НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«До захисту допущено»
В.о. завідувачки кафедри БМК

_____ Світлана АЛХІМОВА
(підпис) (ініціали, ПРІЗВИЩЕ

“___” __*червня*__2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп’ютерні технології в біології та медицині»
зі спеціальності *122 «Комп’ютерні науки»*

на тему: Мобільний застосунок для відстеження фізіологічних змін жінки під час вагітності

Виконала: студентка **IV** курсу, групи _БС-14__

ФЕДОРЧУК АНАСТАСІЯ ВІТАЛІЇВНА
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник: *ст. викл. каф. БМК, Бовсуновська Катерина Сергіївна*
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по ініціали) (підпис)

Консультант з розділів дипломної роботи:
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент: *Сичик Марина Михайлівна, доцент кафедри біомедичної Інженерії, к.т.н., доцент*
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент (-ка) _____
(підпис)

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувачки кафедри БМК
_____ Світлана АЛХІМОВА
«_ ___ »__*квітень*___2025 р.

ЗАВДАННЯ
на дипломну роботу студентці

ФЕДОРЧУК АНАСТАСІЇ ВІТАЛІЇВНІ
(прізвище, ім'я, по батькові)

1. Тема роботи **Мобільний застосунок для відстеження фізіологічних змін жінки під час вагітності**

керівник роботи
Бовсуновська Катерина Сергіївна, ст. викл. каф. БМК
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_26_»_*травня*_ 2025 р. №***1758-с***

2. Термін подання студентом роботи ***22-23 травня 2025 року***

3. Вихідні дані: медичні рекомендації, параметри фізіологічного стану жінок під час вагітності та наявні аналоги мобільних застосунків для відстеження фізіологічних змін жінки під час вагітності.

4. Зміст роботи: вступ, аналоги мобільних застосунків; засоби реалізації мобільних застосунків; обґрунтування вибраних технологій; методика розробки та програмна реалізація застосунку для моніторингу вагітності; узагальнення результатів роботи, загальні висновки.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): в роботі представлено 50 ілюстрацій та презентацію на 15 слайдів.

6. Консультанти розділів роботи[1*]

[1*] Якщо визначені консультанти. Консультантом не може бути зазначено наукового керівника магістерської дисертації.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Дипломної роботи			

7. Дата видачі завдання ***06 квітня 2025р.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 24.12.2024р.	***виконано***
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	***виконано***
3	Виконання основних розділів ДР (Вступ, аналоги мобільних застосунків; засоби реалізації мобільних застосунків; обґрунтування вибраних технологій; методика розробки та програмна реалізація застосунку; узагальнення результатів роботи)	Протягом усієї практики	
4	Перевірка ДР науковим керівником	до 20.05.2025	
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	
9	Захист ДР в ЕК	*16.06 - -21.06.2025*	

Студент _____ **Анастасія ФЕДОРЧУК**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Катерина БОВСУНОВСЬКА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Мобільний застосунок для відстеження фізіологічних змін жінки під час вагітності» виконана студентом кафедри біомедичної кібернетики ФБМІ *Федорчук Анастасією Віталіївною зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 5 розділів (аналоги мобільних застосунків; засоби реалізації мобільних застосунків; обґрунтування вибраних технологій; методика розробки та програмна реалізація застосунку для моніторингу вагітності; узагальнення результатів роботи), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 26 джерела.. Загальний обсяг роботи 80 сторінок.

Актуальність теми. Актуальність теми дипломної роботи пов'язана із задачею створення зручного, інтуїтивно зрозумілого мобільного застосунку для майбутніх мам, які хочуть якісніше слідкувати за власним здоров'ям під час вагітності.

Мета і завдання роботи. Метою роботи є покращення процесу моніторингу фізіологічного стану жінки у період вагітності за допомогою мобільного застосунку, розробленого на основі технологій Dart та Flutter.

Для досягнення мети потрібно вирішити наступні завдання:

1. Аналіз потреб користувачів;
2. Формування функціональних вимог до застосунку;
3. Проєктування структури та інтерфейсу мобільного застосунку;
4. Розробка застосунку за допомогою Dart і Flutter;
5. Забезпечення збереження та обробки даних користувача;
6. Тестування функціоналу та оцінка ефективності.

Використані методи. Під час розробки мобільного застосунку для відстеження фізіологічних змін жінки під час вагітності було використано такі

методи та технології:

1. Мова програмування Dart;
2. Фреймворк Flutter;
3. Середовище розробки Android Studio;
4. Дизайнерський інструмент Figma;
5. Архітектура MVVM;
6. База даних Firebase.

Отримані результати. У результаті проведеного дослідження було розроблено мобільний кросплатформений застосунок для платформ Android та iOS, який дозволяє вагітним жінкам зручно відстежувати фізіологічні та психологічні зміни протягом усього періоду вагітності. Реалізовано ключові функції, такі як календар вагітності, інтерактивні опитувальники стану, моніторинг змін та розділ з корисними порадами.

Ключові слова. Вагітність, мобільний застосунок, Flutter, здоров'я жінки, психологічний стан, моніторинг стану, UI/UX дизайн, цифрова підтримка, календар вагітності, щоденник вагітності.

Бібліографічний опис ДР

Федорчук А. В. Мобільний застосунок для відстеження фізіологічних змін жінки під час вагітності : дипломна роб. бакалавра : 122 Комп'ютері науки / Федорчук Анастасія Віталіївна. – Київ, 2025. – 80 с

ANNOTATION

Diploma work on the topic ‘Mobile application for tracking a woman's physiological changes during pregnancy’ was performed by a student of the Department of Biomedical Cybernetics FBMI Fedorchuk Anastasiia Vitaliivna, speciality 122 ‘Computer Science’ under the educational and professional programme ‘Computer Technologies in Biology and Medicine’ and consists of introduction; 5 chapters (analogues of mobile applications; means of implementing mobile applications; justification of selected technologies; methodology of development and software implementation of the application for pregnancy monitoring; summary of the results), conclusions to each of these chapters; general conclusions; a list of references, which includes 26 sources. The total volume of the paper is 80 pages.

Relevance of the topic. The relevance of the thesis topic is related to the task of creating a convenient, intuitive mobile application for expectant mothers who want to better monitor their own health during pregnancy.

Purpose and objectives of the study. Purpose and objectives of the study. The aim of the study is to improve the process of monitoring the physiological state of a woman during pregnancy using a mobile application developed on the basis of Dart and Flutter technologies.

To achieve the goal, the following tasks must be solved:

1. Analysis of user needs;
2. Formation of functional requirements for the application.;
3. Designing the structure and interface of a mobile application;
4. Developing an application using Dart and Flutter;
5. Ensuring the storage and processing of user data;
6. Functional testing and efficiency assessment.

Methods used. The following methods and technologies were used to develop a mobile application for tracking physiological changes in women during pregnancy:

1. Dart programming language;
2. Flutter framework;
3. Android Studio development environment;
4. Figma design tool;
5. MVVM architecture;
6. Firebase database.

Results. As a result of the research, a cross-platform mobile application was developed for Android and iOS platforms, which allows pregnant women to conveniently track physiological and psychological changes throughout the entire pregnancy period. Key functions such as a pregnancy calendar, interactive status questionnaires, monitoring of changes, and a section with useful tips have been implemented.

Keywords. Pregnancy, mobile application, Flutter, women's health, psychological state, condition monitoring, UI/UX design, digital support, pregnancy calendar, pregnancy diary.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних; впорядкований набір інформації, збережений у цифровому вигляді (зазвичай у табличках) для зручного пошуку та використання.

MVVM – Model-View-ViewModel; архітектурний шаблон проєктування, що використовується для відокремлення логіки користувацького інтерфейсу від бізнес-логіки застосунку.

UI – інтерфейс користувача, елементи, через які користувач взаємодіє із застосунком.

UX – користувацький досвід, загальне враження користувача від взаємодії із застосунком.

API – інтерфейс програмування застосунків; набір протоколів для взаємодії між програмними компонентами.

UUID – універсальний унікальний ідентифікатор, який використовується для створення унікальних ключів у системах зберігання даних.

Provider – бібліотека для управління станом у Flutter-застосунках.

IDE – інтегроване середовище розробки, що забезпечує програмістів інструментами для написання, налагодження та тестування коду в одному інтерфейсі.

ООП – об'єктно-орієнтоване програмування; парадигма програмування, що базується на використанні об'єктів, які об'єднують у собі дані та методи для роботи з ними, сприяючи гнучкості, масштабованості та повторному використанню коду.

ВСТУП

В контексті фізіологічних змін у жінки виникає багато нових відчуттів або симптомів, які можуть бути як нормальною частиною перебігу вагітності, так і потенційними ознаками ускладнень. Часто саме ці відчуття обговорюються з лікарем на прийомі, однак через швидкий ритм життя або психологічну втому майбутні мами не завжди мають змогу фіксувати й аналізувати зміни самостійно. Тож актуальність створення мобільного застосунку, який дозволяє щоденно моніторити самопочуття та фіксувати фізіологічні зміни, є цілком обґрунтованою.

Вагітність викликає значні фізіологічні зміни в організмі жінки, які підтримують ріст плода і готують тіло до пологів. Ці зміни зачіпають всі системи організму, включаючи серцево-судинну, дихальну, ниркову, ендокринну та метаболічну.

Об’єм крові збільшується на 45%, серцевий викид зростає на 40%, а судини розширюються, що знижує артеріальний тиск у першій половині вагітності. Матка тисне на нижню порожнисту вену, що може спричиняти зниження венозного повернення і артеріального тиску в положенні лежачи [19].

Нирки збільшуються в розмірах, швидкість клубочкової фільтрації зростає на 50-85%, що призводить до зниження рівня креатиніну та сечовини в крові. Підвищене вироблення прогестерону розслабляє сечоводи, що може спричинити застій сечі та підвищити ризик інфекцій [19].

Щитовидна залоза працює активніше, підвищується рівень тиреоїдних гормонів, що підтримує метаболізм і розвиток плода. Гормональні зміни також спричиняють підвищення рівня кортизолу та інсулінорезистентність у другій половині вагітності, що допомагає забезпечити плід глюкозою [26].

Обмін речовин перебудовується: організм матері починає зберігати більше жирів і використовувати їх як джерело енергії, щоб забезпечити плід

глюкозою. Зростає рівень холестерину і тригліцеридів у крові. Дихальна система пристосовується до підвищеної потреби в кисні. Збільшується хвилинний об'єм дихання, а рівень вуглекислого газу в крові знижується, що допомагає ефективніше передавати кисень плоду [19].

Кісткова система адаптується до підвищеного споживання кальцію плодом, збільшується всмоктування кальцію в кишечнику, а за необхідності він вивільняється з кісток матері [6].

Ці зміни слід моніторити, як жінці – в межах можливостей, так і лікарю – щоб контролювати правильний хід вагітності. А отже актуальність теми дипломної роботи полягає у важливості вирішення таких питань, як створення зручного, інтуїтивно зрозумілого мобільного застосунку для майбутніх мам, які хочуть якісніше слідкувати за власним здоров'ям під час вагітності.

Мета і завдання роботи

Метою роботи є покращення процесу моніторингу фізіологічного стану жінки у період вагітності за допомогою мобільного застосунку, розробленого на основі технологій Dart та Flutter.

Для досягнення мети потрібно вирішити наступні завдання:

1. Аналіз потреб користувачів;
2. Формування функціональних вимог до застосунку;
3. Проєктування структури та інтерфейсу мобільного застосунку;
4. Розробка застосунку за допомогою Dart і Flutter;
5. Забезпечення збереження та обробки даних користувача;
6. Тестування функціоналу та оцінка ефективності.

Використані методи. В роботі використано: мову програмування Dart, фреймворк Flutter, середовище розробки Android Studio, дизайнерський інструмент Figma, архітектуру MVVM, базу даних Firebase..

Отримані результати. У результаті проведеного дослідження було розроблено мобільний кросплатформений застосунок для платформ Android

та iOS, який дозволяє вагітним жінкам зручно відстежувати фізіологічні та психологічні зміни протягом усього періоду вагітності. Реалізовано ключові функції, такі як календар вагітності, інтерактивні опитувальники стану, моніторинг змін та розділ з корисними порадами.

Структура роботи

Дипломна робота за темою «Мобільний застосунок для відстеження фізіологічних змін жінки під час вагітності» виконана студенткою *Федорчук Анастасією Віталіївною зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині»*, побудована за класичним типом та викладена на 80 сторінках машинописного тексту. Вона складається з: вступу; 5 розділів (аналоги мобільних застосунків; засоби реалізації мобільних застосунків; обґрунтування вибраних технологій; методика розробки та програмна реалізація застосунку для моніторингу вагітності; узагальнення результатів роботи), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 26 джерел. В роботі представлено 50 рисунків і 3 таблиці.

РОЗДІЛ 1
АНАЛОГИ МОБІЛЬНИХ ЗАСТОСУНКІВ

В даному розділі розглянуто особливості перебігу вагітності, проаналізовано потребу в мобільних застосунках для відстеження стану майбутніх мам та аналоги таких мобільних застосунків, їхні особливості та функціонал.

1.1 Огляд мобільних застосунків для ведення вагітності

Складно уявити наше життя в сучасному світі без мобільних застосунків, здається, вони вже заполонили всі сфери нашого життя і, звісно ж, це не оминуло моніторинг вагітності. Такі застосунки дарують можливість майбутнім мамам отримувати інформаційну, медичну та психологічну підтримку не виходячи з дому. Це є дуже важливо, коли, наприклад, банально не вистачає часу відвідати лікаря. Також вони корисні при відстежуванні фізіологічних показників, планувати приймання ліків, дають майбутнім мамам можливість ділитися статистикою з медичними працівниками тощо.

Наприклад, дослідження, проведене в малозабезпечених регіонах Джорджії (США), продемонструвало, що більшість жінок високо оцінили переваги використання таких мобільних додатків для самостійного моніторингу стану здоров’я під час вагітності та після пологів [14]. Виходячи з цього, можна припустити, що впровадження подібних цифрових рішень може бути особливо актуальним і для України, де в окремих регіонах, наприклад віддалених селах, спостерігається обмежений доступ до кваліфікованих акушерських послуг. У цьому випадку мобільні застосунки допомагають не тільки отримувати корисну інформацію, а й краще піклуватися про здоров’я під час вагітності та запобігати можливим ускладненням.

1.1.1 Pregnancy Tracker – Kick Counter

Розглянемо один з найпопулярніших на сьогоднішній день застосунків – Pregnancy Tracker – Kick Counter. Його інтерфейс виконано в м’яких пастельних тонах, що додає відчуття довіри та затишку, а головний екран відразу надає доступ до ключових функцій (рис. 1.1).

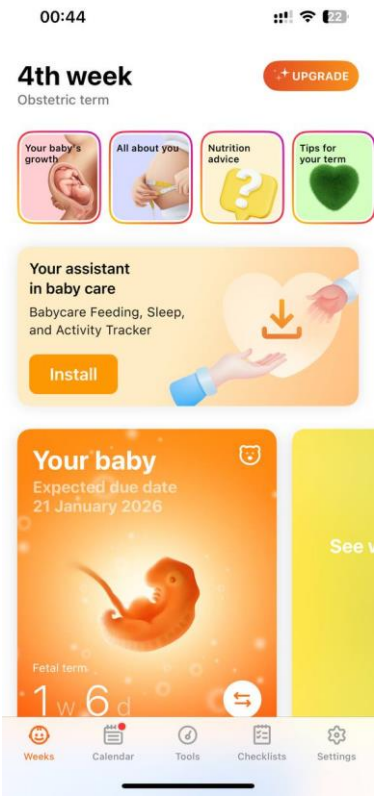

Рисунок 1.1 – Вікно застосунку Pregnancy Tracker – Kick Counter

З цікавого в даному застосунку є функція рахунку активності плоду, це є дуже корисно у третьому триместрі, також є функція нагадувань та календар, де можна додавати різні візити до лікаря, УЗД-дослідження чи особисті нотатки. Також з корисних функцій є поради щодо харчування, спорту та підготовки до пологів. Після аналізу можна виділити такі переваги та недоліки та записати їх в табл. 1.1.

