Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ПРОГРАМНІ ЗАСОБИ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ СТАНУ ЗДОРОВ'Я НА ПЛАТФОРМІ ANDROID

08-54.МКР.007.00.000 ПЗ

Виконав студент 2 курсу, групи 1КІ-22м
спеціальності 123 — Комп'ютерна інженерія
_____ Ковалик В.І.
Керівник: к.т.н., доц. каф. ОТ
_____ Кадук О.В.
« 13 » грудня 2023 р.
Опонент: д.т.н., проф. каф. МБІС
_____ Яремчук Ю.Є.
« 16 » грудня 2023 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
« 19 » грудня 2023 р.

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

д.т.н., проф. О.Д. Азаров
« 26 » вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Ковалик Валентин Ігорович**

1 Тема роботи «Програмні засоби для моніторингу та аналізу стану здоров'я на платформі Android» керівник роботи Кадук Олександр Володимирович к.т.н., доцент, затверджено наказом вищого навчального закладу від **18.09.2023** року №247

2 Строк подання студентом роботи **17.12.2023**.

3 Вихідні дані до роботи: розроблений програмний засіб для моніторингу та аналізу стану здоров'я під платформу Android, написаний мовою Java.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз методів та засобів моніторингу стану здоров'я на платформі Android, методологія та математичні моделі програмного засобу, розробка програмного засобу для моніторингу стану здоров'я, робота з програмним засобом, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): титульний аркуш, існуючі програмні рішення для моніторингу стану здоров'я, архітектура розроблюваного програмного засобу, функціонал користувацького програмного засобу, висновки.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Кадук Олександр Володимирович к.т.н., доцент		
5	Небава Микола Іванович к.е.н., доцент		

7 Дата видачі завдання **19.09.2023** .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	**20.09.2023**	
2	Огляд існуючих рішень	**25.09.2023**	
3	Проєктування програмного засобу	**06.10.2023**	
4	Розробка програмного засобу	**25.10.2023**	
5	Проведення тестування програмного засобу	**05.11.2023**	
6	Попередній захист	**07.11.2023**	
7	Розрахунок економічної частини	**15.11.2023**	
8	Оформлення пояснювальної записки та ілюстративного матеріалу	**20.11.2023**	
9	Виконання магістерської кваліфікаційної роботи	**26.11.2023**	
10	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	**10.12.203**	
11	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	**14.12.2023**	
12	Перевірка «антиплагіат»	**15.12.2023**	
13	Створення презентації, доповіді до захисту МКР	**16.12.2023**	
14	Захист МКР	**19.12.2023**	

Студент _____ (підпис) Ковалик Валентин Ігорович

Керівник _____ (підпис) к.т.н., доц. Кадук Олександр Володимирович

АНОТАЦІЯ

УДК 004.4:616-056.2-047.36

Ковалик В. І. Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2023 — 117 с. На укр. мові. Бібліогр.: 51 назв; рис.: 23; табл.: 10; посил.: 21.

В роботі було розглянуто ключову роль мобільних пристроїв у сфері медицини та програмних засобів на платформі Android. В роботі проведений аналіз сучасних напрямків дослідження в сфері мобільної медицини, а також огляд вже існуючих програмних рішень для моніторингу стану здоров’я. Розроблено та проведено тестування програмного засобу, на мові програмування Java, котрий дозволяє моніторити та аналізувати тенденцію показників стану здоров’я та формувати звітність в форматі PDF.

Ключові слова: програмний засіб, платформа Android, Java, Android Studio, мобільна медицина, мобільний пристрій, моніторинг, аналіз, стан здоров’я, звіт.

ABSTRACT

УДК 004.4:616-056.2-047.36

Kovalyk Valentyn. Software tools for monitoring and analyzing health status on the Android platform. Master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 117 pp. In Ukrainian language. Bibliographer: 51 titles; fig.: 23; tabl. 10; links: 21.

The paper examines the crucial role of mobile devices in the field of medicine and software solutions on the Android platform. The research includes an analysis of current trends in mobile medicine and a review of existing software solutions for health monitoring. A Java-based software tool was developed and tested, allowing the monitoring and analysis of health indicators trends and generating reports in PDF format.

Key words: software tool, Android platform, Java, Android Studio, mobile medicine, mobile device, monitoring, analysis, health status, report.

ЗМІСТ

ВСТУП

Здоров'я — це найцінніший актив кожної людини. Ретельний моніторинг та своєчасний аналіз стану здоров'я можуть врятувати життя, поліпшити його якість та сприяти активній і тривалій діяльності. Сучасні технології дозволяють зберігати інформацію та розширюють можливості людей в цій сфері. Однак, незважаючи на розвиток медичної науки та медичних технологій, збереження та покращення стану здоров'я залишаються актуальними завданнями.

У сучасному світі, де стрес, неврологічні захворювання та інші фактори можуть впливати на стан здоров'я, наявність доступу до зручних інструментів для моніторингу та аналізу стану здоров'я є досить важливим. Нинішнє суспільство характеризується зростаючою увагою до питань здоров'я та збереження фізичної та психічної гармонії. Серед сучасних технологій відіграють ключову роль мобільні пристрої. Зручність та доступність смартфонів роблять їх ідеальними для впровадження таких інструментів у повсякденне життя, а зокрема платформа Android від компанії Google.

Платформа Android, розроблена компанією Google, володіє найбільшим ринковим покриттям серед мобільних операційних систем у світі. Її поширеність і доступність для розробників роблять її ідеальним вибором для створення програмних засобів, спрямованих на моніторинг і поліпшення стану здоров'я. За статистичними даними 2022 року частка ринку пристроїв на операційній системі Android становить 72.2%, а частка їх найбільшого конкурента Apple з їх операційною системою iOS становить 26.99%. Та попри меншу частку ринку, iOS зберігає значну глобальну присутність та має більший потік доходів, ніж у Android. Операційна система Android продовжує зростати як на ринках, що розвиваються, так і на розвинених, окрім ринку Китаю, хоча все ще залишається найбільш популярною операційною системою в цій країні [1].

На сьогоднішній день сфера засобів для моніторингу стану здоров'я на платформі Android переживає значний ріст і розвиток. Багато засобів взаємодіють із сучасними смарт-пристроями, такими як фітнес-годинники, датчики серцебиття, глюкометри тощо. Це дозволяє отримувати більш точні та докладні

дані. Проте, незважаючи на наявність численних медичних додатків для Android, багато з них обмежені функціональністю, не надають засобів аналізу та взаємодії зі зібраними даними, та не дозволяють користувачам зручно створювати звіти або експортувати дані в зручних форматах.

Метою дослідження є аналіз сучасного стану і тенденцій розвитку програмних засобів у сфері моніторингу та аналізу стану здоров’я та розробка програмного засобу, який дозволить користувачам зручно вести облік симптомів, болю та інших показників стану здоров’я, проводити їх простий аналіз, формувати звіти та експортувати їх в форматі PDF.

Задачі дослідження: аналіз актуальних досліджень та існуючих програмних засобів для моніторингу здоров’я на платформі Android, розробка інтерфейсу програмного засобу, реалізація аналітичних інструментів, створення функціоналу для формування звітів, тестування розробленого додатку, проведення розрахунків економічної ефективності.

Об’єктом дослідження є програмні засоби для моніторингу стану здоров’я на платформі Android.

Предметом дослідження є розроблений програмний засіб для обліку, аналізу та моніторингу стану здоров’я на платформі Android.

Методи дослідження передбачає аналіз існуючих програмних рішень для моніторингу здоров'я на платформі Android з метою виявлення недоліків і можливостей для покращення. Далі передбачається розробка програмного засобу з використанням мови програмування Java та впровадження в нього інструментів для моніторингу та аналізу даних. Проведення тестування та апробації додатку дозволить оцінити його ефективність та корисність.

Практичне значення роботи полягає в наданні середньостатистичним користувачам зрозумілий спосіб об’єктивного моніторингу їхнього стану здоров’я, допоможе вчасно виявляти зміни в стані здоров’я. Користувачі матимуть доступ до своїх даних, що сприяє більшій самосвідомості щодо їхнього здоров'я та забезпечує індивідуалізований підхід до медичної допомоги. Це

особливо важливо для людей, які мають хронічні захворювання або повинні постійно спостерігати за певними показниками.

Наукова новизна роботи полягає у тому, що вперше було запропоноване об’єднання в одному програмному засобі для моніторингу та аналізу стану здоров’я використання індивідуалізованого підходу до моніторингу стану здоров'я користувача, спрощення процесу створення записів, впровадження моделі візуалізації даних для аналізу тенденції показників стану здоров'я, формуванні звітності та акценті на підвищенні рівня самосвідомості користувача щодо його здоров'я. Що створить доволі помітний внесок у розвиток інструментів для особистого моніторингу та аналізу стану здоров'я.

Апробація результатів роботи здійснена в доповіді на IX Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем (СПІРН—2023).

Матеріали роботи доповідались та опубліковувались [2]:

В. І. Ковалик / МОБІЛЬНА МЕДИЦИНА ТА РОЛЬ ПРОГРАМНИХ ЗАСОБІВ У МОНІТОРИНГУ СТАНУ ЗДОРОВ’Я НА ПЛАТФОРМІ ANDROID // Тези доповіді. IX Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем (СПІРН—2023). Вінниця 2023 р. Режим доступу: https://conferences.vntu.edu.ua/
index.php/spirn/spirn2023/paper/viewFile/19169/15893

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ МОНІТОРИНГУ СТАНУ ЗДОРОВ’Я НА ПЛАТФОРМІ ANDROID

1.1 Огляд сфери застосування програмних засобів для моніторингу стану здоров’я на платформі Android

Сучасний світ переживає суттєві зміни в підходах до догляду за здоров'ям, діагностики та контролю за фізичним станом. Однією з ключових тенденцій цього процесу є зростання використання мобільних технологій та програмних засобів для моніторингу стану здоров'я. Зокрема, на платформі Android, яка є однією з найпоширеніших операційних систем для смартфонів та планшетів, розроблено широкий спектр додатків, які надають користувачам засоби для ведення контролю над їхнім фізичним та психічним станом.

Приріст числа смартфонів та планшетів з підтримкою Android дозволяє все більшій кількості людей користуватися медичними додатками для ведення контролю за своїм станом здоров'я. Ці програмні засоби надають можливість вести записи про фізичні показники, фізичну активність, спостерігати за симптомами, нагадувати про прийом ліків, аналізувати дані, проводити консультацію з лікарем чи іншим спеціалістом.

Провівши аналіз сфер застосування програмних засобів для моніторингу за станом здоров’я, можна сказати, що сфера застосування таких засобів на платформі Android є дуже різноманітною і розширюється з кожним роком завдяки зростанню популярності смартфонів та розвитку технологій. Розглянемо одні з основних сфер застосування таких засобів.

Програмні засоби для спорту та фітнесу стали дуже популярними, оскільки дозволяють людям відстежувати свій фізичний стан, спортивні досягнення та дотримуватися здорового способу життя. Популярними додатками є Fitbit, Garmin Connect, і програми, які інтегровані з Apple Watch та Android Wear. Додатки для тренувань надають доступ до тренувальних програм і вправ, що допомагають поліпшити фізичну форму. Прикладами таких додатків є Nike Training Club, MyFitnessPal, JEFIT тощо. Додатки для спортивного аналізу допомагають спортсменам та спортивним командам аналізувати їх фізичні показники та

досягнення. Наприклад, Strava для велосипедистів та бігунів, або Swim.com для плавців. Ці додатки зазвичай інтегруються з різними фітнес-пристроями, такими як браслети, годинники та ваги, і допомагають збирати та аналізувати дані про стан здоров'я користувачів, щоб допомогти їм підтримувати здоровий спосіб життя та досягати своїх цілей в області фізичного здоров'я.

Програмні засоби для дієтології та контролю харчування грають важливу роль в спостереженні за станом здоров'я та досягненні харчових цілей. Вони надають можливість користувачам вести облік спожитих продуктів, калорій, жирів, білків, вуглеводів, поживних речовин та інших аспектів харчування. Такі додатки для ведення дієти, рецептів та обліку калорій допомагають користувачам керувати своєю їжею і вести здоровий спосіб життя. В таких додатках користувач може самостійно вказати свої харчові вподобання та потреби, можуть мати базу даних з тисячами продуктів та ресторанів, які полегшують введення інформації. Прикладами таких додатків є MyFitnessPal, Lose It!, Yazio та FatSecret тошо.

Програмні засоби для спостереження за сном допомагають користувачам відстежувати якість та тривалість їхнього сну, а також збирати дані про сон для аналізу та поліпшення режиму сну. Вони можуть бути корисними для тих, хто страждає від проблем із сном або просто хоче поліпшити якість свого сну. Такі додатки зазвичай використовують можливості вашого смартфона чи фітнес-браслета, щоб відстежувати ваші рухи під час сну, фази глибокого та поверхневого сну, на основі даних вмикає будильник в оптимальний час для пробудження. Також можуть надавити графік вашого сну та статистику. Прикладами таких додатків є Sleep Cycle, Pillow, Fitbit, SnoreLab тощо. Ці додатки надають користувачам засоби для ведення обліку сну, аналізу його якості та отримання рекомендацій для поліпшення режиму сну. Вони можуть бути корисними інструментами для підтримки здорового сну та зменшення проблем із сном.

Програмні засоби для психічного здоров’я стають все більш популярними, оскільки психічне здоров'я стає важливим аспектом загального здоров'я та добробуту. Такі додатки допомагають користувачам вести облік свого емоційного

стану, пропонують методи медитації та релаксаційні практики, вправи для зменшення стресу, різноманітні програми для поліпшення психологічного стану, конфіденційні консультації від професіоналів або просто вести розмови з людьми, які проходять схожі емоційні випробування. Деякі додатки використовують штучний інтелект для надання психологічної підтримки. Прикладами таких додатків є Headspace, Calm, 7 Cups, Youper тощо. Ці додатки можуть бути корисними для тих, хто бажає підтримувати психічне здоров'я, навчатися релаксації та медитації, та шукати підтримку у складних моментах. Хоча важливо пам'ятати, що додатки не можуть замінити професійну медичну допомогу, і якщо ви стикаєтеся з серйозними психічними проблемами, важливо звертатись до кваліфікованого фахівця.

Програмні засоби для медичного обліку допомагають користувачам отримати доступ до медичних записів та результатів аналізів, надає можливість планувати прийом до лікаря, завантажувати фотографії рецептів та лікарських препаратів, можуть надавати інформацію про симптоми та хвороби, дозволяє перевіряти рівень цукру в крові і спілкуватися з медичними працівниками. Прикладами таких додатків є MyChart, WebMD, Dexcom G6, Medisafe тощо.

Програмні засоби для дитячого здоров’я та вагітності грають важливу роль в підтримці батьків в період вагітності, народження та росту дитини. Вони надають інформацію, рекомендації та інструменти для ведення обліку та стеження за станом здоров'я батьків та малюка. Такі додатки надають інформацію про розвиток плоду на кожному етапі вагітності, поради від фахівців, форуми для обміну досвідом з іншими майбутніми батьками, допомагають відстежувати розвиток дитини після народження, вести облік прийому медикаментів і відстежувати щеплення. Прикладами таких додатків є What to Expect, The Bump, BabyCenter, Ovia Pregnancy Tracker тощо. Ці додатки допомагають батькам стежити за розвитком дитини, отримувати корисну інформацію і поради. Вони можуть бути корисними для підтримки батьків у цьому важливому періоді життя.

Програмні засоби для контролю медикаментів використовують для нагадування про прийом ліків та контроль лікування, користуються популярністю

серед тих, хто має хронічні хвороби або використовує рецептурні препарати. Вони можуть бути корисними для тих, хто приймає багато різних ліків або хоче впевнитися, що дотримується правильного розкладу прийому медикаментів. Прикладами таких додатків є Medisafe, Pillboxie, CareZone, MyTherapy тощо. Ці додатки допомагають спростити процес контролю медикаментів, знизити ризик помилкового прийому ліків та надають корисну інформацію про ліки.

Програмні засоби для діагностики та моніторингу хвороб застосовують для відстеження свого стану здоров’я, симптомів та результати медичних тестів. Такі додатки дозволяють користувачеві ввівши перелік симптомів отримати інформацію про можливі хвороби та лікування, можуть використовувати штучний інтелект для діагностики симптомів та надавати рекомендації щодо подальших дій. Прикладами таких додатків є Ada, WebMD, HealthTap, Ada Health, mySymptoms тощо. Ці додатки можуть бути корисними для самостійного контролю стану здоров'я, пошуку інформації про симптоми і можливі хвороби, а також для спілкування зі своїми лікарями і отримання консультацій. Проте важливо пам'ятати, що додатки не замінюють професійної медичної діагностики та лікування, і завжди рекомендується звертатися до лікаря для отримання точного діагнозу і плану лікування [2].

Окремо хочеться виділити сферу програмних засобів для загальної освіти про здоров’я. Такі додатки грають важливу роль у підвищенні медичної грамотності та надають користувачам доступ до авторитетних медичних джерел, статей, наукових досліджень, порадам щодо здорового харчування, фізичної активності, контролю стресу, гігієни, різним профілактикам хвороб та іншим аспектам здорового способу життя. Прикладами таких додатків є KidsDoc, TeenMentalHealth, Food Hero, Anatomy 4D тощо. Ці додатки сприяють зростанню медичної грамотності серед користувачів та допомагають їм краще розуміти та керувати своїм здоров'ям. Важливо пам'ятати, що інформація, яку надають ці додатки, не завжди замінює консультацію з лікарем, і важливо довіряти авторитетним джерелам та звертатися до фахівців при серйозних питаннях стосовно здоров'я.

Отже, сфера застосування програмних засобів для моніторингу за станом здоров'я на платформі Android є вкрай різноманітною і важливою. Наведені приклади додатків допомагають користувачам підтримувати здоровий спосіб життя, контролювати стан свого здоров'я, покращувати фізичну активність та дотримуватися дієти. Вони також забезпечують інформацією та надають підтримку для тих, хто стикається з хронічними хворобами, проблемами зі сном, стресом чи психічними розладами.

Завдяки зручності смартфонів та інтеграцією з різними смарт-пристроями, наведені вище приклади додатків дозволяють покращити доступність та якість медичних послуг. Сфера застосування програмних засобів і надалі буде розвиватися та розширюватися, надаючи все більше можливостей для підтримки, моніторингу стану здоров’я та покращення якості життя користувачів.

1.2 Аналіз напрямів дослідження та їх технічної реалізації для моніторингу стану здоров’я

За останні декілька років спостерігається виразний ріст інтересу до розробки та використання програмних засобів для моніторингу стану здоров'я на платформі Android, і це не дивно, оскільки ці засоби можуть впливати на покращення життя та здоров'я мільйонів людей. Дослідження в сфері програмних засобів для моніторингу стану здоров'я на платформі Android включають різні напрямки та технічні реалізації.

Наприклад одним з напрямків досліджень є вимірювання фізіологічних показників, так називають процеси збору даних різних параметрів, які вказують на функціонування організму та його стан здоров’я. Такі показники можуть бути виміряні різними способами та застосовуються для медичних досліджень, діагностик та моніторингу здоров’я. Одними з основних фізіологічних показників є серцевий ритм, кров’яний тиск, рівень кисню у крові, температура тіла, пульс, рівень глюкози в крові тощо.

За допомогою сучасного смартфона можна визначити всі вище перераховані фізіологічні показники. Для визначення серцевого ритму деякі

смартфони мають вбудовані сенсори для вимірювання серцевого ритму через відбиття пульсу від пальця, долоні або іншої ділянки тіла. Також існують додатки, які використовують камеру смартфона для вимірювання серцевого ритму за допомогою алгоритмів обробки зображень [3]. Для вимірювання температури тіла деякі смартфони мають вбудовані термометри, а також існують додатки, які використовують додаткові датчики для цього вимірювання. Для вимірювання пульсу може бути виконано за допомогою вбудованих сенсорів смартфона або спеціальних додатків, які використовують камеру смартфона для вимірювання пульсу шляхом аналізу змін в кольорі шкіри. Для того щоб виміряти рівень кисню в крові (SpO2) деякі сучасні смартфони мають вбудовані сенсори, які визначають рівень кисню в крові через відбиття світла від шкіри.

Однак для деяких показників та для більш точних вимірювань може знадобитися зовнішній пристрій, який підключається до смартфона через USB/Type-C/microUSB-роз'єм або за допомогою Bluetooth та спеціального додатка для зчитування та аналізу даних. Наприклад такі показники як рівень глюкози в крові, кров'яний тиск, рівень кисню в крові тощо. Важливо зауважити, що точність таких вимірювань може бути різною і залежить від якості сенсорів та алгоритмів обробки даних. Для медичних діагнозів та важливих вимірювань рекомендується користуватися медичними пристроями, а не смартфонами. Додатково, завжди важливо дотримуватися інструкцій виробника та консультуватися з медичним фахівцем щодо правильного вимірювання та інтерпретації результатів. Однак прогрес не стоїть на місці, і можливо в нетакому далекому майбутньому, нам буде достатньо одного смартфона для вимірювання всіх перелічених вище і не тільки показників [5].

1.3 Огляд існуючих рішень та додатків для моніторингу стану здоров'я

Під час розгляду питання про створення програмного засобу для моніторингу стану здоров'я користувача було визначено, що найбільш поширеною практикою є використання технологій, що працюють ексклюзивно на платформах iOS або Android.

Таким чином, проведено аналіз програмних засобів, розроблених відомими компаніями.

1.3.1 Аналіз програмного рішення Apple HealthKit

Даний програмний засіб поліпшує надання медичних послуг. Застосунок зберігає інформацію про стан здоров'я та фізичну активність користувача на пристроях iPhone та Apple Watch. Користувач може передавати ці дані своєму лікарю. Розроблено для операційної системи iOS з використанням мови програмування Swift.

