**Методичка: Розробка месенджера на Flutter + Firebase**

**Частина 1. Опис проекту**

**1. Загальна концепція проекту**

Проект **"Flutter Messenger"** — це класичний приклад CRUD (Create, Read, Update, Delete) додатку з елементами реального часу (Real-time). Він ідеально підходить для демонстрації можливостей хмарної платформи Firebase у зв'язці з Flutter.

**Ключові технічні аспекти, які розглядаються:**

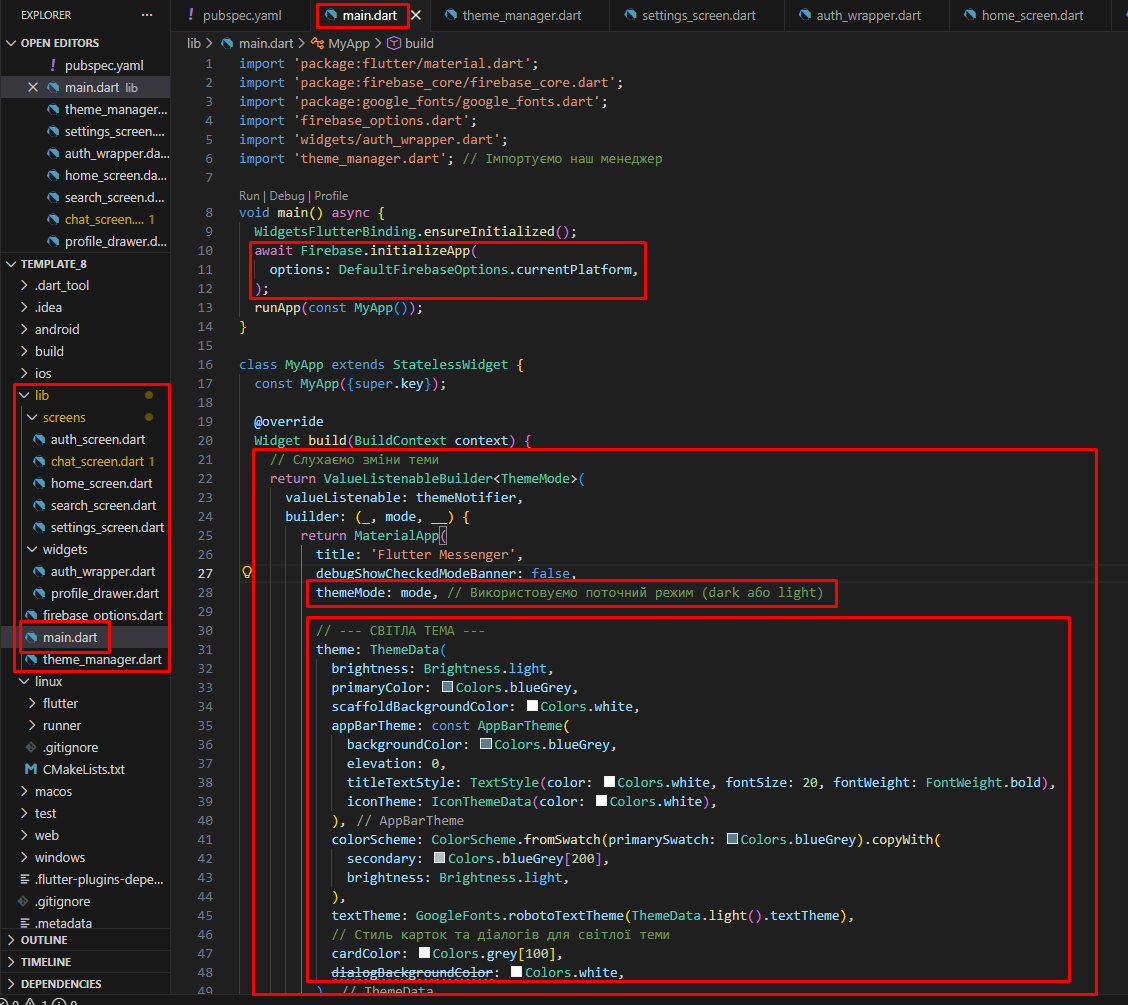
* **State Management:** Використання setState для локального стану UI та StreamBuilder для реактивного стану з бази даних.
* **Asynchronous Programming:** Робота з Future (авторизація, одноразові запити) та Stream (потік повідомлень).
* **NoSQL Database:** Структура колекцій Cloud Firestore (users, chats).
* **Security:** Базова автентифікація (Email/Password) та налаштування правил доступу.
* **Потрібно** установити Node.js та виконати команду npm install -g firebase-tools

**2. Архітектура та ключові файли**

**2.1. Точка входу (main.dart)**

Цей файл відповідає за ініціалізацію Firebase та налаштування глобальної теми.