Таблиця 1.1

Переваги та недоліки застосунку Pregnancy Tracker – Kick Counter

Переваги	Недоліки
Чіткий та зрозумілий інтерфейс	Деякі користувачі зазначають про проблеми оплати річної підписки
Є українська мова	Недостатня підтримка багатоплідної вагітності
Наявність цікавих порад та літератури	Більшість функцій доступні тільки при платній підписці
Наявність зручного календаря, куди можна вносити події та плани	При підтвердженні народження дитини – вся інформація видаляється назавжди
Генерація звітів і можливість відправки власному лікарю	Мало інформації про розвиток малюка на кожному етапі, користувачі зазначають, що часто мають необхідність користуватися додатковими ресурсами

1.1.2 Ovia Pregnancy & Baby Tracker

Другий і не менш корисний застосунок, що ми розглянемо це Ovia Pregnancy & Baby Tracker. Він є досить інтуїтивний та інформативним, супроводжує вагітність від перших тижнів до народження малюка. Він відрізняється деталізованим підходом до відстеження розвитку дитини та здоров'я мами, а також пропонує цікаві інтерактивні функції (рис. 1.2).

Однією з найцікавіших особливостей застосунку є візуалізація розвитку плоду. Кожного тижня вагітності користувачка отримує реалістичну ілюстрацію малюка, а також порівняння його розміру з предметами, фруктами чи тваринами (наприклад, "зараз ваша дитина завбільшки з авокадо"). Це робить процес більш зрозумілим та цікавим.

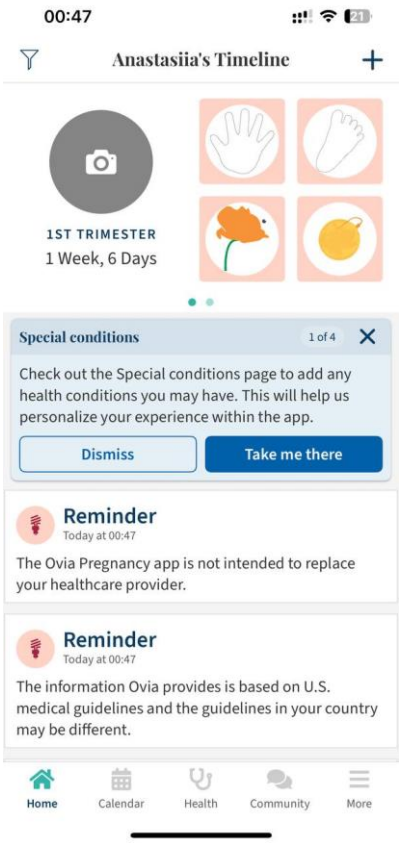

Рисунок 1.2 – Вікно застосунку Ovia Pregnancy & Baby Tracker

Застосунок також має календар вагітності з персоналізованими підказками, трекер показників мами (наприклад вага, артеріальний тиск, сон) та базу імен для майбутніх батьків, які ще не визначилися з ним. Також він має велику спільноту користувачів, де можна обговорити різні питання з іншими мамами. Отже, можна виділити такі переваги та недоліки, запишемо їх в табл. 1.2.

Таблиця 1.2

Переваги та недоліки застосунку Ovia Pregnancy & Baby Tracker

Переваги	Недоліки
Наявність цікавих порад та літератури	Відсутність української мови
Один з найкращих користувацьких інтерфейсів (оцінка 96% [21])	Має дещо нижчі технічні можливості у порівнянні з іншими застосунками (77%, [21]).
Допомога у виборі імені	

Продовж табл. 1.2

Переваги	Недоліки
Реалістичні ілюстрації кожний тиждень	
Порівняння розміру дитини з предметами/їжею/тваринами	
Показ розміру рук та ніг дитини	
Календар для оптимізації вагітності	
Відстеження основних показників мами	
Комунікація з іншими вагітними	
Безкоштовний контент	

1.1.3 AMMA Pregnancy & Baby Tracker

Останній розглянутий застосунок це AMMA Pregnancy & Baby Tracker. Він також корисний та досить популярний, поєднує в собі медично перевірену інформацію з практичними інструментами для майбутніх батьків (рис. 1.3).

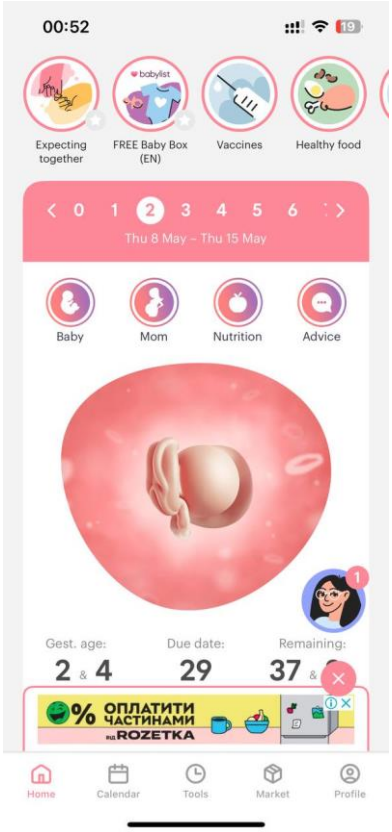

Рисунок 1.3 – Вікно застосунку AMMA Pregnancy & Baby Tracker

Він підійде тим, хто шукає інтуїтивно зрозумілий застосунок з підтримкою української мови. Тут є зручний календар, медичні статті з рекомендаціями ВООЗ і навіть білі шуми для кращого сну – наче дрібнички, але вони роблять вагітність трохи комфортнішою. Також є можливість додати свого партнера – приємний бонус для тих, хто хоче ділитися цим особливим періодом з коханою людиною. Проте застосунок не ідеальний, без платної підписки доведеться постійно стикатися з рекламою на кожному кроці. Також шкода, що розробники не передбачили можливості відстежувати харчування чи стан здоров'я мами – такі функції точно не були б зайвими. Отже, виділимо переваги та недоліки застосунку в табл 1.3.

Таблиця 1.3

Переваги та недоліки застосунку AMMA Pregnancy & Baby Tracker

Переваги	Недоліки
Зручний календар	Дуже багато реклами без підписки преміум
Багато корисних статей перевірених ВООЗ	Користувачі зазначають про велику кількість проблем завантаження фото
Наявність функції "Білі шуми" для сну	Можливість стеження тільки за природньою вагітністю
Наявність української	Багато технічних проблем в роботі на IOS
Можна слідкувати разом зі своїм партнером	Немає можливості слідкувати за дієтою
Наявність корисних інструментів, як: підбір імені, розрахунок кількості поштовхів, вага малюка	Немає можливості слідкувати за показниками здоров'я майбутньої мами

Висновок з розділу 1

У даному розділі було проведено огляд трьох популярних мобільних застосунків для ведення вагітності. Проаналізовано їх функціональні можливості, зручність інтерфейсу, наявність української мови, технічні

особливості та обмеження. Аналіз показав, що жоден із застосунків не забезпечує повний спектр функцій. Це створює передумови для розробки нового застосунку, який врахує ці недоліки й задовольнить потреби українських користувачів у комплексному та доступному рішенні для моніторингу вагітності.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ МОБІЛЬНИХ ЗАСТОСУНКІВ

В даному розділі розглянемо можливі варіанти реалізації мобільних застосунків та сучасні, популярні технології, завдяки яким це є можливим.

2.1 Вибір мови програмування

Під час створення мобільного застосунку ледь не перше, про що ми думаємо це яку ж мову програмування обрати. Від цього вибору залежить не лише технічна реалізація проєкту, а й сумісність із платформами, продуктивність та швидкість розробки. У цьому підрозділі ми розглянемо найпопулярніші мови програмування, що використовуються при розробці мобільних застосунків, а також їхні сильні/слабкі сторони та сферу застосування.

2.1.1 Java

Перша й певно найвідоміша мова – Java. Ця мова програмування відома своєю здатністю адаптуватися до різних систем і середовищ. Спочатку вона була створена для вирішення проблеми сумісності між платформами, а з часом стала невід'ємною частиною світу розробки – від серверної логіки і до мобільних застосунків. Особливість Java полягає у вдало поєднаних простоті та надійності: зрозуміла структура мови, ефективне управління пам'яттю та розвинена екосистема. Її стабільність та широка спільнота роблять Java надійним вибором як для нових проєктів, так і для вже існуючих [8].

Java є основною мовою програмування для створення Android-застосунків, що пояснюється її простотою, підтримкою ООП та величезною спільнотою. З моменту створення Android-платформа підтримує розроблення програм саме на Java, використовуючи спеціальні віртуальні машини Dalvik та Android Runtime (ART), які оптимізовані для роботи на мобільних пристроях з

обмеженими ресурсами [2].

На відміну від стандартної Java Virtual Machine, Android транслює Java-код у власний байт-код у форматі .dex, що виконується в середовищі Dalvik або ART. Це дозволяє ефективно використовувати пам’ять та забезпечувати високу продуктивність [2].

Android SDK надає набір компонентів (Activity, Service, ContentProvider, BroadcastReceiver), які реалізуються на Java і дозволяють створювати повноцінні мобільні застосунки. Ці компоненти відповідають архітектурній моделі MVC, що сприяє модульності та зрозумілості коду [2].

Незважаючи на деякі обмеження у порівнянні зі стандартною Java SE, Android підтримує значну частину її бібліотек, а також має власні, адаптовані до мобільних вимог API [2].

2.1.2 Kotlin

У 2011 році компанія JetBrains анонсувала створення нової мови програмування – Kotlin. Вона була задумана як сучасна альтернатива таким мовам, як Java і Scala, з можливістю запуску коду на віртуальній машині Java (JVM). Основною метою було усунути обмеження, які були у попередніх мовах, та запропонувати розробникам зручніший, компактніший і безпечніший інструмент для створення програмного забезпечення. Уже в 2017 році Google офіційно визнала Kotlin однією з основних мов розробки для Android, що стало поштовхом до її стрімкого поширення [24].

Kotlin швидко стала основою численних застосунків, які щодня використовують мільйони користувачів. Її використали провідні світові компанії, зокрема Google, Uber, Netflix, Capital One, Amazon та інші. Основними аргументами на користь Kotlin є її виразний синтаксис, сучасні можливості, такі як лямбда-вирази, розширення функцій, null-безпека, а також повна сумісність із уже існуючим Java-кодом [24].

Щоб зрозуміти, чому Kotlin став настільки популярним, важливо розглядати його в контексті розвитку Java. Попри надійність та багаторічну перевіреність, Java поступово застарівала. Мова, створена ще у 1995 році, не

враховувала багатьох сучасних принципів проєктування, через що стала громіздкою для вирішення типових завдань. Kotlin, навпаки, була спроєктована з урахуванням актуальних потреб розробників. Водночас її автори змогли зберегти сумісність із Java, усунувши її недоліки й не жертвуючи гнучкістю чи продуктивністю [24].

Ще однією перевагою Kotlin є її мультиплатформність. Хоча й найбільше вона асоціюється з Android-розробкою, можливості мови не обмежуються лише цією сферою. Kotlin дозволяє створювати застосунки для настільних операційних систем (наприклад, Windows та macOS), нативні рішення завдяки Kotlin/Native і також вебінтерфейси на JavaScript. Завдяки цьому мова розглядається не лише як мобільне рішення, а як універсальний інструмент у розробці програмного забезпечення [24].

2.1.3 Swift

Наступна мова – це Swift, сучасна мова, яку створили в Apple для розробки застосунків на айфони, макбуки, годинники та телевізори Apple. Вона поєднує в собі безпечний синтаксис, високу продуктивність та інтеграцію з існуючими фреймворками, такими як Cocoa та Cocoa Touch. Swift була розроблена з урахуванням можливості взаємодії з API, написаними на Objective-C, це дозволяє розробникам спокійно переходити до нової мови без необхідності повного переписування існуючого коду [20].

Основні переваги Swift:

1) механізми, що запобігають помилкам, пов'язаним з нульовими значеннями (null safety);
2) висока швидкість виконання коду;
3) лаконічний та читабельний код;
4) повна підтримка в середовищі розробки Xcode, включаючи інструменти для налагодження та тестування.

Завдяки цим характеристикам, Swift стала основною мовою для розробки застосунків в екосистемі Apple, забезпечуючи розробникам потужний та зручний інструмент для створення сучасних мобільних рішень.

2.1.4 Dart

Створена компанією Google, Dart є сучасною об’єктно-орієнтованою мовою програмування, що вирізняється своєю універсальністю, продуктивністю та зручністю для розробки кросплатформних застосунків. Подібно до Java чи C++, вона використовує парадигму, в якій усе є об’єктом: навіть прості типи даних, функції й навіть null є екземплярами класів. Такий підхід надає мові гнучкість і дозволяє уніфіковано працювати з будь-яким типом даних, значно спрощуючи логіку обробки даних та взаємодії між компонентами [23].

Важливою перевагою Dart є чистий, лаконічний синтаксис, який легко читається та швидко засвоюється навіть новачками. Водночас, мова підтримує потужні інструменти, зокрема строгі типи, розширюваність класів, лямбда-функції, а також вбудовану підтримку асинхронного програмування через async/await. Ці можливості дають змогу створювати складні, високопродуктивні застосунки без зайвого шаблонного коду [23].

Що робить Dart по-справжньому унікальною, так це її глибока інтеграція з Flutter, провідним фреймворком для кросплатформної мобільної розробки. Завдяки Dart, розробники можуть створювати додатки одночасно для Android і iOS з єдиною кодовою базою, отримуючи нативну швидкість та візуальну плавність. Гаряче перезавантаження (hot reload), одне з найбільших досягнень Flutter, стало можливим саме завдяки гнучкості Dart, що дозволяє миттєво бачити зміни в інтерфейсі без повної перезбірки програми. Тобто, простими словами, розробник може бачити зміни одразу на екрані емулятора, що є надзвичайно зручно.

Крім того, Dart компілюється як у байт-код для віртуальної машини (JIT) – зручно для розробки, так і в нативний машинний код (AOT) – що критично для продуктивності у фінальній збірці. Це забезпечує баланс між швидкістю розробки та ефективністю виконання [23].

2.2 Фреймворки для мобільної розробки

У сучасній мобільній розробці фреймворки відіграють ключову роль, забезпечуючи розробникам ефективні інструменти для створення застосунків під різні платформи. Вони дозволяють суттєво зменшити час розробки, повторно використовувати код, а також спрощують підтримку проєктів. Завдяки фреймворкам можна обирати між нативною, кросплатформною чи гібридною розробкою в залежності від поставлених завдань і технічних вимог. У цьому підрозділі буде розглянуто популярні рішення, які використовуються для створення мобільних застосунків.

2.2.1 Flutter

Flutter – це відкритий фреймворк інтерфейсу користувача, створений компанією Google для розробки сучасних, візуально привабливих та високопродуктивних мобільних застосунків під Android і iOS з єдиною кодовою базою. Його унікальність полягає в тому, що він дозволяє створювати повністю нативні застосунки без потреби використовувати окремі мови для логіки та інтерфейсу – замість цього все пишеться мовою Dart, яка компілюється у швидкий ARM-код та JavaScript, готовий до продакшену.

Фреймворк базується на двовимірному рендеринговому рушії Skia, який забезпечує відтворення графіки незалежно від платформи, використовуючи як CPU-рендеринг, так і прискорення за допомогою OpenGL ES2. Цей самий рушій використовується у таких продуктах, як Google Chrome, Chrome OS, Android та Firefox, що підкреслює його стабільність і продуктивність. Завдяки Skia, інтерфейс у Flutter може оновлюватися з частотою до 60 або навіть 120 кадрів на секунду, що забезпечує плавні анімації та швидкий відгук на дії користувача [17].

У Flutter вся логіка інтерфейсу реалізується за допомогою віджетів, які є конфігураціями окремих елементів UI. Кожен віджет – це як "цеглинка LEGO", з якої будується інтерфейс програми. Поєднуючи ці "цеглинки", формується дерево віджетів (widget tree), що відображає структуру

інтерфейсу. Після цього Flutter будує дерево елементів (element tree), яке відповідає за візуалізацію на екрані. Цей підхід дозволяє фреймворку автоматично оновлювати інтерфейс кожного разу, коли змінюється стан даних.

Як вже і зазначалось вище, однією з найсильніших сторін Flutter є функція hot reload, яка дозволяє миттєво оновлювати інтерфейс під час розробки без втрати стану додатку. Це надзвичайно прискорює процес створення та тестування нових функцій, оскільки розробник одразу бачить результат змін прямо на пристрої або емуляторі.

Крім мобільної розробки, Google активно розвиває Flutter і в інших напрямах – для веб-застосунків, десктопних програм (Flutter Desktop) та вбудованих пристроїв (як-от Raspberry Pi або автомобільні мультимедійні системи). Це перетворює Flutter на універсальний фреймворк, який можна використовувати для створення застосунків практично на будь-якій платформі [17].

