Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА ВИБОРУ ПАРТНЕРІВ У БАГАТОКОРИСТУВАЦЬКИХ ВІДЕОІГРАХ»

Виконав: студент 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Валентин КОЛЕСНИК
Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«04» 06 2025 р.
Рецензент: к.т.н., доц. каф. КН
_____ Олексій СІЛАГІН
«08» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"28" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Колеснику Валентину Івановичу

1.Тема роботи: «Інформаційно-пошукова система вибору партнерів у багатокористувацьких відеоіграх»
керівник роботи: Козачко О. М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «10» 03 2025 року № 97
2. Термін подання студентом роботи 31.05.25
3. Вихідні дані до роботи:
1) Дані про профілі користувачів багатокористувацьких відеоігор;
2) Дані про ігри та їх специфіку;
4. Зміст текстової частини:
1) Аналіз предметної області та вимоги до системи;
2) Проєктування архітектури та функціоналу інформаційно-пошукової системи;
3) Реалізація та тестування інформаційно-пошукової системи;
5. Перелік ілюстративного матеріалу:
1) Діаграми (Архітектура інформаційно-пошукової системи, розгортання системи, "сутність-зв'язок" бази даних, USE CASE інформаційно-пошукової системи);
2) Блок-схеми (Алгоритму підбору та фільтрації користувачів, функціонування інформаційно-пошукової системи, алгоритм роботи інформаційно-пошукової системи);
3) Зображення інтерфейсу інформаційно-пошукової системи.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.03.25

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	10.04	вик
2	Порівняльний аналіз проблематики	10.04	20.04	вик
3	Вибір оптимальних інформаційних технологій	20.04	30.04	вик
4	Розроблення інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх	1.05	15.05	вик
5	Оформлення матеріалів до захисту БКР	15.05	30.05	вик

Студент _____ Валентин КОЛЕСНИК

Керівник роботи _____ Олексій КОЗАЧКО

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 88 сторінок формату А4, на яких є 35 рисунків, список використаних джерел містить 27 найменувань.

Метою роботи є розроблення та дослідження інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх на базі мобільного додатку.

В роботі наведено розробку інформаційно-пошукової системи на кросплатформній основі з використанням фреймворків React Native та Expo. Розроблено архітектуру, що забезпечує автономну роботу з локальною базою даних SQLite та безпечну аутентифікацію за допомогою Firebase.

Реалізовано дизайн та функціонал пошуку та фільтрації ігрових партнерів за різноманітними критеріями, а також управління профілями користувачів. Розроблена інформаційно-пошукова система дає можливість автоматизувати процес знаходження сумісних гравців у відеоіграх.

Ключові слова: інформаційно-пошукова система, відеоігри, партнери по грі, React Native, Expo, Firebase, SQLite, пошук, фільтрація, багатокористувацькі ігри, мобільний застосунок, дизайн.

ABSTRACT

The bachelor's thesis consists of 88 pages of A4 format, with 35 figures, and a list of references containing 27 titles.

The aim of the work is to develop and study an information retrieval system for selecting partners in multiplayer video games based on a mobile application.

The paper presents the development of a cross-platform information retrieval system using the React Native and Expo frameworks. An architecture has been developed that provides offline operation with a local SQLite database and secure authentication using Firebase.

The design and functionality of searching and filtering gaming partners by various criteria, as well as managing user profiles were implemented. The developed information retrieval system makes it possible to automate the process of finding compatible players in video games.

Keywords: information retrieval system, video games, game partners, React Native, Expo, Firebase, SQLite, search, filtering, multiplayer games, mobile application, design.

ЗМІСТ

ВСТУП

Актуальність теми. Проблема ефективного пошуку партнерів у багатокористувацьких відеоіграх є вкрай актуальною у сучасному ігровому суспільстві. Зростання конкуренції та складності ігрових механік вимагають від гравців не лише індивідуальних навичок, а й уміння злагоджено працювати в команді. Існуючі механізми пошуку гравців, які вбудовані в ігри, часто є недостатньо гнучкими та не враховують всі аспекти сумісності, такі як стиль гри, досвідченість, ігрові цілі, мови спілкування та особистісні характеристики. Це призводить до розчарувань, зниження мотивації та в кінцевому підсумку до думки перестати грати в ту чи іншу гру для користувачів. Розроблення спеціалізованих інформаційно-пошукових систем, спрямованих на більш точний та персоналізований підбір, є необхідною для покращення користувацького досвіду (UX) та сприяння формуванню ігрових спільнот. Особливу актуальність ця тема набуває для України, де ігрова індустрія активно розвивається, а попит на якісні та ефективні рішення для геймерів постійно зростає.

Мета і завдання роботи. Метою роботи є розроблення та дослідження інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх на базі мобільного додатку. Ця система покликана забезпечити ефективний підбір гравців відповідно до заданих властивостей та за будь-якою грою. Якщо у користувача є більше однієї улюбленої гри, в яку він постійно грає, інформаційно-пошукова система надасть можливість знайти собі подібного партнера з такими самими відеоіграми та з однаковими характеристиками. За допомогою такої наданої можливості Інформаційно-пошукова система вибору партнерів у багатокористувацьких відеоіграх стає набагато кориснішою за пошукову систему всередині відеоігор.

Для досягнення поставленої мети визначено такі завдання:

– Провести аналіз сучасного стану проблеми пошуку партнерів у багатокористувацьких відеоіграх, включаючи огляд внутрішньоігрових

механізмів та сторонніх платформ, з метою виявлення їхніх функціональних та якісних недоліків.

- Обґрунтувати актуальність розробки нової інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх та сформулювати детальні функціональні та нефункціональні вимоги до неї.
- Вибрати та обґрунтувати технологічний стек для розробки мобільного додатку та спроектувати архітектуру програмного забезпечення.
- Розробити логічну структуру бази даних, алгоритми підбору та фільтрації партнерів на основі обраних критеріїв.
- Реалізувати ключові модулі інформаційно-пошукової системи, включаючи функціонал аутентифікації, управління профілем, пошуку та збереження партнерів.
- Провести тестування розробленої системи та оцінити її ефективність, включаючи порівняльний аналіз з існуючими аналогами.

Об'єкт досліджень. Об'єктом досліджень у роботі є процес пошуку та ефективної взаємодії гравців у багатокористувацьких відеоіграх.

Предмет досліджень. Предметом досліджень є методи, моделі та засоби розробки інформаційно-пошукової системи для автоматизованого та оптимізованого вибору партнерів у багатокористувацьких відеоіграх.

Публікації. За результатами виконання даної роботи опубліковано тези доповідей на тему: «Мобільний додаток пошуку партнерів у багатокористувацьких відеоіграх» на LIV Всеукраїнській науково-технічній конференцій факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.) [1].

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГИ ДО СИСТЕМИ

1.1 Аналіз сучасного стану проблеми пошуку партнерів у відеоіграх

Досить високий розвиток індустрії відеоігор сприяв тому, що багатокористувацькі онлайн-ігри посіли провідні позиції на ринку, залучаючи до своїх лав мільйони гравців по всьому світу. До найпопулярніших жанрів належать шутери від першої особи, командні онлайн-арени та ігри у форматі “королівської битви”. Вони передбачають тісну взаємодію між учасниками, тому ефективний підбір гравців у команди стає ключовим чинником для досягнення перемоги, отримання задоволення від процесу та загального ігрового досвіду. Геймери прагнуть знайти напарників не лише для покращення своїх результатів, а й для створення нових знайомств та спільного дозвілля, що свідчить про важливу соціальну складову цієї сфери.

У більшості сучасних онлайн-ігор уже інтегровані спеціальні системи, які автоматично підбирають гравців для формування команд. Такі механізми намагаються знайти баланс між рівнем майстерності учасників і швидкістю пошуку матчу, однак не завжди працюють ідеально. В окремих іграх додатково застосовується система LFG (Looking For Group), яка допомагає знайти партнерів для спільної гри [2].

Приклади функціоналу та принципи роботи типових вбудованих систем підбору гравців у популярних багатококористувацьких іграх:

– Система підбору у відеогрі Valorant. Механізм підбору гравців ґрунтується на внутрішньому рейтингу, який формується з урахуванням особистих досягнень, результатів попередніх матчів та поточного рангу користувача (рис. 1.1). Попри намагання створити команди з приблизно однаковим рівнем майстерності, система зазвичай не бере до уваги такі аспекти, як улюблені ролі гравців, мова спілкування чи зручний час для гри. Це часто спричиняє появу команд з невідповідним розподілом ролей і виникнення мовних труднощів або ж так званих мовних бар’єрів між учасниками.

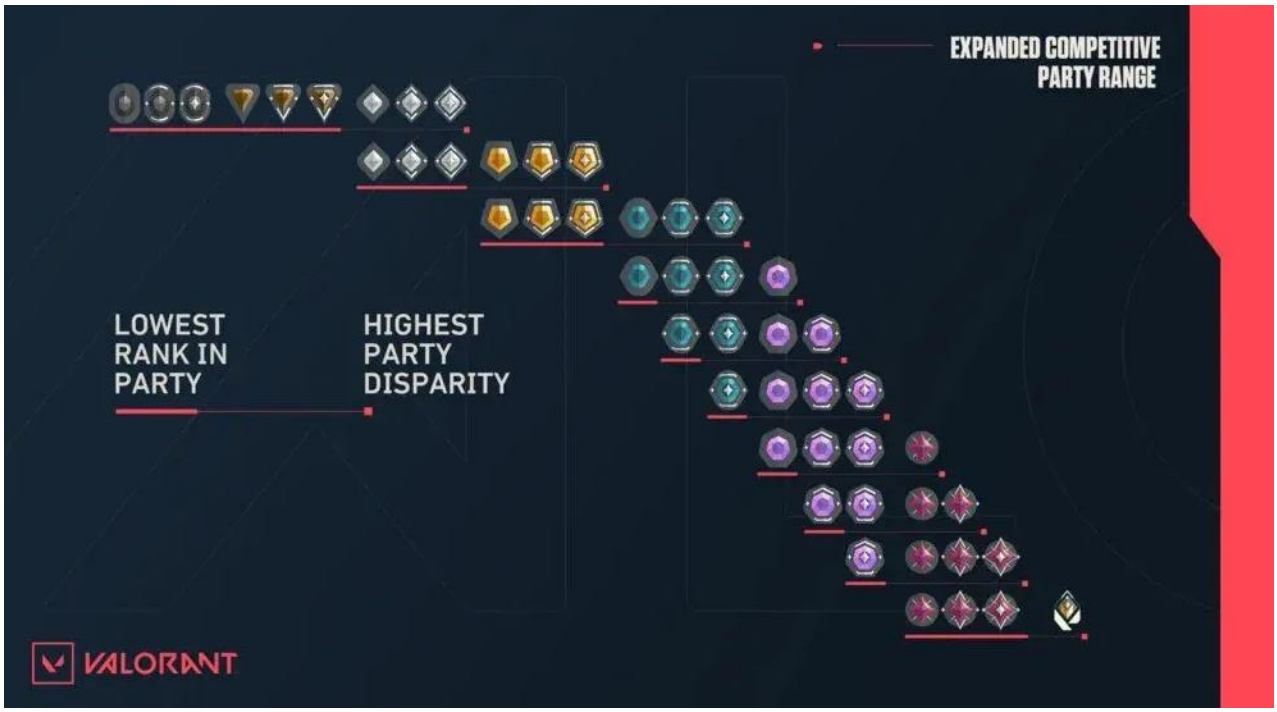

Рисунок 1.1 – Система можливого підбору за рангами у відеогрі Valorant

– Система підбору у відеогрі League of Legends. У грі League of Legends система підбору гравців також базується на рангах, які розділяються на два типи матчів, а саме: нормальні або звичайні та рейтингові. Основним пріоритетом цієї системи є швидкість пошуку гри, що іноді призводить до значних відмінностей у рівнях гравців всередині однієї команди, мовних непорозумінь або ж до ситуацій, коли доводиться грати з учасниками, чиї навички суттєво перевищують власні.
– Система підбору у відеогрі Counter-Strike 2. Система підбору гравців базується на рівні майстерності, репутації та рангу користувача. Хоча вона намагається формувати збалансовані команди, відсутність можливості вибору бажаних ролей або фільтрації за мовою часто призводить до хаотичного складу команд, що негативно позначається на координації та тактичній взаємодії під час матчу.
– Система підбору у відеогрі Dota 2. Система розроблена з акцентом на рольовий підбір, який намагається усунути проблему всіх гравців на одній позиції в одній команді. Однак, навіть з рольовим рангом, гравці часто стикаються з відсутністю належної комунікації через мовні бар'єри або з тим, що стиль гри напарників не відповідає їхнім очікуванням.

– Система LFG (Looking For Group) у відеогрі Overwatch. Ця система працює так, що гравці можуть створити оголошення про пошук групи, вказавши бажані ролі, рівень досвіду, наявність мікрофону та навіть конкретні стратегії (рис. 1.2). Інші гравці можуть переглядати ці оголошення та приєднуватися до групи. Система також може включати рейтинговий матчмейкінг, який автоматично підбирає гравців зі схожим рівнем майстерності.

Рисунок 1.2 – Принцип роботи системи LFG у відеогрі Overwatch

Аналізуючи вбудовані механізми у відеоіграх було виявлено низку суттєвих обмежень, які значно знижують якість ігрового досвіду та ускладнюють пошук справді сумісних партнерів:

– Переважна більшість систем зосереджена лише на кількісних показниках або ж так званих рангах, ігноруючи якісні характеристики гравців.
– Неможливість налаштування фільтрів підбору за індивідуальними потребами гравця.
– Гравці не можуть вказати свій доступний період для гри, що ускладнює знаходження партнерів для регулярних сесій або тренувань разом.

- Системи часто не дозволяють фільтрувати за мовою, що призводить до мовних бар'єрів у багатонаціональних командах.
- Ігрові системи не можуть визначити, чи є гравець агресивним, пасивним, командним чи індивідуалістом, що є критично важливим для створення злагодженої команди.

Існують також сторонні платформи, веб-сайти та мобільні додатки, які розроблені спеціально для того, щоб допомогти гравцям знаходити релевантних партнерів у різних іграх.

Мобільний застосунок GamerLink (рис. 1.3). Це мобільний додаток, що позиціонується як соціальна мережа для геймерів. Дозволяє користувачам створювати профілі, вказувати ігри, в які вони грають, свої уподобання: "конкурентна гра", "фанова гра", "без мікрофона", а також фільтрувати інших гравців за різними критеріями [3].

- Переваги: Підтримка багатьох ігор, можливість створення деталізованого профілю, гнучкі фільтри пошуку та функції спілкування. Націленість саме на пошук партнерів за інтересами.
- Недоліки: Популярність може варіюватися в залежності від регіону та конкретної гри. Не завжди має таку глибоку інтеграцію з ігровою статистикою, як вбудовані рішення. Може потребувати активної користувацької бази для ефективного пошуку.

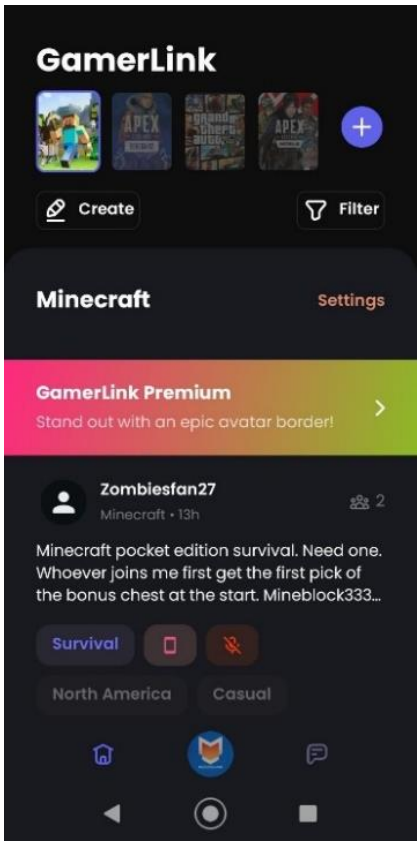

Рисунок 1.3 – Принцип роботи підбору напарника у додатку GamerLink

Веб-платформа LoL Finder (рис. 1.4). Спеціалізована веб-платформа, призначена виключно для гравців доволі популярної багатокористувацької онлайн гри League of Legends. Платформа дозволяє гравцям шукати партнерів для гри, вказуючи свій сервер, ранг, ролі, які вони віддають перевагу, та навіть бажаний настрій такий як: "фанова гра", "серйозна гра". Користувачі можуть переглядати профілі, бачити основну ігрову статистику, через інтеграцію з Riot Games API та спілкуватися з потенційними партнерами по грі [4].

– Переваги: Висока спеціалізація дозволяє глибоко інтегруватися з однією грою та надавати розширені фільтри, специфічні для її механік (ролі, лінії, ранг). Наявність ігрової статистики допомагає оцінити потенційного партнера. Спільнота зосереджена на одній грі, що полегшує пошук.
– Недоліки: Обмеженість лише однією грою League of Legends робить її непридатною для гравців інших тайтлів. Функціонал чату може бути базовим, і для подальшої комунікації гравці часто переходять на інші платформи.

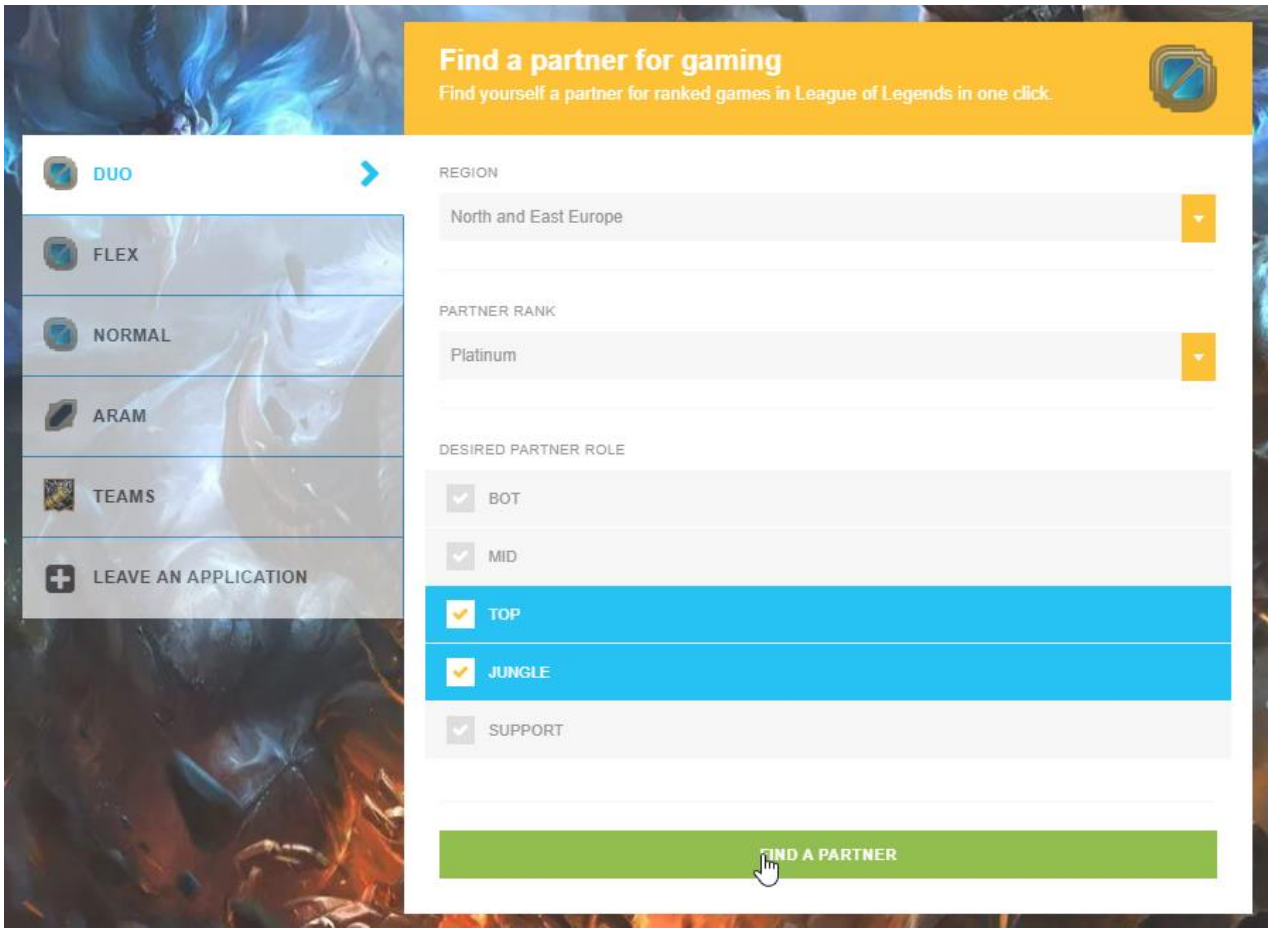

Рисунок 1.4 – Принцип роботи підбору напарника на сайті LoL Finder

Окрім спеціалізованих платформ, гравці активно використовують універсальні комунікаційні сервіси, які, хоч і не призначені виключно для LFG, але стали майже основним інструментом для організації спільної гри.

Платформа Discord. Безкоштовний месенджер та VoIP-додаток, створений для ігрових спільнот. Гравці створюють та приєднуються до серверів, присвячених конкретним іграм, спільнотам або кланам. На серверах існують текстові та голосові канали, багато з яких використовуються саме для пошуку групи, де гравці пишуть свої запити на пошук партнерів [5].

– Переваги: Надзвичайно широка популярність та велика користувацька база. Гнучкість у створенні каналів та ролей, що дозволяє спільнотам самостійно організовувати LFG. Наявність голосового зв'язку.
– Недоліки: Відсутність вбудованої системи фільтрації та персоналізованого підбору за критеріями. Пошук партнерів часто здійснюється вручну, шляхом перегляду повідомлень у чатах. Залежить від добросовісності

адміністрації серверів та самих користувачів у підтримці порядку. Складність пошуку за критеріями вимагає від користувачів додаткових зусиль та часу.

Принцип роботи системи підбору партнерів на платформі Discord, представлено на рисунку 1.5.

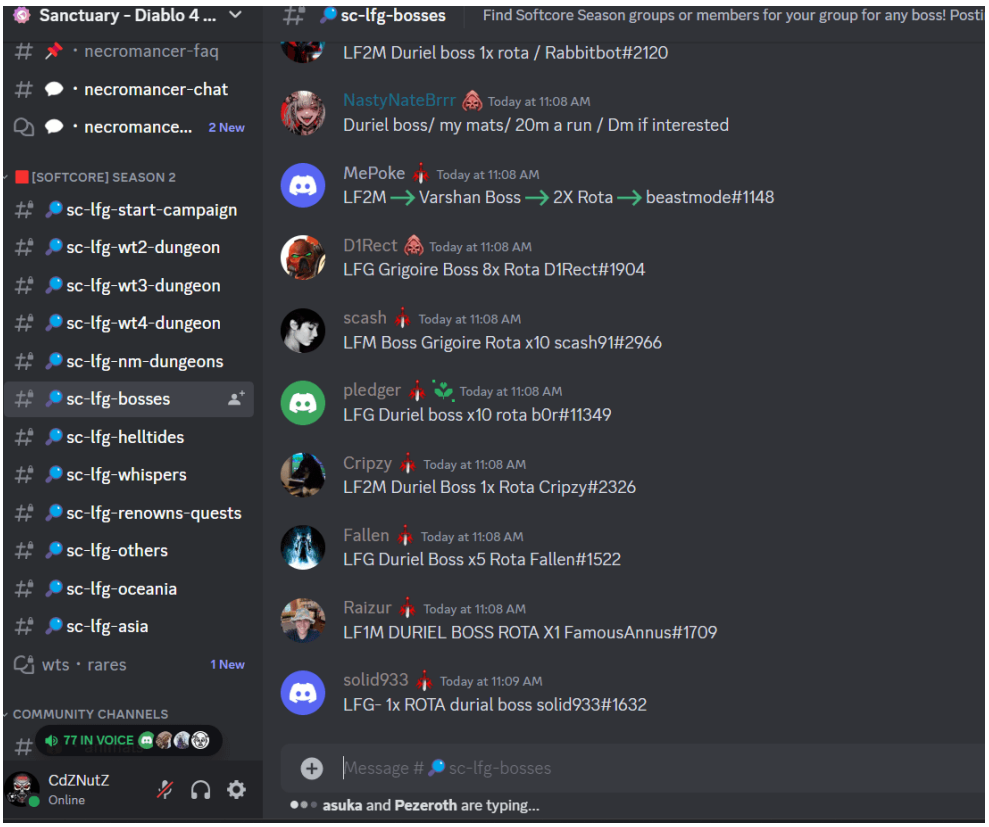

Рисунок 1.5 – Принцип роботи системи LFG у Discord

Аналіз наявних систем підбору гравців демонструє, що кожна з них має як переваги, так і недоліки. Вбудовані механізми в іграх зручні у використанні, проте обмежені рамками конкретної гри та часто недостатньо гнучкі у підборі партнерів. Спеціалізовані LFG-платформи забезпечують більшу свободу налаштувань, але їхня ефективність значною мірою залежить від активності та чисельності користувачів. Універсальні сервіси, такі як Discord, користуються популярністю завдяки розвиненим можливостям комунікації, однак їм бракує складних інструментів автоматизованого пошуку та фільтрації за різноманітними критеріями.

Існує очевидна потреба у створенні рішення, що поєднувало б гнучкість спеціалізованих LFG-платформ із широкою доступністю мобільних додатків, пропонуючи при цьому глибокі можливості персоналізації профілю та ефективні алгоритми пошуку і фільтрації, які охоплюють різні ігри та враховують індивідуальні вподобання користувачів. Саме на подолання цих недоліків спрямоване розроблення нової інформаційно-пошукової системи.

Проведений аналіз також виявив низку ключових проблем, які жодне з існуючих рішень повністю не вирішує:

- Жодна з платформ не надає можливості створити по-справжньому комплексний профіль гравця, що включав би не лише ігрові показники, а й особистісні характеристики, комунікативні вподобання та гнучкий графік доступності.
- Існуючі системи не дозволяють здійснювати пошук за множинними, взаємопов'язаними критеріями, такими як комбінація гри, рангу, бажаної ролі, мови, регіону, періоду гри, стилю гри та особистих тегів.
- Більшість рішень не мають підтримки різних мов спілкування та не враховують регіональні особливості при підборі партнерів, що є критично важливим аспектом у пошуку партнера.
- Поточні системи не пропонують зручних інструментів для синхронізації періодів для гри.
- Найбільшим недоліком є нездатність систем враховувати фактори, такі як темперамент гравця, його схильність до агресії, спокій, бажання співпрацювати, що є ключовим для створення комфортної та продуктивної команди.
- Фрагментованість рішень призводить до необхідності перемикатися між різними платформами, що знижує загальну ефективність пошуку.

Виходячи з цих невирішених проблем, виникає об'єктивна необхідність у розробці нової, якісної інформаційно-пошукової системи. Ця система повинна надати гравцям зручний та потужний інструмент для знаходження партнерів, який не просто поєднує їх за рангом, а дозволяє знайти справді сумісних однодумців, які відповідають їхнім ігровим та особистісним вподобанням.

1.2 Обґрунтування актуальності інформаційно-пошукової системи

Провівши аналіз сучасного стану проблеми пошуку партнерів у багатокористувацьких відеоіграх, стає очевидною наявність значних прогалин в існуючих рішеннях. Внутрішньо-ігрові системи підбору, хоча і забезпечують швидкість знаходження матчу, але вони є надто поверхневими та не здатні враховувати широкий спектр індивідуальних уподобань гравців. Сторонні платформи, такі як Discord сервери та спеціалізовані веб-сайти, хоч і пропонують певну гнучкість, але страждають від відсутності стандартизованих профілів, обмежених можливостей фільтрації. Ці недоліки призводять до значного дискомфорту для гравців, витрат часу на пошук сумісних напарників та, як наслідок, зниження загального задоволення від ігрового процесу [6].

