МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ЦУКРОВОГО ДІАБЕТУ»

Рівень «бакалавр»
Галузь знань 12 «Інформаційні технології»,
спеціальність 122 «Комп'ютерні науки»
Виконав здобувач 4 курсу
Фатєєнков Данило Віталійович

Керівник к.т.н., доцент
Чикунов Павло Олександрович

Рецезент к.т.н., доцент
Задерейко Олександр Владиславович

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет кібербезпеки та інформаційних технологій
Кафедра інформаційних технологій
Галузь знань 12 Інформаційні технології
Спеціальність 122 «Комп'ютерні науки»
Назва освітньої програми «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент Н.І. Логінова
_____ «___» _____ 20__ року

ЗАВДАННЯ
на кваліфікаційну (бакалаврську) роботу
здобувачу Фатєєнкову Данилу Віталійовичу

1. Тема робота «Мобільний застосунок для моніторингу цукрового діабету»
   Керівник роботи: к.т.н, доцент Чикунов Павло Олександрович
   затверджені наказом НУ «ОЮА» від «02» квітня 2024 року № 809-06
2. Дата видачі завдання «15» вересня 2023 року
3. Строк подання здобувачем роботи «20» травня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та вибір засобів розробки	20.10.2023	
2	Визначення функціональних вимог до мобільного застосунку	10.11.2023	
3	Проєктування інтерфейсу користувача	30.11.2023	
4	Розробка алгоритмів моніторингу рівня цукру в крові	25.12.2023	
5	Розробка алгоритмів оповіщення та порад щодо керування цукровим діабетом	12.02.2024	
6	Реалізація мобільного застосунку	19.03.2024	
7	Аналіз результатів та висновки	20.04.2024	
8	Оформлення пояснювальної записки	19.05.2024	

Здобувач _____
(підпис) (ім'я прізвище)

Науковий керівник _____
(підпис) (ім'я прізвище)

АНОТАЦІЯ

Фатєєнков Д. В. Мобільний застосунок для моніторингу цукрового діабету. – Кваліфікаційна робота бакалавра за спеціальністю 122 «Комп'ютерні науки», освітньою програмою «Комп'ютерні науки».

Текст кваліфікаційної роботи складається з 60 сторінок основного матеріалу і 61 сторінки загального тексту. Робота має 3 розділи, 48 рисунків, 4 таблиці, 1 додаток та 15 літературних джерел.

Мета роботи – створення мобільного застосунку для моніторингу цукрового діабету, що функціонуватиме як інструмент для користувачів у веденні контролю за рівнем цукру в крові.

Об'єктом дослідження є специфіка розробки мобільного застосунку для моніторингу цукрового діабету та організація його інтерфейсу та функціоналу.

Предметом дослідження є засоби розробки мобільного застосунку для моніторингу цукрового діабету.

У кваліфікаційній роботі розроблено мобільний застосунок для моніторингу цукрового діабету з можливістю маніпулювання рівнем цукру в крові та іншими параметрами здоров'я. Досліджено специфіку розробки мобільних застосунків для медичних потреб та організацію функціоналу та інтерфейсу з використанням технологій файлів формату .XML. Розроблений застосунок може бути використаний не лише для моніторингу цукрового діабету, а й для інших медичних потреб, що вимагають систематичного контролю та аналізу даних здоров'я.

Ключові слова: цукровий діабет, мобільний застосунок, SQLite, XML, моніторинг.

ABSTRACT

Fatieenkov D. V. Mobile Application for Diabetes Monitoring. – Bachelor's qualification thesis in the specialty 122 "Computer Science", educational program "Computer Science".

The text of the qualification thesis consists of 60 pages of main material and 61 pages of total text. The thesis has 3 chapters, 48 figures, 4 tables, 1 appendix, and 15 literary sources.

The purpose of the study is to create a mobile application for monitoring diabetes, which will function as a tool for users to control their blood sugar levels.

The object of the research is the specifics of developing a mobile application for diabetes monitoring and organizing its interface and functionality.

The subject of the research is the means of developing a mobile application for diabetes monitoring.

In the qualification thesis, a mobile application for diabetes monitoring was developed with the ability to manipulate blood sugar levels and other health parameters. The specifics of developing mobile applications for medical needs and organizing functionality and interface using .XML file format technologies were studied. The developed application can be used not only for diabetes monitoring but also for other medical needs that require systematic control and analysis of health data.

Keywords: diabetes, mobile application, SQLite, XML, monitoring.

ЗМІСТ

ВСТУП

У сучасному світі одним із ключових завдань для медичної науки та практики є постійне вдосконалення методів профілактики цукрового діабету. Зростання поширеності захворювання та зміни у зовнішніх умовах створюють потребу у впровадженні новітніх технологій для ефективного контролю цукрового діабету. Розробка мобільних застосунків для оптимального моніторингу цукрового діабету відіграє важливу роль у розвитку медичної практики та науки.

Останні досягнення в медичній сфері відкривають нові можливості для розробки ефективних стратегій моніторингу та профілактики цукрового діабету. Ці стратегії базуються на аналізі різноманітних клінічних та лабораторних даних, зображень, біологічних сигналів та інших параметрів. Використання останніх технологій дозволяє забезпечити не лише точність моніторингу, а й швидкі та доступні методи контролю, що є важливим у вимогливому медичному середовищі.

Об’єктом дослідження є мобільні застосунки для моніторингу здоров'я.

Предметом дослідження є засоби розробки мобільного застосунку для моніторингу цукрового діабету, організація його інтерфейсу та функціоналу.

Мета бакалаврської роботи полягає у створенні мобільного застосунку для моніторингу цукрового діабету.

Для досягнення цієї мети потрібно вирішити такі завдання:

– аналіз мобільних аналогів в контексті профілактики цукрового діабету;

– дослідження наявних систем, що використовують різні методики профілактики цукрового діабету;

– розробка функціоналу для індивідуалізованої діагностики стану здоров'я людини в контексті цукрового діабету;

– створення мобільного застосунку для моніторингу цукрового діабету;

– розробка та впровадження бази даних з використанням SQLite для зберігання та обробки клінічних даних пацієнтів, результатів записів на прийому, заказ препаратів та інших аспектів;

– реалізація бази знань, що включає інформацію для ефективної діагностики та профілактики цукрового діабету.

Методи досліджень для розв’язанні зазначених завдань роботи:

– метод *аналізу* було використано у процесі дослідження мобільних аналогів в контексті профілактики цукрового діабету, а також вибору сучасного засобів розробки;

– методи *аналогій* використовувався при аналізі наявних застосунків із подібним функціоналом задля виявлення популярних тенденцій;

– метод *абстрагування* та *порівняння* використовувалися для аналізу та узагальнення можливостей, загроз, слабких та сильних сторін серед аналогів програмних застосунків для моніторингу здоров'я;

– методи *спостереження*, *пояснень*, *аналізу* було використано під час проєктування архітектури, моделюванні предметної області та розробки мобільного застосунку, як засоби, що допомагають ґрунтовно охарактеризувати та дослідити той чи інший аспект програмної системи.

Практичне значення здобутих результатів полягає в наданні можливості проводити індивідуалізовану діагностику стану здоров'я в контексті цукрового діабету та приймати рішення щодо подальшого контролю і лікування. Потенційні користувачами можуть бути медичні фахівці та пацієнти, які отримують можливість індивідуалізованих рекомендацій і спостереження за своїм здоров'ям. Розроблене програмне забезпечення призначене для аналізу фізіологічних даних та параметрів, зібраних за допомогою мобільного застосунку під час моніторингу рівня цукру в крові на базі клінічних досліджень. Розроблений застосунок може бути використаний не лише для моніторингу цукрового діабету, а й для інших медичних потреб, що вимагають систематичного контролю та аналізу даних здоров'я.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

1.1 Специфіка розробки мобільних застосунків для моніторингу цукрового діабету

Насьогодні цукровий діабет є дуже поширеним захворюванням, кількість яких тільки збільшується. Найважчим завданням для пацієнта є профілактика захворювання, що вимагає регулярного ведення діабетичного щоденника, розрахунку та коригування дозування інсуліну. Використання мобільних застосунків може суттєво допомогти пацієнтам відстежувати свій стан, контролювати хворобу та покращувати стан здоров’я. Кваліфікаційна робота охоплюватиме розробку мобільного застосунку для людей з діабетом. Він надає користувачеві можливість зручно контролювати та керувати своїми медичними даними, маніпулювати рівнем цукру в крові. Буде проведено аналіз системних вимог, огляд наявних рішень у сфері мобільної медицини, спроєктовано архітектуру клієнтської та серверної частин застосунку розглянуто розвиток інтерфейсу застосунку та проаналізовано результати впровадження сервісу [1].

Мобільний застосунок – це програмне забезпечення, призначене для використання на мобільних пристроях чи планшетах [2]. Ці застосунки можуть виконувати різноманітні функції: від ігор та розважальних програм до корисних інструментів, бізнес-застосунків та послуг. Мобільні застосунки можуть бути доступні для завантаження через магазини застосунків, такі як App Store для iOS-пристроїв та Google Play для Android-пристроїв.

Серед мобільних застосунків для моніторингу здоров’я певну нішу займають медичні застосунки для моніторингу цукрового діабету. Це програмні засоби, спрямовані на надання медичних послуг, сприяння здоров'ю та допомогу в медичних процесах для людей, що хворі на діабети 1 та 2 типів [3]. Вони можуть мати різноманітні функції, а саме:

– відстеження за станом здоров'ям: допомагає користувачам вести журнал про рівень цукру в крові, а також ведення записів про прийом лікарських засобів та інше;

– діагностика та консультування: можливість користувачам отримувати медичні поради, аналіз симптомів та порівняння їх попередніми результатами (відстежування тенденції).

1.2 Аналіз наявних аналогів

На торгівельних платформах мобільних застосунків, таких як Play Market та AppStore є чимало аналогів застосунку для моніторингу цукру в крові. Найпопулярнішими є MySugr, Glucose Buddy, Diabetes:M, Diasend.

MySugr – це мобільний застосунок для охорони здоров'я, створений для допомоги людям із діабетом у більш ефективному керуванні своїм станом [4].

Таблиця 1.1 – SWOT-аналіз застосунку MySugr

Strengths	**Weaknesses**
– зручність і доступність: MySugr пропонує інтуїтивний та зручний інтерфейс, що дозволяє користувачам легко вести моніторинг цукрового діабету; – інтеграція з пристроями моніторингу: MySugr може легко інтегруватися з різними пристроями моніторингу глюкози, що дозволяє автоматично записувати дані і підтримувати актуальність інформації.	– обмежена безкоштовна версія: деякі функції доступні лише у преміум-версії, що може створювати обмеження для користувачів, які не готові або не можуть оплатити плату; – підтримка та оновлення: попри широкі можливості застосунку можуть виникати затримки у підтримці та випуску оновлень, що може вплинути на задоволення користувачів.
Opportunities	**Threats**
– розвиток функціоналу: подальше розширення функціоналу застосунку може зробити його більш привабливим для користувачів та конкурентоспроможним на ринку.	– технологічні ризики: поява нових технологій або зміни у вимогах щодо даних можуть вимагати значних інвестицій у технічні оновлення та відповідність стандартам безпеки даних.

Glucose Buddy – це ще один аналог застосунку для мобільних пристроїв, який допомагає людям із діабетом керувати своїм станом, відстежувати рівень цукру в крові, контролювати дієту та фізичну активність [5].

Таблиця 1.2 – SWOT-аналіз застосунку Glucose Buddy

Strengths	**Weaknesses**
– широка база даних: Glucose Buddy має велику базу даних продуктів харчування, що полегшує користувачам ведення дієти та ефективне керування цукровим діабетом; – зручний інтерфейс: застосунок має інтуїтивний і зручний інтерфейс, що покращує загальний досвід користувача і робить його доступним для широкого кола користувачів; – візуалізація даних: Glucose Buddy надає інструменти візуалізації даних, такі як діаграми та графіки, що дозволяють користувачам легко аналізувати свої тенденції глюкози протягом часу і приймати обґрунтовані рішення щодо свого здоров'я.	– проблеми зі стабільністю: деякі користувачі повідомляють про проблеми зі стабільністю застосунку, такі як викидання і помилки, що можуть ускладнити надійність та використання Glucose Buddy; – обмежені можливості настроювання: застосунок може бути обмеженим у певних налаштуваннях, таких як можливість настроювати нагадування та повідомлення згідно з індивідуальними потребами користувачів, що обмежує його адаптивність до різноманітних потреб користувачів.
Opportunities	**Threats**
– покращення стабільності та продуктивності: вирішення проблем зі стабільністю та покращення загальної продуктивності застосунку може значно підвищити задоволення користувачів і їх лояльність, розмістивши Glucose Buddy як провідний вибір серед застосунків для управління цукровим діабетом; – розширення функціоналу: впровадження нових функцій, таких як інтеграція з пристроями або синхронізація даних у реальному часі, може привернути більше користувачів і надати додаткову вартість, збільшуючи конкурентоспроможність застосунку на ринку.	– конкуренція: ринок застосунків для управління цукровим діабетом дуже конкурентний, і Glucose Buddy може стикатися з загрозою втрати частки ринку конкуруючими застосунками, які пропонують схожі або кращі функції; – технологічні ризики: швидкі технологічні зміни можуть застаріти наявні функції Glucose Buddy, якщо застосунок не встигне за умовами змінюватися та адаптуватися до нових тенденцій та інновацій у галузі управління цукровим діабетом.

Diabetes:M – є аналогом, що пропонує численні функції, які дозволяють відстежувати рівень цукру в крові, прийом ліків, харчування та фізичну активність та генерувати детальні звіти для аналізу стану здоров’я [6].

Таблиця 1.3 – SWOT-аналіз застосунку Diabetes:M

Strengths	**Weaknesses**
– широкий функціонал: Diabetes:M має великий набір функцій, включаючи інтеграцію з пристроями моніторингу глюкози, аналіз даних, нагадування та можливість обміну даними з лікарем; – інтеграція з пристроями: застосунок може легко інтегруватися з різними пристроями моніторингу глюкози, що дозволяє автоматично записувати дані та підтримувати актуальність інформації; – персоналізація: Diabetes:M надає можливість персоналізації і налаштування застосунку з урахуванням індивідуальних потреб користувачів.	– складний інтерфейс: деякі користувачі відзначають складність інтерфейсу застосунку, що може призводити до певних труднощів у користуванні; – неоднорідність функцій: деякі функції можуть працювати менш ефективно або бути менш доступними залежно від регіону або типу пристрою.
Opportunities	**Threats**
– покращення інтерфейсу: робота над спрощенням та поліпшенням інтерфейсу може зробити застосунок більш привабливим для користувачів і покращити їхній загальний досвід користування; – розвиток нових функцій: впровадження нових функцій та можливостей, таких як підтримка штучного інтелекту для аналізу даних або інтеграція з застосунками фітнесу, може привернути нових користувачів та збільшити конкурентоспроможність застосунку.	– конкуренція: за наявності багатьох альтернативних застосунків на ринку, Diabetes:M може зазнати конкуренції, що може вплинути на його популярність та користувацьку базу; – технологічні ризики: різке зміни в технологічних стандартах або технічні проблеми можуть ускладнити розробку та підтримку застосунку, що може вплинути на його стабільність та ефективність.