HealthKit надає центральне сховище даних про здоров’я та фізичну активність на iPhone та Apple Watch. З дозволу користувача програми зв’язуються зі сховищем HealthKit для доступу до цих даних і обміну ними.

Користувачі можуть вибрати свій будь який додаток для відстеження ваги, підрахунку кроків і здоров’я, кожен з яких відкалібрований відповідно до їхніх особистих потреб. Це дозволяє HealthKit вільно обмінюватися потрібними даними з іншими додатками (з дозволу користувача), об’єднаний пакет забезпечує більш персоналізований досвід, ніж будь-яка окрема програма окремо. Ілюстрацію роботи додатка зображено в рисунку 1.1.

Рис. 1.1 — Ілюстрація роботи Apple HealthKit

Навіть при значній популярності цього додатка, виявлено численні недоліки. Важливо зазначити, що він функціонує лише на платформі iOS, обмежуючи можливість використання його на інших пристроях. Крім того, відсутність інтернаціоналізації обмежує його доступність для користувачів, які володіють іншими мовами, крім англійської.

Перевагами даного програмного засобу є зрозумілий інтерфейс, розширені функціональні можливості, можливість роботи без доступу до Інтернету та здатність завантажувати та обмінюватися документами з лікарем. Важливо врахувати, що цей додаток обмежений платформою iOS і має підтримку тільки англійської мови.

1.3.2 Аналіз програмного рішення Google Fit

Програма збирає різноманітну інформацію про стан здоров'я користувача, якою він може поділитися зі своїм лікарем. Додаток пропонує широкий спектр можливостей, до яких входить відстеження фізичної активності, кількості спожитих калорій та гідратація, відстеження циклів сну та їх тривалість, антропометричні дані та відстеження життєвих показників таких як пульс, кров’яний тиск, частоту дихання, рівень глюкози в крові, насиченість крові киснем та температуру тіла.

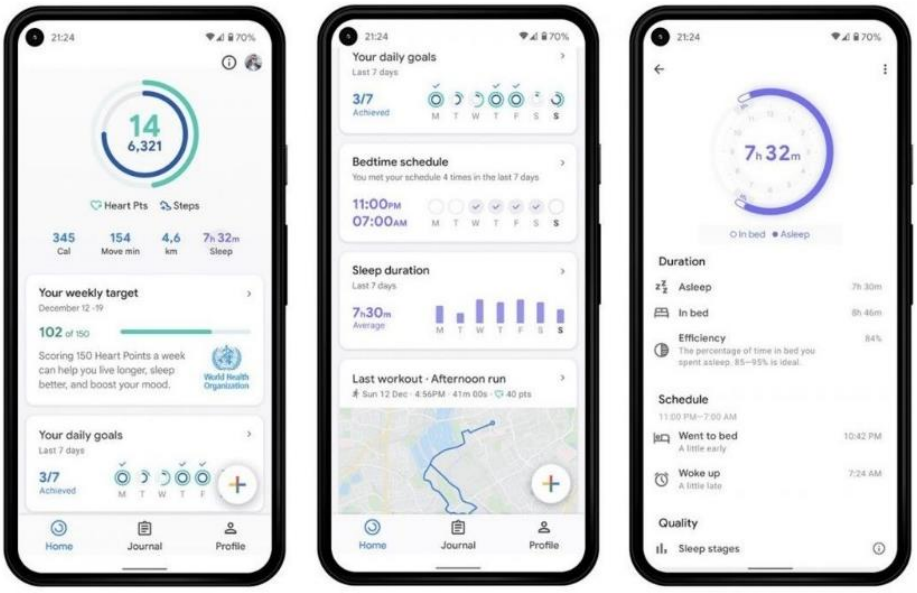


Рис. 1.2 — Ілюстрація роботи Google Fit

1.3.3 Аналіз програмного рішення Medisafe

Програмний засіб Medisafe відзначається значно розширеним функціоналом порівняно з його попередником. В ньому можна вести відстеження будь-яких прописаних ліків, включаючи як регулярні, так і ті, які ви вживаєте за потребою (наприклад, знеболюючі чи снодійні). Налаштувати сповіщення про необхідний прийом медикаментів. Для консультацій з лікарем ви можете вводити дані щодо тиску, рівнів глюкози в крові, ваги та інших параметрів. Існують і менш значущі, але зручні опції, такі як синхронізація нагадувань з годинником Android Wear і можливість призначати різні мелодії для окремих препаратів. На жаль, наразі розробники не включили функцію, яка вказувала б початок і кінець прийому ліків, кількість вже вжитих таблеток і залишкову кількість (програма наразі враховує лише число днів курсу), і відсутня можливість встановлення перерв у прийомі медикаментів.

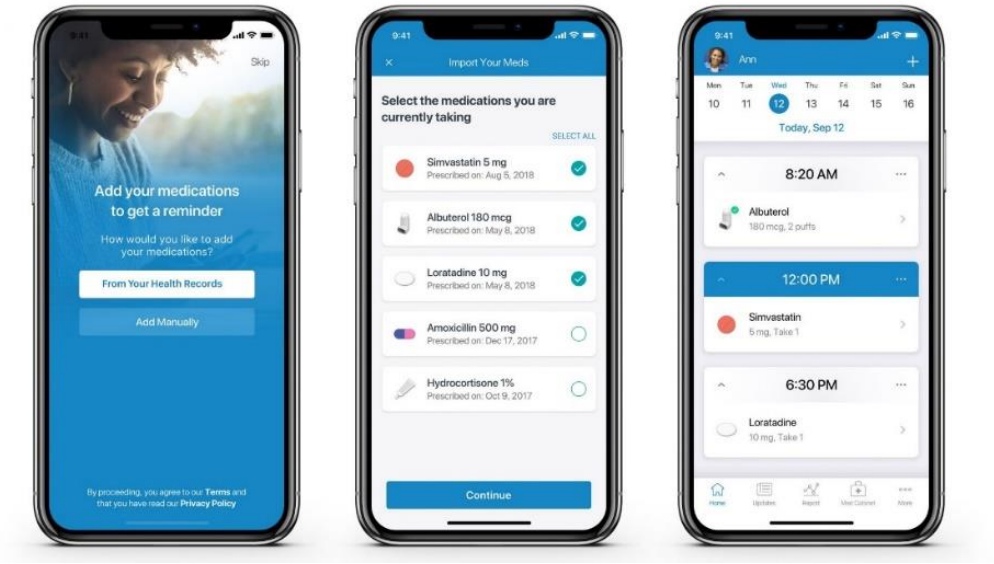


Рис. 1.3 — Ілюстрація роботи Medisafe

Проведений аналіз даного програмного рішення вказує на ряд його переваг, таких як багатофункціональність, можливість роботи в умовах відсутності мережі Інтернет, підтримка інтернаціоналізації та наявність інтуїтивно-зрозумілого інтерфейсу для взаємодії з системою. Проте важливо врахувати, що цей додаток не є кросплатформеним.

2 МЕТОДОЛОГІЯ ТА МАТЕМАТИЧНІ МОДЕЛІ ПРОГРАМНОГО ЗАСОБУ

2.1 Модель візуалізації даних для аналізу показників стану здоров'я

Перед розробкою програмного засобу було важливо обрати математичну модель, яка належним чином враховує різноманітність показників стану здоров'я та забезпечують оптимальний рівень адаптованості до введених користувачем даних. З огляду на завдання програми, вибір пав на модель візуалізації даних.

Візуалізація даних грає важливу роль у представленні та розумінні інформації, особливо коли мова йде про стан здоров'я користувача. Це ефективний спосіб зробити великий обсяг даних більш зрозумілим і доступним для аналізу.

Прийняття інформації шляхом сприйняття візуальних зображень є більш ефективним та дозволяє швидше і ефективніше передавати тенденцію користувачу про стан здоров’я. Дослідження підтверджують, що для людини основним є сприйняття інформації за допомогою зору, і працюючи з візуальною інформацією, продуктивність може зростати на 17%. Використання візуальної інформації допомагає зменшити інформаційне навантаження та утримує увагу [8]. Едвард Тафті, автор провідних книг про візуалізацію, розглядає її як інструмент для відображення даних, стимулювання користувача думати про суть, а не про методологію, і сприяння порівнянню фрагментів інформації.

Графіки та діаграми є потужними інструментами візуалізації даних, які допомагають перетворити складні числові або кількісні інформаційні дані у зрозумілу, легко сприйману форму. Використовуючи різні типи графіків та діаграм, можна визначити тенденції, порівнювати дані, виявляти аномалії та сприяти кращому розумінню зв'язків у наборі даних. Розглянемо деякі типи графіків.

Лінійні графіки використовують лінії для з'єднання точок значень на вісі X та Y. Вони ідеально підходять для відображення тенденцій часового ряду. Використовуються для відстеження змін в часі, порівняння динаміки кількох серій даних [8]. Приклад графіку наведено в рисунку 2.1.

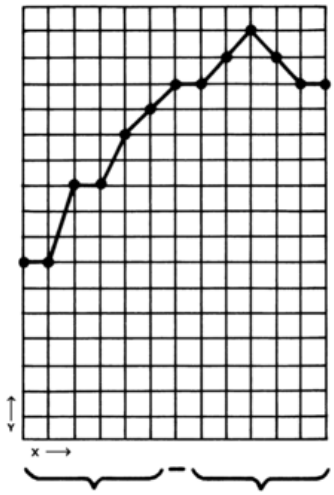


Рис. 2.1 — Приклад лінійного графіку

Стовпчикові графіки використовують стовпці для представлення значень категорій на вісі X та величини на вісі Y. Використовують для порівняння значень між різними категоріями, відображення дискретних даних [9]. Приклад графіку наведено в рисунку 2.2.

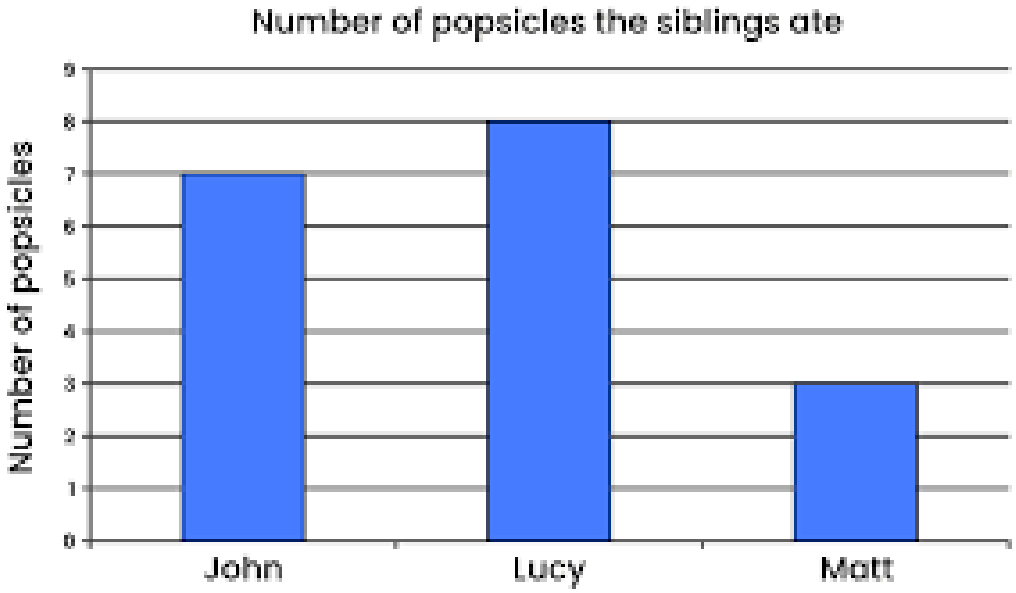


Рис. 2.2 — Приклад стовпчикового графіку

Кругові або секторні діаграми представляють відсоткове співвідношення різних частин відносно цілого кола (360 градусів). Використовують для відображення часток в цілому, виділення важливих частин у відсотковому співвідношенні [10]. Приклад діаграми наведено в рисунку 2.3.

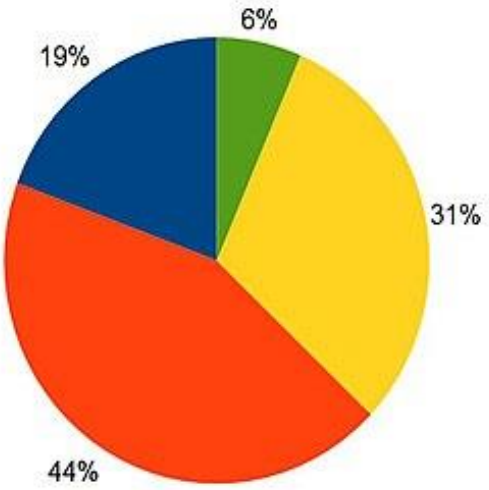


Рис. 2.3 — Приклад кругової діаграми

Гістограми розташовують стовпці поруч, представляючи інтервали значень на вісі X та їх частоту на вісі Y. Використовують для виявлення розподілу даних, оцінка частоти відносно конкретних інтервалів [11]. Приклад діаграми наведено в рисунку 2.4.

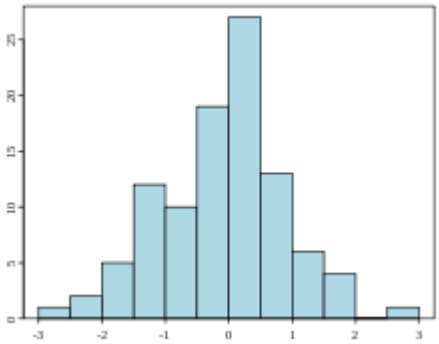


Рис. 2.4 — Приклад гістограми

Графіки та діаграми є невід'ємною частиною інструментарію аналізу даних та надають зручний спосіб відображення складних взаємозв'язків між різними параметрами.

У контексті розроблюваного програмного засобу для аналізу стану здоров'я, найбільш ефективним та інформативним є використання лінійних графіків. Цей тип графіка ідеально підходить для відображення тенденцій у часовому ряді, надаючи користувачеві можливість легко визначати зміни в показниках протягом певного періоду. Лінійний графік дозволяє точно відслідковувати динаміку

показників здоров'я та виявляти потенційні зміни, що робить його важливим інструментом для аналізу та прийняття обґрунтованих рішень в галузі здоров'я.

2.2 Індивідуалізований підхід до моніторингу стану здоров’я

Індивідуалізований підхід до моніторингу стану здоров'я означає використання персоналізованих стратегій та технологій для визначення та відстеження показників здоров'я кожної конкретної людини. Замість загальних підходів до оцінки здоров'я населення в цілому, індивідуалізований моніторинг спрямований на урахування унікальних характеристик, генетичних особливостей, способу життя та інших факторів, які впливають на здоров'я кожної окремої особи.

Цей підхід часто базується на використанні новітніх технологій, таких як носимі пристрої (фітнес-трекери, смарт-годинники, сенсори сну тощо) для моніторингу фізіологічних показників, мобільні додатки для відстеження активності та здоров'я, генетичні аналізи та інші методи. Важливою частиною індивідуалізованого підходу є аналіз отриманих даних та розробка персоналізованих рекомендацій для поліпшення та підтримки здоров'я конкретної особи.

Індивідуалізований підхід в програмному засобі для моніторингу здоров'я може бути реалізований через ряд функцій та можливостей, які дозволяють користувачеві персоналізувати свій досвід та взаємодію з додатком. Наведемо декілька ключових елементів, які можуть бути включені до даного підходу.

Створення власних категорій симптомів або показників, які важливі для його здоров'я. Це може включати різні аспекти, такі як фізичні симптоми, показники настрою, рівень енергії тощо.

Створення записів симптомів, тому програмний засіб повинен дозволяти користувачу регулярно вводити записи щодо своїх симптомів для кожної обраної категорії. Користувач може вносити дані щодня, визначений період часу або при настанні конкретних подій.

Проведення простого аналізу, програмний засіб може автоматично або за запитом користувача виконувати простий аналіз введених даних. Це може включати визначення тенденцій, кореляцій чи інших важливих аспектів здоров'я на основі введених симптомів.

Створення звітів з спостереження, щоб користувач міг генерувати звіти зі своїми спостереженнями, щоб легше розуміти і аналізувати свій стан здоров'я. Ці звіти можуть включати повний перелік потрібних симптомів, показників, графіки, діаграми або інші візуальні елементи для зручного сприйняття інформації.

Персоналізовані рекомендації на основі введених даних та аналізу також є ключовим елементом, однак поки що рівень технологій ще не на такому високому рівні щоб програмний засіб міг надавати користувачеві персоналізовані рекомендації, щодо його стану здоров’я. Адже для надання рекомендацій потрібна медична освіта, мати доступ до медичної бази та досить довгому випробному періоді під пильним наглядом лікарів, що на момент створення магістерської роботи є відсутнім та дуже ресурсозатратним процесом.

На основі проведеного аналізу існуючих програмних рішень та засобів, можна дійти висновку що не використовується або частково використовується індивідуалізований підхід для моніторингу та аналізу стану здоров’я. Оскільки більшість програмних засобів має обмежений функціонал, має невеликий перелік відслідковуваних показників, та не має можливості створювати власні категорії симптомів, створювати звіт з спостереження для консультації у лікаря та в багатьох випадках відсутня можливість простого аналізу або застосовують функціонал окремо ніяк не пов’язуючи його між собою.

Тому пропоную об’єднати в одному програмному засобі для моніторингу та аналізу стану здоров’я використання індивідуалізованого підходу до моніторингу стану здоров'я користувача, спрощення процесу створення записів за допомогою створення категорій, що дозволить пришвидшити процес створення записів, оскільки користувачу не потрібно буде кожного разу вводити назву симптому/показника та виключить можливість не правильного введення назви симптому/показника, впровадження моделі візуалізації даних для аналізу

тенденції показників стану здоров'я, формуванні звітності та акценті на підвищенні рівня самосвідомості користувача щодо його здоров'я. Що створить доволі помітний внесок у розвиток інструментів для особистого моніторингу та аналізу стану здоров'я.

2.3 Методи реалізації програмного засобу для моніторингу та аналізу стану здоров'я

Під час розробки даного програмного продукту було використано модель бізнес-логіки програмного забезпечення. Цей підхід відкриває значні можливості для розширення функціональності системи протягом її життєвого циклу без значних переробок і дозволяє одночасно працювати над розробкою різних її аспектів групам розробників.

При аналізі існуючих рішень, прийнято рішення реалізувати програмний інструмент для ведення записів, моніторингу та аналізу стану здоров'я та можливістю формувати звіт в форматі PDF на мобільній платформі Android. Основною перевагою смартфона є його мобільність, яка забезпечує доступ до програмного засобу в будь-якому місці та часі. Також, портативність пристроїв дає користувачеві зручність завжди бути поблизу пристрою та можливість швидко створювати записи.

Програмний засіб має бути побудований на мобільній платформі "Android" для використання на пристроях, які працюють під управлінням даної операційної системи. Мовою програмування обрано об'єктно-орієнтовану "Java". Для створення додатку використовується інтегроване середовище розробки "Android Studio", а побудова проєкту здійснюється за допомогою системи автоматичної збірки "Gradle".

Для вирішення поставленої задачі знадобляться наступні інструменти:

— персональний комп’ютер (PC);

— середовище розробки Android Studio від компанії Google;

— автоматизована система збірки Gradle;

— емулятор або фізичний пристрій на платформі Android.

2.3.1 Платформа Android

Платформа "Android" виникла як результат співпраці групи "Open Handset Alliance" з метою створення найбільш передової мобільної системи. У контексті розробки програмного забезпечення "Android" виступає як вагомий учасник в області відкритого програмного забезпечення. Перший пристрій, що працював під управлінням цієї системи, був представлений у 2008 році, і для розробки програм для нього використовувався постійно оновлюваний набір розробки "SDK".

Платформа "Android" не поступається операційним системам для персональних комп'ютерів за своїми можливостями. Вона базується на багаторівневій системі з ядром "Linux" і включає в себе вікна, представлення та віджети в підсистемі користувацького інтерфейсу. "Android" підтримує різні засоби підключення, такі як "Wi-Fi", "Bluetooth" та протоколи передачі даних через стільникову мережу. Крім того, вона має вбудовану підтримку сервісів, що ґрунтуються на визначенні місцеположення та акселерометрах, хоча не всі пристрої на цій платформі обладнані відповідним обладнанням. Також є підтримка відеокамери[12].

Платформа "Android" вирішує проблеми графіки завдяки вбудованій підтримці, включаючи бібліотеку "OpenGl", і спрощує завдання збереження даних завдяки наявності популярної бази даних з відкритим кодом "Sqlite".

Операційна система "Android" базується на ядрі "Linux", і для написання додатків використовується мова програмування "Java". Виконання додатків відбувається у віртуальній машині, проте важливо відзначити, що вона не є "JVM". Замість цього використовується відкрита технологія "Dalvik Virtual Machine". Кожен додаток "Android" стартує в екземплярі "Dalvik VM", який вбудований у контрольо-
ваний ядром "Linux" процес.

Компанія Google вкладає значні ресурси в оптимізацію та покращення продуктивності платформи "Android". Оптимізація спрямована на зменшення

розміру, підвищення швидкості, економію енергії акумулятора та зменшення обсягу використаної пам'яті[13].

Архітектура "Android" пропонує унікальні переваги, такі як високий рівень однорідності. Більшість додатків "Android" можуть легко працювати на різних пристроях, що базуються на цій платформі, без додаткових модифікацій. До складу "Android" входить набір базових додатків, таких як клієнти електронної пошти, календар, карти, браузер і програма для управління контактами.

Платформа "Android" дозволяє використовувати всю потужність, що використовується в додатках ядра. Кожен додаток може використовувати можливості іншого додатка, якщо той надає доступ до своєї функціональності. Таким чином, архітектура реалізує принцип багаторазового використання компонентів і додатків. Схему архітектури системи Android зображено на рисунку Б.1 див. Додаток Б.

