МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка месенджера голосових повідомлень на платформі Android**

Виконав: студент 4 курсу, групи ІПЗ-1219бд
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
В.В Горянець
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н., доцент І.А. Скрипник
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення

Рівень вищої освіти _____перший (бакалавський)_____

Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)

Освітня програма ___Програмне забезпечення систем____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т. В. Критська
"_01_" __березня_____2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Горянцю Віталію Валерійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи Розробка месенджера голосових повідомлень на платформі Android

керівник роботи Скрипник Ірина Анатоліївна, к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 29.12.2022 № 1893-с

2. Строк подання студентом кваліфікаційної роботи 14.06.2023

3. Вихідні дані бакалаврської роботи

- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення мобільних застосунків;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- тестування програмної системи та розробка висновків та пропозицій.

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

слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	01.03-10.03.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.03-12.03.23	виконано
3	Аналіз існуючих методів рішення	13.03-14.03.23	виконано
4	Аналіз методів розробки андроїд застосунків	15.03-20.03.23	виконано
5	Аналіз засобів розробки андроїд застосунків	21.03-26.03.23	виконано
6	Узгодження подальших дій з науковим керівником	27.03-28.03.23	виконано
7	Проєктування андроїд застосунку	29.03-05.04.23	виконано
8	Програмна реалізація застосунку	06.04-16.04.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	17.04-19.04.23	виконано
10	Реалізація користувацького інтерфейсу для веб-додатку	20.04-01.05.23	виконано
11	Тестування застосунку	02.05-7.05.23	виконано
12	Оформлення звіту	8.05-15.05.23	виконано
13	Оформлення презентації.	16.05-20.05.23	виконано

Студент _____ В.В. Горянець
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 73

Рисунків – 19

Джерел – 7.

Горянець В.В. Розробка месенджера голосових повідомлень на платформі Android: кваліфікаційна робота бакалавра спеціальності 121 "Інженерія програмного забезпечення" / наук. керівник доцент, к.ф.-м.н., І.А. Скрипник. Запоріжжя: ЗНУ. 2023. 73 с.

Мобільні месенджери є одними з найпопулярніших інструментів спілкування у сучасному світі. Зростання популярності голосових повідомлень створює потребу в розробці спеціалізованого месенджера для обміну голосовими повідомленнями на платформі Android. Однак існуючі додатки не завжди задовольняють всі потреби користувачів, тому дослідження та розробка нового месенджера є актуальною проблемою.

Метою даної кваліфікаційної роботи є розробка месенджера голосових повідомлень на платформі Android з використанням сучасних технологій та практик програмування. Для досягнення цієї мети були визначені основні функціональні вимоги до додатку, проведений аналіз існуючих рішень та обґрунтовано вибір необхідних технологій.

У результаті дослідження був розроблений месенджер, який забезпечить користувачам можливість обміну голосовими повідомленнями з використанням інтуїтивного інтерфейсу та стабільної роботи. Додаток був реалізований з використанням мови програмування Kotlin, фреймворку Android SDK, Firebase platfrom та інших сучасних технологій.

Отриманий месенджер може бути використаний як основа для подальшого розвитку та комерційного застосування у сфері мобільних комунікацій. Результати дослідження можуть бути корисними для розробників мобільних додатків та спеціалістів у галузі програмного забезпечення.

Ключові слова: *месенджер, голосові повідомлення, Android, Kotlin, програмування, мобільні додатки.*

ANNOTATION

Pages — 73.

Drawings — 19.

Sources — 7.

Horyanets V.V. Development of a Voice Message Messenger on the Android Platform: Bachelor's qualification work in the specialty 121 "Software Engineering" / scientific. head I.A. Skrypnyk. Zaporizhzhia: ZNU. 2023. ### p.

Mobile messengers are among the most popular communication tools in the modern world. The increasing popularity of voice messages creates a need for the development of a specialized messenger for exchanging voice messages on the Android platform. However, existing applications do not always meet all user needs, making the research and development of a new messenger a relevant problem.

The aim of this thesis is to develop a voice message messenger on the Android platform using modern technologies and programming practices. To achieve this goal, the main functional requirements for the application were identified, existing solutions were analyzed, and the selection of necessary technologies was justified.

As a result of the research, a messenger will be developed that provides users with the ability to exchange voice messages with an intuitive interface and stable performance. The application will be implemented using the Kotlin programming language, Android SDK framework, Firebase platform, and other modern technologies.

The obtained messenger can serve as a foundation for further development and commercial application in the field of mobile communications. The research results can be useful for mobile app developers and software professionals.

Keywords: *messenger, voice messages, Android, Kotlin, programming, mobile applications*.

ЗМІСТ

ВСТУП

Актуальність теми

У сучасному світі мобільні месенджери стали невід'ємною частиною нашого повсякденного спілкування. Зростання популярності голосових повідомлень створює потребу в розробці спеціалізованих месенджерів для обміну голосовими повідомленнями на платформі Android. Існуючі додатки часто не задовольняють всі потреби користувачів, тому розробка нового месенджера стає актуальною проблемою.

У сучасному світі, де люди все більше використовують мобільні пристрої для спілкування та обміну інформацією, месенджери стали невід'ємною частиною нашого повсякденного життя. Голосові повідомлення надають користувачам можливість висловити свої думки та емоції більш ефективно, порівняно з текстовими повідомленнями. Вони є зручними в ситуаціях, коли немає можливості або бажання писати повідомлення.

У багатьох мобільних месенджерах функціонал голосових повідомлень обмежений або недостатньо розвинутий. Це може становити проблему для користувачів, які бажають зручно і ефективно комунікувати голосом. Також існує потреба у розробці нових месенджерів, які забезпечуватимуть якісний обмін голосовими повідомленнями на платформі Android.

Мета дослідження

Метою даного дипломного проєкту є розробка месенджера голосових повідомлень на платформі Android з використанням мови програмування Kotlin. Проєкт передбачає створення зручного та функціонального додатку, який забезпечуватиме зручну передачу та прослуховування голосових повідомлень. В результаті реалізації проєкту користувачі матимуть можливість ефективно спілкуватися голосом у мобільному месенджері.

Завдання дослідження

1. Огляд особливостей та існуючих рішень (аналогів) в галузі месенджерів голосових повідомлень на платформі Android. Вивчення функціональних можливостей, інтерфейсу користувача та особливостей комунікації у таких додатках.
2. Аналіз існуючих підходів та технологій, що використовуються для розробки месенджерів голосових повідомлень на платформі Android. Дослідження їх переваг, недоліків, широко використовуваних програмних бібліотек та фреймворків.
3. Вибір оптимальних технологій та інструментів розробки для створення месенджера голосових повідомлень на платформі Android. Урахування факторів, таких як ефективність, зручність використання, підтримка різних функціональних можливостей та сумісність з платформою Android.
4. Реалізація функціональності месенджера голосових повідомлень, зокрема здатність до запису, відтворення та передачі голосових повідомлень, управління контактами та створення чатових груп.
5. Оцінка результатів дослідження та розробки, порівняння розробленого месенджера голосових повідомлень з існуючими рішеннями на ринку. Визначення переваг та потенційних обмежень розробленого додатка.

Об’єкт дослідження

Об’єктом дослідження є процес розробки андроїд-додатків.

Предмет дослідження

Предметом дослідження є технології розробки месенджера голосових повідомлень на платформі Android для забезпечення зручної та ефективної комунікації користувачів через голосові повідомлення. Дослідження спрямоване на аналіз особливостей існуючих рішень та підходів до розробки месенджерів на платформі Android з акцентом на голосові повідомлення.

Методи дослідження

Теоретичні – обробка літературних джерел, синтез дослідження та аналіз досліджуваного матеріалу.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає у можливості використання розробленого месенджера голосових повідомлень на платформі Android для поліпшення комунікації користувачів через голосові повідомлення. Отримані результати дослідження та розробки мають наступні практичні переваги:

1. Зручність та ефективність комунікації: Розроблений месенджер голосових повідомлень надає зручний та швидкий спосіб обміну голосовими повідомленнями між користувачами. Це дозволяє поліпшити комунікацію, зокрема в ситуаціях, коли письмове введення є незручним або неможливим.
2. Розширені можливості спілкування: Месенджер голосових повідомлень дозволяє користувачам передавати емоції та нюанси мовлення через голосові повідомлення. Це дозволяє покращити якість спілкування та передачу інформації між користувачами.
3. Гнучкість та налаштовуваність: Розроблений месенджер може бути налаштований та адаптований під потреби конкретних користувачів або груп. Це дозволяє підвищити задоволення користувачів від використання додатку та покращити його використання для різних цільових груп.
4. Потенційні можливості комерційного використання: Розроблений месенджер голосових повідомлень може мати комерційне значення для підприємств або організацій, що працюють у сфері комунікацій та обміну інформацією. Використання месенджера може стати інструментом покращення внутрішньої комунікації у компані

Глосарій

Месенджер. Мобільний додаток або програмне забезпечення, що дозволяє користувачам обмінюватись повідомленнями, файлами, фотографіями тощо через мережу Інтернет.

Голосове повідомлення. Аудіофайл, що містить голосову інформацію та може бути записаний та відтворений за допомогою мікрофона та динаміка пристрою.

Платформа Android. Операційна система для мобільних пристроїв, розроблена компанією Google, яка використовується на багатьох смартфонах та планшетах.

Kotlin. Сучасна мова програмування, яка використовується для розробки додатків на платформі Android. Kotlin є офіційною мовою розробки для Android.

Аплікація. Програмне забезпечення, що виконує конкретну функцію або завдання на мобільному пристрої. В контексті дослідження — мобільний додаток для обміну голосовими повідомленнями.

API (Application Programming Interface). Інтерфейс програмування додатків, який надає набір функцій та процедур для взаємодії між різними програмами або компонентами програмного забезпечення.

Firebase. Облачна платформа від Google, яка надає широкий спектр сервісів для розробки мобільних та веб-додатків, включаючи збереження даних, аутентифікацію, повідомлення тощо.

UX (User Experience). Користувацький досвід, що описує сприйняття та задоволення користувача під час використання

UI (User Interface). Графічний інтерфейс користувача, який включає в себе елементи дизайну, такі як кнопки, меню, поля вводу, які використовуються для взаємодії користувача з додатком.

Push-сповіщення. Сповіщення, які надсилаються на мобільний пристрій користувача безпосередньо з додатку або з сервера, що дозволяє повідомляти про нові повідомлення або події в реальному часі.

Аутентифікація. Процес перевірки та підтвердження ідентичності користувача для доступу до системи або додатку, що забезпечує безпеку та обмеження доступу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

У сучасному цифровому світі месенджери займають важливе місце в комунікації між користувачами. Особливо зростає популярність месенджерів, що підтримують голосові повідомлення, які надають користувачам зручну можливість спілкуватися за допомогою голосу. Зокрема, розробка месенджера голосових повідомлень на платформі Android має великий потенціал і привертає увагу дослідників та розробників.

В літературі проведено дослідження та описано різні технології, що використовуються для розробки месенджерів на платформі Android. Зокрема, вивчаються різні програмні інтерфейси (API), такі як Android Messaging API та Firebase Cloud Messaging, які дозволяють розробникам інтегрувати голосові повідомлення в додаток. Також досліджуються фреймворки та бібліотеки, такі як Android Jetpack і Kotlin Coroutines, які полегшують процес розробки та забезпечують ефективну роботу з голосовими повідомленнями [1, 3].

Література також містить аналіз різних існуючих месенджерів голосових повідомлень на платформі Android. Досліджуються їхні функціональні можливості, інтерфейси користувача, механізми запису та відтворення голосових повідомлень, а також алгоритми їх стиснення і передачі аудіоданих. Цей аналіз дозволяє виявити сильні та слабкі сторони існуючих рішень і використати їх в подальшій розробці месенджера голосових повідомлень на платформі Android [2].

Окрім огляду літературних джерел, варто звернути увагу на тенденції та перспективи розвитку месенджерів голосових повідомлень на платформі Android. Зокрема, досліджуються можливості інтеграції з іншими сервісами,

розширення функціональності за допомогою штучного інтелекту та машинного навчання, покращення якості голосових повідомлень, а також забезпечення високого рівня безпеки та конфіденційності.

Отже, огляд літературних джерел засвідчує актуальність та перспективи розробки месенджера голосових повідомлень на платформі Android. Дослідження різних технологій, аналіз існуючих рішень та врахування тенденцій дозволять розробникам створити ефективний та зручний месенджер, який забезпечить користувачам можливість комунікувати голосом на платформі Android.

