Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка кросплатформенного мобільного додатку»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Катерина КАЛІНЧЕНКО/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим АНДРЮЩЕНКО/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of a cross-platform mobile application»
according to educational curriculum «Software engeneering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1812:	/Kateryna KALINCHENKO/
Scientific Supervisor:	/Vadym ANDRIUSHCHENKO/
Normative controller:	/Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Калінченко Катерині Олегівні
1. Тема роботи: «Розробка кросплатформенного мобільного додатку»
Керівник роботи: Андрющенко Вадим Олександрович, доцент
затверджені наказом № 77 ст від 08.12.2021

2. Строк подання студентом роботи: __.__.202_ р.

3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):

Вступ

Звіт та аналіз вимог до мобільного додатку

Проектування мобільного додатку

Розробка прототипу мобільного додатку

Розробка мобільного додатку

Тестування мобільного додатку

Загальні висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Презентація

Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи
1	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання	31.01.22 - 18.02.2022
2	Програмування та відлагодження програми	19.02.22 - 20.05.22
3	Тестування програми	20.05.22 - 27.05.22
4	Розробка, узгодження та затвердження програмної документації	27.05.22 - 12.06.22
5	Подання кваліфікаційної роботи до кафедри	07.06.22
6	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	22.06.22

Студент _____ Катерина КАЛІНЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. ВадимАНДРЮЩЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 8 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області, а також проводиться опитування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 13 сторінок;

– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних,формалізація задачі, розробка фізичного проекту, приводиться опис об'єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 10 сторінок;

– тесдтування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами «чорного» та «білого» ящика. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 7 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складає 1 сторінку;

– додатки – містить технічне завдання і основну частину робочого проекта.

Кількість таблиць: 3 штук. Кількість рисунків: 14 штук.

Ключові слова: мессенджер, чат, повідомлення.

ЗМІСТ

ВСТУП

Запорукою успіху у досягненні поставленої мети завжди є чіткість поставленної задачі, наскільки добре продумані та прописані вимоги та умови виконання. Найчастіше в процессі роботи підприємств та розробки програмного забезпечення, як окремого випадку, відбуваються деякі прикрі випадки такі як незрозуміла ціль, неточні вимоги або не всі випадки роботи розглядають, що значно затрудняє внутрішні процеси компаній.

Мета роботи полягає у створенні кросплатформенного мобільного додатку, який допоможе забезпечити комунікаційний канал.

Зазвичай люди часто страждають через відсутність чітко поставленої мети та чітких вказівок щодо перебігу роботи. Це неможливо попередити, бо всі мають різний рівень знань та досвіду. Цю прикрість можна трохи згладити та звести майже до мінімуму установивши комунікаційні канали між працівниками та ввести спеціальну корпоративну культуру, що заохочує ставити питання, уточнювати незрозумілі моменти.

Якщо підприємство працює з якимись конфеденційними даними, що небезпечно передавати у відкритому вигляді через існуючі мессенджери, то практичне призначення створення окремого додатку для окремої компанії дало б змогу для спілкування між працівниками.

1. ЗБІР ТА АНАЛІЗ ВИМОГ

Мета встановлення вимог полягає в тому, щоб дати розгорнуте визначення функціональних - а також не функціональних вимог, які учасники проекту очікують затвердити в системі, що реалізується та розгортається.

Методи збору вимог до програмного забезпечення При вивченні предметної області та збору вимог про майбутній об’єкт інформатизації використовується методологія системного аналізу. На цій стадії розробники повинні уточнити межі вивчення функціональних вимог до програми, вхідні та вихідні дані для функціонування програми та визначити коло користувачів майбутньої програми.

Метод бесіди – психологічний вербально-комунікативний метод, що полягає у веденні тематично спрямованого діалогу між дослідником і респондентом з метою отримання відомостей від останнього. Метод бесіди і консультацій найчастіше проводиться з керівниками підприємств і підрозділів або у формі ділової консультації з фахівцями з питань, що 40 мають глобальний характер і належать до визначення проблем і стратегій розвитку та управління підприємством.

Прототипування (prototyping) – це найбільш часто використовуваний сучасний метод виявлення вимог. Програмні прототипи конструюються для візуалізації системи або її частини для замовників з метою отримання їх відгуків. Прототип – це дуже ефективний спосіб виявлення вимог, які важко отримати від замовника за допомогою інших засобів.

Еволюційний прототип (evolutionary), який зберігається після виявлення вимог і використовується для створення кінцевого програмного продукту. Еволюційний прототип націлений на прискорення поставок товарів. Як правило, він концентрується на ясно викладених вимогах, тому першу версію продукту можна надати замовнику досить швидко (хоча її функціональні можливості, як правило, неповні)

При виконанні роботи були використані такі методи збору та аналізу вимог як метод діалогу та метод еволюційного прототипування. Комбінування методу діалогу та еволюційного прототипування дозволило швидко та зручно визначити вимоги та розробляти програму крок за кроком, поки усі вимоги не були покриті та погоджені із замовником.

Після проведеного збору та аналізу вимог, зібрана інформація була проаналізована і на її основі було розроблено бізнес-процес системи.

Бізнес процес для моделювання виконується наступним чином:

1. Користувач а створює акаунт, заходить в створений акаунт.
2. У користувача а є список людей, що вже зареєстровані, він обирає будь-кого (користувач Б) й відправляє йому повідомлення.
3. Користувач Б отримує повідомлення.

Бізнес процес є завершеним, так як користувач отримав усю необхідну інформацію про змодельовану систему у вигляді діаграм та у виглядів чистих розрахунків.

2. ОПИС АНАЛОГІВ

Існує безліч аналогів месенджерів, що частково містять прекрасні приклади, щодо можливого функціоналу та безліч ідей для оновлень додатка. Нижче представлені найбільш розповсюджені мессенджери, що використовуються для створення як робочого процессу так і для особистого спілкування.

Telegram

Telegram — популярний додаток для обміну повідомленнями, який широко використовується, оскільки він пропонує деякі покращені функції конфіденційності та шифрування, а також підтримку функцій чату для великих груп. Він також не пов’язаний з іншими платформами соціальних мереж (наприклад, Facebook Messenger і WhatsApp належать Facebook), що робить сервіс більш привабливим для деяких.

Плюси

Наскрізне шифрування: під час використання режиму секретного чату спілкування повністю шифрується від кінця до кінця, що робить спілкування практично куленепробивним. Ви також можете отримати наскрізне шифрування в таких програмах, як WhatsApp і Signal.

Повідомлення, що самознищуються: секретні повідомлення можна налаштувати на самознищення через певний період часу, що робить їх ще більш безпечними. Це схоже на те, що ви можете робити в таких програмах, як Snapchat, Instagram та Facebook Messenger.

Великі розміри файлів: Telegram підтримує вкладення файлів розміром до 2 ГБ. Це одна з областей, де Telegram перевершує практично всі інші програми для обміну повідомленнями. Тільки Skype підходить навіть віддалено, обмежуючи файли розміром 300 МБ. Більшість додатків набагато більш обмежені — наприклад, WhatsApp становить лише 16 МБ.

Мінуси

Обмежена база користувачів: хоча Telegram має кілька сотень мільйонів активних користувачів щомісяця, це все ще значно менше, ніж понад 1 мільярд активних користувачів у WhatsApp або Facebook Messenger.

Це означає, що є велика ймовірність, що ваші друзі та контакти не використовують програму.

Нові оголошення користувачів (повідомлення про створення аккаунту) можуть порушувати конфіденційність: одна з головних причин, чому багато людей приєднуються до Telegram, — це можливість надсилати безпечні та приватні повідомлення. Ось чому засмучує те, що ваші контакти, які вже в програмі, отримують сповіщення, коли ви приєднуєтеся.

Це місце збору прихильників теорії змови та груп ненависті: оскільки популярні соціальні медіа-сервіси дедалі більше знищують групи QAnon, неонацистів та інші групи ненависті та теоретиків змови, Telegram став привабливим місцем збору для багатьох із цих груп через посилення політики безпеки та дозволеного вмісту, хоча Telegram активізував зусилля, щоб вимкнути облікові записи, якими керують групи ненависті та екстремістів.

Slack

Slack — це корпоративний месенджер. Його запустили в тестовому режимі в серпні 2013 року, публічний реліз відбувся 12 лютого 2014 року. У перший день тестування в системі зареєструвалося 8 тисяч компаній. Завдяки своєму зростанню Slack став бізнес-додатком, який показав найбільше зростання в історії. Станом на 2020 рік сервісом користувалися 750 000 компаній і 12 мільйонів користувачів.

Вартість

Slack в основному безкоштовний. Ви зможете переглядати та шукати лише 10 000 останніх повідомлень вашої команди, і ви будете обмежені 5 Гб пам’яті файлів і 10 додатками або користувацькими інтеграціями (докладніше про це пізніше).

Наразі доступні три рівні цін: безкоштовний, стандартний (8 доларів або 5,25 фунтів стерлінгів на місяць за користувача) і плюс (15 доларів або 9,75 фунтів стерлінгів на місяць за користувача). Четвертий рівень, який називається Enterprise Grid, має індивідуальні ціни для компаній.

Доступність мессенджера

Щоб отримати максимальну віддачу від Slack, ми рекомендуємо вам встановити мобільний додаток (iOS / Android) і настільний додаток (Mac / Windows). В іншому випадку ви будете обмежені доступом до Slack через вікно браузера.

Базові особливості

Налаштування власного профіля

Натисніть своє ім’я користувача або стрілку спадного меню поруч із назвою команди Slack, щоб отримати доступ до меню налаштувань. Звідси ви можете налаштувати свої параметри (теми для бічної панелі, повідомлень, стиль смайлів тощо), переглянути свій профіль та обліковий запис, отримати доступ до сторінки довідки/відгуку, отримати програми Slack (докладніше про це пізніше) і відключитися .

Якщо ви адміністратор, ви також можете отримати доступ до налаштувань команди, виставлення рахунків тощо. Учасники команди бачитимуть лише параметри доступу до інтеграції додатків, налаштувати слабість і перейти до каталогу команди.

Сповіщення про повідомлення

Натисніть значок дзвіночка у верхньому правому куті бічної панелі, щоб налаштувати параметри сповіщень. Ви можете відкласти їх, налаштувати розклад режиму «Не турбувати» тощо. Коли ви вперше входите в Slack, увімкнено майже всі типи сповіщень. Але сервіс пропонує ряд різних способів керування ними.

Наприклад, кожен канал навіть має індивідуальні налаштування сповіщень. Щоб перейти до цих налаштувань, відкрийте канал, який потрібно налаштувати, потім клацніть його назву вгорі екрана та виберіть Параметри сповіщень каналу.

Особливі помітки

Усе, що ви позначили зірочкою, відображатиметься над розділом «Канали», а також під назвою вашої команди та іменем користувача на бічній панелі. Зірочки — це спосіб позначити елемент у Slack як важливий.

Ви можете позначити канали або прямі повідомлення, щоб перемістити їх у верхню частину лівої бічної панелі. Ви також можете позначити повідомлення в Slack, щоб ви могли легко повернутися до них пізніше.

Канали

Під назвою вашої команди Slack та власним іменем користувача ви побачите розділ під назвою «Канали» на бічній панелі. Канали – це чати. Ви можете називати кімнати чату на основі будь-якого, включаючи проект (сценарій), тему (музику) або команду (продажі). І ви можете зробити їх загальнодоступними або приватними.

Якщо канал загальнодоступний, усі члени вашої команди можуть приєднатися, але якщо канал є приватним, приєднатися можуть лише вибрані люди. І, нарешті, якщо ви клацнете назву розділу «Канали», ви побачите нове вікно, яке дозволить вам переглядати та сортувати всі канали.

Створення каналів

Щоб створити новий канал, натисніть кнопку + поруч із пунктом Канали на бічній панелі. Потім ви побачите варіанти створення загальнодоступного або приватного каналу. Крім того, вам буде дозволено назвати канал, запросити інших приєднатися та коротко описати мету каналу.

Прямі повідомлення

Під каналами ви побачите розділ прямих повідомлень на бічній панелі, а потім імена всіх запрошених до вашої команди Slack. Натисніть будь-яке ім’я, щоб надіслати цій людині приватне повідомлення один на один. Якщо натиснути кнопку + поруч із розділом «Прямі повідомлення», відкриється вікно, у якому можна знайти або розпочати бесіду з прямими повідомленнями.

Запросіть людей

Якщо ви адміністратор, ви побачите кнопку, щоб запросити більше людей приєднатися до вашої команди. Ця кнопка знаходиться під розділом «Прямі повідомлення». Якщо ви натиснете його, у вас буде три варіанти: повноправні учасники, обмежені облікові записи та одноканальні гості.

Перший варіант надає запрошеному члену команди повний доступ до повідомлень і файлів у будь-якому загальнодоступному каналі та каталозі команди. Другий варіант дозволяє запрошеному члену команди бачити лише частковий каталог команди та надає йому/їй доступ до вибраних каналів і файлів. І третій варіант обмежує запрошеного члена команди повідомленнями та файлами в одному каналі.

Публічні посилання

Якщо ви хочете поділитися посиланням з кимось не в Slack, вам потрібно створити загальнодоступне посилання. Для цього наведіть курсор на файл, для якого ви хочете створити посилання, натисніть значок із трьома крапками у верхньому правому куті та виберіть Створити зовнішнє посилання. Скопіюйте посилання в буфер обміну та натисніть Готово.

Закріплення повідомлень і файлів

Якщо у вашому каналі є щось, що вам потрібно побачити, або важливий файл, який ви хочете легко знайти, його можна закріпити на каналі, щоб він відображався в меню відомостей про канал. Щоб закріпити, наведіть на нього курсор, потім клацніть гвинтик праворуч і виберіть Закріпити повідомлення. Закріплене повідомлення залишиться там, доки ви його не видалите.

Встановити нагадування

Ви можете використовувати Slack для нагадувань. Наприклад, якщо вам потрібно нагадати собі опублікувати історію за 30 хвилин, ви можете ввести «/remind me in 30 minutes to publish», і Slack нагадає вам приватно.

Skype

Skype – це телекомунікаційна програма, яка спеціалізується на наданні відеочату та голосових дзвінків між комп’ютерами, планшетами, мобільними пристроями, консоллю Xbox One та розумними годинниками через Інтернет. Skype також надає послуги миттєвого обміну повідомленнями. Користувачі можуть передавати текст, відео, аудіо та зображення. Skype дозволяє проводити відеоконференції. Станом на

березень 2020 року Skype використовували 100 мільйонів людей щомісяця і 40 мільйонів людей щодня.

Вперше випущений у серпні 2003 року, Skype створили шведський підприємець Ніклас Зеннстрьом і данський підприємець Янус Фрііс у співпраці з чотирма естонськими розробниками. У вересні 2005 року eBay придбала Skype за 2,6 мільярда доларів. У вересні 2009 року Silver Lake, Andreessen Horowitz та Інвестиційна рада Канадського пенсійного плану оголосили про придбання 65% Skype за 1,9 мільярда доларів у eBay, що приписує підприємству ринкову вартість у 2,92 мільярда доларів. Microsoft купила Skype у травні 2011 року за 8,5 мільярдів доларів. Штаб-квартира підрозділу Skype знаходиться в Люксембурзі, але більшість команди розробників і 44% усіх співробітників підрозділу все ще знаходяться в Таллінні та Тарту, Естонія.

Skype дозволяє користувачам спілкуватися через Інтернет голосом, за допомогою мікрофона, відео за допомогою веб-камери та обміну миттєвими повідомленнями. Skype реалізує бізнес-модель freemium. Дзвінки Skype-to-Skype є безкоштовними, тоді як дзвінки на стаціонарні та мобільні телефони (через традиційні телефонні мережі) оплачуються через систему облікових записів на основі дебету під назвою Skype Credit. Деякі адміністратори мережі заборонили Skype у корпоративних, державних, домашніх і освітніх мережах, посилаючись на такі причини, як неналежне використання ресурсів, надмірне використання пропускної здатності та проблеми безпеки.

Спочатку Skype містив гібридну однорангову і клієнт-серверну систему. З травня 2012 року Skype повністю працює на супервузлах, які керуються Microsoft. Розкриття інформації про масове спостереження 2013 року показало, що Microsoft надала спецслужбам безперешкодний доступ до супервузлів і комунікаційного вмісту Skype.

Нижче наведено деякі інші переваги, які Skype може принести дому та бізнесу сьогодні.

1. Через Skype доступні параметри спільного доступу до екрана.

Проводити віддалену презентацію продажу стає набагато простіше завдяки можливості Skype ділитися екраном. Зустрічі можна проводити за допомогою цього сервісу, а інформацію, якою потрібно поділитися, кожен учасник може самостійно переглядати. Це значно полегшує обговорення, створює індивідуальний вплив та передає інформацію більш запам'ятовуваним способом.

2. Це сервіс, який неймовірно легко встановити.

Все, що потрібно, щоб розпочати роботу в Skype, це знати, як відвідати їхній сайт або знайти його програму та завантажити її. Звідти інструкції на екрані проведуть користувачів через решту процесу налаштування. За лічені хвилини Skype може самостійно налаштувати систему, і єдине, що потрібно зробити користувачам, — це зв'язати обліковий запис із програмою зв'язку або створити новий обліковий запис. Навіть якщо зараз не встановлено Skype, для його запуску потрібно всього кілька хвилин.

3. Це надійний сервіс, який пропонує цілодобові контакти.

Немає встановленого часу, коли людям дозволено використовувати Skype. Його можна використовувати, коли це необхідно, і багато контактів, які дозволені через службу, безкоштовні. Багато дзвінків ініціюються за пару кліків, і дні скинутих дзвінків практично припинилися. Користувачі можуть покладатися на програму для завантаження та здійснення дзвінка через Інтернет або сигналу даних, коли вони хочуть здійснити виклик.

4. Він дозволяє здійснювати груповий дзвінок без необхідності групових функцій.

Однією з найприємніших особливостей Skype є те, що кілька людей можуть спілкуватися один з одним за допомогою одного з'єднання без будь-якого іншого обладнання, послуг або оплати. Поки є місце перед камерою комп'ютера або смартфона, ця особа може брати участь у відеодзвінку, а її голос буде піднятий мікрофоном, щоб вона могла брати участь у розмові. Однак для індивідуальних дзвінків гарнітура може так само добре працювати і для приватних розмов.

5. Платні підписки на Skype дешеві.

У Сполучених Штатах підписка на Skype починається від 2,99 доларів на місяць. У Китаї вартість підписки становить 1,19 доларів на місяць. Існує також безкоштовний місяць Skype Unlimited World, який доступний для користувачів, де можна здійснювати необмежену кількість безкоштовних дзвінків на стаціонарні та мобільні телефони зі Skype, а потім лише 13,99 доларів США на місяць. Це робить його однією з найдоступніших глобальних послуг VoIP, яка доступна сьогодні, і той же обліковий запис можна використовувати на будь-якому комп'ютері чи смартфоні.

