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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Інформаційна система моніторингу фізіологічного стану людини

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 2 курсу, групи 2КІ-21м
спеціальності 123 — Комп'ютерна інженерія
Смачило А.Р.
Керівник к.т.н., доц. каф. ОТ
Тарновський М.Г.
"21" 12 2022 р.
Опонент к.т.н., доц. каф. ЗІ
Войтко В.В.
"22" 12 2022 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.
"23" 12 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров
"15" 09 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Смачилу Андрію Романовичу

1 Тема роботи «Інформаційна система моніторингу фізіологічного стану людини» керівник роботи Тарновський Микола Генадійович к.т.н., доцент, затверджено наказом вищого навчального закладу від 15.09.2022 року №205-А

2 Строк подання студентом роботи 23.12.2022 р.

3 Вихідні дані до роботи: призначення — контроль енергетичного балансу та фізичної активності; реалізація — мобільний додаток; операційна система — Android.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз інформаційних системи моніторингу фізіологічного стану людини та готових рішень, дослідження методів вимірювання фізіологічних показників, розробка інформаційної системи моніторингу фізіологічного стану та тестування одержаних результатів, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, схема компонентів інформаційної системи, лістинг основного програмного коду та коду xml ресурсів.

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	Тарновський Микола Геннадійович к.т.н., доцент		
5	Небава Микола Іванович к.е.н., доцент		

7 Дата видачі завдання 06.09.2022 р.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	24.09	вик.
2	Огляд представлених на ринку аналогів	25.09-07.10	вик.
3	Аналіз методів вимірювання енерговитрат людини	08.10-22.10	вик.
4	Розробка інформаційної системи	23.10-30.10	вик.
5	Створення мобільного застосунку	01.11-20.11	вик.
6	Тестування функціонування системи	21.11-03.12	вик.
7	Розрахунок економічної частини	04-10.12	вик.
8	Оформлення пояснювальної записки та ілюстративного матеріалу	11-17.12	вик.
9	Виконання магістерської кваліфікаційної роботи	18.12	вик.
10	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	19.12	вик.
11	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	19-21.12	вик.
12	Перевірка «антиплагіат»	23.12	вик.
13	Попередній захист	24.11	вик.

Студент Смачило Андрій Романович

Керівник к.т.н., доц. Тарновський Микола Геннадійович

АНОТАЦІЯ

УДКК 004.4

Смачило А.Р. Інформаційна система моніторингу фізіологічного стану людини. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2022, 125 с.

Укр. мовою. Бібліогр.: рис. 17; табл. 11.

У роботі розглянуто принципи побудови інформаційної системи для моніторингу фізіологічного стану людини. В роботі проведений аналіз сучасних підходів до вирішення задачі, вибрано та проаналізовано аналоги, розглянуто модель розробки, проаналізовані відомі види операційних систем, мов програмування та програмного середовища для розробки інформаційної системи та проведене тестування отриманих результатів. Розроблені алгоритми виконання поставлених задач для вирішення проблем. Розроблено схему компонентів інформаційної системи.

Ключові слова: інформаційна система, здоров'я, стан людини, операційні системи, Android, Java.

ANNOTATION

A. R. Smachylo. Information system for monitoring the physiological state of a person. Master's thesis on specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022.

Ukraine language. Bibliography: fig. 17, tab. 11.

The work considers the principles of building an information system for monitoring the physiological state of a person. The paper analyzes modern approaches to solving the problem, selects and analyzes analogs, considers the development model, analyzes known types of operating systems, programming languages, and software environments for information system development, and tests the obtained results. Algorithms have been developed to solve problems. A diagram of the components of the information system has been developed.

Keywords: information system, health, human condition, operating systems, Android, Java.

ЗМІСТ

ВСТУП

Однією з найбільш затребуваних по всьому світу областю є сфера розробки різних інформаційних систем. Щорічно збільшується число додатків, створених безпосередньо для різних платформ. Ринок інформаційних систем неухильно зростає. Наявність програми для смартфонів та комп'ютерів значно підвищує конкурентоспроможність тієї чи іншої сучасної організації на ринку через поширеність використання гаджетів. Досвід використання на практиці показує, що якісні додатки значно зручніше мобільної версії сайту компанії.

Актуальність теми дослідження полягає в тому, що із поширенням небезпечного вірусу COVID-19 та інших небезпечних хвороб люди почали дельніше вивчати питання охорони власного здоров'я. Через дані події виріс попит на різні системи для моніторингу здоров'я, додатки для відстежування прогресу в різних видах та все, що пов'язане із зміцненням імунітету. З кожним днем вчені відкривають нові методи запобігання різних хвороб. Для ефективного лікування недугів потрібно знати детальну інформацію про пацієнта та його фізичний або фізіологічний стан.

Об'єктом дослідження є процес моніторингу фізіологічного стану людини.

Предметом дослідження є методи контролю енергетичного балансу організму людини.

Метою роботи є розширення переліку вихідних даних для оцінювання фізичної активності та енергетичних витрат, що надає можливість більш ефективно підтримувати енергетичний баланс, запобігаючи надлишковій вазі та пов′язаним з нею захворюваннями.

Для досягнення цієї мети необхідно вирішити такі задачі:

— проаналізувати актуальні методи моніторингу стану людини та готові рішення, що представлені на ринку;

— на основі проведених досліджень розробити зручну та інтуїтивну інформаційну систему на доступній для будь-якого користувача операційні системі для моніторингу фізіологічного стану люди.

Для досягнення поставленої в роботі мети використовуються такі методи дослідження:

— системний аналіз;

— методи вимірювання фізичної активності;

— об'єктно-орієнтовані методи проектування.

Наукова новизна полягає в тому, що дістала подальшого розвитку система самозвіту для обчислення енергетичних витрат організму, в якій на відміну від уснуючої використовується більш гнучка система вихідних параметрів, що надає можливість враховувати індивідуальні особливості при контролі фізичної активності.

Практичне значення роботи полягає в тому, що розроблено простий та зручний у використанні мобільний додаток для операційної системи Android для оцінювання фізичної активності та здійснення контролю енергетичнго балансу організма людини.

1 ОГЛЯД І АНАЛІЗ ДЖЕРЕЛ ІНФОРМАЦІЇ

1.1 Інформаційні технології у медецині

Медицина завжди намагалася виміряти різні показники пацієнтів (наприклад, клінічну картину), щоб детально вивчити їх здоров'я та хвороби. Потім інформатика включатиме результати цих вимірювань, експериментів і спостережень у моделі. Останнім часом використання IT-інструментів для медицини (таких як машинне навчання) зазнало прискореного зростання, однак усі моделі такої системи є неповними без реальних даних, особливо на основі реєстрів. Наприклад, дані електронної медичної картки/документації пацієнта (EHR/D) можуть бути надані з достатньою точністю щодо клінічної картини, процедур або супутніх захворювань. Міжнародний стандарт — HL7 (Health Level Seven) забезпечує взаємодію між IT-провайдерами та рішеннями. Однак у багатьох країнах більшість медичних даних є аналоговими і їх необхідно оцифрувати (наприклад, обробка зображення та звуку). Крім того, потужні комп’ютерні засоби та переносні пристрої з кількома датчиками зробили можливим збір, обробку та зберігання даних про окремих пацієнтів або групу пацієнтів. Індивідуальні дані можуть бути використані для оцінки ризику певного розладу здоров'я. Наприклад, ризик захворіти інфекційним захворюванням можна розрахувати на основі даних датчиків і анкет. Дані, зібрані від групи потенційно контактних пацієнтів, можна використовувати для оцінки ризику інфекційних захворювань у певній громаді. Існує занепокоєння щодо Закону про захист даних в ЄС з юридичної точки зору щодо надання такого роду аналізу даних. Потрібно запитувати згоду кожного пацієнта, що може бути дуже важко отримати [1].

Основним застосуванням Blockchain в медицині є відстеження ліків у багатьох країнах. Така методика вже застосовувалася в Африці та Південно-Східній Азії для перевірки оригінальності ліків (підроблені ліки там є великою проблемою). Технологія Blockchain також може допомогти у розслідуванні спалахів, пов’язаних із їжею, де обробка товарів у ланцюжку поставок із характеристиками зберігання або транспортування має вирішальне значення [2].

Об’єднання даних (ОБ) — це багатодоменна галузь, що розвивається, і спрямована на надання даних для розуміння ситуації. У глобальному масштабі виявлення загроз і захист об’єктів є одними з життєвоважливих сфер досліджень ОБ. Системи Fusion мають на меті інтегрувати дані та інформацію датчиків у бази даних, бази знань, контекстну інформацію, завдання користувача тощо, щоб описати ситуації, що динамічно змінюються. У певному сенсі мета злиття інформації полягає в тому, щоб досягти моделювання в реальному часі підмножини світу на основі часткових спостережень за ним.

Здатність об’єднати цифрові дані в корисну інформацію ускладнюється тим фактом, що вхідні дані, отримані з пристрою чи текстові, генеруються в різних форматах, деякі з них неструктуровані. Незалежно від того, чи є інформація неструктурованою за своєю природою, чи відсутні стандарти обміну інформацією (метадані або вони не дотримуються), усе це перешкоджає автоматичній обробці комп’ютерами. Крім того, дані, які потрібно об’єднати, можуть бути неточними, неповними, неоднозначними або суперечливими; воно може бути фальшивим або зіпсованим ворожими заходами. Крім того, велику кількість інформації може бути важко формалізувати, тобто інформацію про зображення. Отже, обмін інформацією часто надто складний. У багатьох випадках існує брак зв’язку між різними джерелами інформації просто тому, що немає механізму для підтримки цього обміну [3].

Великий потік даних, як на пристрої, так і на основі тексту, не в змозі обробити і призводить до затримок у часі, додаткових витрат і навіть до невідповідних рішень через відсутність або неповну інформацію.

Ключовим аспектом у сучасних додатках ОБ є відповідна інтеграція всіх типів інформації або знань: даних спостережень, моделей знань (апріорних або індуктивно-засвоєних) і контекстної інформації. Кожна з цих категорій має відмінну природу та потенційну підтримку результату процесу синтезу.

Дані спостережень — це фундаментальні дані про людину, зібрані за допомогою певних засобів спостереження (датчиків будь-якого типу). Ці дані стосуються спостережуваних характеристик людини, які представляють інтерес.

Контекст і елементи того, що можна назвати контекстною інформацією, можна визначити як «сукупність обставин, що оточують отримані дані, які потенційно мають значення для їх завершення». Через релевантність даних термоядерний синтез передбачає розробку найкращої можливої оцінки з урахуванням цих сторонніх знань. Ми можемо розглядати контекст як фон, тобто не конкретну сутність, подію чи поведінку, що становить головний інтерес, а ту інформацію, яка впливає на формування найкращої оцінки цих елементів.

Набуті знання: у тих випадках, коли апріорні знання для розробки процесу ОБ не можуть бути сформовані, однією з можливостей є спроба вирізати знання за допомогою онлайн-процесів машинного навчання, що працюють на даних спостережень та інших даних. Це процедурні та алгоритмічні методи для виявлення взаємозв'язків між поняттями, що представляють інтерес, які фіксуються системою (дані датчиків і контекстна інформація). Існує компроміс, пов'язаний із спробою розробки повністю автоматизованих алгоритмічних процесів ОБ для складних проблем, де включення людського інтелекту в певний момент процесу може бути набагато більш розумним вибором.

На даний момент існує безліч проблемно-специфічних рішень термоядерного синтезу, більшість з яких зосереджені на конкретних програмах, що дає специфічні для проблеми результати. Можливість автоматичного об'єднання результатів менших проблем у більший контекст досі відсутня.

Електронні пристрої з низькою потужністю та недорогою ціною можна легко придбати та використовувати. Деякі властивості, такі як температура тіла та пульс, можна виміряти з високою точністю, але розробники намагаються націлюватися на все більше функцій. FDA (Федеральне агентство з лікарських засобів), а також EMA (Європейське агентство з лікарських засобів) сертифікують цифрове програмне забезпечення та пристрої для охорони здоров'я. Більшість пристроїв на ринку не задовольняють мінімальним умовам сертифікації та акредитації. Деякі з них як пристрої для моніторингу якості повітря (наприклад, лазерний датчик пилу PM2.5 менш ніж за 5 доларів) масово використовуються в сильно забруднених містах. Однак жодна медична організація не рекомендує використовувати домашні

пристрої через дуже низьку якість даних, і даних, наданих сертифікованими офіційними станціями, має бути достатньо. Більше даних не завжди означає краще, як це показано в цьому прикладі.

Деякі проблеми виникають із додатками та програмним забезпеченням для цифрового здоров'я (mhealth). Сьогодні майже для всього можна знайти програму. Розглянемо прогрес у розпізнаванні зображень за допомогою глибокого навчання смартфоном для вимірювання артеріального тиску за допомогою фотоплетизмографії. Найбільшою перевагою цього методу є простота, тому нічого крім смартфона не потрібно. Однак навіть після калібрування він працює погано для більшості користувачів, є пацієнти, для яких похибка цієї процедури може досягати 30%. Тому програми, що пропонують цю функцію (як-от icare і HDCH), було вилучено з Google Play і App Store, але їхні еквіваленти все ще дуже популярні, наприклад, у Китаї.

Ще одним типом датчиків є Bluetooth-маяки для відстеження соціальних взаємодій, які можна використовувати для контролю інфекційних захворювань і психологічного здоров'я (один із сертифікованих постачальників цих маяків — kontakt.io). Датчики смартфонів або переносних пристроїв можуть обмінюватися даними між собою та збирати контактні дані.

Аналіз і виявлення проблеми зі здоров'ям можна виконувати на мобільному пристрої (телефоні, годиннику, браслеті тощо), використовуючи принципи периферійних обчислень, або в сусідній обчислювальній інфраструктурі.

Об'єднання, обробка та аналітика даних датчиків, зібраних з персональних мобільних пристроїв і пристроїв охорони здоров'я, а також інфраструктури розумного міста виконуються на обчислювальних компонентах Edge/Fog, що забезпечує ефективність і мінімальну затримку при виявленні критичних медичних станів, які вимагають оперативних дій. Крім того, це може забезпечити персоналізовану систему охорони здоров'я для загального благополуччя, де люди можуть отримати медичне обслуговування, адаптоване до їхніх потреб [4].

Крім того, така система повинна підтримувати ефективний збір і аналіз величезної кількості неоднорідних і безперервних даних (Big Data) про здоров'я та

діяльність від групи або натовпу користувачів. Зберігання, агрегація, обробка та аналіз великих даних про здоров'я можуть виконуватися в публічній, приватній або гібридній хмарній інфраструктурі. Результати аналізу та видобутку даних надаються лікарям, медичним працівникам, медичним організаціям, фармацевтичним компаніям за допомогою налаштованої візуальної аналітики та додатків на інформаційній панелі. Перед ширшим застосуванням таких персоналізованих систем охорони здоров'я, які безперервно контролюють здоров'я та діяльність людей і належним чином реагують на критичні події та умови, необхідно вирішити низку проблем у сфері досліджень і розробок.

Загальна архітектура великомасштабної розподіленої системи на основі медичних пристроїв IoT та обробки та аналітики Big Data для дистанційного моніторингу здоров'я та діяльності людей наведена на рис. 1.1.

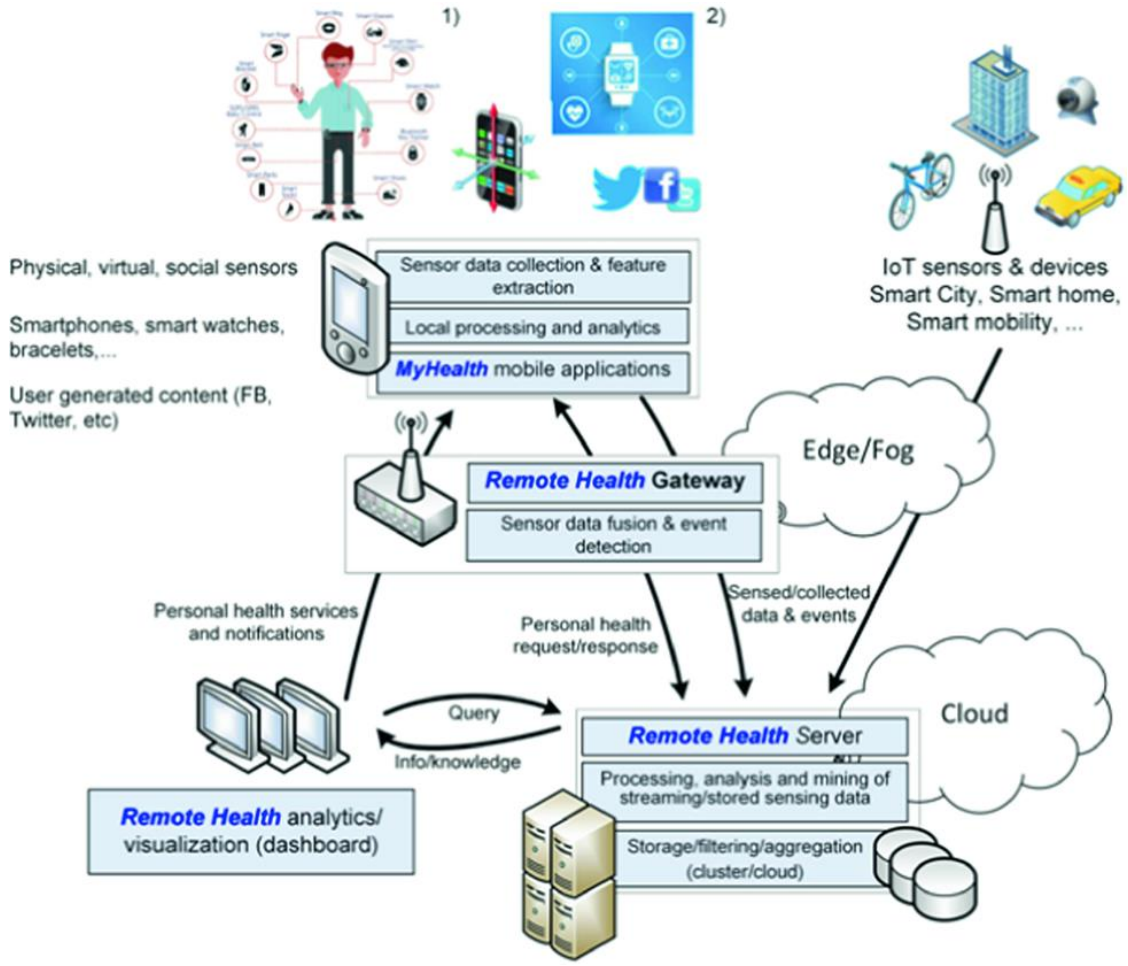

Рисунок 1.1 — Загальна архітектура системи дистанційного моніторингу стану здоров'я та моніторингу людей

Системи дистанційного моніторингу можуть працювати в різних масштабах, забезпечуючи персональне та глобальне зондування. Існує три різні шкали для зондування: особисте, групове та спільнотне.

Персональні або індивідуальні системи моніторингу розроблені для окремої особи та часто зосереджені на зборі та аналізі персональних даних. Типові сценарії включають відстеження режиму фізичних вправ користувачів, вимірювання рівня активності або виявлення симптомів, пов'язаних із психологічними розладами. Хоча зібрані дані можуть бути призначені виключно для використання користувачем, обмін ними з медичним працівником є звичайним явищем. Популярними технологіями персонального моніторингу є Google Glass, FitBit і The Nike+ FuelBand [5].

Якщо окремі особи поділяють спільні інтереси, занепокоєння чи цілі під час участі в програмах моніторингу, вони утворюють групу. Системи групового моніторингу можуть бути популярними в соціальних мережах або підключених групах, де дані можуть вільно ділитися або бути із захистом конфіденційності. Типовими прикладами є додатки для моніторингу здоров'я, включаючи змагання, пов'язані з конкретними цілями: бігова дистанція, втрата ваги, споживання калорій тощо.

Якщо кількість людей, які беруть участь у моніторингу здоров'я та активності, є дуже великою, це називається моніторингом спільноти або натовпу. Моделювання та моніторинг натовпу передбачає колективну аналітику даних на благо спільноти. Однак це передбачає співпрацю осіб, які не довірятимуть один одному, підкреслюючи необхідність надійного захисту конфіденційності та, можливо, низький рівень зобов'язань з боку користувачів. Приклади моніторингу спільноти включають відстеження поширення захворювань у регіоні, пошук закономірностей для конкретних захворювань тощо. Вплив масштабування на програми моніторингу ще буде досліджено, але багато дослідницьких проблем пов'язані з обміном інформацією, правом власності на дані, об'єднанням даних, безпекою та конфіденційність, алгоритми, які використовуються для

інтелектуального аналізу даних, надання корисних відгуків, залишаються відкритими.

1.2 Безпека медичних даних

Усі організації, які збирають, обробляють і зберігають медичні дані, зобов'язані удосконалювати свої стратегії захисту даних відповідно до норм і стандартів міжнародного та місцевого законодавства. Прикладом законів про захист даних, які вимагають США, є Закон про перенесення та підзвітність медичного страхування (HIPAA), примітка 1, яка використовується для відповідності та безпечного впровадження електронних медичних записів (EHR).

