

Final Project Report – Chat Application Using Flutter & Firebase

- Mahmoud mohamed abdelraheem **ID:2205213**
 - Mohamed ezzat aboelfadl **ID:2206166**
 - Abdelrahman Yousry fathy **ID: 2206193**
-

1. Introduction

This project aims to develop a real-time chat application using **Flutter** for the frontend and multiple **Firebase services** as the backend.

The system demonstrates user authentication, real-time message exchange, chat room management, and online/offline presence tracking.

The technologies used include:

- **Firebase Authentication** – for secure user login and registration
 - **Cloud Firestore** – for storing chat rooms and messages
 - **Firebase Realtime Database** – for real-time presence tracking
 - **Provider** – for state management inside the Flutter application
-

2. Firebase Authentication

The screenshot shows the Firebase Authentication interface in the Firebase console. It displays three user entries in a table:

Identifier	Providers	Created	Signed in	User UID
mahmoudmohamed36...	✉️	Dec 5, 2025	Dec 5, 2025	BCSDHIV0zTHQk7mdtW08p...
mahmoudmohamed36...	✉️	Dec 5, 2025	Dec 5, 2025	ybrFWQzqfkhfenCvjuDyznK...
mahmoudmohamed36...	✉️	Dec 4, 2025	Dec 4, 2025	uR0eZqJug9QdpOva6ON9...

2.1 Justification

Firebase Authentication is used because it provides:

- Secure email-and-password authentication
- No need for manual backend implementation
- Easy integration with other Firebase services
- Automatic handling of user sessions and auth tokens

2.2 Implementation

Users can register using:

```
FirebaseAuth.instance.createUserWithEmailAndPassword(email: email, password: password);
```

Users can sign in using:

```
FirebaseAuth.instance.signInWithEmailAndPassword(email: email, password: password);
```

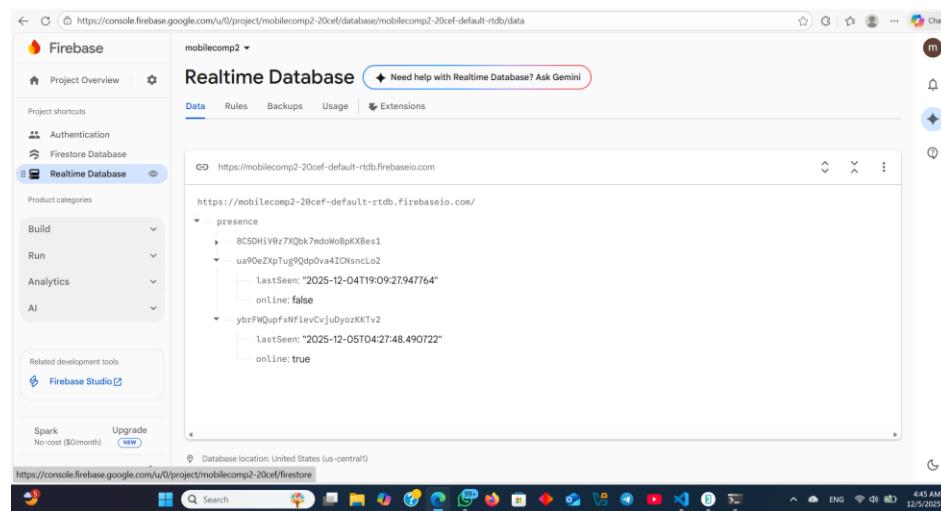
User authentication state is monitored through:

```
_auth.authStateChanges().listen((user) { ... });
```

2.3 Result

- After logging in, the user is automatically redirected to the home screen.
- When logging out, the user returns to the login page.
- The authenticated user's email is displayed in the UI.

3. Firebase Realtime Database



The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with project settings like Project Overview, Authentication, Firestore Database, and Realtime Database (which is selected). The main area shows a list of users under the 'presence' node. Each user entry includes a unique ID, their last seen timestamp, and their current online status (either 'online: true' or 'online: false').

User ID	Last Seen	Online Status
8CSDHIV9z7XbK7mdW0bpX8s1	2025-12-04T19:09:27.947Z	online: false
ua90eZkTug9QdpOva4ICNsncLo2	2025-12-04T23:48:49.072Z	online: true
ybrfWQbpfxNfieCvjuDyozKKTv2	2025-12-05T04:27:48.49072Z	online: true

3.1 Justification

Firebase Realtime Database is used for **presence tracking** because:

- It provides extremely fast real-time updates
- It is optimized for simple data that changes frequently
- Ideal for storing fields like:
 - online: true/false
 - lastSeen: timestamp

3.2 Implementation

Inside auth_service.dart, user presence is updated as follows:

```
_presenceRef.child(user!.uid).set({  
  'online': online,  
  'lastSeen': DateTime.now().toIso8601String(),  
});
```

