МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»
освітньо-кваліфікаційного рівня «магістр»

Виконавець роботи: студент групи ІТ.м-91 Жук Олена Костянтинівна

Кваліфікаційну роботу
захищено на засіданні ЕК
з оцінкою _____ «___» грудня 2020 р.

Науковий керівник _____ к. т. н., доц., Марченко А. В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП
_____В. В. Шендрик
«____» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА СТУДЕНТУ

Жук Олена Костянтинівна

1 Тема роботи *«Мобільний додаток підтримки діяльності «Громадської організації «Клуб спортивного танцю «Силует»*

керівник роботи *Марченко Анна Вікторівна, к. т. н. , доцент* _____,

затверджені наказом по університету від « 26 » листопада 2020 р. № 1824-III

2 Строк подання студентом роботи « » _____ 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку мобільного додатку, матеріали

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі та методи дослідження, моделювання мобільного додатку(МД), розробка МД підтримки діяльності танцювального клубу

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета і задачі, огляд мобільних додатків підтримки діяльності танцювальних клубів, порівняння знайдених мобільних додатків, функціональні вимоги, схема бази даних, контекстна діаграма, діаграма декомпозиції, діаграма варіантів використання, перелік програмних засобів реалізації, приклади реалізації, акт впровадження, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Приміт ка
1	Ідентифікація мети МД	20.04.20-20.04.20	
2	Пошук аналогів	21.04.20-24.04.20	
3	Визначення вимог	27.04.20-08.05.20	
4	Функціональні вимоги	11.05.20-14.05.20	
5	Нефункціональні вимоги	15.05.20-20.05.20	
6	Визначення засобів для реалізації	21.05.20-28.05.20	
7	Планування структури WBS	29.05.20-08.06.20	
8	Планування структури OBS	09.06.20-17.06.20	
9	Створення календарного плану	18.06.20-06.07.20	
10	Розрахунок фінансів	07.07.20-23.07.20	
11	Виявлення ризиків	24.07.20-11.08.20	
12	Налаштування середовища розробки	12.08.20-21.08.20	
13	Створення бази даних	24.08.20-08.09.20	
14	Проектування інтерфейсу МД	09.09.20-06.10.20	
15	Реалізація інтерфейсу доступу до МД	07.10.20-10.11.20	
16	Тестування методом чорного ящику	11.11.20-19.11.20	
17	Виправлення виявлених помилок	20.11.20-16.11.20	
18	Розробка пояснювальної записки	27.11.20-09.12.20	
19	Створення настанови з використання	10.12.20-16.12.20	
20	Архівація	17.12.20-21.12.20	

Студент _____ Жук О. К.
(підпис)

Керівник роботи _____ к. т. н. доц. Марченко А. В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: «Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 31 найменувань, додатків. Загальний обсяг роботи − 106 сторінок, у тому числі 70 сторінок основного тексту, 3 сторінки списку використаних джерел, 35 сторінок додатків.

Перший розділ описує виконаний аналіз предметної області та мобільних додатків, схожих на розроблюваний , завдяки чому чітко зрозуміло актуальність розробки мобільного додатку(МД). Також розділ визначає головні аспекти функціонування МД.

Другий розділ включає в себе визначення головної мети дипломної роботи, основних задач та функціоналу. Крім того, наведено вибір методів та засобів реалізації МД.

Третій розділ визначає опис функціонального моделювання процесу реалізації МД, моделювання бази даних та варіантів використання. Також описані основні принципи функціонування МД.

У четвертому розділі наведено розробку мобільного додатку, що є результатом використання обраних раніше методів та засобів реалізації. Детально описуються етапи розробки та результати реалізації МД.

У результаті розробки отримано мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует», який забезпечує автоматизовану підтримку необхідних бізнес-процесів діяльності танцювального клубу.

Ключові слова: мобільний додаток, бізнес-процес, танцювальний клуб, КСТ Силует, танцівник, тренер.

ЗМІСТ

ВСТУП

За останні пару років спостерігається колосальне зростання розвитку мобільних додатків. Вихід мобільних пристроїв на ринок спричинив розробку різних видів мобільних додатків. Мобільні технології створили серйозну зміну в нашому повсякденному житті. У сучасному стрімкому світі мобільні послуги є двигуном, який забезпечує практично всі види діяльності, включаючи освіту, зв'язок, економіку, комерцію, банківську справу, політику, охорону здоров'я, сільське господарство та соціальну діяльність [1, 2, 3].

Розробка мобільних додатків(МД) - це процедура, за допомогою якої прикладне програмне забезпечення розробляється та створюється для полегшення роботи служб мобільних технологій [1]. Швидке зростання ринку мобільних телефонів та технологічні досягнення призвели до великого попиту на мобільні програми. Отже, прогрес у галузі розробки мобільних додатків є безпрецедентним.

Кожен проект програмного забезпечення має унікальну мету та унікальні особливості, тому процес розробки та фактори, які слід враховувати, можуть відрізнятися залежно від очікувань та цілей зацікавленої сторони. Дизайн та розробка мобільного додатка залежать від вимог зацікавлених сторін та очікуваних послуг. Існує кілька видів мобільних додатків з різними функціональними можливостями, включаючи мобільне навчання, мобільне здоров'я, мобільне сільське господарство, мобільну комерцію, мобільний уряд, мобільний банкінг та мобільну соціальну діяльність [1, 2]. Ці мобільні додатки розробляються різними організаціями або для різних організацій із відповідними цілями.

Розвиток інформаційних технологій може спрощувати управління будь-якої організації шляхом автоматизації виконання бізнес-процесів. На даний час менеджмент клубу спортивного танцю (КСТ), на жаль, відбувається лише у друкованому вигляді та в не дуже зручний та актуальний спосіб. Тому дуже вагомим є розроблення мобільного додатку для підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует».

Під діяльністю КСТ, яка має бути автоматизована, мається на увазі запис на індивідуальні заняття, ведення обліку відвідування, оплата послуг.

Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» розроблено не тільки для тренерів, але й для членів танцювального клубу, що забезпечує дуже зручний та швидкий запис на уроки, оплату занять та представляє засіб оперативної комунікації.

У результаті розробки отримано мобільний додаток, що дозволяє автоматизовано підтримувати всі необхідні бізнес-процеси діяльності КСТ Силует.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

На сьогоднішній день розробка мобільних додатків з примхи переросла в реальну необхідність. Якісно спроектований і налагоджений продукт дозволить не тільки спростити роботу окремої людини або компанії в цілому, але і вирішувати масу бізнес-задач, перебуваючи практично в будь-якій точці земної кулі.

Після прийнятого рішення про розробку мобільного додатку, почався аналіз платформ, під яку доцільно розробити додаток в даний момент часу.

На першому етапі вивчалася інформація про крос-платформні розробки, але після певного аналізу і вивчення певної інформації по розробці, був зроблений висновок, що так як відокремлені платформи мають свої особливості, необхідно почати розробку з якоїсь однієї платформи під мобільні додатки.

Вибрана ОС Android.

Розробка мобільних додатків для Android - це процес створення нових додатків для операційної системи Android. Розробка мобільних додатків перетворюється на домінуючу форму цифрової взаємодії. Клієнти в сучасному світі пересуваються, і вони використовують платформи мобільних додатків. Мобільні додатки Android поступово беруть на себе важливу роль у бізнесі з великою базою клієнтів, а також робочої сили, залежно від зручності доступу до інформації та рішень на вимогу. Операційна система Android пропонує гнучкість та підтримку сторонніх програм.

У цю цифрову еру розробка мобільних додатків діє як віртуальний асистент, який цілком і цілодобово передає бізнес аудиторії. Коли потрібно підібрати відповідну платформу, це дуже важливо для бізнесу та клієнтів. У світі додатків, Android надає найкращий ключ для значної частки мобільного ринку, який щодня зростає.

Змінилася парадигма у бік рішень для мобільності завдяки можливості мобільних додатків запускати персоніфікований зв'язок із клієнтами. Серед усіх смартфонів, що продаються у всьому світі, більшість пристроїв мають операційні системи Android (рис. 1.1) [4]. Пристрої, що підтримуються Android, є відносно економічними та доступними, що робить їх бажаним вибором для клієнтів у країнах із економікою, що розвивається. Пристрої Android у світі надають бізнесу велику базу користувачів для досягнення своїх бізнес-цілей.

Рисунок 1.1 – Частка ринку мобільних операційних систем у всьому світі - вересень 2020 р

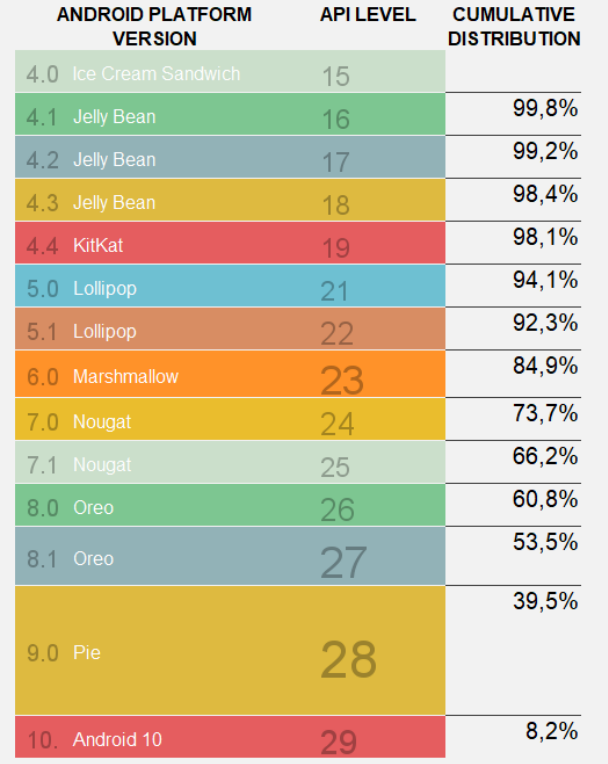
Вибір Android, як платформи для розробки додатків, робить його чудовим вибором для підприємств, які прагнуть розширити свої можливості на світових ринках. Це пов'язано з тим, що Android є провідною платформою у країнах, що ростуть, і за її підрахунками є приблизно 1,4 мільярда унікальних користувачів, і, звичайно, їх кількість з часом зростає. Отже, за допомогою програми для Android можна скористатися можливістю легкого доступу до масового світового ринку.

Програмування під Android в 2020 році продовжує залишатися актуальною темою. Причиною цього є велика кількість техніки під керуванням даної операційної системи, а так само всі мобільні пристрої на базі Android дозволяють охопити величезну кількість людей по всьому світу.

В даний момент часу операційна система налічує більше 7 різних версій і важливим аспектом при розробці мобільних додатків, є необхідність вибору тих версій яке буде підтримувати програму.

На наступному рисунку (рис. 1.2), представлена інформація про те, як розподіляються додатки на пристроях, з різними версіями платформи. Необхідно

розуміти, що дані зібрані по всьому світу і можуть відрізнятися для відокремлених регіонів. Але для розробки мобільного застосування, яке буде використовуватися всередині організації, цих даних цілком достатньо [5].



ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Рисунок 1.2 – Розподіл додатків щодо версій Android

Виходячи з даних наведених з офіційного сайту для розробників під Android, можна зробити висновок, що майже кожен користувач зможе підібрати пристрій на будь-який смак, що тільки з позитивного боку позначається на виборі даної платформи для розробки мобільного застосування [6].

При виборі платформи, для якої буде реалізовано проект кваліфікаційної роботи магістра, стояв вибір між двома ОС – Android і iOS. Після порівняльного аналізу обрана ОС Android.

Перевагами розробки додатків для Android є [7]:

1) Простота розробки та налаштування

Програмне забезпечення Android - це відкрита екосистема, яка дозволяє розробникам компанії з розробки мобільних додатків для Android вільно отримувати

доступ до бажаних розділів коду Android, які можуть знадобитися для їхніх програм. Мобільні додатки для Android з’явилися як потенційно вигідна угода, оскільки вони створили високопродуктивні продукти для мобільних додатків, які охоплять більшу групу аудиторії, ніж вона може мати на платформах для смартфонів.

Розробка мобільних додатків - це постійно зростаюче середовище, лише шляхом адаптації інноваційних та повторюваних процесів; можна зробити свій охоплення кращим. Маючи постійно зростаючу базу користувачів, платформа Android має вбудований відкритий ринок для розробки інноваційних програм. Без сумніву, Андроїд навантажити додаток потужними та спеціальними функціями, які роблять його найбільш бажаним вибором серед користувачів.

Мобільні додатки Android можуть приносити дохід, надаючи певні преміум-функції, за які користувачі повинні платити.

2) Брендинг

Послуги з розробки мобільних додатків для Android можуть допомогти компаніям здобути акценти уваги та душі споживачів для формування лояльності до бренду. У цю епоху соціальних мереж навіть великі бренди стикаються з труднощами, щоб зберегти лояльність до бренду. Мобільні додатки Android залучають клієнтів за допомогою персоналізованого спілкування та забезпечують ефективне обслуговування клієнтів.

3) Сфера інновацій

Щороку Android висуває інноваційні ідеї та тенденції, що символізують майбутнє. Пристрої та технології, що використовуються користувачами для взаємодії з діловими змінами, що стосуються поведінки та потреб користувачів. Платформа Android має найкращу політику, включаючи розробку додатків, що дозволяє розробникам компанії з розробки мобільних додатків проводити експерименти та пропонувати інноваційні ідеї.

4) Зменшення витрат

Більшість послуг, які надаються в бізнесі, можна надати за допомогою мобільних додатків для Android. Це призведе до того, що не потрібно буде платити

працівникам за виконання цієї конкретної роботи, тобто додаток може з легкістю замінити, наприклад, бухгалтера.

Переходячи до перерахування мінусів даної платформи, варто виділити найістотніші:

1) Велика кількість пристроїв одночасно, може бути значним мінусом, так як для того щоб б написати додаток під всі пристрої необхідно витратити велику кількість часу, а в рамках роботи одного розробника або маленької команди, це просто стає неможливим;

2) Тестування додатків на всіх пристроях, просто не реальна задача в даний момент часу;

3) Велика кількість вірусів, які пишуть зловмисники під Android. Всього за 2016 рік було зафіксовано 13 783 вірусів. Відзначається, що за той же час на iOS було виявлено тільки 9 вірусів. Причому експерти компанії відзначили, що в майбутньому кількість вірусів для ОС від Google буде збільшуватися в геометричній прогресії.

Підводячи підсумок недоліків при розробці на Android, слід відзначити одну важливу метрику: кількість рядків коду. Наприклад, при ідентичній функціональності додатків для обох платформ, в iOS на їх реалізацію потрібно 1 596 рядків коду, включаючи файли заголовків, а при створенні Android версії при тих же функціях знадобилося 2109 рядків. Різниця в 32%.

Виходячи з перерахованого можна відзначити, що при досить вагомих мінусах ОС Android, для розробки мобільного додатку в рамках магістерської роботи, а також, з точки зору використання програми у клубі, це найкраща платформа в 2020 році [8].

Розробка додатків під ОС Android, це більш доступний варіант для студентів університету, так як дана платформа не вимагає спеціального устаткування для збірки додатку на відміну від ОС iOS.

1.2 Аналіз додатків підтримки діяльності клубів спортивного танцю

Перед тим, як почати етап розроблення було проведено аналіз аналогів мобільного додатку. На даний час майже кожен танцювальний клуб ставить за мету реалізацію автоматизованого управління, тому більшість клубів прагне впровадження різноманітних інформаційних систем та мобільних додатків для спрощення бізнес-процесів діяльності студії.

Після проведеного пошуку мобільних додатків та інформаційних систем, в мережі Інтернет, які якимось чином спрощують організацію діяльності танцювального клубу, було знайдено велику кількість додатків, проте не всі є вдалими. Далі коротко описано найбільш відомі та популярні програми, за якими було проведено аналіз.

1. Android – додаток «DanceStudio-Pro» [9]

Даний мобільний додаток це новий та простий спосіб для власників танцювальної студії організовувати комунікацію та роботу зі студентами, класами, концертами, заходами, викладачами, відвідуваністю, навчанням.

Для використання цієї програми потрібна щомісячна передплата, однак можна протестувати програму за допомогою безкоштовного демо-аккаунту. На жаль, протестувати та побачити даний додаток у роботі не вийшло.

Далі показан приклад роботи додатку з офіціального сайту (рис. 1.3)

Рисунок 1.3 – Мобільний додаток «DanceStudio-Pro»

2. Мобільний додаток «Fly Dance Studio» [10]

За допомогою цього мобільного додатка ви можете переглядати розклад занять, реєструватися на заняття, переглядати поточні акції, а також переглядати місцезнаходження студії та контактну інформацію.

Даний додаток розроблено спеціально для танцювальної студії «Fly Dance Studio», що знаходиться в Ірландії, Дублін.

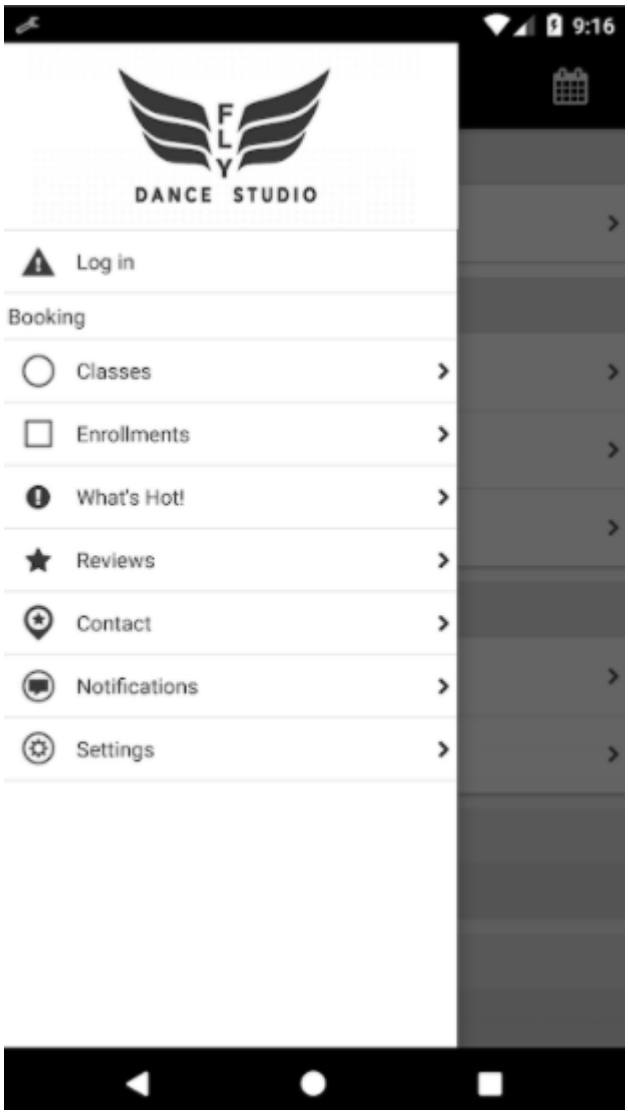


Рисунок 1.4 – Мобільний додаток «Fly Dance Studio»

3. Мобільний додаток «Move Dance Studio» [11]

«Move Dance Studio» (рис. 1.5) було розроблено для клієнтів танцювальної студії, де присутня мобільна реєстрація та оплата навчання. Можна бути в курсі всієї останньої інформації від студії, бачити записи у календарі.

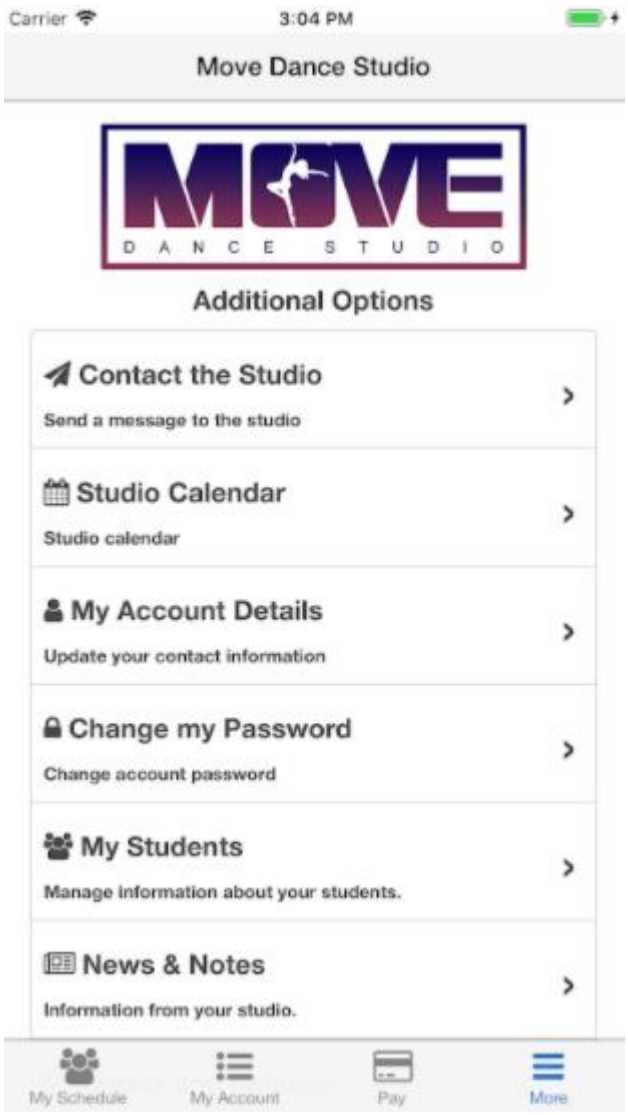


Рисунок 1.5 – Вікно програми «Move Dance Studio»

Усі вище описані мобільні додатки не задовольняють критеріям, за якими проводився пошук програм для підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует». Наприклад, перший мобільний додаток, під назвою «DanceStudio-Pro», зміг би стати кращим кандидатом, якщо б не одне «але», і це щомісячна оплата послуг підтримки, тобто додаток не є безкоштовним. А ось інші описані мобільні додатки розроблені спеціально для окремих танцювальних студій та клубів, що унеможливлює їх використання для потреб клубу спортивного танцю «Силует».

Після знаходження аналогів на розроблюваний мобільний додаток, їх було проаналізовано за декількома критеріями. Для зручності дані аналізу було занесено у таблицю (табл. 1.1). Якщо критерій присутній в мобільному додатку, то ставиться «1», якщо ні – «0». За допомогою даної таблиці можна легко зрозуміти, який мобільний додаток потрібен танцювальному клубу.

Таблиця 1.1 – Порівняльний аналіз мобільних додатків

Критерії	**DanceStudio-Pro**	**Fly Dance Studio**	**Move Dance Studio**	**Власний мобільний додаток**
Зручний та приємний інтерфейс	1	1	0	1
Авторизація та реєстрація тренерів	1	0	0	1
Авторизація та реєстрація клієнтів	0	1	1	1
Запис на заняття	1	1	1	1
Оплата послуг клубу	1	0	1	1
Облік відвідування	0	0	0	1
Доступність за ціною	0	0	0	1
Підсумок	**4**	**3**	**3**	**7**

У результаті проведеного аналізу можна зробити висновок, що для підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» буде оптимально розробити власний мобільний додаток. Дане рішення збереже кошти, які можна буде далі використати для потреб клубу.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі

Головна мета проекту полягає в розробленні мобільного додатку підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» для автоматизації таких бізнес-процесів, як запис членів клубу на індивідуальні заняття, оплата за навчання, облік відвідування танцівників.

Для досягнення мети були визначені такі задачі: аналіз вимог та аналогів, тобто мобільних додатків підтримки діяльності танцювальних клубів, які схожі на розроблюваний додаток, вибір методів та засобів реалізації, розробка бази даних (БД), проектування і моделювання інтерфейсу мобільного додатку, програмна реалізація модулів додатку і тестування роботи отриманого додатку методом чорного ящику [12].

Одним з головних аспектів розробки даного продукту стало те, що це буде не один мобільний додаток, а два. Один додаток з доступом для тренерів, а інший – для членів танцювального клубу. Ці два додатки будуть під’єднуватися до однієї бази даних. Було прийнято рішення розроблювати додаток з клієнт-серверною архітектурою.

Використання розроблюваного мобільного додатку спрямовано на автоматизацію бізнес-процесів клубу спортивного танцю «Силует». Це значно пришвидшить та спростить організацію діяльності клубу і буде краще, чим сьогоденна робота з друкованими паперами.

Для детального опису цілей та задач було сформоване та затверджене технічне завдання, яке знаходиться у додатку А.

2.2 Вибір засобів реалізації

Перед тим, як починати розробку програмної частини Android-додатку, насамперед потрібно обрати засоби реалізації МД. Ґрунтуючись на поставлених меті та задачах, зазначених вище, оптимальними засобами реалізації є:

- середовище розробки Android Studio;
- мова програмування Java;
- мобільна платформа Firebase;
- база даних Cloud Firestore.

1. Середовище розробки Android Studio

Android Studio є офіційним інтегрованим середовищем розробки (IDE) для розробки додатків Android. Воно засновано на IntelliJ IDEA, інтегрованому Java-середовищі розробки програмного забезпечення, і включає засоби редагування коду та інструменти розробника.

Для підтримки розробки додатків в операційній системі Android, Android Studio використовує систему побудови на основі Gradle, емулятору, шаблонів коду та інтеграції Github. Кожен проект в Android Studio має один або кілька способів використання вихідного коду та файлів ресурсів. Ці способи включають модулі додатків Android, модулі бібліотеки та модулі Google App Engine.

