Flutter CHAT APP Project

Prepared for :

Eng. Sara Eldesouky

Prepared by :

Mohamed Osama	2206163
Mohamed Ayman	2206142
Philobatier Ayman	2206161
Ammar Essam	2206211
Omar Elsherbiny	2206206

Project Overview

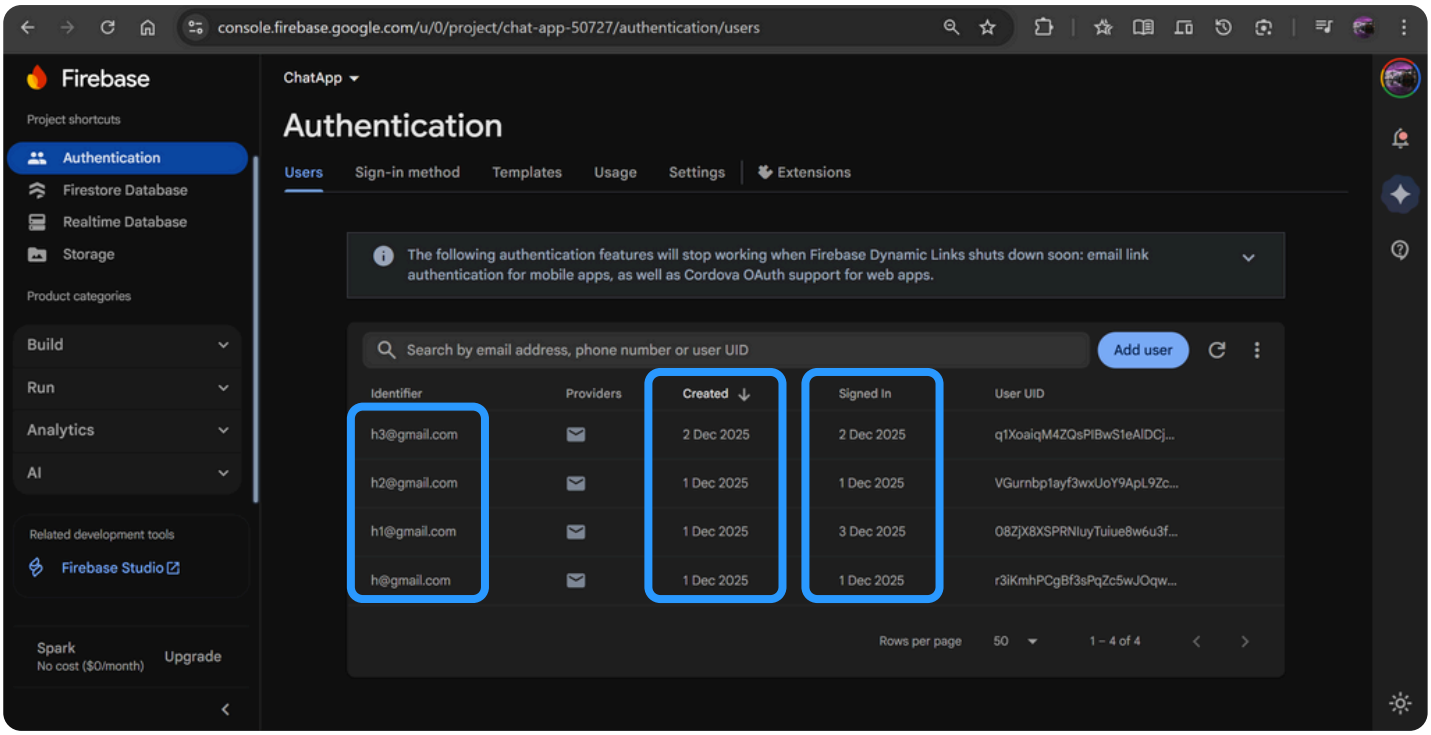
Introduction :

- For our final project, we developed a fully functional real-time chat application using the Flutter framework. The objective was to create a mobile communication tool that mirrors modern messaging platforms like WhatsApp or Telegram. The application solves the problem of seamless, low-latency communication by leveraging the Firebase ecosystem.