1.2 Аналіз існуючих систем комунікації

У сучасному цифровому світі існує багато месенджерів, які надають можливість спілкування за допомогою голосових повідомлень на платформі Android. Проаналізувавши літературні джерела, можна виявити ряд популярних і функціональних систем комунікації, які вже доступні користувачам .

WhatsApp. Цей месенджер, доступний на платформі Android, підтримує голосові повідомлення. Він забезпечує зручний інтерфейс, можливість запису і відтворення голосових повідомлень, а також шифрування даних для забезпечення конфіденційності див. рис. 1.

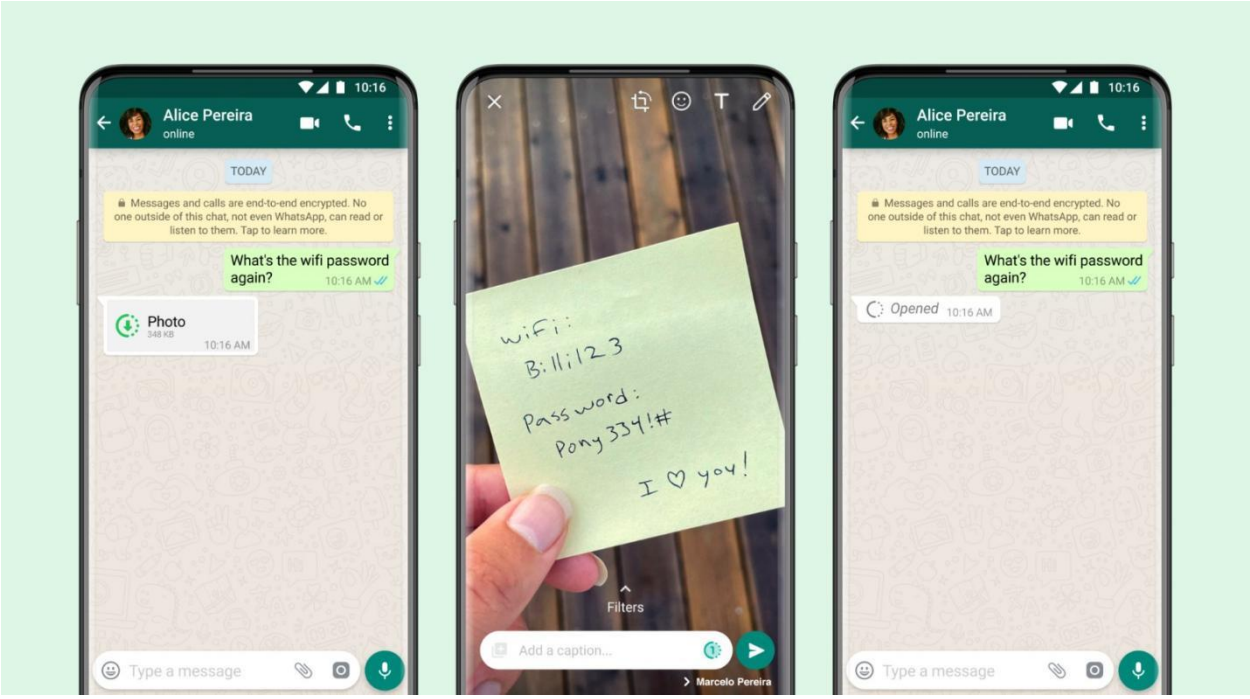

Рис. *1 WhatsApp messenger preview*

Telegram. Ще одна популярна система комунікації на платформі Android, яка підтримує голосові повідомлення. Telegram відомий своїм широким функціоналом, включаючи можливість надсилати голосові повідомлення в реальному часі та зберігати їх для подальшого відтворення зображено на рисунку 2.

Рис. *2 Telegram messenger preview*

Viber. Ця платформа також пропонує функцію голосових повідомлень на Android. Viber дозволяє користувачам записувати та надсилати голосові повідомлення в одиночних чатах або групових розмовах, інтерфейс зображено на рисунку 3.

Рис. *3 Viber messenger preview*

WeChat. Цей месенджер, поширений особливо в Китаї, має вбудовану функцію голосових повідомлень. WeChat дозволяє користувачам обмінюватися голосовими записами високої якості та використовувати їх у мультимедійних повідомленнях зображено на рисунку 4.

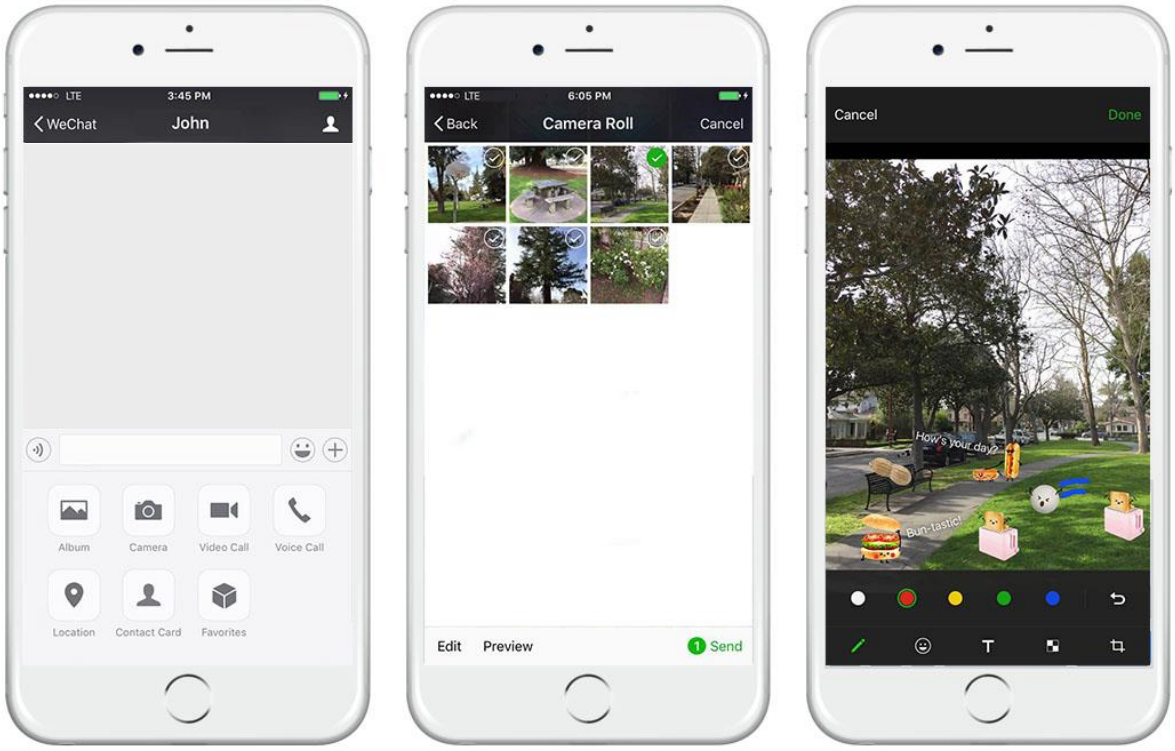

Рис. *4 WeChat messenger preview*

Для надання конкретної статистики про розподіл користувачів месенджерів на різних операційних системах на платформі Android наведу наступні дані:

Згідно зі статистикою з Google Play Store, одним з найпопулярніших месенджерів на платформі Android є WhatsApp, який має понад 5 мільярдів завантажень. Відомо, що WhatsApp підтримує різні операційні системи, включаючи Android, iOS та інші.

Звіти компанії Sensor Tower показують, що серед користувачів Android особливо популярними є такі месенджери, як Telegram, Viber та Facebook Messenger. У 2021 році Telegram мав понад 500 мільйонів активних користувачів на платформі Android, Viber — більше 100 мільйонів, а Facebook Messenger — понад 1 мільярд.

Дослідження ринку месенджерів в Україні вказують на популярність Viber серед українських користувачів. За даними дослідження компанії KIIS

(Київський міжнародний інститут соціології), Viber використовують більше половини (близько 55%) мобільних користувачів в Україні.

Важливо зауважити, що конкретна статистика може змінюватися з часом і залежить від регіону та інших факторів. Варто також враховувати, що на розподіл користувачів на різних месенджерах може мати вплив наявність функціональності та особливостей кожного додатку.

Ці системи комунікації з голосовими повідомленнями на платформі Android вже добре відомі та отримали популярність серед користувачів. Аналізуючи їх функціонал та можливості, можна зробити висновки щодо власної розробки месенджера голосових повідомлень на платформі Android, враховуючи переваги та недоліки існуючих систем і пропонуючи щось нове та унікальне.

Нижче наведені плюси та мінуси деяких вже існуючих месенджерів.

WhatsApp

Переваги:

1. Широко поширений та популярний серед користувачів у багатьох країнах світу.
2. Забезпечує шифрування повідомлень для збереження конфіденційності.
3. Можливість відправляти текстові повідомлення, голосові повідомлення, фото, відео та документи.
4. Функція групових чатів, викликів та відеодзвінків.

Недоліки:

1. Вимагає наявності стабільного Інтернет-з'єднання для коректної роботи.
2. Залежність від номера телефону для реєстрації та використання додатку.
3. Обмежена можливість налаштування приватності.

Telegram:

Переваги:

1. Сильне шифрування для збереження безпеки повідомлень.
2. Можливість створення групових чатів та каналів для великої аудиторії.

3. Підтримка відправки різноманітних мультимедійних файлів, включаючи фото, відео, документи тощо.
4. Розширені можливості конфіденційного чатування, такі як таймер самознищення повідомлень.

Недоліки:

1. Менш популярний серед широкої аудиторії порівняно з іншими месенджерами.
2. Деякі функції можуть бути складними для новачків.
3. Відсутність можливості виконувати відеодзвінки.

Viber:

Переваги:

1. Простий і зрозумілий інтерфейс.
2. Велика кількість стікерів та емодзі для виразного спілкування.
3. Підтримка відеодзвінків та голосових повідомлень.
4. Відправка текстових повідомлень, фото, відео, контактів тощо.

Недоліки:

1. Немає повної енд-ту-енд-шифрування для всіх типів повідомлень.
2. Можливість отримувати небажану рекламу та сповіщення.
3. Деякі функції, такі як групові чати, можуть працювати повільно на деяких пристроях.

1.3 Тенденції розвитку месенджерів

Дослідження має на меті визначити і проаналізувати основні підходи та методи, використовувані при розробці таких систем, а також дослідити існуючі технології та інструменти, що допомагають в їх створенні.

1. Месенджери є невід'ємною частиною сучасного цифрового світу і постійно розвиваються, адаптуючись до змінних потреб користувачів та технологічних інновацій. Оглядаючи літературні джерела та дослідження, можна виділити деякі основні тенденції розвитку месенджерів:

2. *Зростання популярності мобільних месенджерів*. За останні роки спостерігається зростання використання мобільних месенджерів порівняно з традиційними засобами комунікації. Це пов'язано зі зростанням мобільної та смартфонної технологій, а також зручними функціями, які пропонують месенджери, такими як голосові повідомлення, стікери, відеозв'язок тощо.

3. *Застосування шифрування та захисту даних*. Однією з ключових тенденцій є забезпечення безпеки та конфіденційності в месенджерах. Застосування шифрування енд-ті-енд, двофакторної аутентифікації та інших заходів безпеки стає важливим фактором для користувачів.

4. *Розширений функціонал*. Месенджери намагаються внести більше можливостей у свої додатки. Наприклад, можливість відправляти голосові повідомлення, здійснювати групові дзвінки, надсилати гроші, ділитися файлами та фотографіями, створювати "історії" і багато іншого. Такий розширений функціонал робить месенджери більш універсальними та зручними для використання.

5. *Інтеграція з іншими сервісами*. Месенджери постійно покращують свою взаємодію з іншими платформами та сервісами. Наприклад, інтеграція з соціальними мережами, онлайн-магазинами, банківськими системами, розсилками новин тощо. Це дозволяє користувачам здійснювати різноманітні дії безпосередньо в месенджері, спрощуючи комунікацію та забезпечуючи зручність використання.

6. *Інтелектуальні можливості*. З розвитком штучного інтелекту та машинного навчання месенджери стають більш "розумними". Вони можуть пропонувати користувачам автозаповнення тексту, розпізнавання мови, підказки при написанні повідомлень, переклади тощо. Це дозволяє полегшити комунікацію та забезпечує більш плавний та швидкий обмін повідомленнями.

