

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2025 р.

Зав. кафедри, к.т.н., доцент

_____ Лішук Р.І.

**КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА**

**Розробка мобільного застосунку для отримання медичних
консультацій**

Виконав - студент 4 курсу, групи 41-кн
освітньої програми «Комп'ютерні науки»
спеціальності 122 Комп'ютерні науки

Бондар Григорій Сергійович

Науковий керівник - д.т.н., професор

Кулаков Павло Ігорович

Умань - 2025

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1. Об’єкт та предмет дослідження.....	10
1.2. Порівняльний аналіз аналогів розробки	15
1.3. Постановка задачі.....	19
РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ.....	21
2.1. Структура мобільного застосунку	21
2.2. Розробка UML-діаграми мобільного застосунку	23
2.3. Обґрунтування вибору середовища розробки.....	26
РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ.....	30
3.1. Розробка модуля реєстрації та авторизації.....	30
3.2. Створення методу композиції планів тренувань.....	38
3.3. Розробка модуля створення нотаток.....	41
ВИСНОВОК.....	44
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	45
ДОДАТОК А. Лістинг програми.....	47

Вступ

У контексті стрімкого розвитку цифрових технологій мобільні застосунки набули статусу невід'ємного елемента повсякденного життя. Їхня роль є особливо значущою у сфері охорони здоров'я, де вони сприяють оперативному доступу до медичних послуг, ефективному моніторингу стану здоров'я та підвищенню рівня медичної грамотності населення. Актуальність мобільних застосунків медичного призначення постійно зростає, що обумовлено як прогресом інформаційних технологій, так і зростанням суспільного запиту на зручні та доступні медичні сервіси. Ключовою причиною широкого використання таких застосунків є їхня доступність. Завдяки поширенню смартфонів і планшетів, користувачі мають змогу здійснювати самоконтроль стану здоров'я незалежно від фізичного розташування (вдома, на робочому місці, у дорозі). Багато застосунків забезпечують функціонал для вимірювання життєво важливих показників (артеріального тиску, пульсу, рівня цукру в крові), відстеження фізичної активності (кількості кроків) та контролю спожитих калорій. Також значною популярністю користуються додатки для нагадування про медикаментозне лікування та для онлайн-запису до спеціалістів.

Особливо значущими мобільні медичні застосунки стали під час пандемії COVID-19, коли медичні заклади були перевантажені, а особисті візити пацієнтів до лікарів були обмежені. В умовах карантинних обмежень мобільні сервіси надали можливість отримувати онлайн-консультації, зберігати медичну інформацію, моніторити симптоми та своєчасно звертатися за кваліфікованою допомогою. Окрім того, такі застосунки відіграють важливу профілактичну роль у запобіганні захворюванням та формуванні здорового способу життя. Завдяки інтеграції фітнес-трекерів, програм для контролю харчування та сну, користувачі отримують інструменти для кращого розуміння свого організму та запобігання розвитку хронічних патологій.

Актуальність

Незважаючи на значну кількість існуючих мобільних застосунків у сфері охорони здоров'я, вони часто характеризуються певними недоліками: неповною відповідністю потребам користувачів, частковим функціоналом, відсутністю точної інформації або недостатньою інтеграцією механізмів персоналізації. Таким чином, розробка нових, більш досконалих мобільних додатків для підтримки здоров'я зберігає високу актуальність.

Об'єкт дослідження - процес розробки програмних засобів для реалізації мобільного застосунку в галузі охорони здоров'я.

Предмет дослідження - програмні засоби і алгоритми, необхідні для реалізації мобільного застосунку, який виконує функцію персонального консультанта у сфері охорони здоров'я.

Мета дослідження

Метою та завданням кваліфікаційної роботи бакалавра є розробка мобільного застосунку, призначеного для отримання медичних консультацій

Для досягнення зазначеної мети передбачається вирішення наступних задач:

- провести аналіз поточного стану проблеми та існуючих методів її розв'язання;
- розробити архітектуру та створити загальний алгоритм роботи застосунку;
- провести проектування модуля графічного інтерфейсу, який призначений для відображення контенту;
- розробити модуль для зберігання даних користувача;
- створити модуль для формування власних записів та управління ними;
- розробити модуль для відображення планів тренувань;
- створити модуль для вибору тренувальних планів з врахуванням індивідуальних протипоказань.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Об'єкт та предмет дослідження

В останнє десятиліття спостерігається інтенсивне інтегрування інтелектуальних пристроїв, що зумовило значний розвиток потенціалу мобільних технологій у сфері охорони здоров'я. Це призвело до виникнення поняття "мобільна охорона здоров'я" (мЗдоров'я), яке, згідно з визначенням Всесвітньої організації охорони здоров'я (ВООЗ), є "практикою медичного та громадського здоров'я, що підтримується мобільними пристроями, зокрема мобільними телефонами, пристроями спостереження за пацієнтами, персональними цифровими помічниками та іншими бездротовими пристроями".

Потенціал мобільних застосунків у охороні здоров'я дуже великий. Мобільні застосунки відкривають нові можливості для безперервного моніторингу біологічних, поведінкових та екологічних даних, показників здоров'я, а також тенденцій, пов'язаних із людською поведінкою. Вони здатні трансформувати ставлення та поведінку шляхом ефективного розподілу, збору, обробки та інтерпретації медичної інформації, а також через надання різноманітних впливів. Отже, мобільні застосунки можуть бути налаштовані для досягнення широкого спектра цілей, орієнтованих на різні групи користувачів, включаючи медичних працівників, одержувачів медичної допомоги та широку громадськість.

Використання мобільних пристроїв та застосунків є ключовим для оперативної доставки клінічної інформації до медичного персоналу, забезпечуючи актуальними даними фахівців у сільській місцевості як у розвинених, так і в країнах, що розвиваються.

Важливість орієнтації на кінцевого користувача важко переоцінити. Критично важливим для забезпечення широкого впровадження мобільних програм у сфері охорони здоров'я є глибоке розуміння та ефективне вирішення проблем кінцевих користувачів. Сприйняття таких факторів, як функціональність, продуктивність, надійність, простота використання

(наприклад, інтерфейс, час, необхідний для освоєння), а також проблеми конфіденційності, безпосередньо впливають на рівень використання мобільних застосунків як у медичній галузі, так і загалом.

Поява мобільних додатків для здоров'я обумовлена кількома ключовими детермінантами. Спостерігається зростання усвідомлення важливості профілактики та самоконтролю для підтримки здоров'я, що стимулює більшу зацікавленість населення в активній участі у збереженні власного добробуту. Стрімкий прогрес мобільних технологій перетворив смартфони на невід'ємний елемент повсякденного життя. Їхня обчислювальна потужність, портативність та постійне підключення до інтернету створюють оптимальну платформу для надання медичних послуг та інструментів самоконтролю. Зростаюче навантаження на глобальні системи охорони здоров'я вимагає розробки нових, ефективних та економічно доцільних рішень, які б розширили доступ до медичної допомоги та скоротили кількість візитів до лікаря з несуттєвих питань.

Мобільні додатки, призначені для контролю медичних показників, охоплюють широкий спектр можливостей. До найбільш поширених належать:

- відстеження життєво важливих показників: вимірювання та реєстрація динаміки пульсу, артеріального тиску, глюкози в крові, температури тіла, рівня кисню в крові (сатурації) та інших важливих параметрів.
- моніторинг фізичної активності та сну: підрахунок кроків, відстеження відстані, калорій, аналіз тривалості сну та його фаз.
- моніторинг хронічних захворювань: нагадування про необхідність прийому ліків, створення щоденника симптомів, відстеження ефективності лікувальних процедур, надання медичної інформації.
- контроль харчування і гідратації: ведення харчових щоденників, облік калорій, аналіз споживання води.
- психічне здоров'я та благополуччя: методи медитації, моніторинг настрою, нагадування про необхідність відпочинку.

Ці застосунки можуть функціонувати як автономно, використовуючи інтегровані сенсори смартфона або дані, введені користувачем, так і в комбінації з носимими пристроями (фітнес-трекерами, смарт-годинниками) та медичними приладами (глюкометрами, тонометрами), забезпечуючи таким чином більш точний та безперервний збір даних.

Використання мобільних додатків для контролю медичних показників має численні та суттєві переваги. Вони підвищують обізнаність користувачів про стан власного здоров'я. Регулярний моніторинг показників сприяє кращому розумінню організму, виявленню ранніх ознак змін та своєчасній реакції на них. Візуалізація даних у вигляді графіків та таблиць робить інформацію більш зрозумілою та мотивує до здорового способу життя. Ці додатки сприяють активному управлінню здоров'ям, наявність інструментів для самоконтролю дозволяє користувачам самостійно відстежувати ефективність дієти, фізичних вправ, режиму сну та інших факторів, що впливають на самопочуття. Нагадування про прийом ліків та обстеження допомагають дотримуватися медичних рекомендацій. Мобільні застосунки покращують взаємодію між пацієнтами та медичними працівниками, отримані дані можуть бути легко передані лікарю для аналізу стану пацієнта, коригування лікування та прийняття обґрунтованих рішень. Це особливо важливо для осіб з хронічними захворюваннями, що потребують постійного спостереження. Можливість дистанційного моніторингу сприяє скороченню кількості особистих візитів до лікаря, заощаджуючи час та ресурси. Деякі додатки можуть формувати персоналізовані рекомендації та медичну інформацію, ґрунтуючись на введених даних та результатах наукових досліджень. Це може включати поради щодо режиму харчування, фізичної активності, управління психологічним станом та інших сторін здорового способу життя. Застосування мобільних додатків призводить до зниження медичних витрат в довгостроковій перспективі за рахунок профілактики захворювань, ранньої діагностики та ефективного управління хронічними станами.

Попри численні переваги, використання мобільних додатків в галузі охорони здоров'я для контролю медичних показників пов'язане з певними викликами та обмеженнями.