Android Studio використовує функцію миттєвого натискання для надсилання змін коду та ресурсів до запущеної програми. Редактор коду допомагає розробнику писати код і пропонувати заповнення та аналіз коду. Потім програми, вбудовані в Android Studio, компілюються у формат APK для надсилання в Google Play Store.

Вперше програмне забезпечення було оголошено на Google I/O у травні 2013 року, а перша стабільна збірка випущена в грудні 2014 року. Android Studio доступне для платформ Mac, Windows та Linux. Дане середовище розробки замінило Eclipse Android Development Tools (ADT) як основну IDE для розробки додатків Android [13]

2. Мова програмування Java

Хорошим підходом до розробки програмного забезпечення є об'єктно-орієнтований підхід(ООП). Java заснована на концепції ООП. Android в значній мірі покладається на основи Java, такі як класи та об'єкти та інші корисні функції ООП.

Java має великий набір бібліотек. Скористатися цими бібліотеками легко. Android SDK містить багато стандартних бібліотек Java. Вони надають функціональні можливості для структури даних, математичних функцій, імплантації графіки та функцій мережі та багато іншого. Таким чином Java допомагає швидко розробляти програми для Android.

Java має незалежну від платформи функцію, тому ця мова використовується для розробки Android. Java є дуже популярною мовою завдяки своїм чудовим можливостям та продуктивності. Java - найкраща та перша мова для розробки власного програмного забезпечення [14].

3. Мобільна платформа Firebase

Firebase - чудовий серверний варіант для розробки додатків. Платформа, що належить Google, протягом багатьох років розвивалася, щоб забезпечити безліч додатків із найсучаснішими функціями. Платформа пропонує безліч інструментів та послуг, які дозволяють розробникам виконувати завдання швидше та ефективніше. Firebase обробляє внутрішні клопоти, тому розробники мають більше часу для створення чудових функцій інтерфейсу для своїх програм.

Деякі інструменти, пропоновані Firebase, включають базу даних, API та серверну інфраструктуру. Отже, розробнику не доводиться турбуватися про управління серверами. Користувачі Firebase також можуть отримати доступ до додаткових першокласних інструментів для створення додатків, розширення бази даних та монетизації її [15].

4. База даних Cloud Firestore

Google Firestore, також відомий як Cloud Firestore, є частиною платформи розробки додатків Google Firebase. Це хмарна база даних NoSQL для зберігання та синхронізації даних. До Firestore можна безпосередньо отримати доступ через мобільні та веб-програми через власні SDK.

Це дозволяє користувачам користуватися опціями SDK Unity, Java, C ++, Go та Node.js, а також пропонує підтримку REST та RPC API. База даних Firestore забезпечує автоматичне масштабування, підвищену продуктивність, простоту використання, а також забезпечує високий рівень надійності.

Firestore допомагає синхронізувати дані між кількома клієнтськими програмами за допомогою прослуховувачів реального часу.Використовує хмарні ідентифікаційні дані та функції управління доступом від Google для процесу автентифікації. Firestore виконує зберігання даних у формі документів, при цьому документи зберігаються в колекціях. Документи підтримують широкий спектр типів даних, таких як вкладені об'єкти, числа та рядки. Firestore має інтеграцію з Google Cloud Platform та Google Firebase. Компанії віддають перевагу Firestore за рівень безпеки та надійності, який пропонується [16].

3 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Структурно-функціональне моделювання процесу підтримки діяльності КСТ Силует

Перед тим, як починати процес розробки мобільного додатку, важливо побудувати структурно-функціональну модель процесу, який виконується. Побудова функціональної моделі була здійснена у програмі ERWin Process Modeler [17]. Так як було прийняте рішення розробки двох окремих мобільних додатків, функціональні моделі побудовані для виконуваних процесів кожного з додатків.

Для мобільного додатку, який використовується членами клубу, тобто танцівниками, процесом є організація відвідування Громадської організації «Клуб спортивного танцю «Силует».

Вхідними даними є:

– Дані про танцівника;
– Дані з бази даних.

Вихідними даними є:

– Запис даних індивідуального заняття в базі даних;
– Запис даних танцівника в базі даних;
– Квитанція про сплату;
– Запис в календарі;
– Запис оцінки роботи тренера в базі даних.

Керуючими елементами процесу в даній функціональній моделі є:

– Графік тренувань;
– Правила відвідування клубу.

Та механізми це:

– Мобільний додаток танцівника;
– Танцівник;

– База даних Firebase.

Контекстна діаграма(IDEF0) [18], що зображує головний процес функціонування мобільного додатку членів клубу, показано на рис. 3.1.

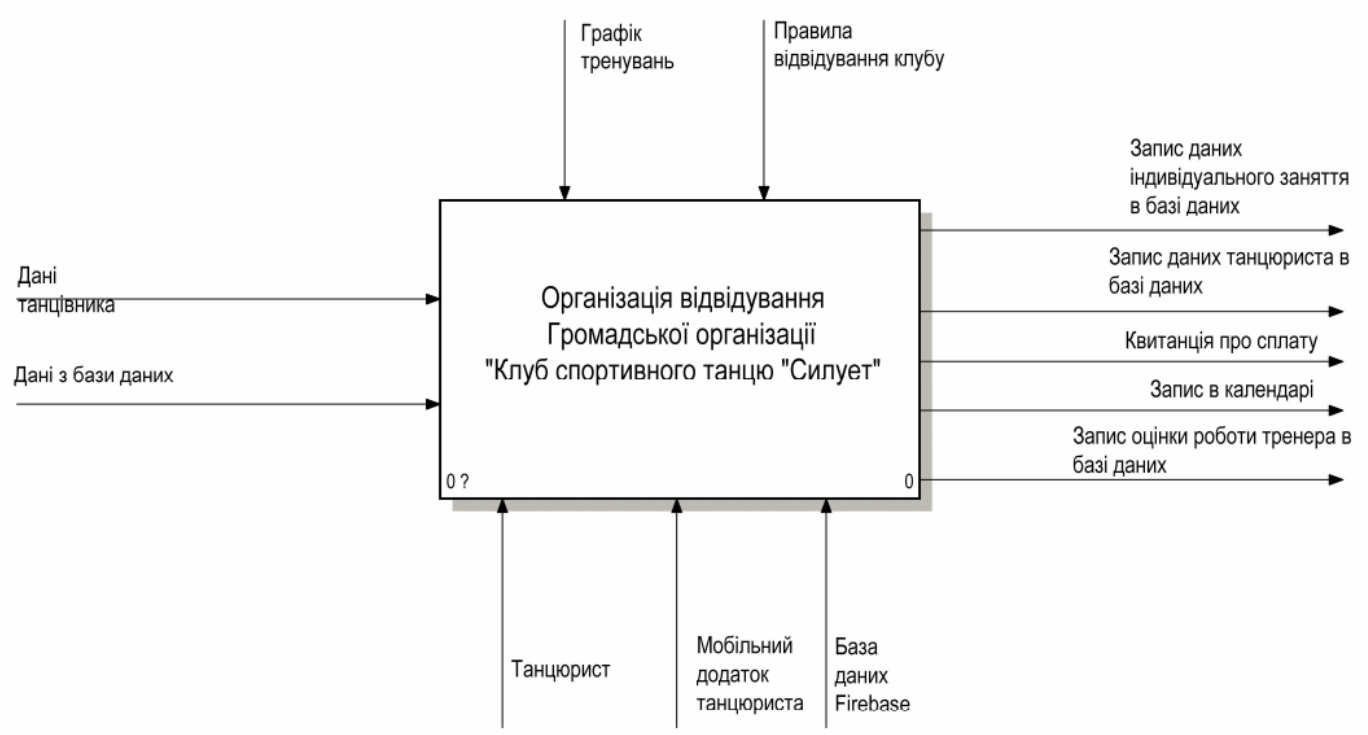


Рисунок 3.1 – Діаграма IDEF0 мобільного додатку членів клубу

Для детального опису роботи мобільного додатку, контекстну діаграму було декомпозовано на п’ять блоків [19]. Тобто додаток має 5 етапів роботи:

– Авторизація танцівника;

– Введення даних танцівника;

– Запис на індивідуальне заняття;

– Оплата навчання;

– Оцінювання тренера.

Характеристика першого етапу «Авторизація танцівника»:

– Вхідні дані: Дані танцівника;

– Вихідні дані: Номер мобільного телефона танцівника;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток танцівника, база даних Firebase, танцівник.

Характеристика другого етапу «Введення даних танцівника»:

– Вхідні дані: Номер мобільного телефона танцівника;

– Вихідні дані: Отримані дані танцівника, запис даних танцівника у базі даних;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика третього етапу «Запис на індивідуальне заняття»:

– Вхідні дані: Отримані дані танцівника;

– Вихідні дані: Запис в календарі, запис даних індивідуального заняття в базі даних;

– Елементи керування: Правила відвідування клубу, графік тренувань;

– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика четвертого етапу «Оплата навчання»:

– Вхідні дані: Запис в календарі;

– Вихідні дані: Запис в базі даних, квитанція про сплату;

– Елементи керування: Правила відвідування клубу, графік тренувань;

– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика п’ятого етапу «Оцінювання тренера»:

– Вхідні дані: Запис в базі даних;

– Вихідні дані: Запис оцінки роботи тренера в базу даних;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Діаграму декомпозиції першого рівня представлено на рис. 3.2.

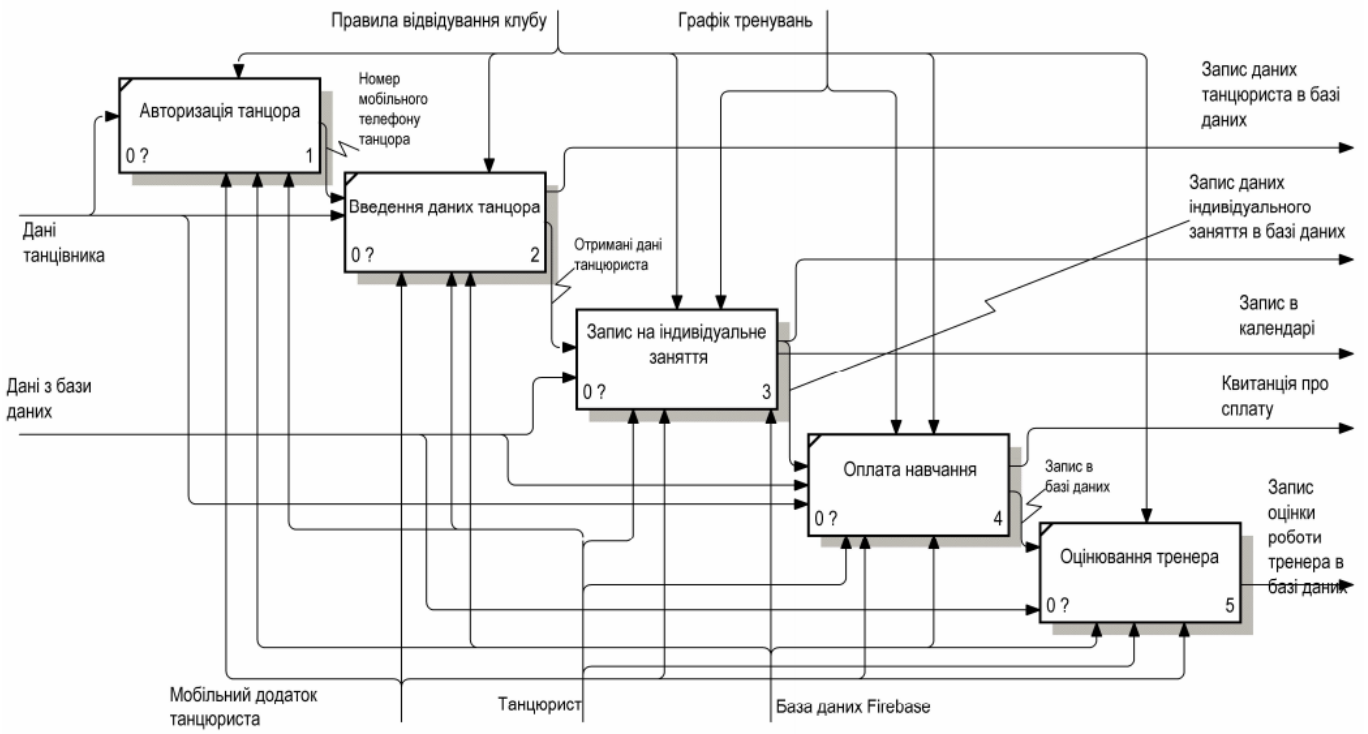


Рисунок 3.2 – Діаграма першого рівня декомпозиції

Для більш детального розуміння роботи додатку було побудовано діаграму другого рівня декомпозиції третього етапу.

Даний етап має 5 блоків:

– Вибір локації танцювального залу;
– Вибір танцювального залу;
– Вибір тренера;
– Вибір часу тренування;
– Підтвердження запису на індивідуальне заняття.

Характеристика першого блоку «Вибір локації танцювального залу»:

– Вхідні дані: Дані з бази даних, отримані дані танцівника;
– Вихідні дані: Обрана локація;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток танцівника, база даних Firebase, танцівник.

Характеристика другого блоку «Вибір танцювального залу»:

– Вхідні дані: Обрана локація, дані з бази даних;
– Вихідні дані: Обраний танцювальний зал;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика третього блоку «Вибір тренера»:

– Вхідні дані: Обраний танцювальний зал, дані з бази даних;
– Вихідні дані: Обраний тренер;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика четвертого блоку «Вибір часу тренування»:

– Вхідні дані: Обраний тренер, дані з бази даних;
– Вихідні дані: Обраний час;
– Елементи керування: Правила відвідування клубу, графік тренувань;
– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Характеристика п’ятого блоку «Підтвердження запису на індивідуальне заняття»:

– Вхідні дані: Обраний час, обраний тренер, обраний зал, обрана локація, отримані дані танцівника, дані з бази даних;
– Вихідні дані: Запис даних індивідуального заняття в базі даних, запис в календарі;
– Елементи керування: Правила відвідування клубу, графік відвідувань;
– Механізми: Мобільний додаток, база даних Firebase, танцівник.

Діаграму представлено на рис. 3.3.

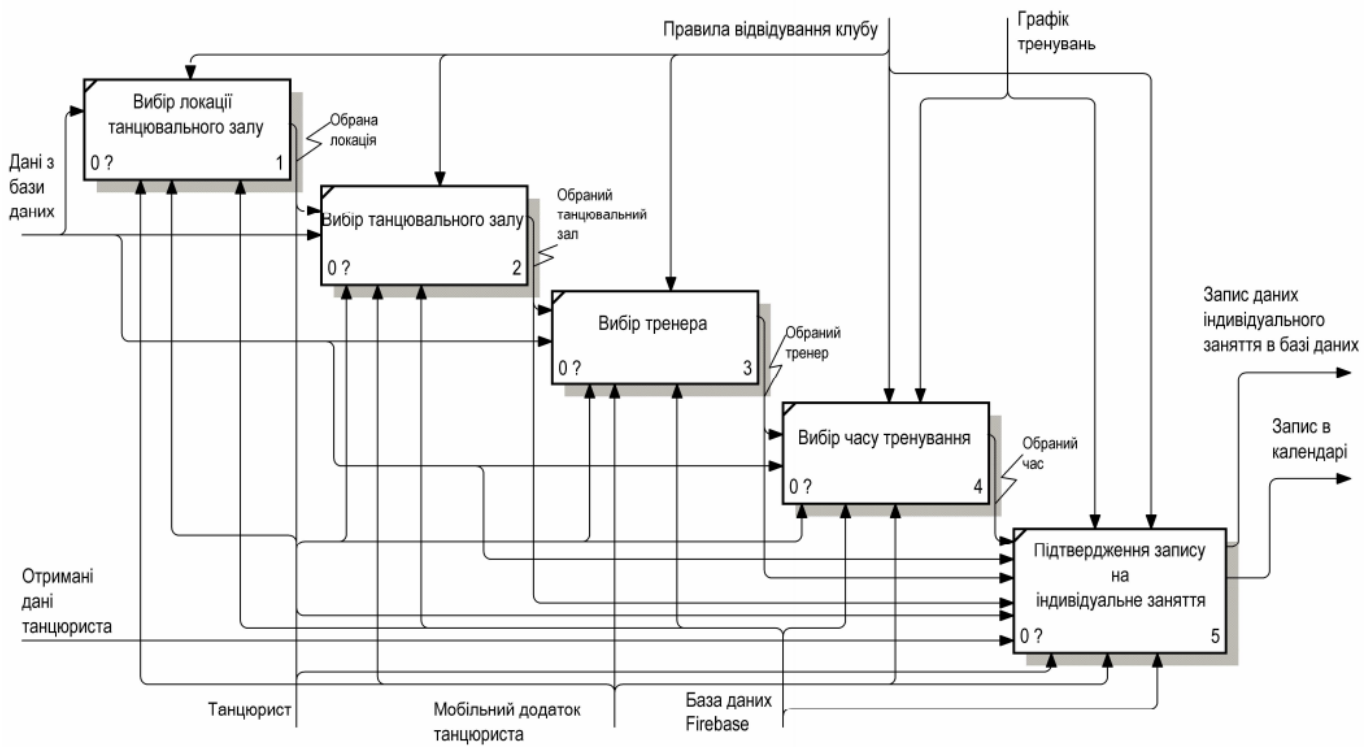


Рисунок 3.3 – Діаграма другого рівня декомпозиції

Для мобільного додатку, який використовується тренерами клубу, процесом є організація проведення тренувань Громадської організації «Клуб спортивного танцю «Силует».

Вхідними даними є:

– Запис в календарі;
– Дані з бази даних.

Вихідними даними є:

– Запис в базі даних;
– Квитанція на оплату.

Керуючими елементами процесу в даній функціональній моделі є:

– Графік тренувань;
– Правила відвідування клубу.

Та механізми це:

– Мобільний додаток тренера;

– Тренер;

– База даних Firebase.

Контекстна діаграма(IDEF0), що зображує головний процес функціонування мобільного додатку тренерів, показано на рис. 3.4.

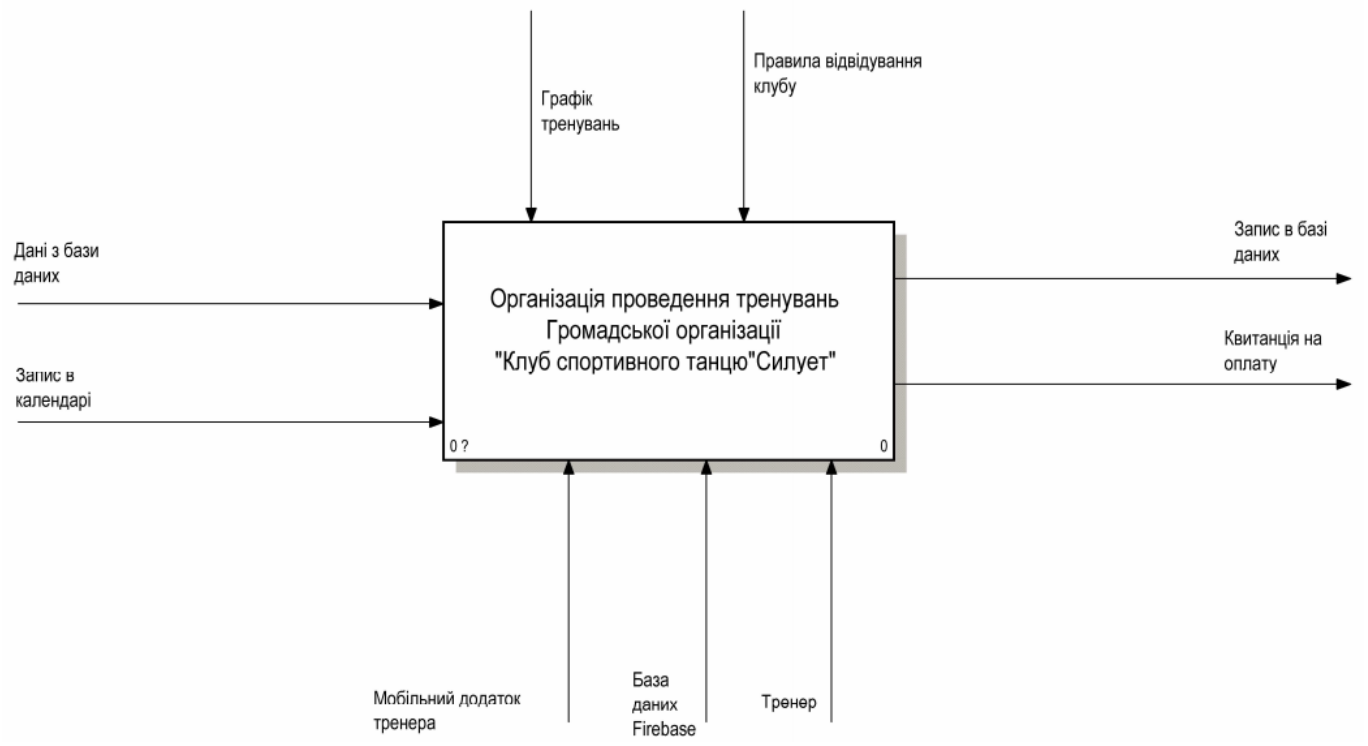


Рисунок 3.4 – Діаграма IDEF0 мобільного додатку тренерів

Для детального опису роботи мобільного додатку, контекстну діаграму було декомпозовано на чотири блоки. Тобто додаток має 4 етапи роботи:

– Вибір локації танцювального залу;

– Вибір танцювального залу;

– Авторизація тренера;

– Підтвердження проведеного індивідуального заняття.

Характеристика першого етапу «Вибір локації танцювального залу»:

– Вхідні дані: Дані з бази даних;

– Вихідні дані: Обрана локація;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика другого етапу «Вибір танцювального залу»:

– Вхідні дані: Обрана локація, дані з бази даних;

– Вихідні дані: Обраний танцювальний зал;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика третього етапу «Авторизація тренера»:

– Вхідні дані: Обрана локація, обраний танцювальний зал, дані з бази даних;

– Вихідні дані: Запис в базі даних, дані тренера;

– Елементи керування: Правила відвідування клубу;

– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика четвертого етапу «Підтвердження проведеного індивідуального заняття»:

– Вхідні дані: Дані тренера, дані з бази даних, запис в календарі;

– Вихідні дані: Запис в базі даних, квитанція про сплату;

– Елементи керування: Правила відвідування клубу, графік тренувань;

– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Діаграму декомпозиції першого рівня процесу «організація проведення тренувань Громадської організації «Клуб спортивного танцю «Силует» представлено на рис. 3.5.

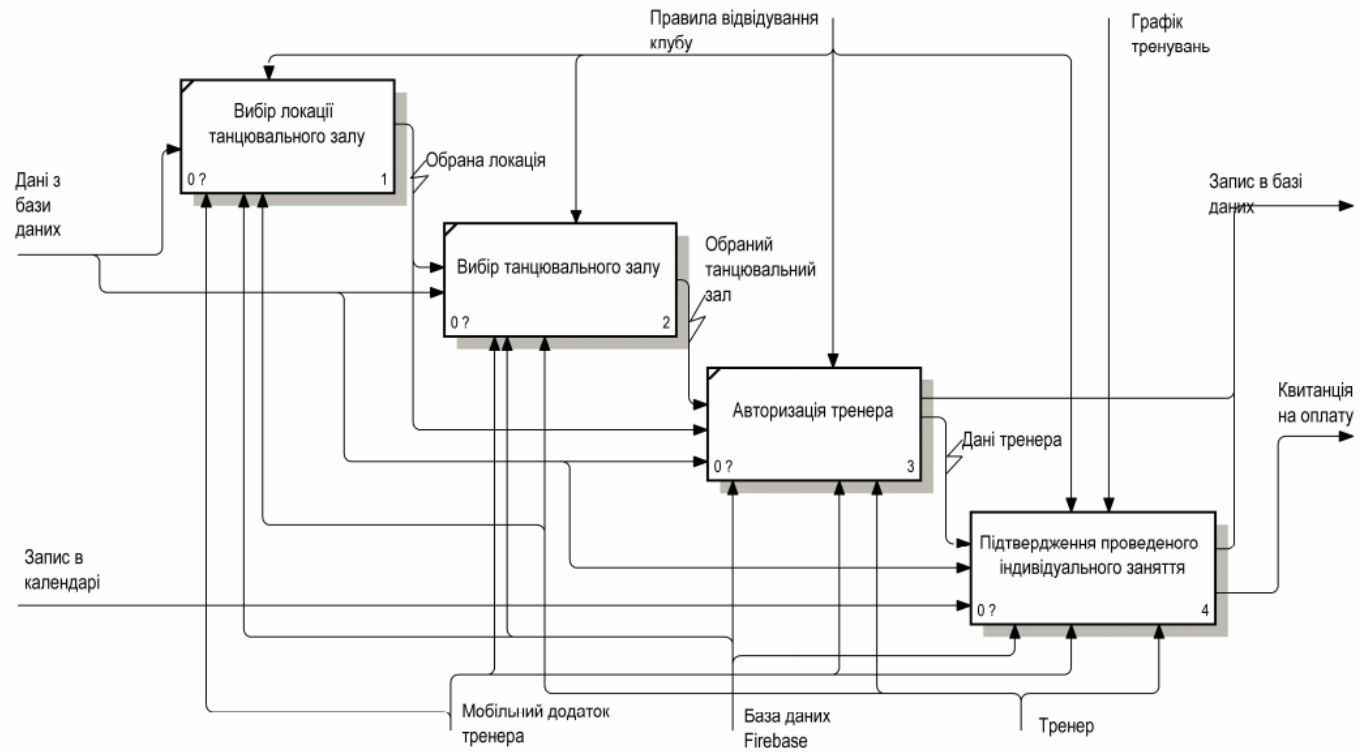


Рисунок 3.5 – Діаграма першого рівня декомпозиції

Для більш чіткого розуміння роботи додатку було побудовано діаграму другого рівня декомпозиції останнього етапу.

