Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

# Бакалаврська кваліфікаційна робота

на тему: «Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту»

Виконав: студент 4 курсу
групи 5ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Ксенченко Я.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Мельник О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Кожем'яко А.В.
(прізвище та ініціали)

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

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
Романюк О.Н.
«21» березня 2025 р.

# З А В Д А Н Н Я
# НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ксенченку Ярославу Володимировичу

1. Тема роботи – «Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту»

Керівник роботи: Мельник Олександр Васильович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студенткою роботи 2 червня 2025 р.

3. Вихідні дані до роботи: тип програми – мобільний застосунок;; засоби організації даних програми – бібліотеки Room, Jetpack Compose, Dagger Hilt; вхідні дані: інформація про емоційний стан; вихідні дані: рекомендації, аналіз емоцій; середовище розробки – Android Studio; мова програмування – Kotlin.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку застосунків для управління емоційним станом; розробка методів та алгоритмів програмного застосунку; розробка застосунку; тестування застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична

цінність одержаних результатів; порівняльний аналіз аналогів, використ
технології; розробка методу аналізу голосу для визначення емоційного ст
розробка методу контекстуального прогнозу емоційного стану; розробка мо
застосунку; тестування застосунку; апробація результатів роботи і публік
висновки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада Консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Мельник О.В., к.т.н., доцент кафедри ПЗ | 24.03.25 | 01.06.25 |

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 застосунків для управління емоційним станом | 25.03.2025 - 06.04.2025 | вик |
| 2 | Розробка методів та алгоритмів програмного застосунку | 07.04.2025 - 20.04.2025 | вик |
| 3 | Варіантний аналіз та вибір мови програмування розробки | 21.04.2025 - 27.04.2025 | вик |
| 4 | Розробка застосунку | 28.04.2025 - 18.05.2025 | вик |
| 5 | Тестування застосунку | 19.05.2025 - 25.05.2025 | вик |
| 6 | Оформлення матеріалів до захисту БКР | 26.05.2025- 30.05.2025 | вик |

Студент _____ Ксенченко Я.В.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Мельник О.В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

УДК 004:005.9

Ксенченко Я.В. Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця : ВНТУ, 2025. 100 с.

У бакалаврській кваліфікаційній роботі було розроблено застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту. Розроблений застосунок дозволяє дозволяє записувати власні емоції, їхню інтенсивність та контекст. Також застосунок дозволяє ділитися емоціями у голосовому режимі та отримувати аналіз не лише сказаного, а і голосу.

У ході роботи було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів, в результаті якого було доведено актуальність власної розробки. Проведено аналіз методів розв’язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження. Розроблено структуру застосунку для управління емоційним станом, розроблено модель системи, алгоритми, метод аналізу голосу для визначення емоційного стану та метод контекстуального прогнозу емоційного стану.

Програмне забезпечення реалізовано з використанням мови програмування Kotlin та середовища розробки Android Studio. У процесі розробки використовувались сучасні бібліотеки та інструменти, такі як Room, Jetpack Compose, Dagger Hilt. Було проведено тестування розробленого застосунку, продемонстровано приклади його використання. Розроблений застосунок виконує поставлені на початку розробки завдання.

Ключові слова: штучний інтелект, емоційний стан, аналіз, мобільний застосунок, Kotlin.

# ABSTRACT

UDC 004:005.9

Ksenchenko Ya.V. Application for determining and managing a person's emotional state based on artificial intelligence algorithms: bachelor's qualification work in the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2025. 100 p.

In the bachelor's qualification work, an application was developed for determining and managing a person's emotional state based on artificial intelligence algorithms. The developed application allows you to record your own emotions, their intensity and context. The application also allows you to share emotions in voice mode and receive an analysis of not only what is said, but also your voice.

During the work, an analysis of the state of the issue of developing an application for determining and managing a person's emotional state was conducted, a comparative analysis of analogues was conducted, as a result of which the relevance of your own development was proven. The analysis of methods for solving the problem of developing an application for determining and managing a person's emotional state was carried out, the research tasks were formulated. The structure of the application for managing the emotional state was developed, a system model, algorithms, a voice analysis method for determining the emotional state and a method for contextual prediction of the emotional state were developed.

The software was implemented using the Kotlin programming language and the Android Studio development environment. Modern libraries and tools such as Room, Jetpack Compose, Dagger Hilt were used in the development process. The developed application was tested, examples of its use were demonstrated. The developed application fulfills the tasks set at the beginning of development.

Keywords: artificial intelligence, emotional state, analysis, mobile application, Kotlin.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження.** Розробка застосунку для визначення та управління емоційним станом людини є актуальною відповіддю на зростаючі труднощі сучасного суспільства, пов’язані зі стресом, вигоранням та загальною нестабільністю психоемоційного фону [1]. Завдяки поєднанню методів комп’ютерного аналізу зображень, обробки сигналів біометричних сенсорів та алгоритмів штучного інтелекту, сучасні мобільні та десктопні рішення можуть своєчасно виявляти зміни у виразі обличчя, тоні голосу, пульсовій активності та іншим фізіологічним показникам користувача [2]. Це дозволяє не лише діагностувати емоційні стани в реальному часі, а й автоматично адаптувати інтерфейс або надати рекомендації для нормалізації психологічного балансу.

Наступним важливим аспектом є інтеграція виявлення емоцій з механізмами самоконтролю та психологічної підтримки [3]. Сучасні застосунки повинні забезпечувати користувача інструментами когнітивно-поведінкової терапії, дихальними вправами, контекстними підказками та відстеженням прогресу в довгостроковій перспективі. Завдяки аналітиці даних про настрої та реакції, система може генерувати персоналізовані плани роботи з емоціями, спрямовані на підвищення рівня стресостійкості та гармонізацію психічного стану [4].

Головною перевагою такого програмного рішення є можливість гнучкого налаштування під індивідуальні особливості користувача: рівень чутливості алгоритмів, бажаний рівень деталізації звітів та інтеграція з носимими пристроями чи платформами дистанційної психологічної підтримки [5]. Таким чином, застосунок стає не лише інструментом для діагностики емоцій, а й комплексною системою управління власним психологічним здоров’ям, здатною оперативно реагувати на зміни в емоційному стані та сприяти формуванню здорових звичок саморегуляції [6].

**Зв’язок роботи з науковими програмами, планами, темами.** Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

**Метою роботи** є удосконалення процесу визначення та управління емоційним станом людини шляхом розробки спеціалізованого програмного забезпечення з використанням алгоритмів штучного інтелекту.

Задачі дослідження:

– розробити архітектуру застосунку для визначення та управління емоційним станом людини;

– розробити алгоритми роботи застосунку для визначення та управління емоційним станом людини;

– розробити функціонал застосунку для визначення та управління емоційним станом людини;

– розробити інтерфейс користувача застосунку;

– провести тестування розробленого застосунку.

**Об’єкт дослідження** – процеси розробки застосунку для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту.

**Предмет дослідження** – методи і засоби реалізації застосунку для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту.

**Методи дослідження.** У процесі досліджень використовувались такі методи: емпіричні методи розробки мобільних застосунків, методи побудови інтерфейсів користувача, методи об’єктно-орієнтованого програмування для розробки модулів застосунку, методи штучного інтелекту для аналізу голосу, методи тестування мобільних застосунків.

**Новизна отриманих результатів:**

1. Подальшого розвитку отримав метод аналізу голосу для визначення емоційного стану, який, на відміну від відомих, аналізує паравербальні характеристики мовлення за допомогою штучного інтелекту, що дозволяє отримати об’єктивну оцінку емоційного стану користувача.

2. Подальшого розвитку отримав метод контекстуального прогнозу емоційного стану, який, на відміну від відомих, попереджає потенційні емоційні виклики на основі існуючих даних про емоційний стан користувача, що дозволяє

користувачам краще підготуватись до них.

**Практична цінність отриманих результатів.** Практичною цінністю є можливість використання розробленої системи для визначення та управління емоційним станом людини, його прогнозування.

**Особистий внесок здобувача.** Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто.

**Апробація матеріалів бакалаврської кваліфікаційної роботи**. Результати бакалаврської кваліфікаційної роботи доповідалися на конференції «Achievements of Science and Applied Research» (May 19- 21, 2025. Dublin, Ireland).

**Публікації.** Тези доповіді опубліковані у матеріалах конференції «Achievements of Science and Applied Research» (May 19- 21, 2025. Dublin, Ireland[7].

**Аналіз**. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто 4 розділи та було використано 25 літературних джерел.

# 1 АНАЛІЗ СТАНУ РОЗВИТКУ ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ НА ОСНОВІ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ

## 1.1 Аналіз стану питання розробки застосунків для управління емоційним станом людини

Ринок ШІ-застосунків для управління емоційним станом демонструє значні темпи зростання: у 2024 році його обсяг досяг приблизно 2,9 млрд USD, а згідно з прогнозами, у періоді 2025–2034 рр. середньорічний темп зростання становитиме близько 21,7 % завдяки потребі в передових інструментах діагностики та терапії у медичній сфері та зростанню попиту на персоналізовані цифрові сервіси у бізнесі й освіті [8].

Академічні дослідження останнім часом зосереджені на застосуванні глибинних нейронних мереж для аналізу емоційного стану як індивідуальних користувачів, так і груп [9]. У численних оглядах детально описано використання згорткових (CNN) та рекурентних (RNN) мереж, а також трансформерів для підвищення точності класифікації на таких датасетах, як FER2013 і AffectNet [10]. Окрему увагу приділено методам статичного й динамічного розпізнавання виразу обличчя, які поєднують просторову та часову інформацію для більш надійного інтерпретування емоційних реакцій [11].

З технічної точки зору, розробка застосунків емоційного ШІ базується на комплексних конвеєрних підходах: захопленні відео- та аудіосигналів, попередній обробці (нормалізації, детекції ключових точок), витонченому інференсі на графічному процесорі та серверному обладнанні з мінімальною затримкою [12].

Етичні та нормативні виклики залишаються визначальними бар’єрами: моделі можуть упереджено працювати з даними користувачів із різним етнічним чи віковим профілем, існують ризики несанкціонованого збору й обробки чутливих даних, а також можливість маніпуляцій через неправильну інтерпретацію емоційних сигналів [13]. Відповідно, дедалі актуальніше питання забезпечення прозорості алгоритмів, аудиту тренувальних даних та розробки

стратегій відповідності GDPR та іншим регуляторним нормам.

Серед головних перспектив розвитку емоційного ШІ виокремлюють:

– мультимодальність – синтез даних з відео, аудіо та текстових повідомлень для більш точної оцінки емоцій;

– Edge AI – перенесення обчислень на кінцеві пристрої для зниження затримок і підвищення захищеності даних;

– персоналізація [14] – адаптація моделей до індивідуальних особливостей користувача на основі попередніх сеансів взаємодії.

Ці напрями спрямовані на створення більш дійсних, точних і етичних застосунків, здатних у реальному часі підтримувати користувача без шкоди приватності й з урахуванням індивідуальних потреб.

### 1.2 Порівняльний аналіз аналогів

Задля визначення власного функціоналу для застосунку для визначення та управління емоційним станом, спершу необхідно проаналізувати існуючі рішення в цьому напрямку, так як їхні функціональні можливості явно демонструють потреби користувачів.

Для цього було обрано такі застосунки: Replika, MindDoc, Tess.

Replika [15] – це застосунок, що виступає в ролі особистого співрозмовника, здатного підтримувати довготривалі бесіди, які допомагають користувачу відчувати себе почутим та зрозумілим, для чого використовуються сучасні алгоритми штучного інтелекту, що дозволяє підлаштовуватись у відповідь на індивідуальний стиль спілкування та настрої користувача.

Застосунок використовує алгоритми машинного навчання та обробки природної мови (LLM) для аналізу вхідних повідомлень. У процесі спілкування він адаптується до особистості користувача, поступово набуваючи більшої гнучкості та розуміння його емоційних потреб. Такий підхід дозволяє створити більш персоналізований досвід, де кожна взаємодія є унікальною та враховує попередні бесіди.

Інтерфейс Replika наведено на рисунку 1.1.

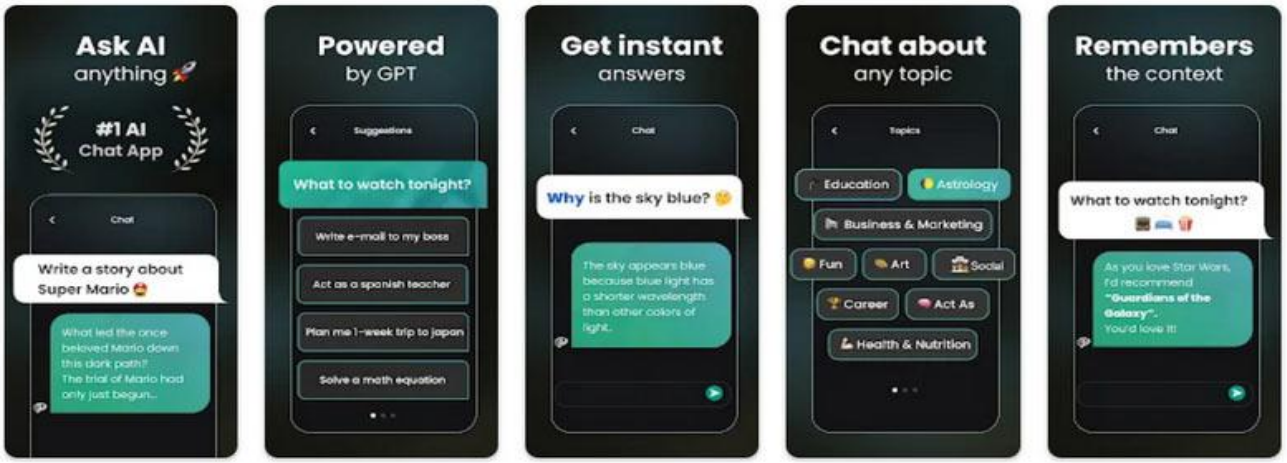


Рисунок 1.1 – Інтерфейс Replika

MindDoc [16] – застосунок, орієнтований на моніторинг психічного здоров'я. Він відслідкововує емоційний стан та виявляє ранні ознаки психологічних труднощів. Інтерфейс програми спроектовано таким чином, щоб користувачі могли легко проходити опитування та отримувати інформативний зворотний зв'язок про свій стан.

Інтерфейс MindDoc наведено на рисунку 1.2.

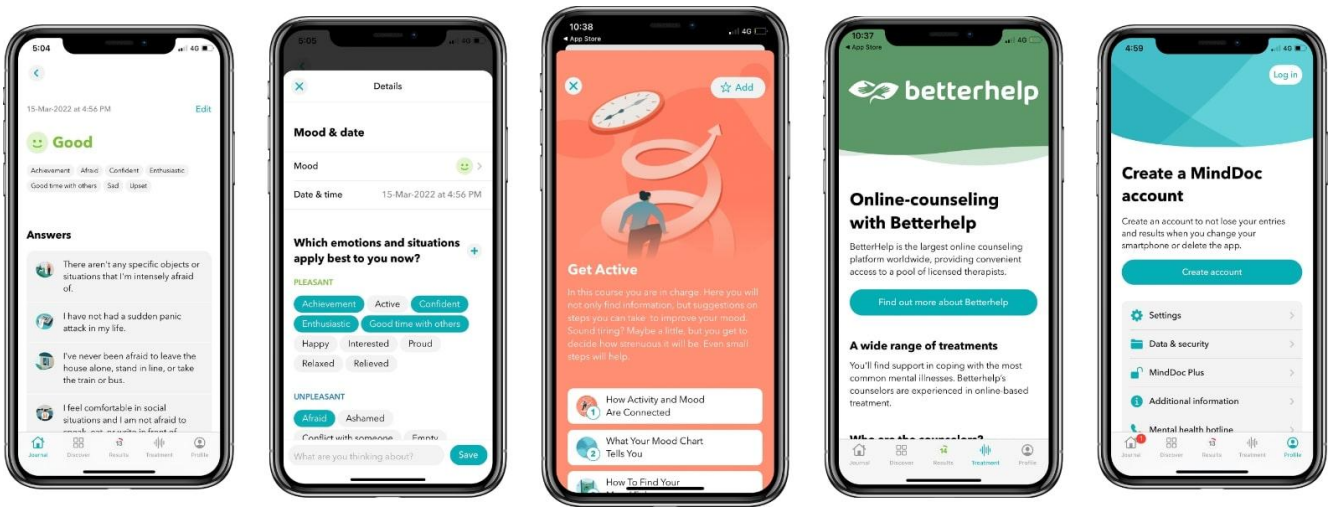


Рисунок 1.2 – Інтерфейс MindDoc

MindDoc проводить регулярне анкетування для збору даних про настрій, що дозволяє алгоритмам штучного інтелекту аналізувати тенденції та закономірності в емоційному стані користувача. За допомогою цього підходу, застосунок може

своєчасно визначати потенційні ризики, пов'язані з погіршенням емоційного стану, і пропонувати індивідуальні рекомендації щодо корекції поведінки та способів зниження стресу.

Tess [17] – це інтерактивний ШІ-чатбот, розроблений для надання психологічної підтримки у режимі реального часу. Його мета – забезпечити користувача ефективними методами подолання стресових ситуацій та емоційних труднощів через персоналізоване спілкування. Завдяки використанню сучасних технологій аналізу тексту, Tess може визначати емоційні нюанси в повідомленнях та відповідно адаптувати свої поради.

В процесі взаємодії Tess аналізує текстові повідомлення користувача, що дозволяє точно визначати емоційний стан та виявляти сигнали, які можуть свідчити про потребу в додатковій підтримці. Завдяки такому підходу, Tess може пропонувати різноманітні методи саморегуляції, від технік релаксації до порад щодо подолання стресу.

Інтерфейс Tess наведено на рисунку 1.3.

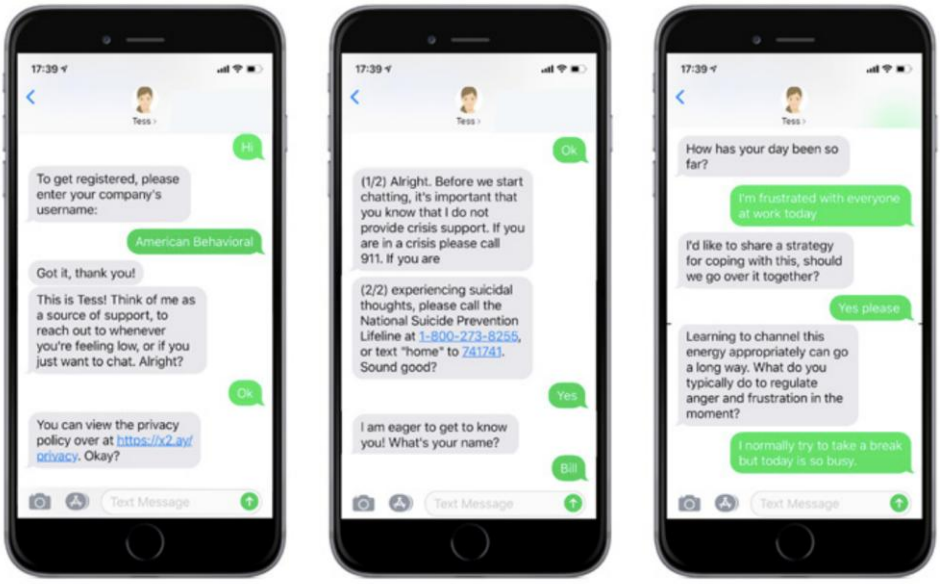


Рисунок 1.3 – Інтерфейс Tess

Для демонстрації основних можливостей та відмінності між розглянутими застосунками, їхніх переваг та недоліків, було сформовано таблицю порівняння

(таблиця 1.1).

Таблиця 1.1 – Порівняльний аналіз аналогів

| Характеристика | Replika | MindDoc | Tess | Власна розробка |
|---|---|---|---|---|
| Інтерактивний ШІ-помічник | + | - | + | + |
| Персоналізація відповідей | + | + | + | + |
| Моніторинг емоцій | + | + | + | + |
| Регулярне оцінювання настрою | - | + | - | + |
| Використання штучного інтелекту | + | + | + | + |
| Загальний бал | 4 | 4 | 4 | 5 |

Таким чином, власна розробка надає більш комплексний функціонал, порівняно із аналогами, забезпечує доступ до усіх необхідних функцій для визначення та управління емоційним станом, тому вона є актуальною.

### 1.3 Аналіз методів розв’язання задачі з розробки застосунку для управління емоційним станом людини

У контексті розробки застосунку для аналізу емоцій можливі п’ять архітектурно-технологічних підходів. Перший із них – повністю локальна розробка, коли всі алгоритми обробки даних (детекція обличчя через MediaPipe FaceMesh, обчислення EAR/MAR, PPG-аналіз для пульсу, локальні TensorFlow Lite-моделі для класифікації голосу та виразу) виконуються безпосередньо на мобільному пристрої. Така схема забезпечує мінімальні затримки та абсолютну конфіденційність користувацьких даних, проте обмежена доступними на смартфоні ресурсами, вимагає оптимізації великих моделей і ускладнює оновлення алгоритмів без перевипуску клієнтської аплікації.

Другий підхід – хмарний, за якого мобільний клієнт лише збирає відео-, аудіо-, текстові дані й відправляє їх на серверну інфраструктуру, де мікросервіси

використовують комп’ютерний зір, аудіо-класифікацію, NLP-аналіз та залучають сторонні великі мовні моделі. Централізоване оновлення моделей, масштабування за потреби та доступ до ресурсів дозволяють досягти високої точності й швидко вводити нові алгоритми, проте така архітектура залежить від стабільності мережі, характеризується неправомірними затримками і потребує додаткових витрат на хмарні обчислення.

Третій, гібридний підхід, у якому на пристрої виконується легкий попередній аналіз (правилово-евристичні алгоритми, базові CNN/TFLite-моделі для швидкого виявлення аномалій), а «важкі» завдання делегуються серверу за потреби. Така схема зберігає частину переваг локальної реалізації та одночасно дозволяє використовувати високопродуктивні моделі в хмарі, але вимагає ретельної синхронізації двох шарів.

З огляду на можливість використання готових платформ і мінімізацію витрат на розробку та підтримку власних моделей, найкращим вибором є cloud-centric архітектурний підхід із готовими ШІ-API. Він забезпечує доступ до перевірених алгоритмів комп’ютерного зору, голосового та текстового аналізу, а також LLM-контексту, дозволяючи зосередитися на інтеграції бізнес-логіки й інтерфейсу користувача, водночас оперативно отримуючи якісні результати без необхідності власноруч тренувати й оптимізувати ШІ-моделі. Такий метод дає змогу масштабувати рішення, оперативно впроваджувати оновлення та гарантує підтримку новітніх технологій із боку провайдерів API.

### 1.4 Постановка задач дослідження

Провівши аналіз методів розв’язання поставленої задачі, аналіз стану застосунку для визначення та управління емоційним станом людини, порівняння аналогів було поставлено такі задачі дослідження:

– розробити архітектуру застосунку для визначення та управління емоційним станом людини;

– розробити алгоритми роботи застосунку для визначення та управління емоційним станом людини;

– розробити функціонал застосунку для визначення та управління емоційним станом людини;

– розробити інтерфейс користувача застосунку;

– провести тестування розробленого застосунку.

Технічне завдання на розробку наведено в додатку А.

### 1.5 Висновки

У першому розділі було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів: Replika, MindDoc, Tess. В результаті аналізу було доведено актуальність власної розробки. Проведено аналіз методів розв’язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження.

