

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмних модулів мобільної системи для контролю
дихання, медитації та релаксу»

Виконав: студент IV курсу
групи 4ПІ-216 спеціальності
121 – Інженерія програмного
забезпечення

(шифр і назва напрямку підготовки, спеціальності)

Лещенко І. О.

(прізвище та ініціали)

Керівник: к.т.н., доц. кафедри ПЗ Ракитянська Г. Б.

(прізвище та ініціали)

Рецензент: к.т.н., доц. кафедри ОТ Городецька О. С.

(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н.

06» червня 2025 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

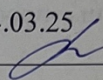
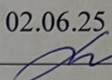
ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

ЗАВДАННЯ НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лещенку Івану Олександровичу

1. Тема роботи – розробка програмних модулів мобільної системи для контролю дихання, медитації та релаксу.
Керівник роботи: Ракитянська Ганна Борисівна., к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.
2. Строк подання студентом роботи: 2 червня 2025.
3. Вихідні дані до роботи: середовище розробки – Xcode, Visual Studio Code; мови розробки – Swift; фреймворки – SwiftUI, Flutter; операційна система – macOS Ventura / Sonoma; емулятор мобільного пристрою – iPhone 14 Pro з операційною системою 17.0.
4. Зміст текстової частини: вступ, аналіз предметної області розвитку засобів забезпечення ментального здоров'я та постановка задач дослідження, розробка методу, моделей та алгоритмів програмного продукту, розробка програмних засобів мобільної системи, тестування роботи мобільної системи, список використаних джерел, додатки, ілюстративна частина.
5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської роботи, тема; актуальність розробки; мета дослідження; задачі дослідження; об'єкт та предмет дослідження; новизна; практична цінність; існуючі аналоги; аналіз аналогів; алгоритм роботи системи; схеми інтерфейсу; реєстрація; налаштування; медитація; контроль дихання; фокус; висновки.

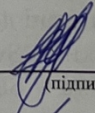
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ракитянська Г. Б., к.т.н., доцент кафедри ПЗ	24.03.25 	02.06.25 

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програм для контролю дихання, медитації та релаксу	25.03.25 – 10.04.25	<i>Виконано</i>
2	Розробка методів, моделі та алгоритмів роботи програми	11.04.25 – 18.04.25	<i>Виконано</i>
3	Розробка програмного забезпечення	19.04.25 – 29.04.25	<i>Виконано</i>
4	Тестування програми	30.04.25 – 15.05.25	<i>Виконано</i>
5	Оформлення матеріалів до захисту БКР	16.05.25 – 30.05.25	<i>Виконано</i>

Студент  Лещенко І. О.
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи  Ракитянська Г. Б.
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота містить 69 сторінок формату А4, на яких подано 34 рисунки і 4 таблиці. Список використаних джерел складається з 20 найменувань.

У бакалаврській кваліфікаційній роботі проведено аналіз стану і перспектив розвитку мобільних застосунків, спрямованих на підтримку дихальних практик, медитації та релаксації. Також досліджено існуючі програмні рішення, виявлено їх переваги та недоліки, а також обґрунтовано доцільність створення власного мобільного застосунку, адаптованого до потреб користувачів iOS-платформи.

У роботі обґрунтовано вибір мови програмування, середовища розробки та технологій, що були використані під час створення застосунку. Результатом є розроблений програмний продукт мобільний застосунок для операційної системи iOS, реалізований за допомогою мови програмування Swift і фреймворку SwiftUI. У якості основного середовища розробки використовувався Xcode.

Отримані результати можуть бути застосовані для підвищення концентрації, зменшення рівня стресу та загального покращення психоемоційного стану користувачів за допомогою регулярних дихальних вправ, медитацій і нагадувань про релаксацію.

Ключові слова: мобільний застосунок, дихання, медитація, релаксація, Swift, SwiftUI, iOS, інтерфейс користувача.

ABSTRACT

The bachelor's thesis consists of 69 A4 pages, containing 34 figures and 4 tables. The list of references includes 20 sources.

The thesis presents an analysis of the current state and development prospects of mobile applications aimed at supporting breathing practices, meditation, and relaxation. Existing software solutions were studied, their strengths and weaknesses were identified, and the feasibility of developing a custom mobile application tailored to the needs of iOS platform users was justified.

The work substantiates the choice of programming language, development environment, and technologies used during the application development process. The result is a software product a mobile application for the iOS operating system, implemented using the Swift programming language and the SwiftUI framework. Xcode was used as the main development environment.

The results of this work can be applied to improve focus, reduce stress levels, and enhance the overall psycho-emotional state of users through regular breathing exercises, meditation sessions, and relaxation reminders.

Keywords: mobile application, breathing, meditation, relaxation, Swift, SwiftUI, iOS, user interface.

ЗМІСТ

ВСТУП.....	4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗВИТКУ ЗАСОБІВ ЗАБЕЗПЕЧЕННЯ МЕНТАЛЬНОГО ЗДОРОВ'Я ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ....	7
1.1 Аналіз сучасного стану мобільних технологій для контролю дихання, медитації та релаксу	7
1.2 Порівняльний аналіз аналогів	7
1.3 Аналіз методів розв'язання задачі	Ошибка! Закладка не определена.
1.4 Постановка задач дослідження.....	Ошибка! Закладка не определена.
1.5 Висновки.....	Ошибка! Закладка не определена.
2 РОЗРОБКА МЕТОДІВ, МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ	16
2.1 Аналіз вхідних даних системи	16
2.2 Розробка методу реалізації в мобільному застосунку модуля контролю дихання	16
2.3 Розробка методу реалізації в мобільному застосунку модуля медитацій.....	19
2.4 Висновки.....	27
3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ МОБІЛЬНОЇ СИСТЕМИ	28
3.1 Варіантний аналіз та обґрунтування засобів програмного додатку	28
3.2 Розробка модуля для авторизації та реєстрації користувача	31
3.3 Розробка модуля налаштувань	36
3.4 Розробка модуля контролю дихання	38
3.5 Розробка модуля медитацій	Ошибка! Закладка не определена.
3.6 Висновки.....	Ошибка! Закладка не определена.
4 ТЕСТУВАННЯ РОБОТИ МОБІЛЬНОЇ СИСТЕМИ	46
4.1 Тестування системи	46
4.2 Тестування розробленого програмного застосунку	49
4.3 Розробка інструкції користувача.....	Ошибка! Закладка не определена.
4.4 Підтримувані пристрої, методи встановлення та розповсюдження додатку.....	61
4.5 Висновки.....	65
ВИСНОВКИ	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	67

ДОДАТКИ	Ошибка! Закладка не определена.
Додаток А. Технічне завдання.....	71
Додаток Б. Протокол перевірки бакалаврської кваліфікаційної роботи на наявність текстових запозичень	74
Додаток В. Лістинг програми	85
Додаток Г. Ілюстративна частина	100

ВСТУП

Обґрунтування вибору теми дослідження. У сьогоднішньому світі, де ритм життя постійно пришвидшується, а рівень стресу зростає, питання збереження психічного та фізичного здоров'я набуває особливої актуальності. Стрес, перевтома, тривожність — це часті супутники сучасної людини, які негативно впливають не лише на емоційний стан, а й на загальне самопочуття. Саме тому зростає інтерес до засобів саморегуляції та методів релаксації, серед яких особливу роль відіграють дихальні практики, медитація та інші техніки усвідомленості.

Одночасно зі зростанням уваги до психоемоційного стану, зростає й роль мобільних технологій. Смартфони, як універсальні засоби щоденної комунікації й організації часу, дедалі частіше використовуються як інструменти підтримки здоров'я. Завдяки розвитку iOS-платформи відкривається можливість створити повноцінну мобільну систему для підтримки релаксації, контролю дихання та медитації [1]. Такі застосунки сприяють самостійній турботі про здоров'я, є доступними, зручними та персоналізованими.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення [2].

Мета та завдання дослідження. Метою є покращення можливостей реабілітації і релаксу за рахунок розробки модуля контролю дихання, модуля медитацій та використання спеціалізованою мобільною системою для контролю дихання і медитації, що дозволяє відновлювати психологічний стан користувача.

Відповідно до поставленої мети потрібно вирішити такі завдання:

- провести аналіз вхідних даних системи;
- розробити методи, моделі та алгоритми;
- провести тестування роботи системи;

Об'єкт і предмет дослідження. Об'єктом дослідження є процес розробки мобільного застосунку з функціями дихальних вправ, медитації та релаксації. Предмет дослідження — методи та засоби розробки мобільної системи для

контролю дихання, медитації та релаксу для забезпечення персоналізованих нагадувань і підтримки психоемоційного стану користувачів.

Методи дослідження. У процесі дослідження використовувалися наступні методи:

- методи проектування архітектури програмних систем для реалізації мобільної системи контролю дихання;
- методи Swift та SwiftUI для створення нативного інтерфейсу;
- принципи мінімалістичного дизайну та user-friendly UI/UX для розробки інтерфейсу;
- тестування працездатності та зручності мобільного застосунку;
- створення логіки нагадувань і таймерів у режимах медитації й дихання.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод реалізації в мобільному застосунку модуля контролю дихання, який, на відміну від існуючих, реалізує персоналізовану взаємодію з користувачем на основі візуальних та тактильних підказок, що дозволяє користувачу підвищити зручність виконання дихальних вправ, отримувати зворотній зв'язок з системою, що сприятиме підвищенню ефективності релаксації, зниженню рівня стресу та покращенню загального психоемоційного стану.

2. Подальшого розвитку отримав метод реалізації в мобільному застосунку модуля медитацій, який, на відміну від існуючих, забезпечує інтеграцію адаптивних сценаріїв медитацій з урахуванням рівня досвіду користувача, поточного стану стомленості чи напруги та цілей релаксації, що дозволяє покращити індивідуалізований підхід до психоемоційного відновлення користувача та підвищити ефективність медитативної практики.

Практичне значення полягає у можливості використання розробленої мобільної системи для підтримки дихальних практик, медитації та релаксації, щоб покращити емоційний стан користувача та знизити рівень його стресу. Розроблене програмне забезпечення може бути корисним для осіб з високим рівнем стресу, тих, хто практикує медитацію, або користувачів, які бажають

зберігати здоровий баланс між роботою та відпочинком. Окрім цього, таке програмне забезпечення може бути використане для інтеграції в інші додатки, наприклад, у сфері здоров'я та продуктивності.

Особистий внесок здобувача. Усі етапи бакалаврської кваліфікаційної роботи від аналізу предметної області до реалізації функціонального прототипу мобільного застосунку виконано автором особисто. У науковій праці [3], опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, програмна реалізація функціоналу.

Апробація та публікація результатів. Результати бакалаврської кваліфікаційної роботи обговорювалися на XXV Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів (Одеса, 17-18 квітня 2025 р.) та опубліковані в тезах доповіді цієї конференції [3].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто чотири розділи та було використано 20 літературних джерел.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗВИТКУ ЗАСОБІВ ЗАБЕЗПЕЧЕННЯ МЕНТАЛЬНОГО ЗДОРОВ'Я ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз сучасного стану мобільних технологій для контролю дихання, медитації та релаксу

Сучасний етап розвитку інформаційних технологій невинно трансформує сферу охорони здоров'я, відкриваючи нові можливості для профілактики стресових станів, поліпшення психофізичного стану та покращення спокою. В сучасному часі високих технологій та високих темпів життя мобільні додатки для контролю дихання, медитації та релаксу стають все більш потрібні [3].

Основними причинами, що сприяють розвитку даної області, можна відокремити зростаючу потребу в інструментах заспокоєння. Люди активно шукають рішення, які дозволяють знизити рівень стресу, покращити сон та відновити психоемоційний стан.

Широке впровадження мобільних технологій, смартфонів стають платформою для алгоритмів, завдяки яким можна покращити свій стан, завдяки зв'язку, аудіо та відеоінструкцій, що дозволяють контролювати фізіологічні процеси.

Розвиток штучного інтелекту дає змогу сучасним вирішенням обробки сигналів і алгоритми дозволяють адаптувати сесії медитації та дихальних вправ до індивідуальних особливостей користувача [4].

Використання мобільних додатків як засобу для релаксації та медитації не лише сприяє профілактиці негативних наслідків стресу, а й стає важливим інструментом для підтримки здоров'я. Розробка програмних модулів для мобільної системи, що забезпечують моніторинг та корекцію дихального ритму, надає значні перспективи для покращення самопочуття користувачів та підвищення ефективності сучасних засобів заспокоєння [5].

1.2 Порівняльний аналіз аналогів

На сьогоднішній день на ринку існує ряд мобільних додатків, які орієнтовані на підтримку психоемоційного стану користувачів. До таких систем належать як міжнародні проекти, так і локальні розробки. Ось приклад деяких застосунків.

Calm [6]. Додаток пропонує широкий спектр аудіоінструкцій для медитації, релаксації, а також спеціальні вправи для контролю дихання. Система включає інтегровані біозворотні механізми, що дозволяють відслідковувати ефективність сесій. На рисунку 1.1 наведене зображення застосунку Calm.

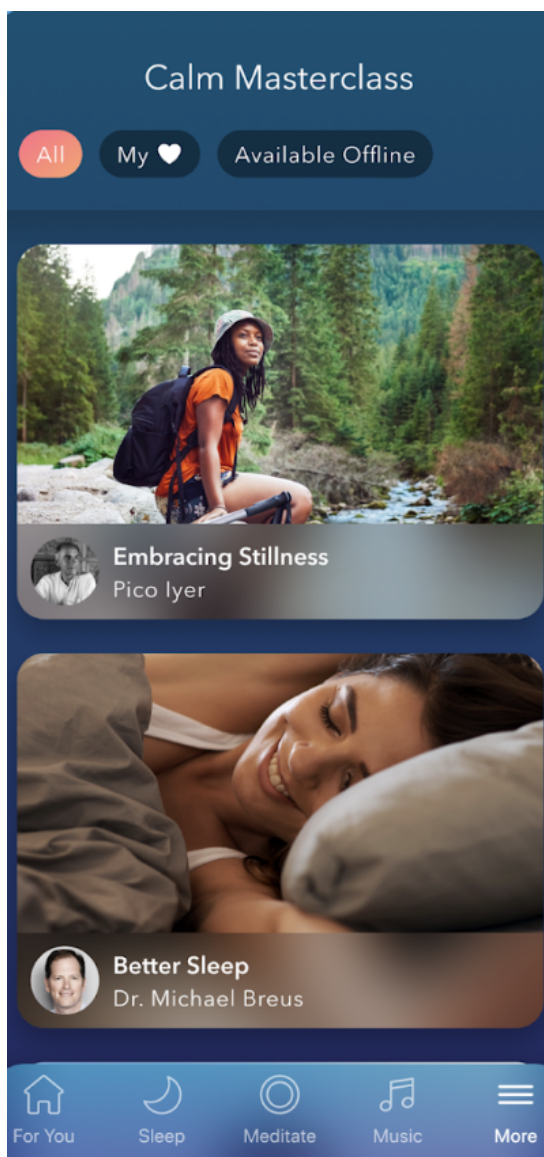


Рисунок. 1.1 – Застосунок Calm

Headspace [7]. Одна з провідних платформ, яка орієнтована на медитативні практики та техніки релаксації. Додаток містить як базові сесії, так і розширені програми для поглибленого занурення у практики усвідомленості. На рисунку 1.2 наведене зображення застосунку Headspace.

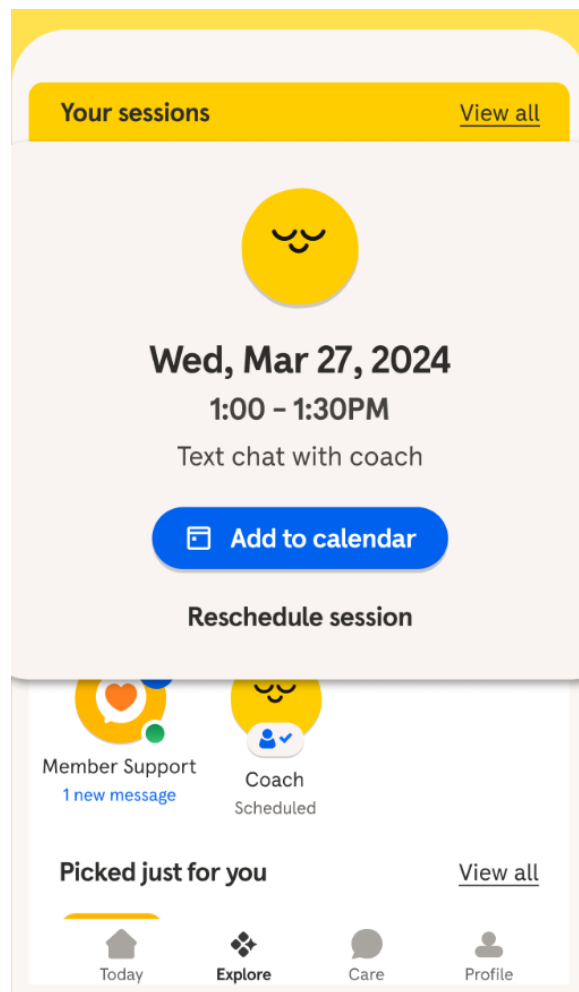


Рисунок. 1.2 – Застосунок Headspace

Breathe2Relax — це мобільний додаток, спеціально розроблений для проведення сесій релаксації з бажанням зниження рівня стресу та емоційного напруження [8]. Його основною перевагою є простий, інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам без зайвих зусиль розпочати вправи на дихання. Програма надає користувачу можливість налаштовувати параметри дихального циклу відповідно до індивідуальних потреб, а також містить візуальні та аудіо підказки для полегшення. Завдяки фокусу на доступності та

ефективності, додаток користується популярністю серед користувачів, які шукають шлях зняти стрес у повсякденному житті. На рисунку 1.3 наведене зображення застосунку Breathe2Relax.



Рисунок. 1.3 – Застосунок Breathe2Relax

З метою здійснення порівняльного аналізу функціональних можливостей існуючих аналогів і запропонованої власної розробки, було складено табл. 1.1, яка наочно демонструє ключові відмінності, переваги та особливості кожного з виборів. Ця таблиця слугує основою для глибшого розуміння доцільності

впровадження власного програмного продукту та його конкурентних переваг у порівнянні з уже наявними системами.

Таблиця 1.1 — Порівняльний аналіз аналогів

Критерії оцінки	Calm	Headspace	Breethe2Relax	Власна розробка
Дихальні вправи	1	1	1	1
Інтерактивні візуалізації	0	0	1	1
Таймери медитації	1	1	0	1
Нагадування про дихання	0	1	1	1
Режим фокусування	0	0	0	1
Сумарний коефіцієнт	2	3	3	5

За результатами порівняльного аналізу видно, що існуючі рішення мають як значні переваги, так і певні недоліки, зокрема у сфері інтеграції біоінформації та адаптації сесій до індивідуальних потреб. Розробка власного програмного модуля Irelax може врахувати недоліки аналогів та забезпечити більш гнучкий, адаптивний та інтегрований підхід до контролю дихання, медитації та релаксу.

1.3 Аналіз методів розв’язання задачі

При розробці мобільної системи для контролю дихання, медитації та релаксації необхідно враховувати важливі аспекти, які включають як апаратні, так і програмні елементи, щоб забезпечити ефективність та зручність використання. Інноваційні функції, що мають на меті покращити досвід

користувача, повинні бути реалізовані через поєднання передових алгоритмів обробки сигналів і високоякісного інтерфейсу користувача [9].

Одним з основних напрямків досліджень є алгоритми обробки сигналів, які використовуються для забезпечення високої точності вимірювань. У цьому контексті важливо розробити алгоритми попередньої обробки біосигналів, що дозволяють мінімізувати вплив шумів та артефактів, які можуть виникати в процесі збору даних [10]. Біосигнали, такі як частота серцебиття, температура тіла, рівень насичення киснем у крові або електрична активність серця, можуть бути схильні до спотворень через різні фактори, рухи користувача або зовнішні шуми. Для цього необхідно розробити алгоритми, які здатні фільтрувати та очищати ці сигнали, а також виділяти корисну інформацію, необхідну для подальшої адаптації медитаційних сесій або дихальних вправ. Такі алгоритми дозволяють досягти більш точної оцінки стану користувача та оптимізувати адаптацію програмних функцій до конкретних фізіологічних характеристик.

Другим важливим напрямком є адаптивність інтерфейсу користувача. Інтерфейс мобільного додатку має бути зручним та інтуїтивно зрозумілим, щоб користувачі могли легко взаємодіяти з системою, не відчуваючи труднощів у налаштуванні або використанні функцій [11]. Особливо важливим є створення інтерактивних аудіо-відео інструкцій, що допоможуть користувачеві виконувати вправи правильно і вчасно. Відображення прогресу користувача у реальному часі, зокрема через графічні інтерфейси, дозволяє не тільки моніторити ефективність виконаних вправ, але й створює елемент мотивації. Зворотний зв'язок є у підтримці позитивного досвіду користувача, оскільки він дозволяє коригувати виконання вправ, вказуючи на можливі помилки або надаючи додаткові рекомендації [12].

З урахуванням цих аспектів, оптимальним підходом до розробки мобільної системи є комбінування методів обробки біосигналів з використанням алгоритмів штучного інтелекту для персоналізації та адаптації системи до індивідуальних потреб користувачів. Крім того, інтеграція з мобільним

інтерфейсом дозволяє забезпечити комфорт користувача завдяки простоті у взаємодії з додатком і можливості налаштувати параметри [13].

Впровадження таких інноваційних функцій не лише підвищить точність вимірювань, але й покращить загальний користувацький досвід, що є критичним для успішного прийняття та використання системи широким колом користувачів. Комбінація передових технологій обробки сигналів, штучного інтелекту та адаптивного інтерфейсу є ключем до створення ефективного мобільного додатку, який забезпечить користувачам можливість підтримувати своє психоемоційне і фізичне здоров'я на високому рівні.

1.4 Постановка задач дослідження

На основі проведеного аналізу потреб користувачів та технологій для мобільних додатків, необхідно вирішити кілька ключових завдань для розробки системи контролю дихання, медитації та релаксації.

Розробка модуля збору даних. На першому етапі необхідно забезпечити просте і ефективне введення даних користувача, без використання складних сенсорів. Користувач може вручну вводити свої фізіологічні параметри, такі як частота дихання чи пульс, або обирати загальні рекомендації, що підходять для його стану. Важливо створити прості алгоритми, які б допомогли аналізувати ці дані і пропонували підходящі вправи або медитації.

Створення адаптивного модуля медитації та релаксу. Друге завдання розробити систему медитацій і релаксацій, яка була б адаптована до індивідуальних потреб користувача, навіть якщо ті не використовують сенсори для збору даних. Система повинна включати аудіо- та відеоінструкції, що допомагають користувачам легко слідувати медитативним вправам та дихальним технікам. Користувач може вибирати тривалість сесій, тип медитації, а також отримувати зворотний зв'язок про прогрес у реальному часі.