2.2.2 React Native

Розглянемо ще один фреймворк, а саме React Native. Його використовують для створення мобільних застосунків за допомогою JavaScript і спеціальної бібліотеки React, яку зробила компанія Facebook. Головна особливість React Native полягає в тому, що він не використовує веб-переглядач для інтерфейсу, як це роблять гібридні платформи на зразок Cordova чи Ionic. Замість цього JavaScript-код компілюється у нативні компоненти, це забезпечує високу швидкість та нативну взаємодію з платформою – Android або iOS.

React Native дозволяє створювати інтерфейси, які виглядають і працюють як справжні нативні елементи, забезпечуючи доступ до платформних API та модулів. Це відкриває можливість створення кросплатформних мобільних застосунків з єдиною кодовою базою, що значно економить час і ресурси. Оскільки JavaScript – одна з найпоширеніших мов програмування у веброзробці, перехід до мобільної розробки з React Native є

логічним і доступним для багатьох розробників [18].

React Native активно розвивається за підтримки Facebook та потужної спільноти, і вже використовується у таких великих продуктах, як Facebook, Instagram, Tesla, Bloomberg, Airbnb, Amazon та Microsoft. Завдяки цьому фреймворк постійно вдосконалюється, з'являються нові інструменти, плагіни й бібліотеки [18].

2.2.3 Xamarin

Наступний фреймфорк – Xamarin. Це фреймворк для створення кросплатформних мобільних застосунків, що дозволяє писати єдиний код на мові C# для різних платформ. Xamarin добре працює з платформою .NET і інструментами від Microsoft, тому дає змогу створювати стабільні та швидкі застосунки, використовуючи один і той самий код для різних операційних систем.

Однією з головних переваг фреймворку є використання Xamarin.Forms – технології, що дозволяє створювати інтерфейси, які працюють однаково на різних пристроях. Однією з найбільших переваг Xamarin.Forms є те, що вона надає можливість розробляти нативні мобільні застосунки одночасно для кількох платформ. Це досягається за рахунок архітектурного поділу рішень на декілька рівнів, що дозволяє розділити бізнес-логіку та інтерфейс користувача [13].

Кросплатформенне рішення Xamarin.Forms зазвичай використовує спільний C#-код, який містить бізнес-логіку та шар доступу до даних. Цей рівень часто називають Core Library. Над ним розташовується UI-шар, також написаний на C#, який відповідає за візуальну частину застосунку. На самому верху – тонкий шар платформозалежного C#-коду, що відповідає за ініціалізацію та запуск застосунку в кожній конкретній операційній системі [13].

Такий підхід дозволяє максимально повторно використовувати код і підтримувати чисту архітектуру. Крім того, Xamarin дозволяє використовувати камеру, GPS, Bluetooth, файлову систему та інші ресурси

пристрою без необхідності переходу на нативний код.

2.2.4 NativeScript

Останній фреймворк який розглянемо – NativeScript. Це фреймворк з відкритим кодом, який дозволяє писати один код і використовувати його для створення застосунків як на Android, так і на iOS. Його розробником є компанія Telerik. Також у NativeScript є можливість створення повністю нативних застосунків без потреби у вивченні таких мов, як Java, Swift або Objective-C. Замість цього розробники можуть використовувати знайомі технології – JavaScript, HTML-подібну розмітку (на базі XML) та CSS для стилізації інтерфейсу.

Архітектура додатків у NativeScript складається з трьох основних складових: логіка реалізується на JavaScript (або на TypeScript), інтерфейс описується за допомогою XML, а візуальне оформлення задається через CSS. При цьому, на відміну від вебзастосунків, інтерфейс у NativeScript відображається у вигляді нативних компонентів відповідної платформи – тобто елементи інтерфейсу виглядають та поводяться так само, як і в застосунках, написаних на платформах Android або iOS нативними засобами [5].

Однією з переваг NativeScript є можливість повного доступу до API операційних систем, що дає змогу працювати з камерами, сенсорами, геолокацією тощо без потреби в сторонніх обгортках або плагінах. Також фреймворк підтримує інтеграцію з Angular, що дозволяє створювати добре структуровані та масштабовані додатки, хоча ця інтеграція не є обов'язковою.

Завдяки тому, що NativeScript орієнтований на розробників із досвідом у вебтехнологіях, поріг входу в розробку мобільних застосунків стає значно нижчим. Це робить фреймворк зручним інструментом як для новачків, так і для професіоналів.

2.3 Середовища розробки

Розробка мобільних застосунків потребує не лише вибору мови програмування чи фреймворку, але й зручного середовища розробки, яке забезпечує інтеграцію інструментів, емуляторів, відлагодження та управління проєктом. Інтегровані середовища розробки (IDE) значно спрощують створення мобільних застосунків, підвищують ефективність команди та дозволяють зосередитися безпосередньо на реалізації функціоналу. У цьому підрозділі розглядаються популярні IDE, які використовуються у мобільній розробці.

2.3.1 Android Studio

Android Studio – це середовище розробки, яке було розроблене компанією Google (рис. 2.1). Воно базується на IntelliJ IDEA та містить усі необхідні інструменти для повноцінної розробки: редактор коду, інтегрований емулятор пристроїв, систему збірки Gradle, засоби відлагодження та аналізу продуктивності. Android Studio підтримує написання коду мовами Java, Kotlin і C++, а також пропонує зручний візуальний редактор інтерфейсу. Завдяки регулярним оновленням, глибокій інтеграції з Android SDK та можливості роботи з сучасними інструментами розробки, ця IDE є найпоширенішим вибором серед Android-розробників.

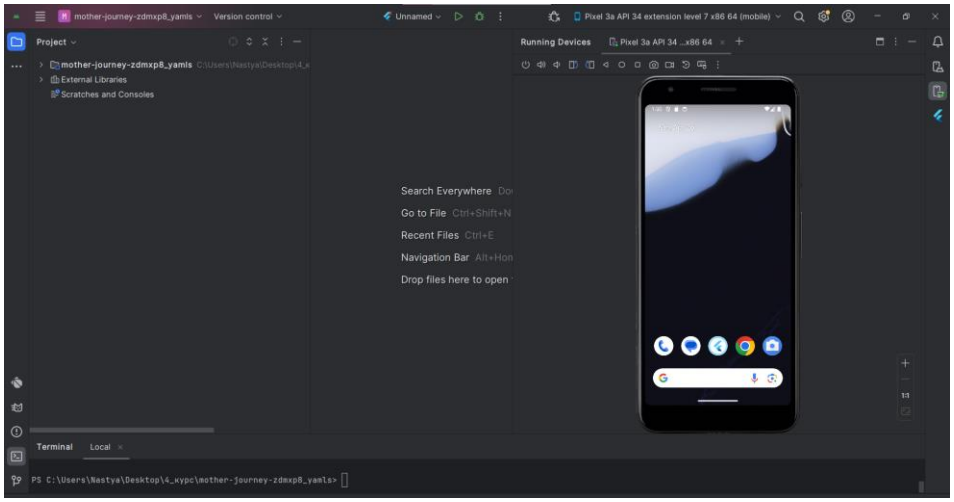

Рисунок 2.1 – Приклад вікна Android Studio

2.3.2 Xcode

Тепер розглянемо Xcode (рис. 2.2). Це офіційне середовище розробки від компанії Apple, призначене для створення застосунків під їхні пристрої. Це комплексне рішення, яке включає в себе редактор коду, інтерфейсний дизайнер (Interface Builder), емулятори пристроїв, інструменти для тестування, профілювання продуктивності та багатофункціональний відлагоджувач. Xcode надає розробникам зручний доступ до всіх необхідних інструментів для створення, запуску та розгортання нативних застосунків.

Основна мова програмування в Xcode це Swift, хоча також підтримується Objective-C. Середовище забезпечує глибоку інтеграцію з усіма сервісами Apple, включно з App Store, iCloud та Core Data. Це робить його основним інструментом для розробки застосунків саме для пристроїв Apple.

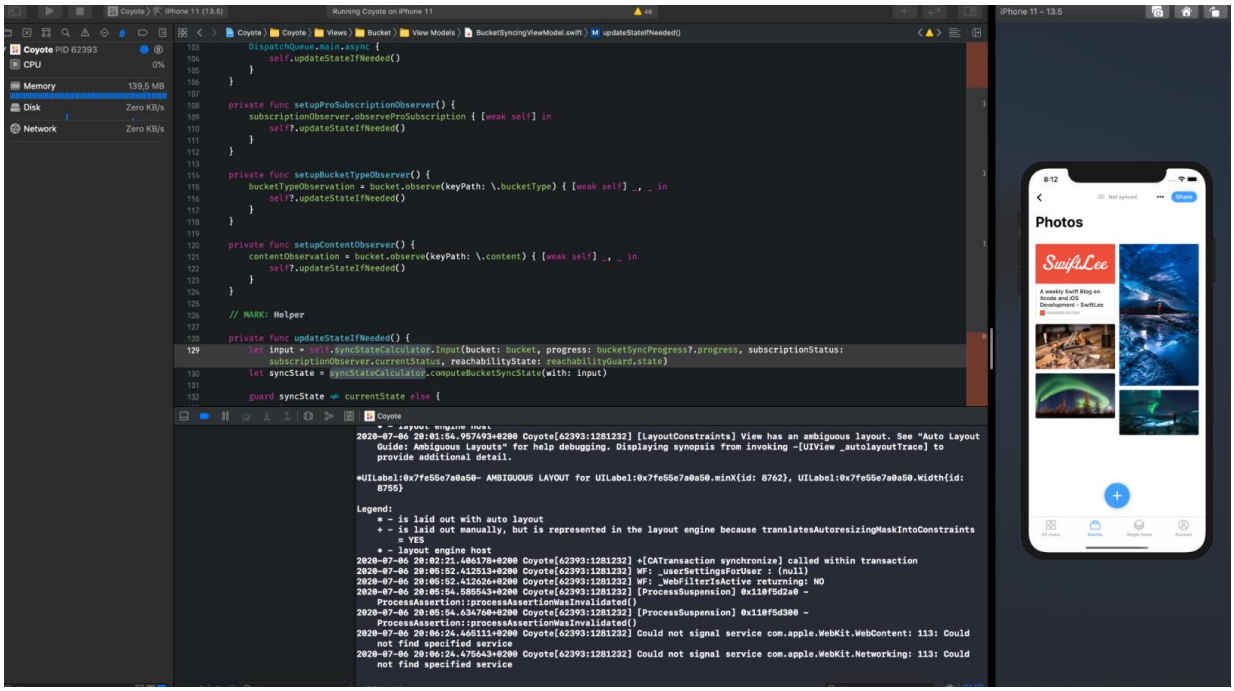

Рисунок 2.2 – Приклад вікна Xcode [12]

2.3.3 Visual Studio Code

Останнє середовище розробки, яке ми розглянемо це Visual Studio Code. Воно розроблене компанією Microsoft, активно використовується для мобільної розробки, особливо у зв'язці з такими фреймворками, як Flutter,

React Native або Ionic (рис. 2.3). Завдяки великій кількості доступних розширень, підтримці багатьох мов програмування, інтеграції з системами контролю версій та зручному налагодженню, VS Code є гнучким інструментом як для початківців, так і для досвідчених розробників. Незважаючи на свою простоту, це середовище дозволяє зручно працювати над розробкою мобільних застосунків та легко адаптується під конкретні потреби розробника.

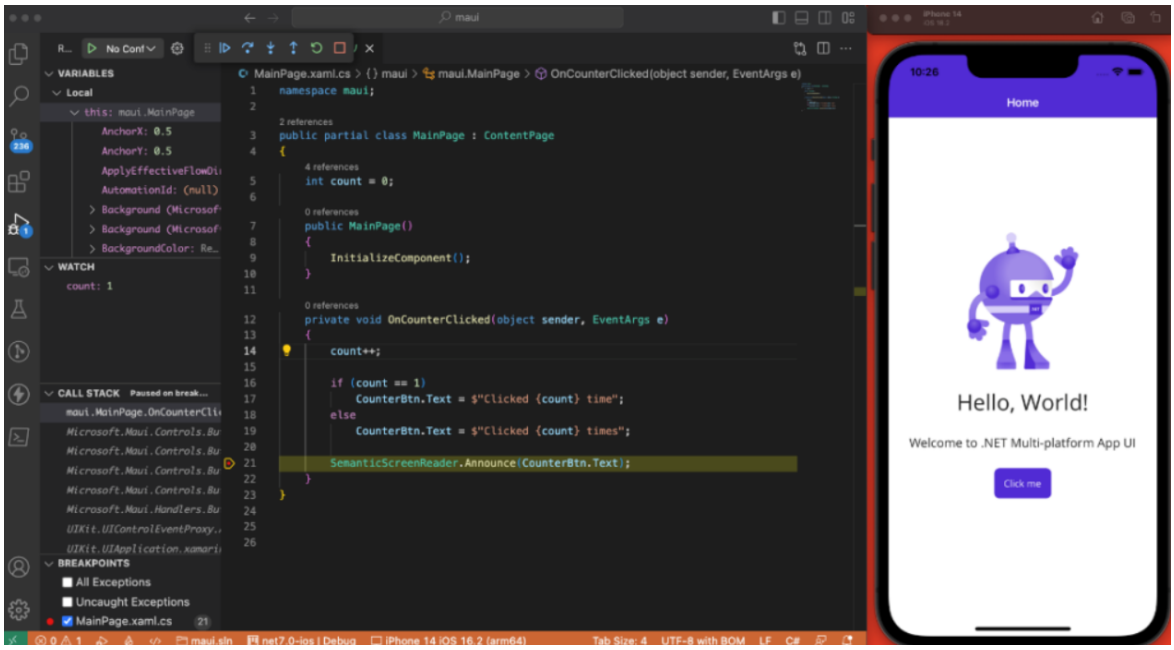

Рисунок 2.3 – Приклад вікна Visual Studio Code [4]

2.4 Інструменти дизайну інтерфейсу

У процесі створення мобільних застосунків важливо не лише реалізувати функціональність, але й забезпечити зручний та привабливий інтерфейс користувача. Інструменти дизайну інтерфейсу дають змогу проєктувати макети екранів, створювати інтерактивні прототипи, тощо. Вони спрощують взаємодію між дизайнерами та розробниками, забезпечуючи точну передачу візуальних рішень у готовий продукт. У цьому підрозділі буде розглянуто найбільш популярні інструменти, що використовуються для UI/UX-дизайну мобільних застосунків.

2.4.1 Figma

Один із найзручніших, інтуїтивно зрозумілих та надзвичайно популярних інструментів для створення дизайну мобільних застосунків сьогодні – це Figma (рис. 2.4). Її популярність пояснюється не лише широким набором функцій, а й доступністю, простотою роботи в команді та інноваційним підходом до організації дизайнерських процесів. Завдяки тому, що Figma працює прямо в браузері, вона доступна на будь-якій операційній системі – Windows, macOS, Linux і навіть Chromebook. Це дає змогу командам, які працюють на різних платформах, ефективно співпрацювати в одному середовищі без встановлення програмного забезпечення [9].

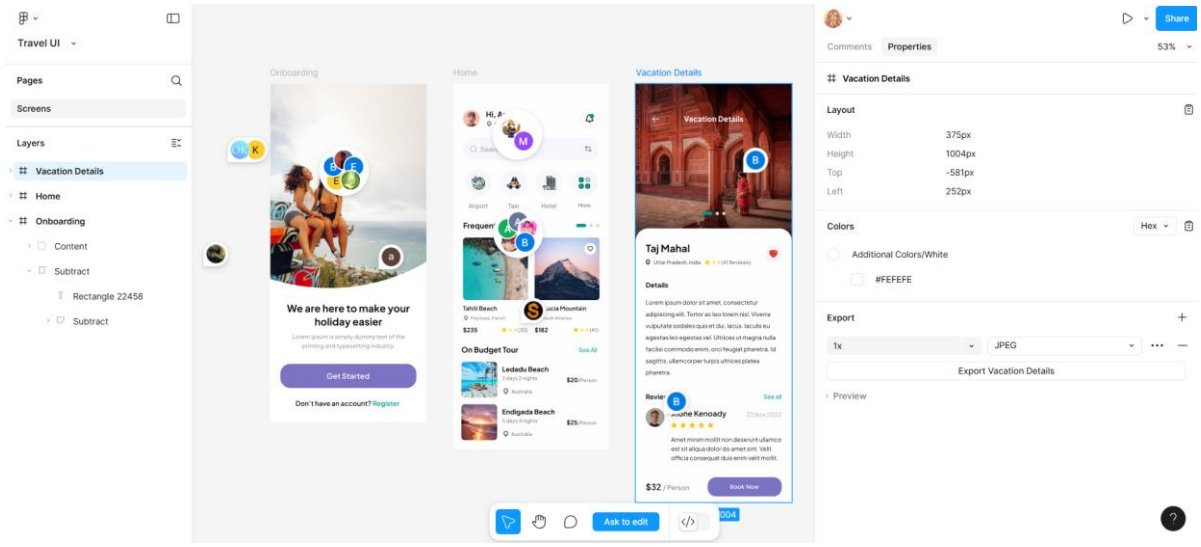

Рисунок 2.4 – Приклад вікна Figma

Однією з найсильніших сторін Figma є її потужна система багаторазового використання компонентів. Дизайнери можуть створювати гнучкі компоненти з індивідуальними налаштуваннями – змінювати розмір, текст, стилі, видимість, а також вкладати одні компоненти в інші без втрати зв’язку з основним шаблоном. Це дозволяє зберігати цілісність дизайну без необхідності «від'єднувати» елементи від базових компонентів. Крім того, Figma надає можливість детального контролю над тим, які компоненти доступні з окремих документів, що значно спрощує побудову структурованої

дизайн-системи без перевантаження проєкту зайвими елементами [9].