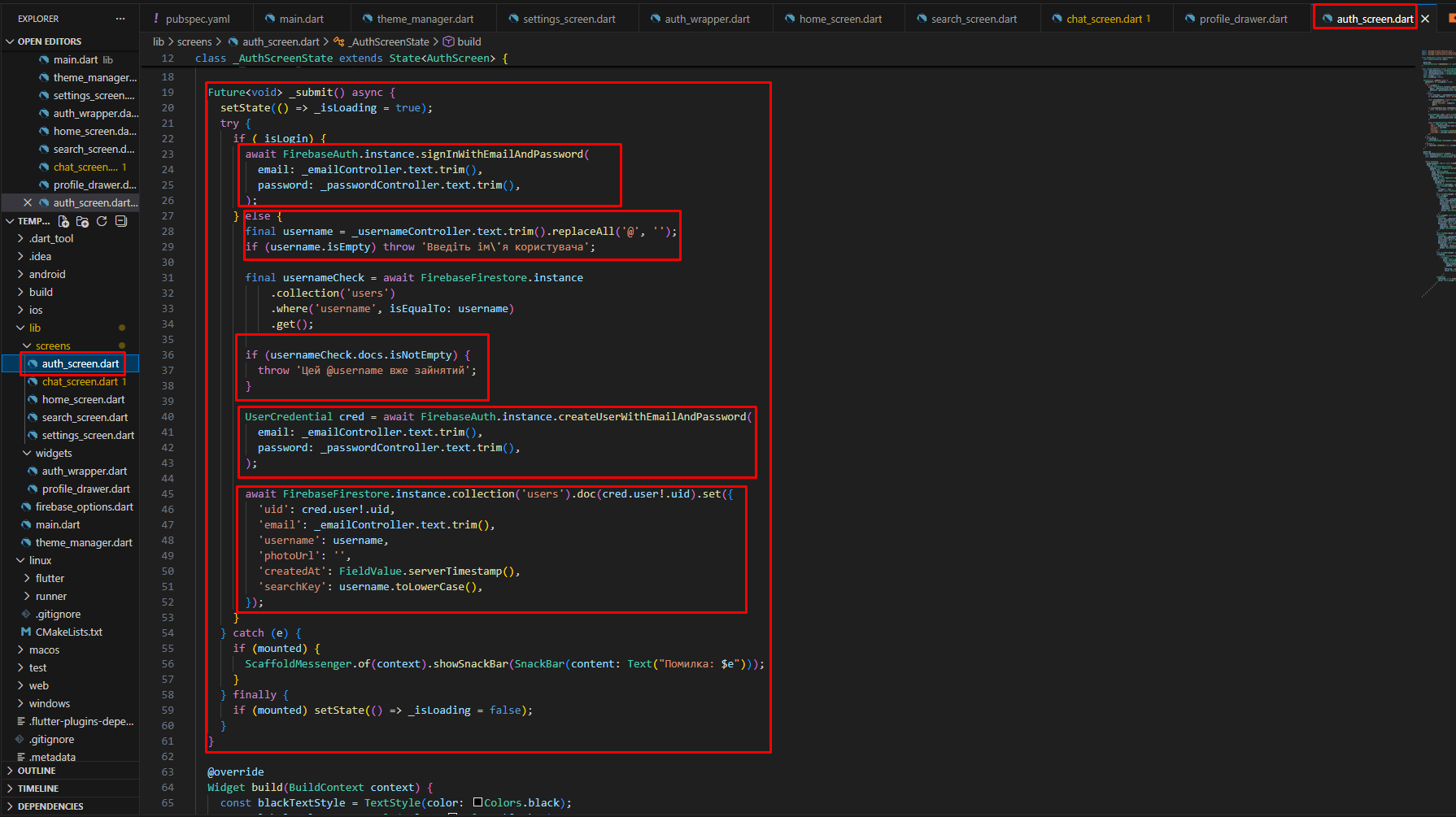
* **Що відбувається:** Виклик Firebase.initializeApp() зв'язує додаток з хмарою. AuthWrapper визначає, який екран показати першим (Логін чи Головна).
* **На що звернути увагу:** Використання ValueListenableBuilder для динамічної зміни теми (світла/темна) без перезавантаження додатку.