# 2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ ЗСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ

## 2.1 Розробка структури застосунку

Система організована відповідно до принципів чистої архітектури з чітким розділенням відповідальності між компонентами. Нижче представлено формальний опис структури пакетів, що забезпечує логічну організацію коду та підвищує його ремонтопридатність.

Кореневий пакет com.emobalance слугує основою для всієї ієрархії пакетів застосунку і містить глобальні класи, зокрема клас Application, що є початковою точкою функціонування програмного забезпечення.

Клас `EmotionalWellbeingApplication` наслідує `Application`. При запуску він виконує ініціалізацію Hilt, налаштовує глобальні корутин-диспетчери та реєструє робітників WorkManager (зокрема `RegulationWorker`). Також тут відбувається конфігурація логування через Timber та завантаження користувацьких налаштувань із DataStore. Завдяки цим крокам створюється єдиний контекст додатка й забезпечується готовність усіх компонентів до роботи.

`Клас MainActivity` виконує встановлення Compose UI через `setContent {}` та налаштовує граф навігації (`NavHost`) для основних екранних маршрутів. На етапі створення він викликає `installSplashScreen()`, відображає splash-екран і чекає на завантаження початкових даних перед переходом до головного інтерфейсу. MainActivity також підключає підтримку WindowInsets. `STTResponse` – дата-клас, що описує відповідь сервісу Speech-to-Text. Містить поля для розпізнаного тексту, часових міток окремих слів та рівня впевненості алгоритму.

`ChatMessage` – модель одного повідомлення в чат-інтерфейсі (від користувача або бота). Поля включають роль (`role`), вміст (`content`) і час надходження (`timestamp`). Надає методи перевірки типу повідомлення й форматування для UI-віджетів. Служить основою для відображення історії діалогу в `ChatService` та пов’язаних екранах. Діаграму класу ChatMessage наведено на

рисунку 2.1.

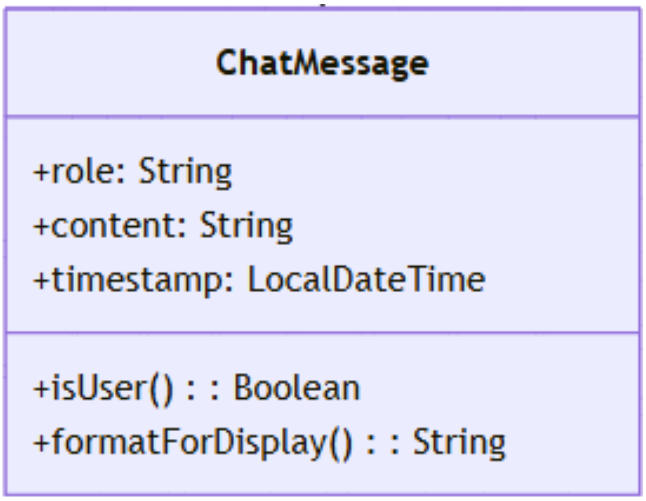


Рисунок 2.1 – Діаграма класу ChatMessage

`SentimentResponse` – клас для результатів тонального аналізу тексту. Містить показники валентності (`valence`), збудження (`arousal`) та детальний опис емоційного стану. Надає функцію `toSummary()`, яка збирає всі поля в один описовий рядок. Використовується при обробці відповіді від Anthropic чи Claude в `EmotionRepositoryImpl`.

`EmotionEntry` – клас для запису про емоційний стан. Містить унікальний ідентифікатор, часову мітку (`LocalDateTime`), рівень інтенсивності (`EmotionIntensity`), тегований контекст та необов'язковий об'єкт `AudioSentiment`. Реалізує методи конвертації в `EmotionEntryEntity` та навпаки. Використовується в бізнес-логіці для відображення історії емоцій (рисунок 2.2).

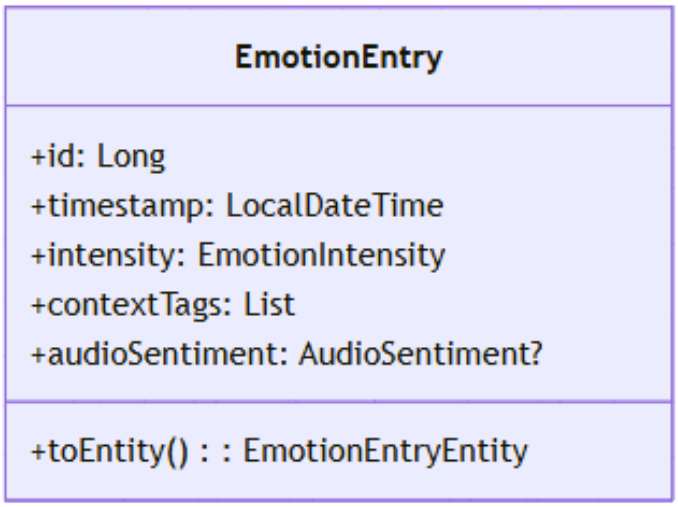


Рисунок 2.2 – Клас EmotionEntry

`EmotionInsight` – бізнес-сутність, що описує аналітичний висновок щодо емоцій. Поля включають тип інсайту (`InsightType`), трендовий бал (`trendScore`) і список рекомендацій. Має метод `formatSummary()`, який формує короткий звіт для UI. Застосовується в `GenerateRegulationStrategiesUseCase` та відображається на екрані Insights.

`InsightType` – перелічувальний тип (`enum`), що визначає категорії аналітичних інсайтів: `TREND`, `CORRELATION`, `PREDICTION`. Додає зручні функції для локалізації назв кожного типу при відображенні. Використовується в інтерфейсі користувача для вибору іконки та стилю секції. Дозволяє розрізняти логіку обробки різних видів інсайтів у кейсах та ViewModel (рисунок 2.3).

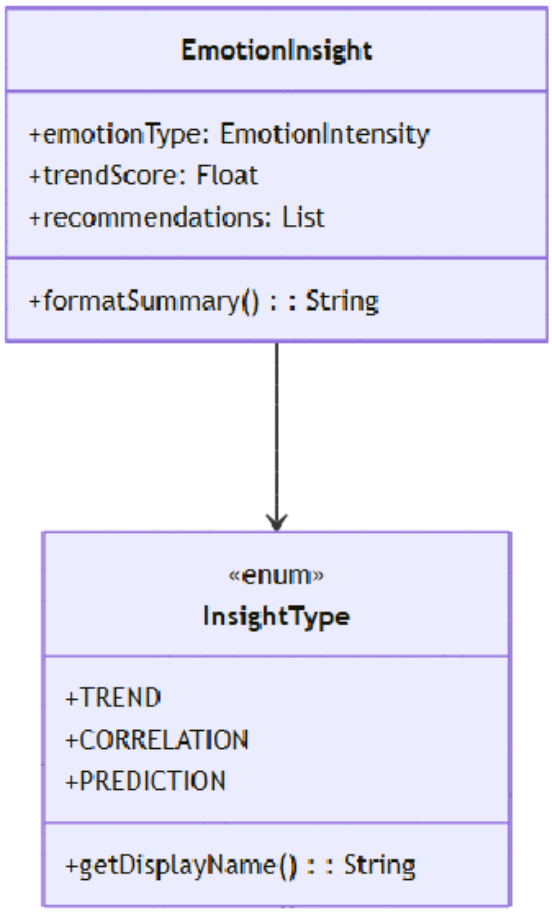


Рисунок 2.3 – Класи EmotionInsight та InsightType

`RegulationWorker` наслідує `CoroutineWorker` і відповідає за планові сповіщення з рекомендаціями з емоційного регулювання. У методі `doWork()` він завантажує записи за останню добу, виконує сценарій `GenerateRegulationStrategiesUseCase` і надсилає локальні нотифікації через `NotificationUtils`. Підтримує політику повторних спроб (backoff) у разі помилок мережі чи бази даних. Графік виконання налаштовується в Hilt-модулі або з зовнішньої конфігурації WorkManager.

Пакет di (Dependency Injection) містить компоненти, відповідальні за впровадження залежностей із використанням фреймворку Hilt. Даний пакет включає модулі, що забезпечують створення та надання залежностей у відповідних контекстах, таких як AppModule, RepositoryModule та інші. Пакет data відповідає за управління джерелами даних і складається з трьох підпакетів. Підпакет local відповідає за локальне зберігання інформації з використанням бібліотеки Room, включаючи визначення сутностей бази даних та

відповідних DAO. Підпакет remote забезпечує взаємодію з віддаленими API, зокрема з Claude API для аналізу емоцій. Підпакет repository містить конкретні реалізації репозиторіїв, що координують отримання та збереження даних від різних джерел.

Модуль Hilt для налаштування Room Database. Надає метод `provideAppDatabase()`, який створює екземпляр `AppDatabase` з конфігурацією міграцій і TypeConverters. Також визначає провайдер для `EmotionDao`. Цей модуль гарантує, що у всьому додатку використовується єдина інстанція бази даних із атомарними операціями.

`DataStoreModule` відповідає за створення `DataStore<Preferences>` через фабрику `PreferenceDataStoreFactory`. Надає обгортку `UserPreferences`, що інкапсулює зчитування та запис налаштувань (muteNotifications, syncFrequency, selectedModel). Забезпечує асинхронну обробку змін із підтримкою Kotlin Flow. Гарантує безпечний доступ до налаштувань із будь-якої гілки виконання.

`RetrofitClient` конфігурує та надає інстанцію `Retrofit` з базовим URL, `GsonConverterFactory` та кастомним `OkHttpClient`. `OkHttpClient` містить інтерцептор для логування запитів і `AuthInterceptor` для додавання API-ключів. Забезпечує таймаути та обробку помилок HTTP. Цей клас слугує єдиним джерелом HTTP-клієнта для всіх віддалених сервісів.

`ApiService` – інтерфейс REST-ендпоінтів для основного бекенду додатка. Містить методи `fetchEmotions()` для отримання записів з віддаленого сервера та `postEmotion()` для надсилання нових записів. Використовується `Retrofit` для автоматичної серіалізації/десеріалізації DTO. Інжектиться в репозиторії для координації локального і віддаленого сховища.

`ClaudeService` – обгортка для HTTP-викликів до Claude API (аналітика тональності). Надає метод `detectEmotion(text: String): EmotionDetectionDto`, який повертає детальний об’єкт із полями емоційного

спектру. Містить налаштування інтерцепторів і таймаутів, специфічні для API Anthropic. Інжектиться в `EmotionRepositoryImpl` для поєднання результатів аналізу з іншими джерелами.

`AnthropicApi` – інтерфейс або сервіс клас для викликів моделі від Anthropic. Має метод `analyzeText(input: String): SentimentResponse` через ендпоінт `/v1/complete`. Надає обробку помилок мережі та перетворення HTTP-помилок у винятки. Використовується в кейсах і репозиторіях для глибшого семантичного аналізу.

`ChatService` – сервіс для організації чат-сесій із LLM. Зберігає історію повідомлень і надсилає їх повністю в запиті, щоб підтримувати контекст. Має метод `sendMessage(messages: List<ChatMessage>): ChatMessage`, який повертає відповідь бота як `ChatMessage`. Підтримує стрімінг відповідей та інкрементальне відображення тексту в UI.

`VoiceRepository` – мервісний клас для завантаження аудіофайлів і отримання результатів Speech-to-Text. Метод `uploadAudio(file: File): STTResponse` виконує Multipart-запит до віддаленого сервера. Обробляє часові мітки й впевненість розпізнавання, перетворюючи їх у `STTResponse`. Підтримує retry-механізм на випадок мережевих помилок.

`EmotionRepositoryImpl` – реалізація інтерфейсу `EmotionRepository`. Інжектить `EmotionDao`, `AnthropicApi`, `ClaudeService` та `ApiService`. При додаванні запису спочатку зберігає дані локально, потім асинхронно надсилає їх на сервер. Методи `getEntries()` і `getInsights()` комбінують результати локального та віддаленого джерел із мапінгом у доменну модель. Підтримує коректну обробку помилок і відкат у разі збоїв. Клас `EmotionRepositoryImpl` наведено на рисунку 2.4.

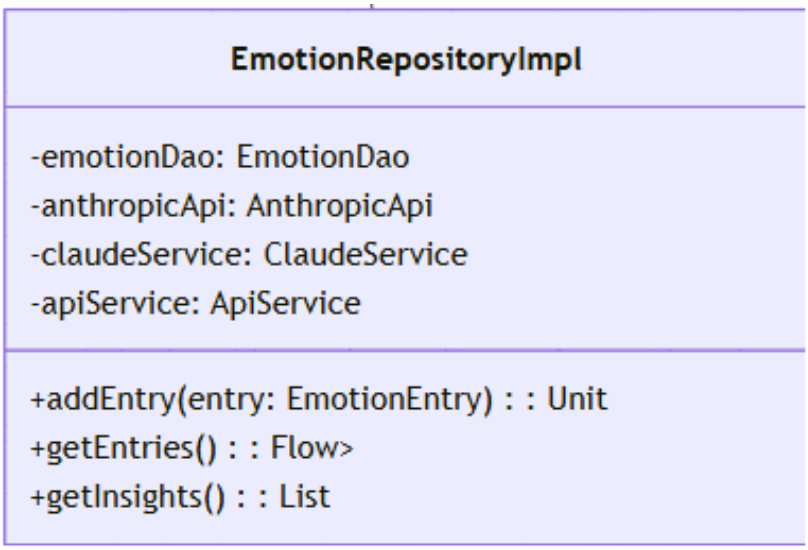


Рисунок 2.4 – Клас `EmotionRepositoryImpl`

`VoiceRepositoryImpl` інжектить `ApiService` для викликів STT-ендоінту. Обгортає HTTP-виклик у корутини та повертає структурований `STTResponse`. Надає обробку прогресу завантаження (через callback або Flow) та retry для надійності. Використовується в `VoiceAnalysisViewModel` для керування станом UI під час запису й розпізнавання.

Пакет domain представляє собою центральний шар бізнес-логіки, незалежний від зовнішніх технологій та фреймворків. У підпакеті model знаходяться бізнес-сутності, що представляють емоційні стани, техніки регуляції та інші об'єкти домену. Підпакет repository містить інтерфейси репозиторіїв, які визначають контракти доступу до даних без деталей реалізації. Підпакет usecase включає класи, які інкапсулюють окремі сценарії використання застосунку, такі як аналіз емоційного стану, прогнозування емоцій та генерація рекомендацій.

`EmotionRepository` – інтерфейс доменного шару для роботи із записами емоцій. Оголошує методи `suspend fun addEntry(entry: EmotionEntry)`, `fun getEntries(): Flow<List<EmotionEntry>>`, `suspend fun getInsights(): List<EmotionInsight>`. Визначає контракт без деталей реалізації. Дозволяє здобувати гнучкість і тестуємість бізнес-логіки.

`VoiceRepository` – інтерфейс для доменного рівня, що описує метод

`suspend fun transcribe(audioFile: File): STTResponse`. Використовується в кейсах і ViewModel для відокремлення логіки розпізнавання. Полегшує підстановку мок-реалізацій під час тестування. Забезпечує абстракцію над конкретними API.

`AnalyzeEmotionUseCase` – клас кейсу використання для аналізу тексту або транскрипту. Викликає методи `VoiceRepository` та/або `EmotionRepository`, щоб отримати початкові дані та сформувати `EmotionInsight`. Інкапсулює перетворення DTO у доменну модель. Забезпечує єдину точку входу для обчислення поточного емоційного стану.

`PredictEmotionUseCase` виконує прогнозування майбутніх емоційних станів на основі історичних даних. Застосовує статистичні методи (ковзне середнє) або ARIMA-модель, інкапсульовану всередині класу. Повертає список `EmotionInsight` із прогнозами на найближчі дні. Використовується для відображення графіків трендів на екрані Insights.

`GenerateRegulationStrategiesUseCase` формує набір рекомендацій для регуляції емоцій на основі поточного стану та контекстуальних факторів. Використовує вхідні `EmotionInsight` і `ContextualFactor` для створення персоналізованих порад. Повертає список рядків із кроками або посиланнями на статті. Викликається в `RegulationWorker` для формування щоденних сповіщень.

Пакет presentation відповідає за компоненти користувацького інтерфейсу та взаємодію з користувачем. Підпакет common містить спільні UI-компоненти та стилі, які використовуються по всьому застосунку. Підпакет navigation інкапсулює логіку навігації між екранами застосунку з використанням Jetpack Navigation. Підпакет features організований відповідно до основних функціональних модулів застосунку: dashboard для головного екрану з інформацією про поточний емоційний стан; analysis для детального аналізу мультимодальних даних; insights для прогнозування емоційних станів та стратегій регуляції; profile для управління налаштуваннями користувача.

`EmotionVisualizer` – Composable-функція, що відображає PieChart емоційних компонентів. Використовує `Canvas` і анімації `animateFloatAsState` для плавного переходу секторів. Приймає список `EmotionInsight` і масштабує їх за інтенсивністю. Підтримує адаптивний розмір і темну/світлу тему.

`VoiceWaveform` – Composable-компонент для візуалізації звукової хвилі під час запису. Приймає масив амплітуд, будує `Path` через `drawPath` і анімує переміщення вздовж осі X. Підключається до `VoiceAnalysisViewModel` через StateFlow прогресу. Додає інтерактивність: можливість паузи/продовження анімації.

`ContextualCorrelationChart` використовує бібліотеку Recharts (через WebView або нативний обгорток) для побудови бар-чарту кореляції контекстуальних факторів. Приймає список `ContextualFactor` і відображає стовпці з коефіцієнтами. Додає підписи та інтерактивні тултіпи. Підтримує масштабування жестами.

`EmotionCheckViewModel` інжектить `VoiceRepository` і `AnalyzeEmotionUseCase`. Керує станом запису (`isRecording`, `recordingProgress`), обробляє результат STT і аналізу за допомогою кейсу. Повертає UI-стан із блоком `EmotionInsight` для відображення на екрані. Підтримує обробку помилок запису й мережевих збоїв (рисунок 2.5).

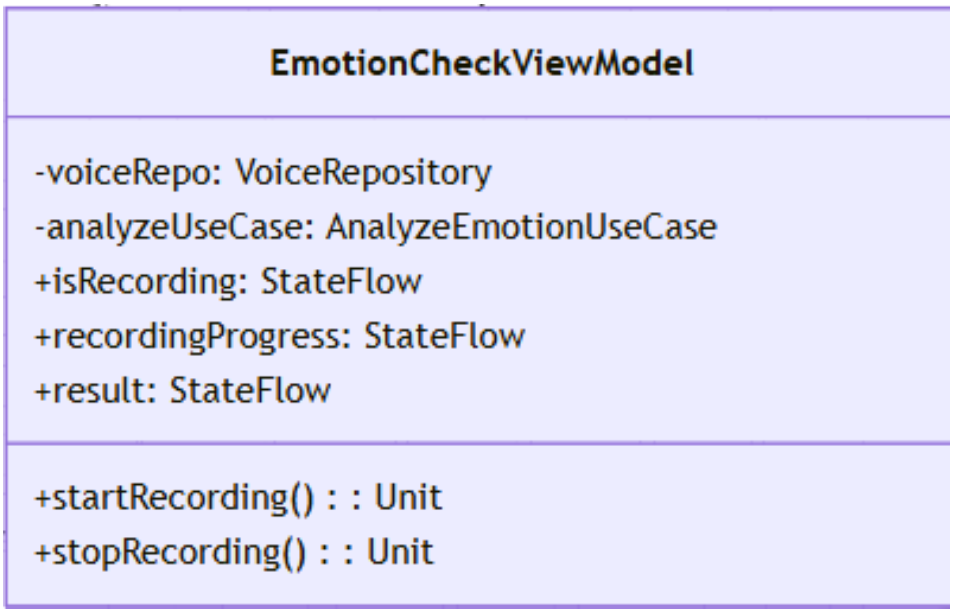


Рисунок 2.5 – Клас `EmotionCheckViewModel`

`EmotionJournalScreen` – Composable-екран, що відображає список попередніх записів. Використовує `LazyColumn` для відображення дат, іконок емоцій та контексту. Додає фільтри за датами та рівнями настрою. Після кліку на запис переходить до детального перегляду інсайтів.

`EmotionJournalViewModel` інжектить `EmotionRepository`. Підписується на Flow списку записів, фільтрує їх за вибраними критеріями дат. Підтримує сортування та пагінацію (через Paging3). Повідомляє UI про стан завантаження та помилки.

`ContextualInsightsScreen` – Composable-екран із таблицею топ-5 контекстуальних факторів і їх кореляційними коефіцієнтами. Підключається до `ContextualInsightsViewModel`. Додає кнопку “Детальніше”, яка відкриває графічну деталізацію у діалозі. Показує індикатор прогресу під час завантаження даних.

`ContextualInsightsViewModel` інжектить `PredictEmotionUseCase` і `EmotionRepository`. Виконує запит на прогнозування та кореляційний аналіз, формує список `ContextualFactor`. Повертає UI-стан із полями `factors`, `isLoading`, `error`. Підтримує оновлення даних за запитом користувача (рисунок 2.6).

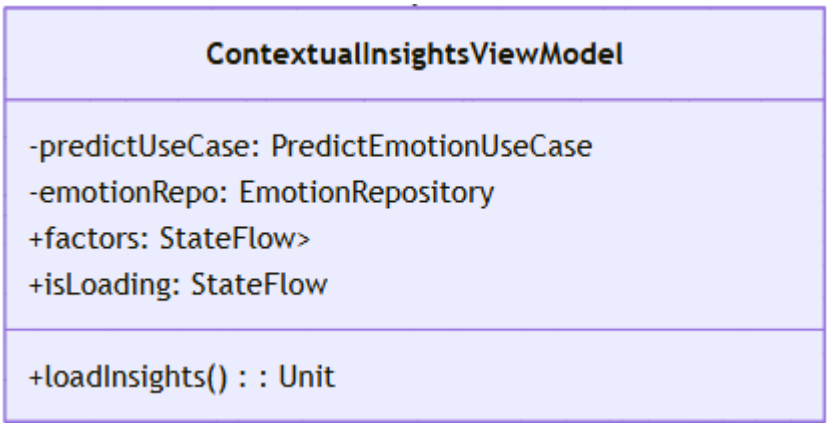


Рисунок 2.6 – Клас ContextualInsightsViewModel

`VoiceAnalysisScreen` – Composable-інтерфейс для роботи з голосом: кнопки Record/Pause/Stop, візуалізація форми хвилі, відображення транскрипції. Підключається до `VoiceAnalysisViewModel`. Додає live-streaming тексту під час обробки. Показує помилки розпізнавання та пропонує повторний запис.

`VoiceAnalysisViewModel` інжектить `VoiceRepository` і `AnalyzeEmotionUseCase`. Керує життєвим циклом `MediaRecorder`, передає аудіобуфери в репозиторій, отримує `STTResponse` і готує `EmotionInsight`. Відправляє UI-стан із оновленими даними та обробляє помилки мережі. Підтримує відміну операції через `viewModelScope.cancel()`.

`SettingsScreen` – Composable-екран із перемикачами (muteNotifications, syncFrequency), радіокнопками для вибору моделі (Claude vs Anthropic) та кнопкою “Експортувати дані”. Підключається до `SettingsViewModel`. Динамічно оновлює UI за змінами DataStore через Flow. Додає confirm-діалоги для змін.

`SettingsViewModel` інжектить `UserPreferences` із DataStore. Забезпечує методи `toggleNotifications()`, `setSyncFrequency(hours: Int)`, `selectModel(model: String)`. Повертає Flow налаштувань для UI та обробляє помилки запису в DataStore. Підтримує одночасне читання/запис через `viewModelScope` (рисунок 2.7).