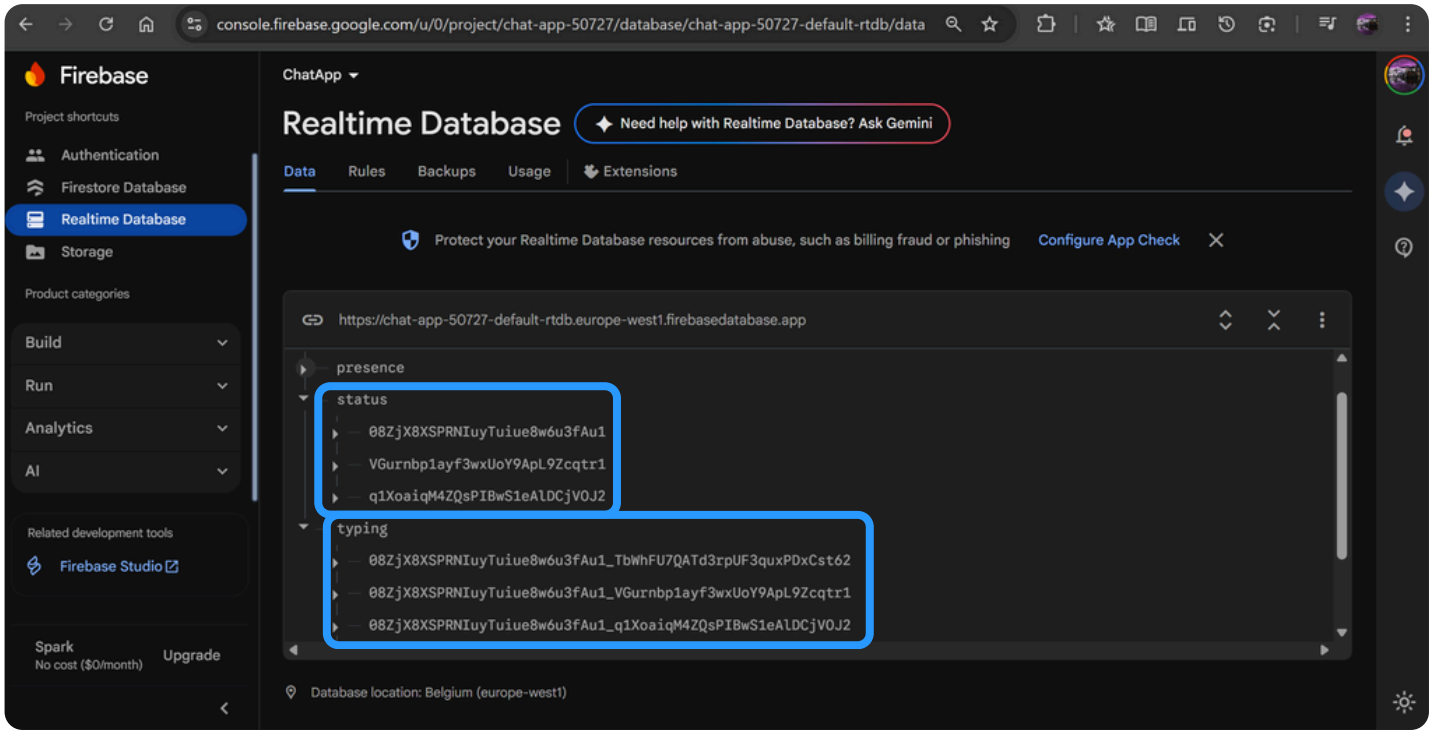
- Key features include secure user authentication, persistent chat history, real-time presence detection (online/offline status), and typing indicators. The app features a modern AMOLED dark interface optimized for mobile devices.