Android пропонує розширений та організований спектр високорівневих API-інтерфейсів для розробки додатків та максимального використання можливостей платформи. Ці інтерфейси відзначаються високим рівнем абстракції, зробивши їх доступними та легкими у використанні під час розробки.

Зовнішні додатки мають можливість взаємодіяти із практично всіма ключовими компонентами платформи Android. Наприклад, доступна можливість отримання інформації з адресної книги користувача, можуть висилати та приймати бродкаст-повідомлення для сповіщення про події або зміни стану, за допомогою інтентів додатки можуть викликати функції інших додатків або системних компонентів. Також додатки можуть взаємодіяти один з одним через власні служби та сервіси. Вони можуть обмінюватися даними, запускати фонові завдання або спільно вирішувати певні завдання Використання API надає можливість створювати додатки, що повністю інтегруються з іншими елементами платформи. Набір інструментів для віджетів у системі Android пропонує широкий вибір надзвичайно

корисних компонентів вже на початковому етапі. Деякі віджети спеціально розроблені для ефективної взаємодії з дотиками, включаючи вбудовані функції,

такі як кінетична прокрутка. Розробники можуть використовувати мову опису інтерфейсу користувача на основі XML для визначення макету та атрибутів віджетів, при цьому описи XML завантажуються в програму через систему ресурсів Android.

На окремі віджети, що описані в макеті XML, можна посилатися по їхньому ідентифікатору в програмі. Також існує можливість програмного створення віджетів та управління ними в користувацькому інтерфейсі під час виконання програми.

Найзручнішим методом для створення макетів є ручне написання XML-описів. Хоча Android SDK включає вбудований інструмент для візуального макету, він не охоплює всі виджети і не завжди працює надійно.

Додатки Android складаються з провайдерів, служб, приймачів та активностей. Усі ці компоненти мають визначені завдання в стеку додатків Android. Їх взаємодія взаємопов'язана, що забезпечує Android високий рівень модульності[15].

Провайдери контенту служать рівнем абстракції для взаємодії з різноманітними джерелами даних та обміну постійними даними між додатками. Вони стандартизують інформаційний доступ за допомогою синтаксису URI, але розробники додатків часто використовують багаторівневі класи обгортки для управління запитами.

Система провайдерів контенту надає декілька переваг для розробників Android. Вона забезпечує високий рівень взаємодії, дозволяючи додаткам обмінюватися даними за уніфікованим методом. Багато ключових джерел даних, таких як контактні списки та системи збереження мультимедіа, доступні через інтерфейси провайдерів контенту за замовчуванням, що спрощує їх використання для сторонніх додатків.

Важливою особливістю Android, що відрізняє її від інших платформ, є підтримка фонових процесів у сторонніх додатках. Це реалізовано за допомогою сервісного компонента Android. Сервіси — це операції без заголовку, які працюють в фоновому режимі тривалий час.

Система повідомлень Android схожа на систему сигналів в Linux. Розробники вбудовують приймачі, які реагують на визначені системні повідомлення чи події, і виконують певні дії. За допомогою системи мовлення можна відслідковувати базові системні події, такі як стан акумуляторної батареї, вхідні SMS-повідомлення, натискання апаратних кнопок, зміна з'єднання тощо.

Система Android Intents відіграє ключову роль у взаємодії всіх цих компонентів. Об'єкти Intents, які містять ідентифікатор дії та URI даних, використовуються для виклику дій, служб та отримувачів. Ідентифікатор дії вказує бажане поведінку, а URI даних, як необов'язковий, визначає місце розташування даних, над якими має виконуватися дія. Загальну схему компонентів в програмних засобах Android зображено на рисунку 2.6.

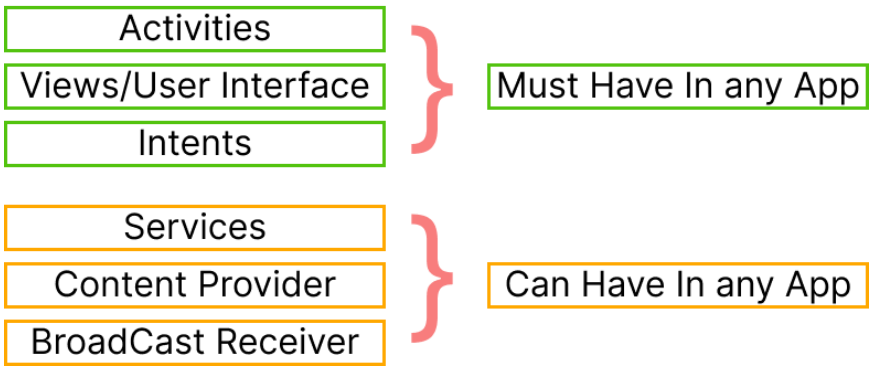


Рис. 2.6 — Схема компонентів в програмних засобах Android

Однією з ключових переваг системи Android є її принцип відкритості. Операційна система Android побудована на відкритому вихідному коді та розповсюджується на безкоштовній основі. Це дає можливість розробникам отримати доступ до вихідного коду та розуміти, як реалізовані властивості та функції додатків. Кожен користувач може приймати участь у вдосконаленні операційної системи.

2.3.2 Мова програмування Java

Для створення даного проєкту використовувалася мова програмування Java, яка на даний момент вважається однією з найпоширеніших та високо оцінюваних.

Розробка Java була ініційована Джеймсом Гослінгом та його командою в компанії Sun Microsystems у 1991 році, і на завершальній стадії вона отримала назву Java, хоча на початку вона була відома як Oak. Основною метою було створення мови, яка б не залежала від конкретної платформи і можна було використовувати для розробки програмного забезпечення для різноманітних побутових пристроїв, таких як мікрохвильові печі, різні дистанційні пристрої та контролери з різними процесорами та архітектурою. Недоліком існуючих мов C та C++ було необхідність компіляції програм для конкретної платформи, що вимагало значних зусиль та ресурсів як фінансової, так і часової.

Створення мови Java було визначене потребою у мові, незалежній від конкретних процесорів, яку можна було використовувати для написання програмного коду, що виконується на різних платформах та в різних середовищах. Приблизно у той самий час, коли формувалися основні особливості мови Java, виникла і широко поширена всесвітня мережа Інтернет. Це значно вплинуло на майбутнє мови програмування Java, яка могла б залишитися корисною, але непомітною для написання програмного коду для побутових пристроїв. Поява Інтернету сприяла значному підвищенню популярності мови Java серед розробників, а сама Java сильно вплинула на розвиток Інтернету. Мова Java не лише спростила процес розробки програм для Інтернету взагалі, але також призвела до створення нового типу прикладних програм, спрямованих на роботу в мережі, що змінило уявлення про середовище мережі. Більше того, Java дозволила вирішити дві основні проблеми програмування того часу, пов'язані з Інтернетом — це безпека та крос-
платформність.

Java відрізняється особливістю, яка вирішує важливі завдання. Компілятор Java не формує виконуваний код, а замість цього створює байт-код. Цей байт-код складається з високооптимізованого набору інструкцій, орієнтованих на виконання в межах віртуальної машини Java (JVM — java virtual machine). Початкова версія віртуальної машини JVM фактично була створена як інтерпретатор байт-коду. Це може викликати певні непорозуміння, оскільки

багато сучасних мов програмування генерують виконуваний код для досягнення максимальної продуктивності. Однак саме те, що програми на Java інтерпретуються віртуальною машиною JVM, сприяє розв'язанню ключових проблем у розробці програмного забезпечення для Інтернету.

Перетворення програми Java в байт-код робить її виконання більш універсальним у різних оточеннях, оскільки потрібно реалізувати лише віртуальну машину JVM для кожної платформи. Якщо на системі встановлено виконавчий пакет, ви можете виконувати будь-яку Java-програму. Важливо зауважити, що всі віртуальні машини JVM на різних платформах та процесорах можуть правильно інтерпретувати однаковий байт-код, незважаючи на деякі різниці в реалізації. Якщо б програма на Java компілювалася в машинозалежний код, то для кожного типу процесорів потрібно було б створити окремі версії програми, що є неефективним. Отже, використання віртуальної машини JVM для виконання байт-коду є простим способом створення кросплатформових програм.

Те, що програма, написана на мові програмування Java, функціонує в умовах віртуальної машини JVM, сприяє збільшенню рівня безпеки. Управління виконанням програми відбувається в межах віртуальної машини JVM, що дозволяє ізолювати програму та мінімізує можливі побічні ефекти виконання, запобігаючи їх виникненню за межами даної системи. Додатково, наявні обмеження в мові програмування Java сприяють підвищенню рівня безпеки системи[17].

Узагальнюючи, коли програма переходить в проміжну форму компіляції, а потім інтерпретується віртуальною машиною JVM, її виконання може бути менш ефективним, ніж у випадку, коли вона скомпільована в виконуваний код. Проте в Java ця різниця в продуктивності майже не відчутна. Байт-код відзначається значними оптимізаціями, тому використання його дозволяє віртуальній машині JVM виконувати програми значно швидше, ніж можна було б очікувати, враховуючи вищезазначені особливості виконання програм на мові програмування Java.

Java була розроблена з початковою ідеєю про інтерпретованість, але це не заважає їй ефективно виконувати компіляцію байт-кода в машинозалежний код для поліпшення продуктивності. Таким чином, технологія HotSpot, яка містить динамічний компілятор (або JIT-компілятор) байт-коду, була впроваджена пізніше. У випадку, якщо динамічний компілятор є частиною віртуальної машини JVM, він компілює фрагменти байт-коду в виконуваний код частинами в реальному часі за потребою. Важливо відзначити, що компіляція всієї програми Java в виконуваний код одночасно є неефективною, оскільки Java проводить різноманітні перевірки, які можуть бути виконані тільки під час виконання. Замість цього динамічний компілятор компілює код під час виконання у випадку потреби. Тільки ті фрагменти байт-коду, яким компіляція призводить до переваг, компілюються, а решта коду програми інтерпретується. Незважаючи на динамічну компіляцію байт-коду, зберігаються характеристики кросплатформенності та безпеки, оскільки віртуальна машина JVM залишається відповідальною за цілісність виконуваного середовища. Схема компіляції програмного коду в Java зображено на рисунку 2.7.

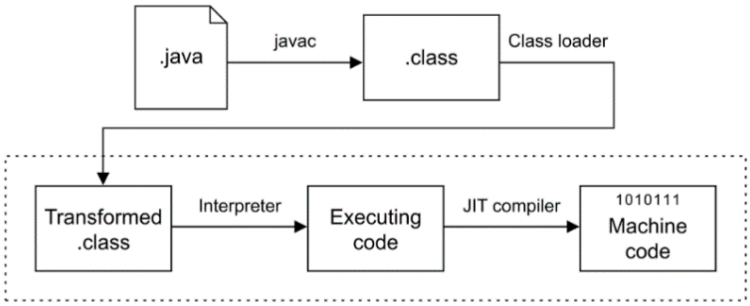


Рис. 2.7 — Компілювання програмного коду в Java

Створення мови програмування Java визначалося переважно необхідністю забезпечення кросплатформовості та безпеки програм. Проте формуванню остаточної версії мови сприяли й інші фактори, такі як простота, кросплатформність, безпека, об’єктно-орієнтованість, надійність, багатопоточність, архітектурна нейтральність, інтерпретація, висока продуктивність, розподільність та динамічність.

Мову програмування Java спроєктовано як просту для вивчення та ефективну для професійних розробників. Для тих, хто вже має досвід у програмуванні, освоїти Java буде неважко, особливо якщо ви знайомі із принципами об’єктно-орієнтованого програмування. Перехід на Java з мови C++ для тих, хто має досвід у розробці програмного забезпечення на C++, вимагає мінімум зусиль. Java успадковує синтаксис та більшість об’єктно-орієнтованих властивостей C++, що полегшує вивчення для багатьох розробників.

Незважаючи на те, що на архітектуру та синтаксис мови Java вплинули попередники, проєктуючи її, не враховувались вимоги сумісності з вихідним кодом інших мов. Це дало команді розробників можливість написати мову Java фактично з нуля. Як результат, вона володіє чітким, практичним та прагматичним підходом до об’єктів. Java вдалося знайти золоту середину між строгим дотриманням принципу, що всі елементи програми є об’єктами, та більш прагматичним підходом. Об’єктна модель в Java є простою та легко розширюється. Навіть цілочисельні типи даних, зберігаються у вигляді високопродуктивних компонентів та не є об’єктами.

Багатоплатформове веб-середовище висуває високі вимоги до програм, оскільки вони повинні надійно працювати в різних системах. Таким чином, розробка надійних програм була важливим пріоритетом при створенні мови програмування Java. З метою забезпечення надійності в Java встановлено ряд обмежень у кількох критичних областях, що дозволяє розробникам виявляти помилки на ранніх етапах розробки. Java також звільняє від потреби турбуватися про багато загальних помилок програмування, оскільки вона є строго типізованою мовою, а отже, перевірка коду відбувається під час компіляції і виконання.

Мова Java була створена відповідно до потреб розробки інтерактивних мережевих програм. Вона підтримує написання багатопотокових програм, які можуть виконувати декілька завдань одночасно. Виконавча система Java включає складні рішення для синхронізації багатьох процесів, що дозволяє створювати стійкі та ефективно працюючі інтерактивні системи. Простий підхід до

організації багатопотокових систем, реалізований в Java, дозволяє розробникам зосереджувати увагу на конкретній поведінці програми, а не на створенні багатопотокової підсистеми. Приклад багатопоточності в Java наведено в рисунку 2.8.

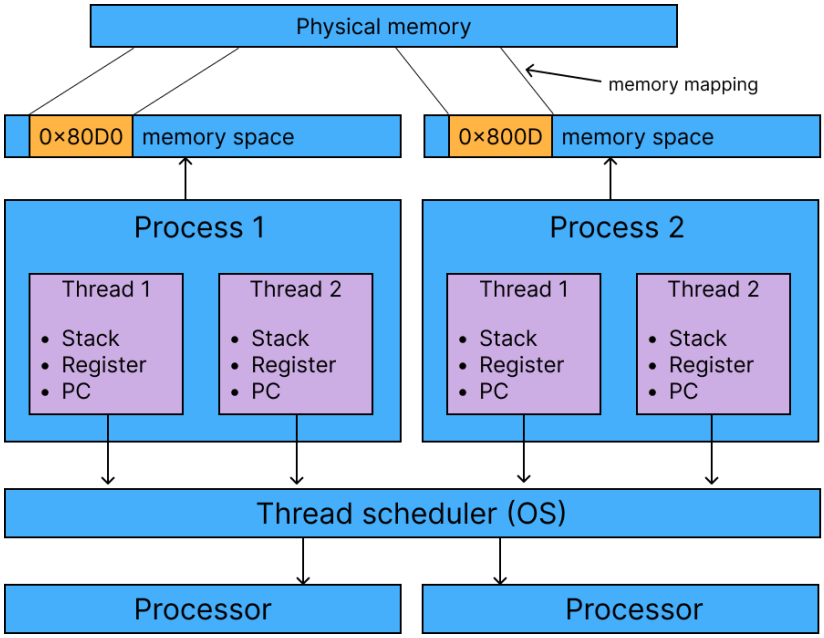


Рис. 2.8 — Приклад багатопоточності в Java

Основною метою, яку розробники Java поставили перед собою, було забезпечення довговічності та кросплатформовості коду. Однією з головних проблем, яку вони стикалися під час розробки Java, була відсутність гарантій того, що код, написаний сьогодні, буде успішно виконуватися завтра, навіть на тому ж самому комп'ютері. Постійні зміни в процесорах та операційних системах можуть призвести до непрацездатності програмного продукту. Щоб змінити цю ситуацію, розробники прийняли жорсткі рішення в самій мові Java та в віртуальній машині JVM. Вони визначили завдання, щоб код, написаний одного разу, виконувався скрізь, в будь-який час і завжди, і в значній мірі це було досягнуто.

Як вже було зазначено, шляхом компіляції програм в проміжне представлення, відоме як байт-код, Java дозволяє створювати міжплатформні програми. Цей код може бути виконаний на будь-якій системі, де реалізована віртуальна машина JVM. Навіть при спробах розробити міжплатформні рішення

вдалося досягти головної мети, хоча і з невеликим зниженням продуктивності системи. Байт-код Java був створений так, щоб його можна було ефективно трансформувати в машинозалежний код для конкретної платформи за допомогою динамічного компілятора. Виконавчі системи Java, які це забезпечують, зберігають всі переваги коду, що не залежить від конкретної платформи.

Мова програмування Java розроблена для використання у розподіленому середовищі Інтернету, оскільки вона підтримує сімейство мережевих протоколів TCP/IP. Звернення до ресурсу за допомогою уніфікованого вказівника інформаційного ресурсу (URL) в мові Java мало відрізняється від звернення до файлу. Також в Java реалізовано віддалений виклик методів програми (RMI — remote method invocation), що дозволяє викликати методи програм через мережу.

Програми на Java включають значний обсяг даних динамічного типу, які використовуються для перевірки повноважень і надання доступу до об'єктів під час виконання програми. Це робить можливим безпечне та ефективне використання динамічної зв'язку коду. Цей аспект особливо важливий для стійкості середовища Java.

2.3.3 Інтегроване середовище розробки Android Studio

На сьогоднішній день, при кожній розробці програмного забезпечення зазвичай використовується інтегроване середовище розробки (IDE). Такі середовища мають власні автоматизовані процеси збірки та запуску програм, компіляції та підкреслення синтаксису та багато інших аспектів, котрі допомагають спростити завдання для розробників і дозволяє новачкам розпочинати творчий
процес без значних труднощів.

Дане інтегроване середовище розробки, як правило, володіє кількома ключовими особливостями, що полегшують роботу з програмним кодом. Нумерація рядків та підсвічування синтаксису істотно сприяють орієнтації у коді. Практично кожне IDE обладнане вбудованим засобом відлагодження додатків. Також важливою є інтеграція з системою контролю версій, що корисна в

командній розробці, де над проєктом може працювати ціла група розробників. Такі системи дозволяють віддалено працювати з одним і тим самим кодом, уникати конфліктів та зберігати єдність правок.

У якості основного інструмента розробки обрано "Android Studio" — це інтегроване середовище розробки, розроблене компанією "Java". Воно забезпечує розробників інструментами для створення додатків і може бути встановлене на різних операційних системах. "Android Studio" є безкоштовним для використання та містить вже готові макети для створення інтерфейсу, що часто становить початковий етап роботи над додатком. Це середовище надає інструменти для розробки додатків для різних пристроїв, таких як смартфони, планшети, годинники та електроніка в автомобілі, приклад наведено в рисунку 2.9.

Android Studio — це офіційне інтегроване середовище розробки, спроєктоване для творення та розробки Android-додатків. Засноване на Intellij IDEA, інтегрованому середовищі розробки для Java, воно включає в себе інструменти для редагування коду та розробки додатків. Перша презентація Android Studio відбулася на конференції Google I/O в травні 2013 року, і перша стабільна версія була випущена в грудні 2014 року. Це середовище розробки доступне для операційних систем, таких як Windows, Mac та Linux, і стало основним інструментом для створення додатків на платформі Android, витіснюючи Eclipse.

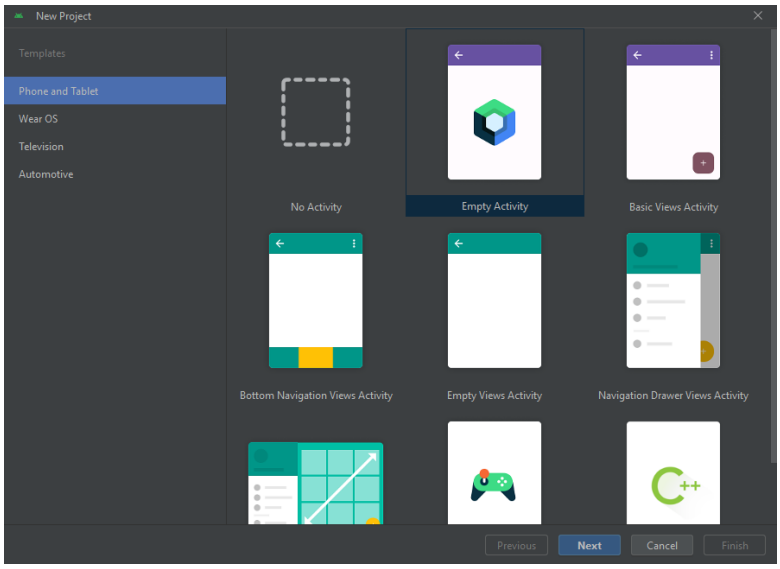


Рис. 2.9 — Вибір форм фактору нового проєкту в Android Studio

Перед початком роботи в Android Studio необхідно завантажити та встановити безкоштовний пакет розробника JDK (Java Development Kit). Після успішної установки цього пакету завантажте Android Studio з офіційного веб-сайту. Після завантаження запускаємо процес встановлення. Під час виконання майстра налаштування Android Studio, необхідно визначити параметри установки відповідно до вимог користувача, вказати шлях для встановлення середовища Android Studio. Після успішної інсталяції інтегрованого середовища розробки можна розпочати створення Android-додатків. Створення нового проєкту наведено в рисунку 2.10

Інтегроване середовище розробки Android Studio використовує віконний інтерфейс, спроектований для оптимізації використання екранного простору та уникнення перевантаження розробника. На даному етапі відображається обмежена кількість вікон, щоб забезпечити зручний та функціональний робочий процес. Крім того, Android Studio використовує контекстні вікна, які виводяться лише при контекстному виклику з конкретних інструментів, та надає можливість розробнику управляти активністю прихованих вікон через зручне меню налаштувань. Це сприяє збереженню зосередженості розробника, поліпшує ефективність роботи. До того ж, Android Studio дозволяє розробникам активувати або деактивувати приховані вікна.