**2.2. Екран авторизації (auth\_screen.dart)**

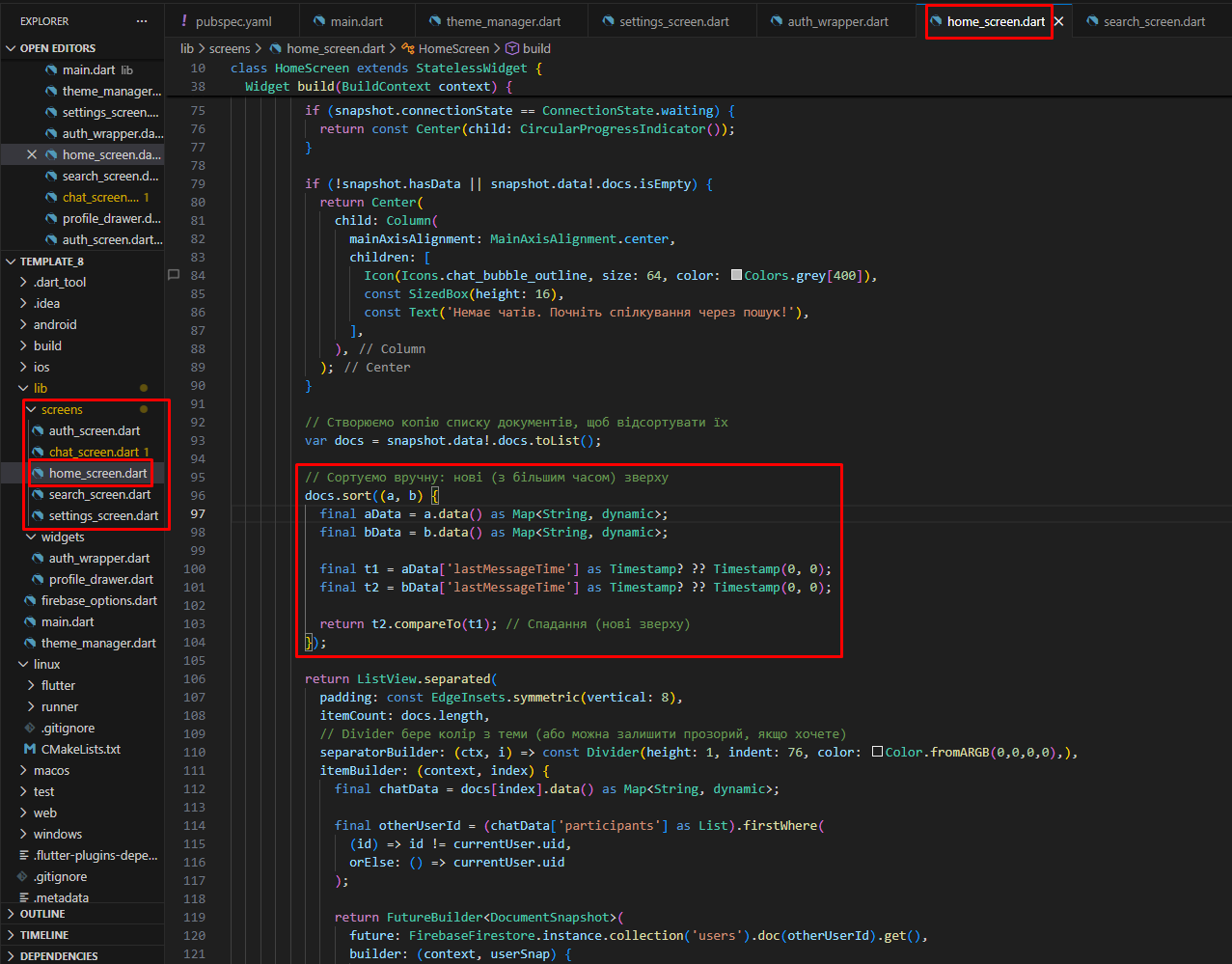
Реалізує логіку "Вхід" та "Реєстрація" на одній формі.