Інтеграція зворотного зв'язку. Важливо розробити алгоритми, які будуть коригувати прогрес користувача на основі його власних відгуків чи почуттів, тобто за допомогою простих запитань після кожної сесії, які допомагають

оцінити рівень стресу або релаксації користувача. Це дозволить автоматично коригувати рекомендації та створювати індивідуалізовані медитації.

Розробка графічного інтерфейсу користувача. Оскільки зручність користування є однією з головних вимог, інтерфейс додатка має бути простим і інтуїтивно зрозумілим. Навігація між режимами медитації, дихання та релаксації має бути легкою та швидкою. Користувачі повинні мати можливість переглядати історію своїх сесій, отримувати загальні рекомендації на основі їхнього досвіду, а також бачити прогрес у вигляді графіків чи статистики.

Цей підхід дозволить створити ефективну мобільну систему, яка відповідає потребам користувачів і забезпечить комфорт у використанні без необхідності складних сенсорних технологій.

Таким чином необхідно провести аналіз вхідних даних системи, розробити метод поетапного керування діями користувача та моделі системи, а також розробити алгоритм системи для моніторингу дихання.

1.5 Висновки

У першому розділі проведено комплексний аналіз сучасного стану мобільних технологій для контролю дихання, медитації та релаксу. Розглянуто особливості використання сенсорних даних для моніторингу фізіологічних параметрів, а також проаналізовано функціональні можливості провідних додатків, таких як Calm, Headspace, Breathe2Relax. Порівняльний аналіз показав, що існуючі рішення мають як переваги, так і недоліки, зокрема у сфері адаптації до індивідуальних потреб користувача та інтеграції біоінформації.

На підставі проведеного аналізу сформульовано завдання, які охоплюють розробку модулів збору та обробки даних, створення адаптивного інтерфейсу для медитації та релаксу, а також забезпечення високого рівня безпеки збереження персональних даних.

Отже, розробка програмних модулів мобільної системи для контролю дихання, медитації та релаксу є надзвичайно актуальним і перспективним напрямком, що поєднує в собі новітні досягнення в галузі мобільних технологій,

збору та обробки біометричних даних, штучного інтелекту та розробки інтуїтивно зрозумілого графічного інтерфейсу.

Реалізація такого застосунку дозволяє користувачеві не лише ефективно виконувати вправи з дихання та медитації, але й отримувати візуальні, звукові або тактильні підказки, що сприяють глибшому зануренню в процес релаксації. Крім того, можливість персоналізації взаємодії з користувачем — шляхом налаштування інтенсивності вправ, тривалості сесій, вибору режимів та ведення статистики — робить систему більш гнучкою, адаптивною та корисною для широкого кола користувачів.

Розробка подібних програмних засобів є важливою частиною цифрової трансформації у сфері ментального здоров'я, оскільки надає доступний та зручний інструмент підтримки самопочуття без потреби у складному обладнанні або спеціальній підготовці. Це відкриває широкі можливості як для індивідуального користування, так і для впровадження у сферу охорони здоров'я, реабілітації та

2 РОЗРОБКА МЕТОДІВ, МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для реалізації програмного продукту, який забезпечує контроль за диханням, медитацією та релаксацією через мобільну систему iOS, будуть використані сучасні методи програмування та штучного інтелекту. Основні технології для розробки включають мову програмування Swift, що є основною для iOS додатків, а також інтеграцію з системами тривимірного моделювання для візуалізації результатів дихальних та медитативних вправ.

Swift – це високопродуктивна мова програмування, яка є основою для розробки мобільних додатків під платформу iOS [14]. Вона підтримує об'єктно-орієнтоване програмування та інтеграцію з технологіями Apple.

Розробка програмного забезпечення виконується у кілька етапів. Спершу буде створено модуль, який забезпечить користувачеві інтуїтивно зрозумілий інтерфейс для взаємодії з системою. Цей модуль відповідатиме за основні функції застосунку, включаючи управління вправами, налаштування нагадувань для дихання та медитації. Далі буде розроблений модуль для відстеження рухів користувача та інтерактивного управління вправами за допомогою візуалізації 3D-моделей.

Після цього буде створено модуль для реального часу візуалізації дихальних та медитативних вправ, що допоможе користувачам коригувати техніку виконання через тривимірну модель. Візуалізація дозволить наочно спостерігати за процесом дихання або медитації, а також дасть можливість інтерактивно взаємодіяти з 3D-моделлю для оптимізації результатів.

2.2 Розробка методу реалізації в мобільному застосунку модуля контролю дихання

У мобільному застосунку для контролю дихання реалізовано спеціальний метод, що забезпечує кероване, структуроване виконання дихальних вправ

відповідно до налаштувань користувача. Основна мета модуля — надати користувачеві ефективний інструмент для покращення психоемоційного стану шляхом правильного дихання, з урахуванням ритму, тривалості вдиху, паузи та видиху.

Метод контролю дихання реалізовано за допомогою анімацій, таймерів, логіки станів та індикаторів фаз, що дозволяють користувачу слідкувати за дихальним ритмом без необхідності самотійного відліку часу. Кожна фаза вправи візуалізується на екрані, що сприяє кращій концентрації та ефективності виконання.

Модель роботи системи для контролю дихання, медитації та релаксації була розроблена через UML діаграму діяльності, яка чітко описує алгоритм взаємодії користувача з програмою. Діаграма діяльності є важливим інструментом для візуалізації та розуміння послідовності дій, що виконуються в системі, а також визначає всі основні етапи, через які проходить користувач, використовуючи мобільний додаток [15].

1. Запуск застосунку відбувається через функцію `main()`, яка виконує ініціалізацію Flutter-середовища та запускає основний інтерфейс застосунку.
2. Ініціалізація локалізації та формату дати виконується за допомогою `initializeDateFormatting()`, що дозволяє адаптувати застосунок під українську локаль (`uk_UA`).
3. Створення головного віджета застосунку `MyApp`, який відповідає за початкову побудову інтерфейсу, тему, маршрути та початковий екран.
4. Визначення першого етапу взаємодії користувача — екрану вітання або ознайомлення з дихальним модулем, де пояснюється користь вправ та правила використання.
5. Поетапне керування здійснюється через `PageView`, `Stepper` або власну реалізацію переходів, де кожен етап — окрема дія: вибір режиму, налаштування фаз дихання, запуск вправи тощо.

6. Менеджер стану (наприклад, Provider) використовується для збереження стану між кроками, зокрема: тривалості фаз, поточного етапу, індикатору виконання вправи.
7. Підказки, анімації та індикатори прогресу допомагають користувачеві орієнтуватися в процесі: коло, яке розширюється/звужується, індикатор фаз або таймер на екрані.
8. Фінальний етап — відображення результату сеансу: повідомлення про завершення вправи, рекомендації, кнопка «Повторити» або «Завершити» .

Розроблений метод реалізації модуля контролю дихання є ефективним підходом до створення функціонального та зручного мобільного застосунку. Завдяки використанню Flutter та декларативного підходу до побудови інтерфейсів, модуль дихальних вправ реалізовано з високою гнучкістю, адаптивністю та простотою в користуванні. Така архітектура дає змогу створити зрозумілий інтерфейс для користувача, де кожна дія логічно пов'язана з наступною від налаштування параметрів до візуального супроводу дихальних фаз.

Модуль контролю дихання побудований за принципами модульності та повторного використання коду, що значно спрощує підтримку й подальший розвиток застосунку. Завдяки чітко структурованій логіці взаємодії та можливості гнучко керувати фазами дихання система адаптується під різні рівні користувачів від новачків до досвідчених практиків.

Модель роботи програми реалізовано у вигляді діаграми класів. Завдяки використанню Flutter та декларативного підходу до побудови інтерфейсів, реалізація поетапної взаємодії стає простою в реалізації, гнучкою у налаштуванні та зручною для користувача. Така архітектура підвищує якість користувацького досвіду, сприяє логічній структурі процесів у застосунку та дозволяє легко масштабувати функціонал у майбутньому. Для кращого розуміння архітектури розроблюваного застосунку було побудовано модель роботи програми діаграми класів (рис. 2.1).



Рисунок 2.1 – Модель роботи програми

Діаграма діяльності забезпечує чітке розуміння послідовності дій, через які користувач проходить при взаємодії з додатком. Вона також визначає функціональні можливості системи, такі як налаштування вправ, візуалізація через 3D-анімацію та інтерактивні підказки, що робить процес виконання медитації та дихальних вправ більш ефективним і зручним для користувача. Система створена так, щоб кожен етап був інтуїтивно зрозумілим, а користувач мав доступ до необхідної інформації та підтримки в реальному часі.

2.3 Розробка методу реалізації в мобільному застосунку модуля медитацій

Процес реалізації модуля контролю дихання в мобільному застосунку побудований на основі фреймворку Flutter і передбачає чітку послідовність етапів, які формують повний цикл дихальної практики від вибору параметрів до завершення сеансу. Реалізація модуля базується на застосуванні сучасних

підходів до побудови інтерфейсу, управління станом та забезпечення інтерактивності, що дозволяє досягнути високої якості користувацького досвіду.

На першому етапі користувач обирає налаштування майбутньої практики. Для збереження та обробки цих параметрів використовується модель стану, що дозволяє передавати дані між екранами без втрати цілісності.

Після підтвердження налаштувань запускається основний екран практики. На цьому етапі реалізується візуальна анімація, яка синхронізується з фазами дихання коло, що поступово розширюється під час вдиху, зупиняється під час затримки та стискається при видиху. Така візуалізація допомагає користувачеві слідувати за ритмом дихання, не відволікаючись на лічильники чи текстові інструкції.

Механізми управління станом забезпечують перехід між фазами, контроль часу, відстеження прогресу виконання та обробку завершення кожного циклу. Наприкінці сеансу відображається підсумок, який може містити коротке повідомлення про завершення.

Загалом, реалізація модуля контролю дихання передбачає просту, логічну та адаптивну архітектуру, яка дозволяє легко змінювати або доповнювати функціонал без порушення загальної структури застосунку. Це створює гнучку платформу для майбутніх оновлень і розширень, зберігаючи при цьому зручність та ефективність дихальної практики для користувача. Кожен етап реалізується через відповідні програмні компоненти, структури даних і механізми управління станом інтерфейсу.

1. Запуск застосунку. Ініціалізація мобільного додатка починається з виклику функції `main()`, яка забезпечує запуск Flutter-середовища та завантаження основного віджета `MyApp()`. Для підтримки української локалізації попередньо виконується `initializeDateFormatting('uk_UA')`.

2. Відображення головного екрана. У класі `MyApp` визначено основну конфігурацію інтерфейсу. Через метод `build()` створюється об'єкт `MaterialApp`, у якому встановлено початковий маршрут до головного екрана (`HomeScreen`). Звідти користувач має змогу перейти до функціоналу дихання.

3. Початок роботи з модулем дихання. Після натискання кнопки «Контроль дихання» здійснюється навігація до `BreathingScreen()`, де реалізовано весь функціонал контролю та візуалізації дихального процесу.

4. Налаштування параметрів дихання. Користувач може задати індивідуальні значення фаз дихання — тривалість вдиху, затримки, видиху, кількість циклів тощо. Ці дані зберігаються у відповідних змінних або передаються через модель стану `BreathingModel`, яка управляється за допомогою `Provider`.

5. Візуалізація дихального циклу. Основна логіка контролю дихання реалізована через анімаційні інструменти Flutter: `AnimationController`, `Tween`, `AnimatedBuilder`. Для кращого візуального сприйняття використовується анімоване коло, яке змінює розмір або прозорість залежно від фази — вдих, затримка, видих.

6. Тактильний супровід. Для посилення ефекту концентрації додається вібрація пристрою під час зміни фаз дихання. Це реалізується через стандартні API доступу до апаратного забезпечення пристрою.

7. Завершення сеансу. Після завершення заданої кількості циклів користувач бачить повідомлення про успішне завершення вправи. Йому пропонується зберегти статистику, повторити сеанс або повернутись до головного меню.

Завдяки використанню можливостей Flutter — таких як `StatefulWidget`, `Provider`, `Navigator`, а також гнучкого декларативного підходу до створення UI — модуль контролю дихання реалізується ефективно, забезпечуючи при цьому зручний, інтуїтивно зрозумілий та візуально привабливий інтерфейс. Такий підхід сприяє підвищенню користувацького комфорту, а також покращенню психоемоційного стану через систематичну практику дихальних вправ.

Опис алгоритм контролю дихання.

1. Користувач відкриває вкладку "Дихання".
2. Система запитує бажаний режим або використовує попередньо збережений.

3. На основі обраного режиму визначаються параметри дихального циклу.
Тривалість вдиху, затримки, видиху.
4. Відображається анімація коло, що розширюється/звужується.
5. Синхронно з анімацією відтворюються звукові підказки якщо обрано.
6. По завершенню сесії показується короткий звіт: тривалість, кількість циклів, рекомендації.
7. Система пропонує зберегти сесію або поділитися результатами.

Опис алгоритму запуску медитації.

Надання користувачу можливість запустити сеанс медитації із заданими параметрами тривалість, фоновий звук, візуалізація.

Опис алгоритму:

1. Користувач обирає вкладку "Медитація".
2. Вибирає тип медитації на концентрацію, релаксацію.
3. Встановлює параметри: Тривалість, фоновий звук дощ, природа, тиша, нагадування або гайдований голос.
4. Починається сеанс. Екран затемнюється або виводиться візуальна медитація, відтворюється звук або інструкції.
5. Користувач може призупинити або завершити сеанс.
6. Після завершення показується повідомлення з підсумком.

Опис алгоритму нагадувань і режиму релаксації

Нагадування користувачеві про необхідність зробити паузу, розслабитися або виконати дихальну практику.

Опис алгоритму:

1. Користувач задає інтервали нагадувань.
2. Система створює локальні нотифікації.
3. У визначений час система надсилає повідомлення: "Час зробити паузу. Глибоко вдихни та розслабся."
4. Якщо користувач відкриває сповіщення. Відразу пропонується коротка сесія релаксації. Запускається відповідний сценарій.
5. Усі нагадування реєструються в журналі активності.

Блок-схема, яка ілюструє загальний процес функціонування мобільної системи для контролю дихання, медитації та релаксації, продемонстровано на рисунку 2.1. Вона відображає основні етапи роботи застосунку — починаючи з ініціалізації користувацького сеансу, вибору режиму дихання, медитація або релаксація, налаштування параметрів сесії, і завершуючи зворотним зв'язком або рекомендаціями після завершення вправи. Така візуалізація дозволяє чітко простежити логіку взаємодії користувача з додатком та зрозуміти внутрішню структуру системи, що сприяє кращому розумінню принципів її роботи. Крім того, система може реагувати на тривалість або темп виконання вправи — наприклад, сповільнювати або пришвидшувати вказівки, якщо користувач відстає або випереджає рекомендований темп.

Завдяки цьому досягається високий рівень персоналізації, що дозволяє користувачеві почуватися більш комфортно та впевнено, навіть якщо це його перший досвід у практиці дихання чи медитації.

Такий підхід сприяє глибшому залученню до процесу та формуванню звички до регулярного використання системи для підтримки психоемоційного балансу.

Блок-схема, яка ілюструє загальний процес функціонування мобільної системи для контролю дихання, медитації та релаксації, продемонстровано на рисунку 2.2.

Ця блок-схема відображає основні етапи взаємодії користувача з мобільною системою, починаючи з моменту запуску застосунку і завершуючи виконанням обраної практики дихання, медитації або релаксації. Процес функціонування можна умовно поділити на такі послідовні блоки:

Блок-схема, яка ілюструє загальний процес функціонування мобільної системи для контролю дихання, медитації та релаксації, продемонстровано на рисунку 2.2.

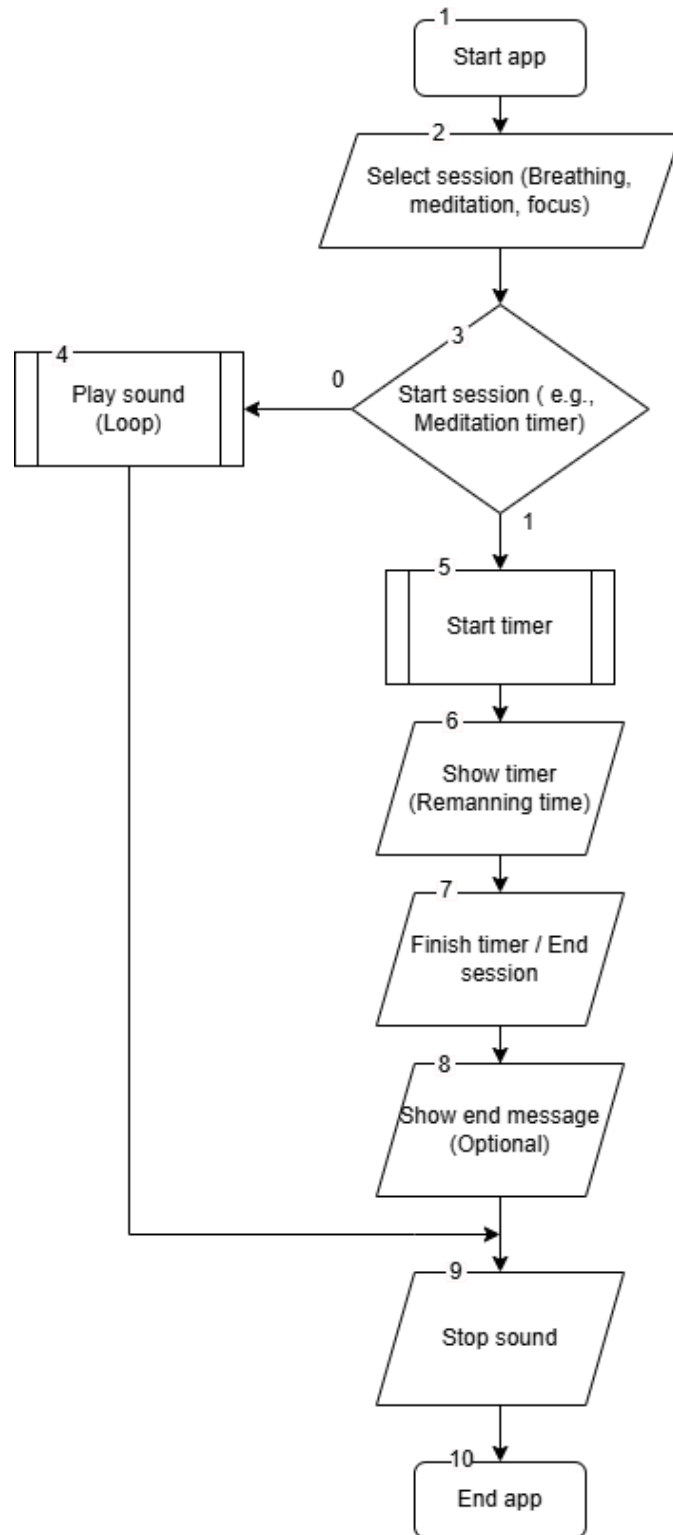


Рисунок 2.2 — Блок-схема загального процесу роботи мобільної системи для контролю дихання, медитації та релаксу.

Блок-схема алгоритму роботи мобільної системи для контролю дихання, медитації та релаксу наведена на рисунку 2.3.

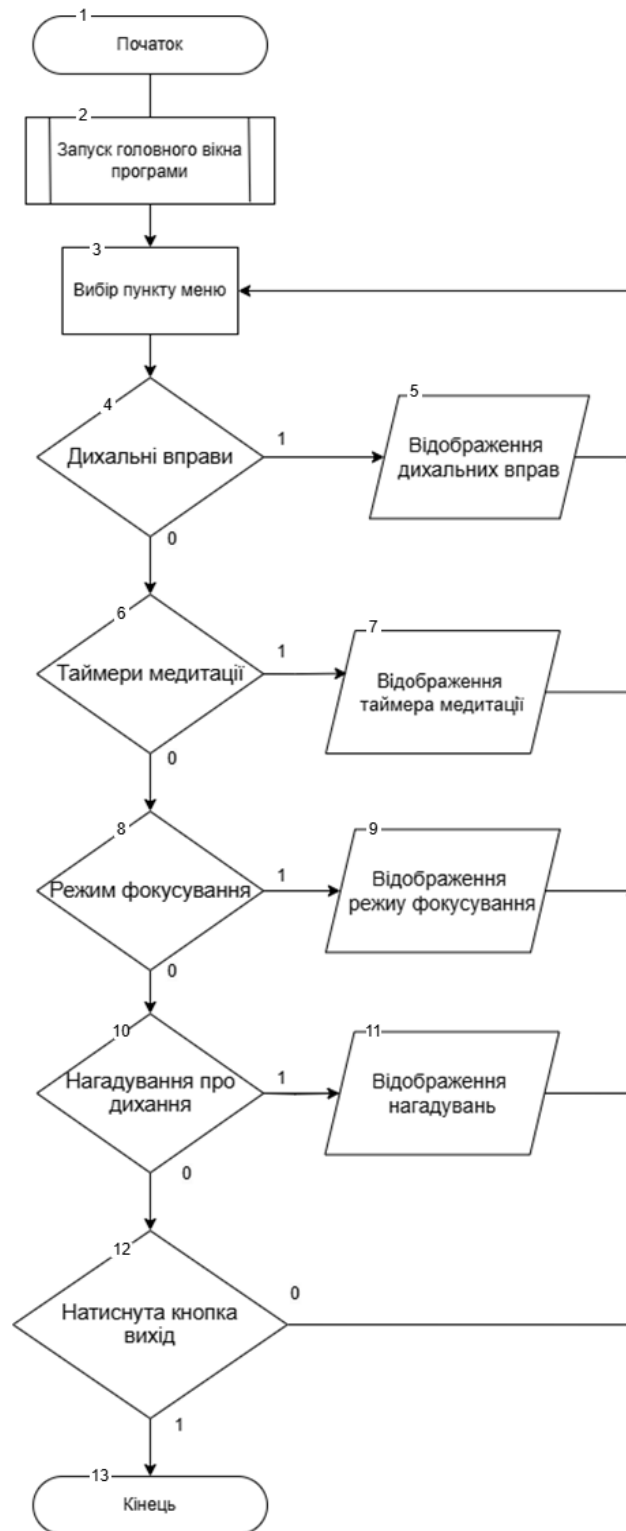


Рисунок 2.3. — Блок-схема алгоритму роботи мобільної системи для контролю дихання, медитації та релаксу.

Алгоритм системи для моніторингу дихання, медитації та релаксації побудований, щоб ефективно забезпечити користувачу інтерактивний досвід,

включаючи налаштування параметрів вправ, перевірку коректності даних і візуалізацію результатів.