Ще однією перевагою є дуже щедрий безкоштовний тариф, завдяки якому Figma стала популярною серед фрилансерів, початківців і дизайнерів з країн, що розвиваються. Платний план зазвичай потрібен лише в разі командної роботи, великої кількості проєктів або створення власних бібліотек компонентів. Усе це, разом із хмарною архітектурою та зручністю командної взаємодії, робить Figma по-справжньому революційним інструментом у сфері UI/UX-дизайну [9].

2.4.2 Adobe XD

Серед інструментів, що поєднують простоту, швидкість і інтеграцію з потужною графічною екосистемою, особливе місце займає Adobe XD. Даний інструмент надає комфортне середовище для створення макетів інтерфейсів, інтерактивних прототипів і є зручним для передачі дизайну розробникам. Adobe XD підтримує як створення статичних екранів, так і налаштування анімованих переходів, що дозволяє моделювати повноцінну логіку роботи застосунку ще до етапу програмування [7].

Завдяки глибокій інтеграції з іншими продуктами Adobe, такими як Photoshop та Illustrator, дизайнери можуть легко використовувати вже створені елементи або редагувати графіку безпосередньо з XD. Також серед переваг можна виділити: підтримку компонентів, автоматичне масштабування для різних пристроїв, режим спільної роботи та можливість попереднього перегляду прототипів на мобільних пристроях. Хоч Adobe XD і поступається Figma у гнучкості хмарної взаємодії, він залишається надійним вибором [7].

2.4.3 Sketch

Для багатьох дизайнерів, які працюють на macOS, Sketch – це універсальний векторний інструмент, створений спеціально для цієї операційної системи. Його простий, але гнучкий інтерфейс та підтримка готових шаблонів дозволяють миттєво розпочати роботу над новим інтерфейсом. Sketch дозволяє працювати над дизайном разом з іншими в реальному часі, що зручно для обговорень, обміну ідеями та швидкого

створення прототипів у команді. [15].

На відміну від багатьох конкурентів, Sketch відзначається гнучкою системою символів, адаптивним дизайном і нативною роботою без підключення до Інтернету. Завдяки цьому дизайнери можуть створювати інтерфейси, не турбуючись про втрату доступу до своїх проєктів. Крім того, програма підтримує відкритий формат файлів і має активну спільноту, яка регулярно створює нові плагіни та шаблони, що значно розширює функціональність базового інструменту [15].

Завдяки функціям, таким як оверлеї, бібліотеки символів, перегляд прототипів і контроль версій, Sketch добре підходить як для початкового створення вайрфреймів, так і для фінальної підготовки дизайну до розробки. Це стабільне рішення для тих, хто надає перевагу глибокій візуальній кастомізації та автономній роботі без залежності від хмарних платформ.

Висновок з розділу 2

У даному розділі були розглянуті сучасні технології для розробки мобільних застосунків, а саме: популярні мови програмування, фреймворки, середовища розробки та інструменти для дизайну, їх слабкі та сильні сторони.

РОЗДІЛ 3
ОБҐРУНТУВАННЯ ВИБРАНИХ ТЕХНОЛОГІЙ

Для реалізації зручного, ефективного та надійного мобільного застосунку було важливо обрати технології, які відповідають вимогам та потребам користувача і безпеки даних, тому саме це ми і розглянемо в даному розділі.

3.1 Обґрунтування кросплатформенності

Одразу зрозуміло, що вагітні жінки можуть мати різну вікову категорію, різний рівень технічної грамотності та можуть мати абсолютно різні смартфони. Перед нами стояла задача розробити застосунок з урахуванням всіх цих аспектів. Саме тому перед розробкою застосунку ми прийшли до висновку, що найкращим варіантом буде зробити його кросплатформеним, тобто з підтримкою як Android так і iOS.

Цей підхід дозволяє значно розширити цільову аудиторію, адже користувачкам не потрібно турбуватись про сумісність пристрою із застосунком. Більше того, це дає можливість підтримувати однаковий функціонал та інтерфейс на обох платформах, що спрощує подальшу розробку та супровід. Для медичного застосунку стабільність і надійність відображення інформації є критично важливими. Саме кросплатформенний підхід забезпечує можливість однаково якісної роботи додатку незалежно від операційної системи.

3.2 Вибір технологій розробки

Після визначення потреби у кросплатформенності наступним логічним кроком був вибір фреймворку та мови програмування. Серед

найпопулярніших варіантів для кросплатформеної розробки були розглянуті: React Native, Xamarin та Flutter. У результаті аналізу було обрано Flutter. Це фреймворк з відкритим кодом, який підтримується компанією Google та працює на мові програмування Dart.

Найзручніше у Flutter те, що з його допомогою можна зробити швидкий і плавний інтерфейс, не пишучи окремий код для кожної платформи – все працює з однієї кодової бази. Для нашого застосунку це стало вирішальним фактором, адже ми прагнули досягти стабільності, швидкодії та гарного візуалу. Крім того, Flutter пропонує велику кількість готових віджетів, що дає змогу швидко розробити зрозумілий і функціональний інтерфейс для майбутніх мам.

Ще одним важливим аргументом на користь Flutter є велика спільнота і велика кількість документації, що значно полегшує процес розробки, вирішення помилок та масштабування проєкту в майбутньому.

3.3 Вибір архітектури

При розробці будь-якого програмного забезпечення надзвичайно важливою є продумана архітектура – саме вона визначає легкість масштабування продукту у майбутньому. У моєму застосунку для моніторингу вагітності я вирішила зробити акцент на чіткому розділенні відповідальностей між частинами коду, що забезпечує стабільність, зрозумілість структури та зручність у підтримці. Архітектура дозволяє швидко знаходити помилки, тестувати окремі компоненти та легко змінювати одну частину проєкту без ризику порушити роботу всієї системи [3].

Для реалізації цього проєкту ми свідомо зупинилися на архітектурній моделі MVVM. Цей підхід особливо ефективний у Flutter-розробці, оскільки забезпечує гнучке управління станом та дозволяє організувати код так, щоб окремі частини не залежали одна від одної без потреби.

Візуальна частина – View відповідає виключно за відображення даних і

реакцію на дії користувача. Вона не містить логіки обробки, не знає, звідки беруться дані чи як вони змінюються. Наприклад, сторінка з календарем або модуль нотаток працює лише з тими даними, які ViewModel вже підготувала [16].

ViewModel виступає як проміжна ланка: вона обробляє взаємодію користувача, викликає відповідні методи сервісів, оновлює стан і передає View вже готову інформацію для відображення. У проєкті, наприклад, сторінка з опитуванням фізичного чи психологічного стану повністю реалізована за цим принципом – логіка побудови питань, збереження відповідей і зміна стану винесені у ViewModel [16].

В Model входять структури даних, сервіси для взаємодії з Firebase (зокрема Firestore, авторизацію) та допоміжну логіку. Наприклад, усі записи про фізичний стан користувачки або нотатки в календарі описані у відповідних моделях, які використовуються для зберігання та обробки [16].

Архітектура MVVM дозволила мені працювати над окремими сторінками незалежно, розділити логіку збереження, обробки даних та інтерфейс, що значно полегшує підтримку і масштабування проєкту.

3.4 Керування станом

У структурі додатку передбачено кілька ключових функцій, зокрема календар вагітності, опитувальники психологічного та фізіологічного стану, а також моніторинг. Тобто у застосунку активно змінюється стан, і ним потрібно ефективно управляти. Саме тому було обрано бібліотеку Provider для керування станом у Flutter [22].

Provider – це один із найпопулярніших і водночас простих у використанні інструментів для керування станом у Flutter. Його перевага полягає в тому, що він добре підходить для застосунків середньої складності, дозволяє гнучко передавати дані між екранами, а також підтримує читабельність і структурування коду. Для додатку, що збирає й обробляє дані

про здоров’я користувачки, важливо, щоб зміни стану відбувались точно, стабільно та без втрат [22].

Крім того, Provider дозволяє зручно реалізувати оновлення інтерфейсу в режимі реального часу без надмірного навантаження на систему. У порівнянні зі складнішими рішеннями, такими як Bloc чи Redux, Provider виявився оптимальним вибором, враховуючи специфіку застосунку.

3.5 Вибір дизайнерських інструментів

Враховуючи специфіку цільової аудиторії, а саме вагітних жінок, інтерфейс застосунку мав бути інтуїтивно зрозумілим, візуально приємним і максимально доступним. Саме тому важливу роль у процесі розробки відіграли інструменти UI/UX-дизайну. Для створення прототипів, макетів та загальної структури інтерфейсу було обрано онлайн-сервіс Figma.

Як вже було розглянуто, Figma – це сучасний, хмарний редактор, що дозволяє працювати над дизайном у режимі реального часу, спільно з командою, не залежно від операційної системи. Це особливо зручно під час тестування макетів, узгодження візуальних рішень та адаптації дизайну під різні розміри екранів.

Для нашого застосунку важливо було створити простий і ніжний інтерфейс, що не перевантажує користувача й водночас дозволяє швидко знайти потрібну інформацію або ввести дані. Figma допомогла нам це реалізувати.

3.6 Вибір бази даних

Для того, аби забезпечити зберігання даних та налаштування авторизації було обрано Firebase. Це хмарне рішення ідеально інтегрується з Flutter, забезпечуючи швидке й просте підключення функцій збереження, автентифікації та аналітики. Firebase дозволяє уникнути необхідності

створювати власний сервер, що значно зменшило обсяг роботи. Також було забезпечено відповідний рівень безпеки даних, що є критично важливим при роботі з персональною інформацією користувачів.

Висновок до розділу 3

У даному розділі розглянуто вибір основних технологій, зокрема фреймворку, мови програмування, архітектури, дизайнерського інструменту та БД, що дозволили ефективно реалізувати мобільний застосунок з урахуванням технічних та функціональних вимог.

РОЗДІЛ 4
МЕТОДИКА РОЗРОБКИ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ ВАГІТНОСТІ

В даному розділі ми розглянули процес створення мобільного застосунку для моніторингу вагітності. Було детально описано вибір кольорової палітри та підхід до UX/UI-дизайну, що забезпечує зручність для майбутніх мам. Також було наведено обґрунтування використання архітектури MVVM та представлено структуру проєкту. Розглянуто використані технології: Flutter, Firebase та інші бібліотеки. Покроково розкрито реалізацію основних функцій: авторизація, календар з нагадуваннями, щоденні опитування стану, перегляд статистики, перегляд корисних порад і створення списку речей до пологового.

4.1 Інтерфейс користувача

Перед розробкою програмного продукту дуже важливо продумати інтерфейс користувача. Чи можемо ми уявити сучасний застосунок без гарного дизайну? Чи зручно буде використовувати застосунок, якщо інтерфейс буде складним та інтуїтивно не зрозумілим? Відповідь на це питання – звичайно ж ні. Саме тому в даному підрозділі ми розкриємо основні моменти дизайну розроблюваного застосунку для відстеження вагітності.

4.1.1 Палітра кольорів

У процесі розробки мобільного застосунку для моніторингу вагітності я приділила особливу увагу вибору кольорової гами. Інтерфейс повинен бути не лише зручним і зрозумілим, а й створювати приємне емоційне враження, яке відповідатиме тематиці застосунку. Основними кольорами, які я обрала для проєкту, стали ніжно-рожевий (рис. 4.1) та бірюзовий (рис. 4.2). Рожевий колір асоціюється з турботою, ніжністю та жіночністю – саме ті якості, які є

невід’ємною частиною періоду вагітності. Він викликає відчуття затишку й спокою, що, на мою думку, є дуже важливим для майбутніх мам, які користуються застосунком щодня.

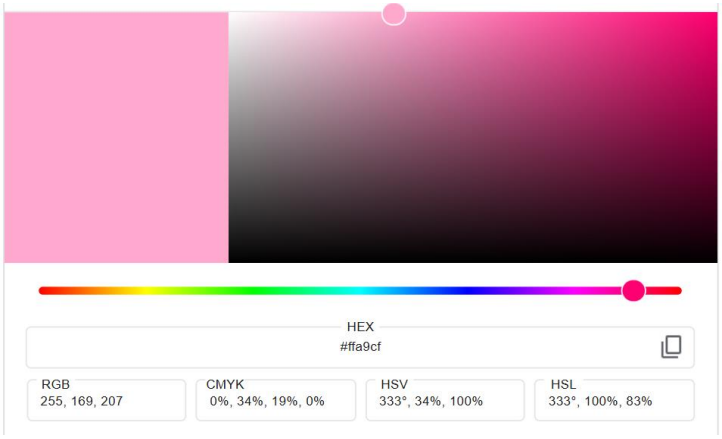

Рисунок 4.1 – Рожевий колір

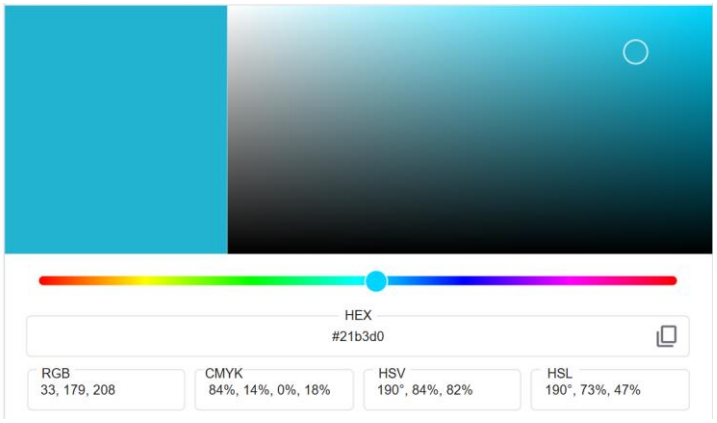

Рисунок 4.2 – Бірюзовий колір

Бірюзовий колір я додала для візуального балансу. Він надає інтерфейсу свіжості, легкості та сучасності, а також добре контрастує з рожевим, завдяки чому допомагає виокремлювати ключові елементи, як-от кнопки чи активні іконки. Обидва кольори використовуються гармонійно та послідовно у всьому дизайні: на сторінках календаря, у формі опитувань, у графіках на сторінці статистики, а також у заголовках і навігаційних панелях. Такий підхід дозволив досягти цілісності інтерфейсу, де візуальні елементи не лише виконують функціональне навантаження, але й підтримують загальну

атмосферу спокою, довіри та комфорту, яку я прагнула передати у цьому застосунку. Також дані кольори дуже часто асоціюються з періодом вагітності, тому я думаю це було чудове рішення.

4.1.2 UX/UI-дизайн мобільного застосунку

Під час створення інтерфейсу я прагнула зробити застосунок максимально зрозумілим і зручним для майбутніх мам. Враховуючи чутливість користувачок до візуального навантаження, я уникала перевантажених екранів і складної навігації. Основні функції: календар, опитування стану, статистика – розміщені на головному екрані або в нижній панелі для швидкого доступу.

Для проєктування використано онлайн-інструмент Figma, який дозволив створити інтерактивні прототипи з урахуванням сучасних принципів UX/UI-дизайну. На рис. 4.3 представлено візуалізацію основних екранів застосунку MotherJorney, включаючи навігацію між розділами: календар вагітності, нотатки, моніторинг стану, опитування та корисні поради.

