НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ _Голуб Б. Л.__
(підпис) (ПІБ)

"_04__" _червня__ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Мобільний додаток для інтерактивної психологічної підтримки»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми
___доцент к.т.н._____ _____ ___Голуб Б. Л.___
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
____доцент к.т.н._____ _____ ____Панкратьєв В.О__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконала: _____ _____Дрозд І. А._____
(підпис) (ПІБ студента)

КИЇВ 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук

_____ / Голуб Б.Л., доцент, к.т.н /
підпис

"_16_" __грудня____2024 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студентці Дрозд Інна Анатоліївні

Спеціальність 122 «Комп'ютерні науки»

1. Тема роботи: Мобільний додаток з інтерактивною психологічною підтримкою

Затверджена наказом ректора НУБіП України № 2246 "С" від 16.12.2024

2. Термін подання завершеної роботи на кафедру 2025 . 05 . 25
рік, місяць, число

3. Вихідні дані до роботи: опис програмного забезпечення

4. Перелік питань що розглядаються:

1. Аналіз проблемної області.
2. Вибір та обґрунтування засобів для розробки системи.
3. Проектування інформаційної системи.
4. Висновки.

Керівник бакалаврської кваліфікаційної роботи_____ / Панкратьєв В.О.
підпис ініціали та прізвище

Завдання прийняла до виконання _____ / Дрозд І.А. /
підпис ініціали та прізвище

Дата отримання завдання 2024. 12 .16
рік, місяць, число

ЗМІСТ

ВСТУП

Актуальність даної розробки полягає в тому, що в умовах сучасних соціальних та психологічних викликів, зокрема війни, психічне здоров'я стає однією з найважливіших складових загального добробуту людини. Люди часто стикаються з тривогою та емоційним виснаженням, що негативно впливає на їхній стан і якість життя. Згідно із дослідженнями Олени Зеленської, тривожність стала однією з основних проблем для українців в умовах війни. У цих умовах мобільний додаток з інтерактивною психологічною підтримкою є надзвичайно актуальним інструментом для надання допомоги у вигляді психологічних тестів, трекерів настрою та консультацій з психологами. Таке програмне забезпечення здатне підтримати емоційне здоров'я людей, забезпечити їм доступ до психологічної підтримки в будь-який час і з будь-якого місця. [1]

Мета розробки програмного забезпечення – імплементація зручних та ефективних інструментів психологічної підтримки для людей, які переживають тривогу, стрес та інші емоційні труднощі. Розробка додатку включає три етапи: моделювання, проектування та розробка. Для моделювання системи було використано об’єктно-орієнтоване моделювання, що включає діаграми прецедентів, активності, послідовності, а для проектування – діаграми пакетів та розгортання. Це дозволяє створити чітке розуміння структури системи та розробити найбільш оптимальне рішення з точки зору архітектури. Мобільний додаток буде розроблено для платформи iOS з використанням мови програмування Swift.

В процесі розробки була опублікована теза «Мобільний додаток для інтерактивної психологічної підтримки» для Всеукраїнської науково-практичної інтернет-конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2025« (м. Київ, 24 квітня 2025 р.) (ДОДАТОК А).

Структуру даної пояснювальної записки до дипломного проєкту можна представити в такій послідовності:

1. Перший розділ, де розглядається системний аналіз психологічної підтримки як об’єкта дослідження та формуються вимоги до програмного забезпечення;
2. Другий розділ, присвячений інформаційному забезпеченню;
3. Третій розділ, що обґрунтовує вибір інструментарію та описує розробку програмного забезпечення;
4. Четвертий розділ, де було проведено тестування якості та відповідності вимогам, сформованим в першому розділі, та впровадження і експлуатацію системи.

1 СИСТЕМНИЙ АНАЛІЗ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ ЯК ОБ'ЄКТУ ДОСЛІДЖЕННЯ

1.1 Постановка завдання

Метою розробки додатку є створення мобільного додатку, яке наддасть зручні та ефективні інструменти для підтримки користувачів, що мають проблеми з тривогою, самоконтролем, нерішучістю та потребують допомоги у своєму особистому та професійному житті.

Для забезпечення цього система має покривати такий функціонал для користувача:

1. Авторизація та реєстрація: користувач може зареєструватися в додатку, створити обліковий запис та увійти в систему для доступу до всіх функцій;
2. Трекер настрою: користувач може записувати свій настрій кожен день. Інтерфейс дає можливість вибирати настрій з кількох варіантів, додавати коментарі та відстежувати зміни настрою протягом часу. Також користувач отримуватиме щоденні сповіщення, щоб не забути записати свій настрій;
3. Психологічні тести: користувач може обрати та пройти будь якиц тест, який допоможе оцінити емоційний стан та краще дізнатися себе;
4. Чат з психологом: У разі потреби користувач може звернутися за підтримкою до психолога через текстовий чат. Для цього треба подати запит і згодом отримає сповіщення про доступність, або недоступність психолога. Вони спілкуються протягом години тільки у текстовому форматі, а історія чатів зберігається для зручності користувача.

Функціонал для психолога:

1. Авторизація та реєстрація: психолог може зареєструватися, пройти верифікацію (перевірка адміністратором сертифікатів, дипломів) і отримати доступ до функцій для проведення консультацій;

2. Чат з користувачами: психолог буде отримувати сповіщення про запит на консультацію, зможе взаємодіяти з користувачами через текстовий чат, надаючи їм психологічну підтримку. Історія чатів зберігається для зручності і можливої подальшої роботи з користувачем;

3. Графік роботи: психолог встановлює доступні години для консультацій, щоб отримувати сповіщення тільки в зазначений час.

Вхідною інформацією у даній системі є:

1. Інформація про користувача: ім’я користувача, електронна пошта, пароль;

2. Інформація про настрій користувача: вибір типу настрою, коментар;

3. Інформація про психологічні тести: вибір тесту, відповіді користувача.

а. Вихідною інформацією у даній системі є:

4. Історія настроїв користувача: звіт по статистиці обраних настроїв за певний період часу;

5. Результати тестів користувача: результати тестів;

6. Історія консультацій: текстові повідомлення у чатах;

7. Сповіщення для користувачів і психологів: про нові запити на консультацію, відповіді психологів, нагадування записати настрій.

1.2 Огляд інформаційних джерел та існуючих рішень

Дослідження предметної області є важливим етапом у розробці будь-якої системи, адже воно дозволяє не лише глибше зрозуміти технології, які використовуються для вирішення поставлених задач, а й проаналізувати існуючі рішення, які вже застосовуються на ринку. Огляд інформаційних джерел і рішень допомагає визначити переваги та недоліки вже наявних інструментів, а також виявити можливості для удосконалення. Такий аналіз є важливим для створення додатку, що відповідає сучасним вимогам користувачів та інтегрує найбільш ефективні інструменти для психологічної підтримки.

Існуючі рішення для даної предметної області можна поділити на: фізичні та електронні, тобто у вигляді програмного забезпечення.

1.2.1 Фізичні існуючі рішення. Найбільш поширенішим фізичним існуючим рішенням можна назвати офлайн сесії з психологом і звичайний блокнот.

Офлайн сесії з психологом – це традиційний спосіб отримання психологічної підтримки, коли користувачі особисто зустрічаються з фахівцем для обговорення своїх проблем. Офлайн консультації дають можливість психологу не лише почути, а й побачити емоційну реакцію клієнта, що дозволяє краще оцінити його стан і забезпечити більш персоналізований підхід.

Переваги:

1. Персоналізована підтримка, що забезпечує глибшу взаємодію між психологом та користувачем;
2. Можливість краще зрозуміти емоційний стан людини через невербальні сигнали (жести, міміка);
3. Постійний психолог, що створює більше довіри.

Недоліки:

1. Висока вартість послуг, що не завжди доступно для широкої аудиторії;
2. Витрачається багато часу на дорогу, що може створювати незручності;

3. Обмеження за часом психолога: не завжди можна знайти, щоб вільний час у психолога збігав із твоїм.

Блокнот/щоденник для записів – це традиційний метод самоспостереження, коли людина записує свої емоції, думки та переживання вручну. Це дозволяє людям зберігати важливі моменти і спостерігати за змінами у своєму емоційному стані. Блокнот часто використовується для рефлексії, допомагає краще зрозуміти свої почуття і думки, а також є чудовим способом знизити стрес.

Переваги:

1. Доступність і простота використання – не потребує спеціальних знань чи навичок;
2. Можливість самостійного рефлексування і спостереження за емоціями без зовнішнього втручання;
3. Особистий, інтимний процес, який дозволяє краще зрозуміти себе;
4. Бюджетно.

Недоліки:

1. Відсутність зворотного зв’язку – ніхто не дає професійних рекомендацій або допомоги;
2. Може бути складно підтримувати регулярність записів без додаткової мотивації;
3. Обмеження у глибокому аналізі – користувач може не здогадуватися про важливі аспекти свого емоційного стану без допомоги психолога, тому може інтерпртувати неправильно свій стан.

1.2.2 Автоматизовані існуючі рішення. Існуючі рішення у вигляді програмного забезпечення пропонують удосконалені можливості для користувачів у порівнянні з фізичними методами психологічної підтримки. Такі рішення можуть надавати швидкий доступ до психологічної допомоги та зручні інструменти для моніторингу емоційного стану, а також зберігання та обробки даних.

Rozmova – це українська платформа з пошуку психолога для терапії як онлайн, так і офлайн. Вона пропонує персоналізований підбір фахівців після короткої анкети, що дозволяє швидко знайти спеціаліста, який максимально відповідає вашим потребам. Користувачі можуть отримати консультацію від кваліфікованих психологів, а також на їх вебсайті пройти тести для кращого пізнання себе (рис. 1). [2]

Рис. 1 Приклад інтерфейсу додатку *«Rozmova»*

Переваги:

1. Персоналізований підбір фахівців: після заповнення анкети система допомагає підібрати психолога, який найкраще відповідає цілям користувача;
2. Велика база перевірених спеціалістів: на платформі працює понад 500 ліцензованих психологів і психотерапевтів пізних напрямків і методів терапії;
3. Зручність користування: немає комісії за пошук, оплата та онлайн дзвінок проходить на самій платформі.

Недоліки:

1. Вартість сесій – ціна за сесію може коливатися від 800 до 3600 грн, що може бути недоступно для частини користувачів;
2. Не всі фахівці можуть підійти з першого разу: підбір терапевта може зайняти час, навіть із персоналізованим підбором;
3. Не всі формати сесій зручні для кожного: хоча платформа пропонує онлайн і офлайн консультації, не в усіх містах є достатньо офлайн-спеціалістів, а онлайн-формат підходить не всім.

Krisenchat Ukrainian – це безкоштовний анонімний сервіс для надання першої психологічної допомоги через чат, доступний 24/7. Користувачі можуть звертатися за підтримкою у кризових ситуаціях через Telegram, WhatsApp або SMS і отримати консультацію від професійних психологів. Сервіс допомагає стабілізувати емоційний стан, впоратися зі стресом, тривожністю чи панічними атаками, а також дає поради щодо подальших дій. *Krisenchat* не замінює довгострокову терапію, але є ефективним способом отримати підтримку "тут і зараз", зберігаючи конфіденційність та комфорт (рис. 2). [3]

Рис. 2 Приклад інтерфейсу чату *«Krisenchat Ukrainian»* в Telegram

Переваги:

1. Анонімність і безпека: підтримка надається анонімно, що дає користувачам відчуття безпеки;
2. Доступність 24/7: сервіс працює цілодобово, що дозволяє звертатися за допомогою в будь-який час;

3. Ціна: повністю безкоштовний сервіс, що робить психологічну допомогу доступною для всіх;
4. Відсутність вікових обмежень: сервіс доступний для всіх, від дітей до дорослих.

Недоліки:

1. Не замінює довгострокову терапію: орієнтований лише на кризову допомогу;
2. Відсутність голосового чи відеозв'язку: лише текстова підтримка може бути незручною для деяких користувачів;
3. Потреба у подальшому супроводі: часто рекомендується звертатися до інших служб, оскільки сервіс не може замінити тривалу терапію;
4. Обмеженість у часі: кожного дня надається тільки година спілкування.