WeChat

WeChat, що належить китайській компанії Tencent, є одним з найпопулярніших додатків для обміну повідомленнями в соціальних мережах у всьому світі. WeChat працює подібно до інших програм обміну повідомленнями, таких як Facebook messenger або WhatsApp, що дозволяє користувачам надсилати або отримувати миттєві повідомлення. Крім цього, він також дозволяє здійснювати аудіо та відеодзвінки. Через нього ви також можете обмінюватися файлами та документами. Ви можете завантажити WeChat у багатьох магазинах додатків і встановити його на свій пристрій. Існують програми WeChat як для смартфонів, так і для комп'ютерів з різними операційними системами. Отже, плюси і мінуси додатку представлені нижче.

Відкриття облікового запису WeChat:

На відміну від багатьох інших платформ створенних для обміну повідомленнями, створення облікового запису може бути дещо складним. Вам потрібно більше, ніж ім'я користувача та пароль. Вас має підтвердити той, хто вже має обліковий запис. Це робиться шляхом сканування QR-кодів іншими обліковими записами.

Ось деякі плюси та мінуси WeChat:

Плюси:

1. Абсолютно безкоштовно:

Використання WeChat безкоштовне і вам нічого не коштує. Це означає, що він пропонуватиме всі свої послуги користувачам, поки він під’єднається до Інтернету. Це одна з причин того, що він має сотні мільйонів активних користувачів щодня.

2. Обмін файлами простіше:

WeChat значно полегшує обмін файлами. Ви можете надсилати або отримувати документи розміром до 100 Мб. Він підтримує PDF, слайд-шоу та інші розширення файлів.

3. Оплата WeChat:

WeChat пропонує користувачам послугу віртуальних гаманців під назвою WeChat Pay. Це швидкий та ефективний спосіб здійснення платежів замість готівки. Користувачі можуть оплатити, спочатку відсканувавши QR-код, підтвердивши суму та оплативши після підтвердження пароля від платника. Це швидка та ефективна форма транзакції. Крім того, це також безпечно. WeChat тепер також дозволяє не китайцям використовувати його сервіс.

4. Додані функції:

Подібно до іншого подібного додатка WeChat дозволяє базову функцію обміну повідомленнями. Окрім цього, є функція під назвою «Моменти», де користувачі можуть ділитися своїми історіями та статусом зі своїми друзями. Крім цього, користувачі отримують можливість ділитися своїм місцезнаходженням з друзями та грати з ними в міні-ігри.

Тепер ми отримали деякі з хороших моментів, тому давайте поговоримо про його недоліки.

Мінуси WeChat:

1. Проблема конфіденційності:

Tencent має доступ до всіх наших конфіденційних даних. Крім того, Tencent також співпрацює з урядом Китаю. Отже, можна з упевненістю

сказати, що уряд стежить за вашою діяльністю через WeChat. Але ви також повинні звернути увагу, що ви погодилися з його політикою під час створення облікового запису WeChat.

2. Питання безпеки:

WeChat не використовує методи наскрізного шифрування для обміну повідомленнями. Отже, можливо, що ваше повідомлення може побачити третя сторона або, ймовірно, використає його неправильно.

На закінчення можна сказати, що WeChat – цікавий додаток. З точки зору деяких послуг, це навіть краще, ніж більшість існуючих програм для обміну повідомленнями в соціальних мережах. Крім того, це цікавіше і веселіше. Він, безумовно, буде більш популярним за межами Китаю, якщо використовувати його навіть невелика група людей. Отже, це плюси і мінуси WeChat.

Висновки до розділу

Як видно із наведених прикладів, що вони реалізовують безліч зручних функцій, але купа з них є непотрібними, відволікають від основної задачі користувача, а саме робочого процесу.

3. ЗОВНІШНЄ ПРОЄКТУВАННЯ

Функціональне призначення

Програма моделює мессенджер з можливістю реєстрації та відправлення особистих повідомлень.

Експлуатаційне призначення

Дане програмне забезпечення допомагає установити комунікаційні канали між співробітниками однієї організації. Програма дає можливість спілкуватися за допомогою особистих повідомлень.

Функціональні вимоги

Користувач може виконувати наступні варіанти використання:

- створити профіль;
- увійти до свого акаунта;
- вийти зі свого акаунта;
- редагувати інформацію про свій профіль;
- отримати список зареєстрованих осіб;
- відправити текстове повідомлення;
- відправити стікер-повідомлення;
- відправити фото-повідомлення;
- отримати всих типів повідомлення.

Діаграма використання програми наведена на рис. 5.1. Дана діаграма може розширюватись при подільшій розробці та впровадженні нових функцій до додатку. На ній представлене тільки те, що на даний момент потребується від додатку.

Рисунок 3.1 - Діаграма використання програми

4. ВНУТРІШНЄ ПРОЕКТУВАННЯ

Статична архітектура системи

Програма поділена на дві частини. Структура зображена на рисунку 4.1. Першу можна назвати авторизація. Вона складається зі сторінки Sign in та Sign up, для того, щоб взаємодіяти з базою даних та авторизувати чи створити користувача створений authProvider. Друга відповідає за безпосередню головну частину додатка, можна назвати цю частину Main. Вона складається з трьох сторінок Home page (або Users page), Chat page, Settings Page. Для взаємодії з базою даних Home page використовує HomeProvvider, Chat page - Chat Provider, Settings Page - Settings Provider.

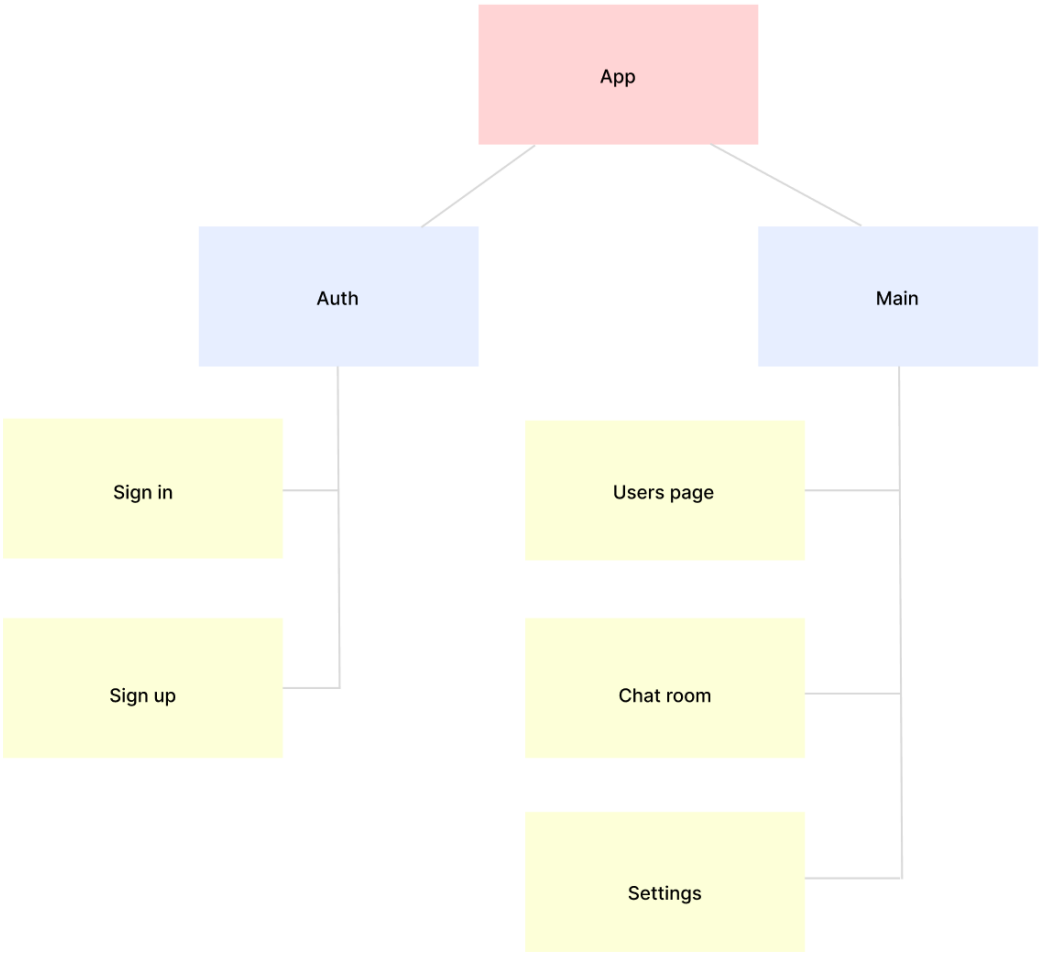

Рисунок 4.1 - Приблизна схема модулів програми

Проектування інтерфейсу

Проект має тільки світлу кольорову гамму. Синій (0xff203152) - акцент, білий (0xffE8E8E8) та сірий (0xffaeaeae).

Приблизний інтерфейс чату представлений на схемі рис. 4.2. Він не обов’язково має співпадати піксель до пікселю, але базовий концепт співпадає.

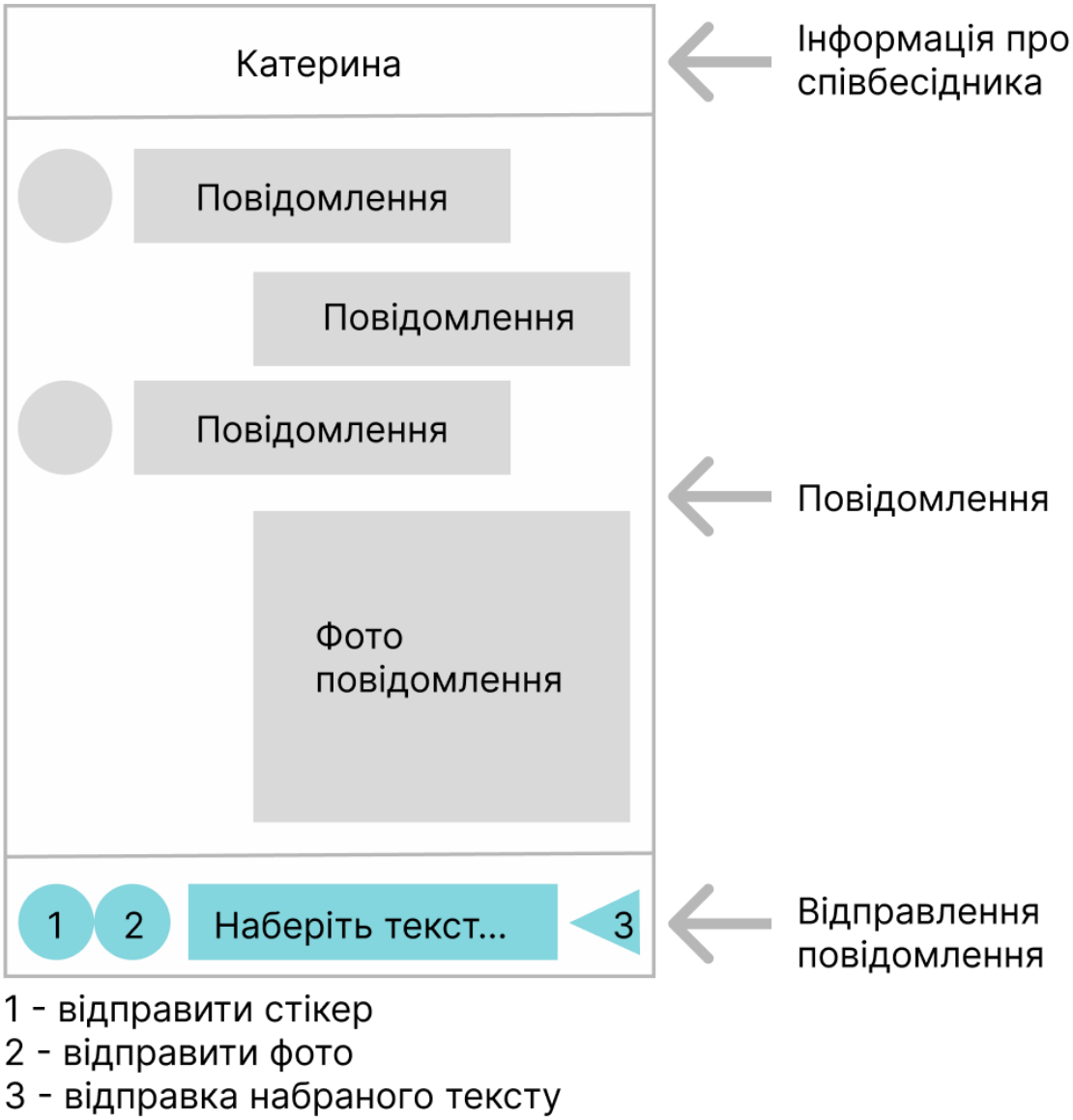

Рисунок 4.2 - Інтерфейс чату

Список користувачів повинен містити інформацію користувачів, кожен користувач має свою “картку”, що виглядає як на рис. 4.3.

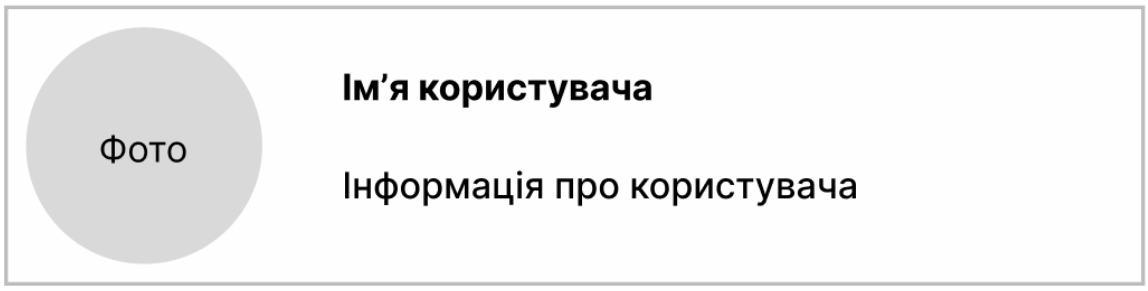

Рисунок 4.3 - Інтерфейс картки користувача

Структура (логічна схема) баз та сховищ даних

В проекті використовується NoSQL база даних. Вона створена за допомогою сервіса Firebase. Структура зображена на рис. 4.4, де Користувач і Повідомлення це назви коллекцій, а все інше виступає документами.

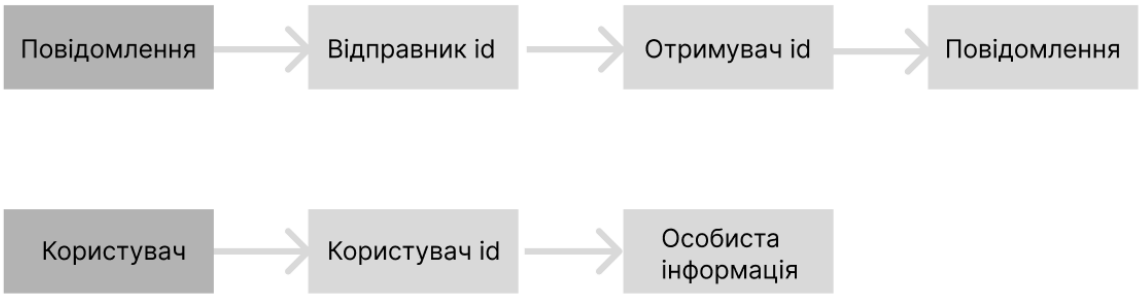

Рисунок 4.4 - Структура БД

Система взаємодіє з базою даних через провайдери, що відправляють запрос і отримують результат. З їх допомогою можна підписатись на стрім подій на відображати зміни (наприклад те, що в чаті змінилась кількість повідомлень).

5. РОЗРОБКА ПРОГРАМИ

Flutter — це безкоштовний мобільний інтерфейс користувача з відкритим вихідним кодом, створений компанією Google і випущений у травні 2017 року. Декількома словами, він дозволяє створювати нативний мобільний додаток лише з однією кодовою базою. Це означає, що ви можете використовувати одну мову програмування та одну кодову базу для створення двох різних програм (для iOS та Android).

Flutter складається з двох важливих частин:

1. SDK (набір програмного забезпечення): набір інструментів, які допоможуть вам розробити ваші програми. Це включає інструменти для компіляції вашого коду в рідний машинний код (код для iOS і Android).

2. Framework (бібліотека інтерфейсу користувача на основі віджетів): набір багаторазових елементів інтерфейсу користувача (кнопки, введення тексту, повзунки тощо), які можна персоналізувати для власних потреб.

Деякі переваги, які пропонує Flutter, а також деякі їхні недоліки.

Переваги.

1. Flutter пропонує широкий спектр переваг як для бізнесменів, так і для розробників. Flutter пропонує хорошу якість за розумну вартість, а також чудову зручність та швидкість.

2. Розробка йде швидше, якщо один і той самий код використовується для додатків iOS і Android. Єдина кодова база Flutter пришвидшує час виходу на ринок і значно скорочує витрати на розробку мобільних додатків.

3. Продуктивність додатків Flutter еквівалентна продуктивності рідних програм реального часу. На відміну від інших фреймворків, додаткам Flutter не потрібне сполучення (міст) для взаємодії з рідними компонентами. Оскільки ці мости зазвичай викликають проблеми з продуктивністю, це дає Flutter рішучу перевагу.

4. Функція «гарячого перезавантаження» (hot reload) Flutter дозволяє розробникам змінювати код на емуляторах, симуляторах або реальних пристроях і бачити результати в режимі реального часу. Змінений код

миттєво перезавантажується та оновлюється під час роботи програми. Перезавантаження не потрібно. Гаряче перезавантаження робить створення інтерфейсу користувача, додавання функцій і виправлення помилок простіше, ніж будь-коли.

Мінуси розробки додатків Flutter:

Хоча розробка додатків Flutter має багато позитивних моментів, важливо також враховувати менші аспекти фреймворка:

1. Flutter відносно новий. Хоча Flutter пропонує багато плагінів і компонентів інтерфейсу користувача, такі фреймворки, як Xamarin і React Native, пропонують набагато більший вибір. Хоча Flutter не буде новою структурою для блоку назавжди, зараз така ситуація.

2. Дарт не дуже популярний. Хоча це чудова мова програмування, розробники часто частіше розглядають такі мови, як Java або Kotlin.

3. Деякі компоненти доступні лише для iOS або Android, але не для обох. Ці типи компонентів частіше підтримують Android, оскільки Flutter походить від Google, і розробники Android зазвичай більше цікавляться Flutter, ніж розробники iOS.