HIPAA встановлює правила захисту конфіденційних даних пацієнтів у своїх Правилах конфіденційності та безпеки. Згідно з HIPAA, компанія, яка обробляє захищену інформацію про здоров'я, повинна зберігати її в безпечній системі, що стосується фізичної безпеки, безпеки мережі та процесу. Правило безпеки HIPAA визначає вимоги до медичної інформації, яка зберігається або передається в електронній формі. Воно вимагає, щоб організації, на які поширюється дія HIPAA, запроваджували такі засоби захисту:

— забезпечення конфіденційності, цілісності та доступності усіх електронних даних, які вони створюють, отримують, зберігають або передають;

— визначення та захист від очікуваних загроз безпеці або цілісності інформації;

— захист від очікуваного, недозволеного використання або розголошення;

— забезпечення дотримання вимог їхньою робочою силою.

Крім того, воно регулює технічні заходи безпеки під час розміщення конфіденційних даних пацієнтів, включаючи доступ до закладу на місці, а також політику щодо використання робочих станцій, електронних носіїв і всіх процесів передачі, видалення, утилізації та повторного використання електронних носіїв або електронної інформації. Воно забезпечує дотримання авторизованих унікальних ідентифікаторів користувачів, процедур екстреного доступу, автоматичного

виходу з системи, шифрування даних і звітів про аудит. Крім того, необхідні журнали відстеження всіх дій на апаратному та програмному забезпеченні [6].

Організації охорони здоров’я повинні забезпечити безпеку та доступність даних пацієнтів як для медичних працівників, так і для пацієнтів.

Примітка 2 до Загального регламенту захисту даних (GDPR) — це положення законодавства ЄС щодо захисту даних і конфіденційності в межах Європейського Союзу (ЄС) і Європейської економічної зони (ЄЕЗ). Воно також визначає умови експорту персональних даних за межі ЄС та ЄЕЗ. У ньому зазначено:

— права суб'єкта даних;

— роль контролера;

— передача персональних даних третім країнам або міжнародним організаціям;

— незалежні наглядові органи;

— співпраця та послідовність;

— засоби правового захисту, відповідальність і штрафи;

— делеговані акти та імплементаційні акти.

Воно визначає дії, які необхідно вжити для забезпечення обробки в системах, які обробляють медичні дані. Кілька прикладів можуть бути:

— у разі порушення власники даних повинні повідомити органи влади;

— суб'єкти мають право на доступ до своїх даних;

— право бути забутим, яке дає людям право вимагати видалення їхніх персональних даних;

— конфіденційність включено за замовчуванням;

Право на виправлення даних має право отримати від контролера без невиправданої затримки. Суб’єкт даних має право на доповнення неповних персональних даних, у тому числі шляхом надання додаткової заяви.

Кожна організація, яка володіє персональними даними третіх сторін, повинна мати уповноваженого із захисту даних.

GDPR накладає суворі штрафи на контролерів і обробників даних за недотримання вимог у розмірі до 20 мільйонів, або 4% світового річного доходу за

попередній фінансовий рік, у разі порушення основних принципів обробки медичних даних.

Щоб отримати необхідний рівень безпеки HIPAA, компанія може піти двома шляхами:

— вибрати авторитетну навчальну компанію HIPAA, яка пропонує сертифікати сертифікації, і самостійно проаналізувати процедури та системи компанії;

— отримати оцінку від незалежного стороннього аудитора, наприклад, Coalfire SystemsFootnote3 або ComplySmartFootnote4.

Для забезпечення стандартів GRPD компанія може:

— отримати індивідуальний сертифікат GDPR для офіцерів із захисту;

— найняти уповноважених експертів.

Щодо відповідності вимогам HIPAA та GDPR, немає жодної центральної незалежної сторони, яка може сертифікувати компанію.

Особливі проблеми GDPR виникають у віддалених системах. По-перше, відповідно до GDPR персональні дані не можуть зберігатися довше, ніж це необхідно. Ось чому необхідно запровадити процедури зберігання, а коли термін дії даних закінчився, їх потрібно видалити із систем. Дані можуть зберігатися в кількох місцях, у кількох постачальників, оброблятися багатьма службами. Повне видалення даних також має враховувати резервні копії та копії даних, що зберігаються на віддаленому обладнанні. Крім того, проблемою може бути реакція на порушення. Повідомлення про порушення мають надаватися власникам даних. Подія порушення має бути чітко визначена.

Обробка персональних даних за межами Європейської економічної зони (ЄЕЗ) є наступною проблемою через кілька місць. Контролерам потрібно буде визначити стратегію даних для кількох країн, враховуючи закони про локалізацію.

Прозорість процедур безпеки медичних систем або сертифікати третьої сторони необхідні для того, щоб переконати контролерів в якості послуг безпеки.

Постачальники медичних послуг повинні підлягати аудиту, щоб забезпечити систему контролю з конфіденційністю та конфіденційністю за проектом.

Слід контролювати медичні послуги, щоб урахувати будь-які зміни в технології та рекомендовані оновлення системи. Вони включають нове обладнання та датчики.

Під час обробки великого набору даних видимість щодо метаданих та анонімізація даних є великою проблемою. Необхідно ретельно перевірити рівень захисту метаданих, відповідні права власності, права на обробку колекцій метаданих, цільове використання метаданих.

1.3 Поради щодо ведення здорового способу життя

МОЗ визначає фізичну активність як довільний рух тіла, що виробляється скелетними м’язами, що потребує витрат енергії. Фізична активність стосується будь-якого руху, у навіть під час дозвілля, під час поїздки транспортом, щоб дістатися до місця та назад, або як частина роботи людини. Фізична активність як середньої, так і високої інтенсивності покращує здоров’я [7].

Популярні способи бути фізично активними включають ходьбу, їзду на велосипеді, катання на колесах, спорт, активний відпочинок на природі та ігри, і це можна робити на будь-якому рівні навичок.

Науково доведено, що періодична фізична активність допомагає запобігти та лікувати неінфекційні захворювання, такі як діабет, інсульт, хвороби серця та деякі види раку. Це також допомагає підтримувати здорову вагу, запобігти гіпертонії та може покращити психічне здоров’я, якість життя та самопочуття.

Скільки фізичної активності рекомендується? Інструкції та рекомендації МОЗ надають деталі для різних вікових груп і окремих груп населення про те, скільки фізичної активності необхідно для гарного здоров’я.

МОЗ рекомендує:

Протягом 24-годинного дня немовлята (до 1 року) повинні:

— бути фізично активними кілька разів на день різними способами, зокрема через інтерактивні ігри на підлозі, більше — краще, а для тих, хто ще не рухається, це включає принаймні 30 хвилин у положенні лежачи (час на животі), розподілених протягом дня під час неспання;

— не бути прив’язаним довше 1 години за один раз (наприклад, колясками, високими стільчиками або прив’язаними на спині вихователя);

— не рекомендується проводити час перед екраном;

— під час малорухливого життя заохочується читання та розповідання історій з вихователем і мати від 14 до 17 годин (від 0 до 3 місяців) або від 12 до 16 годин (від 4 до 11 місяців) якісного сну, включаючи дрімоту.

У 24-годинний день діти від 1 до 2 років повинні:

— приділяти щонайменше 180 хвилин різноманітним видам фізичної активності різної інтенсивності, також включаючи фізичну активність середньої та високої інтенсивності, розподілену протягом дня, більше — краще;

— не бути прив’язаним довше 1 години (наприклад, колясками, високими стільчиками або прив’язаними на спині вихователя) або сидіти протягом тривалого часу;

— для однорічних дітей не рекомендується сидячий час перед екраном (наприклад, перегляд телевізора чи відео, комп’ютерні ігри);

— для дітей віком від 2 років сидячий час перед екраном має становити не більше 1 години, менше — краще;

— під час малорухливого життя заохочується читання та розповідання історій з вихователем;

— мати від 11 до 14 годин якісного сну, включаючи дрімоту, з регулярним часом сну та пробудження.

У 24-годинний день діти від 3 до 4 років повинні:

— приділяти щонайменше 180 хвилин різноманітним видам фізичної активності будь-якої інтенсивності, з яких щонайменше 60 хвилин — це фізична активність від середньої до високої інтенсивності, розподілена протягом дня, більше — краще;

— не бути прив’язаним довше 1 години (наприклад, у колясках) і не сидіти протягом тривалого часу.

— сидячий час перед екраном не повинен перевищувати 1 години;

— коли сидячий спосіб життя, залучення до читання та розповідання історій з вихователем є;

— мати від 10 до 13 годин якісного сну, який може включати дрімоту, з регулярним часом сну та пробудження.

Діти та підлітки від 5 до 17 років:

— повинні виконувати принаймні в середньому 60 хвилин на день фізичної активності середньої та високої інтенсивності, переважно аеробної, протягом тижня;

— мають включати аеробні навантаження інтенсивного навантаження, а також ті, що зміцнюють м’язи та кістки, принаймні 3 дні на тиждень;

— слід обмежити кількість часу, проведеного в сидячому положенні, особливо кількість часу, проведеного перед екраном для відпочинку.

Дорослі віком віком від 18 до 64 роки:

— займатися аеробною фізичною активністю помірної інтенсивності не менше ніж від 150 до 300 хвилин або щонайменше від 75 до 150 хвилин аеробної фізичної активності високої інтенсивності або еквівалентну комбінацію активності середньої та високої інтенсивності протягом тижня;

— також слід виконувати вправи для зміцнення м’язів середньої або більшої інтенсивності, які залучають усі основні групи м’язів 2 або більше днів на тиждень, оскільки це забезпечує додаткові переваги для здоров’я;

— можна збільшити аеробну фізичну активність середньої інтенсивності до понад 300 хвилин або виконувати більше 150 хвилин аеробної фізичної активності високої інтенсивності або еквівалентну комбінацію активності середньої інтенсивності протягом тижня для додаткової користі для здоров’я;

— слід обмежити кількість часу, проведеного в сидячому положенні, адже заміна сидячого часу фізичною активністю будь-якої інтенсивності (включаючи легку інтенсивність) приносить користь для здоров’я;

— щоб допомогти зменшити шкідливий вплив сидячого способу життя на здоров’я, усі дорослі та люди похилого віку повинні прагнути виконувати фізичну

активність від середньої до високої інтенсивності, що перевищує рекомендовані рівні.

Дорослі віком від 65 років — так само, як і для дорослих. Люди похилого віку повинні виконувати різноманітну багатокомпонентну фізичну активність, яка наголошує на функціональному балансі та силових тренуваннях середньої або більшої інтенсивності, 3 або більше днів на тиждень, щоб підвищити функціональну здатність і запобігти падінням.

Всі вагітні та породіллі без протипоказань повинні:

— виконувати принаймні 150 хвилин аеробної фізичної активності середньої інтенсивності протягом тижня;

— включати різноманітні аеробні вправи та вправи для зміцнення м’язів;

— слід обмежити кількість часу, проведеного в сидячому положенні, адже заміна сидячого часу фізичною активністю будь-якої інтенсивності (включаючи легку інтенсивність) приносить користь здоров’ю.

Країни та громади повинні вжити заходів, щоб надати кожному більше можливостей бути активними, щоб збільшити фізичну активність. Це вимагає колективних зусиль, як на національному, так і на місцевому рівнях, у різних секторах і дисциплінах для впровадження політики та рішень, які відповідають культурному та соціальному середовищу країни, щоб просувати, уможливлювати та заохочувати фізичну активність.

Політика збільшення фізичної активності спрямована на те, щоб:

— ходьба, їзда на велосипеді та інші види активного немоторизованого транспорту були доступними та безпечними для всіх;

— політика щодо праці та робочого місця заохочувала активні поїздки на роботу та з роботи та можливості бути фізично активними протягом робочого дня;

— догляд за дітьми, школою та вищими навчальними закладами забезпечували сприятливі та безпечні місця та умови для всіх студентів, щоб вони могли активно проводити свій вільний час;

— початкова та середня школи забезпечували якісне фізичне виховання, яке підтримуватиме дітей у формуванні моделей поведінки, які підтримуватимуть їхню фізичну активність протягом усього життя;

— громадські та шкільні спортивні програми надавали належні можливості для будь-якого віку та здібностей;

— спортивні та рекреаційні заклади надавали можливість кожному отримати доступ і брати участь у різноманітних видах спорту, танцях, фізичних вправах та активному відпочинку;

— постачальники медичних послуг радили і підтримували пацієнтів бути регулярними активними.

1.4 Огляд та аналіз відомих аналогів

Проаналізуємо платформи та мобільні додатки для моніторингу фізіологічного стану людини.

Thinger.io — це платформа IoT, яка є альтернативою з відкритим кодом. Thinger.io є відносно новим для екосистеми IoT, але широко використовується в різних дослідницьких проектах або навіть для освіти. Вона надає готовий до використання хмарний сервіс для підключення пристроїв до Інтернету для виконання будь-якого дистанційного зондування або активації через Інтернет (див. рис 1.2). Вона пропонує безкоштовний рівень для підключення обмеженої кількості пристроїв, але також можна встановити програмне забезпечення поза хмарою для приватного керування даними та пристроями, підключеними до платформи, без будь-яких обмежень [8].

Ця платформа не залежить від апаратного забезпечення, тому можна підключити будь-який пристрій із підключенням до Інтернету, від пристроїв Arduino, Raspberry Pi, пристроїв Sigfox, рішень Lora через шлюзи або пристроїв ARM. Платформа надає деякі стандартні функції, такі як реєстр пристроїв, двонаправлений зв'язок у реальному часі, як для зондування, так і для активації, зберігання даних і конфігурації, тому є можливість зберігати дані часових рядів, керування ідентифікацією та доступом (IAM), щоб дозволити стороннім особам

отримувати доступ до платформи та ресурсів пристрою через REST/Websocket API, Webhooks сторонніх розробників, щоб пристрої могли легко викликати інші веб-сервіси, надсилати електронні листи, SMS, надсилати дані в інші хмари тощо.

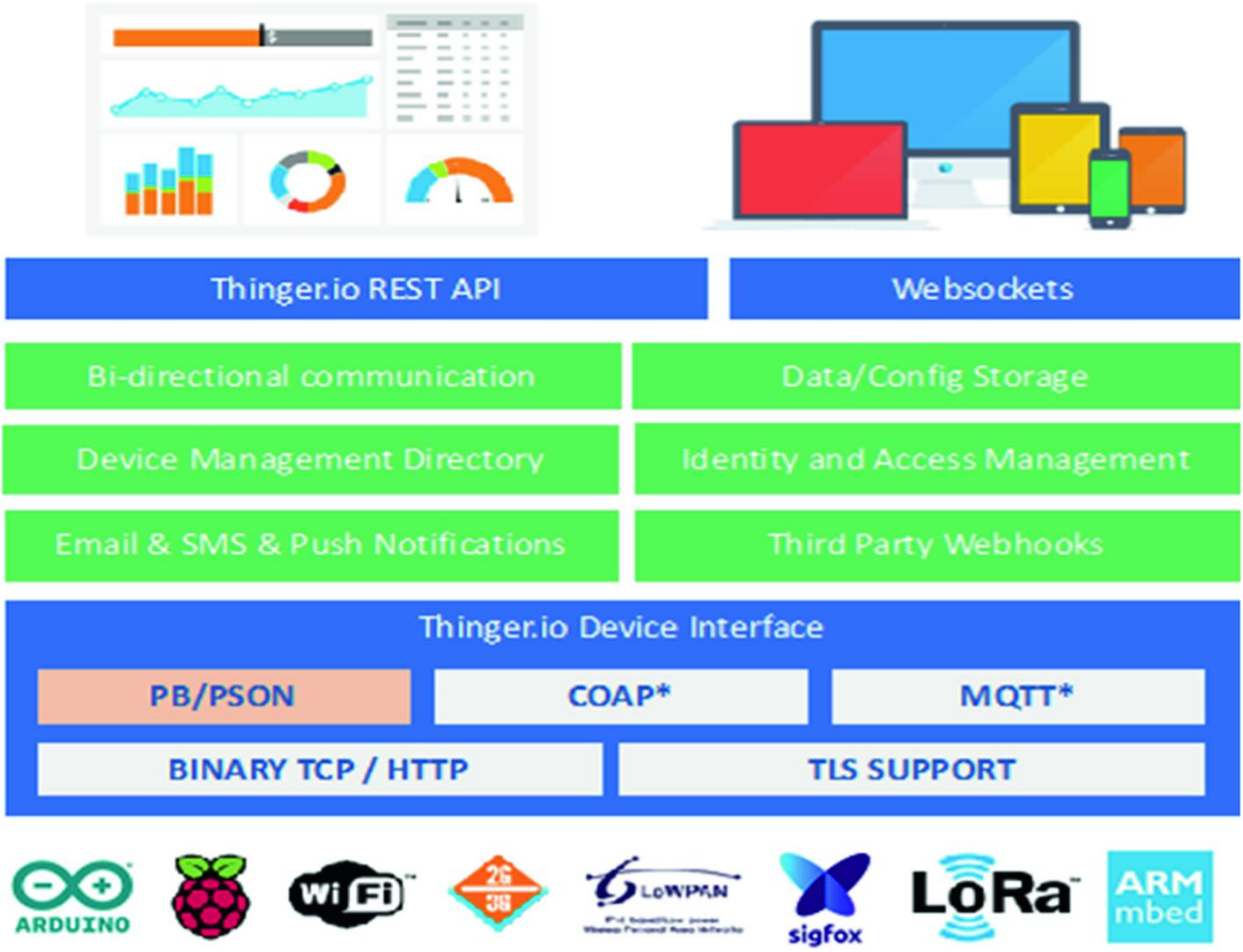

Рисунок 1.2 — Платформа Thinger.io

Вона також надає веб-інтерфейс для керування всіма ресурсами та створення інформаційних панелей для віддаленого моніторингу. Основною перевагою використання цієї платформи, окрім того, що вона є відкритим кодом, є можливість отримати двонаправлений зв’язок із пристроями в режимі реального часу за допомогою стандартних інтерфейсів REST-API. Таким чином, можна розробити будь-який додаток, наприклад, настільний, мобільний, веб-сервіс, який взаємодіє з пристроями за допомогою добре відомого та перевіреного інтерфейсу на основі REST. Тим часом пристрої можуть використовувати більш ефективні (з точки зору пропускної здатності або обсягу пам’яті) бінарні протоколи для зв’язку з хмарою. З іншого боку, ця платформа надає клієнтські бібліотеки для підключення кількох

найсучасніших пристроїв IoT, таких як Arduino, ESP8266, ESP32, LinkitOne, Texas Instruments CC3200, Libellium Waspmotes, Raspberry Pi тощо. Клієнтські бібліотеки забезпечують комплексний спосіб підключення пристроїв і надсилання інформації в хмару без необхідності мати справу зі складними протоколами IoT.

Libelium MySignals. Це платформа для розробки медичних пристроїв і додатків eHealth. Платформу можна використовувати для розробки веб-додатків eHealth і для тестування власних датчиків для медичних програм. MySignals є прикладом комерційного продукту, який пропонує та підтримує іспанська компанія LibeliumFootnote9. Вона дозволяє вимірювати понад 20 біометричних параметрів, таких як пульс, частота дихання, кисень у крові, сигнали електрокардіограми, артеріальний тиск, сигнали м'язової електроміографії, рівні глюкози, шкірно-гальванічна реакція, ємність легенів, хвилі хропіння, положення пацієнта, потік повітря та масштаб тіла. Дані, зібрані датчиками, можуть зберігатися в MySignals або сторонній хмарі для візуалізації у зовнішніх веб-додатках і мобільних додатках. Libelium пропонує розробникам API для доступу до інформації. Одним із недоліків є обмеження для надсилання інформації, що надходить із MySignals, на сторонній хмарний сервер за допомогою безпосереднього радіозв'язку WiFi, яке обмежено лише для апаратної версії — ця опція недоступна для версії програмного забезпечення, що працює на вузлі libelium Atmega 2560.

FIWARE. FIWARE — це IoT-платформа з відкритим вихідним кодом, яка «комбінує компоненти, які забезпечують підключення до IoT із службами Context Information Management і Big Data у хмарі». FIWARE використовує досить прості стандартні API для керування та обміну даними, які можна використовувати для розробки розумних програм. Вона надає розширені можливості хмарного хостингу на основі OpenStack і низку компонентів для функцій, які пропонуються як сервіс. Впровадженню технології FIWARE на ринку електронної охорони здоров'я сприяє проект промислового прискорювача FICHe, запущений Європейською Комісією, і кілька промислових проектів, які були розроблені в контексті FICHe. CardioWeel — це передова система допомоги водієві, яка надає інформацію про стан здоров'я

водія за допомогою електрокардіограми людини (ЕКГ), отриманої з рук водія. Ця система розміщена в хмарі FIWARE з використанням доступних віртуальних машин.

Runkeeper — це застосунок для відстеження різноманітних вправ.

Runkeeper відстежує ходьбу, біг та будь-яку іншу фізичну активність. Він підходить для людей будь-якого рівня фізичної підготовки, і компанія стверджує, що має спільноту з 50 мільйонів користувачів.

Людина може створювати, зберігати та відкривати нові маршрути. Додаток також використовує GPS для моніторингу темпу та пройденої відстані.

Інформаційна панель «Мій план» надає власні плани тренувань на основі відповідей користувача на серію запитань, або людина може вибрати з готових розкладів.

Strava — це мобільний застосунок для відстеження їзди на велосипеді та бігу.

