можна буде вести записи щодо переліку учнів, назв тем для кожного заняття, а також встановлення часу початку заняття та його тривалість.

Для зручності викладача у мобільному додатку буде додано функцію щодо ведення історії оплати за навчання та можливість виставляти рівень успіху кожного учня. Так як програмний продукт орієнтований на широку аудиторію, слід спроектувати оптимальний інтерфейс користувача.

1.5 Інструментальні засоби розробки

Розробка під Android – один із найприбутковіших напрямків розробки програмного забезпечення. В даний час 75,16% всіх мобільних користувачів у всьому світі використовують пристрої Android. Також ймовірно, що операційна система Google Android продовжить зберігати свої позиції на ринку і в майбутньому.

Android пропонує розробникам масу можливостей: це універсальна, відкрита платформа, що використовується мільйонами користувачів по всьому світу, з напрочуд простим у використанні майданчиком для розповсюдження додатків.

Існує безліч інструментів для Android розробників, які допоможуть швидко почати роботу. Але ще краще те, що з кожним роком інструментів стає все більше, а їхня ефективність постійно підвищується [5].

Для різних платформ підходять різні мови, тому спочатку потрібно визначитися з платформою, а далі – з мовою. Розробка мобільних додатків для Android найчастіше виконується на Java – старому доброму об'єктно-орієнтованому, високорівневому мовою, якою написано більше 90% всіх додатків під андроїд.

За останні півроку велику популярність набирає нова мова Kotlin. Поки близько 5% програм у Google Play написані мовою Kotlin, але з кожним роком кількість цих програм зростає. Kotlin був створений, щоб удосконалити багато особливостей Java. Це типобезпечна, виразніша і менш шаблонна мова.

Якщо в Java для вирішення деяких завдань потрібно написати 50 рядків коду, Kotlin може знадобитися лише одна – це скорочує робочий час. До того ж Kotlin здатний взаємодіяти з будь-якими фреймворками, а також із шматочками коду на Java і навіть JavaScript.

Якщо говорити про iOS платформу, то тут також використовуються дві основні мови – Objective C, він перша мова, який був розроблений компанією Apple для написання додатків під iOS. А друга мова – це більш просунута і сучасніша Swift.

Якщо говорити про підтримку старих пропозицій, які були написані раніше, то тут однозначно вам потрібно знати Objective C, нові додатки все частіше пишуться саме на Swift. Варто згадати, що як для однієї, так і для іншої платформи іноді використовується мова C++. Він використовується в тих випадках, коли потрібно досягти максимальної продуктивності від вашої програми.

Дуже рідко, але все ж таки можливо в мобільній розробці зустріти використання таких мов як Python, C# і Unity, особливо після виходу нових фреймворків та бібліотек для них, адже за замовчуванням у цих мовах немає вбудованих інструментів для мобільних пристроїв. Наприклад, вихід фреймворку Kivy Python швидко просунув використання мови Python у мобільному програмуванні [6].

Було прийнято рішення розробити мобільний додаток під операційну систему Андроід. Наступний шаг, це обрати середу розробки. На сьогоднішній день найбільш затребуваними середовищами є:

- Eclipse;
- IntelliJ Idea;
- Android Studio.

Кожна IDE має унікальні характеристики. Про них докладно нижче.

1.5.1 Eclipse

Це безкоштовна IDE, розроблена некомерційною компанією Eclipse Foundation. Ця програма є базою, яка регулює процеси створення програм. Переваги Eclipse:

- інтерфейс перекладено грамотною російською мовою (документація додається);
- добре «ганяє» на комп'ютерах з низькою продуктивністю;
- має додаткові функції (для серверної роботи та аналізу бази даних);
- може підключатися до модулів;
- може працювати у груповому режимі (коли проект створюють кілька осіб одночасно).

Екліпс стала популярною кілька років тому і, як і раніше, займає лідируючі позиції. Хоча після виходу Андроїд Студіо (2014) Google вирішив перевести співпрацю з Eclipse на «другий план».

1.5.2 IntelliJ Idea

IntelliJ IDEA – це Java IDE з вбудованою підтримкою Android, створена JetBrains. Хоча Android Studio є офіційною IDE Android, IntelliJ IDEA також є чудовим вибором для більш простих програм Android. Тим більше що сама Android Studio побудована на базі IntelliJ IDEA. Ви можете знайти докладний посібник з розробки Android у документації IntelliJ IDEA.

З IntelliJ IDEA розробник отримує доступ до таких функцій, як розумне завершення коду, аналіз коду на льоту, інструменти рефакторингу та безліч корисних плагінів JetBrains. Якщо вам цікаво, як він порівнюється з Android Studio, погляньте на порівняння Slant IntelliJ IDEA та Android Studio . IntelliJ IDEA (рис. 7) має версії Community та Ultimate (комерційні).

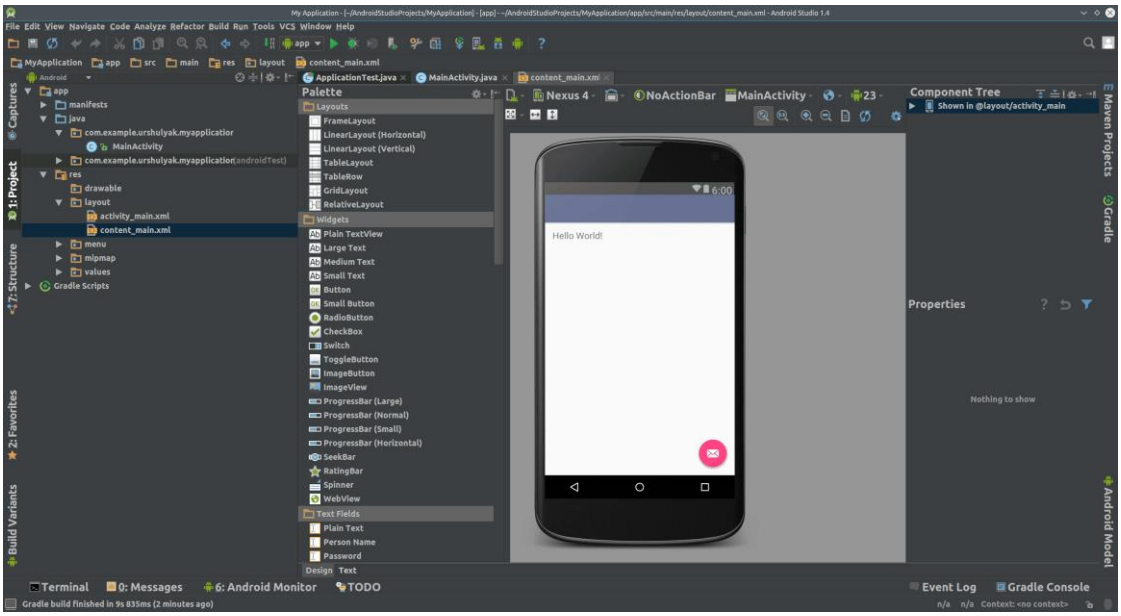

Рисунок 7 – Скрін робочого вікна середи розробки IntelliJ Idea

Чим вона примітна:

- більш оперативне налагодження значень;
- передбачений автозаповнювач методів;
- є рефакторинг;
- інтерфейс більш зрозумілий та лаконічний;
- підходить для тих, хто програмує Java.

Єдиний недолік – IntelliJ Idea потрібно платити.

1.5.3 Android Studio

Android Studio – це офіційне IDE (інтегроване середовище розробки) для розробки під Android, створене Google (рис. 8). Можна використовувати його для створення програм Android в операційних системах Windows, Linux та macOs. Android Studio підтримує всі мови програмування для розробки Android: Java, C/C++ та Kotlin.

Він поставляється в комплекті з Android SDK (Software Development Kit) та деякими додатковими інструментами розробки, такими як AVD Manager та

Android Debug Bridge. Android Studio включає редактор візуального макета, аналізатор APK (Android Package), емулятор пристрою Android, а також інструменти профілювання продуктивності в реальному часі.

Крім цього він включає редактор XML і розширений редактор макетів. Android Studio пропонує цілий набір додаткових інструментів [7].