Ці тенденції вказують на постійний розвиток месенджерів та їх адаптацію до потреб користувачів і сучасних технологій. Вивчення та розуміння цих тенденцій є важливим для успішної розробки месенджера голосових повідомлень на платформі Android.

1.4 Особливості месенджерів на платформі Android

Месенджери для платформи Android мають свої особливості, які варто враховувати. Ось деякі з них:

Інтеграція з операційною системою: Месенджери на Android мають можливість інтегруватися з іншими додатками та сервісами на пристрої, такими як контактна книга, календар, камера тощо. Це дозволяє зручно обмінюватися інформацією та використовувати різноманітні функції безпосередньо з месенджера.

Можливості мультимедіа: Багато месенджерів на Android підтримують відправку фотографій, відео, аудіо та інших мультимедійних файлів. Вони також можуть мати вбудовані редактори фотографій та відеоредактори, що дозволяє змінювати та покращувати зображення прямо в додатку.

Налаштування та персоналізація: Багато месенджерів на Android надають користувачам можливість налаштувати інтерфейс, тему, шрифти та інші параметри, щоб зробити додаток більш зручним та приємним для використання.

Синхронізація на різних пристроях: Деякі месенджери підтримують синхронізацію повідомлень та інших даних між різними пристроями. Це дозволяє користувачам легко перемикатися з одного пристрою на інший, не втрачаючи доступ до своїх повідомлень.

Відеодзвінки та голосові повідомлення: Багато месенджерів на Android підтримують відеодзвінки та голосові повідомлення, що дозволяє користувачам спілкуватися в реальному часі з використанням аудіо та відео замість текстових повідомлень.

Розширені функції: Деякі месенджери на Android мають розширені функції, такі як відправка геолокації, голосові команди, інтеграція зі сторонніми сервісами (наприклад, платежами, музичними стрімінговими

платформами тощо), що розширюють можливості спілкування та використання додатку.

Враховуючи ці особливості, користувачі платформи Android мають доступ до широкого спектру месенджерів з різноманітними функціями та можливостями, що дозволяє їм знайти оптимальний варіант для своїх комунікаційних потреб.

1.5 Постановка завдання

Метою даного дослідження є розробка месенджера голосових повідомлень на платформі Android, який буде володіти сучасними функціональними можливостями та забезпечувати зручний та безпечний обмін голосовими повідомленнями між користувачами.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Огляд літературних джерел, пов'язаних з розробкою месенджерів, особливостями голосового зв'язку та платформою Android.
2. Аналіз існуючих систем комунікації з голосовими повідомленнями на платформі Android з метою виявлення їх переваг та недоліків.
3. Дослідження процесу розробки систем комунікації з голосовими повідомленнями на платформі Android, включаючи аналіз інструментів, технологій та методів, що можуть бути використані.
4. Вивчення тенденцій розвитку месенджерів та голосового зв'язку для визначення потенційних можливостей та напрямків удосконалення розроблюваного месенджера.
5. Аналіз користувачів на різних операційних системах та в різних месенджерах з метою виявлення їх потреб та очікувань.
6. Зібрання конкретної статистики про користувачів месенджерів на різних операційних системах з використанням наукових джерел, опитувань або інших методів дослідження.

7. Аналіз плюсів та мінусів вже існуючих месенджерів для визначення потреби у вдосконаленні та унікальних особливостях розроблюваного месенджера.
8. Виявлення особливостей месенджерів на платформі Android, включаючи їх функціонал, інтерфейс, можливості настройки та інтеграцію з іншими додатками.
9. Формулювання вимог до розроблюваного месенджера з урахуванням отриманих результатів аналізу та досліджень.
10. Розробка месенджера голосових повідомлень на платформі Android згідно з визначеними вимогами та використанням сучасних технологій та інструментів.
11. Тестування та валідація розробленого месенджера з метою перевірки його функціональності, ефективності та безпеки.
12. Оцінка результатів розробки та аналіз їх відповідності поставленим цілям та вимогам.
13. Підготовка науково-технічного звіту та презентації результатів дослідження.

Виконання цих завдань дозволить реалізувати поставлену мету та створити інноваційний месенджер голосових повідомлень на платформі Android, який відповідатиме потребам користувачів та має перспективи для подальшого розвитку.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ МЕСЕНДЖЕРІВ ГОЛОСОВИХ ПОВІДОМЛЕНЬ

2.1 Класифікація месенджерів

Месенджери можна класифікувати за різними критеріями, враховуючи їх функціонал, спосіб комунікації, популярність та інші аспекти. Нижче наведено загальну класифікацію месенджерів на основі деяких ключових ознак:

1. Засновані на протоколах: Месенджери можуть бути засновані на різних протоколах комунікації, таких як SMS, MMS, інтернет-протоколи, VoIP і т.д. Наприклад, деякі месенджери використовують традиційні SMS або MMS для передачі повідомлень, тоді як інші базуються на Інтернет-протоколах для передачі даних через Інтернет.
2. Орієнтація на тип комунікації: Месенджери можуть бути спрямовані на певний тип комунікації, такий як текстові повідомлення, голосові повідомлення, відеовиклики, файловий обмін тощо. Деякі месенджери пропонують широкий спектр комунікаційних можливостей, тоді як інші можуть бути спеціалізовані на конкретному типі комунікації.
3. Кросплатформенність: Деякі месенджери підтримують роботу на різних операційних системах, таких як Android, iOS, Windows, macOS і т.д. Це дозволяє користувачам обмінюватися повідомленнями незалежно від операційної системи їх пристрою.
4. Шифрування та безпека: Деякі месенджери надають особливу увагу приватності та безпеці комунікації, надаючи енд-ту-енд шифрування для захисту повідомлень від несанкціонованого доступу.
5. Соціальна функціональність: Деякі месенджери мають розширений функціонал, який дозволяє користувачам створювати групи чату, обмінюватися статусами, фотографіями, відеозаписами, місцезнаходженням та іншою інформацією.

6. Популярність та користувацька база: Деякі месенджери мають значну користувацьку базу та велику популярність, що забезпечує широкі можливості для комунікації з іншими користувачами.

Це лише деякі ознаки класифікації месенджерів. Важливо враховувати, що ринок месенджерів постійно змінюється, і нові функції та можливості можуть з'являтися з часом.

2.2 Види месенджерів

Залежно від призначення виділяють наступні види месенджерів.

Рис. 5 *Інтерфейс соціального месенджеру Facebook Messenger*

1. Соціальні месенджери.

- WhatsApp: популярний месенджер, який дозволяє обмінюватися текстовими повідомленнями, відео та голосовими дзвінками, а також надавати фотографії та документи.

- Facebook Messenger: інтегрований месенджер, який дозволяє обмінюватися повідомленнями з друзями на платформі Facebook, а також проводити відеозв'язок та надсилати медіафайли, див. рис. 5.
- Viber: месенджер з можливістю відеозв'язку, голосових та текстових повідомлень, а також надсилання файлів та мультимедіа.

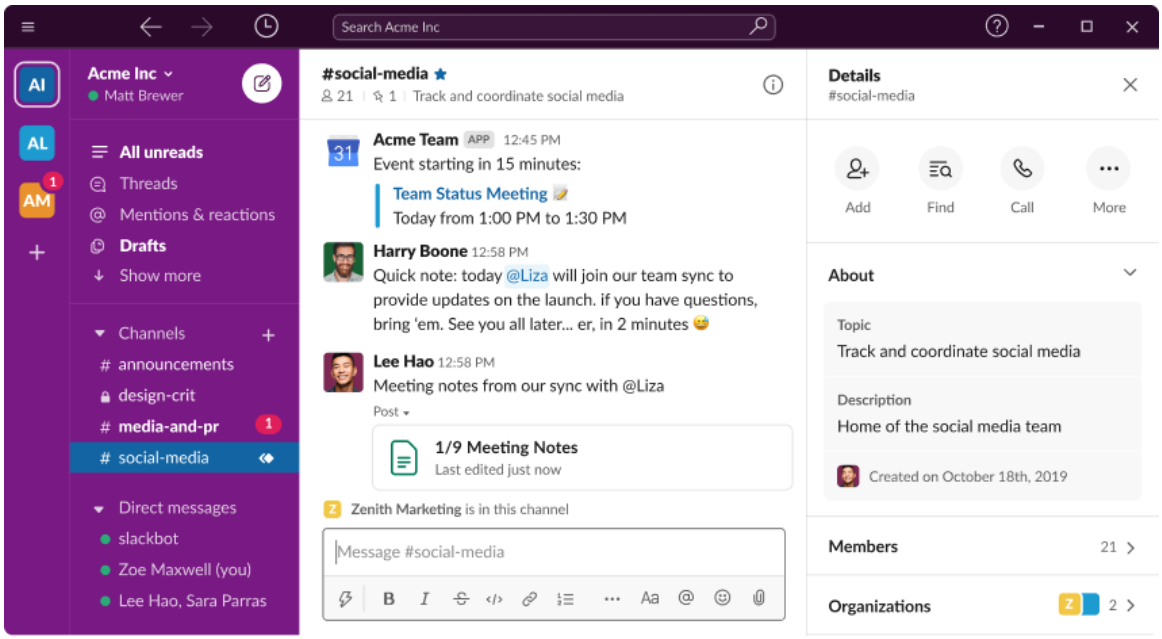

Рис. *6 Інтерфейс корпоративного мессенджеру Slack*

2. Корпоративні месенджери:

- Slack: платформа для комунікації та співпраці в команді, яка надає можливість обмінюватися повідомленнями, спільно працювати над проектами та інтегрувати різноманітні інструменти.
- Microsoft Teams: інструмент для комунікації та співпраці в офісному середовищі, який дозволяє обмінюватися повідомленнями, проводити відеозв'язок та спільно працювати над документами.
- Google Hangouts: месенджер від Google, який підтримує обмін повідомленнями, аудіо- та відеозв'язок, а також можливість проведення онлайн-конференцій.

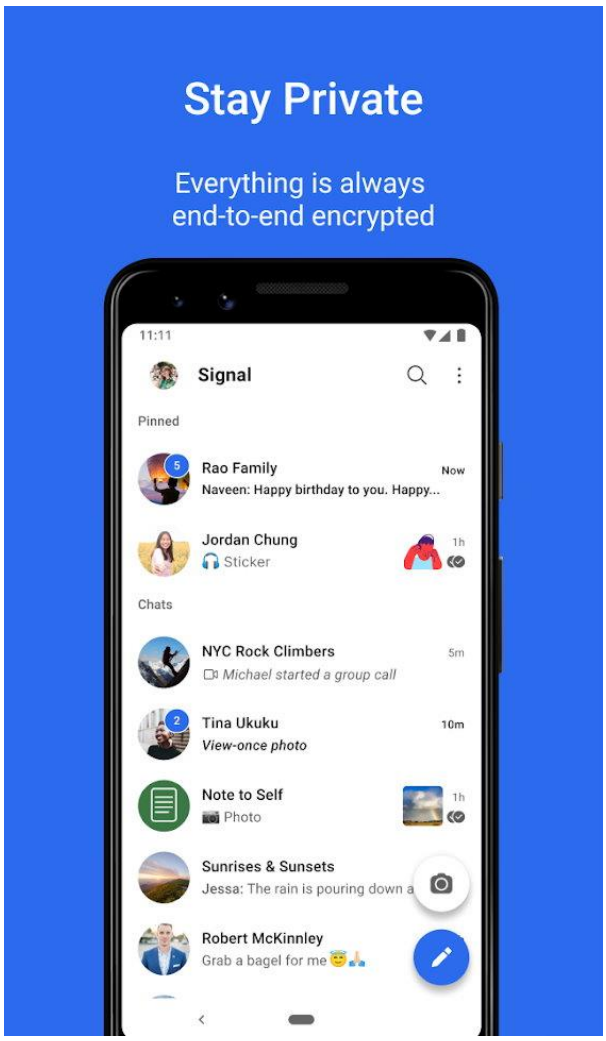

Рис. *7 Інтерфейс конфіденційного мессенджеру Signal*

3. Спеціалізовані месенджери:

- Telegram: месенджер, який відомий своїми розширеними функціями приватності та шифруванням, а також можливістю створювати канали для масового спілкування.
- Signal: месенджер з акцентом на конфіденційність та безпеку, який використовує сильне шифрування для захисту повідомлень користувачів, дивіться рисунок 7 .
- Snapchat: месенджер, який фокусується на обміні фото та відео, що самознищуються після перегляду.