Diasend – це аналог, що забезпечує інтеграцію даних з різних медичних пристроїв і застосунків, дозволяючи користувачам отримувати повний огляд стану здоров’я та аналізувати зібрані дані [7].

Таблиця 1.4 – SWOT-аналіз застосунку Diasend

Strengths	**Weaknesses**
– інтеграція з медичними пристроями: Diasend має можливість інтегруватися з різними медичними пристроями, що дозволяє користувачам легко передавати дані та отримувати детальний аналіз свого стану здоров'я; – персоналізація звітів: застосунок надає можливість користувачам персоналізувати звіти та графіки відповідно до їхніх індивідуальних потреб та вимог; – конфіденційність даних: Diasend забезпечує високий рівень захисту та конфіденційності особистих даних користувачів, що є важливим для багатьох людей з цукровим діабетом.	– проблеми з синхронізацією даних: деякі користувачі повідомляли про проблеми зі синхронізацією даних між застосунком та їхніми пристроями моніторингу, що може призводити до втрати даних або неповної інформації; – обмежений доступ до функцій: деякі функції або звіти можуть бути обмеженими для користувачів без підписки на платний план, що може створювати невдоволеність серед користувачів.

Закінчення табл. 1.4

Opportunities	Threats
– покращення синхронізації даних: робота над вдосконаленням синхронізації даних може підвищити надійність та швидкість передачі інформації між пристроями та застосунком; – розширення функціоналу: додавання нових функцій та можливостей, таких як інтеграція з медичними засобами телемедицини або покращені аналітичні інструменти, може привернути нових користувачів та підвищити конкурентоспроможність застосунку.	– конкуренція: за наявності багатьох альтернативних застосунків на ринку, Diasend може стикнутися з конкуренцією, що може вплинути на його популярність та користувацьку базу; – технічні проблеми: непередбачувані технічні проблеми або зміни в технологічних стандартах можуть стати викликом для застосунку, що може призвести до збоїв у роботі та негативного впливу на користувацький досвід.

Аналіз мобільних застосунків для моніторингу цукрового діабету MySugr, Glucose Buddy, Diabetes:M та Diasend виявив, що всі вони мають значний потенціал для полегшення життя людей з діабетом. Вони надають широкий функціонал для ведення моніторингу глюкози, дієти, активності та інших аспектів управління цією хворобою. Кожен з цих застосунків має свої переваги, такі як зручний інтерфейс, інтеграцію з медичними пристроями, персоналізовані звіти та високий рівень конфіденційності даних.

Однак, у кожного застосунку є свої слабкі сторони: проблеми зі стабільністю, обмеженість безкоштовних версій, проблеми з синхронізацією даних тощо. Ці проблеми можуть вплинути на задоволення користувачів та загальну конкурентоспроможність застосунків.

Дослідження популярних платформ для моніторингу рівня цукру в крові через мобільні застосунки дозволили виявити основні проблеми та зрозуміти можливі напрямки подальшого розвитку. Попри те, що такі застосунки можуть забезпечити зручний спосіб відстеження глюкози в крові, найбільш поширеними проблемами є точність вимірювань, інтеграція з іншими медичними пристроями та системами, а також забезпечення безпеки даних та приватності користувача.

Розуміння цих недоліків є важливим етапом у вдосконаленні мобільних застосунків для моніторингу рівня цукру в крові. З урахуванням цього стає очевидною необхідність постійного вдосконалення та впровадження нових функцій та технологій, спрямованих на покращення точності вимірювань,

інтеграцію з іншими медичними системами та забезпеченням безпеки та конфіденційності даних користувачів. Такі покращення допоможуть зробити мобільні застосунки для моніторингу рівня цукру в крові більш ефективними та доступними для широкого кола користувачів.

1.3 Розробка функціональних вимог

Розробка функціональних вимог для мобільного застосунку для моніторингу цукрового діабету може певні аспекти що унікальні для кожного застосунку. В застосунку бакалаврської роботи основними аспектами є:

- реєстрація:
  - користувачі повинні мати можливість створити обліковий запис за допомогою електронної пошти або інших соціальних мереж;
  - реєстрація може включати створення пароля та налаштування персональних налаштувань, таких як мова і одиниці виміру.
- букінг записів до лікарів:
  - на головному екрані користувач може записатись до певних лікарів на певну дату реальному часі;
- замовлення препаратів:
  - застосунок може включати заказ певних ліків, що специфічні для діабету I й II типів;
  - користувачі можуть переглядати перелік ліків та зробити заказ необхідним для них.
- рекомендації щодо профілактики рівня цукрового діабету:
  - застосунок може надавати персоналізовані рекомендації користувачам, які базуються на даних глюкози;
  - рекомендації можуть стосуватися порад щодо харчування, фізичної активності та інших факторів, що впливають на рівень глюкози.

Ці критерії є важливими для користувачів з цукровим діабетом, оскільки надають інструменти для контролю за станом здоров'я та допомагають зробити

інформативно обґрунтованими та виваженими рішення щодо їхнього харчування і здорового способу життя.

1.4 Засоби розробки

Для розробки мобільного застосунку у сфері моніторингу рівня цукру в крові дійсно важливо вибрати правильні інструменти та мову програмування, що забезпечать ефективність та надійність створюваного програмного забезпечення [8]. Для цього є кілька ключових аспектів, а саме:

- мови програмування:
  - Java/Kotlin (для Android): це найпопулярніші мови для розробки Android-застосунків. Kotlin, зокрема, набуває все більшої популярності завдяки своїй сучасній синтаксису та безпеці;
  - Flutter/Dart: це фреймворк від Google для створення кросплатформних застосунків. Однією з його основних переваг є можливість розробки одразу для Android і iOS, що зменшує витрати часу та ресурсів;
  - React Native (JavaScript): ше один популярний фреймворк для кросплатформної розробки, що дозволяє використовувати JavaScript для створення застосунків, які працюють на обох основних платформах;
- користувацький інтерфейс:
  - XML: використовується в основному для визначення інтерфейсу користувача в Android-застосунках. XML-файли дозволяють структурувати компоненти UI та легко їх налаштовувати;
  - Jetpack Compose (для Android): це сучасний інструмент для створення UI на Android, який дозволяє використовувати декларативний підхід замість XML;
- зберігання та оброблення даних:
  - SQLite: легка база даних, яка часто використовується для зберігання локальних даних у мобільних застосунках. Вона ідеально підходить для застосунків, які потребують швидкого доступу до даних без потреби постійного підключення до інтернету;

- Room (для Android): Абстракція над SQLite, яка спрощує роботу з базами даних у застосунках на Kotlin або Java, забезпечуючи безпечні та ефективні операції з даними.

Вибираючи інструменти та мови програмування для розробки мобільного застосунку для моніторингу рівня цукру в крові, варто враховувати вимоги проєкту, досвід команди розробників та специфіку цільової аудиторії. Завдяки правильному поєднанню технологій, можна створити високопродуктивне, масштабоване та надійне рішення, яке буде зручним і корисним для користувачів..

1.4.1 XML-розмітка

XML-розмітка – це мова розмітки, яка використовується для представлення даних у вигляді структурованого тексту [9]. Однією з основних особливостей є простота та читабельність коду, що полегшує розробку та обслуговування мобільних застосунків для моніторингу рівня цукру в крові.

Основні характеристики та переваги XML-розмітки:

– структурованість: XML дозволяє організувати дані у вигляді структурованих елементів, що сприяє зрозумілості та організованості інформації;

– розширюваність: XML-розмітка дозволяє легко додавати нові елементи та атрибути до документів без потреби зміни старого коду;

– платформонезалежність: XML є платформонезалежним форматом, що означає, що дані можуть бути легко обмінювані між різними платформами та програмами;

– підтримка багатомовності: XML підтримує багатомовні дані, що дозволяє представляти тексти та дані на різних мовах.

Застосування XML-розмітки у мобільних застосунках для моніторингу рівня цукру в крові дозволяє забезпечити чітке та структуроване представлення даних, що є важливим для забезпечення ефективної роботи застосунку та зручного користувацького досвіду.

1.4.2 SQLite

SQLite – це вбудована об'єктно-реляційна система керування базами даних (СКБД), яка призначена для зберігання та обробки даних у мобільних застосунках для моніторингу рівня цукру в крові [10].

Основні характеристики та переваги SQLite:

– простота використання: SQLite легко інтегрується з мобільними застосунками і не вимагає складної конфігурації, що робить його ідеальним вибором для невеликих та середніх проєктів;

– невеликий обсяг: SQLite має малий розмір, що робить його ідеальним для мобільних застосунків з обмеженими ресурсами, такими як мобільні пристрої;

– підтримка стандартів SQL: SQLite використовує стандартний SQL для виконання операцій з базою даних, що полегшує розробку та забезпечує переносимість даних між різними платформами;

– надійність: SQLite відомий своєю надійністю та стабільністю, що робить його гарним вибором для розробки застосунків, які потребують стабільного зберігання даних;

Застосування SQLite в мобільних застосунках для моніторингу рівня цукру в крові дозволяє забезпечити ефективне та надійне зберігання даних пацієнтів без значного впливу на продуктивність та ресурси мобільного пристрою.

1.5 Висновки до розділу 1

У розділі виконано SWOT-аналіз аналогів програмних застосунків для моніторингу здоров'я задля узагальнення можливостей, загроз, слабких та сильних сторін серед них. Тим самим аналіз аналогів допоміг визначити функціональні вимоги створюваного застосунку.

Розробка застосунку для моніторингу рівня цукру в крові вимагає не лише глибокого розуміння медичних аспектів, а й знання сучасних інструментів мобільної розробки. Використання таких технологій, як XML-розмітка та SQLite, надає можливість створити ефективні та масштабовані мобільні застосунки, які можуть відігравати важливу роль у моніторингу та діагностиці стану здоров'я.

2 ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Інформаційна модель предметної області

Мобільний застосунок для профілактики цукрового діабету має модульну, розширювану структуру. В основі його схеми лежить взаємодія різних функціональних аспектів. Застосунок працює так: на клієнтській стороні є обробник подій, що взаємодіє з серверною стороною шляхом відправлення запитів й отримання відповідей. Також на клієнтській стороні присутня логіка, яка відповідає за відображення та обробку даних, а також візуалізацію результатів [11]. Серверна частина має такі компоненти:

Основні функціональні вимоги до застосунку:

– зберігання даних: реалізація можливості введення та зберігання інформації про користувача, картки послуг та товарів й посилань;

– персоналізовані рекомендації: надання користувачеві функцій, що мають сприяти досягненню оптимального контролю рівня цукру в крові та покращенню загального стану здоров'я;

– візуалізація даних: розробка зручного та зрозумілого інтерфейсу для візуалізації даних про картки послуг та товарів й посилань.

2.2 Архітектура застосунку

2.2.1 Структура системи

Розробка мобільного застосунку для профілактики цукрового діабету передбачає використання модульної та розширювальної структури. Ця структура базується на взаємодії, що забезпечує різні функціональні аспекти програми. У застосунку клієнтська сторона включає обробник подій, який здійснює зв'язок з серверною частиною, надсилаючи запити і отримуючи відповіді. Також вона включає клієнтську логіку, яка відповідає за обробку та відображення даних, а також візуалізацію результатів маніпуляцій.

Серверна сторона має такі компоненти:

– маршрутизатори,
– middleware,
– контролери,
– сервіси.

Маршрутизатори встановлюють маршрут для запитів, middleware здійснює додаткову обробку запитів, а контролери відповідають за обробку запитів та взаємодію зі сервісами. Сервіси, у свою чергу, відповідають за логіку бізнес-процесів, обробку даних та взаємодію з базою даних та зовнішніми сервісами. Основним компонентом є SQLiteOpenHelper, який виступає посередником між програмою та СКБД SQLite [12]. Його функціонал дозволяє виконувати операції створення, читання, оновлення та видалення даних з бази даних.

Структурну схему застосунку зазначено на рис. 2.1.

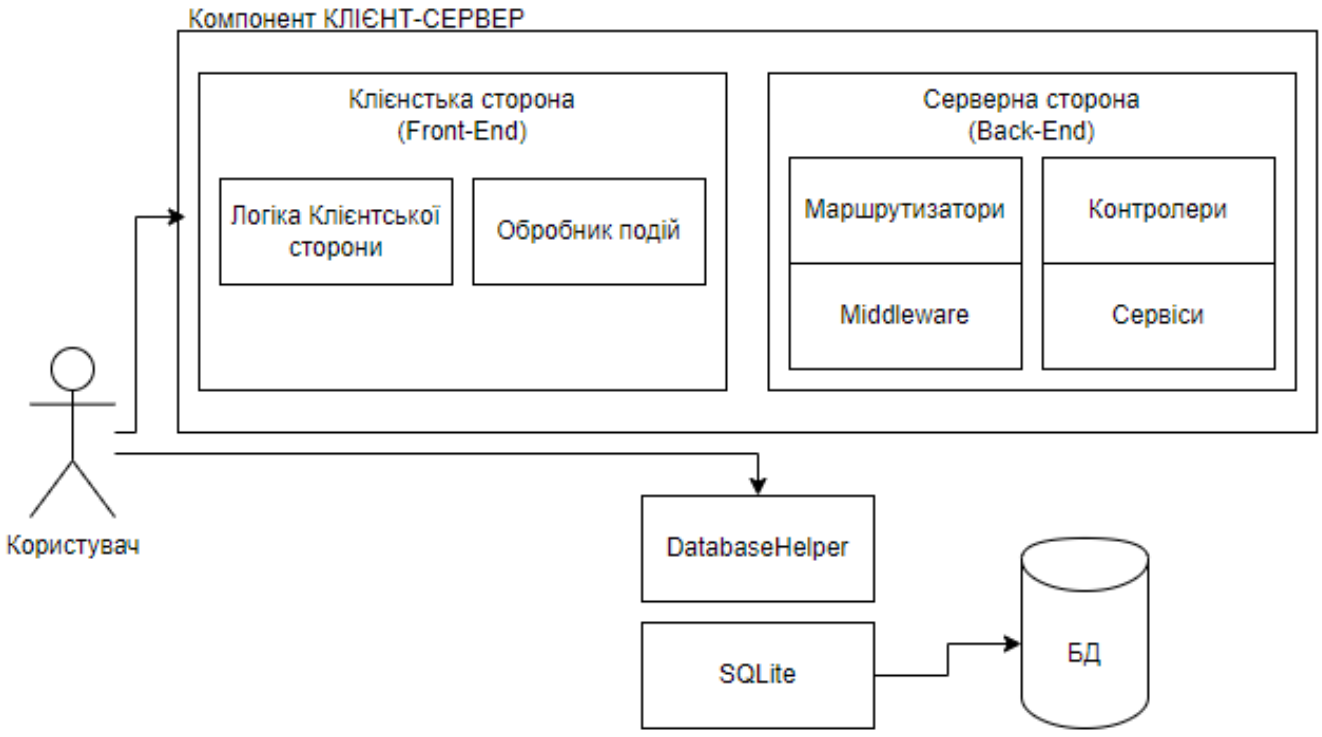


Рисунок 2.1 – Структурна схема застосунку

2.2.2 Use-Case структура застосунку

Use-Case в контексті розробки програмного забезпечення описує специфічні дії або послідовність подій між користувачами або акторами та системою для досягнення конкретної мети [13]. Він дозволяє уявити, як користувач взаємодіє з системою та як система реагує на їхні дії. Use-Case діаграма графічно відображає сценарії використання та взаємодії акторів та системи, відображаючи користувачів, функціональні можливості системи (сценарії) та їх зв'язки. Це надає загальний огляд функціоналу системи та дозволяє краще зрозуміти, які саме дії можуть бути виконані користувачами та як ці дії взаємодіють із системою.