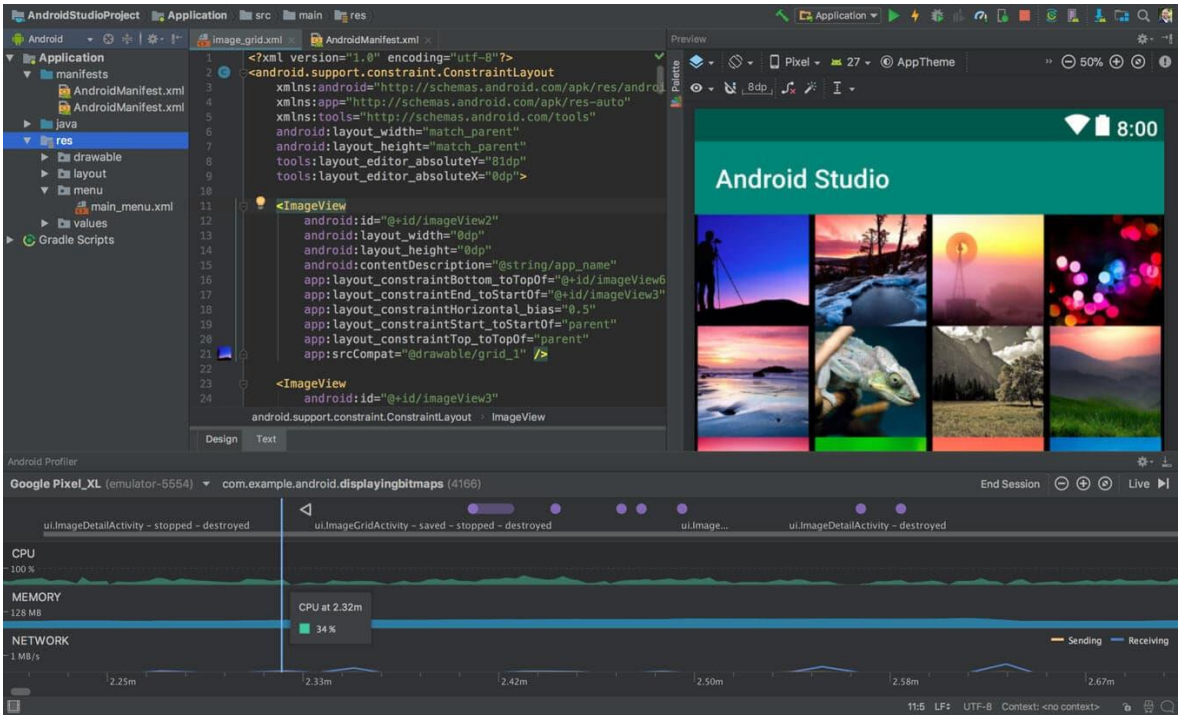

Рисунок 8 – Приклад роботи Android Studio

Інструмент AVD Manager іде в комплекті з Android Studio. Абревіатура AVD розшифровується як "Android Virtual Device", тому, по суті, це емулятор для запуску програм Android на вашому комп'ютері. Це дуже корисний інструмент, який дозволяє вам тестувати свої програми без необхідності встановлювати їх на фізичні пристрої.

AVD Manager дозволяє створювати безліч емуляторів із різними розмірами екрану, специфікаціями та версіями Android (рис. 9). Розробник може побачити, як виглядатиме його творіння на будь-якому пристрої, і цим забезпечите підтримку серед найпопулярніших гаджетів. Продуктивність

інструменту постійно поліпшується, особливо з fast virt, який запускає на вашому комп'ютері версію Android від Intel і усуває необхідність емуляції рівня instruction.

Рисунок 9 – Приклад запуску у емуляторі

IntelliJ Idea та Android Studio підходить для тих, хто:

- розробляє програми двома і більше мовами;
- працює на порівняно потужному ПК (оперативка від 2 ГБ);
- пише проги лише для ОС Андроїд.

Враховуючи те, що Андроїд Студіо – це офіційне дітище Google, створене для ОС Android, ви будете навчатися саме в цій IDE. В цілому, додаток для Андроїд можна створити будь-якою мовою. І ці середовища розробки дозволяють це зробити [8].

1.5.4 Мова програмування Java

Java – одна з потужних мов програмування загального призначення, створена в 1995 році компанією Sun Microsystems (тепер належить Oracle). Java є об'єктно-орієнтованим. Однак він не вважається чисто об’єктно-

орієнтованим, оскільки забезпечує підтримку примітивних типів даних (наприклад, int, char тощо).

Синтаксис Java подібний до C/C++. Але Java не надає низькорівневих функцій програмування, таких як покажчики. Також код Java завжди записується у вигляді класів та об’єктів. Android в значній мірі покладається на мову програмування Java, усі пакети SDK, необхідні для створення програм Android, використовують стандартні бібліотеки Java [9].

Java – офіційна мова програмування підтримується середовищем розробки Android Studio. За даними щорічного опитування ресурсу Stackoverflow, у 2019 році Java увійшов до п'ятірки найпопулярніших мов програмування.

На Java посилається більшість офіційної документації Google, а знайти платні та безкоштовні бібліотеки та керівництва не складе труднощів – їх безліч. Більшість програм для Android, створених в компанії Live Typing до 2019 року, написані на Java. Це дозволило нам реалізувати найрізноманітніші проекти, використовуючи можливості системи Android у повному обсязі.

Як у об'єктно-орієнтованої мови програмування, у нього купа особливостей у вигляді конструкторів класів, винятків, що призводять до падіння додатків під час роботи та інших моментів, які завжди необхідно враховувати при розробці. Втім, код на Java легко читається і структурується, особливо за дотримання прийнятих стандартів його оформлення.

При розробці Java під Android використовуються не тільки Java-класи, що містять код, але також файли маніфесту на мові XML, що надають системі основну інформацію про програму, і системи автоматичного складання Gradle, Maven або Ant, команди в яких пишуться мовами Groovy, POM та XML відповідно. Для верстки UI-частини зазвичай також використовується мова XML.

1.6 PostgreSQL

PostgreSQL – це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом з більш ніж 30-річною активною розробкою, завдяки якій вона заслужила міцну репутацію за надійність, стійкість функцій і продуктивність.

В офіційній документації можна знайти велику кількість інформації, яка описує, як встановити та використовувати PostgreSQL. Спільнота PostgreSQL пропонує багато корисних місць, щоб ознайомитися з технологією, дізнатися, як вона працює, і знайти можливості для кар'єрного росту.

PostgreSQL постачається з багатьма функціями, які допомагають розробникам створювати програми, адміністраторам – захищати цілісність даних і створювати відмовостійке середовище, а також допомагають керувати даними незалежно від того, наскільки великий чи малий набір даних.

Окрім того, що PostgreSQL є безкоштовним і відкритим вихідним кодом, він дуже розширюваний. Наприклад, можнавизначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування без перекомпіляції бази даних.

PostgreSQL намагається відповідати стандарту SQL, якщо така відповідність не суперечить традиційним функціям або може призвести до неправильних архітектурних рішень. Багато функцій, необхідних стандартом SQL, підтримуються, хоча іноді вони мають дещо відмінний синтаксис або функції. З часом можна очікувати подальших кроків у напрямку відповідності.

Починаючи з версії 14 у вересні 2021 року, PostgreSQL відповідає щонайменше 170 із 179 обов’язкових функцій для відповідності SQL:2016 Core. На момент написання цієї статті жодна реляційна база даних не відповідає повній відповідності цьому стандарту [10].

PostgreSQL підтримує як SQL, так і JSON для реляційних і нереляційних запитів відповідно. Багато компаній на ринку використовують PostgreSQL як

недороге рішення для сховища даних, щоб забезпечити ефективну аналітику та статистику користувачів.

Існує два способи підключення проекту Android до проекту PostgreSQL.

- використання веб-сервісів;
- використання jdbc.

Використання веб-служб є кращим вибором, коли справа доходить до підключення Android до бази даних. Але для тих, кому потрібно створити прототип своєї бази даних на етапі розробки, створення API може здатися непотрібним [11].

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

Коли визначено структуру мобільного додатка та всіх програмних елементів, які повинні до нього входити, необхідно подбати про те, щоб правильно спроектувати сам мобільний додаток, і для цього дуже важливо розуміти, що таке мокап, прототипування та UX дизайн. Цим і визначається підхід до проектування технічного завдання для мобільного додатка, адже в цьому випадку технічне завдання не записується текстом, а зображується у вигляді графічного матеріалу, який сприймати та реалізовувати набагато легше.

Дуже важливо: правильно виконати проектування, та записати всі елементи інтерфейсу мобільного додатка, відмалювати прототипи окремих екранів, попрацювати логіку різних елементів меню та кнопок, описати кількість екранів, їх функції та продумати поекранну діаграму переходів між ними (мокап/прототип). Такий підхід позбавляє необхідності довго розбиратися в ТЗ і допомагає протягом короткого проміжку часу швидко зрозуміти, про що ж буде мобільний додаток, які його основні функції і що він має виконувати [12].

2.1 UI design

«Justinmind» – призначений для прототипування графічних інтерфейсів. В ньому можна створити інтерактивні прототипи веб-сайтів та мобільних додатків. Попередній перегляд і тестування своїх каркасів і огляд у веб-переглядачі або як на реальному пристрої. За допомогою «Justinmind» прискорюється проектування, за допомогою в ньому є часто оновлювані бібліотеки інтерфейсу користувача для структурування веб- та мобільних додатків.

Також можна використовувати всі інтерактивні елементи інтерактивного інтерфейсу для веб- та мобільних пристроїв, які попередньо

завантажені в «Justinmind». За допомогою легкого перетягування и можна додати їх до робочої області та за лічені хвилини створити справжні та клікабельні каркаси.

Так як UI мобільного додаток – це сукупність активіті, кожна з яких має власний інтерфейс та функціонал, було спроектовано макети до кожної з них. Стартова, головна, активіті, включає в себе логотип додатку та панель меню (календар, уроки, теми, студенти).

Першим йде «Календар» тому що всі дані прив’язані до конкретної дати та часу. Якщо користувач обирає цей пункт меню, додаток переходить до наступного активіті з планувальником (рис. 10).