Даний етап має 5 блоків:

– Вибір запису з календарю;
– Додавання додаткової послуги;
– Редагування оплачуваних послуг;
– Завантаження фото;
– Підтвердження проведеного заняття.

Другий, третій та четвертий блоки знаходяться паралельно після першого.

Характеристика першого блоку «Вибір запису з календарю»:

– Вхідні дані: Дані з бази даних, дані тренера, запис в календарі;
– Вихідні дані: обраний запис в календарі;
– Елементи керування: графік тренувань;
– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика другого блоку «Додавання додаткової послуги»:

– Вхідні дані: Обраний запис в календарі, дані з бази даних;
– Вихідні дані: Додана додаткова послуга;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика третього блоку «Редагування оплачуваних послуг»:

– Вхідні дані: Обраний запис в календарі, дані з бази даних;
– Вихідні дані: Відредагована оплата;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика четвертого блоку «Завантаження фото»:

– Вхідні дані: Обраний запис в календарі;
– Вихідні дані: Завантажене фото;
– Елементи керування: Правила відвідування клубу;
– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Характеристика п’ятого блоку «Підтвердження запису на індивідуальне заняття»:

– Вхідні дані: Дані з бази даних, додана додаткова послуга, відредагована оплата, завантажене фото;
– Вихідні дані: Запис в базі даних, квитанція на оплату;
– Елементи керування: Правила відвідування клубу, графік відвідувань;
– Механізми: Мобільний додаток тренера, база даних Firebase, тренер.

Діаграму представлено на рис. 3.6.

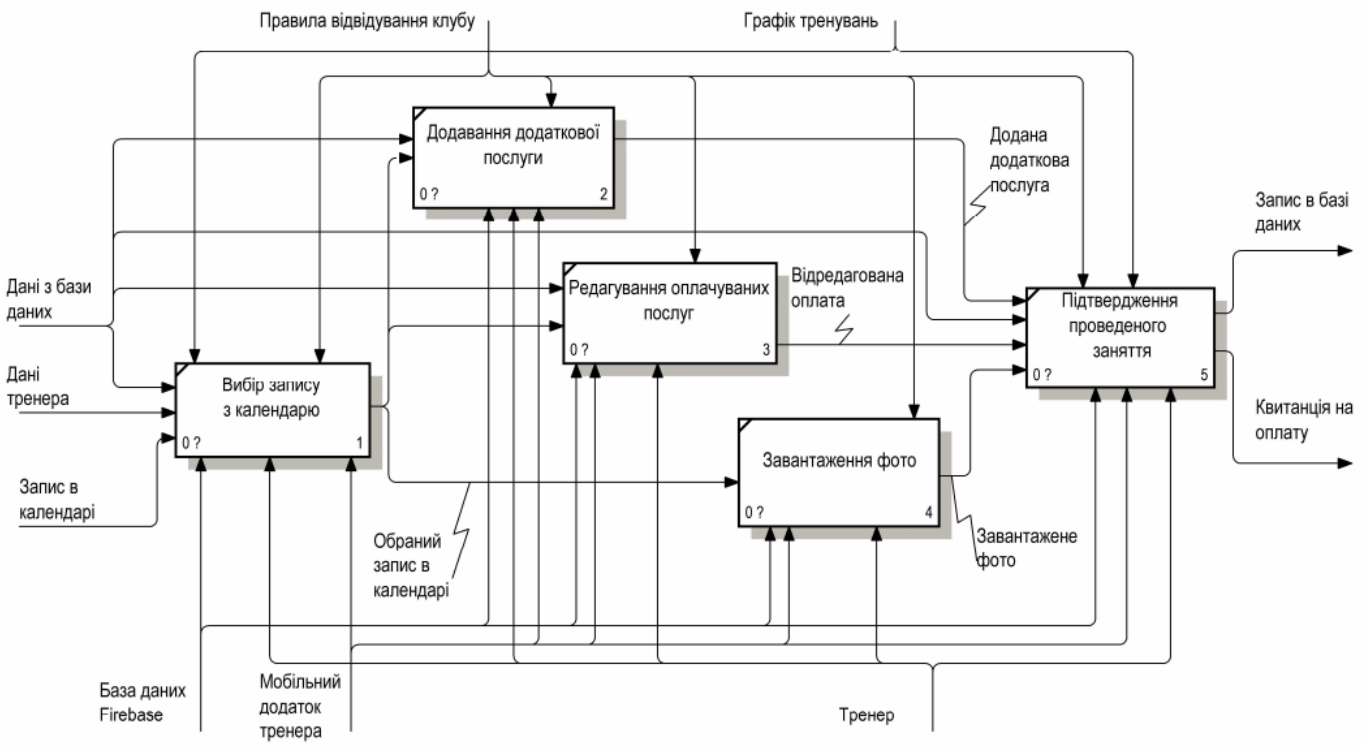


Рисунок 3.6 – Діаграма другого рівня декомпозиції

Усі етапи зв‘язані між собою, тому для того, щоб мобільні додатки мали змогу працювати без помилок, потрібно дотримуватися виконання усіх, зазначених вище, підпроцесів.

3.2 Моделювання варіантів використання мобільного додатку підтримки діяльності КСТ Силует

Діаграма варіантів використання UML - це основна форма системних / програмних вимог до нової програми. У варіантах використання вказується очікувана поведінка (що), а не точний спосіб її здійснення (як). Описані варіанти використання можна позначити як в текстовому, так і візуальному поданні (тобто діаграма варіантів

використання). Ключова концепція моделювання варіантів використання полягає в тому, що це допомагає розробити додаток з точки зору кінцевого користувача. Це ефективний прийом для комунікації поведінки системи з точки зору користувача шляхом вказівки всієї видимої зовні системи поведінки [20].

Для мобільного додатку членів Громадської організації «Клуб спортивного танцю «Силует» були виділені наступні актори:

– Танцівник - член танцювального клубу, який буде працювати з даним додатком та має доступ до усіх варіантів використання;

– Сайт зі змаганнями – сайт, з якого відбувається автоматизований процес вилучення інформації, тобто його парсинг;

– База даних (БД) – сховище даних, до якого додаються або використовуються дані під час роботи з мобільним додатком.

Варіанти використання:

– Авторизація;

– Запис на індивідуальне заняття;

– Оплата послуг танцювального клубу;

– Перегляд майбутнього тренування;

– Перегляд майбутніх змагань;

– Оцінювання тренера.

На рис. 3.7 представлена діаграма варіантів використання мобільного додатку для членів танцювального клубу.

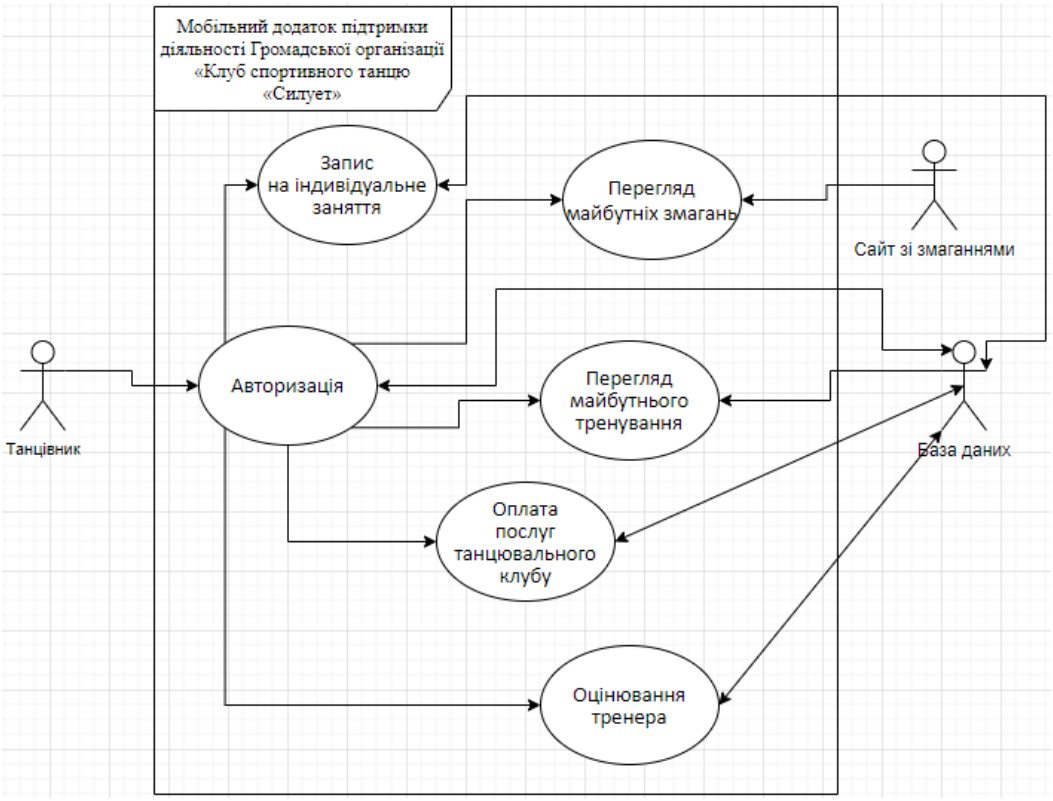


Рисунок 3.7 – Модель варіантів використання мобільного додатку членів танцювального клубу

Для мобільного додатку тренерів Громадської організації «Клуб спортивного танцю «Силует» були виділені наступні актори:

– Тренер - тренер танцювального клубу, який буде працювати з даним додатком та має доступ до усіх варіантів використання;

– Сайт зі змаганнями – сайт, з якого відбувається автоматизований процес вилучення інформації, тобто його парсинг;

– База даних (БД) – сховище даних, до якого додаються або використовуються дані під час роботи з мобільним додатком.

Варіанти використання:

– Авторизація;

– Підтвердження проведеного тренування;

– Перегляд повідомлень від клієнтів(членів клубу, що записані на тренування);

– Перегляд майбутніх змагань;

– Перегляд розкладу.

На рис. 3.8 представлена діаграма варіантів використання мобільного додатку для тренерів танцювального клубу.

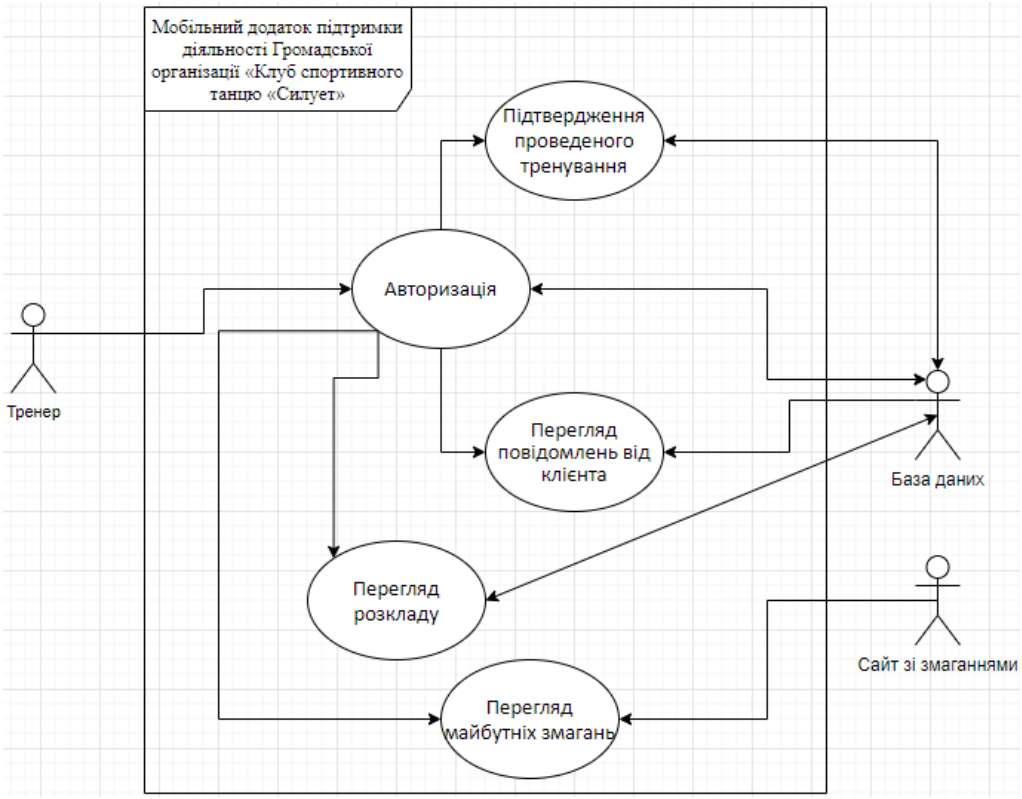


Рисунок 3.8 – Модель варіантів використання мобільного додатку тренерів танцювального клубу

3.3 Проектування бази даних

Виходячи з побудованих моделей варіантів використання мобільні додатки мають зберігати інформацію в базі даних (БД). Дуже зручною у використанні є нереляційна база даних Cloud Firestore.

Cloud Firestore – це документно-орієнтована база даних NoSQL. На відміну від бази даних SQL, в ній відстуня чітко визначена структура таблиць і зв’язків між ними. Натомість дані зберігаються в документах, які упорядковані у колекції. Кожен документ містить набір пар ключ-значення. Cloud Firestore оптимізований для зберігання великих колекцій невеликих документів. Усі документи повинні зберігатися в колекціях. Документи можуть містити підколекції та вкладені об'єкти, обидва з яких можуть включати примітивні поля, такі як рядки, або складні об'єкти, такі як списки [21].

Так як дана база Firestore працює за технологією «schemaless», це означає, що вона не має схеми для всієї бази даних [22]. Тому на рис. 3.9 представлено приклад фізичної структури збереження даних.

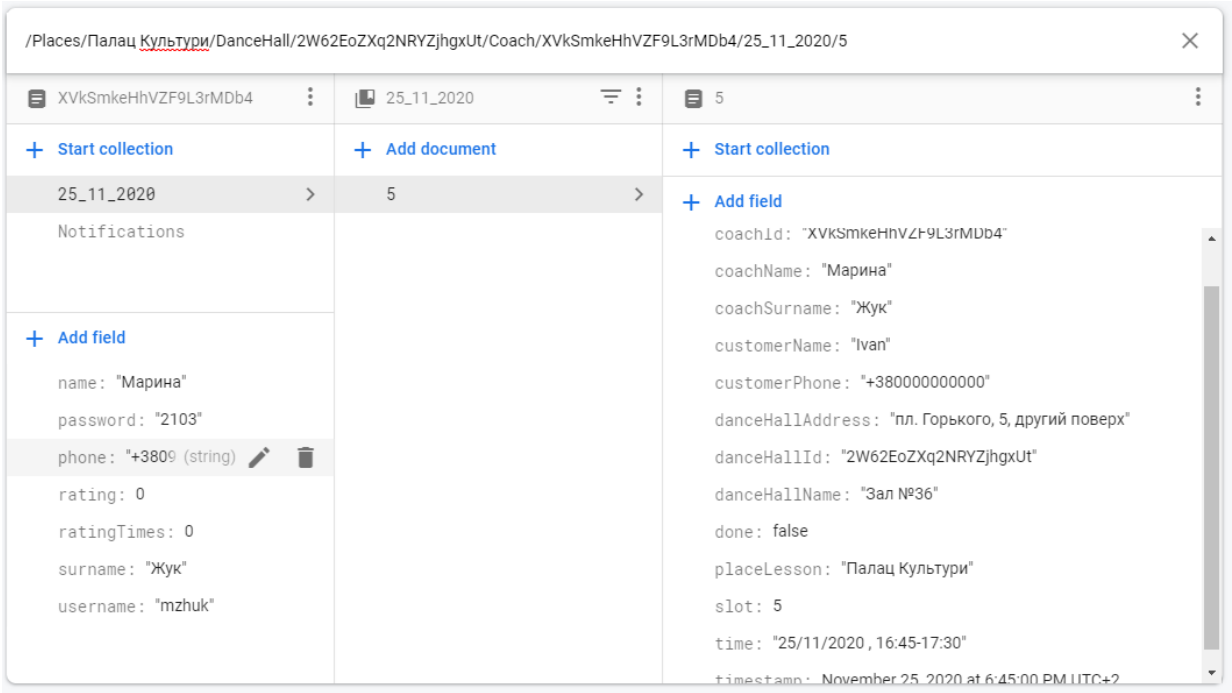


Рисунок 3.9 – Приклад фізичної структури збереження даних

Короткий опис головних колекцій розробленої бази даних представлено в табл. 3.1.

Таблиця 3.1 – Опис головних колекцій бази даних

Колекція	Опис
Banner	Колекція, де зберігаються посилання на фотографії клубу
Payment	Колекція, де зберігаються документи з інформацією про пункти оплати
Places	Колекція, де зберігаються документи з інформацією про локації, танцювальні зали, тренерів та їх записи та повідомлення про майбутні заняття
Tokens	Колекція, де зберігається інформація про поточний вхід користувача в додаток
User	Колекція, де зберігаються зареєстровані користувачі та їх записи на майбутні індивідуальні заняття
DanceHall	Колекція, де зберігається інформація про танцювальні зали
Coach	Колекція, де зберігається інформація про тренерів
Invoices	Колекція, де зберігається інформація про завершені заняття
Services	Колекція, де зберігається інформація про додаткові сервіси (послуги), які може надавати тренер
Notifications	Колекція, де зберігаються повідомлення
Lesson	Колекція, де зберігається інформація про заняття танцівника
Items	Колекція, де зберігається інформація про елементи оплати

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПІДТРИМКИ ДІЯЛЬНОСТІ ГРОМАДСЬКОЇ ОРГАНІЗАЦІЇ «КЛУБ СПОРТИВНОГО ТАНЦЮ «СИЛУЕТ»

4.1 Установка та запуск компонентів мобільного додатку

Однією з важливих функцій розроблюваного мобільного додатку є зберігання та синхронізація даних між користувачами та пристроями за допомогою розміщеної у хмарі бази даних (БД) NoSQL. Cloud Firestore надає синхронізацію та підтримку в режимі офлайн, а також ефективні запити даних [23]. Тому, перш ніж почати розробку МД необхідно створити проект Firebase та потім підключити його до проекту в Android Studio.

Додавання Firebase до програми включає завдання як на консолі Firebase, так і у відкритому проекті Android [24]. Для цього потрібно виконати наступні кроки:

– Крок 1. Створення проекту Firebase.

Перш ніж додати Firebase до своєї програми для Android, необхідно створити проект Firebase для підключення до програми Android. На консолі Firebase треба натиснути «Add project», а потім ввести назву проекту (рис. 4.1).

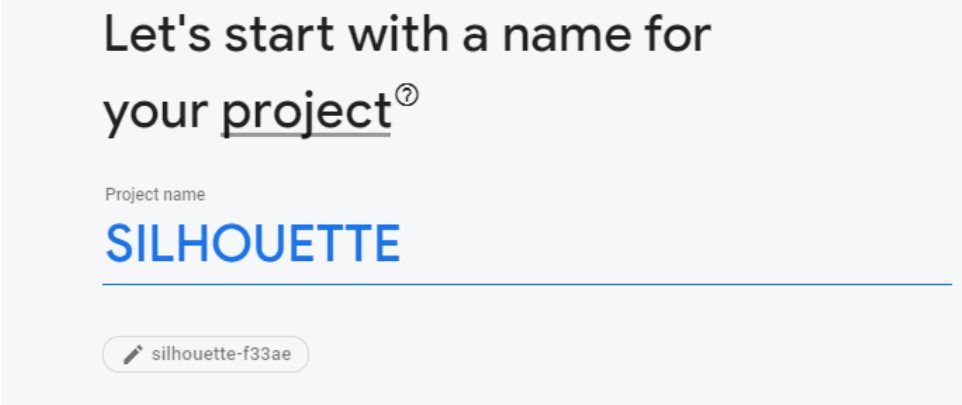


Рисунок 4.1 – Введення назви проекту Firebase

Далі потрібно натиснути «Create project». Firebase автоматично надає ресурси для проекту Firebase(рис. 4.2).

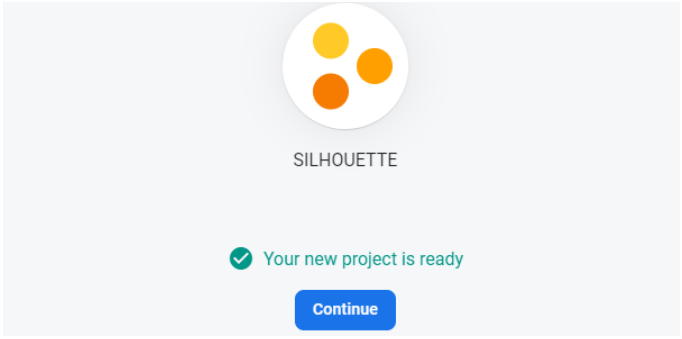


Рисунок 4.2 – Завершення створення Firebase

– Крок 2. Реєстрація додатку Android у Firebase за допомогою Firebase Assistant [25]

Firebase Assistant реєструє додаток до проекту Firebase і додає необхідні Firebase файли, плагіни і залежності для Android проекту.

Для цього потрібно відкрити свій проект Android у Android Studio та отримати доступ до Firebase Assistant, а саме перейти до Tools> Firebase, щоб відкрити панель Асистента (рис. 4.3).

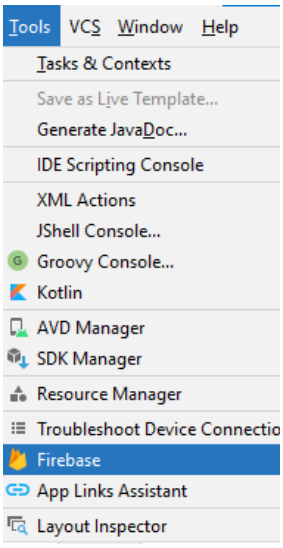


Рисунок 4.3 – Перехід до Tools> Firebase

Далі треба обрати продукти Firebase, які потрібно додати до програми. Для розроблюваного мобільного додатку це: Authentication (рис. 4.4) [26], Firestore (рис. 4.5) [23], Cloud Messaging (рис. 4.6) [27].

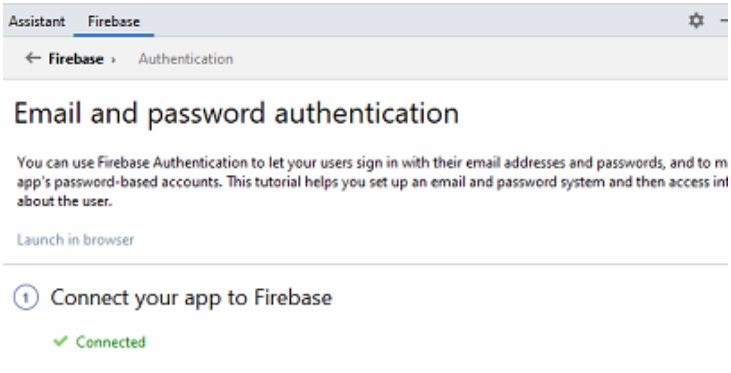


Рисунок 4.4 – Підключення Authentication

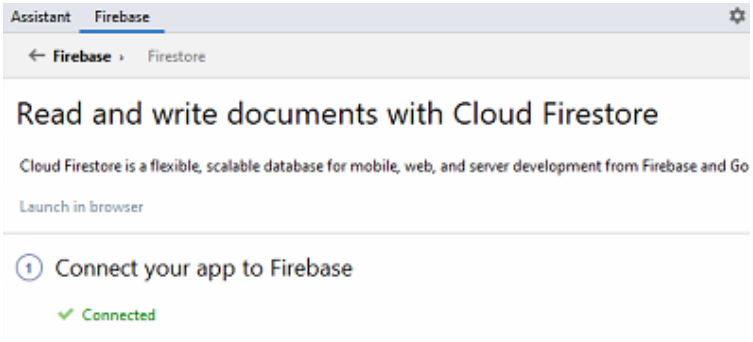


Рисунок 4.5 – Підключення Firestore

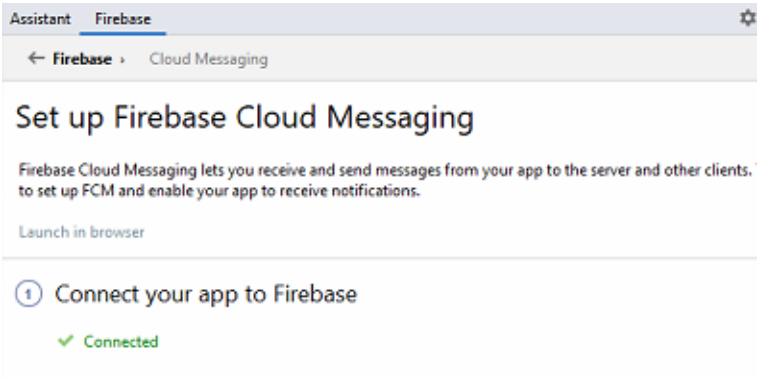


Рисунок 4.6 – Підключення Cloud Messaging

Далі необхідно синхронізувати додаток, щоб переконатися, що всі залежності мають необхідні версії [28].