Програма «Здоров'я» на iPhone – це зручний та безпечний сервіс для збору, зберігання і аналізу даних про фізичний і психічний стан користувача. Вона автоматично збирає інформацію з iPhone, Apple Watch, iPad та сумісних сторонніх додатків. Програма дозволяє відстежувати кроки, сон, активність, менструальний цикл, прийом ліків, а також фіксувати настрій та проходити стандартизовані оцінки психічного здоров'я (рис. 3). [4]

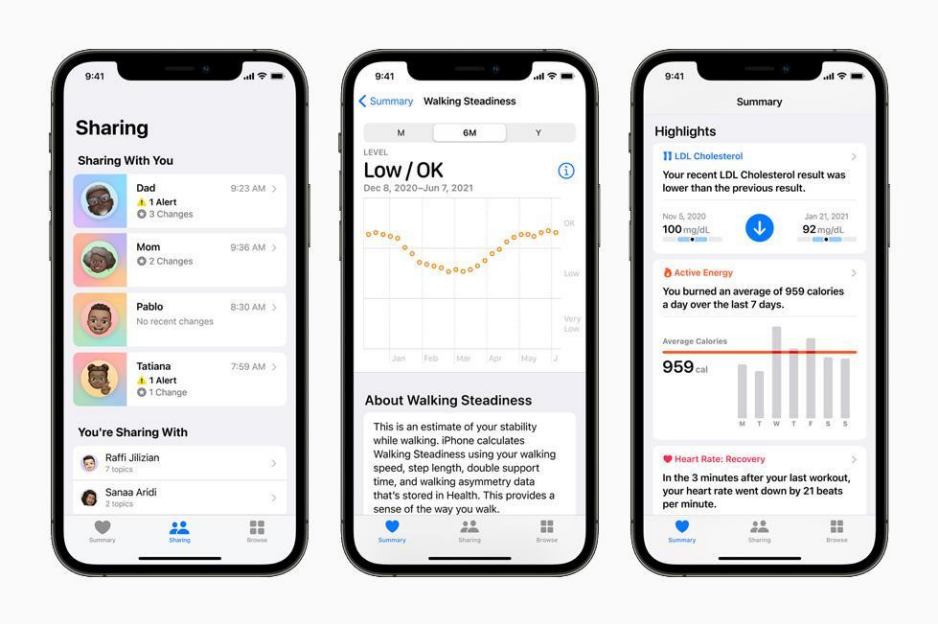


Рис. 3 Приклад інтерфейсу додатку *«Здоров'я»*

Переваги:

1. Централізоване зберігання: всі дані про здоров'я в одному місці;
2. Автоматичний збір даних: програма отримує інформацію з iPhone, Apple Watch та інших додатків;
3. Сповіщення: додаток нагадує про важливі події, наприклад, прийом ліків чи події циклу;
4. Моніторинг настрою: можливість фіксувати настрій і проходити стандартизовані тести для оцінки психічного стану.

Недоліки:

1. Програма не замінює професійну медичну допомогу, а лише допомагає відстежувати стан;
2. Деякі функції доступні не в усіх країнах;
3. Залежність від екосистеми Apple обмежує користувачів інших платформ;
4. Відсутність рекомендацій від лікарів у додатку.

1.2.3 Порівняльна таблиця. Для порівняння існуючих рішень, що були представлені раніше, було вирішено створити порівняльну таблицю на основі метрик, що відображають основні функціональні вимоги для мого додатку (табл. 1).

Таблиця 1

Порівняння існуючих рішень

Метрика	*Сесії з психологом*	*Блокнот*	*Rozmova*	*Krisenchat*	*Здоров'я*
Онлайн	-	-	+-	+	+
Вартість	Платно	Безкоштовно	Платно	Безкоштовно	Безкоштовно
Наявність спеціаліста	+	-	+	+	-
Психологічні тести	-	-	Тільки на вебсайті	-	+-
Анонімність	-	+	-	-	+
Відстеження настрою	-	+	-	-	+

На основі аналізу існуючих рішень можна зробити висновок, що на ринку наразі є різні ефективні сервіси, які надають психологічну допомогу. Однак жодне з цих рішень не пропонує комплексного підходу, який поєднує всі необхідні функції

для користувачів. Враховуючи це, створення інтегрованого рішення, яке поєднує ці функції в одному додатку, є важливим кроком у розвитку сфери психологічної підтримки, що забезпечить користувачам максимальну зручність та ефективність.

1.3 Моделювання предметної області

Моделювання допомагає краще зрозуміти, як працює певна система або область. Один із найзручніших інструментів для цього – мова UML (Unified Modeling Language), яка широко використовується в об'єктно-орієнтованому програмуванні. UML – це набір графічних схем, за допомогою яких можна показати, як функціонує система, які є зв’язки між її частинами, і як відбуваються певні процеси. [5]

Існують два основні типи UML-діаграм: структурні (наприклад, діаграми розгортання та пакетів) та поведінкові (такі як діаграми прецедентів, діаграма діяльності, діаграма послідовності). [6]

У даній роботі для кращого розуміння предметної області було вирішено використати саме поведінкові UML-діаграми, а саме: діаграму прецедентів, діаграму діяльності та діаграму послідовності.

Діаграма прецедентів є однією з ключових у моделюванні систем за допомогою UML. Вона дозволяє наочно показати, які ролі існують у певній предметній області та як саме вони взаємодіють із системою. Така діаграма включає кілька основних елементів: актор – це зовнішній користувач або інша система, що взаємодіє із розроблюваною системою; прецедент – описує функцію або дію, яку виконує система у відповідь на запит актора; сама система – це об’єкт моделювання, тобто те, що ми аналізуємо; а також зв’язки – вони відображають взаємозв’язки між акторами та прецедентами і демонструють логіку взаємодії користувачів із функціональністю системи. [7]

На рис. 1 представлено діаграму прецедентів для системи мобільного додатку.

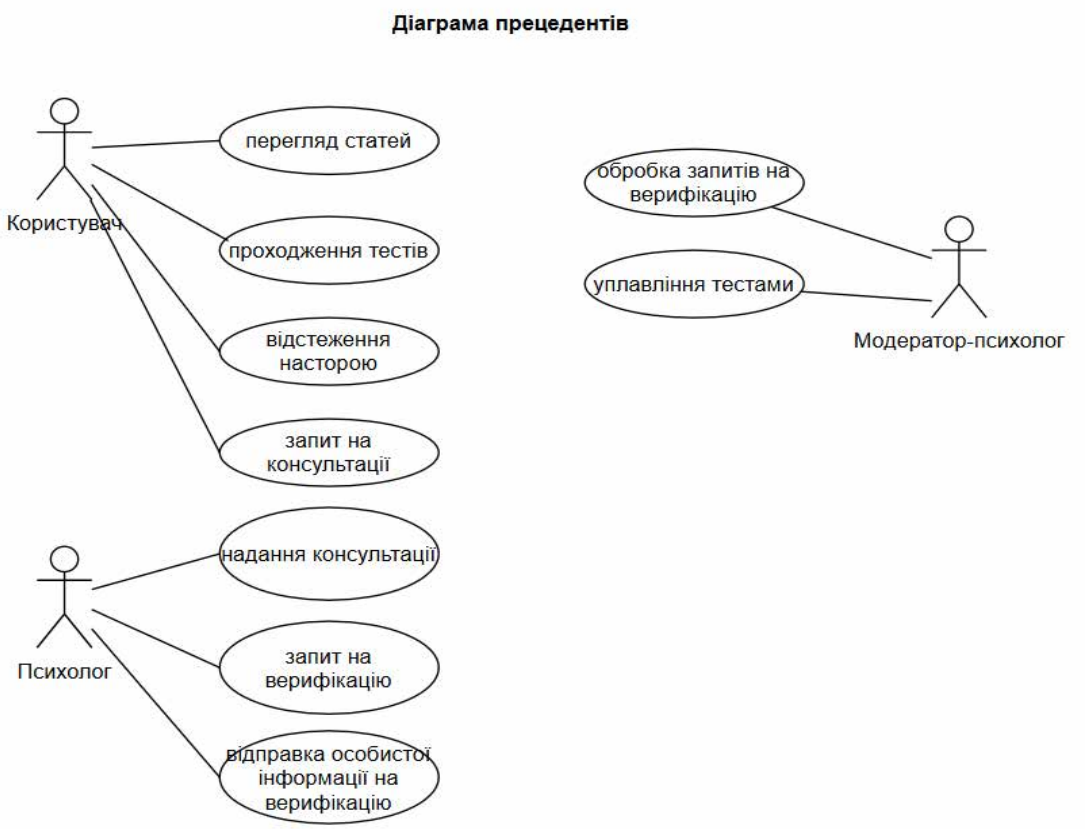


Рис. 4 Діаграма прецедентів

На даній діаграмі представлено три актори: користувач та психолог (які є основними користувачами системи) та модератор психолог, також представлено дев’ять прецедентів. Для користувача: перегляд статей, проходження тестів, відстеження настрою, запит на консультації; для психолога: надання консультації, запит та верифікацію та відправка особистої інвормації на верифікацію; для модератора-психолога: обробка запитів на верифікацію та управління тестами. Представлено лише один тип зв’язку між прецедентами та акторами – асоціація.

Діаграма активності, або діаграма діяльності, є одним із типів UML-діаграм, що використовується для зображення логіки виконання процесів у системі. Вона наочно показує, як переходять дії або кроки одного процесу в інший, створюючи загальний потік. Особливістю цієї діаграми є можливість розгалуження на кілька паралельних гілок – паралелізм, а також об’єднання цих паралельних

потоків у єдину послідовність, що дозволяє зобразити складну логіку процесів, включаючи синхронізацію та паралельне виконання дій – паралелізм [8].

На рис. 5 представлено діаграму діяльності для системи мобільного додатку.

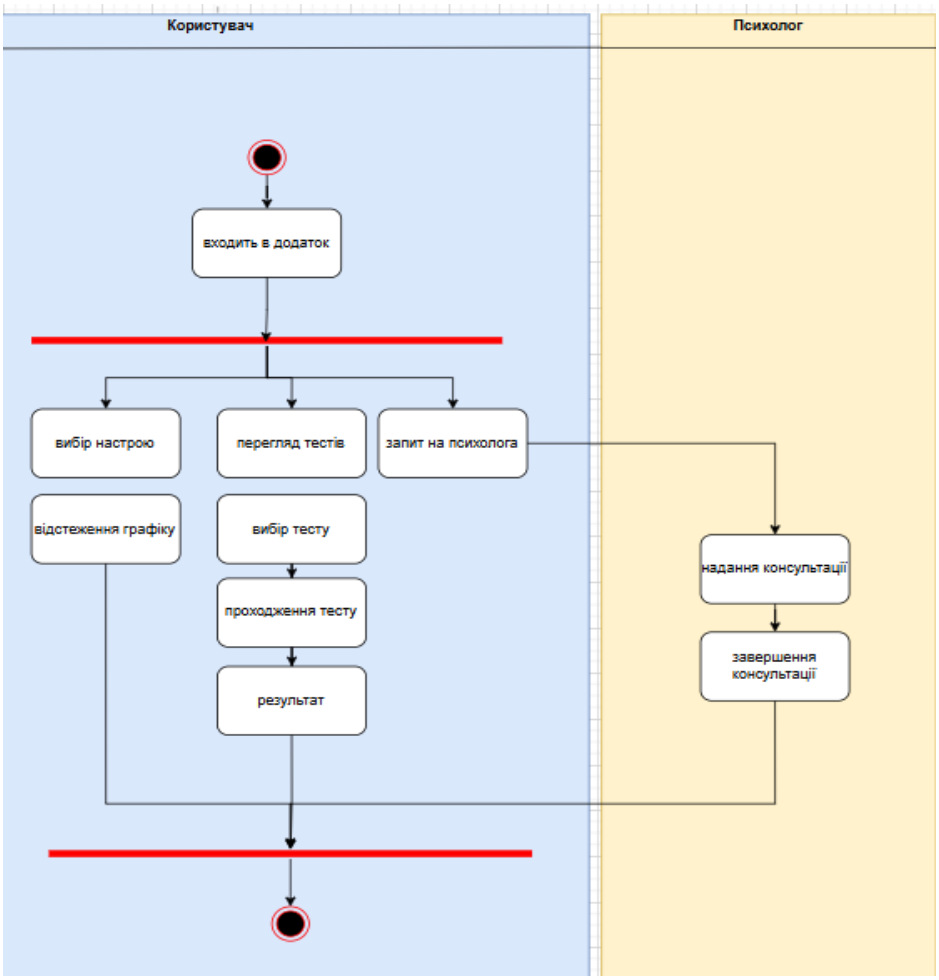


Рис. 5 Діаграма діяльності

Дана діаграма представляє собою варіанти взаємодії між користувачем та психологом через систему. Вона демонструє процеси, які можуть виконуватись паралельно, зокрема вибір настрою, перегляд тестів, вибір тесту, проходження тесту та запит на консультацію з психологом. Користувач може одночасно вибирати настрій, переглядати тести і записуватись на консультацію, що відображається розпаралелюванням потоків. Після завершення цих процесів відбувається злиття потоків, і система переходить до наступної дії, зокрема до надання консультації психологом.

