Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА КРОСПЛАТФОРМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КОНТРОЛЮ ПРИЙОМУ ІНСУЛІНУ

DEVELOPMENT OF A CROSS-PLATFORM MOBILE APPLICATION FOR MONITORING INSULIN INTAKE

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Антонюк Валерій Анатолійович

Керівник:
Бойко Лев Степанович

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____
«___» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Антонюку Валерію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка кросплатформного мобільного застосунку для контролю прийому інсуліну»
Керівник роботи: доцент Бойко Л.С.
затверджені наказом закладу вищої освіти від *«31» грудня 2025 р.* № *541/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра
«29» травня 2026 р.
3. Вихідні дані до роботи: *методичні матеріали з розробки мобільних застосунків, принципи побудови кросплатформних систем, методи організації локального збереження даних, підходи до проектування користувацького інтерфейсу та засоби реалізації функціоналу.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз сучасного стану проблеми, існуючих засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та параметрів надійності проектованої системи, функціонально-структурна схема роботи об'єкта проектування.*
5. Перелік графічного матеріалу: *1 таблиця, 7 рисунків, 8 лістингів*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Бойко Л. С.*		
Специфікація вимог до розробленої системи	*Бойко Л. С.*		
Розробка об’єкта проектування	*Бойко Л. С.*		
Нормоконтроль	*Суринович О. М.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	Бойко Л.С.		

7. Дата видачі завдання *«3» лютого 2026 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об’єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об’єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об’єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об’єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 29.05.2026 р.	

Здобувач вищої освіти _____ Валерій АНТОНЮК

Керівник кваліфікаційної роботи _____ Лев БОЙКО

АНОТАЦІЯ

Антонюк В. А. Розробка кросплатформного мобільного застосунку для контролю прийому інсуліну. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі здійснено аналіз сучасного стану ринку мобільних застосунків для контролю прийому інсуліну, виявлено їхні недоліки та визначено проблеми, які виникають при розробці таких рішень. Сформульовано завдання на розробку власного мобільного застосунку. У другому розділі обґрунтовано вибір технологій та засобів реалізації, визначено функціональні, нефункціональні та технічні вимоги до продукту. У третьому розділі реалізовано практичну частину застосунку: розроблено інтерфейс, логічну взаємодію застосунку, базу даних, а також проведено тестування. У висновках узагальнено основні результати роботи, підтверджено досягнення поставленої мети та окреслено перспективи подальшого розвитку застосунку.

Ключові слова: інсулін, цукровий діабет, React Native, SQLite, кросплатформна розробка, мобільний застосунок, Zustand, Expo.

ABSTRACT

Antoniuk V. A. Development of a cross-platform mobile application for monitoring insulin intake. Bachelor's qualification work of the Educational Program “Software Engineering” in the specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, and a list of references.

The first chapter analyzes the current state of the mobile application market for insulin intake control, identifies their shortcomings, and defines the problems that arise during the development of such solutions. The task of developing a custom mobile application was formulated. The second chapter substantiates the choice of technologies and implementation tools and defines the functional, non-functional, and technical requirements for the product. The third chapter implements the practical part of the application: the interface, application interaction logic, and database were developed, and testing was carried out. The conclusions summarize the main results of the work, confirm the achievement of the set goal, and outline the prospects for further development of the application.

Keywords: insulin, diabetes mellitus, React Native, SQLite, cross-platform development, mobile application, Zustand, Expo.

ЗМІСТ

ВСТУП

Актуальність теми. Цукровий діабет є одним із найпоширеніших ендокринних захворювань у світі. За даними Міжнародної федерації діабету, понад 537 мільйонів дорослих людей живуть із цим діагнозом [1]. В Україні, за офіційною статистикою МОЗ, зареєстровано понад 1,3 мільйона осіб із діабетом [2].

Пацієнти з діабетом 1-го типу потребують довічної замісної інсулінотерапії. Ефективне управління захворюванням вимагає ведення детального щоденника самоконтролю. Помилки в дозуванні або пропуск ін’єкцій можуть призвести до небезпечних для життя гіпоглікемічних або гіперглікемічних станів [3]. Своєчасне виявлення таких станів та оперативне коригування дози інсуліну є критично важливим для запобігання тяжким ускладненням [4].

У цьому контексті актуальним є створення сучасного мобільного застосунку для контролю прийому інсуліну, який поєднує інтуїтивно зрозумілий інтерфейс, можливість розрахунку дози інсуліну, систему нагадувань, Continuous Glucose Monitoring(CGM) та біометричний захист. Інтеграція з системами безперервного моніторингу глюкози дозволяє отримувати дані про рівень цукру в режимі реального часу, що значно підвищує безпеку пацієнта. Використання технології React Native дозволяє реалізувати кросплатформний додаток, що забезпечує його стабільну роботу як на пристроях з операційною системою Android, так і iOS.

Мета кваліфікаційної роботи – розробка кросплатформного мобільного застосунку «Gluksy» для автоматизації контролю прийому інсуліну з функціями ведення журналу ін’єкцій, розрахунку болюсного інсуліну, нагадувань, CGM-моніторингу, статистики та біометричного захисту.

Об’єкт дослідження – процеси автоматизованого контролю інсулінотерапії у пацієнтів із цукровим діабетом.

Предмет дослідження – методи та програмні засоби для кросплатформної реалізації контролю прийому інсуліну з використанням React Native, Expo, SQLite, Zustand, Victory Native та Expo Notifications.

Для досягнення поставленої мети необхідно виконати такі завдання:

— здійснити аналіз предметної області та визначити основні вимоги до розроблюваного мобільного застосунку;

— провести аналіз існуючих технологій і обґрунтувати вибір засобів розробки;

— впровадити адаптивний інтерфейс, що коректно відображається на мобільних пристроях з різними розмірами екранів;

— забезпечити надійне локальне збереження даних користувачів у SQLite;

— реалізувати основні модулі: журнал ін’єкцій, калькулятор болюсу, нагадування, CGM-модуль, статистику та біометричний захист;

— провести комплексне тестування застосунку, включаючи перевірку функціональності, продуктивності та стабільності роботи.

Практичне значення розробки полягає в розробці застосунку, який може бути безкоштовно використаний пацієнтами з цукровим діабетом для ведення щоденника самоконтролю. Додаток працює в офлайн-режимі, підтримує українську мову, забезпечує захист конфіденційних медичних даних через біометричну аутентифікацію та не вимагає реєстрації.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ РІШЕНЬ У СФЕРІ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ КОНТРОЛЮ ПРИЙОМУ ІНСУЛІНУ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сучасний ринок мобільних додатків для контролю діабету інтенсивно розвивається, що зумовлено зростанням кількості пацієнтів із цим захворюванням. Водночас аналіз функціональних можливостей наявних програмних продуктів дозволяє виявити низку проблем, які стримують подальший розвиток даного сегменту. До них належать обмежені можливості персоналізації, складність інтерфейсу, недостатня інтеграція з системами безперервного моніторингу глюкози (CGM), а також відсутність комплексного підходу до захисту конфіденційних медичних даних.