Саме тому розроблення нової, більш розширеної інформаційно-пошукової системи є надзвичайно актуальною і зумовлена кількома ключовими факторами:

- Масштаб та динаміка ігрової індустрії. Ринок багатокористувацьких відеоігор продовжує стрімко зростати, залучаючи мільйони нових гравців. Це створює постійно зростаючу потребу в ефективних інструментах для взаємодії та формування ігрових спільнот.
- Зростання вимог гравців. Сучасні гравці стають більш вимогливими до якості ігрового досвіду. Вони шукають не просто партнерів для заповнення слотів у команді, а однодумців, з якими можна комфортно спілкуватися, відпрацьовувати тактики та досягати спільних цілей. Ігрові системи, що не враховують індивідуальні особливості та вподобання, перестають задовольняти їх потреби.
- Складність командної взаємодії. Багато сучасних ігор вимагають високого рівня командної взаємодії, комунікації та координації. Неможливість знайти гравців з подібним стилем гри, роллю та комунікативними навичками безпосередньо впливає на успіх у грі та здатність отримувати від неї задоволення.

- Відсутність єдиної, комплексної платформи. Це змушує гравців використовувати декілька розрізнених інструментів, що є неефективним та незручним. Централізована система може значно спростити цей процес.
- Психологічний аспект ігрового досвіду. Комфортна атмосфера в команді, відсутність конфліктів та взаємне розуміння є не менш важливими, ніж просто ігрові навички. Існуючі системи практично не враховують психологічну сумісність, що часто призводить до негативних емоцій та "токсичної" поведінки. Розроблювана система, яка дозволить враховувати особистісні теги та характеристики, може вирішити цю проблему.
- Мобільна доступність. З огляду на повсюдне поширення смартфонів, мобільний додаток є найбільш зручним та оперативним інструментом для швидкого пошуку партнерів у будь-який час та в будь-якому місці. Це забезпечує постійний зв'язок між гравцями та можливість миттєво реагувати на пошукові запити.

Отже, розроблення інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх, здатної враховувати розширений набір критеріїв, як ігрових, так і особистісних, забезпечувати швидкий та гнучкий пошук, а також бути доступною на мобільних пристроях, є своєчасною та необхідною. Така система заповнить існуючі ніші та надасть гравцям ефективний інструмент для формування ідеальних ігрових команд, підвищуючи їхнє задоволення від гри та сприяючи розвитку ігрових спільнот.

1.3 Вимоги до інформаційно-пошукової системи вибору партнерів

Для успішної розробки інформаційно-пошукової системи підбору партнерів у багатокористувацьких відеоіграх, яка відповідатиме вимогам цільової аудиторії та сучасним стандартам мобільних додатків, було сформовано комплекс функціональних і нефункціональних вимог. Ці вимоги визначають основу для проєктування архітектури, вибору технологій і подальшої реалізації

системи, забезпечуючи чітке уявлення про необхідний функціонал та якісні характеристики.

Система має підтримувати повний цикл роботи з обліковими записами користувачів, починаючи з реєстрації нових гравців за допомогою електронної пошти та пароля. Важливим компонентом є безпечна та надійна авторизація, яка реалізується через сервіс Firebase Authentication, що гарантує захист даних користувачів та управління їх сесіями [7]. Крім того, система повинна надавати можливість відновлення пароля у разі його втрати, а також забезпечувати гнучке керування профілем, включаючи перегляд, редагування та оновлення персональних даних і налаштувань безпеки.

Таке комплексне формулювання вимог дозволяє створити стабільну, зручну та безпечну платформу, яка відповідатиме сучасним стандартам розробки ігрових мобільних додатків і задовольнить потреби користувачів

Кожен користувач повинен мати змогу створити індивідуальний профіль, який стане основою для ефективного пошуку ігрових партнерів. Цей функціонал передбачає вибір конкретних ігор, у які грає користувач, а також вказівку рівня навичок або рангу для кожної з них, що дозволяє формувати команди з приблизно однаковим рівнем майстерності. Важливою складовою є можливість обирати мови спілкування, що безпосередньо впливає на якість командної взаємодії та допомагає уникнути мовних бар’єрів. Крім того, система має підтримувати додавання особистих тегів або характеристик, які описують ігровий стиль, бажану роль у грі та навіть психологічні особливості гравця, що сприяє пошуку більш сумісних партнерів. Для зручності комунікації передбачена інтеграція з популярними соціальними мережами та ігровими платформами, такими як Discord, Telegram, Steam, Riot Games та Instagram, що дозволяє легко підтримувати контакт із релевантними користувачами. Цей підхід відповідає сучасним тенденціям у сфері підбору гравців і сприяє формуванню більш згуртованих та ефективних команд

Система має надавати розширені інструменти для пошуку гравців за комплексними параметрами, які відповідають сучасним вимогам користувачів. До таких критеріїв належать вибір конкретної гри, бажаний рівень навичок або

ж ранг, регіон для уникнення проблем із пінгом, мова спілкування, оптимальний час для гри, а також особисті теги, характеристики та ігрові ролі, які користувач хоче знайти або запропонувати. Результати пошуку повинні бути представлені у зручному та інформативному вигляді, з можливістю швидко переглянути коротку інформацію про потенційних партнерів, що прискорює процес вибору та переходу до детальних профілів. Для підвищення зручності користувачів передбачена функція збереження улюблених профілів у спеціальному списку, що забезпечує швидкий доступ до них без необхідності повторного пошуку.

Для ефективної роботи з потенційно великою кількістю результатів пошуку, система повинна забезпечувати гнучку та багатофункціональну фільтрацію. Це включає можливість застосування множинних критеріїв одночасно, що дозволяє користувачам швидко орієнтуватися у результатах та знаходити найбільш підходящих кандидатів.

Нефункціональні вимоги описують якісні характеристики системи, що визначають її ефективність, надійність, безпеку та зручність використання.

Система повинна характеризуватися високою швидкістю виконання операцій та запитів, забезпечуючи цільовий час відгуку для типового пошукового запиту. Завантаження даних профілів та результатів пошуку має відбуватися асинхронно, щоб забезпечити плавність роботи інтерфейсу користувача та запобігти його блокуванню. Загалом, інформаційно-пошукова система повинна бути оптимізована для ефективного використання ресурсів мобільного пристрою, мінімізуючи споживання батареї, оперативної пам'яті та процесорного часу, що є критично важливим для мобільних платформ та для ASO (App Store Optimization) [8].

Система повинна гарантувати високий рівень надійності функціонування та безпеки даних користувачів. Це включає захищену аутентифікацію користувачів з використанням перевірених протоколів Firebase Authentication, що забезпечує конфіденційність та цілісність облікових даних. Особлива увага приділяється забезпеченню цілісності даних, що зберігаються в локальній базі даних SQLite та AsyncStorage, оскільки вся ключова інформація системи обробляється на пристрої користувача [9]. Реалізація повинна включати

механізми захисту від несанкціонованого доступу до конфіденційної інформації та мінімізацію ризиків втрати даних.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим, логічно організованим та привабливим для ока, що забезпечує легкість освоєння та приємний досвід взаємодії [10]. Переходи між екранами та взаємодія з інтерактивними елементами повинні супроводжуватися плавними анімаціями, покращуючи загальне сприйняття додатку. Система повинна надавати чіткий візуальний та за необхідності, текстовий зворотний зв'язок на дії користувача, інформуючи його про статус операції або можливі помилки.

Інформаційно-пошукова система на базі мобільного додатку повинна бути кросплатформна, що забезпечує її коректну та стабільну роботу на мобільних пристроях з різними операційними системами – iOS та Android. Інтерфейс користувача має бути адаптуваним до різних розмірів екранів та орієнтацій пристроїв, забезпечуючи однаково зручний досвід незалежно від характеристик пристрою. Також передбачається сумісність з актуальними версіями операційних систем обох платформ для охоплення широкої аудиторії користувачів.

Таким чином, сформовані функціональні та нефункціональні вимоги до інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх закладають міцний фундамент для її подальшої розробки. Вони охоплюють усі основні та ключові аспекти від базового управління користувачами до розширеного функціоналу пошуку з урахуванням індивідуальних уподобань, а також визначають необхідні критерії продуктивності, надійності та зручності використання, орієнтуючись на локальне зберігання даних як ключову особливість архітектури.

1.4 Висновки

У цьому розділі проведено аналіз сучасного стану пошуку партнерів у відеоіграх, який виявив суттєві обмеження існуючих ігрових та сторонніх платформ у врахуванні якісних характеристик гравців, персоналізації пошуку та

підтримці комплексної фільтрації. Обґрунтовано актуальність розробки нової інформаційно-пошукової системи, що зумовлена масштабом ігрової індустрії, зростаючими вимогами гравців та відсутністю єдиного гнучкого рішення. На основі цього сформовано детальні функціональні: управління обліковими записами, розширені профілі, комплексний пошук/фільтрація та нефункціональні: продуктивність, безпека, зручність використання, кросплатформність вимоги до системи. Таким чином, цей розділ заклав міцний фундамент для подальшої розробки, чітко визначивши проблеми та окресливши необхідні характеристики майбутньої системи.

2. АРХІТЕКТУРА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

2.1 Вибір та обґрунтування технологій розробки

Для створення сучасної інформаційно-пошукової системи підбору партнерів у багатокористувацьких відеоіграх було ретельно обрано технологічний стек, який забезпечує високу продуктивність, надійність, масштабованість, швидкість розробки та комфортний користувацький досвід, що також сприяє покращенню ASO (App Store Optimization).

Основою для розробки мобільного додатку стала мова програмування JavaScript, обрана через її універсальність і широку підтримку у сфері мобільної та веб-розробки. Використання JavaScript дозволяє створювати код, який працює на платформах iOS і Android, що значно пришвидшує процес розробки та знижує витрати. Крім того, JavaScript має одну з найбільших і найактивніших спільнот розробників, а також широкий вибір бібліотек, фреймворків та інструментів, що допомагає швидко вирішувати типові завдання та впроваджувати сучасні рішення. Вбудована підтримка асинхронного програмування є ключовою для мобільних додатків, що взаємодіють із базами даних і зовнішніми сервісами, забезпечуючи плавність роботи інтерфейсу без блокування основного потоку. Таким чином, поєднання JavaScript із кросплатформеними технологіями дозволяє створити ефективний, масштабований і зручний додаток, який відповідає сучасним вимогам розробки мобільних ігрових платформ [11].

Комбінація фреймворків React Native та Expo була фундаментальним вибором для реалізації інформаційно-пошукової системи на базі мобільного застосунку, забезпечуючи оптимальний баланс між швидкістю розробки та нативною продуктивністю. Найважливішою перевагою React Native є можливість розробки єдиної кодової бази для обох мобільних платформ – iOS та Android. На відміну від гібридних рішень, React Native компілює код у нативні компоненти інтерфейсу користувача, що забезпечує плавну анімацію, швидкий відгук та ефективне використання ресурсів пристрою користувача. Функції "Hot Reloading" та "Fast Refresh" значно прискорюють процес розробки. Expo, у свою

чергу, значно спрощує розроблення React Native додатків, надаючи готовий набір інструментів та API. Це дозволило швидко розпочати проект без складного налаштування нативних оточень [12]. Expo SDK включає широкий спектр вбудованих модулів для доступу до апаратних можливостей пристрою та інших корисних функцій, що прискорює інтеграцію. Легке тестування на реальних пристроях та спрощена публікація додатку також стали важливими факторами вибору [13].

Для зберігання основних даних користувачів та ігрових профілів було обрано локальну базу даних SQLite. Це рішення базується на перевагах локального зберігання даних можливість роботи офлайн, швидкий доступ до даних та зменшення залежності від інтернет-з'єднання, а також економія мобільного трафіку. Технічно SQLite є легкою у інтеграції з React Native, ефективно зберігає структуровані дані, підтримує SQL-запити та є надійною і стабільною СУБД, що широко використовується в мобільних застосунках. Реляційна модель дозволяє організувати дані у чітко визначених таблицях з підтримкою зв'язків та індексів, що важливо для організації інформації про користувачів, ігри та їхні взаємозв'язки [14].

Для керування процесом аутентифікації користувачів було обрано Firebase Authentication, що є частиною платформи Google Firebase. Цей сервіс забезпечує надійний і безпечний механізм входу, який відповідає сучасним стандартам безпеки, захищаючи облікові дані від поширених атак. Завдяки простій інтеграції з React Native додатками та підтримці різних методів авторизації, реалізація безпечного входу стала значно простішою. Як хмарне рішення, Firebase автоматично масштабується відповідно до зростання кількості користувачів, гарантуючи стабільну та надійну інфраструктуру.

Для організації навігації між екранами використовується бібліотека React Navigation. Вона підтримує різні моделі навігації, зокрема стекову навігацію для послідовного переходу між екранами та вкладки для перемикання між основними розділами додатку. Це дозволяє створити інтуїтивно зрозумілий і гнучкий інтерфейс, що забезпечує комфортне пересування користувача по інформаційно-пошуковій системі. React Navigation також пропонує готові

рішення для управління історією навігації, анімаціями переходів та маршрутизацією, що спрощує реалізацію складної логіки переміщення. Оптимізація бібліотеки гарантує плавну роботу та ефективне використання пам’яті, забезпечуючи швидкі переходи без затримок.

Для покращення користувацького досвіду та розширення функціоналу було інтегровано декілька додаткових бібліотек:

– expo-linear-gradient надає додатку сучасний та візуально привабливий дизайн завдяки використанню градієнтів для елементів інтерфейсу.
– @react-native-async-storage/async-storage була обрана для простого та ефективного локального зберігання невеликих обсягів неструктурованих даних, таких як налаштування користувача та тимчасових даних.
– @expo/vector-icons, яка дозволяє легко додавати різноманітні, масштабовані та привабливі іконки до інтерфейсу, покращуючи навігацію та візуал.

Завдяки компонентній архітектурі React Native та чіткому розподілу відповідальності між модулями досягнуто високого рівня модульності, що значно полегшує тестування, налагодження та масштабування проєкту. Активне використання повторно застосовуваних компонентів не лише прискорює процес розробки, а й забезпечує єдність стилю та поведінки інтерфейсних елементів. Системне управління станом, зокрема з урахуванням локального зберігання даних у SQLite, дозволяє ефективно оновлювати інтерфейс користувача, забезпечуючи швидку та оптимізовану продуктивність.

Обрані технології забезпечують ефективну та якісну розробку, що дозволяє швидко впроваджувати складний функціонал і спрощує подальше обслуговування системи. Вони гарантують високу надійність, продуктивність і безпеку інформаційно-пошуковій системі, а також створюють комфортний користувацький досвід завдяки швидкій роботі та інтуїтивно зрозумілому інтерфейсу. Цей сучасний і оптимізований технологічний стек готовий до подальшого розвитку і відповідає актуальним вимогам мобільної розробки.

2.2 IT-інфраструктура інформаційно-пошукової системи

Проєктування IT-інфраструктури інформаційно-пошукової системи для вибору партнерів у багатокористувацьких відеоіграх передбачає визначення основних компонентів та їх взаємодії для забезпечення стабільної та ефективної роботи. У проєкті застосовано гібридну архітектуру, яка поєднує автономну інформаційно-пошукову систему на базі мобільного додатку із локальною базою даних та інтеграцію з хмарним сервісом для реалізації специфічного функціоналу, що дозволило оптимізувати інфраструктуру [15].

Основою інформаційно-пошукової системи є мобільний додаток на платформі React Native, який працює переважно автономно, зберігаючи ключові дані про користувачів та їхні профілі безпосередньо на пристрої. Ця клієнтська частина відповідає за взаємодію з користувачем, обробку даних і виконання пошукових алгоритмів локально, що забезпечує швидкість та ефективність роботи системи.

Гібридний аспект архітектури полягає у використанні додатково ще хмарного сервісу Firebase Authentication виключно для керування процесами автентифікації та авторизації користувачів. Firebase виступає в ролі серверної частини лише для управління ідентифікацією користувачів, обробляючи запити на реєстрацію, вхід та управління сесіями. Це дозволяє забезпечити доволі високий рівень безпеки та масштабованості для входу в систему, одночасно мінімізуючи залежність основного функціоналу додатку від постійного інтернет-з'єднання .

Ключові компоненти архітектури включають мобільний додаток, що є клієнтською частиною системи, локальну базу даних SQLite, призначену для постійного зберігання всіх основних структурованих даних користувачів, Firebase Authentication, відповідальний за реєстрацію, вхід та управління сесіями користувачів, а також AsyncStorage який являється локальним, неструктурованим сховищем типу "ключ-значення" на пристрої для зберігання налаштувань додатку та тимчасових даних, таких як ідентифікатор поточної сесії користувача [16].

UML-діаграма архітектури інформаційно-пошукової системи вибору партнерів представлена на рисунку 2.1 [17].

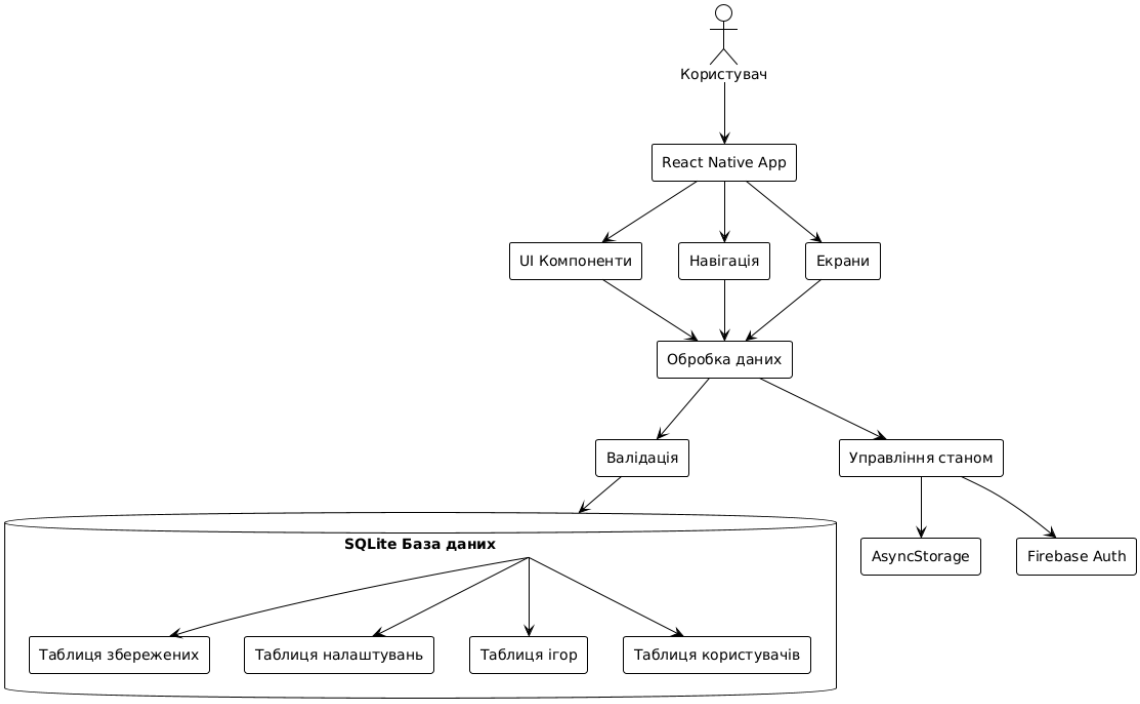

Рисунок 2.1 – Діаграма архітектури інформаційно-пошукової системи.

Взаємодія між компонентами архітектури є ключовою для функціонування системи. Інформаційно-пошукова система взаємодіє з локальною базою даних SQLite через спеціалізовану бібліотеку, що дозволяє виконувати прямі SQL-запити для всіх операцій з даними. Операції з SQLite виконуються асинхронно, запобігаючи блокуванню основного потоку інтерфейсу користувача. Для аутентифікації користувачів система взаємодіє з хмарним сервісом Firebase Authentication. При вході або реєстрації користувача, додаток надсилає відповідні запити до Firebase, який обробляє їх та повертає результат. Ідентифікатор користувача зберігається локально за допомогою AsyncStorage для підтримки активної сесії, що дозволяє користувачу залишатися авторизованим.

Логічна структура мобільного додатку організована за принципом багатошарової архітектури, що забезпечує чітке розділення відповідальності, модульність та легкість у підтримці. Ця структура включає шар представлення,

бізнес-логіки та даних, забезпечуючи доступ до джерел даних таких як SQLite, Firebase Authentication, AsyncStorage.

Архітектура розгортання є відносно простою, оскільки більшість функціоналу зосереджено на клієнтській стороні (рис. 2.2). Мобільний додаток та його локальне зберігання розгортаються безпосередньо на мобільному пристрої користувача. Хмарний сервіс Firebase Authentication, що надає функціонал серверної частини для аутентифікації, розміщується на хмарних сервісах Google, доступ до яких здійснюється за допомогою мережевого з'єднання.

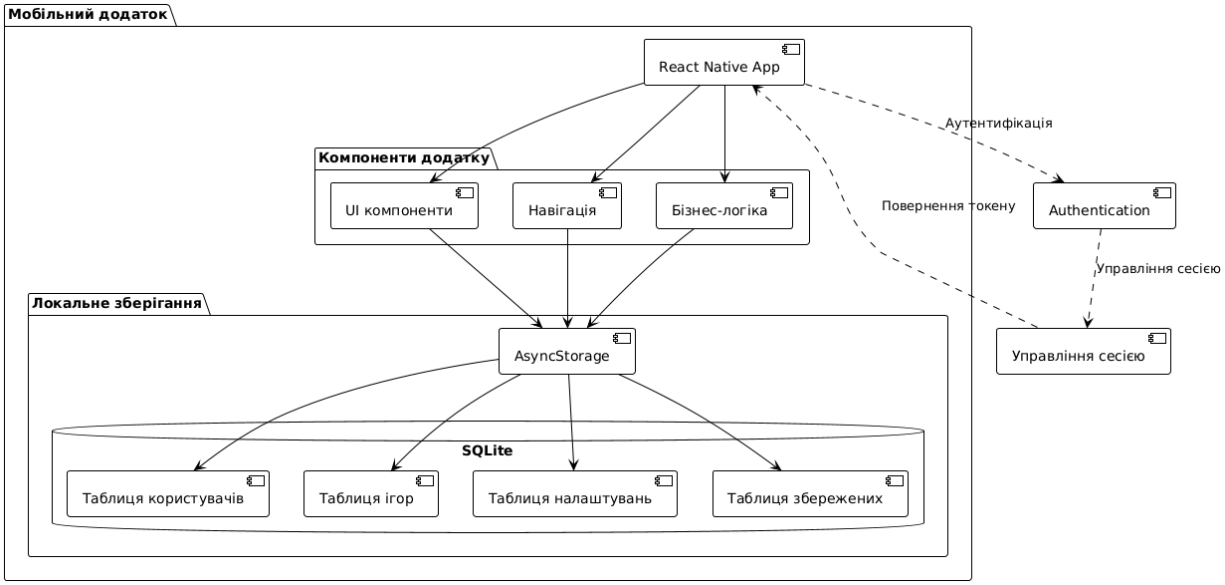

Рисунок 2.2 – Діаграма розгортання системи

Безпека є ключовим аспектом для інформаційно-пошукової системи. Вона забезпечується на декількох рівнях. На рівні аутентифікації використовується Firebase Authentication, яка надає надійні, стандартизовані протоколи безпеки та механізми захисту від типових атак, таких як брутфорс або перебір паролів. Облікові дані користувачів зберігаються на серверах Firebase у зашифрованому вигляді, що мінімізує ризики їх компрометації (рис. 2.3). На клієнтській стороні, дані профілів зберігаються локально в SQLite та AsyncStorage, що знижує ризик їх витоку через мережеві атаки (рис. 2.4). Контроль видимості профілю, який налаштовується користувачем, є важливим елементом приватності та безпеки даних, дозволяючи користувачам самостійно визначати доступність їхньої

інформації. Архітектура додатка також передбачає асинхронну обробку даних для забезпечення стійкості, дозволяючи системі продовжувати функціонувати навіть за умови тимчасових проблем з мережевим з'єднанням, завдяки локальному зберіганню.

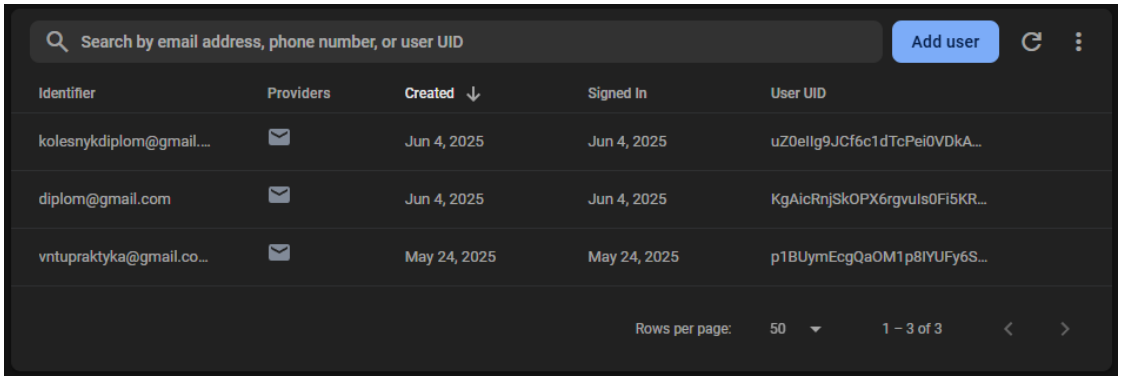

Рисунок 2.3 - Облікові дані користувачів у Firebase Console

id	nickname	region	language	tags	game_time	description	discord	telegram	
Філ...	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
1	CyberNinja	Північна Америка	Англійська	Лідер,Амбіційний	Вечір	Професійний гравець з фокусом на …	cyberninja#1234	cyberninjaTG	cyberni
2	ShadowStriker	Західна Європа	Французька	Спокійний,Надійний	Вечір	Точний та розрахований гравець	shadowstriker#2345	shadowstrikerTG	shadows
3	NightHawk	Південна Азія	Гінді	Спокійний,Точний	Пізній ранок	Точний стрілець з відмінною реакцією	nighthawk#2345	nighthawkTG	nightha
4	PixelWarrior	Європа	Німецька	Аналітичний,Серйозний	Пізній вечір	Стратегічний гравець з досвідом у …	pixelwarrior#5678	pixelwarriorTG	pixelwa
5	SpeedDemon	Південна Америка	Португальська	Емоційний,Веселий	Пізній вечір	Швидкий та точний гравець	speeddemon#7890	speeddemonTG	speedde
6	DragonSlayer	Північна Європа	Шведська	Лідер,Амбіційний	Пізній ранок	Агресивний та цілеспрямований …	dragonslayer#6789	dragonslayerTG	dragons
7	GameSage	Східна Європа	Українська	Комунікабельний,Наставник	Обід	Досвідчений гравець, готовий …	gamesage#9012	gamesageTG	gamesag
8	CrystalStorm	Південна Америка	Іспанська	Аналітичний,Серйозний	Ранок	Стратегічний гравець з відмінним …	crystalstorm#4567	crystalstormTG	crystal
9	DesertFox	Південна Америка	Іспанська	Спокійний,Розумний	Ранок	Розумний та розрахований гравець	desertfox#4567	desertfoxTG	desertf
10	CrystalMage	Північна Америка	Англійська	Аналітичний,Серйозний	Пізній вечір	Стратегічний гравець з відмінним …	crystalmage#0123	crystalmageTG	crystal
11	CrystalKnight	Південна Європа	Грецька	Надійний,Командний гравець	Вечір	Надійний захисник команди	crystalknight#4567	crystalknightTG	crystal
12	FrostGiant	Західна Європа	Нідерландська	Лідер,Серйозний	Обід	Сильний та надійний гравець	frostgiant#8901	frostgiantTG	frostgi
13	ShadowWalker	Азія	Корейська	Спокійний,Точний	Ранок	Точний та непомітний гравець	shadowwalker#6789	shadowwalkerTG	shadoww
14	DesertStorm	Центральна Азія	Турецька	Агресивний,Емоційний	Пізній вечір	Агресивний та швидкий гравець	desertstorm#8901	desertstormTG	deserts
15	IronGuard	Центральна Азія	Турецька	Надійний,Командний гравець	Ранок	Надійний захисник команди	ironguard#8901	ironguardTG	irongua
16	StormRider	Східна Азія	Японська	Емоційний,Веселий	Пізній вечір	Динамічний та енергійний гравець	stormrider#4567	stormriderTG	stormri

Рисунок 2.4 – Таблиця з даними профілів у DB Browser Sqlite

Для розробки використовуються інтегроване середовище розробки Visual Studio Code з відповідними плагінами для JavaScript та React Native [18]. Для тестування додатку застосовуються емулятор віртуальних пристроїв Android Studio Emulator та реальні мобільні пристрої через Expo Go на платформах IOS та Android [19], [20].