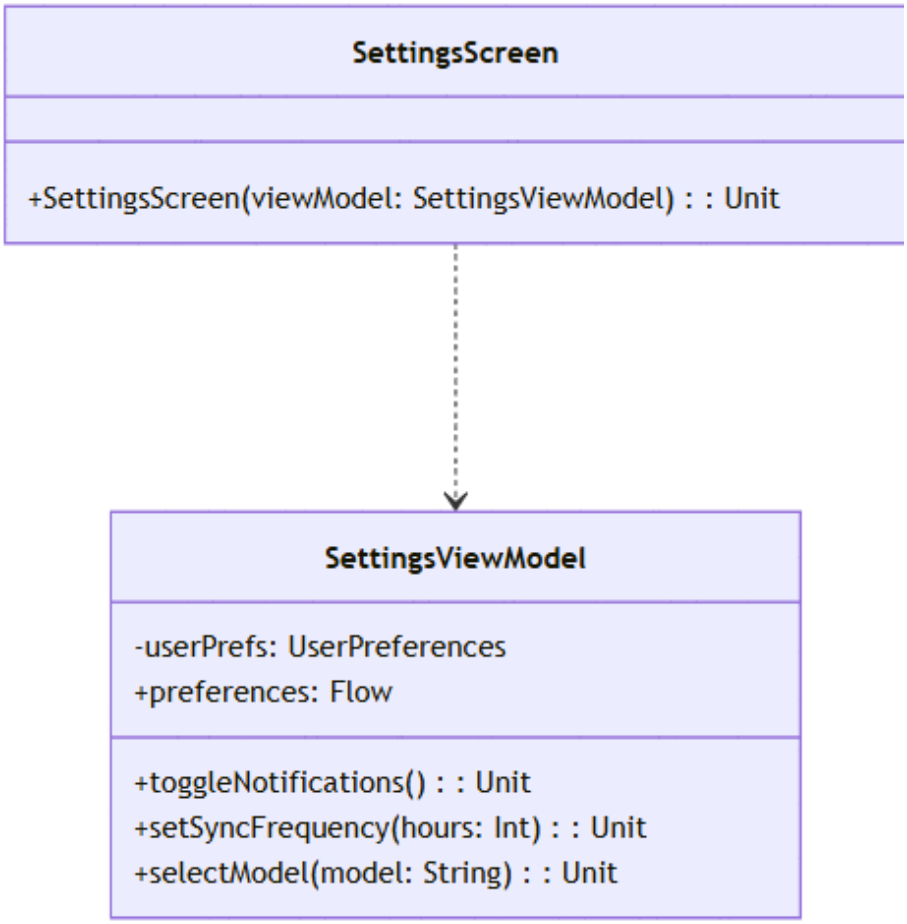


Рисунок 2.7 – Класи SettingsScreen та SettingsViewModel

Пакет utils містить утилітарні класи та функції, що забезпечують допоміжну функціональність для всіх інших компонентів системи, включаючи розширення Kotlin, утиліти для форматування даних, обробники помилок та інші спільні інструменти.

`ErrorHandling` оголошує sealed class `AppError` (Network, Unknown тощо). Має розширення `Throwable.toAppError()`, що перетворює винятки у зрозумілі помилки. Використовується в репозиторіях і кейсах для уніфікації обробки винятків. Підтримує UI-код у відображенні дружніх повідомлень про помилки.

Структура системи наведена на рисунку 2.8.

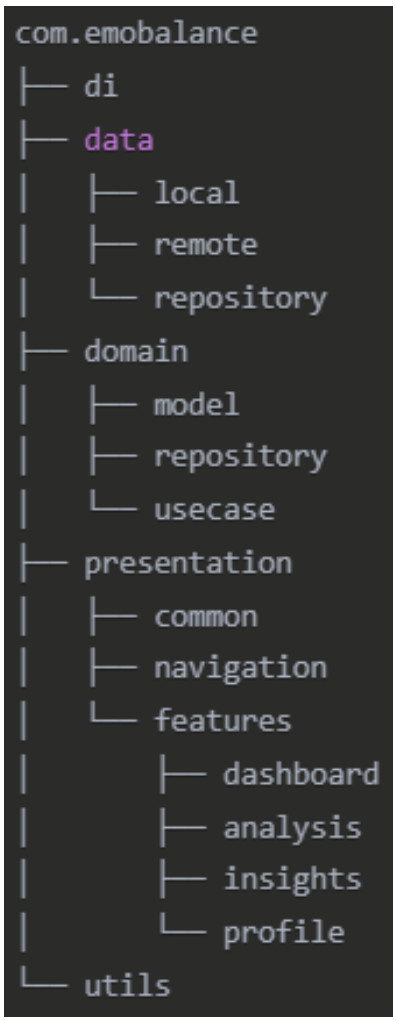


Рисунок 2.8 – Структура системи

Така архітектурна організація забезпечує чіткий поділ відповідальності між різними шарами застосунку, що відповідає принципам SOLID та дозволяє ізольовано тестувати компоненти. Крім того, така структура сприяє маштабованості системи та спрощує процес внесення змін, оскільки модифікації в одній частині застосунку мінімально впливають на інші компоненти.

### 2.2 Розробка моделі застосунку

Модель застосунку представлена діаграмою компонентів, що відображає технологічні складові та функціональні модулі застосунку. Jetpack Compose виступає основним технологічним компонентом для створення інтерфейсу користувача і прямо пов'язаний з Dashboard модулем, який є головним екраном застосунку. Ця зв'язка забезпечує сучасний, декларативний підхід до

розробки інтерфейсу користувача, що дозволяє створювати динамічні елементи інтерфейсу, які реагують на зміни емоційного стану користувача. Компонент Room відповідає за локальне зберігання даних і взаємодіє з системним модулем Storage. Ця взаємодія забезпечує надійну та структуровану базу даних для збереження історії емоційних станів, налаштувань користувача, когнітивних патернів та інших важливих даних застосунку.

Claude API є компонентом для аналізу емоцій і генерації рекомендацій, який прямо взаємодіє з UI модулем Analysis. Це дозволяє здійснювати мультимодальний аналіз емоційного стану користувача та надавати персоналізовані рекомендації за допомогою можливостей штучного інтелекту.

Компонент Hilt використовується для впровадження залежностей і пов'язаний з інтерфейсом користувача модулем Profile. Ця система забезпечує чисту архітектуру застосунку, полегшуючи тестування, мейнтейнабільність та розширюваність коду, особливо при роботі з налаштуваннями профілю користувача.

Завершує модель компонент Coroutines, який забезпечує асинхронне програмування і взаємодіє з системним модулем Async Engine. Ця взаємодія дозволяє ефективно виконувати операції з аналізу емоцій, прогнозування та отримання даних без блокування основного потоку, що важливо для плавної роботи застосунку з інтенсивними обчисленнями.

Модель системи наведена на рисунку 2.9.

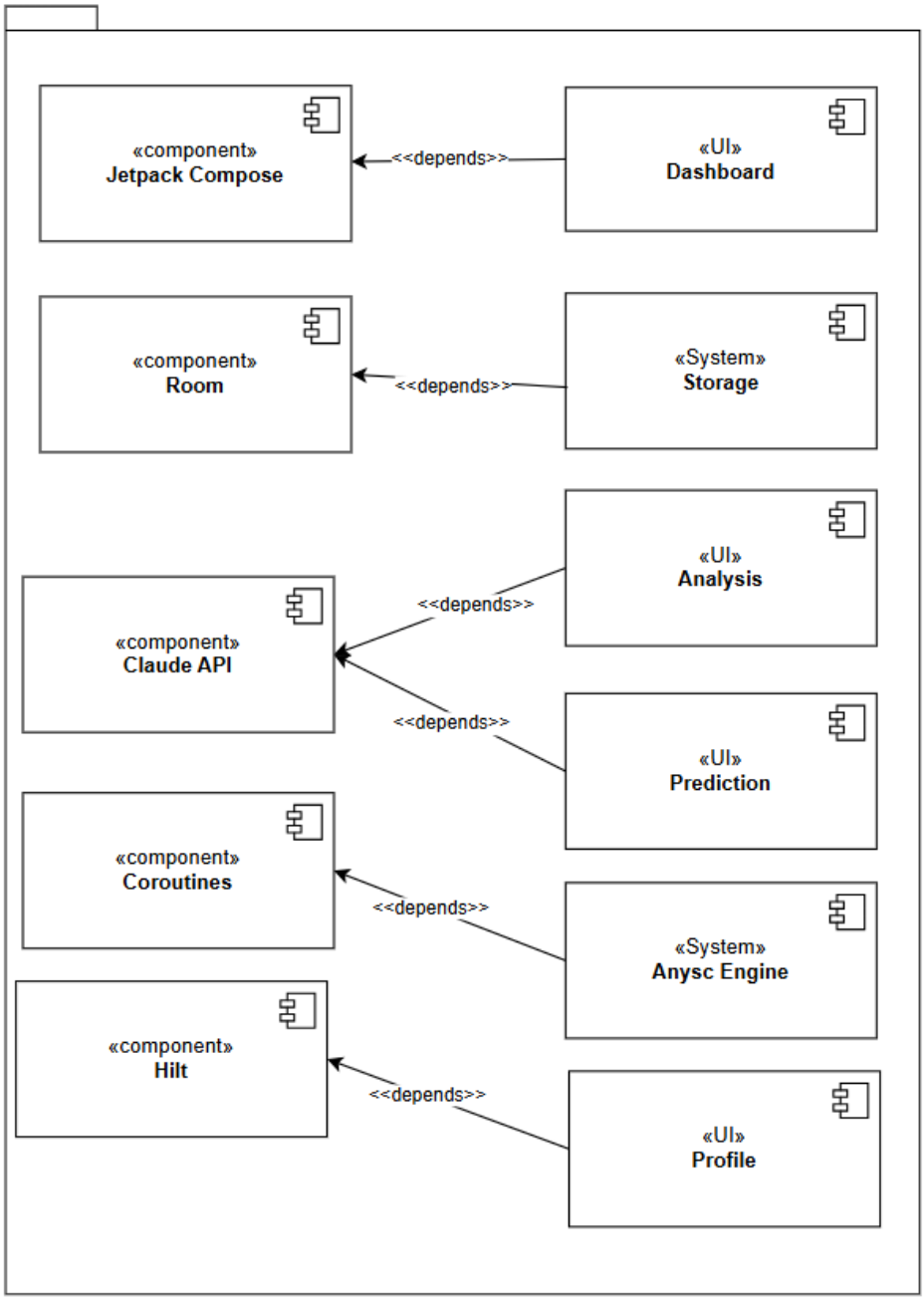


Рисунок 2.9 – Модель системи

Ця модель демонструє чітку структуру застосунку з розділенням на функціональні модулі та технологічні компоненти, що забезпечує оптимальну основу для реалізації двох методів: мультимодального аналізу емоцій з адаптивним зворотним зв'язком та предиктивної емоційної регуляції з когнітивним моделюванням.

### 2.3 Розробка методу аналізу голосу для визначення емоційного стану

Метод аналізу голосових характеристик для визначення емоційного стану користувача представляє собою інноваційний підхід до емоційного моніторингу. На відміну від традиційних методів самооцінки емоцій, які покладаються виключно на суб'єктивний звіт користувача, голосовий аналіз забезпечує об'єктивні дані, аналізуючи паравербальні характеристики мовлення, які часто відображають підсвідомі емоційні стани.

Це дозволяє виявляти емоції, які користувач може не усвідомлювати або не хотіти визнавати, забезпечуючи глибший рівень емоційного самопізнання. Технологія особливо цінна для людей, яким складно ідентифікувати власні емоції, та забезпечує додатковий шар даних для створення більш персоналізованих стратегій емоційного благополуччя.

Етапи роботи методу:

1. Користувач активує функцію запису голосу та розповідає про свій поточний стан або день.

2. Застосунок записує аудіо через MediaRecorder та візуалізує звукову хвилю в реальному часі.

3. Після завершення запису, аудіо аналізується для вилучення характеристик: темпу мовлення, висоти голосу, варіацій гучності та ритмічних патернів.

Ці характеристики передаються до Claude API для аналізу, де вони порівнюються з патернами, характерними для різних емоційних станів.

4. API повертає визначену емоцію, рівень впевненості в цьому визначенні та додаткові метрики.

5. На основі виявленої емоції та контексту користувача генерується персоналізована стратегія подолання або рекомендація.

6. Результати відображаються користувачу візуально, з поясненням виявлених голосових характеристик та їх зв'язку з емоційним станом.

Блок-схема процесу аналізу голосу наведена на рисунку 2.10.

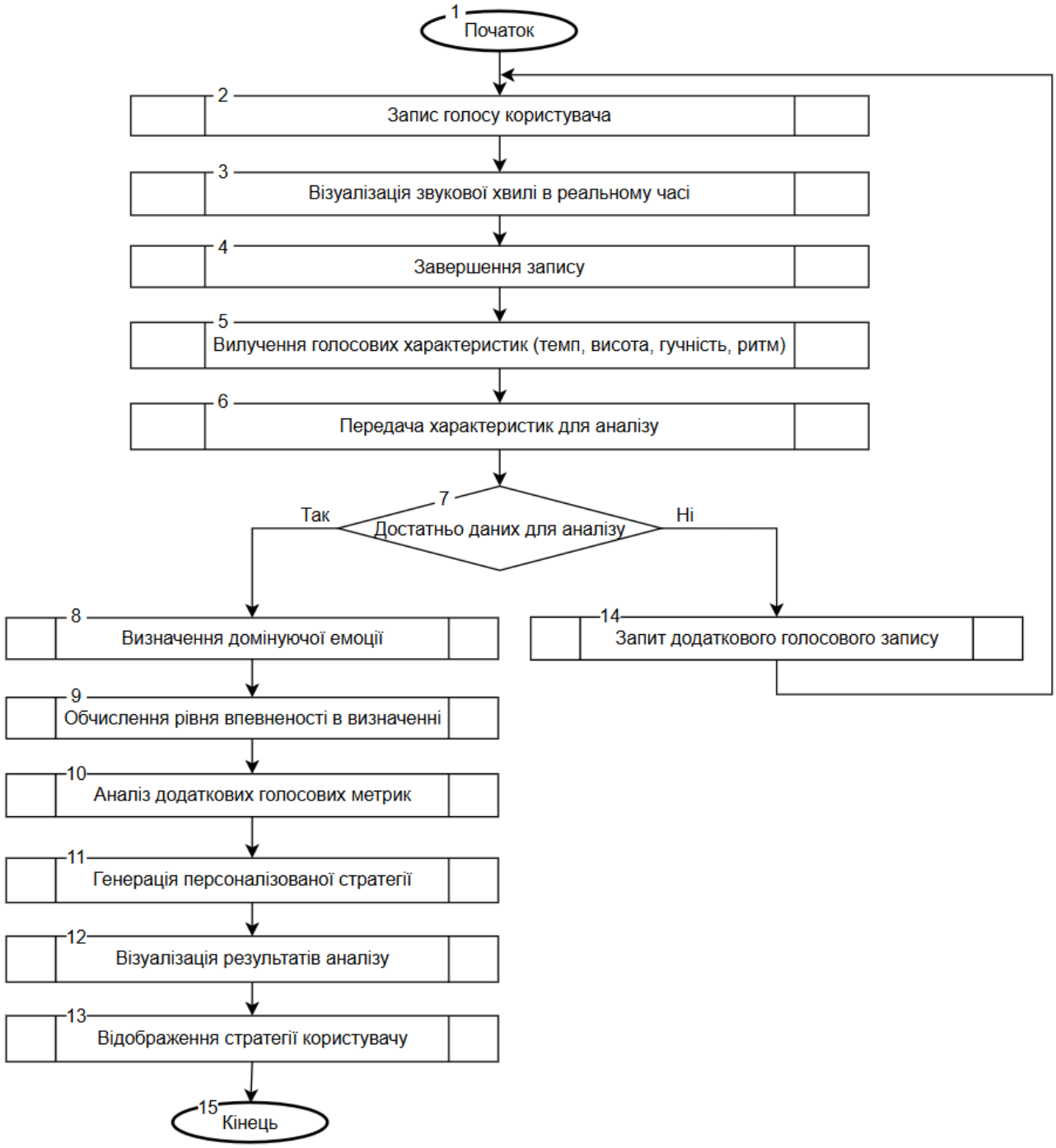


Рисунок 2.10 – Блок-схема процесу аналізу голосу

Впровадження голосового аналізу емоцій розширює можливості застосунку, переводячи його з простого інструменту відстеження настрою в комплексну систему емоційної аналітики. Забезпечуючи багатовимірний підхід до розуміння емоційного стану, застосунок дозволяє користувачам отримати глибші інсайти про свій емоційний досвід та більш ефективно керувати своїм психологічним благополуччям.

### 2.4 Розробка методу контекстуального прогнозу емоційного стану

Метод контекстуального прогнозування емоційного стану використовує аналіз історичних даних, контекстуальних факторів та патернів емоційних реакцій для передбачення можливих майбутніх емоційних станів користувача. На відміну від існуючих методів, які лише реагують на поточні емоції, цей підхід активно випереджає потенційні емоційні труднощі, дозволяючи користувачам підготуватись до них.

Інновація полягає у поєднанні аналізу часових залежностей, контекстуальних кореляцій та машинного навчання для створення проактивних рекомендацій. Це особливо корисно для людей, які прагнуть покращити емоційну регуляцію або мають схильність до частих емоційних коливань, оскільки система діє як емоційний компас, пропонуючи підготовчі стратегії до того, як виникнуть складні емоційні ситуації.

Етапи роботи методу:

1.Система збирає та аналізує історичні дані про емоційні стани користувача, їх інтенсивність та контекстуальні фактори, пов'язані з кожним записом.

2.Алгоритм виявляє статистично значущі кореляції між конкретними контекстуальними факторами (погода, соціальні взаємодії, сон, фізична активність) та емоційними реакціями.

3.На основі виявлених патернів створюється модель прогнозування, що враховує циклічність емоційних станів та значущі тригери.

4.Система аналізує майбутні події з календаря користувача, прогноз погоди, розклад дня та інші доступні дані.

5.Порівнюючи ці майбутні контекстуальні фактори з виявленими емоційними патернами, алгоритм визначає потенційні емоційні тригери на наступний день.

6.Для кожного передбаченого тригера система генерує персоналізовані превентивні стратегії та рекомендації.

7.Користувач отримує повідомлення з прогнозом потенційних емоційних

труднощів та конкретними підготовчими діями.

Блок-схема процесу прогнозування наведена на рисунку 2.11.

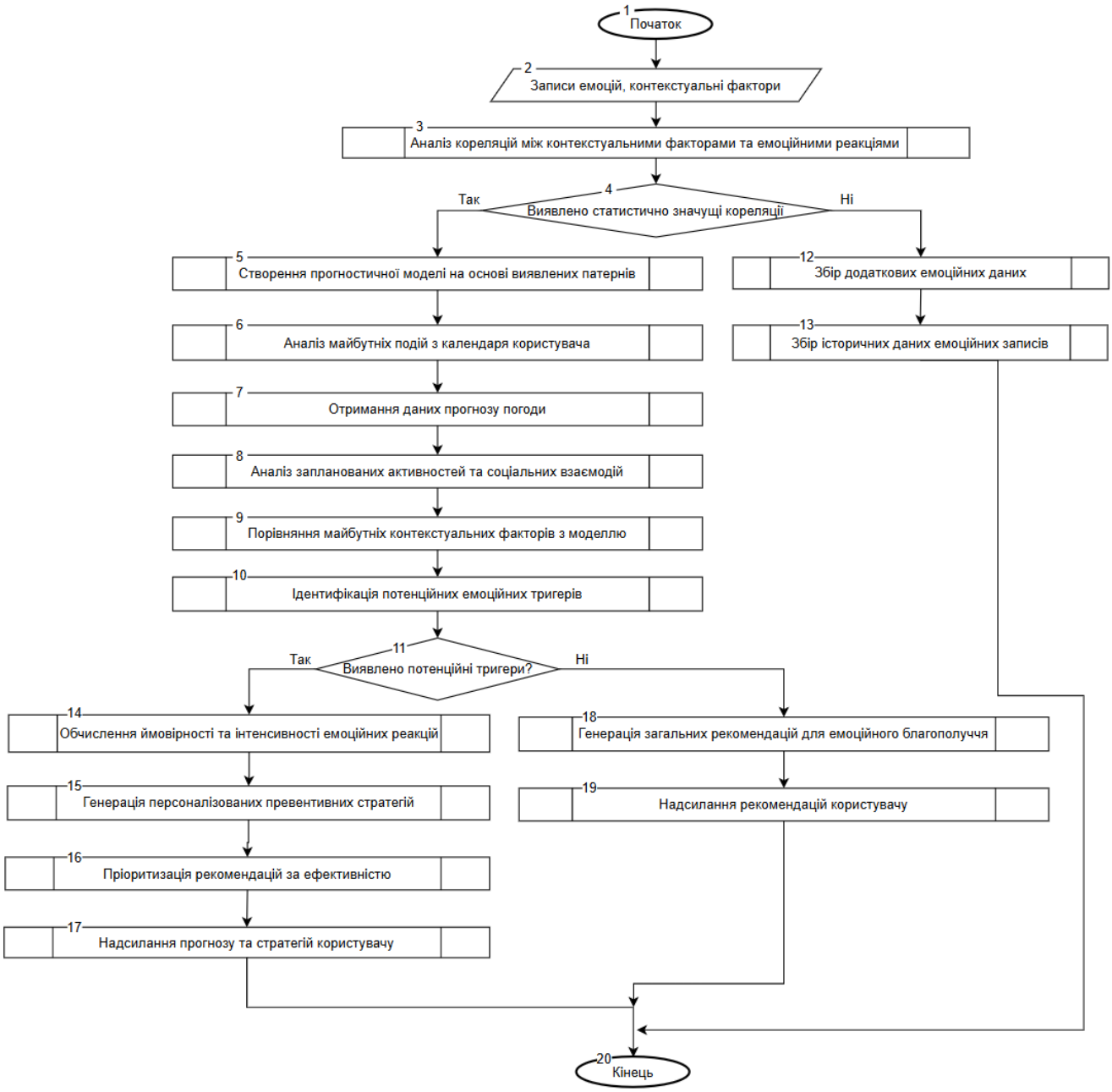


Рисунок 2.11 – Блок-схема процесу контекстуального прогнозування

Впровадження контекстуального прогнозування емоцій трансформує підхід до емоційного благополуччя від реактивного до проактивного. Подібно до того, як прогноз погоди дозволяє людям готуватися до дощу, емоційний прогноз дає користувачам можливість підготуватися до емоційних "штормів", підвищуючи їх здатність до саморегуляції та загальний рівень емоційної стійкості.

## 2.5 Розробка алгоритмів роботи застосунку для управління емоційним станом людини

Розроблено алгоритм збору та аналізу контекстуальних факторів. Блок-схема цього алгоритму наведена на рисунку 2.12.