Ключовим викликом є точність, достовірність та надійність даних, що отримуються за допомогою мобільних пристроїв та додатків. Не всі сенсори та алгоритми проходять клінічну валідацію, що створює ризик отримання неточних або спотворених результатів. Це може призвести до неправильної інтерпретації даних як пацієнтами, так і лікарями, та, як наслідок, до прийняття неправильних рішень щодо лікування.

Іншою проблемою є конфіденційність та безпека персональних медичних даних, отримання, зберігання та передача чутливої інформації потребують надійних механізмів захисту від несанкціонованого доступу, втрати або зловживання. Недостатня увага до безпекових питань може мати серйозні наслідки для користувачів.

Інтероперабельність та взаємодія з існуючими медичними системами також є важливим моментом, для ефективного використання даних з мобільних додатків медичними працівниками необхідна їхня сумісність з електронними медичними документами та іншими інформаційними системами клінік.

Також є ризик перевантаження інформацією користувачів та зростання їх занепокоєння щодо свого здоров'я. Безперервний моніторинг медичних показників може призвести до надмірної самодіагностики та необґрунтованих звернень до медичних установ. Необхідно утримувати баланс між інформуванням та зайвою стурбованістю.

Окрім того, цифрова нерівність може стати перешкодою для тотального впровадження мобільних медичних технологій. Далеко не всі прошарки населення мають однаковий доступ до сучасних гаджетів та інтернет-з'єднання.

Для подолання цих обставин необхідні зусилля розробників, медичних працівників, регуляторних органів та кінцевих користувачів. Важливо розробити та впровадити стандарти якості та безпеки для мобільних медичних додатків,

проводити їх клінічне випробування, забезпечувати надійний захист персональних даних та сприяти підвищенню цифрової грамотності населення.

Майбутнє мобільних застосунків для контролю медичних показників є перспективним. Спостерігається розвиток технологій штучного інтелекту та великих даних, які дозволять аналізувати великі їх обсяги, прогнозувати ризики розвитку захворювань, надавати значно більше персоналізовані рекомендації та підтримувати прийняття лікувальних рішень. Розвиток сучасних сенсорних технологій в перспективі призведе до появи принципово нових можливостей моніторингу різноманітних біомаркерів через шкіру, дихання, слину та інші показники, що зробить процес контролю абсолютно зручним та непомітним. Персоналізована медицина стане більш доступною завдяки мобільним застосункам, які враховують генетичні особливості, особливості життя та індивідуальні особливості кожного користувача при наданні рекомендацій та розробці протоколів лікування. Прогнозується зростання ролі мобільних застосунків у телемедицині шляхом забезпечення дистанційного моніторингу пацієнтів, проведення онлайн-консультацій та надання віддаленої медичної допомоги.

Таким чином, мобільні застосунки для контролю медичних показників є не просто тенденцією, а важливим інструментом, що має потенціал докорінно змінити підходи до охорони здоров'я [1]. Розвиток цих технологій, у поєднанні з відповідальним використанням та правильним регулюванням, відкриє нові можливості для персоналізованої та ефективної медицини майбутнього, де кожен пацієнт матиме можливість активно контролювати своє здоров'я, маючи надійного асистента у власному гаджеті.

Розробка програмних модулів для створення мобільного додатку в медичній галузі є важливим завданням у сучасному цифровому світі. Використання досвіду спеціалізованих компаній з розробки програмного забезпечення дозволяє створювати інноваційні рішення, що покращують користувацький досвід та забезпечують конкурентні переваги.

1.2. Порівняльний аналіз аналогів розробки

Незважаючи на стрімкий розвиток ринку мобільних застосунків для охорони здоров'я (mHealth) та електронного здоров'я (eHealth), що відображається у численних пропозиціях на комерційних платформах, таких як Google Play Store, повне та широкомасштабне впровадження цих технологій у повсякденну медичну практику та для персонального моніторингу здоров'я все ще залишається предметом очікувань. Ефективність та успішність мобільних застосунків у цій галузі значною мірою залежать від глибокого розуміння цільових категорій, функціональних можливостей, що надаються користувачам, та їхньої відповідності потребам аудиторії. Для всебічного розуміння сучасного стану та перспектив розвитку мобільних рішень у галузі охорони здоров'я доцільно провести порівняльний аналіз найбільш відомих та широко впроваджених застосунків. Серед таких варто виділити Ada Health, MyFitnessPal та Fitbit, кожен з яких представляє окремий сегмент ринку mHealth та пропонує унікальний набір функцій. Персоналізована медична консультація на базі штучного інтелекту Ada Health позиціонується як платформа для отримання медичної консультації та попереднього діагнозу на основі аналізу симптомів користувача та наданої інформації про стан здоров'я (Рис. 1.1) [2].

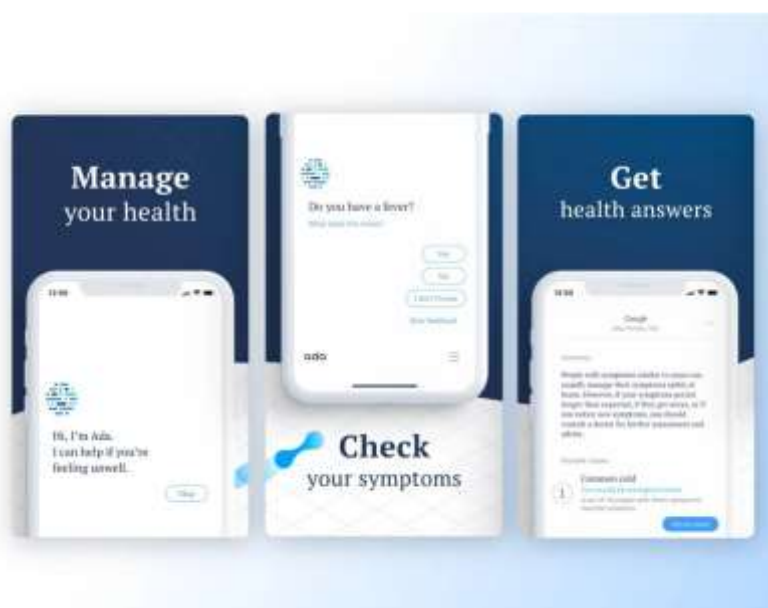


Рисунок 1.1 – Зображення мобільного застосунку «Ada Health»

Цей застосунок активно використовує передові технології штучного інтелекту (ШІ) та алгоритми машинного навчання для обробки та інтерпретації даних, що надаються користувачами. Основним завданням Ada Health є надання користувачам рекомендацій щодо можливих причин їхніх симптомів та пропозицій щодо подальших дій для полегшення стану здоров'я. Це може включати поради щодо самодопомоги, необхідності звернення до лікаря або проходження додаткових обстежень. Крім того, застосунок слугує інформаційним ресурсом, надаючи користувачам доступ до верифікованої інформації про різні захворювання, медичні процедури та варіанти лікування.

MyFitnessPal (Рис. 1.2) [3] є одним з найбільш відомих та широко використовуваних мобільних додатків та веб-сервісів, що сприяють формуванню здорового способу життя шляхом моніторингу харчування та фізичної активності.



Рисунок 1.2 – Зображення мобільного застосунку «MyFitnessPal»

Ця платформа здобула значну популярність як ефективний інструмент для управління вагою, підтримки збалансованого харчування та досягнення фітнес-цілей. Однією з ключових переваг MyFitnessPal є його велика база даних продуктів харчування, що дозволяє користувачам легко реєструвати спожиті калорії та макронутрієнти (білки, жири, вуглеводи). Застосунок також надає можливість відстежувати фізичну активність, інтегруючись з іншими фітнес-

трекерами та пристроями. Крім того, MyFitnessPal має потужну соціальну складову, що відрізняє його від багатьох інших рішень. Активна спільнота користувачів надає платформу для обміну досвідом, отримання порад, підтримки та мотивації. Це створює відчуття приналежності та сприяє стійкості у дотриманні здорового способу життя, оскільки користувачі можуть ділитися своїми досягненнями та долати виклики разом.

Fitbit App є невід'ємною частиною екосистеми Fitbit, що дозволяє користувачам синхронізувати дані, зібрані їхніми пристроями Fitbit (смарт-годинниками, фітнес-браслетами), зі смартфонами або планшетами. Цей застосунок розширює функціональні можливості фізичних пристроїв, надаючи деталізовану аналітику та інтерпретацію зібраних даних (Рис. 1.3) [4].

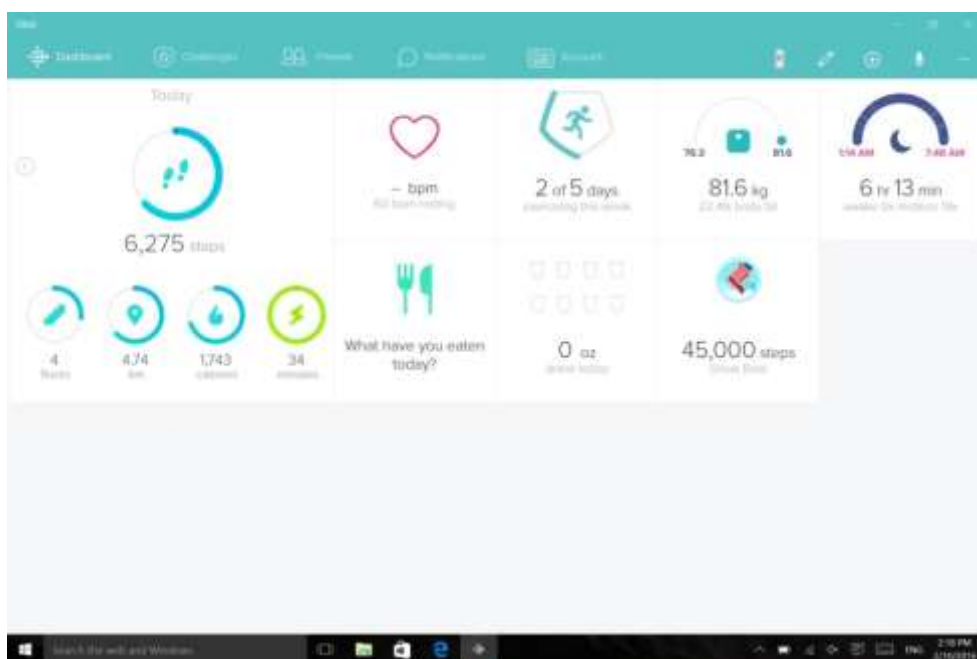


Рисунок 1.3 – Зображення застосунку «Fitbit»

Fitbit App дозволяє користувачам відстежувати широкий спектр показників, що стосуються їхньої фізичної активності та загального стану здоров'я. Крім базового моніторингу, застосунок пропонує низку додаткових функцій, що сприяють досягненню фітнес-цілей. Це включає створення індивідуальних цілей фізичної активності, моніторинг харчування, можливість конкурувати з друзями або членами родини у фітнес-викликах, а також

отримання персоналізованих рекомендацій щодо покращення здоров'я та фізичної форми на основі зібраних даних.