Перевірка коректності даних моделі: на першому етапі алгоритм здійснює перевірку вхідних даних, що надходять від користувача або сенсорів. Це можуть бути дані про частоту дихання, ритм, тривалість медитації або емоційний стан, отримані через відповідні сенсори або вручну введені користувачем. Модель перевіряє коректність введених даних, зокрема чи знаходяться вони в межах нормальних значень для здорової людини. Якщо дані не відповідають очікуваним стандартам, система попереджає користувача або пропонує коригувати введені значення. Цей етап гарантує, що всі дії, здійснені під час виконання вправ, є науково обґрунтованими і відповідають нормам.

Налаштування параметрів дихальних і медитативних вправ: другий етап алгоритму полягає у налаштуванні індивідуальних параметрів для кожної вправи. Це можуть бути такі параметри, як частота вдихів та видихів, тривалість дихальних циклів, кількість повторів або час для кожної фази медитації, час для фокусування на диханні, час для відпочинку чи глибокої медитації). Користувач може вибирати або налаштовувати ці параметри залежно від своїх уподобань і рівня підготовки. Алгоритм адаптує ці налаштування для забезпечення оптимальної ефективності, враховуючи дані, отримані під час попередньої перевірки.

Інтерактивна візуалізація результатів: на заключному етапі алгоритм відповідає за інтерактивну візуалізацію результатів виконаних вправ. Після кожного циклу вправ користувач отримує графічне або числове відображення своїх результатів. Це може бути графік змін частоти дихання, рівня стресу або емоційного стану, а також інші показники, що демонструють ефективність вправ. Візуалізація не лише дозволяє користувачу оцінити свій прогрес, але й надає йому рекомендації щодо корекції дій для покращення результатів у майбутньому. Система також може надавати підказки щодо оптимізації вправ в залежності від поточного фізіологічного стану користувача.

Алгоритм має на меті не лише контроль за виконанням вправ, але й створення інтуїтивно зрозумілого досвіду для користувача, який дозволяє йому покращувати своє здоров'я і відчуття через регулярні практики дихання та медитації. Цей підхід забезпечує максимальну ефективність і результативність системи.

2.4 Висновки

У другому розділі було проведено детальний аналіз вхідних даних, що дозволило створити надійну основу для подальшої розробки програмного забезпечення системи контролю дихання, медитації та релаксації. Розроблені основні алгоритми та модулі забезпечують ефективну обробку даних, що надходять від користувача або сенсорів, що критично важливо для правильної роботи додатку та його взаємодії з користувачем.

Зокрема, було створено модель роботи системи за допомогою UML діаграм, що дозволяє чітко структуровано представити всі взаємодії між компонентами системи, а також показати потоки даних і логіку роботи додатку. Це дозволяє не лише зрозуміти загальну архітектуру системи, але й створити чітку документацію, яка стане в нагоді під час подальшого розвитку проекту.

Розроблені модулі системи є основою для ефективної реалізації функцій дихальних вправ, медитації та релаксації. Вони забезпечують високий рівень персоналізації і здатні значно покращити процес релаксації та відновлення користувачів, що сприяє досягненню бажаних результатів в оптимальний спосіб.

На основі проведеного аналізу та розроблених алгоритмів, система має великий потенціал для ефективного використання, покращення фізіологічного та емоційного стану користувачів, що є основною метою розробки, а саме покращення можливостей реабілітації і релаксу за рахунок використання спеціалізованою мобільної системи для контролю дихання і медитації, що дозволяє відновлювати психологічний стан користувача.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ МОБІЛЬНОЇ СИСТЕМИ

3.1 Варіантний аналіз та обґрунтування засобів програмного додатку

Для вибору мови програмування для розробки мобільного додатку системи контролю дихання, медитації та релаксації важливо враховувати кілька аспектів, таких як продуктивність, сумісність з платформою, доступність бібліотек і фреймворків, а також здатність адаптуватися до змінних умов та потреб користувача. Ось три мови програмування, які можуть бути використані для розробки такого додатку: Python, Swift і Java.

Python є високорівневою інтерпретованою мовою програмування, яка має величезну кількість потужних бібліотек для реалізації різноманітних завдань, включаючи задачі, пов'язані з комп'ютерним зором, штучним інтелектом, нейронними мережами, обробкою даних і машинним навчанням [16]. Завдяки своїй простоті та зручності Python є однією з найбільш популярних мов серед розробників прототипів і наукових дослідників. Однією з основних переваг Python є можливість швидко створювати алгоритми для задач, пов'язаних з медитацією та контролем дихання, без необхідності занурюватися в глибокі технічні деталі. Проте Python має обмеження у використанні на мобільних пристроях, оскільки вимагає додаткових інструментів для компіляції або створення додатків для мобільних платформ. Тому цей варіант краще використовувати для серверної частини додатку або у поєднанні з іншими мовами для мобільної розробки.

Swift є нативною мовою програмування для розробки додатків на платформі iOS, що дає значні переваги в плані інтеграції з екосистемою Apple [17]. Оскільки Swift є основною мовою для створення додатків для iPhone та iPad, вона забезпечує високу продуктивність і швидкість виконання завдань, що особливо важливо для мобільних додатків, де ефективність є критично важливою. Swift дозволяє безпосередньо використовувати потужні фреймворки, такі як CoreML для реалізації машинного навчання та Vision для комп'ютерного зору, що дає можливість додатку аналізувати стан користувача під час медитації

або контролю дихання, а також адаптувати сповіщення залежно від його емоційного чи фізіологічного стану. Однією з головних переваг Swift є його тісна інтеграція з апаратними засобами пристроїв Apple, такими як датчики руху, акселерометри, гіроскопи та інші сенсори, які можна використовувати для створення більш точних і ефективних механізмів контролю дихання та моніторингу фізичного стану користувача. Swift також є оптимізованою для роботи в середовищі iOS, що дозволяє досягти високої продуктивності та швидкості роботи додатку. Завдяки цим можливостям Swift є ідеальним вибором для створення мобільного додатку для платформ Apple, оскільки він забезпечує максимальну продуктивність і ефективність використання всіх функцій пристроїв.

Java — одна з найпопулярніших мов програмування, яка здобула широке визнання завдяки своїй універсальності та стабільності. В основному вона використовується для розробки кросплатформених додатків, зокрема для операційної системи Android [18]. Java дозволяє створювати стабільні та безпечні мобільні додатки завдяки своїй багатій екосистемі та величезній кількості доступних бібліотек. Однак, порівняно з Swift, Java може бути менш продуктивною при розробці додатків для iOS, оскільки ця мова не є рідною для екосистеми Apple, і для її використання на платформі iOS потрібні додаткові інструменти та середовища, такі як інтерпретатори або кросплатформені рішення.

Хоча Java має підтримку бібліотек для комп'ютерного зору та машинного навчання, вона може бути менш оптимізованою для інтеграції з функціями пристроїв Apple, такими як специфічні датчики або можливості для обробки даних у реальному часі. Тому Java може бути хорошим вибором для створення додатків на Android, але для iOS розробки Swift має набагато більше переваг.

У таблицю 3.1 зведено порівняльний аналіз мов програмування.

Таблиця 3.1 — Порівняльний аналіз мов програмування

Характеристика	Python	Swift	Java
Підтримка комп'ютерного зору	+	+	+
Підтримка штучного інтелекту	+	+	+
Підтримка машинного навчання	+	+	+
Розпізнавання обличчя	+	+	+
Розпізнавання емоцій	+	+	-
Кросплатформеність	+	-	+
Інтеграція з iOS/Android	-	+	+
Підтримка датчиків і сенсорів	+	+	-
Загальна оцінка	6	7	6

Отже, кожна з розглянутих мов програмування має свої переваги та недоліки в контексті розробки системи для контролю дихання, медитації та релаксації. Python є потужним інструментом для швидкої розробки прототипів, але менш оптимізований для мобільних додатків. Swift забезпечує чудову інтеграцію з екосистемою Apple і ідеально підходить для розробки додатків для iOS, використовуючи найсучасніші фреймворки для машинного навчання та комп'ютерного зору. Java ж, хоча й є потужною та універсальною мовою, є менш оптимізованою для роботи на iOS і потребує додаткових шляхів для реалізації кросплатформеності.

Мова Swift є найбільш підходящою для розробки мобільних додатків на платформі iOS, оскільки надає всі необхідні інструменти для роботи з мобільними сенсорами та інтеграції з функціями iPhone.

3.2 Розробка модуля для авторизації та реєстрації користувача

Модуль авторизації та реєстрації є одним із ключових компонентів мобільного застосунку для контролю дихання, медитації та релаксації. Його основне призначення забезпечення безпечного доступу користувача до персоналізованих функцій системи. У цьому модулі реалізовано механізми створення нового облікового запису, входу в систему та збереження сесії.

Після запуску застосунку користувач бачить вітальний екран, де пропонуються два варіанти вхід у систему або реєстрація (рис. 3.1). У разі вибору реєстрації користувач вводить необхідні дані, адресу електронної пошти та пароль, а також підтверджує згоду з політикою конфіденційності.

Якщо обліковий запис вже існує, система повідомляє про це і пропонує увійти. У випадку авторизації після введення коректних даних, користувач отримує доступ до головного екрана застосунку.

Також передбачено обробку поширених помилок: введення некоректного email, слабкого пароля або повторне використання вже зареєстрованої адреси. Для підвищення зручності реалізовано збереження активної сесії, що дозволяє користувачеві не проходити авторизацію щоразу при запуску застосунку.

Модуль авторизації та реєстрації забезпечує не лише захист даних, але й комфорт користувача при взаємодії з системою, що є важливою умовою успішного використання застосунку в щоденній практиці.

Окрім базових функцій, модуль підтримує інтеграцію з популярними зовнішніми сервісами, такими як авторизація через Apple ID або Google-акаунт, що значно спрощує процес входу для користувачів, які цінують швидкість та зручність. У перспективі також передбачено реалізацію багатфакторної автентифікації для підвищення рівня безпеки, зокрема для користувачів, які

зберігають у застосунку чутливі персональні дані, пов'язані з психологічним станом або історією практик.

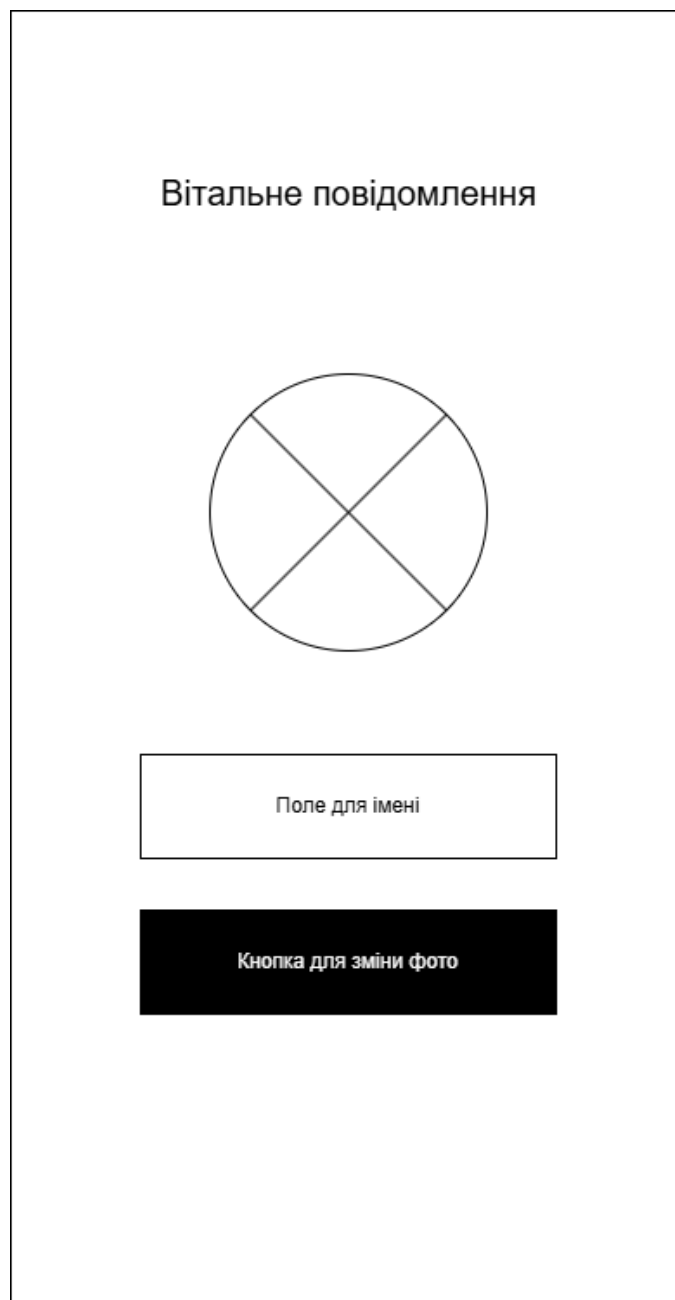
Загалом, модуль авторизації та реєстрації відіграє важливу роль у створенні позитивного першого враження від застосунку, забезпечуючи надійність, доступність і простоту використання. Завдяки продуманому інтерфейсу та функціональній гнучкості він створює основу для персоналізованої взаємодії користувача з системою, сприяючи формуванню довіри до застосунку та стимулюючи регулярне його використання.

The diagram illustrates the layout of a registration screen. At the top, the text "Вітальне повідомлення" (Welcome message) is centered. Below it is a large circle with a diagonal cross, representing a missing profile picture. Underneath the circle are two rectangular input fields: the first is labeled "Поле для імені" (Name field) and the second is labeled "Поле для паролю" (Password field). Below these fields is a solid black rectangular button labeled "Кнопка для входу в акаунт" (Login button). At the bottom of the screen, there is a line of text: "Пропозиції зареєструватися, якщо немає акаунту" (Offer to register if there is no account).

Рисунок 3.1 – Схема інтерфейсу сторіни реєстрації

Такий підхід дозволяє одразу спрямувати користувача до потрібної дії, залежно від того, чи використовував він застосунок раніше. Це зменшує час на пошук необхідного функціоналу й позитивно впливає на користувацький досвід.

Модуль автентифікації також має інтеграцію з базою даних, де зберігаються облікові записи користувачів, що дає змогу коректно перевіряти введені дані під час входу та уникати дублювання під час реєстрації (рис 3.2).



Вітальне повідомлення

Поле для імені

Кнопка для зміни фото

The diagram illustrates the layout of a login page. It is enclosed in a large rectangular frame. At the top, the text 'Вітальне повідомлення' (Welcome message) is centered. Below it is a circular placeholder for a profile picture, represented by a circle with a diagonal cross. Further down is a rectangular input field labeled 'Поле для імені' (Name field). At the bottom is a solid black rectangular button labeled 'Кнопка для зміни фото' (Change photo button).

Рисунок 3.2 – Схема інтерфейсу сторінки авторизації

Окрім цього, він може бути розширений додатковими функціями, такими як автентифікація через Apple ID, біометричний вхід Face ID або Touch ID або відновлення пароля, що ще більше підвищить зручність та безпеку користування застосунком.

Модуль автентифікації є невід’ємною складовою загальної архітектури системи, яка дозволяє налаштувати персоналізовану взаємодію користувача з функціоналом програми та забезпечує захист конфіденційних даних.

Реєстрація нового користувача проходить, після першого запуску застосунку користувач потрапляє на екран входу або реєстрації. Реєстрація реалізована за допомогою email та пароля. (рис 3.3).

```
Auth.auth().createUser(withEmail: email, password: password) { result, error in
    if let error = error {
        self.errorMessage = error.localizedDescription
    } else {
        self.isLoggedIn = true
    }
}
```

Рисунок 3.3 – Реєстрація нового користувача

Після успішної реєстрації користувача автоматично переводять до основного інтерфейсу застосунку.

У випадку, якщо обліковий запис вже існує, користувач може здійснити вхід. Для цього використовується інший метод (рис 3.4).

```
Auth.auth().signIn(withEmail: email, password: password) { result, error in
    if let error = error {
        self.errorMessage = error.localizedDescription
    } else {
        self.isLoggedIn = true
    }
}
```

Рисунок 3.4 – Авторизація

Стан входу зберігається в локальному сховищі за допомогою `@AppStorage`, що дозволяє застосунку автоматично відображати потрібний інтерфейс при повторному запуску.

У разі втрати паролю користувач має можливість його відновити за допомогою електронної пошти. Для цього реалізовано наступний механізм (рис 3.5).

```
Auth.auth().sendPasswordReset(withEmail: email) { error in
    if let error = error {
        self.errorMessage = error.localizedDescription
    } else {
        self.successMessage = "Лист для відновлення паролю надіслано"
    }
}
```

Рисунок 3.5 – Відновлення паролю

Цей метод є асинхронним і дозволяє Firebase самостійно надіслати email із відповідними інструкціями.

Для персоналізації акаунту реалізовано зміну аватару. Користувач може обрати зображення з галереї пристрою. Обраний файл зберігається. Також наведено приклад коду для вибору зображення.

Було використано патерн MVVM для розділення логіки та інтерфейсу. Усі операції з Firebase виконуються у ViewModel, а View реагує лише на зміну стану. Приклад ViewModel для автентифікації.

Отже, реалізація модуля автентифікації забезпечила надійну взаємодію користувача із системою, дозволивши створення особистого облікового запису, збереження персональних налаштувань і безпечне управління доступом до основного функціоналу застосунку. Використання Firebase значно спростило процес впровадження, дозволивши зосередитися на логіці додатку, не витрачаючи ресурси на створення власного серверного API.

3.3 Розробка модуля налаштувань

Модуль налаштувань є невід’ємною частиною будь-якого сучасного мобільного застосунку, оскільки він забезпечує можливість персоналізації взаємодії користувача з програмою. На рисунку 3.9 зображено схема інтерфейсу сторіни налаштувань (див рис. 3.9).

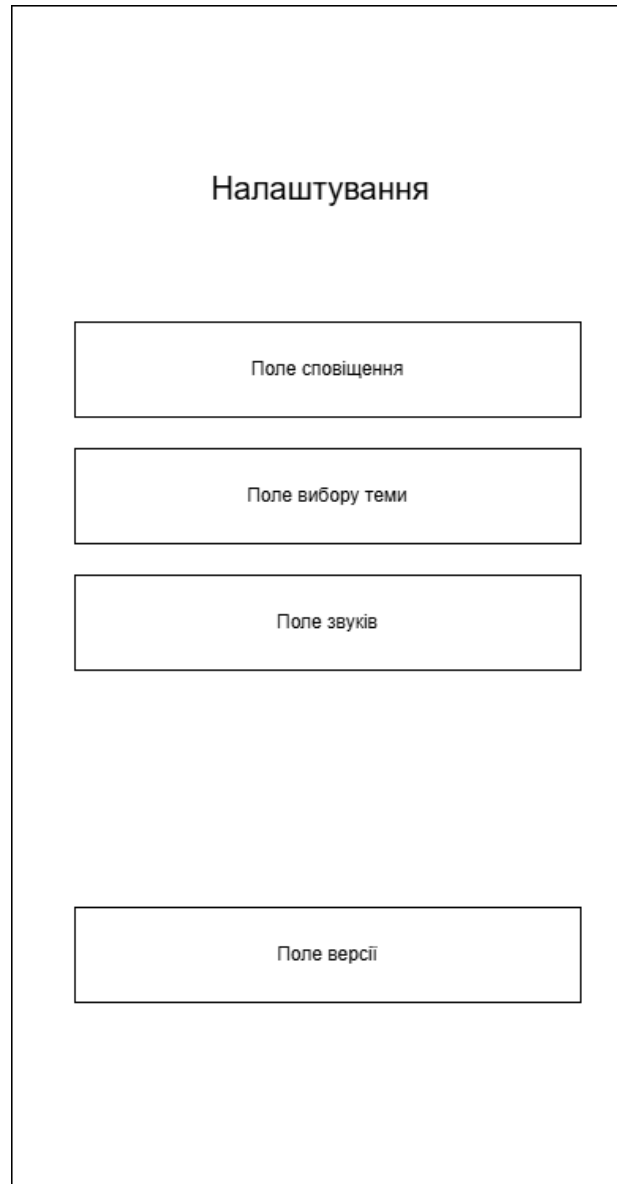


Рисунок 3.9 – Схема інтерфейсу сторіни налаштувань

У межах мобільного застосунку, що призначений для контролю дихання, медитації та релаксації, модуль налаштувань виконує ключову роль у забезпеченні індивідуального користувацького досвіду. Враховуючи специфіку

програми, особливо важливо, щоб користувач мав можливість регулювати параметри, які впливають на візуальне сприйняття, поведінку системи сповіщень, а також звуковий супровід під час практик.

На поточному етапі реалізації система налаштувань містить базовий набір опцій: перемикання темної теми, ввімкнення або вимкнення сповіщень, а також контроль звукових ефектів. Усі параметри зберігаються локально на пристрої користувача, що забезпечує їх збереження навіть після перезапуску застосунку.

Надалі планується розширення функціоналу модуля налаштувань, зокрема додавання можливостей управління часом нагадувань, вибору мови інтерфейсу та інтеграції з системними параметрами iOS для покращеної взаємодії.

Для сучасних мобільних додатків підтримка темної теми є стандартом. Вона покращує візуальне сприйняття інтерфейсу в темному середовищі та знижує навантаження на очі.

Зберігання вибраної теми реалізовано за допомогою `@AppStorage`, що дозволяє зберігати значення у `UserDefaults` (рис 3.10).

```
@AppStorage("isDarkMode") private var isDarkMode = false
```

Рисунок 3.10 – Збереження теми

У налаштуваннях відображається перемикач, що дозволяє увімкнути або вимкнути темну тему (рис 3.11).

```
Toggle("Темна тема", isOn: $isDarkMode)
```

Рисунок 3.11 – Перемикач теми

Для застосування теми до всього інтерфейсу використовується модифікатор `.preferredColorScheme` у `SceneDelegate` або безпосередньо в `View`.

Модуль керування сповіщеннями дозволяє вмикати та вимикати локальні сповіщення. Це необхідно для нагадувань про дихальні вправи, медитації або розтяжки (рис 3.12).

```
@AppStorage("notificationsEnabled") private var notificationsEnabled = true
```

Рисунок 3.12 – Модуль керування сповіщеннями

При зміні значення перемикача система або запитує дозвіл на сповіщення, або скасовує заплановані нагадування (рис 3.13).

```
Toggle("Сповіщення", isOn: $notificationsEnabled)
```

Рисунок 3.13 – Перемикач сповіщень

Оскільки додаток містить функції медитації та дихальних вправ, у деяких режимах можуть використовуватись звукові сигнали

Отже, модуль налаштувань є важливим елементом додатку, що забезпечує базову персоналізацію користувацького досвіду. Навіть з мінімальним набором параметрів темна тема, сповіщення, звуки, він дозволяє користувачеві адаптувати поведінку системи під свої уподобання. У майбутньому планується розширення цього модуля можливостями вибору мови інтерфейсу, створення власного графіка нагадувань та керування фоновою музикою для медитацій.

3.4 Розробка модуля контролю дихання

Модуль контролю дихання є однією з ключових функцій мобільного застосунку. Його основне призначення — допомогти користувачеві підтримувати усвідомлене дихання, зосередитися на ритмічному вдиху та видиху, що сприяє розслабленню, зниженню рівня стресу та покращенню психоемоційного стану.