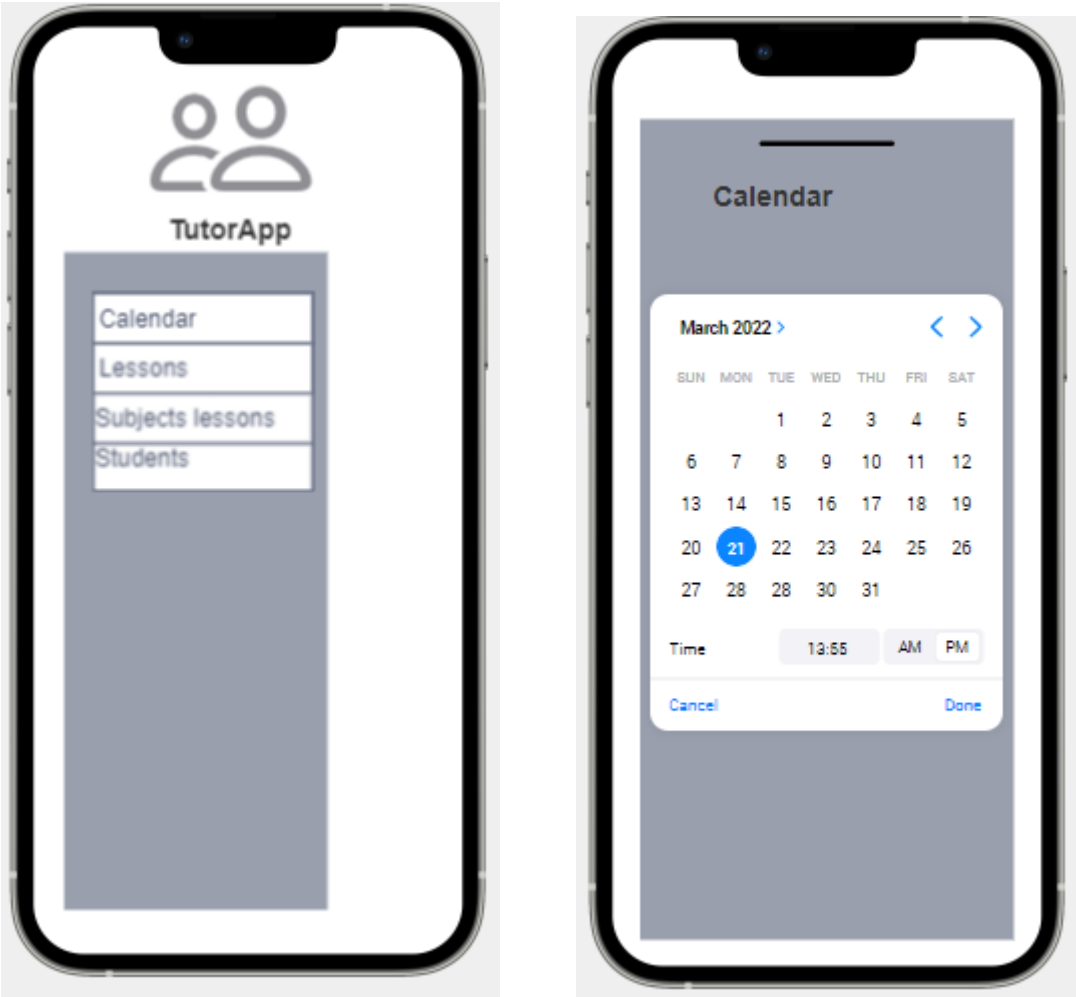

Рисунок 10 – Макеті головного активіті та активіті з планувальником

Активіті “Lessons” буде відображати інформацію щодо теми уроки, дати заняття та його тривалості. Обов’язково необхідно розмістити кнопку для додавання нового івенту. Це приведе до переходу на активіті для заповнення всієї необхідної інформації для репетитора.

Макети цих двох активіті представлено на рисунку 11:

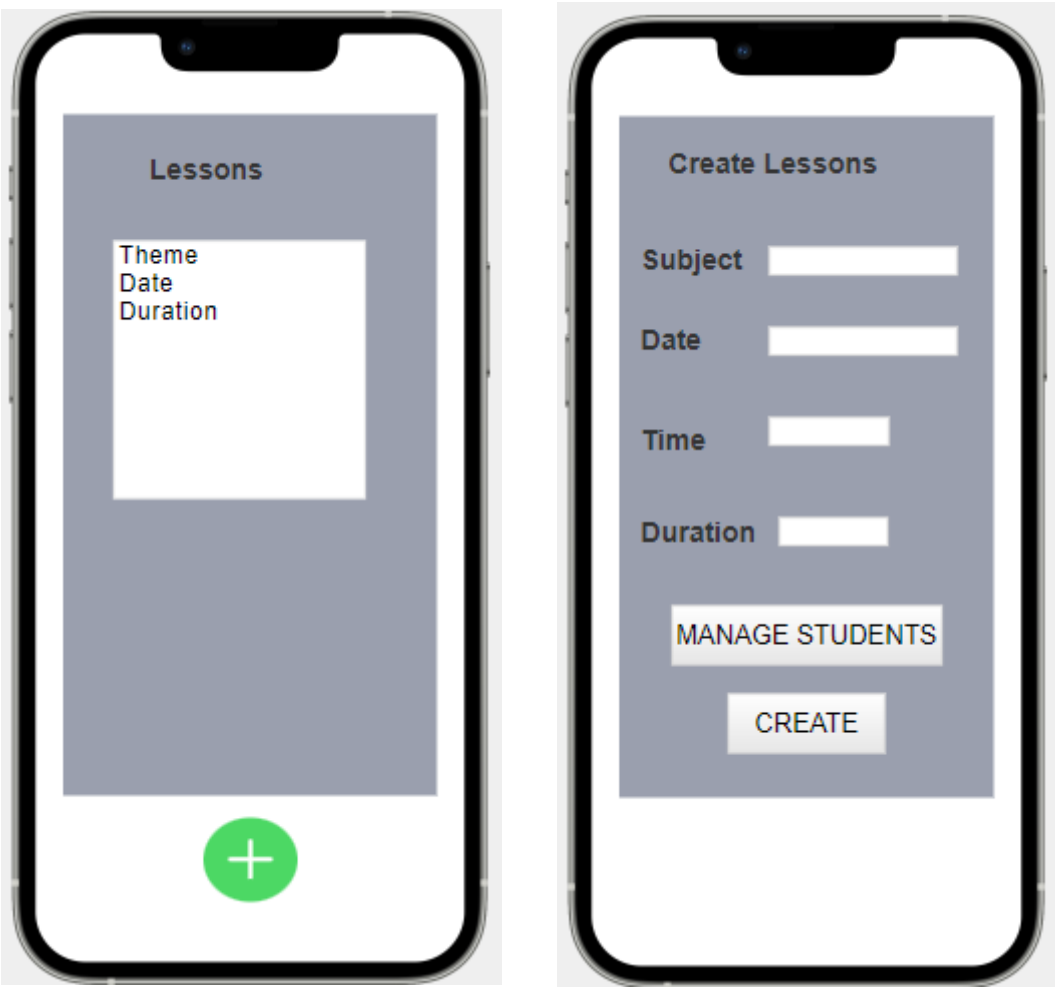

Рисунок 11 – Макети активіті для управляння даними щодо занять

Аналогічним чином для тем занять буде реалізоване два активіті: перше буде відображати перелік створених топікв занять, а друге – для створення чи редагування інформації. Приклад макетів для цих активіті представлено на рисунку 12.

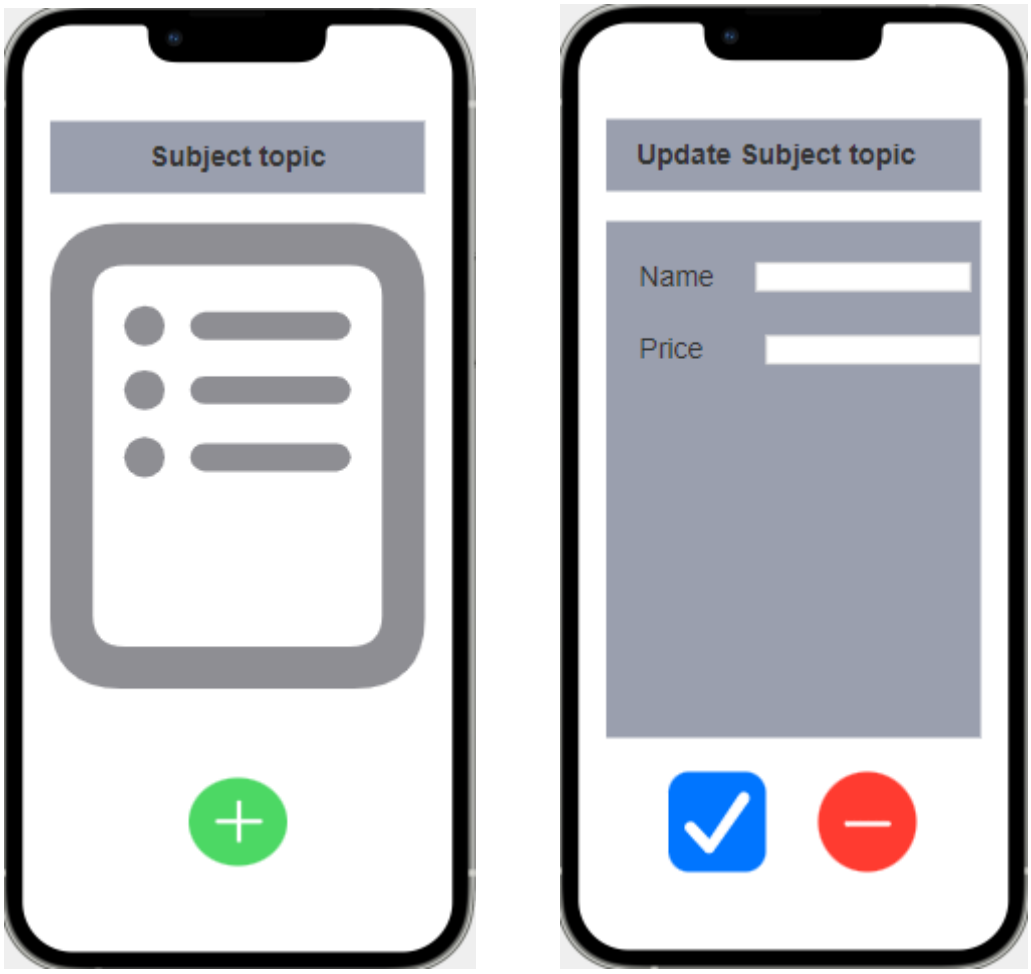

Рисунок 12 – Макети інтерфейсів для активіті з темами уроків

2.2 Діаграма варіантів використання

UML – це інструментарій моделювання, який можна використовувати для побудови діаграми. Цілі діаграми прецедентів в UML – демонстрація різних способів взаємодії користувача з системою.