Результати порівняння аналогів наведено в таблиці 1.1. На основі результатів порівняння власної розробки з іншими мобільними застосунками у галузі охорони здоров'я можна зробити висновок, що власна розробка є кращим рішенням у цій галузі завдяки своїм різноманітним функціям та користувацькому досвіду.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

	AdaHealth	MyFitnessPal	Fitbit	Власна розробка
Персоналізовані консультації вправ	-	-	+	+
Доступ до даних через віджети робочого столу	-	-	+	+
Конфіденційність та безпека	+	-	-	+
Ведення історії	-	+	+	+
Створення нотаток	-	+	+	+
	2	2	3	5

Аналіз демонструє, що пропонована розробка позиціонується як комплексне та мультифункціональне рішення у галузі охорони здоров'я, що дозволяє користувачам моніторити широкий спектр показників, включаючи фізичну активність, харчування та загальний стан здоров'я. На відміну від Ada Health, яка зосереджена переважно на діагностиці симптомів, або MyFitnessPal та Fitbit, які акцентують увагу на конкретних аспектах способу життя (харчування, фізична активність), власна розробка прагне забезпечити цілісний підхід до управління здоров'ям.

Під час розробки мобільних застосунків інженерний процес передбачає вибір оптимальних методів та архітектурних підходів, що відповідають

специфічним потребам проекту, вимогам замовника, доступним ресурсам та цільовій аудиторії. Нативна розробка полягає у створенні окремих застосунків для кожної цільової мобільної платформи, тобто Android та iOS [5]. Кожна з цих платформ має власну екосистему, набір інструментів розробки (SDK), мови програмування (наприклад, Kotlin/Java для Android, Swift/Objective-C для iOS) та гайдлайни щодо дизайну інтерфейсу користувача (UI) та користувацького досвіду (UX). Незважаючи на те, що нативна розробка забезпечує найкращий UX, який відповідає актуальним гайдлайнам кожної з платформ, її основним недоліком є необхідність писати код програми двічі – окремо під Android та окремо під iOS [6]. Це збільшує час розробки, витрати на персонал та ускладнює підтримку двох окремих кодових баз. Гібридна розробка є компромісним підходом, що поєднує елементи нативних та веб-технологій [7]. Застосунки, розроблені за гібридним принципом, використовують веб-технології (HTML, CSS, JavaScript) для побудови інтерфейсу та логіки, які потім "обгортаються" у нативну оболонку за допомогою фреймворків, таких як Apache Cordova або Ionic. Цей підхід дозволяє розробникам створювати додатки зі схожим зовнішнім виглядом, відчуттям і продуктивністю, як традиційні нативні додатки, одночасно використовуючи переваги економії коштів і швидкості веб-розробки.

Для розробки власного мобільного застосунку було прийнято рішення використовувати нативну розробку. Це обґрунтовано низкою критичних факторів, що мають першочергове значення для забезпечення високої якості, функціональності та конкурентоспроможності програмного продукту у сфері охорони здоров'я.

1.3. Постановка задачі

Метою кваліфікаційної роботи бакалавра є розробка мобільного застосунку для отримання медичних консультацій. Розроблений мобільний застосунок повинен бути зручним і практичним рішенням, який забезпечує персональне медичне консультування та надання рекомендацій користувачу

щодо тренувань, залежно від вказаних протипоказань, повинен мати розвинуту базу даних, яка зможе зберігати усі потрібні ресурси.

Завданням розробки є аналіз предметної області та проведення порівняльного аналізу аналогічних додатків медичного призначення, обґрунтування вибору технологій для розробки мобільного застосунку, розробка загальної архітектури мобільного застосунку, розробка мобільного застосунку. Для вирішення поставлених завдань необхідно:

- провести аналіз поточного стану проблеми та існуючих методів її розв'язання;
- розробити архітектуру та створити загальний алгоритм роботи застосунку;
- провести проектування модуля графічного інтерфейсу, який призначений для відображення контенту;
- розробити модуль для зберігання даних користувача;
- створити модуль для формування власних записів та управління ними;
- розробити модуль для відображення планів тренувань;
- створити модуль для вибору тренувальних планів з врахуванням індивідуальних протипоказань.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Структура мобільного застосунку

Kotlin є сучасною, статично типізованою мовою програмування, що працює на віртуальній машині Java (JVM), яка набула значної популярності як альтернатива традиційній Java для розробки мобільних додатків на платформі Android. Kotlin забезпечує спрощення розробки додатків з інтенсивною взаємодією з мережею або базою даних, забезпечуючи плавну роботу інтерфейсу користувача без блокування основного потоку. Ці переваги роблять Kotlin оптимальним вибором для створення високоефективних мобільних застосунків.

Android Studio є офіційним інтегрованим середовищем розробки (IDE) компанії Google, розроблений спеціально для створення програмного забезпечення для платформи Android. Включає в себе редактор коду з підтримкою Kotlin, Java та C++, інструменти для налагодження, вбудований емулятор для тестування на різних пристроях та версіях Android, а також інтеграцію з системою автоматичної збірки Gradle. Така функціональність дозволяє оптимізувати робочий процес та підвищити ефективність розробки.

Розробка мобільного застосунку для медичних консультацій є комплексним процесом, що складається з послідовних і взаємопов'язаних етапів. Початковим і одним з найважливіших етапів створення програмного засобу є проектування та реалізація інтерфейсу користувача (UI). Інтерфейс є першою точкою контакту користувача із застосунком, формуючи перше враження та впливаючи на загальний користувацький досвід (UX). З огляду на це, інтерфейс повинен бути не лише візуально привабливим, але і інтуїтивно зрозумілим.

Наступним критично важливим кроком є побудова графу навігації. Граф навігації є фундаментальним елементом архітектури мобільного застосунку, що визначає логічні зв'язки та переходи між різними екранами користувацького інтерфейсу. Він не лише створює схеми послідовності взаємодії, але й забезпечує коректне відображення контенту при переході між частинами застосунку.

Правильно спроектований граф навігації гарантує плавність користувацького потоку та запобігає заплутаним сценаріям.

Третім етапом є розробка модуля для реєстрації та авторизації. Цей модуль є основою для забезпечення конфіденційності користувацьких даних у медичному застосунку. Він передбачає встановлення зв'язку з базою даних для зберігання облікових записів, аутентифікації користувачів та управління їхніми повноваженнями. Основна мета полягає у забезпеченні доступу до персональних даних лише авторизованим користувачам. Надійні механізми шифрування та контролю доступу є обов'язковими для захисту медичної інформації.

Протягом четвертого етапу буде розроблено модуль для ведення нотаток щодо тренувань або зауважень стосовно здоров'я. Цей функціонал дозволить користувачам фіксувати важливу інформацію, пов'язану з їх фізичним станом. Створені записи будуть зберігатися у базі даних, забезпечуючи їхню конфіденційність та можливість синхронізації між різними пристроями. Це дозволить користувачам систематично відстежувати медичні показники.

Завершальним етапом розробки є створення модуля планів тренувань різних видів та категорій. Цей модуль надає користувачам гнучкі можливості для вибору вправ, які найкраще відповідають їхнім потребам та вподобанням.

Отже, розробка програмних засобів мобільного застосунку для отримання медичних консультацій, заснована на Kotlin та Android Studio, є комплексним завданням, що охоплює декілька послідовних етапів.

Одним із ключових аспектів проектування мобільного застосунку є вибір його архітектури [8]. В сучасній Android-розробці значну популярність здобула архітектура MVVM (Model-View-ViewModel) завдяки своїй здатності ефективно відокремлювати бізнес-логіку від інтерфейсу користувача [9]. Архітектура MVVM складається з трьох основних компонентів:

- Модель (Model): Цей компонент є представленням даних та бізнес-логіки застосунку. Модель відповідає за взаємодію з джерелами даних (наприклад, базами даних), збереження, оновлення та видалення інформації.

- Вид (View): Вид є безпосереднім інтерфейсом користувача розроблюваного мобільного застосунку. Його основна функція полягає у візуальному відображенні даних, наданих ViewModel, та реагуванні на дії користувача, такі як натискання кнопок або введення тексту.

- ViewModel: ViewModel виступає як посередник між Моделлю та Видом, відповідає за збереження та обробку даних, які відображаються на екрані. ViewModel отримує дані від Моделі, трансформує їх у потрібний формат, та надає їх для відображення.

Переваги використання MVVM на Android включають значне полегшення тестування, оскільки ViewModel може бути протестована ізольовано від View. Схема архітектури MVVM наведена на рис. 2.1.