Серед ключових продуктів, що закріпились на ринку, доцільно виокремити низку найбільш популярних. Аналіз таких програмних продуктів дозволяє визначити сучасні тенденції розвитку галузі, а також виявити їхні переваги та недоліки.

MySugr – одне із найпопулярніших рішень для контролю діабету [5]. Даний застосунок пропонує ведення щоденника з фіксацією глюкози, їжі, інсуліну, активності та нотаток, а також інтеграцію з CGM (Dexcom, Freestyle Libre) та формування звітів у форматі PDF.

До переваг MySugr належать: інтуїтивно зрозумілий інтерфейс з елементами гейміфікації, якісна технічна підтримка, регулярні оновлення. До недоліків належать модель поширення freemium (більшість корисних функцій доступні лише за щомісячну плату близько 3 доларів США), залежність від хмарних серверів Roche, відсутність офлайн-режиму, обмежена підтримка української мови.

Diabetes:M – застосунок, який є потужним інструментом для досвідчених пацієнтів [6]. Даний застосунок пропонує розширену аналітику (статистика,

тренди, прогнозування глікемії), калькулятор болюсу з урахуванням вуглеводних коефіцієнтів та факторів чутливості, інтеграцію з CGM та інсуліновими помпами.

До переваг Diabetes:M належать: найкраща аналітика серед аналогів, розрахунок активного інсуліну (IOB), підтримка різноманітних медичних пристроїв. До недоліків належать: складний інтерфейс, висока вартість (близько 5 доларів США на місяць), відсутність української локалізації, велике споживання батареї.

Glooko – застосунок, який орієнтований на клінічне використання [7]. Даний застосунок підтримує понад 70 моделей глюкометрів, інсулінових помп та CGM, забезпечує віддалений доступ лікаря до даних пацієнта та формування звітів для клінічних візитів .

До переваг Glooko належать: найкраща підтримка медичних пристроїв серед аналогів, орієнтація на клінічне використання. До недоліків належать: висока вартість (від 5 доларів США на місяць), відсутність калькулятора болюсу.

LibreLink – застосунок [8], який є офіційним елементом системи Flash CGM Freestyle Libre. Даний застосунок забезпечує зчитування даних із сенсора через NFC (на Android) або Bluetooth (на iOS), відображення поточного рівня глюкози та тренду, ведення щоденника.

Перевагами LibreLink є безкоштовний доступ до базового функціоналу, пряма інтеграція з системою CGM Freestyle Libre, а також інтуїтивно зрозумілий інтерфейс. Недоліками, своєю чергою, виступають жорстка прив’язка до сенсорів власного виробництва (неможливість роботи з іншими CGM), відсутність вбудованого калькулятора болюсу, а також обов’язкова реєстрація в екосистемі Abbott.

У результаті проведеного аналізу було виявлено, що жоден із проаналізованих застосунків не поєднує усі бажані якості: безкоштовність використання, можливість роботи без активного підключення до мережі Інтернет, наявність української локалізації, наявність біометричного захисту та

наявність зручного калькулятора болюсу. Саме цю нішу має заповнити розроблюваний застосунок «Gluksy».

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

У межах даної роботи було проведено аналіз предметної області та досліджено сучасний стан розв’язання поставленої проблеми.

Для досягнення поставленої мети в межах кваліфікаційної роботи необхідно виконати такі технічні та функціональні завдання:

— здійснити аналіз предметної області та визначити основні вимоги до розроблюваного мобільного застосунку;

— провести аналіз існуючих технологій і обґрунтувати вибір засобів розробки;

— реалізувати адаптивний інтерфейс, що коректно відображається на мобільних пристроях з різними розмірами екранів;

— забезпечити надійне локальне збереження даних користувачів у базі даних SQLite;

— реалізувати мобільний застосунок із використанням обраних інструментів та технологій розробки;

— провести комплексне тестування застосунку, включаючи перевірку функціональності, продуктивності та стабільності роботи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

У процесі розробки мобільного застосунку для контролю прийому інсуліну з використанням React Native [9] виникла потреба у чіткому плануванні архітектури програмного забезпечення та визначенні основних принципів побудови системи. Коректно спроєктована архітектура забезпечує стабільність роботи застосунку, зручність масштабування функціоналу, повторне використання компонентів, а також спрощує процес подальшого тестування та підтримки програмного продукту. Особливо важливо це є для мобільних застосунків, які повинні коректно працювати на різних мобільних пристроях, підтримувати адаптивний інтерфейс та забезпечувати швидку взаємодію користувача з системою.

На основі попереднього аналізу предметної області, сучасних програмних рішень та потреб користувачів було сформульовано комплекс функціональних та нефункціональних вимог до мобільного застосунку для контролю прийому інсуліну. До функціональних вимог належить, перш за все, можливість ведення журналу ін’єкцій. Користувач має мати змогу створювати нові записи про введення інсуліну, переглядати вже існуючі, редагувати їх у разі потреби та видаляти застарілі або помилкові записи. Всі записи повинні групуватися за датами для зручності перегляду.

Наступною важливою функцією є розрахунок дози болюсного інсуліну. Система має автоматично визначати рекомендовану дозу на основі введених користувачем даних: кількості вуглеводів у їжі (в грамах) та поточного рівня глюкози (в ммоль/л). При цьому необхідно враховувати індивідуальні коефіцієнти користувача: вуглеводний коефіцієнт (ICR), фактор чутливості до інсуліну (ISF), цільовий рівень глюкози та активний інсулін (IOB), який залишився від попередніх ін’єкцій. Результат розрахунку має округлюватися

до 0,5 одиниці, оскільки більшість інсулінових шприців-ручок мають саме такий крок. Також обов’язковою є наявність системи нагадувань. Користувач повинен мати можливість створювати нагадування, вказуючи їхню назву, час спрацювання та дні тижня для повторення. Нагадування мають надсилатися у вигляді локальних push-сповіщень, які працюють навіть при закритому додатку та без підключення до інтернету.

Для пацієнтів, які використовують системи безперервного моніторингу глюкози (CGM), важливою є інтеграція з Apple HealthKit [10]. Додаток має запитувати дозвіл на читання даних глюкози, після чого імпортувати показники за останні кілька годин та відображати їх у вигляді графіка з кольоровими зонами (гіпоглікемія, цільовий діапазон, гіперглікемія).

Статистика та візуалізація даних є необхідним інструментом для аналізу ефективності лікування. Додаток має відображати графіки денних доз інсуліну, розраховувати середні показники (загальну кількість одиниць інсуліну, кількість ін’єкцій, середню дозу на день, середній рівень глюкози) за вибраний користувачем період (7, 14 або 30 днів). Що стосується налаштувань, користувач має мати можливість обирати тему оформлення (темна, світла або системна), змінювати мову інтерфейсу (українська або англійська), а також вмикати або вимикати біометричне блокування.