Implementation Details :

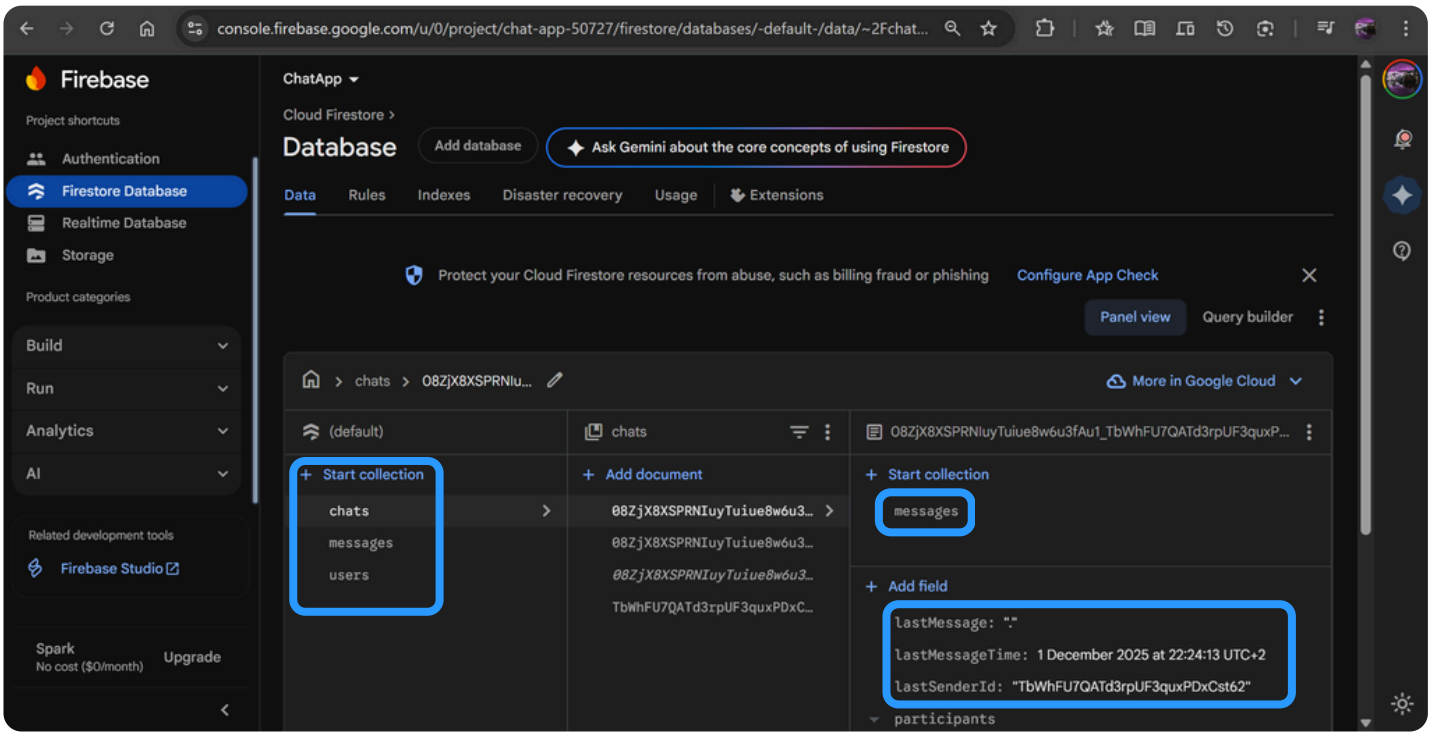
- **Firebase Authentication**
  - We implemented Firebase Authentication to manage user identity securely.
  - Integration: We utilized the firebase_auth package to handle email and password registration and login.
  - Flow: When a user signs up, a unique User ID (UID) is generated. This UID serves as the primary key across our database systems, ensuring data consistency. The app listens to the authStateChanges() stream to automatically persist login sessions (keeping the user logged in even after restarting the app) or redirect unauthenticated users to the Login screen.



- **Firebase Realtime Database**
  - The Realtime Database (RTDB) was specifically employed to handle ephemeral (temporary) data that requires extremely low latency.
  - Usage - Presence System: We used RTDB to track user status. When the app opens, a connection is established, setting a flag status/{uid}/isOnline to true. We utilized the .info/connected feature and onDisconnect() handler to automatically set this flag to false (and update the lastSeen timestamp) if the user loses internet or closes the app.
  - Usage - Typing Indicators: Typing status is stored at typing/{chatRoomId}/{uid}. This data is frequent and temporary; it does not need to be permanently stored, making RTDB the ideal choice.



