МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ANDROID-ДОДАТКУ «CHAT APP» ЗАСОБАМИ FIREBASE»**

Виконала: студентка 2 курсу, групи 8.1210-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.Ю. Смородіна
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний

Кафедра Програмної інженерії

Рівень вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
Лісняк А.О.
(підпис)

« » 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Смородіній Валерії Юріївни
(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) «Розробка Android-додатку «Chat app» засобами Firebase»

керівник роботи (проекту) Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 09 » червня 2021 року № 850-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі. 2. Опис основних бізнес-процесів. 3. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.2. Аналіз предметної області. 3. Збір та аналіз вимог до ПЗ. 4. Функціональна модель основних бізнес процесів. 5. Проектування системи. 6. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Діаграми, таблиці сутностей, презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Кудін О.В., к.ф.-м.н., доцент	01.09.2021	25.09.2021
2	Кудін О.В., к.ф.-м.н., доцент	25.09.2021	17.10.2021
3	Кудін О.В., к.ф.-м.н., доцент	17.10.2021	07.10.2021

7. Дата видачі завдання 09.06.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	01.09.2021	
2.	Збір вихідних даних.	04.09.2021	
3.	Обробка методичних та теоретичних	25.09.2021	
	джерел.		
4.	Розробка першого розділу.	17.10.2021	
5.	Розробка другого розділу	25.10.2021	
6.	Розробка третього розділу.	07.11.2021	
7.	Оформлення та нормоконтроль	18.11.2021	
	кваліфікаційної роботи.		
8.	Захист кваліфікаційної роботи.	09.12.2021	

Студент _____ В.Ю. Смородіна
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ С.П. Швидка
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка Android-додатку «Chat app» засобами Firebase»: 48 с., 22 рис., 5 табл., 10 джерел, 1 додаток.

МЕСЕНДЖЕР, МОБІЛЬНИЙ ДОДАТОК, СЕРВІС ОБМІНУ ПОВІДОМЛЕННЯМИ, ANDROID STUDIO, FIREBASE, FLUTTER.

Об’єкт дослідження – процес передачі повідомлень через мобільний додаток.

Мета дослідження – розробка програмної реалізації мобільного додатку з графічним інтерфейсом для обміну повідомленнями у мережі.

У кваліфікаційної роботі викладені теоретичні відомості про сервіси обміну повідомленнями та мобільний додаток. На основі даних теоретичних відомостей розроблено програмну реалізацію електронного мобільного додатку месенджер та користувацький інтерфейс у середовищі програмування Android Studio у вигляді проекту мобільного додатку, використовуючи мову програмування Dart. В результаті роботи отримано багатокористувацький додаток для сервісу обміну повідомленнями.

SUMMARY

Master’s qualifying paper of "Development Android app “Chat app” with Firebase": 48 pages, 22 figures, 5 tables, 10 references, 1 supplements.

ANDROID STUDIO, DART, FIREBASE, MESSAGING SERVICE, MOBILE APP.

The object of the study is the process of sending messages through a mobile application.

The aim of the study is to develop a mobile application for the messaging service.

Qualification work contains theoretical information about messaging services and a mobile application. Based on the data of theoretical information, the software implementation of the electronic mobile application of the messaging service and the user interface in the Android Studio programming environment in the form of a mobile application project using the Dart programming language have been developed. As a result of work the multiuser application for messaging service is received.

ЗМІСТ

ВСТУП

Світ не стоїть на місці і важко уявити сучасну людину без засобів зв'язку. В даний час більша частина комунікації припадає саме на текстові повідомлення – ми звикли домовлятися про зустрічі та обговорювати нові епізоди серіалів у чатах.

При всій вивченості сервісів обміну повідомленнями, сучасні розробники завжди мають місце для кроку вперед. Це галузь, яка продовжує розвиватись – питання вдосконалення сервісів обміну повідомленнями є відкритим.

Актуальність месенджерів полягає в тому, що більшість людей обирають саме такий спосіб комунікації. Популярність мобільних додатків для сервісів обміну повідомленнями значно зросла, обійшовши соціальні мережі. При цьому написання тексту вимагає додаткової уваги і займає більше часу, ніж надсилання голосового повідомлення чи зображення. Як користувач декількох подібних аналогів, мною було виявлено, що більшість з цих додатків не допомагають спростити процес комунікації та мають занадто складний інтерфейс. Ця сукупність фактів й зумовила напрямок дослідження.

Метою кваліфікаційної роботи є розробка додатку для сервісу, який спростить процес комунікації в мережі. Для досягнення цієї мети я визначила наступні завдання:

- зібрати інформацію для написання технічного завдання;
- порівняти існуючи аналоги месенджерів та виявити їх недоліки;
- провести огляд технології, які будуть використані для створення додатку;
- спроектувати інформаційну систему;
- реалізувати та протестувати створену систему;
- оформити документацію до дипломної роботи.

У ході вирішення поставлених завдань було створено мобільний додаток для сервісу обміну миттєвими повідомленнями, що відповідає вимогам. Додаток представляє собою месенджер, який дає змогу обміну повідомленнями у мережі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

Додатки сервісу обміну ммиттєвими повідомленнями користуються великим попитом за можливість зручної комунікації у мережі. Зараз у них сконцентрована більшість онлайн-комунікації – як особистої, так і робочої.

Оскільки швидкість розвитку подій та прийняття рішень вимагає від людини рухливості в усіх сферах життя, у тому числі в спілкуванні, мобільність є головною перевагою, як і безкоштовні дзвінки та повідомлення. До недоліків можна віднести, що більшість месенджерів не можна вважати повністю захищеними.

Метою створення системи є спрощення процесу онлайн-комунікації. На сьогоднішній день існує багато аналогів, вони усі задовольняють головну мету додатку – обмін повідомленнями, але до вирішення одних і тих же завдань підходять по-різному. ChatApp представляє собою сервіс, готовий надати послугу обміну повідомленнями у мережі.

Додаток передбачає можливості:

- реєстрації;
- авторизації;
- відправлення повідомлень через мобільні дані та WiFi;
- обмін файлами, фотографіями;
- здійснення аудіо і відео дзвінків.

1.2 Огляд технологій

За останні кілька років ми поступово переходили від локальних систем до хмари для зберігання і обробки наших даних. Такі хмарні бази даних, як

Amazon Relational Database, Clustrix Database as a Service, Amazon SimpleDB, зокрема сервіси Firebase мають високу популярність у розробників.

Firebase відноситься до набору інструментів, пропонованих Google для створення масштабованих додатків в хмарі. Основним продуктом є Firebase Realtime Database. Ця база даних в реальному часі має змогу обробляти поновлення, коли пристрій знаходиться в автономному режимі, і синхронізувати зміни при повторному підключенні до мережі. Оскільки для багатьох додатків ідентифікація є необхідною, Firebase надає зручну, призначену для користувача, автентифікацію. Користувач матиме можливість отримати доступ к додатку з будь-якого пристрою.

Android надає можливість збереження даних у декілька способів. Один з них використання SQLite. SQLite – це реляційна база даних, яка має відкритий вихідний код. Ця БД побудована на операційній системі Android, що надає можливість використовувати її без додаткових конфігурацій та доступна на будь-якому Android-пристрої [4].

1.3 User Story (Клієнт)

1.3.1 Доступ до додатка.

Реєстрація клієнта. Як користувач я можу:

– зареєструватися в додатку;

– при неуспішній реєстрації бачити повідомлення про помилку;

Авторизація користувача. Як користувач я можу:

– увійти в програму за допомогою номера телефону і пароля;

– бачити повідомлення про помилку, якщо я ввів невірні дані;

– бачити повідомлення про помилку, якщо я не пройшов реєстрацію;

– відновити пароль.

1.3.2 Робота з чатами.

Як клієнт, я можу працювати з чатами, щоб відправляти повідомлення:

– почати новий чат;
– вибирати існуючий чат;
– відправити текстове повідомлення;
– відправити файл, аудіо- та відео повідомлення;
– почати аудіо- або відео дзвінок.

1.3.3 Робота з контактами.

Як клієнт, я можу працювати з контактами, щоб контролювати список контактів, які в ньому знаходяться:

– бачити список контактів;
– переходити до чату з обраним контактом;
– додати новий контакт;
– здійснювати пошук по контактам.

1.3.4 Робота із дзвінками.

Як клієнт, я можу здійснювати дзвінки:

– бачити список дзвінків;
– здійснювати аудіо дзвінок;
– здійснювати відео дзвінок;
– здійснювати пошук по дзвінкам.

1.4 Огляд аналогів

Останнім часом месенджери стають дедалі популярнішими. У зв'язку з попитом споживачів висока конкуренція. Всі додатки об'єднує подібний один до одного функціонал. У таблиці 1.1 наведені приклади найпопулярніших сервісів обміну повідомленнями у мережі.

Таблиця 1.1 – Аналоги мобільних додатків

	Telegram	**Whatsapp**	**Viber**	**Hangouts**
Переваги	Розширений функціонал, що постійно оновлюється.	Великий вибір функцій та мов роблять додаток доступним для усіх.	Здійснення дзвінків на будь-які мобільні номера, навіть якщо їх власників немає у Viber (за ці дзвінкі окрема плата).	Мультиплатформеність.
Недоліки	Відправлення контактів на сервер.	Проблеми з безпечністю .	Відсутність можливості приховування номеру телефона у налаштуваннях приватності . Нав’язлива реклама.	Надто складний інтерфейс.

Месенджери надані кількома компаніями, кожна з яких має плюси і мінуси. Усі додатки користуються попитом серед користувачів, мають автоматизовану систему та виконують головну функцію – обмін повідомленнями. До недоліків можна віднести проблеми з безпечністю та надто складний інтерфейс у додатках.

1.5 Технічне завдання

1.5.1 Вимоги до системи в цілому

Робота із додатком відбувається наступним чином. При запуску програми відкривається головне вікно. При натисканні кнопки реєстрації користувача відкривається форма реєстрації. При натисканні кнопки «Sign Up» відбувається реєстрація користувача у системі. Користувач бачить повідомлення про реєстрацію. Якщо була натиснута кнопка входу в систему, відкриється вікно автентифікації. Користувач може пройти автентифікацію шляхом введення номера телефону та пароля у відповідні поля та подальшим натисканням кнопки «Sign in». Якщо користувач ввів некоректні дані, то буде виведено повідомлення з помилкою. При успішному вході відкриється головний екран з чатами.