До системи збірки Gradle в Android Studio було додано нові модулі бібліотеки до збірки як залежності. (рис. 4.7-4.8)

```
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.2.1'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation 'com.google.firebase:firebase-firestore:22.0.0'
implementation 'com.google.firebase:firebase-messaging:21.0.0'
implementation 'com.firebaseui:firebase-ui-auth:6.2.0'
implementation 'androidx.legacy:legacy-support-v4:1.0.0'
implementation 'com.google.firebase:firebase-auth:19.3.1'
testImplementation 'junit:junit:4.13.1'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

// Add libraries
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.3.0-alpha03'
implementation 'com.google.android.gms:play-services-auth:19.0.0'
implementation 'com.jakewharton:butterknife:10.2.1'
annotationProcessor 'com.jakewharton:butterknife-compiler:10.2.1'
implementation 'com.github.d-max:spots-dialog:1.1@aar'
implementation 'com.ss.bannerslider:bannerslider:2.0.0'
implementation 'com.squareup.picasso:picasso:2.71828'
implementation 'com.shuhart.stepview:stepview:1.5.1'
implementation 'com.jaredrummler:material-spinner:1.3.1'
implementation 'devs.mulham.horizontalcalendar:horizontalcalendar:1.3.4'
implementation 'com.karumi:dexter:6.2.1'
implementation 'io.paperdb:paperdb:2.7.1'

implementation 'androidx.room:room-runtime:2.2.5'
annotationProcessor 'androidx.room:room-compiler:2.2.5'
implementation 'com.nex3z:notification-badge:1.0.2'

implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
implementation 'io.reactivex.rxjava2:rxjava:2.2.9'
implementation 'androidx.room:room-rxjava2:2.2.5'

implementation 'org.greenrobot:eventbus:3.1.1'
```

Рисунок 4.7 – Gradle app dependencies додатку танцівника

```
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.2.1'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation 'com.google.firebase:firebase-firestore:22.0.0'
implementation 'com.google.firebase:firebase-messaging:21.0.0'
implementation 'com.google.firebase:firebase-storage:19.2.0'
testImplementation 'junit:junit:4.13.1'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

implementation 'com.google.android.material:material:1.3.0-alpha03'
implementation 'com.jakewharton:butterknife:10.2.1'
annotationProcessor 'com.jakewharton:butterknife-compiler:10.2.1'
implementation 'com.github.d-max:spots-dialog:1.1@aar'
implementation 'devs.mulham.horizontalcalendar:horizontalcalendar:1.3.4'
implementation 'io.paperdb:paperdb:2.7.1'
implementation 'com.google.code.gson:gson:2.8.6'

implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
implementation 'io.reactivex.rxjava2:rxjava:2.2.9'

implementation 'com.squareup.picasso:picasso:2.71828'
implementation 'com.karumi:dexter:6.2.1'

implementation 'org.greenrobot:eventbus:3.1.1'

implementation 'org.jsoup:jsoup:1.13.1'
implementation 'com.github.barteksc:android-pdf-viewer:2.8.2'
```

Рисунок 4.8 – Gradle app dependencies додатку тренера

Після цього можна починати роботу з БД. Приклади виконання основних запитів зображено на рис. 4.9-4.11 [29-31]

```
private void loadPaymentItem(String itemMenu) {
    paymentItemRef = FirebaseFirestore.getInstance() FirebaseFirestore
            .collection( collectionPath: "Payment") CollectionReference
            .document(itemMenu) DocumentReference
            .collection( collectionPath: "Items");

    // Get Data
    paymentItemRef.get()
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    iPaymentDataLoadListener.onPaymentDataLoadFailed(e.getMessage());
                }
            }).addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if(task.isSuccessful()) {
                List<PaymentItem> paymentItems = new ArrayList<>();
                for(DocumentSnapshot itemSnapShot:task.getResult()) {
                    PaymentItem paymentItem = itemSnapShot.toObject(PaymentItem.class);
                    paymentItem.setId(itemSnapShot.getId());
                    paymentItems.add(paymentItem);
                }
                iPaymentDataLoadListener.onPaymentDataLoadSuccess(paymentItems);
            }

        }
    });

}
```

Рисунок 4.9 – Приклад витягування даних з БД

```
    // Submit to Coachs document
    DocumentReference lessonDate = FirebaseFirestore.getInstance() FirebaseFirestore
            .collection( collectionPath: "Places") CollectionReference
            .document(Common.place) DocumentReference
            .collection( collectionPath: "DanceHall") CollectionReference
            .document(Common.currentDanceHall.getDanceHallId()) DocumentReference
            .collection( collectionPath: "Coach") CollectionReference
            .document(Common.currentCoach.getCoachId()) DocumentReference
            .collection(Common.simpleDateFormat.format(Common.lessonDate.getTime())) CollectionReference
            .document(String.valueOf(Common.currentTimeSlot));

    // Write data
    lessonDate.set(lessonInformation)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    // If lesson exists - don't create new

                    // Clear cart
                    cartDataSource.clearCart(Common.currentUser.getPhone())
                            .subscribeOn(Schedulers.io())
                            .observeOn(AndroidSchedulers.mainThread())
                            .subscribe(new SingleObserver<Integer>() {
                                @Override
                                public void onSubscribe(Disposable d) { }

                                @Override
                                public void onSuccess(Integer integer) {
                                    addToUserLesson(lessonInformation);
                                }

                                @Override
                                public void onError(Throwable e) {
                                    Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                                }
                            });
                }
            }).addOnFailureListener(e -> Toast.makeText(getContext(), text: "" + e.getMessage(), Toast.LENGTH_SHORT).show());


}, throwable -> Toast.makeText(getContext(), text: ""+throwable.getMessage(), Toast.LENGTH_SHORT).show()));
```

Рисунок 4.10 – Приклад запису даних до БД

```
private void sendNotificationUpdateToUser(String customerEmail) {
    // Get user Token
    FirebaseFirestore.getInstance() FirebaseFirestore
            .collection( collectionPath: "Tokens") CollectionReference
            .whereEqualTo( field: "user", customerEmail) Query
            .get() Task<QuerySnapshot>
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if(task.isSuccessful() && task.getResult().size() > 0) {

                        MyToken myToken = new MyToken();
                        for(DocumentSnapshot tokenSnapshot:task.getResult()) {
                            myToken = tokenSnapshot.toObject(MyToken.class);

                            // Create notification to send
                            FCMSendData fcmSendData = new FCMSendData();
                            Map<String, String> dataSend = new HashMap<>();
                            dataSend.put("update_done", "true");

                            // Information for Rating
                            dataSend.put(Common.RATING_PLACE_KEY, Common.place_name);
                            dataSend.put(Common.RATING_DANCEHALL_ID, Common.selectedDanceHall.getDanceHallId());
                            dataSend.put(Common.RATING_DANCEHALL_NAME, Common.selectedDanceHall.getName());
                            dataSend.put(Common.RATING_COACH_ID, Common.currentCoach.getCoachId());

                            fcmSendData.setTo(myToken.getToken());
                            fcmSendData.setData(dataSend);

                            ifcmService.sendNotification(fcmSendData)
                                    .subscribeOn(Schedulers.io())
                                    .observeOn(Schedulers.newThread())
                                    .subscribe(new Consumer<FCMResponse>() {
                                        @Override
                                        public void accept(FCMResponse fcmResponse) throws Exception {
                                            dialog.dismiss();
                                            dismiss();
                                            EventBus.getDefault()
                                                    .postSticky(new DismissFromBottomSheetEvent( isButtonClick: true));
                                        }
                                    }, throwable -> Toast.makeText(getContext(), throwable.getMessage(), Toast.LENGTH_SHORT).show());
                        }
                    }
                }
            });
```

Рисунок 4.11 – Приклад відправлення повідомлення на додаток танцівника

4.2 Результат реалізації мобільного додатку

Після встановлення мобільного додатку на мобільний телефон користувача в меню програм з‘являється ярлик (рис 4.10-4.11).

Рисунок 4.9 – Ярлик додатку танцівника в меню програм (SilhouetteClientApp)

Рисунок 4.10 – Ярлик додатку тренера в меню програм (SilhouetteStaffApp)

4.2.1 Порядок роботи з додатком танцівника

По натисканню на іконку, МД завантажується. При першому запуску з’являється заставка та пропонується дозволити доступ до календаря (рис.4.11)

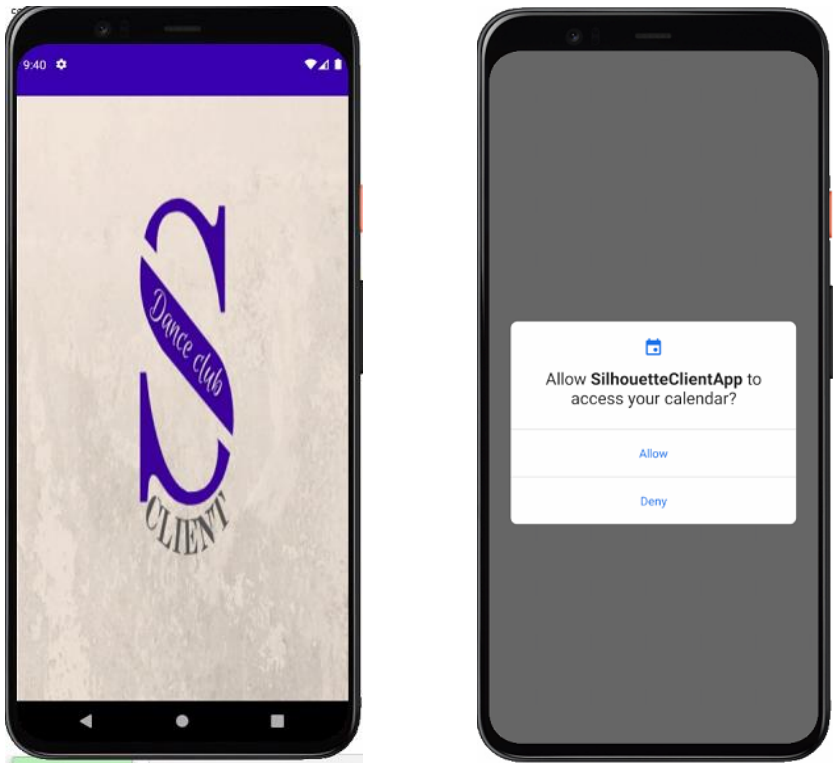


Рисунок 4.11 – Заставка та дозвіл на доступ до календаря (SilhouetteClientApp)

Далі з’являється логотип додатку з кнопкою «Увійти», по натисканню на яку користувач проходить автентифікацію за номером телефону (рис. 4.12).

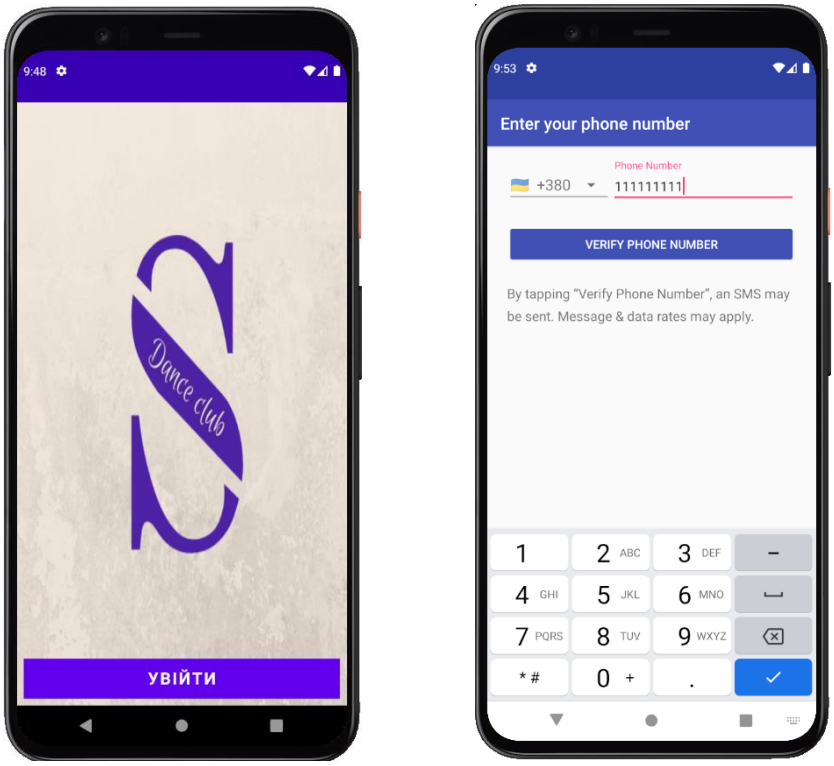


Рисунок 4.12 – Логотип з кнопкою та вікно автентифікації (SilhouetteClientApp)

Після вводу номеру, на телефон приходить повідомлення з кодом для перевірки номеру, після верифікації танцівник вводить свої дані, які вказані у показаному вікні (рис.4.13).

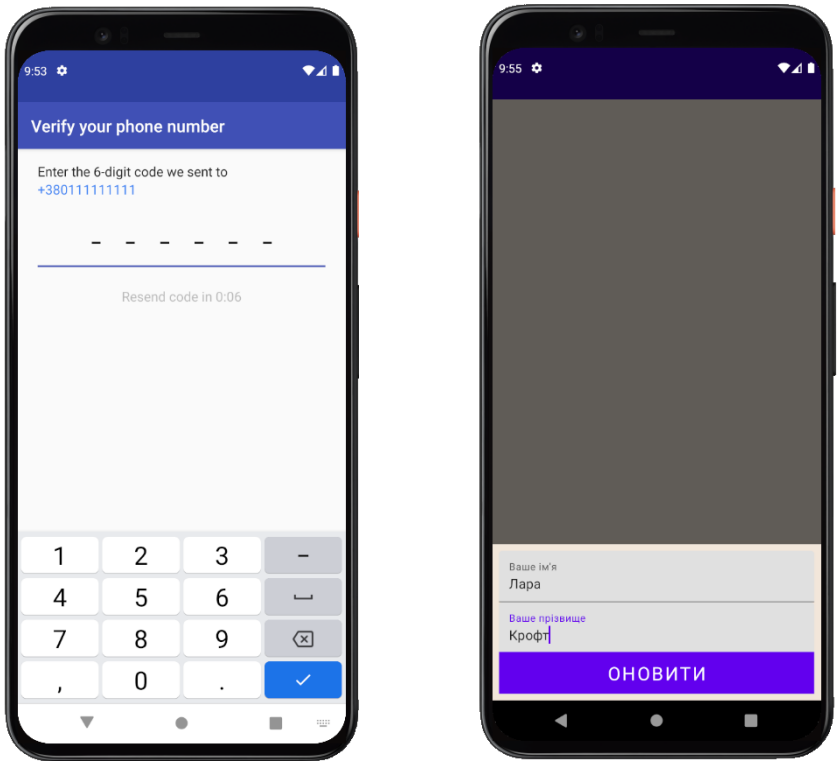


Рисунок 4.12 – Верифікація номеру через код з повідомлення та введення даних танцівника (SilhouetteClientApp)

Далі, по натичканню на кнопку «Оновити» з’являється сторінка мобільного додатку «Головна», де можна побачити дані зареєстрованого танцівника, основні кнопки та альбом з фотографіями членів та тренерів Громадської організації «Клуб спортивного танцю «Силует». Для того, щоб вийти з аккаунту танцівника потрібно натиснути на іконку аккаунту та з’явиться діалогове вікно з підтвердженням, чи дійсно користувач бажає вийти з аккаунту (рис. 4.13).

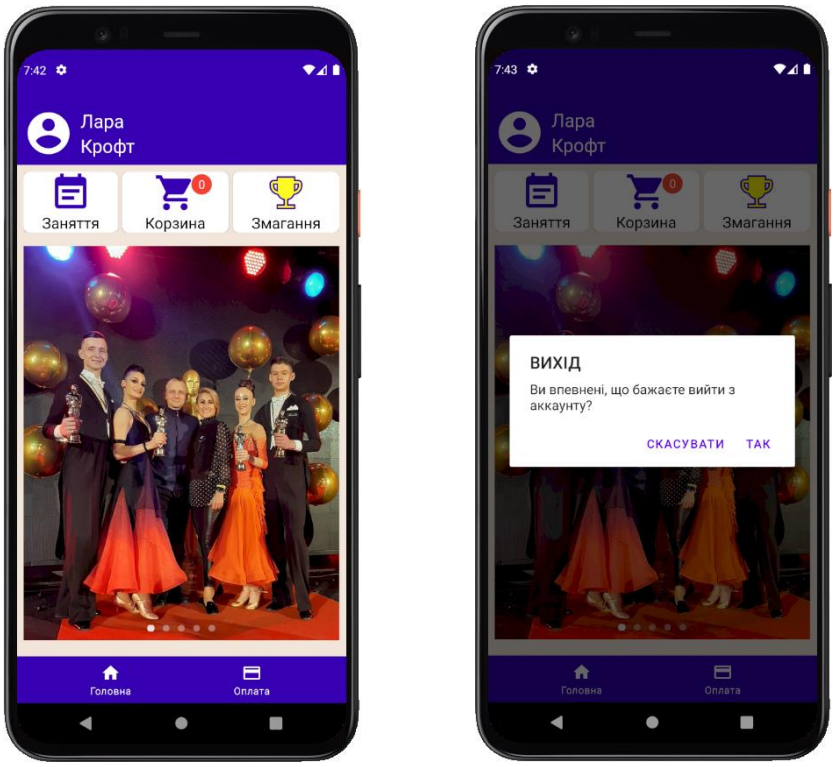


Рисунок 4.13 – Головна сторінка додатку та діалогове вікно з підтвердженням виходу з аккаунту (SilhouetteClientApp)

Якщо танцівник бажає сплатити за місяць тренувань у клубі, можна перейти до другої сторінки з назвою «Оплата» та додати місяць, який потрібно оплатити, в корзину (рис. 4.14). При натисканні на головній сторінці на «Корзина» можна переглянути додані сплати за місяць (рис. 4.15), також корзину за необхідністю можна очистити (рис. 4.16)

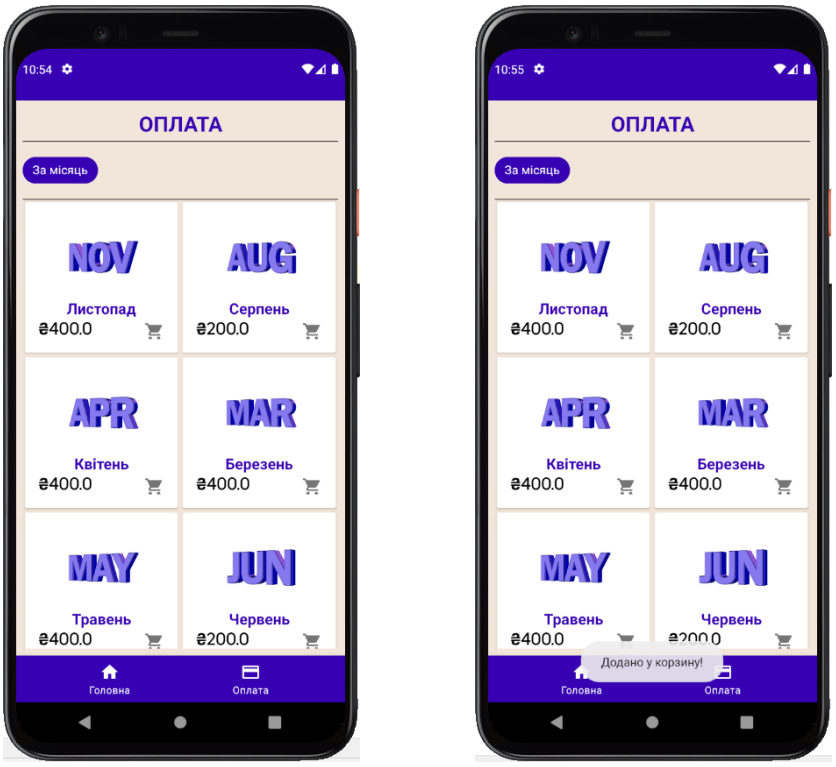


Рисунок 4.14 – Сторінка «Оплата» та додавання потрібного місяця у корзину (SilhouetteClientApp)

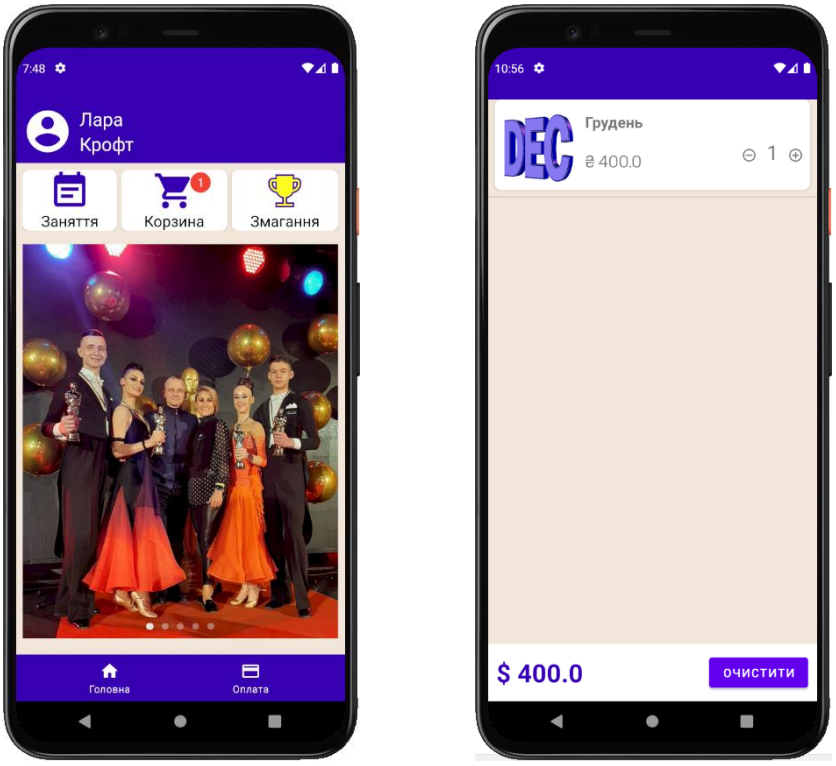


Рисунок 4.15 – Перегляд «Корзини» (SilhouetteClientApp)

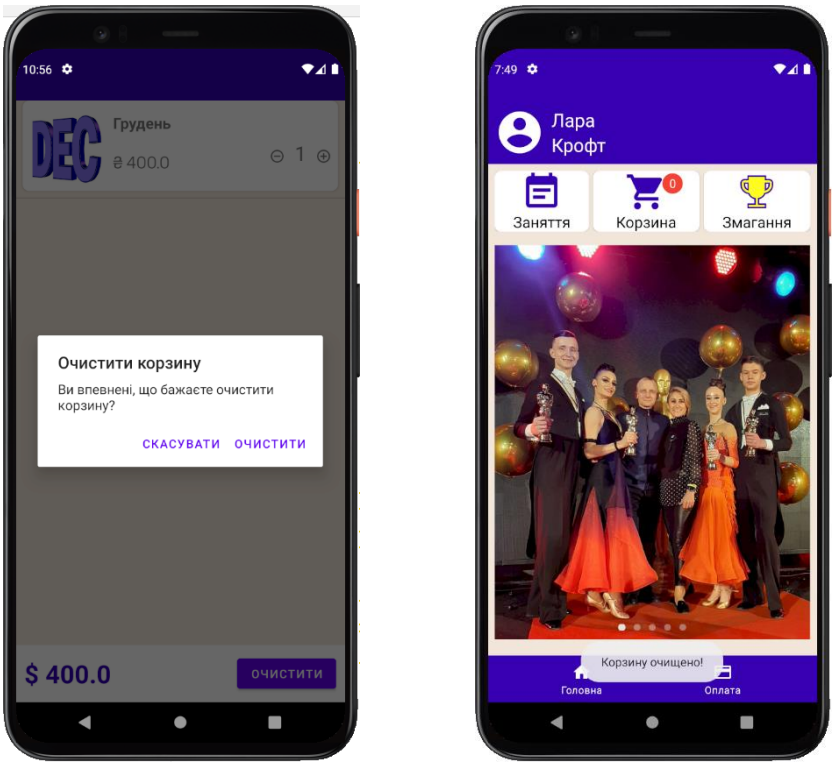


Рисунок 4.16 – Очищення «Корзини» (SilhouetteClientApp)

По натисканню на елемент керування «Заняття» головної сторінки з’являється вікно з покроковим записом на індивідуальне заняття до бажаного тренера (рис. 4.16-4.19).

Якщо в корзині були додані елементи для сплати, вони автоматично очищуються та відсилаються на додаток тренера після запису на заняття.