Діаграма послідовності є одним із типів UML-діаграм, що дозволяє відобразити порядок обміну повідомленнями між об’єктами в межах певного сценарію. Вона показує, як саме об'єкти взаємодіють між собою у часі, коли виконуються ті чи інші дії. Серед основних складових такої діаграми можна виділити лінії життя, які позначають період існування об'єкта в межах сценарію; смуги активації, які демонструють час, коли об'єкт активно виконує певну дію; а також повідомлення – вони можуть бути синхронними, якщо передбачають відповідь, або асинхронними, коли відповідь не очікується. Така візуалізація дозволяє краще зрозуміти хід взаємодії між елементами системи [9].

На рис. 6 представлено діаграму діяльності для системи мобільного додатку.

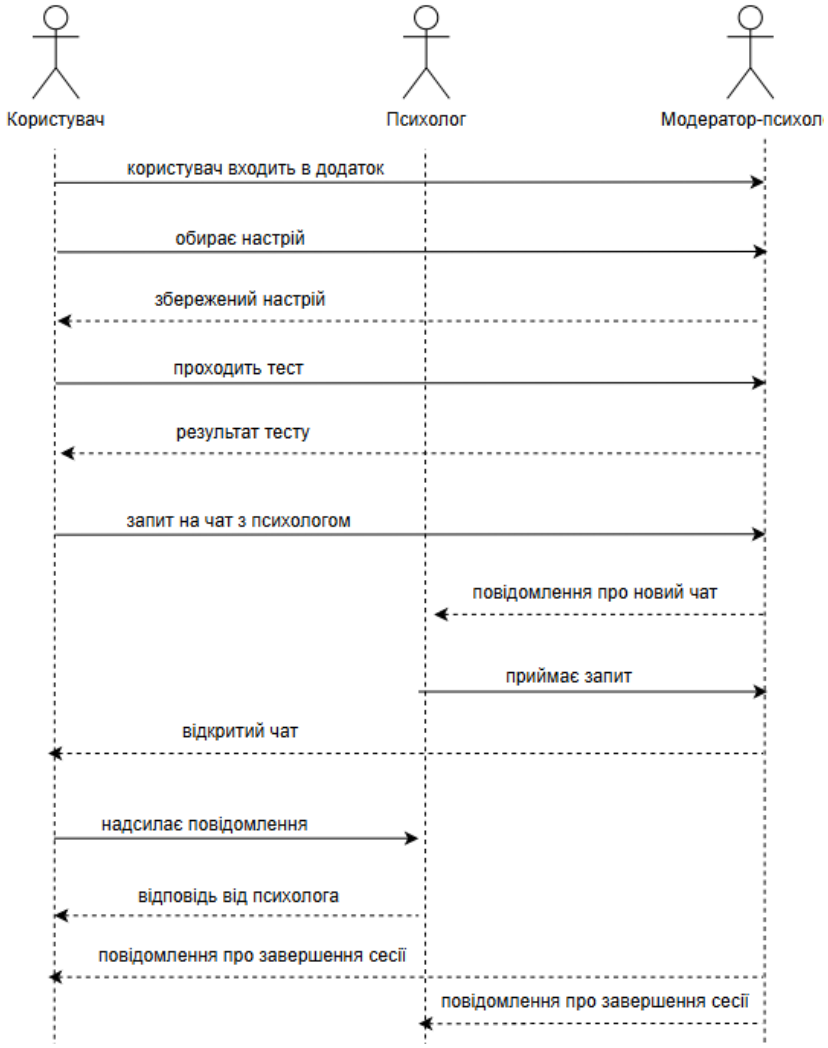


Рис. 6 Діаграма послідовності

На даній діаграмі показано взаємодію трьох акторів: Користувач, Психолог і Елемент системи, що координує взаємодії між ними. Всі актори взаємодіють через послідовність дій, включаючи запис настрою, проходження тестів, запит на чат з психологом і відкриття чату. Користувач ініціює запит на консультацію, отримує відповідь і повідомлення від психолога. Елемент системи обробляє запити на консультацію, приймає їх і надсилає повідомлення про завершення сесії. Також після завершення сеансу надсилаються сповіщення про завершення сесії. Діаграма демонструє, як повідомлення передаються між користувачем, психологом та системою, що дає чітке уявлення про взаємодію учасників.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

База даних є невід'ємною складовою будь-якого програмного забезпечення, адже вона відповідає за збереження, організацію та швидкий доступ до інформації. Незалежно від того, яким способом реалізована система, саме база даних забезпечує її стабільність, захист інформації та підтримання цілісності даних.

Одним із перших кроків у створенні бази даних є розробка логічної моделі – вона дозволяє наочно представити структуру інформації, яку буде збережено, і показати, як окремі частини цієї інформації пов'язані між собою. Така модель описує сутності, їхні властивості та зв'язки між ними у логічному вигляді, що суттєво полегшує процес розуміння майбутньої системи. Це також допомагає виявити можливі помилки ще до етапу фізичної реалізації – наприклад, уникнути дублювання даних або втрати важливої інформації. Логічна модель виступає основою, на якій будується вже безпосередньо реалізована база даних, і дозволяє більш точно визначити, яка саме інформація необхідна для повноцінного функціонування системи [10].

Логічна модель даних складається з трьох основних елементів:

1) сутності (*entities*) – це конкретні об'єкти, про які зберігається інформація, вони відображають таблиці в базі даних та їх поля;

2) атрибути (*attributes*) – це властивості сутностей, які описують їх певні характеристики, вони визначаються в межах сутностей, і допомагають зрозуміти, яка інформація має зберігатись в кожній сутності;

3) відношення (*relationships*) – це зв'язки між сутностями, що визначають характер зв'язків такі як один до одного, один до багатьох, багато до багатьох:

а) один до одного – це тип зв'язку, що вказує на те, що одна сутність може бути пов'язана лише з однією іншою сутністю;

b) один до багатьох – це тип зв’язку, що вказує на те, що одна сутність може мати відношення з багатьма екземплярами іншої сутності;

c) багато до багатьох – це тип зв’язку, що вказує на те, що багато сутностей однієї таблиці можуть бути пов’язаними з багатьма сутностями іншої таблиці [10].

Існує чимало інструментів для створення логічної моделі даних, однак для реалізації цього завдання було обрано Erwin Data Modeler – сучасний та функціональний засіб, що дозволяє ефективно працювати з моделями даних. Його вибір зумовлений низкою переваг, серед яких варто відзначити зручний та зрозумілий інтерфейс, великий набір можливостей для проєктування баз даних, а також функцію автоматичної генерації SQL-коду на основі створеної моделі. Це значно спрощує процес розробки та дозволяє швидше перейти від моделювання до реалізації. Спираючись на детальний аналіз предметної області та сформульовані вимоги, було створено логічну модель даних для програмного забезпечення мобільного додатку [11].

В ДОДАТКУ Б представлено зображено чотирнадцять сутностей:

1) *Користувач:* сутність, що представляє користувачів системи. Вона містить один ключовий атрибут – *ID_користувача* та кілька неключових атрибутів: *прізвище, ім'я, електронна_пошта, пароль, дата_народження, стать, фото_профілю, дата_реєстрації, сертифікат, статус_верифікації* і *статус_активності*.

2) *Психолог:* сутність, що представляє психолога, який надає консультації. Містить ключовий атрибут *ID_психолога* та кілька неключових атрибутів: *прізвище, ім'я, електронна_пошта, пароль, дата_народження, стать, фото_профілю*.

3) *Тип_настрою:* сутність, яка визначає типи настрою користувачів. Містить два атрибути: *ID_типу_настрою* та *назва_типу_настрою*.

4) *Модератор:* модератор відповідає за управління контентом та перевірку психологів. Містить ключовий атрибут *ID_модератора* та кілька неключових атрибутів: *прізвище*, *ім'я*, *пароль*, *електронна_пошта*.

5) *Тест:* це сутність, що містить інформацію про психологічні тести, доступні для користувачів. Включає ключові атрибути *ID_тесту* та *ID_модератора*, а також неключові атрибути: *назва_тесту*, *опис_тесту*, *мінімальний_бал* та *максимальний_бал*.

6) *Консультація:* сутність, яка зберігає інформацію про консультації між психологом і користувачем. Включає ключові атрибути *ID_консультації*, *ID_психолога*, та *ID_користувача*, а також неключові атрибути: *час_початку* і *тривалість*.

7) *Відгук_користувача:* сутність для збереження відгуків користувачів про консультації. Містить ключові атрибути *ID_консультації* та *ID_психолога*, а також неключові атрибути: *оцінка* та *коментар*.

8) *Повідомлення_в_чаті:* це сутність, що зберігає повідомлення, які надсилаються під час консультацій між користувачем і психологом. Містить атрибути: *ID_повідомлення*, *час_повідомлення*, *відправник*, *текст_повідомлення*, *ID_консультації*, та *ID_психолога*.

9) *Графік_психолога:* Сутність, яка визначає час, коли психолог доступний для консультацій. Містить атрибути: *час_початку_періоду* і *час_закінчення_періоду*, а також ключові атрибути: *ID_психолога* та *ID_запису_графіку*.

10) *Щоденник_настрою:* сутність, яка дозволяє користувачам записувати свої настрої. Включає атрибути: *дата_запису_настрою*, *коментар_до_настрою*, *ID_користувача*, *ID_типу_настрою*, і *ID_запису_настрою*.

11) *Результат_тесту:* сутність для збереження результатів психологічних тестів користувачів. Включає атрибути: *ID_результату_тесту*, *ID_користувача*, *ID_тесту*, та *бал*.

12) *Питання_тесту:* сутність для збереження питань, які входять до складу тесту. Включає атрибути: *ID_питання*, *текст_питання*, і *ID_тесту*.

13) *Відповідь_на_тест:* сутність для збереження відповідей користувачів на питання тестів. Включає атрибути: *ID_відповіді*, *текст_відповіді*, *бал_за_відповідь*, та *ID_питання*.

14) *Тип_результату:* сутність для визначення типів результатів тестів. Містить атрибути: *ID_типу_результату* і *опис_типу_результату.*

Всі зв'язки між сутностями в даній логічній моделі даних є переважно типу *"один до багатьох"*, що відображає реальні взаємодії між користувачами, психологами та іншими елементами системи. Наприклад, один користувач може мати кілька записів у щоденнику настрою, або один психолог може проводити багато консультацій. Такий тип зв'язку дозволяє зберігати масштабованість та забезпечує гнучкість у обробці даних. Виключенням є зв'язок між Консультацією та Відгуком користувача, де зв'язок має тип *"один до одного".* Кожен користувач, після консультації з психологом, може залишити лише один відгук. Це дозволяє зберегти чіткість та уникнути дублювання інформації, що є важливим для аналізу та збереження цілісності даних.

Також вся модель відповідає третій нормальній формі, що гарантує відсутність транзитивних залежностей, запобігаючи дублюванню даних і забезпечуючи їх цілісність.

2.2 Вибір системи управління інформаційною базою

Для мобільного додатку важливо мати можливість зберігати та обробляти інформацію. З цією метою було обрано реляційну базу даних.

Реляційна база даних базується на реляційній моделі, де інформація зберігається у вигляді таблиць, які складаються з рядків і стовпців. Кожен рядок відповідає окремому запису, а стовпці містять дані певного типу. Такий підхід дозволяє зручно організовувати інформацію у взаємопов'язаних таблицях і здійснювати складні запити для роботи з даними [12].