На етапі розробки та тестування, моніторинг та налагодження здійснюються за допомогою вбудованих інструментів Expo та стандартних інструментів розробника, які підключаються до React Native. Логування помилок та подій виконується за допомогою консольних виводів.

Обрана архітектура створює надійну базу для подальшого розвитку та масштабування системи. Модульна структура і компонентний підхід значно полегшують додавання нових функцій, підтримку різних ігор, впровадження додаткових фільтрів і покращення продуктивності. Завдяки виділеному шару даних, майбутня інтеграція з віддаленими базами даних для синхронізації або реалізації соціальних функцій буде здійснюватися без суттєвих змін у поточній логіці додатку. Стійкість системи забезпечується автономною роботою ключових функцій на клієнтській стороні та надійністю хмарного сервісу Firebase Authentication.

2.3 Проєктування бази даних

Проєктування бази даних є критично важливим етапом у розробці інформаційно-пошукової системи, оскільки визначає структуру зберігання даних, ефективність пошукових операцій та загальну масштабованість системи. Для інформаційно-пошукової системи було обрано локальну реляційну базу даних SQLite, що забезпечує високу продуктивність та ефективне управління структурованими даними безпосередньо на мобільних пристроях користувача.

Головною метою проєктування бази даних було створення надійної, продуктивної та масштабованої структури для збереження інформації про користувачів та їхні ігрові профілі. Для цього застосовувалися принципи нормалізації до третьої нормальної форми, що дозволило уникнути дублювання даних, чітко розмежувати сутності та забезпечити функціональну залежність усіх полів від первинного ключа. Розподіл даних на логічно пов’язані таблиці мінімізує надмірність і підвищує цілісність інформації. Водночас структура таблиць і типи даних були оптимізовані для швидкого доступу та ефективного виконання пошукових запитів із мінімальним використанням складних операцій об’єднання [21].

База даних складається з чотирьох основних таблиць: users, games, settings та saved_users, кожна з яких відповідає за зберігання певного виду даних, що забезпечує логічну та структуровану організацію інформації.

Таблиця користувачів (users) є так би мовити основною таблицею, яка містить основну інформацію про кожного користувача інформаційно-пошукової системи.

Вона спроектована з наступними полями:

- id: Первинний ключ (INTEGER PRIMARY KEY AUTOINCREMENT), унікально ідентифікує кожного користувача.
- nickname: Текстове поле (TEXT), що зберігає ігровий нікнейм користувача в системі.
- region: Текстове поле (TEXT), що вказує географічний регіон користувача.
- language: Текстове поле (TEXT), що визначає мову спілкування користувача.
- tags: Текстове поле (TEXT), призначене для зберігання тегів користувача, таких як навички, переваги або характеристики користувача, представлених як кома-розділений список.
- game_time: Текстове поле (TEXT), що вказує основний період часу проведення в тій чи іншій відеогрі.
- description: Текстове поле (TEXT), що містить детальний опис профілю користувача.
- discord, telegram, steam, riot, instagram: Текстові поля (TEXT), що зберігають контактні дані користувача в різних соціальних мережах та ігрових платформах.

Ця структура таблиці дозволяє зберігати всю необхідну інформацію про користувачів системи, включаючи їх основні характеристики, ігрові переваги, перелік ігор та соцмереж для подальшої комунікації.

Таблиця games зберігає інформацію про ігри, в які грає кожен користувач. Кожен запис у цій таблиці пов'язаний з конкретним користувачем через поле user_id, який є зовнішнім ключем, що посилається на id у таблиці users. Такий зв'язок формує тип "один-до-багатьох", дозволяючи одному користувачу мати великий список ігор в яких проводить свій вільний час.

Поля цієї таблиці включають:

- id: Первинний ключ (INTEGER PRIMARY KEY AUTOINCREMENT).
- user_id: Зовнішній ключ (INTEGER), що посилається на users(id).
- name: Текстове поле (TEXT) для назви гри.
- rank: Текстове поле (TEXT), що відображає ранг або звання користувача у відповідній грі.
- role: Текстове поле (TEXT), що вказує роль користувача у грі.
- hours: Текстове поле (TEXT), що фіксує кількість годин, проведених у грі.
- nickname: Текстове поле (TEXT), що представляє псевдонім або ж так званий нікнейм користувача саме в цій конкретній грі.

Ця структура таблиці дозволяє зберігати детальну інформацію про ігрові профілі користувачів, включаючи їх ранги, ролі та час, проведений у кожній грі.

Таблиця settings призначена для зберігання індивідуальних налаштувань профілю кожного користувача.

Її поля включають:

- id: Первинний ключ (INTEGER PRIMARY KEY AUTOINCREMENT).
- user_id: Зовнішній ключ (INTEGER), що посилається на users(id).
- excluded: Цілочисельне поле (INTEGER DEFAULT 0), що діє як прапорець видимості профілю.

Ця структура таблиці дозволяє зберігати базові налаштування для кожного користувача, включаючи статус видимості їх профілю в системі.

Таблиця saved_users дозволяє користувачам зберігати профілі інших релевантних для них гравців для швидкого доступу для подальшої комунікації.

Її структура включає:

- id: Первинний ключ (INTEGER PRIMARY KEY AUTOINCREMENT).
- user_id: Зовнішній ключ (INTEGER UNIQUE), що посилається на users(id) збереженого користувача. UNIQUE атрибут гарантує, що один і той самий профіль може бути збережений лише один раз.
- saved_at: Поле типу TIMESTAMP (TIMESTAMP DEFAULT CURRENT_TIMESTAMP), яке автоматично фіксує час збереження профілю.

Ця структура таблиці дозволяє ефективно керувати збереженими профілями користувачів, забезпечуючи унікальність записів та навіть автоматичне відстеження часу їх збереження.

Реляційна модель бази даних передбачає чіткі зв'язки між таблицями для підтримки цілісності даних та ефективного отримання інформації. Між таблицями users та games встановлено зв'язок “один-до-багатьох”, де один користувач може мати інформацію про великий список ігор, а кожен запис про гру належить лише одному користувачеві. Таблиця users пов'язана з таблицею settings зв'язком “один-до-одного”, це означає, що кожен користувач має єдиний набір налаштувань, і кожен набір налаштувань належить лише одному користувачеві. Аналогічно, зв'язок “один-до-одного” існує між users та saved_users, де кожен запис у saved_users стосується унікального збереженого профілю релевантного користувача.

UML-діаграма "сутність-зв'язок" бази даних інформаційно-пошукової системи вибору партнерів представлена на рисунку 2.5.

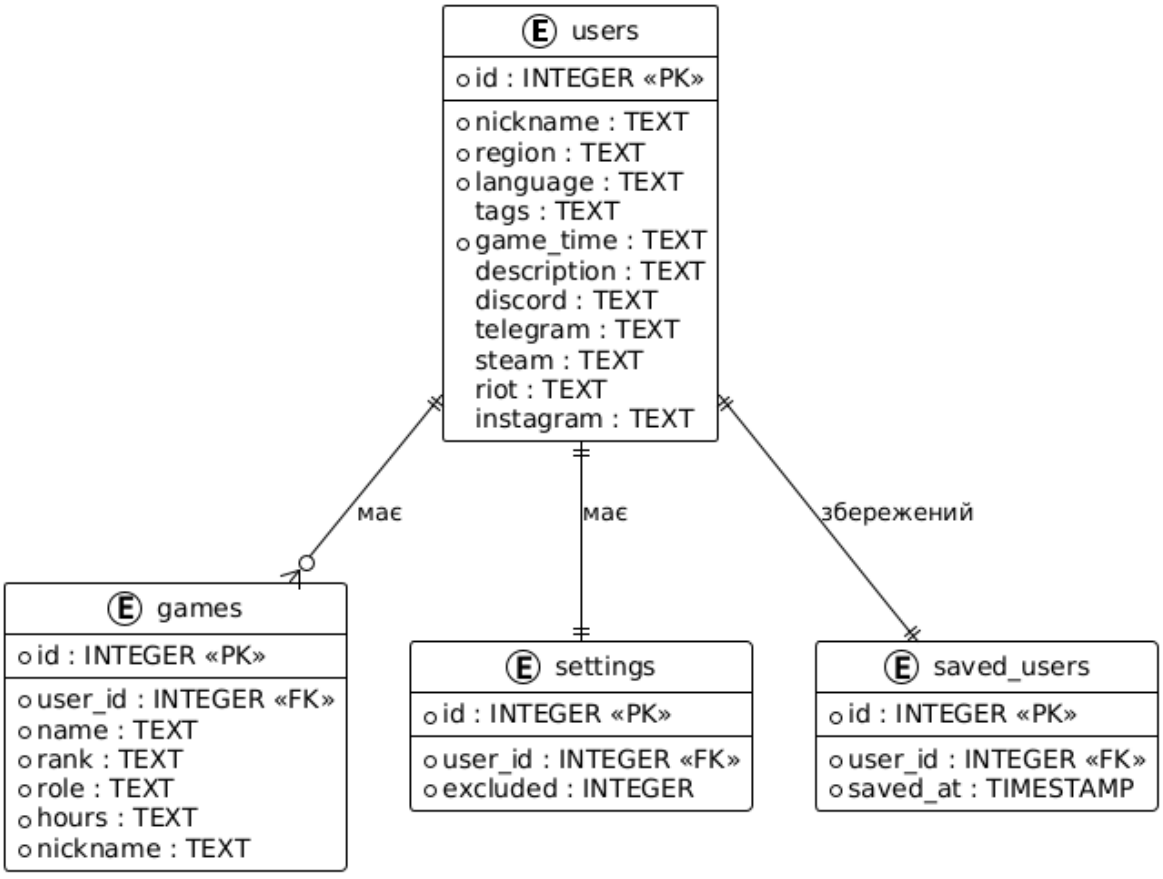

Рисунок 2.5 – Діаграма "сутність-зв'язок" бази даних

Проєктування бази даних базується на принципах нормалізації, що спрямовані на забезпечення цілісності даних та зменшення їх надмірності. Перша нормальна форма гарантує, що всі поля в таблицях містять атомарні, неповторювані значення. Друга нормальна форма забезпечує, що всі неключові атрибути повністю функціонально залежать від первинного ключа, що досягається шляхом розподілу даних на логічно пов’язані таблиці. Третя нормальна форма виключає транзитивні залежності, тобто кожне неключове поле залежить безпосередньо від первинного ключа, що підвищує якість структури бази даних і сприяє її ефективності.

Для підвищення продуктивності запитів було застосовано декілька оптимізаційних підходів. Індекси автоматично створюються для первинних ключів у локальній базі даних SQLite, що забезпечує швидкий доступ до окремих записів. Зовнішні ключі також неявно або явно індексуються для оптимізації операцій об'єднання та пошуку за пов'язаними даними. Використання відповідних типів даних таких як: TEXT, INTEGER, TIMESTAMP сприяє ефективному використанню пам'яті та оптимізації обробки даних. Крім того, застосування обмежень UNIQUE для забезпечення унікальності значень та DEFAULT для встановлення значень за замовчуванням підвищує цілісність та надійність даних у базі.

Таке проєктування бази даних забезпечує ефективне та надійне зберігання інформації, що є фундаментальним для функціонування інформаційно-пошукової системи.

2.4 Алгоритми підбору та фільтрації партнерів

Ключовим компонентом інформаційно-пошукової системи являються алгоритми підбору та фільтрації партнерів, оскільки вони забезпечують ефективний пошук найбільш відповідних гравців за комплексним набором критеріїв. Основна мета алгоритмів, знайти користувачів, які максимально релевантні та відповідають ігровим інтересам та характеристикам, заданим

шукачем. Система працює як потужний інструмент фільтрації та пошуку, що дозволяє користувачам легко та швидко знаходити однодумців для спільної гри.

Дані для пошуку отримуються виключно з локальної бази даних SQLite, що зберігається на мобільних пристроях користувача. Ця база даних містить всю необхідну інформацію, розподілену між основними таблицями, такими як: users для загальної інформації про користувачів, games для деталізації ігрових профілів, settings для зберігання індивідуальних налаштувань, включаючи перемикач excluded, що визначає видимість профілю, та saved_users для профілів, збережених користувачем.

Алгоритми працюють з широким спектром критеріїв пошуку та фільтрації, які користувач може комбінувати як йому заманеться. Це включає базові параметри, такі як регіон, мова спілкування, теги та основний період проводження в грі. Крім того, система дозволяє фільтрувати за специфічними ігровими критеріями, які включають назву гри, її ранг або ігровий навичок, роль у грі, кількість проведених годин та нікнейм користувача в конкретній грі, для можливої додаткової комунікації у тій грі.

На першому етапі, відбувається отримання вхідних даних. Користувач взаємодіє з інтерфейсом додатку, вводячи текст пошуку, вибираючи базові фільтри такі як: регіон, мова, теги, час гри, а також, обираючи конкретні ігри та їх специфічні параметри: назва гри, ранг, роль, години, нікнейм в грі.

Другий етап передбачає базову фільтрацію. Спочатку система виключає з пошуку профілі користувачів, які встановили свій статус як excluded (прихований) у таблиці settings. Це досягається SQL-запитом, який фільтрує користувачів, ідентифікатори яких присутні у таблиці settings з прапорцем excluded = 1. Після цього до решти профілів застосовуються базові фільтри за регіоном, мовою та основним періодом проводження у грі. Якщо користувач вказав нікнейм для пошуку, то на цьому ж етапі відбувається пошук за відповідним полем.

На третьому етапі, здійснюється фільтрація за тегами. Система перевіряє наявність усіх вибраних користувачем тегів у профілях кандидатів. Оскільки теги зберігаються у полі tags як кома-розділений список, алгоритм аналізує цей

рядок для кожного профілю, забезпечуючи відповідність усім зазначеним критеріям або ж фільтрам тегів.

Четвертий етап фокусується на фільтрації за ігровими характеристиками. Для кожного потенційного користувача, що пройшов попередні фільтри, система перевіряє його ігрові профілі або ж список ігор які він додав. Якщо в пошуковому запиті були вказані специфічні ігрові критерії такі як: назва гри, ранг, роль, кількість годин, алгоритм перевіряє відповідність профілю гравця всім цим параметрам у межах кожної ігри. Важливо, що користувач має відповідати всім вказаним ігровим критеріям для кожної обраної гри.

На п'ятому етапі, відбувається формування результатів. Результати не ранжуються за складними алгоритмами релевантності, а видаються у вигляді відфільтрованого списку. Користувачі, які пройшли всі етапи фільтрації, повинні відповідати всім вказаним критеріям пошуку.

Шостий етап полягає у виведенні результатів. Знайдені релевантні профілі які попали під обрані критерії представляються користувачеві у вигляді списку, де кожен профіль містить повну інформацію про користувача та його ігрові характеристики, що дозволяє швидко оцінити потенційних партнерів.

Алгоритми підбору та фільтрації розроблені з урахуванням забезпечення стійкості та оптимальної продуктивності (рис. 2.6). Використання локальної бази даних SQLite дозволяє досягти високої швидкості доступу та обробки даних, оскільки відсутні мережеві затримки. Запити до бази даних оптимізовані за допомогою ефективних SQL-фільтрів, що мінімізує час виконання. Застосування поетапної фільтрації дозволяє послідовно зменшувати обсяг даних, які потребують подальшої обробки, тим самим знижуючи загальне навантаження на систему. Результати пошуку можуть тимчасово кешуватися в стані компонента мобільного додатку, що дозволяє уникнути повторних запитів до бази даних при незначних змінах у фільтрах або навігації, покращуючи відгук інтерфейсу [22].

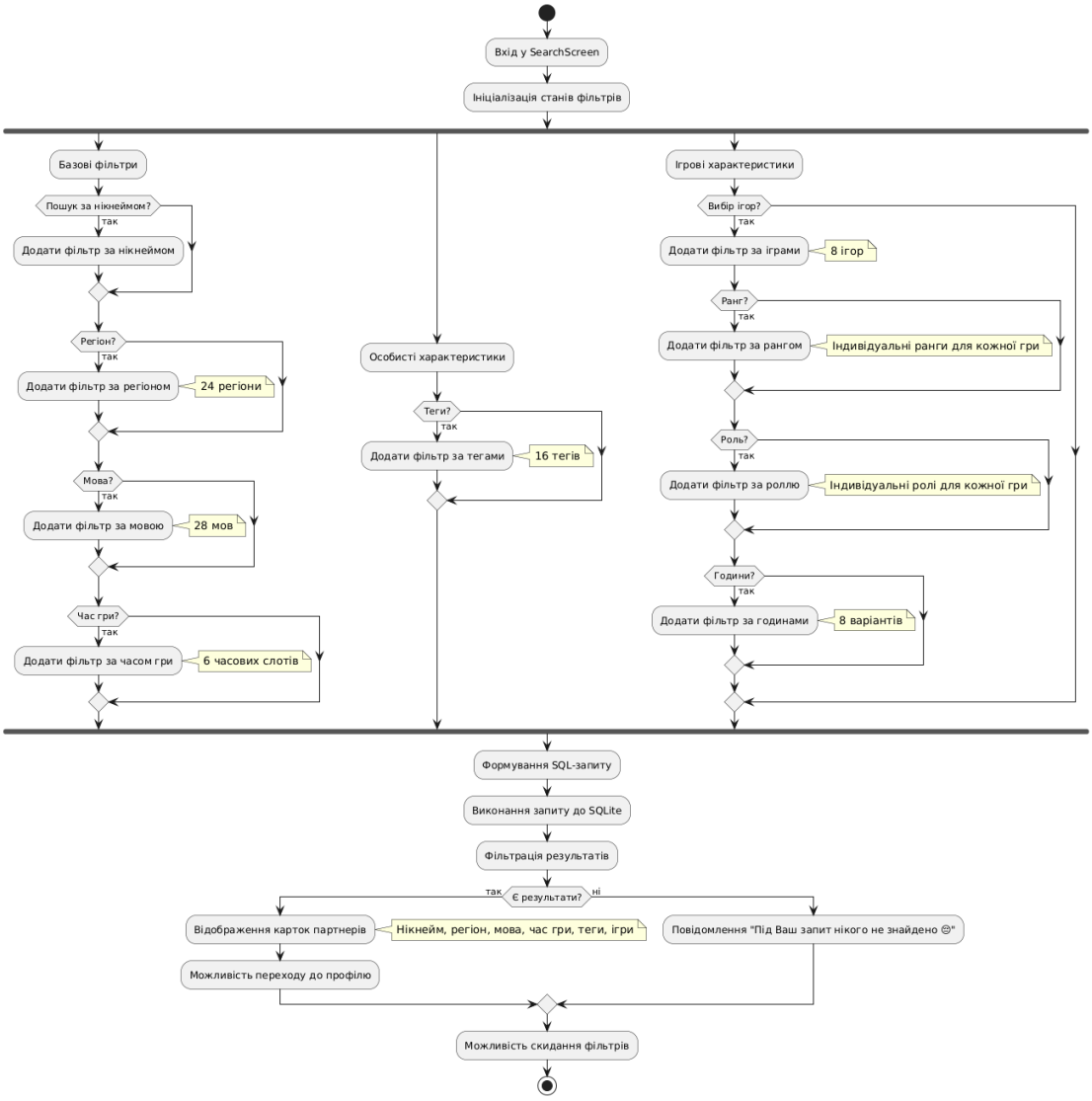

Рисунок 2.6 - Блок-схема алгоритму підбору та фільтрації користувачів

2.5 Висновки

У цьому розділі обґрунтовано вибір технологій розробки, зокрема JavaScript, React Native та Expo для кросплатформової мобільної розробки, SQLite та AsyncStorage для локального зберігання даних, а також Firebase Authentication для безпечної аутентифікації. Запропоновано гібридну архітектуру, що поєднує автономну інформаційно-пошукову систему на базі мобільного додатку із хмарним сервісом для керування ідентифікацією користувачів, забезпечуючи високу швидкість та надійність. Детально спроектовано нормалізовану базу даних SQLite для ефективного зберігання профілів гравців та їх ігрових характеристик. Розроблено поетапні алгоритми

підбору та фільтрації, які використовують локальні дані для швидкого та точного пошуку партнерів за комплексним набором критеріїв.

Таким чином, у розділі закладено міцний технологічний та архітектурний фундамент для ефективної реалізації інформаційно-пошукової системи.

3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

3.1 Опис реалізованих модулів та функціоналу інформаційно-пошукової системи

Інформаційно-пошукова система для підбору партнерів у відеоіграх, яка отримала назву GPFI (Game Partner Finder), є сучасним мобільним додатком, розробленим на кросплатформній основі з використанням фреймворків React Native та Expo. Такий вибір технологій забезпечує високу швидкість розробки, широку сумісність з мобільними пристроями під управлінням операційних систем iOS та Android, а також зручність подальшої підтримки. Ключовою архітектурною особливістю системи є повністю локальна робота з даними. Це забезпечує максимальну автономність функціонування, високий рівень приватності користувачів, оскільки всі персональні та ігрові дані зберігаються безпосередньо на пристрої, а також незалежність від постійного мережевого підключення для виконання основних операцій. Firebase використовується виключно для аутентифікації користувачів, що гарантує надійний та безпечний вхід до системи за допомогою перевіреного сервісу, не перевантажуючи його функціоналом зберігання чутливих даних.

Архітектура додатку побудована на модульному принципі, що значно полегшує розробку, тестування та подальшу підтримку. Кожен модуль відповідає за певний набір функцій, забезпечуючи інтуїтивно зрозумілий та ефективний користувацький досвід. Цей підхід дозволяє легко розширювати функціонал системи, додавати нові можливості та оптимізувати окремі її частини без впливу на всю архітектуру.

Для повного відображення взаємодії користувачів із системою та її загального функціонування, наведено діаграму варіантів використання на рисунку 3.1 та блок-схему функціонування додатку на рисунку 3.2 [23].

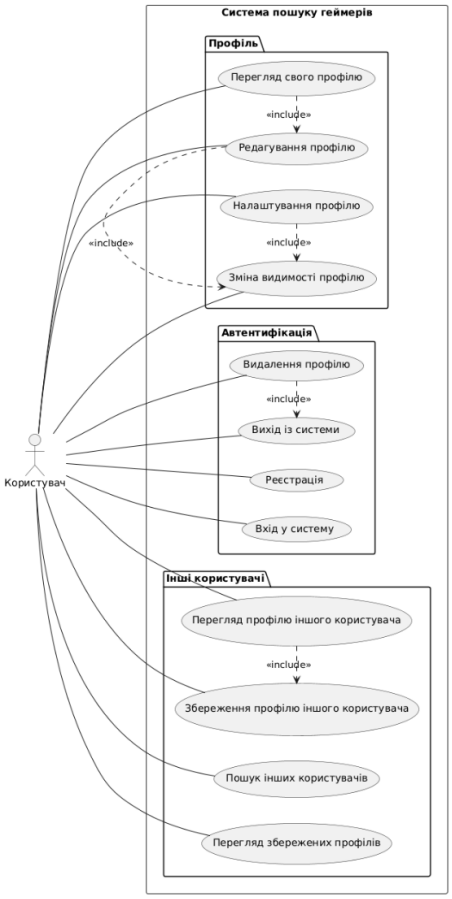

Рисунок 3.1 – Діаграма USE CASE інформаційно-пошукової системи

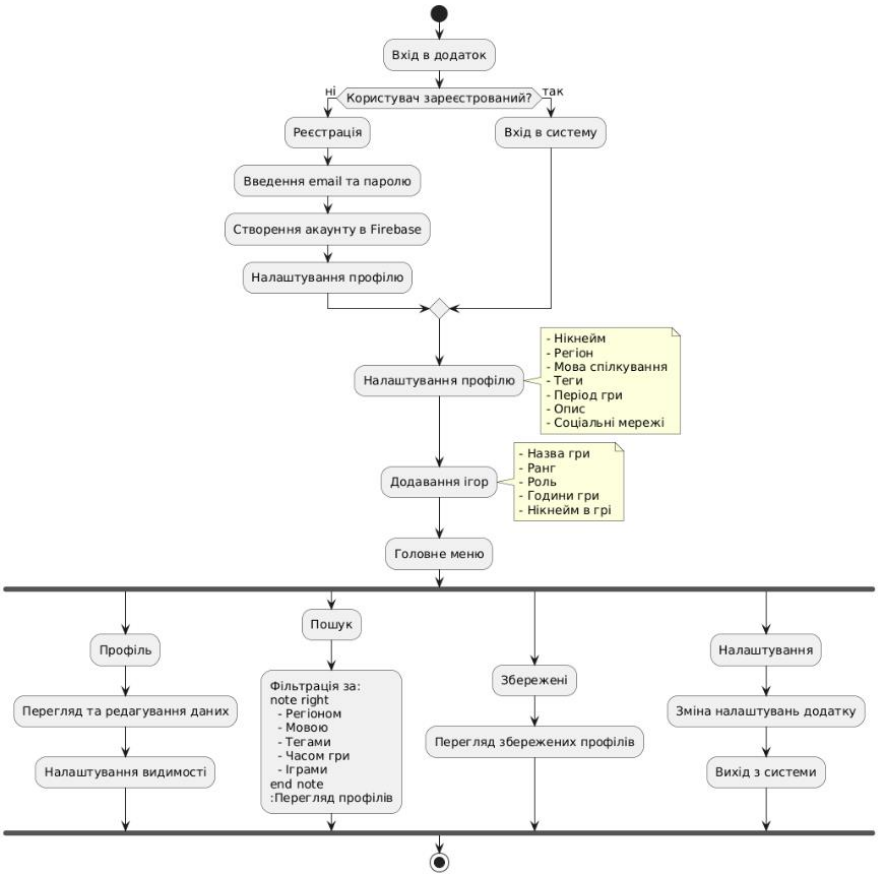

Рисунок 3.2 - Блок-схема функціонування інформаційно-пошукової системи

Модуль пошуку та фільтрації (SearchScreen) є серцем інформаційно-пошукової системи, надаючи користувачам потужні інструменти для знаходження потенційних, релевантних ігрових партнерів. Інтерфейс SearchScreen дозволяє застосовувати різноманітні критерії для точного підбору.

Користувачі можуть здійснювати пошук за нікнеймом для прямого знаходження конкретних гравців. Доступна фільтрація за регіоном та за мовою спілкування, забезпечуючи пошук як і локальних так і глобальних партнерів. Передбачена фільтрація за основним періодом гри, що дозволяє знайти гравців, доступних у бажаний період.