Навігація по додатку може бути здійснена через головний екран з навігаційними кнопками «Chats», «People», «Calls», «Profile».

Для початку спілкування, треба натиснути на кнопку додавання контакту або обрати існуючий чат. Користувач вводить повідомлення у полі введення тексу.

Щоб надіслати документ, аудіо-повідомлення, відео-повідомлення або перейти до Галереї, щоб вибрати фотографію\відео, користувачу треба вибрати та натиснути відповідну іконку. Для відправки обраних файлів користувачу потрібно натиснути кнопку відправки.

Щоб зробити аудіо- або відео дзвінок через вікно чату, натискається відповідна кнопка на екрані.

При натисканні на кнопку «People» відкривається вікно контактів користувача. При натисканні на контакт користувач переходить в чат. При використанні пошукового рядку та введення назви, здійснюється пошук та виведення його результатів.

При натисканні кнопки «Calls» відкривається вікно, що містить останні аудіо- та відео дзвінки.

При натисканні кнопки «Profile» відкривається вікно з інформацією про користувача. Для зміни інформації натискається кнопка «Edit Profile».

1.5.2 Вимоги до структури додатка

Додаток повинен складатися з наступних вікон:

- головне вікно, на якому відображається список чатів;
- вікно контактів;
- вікно дзвінків;
- вікно профілю користувача;
- вікно авторизації користувача;
- вікно реєстрації користувача.

Пункти, з яких складається головне вікно:

- перегляд чатів;
- відкриття існуючого чату;
- додавання нового контакту;
- навігація по додатку за допомогою кнопок «Chats», «People», «Calls», «Profile».

1.5.3 Вимоги до клієнтської частини

Для коректної роботи додатку потрібне виконання наступних вимог:

- операційна система Android не нижче 4.4;
- підключення до мережі Інтернет;
- 35 мб вільного місця на пристрої.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Функціональна модель ІС

Функціональне моделювання дає уявлення про те, що повинна робити система. Існує широкий спектр методів для побудови діаграм процесів, таких як блок-схеми і діаграми потоків даних. IDEF0, у багатьох відношеннях, дуже простий метод.

Головний бізнес процес додатку Chat app – це функція відправки миттєвих повідомлень. Користувач обирає потрібний чат, переходить до діалогу та відправляє повідомлення. Система за своєю структурою є складною та багаторівневою.

У результаті аналізу предметної області було побудовано функціональну модель системи (див. рис. 2.1).

Рисунок 2.1 – Основний процес інформаційної системи у нотації IDEF0

Для загального процесу інформаційної системи може бути представлена декомпозиція блоку на 5 конкретизованих частин (див. рис. 2.2).

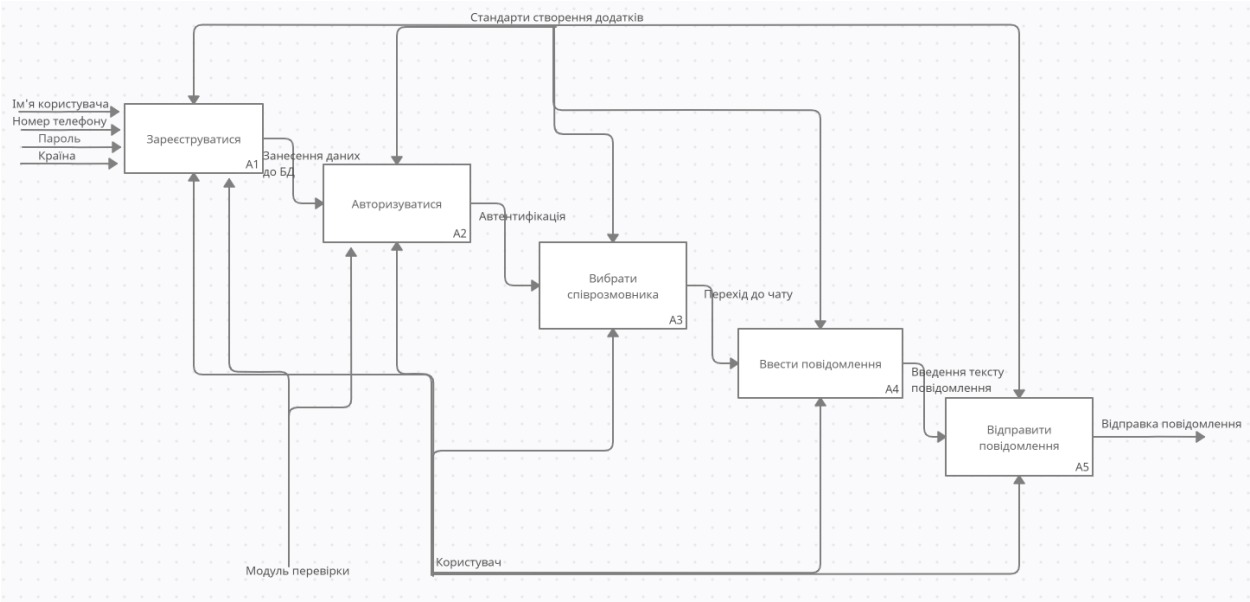

Рисунок 2.2 – Декомпозиція основного блоку у нотації IDEF0

На основі побудованої функціональної моделі було побудовано діаграму діяльності бізнес-процесу у нотації BPMN (див. рис. 2.3–2.4). Стандартна модель і нотація бізнес-процесу BPMN наддасть можливість розуміти свої внутрішні бізнес-процедури в графічної нотації.

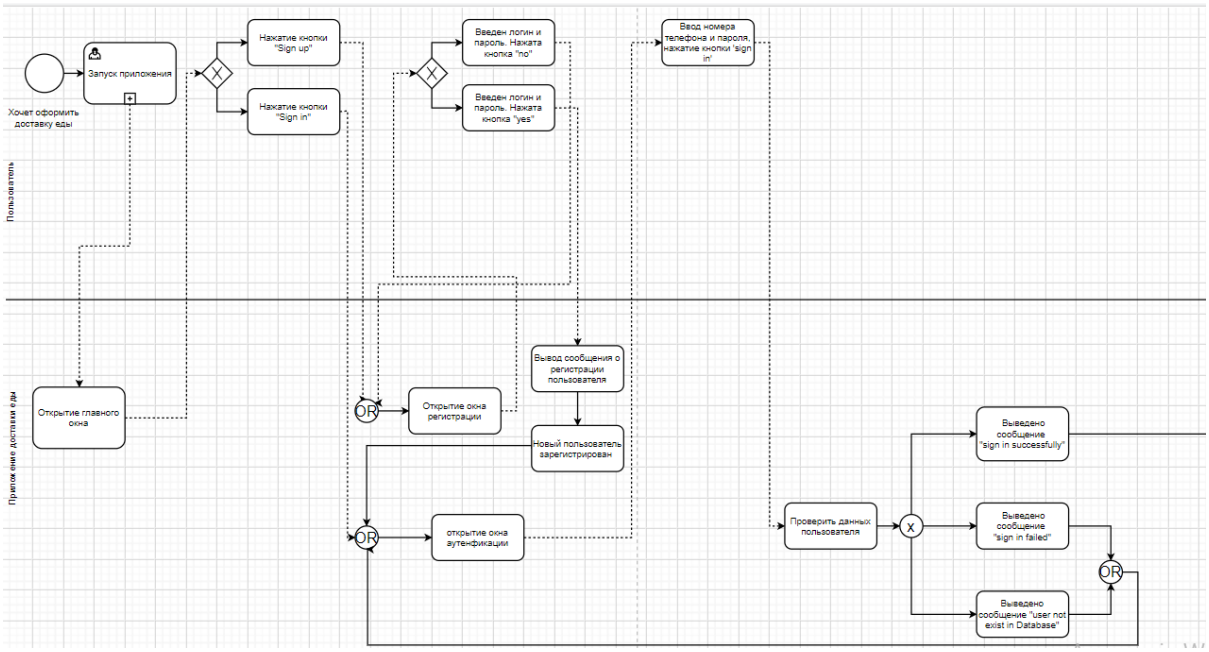

Рисунок 2.3 – Діаграма бізнес-процесу реєстрації та автентифікації у нотації BPMN

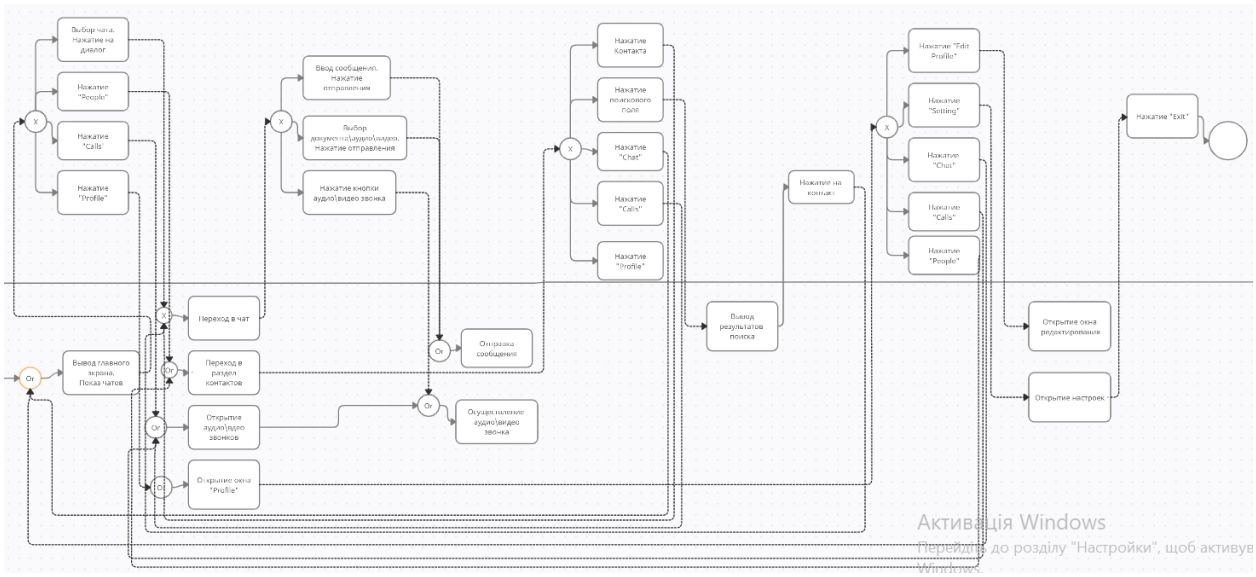

Рисунок 2.4 – Діаграма бізнес-процесу у нотації BPMN

Як показано на діаграмі, користувач взаємодіє з системою протягом всього бізнес-процесу.