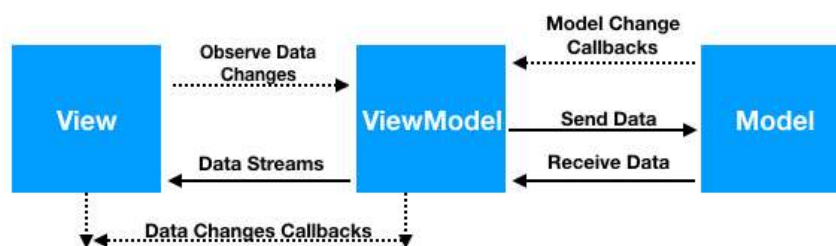


Рисунок 2.1 - Схема архітектури MVVM

2.2. Розробка UML-діаграми мобільного застосунку

При проектуванні та розробці складних програмних систем, таких як мобільні застосунки, критично важливо мати глибоке та чітке розуміння їхньої повної функціональності та взаємодії між окремими компонентами. Це передбачає ретельне визначення всіх робочих процесів, потоків даних та механізмів комунікації між різними частинами програмного забезпечення. Ефективним підходом для досягнення цієї мети є створення деталізованих алгоритмів, які відображають загальний принцип роботи системи та послідовність виконуваних дій. Одним з найбільш ефективних інструментів для візуалізації та моделювання функціональності програмних систем є уніфікована

мова моделювання (UML), зокрема, діаграма діяльності є потужним засобом для демонстрації загальної роботи застосунку, не вдаючись до надмірної деталізації внутрішніх робочих процесів. Такий підхід значно полегшує сприйняття та розуміння комплексної логіки системи для всіх зацікавлених сторін, включаючи розробників, аналітиків та замовників. На рисунку 2.2 представлена UML-діаграма, що ілюструє загальну архітектуру та послідовність основних операцій мобільного застосунку.

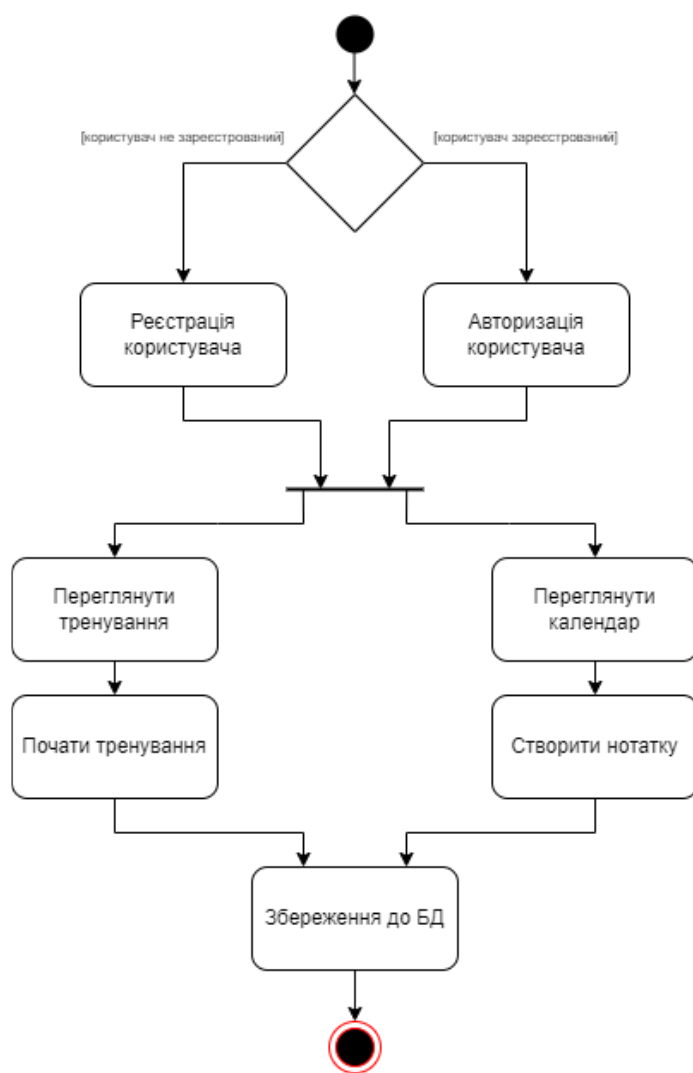


Рисунок 2.2 - UML-діаграма роботи мобільного застосунку

Серед ключових функціональних можливостей мобільного застосунку виділяється модуль, відповідальний за проведення персоналізованих тренувань. Було розроблено детальний алгоритм, що визначає послідовність дій користувача та реакцію системи. Цей алгоритм зображено на рисунку 2.3.

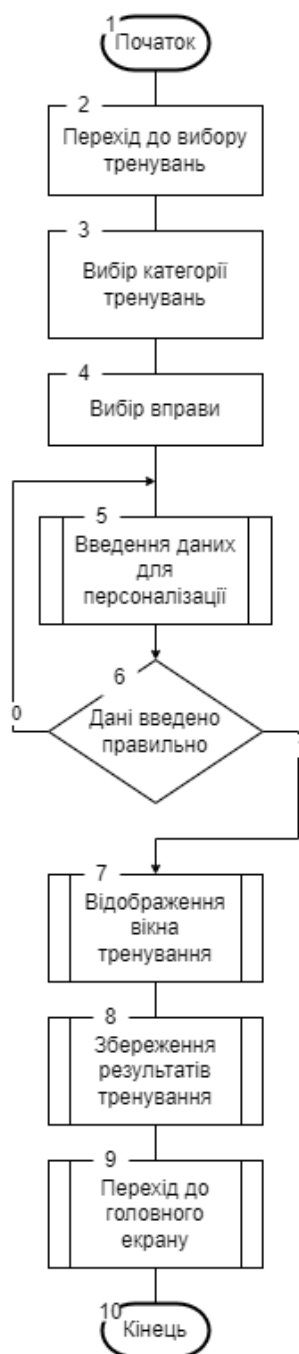


Рисунок 2.3 - Екран проведення тренувань

Спочатку користувач здійснює взаємодію з основним інтерфейсом мобільного застосунку з метою переходу на екран тренувань. Це може бути реалізовано через відповідну кнопку, пункт меню або інший елемент навігації. Після успішного переходу на екран тренувань, користувачу пропонується обрати бажану категорію тренувань. З представленого списку вправ, що належать до обраної категорії, користувач обирає конкретну вправу для виконання. Після

вибору вправи на екрані мобільного пристрою з'являється спеціалізоване діалогове вікно для персоналізації параметрів тренування. Користувачеві надається можливість ввести бажану кількість підходів та кількість повторень для обраної вправи. Цей крок дозволяє адаптувати тренування до індивідуальних потреб користувача. Після підтвердження введених параметрів натисканням кнопки «Продовжити», користувачеві відображається інтерактивне вікно. Це вікно демонструє анімацію, що ілюструє правильну техніку виконання обраної вправи, забезпечуючи чіткі візуальні інструкції. По завершенні виконання вправи всі введені дані для персоналізації та витрачений час автоматично зберігаються в історії виконання вправ. Ця інформація є важливою для аналізу ефективності тренувань. Після успішного збереження даних, користувач автоматично перенаправляється до головного екрану застосунку, що забезпечує безперервність користувацького досвіду. Такий деталізований алгоритмічний опис дозволяє не тільки уявити функціональність модуля проведення тренувань, але й слугує основою для його ефективної програмної реалізації, забезпечуючи відповідність між проектною та фактичною поведінкою системи.

2.3. Обґрунтування вибору середовища розробки

Прийняття обґрунтованих рішень щодо вибору інтегрованого середовища розробки (IDE) та мови програмування є одним з фундаментальних кроків на початкових етапах життєвого циклу розробки програмного забезпечення. Інтегровані середовища розробки (IDE) є комплексними програмними системами, які об'єднують усі необхідні інструменти для створення, тестування та розгортання програмного забезпечення в єдиному середовищі [10].

Android Studio є офіційним та рекомендованим середовищем розробки для платформи Android, розробленим компанією Google [11]. Його архітектура базується на потужній платформі IntelliJ IDEA, що забезпечує інтуїтивно зрозумілий інтерфейс та широкий спектр інструментів, спеціально адаптованих для екосистеми Android. Android Studio надає розширену підтримку для Kotlin,

Java та C++, що робить його універсальним інструментом для розробників. Eclipse історично був одним із популярних інтегрованих середовищ розробки, особливо серед Java-розробників [12]. Eclipse забезпечував підтримку Java та C++, мав емулятор та інструменти для налагодження. Після появи Android Studio, Google припинив активну підтримку ADT для Eclipse, що зробило його менш актуальним вибором для сучасних Android-проектів через відсутність оновлень.

IntelliJ IDEA є ще одним потужним IDE, розробленим компанією JetBrains, яка також є автором мови Kotlin [13]. Фактично, Android Studio є спеціалізованою версією IntelliJ IDEA. Це середовище підтримує широкий спектр мов програмування, включаючи Java, Kotlin та інші, і також інтегрує вбудований емулятор для тестування та підтримує автоматизацію збірки за допомогою Gradle [14]. Порівняльний аналіз на основі можливостей згаданих IDE наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння середовищ розробки

Особливість	Android Studio	Eclipse	IntelliJ IDEA
Офіційно підтримується Google	+	–	–
Система збірки на основі Gradle	+	–	+
Вбудований емулятор	+	+	+
Інтеграція з Git	+	+	+
Інструменти профілювання	+	–	+
Миттєвий запуск для тестування	+	–	+
Екосистема плагінів	+	+	+
Підсумок	7	3	6

На основі порівняльного аналізу можна стверджувати, що Android Studio є оптимальним вибором для запланованої розробки мобільного застосунку. Його

офіційна підтримка з боку Google гарантує постійні оновлення. Використання системи збірки Gradle забезпечує гнучкість та простоту управління проєктом. Наявність емулятора значно прискорює процес тестування та налагодження, що є критично важливим для забезпечення якості програмного продукту.

Вибір мови програмування є ще одним фундаментальним рішенням, яке визначає архітектуру, продуктивність, підтримуваність та масштабованість мобільного застосунку. Для розробки додатків на платформі Android, основними варіантами є Kotlin, Java та C++. Kotlin є сучасною, статично типізованою мовою програмування, що функціонує на віртуальній машині Java (JVM), розробленою компанією JetBrains [15]. Kotlin комбінує функціональні та об'єктно-орієнтовані підходи до програмування, пропонуючи низку інноваційних функцій, які спрощують розробку. Kotlin дозволяє писати значно менше коду порівняно з Java для досягнення тієї ж функціональності, що прискорює розробку та зменшує ймовірність помилок. Kotlin є повністю сумісним з Java, що дозволяє використовувати всі існуючі Java-бібліотеки та фреймворки, а також поступово мігрувати існуючі Java-проєкти на Kotlin.