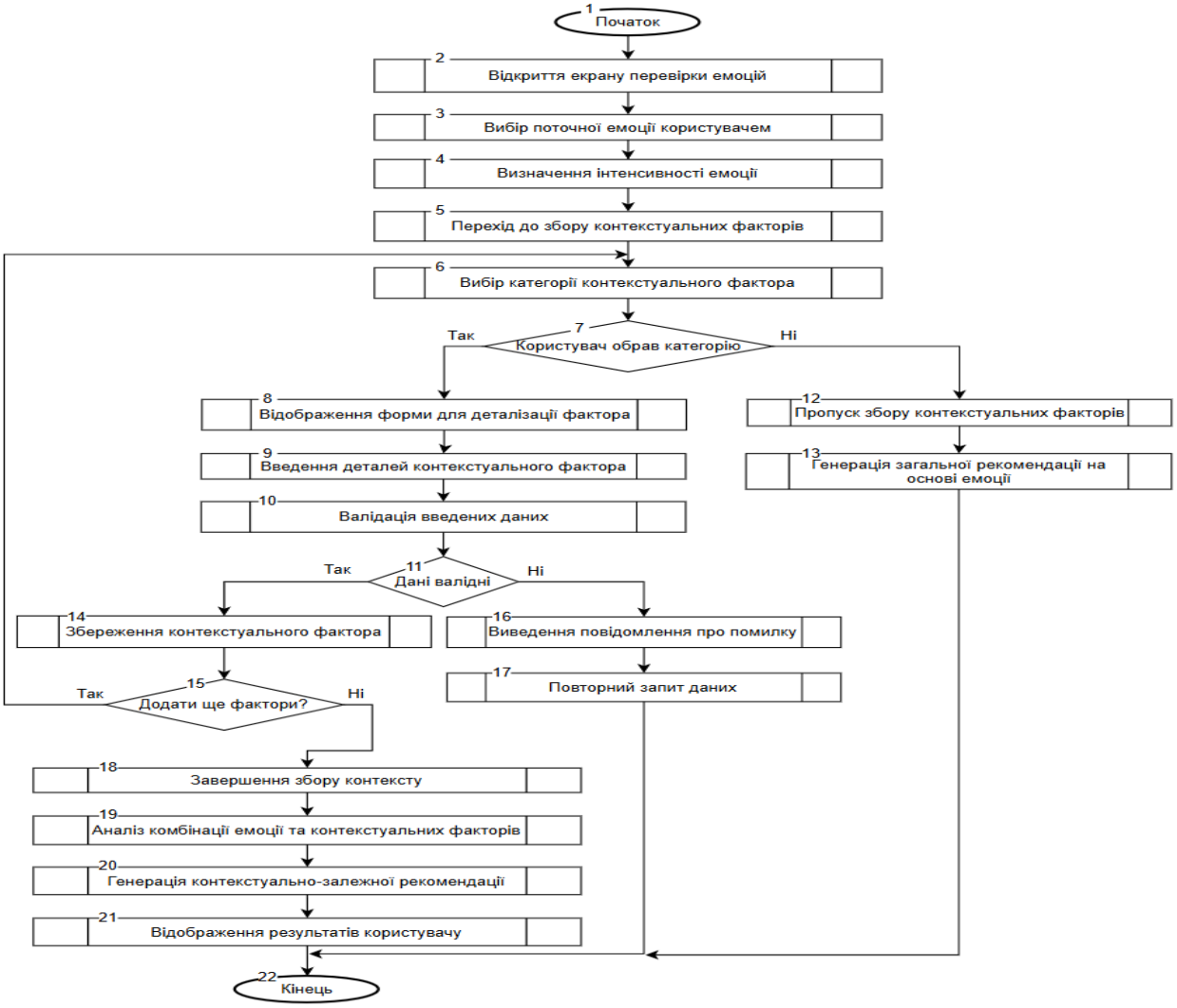


Рисунок 2.12 – Блок-схема алгоритму збору та аналізу контекстуальних факторів

Цей алгоритм забезпечує збір та аналіз важливих контекстуальних даних, що впливають на емоційний стан користувача. Розпочинаючи з вибору поточної емоції та її інтенсивності, система переходить до структурованого збору контекстуальної інформації за різними категоріями: погодні умови, місцезнаходження, фізична активність, соціальні взаємодії тощо.

Для кожної обраної категорії користувач може ввести деталізовану інформацію, яка проходить валідацію перед збереженням. Перевага алгоритму полягає у створенні багатовимірної картини емоційного стану, де враховуються не лише самі емоції, але й ситуативні фактори, що їх супроводжують. На основі комплексного аналізу поєднання емоції з контекстуальними факторами генеруються високоперсоналізовані рекомендації, що враховують специфіку конкретної ситуації користувача.

Алгоритм відстеження емоційних патернів у часі зосереджений на аналізі емоційної динаміки користувача в часовій перспективі. Блок-схема алгоритму відстеження емоційних патернів у часі наведено на рисунку 2.13.

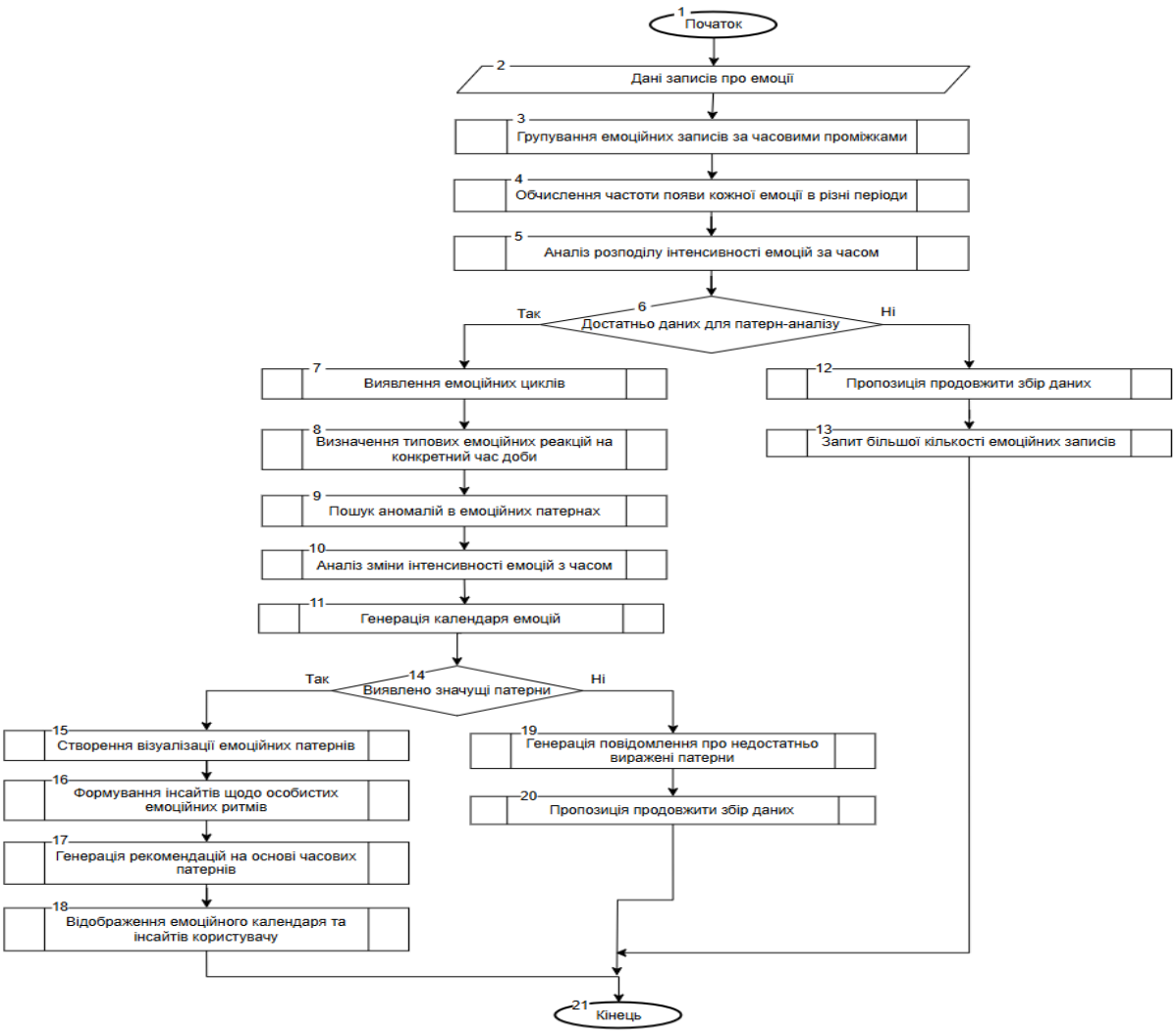


Рисунок 2.13 – Блок-схема алгоритму відстеження емоційних патернів у часі

Система завантажує історичні дані емоційних записів і групує їх за

часовими проміжками (години, дні, тижні), що дозволяє виявити циклічні патерни та залежності.

Аналіз включає обчислення частоти появи різних емоцій у конкретні періоди, розподіл інтенсивності емоційних станів за часом, та пошук аномалій, що виходять за межі типових для користувача патернів.

Основна цінність алгоритму полягає у здатності виявляти неочевидні темпоральні залежності – наприклад, регулярні емоційні спади в певні дні тижня чи години доби.

На основі виявлених значущих патернів створюється візуалізація у формі емоційного календаря та формуються інсайти щодо особистих емоційних ритмів, доповнені персоналізованими рекомендаціями з урахуванням цих часових закономірностей.

Розроблено алгоритм персоналізації стратегій емоційного регулювання, який розпочинає свою роботу з фіксації поточної емоції користувача, її інтенсивності та контексту (час доби, фізіологічний стан, обставини), а також завантаження історії минулих успішних втручань.

Далі відбувається оцінка необхідності регулювання: якщо інтенсивність не перевищує встановленого порогу, система пропонує загальні техніки підтримки благополуччя. Якщо ж потрібне втручання, емоція класифікується за категоріями, і алгоритм шукає в історії максимально схожі випадки; за їх відсутності звертається до загальної бази перевірених стратегій.

Після відбору початкового набору стратегій система ранжує їх за ефективністю або адаптує загальні підходи до поточного індивідуального контексту. Отримані рекомендації перетворюються на чіткий послідовний план дій та відображаються в інтерфейсі.

Після застосування стратегії алгоритм запитує зворотний зв’язок, зберігає оцінки й відповідні дані в історії, щоб у подальшому підвищити точність і персоналізацію рекомендацій.

Блок-схема цього алгоритму наведена на рисунку 2.14.


Рисунок 2.14 – Блок-схема алгоритму персоналізації стратегій емоційного регулювання

Якщо регуляція потрібна, алгоритм спочатку категоризує емоцію за різними параметрами (позитивна/негативна, активуюча/деактивуюча) і шукає схожі ситуації в історії користувача. Особливість підходу полягає в аналізі ефективності раніше застосованих стратегій і їх ранжуванні на основі попереднього досвіду конкретного користувача. Якщо подібних ситуацій не знайдено, система використовує загальну базу знань, але адаптує стратегії до індивідуального контексту. Після застосування рекомендованої стратегії, алгоритм збирає

зворотний зв'язок про її дієвість, що дозволяє постійно вдосконалювати точність і релевантність майбутніх рекомендацій.

Алгоритм інтерактивного емоційного щоденника представляє собою передову систему для всебічного відстеження, аналізу та роботи з емоційними даними користувача.

Блок-схема алгоритму наведена на рисунку 2.15.


Рисунок 2.15 – Блок-схема алгоритму емоційного щоденника

На відміну від традиційних щоденників настрою, які зазвичай обмежуються простою фіксацією емоційних станів, даний алгоритм створює динамічне

середовище, де кожен запис стає відправною точкою для глибшого самопізнання та зростання. Система дозволяє працювати з емоційними даними на різних часових горизонтах – від денного до місячного, забезпечуючи комплексний огляд емоційної динаміки.

Головна особливість алгоритму полягає в його інтерактивності та багатовимірності. Користувач може не лише переглядати хронологію своїх емоційних станів, але й активно взаємодіяти з кожним записом: розгортати деталізовану інформацію про контекст виникнення емоції, перегляати тригери, що її викликали, та стратегії, які були застосовані для регулювання. При цьому система підтримує можливість редагування історичних записів для уточнення даних, а також додавання відкладених рефлексивних нотаток, які з'являються після осмислення ситуації.

Функція запиту комплексних інсайтів є інтегральною частиною алгоритму, яка акумулює та синтезує емоційні дані за обраний період. Система застосовує багатовимірний аналіз для виявлення латентних патернів, емоційних тригерів та ефективних стратегій регулювання, характерних для конкретного користувача.

Таким чином, було розроблено алгоритми для застосунку управління емоційним станом. Розроблені алгоритми дозволяють додавати записи про свій настрій, отримувати рекомендації та персоналізовані стратегії регулювання емоцій.

### 2.6 Висновки

У другому розділі було розроблено структуру застосунку для управління емоційним станом, розроблено модель системи, алгоритми, метод аналізу голосу для визначення емоційного стану та метод контекстуального прогнозу емоційного стану.

# 3 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ТА УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ

## 3.1 Варіантний аналіз та вибір мови програмування розробки

Розглянемо такі мови програмування, як Kotlin, Java, Swift.

Kotlin [18] – сучасна статично типізована мова програмування, розроблена компанією JetBrains. Вона працює на Java Virtual Machine (JVM) та офіційно підтримується Google як основна мова для розробки Android-застосунків.

Kotlin поєднує об'єктно-орієнтований та функціональний підходи до програмування, пропонуючи конциний синтаксис, nullability в системі типів, розширення функцій, лямбда-вирази, та корутини для асинхронного програмування. Це дозволяє розробляти більш безпечний, лаконічний та зрозумілий код порівняно з Java.

Java [19] – об'єктно-орієнтована мова програмування, що була тривалий час основною для розробки Android-застосунків. Вона характеризується строгою типізацією, високою переносимістю завдяки принципу "write once, run anywhere" та автоматичним керуванням пам'яттю через збирач сміття. Java має велику екосистему бібліотек та інструментів, а також значну базу розробників. Проте порівняно з новішими мовами, Java має більш багатослівний синтаксис і повільніше впроваджує сучасні функції програмування.

Swift [20] – мова програмування, розроблена Apple для iOS, macOS та інших платформ Apple. Вона створена як заміна Objective-C з акцентом на безпеку, швидкість та сучасний синтаксис.

Swift пропонує автоматичний підрахунок посилань, опціональні типи, функціональні можливості та інтеграцію з фреймворками Apple. Мова динамічно розвивається і поступово стає міжплатформною, хоча її основне використання залишається в екосистемі Apple.

Для порівняння можливостей цих мов програмування сформуємо таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз мов програмування

| Характеристика | Kotlin | Java | Swift |
|---|---|---|---|
| Null безпека | + | - | + |
| Функції вищого порядку | + | + | + |
| Нативна асинхронність | + | + | + |
| Мультиплатформна розробка | + | + | - |
| Функції-розширення | + | - | + |
| Повноцінна підтримка функціонального програмування | + | + | + |
| Сумарний бал | 6 | 4 | 5 |

Таким чином, Kotlin має більше можливостей порівняно з Java та Swift, особливо в контексті розробки мобільний застосунків, що робить його оптимальним вибором для розробки системи.

### 3.2 Розробка бази даних застосунку

База даних у системі для визначення та управління емоційним станом людини відіграє важливу роль у забезпеченні персоналізованого досвіду управління емоційним станом. Вона служить фундаментальним компонентом для зберігання, аналізу та використання даних [21].

Реляційна структура бази даних дозволяє відстежувати динаміку емоційних станів користувача в часі, зберігати інформацію про дієвість різних стратегій регуляції емоцій, фіксувати індивідуальні когнітивні патерни та тригери емоційних реакцій. Така організація інформації забезпечує постійне вдосконалення алгоритмів прогнозування та рекомендацій, сприяючи формуванню цифрового емоційного профілю користувача, який з часом стає дедалі точнішим і персоналізованим.

Таблиця emotions зберігає історію емоційних станів користувача з часовою міткою та детальним описом. Ця таблиця є основною для відстеження змін емоційного стану та подальшого аналізу трендів. Поля цієї таблиці наведені у таблиці 3.2.

Таблиця 3.2 – Поля таблиці emotions

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, унікальний ідентифікатор запису (UUID) |
| primaryEmotion | TEXT | Основна визначена емоція (JOY, SADNESS, ANGER, тощо) |
| intensity | REAL | Інтенсивність емоції (від 0.0 до 1.0) |
| secondaryEmotions | TEXT | JSON-масив вторинних емоцій |
| timestamp | LONG | Час запису емоційного стану (в мілісекундах) |
| confidence | REAL | Впевненість у визначенні емоції (від 0.0 до 1.0) |
| description | TEXT | Текстовий опис емоційного стану |

Таблиця user_preferences зберігає налаштування користувача, включаючи параметри сповіщень, збору даних, втручань та приватності. Ця таблиця забезпечує персоналізовану роботу додатку відповідно до вподобань користувача. Поля таблиці user_preferences наведені у таблиці 3.3.

Таблиця 3.3 – Поля таблиці user_preferences

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, завжди "user_preferences" |
| notificationsEnabled | INTEGER | Загальний перемикач сповіщень (0/1) |
| emotionalStateNotifications | INTEGER | Сповіщення про емоційний стан (0/1) |
| predictionAlerts | INTEGER | Сповіщення про прогнози (0/1) |
| interventionReminders | INTEGER | Нагадування про регуляцію емоцій (0/1) |
| quietHoursStartHour | INTEGER | Година початку "тихого часу" (0-23) |
| quietHoursStartMinute | INTEGER | Хвилина початку "тихого часу" (0-59) |
| quietHoursEndHour | INTEGER | Година завершення "тихого часу" (0-23) |
| quietHoursEndMinute | INTEGER | Хвилина завершення "тихого часу" (0-59) |
| allowFacialAnalysis | INTEGER | Дозвіл на аналіз виразу обличчя (0/1) |
| allowVoiceAnalysis | INTEGER | Дозвіл на аналіз голосу (0/1) |
| allowBiometricAnalysis | INTEGER | Дозвіл на аналіз біометрики (0/1) |

Продовження таблиці 3.3

| Поле | Тип | Опис |
|---|---|---|
| allowLocationTracking | INTEGER | Дозвіл на відстеження місцезнаходження (0/1) |
| allowMessageAnalysis | INTEGER | Дозвіл на аналіз повідомлень (0/1) |
| preferredInterventionTypes | TEXT | JSON-масив преферентних категорій втручань |
| maximumInterventionDurationMinutes | LONG | Максимальна тривалість втручань (у хвилинах) |
| interventionFrequency | TEXT | Частота втручань (LOW, MEDIUM, HIGH) |
| dataStorageDurationDays | LONG | Тривалість зберігання даних (у днях) |
| allowAnonymizedDataSharing | INTEGER | Дозвіл на анонімний обмін даними (0/1) |
| encryptSensitiveData | INTEGER | Шифрування чутливих даних (0/1) |
| cognitiveModelJson | TEXT | JSON-представлення когнітивної моделі користувача |

Таблиця emotion_regulation_techniques зберігає різноманітні техніки регуляції емоцій, які можуть бути рекомендовані користувачу. Ця таблиця містить детальні інструкції та метадані для кожної техніки.

Таблиця 3.4 – Поля таблиці emotion_regulation_techniques

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, унікальний ідентифікатор техніки |
| title | TEXT | Назва техніки регуляції емоцій |
| description | TEXT | Детальний опис техніки |
| steps | TEXT | JSON-масив кроків для виконання техніки |
| baseEffectiveness | REAL | Базова дієвість техніки (від 0.0 до 1.0) |
| targetEmotions | TEXT | JSON-масив емоцій, на які націлена техніка |
| durationMinutes | INTEGER | Рекомендована тривалість виконання (у хвилинах) |
| category | TEXT | Категорія техніки (BREATHING_EXERCISE, COGNITIVE_REFRAMING, тощо) |

Таблиця cognitive_patterns зберігає виявлені когнітивні патерни користувача,

які впливають на його емоційний стан. Ця таблиця допомагає персоналізувати рекомендації та аналіз.

Таблиця 3.5 – Поля таблиці cognitive_patterns

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, унікальний ідентифікатор патерну |
| patternType | TEXT | Тип когнітивного патерну (CATASTROPHIZING, BLACK_AND_WHITE_THINKING, тощо) |
| description | TEXT | Опис конкретного прояву патерну у користувача |
| intensity | REAL | Інтенсивність патерну (від 0.0 до 1.0) |
| associatedEmotions | TEXT | JSON-масив емоцій, пов'язаних з патерном |
| detectionDate | LONG | Дата виявлення патерну (в мілісекундах) |

Таблиця emotional_triggers відстежує фактори, які викликають певні емоційні реакції у користувача. Ця таблиця допомагає аналізувати причини емоційних станів та робити більш точні прогнози.

Таблиця 3.6 – Поля таблиці emotional_triggers

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, унікальний ідентифікатор тригера |
| source | TEXT | Джерело тригера (CALENDAR, COMMUNICATION, LOCATION, тощо) |
| description | TEXT | Опис тригера |
| potentialImpact | REAL | Потенційний емоційний вплив (від 0.0 до 1.0) |
| anticipatedEmotion | TEXT | Очікувана емоційна реакція |
| detectionDate | LONG | Дата виявлення тригера (в мілісекундах) |

Таблиця intervention_history зберігає історію застосованих користувачем технік регуляції емоцій та їх дієвість. Ця таблиця дозволяє аналізувати дієвість різних стратегій для конкретного користувача.

Таблиця 3.7 – Поля таблиці intervention_history

| Поле | Тип | Опис |
|---|---|---|
| id | TEXT | Первинний ключ, унікальний ідентифікатор запису |
| techniqueId | TEXT | Зовнішній ключ до emotion_regulation_techniques.id |
| startTime | LONG | Час початку застосування техніки (в мілісекундах) |
| durationSeconds | INTEGER | Фактична тривалість застосування (в секундах) |
| emotionBeforeId | TEXT | Зовнішній ключ до emotions.id (стан до) |
| emotionAfterId | TEXT | Зовнішній ключ до emotions.id (стан після) |
| userRating | INTEGER | Оцінка ефективності від користувача (1-5) |
| calculatedEffectiveness | REAL | Розрахована дієвість (від 0.0 до 1.0) |
| notes | TEXT | Нотатки користувача |

База даних побудована за принципами реляційної моделі та підтримує цілісність даних через зовнішні ключі. Основна таблиця `emotions` зберігає історію емоційних станів і пов'язана з іншими таблицями для забезпечення контексту, аналізу причин та відстеження ефективності інтервенцій.

Структура оптимізована для швидкого доступу до поточного та історичних емоційних станів, персоналізованих рекомендацій на основі когнітивної моделі користувача, аналізу тригерів та патернів для точного прогнозування емоцій, вимірювання ефективності різних стратегій регуляції емоцій, контекстуального розуміння емоційних станів користувача.

## 3.3 Розробка інтерфейсу користувача застосунку для управління емоційним станом людини

Інтерфейс користувача застосунку дозволяє зручно та легко додавати записи про свій емоційний стан, переглядати статистику та аналіз свого емоційного стану за різні періоди часу, записувати голосові повідомлення та отримувати аналіз настрою за цими голосовими повідомлення.

Розроблено інтерфейс головного вікна застосунку, який є точкою входу в

застосунок, звідки користувач може отримати доступ до всіх основних функцій застосунку. Воно містить персональне привітання, кнопки для відстеження щоденного емоційного стану, швидкий доступ до основних інструментів (включаючи інноваційні функції голосового аналізу та контекстуальних інсайтів), секцію останніх записів та щоденну пораду для підтримки емоційного здоров'я.

Структура інтерфейсу цього вікна наведена на рисунку 3.1.