Відомі компанії, що використовують Flutter:

1. Google ads. Ця мобільна версія настільної платформи дозволяє користувачам відстежувати рекламні кампанії, статистику та оновлювати бюджети чи ставки в режимі реального часу. Додаток забезпечує миттєві сповіщення, дозволяє редагувати ключові слова та підтримує спілкування з командою Google.

2. Xianyu — це платформа електронної комерції alibaba для вживаних товарів. Flutter дозволив корпорації розробити оптимізовану архітектуру з інноваційними функціями. Близько 50 мільйонів споживачів використовує цю програму.

3. My BMW app

Водії в 47 країнах використовують той самий інтерфейс для безперебійної взаємодії своїх транспортних засобів і мобільних пристроїв.

Додаток підтримує пряме спілкування з дилерами BMW та виробником. Він створений на основі зручності, надійності та безпеки.

Попри всі свої мінуси та недоліки флаттер ідеально підходить для MVP-початківців.

Якщо ви хочете якомога швидше продемонструвати свій продукт інвесторам, Flutter — хороший вибір.

Ось 4 основні причини використовувати його для свого MVP:

1. Розробити мобільний додаток за допомогою Flutter дешевше, оскільки вам не потрібно створювати та підтримувати дві мобільні програми (один для iOS і один для Android).

2. Для створення MVP вам потрібен лише один розробник.

3. Він продуктивний – ви не помітите різниці між рідною програмою та програмою Flutter.

4. Це красиво — ви можете легко використовувати віджети, надані Flutter, і персоналізувати його, щоб створити цінний інтерфейс користувача для своїх клієнтів (нижче ви можете знайти приклади програм, створених за допомогою Flutter).

Саме через це флаттер був обраний для написання даного проекту. Створення інтерфейсної частини зайняло мінімальну кількість часу якщо порівнювати з іншими фреймворками. Дарт (мова фреймворка), хоч і не розповсюджена, але дуже проста та можна сказати інтуїтивно зрозуміла для людей у яких є базові знання програмування та хто знайомий з мовою Java або JavaScript.

Для написання програми був використаний пекедж Provider. Flutter — це декларативна структура. На відміну від імперативної структури, Flutter не дозволяє змінювати віджет, який переважно є компонентом інтерфейсу користувача на екрані, після його визначення. Для того, щоб запобігти зайвим ребілдам сторінки був створений provider. Тобто дане рішення добре підходить для реалізації цього проекта

Firebase це сервіс, що предоставляє безліч можливостей для розробки. Ця платформа від Google проста в використанні та впровадженні в проект. В

ході розробки були використані такі сервіси як authentication для реєстрації користувачів, Storage для збереження файлів та Firestore для бази даних.

Блоксхема програми зображена на рис. 5.2.

Вона представляє головний алгоритм програми, що є структурованим та наочно представляє послідовні кроки виконання програми. Тобто він зображує як користувач може використовувати програму та показує всі розгалуження. Дана діаграма представляє суто флатер частину.

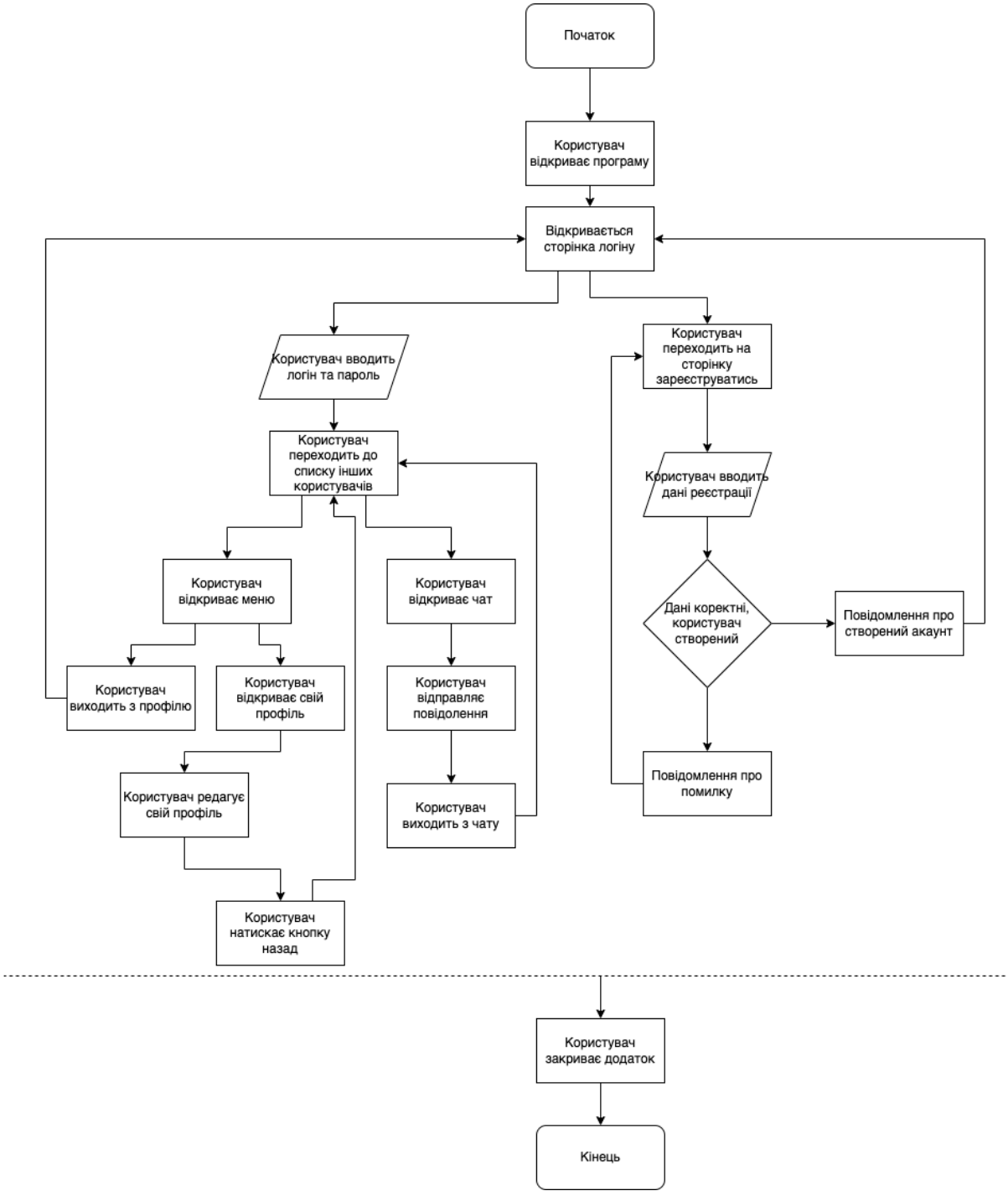

Рисунок 5.2 - Блоксхема роботи програми

.

6. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ ПРОГРАМИ

6.1 Стратегія тестування

Мета тестування − виявити ситуацію, коли результати роботи програми не відповідають вхідним даними. Найпростіший спосіб зробити це: перебрати всі можливі варіанти вхідних даних і перевірити правильність отриманих результатів. На жаль, скористатися цим способом майже ніколи не вдається. Навіть для найпростіших програм кількість варіантів вхідних даних виявляється астрономічним. Звідси сумний висновок: вичерпне тестування (тобто перебір всіх можливих варіантів виконання) для будь-якої нетривіальної програми неможливо.

Під час ведення будь-якого бізнесу однією з головних цілей завжди має бути забезпечення найкращого досвіду для кожного клієнта. Тестування мобільних додатків — це імперська фаза процесу розробки мобільних додатків. Усі основні проблеми, з якими стикається додаток, можна вирішити шляхом успішного тестування мобільного додатка. Крім того, ідеальне тестування мобільного додатка забезпечує легкий запуск програми на робочому сервері без будь-яких недоліків.

6.2 Тестування білою скринькою

Функція для відправлення повідомлень створює екземплярі повідомлень у ваідповідній колекції на Firestore.

```
void sendMessage(String content, int type, String groupChatId, String
currentUserId, String peerId) {
  final datetime = DateTime.now().millisecondsSinceEpoch.toString();
  DocumentReference documentReference = firebaseFirestore
      .collection(FirestoreConstants.pathMessageCollection)
      .doc(groupChatId)
      .collection(groupChatId)
      .doc(datetime);

  MessageModel messageChat = MessageModel(
    idFrom: currentUserId,
    idTo: peerId,
```

```
    timestamp: datetime,
    content: content,
    type: type,
  );


  FirebaseFirestore.instance.runTransaction((transaction) async {
    transaction.set(
      documentReference,
      messageChat.toJson(),
    );
  });
}
```

У явному вигляді функція приймає content (String) - текст повідомлення, type (int) - тип повідомлення (текст, стікер чи фото, інтежер отримується з перечислення при виклику метода), groupChatId (String) - ідентифікатор чата, currentUserId (String) - ідинтифікатор користувача, що відправляє повідомлення, peerId (String) - ідентифікатор користувача якому відправляється повідомлення.

У явному вигляді функція нічого не повертає.

Доцільним буде тестування функції методом припущень, бо зі сторони коду неможливо покривати тестами закриту від нас частину функцій, що знаходяться в сторонніх пекеджах.

Таблиця 6.1. - Тестування програми

Вхідні дані	Вихідні дані (те, що ми бачимо у базі даних)
Що буде якщо відіслати текстове повідомлення?	1653231300653 + Start collection + Add field content: "hi :)" idFrom: "YVP4qcv5l4QohRCldfXlZ6KU4pr1" idTo: "PAHpq2jA6RazsMu4q3GZO481O763" timestamp: "1653231300653" type: 0
Що буде якщо відіслати стікер?	1653231309403 + Start collection + Add field content: "img5" idFrom: "YVP4qcv5l4QohRCldfXlZ6KU4pr1" idTo: "PAHpq2jA6RazsMu4q3GZO481O763" timestamp: "1653231309403" type: 2
Що буде якщо видіслати фотографію?	1653231479809 + Start collection + Add field content: "https://firebasestorage.googleapis.com/v0/b/diploma-auth-d4e2f.appspot.com/o/1653231476949?alt=media&token=e2428540-4b4e-4546-994d-5d86a83d4cf3" idFrom: "YVP4qcv5l4QohRCldfXlZ6KU4pr1" idTo: "PAHpq2jA6RazsMu4q3GZO481O763" timestamp: "1653231479809" type: 1

6.3 Тестування чорною скринькою

Ситуація 1. При логіні невірно введений логін чи пароль. На екрані з’являється повідомлення, що зображено на рисунку 6.1.

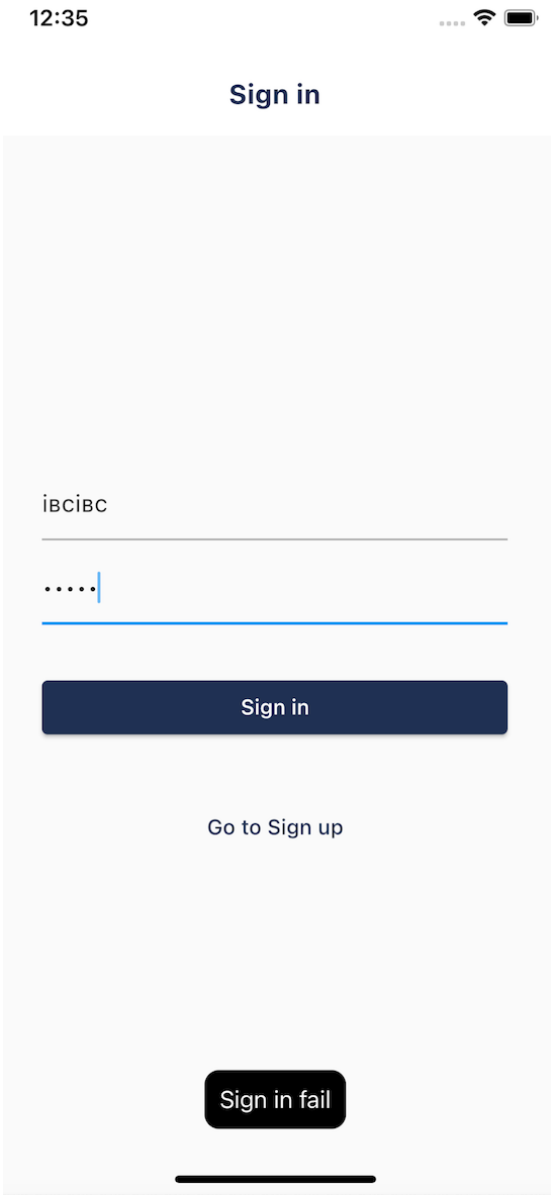

Рисунок 6.1 - Неправильний логін чи пароль

Ситуація 2. При реєстрації пароль та підтвердження паролю не співпадають. Зобажено на рисунку 6.2.

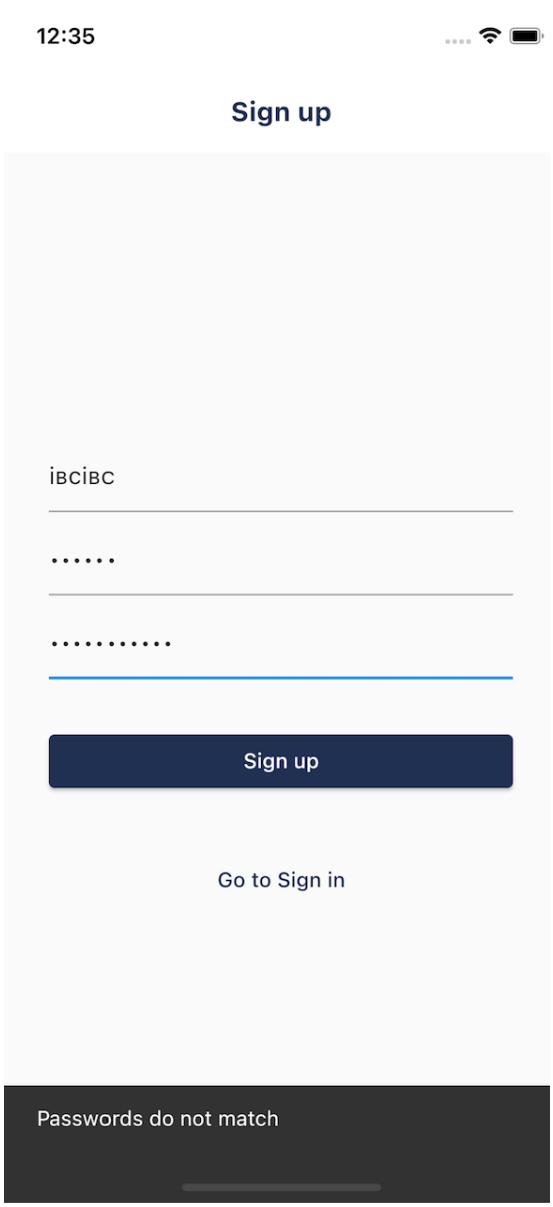

Рисунок 6.2 - Паролі не співпадають

Ситуація 3. При пошуку немає користувачів з нікнеймом. Ситуація зображена на рисунку 6.3.

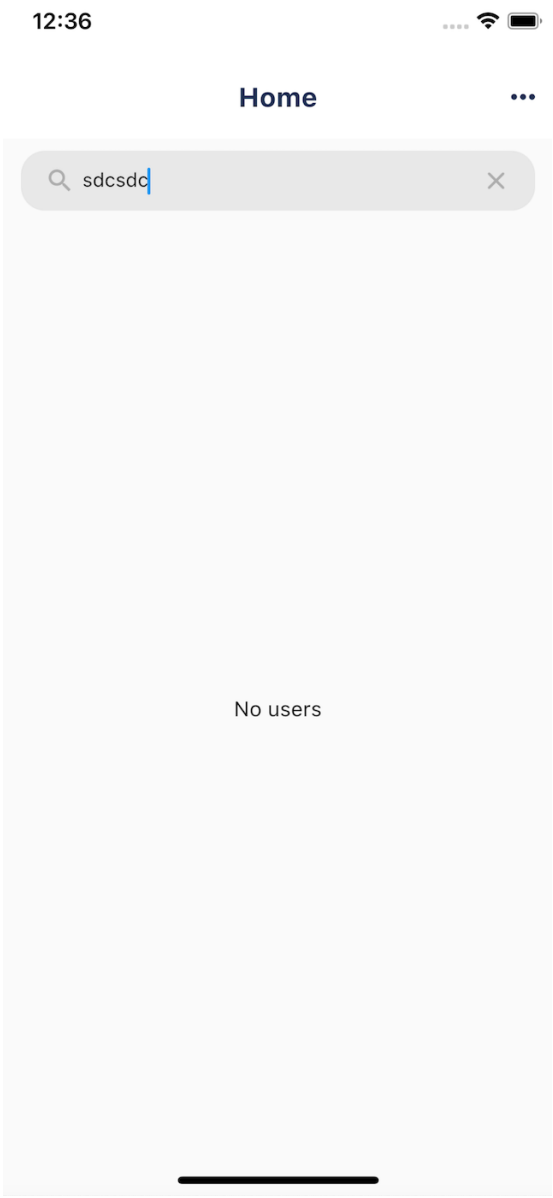

Рисунок 6.3 - Немає користувачів

Результати налагодження

При налагодженні програми було виявлено багато помилок. Наприклад при розміщенні полей на реєстраційній формі та формі логіна було небажане зміщення полей через неправильні їх розміщення та появу клавіатури. Така помилка досить розповсюджена при написанні кода, але вона дуже швидко усувається при перевірці та запуску проекта у дебаг режимі. Результати представлені в таблиці 6.2

Помилки виявлялись на ранній стадії та усувались з моменту їх виявлення. Так як фреймворк предоставляє 3 режими запуску програми (реліз, профайл та дебаг), відлагодження програми та усування помилок не є головною проблемою при розробці як інтерфейсу так і бізнес логіки додатку.

При запуску програми у режимі дебаг можна використовувати Flutter Performance для того щоб мониторити ресурси, що використовує система, як часто сторінка ребілдиться (чим частіше це відбувається, тим більше роботи виконує програма, тим повільніше вона працює).

Таблиця 6.2. Помилки, що виникли при розробці

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
Нижній оверфлоу на 200 логічних пікселей	При відкриванні клавіатури на сторінці логіна поля міняють своє місцеположення та створюють ситуацію	- перемалювати дизайн - змінити параметри розташування елементів - ігнорувати помилку	Були змінені параметри розташування елементів для збереження дизайнів та усунення помилки
При введенні логіна та пароля юзер не переходить в додаток	Юзер вводить правильні логін та пароль, додаток повідомляє, що все введено правильно, юзер залишається на екрані логіна	Перевірити, що відбувається при логіні юзера	Помилка усунена, коли був додатий провайдер для того, щоб моніторити стан залогінений юзер чи ні

7. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ

На завершальному етапі розробки програми було розроблено Чат з реєстрацією та входом до системи. Для запуску програми необхідно відкрити згенерований apk файл на android, або ipa файл на IOS. Після запуску програми, користувач потрапляє до сторінки логін рис. 7.1.