Java була основною мовою програмування для розробки Android-додатків протягом багатьох років і досі залишається популярним вибором серед розробників [16, 17]. Java має одну з найбільших спільнот розробників у світі, що забезпечує доступ до величезної кількості документації, навчальних ресурсів, бібліотек та готових рішень. Мова є стабільною, добре протестованою та широко використовуваною у промисловості. Java підтримується Android Studio та дозволяє створювати додатки для різних типів пристроїв Android. Однак, порівняно з Kotlin, Java може бути більш багатослівною, а її підходи до асинхронності можуть бути складнішими для реалізації.

C++ може бути використаний для розробки Android-додатків за допомогою Android NDK (Native Development Kit) [18-20]. Код C++ компілюється безпосередньо в нативні бібліотеки, що забезпечує найвищий рівень продуктивності. Дозволяє інтегрувати існуючі C++ бібліотеки та алгоритми.

Проте, використання C++ значно ускладнює процес розробки та вимагає ретельного контролю над пам'яттю та безпекою, що може призвести до проблем з витоками пам'яті або збоями застосунку. Порівняльний аналіз мов програмування наведено в таблиці 3.2.

Таблиця 3.2 - Порівняння мов програмування

Особливість	Java	Kotlin	C++
Компільована мова	+	+	+
Збір сміття	+	+	–
Кросплатформеність	+	+	–
Приведення типів	–	+	+
Функції розширення	–	+	–
Null-безпека	–	+	–
Перевантаження операторів	–	+	+
Контроль ресурсів пристрою	–	–	+
Підсумок	3	7	4

З огляду на цілі та вимоги до розробки мобільного застосунку для медичних консультацій, Kotlin виявляється найбільш перспективним та раціональним вибором. Його лаконічний синтаксис та краща читабельність сприяють швидшому та ефективнішому процесу розробки. Вбудована null-безпека значно зменшує ймовірність збоїв, підвищуючи стабільність та надійність застосунку. Додатково, розширена підтримка функціонального програмування та корутин робить код більш підтримуваним та адаптованим до сучасних архітектурних патернів. Ці фактори роблять Kotlin оптимальним рішенням для створення сучасного, надійного та ефективного медичного застосунку.

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1. Розробка модуля реєстрації та авторизації

Для ініціації розробки модуля реєстрації та авторизації користувачів необхідно створити та ініціалізувати зв'язок мобільного застосунку з базою даних. З цією метою було розроблено клас `FirebaseFirestoreService` (рис. 3.1).



Рисунок 3.1 - Діаграма класів підключення до бази даних

Застосунок встановлює початкове підключення до бази даних та здійснює валідацію користувача в системі за допомогою методу `createProfileDetail()`, використовуючи параметр `dataMap`. Блок-схема алгоритму цього методу представлена на рис. 3.2, а пояснення до алгоритму наведені нижче:

- виклик методу `createProfileDetail()`.
- як аргументи методу передаються змінні `dataMap` та `callback`.
- змінна класу `firestore` створюється та ініціюється шляхом виклику вбудованого методу `getInstance()` класу `FirebaseFirestore`.
- за допомогою методу `firestore.collection("profileDetail")` отримуються дані профілів з колекції "Profiles" з бази даних.
- шляхом виклику методу `firestore.document(dataMap["email"])` отримуються всі записи адресів електронних пошт.
- метод `firestore.set(dataMap)` присвоює змінній `firestore` значення, що містить усі дані про користувача, пошта якого була вказана у змінній `dataMap`.

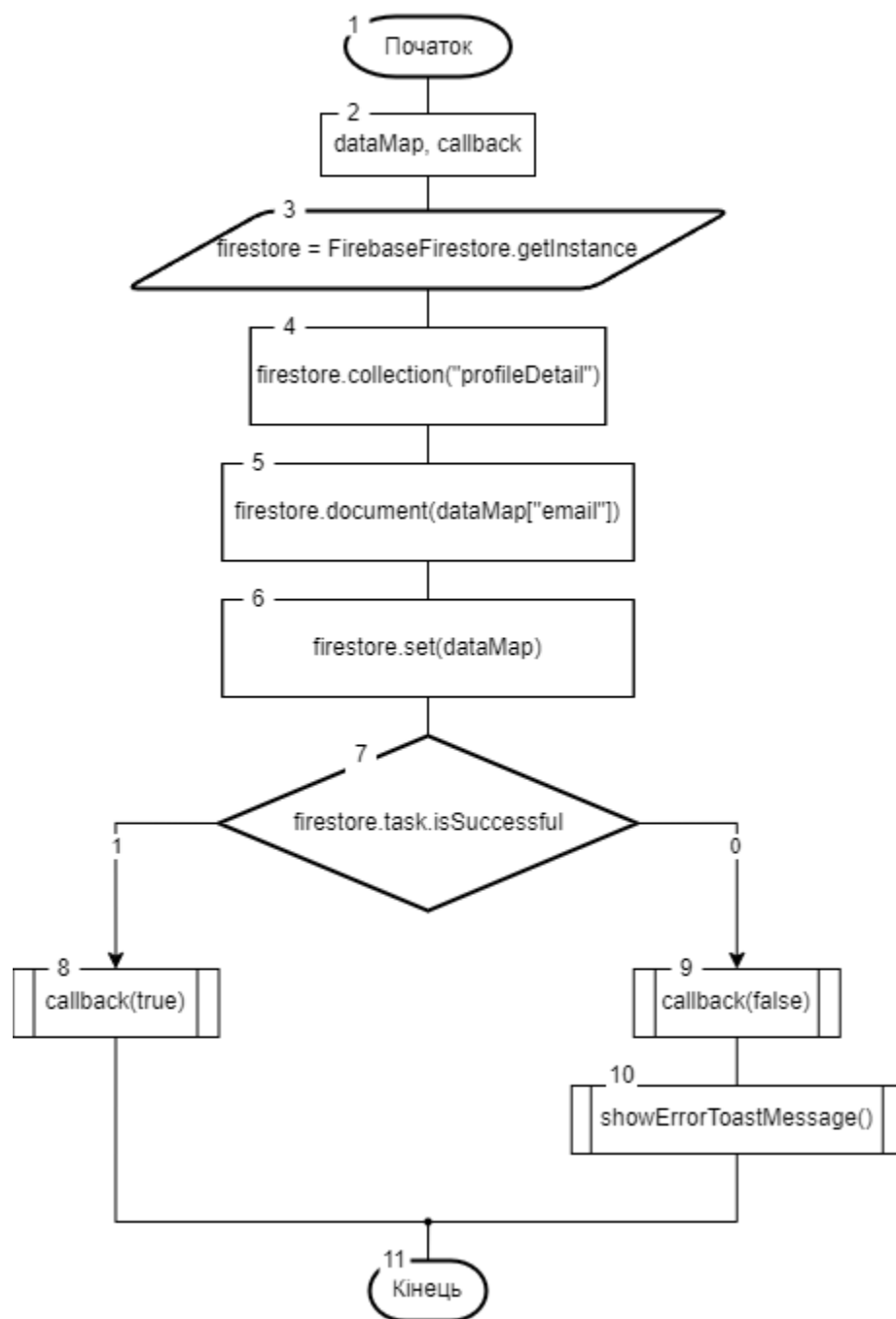


Рисунок 3.2 - Блок-схема алгоритму методу `createProfileDetail`

- за допомогою вбудованої властивості `task` та її властивості `isSuccessful` перевіряється цілісність отриманих даних, повертаючи булеве значення "true" або "false".

- якщо було повернуто значення "true", викликається вбудований метод `callback()`, якому передається значення "true" для завершення з'єднання з базою даних та збереження отриманих даних у змінній `firestore`.

- якщо було повернуто значення "false", викликається `callback()`, якому передається значення "false" для завершення з'єднання з базою даних без збереження отриманих даних у змінній `firestore`.

- викликається вбудований метод `showErrorToastMessage()`, який інформує про виникнення помилки та відображає сповіщення типу Toast, інтегроване в систему Android.

- вихід з методу `createProfileDetail()`.

Для надсилання та збереження оновлених даних використовується метод `updateProfileDetail()`. Блок-схему алгоритму методу наведено на рисунку 3.3, а пояснення до алгоритму наступні:

- виклик методу `updateProfileDetail()`.

- як аргументи методу передаються змінні `email` (що містить електронну пошту користувача), `updateData` (що містить дані для оновлення) та `callback` (змінна, яка повертатиме значення "true" або "false" для завершення операції з'єднання).

- змінна класу `firestore` створюється та ініціюється шляхом виклику вбудованого методу `getInstance()` класу `FirebaseFirestore`.

- за допомогою методу `firestore.collection("profileDetail")` отримуються дані профілів з колекції "Profiles" з бази даних.

- шляхом виклику методу `firestore.document(email)` отримуються записи користувача з вказаною електронною поштою.

- метод `firestore.update(updateData)` присвоює змінній `firestore` значення, що містить оновлені дані користувача.

- за допомогою вбудованої властивості `task` та її властивості `isSuccessful` перевіряється цілісність отриманих даних, повертаючи булеве значення "true" або "false".

- якщо було повернуто значення "true", викликається метод `callback()`, якому передається значення "true" для завершення з'єднання з базою даних та збереження отриманих даних у змінній `firestore`.