У межах цього модуля реалізовано декілька режимів дихання: базовий (рівномірне дихання), анксиолітичний та індивідуальний, де користувач самостійно задає параметри дихального циклу. Кожен режим супроводжується візуальними і звуковими індикаторами, які допомагають користувачу тримати правильний темп дихання. Наприклад, графічні кола або хвилі змінюють розмір

або форму відповідно до фаз вдиху, затримки та видиху, а м'які звуки або сигнали подаються для кращої орієнтації в часі.

На рисунку 3.15 зображено схема інтерфейсу сторінки контролю дихання (див рис. 3.15).

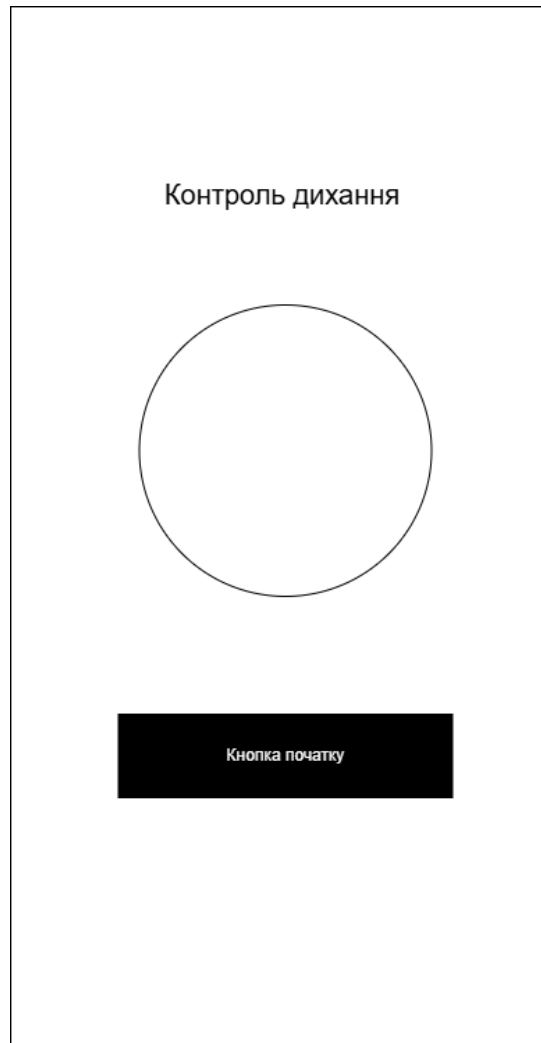


Рисунок 3.15 – Схема інтерфейсу сторінки контролю дихання

Основні функції модуля:

- запуск циклу дихальної практики;
- візуальний таймер для вдиху/видиху;
- повідомлення про завершення сеансу;
- інтерактивна кнопка для початку та паузи.

Інтерфейс користувача

У графічному інтерфейсі центральним елементом є велика кнопка з написом "Вдих" або "Видих", яка змінюється відповідно до поточного етапу дихального циклу. Над кнопкою відображається заголовок "Контроль дихання", а нижче інструкція "Натисніть для початку". Після початку практики текст змінюється відповідно до активного режиму: "Пауза" або "Продовжити". По завершенню сесії з'являється повідомлення "Медитація завершена!".

Функціональність реалізовано у файлі `ContentView.swift` за допомогою `SwiftUI`. Ключові змінні, логіка реалізації та відображення таймера (рис 3.16).

Користувач взаємодіє з модулем через простий інтерфейс з однією кнопкою. Це мінімізує відволікання та дозволяє повністю зосередитися на диханні. Вибір кольору та шрифтів також спрямований на створення заспокійливої атмосфери.

Отже, реалізований модуль дихального контролю є інтуїтивно зрозумілим та технічно простим у використанні. Він відповідає вимогам до мобільних засобів медитації, забезпечуючи базову, але ефективну взаємодію з користувачем. У перспективі можливе розширення модуля за рахунок персоналізації дихальних циклів та збереження статистики.

3.5 Розробка модуля медитацій

Однією з ключових функцій мобільного застосунку є модуль медитацій, який є ключовим у реалізації основної мети системи покращення психоемоційного стану користувача. Цей модуль надає доступ до набору спеціально розроблених заспокійливих практик, що сприяють зниженню рівня стресу, нормалізації емоційного фону, підвищенню концентрації уваги, покращенню якості сну, а також відновленню внутрішньої гармонії. Кожна медитація має індивідуальний дизайн: кольорова палітра, піктограма та опис підібрані, щоб викликати відповідні емоційні асоціації та сприяти візуальному розрізненню варіантів. При виборі однієї з карток відкривається екран виконання

медитації, де користувач бачить детальну інформацію, таймер, а також може керувати процесом.

Крім візуальної складової, модуль забезпечує функціональну гнучкість: у майбутньому передбачено інтеграцію звукових супроводів через AVFoundation, а також підтримку сповіщень за допомогою UserNotifications, які нагадуватимуть про заплановані сеанси медитацій.

Модуль медитацій у додатку не лише виконує роль засобу релаксації, а й слугує повноцінним інструментом психоемоційної саморегуляції, який може бути корисним як у щоденному житті, так і в умовах підвищеного емоційного навантаження (рис 3.18).



Рисунок 3.18 – Схема інтерфейсу сторінки медитації

У межах модуля реалізовано набір інтерактивних медитативних сценаріїв, які були підібрані на основі популярних технік уважності, дихальних вправ і когнітивних підходів до саморегуляції. Кожен сценарій представлено у вигляді візуальної картки, що відображає основну іконку, назву медитації, короткий опис її призначення та тривалість у хвилинах. Завдяки цьому інтерфейс виглядає інтуїтивно зрозумілим і зручним у користуванні.

Медитації поділено за тематикою «Спокій» для глибокої ментальної релаксації, «Енергія» для підвищення життєвого тону, «Сон» для заспокоєння перед відпочинком, «Фокус» для покращення концентрації, «Гармонія» для балансування психоемоційного стану, «Зцілення» для психологічного відновлення. Користувач має змогу швидко обрати практику відповідно до поточних потреб і настрою (рис 3.19).

Для відображення кожної медитації було реалізовано компонент MeditationCard, який відображає:

- іконку медитації;
- назву;
- тривалість;
- короткий опис;
- дію, яка виконується при натисканні.

Ці картки розміщуються у сітці на вкладці "Медитація", що дає змогу швидко вибрати бажану практику. На прикладі, показаному на екрані застосунку, реалізовано такі медитації, як Спокій, Енергія, Зцілення, Сон, Фокус, Гармонія тощо. Кожна з них має власний колірний фон, що сприяє кращому візуальному сприйняттю та асоціативному вибору.

Функціональність окремої медитації

Після вибору користувачем однієї з медитацій, відкривається окремий екран із назвою практики, коротким описом та таймером на відповідну кількість хвилин. Медитація Фокус має тривалість 12 хвилин.

У реалізації модуля використано: SwiftUI для побудови інтерфейсу, AVFoundation для потенційного супроводу звуком, UserNotifications для сповіщень про завершення медитації.

Отже, модуль медитацій поєднує в собі простоту, естетику та функціональність. Він забезпечує користувача гнучким вибором практик відповідно до його потреб, дозволяючи легко інтегрувати релаксацію в повсякденне життя.

3.6 Висновки

У третьому розділі було проведено порівняльний аналіз мов програмування, що можуть бути використані для розробки мобільної системи для моніторингу дихання, медитації та релаксації. Аналіз включав три основні мови програмування Python, Swift та Java, кожна з яких має свої унікальні особливості та застосування в контексті даної розробки.

Першою мовою, що була розглянута, є Python. Вона має значну популярність у сфері наукових досліджень, штучного інтелекту та машинного навчання завдяки великій кількості бібліотек і фреймворків, таких як TensorFlow, PyTorch і OpenCV. Python забезпечує високу швидкість розробки і є ідеальним інструментом для швидкого прототипування алгоритмів, що особливо важливо на етапі тестування і вдосконалення додатку. Однак Python не є оптимальним варіантом для мобільних додатків, особливо для тих, що працюють безпосередньо на пристроях iOS, оскільки для цієї мови потрібні додаткові інструменти і платформи для інтеграції з мобільними пристроями.

Swift, як нативна мова для розробки додатків під iOS, була обрана як основний кандидат для створення цього додатку. Завдяки безпосередній інтеграції з екосистемою Apple, Swift дозволяє максимально ефективно використовувати всі можливості iOS пристроїв, такі як датчики руху, акселерометри та можливості для роботи з мультимедіа. Використання фреймворків CoreML та Vision відкриває широкі можливості для обробки даних і розпізнавання користувача, що є важливим компонентом для розробки

додатків, орієнтованих на здоров'я та релаксацію. Swift також забезпечує високу продуктивність та оптимізацію ресурсів пристроїв Apple, що робить його ідеальним вибором для розробки додатків для цієї платформи.

Останньою мовою, що була розглянута, є Java. Хоча Java є однією з найбільш популярних мов для розробки мобільних додатків на платформі Android, її використання для iOS є менш оптимальним. Java має менше можливостей для інтеграції з усіма апаратними засобами пристроїв Apple, порівняно зі Swift, і може потребувати додаткових інструментів або фреймворків для досягнення необхідної продуктивності та функціональності на платформі iOS.

В результаті проведеного порівняння, для розробки системи для моніторингу дихання, медитації та релаксації було вибрано Swift як основну мову програмування. Цей вибір обґрунтовано найкращою сумісністю з пристроями Apple, високою продуктивністю та можливістю використовувати потужні інструменти для обробки даних і інтеграції з різноманітними сенсорами. Python може бути використаний для серверної частини додатку або для розробки прототипів, а Java не є оптимальним вибором для даної платформи.

Крім вибору мови програмування, у розділі було обґрунтовано архітектуру програмного продукту шляхом поділу його на логічно завершені функціональні модулі. Кожен модуль відповідає окремій групі задач, що дозволяє досягти гнучкості, масштабованості та зручності подальшої підтримки коду.

Модуль медитацій є ключовим функціональним компонентом, який реалізує доступ користувача до різноманітних медитативних сценаріїв. Кожен сценарій представлений окремою карткою з описом, рекомендаціями до використання, візуальним оформленням та інтегрованим таймером. Завдяки цьому користувач може обирати медитації відповідно до власного психоемоційного стану для зменшення тривожності, покращення сну, відновлення концентрації або просто короткого перезавантаження під час робочого дня. Усі сценарії супроводжуються плавною анімацією, звуковими сигналами та вібрацією для більш глибокого занурення.

Модуль дихання забезпечує візуально-часову координацію дихального ритму. У межах цього модуля реалізовано інтерактивний анімований елемент, який керує фазами вдиху, видиху та паузи, з урахуванням встановленої користувачем тривалості. Це дозволяє застосовувати техніки дихальної гімнастики, які використовуються в медичній практиці та психології для боротьби зі стресом, панічними атаками й загальним напруженням.

Модуль фокусування виконує функцію таймера продуктивності, де користувач може встановити тривалість фокус-сесії. Цей модуль орієнтований на техніки тайм-менеджменту, що дозволяє підвищити ефективність роботи, навчання чи творчої діяльності. Після завершення сесії користувач отримує повідомлення або звуковий сигнал про досягнення встановленого часу концентрації.

Модуль профілю містить особисті налаштування користувача, дозволяє зберігати індивідуальні параметри сесій, переглядати історію використання додатку, а також статистику прогресу та досягнень.

Модуль налаштувань надає доступ до конфігурації інтерфейсу та функціоналу. Користувач може обирати темну або світлу тему оформлення, керувати гучністю, вібрацією, частотою нагадувань, мовою інтерфейсу та ін. Це дозволяє адаптувати додаток до індивідуальних потреб та звичок.

У результаті аналітичного обґрунтування та технічного планування, було прийнято доцільне рішення щодо вибору Swift як основної мови реалізації, а також розроблено логічно структуровану архітектуру програмного продукту на основі модульного підходу. Такий підхід дозволяє досягти високої якості програмного забезпечення, забезпечити масштабованість, підтримку та зручне розширення функціоналу в майбутньому.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Для перевірки якості та надійності роботи системи моніторингу дихання, медитації та релаксації було проведено кілька послідовних етапів тестування. Основною метою цього процесу стало оцінювання ефективності функціонування кожного окремого модуля застосунку, а також перевірка їх коректної взаємодії в межах єдиної програмної структури. Зокрема, тестування охоплювало модуль дихальних вправ, медитаційні функції, механізми релаксації, систему сповіщень і налаштувань, а також візуальні та аудіоеlementи. Усі етапи перевірки були спрямовані на виявлення потенційних збоїв, затримок у роботі або конфліктів між модулями. За результатами тестування встановлено, що система демонструє стабільну роботу в інтегрованому режимі, забезпечуючи безперебійну взаємодію всіх функціональних компонентів [19].

Після завершення основного етапу розробки мобільного додатку для контролю дихання, медитації та релаксації було проведено комплексне тестування його функціональності, стабільності роботи та відповідності вимогам користувацького досвіду. Тестування здійснювалося у кілька етапів з метою виявлення можливих недоліків та помилок, а також забезпечення максимальної якості кінцевого продукту.

Перший етап тестування полягав у внутрішній перевірці працездатності додатку безпосередньо у середовищі розробки Xcode. Було створено кілька емуляторів різних моделей iPhone та iPad для моделювання роботи додатку на пристроях із різними характеристиками, розмірами екранів та версіями операційної системи. За допомогою емуляторів перевірялися базові сценарії використання, зокрема запуск додатку, правильність відображення елементів інтерфейсу, виконання функцій нагадувань про дихальні вправи, медитації та періодичне сповіщення користувача про необхідність відпочинку. Особлива увага приділялася стабільності роботи додатку при різкому закритті, згортанні у

фоновий режим та поверненні до активної роботи. Також тестувалася реакція системи на переривання, вхідний дзвінок під час активної сесії медитації.

Другий етап тестування полягав у використанні фізичних пристроїв. Для цього додаток було встановлено на реальні моделі iPhone 8, iPhone 11 та iPad 9-го покоління. Використання реальних пристроїв дозволило точніше перевірити, як додаток поводить себе у реальному середовищі експлуатації у тому числі за умови різних рівнів заряду батареї, активного використання інших додатків у фоновому режимі, зміни мережевих підключень та режимів енергозбереження. Тестування на фізичних пристроях також дозволило переконатися у коректній роботі системи сповіщень та перевірити точність спрацьовування нагадувань згідно із заданим розкладом.

На наступному етапі було організовано тестування через платформу TestFlight. Було сформовано обмежене коло бета-тестерів, яке включало як технічних спеціалістів, так і потенційних кінцевих користувачів різних вікових груп. Це дозволило отримати зворотний зв'язок не лише щодо технічної стабільності роботи додатку, а й щодо зручності інтерфейсу, логіки взаємодії користувача з системою, а також загального рівня задоволення від використання. Під час цього етапу тестувальники відзначали як позитивні сторони зрозумілі налаштування режимів, приємний інтерфейс, ненав'язливі нагадування, так і можливі напрямки покращення, оптимізацію часу реагування на сповіщення при великій кількості паралельних задач у системі.

Окрему увагу в ході тестування було приділено питанням продуктивності. Було проведено серію замірів швидкості завантаження додатку, часу переходу між екранами та плавності анімацій. Для цього використовувалися інструменти профілювання вбудовані у Xcode. Результати тестування показали, що додаток витрачає мінімальну кількість ресурсів системи та не створює додаткового навантаження на батарею пристрою, що є особливо важливим для програм щоденного використання.

Також виконувалося тестування поведінки додатку за різних нестандартних умов, таких як вимушене завершення процесу, блокування

пристрою під час активної сесії медитації або відключення Інтернету під час отримання сповіщення. Додаток показав високу стійкість до подібних ситуацій, зберігаючи дані користувача та продовжуючи роботу без втрати важливої інформації.

Тестування модуля контролю дихання [20]. Протестовано, як система відображає показники дихання користувача в реальному часі, а також як вона дає сповіщення при досягненні високих чи низьких показників. Тестування також включало перевірку реакції програми на зміни в ритмі дихання під час виконання дихальних вправ.

Тестування модуля медитації та релаксації було проведено з метою перевірки його функціональності та відповідності очікуваній поведінці. Зокрема, було протестовано коректність відображення інтерфейсу під час виконання вправ на релаксацію, включаючи перевірку елементів управління, анімацій та загальної зручності взаємодії. Особливу увагу приділено механізму генерації сповіщень система успішно формує нагадування про необхідність виконання медитаційних сесій у заданий користувачем час. Окрім цього, оцінено ефективність візуального супроводу та звукових елементів мелодій, фонів, дихальних сигналів, які відіграють важливу роль у створенні сприятливого середовища для релаксації. За результатами тестування модуль показав стабільну роботу та позитивний вплив на користувацький досвід.

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, оскільки дозволяє своєчасно виявити помилки, недоліки та невідповідності в роботі системи. Воно дає змогу ідентифікувати різні типи дефектів — від логічних помилок у коді до порушення вимог до функціональності, інтерфейсу або продуктивності. За допомогою тестування можна оцінити, наскільки програмний продукт відповідає очікуванням користувача та вимогам, визначеним на етапах проектування.

На рисунку 4.1 зображено загальний принцип тестування, який демонструє процес взаємодії між вхідними даними, програмним модулем, тестами та очікуваним результатом. Тест порівнює фактичний результат виконання

програми з наперед визначеним правильним результатом. Якщо ці результати збігаються — тест вважається пройденим; якщо ж є розбіжність, це свідчить про наявність помилки, яку необхідно виправити.

Така візуалізація дозволяє краще зрозуміти суть тестування як процесу перевірки відповідності між тим, що програма робить насправді, та тим, що вона має робити відповідно до специфікацій. Це забезпечує розробників зворотним зв'язком про якість коду, знижує ризик помилок у фінальній версії продукту та сприяє підвищенню надійності програмної системи в цілому.

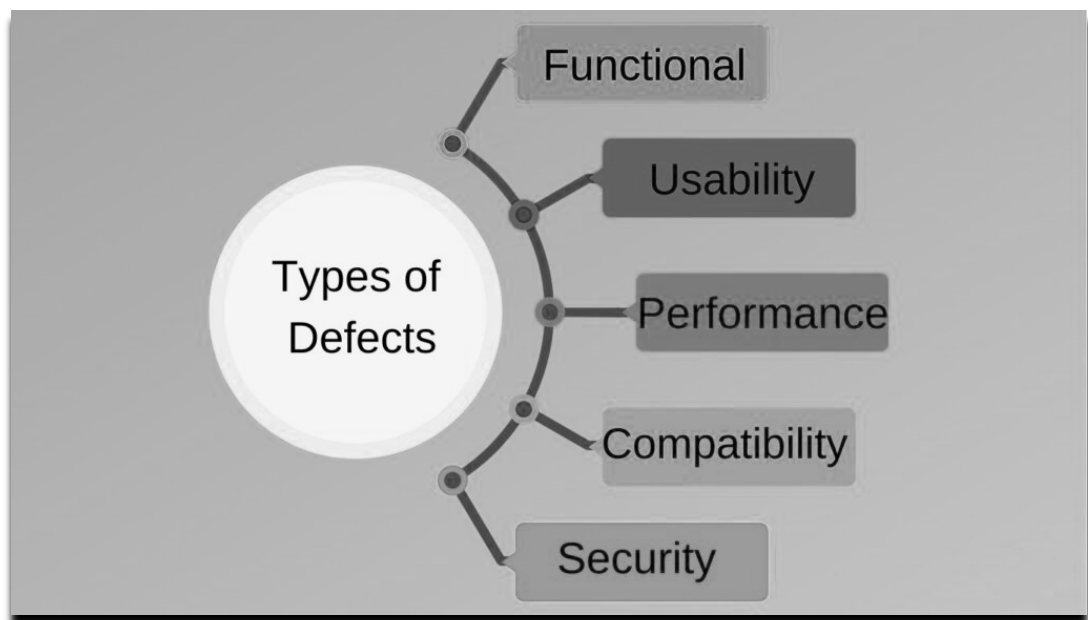


Рисунок. 4.1 – Принцип тестування

Вони є інструментом, який допомагає знайти проблеми та звернути на них увагу. Це означає, що тести мають бути структуровані та документовані, щоб їх можна було повторити в будь-який момент і отримати однакові результати. Це важливо для забезпечення надійності та якості ПЗ.

4.2 Тестування розробленого програмного застосунку

Перевірка стабільності роботи програми здійснювалась шляхом тестування застосунку на різних моделях iPhone, що працюють під керуванням різних версій операційної системи iOS. Такий підхід дозволив виявити

потенційні недоліки з продуктивністю, адаптивністю інтерфейсу та загальною сумісністю програми з різними пристроями Apple. Особливу увагу приділено коректному відображенню графічних елементів, швидкості завантаження екранних форм, а також стабільності роботи основних функцій упродовж тривалого використання. За результатами перевірки було підтверджено, що застосунок стабільно працює в широкому діапазоні конфігурацій, забезпечуючи позитивний користувацький досвід.

Якщо дані були введені неправильно, при натисненні кнопки «Увійти» з'явиться банер з повідомленням про помилку (рис. 4.2).

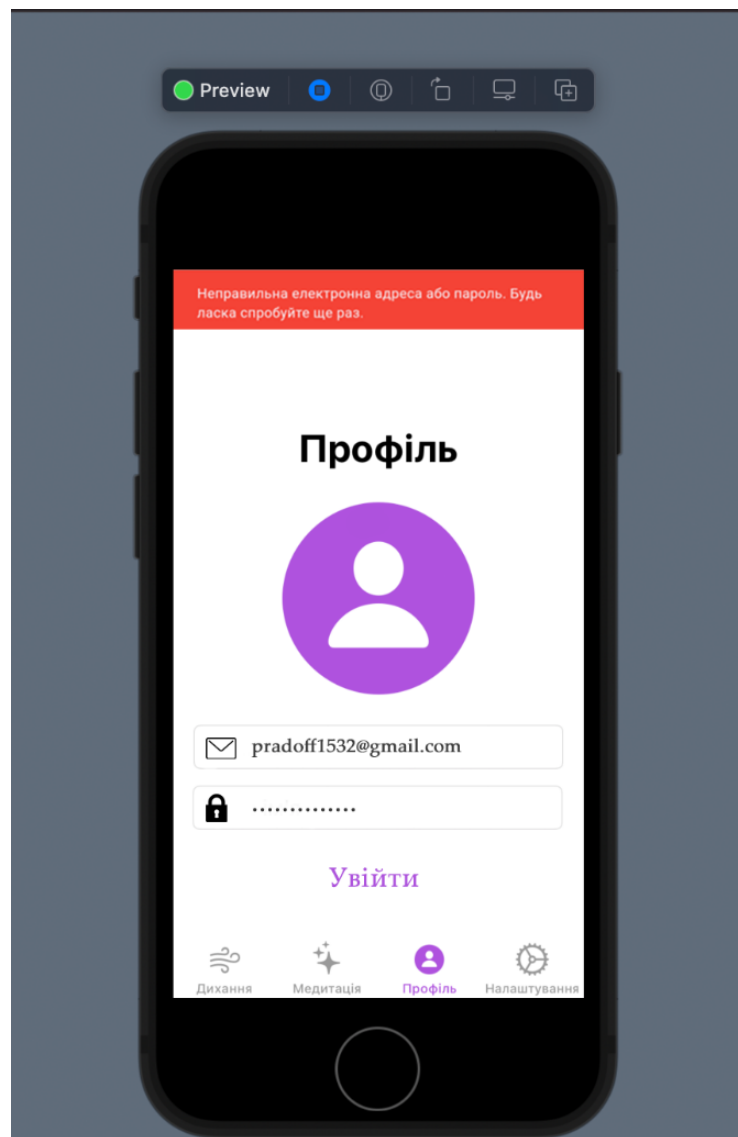


Рисунок. 4.2 – Банер з повідомленням про помилку

У межах функціонального тестування було перевірено роботу вкладки «Профіль», яка є однією з ключових частин мобільного застосунку. Ця вкладка забезпечує взаємодію користувача з елементами персоналізації, включаючи введення імені, перегляд та зміну зображення профілю.