Для забезпечення безпеки даних повинно бути передбачено функціонал управління екстреними контактами та SOS-кнопка. Користувач повинен мати можливість додати номери телефонів близьких людей, які отримають SMS-повідомлення з поточним рівнем глюкози в разі активації SOS-функції.

Для сімей або закладів, де один мобільний пристрій використовують кілька пацієнтів, повинно бути реалізовано мультипрофільність. Користувач може створювати кілька профілів, перемикатися між ними без втрати даних.

Щодо нефункціональних вимог, першочергову увагу приділено продуктивності. Час завантаження будь-якого екрану додатку не повинен перевищувати 1 секунди. Час виконання операцій створення, читання, оновлення та видалення даних (CRUD) має становити не більше 50 мілісекунд. Розрахунок

дози болюсу повинен відбуватися максимально швидко – за час, що не перевищує 100 мілісекунд.

Щодо кросплатформності, додаток повинен коректно працювати на пристроях під управлінням iOS 14 та новіших версій, а також Android починаючи з 8 версії. Інтерфейс має адаптуватися до різних розмірів екранів – від невеликих смартфонів до планшетів.

Однією з ключових нефункціональних вимог є автономність. Додаток має працювати без доступу до мережі Інтернет. Всі дані мають зберігатися локально на пристрої, а жодна інформація не повинна передаватися на віддалені сервери без явної згоди користувача.

Безпека даних повинна забезпечуватися, зокрема, біометричною аутентифікацією. Користувач повинен мати можливість увімкнути блокування додатку за допомогою Face ID, Touch ID або сканера відбитків пальців. При цьому сповіщення, що надсилаються додатком, не повинні містити медичних даних на екрані блокування пристрою.

Додаток повинен бути зручним у використанні. Кількість дій для додавання нової ін’єкції не повинна перевищувати п’яти натискань. Інтерфейс має бути інтуїтивно зрозумілим для користувачів з різним рівнем технічної грамотності, включаючи літніх людей.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для вирішення завдання створення мобільного застосунку для контролю прийому інсуліну необхідно було визначити оптимальний інструментарій. З цією метою було проведено аналіз існуючих підходів до розробки мобільних додатків, зокрема розглянуто як нативні, так і кросплатформні технології.

Нативний підхід передбачає створення окремих застосунків для кожної платформи – Android та iOS. Основними перевагами даного підходу є максимальна продуктивність та повний доступ до усіх можливостей операційної

системи. Серед розглянутих нативних рішень для Android можна виділити Java та Kotlin, для iOS – Swift. Нативний підхід передбачає необхідність використання кількох мов програмування, що ускладнює процес розробки. Натомість альтернативою є кросплатформні технології, які дозволяють створювати застосунки з використанням уніфікованої кодової бази.

Одним з найпопулярніших рішень на ринку кросплатформної розробки є фреймворк React Native [9], технологія з відкритим вихідним кодом, створена компанією Meta. Він використовується для розробки програм, які працюють на Android, iOS та інших платформах.

Враховуючи результати проведеного порівняльного аналізу, для розробки мобільного застосунку «Gluksy» було обрано React Native з наступних причин. Інтеграція з нативними модулями, зокрема SQLite, push-сповіщеннями, біометричною аутентифікацією та Apple HealthKit, виявилася значно простішою завдяки готовим бібліотекам в екосистемі Expo [11]. Розмір кінцевого APK-файлу становить лише 12 МБ, що є суттєво меншим порівняно з Flutter (понад 20 МБ), а це є важливим для застосунку, який поширюватиметься через магазини додатків.

Крім того, наявність великої кількості готових UI-компонентів та бібліотек, таких як Victory Native для побудови графіків та date-fns для роботи з датами, дозволила прискорити розробку. Використання TypeScript [12] забезпечило статичну типізацію, що зменшило кількість потенційних помилок на етапі написання коду. Важливим фактором стала також наявність великої україномовної спільноти та доступної документації, що значно спростило процес навчання та вирішення проблем, які виникали під час розробки.

Оскільки розроблюваний застосунок працює з конфіденційною медичною інформацією, питання безпеки даних було одним із пріоритетних під час проєктування. Було прийнято рішення відмовитися від хмарної синхронізації на користь локального зберігання всіх даних на пристрої користувача. Такий підхід унеможливлює витік даних через злам сервера або перехоплення трафіку.

Додатково було реалізовано механізм біометричної аутентифікації з використанням Face ID на пристроях iOS та сканера відбитків пальців на пристроях Android. Користувач має можливість увімкнути цю функцію в налаштуваннях. Після активації при кожному запуску додатку система вимагатиме підтвердження особи перед відображенням будь-яких даних.

У сповіщеннях, що надсилаються додатком, не відображаються конкретні медичні показники (рівень глюкози, доза інсуліну) на екрані блокування. Це запобігає випадковому розголошенню інформації, якщо телефон знаходиться в руках іншої особи.

Всі запити до бази даних SQLite виконуються з використанням параметризованих запитів, що унеможливлює SQL-ін’єкції. Генерація унікальних ідентифікаторів здійснюється за допомогою криптостійкого генератора випадкових чисел expo-crypto.

Таким чином, розроблений застосунок відповідає сучасним вимогам до безпеки медичних даних, зокрема рекомендаціям OWASP Mobile Security Testing Guide [13].

Для керування станом застосунку було розглянуто три альтернативи: Redux, MobX та Zustand.

Redux [14], є найбільш поширеним рішенням для керування станом у React та React Native додатках. Він базується на принципах єдиного джерела істини (single source of truth), незмінності стану (immutability) та чистих функціях (reducers). Основними перевагами Redux є передбачуваність роботи, простота налагодження завдяки інструментам розробника (Redux DevTools) та велика спільнота.

Однак Redux вимагає значної кількості шаблонного коду (boilerplate). Для реалізації навіть простої дії (наприклад, додавання ін’єкції) необхідно створити тип дії (action type), конструктор дії (action creator), обробник у редюсері (reducer), а також налаштувати store. Для застосунку «Gluksy», який не є надзвичайно складним, використання Redux є надмірним. Кількість коду, необхідного для реалізації базового функціоналу, була б у кілька разів більшою

порівняно з іншими рішеннями. Крім того, Redux потребує обгортання додатку в провайдер (<Provider store={store}>), що ускладнює архітектуру.

MobX використовує принципи реактивного програмування [15]. Він автоматично відстежує залежності між даними та оновлює інтерфейс при зміні стану. Це дозволяє писати менше коду порівняно з Redux, оскільки не потрібно створювати дії та редюсери вручну.

