

# **JABBER**

A Mini Project Report

submitted for the partial fulfillment for the award of degree of

**BACHELOR OF COMPUTER APPLICATIONS**

submitted

by

**K. CHITRAARASU (A19CADB07)**

**R. KANAGASABAPATHY (A19CADB16)**

**S. PRASATH (A19CADB29)**

under the guidance of

**MR. Z. JOHN BERNARD MCA., M.Phil.,**

Assistant Professor, PG Department of Computer Applications



**PG DEPARTMENT OF COMPUTER APPLICATIONS**

**ST. JOSEPH'S COLLEGE OF ARTS & SCIENCE (AUTONOMOUS)**

(Affiliated to Thiruvalluvar University, Vellore)

**CUDDALORE - 607001**

**JUNE - 2022**

# **CERTIFICATE**

This is to certify that the Mini Project entitled

## **JABBER**

being submitted to the St. Joseph's College of Arts & Science (Autonomous),

Affiliated to Thiruvalluvar University, Vellore.

by

**K. CHITRAARASU (A19CADB07)**

**R. KANAGASABAPATHY (A19CADB16)**

**S. PRASATH (A19CADB29)**

for the partial fulfillment for the award of degree of

## **BACHELOR OF COMPUTER APPLICATIONS**

is a bonafide record of work carried out by them,

under my guidance and supervision.

**INTERNAL GUIDE**

**HEAD OF THE DEPARTMENT**

Submitted for the viva-voce examination held on \_\_\_\_\_

**Examiners:**

1. \_\_\_\_\_

2. \_\_\_\_\_

# ACKNOWLEDGEMENT

It is our earnest and sincere desire and ambition to acquire profound knowledge in the study of Computer Applications. We are grateful to God, the almighty who has blessed us abundantly and guide us to complete this work.

It is our pleasure to acknowledge the help and guidance that we had received from different personal and to thank them individually.

We grab this opportunity to thank **Rev. Fr. G. PETER RAJENDIRAM M.A., M.Sc., M.Ed., M.Phil.**, Secretary, **Dr. M. ARUMAI SELVAM M.Sc., M.Phil., Ph.D.**, Principal, St. Joseph's College of Arts & Science (Autonomous), Cuddalore for giving us an opportunity for submitting this project.

We take immense pleasure in conveying our sincere and heartfelt gratitude to **Dr. A. JOHN PRADEEP EBENEZER M.C.A., M.Phil., Ph.D.**, Head, PG Department of Computer Applications for extending the laboratory facilities and moral support.

We are greatly indented and truly feel privileged to extend our thanks to our guide **MR. Z. JOHN BERNARD MCA., M.Phil.**, Assistant Professor, PG Department of Computer Applications for the able guidance and constant encouragement during the development of our project work.

We express our sincere thanks to our Parents for constantly giving us words of courage and motivation to complete this project successfully. We would like to thank Our Friends for their guidance and assistance in the completion of the project.

**K. CHITRAARASU (A19CADB07)**  
**R. KANAGASABAPATHY (A19CADB16)**  
**S. PRASATH (A19CADB29)**

# TABLE OF CONTENTS

<b>CHAPTER NO</b>	<b>NAME OF THE TOPIC</b>	<b>PAGE NO</b>
<b>1</b>	<b>ABSTRACT</b>	<b>1</b>
<b>2</b>	<b>SYSTEM STUDY</b>  2.1 HARDWARE SPECIFICATION  2.2 SOFTWARE SPECIFICATION	<b>2</b>
<b>3</b>	<b>SOFTWARE DESCRIPTION</b>  3.1 FRONT END- FLUTTER  3.2 BACK END - FIREBASE	<b>3</b>
<b>4</b>	<b>SYSTEM ANALYSIS</b>  4.1 EXISTING SYSTEM  4.2 PROPOSED SYSTEM	<b>9</b>
<b>5</b>	<b>PROJECT DESCRIPTION</b>  5.1 PROBLEM DEFINITION  5.2 OVERVIEW OF THE PROJECT  5.3 MODULE DESCRIPTION  5.4 DATABASE DESIGN	<b>11</b>

<b>CHAPTER NO</b>	<b>NAME OF THE TOPIC</b>	<b>PAGE NO</b>
<b>6</b>	<b>SYSTEM TESTING</b>  6.1 UNIT TESTING	15
<b>7</b>	<b>SYSTEM IMPLEMENTATION</b>	17
<b>8</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	18
	<b>APPENDICES</b>  A. CODING  B. SCREENSHOT	19
	<b>BIBILIOGRAPHY AND REFERENCE</b>	32

# **CHAPTER - 1**

## **ABSTRACT**

Jabber is a chatting app which is created by using flutter. Chat refers to the process of communicating, interacting and exchanging message over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audio or video communication via the Internet.

This app runs with any platform such as Linux, windows etc. It runs on the users android device and server applications. which runs on any android device on the network .

A chat application has basic two components, server and client. A server is a computer program or a device that provides functionality for other programs or devices here firebase act as the server. Clients who want to chat with each other connect to the server.

