Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії»

Виконав: студент 4 курсу
групи 5ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Пасіхов О.М.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. Дудник О. В.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«03» 06 2025 р.

ВНТУ – 2025

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

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

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

Пасіхову Олександру Михайловичу

1. Тема роботи – «Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії»

Керівник роботи: Чехместрук Роман Юрійович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова програмування TypeScript.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку систем психологічної підтримки; розробка моделі та алгоритмів програмного застосунку; варіантний аналіз та вибір мови програмування розробки; розробка програмної системи для психологічної підтримки; тестування програмного забезпечення; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Чехместрук Р.Ю., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку систем психологічної підтримки	25.03.25 – 31.03.25	Вик
2	Розробка моделі та алгоритмів програмного застосунку	01.04.25 – 07.04.25	Вик
3	Варіантний аналіз та вибір мови програмування розробки	08.04.25 – 10.04.25	Вик
4	Розробка програмної системи для психологічної підтримки	11.04.25 – 24.04.25	Вик
5	Тестування програмного забезпечення	25.04.25 – 05.05.25	Вик
6	Оформлення матеріалів до захисту БКР	06.05.25 – 06.06.25	Вик

Студент _____ Пасіхов О.М.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Чехместрук Р.Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004:005.9

Пасіхов О.М. Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця : ВНТУ, 2025. 84 с.

На укр. мові. Бібліогр. : 27 назв.; рис.: 33; табл.: 13.

У бакалаврській кваліфікаційній роботі було проведено аналіз стану питання розробки програмної системи діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії, проведено аналіз методів розв’язання задачі з розробки цієї програмної системи. Проведено порівняльний аналіз аналогів, продемонстровано переваги власної розробки. Проведено постановку задач дослідження.

У ході роботи було розроблено інформаційне забезпечення програмної системи діалогової взаємодії для надання психологічної підтримки, розроблено модель системи, яка наводить технології та модулі, які розроблені для роботи системи, взаємозв’язки між ними. Розроблено алгоритми роботи системи, розроблено метод розширеного журналу думок із автоматичним аналізом та метод динамічного добору вправ за показниками користування.

Програмне забезпечення розроблено із використанням середовища розробки Visual Studio Code та мови програмування TypeScript. Під час тестування було продемонстровано її функціонал та доведено її працездатність.

У результаті виконання роботи розроблено веб-застосунок, що дозволяє отримати психологічну підтримку на основі методів когнітивно-поведінкової терапії.

Ключові слова: веб-застосунок, TypeScript, когнітивно-поведінкова терапія.

ABSTRACT

UDC 004:005.9

Pasikhov O.M. Intelligent software system of dialogic interaction for providing psychological support based on cognitive-behavioral therapy methods: bachelor's qualification work in the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2025. 84 p.

In Ukrainian. Bibliography: 27 titles; fig.: 33; tab.: 13.

In the bachelor's qualification work, an analysis of the state of the issue of developing a software system of dialogic interaction for providing psychological support based on cognitive-behavioral therapy methods was conducted, methods for solving the problem of developing this software system were analyzed. A comparative analysis of analogues was conducted, the advantages of our own development were demonstrated. Research tasks were formulated.

In the course of the work, the information support of the software system of dialogic interaction for providing psychological support was developed, a system model was developed that lists the technologies and modules that are designed for the system's operation, the relationships between them. The algorithms of the system's operation were developed, a method of an extended thought journal with automatic analysis and a method of dynamic selection of exercises based on usage indicators were developed.

The software was developed using the Visual Studio Code development environment and the TypeScript programming language. During testing, its functionality was demonstrated and its operability was proven.

As a result of the work, a web application was developed that allows you to receive psychological support based on cognitive behavioral therapy methods.

Keywords: web application, TypeScript, cognitive behavioral therapy.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Поширення стресу, тривоги та депресивних станів у сучасному суспільстві вимагає нових підходів до надання психологічної підтримки, доступних у будь-який час та в будь-якому місці [1]. Програмні системи діалогової взаємодії на основі методів когнітивно-поведінкової терапії стають ефективним інструментом, який здатен охопити широку аудиторію користувачів, не замінюючи традиційну психотерапію, але доповнюючи її інтерактивними консультаціями та вправами. Завдяки сучасним досягненням у галузі обробки природної мови та штучного інтелекту, такі системи можуть аналізувати емоційний стан користувача, пропонувати релевантні техніки самоконтролю й допомагати впроваджувати нові адаптивні стратегії поведінки [2].

Чат-боти реалізують серію структурованих алгоритмів, що базуються на виявленні автоматичних негативних думок, оцінці доказів «за» та «проти», а також побудові альтернативних раціональних переконань [3]. Система зазвичай складається з модулів інтерпретації вводу користувача, контекстного аналізу настрою, набору вправ із психоосвіти та практичних завдань, а також механізмів зворотного зв’язку й моніторингу прогресу. Такий підхід дозволяє користувачеві поступово засвоїти прийоми самодопомоги й застосовувати їх у повсякденному житті, водночас отримуючи як миттєві, так і довгострокові рекомендації [4].

Упровадження диференційованих рівнів персоналізації й адаптивності, зокрема адаптація діалогових сценаріїв під індивідуальні потреби та особливості користувача, сприяє підвищенню ефективності психотерапевтичного процесу [5]. Крім того, інтеграція мультимодальних інтерфейсів — тексти, аудіо- й відеоматеріали — робить взаємодію більш залученою та емоційно насиченою [6]. У результаті програмна система стає доступним, недорогим і масштабованим рішенням для первинної психологічної підтримки, стимулюючи користувачів звертатися за додатковою професійною допомогою при виявленні складніших або хронічних проблем із психічним здоров’ям [7].

Зв'язок роботи з науковими програмами, планами, темами. Робота

виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є покращення процесу надання психологічної підтримки користувачам шляхом розробки системи діалогової взаємодії для надання психологічної підтримки.

Основними задачами дослідження є:

- розробити функціонал програмної системи діалогової взаємодії для надання психологічної підтримки;

- розробити алгоритми роботи програмної системи діалогової взаємодії для надання психологічної підтримки;

- розробити архітектуру програмної системи діалогової взаємодії для надання психологічної підтримки ;

- розробити інтерфейс користувача застосунку;

- провести тестування розробленої системи.

Об'єкт дослідження – процес розробки програмних модулів системи діалогової взаємодії для надання психологічної підтримки.

Предмет дослідження – методи та засоби розробки програмних модулів системи діалогової взаємодії для надання психологічної підтримки.

Методи дослідження. У процесі досліджень використовувались:

- методи реалізації систем для надання психологічної підтримки;

- методи журналу думок з автоматичним аналізом;

- методи динамічного добору вправ;

- методи реалізації інтерфейсу;

- методи тестування.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод розширеного журналу думок із автоматичним аналізом, який, на відміну від відомих, поєднує класичні етапи когнітивно-поведінкової терапії з глибинним аналізом у режимі реального часу, що дозволяє провести детальний аналіз думок.

2. Подальшого розвитку отримав метод динамічного добору вправ за

показниками користування, який, на відміну від відомих, адаптує терапевтичний план на основі реальних даних про ефективність і залученість користувача.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає у можливості використовувати розроблений застосунок для самоаналізу та емоційної регуляції.

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

Апробація результатів роботи. Описані у курсовому проєкті положення доповідались на конференції «Global Trends in the Development of Information Technology and Science 2025».

Публікації. Результати роботи були опубліковані в матеріалах конференції Collection of Scientific Papers "International Scientific Unity" with Proceedings of the 3rd International Scientific and Practical Conference [3].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто 4 розділи та було використано 23 літературні джерела.

1 АНАЛІЗ СТАНУ РОЗВИТКУ СИСТЕМ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

1.1 Аналіз стану питання розробки

Розвиток інтелектуальних програмних систем діалогової взаємодії для надання психологічної підтримки із застосуванням методів когнітивно-поведінкової терапії набуває все більшої популярності [8]. Сучасні технології дозволяють створювати програми, що здатні взаємодіяти з користувачем у режимі реального часу, надаючи персоналізовану допомогу при вирішенні стресових ситуацій та покращенні психоемоційного стану.

Важливим аспектом є інтеграція методів когнітивно-поведінкової терапії, яка передбачає використання алгоритмів для виявлення деструктивних когнітивних схем та подальшого їх коригування. Модель когнітивно-поведінкової терапії наведена на рисунку 1.1.

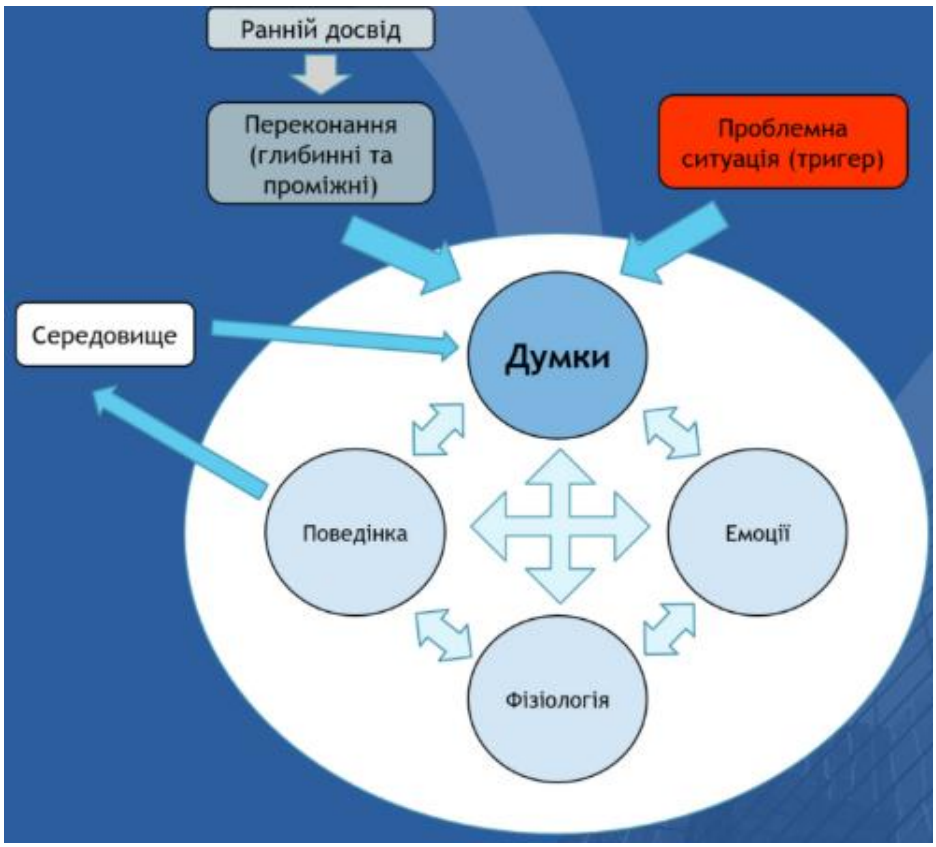

Рисунок 1.1 — Модель когнітивно-поведінкової терапії

Система може здійснювати когнітивне реструктурування, пропонувати

техніки релаксації, допомагати у веденні журналу думок і моніторингу емоцій. Такий підхід дозволяє не лише ідентифікувати негативні патерни мислення, але й надавати користувачу рекомендації, типові для когнітивно-поведінкової терапії, що адаптуються до його попереднього досвіду взаємодії.

За останніми досягненнями у сфері обробки природної мови, зокрема завдяки розвитку трансформерних моделей і глибинного навчання, сучасні системи можуть більш точно інтерпретувати емоційний стан користувача [9]. Ці технології дозволяють створити діалогові системи, здатні враховувати контекст бесіди та індивідуальні особливості. Крім того, інтеграція голосових технологій, аналіз інтонації та навіть відеоаналіз допомагають зробити взаємодію більш природною та наближену до реального людського спілкування.

Серед численних переваг подібних систем слід зазначити їхню доступність, анонімність та економічність. Доступність забезпечується можливістю отримати підтримку цілодобово, а анонімність дозволяє користувачеві відчувати себе більш розслаблено. Проте існують і певні обмеження: сучасні алгоритми досі не здатні повністю відтворити тонкі нюанси людських емоцій та невербальних сигналів, що може призвести до неправильного тлумачення стану користувача. Крім того, навіть найбільш розвинені системи не можуть замінити кваліфіковану психологічну допомогу у випадках серйозних психічних розладів.

Також варто звернути увагу на етичні та правові аспекти використання таких систем. Забезпечення конфіденційності даних користувачів, впровадження сучасних протоколів безпеки та відповідність вимогам, є обов’язковими умовами для їх використання [10]. Водночас важливо визначити межі застосування систем, особливо в ситуаціях, коли користувач може потребувати негайної допомоги від фахівця, та розробити механізми перенаправлення до кваліфікованих психологів або психотерапевтів.

Таким чином, інтелектуальні програмні системи діалогової взаємодії на базі когнітивно-поведінкової терапії мають значний потенціал для розширення доступу до психологічної підтримки та покращення якості життя користувачів. Поєднання передових технологій штучного інтелекту з науково обґрунтованими

терапевтичними методами дозволяє створити інноваційні рішення, проте для їх успішної імплементації важливо враховувати як технічні, так і етичні виклики.

1.2 Порівняльний аналіз аналогів

Woebot [11] — це чат-бот, розроблений для надання психологічної підтримки з використанням методів когнітивно-поведінкової терапії.

Інтерфейс Woebot наведено на рисунку 1.2.

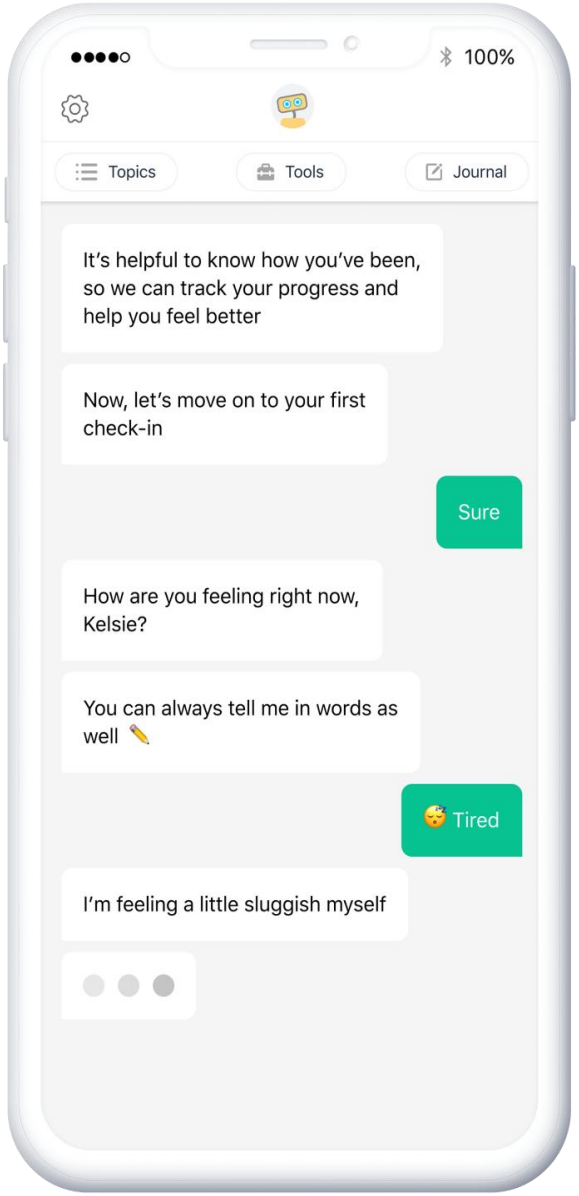

Рисунок 1.2 — Інтерфейс Woebot

Він був створений з метою допомогти користувачам у щоденних стресових

ситуаціях, пропонуючи інтуїтивний та доступний інтерфейс для спілкування. Woebot використовує сучасні алгоритми обробки природної мови, що дозволяє вести живий, персоналізований діалог і адаптуватися до емоційного стану користувача.

Основою роботи Woebot є застосування принципів когнітивно-поведінкової терапії, завдяки яким бот допомагає користувачу розпізнавати і змінювати негативні патерни мислення. Під час взаємодії користувач може отримувати поради, техніки релаксації та завдання, які сприяють самостійній роботі над емоційним станом. Алгоритми, що стоять за Woebot, дозволяють проводити базовий аналіз текстових повідомлень та адаптувати відповіді до індивідуальних потреб кожного користувача.

Woebot відзначається високою доступністю, оскільки надає підтримку 24/7, що особливо важливо для тих, хто шукає негайну допомогу. Завдяки анонімності спілкування, користувачі можуть відкрито обговорювати свої емоції, не турбуючись про конфіденційність. Попри всі переваги, варто пам’ятати, що Woebot не замінює професійну терапію, а служить допоміжним інструментом для самодопомоги.

Wysa [12] — це інтерактивний застосунок для емоційної підтримки, який поєднує методи когнітивно-поведінкової терапії з іншими підходами психічного здоров’я. Програма була розроблена з урахуванням потреб користувачів, які шукають ефективні інструменти для подолання стресу, тривоги та інших негативних емоційних станів. Інтерфейс Wysa робить акцент на простоті та інтуїтивності, що сприяє легкому спілкуванню.

Основною особливістю Wysa є використання діалогових методів для надання підтримки. Застосунок пропонує користувачам набір інтерактивних вправ, медитаційних технік та завдань, що допомагають контролювати емоційні реакції. Завдяки алгоритмам штучного інтелекту, Wysa здатна адаптувати свої відповіді до конкретних потреб і емоційного стану користувача, забезпечуючи більш персоналізований підхід.

Інтерфейс Wysa наведено на рисунку 1.3.

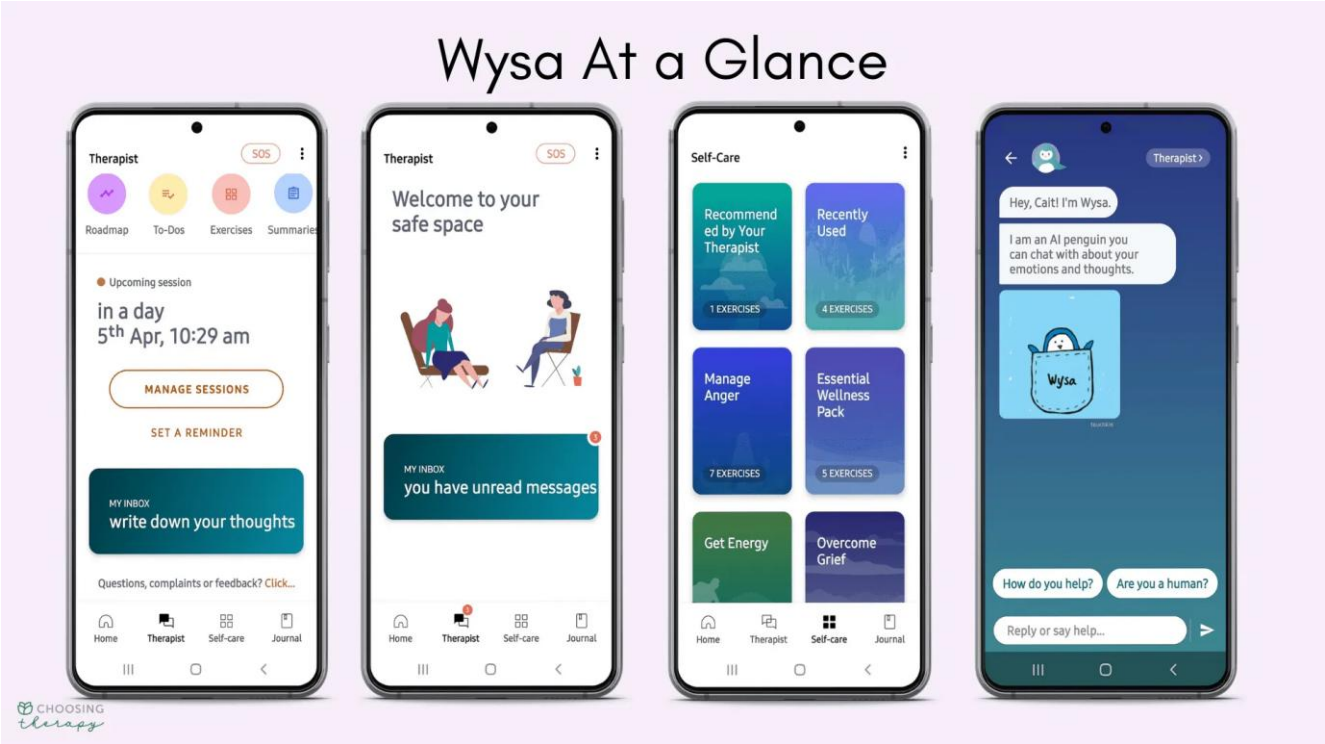

Рисунок 1.3 — Інтерфейс Wysa

Wysa має низку переваг, серед яких особлива увага до конфіденційності та безпеки даних користувача. Програма дає змогу вести анонімний діалог, що важливо для тих, хто відчуває незручність при спілкуванні з іншими. Крім того, Wysa постійно оновлюється і доповнюється новими методами, що дозволяє забезпечувати користувачів сучасними засобами психологічної підтримки, проте вона також не може повністю замінити професійну терапевтичну допомогу.

Youper [13] — це застосунок для моніторингу емоційного стану, який активно використовує методи когнітивно-поведінкової терапії для надання персоналізованої підтримки. Він орієнтований на користувачів, які бажають отримати швидку і доступну допомогу в управлінні своїми емоціями та стресом. Youper поєднує в собі елементи штучного інтелекту та психотерапевтичних технік, створюючи інтерактивне середовище для самопізнання.

Інтерфейс Youper наведено на рисунку 1.4.

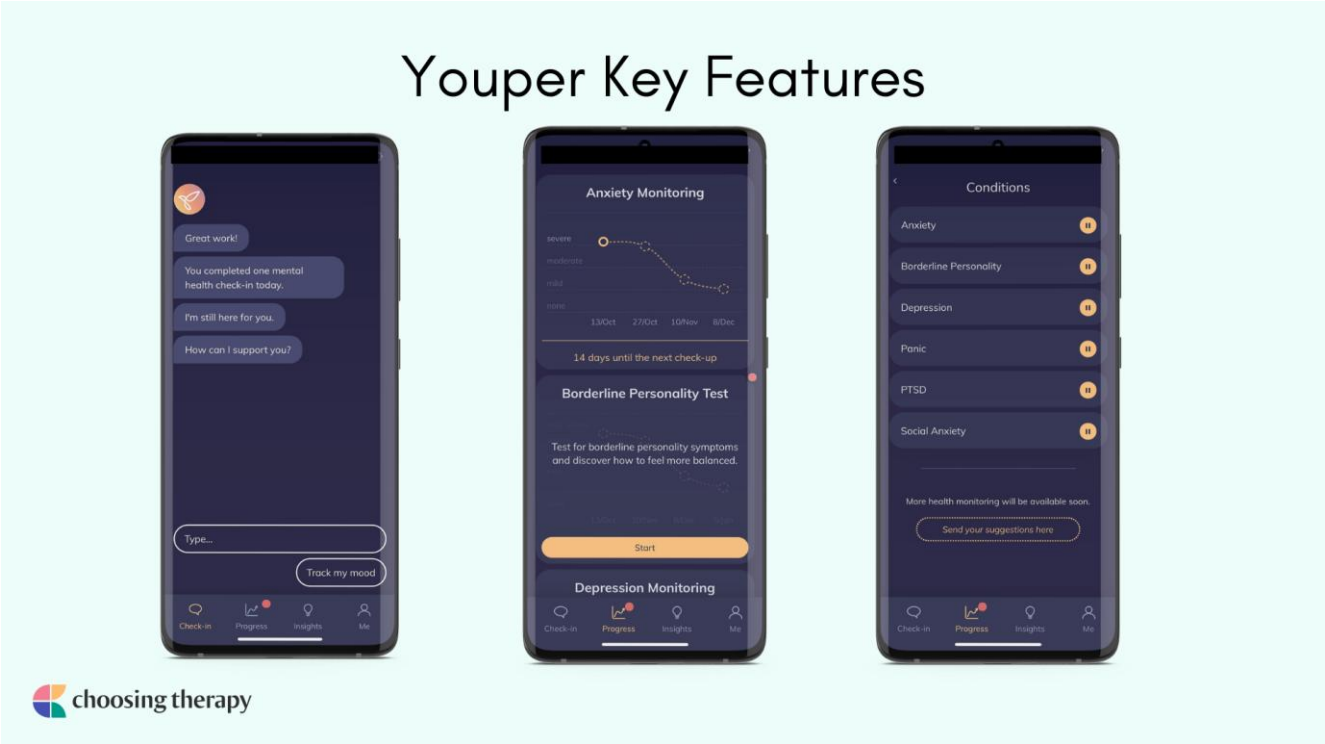

Рисунок 1.4 — Інтерфейс Youper

Youper використовує алгоритми аналізу настрою, що дозволяють виявляти тонкі зміни в емоційному стані користувача на основі його текстових повідомлень. Під час взаємодії застосунок пропонує конкретні вправи та рекомендації, спрямовані на корекцію негативних патернів мислення. Використання методів КПТ у Youper сприяє тому, щоб користувач міг краще розуміти свої емоції і вчасно знаходити способи їх регулювання.

Однією з ключових переваг Youper є інтуїтивний дизайн, який дозволяє користувачам легко орієнтуватися в застосунку та отримувати підтримку у режимі реального часу. Завдяки використанню штучного інтелекту, система може постійно вдосконалювати свої рекомендації, адаптуючись до індивідуальних особливостей кожного користувача. Попри це, як і інші подібні інструменти, Youper є допоміжним засобом і не може замінити професійного психотерапевта при глибоких психологічних проблемах.

Для порівняння можливостей цих застосунків між собою та з власною розробкою, було сформовано таблицю 1.1.

Таблиця 1.1 — Характеристики аналогів

Характеристика	Woebot	Wysa	Youper	Власна розробка
Використання когнітивно-поведінкової терапії	+	+	+	+
Штучний інтелект	-	+	+	+
Діалогова взаємодія	+	+	+	+
Анонімність	+	-	+	+
Персоналізація	+	+	-	+
Моніторинг настрою	+	+	+	+
Сумарний бал	5	5	5	6

Таким чином, власна розробка є актуальною, оскільки власна розробка має вищий сумарний бал за усі аналоги на 16,7% (100% - 5/6*100% = 16,7%).

1.3 Аналіз методів розв’язання задачі

Методи розв'язання задачі розробки інтелектуальної програмної системи діалогової взаємодії для надання психологічної підтримки на основі когнітивно-поведінкової терапії включають комплекс підходів, які можна умовно розділити на алгоритмічні, модельні та гібридні рішення.

Перш за все, одним із базових підходів є використання правил та сценаріїв, які базуються на традиційних алгоритмах обробки природної мови. У таких системах розробники заздалегідь задають шаблони відповідей та алгоритми виявлення ключових елементів мови, що свідчать про емоційний стан користувача. Цей метод дозволяє забезпечити контрольовану та передбачувану взаємодію, адже відповіді формуються за чітко визначеними сценаріями, які включають типові техніки когнітивно-поведінкової терапії, такі як когнітивне реструктурування чи техніки релаксації. Однак такі системи часто обмежені у здатності адаптуватися до нестандартних або незвичних запитів користувачів.

Альтернативою є використання методів машинного навчання та сучасних моделей глибинного навчання, зокрема трансформерних архітектур (наприклад, GPT, BERT) (рисунок 1.5).