Під час тестування було перевірено відображення вкладки на мобільному пристрої, правильність розміщення всіх елементів інтерфейсу, а також коректність функціоналу. Зокрема, було протестовано введення імені у відповідне поле. Застосунок правильно обробляє введення тексту українською мовою, а поле адекватно реагує на дії користувача. Поле для вводу імені користувача зображено на рисунку 4.3 (див рис. 4.3).

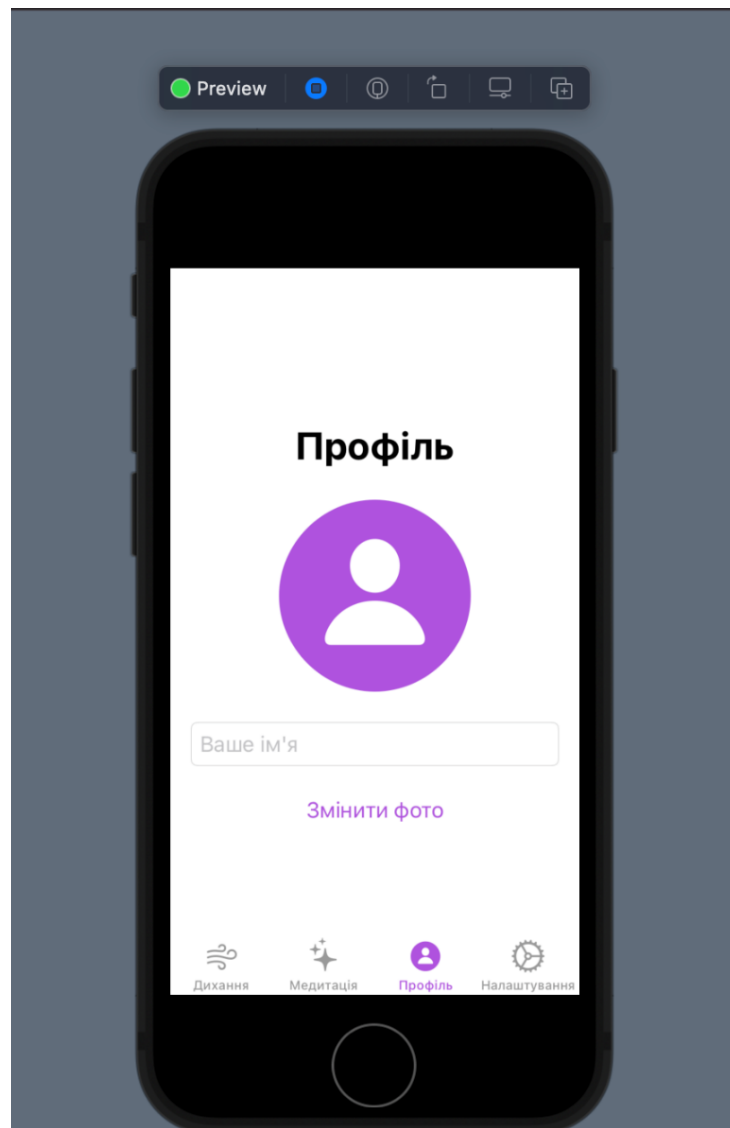


Рисунок. 4.3 – Профіль

Крім того, було перевірено роботу кнопки «Змінити фото». Після її натискання відкривається інтерфейс вибору зображення, що відповідає стандартному системному діалоговому вікну iOS. Функціонал реагує стабільно, без збоїв. Випадків аварійного завершення роботи застосунку в ході тестування не зафіксовано (див рис. 4.4).

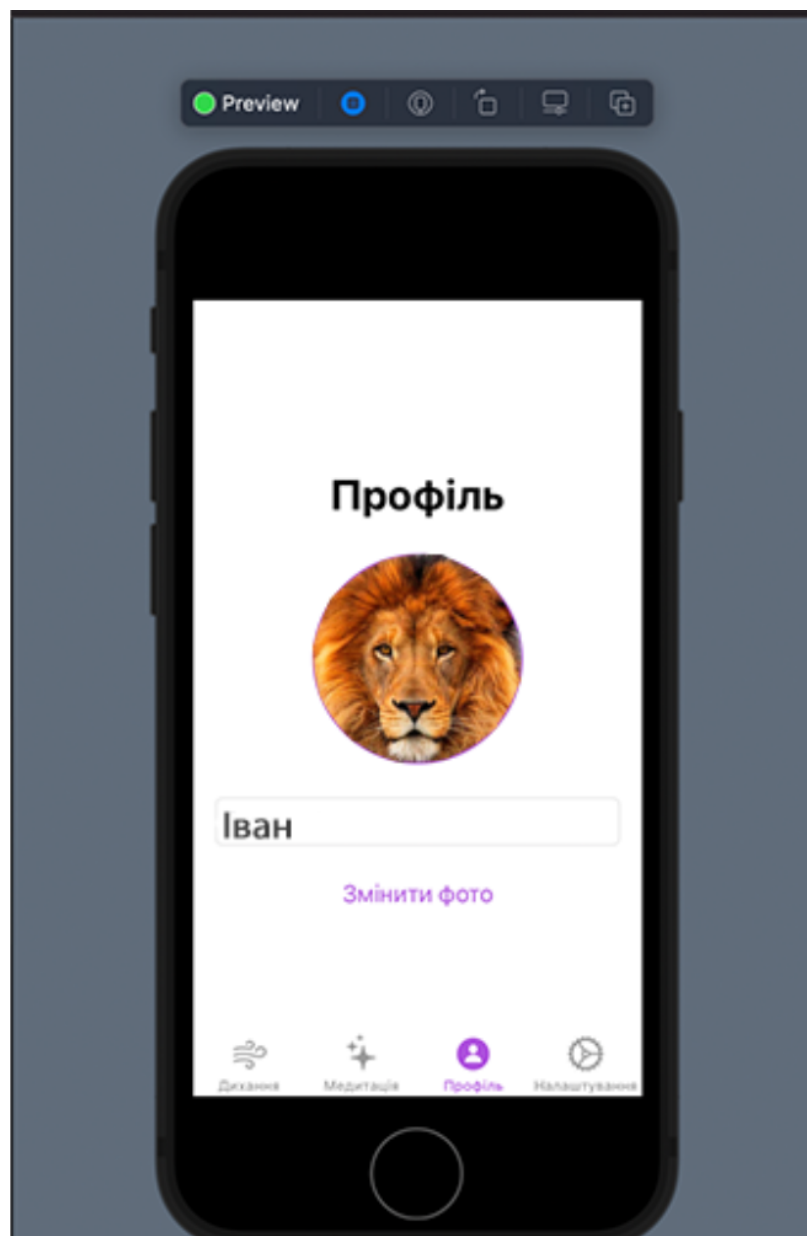


Рисунок. 4.4 – Профіль із фото та ім'ям

У нижній частині вкладки також було протестовано панель навігації між іншими основними розділами застосунку. Перемикання між вкладками

«Дихання», «Медитація», «Профіль» і «Налаштування» відбувається плавно та без затримок.

Після тестування вкладки користувача можна зробити висновок, що її функціонал реалізовано коректно, а інтерфейс відповідає очікуванням з погляду зручності та стабільності.

Під час тестування вкладки «Налаштування» було перевірено її функціональність, відображення елементів інтерфейсу, стабільність роботи та реакцію на дії користувача. Ця вкладка дозволяє користувачеві керувати основними параметрами застосунку: вмикати або вимикати сповіщення, змінювати тему інтерфейсу, а також контролювати звукові ефекти.

У процесі тестування було перевірено: Візуальну коректність інтерфейсу. Всі елементи відображаються правильно: заголовок, перемикачі та текстові мітки сповіщення, тема, звуки, а також блок з інформацією про версію додатку. Роботу перемикачів. Кожен з трьох перемикачів реагує на натискання миттєво. Після зміни стану перемикача зміни зберігаються під час поточної сесії. Перемикачі змінюють свій вигляд зелене/сіро-біле положення, що дозволяє користувачу одразу зрозуміти активний або неактивний стан налаштування. Перевірку стабільності. Жодних збоїв, зависань або аварійних завершень застосунку в результаті взаємодії з елементами вкладки зафіксовано не було. Версію застосунку. У нижній частині вкладки коректно відображається номер поточної версії 1.0.0, що відповідає фактичній версії з Xcode.

Також було перевірено взаємодію вкладки «Налаштування» з іншими екранами програми, зокрема з головним меню, екранами дихання, медитації та профілю. Усі переходи виконуються швидко та без помітних затримок або збоїв, що свідчить про правильну реалізацію навігаційної логіки. Це підтверджує коректну інтеграцію цієї частини застосунку та забезпечує зручність у користуванні (див. рис. 4.5).

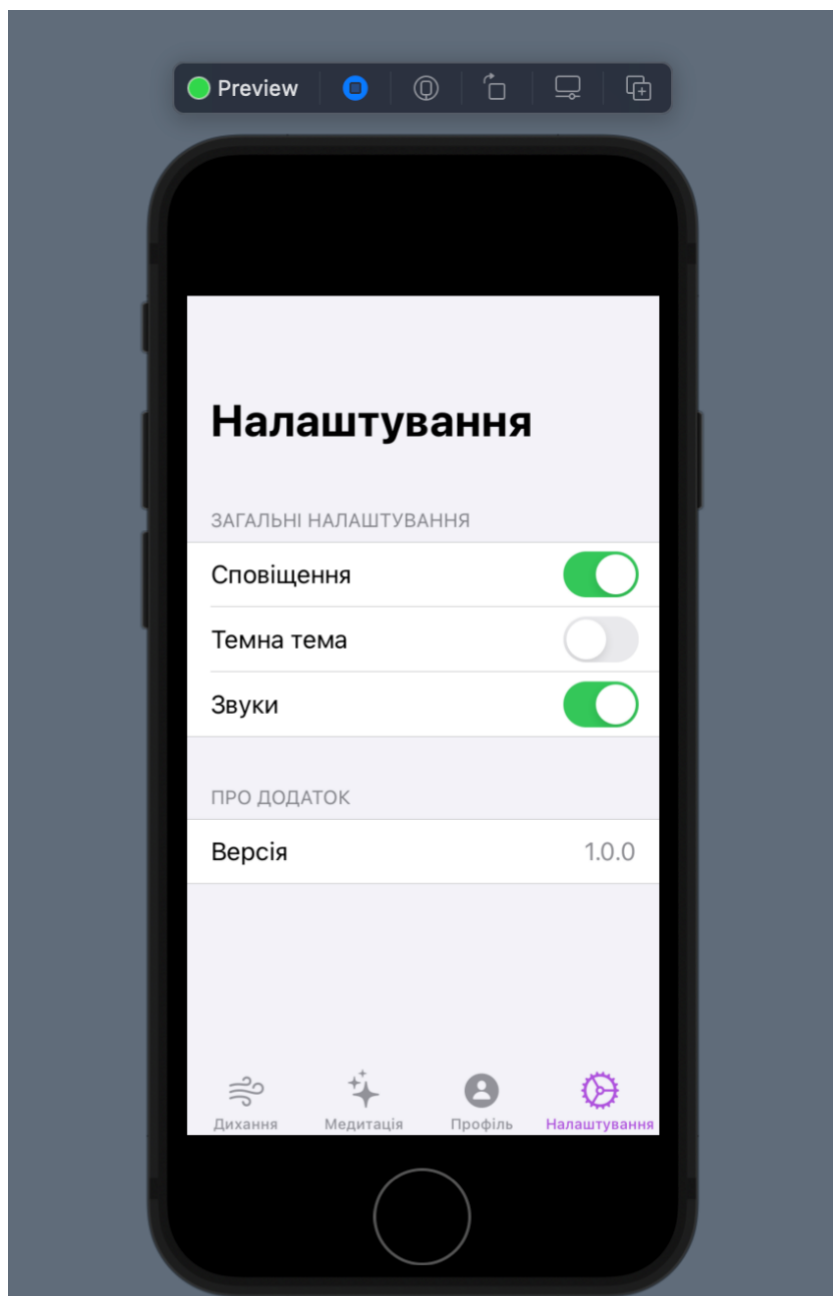


Рисунок. 4.5 – Налаштування

Результати тестування показали, що вкладка «Налаштування» реалізована правильно, її функціонал повністю відповідає заявленим вимогам, а інтерфейс є зручним та інтуїтивно зрозумілим.

Вкладка «Дихання» є однією з ключових функціональних частин мобільного застосунку. Вона призначена для допомоги користувачу в організації дихальних вправ з метою розслаблення та зниження рівня стресу.

Під час тестування були перевірені такі аспекти.

Відображення інтерфейсу. У верхній частині екрана коректно відображається заголовок "Контроль дихання", що чітко визначає функціонал вкладки. У центрі екрану розміщується кругла анімаційна кнопка з написом, що динамічно змінюється відповідно до етапу дихального циклу «Видих».

Функціональність дихального циклу. Після натискання на кнопку з написом «Натисніть для початку» запускається анімація, яка змінює надписи відповідно до дихального сценарію: вдих, затримка, видих. Анімація циклічна та стабільна, тривалість кожного етапу відповідає налаштуванням у коді, а перемикання між фазами відбувається без затримок.

Змінність тексту на кнопці. Під час активного сеансу змінюється текст з "Пауза" на "Продовжити", що дозволяє користувачу зупинити або відновити вправу. Це забезпечує гнучкість використання функції.

Реакція на взаємодію користувача реалізована коректно. Натискання кнопок супроводжується плавною анімацією, без затримок чи збоїв у роботі інтерфейсу. Після завершення вправи система відображає повідомлення "Медитація завершена!" разом із кнопкою "Закрити", яка виконує свою функцію — закриває поточний екран і повертає користувача на головну сторінку застосунку, забезпечуючи зручний та логічний завершальний етап сеансу.

Крім того, під час тестування було перевірено коректність роботи анімацій, що супроводжують дихальні вправи, а також точність відображення вказівок у текстовому та голосовому форматах. Усі елементи інтерфейсу працювали узгоджено: анімація чітко відображала ритм вдиху та видиху, а звукові підказки з'являлися у потрібний момент, не викликаючи плутанини чи затримок. Також перевірялося, як система реагує на переривання сесії або її дострокове завершення — у таких випадках застосунок стабільно повертав користувача до головного меню без помилок чи збоїв. Таким чином, можна зробити висновок, що вкладка «Дихання» не лише відповідає заявленим функціональним вимогам, але й забезпечує комфортне, ефективне та інтуїтивно зрозуміле користування для людей з різним рівнем підготовки (див. рис. 4.6).

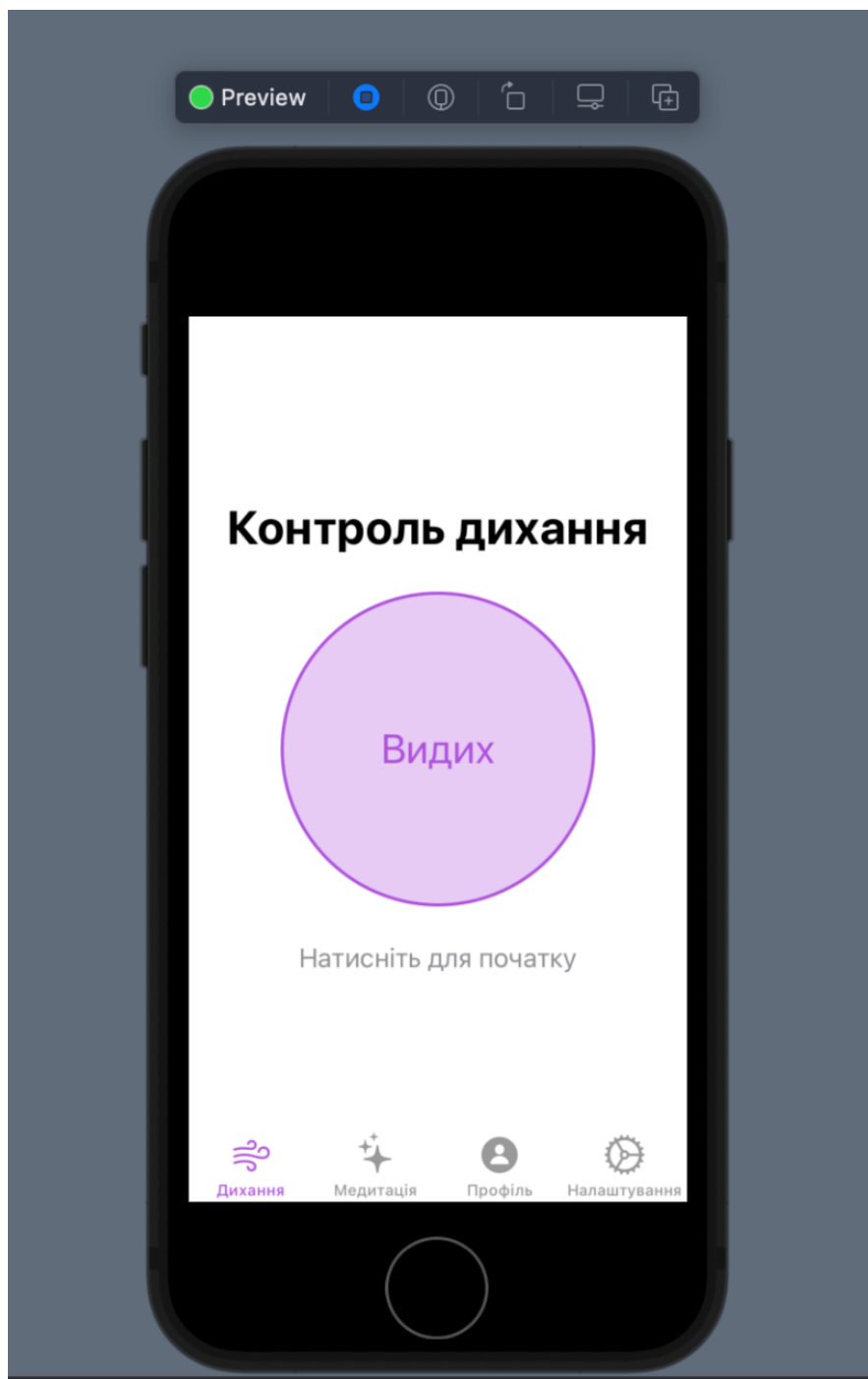


Рисунок. 4.6 – Контроль дихання

Зовнішній вигляд. Візуальні елементи оформлено з урахуванням принципів простоти та естетики: використовуються приємні кольори, великі шрифти, округлі кнопки. Усе це сприяє розслабленню користувача та відповідає тематиці дихальних вправ. Фінальний вигляд інтерфейсу застосунку наведено на рисунку 4.7 (див рис. 4.7).



Рисунок. 4.7 – Фінальний вигляд застосунку

Кожен тип медитації, представлений у мобільному застосунку, характеризується власними попередньо визначеними параметрами, які враховують тривалість сесії, тип аудіосупроводу, рівень візуальної концентрації та інші особливості практики. Ці налаштування були розроблені з урахуванням

різних потреб користувачів від новачків до більш досвідчених практиків. Всі параметри для кожного виду медитації систематизовано та наведено у таблиці 4.1, що дозволяє зручно порівняти варіанти і обрати найбільш підходящий режим для конкретної ситуації або стану користувача.

Таблиця 4.1 — Типи медитації

Назва	Тривалість	Тема	Фон
Спокій	10 хв	Глибоке дихання	Нічне небо
Енергія	15 хв	Активізація уваги	Сонце
Зцілення	20 хв	Внутрішнє очищення	Природа
Сон	25 хв	Релаксація перед сном	Місяць
Фокус	12 хв	Концентрація	Мінімалізм
Гармонія	18 хв	Відновлення балансу	Абстракція

Перевірка роботоздатності вкладки "Фокус" була проведена з метою оцінки стабільності та коректності її функціонування в межах мобільного застосунку. Під час тестування особливу увагу приділено активації режиму фокусування, відображенню інтерфейсу, запуску супровідних елементів звуків, візуальних підказок, а також правильній обробці користувацьких дій початку, паузи та завершення сеансу.

Окремо було протестовано, як система обмежує відволікаючі фактори, вимкнення сповіщень або затемнення екрана, відповідно до призначення цієї функції. За результатами перевірки вкладка "Фокус" показала стабільну роботу без збоїв, забезпечуючи належну підтримку концентрації користувача під час виконання вправ або роботи. Перевірка роботоздатності вкладки фокус наведено на рисунку 4.8 (див рис. 4.8).

