НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмне забезпечення для моніторингу емоцій»

Виконала:
студентка IV курсу, групи КП-91
Дорошенко Софія Олександрівна _____

Керівник:
Асистент кафедри ПЗКС,
Северін Андрій Іванович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПіСКС, к.т.н., доцент,
Клятченко Ярослав Михайлович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка_____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Дорошенко Софії Олександрівни

1. Тема проєкту «Програмне забезпечення для моніторингу емоцій», керівник проекту Северін Андрій Іванович, асистент кафедри ПЗКС, затверджені наказом по університету від «31» травня 2023 р. № 2107-с.

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   – аналіз існуючих рішень;
   – обґрунтування вибору засобів реалізації;
   – огляд розробленої системи;
   – аналіз реалізації програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   – UML діаграма функціональних можливостей системи (креслення);
   – ER-діаграма структури бази даних (креслення);
   – діаграма діяльності використання ПЗ користувачем (плакат);
   – архітектура розроблюваної системи (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2022	
2.	Розроблення та узгодження технічного завдання	25.11.2022	
3.	Розроблення структури програмного забезпечення	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2023	
7.	Програмна реалізація програмного забезпечення	10.03.2023	
8.	Тестування програмного забезпечення	17.03.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2023	
11.	Підготовка графічної частини дипломного проєкту	21.04.2023	
12.	Оформлення документації дипломного проєкту	28.05.2023	

Студент Софія ДОРОШЕНКО

Керівник проєкту Андрій СЕВЕРІН

АНОТАЦІЯ

Даний проєкт присвячений розробці мобільного застосунку, який дозволяє користувачам щоденно отримувати різні завдання, відстежувати та занотовувати свої емоції та отримувати аналіз даних. Застосунок надає можливість користувачам записувати свої емоційні стани за допомогою віджетів або вводити дані вручну. Зібрані дані про емоційний фон супроводжуються інформацією про виконані завдання. Користувачі можуть отримувати звіти та графіки, що відображають залежність між їхнім емоційним станом та виконаними завданнями. Застосунок також надає можливість отримання рекомендацій для покращення емоційного благополуччя.

Під час розробки даного програмного забезпечення було проведено докладний аналіз предметної області та проаналізованопродукти-аналоги. На основі цього, були сформульовані функціональні вимоги до розроблюваного ПЗ. Були проведені дослідження засобів розробки та вибрані відповідні бібліотеки для реалізації функціональних можливостей, а також була обрана відповідна система керування базами даних.

В роботі представлені основні характеристики розробленого програмного забезпечення, детально розглянута роль виконавця завдань, а також надані алгоритми роботи програми з урахуванням ролі. Додатково висвітлені можливості розвитку та розширення застосунку для майбутніх потреб користувачів.

ABSTRACT

This project is dedicated to the development of a mobile application that allows users to receive daily tasks, track and record their emotions, and receive data analysis. The application enables users to record their emotional states using widgets or manually input data. The collected data on emotional background is accompanied by information on completed tasks. Users can receive reports and graphs that depict the relationship between their emotional state and completed tasks. The application also provides recommendations for improving emotional well-being.

During the development of this software, a thorough analysis of the subject domain and analysis of similar products was conducted. Based on this, functional requirements for the developed software were formulated. Research on development tools was carried out, and suitable libraries for implementing the functional capabilities were selected, along with an appropriate database management system.

The work presents the main characteristics of the developed software, provides a detailed examination of the role of task execution, and offers algorithms for the program's operation considering the role. Additionally, possibilities for the application's future development and expansion to meet the users' needs are highlighted.

ДП.045490-01-90 Програмне забезпечення для моніторингу емоцій при виконанні завдань. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045490-02-91	Програмне забезпечення для	4	
	моніторингу емоцій. Технічне		
	завдання		
ДП.045490-03-81	Програмне забезпечення для	50	
	моніторингу емоцій. Пояснювальна		
	записка		
ДП.045490-04-51	Програмне забезпечення для	4	
	моніторингу емоцій. Програма та		
	методика тестування		
ДП.045490-05-34	Програмне забезпечення для	9	
	моніторингу емоцій. Керівництво		
	користувача		
ДП.045490-06-99	Програмне забезпечення для	1	
	моніторингу емоцій. Функціональні		
	можливості системи. UML-діаграма		
ДП.045490-07-99	Програмне забезпечення для	1	
	моніторингу емоцій. Структура		
	бази даних. ER-діаграма		
ДП.045490-08-98	Програмне забезпечення	1	
	для моніторингу емоцій.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЕМОЦІЙ

Технічне завдання

ДП.045490-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Андрій СЕВЕРІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Софія ДОРОШЕНКО

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для моніторингу емоцій.

Галузь застосування: інформаційні технології, особистий розвиток.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджено кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначення для відслідковування власного емоційного стану під час виконання різних завдань. Програма допомагає відстежувати зміни в емоційному фоні та надає загальні поради і рекомендації.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна система повинна забезпечувати такий основні функціональні можливості:

1) реєстрація/автентифікація;
2) вибір типу завдання для виконання;
3) можливість записати емоції та почуття;
4) надання аналізу даних в виді графіків;
5) надання рекомендацій щодо покращення емоційного стану;
6) можливість перегляду архіву даних;
7) можливість додавати свої власні завдання;
8) отримання нагадування про щоденне виконання завдань.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення.

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту 13.11.2022
Розроблення та узгодження технічного завдання..... 25.11.2022
Розроблення структури програмного забезпечення 15.12.2022
Підготовка матеріалів першого розділу дипломного проєкту 30.12.2022
Розроблення дизайну сторінок та графічних елементів..... 03.02.2023
Підготовка матеріалів другого розділу дипломного проєкту..... 19.02.2023
Програмна реалізація програмного забезпечення 10.03.2023
Тестування програмного забезпечення..... 17.03.2023
Підготовка матеріалів третього розділу дипломного проєкту 30.03.2023
Підготовка матеріалів четвертого розділу дипломного проєкту 12.04.2023
Підготовка графічної частини дипломного проєкту 21.04.2023
Оформлення документації дипломного проєкту 26.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"
Завідувач кафедри
_____ Євгенія СУЛЕМА
"___" _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЕМОЦІЙ

Пояснювальна записка

ДП.045490-03-81

"ПОГОДЖЕНО"

Керівник проєкту:
_____ Андрій СЕВЕРІН

Нормоконтроль:
_____ Микола ОНАЙ

Виконавець:
_____ Софія ДОРОШЕНКО

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

БД – база даних.

СУБД – система управління базами даних.

MVC – Model-View-Controller – шаблон проєктування програмного забезпечення.

ORM – Object-Relational Mapping – технологія для зв'язку між об'єктно-орієнтованим програмним забезпеченням та реляційною БД.

Фронтенд – frontend – клієнтська частина – клієнтська сторона користувацького інтерфейсу до програмно-апаратної частини сервісу;

Бекенд – backend – серверна частина – частина сайту, яка відповідає за роботу сайту;

SQL – Structured Query Language – мова запитів до баз даних.

Kotlin – мова програмування для JVM, Android та інших платформ.

Android Studio – інтегроване середовище розробки для Android-приладів, яке підтримує Kotlin.

Coroutines – бібліотека для асинхронного програмування в Kotlin.

REST API – Representational State Transfer Application Programming Interface – архітектурний стиль вебсервісів, який забезпечує обмін даними між клієнтом і сервером.

Ktor – фреймворк для розробки застосунків на Kotlin.

JSON – JavaScript Object Notation – формат обміну даними, який є стандартом у веброзробці.

API – Application Programming Interface – інтерфейс програмування застосунків, який забезпечує доступ до функціональності програмного забезпечення.

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту, який використовується для обміну даними в мережі Інтернет.

ВСТУП

За останні кілька років зростає інтерес до теми психологічного благополуччя та ментального здоров'я [1]. Велика кількість негативних факторів сьогодення, що впливають на наше повсякденне життя мають сильний вплив на нашу емоційну стабільність та загальний стан здоров'я. Швидкий ритм життя, стрес, бажання завжди бути продуктивними знижує спроможність відслідковувати та аналізувати власний емоційний стан. Емоції, що виникають при щоденному виконанні як рутинних, так і нових завдань мають значний вплив на емоційний стан людини не лише протягом дня, а і протягом усього життя.

Було доведено, що коли люди не визнають та не помічають свої емоції, вони мають нижчий рівень благополуччя і перебувають в стресі. Уникнення наших почуттів має велику ціну. Та наявність правильного словника емоцій дозволяє нам побачити справжню проблему, зрозуміти її більш чітко і побудувати план для її вирішення [2].

Для багатьох людей, що прагнуть покращити свій стан здоров'я, щоденне виконання завдань є ключовим елементом [3]. Однак, рутина може бути пов'язана зі стресом та емоційним напруженням. Тому, створення застосунку, який пропонуватиме різні завдання, що дозволять позитивно вплинути на емоційний фон людини, а також надаватиме можливість розрізняти та занотовувати свої емоції, є актуальною задачею.

Метою даного дипломного проєкту є розроблення програмного забезпечення для моніторингу емоцій при виконанні завдань.

Основними функціями розроблювоного програмного забезпечення є:

- отримання завдань, виконання яких стимулюватимуть різного виду емоції та почуття;
- записування емоцій, що виникають при виконанні завдань;
- аналіз отриманих даних та формування рекомендацій.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Постановка проблеми

Актуальною проблемою сьогодення є приділяння недостатньої уваги до емоцій, що виникають в повсякденному житті [4]. Це може призводити до негативних результатів у різних сферах, включаючи психологію, медицину, бізнес та освіту. Розглянемо докладніше дерево проблем, зображене на рис. 1.1.

Рис. 1.1. Дерево проблем

Розпишемо детальніше кожну з його гілок:

1. Недостатня самосвідомість [5]:
   - багато людей не мають достатнього рівня усвідомленості щодо своїх емоцій та їх впливу на загальний стан і добробут;
   - недостатня компетенція в розумінні та управлінні власними емоціями та їх виявлення;
   - багато людей не усвідомлюють, як їхні емоції впливають на їхню продуктивність, виконання завдань та досягнення цілей.

2. Відсутність систематичного моніторингу емоцій:
    - брак системи для відстеження та аналізу емоцій може заважати виявленню зв'язків між емоціями та щоденними завданнями;
    - потреба в допомозі та інструментах для розвитку емоційної грамотності та досягнення більшої самосвідомості.
3. Нездоровий емоційний фон:
    - відсутність ефективних стратегій для керування емоційним фоном може призводити до негативного впливу на розумовий та фізичний добробут людини;
    - поганий емоційний фон може призводити до втрати мотивації та зниження енергії для виконання завдань;
    - відчуття стресу та тривоги можуть заважати концентрації, прийняттю рішень та виконанню завдань.
4. Відсутність достатньої стимуляції для викликання позитивних емоцій:
    - відсутність достатньої кількості особистих інтересів може призводити до монотонності та втрати радості у виконанні повсякденних завдань;
    - недостатнє стимулююче середовище та відсутність позитивно налаштованих соціальних зв'язків можуть впливати на емоційний стан людини та її взаємодію з робочими завданнями;
    - відсутність досягнень та мотивуючих факторів можуть призводити до зменшення позитивних емоцій.

Складності з розумінням власного емоційного фону можуть призводити до незадоволеності, стресу та інших негативних наслідків. Застосунки для моніторингу емоцій можуть допомогти вирішити цю проблему, дозволяючи користувачам приділити більше уваги своїм емоціям та почуттям. Це може бути особливо корисним для вивчення емоційних реакцій на різні інформаційні та стимулюючі фактори та для розвитку емоційного інтелекту [6].

Помічати емоції і назвати їх, відмічати ці почуття, вчити нові назви, що описують емоції, вести щоденник емоцій є важливими інструментами для усвідомлення своїх емоцій, кращого розуміння себе та своїх бажань [7]. Коли користувачі відстежують та розуміють емоції, вони можуть свідоміше сфокусуватись на своїй реакції на різні ситуації та стимули, і в разі потреби змінити її.

Цим самим використання такого програмного забезпечення може мати значний вплив на покращення якості міжособистісних взаємин, збільшити рівень емоційного благополуччя та знизити ризик стресу та інших негативних наслідків. Завдяки підвищенню свідомості, користувачі почнуть приділяти більше уваги своїм емоціям та почуттям, що сприятиме покращенню емоційної стійкості. Використання таких технологій допоможе зрозуміти важливість піклування про своє емоційне благополуччя та вміння керувати своїми емоціями.

Підсумовуючи, основним завданням розроблюваного програмного забезпечення є вирішення проблеми, пов'язаної з недооцінкою важливості приділянню уваги емоційному стану та усвідомленню широкого спектру почуттів, що супроводжують людей щодня. Використання такого застосунку матиме позитивний вплив на загальне здоров'я та благополуччя, що в свою чергу покращить рівень у різних аспектах життя, включаючи відносини, кар'єру та фізичне здоров'я.

1.2. Аналіз наявних аналогів

На даний момент існує багато мобільних- та вебзастосунків, що приділяють увагу темі психологічного та морального благополуччя. У цьому розділі буде розглянуто та проаналізовано декілька прикладів таких застосунків. Зокрема, буде розглянуто їх головну мету, функціональні можливості, недоліки та переваги для користувачів. Також буде здійснено порівняння за наступними критеріями:

1. Наявність безкоштовної версії з повним функціоналом.

2. Можливість записувати емоції під час виконання завдань.
3. Наявність рекомендацій щодо поліпшення емоційного фону.
4. Можливість обирати емоції зі списку.
5. Аналіз даних щодо емоцій.
6. Наявність запропонованих завдань.
7. Вибір фахівця психологічного направлення та(або) консультації з психологом.
8. Наявність курсів та навчальних матеріалів.

1.2.1. *Застосунок VOS*

VOS – це мобільний застосунок, призначений для психологічного благополуччя [8]. Практичні функціональні можливості полягають в допомозі користувачам покращити свій психологічний стан шляхом надання доступу до психологічних інструментів та ресурсів. Крім того, застосунок може допомогти користувачеві знизити рівень стресу та тривоги, підвищити самосвідомість та розвинути навички саморегуляції емоцій.

Технічний функціональні можливості застосунка VOS включає наступне:

- психологічні тести та опитування для оцінки емоційного стану користувача;
- вправи на релаксацію, медитацію та зосередження;
- доступ до психологічної підтримки через відео-консультації з ліцензованими психологами;
- техніки саморегуляції емоцій та стресу;
- можливість вести емоційний журнал для відстеження стану користувача з часом;
- різноманітні курси та навчальні матеріали для покращення психологічного благополуччя.

Далі розглянемо деякі недоліки та переваги цього застосунку. Існують наступні недоліки:

- Відсутність індивідуальних налаштувань. Користувачі можуть знайти деякі вправи і завдання непідходящими для них, але не мають можливості змінити їх або додати свої власні.
- Відсутність детального звітування. Застосунок збирає інформацію про емоційний стан користувача, але не надає детальних звітів або рекомендацій на основі цієї інформації.
- Потребує постійного підключення до Інтернету. Для використання застосунку необхідне підключення до Інтернету, що може бути проблемою для користувачів з обмеженим доступом до Інтернету або низької швидкості зв'язку.
- Незначна вартість. Більшість функцій застосунку є платними, що може стати перешкодою для деяких користувачів.

Незважаючи на це VOS володіє широкими функціональними можливостями, що перекривають всі недоліки. Перш за все, застосунок має інтуїтивно зрозумілий і простий інтерфейс, що робить його легким у використанні. Присутність різноманітних завдань для психологічного благополуччя, таких як медитації, заняття йогою, візуалізації, та інші, робить застосунок цікавим та інтерактивним. Також усі дані користувачів зберігаються в безпечному місці, що забезпечує конфіденційність та приватність даних.

Більш того, користувачі можуть отримати допомогу та консультації від спеціалістів з психології та інших сфер, що робить застосунок більш корисним для тих, хто знаходиться в пошуку допомоги. Такі функціональні можливості, як аналітика прогресу у вирішенні різних задач, допомагає зрозуміти користувачам успіхи та досягнення у підвищенні психологічного благополуччя.

На рис. 1.2 зображено користувацький інтерфейс застосунку VOS.

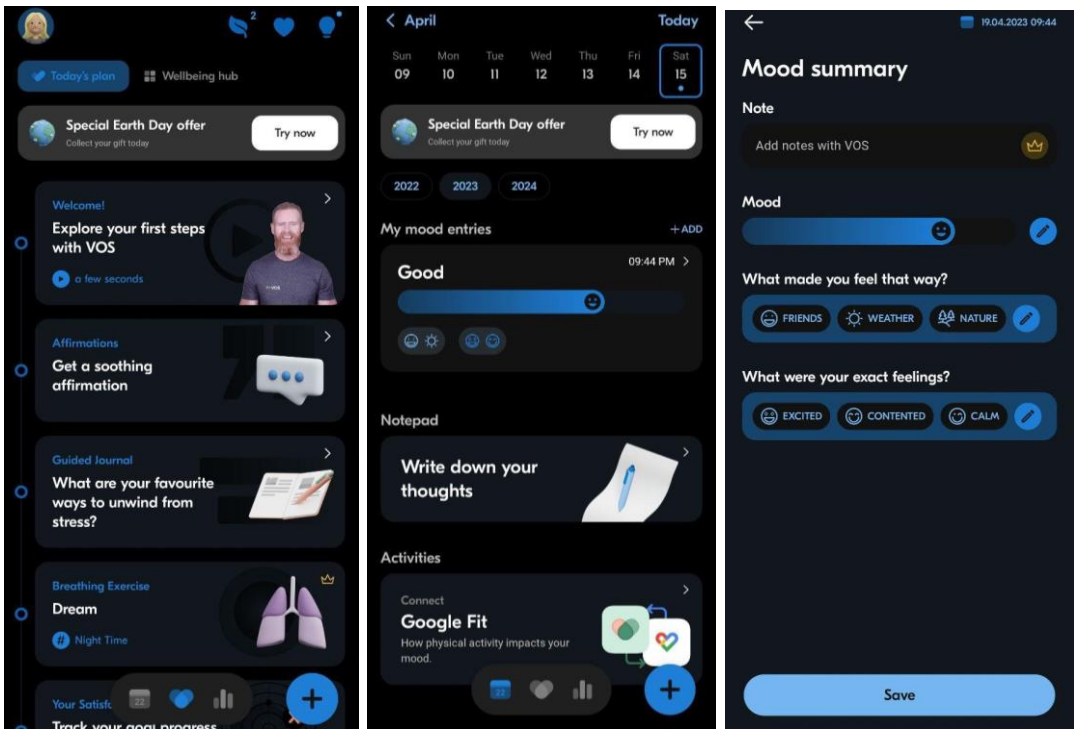


Рис. 1.2. Приклади інтерфейсу застосунку VOS

1.2.2. *Застосунок Thera*

Thera – це сервіс для психотерапії, який надає можливість звернутися до сертифікованого психотерапевта онлайн та отримати консультацію з будь-якого місця. Застосунок також містить функції для ведення щоденника настрою та стеження за прогресом, а також психологічні вправи та ресурси для самостійного покращення психічного здоров'я.