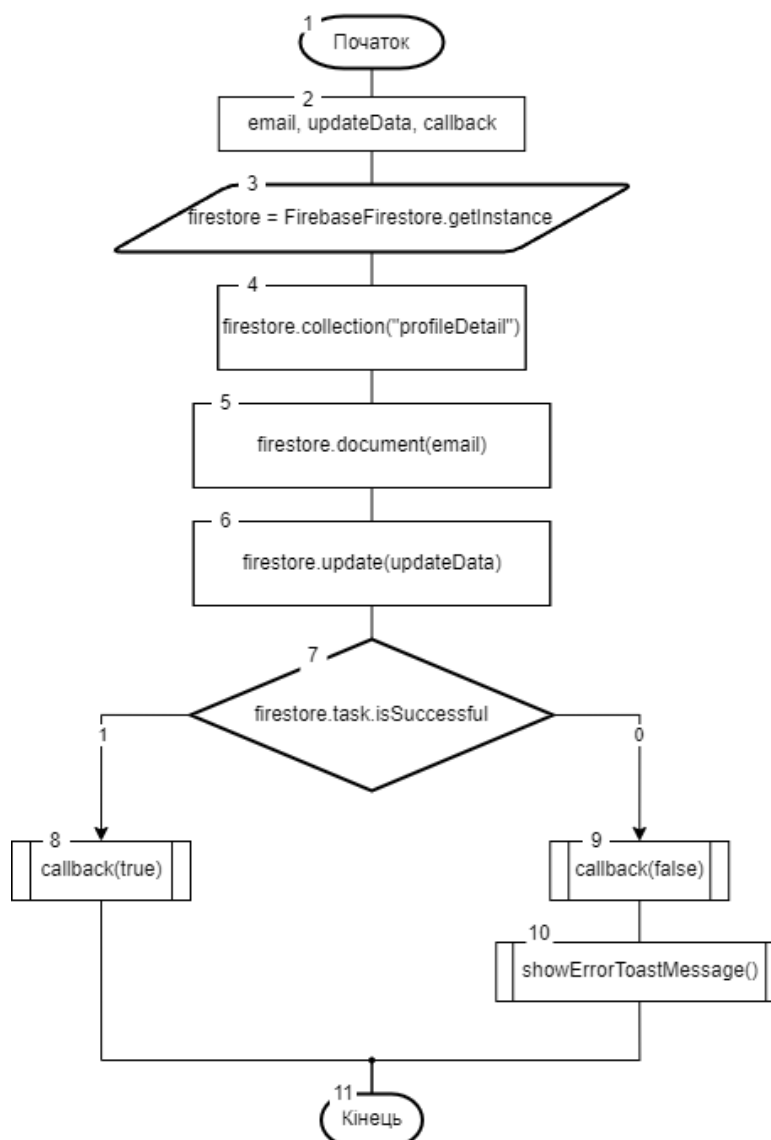


Рисунок 3.3 - Блок-схема алгоритму методу `updateProfileDetails`

- якщо повернуто значення "false", викликається `callback()`, якому передається значення "false" для завершення з'єднання з базою даних без збереження отриманих даних у змінній `firestore`.

- викликається вбудований метод `showErrorMessage()`, який інформує про виникнення помилки та відображає сповіщення типу Toast, інтегроване в систему Android.

- вихід з методу `updateProfileDetail()`.

Використовуючи розроблений ініціалізатор бази даних, необхідно створити клас, який забезпечуватиме функціональність авторизації (входу) та

реєстрації (створення нового облікового запису). З цією метою було розроблено клас FirebaseAuthService (рис. 3.4).

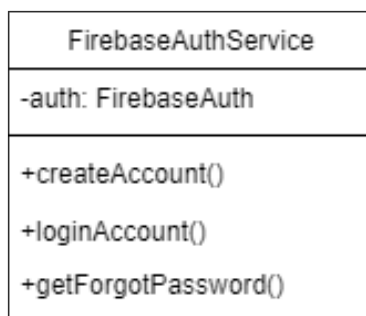


Рисунок 3.4 - Діаграма класів класу FirebaseAuthService

Для здійснення авторизації користувача в мобільному застосунку використовується метод loginAccount() класу FirebaseAuthService. Цей метод приймає аргументи "електронна пошта" та "пароль", після чого за допомогою об'єкта auth звіряє введені дані з даними, що зберігаються в базі даних. У разі виявлення відповідних записів, метод повертає булеве значення "true". Блок-схема алгоритму цього методу представлена на рисунку 3.5, а пояснення до алгоритму наступні:

- виклик методу loginAccount().
- як аргументи методу передаються змінні email (що містить електронну пошту користувача), password (що містить пароль до облікового запису користувача) та callback (змінна, яка повертатиме значення "true" або "false" для завершення операції автентифікації).
- змінна класу auth ініціюється шляхом виклику вбудованого методу getInstance() класу FirebaseAuth для підключення до бази даних.
- до змінної класу auth викликається вбудований метод signInWithEmailAndPassword(), який приймає аргументи email (електронна пошта облікового запису користувача) та password (пароль до цього облікового запису).
- за допомогою вбудованої властивості it.isSuccessful перевіряється успішність автентифікації та повертається булеве значення.

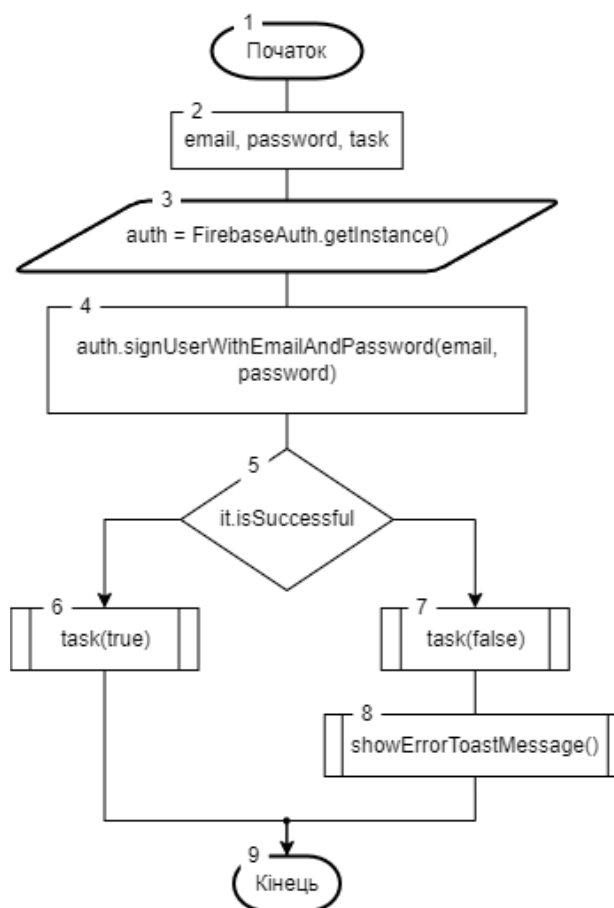


Рисунок 3.5 - Блок-схема алгоритму методу loginAccount

- якщо автентифікація пройшла успішно, змінній task передається значення "true", що завершує з'єднання з базою даних та завантажує всі дані облікового запису користувача.

- якщо автентифікація не вдалася, змінній task передається значення "false", що завершує з'єднання з базою даних без отримання даних облікового запису.

- викликається вбудований метод showMessage(), який інформує про виникнення помилки та відображає сповіщення типу Toast.

- вихід з методу loginAccount().

Для здійснення реєстрації використовується метод createAccount(), якому також передаються аргументи "електронна пошта" та "пароль". На відміну від попереднього методу, цей метод повертає булеве значення "true", якщо дані були успішно збережені в базі даних. Блок-схема алгоритму цього методу представлена на рис. 3.6.

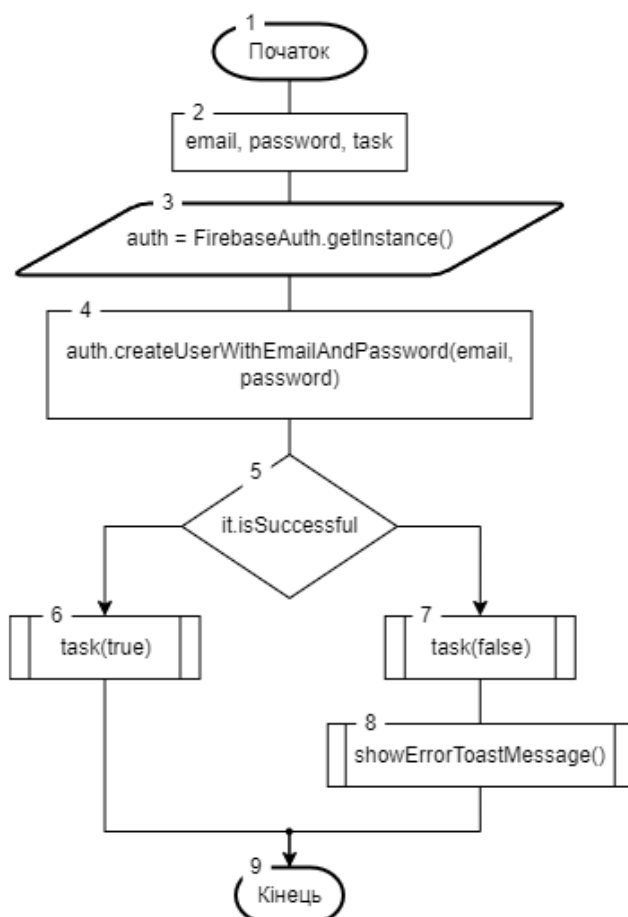


Рисунок 3.6 - Блок-схема алгоритму методу createAccount

Пояснення до алгоритму наступні:

- виклик методу createAccount().
- як аргументи методу передаються змінні email (що містить електронну пошту, за допомогою якої буде створено обліковий запис користувача), password (що містить пароль до облікового запису користувача) та callback (змінна, яка повертатиме значення "true" або "false" для завершення операції реєстрації).
- змінна класу auth ініціюється шляхом виклику вбудованого методу getInstance() класу FirebaseAuth для підключення до бази даних.
- до змінної класу auth викликається вбудований метод createUserWithEmailAndPassword(), який приймає аргументи email (електронна пошта облікового запису користувача) та password (пароль до цього облікового запису).

- за допомогою вбудованої властивості `it.isSuccessful` перевіряється успішність реєстрації та повертається булеве значення.