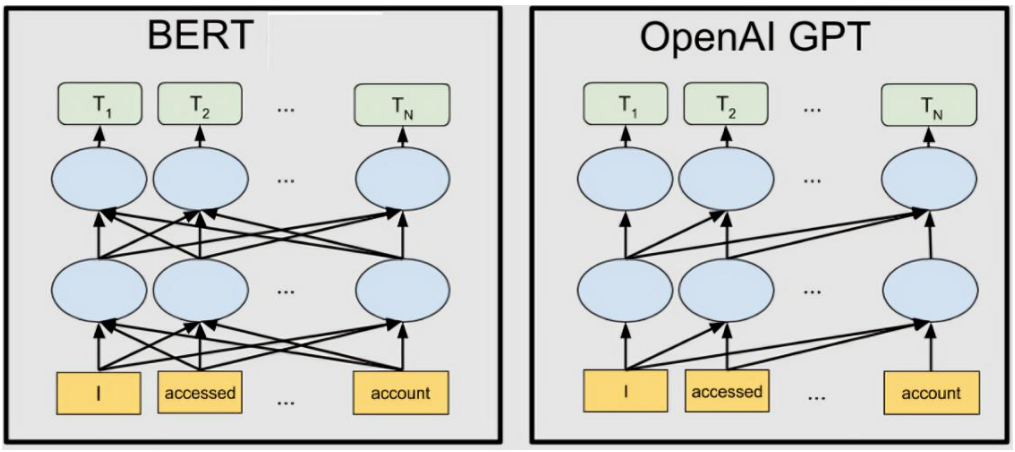

Рисунок 1.5 — Архітектура BERT та GPT

Ці підходи дозволяють системі аналізувати контекст бесіди, виявляти емоційні нюанси та генерувати персоналізовані відповіді. Застосування таких моделей у комбінації з додатковою спеціалізованою підготовкою на даних, пов’язаних із когнітивно-поведінковою терапією, сприяє більш точній інтерпретації психоемоційного стану користувача та адаптації терапевтичних рекомендацій. Проте для цього потрібні великі обсяги даних та ретельне тестування, щоб уникнути потенційно небажаних або неточних відповідей.

У таблиці 1.2 наведено порівняння підходу на основі сценаріїв та машинного навчання.

Таблиця 1.2 — Порівняння підходів до розробки системи діалогової взаємодії для психологічної підтримки

Критерій	Підхід на основі сценаріїв	Машинне навчання
Основний принцип	Заздалегідь визначені шаблони відповідей та алгоритми виявлення ключових елементів	Аналіз контексту та генерація персоналізованих відповідей на основі даних

Продовження таблиці 1.2

Критерій	Підхід на основі сценаріїв	Машинне навчання
Переваги	• Контрольована та передбачувана взаємодія • Надійність у типових ситуаціях • Можливість точного дотримання терапевтичних протоколів • Простота налагодження та корекції	• Гнучкість та адаптивність • Персоналізація відповідей• Виявлення емоційних нюансів • Обробка нестандартних запитів
Недоліки	• Обмежена здатність до адаптації • Складність обробки нестандартних ситуацій • Штучність деяких відповідей • Необхідність ручного програмування всіх сценаріїв	• Потреба у великих обсягах даних • Непередбачуваність деяких відповідей • Складність контролю якості • Високі обчислювальні ресурси
Область застосування	• Базові терапевтичні протоколи • Стандартизовані запити • Технічна підтримка по когнітивно-поведінковій терапії • Навчальні модулі	• Емоційний аналіз • Індивідуалізовані рекомендації • Складні діалоги
Складність розробки	Середня	Висока
Вимоги до ресурсів	Низькі	Високі
Інтерпретованість	Висока	Низька
Масштабованість	Низька	Висока
Швидкість відповіді	Висока	Середня
Якість персоналізації	Низька	Висока

Важливим аспектом є розробка стратегії управління діалогом. Системи можуть використовувати як класичні алгоритми діалогового менеджменту, так і

методи навчання з підкріпленням, що дозволяє системі в режимі реального часу коригувати свої відповіді залежно від реакції користувача. Такий підхід сприяє більш динамічній та природній взаємодії, проте водночас вимагає значних ресурсів для навчання та оптимізації.

Інтеграція методів когнітивно-поведінкової терапії вимагає побудови окремих модулів для аналізу когнітивних спотворень, моніторингу емоційного стану та надання відповідних рекомендацій. У цьому випадку ефективним може бути комбінований підхід: використання правил для визначення базових терапевтичних протоколів і машинного навчання для адаптації до індивідуальних особливостей користувача. Такий гібридний підхід дозволяє зберегти контроль над критичною частиною діалогу, водночас забезпечуючи достатню гнучкість і персоналізацію відповідей.

Крім того, варто зазначити, що оцінка ефективності застосованих методів є невід'ємною складовою процесу розробки. Систематичне тестування через симуляції діалогів, експертну оцінку терапевтичної цінності відповідей та проведення користувацьких досліджень допомагає виявити недоліки та вчасно їх скоригувати. Особлива увага приділяється забезпеченню етичних стандартів: конфіденційності даних, відповідності нормативним вимогам та безпеці користувача, що є критичними в контексті психологічної підтримки.

Таким чином, айбільш ефективним є гібридний підхід, який поєднує стабільність правил з адаптивністю даних, дозволяючи створити систему, здатну не лише точно реагувати на запити користувача, а й надавати кваліфіковану терапевтичну підтримку в режимі реального часу.

1.4 Постановка задач дослідження

Провівши аналіз методів розв’язання поставленої задачі, аналіз стану програмних систем діалогової взаємодії для надання психологічної підтримки, порівняння аналогів було поставлено такі задачі дослідження:

- розробити функціонал програмної системи діалогової взаємодії для надання психологічної підтримки;

- розробити алгоритми роботи програмної системи діалогової взаємодії для надання психологічної підтримки;

- розробити архітектуру програмної системи діалогової взаємодії для надання психологічної підтримки ;

- розробити інтерфейс користувача застосунку;

- провести тестування розробленої системи.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було проведено аналіз стану питання розробки програмної системи діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії, проведено аналіз методів розв’язання задачі з розробки цієї програмної системи. Проведено порівняльний аналіз аналогів, продемонстровано переваги власної розробки. Проведено постановку задач дослідження.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Розробка інформаційного забезпечення

Застосунок забезпечує регулярний і точний моніторинг емоційного стану користувача за допомогою простих, але водночас інформативних шкал. Кожна оцінка записується й автоматично агрегується у відсотковий розподіл настроїв – від «Відмінно» до «Погано». Завдяки такому підходу користувач може легко помітити навіть незначні зміни у власному емоційному фоні, виявити повторювані патерни та своєчасно реагувати на зародження тривожних чи депресивних тенденцій.

Рефлексивний щоденник думок побудований за чіткою послідовністю: спочатку необхідно описати подію чи ситуацію, що спричинила емоційну реакцію, потім зафіксувати автоматичні негативні думки та оцінити їх інтенсивність. Далі користувач шукає факти, які спростовують ці думки, і формулює більш збалансоване альтернативне твердження. Завершальний етап — повторна оцінка емоційного стану — дозволяє порівняти «до» і «після» та побачити ефект від проговорювання і переосмислення своїх переконань.

Бібліотека вправ охоплює різнорівневі методики когнітивно-поведінкової терапії й включає як щоденникові техніки (Thought Record, Gratitude Journal), так і практики релаксації (Deep Breathing, Progressive Muscle Relaxation) чи планування позитивних активностей (Activity Planning). Кожна вправа супроводжується коротким описом мети, детальним покроковим алгоритмом дій і закликом до виконання. Такий формат знижує бар’єр входу: користувач не витрачає час на пошук інструкцій чи інтерпретацію методики, а одразу переходить до практики.

У результаті, розроблено інформаційне забезпечення системи для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії, де збір даних поєднується з глибинним аналізом когнітивних схем і практичним їх коригуванням. Цей замкнений цикл «вимірювання – аналіз – дії – повторна оцінка» створює системний підхід до самоспостереження й самодопомоги, що

лежить в основі когнітивно-поведінкової терапії та сприяє поступовому поліпшенню психоемоційного благополуччя.

2.2 Розробка моделі застосунку

Модель застосунку побудована за принципом чіткого розмежування шарів презентації, бізнес-логіки та зберігання даних. Інтерфейс користувача представлений окремими UI-модулями (чат-інтерфейс, трекер настрою, журнал думок і бібліотека вправ), кожен з яких відповідає за збір вихідних даних та відображення результатів відповідних сервісів. Зв’язок між UI та бекендом здійснюється через залежності «depends», що гарантує, що кожний інтерфейсний елемент звертається лише до свого призначеного сервісу.

Сервісний шар включає şість ключових компонентів: Authentication Service відповідає за автентифікацію та авторизацію користувача; Mood Service опрацьовує й зберігає оцінки настрою; Journal Service керує операціями з Thought Journal, зокрема записом подій і автоматичних думок; Exercise Management Service контролює перелік доступних вправ та фіксує параметри їх виконання. Ці сервіси взаємодіють із загальною базою даних, забезпечуючи централізоване й узгоджене сховище всіх записів.

Аналітичні можливості реалізовані через NLP Analysis Service і Recommendation Service. Перший компонент виконує лінгвістичний аналіз тексту Thought Journal, виокремлює тригерні слова й оцінює емоційний спектр запису, а результати своєї роботи зберігає в базі даних. Recommendation Service аналізує історію змін настрою, результати щоденника та ефективність попередніх вправ, формуючи персоналізовані рекомендації щодо наступних кроків у терапевтичному процесі. Обидва сервіси також використовують єдине сховище для накопичення й використання аналітичних даних.

Notification Service відповідає за генерацію й доставку повідомлень користувачеві — від нагадувань про оновлення настрою до запрошень виконати рекомендовану вправу. Він отримує вхідні дані від Recommendation Service та стан з бази даних, формує контекстуальні пуш-сповіщення й технічні повідомлення.

Така організація комунікацій гарантує, що кожне повідомлення є максимально своєчасним та релевантним особистим паттернам користувача.

Модель застосунку наведена на рисунку 2.1.

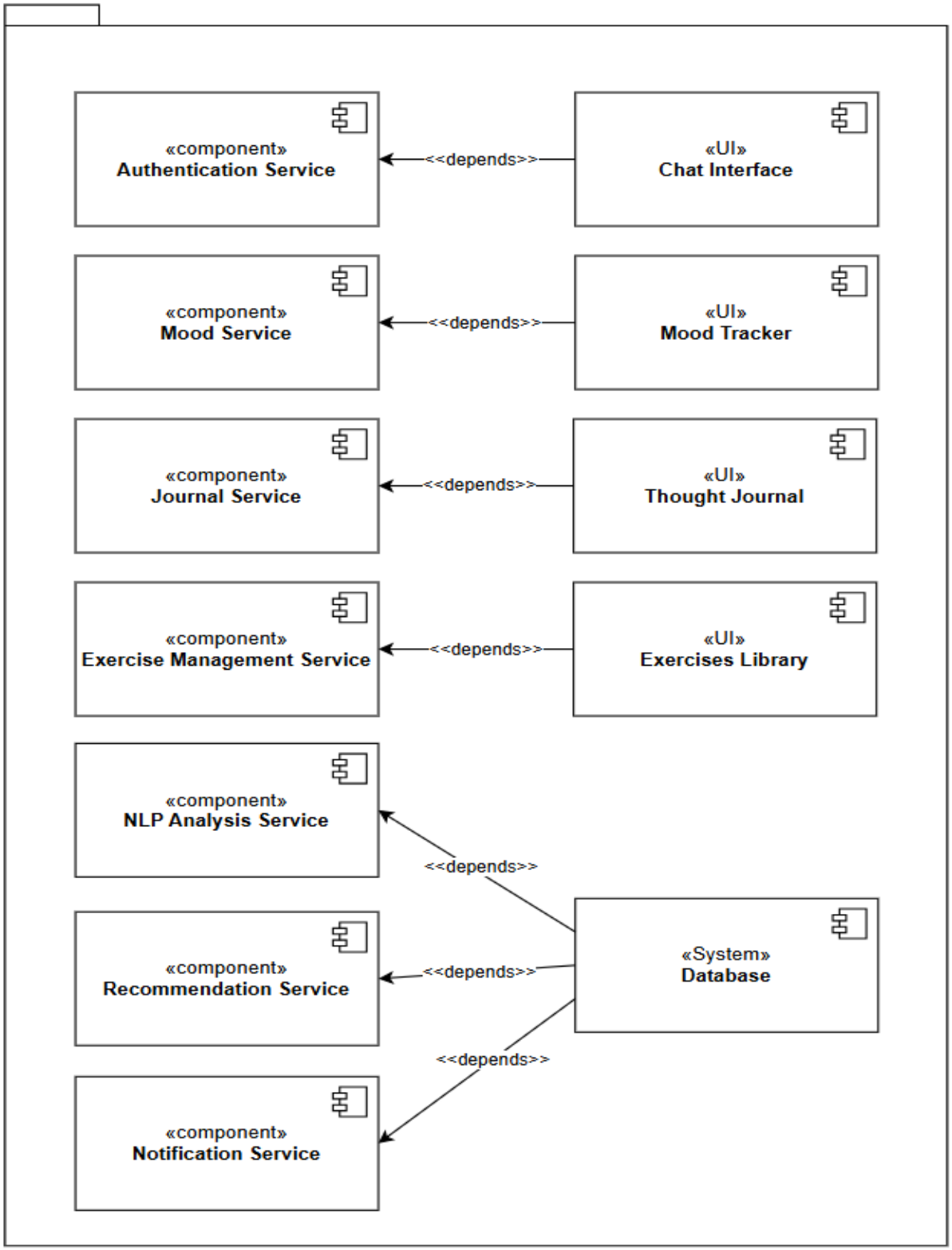

Рисунок 2.1 — Модель застосунку

Отже, запропонована модель застосунку забезпечує високу модульність, спрощує розгортання та тестування окремих компонентів і сприяє гнучкій масштабованості. Чітке визначення відповідальності кожного модуля полегшує подальший розвиток системи — від інтеграції нових аналітичних алгоритмів до впровадження зовнішніх API — без необхідності змінювати суміжні частини архітектури [].

2.3 Розробка методу розширеного журналу думок із автоматичним аналізом

Розширений журнал думок містить в собі не лише електронну форму для запису думок, а й аналітичну система, яка підтримує користувача на кожному кроці переосмислення негативних переконань. Поєднання класичної покрокової методики когнітивно-поведінкової терапії із сучасними інструментами обробки природньої мови дозволяє не лише вручну фіксувати події та емоції, а й автоматично виділяти «тригерні» слова, оцінювати справжню інтенсивність емоцій та шукати в історії щоденника реальні успішні приклади контраргументації.

Такий метод корисний тим, що знижує суб’єктивність самооцінки: користувач бачить явні докази власних досягнень і реальні факти, які спростовують шаблонні негативні думки. Унікальність методу полягає в інтегрованому аналізі — одночасно враховується і текстова інформація, й попередній досвід користувача, що забезпечує персоналізований і глибокий зворотний зв’язок.

Виконання цього методу складається з таких кроків:

1. Запит на введення даних:

1.1. Опис ситуації. Користувачу пропонується велике текстове поле з підказкою «Опишіть, що сталося», де він може вільно викласти подробиці події або ситуації.

1.2. Фіксація автоматичних негативних думок. Після опису ситуації користувач бачить окреме поле «Я думав(ла), що…», у якому є зразкові шаблони

(«Мені здалося, що…», «Я відчував(ла), що…»). Ці підказки допомагають уникнути розмитих формулювань і спрямовують на конкретні когнітивні перекручення.

1.3. Оцінка інтенсивності емоцій. Далі користувач обирає рівень емоційного дискомфорту на шкалі від 1 до 10, де 1 — майже непомітний дискомфорт, а 10 — найбільш інтенсивний. Одразу поруч під шкалою показується опис кожного рівня («майже не відчуваю», «дуже тривожно» тощо), щоб оцінка була коректною.

2. NLP-підготовка тексту:

2.1. Токенізація й лематизація. Введений текст спочатку розбивається на токени за допомогою морфологічного аналізатора, після чого кожен токен приводиться до початкової форми (леми). Це необхідно, щоб «робив», «робити», «роблю» вважалися одним маркером.

2.2. Виявлення «тригерних» слів та фраз. Система порівнює отримані леми зі словником когнітивних викривлень та негативних маркерів. Крім одиничних слів, алгоритм шукає й дво- чи трисловні сполучення, що часто вказують на глобальні узагальнення.

2.3. Формування списку маркерів негативу. Всі знайдені одиночні слова та фрази групуються в єдиний перелік «тригерів», який потім використовується для оцінки емоційної інтенсивності. Кожен маркер супроводжується інформацією про тип викривлення.

3. Оцінка емоційної інтенсивності:

3.1. Розрахунок ваги кожного маркера. Кожному слову чи фразі зі списку призначається числове значення відповідно до потужності його емоційного забарвлення: від 0.5 (легкий негатив) до 2.0 (сильно виражене викривлення).

3.2. Підсумовування в NLP-рахунок (числовий показник, який відображає емоційну інтенсивність або негативне забарвлення тексту, введеного користувачем). Алгоритм підсумовує всі ваги, щоб отримати загальний NLP-рахунок інтенсивності, і нормалізує його в діапазоні 0–10. Це дає змогу порівнювати автоматичний результат із самостійною оцінкою користувача.

3.3. Порівняння з оцінкою користувача. Система зіставляє NLP-рахунок із

значенням оцінки інтенсивності емоцій. Якщо розбіжність перевищує певний поріг (наприклад, ≥2 бали), користувачеві пропонується уточнити одну з оцінок або пояснити, чому автоматична система бачить емоції сильнішими/слабшими.

4. Пошук контраргументів у базі записів:

4.1. Відбір релевантних записів думок. Використовуючи семантичний пошук, система знаходить у історії щоденника 2–3 записи, де користувач успішно спростовував схожі негативні думки.

4.2. Підготовка блоків з доказами. Для кожного знайденого запису генерується короткий витяг: опис успішного кейсу, конкретні факти та висновок, який розбиває неправильне переконання.

5. Формування та відображення зворотного зв’язку:

5.1. Підсумкова звітна картка. У спеціальному вікні відображається три секції: список тригерів із короткими поясненнями, графік порівняння NLP-рахунку та самооцінки, а також блок із контраргументами.

5.2. Інтерактивне представлення. Кожний елемент звіту клікабельний: користувач може розгорнути деталі по конкретному маркеру чи прикладу з запису думок, а також миттєво додати власні коментарі або уточнення.

6. Збереження оновленого запису:

6.1. Додавання альтернативної думки. Після ознайомлення зі звітом користувач формулює та записує більш збалансовану або позитивну альтернативу своїм початковим думкам.

6.2. Збереження метаданих. Система зберігає новий запис думок із повним комплектом метаданих: дату й час створення, NLP-рахунок, список маркерів, обрані контраргументи та текст альтернативної думки. Це забезпечує повну прозорість і можливість подальшого аналізу прогресу.

Отже, автоматичний аналіз журналу думок допомагає користувачеві всебічно усвідомити власні когнітивні викривлення та навчитися шукати реальні докази для їхнього спростування.

2.4 Розробка методу динамічного добору вправ за показниками користування

Динамічний добір вправ – це адаптивна система, яка на основі реальних даних про настрій і ефективність попередніх технік формує персоналізований план подальших дій. Замість загального списку вправ користувач отримує ті, що довели свою користь саме для нього, з урахуванням ступеня покращення настрою та рівня залученості.

Суть методу полягає в тому, що кожна вправа оцінюється за двома головними показниками: середньою зміною настрою «до/після» та частотою виконання. Унікальність підходу полягає у поєднанні цих метрик із контекстом поточного емоційного стану, що дозволяє адаптувати план сесій когнітивно-поведінкової терапії у реальному часі та максимізувати терапевтичний ефект.

Робота методу динамічного добору вправ за показниками користування складається з таких кроків:

1. Збір вхідних даних. На цьому етапі система акумулює всі необхідні показники, щоб мати цілісне уявлення про поточний стан користувача та його взаємодію з вправами.

1.1. Остання оцінка настрою. Після кожної вправи або запису думок користувач оцінює свій емоційний стан за шкалою 1–10. Цей показник зберігається з позначкою дати й часу та служить найактуальнішим маркером загального самопочуття.

1.2. Результат останнього Thought Journal.

- Вичислюється кількість «тригерів» та різниця між «до» і «після» оцінками емоцій (ΔTJ = M_after_TJ – M_before_TJ).

- Додатково зберігаються NLP-рахунки інтенсивності та перелік виявлених когнітивних викривлень, щоб алгоритм міг врахувати глибину пропрацьованих переконань.

1.3. Історія виконаних вправ.

- Ведеться журнал, де для кожної вправи фіксуються, тип, дата й час виконання.

- Оцінки настрою до й після (M_before_i, M_after_i).

- Дані зберігаються у часовому порядку й доступні для статистичної обробки.

2. Аналіз ефективності вправ

Алгоритм розраховує, наскільки кожна техніка сприяє поліпшенню настрою, і наскільки користувач залучений у її виконання.

2.1. Обчислення ΔM_i для кожної вправи.

- Для кожного запису вправи і обчислюється різниця $\Delta M_i = M\_after_i - M\_before_i$.

- Значення може бути негативним (погіршився стан), нульовим або позитивним.

2.2. Розрахунок коефіцієнта залученості E_i.

- E_i = (кількість виконань вправи і за останні N днів) / N.

- Цей коефіцієнт показує регулярність практики: 1.0 означає щоденне виконання, 0.0 – жодного разу.

2.3. Визначення «успішних» вправ.

- Вправа і вважається успішною, якщо $\Delta M_i \geq$ порогове значення (наприклад, +1 бал).

- Усі ΔM_i з негативним або нульовим результатом позначаються як менш релевантні для поточного стану.

3. Сегментація стану - розподіл на категорії необхідний, щоб зрозуміти, яких саме вправ потребує користувач.

3.1. Класифікація поточного настрою:

- S = «Висока тривога», якщо остання оцінка настрою 1–3.

- S = «Нормальний», якщо 4–7.

- S = «Покращення», якщо 8–10.

3.2. Визначення пріоритетних категорій вправ:

- Для «Високої тривоги» пріоритет — дихальні вправи та прогресивна релаксація.

- Для «Нормального» — журнал думок і планування активностей.

- Для «Покращення» — вправи на збереження досягнутого (Gratitude Journal, Mindful Walking).

4. Ранжування та відбір. На цій стадії система відбирає найефективніші та найрелевантніші вправи.

4.1. Формування списку кандидатів.

- Беруться всі вправи з пріоритетної категорії, що мають хоча б одне виконання за останні M днів.

4.2. Обчислення сумарного балу P_i. Для кожної вправи i розраховується $P_i = w_1 \cdot \Delta\bar{M}_i + w_2 \cdot E_i$, де $\Delta\bar{M}_i$ — середнє значення всіх ΔM для вправи i, w_1 і w_2 — ваги (наприклад, 0.7 і 0.3), що відображають пріоритет ефективності перед частотою.

4.3. Відбір топ-3 вправ:

- Сортування кандидатів за P_i у спадаючому порядку.

- Вибір перших трьох для фінальних рекомендацій.

5. Система готує зрозумілі та дієві поради, готові до негайного виконання.

5.1. Підготовка карток. Для кожної з трьох вправ генерується картка з:

- Назвою та коротким описом методики.

- Середнім $\Delta\bar{M}_i$ та E_i із поясненням «очікуваного ефекту».

5.2. Відображення інтерактивних карток. Картки сформовані у вигляді списку або каруселі, де є кнопка «Розпочати вправу» та можливість відкласти на пізніше з автоматичним нагадуванням.

6. Після кожного виконання модель оновлюється, стаючи більш чутливою до потреб користувача.

6.1. Запис нової оцінки «до/після». Після завершення вправи користувач знову вводить M_before і M_after.

6.2. Оновлення ΔM_i та E_i шляхом перерахунку середнього $\Delta\bar{M}_i$ з урахуванням нової сесії, та оновлення коефіцієнта залученості E_i у профілі користувача.

6.3. Перезапуск алгоритму. Модуль рекомендацій автоматично запускається наново, щоб підготувати наступну порцію вправ із урахуванням оновлених показників.

Такий динамічний добір вправ постійно навчається на історії користувача й забезпечує максимально релевантні та ефективні техніки когнітивно-поведінкової терапії. Це підвищує мотивацію до регулярних практик і прискорює досягнення стійкого покращення емоційного стану.

2.5 Розробка діаграм та алгоритмів системи

Діаграма класів ключові типи даних, що лежать в основі роботи системи[14]. MessageType описує структуру кожного повідомлення в чаті (ідентифікатор, вміст, відправник та час). ThoughtType моделює запис автоматичної думки з усіма полями для ситуації, емоцій і альтернативних міркувань згідно з CBT-підходом.

Діаграма класів наведена на рисунку 2.2.

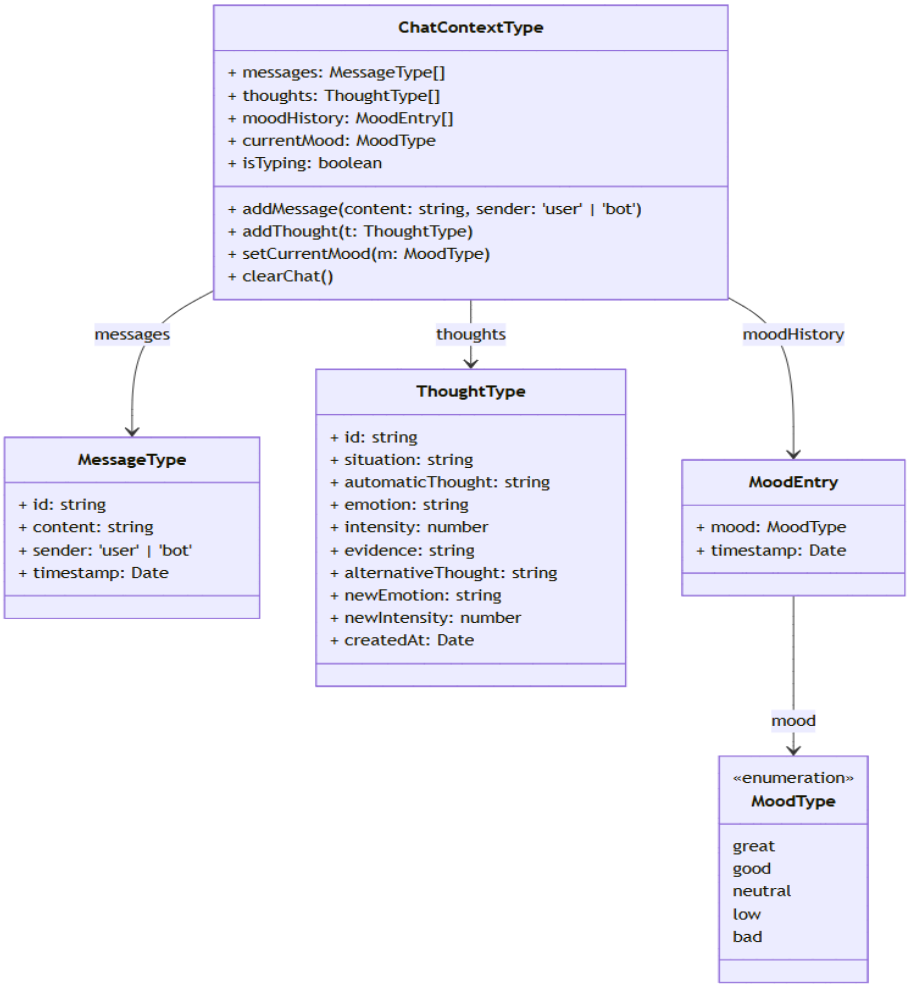

Рисунок 2.2 — Діаграма класів

Для відстеження настрою використовується MoodEntry, який зв’язує значення MoodType (перерахування із п’яти станів: great, good, neutral, low, bad) із часовим штампом. Центральним елементом діаграми є ChatContextType – контекст, який містить масиви повідомлень, думок і записів про настрій, а також методи addMessage(), addThought(), setCurrentMood() і clearChat().