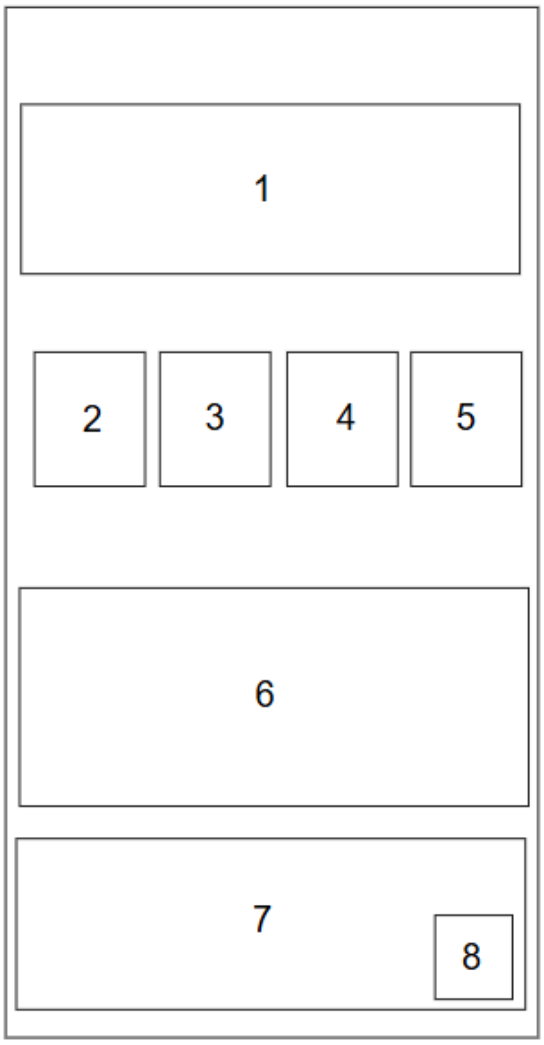


Рисунок 3.1 – Структурна схема головного вікна застосунку

Складові головного вікна застосунку:

1. Вітальне повідомлення.
2. Кнопка для нової перевірки настрою.
3. Кнопка для переходу у журнал емоцій.

4. Кнопка для переходу у вікно з інсайтами.

5. Кнопка для запису голосового повідомлення.

6. Останні записи.

7. Порада дня.

8. Кнопка додавання нового запису про емоції.

Вікно визначення емоційного стану дозволяє користувачеві вказувати емоційний стан в чотири етапи. Перший – вибір поточної емоції, другий – вибір рівня інтенсивності емоції, третій – вказування контексту емоції, четвертий – додавання коментарів.

Структура інтерфейсу вікна визначення емоційного стану наведена на рисунку 3.2.

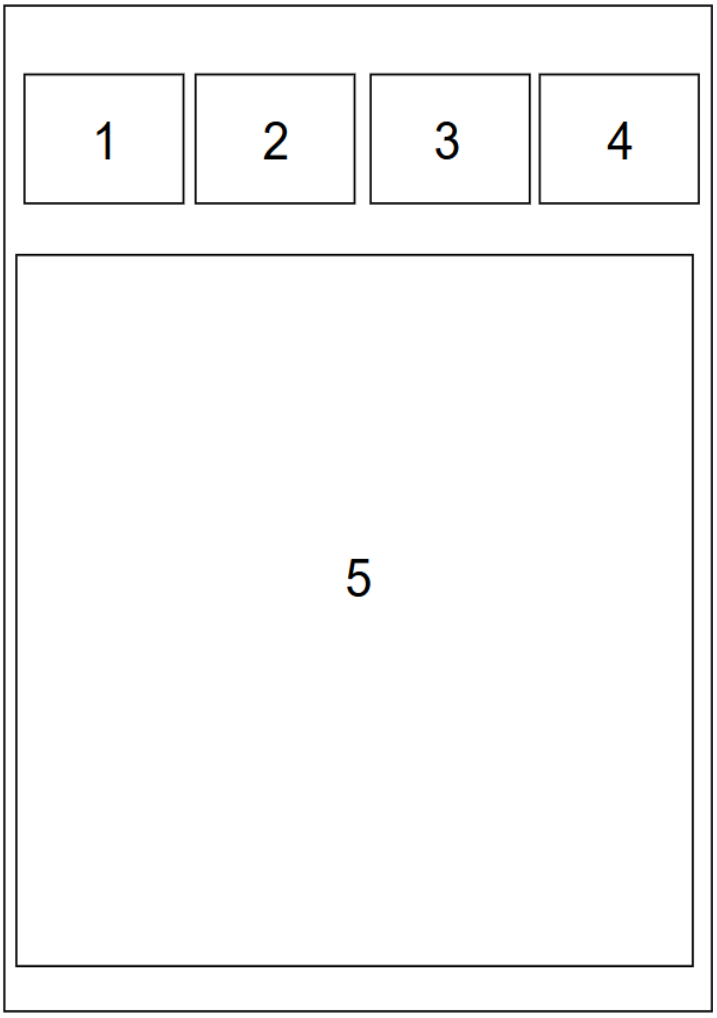


Рисунок 3.2 – Структура інтерфейсу екрану перевірки емоцій

Складові інтерфейсу:

1. Меню визначення емоції.

2. Меню визначення інтенсивності емоції.

3. Меню визначення контексту.

4. Меню запису нотаток.

5. Робоча область меню.

Запис про емоції є підсумком вказаних користувачем даних про емоції. Він відображає емоцію та коротку інформацію про неї, таку як інтенсивність емоції, короткий опис.

Структура інтерфейсу запису про емоції наведена на рисунку 3.3.

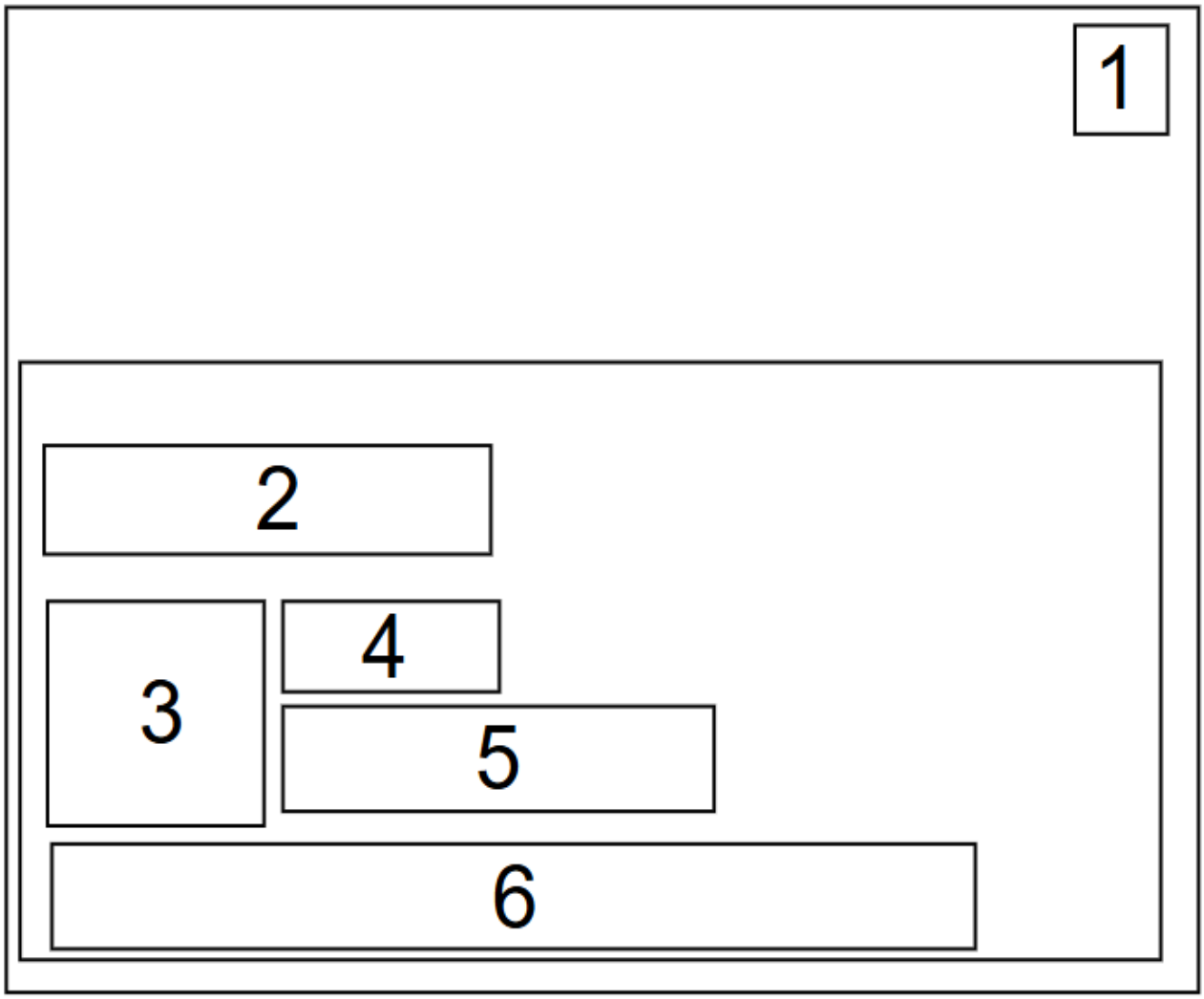


Рисунок 3.3 – Структура інтерфейсу запису про емоції

Складові елементи запису про емоції:

1. Кнопка додавання нового запису.

2. Дата та час.

3. Зображення для характеристики емоції.

4. Назва емоції.

5. Інтенсивність.

6. Опис.

Екран голосового аналізу емоцій дозволяє записувати голосове повідомлення, у якому користувач ділиться емоціями. Під час запису відорбражається анімація гучності звуку та інтонації. Структура інтерфейсу екрану голосового аналізу емоцій наведена на рисунку 3.4.

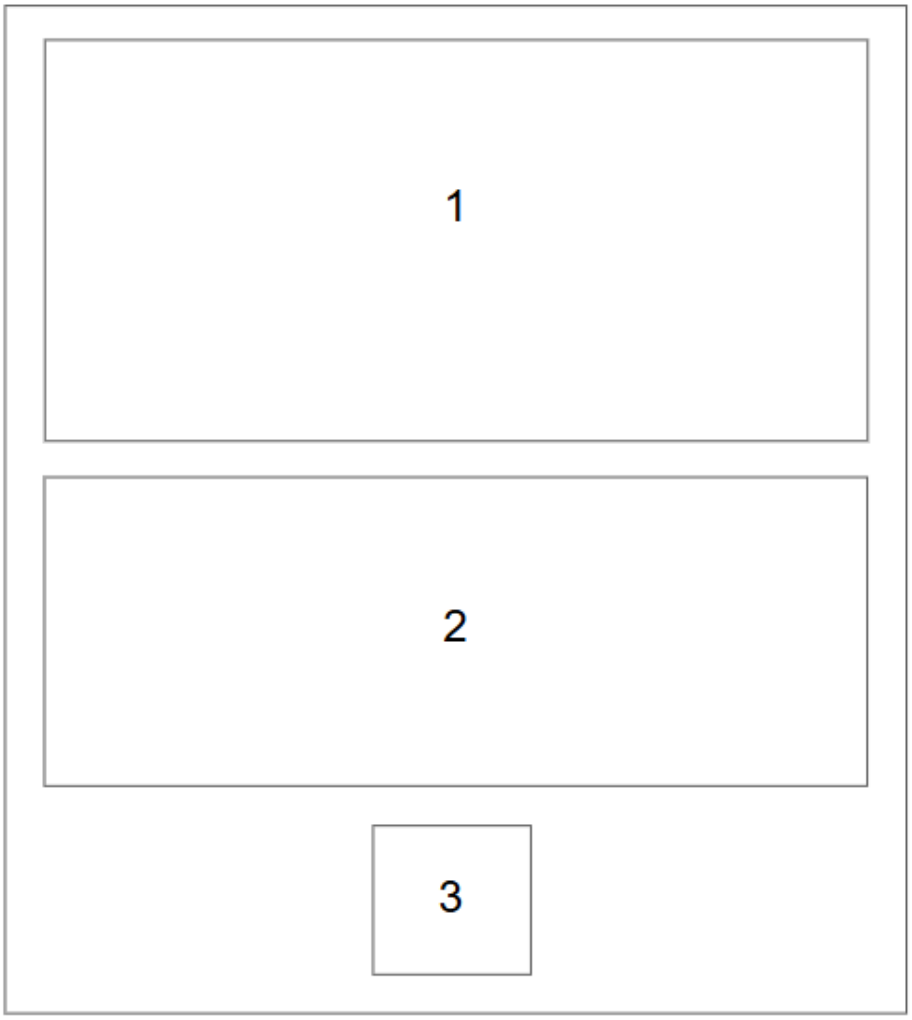


Рисунок 3.4 – Структура інтерфейсу екрану голосового аналізу емоцій

Складові елементи екрану голосового аналізу емоцій:

1.Опис роботи вікна.

2. Візуалізація голосу.

3. Кнопка початку/зупинки запису.

Після аналізу голосового повідомлення відображається запис з результатами аналізу голосу. У ньому вказується визначена емоція з ймовірністю точності визначення, опис характеристик голосу.

Структурна схема екрану аналізу голосу наведена на рисунку 3.5.

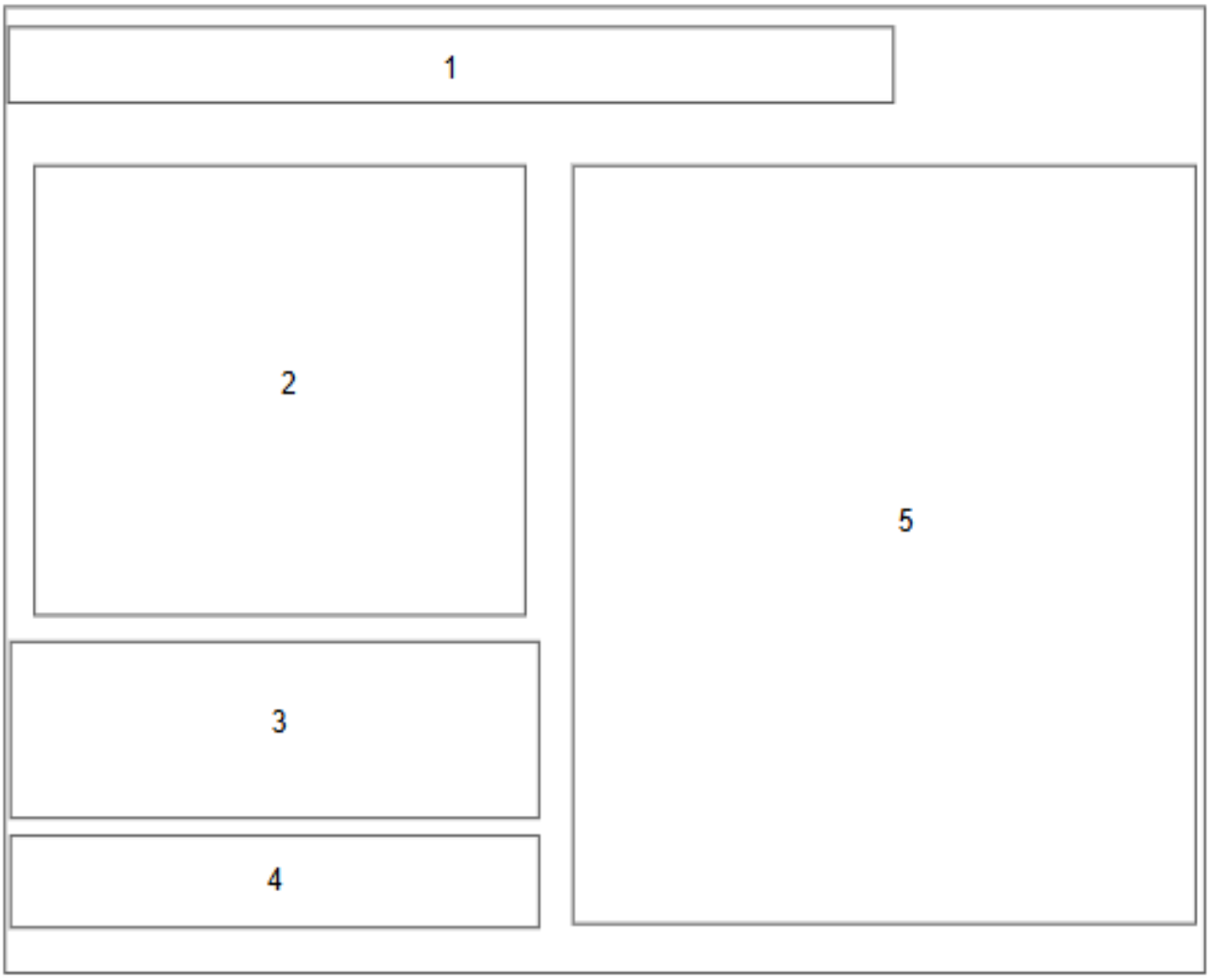


Рисунок 3.5 – Структура інтерфейсу запису з результатами аналізу голосу

Складові інтерфейсу запису з результатами аналізу голосу:

1. Заголовок запису.

2. Зображення, що відповідає за емоцію.

3. Виявлена емоція.

4. Ймовірність точності виявленої емоції.

5. Голосові характеристики.

Отже, розроблений інтерфейс користувача дозволяє додавати записи про емоції користувача, їх інтенсивність та опис, записувати голосові повідомлення з описом своїх емоцій та отримувати аналіз сказаного в повідомленні, а також аналіз голосу.

### 3.4 Висновки

У третьому розділі було проведено порівняльний аналіз мов програмування та обрано мову Kotlin для розробки застосунку  для визначення та управління емоційним станом. Розроблено базу даних застосунку та описано її складові. Розроблена база даних використовується для збереження даних користувача для їх поступової обробки. Розроблено інтерфейс користувача, який забезпечує використання застосунку для управління емоційного стану.

# 4 ТЕСТУВАННЯ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ

## 4.1 Аналіз методів тестування

У процесі забезпечення якості застосунку для визначення та управління емоційним станом користувача на основі алгоритмів штучного інтелекту доцільно розглядати кілька методів тестування.

По-перше, функціональне тестування спрямоване на перевірку коректності реалізації вимог – наприклад, чи правильно інтерпретується вхідне зображення або аудіопотік і чи відповідає емоційний статус очікуваному [22].

По-друге, нелінійне тестування моделі включає в себе валідацію якості класифікації: розрахунок метрик точності, відгуку на зібраних тестових вибірках. Це дозволяє виявити упередження у прогнозах та потребу в донавчанні або балансуванні даних.

Окрему увагу слід приділити нефункціональним аспектам. Тестування продуктивності перевіряє швидкодію системи при реальному навантаженні – наприклад, визначення емоцій у відеопотоці з частотою не менше 15 кадрів за секунду. Тестування на стійкість до збоїв і помилок – як система реагує на нечіткі дані, обриви зв’язку або некоректний формат вхідної інформації. Тестування безпеки оцінює захищеність персональних даних користувача та стійкість до можливих атак на модель (adversarial attacks), щоб виключити витік чутливої інформації або її маніпуляцію.

За рівнем участі людини в процесі тестування виділяють автоматизоване та ручне тестування. Автоматизовані тести доцільні для перевірки регресійних сценаріїв, регулярного моніторингу метрик моделі та відпрацювання великих обсягів даних [23]. Однак у застосунках, де взаємодія з користувачем включає оцінку суб’єктивних відчуттів та емоційного зворотного зв’язку, автоматизація далеко не завжди здатна відтворити тонкі нюанси поведінки реальної людини [24].

Саме тому для даного класу застосунків найбільш доцільним є метод ручного тестування. У його рамках кваліфіковані тестувальники під час виконання

сценаріїв взаємодії можуть оцінити адекватність інтерпретації емоцій, зручність інтерфейсу та психологічний комфорт користувача.

### 4.2 Тестування застосунку для управління емоційним станом людини

Тестування застосунку розпочинається з його головного екрану. Він надає доступ до усього функціоналу застосунку, включаючи записи про емоції, перевірку емоційного стану, журнал емоцій, інсайти, та аналіз голосу. Початковий стан головного екрану застосунку наведено на рисунку 4.1.


Рисунок 4.1 – Початковий стан головного екрану застосунку

Перейдемо до перевірки емоцій. Екран визначення емоцій продемонстрований на рисунку 4.2.

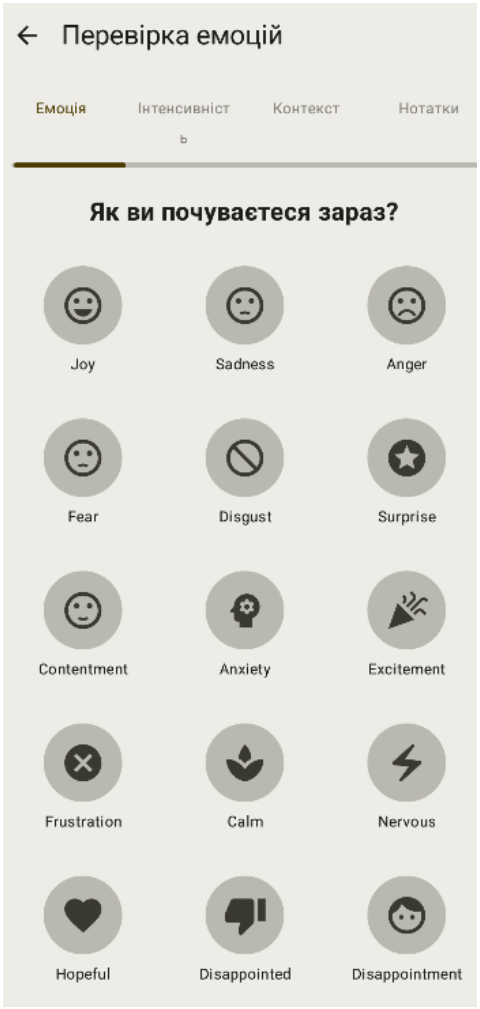


Рисунок 4.2 – Екран визначення емоцій

Оберемо емоцію Joy(радість). При виборі, обрана емоція підсвічується окремим кольором, як продемонстровано на риснуку 4.3.

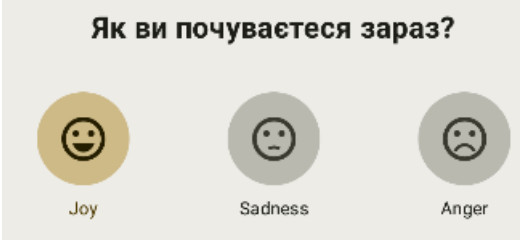


Рисунок 4.3 – Вибір емоції для визначення самопочуття

Після вибору емоції, перейдемо до наступного етапу – визначення рівня

інтенсивності емоції за шкалою від 1 до 10 (рисунок 4.4).

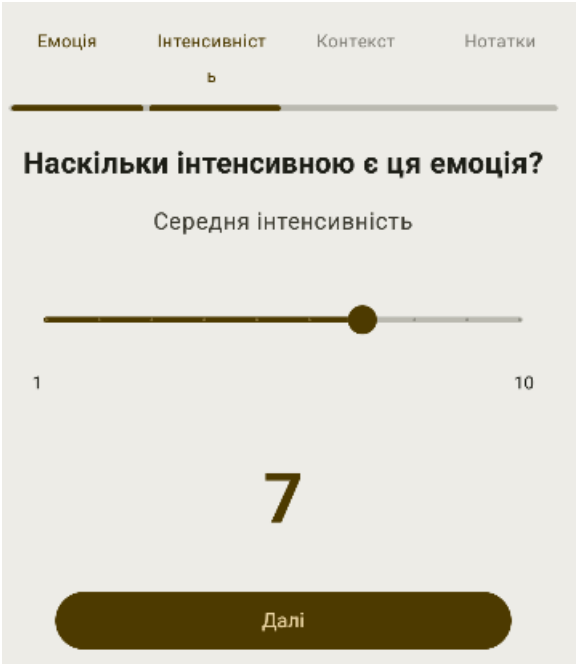


Рисунок 4.4 – Визначення рівня інтенсивності емоції

Наступним після визначення рівня інтенсивності емоції є опис того, що відбувається. Цей етап називається етапом визначення контекстуальних факторів (рисунок 4.5).

