One-To-one Chat app

Documentation

Version 1.3

Last Updated on 3rd April 2021

1

TABLE OF CONTENTS

TABLE OF CONTENTS

PART 1: ABOUT APPLICATION

Application Overview

FIBERCHAT, is an One-To-One Realtime Chatting app like Whatsapp, Viber etc.. It has got all the functionalities like Mobile OTP Login, Chat with User without saving number, read contacts, showing status (istyping, online, Last seen at..) , Double tick on message read by peer, unread message count, hide & Block user, Share media, Set wallpaper & Other cool features.

Most importantly, it has End-to-End Encryption features which make it secure and safe to use application for users. Conversations between users can only be read by themselves. No one including admin can see the sensitive information from the server also since it has got encryption using different algorithms.

Checkout the Demo app to experience the amazing features.

Application Package

FIBERCHAT package has basically 2 applications:

1. Fiberchat User App (Android):

Fiberchat app (***app package name:*** *com.fiberchat.fiberchat*) is for android users

1. Fiberchat User App (iOS):

Fiberchat app (***bundle ID:*** *com.fiberchat.fiberchat*) is for ios users .

Hardware & Software Requirement

Hardware required:

iOS devices (iPhone 4S or newer) and ARM Android devices

Software required:

A android smart phone with operating system Jelly Bean, v16, 4.1.x or newer, and iOS smartphone with iOS 8 or newer. or IOS operating system (OS).
To download and use the functionalities of “Fiberchat” mobile application, you require an Internet connection in your mobile phone.

APIs / Technologies Used

Fiberchat app is built with the following technologies-

a. **Google's Flutter** as app development framework.
b. **Firebase Cloud Firestore** as database for mobile apps.
c. **Firebase Storage** as cloud storage for mobile apps.
d. **Firebase Authentication** for Mobile OTP authentication.
e. **Firebase Messaging** for sending Push Notifications.
f. **Firebase Cloud Functions** to invoke Push Notification.
g. **AGORA SDK** for audio & video calls.
h. **GIPHY GIF** API for Gif in Chat

App Permissions

Fiberchat app requires following Users Permission.

a. **Internet Connectivity** to run app.
b. **Read Storage** to upload Media files.
c. **Write Storage** to save a file on storage.
d. **Read Contacts** to find Users available to chat.
e. **Microphone** access to make calls & record audio.
f. **Camera** access to make video calls.

PART 2: FILES & FOLDERS UNDERSTANDING

2.1. Android Folder (…/ fiberchat /android)

This folder contains all technical information about the android part of the app since it is a cross platform package. Hence, editable information like- App package Name, App name, App logo are inside this folder.

There are hundreds of other files are inside the folder that are native android compiled. It is advisable not to alter those files.

2.2. Assets Folder (…/ fiberchat /assets)

All graphic assets, banners, animation file are inside this folder. You can compare the asset/ images within app and just replace keeping the exact name & format of the asset/image.

2.3. Build Folder (…/ fiberchat /build)

This is a temporary folder which is generated when we compile and run the app. It contains release apk. , app bundle, debug apk. Final app file should be picked from this folder to upload into Playstore.

You can delete this folder anytime and rebuilt this folder anytime using command **flutter run** OR **flutter build appbundle** from the terminal keeping fiberchat folder open in terminal.

2.4. Fonts Folder (…/ fiberchat /fonts)

This folder contains all the Font families and individual font files inside them. You can remove or add fonts in this folder but you have to register the font in **(…/** fiberchat **/pubspec.yaml)** file as per section 4.6.

2.5. Functions folder (…/ fiberchat /functions) optional

This folder contains all the Cloud functions & Server Programms for your app if you add some additional functions in app like – Push Notifications.

2.6. Gen Folder (…/ fiberchat /gen)

You don't need to do anything with folder. This is an auto generated folder contains some of the system information like – Crash Reports etc.

2.7. iOS Folder (…/ fiberchat /ios)

This is another special folder which contains the iOS configuration & code of the flutter application. You can change the iOS splash screen , name, bundle name etc. from this folder.

2.8. Lib Folder (…/ fiberchat /lib)

This is the main folder where you will find Source code of all the Screens. Inside this file there is a **main.dart** file which is the initial file of the app which executes at first when the app is compiled & run.

2.9. Test Folder (…/ fiberchat /test)

This folder is an auto-generated folder containing test widget. You don't need to modify / delete this folder.