Діаграма компонентів демонструє те, як організовано UI-шари та логіку взаємодії в системі.

Діаграма компонентів системи наведена на рисунку 2.3.

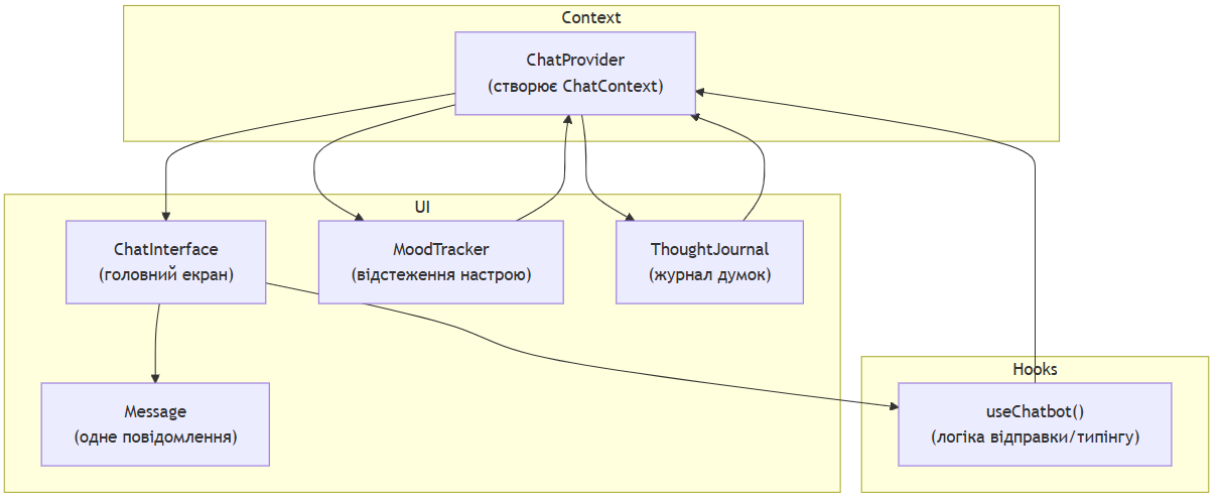

Рисунок 2.3 — Діаграма компонентів

У контейнері Context розташований ChatProvider, який забезпечує створення і надання ChatContext усім дочірнім компонентам. У Hooks знаходиться useChatbot – користувацький хук, що інкапсулює асинхронне надсилання повідомлень, емуляцію набору тексту та зміну стану isTyping. В розділі UI представлені чотири компоненти:

1. ChatInterface – головний екран із полем вводу, листом повідомлень і кнопками керування.

2. Message – відображає одне повідомлення з відповідним форматуванням.

3. MoodTracker – інтерфейс для вибору настрою та візуалізації історії за останні дні.

4. ThoughtJournal – форма для додавання й перегляду щоденних автоматичних думок.

Компоненти інтерфейсу споживають контекст через useChatContext, ChatInterface делегує логіку хуку useChatbot, а сам хук взаємодіє з ChatProvider, викликаючи методи для збереження повідомлень і керування статусом набору. Це чітко відокремлює стан (Context), бізнес-логіку (Hooks) та відображення (UI).

Алгоритм емуляції набору тексту покращує користувацький досвід. Після виклику simulateTyping(text) спочатку встановлюється прапорець isTyping у true, що може покращити інтерфейс користувача, показавши анімацію трьох крапок або індикатор набору. Далі розраховується індивідуальна затримка, яка складається з базового часу та випадкового додатку, пропорційного довжині тексту.

Блок-схема алгоритму емуляції набору тексту наведена на рисунку 2.4.

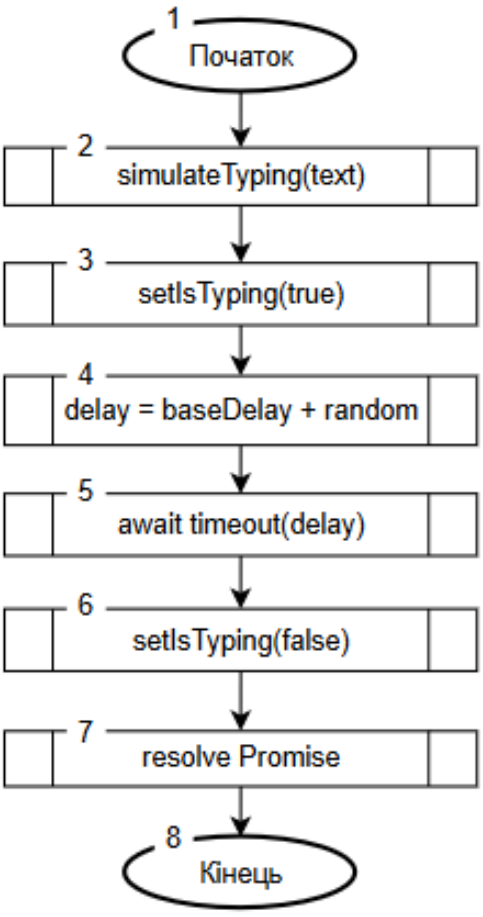

Рисунок 2.4 — Блок-схема алгоритму емуляції набору тексту

Використовуючи await timeout(delay), система чекає цей проміжок, і тільки після завершення паузи змінює isTyping на false та повертає керування в handleSendMessage. Завдяки такому підходу відповідь відображається не одразу, а з паузою, що додає відчуття природності та знижує відчуття «штучності» інтерфейсу.

Діаграма потоку відправки повідомлення ілюструє послідовність дій від моменту, коли користувач натискає кнопку відправлення, до моменту виведення на екран готової відповіді системи.

Діаграма потоку відправки повідомлення наведена на рисунку 2.5.

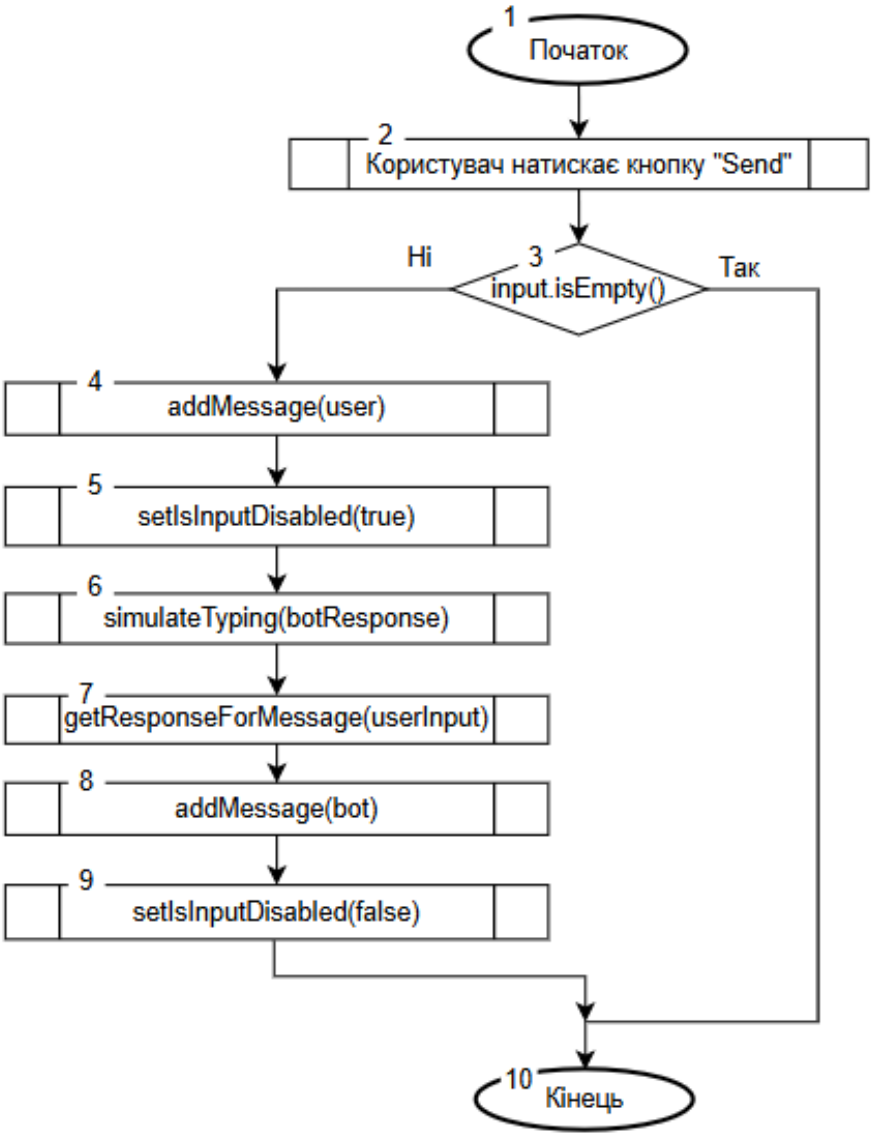

Рисунок 2.5 - Діаграма потоку відправки повідомлення

Спочатку відловлюється клік по кнопці «Send» і виконується перевірка, чи поле вводу містить текст. Якщо поле пусте, алгоритм жодних змін не робить та повертається в початковий стан, гарантувавши, що не буде надсилатися порожнє повідомлення. У разі наявності вводу одразу викликається метод addMessage(user), який додає нову репліку користувача у внутрішній масив messages контексту.

Після цього поле вводу блокується (setIsInputDisabled(true)), щоб запобігти повторним натисканням під час обробки. Далі запускається асинхронна емуляція набору тексту ботом через виклик simulateTyping, і тільки друк завершено, викликається функція getResponseForMessage(userInput), яка генерує відповідь. Отриману відповідь додають у список повідомлень бота (addMessage(bot)), а поле вводу розблоковують (setIsInputDisabled(false)), повертаючи інтерфейс до стану готовності прийняти нові запити. Такий підхід забезпечує зручний користувацький досвід: користувач бачить, що повідомлення обробляється, і не може випадково створити дублікати.

Алгоритм роботи журналу думок відповідає за роботу з журналом автоматичних думок. При кліку на кнопку «+Thought» відкривається форма, в якій користувач заповнює поля: ситуацію (situation), автоматичну думку (automaticThought), емоцію (emotion), інтенсивність (intensity), докази (evidence), альтернативну думку (alternativeThought) та нову емоцію (newEmotion). Після заповнення всіх полів та натискання «Submit» виконується валідація: якщо хоча б одне ключове поле порожнє, запис не додається, що запобігає збереженню неповних даних.

Якщо перевірка успішна, викликається контекстний метод addThought, який приймає об’єкт типу ThoughtType і додає його до масиву thoughts контексту. Після цього форма автоматично очищується та закривається, забезпечуючи готовність до наступного запису. Такий підхід гарантує структурованість даних та їхню відповідність вимогам методам когнітивно-поведінкової терапії.

Блок-схема алгоритму роботи журналу думок наведено на рисунку 2.6.

Рисунок 2.6 — Блок-схема алгоритму роботи журналу думок

Таким чином, розроблені алгоритми забезпечують основний функіонал системи для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії, дозволяють вести журнал думок, відправляти повідомлення штучному інтелекту.

2.6 Висновки

У другому розділі було проведено аналіз інформаційного забезпечення програмної системи діалогової взаємодії для надання психологічної підтримки, розроблено модель системи, яка наводить технології та модулі, які розроблені для роботи системи, взаємозв’язки між ними. Розроблено алгоритми роботи системи, розроблено метод розширеного журналу думок із автоматичним аналізом та метод динамічного добору вправ за показниками користування.

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

3.1 Порівняння та обґрунтування вибору мови програмування розробки

Для вибору мови програмування розробки розглянемо такі мови програмування: TypeScript, JavaScript, Python.

TypeScript [15] — це мова програмування з відкритим кодом, розроблена та підтримувана Microsoft, яка є надмножиною JavaScript. Вона додає статичну типізацію до JavaScript, що дозволяє виявляти помилки на етапі компіляції та забезпечує кращу підтримку IDE з функціями автозаповнення та рефакторингу коду. На рисунку 3.1 наведена архітектура TypeScript.

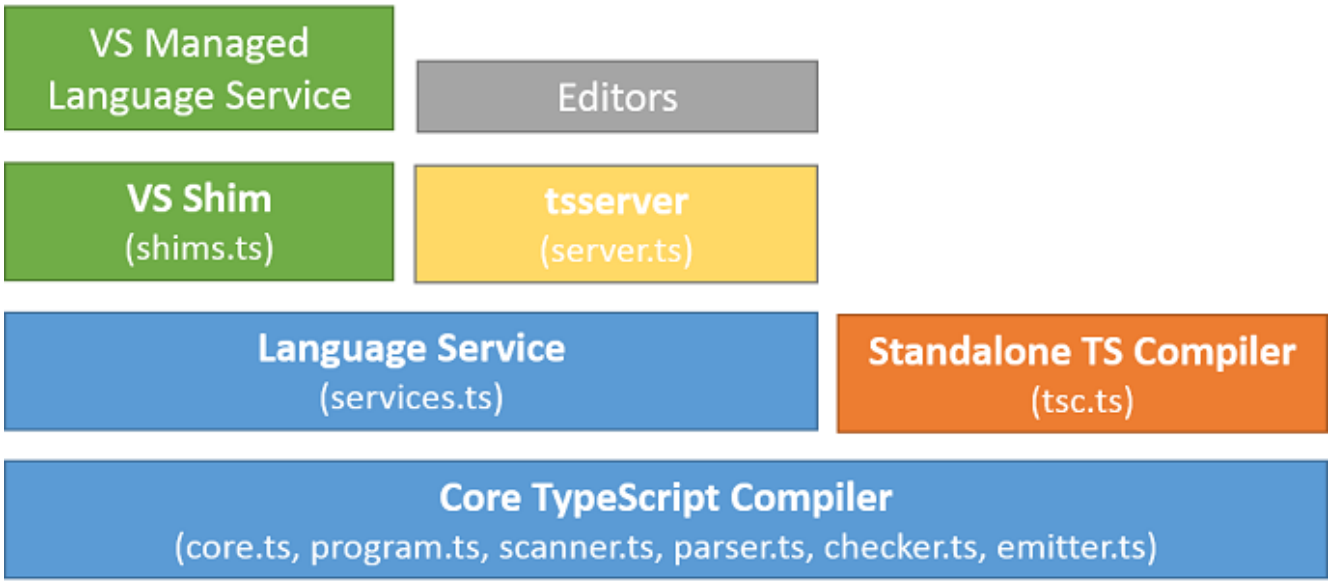

Рисунок 3.1 — Архітектура TypeScript

TypeScript особливо корисний для великих додатків, де типізація покращує читабельність коду, полегшує його підтримку та допомагає уникнути багатьох помилок. Додаток працює на всіх браузерах та платформах, оскільки компілюється у звичайний JavaScript, при цьому зберігаючи всі переваги сучасного ECMAScript.

JavaScript [16] — це інтерпретована мова програмування, яка є основною

мовою для веб-розробки. Вона дозволяє реалізувати складну функціональність на веб-сторінках, робить вміст інтерактивним і динамічним. JavaScript є мовою з динамічною типізацією та підтримує функціональне та об'єктно-орієнтоване програмування.

Незважаючи на свої переваги, JavaScript має обмеження, особливо коли мова йде про великі проекти. Відсутність статичної типізації може призвести до складних для діагностики помилок, а динамічна природа мови може ускладнити рефакторинг та підтримку коду [17].

Python [18] — це високорівнева мова програмування загального призначення з наголосом на читабельність коду. Python підтримує множинні парадигми програмування, включаючи структуроване, об'єктно-орієнтоване та функціональне програмування. Вона має динамічну типізацію, але пропонує можливості анотацій типів.

Архітектура Python наведена на рисунку 3.2.

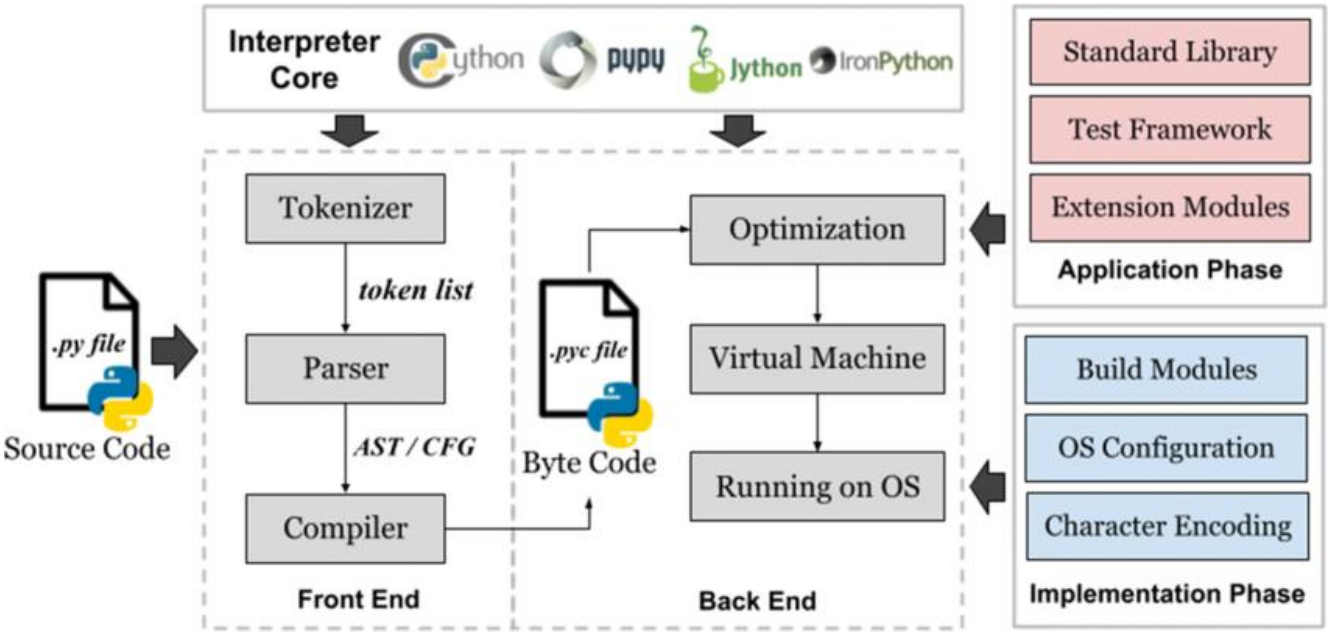

Рисунок 3.2 — Архітектура Python

Python широко використовується у веб-розробці, аналізі даних, штучному інтелекті, наукових обчисленнях та автоматизації. Однак Python не виконується нативно у браузері і потребує додаткових фреймворків для веб-розробки, таких як

Django або Flask, які більше орієнтовані на серверну частину.

Порівняємо можливості мов програмування у таблиці 3.1.

Таблиця 3.1 — Порівняльний аналіз мов програмування

Характеристика	TypeScript	JavaScript	Python
Статична типізація	+	-	-
Підтримка об'єктно-орієнтованого програмування	+	+	+
Виявлення помилок на етапі компіляції	+	-	-
Нативна підтримка веб-розробки	+	+	-
Підтримка асинхронного програмування	+	+	+
Обробка форм у веб-додатках	+	+	+
Проста інтеграція з бібліотеками інтерфейсу для веб розробки	+	+	+
Сумарний бал	7	5	4

Отже, TypeScript є найкращим вибором для розроблюваного застосунку, оскільки він обробляє складні структури даних (відстеження настрою, записи думок з багатьма полями), де статична типізація TypeScript запобігає помилкам під час виконання.

Для додатку, пов'язаного із психологічним здоров'ям, де точність даних є дуже важливою, здатність TypeScript забезпечувати строгу типізацію допомагає запобігти непомітним помилкам, які могли б вплинути на досвід користувача або цілісність даних.

3.2 Порівняння та обґрунтування вибору середовища розробки

Visual Studio Code [19] — це безкоштовний та легковаговий редактор з відкритим вихідним кодом, що пропонує вбудовану підтримку JavaScript і TypeScript із нативним підсвічуванням синтаксису, інтелектуальним автодоповненням (IntelliSense) і графічним налагоджувачем Node.js. Завдяки широкому ринку розширень можна швидко додати підтримку React, ESLint, Prettier, Docker, REST-клієнта та багатьох інших інструментів. Універсальність і

гнучкість VS Code роблять його популярним вибором для швидкого старту розробки веб-застосунку, але для деяких просунутих можливостей (наприклад, робота з базами даних чи out-of-the-box React-сніпети) доводиться інсталювати сторонні плагіни.

Інтерфейс Visual Studio Code наведено на рисунку 3.3.

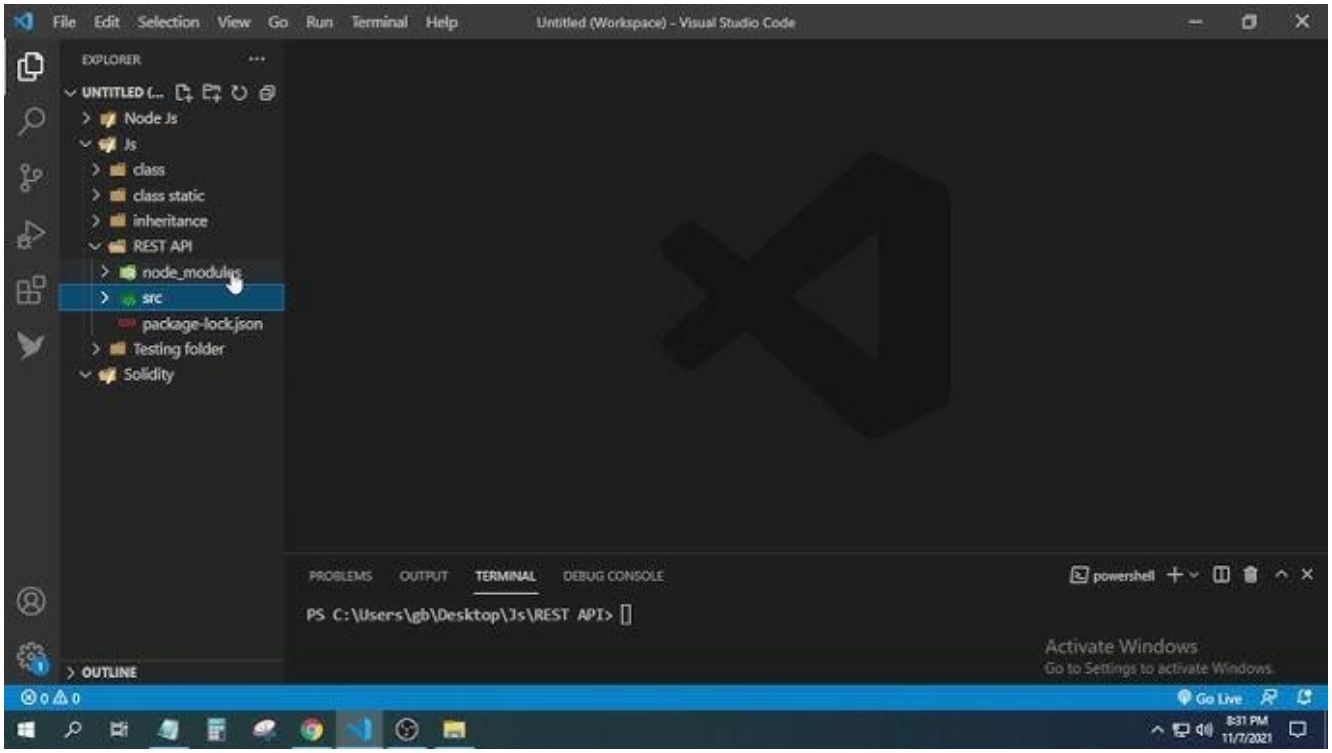

Рисунок 3.3 — Інтерфейс VS Code

WebStorm [20] — комерційне інтегроване середовище розробки, оптимізоване для JavaScript і TypeScript, що надається компанією JetBrains. Воно має вбудовану підтримку React, Angular, Vue, Node.js та популярних фронтенд-фреймворків, інтегрований REST-клієнт, розширений налагоджувач, автоматичний рефакторинг та інструменти контролю версій. WebStorm «із коробки» пропонує готові шаблони для компонентів React і високопродуктивний аналіз коду, що дозволяє підтримувати проєкт структуровано і надійно, але необхідна платна ліцензія.

Інтерфейс Jetbrains Webstorm наведено на рисунку 3.4.

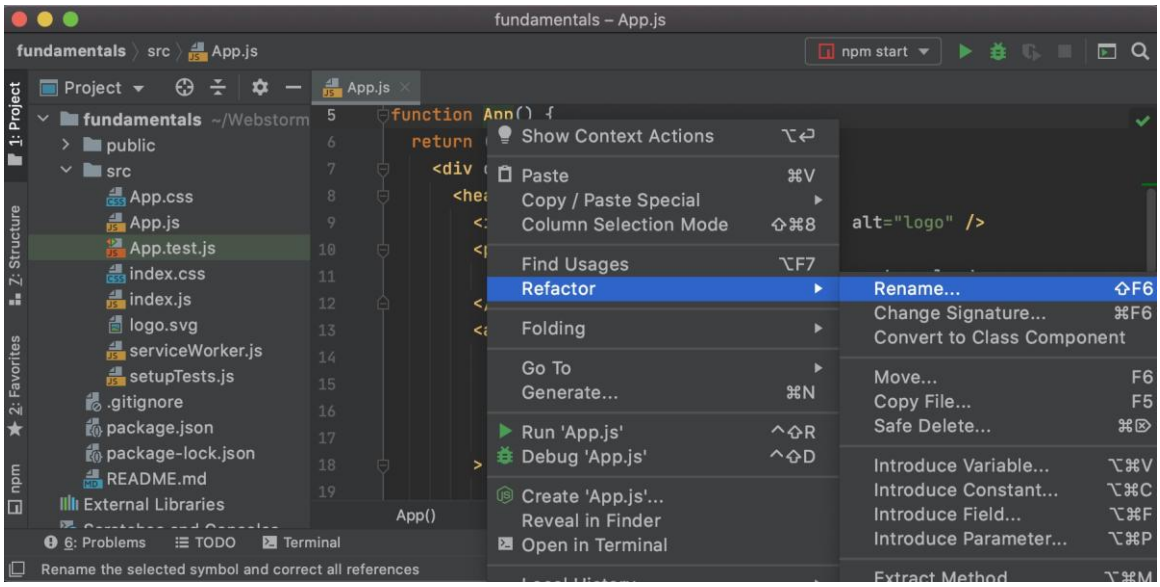

Рисунок 3.4 — Інтерфейс Jetbrains Webstorm

IntelliJ IDEA Ultimate [21] — це повноцінне комерційне IDE для розробки як бекенду (Java, Kotlin, Node.js), так і фронтенду (TypeScript, React).

Інтерфейс IntelliJ IDEA Ultimate наведено на рисунку 3.5.