Для більш тонкого підбору реалізована фільтрація за різними тегами, які описують особисті якості або ігрові вподобання гравця, дозволяючи знайти партнерів із сумісними стилями гри або характером.

Модуль надає можливість вибору популярних відеоігор, таких як Valorant, CS:GO, League of Legends, Dota 2, Apex Legends, Overwatch 2, Fortnite та PUBG. Для кожної обраної гри можна застосувати фільтрацію за рангом з індивідуальними рангами для кожної гри, за ролю з індивідуальними ролями, які відповідають специфіці гри, а також за кількістю годин, проведених у грі, що допомагає оцінити досвід гравця.

Функціонал пошуку реалізовано за допомогою SQL-запитів до локальної SQLite бази даних, що забезпечує оптимізований та швидкий пошук. Результати пошуку миттєво оновлюються на екрані. Система також передбачає обробку сценаріїв відсутності результатів та можливість скидання всіх застосованих фільтрів для нового пошуку.

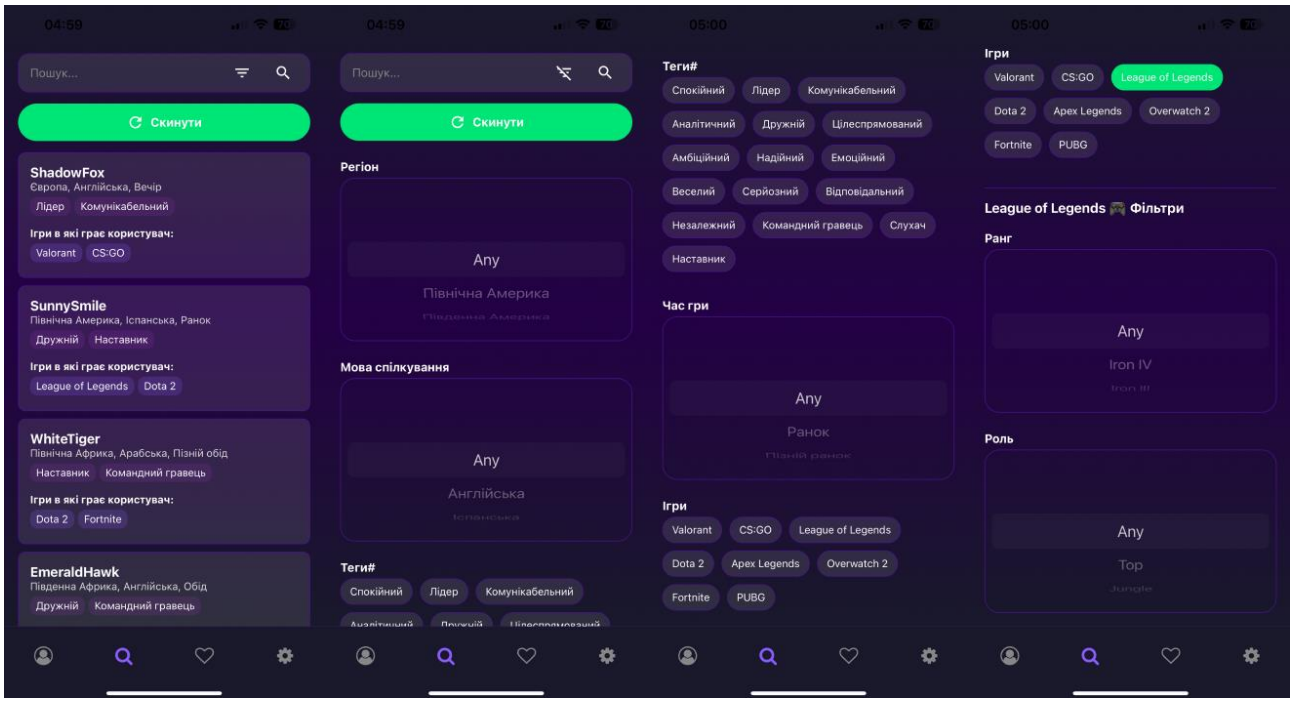

Рисунок 3.3 – Екран пошуку та фільтрація

Модуль збережених профілів (SavedScreen) забезпечує зручний доступ до профілів партнерів, які були позначені користувачем як збережені. Це дозволяє швидко знаходити перевірених або потенційних компаньйонів без необхідності повторного пошуку. На екрані SavedScreen відображається список усіх збережених профілів, з яких можна легко перейти до детального профілю будь-якого партнера. Для кожного збереженого партнера виводиться ключова інформація: нікнейм, регіон, мова, час гри, теги та список ігор, у які він грає.

Система автоматично оновлює список збережених профілів при будь-яких змінах, забезпечуючи актуальність відображуваних даних (рис. 3.4).

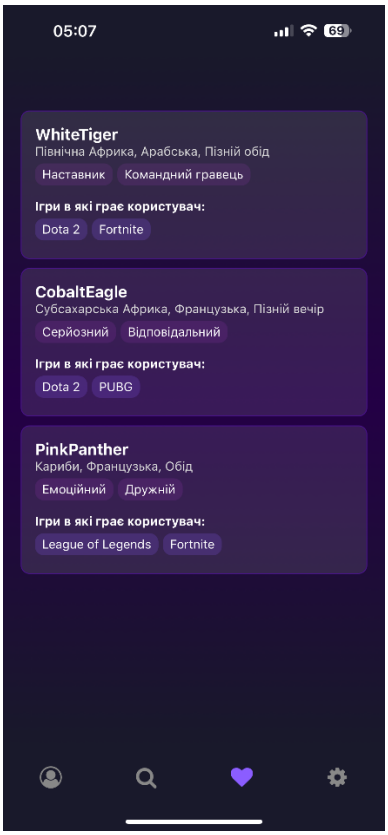

Рисунок 3.4 – Екран збережених профілів.

Модуль налаштування профілю (SetupScreen) дозволяє користувачам створювати та редагувати власні профілі, які потім використовуються системою для підбору партнерів. Користувач може вказати або змінити свій нікнейм, регіон проживання, мову спілкування, додати детальний опис профілю та вибрати бажаний час гри.

Реалізовано зручний інтерфейс для вибору відповідних тегів, що характеризують ігровий стиль та особистість.

Модуль дозволяє додати або оновити посилання на профілі в популярних комунікаційних платформах, таких як Discord, Telegram, Steam, Riot та Instagram, що полегшує зв'язок з потенційними партнерами.

Користувач може додати інформацію про ігри, у які він грає, включаючи вибір конкретних ігор, встановлення свого рангу, вибір ролі та вказання кількості годин, проведених у грі. Також передбачено поле для введення ігрового нікнейму, якщо він відрізняється від основного.

Повна візуалізація модулю налаштування профілю (SetupScreen) представлено на рисунку 3.5.

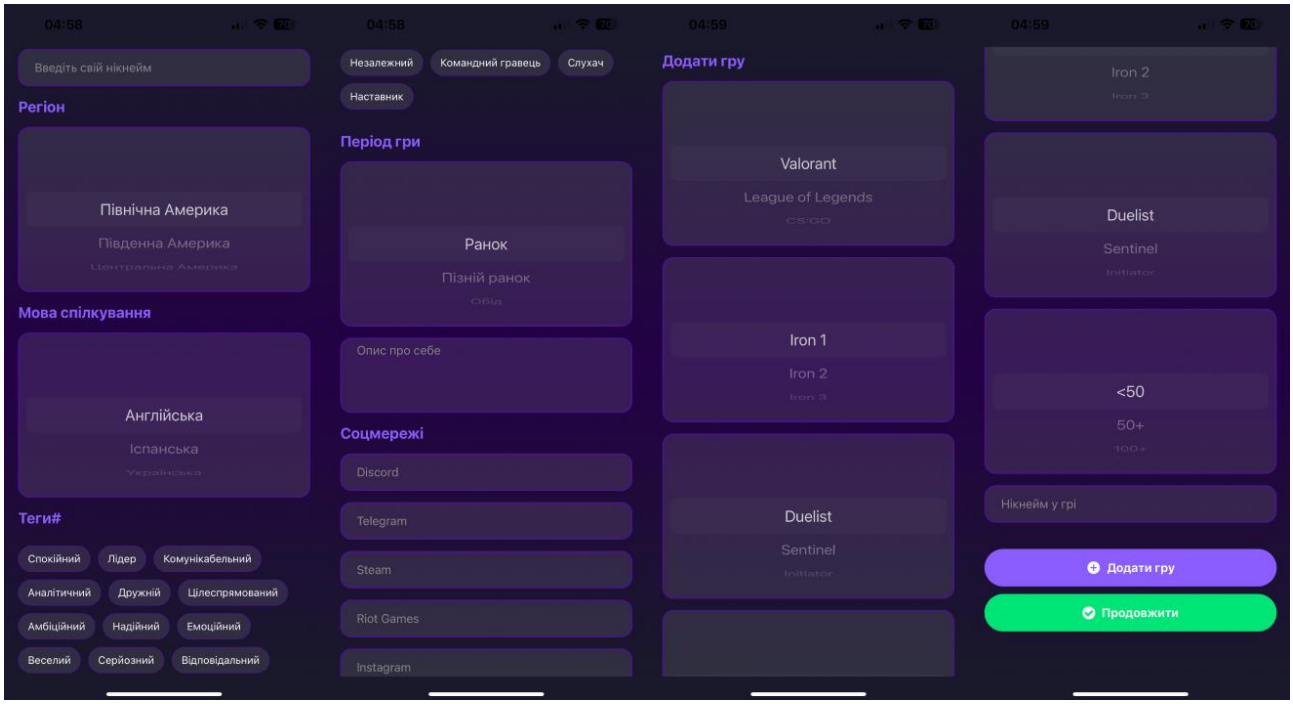

Рисунок 3.5 – Екран налаштування профілю.

Основна взаємодія з даними відбувається через локальну вбудовану базу даних SQLite, яка відповідає за зберігання всієї необхідної інформації для функціонування додатку. Таблиця users зберігає основну інформацію про користувача, його контактні дані та особисті характеристики, що є центральним сховищем профілів. Таблиця games містить детальну інформацію про ігрові профілі користувачів, включаючи ранги, ролі та кількість годин, проведених у кожній грі. Таблиця settings відповідає за зберігання індивідуальних налаштувань користувача, зокрема статус виключення певних параметрів. Таблиця saved_users реалізовує функціонал збереження профілів інших користувачів та фіксує час збереження.

Розроблення інтерфейсу користувача була орієнтована на створення привабливого та функціонального дизайну. Для нього застосовано градієнтний фон, який надає додатку сучасний та естетичний вигляд. Інтерфейс є адаптивним, забезпечуючи коректне відображення та зручність використання на різних мобільних пристроях та розмірах екранів. Система має інтуїтивну навігацію та інтерактивні елементи, такі як кнопки фільтрації, списки вибору, картки профілів та теги, що спрощують взаємодію користувача з додатком.

Окрім описаних модулів, система включає ряд додаткових функцій, що покращують користувацький досвід:

– Користувачі можуть легко зберігати профілі інших гравців для подальшого швидкого доступу.
– Реалізовано можливість переходу до детального профілю будь-якого гравця для ознайомлення з повною інформацією.
– Забезпечена плавна та логічна навігація між усіма основними екранами додатку.
– Система містить механізми обробки помилок, що підвищує її стабільність та надійність.
– Функціонал системи забезпечує автоматичне оновлення відображуваних даних, наприклад, при зміні профілю користувача або збережених профілів.

Блок-схема алгоритм роботи інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх представлено на рисунку 3.6.

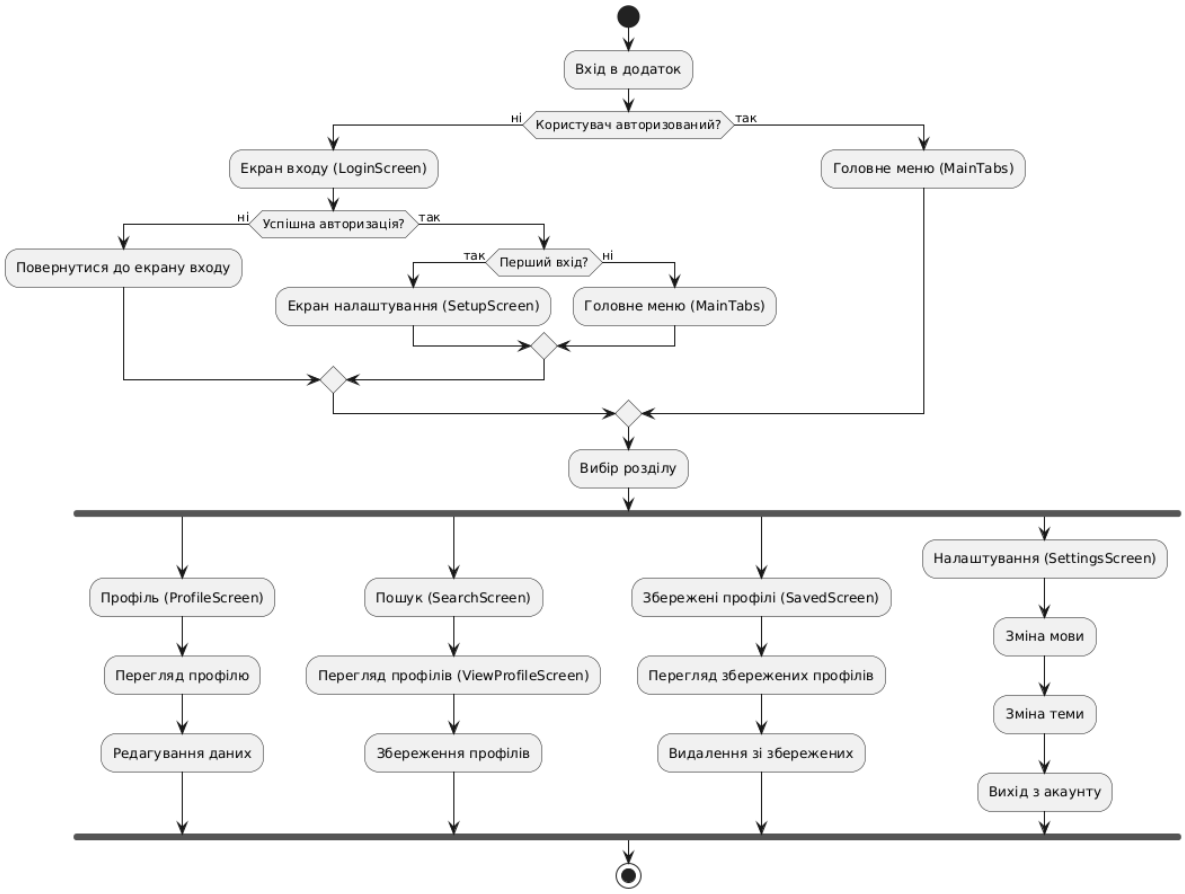

Рисунок 3.6 – Блок-схема алгоритм роботи інформаційно-пошукової системи

Таким чином, опис повністю відображає реалізований функціонал інформаційно-пошукової системи, де локальне зберігання даних використовується як основний спосіб збереження інформації про користувачів і їхні профілі, а Firebase застосовується виключно для аутентифікації користувачів. Такий підхід відповідає сучасним практикам розробки мобільних додатків, поєднуючи швидкість і автономність локального зберігання з надійністю та безпекою хмарного сервісу Firebase Authentication

3.2 Особливості реалізації та оптимізації програмного забезпечення

Розроблене програмне забезпечення інформаційно-пошукової системи було спроектовано та реалізовано з особливим акцентом на продуктивність, ефективність використання ресурсів, зручність для користувача та безпеку даних. Ці аспекти досягаються завдяки низці специфічних особливостей реалізації та всіх оптимізаційних рішень, що застосовані на різних рівнях архітектури додатку, починаючи від управління базою даних до взаємодії з інтерфейсом користувача.

Оптимізація бази даних є фундаментальним елементом для забезпечення високої швидкості та ефективності роботи системи, враховуючи її повністю локальний характер. Використовується локальна база даних SQLite для зберігання всіх даних, що гарантує локальне зберігання. Такий підхід забезпечує швидкий доступ до даних та ефективне використання ресурсів мобільного пристрою користувача. Крім того, особлива увага приділяється оптимізації SQL-запитів, які формуються для пошуку та фільтрації, забезпечуючи їхню максимальну ефективність та мінімальний час відгуку. Для підвищення безпеки даних використовуються параметризовані запити, що роблять неможливими SQL-ін'єкції [24].

Пошуку та фільтрація є критично важливим для основного функціоналу інформаційно-пошукової системи. Ефективна фільтрація досягається через точне формування SQL-запитів з урахуванням усіх вибраних користувачем

критеріїв. Цей процес передбачає побудову SQL-запитів з урахуванням обраних фільтрів, використання параметризованих запитів для підвищення безпеки та оптимізацію шляхом виключення непотрібних умов з пошукових запитів. Для прискорення повторного доступу до часто запитуваних даних може застосовуватись кешування результатів пошуку на клієнтській стороні, що мінімізує звернення до бази даних.

Інтерфейс користувача спрямований на забезпечення плавної та приємної взаємодії. Ефективне відображення великих обсягів даних забезпечується використанням компоненту ScrollView для довгих списків, що оптимізує рендеринг та пам'ять. Відображення карток профілів також оптимізовано для швидкого завантаження та плавного скролінгу. Візуальне сприйняття покращено за рахунок використання градієнтного фону, що надає додатку сучасного вигляду.

Оптимізація станів компонентів є ключовим аспектом продуктивності в React Native. Управління станом реалізовано з використанням хуку useState для локальних станів компонентів, що дозволяє ефективно керувати даними на рівні окремих елементів інтерфейсу. Ефективне оновлення станів фільтрів забезпечує миттєву реакцію на дії користувача. Оптимізоване оновлення списку результатів запобігає зайвим перемальовуванням компонентів. Слід зазначити що передбачено кешування станів для швидкого доступу до раніше завантажених даних або попередніх налаштувань.

Навігація забезпечує безперебійний та швидкий перехід між різними екранами додатку. Реалізовано швидкий перехід між екранами, що підвищує загальну сприйнятливість системи. Збереження стану при навігації дозволяє користувачеві повертатися до попередніх екранів без втрати даних чи контексту. Ефективна обробка повернення на попередній екран гарантує коректну роботу всього навігаційного стеку.

Оптимізація фільтрів як окремий аспект підкреслює гнучкість та швидкість роботи з пошуковими критеріями. Це досягається завдяки модульній структурі фільтрів, що дозволяє легко додавати або модифікувати критерії. Забезпечено швидке оновлення результатів при зміні фільтрів для інтерактивності,

оптимізоване збереження стану фільтрів між сесіями або при переключенні екранів, а також ефективне скидання фільтрів до початкового стану одним натисканням.

Збереження даних стосується ефективності операцій з таблицею збережених профілів. Вона включає оптимізоване збереження профілів, що відбувається швидко та без затримок, ефективне оновлення збережених даних у разі їх зміни, швидкий доступ до збережених профілів, що дозволяє користувачеві миттєво переглядати їхній список, та оптимізоване видалення збережених профілів.

Обробка помилок є ключовим фактором стабільності та надійності програмного забезпечення. Це реалізується через ретельну перевірку введених користувачем даних для запобігання некоректним операціям, коректну обробку сценаріїв відсутності результатів пошуку з відображенням відповідних повідомлень, надання інформативних повідомлень про помилки користувачеві та застосування механізмів відновлення після помилок, що підвищує стійкість системи [25].

Оптимізація продуктивності є загальним принципом, що пронизує всю розробку. Вона охоплює мінімізацію перезавантажень компонентів для зменшення навантаження на процесор, оптимізоване оновлення інтерфейсу, що забезпечує плавність анімації та взаємодії, ефективне використання пам'яті для уникнення витоків та підвисань додатку, а також швидке завантаження даних при запуску та в процесі роботи.

Оптимізація безпеки є пріоритетом для захисту даних користувачів. Це досягається завдяки безпечному зберіганню даних локально на пристрої, що обмежує доступ сторонніх осіб. Захист від SQL-ін'єкцій реалізовано через використання параметризованих запитів. Також застосовується валідація введених даних для запобігання внесенню шкідливої або некоректної інформації.

Ці особливості реалізації та оптимізації дозволяють інформаційно-пошуковій системі функціонувати ефективно, стабільно та безпечно, забезпечуючи при цьому високий рівень зручності для кінцевого користувача.

3.3 Забезпечення безпеки інформаційно-пошукової системи

Безпека інформаційно-пошукової системи є одним із ключових аспектів її розробки та функціонування, навіть за умови, що основна робота з даними відбувається локально на пристрої користувача. Забезпечення конфіденційності, цілісності та доступності даних досягається комплексом заходів, що охоплюють усі рівні взаємодії з інформацією.

У системі Firebase використовується виключно для аутентифікації користувачів, що гарантує надійний і перевірений механізм входу. Водночас, всі дані профілів зберігаються локально за допомогою SQLite, що виключає необхідність передачі персональної інформації на віддалені сервери, підвищуючи рівень приватності.

Захист бази даних є пріоритетним завданням, оскільки саме там зберігається вся інформація користувачів. Безпека SQLite забезпечується використанням параметризованих SQL-запитів, які ефективно запобігають SQL-ін’єкціям, оскільки дані користувача передаються окремо від самого запиту, виключаючи можливість впровадження шкідливого коду. Доступ до бази даних здійснюється виключно через спеціалізовані внутрішні функції додатку, що контролюють усі операції читання та запису. Перед кожним записом у базу обов’язково виконується валідація вхідних даних, що запобігає збереженню некоректної або потенційно шкідливої інформації. Конфіденційні дані зберігаються з дотриманням принципів конфіденційності, використовуючи властивості локального сховища SQLite.

Захист даних користувача охоплює весь життєвий цикл персональної інформації в системі. Безпека профілів забезпечується ретельною валідацією всіх введених користувачем даних, включно з перевіркою формату та відповідності встановленим вимогам, зокрема коректності контактної інформації. Персональні дані захищаються від несанкціонованого доступу завдяки локальному зберіганню на пристрої користувача та відсутності передачі на зовнішні сервери.

Захист пошукових запитів важливий для запобігання зловмисному використанню функціоналу пошуку. Безпека пошуку досягається завдяки

валідації пошукових параметрів, що надходять від користувача, для забезпечення їхньої коректності та виключення шкідливих спроб втручання. Система забезпечує безпечну обробку спеціальних символів у пошукових запитах, щоб уникнути їх інтерпретації як частин SQL-коду. Додатково встановлено обмеження на довжину пошукових запитів, що є мірою запобігання ресурсним атакам.

Захист навігації гарантує, що користувачі можуть отримувати доступ лише до дозволених їм екранів та функцій. Це включає валідацію параметрів навігації, що передаються між екранами, для уникнення їхнього підроблення та запобігання несанкціонованому доступу до певних екранів, що містять конфіденційні налаштування або дані. Реалізована безпечна передача даних між екранами забезпечує цілісність інформації під час переходу.

Захист збережених даних стосується безпеки інформації, яку користувач свідомо зберігає. Він включає валідацію даних безпосередньо перед їх збереженням у таблицю saved_users, а також захист від несанкціонованого доступу до збережених профілів іншими користувачами або програмами. Передбачено безпечне видалення збережених даних, що гарантує їх повне видалення без можливості відновлення.

Захист від шкідливих дій реалізується шляхом постійної валідації всіх введених користувачем даних та перевірки формату даних на відповідність очікуваному типу та структурі. Система захищена від введення шкідливого коду, наприклад, скриптів або команд, у текстові поля, а також має обмеження на довжину введених даних як додатковий бар'єр проти атак переповнення.

Захист конфігурації забезпечує цілісність та незмінність налаштувань системи. Це включає захист конфігураційних файлів від несанкціонованого доступу або модифікації, а також безпечне зберігання налаштувань користувача в локальній базі даних. Здійснюється валідація змін налаштувань, що вносяться користувачем або системою, та захист від несанкціонованих змін конфігурації.

Захист від помилок як захід безпеки є проактивним підходом до підвищення надійності системи. Він включає валідацію вхідних даних на всіх етапах обробки для запобігання помилкам, спричиненим некоректною інформацією.

Передбачена коректна обробка некоректних даних, що виключає їх вплив на стабільність системи. Система має захист від збоїв, що забезпечується стійкістю коду та обробкою винятків, а також механізми відновлення після помилок, які дозволяють додатку продовжувати роботу або коректно завершити операцію.
Захист комунікації в контексті локальної системи стосується безпеки контактної інформації та її використання. Він передбачає валідацію контактних даних, що вводяться користувачем, для забезпечення їхньої достовірності, а також безпечну передачу повідомлень, що стосуються, наприклад, відображення контактної інформації.

Захист інтерфейсу гарантує, що користувач взаємодіє з надійною та безпечною оболонкою додатку. Це включає валідацію всіх введених даних безпосередньо через елементи інтерфейсу, що запобігає введенню шкідливої інформації через UI-компоненти, які потенційно можуть бути використані для атак. Система забезпечує безпечне відображення даних, виключаючи можливість витоку конфіденційної інформації або маніпуляцій із виводом, а також захищає від спроб обходу функціональних обмежень через інтерфейс.

Все це формує багатошаровий рівень захисту інформаційно-пошукової системи, що гарантує її надійну та безпечну роботу в умовах локального зберігання даних і аутентифікації користувачів через Firebase. Це підвищує довіру користувачів і забезпечує стабільність роботи додатку навіть у разі потенційних загроз [26].

3.4 Тестування системи та оцінка ефективності

Після запуску додатку можна провести тестування його функціоналу. Перевірка здійснювалася на платформі iOS за допомогою додатку Expo GO. Також підтверджено коректну роботу на Android. Інтерфейс повністю адаптований під різні розміри екранів, що сприяє покращенню користувацького досвіду та позитивно впливає на ASO (App Store Optimization), також відоме як мобільне SEO (Search Engine Optimization).

Першим екраном, який бачить користувач, є LoginScreen, це екран аутентифікації, де він має змогу як зареєструватися, так і увійти до системи. Доступ до основного функціоналу додатку можливий лише після проходження процедури реєстрації та створення особистого профілю або ж внесення даних за якими користувач вже створив профіль (рис. 3.7).

