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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система для комунікації користувачів з використанням Flutter і REST API»

Виконав: студент 2 курсу, групи 1ІСТ-24м спеціальності 126 – Інформаційні системи та технології
(шифр і назва спеціальності)
Ірина ОСИПЕНКО
(ПІБ студента)

Керівник: к.т.н., професор кафедри АІІТ
Ілона БОГАЧ
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«11» 12 2025 р.

Опонент: д.т.н., проф. каф. КН
Ярослав ІВАНЧУК
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«12» 12 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)
«12» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 126 – Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«26» 09 2025 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Осипенко Ірині Віталіївні
(ПІБ автора повністю)

1. Тема роботи: Інформаційна система для комунікації користувачів з використанням Flutter і REST API.

   Керівник роботи: к.т.н., професор каф. АІІТ Богач І. В.
   Затвердженні наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «12» грудня 2025 року.
3. Вихідні дані до роботи: Операційна система: macOS, оперативна пам'ять від 8гб, мова програмування: Dart, фреймворк: Flutter 3.22.
4. Зміст текстової частини: Вступ; Аналіз предметної області та аналогів системи; Огляд програмних засобів для реалізації поставленої задачі; Розробка та тестування програмного забезпечення; Економічний розділ; Висновки; Список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: Алгоритм аутентифікації користувача в додаток, ER-діаграма структури бази даних Firestore, UML-діаграма прецедентів, Діаграма послідовності процесу обміну повідомленнями та автоматичного перекладу, Багаторівнева архітектура мобільного застосунку на Flutter, Скріншот розробленої інформаційної системи.

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 3	Ілона БОГАЧ, к.т.н., професор кафедри АІІТ	01.10.2025	02.12.2025
4	Наталія БУРЕННІКОВА, д.е.н., проф. каф. ЕптаВМ	20.10.2025	30.11.2025

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	25.09–05.10.2025	викон.
2	Огляд програмних засобів для реалізації поставленого завдання	05.10 – 25.10.2025	викон.
3	Розробка програмного забезпечення	25.10 – 10.11.2025	викон.
4	Тестування розробленого програмного забезпечення	05.11 – 20.11.2025	викон.
5	Підготовка економічної частини	до 01.12.2025	викон.
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.11 – 03.12.2025	викон.
7	Попередній захист роботи	до 03.12.2025	викон.
8	Захист роботи	до 19.12.2025	викон.

Студент _____ (підпис) Ірина ОСИПЕНКО (прізвище та ініціали)

Керівник роботи _____ (підпис) Ілона БОГАЧ (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Осипенко І. В. Інформаційна система для комунікації користувачів з використанням Flutter і REST API.

Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системита технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 120 с.

Українською мовою. Бібліогр.: 34 назв; рис.: 29; табл. 14.

Магістерська кваліфікаційна робота присвячена розробці інформаційної системи, що забезпечує ефективну комунікацію користувачів у режимі реального часу з використанням технологій Flutter для клієнтської частини та REST API для серверної взаємодії.

Практичний результат роботи полягає у створенні мобільного застосунку, який забезпечує стабільний обмін повідомленнями, синхронізацію даних між клієнтом і сервером, а також зручний інтерфейс користувача.

Розроблена система може бути використана для організації внутрішньої комунікації в компаніях, навчальних закладах або як база для створення комерційних чат-додатків.

Ключові слова: інформаційна система, Flutter, REST API, FastAPI, PostgreSQL, чат-додаток, комунікація користувачів.

ABSTRACT

Osypenko I. V. Information system for user communication using Flutter and REST API.

Master's qualification thesis in specialty 126 Information Systems and Technologies, educational program – Information Technologies of Data and Image Analysis. Vinnytsia: VNTU, 2025. 120 p.

In Ukrainian language. Bibliography: 34 titles; Fig.: 29; table. 14.

The master's thesis focuses on the development of an information system for real-time user communication based on a client–server architecture using Flutter for the client side and backend REST API.

Particular attention is given to data exchange mechanisms, REST API structure, authentication flows, and scalability of the system.

The developed mobile application provides real-time data synchronization, message storage, and an intuitive user interface.

The system can be applied in various domains – from corporate communication tools to educational environments and commercial chat platforms.

Keywords: information system, Flutter, REST API, FastAPI, PostgreSQL, chat application, real-time communication.

ЗМІСТ

ВСТУП

*Актуальність роботи***.** У сучасному світі комунікація між користувачами стала невід'ємною частиною соціального та професійного життя. Швидкий розвиток мобільних технологій, розповсюдження смартфонів та постійна присутність користувачів у мережі інтернет створюють попит на швидкі, зручні та надійні засоби обміну інформацією.

Традиційні месенджери, такі як Telegram, Discord чи Slack, забезпечують широкий спектр можливостей, проте більшість із них мають закриту архітектуру, не дозволяють розгортати власні інстанси або гнучко модифікувати бізнес-логіку для специфічних потреб компаній, навчальних закладів чи внутрішніх спільнот.

З цієї причини особливої актуальності набуває створення інформаційної системи для комунікації користувачів, що реалізована з використанням сучасних технологій Flutter та REST API, які забезпечують кросплатформність, масштабованість і швидкість обміну даними.

Використання Flutter дозволяє створювати один застосунок, що однаково добре працює як на Android, так і на iOS, а FastAPI у поєднанні з PostgreSQL забезпечує високу продуктивність серверної частини, легку інтеграцію з іншими сервісами та підтримку стандартів REST.

Таким чином, розробка такої системи є актуальним завданням у сфері інформаційних технологій, оскільки вона вирішує потребу в гнучких, надійних і безпечних інструментах комунікації, здатних адаптуватися під конкретні вимоги організації чи проєкту.

Мета роботи. Метою магістерської роботи є розробка інформаційної системи для комунікації користувачів, яка забезпечує обмін повідомленнями в режимі реального часу, підтримує групові чати, push-сповіщення, аутентифікацію користувачів та зберігання історії діалогів із використанням технологій Flutter та REST API.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області комунікаційних систем і визначити основні принципи їх побудови.

2. Здійснити аналіз існуючих аналогів чат-додатків і порівняти їх функціональні можливості.

3. Обґрунтувати вибір технологій і програмних засобів для реалізації клієнтської та серверної частини системи.

4. Розробити архітектуру інформаційної системи, включно з моделлю бази даних, структурами API-запитів і проєктуванням користувацького інтерфейсу.

5. Реалізувати прототип мобільного застосунку з функціоналом реєстрації, авторизації, створення чатів і надсилання повідомлень.

6. Провести тестування системи для перевірки стабільності, продуктивності та безпеки передачі даних.

7. Підготувати економічне обґрунтування розробки системи.

Об'єктом дослідження є процес обміну інформацією між користувачами в інтерактивних мобільних застосунках.

Предметом дослідження є методи, алгоритми та програмні засоби реалізації комунікаційних інформаційних систем із використанням технологій Flutter і REST API.

У роботі використано такі наукові методи:

1. аналітичний метод – для вивчення предметної області та аналогів систем;

2. системний аналіз – для побудови архітектури клієнт–серверної системи;

3. метод моделювання – для проєктування бази даних та REST API;

4. метод порівняльного аналізу – для вибору оптимальних інструментів розробки;

5. експериментальний метод – для тестування та перевірки працездатності системи.

Наукова-технічний результат роботи полягає у розробці архітектури інтегрованої комунікаційної системи, що поєднує клієнтський застосунок на Flutter і REST API-сервер, із реалізацією двостороннього обміну даними через HTTP/JSON та підтримкою push-сповіщень через Firebase.

Запропоноване рішення дає змогу створювати масштабовані корпоративні або навчальні чат-системи, не залежні від сторонніх хмарних сервісів.

Особливу увагу в роботі приділено забезпеченню інформаційної безпеки мобільних додатків, що передбачає захист даних користувачів під час автентифікації, передавання повідомлень та зберігання інформації на сервері. У процесі розроблення застосунку реалізовано механізми шифрування даних, захисту REST API-запитів, токен-авторизації та безпечного з'єднання через HTTPS, що відповідає сучасним вимогам до безпеки мобільних комунікаційних систем, розроблених на Flutter.

Практична цінність полягає у створенні робочого прототипу мобільного застосунку, який може бути використаний:

- для внутрішньої комунікації в компаніях і навчальних установах;

- як основа для розробки власних корпоративних месенджерів;

- для навчальних цілей при вивченні принципів побудови клієнт–серверних систем і використання Flutter/REST API.

Апробація результатів дослідження. Результати роботи було представлено на Всеукраїнській науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: Дослідження, проблеми, перспективи (МН-2025)» [1].

Впровадження результатів роботи. Результати роботи планується використати в розробках ТОВ «СПІЛЬНА СПРАВА» (Додаток Г).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ СИСТЕМИ

1.1 Сутність інформаційних комунікаційних систем

Інформаційні системи комунікаційного типу призначені для забезпечення взаємодії між користувачами з метою обміну повідомленнями, файлами, мультимедійними матеріалами та іншими типами даних у режимі реального часу. Їхнє основне призначення полягає у створенні безперебійного, швидкого та зручного каналу комунікації, який може функціонувати як у межах невеликих груп користувачів, так і в масштабах багатомільйонних аудиторій. З огляду на активну цифровізацію всіх сфер діяльності – від бізнесу й освіти до соціальних сервісів – попит на такі системи постійно зростає, що зумовлює потребу у створенні гнучких, масштабованих і безпечних платформ.

Сучасні тенденції розвитку інформаційних технологій вимагають, щоб комунікаційні системи підтримували високу швидкість обміну даними, низьку затримку під час передавання повідомлень, стабільну доступність незалежно від навантаження, а також багаторівневий захист інформації. Важливими критеріями при проєктуванні таких систем є можливість інтеграції з іншими сервісами (сховищами даних, аналітичними платформами, сервісами ідентифікації), підтримка мобільних пристроїв та вебінтерфейсів, а також здатність працювати з великими обсягами даних у режимі реального часу. Усі ці вимоги безпосередньо впливають на вибір програмної архітектури, алгоритмів обробки даних, технологічного стеку, бази даних та мережевих протоколів.

Комунікаційні системи поділяються на кілька основних типів залежно від сфери застосування та функціонального призначення. До корпоративних систем належать платформи на кшталт Slack або Microsoft Teams, які забезпечують комунікацію всередині організацій та підтримують додаткові можливості для спільної роботи, зокрема інтеграцію з CRM, календарями, таск-менеджментом тощо. Соціальні системи (Messenger, Telegram, WhatsApp) орієнтовані на масовий сегмент користувачів і зосереджені на швидкому обміні

повідомленнями, функціях каналів та груп, а також мультимедійній комунікації. Освітні або навчальні системи (Google Classroom, Moodle, Microsoft Teams for Education) мають специфічний функціонал для організації дистанційного навчання, включно з відеоконференціями, обміном навчальними матеріалами й оцінюванням. Окрему категорію становлять гібридні системи, які поєднують функції різних типів застосунків і пропонують користувачеві універсальне середовище для спілкування та спільної діяльності.

Незалежно від призначення, усі комунікаційні системи використовують клієнт–серверну архітектуру як основний підхід до передавання та обробки інформації. Клієнтська частина (мобільний або вебдодаток) відповідає за візуалізацію даних та взаємодію з користувачем, тоді як серверна частина забезпечує обробку запитів, зберігання інформації, маршрутизацію повідомлень і управління доступом. Такий підхід гарантує централізований контроль над даними, можливість масштабування системи та забезпечення належного рівня захисту інформації. Загальна структура цього типу систем наведена на рис. 1.1.

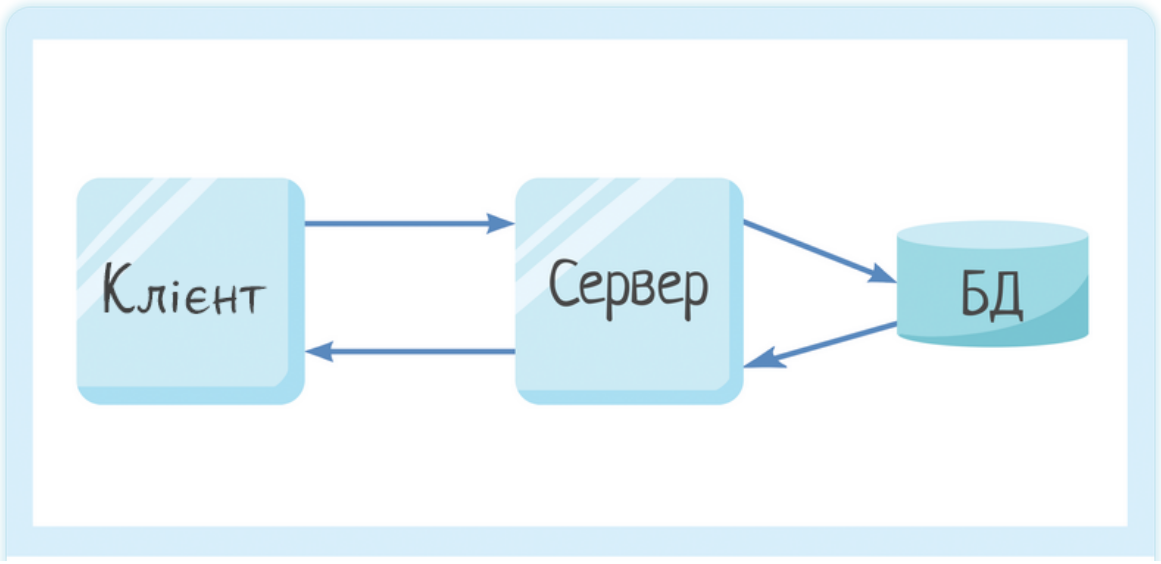

Рисунок 1.1 – Загальна схема комунікаційної системи клієнт–серверного типу

1.2 Архітектура та принципи побудови чат-додатків

Сучасні чат-додатки базуються на моделі клієнт–серверної взаємодії з використанням REST або WebSocket API (рис. 1.2 та рис. 1.3). У більшості випадків структура складається з трьох основних компонентів:

- Клієнт (Frontend) – мобільний або веб-застосунок, що відображає інформацію та надсилає запити до сервера.

- Сервер (Backend) – обробляє запити клієнтів, реалізує бізнес-логіку та забезпечує доступ до бази даних.

- База даних (Database) – зберігає інформацію про користувачів, повідомлення, чати та медіа-файли.

При розробці чат-додатків особливу увагу приділяють таким принципам:

- масштабованість – можливість обслуговування великої кількості користувачів одночасно;

- доступність – забезпечення стабільної роботи системи навіть у випадку відмови одного з вузлів;

- безпека – захист даних під час передачі та зберігання (використання JWT, HTTPS, SSL/TLS);

- зручність користувача – інтуїтивний інтерфейс і швидке реагування системи.

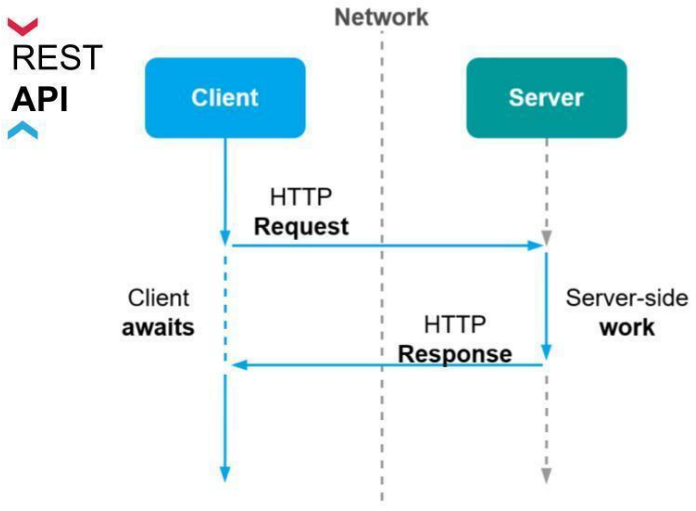

Рисунок 1.2 – Типова архітектура чат-додатку на основі REST API

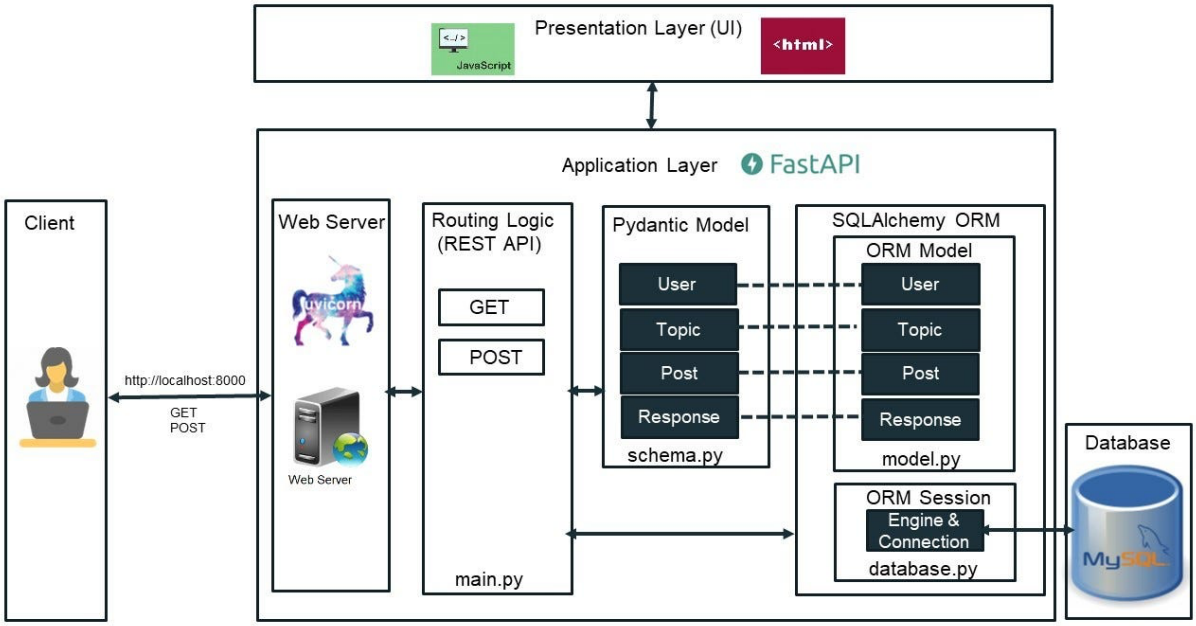

Рисунок 1.3 – Структура REST API FastAPI для обміну повідомленнями

Завдяки архітектурі REST забезпечується стандартизована взаємодія між клієнтом і сервером, що дозволяє легко масштабувати систему та інтегрувати нові модулі (наприклад, push-сповіщення або аналітику).

У реальних промислових рішеннях архітектура чат-систем часто розширюється за рахунок додаткових компонентів:

1. WebSocket-сервер або брокер подій.

На відміну від REST, що працює за принципом "запит–відповідь", WebSocket забезпечує двостороннє з’єднання між клієнтом і сервером, що дозволяє миттєво доставляти нові повідомлення без потреби у періодичному опитуванні сервера. Саме тому WebSocket є стандартом де-факто для реалізації чатів.

2. Черги повідомлень (Message Queue).

Для забезпечення відмовостійкості та обробки великих потоків подій використовують системи черг, такі як RabbitMQ, Kafka або Redis Streams. Вони дозволяють:

– обробляти повідомлення асинхронно;

– забезпечити доставку навіть при пікових навантаженнях;

– масштабувати логіку опрацювання подій.

3. Кешування даних.

Щоб мінімізувати затримки, системи застосовують Redis або Memcached для кешування списків чатів, історії останніх повідомлень, статусу активності користувачів та токенів автентифікації.

4. Load Balancer (балансувальник навантаження).

Забезпечує рівномірний розподіл трафіку між серверними екземплярами та підтримку високої доступності системи.

5. Сервіси аналітики та моніторингу.

Вони дають змогу відстежувати стабільність роботи системи, швидкість доставки повідомлень, активність користувачів і виявляти аномалії.

Архітектура REST використовується переважно для таких операцій:

- реєстрація та автентифікація користувачів,
- створення та керування чатами,
- завантаження медіафайлів,
- отримання історії повідомлень,
- оновлення профілю.

REST забезпечує стандартизовану взаємодію між клієнтом і сервером, що дозволяє легко масштабувати систему, інтегрувати нові модулі (наприклад, push-сповіщення або аналітичні сервіси), а також забезпечувати сумісність з різними клієнтськими платформами.

WebSocket, своєю чергою, забезпечує:

- доставку нових повідомлень у режимі реального часу;
- оперативне передавання статусів активності (online/offline, typing...);
- оновлення інтерфейсу без перезавантаження сторінки;
- синхронізацію між кількома пристроями користувача.

Завдяки поєднанню REST і WebSocket сучасні чат-додатки можуть забезпечувати надійну, масштабовану та швидкодіючу комунікацію. Використання цих технологій дозволяє гнучко розширювати функціональність

системи, інтегрувати додаткові сервіси, а також підтримувати миттєву взаємодію між користувачами, що є ключовою вимогою до платформ реального часу.

1.3 Аналіз аналогів розроблюваної системи

FlaiChat – це сучасний мобільний застосунок для мультимовного спілкування, який поєднує функції чату, автоматичного перекладу повідомлень і розширених можливостей співпраці. Його головна ідея полягає у створенні зручного середовища для комунікації між користувачами, що говорять різними мовами, без необхідності користуватися сторонніми перекладачами чи перемикатися між додатками (рис. 1.4).

Функціональні можливості системи:

1. Автоматичний переклад текстових і голосових повідомлень у режимі реального часу. Кожне повідомлення миттєво перекладається для всіх учасників групи на їхню мову.

2. Підтримка понад 40 мов і голосовий переклад із клонуванням голосу – користувач може спілкуватися рідною мовою, а співрозмовник чує переклад із тим самим тембром голосу.

3. Можливість створення групових чатів без необхідності реєстрації: користувач може приєднатися до розмови за посиланням або QR-кодом.

4. Функція “Threaded Replies” (відповіді в гілках), що полегшує структурування обговорень у великих групах.

5. Інтегровані задачі та нагадування, які дозволяють перетворювати повідомлення на завдання з крайніми термінами.

6. Функція “OnTheFlai” для спонтанного обміну фотографіями, короткими відео або повідомленнями у групах.

7. Підтримка мультиплатформенності – застосунок доступний на Android та iOS.