2.10. Pubspec.yaml File(…/ fiberchat /pubspec.yaml)

This is the register of all 3rd party Plugins, Fonts, Assets entry. This file is the one of the most crucial file of this package.
It is suggested not to modify/change this file code if you are not 100% sure about the code as this leads app crashing and unable to compile also.

2.11. Extra Files

All these are the extra system generated files that you don't need to bother & don't ever delete them:
(…/fiberchat/.flutter-plugins)
(…/fiberchat/.flutter-plugins-dependencies)
(…/fiberchat/.gitignore)
(…/fiberchat/firebase.json)
(…/fiberchat/pubspec.lock)
(…/fiberchat/README.md)
(…/fiberchat/fiberchat.iml)
(…/fiberchat/Chnage_Log.md)
(…/fiberchat/RELEASE_NOTES.md)

PART 3: APP DEBUGGING

Prerequisite

Basic knowledge of flutter. Follow the official flutter get started guide https://flutter.dev/docs/get-started/
System requirement: Android apps can be configured on Windows, macOS or Linux. macOS is mandatory for iOS app configurations.
Basic Knowledge of Android studio or Visual Studio and Xcode.
Firebase

Downloading & Installing FLUTTER

Follow the flutter official docs to install Flutter on your system https://flutter.dev/docs/get-started/install/

Firebase integration

Create a project on firebase:
Add android and iOS applications with the correct unique package name and bundleID.
Follow the steps provided while creating the app. Replace the google-services.json and GoogleService-Info.plist files at the exact locations mentioned on firebase.
Finally, replace all the occurrence of bundle ID and package name inside the android and iOS folder.
Run the app **"flutter run"** to verify the configuration.
To know more – https://firebase.google.com/docs/flutter/setup

For Phone Auth :
Go to the Authentication tab from the Firebase dashboard side nav.
Under the Sign-in method Enable the Phone method.
For more info :
android : https://firebase.google.com/docs/auth/android/phone-auth
iOS: https://firebase.google.com/docs/auth/ios/phone-auth
For more details on Android & Firebase integration check here:
https://help.deligence.com/knowledgebase/android-firebase-configuration/

Configure Database
Go to the database page from the side navbar.
Create a Cloud Firestore database. Select the region. To know more https://firebase.google.com/docs/projects/locations

Configure Storage
Go to the storage page from the side navbar.
Click on get storage to configure storage.

IDE Setup

Install VS Code.
VS Code is a lightweight editor with Flutter app execution and debug support.
https://flutter.dev/docs/get-started/editor?tab=vscode

Setting up Source Code

Download ZIP File Source Code. Drag & Drop Downloaded Source Code Folder At VS Code Icon to open project OR Select downloaded folder and choose open with VS Code.

Section-1: ANDROID Configuration

1.A. Edit theme

1.A.1 Change Overall TextStyle

1. Place your Font File (**.otf** or **.ttf**) in **<path_to_project_root>/lib/fonts.** folder.
2. Register font path at **<path_to_project_root>/pubspec.yaml**. Paste the font name at the bottom font section.

Example-
```
- family: FONTNAME
  fonts:
    - asset: fonts/FONTNAME.ttf
```

4. Save 'pubsepc.yaml' file using CTRL+S to update new font.
5. Open **<path_to_project_root>/lib/main.dart** and inside themeData provide this-

```
fontFamily: 'FONTNAME'
```

1.A.2 Change Color Values

1. Change Color values in **<path_to_project_root>/lib/app_constant.dart** inside you will find all the color values used in the project. Just replace the colour value with your desired one.
2. Save the File using CTRL+S.

1.A.3 Change app Name Label

1. Provide app name in "android:label" field inside **<path_to_project_root>/android/app/src/main/AndroidMainfest.xml** and also change name in **<path_to_project_root>/lib/main.dart; <path_to_project_root>/lib/app_constant.dart**

1.B. Edit app Details

1.B.1 Change app Icon

1. Open **<path_to_project_root>/android/app/src/main/res/** and replace all Icons named “ic_launcher.png” as per respective sizes within different sizes folder.
2. Open **<path_to_project_root>/images/ ,** replace “applogo.png”.
3. Go to lib/constants/app_constants.dart and paste the same path to AppLogo String for the login screen.

1.B.2 Change Splash Screen (if any)

2. Open **<path_to_project_root>/images/ ,** replace “splashscreen.png”.
3. Go to lib/constants/app_constants.dart and paste the same path to SplashScreenPath String for the login screen.