- **Firebase Firestore Database**
  - Cloud Firestore was implemented as the primary persistent storage solution.
  - Usage - User Profiles: User details (Name, Email, UID) are stored in a users collection.
  - Usage - Chat History: Messages are stored in a hierarchical structure: chats/{chatRoomId}/messages/{messageId}. Each message document contains the text, sender ID, receiver ID, timestamp, and read status.
  - Integration: We used StreamBuilder in Flutter to listen to the Firestore collection. This allows the UI to update instantly whenever a new message is added to the database, without requiring a manual refresh.



- **Chat Application (Integration)**
  - **The chat interface brings all components together:**
  - **Auth: Determines who is sending the message (currentUser).**
  - **Firestore: Loads and saves the message history in real-time.**
  - **Realtime DB: Displays "Online" or "Typing..." under the user's name in the AppBar.**
  - **Logic: When a message is opened, the app updates the read field in Firestore, which triggers a UI update to show blue ticks (Read Receipts).**

home_screen.dart (Offline & Online)

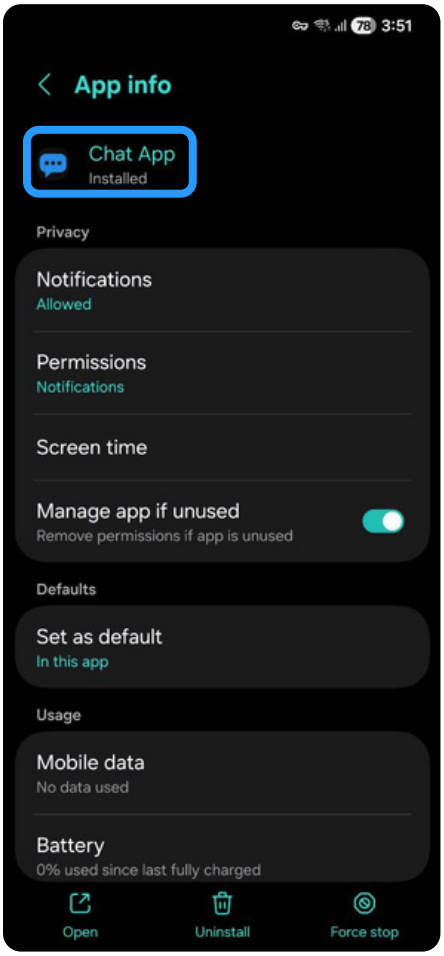
```dart
// Managing User Online Status
Future<void> _setUserStatus(bool isOnline) async {
  // Using Realtime Database for low-latency updates
  final DatabaseReference _database = FirebaseDatabase.instance.ref();

  // Set status to online/offline with a timestamp
  await _database.child('status/${currentUser.uid}').set({
    'isOnline': isOnline,
    'lastSeen': ServerValue.timestamp,
  });

  // Automatically handle disconnection (e.g., app crash or network loss)
  if (isOnline) {
    _database.child('status/${currentUser.uid}').onDisconnect().update({
      'isOnline': false,
      'lastSeen': ServerValue.timestamp,
    });
  }
}
```

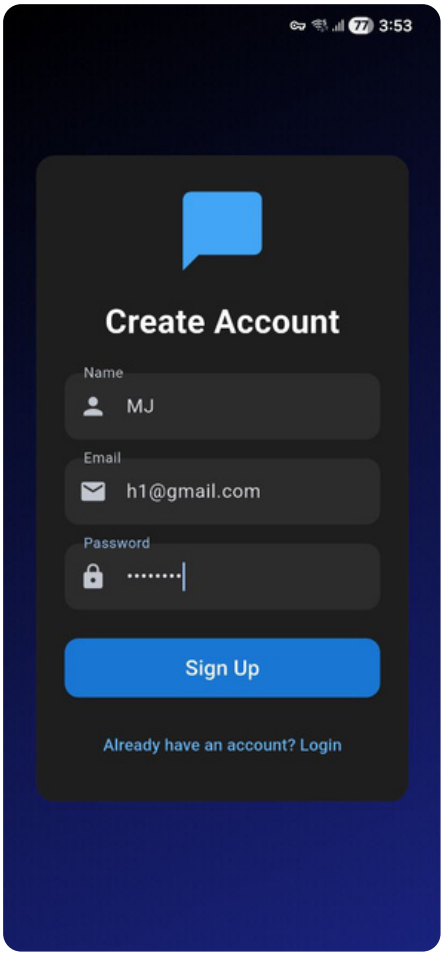
chat_screen.dart (Sending & Retrieving a message)