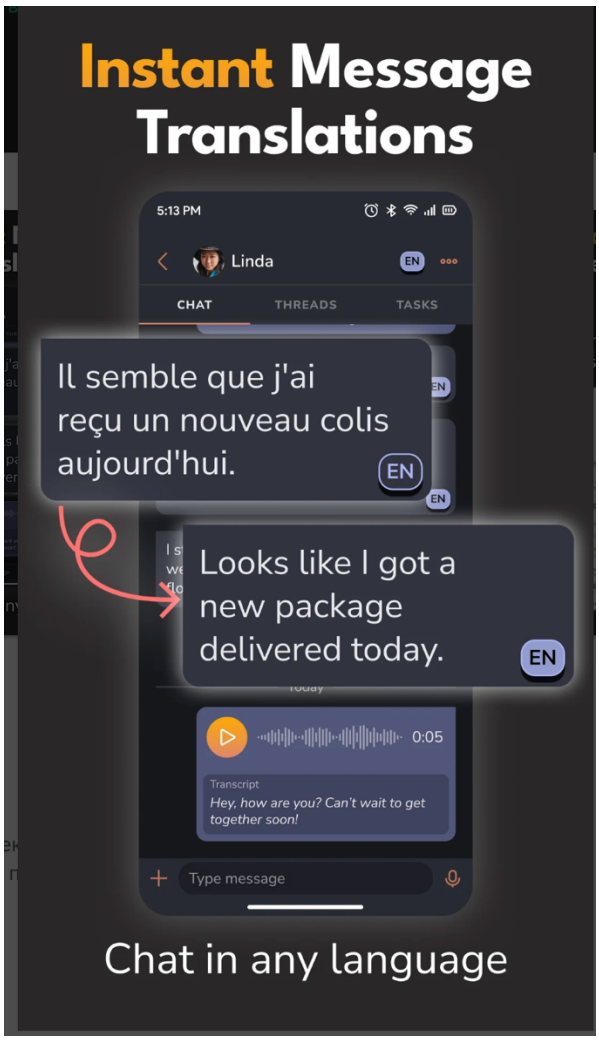

Рисунок 1.4 – Демонстрація домашньої сторінки FlaiChat

Переваги FlaiChat:

1. Забезпечує природну комунікацію між користувачами, які говорять різними мовами, завдяки автоматичному перекладу тексту та голосу.

2. Висока зручність використання: система не потребує реєстрації або введення персональних даних, що робить її доступною та безпечною.

3. Підтримка великої кількості мов сприяє глобальному охопленню користувачів.

4. Наявність додаткових можливостей (нагадування, планування завдань, голосове клонування) розширює сферу застосування системи – від особистого спілкування до колаборації в команді.

5. Простий і інтуїтивний інтерфейс, який не потребує спеціальної підготовки.

Недоліки FlaiChat:

1. Деякі базові функції месенджерів (наприклад, пошук користувачів або фільтрація повідомлень) реалізовані обмежено.

2. У певних випадках спостерігається зниження продуктивності під час роботи з великими групами або мультимедійним контентом.

3. Механізми обробки голосових даних і збереження історії спілкування недостатньо прозорі, що може викликати сумніви щодо інформаційної безпеки.

4. Частина функцій (наприклад, переклад голосових повідомлень у реальному часі) доступна лише у преміум-версії. [3]

2. Beeper – це інноваційний мультиплатформенний месенджер, який об’єднує різні канали комунікації в одному інтерфейсі. Його основна концепція – створити єдину точку доступу до більшості популярних сервісів обміну повідомленнями, включно з WhatsApp, Telegram, Signal, Instagram, Discord, Slack та іншими (рис. 1.5).

Функціональні можливості системи:

1. Єдина платформа для спілкування з контактами з різних сервісів.

2. Підтримка синхронізації між пристроями (телефон, планшет, комп’ютер).

3. Інтеграція з API сторонніх сервісів через відкриту систему плагінів.

4. Уніфікований пошук повідомлень і файлів незалежно від платформи.

5. Шифрування повідомлень за протоколом Matrix для безпечної передачі даних.

6. Можливість керування повідомленнями з єдиного “вхідного потоку” (Unified Inbox).

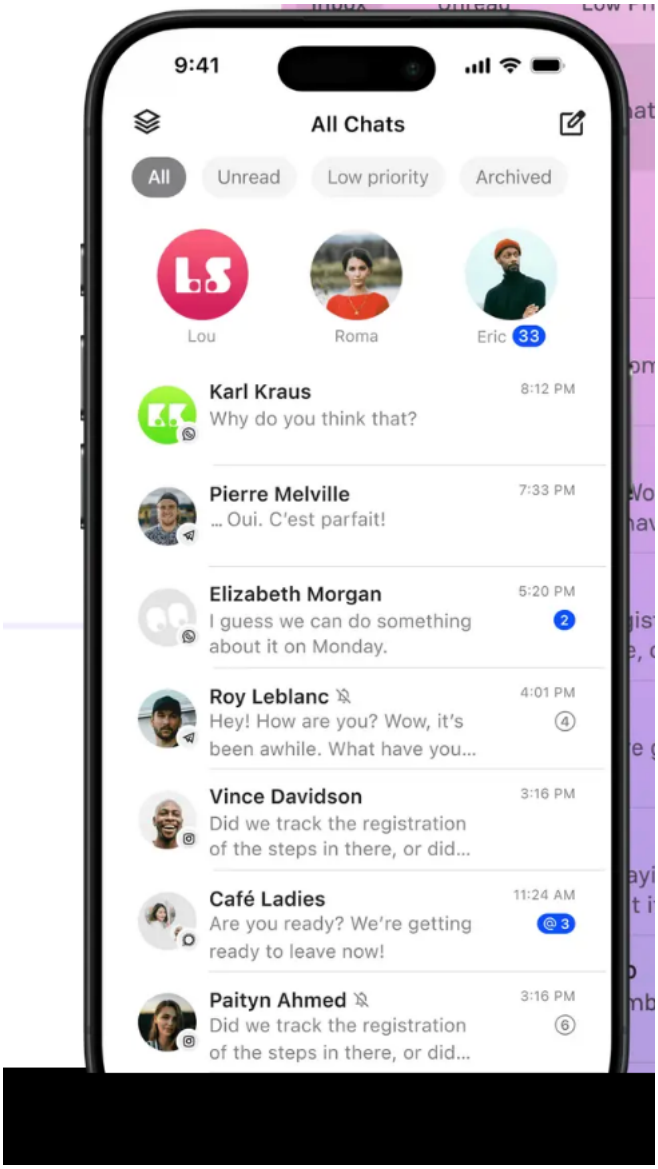

Рисунок 1.5 – Інтерфейс додатку Beeper

Переваги Beeper:

- Зручність у роботі з кількома месенджерами одночасно – користувачу не потрібно перемикатися між додатками.

- Підтримка великої кількості інтеграцій і відкритість API для розширення функціональності.

- Високий рівень безпеки завдяки шифруванню та власній інфраструктурі Matrix.

- Збалансований дизайн і зрозумілий інтерфейс, який нагадує класичну електронну пошту. [4]

- Наявність десктопної версії, що полегшує роботу командам і корпоративним користувачам.

Недоліки Beeper:

- Наявність складного процесу первинного налаштування, який потребує підключення сторонніх акаунтів.

- Частина інтеграцій може втрачати стабільність через зміни у сторонніх API.

- Високі вимоги до ресурсів пристрою при великій кількості активних підключень.

- Обмеження безкоштовного тарифу на кількість підключених сервісів.

3. Наступним додатком розглянемо ShareChat.

ShareChat – це популярна соціально-комунікаційна платформа, орієнтована на мультимовну взаємодію користувачів. Вона поєднує елементи месенджера, спільноти та контентного сервісу, надаючи користувачам можливість спілкуватися, ділитися медіа, створювати пости та долучатися до тематичних груп за інтересами (рис. 1.6). На відміну від традиційних месенджерів, ShareChat акцентує увагу на створенні локальних спільнот, підтримуючи понад 15 мов, включаючи англійську, хінді, бенгальську, тамільську та телугу.

Функціональні можливості системи:

1. Підтримка мультимовного контенту та автоматичний переклад повідомлень між користувачами.

2. Можливість створення тематичних груп для обговорення різних тем – від освіти й технологій до розваг і новин.

3. Функції публікації коротких відео, фотографій, текстових постів і голосових повідомлень.

4. Вбудована система реакцій, коментарів і рекомендаційного контенту.

5. Інструменти приватного чату для безпосередньої комунікації між користувачами.

6. Інтеграція штучного інтелекту для персоналізованої стрічки новин і рекомендацій.

7. Можливість ділитися контентом у зовнішніх соціальних мережах (WhatsApp, Instagram, Telegram тощо).

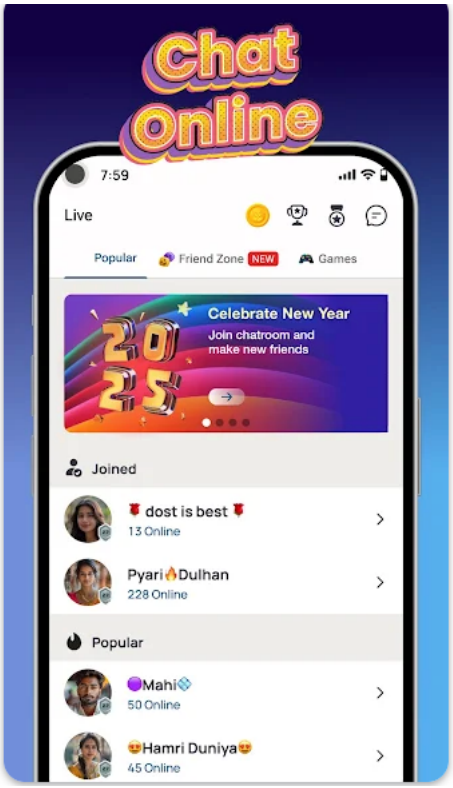

Рисунок 1.6 – Інтерфейс додатку ShareChat

Переваги ShareChat:

- Забезпечує повноцінну мультимовну взаємодію та автоматичний переклад без використання сторонніх додатків.

- Має розвинену екосистему користувацьких груп і тематичних каналів, що сприяє формуванню спільнот за інтересами.

- Поєднує функції месенджера і соціальної мережі, що робить платформу універсальною для обміну інформацією.

- Використовує інтелектуальні алгоритми рекомендацій для підвищення залучення користувачів.

- Простий і привабливий інтерфейс, оптимізований для мобільних пристроїв.

Недоліки ShareChat:

- Основний контент орієнтований на певні регіони (Індію та Південну Азію), що обмежує використання у міжнародному контексті.

- Обмежені можливості для корпоративної або навчальної комунікації – система переважно сфокусована на розважальному та соціальному контенті.

- Наявність реклами та відкритої стрічки може відволікати користувачів від основної мети спілкування.

- Недостатня прозорість політики збереження даних і контролю доступу до публікацій.

1.4 Висновки до розділу

У першому розділі було здійснено ґрунтовний аналіз предметної області та сучасних підходів до побудови комунікаційних інформаційних систем. Детально розглянуто архітектурні моделі, що лежать в основі чат-додатків, зокрема клієнт–серверну взаємодію з використанням REST API та механізмів реального часу. Визначено, що така архітектура забезпечує високий рівень масштабованості, стабільності, зручності при розробленні та подальшій підтримці системи, а також спрощує інтеграцію з додатковими сервісами, серед яких системи автентифікації, модулі обробки медіафайлів, сервіси аналітики та push-сповіщення.

Проведений аналіз аналогів (FlaiChat, Beeper, ShareChat тощо) дав змогу визначити актуальні тенденції ринку та ключові функціональні можливості, які очікують користувачі сучасних комунікаційних платформ. Зокрема, були окреслені такі вимоги, як забезпечення обміну повідомленнями у режимі реального часу, підтримка групової взаємодії, можливість мультимовної комунікації, синхронізація роботи між різними пристроями, а також високий рівень захисту персональних даних. Аналіз показав, що найбільш конкурентоспроможними є системи, які поєднують традиційний функціонал

месенджера з інтелектуальними сервісами — автоматичним перекладом, рекомендаційними алгоритмами, інтеграцією контентних модулів.

Узагальнюючи результати дослідження, можна зробити висновок, що створення власної універсальної комунікаційної системи є доречним і своєчасним, оскільки існуючі аналоги або не повністю задовольняють вимоги певних груп користувачів, або мають обмеження щодо функціональності, масштабованості чи прозорості в обробці даних. Виявлені особливості та недоліки сучасних рішень стали основою для формування вимог до майбутньої системи, що забезпечує цінне підґрунтя для подальших етапів проєктування та розроблення.

2 ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПОСТАВЛЕНОЇ ЗАДАЧІ

Для створення сучасних інформаційних систем комунікаційного типу необхідно використовувати набір інструментів, що забезпечують ефективну розробку, високий рівень продуктивності та можливість масштабування. Огляд програмних засобів є важливим етапом, оскільки вибір технологій впливає на архітектуру майбутньої системи, стабільність роботи, зручність розвитку та подальшу підтримку. На сьогодні розробники мають доступ до широкого спектра платформ, фреймворків і сервісів, що дозволяють створювати високонавантажені мобільні застосунки та веб-системи.

Насамперед доцільно виділити програмні засоби для реалізації клієнтської частини, адже саме інтерфейс користувача забезпечує взаємодію з функціональністю системи. Сучасні фреймворки для створення мобільних застосунків, такі як кросплатформні середовища або нативні SDK, дають змогу поєднати високу продуктивність із широкими можливостями кастомізації інтерфейсу. У процесі вибору враховуються вимоги до продуктивності, підтримки анімацій, адаптивної верстки та інтеграції з серверною частиною.

Окрему групу становлять інструменти для серверної розробки — мови програмування, веб-фреймворки та системи керування базами даних. Вони забезпечують обробку запитів, зберігання даних, виконання бізнес-логіки та підтримку функцій реального часу. Вибір бекенд-технологій визначається необхідністю забезпечити низьку затримку при передачі повідомлень, підтримку великої кількості одночасних підключень, можливість горизонтального масштабування та інтеграцію з зовнішніми сервісами. Особливу увагу приділяють питанню безпеки — використанню протоколів шифрування, токен-автентифікації та механізмів захисту даних.

Також важливими є програмні засоби, що забезпечують управління даними. Системи керування базами даних можуть бути реляційними або NoSQL-

типу залежно від характеру структури даних, інтенсивності читання/запису та потреб у гнучкому масштабуванні. У чат-системах зазвичай використовуються технології, що підтримують швидке читання великих масивів повідомлень і оптимізовані для роботи з нерегулярними або динамічно змінними структурами.

Для організації процесу розробки використовуються інструменти контролю версій, системи для спільної роботи та платформи CI/CD, що забезпечують автоматизацію збирання, тестування та розгортання застосунку. Їх застосування значно скорочує час упровадження нових функцій і підвищує стабільність системи.

Отже, сучасний набір програмних засобів дозволяє створити масштабовану, надійну та гнучку комунікаційну систему, що відповідає сучасним вимогам продуктивності й безпеки. Розглянуті технології формують основу для подальшої реалізації клієнтської й серверної частин, а також забезпечують можливість інтеграції нових функціональних модулів у майбутньому.

2.1 Сучасні тенденції у мобільній розробці

Розвиток мобільних технологій значно розширив спектр підходів до створення додатків, поставивши розробників перед ключовим вибором: нативна розробка чи кросплатформні фреймворки.

Вибір технології визначає подальший розвиток проєкту. У сфері мобільної розробки існує безліч підходів, кожен з яких має свої переваги та сфери застосування.

Нативний підхід передбачає створення унікальних додатків для кожної операційної системи (наприклад, використання Swift для iOS та Kotlin для Android). Це забезпечує найвищу продуктивність, найкращу інтеграцію з платформою та повний доступ до всіх нативних API та функцій пристрою. Це

ідеально підходить для додатків, які потребують інтенсивної роботи з ресурсами пристрою або глибокої взаємодії з ОС.

Недолік: Цей метод вимагає окремої розробки та підтримки кодової бази для кожної платформи, що є дорогим і витратним за часом та ресурсами.

Кросплатформні рішення, такі як Flutter, React Native або Xamarin, набувають популярності (про що свідчить рис. 2.1). Вони пропонують більш гнучкий та економний шлях, дозволяючи розробникам використовувати єдину кодову базу для створення додатків, що працюють на різних платформах [5, 6]. Це суттєво скорочує час розробки та спрощує підтримку проєкту.

Рисунок 2.1 – Порівняння популярності різних фреймворків

Сфера застосування: Ідеально підходять для проєктів з обмеженими ресурсами або тих, що прагнуть швидкого виходу на ринок.

Компроміси: Можуть мати певні обмеження щодо продуктивності та доступності деяких специфічних функцій конкретної платформи.

Для оцінки доцільності використання вибраних технологій здійснено порівняння Flutter, React Native, Kotlin Multiplatform, FastAPI, Django REST і Node.js за основними критеріями. Провівши порівняльний аналіз (табл. 2.1), нашим вибором став Flutter. Цей фреймворк забезпечує високу продуктивність завдяки компіляції в нативний код, а також пропонує широкий спектр готових до використання віджетів та інструментів для створення красивих і функціональних інтерфейсів користувача. Flutter підходить для швидкої розробки програми з динамічним інтерфейсом та забезпечує зручність роботи як для розробників, так і для користувачів незалежно від платформи.

Таблиця 2.1 – Порівняння основних технологій для створення чат-додатків

Технологія	Продуктив-ність	Простота розробки	Підтримка REST API	Кросплатформ-ність	Ліцензія
Flutter	Висока	Висока	Так	Так	Open Source
React Native	Середня	Висока	Так	Так	Open Source
Kotlin Multiplatform	Висока	Середня	Так	Так	Open Source
FastAPI	Дуже висока	Висока	Так	Так	Open Source
Django REST	Середня	Висока	Так	Так	Open Source
Node.js	Висока	Висока	Так	Так	Open Source

Переваги Flutter для мобільного додатка У сучасних умовах розвитку мобільних технологій вибір оптимального фреймворку для створення програмного забезпечення є одним із ключових факторів успішної реалізації проєкту. На ринку мобільної розробки, де конкуренція надзвичайно висока,

розробники прагнуть знайти інструменти, які забезпечують поєднання швидкості розробки, стабільності роботи та зручності візуальної реалізації.

Одним із таких інструментів є Flutter – фреймворк від компанії Google, призначений для кросплатформної розробки мобільних, веб- та десктопних застосунків із єдиною базою коду [7].

Архітектура Flutter

Архітектура Flutter побудована на трирівневій структурі (рис. 2.2):

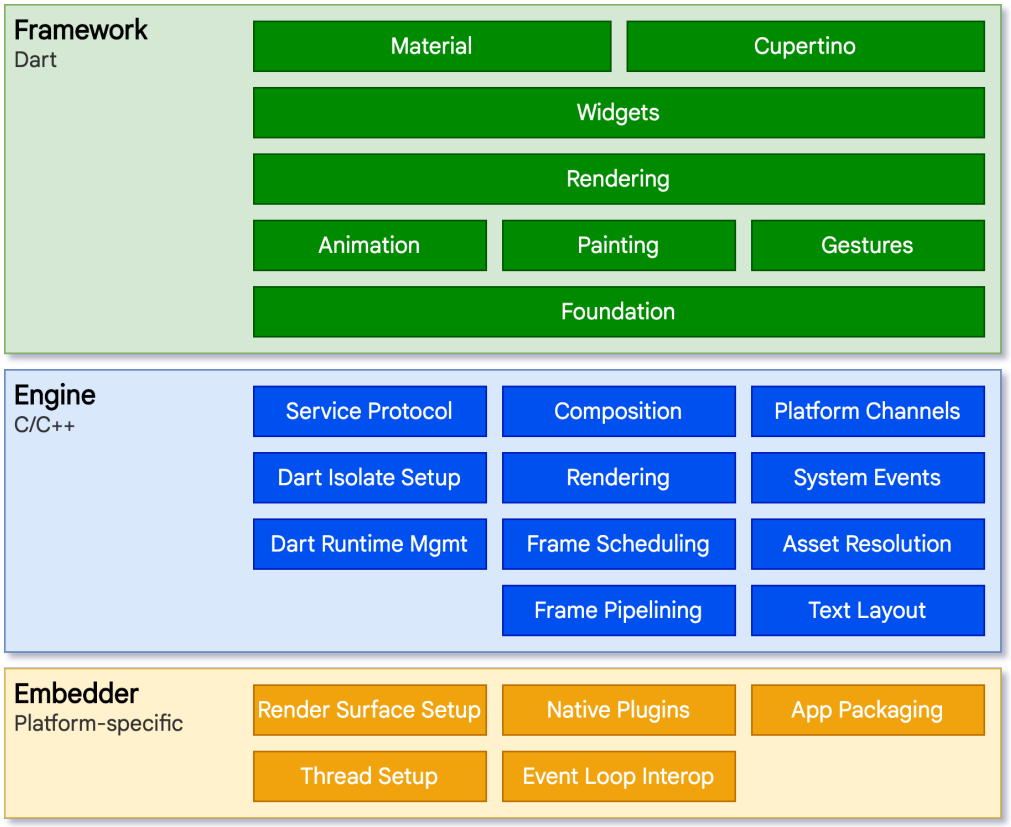

Рисунок 2.2 – Архітектура фреймворку Flutter

1. Framework Layer – рівень, який містить набір віджетів, API для рендерингу, засоби анімації та керування станом. Він дозволяє створювати гнучкий і сучасний інтерфейс користувача.

2. Engine Layer – реалізований мовою C++, відповідає за графічний рендеринг за допомогою рушія Skia, а також за компіляцію у нативний ARM-код.

3. Embedder Layer – забезпечує інтеграцію фреймворку з операційною системою пристрою (Android, iOS, Windows, Linux, macOS).

Такий підхід гарантує високу продуктивність, стабільність та плавну роботу інтерфейсу незалежно від платформи.

Використання Flutter для створення інформаційної системи комунікації користувачів обумовлене його функціональними та технічними перевагами, зокрема:

1. Кросплатформність.

   Flutter дозволяє створювати застосунки одночасно для Android, iOS і Web з використанням єдиної кодової бази. Це значно скорочує час розробки, зменшує витрати на підтримку та забезпечує однаковий користувацький досвід на різних пристроях.
2. Висока продуктивність. Завдяки компіляції у нативний ARM-код і використанню графічного рушія Skia, застосунки на Flutter працюють швидко, забезпечуючи стабільне відтворення анімацій та елементів інтерфейсу.
3. Hot Reload. Функція “гарячого перезапуску” дозволяє розробникам миттєво бачити зміни в коді без повної перезбірки проєкту. Це значно прискорює процес налагодження, тестування інтерфейсу та розробку нових функцій.
4. Гнучка система віджетів. Flutter пропонує широкий набір готових компонентів (Material Design, Cupertino Widgets) і можливість їх глибокої кастомізації. Це дає змогу реалізувати як стандартні, так і унікальні інтерфейси, орієнтовані на потреби користувачів (рис. 2.3).