Use-Case діаграма містить такі елементи:

– актори, що є умовними користувачами, які комунікують із системою,

– сценарії виконання, котрі маю себе на увазі як функціонал системи, що репрезентовані як дії, що можуть бути виконані акторами(користувачами),

– зв’язки між акторами та сценаріями, що вказують, яким чином актори комунікують із системою.

Use-Case діаграму вказано на рис. 2.2.

Актори системи:

– пацієнт, що взаємодіє із застосунком для отримання рекомендацій щодо профілактики стану цукрового діабету.

Взаємодія між актором та варіантами використання:

– пацієнт обов’язково має зареєструватися в застосунку, після автентифікації актор має функції отримання направлення до лабораторії чи перегляду все отриманих направлень, запису до лікарів на певну дату та час, чи навпаки, переглянути все наявні записи, можливість купити препарати чи переглянути кошик із вже придбаними товарами й перелік корисних статей для профілактики діабету.

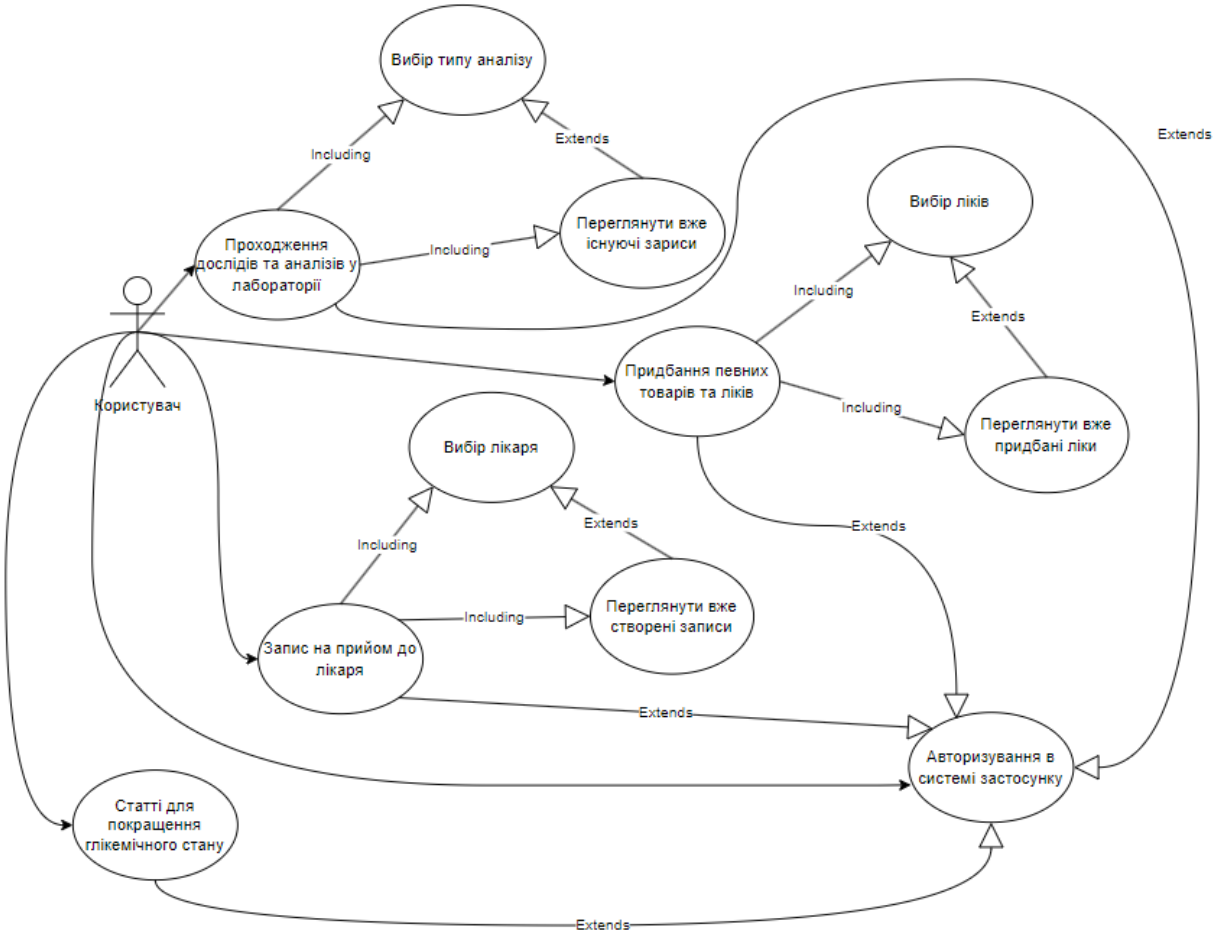


Рисунок 2.2 – Use-case діаграма застосунку

2.2.2 Діаграма діяльності застосунку

Діаграма діяльності – це тип діаграми UML (Unified Modeling Language), яка моделює послідовність дій або процесів у системі [14]. Ця діаграма допомагає уявити послідовність кроків, які виконує користувач у мобільному застосунку для профілактики та рекомендацій для людей, хворих на діабет. В ній відображено послідовність дій, які виконують користувачі під час взаємодії з системою.

Діаграму подано на рис. 2.3.

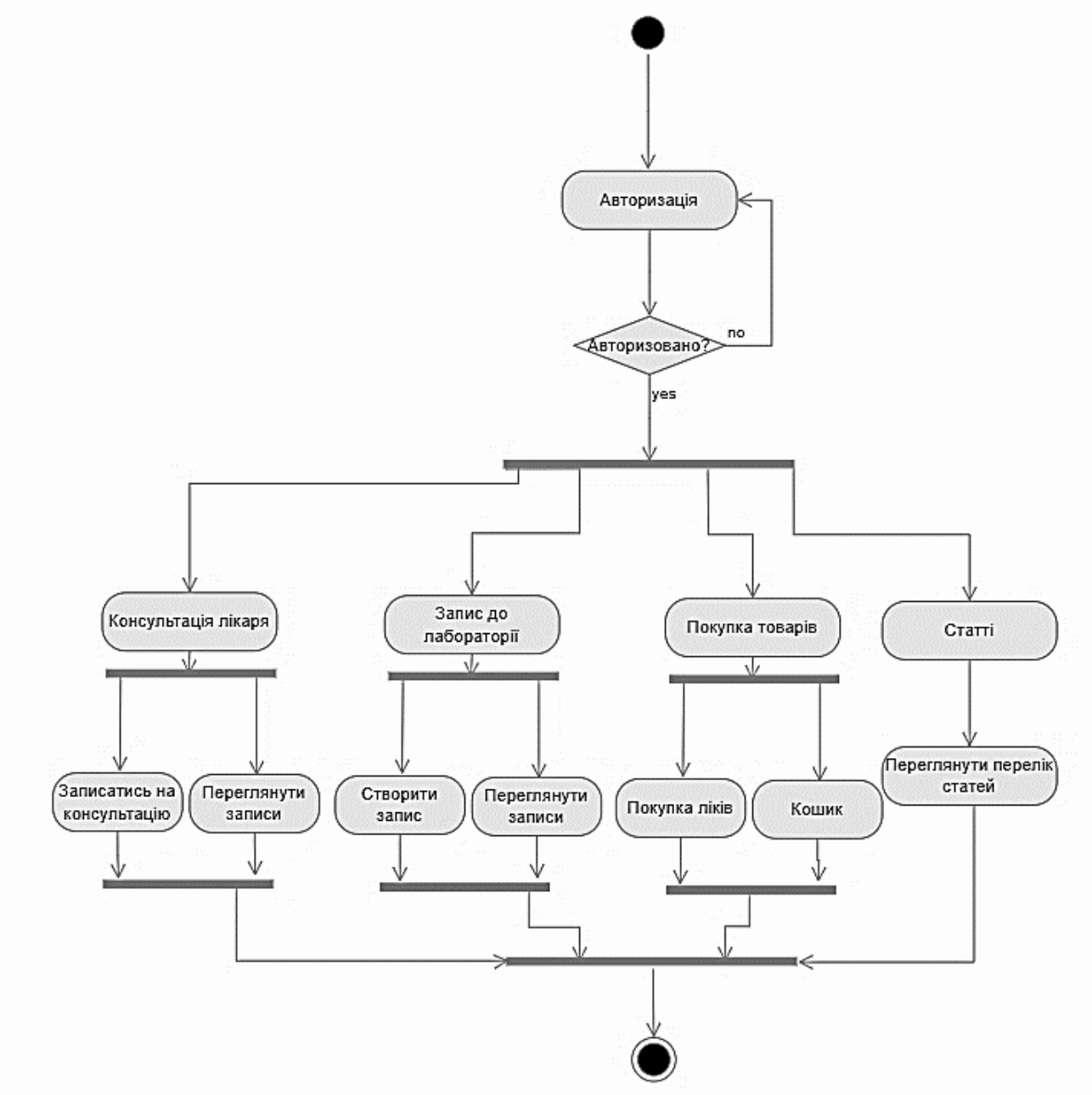


Рисунок 2.3 – Діаграма діяльності застосунку

Авторизація у застосунку є першою дією, яка перевіряється серверною стороною через взаємодію з базою даних для підтвердження успішності. Після цього визначається, який саме користувач авторизувався у системі. Пацієнт може скористатися функціями отримання направлення до лабораторії, перегляду всіх отриманих направлень, запису до лікарів на конкретну дату та час, або ж перегляду всіх наявних записів. Він також може придбати препарати або переглянути кошик із вже придбаними товарами, а також отримати доступ до переліку корисних статей для профілактики діабету.

2.3 Проєктування бази даних

Один з важливих етапів у розробці мобільного застосунку – це створення та організація бази даних. Розпочати з проєктування бази даних має сенс, оскільки це спрощує подальшу розробку серверної частини, дозволяючи здійснювати зручні зв'язки з даними. Це особливо важливо при використанні СКБД, таких як SQLite, де потрібно правильно визначити структуру даних для конкретних сценаріїв використання.

Після успішного реалізації схеми бази даних рекомендується перейти до створення CRUD-операцій (створення, читання, оновлення та видалення). Ці операції є ключовими для взаємодії бази даних з серверною стороною [15].

У мобільному застосунку, призначеному для моніторингу та рекомендацій для людей, які страждають на діабет, можуть бути присутні 3 основних сутності, які представлені у вигляді колекцій: користувач, карта товару та замовлення.

Ця база даних містить всі необхідні дані, з якими серверна сторона застосунку може взаємодіяти за потреби (рис. 2.4).

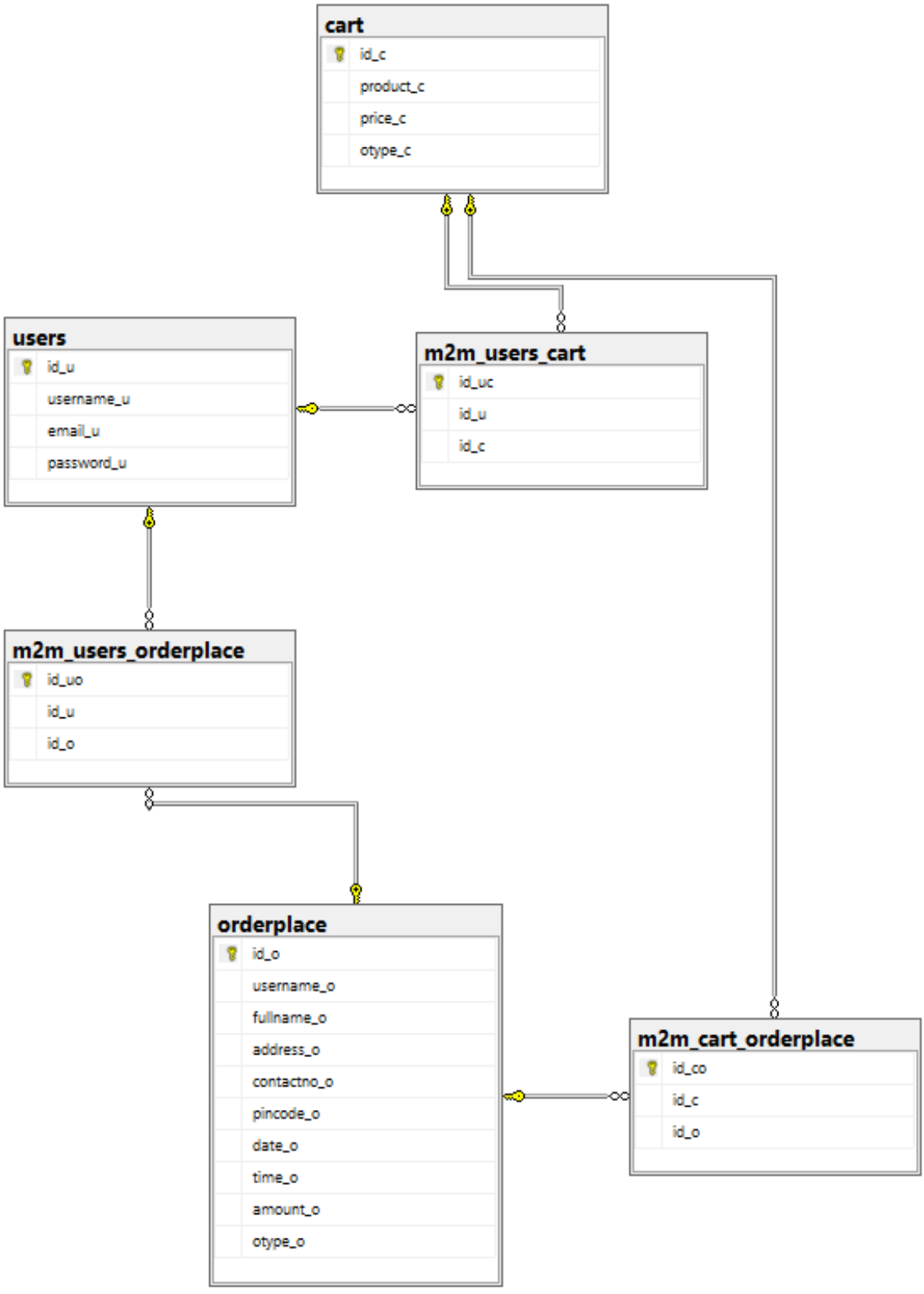


Рис. 2.4 – Діаграма розробленої бази даних

Опис колекцій:

– users: дана колекція зберігає в собі інформацію про користувачів, таку як ПІБ користувача, електронну адресу та зашифрований пароль, необхідні для входу в обліковий запис,

– cart: ця колекція зберігає в собі інформацію про медичні картки продукту чи товару, а саме: назву товару, ціну та тип,

– orderplace: у цій колекції зберігаються показники замовлень на ліки, консультації чи обстеження, а саме: ім’я облікового запису, ПІБ користувача, адреса, контактні реквізити, пін-код замовлення, дата й час, якщо це обстеження чи консультація, кількість – якщо це товар й тип замовлення.