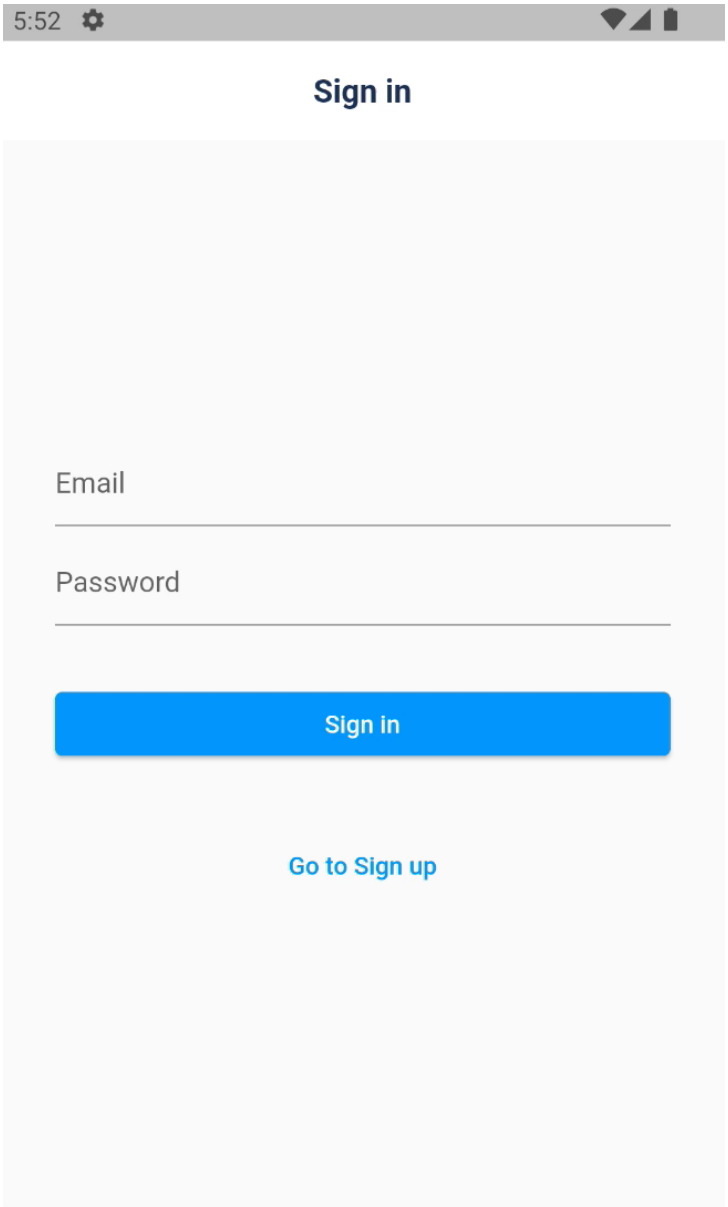

Рисунок 7.1 - Сторінка логіну

Можна перейти до вікна реєстрації та зарієструвати користувача за допомогою імейла рис 7.2.

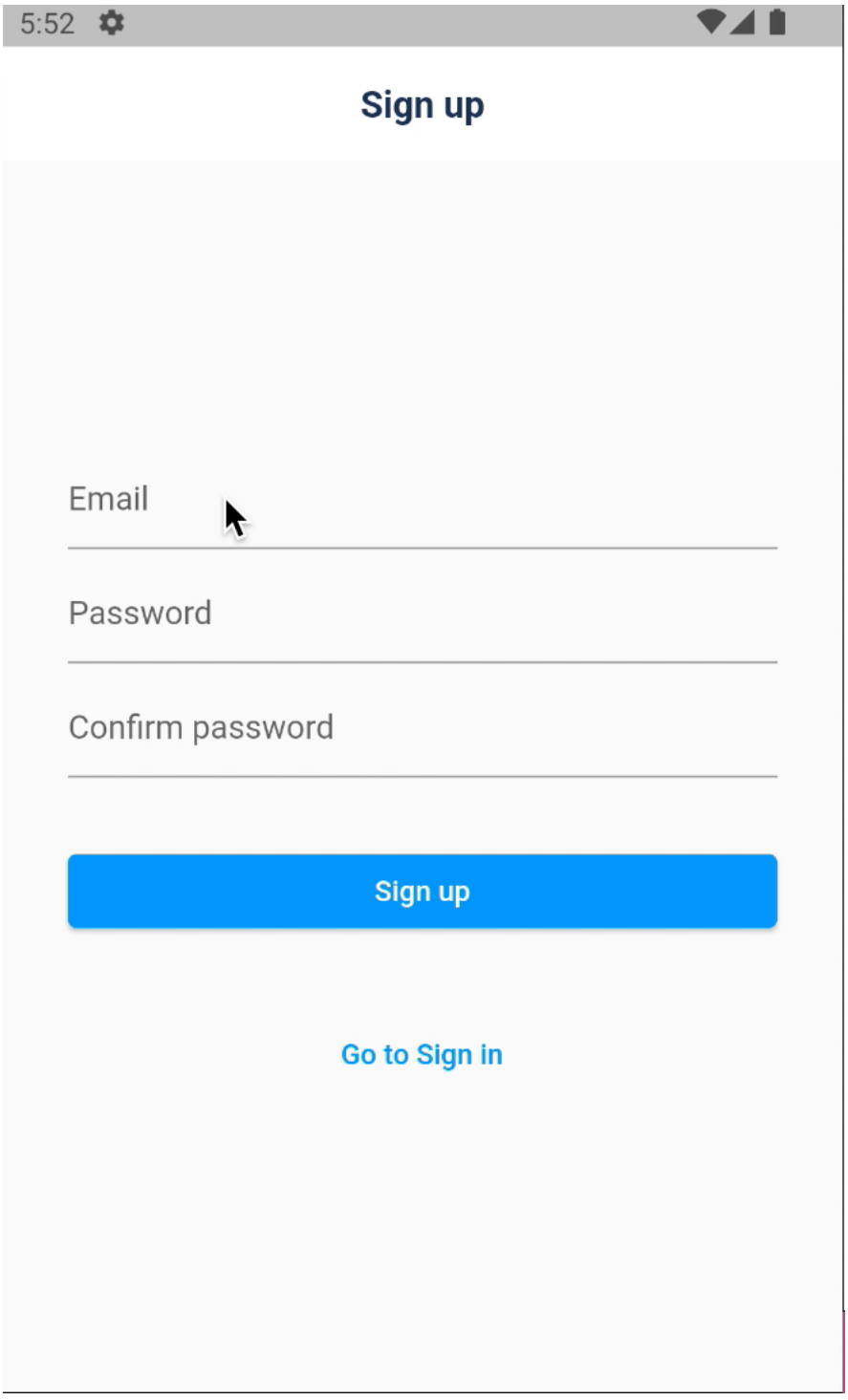

Рисунок 7.2 - Сторінка реєстрації

Можна редагувати профіль рис 7.3.

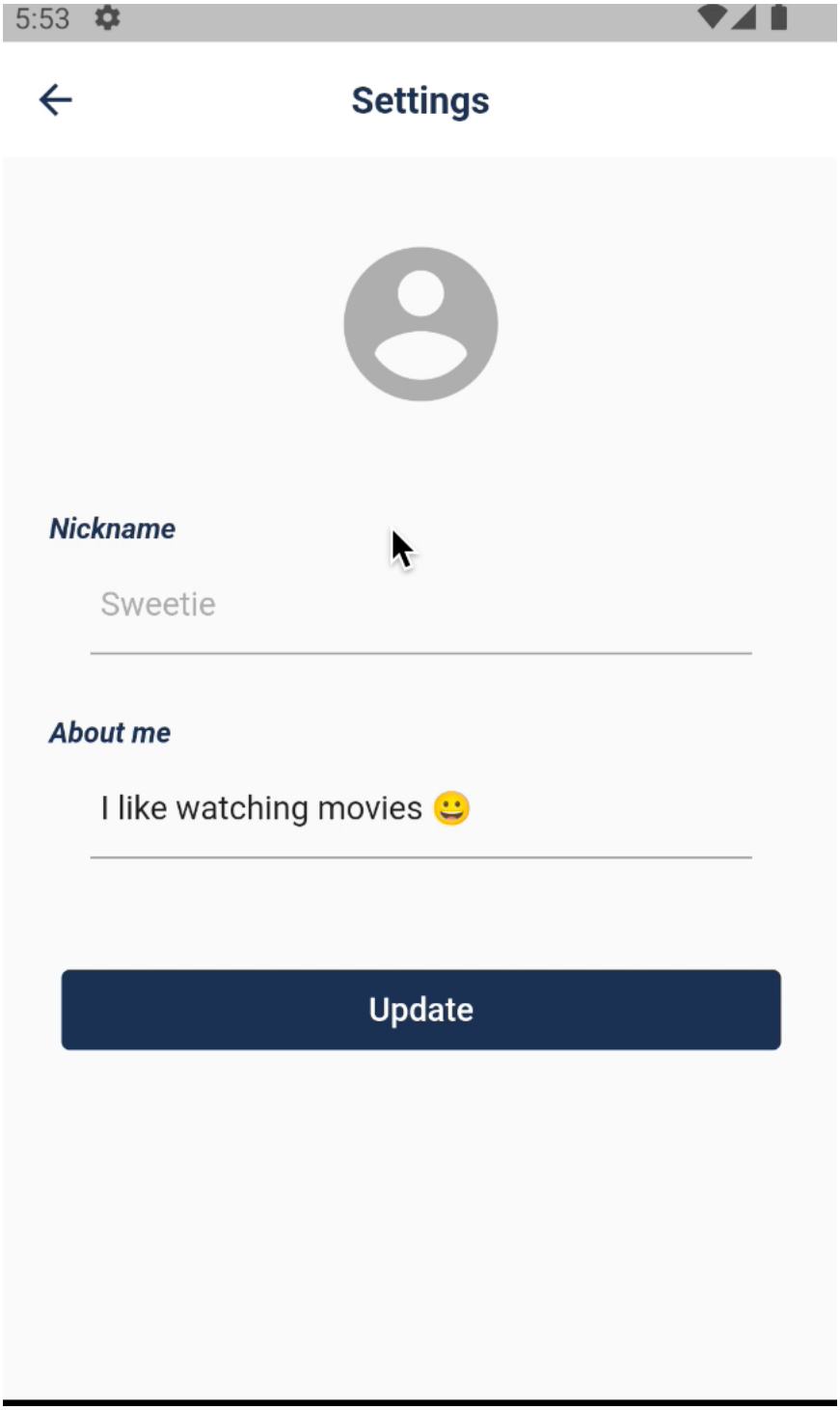

Рисунок 7.3 - Сторінка налаштування профілю

Користувач може переглянути список зареєстрованих осіб рис 7.4.

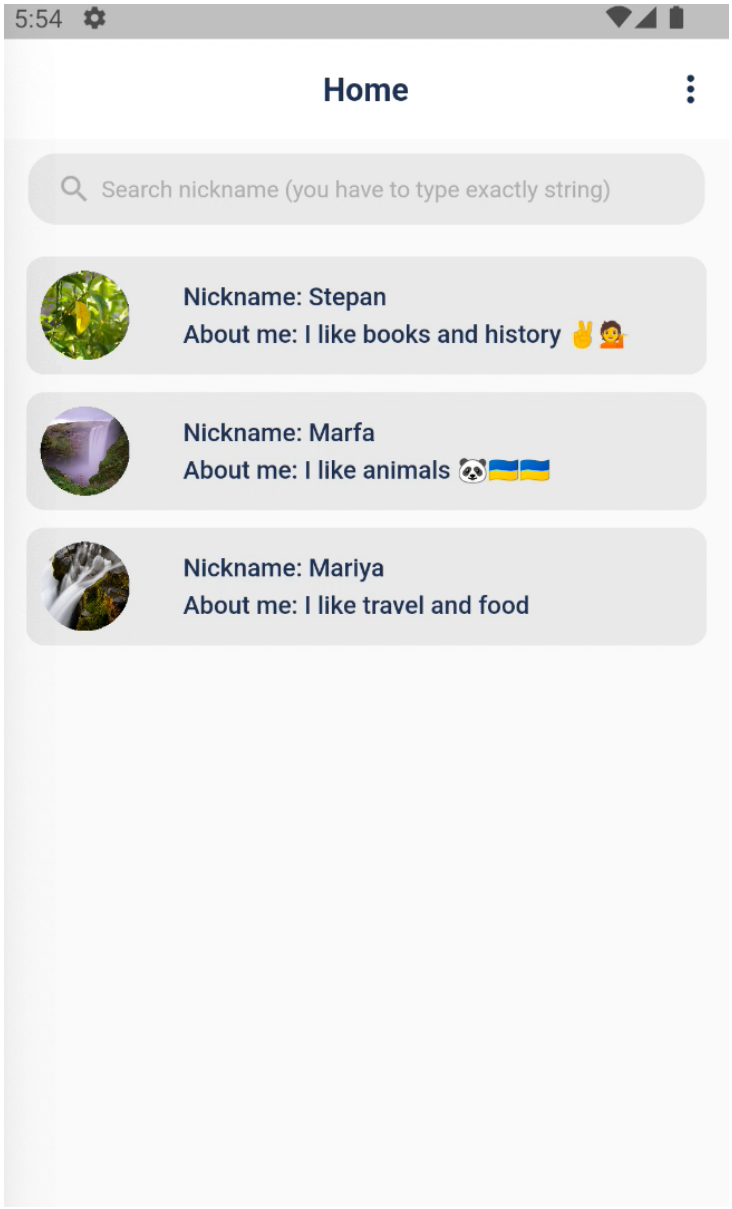

Рисунок 7.4 - Головна сторінка

Користувач має можливість відправити та одразу отримати повідомлення до зареєстрованого користувача рис 7.5.

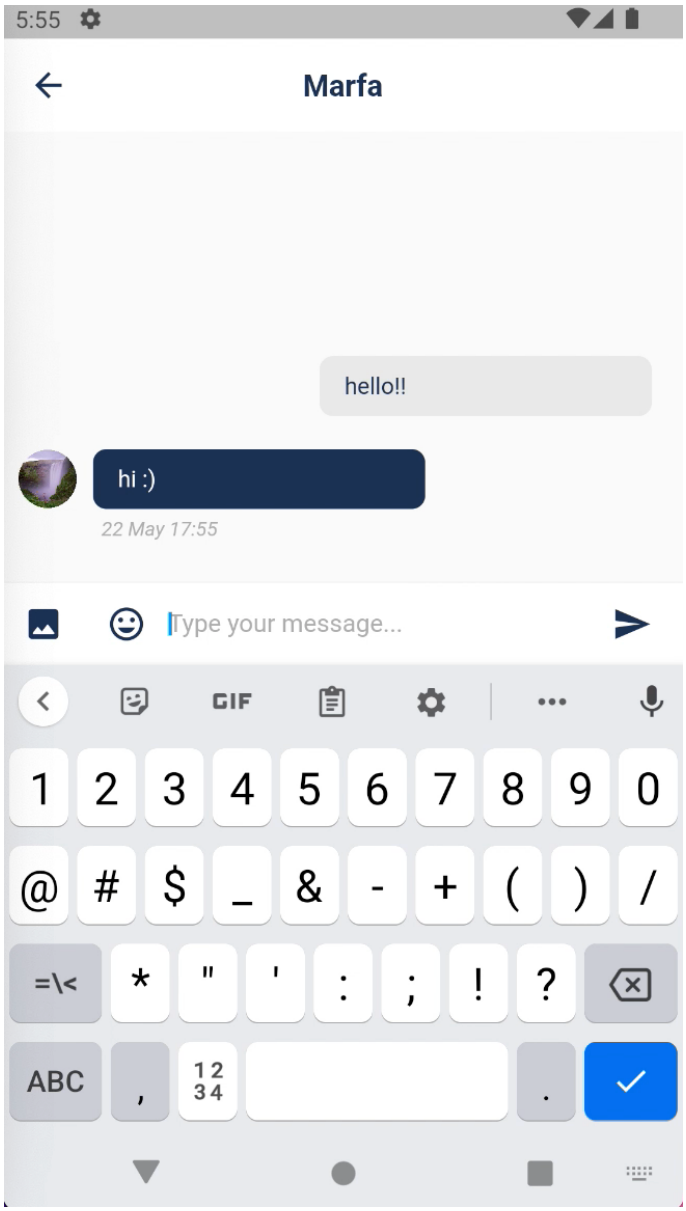

Рисунок 7.5 - Чат

ВИСНОВКИ

В результаті виконання даної роботи було розроблено програму «Чат». Розроблено зручний, приємний інтерфейс, який інтуїтивно зрозумілий і не вимагає додаткових знань чи навичок при роботі з ним. Програма містить окрему структуру, що розділяє бізнес логіку і реалізацію інтерфейсної частини.

Область застосування − це спілкування між бізнес-колегами. Користувачами програми є співробітники будь-якого рівня. Перед проектуванням було проведено аналіз аналогів та збір вимог з метою збільшення ефективності.

Дана програма дозволить бізнесу-власнику впровадити потрібний їм рівень конфеденційності та унеможливить витоку даних. Даний продукт створює безліч можливостей для розширення функціоналу. Всі можливі додаткові функції легко впроваджуються в майбутньому завдяки зрозумілій та простій архітектурі.

ЛІТЕРАТУРА

1. [Електронний ресурс] Технічна документація Flutter — https://flutter.dev/

2. [Електронний ресурс] Документація Flutter про його віджети —https://web.archive.org/web/20171213201309/https://flutter.io/widgets/material/

3. [Електронний ресурс] Technical overview Flutter https://web.archive.org/web/20171213201209/https://flutter.io/technical-overview/

4. Лайза Криспин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = agile Testing: a Practical Guide for Testers and agile Teams. — М. : «Вильямс», 2010. — 464 с.

5. [Електронний ресурс] Документація Telegram — https://telegram.org/apps

6. [Електронний ресурс] Документація Slack — https://api.slack.com/

7. [Електронний ресурс] Overview of the base image of Telegram https://www.nytimes.com/2014/12/03/technology/once-celebrated-in-russia-programmer-pavel-durov-chooses-exile.html

8. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев : ДиаСофт, 2001. — 544 с.