Мобільний застосунок Thera пропонує широкі функціональні можливості, що дозволяють користувачам зосередитися на своєму психічному здоров'ї та благополуччі. Однією з ключових функцій є можливість ведення щоденника настрою. Користувачі можуть вносити дані про свій настрій та відчуття щодня, що допомагає їм відстежувати тенденції та зміни в своєму емоційному стані.

Крім того, застосунок надає доступ до різноманітних психологічних вправ та ресурсів для самостійного покращення психічного здоров'я. Наприклад, можна скористатися медитаційними вправами, техніками релаксації, вправами на зосередженість та багато іншого. Також, застосунок

містить розділ з порадами від сертифікованих психотерапевтів та психологів, що можуть допомогти користувачам знайти відповіді на питання та побороти сталий стрес, тривожність та депресію.

Thera надає можливість відстежувати свій прогрес в покращенні психічного здоров'я. Користувачі можуть створювати свої цілі та плани, встановлювати нагадування та відстежувати свій прогрес виконання цих завдань. Загалом, функціональні можливості застосунку THERA допомагають користувачам зосередитися на своєму психічному благополуччю, пріоритезувати емоційний стан та працювати над своїм розвитком.

Незважаючи на багато переваг, які має застосунок Thera, він також має свої недоліки. Основні з них є наступні:

- У застосунку немає можливості звернутися до ліцензованого психотерапевта, що може лімітувати тих, хто потребує серйозної підтримки.
- Хоча Thera містить різноманітні вправи для поліпшення психологічного здоров'я, їх кількість обмежена і може не задовольнити потреб користувачів.
- Незважаючи на те, що дане програмне забезпечення допомагає багатьом людям, та немає гарантії, що воно підійде всім користувачам. Ефективність застосунку може значно варіюватися в залежності від конкретної ситуації та потреб користувача.

Та якщо користувач хоче більше самостійно вивчати методи психологічної підтримки, то Thera може буде прекрасни вибором.

На рис. 1.3 ми бачимо користувацький інтерфейс застосунку Thera.

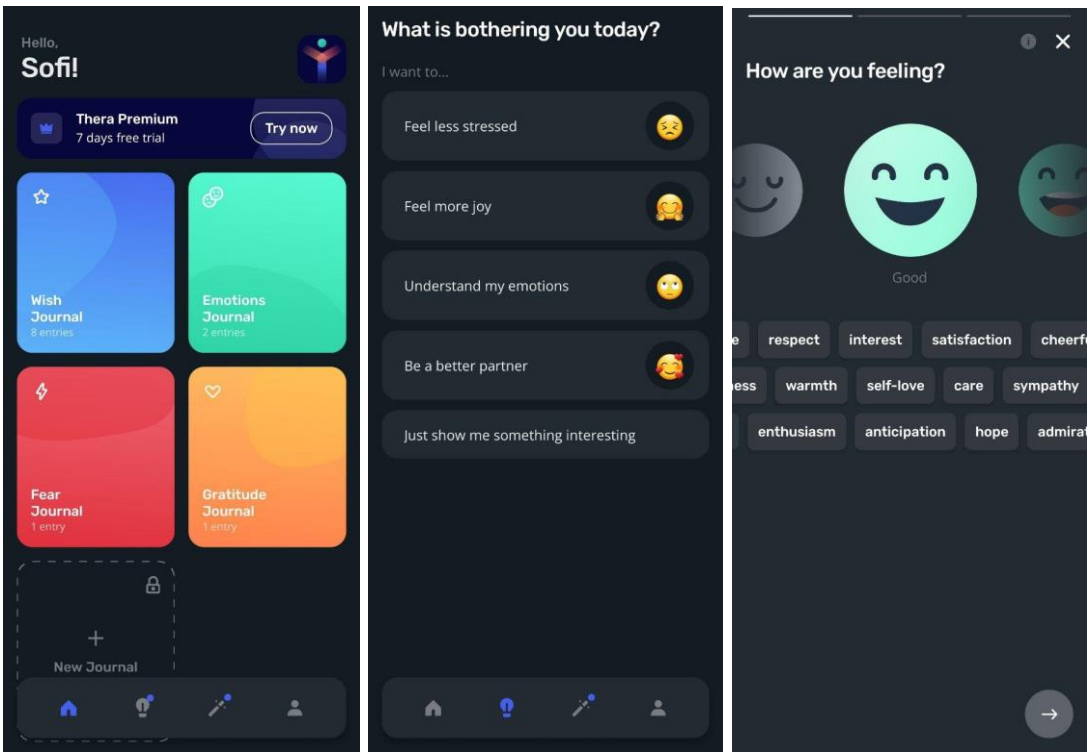


Рис. 1.3. Приклади інтерфейсу застосунку Thera

1.2.3. *Застосунок Daylio*

Daylio – це щоденник настрою для відстеження свого емоційного стану та діяльності [9]. Основна мета ПЗ – допомогти користувачам краще зрозуміти свої емоції, підвищити свідомість та покращити самопочуття.

Цей застосунок має багато практичних функцій. Користувач може вести щоденник настрою за допомогою іконок та коментарів, що дозволяє зберігати записи та отримувати візуальне уявлення про своє емоційне становище протягом часу.

Крім того, користувач може стежити за своїм фізичним здоров'ям, створювати та відстежувати досягнення своїх цілей в різних сферах життя. Застосунок надає можливість переглядати статистику своєї активності та настрою за певний період, а також візуалізувати дані за допомогою діаграм та графіків.

Також, Daylio надсилає користувачеві нагадування про ведення щоденника та може давати підказки щодо збереження позитивного настрою.

Ще однією функцією є автоматичне створення резервних копій даних, щоб запобігти їх втраті.

Розглянемо кілька недоліків застосунку:

- Застосунок пропонує обмежену кількість категорій для відображення настрою, що може бути недостатньо для користувачів з більш складними емоційними станами.
- У застосунку можна налаштувати лише категорії настрою, що вже запропоновані. Немає можливості додавати власні категорії, що може приносити деякі незручності.
- Застосунок не має функції синхронізації з хмарою, що може бути проблемою для користувачів, які хочуть мати доступ до своїх даних на кількох пристроях.
- Деякі додаткові функції, такі як додавання фотографій та нотаток, доступні лише в платній версії, що може бути незручним для користувачів з фінансовою обмеженістю.

Незважаючи на це, застосунок має важливі переваги, такі як:

- Простий та інтуїтивно зрозумілий інтерфейс, що дозволяє легко вести щоденник настрою та встановлювати цілі для покращення психологічного благополуччя.
- Гнучка настройка категорій для ведення щоденника, що дозволяє користувачам вибирати ті категорії, які найбільше відповідають їхнім потребам та цілям.
- Можливість зберігати та аналізувати дані про настрій та цілі, що дозволяє відстежувати свій прогрес та визначати області, в яких потрібно зосередитись для покращення свого психологічного стану.
- Підтримка різних мов та можливість експорту даних у форматі CSV для подальшого аналізу.
- Безкоштовна версія застосунку має достатньо функціональних можливостей для задоволення потреб більшості користувачів.

Хоча у застосунку є кілька недоліків, він все ж є дуже корисним і може допомогти в покращенні психічного здоров'я та підтримці емоційного благополуччя.

На рис. 1.4 зображено користувацький інтерфейс застосунку Daylio.

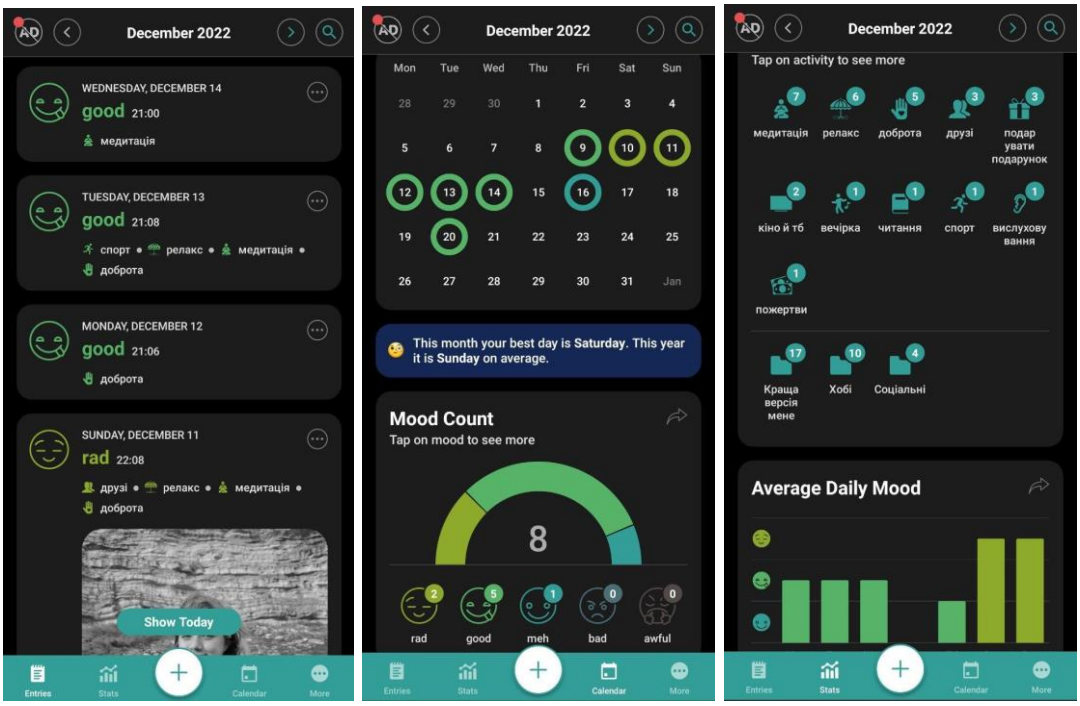


Рис. 1.4. Приклад інтерфейсу застосунку Daylio

1.2.4. *Висновки*

Підсумовуючи аналіз наявних аналогів, розпишемо загальне порівняння (табл. 1.1).

Таблиця 1.1

Порівняння функціональних можливостей аналогів

Критерій	VOS	Thera	Daylio
Безкоштовна версія	Частково	Частково	Так
Запис емоцій	Так	Так	Так
Рекомендації	Так	Так	Ні

Продовження табл. 1.1

Вибір емоцій	Так	Так	Так
Аналіз даних	Так	Так	Так
Пропонування завдань	Так	Так	Ні
Допомога професіоналів	Так	Так	Ні
Курси та матеріали	Так	Так	Ні

Як бачимо, запис та вибір емоцій зі списку, а також аналіз даних наявні в усіх розглянутих застосунках. Отже цей функціонал є найважливішим для розвитку емоційного інтелекту та покращення самосвідомості. Пропонування різних завдань та рекомендації не є функціоналом застосунку Daylio, та наявні в двох інших аналогах. Те ж саме можемо сказати і про таку опцію, як допомога від професіоналів та пропонування різних курсів, матеріалів. Повністю безкоштовна версія є наявною лише в останньому застосунку. VOS та Thera надають можливість безкоштовного використання основного функціоналу, та для використання застосунків в повному обсязі вимагається щомісячний або щорічний грошовий внесок.

Кожен з цих застосунків має свої унікальні особливості та переваги, що дозволяють користувачам зайнятися відслідковуванням та поліпшенням свого емоційного стану. У наступному підрозділі буде розглянуто та сформульовано функціональні вимоги щодо розроблюваного застосунку з урахуванням вище перелічених критеріїв та виконаного аналізу.

1.3. Аналіз вимог до розроблюваного ПЗ

Проаналізувавши наявні застосунки-аналоги, сформулюємо вимоги до розроблюваного програмного забезпечення.

Неодмінно, велика кількість функціональності з вже популярних на ринку застосунків була досліджена і має позитивний вплив на цільову аудиторію. Та все ж даний проєкт несе в собі додаткові цілі для розвитку емоційного інтелекту користувачів, а саме вихід з зони комфорту за допомогою виконанні запропонованих завдань та можливості розрізняти та занотовувати свої емоції при їх виконанні. Це позитивно впливає на креативність, досягнення цілей, росту та розвитку як особистості [10]. Розширення зони комфорту може мати позитивний вплив на особистість, проте важливо зберігати баланс: якщо негативна самомотивація переважає, розширення зони комфорту може спричинити стагнацію особистості [11].

Розглянемо дерево цілей для розробки програмного забезпечення (рис. 1.5).

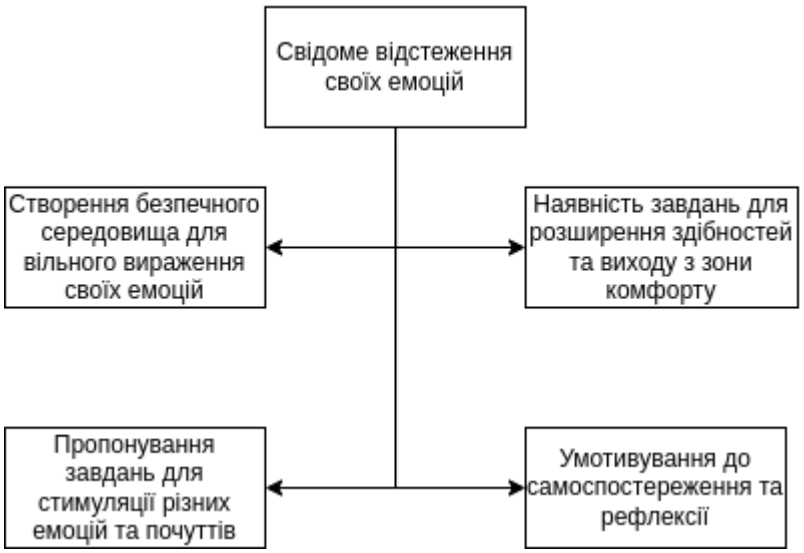


Рис. 1.5. Дерево цілей

Розпишемо кожну з гілок дерева цілей:

1. Створення безпечного середовища для вільного вираження своїх емоцій:
   - забезпечити приватність користувачів, щоб вони почувалися комфортно виражати свої емоції без страху відсудження;
2. Пропонування завдань для стимуляції різних емоцій та почуттів:

- представляти різноманітних завдання, спрямовані на викликання позитивних емоцій, таких як радість, задоволення, гармонія тощо;
- включати вправи на релаксацію, медитацію, глибоке дихання, що сприяють зниженню стресу та покращенню настрою.

3. Умотивування до самоспостереження та рефлексії:
   - стимулювати користувача до регулярного виділення часу для самоспостереження;
   - надавати аналіз даних для рефлексії та висновків;
   - можливість перегляду рекомендацій щодо покращення емоційного фону.
4. Наявність завдань для розширення здібностей та виходу з зони комфорту:
   - запропонувати завдання, що спонукають користувача вийти зі звичних сценаріїв та пробувати новий досвід;
   - включити в додаток завдання на розвиток навичок, таких як самообізнаність, креативність, що допоможуть розширити спектр емоцій.

Тож сформулюємо наступні функціональні вимоги до розроблюваного програмного забезпечення:

- вибір одного з 3 варіантів завдання кожного дня (з поданих тем: спостереження (observing), дослідження (exploring), проживання (experiencing));
- можливість обрати і занотовувати почуття/емоції;
- можливість переглядати місцезнаходження та фото збережені у певному емоційному стані.

Зважаючи на вищенаведене, можна зробити висновок, що розроблюваний додаток матиме наступні конкурентні переваги:

- можливість отримувати різні завдання щодня;
- можливість перегляду аналітики емоцій;

- безкоштовна версія застосунку;
- наявність рекомендацій щодо поліпшення емоційного стану.

Основна перевага даного програмного забезпечення перед аналогами полягає в наявності завдань, які стимулюють різні емоції, і водночас у можливості розуміння та усвідомлення виникаючих емоцій. Користувач має свободу вибору типу завдань та їх виконання, і може відзначити свої почуття до, під час та після виконання. Такий підхід дозволяє акцентувати увагу на повному спектрі почуттів, що виникають при виконанні різних завдань.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Аналіз функціональних вимог

Аналіз функціональних вимог є важливою передумовою проєктування застосунку для моніторингу емоцій під час виконання завдань. Розпишемо основні функціональні вимоги розроблюваного ПЗ:

1. Реєстрація та автентифікація – при першому використанні програми користувач може створити обліковий запис, а в подальшому він зможе автентифікуватись при вході в додаток. Такі дії призначені для конфіденційності даних користувача.
2. Вибір завдань – застосунок повинен надавати користувачам можливість вибору завдання для виконання. Є три типи завдань: observe, explore, experience.
    - Observe – надає можливість сфотографувати або обрати фото з галереї і записати, які емоції користувач відчував в цей момент.
    - Explore – завдання пов’язане з дослідженням нових місцин, їх позначенням на карті та запису емоцій, які були пережиті при перебуванні в новому навколишньому середовищі.
    - Experience – надає можливість користувачам отримати завдання, пов’язане з активностями, як: готування, прогулянка, медитації. танці, читання. Після чого можливо також занотувати почуття, що виникли при виконанні такого типу завдання.
3. Моніторинг емоцій – функція запису емоцій та почуттів під час чи після виконання обраного завдання.
4. Додавання завдань – функція, що надає можливість додавати власні завдання до такого типу завдань, як experience.
5. Аналіз емоцій – можливість аналізу даних про емоційний стан користувача та надання звітності про те, як різні типи завдань впливають на його емоційний стан. Аналіз у виді графів надає

користувачам інформацію про найбільш часті емоції за тиждень, та амплітуду коливань настрою.