Залежно від потреб та вимог користувача, вибір месенджера може бути різним.

2.3 Огляд методів розробки месенджера

При розробці месенджера існують різні методи та підходи, які допомагають розробникам створити ефективний та функціональний продукт. Ось деякі з них:

Рис. *8 Логотип мови програмування Kotlin*

Використання мови програмування Kotlin. Kotlin є сучасною мовою, яка працює на платформі Android і має деякі переваги порівняно з Java:

Короткі та зрозумілі синтаксис: Kotlin має більш зручний та компактний синтаксис порівняно з Java. Він дозволяє розробникам писати менше коду, що сприяє покращенню продуктивності та зменшенню кількості можливих помилок [1].

Надійність та безпека: Kotlin пропонує ряд вбудованих функцій, які допомагають забезпечити безпеку коду і уникнути потенційних помилок. Наприклад, він надає можливість уникати нульових значень (nullable types) і має систему безпечних викликів (safe calls), що сприяє створенню більш безпечних додатків.

Інтероперабельність з Java: Kotlin повністю сумісний з Java, що означає, що ви можете використовувати існуючий Java-код у своєму проекті Kotlin. Це дає можливість поступово переходити на Kotlin, не переписуючи весь код з нуля.

Функціональні можливості: Kotlin підтримує функціональне програмування, такі як лямбда-функції, функції вищих порядків і інші функціональні конструкції. Це дозволяє писати більш компактний та елегантний код, а також полегшує обробку подій та асинхронні операції.

Підтримка Android-фреймворку: Kotlin має нативну підтримку для розробки Android-додатків, що означає, що ви можете використовувати всі можливості Android SDK без будь-яких обмежень.

Загалом, використання мови Kotlin дозволяє розробникам створювати більш ефективні, зрозумілі та надійні додатки для платформи Android, включаючи месенджери голосових повідомлень логотип зображено на рисунку 8.

Рис. 9 *Логотип мови програмування Java*

Використання мови програмування Java. Java є популярним в розробці мобільних додатків, в тому числі і для месенджерів голосових повідомлень. Основні переваги використання Java включають наступне:

Платформова незалежність: Java є мовою програмування, яка працює на великій кількості платформ, включаючи Android. Це означає, що месенджер, розроблений на Java, може працювати на різних пристроях і ОС без необхідності в змінах коду.

Велика спільнота розробників: Java є однією з найпопулярніших мов програмування, тому вона має велику спільноту розробників. Це означає, що ви можете знайти багато матеріалів, документації, плагінів та підтримки для розробки месенджеру.

Багатий екосистема інструментів: Java має широкий вибір інструментів розробки, фреймворків і бібліотек, які полегшують розробку мобільних додатків. Наприклад, фреймворк Android SDK надає розробникам доступ до багатьох функцій Android-платформи для створення месенджеру.

Висока продуктивність: Java відома своєю високою продуктивністю та ефективністю роботи. Це особливо важливо для месенджерів, які потребують швидкої обробки даних та передачі повідомлень в реальному часі.

Безпека: Java має вбудовану систему безпеки, яка допомагає захищати додаток від вразливостей і забезпечує безпеку даних, що передаються в месенджері.

Використання мови програмування Java у розробці месенджеру голосових повідомлень дозволяє розробникам створювати потужні та функціональні додатки, які працюють надійно і забезпечують комфортне користувацьке досвід, логотип зображено на рисунку 9.

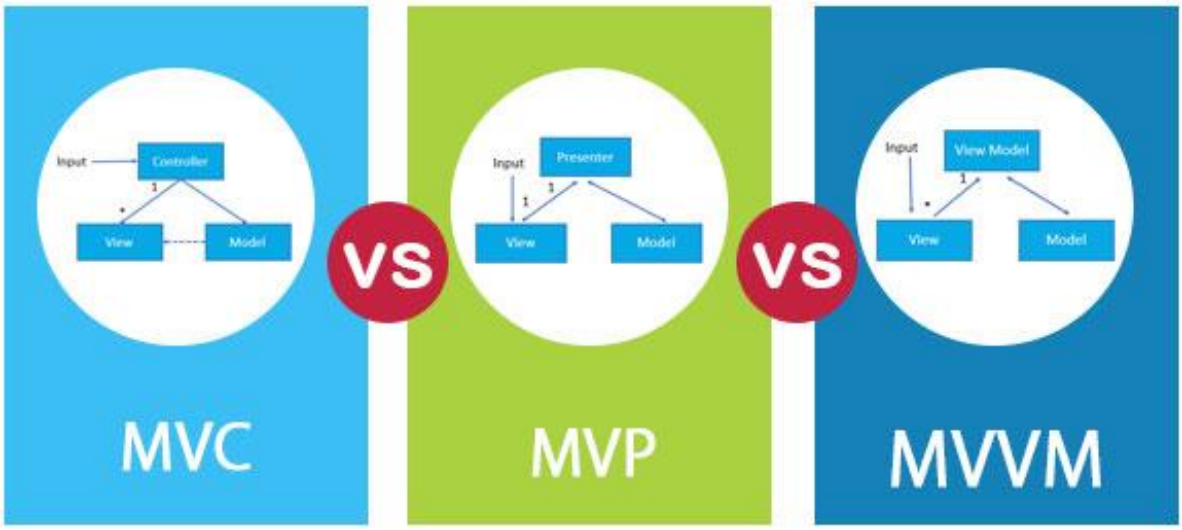

Рис. *10 Порівняння архітектурних підходів*

Використання архітектурних підходів. Для ефективної розробки месенджера рекомендується використовувати архітектурні підходи, такі як MVVM (Model-View-ViewModel) або MVP (Model-View-Presenter). Ці підходи допомагають розділити логіку додатку, користувацький інтерфейс та управління даними, як показано на рисунку 10.

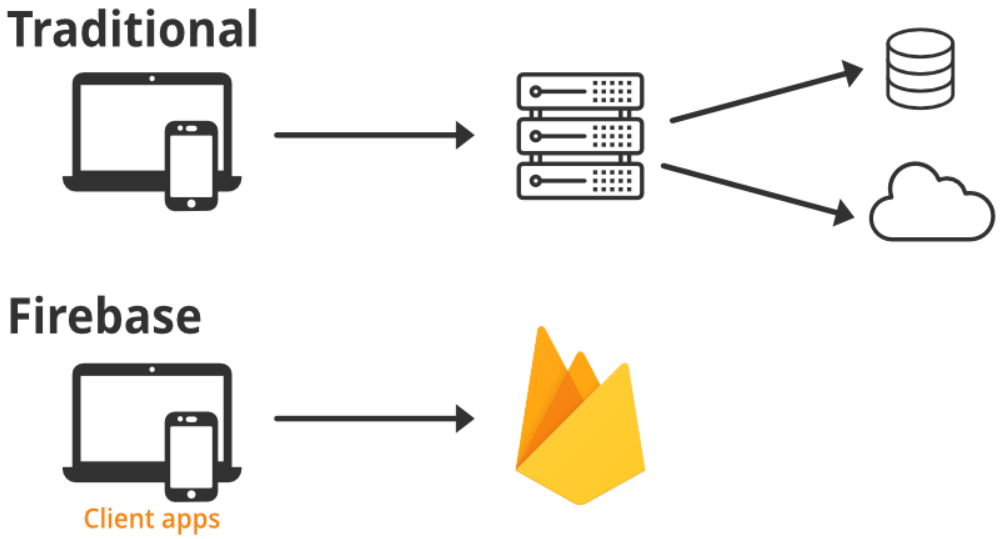

Рис. *11 суть платформи Firebase*

Використання бібліотек та фреймворків. Існує велика кількість готових бібліотек та фреймворків, які спрощують розробку месенджера. Наприклад, бібліотека Retrofit дозволяє легко взаємодіяти з API сервера, а Firebase надає інструменти для реалізації автентифікації, збереження даних та надсилання повідомлень в реальному часі.

Тестування. Для забезпечення якості месенджера рекомендується проводити регулярне тестування. Це включає модульні тести для перевірки окремих компонентів, функціональні тести для перевірки роботи месенджера в цілому, а також тести на навантаження та стабільність.

Оптимізація продуктивності. Для забезпечення швидкості та ефективності роботи месенджера варто приділити увагу оптимізації продуктивності. Це може включати кешування даних, оптимізацію запитів до бази даних, розподілення завдань та інші техніки.

2.4 Засоби розробки Android додатку

Під час розробки Android додатків розробники використовують різні інструменти та середовища, які допомагають їм створювати високоякісні та ефективні додатки. Ось кілька таких засобів:

Android Studio. Це основне інтегроване середовище розробки (IDE) для Android, яке надає розробникам широкий набір інструментів, включаючи редактор коду, інструменти для створення інтерфейсу користувача, налагоджувач та симулятор пристрою для тестування додатків [2].

Мови програмування. Kotlin та Java є основними мовами програмування, які використовуються для розробки Android додатків. Kotlin є більш сучасною та зручною мовою, яка надає багато функцій та покращень порівняно з Java. Обидві мови є популярними серед розробників Android додатків [1].

Android SDK (Software Development Kit). Це комплект розробки програмного забезпечення, який надає Google для розробки Android додатків.

Він включає в себе бібліотеки, інструменти та документацію, необхідні для розробки додатків. Android SDK забезпечує доступ до різноманітних API, які дозволяють використовувати функціональність операційної системи Android, таку як доступ до камери, мережі, баз даних та інших пристроїв.

Firebase. Це платформа розробки мобільних та веб-додатків від Google, яка надає набір інструментів та сервісів для розробки функціональності, такої як аутентифікація користувачів, збереження даних, повідомлення в реальному часі та інші, про суть платформи див. рис. 11[3].

Android Jetpack. Це набір бібліотек та компонентів, розроблений Google, який спрощує розробку Android додатків. Він містить різноманітні компоненти, такі як Navigation, LiveData, ViewModel та інші, які допомагають в управлінні життєвим циклом додатку, роботі зі збереженими даними, навігації та іншими аспектами розробки [5].

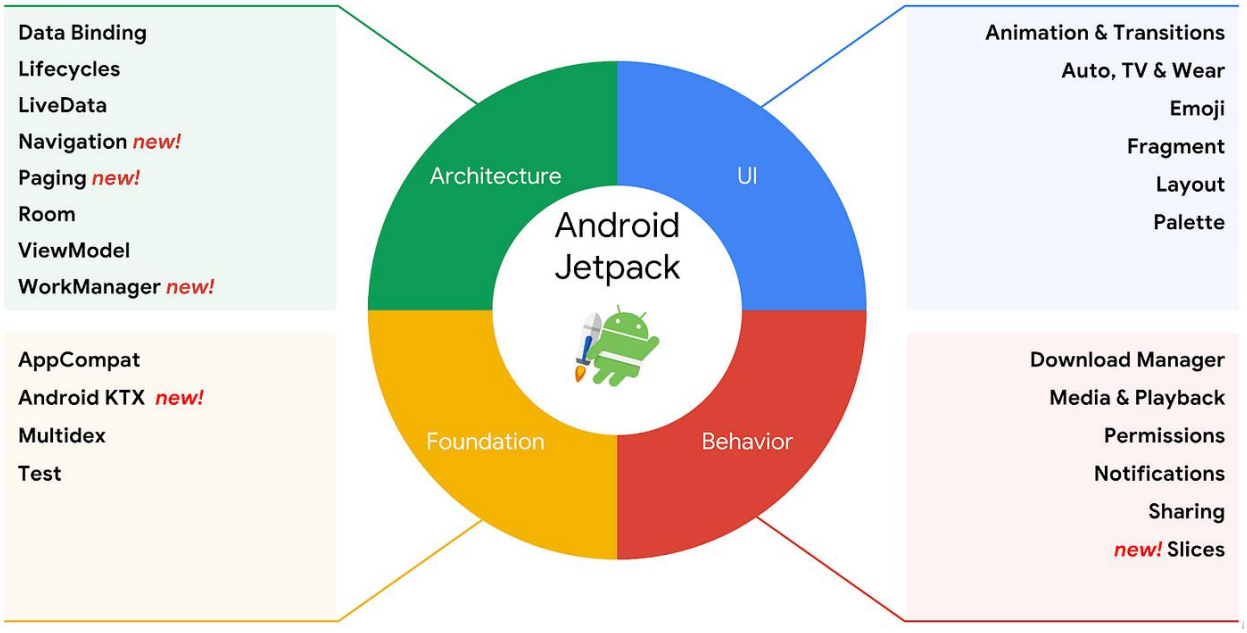

Рис. *12 Функції бібліотеки Jetpack*

Інші інструменти та бібліотеки. У розробці Android додатків використовуються різні інші інструменти та бібліотеки, які допомагають в розробці та вдосконаленні додатків, наприклад, Retrofit для роботи з

мережевими запитами, Picasso або Glide для завантаження та відображення зображень, Room для роботи з базою даних та інші [7].