Проте MobX потребує використання декораторів (@observable, @action, @computed) для позначення реактивних даних та дій. Хоча декоратори є частиною стандарту ECMAScript, їх підтримка в TypeScript вимагає додаткового налаштування (experimentalDecorators: true у файлі tsconfig.json). Крім того, використання декораторів може ускладнювати читання коду для розробників, які не знайомі з цим підходом. Для застосунку «Gluksy» використання MobX було б можливим, але через необхідність додаткового налаштування та меншу поширеність порівняно з Zustand, цей варіант було відхилено.

Zustand є найлегшою бібліотекою з розглянутих (близько 3 КБ у мініфікованому вигляді). Вона не потребує обгортання додатка провайдерами, має простий API на основі хуків та дозволяє створювати сховища стану з мінімальною кількістю коду. Для даного проєкту було обрано Zustand [16].

Для локального збереження даних було використано SQLite – легку реляційну базу даних [17]. Використання даного підходу дозволяє забезпечити швидкий доступ до даних без необхідності підключення до мережі Інтернет.

Для візуалізації статистичних даних та побудови графіків було використано бібліотеку Victory Native [18], яка забезпечує високу продуктивність завдяки використанню SVG-рендерингу, що дозволяє відображати графіки з плавною анімацією без втрати якості при масштабуванні. Дана бібліотека підтримує різноманітні типи графіків (лінійні, стовпчасті, кругові діаграми), що необхідно для відображення денних доз інсуліну та динаміки рівня глюкози. Процес інтеграції в застосунок було спрощено за рахунок зручного API та добре структурованої документації.

Для роботи з датами та часом було використано бібліотеку date-fns [19]. Ця бібліотека є модульною, що дозволяє імпортувати лише необхідні функції (наприклад, format, parseISO, isToday, isYesterday), що позитивно впливає на кінцевий розмір застосунку. date-fns забезпечує підтримку української локалізації, що дозволило коректно відображати назви місяців та днів тижня українською мовою. Крім того, бібліотека має простий та інтуїтивно зрозумілий API, що не потребує глибокого вивчення документації.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КОНТРОЛЮ ПРИЙОМУ ІНСУЛІНУ НА IOS ТА ANDROID

3.1 Практична реалізація об’єкта проєктування

Розробка мобільного застосунку «Gluksy» для контролю прийому інсуліну здійснювалася на базі фреймворку React Native з використанням платформи Expo та мови програмування TypeScript.

З метою забезпечення зручності супроводу та масштабування коду, файлову структуру проєкту побудовано за модульним принципом, де кожна папка містить логічно пов'язані між собою файли.

Структура проєкту має наступний вигляд. Коренева папка app/ містить усі екрани додатка. Завдяки використанню файлової навігації Expo Router, кожен файл у цій директорії автоматично стає окремим маршрутом. Головним екраном є app/index.tsx, а екрани нижньої панелі вкладок (журнал, статистика, болюс, глюкоза, нагадування та налаштування) згруповано в підпапці app/(tabs)/.

Типи даних винесено в папку src/types/, що забезпечує централізоване управління TypeScript-інтерфейсами. Управління станом застосунку здійснюється за допомогою Zustand stores, які знаходяться в src/store/. Взаємодія з нативними модулями (сповіщення, біометрія, Apple HealthKit) винесена в окремі сервіси в папку src/services/. Глобальні конфігурації, такі як тема оформлення, мова інтерфейсу та активний профіль, реалізовані за допомогою React Context і розташовані в src/context/. Повторно використовувані компоненти інтерфейсу, зокрема перемикач профілів, розміщено в окремій директорії src/components/.

Така організація файлів забезпечує прозорість структури, спрощує навігацію по коду та полегшує внесення змін у майбутньому. Для роботи з SQLite використовується модуль expo-sqlite. Він надає синхронний API: db.runSync() для виконання запитів, db.getAllSync() для отримання даних, db.getFirstSync() для отримання одного рядка.

Ініціалізація бази даних відбувається при першому запуску застосунку. Було створено дев'ять таблиць, які відповідають за зберігання різних типів даних: profiles, insulin_types, injections, glucose_readings, reminders, settings, emergency_contacts, sos_events, bolus_profiles. З метою прискорення виконання запитів, що найчастіше використовуються, у базі даних створено відповідні індекси. Для таблиці injections додано індекс за полем injected_at, який оптимізує сортування записів за хронологією. Також увімкнено WAL-режим (Write-Ahead Logging), що дозволяє одночасно виконувати операції читання та запису без взаємного блокування. Крім того, реалізовано механізм початкової ініціалізації даних: у разі виявлення порожньої таблиці insulin_types під час запуску застосунку, до неї автоматично додаються попередньо визначені типи інсуліну (ліст. 3.1).

Лістинг 3.1 – Ініціалізація бази даних (фрагмент)

```
const db = SQLite.openDatabaseSync('insulin_tracker.db');
export function initDatabase(): void {
  db.execSync(`
    PRAGMA journal_mode = WAL;
    CREATE TABLE IF NOT EXISTS injections (
      id TEXT PRIMARY KEY,
      insulin_id TEXT NOT NULL REFERENCES insulin_types(id),
      dose_units REAL NOT NULL CHECK(dose_units > 0),
      injected_at TEXT NOT NULL,
      glucose_before REAL,
      note TEXT
    );
    CREATE INDEX IF NOT EXISTS idx_injections_date ON
injections(injected_at DESC);
  `);  }
```

кінець лістингу 3.1

Наведений фрагмент демонструє створення головної таблиці injections з усіма необхідними обмеженнями: первинний ключ, зовнішнє посилання на таблицю типів інсуліну, перевірка додатного значення дози, а також індекс для прискорення сортування за датою.

Реалізацію базових CRUD-операцій для роботи з таблицею injections наведено нижче (ліст. 3.2).

Лістинг 3.2 – CRUD-операції для ін’єкцій

```
export function addInjection(data: InjectionFormData): Injection {
  const id = uuidv4();
  db.runSync(
    `INSERT INTO injections (id, insulin_id, dose_units,
injected_at, glucose_before, note)
     VALUES (?, ?, ?, ?, ?, ?)`,
    id, data.insulinId, data.doseUnits,
data.injectedAt.toISOString(),
    data.glucoseBefore ?? null, data.note ?? null
  );
  return { id, ...data };
}

export function getInjections(): Injection[] {
  return db.getAllSync(`
    SELECT i.*, t.name as insulinName, t.color as insulinColor
    FROM injections i JOIN insulin_types t ON t.id = i.insulin_id
    ORDER BY i.injected_at DESC
  `);
}
```

кінець лістингу 3.2

У даному модулі реалізовано дві базові операції: додавання нового запису про ін’єкцію та отримання списку всіх ін’єкцій з сортуванням за датою (від

найновіших до найстаріших). Використання JOIN дозволяє одразу отримати назву та колір інсуліну без додаткового запиту.

Zustand дозволяє створювати сховища стану з мінімальною кількістю коду (ліст. 3.3). Так було створено об’єкти для ін'єкцій, нагадувань, CGM-даних та калькулятора болюсу.