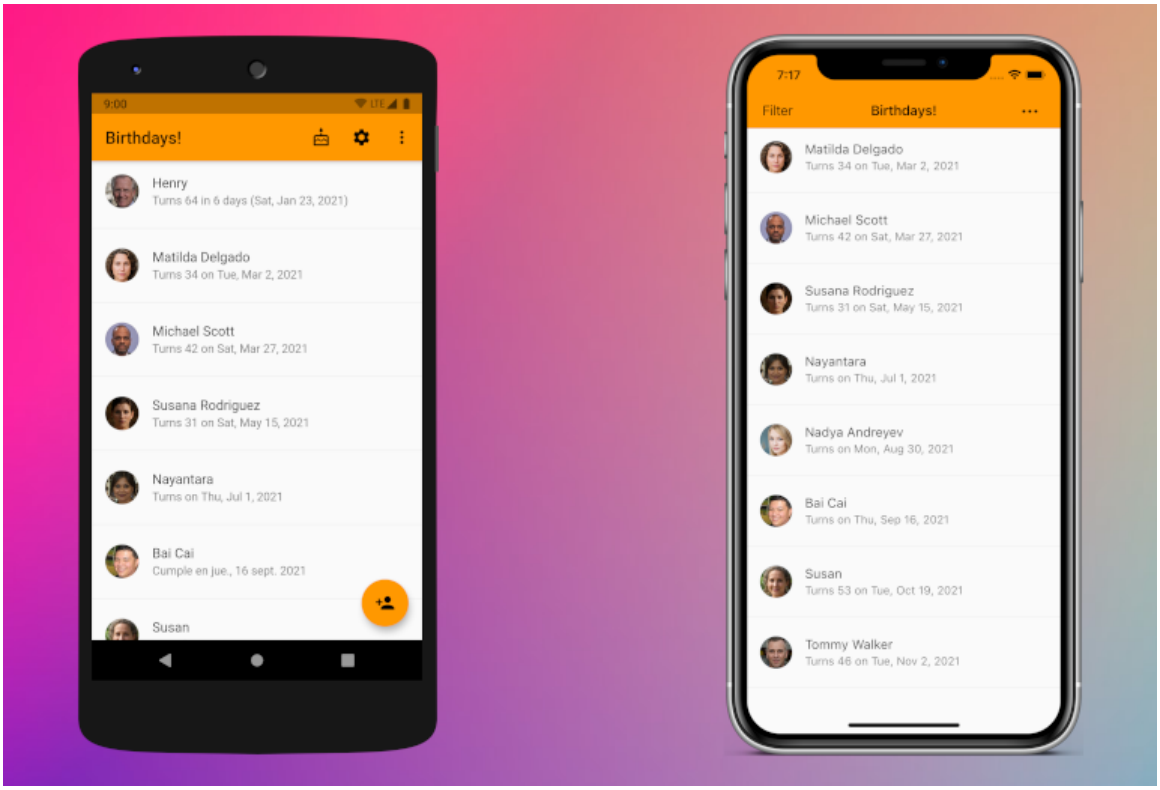

Рисунок 2.3 – Приклад однакових інтерфейсів Flutter на різних девайсах

5. Сучасна екосистема бібліотек. Flutter підтримує численні пакети для роботи з REST API, базами даних і сервісами. У межах даного проєкту використовуються такі бібліотеки, як:
   - http та dio – для виконання HTTP-запитів;
   - provider – для управління станом;
   - firebase_messaging – для отримання push-сповіщень у режимі реального часу.
6. Підтримка від Google і активна спільнота. Flutter активно оновлюється й підтримується спільнотою розробників, що забезпечує його стабільність, актуальність і швидку інтеграцію нових можливостей.

Робота з REST API у Flutter

Одним із головних аспектів інформаційної системи є обмін даними між клієнтською частиною та сервером через REST API. Flutter має вбудовані інструменти для виконання асинхронних запитів (через ключові слова async та await) і підтримує просту інтеграцію з API.

Для обробки даних використовується пакет dio, який забезпечує відправлення запитів типу GET, POST, PUT і DELETE. Дані, отримані у форматі JSON, перетворюються у моделі за допомогою пакета json_serializable, що спрощує серіалізацію та десеріалізацію об’єктів.

Завдяки підтримці патернів Provider та Bloc, бізнес-логіка системи відокремлюється від UI, забезпечуючи стабільність роботи та реактивне оновлення екранів при зміні стану програми.

У межах розроблення інформаційної системи для комунікації користувачів Flutter використано для створення клієнтської частини мобільного застосунку, який дозволяє:

1. здійснювати обмін повідомленнями у реальному часі;
2. отримувати push-сповіщення;
3. переглядати профілі користувачів і редагувати налаштування;
4. інтегруватися з REST API для обміну даними.

На рисунку 2.4 представлено приклад структури екрана чату у Flutter-застосунку, що є однією з ключових частин системи.

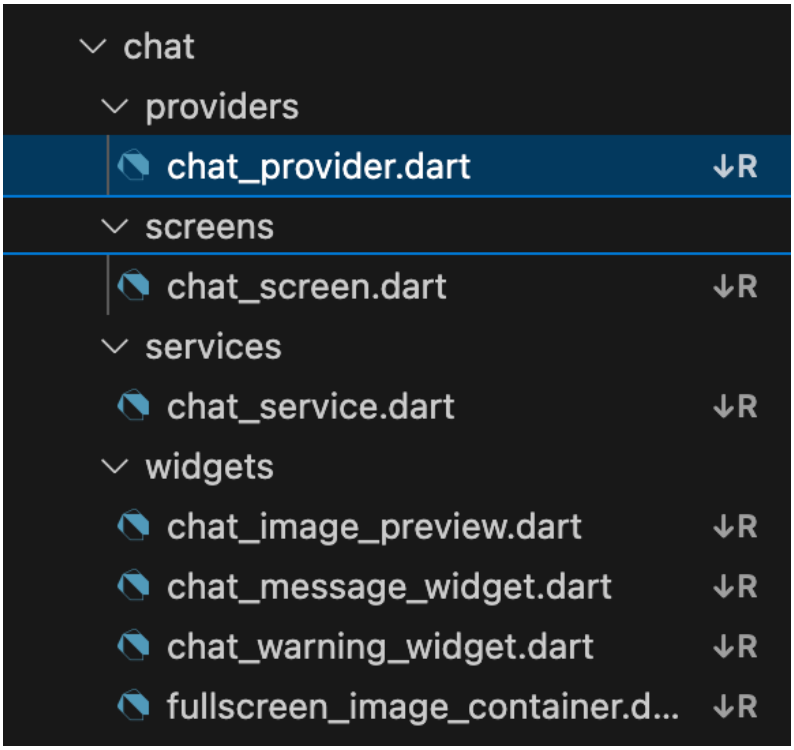

Рисунок 2.4 – Приклад структури екрана чату

Вибір Flutter для розробки мобільного додатку обумовлений його множинними перевагами, включаючи високу продуктивність, гнучкість у дизайні та зручність в управлінні станом програми. Ці характеристики роблять Flutter ідеальним інструментом для створення сучасної, функціональної та привабливої програми, яка буде задовольняти потреби користувачів та забезпечувати стабільну та ефективну роботу.

2.2 Переваги Dart у порівнянні з іншими мовами програмування

Мова Dart є сучасною обєктно орієнтованою мовою програмування, розробленою компанією Google, яка отримала широке використання у мобільній розробці завдяки фреймворку Flutter. Її переваги у порівнянні з іншими мовами визначають актуальність застосування Dart для створення кросплатформних програмних продуктів.

Однією з ключових переваг є підтримка режиму Just in Time та Ahead of Time компіляції [13]. Поєднання цих підходів забезпечує високу швидкодію застосунків у продакшні та можливість миттєвого оновлення інтерфейсу під час розроблення. Порівняно з мовами, що використовують інтерпретацію або не підтримують статичну оптимізацію, застосунки на Dart демонструють вищу продуктивність і меншу затримку рендерингу.

Важливою властивістю Dart є використання власного механізму рендерингу [14]. Це дозволяє працювати незалежно від нативних компонентів операційних систем та уникати фрагментації інтерфейсу, яка є типовою для деяких кросплатформних рішень. На відміну від підходів, що покладаються на WebView або нативні модулі, Dart у поєднанні з Flutter дає змогу досягати стабільності та однорідності інтерфейсу на різних платформах.

Статична типізація Dart сприяє підвищенню надійності програмного забезпечення. Вона зменшує кількість помилок під час виконання та пришвидшує процес тестування [15]. Це особливо важливо під час створення

масштабованих систем. Порівняно з мовами, що використовують динамічну типізацію, Dart забезпечує кращий рівень контролю над типами даних і водночас зберігає зручність роботи завдяки наявності автоматичної інференції типів.

Dart характеризується низьким порогом входу. Синтаксис цієї мови поєднує риси мов C-подібної сім'ї, що полегшує її освоєння розробникам, які мають досвід роботи з Java, JavaScript чи C#. Завдяки цьому час на адаптацію до середовища розроблення істотно зменшується.

Серед додаткових переваг можна виокремити гнучку систему асинхронного програмування, засновану на механізмах Future та Stream. Це забезпечує ефективну роботу з потоками даних у реальному часі та сприяє створенню продуктивних інтерфейсів.

У сукупності зазначені переваги роблять мову Dart конкурентною альтернативою у сфері кросплатформної мобільної розробки, забезпечують високу продуктивність, стабільність та передбачуваність програмного продукту і значно спрощують процес його розроблення та супроводження.

2.3 Інструментальні компоненти екосистеми Flutter та їх роль у розробленні мобільних застосунків

Екосистема Flutter та мови Dart охоплює широкий спектр бібліотек і фреймворків, призначених для оптимізації процесу розроблення, підвищення продуктивності роботи застосунків та покращення загального користувацького досвіду. Завдяки активному розвитку спільноти й підтримці з боку Google, Flutter забезпечує гнучку та масштабовану інфраструктуру для створення мобільних, веб та десктопних застосунків. Бібліотеки, доступні в офіційному репозиторії pub.dev, охоплюють роботу з інтерфейсом, мережеву взаємодію, керування станом, локальне зберігання даних, аналітику, інтеграцію з апаратними можливостями пристрою, локалізацію та мультимедійні функції. Сукупність

таких рішень робить платформу Flutter однією з найбільш повних екосистем для кросплатформної розробки.

Бібліотеки керування станом

Керування станом є ключовим елементом у побудові реактивних інтерфейсів, і Flutter пропонує широку палітру бібліотек для реалізації цього аспекту. Найпоширенішими є:

- Provider – офіційно рекомендована бібліотека для побудови архітектури та управління станом. Забезпечує просту інтеграцію з деревом віджетів та підтримує реактивне оновлення даних.
- Riverpod – сучасне рішення, що пропонує більш гнучку та безпечну модель керування станом, усуває обмеження контексту і підвищує масштабованість архітектури.
- Bloc (Business Logic Component) – бібліотека, що реалізує патерн BLoC і забезпечує чітке розділення бізнес логіки та інтерфейсу. Підходить для великих застосунків із високими вимогами до надійності [16].
- MobX – реактивна бібліотека, орієнтована на простоту опису станів і автоматичне відстеження залежностей [17].

Використання цих рішень дає змогу забезпечити передбачуваність змін стану, уникнути надмірного дублювання логіки та підвищити якість коду, що є критичним для довготривалих та масштабованих проєктів.

Бібліотеки для роботи з інтерфейсом (UI компоненти)

Flutter має власний високопродуктивний рушій рендерингу, однак бібліотеки інтерфейсних компонентів дозволяють розширювати його можливості та формувати сучасний користувацький досвід:

1. Flutter Spinkit – набір анімацій завантаження, що використовується для покращення візуального сприйняття у місцях тимчасової затримки.
2. Lottie – дозволяє інтегрувати анімовані вектори, створені у After Effects, забезпечуючи якісні анімаційні переходи при мінімальному навантаженні на продуктивність.

3. Shimmer – реалізує ефект "поскелетонної" анімації, коли контент ще завантажується.

4. Cupertino Icons – набір іконок у стилі iOS, що дозволяє зберегти єдність інтерфейсів на різних платформах.

Ці інструменти сприяють створенню естетично привабливих інтерфейсів, зменшують ефект очікування та допомагають підвищити загальний рівень задоволеності користувачів.

Бібліотеки для роботи з даними та локальним зберіганням

Організація локального сховища та робота з даними є важливою складовою мобільних застосунків. Найпоширеніші бібліотеки включають:

- Shared Preferences – зберігання простих налаштувань користувача у вигляді пар ключ значення [18].
- Hive – високопродуктивна NoSQL база даних, орієнтована на локальні офлайн сценарії [19].
- Drift (Moor) – реляційна база даних з підтримкою SQL, типів та генерації коду.
- ObjectBox – орієнтована на обєкти база даних з високою швидкодією.

Застосування таких рішень сприяє скороченню часу доступу до даних і створює можливість роботи в офлайн режимі, що є важливим для мобільних застосунків з динамічним контентом.

Бібліотеки для мережевої взаємодії та інтеграції зі сторонніми сервісами

У сучасних інформаційних системах важливим аспектом є обмін даними через мережу. Flutter пропонує низку інструментів:

- Dio – потужний HTTP клієнт з підтримкою перехоплювачів, логування, оброблення помилок, формованих запитів та завантаження файлів.
- Http – легка бібліотека для простих мережевих операцій.
- Firebase SDKs – забезпечують автентифікацію, хмарне зберігання даних, push повідомлення, аналітику та інші хмарні функції [20].

Використання Firebase особливо актуальне для мобільних чатів, систем реального часу та застосунків, що вимагають синхронізації між користувачами.

Бібліотеки для мультимовності та локалізації

Адаптація інтерфейсу під різні мовні стандарти є важливою частиною глобалізації програмного продукту:

- Intl – надає засоби форматування дат, чисел та текстових ресурсів.
- Easy Localization – спрощує організацію мультимовних файлів та їх підключення.
- Translator – дозволяє виконувати автоматичний переклад тексту через API, що є корисним у чатах або навчальних застосунках.

Ці бібліотеки допомагають забезпечити коректне відображення інтерфейсу та покращити доступність застосунку для різних груп користувачів.

Бібліотеки для роботи з анімаціями та графікою

Інтерактивність та візуальна привабливість інтерфейсу значною мірою визначають сприйняття програми. Flutter підтримує:

- Rive – створення інтерактивних анімацій з можливістю реагування на дії користувача в реальному часі.
- Flare – інструмент для створення векторних анімацій та їх інтеграції з Flutter.
- Animated Widgets – вбудовані інструменти Dart для створення плавних переходів без додаткових залежностей.

Ці рішення дозволяють формувати інтерфейс високої якості без значного збільшення складності розроблення.

Утилітарні бібліотеки та допоміжні інструменти

Серед бібліотек, що виконують службові функції, можна виділити:

- Uuid – генерація унікальних ідентифікаторів.
- Shared Preferences – збереження невеликих наборів даних.
- Path Provider – доступ до системних директорій.
- Clipboard – взаємодія з буфером обміну.

Такі бібліотеки спрощують вирішення повсякденних технічних задач та дозволяють розробникам зосередитись на бізнес логіці.

2.4 Вибір моделі розробки програмного забезпечення

Розроблення інформаційної системи для комунікації користувачів є багатокомпонентним процесом, який потребує чіткої організації, координації дій команди та ефективного управління етапами проєкту. Вибір відповідної моделі розробки програмного забезпечення суттєво впливає на якість кінцевого продукту, швидкість його створення та можливість масштабування.

У сучасній практиці розроблення ПЗ найчастіше застосовують такі моделі: водоспадну, V-подібну, ітеративну, інкрементну та гнучкі методології Agile [21]. Кожна з них має свої переваги та недоліки, тому вибір залежить від складності проекту, вимог замовника та досвіду команди.

1. Водоспадна модель

Водоспадна модель є одним із найстаріших підходів до розроблення програмного забезпечення. Вона передбачає послідовне виконання етапів:

1. аналіз вимог;
2. проєктування архітектури системи;
3. реалізація (написання коду);
4. тестування;
5. впровадження;
6. супровід і оновлення.

Основною перевагою водоспадної моделі є її простота, чітка структура та можливість попереднього планування. Проте вона має й істотні недоліки – зокрема, обмежену гнучкість щодо змін вимог та необхідність завершення кожного етапу перед переходом до наступного. Через це модель не підходить для динамічних проектів, де вимоги користувача можуть змінюватися в процесі розроблення.

2. Модель V-подібного процесу

Модель V-подібного процесу (рис. 2.5) є розвитком водоспадної моделі, у якій наголошується на взаємозв’язку між етапами розроблення та тестування.

Для кожного етапу розробки передбачено відповідну фазу перевірки – від модульного до приймального тестування.

Цей підхід забезпечує високий рівень контролю якості програмного продукту та раннє виявлення помилок. Проте, як і у водоспадній моделі, тут бракує гнучкості – внесення змін після початку розробки є складним і ресурсозатратним.

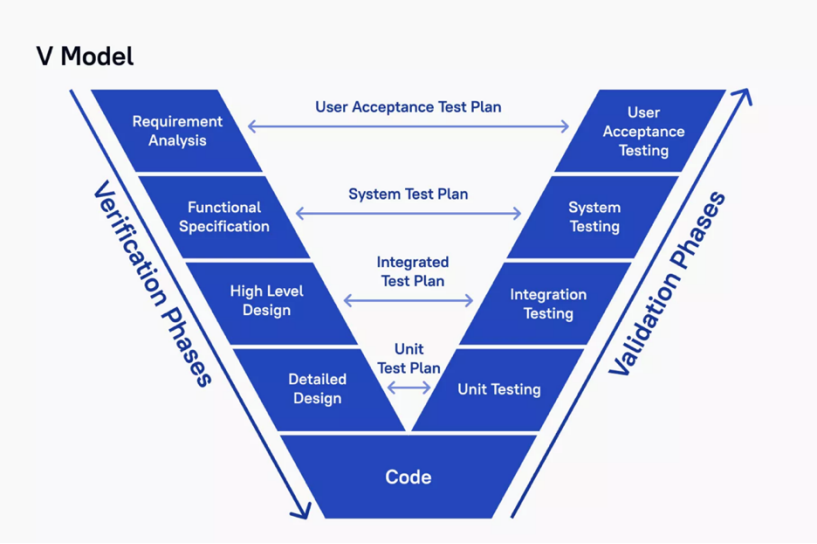

Рисунок 2.5 – Приклад ітеративної моделі розробки

3. Ітеративна модель

Ітеративна модель (рис. 2.6) ґрунтується на багаторазовому повторенні циклів розробки, де кожна ітерація включає етапи аналізу, проєктування, реалізації, тестування та оцінювання. Отримані результати кожного циклу дозволяють поступово вдосконалювати продукт і адаптувати його до змінних вимог користувачів.

Перевагою ітеративного підходу є його гнучкість і можливість раннього виявлення недоліків. Завдяки поступовому вдосконаленню продукту користувачі отримують можливість надавати зворотний зв’язок ще до завершення всієї розробки [22].

Рисунок 2.6– Приклад ітеративної моделі розробки

4. Гнучкі методології (Agile)

Серед сучасних методів управління розробленням програмного забезпечення найбільш популярними є гнучкі методології, зокрема Scrum та Kanban (рис. 2.7). Їхня ключова особливість – поділ процесу розроблення на короткі цикли (спринти), у межах яких команда створює функціональні частини продукту, тестує їх і отримує зворотний зв’язок від замовника.

Основні принципи Agile:

- ітераційність і безперервне вдосконалення продукту;
- гнучке реагування на зміни вимог;
- постійна комунікація між розробниками, замовниками та користувачами;
- регулярне отримання відгуків після кожного спринту;
- орієнтація на кінцеву цінність для користувача.

Методологія Scrum, обрана для даного проєкту, є ефективною для кросплатформних мобільних застосунків, створених за допомогою Flutter та REST API. Вона забезпечує швидку адаптацію до змін і дозволяє координувати роботу над клієнтською та серверною частинами системи одночасно.

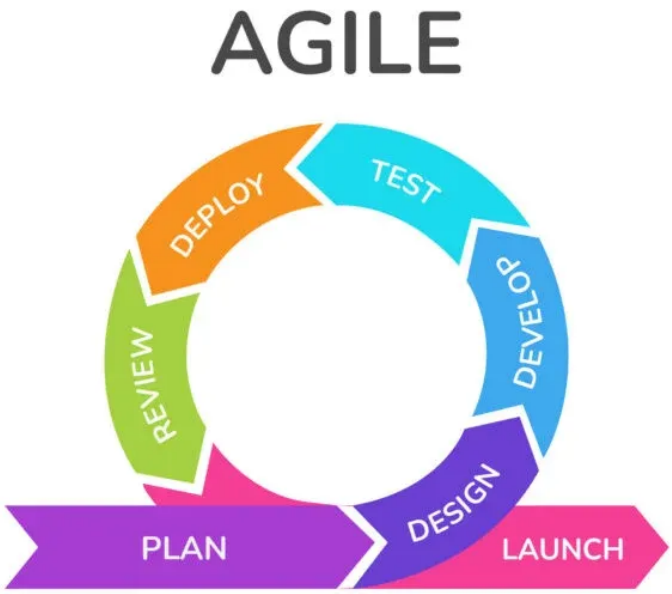

Рисунок 2.7– Приклад моделі розробки Agile

Після аналізу наявних моделей для розроблення інформаційної системи комунікації користувачів було обрано гнучку модель Scrum, що належить до методологій Agile. Її застосування дозволяє ефективно організувати процес розробки, швидко реагувати на зміни вимог, забезпечити постійний зворотний зв'язок і поетапне вдосконалення продукту. Такий підхід є оптимальним для проєктів, реалізованих із використанням Flutter для клієнтської частини та REST API для серверної взаємодії.

2.5 Проектування інтерфейсу

Проєктування інтерфейсу користувача (UI) є одним із найважливіших етапів розроблення інформаційної системи, оскільки саме через інтерфейс здійснюється взаємодія користувача із системою. Якісний інтерфейс забезпечує зручність, швидкість виконання дій та підвищує рівень задоволення користувачів. У процесі створення інформаційної системи для комунікації користувачів було застосовано сучасні принципи UI/UX-дизайну, що відповідають вимогам мобільних додатків на базі Flutter [23].

Окрім базових принципів UI/UX, під час проєктування інтерфейсу було враховано рекомендації Human Interface Guidelines та Material Design 3, які орієнтовані на створення дружнього до користувача середовища. Це дозволило сформувати інтерфейс, який залишається інтуїтивним навіть для користувачів без попереднього досвіду використання чат-застосунків.

Основні принципи проектування інтерфейсу:

1. Зручність використання.