- якщо реєстрація пройшла успішно, змінній `task` передається значення `"true"`, що завершує з'єднання з базою даних та зберігає обліковий.

- якщо реєстрація не вдалася, змінній `task` передається значення `"false"`, що завершує з'єднання з базою даних без збереження облікового запису в базі даних.

- викликається вбудований метод `showErrorToastMessage()`, який інформує про виникнення помилки та відображає сповіщення типу `Toast`, інтегроване в систему `Android`.

- вихід з методу `createAccount()`.

Нижче наведено фрагмент лістингу розробленого програмного забезпечення:

```
package com.app.ebfitapp.service
import android.app.Dialog
import android.widget.Toast
class FirebaseAuthService(private val context: Context) {
    private var auth: FirebaseAuth = FirebaseAuth.getInstance()
    fun createAccount(email: String, password: String, task: (Boolean) ->
    Unit) {
        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener {
                if (it.isSuccessful) task(true)
            }.addOnFailureListener {
                task(false)
                showErrorToastMessage(it.message.toString())
            }
    }
    fun loginAccount( email: String,
        password : String,
        task: (Boolean) -> Unit) {
        auth.signInWithEmailAndPassword(email,password)
```

```

if (it.isSuccessful) task(true)
else task(false)
}.addOnFailureListener() {
task(false)
showErrorToastMessage(it.message.toString())
}
}
fun getForgotPassowrd(sPassword : String , dialog: Dialog,
customProgress: CustomProgress) {
.addOnSuccessListener {
showErrorToastMessage("Please Check Your Email Address")
}.addOnFailureListener {
customProgress.dismiss()
showErrorToastMessage(it.message.toString())
}
}
fun signOut() {
auth.signOut()
}
fun getCurrentUserEmail(): String {
return auth.currentUser?.email.toString()
}
private fun showErrorToastMessage(error: String) {
Toast.makeText(context, error, Toast.LENGTH_LONG).show()
}
}

```

3.2. Створення методу композиції планів тренувань

Першим етапом у створенні модуля відображення планів тренувань є розробка методу `deleteRecordsWithContraindications()`. Його функція полягає в отриманні всіх записів про тренування та видаленні тих, що містять протипоказання, з відповідної колекції. Блок-схема алгоритму цього методу представлена на рис. 3.7.

Пояснення до алгоритму наступні:

- виклик методу `deleteRecordsWithContraindications()`.
- в якості аргументів передаються `firestore` – екземпляр `Firestore`, та `contraindications` – список протипоказань.
- Ініціюється змінна `collectionRef`, яка має посилання на колекцію «`muscleGroups`».

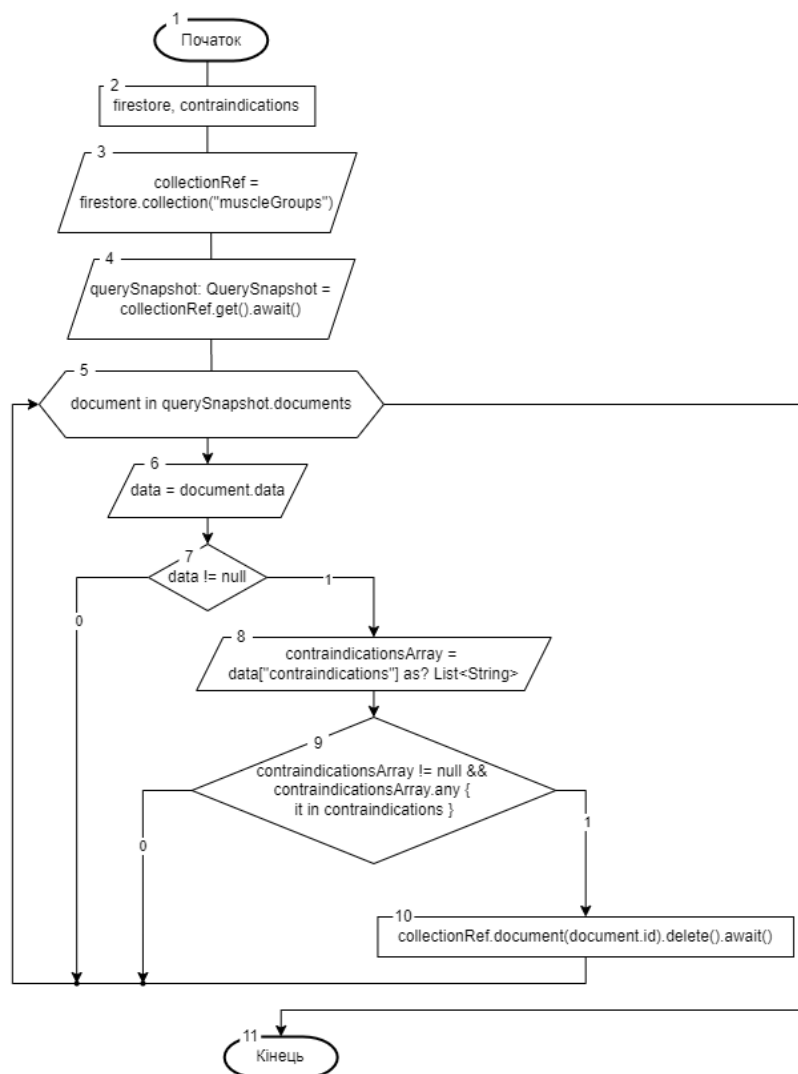


Рисунок 3.7 - Блок-схема алгоритму методу `deleteRecordsWithContraindications`

- ініціюється змінна `querySnapshot`, яка містить записи колекції тренувань, отримані шляхом виклику асинхронного методу `get()` змінної `collectionRef` із застосуванням `await()`.
- цикл ітерує по кожному елементу результату запиту.

- ініціюється змінна `data`, в яку записуються дані елементу у форматі пар «ключ - значення».
- умова перевіряє, чи не є елемент запиту порожнім.
- ініціюється змінна `contraindicationsArray`, в яку записується значення поля «`contraindications`» з даних елемента та конвертується у список рядків.
- умова перевіряє, чи масив протипоказань не є пустим і чи містить він будь-який елемент зі списку `contraindications`.
- асинхронно видаляється елемент за його ідентифікатором з колекції «`muscleGroups`».
- вихід із методу `deleteRecordsWithContraindications()`.

Для отримання тренувань, які будуть відображатися на екрані користувача, було розроблено метод `getMuscleGroups()`. Блок-схема алгоритму роботи методу представлена на рисунку 3.8.

Пояснення до алгоритму наступні:

- виклик методу `getMuscleGroups()`.
- ініціюється змінна `muscleTempList` як об'єкт `ArrayList`.
- умова перевіряє, чи значення `muscleGroupLiveData` є `null`. Якщо так, це вказує на відсутність завантажених даних, що дозволяє продовжити процес отримання даних з `Firestore`.
- встановлюється з'єднання з базою даних та зберігається екземпляр для підключення у змінну `firestoreService`.
- викликається метод для отримання всіх елементів з колекції «`muscleGroups`».
- викликається метод для отримання всіх елементів з колекції «`contraindication`».
- перевіряється успішність виконання запиту за допомогою вбудованої властивості `task.isSuccessful`.
- здійснюється ітерація по кожному елементу, отриманому в результаті запиту.

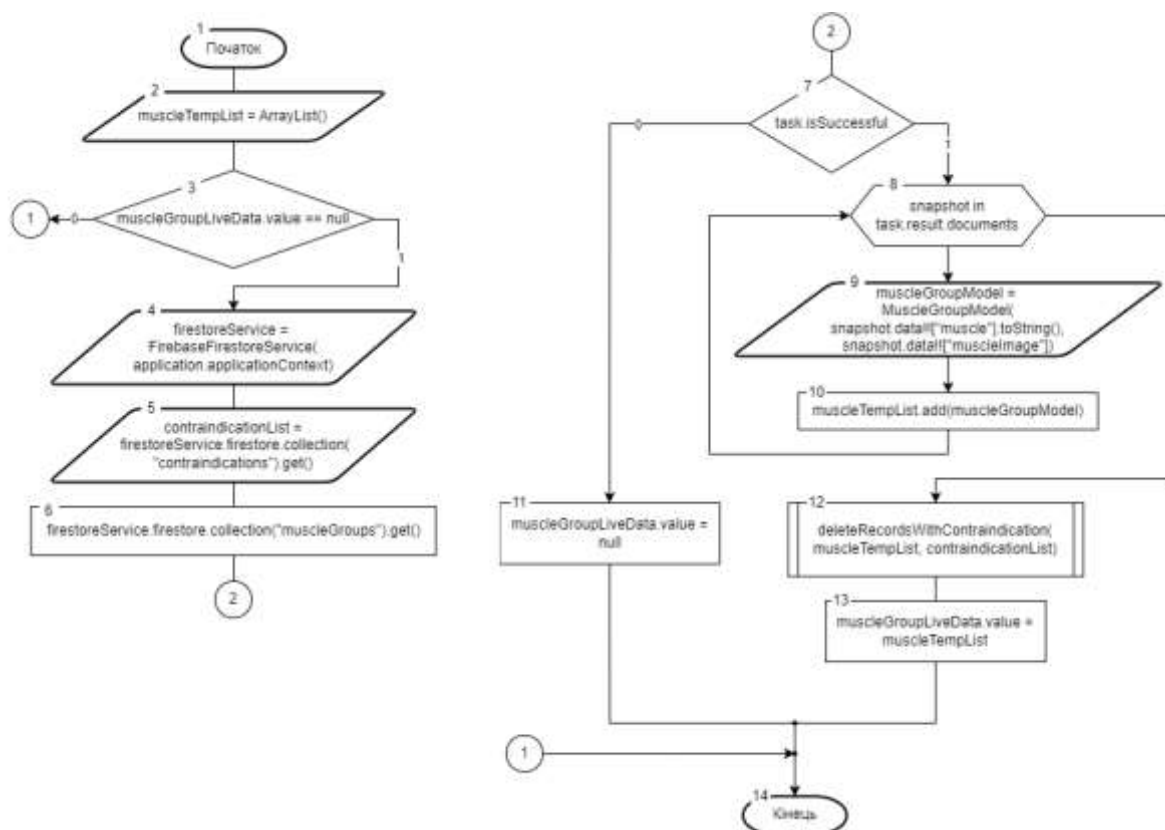


Рисунок 3.8 - Блок-схема алгоритму методу getMuscleGroups()

- створюється об'єкт MuscleGroupModel, де snapshot.data!!["muscle"] та snapshot.data!!["muscleImage"] отримують поля та перетворюються на рядки.
- створена модель додається до тимчасового списку muscleTempList.
- якщо запит не вдавсь, встановлюється значення muscleGroupLiveData в null, що свідчить про відсутність потреби в оновленні чи додаванні даних.
- оновлюється значення LiveData, додаючи записи на екран пристрою після завершення обробки всіх елементів.
- вихід із методу getMuscleGroups().