Лістинг 3.3 – Об’єкт для ін’єкцій

```
export const useInjectionsStore = create((set) => ({
  injections: [],
  load: () => set({ injections: getInjections() }),
  add: (data) => {
    const newInjection = addInjection(data);
    set((state) => ({ injections: [newInjection,
...state.injections] }));
  },
  delete: (id) => {
    deleteInjection(id);
    set((state) => ({ injections: state.injections.filter(i =>
i.id !== id) }));
  },
}));
```

кінець лістингу 3.3

Об’єкт ін’єкцій містить чотири основні методи: завантаження даних з бази, додавання нового запису (новий запис додається на початок списку), видалення запису за ідентифікатором.

Об’єкт нагадувань має дещо складнішу логіку порівняно з іншими сховищами, що зумовлено необхідністю інтеграції з системою локальних push-сповіщень (ліст. 3.4).

При додаванні нового нагадування виконується наступна послідовність дій: генерується унікальний ідентифікатор, створюється об'єкт нагадування з

початковими параметрами (активне, без запланованих сповіщень), викликається функція scheduleReminder, яка планує локальні push-сповіщення на основі часу та днів тижня, отримані ідентифікатори сповіщень зберігаються в об'єкті нагадування, після чого нагадування зберігається в базі даних та додається до стану.

При вимкненні нагадування користувачем (перемикач Switch) спрацьовує метод toggle. Він створює копію нагадування з інвертованим прапорцем enabled. Якщо нагадування вмикається, повторно плануються сповіщення; якщо вимикається – скасовуються всі раніше заплановані сповіщення за їхніми ідентифікаторами. Оновлене нагадування зберігається в базі даних, а стан інтерфейсу оновлюється.

Такий підхід забезпечує узгодженість між станом нагадування в інтерфейсі користувача та фактично запланованими сповіщеннями в операційній системі, навіть після перезапуску додатку.

Лістинг 3.4 – Об’єкт для нагадувань

```
export const useRemindersStore = create((set, get) => ({
  reminders: [],
  load: () => set({ reminders: getReminders() }),
  add: (data) => {
    const id = uuidv4();
    const newReminder = { ...data, id, enabled: true,
notificationIds: [] };
    newReminder.notificationIds = scheduleReminder(newReminder);
    saveReminder(newReminder);
    set((state) => ({ reminders: [...state.reminders, newReminder]
}));
  },
  toggle: (id) => {
    const reminder = get().reminders.find(r => r.id === id);
    const updated = { ...reminder, enabled: !reminder.enabled };
    if (updated.enabled) {
```

```
      updated.notificationIds = scheduleReminder(updated);
    } else {
      cancelReminder(reminder.notificationIds);
    }
    saveReminder(updated);
    set((state) => ({ reminders: state.reminders.map(r => r.id ===
id ? updated : r) }));
  },
}));
```

кінець лістингу 3.4

Журнал ін’єкцій є головним екраном застосунку (рис. 3.1). На ньому відображаються всі записи про введення інсуліну, згруповані за датами («Сьогодні», «Вчора», «дд.мм.рррр»). Кожен запис містить тип інсуліну, дозу, час ін’єкції та рівень глюкози (за наявності).

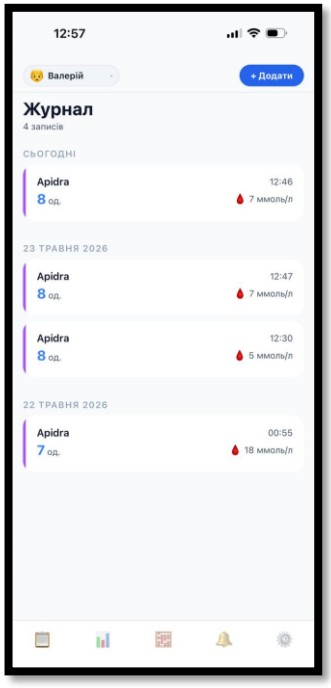

Рисунок 3.1 – Головний екран «Журнал»

Було реалізовано групування записів за датами (ліст. 3.5). Для форматування дат використано бібліотеку date-fns з українською локалізацією [16]. Видалення запису реалізовано за довгим натисканням (long press), що зменшує ризик випадкового видалення.

Лістинг 3.5 – Групування ін’єкцій за датами

```
function groupByDate(injections: Injection[]) {
  const groups: Record<string, Injection[]> = {};
  for (const inj of injections) {
    const date = inj.injectedAt.split('T')[0];
    if (!groups[date]) groups[date] = [];
    groups[date].push(inj);
  }
  return Object.entries(groups).map(([date, items]) => ({ date,
items }));
}

function dateLabel(dateStr: string): string {
  const d = parseISO(dateStr);
  if (isToday(d)) return 'Сьогодні';
  if (isYesterday(d)) return 'Вчора';
  return format(d, 'd MMMM yyyy', { locale: uk });
}
```

кінець лістингу 3.5

Функція groupByDate проходить по масиву ін’єкцій та об’єднує їх за датою (використовується перші 10 символів ISO-рядка). Функція dateLabel форматує дату: для сьогоднішнього дня повертає «Сьогодні», для вчорашнього – «Вчора», для інших дат – повну дату українською мовою.

Калькулятор болюсу є однією з ключових функцій застосунку (ліст. 3.6). Розрахунок дози виконується за формулою (3.1):

$$\frac{Carbs}{ICR} + \frac{Cg - Tg}{ISF} - IOB = D \quad (3.1)$$

де ICR – вуглеводний коефіцієнт;

ISF – фактор чутливості;

IOB – активний інсулін.

Результат округлюється до 0,5 одиниці, оскільки більшість інсулінових шприців-ручок мають саме такий крок.

Лістинг 3.6 – Калькулятор болюсу

```
function calculateBolus(carbs: number, currentGlucose: number,
profile: BolusProfile, iob: number) {
  const carbDose = carbs / profile.carbRatio;
  const correction = (currentGlucose - profile.targetGlucose) /
profile.correctionFactor;
  const rawTotal = carbDose + correction - iob;
  return roundToHalf(Math.max(0, rawTotal));
}
```

кінець лістингу 3.6

Функція calculateBolus обчислює вуглеводну складову, корекційну складову, віднімає активний інсулін та округлює результат. Якщо поточний рівень глюкози нижче 3,9 ммоль/л, калькулятор блокується та показує попередження.

Екран калькулятора болюсного інсуліну з полями введення вуглеводів та глюкози, а також блоком швидкої довідки (рис. 3.2). Для отримання даних з систем безперервного моніторингу глюкози (CGM) було реалізовано інтеграцію з Apple HealthKit [10]. Застосунок запитує дозвіл на читання даних глюкози, після чого отримує інформацію за останні 6 годин (ліст. 3.7). Отримані дані зберігаються в таблицю glucose_readings. Інтеграція виконана згідно з офіційною документацією Apple.