Ключовим аспектом під час розроблення інтерфейсу є простота і зрозумілість у взаємодії. Елементи навігації та керування розташовані інтуїтивно, а виконання основних дій (перегляд повідомлень, відправлення повідомлення, редагування профілю) не потребує зайвих переходів між екранами.

2. Послідовність.

Дизайн системи побудовано з урахуванням єдиних візуальних і функціональних стандартів. Кнопки, заголовки, іконки та кольори повторюються у всіх розділах застосунку, що створює послідовність і спрощує сприйняття інтерфейсу користувачами.

3. Зворотний зв'язок.

Система надає користувачеві оперативний зворотний зв’язок у вигляді сповіщень або візуальних індикаторів. Наприклад, після надсилання повідомлення або зміни налаштувань профілю з’являється повідомлення про успішне виконання дії.

4. Естетика і мінімалізм.

Інтерфейс побудований за принципами мінімалізму – використано обмежену кольорову гаму та сучасну типографіку. Це дозволяє користувачеві зосередитися на контенті та основному функціоналі (рис. 2.8).

Рисунок 2.8 – Приклад основної кольорової гами додатку

5. Доступність.

Система розроблена з урахуванням різних сценаріїв використання, включно з адаптацією під різні розміри екранів і можливість масштабування тексту.

6. Адаптивність та масштабованість інтерфейсу.

Інтерфейс системи розроблений таким чином, щоб коректно відображатися на різних моделях мобільних пристроїв, враховуючи зміну роздільної здатності, орієнтації екрана та системних налаштувань. Для цього використано адаптивні механізми Flutter, зокрема MediaQuery та LayoutBuilder, що дозволяють автоматично підлаштовувати компоненти під доступний простір. Підхід відсоткового масштабування інтерфейсу замість фіксованих значень сприяє правильному відображенню елементів на пристроях різних розмірів. Також забезпечено підтримку системних особливостей платформ Android та iOS, включно з безпечними зонами екрана та системним масштабуванням тексту. Тестування інтерфейсу проводилося на смартфонах і планшетах кількох типів, що підтвердило стабільність роботи UI у різних умовах.

7. Оптимізація продуктивності UI.

Під час розроблення інтерфейсу було враховано необхідність забезпечення плавної роботи навіть у випадках, коли користувач взаємодіє з великими списками чатів та повідомлень. Для підвищення швидкодії застосовано механізм лінивого завантаження елементів списку, кешування останніх повідомлень та оптимізацію кількості перерисовок через використання const-конструкторів і керування станом за допомогою Provider та Streams. Це дозволило суттєво зменшити навантаження на систему та забезпечити комфортну взаємодію користувача з великими обсягами даних.

8. Формування дизайну відповідно до ролі користувача.

Інтерфейс системи враховує різні сценарії взаємодії залежно від ролі користувача. Реалізовано відмінності у відображенні навігаційних елементів, меню та статусної інформації для забезпечення максимальної інформативності. У системі передбачено відображення статусу онлайн/офлайн співрозмовника, індикаторів доставки та прочитання повідомлень, а також різне кольорове оформлення власних і отриманих повідомлень, що полегшує візуальне сприйняття контенту.

9. Робота з анімаціями та мікровзаємодіями.

Для підвищення якості користувацького досвіду застосунок використовує плавні анімації переходів між екранами, мікроанімації під час надсилання повідомлень, анімовані індикатори завантаження та ефекти появи нових елементів. Такі мікровзаємодії роблять інтерфейс живим, покращують зворотний зв’язок і створюють відчуття природності поведінки системи, що позитивно впливає на загальне сприйняття продукту.

11. Інтерактивні елементи управління.

У застосунку реалізовано широкі можливості взаємодії з інтерфейсом за допомогою жестів та контекстних дій. Користувач може видаляти або цитувати повідомлення за допомогою свайпів, використовувати контекстні меню для копіювання чи пересилання повідомлень, а також працювати з динамічними елементами, які змінюють свій стан залежно від активності користувача. Поля

введення повідомлень автоматично розширюються при наборі тексту, що забезпечує більшу зручність під час комунікації.

Методи проектування інтерфейсу

1. Аналіз користувацьких сценаріїв.

На початковому етапі було проаналізовано типові сценарії використання системи – реєстрація, надсилання повідомлення, перегляд чату, редагування профілю. Це дозволило визначити ключові точки взаємодії користувача з додатком.

2. Створення ескізів та прототипів.

Ескізи інтерфейсу (wireframes) створювалися у Figma для швидкої візуалізації логіки роботи додатку. На їх основі було підготовлено інтерактивний прототип, що відображає навігацію між екранами та функціональну структуру (рис. 2.9).

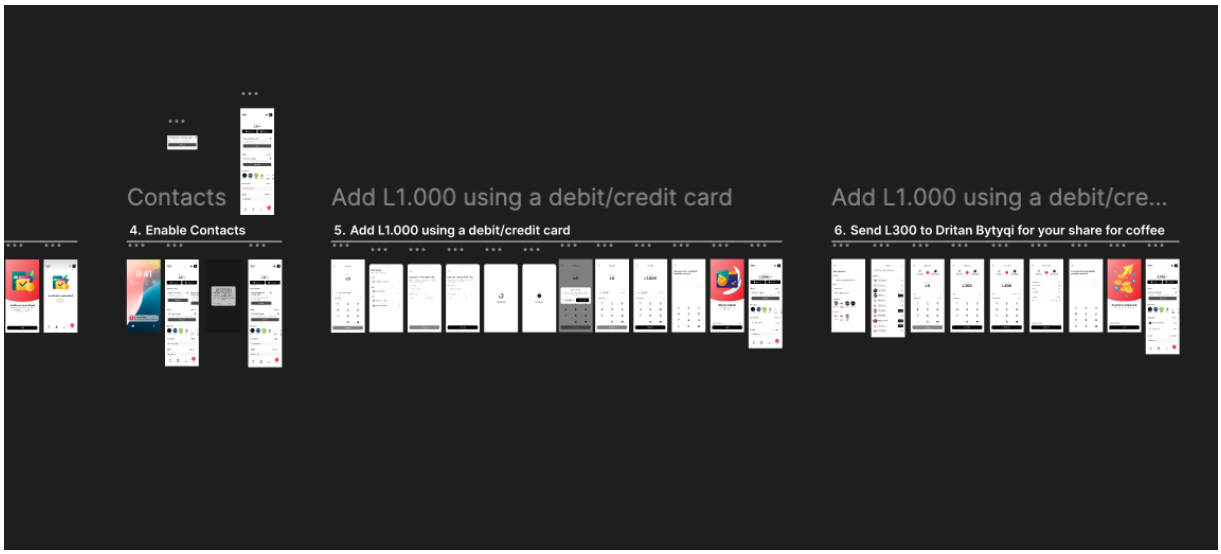

Рисунок 2.9 – Приклад прототипу за допомогою ресурсу Figma

3. Тестування з користувачами.

Прототип було протестовано серед невеликої, але репрезентативної групи потенційних користувачів, до якої увійшли як технічно підготовлені респонденти, так і користувачі без попереднього досвіду роботи з подібними застосунками. Це дозволило оцінити інтерфейс із різних точок зору та зібрати

різноманітний спектр зауважень. Тестування проводилося у формі модераційних сесій, під час яких учасникам пропонувалося виконати низку типових дій: пройти реєстрацію, авторизуватися, знайти співрозмовника, відкрити чат, надіслати та отримати повідомлення, змінити налаштування профілю. Усі дії супроводжувалися спостереженням за поведінкою користувачів, фіксацією часу виконання завдань та аналізом їхніх коментарів.

Особливу увагу було приділено виявленню моментів, де користувачі відчували труднощі: не могли знайти необхідну кнопку, неправильно інтерпретували призначення елемента або витрачали більше часу, ніж очікувалося. Саме такі ситуації свідчать про потребу вдосконалення інтерфейсу. Зокрема, було помічено, що частина користувачів очікувала інший порядок розташування елементів на головному екрані, що вплинуло на подальше уточнення структури. Також було оптимізовано поведінку полів введення та допоміжні підказки, які сприяють більш швидкому орієнтуванню в інтерфейсі.

Аналіз тестування з користувачами проводився з використанням методів UX-досліджень, таких як "think-aloud protocol" у прототипі Figma та оцінка загального рівня задоволеності за шкалою SUS (System Usability Scale). Середній показник зручності перевищив 82 бали зі 100, що свідчить про високий рівень інтуїтивності застосунку. Крім того, учасники позитивно відзначили мінімалістичний стиль, простоту навігації та логічність переходів між екранами.

Таблиця 2.2 – Результати проведеного тестування до та після покращень

Показник	До покращень (версія прототипу до UX-аналізу)	Після покращень (на основі результатів тестування)
Інтуїтивність інтерфейсу	Користувачі повідомляли про неочевидність частини переходів та розміщення окремих елементів.	Покращена логіка навігації, спрощені переходи; інтерфейс став інтуїтивно зрозумілим для більшості користувачів.

Продовження таблиці 2.2

Продуктивність взаємодії	Під час тестування користувачі витрачали більше часу на пошук деяких функцій.	Зменшено кількість зайвих дій; середній час виконання базових сценаріїв скоротився.
Навігація між екранами	Структура навігації містила надмірну кількість кроків у деяких сценаріях.	Навігаційні шляхи оптимізовано: переходи стали логічнішими та швидшими.
Зовнішній вигляд (UI-сприйняття)	Користувачі зазначали потребу в упорядкуванні елементів та покращенні візуальної гармонії.	Мінімалістичний стиль був високо оцінений; інтерфейс став візуально узгодженим.
Рівень задоволеності користувача (SUS)	Початковий рівень оцінювався нижче 82 (середня оцінка не відповідала високим стандартам UX).	Середній SUS-показник підвищився до 82/100, що свідчить про високу зручність застосунку.
Зворотний зв'язок від користувачів	Вказували на «точки тертя» — місця, де дія була незрозумілою або вимагала уточнення.	Після змін користувачі відзначили «логічність», «простоту» та «передбачуваність» взаємодії.
Загальне враження	Додаток сприймався як функціональний, але такий, що потребує вдосконалення з боку UX.	Додаток оцінений як зручний, легкий у використанні та сучасний; загальний досвід покращився.

На основі отриманих результатів було внесено низку доопрацювань: скорочено кількість кроків для виконання найбільш частих дій, підсилено візуальні акценти на важливих елементах, покращено контрастність окремих компонентів для роботи в умовах різного освітлення, а також адаптовано анімації для кращого сприйняття. Після внесення змін прототип було повторно протестовано, і повторне дослідження підтвердило покращення показників взаємодії. Таким чином, тестування з користувачами стало важливим етапом,

який забезпечив об’єктивну оцінку якості інтерфейсу, дав змогу своєчасно усунути недоліки та сформував основу для створення фінальної версії UI, що максимально відповідає потребам цільової аудиторії.

2.6 Висновки до розділу

Для час розроблення клієнтської частини інформаційної системи для комунікації користувачів було обрано фреймворк Flutter, який забезпечує високу продуктивність і кросплатформність додатку. Використання фреймворку Flutter стало ключовим рішенням, оскільки він забезпечує високу продуктивність, узгодженість роботи на різних мобільних платформах та широкі можливості для створення сучасних інтерфейсів. Завдяки єдиному коду для Android та iOS стало можливим оптимізувати витрати часу на розроблення та прискорити ітерації вдосконалення застосунку. У якості методології розробки обрано Scrum зі спринтами тривалістю два тижні, що дозволяє поетапно тестувати та вдосконалювати інтерфейс. Для створення дизайну та інтерактивних прототипів було використано Figma, що дозволило розробити цілісну систему інтерфейсів із дотриманням принципів юзабіліті, кольорової гармонії та візуальної ієрархії. Подальша реалізація інтерфейсних рішень здійснювалася за допомогою Material Design Widgets, що забезпечило узгодженість елементів, зручність навігації та доступність інтерфейсу для користувачів із різними рівнями цифрової компетентності.

Крім того, у процесі розроблення значну увагу приділено оптимізації інтерактивної взаємодії, адаптивності інтерфейсу та забезпеченню стабільної роботи у межах мобільного середовища. Тестування функціональних модулів дозволило підтвердити коректність відображення екранів, швидкість роботи інтерфейсу та відповідність вимогам користувацького досвіду.

У результаті застосовано ефективне поєднання сучасних технологій, методів проєктування та підходів до управління розробкою, що забезпечило

створення зручного, естетично привабливого та функціонального інтерфейсу мобільної системи комунікації. Отриманий результат демонструє готовність клієнтської частини до масштабування, подальшого вдосконалення та повноцінної взаємодії з серверною частиною системи.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для проектування та розробки інформаційної системи для комунікації користувачів на основі технологій Flutter та REST API необхідно чітко визначити функціональні та нефункціональні вимоги. Розгляд вимог має здійснюватися з урахуванням ролей, які присутні у системі, зокрема адміністратора та звичайного користувача.

Функціональні вимоги для користувача з роллю адміністратора включають можливість реєстрації нових користувачів та редагування їхніх профілів, імпорт користувачів з корпоративної ERP системи шляхом завантаження CSV файлу, формування та керування політиками доступу і прав у межах платформних каналів, створення і видалення групових чи приватних каналів комунікації, модерацію контенту з можливістю перегляду та видалення повідомлень, а також керування налаштуваннями сповіщень і параметрами збереження даних. Додатково адміністратор повинен мати можливість перегляду статистичної звітності щодо активності користувачів і каналів, фільтрування звітів за заданими критеріями, перегляду логів подій та налаштування глобальних параметрів системи.

Функціональні вимоги для користувача з роллю працівника (звичайного користувача) передбачають реєстрацію та автентифікацію в системі, перегляд і редагування власного профілю, можливість надсилання та отримання миттєвих повідомлень у приватних або групових чатах, перегляд історії повідомлень та пошук по цій історії, участь у групових каналах з формуванням списків учасників, можливість обміну файлами з урахуванням обмежень на розмір і типи вкладень, отримання push сповіщень та управління налаштуваннями сповіщень, а також перегляд онлайн-статусу інших користувачів і запланованих подій у загальному календарі спільноти.

До нефункціональних вимог відносяться адаптивність інтерфейсу, яка забезпечує коректне відображення та зручність використання на різних

пристроях та роздільних здатностях екранів; кросплатформеність і кросбраузерність у разі наявності веб-інтерфейсу; висока інформаційна безпека, що передбачає захищене зберігання облікових даних, хешування паролів, застосування протоколу захищеного з'єднання для передачі даних та контроль доступу за ролями; продуктивність і масштабованість, що гарантують мінімальні затримки при обміні повідомленнями та стійку роботу при зростаючій кількості одночасних користувачів; надійність і стійкість до збоїв, включаючи механізми повторних спроб запитів, чергування повідомлень і коректну обробку відключень мережі; та можливість локального кешування конфігурацій для роботи в умовах нестабільного з’єднання.

Спрощений алгоритм роботи інформаційної системи комунікації описує послідовність основних операцій: реєстрація або автентифікація користувача, ініціалізація сесії клієнтського застосунку і отримання базової конфігурації з серверу, вибір або створення каналу комунікації, надсилання повідомлення клієнтом до серверу через REST інтерфейс або через канал реального часу, збереження і індексація повідомлення в базі даних серверу, розповсюдження повідомлення іншим учасникам за допомогою push сповіщень або WebSocket, а також можливість адміністрування каналів та перегляду аналітики

Використання UML-діаграм є необхідним етапом проектування, оскільки вони забезпечують наочну візуалізацію внутрішньої структури системи і послідовності її робочих процесів, а також слугують підґрунтям для подальшого впровадження і тестування.

3.1 Проектування бази даних та моделей

Важливим етапом розробки інформаційної системи для комунікації користувачів є проектування бази даних [25]. У межах цього підрозділу сформовано модель бази даних, описано її основні сутності та атрибути, а також обґрунтовано структуру та характер зв’язків між сутностями. Це забезпечує

узгоджене та ефективне функціонування системи, а також коректну взаємодію мобільного застосунку, розробленого у Flutter, із серверною частиною через REST API.

Сутність у моделюванні баз даних є концептуальним елементом, що представляє об’єкт предметної області, інформацію про який необхідно зберігати. Кожна сутність характеризується набором атрибутів, що визначають її властивості та особливості [26].

Атрибут – це характеристика сутності, що описує її окремі аспекти. Для кожного атрибуту визначаються назва, тип даних та наявні обмеження, зокрема унікальність значення, заборона на порожнє значення або використання атрибуту як ключа сутності.

Зв’язки між сутностями описують, яким чином об’єкти в базі даних пов’язані між собою. Вони реалізуються через первинні та зовнішні ключі. У реляційній моделі використовують такі типи зв’язків:

– один до одного (1:1) – кожному запису однієї сутності відповідає один запис іншої;

– один до багатьох (1:N) – одному запису однієї сутності відповідають кілька записів пов’язаної сутності;

– багато до багатьох (M:N) – багато записів однієї сутності можуть бути пов’язані з багатьма записами іншої, що реалізується через проміжну сутність.

Сутність «користувач» (user) призначена для зберігання реєстраційних та основних профільних даних користувачів, які взаємодіють у системі. Атрибути цієї сутності наведено у таблиці 3.1.

Таблиця 3.1– Атрибути сутності «користувач» (user)

Назва атрибуту	Тип даних	Обмеження	Призначення
id	ObjectId	первинний ключ, not null	Ідентифікатор користувача
firstName	String	not null	Ім’я користувача
lastName	String	not null	Прізвище користувача
email	String	not null, унікальне	Адреса електронної пошти

Продовження таблиці 3.1

createdAt	Date	not null	Дата та час створення облікового запису
password	String	not null	Хешований пароль

Сутність «розмова» (чат, канал) відображає логічний простір спілкування користувачів. Це може бути приватний діалог між двома користувачами або групова розмова.

Таблиця 3.2 – Атрибути сутності «розмова» (conversation)

Назва атрибуту	Тип даних	Обмеження	Призначення
id	ObjectId	первинний ключ, not null	Унікальний ідентифікатор розмови
createdBy	ObjectId	зовнішній ключ, not null	Користувач, який створив розмову
createdAt	Date	not null	Дата та час створення
lastMessageAt	Date	допускає null	Дата та час останнього повідомлення у розмові

Сутність «повідомлення» є центральною для системи для комунікації [27] оскільки саме через неї реалізується обмін текстовими даними між користувачами (табл. 3.3).

Таблиця 3.3 – Атрибути сутності «повідомлення» (message)

Назва атрибуту	Тип даних	Обмеження	Призначення
id	ObjectId	первинний ключ, not null	Унікальний ідентифікатор повідомлення
conversation	ObjectId	зовнішній ключ, not null	Розмова, до якої належить повідомлення
sender	ObjectId	зовнішній ключ, not null	Користувач – відправник повідомлення
text	String	допускає null	Текстовий вміст повідомлення
createdAt	Date	not null	Дата та час відправлення

Сформована структура бази даних забезпечує підтримку основних функцій чат-системи: реєстрації та автентифікації користувачів, створення приватних і групових розмов, обміну повідомленнями та передачі вкладень.

Взаємозв’язки між сутностями реалізують гнучку модель, яка може бути розширена додатковими можливостями, такими як статуси прочитання, push-сповіщення чи зберігання історії змін.

3.2 Розробка серверної частини системи на базі Firebase

У рамках реалізації серверної частини використано платформу Firebase, яка підтримує повноцінний REST API для роботи з даними, автентифікацією, медіафайлами та push-сповіщеннями. Flutter-клієнт взаємодіє з бекендом через офіційні SDK, які є обгорткою над REST API Firebase, що повністю відповідає вимогам теми щодо використання REST-архітектури [28].

Розробка серверної частини інформаційної системи для комунікації користувачів базується на використанні хмарної платформи Firebase, яка забезпечує готову інфраструктуру для автентифікації, зберігання даних, керування файлами та надсилання сповіщень. Такий підхід дає змогу зосередитися на реалізації бізнес-логіки мобільного застосунку на Flutter і мінімізувати витрати на розгортання та супровід окремого серверного програмного забезпечення.

У межах проєкту використано такі основні сервіси Firebase:

Firebase Authentication. Відповідає за реєстрацію та автентифікацію користувачів. Підтримуються входи за електронною поштою і паролем, а також авторизація через сторонніх постачальників (Google, Apple, Facebook). Authentication забезпечує безпечне зберігання облікових даних та інтегрується з іншими сервісами Firebase через унікальний ідентифікатор користувача [29].

Cloud Firestore. Використовується як основне сховище структурованих даних [30] чат-системи: колекції користувачів, профілів, розмов, учасників, повідомлень, вкладень, пристроїв і контактів. Firestore підтримує реактивну модель роботи: мобільний застосунок отримує оновлення в реальному часі при

зміні документів, що дає змогу реалізувати миттєву синхронізацію повідомлень між усіма учасниками розмови.

Firebase Storage. Призначений для зберігання медіафайлів та документів, що передаються у повідомленнях. Повідомлення з вкладеннями містять лише посилання на відповідний об’єкт у Storage, що зменшує обсяг даних у Firestore та оптимізує завантаження великих файлів у мобільному застосунку.

Firebase Cloud Messaging (FCM). Забезпечує надсилання push-сповіщень на мобільні пристрої користувачів. На основі токенів пристроїв, що зберігаються у колекції Device, система може повідомляти користувачів про надходження нових повідомлень, запрошення до групового чату або зміну статусу співрозмовника.

Cloud Functions for Firebase. Використовуються для реалізації серверної бізнес-логіки, яка не повинна виконуватися на клієнтській стороні. Зокрема, за допомогою Cloud Functions реалізуються:

- валідація та нормалізація даних при створенні чи оновленні документів у Firestore;
- розсилання push-сповіщень усім учасникам розмови при надсиланні нового повідомлення;
- оновлення службових полів розмови (останнє повідомлення, час останньої активності);
- реалізація механізмів модерації та фільтрації недопустимого контенту.