Presence is updated:

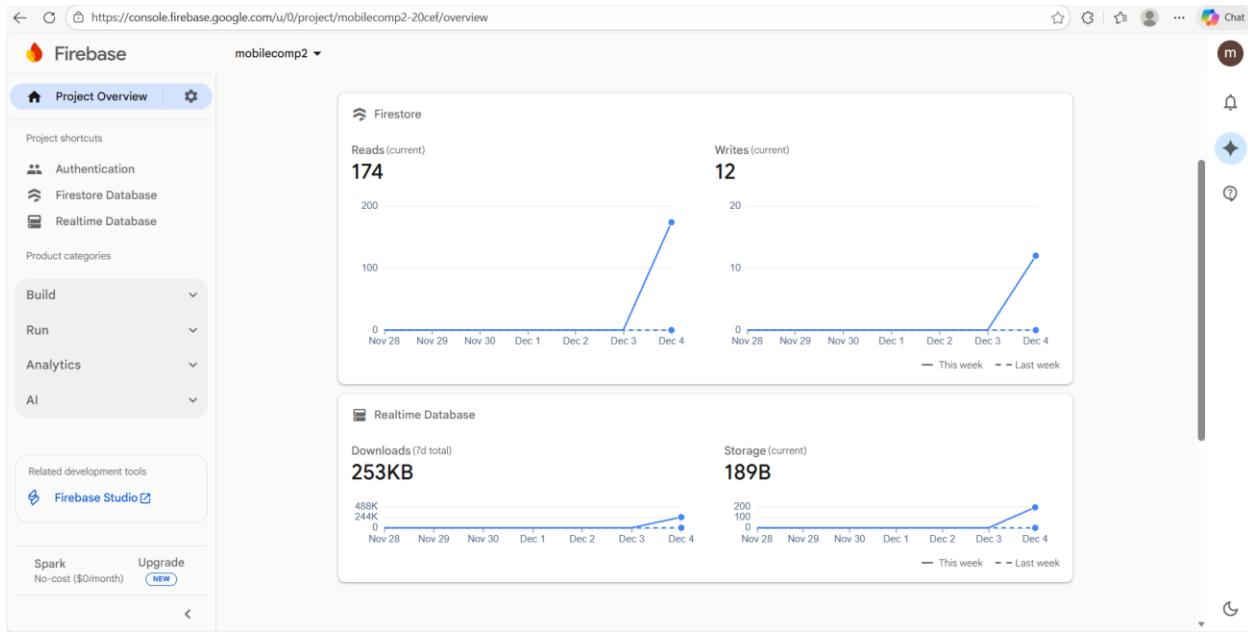
- When the user signs in → online = true
- When the user signs out → online = false

3.3 Result

The home screen displays active users:

2 users online

This number updates in real-time across all running devices.



4. Cloud Firestore Database

The screenshot shows the Cloud Firestore Database interface for the 'mobilecomp2' project. The left sidebar lists 'Project shortcuts' (Authentication, Firestore Database, Realtime Database), 'Product categories' (Build, Run, Analytics, AI), and 'Related development tools' (Firebase Studio). The main area shows a hierarchical database structure under 'chats': '(default)', 'chats', and 'gNj85JCB5C86wdXTJRIS'. The document 'gNj85JCB5C86wdXTJRIS' contains sub-collections 'chats' and 'messages'. The 'messages' collection has one document with fields: 'createdAt' (December 5, 2025 at 4:42:17 AM UTC+2), 'LastMessage' (empty string), and 'name' ('ahmed').

4.1 Justification

Firestore is used for storing chat messages and chat rooms because:

- It supports real-time listeners
- Allows structured and scalable data through Collections and Documents

- Provides powerful queries (orderBy, filters, indexing)
- More appropriate than Realtime Database for complex chat data

4.2 Implementation

Database Structure

chats (Collection)

 └— chatId (Document)

 name: "General Chat"

 createdAt: timestamp

 lastMessage: "Hi"

 lastMessageTime: timestamp

messages (Subcollection)

 └— messageId (Document)

 text: "Hello"

 senderId: "user123"

 senderEmail: "example@gmail.com"

 timestamp: timestamp

Sending a Message

```
FirebaseFirestore.instance
```

```
.collection('chats')
```

```
.doc(chatId)
```

```
.collection('messages')
```

```
.add({
```

```
  'text': text,
```

```
  'senderId': uid,
```

```
  'senderEmail': email,
```

```
'timestamp': DateTime.now(),  
});
```

Also updates the chat preview:

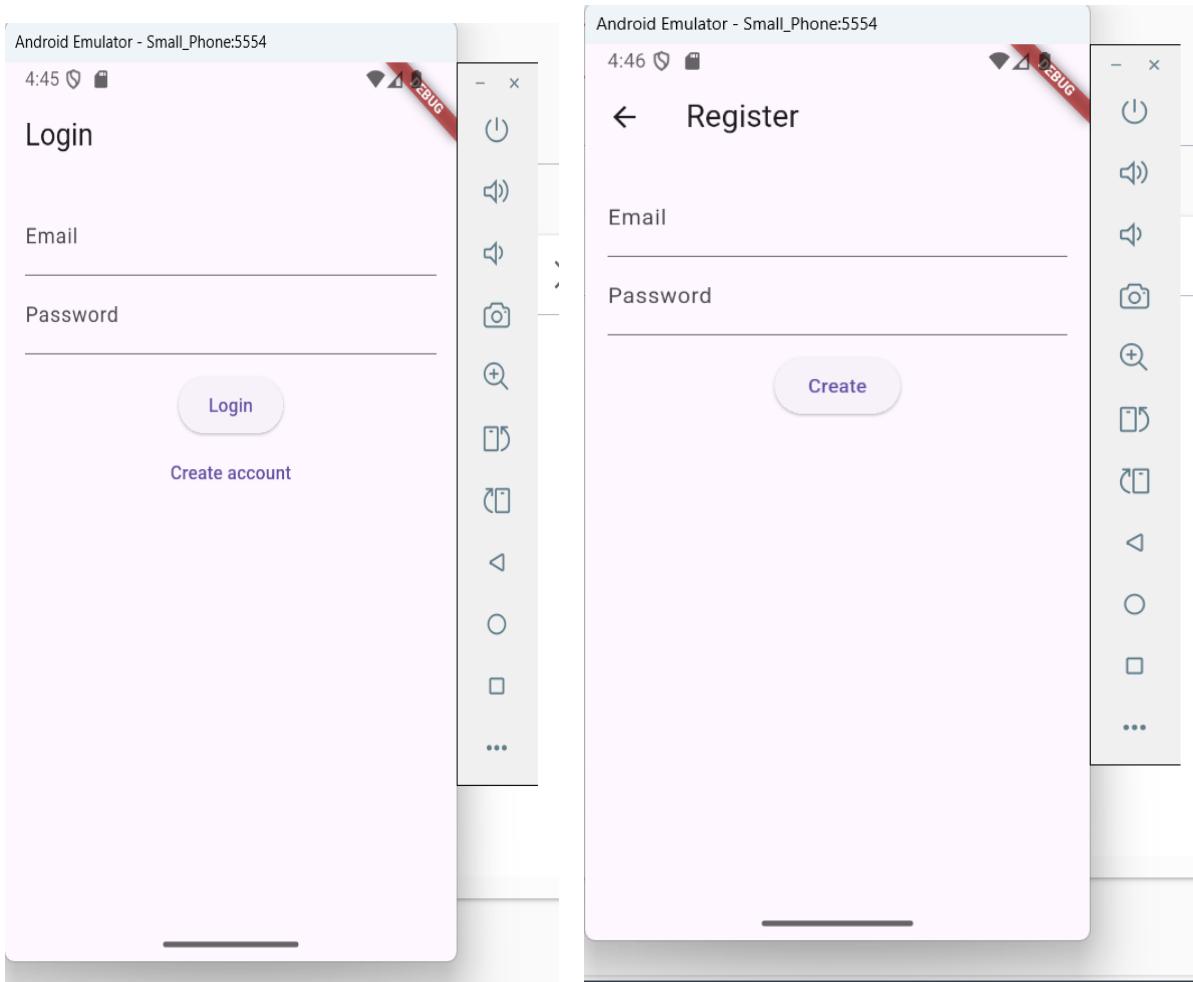
```
FirebaseFirestore.instance.collection('chats').doc(chatId).update({  
  'lastMessage': text,  
  'lastMessageTime': DateTime.now(),  
});
```

Receiving Messages in Real Time

```
StreamBuilder(  
  stream: FirebaseFirestore.instance  
    .collection('chats')  
    .doc(chatId)  
    .collection('messages')  
    .orderBy('timestamp')  
    .snapshots(),  
  ...  
);
```

Messages update instantly on all connected devices.