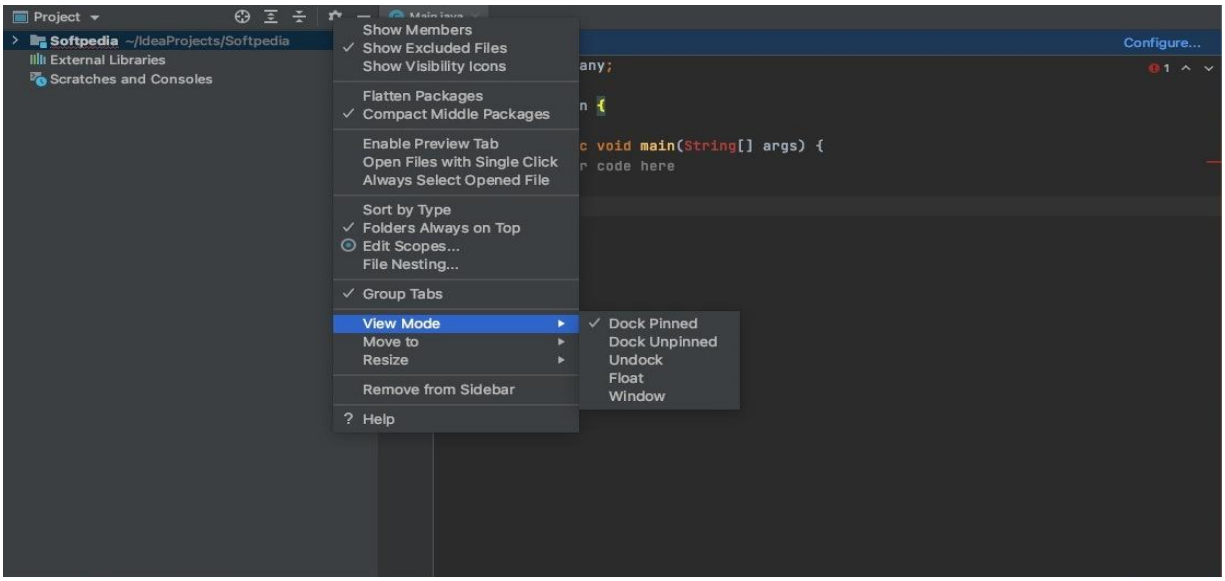

Рисунок 3.5 — Інтерфейс IntelliJ IDEA Ultimate

Воно об’єднує всі переваги WebStorm (поглиблена підтримка JS/TS) з потужними інструментами для роботи з базами даних, вбудованим REST-клієнтом, Docker-плагінами та широкою екосистемою плагінів JetBrains.

IDEA Ultimate підходить для розгортання комплексного веб-застосунком, однак є найдорожчим серед трьох середовищ.

Таблиця 3.2 — Порівняльний аналіз середовищ розробки

Характеристика	VS Code	WebStorm	IntelliJ IDEA Ultimate
Вбудована підтримка TypeScript	+	+	+
Інтегрований налагоджувач Node.js	+	+	+
Вбудована Git-інтеграція	+	+	+
Маркетплейс розширень/плагінів	+	+	+
Вбудований термінал	+	+	+
Безкоштовне використання	+	–	–
Сумарний бал	6	5	5

Таким чином, найкращим вибором для розробки є середовище VS Code, яке має усі переваги, що і аналоги, при цьому є безкоштовним. Тому прийнято рішення використовувати середовище розробки VS Code.

3.3 Проектування бази даних застосунку

База даних спроектована для підтримки ключових функцій системи: автентифікації та профільного обліку користувачів, збереження повного журналу діалогів із ботом, фіксації когнітивних записів у форматі CBT, моніторингу настрою, а також організації й відстеження виконання тематичних вправ. Кожна таблиця відповідає окремому доменному об’єкту й містить поля, необхідні для коректного зберігання й подальшої аналітики даних [22].

Таблиця users зберігає облікові записи користувачів системи. Крім аутентифікаційної інформації (емейл, хеш пароля), тут можуть міститися базові профільні дані (ім’я, дата реєстрації). Надалі до неї можна додавати поля для зберігання аватару, налаштувань або ролей користувача.

Поля таблиці users наведені у таблиці 3.2.

Таблиця 3.2 — Поля таблиці users

Поле	**Тип**	**Опис**
id	UUID	Первинний ключ — унікальний ідентифікатор користувача
email	VARCHAR	Логін/емейл (унікальний)
name	VARCHAR	Ім’я або нікнейм користувача
password_hash	VARCHAR	Хеш пароля
created_at	TIMESTAMP	Час створення облікового запису

Таблиця user_sessions використовується для фіксації кожного сеансу користувача: коли він зайшов у застосунок і коли вийшов. Це дає змогу аналізувати активність — наприклад, загальний час роботи в системі, частоту входів тощо. Поле ended_at може залишатися порожнім, поки сеанс триває.

Поля таблиці user_sessions наведені у таблиці 3.3.

Таблиця 3.3 — Поля таблиці user_sessions

Поле	**Тип**	**Опис**
id	UUID	Первинний ключ — унікальний ідентифікатор сеансу
user_id	UUID	Зовнішній ключ → users.id
started_at	TIMESTAMP	Час початку сеансу
ended_at	TIMESTAMP	Час завершення сеансу (nullable)

Таблиця messages зберігає увесь історичний журнал переписки між користувачем і ботом. Кожен запис містить текст повідомлення, позначку відправника (користувач або бот) і часову мітку. Ці дані можна використовувати для аналізу діалогів, побудови контексту бесіди або відновлення історії при повторному вході.

Поля таблиці messages наведені у таблиці 3.4.

Таблиця 3.4 — Поля таблиці messages

Поле	**Тип**	**Опис**
id	VARCHAR	Первинний ключ — унікальний ідентифікатор повідомлення
content	TEXT	Текст повідомлення
sender	SenderType	Хто відправив: user або bot
timestamp	TIMESTAMP	Час надсилання

Таблиця thoughts реалізує збереження думок користувачів. Користувач фіксує ситуацію, автоматичну думку, емоційний стан до та після переосмислення, а також аргументи на користь і проти первинної думки. Вона є основою для ведення журналу внутрішніх переживань і прогресу в терапевтичній роботі.

Поля таблиці thoughts наведені у таблиці 3.5.

Таблиця 3.5 — Поля таблиці thoughts

Поле	**Тип**	**Опис**
id	VARCHAR	Первинний ключ — унікальний ідентифікатор запису
situation	TEXT	Опис ситуації
automatic_thought	TEXT	Початкова автоматична думка
emotion	TEXT	Початкова емоція
intensity	FLOAT	Інтенсивність початкової емоції (0.0–1.0 або шкала 1–10)
evidence	TEXT	Дані «за» і «проти» для перевірки думки
alternative_thought	TEXT	Альтернативна (реалістична) думка
new_emotion	TEXT	Нова емоція після переосмислення

Продовження таблиці 3.5

new_intensity	FLOAT	Інтенсивність нової емоції
created_at	TIMESTAMP	Час створення запису

Таблиця distortions містить «словник» когнітивних спотворень (наприклад, «катастрофізація», «чорно-біле мислення» тощо). Кожне спотворення має унікальну назву й докладний опис, що використовується для класифікації думок у thought_distortions.

Поля таблиці distortions наведені у таблиці 3.6.

Таблиця 3.6 — Поля таблиці distortions

Поле	**Тип**	**Опис**
id	UUID	Первинний ключ — ідентифікатор спотворення
name	VARCHAR	Назва когнітивного спотворення
description	TEXT	Детальний опис спотворення

Таблиця thought_distortions є зв’язною (багато до багатьох) таблицею між thoughts та distortions. Дозволяє для кожного записаного мисленнєвого патерну вказати одне або кілька спотворень, які до нього застосовані.

Поля таблиці thought_distortions наведені у таблиці 3.7.

Таблиця 3.7 — Поля таблиці thought_distortions

Поле	**Тип**	**Опис**
thought_id	VARCHAR	→ thoughts.id
distortion_id	UUID	→ distortions.id

Таблиця mood_history фіксує періодичні показники настрою користувача (наприклад, щоденні або кілька разів на день). Дає змогу будувати графік змін настрою з часом та виявляти тенденції.

Поля таблиці mood_history наведені у таблиці 3.8.

Таблиця 3.8 — Поля таблиці mood_history

Поле	**Тип**	**Опис**
mood	MoodType	Поточний настрій (great, good, neutral, low, bad)
timestamp	TIMESTAMP	Час фіксації настрою (складовий первинний ключ)

Таблиця exercises містить перелік доступних когнітивних вправ чи завдань у системі, наприклад, техніки релаксації, вправи на зміну мислення, щоденні рефлексії тощо. Користувач може обирати вправи з цього каталогу.

Поля таблиці exercises наведені у таблиці 3.9.

Таблиця 3.9 — Поля таблиці exercises

Поле	**Тип**	**Опис**
id	UUID	Первинний ключ — ідентифікатор вправи
title	VARCHAR	Назва вправи
description	TEXT	Детальний опис вправи
created_at	TIMESTAMP	Дата додавання в каталог

Таблиця completed_exercises фіксує факти виконання вправ користувачем: коли саме він їх виконав і з яким результатом (наприклад, власними коментарями або оцінкою корисності). Це дозволяє відстежувати залученість і прогрес.

Поля таблиці completed_exercises наведені у таблиці 3.10.

Таблиця 3.10 — Поля таблиці completed_exercises

Поле	**Тип**	**Опис**
id	UUID	Первинний ключ — запис виконання вправи
user_id	UUID	→ users.id
exercise_id	UUID	→ exercises.id
completed_at	TIMESTAMP	Час завершення вправи

Продовження таблиці 3.10

Поле	Тип	Опис
user_response	TEXT	Коментарі або відповіді користувача під час вправи

На основі описаних таблиць було спроектовано схему бази даних, яка наведена на рисунку 3.6.

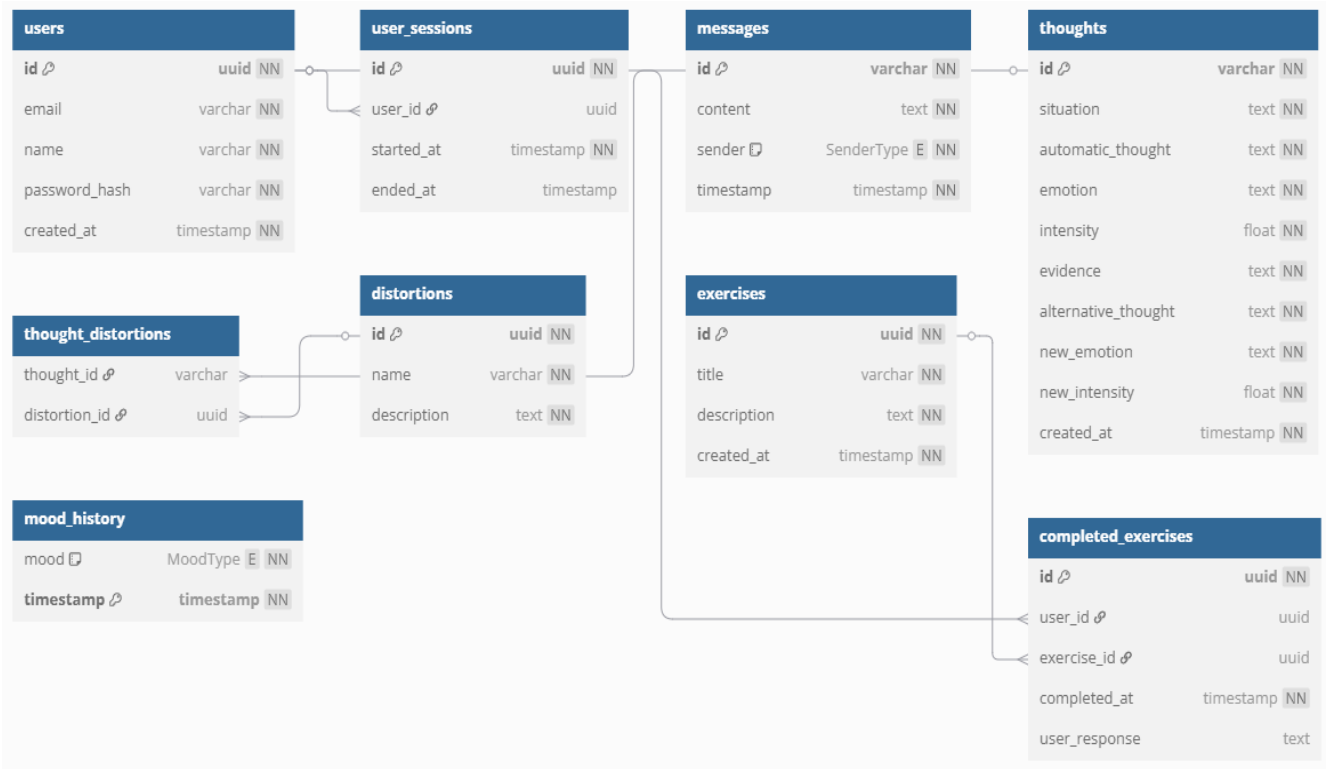

Рисунок 3.6 — Схема бази даних

Кожна з цих таблиць у комбінації забезпечує повноцінну підтримку основних функцій застосунку: ведення діалогів, журналу думок, слідкування за настроєм, а також роботи з набором вправ та аналізом когнітивних спотворень.

3.4 Розробка інтерфейсу користувача

Вікно чату дозволяє вести розмову з чат-ботом, для чого реалізовано поле для введення тексту й кнопку відправлення – отже, користувач може одразу

відповісти, описати свій стан або поставити запитання боту.

Призначення цього вікна – встановити довірливий контакт із користувачем і організувати первинний збір інформації про його емоційний стан у вигляді вільного тексту. Такий «розмовний» підхід знижує бар’єр входу, створює атмосферу підтримки та налаштовує на подальшу роботу з техніками когнітивно-поведінкової терапії.

Структура інтерфейсу вікна чату наведена на рисунку 3.7.

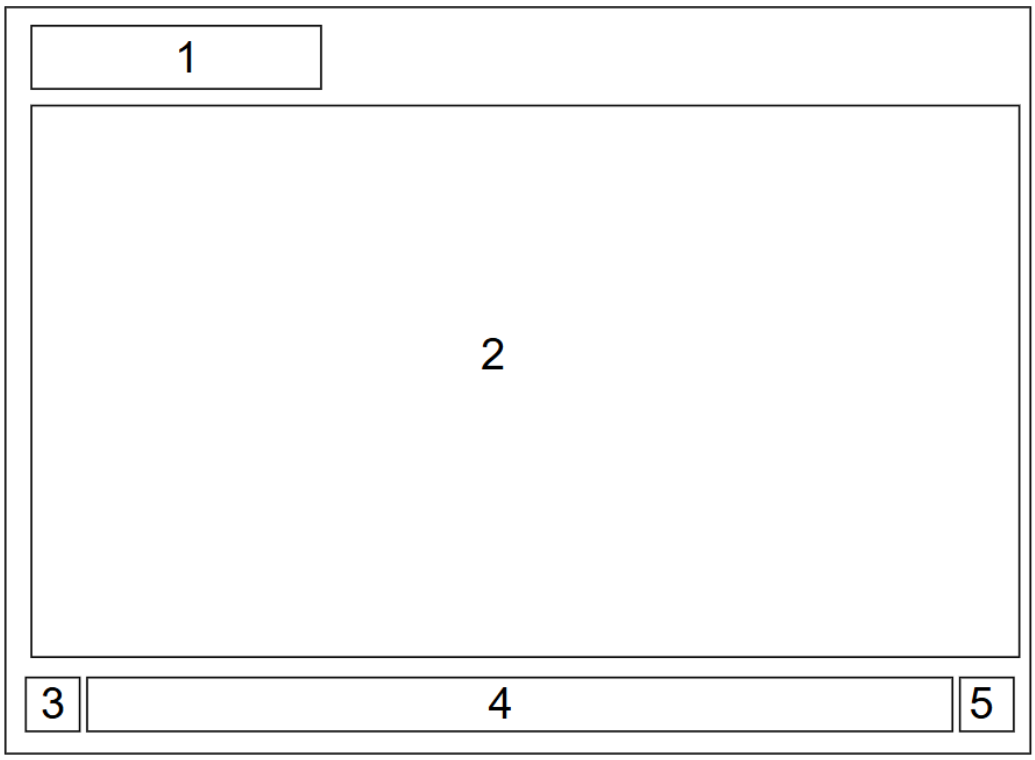

Рисунок 3.7 — Структурна схема інтерфейсу чату

Складові елементи інтерфейсу чату:

1. Назва чату.

2. Область повідомлень.

3. Кнопка для використання смайлів.

4. Поле введення тексту.

5. Кнопка «Відправити».

Трекер настрою надає інтуїтивну шкалу емоцій із п’ятьма піктограмами: від «Great» (великий задоволений стан) до «Bad» (дуже поганий). Користувач клацає

по відповідній іконці, щоб моментально оцінити, як він почувається зараз.

Це вікно дозволяє проводити самоспостереження й одночасно накопичує дані для візуалізації трендів настрою. Регулярні відмітки допомагають виявити як позитивні зрушення, так і потенційні спади, що може стимулювати користувача вчасно звернутися до вправ чи звернутися по додаткову підтримку.

Структура інтерфейсу трекера настрою наведена на рисунку 3.8.

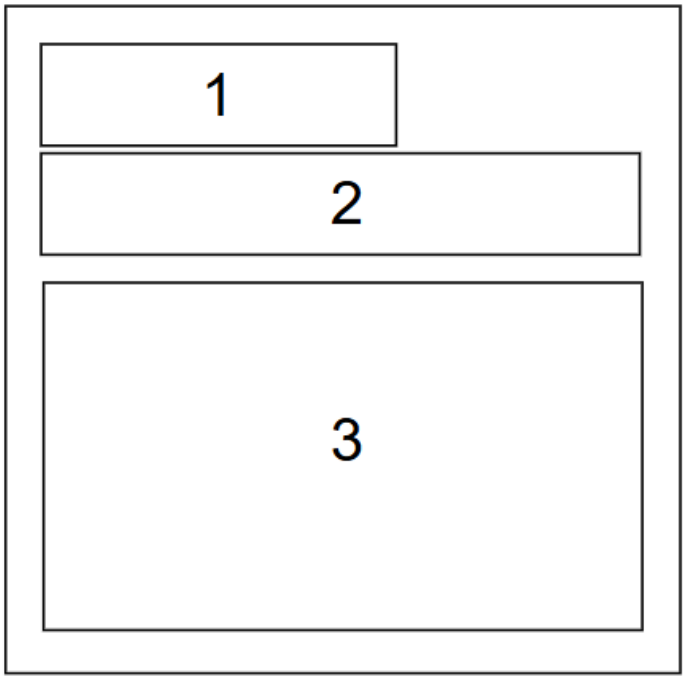

Рисунок 3.8 — Структурна схема інтерфейсу трекера настрою

Складові елементи інтерфейсу трекера настрою:

1. Заголовок трекера настрою.

2. Область оцінювання настрою.

3. Статистика з настроєм за весь час.

Вікно журналу думок дозволяє користувачеві робити опис ситуації, автоматичні негативні думки, первісні емоції й їхню інтенсивність, докази проти цих думок, альтернативні збалансовані думки та повторну оцінку емоцій.

Призначення журналу думок – надати чітку методику запису й аналізу

власних когнітивних викривлень за стандартним алгоритмом когнітивно-поведінкової терапії. Завдяки такій структурі користувач краще усвідомлює свої автоматичні думки, вчиться знаходити аргументи проти них і оцінювати, як це впливає на емоційний стан.

Вікно щоденника думок наведено на рисунку 3.9.

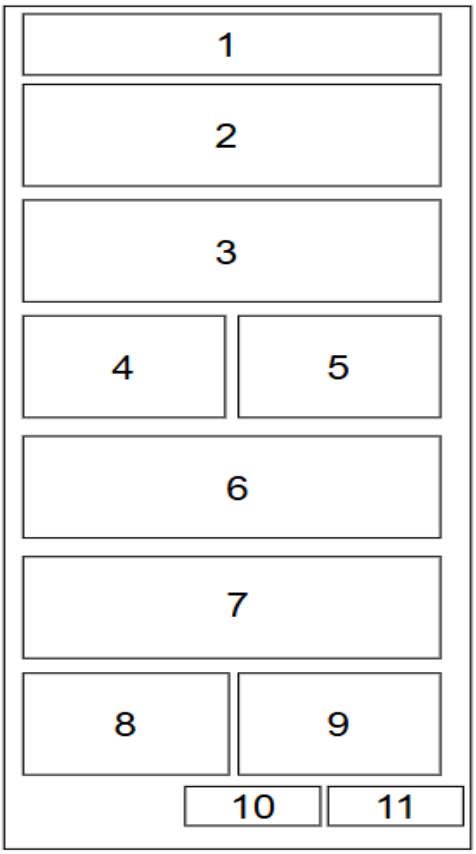

Рисунок 3.9 — Структурна схема інтерфейсу вікна журналу думок

Складові елементи вікна журналу думок:

1. Назва вікна.
2. Поле для опису ситуацію.
3. Поле для «автоматичної думки».
4. Поле для опису емоції.
5. Інтенсивність емоції.
6. Докази проти цієї думки.
7. Альтернативна думка.

8. Нова емоція.

9. Нова інтенсивність.

10. Кнопка «Відмінити».

11. Кнопка «Зберегти».

На рисунку 3.10 наведено інтерфейс запису вправи планування активностей для підвищення настрою й енергії. Кожний крок оформлено як нумерований список: від переліку раніше приємних занять до фіксації дати й часу та оцінки результату.

Мета цього компонента – допомогти користувачеві свідомо включити в свій розклад дії, які приносять радість чи відчуття здобутку, а також відслідкувати їхній вплив на настрій. Такий плановий підхід дозволяє формувати звичку регулярного самопіклування та підвищує ефективність когнітивно-поведінкової терапії через конкретні, вимірювані кроки.

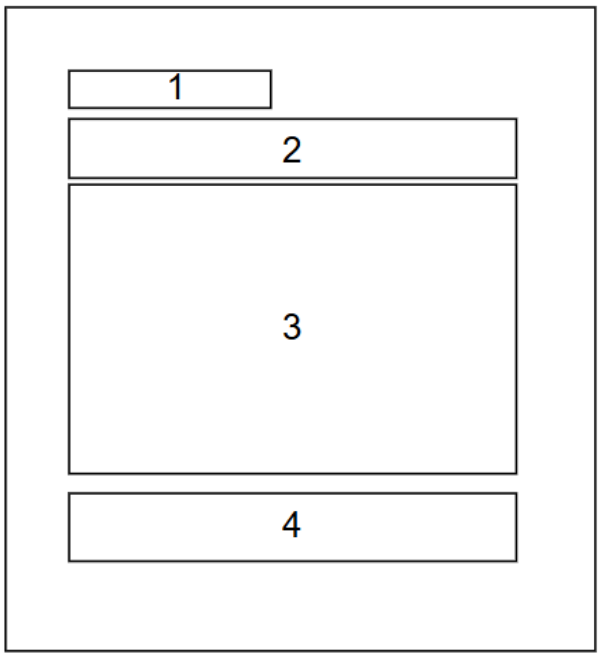

Рисунок 3.10 — Структура інтерфейсу запису вправи

Складові елементи інтерфейсу записи вправи:

1. Назва вправи.

2. Опис вправи.

3. Кроки виконання вправи.

4. Кнопка «Спробувати цю вправу».

Таким чином, розроблений інтерфейс користувача дозволяє проводити швидке самооцінювання настрою, цілеспрямоване планування позитивних активностей через і глибинний аналіз власних думок.

Кожен модуль побудований за чіткою методикою когнітивно-поведінкової терапії: користувач отримує свободу виражати свій стан у тексті, водночас йому пропонується інтуїтивно зрозуміла структура для оцінки емоцій, фіксації автоматичних перекручень і побудови збалансованих альтернатив. Інтерактивні картки вправ і візуалізації трендів настрою формують замкнений цикл «оцінка – аналіз – дія – повторна оцінка», що сприяє усвідомленому самоспостереженню та поступовому покращенню психоемоційного благополуччя.

3.4 Висновки

У третьому розділі було проведено аналіз мов програмування для розробки програмної системи діалогової взаємодії для надання психологічної підтримки, у результаті якого було обрано мову TypeScript. Проведено аналіз середовищ розробки, в результаті якого обрано VS Code. Розроблено інтерфейс системи.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Огляд методів тестування

Кожний сервісний компонент покривається окремими автоматизованими тестами, які перевіряють коректність його внутрішньої логіки. Наприклад, для Recommendation Service — правильність обчислення балів P_i, для Analysis Service — коректне визначення «тригерних» слів і підрахунок ваг. Юніт-тести гарантують, що при зміні коду базового функціоналу не виникатимуть регресії в ядрі бізнес-логіки.

Перевіряється взаємодія між декількома компонентами: наприклад, чи зберігаються записи журналу думок у базу даних і чи відразу доступні для Recommendation Service. Інтеграційні тести можуть підіймати реальну або in-memory БД і проганяти через API-контролери серії запитів, імітуючи поведінку фронтенду [23].

End-to-End тестування охоплює повний цикл роботи інтерфейсу користувача. Інструменти на кшталт Cypress або Playwright імітують дії реального користувача: реєстрацію, оцінювання настрою, заповнення журналу думок, виконання вправ і перевіряють, що під час цього коректно оновлюються графіки та надходять рекомендації. E2E-тести забезпечують впевненість у стабільності всього флоу користувацької взаємодії.

За допомогою JMeter або k6 перевіряють, наскільки швидко обробляються одночасні запити до API, як масштабуються сервіси при сотнях чи тисячах користувачів [24]. Це особливо важливо для модулів, що містять аналіз тексту та побудову персоналізованих звітів.

При тестуванні юзабіліті користувачі або тестувальники вручну проходять через усі функції застосунку, оцінюють зручність інтерфейсу, ясність інструкцій та загальний досвід [25]. Виявляють приховані в UX проблеми, неочевидні помилки в логіці переходів та формулюваннях.

Найбільш доцільним є ручне (експлораторне) тестування, оскільки йдеться передусім про якість користувацького досвіду та коректність роботи інтерфейсних

сценаріїв, які важко автоматизувати повністю через динамічність контенту (наприклад, згенеровані поради, аналіз). Ручне тестування дозволяє імітувати реальні умови використання, оцінити природність «розмови» з ботом, наочність візуалізацій трендів настрою й зручність заповнення журналу думок. Крім того, воно дає змогу оперативно робити скриншоти й демонструвати результати перевірки UI-модулів, що відповідає запиту на наведення скриншотів із описом інтерфейсу. Такий підхід забезпечить глибоке розуміння того, як користувачі справді взаємодіють із застосунком, й дозволить виявити нюанси, які автоматизовані тести можуть пропустити.

4.2 Тестування застосунку

Розпочнемо тестування із чату, у якому користувач може спілкуватися із чат-ботом. Чат для спілкування з чат-ботом наведено на рисунку 4.1.