6. Рекомендації – повинен надавати загальні рекомендації користувачам щодо покращення благополуччя.
7. Архів даних – можливість переглядати збережені дані, такі як фото, емоції, місцезнаходження.

Усі ці вимоги до функціональності застосунку мають бути відповідним чином реалізовані для забезпечення оптимальної ефективності та зручності користування.

2.2. Засоби реалізації застосунку

Для реалізації ПЗ буде використано такі інструменти та технології:

- Платформа розробки – Android.
- Мова програмування – Kotlin.
- Фреймворки – Ktor (backend) та Jetpack Compose (frontend).
- Шаблон проєктування – MVVM.
- СУБД – SQLite.

Детальний аналіз кожного з них буде розглянуто в наступних підрозділах.

2.2.1. *Платформа для розробки*

Для розробки дипломного проєкту була обрана така платформа для розробки як Android. У цьому підрозділі буде порівняно наступні платформи: Android та iOS, та виконані висновки щодо вибору.

Платформа розробки Android – це набір інструментів і середовища, розроблений компанією Google, для створення застосунків, які працюють на операційній системі Android.

Android є відкритою та гнучкою платформою, яка надає розробникам більшу свободу для розширення функціональності застосунків. Великий ринок і аудиторія дозволяють розробникам залучити більше користувачів та

мати ширший охоплення. Розробники Android також мають більшу свободу розміщення продуктів на різних платформах та маркетплейсах. Крім того, інтеграція з Google-сервісами надає розробникам доступ до потужних інструментів та сервісів, що полегшують розробку та взаємодію з екосистемою Google [12].

Звичайно, є деякі недоліки цієї платформи розробки. Наприклад, необхідність підтримки широкого спектру пристроїв і версій операційної системи. Також, через відкритість платформи, Android може бути більш вразливим до безпекових загроз і зловмисного програмного забезпечення, порівняно зі стійкою і контрольованою екосистемою iOS.

Зважаючи на різні фактори та особливості, важливо розглянути платформу розробки мобільних застосунків iOS та порівняти її з Android.

Переваги iOS:

- Висока безпека та стабільність.
- Оптимізована апаратна інтеграція.
- Сприятлива аудиторія для монетизації.
- Оновлення операційної системи.

Недоліки iOS:

- Обмежена свобода розробки.
- Висока вартість пристроїв.
- Обмежена інтеграція зі сторонніми сервісами

Узагалі, iOS є потужною та високоякісною платформою для розробки мобільних застосунків зі своїми перевагами та недоліками [13]. Вибір між Android і iOS залежить від потреб цільової аудиторії та конкретних вимог користувачів.

Вибір Android для мого дипломного проєкту обумовлений декількома факторами: широкий ринок та потенціална аудиторія, гнучкість та свобода розробки, відкритий характер платформи і можливості інтеграції зі сторонніми сервісами. Android надає доступ до великої кількості

користувачів, дозволяє адаптувати додаток до різних пристроїв, і має потенціал для інновацій та розвитку.

2.2.2. *Мова програмування*

Для розробки мобільних застосунків існує кілька популярних мов програмування, які можуть бути використані. Наприклад: Java, C#, React Native, Kotlin та багато інших. Порівняємо перелічені мови програмування та розберемо обраної мови для реалізації програмного забезпечення.

Java є однією з найпопулярніших мов програмування, особливо для розробки мобільних застосунків на платформі Android [14]. Вона відома своєю надійністю, стабільністю та широкою підтримкою. Java пропонує багатий набір бібліотек і фреймворків, що полегшують розробку програмного забезпечення. Однак, Java може бути трохи важкодоступною для новачків і вимагати більше написаного коду порівняно з іншими мовами.

C# є мовою програмування, розробленою компанією Microsoft, і використовується для створення різноманітних застосунків, включаючи мобільні [15]. C# має сучасний синтаксис, хорошу продуктивність та надійність, а також забезпечує зручну роботу зі сторонніми бібліотеками. Однак, використання C# обмежується більшою мірою платформою Windows і не має такої широкої підтримки на інших платформах, зокрема на Android.

React Native є фреймворком для розробки мобільних застосунків, який використовує мову програмування JavaScript [16]. Він дозволяє розробникам створювати кросплатформні застосунки для Android та iOS, використовуючи єдиний код. React Native пропонує швидку розробку, оскільки використовує компонентний підхід. Крім того, він має велику активну спільноту розробників, що забезпечує підтримку та доступ до різних розширень. Однак, React Native може бути трохи повільнішим

порівняно з нативними рішеннями, і деякі специфічні функціональності можуть вимагати використання модулів на нативному рівні.

Kotlin же є новітньою мовою програмування, яка набула популярності для розробки мобільних застосунків на платформі Android [17]. Вона пропонує сучасний синтаксис, безпеку типів, підтримку функціонального програмування та багато інших сучасних функцій. Kotlin є попередньо налаштованою мовою для розробки Android застосунків, що дозволяє зручно і ефективно використовувати різноманітні функціональності платформи. Використання Kotlin сприяє зменшенню кількості написаного коду, полегшує підтримку та підвищує продуктивність розробки. Однак, Kotlin має меншу кількість ресурсів та матеріалів порівняно з Java, яка є більш усталеною мовою для розробки Android застосунків.

Висновуючи з усіх переваг та недоліків розглянутих мов програмування, мій вибір зупинився на мові Kotlin. Це є новітньою мовою з сучасним синтаксисом, безпекою типів та підтримкою функціонального програмування [18]. Вона має пряму підтримку для розробки мобільних додатків на платформі Android і надає багато зручних функціональностей, що полегшують розробку. Kotlin забезпечує ефективну роботу з платформою, зменшує кількість написаного коду і покращує продуктивність розробки. Незважаючи на те, що Kotlin має меншу кількість ресурсів порівняно з Java, яка є більш усталеною мовою для розробки Android додатків, вона пропонує новаторські можливості та потенціал для розвитку.

2.2.3. *Фреймворки*

Розглянемо декілька фреймворків для розробки мобільних застосунків, та опишемо, чому саме для написання ПЗ було обрано Ktor, як фреймворк для розробки серверної частини застосунку, та Jetpack Compose для створення інтерфейсу користувача.

Spring Boot є потужним фреймворком для розробки додатків на основі мови програмування Java та Kotlin [19]. Він пропонує повноцінне рішення для розробки, засноване на принципах інверсії управління та ін'єкції залежностей. Spring Boot дозволяє швидко створювати робочі застосунки, забезпечує багато вбудованих функціональностей, таких як керування транзакціями, безпекою та доступом до баз даних. Він також має широку спільноту розробників, що забезпечує підтримку, багато документації та статей. Однак, Spring Boot може бути більш важким та складним для початківців, вимагати більше конфігурації та мати більші системні вимоги.

Koin є бібліотекою для впровадження залежностей, написана на чистому Kotlin. Він пропонує простий та зрозумілий синтаксис, що полегшує роботу з ін'єкцією залежностей [20]. Koin дозволяє швидко налаштувати та управляти залежностями у застосунку. Однак, порівняно з іншими розв'язками, такими як Spring Boot, Koin може мати обмежені функціональні можливості та меншу активну спільноту, що може вплинути на доступність документації та підтримку.

Ktor є фреймворком для розробки серверних додатків на мові Kotlin. Він пропонує асинхронний та легковаговий підхід до розробки, що сприяє високій продуктивності та масштабованості. Однак, порівняно з іншими фреймворками, такими як Spring Boot, Ktor може мати меншу кількість доступних розширень та бібліотек, що може обмежити функціональні можливості та можливості розробки [21].

Для розробки даного ПЗ було обрано Ktor, адже цей фреймворк переважає завдяки своїй асинхронності, продуктивності та нативній підтримці мови Kotlin. Його легковаговість та простота використання роблять його зручним вибором для розробки серверних додатків. Крім того, Ktor надає гнучкі можливості налаштування та розширення, що дозволяє створити оптимальне середовище для проєкту.

Для розробки інтерфейсу обрано фреймворк Jetpack Compose [22], що є декларативним фреймворком для розробки користувацького інтерфейсу

на платформі Android. Він дозволяє створювати красиві та динамічні інтерфейси за допомогою декларативного синтаксису. Також Jetpack інтегрується з Android-екосистемою та надає доступ до всіх функцій і можливостей платформи Android. Вибором для написання інтерфейсу ПЗ є Jetpack за його особливість створення інтерфейсів з мінімальним кодом, що значно спрощує розробку.

Ktor сам по собі не забезпечує доступ до Jetpack Compose UI, оскільки це різні компоненти. Але можна поєднати їх використовуючи архітектурний паттерн MVVM, який описаний в наступному підрозділі. В цьому паттерні ViewModel відповідає за логіку застосунку і може взаємодіяти з базою даних, що забезпечує Room. В свою чергу, View може використовувати Jetpack Compose для створення інтерфейсу користувача [23].

Таким чином, ми можемо використовувати Jetpack Compose разом з Ktor, об'єднуючи їх за допомогою архітектурного паттерну MVVM.

2.2.4. *Шаблон проєктування*

Шаблони проєктування є важливим інструментом у світі програмування, які допомагають розробникам створювати ефективні та легко зрозумілі програмні рішення. Вони представляють собою загальноприйняті рішення для типових проблем, що виникають під час розробки програмного забезпечення.

Шаблони проєктування MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel) є широко використовуваними при розробці програмного забезпечення, включаючи мобільні застосунки [24]. Основна ідея за всіма цими шаблонами полягає в розділенні компонентів додатку для полегшення управління і підтримки. Тож розглянемо їх та опишемо детальніше обраний шаблон MVVM.

MVC:

- Model відповідає за управління даними та бізнес-логікою.
- View представляє інтерфейс для відображення даних користувачу.

- Controller обробляє вхідні дані від користувача та взаємодіє з моделлю і видом.

Недоліком MVC є складність підтримки бізнес-логіки та велика кількість залежностей між компонентами.

MVP:

- Model відповідає за дані та бізнес-логіку.
- View відображає дані та передає дії користувача до презентера.
- Presenter обробляє дії користувача, виконує бізнес-логіку та оновлює вигляд.

Один з недоліків MVP полягає в складності розробки через збільшену кількість класів і залежностей.

MVVM:

- Model представляє дані та бізнес-логіку.
- View відображає дані та реагує на зміни стану моделі.
- ViewModel підтримує зв'язок між видом та моделлю, обробляє команди від вигляду та оновлює стан моделі.

MVVM є кращим, адже він дозволяє розділити логіку відображення від бізнес-логіки, що полегшує тестування та підтримку коду. ViewModel дозволяє керувати станом продукту та забезпечує більшу гнучкість в управлінні взаємодією між Model та View. Даний шаблон підтримує двостороннє зв'язування даних, що дозволяє автоматично оновлювати View при зміні даних в ViewModel. Один з головних переваг MVVM – це можливість повторного використання коду та розширення застосунку. Кожна складова може бути розроблена окремо та легко взаємодіє з іншими частинами застосунку [25].

2.2.5. *СУБД*

У розробці програмного забезпечення збереження даних є однією з найважливіших задач. Для ефективної та надійної роботи з даними використовуються системи управління базами даних (СУБД) [26]. Реляційні

СУБД використовують табличну структуру для зберігання даних, де дані представлені у вигляді таблиць з рядками та стовпцями. Вони забезпечують зв'язки між таблицями за допомогою ключів та дозволяють виконувати складні операції, такі як фільтрація, сортування та об'єднання даних.
MySQL, PostgreSQL і SQLite є популярними СУБД, які знаходять широке застосування у розробці програмного забезпечення. Кожна з цих СУБД має свої переваги та недоліки, які розглянемо далі.

MySQL є однією з найпопулярніших СУБД у світі програмного забезпечення [27]. Ця реляційна СУБД відкритого коду володіє численними перевагами. Її основні переваги включають простоту використання та налаштування, високу продуктивність та швидкодію, масштабованість та надійність. MySQL також підтримує широкий спектр програмних мов та платформ. Недоліки MySQL включають обмежену підтримку деяких складних операцій та менше функціональних можливостей порівняно з деякими іншими СУБД.

PostgreSQL – потужна та розширювана СУБД з високою надійністю та широкими можливостями реляційної моделі. Вона надає розширені можливості для роботи зі складними структурами даних та дозволяє розробникам створювати власні функції та типи. PostgreSQL також відомий своєю високою надійністю, забезпечуючи ACID-властивості та механізми резервного копіювання та відновлення. Недоліками PostgreSQL є вищі вимоги до ресурсів та складність налаштування порівняно з іншими СУБД [28].

SQLite – це легкий вбудовуваний SQL двигун баз даних, який забезпечує надійність і швидкість. SQLite є частиною стандартної бібліотеки Android, тому його використання є простим і зручним. Він дозволяє створювати бази даних та виконувати операції з ними, такі як створення, оновлення, видалення, вибірка даних, за допомогою структурованого запиту SQL [29].

Зважаючи на проведений аналіз, SQLite має декілька переваг, особливо для простих проектів або мобільних застосунків:

- SQLite не вимагає окремого сервера або налаштувань. Запуск і розгортання бази даних є простим процесом.
- База даних зберігається у вигляді одного файлу, що дозволяє легко розповсюджувати та управляти нею.
- SQLite зазвичай працює швидше ніж MySQL або PostgreSQL, оскільки база даних існує локально, а не на віддаленому сервері.

SQLite є вигідним вибором для розробки даного ПЗ, завдяки своїм універсальним характеристикам та простоті використання [30]. Він є легким, портативним та не вимагає окремого сервера баз даних, що робить його ідеальним для вбудованих систем, мобільних додатків та прототипування. SQLite підтримує майже всі основні функції реляційних баз даних, забезпечує ефективність та надійність, а також простоту у використанні через його SQL-сумісний інтерфейс.

2.3. Висновки

На основі аналізу вимог до функціональності програмного забезпечення та аналізу обраних інструментів було прийнято рішення використовувати мову програмування Kotlin, фреймворк Ktor, архітектурний паттерн MVVM та базу даних SQLite.

Архітектуру розроблювальної системи можна переглянути на рис. 2.1.

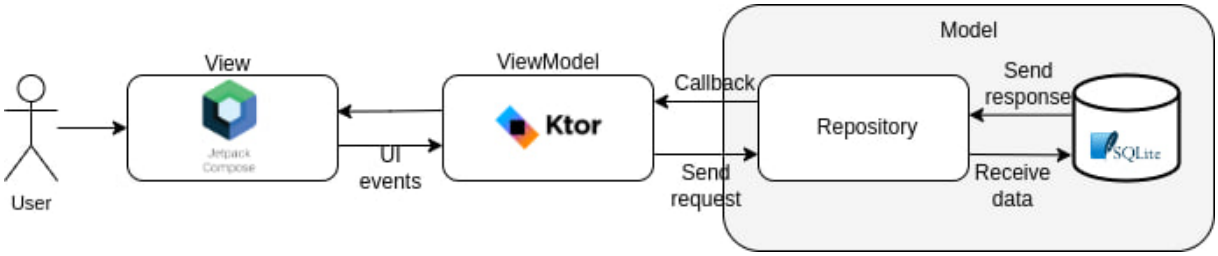


Рис. 2.1. Архітектура розроблювальної системи

Мова програмування Kotlin має багато переваг, таких як безпеку типів, інтероперабельність з Java, багатий набір бібліотек та інструментів, що дозволяє швидко та ефективно розробляти застосунки.

Фреймворк Ktor було обрано через його легкість та швидкість. Зокрема, він має просту інтеграцію з Kotlin та має маленький розмір, що робить його швидким та ефективним. Використання ж Jetpack Compose дозволяє швидко та ефективно створювати привабливий та зручний користувацький інтерфейс, а також забезпечує більш просту та ефективну роботу з даними, що робить його відмінним вибором для нашого проєкту.

Архітектурний паттерн MVVM обраний з метою розділення бізнес-логіки та представлення даних. Він дозволяє зберігати код бізнес-логіки в одному місці, що полегшує його тестування та управління. Крім того, з MVVM можна легко забезпечити зв'язок між різними частинами застосунку та забезпечити швидкий розвиток.

Як БД для застосунку було обрано SQLite з причини його швидкості, ефективності та надійності. SQLite є однією з найпоширеніших та найкращих баз даних для мобільних пристроїв.

В цілому, обрані інструменти та технології дозволяють створити ефективне та функціональне ПЗ для відслідковування емоцій користувачів під час виконання різних завдань.

3. ОГЛЯД РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Загальний опис системи

Розроблена система є мобільним застосунком, призначеним для відслідковування та аналізу емоцій користувачів під час виконання завдань. Вона надає можливість користувачам записувати свої емоційні стани, відстежувати виконані завдання та отримувати звіти та графіки, що відображають залежність між їхнім емоційним станом та виконаними завданнями. Застосунок також надає можливість отримання рекомендацій для покращення емоційного благополуччя.

Відповідно до функціональних вимог проєкту, сформульованих в підрозділі 2.1, можемо виділити такі ролі, як зареєстрований і не зареєстрований користувач.

Незареєстрований користувач має:

- створити акаунт у застосунку, вводячи необхідні особисті дані та обираючи унікальне ім'я користувача і пароль;
- увійти в систему під зареєстрованими даними.

Зареєстрований користувач має:

- обрати тип завдання кожного дня;
- відстежувати свої емоційні стани, занотовуючи свої емоцій та почуття;
- переглядати свою історію емоційних станів;
- мати можливість переглядати аналіз у вигляді графіків, що відображають залежність між його емоційним станом та днем виконання;
- отримати рекомендації для покращення емоційного благополуччя.