3.3. Розробка модуля створення нотаток

Для забезпечення функціоналу створення нотаток користувачем було розроблено метод addToDoItem(). Цей метод приймає два аргументи: todoArray - масив даних, введених користувачем для формування нового запису, та callback

- посилання на булеве значення, що забезпечує зворотний зв'язок щодо успішності виконання методу. Алгоритм роботи цього методу представлений на рисунку 3.9.

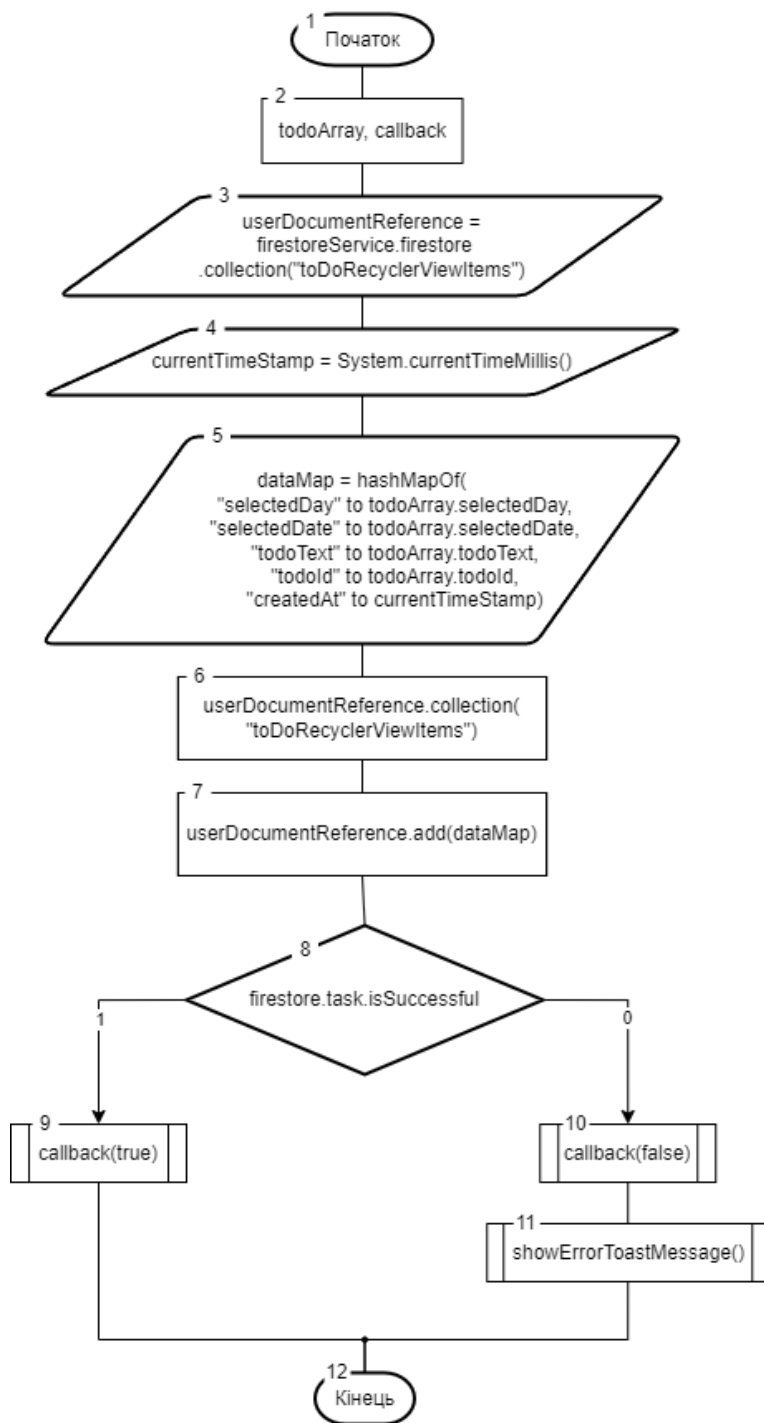


Рисунок 3.9 - Блок-схема алгоритму методу `addToDoItem()`

Пояснення до алгоритму наступні:

- виклик методу `addToDoItem()`.

- ініціалізація змінних `todoArray` та `callback` переданими значеннями аргументів.
- у змінну `userDocumentReference` зберігається посилання для підключення до бази `x Firebase`.
- у змінну `currentTimeStamp` зберігається мітка часу, що відповідає моменту створення нової нотатки.
- ініціюється змінна `dataMap`, яка являє собою набір даних у форматі пар «ключ-значення», до неї записуються такі дані, як день створення нотатки, дата створення нотатки, текст запису, ідентифікатор запису та час створення нотатки.
- здійснюється підключення до колекції, що містить дані про нотатки.
- новостворений запис додається до набору даних колекції нотаток.
- за допомогою вбудованої властивості `it.isSuccessful` перевіряється успішність операції, повертаючи булеве значення.
- якщо операція пройшла успішно, змінній `task` передається значення «true», що завершує з'єднання з базою даних та зберігає обліковий запис користувача.
- якщо операція не вдалася, змінній `task` передається значення «false», що завершує з'єднання з базою даних без збереження облікового запису користувача в базі даних.
- викликається вбудований метод `showErrorToastMessage()`, який інформує про виникнення помилки та відображає сповіщення типу `Toast`, інтегроване в систему `Android`.
- вихід із методу `addToDoItem()`.

Висновок

Протягом виконання кваліфікаційної роботи бакалавра було розроблено мобільний застосунок для отримання медичних консультацій. Мета розробки програмного забезпечення - підвищити якість процесу створення рекомендацій користувачам стосовно тренувальних програм, при цьому враховуючи їх індивідуальні медичні обмеження.

Актуальність теми зумовлена тим, що існуючі мобільні застосунки, спрямовані на підтримку здоров'я, мають недоліки, не повністю відповідають усім потребам користувачів, не завжди надають точну інформацію. Тому розробка власних мобільних додатків для здоров'я є актуальною.

У кваліфікаційній роботі бакалавра проаналізовано стан розвитку мобільних застосунків на платформі Android, розглянуто методи їх створення, проведено порівняльний аналіз аналогів, а також визначено напрямки їх удосконалення і підходи до розробки.

В процесі роботи створено архітектуру мобільного застосунку з використанням структурного шаблону MVVM, сформульовано загальний алгоритм роботи програмного забезпечення та способи подання тренувальних планів з урахуванням індивідуальних медичних протипоказань користувачів.

Здійснено порівняльний аналіз недоліків і переваг сучасних середовищ розробки та мов програмування, за результатами якого вибрано Kotlin як основну мову для створення додатка та Android Studio — як середовище розробки. Розглянуто процес створення ключових модулів, а саме інтерфейсу користувача, модуля персональних тренувальних планів і системи створення особистих записів.

Розроблений мобільний застосунок є зручним і практичним рішенням, яке спрощує отримання медичних консультацій та має потенціал для подальшого розширення функціональності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Types of Software Architecture Patterns [Електронний ресурс]. Режим доступу: <https://www.geeksforgeeks.org/types-of-software-architecture-patterns/>
2. AdaHealth [Електронний ресурс] – Режим доступу до ресурсу: <https://ada.com/>
3. MyFitnessPal [Електронний ресурс] – Режим доступу до ресурсу: <https://www.myfitnesspal.com>
4. The Fitbit App [Електронний ресурс] – Режим доступу до ресурсу: <https://www.fitbit.com/global/us/technology/fitbit-app>
5. Типи мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: <https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types>
6. Кросплатформова розробка додатків [Електронний ресурс] – Режим доступу до ресурсу: <https://wezom.com.ua/ua/blog/krossplatformennayarazrabotka-prilozhenij>
7. Гібридні мобільні додатки [Електронний ресурс] – Режим доступу до ресурсу: <https://web4u.in.ua/blog/g-bridn-mob-l-n-dodatki-ta-h-perevagi-18>
8. Архітектура мобільного застосунку [Електронний ресурс] – Режим доступу до ресурсу: <https://wezom.com.ua/ua/blog/arhitektura-mobilnogoprilozheniya>
9. Логіка відображення стану View в Android: проектуємо і тестуємо [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/33022/>
10. Інтегроване середовище розробки [Електронний ресурс] – Режим доступу до ресурсу: <https://w3schoolsua.github.io/hyperskill/ide.html#gsc.tab=0>
11. Android Studio [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/studio>
12. Eclipse <https://www.eclipse.org/downloads/>
13. IntelliJIdea [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/idea/>
14. Gradle [Електронний ресурс] – Режим доступу до ресурсу: <https://gradle.org/>

15. Kotlin [Електронний ресурс] – Режим доступу до ресурсу: <https://kotlinlang.org/>
16. Java [Електронний ресурс] – Режим доступу до ресурсу: <https://www.java.com/>
17. Огляд і основи мови програмування C++ [Електронний ресурс] – Режим доступу до ресурсу: http://www.znannya.org/?view=Cpp_basics
18. Navigation [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/guide/navigation>
19. Design your navigation graph [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/guide/navigation/design>
20. Build better apps faster with Jetpack Compose [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.android.com/develop/ui/compose>