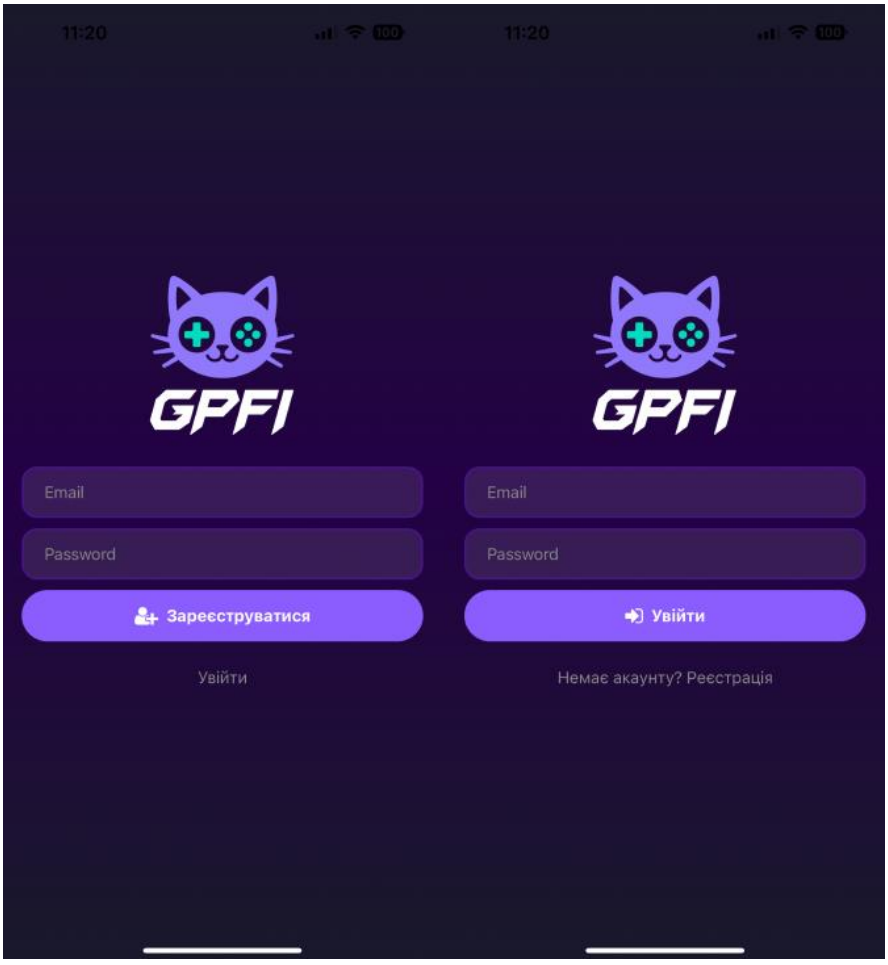

Рисунок 3.7 – Екран аутентифікації

Також було проведено перевірку на обробку помилок під час процесів аутентифікації, як при вході в систему, так і під час реєстрації нового користувача. У випадку виникнення помилок: неправильний формат email, вже зареєстрована адреса, слабкий пароль, тощо, система коректно реагує, виводячи відповідні повідомлення для користувача (рис. 3.8). Це забезпечує зручний зворотний зв'язок і запобігає непередбаченим збоям у роботі додатку.

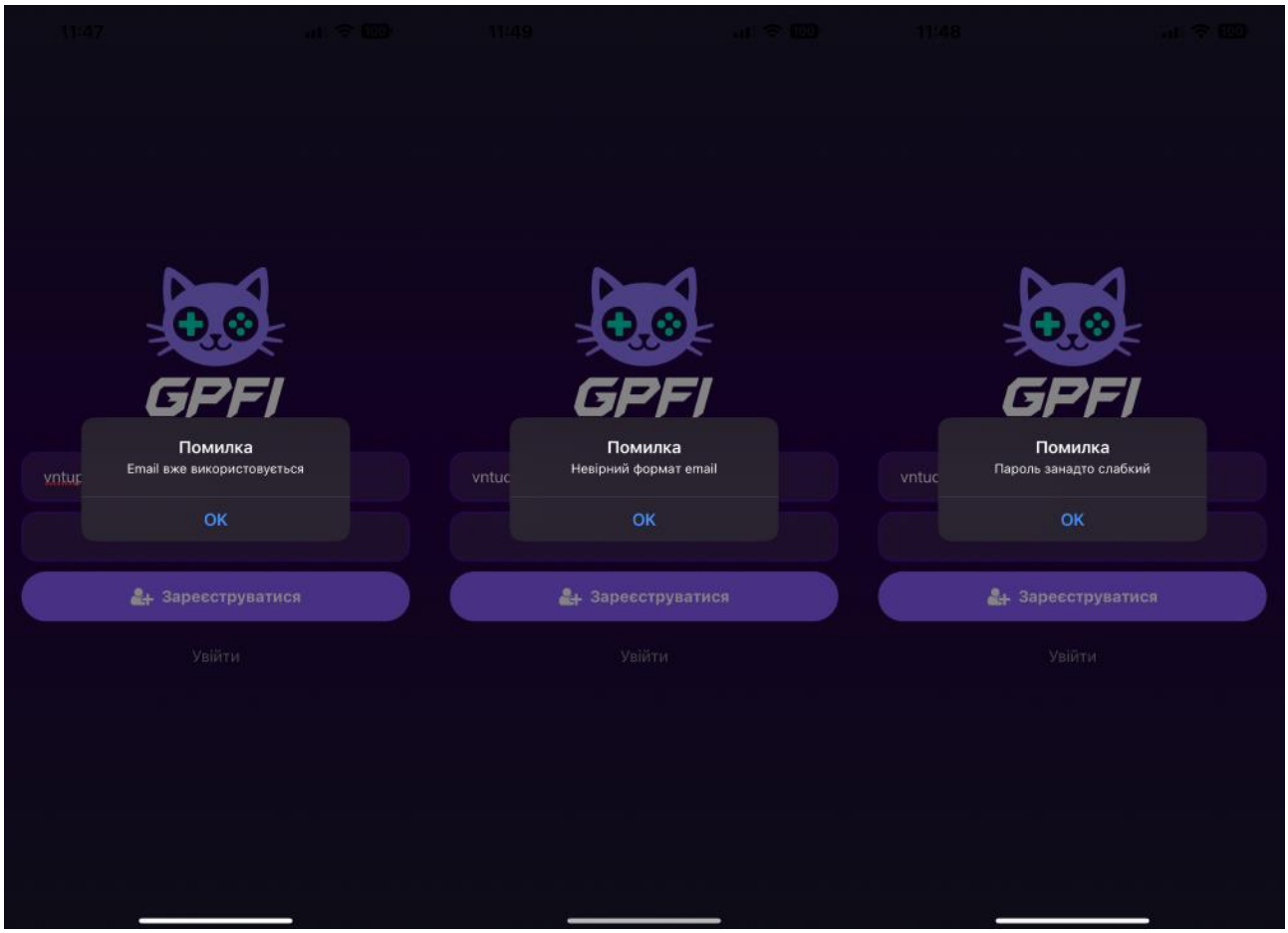

Рисунок 3.8 – Обробка помилок при аутентифікації

Після успішної реєстрації користувач зберігається в системі під своїм унікальним ідентифікатором (UID), що генерується автоматично. Уся логіка обробки та додавання до Firebase console відбувається коректно, що гарантує стабільну реєстрацію та подальшу ідентифікацію користувача в додатку.

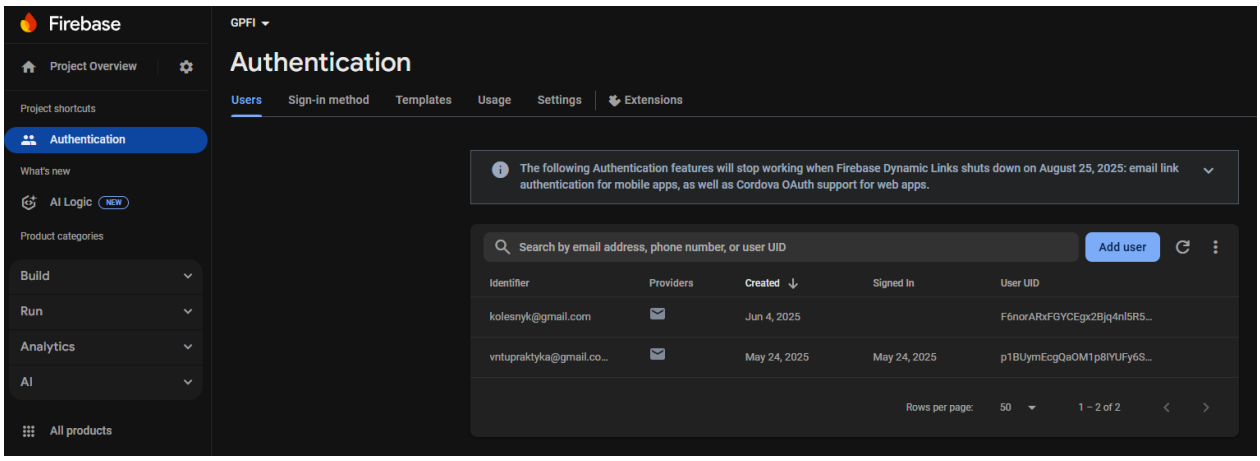

Рисунок 3.9 – Вкладка Authentication в Firebase console

Після реєстрації користувача перекидує на екран SetupScreen, на якому він створює свій профіль з необхідними параметрами з якими буде шукати собі релевантного напарника (рис. 3.10).

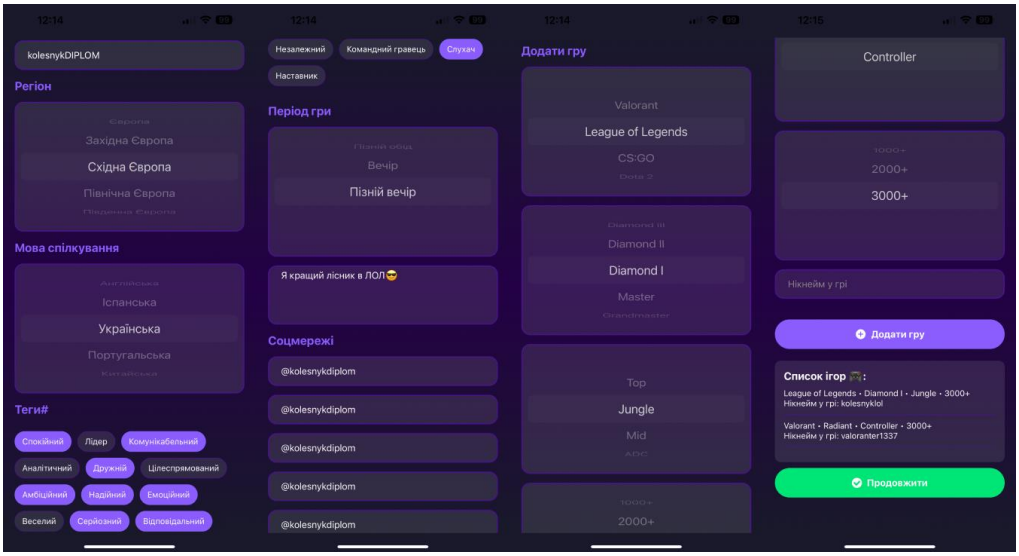

Рисунок 3.10 – Створення профілю

Після успішного створення профілю користувач потрапляє на екран власного профілю (ProfileScreen) (рис. 3.11), де йому відкривається доступ до основної навігації додатку (рис. 3.12). Звідси він може переходити до інших функцій інформаційно-пошукової системи, а також редагувати особисту інформацію при потребі (рис. 3.13).

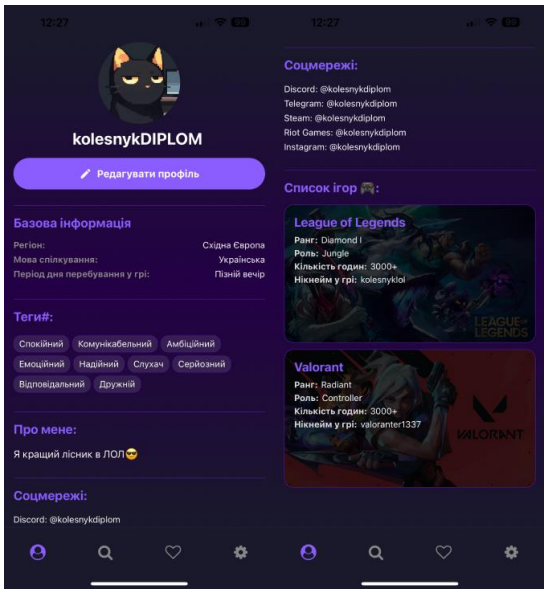

Рисунок 3.11 – Власний профіль користувача

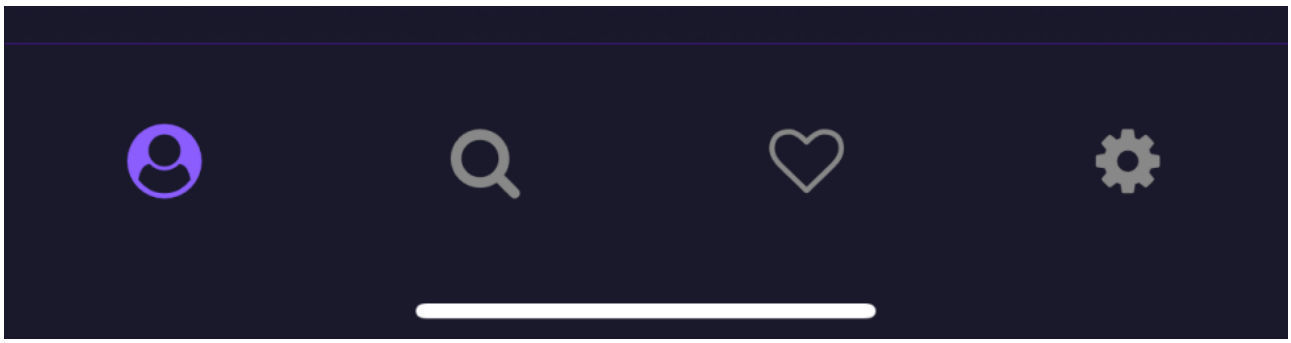

Рисунок 3.12 – Таби навігації додатку

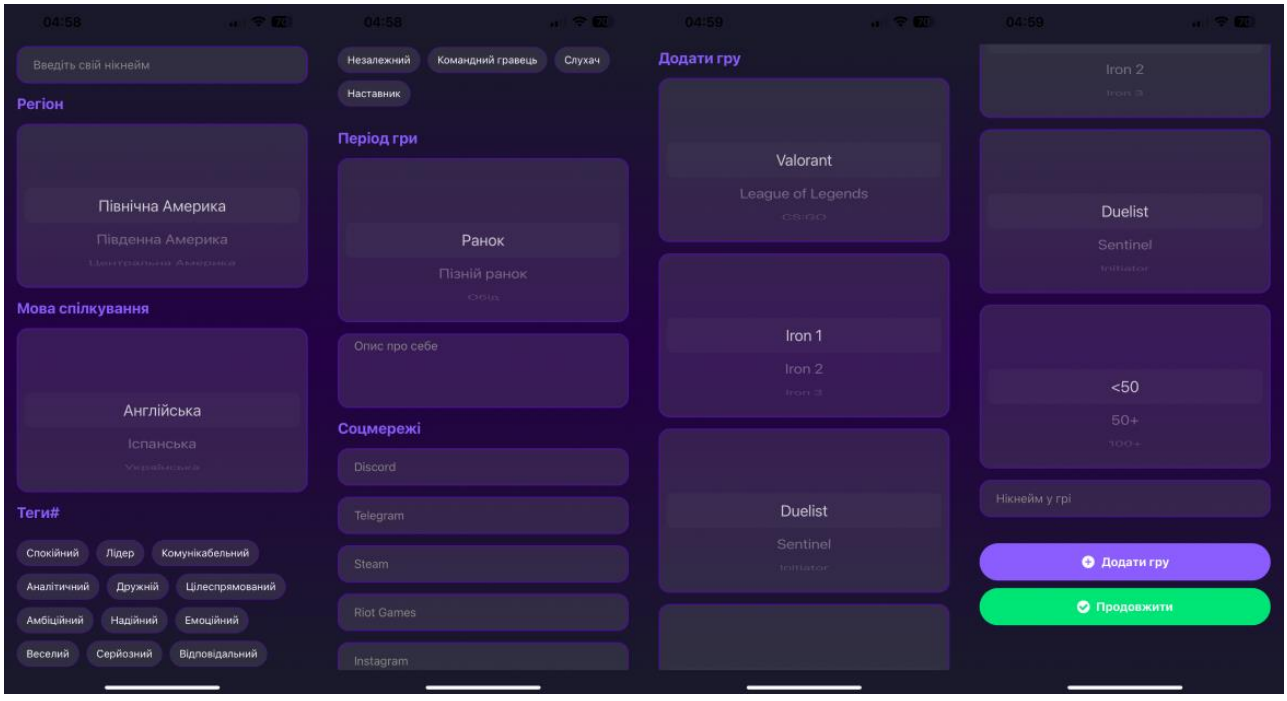

Рисунок 3.13 – Екран редагування профілю

На екрані SearchScreen (рис. 3.14) користувач бачить інтерфейс пошуку, який включає набір фільтрів (рис. 3.15) для уточнення запиту, а також результати пошуку. У списку результатів відображаються інші користувачі з базовою інформацією про себе та переліком ігор, у які вони грають, також у цьому списку ранжується так званий шукач (рис. 3.16). За потреби користувач може швидко очистити всі встановлені фільтри, натиснувши кнопку «Скинути». У разі, якщо за заданими фільтрами не знайдено жодного релевантного користувача, на екрані виводиться відповідне повідомлення “Під Ваш запит нікого не знайдено”. Це дозволяє користувачу зрозуміти, що результати відсутні, і за потреби змінити або скинути фільтри для повторного пошуку.

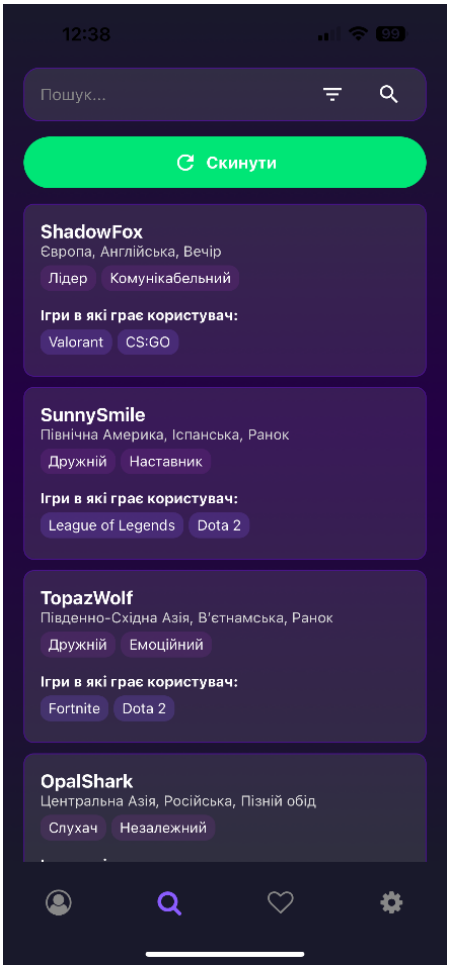

Рисунок 3.14 – Екран пошуку напарників

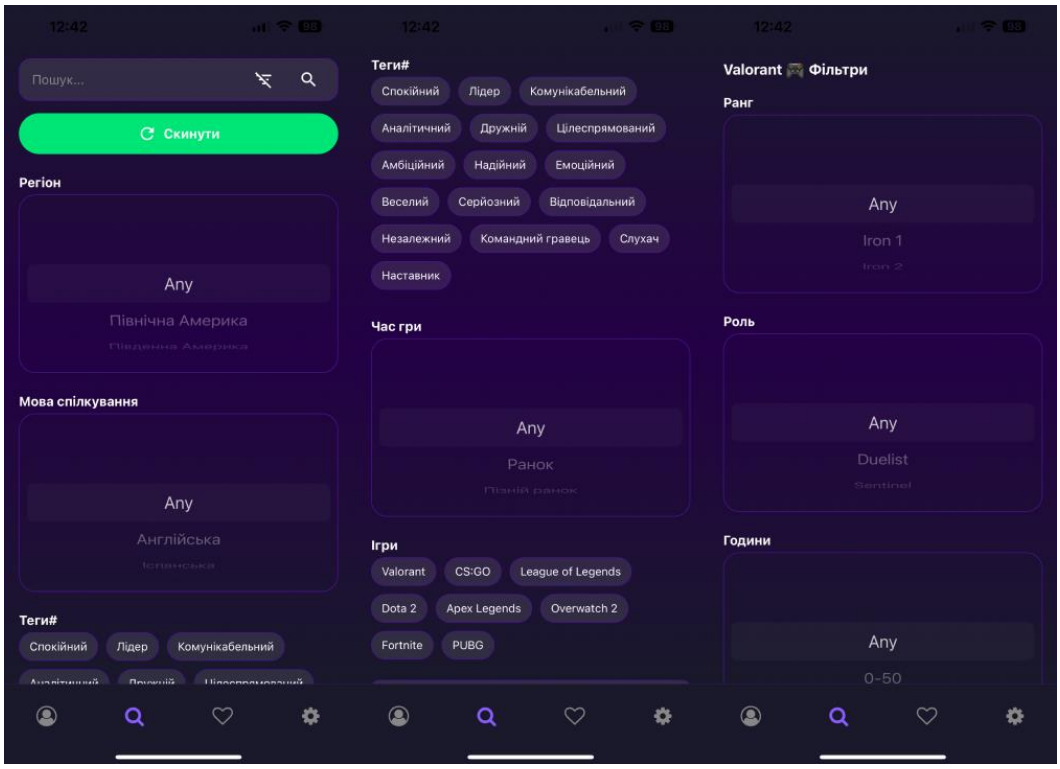

Рисунок 3.15 – Набір фільтрів

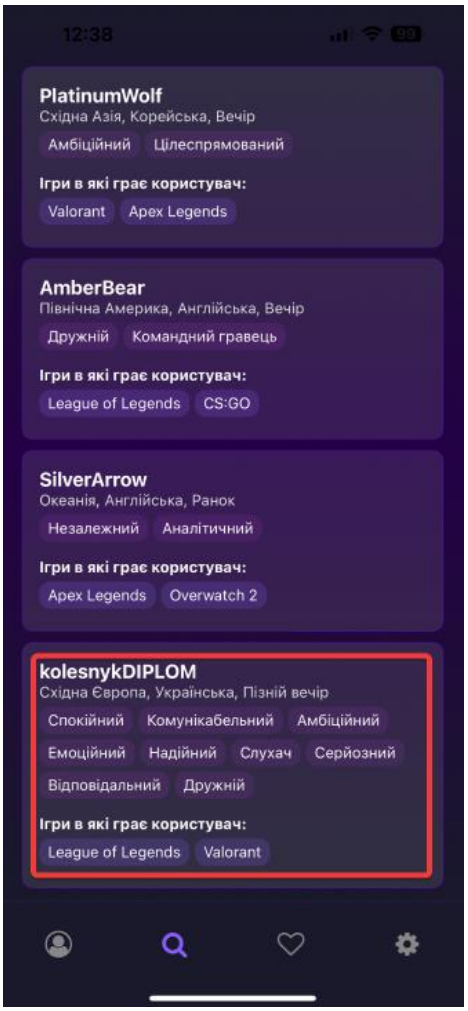

Рисунок 3.16 – Ранжування користувача-шукача в результатах пошуку

Після того як користувач переглянув список релевантних профілів, він має можливість перейти до профілю іншого користувача (рис. 3.17). У цьому профілі можна ознайомитися з детальнішою інформацією, зберегти профіль у вибране для подальшого доступу (рис. 3.18), а також зв’язатися з цією особою пізніше через вказані соціальні мережі або нікнейми в іграх, якщо такі були залишені.

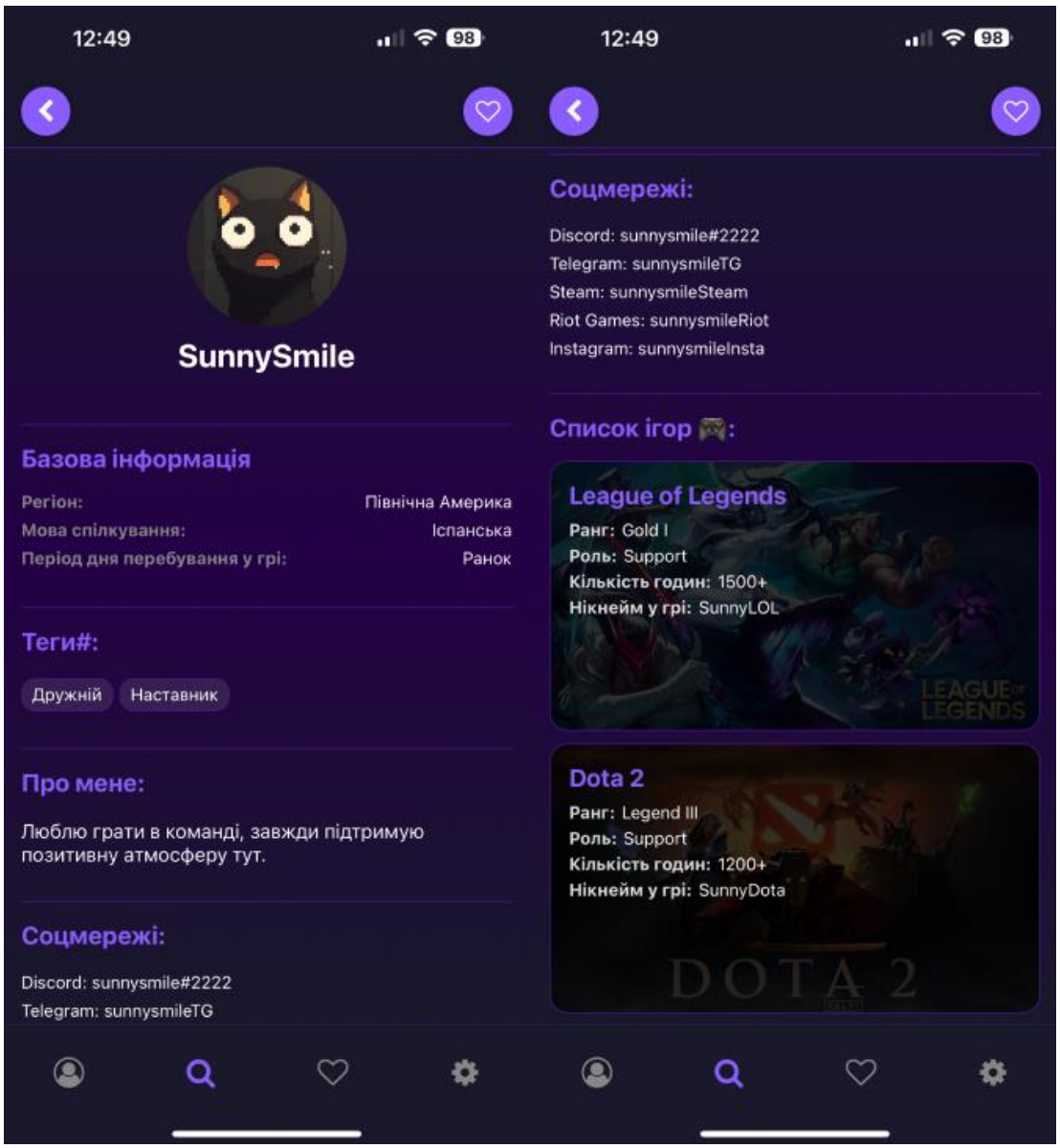

Рисунок 3.17 - Профіль іншого користувача

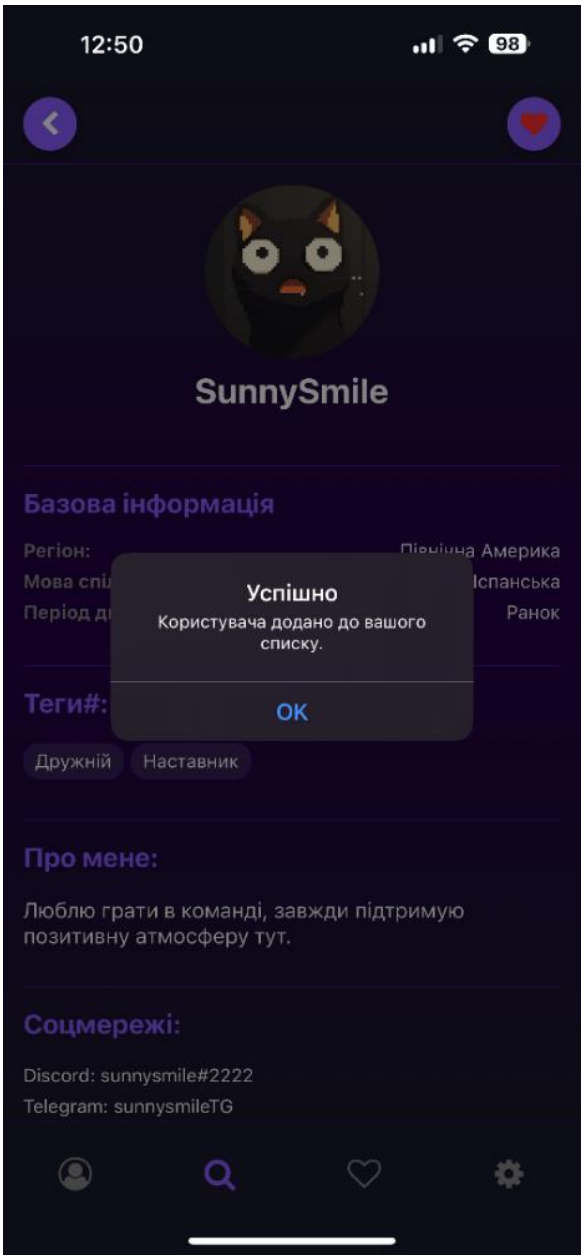

Рисунок 3.18 – Додавання користувача в список збережених

Після того як користувач був збережений, він автоматично додається до списку обраних і відображається на окремому екрані — екрані збережених користувачів (рис. 3.19). Цей екран дозволяє швидко повертатися до профілів, які зацікавили, без необхідності повторного пошуку. У цьому розділі користувач може переглядати основну інформацію про збережених осіб, переходити до їхніх повних профілів, а також за потреби видаляти їх зі списку. Така функціональність забезпечує зручність у використанні додатку, дає змогу формувати власну базу контактів і підтримувати зв'язок із релевантними гравцями або потенційними друзями для спільної гри.