Strava дозволяє користувачам відстежувати свою подорож на велосипеді та бігу за допомогою GPS. Людина також може приєднатися до викликів, спілкуватися з друзями та стежити за ними, а також ділитися фотографіями через додаток.

Програма також сумісна з рядом пристроїв GPS, зокрема:

— телефони;

— годинник;

— монітори пульсу.

Додаток є безкоштовним для використання, але люди можуть отримати доступ до багатьох додаткових функцій, якщо вони платять 7,99 доларів на місяць. До них входять обмін інформацією про місцезнаходження в реальному часі, планування маршруту, встановлення цілей, журнали тренувань і функція Heatmaps, що показує популярні маршрути в поточному місці розташування людини.

Glo — це додаток для понад 4000 завантажуваних класів.

Glo пропонує онлайн-заняття з йоги, медитації, пілатесу та фітнесу. Він призначений як для новачків, так і для більш досвідчених людей.

Користувач відповідає на кілька запитань, щоб персоналізувати свій досвід, і більше 4000 завантажуваних класів доступні за запитом. Викладачі приїжджають з усього світу.

Sworkit — це додаток, який призначений для різних рівнів досвіду.

Sworkit підходить для всіх, від початківців до досвідчених спортсменів. Людина може вирішити, чи хоче вона, щоб її тренування було зосереджено на силі, кардіо, йозі, пілатесі чи розтяжці. Вони також можуть вибирати з різноманітних щоденних або щомісячних завдань і ряду колекцій тренувань.

Додаток також дозволяє користувачам вибирати тренування на основі обладнання, фітнес-цілей і рівня впливу.

Люди можуть встановити індивідуальний 6-тижневий план, створювати тренування з нуля, відстежувати свою продуктивність і спілкуватися сам-на-сам з особистими тренерами. Також доступні нагадування, музика та розваги для дітей.

Диван до 5к — це мобільний додаток для запущеної програми.

Ця програма пропонує 9-тижневу програму бігу. У ньому стверджується, що додаток може навчити користувачів пробігати 5 кілометрів за цей час, дотримуючись цього плану тренувань.

Людина може вибрати одного з чотирьох різних тренерів і слухати свою улюблену музику, використовуючи додаток, який також може розраховувати відстань і темп за допомогою GPS-трекера. Користувачі можуть ділитися своїм прогресом у Facebook.

Додаток відображає результати темпу та дистанції у вигляді зручного для читання графіка, щоб людина могла чітко відстежувати свою продуктивність.

JEFIT — це мобільний додаток та журнал тренажерного залу відстеження тренувань.

JEFIT дозволяє користувачам відстежувати свої тренування та прогрес у важкій атлетиці. За допомогою цього додатка можлива персоналізація, оскільки людина має доступ до понад 1400 вправ з інструкціями.

Компанія стверджує, що додаток простий у використанні та вимагає лише одного кліка, щоб зареєструвати навчання. Час відпочинку також легко контролювати, а також доступні аудіо та відео інструкції.

Людина також може зосередитися на певних м’язах і отримати вказівки щодо відновлення м’язів.

Charity Miles — це програма, яка допомагає збирати гроші на добрі справи. За кожну милю фізичної активності, яку людина проходить, компанія жертвує гроші на благодійну організацію. Людина може вибрати з понад 50 благодійних організацій.

Компанія стверджує, що на сьогодні пожертвувала понад 2 мільйони доларів.

Люди можуть брати участь у будь-якому виді фізичної активності, включаючи ходьбу, танці, їзду на велосипеді та біг.

2 МЕТОДИ ВИМІРЮВАННЯ ФІЗІОЛОГІЧНИХ ПОКАЗНИКІВ ТА КОМПОНЕНТІВ РОЗРОБКИ

2.1 Методи вимірювання фізичної активності

Фізична активність визначається як будь-який рух тіла, вироблений скелетними м’язами, який призводить до витрати енергії. Переваги фізичної активності для підтримки здоров’я були добре задокументовані, особливо в профілактиці та лікуванні хронічних захворювань. Тому точне вимірювання фізичної активності та витрат енергії є важливим як для епідеміологічних досліджень, так і в клінічному контексті. Враховуючи велику кількість доступних методів, важливо мати розуміння кожного, особливо коли потрібно вибрати техніку для використання [9].

Переваги фізичної активності для підтримки здоров’я були добре задокументовані, особливо в профілактиці та лікуванні хронічних захворювань, таких як деякі види раку, діабет 2 типу та серцево-судинні захворювання. У цьому контексті точне вимірювання фізичної активності та енерговитрат є важливим як в епідеміологічних дослідженнях, так і в оцінці ефективності програм втручання. У клінічних умовах оцінка енергетичних витрат дозволяє оцінити потреби пацієнтів у харчових речовинах під час нутритивної підтримки.

Фізична активність визначається як будь-який рух тіла, вироблений скелетними м’язами, що призводить до витрат енергії. Важливо підкреслити, що фізична активність і енерговитрати — це 2 різні поняття. Простіше кажучи, фізична активність — це поведінка, яка призводить до підвищення витрат енергії над рівнем спокою. Загальні витрати енергії (TEE) відносяться до загальної кількості енергії, витраченої протягом 24-годинного періоду, і містять 3 основні компоненти: витрати енергії в стані спокою (REE), тепловий ефект їжі (TEF) і витрати енергії під час діяльності (AEE) [10].

Існують різні методи оцінки фізичної активності та витрат енергії, і кожен з них має переваги та обмеження, як підсумовано в табл. 2.1. Розуміння цих методів

є важливим, щоб вирішити, який метод використовувати для конкретного контексту дослідження.

Таблиця 2.1 — Методи оцінки фізичної активності та витрат енергії

Метод	Переваги	Недоліки
Вода двійного маркування	Високоточний метод, який вважається золотим стандартом для вимірювання загальних витрат енергії	Висока вартість методу (включаючи високу ціну і дороге обладнання для аналізу)
	Надає учасникам свободу діяльності	Експертиза необхідна для персоналу
		Метод не надає жодних конкретних деталей щодо фізичної активності
Пряма калориметрія	Це найточніший метод кількісного визначення швидкості метаболізму	Висока вартість методу
		Необхідне утримання суб’єкта протягом 24 годин або більше
Непряма калориметрія	Точний і неінвазивний метод	Відносно висока вартість
	Надає інформацію про метаболічне паливо, яке спалюється	Для правильного використання методу необхідний навчений персонал
	Дозволяє оцінити витрати енергії в польовому середовищі	
Акселерометрія	Об'єктивне вимірювання фізичної активності	Неточність прогностичних рівнянь для переведення підрахунків активності в витрати енергії, особливо при використанні в ряді різноманітних видів діяльності
	Може використовуватися як в лабораторних, так і в польових умовах	
	Неінвазивний метод і менш обтяжливий для пацієнтів	
	Відносно недорогий	
Монітор серцевого ритму	Об'єктивний інструмент для вимірювання фізичної активності та витрат енергії	Неточно оцінює сидячий спосіб життя та легку діяльність

Продовження таблиці 2.1

Крокометрія	Недорогий і неінвазивний метод	Обмежується лише вимірюванням активності при ходьбі
	Використовується для оцінки найпоширенішої діяльності (ходьба)	Неточно для оцінки пройденої відстані та витраченої енергії
	Може мотивувати людей підтримувати фізичну активність	
Методи самозвіту	Низька вартість, що дозволяє використовувати їх у дослідженнях із великим розміром вибірки	Низька точність і надійність, особливо пов'язана з їх залежністю від пам'яті учасника
	Низьке навантаження на суб'єктів	
	Надайте інформацію про моделі фізичної активності	

REE, найбільша частина TEE, є енергією, необхідною для підтримки основної метаболічної активності, включаючи підтримку температури тіла та підтримку функціонування життєвоважливих органів, таких як мозок, нирки, серце та легені. REE визначається як енергія, витрачена людиною, що голодує, у стані спокою в термонейтральному середовищі. Фактори, що найбільш суттєво впливають на REE, включають склад тіла, стать, температуру тіла, вік, обмеження енергії, а також генетичну та ендокринну систему. Склад тіла: знежирена маса (також звана нежировою масою тіла) є основним визначальним фактором REE, тобто людина з високою знежиреною масою має вищий REE. Стать: REE має

тенденцію бути вищим у чоловіків, ніж у жінок, і це може бути пов’язано з вищим відсотком нежирової маси тіла у чоловіків порівняно з жінками. Вік: люди похилого віку мають нижчий REE порівняно з молодими. Було досліджено, що пов’язане з віком зниження REE не залежить від змін у складі тіла, що свідчить про те, що також можуть бути залучені додаткові метаболічні зміни. Енергетичне обмеження: спроби схуднути шляхом обмеження споживання енергії призводять до зниження REE [11].

REE вимірюється, коли людина, що голодує, відпочиває в комфортних умовах. Тривалість голодування зазвичай становить від 2 до 4 годин. REE трохи вище (приблизно 10%), ніж базові витрати енергії (BEE), що є найнижчими енерговитратами людини, і вимірювання BEE вимагає більш суворих умов. BEE людини визначається, коли людина перебуває в постабсорбційному стані (тобто не приймає їжу принаймні 12 годин), лежить (на спині) і повністю розслаблена (нерухомий стан) — бажано дуже швидко після пробудження від сну в ранок.

AEE є найбільш варіабельним серед компонентів TEE, як на внутрішньо особистісному, так і на міжособистісному рівні. У людей, які ведуть сидячий спосіб життя, він може становити менше половини AEE, тоді як у дуже активних людей, таких як деякі спортсмени або важкі працівники, він може перевищувати AEE в 2 рази або більше.

TEF, також відомий як індукований дієтою термогенез (DIT) — це енергія, необхідна для перетравлення, всмоктування, транспортування та метаболізму їжі, зберігання поживних речовин і видалення відходів. Він являє собою збільшення витрат енергії вище REE, яке можна виміряти протягом кількох годин після їжі. TEF оцінюється приблизно як 10% добової TEE.

2.2 Методи обчислення витрат енергії

2.2.1 Метод води двійного маркування

Метод DLW використовує стабільні ізотопи кисню (^{18}O) і водню (^{2}H) для вимірювання TEE. Метод DLW широко визнаний золотим стандартом для вимірювання TEE і використовувався в різних дослідженнях для перевірки інших

методів. Окрім високої точності, метод DLW представляє перевагу своєї неінвазивної природи та можливості для суб'єктів продовжувати свою звичайну діяльність протягом періоду вимірювання. Метод також має обмежене навантаження на досліджуваних осіб. Однак обмеженням методу є його висока вартість через високу ціну DLW, дороге обладнання та досвід, необхідні для аналізу. Іншим обмеженням методики DLW є те, що вона забезпечує загальний показник середнього щоденного ТЕЕ протягом періоду вимірювання, але не надає жодних конкретних деталей щодо фізичної активності. В даний час цей метод використовується в широкому діапазоні категорій населення, включаючи немовлят, вагітних і годуючих жінок, людей похилого віку.

Метод заснований на наступному принципі: після того, як суб'єкт приймає дозу $^{2}H_2{}^{18}O$, відбувається врівноваження 2 ізотопів із загальною водою в організмі з подальшим їх виведенням з організму, яке відбувається з різною швидкістю. Дейтерій (^{2}H) втрачається з організму лише через воду (H_2O), тоді як ^{18}O втрачається як через воду, так і через вуглекислий газ (CO_2). Швидкість утворення CO_2 (rCO_2) розраховується як різниця між швидкостями елімінації ^{2}H і ^{18}O за такою формулою (2.1)

$$rCO2\ (\text{моль/день}) = 0{,}4554 \times TBW\ (\text{моль}) \times (1{,}007\ k_o - 1{,}041\ k_h) \qquad (2.1)$$

де k_o і k_h (день$^{-1}$) — швидкість елімінації ^{2}H і ^{18}O відповідно.

ТЕЕ розраховується за допомогою модифікованої версії формули Вейра на основі rCO_2 і харчового коефіцієнта (FQ) (2.2)

$$ТЕЕ\ (\text{ккал/день}) = 22{,}4 \times (3{,}9 \times [rCO_2/FQ] + 1{,}1 \times rCO2) \qquad (2.2)$$

Існує 2 основні протоколи для методу DLW: 2-точковий і багатоточковий підходи. 2-точковий протокол як мінімальна форма вимагає 3-х зразків, включаючи базову лінію до введення дози, зразок після введення дози, взятий у день введення дози після того, як ізотопи врівноважені в організмі, і остаточний зразок, взятий наприкінці дослідження (тобто на період від 10 до 14 днів). Багатоточковий

протокол як найбільш екстремальна форма, як правило, передбачає взяття базового зразка до введення дози та зразків щодня після прийому дози до кінця періоду відбору. На практиці 2 підходи були модифікаціями 2 підходів і вони досить схожі.

Що стосується 2-точкового протоколу, то найбільш часто використовуваною формою є модифікований підхід, за якого загалом збирають 5 зразків. Суб'єктам пропонується прийти до клінічного центру або центру збору зразків сечі вранці після нічного голодування, і протокол починається зі збору базового зразка сечі. Через деякий час після цього учасник випиває DLW, приготований на основі TBW суб'єкта. У деяких дослідженнях кількість ізотопної дози в DLW визначалася масою тіла учасника. Через годину після вживання DLW суб'єкти повинні спорожнити сечовий міхур, а час необхідно записати. Однак ця сеча не збирається, оскільки ізотопна рівновага з водою в організмі в цей час ще не встановлена. Через 3 та 4 години після вживання DLW слід зібрати ще 2 зразки сечі. Суб'єкт не повинен пити та їсти протягом 3 або 4 годин під час збору зразків сечі, щоб звести до мінімуму будь-який короткочасний вплив споживання води на збагачення сечі. В останній день експериментального періоду слід зібрати ще 2 зразки сечі приблизно в той самий час попереднього дня. Типові інтервали між початковим і остаточним збором сечі становлять 7, 10 або 14 днів. У багатоточковому підході після введення DLW збирається більше зразків сечі, ніж у 2-точковому протоколі. Протягом періоду дослідження всі зразки повинні бути зібрані в той самий час, що й у попередні дні [12].

Кількість зібраних зразків сечі не є критичним фактором щодо валідності методу DLW. Швидше, вибір частоти вибірки залежить від уподобання дослідника щодо точності методу. 2-точковий протокол представляє перевагу використання меншої кількості зразків і забезпечує точнішу оцінку TEE за умов, коли споживання енергії або водообіг змінюються щодня. З іншого боку, багатоточковий протокол має перевагу усереднення даних і, таким чином, мінімізує аналітичну помилку. Крім того, це дозволяє досліднику оцінити відмінності у витратах енергії для підперіодів у межах метаболічного періоду.

Після збору та зберігання зразків сечі проводять аналіз ізотопно-мас-спектрометричним методом.

Доза DLW базується на розмірі тіла суб'єкта, щоб узгодити збагачення води в організмі з точністю мас-спектрометрії за співвідношенням ізотопів. Враховуючи складність вимірювання TBW, його необхідно оцінити. У більшості досліджень DLW дослідники припускали, що TBW становить 60% маси тіла. Для збагачення міченої води, наявної на ринку, найчастіше використовуються 99 атомних % дейтерію (^{2}H) і 10 атомних % ^{18}O. Міжнародне агентство з атомної енергії (МАГАТЕ) рекомендує дози 0,12 г·кг$^{-1}$ води тіла з 99 атомними % води, поміченої дейтерієм, і 1,80 г·кг$^{-1}$ води тіла з 10 атомними % ^{18}O. При використанні більш збагаченої ^{18}O води дозу слід зменшити. Перед введенням DLW можна стерилізувати, пропускаючи його через фільтр 0,22 μm.

Після вимірювання методом DLW TEE можна використовувати для розрахунку AEE та PAL. Розрахунки включають РЗЕ, які вимірюються непрямою калориметрією або оцінюються за допомогою прогнозних рівнянь. Коли TEF приймається як 10% від TEE, AEE розраховується наступним чином (2.3)

$$\text{AEE (ккал/день)} = 0{,}9 \times \text{TEE (ккал/день)} - \text{REE (ккал/день)} \quad (2.3)$$

Для розрахунку PAL використовується таке рівняння (2.4)

$$\text{PAL} = \text{TEE/REE} \quad (2.4)$$

2.2.2 Пряма калориметрія

Техніка прямої калориметрії вимірює швидкість втрати тепла суб'єктом за допомогою калориметра. Це найточніший метод кількісної оцінки швидкості метаболізму, але його використання обмежене високою вартістю. Існує 4 типи прямих калориметрів, а саме «ізотермічні прямі калориметри» (також відомі як «калориметри теплового потоку або теплопровідності), які працюють, підтримуючи постійну температуру стінки за допомогою рідини постійної температури (зазвичай вода) в сорочці або ванні, що оточує камеру для тварин, або

в мережі мідних трубок, прикріплених до зовнішньої поверхні стіни; «прямі калориметри з радіатором», які складаються з камери, з якої тепло, втрачене суб'єктом, видаляється за допомогою теплообмінника з рідинним охолодженням, «калориметри прямої конвекції», які складаються з ізольованої камери, вентильованої потоком повітря з відомою швидкістю. Система працює шляхом визначення різниці температур і ентальпії між повітрям, що входить і виходить із ізольованої камери, і «прямі диференціальні калориметри», які включають 2 ідентичні камери, одна з яких містить суб'єкта, а інша має електричний нагрівач, налаштований на однакове підвищення температури в обох камерах.

2.2.3 Непряма калориметрія

Техніка непрямої калориметрії базується на вимірюванні об'єму вдихуваного і видихуваного газу, а також концентрації O_2 і CO_2. Для збору газу використовуються різні методи, включаючи мішок Дугласа, навіс і маску для обличчя. Непряма калориметрія є точним і неінвазивним методом, і він може дозволити оцінити витрати енергії на полі за допомогою амбулаторних метаболічних систем. Витрати енергії розраховуються за формулою Вейра, яка виглядає наступним чином (2.5)

$$\text{Витрати енергії (ккал)} = 3{,}941 \times VO_2\text{ (л)} + 1{,}106 \times VCO_2\text{ (л)} \quad (2.5)$$

де VO_2 — об'єм спожитого O_2;

VCO_2 — об'єм утвореного CO_2.

У разі розрахунку REE здебільшого використовується скорочена формула Вейра (2.6)

$$\text{REE (ккал/день)} = (3{,}941 \times VO_2\text{ [л/хв]} + 1{,}106 \times VCO_2 \quad (2.6)$$

де VO_2 — об'єм спожитого O_2;

VCO_2 — об'єм утвореного CO_2.

Непряма калориметрія є найбільш широко використовуваним методом оцінки енергетичного балансу, в тому числі за участю пацієнтів у клінічних умовах. В інших дослідженнях він слугував еталонним методом для оцінки точності інших методів вимірювання витрат енергії, таких як валідація акселерометрів або прогнозних рівнянь для REE. Порівняно з прямою калориметрією, цей метод є більш доступним і представляє перевагу надання інформації про метаболічне паливо, що спалюється, на додаток до вимірювання швидкості метаболізму.

2.2.4 Акселерометрія

Останні досягнення в технології дозволили розробити акселерометри як один із методів вимірювання енергії та витрат фізичної активності. Ці інструменти виявилися надійними, об'єктивними, менш обтяжливими для учасників, універсальними та менш дорогими порівняно з іншими методами оцінки енергії фізичної активності. Сьогодні різні моделі акселерометрів розроблені різними компаніями та доступні на ринку. Для моделей акселерометрів Actigraph і Actical з часом були розроблені різні покоління, як описано в дослідженні Джона та Фрідсона.

Цей метод заснований на вимірюванні прискорення тіла, яке є зміною швидкості в часі і виражається через кратні сили тяжіння ($g = 9{,}8$ м/S^2). Деякі моделі вимірюють прискорення в 1 площині (одноосьовий), 2 площинах (двовісний) або 3 площинах (триосьовий акселерометр). Акселерометри генерують вихідні дані у формі «відліків» за одиницю часу. Щоб перетворити ці підрахунки в одиниці витрат енергії, були розроблені прогнозні рівняння. Деякі з рівнянь дозволяють розрахувати витрати енергії як метаболічні еквіваленти (MET) або ккал/хв. Дослідження перевірки цих рівнянь були проведені за допомогою непрямої калориметрії або методу DLW. У недавньому дослідженні Lyden et al. оцінили точність загальновживаних рівнянь акселерометра для прогнозування витрат енергії та MET. Їх висновки показали, що поточні рівняння прогнозування акселерометра мають багато обмежень при перекладі показників акселерометра на витрати енергії, особливо при використанні в ряді різноманітних видів діяльності.

Першими розробленими моделями Actigraph є одноосьовий акселерометр, а нещодавно були розроблені нові моделі, які вимірюють прискорення в 3 площинах (триосьові акселерометри). У цих останніх моделях витрати енергії можуть бути розраховані шляхом інтеграції відліків у трьох площинах, а саме вертикальній (Y), горизонтальній праворуч-ліворуч (X) і горизонтальній передньо-задній осі (Z). У цьому випадку витрати енергії передбачають так званою векторною величиною.

Одне з питань, яке виникло у зв'язку з розробкою нових поколінь акселерометрів, — це порівнянність їх продуктивності з результатами попередніх поколінь. Було показано, що існує порівняння між одновісним і тривісним поколіннями однієї моделі акселерометра, коли використовуються відліки від вертикальної осі, але це не стосується випадку використання відліків VM від тривісного акселерометра.