Серед переваг реляційних баз даних можна виділити:

- структурованість даних, за рахунок таблиць, що спрощує роботу з даними;
- високий рівень незалежності даних, тобто зміна структури таблиць не впливає на роботу з даними;
- цілісність даних, забезпечена шляхом використання ключів.

Але при роботі з реляційними базами даних варто пам'ятати про деякі особливості, наприклад, перед створенням об'єктів бази даних варто визначити відношення між ними та структуру, потрібно правильно використовувати ключі, щоб не виникало проблем під час виконання запитів [13].

Для даного програмного забезпечення в якості бази даних було обрано MySQL, через низку переваг:

- продуктивність та масштабованість: MySQL має високу продуктивність, що важливо для додатку, який обробляє великі обсяги даних, зокрема при зберіганні результатів тестів, консультацій і налаштувань користувачів. Вона підтримує швидке виконання запитів і дозволяє ефективно працювати навіть із великими базами даних, що важливо для забезпечення безперебійної роботи додатку при зростанні кількості користувачів.

- підтримка транзакцій: MySQL підтримує транзакції, що дозволяє забезпечити цілісність даних під час виконання кількох операцій. Це є критично важливим для додатку, де зберігаються чутливі дані, такі як історія консультацій та психологічних тестів, тому транзакційність допомагає зберегти узгодженість і надійність даних.
- сумісність з iOS: MySQL добре підтримується на платформі iOS через численні бібліотеки та інструменти для інтеграції з мобільними додатками. Це дозволяє ефективно взаємодіяти з базою даних за допомогою популярних інструментів, таких як Node.js, Express, і Sequelize.
- безпека: MySQL має вбудовані функції для захисту даних, що важливо для додатку, який зберігає персональні дані користувачів, історію консультацій та медичні результати. Це включає підтримку шифрування, управління правами доступу та аутентифікації, що дозволяє забезпечити конфіденційність даних.
- масштабованість: MySQL підтримує як локальне, так і розподілене зберігання даних, що дозволяє згодом збільшувати масштаб бази даних, якщо додаток зростатиме в популярності. Ти зможеш легко налаштувати реплікацію і масштабувати систему для обробки більшої кількості користувачів та даних.
- відкритий код і безкоштовність: MySQL є відкритим програмним забезпеченням, що дозволяє зменшити витрати на ліцензування та надає гнучкість у налаштуванні та оптимізації системи відповідно до специфікацій проекту. Крім того, наявність великої спільноти розробників означає, що ти завжди можеш знайти рішення для будь-яких питань.

- легкість в адмініструванні: MySQL має зручні інструменти для адміністрування та моніторингу роботи бази даних, що полегшує підтримку додатку і дозволяє швидко вирішувати будь-які проблеми, що виникають під час роботи з даними [14].

2.3 Створення інформаційної бази

2.3.1 Створення бази даних. Для розробки бази даних та її елементів застосовано SQL (Structured Query Language) – універсальну мову програмування, призначену для роботи з реляційними базами даних. За допомогою SQL можна створювати таблиці, змінювати їх структуру, додавати, видаляти або оновлювати інформацію, а також отримувати потрібні дані за складними запитами [15].

На рис. 7 представлено фрагмент коду, який демонструє процес створення бази даних для мобільного додатку.

```
CREATE DATABASE helpdb
```

Рис. 7 Код створення бази даних

2.3.2 Створення таблиць. Відповідно до логічної моделі даних, розглянутої в розділі 2.1, було реалізовано SQL-код для генерації необхідних таблиць. Як приклад, на рис. 8 показано фрагмент коду, що відповідає за створення таблиці *User* – її структура повністю відображає атрибути та властивості відповідної сутності з логічної моделі [16].

Код створення всіх інших таблиць наведено в ДОДАТКУ Б.

```
CREATE TABLE `user`
(
    `ID_user` int(11)                          NOT NULL,
    `surname`  varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `name`  varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `password`  varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `date_of_birth` date    DEFAULT NULL,
    `sex`  varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `profile_picture` blob,
    `registration_date` date  NOT NULL,
    `sertificate` blob   NOT NULL,
    `verification_status` bit(1)  NOT NULL,
    `activity_status` bit(1)  NOT NULL,
    PRIMARY KEY (`ID_user`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;
```

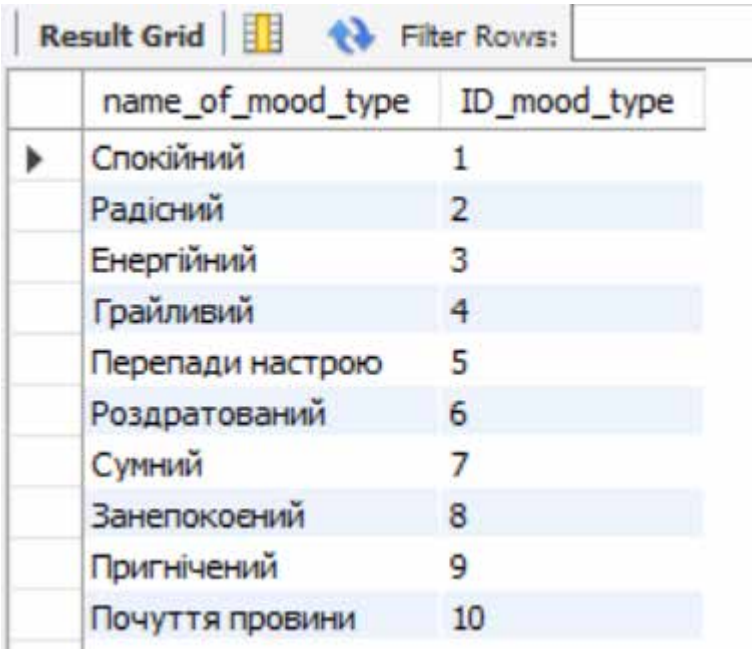
Рис. 8 Код створення таблиці User

2.3.3 Занесення умовно-постійної інформації. Умовно-постійна інформація – це дані, які не підлягають регулярним змінам і зазвичай залишаються незмінними протягом усього життєвого циклу системи. Вони відіграють ключову роль у забезпеченні стабільності та узгодженості роботи програмного забезпечення.

Така інформація зберігається в структурі бази даних окремо від оперативних даних, що дозволяє оптимізувати запити та зменшити навантаження на систему. Її особливість полягає в тому, що вона переважно використовується для зчитування, а не для модифікації.

Включення умовно-постійних даних до архітектури додатка дозволяє підвищити ефективність обробки інформації, спростити підтримку коду та забезпечити консистентність даних у довгостроковій перспективі.

На рис. 9 представлено візуальне предзставлення таблиці з умовнопостійною інформацією для таблиці *Тип настрою*. Це формує довідник для подальшого використання в системі.



name_of_mood_type	ID_mood_type
Спокійний	1
Радісний	2
Енергійний	3
Грайливий	4
Перепади настрою	5
Роздратований	6
Сумний	7
Занепокоєний	8
Пригнічений	9
Почуття провини	10

Рис. 9 Таблиця Типи настрою із заповненою умовно постійною інформацією

2.3.4 Створення процедур. Процедури в SQL – це спеціально створені блоки коду, які містять послідовність команд для виконання конкретних дій з базою даних. Вони діють як готові сценарії, які можна викликати за потреби, що значно спрощує роботу з даними.

Головна перевага процедур полягає в тому, що вони дозволяють уникнути багаторазового написання одного й того самого коду. Замість цього досить один раз створити процедуру та використовувати її знову, що робить код більш структурованим, легшим у підтримці та ефективнішим у роботі [17].

Процедура *get_user_mood_history* дозволяє отримати історію настроїв конкретного користувача. На рис. 10 зображена її реалізація: вона приймає user_id

як вхідний параметр і виводить всі записи про настрої, що були зроблені користувачем, включаючи тип настрою, коментарі та дату запису. Процедура об'єднує таблиці *mood_diary ma mood_type*, щоб забезпечити повну інформацію про настрої користувача.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`get_user_mood_history`(IN user_id INT)
BEGIN
   SELECT
      md.ID_mood_recording,
      md.ID_user,
      mt.name_of_mood_type,
      md.comment_of_mood,
      md.date_of_recording_mood
   FROM mood_diary md
   JOIN mood_type mt ON md.ID_mood_type = mt.ID_mood_type
   WHERE md.ID_user = user_id;
END
```

Рис. 10 Код створення процедури get_user_mood_history

На рис. 11 зображено код процедури *get_user_test_results*, яка дозволяє отримати результати тестів для конкретного користувача. Вона приймає user_id як вхідний параметр і повертає список всіх тестів, які пройшов користувач, разом з результатами. Процедура об'єднує таблиці user_result, user, test та type_of_result для відображення необхідної інформації: назви тесту та результату.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`get_user_test_results`(IN user_id INT)
BEGIN
   SELECT
      u.ID_user,
      u.name,
      u.surname,
      t.title AS test_title,
      tr.text_of_result AS test_result
   FROM user_result ut
   JOIN user u ON ut.ID_user = u.ID_user
   JOIN test t ON ut.ID_test = t.ID_test
   JOIN type_of_result tr ON ut.ID_type_of_result =
tr.ID_type_of_result
   WHERE ut.ID_user = user_id;
END
```

Рис. 11 Код створення процедури get_user_test_results

На рис. 12 зображено код процедури *get_psychologist_consultations*, яка дозволяє психологу переглядати всі консультації, які він/вона провели в певному періоді часу. Процедура приймає psychologist_id, start_date та end_date як параметри та повертає консультації, що проводились психологом у вказаний період. Вона об'єднує таблиці consultation та user для виведення інформації про користувачів та час консультацій.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_psychologist_consultations`(IN psychologist_id INT, IN start_date DATETIME, IN end_date DATETIME)
BEGIN
  SELECT
    c.ID_consultation,
    u.name AS user_name,
    u.surname AS user_surname,
    c.start_time_of_consultation,
    c.duration_of_consultation
  FROM consultation c
  JOIN user u ON c.ID_user = u.ID_user
  WHERE c.ID_psychologist = psychologist_id
   AND c.start_time_of_consultation BETWEEN start_date AND end_date;
END
```

Рис. 12 Код створення процедури get_psychologist_consultations

2.3.4 Створення подань. Тригери в базах даних – це автоматизовані механізми, які активуються у відповідь на певні події, такі як додавання (INSERT), видалення (DELETE) або зміна (UPDATE) даних у таблиці.

Основні особливості тригерів:

- вони прив'язуються до конкретних таблиць і спрацьовують без участі користувача;
- їх головне призначення – контролювати цілісність даних та виконувати додаткові перевірки або дії;
- завдяки автоматизації вони допомагають уникнути рутинних помилок та спрощують логіку роботи з даними [18].

На рис. 13 представлений код створення подання *user_mood_history_view*, що дозволяє користувачеві переглядати свою історію настроїв. Воно поєднує тип настрою, дату запису та коментар користувача, створюючи зручне уявлення про емоційний стан у різні дні. Така структура корисна для візуалізації змін настрою та самоспостереження.

```
DROP VIEW IF EXISTS user_mood_history_view;
DROP VIEW IF EXISTS user_test_results_view;

CREATE VIEW user_mood_history_view AS
SELECT
   md.ID_mood_recording,
   md.ID_user,
   mt.name_of_mood_type,
   md.comment_of_mood,
   md.date_of_recording_mood
FROM mood_diary md
JOIN mood_type mt ON md.ID_mood_type =
mt.ID_mood_type;
```

Рис. 13 Код створення тригера UpdateTaskCreateTime

На рис. 14 представлений код створення подання *user_test_results_view*. Це подання надає користувачу інформацію про результати пройдених психологічних тестів. Воно об'єднує дані про користувача, назву тесту та текстовий результат. Завдяки цьому користувач може легко аналізувати свої результати.

```
CREATE VIEW user_test_results_view AS
SELECT
    u.ID_user,
    u.name,
    u.surname,
    t.title AS test_title,
    tr.text_of_result AS test_result
FROM user_result ut
JOIN user u ON ut.ID_user = u.ID_user
JOIN test t ON ut.ID_test = t.ID_test
JOIN type_of_result tr ON ut.ID_type_of_result =
tr.ID_type_of_result;
```

Рис. 14 Код створення подання user_test_results_view

2.3.5 Створення користувачів. Для забезпечення безпеки системи та доступу користувачів до бази даних було створено роль користувача, який може лише змінювати записи в таблиці (рис. 15).