Я прагнула логічно зв’язати всі функції: основний екран містить меню з переходами до всіх модулів, з календаря можна перейти до створення нотатки або трекерів стану. Кожен розділ має свою стилізацію відповідно до обраної кольорової палітри – рожеві та бірюзові акценти гармонійно підкреслюють функціональні елементи. Наприклад, сторінка трекера настрою оформлена в бірюзових тонах, що асоціюються з легкістю та позитивом, тоді як сторінка з порадами або календар має більш нейтральне, спокійне оформлення.

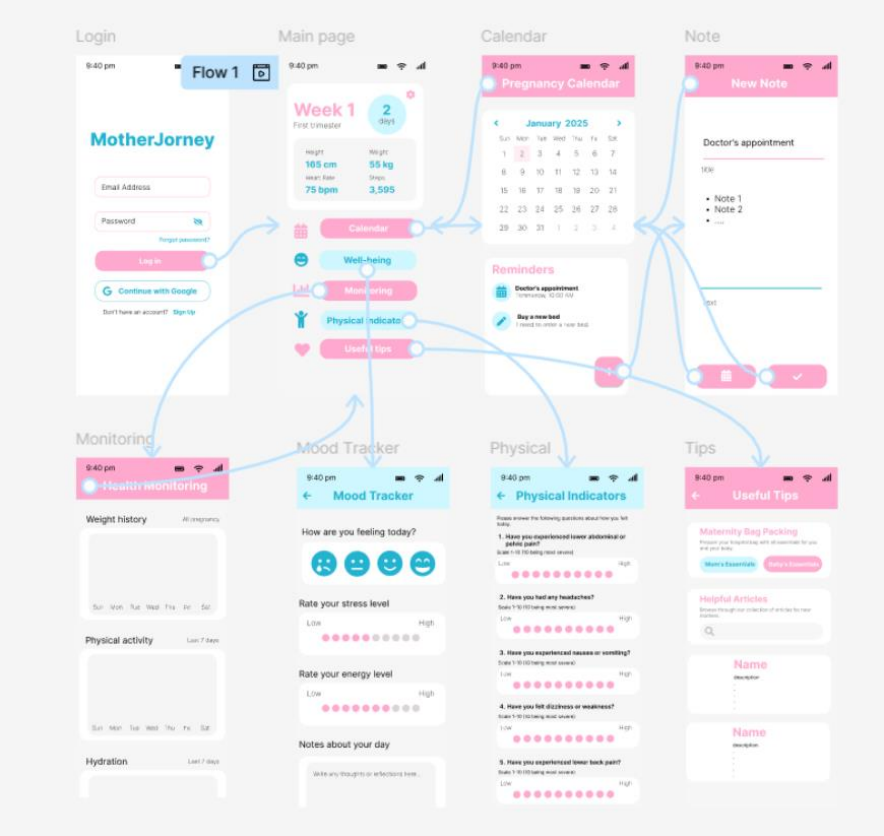

Рисунок 4.3 – Розроблений дизайн за допомогою Figma

Завдяки Figma я мала змогу швидко перевірити зручність навігації, оцінити, чи логічно збудована структура переходів, і протестувати вигляд інтерфейсу на різних етапах взаємодії користувача із застосунком. Цей підхід дозволив до розробки мати чітке бачення всього застосунку й уникнути хаотичності в дизайні.

4.2 Загальна структура проєкту

Працюючи над мобільним застосунком, я намагалась від самого початку продумати чітку та логічну структуру проєкту. Це особливо важливо, коли проєкт зростає, і з’являється потреба додавати нові сторінки, сервіси, ViewModel-и. Щоб уникнути хаосу в коді, я розділила все за принципами MVVM на окремі папки: models, views, viewmodels, services та кілька допоміжних (рис. 4.4).

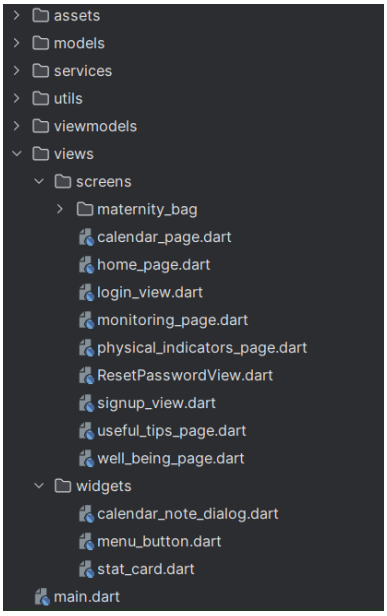

Рисунок 4.4 – Структура проєкту

У models зберігаються класи, що описують дані – наприклад, структура нотатки, фізичного опитування тощо. Вони мають зручні методи для конвертації з та у формат, придатний для Firestore. Це значно полегшує збереження інформації та її подальше використання у ViewModel.

Всі обчислення, фільтрація, збереження і завантаження з БД, а також логіка взаємодії з користувачем винесені у viewmodels. Наприклад, при відповіді на опитування саме ViewModel відповідає за перевірку даних і передачу їх у Firestore. Це дозволяє тримати UI "чистим" – він просто відображає дані, які надає ViewModel.

Найбільша частина проєкту – views. Тут містяться всі екрани застосунку: календар, опитування, поради, реєстрація тощо. Кожен екран має власну структуру, а також деякі елементи винесено у widgets.

Щоб не дублювати код, усю взаємодію з Firebase, SharedPreferences, локальними нагадуваннями та permission'ами я зібрала у services. Це окремі частини коду (класи), які можна просто підключати й використовувати в різних місцях програми.

Крім основних папок, у проєкті є також utils, де наразі зберігається лише

один, але важливий клас – NotificationHelper. Саме він відповідає за ініціалізацію локальних нагадувань, перевірку дозволів на сповіщення та планування повідомлень на конкретний час. Цей клас інкапсулює всю логіку, пов’язану з flutter_local_notifications [11], і забезпечує зручний спосіб виклику повідомлень з будь-якої частини застосунку. Його ізоляція від ViewModel дозволяє уникнути дублювання коду і полегшує підтримку в майбутньому.

Загалом структура проєкту вийшла простою, гнучкою та зручною для підтримки. Я намагалась дотримуватись одних і тих самих принципів у всіх розділах, хоча іноді доводилось адаптувати під специфіку певного функціоналу.

4.3 Використані інструменти та технології

У процесі розробки мобільного застосунку для моніторингу вагітності я обрала сучасні інструменти, що забезпечують зручність, надійність і гнучкість у розвитку. Основою став Flutter – кросплатформовий фреймворк для створення мобільних застосунків Android та iOS, а для бекенду – Firebase з інтеграцією авторизації, хмарного зберігання й повідомлень.

Для керування станом використано Provider, що добре поєднується з архітектурою MVVM. Додаткові бібліотеки забезпечили роботу з календарем, сповіщеннями, графіками та налаштуваннями. У цьому підрозділі розгляну використані інструменти, причини їх вибору та їхній внесок у функціональність застосунку.

4.3.1 Firebase

Firebase став основою для зберігання даних, організації авторизації користувачів та взаємодії з бекендом. На рис. 4.5 представлено діаграму структури бази даних, розроблену для зберігання даних у Firestore.

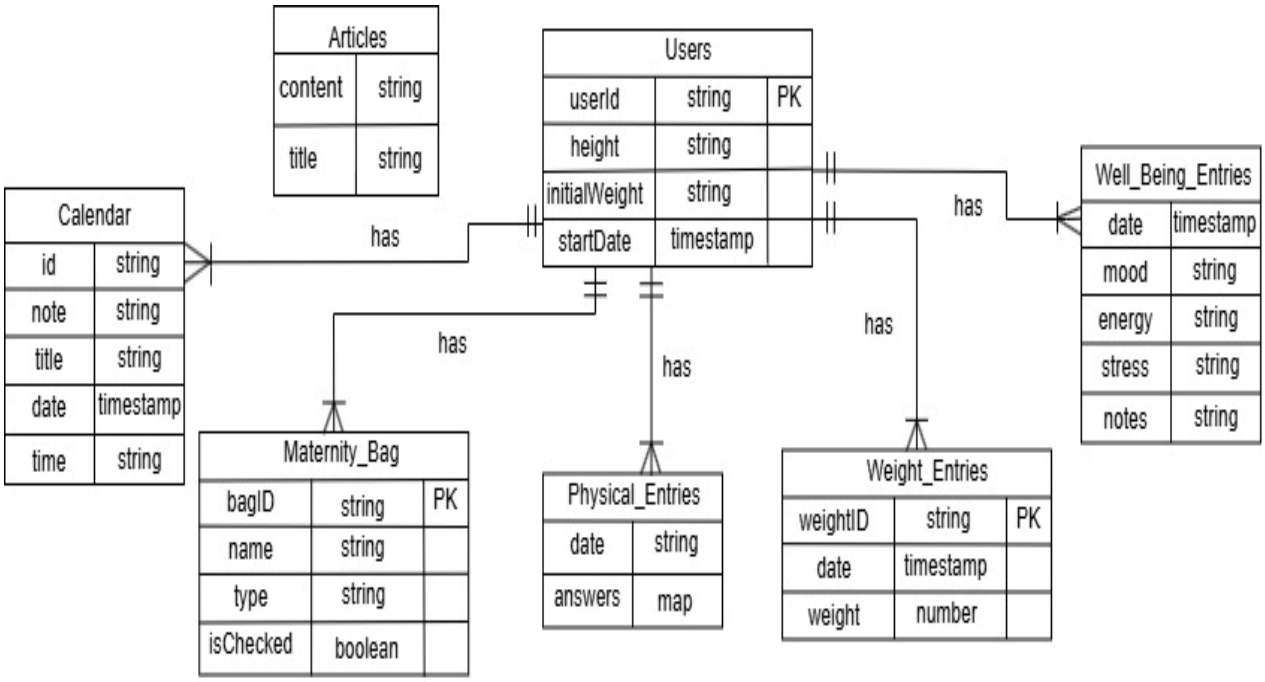

Рисунок 4.5 – Діаграма структури бази даних

Я використала Firestore як хмарну базу даних у режимі реального часу. Це дозволило мені легко зберігати відповіді на опитування, нотатки з календаря, список речей у пологовий будинок і багато іншого (рис. 4.6).

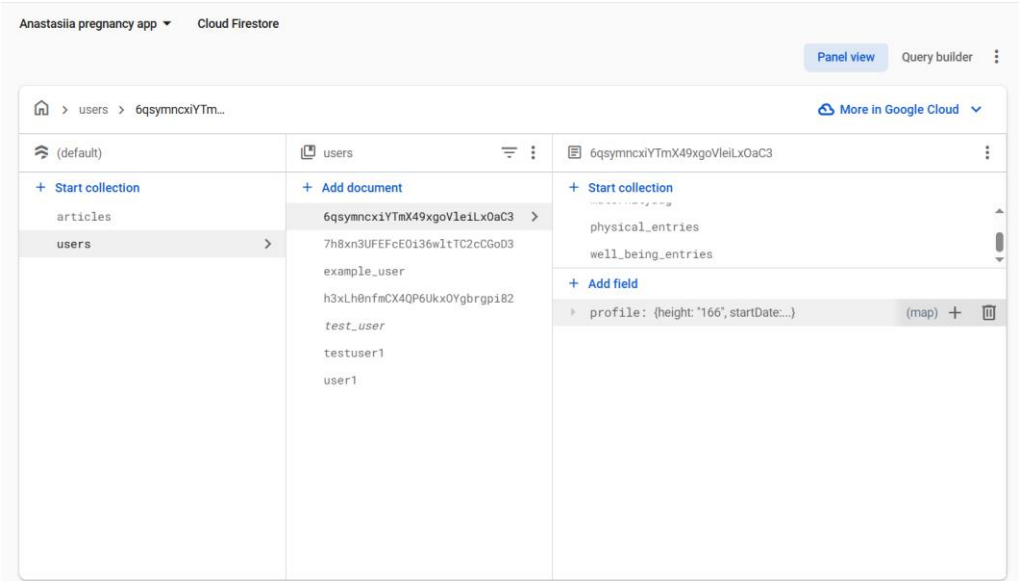

Рисунок 4.6 – Вигляд вікна Firestore

Для авторизації я реалізувала два варіанти входу – за допомогою

електронної пошти та через Google. Це забезпечило зручність для користувачів, які не хочуть створювати новий обліковий запис (рис. 4.7).

Особливо зручним виявився Firebase при роботі з різними типами даних: JSON-структури з Firestore добре поєднуються з моделями у Flutter, тому збереження та читання даних реалізуються швидко і без зайвих труднощів.

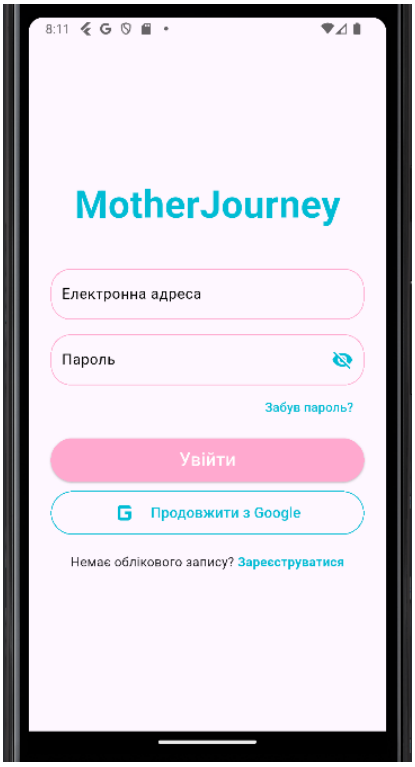

Рисунок 4.7 – Вікно входу в застосунок

4.3.2 Допоміжні бібліотеки

Окрім основних технологій, я також використала низку допоміжних бібліотек, які полегшили реалізацію конкретних функцій (рис. 4.8):

1) table_calendar – дозволяє зручно відображати календар і виділяти дати з нотатками [25].
2) flutter_local_notifications – я використовую її разом із timezone, щоб створювати локальні нагадування на конкретний час. Нагадування працюють і після перезавантаження телефону, що дуже важливо.
3) shared_preferences – для збереження налаштувань або тимчасових

даних, які не повинні потрапляти в БД.

4) fl_chart – дозволяє будувати графіки настрою, активності, гідратації та інших показників. Вона виявилася достатньо гнучкою для мого дизайну.
5) uuid – для генерації унікальних ідентифікаторів нотаток або нагадувань.
6) permission_handler, device_info_plus – для запиту дозволів та перевірки версії Android перед надсиланням сповіщень.

Ці пакети дозволили мені зосередитися на логіці та дизайні, не витрачаючи зайвий час на реалізацію базових речей вручну.

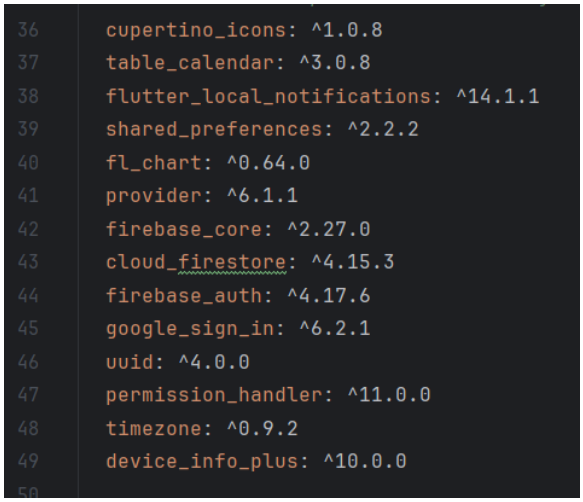

Рисунок 4.8 – Підключені бібліотеки до проєкту

4.4 Реалізація авторизації

Авторизація – одна з базових функцій будь-якого персоналізованого мобільного застосунку. У моєму проєкті MotherJourney я реалізувала гнучку систему входу, яка дозволяє користувачам увійти за допомогою електронної пошти та пароля або через Google-акаунт. Також є можливість реєстрації нового користувача та відновлення пароля в разі його втрати.

Для реалізації архітектури я дотримувався патерну MVVM. Вся логіка

авторизації винесена в окремий сервіс AuthService, який працює безпосередньо з Firebase Authentication [10]. У ViewModel (auth_view_model) містяться всі дані стану: email, пароль, користувач, повідомлення про помилки та індикатор завантаження. Це дозволяє в будь-який момент керувати станом авторизації та реагувати на зміни.

Найперше я підключила Firebase до проєкту за допомогою офіційного FlutterFire CLI. У pubspec.yaml були додані необхідні пакети, а саме firebase_auth, google_sign_in та provider.