2.4 Висновки до розділу 2

У розділі розглянуто ключові аспекти проєктування застосунку з метою забезпечення ефективного та зручного інструменту для хворих на діабет. Розроблено функціональні вимоги, інтерфейс користувача, методи зберігання та аналізу даних, а також персоналізовані рекомендації для кращого керування рівнем цукру в крові. Описано процес розробки та імплементації мобільного застосунку, спрямованого на полегшення контролю за діабетом і покращення якості життя хворих.

3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Розробка запитів до бази даних відповідно до функціоналу застосунку

3.1.1 Створення таблиць

При першому запуску застосунку після створення та під'єднання бази даних створюються таблиці для користувача (user), карток товарів та послуг (cart) та для замовлення (orderplace) відповідно.

На рис. 3.1 – 3.2 наведено фрагменти коду, що реалізовують створення бази даних та подальше створення таблиць користувача, карток та замовлень.

```
18 usages
public class Database extends SQLiteOpenHelper {
    9 usages
    public Database(@Nullable Context context, @Nullable
    String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
```

Рисунок 3.1 – Створення класу Database, що розширює SQLiteOpenHelper та конструктору ініціалізації бази даних у серверному компоненті бази даних

```
@Override
public void onCreate(SQLiteDatabase db) {
    String qry1 = "create table users(username text, email text, password text)";
    db.execSQL(qry1);

    String qry2 = "create table cart(username text, product text, price float, otype text)";
    db.execSQL(qry2);

    String qry3 = "create table orderplace(username text, fullname text, address text, contactno text, pincode int, date text, time text, amoint float, otype text)";
    db.execSQL(qry3);
}
```

Рисунок 3.2 – Перезапис функції onCreate, де після створення бази даних виконується створення 3 таблиць users, cart та orderplace

Цей фрагмент коду створює три таблиці в базі даних SQLite для зберігання різних типів інформації у серверному компоненті бази даних:

– таблиця users для зберігання даних про користувачів (ім'я користувача, електронна пошта, пароль);

– таблиця cart для зберігання товарів, які користувач додав до свого кошика (ім'я користувача, назва товару, ціна товару, тип замовлення);

– таблиця orderplace для зберігання інформації про замовлення (ім'я користувача, повне ім'я, адреса доставки, контактний номер, поштовий індекс, дата, час, сума замовлення, тип замовлення).

3.1.2 Реєстрація користувачів

Для реєстрації нових користувачів використовується метод, який приймає значення користувача та на їх базі створює відповідний запис у базі даних. Прийматися від користувача мають його ім’я облікового запису, електрона адреса та пароль. Фрагмент коду з серверної та клієнтської частини вказані на рис. 3.3 – 3.4 відповідно.

```
1 usage
public void register(String username, String email, String password){
    ContentValues cv = new ContentValues();
    cv.put("username", username);
    cv.put("email", email);
    cv.put("password", password);
    SQLiteDatabase db = getWritableDatabase();
    db.insert( table: "users",  nullColumnHack: null, cv);
    db.close();
}
```

Рисунок 3.3 – Створення функції для створення запису до таблиці користувача серверного компонента бази даних

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = edUsername.getText().toString();
        String email = edEmail.getText().toString();
        String password = edPassword.getText().toString();
        String confirm = edConfirm.getText().toString();
        if(username.isEmpty() || email.isEmpty() || password.isEmpty() || confirm.isEmpty()){
            Toast.makeText(getApplicationContext(), text: "Please fill All Details", Toast.LENGTH_SHORT).show();
        } else{
            if(password.compareTo(confirm) == 0){
                if(isValid(password)){
                    db.register(username, email, password);
                    Toast.makeText(getApplicationContext(), text: "Record Inserted", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent( packageContext: RegisterActivity.this, LoginActivity.class));
                } else{
                    Toast.makeText(getApplicationContext(), text: "Password must contain at least 8 characters, having letter, digit or special symbol", Toast.LENGTH_SHORT).show();
                }
            }else{
                Toast.makeText(getApplicationContext(), text: "Password and confirm password did not match", Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

Рисунок 3.4 – Використання функції для створення запису до таблиці користувача серверної частини реєстрації

Цей фрагмент коду встановлює обробник подій натискання кнопки (btn). При натисканні кнопки виконується:

1. отримуються значення, які ввів користувач в текстові поля `edUsername`, `edEmail`, `edPassword` та `edConfirm`;

2. перевіряється, чи всі поля заповнені. Якщо хоча б одне поле порожнє, виводиться повідомлення про те, що потрібно заповнити всі поля;

3. перевіряється, чи пароль введений користувачем збігається з підтвердженням пароля. Якщо ні, виводиться повідомлення про те, що пароль і підтвердження пароля не збігаються;

4. перевіряється, чи пароль відповідає вимогам. Якщо ні, виводиться повідомлення про те, які вимоги до пароля не виконані;

5. якщо всі перевірки пройдені успішно, викликається метод `register` бази даних (db), який реєструє нового користувача з введеними даними (ім'ям користувача, електронною поштою та паролем). Після цього виводиться повідомлення про успішне додавання запису, і користувач перенаправляється на екран входу в систему (LoginActivity).

Важливо вказати певні правила створення облікового запису, бо акаунт із паролем з однієї літери не є захищеним належним чином. Тому створено функцію

для коригування пароля для користувача. Поле пароля вимагає довжину не менш, ніж 8 символів та обов’язково літери, цифри та хоча б 1 спеціальний символ.

3.1.3 Автентифікація користувачів

На сторінці входу необхідно викликати функцію, яка надає можливість увійти до щойно створеного чи вже наявного облікового засобу. Для цього у серверному компоненті бази даних необхідно декларувати відповідну функцію.

Фрагменти коду серверної та клієнтської частини наведені на рис. 3.5 – 3.6.

```
1 usage
public int login(String username, String password){
    int result = 0;

    String str[] = new String[2];
    str[0] = username;
    str[1] = password;
    SQLiteDatabase db = getReadableDatabase();
    Cursor c = db.rawQuery( sql: "select * from users where username = ? and password = ?", str);
    if(c.moveToFirst()){
        result = 1;
    }
    return result;
}
```

Рисунок 3.5 – Функція для входу до облікового запису користувача у серверному компоненті бази даних

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = edUsername.getText().toString();
        String password = edPassword.getText().toString();
        if (username.isEmpty() || password.isEmpty()) {
            Toast.makeText(getApplicationContext(), text: "Please fill All Details", Toast.LENGTH_SHORT).show();
        } else {
            if(db.login(username, password) == 1){
                Toast.makeText(getApplicationContext(), text: "Login Success", Toast.LENGTH_SHORT).show();
                SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("username", username);
                editor.apply();
                startActivity(new Intent( packageContext: LoginActivity.this, HomeActivity.class));
            }else{
                Toast.makeText(getApplicationContext(), text: "Invalid Username or Password", Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

Рисунок 3.6 – Функція для входу до облікового запису користувача у серверному компоненті входу у запис

Цей фрагмент коду встановлює обробник подій натискання кнопки (btn). При натисканні кнопки виконується таке:

1. отримуються значення, які ввів користувач в текстові поля `edUsername` та `edPassword`;

2. перевіряється, чи всі поля заповнені. Якщо хоча б одне поле порожнє, виводиться повідомлення про те, що потрібно заповнити всі поля;

3. викликається метод `login` бази даних (db) для перевірки користувача за введеним ім'ям користувача та паролем. Якщо метод повертає значення 1, це означає успішний вхід користувача. У цьому випадку зберігається ім'я користувача у спільних налаштуваннях (SharedPreferences), виводиться повідомлення про успішний вхід і користувач перенаправляється на екран домашнього вмісту (HomeActivity);

4. якщо метод `login` повертає будь-яке інше значення, це означає неправильне ім'я користувача або пароль, тому виводиться повідомлення про це.

3.1.4 Керування кошиком

Для маніпуляцій із картками товарів та послугами необхідно створити певний перелік функцій для кожної операції, серед яких і додавання до кошика (рис. 3.7 – 3.8).

```
2 usages
public void addCart(String username, String product, float price, String otype){
    ContentValues cv = new ContentValues();
    cv.put("username", username);
    cv.put("product", product);
    cv.put("price", price);
    cv.put("otype", otype);
    SQLiteDatabase db = getWritableDatabase();
    db.insert( table: "cart",  nullColumnHack: null, cv);
    db.close();
}
```

Рисунок 3.7 – Створення функції для додавання картки послуги або товару до кошика облікового запису користувача у серверному компоненті бази даних

```
btnAddToCart.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
        String username = sharedPreferences.getString( key: "username",  defValue: "").toString();
        String product = tvPackageName.getText().toString();
        float price = Float.parseFloat(it.getStringExtra( name: "text3").toString());

        Database db = new Database(getApplicationContext(),  name: "Dia-Check",  factory: null,  version: 1);
        if(db.checkCart(username, product) == 1){
            Toast.makeText(getApplicationContext(),  text: "Product Already Added", Toast.LENGTH_SHORT).show();
        }else{
            db.addCart(username, product, price, otype: "lab");
            Toast.makeText(getApplicationContext(),  text: "Record Inserted to Cart", Toast.LENGTH_SHORT).show();
            startActivity(new Intent( packageContext: LabTestDetailsActivity.this, LabTestActivity.class));
        }
    }
});
```

Рисунок 3.8 – Використання функції для додавання картки послуги або товару до кошика облікового запису користувача у серверному компоненті картки послуг

Цей фрагмент коду встановлює обробник подій натискання кнопки (btnAddToCart). При натисканні кнопки виконується таке:

1. отримується ім'я користувача збережене у спільних налаштуваннях (SharedPreferences);
2. отримується назва товару, яка відображається на екрані (tvPackageName);
3. отримується ціна товару, яка передається як додаткова інформація через інтент (з текстом "text3");
4. ініціалізується об'єкт бази даних (Database) із зазначенням контексту, назви бази даних та версії;
5. перевіряється, чи вже є даний товар у кошику користувача. Якщо так, виводиться повідомлення про те, що товар вже додано до кошика.

Якщо товар ще не додано до кошика, викликається метод addCart бази даних для додавання товару до кошика користувача з вказаною інформацією (ім'я користувача, назва товару, ціна товару, тип замовлення). Після цього виводиться повідомлення про успішне додавання запису до кошика, і користувач перенаправляється на екран зі списком лабораторних тестів (LabTestActivity).

Функції для перевірки на наявність товарів у кошику наведено на рис. 3.9 – 3.10.

```
2 usages
public int checkCart(String username, String product){
    int result = 0;
    String str[] = new String[2];
    str[0] = username;
    str[1] = product;
    SQLiteDatabase db = getReadableDatabase();
    Cursor c = db.rawQuery( sql: "select * from cart where username = ? and product = ?", str);
    if(c.moveToFirst()){
        result = 1;
    }
    db.close();
    return result;
}
```

Рисунок 3.9 – Створення функції для перевірки на наявність картки послуги або товару до кошика облікового запису користувача у серверному компоненті бази даних

```
btnAddToCart.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
        String username = sharedPreferences.getString( key: "username",  defValue: "").toString();
        String product = tvPackageName.getText().toString();
        float price = Float.parseFloat(it.getStringExtra( name: "text3").toString());

        Database db = new Database(getApplicationContext(),  name: "Dia-Check",  factory: null,  version: 1);
        if(db.checkCart(username, product) == 1){
            Toast.makeText(getApplicationContext(),  text: "Product Already Added", Toast.LENGTH_SHORT).show();
        }else{
            db.addCart(username, product, price,  otype: "lab");
            Toast.makeText(getApplicationContext(),  text: "Record Inserted to Cart", Toast.LENGTH_SHORT).show();
            startActivity(new Intent( packageContext: LabTestDetailsActivity.this, LabTestActivity.class));
        }
    }
});
```

Рисунок 3.10 – Використання функції для перевірки на наявність картки послуги до кошика облікового запису користувача у серверному компоненті картки послуг

Функції для видалення *товару* з кошика наведено на рис. 3.11-3.12.

```
2 usages
public void removeCart(String username, String otype){
    String str[] = new String[2];
    str[0] = username;
    str[1] = otype;
    SQLiteDatabase db = getReadableDatabase();
    db.delete( table: "cart",  whereClause: "username = ? and otype = ?", str);
    db.close();
}
```

Рисунок 3.11 – Створення функції для видалення картки послуги або товару до кошика облікового запису користувача у серверному компоненті бази даних

```
btnBooking.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
        String username = sharedPreferences.getString( key: "username",  defValue: "").toString();

        Database db = new Database(getApplicationContext(),  name: "Dia-Check",  factory: null,  version: 1);
        db.addOrder(username, edname.getText().toString(), edaddress.getText().toString(), edcontact.getText().toString(),
                Integer.parseInt(edpincode.getText().toString()), date.toString(), time.toString(), Float.parseFloat(price[1].toString()),  otype: "lab");
        db.removeCart(username,  otype: "lab");
        Toast.makeText(getApplicationContext(),  text: "Your booking is done successfully", Toast.LENGTH_SHORT).show();
        startActivity(new Intent( packageContext: LabTestBookActivity.this, HomeActivity.class));
    }
});
```

Рисунок 3.12– Використання функції для видалення картки послуги до кошика облікового запису користувача у серверному компоненті картки послуг

Цей фрагмент коду встановлює обробник подій натискання кнопки (btnBooking). При натисканні кнопки виконується таке:

1. отримується ім'я користувача збережене у спільних налаштуваннях (SharedPreferences),
2. ініціалізується об'єкт бази даних (Database) із вказанням контексту, назви бази даних та версії,
3. викликається метод `addOrder` бази даних для додавання замовлення з вказаною інформацією (ім'я користувача, повне ім'я, адреса, контактний номер, поштовий індекс, дата, час, сума замовлення, тип замовлення),
4. викликається метод `removeCart` бази даних для видалення товарів з кошика користувача після оформлення замовлення,
5. виводиться повідомлення про успішне оформлення замовлення,
6. користувач перенаправляється на екран домашнього вмісту (HomeActivity).

Функції для отримання даних про товари у кошику наведено на рис. 3.13 – 3.14.

```
2 usages
public ArrayList getCartData(String username, String otype){
    ArrayList<String> arr = new ArrayList<>();
    SQLiteDatabase db = getReadableDatabase();
    String str[] = new String[2];
    str[0] = username;
    str[1] = otype;
    Cursor c = db.rawQuery( sql: "select * from cart where username = ? and otype = ?", str);
    if(c.moveToFirst()){
        do{
            String product = c.getString( columnIndex: 1);
            String price = c.getString( columnIndex: 2);
            arr.add(product + "$" + price);
        }while (c.moveToNext());
    }
    db.close();
    return arr;
}
```

Рисунок 3.13 – Створення функції для перегляду інформації про картку послуги або товару до кошика облікового запису користувача у серверному компоненті бази даних

```
Database db = new Database(getApplicationContext(), name: "Dia-Check", factory: null, version: 1);

float totalAmount = 0;
ArrayList dbData = db.getCartData(username, otype: "lab");
//Toast.makeText(getApplicationContext(),"" + dbData, Toast.LENGTH_SHORT).show();

packages = new String[dbData.size()][];
for(int i = 0; i < packages.length; i++){
    packages[i] = new String[5];
}

for(int i = 0; i < dbData.size(); i++){
    String arrData = dbData.get(i).toString();
    String[] strData = arrData.split(java.util.regex.Pattern.quote( s: "$"));
    packages[i][0] = strData[0];
    packages[i][4] = "Cost : " + strData[1] + "/-";
    totalAmount = totalAmount + Float.parseFloat(strData[1]);
}
```

Рисунок 3.14– Використання функції для перегляду інформації про картку послуги до кошика облікового запису користувача у серверному компоненті картки послуг

Цей фрагмент коду виконує такі дії:

1. ініціалізує об'єкт бази даних (Database) з використанням контексту застосунку, назви бази даних та версії;

2. отримує дані з кошика користувача для медичних товарів за допомогою методу getCartData;

3. створює масив для зберігання даних про медичні товари, який міститиме різні атрибути для кожного товару;

4. проходиться по кожному запису у результатах бази даних та розбиває його на окремі частини за допомогою методу split. Після цього витягує ім'я товару та його ціну;

5. розраховує загальну суму всіх товарів, додаючи їх ціни до змінної totalAmount.