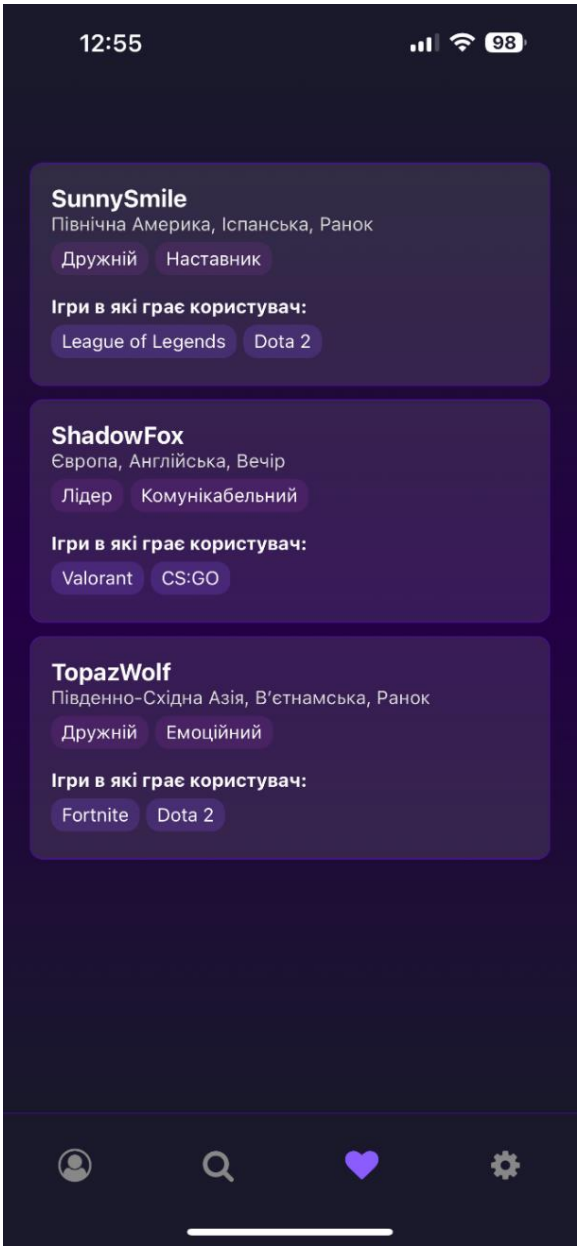

Рисунок 3.19 – Екран збережених користувачів

У разі, якщо користувач натискає на активне “сердечко” на профілі іншого користувача, воно стає неактивним, що означає видалення цього профілю зі списку збережених (рис. 3.20). Після цього на екрані з’являється повідомлення з підтвердженням, “Користувача було видалено з вашого списку”. Це повідомлення допомагає користувачу чітко зрозуміти, що дія виконана успішно, і забезпечує зворотний зв’язок у межах взаємодії з інтерфейсом.

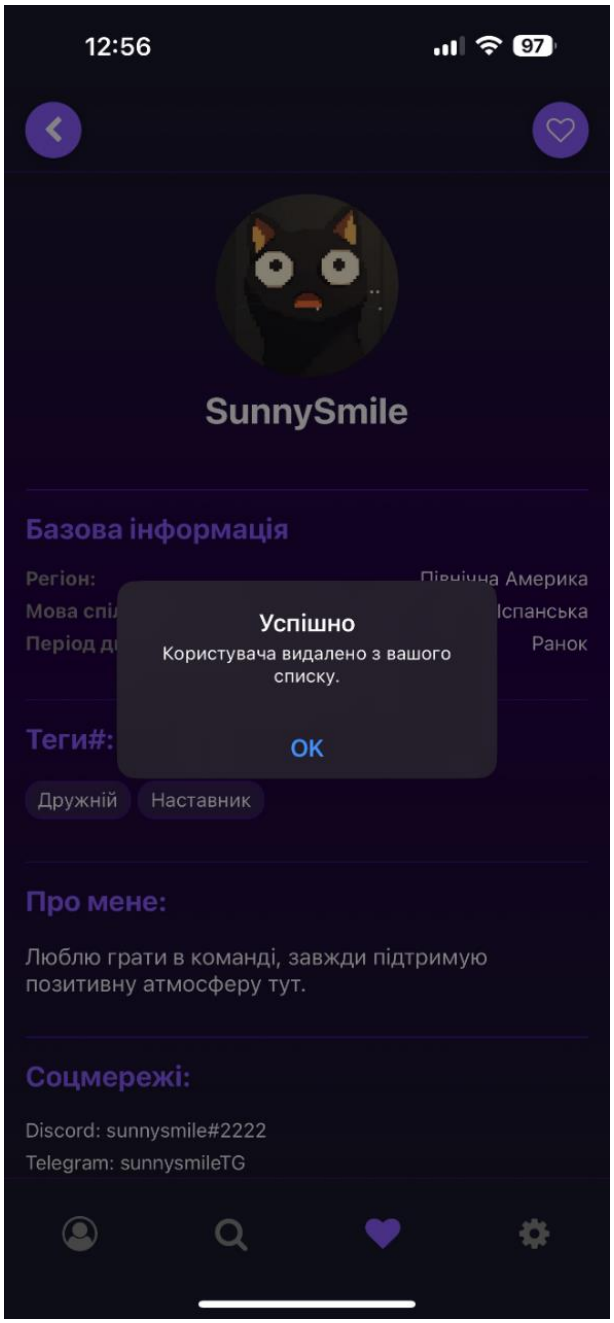

Рисунок 3.20 – Видалення користувача зі списку збережених

Після видалення користувача зі списку збережених, його профіль автоматично зникає зі списку без потреби в оновленні сторінки (рис. 3.21). Це забезпечує миттєвий візуальний зворотний зв’язок і підтверджує, що дія була виконана успішно. Завдяки цьому інтерфейс залишається динамічним і зручним для користувача, дозволяючи ефективно керувати своїм списком збережених партнерів.

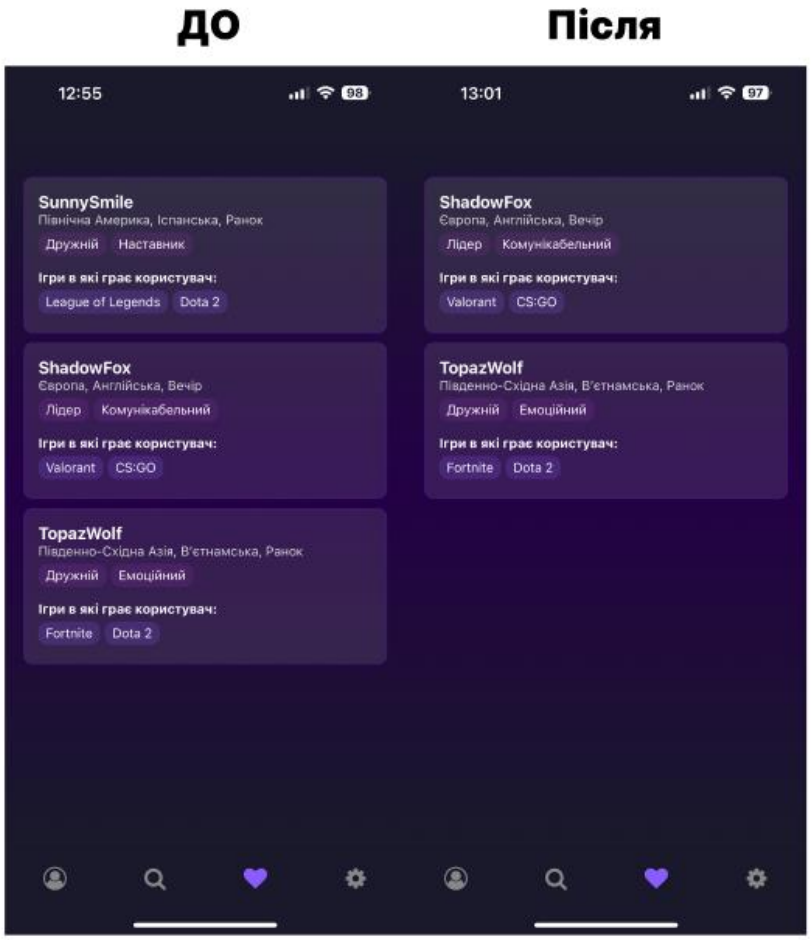

Рисунок 3.21 – Результати видалення користувача

На екрані налаштувань (рис. 3.22) користувачеві буде доступно кілька ключових функцій для керування своїм акаунтом та FAQ. Він може вимкнути показ свого профілю у пошуку, що дозволяє приховати себе від інших користувачів і зробити акаунт приватним (рис. 3.23). Користувач має можливість повністю видалити свій профіль, що призведе до стирання всіх його даних як у додатку, так і в системі аутентифікації. Також є опція виходу з профілю, яка завершить поточну сесію та поверне користувача на екран входу в додаток. Такий набір функцій забезпечує повний контроль над приватністю та безпекою акаунту, а також комфортну роботу з додатком.

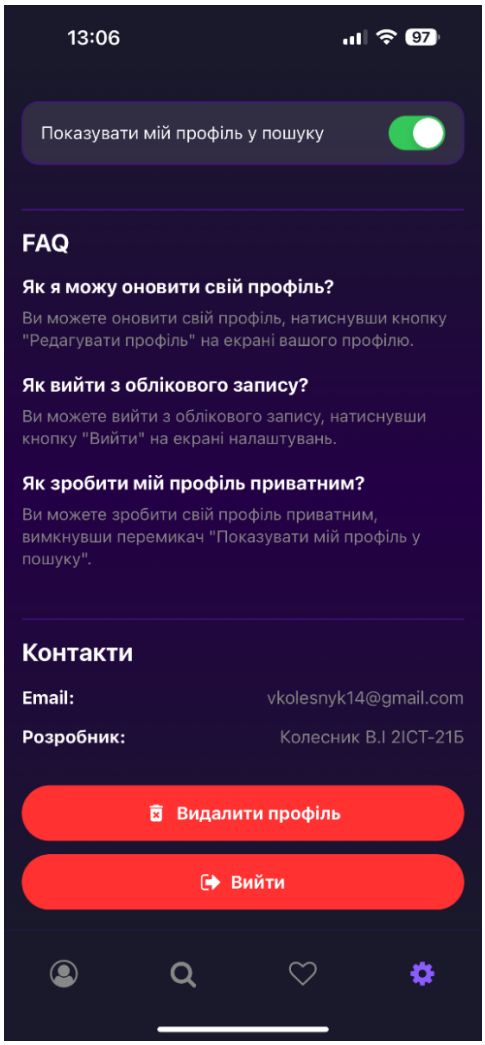

Рисунок 3.22 – Екран налаштувань

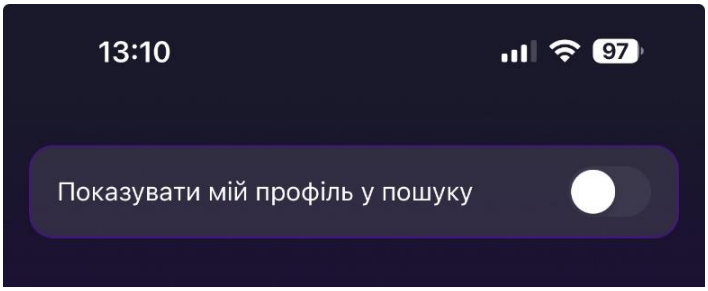

Рисунок 3.23 – Перемикач виключення себе з пошуку

Було проведено тестування функціоналу виключення користувача з пошуку (рис. 3.24). Після активації відповідного перемикача на екрані налаштувань, профіль користувача успішно приховувався від результатів пошуку інших користувачів. При цьому зміни коректно зберігалися в базі даних, а інтерфейс оновлювався в реальному часі, підтверджуючи, що функція працює стабільно і відповідає вимогам приватності.

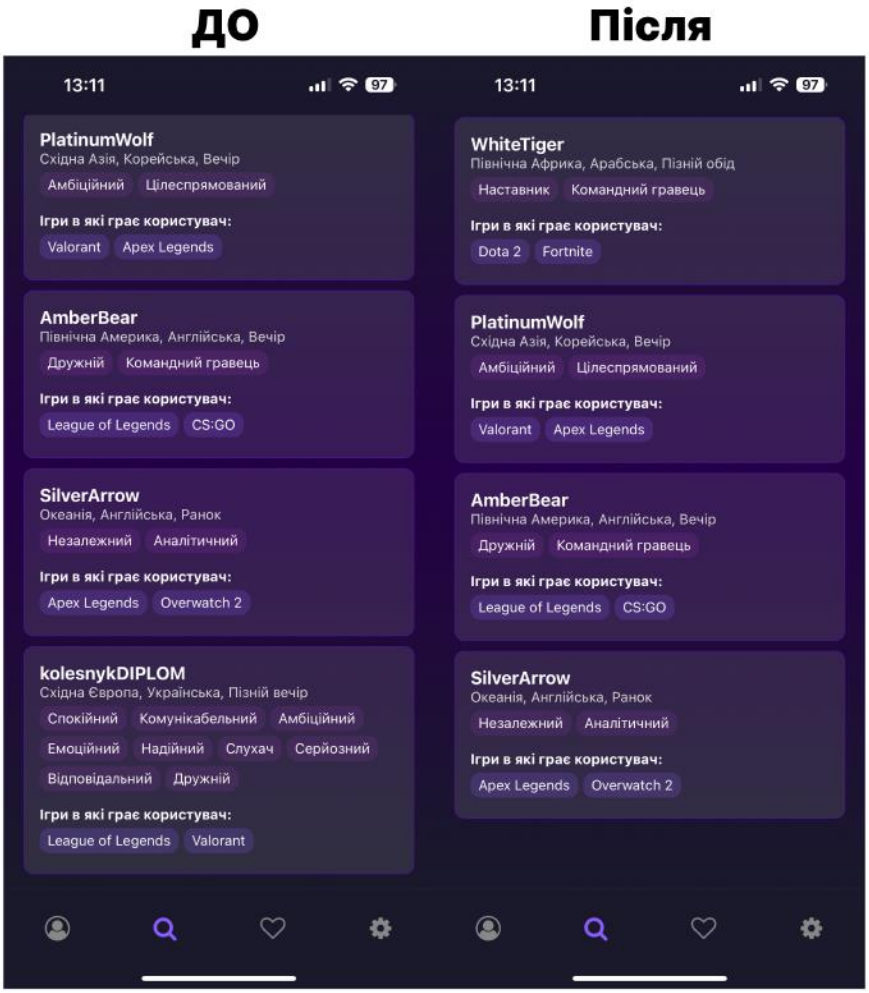

Рисунок 3.24 – Результат функції приватного профілю

Тестування розробленої інформаційно-пошукової системи на базі мобільного застосунку показало, що всі основні та допоміжні функції коректно працюють на платформах iOS та Android. Це свідчить про високу якість реалізації, стабільність і сумісність додатку з різними операційними системами та пристроями. Під час тестування було перевірено не лише базову функціональність, а й сумісність із різними версіями ОС, роздільною здатністю екранів, а також продуктивність і безпеку інформаційно-пошукової системи.

Була здійснена перевірка коректності інтерфейсу, бізнес-логіки, інтеграції з базою даних і зовнішніми сервісами. Використання емуляторів та тестування на реальних пристроях дозволило виявити й усунути потенційні проблеми, пов’язані з різноманітністю апаратних конфігурацій і умов експлуатації. Крім того, було проведено тестування продуктивності, щоб переконатися в ефективному використанні ресурсів пристрою та швидкій реакції інтерфейсу.

Завдяки комплексному підходу до тестування інформаційно-пошукової системи, який забезпечує стабільну роботу, зручний користувацький досвід і відповідає сучасним стандартам якості мобільних застосунків, що є критично важливим для інформаційно-пошукової системи вибору партнерів у відеоіграх.

3.5 Перспективи розвитку інформаційно-пошукової системи GPFI

Подальший розвиток інформаційно-пошукової системи після успішної реалізації основного функціоналу передбачає впровадження низки важливих покращень і нових можливостей, що розширять функціонал, підвищать зручність користування та розширять аудиторію.

Планується інтеграція входу через популярні соціальні мережі, що спростить реєстрацію та підвищить зручність користувачів. Також розглядається впровадження біометричних методів аутентифікації (відбиток пальця, розпізнавання обличчя) для швидкого та безпечного доступу. Для підвищення безпеки облікових записів буде реалізовано підтвердження електронної пошти та скидання пароля.

Управління профілем користувача отримає індикатор завершеності, який стимулюватиме надання повної інформації, а також систему верифікаційних значків для підтвердження даних і ігрових досягнень, що підвищить довіру між гравцями. Додасться історія активності профілю для кращого аналізу взаємодії користувача з системою.

Пошуковий модуль буде суттєво вдосконалено: з’являться більш точні фільтри, наприклад, за часовими поясами та ігровими подіями, функція історії пошуку для швидкого повернення до попередніх запитів, а також пошук за географічною близькістю для організації локальних ігрових спільнот. Оновлення результатів у реальному часі підвищить актуальність пошуку.

Важливим нововведенням стане вбудований чат, що дозволить користувачам спілкуватися безпосередньо в додатку після знаходження партнерів. Для цього планується інтеграція сервісів обміну повідомленнями в

реальному часі, таких як Cloud Firestore, що забезпечить швидку та надійну комунікацію [27].

Архітектурно передбачено перехід від локальної бази SQLite до хостингової бази даних, що підвищить продуктивність, масштабованість і надійність зберігання даних, дозволяючи системі обробляти більші обсяги інформації та збільшувати кількість користувачів без втрати швидкодії. Розглядається використання хмарних рішень, зокрема Cloud Firestore.

Для підвищення видимості системи в інтернеті та магазинах додатків планується впровадження SEO (Search Engine Optimization) та ASO (App Store Optimization). Створення оптимізованого веб-сайту з використанням мікророзмітки ld-json, релевантних метатегів, семантичного ядра, а також підключення Google Search Console і Google Analytics 4 допоможе залучити нову аудиторію. ASO-оптимізація включатиме роботу над назвою, описами, ключовими словами, скріншотами та відеопрев'ю додатку для підвищення позицій у пошуку магазинів і збільшення конверсії встановлень.

Загалом, ці заходи дозволять інформаційно-пошуковій системі GPFI стати більш функціональною, зручною та масштабованою платформою для пошуку партнерів і формування ігрових спільнот, а також суттєво зміцнять її присутність і впізнаваність на ринку.

3.6 Висновки

У цьому розділі детально описано реалізацію інформаційно-пошукової системи та результати її тестування. Розроблення здійснювалось на базі React Native та Expo, з акцентом на локальну роботу з даними (SQLite та AsyncStorage) для максимальної приватності та автономності, а Firebase Authentication використовувався виключно для безпечної аутентифікації. Архітектура системи є модульною, що підтверджується діаграмами Use Case та блок-схемою функціонування.

Детально описано реалізовані модулі:

- Модуль аутентифікації (LoginScreen), забезпечує безпечну реєстрацію та вхід, з коректною обробкою помилок.
- Модуль налаштування профілю (SetupScreen), дозволяє створювати та редагувати деталізовані профілі користувачів, включно з іграми, рангами, ролями, тегами та контактними даними.
- Модуль пошуку та фільтрації (SearchScreen), є центральним компонентом, що надає потужні інструменти для знаходження партнерів за комплексними критеріями (нікнейм, регіон, мова, час, теги, ігрові параметри). Пошук здійснюється через оптимізовані SQL-запити до локальної БД.
- Модуль збережених профілів (SavedScreen), дозволяє керувати списком обраних партнерів, забезпечуючи швидкий доступ та миттєве оновлення даних.
- Модуль власного профілю (ProfileScreen), відображає інформацію про користувача та надає доступ до основної навігації.
- Модуль налаштувань (SettingsScreen), включає функціонал приватного профілю, видалення облікового запису та виходу із системи.

Особливу увагу приділено оптимізації програмного забезпечення, зокрема оптимізації бази даних SQLite, параметризовані запити, пошуку та фільтрації, інтерфейсу користувача ScrollView, управління станами компонентів, навігації, а також обробці помилок.

Розглянуто забезпечення безпеки системи на всіх рівнях: захист аутентифікації Firebase, захист локальної бази даних SQLite, валідація, параметризовані запити, захист даних користувача, пошукових запитів, навігації та конфігурації.

Тестування системи, проведене на платформах iOS та Android через Expo Go, емулятори та реальні пристрої, підтвердило коректне функціонування всіх реалізованих модулів, стабільну роботу інтерфейсу на різних розмірах екранів, ефективну обробку помилок та відповідність вимогам приватності. Числових показників ефективності у цьому розділі не наведено, але підтверджено загальну працездатність.

Також було окреслено ключові напрямки подальшого розвитку інформаційно-пошукової системи GPFI. Планується розширити функціонал за рахунок інтеграції соціальних мереж та біометричної аутентифікації, вдосконалення профілю користувача, а також суттєвого покращення пошукового модуля. Важливим кроком є інтеграція вбудованого чату для комунікації в додатку. З архітектурної точки зору передбачається перехід на хостингову базу даних для забезпечення масштабованості та продуктивності. Для підвищення видимості та залучення аудиторії будуть впроваджені SEO та ASO оптимізації. Зазначені заходи дозволять GPFI стати більш функціональною, зручною та масштабованою платформою, що зміцнить її позиції на ринку.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було проведено детальний аналіз предметної області, пов’язаної з пошуком партнерів у багатокористувацьких відеоіграх, було виявлено значні функціональні та якісні недоліки існуючих внутрішньоігрових механізмів та сторонніх платформ. Ці обмеження, зокрема відсутність комплексних спеціалізованих рішень з глибоким підбором за розширеними критеріями, обґрунтувано актуальність та необхідність розробки нової інформаційно-пошукової системи GPFI.

Вибір технологічного стеку, що включає JavaScript та кросплатформний фреймворк React Native у поєднанні з Expo, виявився оптимальним рішенням для реалізації мобільного додатку. Цей вибір забезпечив високу швидкість та ефективність розробки, широку сумісність із різноманітними мобільними пристроями та зручність подальшої підтримки й масштабування системи. Обґрунтованість такого підходу базувалася на ретельному порівнянні критеріїв продуктивності, легкості інтеграції компонентів, наявності документації та активної спільноти підтримки. Ключовим архітектурним рішенням стало використання локальної бази даних SQLite для зберігання інформації про користувачів та їхні ігрові профілі. Такий підхід гарантує автономність роботи додатку, високий рівень конфіденційності даних користувачів та незалежність від постійного мережевого підключення, що є значною перевагою над онлайн-орієнтованими аналогами. Firebase, у свою чергу, цілеспрямовано застосовується виключно для аутентифікації, забезпечуючи надійний та безпечний вхід до системи без компромісів щодо локального зберігання профілів.

Розроблена інформаційно-пошукова система GPFI демонструє високу ефективність у знаходженні ігрових партнерів за широким спектром критеріїв, включаючи ігрові характеристики, особисті теги, регіони та мовні уподобання, а також бажаний період гри зі своїм майбутнім партнером. Висока швидкодія та точність підбору досягаються за рахунок оптимізованих алгоритмів підбору та фільтрації, реалізованих через ефективні SQL-запити до локальної бази даних та раціональне управління станом інтерфейсу. Обґрунтованість рішень щодо

логічної структури бази даних та алгоритмів підтверджується стабільною роботою системи. Модульність та масштабованість архітектури додатку, що були спроектовані за допомогою UML-діаграм, дозволяють легко розширювати її функціонал. Надійність функціонування системи підтверджена ретельним тестуванням, а її безпека забезпечена комплексною валідацією даних та застосуванням параметризованих запитів, що успішно запобігає загрозам SQL-ін’єкцій. Зручність користування системою забезпечується інтуїтивно зрозумілим інтерфейсом з адаптивним дизайном та плавною навігацією, що покращує загальний користувацький досвід, що впливає на оптимізацію ASO.

Основна цінність розробленої системи GPFI полягає у значному спрощенні та оптимізації процесу пошуку ігрових партнерів, що суттєво знижує часові витрати користувачів і сприяє формуванню більш сумісних та ефективних ігрових спільнот.

Основним обмеженням поточної реалізації є відсутність вбудованої онлайн-синхронізації профілів, що може обмежувати можливості обміну даними між пристроями без ручного експорту або імпорту.

Подальший розвиток GPFI передбачає розширення функцій аутентифікації, включаючи інтеграцію з соціальними мережами та біометрією, покращення управління профілем із додаванням нових полів і верифікаційних значків, впровадження вбудованого чату для безпосередньої комунікації між користувачами, перехід на хостингову базу даних для централізованого зберігання та синхронізації профілів, а також оптимізацію SEO (Search Engine Optimization) та ASO (App Store Optimization) для розширення аудиторії та підвищення впізнаваності системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Колесник В. І., Козачко О. М., Горячев Г. В. Мобільний додаток пошуку партнерів у багатокористувацьких відеоіграх // LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24337 (дата звернення: 23.03.2025).

2. What does LFG mean?. URL: https://brandmentions.com/wiki/What_does_LFG_mean%3F (дата звернення: 19.05.2025).

3. Мобільний застосунок GamerLink. URL: https://play.google.com/store/apps/details/GamerLink_LFG_Teams_Friends?id=gg.gamerlink.gamerlink&hl=uk&pli=1 (дата звернення: 19.05.2025).

4. Веб-платформа LoL Finder. URL: https://lolfinder.net/en (дата звернення: 19.05.2025).

5. Платформа Discord. URL: https://discord.com (дата звернення: 19.05.2025).

6. Véron M., Marin O., Monnet S. Matchmaking in multi-player on-line games: Studying user traces to improve the user experience // *Proceedings of the 24th International Workshop on Network and OS Support for Digital A/V (NOSSDAV 2014)*, Singapore, March 2014. DOI: 10.1145/2578260.2578265. (дата звернення: 19.05.2025).

7. Firebase Authentication. URL: https://firebase.google.com/products/auth (дата звернення: 19.05.2025).

8. Ojha A. Advanced Guide to ASO (APP Store Optimization) with Digital Marketing. Bilaspur : Evincepub Publishing, 2020. 164 с. URL: https://books.google.com.ua/books?id=UR7cEAAAQBAJ&printsec=frontcover&hl=en#v=onepage&q&f=false (дата звернення: 19.05.2025).