Використання цих засобів та технологій сприяє ефективній розробці Android додатків і забезпечує розробникам необхідні інструменти для створення функціональних, надійних та зручних для використання месенджерів.

2.5 Фреймворки

У цьому розділі розглядаються деякі ключові фреймворки, які можуть бути використані під час розробки месенджера голосових повідомлень на платформі Android. Ці фреймворки надають зручні інструменти та функціональні можливості для спрощення розробки та покращення користувацького досвіду додатка.

Firebase. Є одним з найпопулярніших фреймворків для розробки мобільних додатків на платформі Android. Він надає різноманітні сервіси, які можуть бути корисними при розробці месенджера голосових повідомлень. Наприклад, Firebase Authentication дозволяє реалізувати систему аутентифікації користувачів, Firebase Realtime Database — зберігати та синхронізувати дані між різними пристроями в реальному часі, а Firebase Cloud Messaging — надсилати повідомлення користувачам через різні канали зв'язку. Використання Firebase може спростити розробку серверної інфраструктури та забезпечити надійну комунікацію з клієнтами [3].

Retrofit. Є потужною бібліотекою для роботи з мережевими запитами в Android додатках. Вона дозволяє зручно взаємодіяти з веб-серверами, використовуючи різні протоколи, такі як HTTP, та перетворювати вхідні та вихідні дані в об'єкти Java або Kotlin. Retrofit спрощує процес взаємодії з сервером, забезпечуючи легке налаштування та інтеграцію з існуючими розробками месенджера голосових повідомлень [1].

Dagger 2. Є фреймворком для залежностей, який допомагає забезпечити ефективне управління залежностями в Android додатках. Він базується на інверсії керування та впровадженні залежностей та надає інструменти для автоматичного створення, конфігурування та управління об'єктами. Dagger 2 спрощує створення розширюваних та легко тестованих компонентів додатку, зокрема, для реалізації функціоналу месенджера голосових повідомлень.

Mikepenz Material Drawer. Є фреймворком, який допомагає створювати навігаційне меню в стилі Material Design в Android додатках. Цей фреймворк надає широкі можливості для налаштування та персоналізації меню, включаючи вибір розміщення, стиль, колірну схему та інші параметри. Завдяки Mikepenz Material Drawer, розробники можуть швидко створювати ефектне та функціональне навігаційне меню для свого месенджера голосових повідомлень, покращуючи загальний вигляд та навігацію в додатку[6].

Picasso. Є однією з найпопулярніших бібліотек для роботи з відображенням зображень в Android додатках. Вона надає простий та зручний інтерфейс для завантаження зображень з різних джерел (локальне сховище, мережа) та їх відображення у відповідних елементах інтерфейсу користувача. Picasso автоматично оптимізує розмір та кешує зображення, що допомагає покращити продуктивність та швидкість завантаження зображень в месенджері голосових повідомлень [7].

Ці фреймворки — Mikepenz Material Drawer та Picasso — можуть бути корисними при розробці месенджера голосових повідомлень, дозволяючи створювати елегантні інтерфейси, ефективно керувати навігацією та відображенням зображень. Розробники можуть використовувати ці фреймворки для поліпшення зовнішнього вигляду додатка, підвищення зручності користування та покращення загального враження від використання месенджера голосових повідомлень.

Використання цих фреймворків може значно полегшити розробку месенджера голосових повідомлень, забезпечити більшу швидкість розробки та

вдосконалити функціональність та користувацький досвід додатка. Вибір конкретних фреймворків залежить від вимог проекту, особистих вподобань розробників та специфіки розробки месенджера голосових повідомлень.

3 РОЗРОБКА МЕСЕНДЖЕРУ ГОЛОСОВИХ ПОВІДОМЛЕНЬ

3.1 Опис предметної області

Предметною областю даної дипломної роботи є розробка месенджера голосових повідомлень на платформі Android. У сучасному світі зростає популярність месенджерів, які дозволяють користувачам обмінюватись голосовими повідомленнями швидко, зручно та ефективно. Голосові повідомлення дозволяють передавати емоції, інтонацію та нюанси мовлення, що робить комунікацію більш природною та особистою.

У даному дослідженні фокус робиться на розробці месенджера голосових повідомлень для платформи Android. Android є однією з найпопулярніших мобільних платформ, яка має велику базу користувачів та розширені можливості для розробки додатків. Розробка месенджера голосових повідомлень на платформі Android вимагає використання спеціальних інструментів, фреймворків та технологій, щоб забезпечити якісну та ефективну роботу додатку [2].

Метою даної дипломної роботи є розробка та реалізація месенджера голосових повідомлень на платформі Android з використанням сучасних технологій та інструментів. Для досягнення цієї мети необхідно провести аналіз вже існуючих рішень у цій галузі, визначити вимоги до додатку, розробити архітектуру та реалізувати необхідні функціональні модулі. Крім того, будуть проведені тести та оцінка продуктивності розробленого месенджера голосових повідомлень.

Розробка месенджера голосових повідомлень на платформі Android має великий потенціал і може знайти широке застосування в особистій та професійній комунікації. Цей дипломний проект спрямований на створення зручного та функціонального додатку, який забезпечить користувачам можливість обміну голосовими повідомленнями та покращить їх комунікаційний досвід.

3.2 Основні етапи створення месенджеру

1. Концепція месенджеру голосових повідомлень полягає в створенні додатку, який надає користувачам зручний спосіб спілкування через голосові повідомлення. Основні принципи цієї концепції включають:

Голосові повідомлення як основний засіб комунікації: Месенджер зосереджується на передачі голосових повідомлень, дозволяючи користувачам обмінюватись голосовими записами замість письмових повідомлень. Це дозволяє передавати емоції та нюанси мовлення, що робить комунікацію більш живою та природною.

Інтуїтивний та простий інтерфейс: Месенджер пропонує інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко записувати, відтворювати та надсилати голосові повідомлення. Інтерфейс повинен бути простим у використанні, зрозумілим для широкого кола користувачів, незалежно від їх технічних навичок.

Функціональні можливості голосових повідомлень: Месенджер надає різноманітні функціональні можливості пов'язані з голосовими повідомленнями, такі як запис, прослуховування, відправка голосових повідомлень.

Інтеграція з іншими функціями мобільного пристрою: Месенджер може бути інтегрований з іншими функціями мобільного пристрою, такими як контактна книга, камера, тощо. Це дозволяє користувачам зручно взаємодіяти з месенджером та використовувати його для різних цілей.

2. Для успішної реалізації проекту з розробки месенджера голосових повідомлень на платформі Android, необхідно пройти через декілька основних етапів. Кожен з цих етапів має свою унікальну роль і важливість для досягнення поставлених цілей. Нижче розглянуто основні етапи створення месенджера:

Аналіз вимог: Перший етап передбачає ретельний аналіз вимог до месенджера голосових повідомлень. Це включає визначення функціональних та

нефункціональних вимог, визначення цільової аудиторії, вивчення особливостей комунікаційних потреб користувачів, визначення основних функцій та можливостей додатку.

Проектування архітектури: На цьому етапі розробляється загальна архітектура месенджера голосових повідомлень. Визначаються основні компоненти системи, структура бази даних, способи зберігання та передачі голосових повідомлень, взаємодія з сервером та інші технічні аспекти. Розробляється детальний план роботи та встановлюються технічні вимоги до додатку.

Реалізація функціональності: На цьому етапі розробляються всі необхідні функціональні модулі месенджера голосових повідомлень. Це включає реалізацію можливостей відправки та отримання голосових повідомлень, реєстрацію користувачів, керування контактами, управління повідомленнями та інші функції, які сприяють зручному та ефективному використанню додатку.

Тестування та відлагодження: Після реалізації функціональності проводиться тестування месенджера голосових повідомлень з метою виявлення та усунення помилок, а також перевірки на відповідність вимогам. Тестування може включати модульні тести, інтеграційні тести та тестування з реальними користувачами для збору фідбеку.

3.3 Проектування месенджеру

За існуючим стандартом визначенням проектування є процес визначення архітектури, компонентів, інтерфейсу та інших показників системи. Результатом проектування є проект — цілісна сукупність моделей, властивостей або характеристик, описаних у формі, придатної для реалізації системи. Для проектування веб-додатку юридичної компанії були використані UML діаграми, які сприяють кращому розумінню та візуальному відображенню бізнес-процесів.

Діаграма варіантів використання (діаграма прецедентів) у UML — діаграма, яка візуально зображує взаємозв'язки між акторами та варіантами використання, а також є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Для побудови діаграми варіантів використання необхідно:

- Визначити акторів — групи дійових осіб, які працюють з системою по-різному через різні права доступу.
- Ідентифікувати якнайбільше варіантів використання — процесів, які можуть виконувати актори, детальніше на рисунку 13.

Рис. *13 Use Case Diagram*

У цій UML-діаграмі варіантів використання користувач виконує різні дії (функції) у месенджері голосових повідомлень. Користувач може відправляти текстові або голосові повідомлення, реєструватися за допомогою СМС, видаляти повідомлення, створювати чат, очищати чат, змінювати аватар та нікнейм.

Ця UML-діаграма варіантів використання допомагає уявити основні функції, що повинні бути реалізовані в месенджері голосових повідомлень та як користувач взаємодіє з системою

3.4 Програмна реалізація застосунку

Для реалізації збереження даних користувачів, повідомлень та файлів використовується Firebase — це платформа розробки мобільних та веб-додатків, яка надає набір інструментів та сервісів для розробки функціональних і потужних додатків. У контексті месенджеру голосових повідомлень, Firebase використаний для наступних цілей:

Аутентифікація та управління користувачами: Firebase надає вбудовані механізми аутентифікації, такі як автентифікація за допомогою електронної пошти, соціальні мережі або номер телефону. Це дозволяє реалізувати функціонал логіну та реєстрації користувачів у месенджері наведено у лістингу 1 [3, 5].

Лістинг 1. *Модель сутності користувача*.

```
data class UserModel(
    val id: String = "",
    var username: String = "",
    var bio: String = "",
```

```
    var fullname: String = "",
    var state: String = "",
    var phone: String = "",
    var photoUrl: String = "empty"
)
```

Збереження та синхронізація даних: Firebase має Realtime Database та Firestore — бази даних в реальному часі, які забезпечують збереження та синхронізацію даних між клієнтськими пристроями та сервером. Це може бути використано для збереження текстових повідомлень, фото, файлів та відео, що надсилаються користувачами.

Збереження медіафайлів: Firebase має службу зберігання об'єктів (Firebase Storage), яка дозволяє завантажувати та зберігати файли, такі як фото, відео або аудіофайли. Це може бути використано для збереження голосових повідомлень у месенджері наведено у лістингу 2 [5].

Лістинг 2. *Універсальна модель сутності для файлів*

```
data class CommonModel(
    val id: String = "",
    var username: String = "",
    var bio: String = "",
    var fullname: String = "",
    var state: String = "",
    var phone: String = "",
    var photoUrl: String = "empty",

    var text: String = "",
    var type: String = "",
    var from: String = "",
    var timeStamp: Any = "",
    var fileUrl: String = "empty",
```

```
    var lastMessage:String = "",
    var choice:Boolean = false
) {
    override fun equals(other: Any?): Boolean {
        return (other as CommonModel).id == id
    }
}
```

Після запуску застосунку користувач потрапляє у вікно реєстрації та авторизації, на якій знаходиться поле для вводу телефону, для аутентифікації та реєстрації користувачів. Вікно реєстрації та аутентифікації показано на рисунку 15.