2.2.5 Монітор серця

Монітори серцевого ритму є одними з широко використовуваних об'єктивних інструментів для вимірювання фізичної активності та витрат енергії. Їх використання ґрунтується на передбачуваному зв'язку між частотою серцевих скорочень, інтенсивністю активності та споживанням кисню, оскільки фізична активність змушує серце доставляти більше кисню до м'язових клітин, що тренуються. Дослідження встановили, що частота серцевих скорочень змінюється пропорційно інтенсивності діяльності та споживанню кисню під час фізичних навантажень середньої та інтенсивної. Однак ця кореляція є низькою у випадку сидячої та легкої діяльності, і це залишається одним із обмежень моніторів серцевого ритму. На зв'язок між частотою серцевих скорочень і VO2 впливають такі фактори, як специфічне використання м'язової маси, тип діяльності, рівень фізичної підготовки та інші фактори, пов'язані з фізичними вправами. Також повідомлялося, що деякі фактори, пов'язані з фізичними навантаженнями, впливають на зв'язок між частотою серцевих скорочень і VO2. Дослідження повідомляють про електричні або магнітні перешкоди вимірюванню частоти серцевих скорочень такими пристроями, як комп'ютери, мікрохвильові печі,

телевізори та моторизоване тренажерне обладнання, що призводить до нестабільних показань і втрати даних. Пропозиції щодо зменшення цієї проблеми електричних перешкод включають утримання монітора серцевого ритму на відстані щонайменше 1 метра від електричного кола електричних пристроїв і розміщення приймача частоти серцевих скорочень близько до передавача, таким чином посилюючи сигнал передавача відносно сигналу перешкод.

Незважаючи на ці обмеження, монітори серцевого ритму залишаються популярними серед дослідників, оскільки вони мають переваги відносно низької вартості, неінвазивності та універсальності. Їх використання дає об’єктивну та достовірну інформацію про енерговитрати та інтенсивність і тривалість діяльності. Різні дослідження показали ефективність цих інструментів у контрольованих умовах, а також у вільних умовах життя.

2.2.6 Кілометрія

Крокоміри використовуються для вимірювання ходьби, яка є однією з найбільш часто виконуваних видів діяльності та сприяє значній частці фізичного АЕЕ в анкетах і журналах фізичної активності. Основний вихід крокомірів – це підрахунок кроків. На додаток до цього можна використовувати крокоміри для оцінки пройденої відстані шляхом множення кількості кроків на довжину кроку. На довжину кроку впливають такі фактори, як швидкість ходьби, зріст, вік і стать. Деякі крокоміри відображають витрати енергії як кілокалорій, але неясно, чи вони відображають валові чи чисті кілокалорій. Було оцінено придатність 10 електронних крокомірів для вимірювання кроків, відстані та витрат енергії та виявили, що ці інструменти є найточнішими для оцінки кроків, менш точними для оцінки відстані та ще менш точними для оцінки кілокалорій. Незважаючи на їх низьку точність, крокоміри мають перевагу низької вартості, і їх можна використовувати як інструменти самоконтролю для людей, які хочуть підтримувати свій PAL.

2.2.7 Система самозвіту

Найбільш широко використовувані методи самозвіту для оцінки фізичної активності включають такі методи, як анкети щодо фізичної активності, записи та щоденники фізичної активності. Вони використовувалися у великих когортних дослідженнях, які дозволили встановити захисну роль фізичної активності проти захворювань, включаючи розлади, пов'язані з метаболічним синдромом (резистентність до інсуліну, діабет 2 типу, дисліпідемія, гіпертонія та ожиріння), серцеві та легеневі захворювання (хронічна обструктивна хвороба захворювання легень, ішемічна хвороба серця, хронічна серцева недостатність і переміжна кульгавість), захворювання м'язів, кісток і суглобів (остеоартрит, ревматоїдний артрит, остеопороз, фіброміалгія та синдром хронічної втоми), рак, депресія, астма та діабет 1 типу. Обмеження цих самозвітних методів оцінки фізичної активності полягає в їх низькій точності та надійності, як показано різними дослідженнями валідації з використанням об'єктивних показників фізичної активності або витрат енергії, таких як метод DLW та акселерометри. Проте вони доступніші за рахунок низької вартості порівняно з іншими методами оцінки фізичної активності. Вони надають інформацію про моделі фізичної активності учасників.

Найбільш часто використовувані опитувальники фізичної активності включають Міжнародний опитувальник фізичної активності (IPAQ), 7-денний опитувальник фізичної активності (PAR), опитувальник модифікованої активності (MAQ), попередній тиждень Анкета модифікованої активності (PWMAQ), Анкета недавньої фізичної активності (RPAQ) і Згадування фізичної активності минулого дня (PDPAR). Ці опитувальники становлять низький тягар для суб'єктів, але вони обмежені тим, що вони покладаються на пам'ять учасника про виконану фізичну активність, збільшуючи ризик зміщення пам'яті. Оскільки записи фізичної активності та щоденники вимагають від учасників запису виконаної діяльності, вони мінімізують зміщення пам'яті у випадку, якщо вони виконані вчасно. Однак навантаження на учасників є вищим порівняно з анкетами фізичної активності, а затримка запису може призвести до упередженості пам'яті та реактивності.

2.3 Огляд і порівняння операційних систем для інформаційної системи

Операційна система (ОС) — це програма, яка після початкового завантаження на комп’ютер програмою завантажувачем керує всіма іншими прикладними програмами на комп’ютері. Прикладні програми використовують операційну систему, надаючи запити на послуги через визначений інтерфейс прикладної програми (API). Окрім того, користувач має змогу безпосередньо взаємодіяти з операційною системою через інтерфейс користувача, такий як інтерфейс командного рядка (CLI) або графічний UI (GUI) [13].

Операційна система приносить значні переваги комп’ютерному програмному забезпеченню та розробці програмного забезпечення. Без операційної системи кожна програма мала б включати власний інтерфейс користувача, а також комплексний код, необхідний для обробки всіх низькорівневих функціональних можливостей основного комп’ютера, таких як дискове сховище, мережеві інтерфейси тощо. Враховуючи величезну кількість базового апаратного забезпечення, це значно збільшить розмір кожної програми та зробить розробку програмного забезпечення непрактичною.

Натомість багато звичайних завдань, таких як надсилання мережевого пакету або відображення тексту на стандартному пристрої виводу, такому як дисплей, можна перекласти на системне програмне забезпечення, яке служить посередником між програмами та обладнанням. Системне програмне забезпечення забезпечує послідовний і повторюваний спосіб взаємодії додатків із апаратним забезпеченням, при цьому програмам не потрібно знати будь-які подробиці про апаратне забезпечення.

Поки кожна програма отримує доступ до тих самих ресурсів і служб однаково, це системне програмне забезпечення — операційна система, може обслуговувати майже будь-яку кількість програм. Це значно скорочує кількість часу та кодування, необхідних для розробки та налагодження програми, гарантуючи, що користувачі можуть контролювати, налаштовувати та керувати апаратним забезпеченням системи через загальний і добре зрозумілий інтерфейс.

Після встановлення на комп’ютер ОС надіється на бібліотеку драйверів пристроїв для того, щоб адаптувати служби операційної системи до конкретного апаратного середовища. Після цього кожна програма може виконувати будь-який виклик до пристрою зберігання даних, але операційня система отримує даний виклик і використовує належний драйвер для перетворення виклику в дії (команди), які необхідні для базового обладнання на даному комп’ютері. Сьогодні ОС надає комплексну платформу, що ідентифікує, налаштовує та керує діапазоном обладнання, який включає в себе процесори, пристрої пам'яті та управління пам'яттю, набори різноманітних мікросхем, зберігання, комунікаційні порти, такі як Video Graphics Array (VGA), High-Definition Multimedia Interface (HDMI) і Universal Serial Bus (USB); та підсистемні інтерфейси, такі як Peripheral Component Interconnect Express (PCIe).

ОС надає три основні можливості: вона надає інтерфейс користувача через CLI або GUI: він запускає та керує виконанням програм і він визначає та дає системні апаратні ресурси програмам - як правило, через стандартизований API.

Кожна операційна система потребує інтерфейс користувача, що дає змогу користувачам і адміністраторам взаємодіяти з операційною системою, щоб встановити, конфігурувати та навіть усунути неполадки ОС та базового апаратного забезпечення. Існує два основних типи інтерфейсу користувача: CLI та GUI.

CLI (вікно термінального режиму) надає текстовий інтерфейс, де користувач покладається на звичайну клавіатуру для введення конкретних команд, аргументів і параметрів, які пов’язані з конкретними завданнями. Графічний інтерфейс (робочий стіл) надає візуальний інтерфейс за допомогою символів і піктограм, де користувачі покладаються на жести, які надаються пристроями людського інтерфейсу, такими як сенсорні екрани, сенсорні панелі та миші.

Графічний інтерфейс користувача найбільше використовується звичайним або кінцевим користувачем, який в першу чергу зацікавлений в маніпулюванні файлами та програмами, наприклад подвійним натисканням піктограми файлу, щоб внаслідок цього відкрити файл у програмі. Інтерфейс командного рядка залишається популярним серед досвідчених користувачів і системних

адміністраторів, які повинні регулярно використовувати багато детальних і повторюваних команд, наприклад створення та виконання різноманітних сценаріїв для налаштування персональних комп’ютерів (ПК) для нових співробітників.

До поширених настільних операційних систем належать такі:

Windows — флагманська операційна система Microsoft, де-факто стандарт для домашніх і робочих комп’ютерів. Представлена в 1985 році ОС на основі графічного інтерфейсу користувача відтоді була випущена в багатьох версіях. Зручна Windows 95 значною мірою сприяла швидкому розвитку персональних комп'ютерів.

Mac OS — це операційна система для лінійки ПК і робочих станцій Apple Macintosh.

Unix — це багатокористувацька операційна система, створена для гнучкості та адаптивності. Спочатку розроблена в 1970-х роках, Unix була однією з перших операційних систем, написаних мовою C.

Linux — це Unix-подібна операційна система, розроблена, щоб надати користувачам ПК безкоштовну або недорогу альтернативу. Linux має репутацію ефективної та швидкодіючої системи.

Мобільні операційні системи розроблені для задоволення унікальних потреб мобільних комп’ютерів і комунікаційних пристроїв, таких як смартфони та планшети. Мобільні пристрої, як правило, пропонують обмежені обчислювальні ресурси порівняно з традиційними ПК, тому ОС має бути зменшена за розміром і складністю, щоб мінімізувати використання власних ресурсів, одночасно забезпечуючи достатні ресурси для однієї або кількох програм, що працюють на пристрої. Мобільні операційні системи, як правило, наголошують на ефективній продуктивності, швидкому реагуванні користувачів і пильній увазі до завдань обробки даних, таких як підтримка потокового мультимедіа. Apple iOS і Google Android є прикладами мобільних операційних систем.

Не всі обчислювальні пристрої загального призначення. Величезний асортимент спеціалізованих пристроїв, включаючи домашні цифрові помічники, банкомати, системи в літаках, термінали роздрібної торгівлі (POS) і пристрої

Інтернету речей (IoT), включає комп'ютери, для яких потрібна операційна система. Принципова відмінність полягає в тому, що пов'язаний обчислювальний пристрій виконує лише одну важливу функцію, тому ОС значно спрощена та призначена як для продуктивності, так і для стійкості. ОС має працювати швидко, не виходити з ладу та витончено обробляти всі помилки, щоб продовжувати роботу за будь-яких обставин. У більшості випадків ОС постачається на мікросхемі, вбудованій у фактичний пристрій. Наприклад, медичний пристрій, який використовується в апараті життєзабезпечення пацієнта, використовує вбудовану ОС, яка має працювати надійно, щоб підтримувати життя пацієнта. Вбудований Linux є одним із прикладів вбудованої ОС.

Мережева операційна система (NOS) — це інша спеціалізована ОС, призначена для полегшення зв'язку між пристроями, що працюють у локальній мережі (LAN). NOS забезпечує комунікаційний стек, необхідний для розуміння мережевих протоколів для створення, обміну та розкладання мережевих пакетів. Сьогодні концепція спеціалізованої NOS значною мірою застаріла, оскільки інші типи ОС значною мірою обслуговують мережевий зв'язок. Windows 10 і Windows Server 2019, наприклад, містять широкі мережеві можливості. Концепція NOS все ще використовується для деяких мережевих пристроїв, таких як маршрутизатори, комутатори та брандмауери, і виробники можуть використовувати власні NOS, включаючи Cisco Internetwork Operating System (IOS), RouterOS і ZyNOS.

Коли обчислювальний пристрій має взаємодіяти з реальним світом у постійних і повторюваних часових обмеженнях, виробник пристрою може вибрати використання операційної системи реального часу (RTOS). Наприклад, промислова система керування може керувати роботою великої фабрики чи електростанції. Таке обладнання вироблятиме сигнали від безлічі датчиків, а також надсилатиме сигнали для роботи клапанів, приводів, двигунів та незліченної кількості інших пристроїв. У таких ситуаціях промислова система управління повинна швидко й передбачувано реагувати на зміну реальних умов — інакше може статися катастрофа. RTOS має функціонувати без буферизації, затримок обробки та інших

затримок, які цілком прийнятні в інших типах операційних систем. Два приклади RTOS включають FreeRTOS і VxWorks.

Відмінності між типами операційних систем не є абсолютними, і деякі операційні системи можуть мати спільні характеристики з іншими. Наприклад, операційні системи загального призначення зазвичай включають мережеві можливості, наявні в традиційних NOS. Подібним чином вбудована операційна система зазвичай включає атрибути RTOS, тоді як мобільна операційна система все ще може запускати численні програми одночасно, як і інші операційні системи загального призначення.

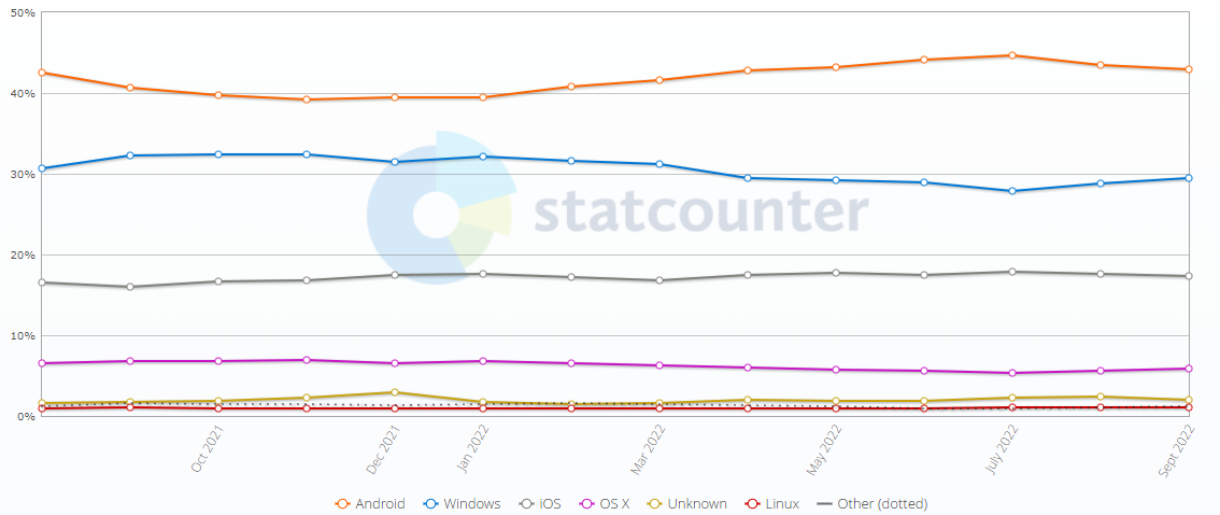

Рисунок 2.1 – Доля операційних систем на світовому ринку

Microsoft Windows (також згадується як Windows або Win) — це графічна ОС, розроблена й опублікована Microsoft. Вона надає можливість зберігати файли, запускати програмне забезпечення, грати в ігри, переглядати відео та підключатися до мережі Інтернет [14].

Операційна система Microsoft Windows була вперше презентована з версією 1.0 10 листопада 1983 року. Після цього було випущено більш ніж десять версій Windows, включаючи версію Windows 11.

Microsoft, починаючи з Windows XP, випустила різні версії Windows. Кожні з цих версій Windows включають однакову основну ОС, але деякі версії мають додаткові функції за окрему плату.

Двома найпоширенішими додатковими версіями Windows для ПК є Windows Home і Windows Professional.

Windows Home (Win Home) — це базова версія Windows. Вона забезпечує всі основні функції Windows, такі як підключення до мережі Інтернет, перегляд веб-сторінок, перегляд відеофайлів, використання офісного ПЗ та інше. Це найдешевша версія Windows, яка попередньо встановлена на більшості нових ПК [15].

Windows Professional (Windows Pro або Win Pro) — це розширена версія Windows для досвідчених користувачів. Вона містить усі функції Windows Home, а також додаткові:

— віддалений робочий стіл — він дозволяє віддалено керувати іншим комп'ютером Windows, підключеним до мережі Інтернет;

— Bitlocker — вбудоване шифрування файлів Microsoft;

— Trusted Boot — забезпечує шифрування завантажувача, таким чином захищаючи ПК від руткітів;

— Hyper-V — гіпервізор ОС Windows для запуску віртуальних машин, еквівалент програмного забезпечення сторонніх розробників, наприклад VirtualBox;

— пісочниця Windows — надає легкий екземпляр ОС Windows 10 із ізольованим програмним середовищем, яке можна використовувати «Windows у Windows» для безпечного запуску ненадійного або підозрілого ПЗ, але Windows Sandbox вимагає Windows Insider збірки Windows 10 Enterprise або Pro;

— керування груповою політикою — адміністратори мають змогу визначати групову політику для керування декількома користувачами Windows у компанії чи організації;

— підтримка понад 128 ГБ оперативної пам'яті;

— розширені параметри інсталяції Windows Update, включаючи більш гнучке планування та відкладення на термін до 35 днів.

Linux (вимовляється як «linnucks») — це монолітне ядро з відкритим кодом і сімейство операційних систем, заснованих на цьому ядрі. Ядро Linux спочатку було розроблено Лінусом Торвальдсом, який оголосив про це в групі новин

comp.os.minix 25 серпня 1991 року. Відтоді його було перенесено на комп'ютерні архітектури, включаючи x86-64, x86, ARM, RISC і DEC Alpha. . Він ліцензований під версією 2 GPL [16].

Розробники мають доступ до всього вихідного коду Linux, і відповідно до умов ліцензії їм дозволяється змінювати та поширювати його.

Зараз Linux використовується кількома мільйонами користувачів у всьому світі. Склад користувачів варіюється від приватних користувачів, навчальних центрів, університетів, дослідницьких центрів і компаній. Нижче наведено приклади того, де сьогодні використовується Linux:

— телефони та планшети Android – телефони та планшети Android використовують форму Linux;

— сервери — переважна більшість веб-серверів, які запускають багато веб-сторінок (включно з цією), використовують Linux;

— суперкомп'ютери, які використовують операційну систему на базі Linux;

— телевізори, фотоапарати, DVD-програвачі та більшість пристроїв, які використовують певну форму комп'ютера;

— Amazon — багато комп'ютерів, які допомагають запускати даний сервіс;

— Google — комп'ютери, які допомагають запускати Google і результати пошуку Google, використовують Linux;

— літаки — комп'ютери та екрани літаків, які знаходяться у літаку, використовують Linux;

— поштова служба США – комп'ютери та сервери, які допомагають запускати системи для сортування та керування поштою в США;

— NYSE — Нью-Йоркська фондова біржа використовує Linux для підтримки своєї біржі;

— LHC — великий адронний колайдер використовує Linux;

— OLPC — програма One Laptop Per Child використовувала Linux на всіх своїх комп'ютерах.

Тисячі організацій, корпорацій і окремих людей допомагають розробляти Linux і кожен з його різноманітних дистрибутивів.

Linux можна отримати двома різними способами. Всі необхідні компоненти можна завантажити безкоштовно з інтернету, а значить, зібрати операційну систему можна практично за безцінь. Альтернативою є використання так званого дистрибутива, який є різновидом Linux, який пропонують багато компаній. Вони містять широкий спектр додатків і повних програм, які значно спрощують установку Linux. Було випущено сотні різних дистрибутивів Linux. Чудовим сайтом, який містить список майже всіх дистрибутивів і рейтинги, є DistroWatch.

MacOS, раніше (1984–2001) Mac OS і (2001–2016) Mac OS X, операційна система (ОС), яка була розроблена американською компанією Apple Inc. Дана операційна система була представлена в 1984 році для запуску лінійки персональних комп’ютерів під назвою Macintosh. Macintosh став ознаменуванням ери систем графічного інтерфейсу користувача (GUI), і ця подія надихнула компанію Microsoft на розробку власного GUI, операційну систему Windows [17].