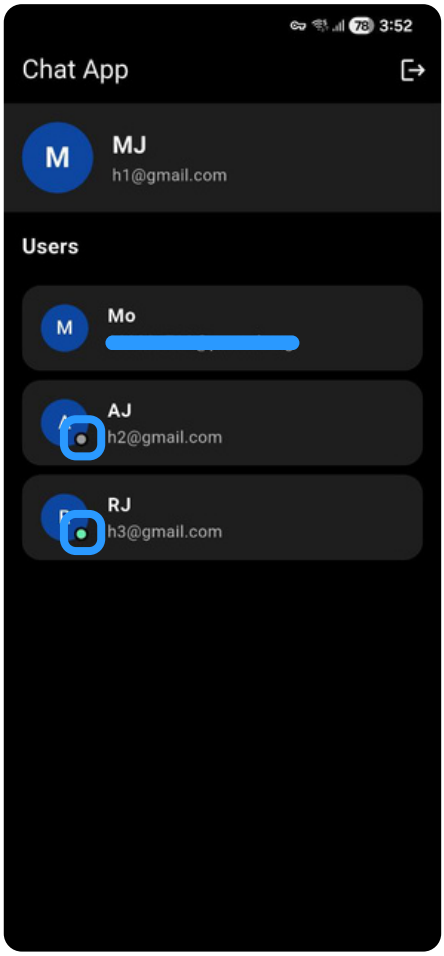
```dart
// Sending a Message to Firestore
Future<void> _sendMessage() async {
  await FirebaseFirestore.instance
      .collection('chats')
      .doc(chatRoomId)
      .collection('messages')
      .add({
    'text': _messageController.text.trim(),
    'senderId': currentUser.uid,
    'receiverId': widget.receiverId,
    'timestamp': FieldValue.serverTimestamp(),
    'read': false, // Initial read status
  });
}

// Listening for New Messages in Real-time
StreamBuilder<QuerySnapshot>(
  stream: FirebaseFirestore.instance
      .collection('chats')
      .doc(chatRoomId)
      .collection('messages')
      .orderBy('timestamp', descending: false) // Ordered chronologically
      .snapshots(),
  builder: (context, snapshot) {
    if (!snapshot.hasData) return CircularProgressIndicator();

    // UI rendering logic here...
  },
)
```

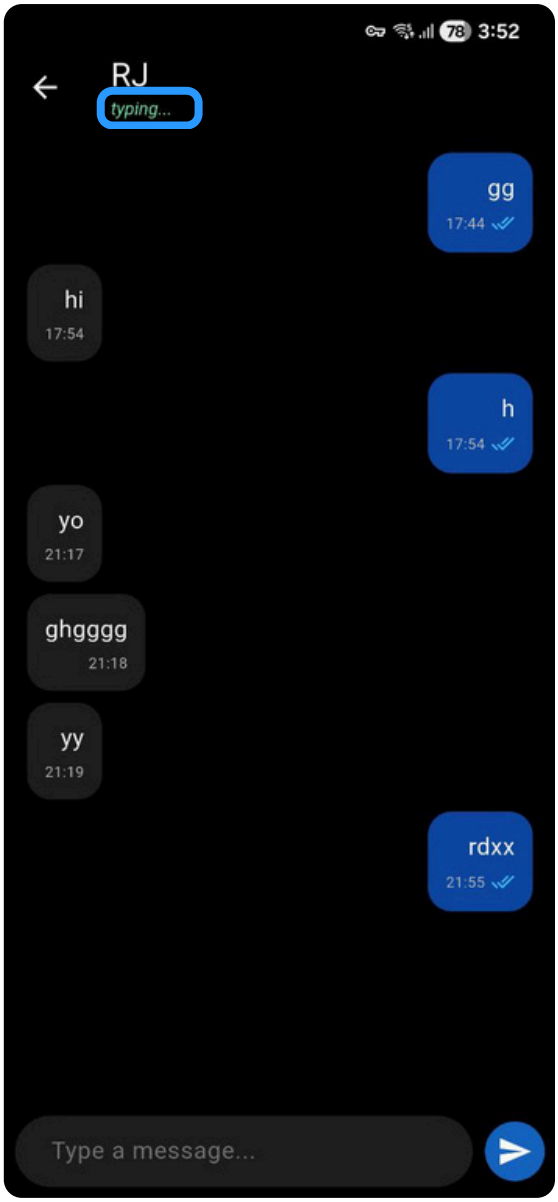
auth_screen.dart (Login & Register)