```
DECLARE @username1 VARCHAR(255);
DECLARE @password1 VARCHAR(255);
SELECT @username1 = userName, @password1 = userPassword
FROM User
WHERE userName = 'username1';
CREATE USER @username1 IDENTIFIED BY @password1;
GRANT INSERT, UPDATE, DELETE helpbd TO @username1;
```

Рис. 15 Код створення користувача та надання йому прав

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Діаграма пакетів слугує ефективним інструментом для проектування модульної архітектури програмного забезпечення, організовуючи класи, інтерфейси та інші компоненти у логічні групи. Вона наочно відображає структуру системи, спрощуючи аналіз взаємозв'язків між її частинами та керування складністю розробки. Використання цього інструменту дозволяє застосовувати модульний підхід, що передбачає чітке розмежування функціональних блоків на підсистеми, що значно покращує процес розробки, супроводу та тестування програмного продукту [19].

Для розробки цього програмного забезпечення обрана архітектура *Model-View-ViewModel (MVVM)*. MVVM є шаблоном проектування, що забезпечує чітке розділення графічного інтерфейсу (View) від бізнес-логіки (Model) за допомогою проміжного компонента ViewModel. Це дозволяє спростити розробку та тестування програмного забезпечення, адже бізнес-логіка може бути переносною та повторно використовуваною, тоді як інтерфейс може залежати від платформи. ViewModel виступає посередником між Model та View, забезпечуючи обмін даними та взаємодію з користувачем [20].

На рис. 16 діаграма пакетів зображує архітектуру яка чітко розділяє дані, бізнес-логіку та інтерфейс користувача. Шар Model містить пакети, що відповідають за обробку та збереження даних, такі як *Mood, Consultation* і *Test*. Шар ViewModel включає пакети *Domain Mood*, *Domain Consultation* і *Domain Test*, які здійснюють бізнес-логіку та готують дані для передачі до View. Шар View представлений пакетами *Presentation Mood, Presentation Consultation* та *Presentation Test*, що відповідають за відображення інформації на інтерфейсі користувача.

Серверна частина включає *Rest API Server*, який обробляє запити від мобільного додатку, та *MySQL Database*, що зберігає дані. Адміністративна панель складається з пакетів *Psychologist Verification* для перевірки психологів та *Test Management* для управління тестами. Зв'язки між пакетами позначають залежність між компонентами, що дозволяє ефективно обмінюватися даними та забезпечувати правильну роботу додатку.

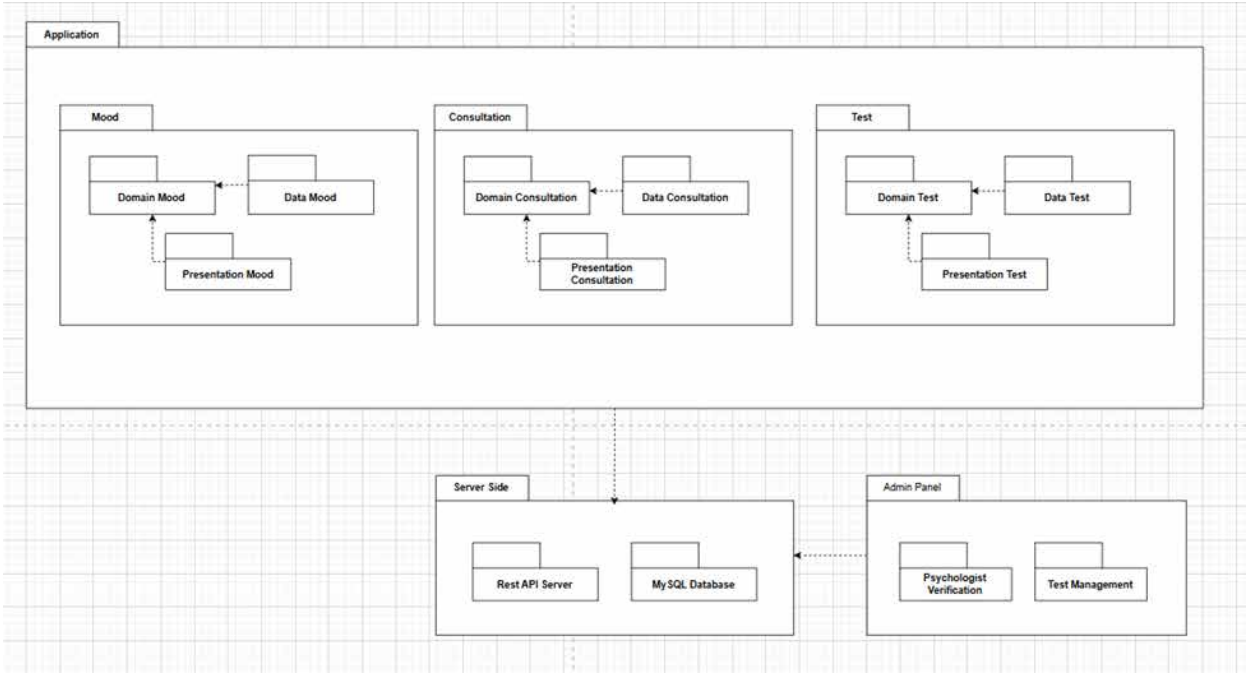


Рис. 16 Діаграма пакетів

3.2 Вибір інструментарію для створення ППЗ

Для створення інтерфейсу користувача мобільного додатку було обрано *Figma* – сучасний інструмент для векторного дизайну та прототипування, через такі переваги:

- крос-платформенність – працює у веббраузері та є десктопний додаток (Windows, macOS), що дозволяє використовувати її на будь-якому пристрої без встановлення додаткового ПЗ;
- прототипування та анімація – інструмент дозволяє створювати інтерактивні прототипи з анімацією переходів, що допомагає візуалізувати поведінку додатку перед початком розробки;
- система компонентів і стилів – можливість створювати перевикористовувані компоненти (кнопки, форми, меню) та глобальні стилі спрощує підтримку дизайн-системи та забезпечує консистентність інтерфейсу;
- безкоштовний план – доступ до всіх необхідних для початку роботи функцій без окремої доплати [21]

Для розробки самого програмного забезпечення було обрано мову програмування *Swift,* через такі переваги:

- висока продуктивність: компільована мова з оптимізацією під процесори Apple;
- сучасний синтаксис: лаконічний та зрозумілий код з підтримкою функціонального програмування;
- безпека пам'яті: автоматичне керування пам'яттю (ARC) запобігає витокам;
- повна інтеграція з iOS: нативна підтримка всіх фреймворків Apple;
- швидка розробка - інтерактивна playground-середа для швидкого тестування ідей;

- стабільність: строга типізація зменшує кількість runtime-помилок [22].

Як інтегроване середовище розробки було вирішено використати *XCode*. Серед переваг даного середовища можна виділити:

- уніфікований інструментарій: єдине середовище для дизайну, кодування та тестування;
- візуальний редактор інтерфейсів: Interface Builder для швидкого створення UI;
- потужний дебагер: інструменти аналізу пам'яті та продуктивності;
- симулятори пристроїв: тестування на всіх моделях iPhone та iPad;
- безкоштовний доступ: повноцінне професійне середовище від Apple [23].

Для реалізації серверного компоненту системи було обрано *фреймворк Node.JS* через його такі переваги:

- висока продуктивність: асинхронна архітектура на основі подій дозволяє обробляти тисячі запитів одночасно;
- екосистема npm: найбільший репозиторій бібліотек для будь-яких задач;
- масштабованість: ідеально підходить для мікросервісної архітектури;
- швидка розробка: мінімальні накладні витрати на створення API.

Середовище розробки *Visual Studio Code:*

- інтелектуальне автодоповнення: розумна підтримка JavaScript/TypeScript
- вбудований термінал: можливість запуску сервера без перемикання між програмами;
- потужний дебагер: інструменти для налагодження серверного коду;
- багатий вибір розширень: підтримка Docker, ESLint, REST-клієнтів тощо [24].

Ці інструменти забезпечують ефективну, масштабовану та зручну систему для взаємодії з базою даних, що підходить для швидкої розробки і подальшого масштабування.

3.3 Алгоритмізація та програмування програмних модулів

Алгоритмізація та програмування окремих частин програмного забезпечення є важливим етапом у розробці додатку. В цьому розділі ми розглянемо алгоритми для основних функцій програми та їх реалізацію в програмному коді.

Одним з основних алгоритмів є авторизація користувача, що передбачає перевірку введених даних та забезпечення доступу до системи. Алгоритм авторизації користувача полягає у перевірці наявності користувача з таким самим email, порівнянні введеного паролю з збереженим у базі даних та наданні доступу у разі успішної перевірки. Блок-схема цього алгоритму зображена на рис. 17.

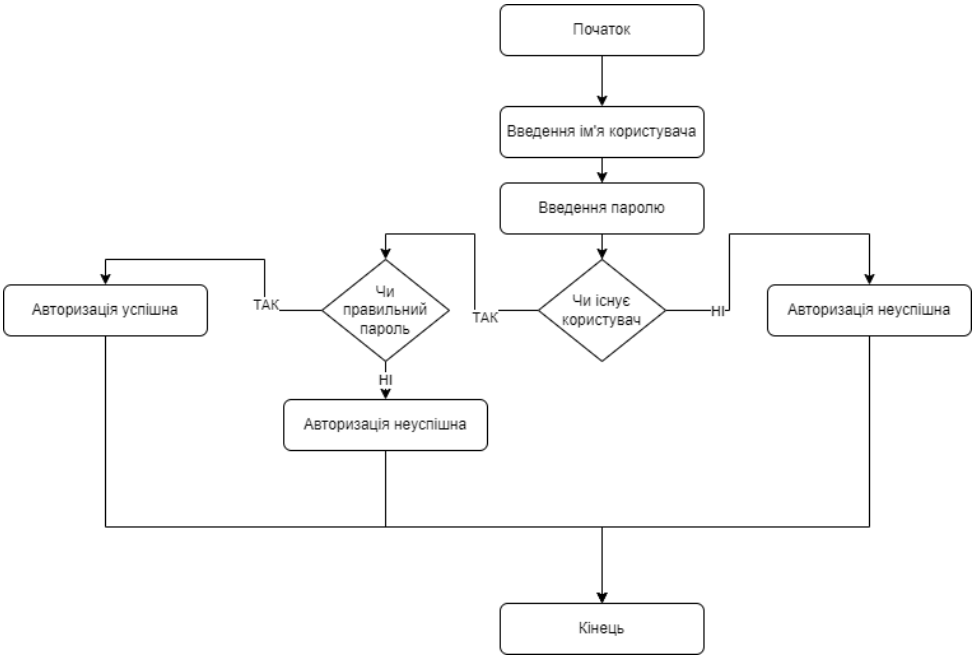


Рис. 17 Алгоритм авторизації користувача в системі

Частина коду для реалізації алгоритму авторизації користувача:

```
const bcrypt = require('bcrypt');
const { User } = require('./models'); // Імпортуємо модель користувача з Sequelize

async function loginUser(email, password) {
  try {
    // Перевірка, чи існує користувач з таким email
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return { success: false, message: "Користувач не знайдений" };
    }

    // Порівняння введеного паролю з хешованим паролем з бази даних
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      return { success: false, message: "Невірний пароль" };
    }

    // Якщо авторизація успішна, повертаємо користувача
    return { success: true, message: "Авторизація успішна", user };
  } catch (error) {
    console.error(error);
```

Рис. 18 Авторизація користувача

Ще один прикоад блок схеми по додаванню запису настрою (рис. 19).

Рис. 19 Запис настрою користувача

Частина коду:

```
const mongoose = require('mongoose');

// Схема для запису настрою
const moodDiarySchema = new mongoose.Schema({
  ID_user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  ID_mood_type: { type: mongoose.Schema.Types.ObjectId, ref: 'MoodType', required: true },
  comment_of_mood: { type: String, required: false },
  date_of_recording_mood: { type: Date, default: Date.now }
});

const MoodDiary = mongoose.model('MoodDiary', moodDiarySchema);