Маркетинг компанії Apple для представлення Macintosh зосереджувався на інтуїтивно зрозумілій простоті використання ОС. Відмінно від майже всіх інших сучасних персональних комп'ютерів, Mac OS (спочатку називалася просто системним програмним забезпеченням із додаванням номера версії) базувалася на графічному процесорі. Замість того, щоб не вводити команди та великі шляхи до каталогів у текстових підказках, користувач переміщував вказівник миші для того, щоб візуально переміщуватися у Finder — серії віртуальних файлів і папок, представлених значками. Більшість комп'ютерних ОС зрештою перейняли модель GUI. В 1980-х роках Apple уклала угоду, яка дозволяла компанії Microsoft використовувати деякі аспекти інтерфейсу ОС Mac у ранніх версіях операційної системи Windows. Але, за винятком періоду в 1990-х роках, Mac OS ніколи не отримувала ліцензії на використання з ПК інших виробників, крім Apple [18].

Пізніші випуски системи Mac OS представили такі функції, як обмін файлами в мережі Інтернет, перегляд мережі та декілька облікових записів користувача. У 1996 році Apple придбала конкуруючу компанію NeXT Computers, яку заснував

Стів Джобс після його відходу з Apple, а вже у 2001 році компанія випустила Mac OS X, значну переробку на основі системи NextStep і останньої версії ОС від Apple. OS X працювала на основі ядра UNIX (основний програмний код) і пропонувала такі технічні переваги, як захист пам'яті та попереджувальна багатозадачність, також удосконаленіший Finder, красивий інтерфейс, який мав назву Aqua та зручну графічну панель «Dock» для запуску програм, які часто використовуються. Оновлення OS X додали багато функцій, такі як автоматичне резервне копіювання та менеджер «Інформаційна панель» для невеликих зручних програм, які називаються віджетами.

У 2007 році Apple представила низку мобільних пристроїв, які мали доступ до мережі Інтернет, включаючи смартфони iPhone і планшет iPad. Невдовзі Apple підкреслила здатність операційної системи OS X підключатися до даних пристроїв. В 2011 році компанія Apple представила сервіс iCloud, службу хмарних обчислень, яка дозволяла користувачам обмінюватися будь-якими даними між усіма пристроями компанії Apple як для OS X, так само і для мобільної операційної системи iOS. Apple створила більше функцій, що дають змогу підключати пристрої до послідовних оновлень OS X, iOS і пізніших версій watchOS (операційної системи для смарт-годинників Apple Watch). Всі ці функції мали можливість приймати телефонні дзвінки (на iPhone) та інструменти швидкого обміну даними (фотографіями чи текстом) між пристроями. У 2016 році Apple перейменувала систему MacOS, щоб відповідати назвам iOS і watchOS.

Операційна система Android найчастіше використовується на різних мобільних платформах у всьому світі. На кінець 2020 року вона займає приблизно 75% акцій світового ринку. Така компанія, як Open Handset Alliance, розробила перший Android, який базується на налаштованій версії ядра Linux, а також іншого програмного забезпечення з відкритим кодом. На початковому етапі 2005 року Google спонсорувала проект і отримала всю компанію. У вересні 2008 року на ринку було випущено перший пристрій Android, який домінував у індустрії мобільних пристроїв завдяки кільком функціям, таким як зручність користувача, величезна підтримка спільноти, налаштування, виробництво пристроїв Android у

великих компаніях. Отже, ринок вивчає попит на розробку пристроїв з підтримкою Android за допомогою розумних розробників. Таким чином, операційна система Android стала повним набором операційних систем для різних пристроїв, таких як мобільні телефони, ноутбуки, смарт-телевізори, планшети, приставки тощо [19].

Android — це операційна система на базі Linux, розроблена в основному для мобільних пристроїв із сенсорним екраном, таких як смартфони та планшетні комп'ютери. Операційна система значно розвинулася за останні 15 років, починаючи від чорно-білих телефонів і закінчуючи останніми смартфонами чи міні-комп'ютерами. Однією з найпоширеніших мобільних ОС сьогодні є android. Android — це програмне забезпечення, яке було засновано в Пало-Альто в Каліфорнії в 2003 році.

Android — потужна операційна система, яка підтримує велику кількість програм на смартфонах. Ці програми більш зручні та розширені для користувачів. Апаратне забезпечення, яке підтримує програмне забезпечення Android, базується на платформі архітектури ARM. Android — це операційна система з відкритим вихідним кодом, що означає, що вона безкоштовна і будь-хто може нею користуватися. Android має мільйони доступних програм, які можуть допомогти вам керувати своїм життям так чи інакше, і вони доступні за низькою ціною на ринку, тому Android дуже популярний.

Розробка Android підтримує повну мову програмування Java. Навіть інші пакети API та JSE не підтримуються. Перша версія 1.0 набору для розробки Android (SDK) була випущена в 2008 році, а остання оновлена версія — це Jelly Bean.

Унікальні особливості та характеристики операційної системи android включають наступне:

— комунікація ближнього поля (NFC);
— альтернативні клавіатури;
— ІЧ-передача;
— безсенсорне керування;
— автоматизація;
— бездротове завантаження програм;

— зберігання та заміна батареї;

— спеціальний головний екран;

— віджети;

— спеціальні ПЗУ;

— макет гарнітури;

— зберігання;

— підключення: GSM/EDGE, IDEN, CDMA, Bluetooth, WI-FI, EDGE,3G, NFC, LTE, GPS;

— повідомлення: SMS, MMS, C2DM (може надсилати повідомлення на пристрій), GCM (може надсилати повідомлення Google);

— багатомовна підтримка;

— мультитач;

— відеодзвінок;

— скріншот;

— зовнішня пам'ять;

— підтримка потокового медіа;

— оптимізована графіка.

Android — це операційна система, яка являє собою стек програмних компонентів, який поділено на п’ять розділів і чотири основні рівні:

— ядро Linux;

— бібліотеки;

— середа виконання Android;

— фреймворк програми;

— додатки.

Після Android iOS визнана другою найбільшою операційною платформою для мобільних пристроїв у всьому світі. Вона має інтуїтивно зрозумілий дизайн, орієнтований на користувача, і розробники додатків використовують його для розробки додатків, запущених через магазин додатків iOS [20].

Крім того, згідно з даними в червні 2021 року, Apple iOS займає близько 26,3% частки ринку мобільних телефонів, поступаючись Android з приблизно 73,3% ринку.

Ця операційна система широко відома як базове програмне забезпечення, яке дозволяє користувачам iPhone взаємодіяти зі своїми мобільними телефонами за допомогою жестів, як-от торкання, змахування та зведення щипків.

ОС для iPhone, iPad та інших мобільних пристроїв Apple базується на macOS. Ця операційна система працює в лінійці ноутбуків і настільних комп'ютерів Mac від Apple. Крім того, вона створена для безперебійного та легкого підключення до мережі між широким спектром продуктів Apple.

Простіше кажучи, мобільна ОС, розроблена ексклюзивно Apple Inc. для свого апаратного забезпечення — це iOS. Це операційна система, яка посилює мобільні пристрої різних компанії.

Програмне забезпечення, створене для використання на пристроях iPhone від Apple на базі iOS, є додатком iPhone. Такі програми доступні в Apple App Store і створені для роботи в мобільній операційній системі Apple iOS.

Apple мотивує розробників програмувати свої програми для iPhone для завантаження з App Store. Крім того, компанія також випустила SDK разом із проектами зразків коду, щоб допомогти розробникам зробити крок вперед до початку.

В Apple Store можна завантажити програми для iPhone безкоштовно або придбати їх. Доходи від завантажень або придбання програм для iPhone розподіляються між Apple і розробником програмного забезпечення. У додатках для iPhone покупки в програмі дають розробникам додатковий варіант прибутку.

Розробка мобільних програм для апаратного забезпечення Apple, включаючи iPad, iPhone та iPod Touch, є розробкою програм для iOS. Програмне забезпечення написано мовою програмування Objective-C або Swift і розгорнуто в App Store для завантаження користувачами.

Розробники використовують пакет розробки програмного забезпечення iOS (SDK) для створення програм для мобільних пристроїв Apple. SDK містить

інтерфейси та інструменти для створення, встановлення, запуску та тестування програм.

Рідні програми написані з використанням системних фреймворків iOS разом із мовою програмування Objective-C.

Перед запуском програми в App Store вона проходить перевірку якості.

В iOS програми керуються подіями. Об’єкт/елемент керування (наприклад, UIButton) прослуховує певні події (наприклад, натискання та торкання). Коли користувач запускаєте подію, об’єкт викликає попередньо встановлену дію, пов’язану з подією.

У той час як програми для Android розробляються в основному за допомогою Java і Kotlin, програми для iOS створюються за допомогою Swift. Суттєва відмінність між цими двома мовами програмування полягає в тому, що розробка додатків iOS за допомогою Swift вимагає написання менше коду. Ось чому проекти кодування програм для iOS завершуються швидше порівняно з програмами, розробленими для телефонів Android.

Після детального аналізу операційних систем, їх особливостей та поширеністю на ринку було вирішено обрати ОС Android для розробки інформаційної системи моніторингу фізіологічного стану людини.

2.4 Огляд мов програмування для Android

Хоча Kotlin є офіційною мовою для Android, існує багато інших мов, які можна використовувати для розробки програм Android. Подробиці про них наведено нижче.

Спочатку Java була офіційною мовою для розробки додатків для Android (але тепер її замінив Kotlin), і, отже, вона також є найбільш використовуваною мовою. Багато програм у магазині Play Market створено на Java, і це також найбільш підтримувана мова Google. На додаток до всього цього, Java має чудову онлайн-спільноту для підтримки у разі будь-яких проблем.

Однак Java є складною мовою для початківців, оскільки вона містить такі складні теми, як конструктори, винятки нульового покажчика, паралелізм,

перевірені винятки тощо. Крім того, Android Software Development Kit (SDK) підвищує складність на новий рівень!

Загалом, Java — чудова мова, щоб відчути всі переваги розробки додатків для Android. Однак це може бути трохи складніше для початківців, які воліють почати з чогось легшого, а потім повернутися до цього.

Зараз Kotlin є офіційною мовою для розробки додатків для Android, оголошеною Google у 2019 році. Kotlin — це кросплатформна мова програмування, яка може використовуватися як альтернатива Java для розробки додатків для Android. У 2017 році він також був представлений як додаткова «офіційна» мова Java. Kotlin може взаємодіяти з Java і працює на віртуальній машині Java.

Єдина значна відмінність полягає в тому, що Kotlin видаляє зайві функції Java, такі як винятки нульового покажчика. Це також усуває необхідність закінчувати кожен рядок крапкою з комою. Початківцям набагато простіше випробувати Kotlin порівняно з Java, і його також можна використовувати як «точку входу» для розробки додатків для Android.

C++ можна використовувати для розробки додатків Android за допомогою Android Native Development Kit (NDK). Однак програму неможливо створити повністю за допомогою C++, і NDK використовується для реалізації частин програми у рідному коді C++. Це допомагає використовувати бібліотеки коду C++ для програми за потреби.

Хоча в деяких випадках C++ корисний для розробки додатків для Android, його набагато складніше налаштувати та він набагато менш гнучкий. Це також може призвести до збільшення кількості помилок через підвищену складність. Отже, краще використовувати Java порівняно з C++, оскільки вона не забезпечує достатнього приросту, щоб компенсувати необхідні зусилля.

C# дуже схожий на Java, тому він ідеально підходить для розробки програм для Android. Як і Java, C# також реалізує збирання сміття, тому є менше шансів витоку пам’яті. Крім того, C# має чистіший і простіший синтаксис, ніж Java, що робить кодування з його допомогою порівняно легшим.

Раніше найбільшим недоліком C# було те, що він міг працювати лише в системах Windows, оскільки використовував .NET Framework. Однак цю проблему вирішив Xamarin (раніше Mono для Android) — це кросплатформна реалізація спільної мовної інфраструктури. Інструменти Android можна використовувати для написання рідних програм для Android і спільного використання коду на кількох платформах.

Python можна використовувати для розробки програм Android, навіть якщо Android не підтримує власну розробку Python. Це можна зробити за допомогою різних інструментів, які перетворюють програми Python на пакети Android, які можна запускати на пристроях Android.

Прикладом цього є Kivy, бібліотека Python з відкритим кодом, яка використовується для розробки мобільних програм. Вона підтримує Android, а також заохочує швидку розробку програм. Однак недоліком цього є те, що Kivy не матиме нативних переваг, оскільки він не підтримується нативно.

Програми для Android можна створювати за допомогою HTML, CSS і JavaScript за допомогою фреймворку Adobe PhoneGap, який підтримує Apache Cordova. Фреймворк PhoneGap в основному дозволяє використовувати навички веб-розробки для створення гібридних програм, які відображаються через «WebView», але упаковані як програма.

Хоча фреймворку Adobe PhoneGap достатньо для базових завдань у сфері розробки програм для Android, він навряд чи потребує багато програмування, окрім JavaScript. І оскільки навіть для створення пристойної програми потрібно багато працювати, краще використовувати інші мови. Але якщо зручно працювати з Javascript, ви можна вивчити React Native, який є фреймворком з відкритим вихідним кодом, який зараз дуже затребуваний. Можна розробляти гарні та потужні гібридні програми за допомогою нативної системи React, тобто програма буде працювати як на Android, так і на iOS.

Dart — це мова програмування з відкритим вихідним кодом, яка є основою фреймворку Flutter, який сьогодні набуває великої популярності завдяки своїй здатності створювати гарні та продуктивні програми для Інтернету, комп’ютера та

мобільних пристроїв за менший час. Ключова перевага dart полягає в тому, що він розроблений Google як клієнтська оптимізована мова для швидких програм на будь-якій платформі. Dart головним чином зосереджується на полегшенні розробки інтерфейсу користувача для розробників за допомогою таких функцій, як гаряче перезавантаження, яке дозволяє розробникам миттєво бачити зміни під час роботи над програмою. Dart також відомий своєю високою продуктивністю, він компілюється в машинний код ARM і x64 для мобільних пристроїв, комп'ютерів і серверної частини.

Corona — це набір для розробки програмного забезпечення, який можна використовувати для розробки програм Android за допомогою Lua. Він має два режими роботи, а саме Corona Simulator і Corona Native. Corona Simulator використовується для безпосереднього створення програм, тоді як Corona Native використовується для інтеграції коду Lua з проектом Android Studio для створення програми з використанням нативних функцій.

Хоча Lua трохи обмежена порівняно з Java, вона також набагато простіша та має легшу криву навчання. Крім того, є функції монетизації збірки, а також різні ресурси та плагіни, які збагачують досвід розробки додатків. Corona в основному використовується для створення графічних програм та ігор, але не обмежується цим.

Існує багато програм, таких як Chat Messenger, музичні плеєри, ігри, калькулятори тощо, які можна створити за допомогою вищевказаних мов. І немає мови, яку можна було б назвати «правильною мовою» для розробки програм Android.

Після проведення аналізу популярних мов програмування для Android було вирішено вибрати для розробки саме Java через її підтримку, надійність та функціональність.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ФІЗІОЛОГІЧНОГО СТАНУ ЛЮДИНИ

3.1 Розробка основних компонентів застосунку

3.1.1 Розробка життєвих циклів мобільного застосунку для Android

Життєвий цикл застосунку в Android розпочинається, коли користувач натискає на іконку додатку. Такий дотик спричиняє до виконання методу onCreate() (рис. 3.1) режим очікування (в термінології Android, активності) додатку. Даний метод містить код для відображення на екрані потрібних елементів, які включені в розмітці потрібної активності. Розробник повинен доповнити даний метод кодом, який буде ініціалізувати змінні та об'єкти розмітки до значень, які необхідниі, щоб користувач почав взаємодію з застосунком.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    MyMenuActivity.isOpen = true;
    setContentView(R.layout.activity_my_menu);
    findViewById(R.id.from_menu_to_diary).setOnClickListener(new ToWindowOnClick(this, DiaryActivity.class));
    findViewById(R.id.from_menu_to_diary_menu).setOnClickListener(new ToWindowOnClick(this, DiaryActivity.class));
    findViewById(R.id.from_menu_to_profile).setOnClickListener(new ToWindowOnClick(this, ProfileActivity.class));
    findViewById(R.id.from_menu_to_dishes).setOnClickListener(new ToWindowOnClick(this, DishActivity.class));
    findViewById(R.id.from_menu_to_exercises).setOnClickListener(new ToWindowOnClick(this, ExerciseActivity.class));
    findViewById(R.id.from_menu_to_statistic).setOnClickListener(new ToWindowOnClick(this, StatisticActivity.class))
}
```

Рисунок 3.1 – Реалізація методу onCreate() у застосунку

Після того, як активність застосунку розпочалася, йде метод onResume(). Цей метод не обов'язково реалізовувати, але це корисно робити, щоб повернути застосунок в стан виконання, у якому він знаходився до переходу в стан паузи. Це, зокрема, відноситься до різних системних служб, які використовуються застосунком (наприклад, служб GPS або камери), перезапуску анімації, а також до будь-яких інших налаштувань, які необхідні для відновлення роботи користувача з додатком.

Коли користувач вирішує зупинити взаємодію з застосунком, розпочинається гілка його видалення. Ніякий з методів, які запущені на гілці видалення, не є обов'язковими для реалізації. Проте, вони часто виявляються корисними та тому їх потрібно ретельно розглянути на предмет реалізації. Перший метод, який

виконується — це метод onPause(). Даний метод повинен використовуватися для призупинки використаних застосунком служб, для зупинки анімації, для збереження необхідної інформації про стан для того, щоб користувач міг відновити користування застосунком з того стану, в якому він його залишив.

Якщо застосунок збирається стати невидимим, викликається на виконання метод onStop(). Цей метод повинен гарантувати, що всі важливі дані додатку передані на постійне зберігання для того, щоб вони не загубилися, коли системні ресурси цього додатку будуть перерозподілені між іншим додатками.

Якщо додаток має завершити свою роботу та знищити всі свої дані з оперативної пам'яті, то операційною системою викликається на виконання метод onDestroy() (рис. 3.2).

```
protected void onDestroy() {
    super.onDestroy();
    db.close(); // close DB
}
```

Рисунок 3.2 – Реалізація методу onDestroy() у застосунку

3.1.2 Реалізація основного функціоналу застосунку

При запуску застосунку першим буде відображатися екран «Меню», через який користувач буде переміщуватися по іншим екранам (лістинг 3.1).

Лістинг 3.1 — Екран «Меню»

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
MyMenuActivity.isOpen = true;
setContentView(R.layout.activity_my_menu);
findViewById(R.id.from_menu_to_diary).setOnClickListener(new
ToWindowOnClick(this, DiaryActivity.class));
findViewById(R.id.from_menu_to_diary_menu).setOnClickListener(new
ToWindowOnClick(this, DiaryActivity.class));
```

```
findViewById(R.id.from_menu_to_profile).setOnClickListener(new
ToWindowOnClick(this, ProfileActivity.class));
findViewById(R.id.from_menu_to_dishes).setOnClickListener(new
ToWindowOnClick(this, DishActivity.class));
findViewById(R.id.from_menu_to_exercises).setOnClickListener(new
ToWindowOnClick(this, ExerciseActivity.class));
findViewById(R.id.from_menu_to_statistic).setOnClickListener(new
ToWindowOnClick(this, StatisticActivity.class));
 }
```

У щоденнику користувач буде повинен вибирати спожиті страви та виконані фізичні вправи за день, які будуть вибиратися із переліку вже створених об’єктів. Отже потрібно буде звертатися до бази даних, в якій будуть знаходитися дані об’єкти. Використовуючи дані, які були внесені користувачем, застосунок обраховуватиме калорії. Також користувач може викликати календар для переходу між днями (лістинг 3.2).

Лістинг 3.2 – Екран «Щоденник»

```
protected void onCreate(Bundle savedInstanceState) {
profile = Profile.getProfile(this);
db = DataBase.getDataBase(this);
setViewDay(new MyDate());
dishesData =
DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate()));
String[] from = new String[]{DataBase.DISH_COLUMN_NAME,
DataBase.DAYS_DISH_COLUMN_WEIGHT};
int[] to = new int[]{R.id.db_item_name, R.id.db_item_right_text};
dishesAdapter = new SimpleAdapter(this, dishesData, R.layout.database_item,
from, to);
exercisesData =
DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.getDate()));
```

```
from = new String[]{DataBase.EXERCISE_COLUMN_NAME,
DataBase.DAYS_EXERCISE_COLUMN_QUANTITY};
exercisesAdapter = new SimpleAdapter(this, exercisesData,
R.layout.database_item, from, to);
}
```

У екрані «Профіль» користувач вносить дані про себе (стать, вага, вік та зріст). Ці дані використовуються для обрахунку калорій, які будуть використовуватися у щоденнику. Для обрахунку використовується спеціальна формула (лістинг 3.3).