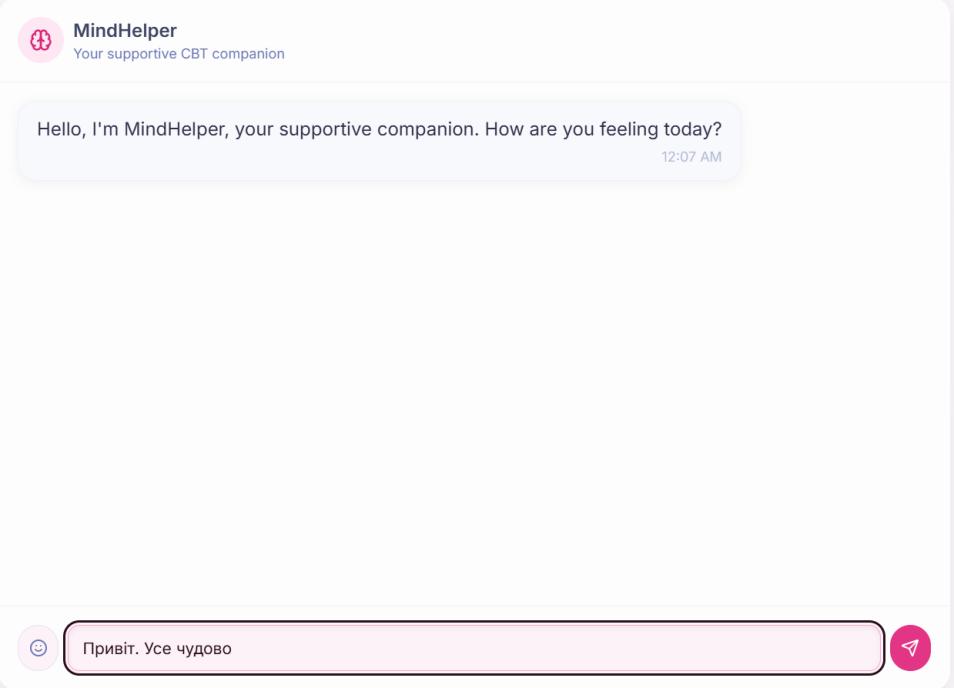

Рисунок 4.1 - Чат для спілкування з чат-ботом

Відправимо повідомлення. Після цього, формується відповідь та надсилається, як на рисунку 4.2. Чат виглядає як спілкування із живим користувачем.

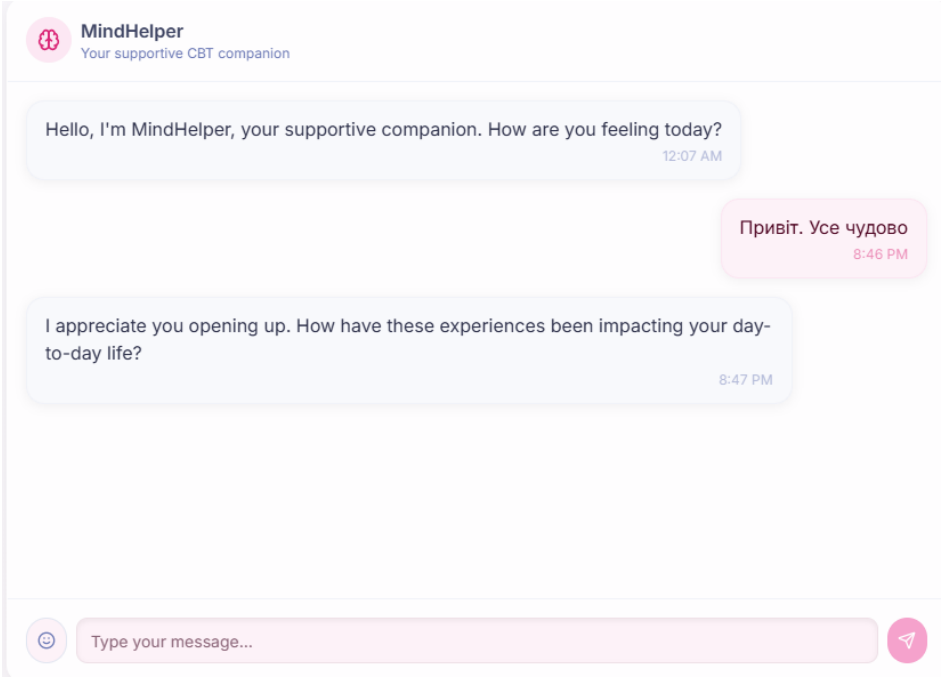

Рисунок 4.2 — Спілкування з чат-ботом

Перейдемо до модуля моніторингу настрою, який продемонстровано на рисунку 4.3. Для визначення свого почуття на сьогодні, натиснемо зображення із піднятим пальцем вверх, який відповідає за прекрасне самопочуття. Ця інформація впливає на статистику настрою протягом усього часу.

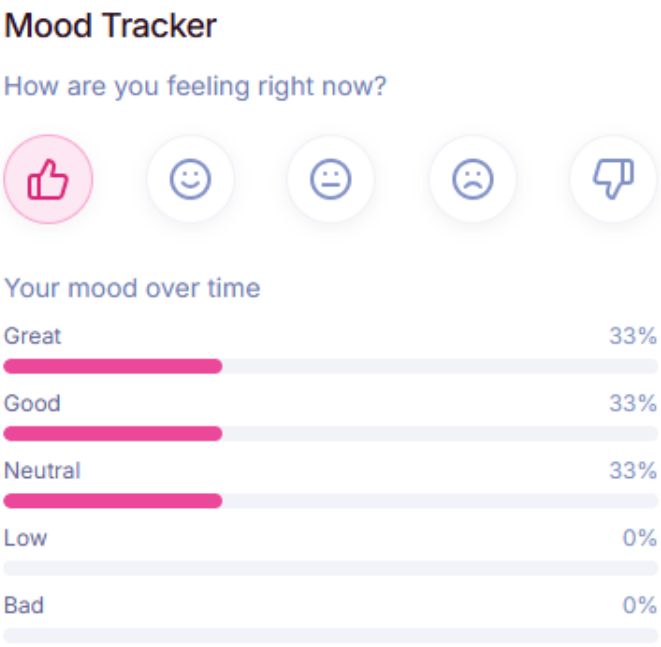

Рисунок 4.3 — Модуль моніторингу настрою

Оберемо також інший емоційний стан, який відповідає за хороше

самопочуття. В результаті, підсвічується новий вибір у списку, а також змінюється статистика настрою за увесь час (рисунок 4.4).

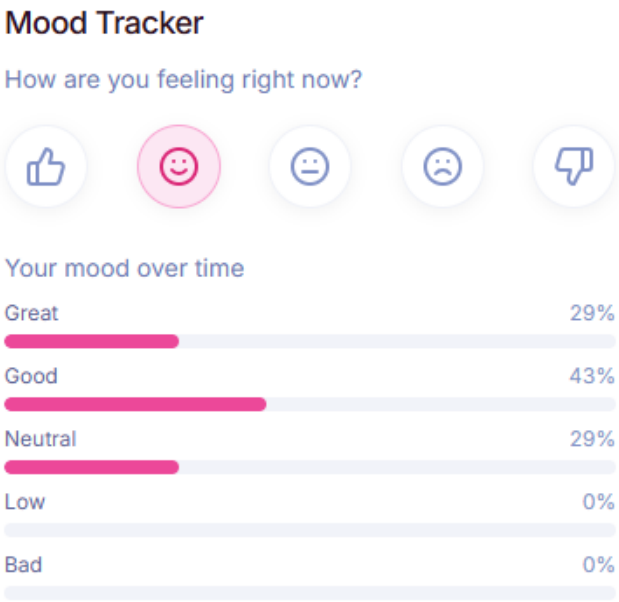

Рисунок 4.4 — Зміна статистики настрою

Перейдемо до журналу думок. Початковий його стан продемонстровано на рисунку 4.5.

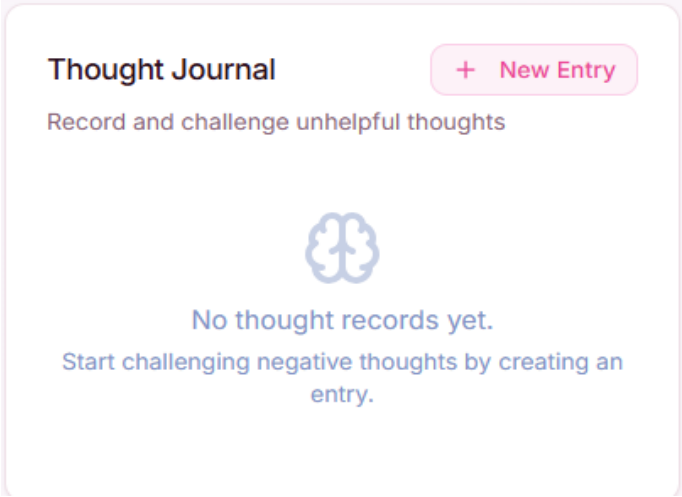

Рисунок 4.5 — Початковий стан журналу думок

Натиснемо “New Entry” для додавання нового запису у журналі думок.

Відкриється анкета, як на рисунку 4.6.

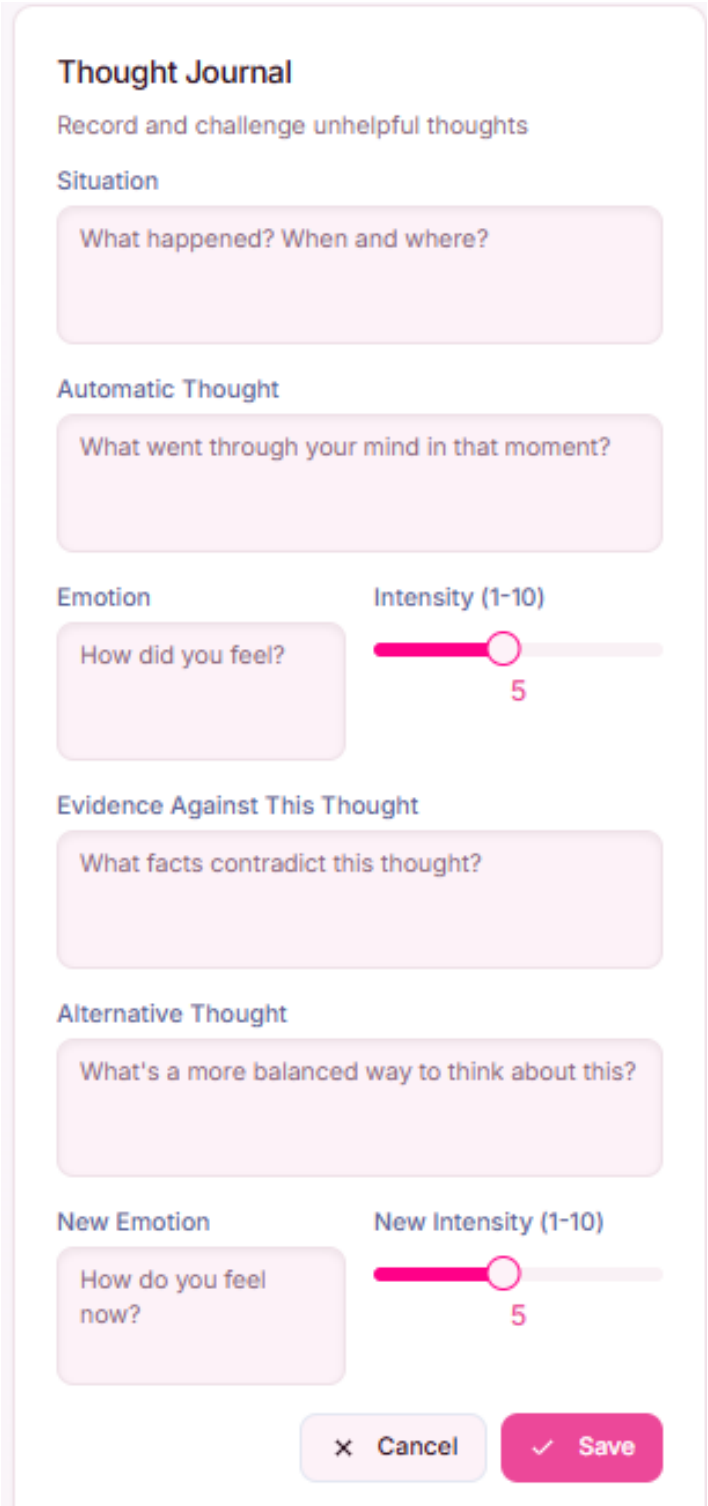

Рисунок 4.6 — Анкета для заповнення журналу думок

Заповнимо цю анкету, як на рисунку 4.7.

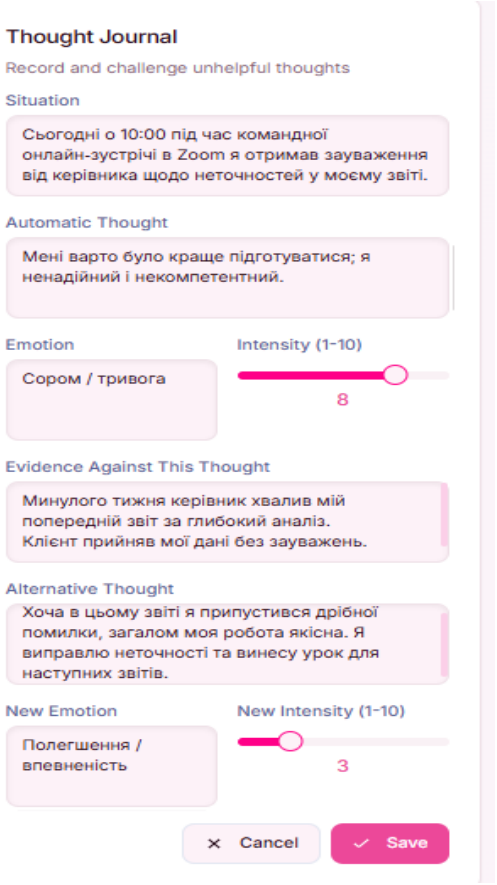

Рисунок 4.7 — Заповнення анкети для запису журналу думок

Збережемо записані дані. У результаті вони відображаються у журналі думок, як на рисунку 4.8. Окрім початкової та кінцевої емоції, відображається також рівень зміни емоційного стану.

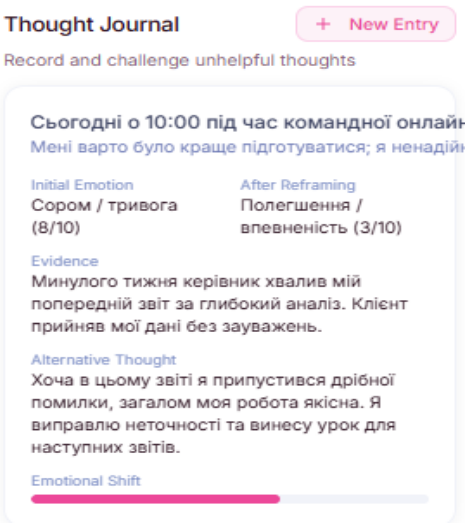

Рисунок 4.8 — Журнал думок

Після введеного запису, на основі отриманих даних, рекомендується виконати вправу для покращення емоційного стану (рисунок 4.9).

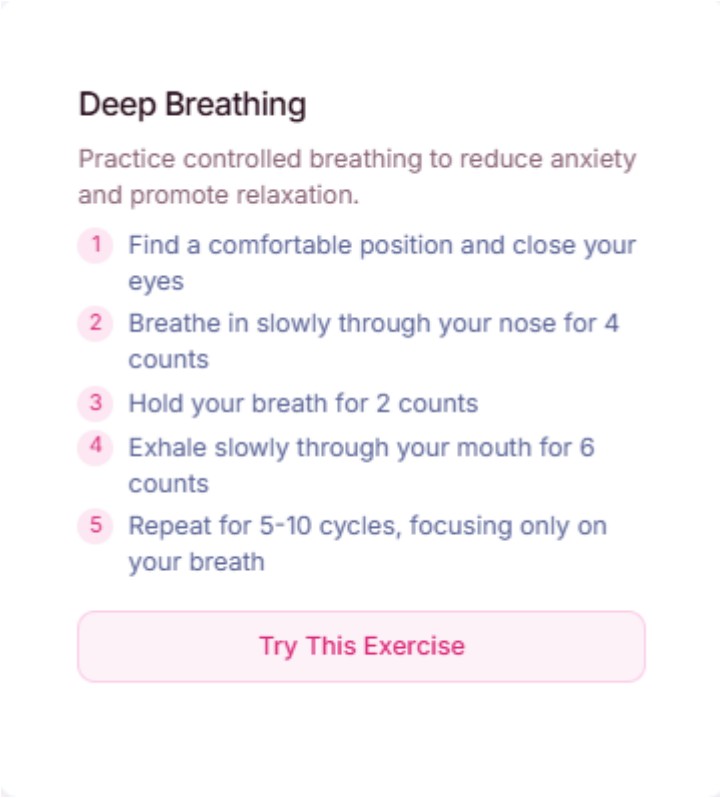

Рисунок 4.9 — Рекомендація для покращення емоційного стану

З метою визначення швидкодії використання застосунку, швидкості відповіді застосунку, його доступності, ефективності роботи та оптимізації пошукових систем, було використано застосунок Lighthouse.

Lighthouse [26] - це інструмент розробника, створений Google для аналізу якості веб-сайтів. Він дозволяє оцінювати продуктивність, доступність, SEO-оптимізацію та відповідність найкращим практикам веб-розробки. Lighthouse можна використовувати як розширення Chrome, через інструменти розробника браузера, в командному рядку або як Node модуль. Коли запускається аудит Lighthouse, він виконує серію тестів на вказаній сторінці і генерує звіт з оцінками та рекомендаціями щодо покращення.

Аналіз продуктивності застосунку для мобільних пристроїв у Lighthouse наведений на рисунку 4.10.

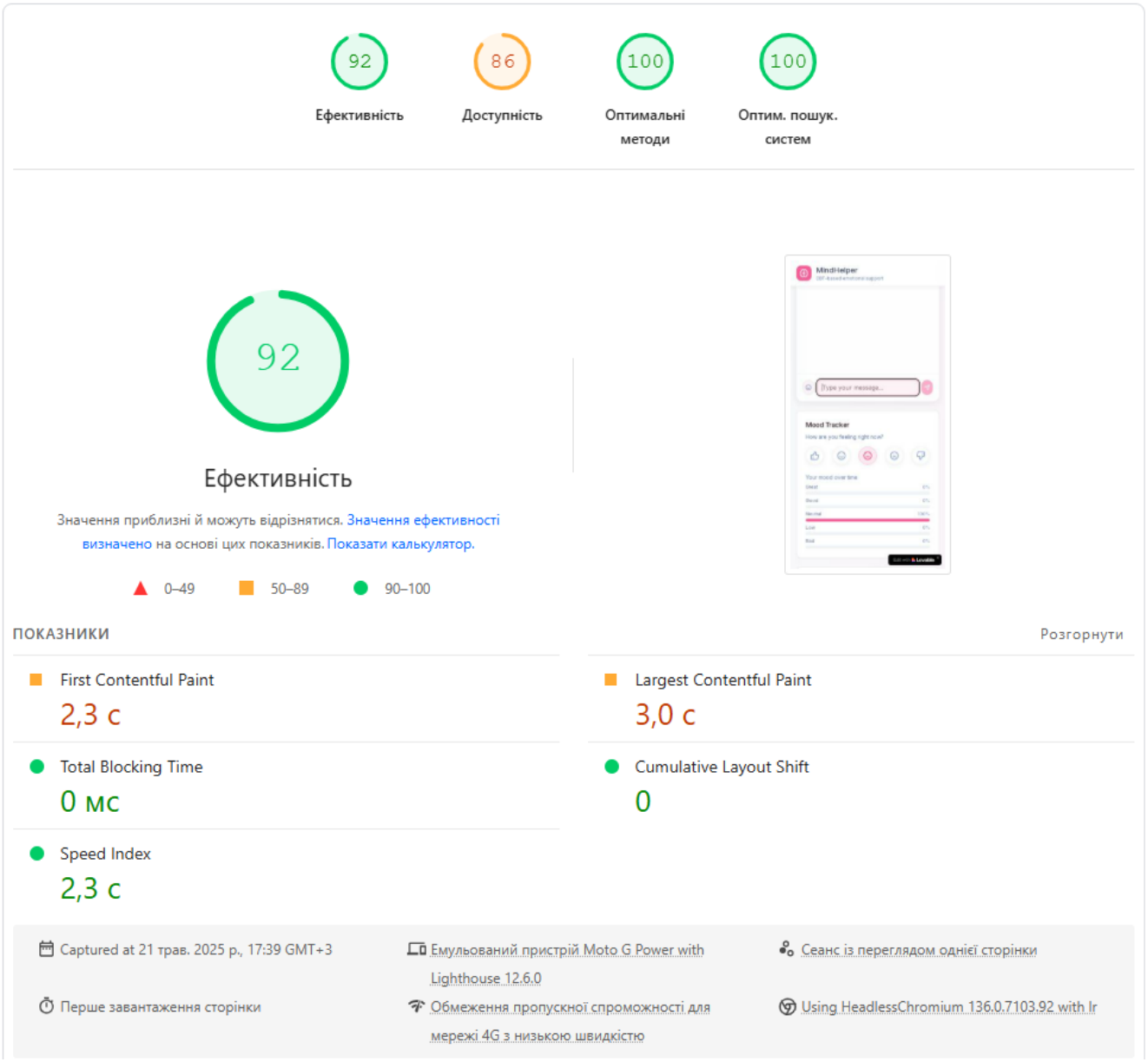

Рисунок 4.10 — Аналіз продуктивності для мобільних пристроїв

Результати аудиту Lighthouse показують відмінну загальну продуктивність веб-застосунку. Оцінка ефективності складає 92/100, що відповідає за швидке та оптимізоване завантаження сторінки. First Contentful Paint (2,3 с) показує, як швидко з'являється перший контент на сторінці після її завантаження. Largest Contentful Paint (3,0 с) вимірює, коли завантажився найбільший видимий елемент сторінки, що важливо для сприйняття користувачем швидкості завантаження. Total Blocking Time (0 мс) демонструє, що під час завантаження сторінка залишається повністю інтерактивною, без затримок у відповіді на дії користувача. Cumulative Layout Shift (0) свідчить про ідеальну візуальну стабільність — елементи не

зміщуються під час завантаження, що запобігає випадковим кліками та покращує зручність використання.

Застосунок має показник 100/100 в категоріях «Оптимальні методи» та «Оптимізація пошукових систем», що позитивно впливає на SEO-рейтинг та слідування стандартам веб-розробки. Тестування проводилось на емульованому мобільному пристрої з обмеженням швидкості мережі 4G, що підтверджує стабільну роботу застосунку навіть в не найкращих умовах з'єднання.

На рисунку 4.11 наведено результат аналізу для комп’ютерів.

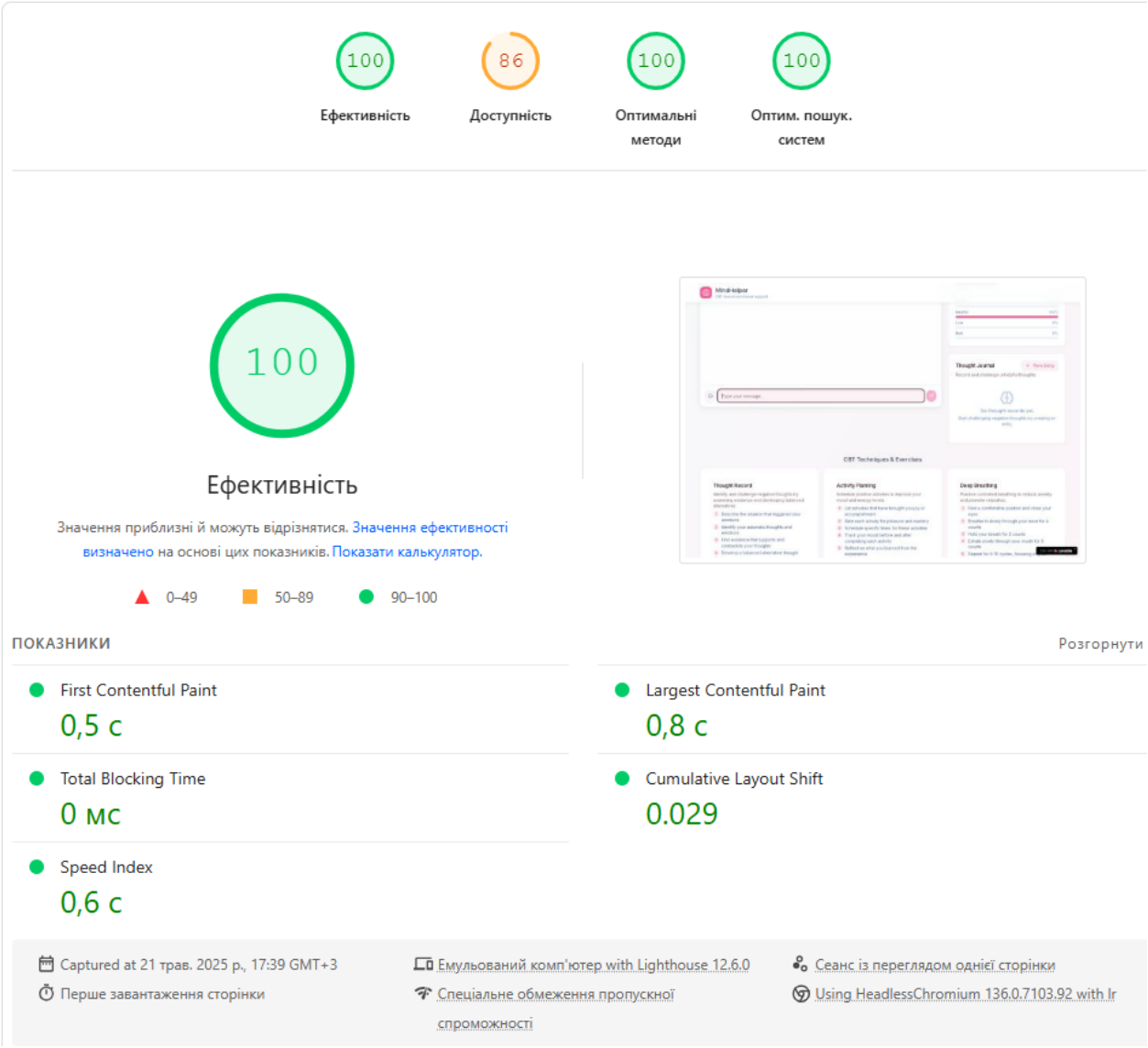

Рисунок 4.10 — Аналіз продуктивності для комп’ютерів

Показники продуктивності для комп’ютерів є аналогічними тому, що і для мобільних пристроїв.

Показник доступності має оцінку 86/100. Він показує, наскільки сайт зручний для людей з обмеженими можливостями. Попри деякі недоліки, що наведені на рисунку 4.11 цей показник все ще вважається високим.