9. [Електронний ресурс] Документація Skype https://docs.microsoft.com/en-us/skypeforbusiness/

ДОДАТКИ

ДОДАТОК А
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
18.02.22

КРОСПЛАТФОРМЕННИЙ МОБІЛЬНИЙ ДОДАТОК “ЧАТ”

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01245-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКИН
18.02.22

Керівник розробки
Вадим АНДРЮЩЕНКО
18.02.22

Виконавець
Катерина КАЛІНЧЕНКО
18.02.22

Нормоконтролер
Олена КУРОП’ЯТНИК
18.02.22

2022

ЗАТВЕРДЖЕНО
44165850.01245-01-ЛЗ

Кросплатформенний мобільний додаток “Чат”

Технічне завдання

44165850.01245-01

Листів 13

2022

ЗМІСТ

ВВЕДЕННЯ

Програмний додаток «Чат», що розробляється призначений для обміну особистими повідомленнями.

Причини виникнення розробки ПЗ: аналоги існують, але замовнику потрібен його власний додаток.

Існує багато мессенджерів, але всі вони мають своїх власників, тому клієнт потребує свій власний додаток, що втілює базові функції.

Область застосувань: програмний продукт призначений для клієнтського використання аудиторією 12-99 років.

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є навчальний план для спеціальності 121 «Інженерія програмного забезпечення», затверджений ректором «Українського державного університету науки і технологій», від 30.06.2018 року.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт призначений для обміну особистими повідомленнями між користувачами.

Експлуатаційне призначення – за допомогою програмного продукту забезпечується обмін особистими повідомленнями між співробітниками однієї організації.

3. ВИМОГИ ДО ПРОГРАМИ

1. Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

Програма (мобільний додаток) повинна забезпечити можливість створення профілю, входу до аккаунту та відправки особистих повідомлень.

До неї належать такі основні сутності:

- користувачі;
- повідомлення;
- чати.

Вихідні дані:

- повідомлення (рис. 3.1).

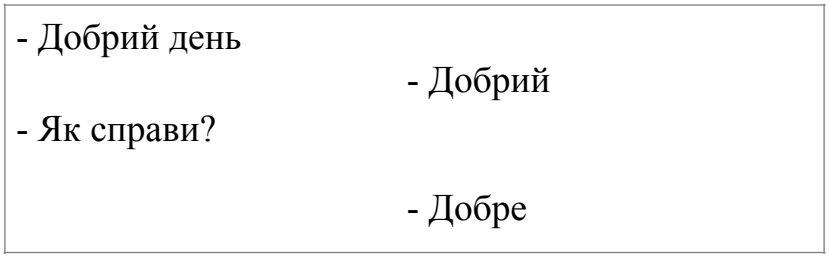

Рисунок. 3.1 - Приклад повідомлень

В програмі необхідно реалізувати можливість обміну особистими повідомленнями між користувачами.

Дані вводяться з клавіатури телефона.

Самі записи зберігаються у базі даних Firestore сервісу Firebase.

2. Вимоги до надійності

Вимоги до надійності наступні:

- для полів ведення необхідно забезпечити контроль ведених даних;
- при оновленні даних необхідно забезпечити показ відповідних повідомлень про стан роботи програми та повідомлення про результати виконання операцій.

3. Умови експлуатації

Програмний продукт повинен використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.31-99 (див. табл. 3.1).

Таблиця 3.1 - Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.1-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		оптимальна	оптимальна	оптимальна
Холодна	легка-1 а	22 - 24	40 - 60	0,1
	легка-1 б	21 - 23	40 - 60	0,1
Тепла	легка-1 а	23 - 25	40 - 60	0,1
	легка-1 б	22 - 24	40 - 60	0,2

Працювати з програмою може людина, що має навички роботи з телефоном та ознайомлена з керівництвом користувача.

4. Вимоги до складу та параметрів технічних засобів

Розроблюваний програмний продукт повинен використовуватись на мобільному телефоні, що мають доступ до інтернету та операційну систему або IOS, або Android.

5. Вимоги до інформаційної і програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем від Android 30 і новіші та IOS 14 і новіші.

6. Вимоги до маркування і упаковки

Упаковка продукту: код продукту зберігається на Github до якого замовник має доступ та на Firebase до якого замовник так само має доступ. Продукт має містити інформацію: назва програми, розробник, контактні дані. Приклад маркування приведений на рис. 3.2.

Чат

Калінченко Катерина
0473927748

Рисунок 3.2 - Маркування

4.7. Вимоги до транспортування і зберігання

Додаток має бути на Firebase, код продукту зберігається на Github. Транспортування мережею інтернет.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- специфікація;
- текст програми;
- опис програми.

Вся документація до програмного додатку повинна задовольняти вимоги до програмної документації.

5. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

В таблиці 5.1 приведені стадії та етапи розробки програмного продукту.

Таблиця 5.1 - Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.22 - 18.02.22
Робочий проект	Програмування та відладка програми.	19.02.22 - 01.05.22
	Тестування програми	02.05.22 - 24.05.22
	Розробка, узгодження і затвердження програмної документації.	25.05.22 - 12.06.22

6. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доцент Андрющенко В. О.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун—т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид—во Дніпропетр. нац. ун—ту залізн. трансп. ім. акад. В. Лазаряна, 2009. — 38 с

ЗАТВЕРДЖЕНО
44165850.01245–01 12–01 ЛЗ

Додаток Б

КРОСПЛАТФОРМЕННИЙ МОБІЛЬНИЙ ДОДАТОК ЧАТ

Текст програми

44165850.01245 01 12 01

Листів 45

2022

АНОТАЦІЯ

Документ 44165850.01245 01 12 01 «Кросплатформенний мобільний додаток Чат. Текст програми» входить до складу програмної документації на програму, що виконує роль мобільного додатку для комунікації.

У даному документі представлений текст програм. Програми написані на мові Dart з використанням фреймворку Flutter у програмному середовищі Android Studio.

ЗМІСТ

1. СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛЕЙ

Схема взаємодії програмних подулів зображена на рисунку 1.1

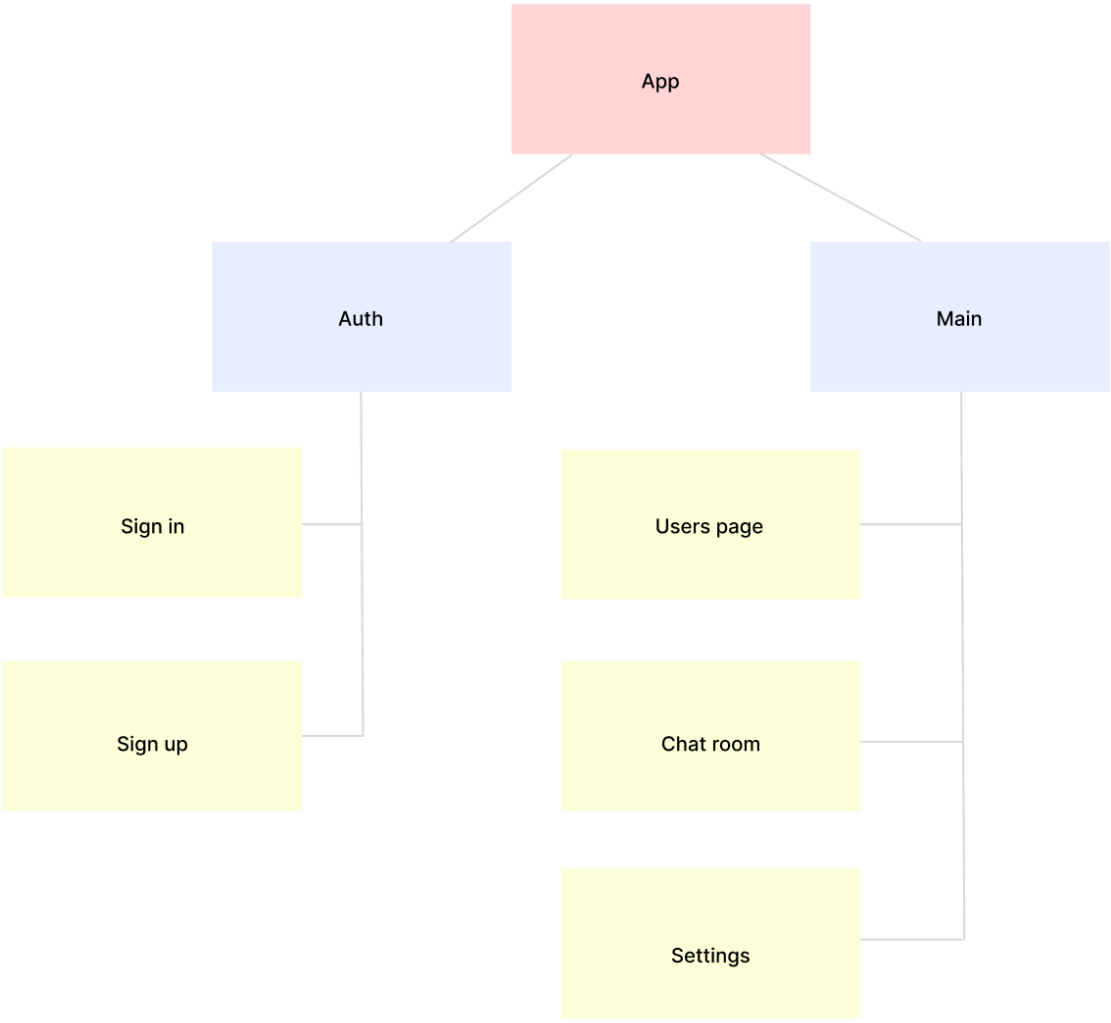

Рисунок 1.1 — Схема взаємодії модулів

2. ТЕКСТ ПРОГРАМИ

app_constants.dart

```
class AppConstants {

  static const appTitle = "Flutter Chat Demo";

  static const loginTitle = "Login";

  static const homeTitle = "Home";

  static const settingsTitle = "Settings";

  static const fullPhotoTitle = "Full Photo";

}
```

color_constants.dart

```
import 'dart:ui';


class ColorConstants {

  static const themeColor = Color(0xfff5a623);

  static const primaryColor = Color(0xff203152);

  static const greyColor = Color(0xffaeaeae);

  static const greyColor2 = Color(0xffE8E8E8);

}
```

firestore_constants.dart

```
class FirestoreConstants {

  static const pathUserCollection = "users";

  static const pathMessageCollection =
"messages";

  static const nickname = "nickname";

  static const aboutMe = "aboutMe";

  static const photoUrl = "photoUrl";

  static const id = "id";

  static const chattingWith = "chattingWith";

  static const idFrom = "idFrom";

  static const idTo = "idTo";

  static const timestamp = "timestamp";

  static const content = "content";

  static const type = "type";

}
```

message_model.dart

```
import 'package:chat_demo/constants/
firestore_constants.dart';
import 'package:cloud_firestore/
cloud_firestore.dart';


class MessageModel {

  String idFrom;

  String idTo;

  String timestamp;

  String content;

  int type;


  MessageModel({

    required this.idFrom,

    required this.idTo,

    required this.timestamp,

    required this.content,

    required this.type,

  });


  Map<String, dynamic> toJson() {

    return {

      FirestoreConstants.idFrom: idFrom,

      FirestoreConstants.idTo: idTo,

      FirestoreConstants.timestamp: timestamp,

      FirestoreConstants.content: content,

      FirestoreConstants.type: type,

    };

  }


  factory
MessageModel.fromDocument(DocumentSnapshot doc)
{

    String idFrom =
doc.get(FirestoreConstants.idFrom);

    String idTo =
doc.get(FirestoreConstants.idTo);
```

```
    String timestamp =
doc.get(FirestoreConstants.timestamp);

    String content =
doc.get(FirestoreConstants.content);
```



```
int type = doc.get(FirestoreConstants.type);

    return MessageModel(idFrom: idFrom, idTo:
idTo, timestamp: timestamp, content: content,
type: type);

  }

}
```

user_chat_model.dart

```
import 'package:chat_demo/constants/
firestore_constants.dart';

import 'package:cloud_firestore/
cloud_firestore.dart';


// ignore_for_file: empty_catches

class UserModel {

  String id;

  String photoUrl;

  String nickname;

  String aboutMe;


  UserModel({required this.id, required
this.photoUrl, required this.nickname, required
this.aboutMe});


  Map<String, String> toJson() {

    return {

      FirestoreConstants.nickname: nickname,

      FirestoreConstants.aboutMe: aboutMe,

      FirestoreConstants.photoUrl: photoUrl,

    };

  }


  factory
UserModel.fromDocument(DocumentSnapshot doc) {

    String aboutMe = "";

    String photoUrl = "";

    String nickname = "";

    try {

      aboutMe =
doc.get(FirestoreConstants.aboutMe);

    } catch (e) {}

    try {

      photoUrl =
doc.get(FirestoreConstants.photoUrl);

    } catch (e) {}

    try {

      nickname =
doc.get(FirestoreConstants.nickname);

    } catch (e) {}

    return UserModel(

      id: doc.id,

      photoUrl: photoUrl,

      nickname: nickname,

      aboutMe: aboutMe,

    );

  }

}
```

chat_page.dart

```
import 'dart:async';

import 'dart:io';


import 'package:chat_demo/constants/
color_constants.dart';

import 'package:chat_demo/constants/
firestore_constants.dart';

import 'package:chat_demo/models/
message_model.dart';

import 'package:chat_demo/pages/
full_photo_page.dart';

import 'package:chat_demo/pages/
sign_in_page.dart';

import 'package:chat_demo/providers/
auth_provider.dart';
```

```
import 'package:chat_demo/providers/
chat_provider.dart';

import 'package:chat_demo/widgets/
loading_view.dart';

import 'package:cloud_firestore/
cloud_firestore.dart';

import 'package:firebase_storage/
firebase_storage.dart';

import 'package:flutter/material.dart';

import 'package:fluttertoast/fluttertoast.dart';

import 'package:image_picker/image_picker.dart';

import 'package:intl/intl.dart';

import 'package:provider/provider.dart';


class ChatPage extends StatefulWidget {

  const ChatPage({Key? key, required
this.arguments}) : super(key: key);


  final ChatPageArguments arguments;


  @override

  ChatPageState createState() =>
ChatPageState();

}


class ChatPageState extends State<ChatPage> {

  late String currentUserId;


  List<QueryDocumentSnapshot> listMessage = [];

  int _limit = 20;

  final int _limitIncrement = 20;

  String groupChatId = "";


  File? imageFile;

  bool isLoading = false;

  bool isShowSticker = false;

  String imageUrl = "";


  final TextEditingController
textEditingController = TextEditingController();

  final ScrollController listScrollController =
ScrollController();

  final FocusNode focusNode = FocusNode();


  late ChatProvider chatProvider;

  late AuthProvider authProvider;


  @override

  void initState() {

    super.initState();

    chatProvider = context.read<ChatProvider>();

    authProvider = context.read<AuthProvider>();


    focusNode.addListener(onFocusChange);


listScrollController.addListener(_scrollListener
);

    readLocal();

  }


  _scrollListener() {

    if (listScrollController.offset >=


listScrollController.position.maxScrollExtent &&

        !
listScrollController.position.outOfRange &&

        _limit <= listMessage.length) {

      setState(() {

        _limit += _limitIncrement;

      });

    }

  }


  void onFocusChange() {
```

```
    if (focusNode.hasFocus) {
      // Hide sticker when keyboard appear
      setState(() {
        isShowSticker = false;
      });
    }
  }

  void readLocal() {
    if
(authProvider.getUserFirebaseId()?.isNotEmpty ==
true) {
      currentUserId =
authProvider.getUserFirebaseId()!;
    } else {
      Navigator.of(context).pushAndRemoveUntil(
        MaterialPageRoute(builder: (context) =>
const SignInPage()),
        (Route<dynamic> route) => false,
      );
    }
    String peerId = widget.arguments.peerId;
    if (currentUserId.compareTo(peerId) > 0) {
      groupChatId = '$currentUserId-$peerId';
    } else {
      groupChatId = '$peerId-$currentUserId';
    }

    chatProvider.updateDataFirestore(
      FirestoreConstants.pathUserCollection,
      currentUserId,
      {FirestoreConstants.chattingWith: peerId},
    );
  }

  Future getImage() async {
    ImagePicker imagePicker = ImagePicker();
    PickedFile? pickedFile;

    pickedFile = await
imagePicker.getImage(source:
ImageSource.gallery);
    if (pickedFile != null) {
      imageFile = File(pickedFile.path);
      if (imageFile != null) {
        setState(() {
          isLoading = true;
        });
        uploadFile();
      }
    }
  }
  void getSticker() {
    // Hide keyboard when sticker appear
    focusNode.unfocus();
    setState(() {
      isShowSticker = !isShowSticker;
    });
  }

  Future uploadFile() async {
    String fileName =
DateTime.now().millisecondsSinceEpoch.toString()
;
    UploadTask uploadTask =
chatProvider.uploadFile(imageFile!, fileName);
    try {
      TaskSnapshot snapshot = await uploadTask;
      imageUrl = await
snapshot.ref.getDownloadURL();
      setState(() {
        isLoading = false;
        onSendMessage(imageUrl,
TypeMessage.image);
```

```
      });
    } on FirebaseException catch (e) {
      setState(() {
        isLoading = false;
      });
      Fluttertoast.showToast(msg: e.message ??
e.toString());
    }
  }

  void onSendMessage(String content, int type) {
    if (content.trim().isNotEmpty) {
      textEditingController.clear();
      chatProvider.sendMessage(
          content, type, groupChatId,
currentUserId, widget.arguments.peerId);
      listScrollController.animateTo(0,
          duration: const Duration(milliseconds:
300), curve: Curves.easeOut);
    } else {
      Fluttertoast.showToast(
          msg: 'Nothing to send',
backgroundColor: ColorConstants.greyColor);
    }
  }

  Widget buildItem(int index, DocumentSnapshot?
document) {
    if (document != null) {
      MessageModel messageChat =
MessageModel.fromDocument(document);
      if (messageChat.idFrom == currentUserId) {
        // Right (my message)
        return Row(
          children: <Widget>[
            messageChat.type == TypeMessage.text
                // Text
                ? Container(
                    child: Text(
                      messageChat.content,
                      style:
                          const TextStyle(color:
ColorConstants.primaryColor),
                    ),
                    padding: const
EdgeInsets.fromLTRB(15, 10, 15, 10),
                    width: 200,
                    decoration: BoxDecoration(
                        color:
ColorConstants.greyColor2,
                        borderRadius:
BorderRadius.circular(8)),
                    margin: EdgeInsets.only(
                        bottom:
isLastMessageRight(index) ? 20 : 10, right: 10),
                  )
                : messageChat.type ==
TypeMessage.image
                    // Image
                    ? Container(
                        child: OutlinedButton(
                          child: Material(
                            child:
Image.network(
messageChat.content,
                              loadingBuilder:
(BuildContext context,
                                  Widget child,
ImageChunkEvent? loadingProgress) {
                                if
(loadingProgress == null) return child;
                                return
Container(
                                  decoration:
const BoxDecoration(
```

```
                                color:
ColorConstants.greyColor2,
borderRadius: BorderRadius.all(
Radius.circular(8),
                                ),
                              ),
                              width: 200,
                              height: 200,
                              child: Center(
                                child:
CircularProgressIndicator(
                                  color:
ColorConstants.themeColor,
                                  value:
loadingProgress.expectedTotalBytes !=
null
                                      ?
loadingProgress
.cumulativeBytesLoaded /
loadingProgress
.expectedTotalBytes!
                                      :
null,
                                ),
                              ),
                            );
                          },
                          errorBuilder:
(context, object, stackTrace) {
                            return Material(
                              child:
Image.asset(
                                'assets/
images/img_not_available.jpeg',
                                width: 200,
                                height: 200,
                                fit:
BoxFit.cover,
                              ),
                              borderRadius:
const BorderRadius.all(
Radius.circular(8),
                              ),
                              clipBehavior:
Clip.hardEdge,
                            );
                          },
                          width: 200,
                          height: 200,
                          fit: BoxFit.cover,
                        ),
                        borderRadius:
                            const
BorderRadius.all(Radius.circular(8)),
                        clipBehavior:
Clip.hardEdge,
                      ),
                      onPressed: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder:
(context) => FullPhotoPage(
                              url:
messageChat.content,
                            ),
                          ),
                        );
                      },
                      style: ButtonStyle(
```

```
                            padding:
MaterialStateProperty.all<EdgeInsets>(
                                const
EdgeInsets.all(0))),
                        ),
                        margin: EdgeInsets.only(
                            bottom:
isLastMessageRight(index) ? 20 : 10,
                            right: 10),
                      )
                    // Sticker
                    : Container(
                        child: Image.asset(
                          'assets/images/$
{messageChat.content}.gif',
                          width: 100,
                          height: 100,
                          fit: BoxFit.cover,
                        ),
                        margin: EdgeInsets.only(
                            bottom:
isLastMessageRight(index) ? 20 : 10,
                            right: 10),
                      ),
          ],
          mainAxisAlignment:
MainAxisAlignment.end,
        );
      } else {
        // Left (peer message)
        return Container(
          child: Column(
            children: <Widget>[
              Row(
                children: <Widget>[
                  isLastMessageLeft(index)
                      ? Material(
                          child: Image.network(
widget.arguments.peerAvatar,
                            loadingBuilder:
(BuildContext context, Widget child,
                                ImageChunkEvent?
loadingProgress) {
                              if
(loadingProgress == null) return child;
                              return Center(
                                child:
CircularProgressIndicator(
                                  color:
ColorConstants.themeColor,
                                  value:
loadingProgress.expectedTotalBytes !=
                                          null
                                      ?
loadingProgress.cumulativeBytesLoaded /
loadingProgress.expectedTotalBytes!
                                      : null,
                                ),
                              );
                            },
                            errorBuilder:
(context, object, stackTrace) {
                              return const Icon(
Icons.account_circle,
                                size: 35,
                                color:
ColorConstants.greyColor,
                              );
                            },
                            width: 35,
                            height: 35,
                            fit: BoxFit.cover,
                          ),
```

```
                        borderRadius: const
BorderRadius.all(
                          Radius.circular(18),
                        ),
                        clipBehavior:
Clip.hardEdge,
                      )
                    : Container(width: 35),
                messageChat.type ==
TypeMessage.text
                    ? Container(
                        child: Text(
                          messageChat.content,
                          style: const
TextStyle(color: Colors.white),
                        ),
                        padding: const
EdgeInsets.fromLTRB(15, 10, 15, 10),
                        width: 200,
                        decoration:
BoxDecoration(
                            color:
ColorConstants.primaryColor,
                            borderRadius:
BorderRadius.circular(8)),
                        margin: const
EdgeInsets.only(left: 10),
                      )
                    : messageChat.type ==
TypeMessage.image
                        ? Container(
                            child: TextButton(
                              child: Material(
                                child:
Image.network(
messageChat.content,
loadingBuilder: (BuildContext context,
Widget
child,
ImageChunkEvent? loadingProgress) {
                                    if
(loadingProgress == null) return child;
                                    return
Container(
decoration: const BoxDecoration(
                                        color:
ColorConstants.greyColor2,
borderRadius: BorderRadius.all(
Radius.circular(8),
                                        ),
                                    ),
                                    width:
200,
                                    height:
200,
                                    child:
Center(
                                      child:
CircularProgressIndicator(
color: ColorConstants.themeColor,
value: loadingProgress
.expectedTotalBytes !=
null
? loadingProgress
.cumulativeBytesLoaded /
loadingProgress
.expectedTotalBytes!
```

```
: null,
                                    ),
                                ),
                            );
                        },
                        errorBuilder:
(context, object, stackTrace) =>
Material(
                                child:
Image.asset(
                                  'assets/
images/img_not_available.jpeg',
                                  width:
200,
                                  height:
200,
                                  fit:
BoxFit.cover,
                                ),
borderRadius: const BorderRadius.all(
Radius.circular(8),
                                ),
clipBehavior: Clip.hardEdge,
                            ),
                            width: 200,
                            height: 200,
                            fit:
BoxFit.cover,
                        ),
                        borderRadius:
const BorderRadius.all(
Radius.circular(8)),
                        clipBehavior:
Clip.hardEdge,
                    ),
```