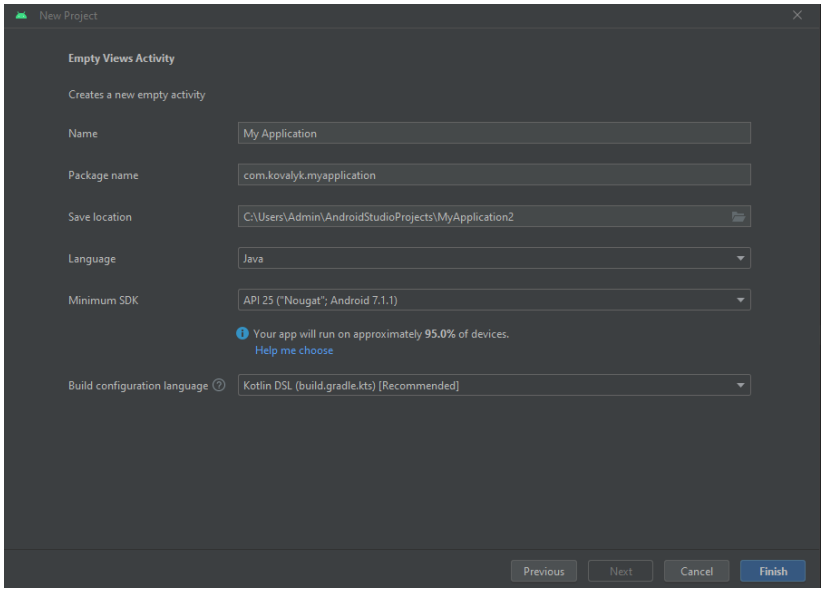


Рис. 2.10 — Створення нового проєкту в Android Studio

Однією з ключових функцій будь-якого інтегрованого середовища розробки є навігація. Проєкти Android часто включають в себе багато тек, каталогів та файлів, які організовані відповідно до структури проєкту наведено в рисунку 2.11.

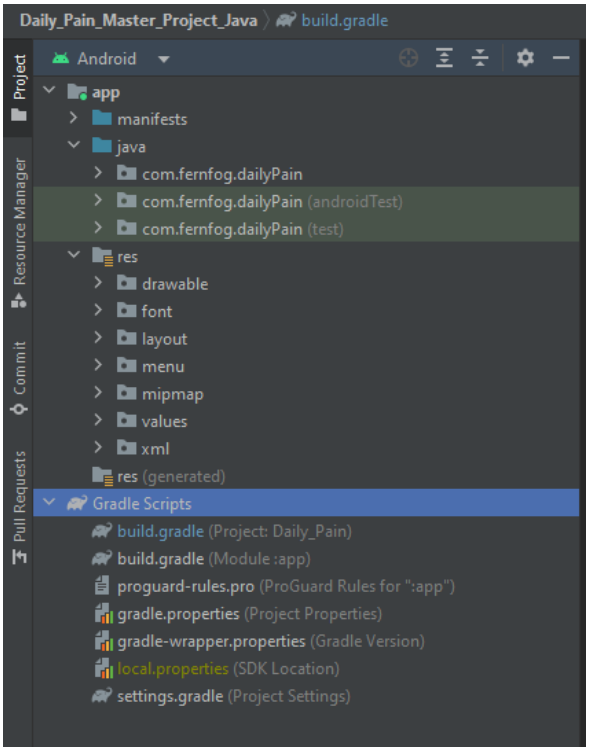


Рис. 2.11 — Проєктна структура в Android Studio

Ця файлова структура загалом називається проєктом і представляє собою організаційну одиницю, що визначає повне програмне рішення. Проєкти додатків, зазвичай, містять файли вихідного коду Java або Kotlin, файли конфігурації XML, зображення, стилі та інші ресурси, які систематизовані в організаційній структурі. Android Studio допомагає розробникам структуризувати проєкт під час створення програми. Під час збірки проєкту виникають файли конфігурації, а каталоги структуризуються відповідно до ієрархії.

Під час створення додатків у розробковому середовищі Android Studio доведеться провести компіляцію та запуск додатка кілька разів. Створену програму для Android можна перевірити, встановити та запустити на реальному або віртуальному пристрої Android (AVD — Android Virtual Device), відомому як емулятор, приклад наведено в рисунку 2.12. Перед використанням віртуального пристрою його потрібно завантажити та налаштувати. Практично всі можливості

реального пристрою Android надає емулятор. На ньому можна симулювати текстові повідомлення, вхідні дзвінки, імітувати обертання екрану, визначати місцезнаходження пристрою, отримувати доступ до сервісів Google Play тощо [14].

Рис. 2.12 — Емулятор в Android Studio

Тестування додатка в емуляторі в певному відношенні є зручнішим та швидшим, ніж на реальному пристрої. Оскільки, передача даних в емулятор може бути швидшою, ніж на підключеному фізичному пристрої.

2.3.4 Автоматизована система збірки Gradle

Під час розробки програмного засобу у відокремленому середовищі розробки Android Studio відсутність автоматизації призводить до повторюваних та нудних завдань, що великою мірою збільшують ризик помилок. На кожному етапі розробки програмного засобу, від компіляції вихідного коду до завершальної збірки для релізу та перевірки на виробництві, потребує ручної участі. Автоматизація проєкту спрощує цей процес, зменшує ручне втручання та мінімізує ризик виникнення проблем у програмному забезпеченні.

Автоматизація проєкту приносить кілька очевидних переваг у процесі розробки програмного продукту. По-перше, вона усуває необхідність ручного

втручання в процес збірки. Ручне виконання операцій для створення та постачання програмного продукту вимагає великої кількості часу і підвищує ризик виникнення помилок. Розробники та системні адміністратори можуть зосередитися на інших завданнях, які вимагають їхньої участі, замість витрачання часу на ручну компіляцію. Любий етап в розробці, який може бути автоматизований, повинен бути автоматизований.

Додатковою перевагою є можливість створення повторюваних збірок. Типова конструкція програмного проєкту включає передбачувані та впорядковані кроки. Наприклад, розробник може компілювати вихідний код, запускати тести та виконувати збірку з отриманням очікуваного результату. Ці кроки повторюються щодня. Автоматизація робить цей процес простим та однотипним, як натискання кнопки. Результати цього процесу повинні бути однаковими для всіх, хто використовує збірку проєкту.

Ще однією перевагою є можливість створення переносних збірок. Можливість
використовувати інтегроване середовище розробки обмежена для розробників. Зазвичай розробник повинен встановлювати певний продукт на своєму пристрої. Крім того, інтегроване середовище розробки може бути доступним лише для певної операційної системи. Автоматизована збірка не має вимог до певного середовища виконання, що дозволяє запускати її з будь-якого пристрою та в будь-який час. Зручно, якщо автоматизовані завдання виконуються з командного рядка, що дозволяє запускати процес збірки без обмежень, які стосуються операційної системи чи середовища розробки[16].

Автоматизація проєкту поділяється на різні типи, зокрема збірка на вимогу. В цьому випадку користувач ініціює процес побудови на конкретному пристрої. Переважно система контролю версій (VCS) відповідає за моніторинг версій збірок та файлів вихідного коду. Здебільшого, користувач запускає скрипт у командному рядку, який автоматизовано виконує певні завдання у заздалегідь визначеному порядку, зазвичай це копіювання файлів, формування результатів

або компіляція коду. Така збірка досить часто проводиться щоденно декілька разів.

Ще одним видом автоматизації є автоматично запущені збірки. У випадку, коли розробник займається гнучкою розробкою програмного забезпечення, важливо для нього отримати швидкий зворотний зв'язок щодо стану проєкту. Такий тип автоматизації активується під час перевірки коду системою контролю версій. Розробник отримує важливу інформацію про те, чи може вихідний код бути зібраним без помилок, та про можливі програмні дефекти, виявлені під час блочної помилки або тестування інтеграції.

Третім типом автоматизації є заплановані збірки, які ґрунтуються на плануванні завдань у визначений час. Вони виконуються в певний інтервал часу або в конкретний час і часто здійснюються на віддаленому сервері. Цей спосіб автоматизації особливо ефективний при генерації звітів чи документації проєкту. Запуск запланованих збірок, який виконується за необхідності, зазвичай розглядається як процес безперервної інтеграції (CI).

Протягом значного періоду часу процеси компіляції та упакування програмного забезпечення здійснювались за допомогою простих засобів. Однак з настанням нового етапу у розробці програмного забезпечення виникла необхідність в автоматизації процесу побудови проєктів. Сучасні проєкти включають в себе різноманітні та обширні програмні стеки, використовують множину мов програмування та впроваджують різноманітні стратегії тестування. З підвищенням гнучкості в розробці необхідно, щоб процес компіляції підтримував швидку інтеграцію коду і забезпечував ефективну поставку продукту в тестові та виробничі середовища. Зазвичай встановлені інструменти компіляції не відповідають цим вимогам в їхній простій формі. Розробники програмного забезпечення витрачають значний час на освоєння XML-коду компіляції, і вони не можуть визначати власну логіку побудови при додаванні сценарію компіляції, який неможливо змінити.

Однак існує спрощений та виразний метод для виконання цих завдань — застосунок Gradle. Система автоматизованої збірки Gradle була вперше випущена

у 2007 році і є сучасним етапом еволюції інструментів збірки для JVM. Gradle ґрунтується на досвіді попередніх інструментів автоматизованої побудови, таких як Ant та Maven, піднімаючи їхні найкращі концепції на новий рівень. Дотримуючись концепції щодо побудови, Gradle дозволяє декларативно моделювати вашу предметну область за допомогою потужної та виразної доменної мови DSL, яка реалізована на Groovy замість XML. Gradle дає можливість писати власну логіку мовами Groovy або Java. Java має значну кількість бібліотек і фреймворків, і система управління залежностями використовується для автоматичного завантаження артефактів зі сховища та їхнього внесення в код програми [18]. Набір функцій системи Gradle наведено на рисунку 2.13.

Щоб вирішити недоліки існуючих рішень в управлінні залежностями, Gradle пропонує свою власну реалізацію. Ця реалізація не лише добре налаштована, а й активно працює на максимальну сумісність з існуючими інфраструктурами управління залежностями, такими як Maven та Ivy, а також активно підтримує визначення та організацію багатьох проєктних збірок, а також моделювання залежностей між різними проєктами.

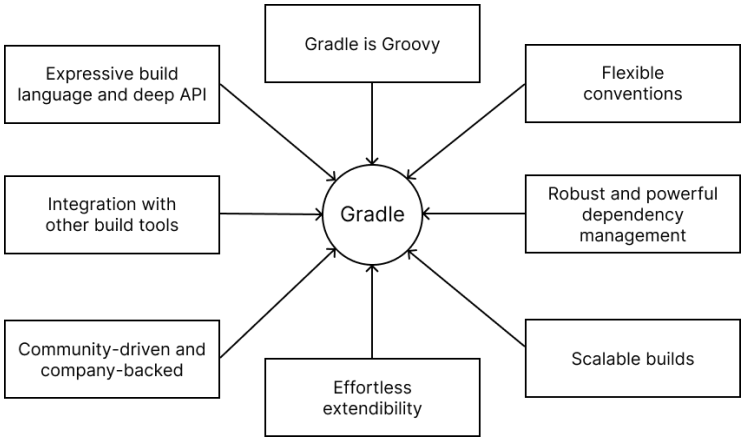


Рис. 2.13 — Набір функцій системи Gradle

Включення автоматизації проєкту в повсякденну роботу є невід'ємною частиною для розробника. Сценарії збірки Gradle використовують декларативний підхід, вони легко читаються і чітко виражають свої наміри. Використання коду на Groovy замість XML, і врахування елементів філософії Build-by-Convention,

дозволяє значно скоротити обсяг сценарію збірки і робить його набагато більш зрозумілим. Для порівняння приклад обсягу коду для Gradle (див. Лістинг 1), а для Maven (див Лістинг 2).

Лістинг 1 — Приклад коду для Gradle

```
apply plugin:'java'
apply plugin:'checkstyle'
apply plugin:'findbugs'
apply plugin:'pmd'
version ='1.0'
repositories {
mavenCentral() }
dependencies {
testCompile group:'junit', name:'junit', version:'4.11'
}
```

Лістинг 2 — Приклад коду для Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.programming.mitra</groupId>
<artifactId>java-build-tools</artifactId>
<packaging>jar</packaging>
<version>1.0</version>
 <dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>2.3.2</version>
</plugin>
</plugins>
</build>
</project>
```

Кожна побудова в Gradle розпочинається зі сценарію. Стандартне ім'я для скрипту Gradle — build.gradle. При виклику основної команди в командному рядку система збірки автоматично шукає файл з таким самим ім'ям. Якщо файл не знайдено, виводиться повідомлення про помилку. Після створення цього файлу необхідно визначити одне чи кілька завдань [19].

Отже, Gradle — це автоматизована система збірки, яка базується на декларативному та виразному Groovy DSL. Поєднує гнучкість та простоту розширення конфігурації та підтримкою традиційного управління залежностями, що робить його першочерговим вибором для збірки проєктів з відкритим кодом.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ СТАНУ ЗДОРОВ'Я

3.1 Опис реалізації програмного засобу

Програмний засіб для моніторингу та аналізу стану здоров'я буде складатися з декількох активностей та фрагментів. Основна активність матиме 4 фрагменти з основним функціоналом програмного засобу. Розроблений додаток включатиме допоміжні вікна для взаємодії користувача та програмного засобу. Дана структура буде легкою для сприйняття користувачем завдяки чітко організованим та зрозумілим модулям.

На рисунку 3.1 наведена структурна схема, де розміщуються програмні модулі.

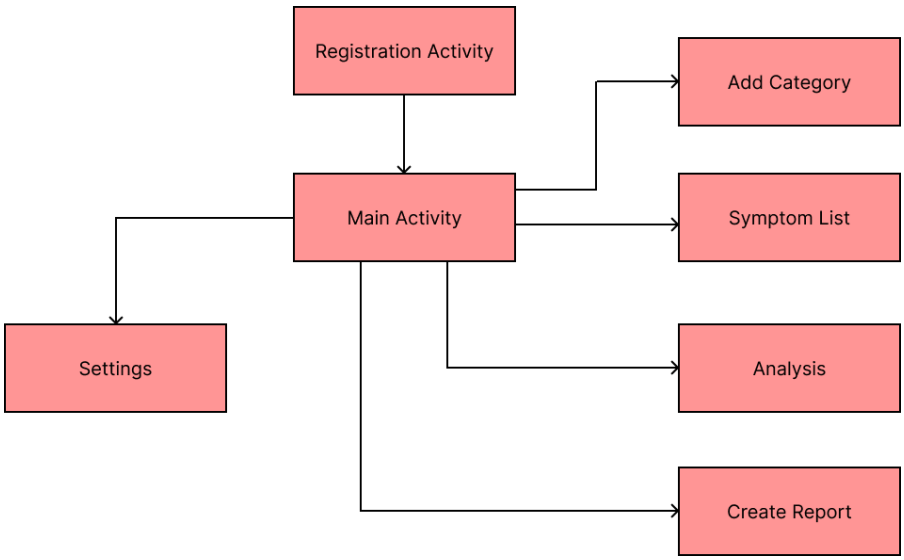


Рис. 3.1 — Структурна схема програмного засобу

3.2 Розробка інтерфейсу програмного засобу

Графічний інтерфейс користувача повинен відповідати наступним критеріям: чіткість, відповідність, виразність та послідовність. Щоб відповісти цим вимогам, був розроблений інтерфейс програмного продукту, який включає кілька модулів. Коли користувач відкриває додаток запускається процес перевірки, чи присутній в базі даних обліковий запис, якщо ні відкривається вікно реєстрації, якщо присутній відкривається сторінка створення категорії. Де

користувач може створювати та видаляти категорії, натиснувши на категорію відкриється модальне вікно, за допомогою якого можна створити запис показників для певної категорії. Після чого за допомогою нижнього навігаційного меню користувач може перейти на сторінку переліку створених ним записів, де може видаляти помилкові або не потрібні записи. Вікно аналізу надає можливість проаналізувати введені користувачем дані за допомогою лінійного графіку, та згрупованим показникам за рівнем болю. Вікно створення звіту дозволяє для обраної категорії сформувати документ в форматі PDF для подальшої консультації у лікаря. Вікно налаштувань дозволяє користувачу видалити обліковий запис. Розроблений дизайн інтерфейсу за допомогою векторного онлайн-сервісу розробки інтерфейсів Figma, зображено на рисунку 3.2.

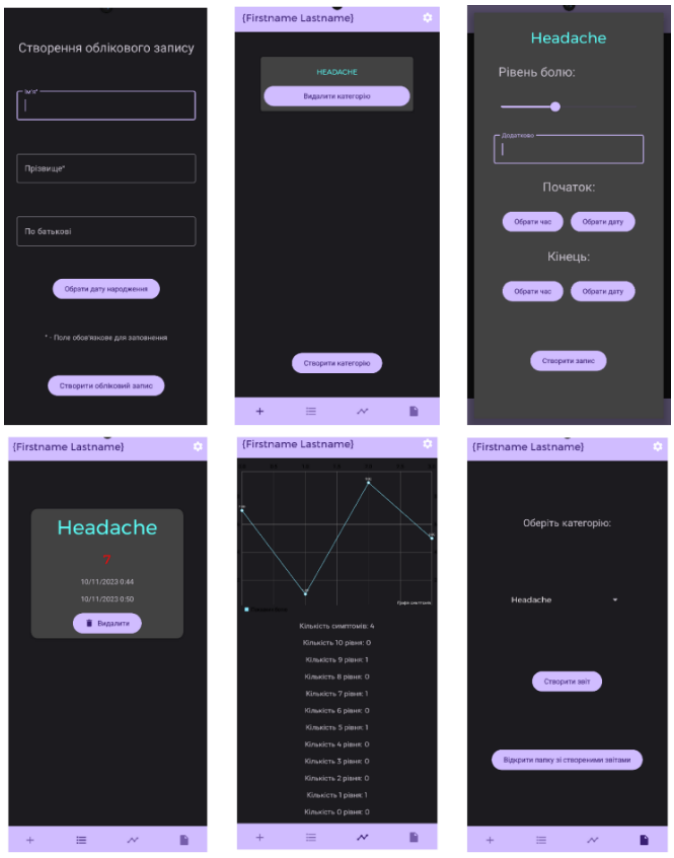


Рис. 3.2 — Розроблений дизайн інтерфейсів за допомогою Figma

3.2.1 Сторінка реєстрації

Дане вікно слугує для створення локального облікового запису для роботи з програмним засобом. Воно складається з різних компонентів:

— 3-х полів для вводу інформації;

— кнопка для вибору дати народження;

— кнопка для створення облікового запису.

3.2.2 Сторінка створення категорії

Дане вікно призначене для створення категорій, а також для створення записів певної категорії. Воно складається з різних компонентів:

— кнопка для створення нової категорії;

— компонент для відображення переліку категорій;

— модальне вікно для створення запису.

3.2.3 Сторінка переліку записів

Дане вікно слугує для моніторингу створених записів, а також видалення помилкових чи непотрібних записів. Воно складається з різних компонентів:

— компонент для відображення переліку записів;

— кнопка для видалення запису.

3.2.4 Сторінка аналізу

Дане вікно слугує для проведення аналізу показників стану здоров’я, визначені їх тенденцій за рівнем болю, а також загальної кількості симптомів та групування кількості симптомів за рівнем болю. Воно складається з різних компонентів:

— лінійний графік;

— перелік кількості симптомів за рівнем болю;

3.2.5 Сторінка створення звіту

Дане вікно призначене для створення звіту спостереження на основі введених користувачем даних. Воно складається з різних компонентів:

— випадаючого списку категорій;

— кнопки формування звіту;

— кнопки відкриття директорії місцезнаходження сформованих звітів;

3.2.6 Сторінка налаштувань

Дане вікно призначене для видалення створених користувачем облікового запису, створених категорій та створених записів. Воно складається з кнопки для видалення.

3.3 Розробка функціональності програмного засобу

Створено проєкт. За замовчуванням маємо 2 файли MainActivity.java та activity_main.xml. Тому створимо нову активність, де буде створюватися обліковий запис. У файлі activity_registration.xml, який представляє розмітку активності, визначимо наступний код (див. Лістинг 3).

Лістинг 3 — Розмітка файлу activity_registration.xml

```
<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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout"
    android:layout_width="358dp"
    android:layout_height="wrap_content"
    android:hint="@string/name">
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/user_first_name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout2"
    android:layout_width="358dp"
    android:layout_height="wrap_content"
    android:hint="@string/last_name">
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/user_last_name"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>
<Button
    android:id="@+id/choiseOfDate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Обрати дату народження"/>
<Button
    android:id="@+id/account_reg"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/create_account_button"
    android:textSize="15sp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Далі напишемо функціонал для нашої активності, щоб при натисканні на кнопку реєстрації, для цього в методі onCreate потрібно встановити setOnClickListener на нашу кнопку (див. Лістинг 4). Однак перед цим потрібно створити клас DBHandler.java в якому будуть оголошені функції, та методи для взаємодії з базами даних, лістинг коду DBHandler.java див. Додаток В.