Після ініціалізації Firebase я створила сервіс авторизації AuthService, де розмістив основні методи:

- signInWithEmail
- signUpWithEmail
- signInWithGoogle
- signOut

Приклад методу входу через пошту зображено на рис. 4.9.

```
Future<User?> signInWithEmail(String email, String password) async {
  final credential = await _auth.signInWithEmailAndPassword(email: email, password: password);
  return credential.user;
}
```

Рисунок 4.9 – Шматок коду, що відповідає за авторизацію через пошту

Авторизацію через Google вдалося реалізувати доволі швидко, але були нюанси з конфігурацією SHA-1 сертифікату. Після налаштування все працює як слід (рис. 4.10).

```dart
Future<User?> signInWithGoogle() async {
  final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();
  final GoogleSignInAuthentication? googleAuth = await googleUser?.authentication;

  final credential = GoogleAuthProvider.credential(
    accessToken: googleAuth?.accessToken,
    idToken: googleAuth?.idToken,
  );

  final userCredential = await _auth.signInWithCredential(credential);
  return userCredential.user;
}
```

Рисунок 4.10 – Шматок коду, що відповідає за авторизацію через Google

У AuthViewModel реалізовані всі основні дії: вхід, реєстрація, вихід, відновлення пароля, а також зміна стану isLoading та error. Це дозволяє автоматично оновлювати інтерфейс у разі помилки або поки триває авторизація. Наприклад код, що відповідає за обробку логіну зображений на рис. 4.11.

Для сторінки входу я використала стиль, який відповідає кольоровій гамі всього застосунку – ніжно-рожевий з бірюзовим. Пароль можна приховувати/показувати, є кнопка "Забули пароль?" та вхід через Google. Якщо виникає помилка (наприклад, неправильна пошта чи пароль), вона відображається прямо під формою.

```dart
Future<void> loginWithEmail() async {
  try {
    isLoading = true;
    notifyListeners();
    user = await _authService.signInWithEmail(email, password);
    error = null;
  } catch (e) {
    error = e.toString();
  } finally {
    isLoading = false;
    notifyListeners();
  }
}
```

Рисунок 4.11 – Шматок коду, що відповідає за обробку логіну

Форма реєстрації майже ідентична входу (рис. 4.12), але після натискання "Зареєструватися" створюється новий користувач у Firebase. У

випадку успіху користувача перекидає на домашню сторінку (рис. 4.13). Якщо сталася помилка (наприклад, акаунт уже існує), вона також відображається.

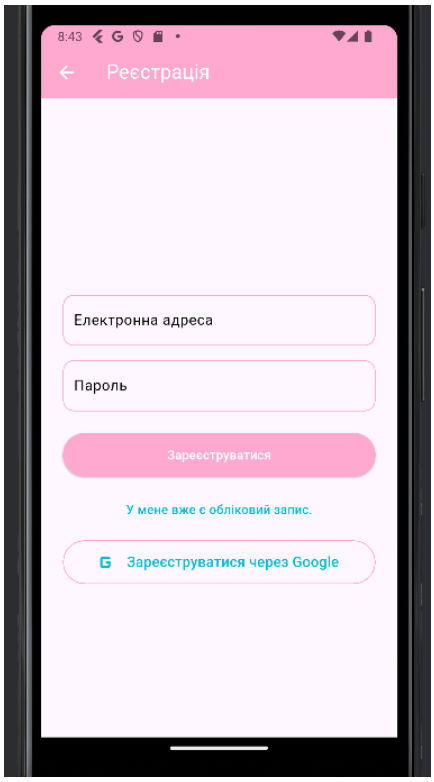

Рисунок 4.12 – Вікно реєстрації

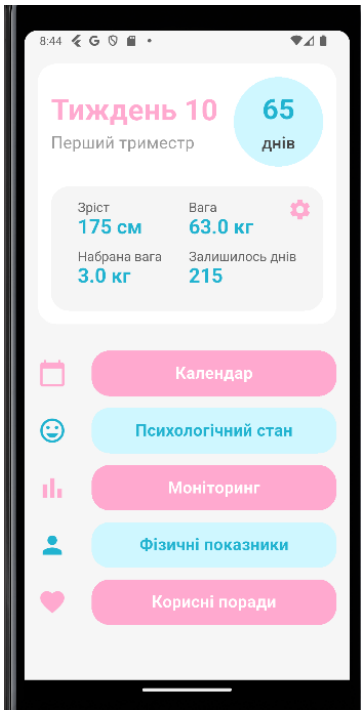

Рисунок 4.13 – Вікно домашньої сторінки

Також є ще одна сторінка – відновлення пароля. Це простий екран, де користувач вводить свою електронну пошту, і якщо така є в базі – отримає лист для скидання пароля (рис. 4.14). Зворотній зв'язок також передбачено – у випадку помилки вона виводиться на екран.

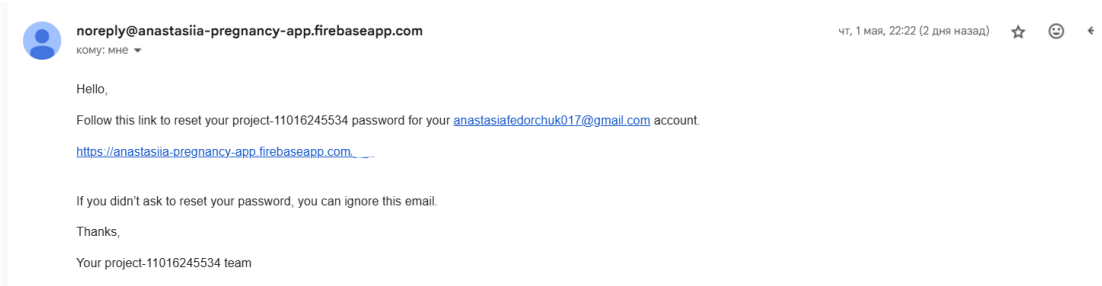

Рисунок 4.14 – Лист для скидання пароля

Таким чином, реалізація авторизації в додатку MotherJourney є повноцінною, зручною та розширюваною. Якщо в майбутньому виникне потреба додати, наприклад, авторизацію через Apple ID чи телефон, архітектура дозволяє це зробити без значного переписування коду.

4.5 Календар та нагадування

Однією з найважливіших функцій мого застосунку став вбудований календар із можливістю додавання нотаток і нагадувань. Завдяки цій функції користувачі можуть зручно планувати обстеження, вести щоденник вагітності, а також отримувати сповіщення про важливі події.

Реалізація цього функціоналу вимагала інтеграції кількох компонентів – календаря, бази даних Firestore, локальних push-сповіщень і власної логіки управління ViewModel. Нижче детально опишу підхід, який я використала.

4.5.1 Вибір і реалізація компонента календар

Для відображення календаря я обрала бібліотеку table_calendar, оскільки вона проста у використанні, має зрозумілу API та дозволяє повністю

кастомізувати вигляд під дизайн застосунку.

На сторінці CalendarPage я додала віджет TableCalendar [25], в якому встановлюється обраний день, стилізуються сьогоднішня дата та вибрані дні (рис. 4.15). Також я обгорнула календар у контейнер з білим фоном і заокругленням, щоб зберегти загальний візуальний стиль програми (рис. 4.16).

```
child: TableCalendar(
  locale: 'uk', // додано
  firstDay: DateTime.utc(2020, 1, 1),
  lastDay: DateTime.utc(2030, 12, 31),
  focusedDay: viewModel.selectedDate,
  selectedDayPredicate: (day) => isSameDay(day, viewModel.selectedDate),
  onDaySelected: (selectedDay, focusedDay) {
    viewModel.selectDate(selectedDay);
  },
  calendarFormat: CalendarFormat.month,
  availableCalendarFormats: const {
    CalendarFormat.month: 'Місяць',
  },
  calendarStyle: CalendarStyle(
    selectedDecoration: BoxDecoration(
      color: Color(0xFF21B3D0),
      shape: BoxShape.circle,
    ),  // BoxDecoration
    todayDecoration: BoxDecoration(
      color: Color(0xFFFFA9CF),
      shape: BoxShape.circle,
    ),  // BoxDecoration
  ),  // CalendarStyle
),  // TableCalendar
```

Рисунок 4.15 – Шматок коду, що відповідає за створення календаря

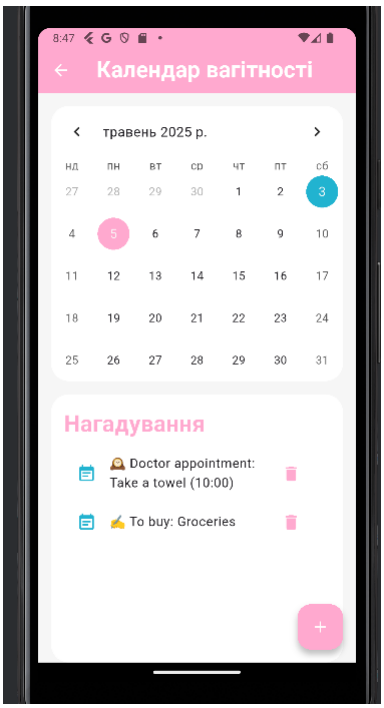

Рисунок 4.16 – Вікно сторінки календаря

4.5.2 Додавання нотаток та нагадувань

Натиснувши кнопку “+”, користувач бачить діалогове вікно, в якому можна ввести заголовок, текст нотатки та за бажанням обрати час для нагадування. Діалог реалізований через власний StatefulWidget – CalendarNoteDialog.

Це діалогове вікно (рис. 4.17) має дві текстові форми та іконку для виклику TimePicker’а (рис. 4.18). Останній я стилізувала у рожевих тонах, що відповідає загальній палітрі застосунку (рис. 4.19).

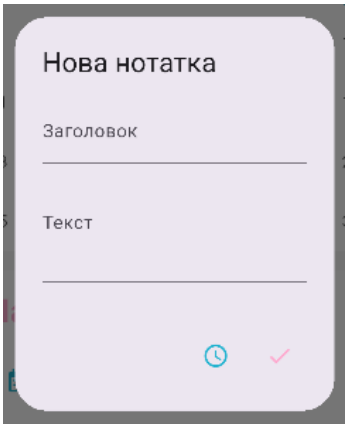

Рисунок 4.17 – Діалогове вікно для створення нотатки

```
builder: (context, child) {
  return Theme(
    data: Theme.of(context).copyWith(
      timePickerTheme: TimePickerThemeData(
        dialHandColor: Color(0xFFFFA9CF),
        dialBackgroundColor: Colors.white,
        hourMinuteColor: Color(0xFFFFE3ED),
        hourMinuteTextColor: Colors.black,
        entryModeIconColor: Color(0xFFFFA9CF),
        backgroundColor: Colors.white,
      ),  // TimePickerThemeData
      colorScheme: ColorScheme.light(
        primary: Color(0xFFFFA9CF),
        onPrimary: Colors.white,
        onSurface: Colors.black,
      ),  // ColorScheme.light
    ),
    child: child!,
  );  // Theme
},
```

Рисунок 4.18 – Шматок коду, що відповідає за стилізацію TimePicker’а

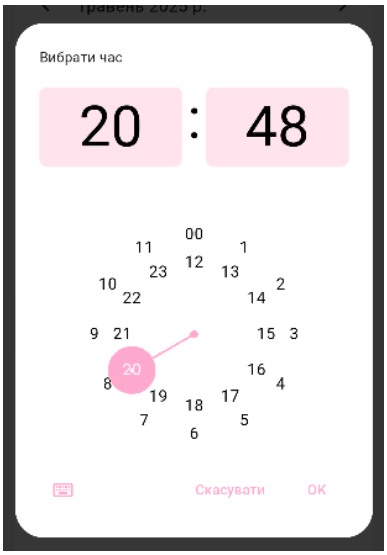

Рисунок 4.19 – Вигляд TimePicker'a

При збереженні нагадування викликається метод ViewModel addReminder(...), який створює об'єкт CalendarReminder, зберігає його у Firestore та, якщо задано час, реєструє локальне сповіщення.

Щоб уникнути надсилання повідомлень у минулому, я реалізувала просту перевірку зображену на рис. 4.20.

```
  if (scheduledDateTime.isAfter(DateTime.now())) {
    await NotificationHelper().scheduleNotification(reminder, scheduledDateTime);
  } else {
    print("Час уже в минулому - нагадування не буде заплановано.");
  }
}
```

Рисунок 4.20 – Шматок коду, що відповідає за перевірку дати

4.5.3 Збереження даних у Firestore

Для зберігання нотаток я створила окрему підколекцію calendar у Firestore для кожного користувача. Об'єкт CalendarReminder серіалізується в карту (Map), яку можна зручно записати у Firestore. Поле TimeOfDay я зберігаю як строку у форматі години:хвилини, а DateTime зберігається як тип Timestamp, який підтримує Firestore (рис. 4.21).

```
Map<String, dynamic> toMap() {
  return {
    'id': id,
    'title': title,
    'note': note,
    'date': date,
    'time': time != null ? '${time!.hour}:${time!.minute}' : null,
  };
}
```

Рисунок 4.21 – Шматок коду, що відповідає за серіалізацію в карту

Щоб отримати всі нагадування на конкретний день, я формую запит, який перевіряє, чи значення поля date лежить у межах доби (рис. 4.22, 4.23).

Видалення реалізоване просто, а саме через doc(id).delete(). Я також додала підтвердження перед видаленням через стандартний AlertDialog.

Таким чином, модуль календаря і нагадувань не тільки допомагає майбутнім мамам краще організовувати свій час, але й інтегрується з Firebase і локальними сповіщеннями, створюючи повноцінний і зручний користувацький досвід.

```
final snapshot = await _calendarCollection
    .where('date', isGreaterThanOrEqualTo: Timestamp.fromDate(start))
    .where('date', isLessThan: Timestamp.fromDate(end))
    .get();
```

Рисунок 4.22 – Шматок коду, що відповідає за запит

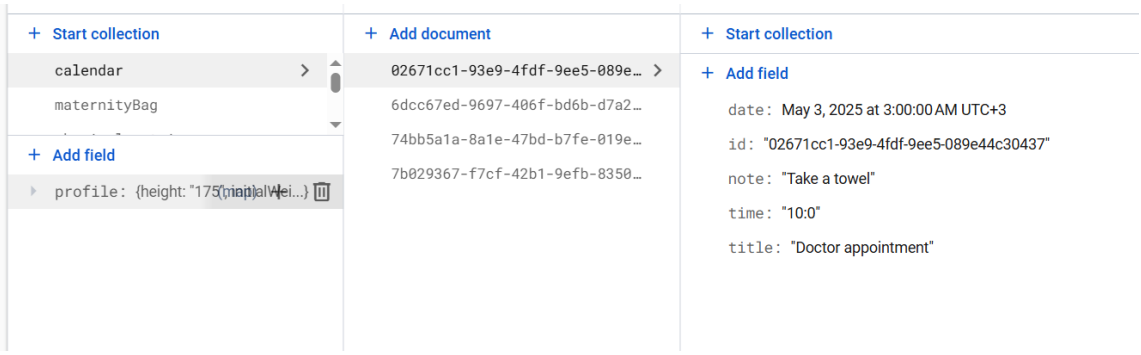

Рисунок 4.23 – Збережена нотатка у Firestore

4.6 Щоденні опитування стану вагітної

Однією з ключових функцій мого застосунку є можливість щоденного моніторингу як фізичного, так і психологічного стану вагітної жінки. Таке опитування дозволяє жінці самостійно відстежувати свій стан, а також може бути корисним для лікаря під час консультацій. Опитування поділені на дві окремі категорії: психологічні показники та фізичні симптоми. Вони з'являються у вигляді інтерфейсу з відповідними шкалами чи перемикачами. Збереження даних відбувається автоматично у хмару через Firebase.

4.6.1 Структура опитувань

Відстежування психологічного стану складається з трьох ключових показників: настрій, рівень стресу та рівень енергії (рис. 4.24). Настрій оцінюється користувачем за шкалою від 1 до 4, що відповідає іконкам-емоціям (від сумного до дуже щасливого). Рівень стресу та рівень енергії оцінюються за шкалою від 1 до 10. Крім того, доступне текстове поле для короткої нотатки, що є дуже корисним. Також реалізована можливість перегляду записів за інші дні.