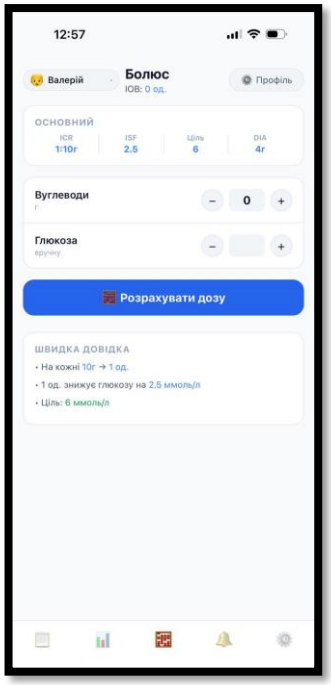

Рисунок 3.2 – Екран «Калькулятор болюсу»

Лістинг 3.7 – Запит дозволу для Apple HealthKit

```
await HealthKit.requestPermissions({
  permissions: { read: [HealthKit.Permissions.Glucose] }
});

const startDate = new Date(Date.now() - 6 * 60 * 60 * 1000);
const samples = await HealthKit.getGlucoseSamples({
  startDate: startDate.toISOString(),
  endDate: new Date().toISOString(),
});
```

кінець лістингу 3.7

Нагадування є необхідною функцією, оскільки пацієнти часто забувають про необхідність вчасно зробити ін’єкцію. Механізм роботи нагадувань полягає в наступному: користувач створює нагадування, вказуючи назву, час та дні тижня. Додаток обчислює найближчу дату та планує сповіщення з повторенням щотижня (рис. 3.3).

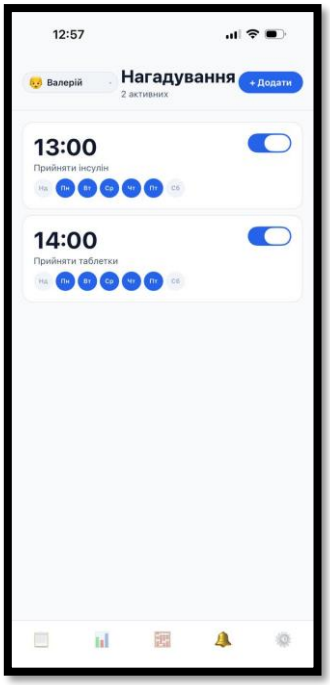

Рисунок 3.3 – Екран «Нагадування»

З метою аналізу ефективності контролю діабету було створено сторінку статистики. Для отримання денних доз інсуліну використовується SQL-запит з агрегацією.

Логіка побудови графіків передбачає обробку розривів у даних: пропущені дні автоматично доповнюються нульовими показниками модулем візуалізації, що дозволяє не переривати часову шкалу. Для візуалізації даних використовується бібліотека Victory Native [18].

Також було реалізовано екран статистики (рис. 3.4), який відображає агреговані показники за вибраний період (7/14/30 днів), стовпчасту діаграму денних доз та детальний перелік днів.

Для захисту конфіденційних даних було додано біометричне блокування (ліст. 3.8). Перевірка виконується в кореневому компоненті додатку до того, як виведеться будь-який екран. Якщо користувач не пройшов аутентифікацію, додаток не запускається.

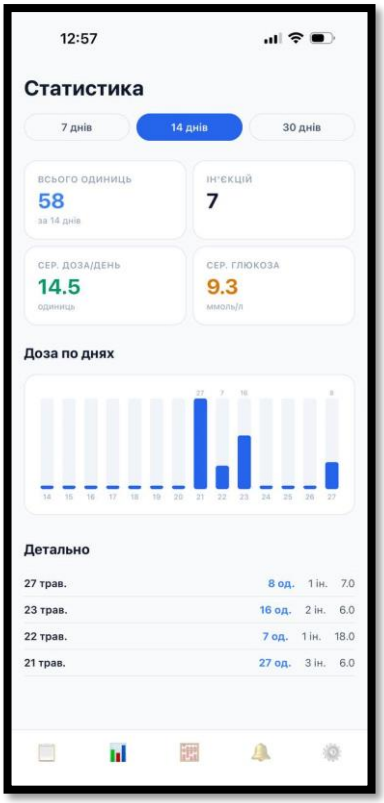

Рисунок 3.4 – Екран «Статистика»

Лістинг 3.8 – Перевірка біометрії

```
async function checkBiometric(): Promise<boolean> {
  const isEnabled = getSetting('biometricLock') === '1';
  if (!isEnabled) return true;
  const result = await LocalAuthentication.authenticateAsync({
    promptMessage: 'Розблокуйте Gluksy',
    fallbackLabel: 'Використати пароль',
  });
  return result.success;
}
```

кінець лістингу 3.8

Забезпечення конфіденційності медичних даних досягається трьома ключовими рішеннями: повним локальним зберіганням інформації на пристрої користувача, відмовою від обов’язкової реєстрації, а також обмеженням вмісту push-сповіщень, які не містять медичних показників на екрані блокування.

Проте локальне зберігання даних створює ризик їх втрати при фізичному пошкодженні пристрою, скиданні системи до заводських налаштувань або випадковому видаленні застосунку. Для мінімізації цього ризику до складу застосунку включено механізм резервного копіювання.

У поточній версії застосунку реалізовано функцію ручного експорту даних. Користувач може в налаштуваннях обрати опцію «Експорт даних», після чого система створює файл у форматі JSON, який містить всі записи про ін’єкції, типи інсуліну, нагадування, показники глюкози, профілі та налаштування. Експортований файл може бути збережений у хмарному сховищі (Google Drive, iCloud, Dropbox) або на зовнішньому носії.

Для відновлення даних після повторної інсталяції додатку користувач може обрати опцію «Імпорт даних» та вказати раніше збережений файл. Система перевіряє цілісність файлу та структуру даних перед імпортом.

У разі виявлення пошкоджень або невідповідності структури, імпорт скасовується з відповідним повідомленням.

У майбутніх версіях планується реалізувати автоматичне резервне копіювання із заданим інтервалом (щодня, щотижня, щомісяця) та шифрування резервних копій за допомогою алгоритму AES-256 для забезпечення додаткового рівня безпеки.

Екран налаштувань реалізований наступним чином – користувач може змінювати мову, керувати профілями, обирати тему оформлення, налаштовувати одиниці виміру глюкози, керувати безпекою, додавати типи інсуліну та переглядати інформацію про застосунок (рис. 3.5).

Таблиця profiles має поле is_active, а всі інші таблиці мають поле profile_id. При перемиканні профілю додаток фільтрує дані за іншим profile_id.

Темізація реалізована з підтримкою трьох варіантів оформлення: темна, світла та системна.Локалізація підтримує дві мови: українську та англійську.