Структура проєкту серверної частини в середовищі Firebase представлена на рисунку 3.1. Вона включає:

1. Конфігураційні файли Firebase для підключення мобільного застосунку (рис. 3.1);

2. Правила безпеки Firestore та Storage, які визначають права доступу користувачів до даних залежно від їхнього ідентифікатора та ролі (рис. 3.2).

```
39
40  static const FirebaseOptions web = FirebaseOptions(
41    apiKey: 'AIzaSyAEs2ZWzmo2p_Me2fVOu74VvR6CFosrBCU',
42    appId: '1:924619132236:web:3abc604d83dd48b5f013e9',
43    messagingSenderId: '924619132236',
44    projectId: 'chatty-3e6e5',
45    authDomain: 'chatty-3e6e5.firebaseapp.com',
46    storageBucket: 'chatty-3e6e5.firebasestorage.app',
47    measurementId: 'G-NNBNPFYLHS',
48  );
49
50  static const FirebaseOptions android = FirebaseOptions(
51    apiKey: 'AIzaSyD2B8Wusp_xhoNyMz_nvSgF63zQb9zrq4Y',
52    appId: '1:924619132236:android:f84d390c2e1a38cbf013e9',
53    messagingSenderId: '924619132236',
54    projectId: 'chatty-3e6e5',
55    storageBucket: 'chatty-3e6e5.firebasestorage.app',
56  );
```

Рисунок 3.1 – Налаштування для Веб додатку та Андроїд

```
1  rules_version = '2';
2  service cloud.firestore {
3    match /databases/{database}/documents {
4      match /{document=**} {
5        allow read, write: if request.auth != null;
6      }
7    }
8  }
```

Рисунок 3.2 – Налаштування правил доступу до тестової бази даних

Запропонована архітектура бекенду на базі Firebase забезпечує високий рівень масштабованості, відмовостійкості та безпеки, а також спрощує інтеграцію з мобільним застосунком на Flutter через офіційні клієнтські бібліотеки. Це створює підґрунтя для подальшого розширення функціональності чат-системи без суттєвої зміни її базової структури.

3.3. Розробка клієнтської частини системи

Розробка клієнтської частини інформаційної системи для комунікації користувачів є ключовим етапом, який забезпечує взаємодію користувача із функціональними можливостями застосунку. Мобільна клієнтська частина

розроблена з використанням фреймворку Flutter, що дає змогу створити кросплатформний застосунок з єдиною кодовою базою для Android та iOS. Основною вимогою при розробці інтерфейсу є забезпечення швидкого обміну повідомленнями, інтуїтивної навігації та коректної роботи з Firebase-сервісами в режимі реального часу. На рисунку 3.3 наведено структуру клієнтської частини проєкту, що реалізована в каталозі lib/ мобільного застосунку.

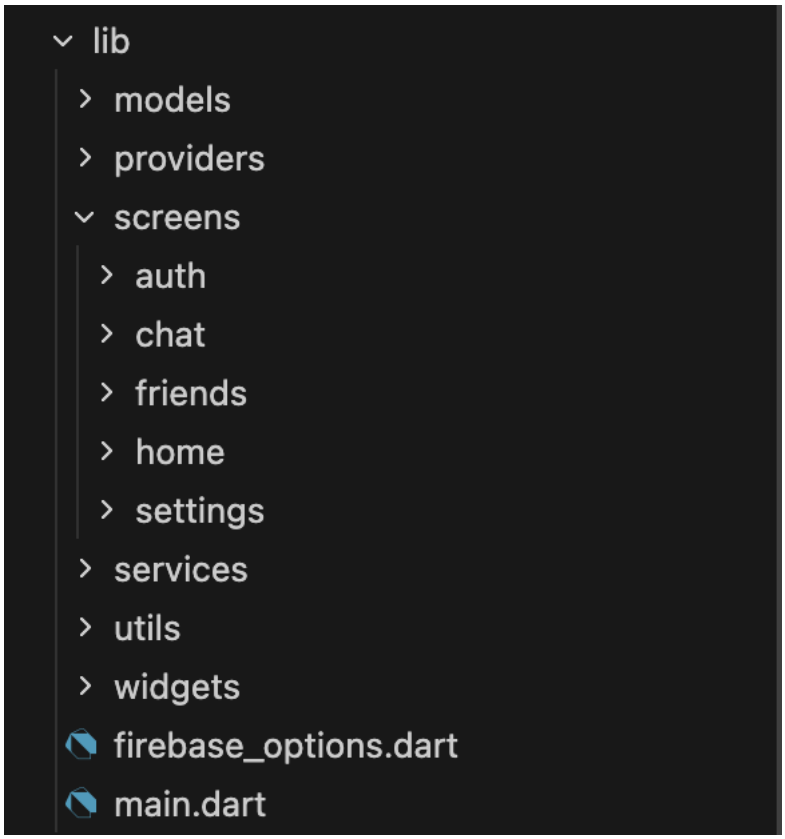

Рисунок 3.3 – Налаштування структура файлів додатку

Опис директорій клієнтської частини проєкту

1. models – містить моделі даних, що описують структури об’єктів у Firestore, зокрема моделі користувача, повідомлення та чату.

2. providers – містить провайдери стану (наприклад, UserProvider, ChatProvider), через які реалізовано управління даними та реакцію інтерфейсу на зміни стану.

3. screens – містить екрани застосунку, згруповані за логічними модулями:

– auth – екрани реєстрації та авторизації;

– chat – список чатів, сторінка чату, екран створення нового чату;

– friends – список друзів, додавання/пошук користувачів;

– home – головний екран;

– settings – налаштування профілю та застосунку.

4. services – реалізує взаємодію з Firebase: FirebaseAuthService, FirestoreChatService, FirebaseStorageService, PushNotificationsService.

5. utils – містить допоміжні класи, валідатори, константи, формувачі дат, стилі тощо.

6. widgets – містить багаторазові UI-компоненти (message bubble, input field, chat tile).

7. firebase_options.dart – конфігураційний файл, який генерується Firebase CLI та містить ключі доступу до проєкту.

8. main.dart – точка входу в застосунок, з ініціалізацією Firebase та декларацією маршрутизації.

Flutter-навігацію реалізовано за допомогою MaterialApp та Navigator.

На рисунку 3.4 наведено фрагмент коду, що відповідає за маршрути застосунку.

```dart
/// Route names as constants
class AppRoutes {
  static const String home = '/home';
  static const String login = '/login';
  static const String phoneLogin = '/phone-login';
  static const String profileCreation = '/profile-creation';
  static const String chat = '/chat';
  static const String chats = '/chats';
  static const String friends = '/friends';
  static const String settings = '/settings';
}
```

Рисунок 3.4 – Фрагмент маршрутизації Flutter-додатку

Для забезпечення обміну даними в режимі реального часу використовується набір сервісів Firebase:

Firebase Authentication – реєстрація, авторизація, відновлення доступу.

Cloud Firestore – зберігання повідомлень, чатів та даних користувача.

Firebase Storage – завантаження та зберігання фотографій профілю та медіафайлів.

Firebase Cloud Messaging (FCM) – надсилання push-сповіщень про нові повідомлення.

На рисунку 3.5 наведено приклад сервісу авторизації.

```
Future<UserCredential?> signInWithEmail({
  required String email,
  required String password,
  required Function(String error) onError,
}) async {
  try {
    return await _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
  } on FirebaseAuthException catch (e) {
    onError(_getAuthErrorMessage(e.code));
    return null;
  } catch (e) {
    onError(e.toString());
    return null;
  }
}
```

Рисунок 3.5 – Фрагмент коду авторизації Firebase Authentication

Для роботи з чатами використовується Firestore Stream, що дозволяє отримувати повідомлення в реальному часі без ручного оновлення сторінки (рисунок 3.6).

Інтерфейс користувача мобільного застосунку розроблено відповідно до принципів зручності, інтуїтивності та адаптивності, що забезпечує комфортну взаємодію з основними функціями системи. Оскільки застосунок реалізує функціонал обміну повідомленнями у режимі реального часу, важливою вимогою до інтерфейсу є мінімальне когнітивне навантаження на користувача,

швидкість доступу до основних елементів та логічна послідовність переходів між екранами.

```
91
92    void _listenToChatUpdates(String chatId) {
93      _currentChatSubscription?.cancel();
94      _currentChatSubscription = _firestoreService.getChatStream(chatId).listen((
95        chat,
96      ) {
97        if (chat != null) {
98          _currentChat = chat;
99          notifyListeners();
100       }
101     });
102   }
```

Рисунок 3.6 – Фрагмент коду стріму повідомлень

Основні екрани системи можна умовно поділити на такі групи:

- екрани автентифікації (login, registration);
- екрани головної навігації (список чатів/друзів, домашній екран);
- екран перегляду діалогу (чат);
- екрани додаткових функцій (пошук і додавання друзів, налаштування).

Нижче наведено детальний опис ключових інтерфейсних компонентів застосунку.

Екран авторизації користувача

Екран авторизації є початковою точкою взаємодії користувача із застосунком. Його функціональність охоплює введення облікових даних, валідацію введеної інформації, відображення можливих помилок та передавання даних у Firebase Authentication для входу.

Інтерфейс побудовано таким чином, щоб користувач мав змогу швидко виконати авторизацію та перейти до функціоналу застосунку. На екрані розміщено поля введення електронної пошти та пароля, кнопку входу та допоміжні елементи («Створити акаунт», «Забули пароль?») (рис. 3.7).

Екран має мінімалістичний дизайн, що знижує відволікаючі чинники та прискорює виконання дії.

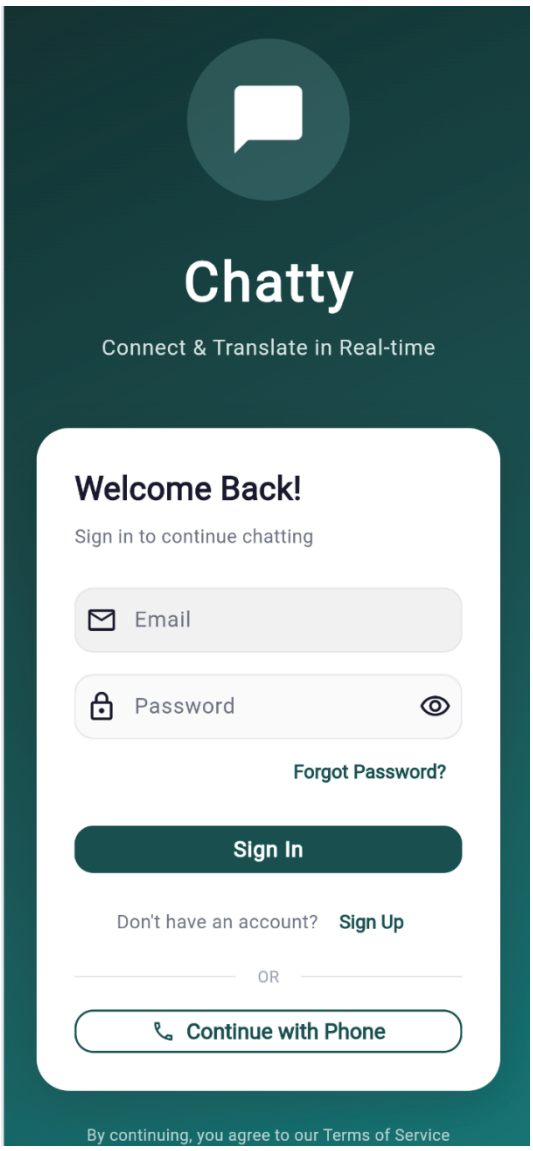

Рисунок 3.7 – Екран авторизації користувача

Головний екран застосунку

Після успішної автентифікації користувач переходить на головний екран, який містить базову навігацію системи.

Головний екран виконує функцію хабу, з якого користувач отримує доступ до списку друзів, активних чатів, налаштувань профілю та можливості створювати нові бесіди.

Цей екран забезпечує такі можливості:

Перегляд списку друзів або активних чатів.

Швидкий перехід до діалогу одним натисканням.

Доступ до меню налаштувань.

Індикатори непрочитаних повідомлень.

Дизайн побудовано з урахуванням принципів Material Design, що забезпечує легку візуальну структуризацію контенту та швидку орієнтацію користувача (рис. 3.8).

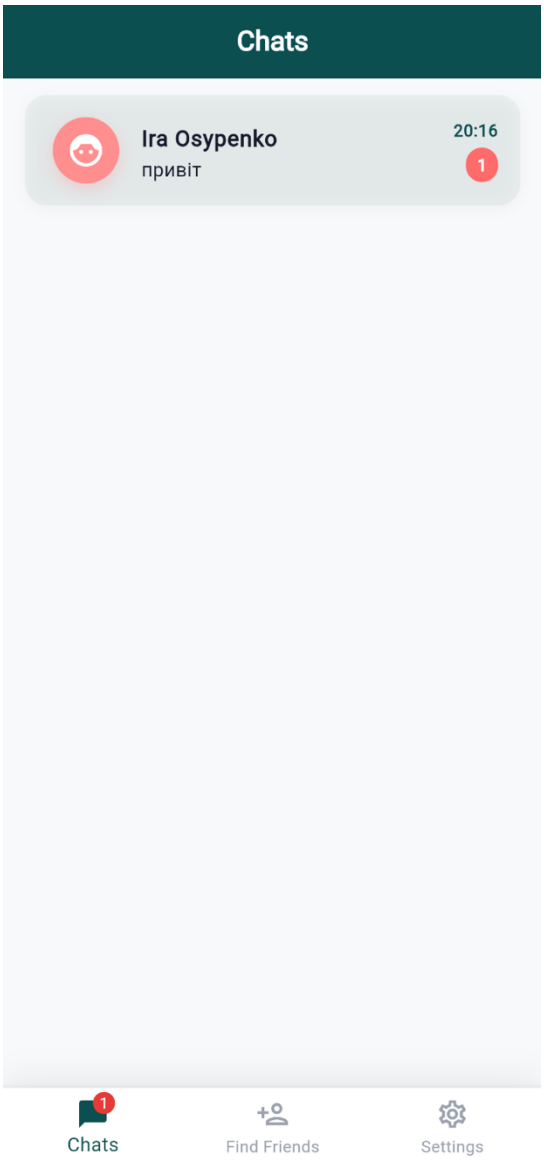

Рисунок 3.8– Головний екран застосунку (список друзів/чатів)

Інтерфейс перегляду чату

Екран перегляду чату є центральним елементом системи. На ньому відображаються всі повідомлення між двома користувачами.

Основні елементи інтерфейсу чату:

Список повідомлень, відсортований у зворотньому хронологічному порядку.

- «Message bubbles» різного стилю для вхідних і вихідних повідомлень.
- Підтримка текстових повідомлень, емодзі та медіафайлів (за потреби).
- Індикація часу відправлення та статусу (доставлено/прочитано).
- Поле введення тексту з можливістю прикріплення файлів та кнопкою надсилання.
- Автоматичне прокручування до останнього повідомлення.

Архітектурно екран інтегровано з Firebase Cloud Firestore через потокові підписки (streams), що забезпечує миттєве оновлення інтерфейсу при надходженні нового повідомлення (рис.3.9).

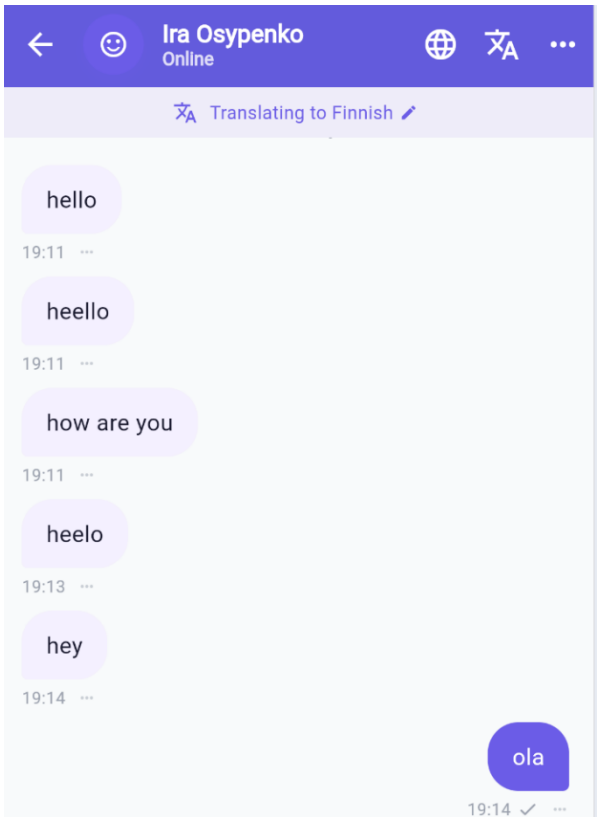

Рисунок 3.9 – Інтерфейс одного чату (message bubbles)

Екран пошуку та додавання друзів

Для забезпечення можливості розширення кола контактів у системі реалізовано окремий екран пошуку й додавання нових друзів.

На цьому екрані користувач може:

- здійснювати пошук інших користувачів за email;

- переглядати базову інформацію профілю знайдених людей;
- переглядати список потенційних контактів або рекомендацій .

Екран побудований зі зручним списком результатів пошуку та інтерактивними елементами, які дозволяють швидко виконати дію (рис. 3.10).

- Взаємодія з Firebase включає:
- отримання списку користувачів з Firestore;
- оновлення списку друзів;

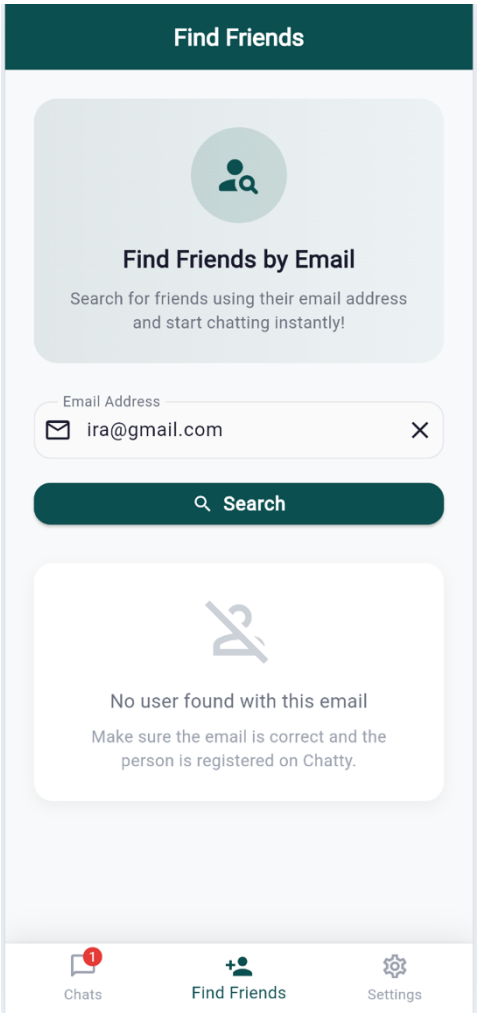

Рисунок 3.10 – Екран додавання та пошуку друзів

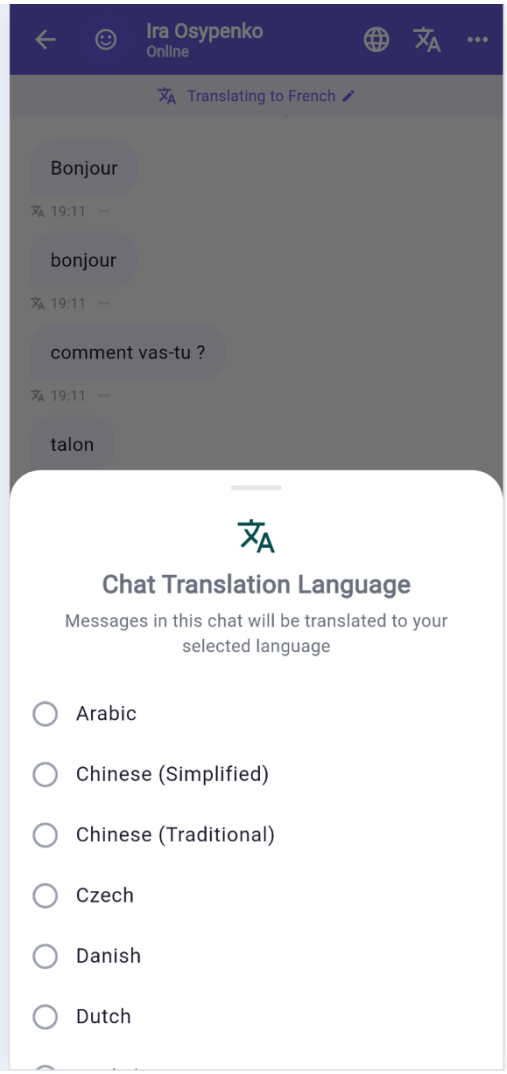

Рисунок 3.11 – Діалогове вікно вибору мови перекладу

Система підтримує автоматичний переклад повідомлень у режимі реального часу.

Після натискання на відповідну іконку відкривається діалогове вікно Chat Translation Language, де користувач може обрати одну з понад десяти доступних мов (Arabic, Chinese, Czech, Finnish тощо).

Обрана мова застосовується миттєво, що значно полегшує спілкування між користувачами, які говорять різними мовами (рис. 3.11).

Екран Settings містить інформацію про профіль користувача: ім’я, email, вік та стать. Звідси також користувач може:

- оновити ім’я та прізвище,
- вказати або виправити вік,

- змінити стать.

Візуальна частина містить вибір аватарів у вигляді іконок, що підвищує персоналізацію профілю (рис.3.12). Передбачені кнопки Save та Cancel, що забезпечують контроль над змінами.

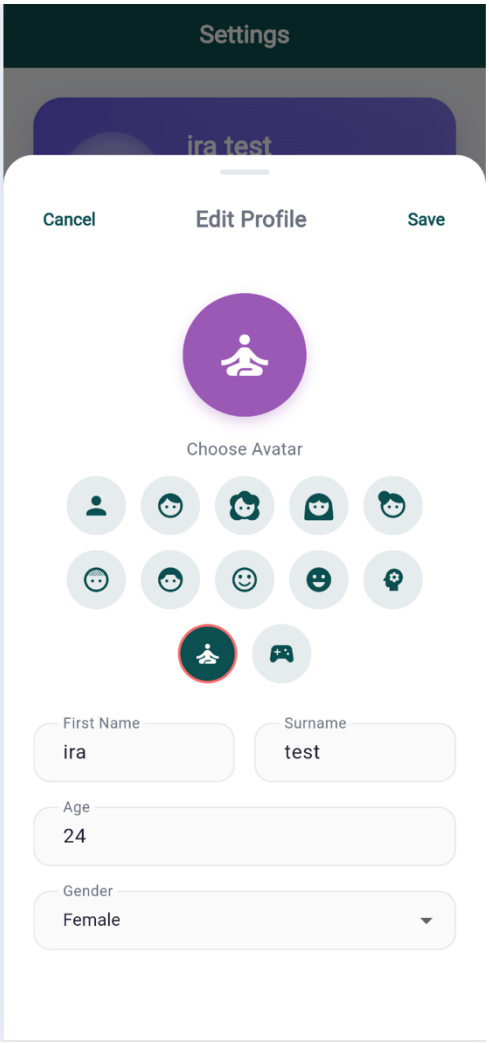

Рисунок 3.12 – Екран редагування профілю користувача

Екран Settings (рис. 3.13) містить інформацію про профіль користувача: ім’я, email, вік та стать. Звідси також відкривається доступ до:

- редагування профілю,
- зміни теми чату,
- перегляду додаткових параметрів.