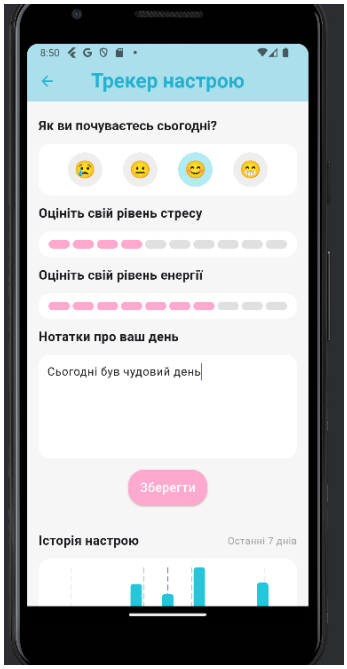

Рисунок 4.24 – Вікно опитування психологічного стану

У моделі опитування психологічного стану використовується клас WellBeingEntry, де дані зберігаються з поточною датою. Код збереження зображено на рис. 4.25.

```
  await _firestore
      .collection('users')
      .doc(userId)
      .collection('well_being_entries')
      .doc(dateKey)
      .set(entry.toMap(), SetOptions(merge: true));
}
```

Рисунок 4.25 – Шматок коду, що відповідає за збереження

Фізичне самопочуття охоплює ширший спектр питань (рис. 4.26). Даний опитувальник заснований на щоденнику самопочуття вагітної з наказу № 417 від 15.07.2011 “Про організацію амбулаторної акушерсько-гінекологічної допомоги в Україні” та додатково був доповнений [1]. Частина оцінюється за шкалою від 1 до 10 (наприклад, біль, головний біль, нудота), інші – як запитання з відповідями так/ні або вибором з трьох варіантів (наприклад, кількість води, рухи дитини). Ось список всіх питань:

1) Чи відчували ви біль внизу живота або таза? (шкала 1–10)
2) Чи були головні болі? (1–10)
3) Чи була нудота або блювота? (1–10)
4) Чи було запаморочення або слабкість? (так/ні)
5) Чи відчували біль у попереку? (1–10)
6) Чи помітили набряки (ноги, руки, обличчя)? (так/ні)
7) Чи були судоми в ногах? (так/ні)
8) Чи було печіння, печія або проблеми з травленням? (1–10)
9) Чи були кров’янисті виділення? (так/ні)
10) Чи були незвичні або рясні виділення? (так/ні)
11) Чи відчували напругу або скорочення матки? (так/ні)
12) Чи була фізична активність (прогулянка, зарядка тощо)? (так/ні)

13) Скільки води сьогодні випили? (вибір: <1л / 1–1.5л / >1.5л)

14) Чи приймали призначені вітаміни або ліки? (так/ні)

15) Чи відчували рухи дитини? (так/ні/ще не відчуваю)

Рисунок 4.26 – Вікно опитування фізичного стану

Для зберігання відповідей опитування фізичного самопочуття використовуються об’єкти PhysicalEntry, а всі відповіді зберігаються як Map<String, dynamic> (рис. 4.27).

```
final entry = PhysicalEntry(date: key, answers: Map.from(answers));
savedEntries[key] = entry;
await _firebaseService.saveEntry(entry);
```

Рисунок 4.27 – Шматок коду, що відповідає за збереження

4.6.2 Обробка відповідей

Після збереження дані відображаються на окремій сторінці. Для кожного дня можна подивитися, які показники були введені, що дозволяє користувачці легко спостерігати за динамікою (рис. 4.28).

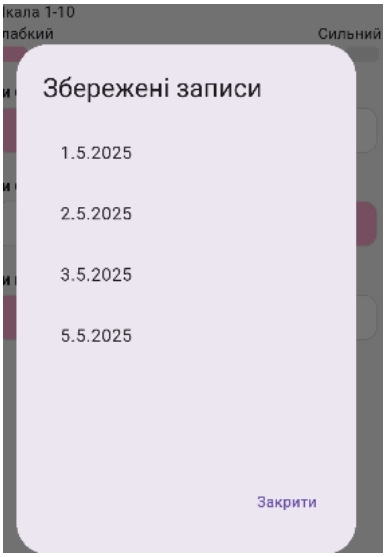

Рисунок 4.28 – Діалогове вікно з відповідями на питання по датах

У ViewModel, при завантаженні перевіряється, чи вже є записи за сьогодні (рис. 4.29).

```
if (moodEntries.containsKey(todayKey)) {
  final entry = moodEntries[todayKey]!;

  selectedMood = entry.mood;
  stressLevel = entry.stress;
  energyLevel = entry.energy;
  note = entry.note;
}
```

Рисунок 4.29 – Шматок коду, що відповідає за перевірку наявності записів

Таким чином, щоденні опитування не лише допомагають жінці краще розуміти свій стан, а й створюють структуру даних, яка з часом може стати основою для рекомендацій або попередження ризиків.

4.7 Сторінка статистики

Одним із найцікавіших і найінформативніших розділів застосунку для мене стала сторінка моніторингу стану користувача. Тут я зібрала графіки, що

відображають динаміку основних фізичних показників вагітної жінки протягом останнього тижня або повного періоду спостереження.

На момент розробки я виділила кілька ключових аспектів, які потребують візуалізації: вага, фізична активність, гідратація, рухи плоду та індекс дискомфорту. Усі графіки були реалізовані з використанням пакету fl_chart, що дозволив створити приємні для ока, адаптивні та досить гнучкі діаграми.

Я вирішила, що найкраще і найінформативніше буде відображати зміни ваги протягом усієї вагітності, тому графік тут будувався на основі повного списку записів користувача. Для цього я просто проходила по списку weights, і кожен елемент перетворювався у точку FlSpot (рис. 4.30).

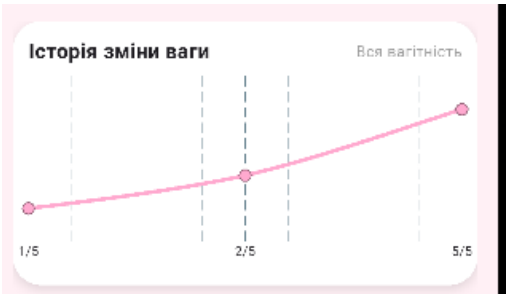

Рисунок 4.30 – Графік зміни ваги

Далі графік фізичної активності (рис. 4.31). Він показує, наскільки часто користувач виконував фізичні вправи протягом останніх 7 днів. Значення записуються у форматі від 0 до 10, залежно від відповіді користувача. Основна логіка побудови точок показана на рис. 4.32.

Рисунок 4.31 – Графік фізичної активності

```
final y = val is int ? val.toDouble() : double.tryParse(val.toString()) ?? 0;
```

Рисунок 4.32 – Шматок коду, що відповідає за логіку побудови точок

Графік гідратації (рис. 4.33) потребував спеціального підходу до кодування відповіді, адже користувач обирав з трьох варіантів:

1) "<1л" -> 0
2) "1–1.5л" -> 1
3) ">1.5л" -> 2

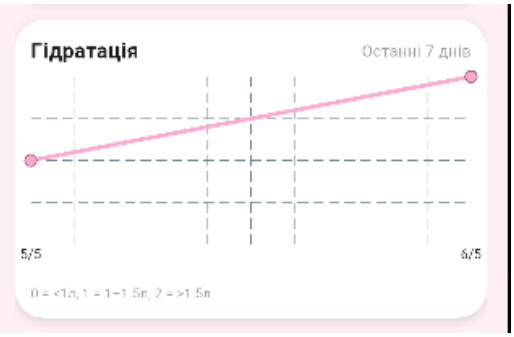

Рисунок 4.33 – Графік гідратації

Один із найважливіших показників – це спостереження за рухами дитини. Тут я вирішила використати числову інтерпретацію строкових відповідей:

1) "Так" -> 2
2) "Ще не відчувала" -> 1
3) "Ні" -> 0

Цей графік також супроводжується легендою, яка допомагає користувачу інтерпретувати дані (рис 4.34).

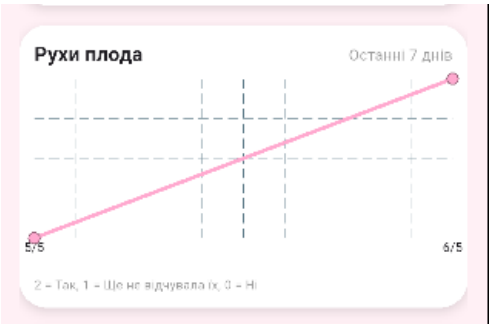

Рисунок 4.34 – Графік рухів дитини

Останній графік – індекс дискомфорту. Цей показник був для мене найскладнішим у реалізації. Він включає в себе шість симптомів:

1) нудота
2) біль у попереку
3) запаморочення
4) судоми ніг
5) напруга в матці
6) набряки

Кожен із них оцінювався за шкалою 0-10, і сумарний показник індексу міг досягати 60. Розрахунок зображено на рис. 4.35.

```
final spots = entries.asMap().entries.map((e) {
  double index = 0;
  for (var key in symptomKeys) {
    final val = e.value.answers[key];
    final parsed = val is int ? val : int.tryParse(val.toString()) ?? 0;
    index += parsed;
  }
```

Рисунок 4.35 – Шматок коду, що відповідає за розрахунок

Якщо індекс > 20, графік підсвічується червоним фоном – своєрідне попередження для користувача звернутись до лікаря (рис. 4.36). Це реалізовано через belowBarData з параметром applyCutOffY.

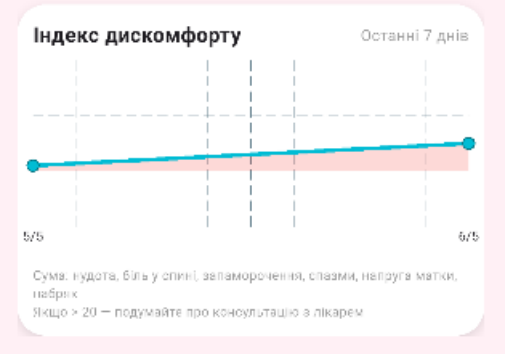

Рисунок 4.36 – Графік індекс дискомфорту

Цей розділ допоміг мені не лише краще зрозуміти стан користувача, а й з технічної точки зору опрацювати багато цікавих підходів до обробки та візуалізації даних. Я намагалась зробити все максимально зрозумілим, щоб навіть людина без технічної підготовки могла отримати корисну інформацію лише з одного погляду на графік.

4.8 Сторінка корисних порад

Однією з цілей мого мобільного додатку було не просто надати майбутній мамі інструменти для контролю стану вагітності, а створити відчуття підтримки, допомоги й організованості у непростий, але прекрасний період життя. Саме тому я додала розділ із корисними порадами. Тут можна знайти все – від списків речей у пологовий для мами і малюка, до збережених статей, які пояснюють нюанси підготовки до пологів або догляду за немовлям.

Я прагнула зробити цю сторінку зручною і зрозумілою навіть для жінок, які раніше ніколи не користувалися подібними застосунками.

4.8.1 Реалізація списку в пологовий для мами та дитини

Ідея створити інтерактивний список речей у пологовий з’явилася, коли я перечитувала історії від вагітних, де вони збирали собі такі сумочки. Хотілося мати зручний чеклист, де можна поставити галочку напроти зібраного предмета, а також додати щось своє, унікальне, наприклад, улюблену заколку чи ароматичну олійку.

У моєму застосунку реалізовано два окремі списки – для мами й для дитини. Кожен користувач при першому вході отримує типовий перелік речей, який можна редагувати: додавати власні пункти або видаляти зайві.

З технічного боку, кожен елемент зберігається у Firebase під користувацьким ідентифікатором. Я додала спеціальне поле type, що дозволяє фільтрувати речі на "mom" і "baby", і це, до речі, значно спростило логіку відображення. Для зручності роботи зі списками я створила сервіс MaternityBagService, який відповідає за додавання, видалення та оновлення елементів, а також початкову ініціалізацію (рис. 4.37).

```
Future<void> addItem(String userId, String type, String name) async {
  await _firestore
      .collection('users')
      .doc(userId)
      .collection('maternityBag')
      .doc(type)
      .collection('items')
      .add({
    'name': name,
    'isChecked': false,
    'type': type,
  });
}
```

Рисунок 4.37 – Шматок коду addItem

Що важливо – якщо користувач відкриває сторінку вперше, автоматично завантажується типовий список. Я спеціально прописала його у вигляді масиву рядків для мами (рис. 4.38) і аналогічно для малюка.

Інтерфейс максимально інтуїтивний (рис. 4.39). Я розмістила чекбокси ліворуч, кнопку видалення праворуч, а нижче додала текстове поле для нових речей. Усе оформлено в рожевих відтінках для затишку і впізнаваності.

```
List<String> _getDefaultMomItems() => [
  'Паспорт',
  'ІПН',
  'Обмінна карта',
  'Нічна сорочка',
  'Капці',
  'Пляшка негазованої води',
  'Вологі та сухі серветки',
  'Телефон і зарядка',
  'Резинка для волосся (без металу)',
  'Післяпологові прокладки',
  'Одноразова або зручна білизна',
  'Бюстгальтер для годування',
  'Прокладки для грудей',
  'Рушники',
  'Засоби гігієни',
  'Столові прибори, чашка, тарілка',
  'Одяг на виписку',
];
```

Рисунок 4.38 – Шматок коду з типовим списком

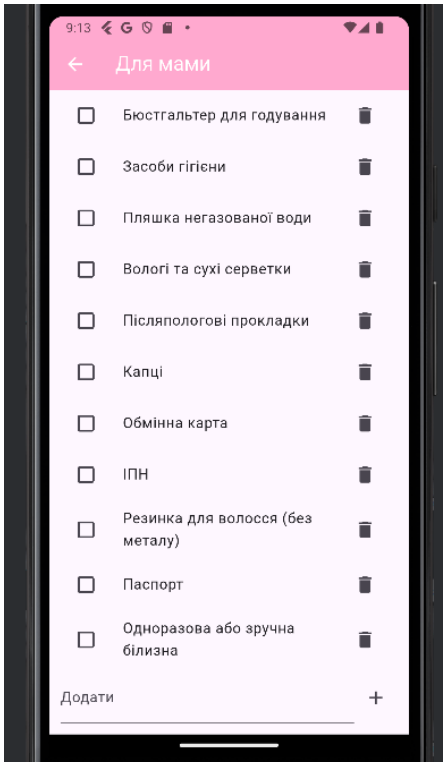

Рисунок 4.39 – Вікно корзини в пологовий для мами

4.8.2 Збереження та відображення корисних статей

Ще з самого початку проєктування мого застосунку я хотіла додати розділ, який би став своєрідною віртуальною бібліотекою для майбутніх матусь. Я вирішила зробити простий і зручний доступ до корисних, і що

важливо медично підтверджених, статей прямо в застосунку.

Звідки беруться статті? Усі статті я зберігаю в БД Firebase Firestore у вигляді колекції articles. Кожна стаття містить два основних поля – title та content. Базова модель, яку я використовую для зберігання зображена на рис. 4.40.

```dart
class Article {
  final String title;
  final String content;

  Article({required this.title, required this.content});
}
```

Рисунок 4.40 – Код базової моделі статті

Для завантаження всіх статей я написала окремий сервіс (рис. 4.41). Метод loadArticles() підключається до Firebase Firestore. Він надсилає запит до колекції articles, а у відповідь повертає список документів. Далі кожен документ перетворюється на об’єкт Article, з якого й будується список для відображення в інтерфейсі.

```dart
class ArticleService {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  Future<List<Article>> loadArticles() async {
    try {
      final snapshot = await _firestore.collection('articles').get();

      return snapshot.docs.map((doc) {
        final data = doc.data();
        return Article(
          title: data['title'] ?? 'No Title',
          content: data['content'] ?? '',
        );
      }).toList();
    } catch (e) {
      print('Помилка завантаження статей: $e');
      return [];
    }
  }
}
```

Рисунок 4.41 – Код сервісу для завантаження статей

Після завантаження дані потрапляють у ViewModel. Там я одразу роблю можливість пошуку по заголовку або тексту статті. Це зручно, коли користувачка хоче швидко знайти, наприклад, щось про грудне вигодовування або підготовку до кесаревого розтину (рис. 4.42). Якщо натиснути на статтю – відкриється її повний текст.