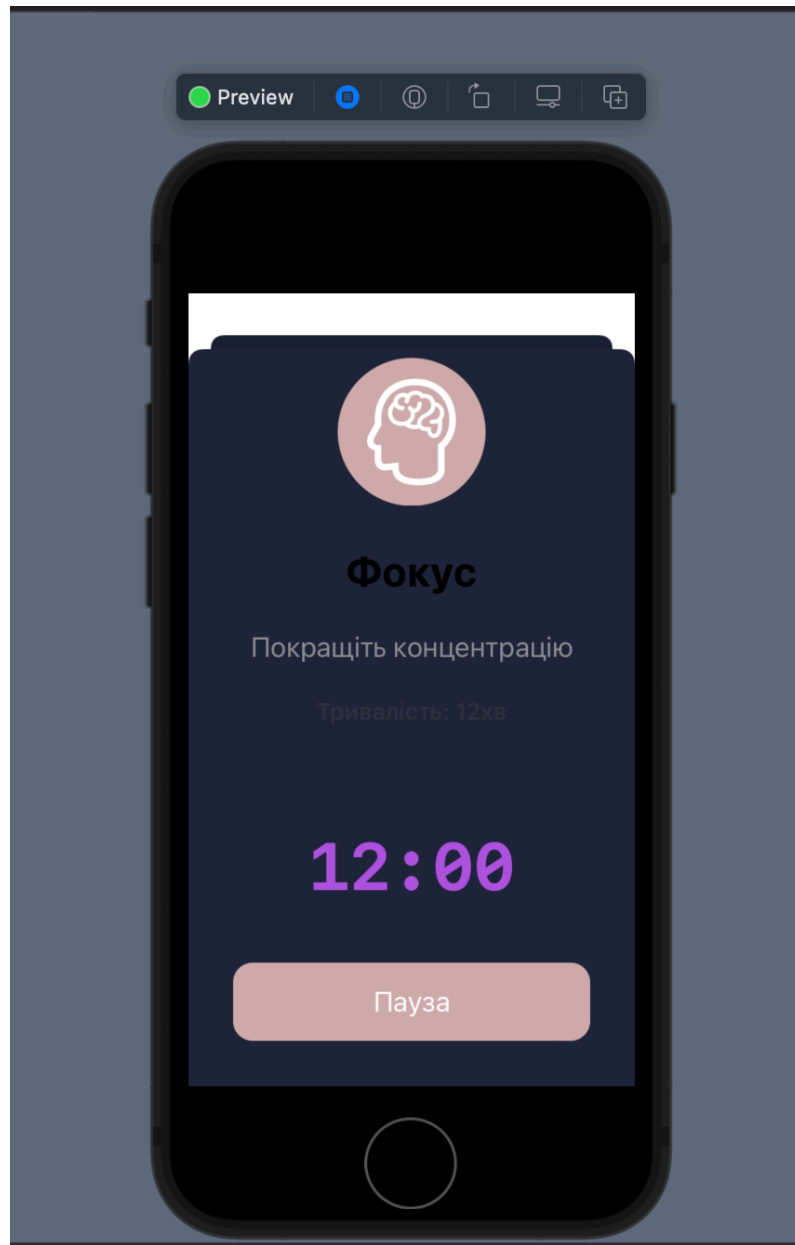


Рисунок. 4.8 – Перевірка роботоздатності вкладки Фокус

Після завершення сеансу медитації, коли таймер відліку досягає нульового значення (00:00), система автоматично повідомляє користувача про завершення вправи. На екрані з'являється відповідне повідомлення з інструкцією, що сесія завершена, та пропонується натиснути кнопку “Вихід”. Натискання цієї кнопки завершує активний сеанс, закриває поточний екран медитації та повертає користувача до головного інтерфейсу застосунку. Такий підхід забезпечує чітке завершення практики без плутанини, дозволяє користувачу м'яко вийти з

медитативного стану та підготуватись до подальшої взаємодії з додатком (див. рис. 4.9).

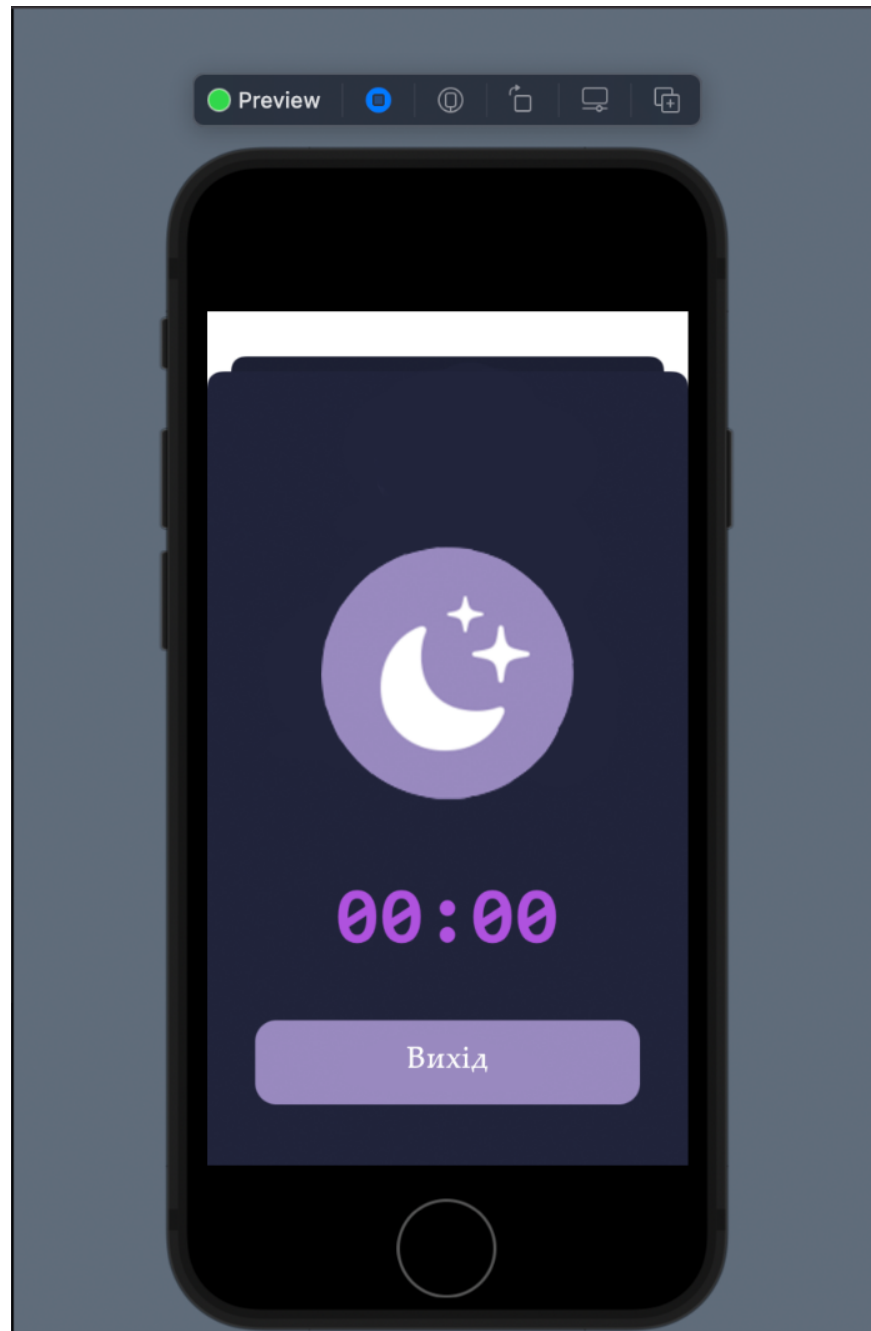


Рисунок. 4.9 – Закінчення медитації

Програма повинна стабільно працювати без зависань чи витоків пам'яті, а також забезпечувати ефективну взаємодію між модулями на різних пристроях.

У результаті проведеного тестування були виявлені та усунені незначні недоліки, серед яких: некоректне відображення тексту на пристроях з

нестандартними розмірами екранів та затримка у відображенні деяких сповіщень після перезавантаження пристрою. Після виправлення цих недоліків додаток був повторно протестований і успішно пройшов фінальну перевірку на відповідність всім заявленим вимогам.

Процес тестування дозволив не лише перевірити правильність роботи всіх функціональних компонентів, але й суттєво підвищити загальну якість продукту, що створює передумови для його успішної експлуатації широким колом користувачів.

4.3 Розробка інструкції користувача

Для зручності користувачів була розроблена інструкція, яка містить такі основні розділи. Огляд програми. Коротке пояснення основних функцій програми, таких як контроль дихання, медитація та релаксація. Початок роботи з програмою. Покрокова інструкція з налаштування програми, підключення до пристроїв та першого запуску. Виконання дихальних вправ. Інструкції щодо використання програми для виконання дихальних вправ, включаючи підказки щодо тривалості вдихів і видихів. Медитаційні вправи. Опис доступних медитаційних вправ, а також способи налаштування їх тривалості та інтенсивності для максимального ефекту. Налаштування програми. Як змінювати параметри програми, включаючи звукові сповіщення, інтерфейс та інші настройки для підвищення комфорту використання. Поради щодо використання. Рекомендації для покращення результатів при виконанні дихальних і медитаційних вправ, а також корисні поради для підтримки релаксації.

4.4 Підтримувані пристрої, методи встановлення та розповсюдження додатку

Розроблена мобільна система для контролю дихання, медитації та релаксації спеціально створюється для екосистеми компанії Apple, а саме для

роботи на пристроях, що працюють під управлінням операційної системи iOS та iPadOS. Основними типами пристроїв, на яких може функціонувати додаток є:

iPhone: смартфони Apple, починаючи з моделей, що підтримують мінімальні технічні вимоги додатку. Для забезпечення коректної роботи рекомендується підтримка пристроїв з процесорами серії A11 Bionic та новішими, тобто з iPhone 8 і новіших моделей.

iPad: планшети Apple під керуванням iPadOS, починаючи з iPad п'ятого покоління та новіших, які забезпечують достатню обчислювальну потужність для стабільної роботи додатку.

iPod Touch: хоча цей пристрій вважається застарілим, остання його версія може підтримувати роботу додатку за умови відповідності версії операційної системи.

Для коректного функціонування додатку встановлені мінімальні технічні вимоги, продемонстровані на таблиці 4.2

Таблиця 4.2 — Мінімальні технічні вимоги для роботи додатку

Параметр	Мінімальна вимога
Операційна система	iOS 15 або новіша версія
Підтримувані пристрої	iPhone 8 і новіші; iPad п'ятого покоління і новіші
Вільне місце на пристрої	Щонайменше 100 МБ
Підключення до Інтернету	Обов'язкове для завантаження та оновлення
Додаткові апаратні можливості	Акселерометр, таймери, базові сенсори пристрою

Найбільш стандартним та безпечним способом розповсюдження додатку є використання офіційного магазину App Store. Процес завантаження для проходить такі кроки:

1. Користувач заходить в App Store через свій пристрій.
2. У пошуковому полі вводиться назва додатку.
3. Вибирає потрібний додаток зі списку результатів пошуку.

4. Натискає кнопку Завантажити або Отримати.
5. Після встановлення додаток з'являється на головному екрані пристрою та готовий до використання.

Додаток також може містити в описі посилання на вебсайт розробника або матеріали, які детально пояснюють його функціонал. Щоб додаток став доступним широкому загалу, розробник має пройти певні обов'язкові етапи. Участь в програмі розробника Apple є обов'язковою умовою для розповсюдження додатків через App Store. Вимоги для публікації оформлення облікового запису розробника індивідуального або організації, щорічна оплата внеску в розмірі \$99 може змінюватися залежно від країни. App Store Connect це веб-портал для управління додатками. Тут потрібно зареєструвати новий додаток, вказати інформацію про продукт: назву, опис, категорії Здоров'я та фітнес, ключові слова, політику конфіденційності, додати маркетингові матеріали: скріншоти, прев'ю-відео, іконки.

Apple уважно перевіряє всі додатки на предмет відповідності технічним стандартам, відсутності шкідливого коду або порушення політики конфіденційності, коректності функціонування на різних пристроях і версіях iOS.

Тривалість перевірки може змінюватися від декількох годин до декількох днів, залежно від завантаженості служби.

Після проходження перевірки додаток стає доступним для користувачів App Store у всьому світі або в обраних регіонах. До моменту офіційної публікації існує можливість обмеженого розповсюдження через TestFlight офіційний сервіс Apple для тестування додатків. Можна запросити до 10 000 тестувальників через унікальне посилання або електронну пошту. Доступ до тестової версії діє обмежений час. Відгуки від тестувальників допомагають виявити помилки, удосконалити інтерфейс, оптимізувати роботу системи. Ad Hoc Distribution. Поширення збірок на обмежену кількість пристроїв за допомогою реєстрації UDID пристроїв. Enterprise Distribution Поширення додатку в межах закритої корпоративної мережі.

Публікація додатку в офіційному магазині дає розробнику та кінцевому користувачу такі переваги як, гарантія якості: користувач впевнений, що додаток перевірено на безпеку та відповідність вимогам, доступ до глобального ринку: App Store працює у понад 175 країнах, можливості монетизації: прямі продажі додатку, покупки всередині додатку, реклама, легке оновлення: регулярне випускання нових версій і виправлень без потреби видаляти додаток, маркетингова підтримка: можливість потрапити в тематичні підбірки App Store, що значно підвищує кількість завантажень.

Отже, розроблений мобільний додаток орієнтований на сучасні пристрої екосистеми Apple, що працюють під управлінням операційних систем iOS та iPadOS. Визначення мінімальних технічних вимог дозволяє гарантувати стабільну і коректну роботу додатку на більшості актуальних моделей iPhone та iPad, забезпечуючи тим самим широкий охопит користувачів.

Процес розповсюдження додатку через офіційний магазин App Store забезпечує як високу якість продукту, так і відповідність усім стандартам безпеки та конфіденційності, встановленим компанією Apple. Для розробника важливим є дотримання усіх етапів публікації від реєстрації в програмі Apple Developer Program до успішного проходження перевірки на відповідність вимогам App Store.

Можливість попереднього тестування додатку через платформу TestFlight дозволяє виявити та виправити недоліки ще до моменту публікації, що суттєво підвищує якість фінального продукту. Зі сторони користувача процес встановлення додатку є простим і зрозумілим, що сприяє зручності користування з перших хвилин після завантаження.

Дотримання технічних вимог, правильна організація процесу тестування та розповсюдження через App Store створюють необхідні умови для успішної інтеграції додатку у повсякденне життя користувачів, сприяючи реалізації його основної мети підтримці здоров'я, зменшенню стресу та покращенню загального стану людини за допомогою інструментів для дихання, медитації та релаксації.

4.5 Висновки

У цьому розділі було розглянуто процес тестування програми та її основних модулів, що включають контроль дихання, медитаційні вправи та функції релаксації. Програма продемонструвала хорошу стабільність і точність в роботі при різних умовах освітлення та на різних пристроях. Також було підготовлено детальну інструкцію для користувачів, яка дозволяє легко освоїти програму та отримати максимальний ефект від її використання.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено мобільну систему під iOS для контролю дихання, медитації та релаксу. Застосунок реалізовано у вигляді окремих функціональних модулів, що забезпечують такі можливості:

- реєстрацію та авторизацію користувачів з безпечним збереженням облікових даних;
- запуск і контроль дихальних вправ із візуальним та звуковим супроводом для підвищення ефективності практики;
- функціонал для медитації — з можливістю вибору тривалості;
- реалізацію режимів релаксації та фокусування, які допомагають знизити рівень стресу та підтримувати концентрацію;
- профіль користувача.

Для реалізації системи було розроблено метод, архітектурну модель та алгоритми роботи застосунку. Застосовано мову програмування Swift у середовищі Xcode із використанням сучасних фреймворків iOS для побудови зручного інтерфейсу та взаємодії з системними можливостями пристрою.

У першому розділі кваліфікаційної роботи проведено аналіз існуючих мобільних рішень для дихальних практик і медитацій, виявлено їхні сильні сторони та недоліки. У другому розділі розроблено метод і логічну структуру застосунку, спроектовано основні алгоритми функціонування модулів. Третій розділ присвячено вибору інструментів реалізації, розробці архітектури програмного забезпечення та користувацького інтерфейсу. У четвертому розділі виконано тестування застосунку, підтверджено його працездатність і відповідність встановленим функціональним вимогам. Також надано інструкції щодо встановлення застосунку на пристрій користувача та описано мінімальні вимоги до системи.

Розроблена система може бути використана як самостійний мобільний засіб для підтримки ментального здоров'я, зниження стресу та покращення загального самопочуття користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Swift. [Електронний ресурс] – Режим доступу до ресурсу: https://book.swift.org.ua/book/1_language_guide/0_the_basics/ (дата звернення: 08.03.2025)
2. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.
3. Бевз С.В. Розробка мобільної системи для контролю дихання та реабілітації / С.В. Бевз, С.М. Бурбело, Д.В. Рибак, І.О. Лещенко // Стан, досягнення і перспективи інформаційних систем і технологій / Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. Одеса, 17-18 квітня 2025 р. – Одеса, Видавництво ОНТУ, 2025 р. – С.91-93.
4. Штучний інтелект та аналіз даних [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ranktracker.com/uk/blog/ai-and-data-analytics-enhancing-content-marketing-strategies/> (дата звернення: 09.03.2025)
5. Мобільні додатки для психічного здоров'я – перспективи розвитку року <https://mediacom.com.ua/mobilni-dodatki-dlya-psixichnogo-zdorovya-yak-voni-zminyatsya-do-2024-roku> <https://mediacom.com.ua/mobilni-dodatki-dlya-psixichnogo-zdorovya-yak-voni-zminyatsya-do-2024-roku/> (дата звернення: 11.03.2025)
6. Calm [Електронний ресурс] – Режим доступу до ресурсу: <https://mind.ua/news/20217074-dodatok-dlya-meditaciyi-calm-nabrav-populyarnosti-v-pandemiyu-i-podvoyiv-ocinku-do-22-mlrd> (дата звернення: 08.03.2025)
7. Headspace [Електронний ресурс] – Режим доступу до ресурсу: <https://www.headspace.com/> (дата звернення: 12.03.2025)

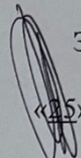
8. Breathe2Relax [Електронний ресурс] – Режим доступу до ресурсу: <https://staging-mobile.health.mil/Apps/PWA-Apps/Breathe2Relax> (дата звернення: 12.03.2025)
9. Короткі дихальні вправи [Електронний ресурс] – Режим доступу до ресурсу: <https://nauka.ua/news/korotki-dihalni-vpravi-pokrashchili-nastrij-krashche-za-meditaciyu> (дата звернення: 15.03.2025)
10. Етапи аналізу біосигналів [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/7459415/page:47/> (дата звернення: 15.03.2025)
11. Адаптивна верстка [Електронний ресурс] – Режим доступу до ресурсу: <https://sendpulse.ua/blog/responsive-and-adaptive-design> (дата звернення: 15.03.2025)
12. Фідбек: для чого потрібен і як давати зворотний зв'язок [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/fidbek-tse/> (дата звернення: 17.03.2025)
13. Інтеграція API - що це таке? [Електронний ресурс] – Режим доступу до ресурсу: <https://bitfaktura.com.ua/blog/Intehratsiya-API-shcho-tse-take> (дата звернення: 17.03.2025)
14. Swift, що це? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.run-it.com.ua/shcho-take-mova-prohramuvannia-swift/> (дата звернення: 19.03.2025)
15. UML [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/40575/> (дата звернення: 20.03.2025)
16. Python [Електронний ресурс] – Режим доступу до ресурсу: <https://www.python.org/> (дата звернення: 11.03.2025)
17. Java [Електронний ресурс] – Режим доступу до ресурсу: <https://www.java.com/ua/> (дата звернення: 11.03.2025)
18. Swift [Електронний ресурс] – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Swift_\(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F\)](https://uk.wikipedia.org/wiki/Swift_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) (дата звернення: 11.03.2025)

19. Схема компліментарного поєднання [Електронний ресурс] – Режим доступу до ресурсу: <https://social.org.ua/2891-kruta-shpargalka-po-poiednannyu-koloriv.html> (дата звернення: 15.03.2025)
20. Посібник із тестування додатків iOS: вручну та автоматизація [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/getting-started-with-ios-testing.html> (дата звернення: 16.03.2025)

ДОДАТКИ

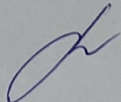
Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

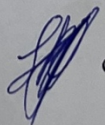
ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
 «25» березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу
«Розробка програмних модулів мобільної системи для контролю дихання,
медитації та релаксу»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:

 к.т.н., доц. Ракитянська Г. Б.
«24» березня 2025 р.

Виконав:

 студент гр. 4ПІ-216 Лещенко І. О.
«24» березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмних модулів мобільної системи для контролю дихання, медитації та релаксу»

Галузь застосування: Сфера охорони здоров'я, мобільні додатки для дихальних практик, медитації та релаксації, психо та фізіологічне відновлення, цифрове благополуччя.

Ч2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Мета дослідження – покращення можливостей реабілітації і релаксу за рахунок розробки та використання спеціалізованою мобільної системи для контролю дихання і медитації, що дозволяє відновлювати психологічний стан користувача.

Призначення роботи – розробка мобільного застосунку для iOS, який допоможе користувачам керувати своїм диханням, проводити сеанси медитації та релаксації.

4. Вихідні дані для проведення НДР

1. Огляд сучасних мобільних додатків для дихальних технік і медитацій.
2. Сучасні методи контролю дихання, психоемоційного стану та релаксації за допомогою мобільних технологій.
3. Документація та посібники по розробці iOS додатків (Swift, Xcode).
4. Інтерфейси та алгоритми для інтеграції дихальних вправ і медитацій у мобільні додатки.

5. Технічні вимоги

Комп'ютер з 64-розрядним (x64) процесором, тактовою частотою 2 ГГц, об'ємом оперативної пам'яті 8 ГБ та розміром жорсткого диска 256 ГБ. Операційна система macOS для розробки додатків під iOS. З клієнтського боку: мобільний пристрій на базі iOS (версія не нижче iOS 13).

6. Конструктивні вимоги

Мобільний застосунок має бути зручним для користувачів, з інтуїтивно зрозумілим інтерфейсом для виконання дихальних вправ і медитацій. Програмне забезпечення повинно бути доступне для всіх користувачів з різним рівнем технічної підготовки. Додаток має бути оптимізованим для різних моделей iPhone, щоб забезпечити стабільну роботу на різних пристроях.

7. Перелік технічної документації, що пред'являється по закінченню робіт

- Пояснювальна записка до бакалаврської кваліфікаційної роботи;
- Технічне завдання;
- Лістинги програмного коду (з деталями реалізації методів і алгоритмів);
- Інтерфейс користувача та опис його функціональності;
- Звіт про проведені тести та їх результати.

8. Вимоги до рівня уніфікації та стандартизації

Розробка програмного забезпечення має відповідати міжнародним стандартам для мобільних додатків. Також повинні бути дотримані вимоги до безпеки даних користувачів та персональної інформації (GDPR, Apple Guidelines).

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програм для контролю дихання, медитації та релаксу	25.03.25 – 10.04.25
2	Розробка методів, моделі та алгоритмів роботи програми	11.04.25 – 18.04.25
3	Розробка програмного забезпечення	19.04.25 – 29.04.25
4	Тестування програми	30.04.25 – 15.05.25
5	Оформлення матеріалів до захисту БКР	16.05.25 – 30.05.25

10. Порядок контролю та прийняття

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником відповідно до затвердженого плану. Прийняття роботи здійснюється комісією згідно з графіком захисту.

**Додаток Б. Протокол перевірки бакалаврської кваліфікаційної
роботи на наявність текстових запозичень**

**ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ**

Назва роботи: «Розробка програмних модулів мобільної системи для контролю
дихання, медитації та релаксу»

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ракитянська Г. Б.

Оригінальність	94 %
Схожість	3 %

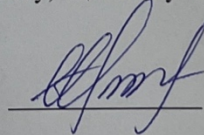
Аналіз звіту подібності

■ Запозичення, виявлені 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 автора. Роботу направити на доопрацювання.