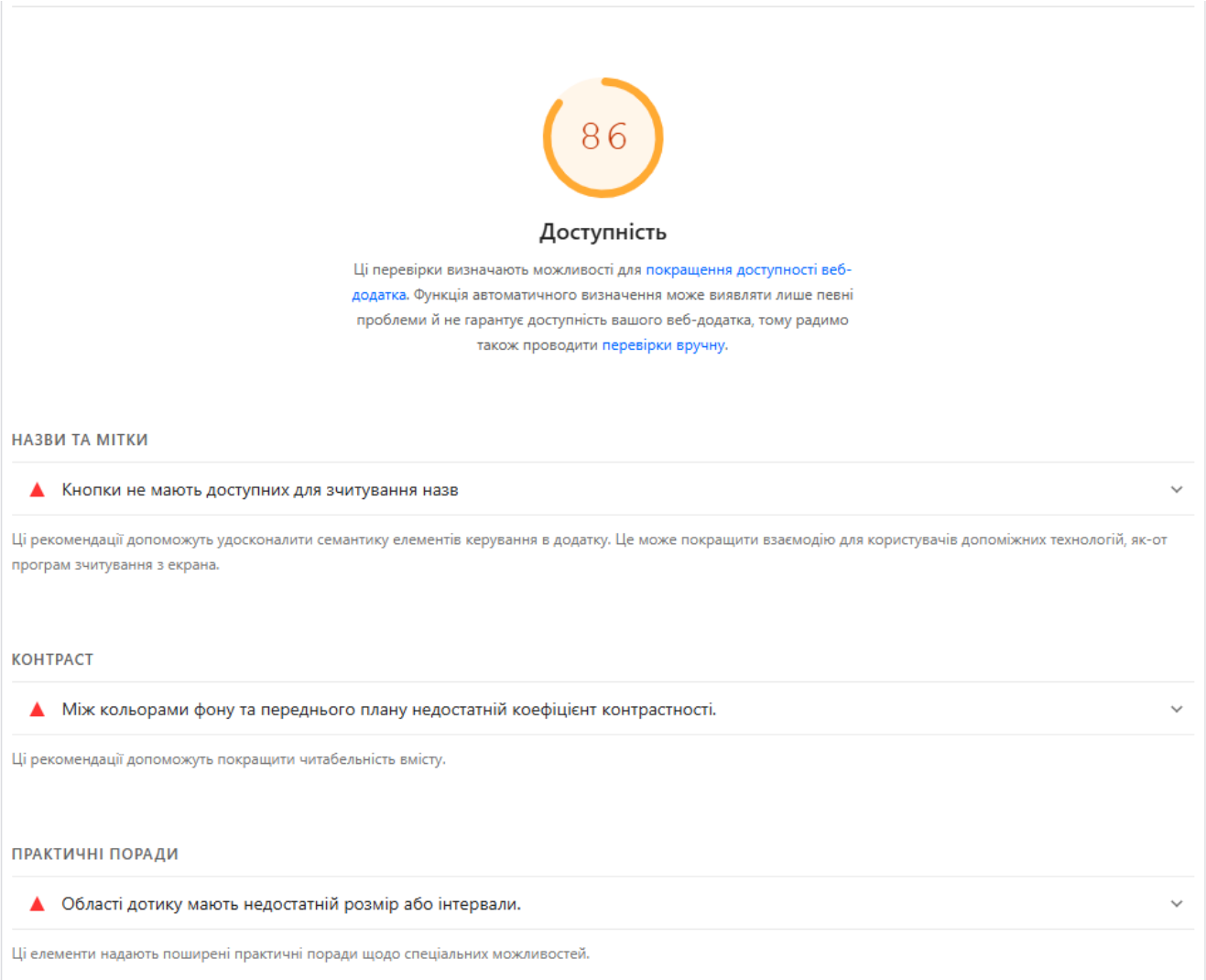

Рисунок 4.11 — Аналіз доступності

Також було проведено тестування часу відповіді чат-бота на повідомлення користувача. На час відповіді впливає розмір повідомлення користувача та розмір відповіді чат-бота. Було проведено 10 тестів, та сформовано графік, який наведено на рисунку 4.12. Час відповіді чат-бота складав від 0.7 до 2.0 секунд, середнє значення становить 1.4 секунди.

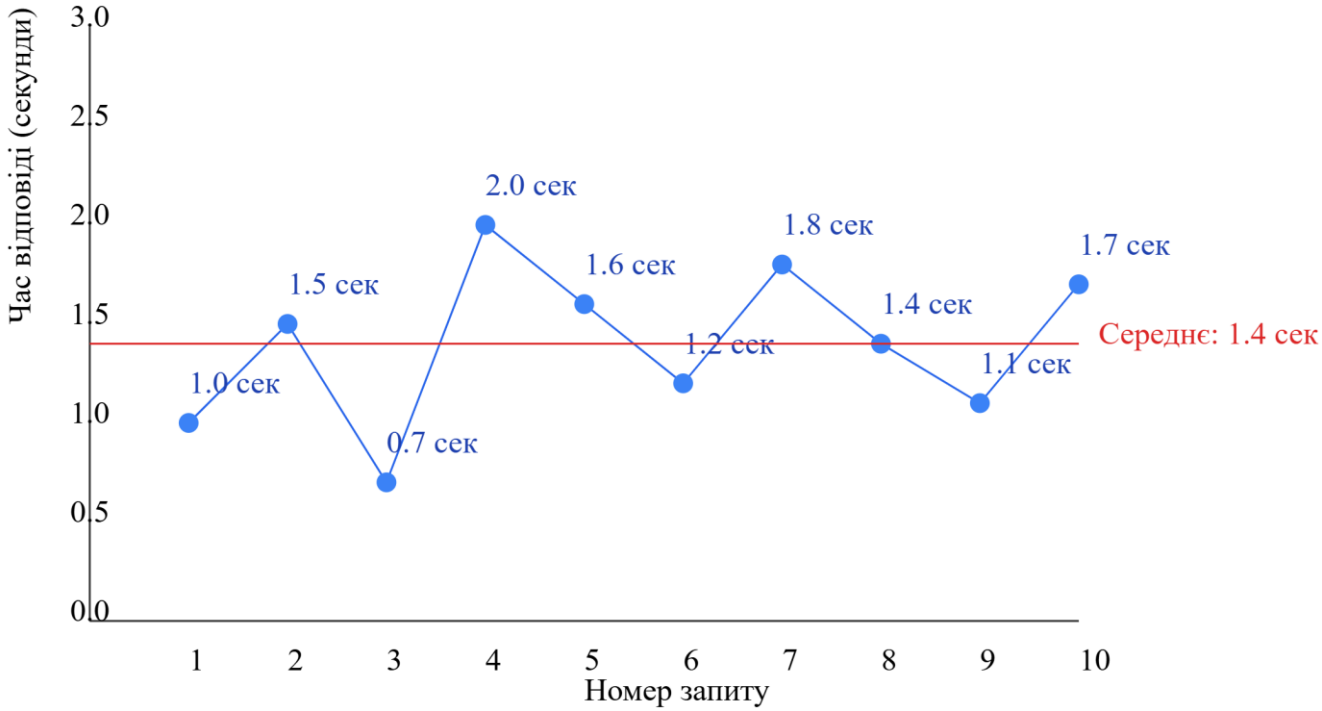

Рисунок 4.12 — Графік часу відповіді

Отже, було проведено тестування системи, продемонстровано її функціонал. Розроблена система виконує поставлені на початку розробки задачі.

4.3 Висновки

У четвертому розділі, було проведено аналіз методів тестування програмної системи діалогової взаємодії для надання психологічної підтримки, обрано метод ручного тестування. Проведено тестування системи, продемонстровано її функціонал. Розроблена система виконує поставлені на початку розробки задачі.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи, було розроблено програмну систему діалогової взаємодії для надання психологічної підтримки. Для розробки було використано мову програмування TypeScript, фреймворки React та середовище розробки VS Code. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок [27].

У першому розділі було проведено аналіз стану питання розробки програмної системи діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії, проведено аналіз методів розв’язання задачі з розробки цієї програмної системи. Проведено порівняльний аналіз аналогів, продемонстровано переваги власної розробки. Проведено постановку задач дослідження.

У другому розділі було розроблено інформаційне забезпечення програмної системи діалогової взаємодії для надання психологічної підтримки, розроблено модель системи, яка наводить технології та модулі, які розроблені для роботи системи, взаємозв’язки між ними. Розроблено алгоритми роботи системи, розроблено метод розширеного журналу думок із автоматичним аналізом та метод динамічного добору вправ за показниками користування.

У третьому розділі було проведено аналіз мов програмування для розробки програмної системи діалогової взаємодії для надання психологічної підтримки, у результаті якого було обрано мову TypeScript. Проведено аналіз середовищ розробки, в результаті якого обрано VS Code. Розроблено інтерфейс системи.

У четвертому розділі, було проведено аналіз методів тестування програмної системи діалогової взаємодії для надання психологічної підтримки, обрано метод ручного тестування. Проведено тестування системи, продемонстровано її функціонал. Розроблена система виконує поставлені на початку розробки задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Philippe TJ, Sikder N, Jackson A, Koblanski ME, Liow E, Pilarinos A, Vasarhelyi K. Digital Health Interventions for Delivery of Mental Health Care: Systematic and Comprehensive Meta-Review. JMIR Ment Health 2022;9(5):e35159

2. Almuqrin A, Hammoud R, Terbagou I, Tognin S, Mechelli A. Smartphone apps for mental health: systematic review of the literature and five recommendations for clinical translation. BMJ Open. 2025 Feb 11;15(2):e093932. doi: 10.1136/bmjopen-2024-093932. PMID: 39933815; PMCID: PMC11815452.

3. Chekhmestruk R., Pasikhov O. DEVELOPMENT OF AN INTELLIGENT DIALOGUE INTERACTION SOFTWARE SYSTEM FOR PROVIDING PSYCHOLOGICAL SUPPORT BASED ON COGNITIVE BEHAVIORAL THERAPY METHODS. Global Trends in the Development of Information Technology and Science: Collection of Scientific Papers "International Scientific Unity" with Proceedings of the 3rd International Scientific and Practical Conference. April 2-4, 2025. Stockholm, Sweden. p.114-117.

4. Lian H, Lu C, Li S, Zhao Y, Tang C, Zong Y. A Survey of Deep Learning-Based Multimodal Emotion Recognition: Speech, Text, and Face. Entropy. 2023; 25(10):1440. https://doi.org/10.3390/e25101440

5. Nikola Kovacevic, Christian Holz, Markus Gross, and Rafael Wampfler. 2024. On Multimodal Emotion Recognition for Human-Chatbot Interaction in the Wild. In Proceedings of the 26th International Conference on Multimodal Interaction (ICMI '24). Association for Computing Machinery, New York, NY, USA, 12–21. https://doi.org/10.1145/3678957.3685759

6. Fitzpatrick KK, Darcy A, Vierhile M. Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial. JMIR Ment Health. 2017 Jun 6;4(2):e19. doi: 10.2196/mental.7785. PMID: 28588005; PMCID: PMC5478797.

7. Ezawa ID, Hollon SD. Cognitive restructuring and psychotherapy outcome: A meta-analytic review. Psychotherapy (Chic). 2023 Sep;60(3):396-406. doi: 10.1037/pst0000474. Epub 2023 Mar 13. PMID: 36913269; PMCID: PMC10440210.

8. CBT Coping Skills and Strategies [Електронний ресурс] – Режим доступу до ресурсу: https://www.verywellmind.com/cognitive-behavioral-coping-strategies-2797612?utm_source=chatgpt.com

9. Bunyi J, Ringland KE, Schueller SM. Accessibility and Digital Mental Health: Considerations for More Accessible and Equitable Mental Health Apps. Front Digit Health. 2021 Sep 29;3:742196. doi: 10.3389/fdgth.2021.742196. PMID: 34713206; PMCID: PMC8521906.

10. Koh J, Tng GYQ, Hartanto A. Potential and Pitfalls of Mobile Mental Health Apps in Traditional Treatment: An Umbrella Review. J Pers Med. 2022 Aug 25;12(9):1376. doi: 10.3390/jpm12091376. PMID: 36143161; PMCID: PMC9505389.

11. Woebot [Електронний ресурс] — Режим доступу до ресурсу: https://woebothealth.com/

12. Wysa [Електронний ресурс] — Режим доступу до ресурсу: https://www.wysa.com/

13. Youper [Електронний ресурс] — Режим доступу до ресурсу: https://www.youper.ai/

14. Методичні вказівки до організації самостійної роботи студентів з дисципліни «Об`єктно-орієнтоване програмування» для студентів спеціальності 121 «Інженерія програмного забезпечення» (денної та заочної форм навчання)/ Уклад. Д.І. Катєльніков,– Вінниця: ВНТУ, 2018. – 52 с.

15. Dan Vanderkam. Effective TypeScript: 83 Specific Ways to Improve Your TypeScript. Farnham: O’Reilly Media, 2024. 401c.

16. Yanko Belov. JavaScript Masterclass: A comprehensive guide to mastering JavaScript programming. Noida: BPB Publications, 2024. 384c.

17. Методичні вказівки до виконання практичних робіт з дисципліни «Основи WEB-дизайну» [Електронний ресурс] / уклад. Р. Ю. Чехместрук. – Вінниця : ВНТУ, 2025. – 42 с.

18. Dilyan Grigorov. Introduction to Python and Large Language Models. Нью-Йорк: Apress, 2024. 380с.

19. VS Code [Електронний ресурс] — Режим доступу до ресурсу: https://code.visualstudio.com/

20. Jetbrains Webstorm [Електронний ресурс] — Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/

21. IntelliJ IDEA Ultimate [Електронний ресурс] — Режим доступу до ресурсу: https://www.jetbrains.com/idea/

22. О.Н.Романюк, Т.О.Савчук. Організація баз даних і знань. Навчальний посібник. — Вінниця: УНІВЕРСУМ-Вінниця, 2003. — 217 с.

23.

24. What is software testing? [Електронний ресурс] — Режим доступу до ресурсу: https://www.ibm.com/think/topics/software-testing

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

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

27. Lighthouse [Електронний ресурс] — Режим доступу до ресурсу: https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk

28. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О.Н. Романюк, Г.О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

65

Додаток А – Технічне завдання

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

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

Технічне завдання
на бакалаврську кваліфікаційну роботу «Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Чехместрук Р.Ю.
«24» 03 2025 року.
Виконав:
студент гр. 5ПІ-21б Пасіхов О.М.
«24» 03 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №18 від 18 лютого 2025 р.

3. Мета та призначення розробки.

Метою роботи є покращення процесу надання психологічної підтримки користувачам шляхом розробки системи діалогової взаємодії для надання психологічної підтримки.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Philippe TJ, Sikder N, Jackson A, Koblanski ME, Liow E, Pilarinos A, Vasarhelyi K. Digital Health Interventions for Delivery of Mental Health Care: Systematic and Comprehensive Meta-Review. JMIR Ment Health 2022;9(5):e35159

2. Dan Vanderkam. Effective TypeScript: 83 Specific Ways to Improve Your TypeScript. Farnham: O’Reilly Media, 2024. 401c.

3. VS Code [Електронний ресурс] — Режим доступу до ресурсу: https://code.visualstudio.com/

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

5. Технічні вимоги

Вихідні дані до роботи: середовище розробки WebStorm, мова розробки TypeScript, бібліотека React, протокол передачі даних HTTP, системи управління базою даних PostgreSQL, алгоритм кластеризації.

6. Конструктивні вимоги.

Інтерфейс програми поовинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

–пояснювальна записка до бакалаврської кваліфікаційної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку систем психологічної підтримки	25.03.25 – 31.03.25
2	Розробка моделі та алгоритмів програмного застосунку	01.04.25 – 07.04.25
3	Варіантний аналіз та вибір мови програмування розробки	08.04.25 – 10.04.25
4	Розробка програмної системи для психологічної підтримки	11.04.25 – 24.04.25
5	Тестування програмного забезпечення	25.04.25 – 05.05.25
6	Оформлення матеріалів до захисту БКР	06.05.25 – 06.06.25

10. Порядок контролю та прийняття.

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

68

Додаток Б – Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Інтелектуальна програмна система діалогової взаємодії для надання психологічної підтримки на основі методів когнітивно-поведінкової терапії

Тип роботи: бакалаврська кваліфікаційна робота

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,3 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

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

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

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

Експертна комісія:

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

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

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Пасіхов О.М.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програми

```
import { Toaster } from "@/components/ui/toaster";
import { Toaster as Sonner } from "@/components/ui/sonner";
import { TooltipProvider } from "@/components/ui/tooltip";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Index from "./pages/Index";
import NotFound from "./pages/NotFound";

const queryClient = new QueryClient();

const App = () => (
  <QueryClientProvider client={queryClient}>
    <TooltipProvider>
      <Toaster />
      <Sonner />
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Index />} />
          {/* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "*" ROUTE */}
          <Route path="*" element={<NotFound />} />
        </Routes>
      </BrowserRouter>
    </TooltipProvider>
  </QueryClientProvider>
);

export default App;
```

```
import { createRoot } from 'react-dom/client'
import App from './App.tsx'
import './index.css'

createRoot(document.getElementById("root")!).render(<App />);
// Possible user message categories
type MessageCategory =
  | 'greeting'
```

```
  | 'mood_positive'
  | 'mood_negative'
  | 'anxiety'
  | 'depression'
  | 'stress'
  | 'sleep'
  | 'general_question'
  | 'gratitude'
  | 'thought_distortion'
  | 'suicidal'
  | 'abuse'
  | 'substance'
  | 'unknown';

// CBT techniques
export const cbtTechniques = {
  cognitiveRestructuring: {
    name: 'Cognitive Restructuring',
    description: 'Identifying and challenging irrational or negative thoughts and replacing them with
balanced, rational thoughts.',
    steps: [
      'Identify the negative thought.',
      'Examine the evidence for and against this thought.',
      'Consider alternative perspectives.',
      'Generate a balanced thought that takes all evidence into account.'
    ],
    example: "Instead of thinking \"I always fail at everything,\" consider \"I have had some setbacks,
but I have also had successes in the past.\""
  },

  behavioralActivation: {
    name: 'Behavioral Activation',
    description: 'Engaging in positive and rewarding activities to improve mood and energy levels.',
    steps: [
      'Make a list of activities that have brought you joy or a sense of accomplishment in the past.',
      'Schedule these activities into your daily routine.',
      'Start with small, manageable activities.',
      'Notice and acknowledge how you feel before and after the activity.'
    ],
    example: 'Taking a short walk, calling a friend, or completing a small creative project.'
  },

  exposureTechnique: {
    name: 'Gradual Exposure',
    description: 'Gradually facing fears and anxiety-provoking situations to reduce avoidance
behaviors.',
    steps: [
      'Create a hierarchy of feared situations from least to most anxiety-provoking.',
      'Start with the least anxiety-provoking situation.',
      'Practice relaxation techniques to manage anxiety.',
      'Gradually work your way up the hierarchy as your anxiety decreases.'
    ],
```

```
    example: 'If you have social anxiety, start by saying hello to one person, then gradually work up to participating in group conversations.'
  },

  mindfulness: {
    name: 'Mindfulness Practice',
    description: 'Focusing attention on the present moment without judgment.',
    steps: [
      'Find a quiet space where you won\'t be disturbed.',
      'Focus on your breath or bodily sensations.',
      'When your mind wanders, gently bring your attention back to the present.',
      'Practice observing thoughts without judgment or attachment.'
    ],
    example: 'Spending 5 minutes focusing on your breath, or mindfully eating a meal by paying attention to the tastes, textures, and sensations.'
  },

  problemSolving: {
    name: 'Structured Problem-Solving',
    description: 'A systematic approach to addressing specific problems and challenges.',
    steps: [
      'Clearly define the problem.',
      'Brainstorm possible solutions without judging them.',
      'Evaluate the pros and cons of each solution.',
      'Choose and implement the most appropriate solution.',
      'Review the outcome and adjust as needed.'
    ],
    example: 'If you\'re overwhelmed at work, identify specific stressors, list possible solutions like delegating tasks, then evaluate which solution is most feasible.'
  }
};

// Helper functions for response generation
const getRandomElement = <T,>(array: T[]): T => {
  return array[Math.floor(Math.random() * array.length)];
};

// Detect potential crisis situations
const detectCrisis = (message: string): boolean => {
  const crisisKeywords = [
    'suicide', 'kill myself', 'end my life', 'don\'t want to live',
    'want to die', 'harm myself', 'hurt myself'
  ];

  return crisisKeywords.some(keyword =>
    message.toLowerCase().includes(keyword)
  );
};

// Message categorization
const categorizeMessage = (message: string): MessageCategory => {
  message = message.toLowerCase();
```

```
  // Check for crisis situations first
  if (detectCrisis(message)) {
    return 'suicidal';
  }

  if (message.match(/^(hi|hello|hey|greetings)/)) {
    return 'greeting';
  }

  if (message.match(/(happy|good|great|excellent|amazing|wonderful|fantastic|glad|positive)/)) {
    return 'mood_positive';
  }

  if (message.match(/(sad|down|unhappy|depressed|awful|terrible|hopeless|bad|miserable|low)/)) {
    return 'mood_negative';
  }

  if (message.match(/(anxiety|anxious|nervous|worry|panic|phobia|scared|frightened|fear)/)) {
    return 'anxiety';
  }

  if (message.match(/(depressed|depression|hopeless|empty|numb|pointless|worthless)/)) {
    return 'depression';
  }

  if (message.match(/(stress|stressed|overwhelmed|pressure|burden|burnout)/)) {
    return 'stress';
  }

  if (message.match(/(sleep|insomnia|tired|fatigue|exhausted|rest|dream|nightmare)/)) {
    return 'sleep';
  }

  if (message.match(/(thank|grateful|appreciate|gratitude)/)) {
    return 'gratitude';
  }

  if
(message.match(/(always|never|everyone|nobody|everything|nothing|should|must|can't|catastrophe)/)) {
    return 'thought_distortion';
  }

  if (message.match(/(abuse|hurt|hit|violence|threat|threaten|unsafe)/)) {
    return 'abuse';
  }

  if (message.match(/(alcohol|drug|substance|addiction|hooked|withdrawal)/)) {
    return 'substance';
  }

  if (message.includes('?')) {
```

```
    return 'general_question';
  }

  return 'unknown';
};

// Crisis resources
const crisisResources = [
  "If you're having thoughts of suicide, please reach out for immediate help:",
  "• National Suicide Prevention Lifeline: 988 or 1-800-273-8255",
  "• Crisis Text Line: Text HOME to 741741",
  "• 911 or go to your nearest emergency room",
  "Remember, you don't have to face these feelings alone. Professional help is available, and they can provide the support you need right now."
];

// Response templates
const responseTemplates = {
  greeting: [
    "Hello! It's nice to connect with you. How are you feeling today?",
    "Hi there! I'm here to support you. How has your day been going?",
    "Hello! I'm your supportive companion. What brings you here today?"
  ],

  mood_positive: [
    "I'm glad to hear you're feeling positive! What's contributed to your good mood?",
    "That's wonderful to hear. Acknowledging positive emotions is just as important as addressing challenges. What's been going well for you?",
    "It's great that you're feeling good. Would you like to explore ways to maintain this positive state?"
  ],

  mood_negative: [
    "I'm sorry to hear you're feeling down. Would you like to talk about what might be contributing to these feelings?",
    "It sounds like you're going through a difficult time. Sometimes identifying specific thoughts behind these feelings can help us address them. Would you like to try that?",
    "When we're feeling low, our thoughts often become more negative. Could you share what's been going through your mind recently?"
  ],

  anxiety: [
    "Anxiety can be really challenging. One technique that might help is deep breathing. Would you like to try a quick breathing exercise together?",
    "When anxiety takes hold, our thoughts often race to worst-case scenarios. Could you share what specifically is causing you worry right now?",
    "Anxiety often involves physical sensations as well as worried thoughts. Are you noticing any physical symptoms like tension or rapid heartbeat?"
  ],

  depression: [
    "Depression can make even small tasks feel overwhelming. Is there one small, manageable activity you might try today?",
```

```
    "When we're depressed, we often lose interest in things we used to enjoy. Have you noticed this happening for you?",
    "Depression often involves negative thoughts about yourself, the world, or the future. Have you noticed any patterns in your thoughts lately?"
  ],

  stress: [
    "Stress can build up without us realizing it. What are some signs that tell you you're feeling stressed?",
    "Managing stress often starts with identifying its sources. What situations or responsibilities feel most overwhelming right now?",
    "Taking breaks is essential when dealing with stress. Is there a small way you could build in a moment of calm today?"
  ],

  sleep: [
    "Sleep troubles can significantly impact our mental well-being. Have you noticed any patterns with your sleep difficulties?",
    "Creating a consistent sleep routine can help. What does your current bedtime routine look like?",
    "Sometimes racing thoughts keep us awake. Would you like to explore some relaxation techniques that might help with falling asleep?"
  ],

  general_question: [
    "That's a thoughtful question. While I can offer CBT-based perspectives, it's important to remember that I'm designed to support, not replace professional guidance.",
    "I appreciate your curiosity. I can share some CBT perspectives on this, though remember that everyone's experience is unique.",
    "Great question. I can offer some thoughts based on CBT principles, which focus on how our thoughts, feelings, and behaviors connect."
  ],

  gratitude: [
    "You're very welcome. I'm here to support you whenever you need it.",
    "I'm glad our conversation has been helpful. Expressing gratitude is actually a positive practice for mental well-being.",
    "You're welcome. Recognizing moments of appreciation, even small ones, can be a powerful tool for shifting perspective."
  ],

  thought_distortion: [
    "I noticed some language that might reflect all-or-nothing thinking. In CBT, we try to find the middle ground between extremes. How might a more balanced perspective look?",
    "Words like 'always' and 'never' can trap us in rigid thinking patterns. Could there be exceptions to this situation?",
    "That sounds like it might involve some 'should' statements, which can create unrealistic expectations. What would happen if you replaced 'should' with 'could' or 'would like to'?"
  ],

  suicidal: [crisisResources.join('\n')],

  abuse: [
```

```
    "I'm concerned about what you've shared. Your safety is important. If you're in immediate danger, please call emergency services at 911.",
    "Thank you for trusting me with this information. No one deserves to experience abuse. The National Domestic Violence Hotline (1-800-799-7233) has trained advocates available 24/7.",
    "What you're describing sounds very difficult. It's important that you know there are resources available to help. Would you like me to share some options for support?"
  ],

  substance: [
    "Struggles with substances can be complex. The SAMHSA National Helpline (1-800-662-4357) offers free, confidential support for individuals and families facing substance use disorders.",
    "Thank you for sharing this. Many people find that talking to a professional specializing in substance use can provide valuable support. Would you like information about resources?",
    "That sounds challenging. Recovery often involves both addressing the substance use itself and the underlying feelings or situations connected to it. Have you been able to identify any patterns or triggers?"
  ],

  unknown: [
    "Thank you for sharing. Could you tell me more about how this has been affecting your thoughts and feelings?",
    "I appreciate you opening up. How have these experiences been impacting your day-to-day life?",
    "I'm here to support you. Would it help to explore some coping strategies related to what you're experiencing?"
  ]
};

// Main function to get response for a user message
export const getResponseForMessage = async (message: string): Promise<string> => {
  // Categorize the message
  const category = categorizeMessage(message);

  // Get appropriate response templates for the category
  const templates = responseTemplates[category];

  // Return a random response from the templates
  return getRandomElement(templates);
};

// CBT exercises
export const cbtExercises = [
  {
    id: 'thought-record',
    title: 'Thought Record',
    description: 'Identify and challenge negative thoughts by examining evidence and developing balanced alternatives.',
    steps: [
      'Describe the situation that triggered your emotions',
      'Identify your automatic thoughts and emotions',
      'Find evidence that supports and contradicts your thoughts',
      'Develop a balanced alternative thought',
      'Rate how you feel after considering the balanced thought'
```

```
    ]
  },
  {
    id: 'behavioral-activation',
    title: 'Activity Planning',
    description: 'Schedule positive activities to improve your mood and energy levels.',
    steps: [
      'List activities that have brought you joy or accomplishment',
      'Rate each activity for pleasure and mastery',
      'Schedule specific times for these activities',
      'Track your mood before and after completing each activity',
      'Reflect on what you learned from the experience'
    ]
  },
  {
    id: 'breathing-exercise',
    title: 'Deep Breathing',
    description: 'Practice controlled breathing to reduce anxiety and promote relaxation.',
    steps: [
      'Find a comfortable position and close your eyes',
      'Breathe in slowly through your nose for 4 counts',
      'Hold your breath for 2 counts',
      'Exhale slowly through your mouth for 6 counts',