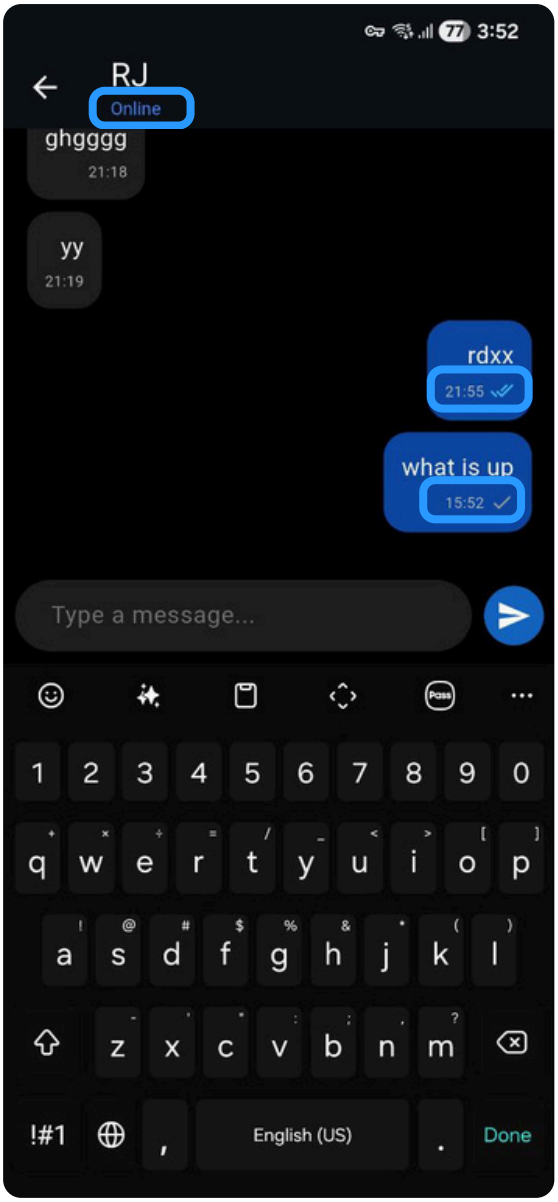
```dart
Future<void> _submitForm() async {
  if (!_formKey.currentState!.validate()) return;

  try {
    UserCredential userCredential;
    if (_isLogin) {
      // Login existing user
      userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text,
      );
    } else {
      // Register new user
      userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: _emailController.text.trim(),
        password: _passwordController.text,
      );

      // Create user profile in Firestore
      await FirebaseFirestore.instance.collection('users').doc(userCredential.user!.uid).set({
        'name': _nameController.text.trim(),
        'email': _emailController.text.trim(),
        'createdAt': FieldValue.serverTimestamp(),
      });
    }
  } on FirebaseAuthException catch (e) {
    // Handle errors (e.g., weak password, email already in use)
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(e.message!)));
  }
}
```


3:51
App info
Chat App
Installed
Privacy
Notifications
Allowed
Permissions
Notifications
Screen time
Manage app if unused
Remove permissions if app is unused
Defaults
Set as default
In this app
Usage
Mobile data
No data used
Battery
0% used since last fully charged
Open
Uninstall
Force stop


3:53
Create Account
Name
MJ
Email
h1@gmail.com
Password
Sign Up
Already have an account? Login


3:52
Chat App
MJ
h1@gmail.com
Users
Mo
AJ
h2@gmail.com
RJ
h3@gmail.com


3:52
RJ
typing...
gg
17:44
hi
17:54
h
17:54
yo
21:17
ghgggg
21:18
yy
21:19
rdxx
21:55
Type a message...


3:52
RJ
Online
ghgggg
21:18
yy
21:19
rdxx
21:55
what is up
15:52
Type a message...
English (US)
Done