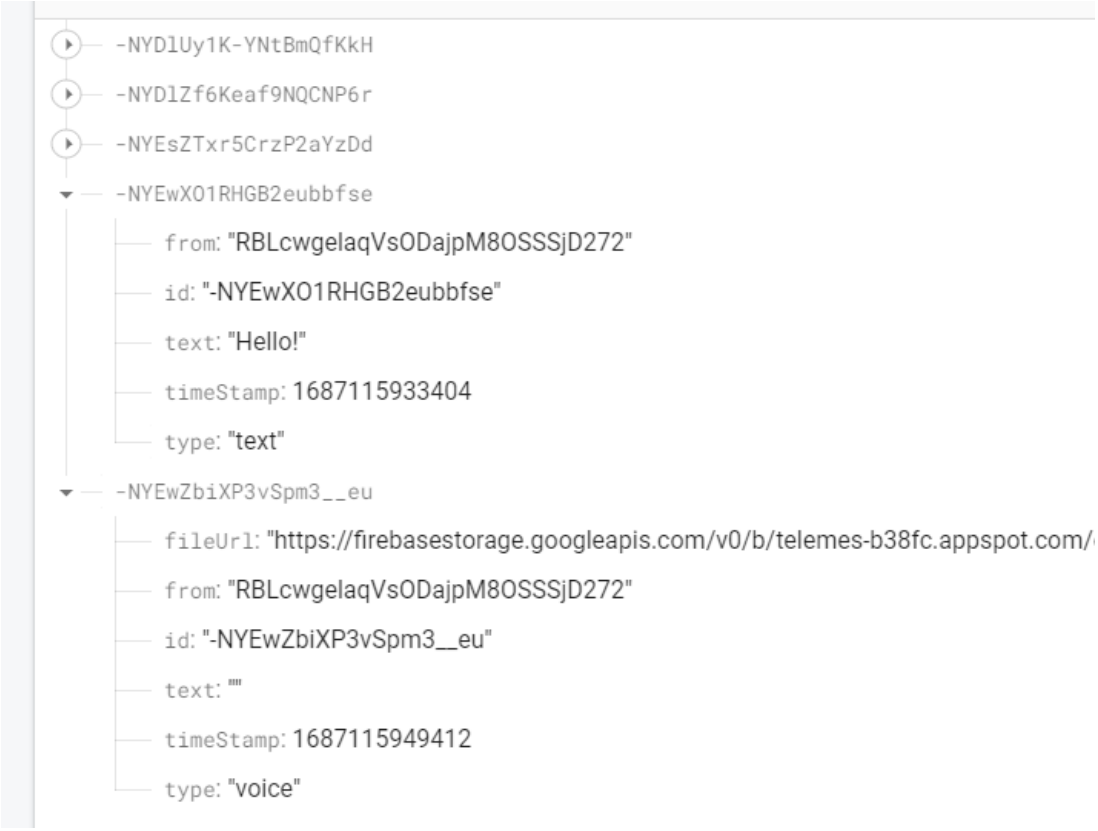

Рис. *14 Приклад збережених даних у Firebase Realtime Database*

Користувач застосунку має можливість ввести свій номер телефону, після чого на його номер телефону приходить СМС повідомлення з кодом для

авторизації, якщо користувача з таким номером ще не існує, тоді він автоматично додається до бази даних користувачів з випадковим нікнеймом та ідентифікаційним номером, див. рис. 15. Програмний код реалізації авторизації та реєстрації за номером телефону наведено у лістингу 3 [5].

Рис. *15 Вікно реєстрації та авторизації*

Лістинг *3. Авторизація та реєстрація користувача*

Головне вікно

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/register_toolbar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        style="@style/mainToolbar" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:id="@+id/data_container"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintTop_toBottomOf="@id/register_toolbar"
        android:layout_height="0dp" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Код дизайну фрагмента для введення номеру телефона

```
     <?xml version="1.0" encoding="utf-8"?>
     <androidx.constraintlayout.widget.ConstraintLayout xmlns:an-
droid="http://schemas.android.com/apk/res/android"
         xmlns:app="http://schemas.android.com/apk/res-auto"
         xmlns:tools="http://schemas.android.com/tools"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
```

```
        tools:context=".ui.screens.register.EnterPhoneNumber-
Fragment">

        <EditText
            android:hint="Enter phone number (+380)"
            android:id="@+id/register_input_phone_number"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            style="@style/editText" />

        <com.google.android.material.floatingactionbutton.Float-
ingActionButton
            android:id="@+id/register_btn_next"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_btn_next"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_margin="@dimen/app_margin"
            app:layout_constraintBottom_toBottomOf="parent"
            android:backgroundTint="@color/colorPrimary"/>


    </androidx.constraintlayout.widget.ConstraintLayout>
```

Скриптова частина коду. Клас EnterPhoneNumberFragment:

```
    /* Фрагмент для введення номера телефону під час реєстрації
*/

    class EnterPhoneNumberFragment :
Fragment(R.layout.fragment_enter_phone_number) {

        private lateinit var mPhoneNumber: String
```

```
        private lateinit var mCallback:
PhoneAuthProvider.OnVerificationStateChangedCallbacks

        override fun onStart() {
            super.onStart()

            /* Callback який повертає результат верифікації */
            mCallback = object :
PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
                override fun onVerificationCompleted(credential:
PhoneAuthCredential) {
                    /* Функція спрацьовує якщо верифікація вже
була зроблена,
                    * користувач авторизується в додатку без
підтвердження смс */

AUTH.signInWithCredential(credential).addOnCompleteListener {
task ->
                        if (task.isSuccessful) {
                            showToast("Добро пожаловать")
                            restartActivity()
                        } else
showToast(task.exception?.message.toString())
                    }
                }

                override fun onVerificationFailed(p0:
FirebaseException) {
                    /* Функція спрацьовує, якщо верифікація не
вдалася*/
                    showToast(p0.message.toString())
                }

                override fun onCodeSent(id: String, token:
PhoneAuthProvider.ForceResendingToken) {
```

```
                    /* Функція спрацьовує, якщо верифікація не
вдалася*/
                    replaceFragment(
                        EnterCodeFragment(
                            mPhoneNumber,
                            id
                        )
                    )
                }
            }
            register_btn_next.setOnClickListener { sendCode() }
        }

        private fun sendCode() {
            /* Функція перевіряє поле для введення номера
телефону, якщо поле порожнє виводить повідомлення.
              * Якщо поле не порожнє, то починається процедура
авторизації/реєстрації */
            if
(register_input_phone_number.text.toString().isEmpty()) {

showToast(getString(R.string.register_toast_enter_phone))
            } else {
                authUser()
            }
        }

        private fun authUser() {
            /* Ініціалізація */
            mPhoneNumber =
register_input_phone_number.text.toString()
            PhoneAuthProvider.getInstance().verifyPhoneNumber(
                mPhoneNumber,
                60,
                TimeUnit.SECONDS,
                APP_ACTIVITY,
```

```
                mCallback
            )
        }
    }
```

Код дизайну фрагмента для введення коду з СМС

```
    <?xml version="1.0" encoding="utf-8"?>
    <androidx.constraintlayout.widget.ConstraintLayout xmlns:an-
droid="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".ui.screens.register.EnterCodeFragment">

        <ImageView
            android:id="@+id/register_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/vertical_large_mar-
gin"
            android:src="@drawable/register_image"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            android:contentDescription="@string/cd_register_im-
age" />

        <TextView
            android:id="@+id/register_text_enter_code"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:gravity="center"
```

```
            android:layout_margin="@dimen/app_small_margin"
            android:textColor="@color/colorBlack"
            android:textSize="@dimen/normalText"
            android:text="Enter code"
            app:layout_constraintTop_toBottomOf="@id/regis-
ter_image"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/register_text_we_sent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            style="@style/smallText"
            android:gravity="center"
            android:layout_margin="@dimen/app_small_margin"
            android:text="We sent you SMS with code"
            app:layout_constraintTop_toBottomOf="@+id/regis-
ter_text_enter_code"/>

        <EditText
            style="@style/editText"
            android:layout_width="wrap_content"
            android:id="@+id/register_input_code"
            android:autofillHints=""
            android:maxLength="6"
            android:gravity="center"
            android:hint="@string/register_hint_default_code"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/regis-
ter_text_we_sent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

Скриптова частина коду. Клас EnterCodeFragment:

```
    /* Фрагмент для введення коду підтвердження під час
реєстрації */

    class EnterCodeFragment(val phoneNumber: String, val id:
String) :
        Fragment(R.layout.fragment_enter_code) {


        override fun onStart() {
            super.onStart()
            APP_ACTIVITY.title = phoneNumber

register_input_code.addTextChangedListener(AppTextWatcher {
                val string = register_input_code.text.toString()
                if (string.length == 6) {
                    enterCode()
                }
            })
        }

        private fun enterCode() {
            /* Функція перевіряє код, якщо все нормально,
здійснює створення інформації про користувача в базі даних.*/
            val code = register_input_code.text.toString()
            val credential = PhoneAuthProvider.getCredential(id,
code)

AUTH.signInWithCredential(credential).addOnCompleteListener
{ task ->
                if (task.isSuccessful) {
                    val uid = AUTH.currentUser?.uid.toString()
                    val dateMap = mutableMapOf<String, Any>()
                    dateMap[CHILD_ID] = uid
```

```
                    dateMap[CHILD_PHONE] = phoneNumber


REF_DATABASE_ROOT.child(NODE_USERS).child(uid)
                        .addListenerForSingleValueEvent(AppValue
EventListener{

                            if (!it.hasChild(CHILD_USERNAME)){
                                dateMap[CHILD_USERNAME] = uid
                            }

                            REF_DATABASE_ROOT.child(
                                NODE_PHONES
                            ).child(phoneNumber).setValue(uid)
                                .addOnFailureListener
{ showToast(it.message.toString()) }
                                .addOnSuccessListener {
                                    REF_DATABASE_ROOT.child(
                                        NODE_USERS
                                    ).child(uid).updateChildren(
dateMap)
                                        .addOnSuccessListener {
                                            showToast("Welcome")
                                            restartActivity()
                                        }
                                        .addOnFailureListener
{ showToast(it.message.toString()) }
                                }
                        })



                } else
showToast(task.exception?.message.toString())
            }
        }
```

```
}
```

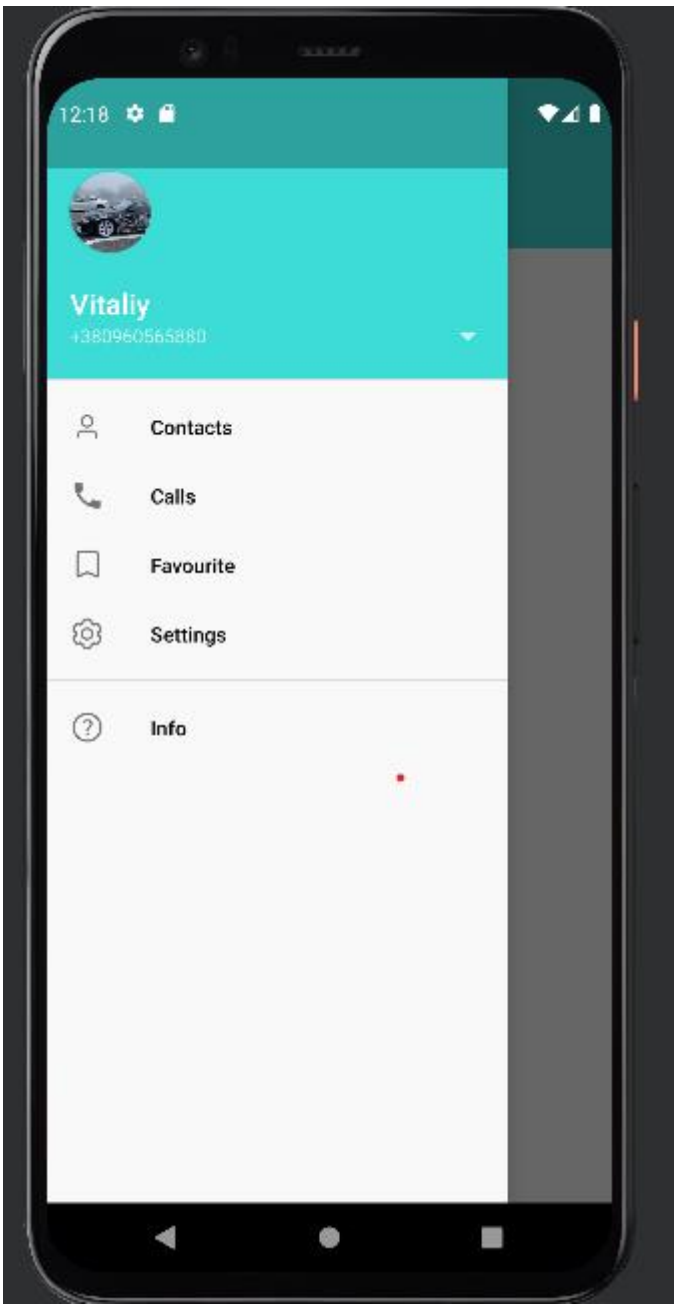

Рис. *16 Вікно бокового меню фреймворку Mikepenz*

Після логіну, або реєстрації відкривається головне меню з активними чатами та боковим меню, реалізованим за допомогою фреймворку Mikepenz Material Drawer наведено у лістингу 4, Рис. 16 [6].

Лістинг *4. Реалізація бокового меню*

/* Об'єкт, що реалізує бічне меню Navigation Drawer */