Лістинг 4 — Встановлення setOnClickListener в методі onCreate

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_registration);
  button = findViewById(R.id.choiseOfDate);
  submitButton = findViewById(R.id.account_reg);
  firstName = findViewById(R.id.textInputLayout);
  lastName = findViewById(R.id.textInputLayout2);
  patronymic = findViewById(R.id.textInputLayout3);
  submitButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    String firstNameT = firstName.getEditText().getText().toString().trim();
    String lastNameT = lastName.getEditText().getText().toString().trim();
    String patronymicT = patronymic.getEditText().getText().toString().trim();
  if (!firstNameT.isEmpty() && !lastNameT.isEmpty() &&
!patronymicT.isEmpty() && birth != null) {
    DBHandler dbHandler = new DBHandler(RegistrationActivity.this);
      dbHandler.addNewUser(firstNameT, lastNameT, patronymicT, birth);
      Toast.makeText(RegistrationActivity.this, "Обліковий запис створено",
Toast.LENGTH_LONG).show();
       Intent intent = new Intent(RegistrationActivity.this, MainActivity.class);
```

```
                startActivity(intent);
                    finish();
            } else {
                Toast.makeText(RegistrationActivity.this, "Не заповнено обов'язкове
поле", Toast.LENGTH_LONG).show();
            }
            }
           });
button.setOnClickListener(new View.OnClickListener() {
            @Override
                public void onClick(View v) {openDatePicker();}
            });
    }
```

Тепер створимо нову активність StartupActivity.java, яка буде запускати процес перевірки про те чи вже створено обліковий засіб чи ще ні, і в залежності від отриманої відповіді, відкриватиме певну активність (див. Лістинг 5).

Лістинг 5 — Перевірка на наявність облікового запису

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_startup);
      DBHandler dbHandler = new DBHandler(this);
      if (dbHandler.checkDatabaseExists(this) && dbHandler.hasUsers()) {
         Intent intent = new Intent(this, MainActivity.class);
         startActivity(intent);
      } else {
         Intent intent = new Intent(this, RegistrationActivity.class);
         startActivity(intent);
      }
   }
```

Тепер оскільки StartupActivity повинна запускатися першою нам потрібно повідомити про це AndroidManifest.xml (див. Лістинг 6).

Лістинг 6 — Заміна активності в AndroidManifest.xml

```
<activity
   android:name=".StartupActivity"
   android:exported="true">
      <intent-filter>
         <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

Створимо вікно переліку категорій fragment_add_symptom.xml додамо до нього розмітку та заповнимо файл AddSymptomFragment.java функціоналом в методі onCreateView (див. Лістинг 7).

Лістинг 7 — Вміст метода onCreateView

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_add_symptom, container,
false);
        Button button = view.findViewById(R.id.openDialog);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            AddSymptomCategoryDialog addSymptomCategoryDialog = new
AddSymptomCategoryDialog(getContext());
                addSymptomCategoryDialog.show();
            }
        });
        DBHandler dbHandler = new DBHandler(getContext());
        List<SymptomCategory> allCategories =
dbHandler.getAllCategoriesWithColors();
        scrollView = view.findViewById(R.id.symptomCategories);
        for (SymptomCategory category : allCategories) {
            CardView cardView = new CardView(view.getContext());
            LinearLayout.LayoutParams cardViewParams = new
LinearLayout.LayoutParams(
                    800,
                    ViewGroup.LayoutParams.WRAP_CONTENT);
            cardViewParams.setMargins(0, 16, 0, 16);
            cardView.setLayoutParams(cardViewParams);
            cardView.setRadius(8);
            cardView.setElevation(8);
            LinearLayout cardContentLayout = new
LinearLayout(view.getContext());
            cardContentLayout.setOrientation(LinearLayout.VERTICAL);
            cardContentLayout.setLayoutParams(new ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            cardContentLayout.setOrientation(LinearLayout.VERTICAL);
            cardContentLayout.setGravity(Gravity.CENTER);
```

```
                cardContentLayout.setPadding(16, 16, 16, 16);
                Button buttonOpenAdd = new Button(view.getContext());
                buttonOpenAdd.setBackgroundColor(Color.parseColor("#00FFFFFF"));
                buttonOpenAdd.setText(category.getName());
                buttonOpenAdd.setTextColor(Color.parseColor("#" +
category.getColor()));
                Typeface customFont = ResourcesCompat.getFont(getContext(),
R.font.montserrat);
                buttonOpenAdd.setTypeface(customFont);
                buttonOpenAdd.setGravity(Gravity.CENTER);
              }
            return view;
        }
```

Створимо сторінку для проведення аналізу введених користувачем показників, для візуалізації та визначені тенденції показників за рівнем болю скористаємось LineChart (див. Лістинг 8).

Лістинг 8 — Налаштування LineChart в файлі AnalisFragment.java

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_analis, container, false);
    LineChart lineChart = view.findViewById(R.id.lineChart);
    DBHandler dbHandler = new DBHandler(getContext());
    List<Symptome> symptomes = dbHandler.getAllSymptomes();
    LineDataSet dataSet = new LineDataSet(entries, "Показник болю");
    dataSet.setValueTextColor(Color.WHITE);
    LineData lineData = new LineData(dataSet);
    lineChart.setData(lineData);
    Description description = new Description();
    description.setTextColor(Color.WHITE);
    description.setText("Графік симптомів");
    lineChart.setDescription(description);
    lineChart.invalidate();
    return view;
  }
```

Створимо сторінку для формування звіту fragment_report.xml та додамо до нього розмітку сторінки (див. Лістинг 9), також в файл ReportFragment.java додамо функціонал для створення звіту див. Додаток Г

Лістинг 9 — Розмітка сторінки fragment_report.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ReportFragment">
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="240dp"
        android:layout_height="60dp"/>
    <Button
        android:id="@+id/testBSLKFJM"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/createReportButton"/>
    <Button
        android:id="@+id/openDirectoryButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/openDirectoryButton"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Повний код всього програмного засобу буде доступний за посиланням на репозиторій, що знаходиться на сайті GitHub.com [20].

4 ТЕСТУВАННЯ ТА РОБОТА З ПРОГРАМНИМ ЗАСОБОМ

4.1 Методика проведення тестування

Метою проведення тестування мобільного програмного засобу Daily Pain є перевірка правильної функціональності всіх його можливостей на різних версіях операційної системи Android за звичайними сценаріями використання. Приблизно 10% часу буде відведено на тестування незвичайних або потенційно викликаних помилок у сценаріях використання.

Підсумковими матеріалами тестування будуть:

— висновок тестування, щодо загального стану програмного засобу, наданий розробникам та менеджерам продукту для отримання об'єктивної карти ефективності;

— звіт із результатів тестування поточного покриття, включаючи типові сценарії використання;

— задокументовані дефекти, баги.

Тестування буде виконано вручну з позиції кінцевого користувача мобільного застосунку.

Умовами для тестування є те що програмний засіб повинен відповідати вимогам кінцевого користувача та забезпечувати необхідні функціональності, пов’язаних з моніторингом та аналізом стану здоров’я, а також в формуванні звітності в форматі PDF.

Наведений нижче тест-план є формальним, і для створення докладного плану тестування необхідно мати розуміння поточного стану проєкту. Після першого виконання функціональних тестів будуть виконані модифікації та вдосконалення до тест-плану. Цей перший запуск функціональних тестів надасть ясне усвідомлення стійкості системи та визначення комплекту тестів, які проводитимуться в кожній конфігурації. Даний метод дозволить скласти детальний звіт про тестування продукту та акцентувати увагу на можливих проблемних аспектах.

Планується провести чотири етапи процесу тестування з такими завданнями:

— на першому етапі заплановано складання тест-плану та часткове виконання функціонального тестування;

— другий етап буде приділений докладному тестуванню функціональності з виявленням та описом виявлених дефектів;

— третій етап включає перевірку вирішених розробниками помилок та проведення регресійного тестування;

— четвертий етап передбачає тестування дизайну продукту з описом виявлених дефектів.

Такий підхід сприяє максимальній деталізації глибини тестування, що в свою чергу дозволяє точно визначити ресурси, витрачені на кожен етап, та дозволяє розробникам виправляти дефекти на ранніх стадіях проєкту.

Затверджені для тестування ОС:

— Android 7.0 “Nougat”;

— Android 8.1 “Oreo”;

— Android 11.0 “Red Velvet Cake”.

Цілю функціонального тестування є виявлення невідповідностей до технічного завдання, функціональних недоліків і розходжень від очікувань користувача шляхом виконання як стандартних, так і нетривіальних тестових сценаріїв. Нижче наведені описи процесів.

Реєстрація:

— реєстрація облікового запису;

— перевірка на наявність вже створеного облікового запису.

Сторінка категорій:

— створення категорії;

— функція відображення створених категорій;

— видалення категорії;

— створення записів для певної категорії.

Сторінка записів:

— функція відображення переліку створених записів;

— видалення записів.

Сторінка аналізу:

— функція відображення показників на лінійному графіку;

— функція відображення згрупованих показників за рівнем болю.

Сторінка формування звіту:

— вибір певної категорії з випадаючого списку;

— створення звіту;

— відкриття директорії місцезнаходження вже сформованих звітів.

Сторінка налаштувань має функціонал видалення облікового запису та закриття програми.

Ціль регресійного тестування полягає в перевірці змін, внесених у програмний продукт, з метою забезпечення відсутності помилок у вже протестованих частинах програмного забезпечення, які можуть виникнути внаслідок нової версії операційної системи. Під час проведення регресійного тестування будуть використовуватися наступні види тестів:

— верифікаційні тести;

— тестування на інших версіях ОС;

— тести суміжного функціоналу.

Ціль тестування дизайну інтерфейсу є перевірка відповідності дизайну програмного засобу, до затвердженого дизайну програмного засобу. Підчас здійснення тестування буде перевірено об’єкти:

— реєстраційна форма;

— сторінка категорій;

— вікно створення категорії;

— вікно створення запису;

— сторінку переліку записів;

— сторінку аналізу;

— сторінку створення звітності.

План робіт для проведення тестування програмного засобу Daily Pain зображено в табл. 4.1.

Таблиця 4.1 — План проведення робіт з тестування Daily Pain

Завдання	**Робочий обсяг**	**Термін початку**	**Термін завершення**
Розробка тест-плану	4 годин	29.10.2023	30.10.2023
Виконання тестування	6 годин	31.10.2023	02.11.2023
Аналіз тестування	2 години	03.11.2023	03.11.2023
Висновки тестування	3 години	04.11.2023	04.11.2023

Створимо за допомогою Microsoft Excel чек-ліст для програмного засобу Daily Pain на Android 7.0 “Nougat”, проведемо перевірку функціоналу програмного засобу та внесемо дані в табл. 4.2.

Таблиця 4.2 — Чек-ліст програмного засобу Daily Pain на Android 7.0

Daily Pain	Google Pixel 6
	Android 7.0
Встановлення додатку	
Перевірка відсутності аварійного завершення програми	Passed
Перевірка встановлення програми на пристрій	Passed
Функції на пристроях	
Перевірка голосового введення даних (якщо підтримується)	Passed
Перевірка відображення у портретній/ландшафтній орієнтації	Passed
Перевірка коректності формування звіту	Passed
Графічні елементи	
Перевірка швидкості відгуку елемента	Passed
Розмір елемента дає можливість натиснути на нього	Passed
Реакція програмного засобу на зовнішні переривання	
Реагування програмного засобу на блокування/розблокування екрану	Passed
Реагування програмного засобу на заряджання пристрою	Passed
Реагування прогрманого засобу на вимкнення та підключення до мережі Wi-Fi	Passed
Реагування програмного засобу на вхідні та вихідні дзвінки	Passed

Створимо за допомогою Microsoft Excel чек-ліст для програмного засобу Daily

Pain на Android 8.1 “Oreo”, проведемо перевірку функціоналу програмного засобу та внесемо дані в табл. 4.3.

Таблиця 4.3 — Чек-ліст програмного засобу Daily Pain на Android 8.1

Daily Pain	Google Pixel 6
	Android 8.1
Встановлення додатку	
Перевірка відсутності аварійного завершення програми	Passed
Перевірка встановлення програми на пристрій	Passed
Функції на пристроях	
Перевірка голосового введення даних (якщо підтримується)	Passed
Перевірка відображення у портретній/ландшафтній орієнтації	Passed
Перевірка коректності формування звіту	Passed
Графічні елементи	
Перевірка швидкості відгуку елемента	Passed
Розмір елемента дає можливість натиснути на нього	Passed
Реакція програмного засобу на зовнішні переривання	
Реагування програмного засобу на блокування/розблокування екрану	Passed
Реагування програмного засобу на заряджання пристрою	Passed
Реагування прогрманого засобу на вимкнення та підключення до мережі Wi-Fi	Passed
Реагування програмного засобу на вхідні та вихідні дзвінки	Passed

Створимо за допомогою Microsoft Excel чек-ліст для програмного засобу Daily Pain на Android 11.0 “Red Velvet Cake”, проведемо перевірку функціоналу програмного засобу та внесемо дані в табл. 4.4.

Таблиця 4.4 — Чек-ліст програмного засобу Daily Pain на Android 11.0

Daily Pain	Google Pixel 6
	Android 11.0
Встановлення додатку	
Перевірка відсутності аварійного завершення програми	Passed
Перевірка встановлення програми на пристрій	Passed
Функції на пристроях	
Перевірка голосового введення даних (якщо підтримується)	Passed
Перевірка відображення у портретній/ландшафтній орієнтації	Passed
Перевірка коректності формування звіту	Passed
Графічні елементи	
Перевірка швидкості відгуку елемента	Passed
Розмір елемента дає можливість натиснути на нього	Passed
Реакція прогрмного засобу на зовнішні переривання	
Реагування програмного засобу на блокування/розблокування екрану	Passed
Реагування програмного засобу на заряджання пристрою	Passed
Реагування прогрманого засобу на вимкнення та підключення до мережі Wi-Fi	Passed
Реагування програмного засобу на вхідні та вихідні дзвінки	Passed

За результатами тестування на різних версіях, можна заявити, що починаючи від версії Android 7.0 програмний засіб працює коректно та стабільно.

Перевагами програмного засобу Daily Pain на платформі Android є те що він не містить реклами, зручний у користуванні, займає дуже мало пам’яті пристрою, чим є дуже зручним для завантаження, відображає лише ту інформацію яка потрібна користувачеві та може сформувати звіт в форматі PDF.

4.2 Робота з додатком

Програмний засіб Daily Pain розроблений для застосування на мобільних пристроях, таких як смартфони та планшети, що працюють під управлінням операційної системи Android. Користувачу доступно створення локального облікового запису, створення категорії симптомів, створення записів для певної категорії, проводити простий аналіз введених показників та сформувати звіт спостереження в форматі PDF.

4.2.1 Системні вимоги та встановлення

Програма інсталюється за допомогою "apk" файлу на пристрій користувача, який автоматично налаштовується. Так як додаток був розроблений для платформи "Android", необхідно, щоб пристрій користувача працював під цією операційною системою. Для оптимальної роботи додатку вимагається версія операційної системи не нижче 7.0.

4.2.2 Інструкція користувача з використання програмного засобу

При першому відкриті програмного засобу, програма запропонує користувачу створити локальний обліковий запис для роботи в додатку, без нього користувач не
зможе використовувати весь функціонал програми, в подальшому коли вже створено обліковий запис, сторінка реєстрації відкриватися не буде. Сторінка реєстрації зображено на рисунку 4.1.

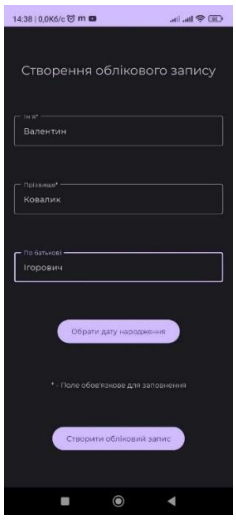


Рис. 4.1 — Сторінка реєстрації Daily Pain

Після створення облікового запису, користувач буде перенаправлений на сторінку категорій, де він може створювати нові категорії, видаляти не потрібні категорії, Щоб створити нову категорію потрібно натиснути на кнопку «Створити категорію», після чого з’явиться вікно для створення категорії, де користувачу потрібно ввести назву нової категорії та обрати колір для нової категорії. Якщо натиснути на вже створену категорію з’явиться вікно створення запису для певної категорії. Після того як з’явиться вікно створення запису користувач може обрати за допомогою повзунка рівень болю для симптома від 0 (дуже слабкий) до 10 (дуже сильний), ввести додаткову інформацію, та встановити час початку та кінця дії симптому. Після чого натиснути кнопку «Створити запис». Сторінка категорії, вікно створення категорії та вікно створення запису зображені на рисунку 4.2.

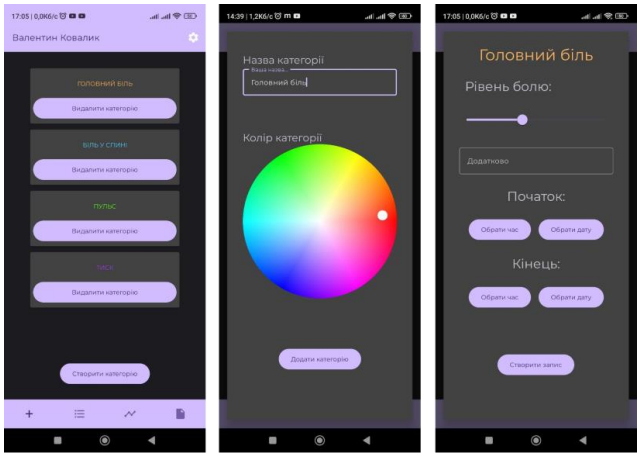


Рис.4.2 — Сторінка категорії, вікна створення категорії та запису

Коли користувач створить свій перший запис він може перейти до сторінки списку записів, за допомогою нижнього навігаційного меню натиснувши 2 кнопку з лівої сторони, та переглянути його. Також користувач може видалити запис якщо він був створено помилково або вже є неактуальним, натиснувши кнопку «Видалити». Сторінка переліку записів зображена на рисунку 4.3.

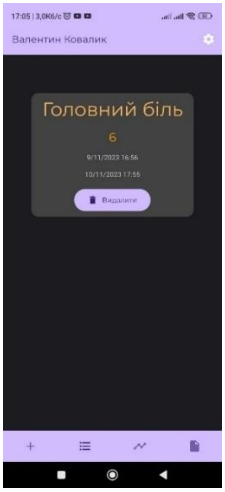


Рис. 4.3 — Сторінка переліку записів

Для того щоб проаналізувати введені показники, користувач може перейти на сторінку аналізу натиснувши на 3 кнопку з ліва на нижньому навігаційному меню. На сторінці аналізу присутній лінійний графік на якому можна проаналізувати тенденцію введених користувачем симптомів за рівнем болю, також згруповано загальну кількість симптомів та окремо за кожним рівнем. Сторінку аналізу зображено на рисунку 4.4.

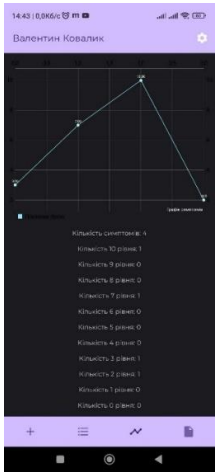


Рис. 4.4 — Сторінка аналізу

Для того щоб сформувати звіт на основі введених користувачем показників потрібно перейти на сторінку створення звіту, для цього потрібно натиснути 4 кнопку з ліва на нижньому навігаційному меню. Де потрібно обрати за допомогою випадаючого списку, потрібну категорію для формування звіту та натиснути кнопку «Створити звіт». Сторінку створення звіту зображено на рисунку 4.5.

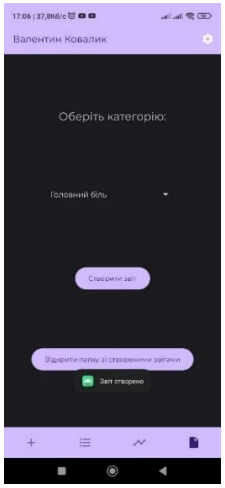
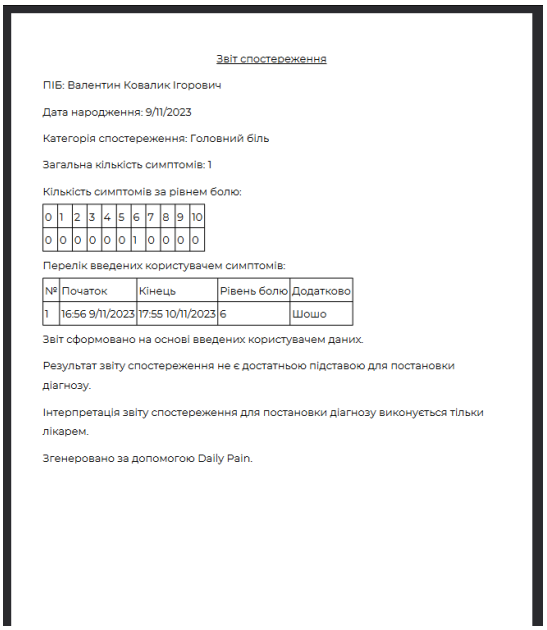


Рис. 4.5 — Сторінка створення звіту

Для того щоб відкрити папку з файлами PDF, потрібно на сторінці створення звіту натиснути кнопку «Відкрити папку зі сформованими звітами». Приклад сформованого програмою звіту зображено на рисунку 4.6.



Звіт спостереження

ПІБ: Валентин Ковалик Ігорович

Дата народження: 9/11/2023

Категорія спостереження: Головний біль

Загальна кількість симптомів: 1

Кількість симптомів за рівнем болю:

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	1	0	0	0	0

Перелік введених користувачем симптомів:

№	Початок	Кінець	Рівень болю	Додатково
1	16:56 9/11/2023	17:55 10/11/2023	6	Шошо

Звіт сформовано на основі введених користувачем даних.

Результат звіту спостереження не є достатньою підставою для постановки діагнозу.

Інтерпретація звіту спостереження для постановки діагнозу виконується тільки лікарем.

Згенеровано за допомогою Daily Pain.

Рис. 4.6 — Приклад сформованого програмою звіту

5 ЕКОНОМІЧНА ЧАСТИНА

Дослідження, яке представлене у магістерській роботі та присвячене розробці та аналізу «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android», віднесено до проєктів, спрямованих на виведення на ринок. Виведення на ринок було визначено під час виконання самої роботи і розглядається як етап комерціалізації науково-технічної розробки. Цей напрямок розглядається як пріоритетний, оскільки отримані результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди.

Проте для успішної реалізації цього процесу вирішальним є здійснення пошуку зацікавленого інвестора, який виявить інтерес до втілення даного проекту, та переконання його у доцільності вкладання інвестицій у цю розробку. З цією метою були визначені наступні етапи виконання робіт:

1) проведення комерційного аудиту науково-технічної розробки, тобто визначення науково-технічного рівня та комерційного потенціалу;

2) розрахунок витрат на здійснення науково-технічної розробки;

3) розрахунок економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, розробленими В.О. Козловським, О.Й. Лесько та В.В. Кавецьким [21] та наведеними в табл. 5.1

Таблиця 5.1 — Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 5.1

10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до табл. 5.2. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів з Вінницького національного технічного університету, кафедри «Обчислювальної техніки»: кандидат технічних наук, доцент Савицька Людмила Анатоліївна; доктор технічних наук, професор Мартинюк Тетяна Борисівна; кандидат технічних наук, доцент Богомолов Сергій Віталійович. За результатами оцінки буде визначено науково-технічний рівень та комерційні потенціали розробки. Та буде зроблено висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів.

Таблиця 5.2 — Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Савицька Людмила Анатоліївна	Мартинюк Тетяна Борисівна	Богомолов Сергій Віталійович
	Бали:		
1. Технічна здійсненність концепції	3	4	3
2. Ринкові переваги (наявність аналогів)	2	3	3
3. Ринкові переваги (ціна продукту)	3	2	2
4. Ринкові переваги (технічні властивості)	4	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	1	3
6. Ринкові перспективи (розмір ринку)	2	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	3
9. Практична здійсненність (наявність фінансів)	2	2	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	$СБ_1$=34	$СБ_2$=36	$СБ_3$=39
Середньоарифметична сума балів $СБ_c$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{34+36+39}{3} = 36$		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації [21], наведені в табл. 5.3.

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android» становить 36 балів, що, відповідно до таблиці 5.3 рівень комерційного потенціалу розробки вище середнього, що свідчить про комерційну важливість проведення даних досліджень.

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи, здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями (витрати на оплату праці, відрахування на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, спецустаткування для наукових (експериментальних) робіт, програмне забезпечення для наукових (експериментальних) робіт, витрати на роботи, які виконують сторонні підприємства, установи і організації, інші витрати, накладні витрати.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та

іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=23 дні.

$$З_о = 15000 \cdot 8 / 23 = 5217 \text{ грн.}$$

Проведені розрахунки зведемо до табл. 5.4.

Таблиця 5.4 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	15000	652,1	8	5217
Системний адміністратор	13000	565,2	15	8478
Інженер-програміст	25000	1086,9	30	32608
Всього				46303

Додаткова заробітна плата дослідників та робітників.

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.2)$$

де $Н_{дод}$ — норма нарахування додаткової заробітної плати. Приймемо 12%.

$$З_{дод} = (46303) \cdot 12 / 100\% = 5556 \text{ грн.}$$

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (5.3)$$

де *Нзп* — норма нарахування на заробітну плату. Приймаємо 22%.

$$Зн = (46303+5556) \cdot 22 / 100\% = 11408 \text{ грн.}$$

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання.

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.4)$$

де *Hj* — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

Цj — вартість матеріалу j-го найменування, грн/кг;

Кj — коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Bj — маса відходів j-го найменування, кг;

Цвj — вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до табл. 5.5.

Таблиця 5.5 — Витрати за матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А4	150	1	150
Ручка	27	1	27
Flesh-накопичувач 64Гб	200	1	200
Всього			377

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі ($К_в$), які використовують при проведенні НДР на тему «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в роботі відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення. Програмні засоби та інтегровані середовища розробки, які були використанні при написанні магістерської роботи є безкоштовними.

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, устаткування, інших приладів і пристроїв, приміщень та програмному забезпеченню тощо, в дослідній організації або на підприємстві, котрі необхідні для проведення науково-дослідної роботи, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.5)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$A_{обл} = (32450 \cdot 1) / (2 \cdot 12) = 1352{,}08 \text{ грн.}$$

Проведені розрахунки зведемо до табл. 5.6.

Таблиця 5.6 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	32450	2	1	1352,08
Мобільний пристрій	10000	2	1	416,67
Всього				1768,75

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.6)$$

де W_{yi} — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,2 грн;

$K_{вni}$ — коефіцієнт, що враховує використання потужності, $K_{вni}$ <1;

η_i — коефіцієнт корисної дії обладнання, η_i <1.

$$B_e = 0{,}55 \cdot 310 \cdot 7{,}2 \cdot 0{,}5 / 0{,}8 = 767{,}25 \text{ грн.}$$

5.2.9 Службові відрядження

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (5.7)$$

де $H_{св}$ — норма нарахування за статтею «Службові відрядження», приймемо що $H_{св}$ = 25%.

$$B_{св} = (46303) \cdot 25 / 100\% = 11575{,}75 \text{ грн.}$$

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

До статті «Витрати на роботи, які виконують сторонні підприємс тва, установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.8)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$$I_в = (46303) \cdot 60 / 100\% = 27781{,}8 \text{ грн.}$$

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу тощо.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.9)$$

де $H_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $H_{нзв}$ = 110%.

$$B_{нзв} = (46303) \cdot 110 / 100\% = 50933{,}3 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему «Програмні засоби для моніторингу та аналізу стану здоров'я на платформі Android» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}, \ldots (5.10)$$

$B_{заг}$=46303+5556+11408+377+1768,75+767,25+11575,75+27781,8+50933,3 =156470,85 грн

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.11)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,9.

$$ЗВ = 156470{,}85 / 0{,}9 = 173856{,}5 \text{ грн}$$

Отже, загальний рівень витрат для завершення науково-дослідної роботи становить 173856,5 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

В даному випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN — збільшення кількості споживачів продукту, у періоди часу, що аналізуються;

N — кількість споживачів які використовували програмний засіб у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особи;

$Ц_б$ — вартість програмного продукту у році до впровадження програмного засобу, приймемо 1500,00 грн;

$\pm\Delta Ц_о$ — зміна вартості програмного продукту від впровадження результатів, приймемо зростання на 150,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.12)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт =0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =30%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році =18%;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1=(1\cdot150+1500\cdot700)\cdot0,83\cdot0,3\cdot(1-0,18/100\%)=261016,67 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2=(1\cdot150+1500\cdot(700+650))\cdot0,83\cdot0,3\cdot(1-0,18/100\%)=503354,68 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3=(1\cdot150+1500\cdot(700+650+600))\cdot0,83\cdot0,3\cdot(1-0,18/100\%)=727051,30 \text{ грн.}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4=(1\cdot150+1500\cdot(700+650+600+550))\cdot0,83\cdot0,3\cdot(1-0,18/100\%)$$
$$=932106,53 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП=\sum_{i=1}^{T}\frac{\Delta П_i}{(1+\tau)^t}, \quad (5.13)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, =18%;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП=261016,67 /(1+0,18)^1+503354,68 /(1+0,18)^2+727051,30 /(1+0,18)^3$$

$$+932106,53 /(1+0,18)^4=1505978,11 \text{ грн}$$

Далі розрахуємо величину початкових інвестицій *PV* , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ, \quad (5.14)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =3;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 173856,50 грн.

$$PV = 2 \cdot 173856,50 = 347713,00 \text{ грн.}$$

Абсолютний економічний ефект $Е_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.15)$$

де *ПП* — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1505978,11 грн;

PV — теперішня вартість початкових інвестицій 347713,00 грн.

$$Е_{абс}= 1505978,11 - 347713,00 = 1158265,11$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1+\frac{E_{абс}}{PV}} - 1, \quad (5.16)$$

де $E_{абс}$— абсолютний економічний ефект вкладених інвестицій, 1158265,11 грн;

PV — теперішня вартість початкових інвестицій, 347713,00 грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в= (1 + 1158265,11 /347713,00)^{1/4} - 1 = 0,442$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f , \quad (5.17)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$$\tau_{мін} = 0,1+0,2 = 0,3$$

Оскільки 0,3 < 0,442 це свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою

«Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.18}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 0{,}442 = 2{,}3 \text{ (2 роки і 4 місяця)}$$

Так як $T_{ок}$ < 4-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впро-

вадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android» становить 36 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 2 роки і 4 місяці, що менше 4-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android».

ВИСНОВКИ

Під час створення магістерської кваліфікаційної роботи було проведено огляд сфери застосування програмних засобів для моніторингу стану здоров’я на платформі Android, був здійснений аналіз актуальних напрямів дослідження в сфері мобільної медицини, а також огляд вже існуючих програмних рішень для моніторингу стану здоров’я.

Також було проведено аналіз та огляд програмних технологій і інструментів, які використовувалися під час створення програмного засобу для моніторингу та аналізу стану здоров’я. Обрано модель візуалізації даних для аналізу тенденції показників стану здоров’я за рівнем болю. Також було обґрунтовано вибір конкретних засобів під час розробки програмного засобу.

Проведена розробка інтерфейсу, а також розробка функціоналу програмного засобу для моніторингу та аналізу стану здоров’я. До яких входить сторінка реєстрації, сторінка категорій, сторінка переліку створених записів, сторінку аналізу, сторінки формування звітності.

Проведено тестування розробленого програмного засобу в результаті якого, додаток зарекомендував себе як досить зручний у користуванні, стійким до внесення користувачем невірних та помилкових дій. Також було розглянуто детальну інструкцію роботи з програмним засобом, для кращого розуміння всіх можливостей програмного засобу.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Програмні засоби для моніторингу та аналізу стану здоров’я на платформі Android» становить 36 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 2 роки і 4 місяці, що менше 4-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Частка ринку Android та iOS: статистика 2022 року [Електронний ресурс] — Режим доступу: https://root-nation.com/ua/news-ua/it-news-ua/ua-android-ios-statistika-2022/

2. В. І. Ковалик / МОБІЛЬНА МЕДИЦИНА ТА РОЛЬ ПРОГРАМНИХ ЗАСОБІВ У МОНІТОРИНГУ СТАНУ ЗДОРОВ'Я НА ПЛАТФОРМІ ANDROID // Тези доповіді. IX Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем (СПІРН-2023). Вінниця 2023 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/spirn/spirn2023/paper/viewFile/19169/15893

3. Мобільна платформа Android [Електронний ресурс]. — Режим доступу: http://itc.ua/articles/mobilnaya-platforma-android-pyat-let-istorii/

4. Instant Heart Rate: HR Monitor [Електронний ресурс] — Режим доступу: https://play.google.com/store/apps/details?id=si.modula.android.instantheartrate&hl=en

5. Додаток Фітнес неправильно розпізнає активність — Пристрій Android — Google Fit Довідка [Електронний ресурс] — Режим доступу: https://support.google.com/fit/answer/6075068?hl=uk&co=GENIE.Platform%3DAndroid

6. Мінцер О. П. Особливості діагностики стану здоров'я пацієнта з позиції мобільної медицини. Постановка проблеми [Текст] / О. П. Мінцер, Я. О. Шевченко // Медична інформатика та інженерія. — 2016. — № 4 — С. 31–36.

7. 13 Scientific Reason Why Your Brain Craves Infographics. — [Електронний ресурс] — Режим доступу: https://neomam.com/interactive/13reasons/

8. Лінійна діаграма — Вікіпедія — [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B0_%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B

9. Стовпчикова діаграма — Вікіпедія — [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D0%B2%D0%BF

%D1%87%D0%B8%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0

10. Секторна діаграма — Вікіпедія — [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D0%B0_%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0

11. Гістограма — Вікіпедія [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%93%D1%96%D1%81%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0

12. Все про Android [Електронний ресурс] — Режим доступу: http://androiddocs.ru/

13. Google Android [Електронне джерело] — Режим доступу: http://www.androidtalk.ru/google-android/

14. Android Emulator [Електронний ресурс] — Режим доступу: http://developer.android.com/tools/help/emulator.html

15. Архітектура Android-додатків [Електронний ресурс] — Режим доступу: http://habrahabr.ru/post/141201/

16. Dream(sheep++): A developer’s introduction to Google Android [Електронний ресурс] — Режим доступу до ресурсу: https://arstechnica.com/gadgets/2009/02/an-introduction-to-google-android-for-developers/

17. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009 — Режим доступу: https://habr.com/post/53536/

18. Bill Phillips, Chris Stewart & Kristin Marsicano — Android Programming: The Big Nerd Ranch Guide [Електронний ресурс] — 2015. — Режим доступу: https://www.amazon.com/Android-Programming-Nerd-Ranch-Guide/dp/0134171454

19. П. Дейтел, Х. Дейтел, А. Уолд — Android для разработчиков [Електронний ресурс] — 2016. — Режим доступу: https://www.ozon.ru/context/detail/id/136331151/

20. Репозиторій Daily_Pain_Master_Project_Java [Електронний ресурс] — Режим доступу: https://github.com/xo4yxypmy/Daily_Pain_Master_Project_Java

21. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

« 29 » вересня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Програмні засоби для моніторингу та аналізу стану здоров'я на платформі Android"

08-54.МКР.007.00.000 ТЗ

Науковий керівник: доцент к.т.н.

_____ Кадук О.В.

Студент групи 1КІ-22м

_____ Ковалик В.І.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Важливим є актуальність дослідження у напрямку магістерської роботи, яка обумовлена тим, що в даний час сфера засобів для моніторингу стану здоров'я на платформі Android переживає значний ріст і розвиток. Багато засобів взаємодіють із сучасними смарт-пристроями, такими як фітнес-годинники, датчики серцебиття, глюкометри тощо. Це дозволяє отримувати більш точні та докладні дані. Проте, незважаючи на наявність численних медичних додатків для Android, багато з них обмежені функціональністю, не надають засобів аналізу та взаємодії зі зібраними даними, та не дозволяють користувачам зручно створювати звіти або експортувати дані в зручних форматах.

1.2 Наказ про затвердження теми МКР №247 від 18.09.2023 р.

2 Мета МКР і призначення розробки

2.1 Мета роботи — є аналіз сучасного стану і тенденцій розвитку програмних засобів у сфері моніторингу та аналізу стану здоров’я та розробка програмного засобу, який дозволить користувачам зручно вести облік симптомів, болю та інших показників стану здоров’я, проводити їх простий аналіз, формувати звіти та експортувати їх в форматі PDF.

2.2 Призначення розробки — в наданні середньостатистичним користувачам зрозумілий спосіб об’єктивного моніторингу та аналізу тенденцій симптомів що сприятиме більшій самосвідомості щодо їхнього стану здоров'я та забезпечить індивідуалізований підхід до кожного користувача .

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих досліджень та програмних рішень у моніторингу та аналізу стану здоров’я.

3.2 Розробка структури та функціональності програмно засобу для моніторингу стану здоров’я.

3.3 Проведення тестування розробленого програмного засобу для перевірки справності роботи та виявлення несправностей.

3.4 Виконання розрахунків для доведення доцільності науково-дослідної роботи з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — використати, модель візуалізації даних, для моніторингу та аналізу тенденції симптомів за рівнем болю.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз існуючих досліджень та програмних рішень у моніторингу та аналізу стану здоров’я	21.09.2023	25.09.2023	Розділ 1
2	Дослідження методів розробки програмного засобу для моніторингу та аналізу	26.09.2023	06.10.2023	Розділ 2
3	Розробка програмного засобу та реалізація функціональності	07.10.2023	25.10.2023	Розділ 3
4	Проведення тестування програмного засобу	26.10.2023	05.11.2023	Розділ 4
5	Підготовка розрахунків економічної частини	06.11.2023	15.11.2023	Розділ 5
5	Апробація та впровадження результатів дослідження	15.11.2023	17.11.2023	Тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	18.11.2023	01.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	02.12.2023	16.12.2023	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— методичні вказівки до виконання студентами-магістрантами технічних спеціальностей економічної частини магістерських кваліфікаційних робіт;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Схема архітектури системи Android

Рис. Б.1 — Схема архітектури системи Android

ДОДАТОК В

Лістинг файлу DBHandler.java