Інтерфейс структурований та виконаний у спокійному візуальному стилі.

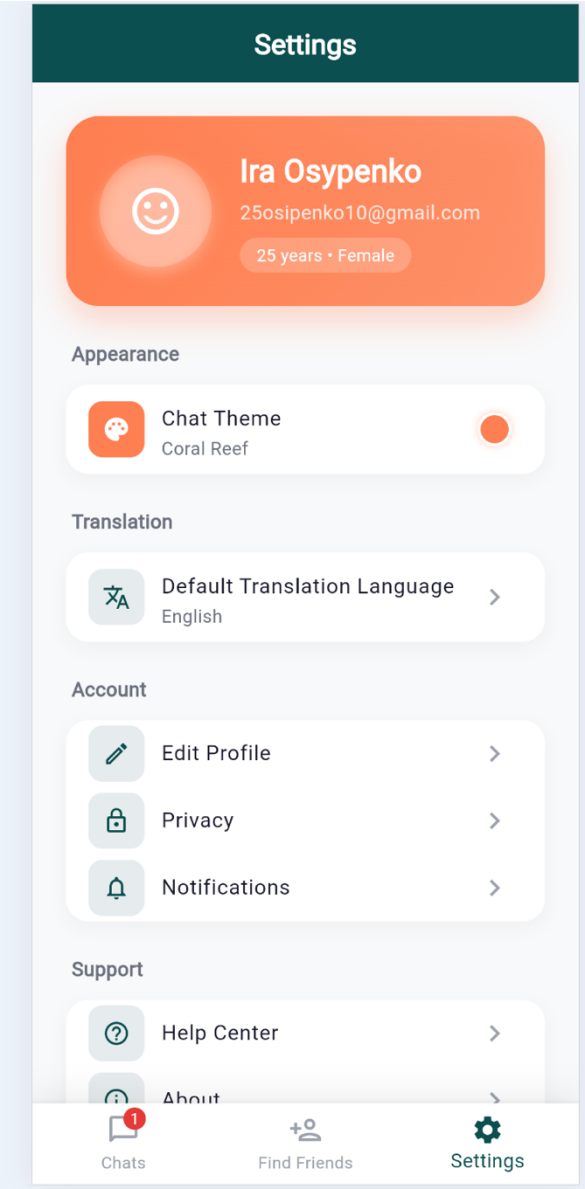

Рисунок 3.13 – Екран вибору теми чату

Система підтримує гнучке налаштування зовнішнього вигляду інтерфейсу: користувач може обрати одну з кількох заздалегідь визначених тем (Ocean Teal, Purple Dream, Rose Pink, Midnight Dark тощо, рис. 3.14).

Кожна тема візуально демонструється у вигляді прикладу "message bubbles", що дозволяє легко обрати бажаний стиль.

Ця функція підвищує персоналізацію системи та покращує комфорт взаємодії.

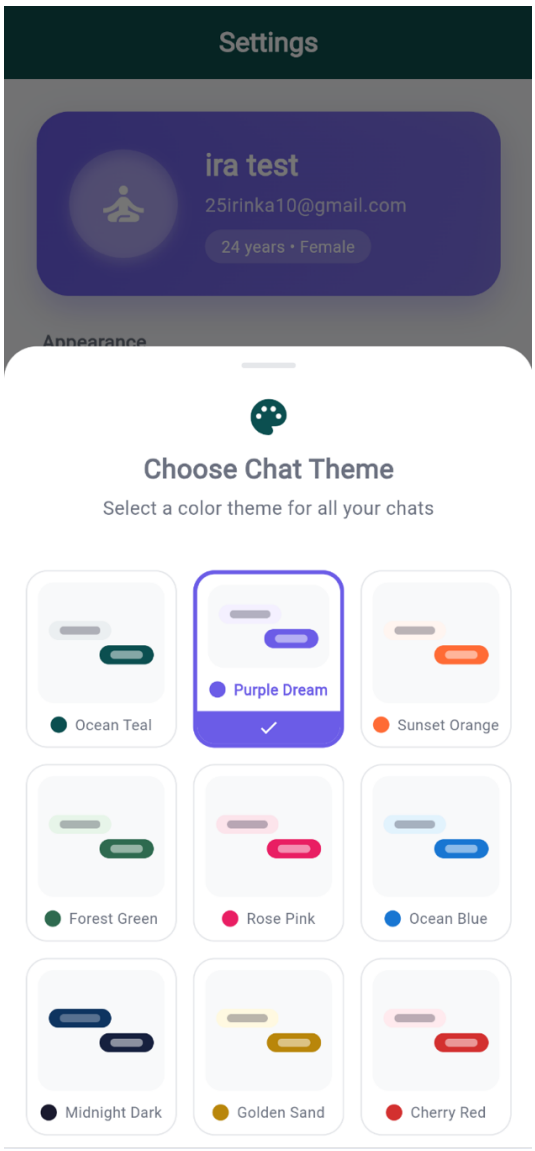

Рисунок 3.14 – Екран налаштувань теми додатку

3.4 Тестування інформаційної системи для комунікації користувачів

Тестування програмного забезпечення є невід'ємною складовою процесу розробки інформаційних систем, оскільки воно забезпечує перевірку коректності функціонування продукту [31], його надійності, стійкості та відповідності вимогам. Проведення тестування дозволяє виявити помилки на ранніх етапах розробки, оцінити взаємодію між клієнтською частиною Flutter-додатка та серверною інфраструктурою Firebase, а також підтвердити працездатність ключових функцій системи у реальних умовах експлуатації.

Основною метою тестування системи для комунікації користувачів є встановлення того, чи виконує програмне забезпечення свої функції відповідно до технічних вимог: чи працює модуль автентифікації, чи коректноу відбувається передача повідомлень у режимі реального часу, чи забезпечено стабільну синхронізацію даних між клієнтами, а також чи дотримані вимоги до безпеки та захисту даних.

У межах розробки системи були застосовані такі види тестування:

Функціональне тестування – перевірка відповідності реалізованих функцій вимогам користувача. До функціонального тестування належало тестування модулів автентифікації, реєстрації, списку друзів, передачі повідомлень, роботи сповіщень та синхронізації чатів у Firebase.

Нефункціональне тестування – оцінювання продуктивності системи, швидкості завантаження чатів, зручності взаємодії з інтерфейсом, узгодженості дизайну та поведінки додатка під різними умовами.

Тестування безпеки – перевірка коректності застосування механізмів захисту Firebase Authentication, правил доступу Firestore Security Rules, а також аналіз можливих векторів несанкціонованого доступу до персональних даних користувачів [32, 33].

Регресійне тестування – повторне тестування системи після внесення змін у бізнес-логіку або інтерфейс. Воно дозволило переконатися, що додані або оновлені функції не спричинили появи нових дефектів.

Етапи процесу тестування

1. Процес тестування відбувався у кілька етапів:

2. Підготовка тестових сценаріїв відповідно до вимог системи.

3. Виконання тестів, фіксація результатів, аналіз взаємодії з Firebase.

4. Аналіз результатів, усунення виявлених дефектів та проблем та після цього повторне тестування.

Для кожної функції було створено тест-кейси, що містять опис ситуації, умови виконання, послідовність дій користувача та очікуваний результат. Це

забезпечує повторюваність тестування та можливість його об’єктивного оцінювання.

Таблиця 3.4 – Тест-кейси для перевірки

№	Тест-кейс
1	2
1	Авторизація користувача Мета: Перевірити можливість входу користувача до системи через Firebase Authentication. Передумови: Користувач має зареєстрований обліковий запис у системі. Кроки: 1. Відкрити екран авторизації. 2. Ввести email та пароль. 3. Натиснути кнопку «Увійти». Очікуваний результат: Користувач успішно проходить авторизацію, відкривається головний екран зі списком чатів.
2	Реєстрація нового користувача Мета: Перевірити можливість створення нового облікового запису. Передумови: Користувач не зареєстрований у системі. Кроки: 1. Відкрити екран реєстрації. 2. Ввести коректні дані (ім’я, email, пароль). 3. Натиснути «Зареєструватися». Очікуваний результат: Створюється новий запис у Firebase Authentication та Firestore, користувач автоматично авторизується.
3	Надсилання повідомлення у чаті Мета: Перевірити можливість надсилання текстових повідомлень у режимі реального часу.

Продовження таблиці 3.4

1	2
3	Передумови: Користувач авторизований, відкритий будь-який чат. Кроки: 1. У полі введення написати повідомлення. 2. Натиснути кнопку «Надіслати». Очікуваний результат: Повідомлення одразу відображається у чаті та зберігається у Firestore.
4	Отримання повідомлень у режимі реального часу Мета: Перевірити, чи відображаються повідомлення, надіслані іншими користувачами, без перезавантаження екрану. Передумови: Відкритий чат із активним співрозмовником. Кроки:1. Дочекатися надходження нового повідомлення. Очікуваний результат: Повідомлення з’являється у чаті миттєво, завдяки WebSocket/Firebase realtime listeners.
5	Перегляд історії повідомлень Мета: Перевірити, чи коректно завантажується історія листування з Firestore. Передумови: У чаті є збережені повідомлення. Кроки: 1. Відкрити список чатів. 2. Обрати чат із історією. Очікуваний результат: Всі повідомлення відображаються у хронологічному порядку.

Продовження таблиці 3.4

1	2
6	Пошук користувача за email Мета: Перевірити роботу пошуку користувачів у Firestore. Передумови: Існує інший зареєстрований користувач. Кроки: 1. Відкрити вкладку «Друзі». 2. Ввести email у пошукове поле. Очікуваний результат: Знайдений користувач відображається у списку результатів пошуку.
7	Вихід із системи Мета: Перевірити коректність деавтентифікації користувача. Передумови: Користувач авторизований. Кроки: 1. Відкрити меню профілю. 2. Обрати пункт «Вийти». Очікуваний результат: Користувача перенаправлено на екран авторизації, Firebase завершує сесію.

Проведене тестування підтвердило працездатність розробленої системи та її відповідність визначеним функціональним і нефункціональним вимогам. Усі основні модулі – автентифікація, реєстрація, передача та отримання повідомлень, завантаження історії чатів, пошук користувачів, а також вихід із системи – продемонстрували стабільну роботу у взаємодії з Firebase Authentication та Firestore.

Особливу увагу було приділено коректності обробки подій у режимі реального часу, що є критично важливим сповіщень-додатків. Результати тестування засвідчили, що система забезпечує миттєву появу нових повідомлень завдяки механізмам «listening» у Firestore, а також правильно обробляє оновлення стану чатів та синхронізацію даних між різними клієнтами.

Під час тестування безпеки було перевірено коректність застосування правил доступу Firebase Security Rules, що гарантують захист персональних даних користувачів та запобігають несанкціонованому доступу. Перевірка різних сценаріїв (доступ до чужих чатів, некоректні запити, маніпуляція ідентифікаторами) підтвердила, що система адекватно блокує потенційно небезпечні операції.

Регресійне тестування, проведене після внесення змін у логіку та інтерфейс, не виявило нових критичних помилок. Це свідчить про те, що внутрішня архітектура застосунку є стійкою до розширення та модифікацій.

У результаті всіх етапів тестування встановлено, що система є функціонально завершеною, стабільною та готовою до практичного використання. Розроблений програмний продукт забезпечує надійну комунікацію між користувачами, відповідає технічним вимогам і демонструє високий рівень якості, необхідний для подальшого впровадження і експлуатації.’

3.5 Висновки до розділу

У даному розділі було детально розглянуто процес розробки інформаційної системи для комунікації користувачів з використанням Flutter та REST API. Визначено функціональні та нефункціональні вимоги, які лягли в основу формування архітектури системи, вибору технологічного стеку та встановлення критеріїв її якості, продуктивності, безпеки та масштабованості.

Виконано проектування бази даних, що забезпечує цілісне, раціональне та безпечне зберігання інформації про користувачів, розмови та повідомлення. Запропонована структура бази даних підтримує реалізацію основних функцій чат системи, зокрема реєстрацію та автентифікацію користувачів, створення приватних і групових розмов, обмін повідомленнями в режимі реального часу, а також можливість подальшого розширення функціоналу.

Реалізовано клієнтську частину системи на базі фреймворку Flutter, орієнтовану на зручність, інтуїтивну зрозумілість та комфортну взаємодію користувачів із мобільним застосунком. Серверну частину побудовано з використанням хмарної платформи Firebase, яка забезпечує надійну обробку запитів, синхронізацію даних у реальному часі, безпечну автентифікацію та підтримку сповіщень. Взаємодія між клієнтською та серверною частинами реалізована відповідно до принципів REST архітектури.

Проведене тестування інформаційної системи підтвердило коректність реалізації функціональних можливостей, стабільність роботи клієнтської та серверної частин, відповідність системи визначеним вимогам, а також ефективність механізмів обміну даними та захисту персональної інформації користувачів.

Отримані результати свідчать про готовність розробленої інформаційної системи для комунікації користувачів до практичного впровадження та подальшої експлуатації. Створена система є надійною, масштабованою та може бути основою для подальшого розвитку, зокрема шляхом розширення функціоналу, інтеграції додаткових сервісів та вдосконалення користувацького інтерфейсу.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічні розробки мають перспективу впровадження лише за умови їх відповідності сучасним тенденціям цифровізації, технологічного прогресу та економічної доцільності. Одним із ключових аспектів оцінки результатів інженерних і програмних проєктів є аналіз їх комерційної цінності та можливостей масштабування, що визначає подальші шляхи впровадження та потенціал виходу на ринок.

Магістерська кваліфікаційна робота на тему «Інформаційна система для комунікації користувачів з використанням Flutter і REST API» належить до прикладних програмних розробок, орієнтованих на реальне застосування в умовах сучасного цифрового середовища. Створена система може бути інтегрована в освітні, корпоративні або комерційні платформи, забезпечуючи швидку, зручну та безпечну взаємодію між користувачами. Використання Flutter дозволяє реалізувати кросплатформність і зменшити витрати на розробку, а REST API забезпечує масштабованість і можливість використання системи в різних інформаційних інфраструктурах.

Результати роботи мають потенціал комерціалізації, оскільки потреба в ефективних засобах комунікації актуальна для широкого спектра організацій – закладів освіти, бізнес-структур, сервісів підтримки клієнтів, внутрішніх корпоративних платформ. Рішення щодо виходу продукту на ринок може бути ухвалене після оцінювання його конкурентоспроможності, надійності та технічних переваг порівняно з існуючими аналогами.

Для цього необхідно здійснити аналіз очікуваного економічного ефекту від впровадження системи, оцінити можливість масштабування продукту та визначити його привабливість для інвесторів. Важливо також провести аудит ринку потенційних користувачів та партнерів, обґрунтувавши доцільність розробки з погляду зниження витрат, підвищення ефективності комунікації та можливостей адаптації системи під конкретні потреби замовника.

4.1 Оцінювання комерційного та технологічного потенціалу розробки програмного забезпечення

Метою комерційного та технологічного аудиту є визначення науково-технічного рівня та оцінка комерційної перспективності створеної інформаційної системи для комунікації користувачів. Аудит спрямований на визначення конкурентних переваг, інноваційності технологічних рішень та перспективності програмного продукту для подальшого впровадження в ринковому середовищі.

Для здійснення об’єктивної оцінки пропонується використовувати п’ятибальну шкалу за 12 критеріями [34], узгодженими з вимогами до сучасних інформаційних технологій. Відповідні критерії наведені в таблиці 4.1 і дозволяють комплексно проаналізувати рівень інноваційності, технічну реалізацію та можливості комерціалізації розробки.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в

Продовження таблиці 4.1

Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінки науково-технічного рівня та комерційного потенціалу розробки оформили у вигляді таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	3
2. Ринкові переваги (наявність аналогів)	3	4	3
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	4	3
8. Практична здійсненність (наявність фахівців)	3	4	3
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	4	3	4

Продовження таблиці 4.2

12. Практична здійсненність (розробка документів)	3	4	4
Сума балів	42	44	43
Середньоарифметична сума балів $СБ_с$	43		

На основі розрахунків, представлених у таблиці 4.2, зробимо висновок щодо науково-технічного рівня та комерційного потенціалу розробки, використовуючи рекомендації, наведені в таблиці 4.3.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки на тему «Інформаційна система для комунікації користувачів з використанням Flutter і REST API» становить 43 бали. Відповідно до таблиці 4.3, це свідчить про високу комерційну значущість запропонованої інформаційної системи та доцільність її подальшої комерціалізації.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Під час планування, обліку та калькулювання собівартості програмної розробки та впровадження інформаційної системи для комунікації користувачів з використанням Flutter і REST API витрати групуються за такими статтями:

– витрати на оплату праці;

– відрахування на соціальні заходи;

– матеріали та комплектуючі (у разі використання тестового обладнання);

– електроенергія та інші комунальні витрати, необхідні для забезпечення роботи технічних засобів;

– витрати на службові відрядження (за необхідності проведення зовнішніх консультацій або впровадження у партнерських організаціях);

– спеціальне обладнання для тестування програмного забезпечення;

– програмне забезпечення, необхідне для розробки, тестування та налагодження системи;

– витрати на послуги сторонніх організацій (наприклад, аудит безпеки, UX-тестування);

– інші витрати;

– накладні (загальновиробничі) витрати.

До статті «Витрати на оплату праці» відносяться витрати на основну та додаткову заробітну плату розробників, системних аналітиків, тестувальників, дизайнерів, DevOps-інженерів та інших фахівців, які безпосередньо залучені до реалізації інформаційної системи.

Такі витрати визначаються відповідно до посадових окладів працівників, розцінок та тарифних ставок відповідно до чинної системи оплати праці в організації. До цієї статті також входять премії, компенсації та інші встановлені виплати.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{пі} * t_і}{Т_р} \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

$М_{пі}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

$Т_р$ – середня кількість робочих днів в місяці, Тр=21…23 дні.

$$З_о = \frac{32000 * 30}{22} = 43636{,}36 грн.$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Менеджер проекту	32000	1454,5	30	43636,36
Розробник програмного забезпечення	30000	1363,6	50	68181,81
Всього				111818,17

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{п} С_i * t_i \quad (4.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду $С_i$ можна визначити за формулою:

$$С_i = \frac{М_м * К_i * К_с}{Т_р * t_{зм}} \quad (4.3)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

$$С_і = \frac{8000 * 1{,}1 * 1{,}65}{22 * 8} = 82{,}50 \text{ грн.}$$

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Монтаж робочого обладнання (ПК, роутер, серверне підключення)	6	2	1,1	82,50	495,00
Підготовка робочого місця	4	2	1,1	82,50	330,00
Встановлення програмного забезпечення (Flutter SDK, емулятори, API-інструменти)	8	5	1,7	133,57	1020,00

Продовження таблиці 4.5

Перевірка роботи системи (REST API, авторизація, чат-комунікація)	8	2	1,1	82,50	660,00
Всього					2505,00

Додаткова заробітна плата дослідників та робітників

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$$З_{дод} = (111818{,}17 + 2505{,}00) * \frac{10}{100} = 11432{,}32 \text{ грн.}$$

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) * \frac{Н_{зп}}{100\%} \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (111818{,}17 + 2505{,}00 + 11432{,}32) * \frac{22}{100} = 27666{,}21 \text{ грн.}$$

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$М = \sum_{j=1}^{п} Н_j * Ц_j * К_j - \sum_{j=1}^{n} В_j \times Ц_{в\ j} \quad (4.6)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$$М = 2 * 180 * 1{,}1 - 0{,}0 * 0{,}0 = 396{,}00 \text{ грн.}$$

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Картриджі / тонер	180	2	0	0	396,00
Папір для друку документації	55	2	0	0	121,00
Канцелярія	120	3	0	0	396,00
USB флешка	160	1	0	0	176,00
Всього					1089,00

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{к} Ц_{прг} * С_{прг.i} * К_i \quad (4.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{прг.і}$. – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($К_і$ = 1,10…1,12);

k – кількість найменувань програмних засобів.

$$В_{прг} = 845 * 1 * 1,1 = 2788,5 \text{ грн.}$$

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
OpenAI	1	845	929,5
Figma (комерційна версія)	1	1100	1210
Всього			2139,5

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{87000,0 * 2}{5 * 12} = 2900,0 \text{ грн.}$$

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Macbook M3	87000,0	5	2	2900,0
Офісне приміщення	20000,0	4	2	833,3
Оргтехніка	9000,0	3	2	500
Всього				4233,33

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{уі} * t_і * Ц_е * К_{впі}}{\eta_і} \quad (4.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,50 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$

$$В_е = \frac{0{,}60 * 280 * 7{,}50 * 0{,}95}{0{,}97} = 1234{,}02 \text{ грн.}$$

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} \quad (4.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = (111818{,}17 + 2505{,}00) * \frac{22}{100} = 25151{,}10 \text{ грн.}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Macbook M3	0,60	280	1234,02
Офісне приміщення	0,32	250	587,63
Оргтехніка	0,20	12	17,63
Всього			1839,28

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{сп}}{100\%} \quad (4.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = (111818{,}17 + 2505{,}00) * \frac{40}{100} = 45729{,}27 \text{ грн.}$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = (111818{,}17 + 2505{,}00) * \frac{70}{100} = 80026{,}22 \text{ грн.}$$

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (111818{,}17 + 2505{,}00) * \frac{110}{100} = 130096{,}43 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.14)$$

$$В_{заг} = 111818{,}17 + 2505{,}00 + 11432{,}32 + 28621{,}20 + 1166{,}55 + 7700 + 2134{,}63 + 1906{,}27 + 25151{,}10 + 45729{,}27 + 80026{,}22 + 130096{,}43 = 449671{,}74 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.15)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9

$$ЗВ = \frac{449671,74}{0,7} = 642388,20 \text{ грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах основним позитивним результатом для потенційного інвестора від впровадження науково-технічної розробки є зростання чистого прибутку.

Дослідження за темою «Інформаційна система для комунікації користувачів з використанням Flutter і REST API» передбачає можливість комерціалізації програмного продукту протягом трьох років на ринку. Майбутній економічний ефект у цьому випадку формуватиметься на основі таких даних::

Збільшення кількості споживачів продукту в аналізовані періоди часу завдяки покращенню його характеристик (ΔN):

1-й рік – 80 користувачів;

2-й рік – 120 користувачів;

3-й рік – 180 користувачів.

Кількість споживачів, які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, становить 150 користувачів (N).

Вартість програмного продукту до впровадження результатів розробки – 150 000 грн ($Ц_б$).

Зміна вартості програмного продукту від впровадження результатів науково-технічної розробки – 6 000 грн ($\pm\Delta Ц_б$).