      'Repeat for 5-10 cycles, focusing only on your breath'
    ]
  },
  {
    id: 'gratitude-practice',
    title: 'Gratitude Journal',
    description: "Cultivate positive emotions by regularly acknowledging things you're grateful for.",
    steps: [
      'Each day, write down 3 things you\'re grateful for',
      'Include why each thing matters to you',
      'Try to find new things each day, even small details',
      'Notice how focusing on gratitude affects your mood',
      'Review your entries weekly to recognize positive patterns'
    ]
  },
  {
    id: 'progressive-relaxation',
    title: 'Progressive Muscle Relaxation',
    description: 'Reduce physical tension by systematically tensing and relaxing muscle groups.',
    steps: [
      'Find a quiet space where you won\'t be disturbed',
      'Starting with your feet, tense the muscles for 5-10 seconds',
      'Release the tension and notice the feeling of relaxation',
      'Progress upward through each muscle group in your body',
      'End with a few moments of mindful awareness of your relaxed state'
    ]
  }
];
```

```
import React from 'react';
import { ChatProvider } from '@/context/ChatContext';
import Header from '@/components/Header';
import ChatInterface from '@/components/ChatInterface';
import MoodTracker from '@/components/MoodTracker';
import ThoughtJournal from '@/components/ThoughtJournal';
import { cbtExercises } from '@/utils/cbtTechniques';
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { HeartPulse, Brain, Info } from 'lucide-react';

const Index = () => {
  return (
    <ChatProvider>
      <div className="min-h-screen bg-gradient-to-br from-harmony-50 to-mind-50">
        <Header />

        <main className="pt-20 pb-10 px-4 max-w-screen-xl mx-auto">
          <div className="text-center mb-8 animate-fade-up">
            <div className="relative inline-block">
              <span className="inline-block px-3 py-1 bg-mind-100 text-mind-600 rounded-full text-xs
font-medium mb-2 animate-pulse-slow">
                Your Empathetic CBT Companion
              </span>
            </div>
            <h2 className="text-3xl font-medium text-harmony-900 mb-2">Welcome to
MindHelper</h2>
            <p className="text-harmony-600 max-w-xl mx-auto">
              A supportive space to process your thoughts, track your moods, and practice cognitive
behavioral techniques.
            </p>
          </div>

          <div className="grid grid-cols-1 lg:grid-cols-3 gap-6 animate-fade-up" style={{
animationDelay: '100ms' }}>
            <div className="lg:col-span-2 h-[600px]">
              <ChatInterface />
            </div>

            <div className="space-y-6">
              <MoodTracker />
              <ThoughtJournal />
            </div>
          </div>

          <div className="mt-10 animate-fade-up" style={{ animationDelay: '200ms' }}>
            <h3 className="text-xl font-medium text-harmony-900 mb-4 text-center">
              CBT Techniques & Exercises
            </h3>

            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
```

```
          {cbtExercises.map((exercise, index) => (
            <Card key={exercise.id} className="cbt-card overflow-hidden">
              <CardHeader className="pb-2">
                <CardTitle className="text-lg font-medium">
                  {exercise.title}
                </CardTitle>
                <CardDescription>
                  {exercise.description}
                </CardDescription>
              </CardHeader>
              <CardContent>
                <ul className="space-y-1 text-sm">
                  {exercise.steps.map((step, i) => (
                    <li key={i} className="flex items-start gap-2">
                      <span className="inline-flex h-5 w-5 shrink-0 items-center justify-center rounded-
full bg-mind-100 text-mind-600 text-xs">
                        {i + 1}
                      </span>
                      <span className="text-harmony-700">{step}</span>
                    </li>
                  ))}
                </ul>

                <Button
                  variant="outline"
                  className="w-full mt-4 border-mind-200 text-mind-600 hover:bg-mind-50 hover:text-
mind-700"
                >
                  Try This Exercise
                </Button>
              </CardContent>
            </Card>
          ))}
        </div>
      </div>

      <div className="mt-12 text-center text-harmony-600 text-sm animate-fade-up" style={{
animationDelay: '300ms' }}>
        <p className="flex items-center justify-center gap-1">
          <Info className="h-4 w-4" />
          <span>This is a CBT-based support tool, not a replacement for professional mental health
care.</span>
        </p>
      </div>
    </main>
  </div>
 </ChatProvider>
);
};

export default Index;
```

```
import { clsx, type ClassValue } from "clsx"
import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}
import * as React from "react"

const MOBILE_BREAKPOINT = 768

export function useIsMobile() {
  const [isMobile, setIsMobile] = React.useState<boolean | undefined>(undefined)

  React.useEffect(() => {
    const mql = window.matchMedia(`(max-width: ${MOBILE_BREAKPOINT - 1}px)`)
    const onChange = () => {
      setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    }
    mql.addEventListener("change", onChange)
    setIsMobile(window.innerWidth < MOBILE_BREAKPOINT)
    return () => mql.removeEventListener("change", onChange)
  }, [])

  return !!isMobile
}
import * as React from "react"

import type {
  ToastActionElement,
  ToastProps,
} from "@/components/ui/toast"

const TOAST_LIMIT = 1
const TOAST_REMOVE_DELAY = 1000000

type ToasterToast = ToastProps & {
  id: string
  title?: React.ReactNode
  description?: React.ReactNode
  action?: ToastActionElement
}

const actionTypes = {
  ADD_TOAST: "ADD_TOAST",
  UPDATE_TOAST: "UPDATE_TOAST",
  DISMISS_TOAST: "DISMISS_TOAST",
  REMOVE_TOAST: "REMOVE_TOAST",
} as const

let count = 0

function genId() {
```

```
  count = (count + 1) % Number.MAX_SAFE_INTEGER
  return count.toString()
}

type ActionType = typeof actionTypes

type Action =
  | {
      type: ActionType["ADD_TOAST"]
      toast: ToasterToast
    }
  | {
      type: ActionType["UPDATE_TOAST"]
      toast: Partial<ToasterToast>
    }
  | {
      type: ActionType["DISMISS_TOAST"]
      toastId?: ToasterToast["id"]
    }
  | {
      type: ActionType["REMOVE_TOAST"]
      toastId?: ToasterToast["id"]
    }

interface State {
  toasts: ToasterToast[]
}

const toastTimeouts = new Map<string, ReturnType<typeof setTimeout>>()

const addToRemoveQueue = (toastId: string) => {
  if (toastTimeouts.has(toastId)) {
    return
  }

  const timeout = setTimeout(() => {
    toastTimeouts.delete(toastId)
    dispatch({
      type: "REMOVE_TOAST",
      toastId: toastId,
    })
  }, TOAST_REMOVE_DELAY)

  toastTimeouts.set(toastId, timeout)
}

export const reducer = (state: State, action: Action): State => {
  switch (action.type) {
    case "ADD_TOAST":
      return {
        ...state,
        toasts: [action.toast, ...state.toasts].slice(0, TOAST_LIMIT),
```

```
      }

    case "UPDATE_TOAST":
      return {
        ...state,
        toasts: state.toasts.map((t) =>
          t.id === action.toast.id ? { ...t, ...action.toast } : t
        ),
      }

    case "DISMISS_TOAST": {
      const { toastId } = action

      // ! Side effects ! - This could be extracted into a dismissToast() action,
      // but I'll keep it here for simplicity
      if (toastId) {
        addToRemoveQueue(toastId)
      } else {
        state.toasts.forEach((toast) => {
          addToRemoveQueue(toast.id)
        })
      }

      return {
        ...state,
        toasts: state.toasts.map((t) =>
          t.id === toastId || toastId === undefined
            ? {
                ...t,
                open: false,
              }
            : t
        ),
      }
    }
    case "REMOVE_TOAST":
      if (action.toastId === undefined) {
        return {
          ...state,
          toasts: [],
        }
      }
      return {
        ...state,
        toasts: state.toasts.filter((t) => t.id !== action.toastId),
      }
  }
}

const listeners: Array<(state: State) => void> = []

let memoryState: State = { toasts: [] }
```

```
function dispatch(action: Action) {
  memoryState = reducer(memoryState, action)
  listeners.forEach((listener) => {
    listener(memoryState)
  })
}

type Toast = Omit<ToasterToast, "id">

function toast({ ...props }: Toast) {
  const id = genId()

  const update = (props: ToasterToast) =>
    dispatch({
      type: "UPDATE_TOAST",
      toast: { ...props, id },
    })
  const dismiss = () => dispatch({ type: "DISMISS_TOAST", toastId: id })

  dispatch({
    type: "ADD_TOAST",
    toast: {
      ...props,
      id,
      open: true,
      onOpenChange: (open) => {
        if (!open) dismiss()
      },
    },
  })

  return {
    id: id,
    dismiss,
    update,
  }
}

function useToast() {
  const [state, setState] = React.useState<State>(memoryState)

  React.useEffect(() => {
    listeners.push(setState)
    return () => {
      const index = listeners.indexOf(setState)
      if (index > -1) {
        listeners.splice(index, 1)
      }
    }
  }, [state])
```

```
  return {
    ...state,
    toast,
    dismiss: (toastId?: string) => dispatch({ type: "DISMISS_TOAST", toastId }),
  }
}

export { useToast, toast }

import { useState, useEffect, useCallback } from 'react';
import { useChatContext } from '@/context/ChatContext';
import { getResponseForMessage } from '@/utils/cbtTechniques';

export const useChatbot = () => {
  const { addMessage, setIsTyping } = useChatContext();
  const [userInput, setUserInput] = useState('');
  const [isInputDisabled, setIsInputDisabled] = useState(false);

  const simulateTyping = useCallback((text: string) => {
    return new Promise<void>((resolve) => {
      setIsTyping(true);

      // Calculate a realistic typing delay (roughly 30-70ms per character, with variance)
      const baseDelay = 1000; // Initial thinking time
      const perCharDelay = Math.min(
        3000, // Cap the total delay at 3 seconds
        baseDelay + text.length * 30 + Math.random() * 500
      );

      setTimeout(() => {
        setIsTyping(false);
        addMessage(text, 'bot');
        resolve();
      }, perCharDelay);
    });
  }, [addMessage, setIsTyping]);

  const handleSendMessage = useCallback(async () => {
    if (!userInput.trim()) return;

    const trimmedInput = userInput.trim();
    setUserInput('');
    setIsInputDisabled(true);

    // Add user message immediately
    addMessage(trimmedInput, 'user');

    try {
      // Get chatbot response
      const response = await getResponseForMessage(trimmedInput);

      // Simulate typing for response
```

```
      await simulateTyping(response);
    } catch (error) {
      console.error('Error getting chatbot response:', error);
      await simulateTyping("I'm sorry, I'm having trouble processing that right now. Could you try again
or phrase it differently?");
    } finally {
      setIsInputDisabled(false);
    }
  }, [userInput, addMessage, simulateTyping]);

  return {
    userInput,
    setUserInput,
    handleSendMessage,
    isInputDisabled
  };
};

import React, { createContext, useContext, useState, useEffect } from 'react';

// Types
export type MessageType = {
  id: string;
  content: string;
  sender: 'user' | 'bot';
  timestamp: Date;
};

export type MoodType = 'great' | 'good' | 'neutral' | 'low' | 'bad';

export type ThoughtType = {
  id: string;
  situation: string;
  automaticThought: string;
  emotion: string;
  intensity: number;
  evidence: string;
  alternativeThought: string;
  newEmotion: string;
  newIntensity: number;
  createdAt: Date;
};

type ChatContextType = {
  messages: MessageType[];
  addMessage: (content: string, sender: 'user' | 'bot') => void;
  isTyping: boolean;
  setIsTyping: (typing: boolean) => void;
  currentMood: MoodType;
  setCurrentMood: (mood: MoodType) => void;
  moodHistory: Array<{ mood: MoodType; timestamp: Date }>;
  thoughts: ThoughtType[];
```

```
  addThought: (thought: Omit<ThoughtType, 'id' | 'createdAt'>) => void;
  clearChat: () => void;
};

// Create context
const ChatContext = createContext<ChatContextType | undefined>(undefined);

// Create the provider component
export const ChatProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const [messages, setMessages] = useState<MessageType[]>([]);
  const [isTyping, setIsTyping] = useState(false);
  const [currentMood, setCurrentMood] = useState<MoodType>('neutral');
  const [moodHistory, setMoodHistory] = useState<Array<{ mood: MoodType; timestamp: Date
}>>([]);
  const [thoughts, setThoughts] = useState<ThoughtType[]>([]);
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

ІНТЕЛЕКТУАЛЬНА ПРОГРАМНА СИСТЕМА ДІАЛОГОВОЇ ВЗАЄМОДІЇ ДЛЯ НАДАННЯ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ НА ОСНОВІ МЕТОДІВ КОГНІТИВНО-ПОВЕДІНКОВОЇ ТЕРАПІЇ

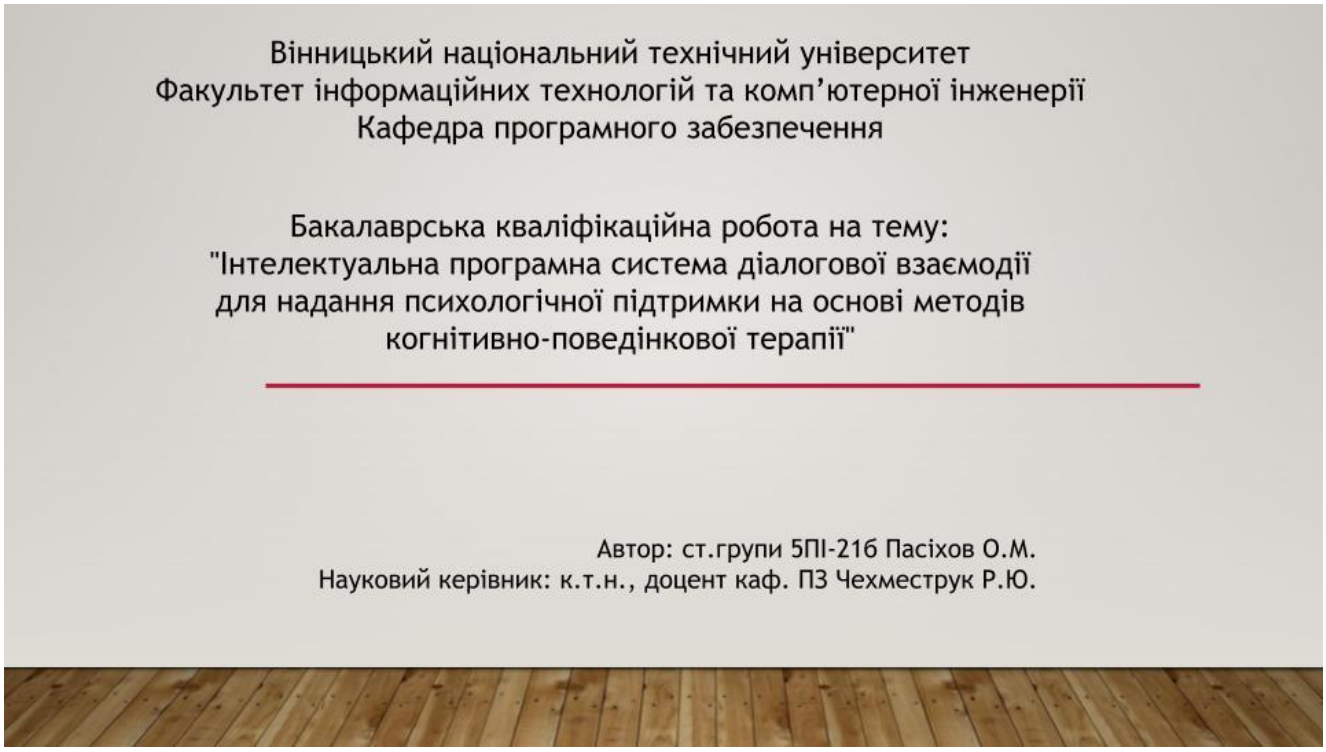

Рисунок Г.1 — Титульний слайд

Рисунок Г.2 — Актуальність теми

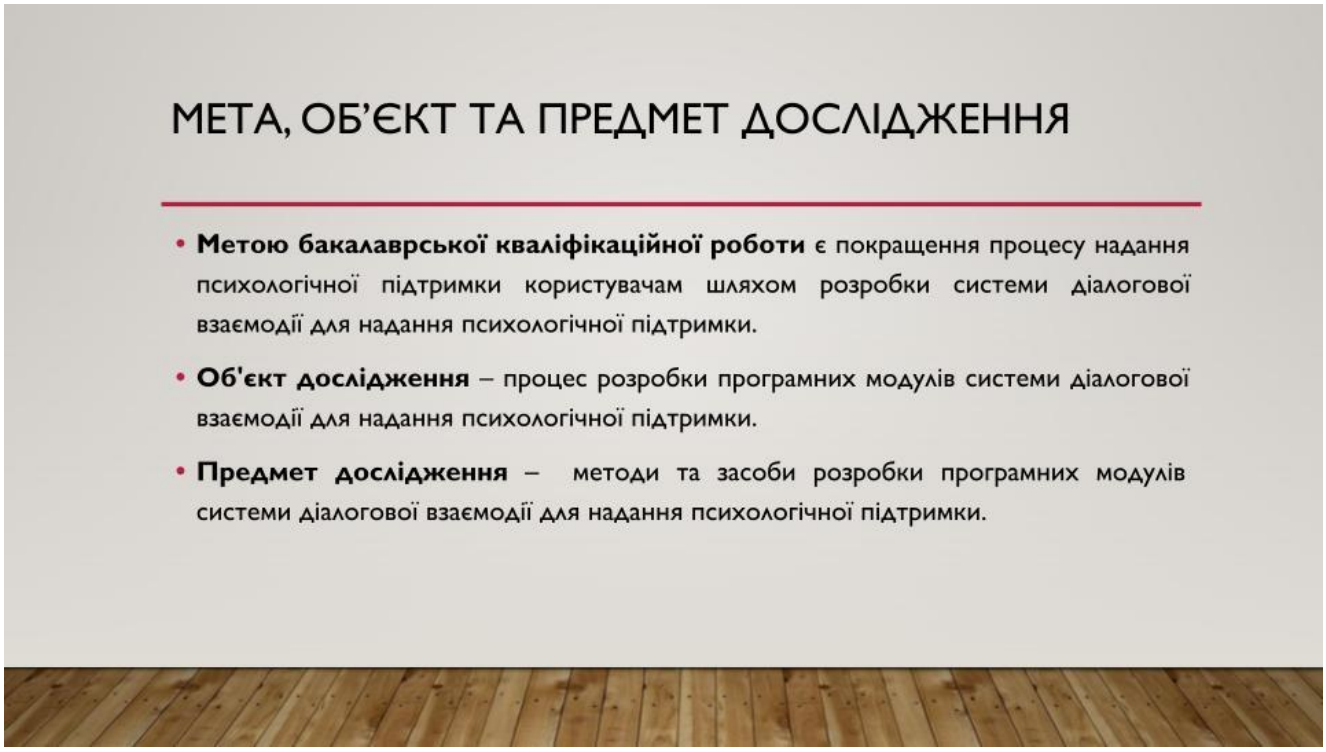

Рисунок Г.3 — Мета, об’єкт та предмет дослідження

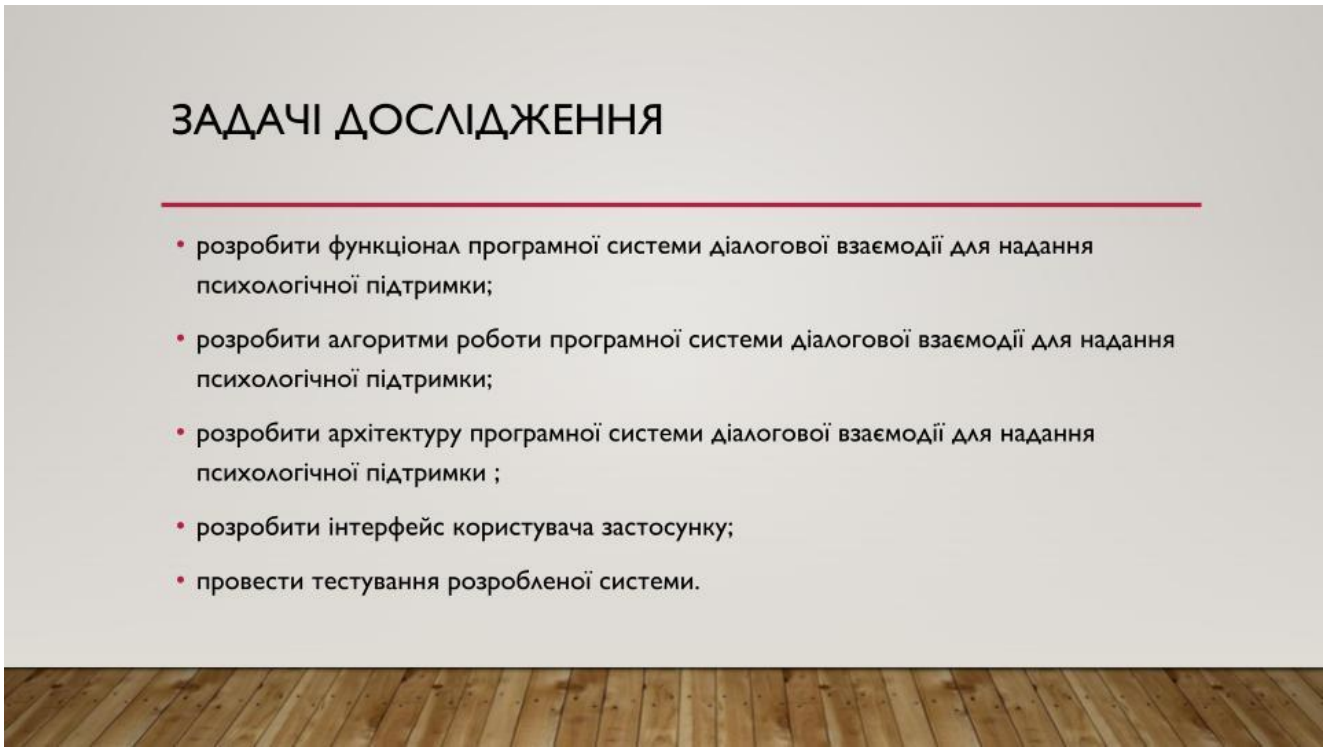

Рисунок Г.4 — Завдання дослідження

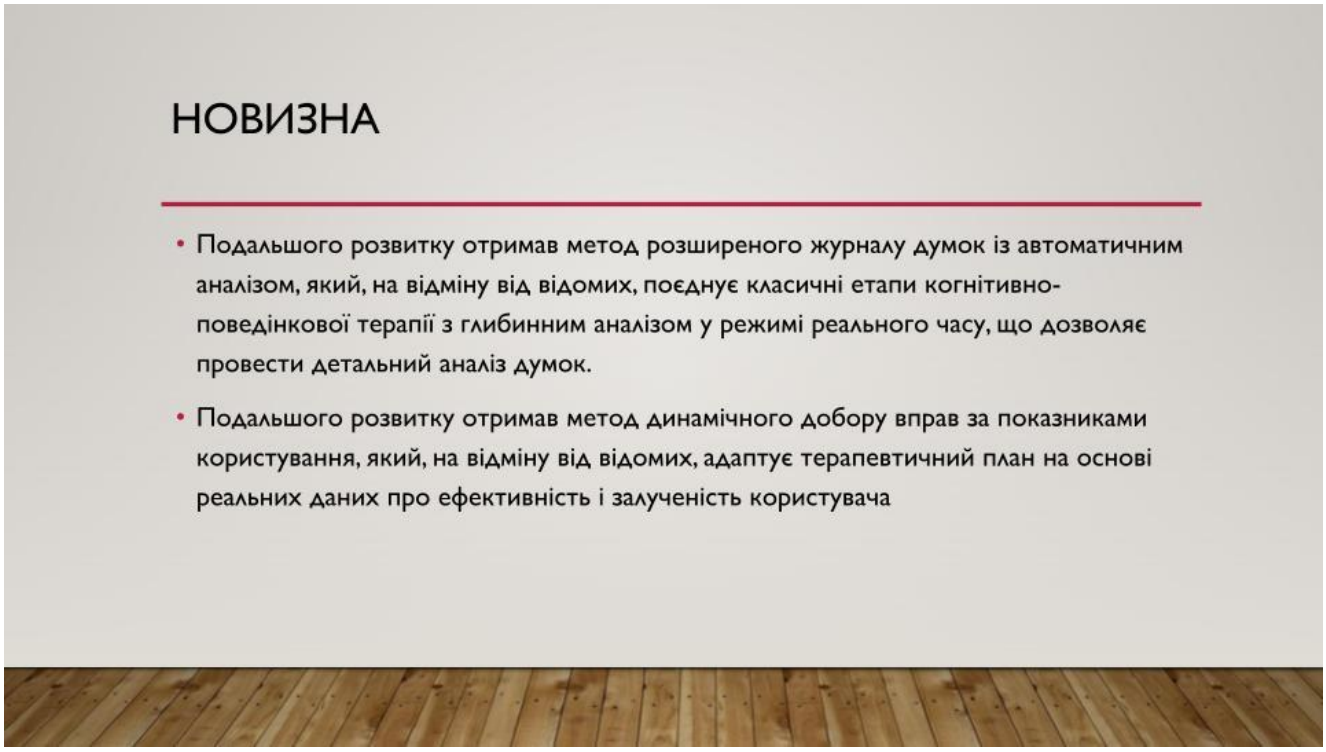

Рисунок Г.5 — Новизна

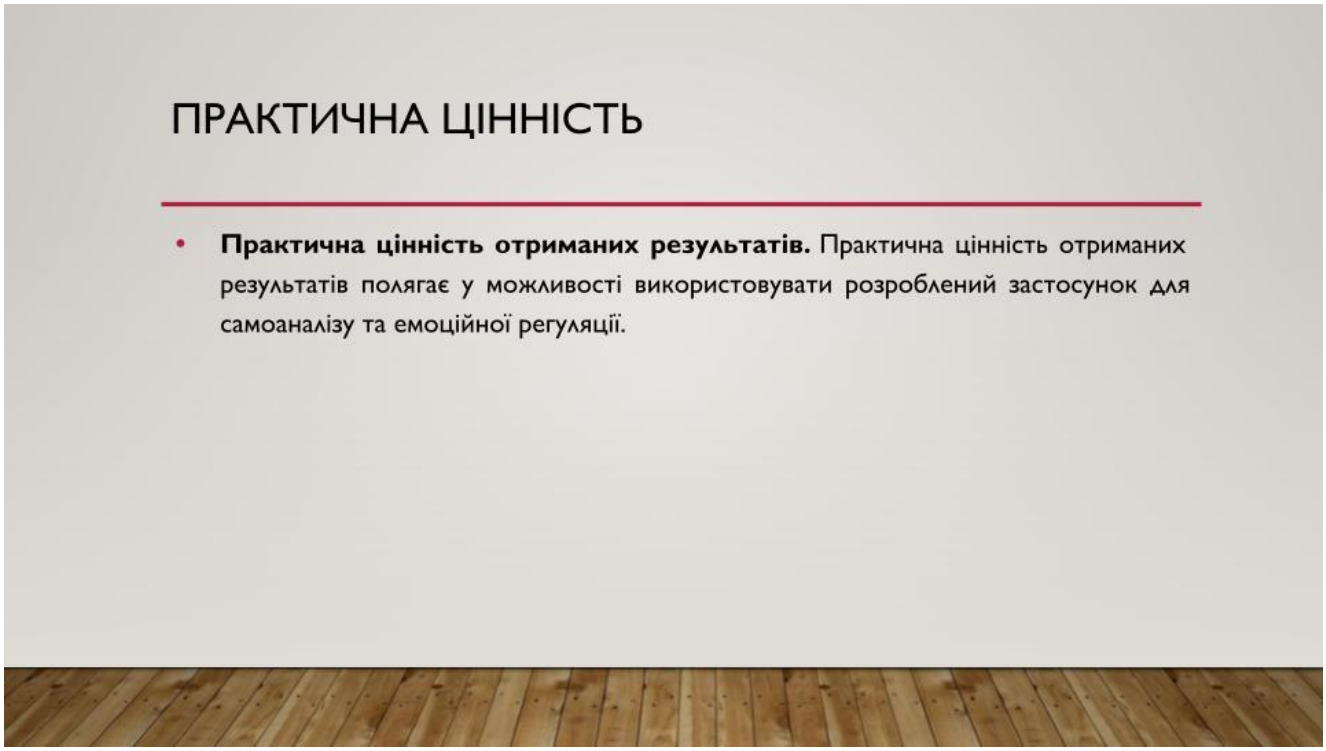

Рисунок Г.6 — Практична цінність отриманих результатів

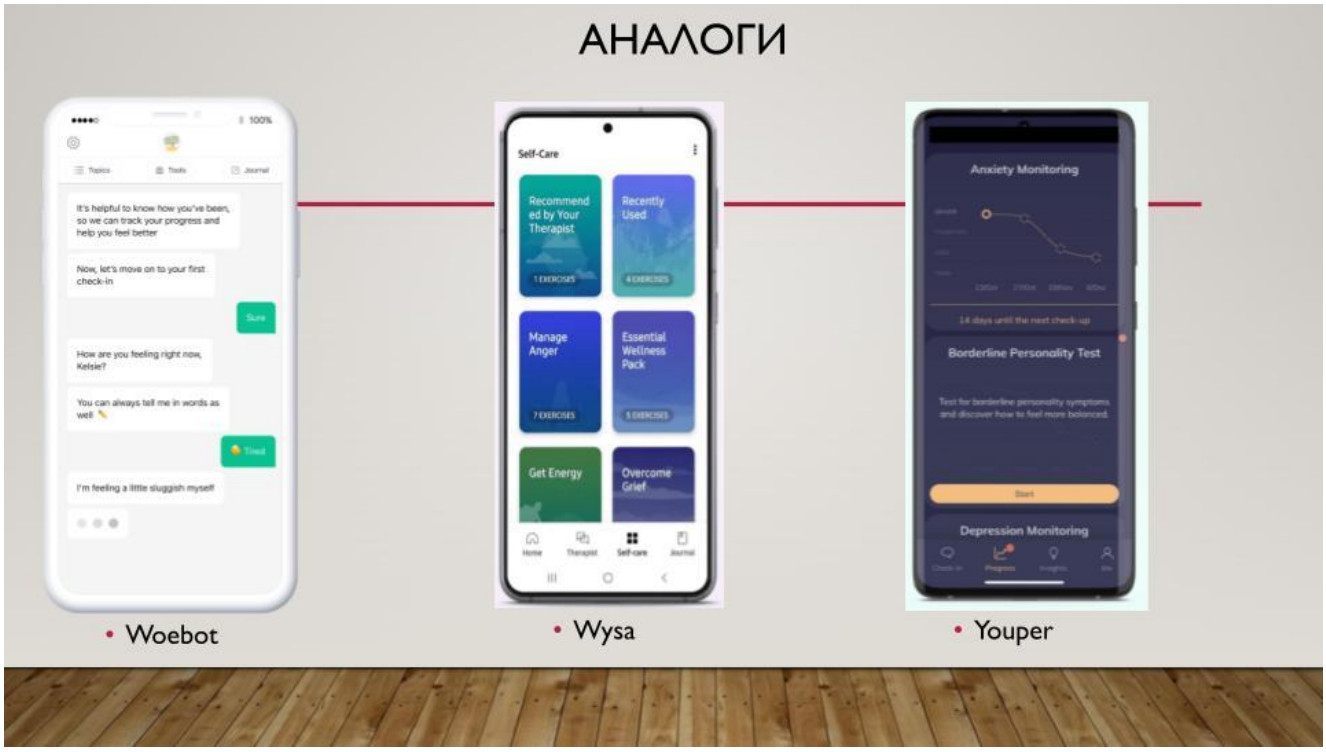

Рисунок Г.7 — Аналоги

Характеристика	Woebot	Wysa	Youper	Власна розробка
Використання когнітивно-поведінкової терапії	+	+	+	+
Штучний інтелект	-	+	+	+
Діалогова взаємодія	+	+	+	+
Анонімність	+	-	+	+
Персоналізація	+	+	-	+
Моніторинг настрою	+	+	+	+
Сумарний бал	5	5	5	6

Рисунок Г.8 — Порівняльний аналіз аналогів

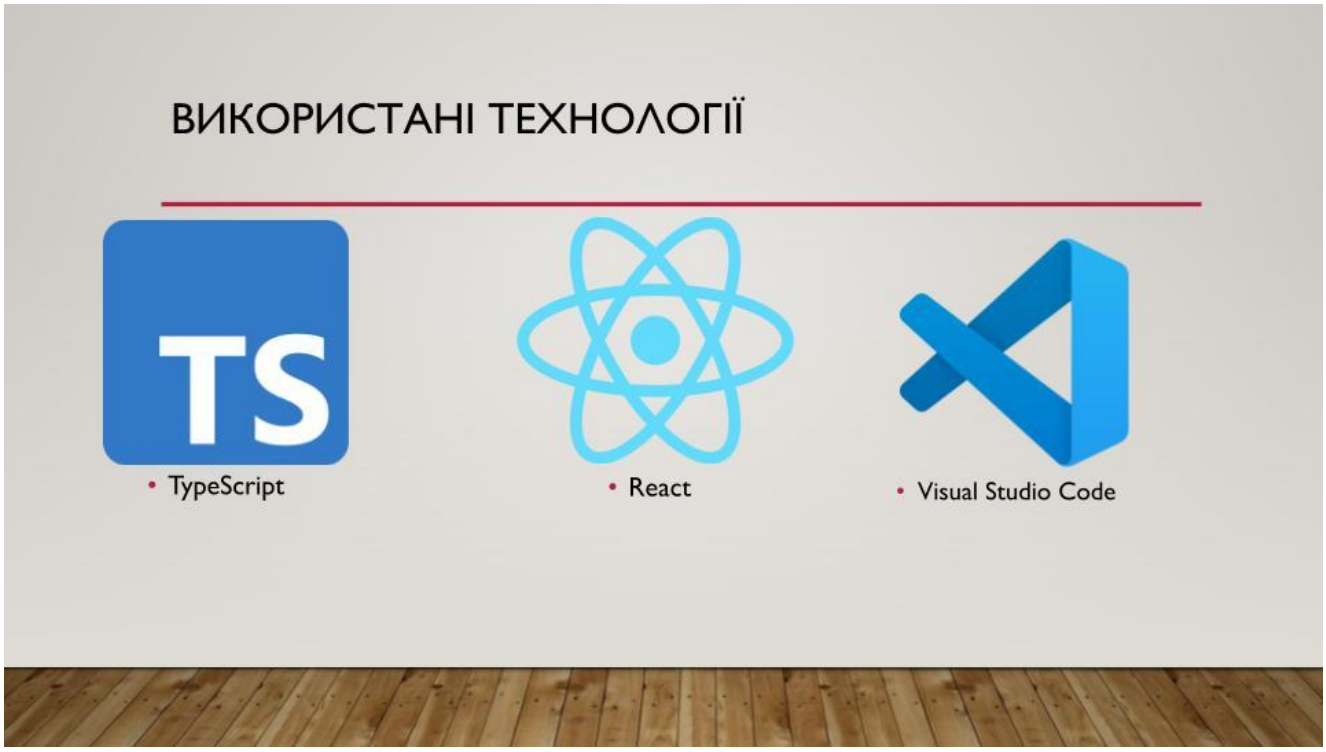

Рисунок Г.9 — Використані технології

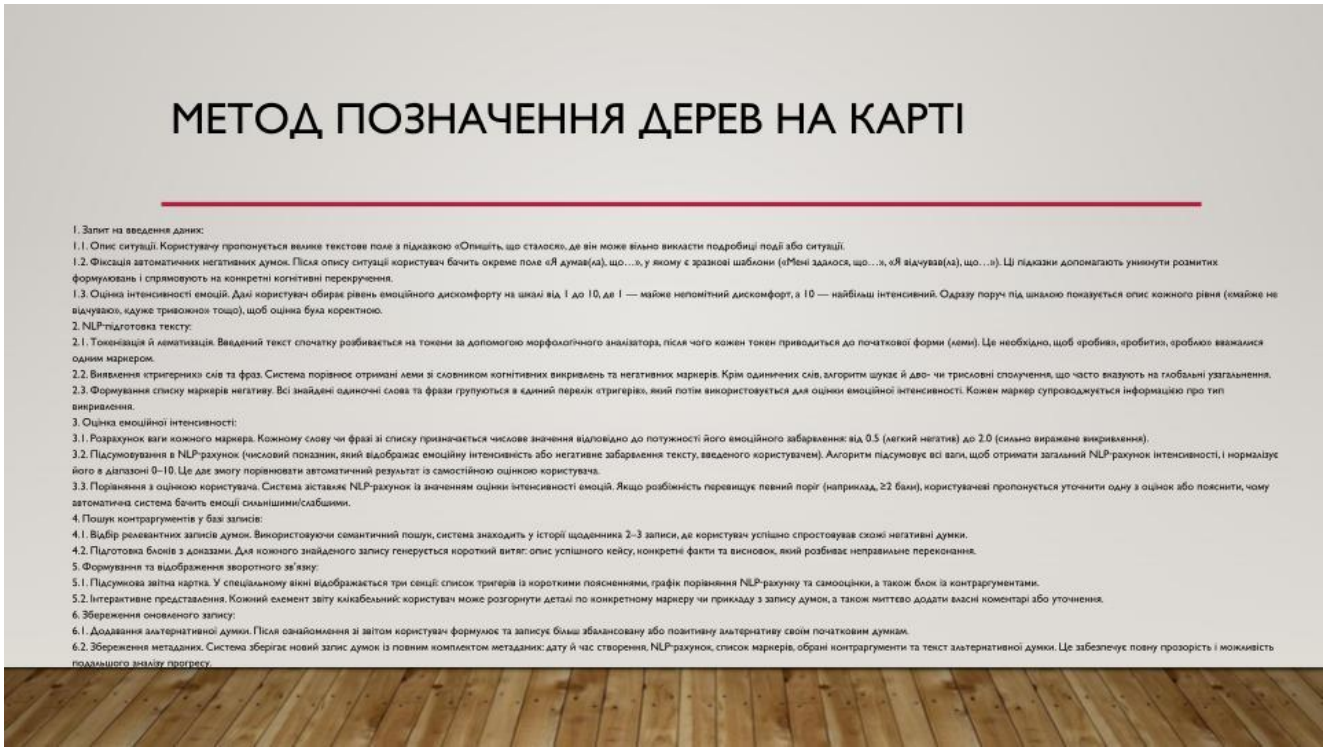

Рисунок Г.10 — Розробка методу позначення дерев на карті

Рисунок Г.11 — Модель системи

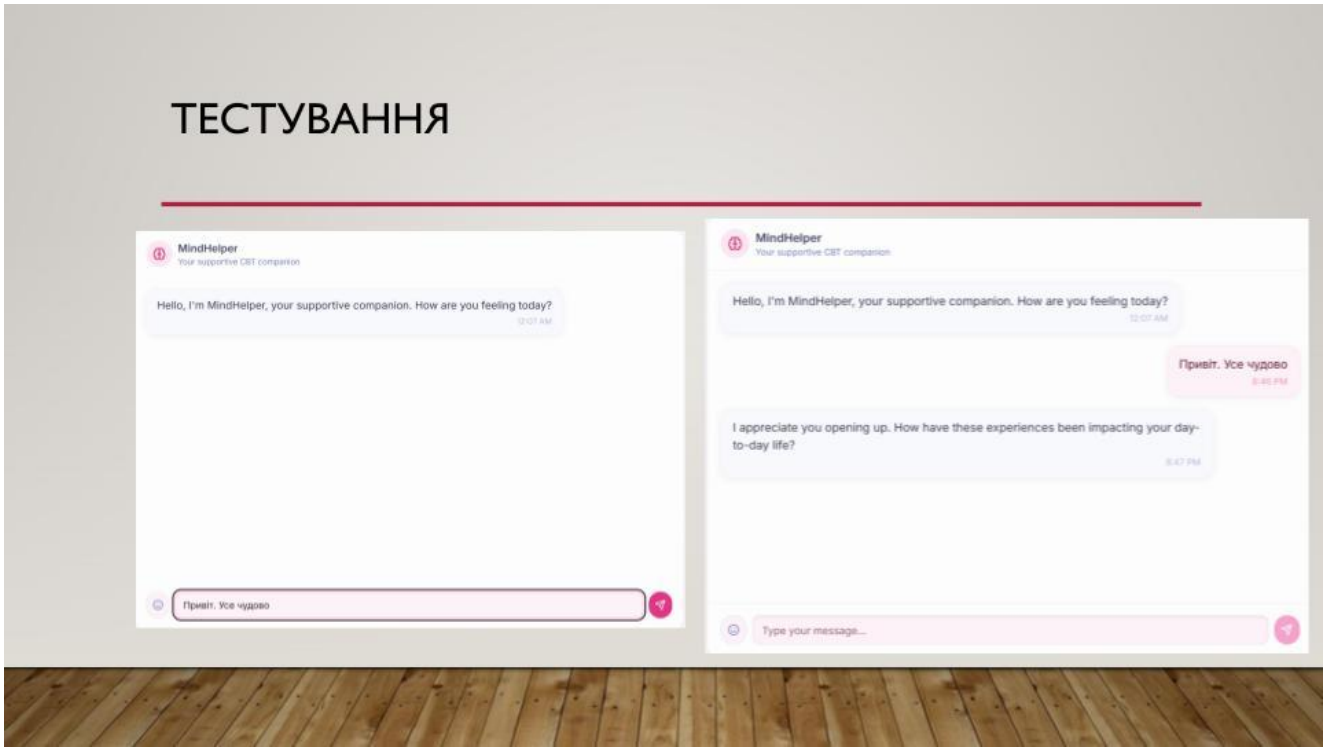

Рисунок Г.12 – Тестування

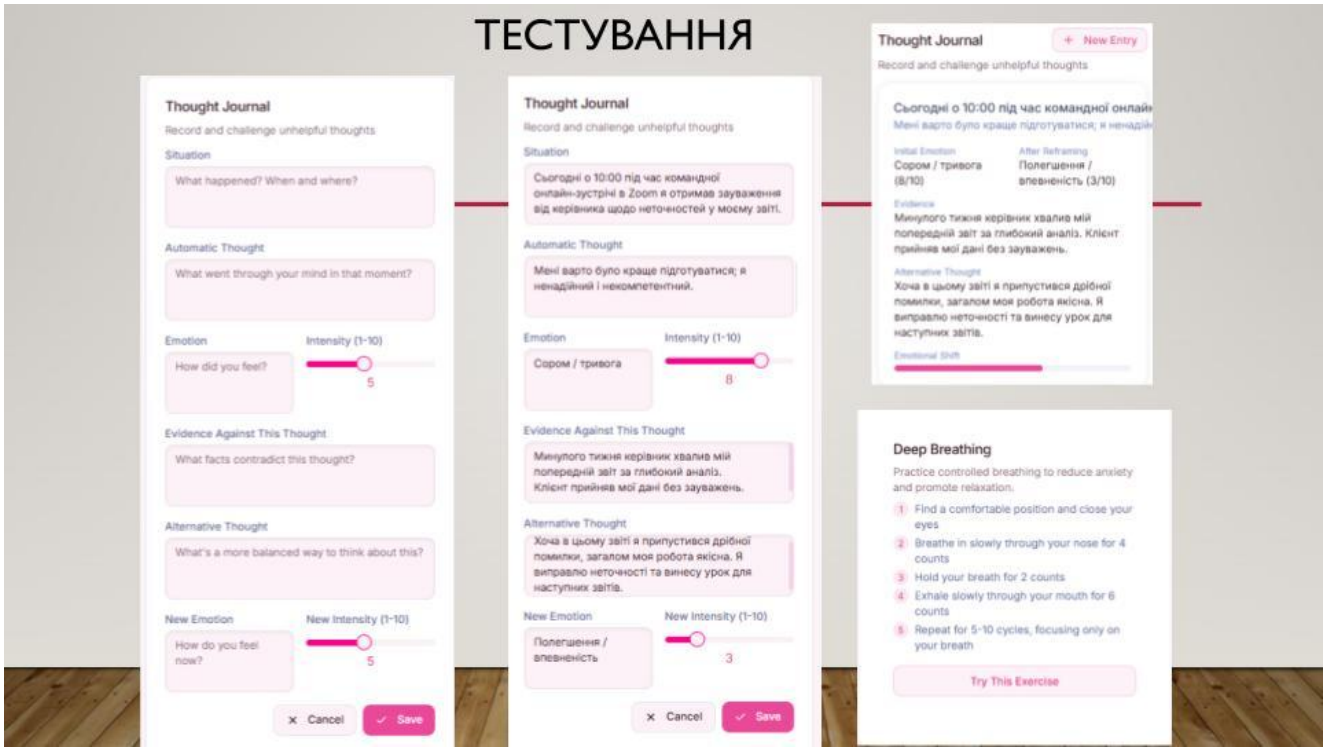

Рисунок Г.13 — Тестування спілкування та налаштувань

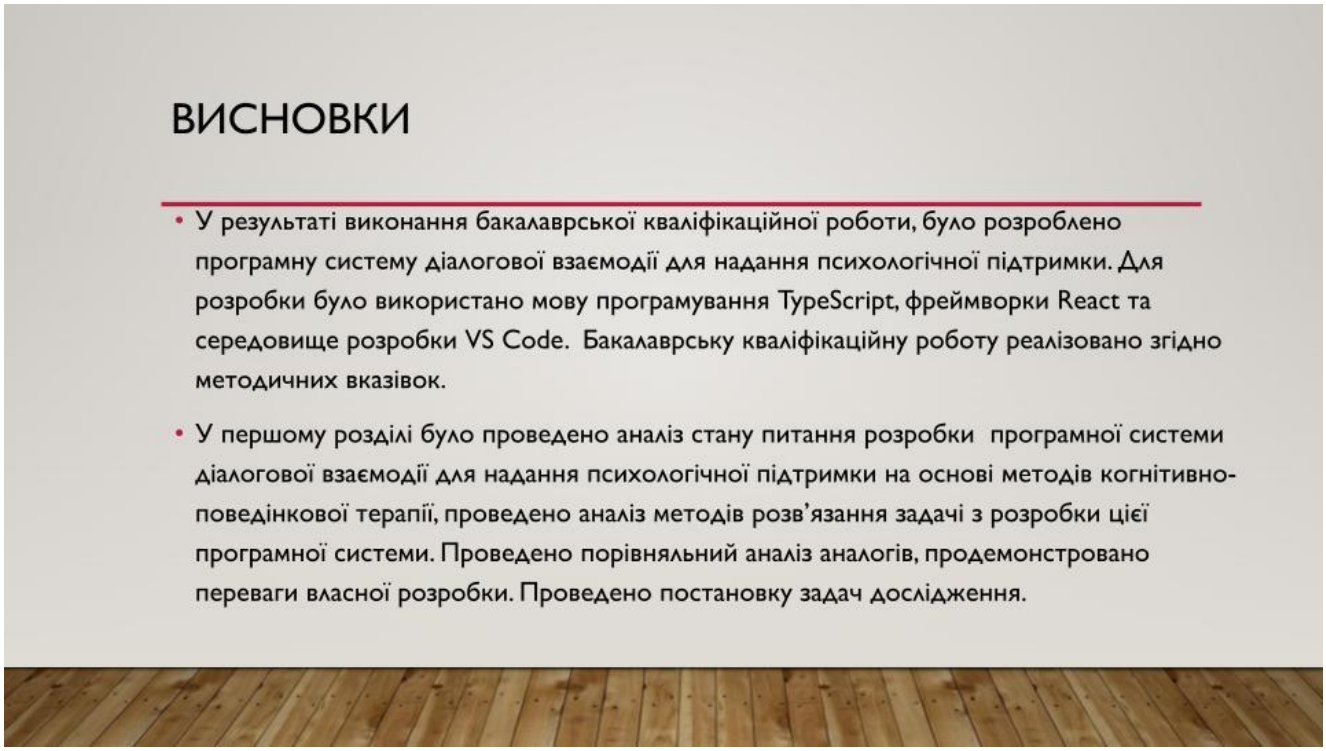

Рисунок Г.14 — Висновки (частина 1)

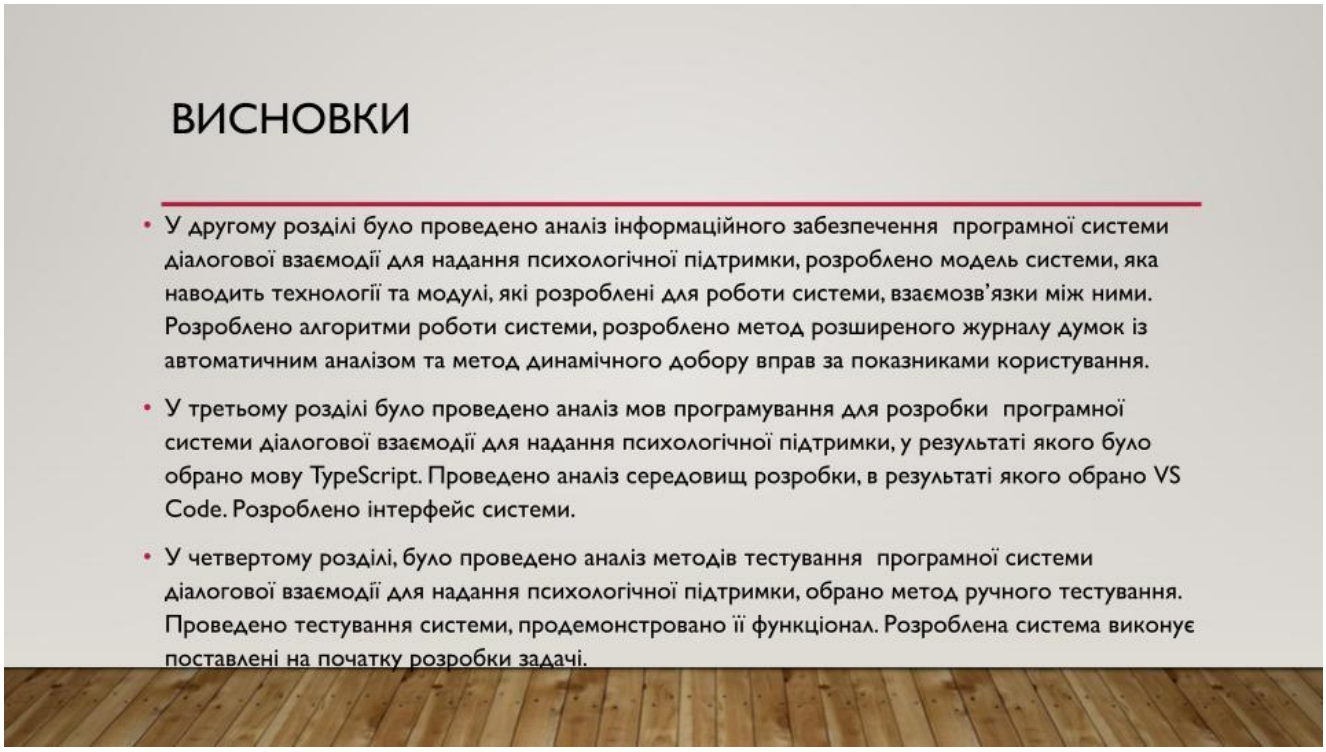

Рисунок Г.15 — Висновки (частина 2)

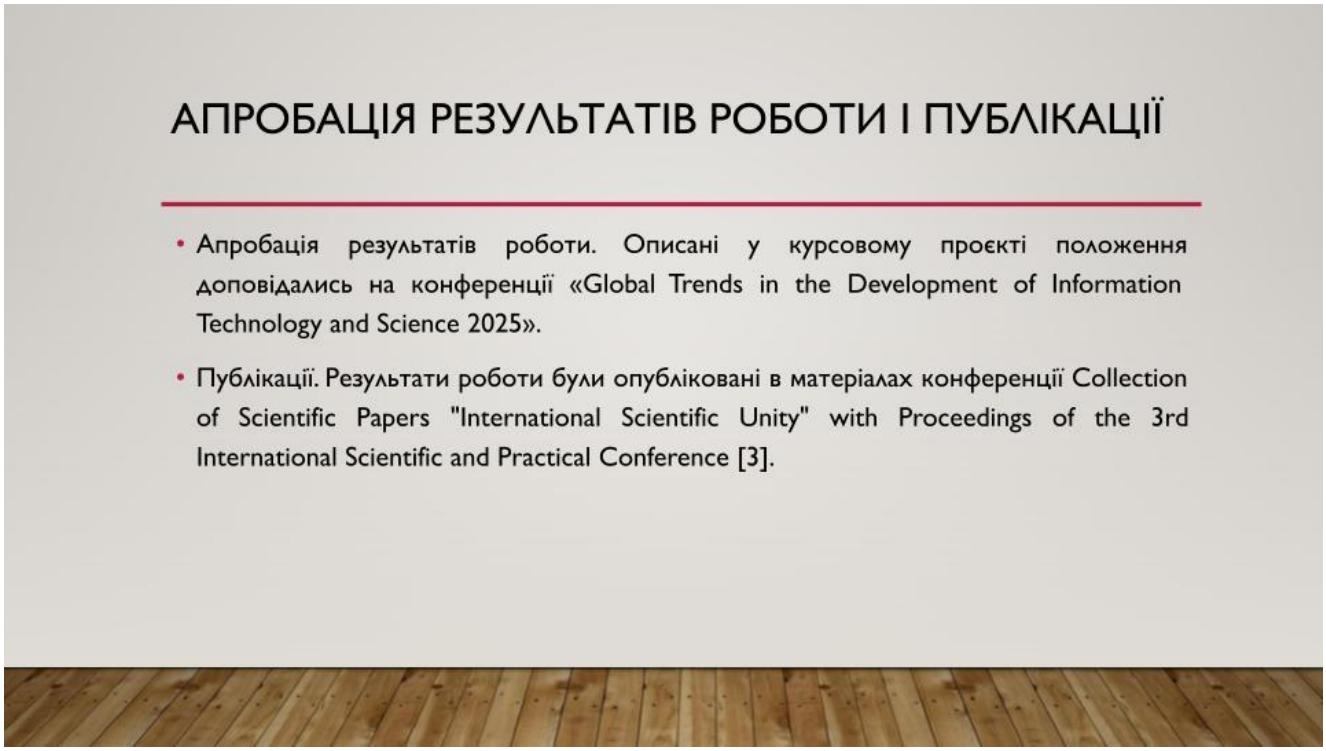

Рисунок Г.16 — Апробація результатів роботи і публікації