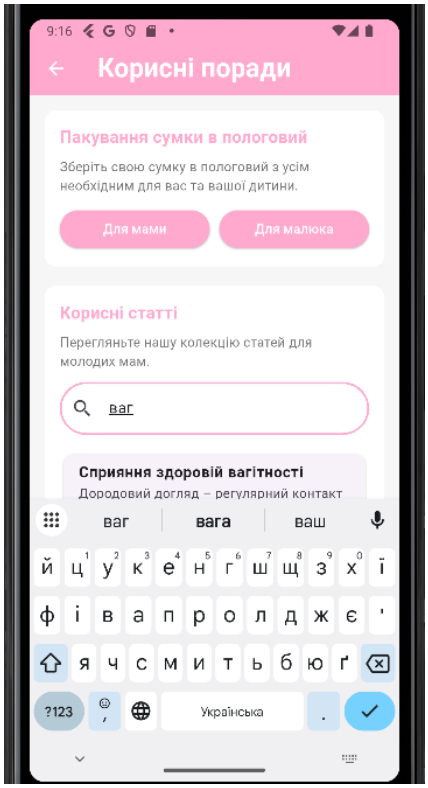

Рисунок 4.42 – Демонстрація роботи пошуку

4.9 Захист інформації користувачів

У застосунку використовуються Firebase Authentication та Firestore, які надають вбудовані механізми безпеки. Авторизація реалізована через електронну пошту або акаунт Google, що дозволяє уникнути зберігання паролів на стороні клієнта. Firebase автоматично забезпечує шифрування даних на сервері та під час передачі через TLS (Transport Layer Security), що виключає можливість їх перехоплення третім особам.

Зберігання особистих даних – таких як нотатки, відповіді на опитування,

календар подій тощо – відбувається у Firestore. Доступ до цих даних обмежено за допомогою правил безпеки Firebase, які перевіряють автентифікацію користувача перед кожною операцією читання чи запису. Наприклад, кожна користувачка має доступ лише до власних записів, що гарантує приватність і виключає несанкціонований доступ.

На клієнтській стороні не відбувається зберігання критичних даних у відкритому вигляді. Тимчасові налаштування, такі як статус входу або обрані налаштування, зберігаються у shared_preferences, але туди не записується конфіденційна інформація. Для автентифікації використовуються токени Firebase, які мають обмежений термін дії та автоматично оновлюються.

Під час створення застосунку дотримувалися принципів безпеки на етапі розробки. У коді реалізовано перевірки введених даних, контроль доступу до функцій, а також захист від потенційних помилок, які можуть порушити безпеку.

4.10 Аналіз економічної ефективності

У цьому розділі було розглянуто ключові підходи, що можуть бути використані при розробці програмного забезпечення для моніторингу фізіологічного стану жінки в період вагітності. Основну увагу було зосереджено на пошуку рішень, які поєднують практичну цінність для майбутніх мам із раціональним використанням ресурсів. Для досягнення оптимального балансу між зручним функціоналом та витратами на його реалізацію, було використано метод функціонально-вартісного аналізу (ФВА). Після виділення з головної функції підфункції, було сворено морфологічну карту зображену на рис. 4.43.

В результаті після аналізу з усіх варіантів було обрано застосування кросплатформенного підходу, орієнтованого на підтримку двох мобільних операційних систем – Android і iOS. Для розробки було обрано мову Dart у поєднанні з фреймворком Flutter, середовищем Android Studio, хмарною базою

даних Firebase, а також використано архітектурний шаблон MVVM для забезпечення розділення логіки та інтерфейсу.

Загальні витрати на реалізацію проєкту за обраним технічним рішенням становлять 332 233,21 гривень.

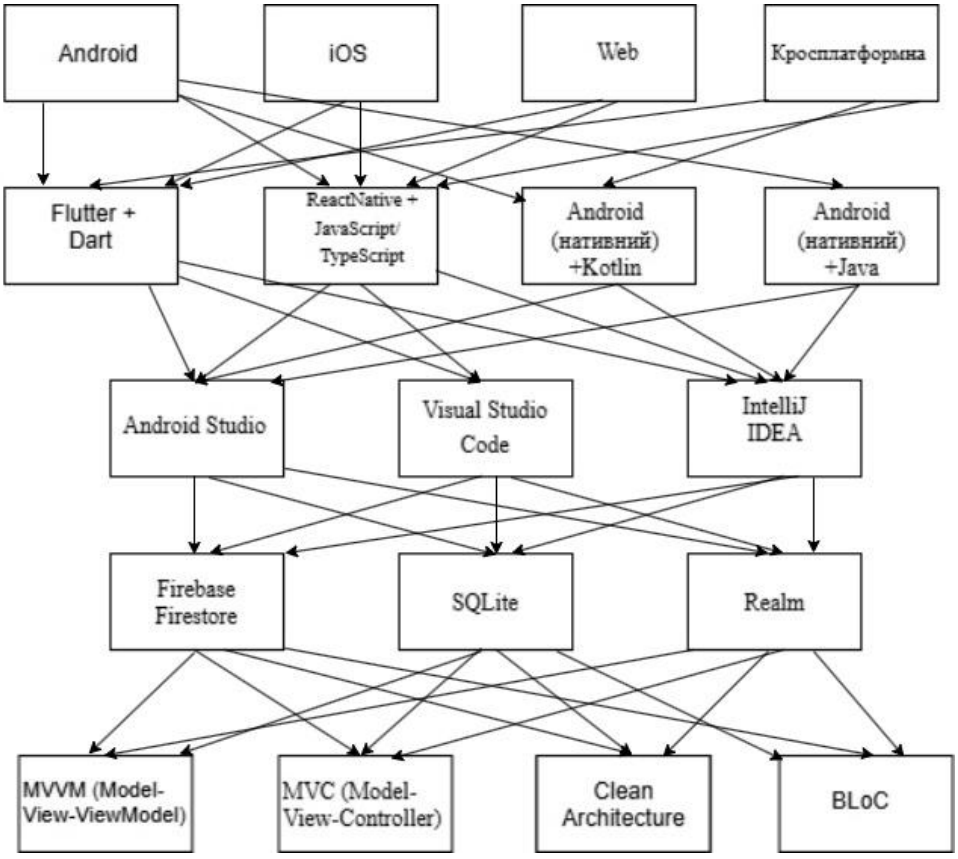

Рисунок 4.43 – Морфологічна карта

Висновок до розділу 4

В даному розділі було описано роботу розробленого мобільного застосунку для підтримки жінок під час вагітності. Застосунок реалізовано з використанням архітектури MVVM, що забезпечує чисту структуру коду й зручне розділення відповідальностей. Основу інтерфейсу побудовано за допомогою Flutter, а для зберігання та обробки даних використано Firebase.

Кожен модуль застосунку має чітко окреслену функцію: від авторизації та календаря з нагадуваннями, до анкетування психологічного стану,

статистики, корисних порад та інтерактивних чеклістів. Значну увагу приділено зручності інтерфейсу, логіці побудови екранів та гнучкості у майбутньому масштабуванні.

Також в розділі було розглянуто ключові заходи безпеки, що були впроваджені під час розробки мобільного застосунку та результати проведеного аналізу його економічної ефективності.

РОЗДІЛ 5
УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ РОБОТИ

У цьому розділі узагальнено основні результати реалізованого проєкту, проведено порівняльний аналіз із наявними аналогами, а також визначено ключові переваги, особливості й перспективи подальшого розвитку мобільного застосунку.

5.1 Порівняння із застосунками-аналогами

Порівнюючи розроблений застосунок з аналогічними мобільними рішеннями для моніторингу вагітності, можна зробити висновок, що хоч він і має менший обсяг функцій, однак орієнтований на корисність функціоналу. У процесі роботи було свідомо обрано підхід мінімалізму, щоб не перевантажувати інтерфейс і забезпечити простоту користування. Замість широкого спектра опцій, часто наявного в комерційних продуктах, основна увага була зосереджена на дійсно корисних і затребуваних функціях, наприклад: зручний календар вагітності, моніторинг психоемоційного та фізіологічного стану, візуалізація змін у вигляді статистики, а також наданні корисної інформації та порад.

Завдяки цьому застосунок не лише виконує свої основні завдання, а й є комфортним для щоденного користування середньостатистичним користувачем. Простий та зрозумілий інтерфейс у пастельних кольорах робить взаємодію приємною, не викликає візуальної втоми. Такий підхід є суттєвою перевагою у порівнянні з громіздкими й перевантаженими інтерфейсами деяких аналогів. Також, що важливо, застосунок реалізовано повністю на українській мові, що є дуже зручно для українських майбутніх матусь.

5.2 Перспективи розвитку

В майбутньому можна додати в застосунок можливість інтеграції зі смарт-годинником або фітнес браслетом для ще більш точного і автоматичного відстеження стану жінки. Крім цього, перспективним є впровадження системи персоналізованих рекомендацій, яка б аналізувала зібрані дані та пропонувала корисні поради відповідно до фізичного і психоемоційного стану користувачки. Також, на мою думку, варто розглянути можливість додавання свого партнера, завдяки чому можна було б ділитись своїми переживаннями і разом проходити цей важливий етап в житті.

Висновок до розділу 5

У даному розділі проаналізовано особливості створеного мобільного застосунку в контексті наявних аналогів, а також розглянуто його основні переваги та перспективи розвитку. Мінімалістичний підхід в розробці дозволив зосередитися на ключових функціях, забезпечити зручність використання та створити можливості для подальшого вдосконалення продукту.

ЗАГАЛЬНІ ВИСНОВКИ

Результати роботи

Результатом даної роботи стала розробка мобільного застосунку для відстеження фізіологічних змін жінки під час вагітності. Всіх цілей було досягнуто, адже застосунок вийшов дуже лаконічний, простий і найголовніше – корисний для майбутніх мам.

Було розроблено такі функції:

1) Календар вагітності – для легкого записування важливої інформації та коротких нотаток, створення нагадувань на окремий час та дату;
2) Опитувальник психологічного стану – для відстеження ментального стану майбутніх мам;
3) Опитувальник фізіологічного стану – для відстеження здоров'я вагітної;
4) Моніторинг – для зручного перегляду змін в організмі як для лікаря, так і для користувачки;
5) Корисні поради – для допомоги в збиранні кошика в пологовий для мами та дитини та для пошуку цікавих і головне корисних статей про вагітність.

Цей мобільний застосунок є особливо актуальним для України, де, з одного боку, рівень цифровізації медицини поступово зростає, а з іншого – жінки часто не мають достатньої кількості зручних, безкоштовних і українськомовних інструментів для повноцінного моніторингу свого стану під час вагітності.

Застосунок не лише спрощує взаємодію з лікарями, а й сприяє зниженню тривожності завдяки регулярному контролю емоційного та фізичного стану.

Зважаючи на складну соціально-економічну ситуацію, а також наслідки війни, підтримка ментального здоров’я майбутніх мам набуває особливої ваги.

Цифровий підхід допомагає зробити цей процес більш зручним, безпечним і таким, що підлаштовується під кожну жінку. Завдяки цьому майбутня мама краще розуміє, що відбувається з її організмом, почувається впевненіше та спокійніше.

Висновки

У результаті проведеної роботи було реалізовано повноцінний мобільний застосунок, який здатен суттєво полегшити щоденне життя вагітних жінок, надаючи їм зручний інструмент для контролю свого фізіологічного та психологічного стану. Простий інтерфейс, інформативність і функціональність роблять застосунок універсальним помічником у такий відповідальний період.

Цей проєкт має практичну цінність і великий потенціал для подальшого розвитку. Його використання може стати корисним інструментом підтримки вагітності та сприяти формуванню уважнішого ставлення до здоров’я майбутніх мам в Україні. Саме тому застосунок зможе стати справді корисним продуктом для багатьох жінок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Наказ № 417 від 15.07.2011 Про організацію амбулаторної акушерсько-гінекологічної допомоги в Україні // ZakonOnline [Електронний ресурс]. – Режим доступу: https://zakononline.com.ua/documents/show/52139___695699 (дата звернення: 19.05.2025).
2) Поляков А. О., Федорченко В. М., Шматко О. В. Аналіз методів і технологій розроблення мобільних додатків для платформи Android : навч. посіб. Харків : ХНЕУ ім. С. Кузнеця, 2017. 286 с.
3) A Guide To Modern Software Architectures: Building Systems for the Digital Age [Електронний ресурс]. – TechWings, 2021. – 59 с. – Режим доступу: https://storage.googleapis.com/techwings-public/ebooks/a-guide-to-modern-software-architectures-building-systems-for-the-digital-age.pdf
4) Announcing .NET mobile Debugging in VS Code. Mobile development in VS Code with Uno Platform or .NET MAUI // Uno Platform [Електронний ресурс]. – Режим доступу: https://platform.uno/blog/announcing-net-mobile-debugging-in-vs-code-mobile-development-in-vs-code-with-uno-platform-or-net-maui/ (дата звернення: 03.05.2025).
5) Brainstein M., Brainstein N. The NativeScript Book: building mobile apps with skills you already have. [s.l.] : The Brosteins, 2018. 481 p.
6) Calcium and bone disorders in pregnancy - PMC // PMC Home [Електронний ресурс]. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC3354840/?utm_source=chatgpt.com (дата звернення: 19.05.2025).
7) Chloe. The ultimate guide to adobe XD tutorials: from beginner to expert // Mockplus - Design, Prototype & Collaborate Better and Faster [Електронний ресурс]. – Режим доступу: https://www.mockplus.com/blog/post/adobe-xd-tutorial (дата звернення: 07.05.2025).

8) Eckel B. Thinking in Java. 4th ed. Stoughton, Massachusetts : Courier, 2006. 1079 p.
9) Fedorenko E. Designing in Figma: the complete guide to designing with reusable components and styles in Figma. Middletown, DE : [s.n.], 2020. 152 p.
10) Firebase Authentication // Firebase [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/auth (дата звернення: 19.05.2025).
11) flutter_local_notifications | Flutter package // Dart packages [Електронний ресурс]. – Режим доступу: https://pub.dev/packages/flutter_local_notifications (дата звернення: 19.05.2025).
12) Full-screen development with Xcode and the Simulator // SwiftLee [Електронний ресурс]. – Режим доступу: https://www.avanderlee.com/workflow/full-screen-xcode-simulator/ (дата звернення: 03.05.2025).
13) Hermes D., Mazloumi N. Building Xamarin.Forms Mobile Apps Using XAML: Mobile Cross-Platform XAML and Xamarin.Forms Fundamentals. New York : Apress, 2019. 445 p.
14) Implementing a self-monitoring application during pregnancy and postpartum for rural and underserved women: A qualitative needs assessment study // PMC Home [Електронний ресурс]. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC9295984/ (дата звернення: 03.05.2025).
15) Jesse. 20 Best Sketch Alternatives for UI/UX Design // Mockplus - Design, Prototype & Collaborate Better and Faster [Електронний ресурс]. – Режим доступу: https://www.mockplus.com/blog/post/sketch-alternatives-ui-ux-design (дата звернення: 03.05.2025)
16) Kim G. Mastering MVVM: A Comprehensive Guide to the Model-View-ViewModel Architecture // DEV Community [Електронний ресурс].

– Режим доступу: https://dev.to/mochafreddo/mastering-mvvm-a-comprehensive-guide-to-the-model-view-viewmodel-architecture-221g (дата звернення: 19.05.2025).

17) Napoli M. L. Beginning Flutter: A Hands-On Guide To App Development. Indianapolis, Indiana : John Wiley & Sons, 2020. 531 p.

18) Dabit N. React Native in Action. Shelter Island, NY : Manning, 2018. 336 p.

19) Natural Physiological Changes During Pregnancy // PMC Home [Електронний ресурс]. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC10964813/ (дата звернення: 19.05.2025).

20) Neuburg M. iOS 12 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics. 5th ed. Sebastopol, CA : O'Reilly Media, 2018. 652 p.

21) Pregnancy apps for self-monitoring: scoping review of the most popular global apps available in Australia // PubMed [Електронний ресурс]. – Режим доступу: https://pubmed.ncbi.nlm.nih.gov/36673768/ (дата звернення: 03.05.2025).

22) Provider | Flutter package // Dart packages [Електронний ресурс]. – Режим доступу: https://pub.dev/packages/provider (дата звернення: 19.05.2025).

23) Sinha S. Beginning Flutter with Dart. Leanpub, 2020. 574 p.

24) Skeen J., Greenhalgh D. Kotlin Programming: The Big Nerd Ranch Guide. Atlanta, GA : Big Nerd Ranch, LLC, 2018. 549 с.

25) table_calendar - Dart API docs // Dart and Flutter packages [Електронний ресурс]. – Режим доступу: https://pub.dev/documentation/table_calendar/latest/ (дата звернення: 19.05.2025).

26) Thyroid Function in Pregnancy and Its Influences on Maternal and Fetal Outcomes // PMC Home [Електронний ресурс]. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC4338651/ (дата звернення: 19.05.2025).