Отримані дані про товари зберігаються у масиві packages. Кожен елемент масиву packages містить дані про один товар: назву, ціну та інші атрибути.

3.1.5 Обробка замовлень

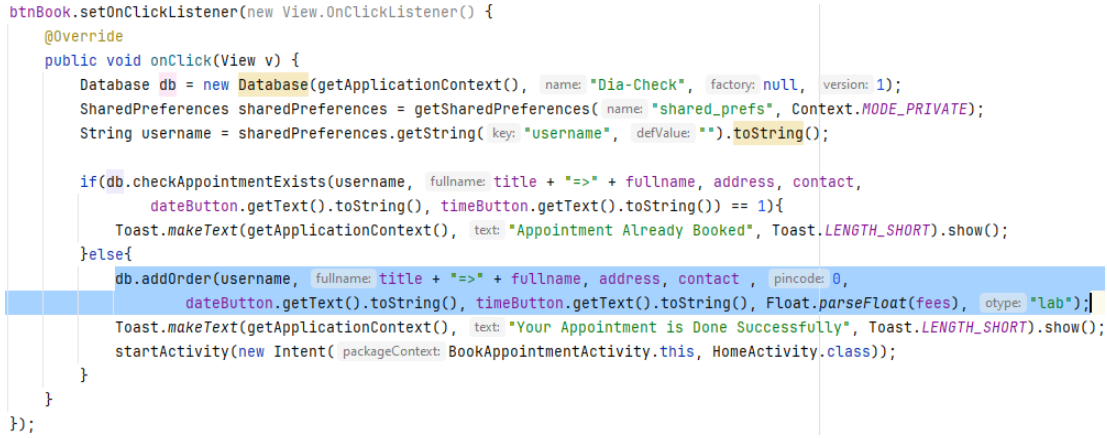
Для створення замовлення необхідно надати застосунку перелік функцій для маніпулюванням замовленнями. Серед яких:

– додавання замовлення (рис. 3.15 – 3.16);

– перевірка наявності замовлення (рис. 3.17 – 3.18).

```
3 usages
public void addOrder(String username, String fullname, String address,
                     String contact, int pincode, String date, String time, float price, String otype){
    ContentValues cv = new ContentValues();
    cv.put("username", username);
    cv.put("fullname", fullname);
    cv.put("address", address);
    cv.put("contactno", contact);
    cv.put("pincode", pincode);
    cv.put("date", date);
    cv.put("time", time);
    cv.put("amount", price);
    cv.put("otype", otype);
    SQLiteDatabase db = getWritableDatabase();
    db.insert( table: "orderplace",  nullColumnHack: null, cv);
    db.close();
}
```

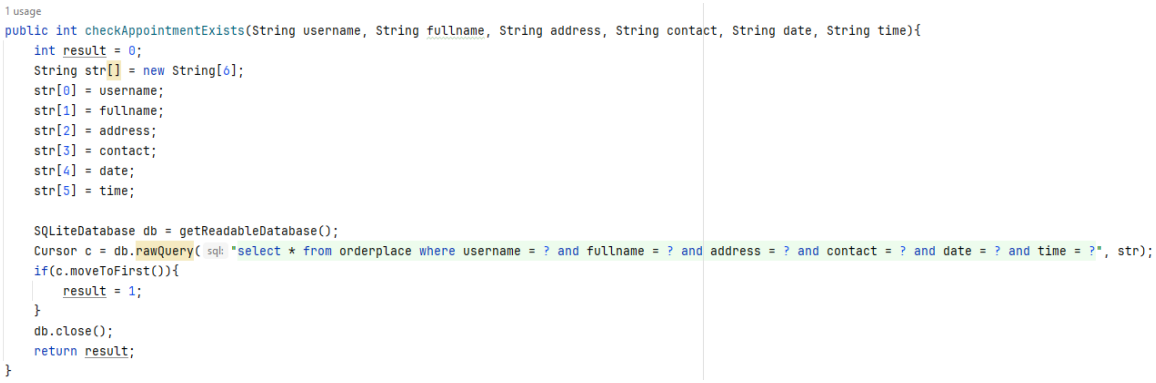
Рисунок 3.15 – Створення функції для додавання замовлення облікового запису користувача у серверному компоненті бази даних



```
btnBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Database db = new Database(getApplicationContext(), name: "Dia-Check", factory: null, version: 1);
        SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
        String username = sharedPreferences.getString( key: "username", defValue: "").toString();

        if(db.checkAppointmentExists(username, fullname: title + "=>" + fullname, address, contact,
                dateButton.getText().toString(), timeButton.getText().toString()) == 1){
            Toast.makeText(getApplicationContext(), text: "Appointment Already Booked", Toast.LENGTH_SHORT).show();
        }else{
            db.addOrder(username, fullname: title + "=>" + fullname, address, contact , pincode: 0,
                    dateButton.getText().toString(), timeButton.getText().toString(), Float.parseFloat(fees), otype: "lab");
            Toast.makeText(getApplicationContext(), text: "Your Appointment is Done Successfully", Toast.LENGTH_SHORT).show();
            startActivity(new Intent( packageContext: BookAppointmentActivity.this, HomeActivity.class));
        }
    }
});
```

Рисунок 3.16 – Використання функції для додавання замовлення облікового запису користувача у серверному компоненті букінгу консультації із лікарем



```
1 usage
public int checkAppointmentExists(String username, String fullname, String address, String contact, String date, String time){
    int result = 0;
    String str[] = new String[6];
    str[0] = username;
    str[1] = fullname;
    str[2] = address;
    str[3] = contact;
    str[4] = date;
    str[5] = time;

    SQLiteDatabase db = getReadableDatabase();
    Cursor c = db.rawQuery( sql: "select * from orderplace where username = ? and fullname = ? and address = ? and contact = ? and date = ? and time = ?", str);
    if(c.moveToFirst()){
        result = 1;
    }
    db.close();
    return result;
}
```

Рисунок 3.17 – Створення функції для перевірки наявності замовлення облікового запису користувача у серверному компоненті бази даних

```
btnBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Database db = new Database(getApplicationContext(), name: "Dia-Check", factory: null, version: 1);
        SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
        String username = sharedPreferences.getString( key: "username", defValue: "").toString();

        if(db.checkAppointmentExists(username, fullname: title + "=>" + fullname, address, contact,
                dateButton.getText().toString(), timeButton.getText().toString()) == 1){
            Toast.makeText(getApplicationContext(), text: "Appointment Already Booked", Toast.LENGTH_SHORT).show();
        }else{
            db.addOrder(username, fullname: title + "=>" + fullname, address, contact , pincode: 0,
                    dateButton.getText().toString(), timeButton.getText().toString(), Float.parseFloat(fees), otype: "lab");
            Toast.makeText(getApplicationContext(), text: "Your Appointment is Done Successfully", Toast.LENGTH_SHORT).show();
            startActivity(new Intent( packageContext: BookAppointmentActivity.this, HomeActivity.class));
        }
    }
});
```

Рисунок 3.18 – Функція для перевірки наявності замовлення облікового запису користувача у серверному компоненті букінгу консультації з лікарем

3.2 Розробка серверної частини з використанням імпортованих Java-UI класів

Для створення серверної частини мобільного застосунку використані імпортовані Java-UI класи, щоб забезпечити функціональність та зручність взаємодії з користувачем.

Щоб коректно працювати з усіма аспектами клієнтського й серверного компонентів необхідно імпортувати усі необхідні клас й компоненти. Буде розглянуто клас серверного компонента картки послуги обстеження у лабораторії (рис. 3.19).

```
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.TimePicker;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
```

Рисунок 3.19 – Перелік імпортованих елементів

У класі самого серверного компонента треба проініціалізувати всі функціональні елементи клієнтського компонента для їх успішного маніпулювання всередині серверної частини (рис. 3.20 – 3.21).

```
7 usages
HashMap<String, String> item;
3 usages
ArrayList list;
2 usages
SimpleAdapter sa;
3 usages
TextView tvTotal;
2 usages
ListView lst;
3 usages
private DatePickerDialog datePickerDialog;
2 usages
private TimePickerDialog timePickerDialog;
4 usages
private Button dateButton, timeButton, btnCheckout, btnBack;
11 usages
private String[][] packages = {};
```

Рисунок 3.20 – Декларування необхідних змінних для маніпулювання функціями серверної та клієнтської частин застосунку

```
dateButton = findViewById(R.id.buttonBMCartDatePicker);
timeButton = findViewById(R.id.buttonBMCartTimerPicker);
btnCheckout = findViewById(R.id.buttonBMCartCheckout);
btnBack = findViewById(R.id.buttonBMCartBack);
tvTotal = findViewById(R.id.textViewBMCartTotalPrice);
lst = findViewById(R.id.listViewBMCart);
```

Рисунок 3.21 – Ініціалізація необхідних змінних для маніпулювання функціями серверної та клієнтської частин застосунку

Також необхідно комунікувати з базою даних і передавати дані між сторінками застосунку. Для трансферу даних використано клас SharedPreferences, тоді як для роботи з базою даних – Database.

На рис. 3.22 наведено фрагмент коду з декларуванням та ініціалізацією трансферу даних між сторінками та бази даних.

```
SharedPreferences sharedPreferences = getSharedPreferences( name: "shared_prefs", Context.MODE_PRIVATE);
String username = sharedPreferences.getString( key: "username",  defValue: "").toString();

Database db = new Database(getApplicationContext(), name: "Dia-Check",  factory: null,  version: 1);
```

Рисунок 3.22 – Створення та ініціалізація необхідних змінних для маніпулювання трансфером даних та бази даних серверної частини застосунку

Важливим є коректна робота обробників подій для кнопок та інших елементів клієнтського компонента.

На рис. 3.23 зазначено фрагменту коду що реалізує обробники події для сторінки карток з консультації з лікарями.

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent( packageContext: DoctorDetailsActivity.this, FindDoctorActivity.class));
    }
});

lst.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
        Intent it = new Intent( packageContext: DoctorDetailsActivity.this, BookAppointmentActivity.class);
        it.putExtra( name: "text1", doctor_details[i][0]);
        it.putExtra( name: "text2", doctor_details[i][1]);
        it.putExtra( name: "text3", doctor_details[i][3]);
        it.putExtra( name: "text4", doctor_details[i][4]);
        startActivity(it);
    }
});
```

Рисунок 3. 23 – Створення обробників подій для кнопки повернення на попередню сторінку та перелік лікарів серверної частини карток лікарів

Розроблена діаграма класів для створеного застосунку наведено у додатку А.

3.3 Розробка клієнтської частини з використанням XML-розмітки

Для розробки клієнтської частини мобільного застосунку використовувалася XML-розмітка, яка дозволяє чітко визначити структуру інтерфейсу користувача та забезпечити його зручність і привабливість. У роботі детально описано XML-розмітку, що відповідає за вигляд сторінки карток запису на обстеження.

Розмітка інтерфейсу користувача для побудована на основі ConstraintLayout, що дозволяє створити гнучкий і адаптивний макет. XML-файл починається з декларації версії та кодування (рис. 3.24):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".CartLabActivity">
```

Рисунок 3.24 – Декларація версії та кодування розмітки, а також певна параметризація

Далі було створено перші елементи розмітки, а саме заголовків (рис. 3.25).

```
<TextView
    android:id="@+id/titleDDName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dia-Check"
    android:textColor="@color/colorBlack"
    android:textSize="34sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.025" />
```

Рисунок 3.25 – Декларація текстового заголовку та певна атрибуція

Усі зміни відображаються у конструкторі сторінки (рис. 3.26).

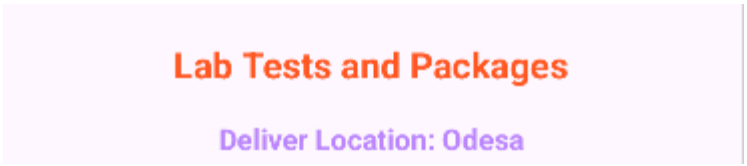


Рисунок 3.26 – Результат декларації текстових заголовків

Для відображення списку лабораторних тестів і пакетів використовується ListView. Він розташований нижче підзаголовка і займає значну частину екрану (рис. 3.27):

```
<ListView
    android:id="@+id/listViewBMCart"
    android:layout_width="363dp"
    android:layout_height="353dp"
    android:layout_marginTop="12dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="@+id/titleViewBMCartTitle"
    app:layout_constraintHorizontal_bias="0.496"
    app:layout_constraintStart_toStartOf="@+id/titleViewBMCartTitle"
    app:layout_constraintTop_toBottomOf="@+id/titlePNTitle"
    app:layout_constraintVertical_bias="0.0" />
```

Рисунок 3.27 – Декларація списку елементів та певна атрибуція

Список елементів матиме вигляд на рис. 3.28.

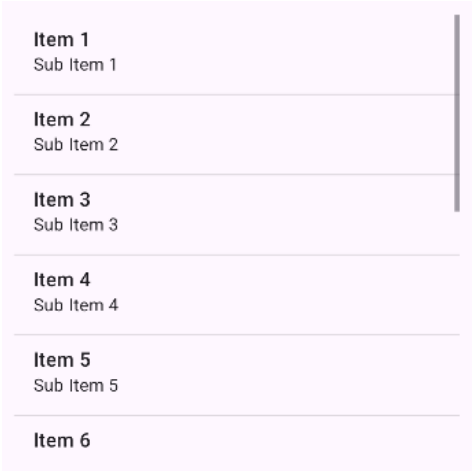


Рисунок 3.28 – Результат декларації списку елементів

Далі були створені заголовки для загальної вартості, кнопок вибору дати й часу тощо (рис. 3.29 – 3.30).

```
<Button
    android:id="@+id/buttonBMCartBack"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:background="@drawable/btn_bg"
    android:text="Back"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.092"
    app:layout_constraintStart_toStartOf="parent" />

<TextView
    android:id="@+id/textViewBMCartTotalPrice"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Total Cost"
    android:textColor="#FFBB86FC"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/listViewBMCart"
    app:layout_constraintVertical_bias="0.057" />
```

Рисунок 3.29 – Декларація списку елементів та певна атрибуція

Рисунок 3.30 – Результат декларації заголовків та кнопок

3.4 Функціонал програмного застосунку

Коли користувач відкриває застосунок, першим вікном є вікно входу в обліковий запис або логіна. Йому надані 2 поля вводу для ім’я облікового запису та пароля, кнопка увійти та посилання на сторінку реєстрації, якщо ще не був створений новий обліковий запис. Сторінка має вигляд на рис. 3.31.