Лістинг 3.3 — Створення екрану «Профіль»

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_profile);
profile = Profile.getProfile(this);
findViewById(R.id.from_profile_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
((EditText) findViewById(R.id.profile_age)).setText(profile.getAge() + "");
((EditText) findViewById(R.id.profile_height)).setText(profile.getHeight() + "");
((EditText) findViewById(R.id.profile_weight)).setText(profile.getWeight() +
"");
((RadioButton)
findViewById(R.id.profile_woman_radio_button)).setChecked(!profile.getGende
r());
((EditText)
findViewById(R.id.profile_aim_kal_number)).setText(profile.getAimCalorie() +
"");
optimalCalorieNumber = ((TextView)
findViewById(R.id.profile_optimal_kal_number));
```

```
optimalCalorieNumber.setText(profile.calculateCalories() + " " +
getString(R.string.kilocalories));
}
```

У розділі «Страви» знаходиться список страв, які створює, редагує та видаляє сам користувач. При редагуванні чи створення нової страви з'являється спеціальний екран для цих дій. Всі об'єкти заносяться в базу даних застосунку. Значення калорій та назву страви вводить користувач (Лістинг 3.4).

Лістинг 3.4 — Реалізація екрану «Страви»

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_dish);
findViewById(R.id.from_dishes_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
db = DataBase.getDataBase(this);
data=DataBase.cursorToArrayList(db.getDishes());
String[] from = new String[] {DataBase.DISH_COLUMN_NAME,
DataBase.DISH_COLUMN_CALORIES_PER_100_GM }; // columns names
int[] to = new int[] { R.id.db_item_name, R.id.db_item_right_text}; // places to
write (View id)
sAdapter = new SimpleAdapter(this, data ,  R.layout.database_item, from, to);
listView= findViewById(R.id.dish_list_view);
listView.setAdapter(sAdapter);
listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
selectedElementId = id;
}
```

```
});
```

Розділ «Вправи» складається із списку фізичних активностей, які можна також редагувати, створювати та редагувати. Всі об’єкти також заносяться у базу даних застосунку. Аналогічно, як у попередньому розділі, для створення та редагування вправ викликається окремий екран, де користувач змінює загальну формулу обрахунку калорій для усіх вправ (лістинг 3.5).

Лістинг 3.5 – Реалізація екрану «Вправи»

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_exercise);
findViewById(R.id.from_exercises_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
db = DataBase.getDataBase(this);
data = DataBase.cursorToArrayList(db.getExercises());
String[] from = new String[]{DataBase.EXERCISE_COLUMN_NAME};
int[] to = new int[]{R.id.db_item_name};
sAdapter = new SimpleAdapter(this, data, R.layout.database_item, from, to);
listView = (ListView) findViewById(R.id.exercise_list_view);
listView.setAdapter(sAdapter);
listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
selectedElementId = id;
}
});
}
```

У розділі «Статистика» візуалізуються витрачені калорії із кожної страви, які були вибрані у щоденнику (лістинг 3.6).

Лістинг 3.6 – Реалізація екрану «Статистика»

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_statistic);
findViewById(R.id.from_statistic_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
barChart = findViewById(R.id.statistic_histogram);
db = DataBase.getDataBase(this);
profile = Profile.getProfile(this);
showDiagram();
}
private void showDiagram() {
switch (diagramID) {
case DAY_DIAGRAM:
showDayDiagram();
case MONTH_DIAGRAM:
showMonthDiagram();
break;
case YEAR_DIAGRAM:
showYearDiagram();
break;
}
}
```

3.2 Розробка системних компонентів

3.2.1 GPS

Об’єкт Location представляє географічне розташування, яке може складатися з широти, довготи, позначки часу та іншої інформації, такої як пеленг, висота та швидкість. Існують наступні важливі методи, які можна використовувати з об’єктом Location для отримання інформації про місцезнаходження.

Щоб отримати поточне місцезнаходження, створюємо клієнт розташування, який є об’єктом LocationClient, підключаємо його до служб визначення місцезнаходження за допомогою методу connect(), а потім викликаємо його методом getLastLocation(). Цей метод повертає останнє місцезнаходження у формі об’єкта Location, який містить координати широти та довготи та іншу інформацію, як пояснено вище. Щоб мати функціональність на основі розташування, доведеться реалізувати два інтерфейси:

— GooglePlayServicesClient.ConnectionCallbacks;

— GooglePlayServicesClient.OnConnectionFailedListener.

Ці інтерфейси забезпечують наступні важливі методи зворотного виклику, які потрібно реалізувати у вашому активності.

Далі слід створити клієнт розташування в методі onCreate() класу активності, а потім підключити його в onStart(), щоб служби визначення місцезнаходження зберігали поточне місцезнаходження, поки діяльність була повністю видимою. Потрібно від’єднати клієнт у методі onStop(), щоб, коли програма не відображається, служби визначення місцезнаходження не підтримувала поточне місцезнаходження. Це значною мірою допомагає заощадити заряд акумулятора.

Для оновлення місцеположення, окрім вищезазначених інтерфейсів, потрібно буде також реалізувати інтерфейс LocationListener. Цей інтерфейс забезпечує наступний метод зворотного виклику, який потрібно реалізувати у класі активності.

Тепер, якщо програмі потрібна висока точність місцезнаходження, вона повинна створити запит про місцезнаходження з setPriority(int) у значення

PRIORITY_HIGH_ACCURACY і setInterval(long) у 5 секунд. Можна також використовувати більший інтервал або інші пріоритети, наприклад PRIORITY_LOW_POWER, щоб запитувати точність рівня "місто" або PRIORITY_BALANCED_POWER_ACCURACY для точності рівня "блок".

Діяльність має наполегливо розглянути питання про видалення всіх запитів про місцезнаходження під час переходу у фоновий режим (наприклад, у onPause()) або принаймні змінити запит на більший інтервал і нижчу якість, щоб заощадити енергоспоживання.

Далі використовуємо метод Geocoder.getFromLocation(), щоб отримати адресу для заданої широти та довготи. Цей метод є синхронним і може зайняти багато часу, щоб виконати свою роботу, тому потрібно викликати метод з методу doInBackground() класу AsyncTask.

Для використання AsyncTask має бути підкласом, і підклас замінить метод doInBackground(Params...) для виконання завдання у фоновому режимі, а метод onPostExecute(Result) викликається в потоці інтерфейсу користувача після завершення фонового обчислення та під час відобразити результат. В AyncTask доступний ще один важливий метод — execute(Params... params), цей метод виконує завдання із зазначеними параметрами.

Нижче наведено вміст файлу основної діяльності MainActivity.java із підключеними методами для виявлення розташування (Лістинг 3.7).

Лістинг 3.7 — Код файлу MainActivity.java

```
public class MainActivity extends Activity {
  Button btnShowLocation;
  private static final int REQUEST_CODE_PERMISSION = 2;
  String mPermission = Manifest.permission.ACCESS_FINE_LOCATION;
  // GPSTracker class
  GPSTracker gps;
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  try {
    if (ActivityCompat.checkSelfPermission(this, mPermission)
      != MockPackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(this, new String[]{mPermission},
        REQUEST_CODE_PERMISSION);
      // If any permission above not allowed by user, this condition will
        execute every time, else your else part will work
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

3.2.2 Gradle

3.2.2.1 Процес перетворення вихідного коду в програму Android

Вихідні файли Java перетворюються на файли класів Java компілятором Java. Android SDK містить інструмент під назвою dx, який перетворює файли класів Java у файл .dex (Dalvik Executable). Усі файли класів програми розміщено в цьому файлі .dex. Під час цього процесу перетворення надлишкова інформація у файлах класів оптимізується у файлі .dex. Наприклад, якщо той самий рядок знайдено в різних файлах класу, файл .dex містить лише одне посилання на цей рядок. Тому ці файли .dex набагато менші за розміром, ніж відповідні файли класів.

Файл .dex та інші ресурси, наприклад зображення та XML-файли, упаковуються у файл .apk (пакет Android). Програма aapt (Android Asset Packaging Tool) виконує цей крок.

Отриманий файл .apk містить усі необхідні дані для запуску програми Android і може бути розгорнуто на пристрої Android за допомогою інструменту adb.

Починаючи з Android 5.0, Android RunTime (ART) використовується як середовище виконання для всіх програм Android. ART використовує комбінацію компіляції Ahead Of Time і _Just In Time_. Під час інсталяції програми на пристрої Android код програми транслюється в машинний код.

Інструмент dex2oat бере файл .dex, створений ланцюгом інструментів Android, і компілює його у виконуваний і компонований формат (файл ELF). Цей файл містить код dex, скомпільований рідний код і метадані. Збереження коду .dex дозволяє існуючим інструментам працювати.

Система збірки Gradle розроблена для підтримки складних сценаріїв створення додатків Android:

— мультирозповсюдження: один і той же додаток необхідно налаштувати для кількох клієнтів або компаній;

— multi-apk: підтримка створення кількох apk для різних типів пристроїв із повторним використанням частин коду.

Запускається збірка Gradle за допомогою командного рядка. Ось огляд важливих завдань Android Gradle, які представлені в таблиці 3.1.

Таблиця 3.1 — Цілі збірки Android Gradle

Команда	Опис
./gradlew build	Проект побудови, запускає завдання зібрати та перевірити
./gradlew clean build	Побудова проекта з нуля
./gradlew test	Виконання тестів
./gradlew connectedAndroidTest	Проведення випробування приладів

Для перегляду всіх доступних завдань використовуємо команду gradlew wrapper (Лістинг 3.8).

Лістинг 3.8 — Перегляд доступних завдань

```
gradle build

# alternatively speedup second grandle build by holding it in memory

# gradle build –daemon
```

Ця команда створює в папці збірки вихід збірки Gradle. За замовчуванням збірка Gradle створює два файли .apk у папці build/outputs/apk.

Для створення та запуску модульних тестів на JVM використовуємо команду gradle test.

Для створення та запуску інструментальних тестів на пристрої Android використовуємо команду gradle connectedCheck.

3.2.2.2 Робота з ресурсами

Для перегляду всіх доступних завдань використовуємо команду gradlew wrapper (Лістинг 3.9).

Лістинг 3.9 — Перегляд доступних завдань

```
gradle build
# alternatively speedup second grandle build by holding it in memory
# gradle build –daemon
```

Ця команда створює в папці збірки вихід збірки Gradle. За замовчуванням збірка Gradle створює два файли .apk у папці build/outputs/apk.

Для створення та запуску модульних тестів на JVM використовуємо команду gradle test.

Для створення та запуску інструментальних тестів на пристрої Android використовуємо команду gradle connectedCheck.

3.4 Підключення баз даних

База даних SQLite — це база даних з відкритим вихідним кодом, яка надається в Android і використовується для зберігання даних у пристрої користувача у формі текстового файлу. Ми можемо виконувати багато операцій з цими даними, наприклад додавати нові дані, оновлювати, читати та видаляти ці дані. SQLite — це автономна база даних, яка локально зберігається на пристрої користувача, і не потрібно створювати жодних з'єднань для підключення до цієї бази даних. Дані зберігаються в базі даних SQLite у вигляді таблиць. Коли дані зберігаються в базі даних SQLite, вони впорядковуються у формі таблиць, схожих на таблиці Excel.

Нижче наведено кілька важливих методів, які будуть використовуватись в інтеграції бази даних SQLite в даному Android застосунку (див. табл. 3.2).

Таблиця 3.2 — Важливі методи в базі даних SQLite

Метод	Опис
getColumnNames()	Цей метод використовується для отримання масиву імен стовпців нашої таблиці SQLite.
getCount()	Цей метод повертає кількість рядків у курсорі.
isClosed()	Цей метод повертає логічне значення, коли наш курсор закрито.
getColumnCount()	Цей метод повертає загальну кількість стовпців у нашій таблиці.
getColumnName(int columnIndex)	Цей метод поверне ім'я стовпця, коли ми передали в нього індекс нашого стовпця.
getColumnIndex(String columnName)	Цей метод поверне індекс нашого стовпця з назви стовпця.

Додаємо дозволи на доступ до сховища у файлі AndroidManifest.xml (див. Лістинг 3.10)

Лістинг 3.10 — Дозвіл на доступ до сховища

```
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Далі переходимо до файлу res > layout > activity_main.xml і додаємо наведений нижче код (Лістинг 3.11).

Лістинг 3.11 — Код для роботи з базами даних в activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
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"
android:orientation="vertical"
tools:context=".MainActivity">
<!--Edit text to enter course name-->
<EditText
android:id="@+id/idEdtCourseName"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:hint="Enter course Name" />
```

```
<!--edit text to enter course duration-->
<EditText
    android:id="@+id/idEdtCourseDuration"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:hint="Enter Course Duration" />
<!--edit text to display course tracks-->
```

Далі створюємо новий клас Java для виконання операцій SQLite. Переходимо до файлу java > Ccounter > права кнопка миші > створити > клас Java і називаємо його як DBHandler і додаємо до нього наведений нижче код (Лістинг 3.12).

Лістинг 3.12 — Клас Java для виконання операцій SQLite

```
public class DBHandler extends SQLiteOpenHelper {
private static final String DB_NAME = "coursedb";
private static final int DB_VERSION = 1;
private static final String TABLE_NAME = "mycourses";
    private static final String ID_COL = "id";
    private static final String NAME_COL = "name";
private static final String DURATION_COL = "duration";
    private static final String DESCRIPTION_COL = "description";
private static final String TRACKS_COL = "tracks";
public DBHandler(Context context) {
```

```
    super(context, DB_NAME, null, DB_VERSION);
}

    @Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + TABLE_NAME + " ("
            + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + NAME_COL + " TEXT,"
            + DURATION_COL + " TEXT,"
            + DESCRIPTION_COL + " TEXT,"
            + TRACKS_COL + " TEXT)";
    db.execSQL(query);
}

    public void addNewCourse(String courseName, String courseDuration, String
courseDescription, String courseTracks) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(NAME_COL, courseName);
    values.put(DURATION_COL, courseDuration);
    values.put(DESCRIPTION_COL, courseDescription);
    values.put(TRACKS_COL, courseTracks);
    db.insert(TABLE_NAME, null, values);
    db.close();
}
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```

4 ТЕСТУВАННЯ ОДЕРЖАНИХ РЕЗУЛЬТАТІВ

4.1 Тестування застосунку

Було протестовано здатність інформаційної системи моніторингу рахувати кроки, пройдені користувачем та визначення швидкості при ходьбі за допомогою мобільного телефону із вімкненою службою GPS. Результат показано на рисунку 4.1.

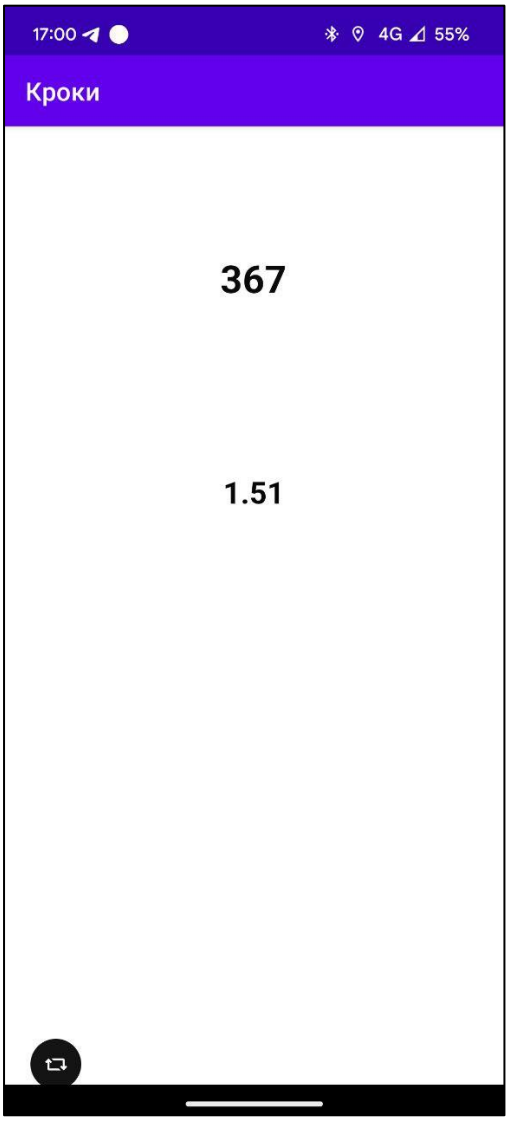

Рисунок 4.1 — Результат підрахунку кроків та визначення швидкості

Дана функція працювати лише на пристроях, які мають у своїй специфікації відповідні GPS модулі.

Інший функціонал мобільного застосунку було протестовано в середовищі розробки Android Studio за допомогою віртуального девайсу (див. рис. 4.2).

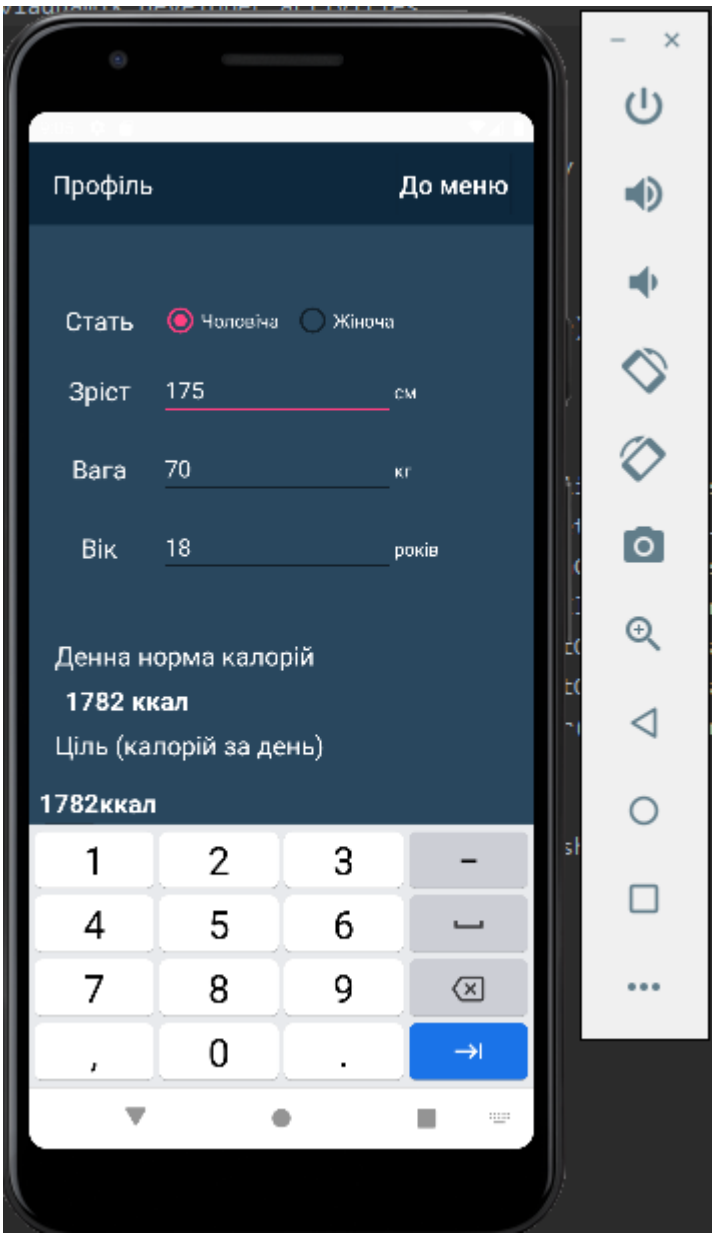

Рисунок 4.2 — Тестування функціонування за допомогою віртуального девайсу

4.2 Інструкція користувача

Користувач має володіти мобільним пристроєм під керуванням операційної системи Android не нижче версії 4.1. Додаток інсталюється на мобільний пристрій користувача через відкриття “apk” файлу та його автоматичного налаштування.

Користування функціоналом інформаційної системи розпочинається з початкового екрану «Меню». Тут знаходяться кнопки із посиланнями на усі інші екрани (див. рис. 4.3).

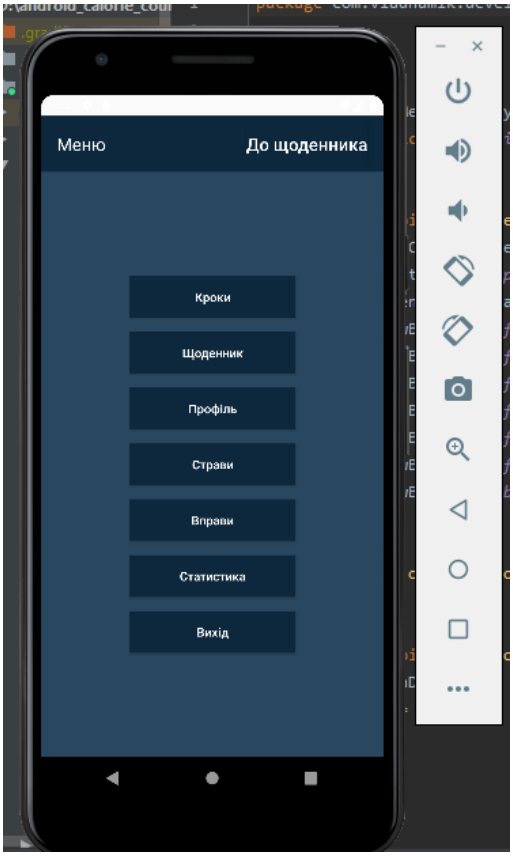

Рисунок 4.3 — Вигляд екрану «Меню»

На екрані «Щоденник» користувач системи заповнює дані щодо спожитих та витрачених калорій (рис. 4.4).