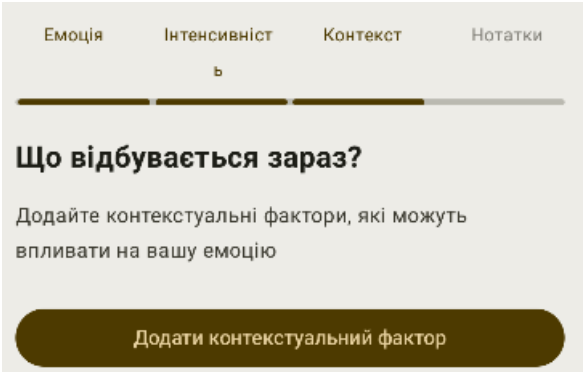


Рисунок 4.5 – Етап визначення контекстуальних факторів

Натиснемо «Додати контекстуальний фактор», після чого відкриється діалогове вікно, яке зображене на рисунку 4.6.

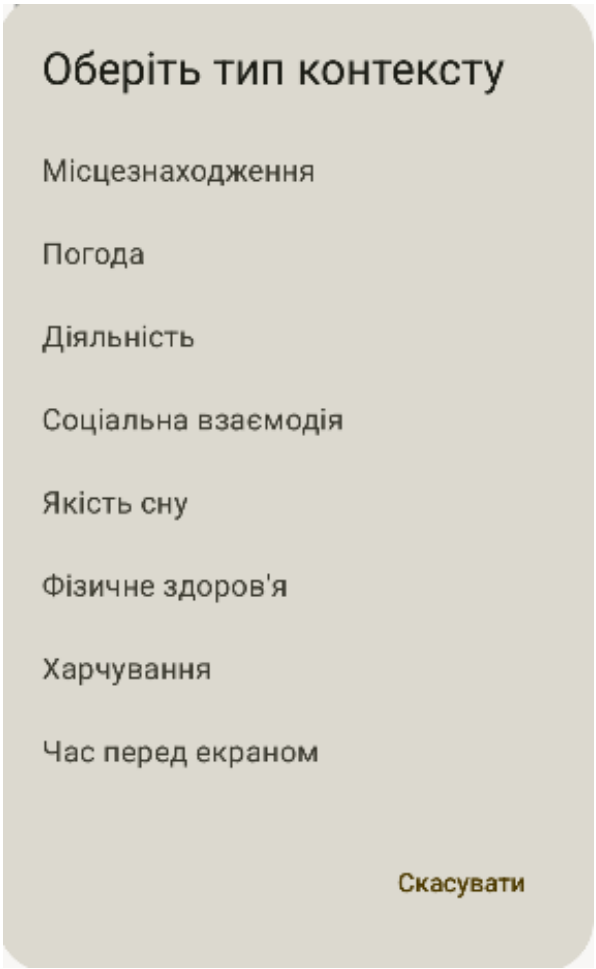


Рисунок 4.6 – Діалогове вікно вибору типу контексту

Оберемо «Якість сну». Після цього відкриється інше діалогове вікно, яке пропонує додати деталі до цього контекстуального фактору (рисунок 4.7).

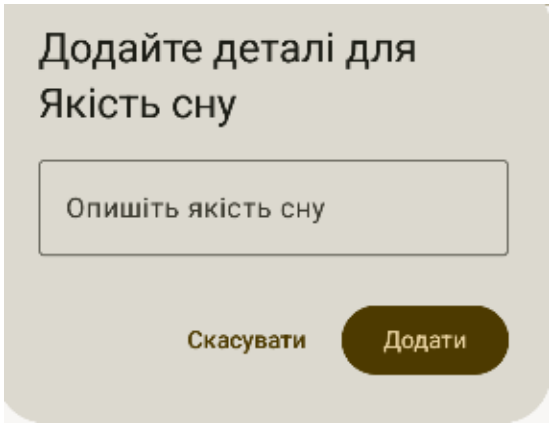


Рисунок 4.7 – Детальний опис контексту

Результат додавання контекстуального фактора наведено на рисунку 4.8.

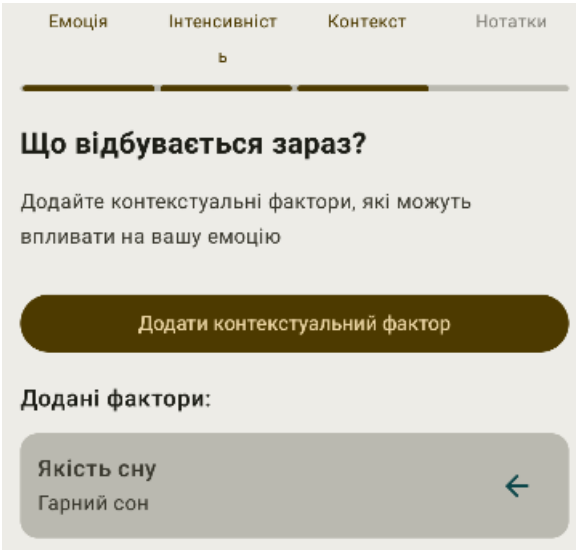


Рисунок 4.8 – Доданий фактор

Останнім етапом є додавання нотаток про емоцію, де користувач може описати усе, що забажає, що стосується емоції (рисунок 4.9).

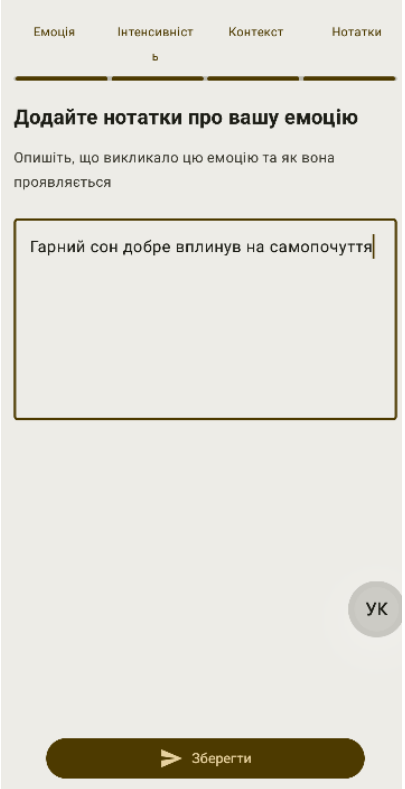


Рисунок 4.9 – Додавання нотатки про емоцію

У результаті, додається новий допис до журналу емоцій, який наведено на рисунку 4.10.

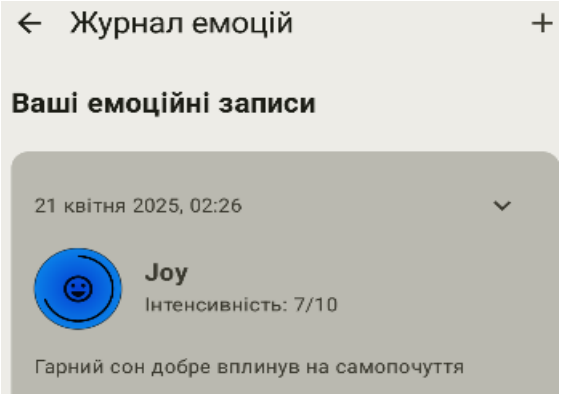


Рисунок 4.10 – Журнал емоцій

На рисунку 4.11 наведено екран контекстуальних інсайтів. Він містить візуалізацію емоцій та динаміку їх зміни між позитивними та негативними. Після додавання емоції, вона відображається у цьому екрані.

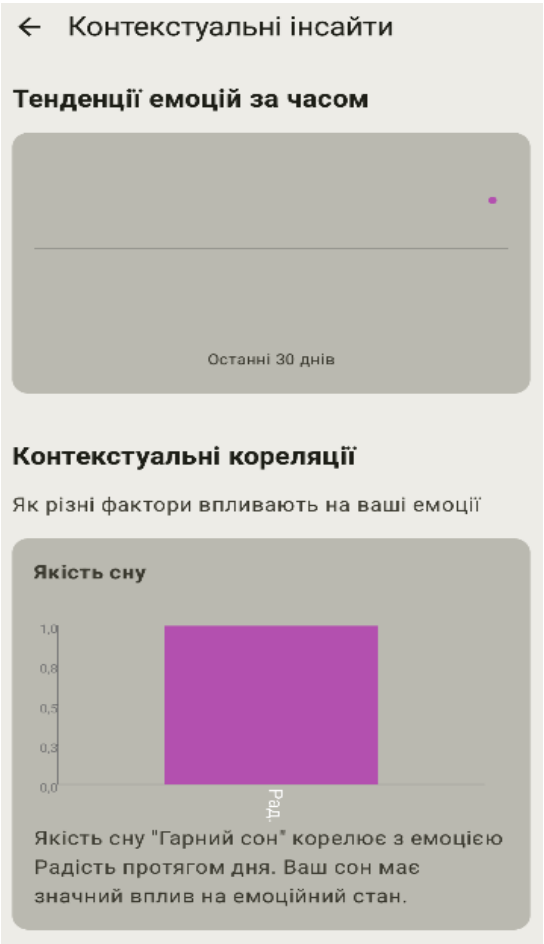


Рисунок 4.11 – Екран контекстуальних інсайтів

На основі наявних даних, відображається прогноз емоційного стану та

надаються персоналізовані рекомендації за умови, якщо даних про емоційний стан користувача достатньо (рисунок 4.12).


Рисунок 4.12 – Прогноз емоційного стану та персоналізовані рекомендації

Режим голосового аналізу емоцій дозволяє записувати голосове повідомлення та розповідати про свій емоційний стан. Система проаналізує слова, тон, темп та інтонацію голосу. Екран голосового аналізу емоцій продемонстрований на рисунку 4.13.


Рисунок 4.13 – Екран голосового аналізу емоцій

Процес запису голосового повідомлення продемонстровано на рисунку 4.14. Під час процесу відбувається візуалізація гучності та інтонації голосу.


Рисунок 4.14 – Процес запису голосового повідомлення

По завершенню запису, відбувається аналіз повідомлення, протягом якого відображається анімація, як на рисунку 4.15.

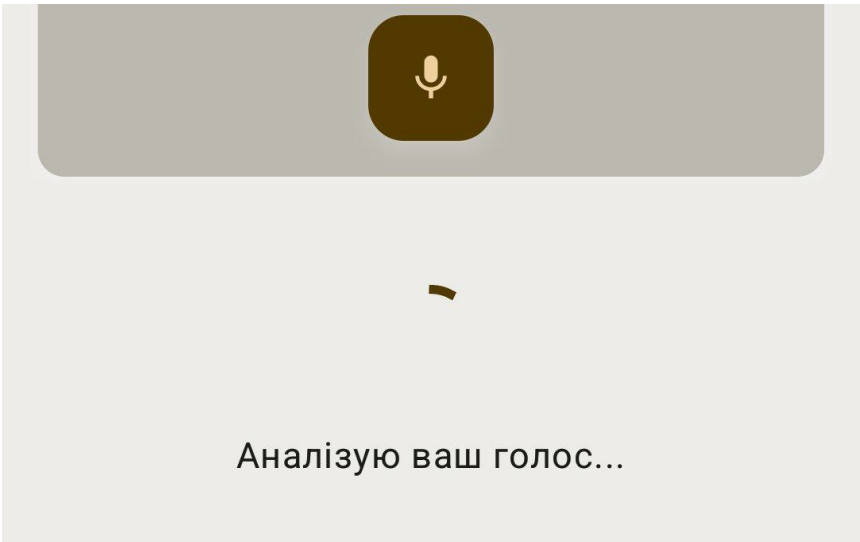


Рисунок 4.15 – Анімація при аналізі голосу

Результат аналізу наведено на рисунку 4.16. Він містить виявлену емоцію,

ймовірність точності аналізу, та характеристики голосу.


Рисунок 4.16 – Результат аналізу голосового повідомлення

Таким чином, було проведено тестування розробленого застосунку ддля визначення та управління емоційним станом, продемонстровано приклади його використання. Розроблений застосунок виконує поставлені на початку розробки завдання.

### 4.3 Висновки

У четвертому розділі, було проведено аналіз методів тестування застосунку для визначення та управління емоційним станом, обрано метод ручного тестування. Було проведено тестування розробленого застосунку, продемонстровано приклади його використання. Розроблений застосунок виконує поставлені на початку розробки завдання.

# ВИСНОВКИ

У рамках курсового проєкту було розроблено застосунок для визначення й управління емоційним станом. Розроблений застосунок дозволяє записувати власні емоції, їхню інтенсивність та контекст. Також застосунок дозволяє ділитися емоціями у голосовому режимі та отримувати аналіз не лише сказаного, а і голосу. Застосунок розроблено мовою програмування Kotlin у середовищі розробки Android Studio. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок [25].

Під час виконання бакалаврської кваліфікаційної роботи було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів, в результаті якого було доведено актуальність власної розробки. Проведено аналіз методів розв’язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження.

У першому розділі було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів: Replika, MindDoc, Tess. В результаті аналізу було доведено актуальність власної розробки. Проведено аналіз методів розв’язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження.

У другому розділі було розроблено структуру застосунку для управління емоційним станом, розроблено модель системи, алгоритми, метод аналізу голосу для визначення емоційного стану та метод контекстуального прогнозу емоційного стану.

У третьому розділі було проведено порівняльний аналіз мов програмування та обрано мову Kotlin для розробки застосунку для визначення та управління емоційним станом. Розроблено базу даних застосунку та описано її складові. Розроблена база даних використовується для збереження даних користувача для їх поступової обробки. Розроблено інтерфейс користувача, який забезпечує

використання застосунку для управління емоційного стану.

У четвертому розділі, було проведено аналіз методів тестування застосунку для визначення та управління емоційним станом, обрано метод ручного тестування. Було проведено тестування розробленого застосунку, продемонстровано приклади його використання. Розроблений застосунок виконує поставлені на початку розробки завдання.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Market Insights. (2024). Emotion AI Market Share, Trends, and Forecast 2025–2034 [Електронний ресурс] – Режим доступу до ресурсу: https://www.gminsights.com/industry-analysis/emotion-ai-market

2. World Health Organization. (2019). Burn-out an “occupational phenomenon”: International Classification of Diseases [Електронний ресурс] – Режим доступу до ресурсу: https://www.who.int/mental_health/evidence/burn-out/en/

3. Ekman, P., & Friesen, W. V. (1978). Facial Action Coding System: A technique for the measurement of facial movement. Palo Alto, CA: Consulting Psychologists Press.

4. Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Bengio, Y. (2013). Challenges in representation learning: Facial expression recognition challenge [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

5. Mollahosseini, A., Chan, D., & Mahoor, M. H. (2017). AffectNet: A database for facial expression, valence, and arousal computing in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 1–8). https://doi.org/10.1109/FG.2017.20

6. Firth, J., Torous, J., Nicholas, J., Carney, R., Pratap, A., Rosenbaum, S., & Sarris, J. (2017). The efficacy of smartphone-based mental health interventions for depression: Meta-analysis of randomized controlled trials. World Psychiatry, 16(3), 287–298. https://doi.org/10.1002/wps.20472

7. Ксенченко Я.В., Мельник О.В. РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ ТА УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ НА ОСНОВІ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ / Collection of Scientific Papers with the Proceedings of the 2nd International Scientific and Practical Conference «Achievements of Science and Applied Research» (May 19- 21, 2025. Dublin, Ireland). European Open Science Space, 2025. 288 p. С.77-80.

8. Poria, S., Cambria, E., Bajpai, R., & Hussain, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion, 37, 98–125. https://doi.org/10.1016/j.inffus.2017.02.008

9. Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. T. (2010). A survey of mobile phone sensing. IEEE Communications Magazine, 48(9), 140–150. https://doi.org/10.1109/MCOM.2010.5560598

10. General Data Protection Regulation (GDPR) (EU) 2016/679. (2016). Official Journal of the European Union [Електронний ресурс] – Режим доступу до ресурсу: https://eur-lex.europa.eu/eli/reg/2016/679/oj

11. Cummings, J., Bruni, E., & Berkowitz, B. (2016). The ethics of artificial intelligence: Issues and initiatives. Harvard University Press.

12. Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., ... & Cardoso, M. J. (2020). The future of digital health with federated learning. NPJ Digital Medicine, 3(1), 1–7. https://doi.org/10.1038/s41746-020-00323-1

13. Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., ... & Wang, Y. (2017). Artificial intelligence in healthcare: Past, present and future. Stroke and Vascular Neurology, 2(4), 230–243. https://doi.org/10.1136/svn-2017-000101

14. Poria, S., Cambria, E., & Gelbukh, A. (2016). Affective computing and sentiment analysis. IEEE Intelligent Systems, 31(2), 102–107. https://doi.org/10.1109/MIS.2016.28

15. Replika [Електронний ресурс] – Режим доступу до ресурсу: https://replika.com/

16. MindDoc [Електронний ресурс] – Режим доступу до ресурсу: https://minddoc.com/us/en

17. Tess [Електронний ресурс] – Режим доступу до ресурсу: https://www.x2ai.com/individuals

18. Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/

19. Herbert Schildt. Java: A Beginner's Guide, Tenth Edition. New York: McGraw Hill, 2024. 768c.

20. Ahmad Sahar. iOS 18 Programming for Beginners. Birmingham: Packt Publishing, 2024. 914c.

21. Методичні вказівки для самостійної роботи студентів під час виконання курсових робіт з дисципліни «Бази даних» для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад.: О. Н. Романюк, О. В. Романюк, А. В. Денисюк. – Вінниця : ВНТУ, 2022. – 71 с.

22. The different types of software testing [Електронний ресурс] – Режим доступу до ресурсу:https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

23. Аналіз методів тестування інтерфейсу користувача [Електронний ресурс]/ Д. В. Котлярчук, О. В. Романюк // Матеріали XLIX науково-технічної конференції підрозділів ВНТУ, Вінниця, 27-28 квітня 2020 р. – Електрон. текст. дані. – 2020. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9764.

24. Different Types of Testing in Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.perfecto.io/resources/types-of-testing

25. 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.

# 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.
«05» 03 2025 р.

**Технічне завдання**
**на бакалаврську кваліфікаційну роботу «Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту»**
**за спеціальністю**
**121 – Інженерія програмного забезпечення**

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Мельник О.В.
«24» 03 2025 року.

Виконав:
студент гр. 5ПІ-21б Кеенченко Я.В.
«24» 03 2025 року.

м. Вінниця – 2025 рік

**1. Найменування та галузь застосування**

Бакалаврська кваліфікаційна робота: «Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту».

Галузь застосування – мобільні технології.

**2. Підстава для розробки.**

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

**3. Мета та призначення розробки.**

Метою роботи є удосконалення процесу визначення та управління емоційним станом людини шляхом розробки спеціалізованого програмного забезпечення з використанням алгоритмів штучного інтелекту.

**4. Вихідні дані для проведення НДР**

1. Global Market Insights. (2024). Emotion AI Market Share, Trends, and Forecast 2025–2034 [Електронний ресурс] – Режим доступу до ресурсу: https://www.gminsights.com/industry-analysis/emotion-ai-market

2. Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., ... & Wang, Y. (2017). Artificial intelligence in healthcare: Past, present and future. Stroke and Vascular Neurology, 2(4), 230–243. https://doi.org/10.1136/svn-2017-000101

3. Poria, S., Cambria, E., Bajpai, R., & Hussain, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion, 37, 98–125. https://doi.org/10.1016/j.inffus.2017.02.008

4. Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/

**5. Технічні вимоги**

Вхідні дані – інформація про емоційний стан.

Вихідні дані – рекомендації, аналіз емоцій.

**6. Конструктивні вимоги.**

Інтерфейс програми поовинен відповідати естетичним та ергономічним

вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

–пояснювальна записка до бакалаврської кваліфікаційної роботи;

–технічне завдання;

–лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

**9. Стадії та етапи розробки**

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз стану розвитку застосунків для управління емоційним станом | 25.03.2025 - 06.04.2025 |
| 2 | Розробка методів та алгоритмів програмного застосунку | 07.04.2025 - 20.04.2025 |
| 3 | Варіантний аналіз та вибір мови програмування розробки | 21.04.2025 - 27.04.2025 |
| 4 | Розробка застосунку | 28.04.2025 - 18.05.2025 |
| 5 | Тестування застосунку | 19.05.2025 -25.05.2025 |
| 6 | Оформлення матеріалів до захисту БКР | 26.05.2025-30.05.2025 |

**10. Порядок контролю та прийняття.**

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

**Додаток Б – Протокол перевірки на наявність текстових запозичень**

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту»

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3,2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

□У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

□У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черповолик Г. О.