// Функція для додавання запису настрою
async function addMoodRecord(userId, moodTypeId, comment) {
  // Перевірка, чи є вже запис настрою на поточний день для цього користувача
  const existingRecord = await MoodDiary.findOne({
    ID_user: userId,
    date_of_recording_mood: { $gte: new Date(new Date().setHours(0, 0, 0, 0)) }
```

Рис. 19 Додавання настрою

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Процес тестування є критично важливим етапом розробки програмних продуктів, що забезпечує їхню стабільність та відповідність очікуваним стандартам якості. Існує різноманіття підходів до перевірки програмних рішень, кожен з яких має свої специфічні цілі та завдання.

Тестування програмного забезпечення класифікується за різними ознаками, кожна з яких відображає певний аспект процесу перевірки якості продукту [25].

Серед основних видів тестування можна виділити:

1) Функціональне тестування: Перевірка того, чи працює програмне забезпечення відповідно до вимог та функціональних специфікацій. Це тестування основних функцій додатку, таких як реєстрація користувача, виконання операцій чи взаємодія з іншими частинами системи.

2) Нефункціональне тестування: Оцінка якості програмного забезпечення в аспектах, не пов'язаних безпосередньо з функціональністю, таких як продуктивність, безпека, юзабіліті, сумісність та ін. Прикладом є тестування продуктивності або тестування навантаження.

3) Тестування продуктивності: Вивчення швидкості та ефективності роботи програмного забезпечення, особливо за умов високих навантажень. Це включає тестування часу відгуку, пропускної здатності та ресурсної ефективності.

4) Тестування безпеки: Перевірка захищеності програмного забезпечення від можливих загроз, таких як несанкціонований доступ, злом, шкідливі програми, витік інформації та інші уразливості.

5) Тестування сумісності: Оцінка сумісності програмного забезпечення з різними операційними системами, браузерами, пристроями та іншими програмами. Це важливо для забезпечення стабільної роботи додатку на різних платформах.

6) Тестування користувацького інтерфейсу (UI testing): Перевірка зручності та функціональності інтерфейсу програми. Це включає перевірку елементів інтерфейсу, таких як кнопки, поля введення, навігаційні елементи, а також перевірку зручності використання додатку.

7) Регресійне тестування: Тестування програмного забезпечення після внесення змін (наприклад, після виправлення помилок чи додавання нових функцій) для перевірки, що нові зміни не вплинули на роботу вже існуючих функцій.

8) Інтеграційне тестування: Перевірка взаємодії між різними модулями або компонентами програмного забезпечення. Це тестування на предмет того, як окремі частини системи працюють разом.

9) Системне тестування: Комплексне тестування всіх аспектів системи як цілого, щоб забезпечити правильну взаємодію всіх її компонентів у реальних умовах.

10) Альфа- та бета-тестування: Альфа-тестування проводиться командою розробників або тестувальників в межах компанії, тоді як бета-тестування проводиться на реальних користувачах поза межами компанії для отримання відгуків і виявлення додаткових помилок до офіційного запуску.

Для тестування програмного забезпечення мобільного додатку було вирішено використати два типи тестування: юніт тестування та тестування користувацького інтрефейсу, для перевірки коректності роботи як програмного коду, так і правильності побудови користувацького інтерфейсу.

Переваги юніт тестування:

1) раннє виявлення дефектів – дозволяє знаходити та усувати помилки на початкових етапах розробки, що значно знижує вартість їх виправлення.

2) Безпека модифікацій – забезпечує можливість внесення змін до коду з мінімальним ризиком порушення існуючої функціональності.

3) Жива документація – тестовий код слугує практичним описом очікуваної поведінки системи, актуалізуючись разом із основним кодом.

4) Архітектурні покращення – стимулює розробників до створення більш модульного та структурованого коду з чітко визначеними залежностями [26].

Переваги тестування інтерфейсу користувача:

1) валідація візуальної складової – дозволяє виявляти проблеми візуального представлення даних, помилки верстки та некоректну поведінку інтерактивних елементів.

2) оптимізація взаємодії – допомагає ідентифікувати проблемні місця в UX-дизайні, такі як нелогічна навігація чи неінтуїтивні інтерфейсні рішення.

3) забезпечення якості – дає можливість усунути критичні помилки інтерфейсу до релізу, підвищуючи загальну якість продукту та задоволеність кінцевих користувачів.

На рис. 20 представлені юніт тести для методів авторизації користувача.

Цей юніт-тест призначений для перевірки правильності виконання методу authenticateUser. Його реалізацію можна умовно поділити на три основні етапи:

1) підготовка (arrange) – здійснюється налаштування вхідних даних, зокрема визначаються значення змінних username та password, які слугують валідними обліковими даними;

2) виклик (act) – на цьому кроці відбувається безпосередній виклик методу, що тестується, із передачею підготовлених параметрів, а також фіксація отриманого результату;

3) перевірка (assert) – фінальний етап, де проводиться зіставлення отриманого результату з очікуваним, що дозволяє переконатися в успішній авторизації користувача.

```
const bcrypt = require('bcrypt');
const { User } = require('./models'); // Імпортуємо модель користувача з Sequelize

async function loginUser(email, password) {
  try {
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return { success: false, message: "Користувач не знайдений" };
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      return { success: false, message: "Невірний пароль" };
    }