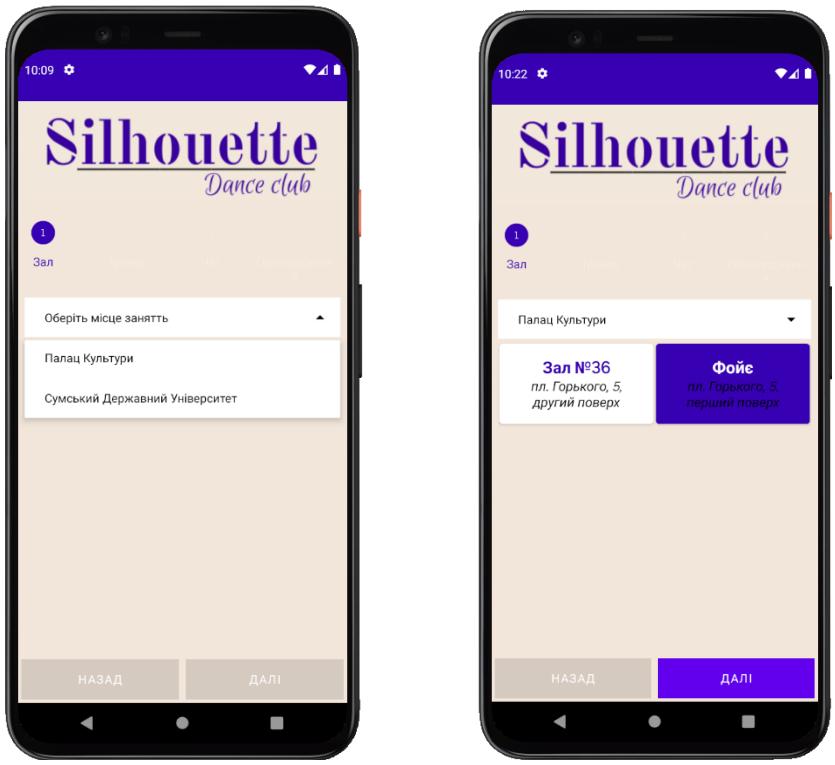


Рисунок 4.16 – Перший крок: вибір місця та зали для заняття (SilhouetteClientApp)

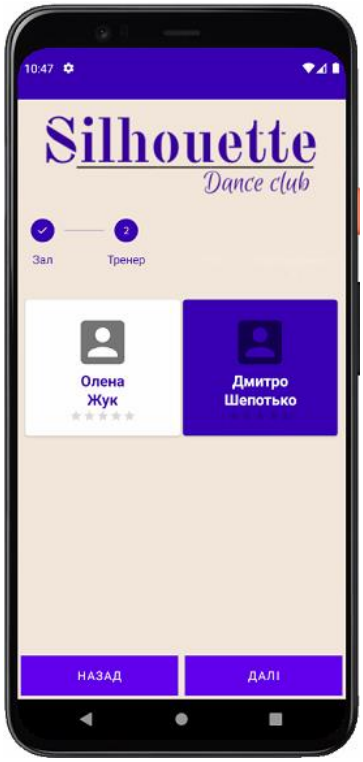


Рисунок 4.17 – Другий крок: вибір тренера в обраній залі (SilhouetteClientApp)

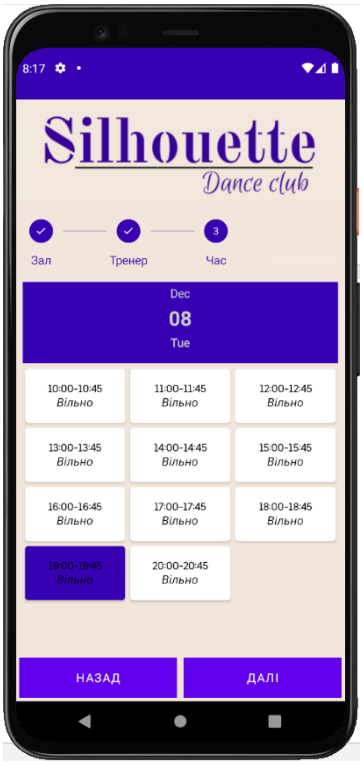


Рисунок 4.18 – Третій крок: вибір часу тренування з тренером (SilhouetteClientApp)

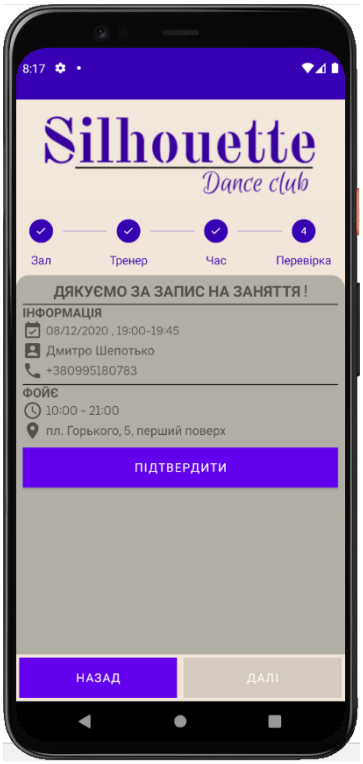


Рисунок 4.19 – Четвертий крок: підтвердження запису (SilhouetteClientApp)

Коли танцівник успішно записався на індивідуальне заняття на головній сторінці додатку з’являється віконце з інформацією про запис, заняття можна змінити або видалити. Також запис додається в календар танцівника (рис. 4.20)

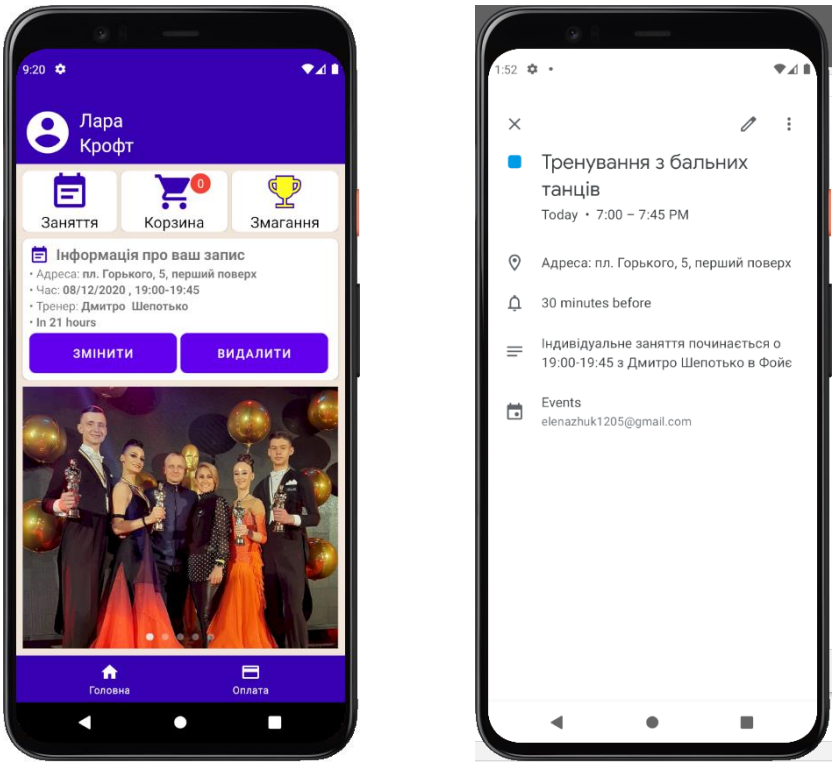


Рисунок 4.20 – Перегляд інформаціїї про тренування та запис в календарі танцівника (SilhouetteClientApp)

Після натискання на елемент керування «Змагання» на головній сторінці з’являється вікно з майбутніми змаганнями. Для того, щоб переглянути більш детальну інформацію про бажане змагання, потрібно натиснути на нього (рис. 4.21).

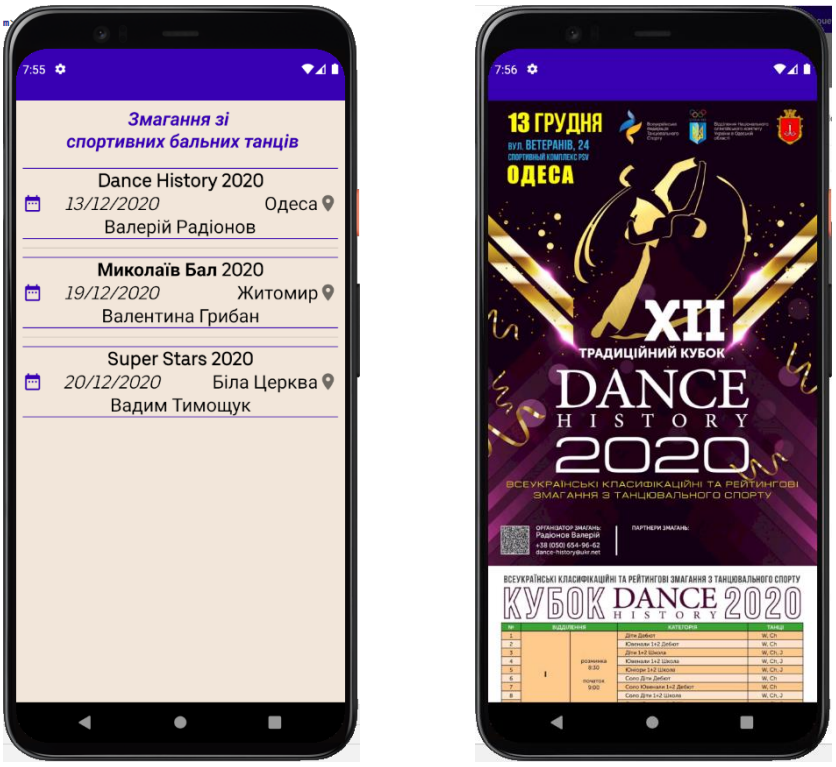


Рисунок 4.21 – Перегляд майбутніх змагань та детальної інформації (SilhouetteClientApp)

Після того, як тренер завершує та підтверджує проведене тренування з танцівником, в додатку танцівника з’являється діалогове вікно з сумою для сплати та можливістю оцінити тренера, з яким було індивідуальне заняття (рис. 4.22).

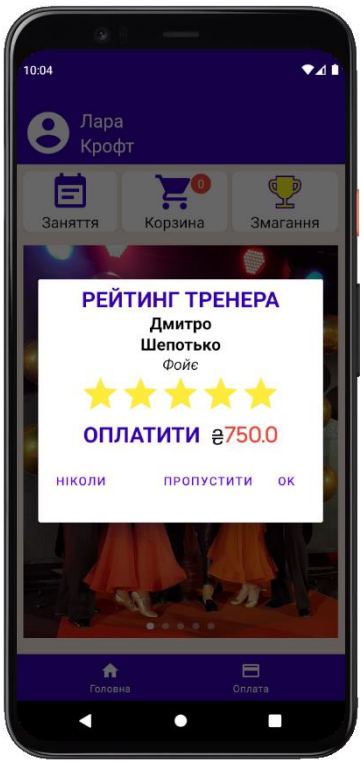


Рисунок 4.22 – Діалогове вікно з сумою для сплати на оцінкою тренера (SilhouetteClientApp)

4.2.2 Порядок роботи з додатком тренера

При першому запуску з’являється заставка цього додатку (рис. 4.23) та пропонується дозволити доступ до камери та до фотографій та медіа на пристрої (рис. 4.24)

Рисунок 4.23 – Заставка додатку тренера (SilhouetteClientApp)

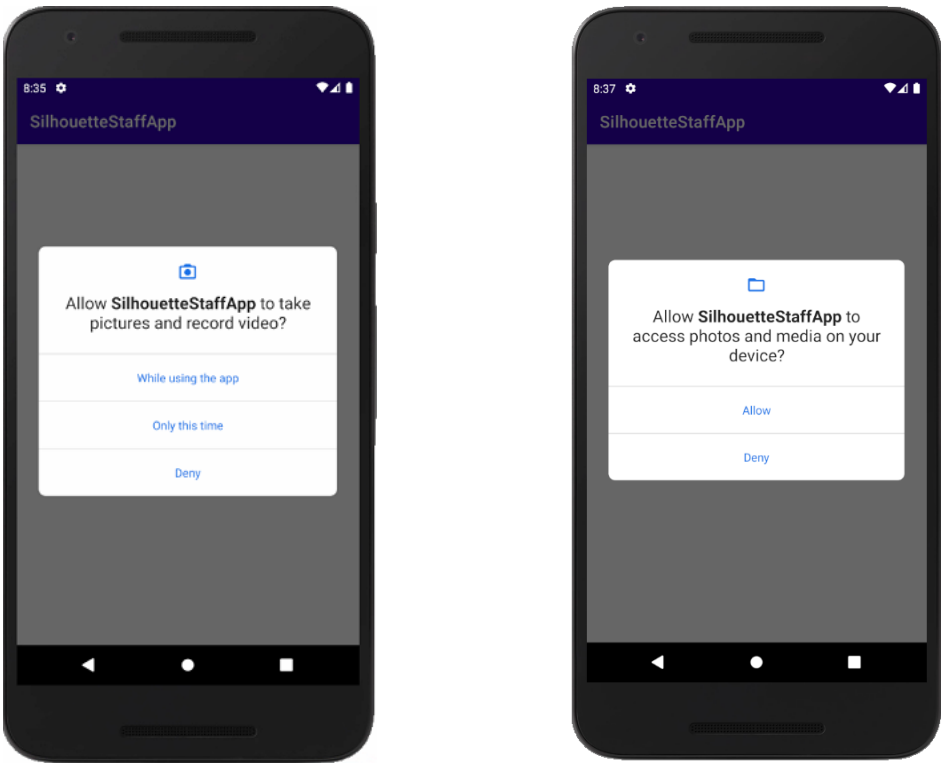


Рисунок 4.24 – Дозвіл на доступ до камери та до фотографій та медіа (SilhouetteStaffApp)

Далі з’являється вікно з вибором місця роботи тренера. По натисканню на локацію показуються усі танцювальні зали, які є (рис. 4.25). Потім тренер обирає залу, де він закріплен та вводить свій логін та пароль для входу в додаток, при вірному вводу показується головне вікно додатку, при неправильному з’являється повідомлення (рис. 4.26).

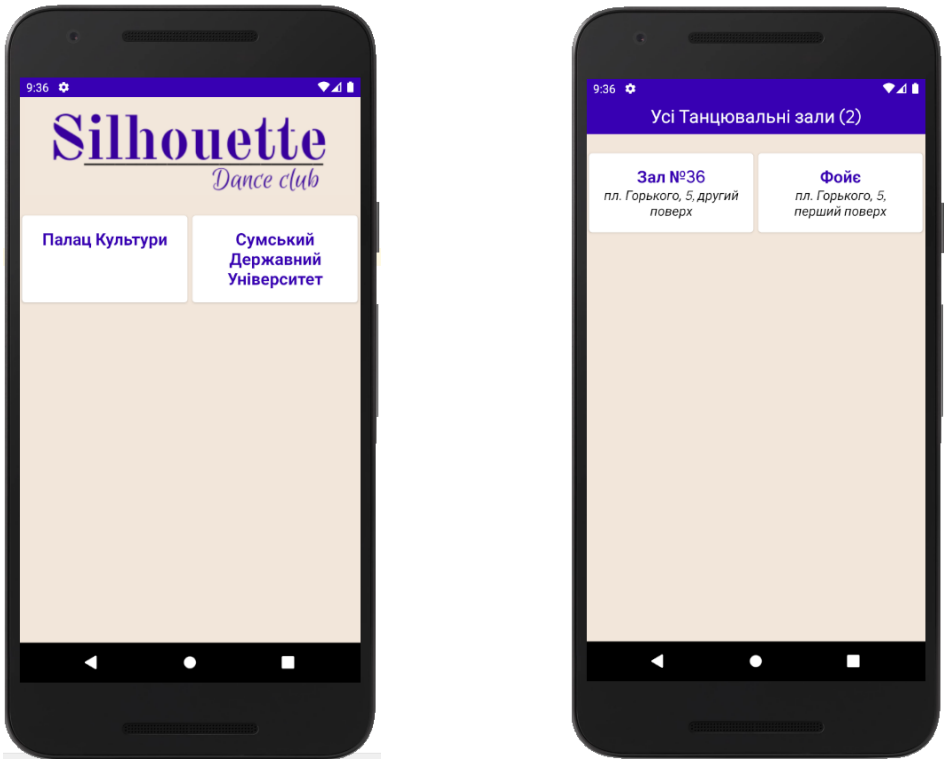


Рисунок 4.25 – Вікно з вибором місця роботи та зали тренера (SilhouetteStaffApp)

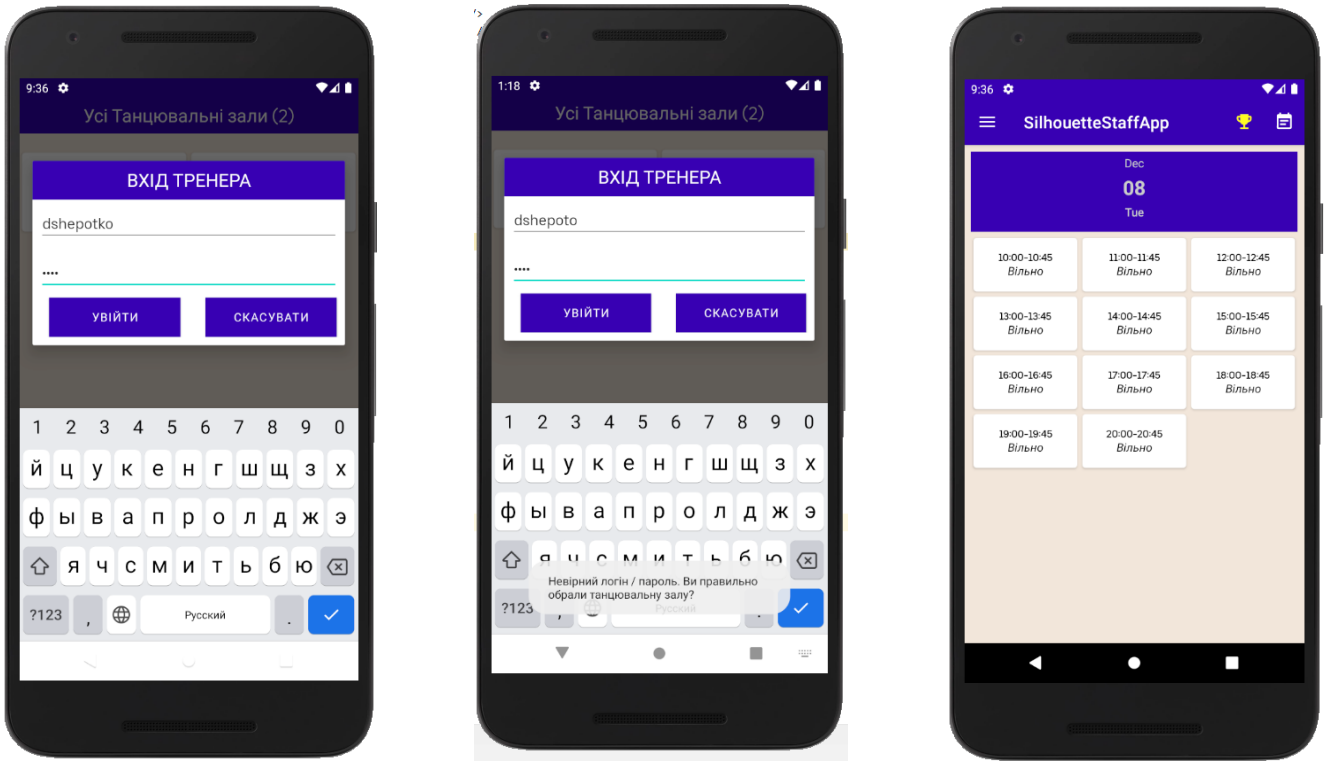


Рисунок 4.26 – Вікно вводу логіну та паролю тренера та головне вікно додатку (SilhouetteStaffApp)

Для того, щоб вийти з аккаунту тренера потрібно в боковому меню навігації натиснути на елемент «Вийти» та з’явиться діалогове вікно з підтвердженням, чи дійсно користувач бажає вийти з аккаунту (рис. 4.27)

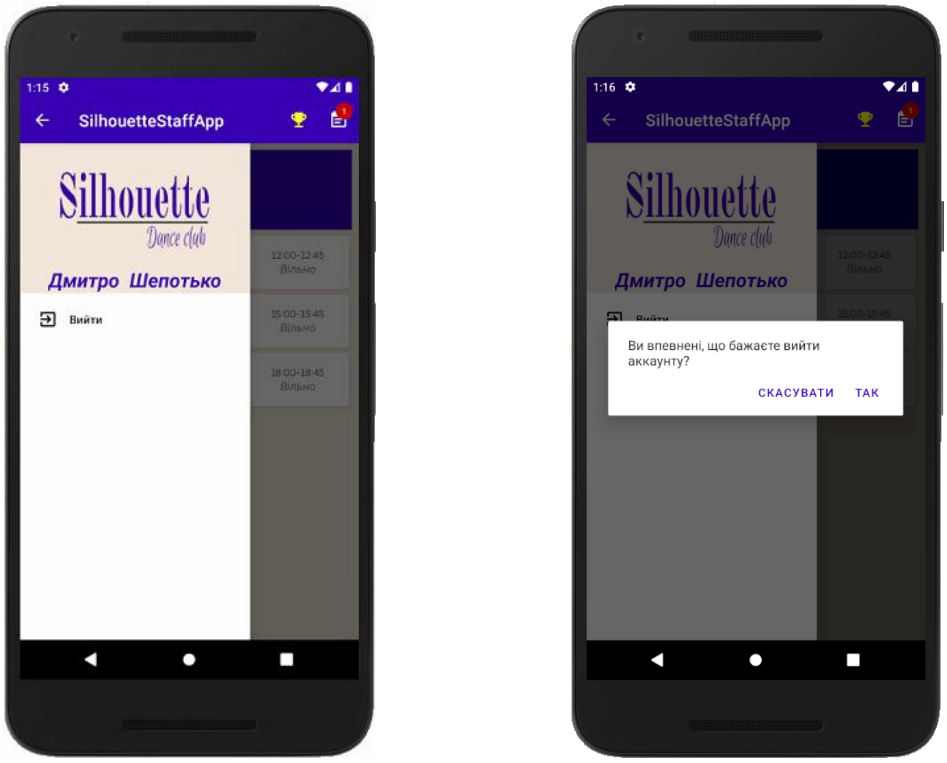


Рисунок 4.27 – Бокове меню навігації та діалогове вікно з підтвердженням виходу з аккаунту (SilhouetteStaffApp)

Коли танцівник записався на індивідуальне заняття, на телефон тренера та на додаток приходить повідомлення про новий запис, з інформацією у кого заняття на о котрій годині. Натиснувши на іконку повідомлення у верхньому правому куті головної сторінки, можна подивитися усі отримані повідомлення (рис.4.28-4.29).

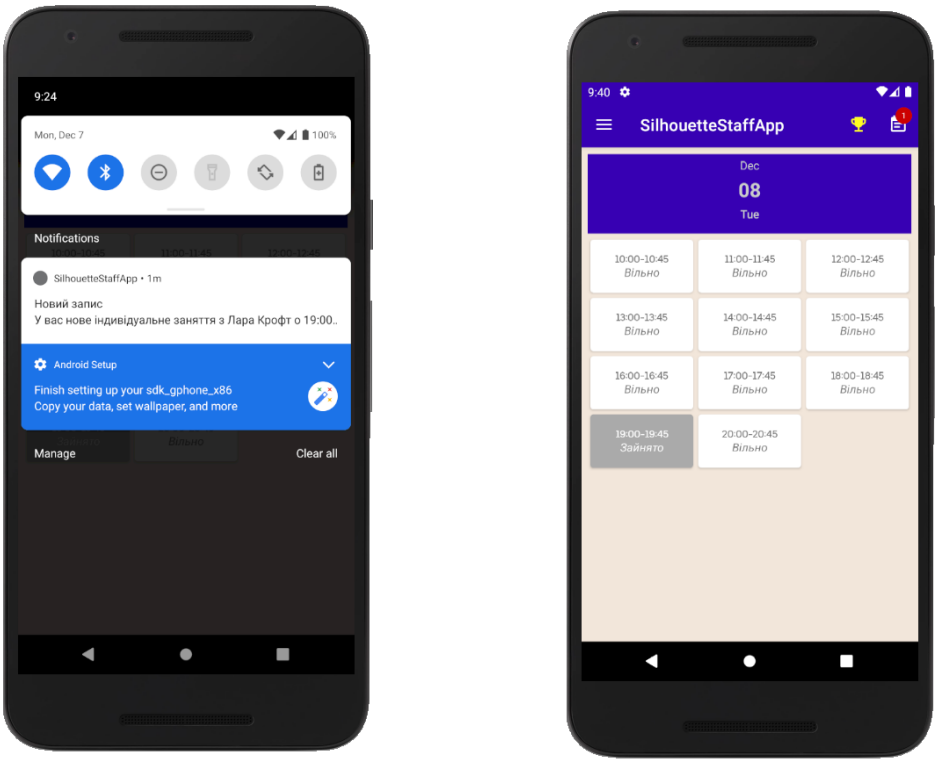


Рисунок 4.28 – Повідомлення про новий запис (SilhouetteStaffApp)

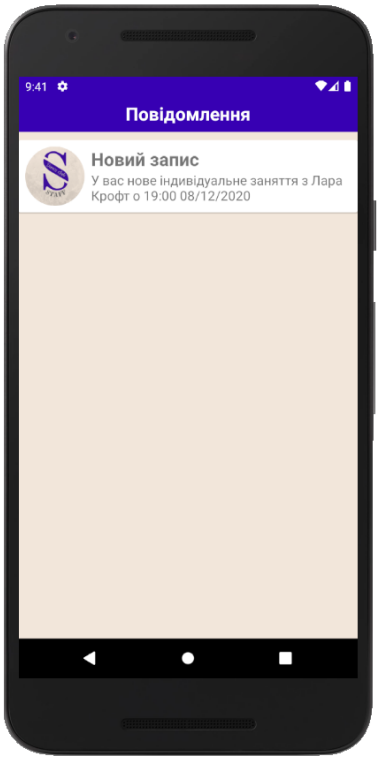


Рисунок 4.29 – Перегляд повідомлення в додатку (SilhouetteStaffApp)

Для того, щоб завершити на підтвердити проведене заняття необхідно натиснути на час заняття та з’явиться вікно підтвердження (4.30). На даній сторінці можна додати сервіси(послуги), які тренер надав та оплату танцівника за місяць (4.31). Також, за згодою танцівника у правилах клубу, є можливість спостерігати за прогресом, сфотографувавши танцівника на занятті (рис. 4.32).