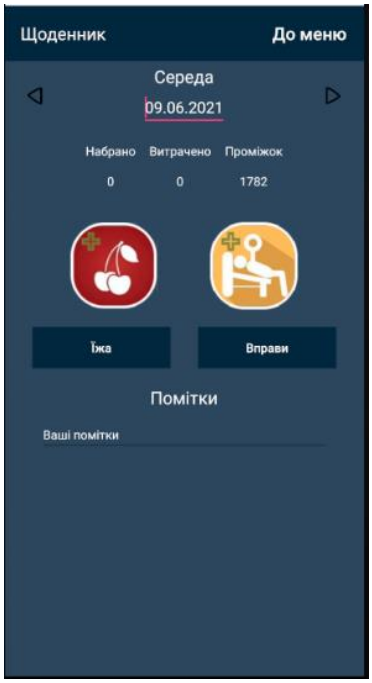

Рисунок 4.4 — Вигляд екрану «Щоденник»

На екрані “Страви” знаходиься створена база даних різних продуктів із заздалегіть підрахованими калоріями. Дану базу можна поповнювати та змінювати вже збережені дані (рис. 4.5).

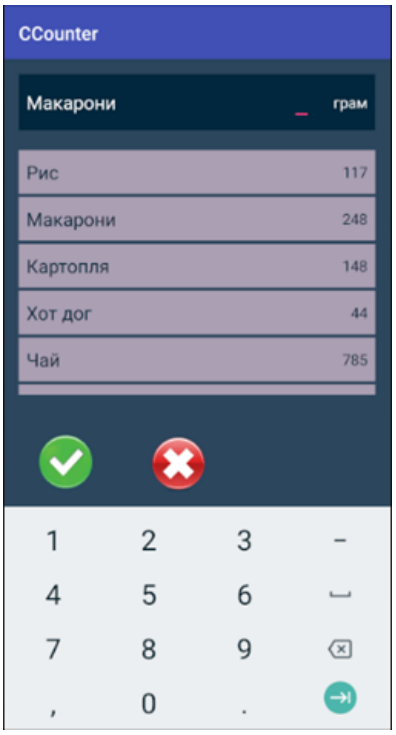

Рисунок 4.5 — Екран вибору страв

Щоб додати виконану фізичну вправу, потрібно натиснути на жовту кнопку. Потім потрібно вибрати вправу із списка та вказати час та кількість виконання. Для підтвердження натиснути на зелену кнопку. Для відміни натиснути червону кнопку (рис. 4.6).

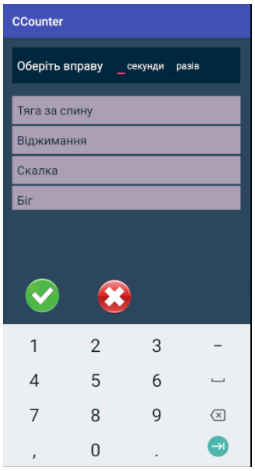

Рисунок 4.6 – Екран вибору вправ

Для переключення днів натиснути на поле із датою та вибрати потрібний день у календарі, що з’явився. Найнижче знаходиться поле для поміток користувача

На екрані «Профіль» користувач вносить свої дані для подальшого обрахунку калорій. У поля вік, зріст та вага користувач вводить числа, а навпроти статі вибирає один із двох варіантів (рис. 4.7).

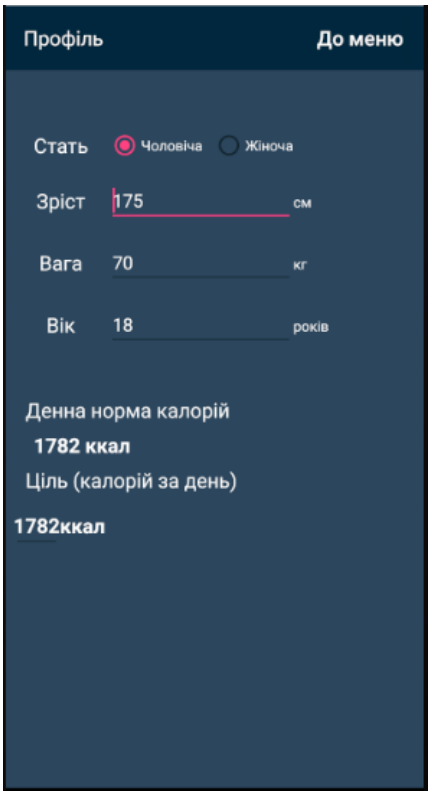

Рисунок 4.7 – Вигляд екрану «Профіль»

На екрані «Страви» міститься список страв користувача. Для видалення страви потрібно натиснути на червоний хрест, вибрати страву та повторно натиснути на ту саму кнопку (рис. 4.8).

Щоб внести зміни в існуючі страви, потрібно натиснути на зелений олівець, вибрати страву та повторно натиснути на кнопку. З’явиться екран для редагування страви, де можна змінити кількість калорій на 100 грам. А щоб створити нову страву, потрібно натиснути на зелений хрестик. Тоді з’явиться такий самий екран, що був для редагування (рис. 4.9).

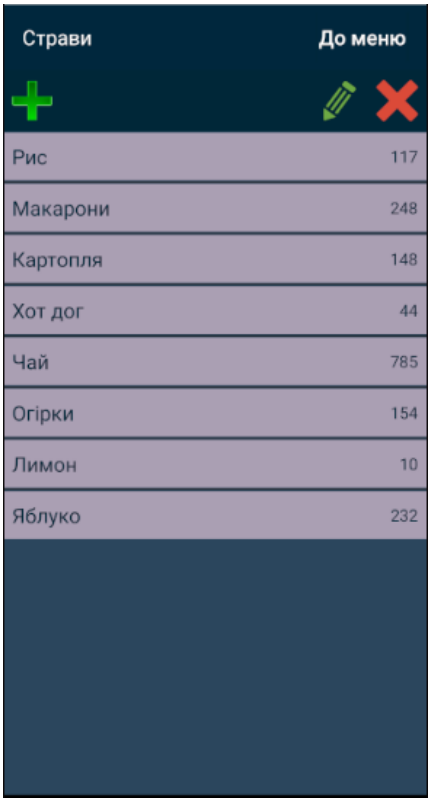

Рисунок 4.8 – Вигляд екрану «Страви»

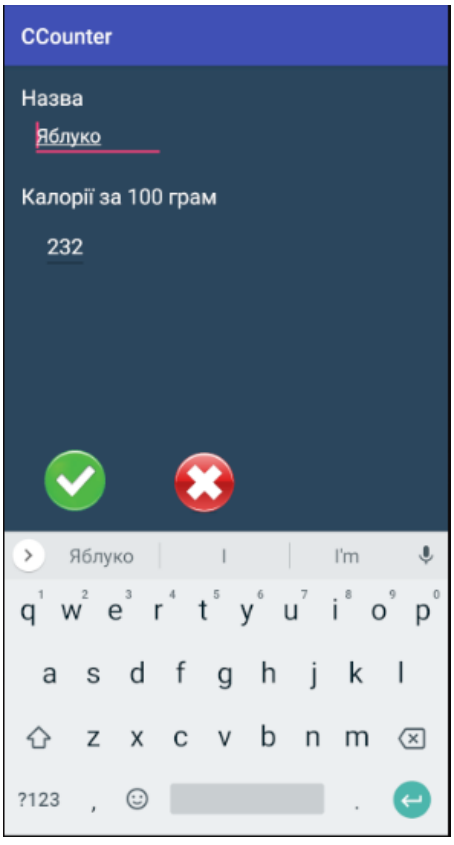

Рисунок 4.9 – Вигляд екрану створення та редагування страв

На екрані «Вправи» міститься список фізичних вправ користувача. Для видалення страви потрібно натиснути на червоний хрест, вибрати вправу та повторно натиснути на ту саму кнопку (рис. 4.10).

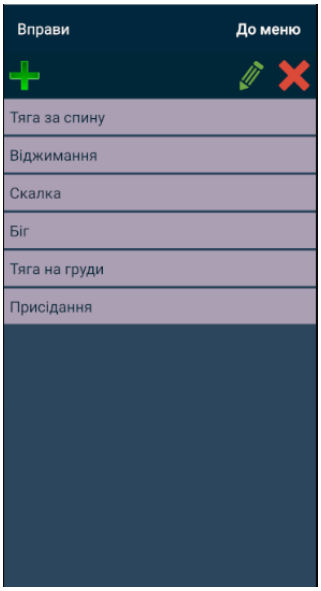

Рисунок 4.10 – Вигляд екрану «Вправи»

Щоб внести зміни в існуючі фізичні вправи, потрібно натиснути на зелений олівець, вибрати вправу та повторно натиснути на кнопку. З’явиться екран для редагування вправи, де можна змінити формулу для розрахунку витрачених калорій. А щоб створити нову вправу, потрібно натиснути на зелений хрестик. Тоді з’явиться такий самий екран, що був для редагування (рис. 4.11).

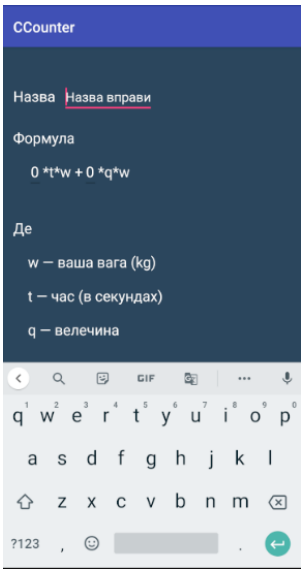

Рисунок 4.11 – Вигляд екрану створення та редагування фізичних вправ

На екрані «Статистика» демонструються спожиті калорії за проміжок часу. Кожна страва представляється окремо для зручного порівняння (рис. 4.12)

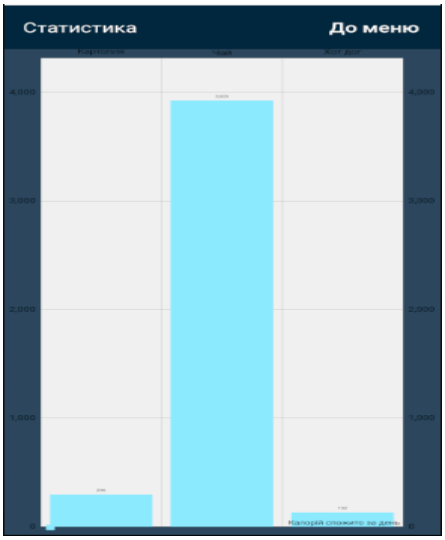

Рисунок 4.12 – Екран «Статистика»

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового інформаційної системи моніторингу фізіологічного стану людину, для відслідковування ваги та спожитих калорій, яка має назву Ccounter. Через пандемію коронавірусу люди почали ретельніше слідкувати за своїм здоров'ям і шукати для цих цілей різні застосунки, які б були завжди під рукою. Особливістю програми є те, що даний андроїд-застосунок має базу вправ та страв, які користувач може створювати сам та при необхідності змінювати у буль-яки момент. Також присутній профіль користувача із потрібною для застосунку інформацією (ріст, вага та стать) та щоденник, де ведуться записи. Аналогом може бути 7 Minute Workout Pro — за ціною 74,99 грн. або Home Workouts No Equipment Pro — за ціною 43,99 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	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 роботоздат-ність продукту в реальних умовах

Продовження таблиці 5.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п 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 властивості 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 .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на 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 із 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
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

Продовження таблиці 5.1

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

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	2
Наявність аналогів на ринку	3	3	3
Цінова політика	3	3	3
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	3	2	3
Конкурентоспроможність	3	3	3
Фахівці з технічної і комерційної реалізації	3	3	2
Фінансування	2	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	3	3	4
Сума	37	38	37
Середньоарифметична сума балів	(37+38+37) / 3 = 37,33		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є вище середнього, що досягається за рахунок того, що програмний продукт є інформаційною системою моніторингу фізіологічного стану людину, для відслідковування ваги та спожитих калорій, яка має назву Ccounter. Особливістю програми є те, що даний андроїд-застосунок має базу вправ та страв, які користувач може створювати сам та при необхідності змінювати у буль-яки момент. Також присутній профіль користувача із потрібною для застосунку інформацією (ріст, вага та стать) та щоденник, де ведуться записи та відстежуються швидкість та кроки.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою (5.1)

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{5.1}$$

де M — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів в місяці, 23 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	33000	1434,78	34	48782,609
Програміст	30000	1304,35	34	44347,826
Всього				93130,43

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.3 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання

Додаткова заробітна плата прийнято розраховувати як 14,35 % від основної заробітної плати розробників та робітників (5.2)

$$З_д = З_о \cdot 14{,}35\ \% / 100\ \%\ , \qquad (5.2)$$

$$З_д = (93130{,}43 \cdot 14{,}35\ \% / 100\ \%\ ) = 13364{,}22\ \ (\text{грн.})$$

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати (5.3)

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$$Н_з = (93130{,}43 + 13364{,}22) \cdot 22\ \% / 100\ \% = 23428{,}82\ (\text{грн.})$$

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.4 Амортизація обладнання, яке використовувалось для проведення розробки

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою (5.4)

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 27000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,48 міс. (5.5)

$$A_{обл} = \frac{27000}{2} \times \frac{1{,}78}{12} = 1663{,}04 \ грн. \quad (5.5)$$

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою (5.6)

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,45 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,45 \cdot 8 \cdot 34 \cdot 6,2 = 682,992$ (грн.)

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.4. Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів (Android Studio) є безкоштовною, то $В_{нем.ак.}$ = 0 грн.

Таблиця 5.4 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія (Acer Aspire E5-575G, Xiaomi Redmi Note 11 4/128)	27000	2	1,48	1663,043
Офісне обладнення (меблі)	21000	4	1,48	646,739
Приміщення	810000	20	1,48	4989,130
Всього				7298,91

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з

регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою (5.6)

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,45 кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,45 \cdot 8 \cdot 34 \cdot 6,2 = 682,992 \text{ (грн.)}$$

5.5 Інші витрати та загальновиробничі 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» розраховуються як 50…100% від суми основної заробітної плати дослідників (5.7)

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$I_в = 93130,43 * 90\% / 100\% = 83817,39 \text{ (грн.)}$$

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 та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників (5.8)

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

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 93130,43 * 130\ \% / 100\ \% = 121070 \text{ (грн.)}$$

5.6 Витрати на проведення науково-дослідної роботи

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 93130,43+13364,22+23428,82+7298,91+682,99+83817,39+$$

$$+121070 = 342792,34 \text{ грн.}$$

5.7 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою (5.9)

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \qquad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 342792,34 / 0,5 = 685585 \text{ грн.}$$

5.8 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним 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.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої

комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.9 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних (5.10)

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 20 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 5 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 250000 шт., протягом другого року – на 200000 шт., протягом третього року на 150000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*5 + (20 + 5)*250000)* 0,8333* 0,4) * (1 - 0,18) = 1366666,61 грн.

$\Delta П_2$ = (0*5 + (20 + 5)*(250000+200000)* 0,8333* 0,4) * (1 - 0,18) = 3074999,87 грн.

$\Delta П_3$ = (0*5+(20+5)*(250000+200000+150000)*0,833*0,4)*(1-0,18) = 4099999,83 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 8541666,33 грн.

5.10 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки (5.11)

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках);

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (1366666,612/(1+0,1)^1)+(3074999,877/(1+0,1)^2)+(4099999,836/(1+0,1)^3) = 1242424,19 + 2541322,212 + 3080390,56 = 6864136,966 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу (5.12)

$$PV = k_{інв} * ЗВ, \qquad (5.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 685585 = 1371169{,}35 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме (5.13)

$$E_{абс} = ПП - PV, \tag{5.13}$$

$$E_{абс} = 6864136{,}966 - 1371169{,}35 = 5492967{,}62 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу (5.14)

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.14}$$

де $T_ж$ — життєвий цикл наукової розробки, роки.

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою (5.15)

$$\tau = d + f, \tag{5.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19 .$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою (5.16)

$$T_{ок} = \frac{1}{E_в}, \tag{5.16}$$

де $T_{ок}$ = 1 / 0,711 = 1,41 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 1,41 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 685585 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 1,41 роки.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи був проведений аналіз існуючих систем та методів моніторингу фізіологічного людини, а також аналіз готових рішень від різних компаній. Була доведена актуальність дослідження існуючих систем та методів, проведене обґрунтування розробки методу для інформаційної системи моніторингу фізіологічного стану людини, визначена задача роботи.

На основі результатів аналізу була розроблена інформаційна система моніторингу фізіологічного стану людини та проведений аналіз розробленого методу.

У ході виконання роботи було проведене тестування використання нової інформаційної системи моніторингу фізіологічного стану людини у різних умовах та доведена її конкурентноспроможність.

Для практичного застосування було розроблено спеціальний Android застосунок, який спрощує збір необхідних даних.

За результатами магістерської кваліфікаційної роботи можна зробити висновок, що поставлена мета - аналіз сучасного стану і тенденцій розвитку світових розробок у сфері моніторингу фізіологічного стану людини та розробка програмного продукту для відстеження необхідних показників була виконана.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Information System for Monitoring Personal Health Status Based on Big Data Analysis [Електронний ресурс] – Режим доступу до ресурсу: https://ceur-ws.org/Vol-2917/paper44.pdf

2. Internet of things-enabled real-time health monitoring system using deep learning [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s00521-021-06440-6

3. Remote patient monitoring: a comprehensive study [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s12652-017-0598-x

4. Medical Data Processing and Analysis for Remote Health and Activities Monitoring [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-030-16272-6_7

5. A Detailed Research on Human Health Monitoring System Based on Internet of Things [Електронний ресурс] – Режим доступу до ресурсу: https://www.hindawi.com/journals/wcmc/2021/5592454/

6. eValuate - A Sports Analytics mHealth App [Електронний ресурс] – Режим доступу до ресурсу: https://www.diva-portal.org/smash/get/diva2:1451496/FULLTEXT01.pdf

7. Physical activity [Електронний ресурс] – Режим доступу до ресурсу: https://www.who.int/news-room/fact-sheets/detail/physical-activity

8. A Focused Review of Smartphone Diet-Tracking Apps: Usability, Functionality, Coherence With Behavior Change Theory, and Comparative Validity of Nutrient Intake and Energy Estimates [Електронний ресурс] – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6543803/

9. Measurement Methods for Physical Activity and Energy Expenditure: a Review [Електронний ресурс] – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5426207/

10. Multi-sensory Cyber-Physical Therapy System for Elderly Monitoring [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-39949-2_9

11. Calorie counting smart phone apps: Effectiveness in nutritional awareness, lifestyle modification and weight management among young Indian adults [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/333777044_Calorie_counting_smart_phone_apps_Effectiveness_in_nutritional_awareness_lifestyle_modification_and_weight_management_among_young_Indian_adults

12. How are Calories Calculated on the ELEMNT and Wahoo Fitness App? [Електронний ресурс] – Режим доступу до ресурсу: https://support.wahoofitness.com/hc/en-us/articles/204280754-How-are-Calories-Calculated-on-the-ELEMNT-and-Wahoo-Fitness-App-

13. Operating System Market Share Worldwide [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share

14. What is an operating system? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/operating-system-OS

15. Windows [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerhope.com/jargon/w/windows.htm

16. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerhope.com/jargon/l/linux.htm

17. macOS [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerhope.com/jargon/m/macos.htm

18. MacOS [Електронний ресурс] – Режим доступу до ресурсу: https://www.britannica.com/technology/Mac-OS

19. What is an Android Operating System & Its Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.elprocus.com/what-is-android-introduction-features-applications/

20. What Is iOS? – A Brief Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.emizentech.com/blog/what-is-ios.html

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..

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
"Комп'ютерна система моніторингу регіонального радіомовлення"
08-23.МКР.027.00.000 ТЗ

Науковий керівник: доцент к.т.н.
_____ Тарновський М.Г.

Студент групи 2КІ-21м
_____ Смачило А.Р.

ВНТУ 2022

1 Підставою для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Важливим є актуальність дослідження у напрямку магістерської роботи, яка обумовлена тим, що із поширенням небезпечного вірусу COVID-19 та інших небезпечних хвороб люди почали дельніше вивчати питання охорони власного здоров'я. Через дані події виріс попит на різні системи для моніторингу здоров'я, додатки для відстежування прогресу в різних видах та все, що пов'язане із зміцненням імунітету. З кожним днем вчені відкривають нові методи запобігання різних хвороб. Для ефективного лікування недугів потрібно знати детальну інформацію про пацієнта та його фізичний або фізіологічний стан.;

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розширення функціоналу заповнення даних при моніторингу фізіологічного стану людини;

2.2 Призначення розробки — інформаційна система, яка здатна відстежувати та зберігати показники фізичної активності для запобігання можливим захворюванням.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих принципів та технологій інформаційних систем моніторингу фізіологічного стану людини;

3.2 Розробка структури та схеми компонентів застосунку для інформаційної системи;

3.3 На основі створеної структури та схеми компонентів розробити застосунок для коректного функціонування інформаційної системи;

3.4 Проведення необхідних тестувань розробленого застосунку для перевірки правильності виконання поставлених задач.

4 Вимоги до виконання МКР

Головна вимога — розробити інформаційну систему моніторингу фізіологічного стану людини, використовуючи сучасні та актуальні методи й засоби.

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 використання інформаційних технологій в медецині та відомих аналогів			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Дослідження методів вимірювання фізіологічних показників та вибір компонентів розробки			Розділ 2
3	Розробка інформаційної системи моніторингу фізіологічного стану людини			Розділ 3
4	Тестування одержаних результатів			Розділ 4
5	Підготовка економічної частини			Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації			ПЗ, графіч. матеріал і презентація
7	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат			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, довідка про відповідність оформлення МКР діючим вимогам.