* **Логіка:**
  1. Якщо \_isLogin = true -> викликається signInWithEmailAndPassword.
  2. Якщо \_isLogin = false -> спочатку перевіряється унікальність username через запит до Firestore, потім створюється акаунт в Auth, і фінально записуються дані користувача в колекцію users.
* **Особливість:** Поле searchKey створюється автоматично (в нижньому регістрі) для реалізації пошуку без урахування регістру (case-insensitive search).

**2.3. Головний екран (home\_screen.dart)**

Відображає список активних діалогів користувача.

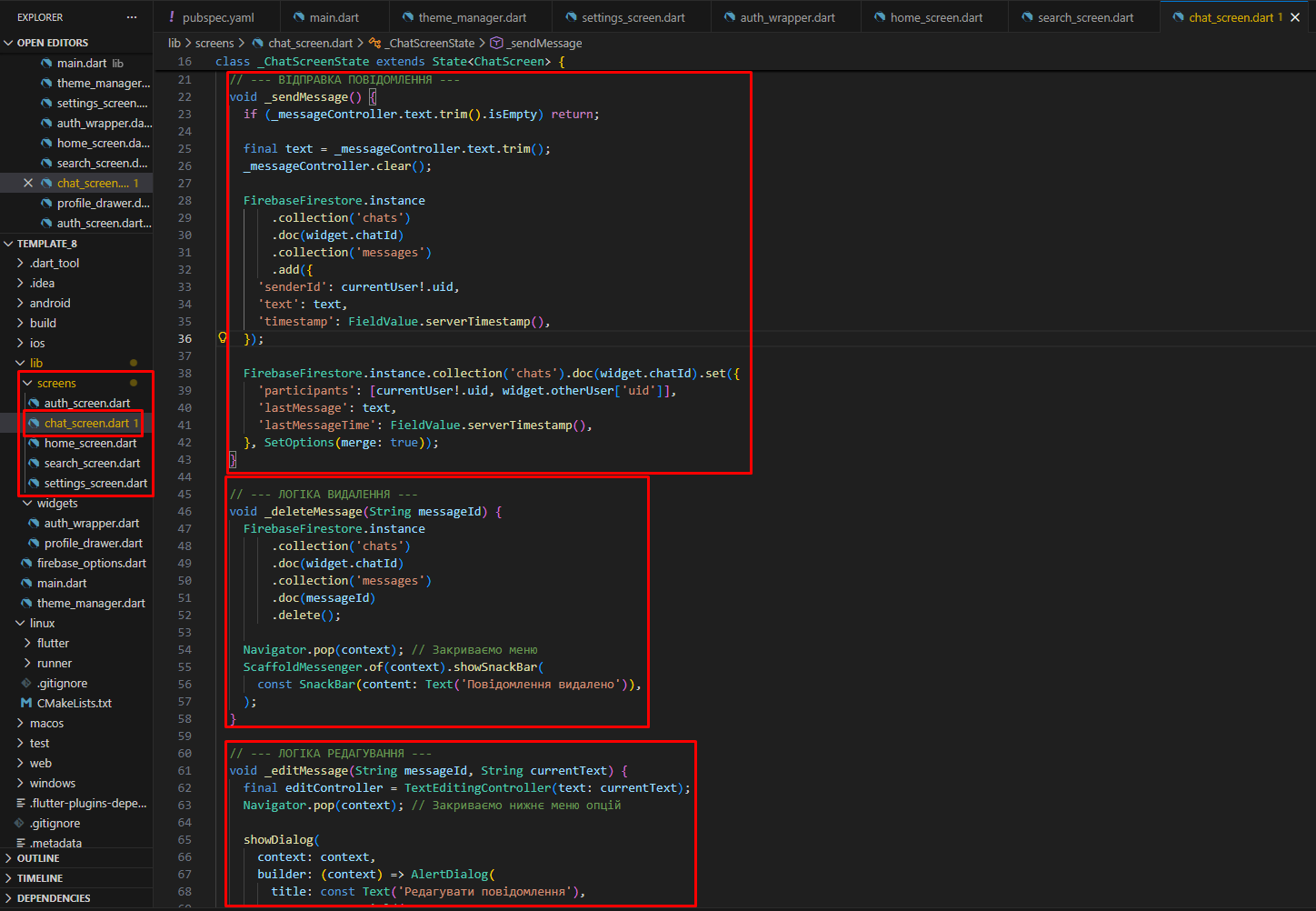
* **Механізм:** StreamBuilder слухає колекцію chats, де масив participants містить uid поточного користувача.
* **Сортування:** Оскільки Firestore має обмеження на складні запити, отримані дані сортуються вручну на клієнті за полем lastMessageTime, щоб підняти нові повідомлення вгору.
* **Візуалізація:** Використовується FutureBuilder всередині ListView для підтягування даних співрозмовника (ім'я, аватар) за його ID.