```
package com.fernfog.dailyPain;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import com.fernfog.dailyPain.objects.SymptomCategory;
import com.fernfog.dailyPain.objects.Symptome;
import com.fernfog.dailyPain.objects.User;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
public class DBHandler extends SQLiteOpenHelper {
    public DBHandler(Context context) {
        super(context, "myDataBase", null, 5);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String queryUsers = "CREATE TABLE users (id INT PRIMARY KEY,
first_name TEXT, last_name TEXT, patronymic TEXT, birthday TEXT)";
        String querySymptomCategories = "CREATE TABLE symptomeCategories
(id INT PRIMARY KEY, nameOfSymptomCategory TEXT,
colorOfSymptomCategory TEXT)";
        String querySymptoms = "CREATE TABLE symptoms (id INTEGER
PRIMARY KEY, painLVL FLOAT, categoryId INTEGER, nameOfCategory
TEXT, startOfPain TEXT, endOfPain TEXT, startTime TEXT, endTime TEXT,
additional TEXT)";
        db.execSQL(queryUsers);
        db.execSQL(querySymptomCategories);
        db.execSQL(querySymptoms);
    }
    public void addNewUser(String first_name, String last_name, String
patronymic, String birthday) {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            ContentValues values = new ContentValues();
            values.put("first_name", first_name);
            values.put("last_name", last_name);
            values.put("patronymic", patronymic);
            values.put("birthday", birthday);
            db.insert("users", null, values);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
    public void addNewSymptomeCategory(String nameOfSymptomCategory,
String colorOfSymptomCategory) {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            ContentValues values = new ContentValues();
            values.put("nameOfSymptomCategory", nameOfSymptomCategory);
            values.put("colorOfSymptomCategory", colorOfSymptomCategory);
            db.insert("symptomeCategories", null, values);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void addNewSymptom(float painLVL, String categoryName, String
startOfPain, String endOfPain, String startTime, String endTime, String
additional) {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            int categoryId = getCategoryIdByName(categoryName, db);
            ContentValues values = new ContentValues();
            values.put("categoryId", categoryId);
            values.put("painLVL", Math.round(painLVL));
            values.put("startOfPain", startOfPain);
            values.put("endOfPain", endOfPain);
            values.put("startTime", startTime);
            values.put("endTime", endTime);
            values.put("nameOfCategory", categoryName);
            values.put("additional", additional);
            db.insert("symptoms", null, values);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public List<Symptome> getAllSymptomes() {
        List<Symptome> symptomes = new ArrayList<>();
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            String query = "SELECT * FROM symptoms";
            try (Cursor cursor = db.rawQuery(query, null)) {
                if (cursor != null && cursor.moveToFirst()) {
                    do {
                        float painLVL =
cursor.getFloat(cursor.getColumnIndexOrThrow("painLVL"));
                        String startOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("startOfPain"));
                        String endOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("endOfPain"));
```

```
                String startTime =
cursor.getString(cursor.getColumnIndexOrThrow("startTime"));
                String endTime =
cursor.getString(cursor.getColumnIndexOrThrow("endTime"));
                String nameOfCategory =
cursor.getString(cursor.getColumnIndexOrThrow("nameOfCategory"));
                String additional =
cursor.getString(cursor.getColumnIndexOrThrow("additional"));
                Symptome category = new Symptome(startOfPain, startTime,
endOfPain, endTime, nameOfCategory, painLVL, additional);
                symptomes.add(category);
            } while (cursor.moveToNext());
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return symptomes;
  }
  public List<Symptome> getAllSymptomesInCategory(String categoryName) {
    List<Symptome> symptomes = new ArrayList<>();
    try (SQLiteDatabase db = this.getReadableDatabase()) {
      String query = "SELECT * FROM symptoms WHERE nameOfCategory
= ?";
      try (Cursor cursor = db.rawQuery(query, new String[]{categoryName})) {
        if (cursor != null && cursor.moveToFirst()) {
          do {
            float painLVL =
cursor.getFloat(cursor.getColumnIndexOrThrow("painLVL"));
            String startOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("startOfPain"));
            String endOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("endOfPain"));
            String startTime =
cursor.getString(cursor.getColumnIndexOrThrow("startTime"));
            String endTime =
cursor.getString(cursor.getColumnIndexOrThrow("endTime"));
            String nameOfCategory =
cursor.getString(cursor.getColumnIndexOrThrow("nameOfCategory"));
            String additional =
cursor.getString(cursor.getColumnIndexOrThrow("additional"));
```

```
                    Symptome symptome = new Symptome(startOfPain, startTime,
endOfPain, endTime, nameOfCategory, painLVL, additional);
                    symptomes.add(symptome);
                } while (cursor.moveToNext());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return symptomes;
}
public List<Symptome> getAllSymptomesWithPainLevel(float painLevel) {
    List<Symptome> symptomes = new ArrayList<>();
    try (SQLiteDatabase db = this.getReadableDatabase()) {
        String query = "SELECT * FROM symptoms WHERE painLVL = ?";
        try (Cursor cursor = db.rawQuery(query, new
String[]{String.valueOf(painLevel)})) {
            if (cursor != null && cursor.moveToFirst()) {
                do {
                    float painLVL =
cursor.getFloat(cursor.getColumnIndexOrThrow("painLVL"));
                    String startOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("startOfPain"));
                    String endOfPain =
cursor.getString(cursor.getColumnIndexOrThrow("endOfPain"));
                    String startTime =
cursor.getString(cursor.getColumnIndexOrThrow("startTime"));
                    String endTime =
cursor.getString(cursor.getColumnIndexOrThrow("endTime"));
                    String nameOfCategory =
cursor.getString(cursor.getColumnIndexOrThrow("nameOfCategory"));
                    String additional =
cursor.getString(cursor.getColumnIndexOrThrow("additional"));