1.B.3 Change Onboarding Screens (if any)

1. This project does not have onboarding screen.

1.B.4 Change App Version

Replace “version number + version code” in all these files:

1. **<path_to_project_root>/pubspec.yaml**
2. **<path_to_project_root>/lib/main.dart**
3. **<path_to_project_root>/android/local.properties**

1.B.5 Change Other Config

1. Open **<path_to_project_root>/lib/app_constant.dart** and edit other prefrences.

Section-2: iOS Configuration

2.A. Edit App Details

2.A.1 Change Overall TextStyle

1. Place your Font File (**.otf** or **.ttf**) in **<path_to_project_root>/lib/fonts.** folder.
2. Register font path at **<path_to_project_root>/pubspec.yaml**. Paste the font name at the bottom font section.

Example-
```
- family: FONTNAME
  fonts:
    - asset: fonts/FONTNAME.ttf
```

4. Save 'pubsepc.yaml' file using CTRL+S to update new font.
5. Open **<path_to_project_root>/lib/main.dart** and inside themeData provide this-

```
fontFamily: 'FONTNAME'
```

2.A.2 Change Color Values

1. Change Color values in **<path_to_project_root>/lib/app_constant.dart** inside you will find all the color values used in the project. Just replace the colour value with your desired one.
2. Save the File using CTRL+S.

2.A.3 Change app Name Label

1. Provide app name inside **<path_to_project_root>/ios/runner/info.plist;**

PART 4: COMPILE & PUBLISH APP (Android)

4.1 App Signing

1. Every app needs to be signed by a keystore which contains signing information by the developer/company.
2. The keystore file is present in
<path_to_project_root>/android/app/key.jks
3. The keystore should be same for each app update for respective apps. However, if you want to change keystore, please contact Play Console support team.
4. The default Keystore details provided by developer/company is shared in Secret Information section.

4.2 Build Debug App (Test mode)

1. To test a Flutter app a real device or simulator must be connected with the computer.
2. Open project root folder at the terminal.
3. Execute the command "*flutter run*" to build a debug app. Press "r" to hot reload and "R" to hot restart the debug app while making changes to the code.

4.3 Build Release App

1. Open project root folder at the terminal.
2. Once app debugging is successful, execute command "*flutter build appbundle*" to generate an app bundle to be submitted for playstore. You can also generate release ".apk" file to install directly on the smart phone by executing command "flutter run –release".

4.4 Publish New App/ Update in Google Playstore

1. Once app bundle is generated login to your Play Console using credentials provided in the Secret Information section.
2. Select the app from the Play Console for which the app bundle is to be submitted.
3. Navigate to "Manage Release" section from Play Console Dashboard. Click on "Create Release" and upload the generated app bundle. Version number will be automatically fetched and displayed for this release.
4. Review thoroughly by clicking on "REVIEW" button, after that click on "START ROLLOUT TO PRODUCTION". Hence, your request for the app update is successfully submitted and it will be live in the Playstore once approved by the Google Playstore Team. You will be notified via email.

PART 5: COMPILE & PUBLISH APP (iOS)

5.1 Installing/updating Pod file

Run following command in iOS path:

pod install
pod update
pod repo update
pod install --repo-update

5.2 Build Debug App (Test mode)

1. To test a Flutter app a real device or simulator must be connected with the computer.
2. Open project root folder at the terminal.
3. Execute the command "*flutter run*" to build a debug app. Press "r" to hot reload and "R" to hot restart the debug app while making changes to the code.

5.3 Build Release App

Please refer official documentation step wise provided by flutter team. It is updated and much useful:
https://flutter.dev/docs/deployment/ios

5.4 Publish in Apple appstore

Please refer official documentation step wise provided by flutter team. It is updated and much useful:
https://flutter.dev/docs/deployment/ios

PART 6: MONITOR - APP, FIREBASE & APIs

Monitor App using Firebase SDK

To monitor your Firebase project login to Firebase Console using credentials provided at the Secret Information section.

6.1 Firebase Console Basics

1. Flutter app is backed by Firebase backend products like – **Cloud Firestore** as No-SQL database, **Firebase Storage** as cloud storage for app, **Firebase Authentication** for user mobile authentication, **Firebase Functions** as cloud functions, **Firebase Messaging** for push notifications, **Remote Config** for app version control and app links, **Stream View** for real time users count across the globe.