З висновком експертної комісії ознайомлений

Керівник _____ Мельник О.В. к.т.н., доцент каф.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ксенченко Я.В
(підпис) (прізвище, ініціали)

# Додаток В – Лістинг програми

```
@OptIn(ExperimentalPermissionsApi::class)
@Composable
fun VoiceAnalysisScreen(
    navController: NavController ,
    viewModel: VoiceAnalysisViewModel = hiltViewModel()
) {
    val uiState by viewModel.uiState.collectAsState()
    val context = LocalContext.current
    val micPermissionState = rememberPermissionState(Manifest.permission.RECORD_AUDIO)

    val storagePermissionState = if (Build.VERSION.SDK_INT <= Build.VERSION_CODES.S_V2) {
        rememberPermissionState(Manifest.permission.WRITE_EXTERNAL_STORAGE)
    } else null

    LaunchedEffect(Unit) {
        if (!micPermissionState.status.isGranted) {
            micPermissionState.launchPermissionRequest()
        }

        storagePermissionState?.let {
            if (!it.status.isGranted) {
                it.launchPermissionRequest()
            }
        }
    }

    val requestPermissionLauncher = rememberLauncherForActivityResult(
        ActivityResultContracts.RequestPermission()
    ) { isGranted: Boolean ->
        if (isGranted) {
            // Якщо дозвіл надано, починаємо запис
            viewModel.startRecording()
        } else {
            // Якщо дозвіл не надано, показуємо повідомлення
            Toast.makeText(
                context,
                "Для запису голосу потрібен дозвіл на використання мікрофона",
                Toast.LENGTH_SHORT
            ).show()
        }
    }

    val hasRecordPermission = ContextCompat.checkSelfPermission(
        context,
        Manifest.permission.RECORD_AUDIO
    ) == PackageManager.PERMISSION_GRANTED

    val arePermissionsGranted = micPermissionState.status.isGranted &&
            (storagePermissionState == null || storagePermissionState.status.isGranted)
```

```
Column(
    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp),
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Text(
        text = "Голосовий аналіз емоцій",
        fontSize = 22.sp,
        fontWeight = FontWeight.Bold,
        modifier = Modifier.padding(bottom = 24.dp)
    )

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(bottom = 24.dp),
        elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Column(
            modifier = Modifier.padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Text(
                text = "Розкажіть про свої почуття",
                fontWeight = FontWeight.Medium,
                fontSize = 18.sp,
                modifier = Modifier.padding(bottom = 8.dp)
            )

            Text(
                text = "Запишіть голосове повідомлення, в якому ви розповідаєте про свій емоційний стан. " +
                        "AI проаналізує не тільки ваші слова, але й тон, темп та інтонацію голосу.",
                modifier = Modifier.padding(bottom = 16.dp)
            )

            if (uiState.isRecording) {
                VoiceWaveform(
                    amplitudes = uiState.audioAmplitudes,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(120.dp)
                        .clip(RoundedCornerShape(12.dp))
                        .background(MaterialTheme.colorScheme.primaryContainer)
                        .padding(8.dp)
                )
            } else {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(120.dp)
```

```
            .clip(RoundedCornerShape(12.dp))
            .background(MaterialTheme.colorScheme.surfaceVariant)
            .padding(8.dp),
        contentAlignment = Alignment.Center
    ) {
        Text("Натисніть кнопку мікрофона, щоб розпочати запис")
    }
}

Spacer(modifier = Modifier.height(16.dp))

FloatingActionButton(
    onClick = {
        if (hasRecordPermission) {
            if (uiState.isRecording) {
                viewModel.stopRecording()
            } else {
                viewModel.startRecording()
            }
        } else {
            // Запускаємо запит дозволу
            requestPermissionLauncher.launch(Manifest.permission.RECORD_AUDIO)
        }
    },
    containerColor = if (uiState.isRecording) Color.Red else MaterialTheme.colorScheme.primary
) {
    Icon(
        imageVector = if (uiState.isRecording) Icons.Default.Stop else Icons.Default.Mic,
        contentDescription = if (uiState.isRecording) "Зупинити запис" else "Почати запис"
    )
}

// Додаємо пояснення, якщо дозвіл не надано
if (!hasRecordPermission) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp),
        colors = CardDefaults.cardColors(
            containerColor = MaterialTheme.colorScheme.errorContainer
        )
    ) {
        Column(
            modifier = Modifier.padding(16.dp)
        ) {
            Text(
                text = "Потрібен дозвіл на використання мікрофона",
                fontWeight = FontWeight.Bold,
                color = MaterialTheme.colorScheme.onErrorContainer
            )

            Text(
                text = "Для запису голосу потрібен дозвіл на використання мікрофона. Натисніть на кнопку мікрофона, щоб надати
```

```
дозвіл.",
                        color = MaterialTheme.colorScheme.onErrorContainer,
                        modifier = Modifier.padding(top = 8.dp)
                    )

                    Button(
                        onClick = { requestPermissionLauncher.launch(Manifest.permission.RECORD_AUDIO) },
                        modifier = Modifier
                            .align(Alignment.End)
                            .padding(top = 8.dp)
                    ) {
                        Text("Надати дозвіл")
                    }
                }
            }
        }
    }
}

// Результати аналізу
if (uiState.isAnalyzing) {
    CircularProgressIndicator(
        modifier = Modifier.padding(24.dp)
    )
    Text("Аналізую ваш голос...")
}

uiState.audioSentiment?.let { sentiment ->
    Card(
        modifier = Modifier.fillMaxWidth(),
        elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Column(
            modifier = Modifier.padding(16.dp)
        ) {
            Text(
                text = "Результати аналізу голосу",
                fontWeight = FontWeight.Bold,
                fontSize = 18.sp,
                modifier = Modifier.padding(bottom = 16.dp)
            )

            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.SpaceBetween
            ) {
                Column(modifier = Modifier.weight(1f)) {
                    EmotionVisualizer(
                        emotion = sentiment.dominantEmotion,
                        confidence = sentiment.emotionConfidence,
                        modifier = Modifier
                            .size(100.dp)
                            .align(Alignment.CenterHorizontally)
```

```
            )

            Text(
                text = "Виявлена емоція: ${sentiment.dominantEmotion.name.lowercase().replaceFirstChar { it.uppercase() }}",
                fontWeight = FontWeight.Medium,
                modifier = Modifier.padding(top = 8.dp, bottom = 4.dp)
            )

            Text(
                text = "Впевненість: ${(sentiment.emotionConfidence * 100).toInt()}%",
                color = MaterialTheme.colorScheme.onSurfaceVariant
            )
        }

        Divider(
            modifier = Modifier
                .width(1.dp)
                .height(120.dp)
                .padding(horizontal = 16.dp),
            color = MaterialTheme.colorScheme.outlineVariant
        )

        Column(modifier = Modifier.weight(1f)) {
            Text(
                text = "Голосові характеристики:",
                fontWeight = FontWeight.Medium,
                modifier = Modifier.padding(bottom = 8.dp)
            )

            val voiceFeatures = listOf(
                "Темп" to formatFeatureValue(sentiment.tempo),
                "Висота" to formatFeatureValue(sentiment.pitch),
                "Варіації гучності" to formatFeatureValue(sentiment.volumeVariation)
            )

            voiceFeatures.forEach { (name, value) ->
                Row(
                    modifier = Modifier
                        .fillMaxWidth()
                        .padding(vertical = 4.dp),
                    horizontalArrangement = Arrangement.SpaceBetween
                ) {
                    Text(text = name)
                    Text(
                        text = value,
                        fontWeight = FontWeight.Medium
                    )
                }
            }
        }
    }

    uiState.copingStrategy?.let { strategy ->
```

```
                Divider(modifier = Modifier.padding(vertical = 16.dp))

                Text(
                    text = "Персоналізована порада:",
                    fontWeight = FontWeight.Medium,
                    modifier = Modifier.padding(bottom = 8.dp)
                )

                Text(text = strategy)
            }

            Button(
                onClick = { viewModel.saveAnalysisToJournal() },
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(top = 16.dp)
            ) {
                Text("Зберегти до журналу")
            }
        }
    }

    if (uiState.error != null) {
        Text(
            text = uiState.error!! ,
            color = MaterialTheme.colorScheme.error,
            modifier = Modifier.padding(top = 16.dp)
        )
    }
  }
}

private fun formatFeatureValue(value: Float): String {
  return when {
    value < 0.4f -> "Низький"
    value < 0.7f -> "Середній"
    else -> "Високий"
  }
}

@HiltViewModel
class VoiceAnalysisViewModel @Inject constructor(
  @ApplicationContext private val context: Context ,
  private val repository: EmotionRepository ,
  private val claudeService: ClaudeService
) : ViewModel() {

  private val _uiState = MutableStateFlow(VoiceAnalysisUiState())
  val uiState: StateFlow<VoiceAnalysisUiState> = _uiState.asStateFlow()

  private var mediaRecorder: MediaRecorder? = null
  private var audioFilePath: String? = null
```

```
private var recordingStartTime: Long = 0L


private fun normalizeAmplitude(amplitude: Int): Float {
    // Нормалізація амплітуди для візуалізації (0.0 - 1.0)
    val amplitudeDb = if (amplitude > 0) 20 * Math.log10(amplitude.toDouble()) else 0.0
    // Припущення, що мінімальний рівень гучності ~ 0 дБ, максимальний ~ 90 дБ
    return (max(0.0, min(amplitudeDb, 90.0)) / 90.0).toFloat()
}

private fun analyzeVoiceRecording() {
    viewModelScope.launch {
        _uiState.update { it.copy(isAnalyzing = true) }

        try {
            // Тут був би справжній аналіз аудіо файлу для отримання характеристик голосу
            // Для демонстрації використовуємо симульовані дані

            val audioFeatures = simulateAudioFeatureExtraction()

            // Використання Claude API для аналізу голосових характеристик
            // Використання Claude API для аналізу голосових характеристик
            val sentiment = claudeService.analyzeVoiceEmotions(audioFeatures)

            // Отримання стратегії подолання на основі виявленої емоції
            val copingStrategy = claudeService.generateContextualCopingStrategy(
                sentiment.dominantEmotion,
                (sentiment.emotionConfidence * 10).toInt(),
                listOf(ContextualFactor(ContextType.ACTIVITY, "Голосовий щоденник"))
            )

            _uiState.update { it.copy(
                audioSentiment = sentiment,
                copingStrategy = copingStrategy,
                isAnalyzing = false
            ) }

        } catch (e: Exception) {
            _uiState.update { it.copy(
                isAnalyzing = false,
                error = "Помилка аналізу: ${e.message}"
            ) }
        }
    }
}

private fun simulateAudioFeatureExtraction(): Map<String, Float> {
    // Симуляція вилучення характеристик з аудіо запису
    // В реальному застосунку тут був би повноцінний аналіз аудіо-файлу
    return mapOf(
        "tempo" to (0.4f + Math.random() * 0.6f).toFloat(),
        "pitchAvg" to (0.3f + Math.random() * 0.7f).toFloat(),
        "pitchVar" to (0.2f + Math.random() * 0.8f).toFloat(),
```

```
        "volumeVar" to (0.3f + Math.random() * 0.7f).toFloat(),
        "speechRate" to (0.4f + Math.random() * 0.6f).toFloat(),
        "pausesFreq" to (0.3f + Math.random() * 0.7f).toFloat()
    )
}

fun startRecording() {
    viewModelScope.launch {
        try {
            val file = withContext(Dispatchers.IO) {
                val cacheDir = context.cacheDir
                File(cacheDir, "audio_${System.currentTimeMillis()}.mp3").apply {
                    createNewFile()
                }
            }

            audioFilePath = file.absolutePath

            withContext(Dispatchers.IO) {
                try {
                    // Використовуйте правильну ініціалізацію залежно від версії Android
                    mediaRecorder = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
                        MediaRecorder(context)
                    } else {
                        @Suppress("DEPRECATION")
                        MediaRecorder()
                    }

                    mediaRecorder?.apply {
                        // Правильний порядок виклику методів:
                        setAudioSource(MediaRecorder.AudioSource.MIC)
                        setOutputFormat(MediaRecorder.OutputFormat.MPEG_4)
                        setAudioEncoder(MediaRecorder.AudioEncoder.AAC)
                        setOutputFile(audioFilePath)

                        try {
                            prepare()
                            start()
                            android.util.Log.d("VoiceAnalysis", "Запис розпочато успішно")
                        } catch (e: Exception) {
                            android.util.Log.e("VoiceAnalysis", "Помилка при запуску запису: ${e.message}", e)
                            throw e
                        }
                    }
                } catch (e: Exception) {
                    android.util.Log.e("VoiceAnalysis", "Помилка при ініціалізації MediaRecorder: ${e.message}", e)
                    throw e
                }
            }

            recordingStartTime = System.currentTimeMillis()
            _uiState.update { it.copy(
                isRecording = true,
```

```
                recordingDuration = 0,
                audioAmplitudes = emptyList(),
                error = null
            ) }

            // Моніторинг амплітуди для візуалізації
            while (uiState.value.isRecording) {
                val amplitude = try {
                    mediaRecorder?.maxAmplitude ?: 0
                } catch (e: Exception) {
                    0
                }
                val normalizedAmplitude = normalizeAmplitude(amplitude)

                val newAmplitudes = uiState.value.audioAmplitudes.toMutableList()
                if (newAmplitudes.size >= 50) {
                    newAmplitudes.removeAt(0)
                }
                newAmplitudes.add(normalizedAmplitude)

                val duration = (System.currentTimeMillis() - recordingStartTime) / 1000

                _uiState.update { it.copy(
                    audioAmplitudes = newAmplitudes,
                    recordingDuration = duration.toInt()
                ) }

                delay(100) // Оновлення 10 разів на секунду
            }

        } catch (e: Exception) {
            android.util.Log.e("VoiceAnalysis", "Помилка запису: ${e.message}", e)
            _uiState.update { it.copy(
                isRecording = false,
                error = "Помилка запису: ${e.message}"
            ) }

            // Коректне вивільнення ресурсів у випадку помилки
            mediaRecorder?.release()
            mediaRecorder = null
        }
    }
}

fun stopRecording() {
    viewModelScope.launch {
        try {
            withContext(Dispatchers.IO) {
                try {
                    mediaRecorder?.apply {
                        stop()
                        release()
                    }
```

```
            } catch (e: Exception) {
                android.util.Log.e("VoiceAnalysis", "Помилка при зупинці запису: ${e.message}", e)
            }
            mediaRecorder = null
        }

        _uiState.update { it.copy(isRecording = false) }

        analyzeVoiceRecording()

    } catch (e: Exception) {
        _uiState.update { it.copy(
            error = "Помилка при зупинці запису: ${e.message}"
        ) }
    }
  }
}

fun saveAnalysisToJournal() {
  viewModelScope.launch {
    uiState.value.audioSentiment?.let { sentiment ->
        val entry = EmotionEntry(
            id = UUID.randomUUID().toString(),
            timestamp = LocalDateTime.now(),
            primaryEmotion = sentiment.dominantEmotion,
            intensity = (sentiment.emotionConfidence * 10).toInt(),
            notes = "Запис голосу: виявлено ${sentiment.dominantEmotion.name.lowercase()}",
            contextualFactors = listOf(ContextualFactor(ContextType.ACTIVITY, "Голосовий щоденник")),
            audioRecordingPath = audioFilePath,
            audioSentiment = sentiment
        )

        repository.saveEmotionEntry(entry)

        _uiState.update { it.copy(
            saved = true,
            audioSentiment = null,
            copingStrategy = null,
            audioAmplitudes = emptyList()
        ) }
    }
  }
}

override fun onCleared() {
  super.onCleared()
  try {
    mediaRecorder?.apply {
        if (uiState.value.isRecording) {
            stop()
        }
        release()
    }
```

```
        } catch (e: Exception) {
            android.util.Log.e("VoiceAnalysis", "Помилка при вивільненні ресурсів: ${e.message}", e)
        }
        mediaRecorder = null
    }
}

data class VoiceAnalysisUiState(
    val isRecording: Boolean = false,
    val isAnalyzing: Boolean = false,
    val recordingDuration: Int = 0, // в секундах
    val audioAmplitudes: List<Float> = emptyList(),
    val audioSentiment: AudioSentiment? = null,
    val copingStrategy: String? = null,
    val saved: Boolean = false,
    val error: String? = null
)

package com.example.emobalance2

import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.vector.ImageVector
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.navigation.NavController
import java.time.format.DateTimeFormatter

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun HomeScreen(
    navController: NavController,
    viewModel: HomeViewModel = hiltViewModel()
) {
```

```
val uiState by viewModel.uiState.collectAsState()

LaunchedEffect(Unit) {
    viewModel.loadData()
}

Scaffold(
    topBar = {
        TopAppBar(
            title = { Text("Емоційне благополуччя") },
            actions = {
                IconButton(onClick = { navController.navigate("settings") }) {
                    Icon(Icons.Default.Settings, contentDescription = "Налаштування")
                }
            }
        )
    },
    floatingActionButton = {
        FloatingActionButton(
            onClick = { navController.navigate("emotion_check") },
            containerColor = MaterialTheme.colorScheme.primary
        ) {
            Icon(
                imageVector = Icons.Default.Add,
                contentDescription = "Додати запис про емоції"
            )
        }
    }
) { paddingValues ->
    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(paddingValues)
            .padding(horizontal = 16.dp),
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        // Вітання з користувачем
        item {
            WelcomeSection(uiState.userName)
        }

        // Сьогоднішній настрій
        item {
            TodaysMoodSection(
                currentMood = uiState.currentMood,
                onCheckEmotionClick = { navController.navigate("emotion_check") }
            )
        }

        // Швидкі дії
        item {
            QuickActionsSection(
                onEmotionCheckClick = { navController.navigate("emotion_check") },
```

```
                onJournalClick = { navController.navigate("emotion_journal") },
                onInsightsClick = { navController.navigate("contextual_insights") },
                onVoiceAnalysisClick = { navController.navigate("voice_analysis") }
            )
        }

        // Останні записи
        item {
            Text(
                text = "Останні записи",
                fontSize = 18.sp,
                fontWeight = FontWeight.Bold,
                modifier = Modifier.padding(vertical = 8.dp)
            )
        }

        if (uiState.recentEntries.isEmpty()) {
            item {
                EmptyEntriesPlaceholder(
                    onAddEntryClick = { navController.navigate("emotion_check") }
                )
            }
        } else {
            items(uiState.recentEntries) { entry ->
                EmotionEntryCard(
                    entry = entry,
                    onClick = { navController.navigate("emotion_journal") }
                )
            }
        }

        // Персоналізовані поради
        if (uiState.dailyTip != null) {
            item {
                DailyTipCard(tip = uiState.dailyTip!!)
            }
        }

        // Додаткове місце внизу для FAB
        // Додаткове місце внизу для FAB
        item {
            Spacer(modifier = Modifier.height(80.dp))
        }
    }
  }
}

@Composable
fun WelcomeSection(userName: String) {
  Column(
    modifier = Modifier.padding(vertical = 16.dp)
  ) {
    Text(
```

```
            text = "Привіт, $userName",
            fontSize = 24.sp,
            fontWeight = FontWeight.Bold
        )

        Text(
            text = "Як ви почуваєтеся сьогодні?",
            fontSize = 16.sp,
            color = MaterialTheme.colorScheme.onSurfaceVariant,
            modifier = Modifier.padding(top = 4.dp)
        )
    }
}

@Composable
fun TodaysMoodSection(
    currentMood: EmotionEntry?,
    onCheckEmotionClick: () -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp),
        elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        if (currentMood != null) {
            Row(
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(16.dp),
                verticalAlignment = Alignment.CenterVertically
            ) {
                EmotionVisualizer(
                    emotion = currentMood.primaryEmotion,
                    confidence = currentMood.intensity / 10f,
                    modifier = Modifier.size(60.dp)
                )

                Column(
                    modifier = Modifier
                        .weight(1f)
                        .padding(start = 16.dp)
                ) {
                    Text(
                        text = "Сьогодні ви відчуваєте:",
                        fontSize = 14.sp,
                        color = MaterialTheme.colorScheme.onSurfaceVariant
                    )

                    Text(
                        text = currentMood.primaryEmotion.name.lowercase().replaceFirstChar { it.uppercase() },
                        fontSize = 20.sp,
                        fontWeight = FontWeight.Bold,
```

```
                    modifier = Modifier.padding(vertical = 4.dp)
                )

                LinearProgressIndicator(
                    progress = currentMood.intensity / 10f,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(8.dp)
                        .clip(RoundedCornerShape(4.dp))
                )

                Text(
                    text = "Інтенсивність: ${currentMood.intensity}/10",
                    fontSize = 12.sp,
                    modifier = Modifier.padding(top = 4.dp)
                )
            }
        }
    } else {
        Column(
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Text(
                text = "Ви ще не відзначили свій настрій сьогодні",
                textAlign = TextAlign.Center,
                modifier = Modifier.padding(vertical = 8.dp)
            )

            Button(onClick = onCheckEmotionClick) {
                Text(text = "Перевірте свої емоції")
            }
        }
    }
}

@Composable
fun QuickActionsSection(
    onEmotionCheckClick: () -> Unit,
    onJournalClick: () -> Unit,
    onInsightsClick: () -> Unit,
    onVoiceAnalysisClick: () -> Unit
) {
    Column(
        modifier = Modifier.padding(vertical = 8.dp)
    ) {
        Text(
            text = "Швидкі дії",
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold,
```

```
            modifier = Modifier.padding(bottom = 8.dp)
        )

        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            QuickActionItem(
                icon = Icons.Default.Add,
                label = "Нова перевірка",
                onClick = onEmotionCheckClick,
                modifier = Modifier.weight(1f)
            )

            QuickActionItem(
                icon = Icons.Default.Book,
                label = "Журнал",
                onClick = onJournalClick,
                modifier = Modifier.weight(1f)
            )

            QuickActionItem(
                icon = Icons.Default.Analytics,
                label = "Інсайти",
                onClick = onInsightsClick,
                modifier = Modifier.weight(1f)
            )

            QuickActionItem(
                icon = Icons.Default.Mic,
                label = "Голос",
                onClick = onVoiceAnalysisClick,
                modifier = Modifier.weight(1f)
            )
        }
    }
}

@Composable
fun QuickActionItem(
    icon: ImageVector,
    label: String,
    onClick: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        modifier = modifier
            .padding(horizontal = 4.dp)
            .clickable(onClick = onClick),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Box(
            modifier = Modifier
```

```
                .size(48.dp)
                .clip(CircleShape)
                .background(MaterialTheme.colorScheme.primaryContainer),
            contentAlignment = Alignment.Center
        ) {
            Icon(
                imageVector = icon,
                contentDescription = label,
                tint = MaterialTheme.colorScheme.onPrimaryContainer
            )
        }

        Text(
            text = label,
            fontSize = 12.sp,
            textAlign = TextAlign.Center,
            modifier = Modifier.padding(top = 4.dp)
        )
    }
}

@Composable
fun EmotionEntryCard(
    entry: EmotionEntry,
    onClick: () -> Unit
) {
    val formatter = DateTimeFormatter.ofPattern("dd MMM, HH:mm")

    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 4.dp)
            .clickable(onClick = onClick),
        elevation = CardDefaults.cardElevation(defaultElevation = 2.dp)
    ) {
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(12.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            EmotionVisualizer(
                emotion = entry.primaryEmotion,
                confidence = entry.intensity / 10f,
                modifier = Modifier.size(40.dp)
            )

            Column(
                modifier = Modifier
                    .weight(1f)
                    .padding(start = 16.dp)
            ) {
                Text(
```

```
                text = entry.primaryEmotion.name.lowercase().replaceFirstChar { it.uppercase() },
                fontWeight = FontWeight.Medium,
                fontSize = 16.sp
            )

            Text(
                text = entry.timestamp.format(formatter),
                fontSize = 12.sp,
                color = MaterialTheme.colorScheme.onSurfaceVariant
            )

            if (entry.notes.isNotBlank()) {
                Text(
                    text = entry.notes.take(50) + if (entry.notes.length > 50) "..." else "",
                    fontSize = 14.sp,
                    modifier = Modifier.padding(top = 4.dp)
                )
            }
        }
    }
}

@Composable
fun EmptyEntriesPlaceholder(
    onAddEntryClick: () -> Unit
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .height(160.dp)
            .clip(RoundedCornerShape(12.dp))
            .background(MaterialTheme.colorScheme.surfaceVariant),
        contentAlignment = Alignment.Center
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Icon(
                imageVector = Icons.Default.AddCircle,
                contentDescription = "Додати запис",
                tint = MaterialTheme.colorScheme.onSurfaceVariant.copy(alpha = 0.7f),
                modifier = Modifier.size(48.dp)
            )

            Text(
                text = "Немає записів про емоції",
                fontSize = 16.sp,
                color = MaterialTheme.colorScheme.onSurfaceVariant,
                modifier = Modifier.padding(vertical = 8.dp)
            )

            Button(onClick = onAddEntryClick) {
```

```
            Text(text = "Створити перший запис")
        }
    }
  }
}

@Composable
fun DailyTipCard(tip: String) {
  Card(
    modifier = Modifier
      .fillMaxWidth()
      .padding(vertical = 8.dp),
    colors = CardDefaults.cardColors(
      containerColor = MaterialTheme.colorScheme.secondaryContainer
    )
  ) {
    Column(
      modifier = Modifier.padding(16.dp)
    ) {
      Text(
        text = "Порада дня",
        fontWeight = FontWeight.Bold,
        fontSize = 16.sp,
        color = MaterialTheme.colorScheme.onSecondaryContainer
      )

      Text(
        text = tip,
        fontSize = 14.sp,
        color = MaterialTheme.colorScheme.onSecondaryContainer,
        modifier = Modifier.padding(top = 8.dp)
      )
    }
  }
}
```

**Додаток Г – Ілюстративна частина**

# ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСТОСУНОК ДЛЯ ВИЗНАЧЕННЯ ТА УПРАВЛІННЯ ЕМОЦІЙНИМ СТАНОМ ЛЮДИНИ НА ОСНОВІ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ

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

Бакалаврська кваліфікаційна робота на тему:
«Застосунок для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту»

Автор: ст.групи 5ПІ-21б Ксенченко Я.В.
Науковий керівник: к.т.н., доцент каф. ПЗ Мельник О.В.

Вінниця - 2025

Рисунок Г.1 — Титульний слайд

# Актуальність теми

Розробка застосунку для визначення та управління емоційним станом людини є актуальною відповіддю на зростаючі труднощі сучасного суспільства, пов'язані зі стресом, вигоранням та загальною нестабільністю психоемоційного фону. Завдяки поєднанню методів комп'ютерного аналізу зображень, обробки сигналів біометричних сенсорів та алгоритмів штучного інтелекту, сучасні мобільні та десктопні рішення можуть своєчасно виявляти зміни у виразі обличчя, тоні голосу, пульсовій активності та іншим фізіологічним показникам користувача. Це дозволяє не лише діагностувати емоційні стани в реальному часі, а й автоматично адаптувати інтерфейс або надати рекомендації для нормалізації психологічного балансу.