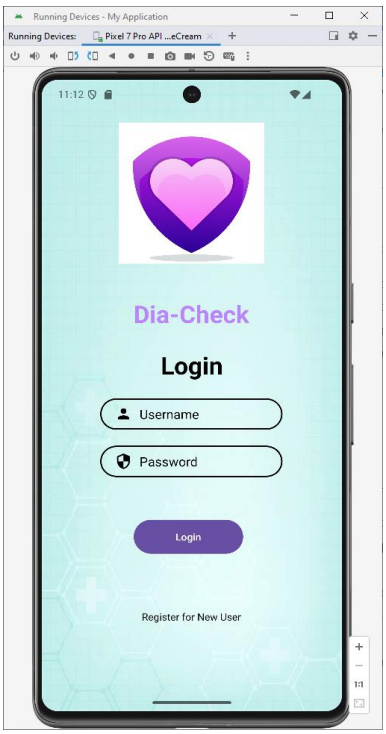


Рисунок 3.31 – Сторінка логіна

Нехай буде з’ясовано що користувач поки не має обліковий запис та натиснув на посилання для створення облікового запису. Його перенесе до сторінки реєстрації що містить 4 поля вводу для назви облікового запису, електронної адреси, та пароля та його підтвердження (рис. 3.32).

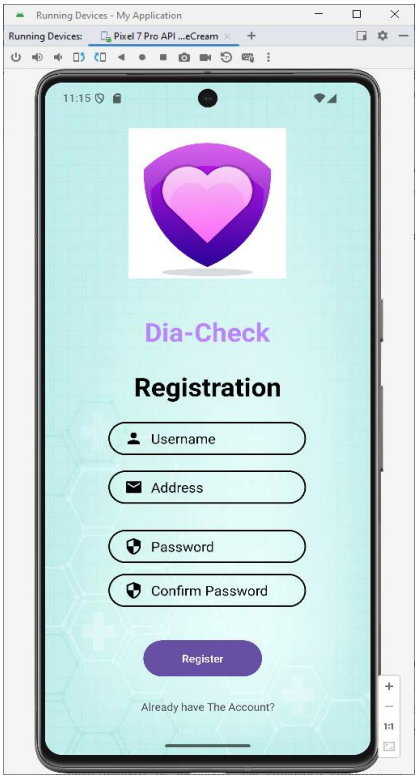


Рисунок 3.32 – Сторінка реєстрації

Після входу до облікового запису користувач переходить до домашньої сторінки застосунку, що надає функціонал для проходження обстеження, придбання ліків, запису на консультацію лікаря, перегляд корисних статей та вихід з облікового запису. На рис.3.33 показано вигляд домашньої сторінки.

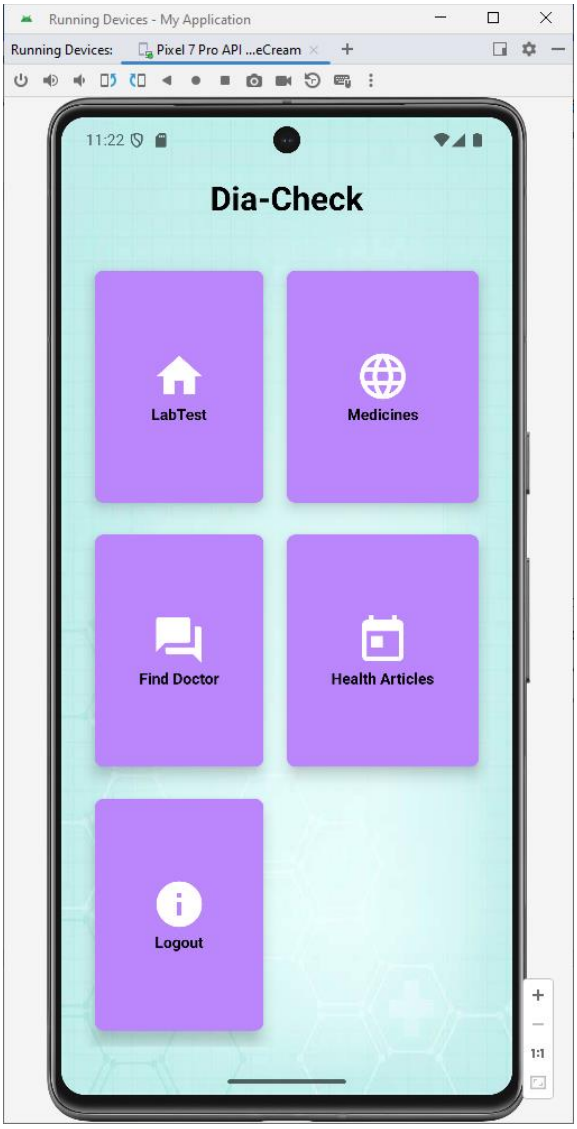


Рисунок 3.33 – Домашня сторінка

Якщо користувач вибрав проходження обстеження, після відкриття сторінки обстежень перед ним з’являється перелік обстежень, доступних для проходження, після натискання на які користувач має змогу продивитися деталі обстежень та їхню вартість. Після натискання на кнопку кошика треба вказати час і дату обстеження, і після натискання кнопки підтвердження – вказати свої контактні реквізити. Перелік сторінок показаний на рис. 3.34 – 3.36.