```
class AppDrawer {

    private lateinit var mDrawer: Drawer
    private lateinit var mHeader: AccountHeader
    private lateinit var mDrawerLayout: DrawerLayout
    private lateinit var mCurrentProfile:ProfileDrawerItem

    fun create() {
/* Створення бокового меню */
        initLoader()
        createHeader()
        createDrawer()
        mDrawerLayout = mDrawer.drawerLayout
    }

    fun disableDrawer() {
        /* Відключення висувного меню */

mDrawer.actionBarDrawerToggle?.isDrawerIndicatorEnabled = false

APP_ACTIVITY.supportActionBar?.setDisplayHomeAsUpEnabled(true)

mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CL
OSED)
        APP_ACTIVITY.mToolbar.setNavigationOnClickListener {

APP_ACTIVITY.supportFragmentManager.popBackStack()
        }
    }

    fun enableDrawer() {
        /* Увімкнення висувного меню */
```

```
APP_ACTIVITY.supportActionBar?.setDisplayHomeAsUpEnabled(false)

mDrawer.actionBarDrawerToggle?.isDrawerIndicatorEnabled = true

mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLOCKED)
            APP_ACTIVITY.mToolbar.setNavigationOnClickListener {
                mDrawer.openDrawer()
            }
        }

        private fun createDrawer() {
            /* Створення дравера */
            mDrawer = DrawerBuilder()
                .withActivity(APP_ACTIVITY)
                .withToolbar( APP_ACTIVITY.mToolbar)
                .withActionBarDrawerToggle(true)
                .withSelectedItem(-1)
                .withAccountHeader(mHeader)
                .addDrawerItems(
                    PrimaryDrawerItem().withIdentifier(100)
                        .withIconTintingEnabled(true)
                        .withName("Создать группу")
                        .withSelectable(false)

.withIcon(R.drawable.ic_menu_create_groups),
                    PrimaryDrawerItem().withIdentifier(101)
                        .withIconTintingEnabled(true)
                        .withName("Create secret chat")
                        .withSelectable(false)

.withIcon(R.drawable.ic_menu_secret_chat),
                    PrimaryDrawerItem().withIdentifier(102)
                        .withIconTintingEnabled(true)
                        .withName("Create channel")
                        .withSelectable(false)
```

```
.withIcon(R.drawable.ic_menu_create_channel),
                    PrimaryDrawerItem().withIdentifier(103)
                        .withIconTintingEnabled(true)
                        .withName("Contacts")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_contacts),
                    PrimaryDrawerItem().withIdentifier(104)
                        .withIconTintingEnabled(true)
                        .withName("Calls")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_phone),
                    PrimaryDrawerItem().withIdentifier(105)
                        .withIconTintingEnabled(true)
                        .withName("Favourite")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_favorites),
                    PrimaryDrawerItem().withIdentifier(106)
                        .withIconTintingEnabled(true)
                        .withName("Settings")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_settings),
                    DividerDrawerItem(),
                    PrimaryDrawerItem().withIdentifier(108)
                        .withIconTintingEnabled(true)
                        .withName("Пригласить друзей")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_invate),
                    PrimaryDrawerItem().withIdentifier(109)
                        .withIconTintingEnabled(true)
                        .withName("Info")
                        .withSelectable(false)
                        .withIcon(R.drawable.ic_menu_help)
                ).withOnDrawerItemClickListener(object :
Drawer.OnDrawerItemClickListener {
                    override fun onItemClick(
```

```
                view: View?,
                position: Int,
                drawerItem: IDrawerItem<*>
            ): Boolean {
               clickToItem(position)
                return false
            }
        }).build()

}

private fun clickToItem(position:Int){
    when (position) {
        1 -> replaceFragment(AddContactsFragment())
        7 -> replaceFragment(SettingsFragment())
        4 -> replaceFragment(ContactsFragment())
    }
}

private fun createHeader() {
    /* Створення хедера*/
    mCurrentProfile = ProfileDrawerItem()
        .withName(USER.fullname)
        .withEmail(USER.phone)
        .withIcon(USER.photoUrl)
        .withIdentifier(200)
    mHeader = AccountHeaderBuilder()
        .withActivity(APP_ACTIVITY)
        .withHeaderBackground(R.drawable.header)
        .addProfiles(
            mCurrentProfile
        ).build()
}
```

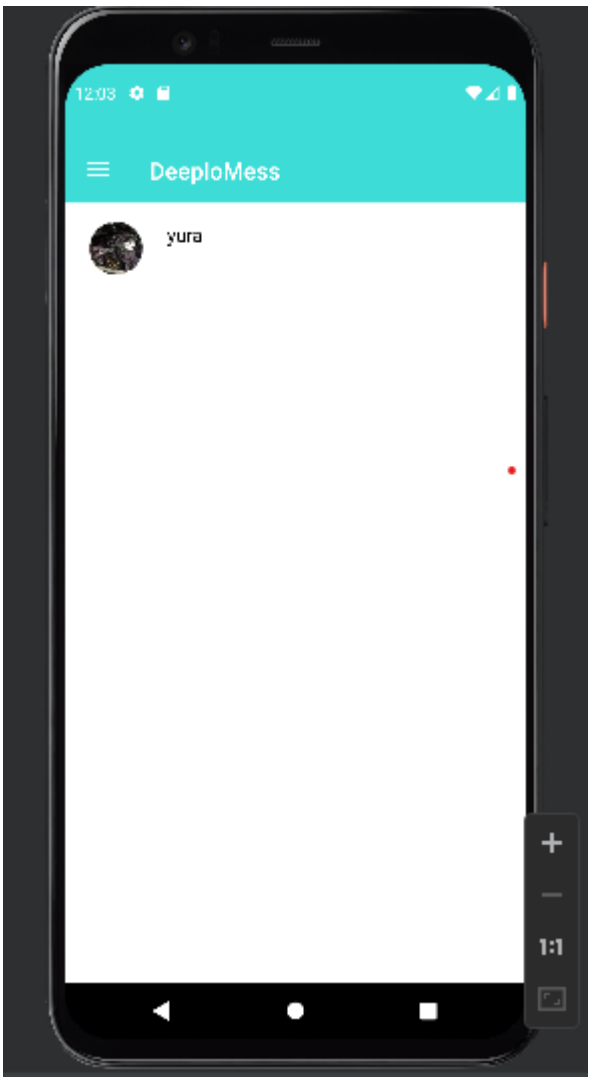

Рис. 17 *Вікно зі списком активних чатів*

У боковому меню можна зайти у вкладинку контакти та написати користувачу, який є у контактній книзі, тоді він відобразиться у списку ваших активних чатів (Рис 17), наведено у лістингу 5.

Лістинг 5. *Реалізація відображення списку активних чатів*

```
class MainListAdapter :
RecyclerView.Adapter<MainListAdapter.MainListHolder>() {

    private var listItems = mutableListOf<CommonModel>()
```

```
    class MainListHolder(view: View) :
RecyclerView.ViewHolder(view) {
        val itemName: TextView = view.main_list_item_name
        val itemLastMessage: TextView =
view.main_list_last_message
        val itemPhoto: CircleImageView =
view.main_list_item_photo
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): MainListHolder {
        val view =

LayoutInflater.from(parent.context).inflate(R.layout.main_list_i
tem, parent, false)

        val holder = MainListHolder(view)
        holder.itemView.setOnClickListener {
            when(listItems[holder.adapterPosition].type){
                TYPE_CHAT
->replaceFragment(SingleChatFragment(listItems[holder.adapterPos
ition]))
                TYPE_GROUP ->
replaceFragment(GroupChatFragment(listItems[holder.adapterPositi
on]))
            }
        }
        return holder
    }

    override fun getItemCount(): Int = listItems.size

    override fun onBindViewHolder(holder: MainListHolder,
position: Int) {
        holder.itemName.text = listItems[position].fullname
```

```
        holder.itemLastMessage.text =
listItems[position].lastMessage

holder.itemPhoto.downloadAndSetImage(listItems[position].photoUr
l)
    }

    fun updateListItems(item:CommonModel){
        listItems.add(item)
        notifyItemInserted(listItems.size)
    }
}
```

Рис. *18 Вікно з чатом*

Лістинг 6. *Реалізація функціоналу чату*

```
class SingleChatFragment(private val contact: CommonModel) :
    BaseFragment(R.layout.fragment_single_chat) {

    private lateinit var mListenerInfoToolbar: AppValueEventLis-
tener
```

```
    private lateinit var mReceivingUser: UserModel
    private lateinit var mToolbarInfo: View
    private lateinit var mRefUser: DatabaseReference
    private lateinit var mRefMessages: DatabaseReference
    private lateinit var mAdapter: SingleChatAdapter
    private lateinit var mRecyclerView: RecyclerView
    private lateinit var mMessagesListener: AppChildEventLis-
tener
    private var mCountMessages = 10
    private var mIsScrolling = false
    private var mSmoothScrollToPosition = true
    private lateinit var mSwipeRefreshLayout: SwipeRefreshLayout
    private lateinit var mLayoutManager: LinearLayoutManager
    private lateinit var mAppVoiceRecorder: AppVoiceRecorder
    private lateinit var mBottomSheetBehavior: BottomSheetBehav-
ior<*>

    override fun onResume() {
        super.onResume()
        initFields()
        initToolbar()
        initRecycleView()
    }

    @SuppressLint("ClickableViewAccessibility")
    private fun initFields() {
        setHasOptionsMenu(true)
        mBottomSheetBehavior= BottomSheetBehavior.from(bot-
tom_sheet_choice)
        mBottomSheetBehavior.state = BottomSheetBehav-
ior.STATE_HIDDEN
        mAppVoiceRecorder = AppVoiceRecorder()
        mSwipeRefreshLayout = chat_swipe_refresh
        mLayoutManager = LinearLayoutManager(this.context)
        chat_input_message.addTextChangedListener(AppTextWatcher
{
```

```
            val string = chat_input_message.text.toString()
            if (string.isEmpty() || string == "Запись") {
                chat_btn_send_message.visibility = View.GONE
                chat_btn_attach.visibility = View.VISIBLE
                chat_btn_voice.visibility = View.VISIBLE
            } else {
                chat_btn_send_message.visibility = View.VISIBLE
                chat_btn_attach.visibility = View.GONE
                chat_btn_voice.visibility = View.GONE
            }
        })

        chat_btn_attach.setOnClickListener { attach() }

        CoroutineScope(Dispatchers.IO).launch {
            chat_btn_voice.setOnTouchListener { v, event ->
                if (checkPermission(RECORD_AUDIO)) {
                    if (event.action == MotionEvent.ACTION_DOWN)
{
                        //TODO record
                        chat_input_message.setText("Запись")
                        chat_btn_voice.setColorFilter(
                            ContextCompat.getColor(
                                APP_ACTIVITY,
                                R.color.primary
                            )
                        )
                        val messageKey = getMessageKey(con-
tact.id)
                        mAppVoiceRecorder.startRecord(mes-
sageKey)
                    } else if (event.action == MotionEvent.AC-
TION_UP) {
                        //TODO stop record
                        chat_input_message.setText("")
                        chat_btn_voice.colorFilter = null
```

```
                        mAppVoiceRecorder.stopRecord { file,
messageKey ->
                            uploadFileToStorage(Uri.from-
File(file),messageKey,contact.id, TYPE_MESSAGE_VOICE)
                            mSmoothScrollToPosition = true
                        }
                    }
                }
                true
            }
        }