На рис. 3.1 наведено діаграму варіантів використання ПЗ для користувача.

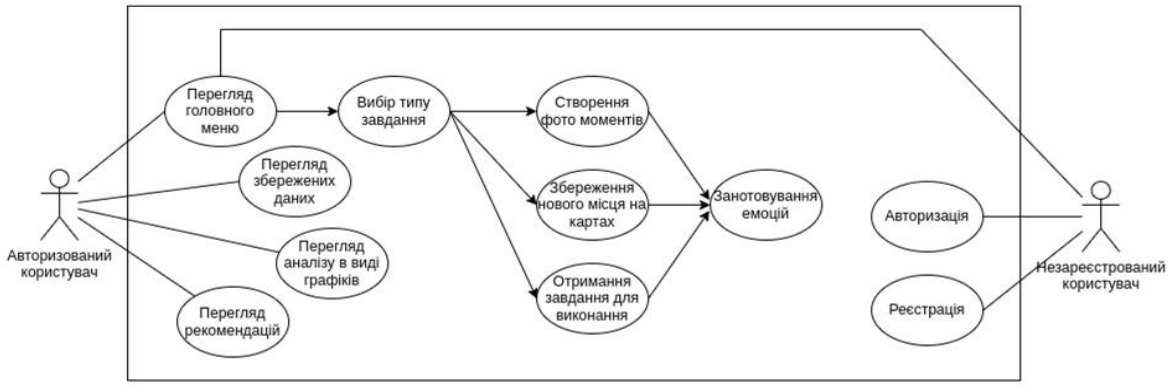


Рис. 3.1. UML-діаграма функціональних можливостей системи

Після реєстрації чи авторизації користувач потрапляє на головний екран з вибором типу завдання. Зробивши вибір, він отримує завдання та спливаюче вікно з можливістю записати свої емоції та почуття при виконанні завдання. Також користувач має можливість перегляду збережених даних та графіків з аналізом емоційного стану. З головного меню користувач може перейти на сторінку з рекомендаціями щодо покращення емоційного стану та благополуччя в цілому.

Основний сценарій використання системи з точки зору виконавця включає наступні кроки (рис. 3.2). Після запуску програми, виконавець реєструється або увійшов до існуючого облікового запису. Після успішного входу в систему, користувач може почати виконувати свої завдання.

Застосунок надає можливість вводити та записувати свої емоції після виконання завдань. Виконавець може вибирати емоції зі списку або вводити власні. Після запису емоцій, система зберігає цю інформацію у базі даних для подальшого аналізу. Застосунок також надає рекомендації щодо поліпшення емоційного фону.

Враховуючи цей основний сценарій користувач системи, виконавець може отримувати підтримку та контроль над своїми емоціями під час виконання завдань, а також знаходити нові способи розвитку та покращення свого емоційного стану.

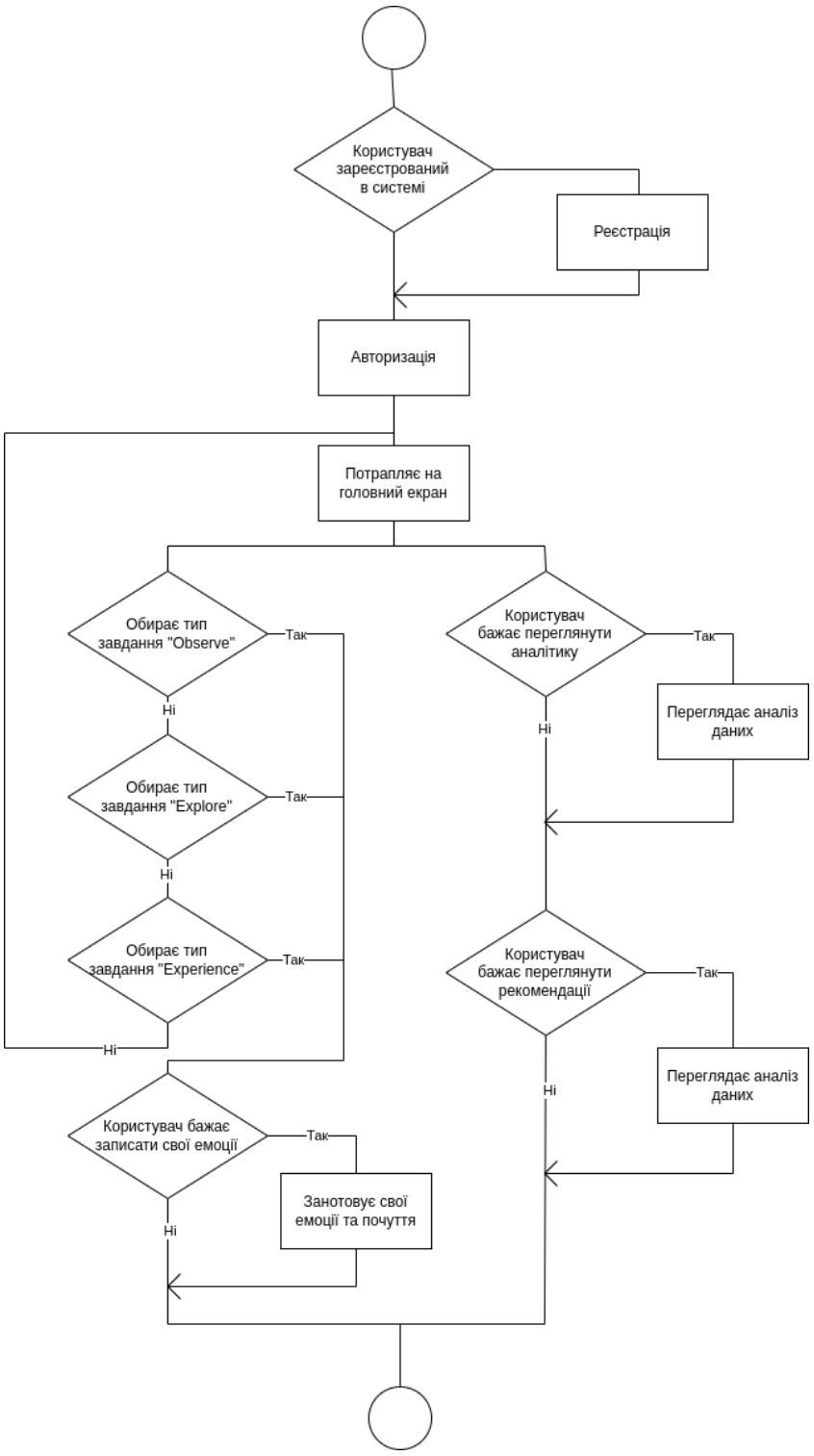


Рис. 3.2. Діаграма діяльності використання ПЗ користувачем

3.2. Структура ПЗ

Застосунок можна умовно розділити на два компоненти: клієнтську (фронтенд) та серверну (бекенд) частини. Кожна з цих частин відповідає за свою логіку в системі, проте вони тісно взаємодіють між собою.

Бекенд виконує наступні функції в даному ПЗ:

1. Автентифікація та авторизація користувачів: обробка запитів аутентифікації та авторизації користувачів, перевірка їхніх облікових даних.
2. Зберігання та управління даними користувачів: створення, зберігання та оновлення даних користувачів в базі даних. Це включає інформацію про їхні емоційні стани, виконані завдання та інші відповідні дані.
3. Обробка запитів від фронтенду, таких як запис емоційного стану користувача, запит на отримання звітів або графіків, оновлення даних тощо. Він обробляє ці запити, взаємодіє з базою даних, проводить розрахунки та повертає відповіді фронтенду з необхідною інформацією.
4. Генерація звітів про емоційний стан користувачів , виконані завдання та інші фактори. На основі цього аналізу генеруються звіти, графіки, які можуть бути використані фронтендом для відображення користувачам.

Розглянемо функціонал фронтенду:

1. Реєстрація та вхід в систему: можливість новим користувачам зареєструватися в системі шляхом введення облікових даних та інформації профілю. Крім того, він також надає можливість входу в систему для зареєстрованих користувачів, які вводять свої облікові дані для отримання доступу до функціоналу застосунку.
2. Запис емоційного стану: інтерфейс для записування їхніх емоційних станів. Це може включати вибір іконок та текстові описи. Записані емоційні стани передаються на бекенд для збереження в базі даних.
3. Перегляд історії емоційних станів, що може включати відображення списку записів з емоціями та відповідними датами.

4. Отримання звітів та графіків: надання користувачам можливості отримати звіти та графіки, які відображають залежність між їхніми емоційними станами та виконанням завдань. Це дозволяє користувачам отримати візуальне представлення своїх емоційних тенденцій та зробити аналіз свого емоційного благополуччя.
5. Рекомендації та поради: надання користувачам персоналізованих рекомендацій та порад на основі їхніх зібраних даних про емоційний стан. Наприклад, на основі аналізу даних система може рекомендувати користувачеві активності або вправи для поліпшення настрою або зняття стресу.

Фронтенд та бекенд у даному застосунку працюють взаємодоповнюючи, щоб забезпечити користувачам повноцінний функціонал. Фронтенд відповідає за взаємодію з користувачем, надаючи зручний інтерфейс для реєстрації, входу, запису емоційних станів, перегляду історії, отримання звітів та рекомендацій. Бекенд відповідає за обробку запитів, збереження та управління даними користувачів. Він забезпечує безпеку, аутентифікацію та авторизацію користувачів, аналізує зібрані дані та генерує звіти.

Працюючи разом, фронтенд та бекенд створюють потужну систему відслідковування емоцій, яка допомагає користувачам бути більш освіченими в психології, розуміти вплив емоційного фону на їхнє благополуччя та приймати відповідні дії для поліпшення свого емоційного стану.

3.3. Структура бази даних

В якості основної бази даних було обрано SQLite. На рис. 3.3 зображено ERD-діаграму бази даних системи.

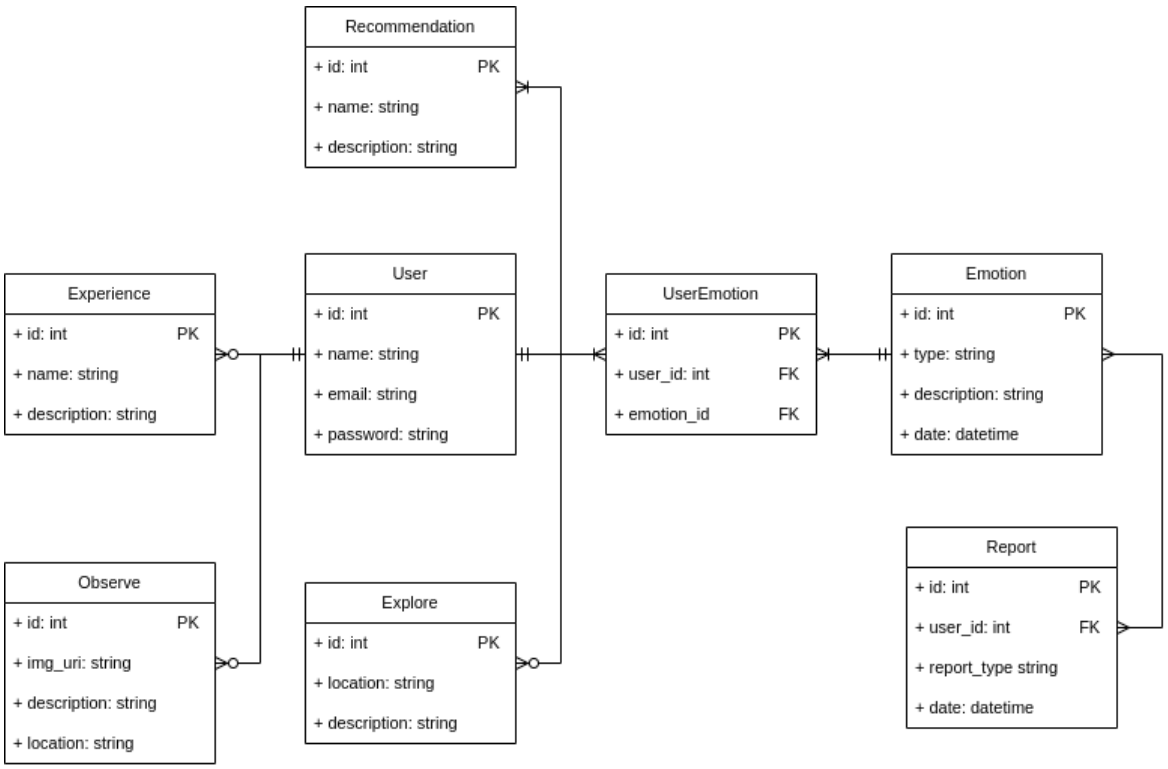


Рис. 3.3. ERD-діаграма бази даних

Далі розпишемо детально кожну з таблиць СУБД:

Таблиця "User" містить наступні дані про користувача:

- id: унікальний ідентифікатор користувача;
- ім'я: ім'я користувача;
- email: адреса електронної пошти користувача;
- пароль: захешований пароль користувача.

Таблиця "Emotion" містить дані про емоції:

- id: унікальний ідентифікатор емоції;
- type: тип або назва емоції (наприклад, щастя, смуток, стрес);
- description: детальний і більш розширений опис того, що відчуває користувач;
- date: дата, коли була зафіксована емоція.

Таблиця "UserEmotion" містить дані про емоції користувача:

- id: унікальний ідентифікатор емоції;
- user_id: зовнішній ключ, що посилається на id користувача з таблиці " User";

- emotion_id: зовнішній ключ, що посилається на id емоції з таблиці "Emotions";

Таблиця “Experience” зберігає дані про одне з трьох типів завдання:

- id: унікальний ідентифікатор завдання;
- name: назва завдання;
- description: детальний опис завдання.

Таблиця “Explore” зберігає дані про одне з трьох типів завдання:

- id: унікальний ідентифікатор завдання;
- location: місцезнаходження користувача;
- description: детальний опис завдання.

Таблиця “Observe” зберігає дані про одне з типів завдання:

- id: унікальний ідентифікатор завдання;
- img_uri: силку на збережене фото;
- description: детальний опис фото;
- location: місцезнаходження збереженого фото;

Таблиця "Report" зберігає інформацію про аналіз даних:

- id: унікальний ідентифікатор звіту;
- user_id: зовнішній ключ, що посилається на id користувача з таблиці " User";
- report_type: тип або назва звіту (наприклад, графік емоцій за часом);
- date: дата, коли був згенерований звіт.

Таблиця "Recommendation" зберігає інформацію про рекомендації:

- id: унікальний ідентифікатор рекомендації;
- name: назва рекомендації;
- description: детальний опис.

4. АНАЛІЗ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації застосунку

Відповідно до вимог до ПЗ, які були розглянуті в підрозділі 1.3, було обрано наступні особливості реалізації даного застосунку:

1. Застосунок повинен забезпечувати можливість користувачам записувати свої емоційні стани за допомогою віджетів або вручну вводити дані. Для цього можуть використовуватися різні формати вводу, такі як вибір зі списку емоцій або власноручне введення текстового опису.
2. Застосунок повинен надавати можливість отримувати звіти та аналізувати дані про залежність між їхнім емоційним станом та виконаними завданнями. Даний функціонал можливий за допомогою створення графіків, діаграм та інших візуальних засобів для відображення даних.
3. Застосунок може надавати рекомендації та поради щодо покращення їхнього емоційного стану. Це може включати рекомендації щодо виконання певних завдань, практикування медитації або відпочинку, підказки щодо збалансованого харчування та фізичної активності.
4. Застосунок повинен мати привабливий дизайн, зручну навігацію та логічну організацію функцій, що дозволяють користувачам легко взаємодіяти з продуктом та виконувати потрібні дії без зайвих зусиль. Забезпечення зручного інтерфейсу допомагає залучати та утримувати користувачів, покращує їхнє враження від використання застосунку та сприяє досягненню поставлених цілей. Це буде розглянуто більш детально в підрозділі 4.6.
5. Важливим етапом розробки застосунку є його тестування для виявлення та усунення помилок. Про тестування ПЗ поговоримо в підрозділі 4.7. Крім того, після випуску додатку до користувачів,

важливо забезпечити його підтримку та вирішення проблем, які можуть виникнути.

6. При розробці застосунку слід враховувати можливість його масштабування для відповіді на зростаючу кількість користувачів. Додаток повинен бути доступним з різних платформ, таких як мобільні пристрої та веб-браузери, забезпечуючи максимальну доступність для користувачів. Саме такі можливі подальші вдосконалення будуть розкриті більш широко в підрозділі 4.8.

4.2. Дизайн та інтерфейс користувача

При вході в застосунок, користувач потрапляє на сторінку авторизації (рис. 4.1) , або реєстрації, у разі відсутності акаунту. При введенні даних є перевірка на правильність введення email, а також перевірка на складність паролю. При неправильно введених даних буде видана помилка.

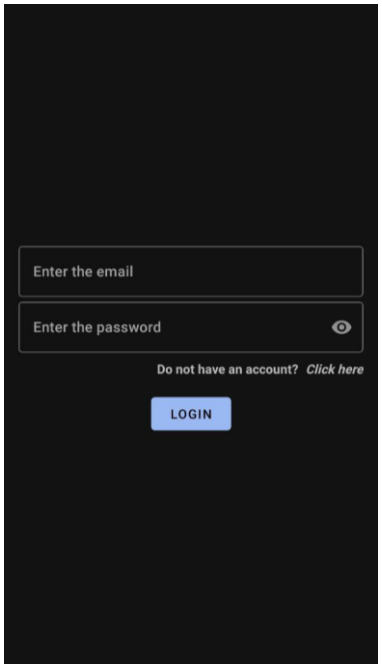


Рис. 4.1. Сторінка авторизації

Після етапів реєстрації та/або авторизації користувач потрапляє на сторінку головного меню (рис. 4.2), де він має можливість обрати тип завдання на сьогодні.