9. Expo SQLite Documentation. URL: https://docs.expo.dev/versions/latest/sdk/sqlite/ (дата звернення: 19.05.2025).

10. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. Berkeley : New Riders, 2014. 216 с. URL: https://www.google.com.ua/books/edition/Don_t_Make_Me_Think_Revisited/QlduAgAAQBAJ?hl=en&gbpv=0 (дата звернення: 20.05.2025).

11. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 687 p. URL: https://books.google.com/books?id=b5G4zAEACAAJ (дата звернення: 20.05.2025).

12. React Native Documentation. URL: https://reactnative.dev/ (дата звернення: 20.05.2025).

13. Expo Documentation. URL: https://docs.expo.dev/ (дата звернення: 20.05.2025).

14. Owens M., Allen G. The Definitive Guide to SQLite. Berkeley : Apress, 2010. 464 с. URL: https://books.google.com/books?id=VsZ5bUh0XAkC (дата звернення: 20.05.2025).

15. Chatzigeorgiou A., Manakos A. Investigating the evolution of bad smells in object-oriented code // Proceedings of the 7th International Conference on the Quality of Information and Communications Technology (QUATIC), 29 Sept – 2 Oct 2010, Porto, Portugal. Porto : IEEE, 2010. P. 106–115. DOI: https://doi.org/10.1109/QUATIC.2010.16. (дата звернення: 21.05.2025).

16. AsyncStorage Documentation. URL: https://reactnative.dev/docs/asyncstorage (дата звернення: 21.05.2025).

17. Глонь О. Застосування UML для проектування інтелектуальних систем // Контроль та управління в складних системах : матеріали Міжнар. наук.-техн. конф. (КУСС-2005), 24–27 трав. 2005 р., м. Дніпро. Дніпро : ДНУ, 2005. С. 15.

18. Visual Studio Code. Getting started. URL: https://code.visualstudio.com/docs (дата звернення: 21.05.2025).

19. Run apps on the Android Emulator. URL: https://developer.android.com/studio/run/emulator (дата звернення: 21.05.2025).

20. Додаток Expo Go. URL: https://play.google.com/store/apps/details?id=host.exp.exponent&hl=en (дата звернення: 21.05.2025).

21. Малахов Є. В. Основи проектування баз даних. Видавництво АО БАХВА. 153 с. URL: https://www.google.com.ua/books/edition/Основи_проектування_б/JosfyGdYX2oC?hl=en&gbpv=0&kptab=overview (дата звернення: 21.05.2025).

22. Kreibich J. A. Using SQLite. O'Reilly Media, 2010. 530 с. URL: https://www.google.com.ua/books/edition/Using_SQLite/v5OYlkt6uKYC?hl=en&gbpv=1 (дата звернення: 21.05.2025).

23. Diagram of use cases (UseCase diagram). URL: https://flexberry.github.io/en/fd_use-case-diagram.html (дата звернення: 26.05.2025).

24. Техніки оптимізації бази даних. URL: https://data-life-ua.com/db/11-database-optimization-techniques/ (дата звернення: 26.05.2025).

25. Control flow and error handling. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling (дата звернення: 26.05.2025).

26. Ultimate Firebase for iOS and Android Applications: Leverage Firebase's Full Suite to Craft Secure, Scalable and High-Performance Apps Across iOS and Android Platforms. Orange Education Pvt Ltd, 2024. 151 p. URL: https://www.google.com.ua/books/edition/Ultimate_Firebase_for_iOS_and_Android_Ap/Ih4tEQAAQBAJ?hl=en&gbpv=0 (дата звернення: 26.05.2025).

27. Cloud Firestore. URL: https://firebase.google.com/docs/firestore (дата звернення: 26.05.2025).

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА ВИБОРУ ПАРТНЕРІВ У БАГАТОКОРИСТУВАЦЬКИХ ВІДЕОІГРАХ

08-34.БКР.010.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Олексій КОЗАЧКО
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Валентин КОЛЕСНИК
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) React Native, навчальний посібник. URL: https://reactnative.dev/

2) Expo Documentation. URL: https://docs.expo.dev/

3) Firebase Authentication. URL: https://firebase.google.com/products/auth

4) Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 687 p.

3. Мета і призначення роботи.

Метою роботи є розроблення та дослідження інформаційно-пошукової системи вибору партнерів у багатокористувацьких відеоіграх на базі мобільного додатку.

4. Вихідні дані для проведення робіт:

Локальна база даних SQLite, що містить дані про профілі користувачів та їхні ігрові характеристики; платформа React Native в середовищі розробки Visual Studio Code з даними про ігрові профілі гравців та профілі користувачів

5. Методи дослідження:

Дослідження існуючих систем для пошуку ігрових партнерів; аналіз предметної області та формування вимог; розроблення UML-діаграм; проєктування структури локальної бази даних; реалізація функціоналу пошуку, фільтрації та управління профілями; інтеграція сервісу Firebase Authentication; тестування функціоналу інформаційно-пошукової системи.

Етапи роботи і терміни їх виконання:

a) Аналіз предметної області та вимог до системи _____ – _____
b) Проєктування архітектури системи та бази даних _____ – _____
c) Вибір оптимальних інформаційних технологій _____ – _____
d) Розроблення інформаційно-пошукової системи _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання функціональної інформаційно-пошукової системи автоматизації процесу підбору та фільтрації партнерів у відеоіграх.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Валентин КОЛЕСНИК

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційно-пошукова система вибору партнерів у багатокористувацьких відеоіграх»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,74 %

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:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ

Здобувач _____ (підпис) Валентин КОЛЕСНИК

Додаток В

(довідниковий)