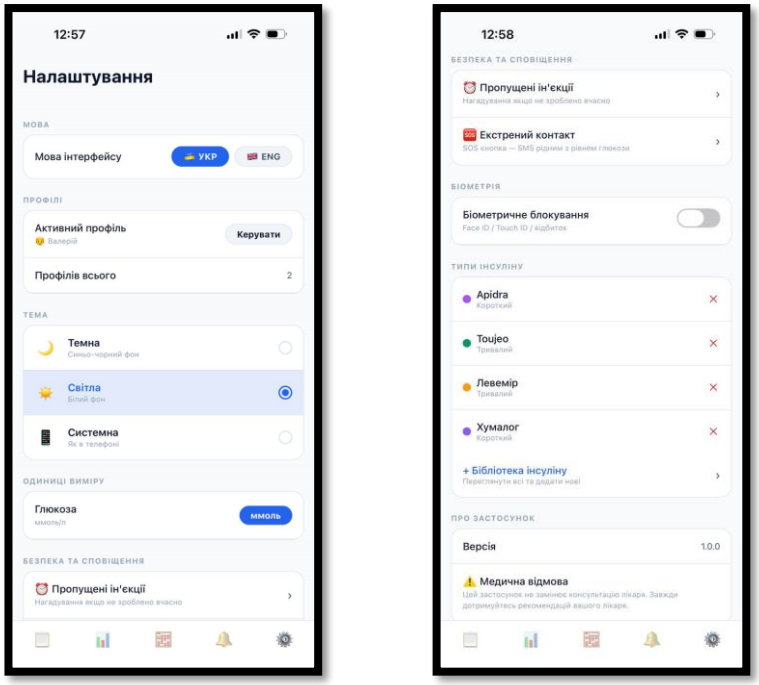

Рисунок 3.5 – Екран «Налаштування»

3.2 Реалізація інтерфейсу

Проєктування користувацького інтерфейсу (UI) мобільного застосунку було спрямовано на створення інтуїтивно зрозумілого, візуально привабливого рішення. Основний акцент під час розробки було зроблено на забезпеченні послідовного стилю оформлення, адаптивності елементів інтерфейсу та зручної навігації між основними функціями застосунку.

Під час проєктування інтерфейсу враховувались сучасні принципи UI/UX-дизайну, зокрема мінімалістичний стиль, логічне розташування елементів керування та швидкий доступ до ключового функціоналу.

Для забезпечення комфортної роботи користувача було реалізовано нижню навігаційну панель (рис. 3.6), яка дозволяє легко переміщатися між журналом ін’єкцій, статистикою, калькулятором болюсу, CGM-графіком, нагадуваннями та налаштуваннями. Також для всіх екранів було розроблено темний режим.

Рисунок 3.6 – Нижня панель навігації

3.3 Тестування та налагодження інформаційно-комп'ютерної системи

Під час розробки було проведено комплексне тестування з метою перевірки коректності роботи функціоналу та стабільності користувацького інтерфейсу. Тестування проводилося на трьох реальних пристроях (iPhone 14 Pro, iPhone SE 2022, Samsung Galaxy A53).

Було виконано 25 ручних тест-кейсів, які підтвердили коректність роботи всіх функцій. Також було написано 28 модульних тестів з використанням фреймворку Jest, які охоплюють ключові функції: округлення до 0.5, розрахунок дози болюсу, врахування активного інсуліну, блокування при гіпоглікемії.

Результати модульного тестування наступні**:** усі 28 тестів пройшли успішно, покриття коду склало приблизно 85. Підхід до модульного тестування базується на методології Test-Driven Development.

Тестування продуктивності на iPhone 14 Pro показало наступні результати (табл. 3.1).

Таблиця 3.1 – Результати тестування продуктивності на iPhone 14 Pro

Операція	Час виконання	Вимога	Статус
Запуск додатку (холодний старт)	1,8 с	< 3 с	Виконано
Відкриття журналу з 50 записами	245 мс	< 1 с	Виконано
Розрахунок дози болюсу	12 мс	< 100 мс	Виконано

Функціональне тестування проводилося за наступними сценаріями.

Додавання ін'єкції. Перевірялася коректність створення нового запису. Користувач натискає кнопку «+ Додати», обирає тип інсуліну зі списку, вводить дозу 4,5 одиниці, за бажанням – рівень глюкози 7,2 ммоль/л та нотатку «Перед сніданком». Після збереження новий запис має з'явитися на початку списку в журналі. Тест вважався пройденим, якщо всі дані відображалися коректно.

Видалення ін’єкції. Перевірялася можливість видалення існуючого запису. Користувач натискає і утримує палець на записі протягом 0,5 секунди. Після появи діалогового вікна з підтвердженням користувач натискає «Видалити». Запис зникає зі списку. Тест вважався пройденим, якщо запис видалявся без помилок, а інші записи залишалися незмінними.

Розрахунок дози болюсу. Перевірялася правильність обчислень калькулятора. Користувач вводить вуглеводи – 45 грамів, поточну глюкозу – 8,5 ммоль/л. Активний профіль має параметри: ICR = 10, ISF = 2, ціль = 6 ммоль/л, IOB = 0. Очікуваний результат: (45/10) + (8,5-6)/2 = 4,5 + 1,25 = 5,75, що після округлення до 0,5 дає 6,0 одиниць. Тест вважався пройденим, якщо калькулятор показав саме 6,0 одиниць та відобразив детальний розрахунок.

Блокування при гіпоглікемії. Перевірявся захисний механізм калькулятора. Користувач вводить глюкозу 3,5 ммоль/л (нижче критичного порогу 3,9 ммоль/л). Калькулятор має заблокувати кнопку розрахунку та показати попередження червоного кольору. Тест вважався пройденим, якщо попередження з'явилося, а кнопка розрахунку стала неактивною.

Створення нагадування. Перевірялася можливість створення нового нагадування. Користувач переходить на екран «Нагадування», натискає «+ додати», вводить назву «Вечірній інсулін», обирає час 21:00 та дні тижня – понеділок, середу, п’ятницю. Після збереження нагадування має з’явитися в списку. Тест вважався пройденим, якщо нагадування відображалося з правильною назвою, часом та індикатором вибраних днів.

Отримання push-сповіщення. Перевірялася доставка локального сповіщення. Після створення нагадування на час 21:00 тестувальник очікував до

настання цього часу. У визначений час на пристрій мало надійти сповіщення з текстом «Вечірній інсулін». Тест вважався пройденим, якщо сповіщення з’являлося на екрані блокування.

Всі шість сценаріїв були успішно пройдені на всіх тестових пристроях (iPhone 14 Pro, iPhone SE 2022, Samsung Galaxy A53, а також на емуляторах).

3.4 Розробка інсталяційного пакета

Для розповсюдження мобільного застосунку, створеного за допомогою фреймворку React Native та платформи Expo, потрібно сформувати інсталяційний пакет у форматі APK для операційної системи Android або IPA для iOS (рис. 3.7). У межах даного проєкту було реалізовано генерацію Android-пакета. Вибір формату залежить від цільової платформи: APK призначений для пристроїв під управлінням Android, тоді як IPA – для екосистеми Apple (iOS).