**2.4. Екран чату (chat\_screen.dart)**

Серце додатку — обмін повідомленнями в реальному часі.

* **Відправка:** Повідомлення додається в підколекцію messages, а в документі самого чату оновлюється lastMessage та час. Це потрібно для коректного відображення списку чатів.
* **Видалення/Редагування:** Реалізовано через спливаюче меню (Bottom Sheet), що викликається довгим натисканням (onLongPress). Зміни миттєво відображаються у всіх учасників завдяки Stream.

**3. Інструкція: Підключення власного Firebase**

Оскільки в наданому коді прописані налаштування демонстраційного проекту (temp8...), студенту необхідно підключити свою власну базу даних для коректної роботи.

**Крок 1. Створення проекту**

1. Перейдіть на сайт [console.firebase.google.com](https://console.firebase.google.com).
2. Натисніть **"Add project" (Додати проект)**.
3. Введіть назву (наприклад, MyChatApp). Google Analytics можна вимкнути для спрощення.

**Відео як створити БД та налаштування кроку 2**

**Крок 2. Налаштування Аутентифікації**

1. У лівому меню консолі оберіть **Build -> Authentication**.
2. Натисніть **"Get started"**.
3. У списку "Sign-in method" оберіть **Email/Password**.
4. Увімкніть перемикач **Enable** і натисніть **Save**.

**Крок 3. Створення Бази Даних**

1. У лівому меню оберіть **Build -> Firestore Database**.
2. Натисніть **"Create database"**.
3. Виберіть локацію сервера (рекомендовано eur3 для Європи або us-central за замовчуванням).
4. **Важливо:** На етапі налаштування правил безпеки оберіть **Start in test mode** (Тестовий режим). Це дозволить записувати дані без складних налаштувань прав доступу на час розробки.
   * *Альтернатива:* Якщо база вже створена, змініть правила у вкладці "Rules" на: allow read, write: if true;

**Відео для налаштування кроку 3**

**Крок 4. Генерація конфігурації (firebase\_options.dart)**

Найпростіший спосіб підключити проект до коду — використати FlutterFire CLI.

P.S. **Потрібно** установити Node.js та виконати команду в терміналі npm install -g firebase-tools

1. Відкрийте термінал у папці проекту (VS Code -> Terminal).
2. Якщо ви не залогінені, введіть: firebase login і дотримуйтесь інструкцій у браузері.
3. Запустіть конфігуратор командою:

flutterfire configure

1. Стрілками виберіть щойно створений вами проект зі списку та натисніть Enter.
2. Виберіть платформи (Android, iOS, Web, macOS) клавішею Space та натисніть Enter.
3. Файл lib/firebase\_options.dart буде автоматично перезаписаний вашими ключами.

**Відео для конфігурації БД Firebase**

**4. Структура Бази Даних**

Для кращого розуміння ментором і студентами, як виглядають дані у хмарі:

**Колекція users (Користувачі):**

// Документ ID: uid\_користувача (від Firebase Auth)

{

"uid": "user\_id\_123",

"username": "alex\_dev",

"email": "alex@gmail.com",

"photoUrl": "base64\_string...", // довгий рядок картинки

"searchKey": "alex\_dev", // для пошуку

"createdAt": Timestamp

}

Колекція chats (Чати):

// Документ ID: user1\_user2 (склеєні ID учасників)