В UML діаграми варіантів використання можуть оцінювати відомості про користувачів вашої системи (также відомих як діючі особи) та їх взаємодій із системою.

Діаграмма варіантів використання не містить великої кількості деталей. Вона відображає високоуровневий огляд відносин між варіантами використання, діючими особами та системами. Експерти рекомендують

використовувати варіанти використання діаграм для доповнення до більш описових текстових варіантів використання.

Приклад діаграми Use-Case представлено на рисунку 13:

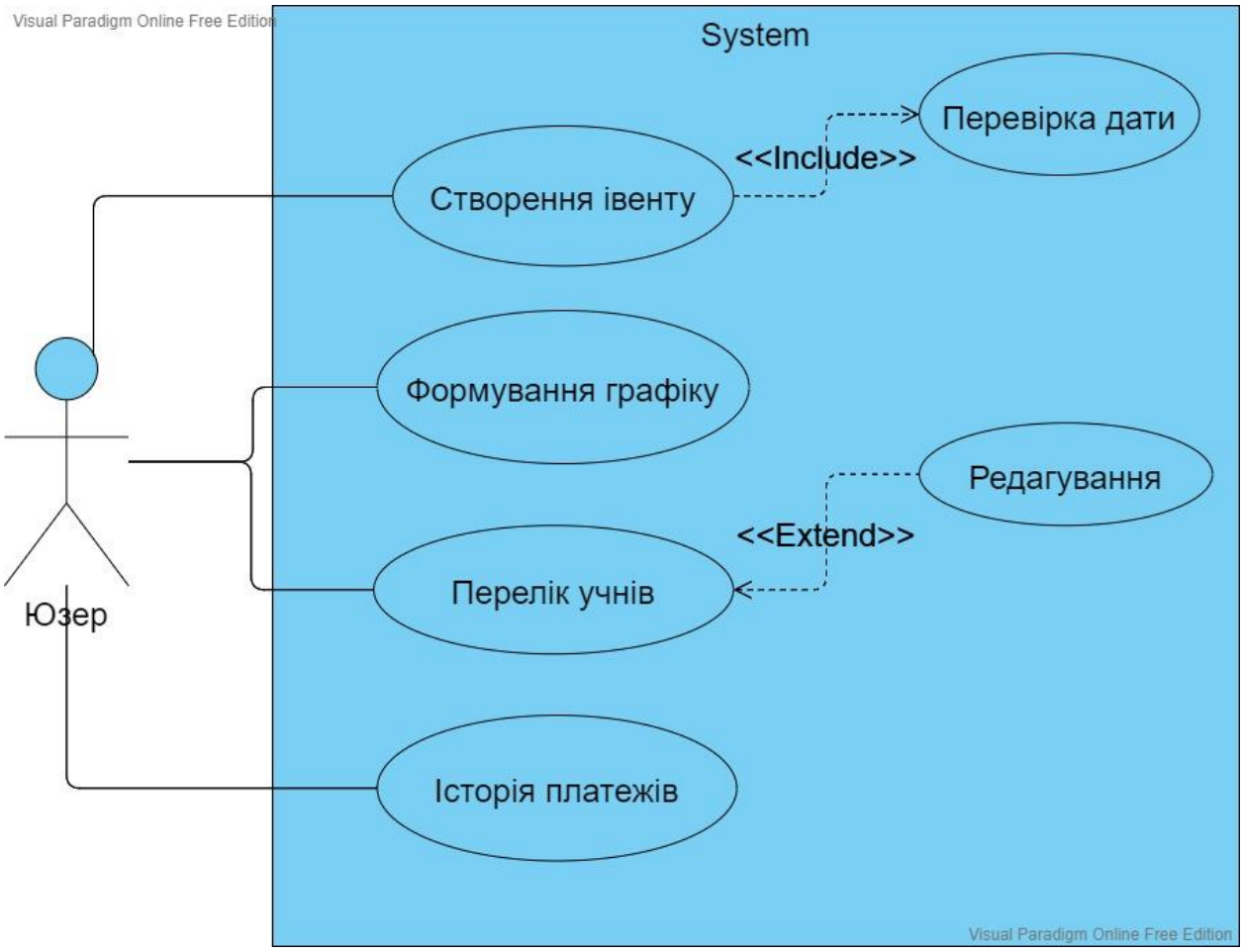

Рисунок 13 – Діаграма варіантів використання

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Файл маніфесту

Кожен проект програми повинен мати файл AndroidManifest.xml у корені вихідного набору проекту. Файл маніфесту описує важливу інформацію про вашу програму для інструментів складання Android, операційної системи Android і Google Play.

Серед багатьох інших речей, файл маніфесту повинен оголошувати наступне:

1. Компоненти програми, які включають усі види діяльності, послуги, приймачі мовлення та постачальників вмісту. Кожен компонент повинен визначати основні властивості, такі як назва його класу Kotlin або Java. Він також може оголошувати такі можливості, як конфігурації пристрою, які він може обробляти, і фільтри намірів, які описують, як можна запустити компонент.
2. Дозволи, необхідні програмі для доступу до захищених частин системи або інших програм. Він також оголошує будь-які дозволи, які повинні мати інші програми, якщо вони хочуть отримати доступ до вмісту цієї програми. Докладніше про дозволи.
3. Апаратні та програмні функції, необхідні для програми, що впливають на те, на яких пристроях можна встановити програму з Google Play. Дізнайтеся більше про сумісність пристроїв.

При використанні Android Studio для створення мобільного додатку, файл маніфесту створюється автоматично, і більшість основних елементів маніфесту додається під час створення програми (особливо при використанні шаблонів коду) [12]. Розглянемо структуру файлу-маніфесту. В головному тегі через атрибут package вказується назва пакти з нашим проектом.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
```

```
package="jc.aae.tutorapp">
```

Елемент <application> є основним елементом маніфесту та містить безліч додаткових елементів, що визначають структуру та роботу додатків. Порядок розташування елементів, що знаходяться на одному рівні, вироблений.

```
<application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.TutorApp">
```

Далі через теги <activity> в маніфесті описуємо актівіті під кожний функціонал мобільного додатку: для контролю знань студента, для фіксування дати та часу заняття, для оновлення даних щодо учнів викладача та інше.

```
<activity
        android:name=".ManageStudentKnowledgeActivity"
        android:exported="true" />
    <activity
        android:name=".UpdateStudentKnowledgeActivity"
        android:exported="true" />
    <activity
        android:name=".ManageStudentsActivity"
        android:exported="true" />
    <activity
        android:name=".UpdateStudent"
        android:exported="true" />
    <activity
        android:name=".UpdateSubjectTopic"
        android:exported="true" />
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:label="@string/title_activity_main"
      android:theme="@style/Theme.TutorApp.NoActionBar">
```

Елемент <intent-filter> в MainActivity вказує, як дана діяльність буде використовуватися.

За допомогою дії android:name="android.intent.action.MAIN", що дана активність буде вхідною точкою в додатку і не повинна отримувати якісь дані з вибуху.

Елемент категорії android:name="android.intent.category.LAUNCHER" вказує, що MainActivity буде представляти стартовий екран, який відображається при запуску додатків.