6.2 Manage Firebase Billing

1. Firebase billing is the monthly billing for the usage of the app consisting of the Firebase Products based on following key points:
   a. Documents read, write and delete counts.
   b. Amount of GB stored in Firebase Storage and its bandwidth.
   c. Cloud Functions invoked by user activities.
2. By default Firebase billing is set to SPARK PLAN for each project that is under free tier quota. For more pricing information visit https://firebase.google.com/docs/firestore/pricing.
3. Once the free tier quota is consumed Firebase restricts and interrupts the usage of the app. To continue, it should be upgraded to the BLAZE PLAN which bills you monthly depending upon the app usage.
4. Real time read, write and delete count can be monitored through "usage" tab in **Database** and **Storage** section.
5. You can preset the monthly budget for Firebase billing which reminds you once your usage limit is about to exceed your preset amount.

6.3 App Engine – Cloud Console

1. Every Firebase project is also a Google Cloud Project (GCP), so it can be monitored through Google Cloud Console other than Firebase Console.

6.4 App Version Control

Change the version of the app at the document present in cloud firestore database (path: *version/userapp/).* Also provide the url

Monitor using Google Play Console

To monitor your Google Play Console Account, login to Play Console using credentials provided at the Secret Information section.

6.5 Manage Developer Info

1. Go to Settings>Developer Account, manage account details, users and permissions, activity log, API access, linked accounts, payment settings and bench marking preferences.

6.6 Publish/Unpublish App

1. Publish app as per section 5.4.
2. In the current version of Google Play Console as of (August 2020) here is the procedure of unpublishing any app.
   a. Choose the app you wish to unpublish.
   b. Go to Store presence > Pricing & distribution.
   c. You will see unpublish button in the app availability section. After clicking Unpublish a dialog will pop up, confirm and save the changes.

6.7 Store Listing

1. To manage the listing information of your app in Google Playstore go to **Store presence** to edit store listings, store listing experiments, custom store listings, pricing & distribution, content rating, app content.

6.8 Manage Release

1. In this section, all your app versions are listed along with version code and version number with real time statistics.

6.9 User Metrics

1. App installs, Active devices, device type, uninstalls all these datas will be shown under **Statistics** section and **User acquisition** section in Google Play Console.

6.10 User Rating & Review

1. Find ratings, review analysis, reviews under **User feedback** section in Google Play Console. You can also respond to the user reviews.

PART 7: 3RD PARTY INTEGRATIONS

Admob Integration

To configure Admob showing banner, interstitial , Video ad in this application. You need to do perform following tasks:

1. Replace admob id values inside **lib/constant/app_constant.dart**:

2. Change the admob Id for android & iOS as per the official plugin : https://pub.dev/packages/admob_flutter

Agora Integration

To configure Agora for Audio & Video calls in this application. You need to have Agora APP ID from the Agora Console. Please Agora is not a free usage API and usage will be chargeable by Agora. To get an APP ID easily, please refer the following steps:

1.**Agora's developer account:** To build an app that is using Agora's SDK you will need an AppID. So to grab your own AppID go to https://console.agora.io/ and quickly create an account and log in to your dashboard.

2. Now navigate to the project management tab and click on the create button. This will let you give a suitable name for your project and give you an AppID linked with it.

3. Open the source Code in your preferable editor and Go to: *lib/constants/app_constant.dart* and set your agora APP ID there.

GIPHY GIF Integration

To setup Giphy API for GIF, get the Giphy API Key from their official site and paste the same code in *lib/constants/app_constant.dart* .

Push Notifications Services

To setup push Notifications for Fiberchat, you need to host the Cloud function in any server. Which triggers the push notification to user devices on event. In this case we have used Firebase Functions API provided in Firebase console.

1. Upload Cloud Functions present in the Functions folder in the application package you have purchased .
2. Login to firebase console in a browser & setup the firebase function required for the first time only.
3. Login to firebase CLI using *"firebase login"* command.
4. Run *"firebase deploy –only functions".*
5. *Please Don't re-write the index.js as it will erase all the push notification codes.*
6. Your functions shall be deployed successfully.
7. You can test if its working perfectly.

Mutli-Language Translation

Localization Support has been added to Fiberchatv_1.0.7 onwards. To add a local language in the app, just follow these simple steps:

1. Clone & Translate the String values present inside the lib/Services/Localization/json_languages/en.json and name the file [LANGUAGE_CODE].json
2. Register all those file entry in pubspec.yaml.
3. Include the language - [LANGUAGE_CODE].json wherever the en.json is present inside lib/Services/Localization/demo_localization.dart & language_constants.dart & language.dart.
4. Once you successfully add the languages, just call the function **setLocale(**LANGUAGE_CODE**)** present in inside lib/Services/Localization/language_constants.dart . Call this function wherever your user need to change the language in app.
5. You are done with language translation.