2.2 Інформаційна модель ІС

Діаграма класів, будучи логічним поданням моделі, являє детальну інформацію про структуру моделі системи з використанням термінології класів об'єктно програмування, а саме: о внутрішньому устрою системи (про архітектуру системи). На діаграмі класів можуть бути вказані внутрішня структура і типи відносин між окремими об'єктами і підсистемами [1]. Схема бази даних приведена на рис. 2.5, вона відображає таблиці і зв’язки між ними.

База даних для мобільного додатку складається з 4 таблиць, кожна з яких характеризує окрему сутність.

Для кожної сутності бази даних відповідно сформовані таблиці, зображені нижче.

Таблиця 2.1 «User» містить інформацію про користувача, такі як ім’я, пароль та статус.

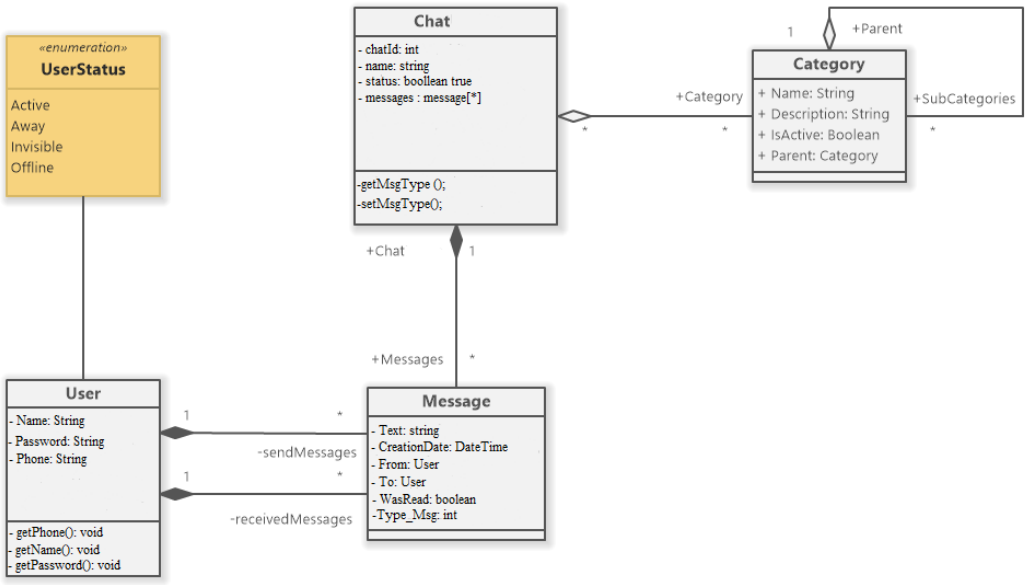

Рисунок 2.5 – ER-діаграма

Таблиця 2.1 – Users

Назва поля	Тип даних	Властивості	Опис
Phone	Int		Містить номер телефону користувача
Name	String		Містить ім’я користувача
Password	String		Містить пароль для входу у систему

У таблиці 2.2 «Category» міститься інформація щодо категорії.

Таблиця 2.2 – Category

Назва поля	Тип даних	Властивості	Опис
Name	String		Містить назву категорії
Description	String		Містить опис категорії
IsActive	Boolean		Містить інформацію щодо активності

У таблиці 2.3 «Message» міститься інформація про статус користувача.

Таблиця 2.3 – Message

Назва поля	Тип даних	Властивості	Опис
Text	String		Містить повідомлення
CreationDate	DataTime		Містить інформацію про час
From	String	FK на таблицю «User»	Містить інформацію про відправника
To	String	FK на таблицю «Chat»	Містить інформацію про отримувача
WasRead	Boolean		Статус повідомлення
Type_Msg	Int		Тип повідомлення

У таблиці 2.4 «Chat» міститься інформація про чат.

Таблиця 2.4 – Chat

Назва поля	Тип даних	Властивості	Опис
ChatId	Int	unique	Містить id чату
Name	String		Містить інформацію щодо назви
Status	Boolean		Містить статус
Message	String	FK на таблицю «Message»	Містить повідомлення

2.3 Діаграма варіантів використання системи

За допомогою діаграми варіантів використання виконуються опис функціональних вимог до системи з метою кращого розуміння роботи системи. Chat App можна розглядати з точки зору користувача.

При використанні системи користувачу доступні наступні варіанти (див. рис. 2.6).

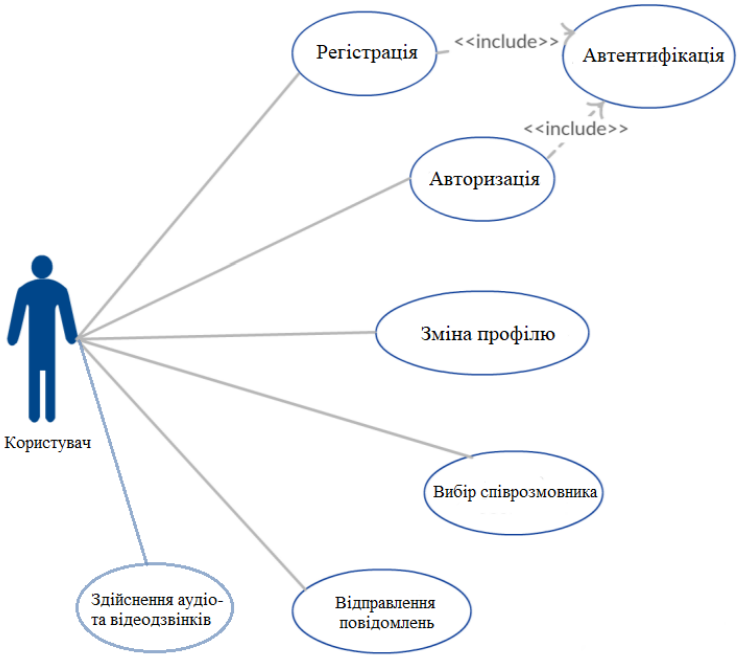

Рисунок 2.6 – Діаграма використання Користувачем

2.4 Діаграма класів

Для моделювання майбутніх класів, їх атрибутів та методів, а також зв’язків використовуються діаграми класів. Для побудови діаграми класів було обрано методологію IDEF4.

Об'єктно-орієнтований метод є основною методологією проектування для сучасного проектування і розробки програмного забезпечення. Особливістю стандарту IDEF4 є можливість подання впливу спадковості

класів, складу об'єктів, функціональної декомпозиції і поліморфізму на проектування об'єктів [2].

Функціонально, додаток складається з наведених нижче класів, які є схемою уявлення Android-додатків.

Кожен екран користувача інтерфейсу представлений класом Activity і по суті є окремою формою додатка. Android-додаток здатен складатися з декількох активностей і може перемикатися між ними під час виконання програми. Для МД було побудовано діаграму класів (див. рис. 2.7–2.8).