44165850.01245 01 12 01
20

```
                    onPressed: () {
Navigator.push(
                        context,
MaterialPageRoute(
                          builder:
(context) => FullPhotoPage(
                            url:
messageChat.content),
                        ),
                      );
                    },
                    style:
ButtonStyle(
                      padding:
MaterialStateProperty.all<EdgeInsets>(
const EdgeInsets.all(0))),
                    ),
                    margin: const
EdgeInsets.only(left: 10),
                  )
                : Container(
                    padding: const
EdgeInsets.only(left: 10),
                    child:
Image.asset(
                      'assets/images/$
{messageChat.content}.gif',
                      width: 100,
                      height: 100,
                      fit:
BoxFit.cover,
```

```
                                ),
                                margin:
EdgeInsets.only(
                                    bottom:
isLastMessageRight(index) ? 20 : 10,
                                    right: 10),
                            ),
                  ],
                ),

                // Time
                isLastMessageLeft(index)
                    ? Container(
                        child: Text(
                          DateFormat('dd MMM
kk:mm').format(
DateTime.fromMillisecondsSinceEpoch(
int.parse(messageChat.timestamp))),
                          style: const TextStyle(
                              color:
ColorConstants.greyColor,
                              fontSize: 12,
                              fontStyle:
FontStyle.italic),
                        ),
                        margin:
                            const
EdgeInsets.only(left: 50, top: 5, bottom: 5),
                      )
                    : const SizedBox.shrink()
              ],
              crossAxisAlignment:
CrossAxisAlignment.start,
            ),
            margin: const EdgeInsets.only(bottom:
10),
          );
    }
  } else {
    return const SizedBox.shrink();
  }
}

bool isLastMessageLeft(int index) {
  if ((index > 0 &&
          listMessage[index -
1].get(FirestoreConstants.idFrom) ==
              currentUserId) ||
      index == 0) {
    return true;
  } else {
    return false;
  }
}

bool isLastMessageRight(int index) {
  if ((index > 0 &&
          listMessage[index -
1].get(FirestoreConstants.idFrom) !=
              currentUserId) ||
      index == 0) {
    return true;
  } else {
    return false;
  }
}

Future<bool> onBackPress() {
  if (isShowSticker) {
    setState(() {
      isShowSticker = false;
    });
  } else {
```

```
      chatProvider.updateDataFirestore(
        FirestoreConstants.pathUserCollection,
        currentUserId,
        {FirestoreConstants.chattingWith: null},
      );
      Navigator.pop(context);
    }

    return Future.value(false);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          widget.arguments.peerNickname,
          style: const TextStyle(color:
ColorConstants.primaryColor),
        ),
        centerTitle: true,
      ),
      body: SafeArea(
        child: WillPopScope(
          child: Stack(
            children: <Widget>[
              Column(
                children: <Widget>[
                  // List of messages
                  buildListMessage(),

                  // Sticker
                  isShowSticker ? buildSticker()
: const SizedBox.shrink(),

                  // Input content
                  buildInput(),
                ],
              ),

              // Loading
              buildLoading()
            ],
          ),
          onWillPop: onBackPress,
        ),
      ),
    );
  }

  Widget buildSticker() {
    return Expanded(
      child: Container(
        child: Column(
          children: <Widget>[
            Row(
              children: <Widget>[
                TextButton(
                  onPressed: () =>
onSendMessage('img1', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img1.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img2', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img2.gif',
```

```
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img3', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img3.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                )
              ],
              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            ),
            Row(
              children: <Widget>[
                TextButton(
                  onPressed: () =>
onSendMessage('img4', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img4.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img5', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img5.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img6', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img6.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                )
              ],
              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            ),
            Row(
              children: <Widget>[
                TextButton(
                  onPressed: () =>
onSendMessage('img7', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img7.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img8', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img8.gif',
```

```
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                ),
                TextButton(
                  onPressed: () =>
onSendMessage('img9', TypeMessage.sticker),
                  child: Image.asset(
                    'assets/images/img9.gif',
                    width: 90,
                    height: 90,
                    fit: BoxFit.cover,
                  ),
                )
              ],
              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            )
          ],
          mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
        ),
        decoration: const BoxDecoration(
          border: Border(
            top: BorderSide(
              color: ColorConstants.greyColor2,
              width: 0.5,
            ),
          ),
          color: Colors.white,
        ),
        padding: const EdgeInsets.all(5),
        height: 300,
      ),
    );
  }

  Widget buildLoading() {
    return Positioned(
      child: isLoading ? const LoadingView() :
const SizedBox.shrink(),
    );
  }

  Widget buildInput() {
    return Container(
      child: Row(
        children: <Widget>[
          // Button send image
          Material(
            child: Container(
              margin: const
EdgeInsets.symmetric(horizontal: 1),
              child: IconButton(
                icon: const
Icon(Icons.image_rounded),
                onPressed: getImage,
                color:
ColorConstants.primaryColor,
              ),
            ),
            color: Colors.white,
          ),
          Material(
            child: Container(
              margin: const
EdgeInsets.symmetric(horizontal: 1),
              child: IconButton(
                icon: const
Icon(Icons.insert_emoticon),
                onPressed: getSticker,
                color:
ColorConstants.primaryColor,
```

```
            ),
          ),
          color: Colors.white,
        ),

        // Edit text
        Flexible(
          child: TextField(
            onSubmitted: (value) {
onSendMessage(textEditingController.text,
TypeMessage.text);
            },
            style: const TextStyle(
                color:
ColorConstants.primaryColor, fontSize: 15),
            controller: textEditingController,
            decoration: const
InputDecoration.collapsed(
              hintText: 'Type your
message...',
              hintStyle: TextStyle(color:
ColorConstants.greyColor),
            ),
            focusNode: focusNode,
          ),
        ),

        // Button send message
        Material(
          child: Container(
            margin: const
EdgeInsets.symmetric(horizontal: 8),
            child: IconButton(
              icon: const Icon(Icons.send),
              onPressed: () =>
onSendMessage(textEditingController.text,
TypeMessage.text),
                color:
ColorConstants.primaryColor,
              ),
            ),
            color: Colors.white,
          ),
        ],
      ),
      width: double.infinity,
      height: 50,
      decoration: const BoxDecoration(
          border: Border(
            top: BorderSide(color:
ColorConstants.greyColor2, width: 0.5)),
          color: Colors.white),
    );
  }

  Widget buildListMessage() {
    return Flexible(
      child: groupChatId.isNotEmpty
          ? StreamBuilder<QuerySnapshot>(
              stream:
chatProvider.getChatStream(groupChatId, _limit),
              builder: (BuildContext context,
                  AsyncSnapshot<QuerySnapshot>
snapshot) {
                if (snapshot.hasData) {
                  listMessage =
snapshot.data!.docs;
                  if (listMessage.isNotEmpty) {
                    // 1653 21 7079433 //1653 15
1653257
                    return ListView.builder(
```

```
                    padding: const
EdgeInsets.all(10),
                    itemBuilder: (context,
index) =>
                        buildItem(index,
snapshot.data?.docs[index]),
                    itemCount:
snapshot.data?.docs.length,
                    reverse: true,
                    controller:
listScrollController,
                  );
                } else {
                  return const Center(child:
Text("No message here yet..."));
                }
              } else {
                return const Center(
                  child:
CircularProgressIndicator(
                    color:
ColorConstants.themeColor,
                  ),
                );
              }
            },
          )
        : const Center(
            child: CircularProgressIndicator(
              color:
ColorConstants.themeColor,
            ),
          ),
    );
  }
}

class ChatPageArguments {
  final String peerId;
  final String peerAvatar;
  final String peerNickname;

  ChatPageArguments(
      {required this.peerId,
      required this.peerAvatar,
      required this.peerNickname});
}
```

full_photo.dart

```
import 'package:chat_demo/constants/
app_constants.dart';
import 'package:chat_demo/constants/
color_constants.dart';
import 'package:flutter/material.dart';
import 'package:photo_view/photo_view.dart';

class FullPhotoPage extends StatelessWidget {
  final String url;

  const FullPhotoPage({Key? key, required
this.url}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          AppConstants.fullPhotoTitle,
          style: TextStyle(color:
ColorConstants.primaryColor),
        ),
        centerTitle: true,
      ),
      body: PhotoView(
        imageProvider: NetworkImage(url),
      ),
```

```
    );
  }
}
```

