# Week 6 Tasks

**Week 6: Final Project and Presentation**

**Task 1:** Plan your final project: Define the scope, requirements,

and timeline.

**Task 2:** Develop the final project incorporating all the skills

learned.

**Task 3:** Test the app thoroughly and fix any bugs.

**Task 4:** Create comprehensive documentation for your project.

**Task 5:** Prepare and deliver a presentation showcasing your final

project.

****Chat App Final Project**  
Developed in **Flutter + Firebase**  
By: Nabiha Akhtar**

# Project Overview

This project is a fully functional chat application built using Flutter. It incorporates Firebase services for authentication and real-time chat functionality. The app follows MVVM architecture and uses Provider for state management.

### ****Introduction****

A real-time chat application built using Flutter and Firebase.  
It supports secure authentication, real-time messaging, message status, and user presence.

**1. Project Setup and Firebase Initialization:**

This step involves creating a new Flutter project and integrating Firebase using the `firebase\_core` package. Firebase is initialized in the `main.dart` file using `Firebase.initializeApp()`. Configuration is set up using `firebase\_options.dart` generated by the FlutterFire CLI.

**2. User Interface Design (Login, Signup, Home, Chat Screens):**

The app has a clean UI built using Flutter widgets. Major screens include:

* Login Screen – For existing users to sign in.
* Signup Screen- For new users to create accounts.
* Home Screen – Displays the list of chats or users.
* Chat Screen - Shows real-time messaging.

**3. Firebase Authentication with Login and Signup:**

Uses `firebase\_auth` to allow users to:

1. Sign up with email and password.
2. Log in and maintain authentication state.
3. Handle errors like incorrect credentials.
4. **Firestore Database Integration for Message Storage:**

Messages and user data are stored in Cloud Firestore collections.

Each chat has a document with metadata.

Messages are stored in subcollections under each chat.

**5. Sending and Receiving Messages:**

Users can send text messages using a `TextField` and `send` button. Messages are saved to Firestore and received in real-time using `StreamBuilder`.

**6. Realtime Chat Functionality using Firestore Streams:**

Chat messages are streamed in real time using `FirebaseFirestore.instance.collection(...).snapshots()` and displayed instantly in the UI using `StreamBuilder`.

**7. App Lifecycle Handling using WidgetsBindingObserver:**

App state is monitored using `WidgetsBindingObserver` to:

* Track user presence (online/offline).
* Update last seen timestamp when the app is paused or resumed.

**8. Typing Indicator during Conversation:**

When a user is typing, a flag in Firestore is updated.

This is shown to the other user in real-time (e.g., “User is typing…”).

**9. Firestore Indexing for Optimized Querying:**

Firestore automatically creates single-field indexes. For complex queries (e.g., ordering by timestamp and filtering), composite indexes are manually added in Firebase console for performance and scalability.

# Key Features Covered

✅ Project Setup and Firebase Initialization

✅ User Interface Design (Login, Signup, Home, Chat Screens)

✅ Firebase Authentication with Login and Signup

✅ Firestore Database Integration for message storage

✅ Sending and Receiving Messages

✅ Realtime Chat Functionality using Firestore streams

✅ Last Message and Timestamp display in chat list

✅ Unread Message Counter per chat

✅ Message Read Status (Mark as Read)

✅ Online and Last Seen Status tracking

✅ App Lifecycle handling using WidgetsBindingObserver

✅ Typing Indicator during conversation

✅ Firestore Indexing for optimized querying

✅ Chat Pagination to load old messages

✅ Emoji Picker integration in chat screen

✅ Block/Unblock functionality

✅ Input Validation on forms (email, password)

✅ Provider for State Management

✅ MVVM Architecture Pattern

# Conclusion

The chat application is designed to provide smooth real-time messaging using Firebase. With the implementation of advanced features like typing indicator and online presence, it ensures a complete user experience. The project is suitable as a final year project or portfolio app for learning advanced Flutter with Firebase.