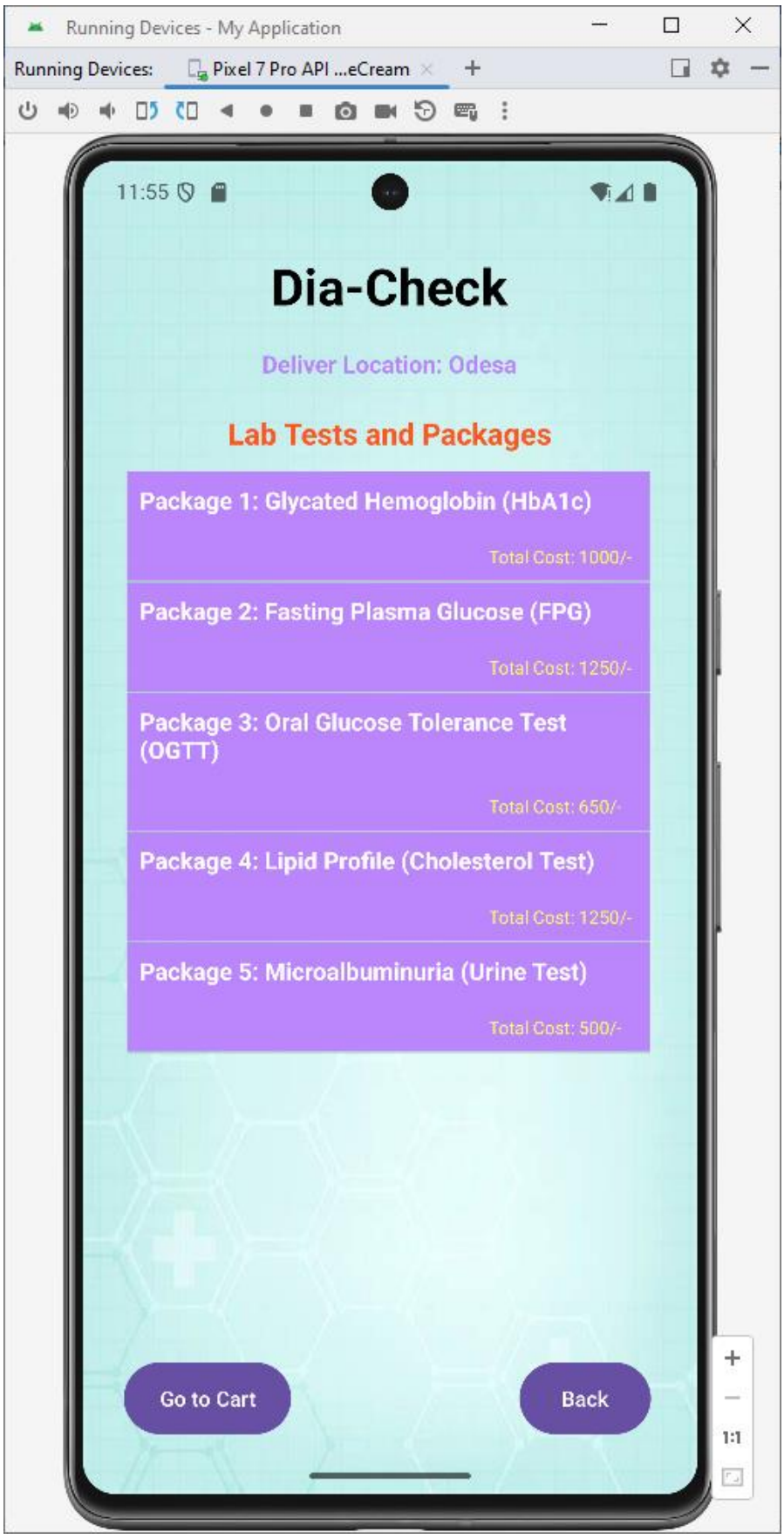


Рисунок 3.33 – Сторінка перегляду списку обстежень

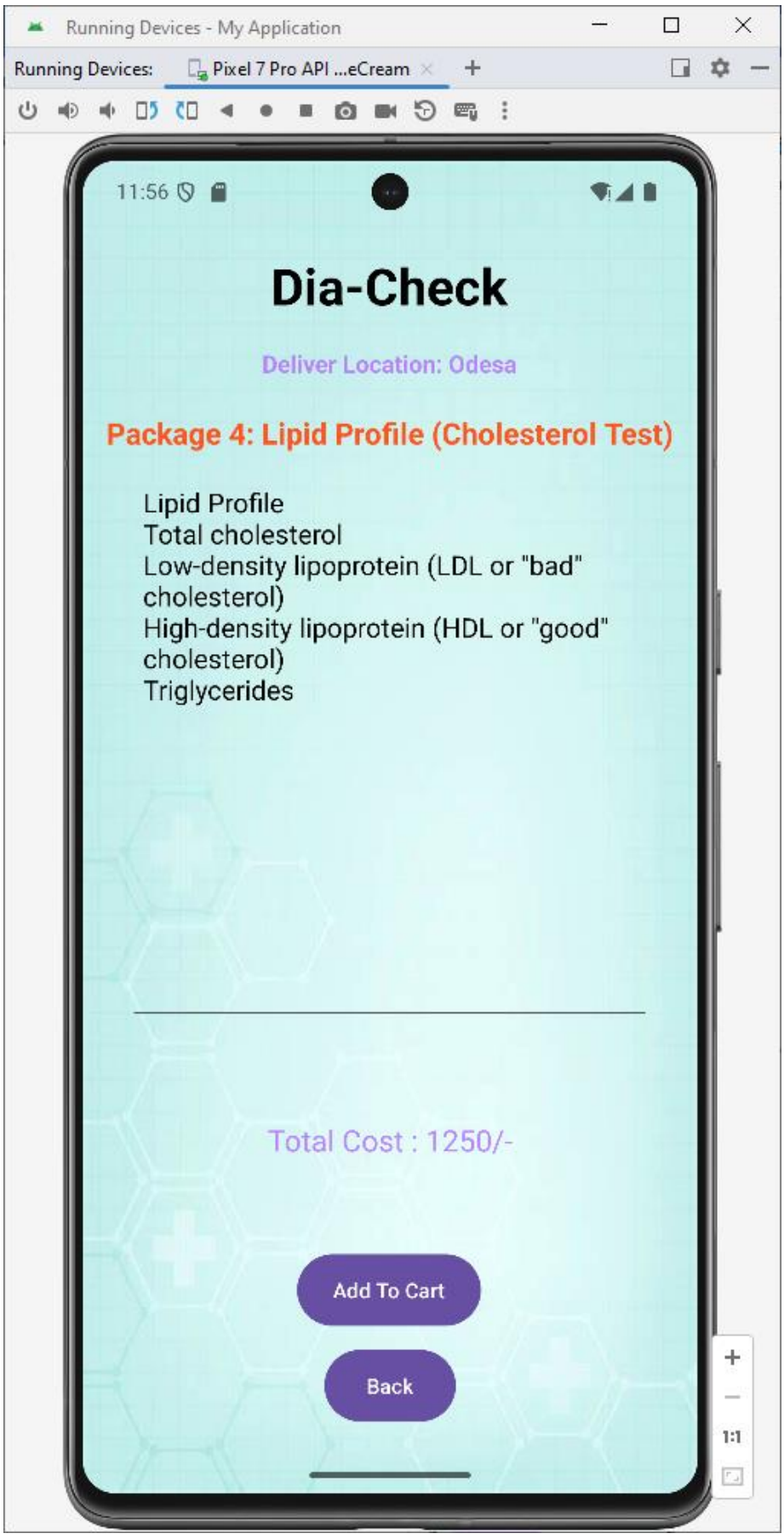


Рисунок 3.34 – Детальна інформація щодо обстеження при натисканні на нього

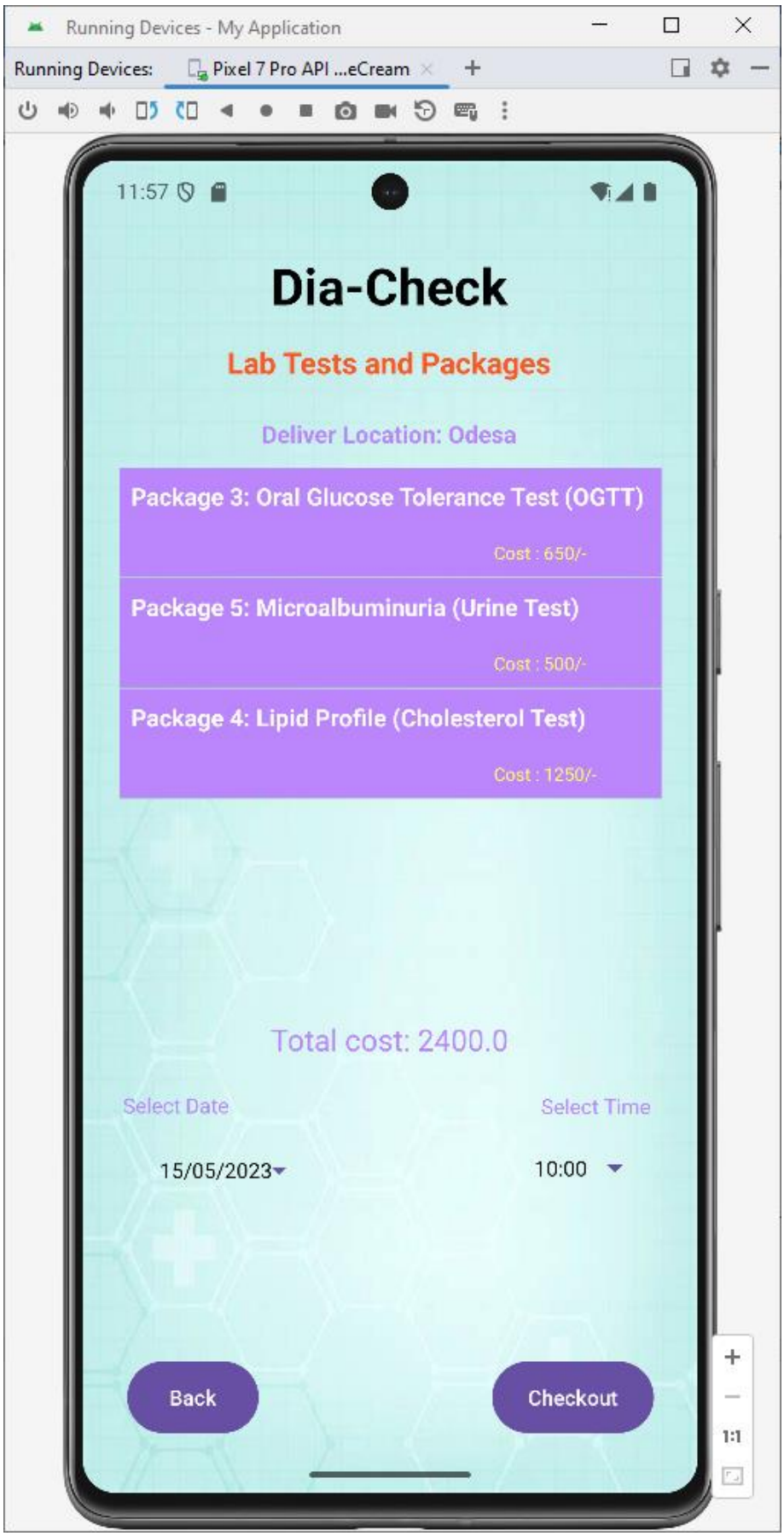


Рисунок 3.35 – Сторінка з кошиком обстежень

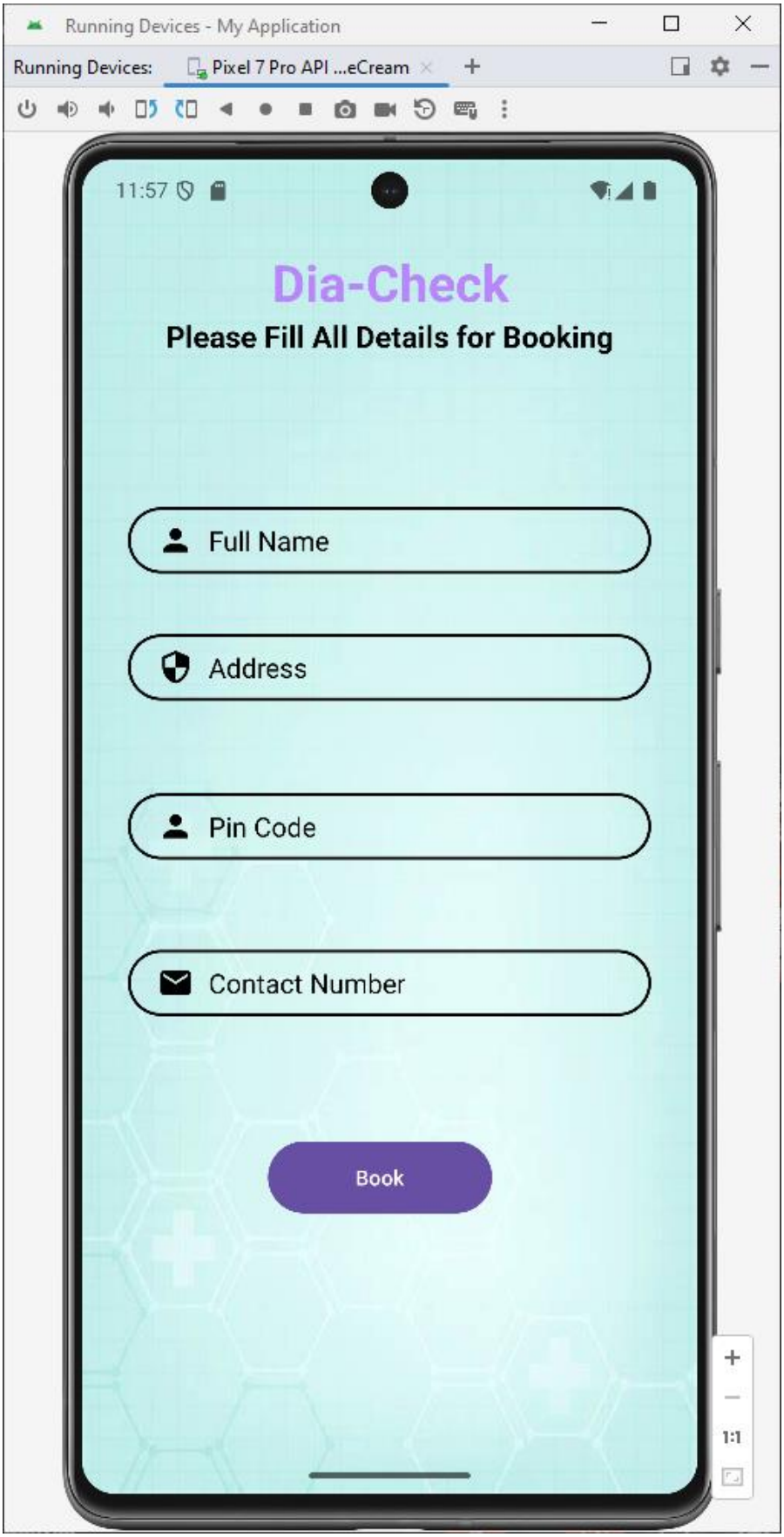


Рисунок 3.36 – Контактні реквізити

Далі, якщо користувач вирішив придбати ліки, на відповідній сторінці застосунку він побачить перелік ліків, при натисканні на які користувач має змогу побачити їх опис та вартість. При натисканні на кнопку кошика він побачить дату і час, коли ліки надійдуть, а після натискання кнопки підтвердження треба буде вказати свої контактні реквізити (рис. 3.37 – 3.40).

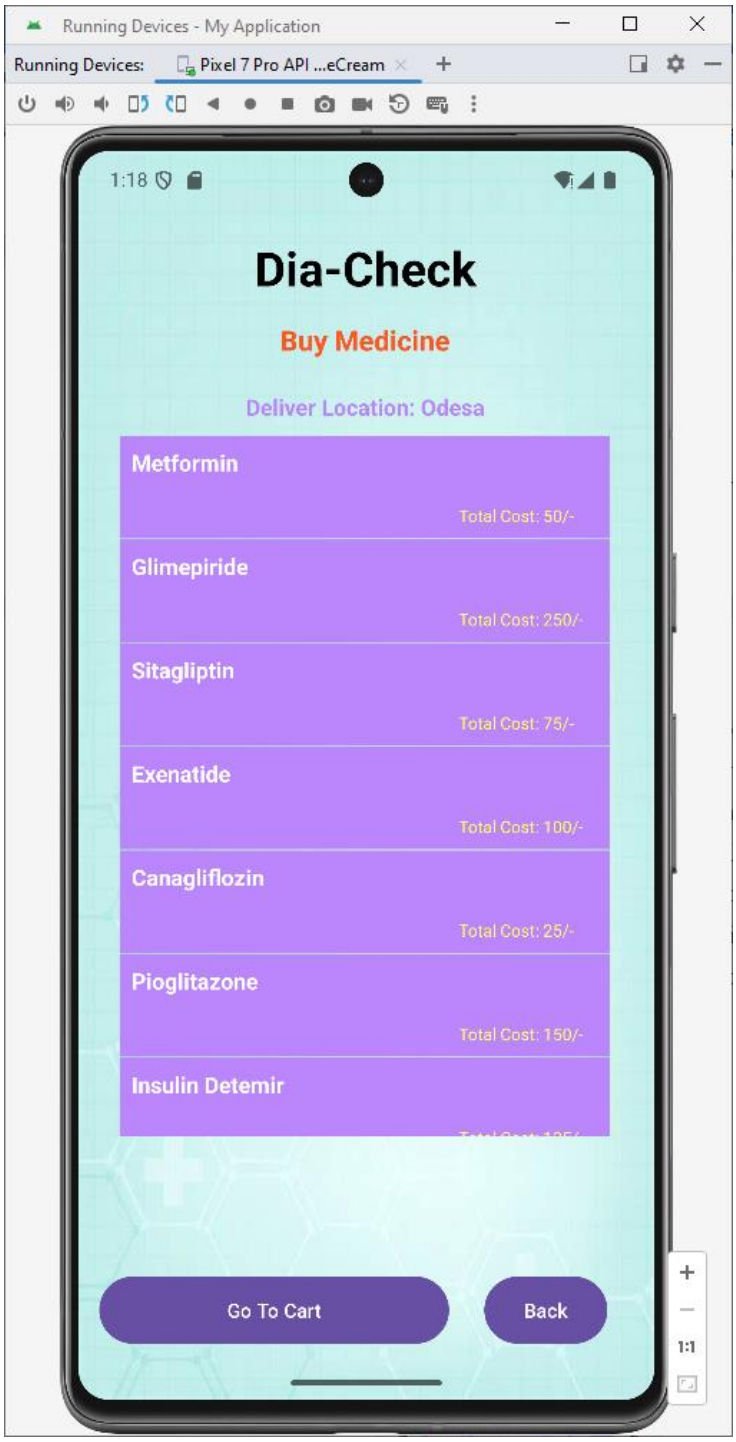


Рисунок 3.37 – Сторінка перегляду певного виду лікаря

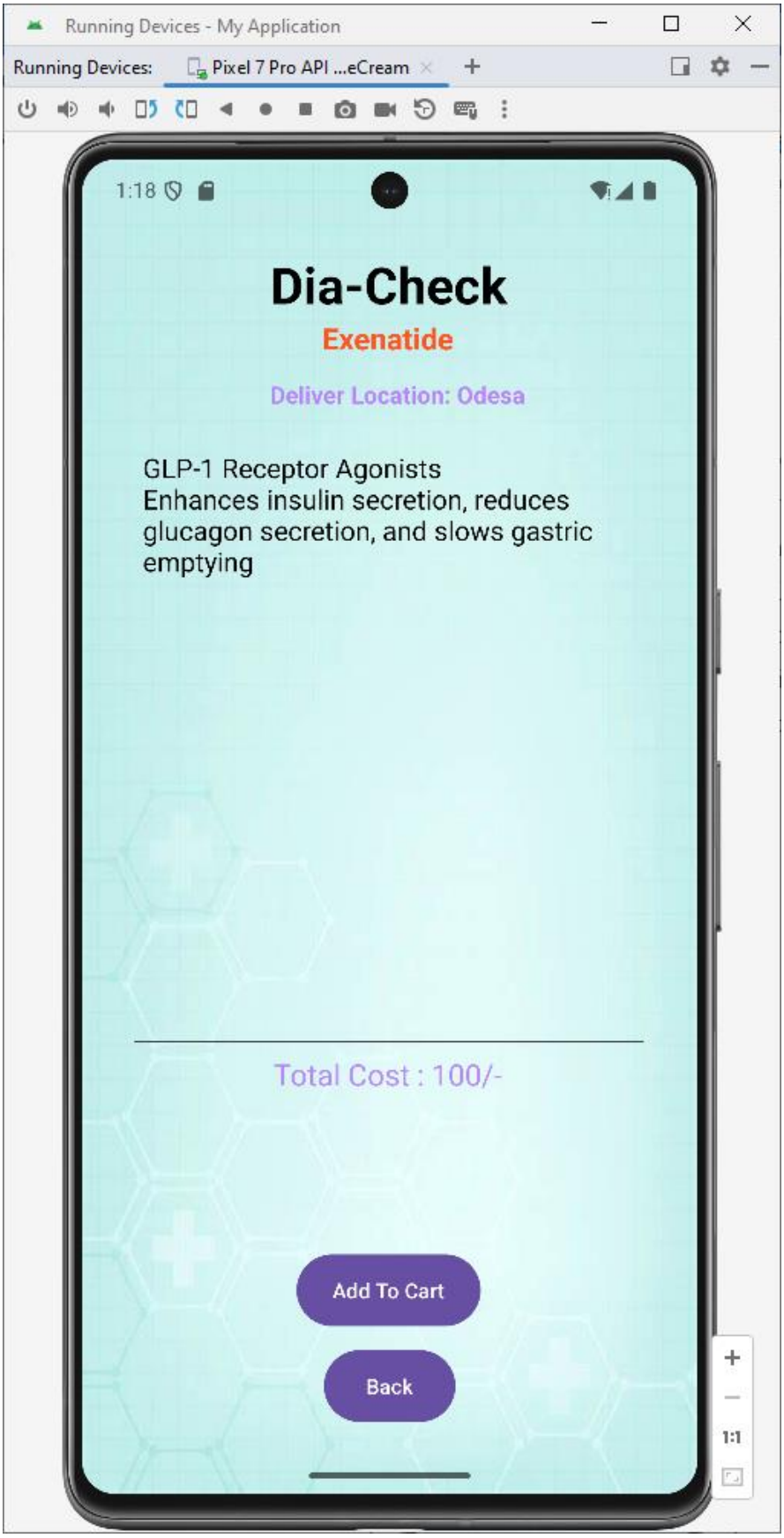


Рисунок 3.38 – Детальна інформація щодо ліків при натисканні на них

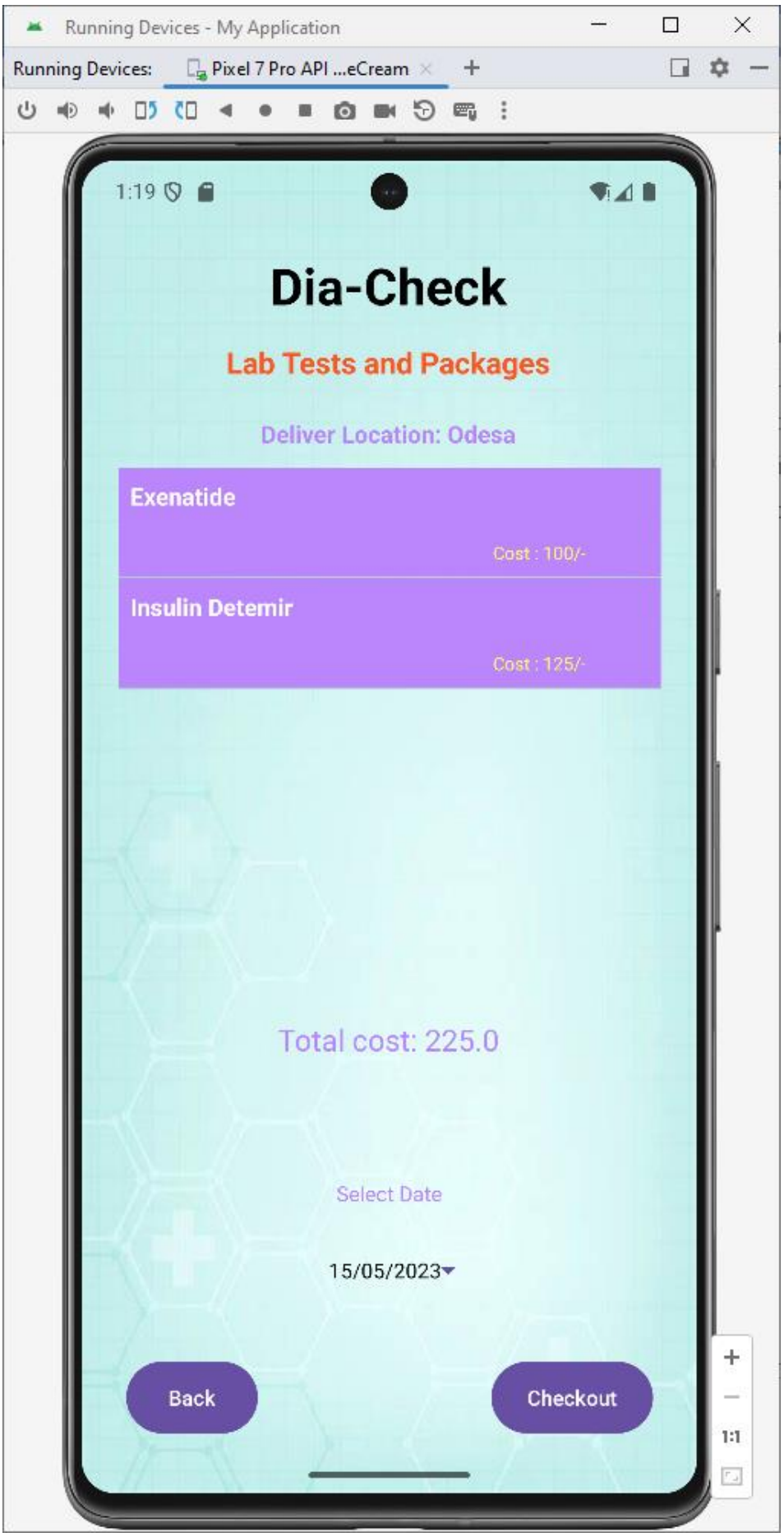


Рисунок 3.39 – Сторінка з кошиком ліків

Рисунок 3.40 – Контактні реквізити

Наступної дією користувача є запис на прийом до лікаря. При переході на сторінку лікарів з’являється перелік лікарів різних напрямків, при натисканні на яких, можна вибрати з переліку певного медичного спеціаліста. Далі користувач має змогу продивитися деталі про лікаря та вартість його прийому, після вибору дати та часу прийому. Перелік цих сторінок наведено на рис. 3.41 – 3.44.