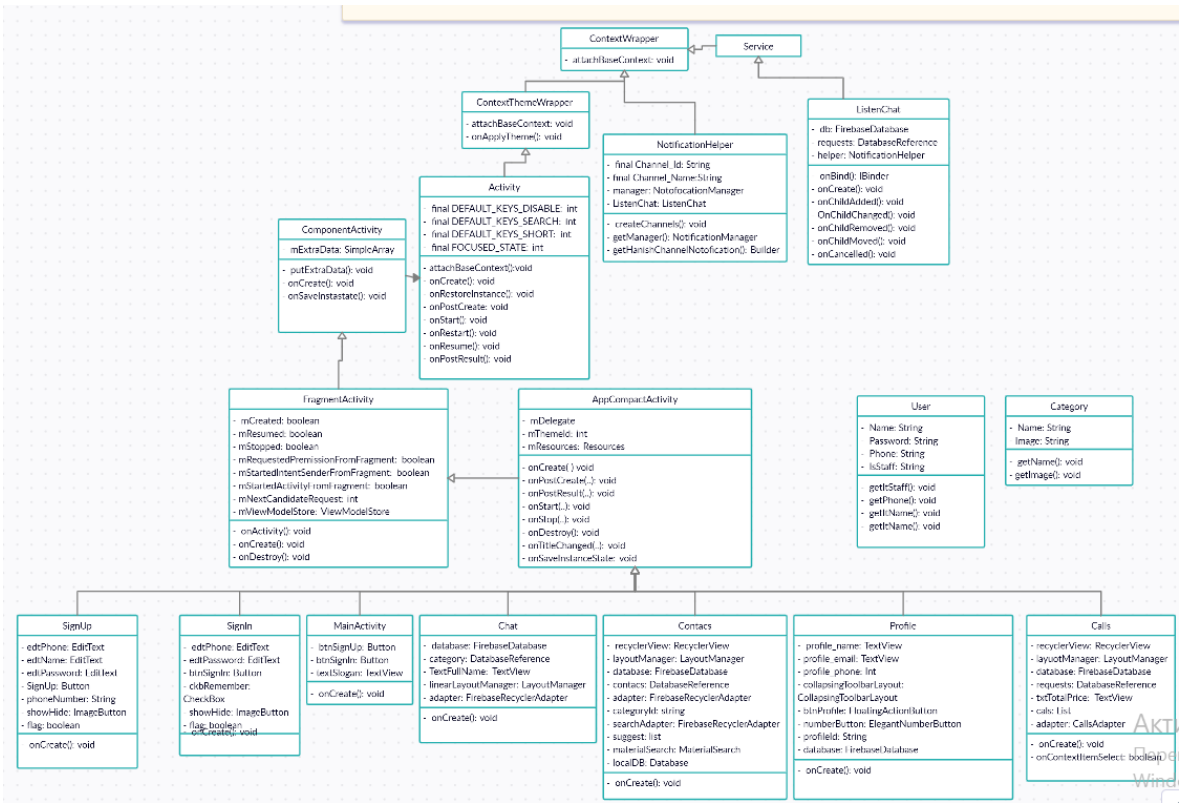

Рисунок 2.7 – Діаграма класів

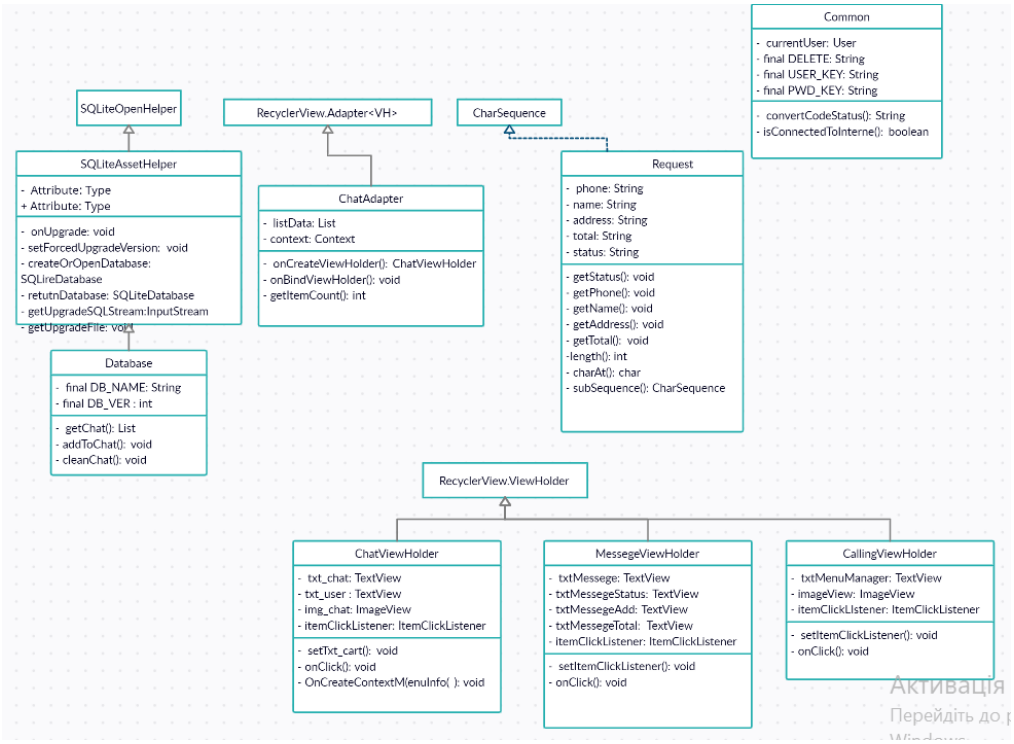

Рисунок 2.8 – Діаграма класів

Список основних активностей:

- MainActivity – активність, містить кілька елементів управління, які дозволяють вибрати ту чи іншу дію. Кнопка «Sign Up», переходить до активності SignUp. Кнопка «Sign In», переходить до активності SignIn.
- SignUp – активність, що містить поля для вводу даних (номер телефону, ім’я, пароль) для реєстрації. Містить кнопку «Sign Up», що підтверджує реєстрацію.
- SignIn – активність, що містить поля для вводу даних (номер телефону, пароль) для входу у додаток. Містить кнопку «Sign In», що підтверджує вхід.
- Chat – основна активність, що містить перелік існуючих діалогів, що може обрати користувач. Елементи представлені у виді фрагментів, при обранні користувачем певного елемента відбувається перехід до обраного пункту.

- Contacs – активність, що містить перелік контактів, що може обрати користувач. Елементі категорій представлені у виді фрагментів, при обранні користувачем певного елемента відбувається перехід до обраного контакту. Також, присутнє пошукове поле.
- Calls – активність, що містить перелік аудіо- та відео дзвінків, які були здійснені за останній час. Присутня кнопка «Call», що дозволяє здійснити дзвінок.
- Profile – активність, що виводить інформацію користувача.
- Messages – активність, що містить перелік компонентів, які представлені текстовим, файловим, аудіо- або відео повідомленням. Елементі представлені у виді фрагментів, при обранні користувачем певного елемента відбувається перехід до обраного компонента.

Список основник класів-сутностей:

- User – клас, що представляє сутність користувача у додатку. Містить гетери та сетери для роботи з даними користувача.
- Category – клас, що представляє собою об’єкт категорій.
- Message – клас, що є представляє собою повідомлення.

2.5 Діаграми взаємодії: послідовності та кооперації

Діаграма послідовності і кооперації є найбільш часто використовуваними діаграмами взаємодії. Діаграма послідовності UML – це діаграма взаємодії, яка детально описує, як виконуються операції. Вона показує взаємодію між об'єктами в спільній роботі. Діаграми послідовності візуально показують порядок взаємодії [6].

Складові додатку, які задіяні у разі відправлення повідомлення (див. рис. 2.9).

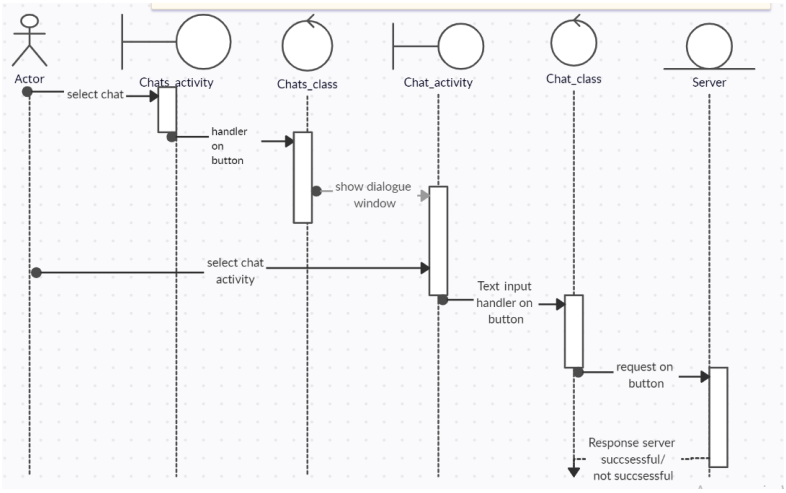

Рисунок 2.9 – Діаграма кооперації сценарію «відправлення повідомлення»

Користувач, у разі натиснення кнопки «Chat», обирає чат. Після вибору чату відбувається відкриття діалогового вікна. Для відправлення повідомлення користувач вводить та натискає кнопку відправки, потім додаток формує запит до сервера для отримання відповіді, щодо вдалої або скасованої операції.

Складові додатку, які задіяні у разі здійснення дзвінка (див. рис. 2.10).

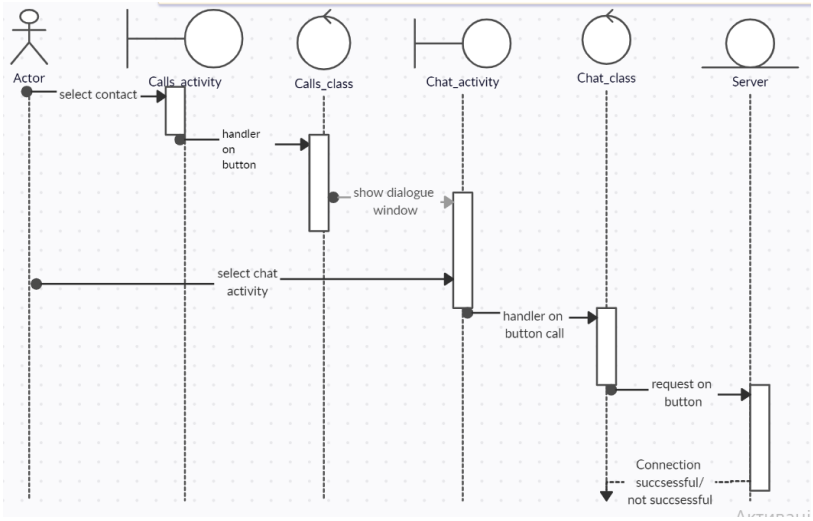

Рисунок 2.10 – Діаграма кооперації сценарію «здійснення дзвінка»

Для здійснення аудіо дзвінка у додатку, Користувач обирає контакт, з пункту Calls, натискає на відповідний контакт.

Додаток відкриває діалогове вікно, Користувач натискає на кнопку здійснення дзвінка.

Додаток формує запит до серверу для з’єднання.

Складові додатку, які задіяні у разі пошуку контакту (див. рис. 2.11).

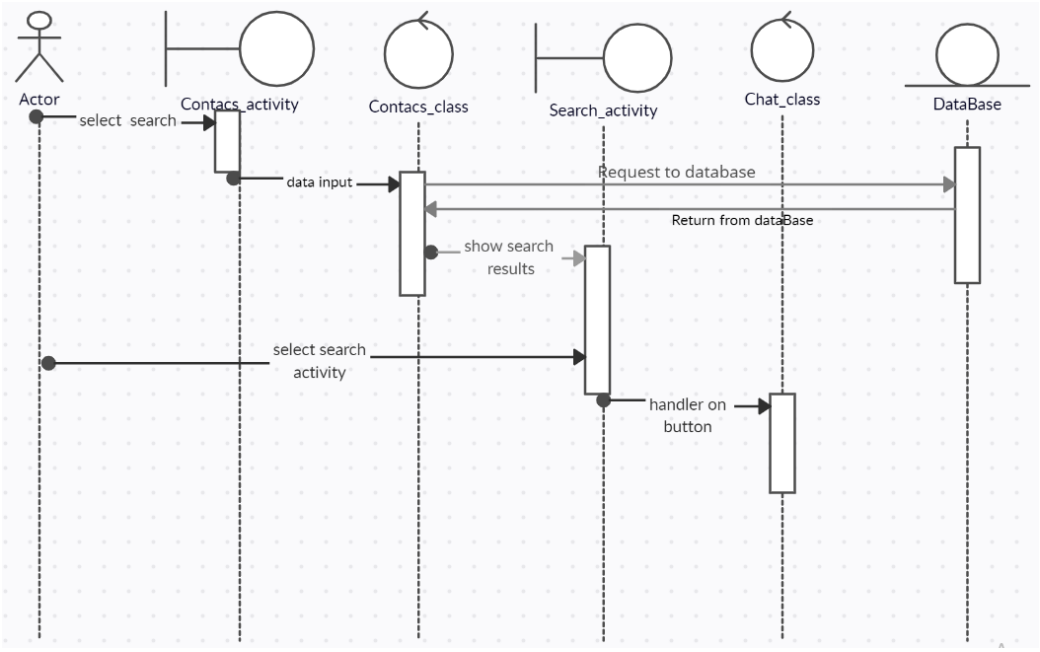

Рисунок 2.11 – Діаграма кооперації сценарію «пошук контакту»

Для пошуку контакту, користувач повинен з пункту контактів натиснути на пошукове поле. Для здійснення пошуку користувач повинен ввести дані та натиснути на пошук. Після цього додаток формує запит до бази даних. Результатом є пошук та виведення результатів.