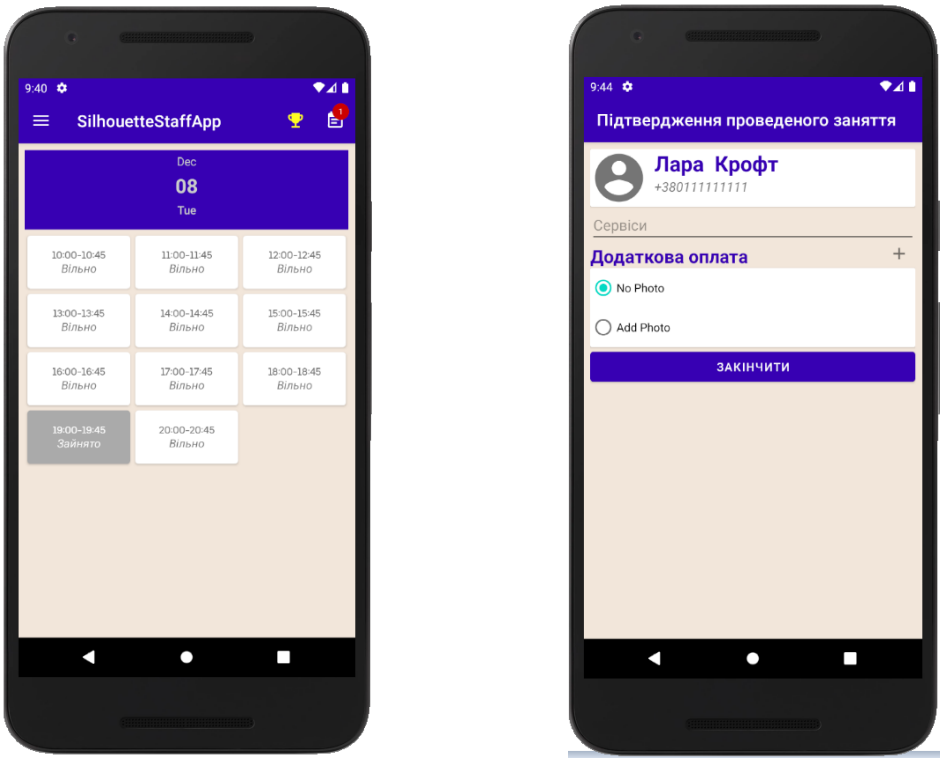


Рисунок 4.30 – Перехід до підтвердження проведеного заняття (SilhouetteStaffApp)

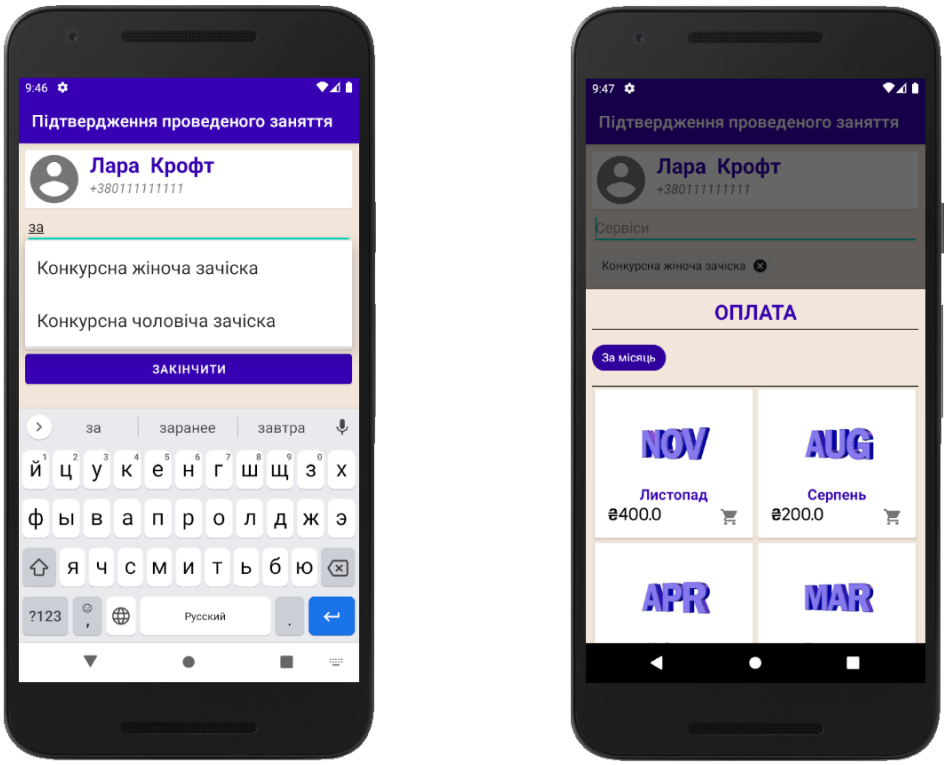


Рисунок 4.31 – Додавання додаткових сервісів та оплати (SilhouetteStaffApp)

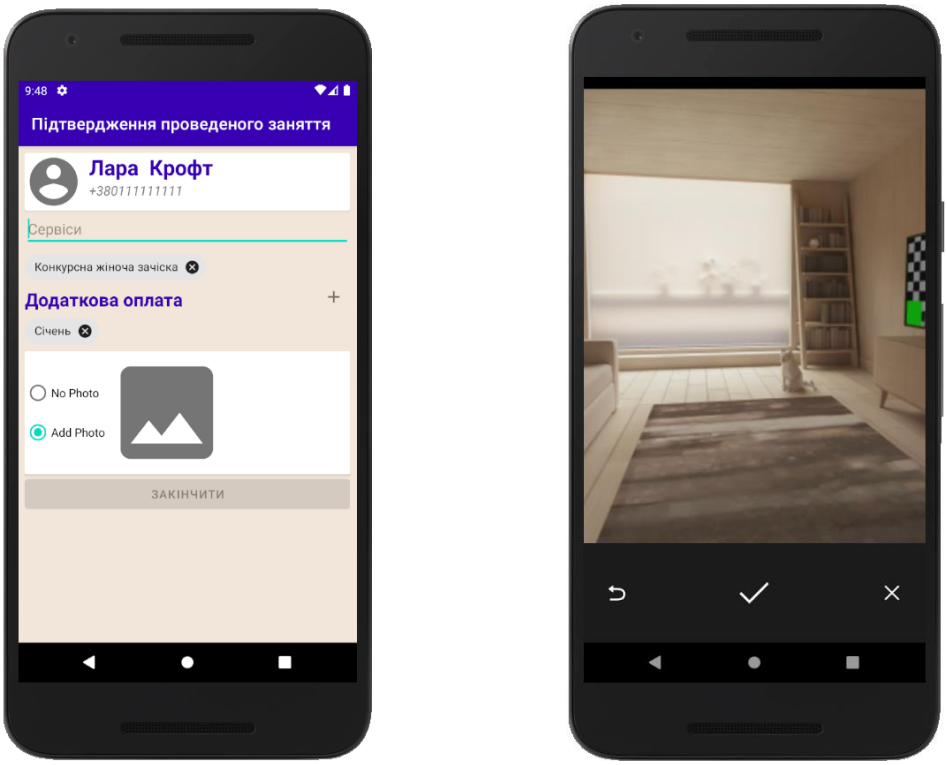


Рисунок 4.32 – Додавання прогресу танцівника у вигляді фотографії (SilhouetteStaffApp)

По завершенню редагування проведеного заняття необхідно натиснути на кнопку «Завершити», після чого виникає вікно, де можна перевірити усі дані про заняття та побачити повну ціну, що має оплатити танцівник (рис. 4.33). Після натискання на кнопку «Підтвердити», на головній сторінці додатку тренера проведене заняття міняє свій колір та підпис (рис. 4.34) та на додаток танцівника, з яким було тренування надсилається повідомлення з оцінюванням тренера та оплатою.

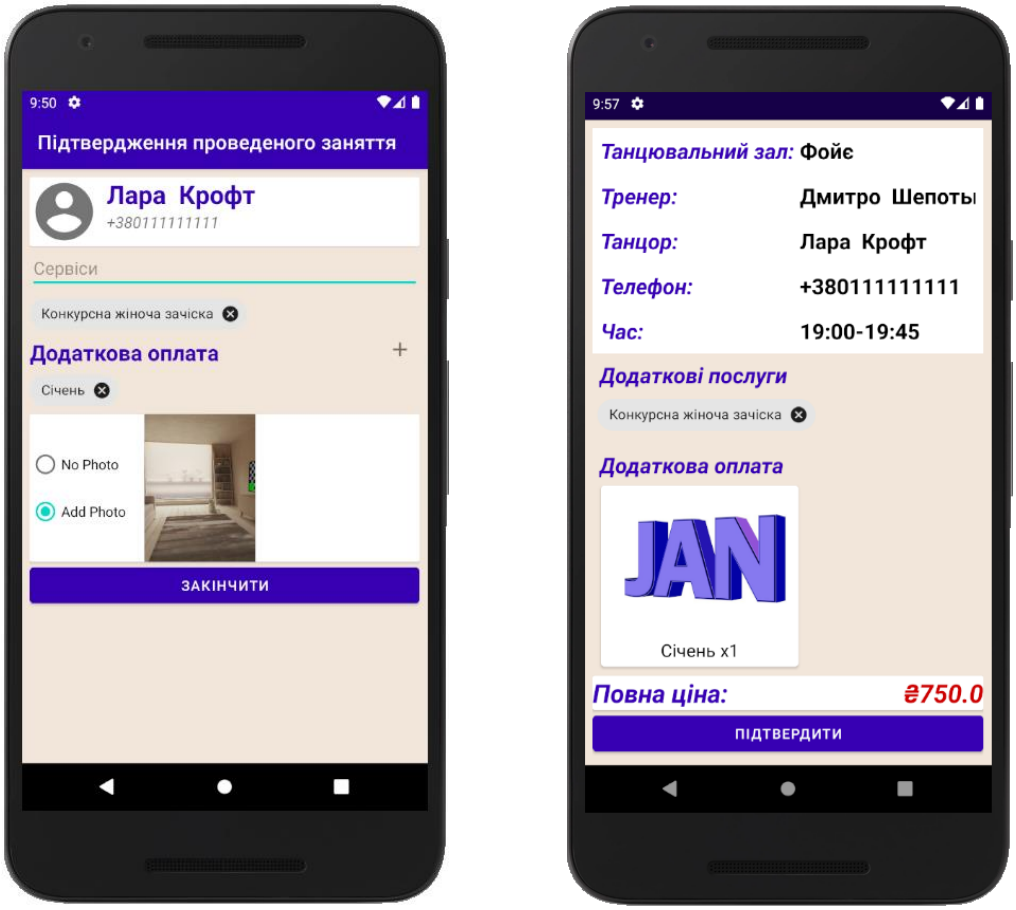


Рисунок 4.33 – Перевірка відредагованого заняття (SilhouetteStaffApp)

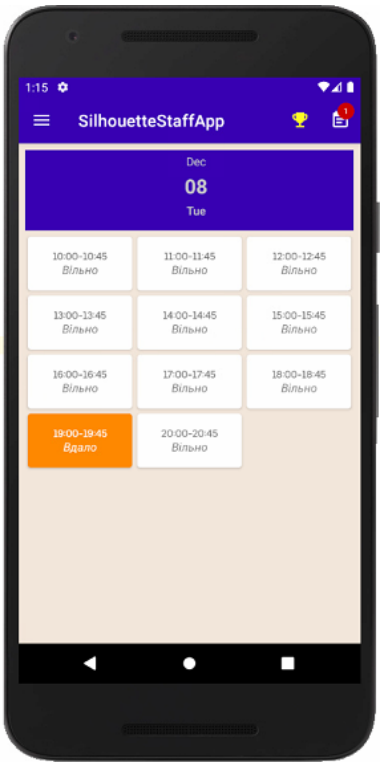


Рисунок 4.34 – Завершене заняття (SilhouetteStaffApp)

Також, аналогічно з додатком танцівника, можна переглядати та обирати майбутніх змагань, натиснувши на іконку з кубком. Приклади представлені у пункті 4.2.1.

ВИСНОВОК

З метою розробки мобільного додатку підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» було досліджено актуальність проблеми. Після огляду аналогів розроблюваного мобільного додатку було прийняте рішення розробити унікальний мобільний додаток, спеціалізований під потреби КСТ Силует..

Для досягнення мети були реалізовані такі задачі: аналіз вимог та аналогів розроблюваного мобільного додатку для підтримки діяльності «Клубу спортивного танцю «Силует», вибір методів та засобів реалізації, проектування і моделювання інтерфейсу, програмна реалізація модулів мобільного додатку і тестування роботи його методом чорного ящику.

Згідно з отриманим технічним завданням були обрані засоби реалізації.

Код програми розроблено на мові Java у середовищі Android Studio через популярність даного середовища та дуже зручного використання.

Для розробки бази даних була обрана нереляційна база даних Cloud Firestore, яка забезпечує дуже гарний рівень безпеки та паралельний доступ до даних.

При реалізації архітектури МД, як клієнт-серверної, було здійснено підключення проектів до Firebase, де зберігається база даних, що виступає у ролі серверу.

Тестування розроблених мобільних додатків методом чорного ящику показало, що отримані додатки задовольняють визначеним вимогам, зручні у використанні.

У результаті розробки отримано мобільний додаток, що забезпечує автоматизовану підтримку основних бізнес-процесів діяльності Громадської організації «Клуб спортивного танцю «Силует».

Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» впроваджено в роботу танцювального клубу, що дозволило підвищити продуктивність виконання основних типів робіт, зменшити людські фактори похибок, підвищити якість діяльності Громадської організації «Клуб

спортивного танцю «Силует», що підтверджується актом впровадження, який можна переглянути у Додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mobile Applications for Agriculture and Rural Development_[Електронний ресурс]. – 2012. – Режим доступу до ресурсу: http://documents1.worldbank.org/curated/en/167301467999716265/pdf/96226-REVISED-WP-PUBLIC-Box391469B-Mobile-Applications-for-ARD-v8S-Complete.pdf/

2. Murugesan, S. Mobile apps in Africa. // IT Professional / , 2013. – С. 8–11.

3. El-Hussein, M.O.M. and Cronje, J.C. Defining Mobile Learning in the Higher Education Landscape.. // Educational Technology & Society / , 2010. – С. 12–21.

4. Mobile Operating System Market Share Worldwide [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/worldwide

5. SDK Platform release notes [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/releases/platforms/ .

6. Distribution dashboard | Android Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/about/dashboards/index.html

7. Growing Importance Of Android Mobile App Development In Business [Електронний ресурс] – Режим доступу до ресурсу: https://jayamwebsolutions.com/blogs/Growing-Importance-of-Android-Mobile-App-Development-In-Business.html

8. Android vs iOS: Which Platform to Build Your App for First? [Електронний ресурс] – Режим доступу до ресурсу: https://bitly.su/H7Ms7I .

9. Android App | Dance Studio Pro [Електронний ресурс] – Режим доступу до ресурсу: https://dancestudio-pro.com/android-app/

10. Fly Dance Studio [Електронний ресурс] – Режим доступу до ресурсу: https://goo.su/2oo3/

11. Move Dance Studio [Електронний ресурс] – Режим доступу до ресурсу:https://play.google.com/store/apps/details?id=com.dancestudio_pro.Move5333&hl=en/

12. Black Box Testing [Електронний ресурс] – Режим доступу до ресурсу: https://softwaretestingfundamentals.com/black-box-

testing/#:~:text=BLACK%20BOX%20TESTING%2C%20also%20known,%2Dfunctional%2C%20though%20usually%20functional/

13. Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://searchmobilecomputing.techtarget.com/definition/Android-Studio

14. Why java for android development? [Електронний ресурс] – Режим доступу до ресурсу: https://www.shibajidebnath.com/java-android-development/

15. What is Firebase? [Електронний ресурс] – Режим доступу до ресурсу: https://blog.back4app.com/firebase/

16. What is Google Firestore? [Електронний ресурс] – Режим доступу до ресурсу: https://blog.back4app.com/what-is-firestore/

17. ERwin Process Modeler [Електронний ресурс] – Режим доступу до ресурсу: https://pro-spo.ru/information-required-to-install/1702-erwin/

18. Методология IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://itteach.ru/bpwin/metodologiya-idef0/

19. Знакомство с нотацией IDEF0 и пример использования [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/trinion/blog/322832/

20. What is Use Case Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

21. Cloud Firestore Data model [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/firestore/data-model/

22. The Value of Schemaless Databases [Електронний ресурс] – Режим доступу до ресурсу: https://blog.couchbase.com/the-value-of-schema-less-databases/

23. Cloud Firestore Store and sync app data at global scale [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/products/firestore

24. Add Firebase to your Android project [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/android/setup#console/

25. Learn more about Android and Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/android/learn-more#firebase-assistant

26. Get Started with Firebase Authentication on Android [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/auth/android/start/

27. Set up a Firebase Cloud Messaging client app on Android [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/cloud-messaging/android/client/

28. Android Gradle plugin release notes [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/releases/gradle-plugin/

29. Add data to Cloud Firestore [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/firestore/manage-data/add-data/

30. Get data with Cloud Firestore [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/firestore/query-data/get-data/

31. Send messages [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/cloud-messaging/android/send-multiple/

Додаток А

Технічне завдання

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку мобільного додатку «Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует»

Суми 2020

1 Призначення й мета створення мобільного додатку

1.1 Призначення мобільного додатку

Мобільний додаток (МД) повинен проводити підтримку записів членів клубу на індивідуальні заняття, оплати за заняття в клубі, обліку відвідування танцівників.

1.2 Мета створення мобільного додатку

Автоматизація бізнес-процесів діяльності Громадської організації «Клуб спортивного танцю «Силует».

1.3 Цільова аудиторія

До цільової аудиторії МД можна віднести такі групи:

1. Тренер;
2. Танцівник.

2 Вимоги до мобільного додатку

2.1 Вимоги до мобільного додатку в цілому

2.1.1 Вимоги до структури й функціонування мобільного додатку

Розроблювані мобільні додатки мають бути реалізовані у вигляді Android-додатку зі зручним для використання інтерфейсом та функціоналом за клієнт-серверною архітектурою.

2.1.2 Вимоги до користувачів

Для використання даного МД від користувачів не вимагаються спеціальні технічні навички, за винятком загальних навичок користування мобільним пристроєм.

2.1.3 Вимоги до збереження інформації

Вся основна інформація буде зберігатися в базі даних Cloud Firestore.

2.1.4 Вимоги до розмежування доступу

Кожним з розроблюваних додатків користується або тренер танцювального клубу, або танцівник.

В мобільному додатку танцівника можна:

- записуватися на індивідуальні заняття;
- оцінювати роботу тренера;
- сплачувати за заняття;
- обирати майбутні змагання.

В мобільному додатку тренера можна:

- завершувати проведене заняття;
- переглядати графік;
- обирати майбутні змагання.

2.2 Вимоги до програмного забезпечення

Головною вимогою є наявність мобільного пристрою з операційної системою Android з підключенням до мережі Інтернет.

Для роботи МД необхідно мати наступні мінімальні системні вимоги:

- Версія платформи Android: 5.0 і вище.
- Роздільна здатність екранів Android: mdpi (320x48px), hdpi (480x800px), xhdpi (720x1280px), xhdpi (768x1280px).

3 Склад і зміст робіт зі створення МД

Детальний опис етапів розроблення мобільного додатку підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» описано у 6 пункті Додатку Б.

4 Порядок контролю та приймання

Перевірка придатного та належного функціонування мобільного додатку здійснюється керівником дипломного проекту та незалежним користувачем з метою виявлення незручностей у використанні та можливих помилок.

Хід виконання дипломного проекту перевіряється за допомогою календарного плану, зазначеного у 6 пункті Додатку Б.

Додаток Б.

Планування робіт

1. Деталізація мети мобільного додатку методом SMART.

Ціллю розробки є отримати мобільний додаток, що реалізує запис членів клубу на індивідуальні заняття, оплата за навчання, облік відвідування танцівників.

Узагальнений вигляд мети полягає у вивченні, як повинен працювати мобільний додаток для підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует».

Загальний вигляд мети полягає в тому, щоб діяльність танцювального клубу здійснювалась за підтримки цього мобільного додатку та, завдяки йому, полегшена в організації.

Реалізація мобільного додатку створена в форматі Android-додатку на основі мови програмування Java та баз даних NoSQL за клієнт-серверною архітектурою.

Найбільш важливими результатами є:

– підключення до бази даних NoSQL з Android Studio;

– паралельна робота мобільних додатків тренерів та членів танцювального клубу з постійним оновленням даних;

– повернення результату.

Мобільний додаток буде виконано вчасно, що підтверджується календарним планом.

2. Планування змісту структури робіт.

WBS-структурою, або структурною декомпозицією робіт (Work Breakdown Structure), є представлення мобільного додатку, яке виконане у вигляді ієрархічної структури робіт та досягається за допомогою послідовної декомпозиції. Дана структура детально планує та проводить оцінку вартості, визначає та розподіляє персональну відповідальність виконавців. На основі робіт і результатів, що

визначають зміст мобільного додатку було побудовано WBS-структуру, яку можна побачити на рис. Б.1.

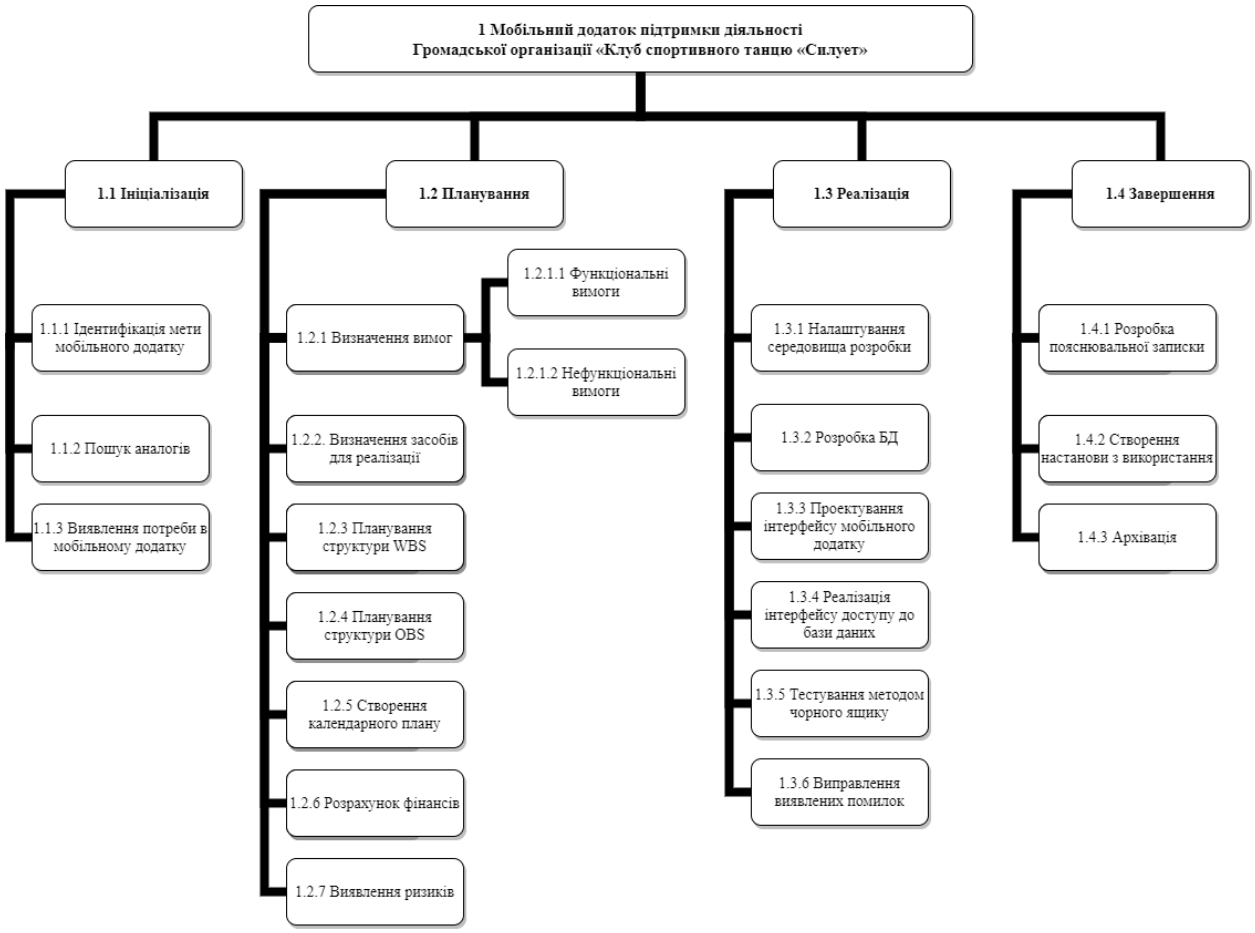


Рисунок Б.1 – WBS структура проекту

3. Планування структури виконавця для впровадження готового мобільного додатку

OBS-структура, або організаційна структура проекту (Organizational Breakdown Structure), представляє собою ієрархічну структуру управління мобільним додатком та показує відносини між її учасниками. За допомогою графічного відображення можна визначити учасників мобільного додатку (фізичних та юридичних осіб) та їхніх відповідальних осіб, що залучені до реалізації даного додатку. На рис. Б.2 показано OBS-структуру.