    }

    private fun attach() {
        mBottomSheetBehavior.state = BottomSheetBehav-
ior.STATE_EXPANDED
        btn_attach_file.setOnClickListener { attachFile() }
        btn_attach_image.setOnClickListener { attachImage() }
    }

    private fun attachFile(){
        val intent = Intent(Intent.ACTION_GET_CONTENT)
        intent.type = "*/*"
        startActivityForResult(intent, PICK_FILE_REQUEST_CODE)
    }


    private fun attachImage() {
        CropImage.activity()
            .setAspectRatio(1, 1)
            .setRequestedSize(250, 250)
            .start(APP_ACTIVITY, this)
    }

    private fun initRecycleView() {
```

```
        mRecyclerView = chat_recycle_view
        mAdapter = SingleChatAdapter()
        mRefMessages = REF_DATABASE_ROOT
            .child(NODE_MESSAGES)
            .child(CURRENT_UID)
            .child(contact.id)
        mRecyclerView.adapter = mAdapter
        mRecyclerView.setHasFixedSize(true)
        mRecyclerView.isNestedScrollingEnabled = false
        mRecyclerView.layoutManager = mLayoutManager
        mMessagesListener = AppChildEventListener {
            val message = it.getCommonModel()

            if (mSmoothScrollToPosition) {
                mAdapter.addItemToBottom(AppViewFac-
tory.getView(message)) {
                    mRecyclerView.smoothScrollToPosi-
tion(mAdapter.itemCount)
                }
            } else {
                mAdapter.addItemToTop(AppViewFac-
tory.getView(message)) {
                    mSwipeRefreshLayout.isRefreshing = false
                }
            }

        }
        mRefMessages.limitToLast(mCountMessages).addChildEvent-
Listener(mMessagesListener)

        mRecyclerView.addOnScrollListener(object : Recycler-
View.OnScrollListener() {

            override fun onScrolled(recyclerView: RecyclerView,
dx: Int, dy: Int) {
                super.onScrolled(recyclerView, dx, dy)
```

```
println(mRecyclerView.recycledViewPool.getRecy-
cledViewCount(0))
                if (mIsScrolling && dy < 0 && mLayoutMan-
ager.findFirstVisibleItemPosition() <= 3) {
                    updateData()
                }
            }

            override fun onScrollStateChanged(recyclerView: Re-
cyclerView, newState: Int) {
                super.onScrollStateChanged(recyclerView,
newState)
                if (newState == AbsListView.OnScrollLis-
tener.SCROLL_STATE_TOUCH_SCROLL) {
                    mIsScrolling = true
                }
            }
        })

        mSwipeRefreshLayout.setOnRefreshListener { up-
dateData() }
    }
```

Для запису голосових повідомлень було використано вбудований клас MediaRecorder, який дозволяє записувати звук з вбудованого та зовнішнього мікрофону, реалізація продемонстрована у лістингу 7.

Діаграма станів MediaRecorder, зображеного на рисунку 19 відображає різні стани, в яких може перебувати об'єкт MediaRecorder під час запису аудіо або відео. Основна мета цієї діаграми — показати послідовність переходів між цими станами від початку запису до завершення або припинення запису.

Основні стани, які відображені на діаграмі, включають:

Ініціалізація (Initialization): Початковий стан, коли об'єкт MediaRecorder ще не ініціалізований і готовий до використання.

Підготовка (Prepared): Об'єкт MediaRecorder переходить у цей стан після ініціалізації. В цьому стані відбувається налаштування параметрів запису, таких як джерело аудіо або відео, формат файлу тощо.

Готовий до запису (Ready): Після підготовки об'єкт MediaRecorder готовий приймати команди для початку запису.

Запис (Recording): Об'єкт MediaRecorder переходить у цей стан після запуску запису. В цьому стані відбувається фактичний процес запису аудіо або відео.

Зупинка (Stop): Після завершення запису об'єкт MediaRecorder переходить у цей стан для зупинки процесу запису та підготовки до нового запису або збереження результатів.

Відключено (Released): Кінцевий стан, коли об'єкт MediaRecorder повністю відключений та готовий до знищення або повторного використання.

Діаграма станів MediaRecorder демонструє послідовність переходів між цими станами залежно від дій користувача або зовнішніх подій, таких як початок або припинення запису. Вона допомагає розуміти поведінку об'єкта MediaRecorder та контролювати процес запису аудіо або відео в додатку на платформі Android [5].

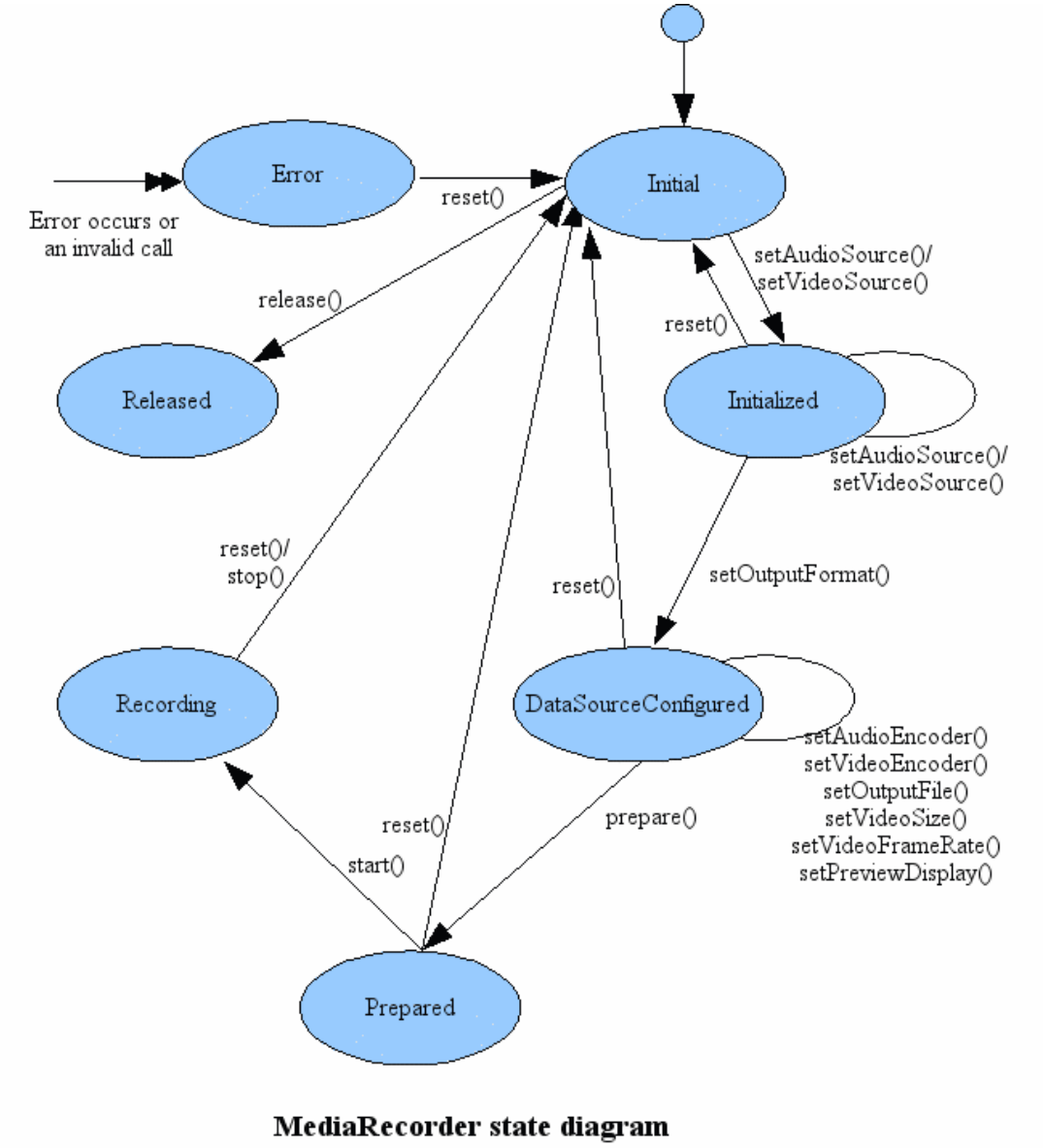

Рис. *19 Діаграма станів класу MediaRecorder*

Лістинг *7. Реалізація запису звуку з мікрофона*

```
class AppVoiceRecorder {

    private val mMediaRecorder = MediaRecorder()
    private lateinit var mFile: File
    private lateinit var mMessageKey: String

    fun startRecord(messageKey: String) {
```

```
        try {
            mMessageKey = messageKey
            createFileForRecord()
            prepareMediaRecorder()
            mMediaRecorder.start()
        } catch (e: Exception) {
            showToast(e.message.toString())
        }

    }

    private fun prepareMediaRecorder() {
        mMediaRecorder.apply {
            reset()
            setAudioSource(MediaRecorder.AudioSource.DEFAULT)
            setOutputFormat(MediaRecorder.OutputFormat.DEFAULT)
            setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT)
            setOutputFile(mFile.absolutePath)
            prepare()
        }

    }

    private fun createFileForRecord() {
        mFile = File(APP_ACTIVITY.filesDir, mMessageKey)
        mFile.createNewFile()
    }

    fun stopRecord(onSuccess: (file: File, messageKey: String) ->
Unit) {
        try {
            mMediaRecorder.stop()
            onSuccess(mFile, mMessageKey)
        } catch (e: Exception) {
            showToast(e.message.toString())
            mFile.delete()
```

```
        }
    }

    fun releaseRecorder() {
        try {
            mMediaRecorder.release()
        } catch (e: Exception) {
            showToast(e.message.toString())
        }
    }

}
```

3.5 Висновки з реалізації додатку

1. У цій роботі було проведено детальне проектування месенджера голосових повідомлень. Було створено UML діаграми, що допомогли зрозуміти структуру системи та функціональність кожного етапу взаємодії користувача з додатком.

2. Для розробки месенджера голосових повідомлень було використано різні фреймворки та інструменти, включаючи Firebase для забезпечення функцій відправки повідомлень, реєстрації, аутентифікації користувачів та зберігання даних. Також були використані фреймворки Material Drawer та Picasso для покращення інтерфейсу та обробки медіа-ресурсів [7].

3. У результаті дослідження та розробки цього месенджера голосових повідомлень було досягнуто поставлені цілі та завдання. Додаток забезпечує зручне та ефективну спілкування за допомогою голосових повідомлень, дозволяє користувачам обмінюватися текстовими повідомленнями, відправляти фотографії, відео та інші медіа-файли. Він також надає зручний інтерфейс для управління обліковими записами, зміни аватару та нікнейму.

ВИСНОВКИ

1. У цій дипломній роботі була розглянута розробка месенджера голосових повідомлень для платформи Android. Результати досліджень показали, що месенджери з голосовими повідомленнями мають значний потенціал і відіграють важливу роль у сучасному спілкуванні.

2. Під час аналізу існуючих систем комунікації з голосовими повідомленнями на платформі Android було виявлено різноманітні рішення з різними функціональними можливостями та інтерфейсами. Огляд розробки систем комунікації з голосовими повідомленнями також показав, що цей ринок постійно розвивається, а розробники шукають нові шляхи для покращення функціональності та зручності використання.

3. Аналіз користувачів на різних операційних системах та в різних месенджерах показав, що користувачі мають різні вимоги та переваги щодо функцій месенджерів. Деякі користувачі надають перевагу простим та легким у використанні месенджерам, тоді як інші більше цікавляться розширеними функціями та можливостями налаштування.

4. Загалом, розробка месенджера голосових повідомлень на платформі Android є складним завданням, що вимагає глибокого розуміння предметної області, використання відповідних фреймворків та інструментів, а також врахування потреб користувачів. Проектування та реалізація такого додатку може виявитися вигідною для комунікації та спілкування в сучасному світі, де голосові повідомлення стають все більш популярними та зручними способами обміну інформацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Josh Skeen and David Greenhalgh, Andrew Bailey. Kotlin Programming: The Big Nerd Ranch Guide, 2021. 541 p.

2. Mastering Android Development with Kotlin, Milos Vasic, 2017. 378 p.

3. Firebase Essentials for Android, Neil Smyth, 2017. 534 p.

4. Pro Git, Scott Chacon and Ben Straub, 2014. 440 p.

5. Офіційна документація Android та Firebase, URL: https://firebase.google.com/docs (дата звернення: 03.03.2023р.). https://developer.android.com/docs (дата звернення: 03.03.2023р.).

6. Mikepenz Material Drawer FAQ, URL: https://github.com/mikepenz/MaterialDrawer/tree/develop/FAQ(дата звернення: 04.13.2023р.).

7. Picasso Framework Documentation, URL: https://picassosr.readthedocs.io/en/latest/(дата звернення: 04.14.2023р.).