3 РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Середовище розробки

Середовищем для розробки Android-додатку було обрано Android Studio. Це корисний інструмент для всіх розробників Android, зроблений співпрацею Google та JetBrains. Android Studio – є офіційним IDE для розробки Android [8].

Android – технологічна платформа з власним набором інструментів для її підтримки. Одноразове завантаження включає все необхідне для початку розробки програм. До нього входить:

- комплект розвитку (SDK);
- розподілений інструмент управління джерелами (Git);
- інструмент збирання, що дозволяє керувати бібліотеками та бібліотечними проектами, запускати інструментарій, тестити та створити умовні побудови (Gradle);
- живі макети – подання програми в реальному часі (layout);
- інфраструктура для завантаження багатьох Android екземплярів емулятора.

Всі ці факти значно спрощують процес створення мобільного додатку та роблять Android Studio безумовним лідером серед існуючих середовищ для написання додатків для Android.

3.2 Структура додатку

Програмний додаток для ОС Android складається з набору активностей, кожній з яких відповідає екран програми. Кожна активність представлена класом, що реалізовано на мові Dart, які зберігається в файлі з розширенням

«.dart» [9]. Кожній активності відповідає XML-файл, що містить опис дизайну активності. У XML-файлі міститься у вигляді XML-коду розташування об’єктів, що візуалізуються на екрані. При запуску активності ОС Android автоматично розпізнає розмір екрану мобільного пристрою і виводить контент у відповідністю з розміткою XML-файла [10]. Це дозволяє коректно відображати активність в незалежності від діагоналі екрану пристрою. Детальний код активностей (див.додаток А).

3.3 Розробка інтерфейсу

Інтерфейс (UI) визначає спосіб взаємодії з інформаційною системою. Для успішного просування додатка інтерфейс повинен бути простим та зрозумілим у використанні [7].

Графічний інтерфейс користувача ґрунтується на ієрархії об'єктів View і ViewGroup. Об'єкти View представляють віджети, такі як кнопки або текстові поля. Об'єкти ViewGroup є контейнери для віждетов, керують їхнім розташуванням і компонуванням. Основу обміну інформацією користувача із системою складають діалоги.

Також необхідно враховувати особливості розробки дизайну користувальницького інтерфейсу:

- візуальна ієрархія (найбільш важливі елементи виділяють сильніше, при тому вони повинні знаходитись на одному рівні з іншими елементами);
- лінія погляду користувача (перегляд екрану додатка зліва-направо і зверху-вниз);
- гнучкість дизайну інтерфейсу (на будь-якому пристрої повинно виглядати гармонійно);
- масштабування елементів управління (розташування для максимальної зручності при використанні додатку).

Враховуючи ряд описаних особливостей на які варто звернути увагу і дотримуватися їх при розробці дизайну UI для мобільних додатків, було розроблено схемі усіх екранів з відображенням ключових елементів інтерфейсу.

3.4 Керівництво користувача

При запуску додатка відкривається вікно автентифікації. При натисканні кнопки реєстрації користувача відкривається форма реєстрації (див. рис. 3.1).

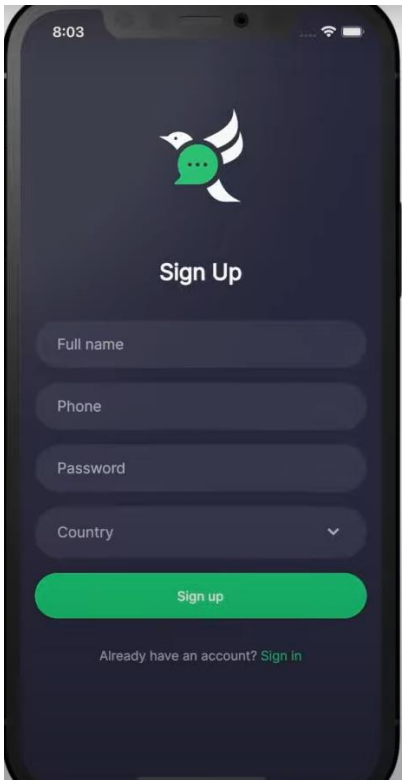

Рисунок 3.1 – Вікно реєстрації

При натисканні кнопки «Sign Up» відбувається реєстрація користувача у системі. Користувач бачить повідомлення про реєстрацію.

При натисканні кнопки входу, відкривається форма входу в додаток. Користувач може пройти автентифікацію шляхом введення номера телефону та пароля у відповідні поля (див. рис. 3.2).

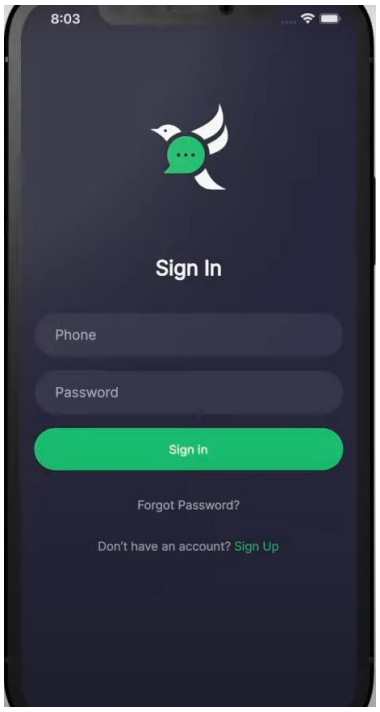

Рисунок 3.2 – Вікно входу

При натисканні кнопки «Sign In», якщо данні були правильно введені, відбувається вхід користувача у систему, інакше користувач бачить повідомлення про помилку. При успішному вході в систему відкривається головне вікно програми (див. рис. 3.3):

- Перегляд чатів;
- Відкриття існуючого чату;
- Додавання нового контакту;
- Навігація по додатку за допомогою кнопок «Chats», «People», «Calls», «Profile».

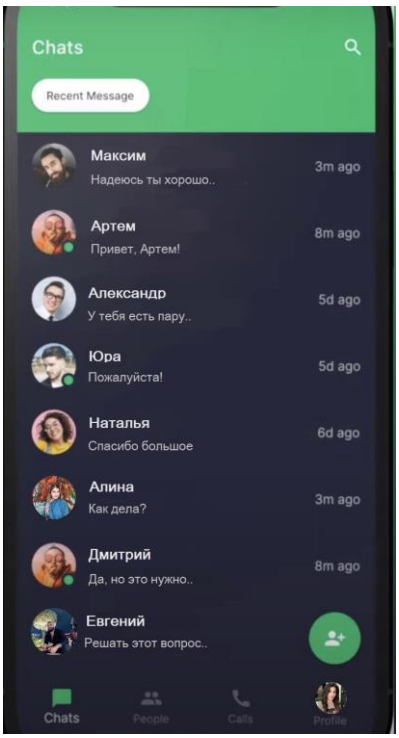

Рисунок 3.3 – Головне вікно програми «Chat»

При натисканні існуючого чату відкривається вікно (див.рис.3.4) з діалогом.

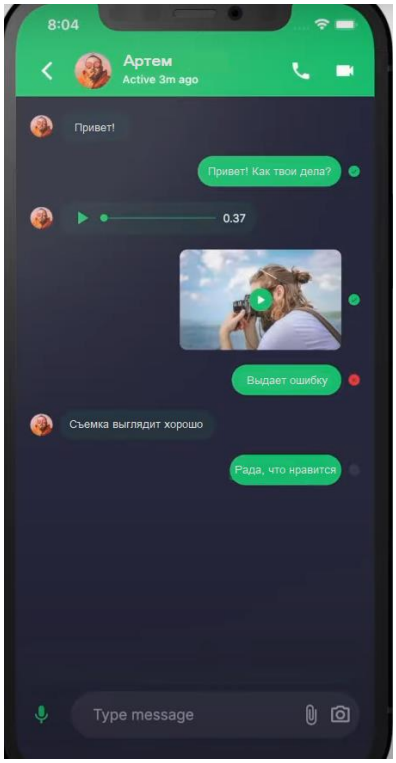

Рис. 3.4 – Діалогове вікно

Щоб надіслати документ, аудіо-повідомлення, відео-повідомлення або перейти до Галереї, щоб вибрати фотографію/відео, користувачу треба вибрати та натиснути відповідну іконку (див. рис. 3.5).

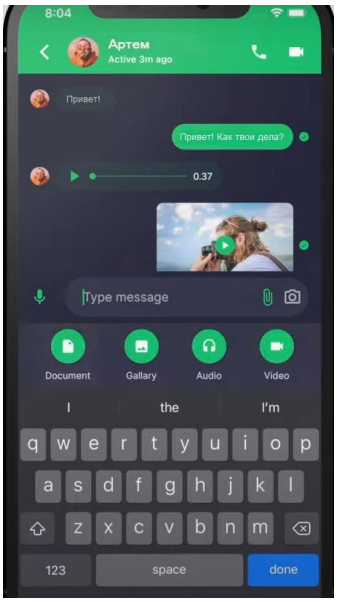

Рисунок 3.5 – Вікно відправлення файлового повідомлення

Для здійснення аудіо- або відео дзвінка через вікно чату, натискається відповідна кнопка на екрані (див. рис. 3.6).

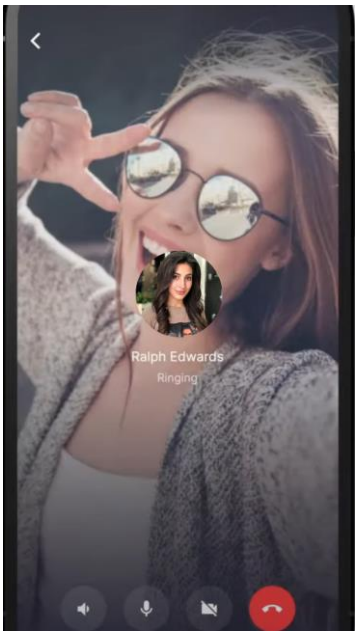

Рисунок 3.6 – Здійснення відео дзвінка

Для перегляду контактів користувача, треба перейти до розділу «People» (див. рис. 3.7).