home_page.dart

```
import 'dart:async';
import 'dart:io';

import 'package:chat_demo/constants/
color_constants.dart';
import 'package:chat_demo/constants/
firestore_constants.dart';
import 'package:chat_demo/models/
popup_choices_model.dart';
import 'package:chat_demo/models/
user_chat_model.dart';
import 'package:chat_demo/pages/chat_page.dart';
import 'package:chat_demo/pages/
settings_page.dart';
import 'package:chat_demo/pages/
sign_in_page.dart';
import 'package:chat_demo/providers/
auth_provider.dart';
import 'package:chat_demo/providers/
home_provider.dart';
import 'package:chat_demo/utils/debouncer.dart';
import 'package:chat_demo/utils/utilities.dart';
import 'package:chat_demo/widgets/
loading_view.dart';
import 'package:cloud_firestore/
cloud_firestore.dart';
import 'package:firebase_messaging/
firebase_messaging.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import '../constants/app_constants.dart';
import 'package:flutter_local_notifications/
flutter_local_notifications.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:provider/provider.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State createState() => HomePageState();
}

class HomePageState extends State<HomePage> {
  HomePageState({Key? key});

  final FirebaseMessaging firebaseMessaging =
FirebaseMessaging.instance;
  final FlutterLocalNotificationsPlugin
flutterLocalNotificationsPlugin =
FlutterLocalNotificationsPlugin();
  final ScrollController listScrollController =
ScrollController();

  int _limit = 20;
  final int _limitIncrement = 20;
  String _textSearch = "";
  bool isLoading = false;

  late AuthProvider authProvider;
  late String currentUserId;
  late HomeProvider homeProvider;
  Debouncer searchDebouncer =
Debouncer(milliseconds: 300);
  StreamController<bool> btnClearController =
StreamController<bool>();
  TextEditingController searchBarTec =
TextEditingController();

  List<PopupChoicesModel> choices =
<PopupChoicesModel>[
```

```
    PopupChoicesModel(title: 'Settings', icon:
Icons.settings),
    PopupChoicesModel(title: 'Log out', icon:
Icons.exit_to_app),
  ];

  @override
  void initState() {
    super.initState();
    authProvider = context.read<AuthProvider>();
    homeProvider = context.read<HomeProvider>();

    if
(authProvider.getUserFirebaseId()?.isNotEmpty ==
true) {
      currentUserId =
authProvider.getUserFirebaseId()!;
    } else {
      Navigator.of(context).pushAndRemoveUntil(
        MaterialPageRoute(builder: (context) =>
const SignInPage()),
        (Route<dynamic> route) => false,
      );
    }
    registerNotification();
    configLocalNotification();

listScrollController.addListener(scrollListener)
;
  }

  @override
  void dispose() {
    super.dispose();
    btnClearController.close();
  }

  void registerNotification() {
    firebaseMessaging.requestPermission();

FirebaseMessaging.onMessage.listen((RemoteMessag
e message) {
      if (kDebugMode) {
        print('onMessage: $message');
      }
      if (message.notification != null) {
        showNotification(message.notification!);
      }
      return;
    });

    firebaseMessaging.getToken().then((token) {
      if (kDebugMode) {
        print('push token: $token');
      }
      if (token != null) {

homeProvider.updateDataFirestore(FirestoreConsta
nts.pathUserCollection, currentUserId,
{'pushToken': token});
      }
    }).catchError((err) {
      Fluttertoast.showToast(msg:
err.message.toString());
    });
  }

  void configLocalNotification() {
    AndroidInitializationSettings
initializationSettingsAndroid = const
AndroidInitializationSettings('app_icon');
    IOSInitializationSettings
initializationSettingsIOS = const
IOSInitializationSettings();
    InitializationSettings
initializationSettings =
```

```
        InitializationSettings(android:
initializationSettingsAndroid, iOS:
initializationSettingsIOS);

flutterLocalNotificationsPlugin.initialize(initi
alizationSettings);
  }

  void scrollListener() {
    if (listScrollController.offset >=
listScrollController.position.maxScrollExtent &&
        !
listScrollController.position.outOfRange) {
      setState(() {
        _limit += _limitIncrement;
      });
    }
  }

  void onItemMenuPress(PopupChoicesModel choice)
{
    if (choice.title == 'Log out') {
      handleSignOut();
    } else {
      Navigator.push(context,
MaterialPageRoute(builder: (context) => const
SettingsPage()));
    }
  }

  void showNotification(RemoteNotification
remoteNotification) async {
    AndroidNotificationDetails
androidPlatformChannelSpecifics =
AndroidNotificationDetails(
      Platform.isAndroid ?
'com.example.chat_demo' :
'com.example.chatDemo',
      'Chat',
      playSound: true,
      enableVibration: true,
      importance: Importance.max,
      priority: Priority.high,
    );
    IOSNotificationDetails
iOSPlatformChannelSpecifics = const
IOSNotificationDetails();
    NotificationDetails platformChannelSpecifics
=
        NotificationDetails(android:
androidPlatformChannelSpecifics, iOS:
iOSPlatformChannelSpecifics);

    if (kDebugMode) {
      print(remoteNotification);
    }

    await flutterLocalNotificationsPlugin.show(
      0,
      remoteNotification.title,
      remoteNotification.body,
      platformChannelSpecifics,
      payload: null,
    );
  }

  Future<bool> onBackPress() {
    openDialog();
    return Future.value(false);
  }

  Future<void> openDialog() async {
    switch (await showDialog(
        context: context,
        builder: (BuildContext context) {
          return SimpleDialog(
            clipBehavior: Clip.hardEdge,
```

```
            shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),

            contentPadding: EdgeInsets.zero,

            children: <Widget>[

              Container(

                color:
ColorConstants.themeColor,

                padding: const
EdgeInsets.only(bottom: 10, top: 10),

                child: Column(

                  mainAxisSize:
MainAxisSize.min,

                  children: <Widget>[

                    Container(

                      child: const Icon(

                        Icons.exit_to_app,

                        size: 30,

                        color: Colors.white,

                      ),

                      margin: const
EdgeInsets.only(bottom: 10),

                    ),

                    const Text(

                      'Exit app',

                      style: TextStyle(color:
Colors.white, fontSize: 18, fontWeight:
FontWeight.bold),

                    ),

                    const Text(

                      'Are you sure to exit
app?',

                      style: TextStyle(color:
Colors.white70, fontSize: 14),

                    ),

                  ],

                ),

              ),

              SimpleDialogOption(

                onPressed: () {

                  Navigator.pop(context, 0);

                },

                child: Row(

                  children: <Widget>[

                    Container(

                      child: const Icon(

                        Icons.cancel,

                        color:
ColorConstants.primaryColor,

                      ),

                      margin: const
EdgeInsets.only(right: 10),

                    ),

                    const Text(

                      'Cancel',

                      style: TextStyle(color:
ColorConstants.primaryColor, fontWeight:
FontWeight.bold),

                    )

                  ],

                ),

              ),

              SimpleDialogOption(

                onPressed: () {

                  Navigator.pop(context, 1);

                },

                child: Row(

                  children: <Widget>[

                    Container(

                      child: const Icon(

                        Icons.check_circle,

                        color:
ColorConstants.primaryColor,

                      ),

                      margin: const
EdgeInsets.only(right: 10),
```

```
                    ),
                    const Text(
                      'Yes',
                      style: TextStyle(color:
ColorConstants.primaryColor, fontWeight:
FontWeight.bold),
                    )
                  ],
                ),
              ),
            ],
          );
        })) {
      case 0:
        break;
      case 1:
        exit(0);
    }
  }

  Future<void> handleSignOut() async {
    authProvider.handleSignOut();
    Navigator.of(context).pushAndRemoveUntil(
      MaterialPageRoute(builder: (context) =>
const SignInPage()),
      (Route<dynamic> route) => false,
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          AppConstants.homeTitle,
          style: TextStyle(color:
ColorConstants.primaryColor),
        ),
        centerTitle: true,
        actions: <Widget>[buildPopupMenu()],
      ),
      body: WillPopScope(
        child: Stack(
          children: <Widget>[
            // List
            Column(
              children: [
                buildSearchBar(),
                Expanded(
                  child:
StreamBuilder<QuerySnapshot>(
                    stream:
homeProvider.getStreamFireStore(FirestoreConstan
ts.pathUserCollection, _limit, _textSearch),
                    builder: (BuildContext
context, AsyncSnapshot<QuerySnapshot> snapshot)
{
                      if (snapshot.hasData) {
                        if
((snapshot.data?.docs.length ?? 0) > 0) {
                          return
ListView.builder(
                            padding: const
EdgeInsets.all(10),
                            itemBuilder:
(context, index) => buildItem(context,
snapshot.data?.docs[index]),
                            itemCount:
snapshot.data?.docs.length,
                            controller:
listScrollController,
                          );
                        } else {
                          return const Center(
```

```
                            child: Text("No
users"),
                          );
                        }
                      } else {
                        return const Center(
                          child:
CircularProgressIndicator(
                            color:
ColorConstants.themeColor,
                          ),
                        );
                      }
                    },
                  ),
                ),
              ],
            ),

            // Loading
            Positioned(
              child: isLoading ? const
LoadingView() : const SizedBox.shrink(),
            )
          ],
        ),
        onWillPop: onBackPress,
      ),
    );

  }

  Widget buildSearchBar() {
    return Container(
      height: 40,
      child: Row(
        crossAxisAlignment:
CrossAxisAlignment.center,
        children: [
          const Icon(Icons.search, color:
ColorConstants.greyColor, size: 20),
          const SizedBox(width: 5),
          Expanded(
            child: TextFormField(
              textInputAction:
TextInputAction.search,
              controller: searchBarTec,
              onChanged: (value) {
                searchDebouncer.run(() {
                  if (value.isNotEmpty) {

btnClearController.add(true);
                    setState(() {
                      _textSearch = value;
                    });
                  } else {

btnClearController.add(false);
                    setState(() {
                      _textSearch = "";
                    });
                  }
                });
              },
              decoration: const
InputDecoration.collapsed(
                hintText: 'Search nickname (you
have to type exactly string)',
                hintStyle: TextStyle(fontSize:
13, color: ColorConstants.greyColor),
              ),

              style: const TextStyle(fontSize:
13),
            ),
```

```
          ),
          StreamBuilder<bool>(
              stream: btnClearController.stream,
              builder: (context, snapshot) {
                return snapshot.data == true
                    ? GestureDetector(
                        onTap: () {
                          searchBarTec.clear();

btnClearController.add(false);
                          setState(() {
                            _textSearch = "";
                          });
                        },
                        child: const
Icon(Icons.clear_rounded, color:
ColorConstants.greyColor, size: 20))
                    : const SizedBox.shrink();
              }),
        ],
      ),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(16),
        color: ColorConstants.greyColor2,
      ),
      padding: const EdgeInsets.fromLTRB(16, 8,
16, 8),
      margin: const EdgeInsets.fromLTRB(16, 8,
16, 8),
    );
  }


  Widget buildPopupMenu() {
    return PopupMenuButton<PopupChoicesModel>(
      onSelected: onItemMenuPress,

      itemBuilder: (BuildContext context) {
        return choices.map((PopupChoicesModel
choice) {
          return
PopupMenuItem<PopupChoicesModel>(
              value: choice,
              child: Row(
                children: <Widget>[
                  Icon(
                    choice.icon,
                    color:
ColorConstants.primaryColor,
                  ),
                  Container(
                    width: 10,
                  ),
                  Text(
                    choice.title,
                    style: const
TextStyle(color: ColorConstants.primaryColor),
                  ),
                ],
              ));
        }).toList();
      },
    );
  }


  Widget buildItem(BuildContext context,
DocumentSnapshot? document) {
    if (document != null) {
      UserModel userChat =
UserModel.fromDocument(document);
      if (userChat.id == currentUserId) {
        return const SizedBox.shrink();
      } else {
        return Container(
          child: TextButton(
```

```
            child: Row(
              children: <Widget>[
                Material(
                  child:
userChat.photoUrl.isNotEmpty
                      ? Image.network(
                          userChat.photoUrl,
                          fit: BoxFit.cover,
                          width: 50,
                          height: 50,
                          loadingBuilder:
(BuildContext context, Widget child,
ImageChunkEvent? loadingProgress) {
                            if (loadingProgress
== null) return child;
                            return SizedBox(
                              width: 50,
                              height: 50,
                              child: Center(
                                child:
CircularProgressIndicator(
                                  color:
ColorConstants.themeColor,
                                  value:
loadingProgress.expectedTotalBytes != null
                                      ?
loadingProgress.cumulativeBytesLoaded /
loadingProgress.expectedTotalBytes!
                                      : null,
                                ),
                              ),
                            );
                          },
                          errorBuilder:
(context, object, stackTrace) {
                            return const Icon(
Icons.account_circle,
                              size: 50,
                              color:
ColorConstants.greyColor,
                            );
                          },
                        )
                      : const Icon(
                          Icons.account_circle,
                          size: 50,
                          color:
ColorConstants.greyColor,
                        ),
                  borderRadius: const
BorderRadius.all(Radius.circular(25)),
                  clipBehavior: Clip.hardEdge,
                ),
                Flexible(
                  child: Container(
                    child: Column(
                      children: <Widget>[
                        Container(
                          child: Text(
                            'Nickname: $
{userChat.nickname}',
                            maxLines: 1,
                            style: const
TextStyle(color: ColorConstants.primaryColor),
                          ),
                          alignment:
Alignment.centerLeft,
                          margin: const
EdgeInsets.fromLTRB(10, 0, 0, 5),
                        ),
                        Container(
                          child: Text(
                            'About me: $
{userChat.aboutMe}',
                            maxLines: 1,
```

```
                            style: const
TextStyle(color: ColorConstants.primaryColor),
                          ),
                          alignment:
Alignment.centerLeft,
                          margin: const
EdgeInsets.fromLTRB(10, 0, 0, 0),
                        )
                      ],
                    ),
                    margin: const
EdgeInsets.only(left: 20),
                  ),
                ),
              ],
            ),
            onPressed: () {
              if (Utilities.isKeyboardShowing())
{

Utilities.closeKeyboard(context);
              }
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) =>
ChatPage(
                    arguments:
ChatPageArguments(
                      peerId: userChat.id,
                      peerAvatar:
userChat.photoUrl,
                      peerNickname:
userChat.nickname,
                    ),
                  ),
                ),
              );
            },
            style: ButtonStyle(
              backgroundColor:
MaterialStateProperty.all<Color>(ColorConstants.
greyColor2),
              shape:
MaterialStateProperty.all<OutlinedBorder>(
                const RoundedRectangleBorder(
                  borderRadius:
BorderRadius.all(Radius.circular(10)),
                ),
              ),
            ),
          ),
          margin: const EdgeInsets.only(bottom:
10, left: 5, right: 5),
        );
      }
    } else {
      return const SizedBox.shrink();
    }
  }
}
```

settings_page.dart

```
import 'dart:async';
import 'dart:io';

import 'package:chat_demo/constants/
color_constants.dart';
import 'package:chat_demo/constants/
firestore_constants.dart';
import 'package:chat_demo/models/
user_chat_model.dart';
import 'package:chat_demo/providers/
setting_provider.dart';
import 'package:chat_demo/widgets/
loading_view.dart';
import 'package:firebase_storage/
firebase_storage.dart';
```

```dart
import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:chat_demo/constants/
app_constants.dart';

import 'package:fluttertoast/fluttertoast.dart';

import 'package:image_picker/image_picker.dart';

import 'package:provider/provider.dart';


class SettingsPage extends StatelessWidget {

  const SettingsPage({Key? key}) : super(key:
key);


  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text(

          AppConstants.settingsTitle,

          style: TextStyle(color:
ColorConstants.primaryColor),

        ),

        centerTitle: true,

      ),

      body: const SettingsPageState(),

    );

  }

}


class SettingsPageState extends StatefulWidget {

  const SettingsPageState({Key? key}) :
super(key: key);


  @override

  State createState() =>
SettingsPageStateState();

}


class SettingsPageStateState extends
State<SettingsPageState> {

  TextEditingController? controllerNickname;

  TextEditingController? controllerAboutMe;


  String id = '';

  String nickname = '';

  String aboutMe = '';

  String photoUrl = '';


  bool isLoading = false;

  File? avatarImageFile;

  late SettingProvider settingProvider;


  final FocusNode focusNodeNickname =
FocusNode();

  final FocusNode focusNodeAboutMe =
FocusNode();


  @override

  void initState() {

    super.initState();

    settingProvider =
context.read<SettingProvider>();

    readLocal();

  }


  void readLocal() {

    setState(() {

      id =
settingProvider.getPref(FirestoreConstants.id) ?
? "";

      nickname =
settingProvider.getPref(FirestoreConstants.nickn
ame) ?? "";

      aboutMe =
settingProvider.getPref(FirestoreConstants.about
Me) ?? "";
```

```
      photoUrl =
settingProvider.getPref(FirestoreConstants.photo
Url) ?? "";
    });


    controllerNickname =
TextEditingController(text: nickname);
    controllerAboutMe =
TextEditingController(text: aboutMe);
  }


  Future getImage() async {
    ImagePicker imagePicker = ImagePicker();
    PickedFile? pickedFile = await
imagePicker.getImage(source:
ImageSource.gallery).catchError((err) {
      Fluttertoast.showToast(msg:
err.toString());
    });
    File? image;
    if (pickedFile != null) {
      image = File(pickedFile.path);
    }
    if (image != null) {
      setState(() {
        avatarImageFile = image;
        isLoading = true;
      });
      uploadFile();
    }
  }


  Future uploadFile() async {
    String fileName = id;
    UploadTask uploadTask =
settingProvider.uploadFile(avatarImageFile!,
fileName);
    try {
      TaskSnapshot snapshot = await uploadTask;
      photoUrl = await
snapshot.ref.getDownloadURL();
      UserModel updateInfo = UserModel(
        id: id,
        photoUrl: photoUrl,
        nickname: nickname,
        aboutMe: aboutMe,
      );
      settingProvider
          .updateDataFirestore(FirestoreConstant
s.pathUserCollection, id, updateInfo.toJson())
          .then((data) async {
        await
settingProvider.setPref(FirestoreConstants.photo
Url, photoUrl);
        setState(() {
          isLoading = false;
        });
        Fluttertoast.showToast(msg: "Upload
success");
      }).catchError((err) {
        setState(() {
          isLoading = false;
        });
        Fluttertoast.showToast(msg:
err.toString());
      });
    } on FirebaseException catch (e) {
      setState(() {
        isLoading = false;
      });
      Fluttertoast.showToast(msg: e.message ??
e.toString());
    }
  }


  void handleUpdateData() {
```

```
    focusNodeNickname.unfocus();
    focusNodeAboutMe.unfocus();

    setState(() {
      isLoading = true;
    });
    UserModel updateInfo = UserModel(
      id: id,
      photoUrl: photoUrl,
      nickname: nickname,
      aboutMe: aboutMe,
    );
    settingProvider
        .updateDataFirestore(FirestoreConstants.pathUserCollection, id, updateInfo.toJson())
        .then((data) async {
      await settingProvider.setPref(FirestoreConstants.nickname, nickname);
      await settingProvider.setPref(FirestoreConstants.aboutMe, aboutMe);
      await settingProvider.setPref(FirestoreConstants.photoUrl, photoUrl);

      setState(() {
        isLoading = false;
      });

      Fluttertoast.showToast(msg: "Update success");
    }).catchError((err) {
      setState(() {
        isLoading = false;
      });
      Fluttertoast.showToast(msg: err.toString());
    });
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: <Widget>[
        SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              // Avatar
              CupertinoButton(
                onPressed: getImage,
                child: Container(
                  margin: const EdgeInsets.all(20),
                  child: avatarImageFile == null
                      ? photoUrl.isNotEmpty
                          ? ClipRRect(
                              borderRadius: BorderRadius.circular(45),
                              child: Image.network(
                                photoUrl,
                                fit: BoxFit.cover,
                                width: 90,
                                height: 90,
                                errorBuilder: (context, object, stackTrace) {
                                  return const Icon(
```

```
Icons.account_circle,
                                    size: 90,
                                    color:
ColorConstants.greyColor,
                                  );
                                },
                                loadingBuilder:
(BuildContext context, Widget child,
ImageChunkEvent? loadingProgress) {
                                  if
(loadingProgress == null) return child;
                                  return
SizedBox(
                                    width: 90,
                                    height: 90,
                                    child:
Center(
                                      child:
CircularProgressIndicator(
                                        color:
ColorConstants.themeColor,
                                        value:
loadingProgress.expectedTotalBytes != null
                                            ?
loadingProgress.cumulativeBytesLoaded /
loadingProgress.expectedTotalBytes!
                                            :
null,
                                      ),
                                    ),
                                  );
                                },
                              ),
                            )
                          : const Icon(
Icons.account_circle,
                              size: 90,
                              color:
ColorConstants.greyColor,
                            )
                          : ClipRRect(
                              borderRadius:
BorderRadius.circular(45),
                              child: Image.file(
                                avatarImageFile!,
                                width: 90,
                                height: 90,
                                fit: BoxFit.cover,
                              ),
                            ),
                      ),
                    ),

                    // Input
                    Column(
                      children: <Widget>[
                        // Username
                        Container(
                          child: const Text(
                            'Nickname',
                            style: TextStyle(
                                fontStyle:
FontStyle.italic, fontWeight: FontWeight.bold,
color: ColorConstants.primaryColor),
                          ),
                          margin: const
EdgeInsets.only(left: 10, bottom: 5, top: 10),
                        ),
                        Container(
                          child: Theme(
                            data:
Theme.of(context).copyWith(primaryColor:
ColorConstants.primaryColor),
                            child: TextField(
```

```
                        decoration: const
InputDecoration(
                          hintText: 'Sweetie',
                          contentPadding:
EdgeInsets.all(5),
                          hintStyle:
TextStyle(color: ColorConstants.greyColor),
                        ),
                        controller:
controllerNickname,
                        onChanged: (value) {
                          nickname = value;
                        },
                        focusNode:
focusNodeNickname,
                      ),
                    ),
                    margin: const
EdgeInsets.only(left: 30, right: 30),
                  ),

                  // About me
                  Container(
                    child: const Text(
                      'About me',
                      style: TextStyle(
                          fontStyle:
FontStyle.italic, fontWeight: FontWeight.bold,
color: ColorConstants.primaryColor),
                    ),
                    margin: const
EdgeInsets.only(left: 10, top: 30, bottom: 5),
                  ),
                  Container(
                    child: Theme(
                      data:
Theme.of(context).copyWith(primaryColor:
ColorConstants.primaryColor),
                      child: TextField(
                        decoration: const
InputDecoration(
                          hintText: 'Fun, like
travel and play board games...',
                          contentPadding:
EdgeInsets.all(5),
                          hintStyle:
TextStyle(color: ColorConstants.greyColor),
                        ),
                        controller:
controllerAboutMe,
                        onChanged: (value) {
                          aboutMe = value;
                        },
                        focusNode:
focusNodeAboutMe,
                      ),
                    ),
                    margin: const
EdgeInsets.only(left: 30, right: 30),
                  ),
                ],
                crossAxisAlignment:
CrossAxisAlignment.start,
              ),

              // Button
              Container(
                padding: const
EdgeInsets.symmetric(horizontal: 16),
                child: TextButton(
                  onPressed: handleUpdateData,
                  child: const Text(
                    'Update',
                    style: TextStyle(fontSize:
16, color: Colors.white),
                  ),
                  style: ButtonStyle(
```

```
                    backgroundColor:
MaterialStateProperty.all<Color>(ColorConstants.
primaryColor),

                    padding:
MaterialStateProperty.all<EdgeInsets>(

                      const
EdgeInsets.fromLTRB(30, 10, 30, 10),

                    ),

                  ),

                ),

                margin: const
EdgeInsets.only(top: 50, bottom: 50),

              ),

            ],

          ),

          padding: const EdgeInsets.only(left:
15, right: 15),

        ),


        // Loading

        Positioned(child: isLoading ? const
LoadingView() : const SizedBox.shrink()),

      ],

    );

  }

}
```