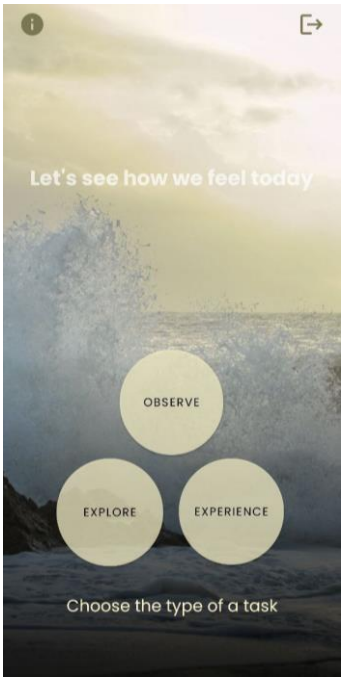


Рис. 4.2. Головне меню

При виборі першого завдання “Observe” користувач отримує повідомлення про завдання зробити або завантажити фото та занотувати, які виникли емоцій в той момент (рис. 4.3). Таким чином, є можливість скористатися камерою або обрати фото з галереї та опинитись на новій сторінці з обраним фото для збереження моменту. Збережені фото з відповідними даними зберігаються в БД застосунку. Самі фото, їх назву, локацію та дату створення можна переглянути на наступній сторінці. Також їх можна видаляти по бажанню.

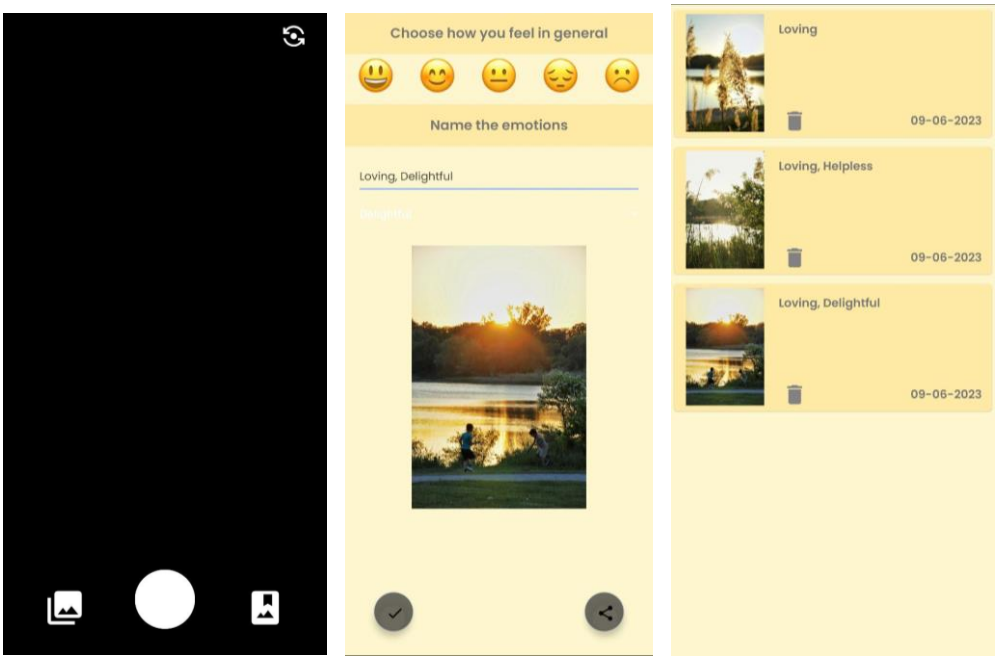


Рис. 4.3 Функціональні можливості типу завдання “Observe”

При виборі опції “Explore” користувачу пропонується відвідати нове місце цього дня (це може бути кімната в університеті, магазин, вулиця), де він ніколи не був раніше. Нове місце можна обрати на карті та записати емоції, які виникли при відвідуванні цього місця, у новому вікні (рис. 4.4).

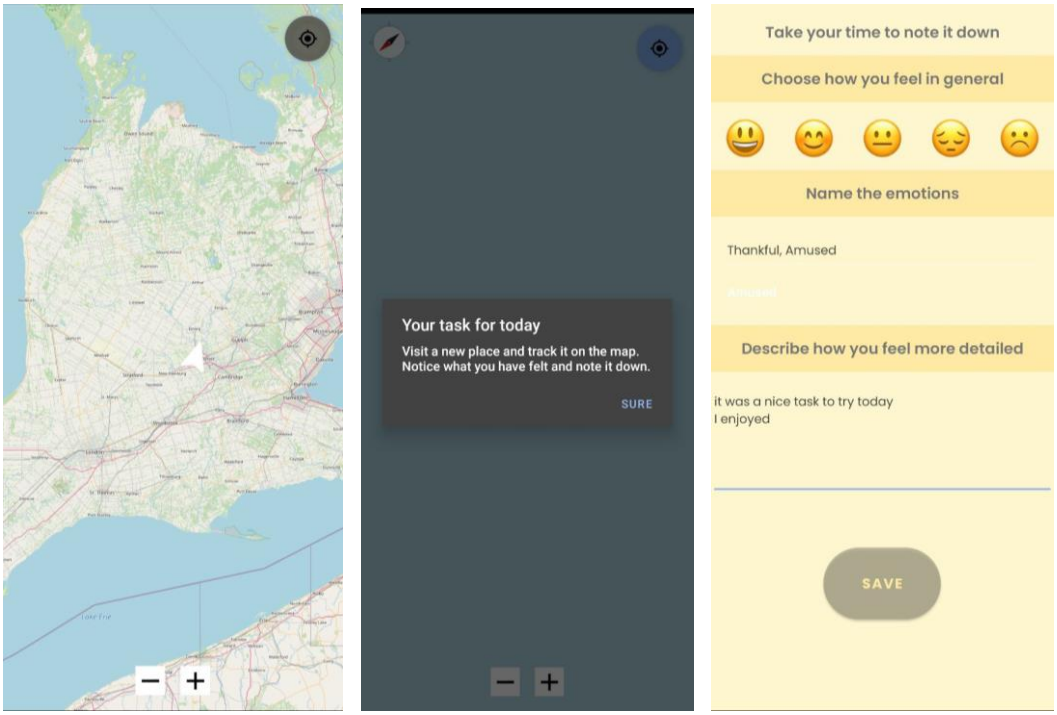


Рис. 4.4. Функціональні можливості типу завдання “Explore”

При виборі третього типу завдання “Experience”, користувачу буде надано випадкове завдання для виконання протягом дня (рис. 4.5). Наприклад: потанцювати під улюблену пісню, намалювати картину пензлем, приготувати щось солодке, помедитувати, прослухати певний трек, тощо. І записати емоції та почуття, що виникли при виконання цих завдань.

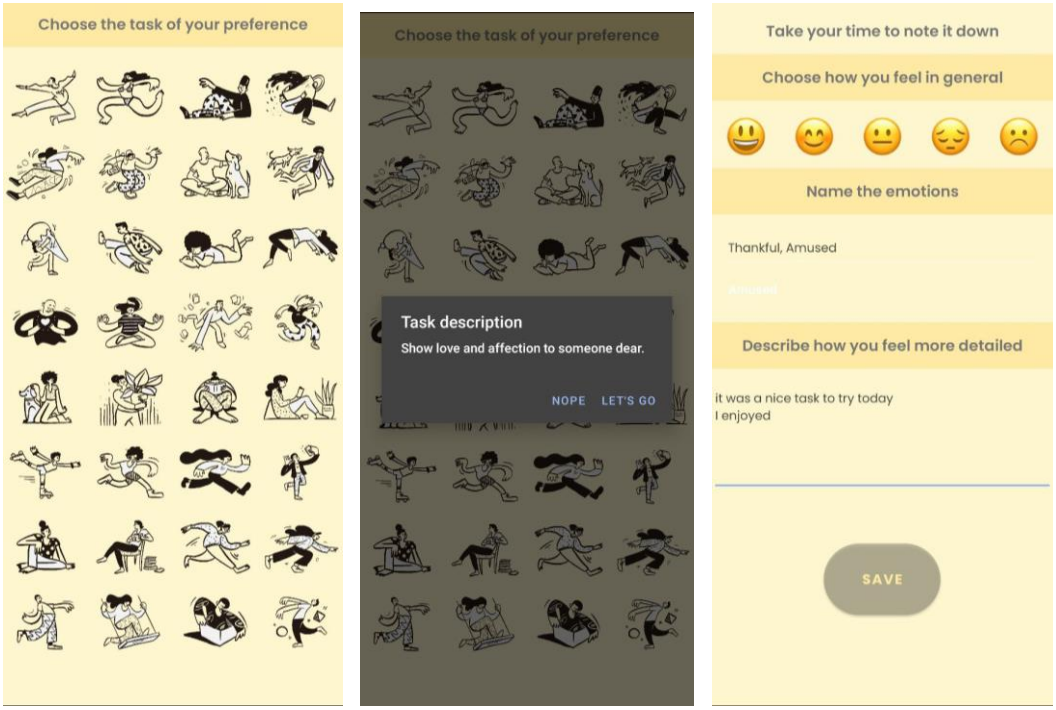


Рис. 4.5. Функціональні можливості типу завдання “Experience”

Записані дані будуть збережені та відображені у виді графіка який користувач має можливість відкрити з головного меню (рис. 4.6). На графіку зображено середнє статистичне від настрою користувача кожного дня тижня. Це може бути цінним інструментом для самоаналізу та рефлексії.

Також користувач може скористатись після аналізу результатів графіка такою функціональною можливістю, як перегляд рекомендацій щодо покращення емоційного стану та благополуччя в цілому.

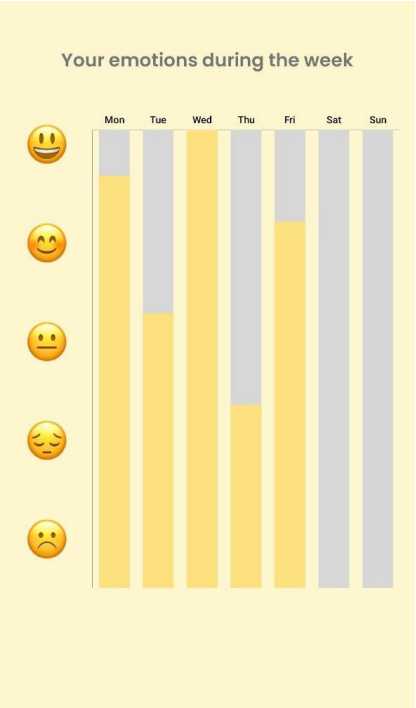


Рис. 4.6. Графік з аналізом емоцій користувача за тиждень

4.3. Тестування застосунку

Під час розробки застосунку про емоції було приділено значну увагу тестуванню, що допомагає забезпечити якість та надійність програмного забезпечення. Тестування включає модульне тестування, інтеграційне тестування та ручне тестування різних частин застосунку.

Нижче наведені основні типи тестів та їх опис:

1. Модульне тестування:
   - Аутентифікація: включає перевірку правильності реєстрації нового користувача, входу в систему зареєстрованих користувачів та перевірку різних сценаріїв входу з неправильними обліковими даними.
   - Запис емоцій: включає тестування збереження емоційних записів користувача, перевірку правильності обробки та відображення цих записів.
2. Інтеграційне тестування:
   - Менеджер завдань та запис емоцій: тестує взаємодію між модулями менеджера завдань та запису емоцій, включаючи

обробку інформації про виконання завдань та оновлення емоційних записів.

- Запис емоцій та звіти: перевіряє взаємодію між модулями запису емоцій та звітів, включаючи створення звітів на основі інформації про емоційний стан користувача.

3. Ручне тестування:

- Інтерфейс користувача: перевірка правильності відображення інтерфейсу користувача, включаючи переходи між сторінками, відображення тексту, кнопок та інших елементів інтерфейсу.
- Запис емоцій: перевірка можливості внесення емоційних записів вручну або вибору з наданого списку, перевірка коректності відображення даних про емоційний стан користувача.
- Звіти: Перевірка правильності генерації звітів на основі зібраних даних, перевірка точності відображення графіків та статистики емоційного стану користувача.

Далі (табл. 4.1) подані дані про тест-кейси для користувацького модулю.

Таблиця 4.1

Тестування функціональних можливостей програмного забезпечення

Назва тестування	Виконані дії
Реєстрації нового користувача	● перевірка правильності заповнення обов'язкових полів форми реєстрації; ● перевірка валідації введених даних; ● перевірка створення нового користувача в базі даних;
Вхід в систему	● перевірка валідації введених облікових даних; ● перевірка успішного входу користувача в систему;

Продовження табл. 4.1

Функціональні можливостей завдань	● вибір завдання та перевірка його відображення; ● редагування існуючого завдання та перевірка оновлення його даних; ● видалення завдання та перевірка його відсутності у списку завдань;
Функціональні можливості запису емоцій	● внесення нового емоційного запису та перевірка його відображення; ● редагування існуючого емоційного запису та перевірка оновлення його даних; ● видалення емоційного запису та перевірка його відсутності;
Тестування звітів	● генерація звіту на основі зібраних даних та перевірка правильності відображення статистики емоційного стану користувача; ● перевірка точності відображення графіків та діаграм в звітах; ● перевірка правильності фільтрації та сортування даних у звітах;
Тестування збереження даних	● введення та збереження емоційного стану, включаючи перевірку правильного вибору емоції та збереження даних; ● введення та збереження фотографій, при виборі завдання, що потребує збереження зображення; ● введення та збереження дати, пов'язаної з емоційним записом, і перевірка правильності її збереження та відображення.

В ході тестування здійснювалися додаткові тест-кейси, щоб переконатися в коректній роботі всіх основних функцій та складових частин застосунку. Вони допоможуть забезпечити високу якість та надійність застосунку та виявити можливі помилки.

4.4. Рекомендації щодо подальшого вдосконалення

Зробивши аналіз даного ПЗ та звернувши увагу на швидкий розвиток технологій, а також на аналоги застосунку, які вже мають перевагу на ринку, вкажемо рекомендації щодо подальшого вдосконалення застосунку.

Розширення функціональності підвищать цінність застосунку для користувачів. Наприклад, можливість додавання сповіщень або нагадувань для виконання завдань, зберігання та аналізування інших типів даних (наприклад, фізична активність або сон) стануть у нагоді при щоденному використанні застосунку. Поліпшення інтерфейсу є також важливим удосконаленням. Звернувши увагу на ергономіку, використання кольорів, шрифтів і іконок, а також на навігацію та організацію контенту, можна зробити більш інтуїтивно зрозумілий, зручний і привабливий для користувачів інтерфейс.

Можливе таке удосконалення, як розширення функціональності аналітики задля надання можливості користувачам отримувати більше інсайтів і статистики щодо залежності між їхнім емоційним станом та виконанням завдань. Додаткові графіки, діаграми або звіти, які допоможуть користувачам краще розуміти їхні емоції та зміни в них завжди будуть в нагоді. Додавання підтримки різних мов забезпечить доступність для більш широкої аудиторії користувачів.

Важливим кроком в покращенні продукту є його оптимізація. Це може включати покращення швидкості завантаження сторінок, оптимізацію запитів до бази даних, кешування даних або використання механізмів асинхронного завантаження. Варто ще надати користувачам доступ до технічної підтримки, яка може допомогти їм у разі виникнення проблем або питань щодо використання застосунку.

Такі удосконалення як можливість моніторингу використання та аналітики, тестування, оновлення та виправлення помилок, забезпечить неодмінне покращення якості застосунку і підвищить рівень задоволення користувачів.

ВИСНОВКИ

Зважаючи на розглянуті аспекти та використані інструменти у розробці програмного забезпечення для відслідковування емоцій, можна зробити наступні висновки.

В даному дипломному проєкті було успішно реалізовано мобільний застосунок, який забезпечує моніторинг та аналіз емоцій користувача під час виконання щоденних завдань. Для розробки застосунку було використано мову програмування Kotlin, що забезпечило ефективність та зручність у роботі.

З метою забезпечення зручного та ефективного взаємодії з базою даних, було використано SQLite разом з фреймворком Room. Це дозволило зберігати та обробляти дані з максимальною швидкістю та надійністю, а також забезпечило легкість розробки завдяки вбудованому ORM-підходу. Окрім того, для розробки візуально привабливого інтерфейсу користувача, було використано Jetpack Compose. Цей інструментарій надає багато можливостей для швидкого створення і кастомізації елементів інтерфейсу, а також для зручної обробки подій та взаємодії з користувачем.

В результаті розробки застосунку застосовані інструменти та технології дозволили створити функціональний та зручний продукт, який допомагає користувачам відслідковувати їхні емоції та аналізувати їх в контексті щоденних завдань. Процес розробки дав змогу поглибитись у сферу аналізу емоцій, розробки інтерфейсу користувача та роботи з базою даних.