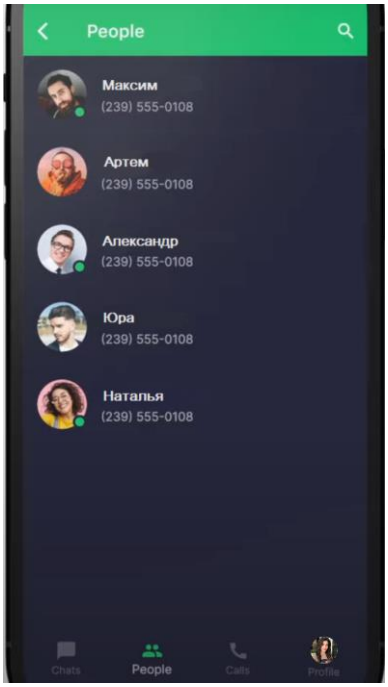

Рисунок 3.7 – Розділ «People»

При натисканні пошукового рядку (див. рис. 3.8) та введення назви, здійснюється пошук та виведення його результатів.

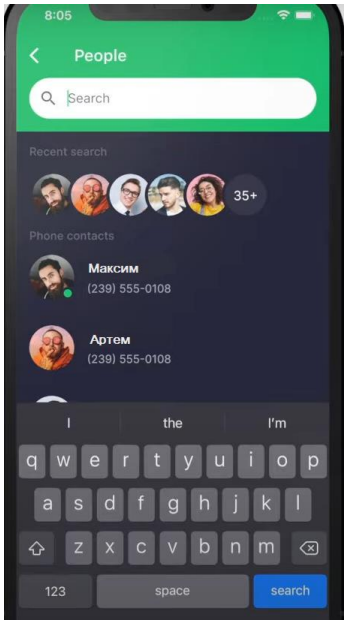

Рисунок 3.8 – Пошуковий рядок

Для отримання інформації, що містить останні аудіо- та відео дзвінки, треба перейти до розділу «Calls» (див. рис.3.8).

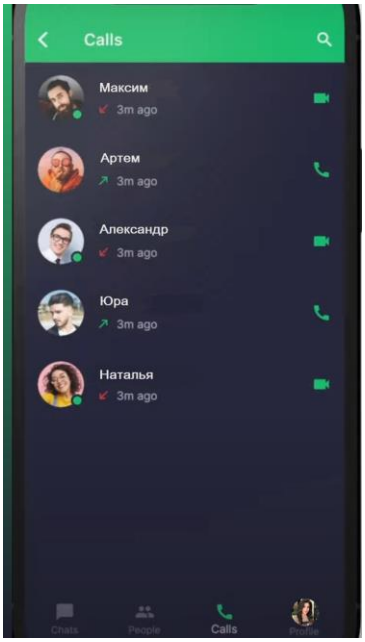

Рисунок 3.8 – Розділ «Calls»

При натисканні кнопки «Profile» відкривається вікно (див. рис.3.9), що містить інформацію щодо користувача. Кнопка «Chat» поверне користувача на головне вікно з існуючими чатами (див. рис.3.3).

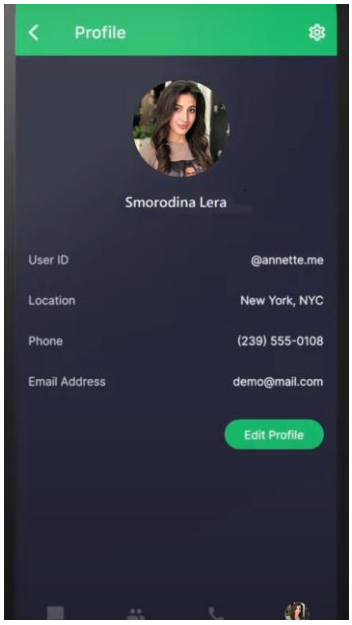

Рисунок 3.9 – Розділ «Profile»

Якщо в налаштуваннях буде натиснута кнопка «Sign out», буде виконано вихід із системи та відкриється вікно автентифікації (див. рис.3.10).

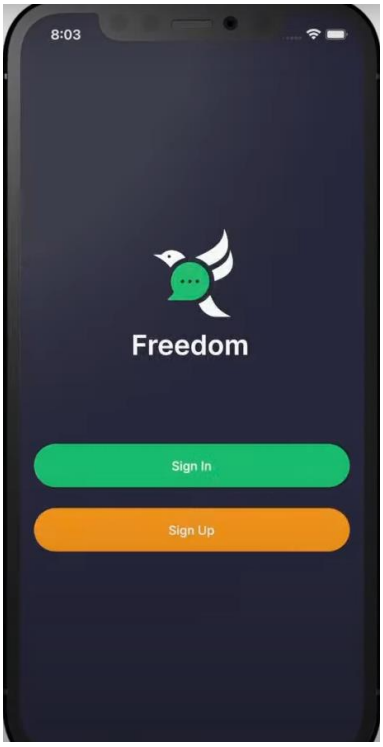

Рисунок 3.10 – Вікно автентифікації

ВИСНОВКИ

У процесі розробки Android додатку сервісу обміну миттєвими повідомленнями було проведено дослідження предметної області, спроектований та розроблений інтерфейс користувача, запрограмована логіка додатку.

Для створення додатку була обрана мова програмування Dart та застосовані Firebase, SQLite технології.

Розроблена система має простий та зрозумілий користувачеві інтерфейс. Додаток представляє собою сервіс, який надає можливість обміну миттєвими повідомленнями. Завдяки досягнутим цілям додаток передбачає можливість реєстрації та авторизації, обмін миттєвими повідомленнями, аудіо- та відео повідомленнями, здійснювати аудіо- та відео дзвінки.

У подальшому розвитку додаток може бути розширений. Таким чином мета кваліфікаційної роботи досягнута.

ПЕРЕЛІК ПОСИЛАНЬ

1. Коцюба И. Ю., Чунаев А. В, Шиков А. Н. Основи проектування інформаційних систем, Санкт-Петербург : Университет ИТМО, 2015. 206 с.

2. Mayer R. J. Information Integration for Concurrent Engineering (IICE) — IDEF4 Object-oriented Design Method Report, version 2.0. KBSI Co., 1995. 634 p.

3. Брайан Х. Программирование под Android. Для профессионалов, Санкт-Петербург : BBM, 2013. 592 с.

4. Aliferi C. Android Programming Cookbook: Kick-start your Android Projects, Code Geeks, 2018. 143 p.

5. Darwin F. Android Cookbook: Problems and Solutions for Android Developers 2nd Edition, O'Reilly Media, 2017. 838 p.

6. Moore M., Moore M. Handbook of Distance Education, 2003. 436 p.

7. Stankov S., Žitko B. and Grubišić A. Ontology as a Foundation for Knowledge Evaluation in Intelligent E-learning Systems. *AIED'05 Workshop SW-EL'05: Applications of Semantic Web Technologies for E-Learning*: Papers of 12th International Conference on Artificial Intelligence in Education (AIED 2005). Amsterdam, 2005. P. 361–367.

8. Moore K., Katz M., Ngo V. Flutter Apprentice. England, 2021. 768 p.

9. Martins F. Flutter and Dart the Complete Guide: Create Cross-Platform Mobile Apps With Google's Latest Open-Source SDK Through Flutter and Dart. En, 2021. 345 p.

10. Simone A., Kayfitz B. Google Flutter 2 Cookbook: Over 100 proven techniques and solutions to mobile development with Flutter and Dart. En, 2020. 587 p.

Додаток А

Програмна реалізація

Проект доступний за посиланням:
https://github.com/ValerySmorodina/ChatApp

Детальний код основних активностей :

1.Main Activity

```
import 'package:animations/animations.dart';
import 'package:chat/screens/calls/calls_hsitory_screen.dart';
import 'package:chat/screens/chats/chats_screen.dart';
import 'package:chat/screens/contacts/contacts_screen.dart';
import 'package:chat/screens/profile/profile_screen.dart';
import 'package:flutter/material.dart';

class MainScreen extends StatefulWidget {
  @override
  _MainScreenState createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  int pageIndex = 0;

  List<Widget> pageList = <Widget>[
    ChatsScreen(),
    ContactsScreen(),
    CallsHistoryScreen(),
    ProfileScreen(),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: PageTransitionSwitcher(
        transitionBuilder: (
          Widget child,
          Animation<double> animation,
          Animation<double> secondaryAnimation,
        ) {
          return FadeThroughTransition(
            animation: animation,
            secondaryAnimation: secondaryAnimation,
            child: child,
          );
        },
        child: pageList[pageIndex],
      ),
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        currentIndex: pageIndex,
        onTap: (value) {
          setState(() {
            pageIndex = value;
          });
        },
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.messenger), label: "Chats"),
          BottomNavigationBarItem(icon: Icon(Icons.people), label: "People"),
          BottomNavigationBarItem(icon: Icon(Icons.call), label: "Calls"),
          BottomNavigationBarItem(
            icon: CircleAvatar(
              radius: 14,
              backgroundImage: AssetImage("assets/images/user_2.png"),
            ),
            label: "Profile",
          ),
        ],
```

```
    ),
  );
```