```
 <intent-filter>
                <action
  android:name="android.intent.action.MAIN" />
                <category
  android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".UpdateLesson"
            android:exported="true" />
    </application>
</manifest>
```

Далі розглянемо як задається дизайн до активіті та фрагментів інтерфейсу користувача. Приклад розмітки головної активіті представлено на рисунку 13. Для неї обрано зображення у якості іконки мобільного додатку (рис. 14).

Рисунок 13 – Іконка мобільного додатку

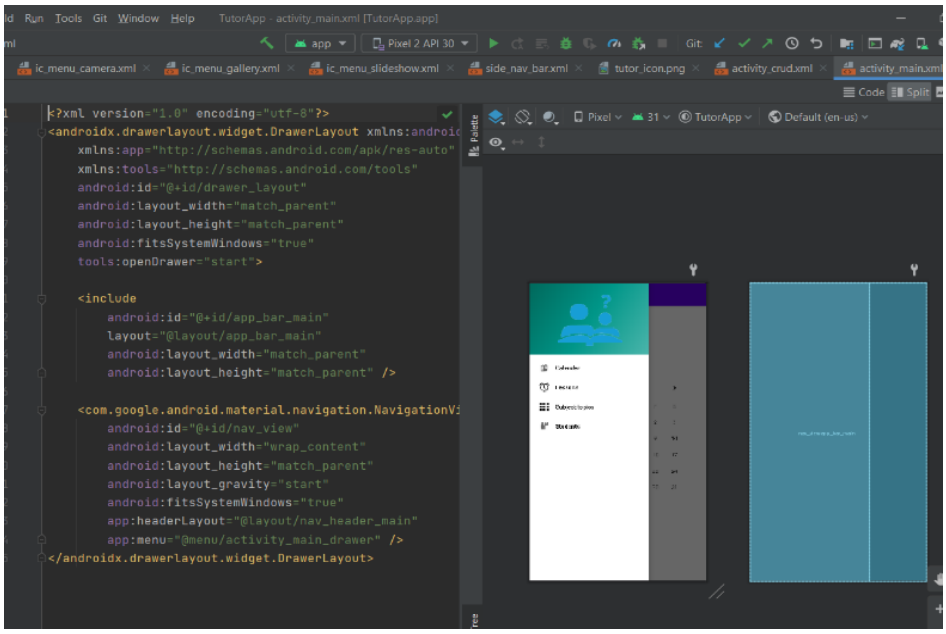

Рисунок 14 – Приклад дизайну головної активіті

На головній активіті є чотири навігації – календар для перегляду та управління уроками на підставі дати, навігація з управління уроками, студетами та темами (рис. 15).

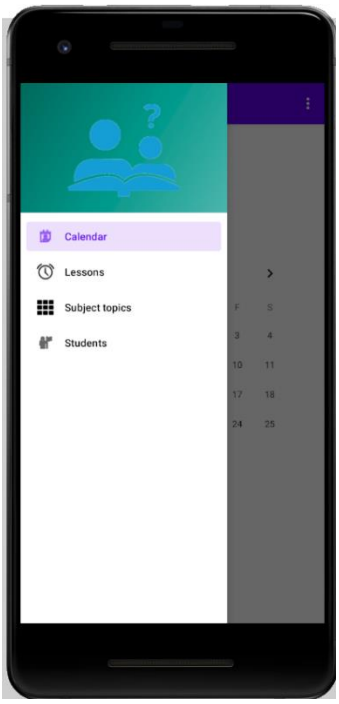

Рисунок 15 – Скрін екрану навігації

Фрагменти навігаційного меню представлено на рисунку 16. Всього 4 класи (StudentFragment. CalendarFragment, LessonsFragment, SubjectFragment)/

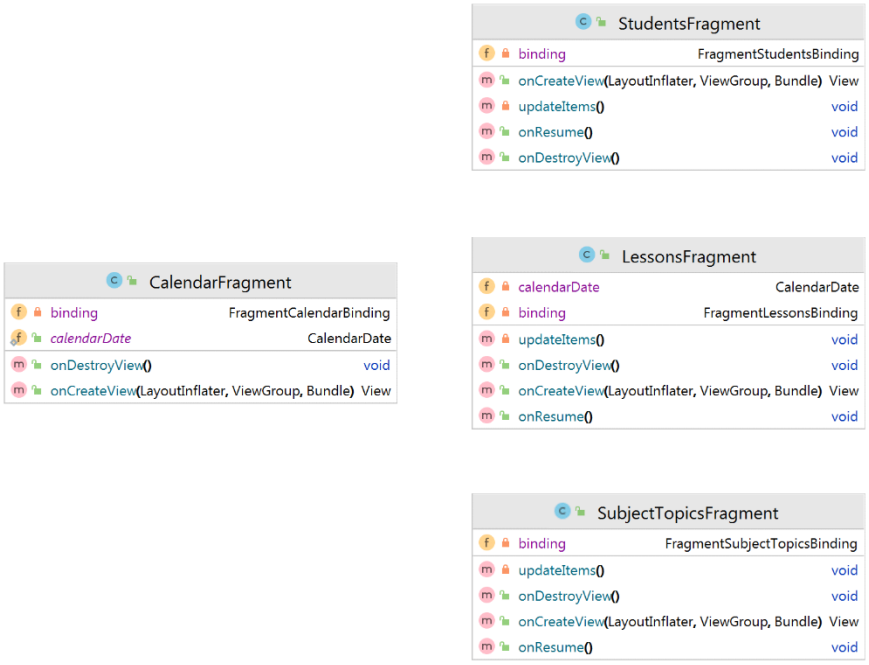

Рисунок 16 – Фрагменти навігаційного меню мобільного додатку

3.2 Реалізація активіті

Через закладку «Календар» користувач переходить до меню управління уроками на обрану дату (рис. 17). Фрагмент календаря реалізована наступним чином: спочатку отримуємо байдинг (за його допомогою можна отримувати елементи на фрагменті) та створюємо поле для зберігання обраної дати.

```
public class CalendarFragment extends Fragment
{
      private FragmentCalendarBinding binding;
          public static CalendarDate calendarDate;
```

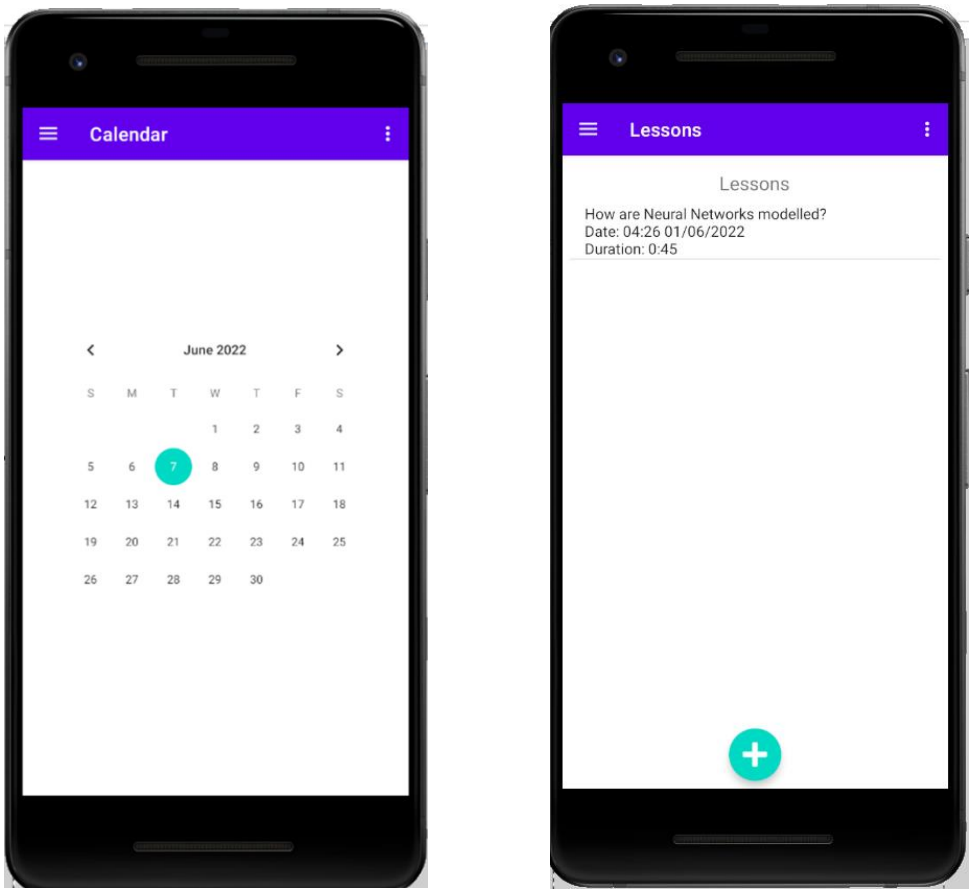

Рисунок 17 – Приклад переходу до управління уроками

Наступним кроком через метод onCreateView() ініціалізуємо байдинг і отримуємо рут. Далі йде отримання в'ю календаря з id

```
 Public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState)
    {
        binding = FragmentCalendarBinding.inflate(inflater,
container, false);
         View root = binding.getRoot();
          CalendarView calendarView = binding.calendarView;
```

Створюємо джава-календар для того, щоб отримати конкретні дані з календаря. Виставляємо поточну дату в календарі і виставляємо слухач на натискання дати на календарі.

```
        Calendar calendar = Calendar.getInstance();
            calendar.setTimeInMillis(calendarView.getDate());

    // створюємо дані календаря
    calendarDate = New CalendarDate(calendar.get(Calendar.YEAR),
    calendar.get(Calendar.MONTH),
    calendar.get(Calendar.DAY_OF_MONTH));
            calendarView.setOnDateChangeListener
            (
                (view, year, month, day) ->
                {
```

Створюємо оновлені дані календаря та переходимо у фрагмент управління уроками:

```
CalendarFragment.calendarDate = New CalendarDate(year, month,
                                        day);
Navigation.findNavController(view).navigate(R.id.nav_lessons);
                }    );
          return root;
         }
    public void onDestroyView()
    {
        super.onDestroyView();
        binding = null;
    }
}
```

3.3 Опис реалізації фрагменту «Lessons»

Binding (прив'язка уявлень) – необхідна для спрощення написання коду, що взаємодіє з уявленнями. Як тільки прив'язка представлення включена до модуля, він створює клас прив'язки для кожного файлу макета XML, що є у цьому модулі. Примірник класу прив'язки містить прямі посилання на всі уявлення, що мають ідентифікатор у відповідному макеті.

Розглянемо клас «LessonsFragment»: тут є поле для байдингів та дані календаря:

```
    public class LessonsFragment extends Fragment
    {
        private FragmentLessonsBinding binding;
```

```
    private CalendarDate calendarDate;
```

Далі у методі onCreateView() створюємо байдинг та отримуємо з інтенту екстра дані на ім'я.