У цілому, дипломний проєкт дозволяє успішно реалізувати застосунок для відслідковування емоцій, що надає користувачам можливість пізнати краще свої уподобання щодо щоденних завдань та свій емоційний стан. Ці та інші можливості, що надає дане ПЗ, безпосередньо впливають на підвищення самосвідомості та покращення благополуччя.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

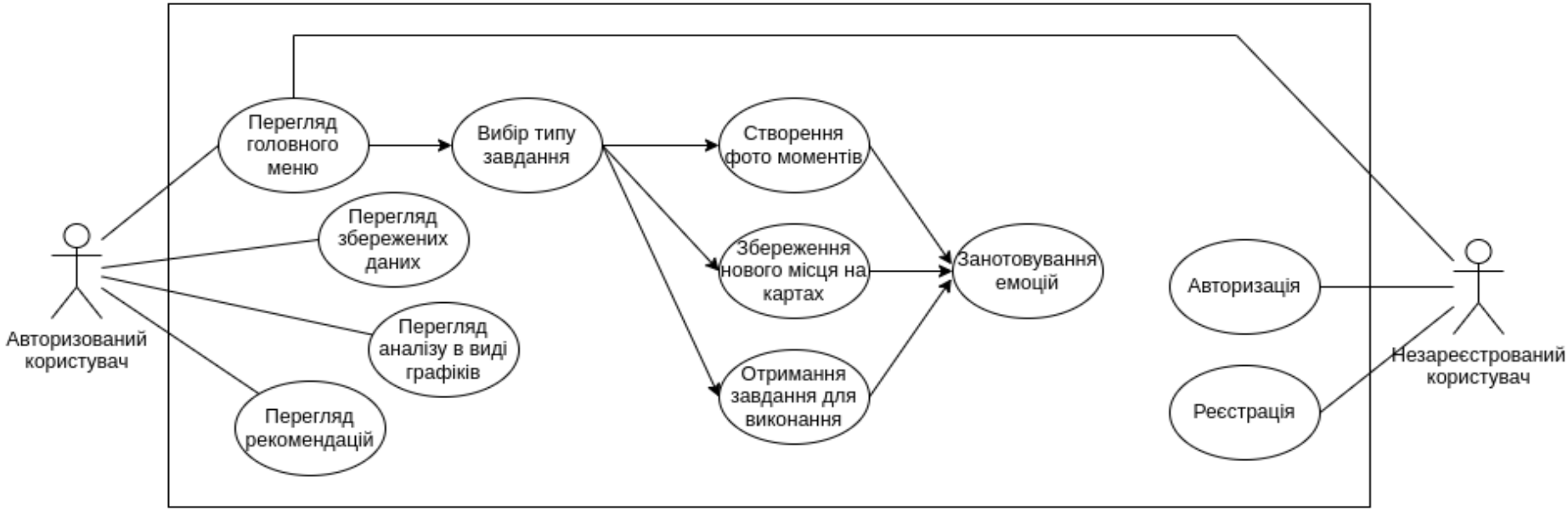
1. Can We Increase Psychological Well-Being [Електронний ресурс]. – Режим доступу: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0158092
2. 3 ways to better understand your emotions [Електронний ресурс]. – Режим доступу: https://hbr.org/2016/11/3-ways-to-better-understand-your-emotions
3. 10 things to try [Електронний ресурс]. – Режим доступу: https://psychcentral.com/depression/daily-routine-for-depression
4. Емоційний стан [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/
5. Самосвідомість [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/
6. Recognizing and managing emotions [Електронний ресурс]. – Режим доступу: https://www.skillsyouneed.com/ps/managing-emotions.html
7. 5 ways to know your feelings better [Електронний ресурс]. – Режим доступу: https://kidshealth.org/en/teens/emotional-awareness.html
8. VOS [Електронний ресурс]. – Режим доступу: https://vos.health/about
9. Daylio [Електронний ресурс]. – Режим доступу: https://daylio.net/
10. Reasons to step outside of the comfort zone [Електронний ресурс]. – Режим доступу: https://www.huffpost.com/entry/stepping-outside-your-comfort-zone
11. Зона комфорту [Електронний ресурс]. – Режим доступу: http://psychologis.com.ua/zona_komforta.htm
12. 8 Reasons You Must Go for an Android First Development [Електронний ресурс]. – Режим доступу: https://www.learntek.org/blog/android-first-development/
13. 10 things iOS does better than Android [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/ios-vs-android-1068950/

14. Java [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Java
15. Is C# good for mobile development? [Електронний ресурс]. – Режим доступу: https://www.quora.com/Is-C-good-for-mobile-development
16. Розробка мобільних додатків на React Native [Електронний ресурс]. – Режим доступу: https://brander.ua/what-we-offer/application-development/rozrobka-mobilnykh-dodatkiv-na-react-native
17. Kotlin [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Kotlin_(programming_language)
18. 7 Reasons Why Kotlin Is Your Best Bet for Android App Development [Електронний ресурс]. – Режим доступу: https://successive.tech/blog/kotlin-for-android-app-development/
19. 10 Reasons Why You should use Spring Boot [Електронний ресурс]. – Режим доступу: https://springhow.com/why-use-spring-boot/
20. Koin [Електронний ресурс]. – Режим доступу: https://insert-koin.io/docs/setup/why/
21. Ktor [Електронний ресурс]. – Режим доступу: https://ktor.io/docs/welcome.html
22. Jetpack Compose Tutorial [Електронний ресурс]. – Режим доступу: https://developer.android.com/jetpack/compose/tutorial
23. Implementing MVVM in Android with Jetpack [Електронний ресурс]. – Режим доступу: https://sphericalkat.medium.com/
24. MVC vs MVP vs MVVM – What is the difference? [Електронний ресурс]. – Режим доступу: https://www.inapps.net/mvc-vs-mvp-vs-mvvm-what-is-the-difference/
25. MVVM [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm
26. СУБД [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/

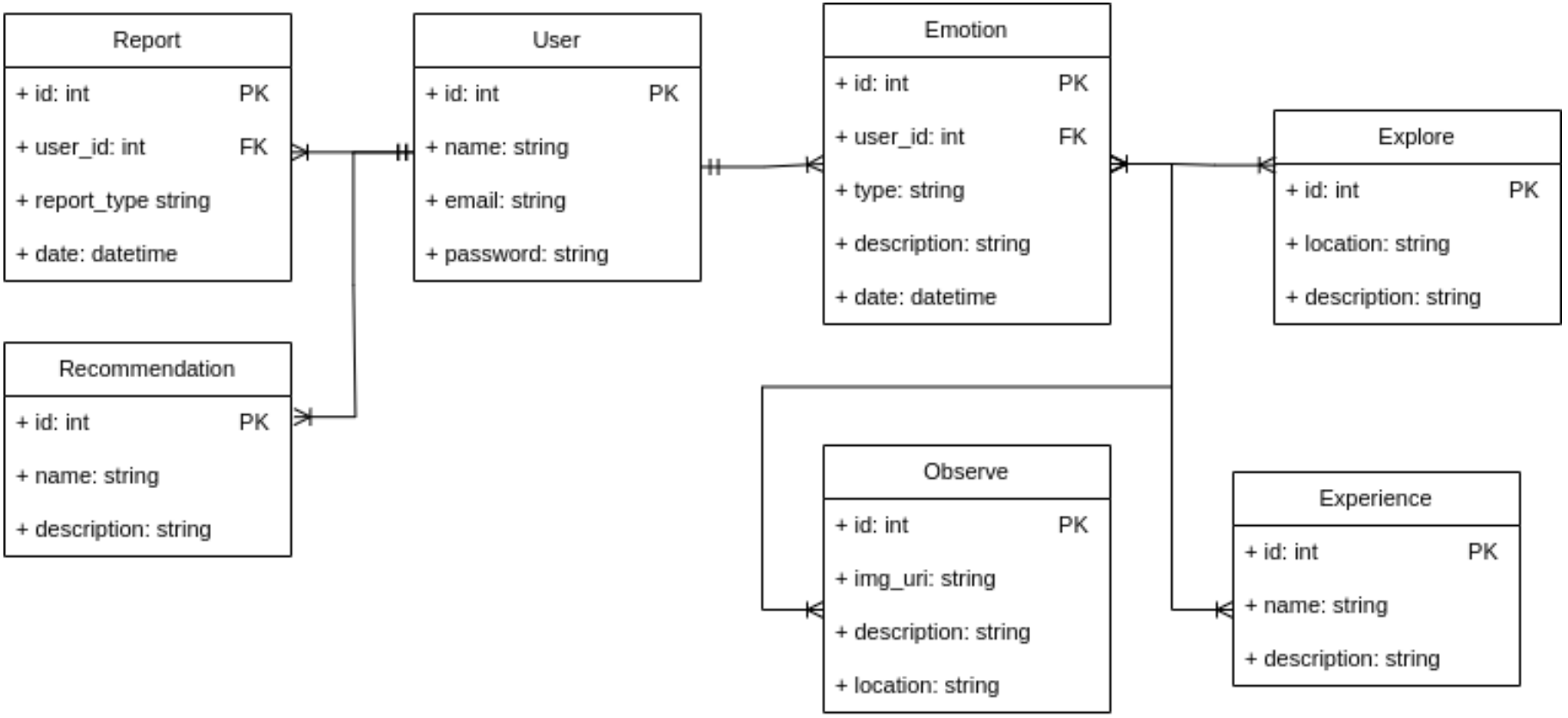
27. MySQL [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/MySQL
28. PostgreSQL [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/PostgreSQL
29. SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html
30. Why you should use SQLite [Електронний ресурс]. – Режим доступу: https://www.infoworld.com/article/3331923/why-you-should-use-sqlite.html

ДОДАТКИ

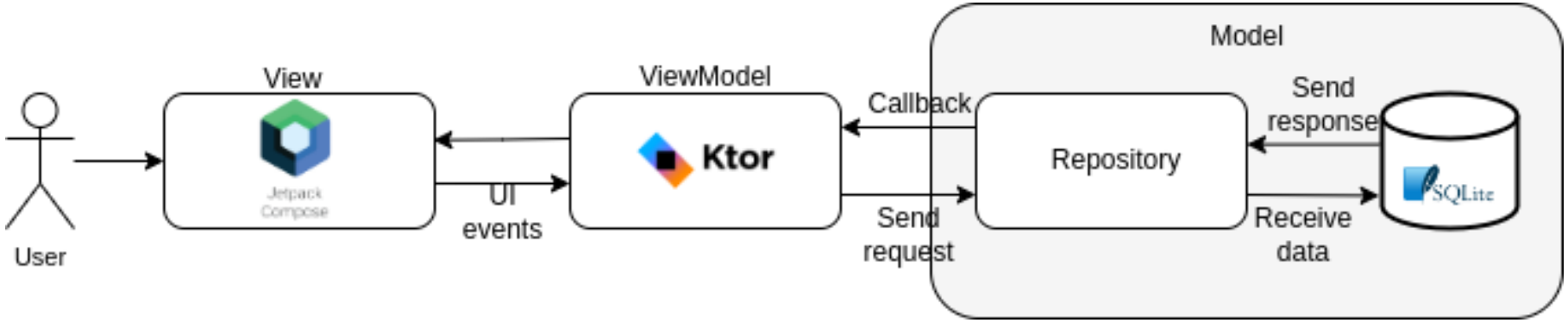
Додаток 1
Копії графічних матеріалів



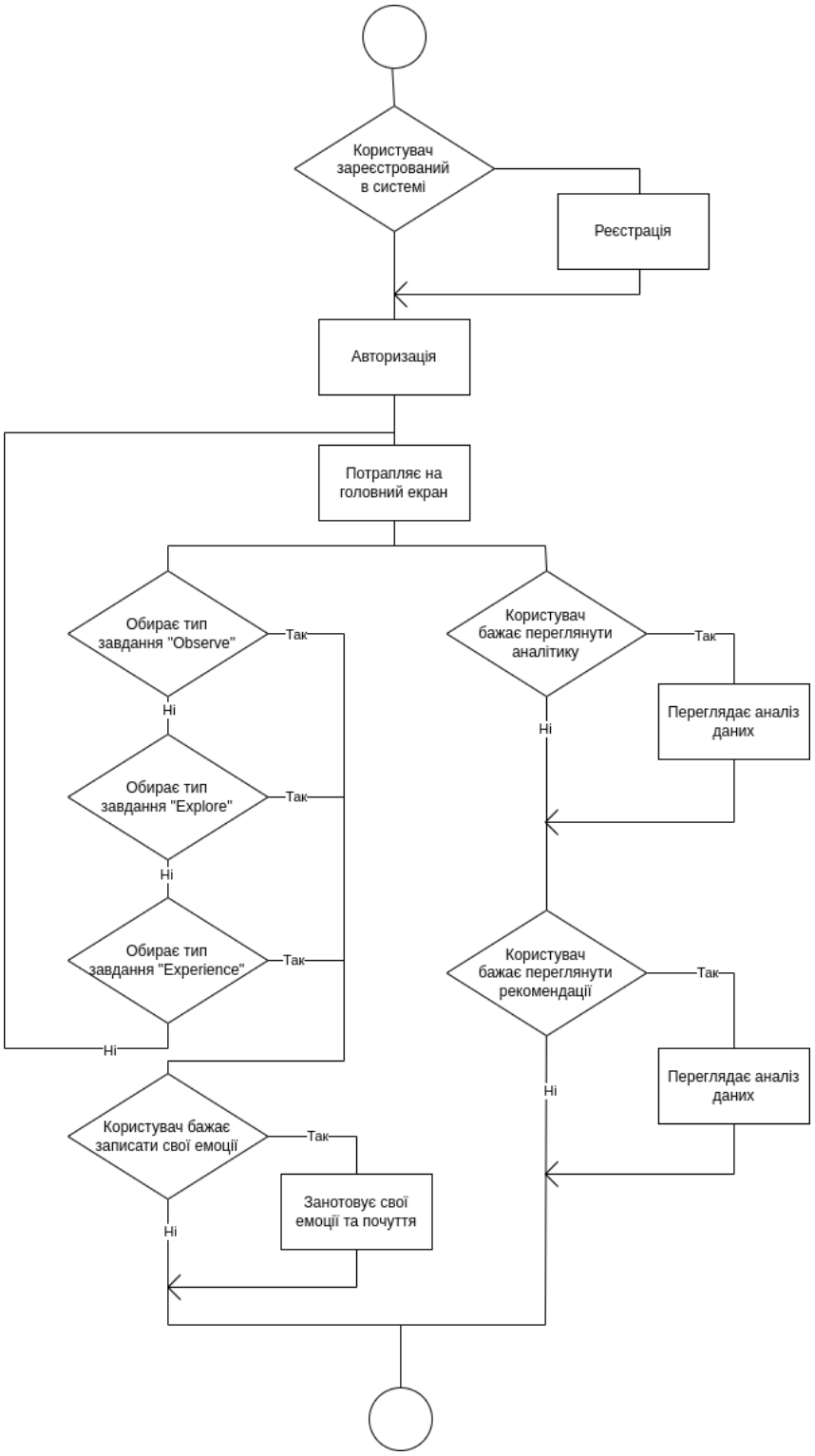
ДП.045490-06-99
Програмне забезпечення для моніторингу емоцій
Функціональні можливості системи. UML-діаграма



ДП.045490-07-99
Програмне забезпечення для моніторингу емоцій
Структура бази даних. ER-діаграма



Архітектура розроблювальної системи
Дорошенко С.О., група КП-91



Діаграма діяльності використання ПЗ користувачем
Дорошенко С.О., група КП-91

Додаток 2
Лістинг програми

Фрагменти коду backend

```
import android.app.AlertDialog
import android.content.DialogInterface
import android.content.Intent
import android.net.Uri
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import pt.ipt.finalproject.CameraActivity
import pt.ipt.finalproject.MainActivity
import pt.ipt.finalproject.MapTracking
import pt.ipt.finalproject.R
import pt.ipt.finalproject.models.Moment
import pt.ipt.finalproject.utilities.Constant

class MomentsAdapter(private var list: ArrayList<Moment>) :
   RecyclerView.Adapter<MomentsAdapter.MyViewHolder>() {


   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
       val view =

LayoutInflater.from(parent.context).inflate(R.layout.saved_pic_layout,
parent, false)
       return MyViewHolder(view)
   }
   override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
       val pos = list[position]
       holder.apply {
           val cxt = this.view.context
           val id = pos.id
           val imgUri = pos.ImgUri
           val description = pos.description
           val date = pos.date
           val location = pos.location

           ivImg.setImageURI(Uri.parse(imgUri))
           tvDescription.text = description
           tvDate.text = date


           but.setOnClickListener {
               Constant.helper.deleteMoment(id)
               val intent = Intent(cxt, MainActivity::class.java)
               cxt.startActivity(intent)
           }
           view.setOnClickListener {
               val intent = Intent(cxt, MapTracking::class.java)
               intent.putExtra("imgUri", imgUri)//is made to connect with db
and show the right image
```

```
                intent.putExtra("description", description)//string name and
keyidentifier
                intent.putExtra("date", date)
                intent.putExtra("location", location)
                cxt.startActivity(intent)
            }
        }
    }
    override fun getItemCount(): Int {
        return list.size
    }

    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

        val tvDescription: TextView = itemView.findViewById(R.id.img_desc)
        val tvDate: TextView = itemView.findViewById(R.id.img_date)
        val ivImg: ImageView = itemView.findViewById(R.id.sv_img)
        val but: View = itemView.findViewById(R.id.delet)
        val view = itemView
    }
 }



class EmotionsActivity : AppCompatActivity() {

    private lateinit var binding: ActivityEmotionsBinding
    private lateinit var adapter: ImageAdapter
    private lateinit var nameEmotions: String
    var count = 0
    var typeId = 0


    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityEmotionsBinding.inflate(layoutInflater)
        setContentView(binding.root)


        setupEmoji()
        setupSpinner()
        setupDesc()
        setListeners()

        val autoCompleteTextView =
findViewById<AutoCompleteTextView>(R.id.inputEmotions)


        val res = resources
        val resId = res.getIdentifier("emotions", "array", packageName)
        val emotions = res.getStringArray(resId)
        val adapter = ArrayAdapter(
            this,
            android.R.layout.simple_dropdown_item_1line,
            emotions
        )
        autoCompleteTextView.setAdapter(adapter)
```

```
    }

    private fun setupDesc() {
        val editText = findViewById<EditText>(R.id.inputDesc)
        editText.addTextChangedListener { text ->
            val textString = text.toString()
            // Save the text to a variable or a database
        }
    }

    private fun setupEmoji() {
        val gridLayout: GridLayout = findViewById(R.id.gridLayout1)

        val imageResourceIds = listOf(
            R.drawable.happy,
            R.drawable.good,
            R.drawable.okay,
            R.drawable.sad,
            R.drawable.verysad,
        )

        val taskDescArray = resources.getStringArray(R.array.emo_desc)

        val imageResources = imageResourceIds.mapIndexedNotNull { index,
resourceId ->
            val description = taskDescArray.getOrNull(index)
            if (description != null) {
                ImageData(resourceId, description)
            } else {
                null
            }
        }

        for (i in imageResources.indices) {
            val imageView = ImageView(this)
            val layoutParams = GridLayout.LayoutParams().apply {
                width = 130
                height = 130
                columnSpec =
                    GridLayout.spec(GridLayout.UNDEFINED, 1f) // Equal width
for each column
            }
            imageView.layoutParams = layoutParams
            val typeEmotions = imageResources[i].resourceId


            val imageResource = imageResources[i].resourceId
            val description = imageResources[i].description

            imageView.setImageResource(imageResource)
            imageView.contentDescription = description

            imageView.setOnClickListener {


                if (count > 0) {
                    println(count)
                    imageView.isClickable = false
                } else {
                    count = count + 1
                    if (typeEmotions == 2131230890)
                        typeId = 5
                    else if (typeEmotions == 2131230887)
```

```
                    typeId = 4
                else if (typeEmotions == 2131231008)
                    typeId = 3
                else if (typeEmotions == 2131231024)
                    typeId = 2
                else if (typeEmotions == 2131231039)
                    typeId = 1

                println(typeEmotions)
            }
            /** Додавання обраного до бази даних для графіку**/
        }


            gridLayout.addView(imageView)
        }
    }

    private fun setupSpinner() {
        ArrayAdapter.createFromResource(
            this,
            R.array.emotions,
            android.R.layout.simple_spinner_item
        ).also { adapter ->
            // Specify the layout to use when the list of choices appears

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_ite
m)
            // Apply the adapter to the spinner
            emSpinner.adapter = adapter
        }

        emSpinner.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener {
            override fun onItemSelected(parent: AdapterView<*>?, view: View?,
pos: Int, id: Long) {
                val selectedItem = parent!!.getItemAtPosition(pos).toString()
                if (inputEmotions.isEmpty())
                    inputEmotions.append(selectedItem)
                else inputEmotions.append(", $selectedItem")
                nameEmotions = selectedItem
            }

            override fun onNothingSelected(p0: AdapterView<*>?) {
                TODO("Not yet implemented")
            }
        }
    }

    @RequiresApi(Build.VERSION_CODES.O)
    private fun collectEmotions() {
        // val description = inputDesc.text.toString().trim()
        val emotions = inputEmotions.text.toString().trim()
        if (emotions.isEmpty()) {
            displayToast("Please try to distinguish your emotions")
        } else if (typeId < 1) {
            displayToast("Please try to distinguish your general mood")
        } else {
          //  val date: String =
           //     SimpleDateFormat("dd-MM-yyyy",
Locale.getDefault()).format(Date())
            val currentDate = LocalDate.now()

            // Get the day of the week as a string
```

```
            val dayOfWeek =
currentDate.dayOfWeek.getDisplayName(TextStyle.FULL, Locale.getDefault())

            // Print the day of the week
            println(dayOfWeek)
            helper.saveEmotions(
                typeId,
                emotions,
                dayOfWeek
            )
            Intent(applicationContext, MainActivity::class.java).also {
                startActivity(it)
            }
            showToast("Well done!")
        }
    }

    @RequiresApi(Build.VERSION_CODES.O)
    private fun setListeners() {
        binding.save.setOnClickListener {
            with(Intent(Intent.ACTION_SEND)) {
                type = "image/*"
                collectEmotions()
            }
        }
    }

    private fun showToast(message: String) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
    }

    private fun EditText.isEmpty(): Boolean {
        val etMessage = findViewById<EditText>(R.id.inputEmotions)
        val msg: String = etMessage.text.toString()
        return msg.trim().isEmpty()
    }

    private fun isUriEmpty(uri: Uri?): Boolean {
        return uri == null || uri == Uri.EMPTY
    }

}
```