2.SignUp Activity

```
import 'package:chat/screens/auth/sign_in_screen.dart';
import 'package:chat/screens/auth/verification_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:form_field_validator/form_field_validator.dart';

import '../../constants.dart';
import '../../components/primary_button.dart';

class SignUpScreen extends StatelessWidget {
 final _formKey = GlobalKey<FormState>();
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   body: SafeArea(
    child: LayoutBuilder(builder: (context, constraints) {
     return SingleChildScrollView(
      padding: const EdgeInsets.symmetric(horizontal: defaultPadding),
      child: Column(
       children: [
        SizedBox(height: constraints.maxHeight * 0.08),
        SvgPicture.asset(
         MediaQuery.of(context).platformBrightness == Brightness.dark
           ? logoDarkTheme
           : logoLightTheme,
        ),
        SizedBox(height: constraints.maxHeight * 0.08),
        Text(
         "Sign Up",
         style: Theme.of(context)
           .textTheme
           .headline5!
           .copyWith(fontWeight: FontWeight.bold),
        ),
        SizedBox(height: constraints.maxHeight * 0.05),
        Form(
         key: _formKey,
         child: Column(
          children: [
           TextFormField(
            validator: RequiredValidator(errorText: requiredField),
            decoration: InputDecoration(hintText: 'Full name'),
            onSaved: (name) {
             // Save it
            },
           ),
           SizedBox(height: defaultPadding),
           TextFormField(
            validator: RequiredValidator(errorText: requiredField),
            decoration: InputDecoration(hintText: 'Phone'),
            keyboardType: TextInputType.phone,
            onSaved: (phone) {
             // Save it
            },
           ),
           Padding(
            padding: const EdgeInsets.symmetric(
              vertical: defaultPadding),
            child: TextFormField(
             validator: passwordValidator,
             decoration: InputDecoration(hintText: 'Password'),
             obscureText: true,
             onSaved: (passaword) {
              // Save it
             },
            ),
           ),
           DropdownButtonFormField(
            items: countries,
            icon: Icon(Icons.expand_more),
            onSaved: (country) {
             // save it
            },
            onChanged: (value) {},
            decoration: InputDecoration(hintText: 'Country'),
           ),
           Padding(
            padding: const EdgeInsets.symmetric(
              vertical: defaultPadding),
```

```
                child: PrimaryButton(
                  text: 'Sign up',
                  press: () {
                    if
(_formKey.currentState!.validate()) {
_formKey.currentState!.save();
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) =>
VerificationScreen(),
                        ),
                      );
                    }
                  },
                ),
              ),
              TextButton(
                onPressed: () =>
Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(
                    builder: (context) =>
SignInScreen(),
                  ),
                ),
                child: Text.rich(
                  TextSpan(
                    text: "Already have an
account? ",
                    children: [
                      TextSpan(
                        text: "Sign in",
                        style: TextStyle(
                          color:
Theme.of(context).primaryColor),
                      ),
                    ],
                  ),
                  style:
Theme.of(context).textTheme.bodyText2!.copy
With(
                    color: Theme.of(context)
                      .textTheme
                      .bodyText1!
                      .color!
                      .withOpacity(0.64),
                  ),
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}

// only for demo
List<DropdownMenuItem<String>>? countries
= [
 "Ukraine",
 "Russia",
 'Poland',
].map<DropdownMenuItem<String>>((String
value) {
 return DropdownMenuItem<String>(value:
value, child: Text(value));
}).toList();
```

3. SignIn Activity

```
import
'package:chat/components/primary_button.dart';
import 'package:chat/constants.dart';
import
'package:chat/screens/auth/forgot_password_scr
een.dart';
import
'package:chat/screens/auth/sign_up_screen.dart';
import
'package:chat/screens/chats/chats_screen.dart';
import
'package:chat/screens/main/main_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import
'package:form_field_validator/form_field_valida
tor.dart';

class SignInScreen extends StatelessWidget {
 final _formKey = GlobalKey<FormState>();
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   body: SafeArea(
    child: LayoutBuilder(
     builder: (context, constraints) {
      return SingleChildScrollView(
       padding: const
EdgeInsets.symmetric(horizontal:
defaultPadding),
```

```
          child: Column(
            children: [
              SizedBox(height:
constraints.maxHeight * 0.1),
              SvgPicture.asset(

MediaQuery.of(context).platformBrightness ==
Brightness.dark
                  ? logoDarkTheme
                  : logoLightTheme,
              ),
              SizedBox(height:
constraints.maxHeight * 0.1),
              Text(
                "Sign In",
                style: Theme.of(context)
                    .textTheme
                    .headline5!
                    .copyWith(fontWeight:
FontWeight.bold),
              ),
              SizedBox(height:
constraints.maxHeight * 0.05),
              Form(
                key: _formKey,
                child: Column(
                  children: [
                    TextFormField(
                      validator:
                          RequiredValidator(errorText:
requiredField),
                      decoration:
InputDecoration(hintText: 'Phone'),
                      keyboardType:
TextInputType.phone,
                      onSaved: (phone) {
                        // Save it
                      },
                    ),
                    Padding(
                      padding: const
EdgeInsets.symmetric(
                          vertical: defaultPadding),
                      child: TextFormField(
                        validator: passwordValidator,
                        obscureText: true,
                        decoration:
InputDecoration(hintText: 'Password'),
                        onSaved: (passaword) {
                          // Save it
                        },
                      ),
                    ),
                    PrimaryButton(
                      text: 'Sign in',
                      press: () {
                        if
(_formKey.currentState!.validate()) {

_formKey.currentState!.save();
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(
                              builder: (context) =>
MainScreen(),
                            ),
                          );
                        }
                      },
                    ),
                    SizedBox(height:
defaultPadding),
                    TextButton(
                      onPressed: () =>
Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) =>
ForgotPasswordScreen(),
                        ),
                      ),
                      child: Text(
                        'Forgot Password?',
                        style:

Theme.of(context).textTheme.bodyText2!.copy
With(
                            color: Theme.of(context)
                                .textTheme
                                .bodyText1!
                                .color!
                                .withOpacity(0.64),
                          ),
                    ),
                    TextButton(
                      onPressed: () {
                        Navigator.pushReplacement(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
SignUpScreen(),
                          ),
                        );
                      },
                      child: Text.rich(
                        TextSpan(
                          text: "Don’t have an account?
",
                          children: [
                            TextSpan(
                              text: "Sign Up",
                              style: TextStyle(
                                color:
```

```
Theme.of(context).primaryColor),
                    ),
                  ],
                ),
                style:

Theme.of(context).textTheme.bodyText2!.copy
With(
                    color: Theme.of(context)
                        .textTheme
                        .bodyText1!
                        .color!
                        .withOpacity(0.64),
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ],
  ),
);
  },
 ),
),
);
 }
}
```

4. Chat Activity

```
import 'package:chat/constants.dart';
import
'package:chat/screens/search/search_screen.dart';
import 'package:flutter/material.dart';

import 'components/body.dart';

class ChatsScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: buildAppBar(context),
   body: Body(),
   floatingActionButton:
FloatingActionButton(
    onPressed: () {},
    backgroundColor: primaryColor,
    child: Icon(
     Icons.person_add_alt_1,
     color: Colors.white,
    ),
   ),
  );
 }

 AppBar buildAppBar(BuildContext context) {
  return AppBar(
   automaticallyImplyLeading: false,
   title: Text("Chats"),
   actions: [
    IconButton(
     icon: Icon(Icons.search),
     onPressed: () {
      Navigator.push(
       context,
       MaterialPageRoute(
        builder: (context) => SearchScreen(),
       ));
     },
    ),
   ],
  );
 }
}
```

5. Message Activity

```
import 'audio_message.dart';
import 'text_message.dart';
import 'video_message.dart';

class Message extends StatelessWidget {
 const Message({
  Key? key,
  required this.message,
 }) : super(key: key);

 final ChatMessage message;

 @override
 Widget build(BuildContext context) {
  Widget messageContaint(ChatMessage
message) {
   switch (message.messageType) {
    case ChatMessageType.text:
     return TextMessage(message: message);
    case ChatMessageType.audio:
     return AudioMessage(message: message);
    case ChatMessageType.video:
     return VideoMessage();
    default:
     return SizedBox();
   }
  }
```

```
    return Padding(
      padding: const EdgeInsets.only(top: defaultPadding),
      child: Row(
        mainAxisAlignment:
            message.isSender ? MainAxisAlignment.end : MainAxisAlignment.start,
        children: [
          if (!message.isSender) ...[
            CircleAvatar(
              radius: 12,
              backgroundImage: AssetImage("assets/images/user_2.png"),
            ),
            SizedBox(width: defaultPadding / 2),
          ],
          messageContaint(message),
          if (message.isSender) MessageStatusDot(status: message.messageStatus)
        ],
      ),
    );
  }
}

class MessageStatusDot extends StatelessWidget {
  final MessageStatus? status;

  const MessageStatusDot({Key? key, this.status}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    Color dotColor(MessageStatus status) {
      switch (status) {
        case MessageStatus.not_sent:
          return errorColor;
        case MessageStatus.not_view:
          return Theme.of(context).textTheme.bodyText1!.color!.withOpacity(0.1);
        case MessageStatus.viewed:
          return primaryColor;
        default:
          return Colors.transparent;
      }
    }

    return Container(
      margin: EdgeInsets.only(left: defaultPadding / 2),
      height: 12,
      width: 12,
      decoration: BoxDecoration(
        color: dotColor(status!),
        shape: BoxShape.circle,
      ),
      child: Icon(
        status == MessageStatus.not_sent ? Icons.close : Icons.done,
        size: 8,
        color: Theme.of(context).scaffoldBackgroundColor,
      ),
    );
  }
}
```

6. CallingAudio Activity

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

import 'components/call_bg.dart';
import 'components/call_option.dart';

class AudioCallingScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      extendBodyBehindAppBar: true,
      appBar: AppBar(
        backgroundColor: Colors.transparent,
      ),
      body: CallBg(
        image: Image.asset(
          "assets/images/call_bg.png",
          fit: BoxFit.cover,
        ),
        child: SafeArea(
          child: Column(
            children: [
              Spacer(),
              CircleAvatar(
                radius: 50,
                backgroundImage: AssetImage("assets/images/user_2.png"),
              ),
              SizedBox(height: defaultPadding),
              Text(
                "Ralph Edwards",
                style: Theme.of(context)
                    .textTheme
                    .subtitle1!
                    .copyWith(color: Colors.white),
              ),
              SizedBox(height: defaultPadding / 2),
              Text(
                "Ringing",
                style: TextStyle(color:
```

```
Colors.white70),
          ),
          Spacer(),
          Padding(
            padding: const EdgeInsets.symmetric(
              horizontal: defaultPadding * 2,
vertical: defaultPadding),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
              children: [
                CallOption(
                  icon: Icon(Icons.volume_down),
                  press: () {},
                ),
                CallOption(
                  icon: Icon(Icons.mic),
                  press: () {},
                ),
                CallOption(
                  icon: Icon(Icons.videocam_off),
                  press: () {},
                ),
                CallOption(
                  icon: Icon(Icons.call_end),
                  color: errorColor,
                  press: () {},
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

7. CallingVideo Activity

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

import 'components/call_bg.dart';
import 'components/call_option.dart';

class VideoCallingScreen extends
StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   extendBodyBehindAppBar: true,
   appBar: AppBar(
    backgroundColor: Colors.transparent,
   ),
   body: CallBg(
    image: Image.asset(
     "assets/images/call_bg.png",
     fit: BoxFit.cover,
    ),
    child: SafeArea(
     child: Column(
      children: [
       Spacer(),
       // Preview of my camera
       Align(
        alignment: Alignment.centerRight,
        child: Container(
         margin:
EdgeInsets.all(defaultPadding),
         height: 120,
         width: 96,
         decoration: BoxDecoration(
          borderRadius:
           const
BorderRadius.all(Radius.circular(defaultPadding
)),
          boxShadow: [
           BoxShadow(
            offset: Offset(0, 24),
            blurRadius: 40,
            color: Colors.black38,
           ),
          ],
          image: DecorationImage(
           fit: BoxFit.cover,
           image:
AssetImage("assets/images/call_bg.png"),
          ),
         ),
        ),
       ),
       Padding(
        padding: const EdgeInsets.symmetric(
          horizontal: defaultPadding * 2,
vertical: defaultPadding),
        child: Row(
         mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
         children: [
          CallOption(
           icon: Icon(Icons.volume_down),
           press: () {},
          ),
          CallOption(
           icon: Icon(Icons.mic),
           press: () {},
          ),
          CallOption(
           icon: Icon(Icons.videocam_off),
           press: () {},
          ),
```