                    Symptome symptome = new Symptome(startOfPain, startTime,
endOfPain, endTime, nameOfCategory, painLVL, additional);
                    symptomes.add(symptome);
                } while (cursor.moveToNext());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return symptomes;
    }
    private int getCategoryIdByName(String categoryName, SQLiteDatabase db)
{
        int categoryId = -1;
        String query = "SELECT id FROM symptomeCategories WHERE
nameOfSymptomCategory = ?";
        try (Cursor cursor = db.rawQuery(query, new String[]{categoryName})) {
            if (cursor != null && cursor.moveToFirst()) {
                categoryId = cursor.getInt(cursor.getColumnIndexOrThrow("id"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return categoryId;
    }
    public SymptomCategory getCategoryById(int categoryId) {
        SymptomCategory category = null;
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            String query = "SELECT * FROM symptomeCategories WHERE id = ?";
            try (Cursor cursor = db.rawQuery(query, new
String[]{String.valueOf(categoryId)})) {
                if (cursor != null && cursor.moveToFirst()) {
                    String categoryName =
cursor.getString(cursor.getColumnIndexOrThrow("nameOfSymptomCategory"));
                    String categoryColor =
cursor.getString(cursor.getColumnIndexOrThrow("colorOfSymptomCategory"));
                    category = new SymptomCategory(categoryName, categoryColor);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return category;
    }
    public List<SymptomCategory> getAllCategoriesWithColors() {
        List<SymptomCategory> categories = new ArrayList<>();
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            String query = "SELECT * FROM symptomeCategories";
            try (Cursor cursor = db.rawQuery(query, null)) {
```

```
            if (cursor != null && cursor.moveToFirst()) {
                do {
                    String categoryName =
cursor.getString(cursor.getColumnIndexOrThrow("nameOfSymptomCategory"));
                    String categoryColor =
cursor.getString(cursor.getColumnIndexOrThrow("colorOfSymptomCategory"));

                    SymptomCategory category = new
SymptomCategory(categoryName, categoryColor);
                    categories.add(category);
                } while (cursor.moveToNext());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return categories;
}
public User getUser() {
    User user = null;
    try (SQLiteDatabase db = this.getReadableDatabase()) {
        String query = "SELECT * FROM users";
        try (Cursor cursor = db.rawQuery(query, null)) {
            if (cursor.moveToFirst()) {
                String first_name =
cursor.getString(cursor.getColumnIndexOrThrow("first_name"));
                String last_name =
cursor.getString(cursor.getColumnIndexOrThrow("last_name"));
                String patronymic =
cursor.getString(cursor.getColumnIndexOrThrow("patronymic"));
                String birthday =
cursor.getString(cursor.getColumnIndexOrThrow("birthday"));
                user = new User(first_name, last_name, patronymic, birthday);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return user;
}
public boolean hasUsers() {
```

```
        boolean hasUsers = false;
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            String query = "SELECT * FROM users";
            try (Cursor cursor = db.rawQuery(query, null)) {
                hasUsers = cursor.getCount() > 0;
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return hasUsers;
    }
    public int getSymptomId(float painLVL, String categoryName, String
startOfPain, String endOfPain) {
        int symptomId = -1;
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            int categoryId = getCategoryIdByName(categoryName, db);
            String query = "SELECT id FROM symptoms WHERE categoryId = ?
AND painLVL = ? AND startOfPain = ? AND endOfPain = ?";
            try (Cursor cursor = db.rawQuery(query, new
String[]{String.valueOf(categoryId), String.valueOf(painLVL), startOfPain,
endOfPain})) {
                if (cursor != null && cursor.moveToFirst()) {
                    symptomId = cursor.getInt(cursor.getColumnIndexOrThrow("id"));
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return symptomId;
    }
    public void deleteSymptom(int symptomId) {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            db.delete("symptoms", "id = ?", new
String[]{String.valueOf(symptomId)});
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void deleteCategory(String categoryName) {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            int categoryId = getCategoryIdByName(categoryName, db);
```

```
            db.delete("symptoms", "categoryId = ?", new
String[]{String.valueOf(categoryId)});
            db.delete("symptomeCategories", "nameOfSymptomCategory = ?", new
String[]{categoryName});
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public String getCategoryColorByName(String categoryName) {
        String color = "000000";
        try (SQLiteDatabase db = this.getReadableDatabase()) {
            String query = "SELECT colorOfSymptomCategory FROM
symptomeCategories WHERE nameOfSymptomCategory = ?";
            try (Cursor cursor = db.rawQuery(query, new String[]{categoryName})) {
                if (cursor != null && cursor.moveToFirst()) {
                    color =
cursor.getString(cursor.getColumnIndexOrThrow("colorOfSymptomCategory"));
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return color;
    }
    public void DELTEALLLLLLL() {
        try (SQLiteDatabase db = this.getWritableDatabase()) {
            db.execSQL("DELETE FROM users");
            db.execSQL("DELETE FROM symptomeCategories");
            db.execSQL("DELETE FROM symptoms");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static boolean checkDatabaseExists(Context context) {
        File dbFile = context.getDatabasePath("myDataBase");
        return dbFile.exists();
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS users");
        db.execSQL("DROP TABLE IF EXISTS symptomeCategories");
        db.execSQL("DROP TABLE IF EXISTS symptoms");
        onCreate(db);
```

```
            }
    }
```

ДОДАТОК Г

Лістинг файлу ReportFragment.java

```
package com.fernfog.dailyPain;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.os.Environment;
import android.provider.DocumentsContract;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;
import com.fernfog.dailyPain.objects.SymptomCategory;
import com.fernfog.dailyPain.objects.Symptome;
import com.fernfog.dailyPain.objects.User;
import com.itextpdf.io.font.PdfEncodings;
import com.itextpdf.kernel.font.PdfFont;
import com.itextpdf.kernel.font.PdfFontFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.Cell;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;
import com.itextpdf.layout.property.TextAlignment;
import com.itextpdf.layout.property.UnitValue;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
public class ReportFragment extends Fragment {
    Button button;
    Context context;
    static DBHandler dbHandler;
```

```
    static List<SymptomCategory> symptomes;
    String categorySelected;
    Button openFileDirectory;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_report, container, false);
        button = view.findViewById(R.id.testBSLKFJM);
        Spinner spinner = view.findViewById(R.id.spinner);
        context = getContext();
        dbHandler = new DBHandler(context);
        symptomes = dbHandler.getAllCategoriesWithColors();
        List<String> symptomesNames = new ArrayList<>();
        openFileDirectory = view.findViewById(R.id.openDirectoryButton);
        for (SymptomCategory symptome : symptomes) {
            symptomesNames.add(symptome.getName());
        }
        for (String categoryName : symptomesNames) {
            Log.d("CategoryName", categoryName);
        }
        ArrayAdapter<String> adapter = new ArrayAdapter<>(
                context,
                android.R.layout.simple_spinner_item,
                symptomesNames
        );

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown
_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                categorySelected = parent.getItemAtPosition(position).toString();
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        openFileDirectory.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                File myPdfFolder = new
File(Environment.getExternalStoragePublicDirectory(
                        Environment.DIRECTORY_DOCUMENTS), "MyPDFs");
                Intent intent = new
Intent(Intent.ACTION_OPEN_DOCUMENT_TREE);
                Uri uri = Uri.fromFile(myPdfFolder);
                intent.putExtra(DocumentsContract.EXTRA_INITIAL_URI, uri);
                startActivity(intent);
            }
        });
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                generatePdf(categorySelected, view.getContext());
            }
        });
        return view;
    }
    public static void generatePdf(String category, Context context) {
        User user = dbHandler.getUser();
        File pdfDirectory = new
File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_DOCUMENTS), "MyPDFs");
        if (!pdfDirectory.exists()) {
            pdfDirectory.mkdirs();
        }
        Calendar calendar = Calendar.getInstance();
        Date currentDate = calendar.getTime();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH.mm.ss", Locale.getDefault());
        String formattedDate = dateFormat.format(currentDate);
        String filePath = new File(pdfDirectory, "звіт-" + formattedDate +
".pdf").getAbsolutePath();
        try {
            String FONT = "/assets/fonts/Montserrat-Regular.ttf";
            PdfFont font = PdfFontFactory.createFont(FONT,
PdfEncodings.IDENTITY_H);
            PdfWriter writer = new PdfWriter(filePath);
            PdfDocument pdf = new PdfDocument(writer);
            Document document = new Document(pdf);
            document.setFont(font);
            List<Symptome> symptomeList =
dbHandler.getAllSymptomesInCategory(category);
```

```
        document.add(new Paragraph("Звіт
спостереження").setTextAlignment(TextAlignment.CENTER).setUnderline());
        document.add(new Paragraph("ПІБ: " + user.getFirstName() + " " +
user.getLastName() + " " + user.getPatronymic()));
        document.add(new Paragraph("Дата народження: " +
user.getBirthday()));
        document.add(new Paragraph("Категорія спостереження: " +
category));
        document.add(new Paragraph("Загальна кількість симптомів: " +
symptomeList.size()));
        document.add(new Paragraph("Кількість симптомів за рівнем болю:
"));
        Table table_ = new Table(UnitValue.createPercentArray(new float[]{1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1}));
        Cell headerCell1_ = new Cell().add(new Paragraph("0"));
        Cell headerCell2_ = new Cell().add(new Paragraph("1"));
        Cell headerCell3_ = new Cell().add(new Paragraph("2"));
        Cell headerCell4_ = new Cell().add(new Paragraph("3"));
        Cell headerCell5_ = new Cell().add(new Paragraph("4"));
        Cell headerCell6_ = new Cell().add(new Paragraph("5"));
        Cell headerCell7_ = new Cell().add(new Paragraph("6"));
        Cell headerCell8_ = new Cell().add(new Paragraph("7"));
        Cell headerCell9_ = new Cell().add(new Paragraph("8"));
        Cell headerCell10_ = new Cell().add(new Paragraph("9"));
        Cell headerCell11_ = new Cell().add(new Paragraph("10"));
        table_.addCell(headerCell1_);
        table_.addCell(headerCell2_);
        table_.addCell(headerCell3_);
        table_.addCell(headerCell4_);
        table_.addCell(headerCell5_);
        table_.addCell(headerCell6_);
        table_.addCell(headerCell7_);
        table_.addCell(headerCell8_);
        table_.addCell(headerCell9_);
        table_.addCell(headerCell10_);
        table_.addCell(headerCell11_);
        for (float i = 0; i < 11; i++) {
            Cell cell1 = new Cell().add(new
Paragraph(String.valueOf(dbHandler.getAllSymptomesWithPainLevel(i).size())))
;
            table_.addCell(cell1);
        }
        document.add(table_);
        document.add(new Paragraph("Перелік введених користувачем
симптомів: "));
```