Фрагменти коду UI

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".UserAuthentication">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout3"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.37" >

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/et_email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/enter_the_email" />

    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textInputLayout2"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textInputLayout3"
        app:passwordToggleEnabled="true">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/et_psw"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="@string/enter_paa" />

    </com.google.android.material.textfield.TextInputLayout>

    <Button
        android:id="@+id/btn_login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/login"
        app:layout_constraintEnd_toEndOf="@+id/textInputLayout2"
        app:layout_constraintStart_toStartOf="@+id/textInputLayout2"
        app:layout_constraintTop_toBottomOf="@+id/tv_statusLabel" />

    <TextView
        android:id="@+id/tv_statusLabel"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:gravity="end"
        android:text="@string/do_not_have_an_account"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@+id/tv_changeScreenStatus"
        app:layout_constraintEnd_toStartOf="@+id/tv_changeScreenStatus"
```

```
        app:layout_constraintTop_toTopOf="@+id/tv_changeScreenStatus" />

    <TextView
        android:id="@+id/tv_changeScreenStatus"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/click_here"
        android:textStyle="bold|italic"
        app:layout_constraintEnd_toEndOf="@+id/textInputLayout2"
        app:layout_constraintTop_toBottomOf="@+id/textInputLayout2" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Програмне забезпечення для моніторингу емоцій

Виконав: Дорошенко Софія Олександрівна

Керівник: асистент кафедри ПЗКС Северін Андрій Іванович

Київ – 2023

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити програмне забезпечення для моніторингу емоцій та почуттів при виконанні різних завдань.

Завдання:

1. Проаналізувати існуючі аналоги.
2. Вибір технологій розробки програмного забезпечення.
3. Розробити програмне забезпечення, що відповідає поставленим функціональним вимогам.
4. Протестувати розроблений продукт на предмет відсутності помилок.

АКТУАЛЬНІСТЬ

Зростання рівня стресу, перевантаження роботою та зменшення емоційного благополуччя впливають на наше здоров'я та якість життя. Усвідомлення емоцій та контроль над реакціями є важливими для досягнення позитивного емоційного стану.

Застосунки для розвитку емоційного інтелекту надають користувачам можливість покращити свою емоційну свідомість, зрозуміти фактори, що впливають на їхні емоції, та знайти шляхи для поліпшення свого емоційного стану. Їх використання може позитивно позначитися на самопочутті, ефективності та загальному благополуччі.

ІСНУЮЧІ АНАЛОГИ

VOS

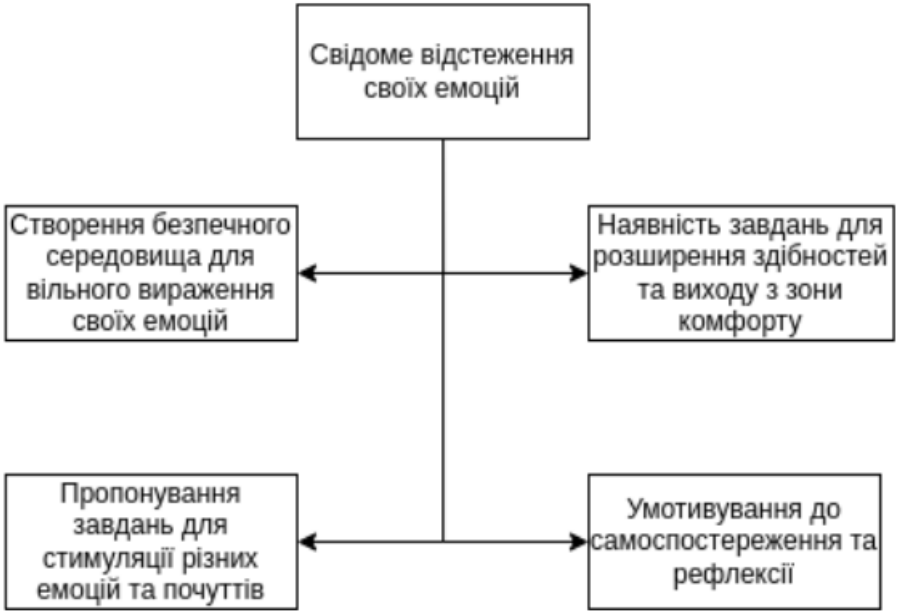
Thera

Daylio

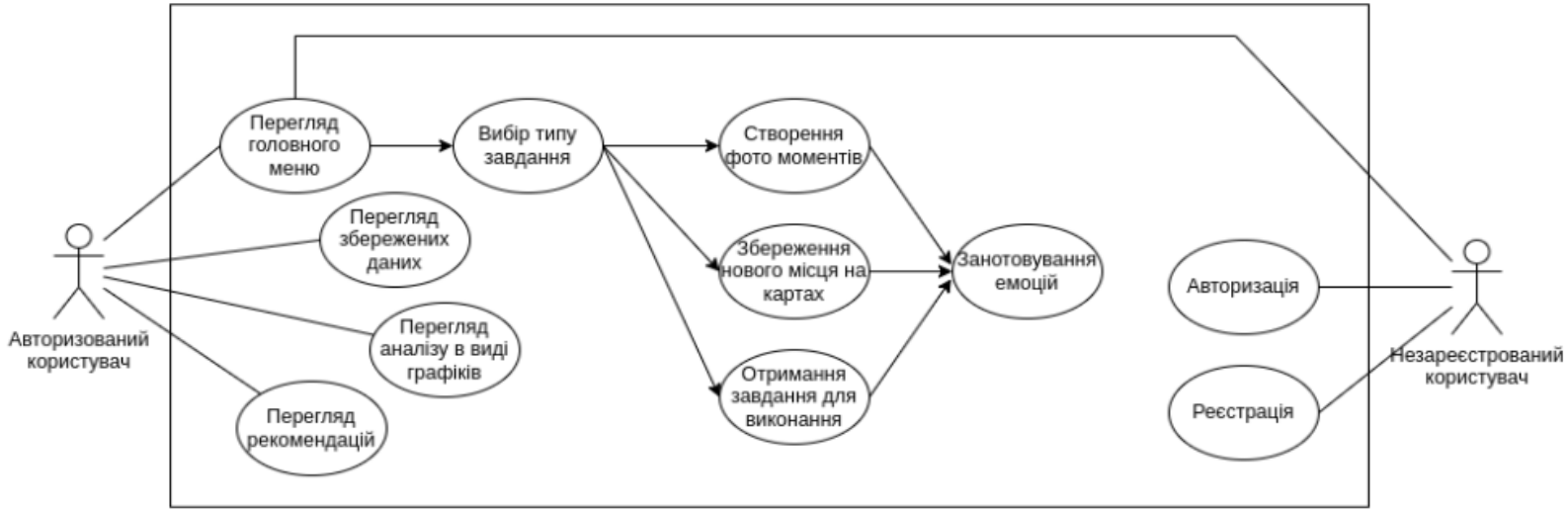
ЗАСОБИ РЕАЛІЗАЦІЇ

Тип ПЗ – мобільний застосунок.
Мова програмування – Kotlin.
Клієнтська частина – Jetpack Compose.
Серверна частина – Ktor.
СУБД – SQLite.

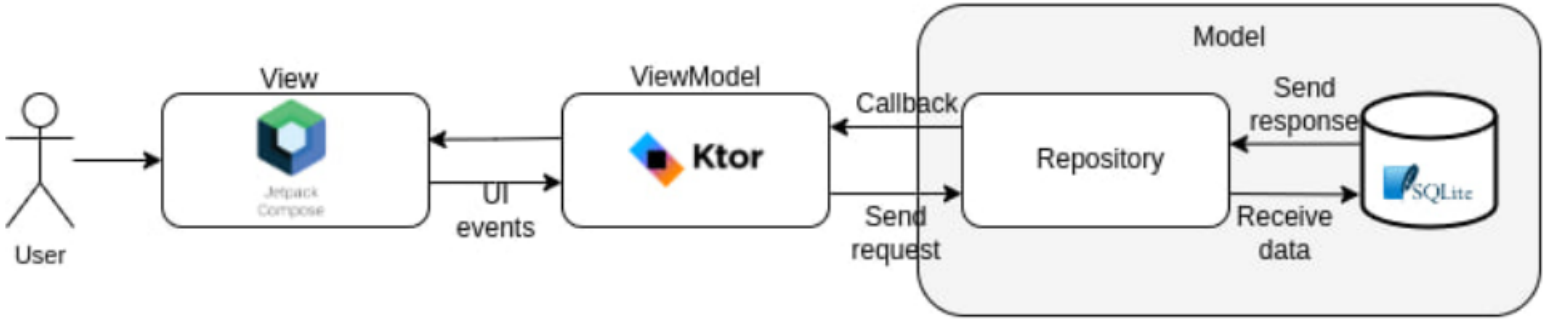
ДЕРЕВО ЦІЛЕЙ



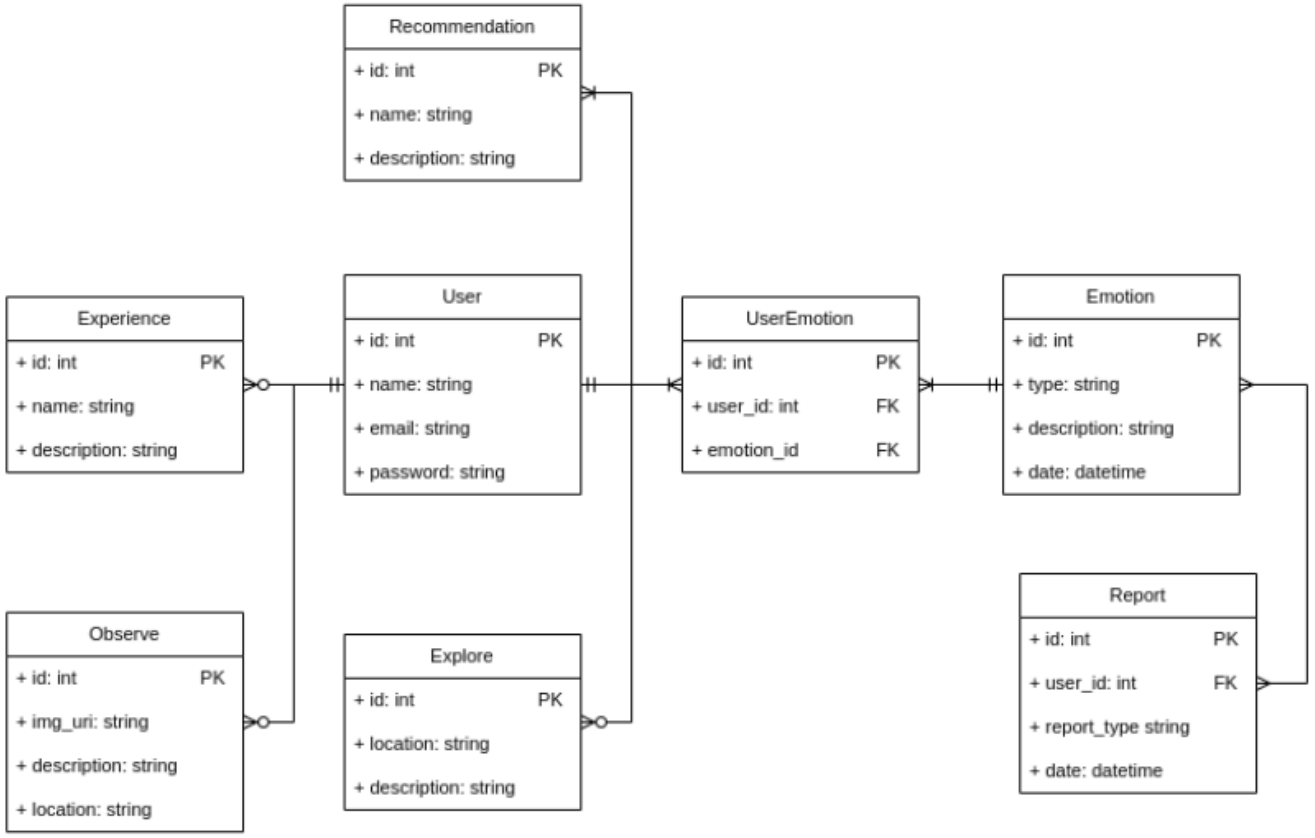
ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ЗАСТОСУНКУ



АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



СТРУКТУРА БАЗИ ДАНИХ



НАПРЯМКИ РОЗВИТКУ

Розширення функціональних можливостей:

- додавання сповіщень та нагадувань;
- можливість збереження і інших типів даних (напр. сон, фіз. активність);
- локалізація застосунку.

Оптимізація застосунку:

- покращення швидкості завантаження сторінок;
- оптимізація запитів до бази даних;
- кешування даних.

Покращення аналітики:

- надання можливості користувачам отримувати більше інсайтів і статистики щодо залежності між їхнім емоційним станом та виконанням завдань.

ПОРІВНЯННЯ З АНАЛОГАМИ

Критерій	VOS	Thera	Daylio	Розроблене ПЗ
Безкоштовна версія	Частково	Частково	Так	Так
Запис емоцій	Так	Так	Так	Так
Рекомендації	Так	Так	Ні	Так
Вибір емоцій	Так	Так	Так	Так
Аналіз даних	Так	Так	Так	Так
Пропонування завдань	Так	Так	Ні	Так
Допомога професіоналів	Так	Так	Ні	Ні
Курси та матеріали	Так	Так	Ні	Ні

ВИСНОВКИ

1. Проаналізовано існуючі програмні рішення.
2. Розглянуто актуальність проблеми та можливість рішення.
3. Сформульовано основні вимоги до системи.
4. Спроєктовано архітектуру системи та структуру бази даних.
5. Розроблено програмне забезпечення.
6. Протестовано за допомогою ручного тестування.
7. Представлено напрямки подальшого розвитку.

Розробка виконана у повному обсязі згідно з положенням Технічного завдання, тестування продукту проведено відповідно до затвердженої програми та методики тестування.