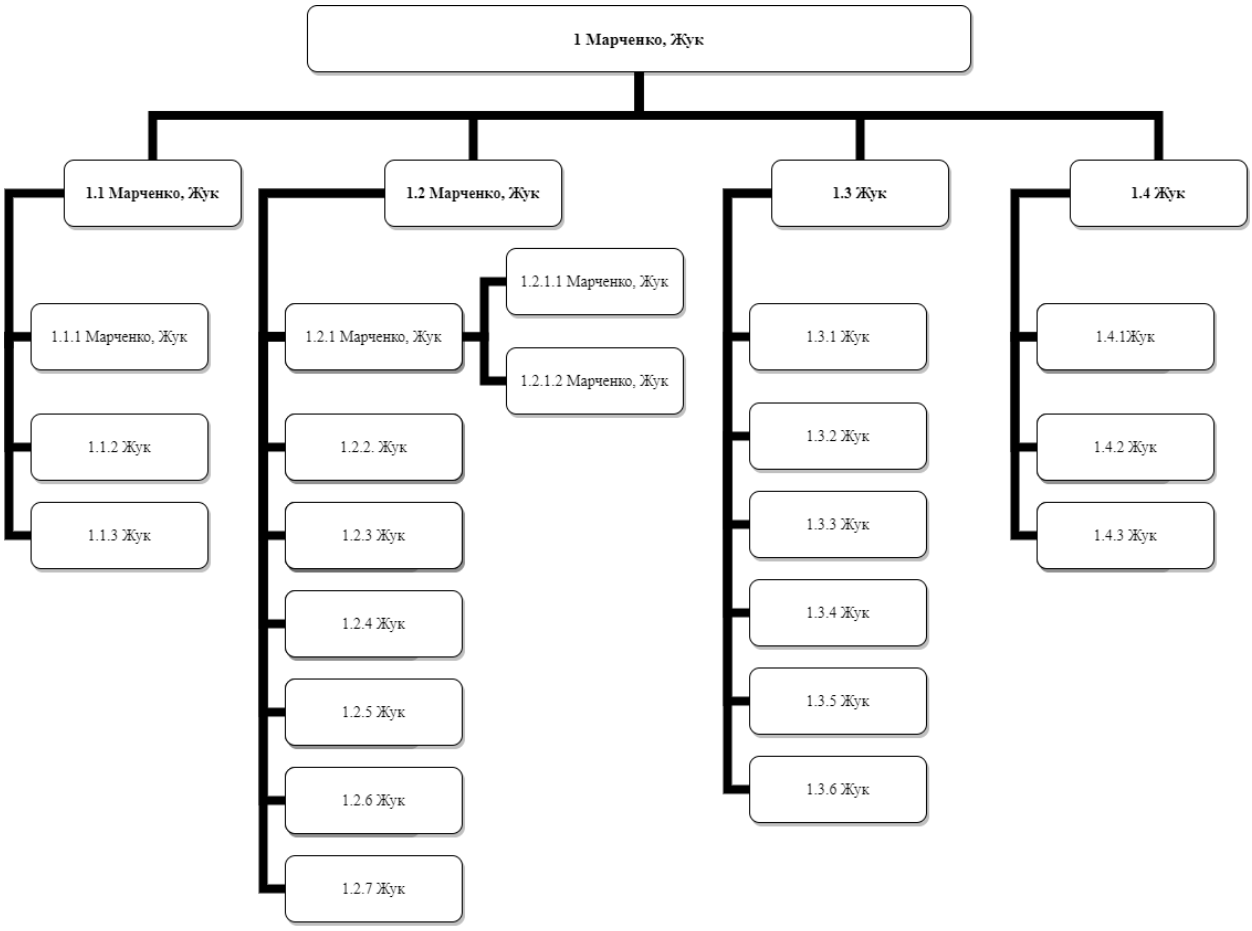


Рисунок Б.2 – OBS структура

4. Побудова матриці відповідальності

Матриця відповідальності використовується для відображення зв’язків між пакетами робіт або операціями та членами команди. Матричний формат показує всі операції, які виконує одна людина або всіх людей, що задіяні в виконанні того чи іншого етапу створення мобільного додатку. А також забезпечує наділення відповідальності одній людині для виконання одного завдання задля попередження різних невідповідностей.

Для побудови матриці відповідальності у вигляді таблиці за основу були взяті OBS та WBS структури. У табл. Б.1 наведено матрицю відповідальності.

Таблиця Б.1 – Матриця відповідальності

WBS\OBS	Жук	Марченко
1.1.1 Ідентифікація мети мобільного додатку		
1.1.2 Пошук аналогів		
1.2.1 Визначення вимог		
1.2.1.1 Функціональні вимоги		
1.2.1.2 Нефункціональні вимоги		
1.2.2 Визначення засобів для реалізації		
1.2.3 Планування структури WBS		
1.2.4 Планування структури OBS		
1.2.5 Створення календарного плану		
1.2.6 Розрахунок фінансів		
1.2.7 Виявлення ризиків		
1.3.1 Налаштування середовища розробки		
1.3.2 Створення бази даних		
1.3.3 Проектування інтерфейсу мобільного додатку		
1.3.4 Реалізація інтерфейсу доступу до бази даних		
1.3.5 Тестування методом чорного ящику		
1.3.7 Виправлення виявлених помилок		
1.4.1 Розробка пояснювальної записки		
1.4.2 Створення настанови з використання		
1.4.3 Архівація		

5. Розробка PDM мережі

PDM мережа була побудована за допомогою програми MS Project 2016. Мережу зображено на рис. Б.3. Використання моделі мобільного додатку, побудованої в даному програмному середовищі, дозволяє контролювати і оптимізувати план виконання робіт, наочно відстежувати хід його виконання.

Метод побудови PDM мережі спрямовано на аналіз часу, потрібного для виконання кожного окремого етапу створення мобільного додатку, а також мінімального необхідного часу для виконання мобільного додатку.

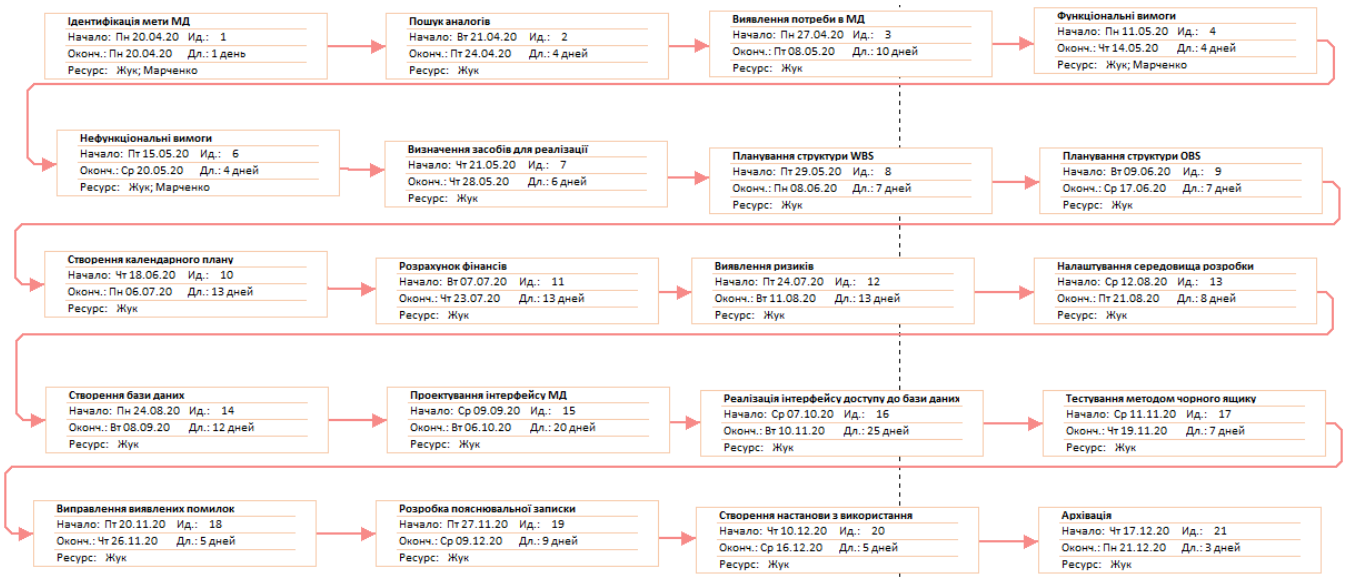


Рисунок Б.3 – PDM-мережа

6. Побудова календарного графіку виконання мобільного додатку

Для того щоб мати реальне уявлення про тривалість виконання окремих робіт з урахуванням обмежень у використанні ресурсів, а також тривалість в цілому з урахуванням вихідних та святкових днів, будують календарний графік робіт, що також називають діаграмою Ганта. Дану діаграму наведено на рис. Б.4.

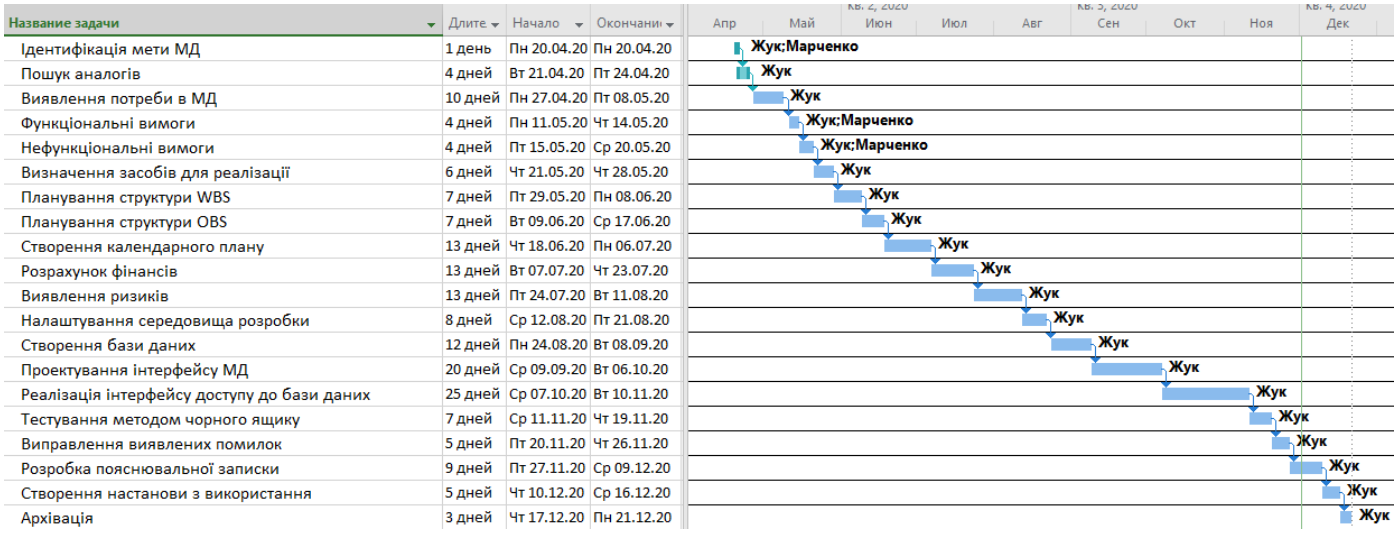


Рисунок Б.4 – Календарний графік робіт

7. Управління ризиками мобільного додатку

Перед початком реалізації мобільного додатку необхідно передбачити усі можливі ризики, що можуть виникнути та вплинути на якість та час його розробки.

Були виявлені такі ризики:

- R1 – зміна ТЗ на етапі розробки;
- R2 – не виявлені помилки на етапі розробки мобільного додатку;
- R3 – порушено дотримання календарного плану;
- R4 – несанкціонований доступ до бази даних;
- R5 – хвороба розробника;
- R6 – некоректне тестування;
- R7 – некоректна робота мобільного додатку.

За допомогою оцінки були визначені ймовірності появи усіх можливих ризиків. Згідно з цим було побудовано табл. Б.2.

Таблиця Б.2 – Ймовірність виникнення ризиків

Ймовірність виникнення	R1	R2	R3	R4	R5	R6	R7
Слабо ймовірний	■			■			
Малоймовірний		■	■			■	
Ймовірний					■		
Дуже ймовірний						■	
Майже можливий							■

Наступним кроком було зроблено таблицю тих втрат, які можуть з’явитися в процесі проектування та розробки при виникненні ризиків (табл. Б.3).

Таблиця Б.3 – Втрати при виникненні ризиків

Значимість впливу	R1	R2	R3	R4	R5	R6	R7
мінімальна						■	
низька						■	
середня					■		
висока							■
максимальна	■	■		■			

Після побудови двох таблиць була створена матриця впливу (ймовірностей і наслідків) ризиків(табл. Б), де білим кольором позначено неважливі ризики, темнішим – помірні, дуже темним – критичні.

Таблиця Б.4 – Матриця ймовірність-втрати

Ймовірність			
		R2	
		R4	R6
	R3		
		R1	R5
	R7		
Вплив			

Виходячи з цього, було визначено три критичних ризики, такі як:

- R2 – не виявлені помилки на етапі розробки мобільного додатку;
- R4 – несанкціонований доступ до бази даних;
- R6 – некоректне тестування;

Виникнення першого критичного ризику можна уникнути лише за допомогою розробника, а саме через зміну програмного коду. Другий критичний ризик малоймовірний, але при його виникненні потрібне буде втручання розробника, який зможе налагодити доступ до бази даних. Третій критичний ризик, а саме некоректне тестування, може створити проблеми у подальшому використанні вже реалізованого мобільного додатку.

8. Формування бюджету мобільного додатку

Завершенням планування мобільного додатку є етап розподілу бюджету. Перед тим, як визначити бюджет, необхідно виділити тих виконавців, які брали участь в даній роботі. Були визначені такі працівники як:

- Розробник (Жук);
- Керівник (Марченко);

Бюджет – це загальні майбутні витрати, які необхідні безпосередньо для створення мобільного додатку. Тобто це витрати на фінансування всіх робіт, передбачених WBS-структурою. Витрати розподіляються у часі на основі календарного плану реалізації робіт. Визначення бюджету на витрачання заробітної плати (табл. Б.5).

Таблиця Б.5 – Бюджет на заробітну плату

Посада	За 1 год	Робочих годин	Сума
Розробник	180	176	31680
Керівник	170	18	3060

Отже, бюджет заробітної плати складає *34740 грн*.

Також на менеджмент мобільного додатку було додано 20% з загального бюджету. Тому загальний бюджет дорівнює *41688 грн.*

Додаток В.

Лістинг коду

HomeFragment.java(SlhouetteClientApp)

```
package com.example.silhouetteclientapp.Fragments;


import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.cardview.widget.CardView;
import androidx.fragment.app.Fragment;


import android.text.TextUtils;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.example.silhouetteclientapp.Adapter.HomeSliderAdapter;
import com.example.silhouetteclientapp.CartActivity;
import com.example.silhouetteclientapp.Common.Common;
import com.example.silhouetteclientapp.CompetitionActivity;
import com.example.silhouetteclientapp.Database.CartDataSource;
import com.example.silhouetteclientapp.Database.CartDatabase;
import com.example.silhouetteclientapp.Database.LocalCartDataSource;
import com.example.silhouetteclientapp.Interface.IBannerLoadListener;
import com.example.silhouetteclientapp.Interface.ILessonInfoLoadListener;
import com.example.silhouetteclientapp.Interface.ILessonInformationChangeListener;
import com.example.silhouetteclientapp.LessonActivity;
import com.example.silhouetteclientapp.MainActivity;
import com.example.silhouetteclientapp.Model.Banner;
import com.example.silhouetteclientapp.Model.Competition;
import com.example.silhouetteclientapp.Model.LessonInformation;
import com.example.silhouetteclientapp.R;
import com.example.silhouetteclientapp.Service.PicassoImageLoadingService;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.Timestamp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
```

```
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.ListenerRegistration;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.nex3z.notificationbadge.NotificationBadge;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;
import dmax.dialog.SpotsDialog;
import io.paperdb.Paper;
import io.reactivex.SingleObserver;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.disposables.Disposable;
import io.reactivex.schedulers.Schedulers;
import ss.com.bannerslider.Slider;



public class HomeFragment extends Fragment implements IBannerLoadListener, ILessonInfoLoadListener,
ILessonInformationChangeListener {


    private Unbinder unbinder;

    AlertDialog dialog;

    CartDataSource cartDataSource;

    @BindView(R.id.notification_badge)
    NotificationBadge notificationBadge;

    @BindView(R.id.layout_user_information )
    LinearLayout layout_user_information;
    @BindView(R.id.txt_user_name)
    TextView  txt_user_name;
    @BindView(R.id.txt_user_surname)
    TextView  txt_user_surname;
    @BindView(R.id.banner_slider)
    Slider banner_slider;

    @BindView(R.id.card_lesson_info)
    CardView card_lesson_info;
       @BindView(R.id.txt_dance_hall_address)
    TextView txt_dance_hall_address;
    @BindView(R.id.txt_dance_hall_coach_name)
    TextView txt_dance_hall_coach_name;
    @BindView(R.id.txt_dance_hall_coach_surname)
    TextView txt_dance_hall_coach_surname;
    @BindView(R.id.txt_time)
    TextView txt_time;
    @BindView(R.id.txt_time_remain)
    TextView txt_time_remain;

    @OnClick(R.id.layout_user_information)
    void onLogOutDialog() {
```

```
    AlertDialog.Builder builder=new AlertDialog.Builder(getContext());
   builder.setTitle("ВИХІД")
          .setMessage("Ви впевнені, що бажаєте вийти з аккаунту?")
          .setCancelable(false)
          .setPositiveButton("ТАК", (dialog, which) -> {
              Common.currentCoach=null;
              Common.currentLesson=null;
              Common.currentDanceHall=null;
              Common.currentLessonId="";
              Common.currentTimeSlot=-1;
              Common.currentUser=null;
              FirebaseAuth.getInstance().signOut();

              Intent intent = new Intent(getContext(), MainActivity.class);
              intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
              startActivity(intent);
              getActivity().finish();

          }).setNegativeButton("СКАСУВАТИ", (dialog, which) -> dialog.dismiss());
   AlertDialog dialog =builder.create();
   dialog.show();



}

@OnClick(R.id.btn_delete_lesson)
void deleteLesson() {
    deleteLessonFromCoach(false);
}

@OnClick(R.id.btn_change_lesson)
void changeLesson()  {
    changeLessonFromUser();
}

private void changeLessonFromUser() {
    // Show dialog
    AlertDialog.Builder confirmDialog = new AlertDialog.Builder(getActivity())
          .setCancelable(false)
          .setTitle("Підтвердження")
          .setMessage("Ви впевнені, що хочете змінити ваш запис на заняття?\\nВаш поточний запис буде видалено.")
          .setNegativeButton("СКАСУВАТИ", new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialog, int which) {
                  dialog.dismiss();
              }
          }).setPositiveButton("ЗМІНИТИ", new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialog, int which) {
                  deleteLessonFromCoach(true);

              }
          });
    confirmDialog.show();
}

private void deleteLessonFromCoach(boolean isChanged) {

    if(Common.currentLesson != null) {

        dialog.show();
```

```
        // Get lesson information in coach object
        DocumentReference coachLessonInfo = FirebaseFirestore.getInstance()
                .collection("Places")
                .document(Common.currentLesson.getPlaceLesson())
                .collection("DanceHall")
                .document(Common.currentLesson.getDanceHallId())
                .collection("Coach")
                .document(Common.currentLesson.getCoachId())
                .collection(Common.convertTimeStampToStringKey(Common.currentLesson.getTimestamp()))
                .document(Common.currentLesson.getSlot().toString());

        // Delete
        coachLessonInfo.delete().addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(getContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        }).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Delete from User
                deleteLessonFromUser(isChanged);

            }
        });

    } else {
        Toast.makeText(getContext(), "Current Lesson must not be null", Toast.LENGTH_SHORT).show();
    }


}


private void deleteLessonFromUser(boolean isChanged) {
    if(!TextUtils.isEmpty(Common.currentLessonId)) {
        DocumentReference userBookingInfo = FirebaseFirestore.getInstance()
                .collection("User")
                .document(Common.currentUser.getPhone())
                .collection("Lesson")
                .document(Common.currentLessonId);

        // Delete
        userBookingInfo.delete().addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(getContext(), e.getMessage(), Toast.LENGTH_SHORT).show();

            }
        }).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Delete from Calendar
                try {
                    Paper.init(getActivity());
                    if(Paper.book().read(Common.EVENT_URI_CACHE) != null) {
                        String eventString = Paper.book().read(Common.EVENT_URI_CACHE).toString();
                        Uri eventUri = null;
                        if(eventString != null && !TextUtils.isEmpty(eventString)) {
                            eventUri = Uri.parse(eventString);
                        }
```

```
                if(eventUri != null) {
                    getActivity().getContentResolver().delete(eventUri, null, null);
                }
            }
        } catch(Exception e) {

        }


        Toast.makeText(getActivity(), "Запис видалено вдало!", Toast.LENGTH_SHORT).show();

        // Refresh
        loadUserLesson();

        // Check if booking has changed
        if(isChanged) {
            iLessonInformationChangeListener.onLessonInformationChange();
        }
        dialog.dismiss();

      }
    });

  }
  else {
    dialog.dismiss();
    Toast.makeText(getContext(), "Lesson information must not be empty", Toast.LENGTH_SHORT).show();
  }
}


@OnClick(R.id.card_view_lesson)
void lesson() {
  startActivity(new Intent(getActivity(), LessonActivity.class));
}

@OnClick(R.id.card_view_competition)
void competition() {
  startActivity(new Intent(getActivity(), CompetitionActivity.class));
}

@OnClick(R.id.card_view_cart)
void openCartActivity() {

  if(cartDataSource.countItemsInCart(Common.currentUser.getPhone())==null)
    Toast.makeText(getContext(), "В корзині нічого немає!", Toast.LENGTH_SHORT).show();
  else
  startActivity(new Intent(getActivity(), CartActivity.class));
}


// FireStore
CollectionReference bannerRef;

// Interface
IBannerLoadListener iBannerLoadListener;
ILessonInfoLoadListener iLessonInfoLoadListener;
ILessonInformationChangeListener iLessonInformationChangeListener;

ListenerRegistration userLessonListener = null;
EventListener<QuerySnapshot> userLessonEvent = null;
```

```
    public HomeFragment() {
        bannerRef = FirebaseFirestore.getInstance().collection("Banner");

    }

    @Override
    public void onResume() {
        super.onResume();
        loadUserLesson();
        countCartItems();
    }

    private void loadUserLesson() {
        CollectionReference userLesson = FirebaseFirestore.getInstance()
                .collection("User")
                .document(Common.currentUser.getPhone())
                .collection("Lesson");

        // Get current date
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.DATE, 0);
        calendar.set(Calendar.HOUR_OF_DAY, 0);
        calendar.set(Calendar.MINUTE, 0);

        Timestamp todayTimeStamp = new Timestamp(calendar.getTime());

        // Select lesson information from Firebase where timestamp >= today and done=false
        userLesson
                .whereGreaterThanOrEqualTo("timestamp", todayTimeStamp)
                .whereEqualTo("done", false)
                .limit(1) // take 1
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if(task.isSuccessful()) {
                            if(!task.getResult().isEmpty()) {
                                for(QueryDocumentSnapshot queryDocumentSnapshot:task.getResult()) {
                                    LessonInformation lessonInformation = queryDocumentSnapshot.toObject(LessonInformation.class);
                                    iLessonInfoLoadListener.onLessonInfoLoadSuccess(lessonInformation,
queryDocumentSnapshot.getId());
                                    break; // Exit loop as soon as possible
                                }

                            }
                            else {
                                iLessonInfoLoadListener.onLessonInfoLoadEmpty();

                            }
                        }

                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                iLessonInfoLoadListener.onLessonInfoLoadFailed(e.getMessage() );

            }
        });

        // Make real time listener
        if(userLessonEvent != null) {
```

```
        if(userLessonListener == null) {
            userLessonListener = userLesson
                    .addSnapshotListener(userLessonEvent);
        }


    }
}

@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    dialog = new SpotsDialog.Builder().setContext(getContext()).setCancelable(false).build();

}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_home, container, false);
    unbinder = ButterKnife.bind(this, view);

    cartDataSource = new LocalCartDataSource(CartDatabase.getInstance(getContext()).cartDAO());

    // Init
    Slider.init(new PicassoImageLoadingService());
    iBannerLoadListener = this;
    iLessonInfoLoadListener = this;
    iLessonInformationChangeListener = this;

    // Check if logged
    FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();

    if(user!=null) {
        setUserInformation();
        loadBanner();
        initRealTimeUserBooking();
        loadUserLesson();
        countCartItems();
    }

    return view;
}

private void initRealTimeUserBooking() {
    if(userLessonEvent == null) {
        userLessonEvent = new EventListener<QuerySnapshot>() {

            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException
e) {
                loadUserLesson();

            }
        };
    }
}

private void countCartItems() {
    cartDataSource.countItemsInCart(Common.currentUser.getPhone())
```

```
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(new SingleObserver<Integer>() {
                @Override
                public void onSubscribe(Disposable d) {

                }

                @Override
                public void onSuccess(Integer integer) {
                    notificationBadge.setText(String.valueOf(integer));

                }

                @Override
                public void onError(Throwable e) {
                    Toast.makeText(getContext(), ""+e.getMessage(), Toast.LENGTH_SHORT).show();

                }
            });

    }

    private void loadBanner() {
        bannerRef.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                List<Banner> banners = new ArrayList<>();
                if(task.isSuccessful()) {
                    for(QueryDocumentSnapshot bannerSnapshot:task.getResult()) {
                        Banner banner = bannerSnapshot.toObject(Banner.class);
                        banners.add(banner);
                    }
                    iBannerLoadListener.onBannerLoadSuccess(banners);
                }

            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                iBannerLoadListener.onBannerLoadFailed(e.getMessage());
            }
        });
    }

    private void setUserInformation() {
        layout_user_information.setVisibility(View.VISIBLE);
        txt_user_name.setText(Common.currentUser.getName());
        txt_user_surname.setText(Common.currentUser.getSurname());
    }


    @Override
    public void onBannerLoadSuccess(List<Banner> banners) {
        banner_slider.setAdapter(new HomeSliderAdapter(banners));

    }

    @Override
    public void onBannerLoadFailed(String message) {
        Toast.makeText(getActivity(), message , Toast.LENGTH_SHORT).show();
```

```
    }

    @Override
    public void onLessonInfoLoadEmpty() {
        card_lesson_info.setVisibility(View.GONE);

    }

    @Override
    public void onLessonInfoLoadSuccess(LessonInformation lessonInformation, String lessonId) {

        Common.currentLesson = lessonInformation;
        Common.currentLessonId = lessonId;

        txt_dance_hall_address.setText(lessonInformation.getDanceHallAddress());
        txt_dance_hall_coach_name.setText(lessonInformation.getCoachName());
        txt_dance_hall_coach_surname.setText(lessonInformation.getCoachSurname());
        txt_time.setText(lessonInformation.getTime());
        String dateRemain = DateUtils.getRelativeTimeSpanString(
                Long.valueOf(lessonInformation.getTimestamp().toDate().getTime()),
                Calendar.getInstance().getTimeInMillis(), 0).toString();

        txt_time_remain.setText(dateRemain);

        card_lesson_info.setVisibility(View.VISIBLE );

        dialog.dismiss();



    }

    @Override
    public void onLessonInfoLoadFailed(String message) {
        Toast.makeText(getContext(), message, Toast.LENGTH_SHORT).show();

    }

    @Override
    public void onLessonInformationChange() {
        // Start activity Booking
        startActivity(new Intent(getActivity(), LessonActivity.class));
    }

    @Override
    public void onDestroy() {
        if(userLessonListener != null) {
            userLessonListener.remove();
        }
        super.onDestroy();
    }
}
```