Фрагмент лістингу програми

Код файлу додатку App.js

```
import { useEffect } from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { FontAwesome } from "@expo/vector-icons";
import LoginScreen from "./screens/LoginScreen";
import SetupScreen from "./screens/SetupScreen";
import ProfileScreen from "./screens/ProfileScreen";
import SearchScreen from "./screens/SearchScreen";
import SettingsScreen from "./screens/SettingsScreen";
import ViewProfileScreen from "./screens/ViewProfileScreen";
import SavedScreen from "./screens/SavedScreen";

const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();

const SearchStack = createNativeStackNavigator();

function SearchStackNavigator() {
 return (
  <SearchStack.Navigator
   screenOptions={{
    headerShown: false,
    contentStyle: { backgroundColor: "transparent" },
   }}
  >
   <SearchStack.Screen name="Search" component={SearchScreen} />
```

```
      <SearchStack.Screen
        name="ViewProfile"
        component={ViewProfileScreen}
        options={{
          gestureEnabled: false
        }}
      />
    </SearchStack.Navigator>
  );
}

const SavedStack = createNativeStackNavigator();

function SavedStackNavigator() {
  return (
    <SavedStack.Navigator
      screenOptions={{
        headerShown: false,
        contentStyle: { backgroundColor: "transparent" },
      }}
    >
      <SavedStack.Screen name="Saved" component={SavedScreen} />
      <SavedStack.Screen
        name="ViewProfile"
        component={ViewProfileScreen}
        options={{
          gestureEnabled: false
        }}
      />
    </SavedStack.Navigator>
  );
}
```

```
function MainTabs() {
  useEffect(() => {
    console.log("MainTabs rendered");
  }, []);

  return (
    <Tab.Navigator
      screenOptions={({ route }) => ({
        headerShown: false,
        tabBarShowLabel: false,
        tabBarStyle: {
          backgroundColor: "#19192B",
          borderTopColor: "#4A148C",
          justifyContent: 'center',
          alignItems: 'center',
          height: 100,
          paddingTop: 20,
        },
        tabBarIcon: ({ focused, color, size }) => {
          let iconName;

          if (route.name === "Профіль") {
            iconName = focused ? "user-circle" : "user-circle-o";
          } else if (route.name === "Пошук") {
            iconName = focused ? "search" : "search";
          } else if (route.name === "Налаштування") {
            iconName = focused ? "cog" : "cog";
          } else if (route.name === "Збережені") {
            iconName = focused ? "heart" : "heart-o";
          }

          return <FontAwesome name={iconName} size={size} color={color} />;
        },
```

```
        tabBarActiveTintColor: "#8B5DFF",
        tabBarInactiveTintColor: "#888",
      })}
    >
      <Tab.Screen name="Профіль" component={ProfileScreen} />
      <Tab.Screen name="Пошук" component={SearchStackNavigator} />
      <Tab.Screen name="Збережені" component={SavedStackNavigator} />
      <Tab.Screen name="Налаштування" component={SettingsScreen} />
    </Tab.Navigator>
  );
}

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        initialRouteName="Login"
        screenOptions={{
          headerShown: false,
          contentStyle: { backgroundColor: "transparent" },
          gestureEnabled: false,
        }}
      >
        <Stack.Screen
          name="Login"
          component={LoginScreen}
          options={{
            gestureEnabled: false,
          }}
        />
        <Stack.Screen name="Setup" component={SetupScreen} />
        <Stack.Screen name="Main" component={MainTabs} />
      </Stack.Navigator>
```

```
    </NavigationContainer>
  );
}
///////////////
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";

const firebaseConfig = {
  apiKey: "AIzaSyAtnhNfFpAyGsHr6k8luHkmi0X_cTV9tz0",
  authDomain: "gpfi-cf949.firebaseapp.com",
  projectId: "gpfi-cf949",
  storageBucket: "gpfi-cf949.firebasestorage.app",
  messagingSenderId: "594689127995",
  appId: "1:594689127995:web:d5bf0cf24d35ae86cace1e",
};
const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export default app;

///////////////
const { getDefaultConfig } = require('@expo/metro-config');
const config = getDefaultConfig(__dirname);
config.resolver.sourceExts.push('cjs');
config.resolver.unstable_enablePackageExports = false;
module.exports = config;
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА ВИБОРУ ПАРТНЕРІВ У БАГАТОКОРИСТУВАЦЬКИХ ВІДЕОІГРАХ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

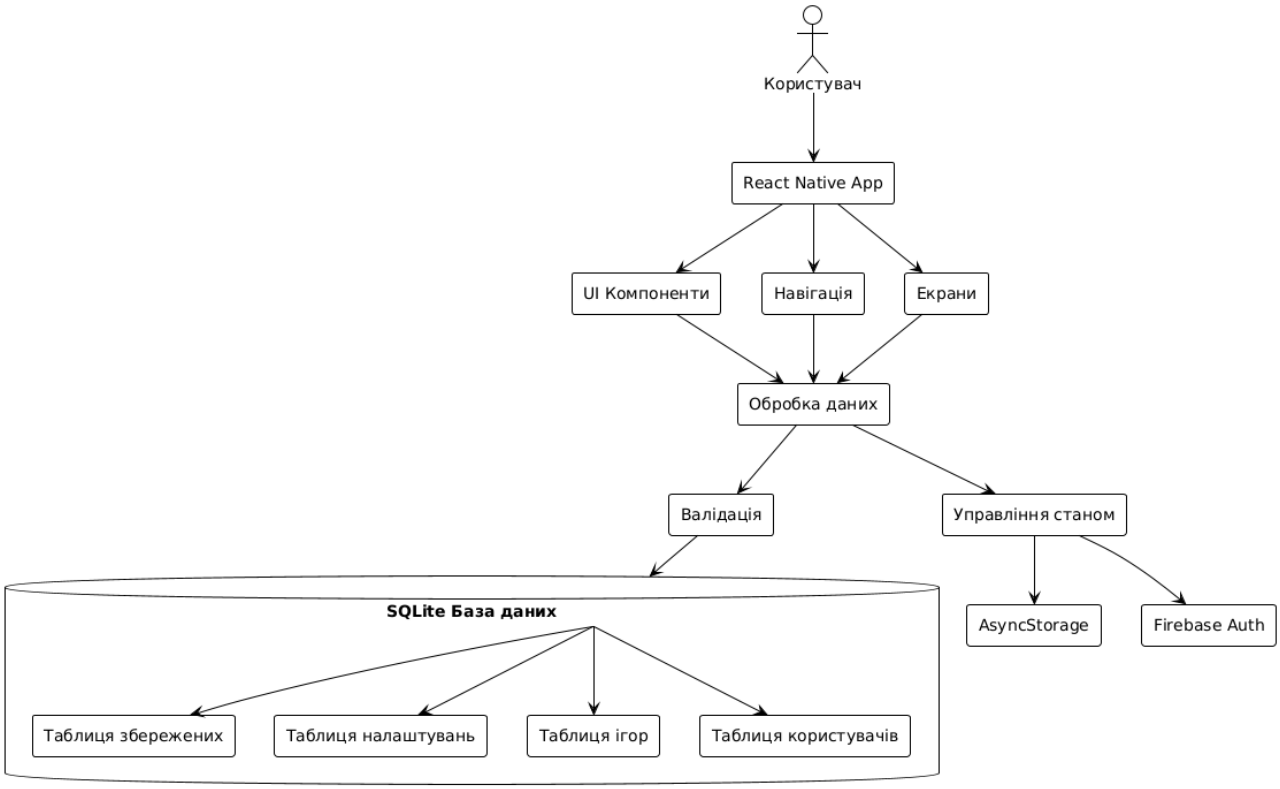

Рисунок Г.1 – Діаграма архітектури інформаційно-пошукової системи

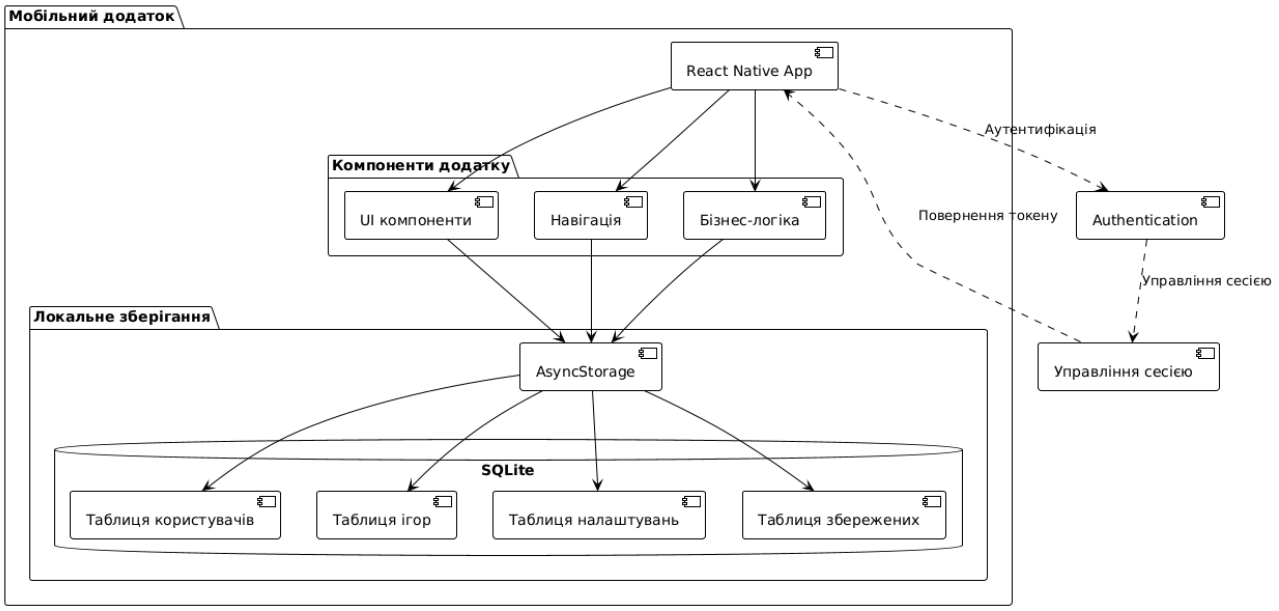

Рисунок Г.2 – Діаграма розгортання системи

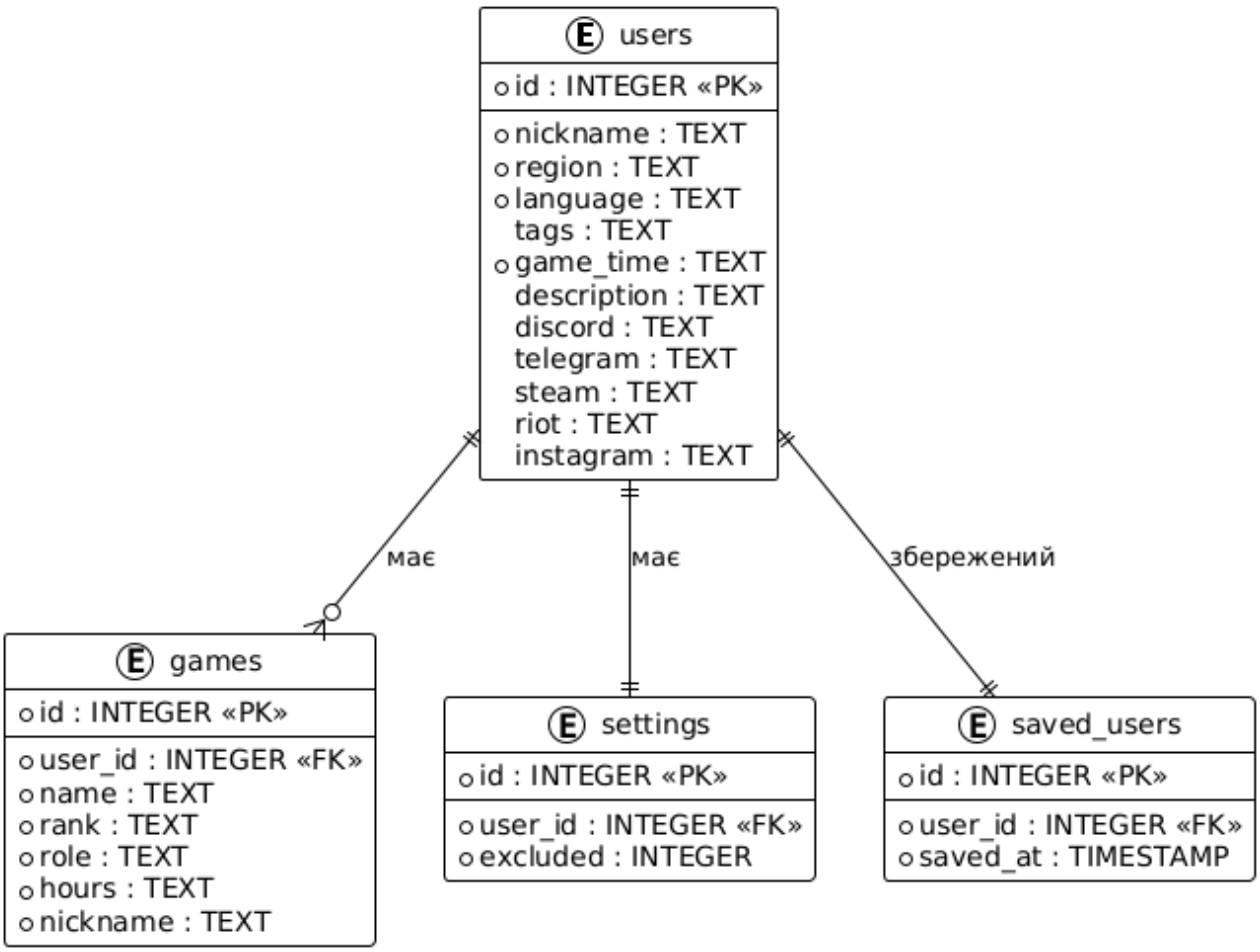

Рисунок Г.3 – Діаграма "сутність-зв'язок" бази даних

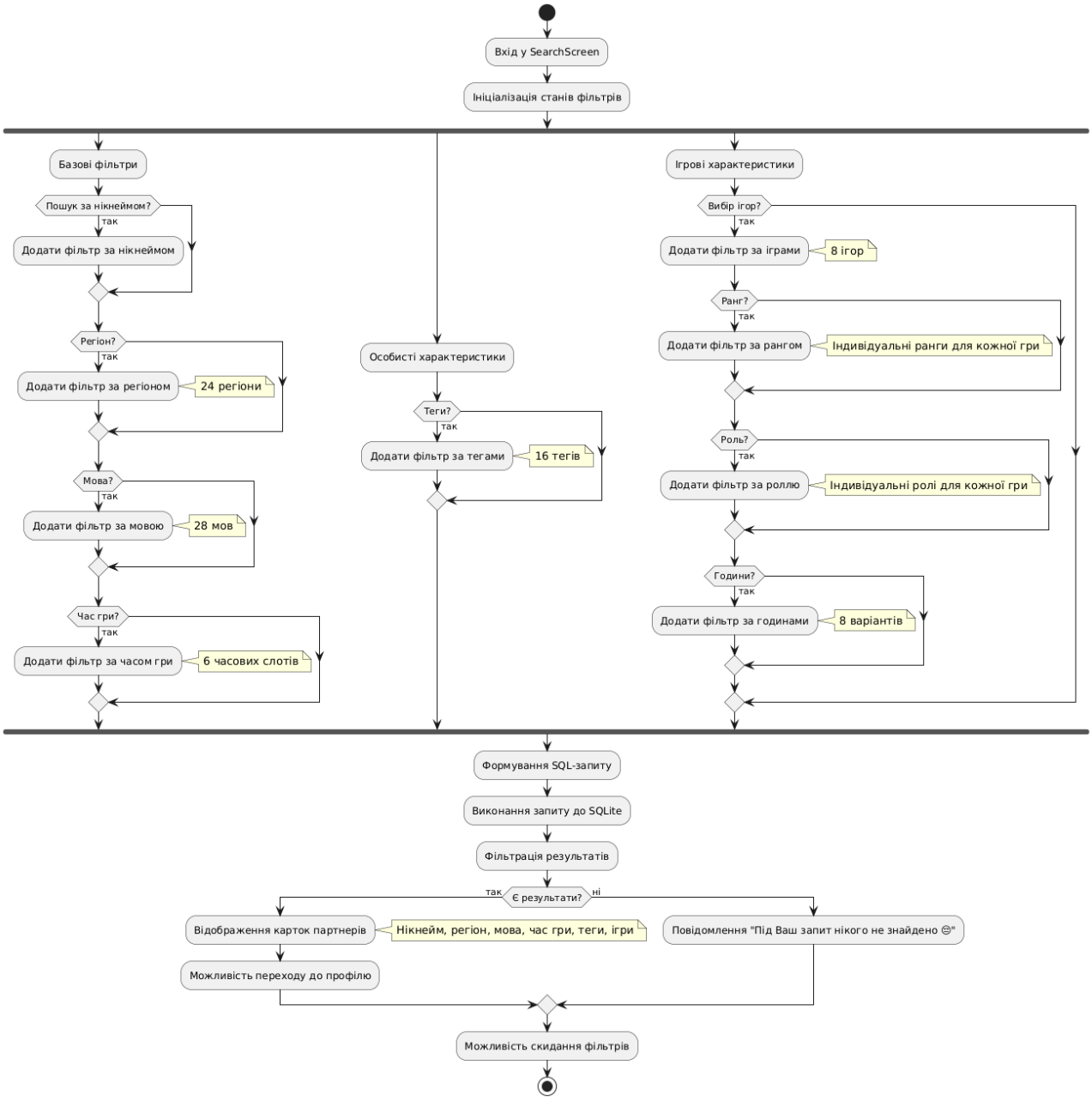

Рисунок Г.4 – Блок-схема алгоритму підбору та фільтрації користувачів

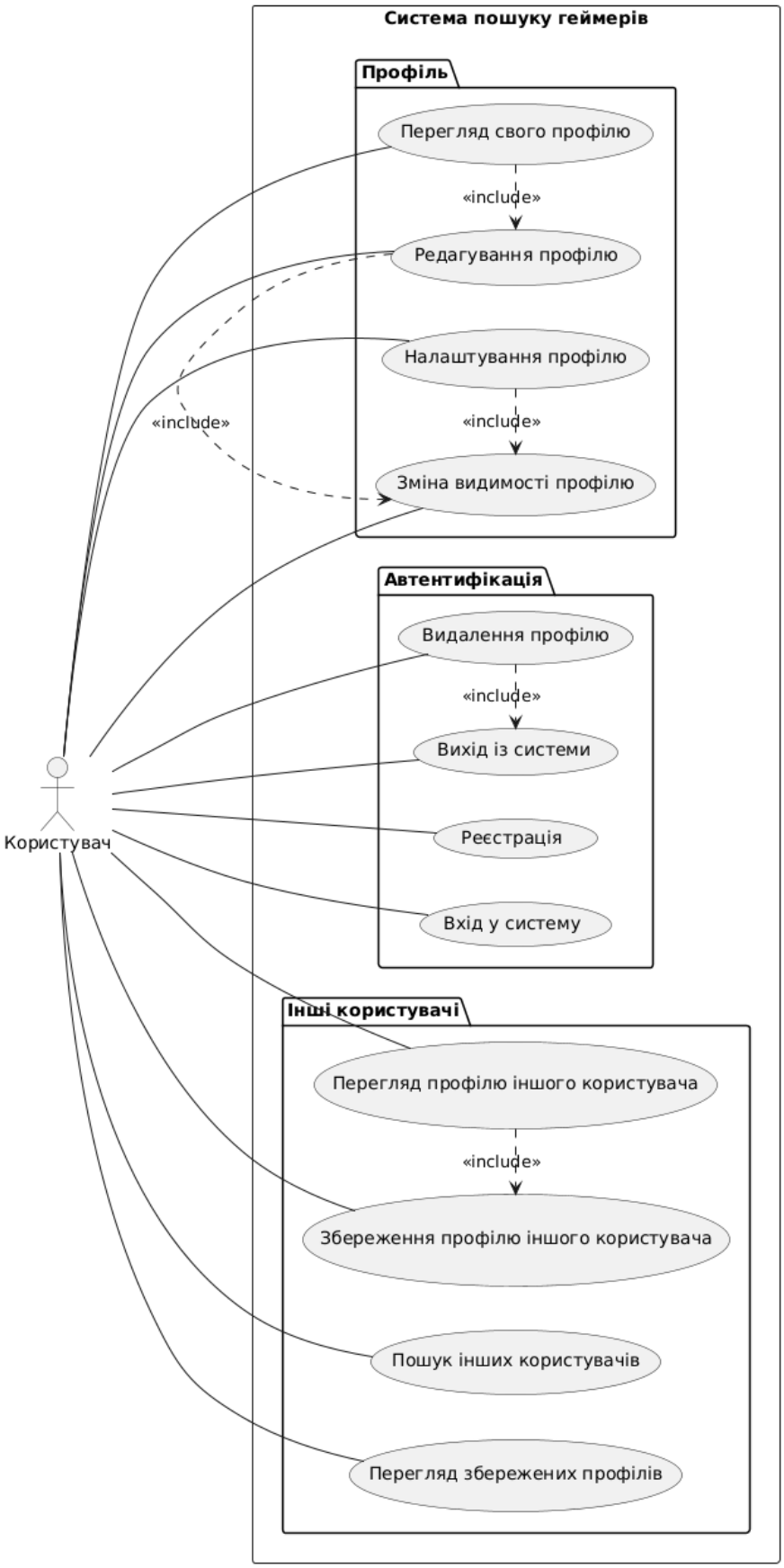

Рисунок Г.5 - Діаграма USE CASE інформаційно-пошукової системи

Рисунок Г.6 - Блок-схема функціонування інформаційно-пошукової системи

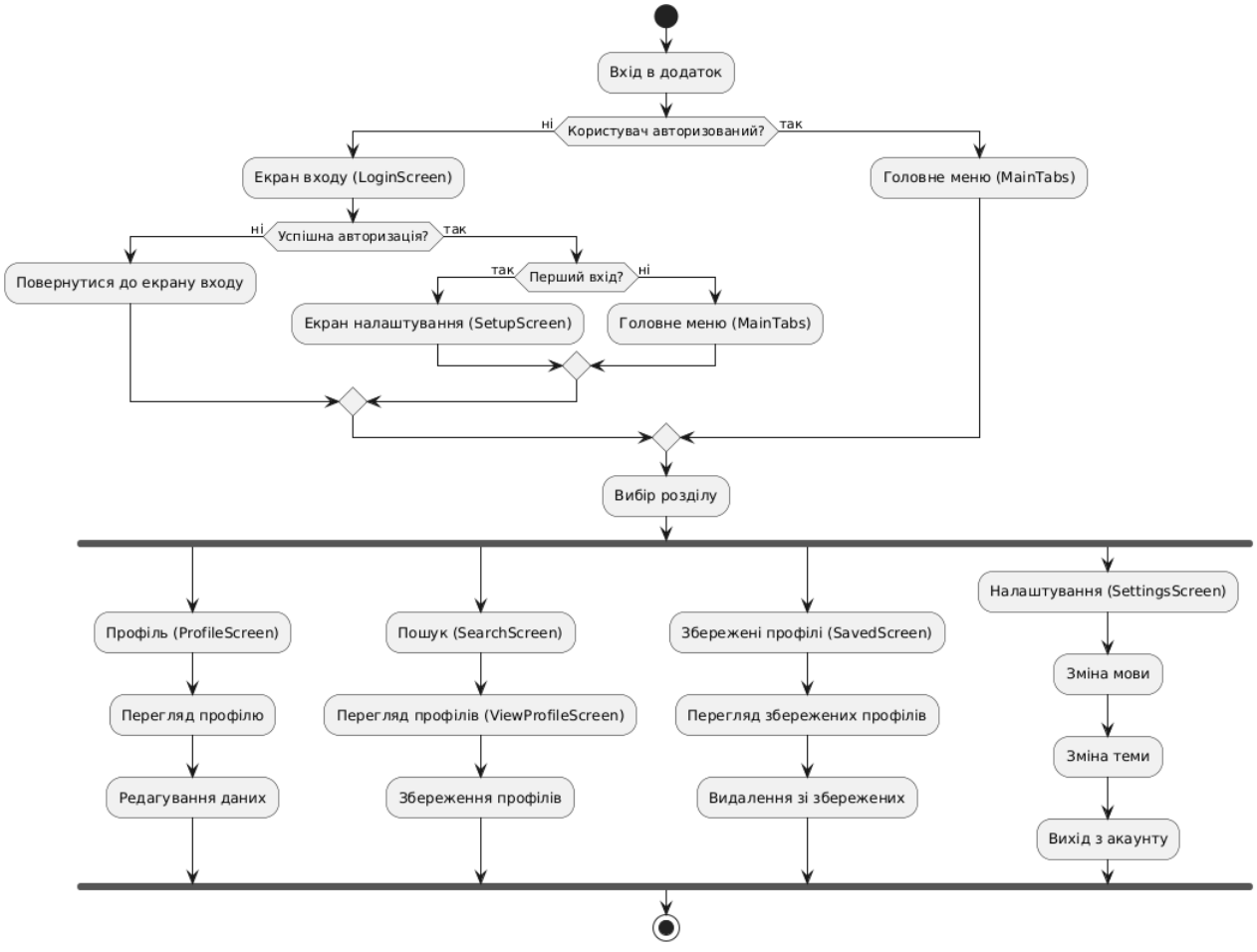

Рисунок Г.7 – Блок-схема алгоритм роботи інформаційно-пошукової системи

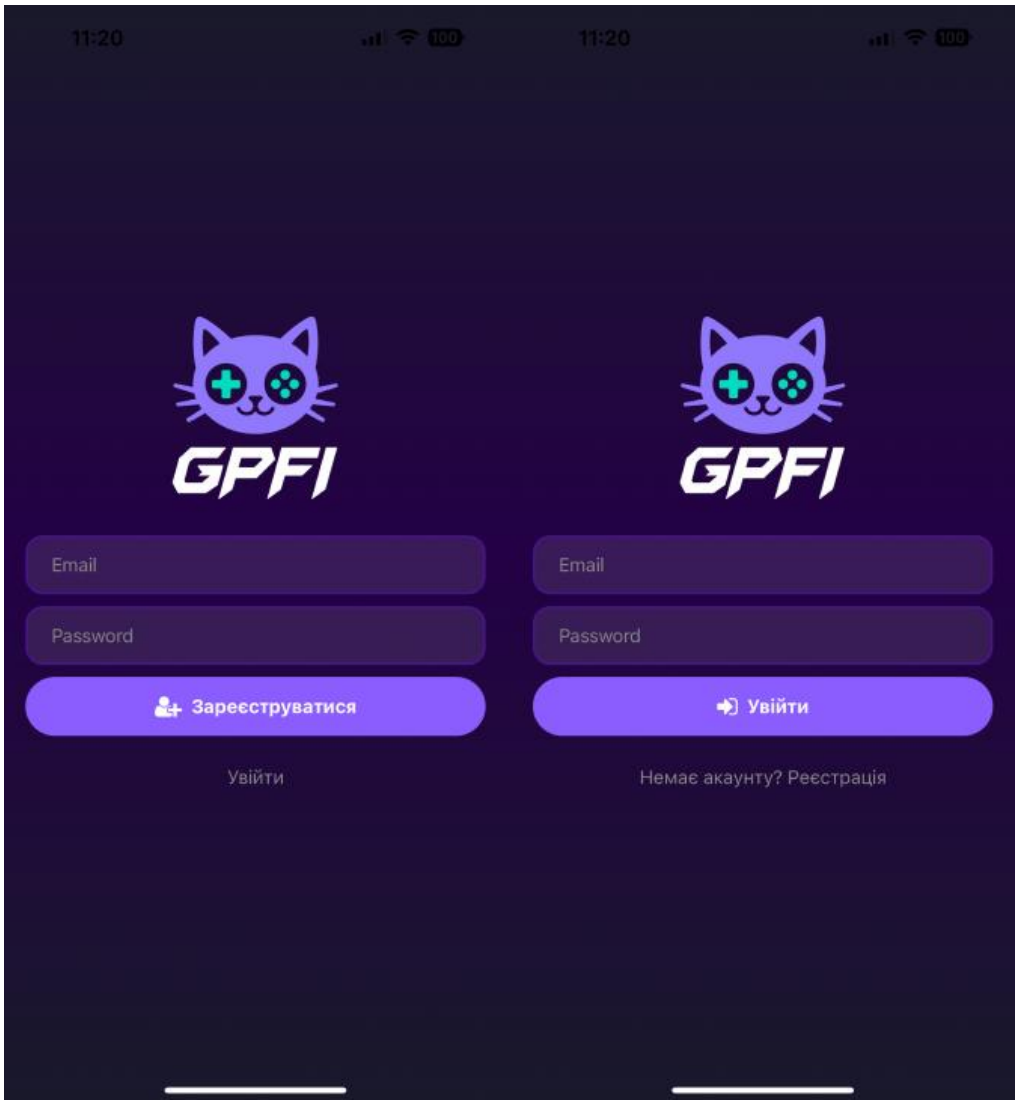

Рисунок Г.8 – Екран аутентифікації

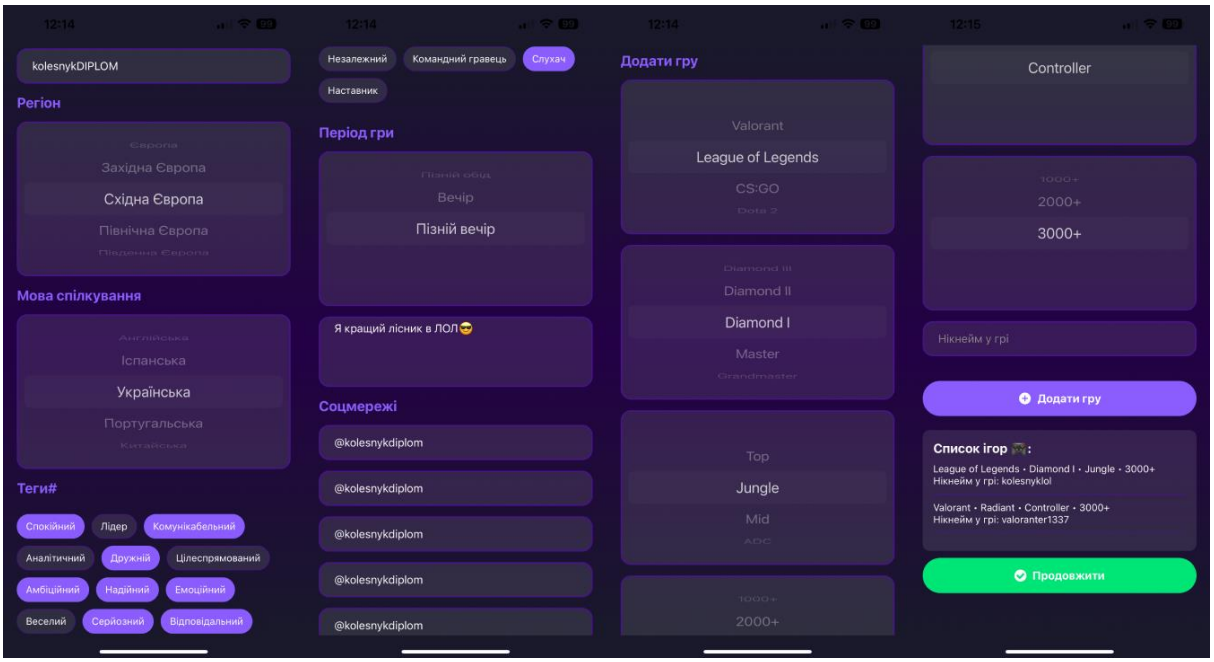

Рисунок Г.9 – Екран створення профілю

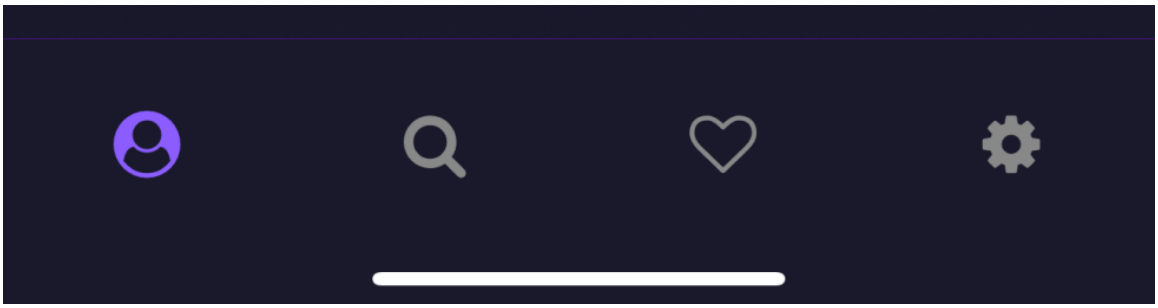

Рисунок Г.10 – Таби навігації додатку

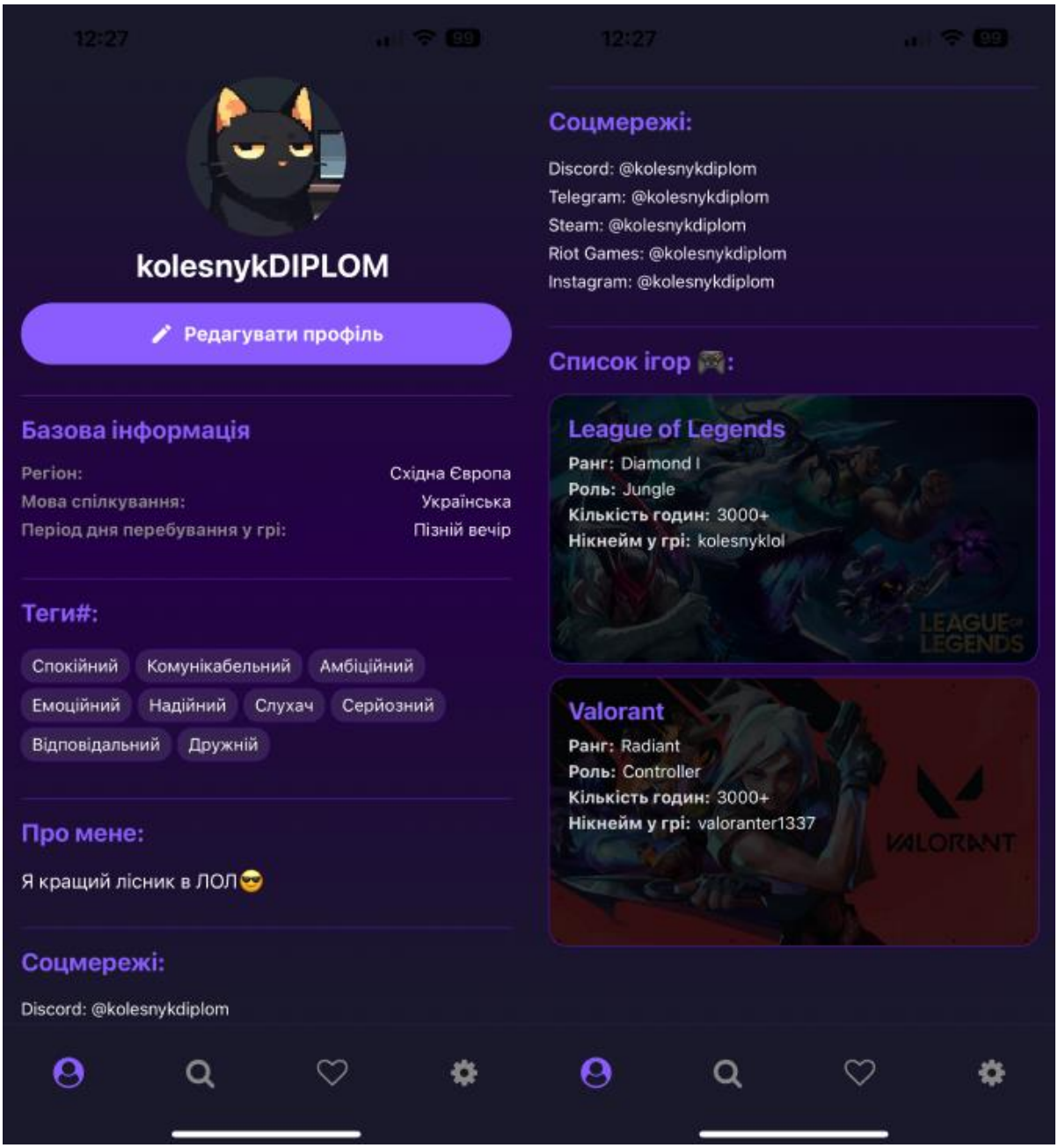

Рисунок Г.11 – Екран профілю користувача

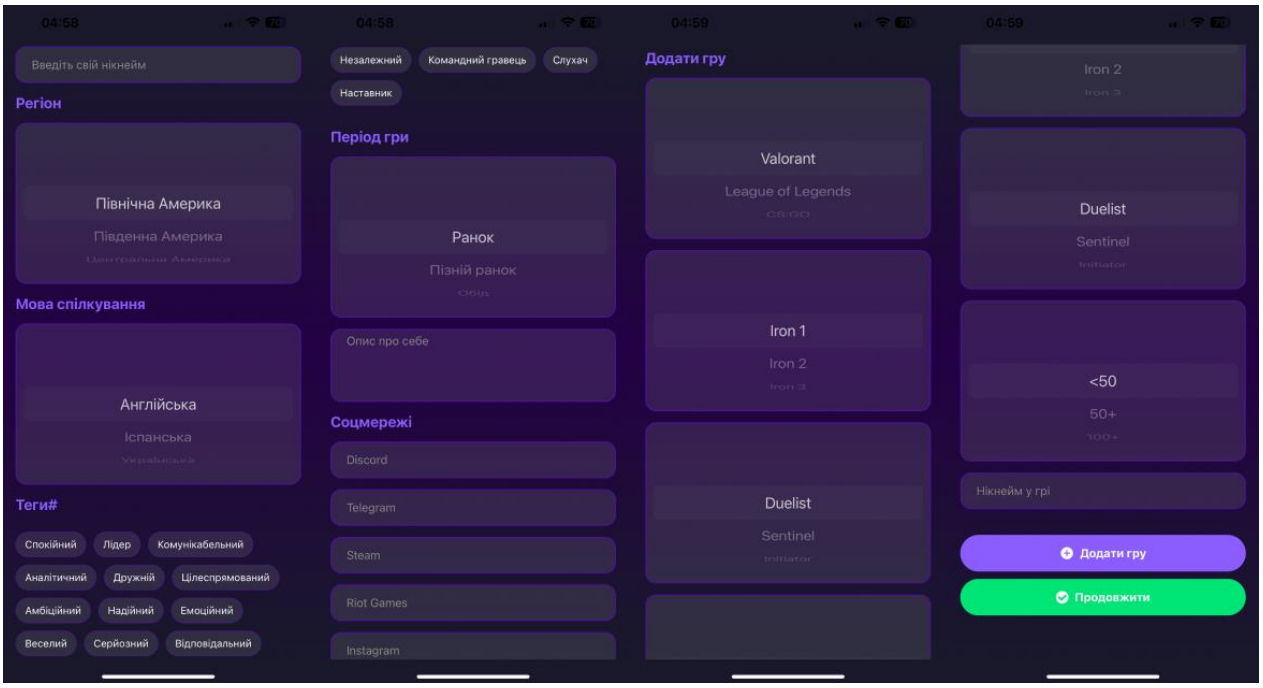

Рисунок Г.12 – Екран редагування профілю

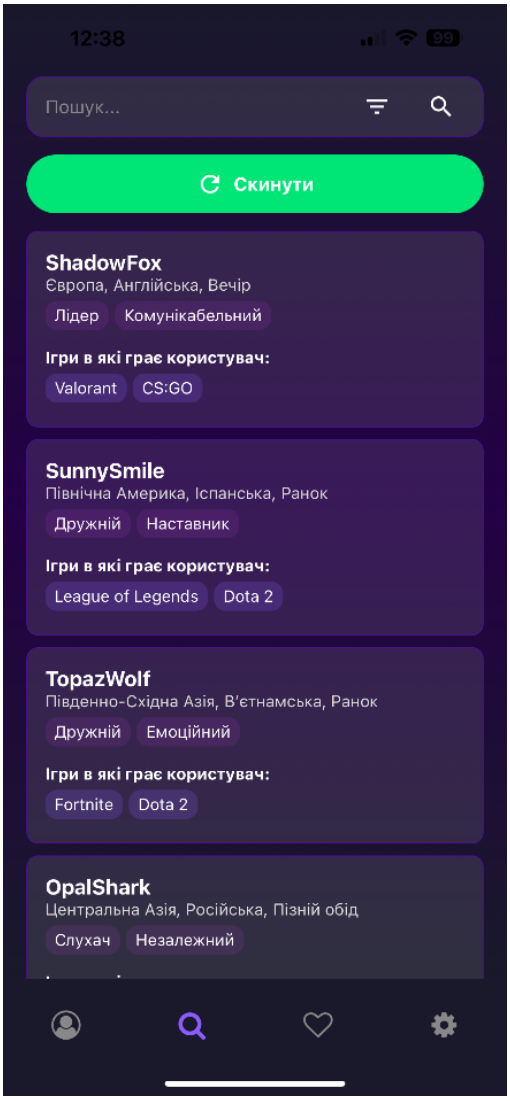

Рисунок Г.13 – Екран пошуку напарників

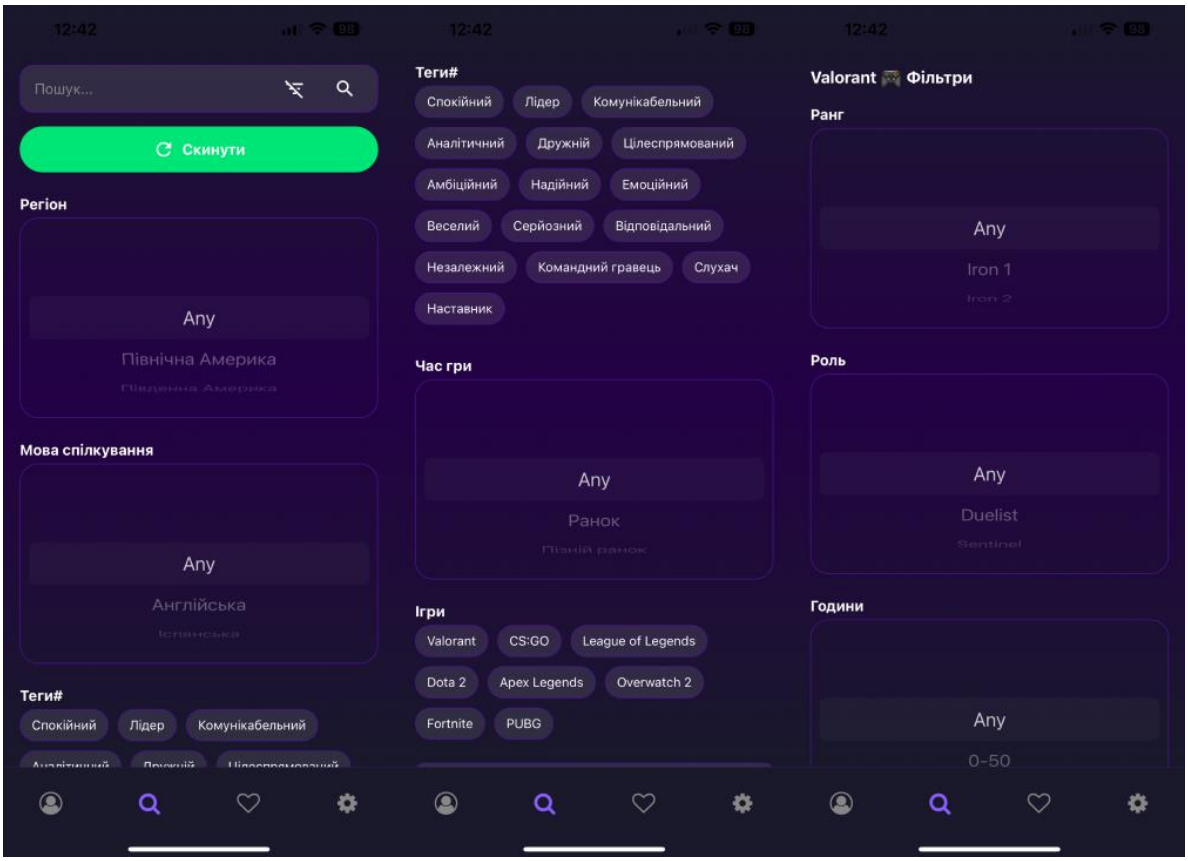

Рисунок Г.14 – Набір фільтрів

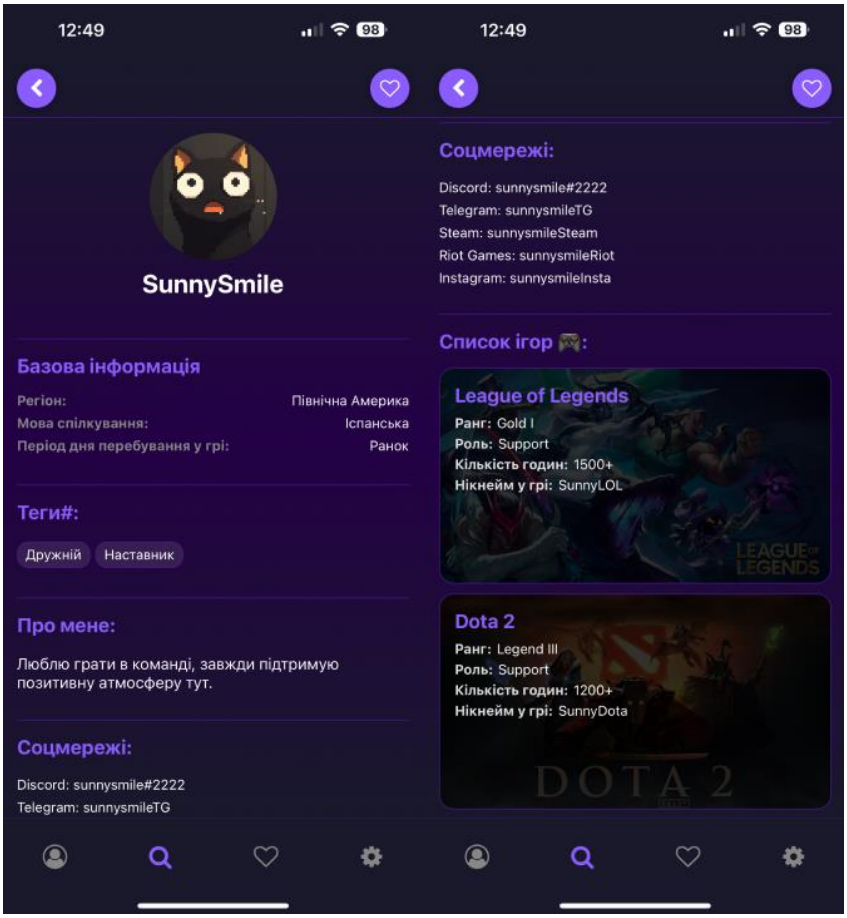

Рисунок Г.15 – Екран профілю іншого користувача

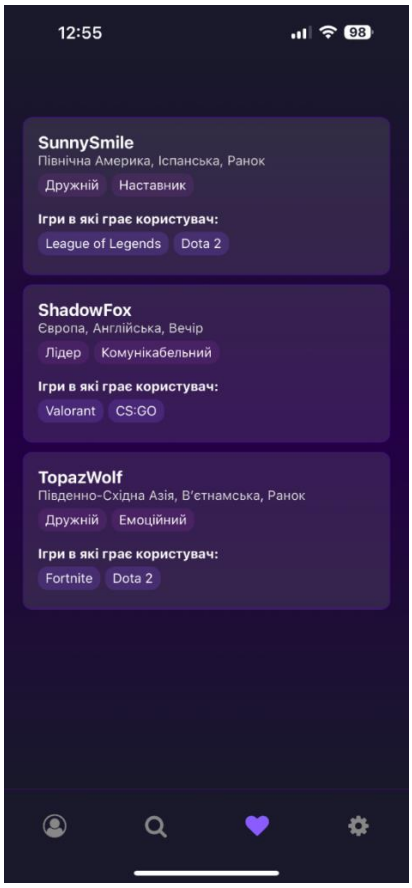

Рисунок Г.16 – Екран збережених користувачів

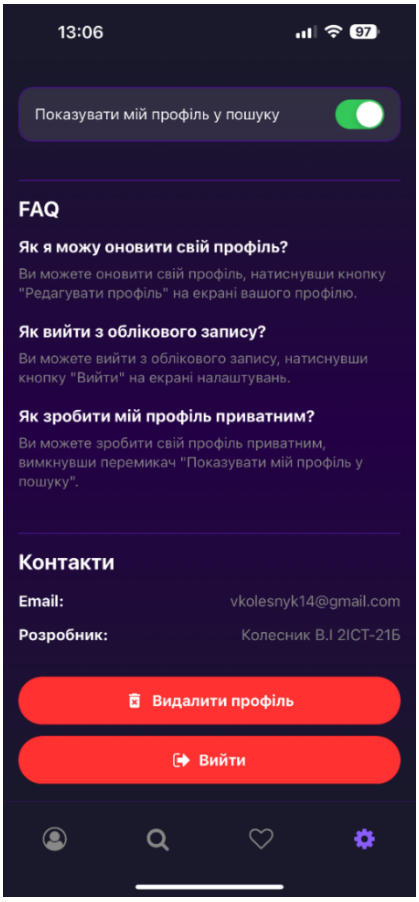

Рисунок Г.17 – Екран налаштувань