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 оформлювання 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»;

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

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами 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 компонентів застосунку

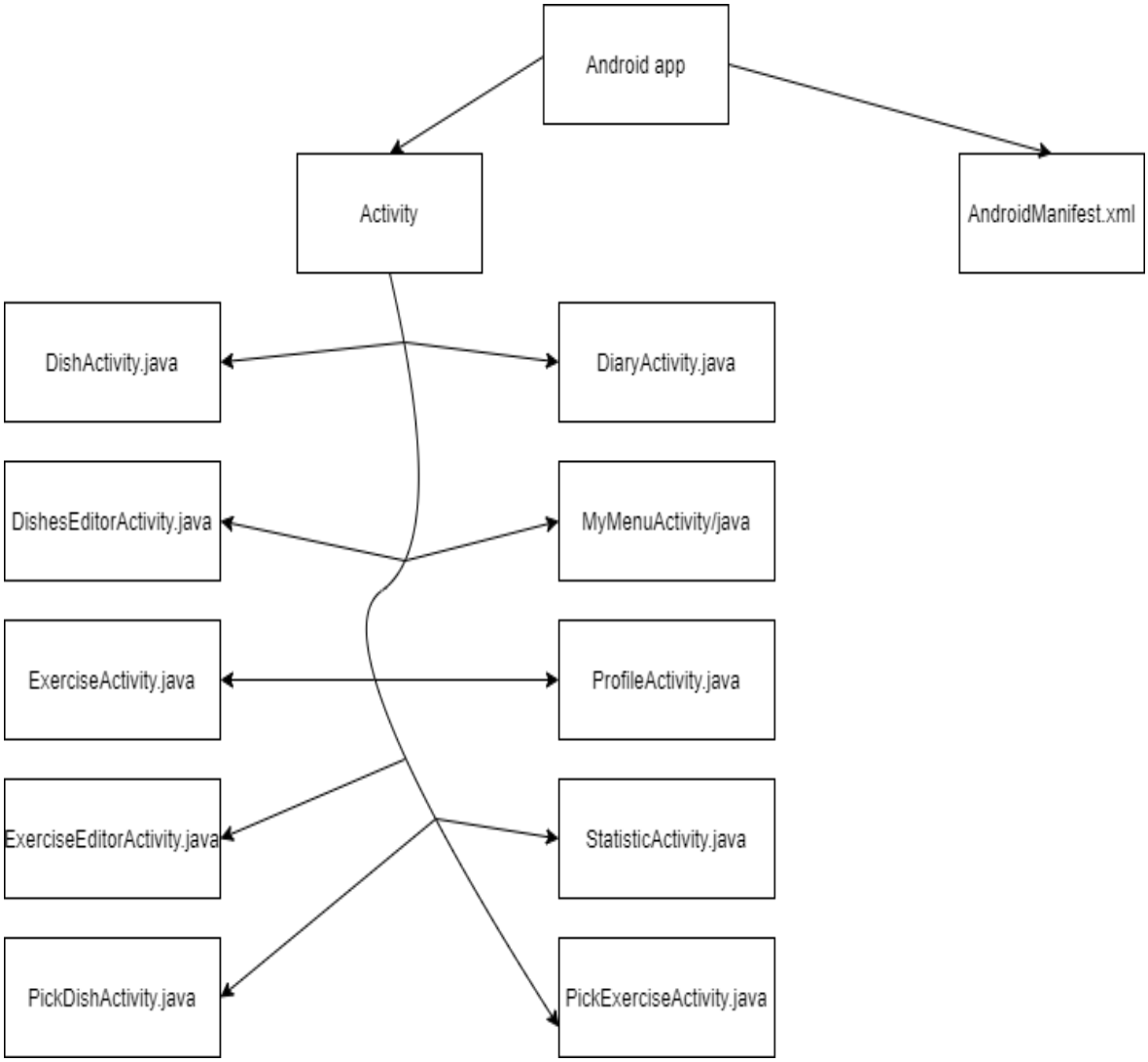

Рисунок Б.1 — Схема компонентів застосунку

ДОДАТОК В

Файли Java

Лістинг В.1 — DiaryActivity.java

```
public class DiaryActivity extends AppCompatActivity {
    Profile profile;
    Day viewDay;
    DataBase db;
    SimpleAdapter dishesAdapter;
    ArrayList<Map<String, Object>> dishesData;
    SimpleAdapter exercisesAdapter;
    ArrayList<Map<String, Object>> exercisesData;
    private static final int DATE_DIALOG = 1;
    private static final int VIEW_DISHES_DIALOG = 2;
    private static final int VIEW_EXERCISES_DIALOG = 3;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_diary);
        findViewById(R.id.from_diary_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
        findViewById(R.id.diary_set_date).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDialog(DATE_DIALOG);
            }
        });
```

```
        findViewById(R.id.diary_left_arrow).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changeViewDay(viewDay.getPreviousDay(db).getDate());
            }
        });
        findViewById(R.id.diary_right_arrow).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changeViewDay(viewDay.getNextDay(db).getDate());
            }
        });
        findViewById(R.id.diary_dish_add_button).setOnClickListener(new
ToWindowOnClick(this, PickDishActivity.class) {
            @Override
            public void onClick(View v) {
                PickDishActivity.date = viewDay.getDate();
                super.onClick(v);
            }
        });
        findViewById(R.id.diary_exercise_add_button).setOnClickListener(new
ToWindowOnClick(this, PickExerciseActivity.class) {
            @Override
            public void onClick(View v) {
                PickExerciseActivity.date = viewDay.getDate();
                super.onClick(v);
            }
        });
```

```
        profile = Profile.getProfile(this);
        db = DataBase.getDataBase(this);
        setViewDay(new MyDate()); // set current date
        dishesData =
DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate()));
        String[] from = new String[]{DataBase.DISH_COLUMN_NAME,
DataBase.DAYS_DISH_COLUMN_WEIGHT};//columns names
        int[] to = new int[]{R.id.db_item_name, R.id.db_item_right_text}; // places to
write (View id)
        dishesAdapter = new SimpleAdapter(this, dishesData, R.layout.database_item,
from, to);
        exercisesData =
DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.getDate()));
        from = new String[]{DataBase.EXERCISE_COLUMN_NAME,
DataBase.DAYS_EXERCISE_COLUMN_QUANTITY};//columns names
        exercisesAdapter = new SimpleAdapter(this, exercisesData,
R.layout.database_item, from, to);
    }
    @Override
    protected Dialog onCreateDialog(int id) {
        AlertDialog.Builder adb;
        switch (id) {
            case DATE_DIALOG:
                Calendar calendar = Calendar.getInstance();
                calendar.setTime(viewDay.getDate());
                return new DatePickerDialog(this, myCallBack,
calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH));
            case VIEW_DISHES_DIALOG:
                adb = new AlertDialog.Builder(this);
```

```
            adb.setTitle(getString(R.string.eaten_dishes));
            adb.setAdapter(dishesAdapter, null);
            return adb.create();
        case VIEW_EXERCISES_DIALOG:
            adb = new AlertDialog.Builder(this);
            adb.setTitle(getString(R.string.exercises_per_day));
            adb.setAdapter(exercisesAdapter, null);
            return adb.create();
      }
      return super.onCreateDialog(id);
   }
   @Override
   protected void onPrepareDialog(int id, Dialog dialog) {
      super.onPrepareDialog(id, dialog);
      switch (id) {
         case DATE_DIALOG:
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(viewDay.getDate());
            ((DatePickerDialog) dialog).updateDate(calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH));
            break;
         case VIEW_DISHES_DIALOG:
            break;
         case VIEW_EXERCISES_DIALOG:
            break;
      }
   }
   DatePickerDialog.OnDateSetListener myCallBack = new
DatePickerDialog.OnDateSetListener() {
      public void onDateSet(DatePicker view, int year, int monthOfYear,
```

```
                    int dayOfMonth) {
            String newDate = dayOfMonth + "." + (monthOfYear + 1) + "." + year;
            try {
                changeViewDay(new MyDate(Day.format.parse(newDate)));
            } catch (ParseException e) {
                Toast.makeText(DiaryActivity.this, getString(R.string.invalid_date),
Toast.LENGTH_SHORT).show();
            } catch (NullPointerException e) {
                Toast.makeText(DiaryActivity.this, getString(R.string.invalid_year),
Toast.LENGTH_SHORT).show();
            }
        }
    };
    public void onViewDayDishes(View v) {
        showDialog(VIEW_DISHES_DIALOG);
    }

    public void onViewDayExercises(View v) {
        showDialog(VIEW_EXERCISES_DIALOG);
    }
    private void changeViewDay(MyDate date) {
        saveViewDay();
        setViewDay(date);
    }
    private void saveViewDay()//закончить
    {
        viewDay.setRecord(((EditText)
findViewById(R.id.diary_record)).getText().toString());
        for (Map<String, Object> dish : dishesData)
```

```
            viewDay.addDish(db.getDish((String)
dish.get(DataBase.DISH_COLUMN_NAME)), Integer.parseInt((String)
dish.get(DataBase.DAYS_DISH_COLUMN_WEIGHT)));
        for (Map<String, Object> exercise : exercisesData)
            viewDay.addExercise(db.getExercise((String)
exercise.get(DataBase.EXERCISE_COLUMN_NAME)), Integer.parseInt((String)
exercise.get(DataBase.DAYS_EXERCISE_COLUMN_TIME)),
Integer.parseInt((String)
exercise.get(DataBase.DAYS_EXERCISE_COLUMN_QUANTITY)));
        db.saveDay(viewDay);
    }
    private void setViewDay(MyDate date) {
        setViewDay(Day.getDayByDate(db, date));
    }
    private void setViewDay(Day day) {
        viewDay = day;
        if (dishesData != null) {
            dishesData.clear();
dishesData.addAll(DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate(
))));
            dishesAdapter.notifyDataSetChanged();
        }
        if (exercisesData != null) {
            exercisesData.clear();
exercisesData.addAll(DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.get
Date())));
            exercisesAdapter.notifyDataSetChanged();
        }
```

```
        ((TextView)
findViewById(R.id.diary_weekday)).setText(Day.getDayOfWeekByDate(day.getDate()
));
        ((EditText)
findViewById(R.id.diary_set_date)).setText(Day.format.format(day.getDate()));
        ((EditText) findViewById(R.id.diary_record)).setText(day.getRecord());
        updateCaloriesRow();
    }
    protected void onRestart() {
        dishesData.clear();

dishesData.addAll(DataBase.cursorToArrayList(db.getAllDayDishes(viewDay.getDate(
))));
        dishesAdapter.notifyDataSetChanged();
        exercisesData.clear();
exercisesData.addAll(DataBase.cursorToArrayList(db.getAllDayExercises(viewDay.get
Date())));
        exercisesAdapter.notifyDataSetChanged();
        updateCaloriesRow();
        super.onRestart();
    }
    private void updateCaloriesRow() {
        Integer receivedCalories = viewDay.getReceivedCalories();
        Integer spentCalories = viewDay.getSpentCalories(profile.getWeight());
        ((TextView)
findViewById(R.id.diary_calorie_get)).setText(receivedCalories.toString());
        ((TextView)
findViewById(R.id.diary_calorie_spend)).setText(spentCalories.toString());
        ((TextView) findViewById(R.id.diary_calorie_need)).setText(((Integer)
(profile.getAimCalorie() + spentCalories - receivedCalories)).toString());
```

```
        findViewById(R.id.diary_calorie_get).requestLayout();
        findViewById(R.id.diary_calorie_spend).requestLayout();
        findViewById(R.id.diary_calorie_need).requestLayout();
    }
    @Override
    protected void onDestroy() {
        saveViewDay();
        db.close();
        super.onDestroy();
    }
}
```

Лістинг В.2 — DishActivity.java

```
public class DishActivity extends AppCompatActivity{
    DataBase db;
    Map<String, Object> map;
    ArrayList<Map<String, Object>> data;
    SimpleAdapter sAdapter;
    long selectedElementId=-1;
    ListView listView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dish);
        findViewById(R.id.from_dishes_to_menu).setOnClickListener(new
ToWindowOnClickWithClosing(this, MyMenuActivity.class));
        db = DataBase.getDataBase(this);
        data=DataBase.cursorToArrayList(db.getDishes());
```

```
        String[] from = new String[] {DataBase.DISH_COLUMN_NAME,
DataBase.DISH_COLUMN_CALORIES_PER_100_GM }; // columns names
        int[] to = new int[] { R.id.db_item_name, R.id.db_item_right_text}; // places to
write (View id)
        sAdapter = new SimpleAdapter(this, data ,  R.layout.database_item, from, to);
        listView= findViewById(R.id.dish_list_view);
        listView.setAdapter(sAdapter);
        listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
                selectedElementId = id;
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, 1, 0, getString(R.string.order_by_alphabet));
        menu.add(0, 2, 1, getString(R.string.order_by_caloricity));
        menu.add(0, 3, 2, getString(R.string.order_by_id));
        return super.onCreateOptionsMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId())
        {
            case 1: DataBase.dishSortId=1;
                break;
```

```
            case 2: DataBase.dishSortId=2;
                break;
            case 3: DataBase.dishSortId=0;
                break;
        }
        data.clear();
        data.addAll(DataBase.cursorToArrayList(db.getDishes()));
        sAdapter.notifyDataSetChanged();
        return super.onOptionsItemSelected(item);
    }
    public void onDeleteDish(View view)
    {
        if(selectedElementId<0)
            Toast.makeText(this, getString(R.string.pick_dish),
Toast.LENGTH_SHORT).show();
        else
        {
            String
dishName=(String)data.get((int)selectedElementId).get(DataBase.DISH_COLUMN_N
AME);
            db.deleteDish(dishName);
            data.remove((int)selectedElementId);
            selectedElementId=-1;
            sAdapter.notifyDataSetChanged();
        }
    }
    public void onEditDish(View view)
    {
        if (selectedElementId<0) {
```

```
            Toast.makeText(this, getString(R.string.pick_dish),
Toast.LENGTH_SHORT).show();
            return;
        }
        Dish dish = new Dish();
        map=data.get((int)selectedElementId);
        dish.setName(map.get(DataBase.DISH_COLUMN_NAME).toString());
dish.setCalories(Integer.parseInt(map.get(DataBase.DISH_COLUMN_CALORIES_PE
R_100_GM).toString()));
        DishesEditorActivity.dish=dish;
        onCreateDish(view);
    }
    public void onCreateDish(View view)
    {
        Intent intent = new Intent(this, DishesEditorActivity.class);
        startActivity(intent); // transfer control to editor
        selectedElementId=-1;
    }
    @Override
    protected void onRestart() {
        data.clear();
        data.addAll(DataBase.cursorToArrayList(db.getDishes()));
        sAdapter.notifyDataSetChanged();
        super.onRestart();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        db.close(); // close DB
```

ДОДАТОК Г

Файли XML

Лістинг Г.1 — activity_dish.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    style="@style/background_style"
    tools:context="com.vladnamik.developer.activities.DishActivity">
    <RelativeLayout
        style="@style/header_style">
        <TextView
            style="@style/header_text"
            android:text = "@string/dishes">
        </TextView>
        <Button
            style="@style/header_button"
            android:id="@+id/from_dishes_to_menu"
            android:text="@string/to_menu">
        </Button>
    </RelativeLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:orientation="horizontal"
        android:background="@color/my_db_menu_color"
        android:id="@+id/db_dish_menu">
        <ImageButton
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:background="@null"
```

```
        android:scaleType="centerInside"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:layout_centerVertical="true"
        android:id="@+id/db_dish_add"
        android:onClick="onCreateDish"
        android:src="@drawable/add_item"
        android:contentDescription="@string/create" />
```

Лістинг Г.2 — activity_my_menu.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context="com.vladnamik.developer.activities.MyMenuActivity"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    style="@style/background_style">
    <RelativeLayout
      style="@style/header_style">
      <TextView
        style="@style/header_text"
        android:text = "@string/menu">
      </TextView>
      <Button
        style="@style/header_button"
        android:id="@+id/from_menu_to_diary"
        android:text="@string/to_diary">
      </Button>
    </RelativeLayout>
    <ScrollView
      android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:id="@+id/scroll_view_menu">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="horizontal">
    <View
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight=".16"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_weight=".1">
        <Button
            android:id="@+id/from_menu_to_diary_menu"
            style="@style/menu_button"
            android:text="@string/diary"
            >
        </Button>
```

ДОДАТОК Д

Gradle Scripts

Лістинг Д.1 — build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 27
    buildToolsVersion "27.0.3"
    defaultConfig {
        applicationId "com.vladnamik.developer"
        minSdkVersion 15
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}
dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    testImplementation 'junit:junit:4.12'
    api 'com.android.support:appcompat-v7:27.1.1'
    api 'com.android.support:design:27.1.1'
}
buildscript {
```

```
    repositories {
        mavenLocal()
        maven { url 'https://maven.google.com' }
        mavenCentral()
        jcenter()
        google()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.2'
    }
}
allprojects {
    repositories {
        mavenLocal()
        maven { url 'https://maven.google.com' }
        mavenCentral()
        jcenter()
        google()
    }
}
```

ДОДАТОК Е

Файли IML

Лістинг Е.1 — android_calorie_counter.iml

```
<?xml version="1.0" encoding="UTF-8"?>
<module external.linked.project.id=":app"
external.linked.project.path="$MODULE_DIR$"
external.root.project.path="$MODULE_DIR$/.." external.system.id="GRADLE"
external.system.module.group="MyApplication"
external.system.module.version="unspecified" type="JAVA_MODULE" version="4">
  <component name="FacetManager">
    <facet type="android-gradle" name="Android-Gradle">
      <configuration>
        <option name="GRADLE_PROJECT_PATH" value=":app" />
      </configuration>
    </facet>
    <facet type="android" name="Android">
      <configuration>
        <option name="SELECTED_BUILD_VARIANT" value="debug" />
        <option name="ASSEMBLE_TASK_NAME" value="assembleDebug" />
        <option name="COMPILE_JAVA_TASK_NAME"
value="compileDebugSources" />
        <afterSyncTasks>
          <task>generateDebugSources</task>
        </afterSyncTasks>
        <option name="ALLOW_USER_CONFIGURATION" value="false" />
        <option name="MANIFEST_FILE_RELATIVE_PATH"
value="/src/main/AndroidManifest.xml" />
        <option name="RES_FOLDER_RELATIVE_PATH" value="/src/main/res" />
```

```
      <option name="RES_FOLDERS_RELATIVE_PATH"
value="file://$MODULE_DIR$/src/main/res" />
      <option name="ASSETS_FOLDER_RELATIVE_PATH"
value="/src/main/assets" />
    </configuration>
   </facet>
  </component>
  <component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_7">
   <output url="file://$MODULE_DIR$/build/intermediates/classes/debug" />
   <output-test url="file://$MODULE_DIR$/build/intermediates/classes/test/debug" />
   <exclude-output />
   <content url="file://$MODULE_DIR$">
    <sourceFolder url="file://$MODULE_DIR$/build/generated/source/apt/debug"
isTestSource="false" generated="true" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/source/r/debug"
isTestSource="false" generated="true" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/source/aidl/debug"
isTestSource="false" generated="true" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/buildConfig/debug"
isTestSource="false" generated="true" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/source/rs/debug"
isTestSource="false" generated="true" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/res/rs/debug"
type="java-resource" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/res/resValues/debug"
type="java-resource" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/apt/androidTest/debug"
isTestSource="true" generated="true" />
```

```
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/r/androidTest/debug"
isTestSource="true" generated="true" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/aidl/androidTest/debug"
isTestSource="true" generated="true" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/buildConfig/androidTest/debug"
isTestSource="true" generated="true" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/source/rs/androidTest/debug"
isTestSource="true" generated="true" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/res/rs/androidTest/debug" type="java-
test-resource" />
    <sourceFolder
url="file://$MODULE_DIR$/build/generated/res/resValues/androidTest/debug"
type="java-test-resource" />
    <sourceFolder url="file://$MODULE_DIR$/build/generated/source/apt/test/debug"
isTestSource="true" generated="true" />
    <sourceFolder url="file://$MODULE_DIR$/src/debug/res" type="java-resource" />
    <sourceFolder url="file://$MODULE_DIR$/src/debug/resources" type="java-
resource" />
    <sourceFolder url="file://$MODULE_DIR$/src/debug/assets" type="java-resource"
/>
    <sourceFolder url="file://$MODULE_DIR$/src/debug/aidl" isTestSource="false" />
    <sourceFolder url="file://$MODULE_DIR$/src/debug/java" isTestSource="false"
/>
    <sourceFolder url="file://$MODULE_DIR$/src/debug/rs" isTestSource="false" />
```

<sourceFolder url="file://$MODULE_DIR$/src/debug/shaders" isTestSource="false" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/res" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/resources" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/assets" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/aidl" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/java" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/rs" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/testDebug/shaders" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/res" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/resources" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/assets" type="java-test-resource" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/aidl" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/java" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/rs" isTestSource="true" />
<sourceFolder url="file://$MODULE_DIR$/src/androidTestDebug/shaders" isTestSource="true" />

ДОДАТОК Ж

Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи: Інформаційна система моніторингу фізіологічного стану людини

Тип роботи: магістерська кваліфікаційна робота

(БДР, МКР)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 87.1% Схожість 12.9%

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 _____ Смачило А.Р.

(підпис) (прізвище, ініціали)

Керівник роботи _____ Тарновський М.Г.

(підпис) (прізвище, ініціали)