Common.java (SilhouetteClientApp)

```
package com.example.silhouetteclientapp.Common;

import android.app.AlertDialog;
import android.app.Notification;
```

```
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Build;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.AppCompatRatingBar;
import androidx.core.app.NotificationCompat;

import com.example.silhouetteclientapp.Model.Coach;
import com.example.silhouetteclientapp.Model.DanceHall;
import com.example.silhouetteclientapp.Model.LessonInformation;
import com.example.silhouetteclientapp.Model.MyToken;
import com.example.silhouetteclientapp.Model.User;
import com.example.silhouetteclientapp.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.Timestamp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import io.paperdb.Paper;

public class Common {
    public static final String KEY_ENABLE_BUTTON_NEXT = "ENABLE_BUTTON_NEXT";
    public static final String KEY_DANCEHALL_STORE = "DANCEHALL_SAVE";
    public static final String KEY_COACH_LOAD_DONE = "COACH_LOAD_DONE";
    public static final String KEY_DISPLAY_TIME_SLOT = "DISPLAY_TIME_SLOT";
    public static final String KEY_STEP = "STEP";
    public static final String KEY_COACH_SELECTED = "COACH_SELECTED";
    public static final int TIME_SLOT_TOTAL = 11 ;
    public static final Object DISABLE_TAG = "DISABLE";
    public static final String KEY_TIME_SLOT = "TIME_SLOT";
    public static final String KEY_CONFIRM_LESSON = "CONFIRM_LESSON";
    public static final String EVENT_URI_CACHE = "URI_EVENT_SAVE";
    public static final String TITLE_KEY = "title";
    public static final String CONTENT_KEY = "content";
    public static final String LOGGED_KEY = "UserLogged";
    public static final String RATING_INFORMATION_KEY = "RATING_INFORMATION";

    public static final String RATING_PLACE_KEY = "RATING_PLACE";
    public static final String RATING_DANCEHALL_ID = "RATING_DANCEHALL_ID";
```

```
public static final String RATING_DANCEHALL_NAME = "RATING_DANCEHALL_NAME";
public static final String RATING_COACH_ID = "RATING_COACH_ID";
public static final String TOTAL_PRICE = "TOTAL_PRICE";

public static String IS_LOGIN = "IsLogin";
public static User currentUser;
public static DanceHall currentDanceHall;
public static Coach currentCoach;
public static int step = 0;
public static String place = "";
public static int currentTimeSlot = -1 ;
public static Calendar lessonDate = Calendar.getInstance();
public static SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd_MM_yyyy");
public static LessonInformation currentLesson;
public static String currentLessonId="";

public static String convertTimeSlotToString(int slot) {
   switch (slot)
   {
      case 0:
         return "10:00-10:45";
      case 1:
         return "11:00-11:45";
      case 2:
         return "12:00-12:45";
      case 3:
         return "13:00-13:45";
      case 4:
         return "14:00-14:45";
      case 5:
         return "15:00-15:45";
      case 6:
         return "16:00-16:45";
      case 7:
         return "17:00-17:45";
      case 8:
         return "18:00-18:45";
      case 9:
         return "19:00-19:45";
      case 10:
         return "20:00-20:45";
      default:
         return "Closed";
   }
}

public static String convertTimeStampToStringKey(Timestamp timestamp) {
   Date date = timestamp.toDate();
   SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd_MM_yyyy");
   return simpleDateFormat.format(date);
}

public static String formatPaymentItemName(String name) {
   return name.length() > 13 ? new StringBuilder(name.substring(0,10)).append("...").toString() : name;
}

public static void showNotification(Context context, int noti_id, String title, String content, Intent intent) {
   PendingIntent pendingIntent = null;
   if(intent != null) {
      pendingIntent = PendingIntent.getActivity(
            context,
            noti_id,
```

```
            intent,
            PendingIntent.FLAG_UPDATE_CURRENT);
    }
    String NOTIFICATION_CHANNEL_ID = "silhouette_client_app";
    NotificationManager notificationManager =
(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);

    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
      NotificationChannel notificationChannel = new NotificationChannel(NOTIFICATION_CHANNEL_ID,
            "Silhouette Staff App ",
            NotificationManager.IMPORTANCE_DEFAULT);

      // Configure the notification channel
      notificationChannel.setDescription("Staff app");
      notificationChannel.enableLights(true);
      notificationChannel.setLightColor(Color.RED);
      notificationChannel.setVibrationPattern(new long[]{0, 1000, 500, 1000});
      notificationChannel.enableVibration(true);
      notificationManager.createNotificationChannel(notificationChannel);
    }

    NotificationCompat.Builder builder = new NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID);
    builder
        .setContentTitle(title)
        .setContentText(content)
        .setAutoCancel(false)
        .setSmallIcon(R.mipmap.ic_launcher)
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), R.mipmap.ic_launcher));


    if(pendingIntent != null)
      builder.setContentIntent(pendingIntent);
    Notification mNotification = builder.build();

    notificationManager.notify(noti_id, mNotification);


  }

  public static void showRatingDialog(Context context, String placeName, String danceHallId, String danceHallName, String
coachId, String price) {
    DocumentReference coachNeedRateRef = FirebaseFirestore.getInstance()
        .collection("Places")
        .document(placeName)
        .collection("DanceHall")
        .document(danceHallId)
        .collection("Coach")
        .document(coachId);

    coachNeedRateRef.get()
        .addOnFailureListener(new OnFailureListener() {
          @Override
          public void onFailure(@NonNull Exception e) {
            Toast.makeText(context, e.getMessage(), Toast.LENGTH_SHORT).show();
          }
        }).addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
      @Override
      public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if(task.isSuccessful()) {
          Coach coachRate = task.getResult().toObject(Coach.class);
          coachRate.setCoachId(task.getResult().getId());
```

```
// Create view for dialog
View view = LayoutInflater.from(context).inflate(R.layout.layout_rating_dialog, null);

// Widget
TextView txt_dance_hall_name = (TextView)view.findViewById(R.id.txt_dance_hall_name);
TextView txt_coach_name = (TextView)view.findViewById(R.id.txt_coach_name);
TextView txt_coach_surname = (TextView)view.findViewById(R.id.txt_coach_surname);
TextView txt_total_price = (TextView)view.findViewById(R.id.txt_total_price);
AppCompatRatingBar ratingBar = (AppCompatRatingBar)view.findViewById(R.id.rating);

// Set information
txt_coach_name.setText(coachRate.getName());
txt_coach_surname.setText(coachRate.getSurname());
txt_dance_hall_name.setText(danceHallName);
txt_total_price.setText(price);


// Create dialog
AlertDialog.Builder builder = new AlertDialog.Builder(context)
      .setView(view)
      .setCancelable(false)
      .setPositiveButton("OK", new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int which) {
            // Update rating in FireStore
            Double original_rating = coachRate.getRating();
            Long ratingTimes = coachRate.getRatingTimes();
            float userRating = ratingBar.getRating();

            Double finalRating = (original_rating + userRating);

            // Update coach
            Map<String,Object> data_update = new HashMap<>();
            data_update.put("rating", finalRating);
            data_update.put("ratingTimes", ++ratingTimes);

            coachNeedRateRef.update(data_update)
                  .addOnFailureListener(new OnFailureListener() {
                     @Override
                     public void onFailure(@NonNull Exception e) {
                        Toast.makeText(context, ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                     }
                  }).addOnCompleteListener(new OnCompleteListener<Void>() {
               @Override
               public void onComplete(@NonNull Task<Void> task) {
                  if(task.isSuccessful()) {
                     Toast.makeText(context, "Дякуємо за оцінку!", Toast.LENGTH_SHORT).show();

                     // Remove key
                     Paper.init(context);
                     Paper.book().delete(Common.RATING_INFORMATION_KEY);
                  }
               }
            });



         }
      }).setNegativeButton("ПРОПУСТИТИ", (dialog, which) -> {
         // Dismiss dialog
         dialog.dismiss();
      }).setNeutralButton("НІКОЛИ", (dialog, which) -> {
```

```
                    // Never ask for rating
                    Paper.init(context);
                    Paper.book().delete(Common.RATING_INFORMATION_KEY);

                });
            AlertDialog dialog = builder.create();
            dialog.show();

        }
    }
});



}


public static enum TOKEN_TYPE {
    CLIENT,
    COACH,
    MANAGER
}

public static void updateToken(Context context, String token) {

    FirebaseUser user= FirebaseAuth.getInstance().getCurrentUser();
    if(user != null)
    {
        if(!TextUtils.isEmpty(user.getPhoneNumber()))
        {
            MyToken myToken= new MyToken();
            myToken.setToken(token);
            myToken.setToken_type(TOKEN_TYPE.CLIENT);
            myToken.setUser(user.getPhoneNumber());

            // Submit to FireStore
            FirebaseFirestore.getInstance()
                    .collection("Tokens")
                    .document(user.getPhoneNumber())
                    .set(myToken)
                    .addOnCompleteListener(task -> {
                    });
        }
    } else {
        Paper.init(context);
        String localUser = Paper.book().read(Common.LOGGED_KEY);
        if(localUser != null) {
            if(!TextUtils.isEmpty(localUser)) {
                MyToken myToken= new MyToken();
                myToken.setToken(token);
                myToken.setToken_type(TOKEN_TYPE.CLIENT);
                myToken.setUser(localUser);

                // Submit to FireStore
                FirebaseFirestore.getInstance()
                        .collection("Tokens")
                        .document(localUser)
                        .set(myToken)
                        .addOnCompleteListener(task -> {
                        });
            }
        }
```

```
        }

    }
}
```

CompetitionActivity.java

```
package com.example.silhouetteclientapp;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.silhouetteclientapp.Adapter.MyCompetitionAdapter;
import com.example.silhouetteclientapp.Model.Competition;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

public class CompetitionActivity extends AppCompatActivity {

    Document doc;

    private Thread secThread;
    private final String MY_LOG="MYLOG";
    private Runnable runnable;

    @BindView(R.id.listView)
    ListView listView;



    private MyCompetitionAdapter adapter;
    private List<Competition> arrayList;



    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_competition);
        ButterKnife.bind(this);
```

```
    init();
}

private void init() {

    listView = findViewById(R.id.listView);
    arrayList = new ArrayList<>();
    adapter = new MyCompetitionAdapter(this,R.layout.layout_competition_item,arrayList,getLayoutInflater());

    listView.setAdapter(adapter);
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

            String link_comp=arrayList.get(position).getLink();


            Intent intent = new Intent(com.example.silhouetteclientapp.CompetitionActivity.this, PDFActivity.class);
            intent.putExtra("link_comp", link_comp);
            startActivity(intent);
        }
    });
    runnable=new Runnable() {
        @Override
        public void run() {
            getWeb();
        }
    };
    secThread=new Thread(runnable);
    secThread.start();

}

private void getWeb() {
    String urlHome="https://www.audsf.org/";


    try {
        doc = Jsoup.connect(urlHome+"competition/2020").get();
        Elements tables=doc.getElementsByClass("tournament");
        Element our_table=tables.get(0);
        Elements elements_from_table=our_table.children();


        for(int i = 0;i < our_table.childrenSize();i++ )
        {
            if(our_table.children().get(i).child(1).text().equals("Реєстрація")) {
                Competition items = new Competition();
                items.setDate(elements_from_table.get(i).child(0).text());
                items.setCity(elements_from_table.get(i).child(4).text());
                items.setTitle(elements_from_table.get(i).child(2).text());
                items.setOrganizer(elements_from_table.get(i).child(5).text());

                Element info=elements_from_table.get(i).child(13);
                Elements info_elements=info.getElementsByClass("btn");
                String entry=info_elements.get(0).attr("data-href");

                items.setLink(urlHome+info_elements.get(0).text()+"/"+entry);

                Log.d(MY_LOG, items.getLink());
```

```
                arrayList.add(items);
            }
        }
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                adapter.notifyDataSetChanged();
            }
        });

    } catch (IOException e) {
        e.printStackTrace();
    }

  }


}
```

TotalPriceFragment.java

```
package com.example.silhouettestaffapp.Fragments;

import android.app.AlertDialog;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.silhouettestaffapp.Adapter.MyConfirmPaymentItemAdapter;
import com.example.silhouettestaffapp.Common.Common;
import com.example.silhouettestaffapp.Model.CartItem;
import com.example.silhouettestaffapp.Model.CoachService;
import com.example.silhouettestaffapp.Model.EventBus.DismissFromBottomSheetEvent;
import com.example.silhouettestaffapp.Model.FCMResponse;
import com.example.silhouettestaffapp.Model.FCMSendData;
import com.example.silhouettestaffapp.Model.Invoice;
import com.example.silhouettestaffapp.Model.MyToken;
import com.example.silhouettestaffapp.R;
import com.example.silhouettestaffapp.Retrofit.IFCMService;
import com.example.silhouettestaffapp.Retrofit.RetrofitClient;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomsheet.BottomSheetDialogFragment;
import com.google.android.material.chip.Chip;
import com.google.android.material.chip.ChipGroup;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;
```

```
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import org.greenrobot.eventbus.EventBus;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;
import dmax.dialog.SpotsDialog;
import io.reactivex.functions.Consumer;
import io.reactivex.schedulers.Schedulers;

public class TotalPriceFragment extends BottomSheetDialogFragment {

    Unbinder unbinder;

    @BindView(R.id.chip_group_services)
    ChipGroup chip_group_service;

    @BindView(R.id.recycler_view_payment)
    RecyclerView recycler_view_payment;

    @BindView(R.id.txt_dance_hall_name)
    TextView txt_salon_name;

    @BindView(R.id.txt_coach_name)
    TextView txt_coach_name;

    @BindView(R.id.txt_coach_surname)
    TextView txt_coach_surname;

    @BindView(R.id.txt_time)
    TextView txt_time;

    @BindView(R.id.txt_client_name)
    TextView txt_client_name;

    @BindView(R.id.txt_client_surname)
    TextView txt_client_surname;

    @BindView(R.id.txt_client_phone)
    TextView txt_client_phone;

    @BindView(R.id.txt_total_price)
    TextView txt_total_price;

    @BindView(R.id.btn_confirm)
    Button btn_confirm;

    HashSet<CoachService> servicesAdded;
    //List<PaymentItem> shoppingItemList;

    IFCMService ifcmService;

    AlertDialog dialog;

    String image_url;
```

```
    private static TotalPriceFragment instance;


    public static TotalPriceFragment getInstance( ) {
        return instance == null ? new TotalPriceFragment() : instance;
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {

        View itemView = inflater.inflate(R.layout.fragment_total_price, container, false);
        unbinder = ButterKnife.bind(this, itemView);
        init();
        initView();
        getBundle(getArguments());
        setInformation();
        return itemView;
    }

    private void setInformation() {
        txt_salon_name.setText(Common.selectedDanceHall.getName());
        txt_coach_name.setText(Common.currentCoach.getName());
        txt_coach_surname.setText(Common.currentCoach.getSurname());
        txt_time.setText(Common.convertTimeSlotToString(Common.currentLessonInformation.getSlot().intValue()));
        txt_client_name.setText(Common.currentLessonInformation.getClientName());
        txt_client_surname.setText(Common.currentLessonInformation.getClientSurname());
        txt_client_phone.setText(Common.currentLessonInformation.getClientPhone());

        if(servicesAdded.size() > 0) {

            // Add to chip group
            int i = 0;

            for(CoachService service:servicesAdded) {
                Chip chip = (Chip) getLayoutInflater().inflate(R.layout.chip_item, null);
                chip.setText(service.getName());
                chip.setTag(service);
                chip.setOnCloseIconClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        servicesAdded.remove((v.getTag()));
                        chip_group_service.removeView(v);

                        calculatePrice();
                    }
                });

                chip_group_service.addView(chip);
                i++;
            }
        }

        if(Common.currentLessonInformation.getCartItemList() != null) {

            if(Common.currentLessonInformation.getCartItemList().size() > 0) {
                MyConfirmPaymentItemAdapter adapter = new MyConfirmPaymentItemAdapter(getContext(),
Common.currentLessonInformation.getCartItemList());
                recycler_view_payment.setAdapter(adapter);
```

```
            }
            calculatePrice();
        }
    }

    private double calculatePrice() {
        double price = Common.DEFAULT_PRICE;

        for(CoachService service:servicesAdded)
            price += service.getPrice();

        if(Common.currentLessonInformation.getCartItemList() != null) {
            for(CartItem cartItem: Common.currentLessonInformation.getCartItemList())
                price += (cartItem.getProductPrice() * cartItem.getProductQuantity());
        }

        txt_total_price.setText(new StringBuilder(Common.MONEY_SIGN).append(price));

        return price;
    }

    private void getBundle(Bundle arguments) {
        this.servicesAdded = new Gson()
                .fromJson(arguments.getString(Common.SERVICES_ADDED),
                        new TypeToken<HashSet<CoachService>>(){}.getType());

        image_url = arguments.getString(Common.IMAGE_DOWNLOADABLE_URL);
    }

    private void initView() {
        recycler_view_payment.setHasFixedSize(true);
        recycler_view_payment.setLayoutManager(new LinearLayoutManager(getContext(),
LinearLayoutManager.HORIZONTAL, false));
        btn_confirm.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dialog.show();

                // Update lesson information, set done = true
                DocumentReference lessonSet = FirebaseFirestore.getInstance()
                        .collection("Places")
                        .document(Common.place_name)
                        .collection("DanceHall")
                        .document(Common.selectedDanceHall.getDanceHallId())
                        .collection("Coach")
                        .document(Common.currentCoach.getCoachId())
                        .collection(Common.simpleDateFormat.format(Common.lessonDate.getTime()))
                        .document(Common.currentLessonInformation.getLessonId());


                lessonSet
                        .get()
                        .addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                            @Override
                            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                                if(task.isSuccessful()) {
                                    if(task.getResult().exists()) {
                                        // Update
                                        Map<String,Object> dataUpdate = new HashMap<>();
                                        dataUpdate.put("done", true);
                                        lessonSet.update(dataUpdate)
                                                .addOnFailureListener(new OnFailureListener() {
```

```
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    dialog.dismiss();
                                    Toast.makeText(getContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
                                }
                            }).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful()) {
                                // If update is complete - create invoice
                                createInvoice();
                            }
                        }
                    });
                }
            }

        }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        dialog.dismiss();
        Toast.makeText(getContext(), e.getMessage(), Toast.LENGTH_SHORT).show();

    }
});




    }
});
}

private void createInvoice() {
    CollectionReference invoiceRef = FirebaseFirestore.getInstance()
            .collection("Places")
            .document(Common.place_name)
            .collection("DanceHall")
            .document(Common.selectedDanceHall.getDanceHallId())
            .collection("Invoices");

    Invoice invoice = new Invoice();
    invoice.setCoachId(Common.currentCoach.getCoachId());
    invoice.setCoachName(Common.currentCoach.getName());
    invoice.setCoachSurname(Common.currentCoach.getSurname());

    invoice.setDanceHallId(Common.selectedDanceHall.getDanceHallId());
    invoice.setDanceHallName(Common.selectedDanceHall.getName());
    invoice.setDanceHallAddress(Common.selectedDanceHall.getAddress());

    invoice.setClientName(Common.currentLessonInformation.getClientName());
    invoice.setClientSurname(Common.currentLessonInformation.getClientSurname());
    invoice.setClientPhone(Common.currentLessonInformation.getClientPhone());

    invoice.setImageUrl(image_url);

    invoice.setCoachServices(new ArrayList<CoachService>(servicesAdded));
    invoice.setPaymentItemList(Common.currentLessonInformation.getCartItemList());
    invoice.setFinalPrice(calculatePrice());
```

```
    invoiceRef.document()
            .set(invoice)
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(getContext(), e.getMessage() , Toast.LENGTH_SHORT).show();
                }
            }).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if(task.isSuccessful()) {
                sendNotificationUpdateToUser(Common.currentLessonInformation.getClientPhone());

            }
        }
    });

}

private void sendNotificationUpdateToUser(String clientPhone) {
    // Get user Token
    FirebaseFirestore.getInstance()
            .collection("Tokens")
            .whereEqualTo("user", clientPhone)
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if(task.isSuccessful() && task.getResult().size() > 0) {

                        MyToken myToken = new MyToken();
                        for(DocumentSnapshot tokenSnapshot:task.getResult()) {
                            myToken = tokenSnapshot.toObject(MyToken.class);

                            // Create notification to send
                            FCMSendData fcmSendData = new FCMSendData();
                            Map<String, String> dataSend = new HashMap<>();
                            dataSend.put("update_done", "true");

                            // Information for Rating
                            dataSend.put(Common.RATING_PLACE_KEY, Common.place_name);
                            dataSend.put(Common.RATING_DANCEHALL_ID, Common.selectedDanceHall.getDanceHallId());
                            dataSend.put(Common.RATING_DANCEHALL_NAME, Common.selectedDanceHall.getName());
                            dataSend.put(Common.RATING_COACH_ID, Common.currentCoach.getCoachId());
                            dataSend.put(Common.TOTAL_PRICE, String.valueOf(calculatePrice()));

                            fcmSendData.setTo(myToken.getToken());
                            fcmSendData.setData(dataSend);

                            ifcmService.sendNotification(fcmSendData)
                                    .subscribeOn(Schedulers.io())
                                    .observeOn(Schedulers.newThread())
                                    .subscribe(new Consumer<FCMResponse>() {
                                        @Override
                                        public void accept(FCMResponse fcmResponse) throws Exception {
                                            dialog.dismiss();
                                            dismiss();
                                            EventBus.getDefault()
                                                    .postSticky(new DismissFromBottomSheetEvent(true));
                                        }
```

```
                                }, throwable -> Toast.makeText(getContext(), throwable.getMessage(),
Toast.LENGTH_SHORT).show());
                        }
                    }
                }
            });

    }

    private void init() {
        dialog = new SpotsDialog.Builder().setContext(getContext()).setCancelable(false).build();
        ifcmService = RetrofitClient.getInstance().create(IFCMService.class);
    }
}
```

Додаток Г.

Копії акту впровадження МД

ЗАТВЕРДЖУЮ

Президент Громадської організації

«Клуб спортивного танцю «Силует»

Заслужений працівник культури України

_____ Гараніна О. Я.

«01» грудня 2020 р.

Акт

Впровадження результатів дипломного проекту

студента Сумського державного університету

Жук Олени Костянтинівни

Даним актом підтверджується, що результати роботи студента Жук О. К. на тему «Мобільний додаток підтримки діяльності Громадської організації «Клуб спортивного танцю «Силует» впроваджено в роботу Громадської організації «Клуб спортивного танцю «Силует».

Даний мобільний додаток автоматизує важливі бізнес-процеси організації діяльності танцювального клубу та забезпечує збереження даних про відвідування та оплату занять клубу.

Впровадження мобільного додатку в роботу клубу дозволило підвищити продуктивність виконання основних типів робіт, зменшити людські фактори похибок, підвищити якість діяльності Громадської організації «Клуб спортивного танцю «Силует». Розроблена база даних забезпечує цілісність та надійність збереження даних, швидку формування вибірок за різними критеріями.

Президент Громадської організації

«Клуб спортивного танцю «Силует»

Заслужений працівник культури України

Гараніна О. Я.