☐ 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.

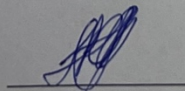
Особа, відповідальна за перевірку



Черноволик Г. О.

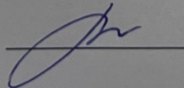
Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism.

Автор роботи



Лещенко І. О.

Керівник роботи



Ракитянська Г. Б.

Додаток В. Лістинг програми

```
import SwiftUI

import AVFoundation

@main

struct BreathingMeditationApp: App {

    var body: some Scene {

        WindowGroup {

            ContentView()

        }

    }

}

struct ContentView: View {

    var body: some View {

        NavigationView {

            VStack(spacing: 40) {

                Text("Relax 347")

                .font(.largeTitle)

                .bold()

                NavigationLink(destination: BreathingView()) {

                    MenuButton(title: "Контроль дихання", color: .blue)

                }

                NavigationLink(destination: MeditationView()) {

                    MenuButton(title: "Медитація", color: .purple)

                }

            }

        }

    }

}
```



```

        }
    }
    .padding()
}
}
}

```

```

struct MenuButton: View {
    var title: String
    var color: Color

    var body: some View {
        Text(title)
        .foregroundColor(.white)
        .padding()
        .frame(maxWidth: .infinity)
        .background(color)
        .cornerRadius(15)
        .shadow(radius: 10)
    }
}

```

// MARK: - Контроль дыхания

```

struct BreathingView: View {
    @State private var isInhaling = true
    @State private var scale: CGFloat = 1.0
    @State private var counter = 0
    let totalCycles = 6

```

```

let inhaleDuration = 4.0

let exhaleDuration = 4.0

var body: some View {
    VStack(spacing: 30) {
        Text("Контроль дыхания")
            .font(.title2)

        Spacer()

        Circle()
            .fill(isInhaling ? Color.blue.opacity(0.6) : Color.green.opacity(0.6))
            .frame(width: 200 * scale, height: 200 * scale)
            .animation(.easeInOut(duration: isInhaling ? inhaleDuration : exhaleDuration), value:
scale)

        Text(isInhaling ? "Вдыхай..." : "Выдыхай...")
            .font(.title)
            .foregroundColor(.gray)

        Spacer()

        Text("Раунд: \(counter)^\(totalCycles)")

        Button(action: startBreathing) {
            Text("Почати")
                .foregroundColor(.white)
                .padding()
        }
    }
}

```

```

        .frame(maxWidth: .infinity)

        .background(Color.blue)

        .cornerRadius(12)

    }

    .padding(.horizontal)

}

.padding()

}

```

```

func startBreathing() {

    counter = 0

    runCycle()

}

```

```

func runCycle() {

    guard counter < totalCycles else { return }

    isInhaling = true

    scale = 1.3

```

```

DispatchQueue.main.asyncAfter(deadline: .now() + inhaleDuration) {

    isInhaling = false

    scale = 0.7

```

```

DispatchQueue.main.asyncAfter(deadline: .now() + exhaleDuration) {

    counter += 1

    runCycle()

}

```

```

    }
}
}

// MARK: - Медитація

struct MeditationView: View {

    @State private var remainingTime = 60

    @State private var timer: Timer?

    @State private var isMeditating = false

    var body: some View {
        VStack(spacing: 30) {
            Text("Медитація")
                .font(.title2)

            Text("\(remainingTime) сек.")
                .font(.largeTitle)
                .monospacedDigit()

            Button(action: {
                if isMeditating {
                    stopMeditation()
                } else {
                    startMeditation()
                }
            }) {
                Text(isMeditating ? "Завершити" : "Почати")
                    .foregroundColor(.white)
            }
        }
    }
}

```

```

        .padding()

        .frame(maxWidth: .infinity)

        .background(isMeditating ? Color.red : Color.green)

        .cornerRadius(12)
    }

    .padding(.horizontal)
}

.padding()
}

func startMeditation() {
    isMeditating = true
    remainingTime = 60

    timer = Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { t in
        if remainingTime > 0 {
            remainingTime -= 1
        } else {
            stopMeditation()
        }
    }
}

func stopMeditation() {
    timer?.invalidate()
    timer = nil
    isMeditating = false
} let inhaleDuration = 4.0

```

```

let exhaleDuration = 4.0

var body: some View {
    VStack(spacing: 30) {
        Text("Контроль дыхания")
            .font(.title2)

        Spacer()

        Circle()
            .fill(isInhaling ? Color.blue.opacity(0.6) : Color.green.opacity(0.6))
            .frame(width: 200 * scale, height: 200 * scale)
            .animation(.easeInOut(duration: isInhaling ? inhaleDuration : exhaleDuration), value:
scale)

        Text(isInhaling ? "Вдыхай..." : "Выдыхай...")
            .font(.title)
            .foregroundColor(.gray)

        Spacer()

        Text("Раунд: \(counter)\^(totalCycles)")

        Button(action: startBreathing) {
            Text("Почати")
                .foregroundColor(.white)
                .padding()
        }
    }
} // ContentView.swift

```

```

// relax2

//

//

import SwiftUI

import UserNotifications

import AVFoundation

/// Представлення окремої картки медитації, яку користувач може обрати для запуску сесії
struct MeditationCard: View {

    let icon: String      // Іконка медитації (SF Symbol)

    let title: String     // Назва медитації

    let duration: Int     // Тривалість у хвилинах

    let description: String // Короткий опис

    let color: Color      // Колір фону картки

    let action: () -> Void // Дія при натисканні на картку

    var body: some View {

        Button(action: action) {

            VStack(alignment: .leading, spacing: 8) {

                HStack {

                    Image(systemName: icon)

                    .font(.title)

                    .foregroundColor(.white)

                    Text(title)

                    .font(.headline)

                    .foregroundColor(.white)

                }

            }

        }

    }
}

```

```

        Text(description)
            .font(.subheadline)
            .foregroundColor(.white.opacity(0.8))
        Text("\ (duration) хв")
            .font(.caption)
            .foregroundColor(.white.opacity(0.6))
    }
    .padding()
    .background(color)
    .cornerRadius(12)
}
}
}

/// Основна логіка запуску сесії медитації
private func startSession(title: String, duration: Int) {
    // Встановлення назви поточної сесії
    currentSession = title

    // Встановлення залишку часу у секундах
    timeRemaining = duration * 60

    // Показати таймер
    showingTimer = true

    // Завантаження та запуск звуку медитації
    if let soundURL = Bundle.main.url(forResource: "meditation", withExtension: "mp3") {
        do {

```



```

        audioPlayer = try AVAudioPlayer(contentsOf: soundURL)

        audioPlayer?.numberOfLoops = -1 // Бескінечне повторення

        audioPlayer?.play()

    } catch {

        print("Помилка відтворення звуку: \(error)")

    }

}

}

}

swift

/// Представлення таймера під час активної медитації

struct TimerView: View {

    @Binding var timeRemaining: Int // Залишок часу у секундах (зв'язаний з ContentView)

    @Binding var showingTimer: Bool // Прапорець, що показує чи відображати таймер

    let sessionName: String // Назва поточної сесії медитації

    // Створення таймера, який тикає кожну секунду

    let timer = Timer.publish(every: 1, on: .main, in: .common).autoconnect()

    var body: some View {

        VStack(spacing: 30) {

            Text(sessionName)

                .font(.title)

                .foregroundColor(.white)

            // Відображення залишку часу

            Text(timeString(from: timeRemaining))

                .font(.system(size: 70, weight: .bold))

                .foregroundColor(.white)

```

```

// Кнопка завершення сесії

Button(action: {
    showingTimer = false
}) {
    Text("Завершити")
        .foregroundColor(.white)
        .padding()
        .background(Color.red.opacity(0.8))
        .cornerRadius(10)
    }
}

.onReceive(timer) { _ in
    // Щосекундне оновлення таймера
    if timeRemaining > 0 {
        timeRemaining -= 1
    } else {
        showingTimer = false // Завершення сесії
    }
}
}

/// Перетворення кількості секунд у формат хв:сек (наприклад, 05:32)
private fun timeString(from seconds: Int) -> String {
    let minutes = seconds / 60
    let remainingSeconds = seconds % 60
    return String(format: "%02d:%02d", minutes, remainingSeconds)
}

```

```
}
```

```
swift
```

```
/// Розширення View для зміни кольору заголовку навігаційного бару
```

```
extension View {
```

```
    func navigationBarTitleTextColor(_ color: Color) -> some View {
```

```
        self.modifier(NavigationBarTitleModifier(color: color))
```

```
    }
```

```
}
```

```
swift
```

```
/// Модифікатор для кастомізації зовнішнього вигляду заголовка навігаційного бару
```

```
struct NavigationBarTitleModifier: ViewModifier {
```

```
    let color: Color
```

```
    func body(content: Content) -> some View {
```

```
        content
```

```
        .onAppear {
```

```
            let appearance = UINavigationBarAppearance()
```

```
            appearance.configureWithTransparentBackground()
```

```
            appearance.titleTextAttributes = [.foregroundColor: UIColor(color)]
```

```
            appearance.largeTitleTextAttributes = [.foregroundColor: UIColor(color)]
```

```
            UINavigationBar.appearance().standardAppearance = appearance
```

```
            UINavigationBar.appearance().compactAppearance = appearance
```

```
            UINavigationBar.appearance().scrollEdgeAppearance = appearance
```

```
        }
```

```
    }
```

```
}
```

```
swift
```

```

/// Прев'ю для SwiftUI canvas (Xcode)

struct ContentView_Previews: PreviewProvider {

    static var previews: some View {

        ContentView()

    }

import XCTest

@testable import relax2

class relax2Tests: XCTestCase {

    override func setUpWithError() throws {

        // Put setup code here. This method is called before the invocation of each test method in the
        class.

    }

    override func tearDownWithError() throws {

        // Put teardown code here. This method is called after the invocation of each test method in the
        class.

    }

    func testExample() throws {

        // This is an example of a functional test case.

        // Use XCTAssert and related functions to verify your tests produce the correct results.

        // Any test you write for XCTest can be annotated as throws and async.

        // Mark your test throws to produce an unexpected failure when your test encounters an uncaught
        error.

        // Mark your test async to allow awaiting for asynchronous code to complete. Check the results
        with assertions afterwards.

    }

```

```
func testPerformanceExample() throws {
    // This is an example of a performance test case.
    self.measure {
        // Put the code you want to measure the time of here.
    }
}
```

Ivan, [28.04.2025 12:17]

```
//
// relax2UITests.swift
// relax2UITests
//
// Created by mak on 16.04.2025.
//
```

```
import XCTest
```

```
class relax2UITests: XCTestCase {
```

```
    override func setUpWithError() throws {
```

```
        // Put setup code here. This method is called before the invocation of each test method in the
        class.
    }
```

```
    // In UI tests it is usually best to stop immediately when a failure occurs.
```

```
    continueAfterFailure = false
```

```
    // In UI tests it's important to set the initial state - such as interface orientation - required for
    your tests before they run. The setUp method is a good place to do this.
```

```
}
```

```

override func tearDownWithError() throws {

    // Put teardown code here. This method is called after the invocation of each test method in the
    class.

}

func testExample() throws {

    // UI tests must launch the application that they test.

    let app = XCUIApplication()
    app.launch()

    // Use recording to get started writing UI tests.

    // Use XCTAssert and related functions to verify your tests produce the correct results.

}

func testLaunchPerformance() throws {
    if #available(macOS 10.15, iOS 13.0, tvOS 13.0, watchOS 7.0, *) {

        // This measures how long it takes to launch your application.

        measure(metrics: [XCTApplicationLaunchMetric()]) {
            XCUIApplication().launch()

        }

    }
}

//

// relax2UITestsLaunchTests.swift

// relax2UITests

//

// Created by mak on 16.04.2025.

```

```
//

import XCTest

class relax2UITestsLaunchTests: XCTestCase {

    override class var runsForEachTargetApplicationUIConfiguration: Bool {
        true
    }

    override func setUpWithError() throws {
        continueAfterFailure = false
    }

    func testLaunch() throws {
        let app = XCUIApplication()
        app.launch()

        // Insert steps here to perform after app launch but before taking a screenshot,
        // such as logging into a test account or navigating somewhere in the app

        let attachment = XCTAttachment(screenshot: app.screenshot())
        attachment.name = "Launch Screen"
        attachment.lifetime = .keepAlways
        add(attachment)
    }
}

import SwiftUI

struct ContentView: View {
```

```

var body: some View {
    TabView {
        MeditationView()

        .tabItem {
            Label("Медитація", systemImage: "leaf")
        }

        BreathingView()

        .tabItem {
            Label("Дихання", systemImage: "wind")
        }

        AccountView()

        .tabItem {
            Label("Акаунт", systemImage: "person.circle")
        }

        SettingsView()

        .tabItem {
            Label("Налаштування", systemImage: "gear")
        }
    }
}

#Preview {
    ContentView()
} import SwiftUI

```



```

struct MeditationView: View {
    var body: some View {
        NavigationView {
            List {
                ForEach(Meditation.sampleMeditations) { meditation in
                    NavigationLink(destination: MeditationDetailView(meditation: meditation)) {
                        Text(meditation.title)
                    }
                }
            }
            .navigationTitle("Медитації")
        }
    }
}

```

```

struct Meditation: Identifiable {
    let id = UUID()
    let title: String
    let description: String

    static let sampleMeditations = [
        Meditation(title: "Ранкова медитація", description: "Почніть день спокійно та усвідомлено."),
        Meditation(title: "Антистрес", description: "Заспокойте розум у важкий день."),
        Meditation(title: "Перед сном", description: "Розслабтеся і підготуйтеся до сну.")
    ]
}

```

```

struct MeditationDetailView: View {
    let meditation: Meditation

    var body: some View {
        VStack(spacing: 20) {
            Text(meditation.title)
                .font(.largeTitle)
                .padding()

            Text(meditation.description)
                .padding()

            Spacer()

            Button("Почати медитацію") {
                // Тут логіка початку медитації
            }
                .buttonStyle(.borderedProminent)
                .padding()
        }
    }
} import SwiftUI

```

```

struct BreathingView: View {
    @State private var isBreathing = false

    var body: some View {
        VStack(spacing: 40) {

```

```

Text("Дихальні вправи")
    .font(.largeTitle)

if isBreathing {
    Text("Вдих... Видих...")
        .font(.title)
        .transition(.opacity)
}

Button(isBreathing ? "Зупинити" : "Почати") {
    withAnimation {
        isBreathing.toggle()
    }
}

.buttonStyle(.borderedProminent)
.padding()
}
}
} import SwiftUI

struct AccountView: View {
    @State private var username = "Іван"
    @State private var email = "ivan@example.com"

    var body: some View {
        NavigationView {
            Form {
                Section(header: Text("Інформація про акаунт")) {

```

```
HStack {
    Text("Ім'я")
    Spacer()
    Text(username)
    .foregroundColor(.gray)
}
```

```
HStack {
    Text("Email")
    Spacer()
    Text(email)
    .foregroundColor(.gray)
}
}
```

```
Section {
    Button("Вийти з акаунту") {
        // Логіка виходу
    }
    .foregroundColor(.red)
}
}
.navigationTitle("Акаунт")
}
}
} import SwiftUI
```

```
struct SettingsView: View {
```

```
@AppStorage("notificationsEnabled") private var notificationsEnabled = true
```

```
var body: some View {
    NavigationView {
        Form {
            Section(header: Text("Нагадування")) {
                Toggle("Нагадування про дихання", isOn: $notificationsEnabled)
            }

            Section {
                NavigationLink("Про додаток", destination: AboutView())
            }
        }
        .navigationTitle("Налаштування")
    }
}
```

```
struct AboutView: View {
    var body: some View {
        VStack(spacing: 20) {
            Text("Time Manager")
                .font(.largeTitle)

            Text("Версія 1.0")
                .foregroundColor(.gray)

            Text("Додаток для контролю дихання, медитації та релаксації.")
        }
    }
}
```

```

        .multilineTextAlignment(.center)
        .padding()

        Spacer()
    }
    .padding()
}
} import SwiftUI

struct ExitButtonView: View {
    var body: some View {
        Button("Завершити програму") {
            exitApp()
        }
        .buttonStyle(.borderedProminent)
        .padding()
    }

    private func exitApp() {
        DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
            exit(0)
        }
    }
}

} Section {
    Button("Вийти з програми") {
        exitApp()
    }

    .foregroundColor(.red)

```

```
}
```

```
private func exitApp() {
    DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
        exit(0)
    }
} import SwiftUI
```

```
struct ExitButtonView: View {
    @State private var showAlert = false

    var body: some View {
        Button("Завершити програму") {
            showAlert = true
        }
        .alert("Ви впевнені?", isPresented: $showAlert) {
            Button("Так", role: .destructive) {
                exitApp()
            }
            Button("Ні", role: .cancel) {}
        }
        .buttonStyle(.borderedProminent)
        .padding()
    }

    private func exitApp() {
        DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
            exit(0)
        }
    }
}
```