```
              CallOption(
                icon: Icon(Icons.call_end),
                color: errorColor,
                press: () {},
              ),
            ],
          ),
        ),
      ],
    ),
  ),
 );
 }
}
```

8. Calls Activity

```
import
'package:chat/components/circle_avatar_with_active_indicator.dart';
import
'package:chat/screens/search/components/body.dart';
import 'package:flutter/material.dart';

import '../../../constants.dart';

class CallHistoryCard extends StatelessWidget {
 const CallHistoryCard({
  Key? key,
  required this.name,
  required this.time,
  required this.isActive,
  required this.isVideoCall,
  required this.isOutgoingCall,
  required this.image,
  required this.press,
 }) : super(key: key);

 final String name, time, image;
 final bool isActive, isVideoCall,
isOutgoingCall;
 final VoidCallback press;

 @override
 Widget build(BuildContext context) {
  return ListTile(
   contentPadding: EdgeInsets.symmetric(
    horizontal: defaultPadding,
    vertical: defaultPadding / 2,
   ),
   onTap: () {},
   leading: CircleAvatarWithActiveIndicator(
    image: image,
    isActive: isActive,
    radius: 28,
   ),
   title: Text(name),
   subtitle: Padding(
    padding: const
EdgeInsets.symmetric(vertical: defaultPadding /
2),
    child: Row(
     children: [
      Icon(
       isOutgoingCall ? Icons.north_east :
Icons.south_west,
       size: 16,
       color:
        isOutgoingCall ?
Theme.of(context).primaryColor : errorColor,
      ),
      SizedBox(width: defaultPadding / 2),
      Text(
       time,
       style: TextStyle(
        color: Theme.of(context)
          .textTheme
          .bodyText1!
          .color!
          .withOpacity(0.64),
       ),
      ),
     ],
    ),
   ),
   trailing: Icon(
    isVideoCall ? Icons.videocam : Icons.call,
    color: Theme.of(context).primaryColor,
   ),
  );
 }
}
```

9. Contacs Activity

```
import
'package:chat/components/circle_avatar_with_active_indicator.dart';
import 'package:flutter/material.dart';

import '../../../constants.dart';

class ContactCard extends StatelessWidget {
 const ContactCard({
  Key? key,
  required this.name,
  required this.number,
  required this.image,
  required this.isActive,
  required this.press,
```

```
  }) : super(key: key);

  final String name, number, image;
  final bool isActive;
  final VoidCallback press;

  @override
  Widget build(BuildContext context) {
    return ListTile(
      contentPadding: EdgeInsets.symmetric(
        horizontal: defaultPadding, vertical:
defaultPadding / 2),
      onTap: () {},
      leading: CircleAvatarWithActiveIndicator(
        image: image,
        isActive: isActive,
        radius: 28,
      ),
      title: Text(name),
      subtitle: Padding(
        padding: const EdgeInsets.only(top:
defaultPadding / 2),
        child: Text(
          number,
          style: TextStyle(
            color:

Theme.of(context).textTheme.bodyText1!.color!
.withOpacity(0.64),
          ),
        ),
      ),
    );
  }
}
```

10. Search Activity

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

import
'../../../components/recent_search_contacts.dart';

class Body extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      padding: EdgeInsets.symmetric(vertical:
defaultPadding),
      child: Column(
        children: [
          RecentSearchContacts(),
          SizedBox(height: defaultPadding),
          // you can show suggested style for search
result
          SuggestedContacts()
        ],
      ),
    );
  }
}

class SuggestedContacts extends
StatelessWidget {
  const SuggestedContacts({
    Key? key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const
EdgeInsets.symmetric(horizontal:
defaultPadding),
          child: Text(
            "Suggested",
            style:
Theme.of(context).textTheme.subtitle2!.copyWi
th(
                  color: Theme.of(context)
                      .textTheme
                      .subtitle2!
                      .color!
                      .withOpacity(0.32),
                ),
          ),
        ),
        SizedBox(height: defaultPadding),
        ...List.generate(
          demoContactsImage.length,
          (index) => ListTile(
            contentPadding: EdgeInsets.symmetric(
                horizontal: defaultPadding, vertical:
defaultPadding / 2),
            leading: CircleAvatar(
              radius: 24,
              backgroundImage:
AssetImage(demoContactsImage[index]),
            ),
            title: Text("Jenny Wilson"),
            onTap: () {},
          ),
        ),
      ],
    );
  }
}
```

```
final List<String> demoContactsImage =
    List.generate(5, (index) =>
"assets/images/user_${index + 1}.png");
```

11. Profile Activity

```
import
'package:chat/components/primary_button.dart';
import 'package:chat/constants.dart';
import
'package:chat/screens/profile/edit_profile_screen
.dart';
import 'package:flutter/material.dart';

import 'components/info.dart';
import 'components/profile_pic.dart';

class ProfileScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Profile"),
        actions: [
          IconButton(
            icon: Icon(Icons.settings_outlined),
            onPressed: () {},
          )
        ],
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.symmetric(horizontal:
defaultPadding),
        child: Column(
          children: [
            ProfilePic(image:
"assets/images/user_2.png"),
            Text(
              "Annette Black",
              style:
Theme.of(context).textTheme.headline6,
            ),
            Divider(height: defaultPadding * 2),
            Info(
              infoKey: "User ID",
              info: "@annette.me",
            ),
            Info(
              infoKey: "Location",
              info: "New York, NYC",
            ),
            Info(
              infoKey: "Phone",
              info: "(239) 555-0108",
            ),
            Info(
              infoKey: "Email Address",
              info: "demo@mail.com",
            ),
            SizedBox(height: defaultPadding),
            Align(
              alignment: Alignment.centerRight,
              child: SizedBox(
                width: 120,
                child: PrimaryButton(
                  padding: EdgeInsets.all(5),
                  text: "Edit Profile",
                  press: () => Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
EditProfileScreen(),
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

import 'package:flutter/material.dart';

import '../../../constants.dart';

class Info extends StatelessWidget {
  const Info({
    Key? key,
    required this.infoKey,
    required this.info,
  }) : super(key: key);

  final String infoKey, info;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const
EdgeInsets.symmetric(vertical: defaultPadding),
      child: Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: [
          Text(
            infoKey,
            style: TextStyle(
              color: Theme.of(context)
                  .textTheme
                  .bodyText1!
                  .color!
```

```
                .withOpacity(0.8),
            ),
          ),
          Text(info),
        ],
      ),
    );
  }
}
```

12. EditProfile Activity

```
import
'package:chat/components/primary_button.dart';
import 'package:chat/constants.dart';
import
'package:chat/screens/search/components/body.d
art';
import 'package:flutter/material.dart';

import 'components/profile_pic.dart';
import 'components/user_info_edit_field.dart';

class EditProfileScreen extends StatelessWidget
{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Edit Profile"),
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.symmetric(horizontal:
defaultPadding),
        child: Column(
          children: [
            ProfilePic(
              image: demoContactsImage[0],
              imageUploadBtnPress: () {},
            ),
            Divider(),
            Form(
              child: Column(
                children: [
                  UserInfoEditField(
                    text: "Name",
                    child: TextFormField(
                      initialValue: "Annette Black",
                    ),
                  ),
                  UserInfoEditField(
                    text: "Email",
                    child: TextFormField(
                      initialValue:
"annette@gmail.com",
                    ),
                  ),
                  UserInfoEditField(
                    text: "Phone",
                    child: TextFormField(
                      initialValue: "(316) 555-0116",
                    ),
                  ),
                  UserInfoEditField(
                    text: "Address",
                    child: TextFormField(
                      initialValue: "New York, NVC",
                    ),
                  ),
                  UserInfoEditField(
                    text: "Old Password",
                    child: TextFormField(
                      obscureText: true,
                      initialValue: "demopass",
                      decoration: InputDecoration(
                        suffixIcon: Icon(
                          Icons.visibility_off,
                          size: 20,
                        ),
                      ),
                    ),
                  ),
                  UserInfoEditField(
                    text: "New Password",
                    child: TextFormField(
                      decoration:
InputDecoration(hintText: "New Password"),
                    ),
                  ),
                ],
              ),
            ),
            SizedBox(height: defaultPadding),
            Row(
              mainAxisAlignment:
MainAxisAlignment.end,
              children: [
                SizedBox(
                  width: 100,
                  child: PrimaryButton(
                    padding: EdgeInsets.all(5),
                    color: Theme.of(context)
                        .textTheme
                        .bodyText1!
                        .color!
                        .withOpacity(0.08),
                    text: "Cancel",
                    press: () {},
                  ),
                ),
                SizedBox(width: defaultPadding),
                SizedBox(
                  width: 130,
```

```
            UserInfoEditField(
              text:          "Old          Password",
              child:                 TextFormField(
                obscureText:                    true,
                initialValue:            "demopass",
                decoration:          InputDecoration(
                  suffixIcon:                  Icon(
                    Icons.visibility_off,
                    size:                         20,
                  ),
                ),
              ),
            ),
            UserInfoEditField(
              text:          "New          Password",
              child:                 TextFormField(
                decoration:
InputDecoration(hintText:   "New   Password"),
              ),
            ),
          ],
        ),
      ),
      SizedBox(height:          defaultPadding),
      Row(
        mainAxisAlignment:
MainAxisAlignment.end,
        children:                                [
          SizedBox(
            width:                              100,
            child:                  PrimaryButton(
              padding:             EdgeInsets.all(5),
              color:               Theme.of(context)
                  .textTheme
                  .bodyText1!
                  .color!
                  .withOpacity(0.08),
child:                               PrimaryButton(
              padding:             EdgeInsets.all(5),
              text:          "Save          Update",
              press:               ()              {},
            ),
          )
```