Наступним важливим аспектом є інтеграція виявлення емоцій з механізмами самоконтролю та психологічної підтримки. Сучасні застосунки повинні забезпечувати користувача інструментами когнітивно-поведінкової терапії, дихальними вправами, контекстними підказками та відстеженням прогресу в довгостроковій перспективі. Завдяки аналітиці даних про настрої та реакції, система може генерувати персоналізовані плани роботи з емоціями, спрямовані на підвищення рівня стресостійкості та гармонізацію психічного стану.

Головною перевагою такого програмного рішення є можливість гнучкого налаштування під індивідуальні особливості користувача: рівень чутливості алгоритмів, бажаний рівень деталізації звітів та інтеграція з носимими пристроями чи платформами дистанційної психологічної підтримки. Таким чином, застосунок стає не лише інструментом для діагностики емоцій, а й комплексною системою управління власним психологічним здоров'ям, здатною оперативно реагувати на зміни в емоційному стані та сприяти формуванню здорових звичок саморегуляції.

Рисунок Г.2 — Актуальність теми

# Мета, об’єкт та предмет дослідження

Метою роботи є удосконалення процесу визначення та управління емоційним станом людини шляхом розробки спеціалізованого програмного забезпечення з використанням алгоритмів штучного інтелекту.
Об’єкт дослідження – процеси розробки застосунку для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту.
Предмет дослідження – методи і засоби реалізації застосунку для визначення та управління емоційним станом людини на основі алгоритмів штучного інтелекту.

Рисунок Г.3 — Мета, об’єкт та предмет дослідження

# Завдання дослідження

Задачі дослідження:
– розробити архітектуру застосунку для визначення та управління емоційним станом людини;
– розробити алгоритми роботи застосунку для визначення та управління емоційним станом людини;
– розробити функціонал застосунку для визначення та управління емоційним станом людини;
– розробити інтерфейс користувача застосунку;
– провести тестування розробленого застосунку.

Рисунок Г.4 — Завдання дослідження

# Наукова новизна

1. Подальшого розвитку отримав метод аналізу голосу для визначення емоційного стану, який, на відміну від відомих, аналізує паравербальні характеристики мовлення за допомогою штучного інтелекту, що дозволяє отримати об’єктивну оцінку емоційного стану користувача.
2. Подальшого розвитку отримав метод контекстуального прогнозу емоційного стану, який, на відміну від відомих, попереджає потенційні емоційні виклики на основі існуючих даних про емоційний стан користувача, що дозволяє користувачам краще підготуватись до них.

Рисунок Г.5 — Наукова новизна

# Практична цінність отриманих результатів

Практичною цінністю є можливість використання розробленої системи для визначення та управління емоційним станом людини, його прогнозування.

Рисунок Г.6 — Практична цінність отриманих результатів

# Аналоги

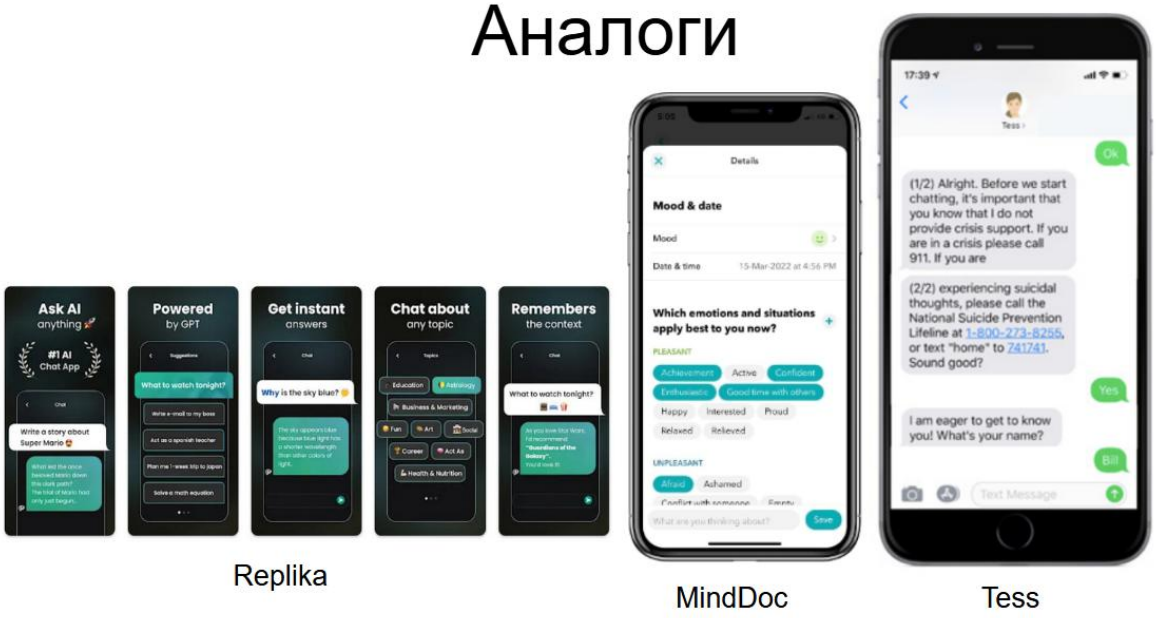


Рисунок Г.7 — Аналоги

# Порівняльний аналіз аналогів

| Характеристика | Replika | MindDoc | Tess | Власна розробка |
|---|---|---|---|---|
| Інтерактивний ШІ-помічник | + | - | + | + |
| Персоналізація відповідей | + | + | + | + |
| Моніторинг емоцій | + | + | + | + |
| Регулярне оцінювання настрою | - | + | - | + |
| Використання штучного інтелекту | + | + | + | + |
| Загальний бал | 4 | 4 | 4 | 5 |

Рисунок Г.8 — Порівняльний аналіз аналогів

# Використані технології


Рисунок Г.9 — Використані технології

## Розробка методу аналізу голосу для визначення емоційного стану

1. Користувач активує функцію запису голосу та розповідає про свій поточний стан або день.
2. Застосунок записує аудіо через MediaRecorder та візуалізує звукову хвилю в реальному часі.
3. Після завершення запису, аудіо аналізується для вилучення характеристик: темпу мовлення, висоти голосу, варіацій гучності та ритмічних патернів.
Ці характеристики передаються до Claude API для аналізу, де вони порівнюються з патернами, характерними для різних емоційних станів.
4. API повертає визначену емоцію, рівень впевненості в цьому визначенні та додаткові метрики.
5. На основі виявленої емоції та контексту користувача генерується персоналізована стратегія подолання або рекомендація.
6. Результати відображаються користувачу візуально, з поясненням виявлених голосових характеристик та їх зв'язку з емоційним станом.

Рисунок Г.10 — Розробка методу аналізу голосу для визначення емоційного стану

Розробка методу аналізу голосу для визначення емоційного стану

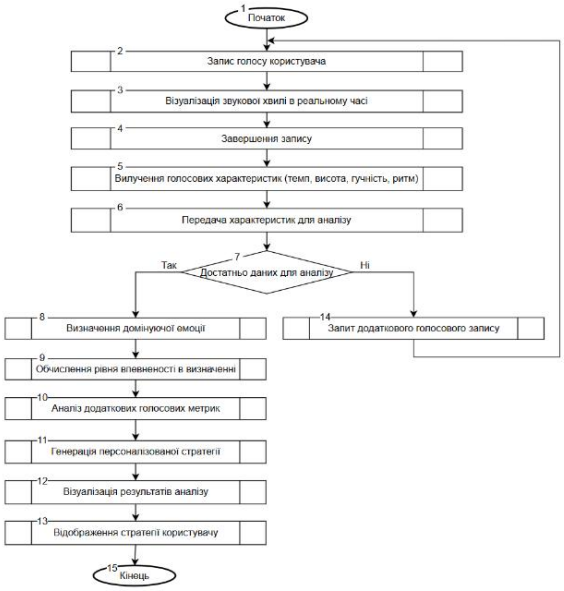


Рисунок Г.11 — Блок-схема методу аналізу голосу для визначення емоційного стану

# Розробка методу контекстуального прогнозу емоційного стану

1. Система збирає та аналізує історичні дані про емоційні стани користувача, їх інтенсивність та контекстуальні фактори, пов'язані з кожним записом.
2. Алгоритм виявляє статистично значущі кореляції між конкретними контекстуальними факторами (погода, соціальні взаємодії, сон, фізична активність) та емоційними реакціями.
3. На основі виявлених патернів створюється модель прогнозування, що враховує циклічність емоційних станів та значущі тригери.
4. Система аналізує майбутні події з календаря користувача, прогноз погоди, розклад дня та інші доступні дані.
5. Порівнюючи ці майбутні контекстуальні фактори з виявленими емоційними патернами, алгоритм визначає потенційні емоційні тригери на наступний день.
6. Для кожного передбаченого тригера система генерує персоналізовані превентивні стратегії та рекомендації.
7. Користувач отримує повідомлення з прогнозом потенційних емоційних труднощів та конкретними підготовчими діями.

Рисунок Г.12 — Розробка методу контекстуального прогнозу емоційного стану

Розробка контекстуального прогнозу емоційного стану

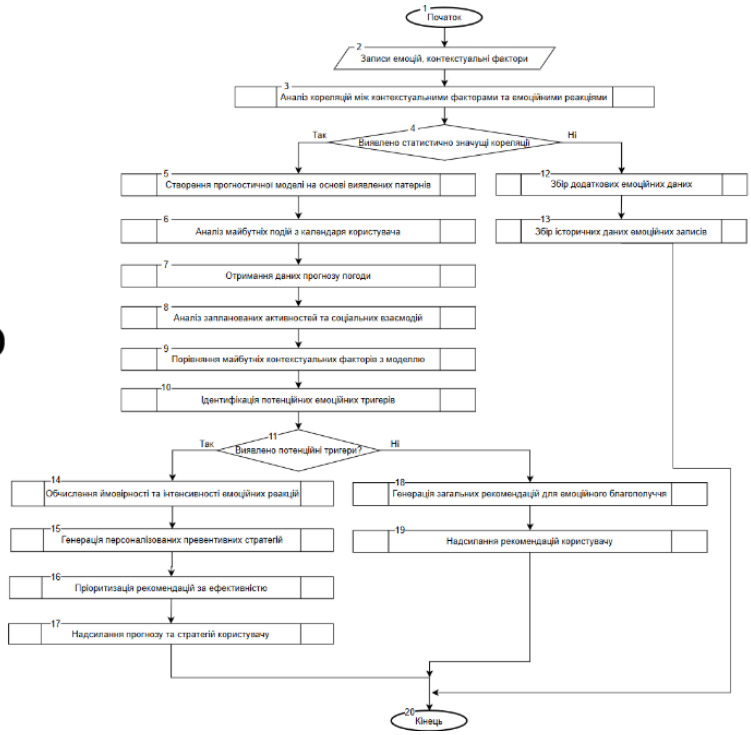


Рисунок Г.13 — Розробка контекстуального прогнозу емоційного стану

Розробка моделі застосунку

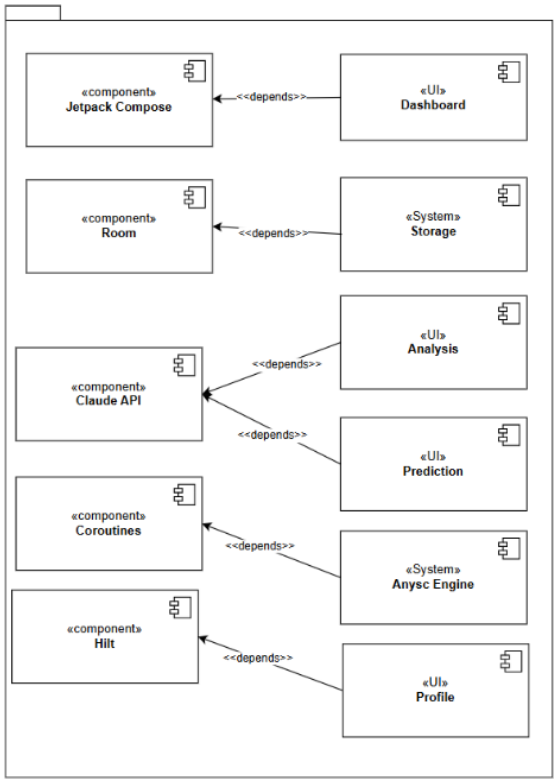


Рисунок Г.14 — Розробка моделі застосунку

Тестування системи

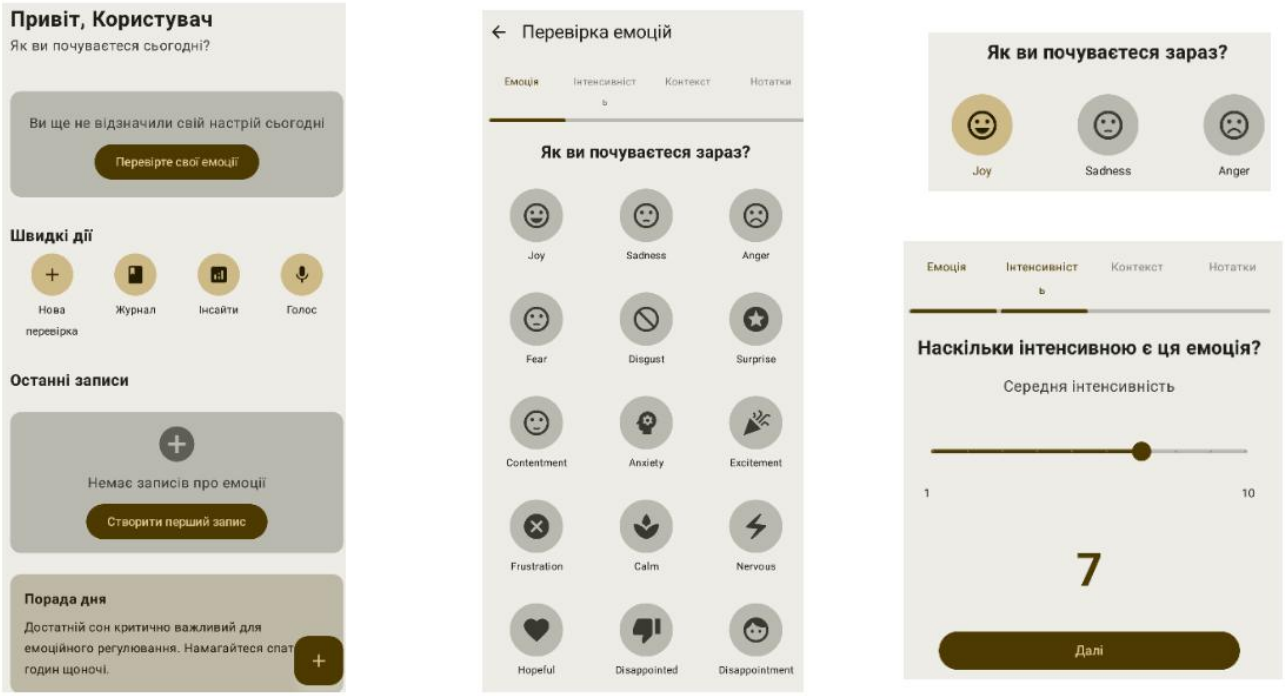


Рисунок Г.15 — Тестування

# Тестування системи

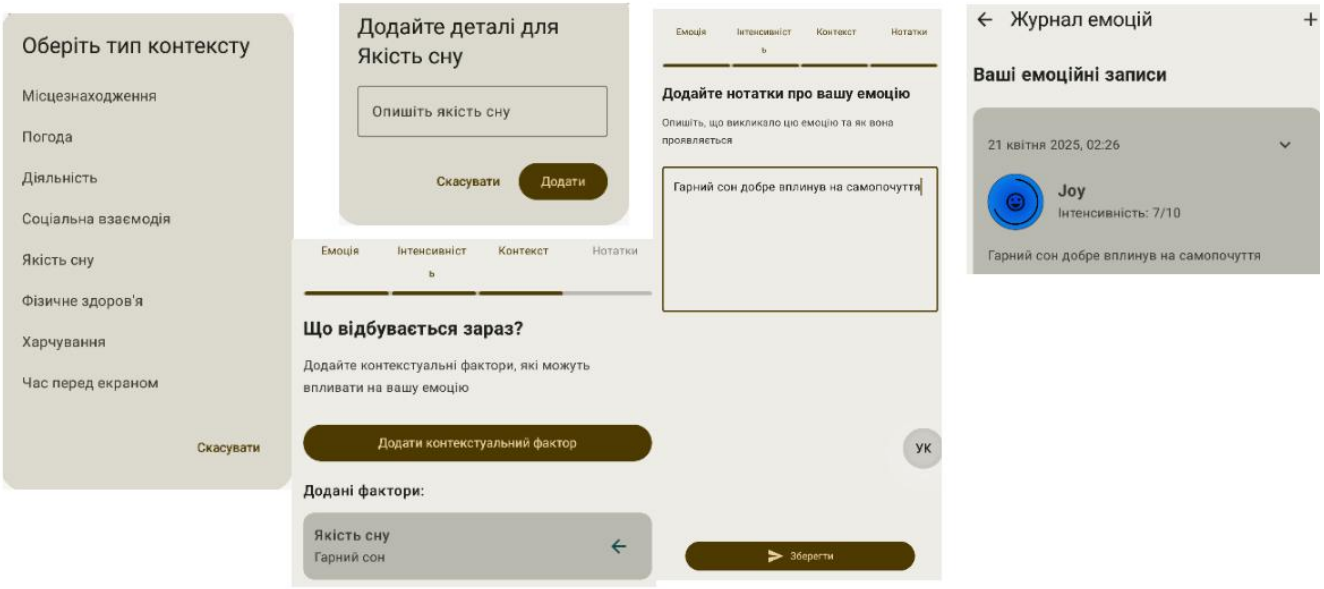


Рисунок Г.16 — Тестування системи

# Тестування системи


Рисунок Г.17 — Тестування системи

# Тестування системи

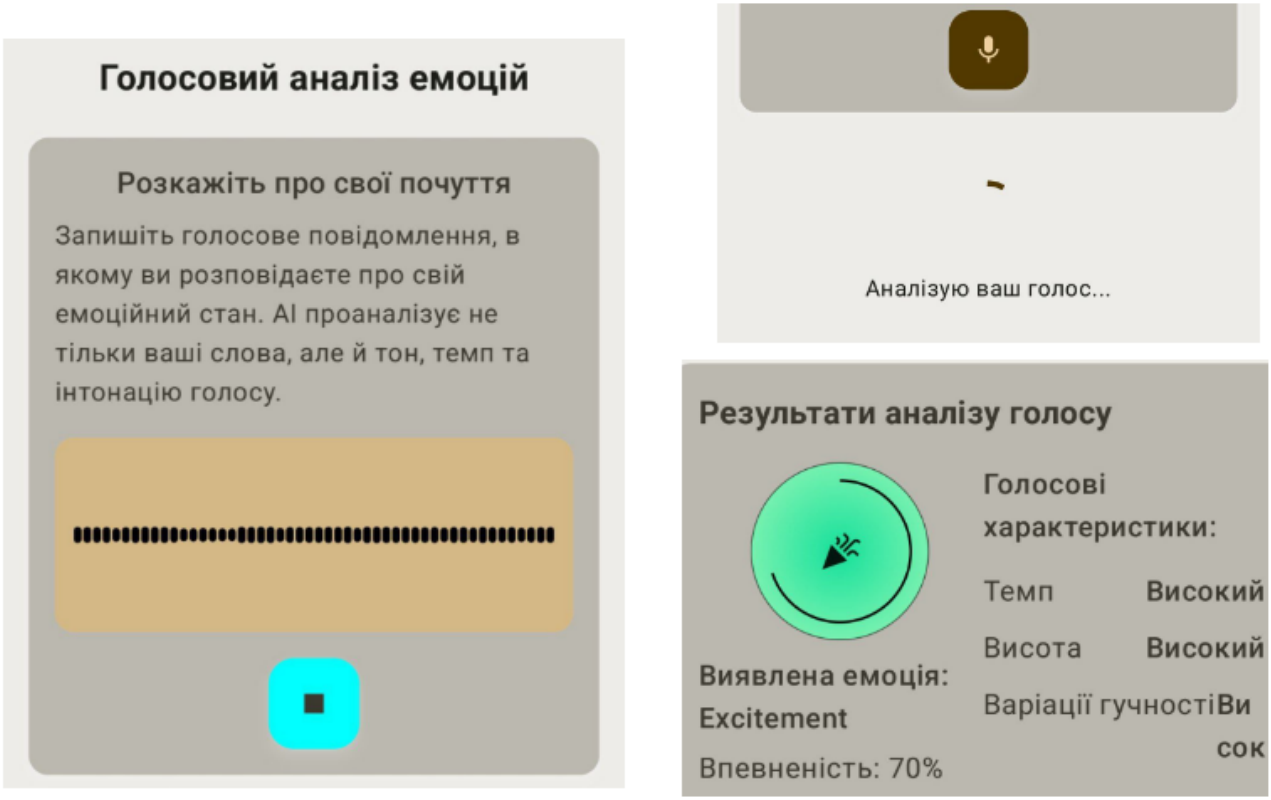


Рисунок Г.18 — Тестування системи

# Висновки

У рамках курсового проєкту було розроблено застосунок для визначення й управління емоційним станом. Розроблений застосунок дозволяє записувати власні емоції, їхню інтенсивність та контекст. Також застосунок дозволяє ділитися емоціями у голосовому режимі та отримувати аналіз не лише сказаного, а і голосу. Застосунок розроблено мовою програмування Kotlin у середовищі розробки Android Studio. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок.

Під час виконання бакалаврської кваліфікаційної роботи було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів, в результаті якого було доведено актуальність власної розробки. Проведено аналіз методів розв’язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження.

Рисунок Г.19 — Висновки (частина 1)

# Висновки

У першому розділі було проведено аналіз стану питання розробки застосунку для визначення та управління емоційним станом людини, проведено порівняльний аналіз аналогів: Replika, MindDoc, Tess. В результаті аналізу було доведено актуальність власної розробки. Проведено аналіз методів розв'язання задачі з розробки застосунку для визначення та управління емоційним станом людини, проведено постановку задач дослідження.
У другому розділі було розроблено структуру застосунку для управління емоційним станом, розроблено модель системи, алгоритми, метод аналізу голосу для визначення емоційного стану та метод контекстуального прогнозу емоційного стану.
У третьому розділі було проведено порівняльний аналіз мов програмування та обрано мову Kotlin для розробки застосунку для визначення та управління емоційним станом. Розроблено базу даних застосунку та описано її складові. Розроблена база даних використовується для збереження даних користувача для їх поступової обробки. Розроблено інтерфейс користувача, який забезпечує використання застосунку для управління емоційного стану.
У четвертому розділі, було проведено аналіз методів тестування застосунку для визначення та управління емоційним станом, обрано метод ручного тестування. Було проведено тестування розробленого застосунку, продемонстровано приклади його використання. Розроблений застосунок виконує поставлені на початку розробки завдання.

Рисунок Г.20 — Висновки (частина 2)

# Апробація результатів роботи і публікації

**Апробація матеріалів бакалаврської кваліфікаційної роботи**. Результати бакалаврської кваліфікаційної роботи доповідалися на конференції «Achievements of Science and Applied Research» (May 19- 21, 2025. Dublin, Ireland).
**Публікації.** Тези доповіді опубліковані у матеріалах конференції «Achievements of Science and Applied Research» (May 19- 21, 2025. Dublin, Ireland.

Рисунок Г.21 — Апробація результатів роботи і публікації