```
 Public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState)
    {
        binding = FragmentLessonsBinding.inflate(inflater,
container, false);
        View root = binding.getRoot();
        calendarDate = (CalendarDate)
getActivity().getIntent().getSerializableExtra("date");
```

Після цього оновлюємо уроки і виставляємо слухач на кнопку додавання нового уроку.

Далі створимо інтент для переходу на активіти по оновленню/створенню уроку. Для цього кладемо екстра дані в інтент – дату та стартуємо нову активіту:

```
updateItems();
    binding.addNewItemButton.setOnClickListener
    (
        (view) ->
        {
            Intent intent = new Intent(getActivity(),
                                       UpdateLesson.class);
            intent.putExtra("date", calendarDate);
            getActivity().startActivity(intent);
        }
    );
    return root;
}


public void onResume()
{
    super.onResume();
    updateItems();
}
```

Через метод private void updateItems() отримуємо доступ до репозиторію уроків. При умові якщо даних календаря немає, то не фільтруємо уроки і отримуємо все, що є в БД, якщо є – отримуємо час початку дня за заданою датою та фільтруємо уроки по заданому ренджу:

```
    {  Lesson[] lessons;
        try (Repository<Lesson> repository = new
Repository<>(Lesson.class, getActivity()))
        {
         if (calendarDate == null)
         lessons = repository.getEntities().toArray(new
Lesson[0]);
            else
            {
LocalDateTime start = LocalDateTime.of(calendarDate.year,
calendarDate.month, calendarDate.day, 0, 0);
       LocalDateTime end =
LocalDateTime.of(calendarDate.year, calendarDate.month,
calendarDate.day, 23, 59);
     lessons = repository.getEntities().stream()filter((l) ->
l.dateTime.isAfter(start) && l.dateTime.isBefore(end))
                    .toArray(Lesson[]::new);
            }
        }
```

Наступним кроком слід створити адаптер. Після цього отримуємо лист та виставляємо адаптер з слухачем на урок при натисканні:

```
ArrayAdapter<Lesson> adapter = new ArrayAdapter<>(getActivity(),
android.R.layout.simple_list_item_1, lessons);

ListView itemsLV = binding.activityItemsList;
    itemsLV.setAdapter(adapter);
    itemsLV.setOnItemClickListener
            (
                (parent, view, position, id) ->
                {
```

Отримуємо натиснутий урок і створюємо інтент. Далі кладемо урок у екстра дані та стартуємо активіті:

```
Lesson lesson = (Lesson) parent.getAdapter().getItem(position);
```

```
  Intent intent = new Intent(getActivity(), UpdateLesson.class)
            intent.putExtra("lesson", lesson);
   getActivity().startActivity(intent);
             }  ); }


    public void onDestroyView()
    {
        super.onDestroyView();
        binding = null;
    }
}
```

В результаті, якщо у мобільному додатку натиснути кнопку «Добавить», то на екрані користувачу зявиться наступне повідомлення щодо назви уроку, суми оплати за урок, дату та час заняття (рис. 18).

Якщо натиснути кнопку «Створити», то отримаємо наступну помилку, оскільки тривалість повинна бути мінімум як 30 хвилин (рис. 19).

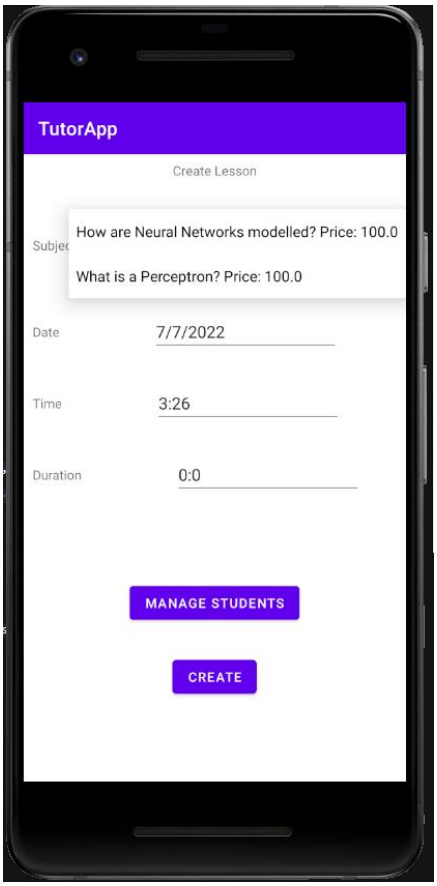

Рисунок 18 – Приклад повідомлення з розкладом

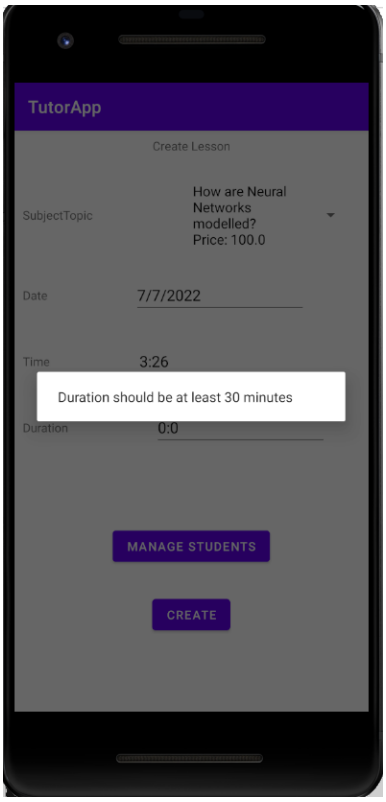

Рисунок 19 – Помилка при створенні уроку

3.3 Логіка оновлення/створення уроку

На першому кроці слід обрати час (Select time) та тривалість (Select duration) для запланового уроку, як показано на рисунку 20.

Опишемо код реалізації для оновлення записів уроку. Для цього використовуємо клас «UpdateLesson», завдяки йому виставляємо слухач на дату, на деякий час та тривалість:

```
public class UpdateLesson extends AppCompatActivity
{
    private DatePickerDialog.OnDateSetListener dateListener;
    private TimePickerDialog.OnTimeSetListener timeListener;
    private TimePickerDialog.OnTimeSetListener
durationListener;
    private LocalDateTime localDateTime;
    private Duration lessonDuration = Duration.ZERO;
```

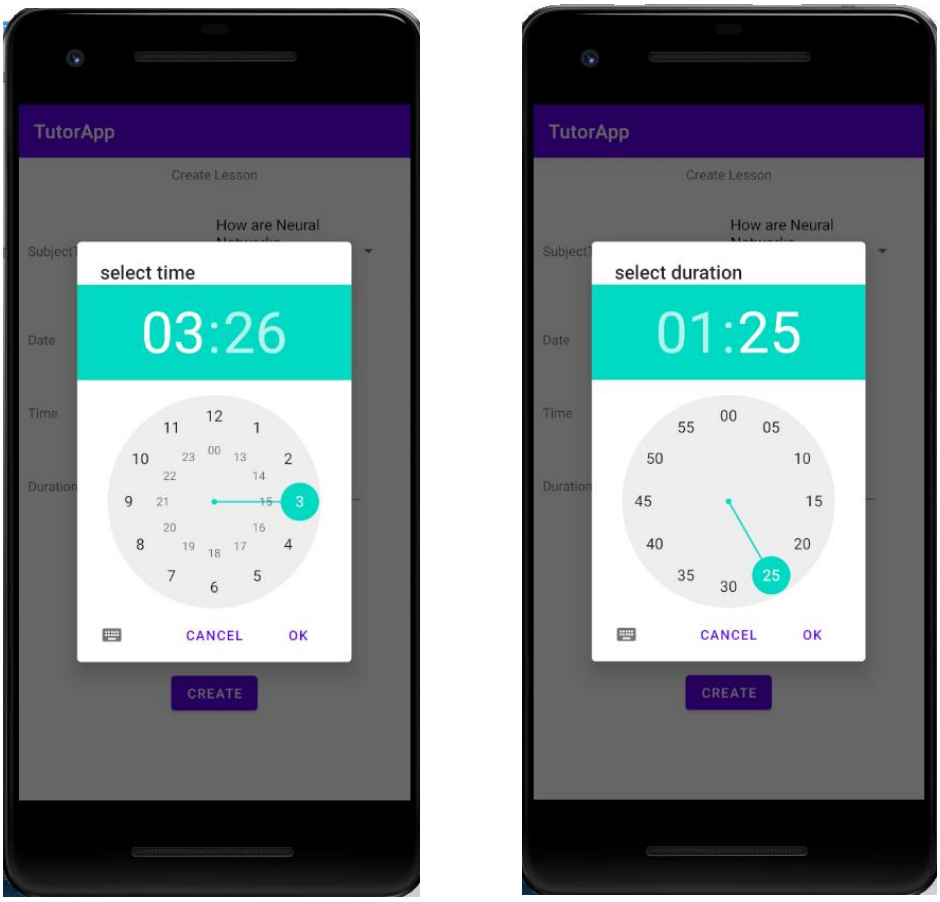

Рисунок 20 – Приклад фіксування дати та тривалості уроку

Далі через метод onCreate() отримуємо дані календаря, а також отримуємо урок, якщо він був переданий в інтенді – інакше створюємо новий:

```
    protected void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_update_lesson);

    CalendarDate calendarDate = (CalendarDate)
getIntent().getSerializableExtra("date");

    Lesson lesson = getIntent().hasExtra("lesson")?(Lesson)
getIntent().getSerializableExtra("lesson"):new Lesson();
```

Отримуємо назву активіті, спінер для відображення предметів (дропдаун), дату та час, тривалість.

```
        TextView title = findViewById(R.id.updateLessonTitle);
        Spinner spinner = findViewById(R.id.subjectTopicSpinner);
        EditText date = findViewById(R.id.lessonDate);
        EditText time = findViewById(R.id.lessonTime);
        EditText duration = findViewById(R.id.lessonDuration);
```

Якщо дані календаря є, створюємо локальну дату з його основі, а інакше створюємо дату на даний момент та отримуємо доступ до тем

```
if (calendarDate! = null)
           localDateTime = LocalDateTime.of(calendarDate.year,
 calendarDate.month, calendarDate.day, 9, 0);
       else
           localDateTime = LocalDateTime.now();
       SubjectTopic[] subjectTopics;
       try (Repository<SubjectTopic> subjectTopicRepository = New
 Repository<>(SubjectTopic.class, this))
       {
     subjectTopics =
 subjectTopicRepository.getEntities().toArray(new
 SubjectTopic[0]);
       }
```

Створюємо адаптер для тем та тип відображення в адаптері як у дропдауні.

```
ArrayAdapter<SubjectTopic> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1,subjectTopics);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
```

Далі ставимо спинеру адаптер, слухач на натискання дати та створюємо діалог вибору дати та виставляємо там дані локальної дати.