```
            Table table = new Table(5);
            Cell headerCell1 = new Cell().add(new Paragraph("№"));
            Cell headerCell2 = new Cell().add(new Paragraph("Початок"));
            Cell headerCell3 = new Cell().add(new Paragraph("Кінець"));
            Cell headerCell4 = new Cell().add(new Paragraph("Рівень болю"));
            Cell headerCell5 = new Cell().add(new Paragraph("Додатково"));
            table.addCell(headerCell1);
            table.addCell(headerCell2);
            table.addCell(headerCell3);
            table.addCell(headerCell4);
            table.addCell(headerCell5);
            for (int i = 0; i < symptomeList.size(); i++) {
                Symptome symptome = symptomeList.get(i);
                Cell cell1 = new Cell().add(new Paragraph(String.valueOf(i + 1)));
                Cell cell2 = new Cell().add(new
Paragraph(symptome.getStartOfPainTime() + " " +
symptome.getStartOfPainDate()));
                Cell cell3 = new Cell().add(new
Paragraph(symptome.getEndOfPainTime() + " " +
symptome.getEndOfPainDate()));
                Cell cell4 = new Cell().add(new
Paragraph(String.valueOf(Math.round(symptome.getPainLvl()))));
                Cell cell5 = new Cell().add(new
Paragraph(String.valueOf(symptome.getAdditional())));
                table.addCell(cell1);
                table.addCell(cell2);
                table.addCell(cell3);
                table.addCell(cell4);
                table.addCell(cell5);
            }
            document.add(table);
            document.add(new Paragraph("Звіт сформовано на основі введених
користувачем даних."));
            document.add(new Paragraph("Результат звіту спостереження не є
достатньою підставою для постановки діагнозу."));
            document.add(new Paragraph("Інтерпретація звіту спостереження для
постановки діагнозу виконується тільки лікарем."));
            document.add(new Paragraph("Згенеровано за допомогою Daily
Pain."));
            document.close();
            Toast.makeText(context, "Звіт створено",
Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            }
    }
```

ДОДАТОК Д

Демонстраційний матеріал

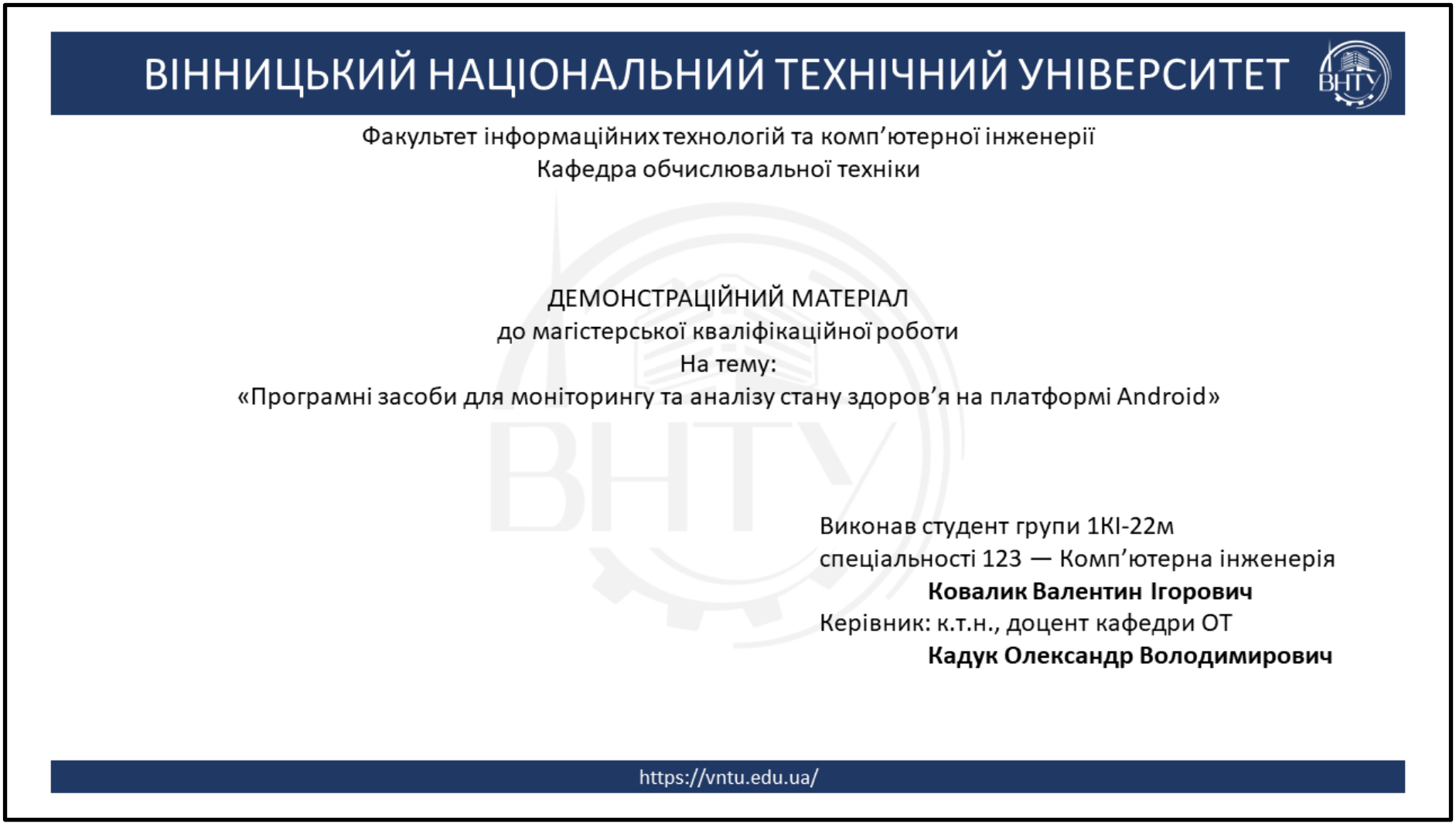


Рис. Д.1 — Слайд №1

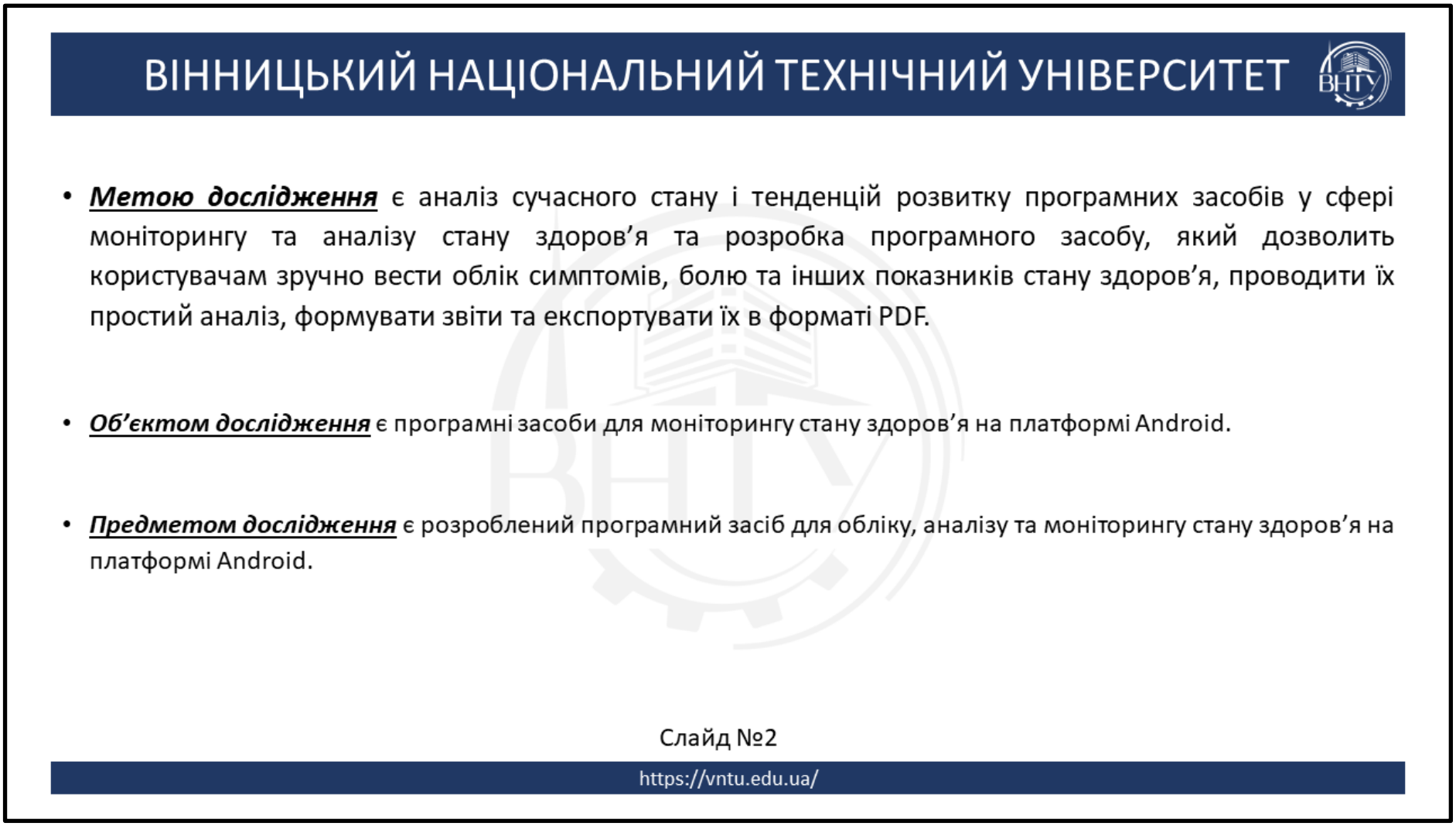


Рис. Д.2 — Слайд №2

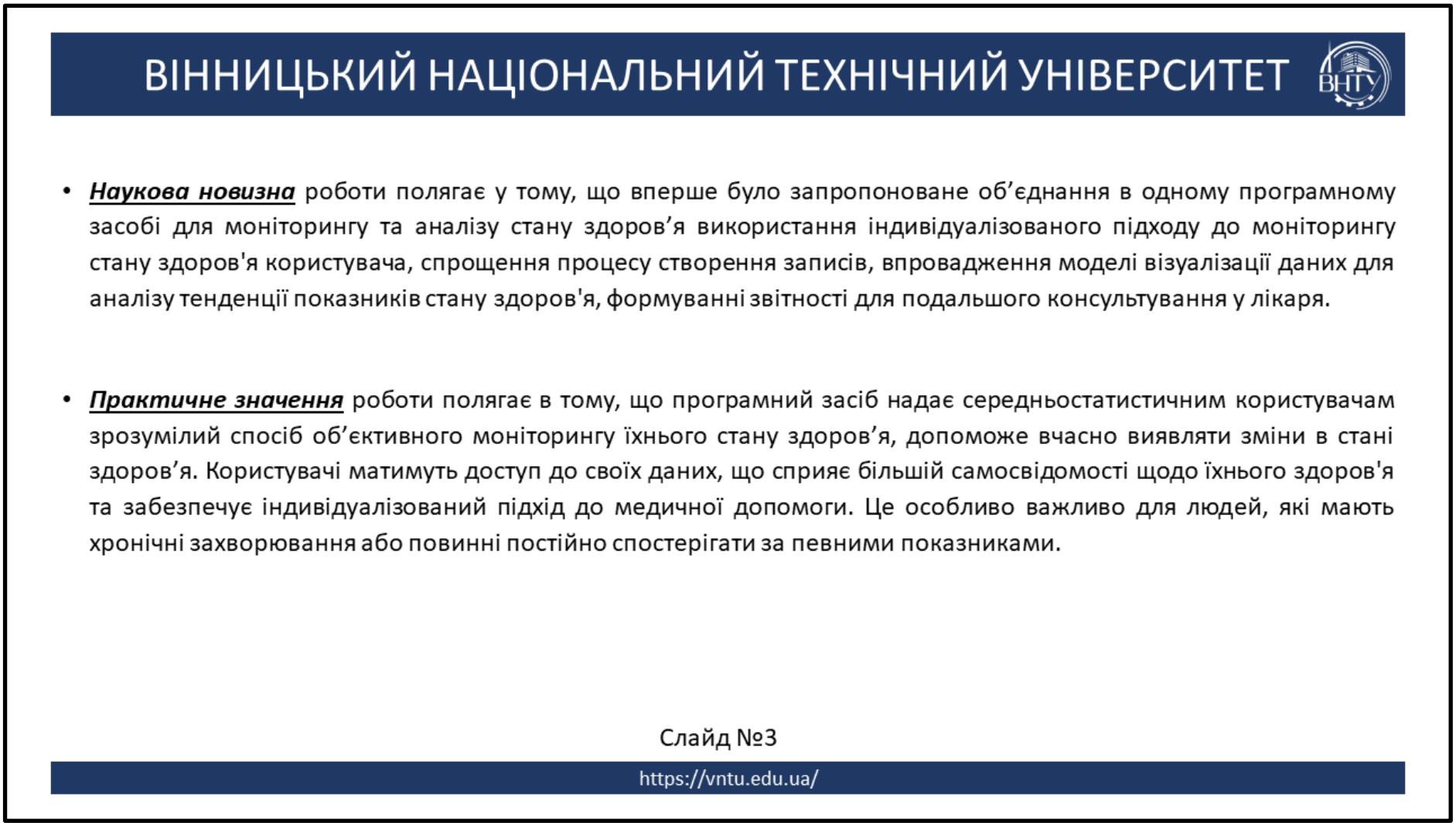


Рис. Д.3 — Слайд №3

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Задачі дослідження:

— аналіз актуальних досліджень та існуючих програмних рішень для моніторингу та аналізу стану здоров'я на платформі Android;

— розробка інтерфейсу програмного засобу;

— розробка програмного засобу та реалізація аналітичних інструментів;

— створення функціоналу для формування звітності;

— тестування розробленого програмного засобу;

— проведення розрахунків економічної ефективності.

Слайд №4

https://vntu.edu.ua/

Рис. Д.4 — Слайд №4

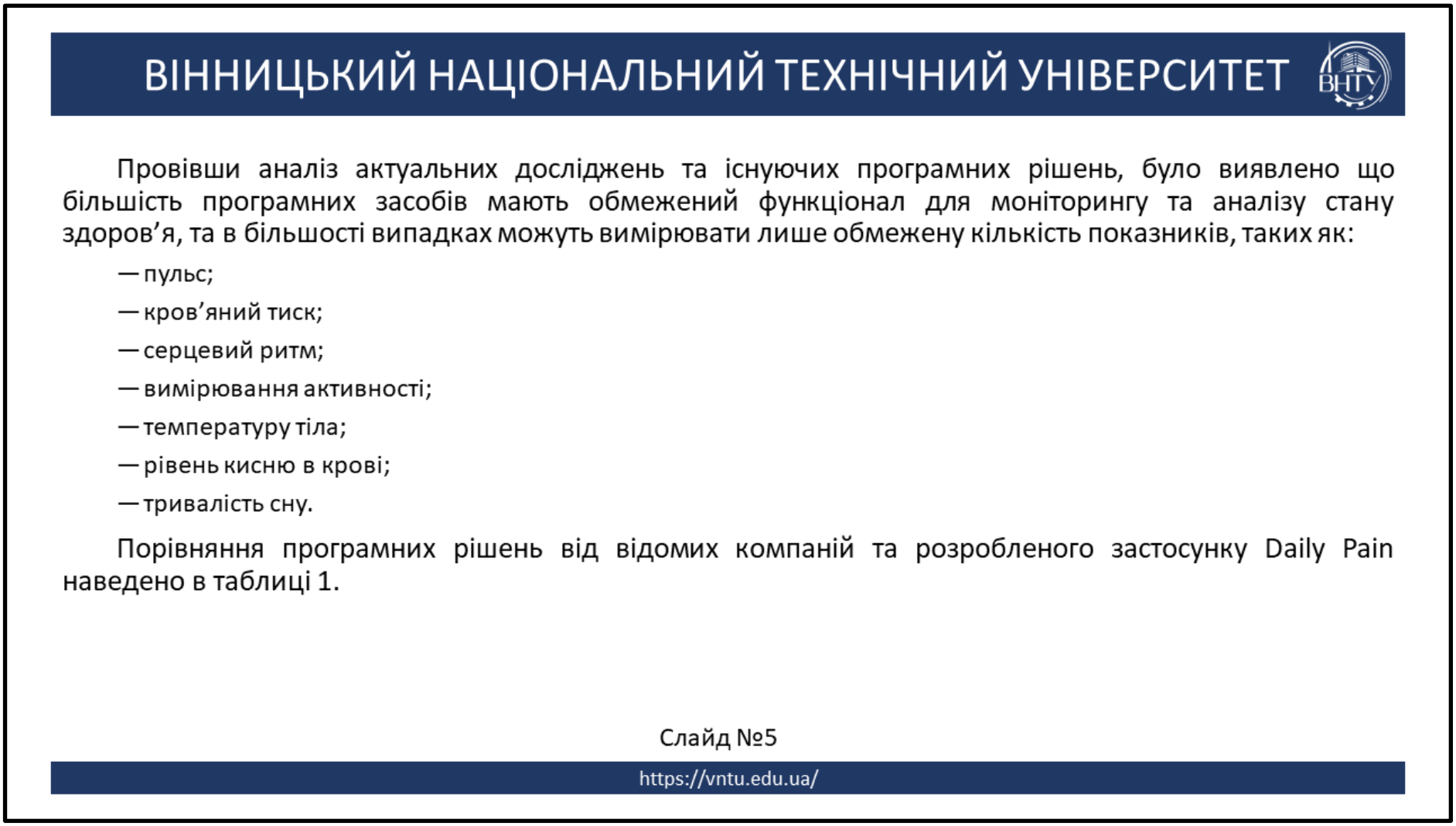


Рис. Д.5 — Слайд №5

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Таблиця 1 — Порівняння програмних засобів від відомих компаній та розробленого застосунку Daily Pain

Критерій порівнювання	Daily Pain	Health Kit	Google Fit	Medisafe
Кросплатформеність	Ні	Ні	Так	Ні
Додавання власних показників	Так	Ні	Ні	Ні
Взаємодія зі сторонніми пристроями	Ні	Так	Так	Ні
Створення записів про показник	Так	Ні	Ні	Ні
Проведення аналізу показників	Так	Так	Так	Так
Можливість формування звіту спостереження	Так	Так	Ні	Так
Необхідність підключення до мережі Інтернет	Ні	Ні	Ні	Ні

Слайд №6

https://vntu.edu.ua/

Рис. Д.6 — Слайд №6

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Перед тим як приступити до розробки самого програмного засобу, було розроблено попередній дизайн інтерфейсу в векторному сервісі розробки інтерфейсів Figma рисунок 1.

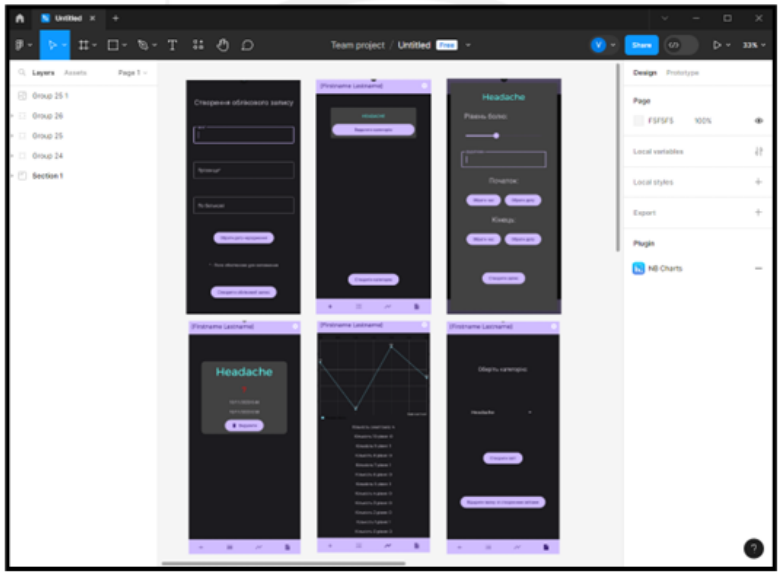


Рис. 1 — Розробка інтерфейсу в Figma

Слайд №7

https://vntu.edu.ua/

Рис. Д.7 — Слайд №7

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Для розробки програмного засобу Daily Pain на платформі Android було обрано офіційне інтегроване середовище розробки Android Studio та мову програмування Java.

Перевагами Android Studio є:

— офіційна IDE для Android;
— повна підтримка Java;
— ефективність та продуктивність;
— емулятор та можливість підключення реального пристрою;
— широкий вибір інструментів та плагінів.

Перевагами Java є:

— об'єктно-орієнтована мова програмування;
— платформова незалежність;
— велика кількість бібліотек та фреймворків;
— велика спільнота розробників;
— масштабованість та продуктивність.

Слайд №8

https://vntu.edu.ua/

Рис. Д.8 — Слайд №8

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Створено сторінку для створення категорій, вікно для створення категорій та вікно для створення записів, рисунок 2.

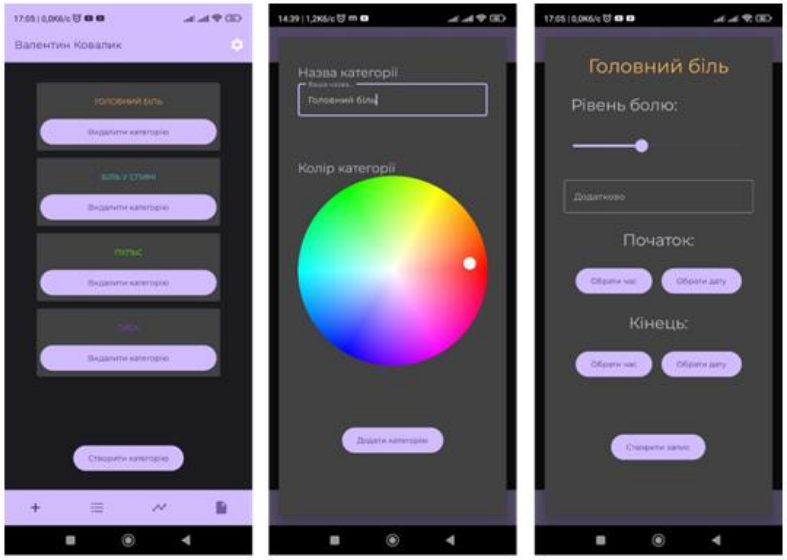


Рис. 2 — Сторінка створення категорії та запису

Слайд №9

https://vntu.edu.ua/

Рис. Д.9 — Слайд №9

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Для проведення моніторингу та аналізу показників стану здоров'я, створено сторінку переліку записів, сторінку аналізу та сторінку для створення звітності, рисунок 3.

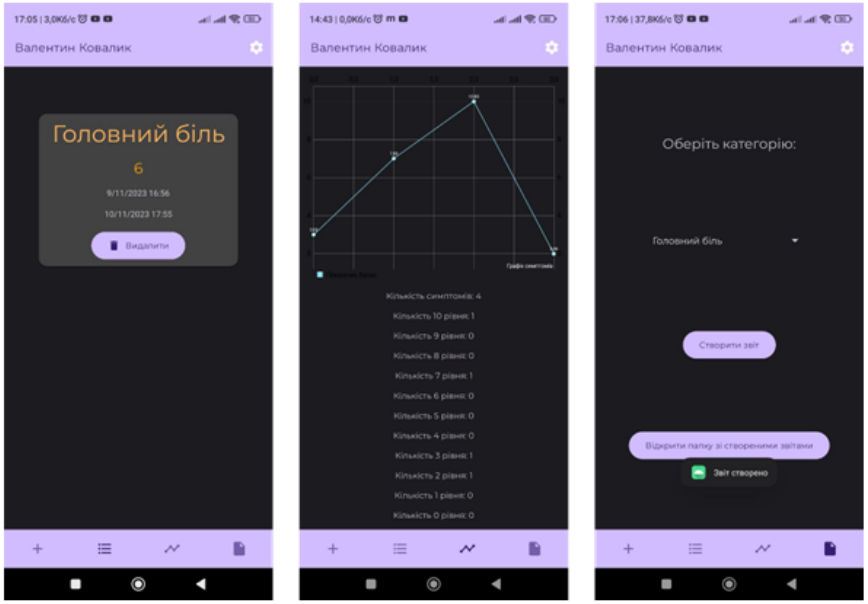


Рис. 3 — Сторінка переліку записів, сторінка аналізу, сторінка створення звіту

Слайд №10

https://vntu.edu.ua/

Рис. Д.10 — Слайд №10

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

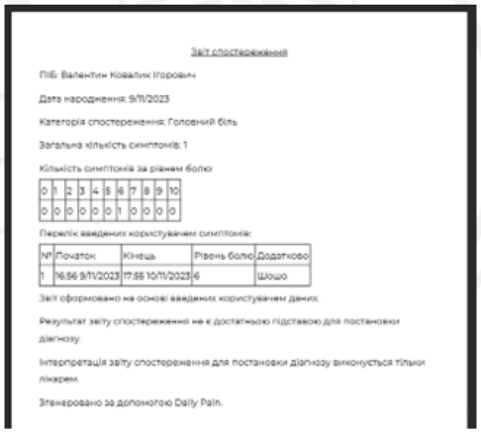
Звіт з спостереження формується для однієї категорії, котру обирає користувач, окрім даних введених користувачем в звіті присутні 3 особливі примітки:

1) Звіт сформовано на основі введених користувачем даних.

2) Результат звіту спостереження не є достатньою підставою для постановки діагнозу.

3) Інтерпретація звіту спостереження для постановки діагнозу виконується тільки лікарем.

Приклад звіту наведено в рисунку 4.



Звіт спостереження

ПІБ: Валентин Ковалик Ігорович

Дата народження: 9/11/2023

Категорія спостереження: Головний біль

Загальна кількість симптомів: 1

Кількість симптомів за рівнем болю:

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	1	0	0	0	0

Перелік введених користувачем симптомів:

№	Початок	Кінець	Рівень болю	Додатково
1	16:56 9/11/2023	17:55 10/11/2023	6	Шошо

Звіт сформовано на основі введених користувачем даних.

Результат звіту спостереження не є достатньою підставою для постановки діагнозу.

Інтерпретація звіту спостереження для постановки діагнозу виконується тільки лікарем.

Згенеровано за допомогою Daily Pain.

Рис. 4 — Приклад сформованого звіту

Слайд №11

https://vntu.edu.ua/

Рис. Д.11 — Слайд №11

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

За результатами тестування розробленого програмного засобу, додаток зарекомендував себе як досить зручний у користуванні, стійким до внесення користувачем невірних та помилкових дій. Чек-ліст тестування засобу на Android 7.0 наведено в таблиці 2.

Таблиця 2 — Чек-ліст програмного засобу Daily Pain на Android 7.0

Daily Pain	Google Pixel 6
	Android 7.0
Встановлення додатку	
Перевірка відсутності аварійного завершення програми	Passed
Перевірка встановлення програми на пристрій	Passed
Функції на пристроях	
Перевірка голосового введення даних (якщо підтримується)	Passed
Перевірка відображення у портретній/ландшафтній орієнтації	Passed
Перевірка коректності формування звіту	Passed
Графічні елементи	
Перевірка швидкості відгуку елемента	Passed
Розмір елемента дає можливість натиснути на нього	Passed
Реакція програмного засобу на зовнішні переривання	
Реагування програмного засобу на блокування/розблокування екрану	Passed
Реагування програмного засобу на заряджання пристрою	Passed
Реагування програмного засобу на вимкнення та підключення до мережі Wi-Fi	Passed
Реагування програмного засобу на вхідні та вихідні дзвінки	Passed

Слайд №12

https://vntu.edu.ua/

Рис. Д.12 — Слайд №12

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Таблиця 3 — Показники економічної ефективності

Показник	Сума витрат/надходжень, грн.
Загальні витрати	173856,5
Збільшення чистого прибутку 1-го року	261016,67
Збільшення чистого прибутку 2-го року	503354,68
Збільшення чистого прибутку 3-го року	727051,30
Збільшення чистого прибутку 4-го року	932106,53
Приведена вартість збільшення всіх чистих прибутків	1505978,11
Величина початкових інвестицій	347713,00
Абсолютний економічний ефект	1158265,11

Період окупності інвестицій, становить 2 роки і 4 місяці, що є менше ніж 4 роки запланованого корисного використання

Слайд №13

https://vntu.edu.ua/

Рис. Д.13 — Слайд №13

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Висновки

- На сьогоднішній день програмні засоби для моніторингу та аналізу стану здоров'я є вкрай обмеженими в функціоналі, мають не велику кількість показників для моніторингу, та не надають можливість формувати звітність.
- За допомогою Android Studio та мови програмування Java, було розроблено програмний засіб Daily Pain, котрий дозволяє користувачам створювати записи симптомів/показників, проводити їх простий аналіз, формувати звітність для певного симптому/показника для консультації у лікаря.
- За результатами тестування розробленого програмного засобу, застосунок зарекомендував себе стійким до внесення користувачем невірних та помилкових дій, зручним у користуванні та має інтуїтивно зрозумілий інтерфейс.
- За результатами розрахунків економічної ефективності, рівень комерційного потенціалу розробки є вище середнього, потребує не значних інвестицій та має термін окупності 2 роки і 4 місяці.

Слайд №14

https://vntu.edu.ua/

Рис. Д.14 — Слайд №14

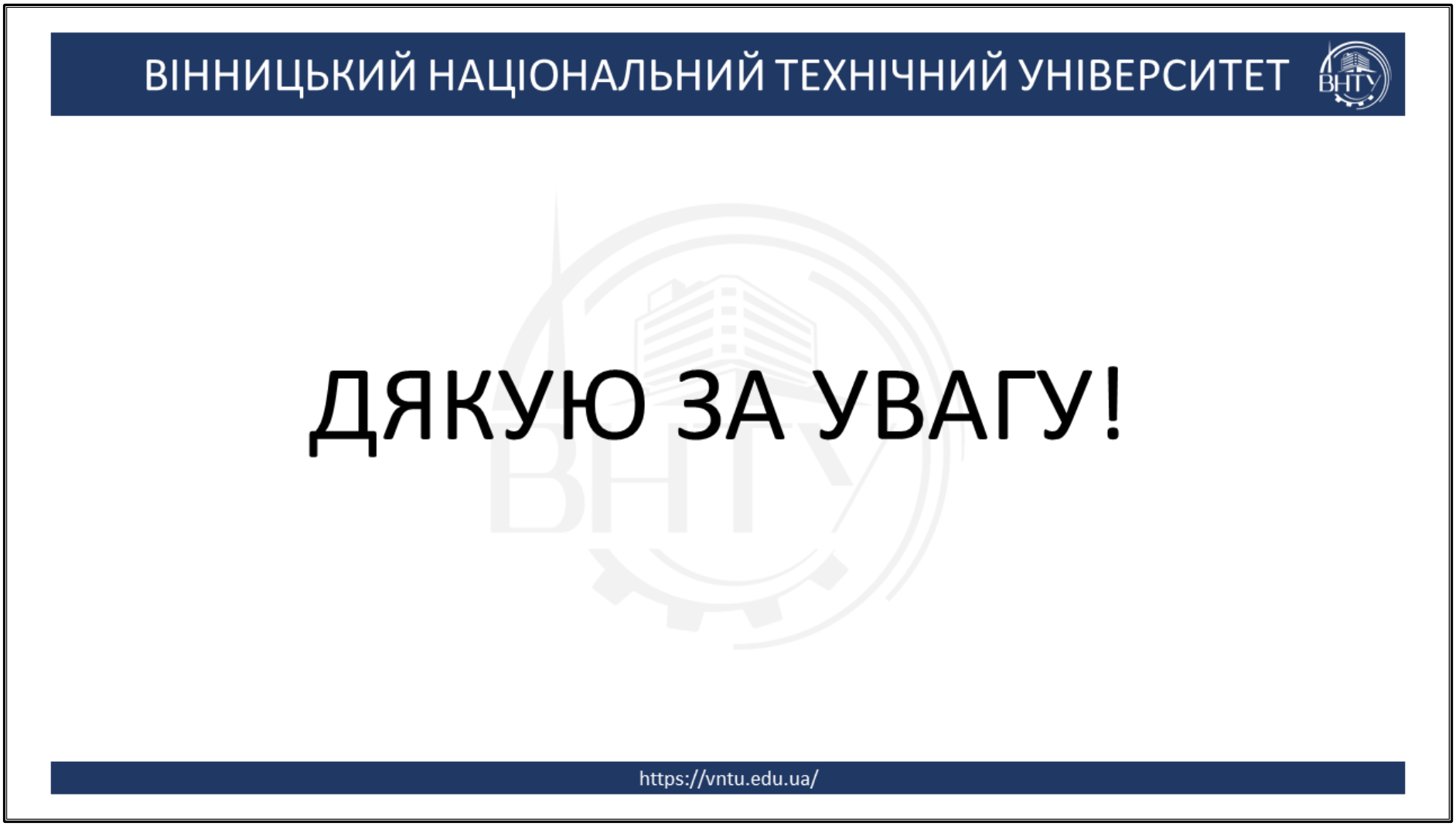


Рис. Д.15 — Слайд №15

ДОДАТОК Е

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційний ресурс з налаштувань системних параметрів комп'ютера

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93,3% Схожість 6,7%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ковалик В.І.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кадук О.В.
(підпис) (прізвище, ініціали)