Можливе збільшення чистого прибутку потенційного інвестора для кожного з трьох років, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, розраховується за формулою.

$$\Delta П_i = (\pm\Delta Ц_б * N + Ц_б * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}) \quad (4.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 30\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

1-й рік: $\Delta П_1 = (6000 \times 150 + 150000 \times 80) \times 0,83 \times 0,3 \times \left(1 - \frac{0,18}{100}\right) = 3206325,22$ (грн.)

2-й рік: $\Delta П_2 = \left(6000 \times 120 + 150000 \times (80 + 120)\right) \times 0,83 \times 0,3 \times \left(1 - \frac{0,18}{100}\right) = 7680236,38$ (грн.)

3-й рік: $\Delta П_3 = \left(6000 \times 150 + 150000 \times (80 + 120 + 180)\right) \times 0,83 \times 0,3 \times \left(1 - \frac{0,18}{100}\right) = 14391041,78$ (грн.)

Отже, за результатами обчислень, впровадження розробки призведе до значної комерційної вигоди, що виявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними критеріями, що визначають обґрунтованість фінансування наукової розробки певним інвестором, є абсолютна та відносна ефективність інвестицій, а також термін їх окупності.

На першому етапі розраховується теперішня вартість інвестицій (PV), вкладених у наукову розробку.

Розмір початкових інвестицій, які потенційний інвестор має внести для впровадження і комерціалізації науково-технічної розробки:

$$\text{PV} = k_{інв} * ЗВ \tag{4.17}$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 642388,20 грн.

$$\text{PV} = 3 * 642388{,}20 = 1927164{,}6\text{грн.}$$

Тоді абсолютний економічний ефект $\text{Е}_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{\text{абс}} = (\text{ПП} - \text{PV}) \tag{4.18}$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{\Delta П_1}{(1+\tau)^t} \quad (4.19)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{3206325,22}{(1+0,1)^1} + \frac{7680236,38}{(1+0,1)^2} + \frac{14391041,78}{(1+0,1)^3} = 20075347,12 \text{ грн.}$$

$$Е_{абс} = 20075347,12 - 1927164,6 = 18148182.52 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, встановлено, що проведення наукових досліджень для розробки програмного продукту та його подальше впровадження принесуть прибуток. Це підтверджує доцільність проведення досліджень.

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1 \quad (4.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{18148182.52}{1927164,6}} - 1 = 1,24$$

Порівняємо Ев з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає мінімальну дохідність, нижче якої інвестиції не будуть здійснюватися.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за формулою:

$$\tau_{min} = d + f \quad (4.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; f = 0,4.

d = 0,2.

$$\tau_{min} = 0,2 + 0,4 = 0,6$$

Оскільки $E_в$ = 130% > τ_{min} = 60%, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \quad (4.22)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{1,24} = 0,81$$

Якщо $T_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.5 Висновки до розділу

Згідно з проведеними розрахунками та аналізом комерційного потенціалу, рівень перспективності розробки за темою «Інформаційна система для комунікації користувачів з використанням Flutter і REST API» становить 43 бали, що вказує на її високу актуальність і значущість для ринку цифрових сервісів.

Розрахований термін окупності проєкту становить 0,81 року, що є значно меншим за типовий трирічний інвестиційний цикл. Це підтверджує, що створена система є економічно привабливою для потенційних інвесторів і має високі шанси на успішну комерціалізацію.

Результати оцінки свідчать, що подальший розвиток та вдосконалення інформаційної системи, зокрема оптимізація функціоналу для взаємодії користувачів, інтеграція нових REST API-сервісів та розширення мобільних можливостей на Flutter, є доцільними. Отже, розробка має значний потенціал для

впровадження у практичну діяльність підприємств і сервісів, що потребують ефективних інструментів комунікації.

Отримані результати підтверджують як технічну, так і економічну доцільність реалізації проєкту, а також перспективність його подальшого впровадження й комерційного розвитку.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було виконано комплексне дослідження, спрямоване на проєктування та розроблення інформаційної системи для комунікації користувачів із використанням сучасних технологій Flutter та REST API. На основі детального аналізу предметної області, наявних аналогів та сучасних тенденцій у сфері мобільних комунікаційних сервісів було сформовано вимоги до функціональності, продуктивності, масштабованості, зручності використання та інформаційної безпеки.

Вагомим результатом роботи стало обґрунтування вибору Flutter як основної технології клієнтської частини. У дослідженні доведено, що Flutter забезпечує високу швидкість розроблення, можливість створення кросплатформних застосунків із єдиною кодовою базою, багатий набір UI-компонентів, а також надійні засоби побудови анімацій та адаптивного інтерфейсу. Це дозволило створити інтуїтивний і візуально цілісний користувацький досвід як для Android, так і для iOS пристроїв. Окрему увагу приділено архітектурі застосунку, що базується на принципах розділення відповідальності, повторного використання компонентів та мінімізації залежностей. Проведене порівняння показало, що швидкість розробки інтерфейсу на Flutter є приблизно на 32–45 % вищою, ніж при створенні окремих Android/iOS застосунків. Тестові збірки продемонстрували стабільний показник FPS на рівні 55–60 кадрів/с, а час рендерингу ключових UI-компонентів не перевищував 9–12 мс, що відповідає вимогам продуктивних мобільних застосунків.

Застосування REST API під час побудови серверної частини дало змогу сформувати гнучку та масштабовану клієнт-серверну архітектуру, що підтримує стабільний обмін даними, швидке розширення функціоналу та просту інтеграцію з іншими сервісами. Було реалізовано функції реєстрації та автентифікації користувачів, обмін текстовими повідомленнями, пошук користувачів,

керування профілем, перегляд списків контактів та налаштування темної/світлої теми інтерфейсу. Усі ці можливості формують основу для повноцінної комунікаційної системи.

Особливе місце в роботі займають питання інформаційної безпеки. Було реалізовано токен-автентифікацію, захист кінцевих точок API, шифрування даних, обробку помилок, а також дотримано вимог щодо захисту персональних даних. Розглянуто потенційні загрози, включаючи перехоплення даних, спроби SQL- або API-ін’єкцій, несанкціоноване використання токенів та інші вектори атак. Запроваджені механізми забезпечили належний рівень захисту як на рівні передачі, так і на рівні зберігання інформації.

Проведене функціональне, нефункціональне, регресійне та інтеграційне тестування підтвердило відповідність розробленого програмного забезпечення сформульованим вимогам. Система продемонструвала стабільність роботи, коректний обмін повідомленнями в режимі реального часу, швидку синхронізацію даних між клієнтами та відсутність критичних помилок. Тестування продуктивності показало, що застосунок коректно обробляє типові сценарії навантаження та має достатній запас для подальшого масштабування.

Важливою складовою дослідження стала економічна оцінка доцільності розроблення. Розрахований інтегральний показник комерційного потенціалу становить 43 бали, що свідчить про високі перспективи впровадження системи в умовах сучасного ринку цифрових сервісів. Визначений термін окупності проекту становить 0,81 року, що демонструє високу інвестиційну привабливість та ефективність використаних технологій, а рентабельність інвестицій — близько 123 % у перший рік використання**.**

Окрему увагу приділено питанням подальшого розвитку та масштабованості системи. У роботі запропоновано напрями удосконалення, серед яких: розширення функціоналу чатів (групові чати, передавання зображень і відео), впровадження push-сповіщень, покращення алгоритмів пошуку та рекомендацій, інтеграція зі сторонніми сервісами (хмарне зберігання, сервіси аналітики), створення веб-версії застосунку та адміністративної панелі для

модерації контенту й керування системою. Це забезпечує можливість еволюції розробленої системи у повноцінну комунікаційну платформу.

Усі поставлені задачі магістерського дослідження виконано в повному обсязі. Створене програмне забезпечення відповідає сучасним вимогам до ергономічності, продуктивності, безпеки, масштабованості та користувацького досвіду. Результати роботи мають високу практичну значущість і можуть бути використані як у рамках підприємницьких стартапів, так і в корпоративних рішеннях, орієнтованих на внутрішню комунікацію.

Отже, підсумовуючи виконане дослідження, можна зробити висновок, що створена інформаційна система повністю відповідає поставленим цілям, має інноваційні риси, підтверджену ефективність, а також економічну доцільність її подальшого розвитку, впровадження та масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Забезпечення інформаційної безпеки мобільних додатків на Flutter / Осипенко І.В., Богач І.В. // Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» 2025», Вінниця, 2025: веб сайт. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25461. (дата звернення 10.10.2025).

2. Кузьмін, В. І. Розробка мобільних додатків: теорія та практика. Київ : Видавництво Національного технічного університету України "КПІ", 2020.

3. Магазин додатків Google Play Market. Блок з фідбеком користувачів: веб сайт. URL: https://play.google.com/store/apps/details?id=chat.glib.groupchat&hl=uk. (дата звернення 10.10.2025).

4. Офіційний сайт Beeper: веб сайт. URL: https://www.beeper.com. (дата звернення 10.10.2025).

5. Кравченко П. Г. Основи розробки мобільних додатків для платформ Android та iOS. Харків : Видавництво ХНЕУ, 2019. 310 с.

6. Flutter vs React Native: Що буде краще у 2023 році: веб сайт. URL: https://stfalcon.com/uk/blog/post/flutter-vs-react-native-which-will-do-better (дата звернення 10.10.2025).

7. Використання фреймворку Flutter для розробки мультиплатформних додатків / Осипенко І.В., Богач І.В. // Матеріали LIII Всеукраїнської науково-практичної конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024) 2024», Вінниця, 2024. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20387. (дата звернення 10.10.2025).

8. Офіційна документація Flutter: веб сайт. URL: https://flutter.dev/docs (дата звернення 10.10.2025).

9. Офіційна документація мови програмування Dart: веб сайт. URL: https://dart.dev/guides (дата звернення 10.10.2025).

10. Офіційна документація пакету json_serializable: 6.8.0: веб-сайт. URL: https://pub.dev/packages/json_serializable (дата звернення 10.10.2025).

11. Офіційна документація dio: ^5.4.3+1. URL: https://pub.dev/packages/dio (дата звернення 10.10.2025).

12. Документація до пакету flutter_bloc: веб-сайт. URL: https://bloclibrary.dev/ (дата звернення 12.10.2025).

13. Dart VM and Compilation Model. Google Developers: веб сайт. URL: https://dart.dev/platforms (дата звернення 12.10.2025).

14. Inside Flutter. Google Engineering Documentation: веб сайт. URL: https://docs.flutter.dev/resources/inside-flutter (дата звернення 12.10.2025).

15. Dart Type System. Google Developers: веб сайт. URL: https://dart.dev/language/type-system (дата звернення 12.10.2025).

16. Документація до пакету flutter_bloc. веб сайт. URL: https://bloclibrary.dev/ (дата звернення 12.10.2025).

17. MobX for Flutter Documentation. Pub.dev. веб сайт. URL: https://pub.dev/packages/mobx (дата звернення 12.10.2025).

18. Shared Preferences. Pub.dev. веб сайт. URL: https://pub.dev/packages/shared_preferences (дата звернення 12.10.2025).

19. Hive Database. Pub.dev. веб сайт. URL: https://pub.dev/packages/hive (дата звернення 12.10.2025).

20. Firebase for Flutter. Google Developers: веб сайт. URL: https://firebase.flutter.dev (дата звернення 12.10.2025).

21. Implementing organizational project management: a practice guide. by Project Management Institute. 2014. ISBN: 9781628250824

22. Esther Cohen «The Definitive Guide to Project Management Methodologies» April 25, 2022 – https://www.workamajig.com/blog/projectmanagement-methodologies

23. Інноваційні методи управління проектами./ Смолич Д. В. // Економічний форум. 2019. № 4. С. 50–53.

24. Аналіз необхідних комунікативних навичок при розробці програмного забезпечення / Сидорова, М. Г., Байбуз, О. Г., Лапець, О. В. // Актуальні проблеми автоматизації та інформаційних технологій. № 25.2021. С. 152-157.

25. Date C. J. An Introduction to Database Systems. 8th ed. Addison-Wesley, 2003. 1024 p.

26. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. Pearson, 2016. 1200 p.

27. Firebase Cloud Firestore Documentation. Google Developers. веб сайт. URL: https://firebase.google.com/docs/firestore (дата звернення 12.10.2025).

28. Firebase REST API Reference. Google Developers. веб-сайт. URL: https://firebase.google.com/docs/reference/rest (дата звернення 18.10.2025).

29. Firebase Authentication Documentation. Google Developers. веб-сайт. URL: https://firebase.google.com/docs/auth (дата звернення 18.10.2025).

30. Realtime Database vs Firestore. Google Developers. веб-сайт. URL: https://firebase.google.com/docs/database/rtdb-vs-firestore (дата звернення 18.10.2025).

31. Sommerville I. Software Engineering. 10th Edition. Pearson Education, 2016. 804 p

32. Firebase Security Rules Documentation: веб сайт. URL: https://firebase.google.com/docs/rules (дата звернення: 20.10.2025).

33. Flutter Testing Documentation: веб сайт. URL: https://docs.flutter.dev/testing (дата звернення: 20.10.2025).

34. Козловський В. О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця : ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«Інформаційна система для комунікації користувачів з використанням Flutter і REST API»
08-31.МКР.005.02.000 ТЗ

Керівник роботи:
к.т.н., проф. каф. АІІТ
Ілона БОГАЧ
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1ІСТ-24м
Ірина ОСИПЕНКО
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Інформаційна система для комунікації користувачів з використанням Flutter і REST API.

Розроблення мобільних клієнт-серверних систем, орієнтованих на організацію текстової комунікації між користувачами. Застосунок належить до сфери інформаційних технологій, а саме до мобільної розробки, мережевих сервісів та архітектури клієнт-серверних систем.

Система використовується для забезпечення обміну повідомленнями в режимі реального часу, керування списком контактів, зберігання історії спілкування та підтримки налаштувань користувача. Взаємодія між мобільним клієнтом і серверною частиною реалізується за допомогою REST API, що забезпечує стандартизований протокол обміну даними, масштабованість і сумісність із сучасними веб сервісами.

Використання фреймворку Flutter дає змогу створити єдиний кросплатформний застосунок для iOS та Android із високою продуктивністю, а REST API – забезпечити надійну інтеграцію, безпеку та ефективність роботи із серверними ресурсами. Система може застосовуватися у соціальних платформах, корпоративних чатах, сервісах підтримки, внутрішніх комунікаційних інструментах організацій та інших сервісах, що вимагають структурованого та швидкого обміну текстовими повідомленнями.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є розроблення мобільної інформаційної системи для забезпечення ефективної та безпечної комунікації користувачів, яка включає:

- Реалізацію механізму автентифікації та авторизації на основі Firebase Authentication з підтримкою реєстрації, входу за електронною поштою, відновлення пароля та безпечного керування сесіями користувача.

- Організацію обміну повідомленнями в режимі реального часу з використанням хмарної бази даних Cloud Firestore, що забезпечує миттєву синхронізацію чатів, зберігання історії переписки та оброблення подій (нові повідомлення, статус "online", відображення прочитання).
- Інтеграцію REST API для виконання допоміжних операцій, зокрема автоматичного перекладу повідомлень за допомогою сторонніх API-сервісів, що забезпечує мультимовну комунікацію між користувачами.
- Побудову сучасного інтерфейсу мобільного застосунку на основі Flutter, із застосуванням анімацій, кастомних віджетів та адаптивних UI-рішень для забезпечення зручності взаємодії з системою на пристроях із різними розмірами екранів.

Система призначена для організації швидкої та зручної взаємодії між користувачами шляхом обміну текстовими повідомленнями в реальному часі, забезпечення доступу до особистих чатів, керування контактами та підтримки багатомовного спілкування.

4. Джерела розробки

1. Офіційна документація Flutter: веб сайт. URL: https://flutter.dev/docs (дата звернення 10.10.2025).
2. Офіційна документація мови програмування Dart: веб сайт. URL: https://dart.dev/guides (дата звернення 10.10.2025).
3. Офіційна документація dio: ^5.4.3+1. URL: https://pub.dev/packages/dio (дата звернення 10.10.2025).
4. Dart VM and Compilation Model. Google Developers: веб сайт. URL: https://dart.dev/platforms (дата звернення 12.10.2025).
5. Inside Flutter. Google Engineering Documentation: веб сайт. URL: https://docs.flutter.dev/resources/inside-flutter (дата звернення 12.10.2025).

5. Показники призначення

5.1. Основні технічні характеристики системи

Функціональні можливості:

- Реєстрація та авторизація користувачів, включаючи відновлення пароля через Firebase Authentication.
- Відправлення та отримання текстових повідомлень у режимі реального часу завдяки Firestore Streams.
- Керування профілем користувача: редагування імені, статусу, списку друзів.
- Пошук користувачів за адресою електронної пошти або унікальним ідентифікатором.
- Індикація онлайн-статусу та статусів повідомлень (“доставлено”, “переглянуто”).
- Оптимізована робота з великими списками чатів і повідомлень (lazy loading, caching).

5.2. Мінімальні системні вимоги

5.2.1 Апаратне забезпечення:

- Процесор: тактова частота не менше 2 GHz, рекомендовано 4-ядерний CPU;
- Оперативна пам'ять: 8 GB;
- Наявність GPU: бажано для прискорення рендерингу Flutter UI;
- Вільне місце на диску: не менше 8 GB для SDK, бібліотек та кешу.
- Мережа: стабільне Інтернет-з’єднання (мінімум 10 Mbps) для роботи з Firebase).

5.2.2 Програмне забезпечення:

- Операційна система:macOS 10.15+
- Flutter: версія 3.22 або вище;
- Dart SDK: версія 3.0 або вище (у складі Flutter);
- Android Studio / Xcode (для збірки додатку);
- Firebase CLI для налаштування середовища.

5.3. Вхідні дані

5.3.1. Дані користувача

- електронна пошта та пароль при реєстрації/логіні;
- ім’я користувача;
- статус, додаткові контактні дані;
- список унікальних ідентифікаторів друзів.

5.3.2. Дані повідомлень

- текст повідомлення;
- вкладення (зображення, аудіо, документи);
- службові поля (час відправлення, відправник, статус доставки).

5.3.3. Дані пристроїв

- FCM-токен пристрою для надсилання push-сповіщень;
- інформація про платформу (Android/iOS).

5.3.4 Дані для навігації інтерфейсу

- стан авторизації користувача;
- вибраний чат;
- параметри теми (dark/light).

6. Економічні показники

До економічних показників входять:

- витрати на розробку – до 250 тис. грн.
- узагальнений коефіцієнт якості розробки – більше 2-х
- термін окупності – не більше 1 року

7. Стадії розробки:

1. Розділ 1 «Аналіз предметної області та аналогів системи» має бути виконаний до 05.10.2025 р.
2. Розділ 2 «Огляд програмних засобів для реалізації поставленої задачі» має бути виконаний до 25.10.2025 р.
3. Розділ 3 «Розробка та тестування програмного забезпечення» має бути виконаний до 20.11.2025 р.
4. Розділ 4 «Економічний розділ» має бути виконаний до 01.12.2025 р.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025.
2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.
3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025 р.

Додаток Б (обов’язковий)