Додаток Г. Ілюстративна частина

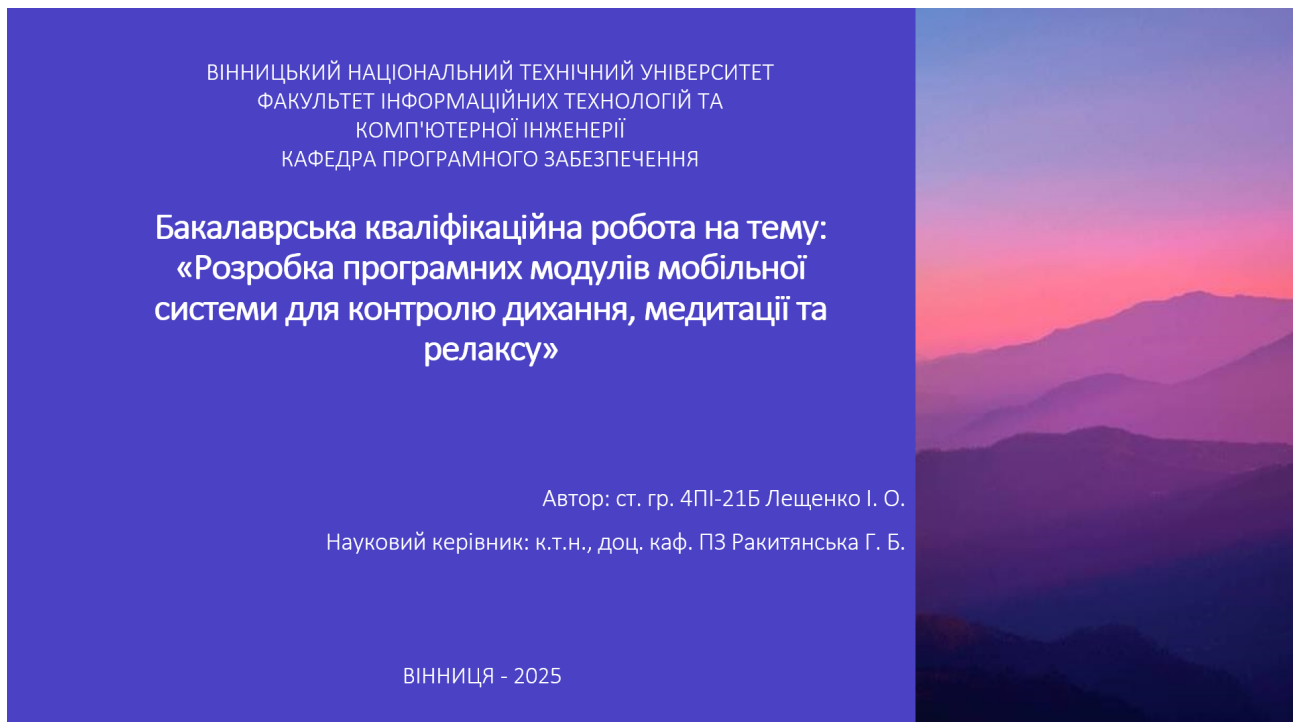


Рисунок Г.1 – Титульний слайд

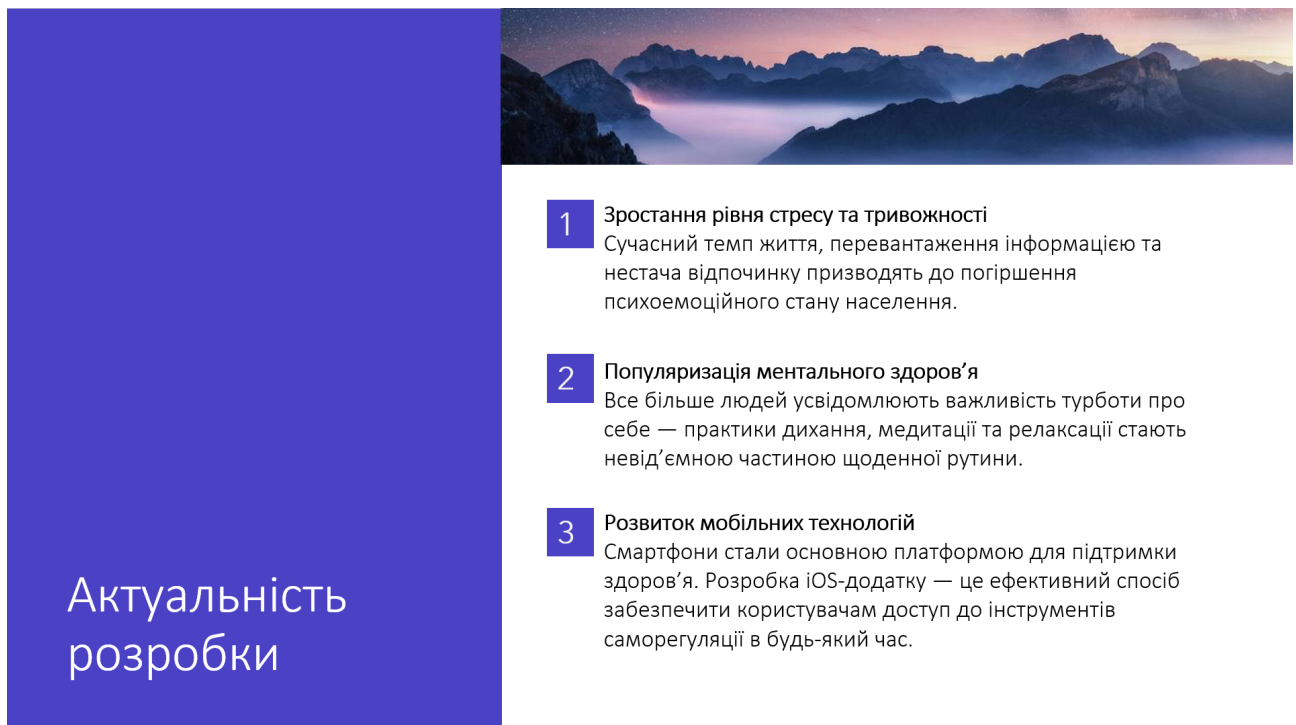


Рисунок Г.2 – Актуальність розробки

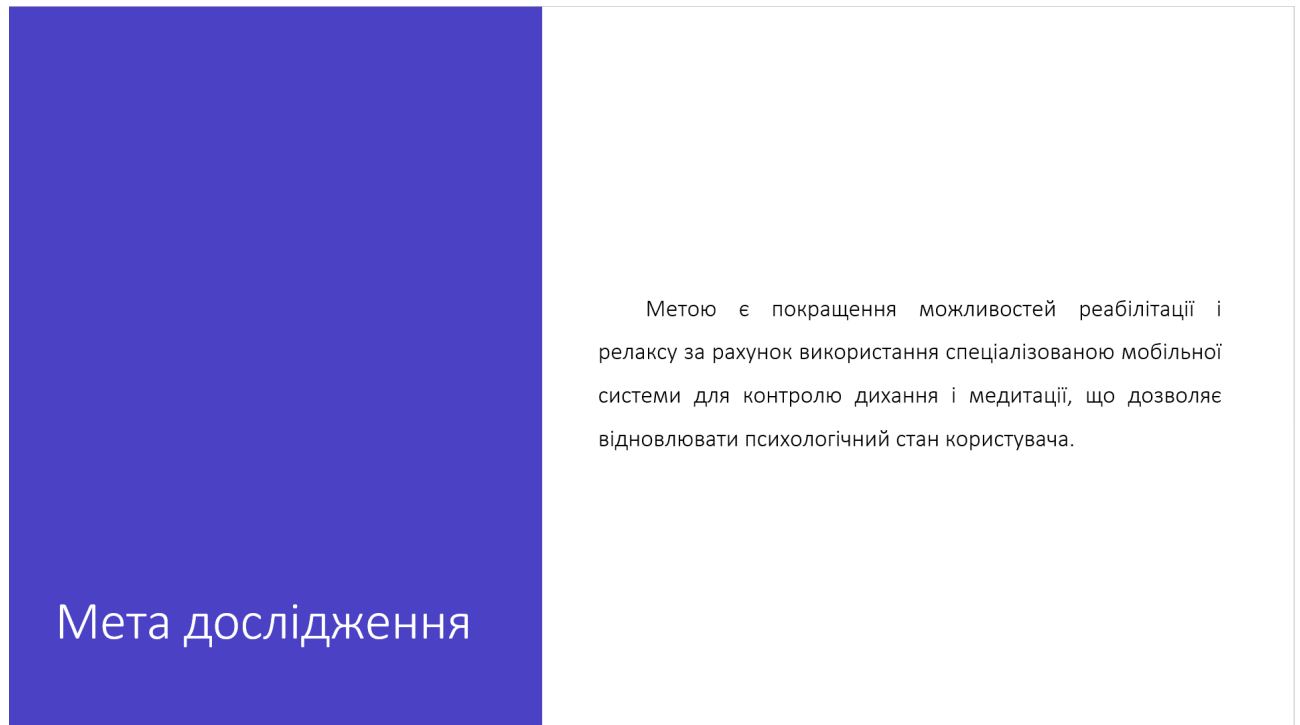


Рисунок Г.3 – Мета дослідження

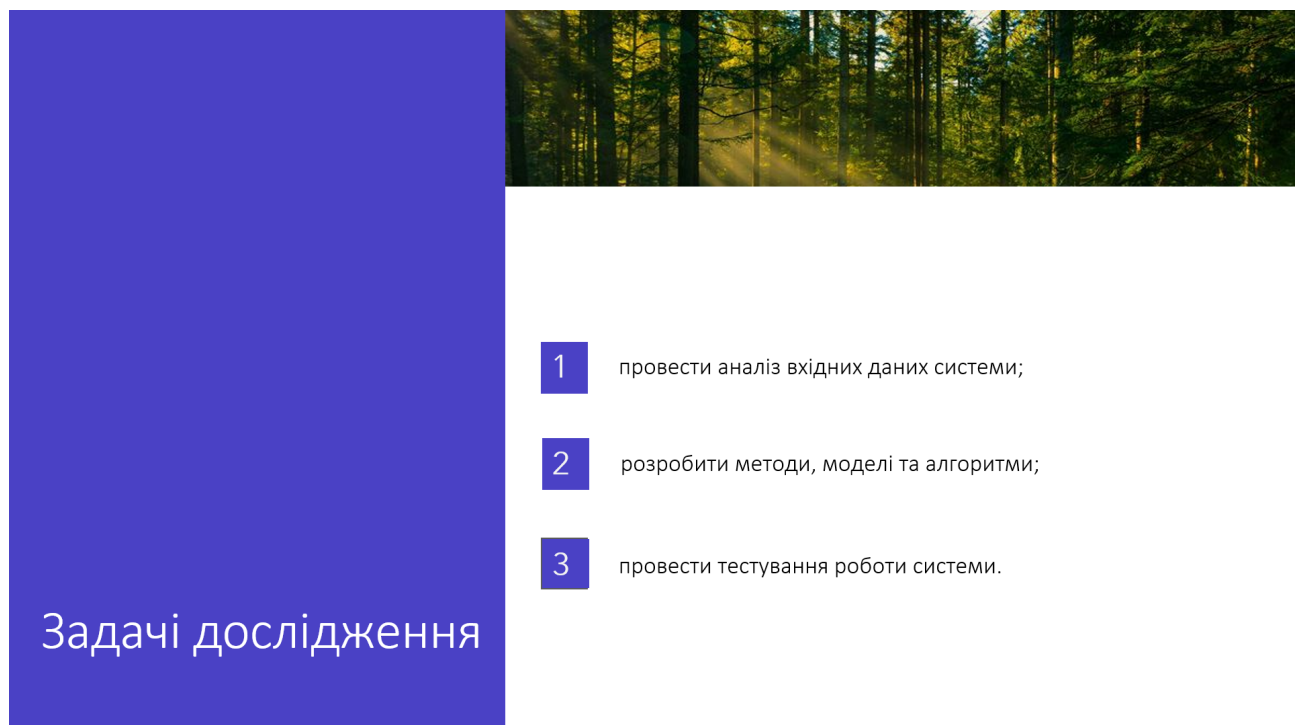


Рисунок Г.4 – Задачі дослідження

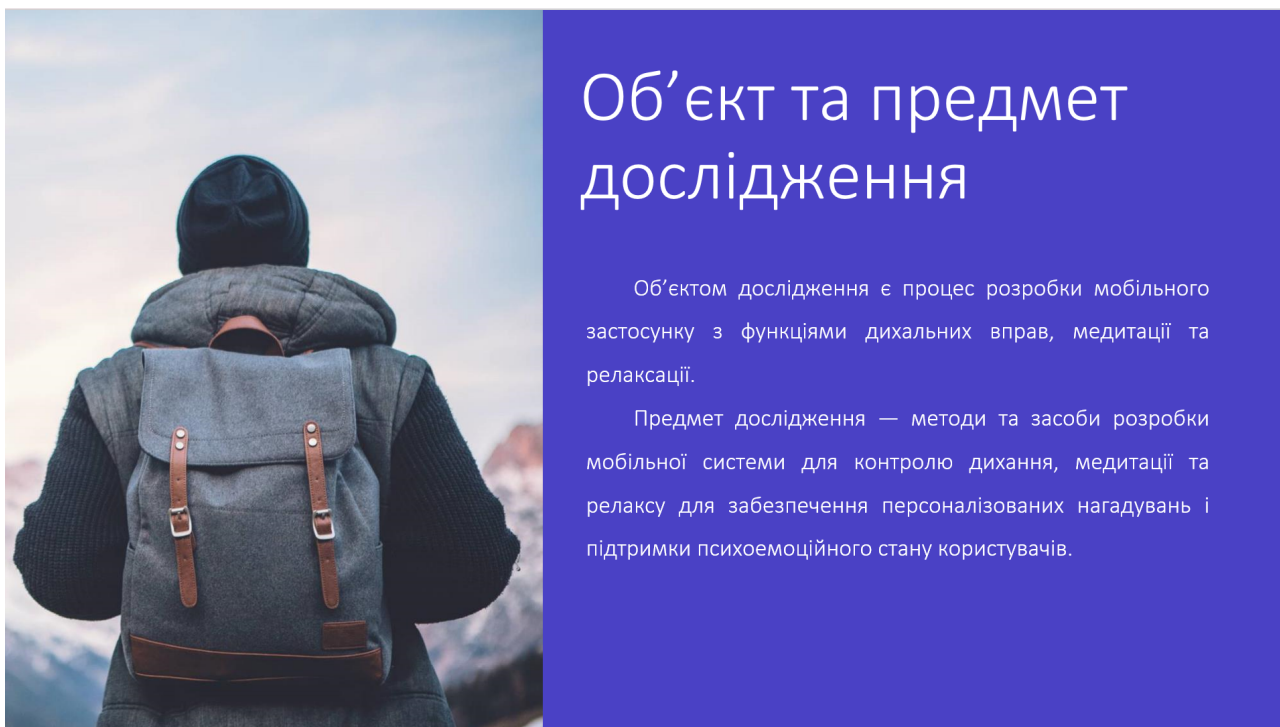
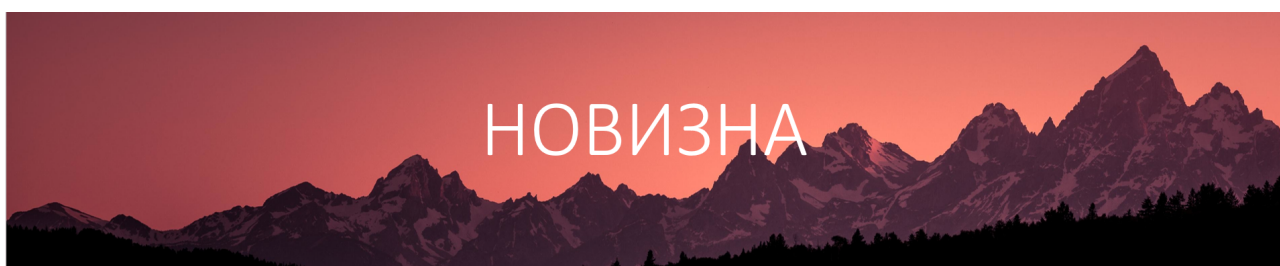


Рисунок Г5 – Об'єкт та предмет дослідження



1 Подальшого розвитку отримав метод реалізації в мобільному застосунку модуля контролю дихання, який, на відміну від існуючих, реалізує персоналізовану взаємодію з користувачем на основі візуальних та тактильних підказок, що дозволяє користувачу підвищити зручність виконання дихальних вправ, отримувати зворотній зв'язок з системою, що сприятиме підвищенню ефективності релаксації, зниженню рівня стресу та покращенню загального психоемоційного стану.

2 Подальшого розвитку отримав метод реалізації в мобільному застосунку модуля медитацій, який, на відміну від існуючих, забезпечує інтеграцію адаптивних сценаріїв медитацій з урахуванням рівня досвіду користувача, поточного стану стомленості чи напруги та цілей релаксації, що дозволяє покращити індивідуалізований підхід до психоемоційного відновлення користувача та підвищити ефективність медитативної практики.

Рисунок Г.6 – Новизна

ПРАКТИЧНА ЦІННІСТЬ

1 Практична цінність полягає у можливості використання розробленої мобільної системи для покращення стану користувача та створення зручного та доступного інструменту для покращення емоційного стану та зниження рівня стресу. Розроблене програмне забезпечення може бути корисним для осіб з високим рівнем стресу, тих, хто практикує медитацію, або користувачів, які бажають зберегти здоровий баланс між роботою та відпочинком. Окрім цього, таке програмне забезпечення може бути використане для інтеграції в інші додатки, наприклад, у сфері здоров'я та продуктивності.

Рисунок Г.7 – Практична цінність

ІСНУЮЧІ АНАЛОГИ



Рисунок Г.8 – Існуючі аналоги

АНАЛІЗ АНАЛОГІВ

Критерії оцінки	Calm	Headspace	Breethe2Relax	Власна розробка
Дихальні вправи	1	1	1	1
Інтерактивні візуалізації	0	0	1	1
Таймери медитації	1	1	0	1
Нагадування про дихання	0	1	1	1
Режим фокусування	0	0	0	1
Сумарний коефіцієнт	2	3	3	5

Порівняльний аналіз показав недоліки існуючих рішень у персоналізації та інтеграції даних. Власна розробка пропонує гнучкий і адаптивний підхід для ефективного контролю дихання, медитації та релаксу.

Рисунок Г.9 – Аналіз аналогів

Алгоритми роботи системи

Блок-схема, яка ілюструє загальний процес функціонування мобільної системи для контролю дихання, медитації та релаксації.

Вона відображає основні етапи роботи застосунку починаючи з ініціалізації користувацького сеансу, вибору режиму дихання, медитація або релаксація, налаштування параметрів сесії, і завершуючи зворотним зв'язком або рекомендаціями після завершення вправи.

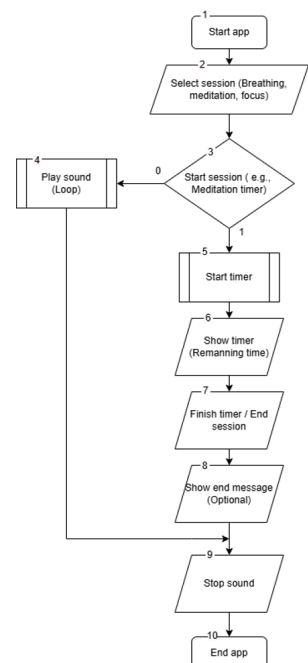


Рисунок Г.10 –Алгоритм роботи системи

Алгоритми роботи системи

Алгоритм системи для моніторингу дихання, медитації та релаксації перевіряє коректність даних, налаштовує індивідуальні параметри вправ і візуалізує результати, забезпечуючи користувачу інтерактивний та ефективний досвід для покращення здоров'я та самопочуття.

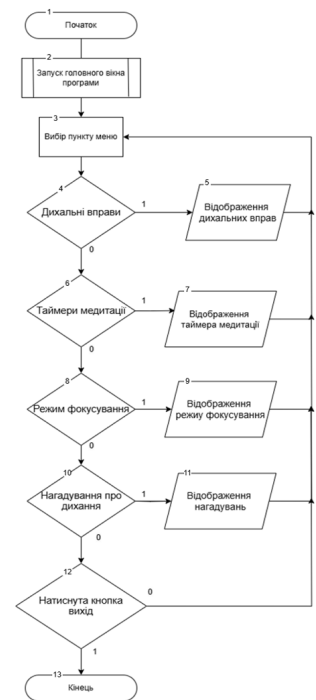


Рисунок Г.11 –Алгоритм роботи системи (2)

СХЕМИ ІНТЕРФЕЙСУ

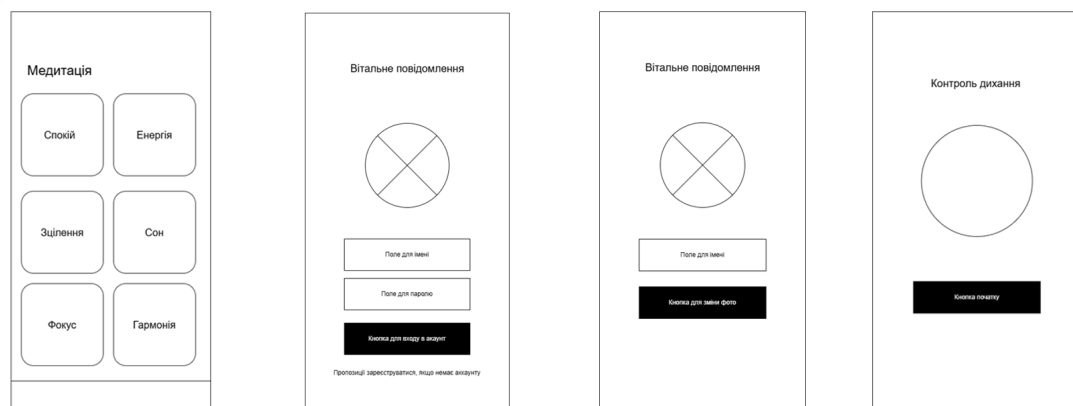


Рисунок Г.12 – Схеми інтерфейсу

Реєстрація

Реєстрація реалізована через просту форму з полем для імені користувача.

Після введення імені та натискання кнопки «Увійти» здійснюється перехід до головного меню.

У разі помилки введення виводиться банер з повідомленням.

Інтерфейс мінімалістичний, зручний для швидкого старту без зайвих дій.

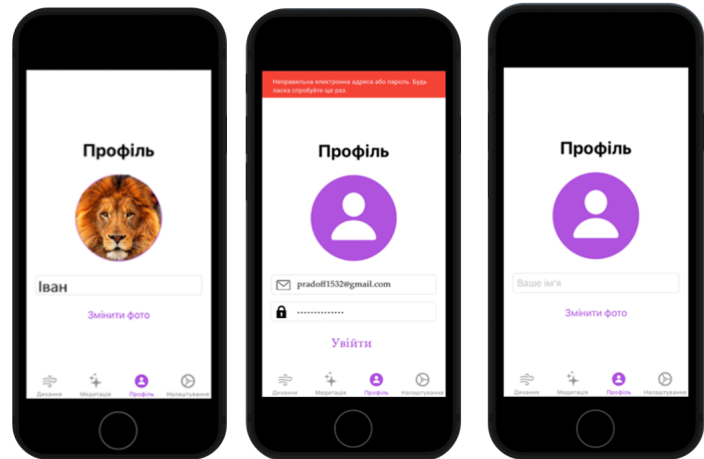


Рисунок Г.13 – Реєстрація

НАЛАШТУВАННЯ

У вкладці «Налаштування» користувач може вмикати сповіщення, перемикаати темну тему та керувати звуками.

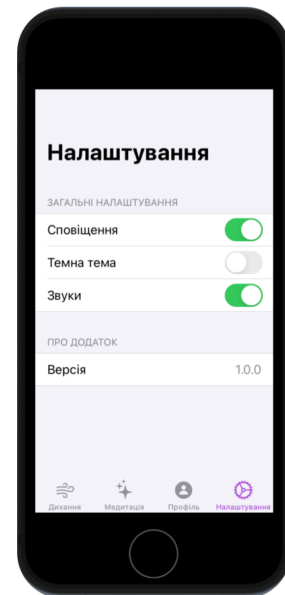


Рисунок Г.14 – Налаштування



Інтерфейс оформлено просто й естетично: м'які кольори, великі шрифти, округлі кнопки — усе для зручності та розслаблення користувача.

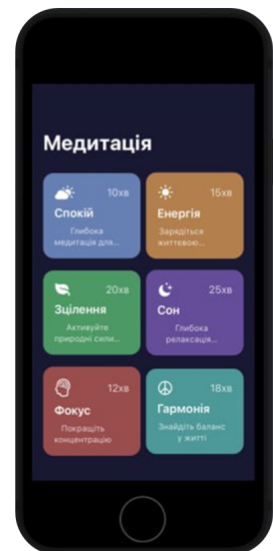
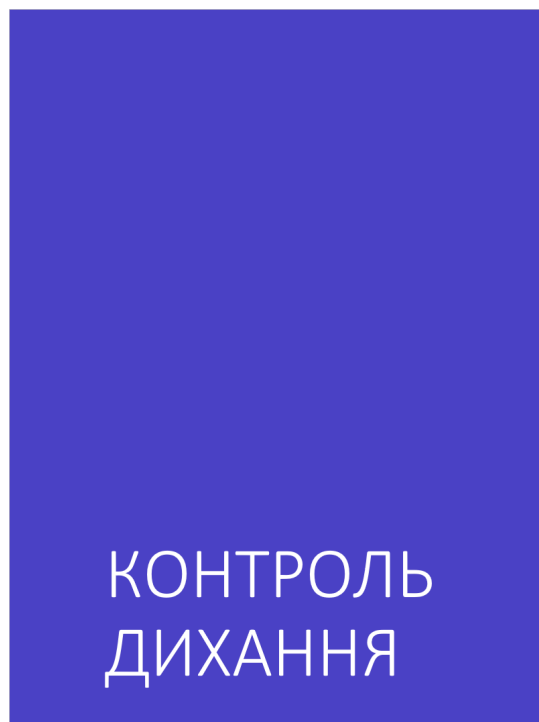


Рисунок Г.15 – Медитація



Вкладка «Дихання» працює стабільно, має зручний інтерфейс і ефективно допомагає контролювати дихання.

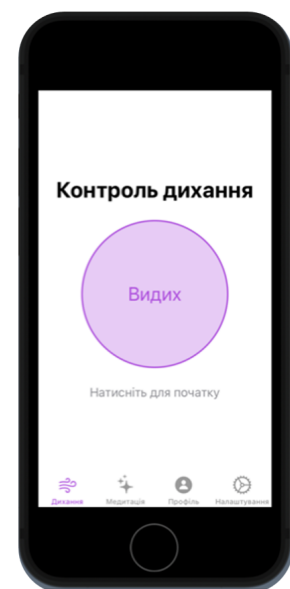


Рисунок Г.16 – Контроль дихання



Вкладка «Фокус» працює стабільно, режим активується без збоїв, інтерфейс зручний.

Після закінчення медитації, коли таймер доходить до 00:00, додаток пропонує натиснути кнопку “Вихід”, та закінчити сеанс, та вийти .

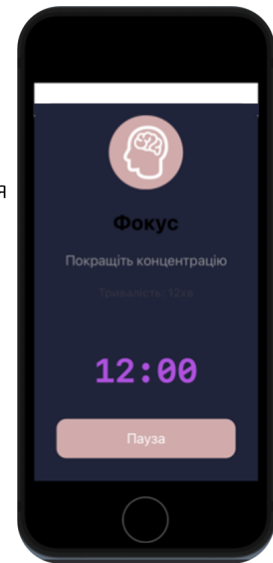


Рисунок Г.17 – Фокус



У результаті виконання роботи було створено повноцінний iOS-застосунок для контролю дихання, медитації та релаксу.

Система реалізує всі основні функції, успішно пройшла тестування та може використовуватись для підтримки ментального здоров'я користувачів у повсякденному житті.

Для реалізації системи було розроблено метод, архітектурну модель та алгоритми роботи застосунку. Застосовано мову програмування Swift у середовищі Xcode із використанням сучасних фреймворків iOS для побудови зручного інтерфейсу та взаємодії з системними можливостями пристрою.

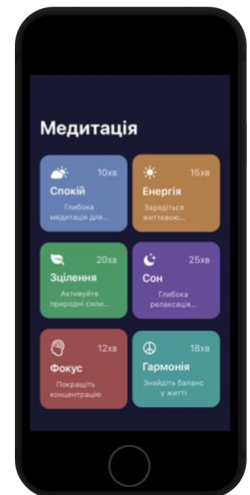


Рисунок Г.18 – Висновки