Рисунок 3.7 – Піктограма застосунку «Gluksy»

Процес створення інсталяційного пакета здійснювалося за допомогою вбудованих засобів Expo, зокрема з використанням команди (eas build --platform android). Дана команда забезпечує створення оптимізованої release-версії

мобільного застосунку, призначеної для подальшого встановлення на мобільні пристрої користувачів. Збірка виконується в хмарному середовищі EAS Build (Expo Application Services), що дозволяє отримати готовий APK-файл без необхідності локального налаштування Android SDK та інструментів збірки.

У результаті виконання процесу збірки формується APK-файл, який містить усі необхідні компоненти застосунку та може бути використаний для поширення, встановлення й тестування програмного продукту. Отриманий файл можна безпосередньо встановити на Android-пристрій через файловий менеджер або adb (Android Debug Bridge), а також завантажити в Google Play Console для публікації в магазині додатків.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено кросплатформний мобільний застосунок «Gluksy» для контролю прийому інсуліну. Повний цикл розробки було виконано з використанням фреймворку React Native, платформи Expo, мови програмування TypeScript, бази даних SQLite та бібліотеки Zustand.

На початковому етапі роботи було проведено аналіз предметної області та досліджено наявні аналоги, зокрема MySugr, Diabetes:M, Glooko та LibreLink. У результаті аналізу було виявлено їхні основні недоліки, які полягають у платній моделі поширення, залежності від постійного підключення до мережі Інтернет, відсутності української локалізації та біометричного захисту даних.

На основі проведеного аналізу було сформульовано функціональні та нефункціональні вимоги до системи, які визначили необхідний набір можливостей та якісних характеристик майбутнього програмного продукту.

Наступним етапом стало обґрунтування вибору технологічного стеку. Для реалізації застосунку було обрано React Native у поєднанні з Expo, що забезпечує кросплатформність та спрощує процес розробки. Мова TypeScript дозволила забезпечити статичну типізацію та зменшити кількість потенційних помилок. Для керування станом застосунку використано бібліотеку Zustand, для локального збереження даних – Expo SQLite, для push-сповіщень – Expo Notifications, для біометричної аутентифікації – Expo Local Authentication, а для візуалізації графіків – Victory Native.

Під час практичної реалізації було розроблено базу даних SQLite, яка складається з дев’яти таблиць, що забезпечують зберігання всієї необхідної інформації про користувачів, ін’єкції, типи інсуліну, показники глюкози, нагадування та інші сутності. Для оптимізації швидкодії запитів було створено відповідні індекси та увімкнено WAL-режим роботи бази даних.

У ході розробки було реалізовано всі основні модулі застосунку: журнал ін’єкцій з автоматичним групуванням записів за датами, калькулятор болюсного

інсуліну, який враховує індивідуальні коефіцієнти користувача (вуглеводний коефіцієнт, фактор чутливості, активний інсулін) та виконує округлення результату до 0,5 одиниці, інтеграцію з Apple HealthKit для імпорту даних глюкози з систем безперервного моніторингу, систему локальних push-нагадувань, модуль статистики з графіками денних доз, біометричне блокування з підтримкою Face ID та Touch ID, а також мультипрофільність, темну та світлу тему оформлення, українську та англійську локалізацію.

Для перевірки працездатності розробленого застосунку було проведено комплексне тестування, яке включало функціональне тестування, модульне тестування з використанням фреймворку Jest (28 тестів, покриття коду склало приблизно 85 %) та тестування продуктивності. Результати тестування продуктивності показали, що запуск застосунку становить 1,8 секунди, а розрахунок дози болюсу виконується за 12 мілісекунд. У процесі тестування було виявлено та усунуто шість помилок, що дозволило забезпечити стабільну роботу програми.

Практичне значення розробленого застосунку полягає в тому, що він готовий до використання пацієнтами з цукровим діабетом для ведення щоденника самоконтролю, розрахунку доз інсуліну та моніторингу рівня глюкози. Застосунок не потребує реєстрації, не передає дані на зовнішні сервери та працює без доступу до Інтернету, що забезпечує повну конфіденційність медичних даних користувача.

Перспективи подальшого розвитку застосунку включають інтеграцію з Google Health Connect для пристроїв на платформі Android, реалізацію функції експорту даних у форматі PDF для надання лікарю, а також розробку механізму прогнозування глікемії на основі історичних даних із використанням методів машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. International Diabetes Federation. IDF Diabetes Atlas. 10th ed. Brussels : International Diabetes Federation, 2021. 168 p. URL: https://doi.org/10.1136/bmjdrc-2021-002122 (дата звернення: 15.02.2026).

2. Міністерство охорони здоров'я України. Статистичні дані щодо захворюваності на цукровий діабет в Україні. Київ : МОЗ України, 2024. 45 с.

3. American Diabetes Association. Standards of Medical Care in Diabetes—2024. Diabetes Care. 2024. Vol. 47, Suppl. 1. P. S1–S300.

4. Cryer P. E. Hypoglycemia in type 1 diabetes mellitus. Endocrinology and Metabolism Clinics of North America. 2019. Vol. 39, No. 3. P. 641–654.

5. MySugr (Roche). URL: https://mysugr.com (дата звернення: 25.02.2026).

6. Diabetes:M (Sirma Medical). URL: https://diabetes-m.com (дата звернення: 25.02.2026).

7. Glooko. URL: https://glooko.com (дата звернення: 25.02.2026).

8. LibreLink (Abbott). URL: https://www.freestyle.abbott/uk-en/getting-started/librelink.html (дата звернення: 25.02.2026).

9. React Native Documentation. URL: https://reactnative.dev/docs/getting-started (дата звернення: 10.03.2026).

10. Apple HealthKit Documentation. URL: https://developer.apple.com/documentation/healthkit (дата звернення: 15.03.2026).

11. Expo SDK Documentation. URL: https://docs.expo.dev (дата звернення: 20.03.2026).

12. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 25.03.2026).

13. OWASP Mobile Security Testing Guide. URL: https://owasp.org/www-project-mobile-security-testing-guide/ (дата звернення: 25.03.2026).

14. Redux Documentation. URL: https://redux.js.org (дата звернення: 27.03.2026).

15. MobX Documentation. URL: https://mobx.js.org (дата звернення: 28.03.2026).

16. Zustand Documentation. URL: https://docs.pmnd.rs/zustand (дата звернення: 29.03.2026).

17. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 30.03.2026).

18. Victory Native Documentation. URL: https://formidable.com/open-source/victory/docs/native (дата звернення: 5.04.2026).

19. Date-fns Documentation. URL: https://date-fns.org/docs/ (дата звернення: 5.04.2026).