{

"participants": ["user\_id\_1", "user\_id\_2"],

"lastMessage": "Привіт, як справи?",

"lastMessageTime": Timestamp

}

**Підколекція chats/{chatId}/messages (Повідомлення):**

// Документ ID: auto-generated

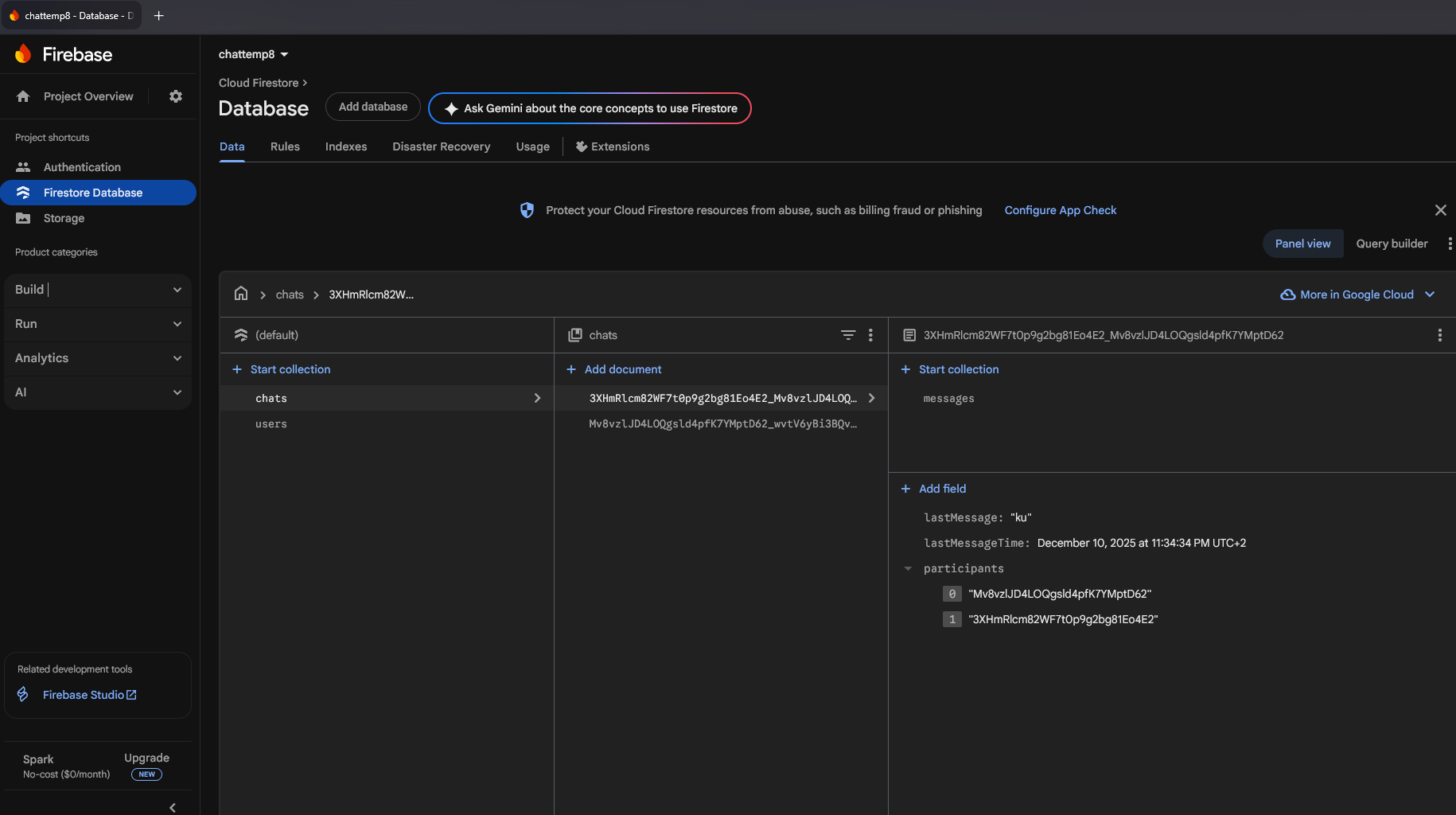
{

"senderId": "user\_id\_1",

"text": "Привіт, як справи?",

"timestamp": Timestamp

}



**Частина 2. Покрокове завдання для учня (15 хвилин)**

**Завдання 1. Персоналізація входу (3 хвилини)**

Змініть колір фону на екрані авторизації, щоб він відповідав вашому улюбленому кольору.

1. Відкрийте файл lib/auth\_screen.dart (або lib/screens/auth\_screen.dart).
2. Знайдіть віджет Scaffold.
3. Змініть параметр backgroundColor. Зараз там Color.fromARGB(255, 50, 62, 71). Спробуйте замінити на Colors.deepPurple або свій варіант.
4. Зробіть **Hot Reload** і перевірте екран входу.

**Завдання 2. Стилізація повідомлень (7 хвилин)**

Змініть вигляд "бульбашок" повідомлень у світлій темі.