    return { success: true, message: "Авторизація успішна", user };
  } catch (error) {
    console.error(error);
    return { success: false, message: "Помилка при авторизації" };
  }
```

Рис. 20 Юніт тест реєстрації юзера

Для тестування користувацького інтерфейсу також було вирішено вибрати компонент авторизації. На рис. 21 зображено вікно входу в систему [27].

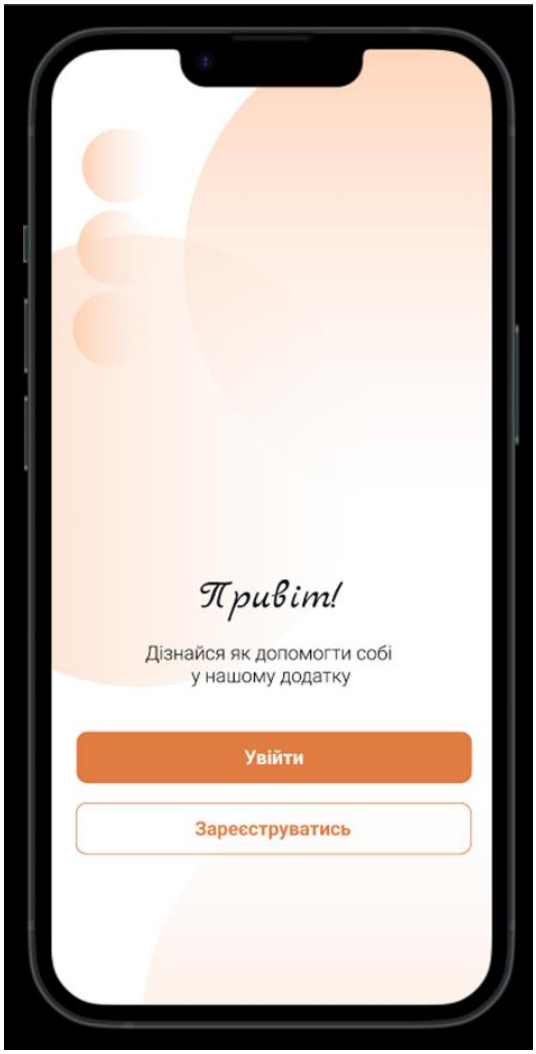


Рис. 21 Вікно входу до системи

При натисканні «Увійти», має відкриватись вікно авторизації (рис. 22).

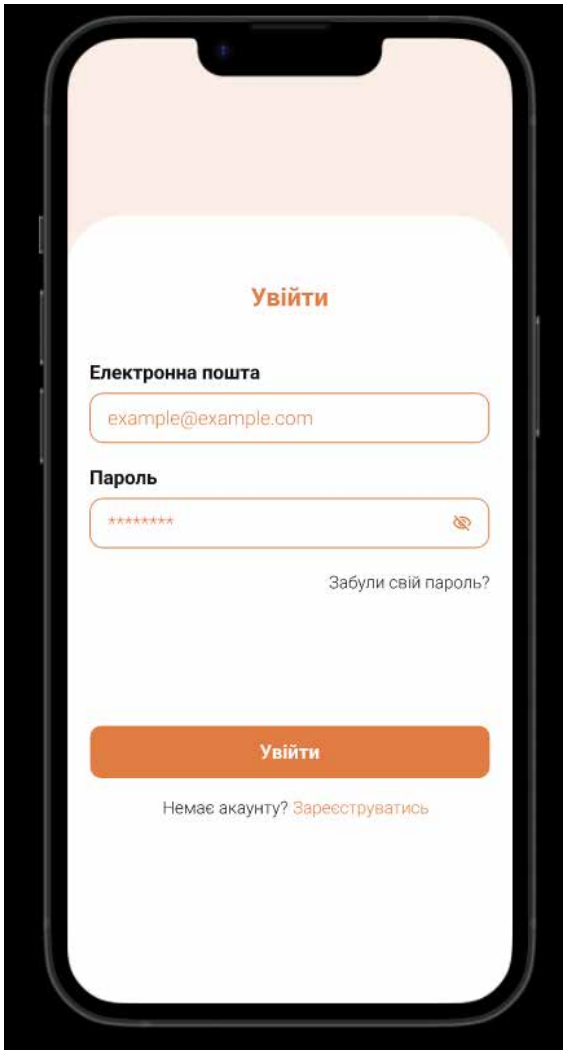


Рис. 22 Вікно авторизації

При введенні даних та натисканні на кнопку Login має відкриватись домашня сторінка користувача (рис. 23).

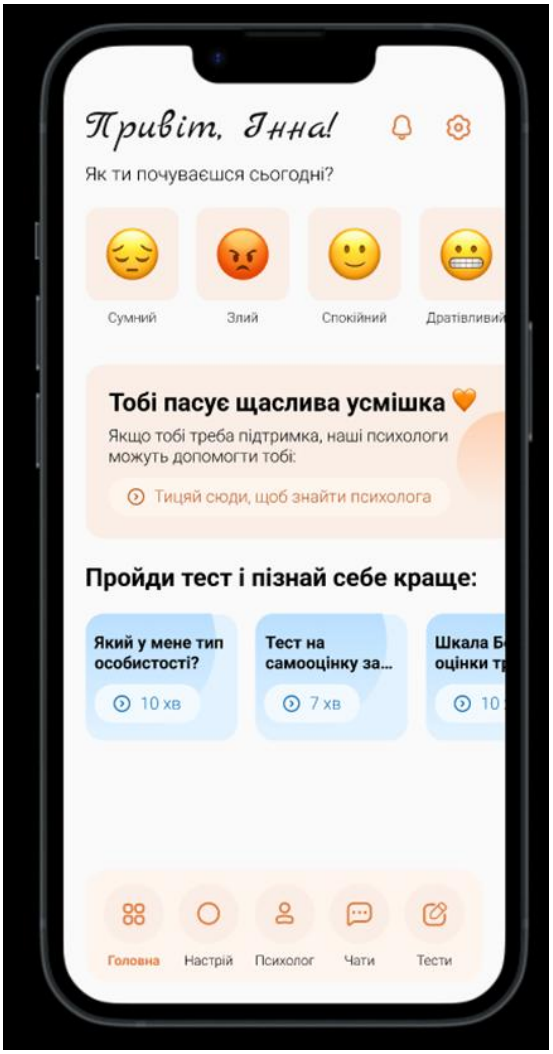


Рис. 23 Вікно головної сторінки

При введенні неправильних чи некоректних даних користувач має отримувати повідомлення про помилку (рис. 24).

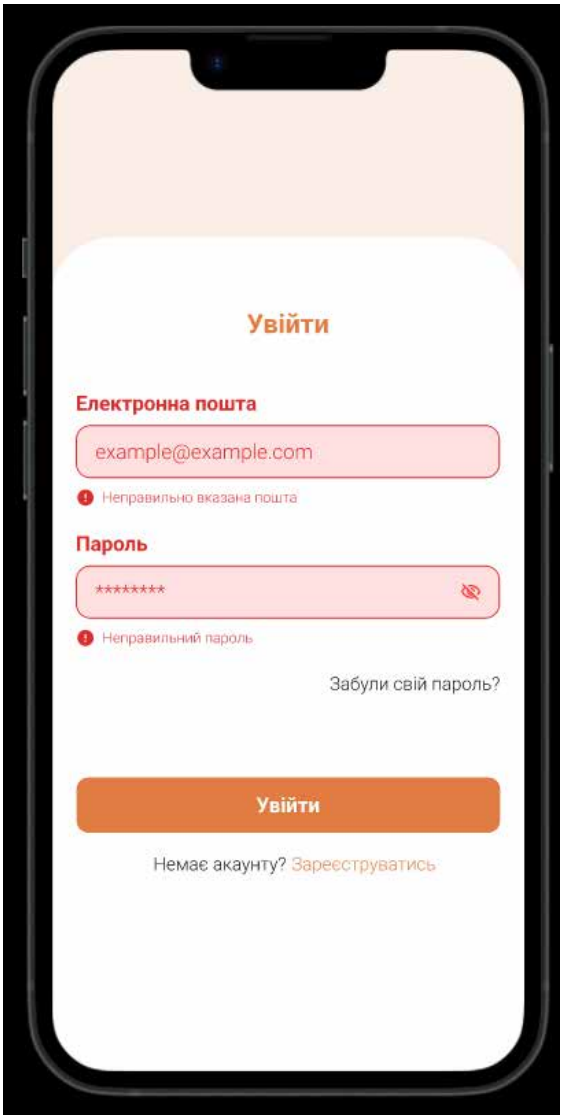


Рис. 24 Вікно авторизації при неправильно введених даних

При перевірці авторизації двома видами тестування не було виявлено ніяких помилок, даний компонент системи працює коректно.

4.2 Вимоги до апаратного та програмного забезпечення

Для програмного забезпечення мобільного додатку є такі мінімальні вимоги:

Таблиця 2

Апаратні вимоги

Мобільний пристрій (iOS)	
Операційна система	iOS 12.0 або новіша
Процесор	A10 Fusion або новіший
Оперативна пам'ять	мінімум 2 ГБ
Вільна пам'ять	мінімум 100 МБ
Сервер	
Процесор	2 vCPU
Оперативна пам'ять	4 GB RAM
Місце на диску	50 GB SSD
Операційна система	Ubuntu 20.04 LTS або новіша

Таблиця 3

Програмні вимоги

Мобільний пристрій (iOS)	
Операційна система	iOS 12.0 або новіша
Сервер	
Node.js	14 або новіша версія
Express	для API
Sequelize	для взаємодії з MySQL
MySQL	8 або новіша версія для зберігання даних

4.3 Склад інсталяційного пакету

В ході проєктування інсталяційного пакету та фізичної організації програмного забезпечення було спроєктовано діаграму розміщення. Діаграма розміщення відображає розподіл усіх компонентів системи на апаратні ресурси, показуючи, як програмні модулі, сервери, бази даних та інші частини системи взаємодіють між собою. Вона дає змогу чітко уявити, як система буде розгорнута на конкретному обладнанні або в хмарному середовищі, що сприяє ефективному плануванню процесу впровадження та налаштування програмного забезпечення. Це допомагає визначити оптимальну топологію системи для забезпечення її стабільної та масштабованої роботи [28].

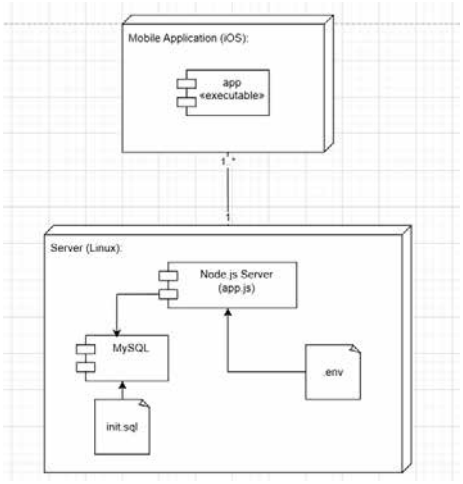


Рис. 25 Діаграма розміщення

На рис. 25 представлена діаграма розгортання системи, яка демонструє архітектуру та взаємодію між фізичними вузлами системи. Ця діаграма складається з двох основних компонентів: серверу та комп'ютерів, що виконують роль клієнтів. Взаємодія між цими вузлами здійснюється через протокол TCP/IP, що забезпечує комунікацію між клієнтами та сервером.

Інсталяційний пакет системи включатиме декілька важливих файлів. Серед них: файл налаштування сервера, який забезпечує зв'язок між клієнтами та

сервером, а також файли, що містять запити для занесення інформації в базу даних, і тести для перевірки функціональності системи.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено мобільний додаток з інтерактивною психологічною підтримкою, орієнтований на допомогу користувачам, що переживають стрес, тривогу та інші емоційні труднощі. Метою розробки було створення доступного та ефективного інструменту, який надає користувачам можливість контролювати свій емоційний стан за допомогою трекера настрою, проходити психологічні тести та отримувати консультації від психологів.

У ході роботи було проведено системний аналіз предметної області, вибрано інструменти для розробки та спроектовано архітектуру програмного забезпечення. Зокрема, були розроблені діаграми для моделювання системи (ER-діаграми, діаграми діяльності, діаграми послідовності), створено базу даних і визначено структуру таблиць для зберігання інформації. Вибір технологій, таких як MySQL для бази даних і Swift для розробки мобільного додатку, забезпечив високу продуктивність і безпеку системи.

Результатом роботи є готовий продукт, який включає мобільний додаток, що реалізує функції трекера настрою, психологічних тестів та безкоштовних консультацій через чат з психологами. Цей додаток забезпечує доступність психологічної допомоги, що є надзвичайно важливим у сучасних умовах, зокрема в період військових та соціальних криз.

Отже, розроблений мобільний додаток може стати важливим інструментом для підтримки ментального здоров'я, надаючи користувачам зручний та доступний спосіб отримати психологічну допомогу та відстежувати емоційний стан.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) В Україні презентували результати першого з початку повномасштабної війни дослідження психологічного стану населення. URL: https://moz.gov.ua/uk/v-ukraini-prezentuvali-rezultati-pershogo-z-pochatku-povnomasshtabnoi-vijni-doslidzhennja-psihologichnogo-stanu-naselennja--
2) Знайдіть психотерапевта, який вас зрозуміє. URL: https://www.rozmova.me/
3) Krisenchat Ukrainian. URL: https://krisenchat.de/uk/ukraine
4) Meaningful insights. Backed by science. URL: https://www.apple.com/health/
5) Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: https://dou.ua/forums/topic/40575/
6) UML для бізнес-моделювання: для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html
7) Діаграма прецедентів. URL: https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2
8) Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.html
9) Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/
10) З Модель діаграми зв’язків сутностей (ER) із прикладом СУБД. URL: https://www.erwin.com/products/erwin-data-modeler/
11) erwin Data Modeler. URL: https://www.erwin.com/products/erwin-data-modeler/

12) Реляційні бази даних усе, що необхідно про них знати. URL: https://foxminded.ua/reliatsiini-bazy-danykh/
13) Типи баз даних: особливості, відмінності та приклади. URL: https://dou.ua/lenta/articles/types-of-databases/
14) MySQL. URL: https://www.mysql.com/
15) What is SQL (Structured Query Language)?. URL: https://aws.amazon.com/what-is/sql/#:~:text=Structured%20query%20language%20(SQL)%20is%20a%20standard%20language%20for%20database,undergoes%20continual%20upgrades%20and%20improvements
16) Створення таблиці та додавання полів. URL: https://support.microsoft.com/uk-ua/topic/
17) Збережені процедури в SQL. URL: https://acode.com.ua/stored-procedures-sql/
18) SQL Triggers. URL: https://www.geeksforgeeks.org/sql-trigger-student-database/
19) Introduction to Model View View Model (MVVM). URL: https://online.visual-paradigm.com/ru/diagrams/templates/package-diagram/package-diagram-overview/
20) Знайдіть психотерапевта, який вас зрозуміє. URL: https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/
21) Release Notes. URL: https://www.figma.com/release-notes/
22) Swift. URL: https://www.swift.com/
23) Xcode. URL: https://developer.apple.com/xcode/
24) Run JavaScript Everywhere. URL: https://nodejs.org/en
25) Run JavaScript Everywhere. URL: https://nodejs.org/en

26) What Is Unit Testing? URL: https://smartbear.com/learn/automated-testing/what-is-unit-testing/

27) UI тестування: що це таке і навіщо воно потрібне. URL: https://megasite.ua/ua/ui-testirovanie-chto-eto-takoe-i-zachem-ono-nugno

28) Діаграма розгортання: Підручник з UML із ПРИКЛАДОМ. URL: https://www.guru99.com/uk/deployment-diagram-uml-example.html

ДОДАТОК А

ОПУБЛІКОВАНІ ТЕЗИ НА V ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ ІНТЕРНЕТ КОНФЕРЕНЦІЇ СТУДЕНТІВ І АСПІРАНТІВ «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ 2025» (24 КВІТНЯ 2025 РОКУ)

Сторінок – 3

2025

УДК 004.4:378.147

МОБІЛЬНИЙ ДОДАТОК З ІНТЕРАКТИВНОЮ ПСИХОЛОГІЧНОЮ ПІДТРИМКОЮ

Дрозд І.А., *студентка ОКР «Бакалавр», 4 курс,*
науковий керівник Панкратьєв В.О.
Національний університет біоресурсів і природокористування України

Вступ. У сучасному світі, де питання ментального здоров'я набувають все більшої ваги, особливо важливим є створення доступних і ефективних інструментів для підтримки людей у складних життєвих ситуаціях. В умовах війни в Україні, коли стрес і психологічне виснаження стають невід'ємною частиною життя, мобільні технології можуть стати важливим інструментом для надання допомоги. Мобільний додаток, що забезпечує інтерактивну психологічну підтримку, дозволяє користувачам не тільки слідкувати за своїм настроєм, а й отримувати психологічну допомогу безкоштовно. Це також є важливою можливістю для психологів надавати підтримку, здобуваючи досвід у роботі з людьми.

Актуальність. Сучасні психологічні проблеми, такі як стрес, тривога, депресія, особливо загострилися в умовах війни в Україні. Втрата близьких, стрес через постійну невизначеність і небезпеку ставлять громадян у складні емоційні умови, що вимагають негайної підтримки. Однак доступ до традиційних психологічних консультацій залишається обмеженим через фінансові та логістичні труднощі. Мобільний додаток з інтерактивною психологічною підтримкою дозволяє забезпечити безкоштовну, оперативну допомогу, доступну для всіх бажаючих у будь-який час. Також важливою особливістю є можливість надання консультацій не тільки досвідченими психологами, а й молодими фахівцями, які можуть проходити практику на платформі.

Мета дослідження. Розробка мобільного додатка, який дозволяє забезпечити інтерактивну психологічну підтримку користувачам через функції трекера настрою, психологічних тестів та безкоштовних консультацій з психологами.

Основна ідея. Основна ідея додатку полягає в наданні безкоштовної психологічної допомоги через зручний мобільний інтерфейс. Користувачі можуть слідкувати за своїм емоційним станом, проходити тести для оцінки психологічного стану та отримувати консультації через чат з психологами. Додаток працює на платформі iOS, розроблений за допомогою Swift, що забезпечує високу продуктивність та зручність користування.

Реалізація структури бази даних у вигляді ER діаграми зображено на рис 1.

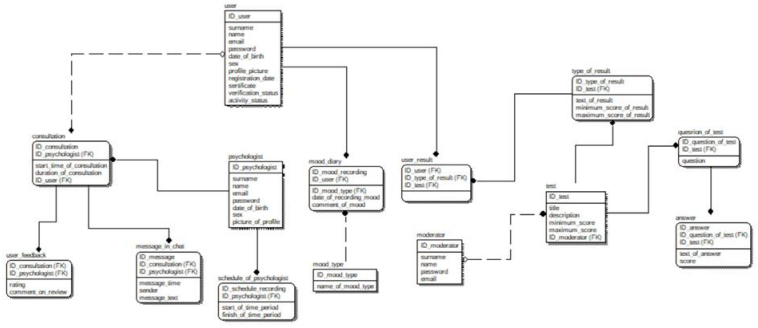


Рис. 1 ER діаграма

Завдання та функції системи. Розробка мобільного додатка для інтерактивної психологічної підтримки включає низку завдань і функцій, які забезпечують ефективне управління психологічною підтримкою користувачів. Основні завдання системи включають:

1. Трекер настрою – користувачі можуть вибирати свій настрій за допомогою емодзі, додавати коментарі та переглядати статистику зміни настроїв.
2. Психологічні тести – створення та проведення тестів для визначення типу особистості, рівня стресу та інших психоемоційних станів.
3. Консультації з психологами – можливість отримати безкоштовну консультацію через чат з сертифікованими психологами.
4. Інтеграція з сервером – взаємодія з сервером через REST API для забезпечення обміну даними між мобільним клієнтом і серверною частиною.
5. Інтерфейс користувача – простота та інтуїтивність взаємодії з додатком, підтримка кількох мов, адаптивність до різних екранів пристроїв.

Висновки. У результаті дослідження був розроблений мобільний додаток, що забезпечує інтерактивну психологічну підтримку користувачів. Архітектура додатку включає використання сучасних технологій для реалізації зручного і безпечного сервісу, який дозволяє користувачам слідкувати за своїм емоційним станом, проходити тести, отримувати консультації психологів і доступати до корисних матеріалів. Це рішення дозволяє значно покращити доступність психологічної підтримки, зменшити стрес і підвищити рівень ментального здоров’я серед користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft Docs. SQL Server Management Studio (SSMS). – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/sql/ssms
2. UML 2.5.1 Specification – Object Management Group. – [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/UML
3. Макаренко М.О. Проектування інформаційних систем: навчальний посібник. – К.: КНЕУ, 2020. – 256 с.
4. Концепція розвитку цифрових компетентностей: Розпорядження КМУ від 03.03.2021 № 167-р. – [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/167-2021-р

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДИПЛОМ
ІІ ступеню
НАГОРОДЖУЄТЬСЯ

Дрозд Інна Анатоліївна

студентка групи КН-21003б ОС «Бакалавр»

за доповідь «Мобільний додаток з інтерактивною психологічною підтримкою»
на VII Всеукраїнській науково-практичній конференції студентів і аспірантів
«Теоретичні та прикладні аспекти розробки комп'ютерних систем '2025»

Завідувач кафедри комп'ютерних наук НУБіП України **Б.Л. Голуб**

24 квітня 2025 р.

ДОДАТОК Б

SQL ЗАПИТИ

Сторінок – 11

2025