sign_in.dart

```
import 'package:chat_demo/pages/home_page.dart';

import 'package:chat_demo/pages/
sign_up_page.dart';

import 'package:chat_demo/pages/
splash_page.dart';

import 'package:chat_demo/providers/
auth_provider.dart';

import 'package:flutter/material.dart';

import 'package:fluttertoast/fluttertoast.dart';

import 'package:provider/provider.dart';


class SignInPage extends StatefulWidget {

  const SignInPage({Key? key}) : super(key:
key);


  @override

  State<SignInPage> createState() =>
_SignInPageState();

}


class _SignInPageState extends State<SignInPage>
{

  final TextEditingController _emailCtr =
TextEditingController();

  final TextEditingController _pwCtr =
TextEditingController();


  @override

  void dispose() {

    _emailCtr.dispose();

    _pwCtr.dispose();

    super.dispose();

  }


  @override

  Widget build(BuildContext context) {

    AuthProvider authProvider =
Provider.of<AuthProvider>(context);

    switch (authProvider.status) {

      case Status.authenticateError:

        Fluttertoast.showToast(msg: "Sign in
fail");

        break;

      case Status.authenticated:

        Fluttertoast.showToast(msg: "Sign in
success");


WidgetsBinding.instance?.addPostFrameCallback((_
) {
```

```
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context)
=> const HomePage()),
          );
        });
        break;
      default:
        break;
    }

    return Scaffold(
      appBar: AppBar(
        title: const Text('Sign in'),
        centerTitle: true,
      ),
      body: Padding(
        padding: const
EdgeInsets.symmetric(horizontal: 32),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.stretch,
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            TextFormField(
              controller: _emailCtr,
              decoration: const
InputDecoration(hintText: 'Email'),
            ),
            const SizedBox(height: 8),
            TextFormField(
              controller: _pwCtr,
              obscureText: true,
              decoration: const
InputDecoration(hintText: 'Password'),
            ),
            const SizedBox(height: 32),
            ElevatedButton(
              onPressed: () =>
_signIn(authProvider),
              child: const Text('Sign in'),
            ),
            const SizedBox(height: 32),
            TextButton(
              onPressed: _signUp,
              child: const Text('Go to Sign
up'),
            ),
          ],
        ),
      ),
    );
  }

  Future<void> _signIn(AuthProvider
authProvider) async {
    await
authProvider.signInWithEmailAndPassword(
      _emailCtr.text.trim(),
      _pwCtr.text.trim(),
    );
  }

  void _signUp() => Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const
SignUpPage(),
        ),
      );
}
```

sign_up.dart

```
import 'package:chat_demo/pages/home_page.dart';

import 'package:chat_demo/pages/
sign_in_page.dart';

import 'package:chat_demo/pages/
splash_page.dart';

import 'package:chat_demo/providers/
auth_provider.dart';

import 'package:flutter/material.dart';

import 'package:fluttertoast/fluttertoast.dart';

import 'package:provider/provider.dart';


class SignUpPage extends StatefulWidget {

  const SignUpPage({Key? key}) : super(key:
key);


  @override

  State<SignUpPage> createState() =>
_SignUpPageState();

}


class _SignUpPageState extends State<SignUpPage>
{

  final TextEditingController _emailCtr =
TextEditingController();

  final TextEditingController _pwCtr =
TextEditingController();

  final TextEditingController _confirmPwCtr =
TextEditingController();


  @override

  void dispose() {

    _emailCtr.dispose();

    _pwCtr.dispose();

    _confirmPwCtr.dispose();

    super.dispose();

  }


  @override

  Widget build(BuildContext context) {

    AuthProvider authProvider =
Provider.of<AuthProvider>(context);

    switch (authProvider.status) {

      case Status.authenticateError:

        Fluttertoast.showToast(msg: "Sign up
fail");

        break;

      case Status.authenticated:

        Fluttertoast.showToast(msg: "Sign up
success");


WidgetsBinding.instance?.addPostFrameCallback((_
) {

          Navigator.pushReplacement(

            context,

            MaterialPageRoute(builder: (context)
=> const HomePage()),

          );

        });

        break;

      default:

        break;

    }


    return Scaffold(

      appBar: AppBar(

        title: const Text('Sign up'),

        centerTitle: true,

      ),

      body: Padding(

        padding: const
EdgeInsets.symmetric(horizontal: 32),

        child: Column(

          crossAxisAlignment:
CrossAxisAlignment.stretch,
```

```
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            TextFormField(
              controller: _emailCtr,
              decoration: const
InputDecoration(hintText: 'Email'),
            ),
            const SizedBox(height: 8),
            TextFormField(
              controller: _pwCtr,
              obscureText: true,
              decoration: const
InputDecoration(hintText: 'Password'),
            ),
            const SizedBox(height: 8),
            TextFormField(
              controller: _confirmPwCtr,
              obscureText: true,
              decoration: const
InputDecoration(hintText: 'Confirm password'),
            ),
            const SizedBox(height: 32),
            ElevatedButton(
              onPressed: () =>
_signUp(authProvider),
              child: const Text('Sign up'),
            ),
            const SizedBox(height: 32),
            TextButton(
              onPressed: _signIn,
              child: const Text('Go to Sign
in'),
            ),
          ],
        ),
      ),
    );
  }

  Future<void> _signUp(AuthProvider
authProvider) async {
    if (_pwCtr.text != _confirmPwCtr.text) {
      const snackBar = SnackBar(
        content: Text('Passwords do not match'),
      );

ScaffoldMessenger.of(context).showSnackBar(snack
Bar);

      return;
    }

    await
authProvider.signUpWithEmailAndPassword(
      _emailCtr.text.trim(),
      _pwCtr.text.trim(),
    );
  }

  void _signIn() =>
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const
SignInPage(),
        ),
      );
}
```

splash_page.dart

```
import 'package:chat_demo/constants/
color_constants.dart';
```

```
import 'package:chat_demo/pages/home_page.dart';

import 'package:chat_demo/pages/
sign_in_page.dart';

import 'package:chat_demo/providers/
auth_provider.dart';

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';


class SplashPage extends StatefulWidget {

  const SplashPage({Key? key}) : super(key:
key);


  @override

  SplashPageState createState() =>
SplashPageState();

}


class SplashPageState extends State<SplashPage>
{

  @override

  void initState() {

    super.initState();

    Future.delayed(const Duration(seconds: 1),
() {

      checkSignedIn();

    });

  }


  void checkSignedIn() async {

    AuthProvider authProvider =
context.read<AuthProvider>();

    bool isLoggedIn = await
authProvider.isLoggedIn();

    if (isLoggedIn) {

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(builder: (context) =>
const HomePage()),

      );

      return;

    }

    Navigator.pushReplacement(

      context,

      MaterialPageRoute(builder: (context) =>
const SignInPage()),

    );

  }


  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: const [

            SizedBox(height: 20),

            SizedBox(

              width: 20,

              height: 20,

              child:


CircularProgressIndicator(color:
ColorConstants.themeColor),

            ),

          ],

        ),

      ),

    );

  }

}
```

auth_provider.dart

```
import 'package:chat_demo/constants/
firestore_constants.dart';
```

```
import 'package:chat_demo/models/
user_chat_model.dart';

import 'package:cloud_firestore/
cloud_firestore.dart';

import 'package:firebase_auth/
firebase_auth.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/
shared_preferences.dart';


enum Status {

  uninitialized,

  authenticated,

  authenticating,

  authenticateError,

  authenticateCanceled,

}


class AuthProvider extends ChangeNotifier {

  final FirebaseAuth _auth =
FirebaseAuth.instance;

  final FirebaseAuth firebaseAuth;

  final FirebaseFirestore firebaseFirestore;

  final SharedPreferences prefs;


  Status _status = Status.uninitialized;


  Status get status => _status;


  AuthProvider({

    required this.firebaseAuth,

    required this.prefs,

    required this.firebaseFirestore,

  });


  String? getUserFirebaseId() {

    return
prefs.getString(FirestoreConstants.id);

  }


  Future<bool> isLoggedIn() async {

    bool isLoggedIn = _auth.currentUser != null;

    if (isLoggedIn &&


prefs.getString(FirestoreConstants.id)?.isNotEmp
ty == true) {

      return true;

    } else {

      return false;

    }

  }


  Future<void> signInWithEmailAndPassword(String
email, String password) async {

    _status = Status.authenticating;

    notifyListeners();


    try {

      await _auth

          .signInWithEmailAndPassword(email:
email, password: password)

          .then((user) async {

        await handleLogin(user);

      });

    } on FirebaseAuthException catch (e) {}

  }


  Future<void> signUpWithEmailAndPassword(String
email, String password) async {

    try {

      _auth

          .createUserWithEmailAndPassword(

        email: email,
```

```
        password: password,
      )
          .then((user) async {
        await handleLogin(user);
      });
    } on FirebaseAuthException catch (e) {
    } catch (e) {}
  }

  Future<bool> handleLogin(UserCredential
userCredential) async {
    final firebaseUser = userCredential.user;

    if (firebaseUser != null) {
      final QuerySnapshot result = await
firebaseFirestore
          .collection(FirestoreConstants.pathUse
rCollection)
          .where(FirestoreConstants.id,
isEqualTo: firebaseUser.uid)
          .get();
      final List<DocumentSnapshot> documents =
result.docs;
      if (documents.isEmpty) {
        // Writing data to server because here
is a new user
        firebaseFirestore
            .collection(FirestoreConstants.pathU
serCollection)
            .doc(firebaseUser.uid)
            .set({
          FirestoreConstants.nickname:
firebaseUser.displayName,
          FirestoreConstants.photoUrl:
firebaseUser.photoURL,
          FirestoreConstants.id:
firebaseUser.uid,
          'createdAt':
DateTime.now().millisecondsSinceEpoch.toString()
,
          FirestoreConstants.chattingWith: null
        });

        // Write data to local storage
        User? currentUser = firebaseUser;
        await
prefs.setString(FirestoreConstants.id,
currentUser.uid);
        await prefs.setString(
            FirestoreConstants.nickname,
currentUser.displayName ?? "");
        await prefs.setString(
            FirestoreConstants.photoUrl,
currentUser.photoURL ?? "");
      } else {
        // Already sign up, just get data from
firestore
        DocumentSnapshot documentSnapshot =
documents[0];
        UserModel userChat =
UserModel.fromDocument(documentSnapshot);
        // Write data to local
        await
prefs.setString(FirestoreConstants.id,
userChat.id);
        await
prefs.setString(FirestoreConstants.nickname,
userChat.nickname);
        await
prefs.setString(FirestoreConstants.photoUrl,
userChat.photoUrl);
        await
prefs.setString(FirestoreConstants.aboutMe,
userChat.aboutMe);
      }
      _status = Status.authenticated;
      notifyListeners();
      return true;
    } else {
      _status = Status.authenticateError;
      notifyListeners();
```

```
      return false;
    }
  }

  Future<void> handleSignOut() async {
    _status = Status.uninitialized;
    await firebaseAuth.signOut();
  }
}
```

chat_provider.dart

```
import 'dart:io';

import 'package:chat_demo/constants/
firestore_constants.dart';
import 'package:chat_demo/models/
message_model.dart';
import 'package:cloud_firestore/
cloud_firestore.dart';
import 'package:firebase_storage/
firebase_storage.dart';
import 'package:shared_preferences/
shared_preferences.dart';

class ChatProvider {
  final SharedPreferences prefs;
  final FirebaseFirestore firebaseFirestore;
  final FirebaseStorage firebaseStorage;

  ChatProvider({required this.firebaseFirestore,
required this.prefs, required
this.firebaseStorage});

  String? getPref(String key) {
    return prefs.getString(key);
  }

  UploadTask uploadFile(File image, String
fileName) {
    Reference reference =
firebaseStorage.ref().child(fileName);
    UploadTask uploadTask =
reference.putFile(image);
    return uploadTask;
  }

  Future<void> updateDataFirestore(String
collectionPath, String docPath, Map<String,
dynamic> dataNeedUpdate) {
    return
firebaseFirestore.collection(collectionPath).doc
(docPath).update(dataNeedUpdate);
  }

  Stream<QuerySnapshot> getChatStream(String
groupChatId, int limit) {
    return firebaseFirestore
        .collection(FirestoreConstants.pathMessa
geCollection)
        .doc(groupChatId)
        .collection(groupChatId)
        .orderBy(FirestoreConstants.timestamp,
descending: true)
        .limit(limit)
        .snapshots();
  }

  void sendMessage(String content, int type,
String groupChatId, String currentUserId, String
peerId) {
    final datetime =
DateTime.now().millisecondsSinceEpoch.toString()
;
    DocumentReference documentReference =
firebaseFirestore
        .collection(FirestoreConstants.pathMessa
geCollection)
        .doc(groupChatId)
```

```
        .collection(groupChatId)
        .doc(datetime);

    MessageModel messageChat = MessageModel(
      idFrom: currentUserId,
      idTo: peerId,
      timestamp: datetime,
      content: content,
      type: type,
    );

FirebaseFirestore.instance.runTransaction((trans
action) async {
      transaction.set(
        documentReference,
        messageChat.toJson(),
      );
    });
  }
}

class TypeMessage {
  static const text = 0;
  static const image = 1;
  static const sticker = 2;
}
```

home_provider.dart

```
import 'package:chat_demo/constants/
firestore_constants.dart';
import 'package:cloud_firestore/
cloud_firestore.dart';

class HomeProvider {
  final FirebaseFirestore firebaseFirestore;

  HomeProvider({required
this.firebaseFirestore});

  Future<void> updateDataFirestore(String
collectionPath, String path, Map<String, String>
dataNeedUpdate) {
    return
firebaseFirestore.collection(collectionPath).doc
(path).update(dataNeedUpdate);
  }

  Stream<QuerySnapshot>
getStreamFireStore(String pathCollection, int
limit, String? textSearch) {
    if (textSearch?.isNotEmpty == true) {
      return firebaseFirestore
          .collection(pathCollection)
          .limit(limit)
          .where(FirestoreConstants.nickname,
isEqualTo: textSearch)
          .snapshots();
    } else {
      return
firebaseFirestore.collection(pathCollection).lim
it(limit).snapshots();
    }
  }
}
```

setting_provider.dart

```
import 'dart:io';

import 'package:cloud_firestore/
cloud_firestore.dart';
import 'package:firebase_storage/
firebase_storage.dart';
```

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


class SettingProvider {
  final SharedPreferences prefs;
  final FirebaseFirestore firebaseFirestore;
  final FirebaseStorage firebaseStorage;

  SettingProvider({
    required this.prefs,
    required this.firebaseFirestore,
    required this.firebaseStorage,
  });

  String? getPref(String key) {
    return prefs.getString(key);
  }

  Future<bool> setPref(String key, String value)
async {
    return await prefs.setString(key, value);
  }

  UploadTask uploadFile(File image, String
fileName) {
    Reference reference =
firebaseStorage.ref().child(fileName);
    UploadTask uploadTask =
reference.putFile(image);
    return uploadTask;
  }

  Future<void> updateDataFirestore(String
collectionPath, String path, Map<String, String>
dataNeedUpdate) {
    return
firebaseFirestore.collection(collectionPath).doc
(path).update(dataNeedUpdate);
  }
}
```

debouncer.dart

```
import 'dart:async';
import 'dart:ui';

class Debouncer {
  final int milliseconds;
  Timer? _timer;

  Debouncer({required this.milliseconds});

  run(VoidCallback action) {
    _timer?.cancel();

    _timer = Timer(Duration(milliseconds:
milliseconds), action);
  }
}
```

utilities.dart

```
import 'package:flutter/material.dart';

class Utilities {
  static bool isKeyboardShowing() {
    if (WidgetsBinding.instance != null) {
      return
WidgetsBinding.instance!.window.viewInsets.botto
m > 0;
    } else {
      return false;
    }
  }
```

```
  static closeKeyboard(BuildContext context) {
    FocusScopeNode currentFocus =
FocusScope.of(context);
    if (!currentFocus.hasPrimaryFocus) {
      currentFocus.unfocus();
    }
  }
}
```

loading_view.dart

```
import 'package:chat_demo/constants/
color_constants.dart';
import 'package:flutter/material.dart';
class LoadingView extends StatelessWidget {
  const LoadingView({Key? key}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    return Container(
      child: const Center(
        child: CircularProgressIndicator(
          color: ColorConstants.themeColor,
        ),
      ),
      color: Colors.white.withOpacity(0.8),
    );
  }
}
```

main.dart

```
import 'package:chat_demo/constants/
app_constants.dart';
import 'package:chat_demo/constants/
color_constants.dart';
import 'package:chat_demo/pages/
splash_page.dart';
import 'package:chat_demo/providers/
auth_provider.dart';
import 'package:chat_demo/providers/
chat_provider.dart';
import 'package:chat_demo/providers/
home_provider.dart';
import 'package:chat_demo/providers/
setting_provider.dart';
import 'package:cloud_firestore/
cloud_firestore.dart';
import 'package:firebase_auth/
firebase_auth.dart';
import 'package:firebase_core/
firebase_core.dart';
import 'package:firebase_storage/
firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:shared_preferences/
shared_preferences.dart';
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  SharedPreferences prefs = await
SharedPreferences.getInstance();
  runApp(MyApp(prefs: prefs));
}

class MyApp extends StatelessWidget {
  final SharedPreferences prefs;
  final FirebaseFirestore firebaseFirestore =
FirebaseFirestore.instance;
  final FirebaseStorage firebaseStorage =
FirebaseStorage.instance;

  MyApp({Key? key, required this.prefs}) :
super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider<AuthProvider>(
          create: (_) => AuthProvider(
            firebaseAuth: FirebaseAuth.instance,
            prefs: prefs,
            firebaseFirestore:
firebaseFirestore,
          ),
        ),
        Provider<SettingProvider>(
          create: (_) => SettingProvider(
            prefs: prefs,
            firebaseFirestore:
firebaseFirestore,
            firebaseStorage: firebaseStorage,
          ),
        ),
        Provider<HomeProvider>(
          create: (_) => HomeProvider(
            firebaseFirestore:
firebaseFirestore,
          ),
        ),
        Provider<ChatProvider>(
          create: (_) => ChatProvider(
            prefs: prefs,
            firebaseFirestore:
firebaseFirestore,
            firebaseStorage: firebaseStorage,
          ),
        ),
      ],
      child: MaterialApp(
        title: AppConstants.appTitle,
        theme: ThemeData(
          appBarTheme: const AppBarTheme(
            elevation: 0,
            color: Colors.white,
            iconTheme: IconThemeData(
              color:
ColorConstants.primaryColor,
            ),
            actionsIconTheme: IconThemeData(
              color:
ColorConstants.primaryColor,
            ),
            titleTextStyle: TextStyle(
              color:
ColorConstants.primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 18,
            ),
          ),
          primaryColor:
ColorConstants.themeColor,
        ),
        home: const SplashPage(),
        debugShowCheckedModeBanner: false,
      ),
    );
  }
}
```