```
  spinner.setAdapter(adapter);
    date.setOnClickListener
     (
      (event) ->
       DatePickerDialog datePickerDialog = New DatePickerDialog
            (this,
             android.R.style.Theme_Holo_Light_Dialog_MinWidth,
                     dateListener, localDateTime.getYear(),
                    localDateTime.getMonth().getValue(),
```

```
                localDateTime.getDayOfMonth()
            );
```

Виставляємо режим відображення й показуємо. У слухачі дати читаємо дані і на їхньому оновлення оновлюємо лише рік/місяць/день: виставляємо відповідні параметри у текстовому полі дати.

```
 datePickerDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
   datePickerDialog.show();

        dateListener = (datePicker, year, month, day) ->
        {
         localDateTime = LocalDateTime.of(year, month - 1, day,
            localDateTime.getHour(), localDateTime.getMinute());
            date.setText(localDateTime.getDayOfMonth() + "/" +
            (localDateTime.getMonth().getValue() + 1) + "/" +
            localDateTime.getYear());
        };
        time.setOnClickListener
        (       (event) ->
            {TimePickerDialog timePickerDialog = New
            TimePickerDialog(this, timeListener,
            localDateTime.getHour(), localDateTime.getMinute(),
            true);
                timePickerDialog.setTitle("select time");
                timePickerDialog.show();
            }   );
```

Далі у слухачі дати читаємо дані і на їхньому оновлення оновлюємо лише годинник/хвилини та виставляємо відповідні параметри у текстовому полі часу, створюємо діалог вибору часу та виставляємо там дані локальної дати і все показуємо.

У слухачі дати читаємо дані і на їхньому носі оновлюємо тривалість і виставляємо відповідні параметри у текстовому полі тривалості

```
    durationListener = (timePicker, hour, minute) ->
        {  lessonDuration = Duration.ofMinutes(hour * 60 +
            minute);
            duration.setText(lessonDuration.toHours() + ":" +
            (lessonDuration.toMinutes() % 60));
        };
```

Якщо урок існує в БД, тоді викладаємо заголовок активіті як "оновити", а інакше виявляємо заголовок активіті як "створити".

```
if (Repository.findIdField(lesson).isSet())
            title.setText("Update Lesson");
        else
            title.setText("Create Lesson");
```

Далі виставляємо дані текстових полів на підставі локальної дати та тривалості і так само на підставі того чи створюється урок або оновлюється виставляємо текст кнопки:

```
        date.setText(localDateTime.getDayOfMonth() + "/" +
(localDateTime.getMonth().getValue() + 1) + "/" +
localDateTime.getYear());
        time.setText(localDateTime.getHour() + ":" +
localDateTime.getMinute());
        duration.setText(lessonDuration.toHours() + ":" +
(lessonDuration.toMinutes() % 60));
        button button = findViewById(R.id.createLesson);
        if (Repository.findIdField(lesson).isSet())
            button.setText("Update");
        else
            button.setText("Create");
```

Слухач на кнопці створити/оновити реалізовано наступним чином: отримуємо вибрану тему, якщо тема не задана, показати помилку:

```
button.setOnClickListener
        ( (event) ->
            { SubjectTopic subjectTopic = (SubjectTopic)
spinner.getSelectedItem();
     if (subjectTopic == null)
        { ErrorAlert.show(this, "Subject topic is not set");
return; }
```

Далі задаємо всі параметри в уроці (дата, тема та інше) і перевіряємо тривалість уроку, вона має бути не менше 30 хвилин:

```
   lesson.dateTime = localDateTime;
```

```
                lesson.subjectTopicId=subjectTopic.id;
                lesson.duration = lessonDuration;
                if (lessonDuration.toMinutes() % 60 < 30 &&
lessonDuration.toHours() < 1)
                {
                    ErrorAlert.show(this, "Duration should be at
least 30 minutes");
                    return;
                }
```

Шукаємо урок у БД і закриваємо активіті:

```
        try (Repository<Lesson> repository=new
Repository<>(Lesson.class, this))
                {repository.add(lesson, true);}
                finish();
            });
```

Приклад UI зі списком тем уроків та активіті з оновлення уроків на рисунку 21:

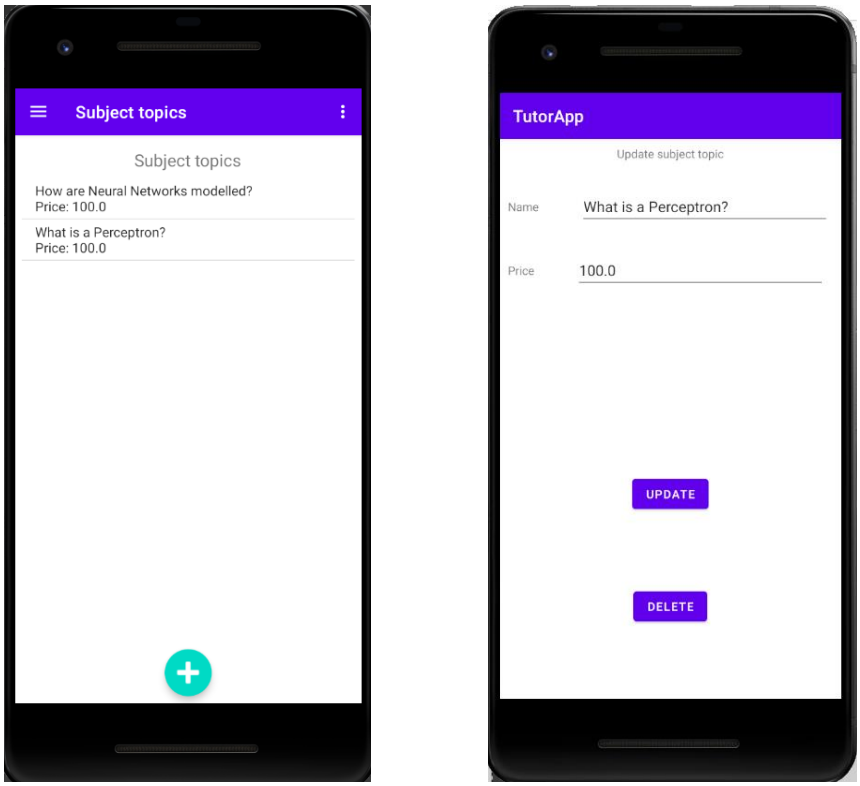

Рисунок 21 – Приклад списку та оновлення уроків

На рисунку 22 відображено скрін активіті зі списком студентів викладача. При натискані на відповідний запис викладач може перейти до повного запису інформації щодо обранного учня.

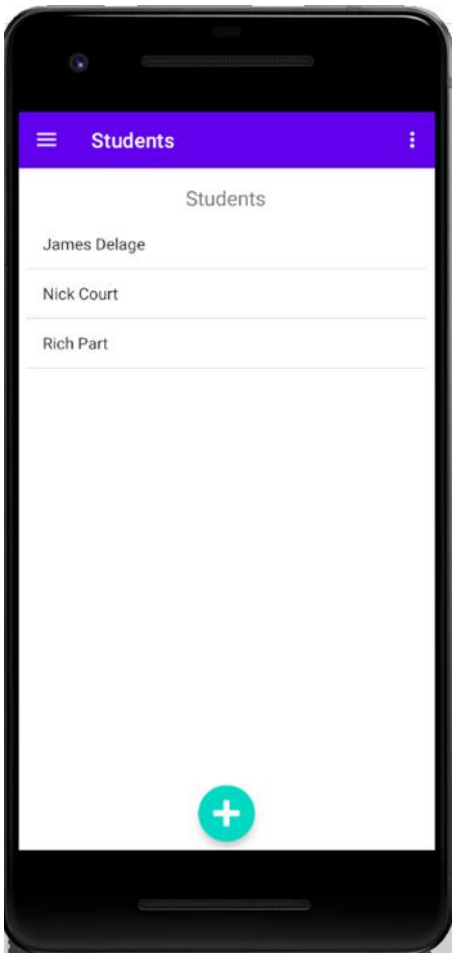

Рисунок 22 – Закладка «Students»

Для оновлення інформації по студентам реалізована закладка «Оновлення» (рис. 23).

Кнопка з управління студентами в «Уроки» реалізована наступним чином: слід перенаправити на активіті вибору студентів на урок, а також поставити поточний урок у екстра дані.

```
button manageStudents = findViewById(R.id.manageStudentsButton);
        manageStudents.setOnClickListener
```

```
        (   (event) ->
            {   Intent intent = new Intent(this,
ManageStudentsActivity.class);
                intent.putExtra("lesson", lesson);
                this.startActivity(intent);
            } );
```

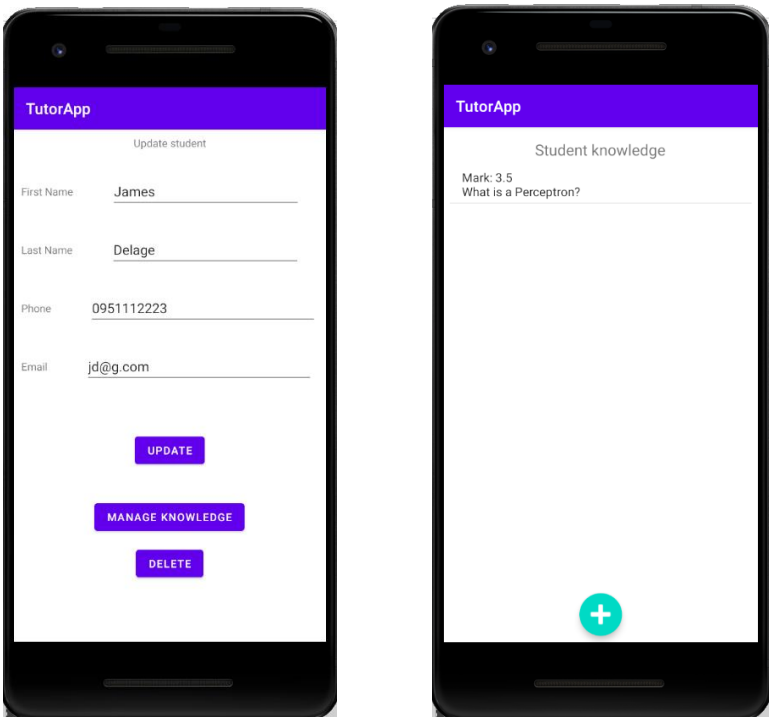

Рисунок 23 – Вкладка на оновлення запису по студентам

Якщо урок створюється, тоді кнопка delete буде невидима:

```
Button delete = findViewById(R.id.deleteLesson);
       if (Repository.findIdField(lesson).isSet())
           delete.setVisibility(View.VISIBLE);
       else
           delete.setVisibility(View.GONE);
          delete.setOnClickListener
   ( (event) -> {// видаляємо урок з Бд
try(Repository<Lesson>repository=newRepository<>(Lesson.class,
                this))
               {   repository.remove(lesson);
                   finish();                } }
```

Після кожного заняття викладач може виставити оцінку успіху для студента, як показано на рисунку 24.

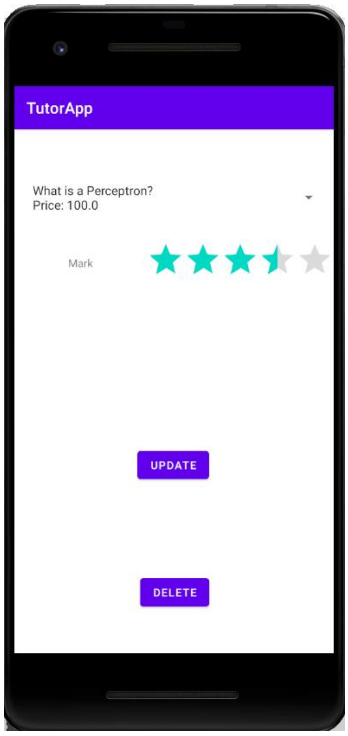

Рисунок 24 – Оновлення оцінки студента

На рисунку 25 представлено діаграму сутностей для мобільного додатку.

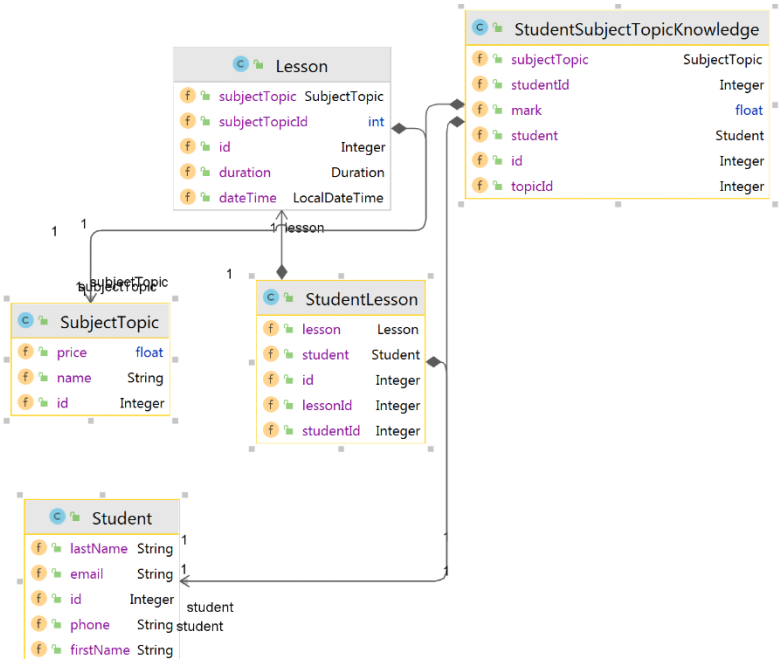

Рисунок 25 – Діаграма сутностей

Всі класи системи представлено на рисунку 26:

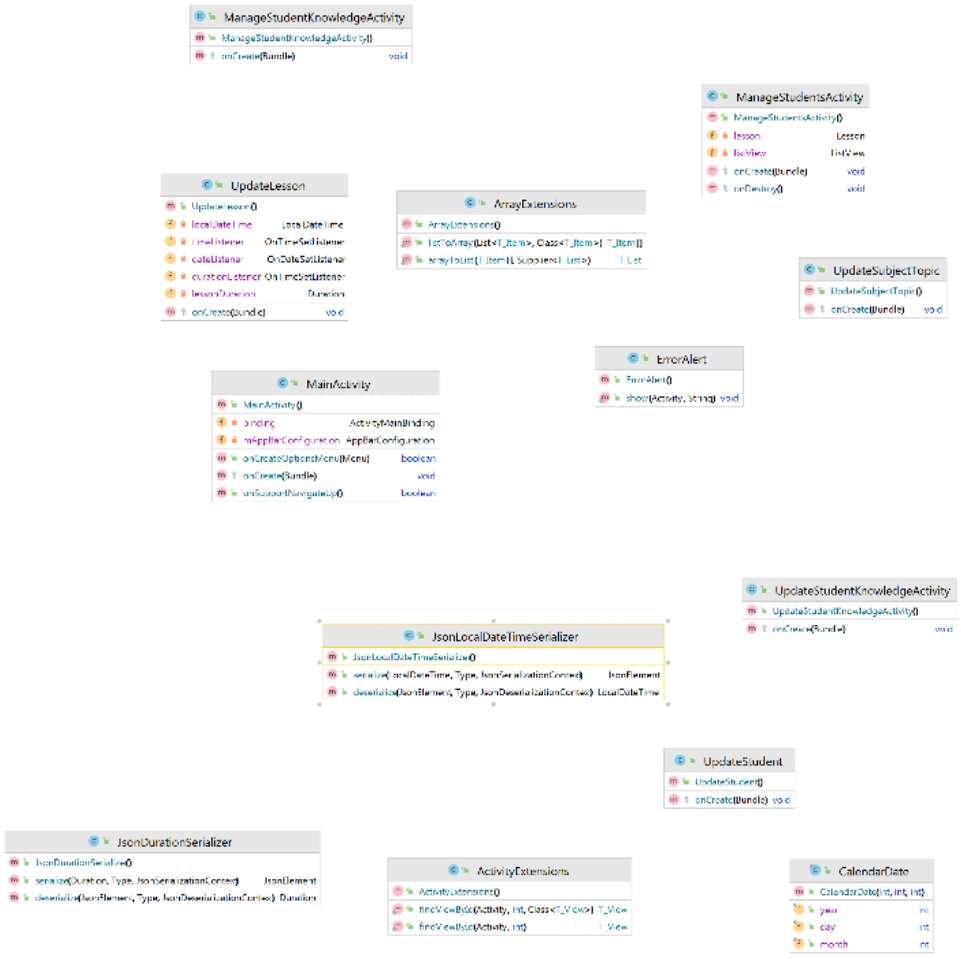

Рисунок 26 – Всі класи мобільного додатку

ВИСНОВКИ

Швидкими темпами зростає попит на розробку мобільних додатків. Зараз існує багато інструментальних засобів розробки та технлогій, за допомогою яких можна створити такий програмний продук.

Для релізації дипломного проекту були обрані Android Studio, та мова програмування Java. На стадії проектування – уніфікована мова моделювання UML та середа проектування Justinmind для UI design.

Мобільний додаток для репетитора допоможе користувачу вести графік своїх занять, список учнів, відмічати рівень їх успішності навчання та багато іншого. Реалізація логіки ядра додатку дозволяє у подальшому вносити додатковий функціонал.

На даний час мобільний додаток готовий для використання та не потребує платного завантаження чи авторизації при його використанні.

СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tutor App Development: 15 Must-Have Features For Your Tutor App. URL: https://savvycomsoftware.com/blog/tutor-app-development/ (дата звертання 21.04.2022)
2. 10 best teacher apps for Android. URL: https://www.androidauthority.com/best-teacher-apps-android-584742/ (дата звертання 21.04.2022)
3. Light – tutor's calendar. URL: https://apps.apple.com/ua/app/light-tutors-calendar/id1535036589 (дата звертання 25.04.2022)
4. 10 Best Android Apps for Teachers. URL: https://www.educationalappstore.com/best-apps/5-best-android-apps-for-the-classroom (дата звертання 25.04.2022)
5. Лучшие средства разработки под Андроид. URL: https://medium.com/nuances-of-programming/лучшие-средства-разработки-под-андроид (дата звертання 04.05.2022)
6. Разработка мобильных приложений от А до Я: полный гайд. URL: https://dan-it.com.ua/blog/razrabotka-mobilnyh-prilozhenij-ot-a-do-ja-polnyj-gajd/ (дата звертання 10.05.2022)
7. 12 ключевых инструментов и ресурсов для разработки под Android. URL: https://senior.ua/articles/12-klyuchevyh-instrumentov-i-resursov-dlya-razrabotki-pod-android (дата звертання 14.05.2022)
8. Android Studio. Установка. URL: https://idf.ua/wp-content/uploads/2020/05/Install_Android_Studio_Guide.pdf (дата звертання 14.05.2022)
9. Learn Java For Android App Development – A Complete Guide. URL: https://www.geeksforgeeks.org/learn-java-for-android-app-development-a-complete-guide/ (дата звертання 18.05.2022)

10. PostgreSQL: The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/ (дата звертання 18.05.2022)
11. How To Connect An Android Project To A Postgresql Database. URL: https://medium.com/cyber-explorer/how-to-connect-an-android-project-to-a-postgresql-database-663cb0f5ba19 (дата звертання 24.05.2022)
12. App Manifest Overview. URL: https://developer.android.com/guide/topics/manifest/manifest-intro (дата звертання 30.05.2022)