ПЕРЕВІРКА НА ПЛАГІАТ

UNICHECK

User name:
Северін Андрій Іванович

Check ID:
1015513158

Check date:
08.06.2023 18:04:05 EEST

Check type:
Doc vs Internet + Library

Report date:
08.06.2023 18:35:42 EEST

User ID:
100012354

File name: **ПЗ_Дорошенко_Софія**

Page count: **37** Word count: **7207** Character count: **62435** File size: **44.57 KB** File ID: **1015167236**

2.8% Matches

Highest match: **0.62%** with Internet source (https://ela.kpi.ua/bitstream/123456789/42545/1/Yereshko_bakalavr.pdf)

1.72% Internet sources	43	Page 39
2.4% Library sources	135	Page 39

0.1% Quotes

Quotes	1	Page 40

Exclusion of references is off

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЕМОЦІЙ

Програма та методика тестування

ДП.045490-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Андрій СЕВЕРІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Софія ДОРОШЕНКО

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для моніторингу емоцій являє собою систему, яка реалізована на мові програмування Kotlin. Програма створена за допомогою фреймворків Ktor та Jetpack Compose.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка наступних елементів:

1) функціональна працездатність елементів графічного інтерфейсу мобільного застосунку;
2) відповідність функціональних можливостей застосунку вимогам технічного завдання;
3) коректна обробка REST запитів;
4) зручність роботи з застосунком;
5) функціональні можливості реєстрації та автентифікації.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується за допомогою ручного тестування, автоматизованого тестування та проводиться тестування на реальних пристроях. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Використовуються наступні методи:

1) інтеграційне тестування;
2) функціональне тестування;
3) тестування продуктивності програмного забезпечення;
4) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність програмного забезпечення перевіряється шляхом:

1) динамічного ручного тестування на відповідність функціональним вимогам;
2) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
3) статичного тестування коду;
4) тестування мобільного застосунку на різних пристроях;
5) тестування стабільності роботи при різних умовах;
6) тестування зручності використання;
7) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЕМОЦІЙ

Керівництво користувача

ДП.045490-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Андрій СЕВЕРІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Софія ДОРОШЕНКО

2023

ЗМІСТ

1. Опис структури мобільного застосунку

Програмне забезпечення для моніторингу емоцій складається зі статичних сторінок. Додаток виконаний англійською мовою.

Сторінки застосунку:

- сторінка привітання;
- сторінка авторизації/реєстрації нового користувача;
- сторінка головного меню;
- сторінка з типом завдання “Experience”;
- сторінка з типом завдання “Observe”;
- сторінка з типом завдання “Explore”;
- сторінка з аналізом даних в виді графіка.

Для повноцінного користування застосунком потрібно зайти в систему або зареєструватися, якщо користувач вперше використовує застосунок. Пройшовши аутентифікацію, стають доступні різні завдання для виконання.

2. Опис початку роботи з застосунком

При запуску програмного забезпеченння користувач потрапляє на сторінку з назвою та коротким описом суті застосунку (рис. 1.1). Графічний інтерфейс сторінок формується відповідно до ролей користувача, а саме авторизованого та неавторизованого користувача системи. Детальніше розглянемо переліки доступних сторінок для кожної з зазначених ролей.

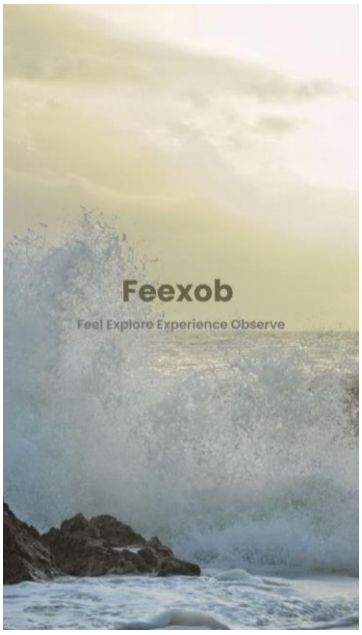


Рис. 1.1. Сторінка входу

Неавторизований користувач потрапляє на сторінку авторизації та реєстрації. В залежності від потреби, користувач вибирає переходити на сторінку входу, або на сторінку реєстрації. Навігація між наведеними сторінками виконується за рахунок взаємодії користувача з відповідними графічними елементами керування, які виділені шрифтом (рис. 1.2).

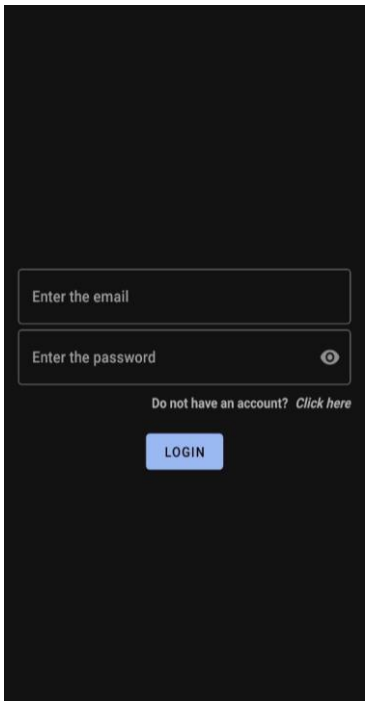


Рис. 1.2. Сторінка неавторизованого користувача

3. Процедура обрання типу завдання

На головній сторінці зображеній на рис. 2.1 користувач має можливість обрати тип завдання для виконання, переглянути аналіз даних, або ж вийти з акаунту.

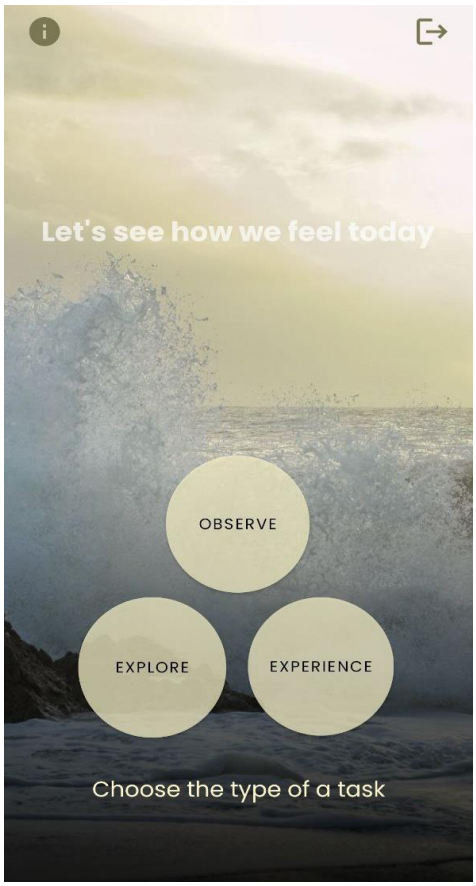


Рис. 3.1. Головне меню

При виборі першого завдання “Observe” користувач отримує повідомлення з завданням зробити або завантажити фото та занотувати, які виникли емоцій в той момент. Таким чином, є можливість скористатися камерою або обрати фото з галереї та опинитись на новій сторінці з обраним фото для збереження моменту (рис. 2.2).

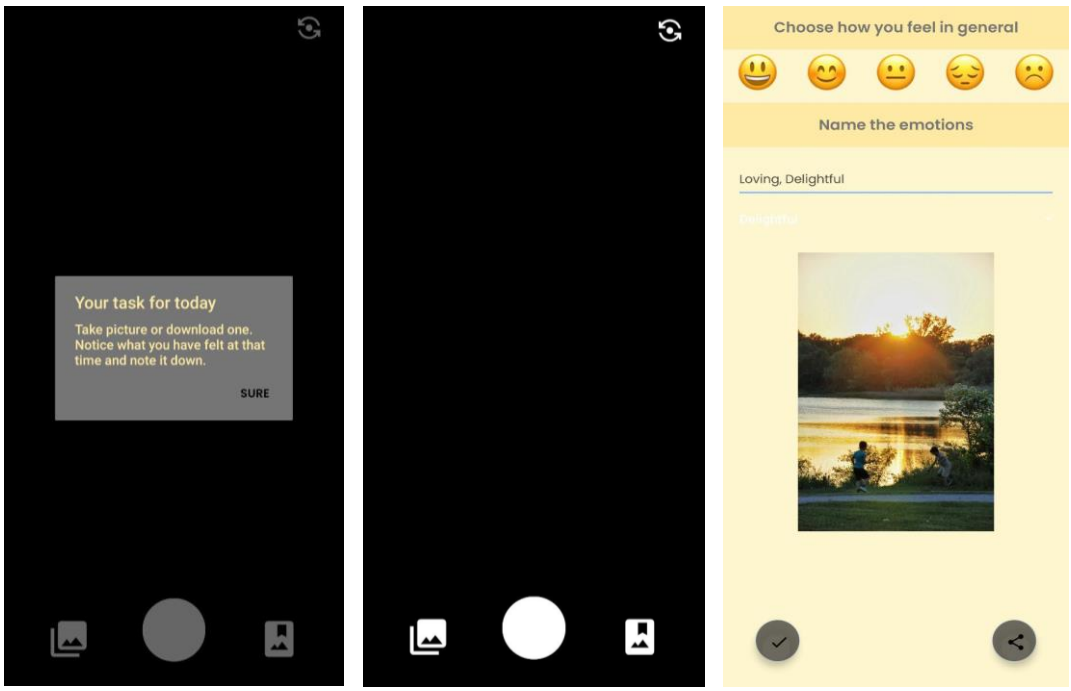


Рис. 3.2. Сторінки типу завдання “Observe”

Фото з відповідними даними зберігаються в БД застосунку. Самі фото, їх назву, локацію та дату створення можна переглянути на наступній сторінці (рис. 2.3). Також їх можна видаляти по бажанню.

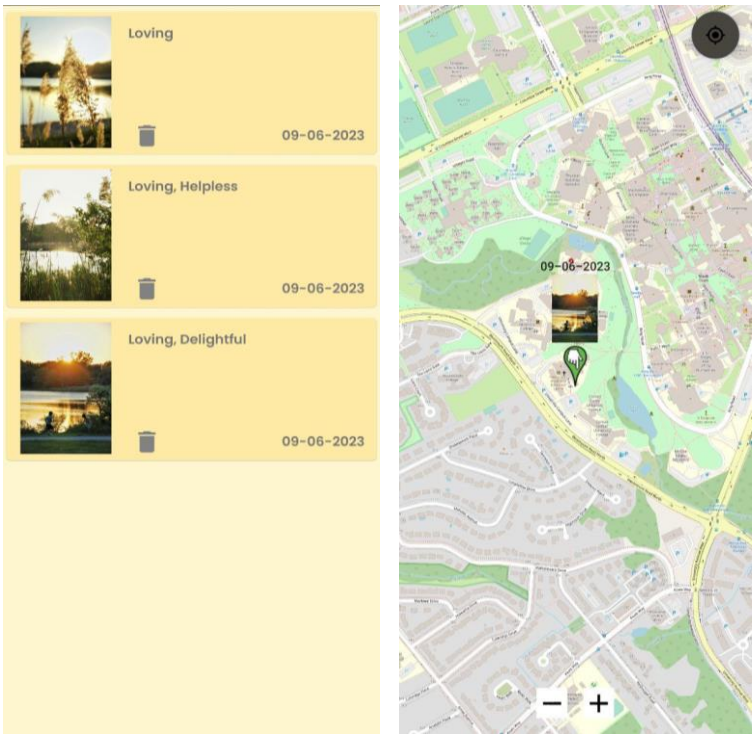


Рис. 3.3. Збереження даних

При виборі опції “Explore” користувачу пропонується відвідати нове місце цього дня (це може бути кімната в університеті, магазин, вулиця), де він ніколи не був раніше. Нове місце можна обрати на карті та записати емоції, які виникли при відвідуванні цього місця, у новому вікні (рис. 3.4).

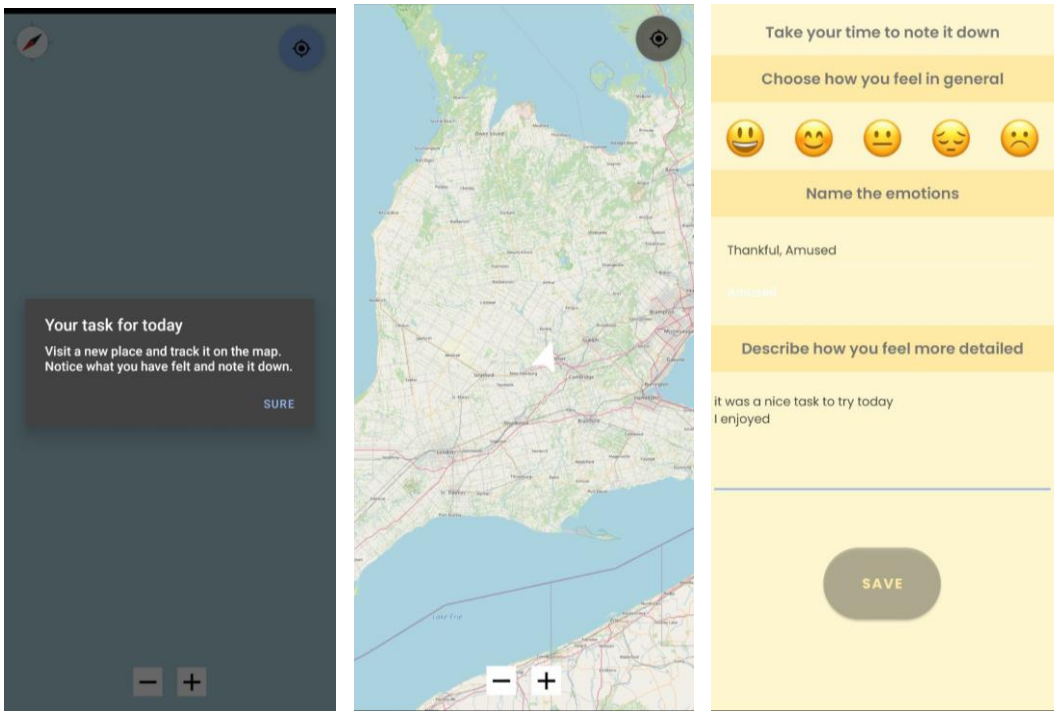


Рис. 3.4. Сторінки типу завдання “Explore”

При виборі третього типу завдання “Experience”, користувачу буде надано випадкове завдання для виконання протягом дня (рис. 3.5). Наприклад: потанцювати під улюблену пісню, намалювати картину пензлем, приготувати щось солодке, помедитувати, прослухати певний трек, тощо. І записати емоції та почуття, що виникли при виконання цих завдань.

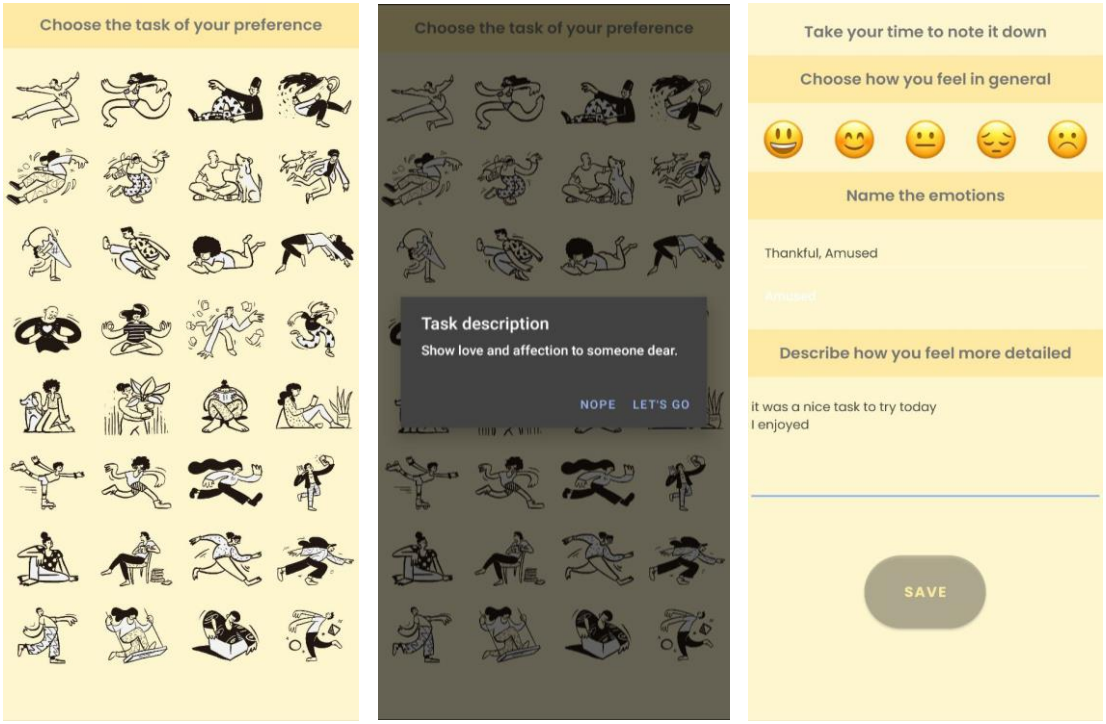


Рис. 3.5. Сторінки типу завдання “Experience”

4. Аналіз даних користувача

Записані емоції користувача будуть збережені та відображені у виді графіка, який користувач має можливість відкрити з головного меню (рис. 4.1). На графіку зображено середнє статистичне від настрою користувача кожного дня тижня. Це може бути цінним інструментом для самоаналізу та рефлексії.

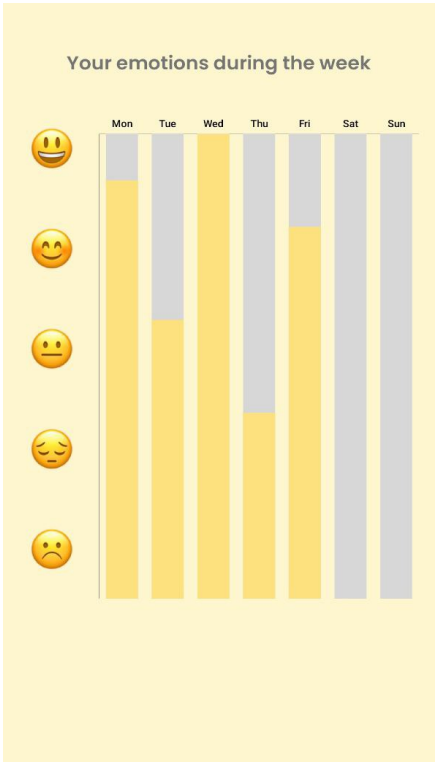


Рис. 4.1. Графік з аналізом емоцій користувача за тиждень