5. Chat Application Implementation

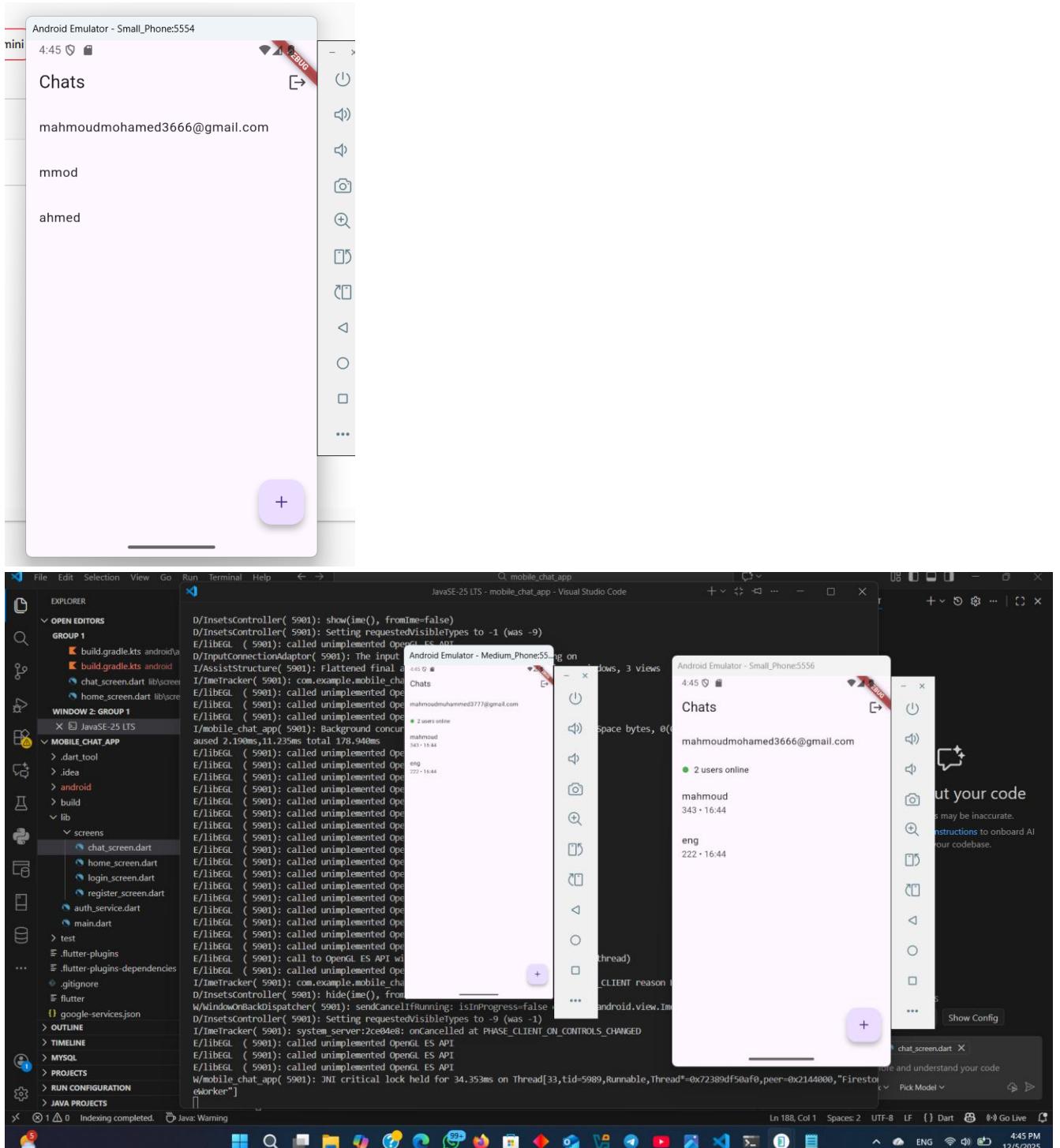


5.1 Home Screen

The Home Screen displays:

- The logged-in user's email
- A list of chat rooms stored in Firestore
- The number of online users from Realtime Database
- A Floating Action Button that allows creating new chat rooms

5.2 Chat Screen



The Chat Screen includes:

- Live streaming of messages using StreamBuilder

- Message bubbles displayed left/right depending on the sender
- A message input field + send button
- Automatic scroll to the latest message
- Updating chat preview (lastMessage) on Firestore

5.3 Multi-Device Real-Time Chat

By running multiple emulators, users can:

- Send messages to each other instantly
 - See presence changes (online/offline) immediately
 - Share chat rooms and message history
-

6. Features Implemented

Feature	Status
Firebase Authentication	✓ Implemented
Realtime Database Presence Tracking	✓ Implemented
Firestore Chat Rooms	✓ Implemented
Real-Time Messages	✓ Implemented
Multi-User / Multi-Device Chat	✓ Implemented
Last Message Preview	✓ Implemented
Online User Counter	✓ Implemented

7. Conclusion

This project successfully demonstrates how Flutter and Firebase can be combined to build a complete real-time chat application without traditional backend servers.

The system uses each Firebase service in an optimal way:

- **Authentication** for login
- **Firestore** for chat logic
- **Realtime Database** for fast presence tracking

The app is scalable, real-time, and easy to extend with features such as:

- Sending images or files
- Group chats
- Notifications
- Typing indicators
- Read receipts