1. Відкрийте файл lib/chat\_screen.dart.
2. Знайдіть метод build (всередині ListView.builder).
3. Знайдіть змінні myBubbleColor (колір моїх повідомлень) та otherBubbleColor (колір повідомлень співрозмовника).
4. У тернарному операторі (isDark ? ... : ...) змініть кольори для **світлої теми** (частина після двокрапки).
   * *Наприклад:* Замість Colors.blue[600] поставте Colors.green (стиль WhatsApp).
5. Трохи нижче знайдіть borderRadius у BoxDecoration. Змініть значення Radius.circular(12) на Radius.circular(20), щоб повідомлення стали більш круглими.

**Завдання 3. Зміна тексту-підказки (5 хвилин)**

Змініть текст у полі введення повідомлення.

1. Залишаємось у файлі lib/chat\_screen.dart.
2. Прокрутіть код у самий низ до віджета TextField.
3. Знайдіть властивість decoration -> hintText.
4. Замініть текст 'Повідомлення...' на власний креативний варіант, наприклад: 'Напишіть щось цікаве...'.
5. Перевірте зміни у запущеному додатку.

**Частина 3. Самостійні завдання для студента**

*Увага: У цій частині немає готових рішень "скопіюй-встав". Вам потрібно проаналізувати код і реалізувати логіку самостійно.*

**Завдання №1 (Logic & UI): Підтвердження видалення**

**Опис проблеми:** Зараз, коли ви натискаєте "Видалити" у меню повідомлення, воно зникає миттєво. Це небезпечно, адже користувач може натиснути кнопку випадково. **Завдання:** Додайте діалогове вікно підтвердження (AlertDialog), яке з'являється після натискання кнопки "Видалити". Воно має запитувати: "Ви дійсно хочете видалити це повідомлення?" і мати дві кнопки: "Так" і "Ні". Видаляти повідомлення з бази (Firestore) можна тільки після натискання "Так".

**Підказки:**

* Знайдіть функцію \_showMessageOptions у файлі chat\_screen.dart.
* Логіка видалення зараз викликається в onTap (рядок onTap: () => \_deleteMessage(messageId)).
* Вам потрібно змінити це: замість прямого виклику видалення, викликайте showDialog.
* Функцію \_deleteMessage перемістіть у onPressed кнопки "Так" вашого діалогу.

**Завдання №2 (Feature): Відображення часу повідомлення**

**Опис проблеми:** На головному екрані (HomeScreen) час останнього повідомлення відображається, але всередині чату (ChatScreen) ми бачимо тільки текст повідомлення. **Завдання:** Додайте відображення часу відправки (година:хвилина) маленьким сірим шрифтом у нижньому правому кутку кожної "бульбашки" повідомлення.

**Підказки:**

* У StreamBuilder (chat\_screen.dart) ви маєте доступ до об'єкта msg, який містить поле timestamp.
* Зверніть увагу: timestamp може бути null відразу після відправки (поки сервер не підтвердить запис). Використовуйте перевірку msg['timestamp'] != null.
* Вам знадобиться клас DateFormat (з пакету intl), який вже імпортовано і використовується у home\_screen.dart. Подивіться, як він реалізований там.
* Щоб розмістити час під текстом повідомлення, обгорніть Text повідомлення у Column (всередині "Container-бульбашки") і додайте ще один Text для часу.

**Завдання №3 (UX/Validation): Блокування пустого вводу**

**Опис проблеми:** Зараз кнопка відправки ("літачок") завжди виглядає активною. Якщо натиснути її з пустим полем, нічого не відбувається (код захищений), але візуально користувач цього не розуміє. **Завдання:**

1. Кнопка "літачок" має бути сірого кольору і неактивна (disabled), якщо поле введення пусте.
2. Як тільки користувач ввів хоча б один символ — кнопка стає кольоровою (синьою або зеленою) і активною.

**Підказки:**

* Працюйте у файлі chat\_screen.dart.
* Вам потрібно "слухати" зміни тексту. Для цього у метод initState можна додати \_messageController.addListener(...).
* Створіть нову змінну стану bool \_isComposing = false;. Вона має ставати true, коли \_messageController.text не порожній.
* Використовуйте цю змінну в методі build, щоб змінювати колір кнопки (color: \_isComposing ? ... : ...) та її дію (onPressed: \_isComposing ? \_sendMessage : null). *Примітка: якщо onPressed дорівнює null, кнопка автоматично стає неактивною.*