```
use helpdb;



CREATE TABLE `user`
(
    `ID_user`           int(11)                          NOT NULL,
    `surname`           varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `name`              varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `email`             varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `password`          varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `date_of_birth`     date                             DEFAULT NULL,
    `sex`               varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `profile_picture`   blob,
    `registration_date` date                             NOT NULL,
    `sertificate`       blob                            NOT NULL,
    `verification_status` bit(1)                         NOT NULL,
    `activity_status`   bit(1)                           NOT NULL,
    PRIMARY KEY (`ID_user`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
```

```
COLLATE = utf8mb4_unicode_ci;


CREATE TABLE `psychologist`
(
   `ID_psychologist`    int(11)                          NOT NULL,
   `surname`           varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `name`              varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `email`             varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `password`          varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `date_of_birth`      date                             DEFAULT NULL,
   `sex`               varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
   `picture_of_profile` blob,
   PRIMARY KEY (`ID_psychologist`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;


CREATE TABLE `mood_type`
(
```

```
 `name_of_mood_type` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `ID_mood_type`     int(11)                          NOT NULL,
   PRIMARY KEY (`ID_mood_type`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `moderator`
(
   `ID_moderator` int(11)                          NOT NULL,
   `surname`      varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `name`         varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `password`     varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `email`        varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   PRIMARY KEY (`ID_moderator`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `test`
```

```
(
    `ID_test`       int(11)                         NOT NULL,
    `title`         varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `description`   varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `minimum_score` int(11)                         NOT NULL,
    `maximum_score` int(11)                         NOT NULL,
    `ID_moderator`  int(11) DEFAULT NULL,
    PRIMARY KEY (`ID_test`),
    KEY `R_24` (`ID_moderator`),
    CONSTRAINT `test_ibfk_1` FOREIGN KEY (`ID_moderator`) REFERENCES
`moderator` (`ID_moderator`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;


CREATE TABLE `consultation`
(
    `ID_consultation`           int(11)  NOT NULL,
    `start_time_of_consultation` datetime NOT NULL,
    `duration_of_consultation`   datetime NOT NULL5
```

```
 `ID_psychologist`          int(11)  NOT NULL,
   `ID_user`                int(11)  NOT NULL,
   PRIMARY KEY (`ID_consultation`, `ID_psychologist`),
   KEY `R_11` (`ID_psychologist`),
   KEY `R_26` (`ID_user`),
   CONSTRAINT `consultation_ibfk_1` FOREIGN KEY (`ID_psychologist`)
REFERENCES `psychologist` (`ID_psychologist`),
   CONSTRAINT `consultation_ibfk_2` FOREIGN KEY (`ID_user`) REFERENCES
`user` (`ID_user`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `user_feedback`
(
   `rating`          int(11) NOT NULL,
   `comment_on_review` varchar(500) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
   `ID_consultation`   int(11) NOT NULL,
   `ID_psychologist`   int(11) NOT NULL,
```

```
PRIMARY KEY (`ID_consultation`, `ID_psychologist`),
   CONSTRAINT `user_feedback_ibfk_1` FOREIGN KEY (`ID_consultation`,
`ID_psychologist`) REFERENCES `consultation` (`ID_consultation`,
`ID_psychologist`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `message_in_chat`
(
   `ID_message`      int(11)                          NOT NULL,
   `message_time`    datetime                         NOT NULL,
   `sender`          bit(1)                           NOT NULL,
   `message_text`    varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `ID_consultation` int(11)                          NOT NULL,
   `ID_psychologist` int(11)                          NOT NULL,
   PRIMARY KEY (`ID_message`, `ID_consultation`, `ID_psychologist`),
   KEY `R_13` (`ID_consultation`, `ID_psychologist`),
   CONSTRAINT `message_in_chat_ibfk_1` FOREIGN KEY (`ID_consultation`,
`ID_psychologist`) REFERENCES `consultation` (`ID_consultation`,
`ID_psychologist`)
```

```
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `schedule_of_psychologist`
(
    `start_of_time_period`  datetime NOT NULL,
    `finish_of_time_period` datetime NOT NULL,
    `ID_psychologist`       int(11)  NOT NULL,
    `ID_schedule_recording` int(11)  NOT NULL,
    PRIMARY KEY (`ID_schedule_recording`, `ID_psychologist`),
    KEY `R_7` (`ID_psychologist`),
    CONSTRAINT `schedule_of_psychologist_ibfk_1` FOREIGN KEY
(`ID_psychologist`) REFERENCES `psychologist` (`ID_psychologist`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `mood_diary`
(
```

```
 `date_of_recording_mood` date    NOT NULL,
   `comment_of_mood`        varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
   `ID_user`                int(11) NOT NULL,
   `ID_mood_type`           int(11) NOT NULL,
   `ID_mood_recording`      int(11) NOT NULL,
   PRIMARY KEY (`ID_mood_recording`, `ID_user`),
   KEY `R_14` (`ID_user`),
   KEY `R_15` (`ID_mood_type`),
   CONSTRAINT `mood_diary_ibfk_1` FOREIGN KEY (`ID_user`) REFERENCES `user` (`ID_user`),
   CONSTRAINT `mood_diary_ibfk_2` FOREIGN KEY (`ID_mood_type`) REFERENCES `mood_type` (`ID_mood_type`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;


CREATE TABLE `type_of_result`
(
   `ID_type_of_result`       int(11)                         NOT NULL,
```

```
 `text_of_result`          varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `minimum_score_of_result` int(11)                              NOT NULL,
   `maximum_score_of_result` int(11)                              NOT NULL,
   `ID_test`                 int(11)                              NOT NULL,
   PRIMARY KEY (`ID_type_of_result`, `ID_test`),
   KEY `R_18` (`ID_test`),
   CONSTRAINT `type_of_result_ibfk_1` FOREIGN KEY (`ID_test`) REFERENCES
`test` (`ID_test`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;


CREATE TABLE `user_result`
(
   `ID_type_of_result` int(11) NOT NULL,
   `ID_user`           int(11) NOT NULL,
   `ID_test`           int(11) NOT NULL,
   PRIMARY KEY (`ID_user`, `ID_type_of_result`, `ID_test`),
   KEY `R_17` (`ID_type_of_result`, `ID_test`),
```

```
CONSTRAINT `user_result_ibfk_1` FOREIGN KEY (`ID_type_of_result`, `ID_test`)
REFERENCES `type_of_result` (`ID_type_of_result`, `ID_test`),
   CONSTRAINT `user_result_ibfk_2` FOREIGN KEY (`ID_user`) REFERENCES
`user` (`ID_user`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;

CREATE TABLE `quesrion_of_test`
(
   `ID_question_of_test` int(11)                    NOT NULL,
   `question`          varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `ID_test`           int(11)                      NOT NULL,
   PRIMARY KEY (`ID_question_of_test`, `ID_test`),
   KEY `R_19` (`ID_test`),
   CONSTRAINT `quesrion_of_test_ibfk_1` FOREIGN KEY (`ID_test`)
REFERENCES `test` (`ID_test`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;
```

```
CREATE TABLE `answer`
(
   `ID_answer`          int(11)                          NOT NULL,
   `text_of_answer`     varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
   `score`              int(11)                          NOT NULL,
   `ID_question_of_test` int(11)                         NOT NULL,
   `ID_test`            int(11)                          NOT NULL,
   PRIMARY KEY (`ID_answer`, `ID_question_of_test`, `ID_test`),
   KEY `R_20` (`ID_question_of_test`, `ID_test`),
   CONSTRAINT `answer_ibfk_1` FOREIGN KEY (`ID_question_of_test`, `ID_test`)
REFERENCES `quesrion_of_test` (`ID_question_of_test`, `ID_test`)
) ENGINE = InnoDB
  DEFAULT CHARSET = utf8mb4
  COLLATE = utf8mb4_unicode_ci;
```

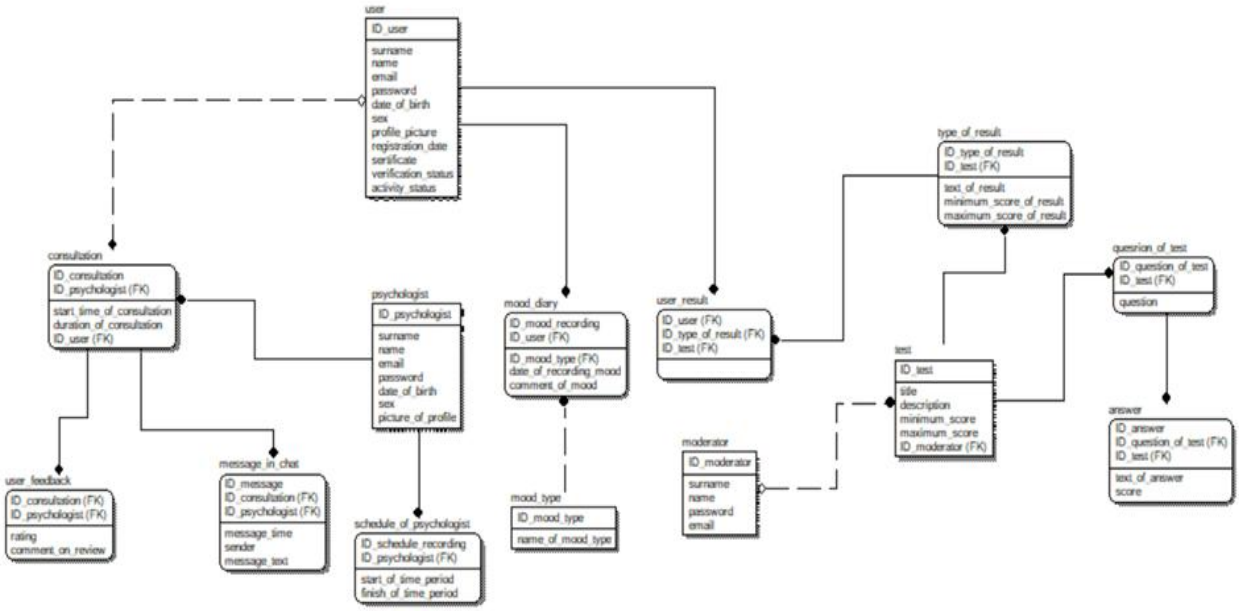


Рис. 1 ER-діаграма