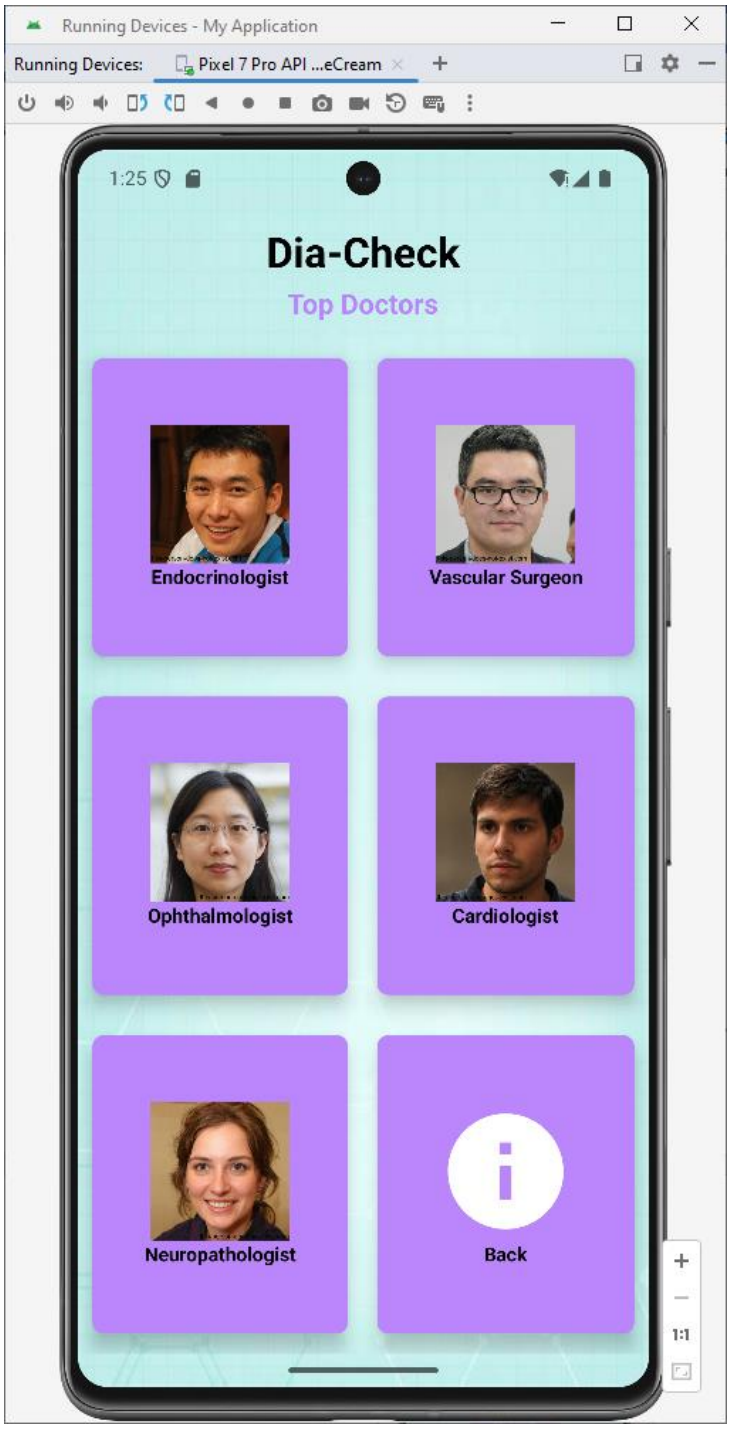


Рисунок 3.41 – Сторінка перегляду певного спеціаліста

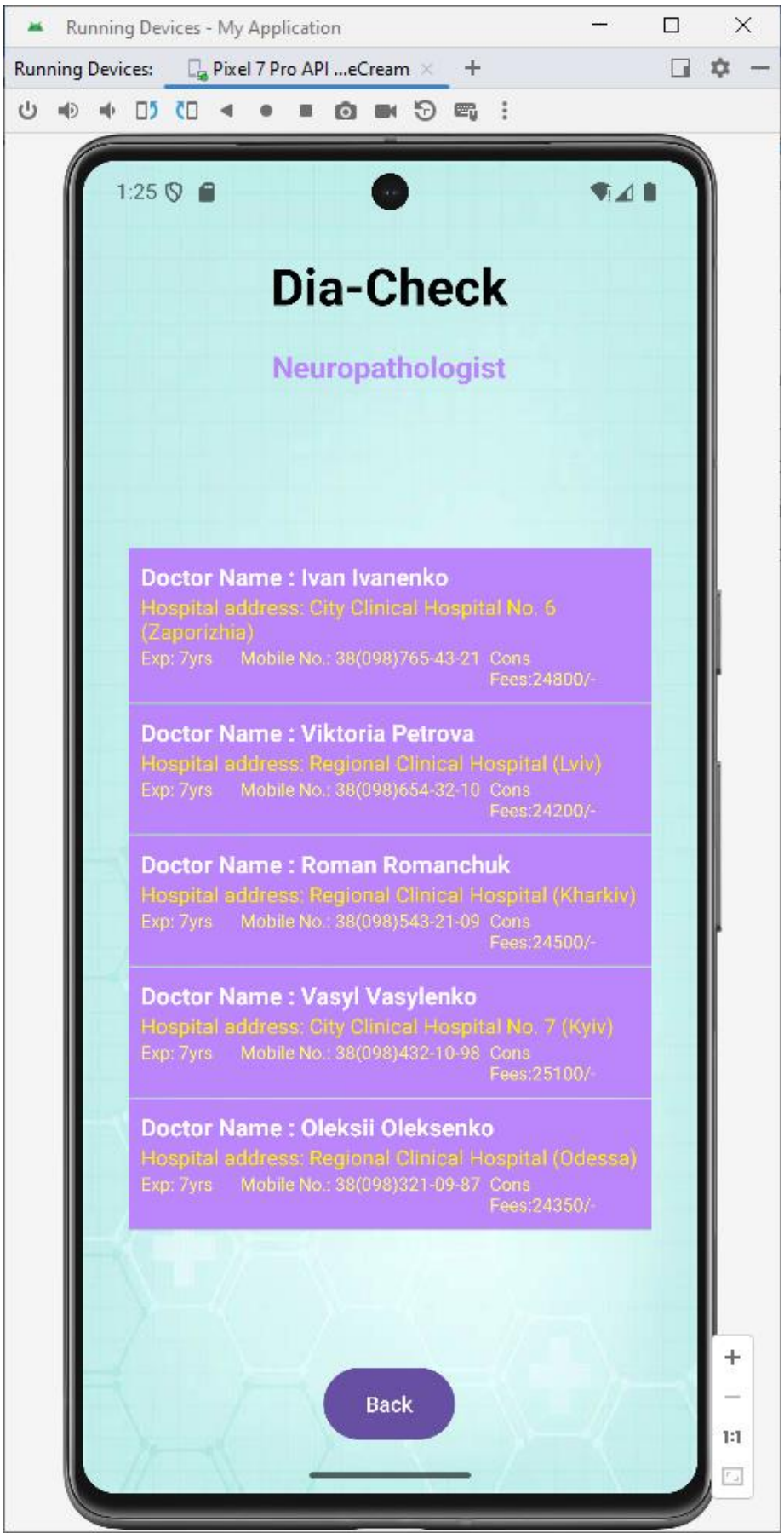


Рисунок 3.42 – Перелік лікарів з детальною інформацією про них

Рисунок 3.43 – Сторінка з букінгом зустрічі

Рисунок 3.44 – Сторінка перегляду певного виду медичних публікацій

Отже, у роботі реалізовано повний функціонал застосунку для індивідуалізованої діагностики стану здоров'я людини в контексті цукрового діабету. Створено інтуїтивно зручний інтерфейс для візуалізації даних про картки послуг та товарів, що дозволяє користувачам легко вести моніторинг цукрового діабету та отримувати персоналізовані рекомендації для кращого керування рівнем цукру в крові.

3.5 Висновки до розділу 3

У розділі розглянуто основні етапи розробки мобільного застосунку, зокрема серверної та клієнтської частин. Розробка мобільного застосунку включала як серверну, так і клієнтську частини, що дозволило створити повноцінний функціональний продукт. Використання Java-UI класів для серверної частини та XML-розмітки для клієнтської частини забезпечило високий рівень інтерактивності та зручності для кінцевого користувача. Застосунок успішно виконує основні завдання.

Описано процес розробки та імплементації мобільного застосунку, спрямованого на полегшення контролю за діабетом і покращення якості життя хворих.

Розроблений мобільний застосунок для моніторингу цукрового діабету має потужний функціонал, спрямований на полегшення контролю за хворобою та покращення якості життя пацієнтів. Його збереження даних, аналіз та персоналізовані рекомендації забезпечують ефективний інструмент для профілактики діабету.

ВИСНОВКИ

Розроблене програмне забезпечення призначене для аналізу фізіологічних даних та параметрів, зібраних за допомогою мобільного застосунку під час моніторингу рівня цукру в крові на базі клінічних досліджень. Програма дозволяє проводити індивідуалізовану діагностику стану здоров'я в контексті цукрового діабету та приймати рішення щодо подальшого контролю та лікування. Потенційні користувачі включають медичних фахівців та пацієнтів, які отримують можливість індивідуалізованих рекомендацій та спостереження за своїм здоров'ям.

У першому розділі виконано SWOT-аналіз аналогів програмних застосунків для моніторингу здоров'я задля узагальнення можливостей, загроз, слабких та сильних сторін серед них. Тим самим аналіз аналогів допоміг визначити функціональні вимоги створюваного застосунку. Розробка застосунку для моніторингу рівня цукру в крові вимагає не лише глибокого розуміння медичних аспектів, а й знання сучасних інструментів мобільної розробки. Використання таких технологій, як XML-розмітка та SQLite, надає можливість створити ефективні та масштабовані мобільні застосунки, які можуть відігравати важливу роль у моніторингу та діагностиці стану здоров'я.

У другому розділі розглянуто ключові аспекти проєктування застосунку з метою забезпечення ефективного та зручного інструменту для хворих на діабет. Розроблено функціональні вимоги, інтерфейс користувача, методи зберігання та аналізу даних, а також персоналізовані рекомендації для кращого керування рівнем цукру в крові. Описано процес розробки та імплементації мобільного застосунку, спрямованого на полегшення контролю за діабетом і покращення якості життя хворих.

Третій розділ присвячений розробленю мобільного застосунку, його серверної та клієнтської частин. Використання Java-UI класів для серверної частини та XML-розмітки для клієнтської частини забезпечило високий рівень інтерактивності та зручності для кінцевого користувача. Розроблений мобільний застосунок для моніторингу цукрового діабету має потужний функціонал, спрямований на полегшення контролю за хворобою та покращення якості життя

пацієнтів. Його збереження даних, аналіз та персоналізовані рекомендації забезпечують ефективний інструмент для профілактики діабету.

Розроблений застосунок може бути використаний не лише для моніторингу цукрового діабету, а й для інших медичних потреб, що вимагають систематичного контролю та аналізу даних здоров'я.

Подальше розширення функціоналу створеного мобільного застосунку може стосуватися впровадження нових функцій та можливостей, таких як: інтеграція з медичними засобами телемедицини, синхронізація даних у реальному часі або покращені аналітичні інструменти, може привернути більше користувачів та підвищити конкурентоспроможність застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ferreira E.S., Franco F.A., Santos M. M., Levcovitz A.A., Dias M.A., Moreira T.R. The efectiveness of mobile application for monitoring diabetes mellitus and hypertension in the adult and elderly population: systematic review and meta-analysis. *BMC Health Services Research*. 2023. Vol. 23:855. DOI: 10.1186/S12913-023-09879-6.

2. Мобільний застосунок. Wikipedia. 2023. URL: https://uk.wikipedia.org/wiki/Мобільний Застосунок.

3. Програми для вимірювання рівня глюкози та діабету за допомогою смартфона. URL: https://lerartigos.com/uk/apps-to-measure-glucose-and-diabetes-by-smartphone/.

4. mySugr Your diabetes data, simply there.URL: https://www.mysugr.com.

5. Azumio Glucose Buddy. URL: https://www.glucosebuddy.com.

6. Sirma Diabetes:M. URL: https://diabetes-m.com.

7. glooko Diasend. URL: https://diasend.com.

8. Wikipedia Розробка застосунків для мобільних пристроїв. 2024. URL: https://uk.wikipedia.org/wiki/Розробка застосунків для мобільних пристроїв.

9. Google Layouts in views. URL: https://developer.android.com/XML-layouts.

10. Google Save data using SQLite. URL: https://developer.android.com/SQLite.

11. Wikipedia Інформаційна модель. URL: https://uk.wikipedia.org/wiki/Інформаційна модель.

12. Google SQLiteOpenHelper. URL: https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.

13. LucidChart UML Use Case Diagram Tutorial. URL: https://www.lucidchart.com/pages/uml-use-case-diagram.

14. А. Вокер Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.

15. Techy PiD SQLite CRUD operation with example in android. URL: https://www.techypid.com/sqlite-crud-operation-with-example-in-android.

ДОДАТКИ

Додаток А. Діаграма класів