Ілюстративна частина

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ КОМУНІКАЦІЇ КОРИСТУВАЧІВ З ВИКОРИСТАННЯМ FLUTTER І REST API

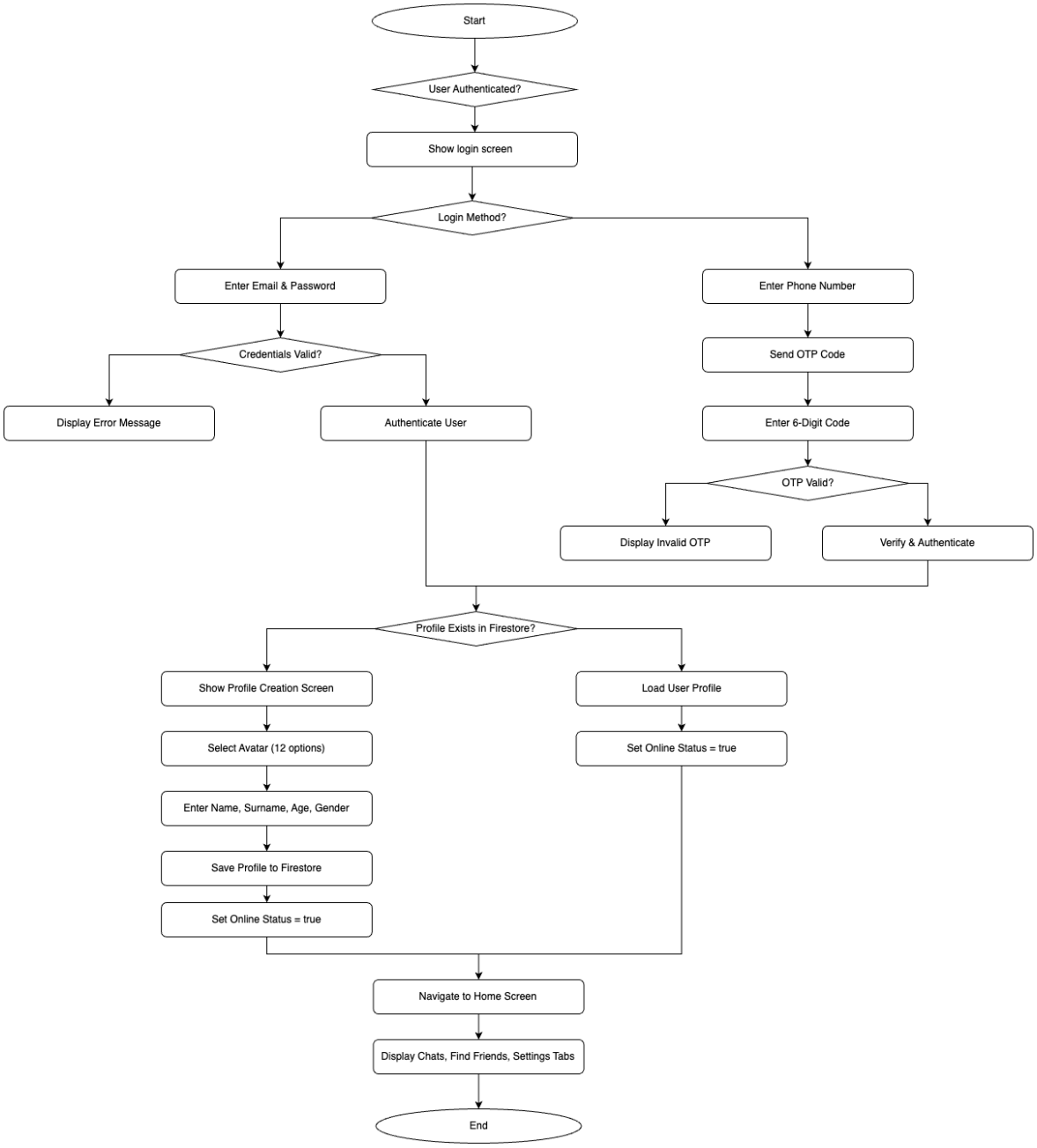

Рисунок Б.1 – Алгоритм аутентифікації користувача в додаток

Рисунок Б.2 – ER-діаграма структури бази даних Firestore

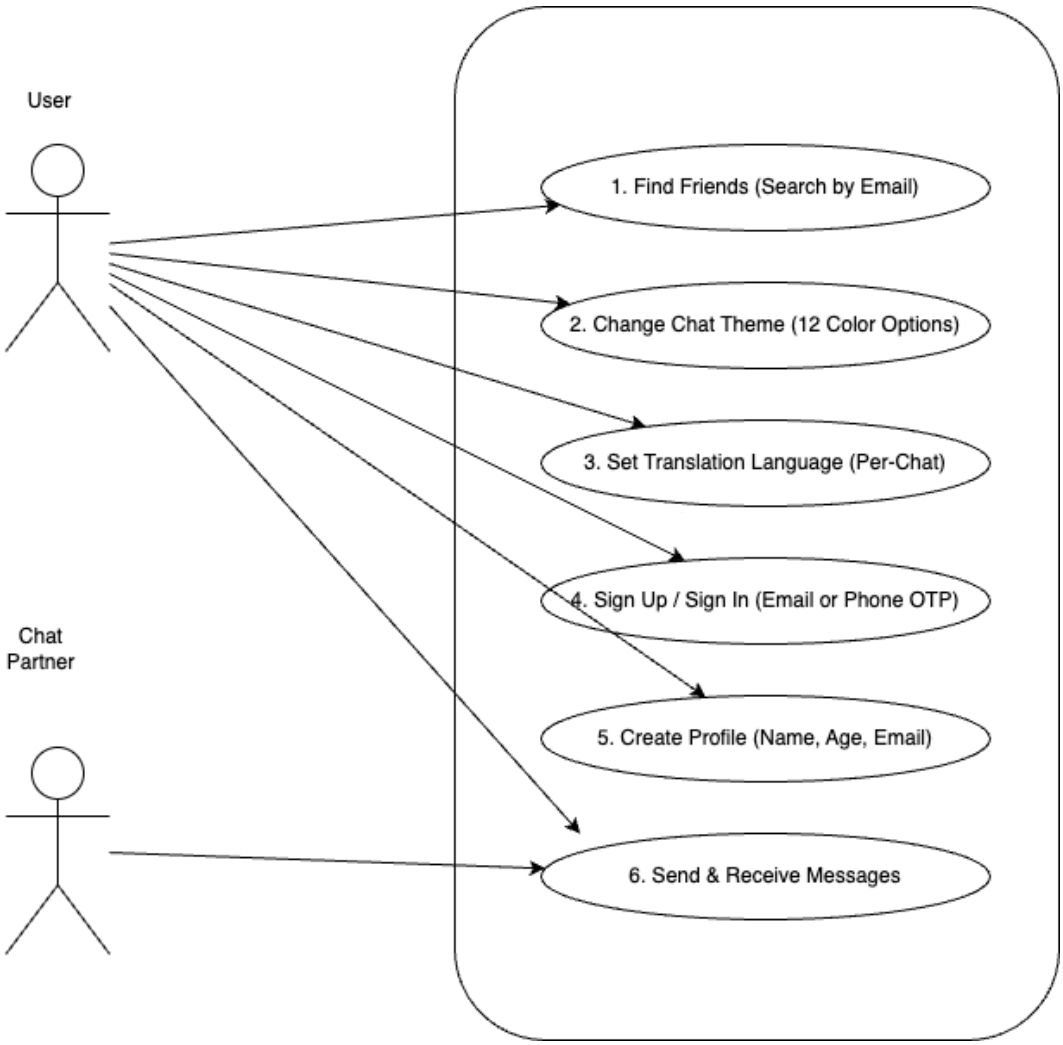

Рисунок Б.3 – UML-діаграма прецедентів для інформаційної системи для комунікації користувачів

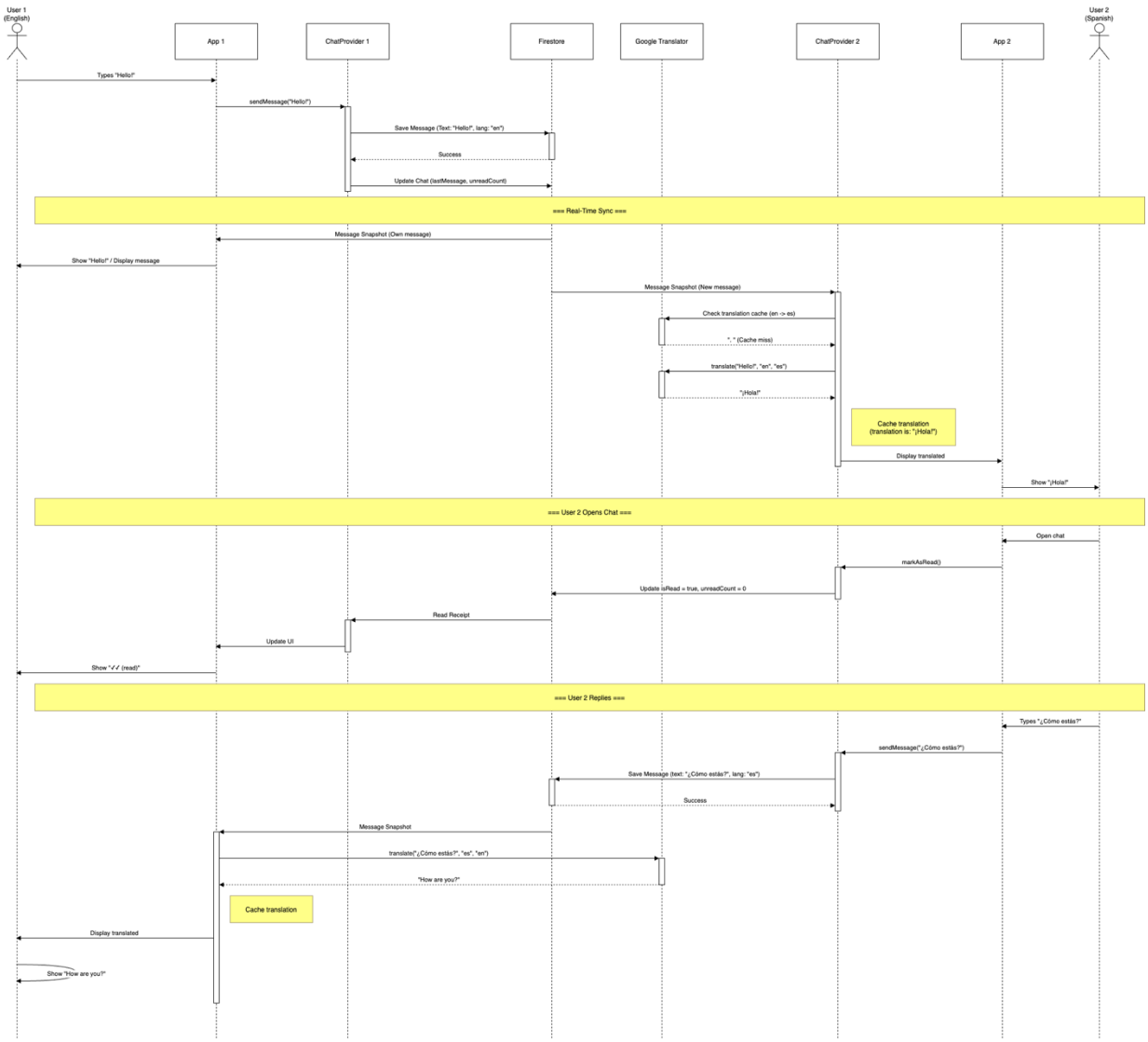

Рисунок Б.4 – Діаграма послідовності процесу обміну повідомленнями та автоматичного перекладу

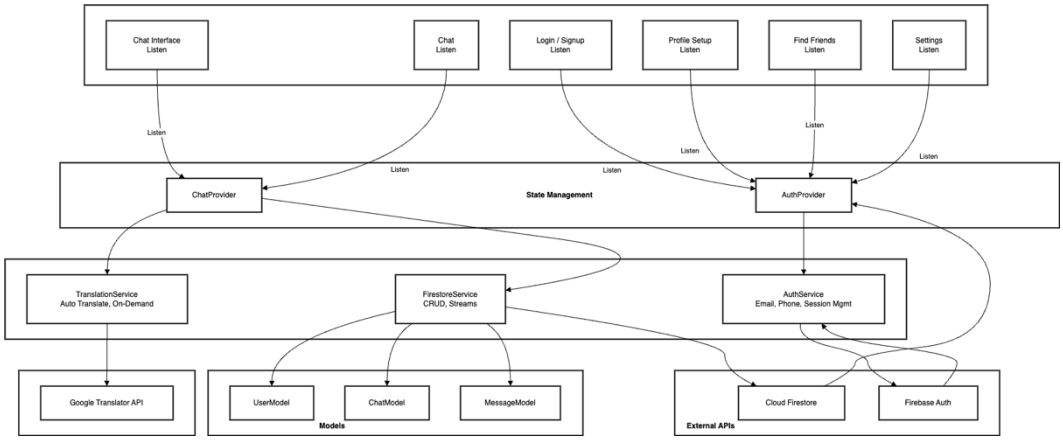

Рисунок Б.5 – Багаторівнева архітектура мобільного застосунку на Flutter

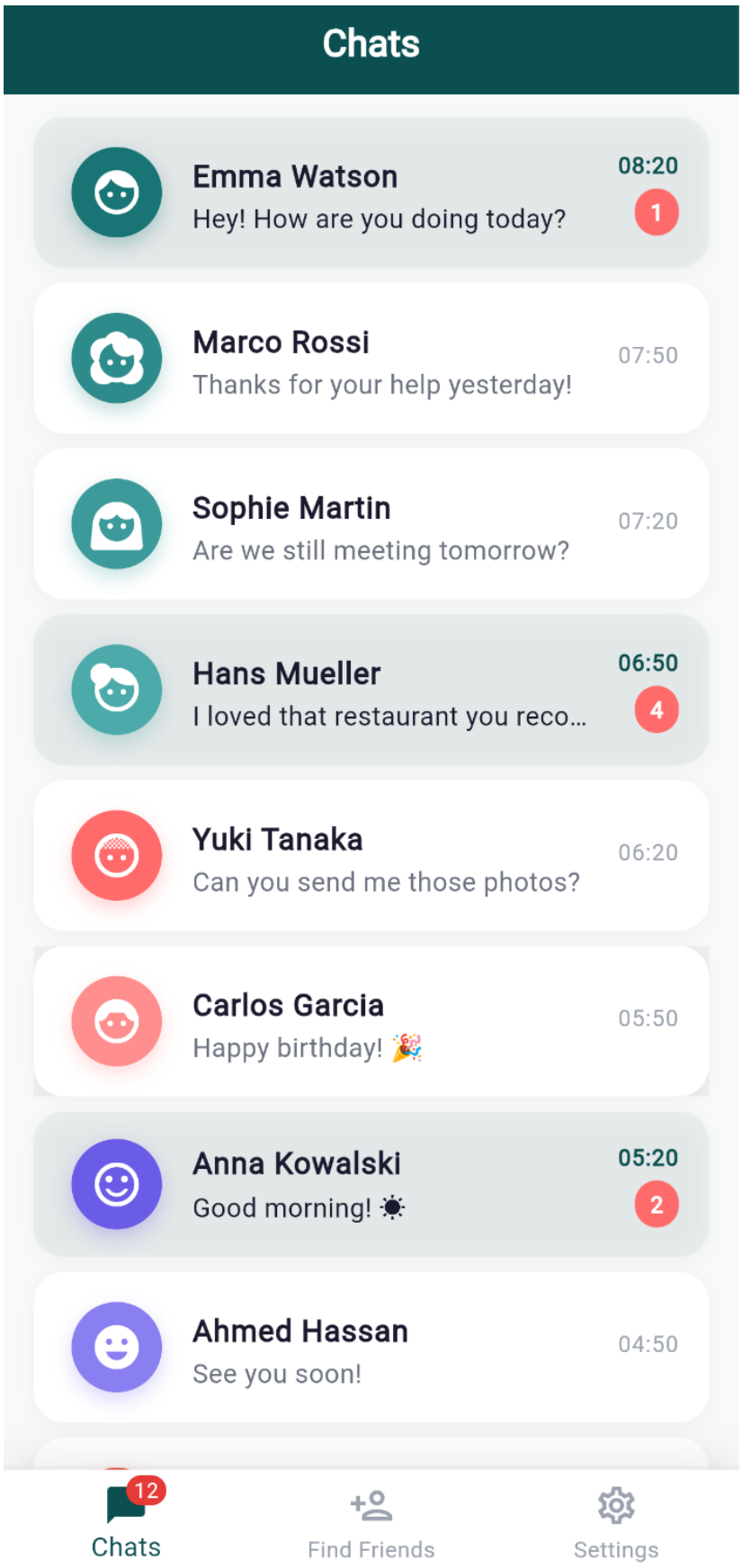

Рисунок Б.6 – Скріншот розробленої інформаційної системи

Додаток В (обов’язковий)

Лістинг програмного коду

1. Аутетифікація за допомогою пошти та паролю

```
Future<UserCredential?> signUpWithEmail({
  required String email,
  required String password,
  required Function(String error) onError,
}) async {
  try {
    return await _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
  } on FirebaseAuthException catch (e) {
    onError(_getAuthErrorMessage(e.code));
    return null;
  } catch (e) {
    onError(e.toString());
    return null;
  }
}
```

2. Створення профілю користувача

```
Future<bool> createProfile(UserModel user) async {
  _status = AuthStatus.loading;
  notifyListeners();

  try {
    if (_authService.currentUser?.email == null) {
      final emailExists = await _firestoreService.emailExists(user.email);
```

```
      if (emailExists) {
        _status = AuthStatus.needsProfile;
        _errorMessage = 'This email is already registered';
        notifyListeners();
        return false;
      }
    }

    await _firestoreService.createUser(user);
    _currentUser = user;
    _status = AuthStatus.authenticated;
    notifyListeners();
    return true;
  } catch (e) {
    _status = AuthStatus.needsProfile;
    _errorMessage = e.toString();
    notifyListeners();
    return false;
  }
}
```

3. Пошук друзів за email

```
Future<UserModel?> getUserByEmail(String email) async {
  final querySnapshot = await _usersCollection
      .where('email', isEqualTo: email.toLowerCase())
      .limit(1)
      .get();

  if (querySnapshot.docs.isNotEmpty) {
    return UserModel.fromFirestore(querySnapshot.docs.first);
  }
```

```
  return null;
}
```

4. Відправка повідомлення з автоматичним перекладом

```
Future<void> sendMessage({
  required String text,
  required UserModel sender,
  required String recipientId,
}) async {
  if (_currentChat == null || text.trim().isEmpty) return;

  final messageId = _uuid.v4();
  final message = MessageModel(
    id: messageId,
    chatId: _currentChat!.id,
    senderId: sender.id,
    senderName: sender.fullName,
    originalText: text.trim(),
    originalLanguage: sender.preferredLanguage,
    timestamp: DateTime.now(),
  );

  try {
    await _firestoreService.sendMessage(message);
    await _firestoreService.updateChatLastMessage(
      chatId: _currentChat!.id,
      message: text.trim(),
      senderId: sender.id,
      recipientId: recipientId,
    );
  } catch (e) {
```

```
    _errorMessage = e.toString();
    notifyListeners();
  }
}
```

5. Сервіс перекладу

```
import 'package:translator/translator.dart';

class TranslationService {
  final GoogleTranslator _translator = GoogleTranslator();

  static const Map<String, String> supportedLanguages = {
    'en': 'English',
    'es': 'Spanish',
    'fr': 'French',
    'de': 'German',
    'it': 'Italian',
    'pt': 'Portuguese',
    'uk': 'Ukrainian',
    'zh-cn': 'Chinese (Simplified)',
    'zh-tw': 'Chinese (Traditional)',
    'ja': 'Japanese',
    'ko': 'Korean',
    'ar': 'Arabic',
    'hi': 'Hindi',
    'tr': 'Turkish',
    'pl': 'Polish',
    'nl': 'Dutch',
    'sv': 'Swedish',
    'da': 'Danish',
    'no': 'Norwegian',
```

```
  'fi': 'Finnish',
  'cs': 'Czech',
  'el': 'Greek',
  'he': 'Hebrew',
  'th': 'Thai',
  'vi': 'Vietnamese',
  'id': 'Indonesian',
  'ms': 'Malay',
};

Future<String> translate({
  required String text,
  required String from,
  required String to,
}) async {
  if (text.isEmpty || from == to) {
    return text;
  }

  try {
    final translation = await _translator.translate(text, from: from, to: to);
    return translation.text;
  } catch (e) {
    // Return original text if translation fails
    return text;
  }
}

Future<String> detectLanguage(String text) async {
  if (text.isEmpty) {
```

```
      return 'en';
    }

    try {
      final translation = await _translator.translate(text, to: 'en');
      return translation.sourceLanguage.code;
    } catch (e) {
      return 'en';
    }
  }

  static String getLanguageName(String code) {
    return supportedLanguages[code] ?? code.toUpperCase();
  }

  static List<MapEntry<String, String>> getLanguageList() {
    return supportedLanguages.entries.toList()
      ..sort((a, b) => a.value.compareTo(b.value));
  }
}
```

6. Домашній екран додатку з відображенням всіх чатів

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../../providers/auth_provider.dart';
import '../../providers/chat_provider.dart';
import '../chat/chats_list_screen.dart';
import '../friends/find_friends_screen.dart';
import '../settings/settings_screen.dart';

class HomeScreen extends StatefulWidget {
```

```
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _currentIndex = 0;

  final List<Widget> _screens = [
    const ChatsListScreen(),
    const FindFriendsScreen(),
    const SettingsScreen(),
  ];

  @override
  void initState() {
    super.initState();
    _initializeChats();
  }

  void _initializeChats() {
    final authProvider = Provider.of<AuthProvider>(context, listen: false);
    final chatProvider = Provider.of<ChatProvider>(context, listen: false);

    if (authProvider.currentUser != null) {
      chatProvider.listenToChats(authProvider.currentUser!.id);
    }
  }
```

```
        @override
        Widget build(BuildContext context) {
          return Scaffold(
            body: IndexedStack(
              index: _currentIndex,
              children: _screens,
            ),
            bottomNavigationBar: Container(
              decoration: BoxDecoration(
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withValues(alpha: 0.08),
                    blurRadius: 20,
                    offset: const Offset(0, -5),
                  ),
                ],
              ),
              child: BottomNavigationBar(
                currentIndex: _currentIndex,
                onTap: (index) => setState(() => _currentIndex = index),
                items: [
                  BottomNavigationBarItem(
                    icon: Consumer<ChatProvider>(
                      builder: (context, chatProvider, _) {
                        final authProvider = Provider.of<AuthProvider>(context, listen:
false);
                        final unreadCount = authProvider.currentUser != null
                            ?
chatProvider.getTotalUnreadCount(authProvider.currentUser!.id)
                            : 0;
```

```
                return Badge(
                  isLabelVisible: unreadCount > 0,
                  label: Text(unreadCount.toString()),
                  child: const Icon(Icons.chat_bubble_outline_rounded),
                );
              },
            ),
            activeIcon: Consumer<ChatProvider>(
              builder: (context, chatProvider, _) {
                final authProvider = Provider.of<AuthProvider>(context, listen:
false);
                final unreadCount = authProvider.currentUser != null
                    ?
chatProvider.getTotalUnreadCount(authProvider.currentUser!.id)
                    : 0;

                return Badge(
                  isLabelVisible: unreadCount > 0,
                  label: Text(unreadCount.toString()),
                  child: const Icon(Icons.chat_bubble_rounded),
                );
              },
            ),
            label: 'Chats',
          ),
          const BottomNavigationBarItem(
            icon: Icon(Icons.person_add_outlined),
            activeIcon: Icon(Icons.person_add),
            label: 'Find Friends',
```

```
            ),
            const BottomNavigationBarItem(
              icon: Icon(Icons.settings_outlined),
              activeIcon: Icon(Icons.settings),
              label: 'Settings',
            ),
          ],
        ),
      ),
    );
  }
}
```

Додаток Г

Акт впровадження

Науково-виробниче підприємство
“СПІЛЬНА СПРАВА” Товариство з обмеженою відповідальністю
Україна, 21000, м. Вінниця, вул. Магістратська, 36/3, тел. +38(096)-558-24-64
код ЄДРПОУ 31041670, код платників податків 310416702282, свідоцтво № 0183329
IBAN : UA 41 322313 0000026004000003226 в філії АТ «Укрексімбанк» в м. Вінниці

Затверджую

Директор
НВП «СПІЛЬНА СПРАВА», ТОВ

Малачковська Роксолана Ігорівна
«09» грудня 2025 р.

АКТ

впровадження результатів магістерської кваліфікаційної роботи

Осипенко Ірини Віталіївни «Інформаційна система для комунікації користувачів з використанням Flutter і REST API»

Комісія у складі головного спеціаліста, керівника відділу обробки даних С.Житанського та керівника відділу розробки інформаційних систем О. Кириленка склали цей акт про те, що у Науково-виробничому підприємстві “Спільна Справа”, ТОВ впроваджуються результати, які отримані магістром Осипенко І.В. під час виконання кваліфікаційної роботи на тему «Інформаційна система для комунікації користувачів з використанням Flutter і REST API», які мають практичну цінність.

Розроблена інформаційна система забезпечує ефективну взаємодію між користувачами за рахунок використання кросплатформного фреймворку Flutter та архітектури REST API, що дозволяє організувати надійний обмін даними між клієнтською та серверною частинами. Запропоновані архітектурні рішення сприяють підвищенню зручності користування системою, масштабованості та продуктивності програмного забезпечення.

Таким чином, актуальність кваліфікаційної роботи Осипенко І. В. не викликає сумнівів, а запропоновані методи, підходи та результати їх застосування можуть бути використані у практичній діяльності підприємства.

Науково-виробничому підприємству “Спільна Справа”, ТОВ магістром Осипенко І.В. передано програмне забезпечення, яке дозволяє покращити процеси комунікації користувачів, оптимізувати інформаційний обмін та підвищити ефективність використання сучасних мобільних технологій.

Члени комісії:

Головний спеціаліст Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО

Додаток Д (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна система для комунікації користувачів з використанням Flutter і REST API»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 4.38 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного 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 академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Овчинников К.В., доц. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Богач І.В., проф. каф. АІІТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Осипенко І.В.
(підпис) (прізвище, ініціали)