The chat application we are going to make will be more like a chat room, rather than a peer to peer chat .So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user.

## **CHAPTER - 2**

### **SYSTEM STUDY**

#### **2.1 HARDWARE REQUIREMENTS**

- SYSTEM : INTEL(R) CORE(TM) 1.60GHZ
- RAM : 8 GB
- INTERNAL STORAGE : 256 GB .

#### **2.2 SOFTWARE REQUIREMENTS**

- OPERATING SYSTEM : Android 10
- SOFTWARE: : ANDROID STUDIO
- FRAMEWORK : FLUTTER
- PROGRAMMING LANGUAGE : DART
- BACK END : FIREBASE

## **CHAPTER - 3**

### **SOFTWARE DESCRIPTION**

#### **3.1 FRONT END**

##### **3.1.1 FLUTTER**

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform application for Android, IOS, Linux, macOS, windows and the web from a single codebase

The first version of Flutter was known as "Sky" and ran on the Android operating system. It was unveiled at the 2015 Dart developer summit with the stated intent of being able to render consistently at 120 frames per second. During the keynote of Google Developer Days in Shanghai in September 2018, Google announced Flutter Release Preview 2, the last major release before Flutter 1.0. On December 4th of that year, Flutter 1.0 was released at the Flutter Live event, denoting the first stable version of the framework. On December 11, 2019, Flutter 1.12 was released at the Flutter Interactive event.

On May 6, 2020, the Dart software development kit (SDK) version 2.8 and Flutter 1.17.0 were released, adding support for the Metal API which improves performance on iOS devices by approximately 50%, as well as new Material widgets and network tracking development tools.

On March 3, 2021, Google released Flutter 2 during an online Flutter Engage event. This major update brought official support for web-based applications with a new Canvas Kit renderer and web specific widgets, early-access desktop application support for Windows, macOS, and Linux and improved Add-to-App APIs. This release also utilized Dart 2.0 that featured sound null-safety, which caused many breaking changes and issues with many external packages; however, the Flutter team included instructions and tools to mitigate these issues.



Flutter is a simple and high-performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native frame work.

Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML.

To be specific, Flutter application is itself a widget. Flutter widgets also supports animations and gestures. The application logic is based on reactive programming. Widget may optionally have a state. By changing the state of the widget, Flutter will automatically (reactive programming) compare the widget's state (old and new) and render the widget with only the necessary changes instead of re-rendering the whole widget.

## **Features of Flutter**

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms.
- High performance application

## **DART**

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks.

Languages are defined by their technical envelope-the choices made during development that shape the capabilities and strengths of a language. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hot reload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and desktop).

Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.

## **Dart: The language**

The Dart language is type safe; it uses static type checking to ensure that a variable's value always matches the variable's static type. Sometimes, this is referred to as sound typing. Although types are mandatory, type annotations are optional because of type inference. The Dart typing system is also flexible, allowing the use of a dynamic type combined with runtime checks, which can be useful during experimentation or for code that needs to be especially dynamic.

Dart offers sound null safety, meaning that values can't be null unless you say they can be. With sound null safety, Dart can protect you from null exceptions at runtime through static code analysis. Unlike many other null-safe languages, when Dart determines that a variable is non-nullable, that variable is always non-nullable. If you inspect your running code in the debugger, you'll see that non-nullability is retained at runtime (hence sound null safety).

## **Dart: The libraries**

Dart has a rich set of core libraries, providing essentials for many everyday programming tasks:

- Built-in types, collections, and other core functionality for every Dart program (dart:core)
- Richer collection types such as queues, linked lists, hashmaps, and binary trees (dart:collection)
- Encoders and decoders for converting between different data representations, including JSON and UTF-8 (dart:convert)
- Mathematical constants and functions, and random number generation (dart:math)
- File, socket, HTTP, and other I/O support for non-web applications (dart:io)

- Support for asynchronous programming, with classes such as `Future` and `Stream` (`dart:async`)
- Lists that efficiently handle fixed-sized data (for example, unsigned 8-byte integers) and SIMD numeric types (`dart:typed_data`)
- Foreign function interfaces for interoperability with other code that presents a C-style interface (`dart:ffi`)
- Concurrent programming using isolates—independent workers that are similar to threads but don't share memory, communicating only through messages (`dart:isolate`)
- HTML elements and other resources for web-based applications that need to interact with the browser and the Document Object Model (DOM) (`dart:html`)

## Dart: The platforms

Dart's compiler technology lets you run code in different ways:

- **Native platform:** For apps targeting mobile and desktop devices, Dart includes both a Dart VM with just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler for producing machine code.
- **Web platform:** For apps targeting the web, Dart includes both a development time compiler (`dartdevc`) and a production time compiler (`dart2js`). Both compilers translate Dart into JavaScript.

The Flutter framework is a popular, multi-platform UI toolkit that's powered by the Dart platform, and that provides tooling and UI libraries to build UI experiences that run on iOS, Android, macOS, Windows, Linux, and the web.

Dart Native (machine code JIT and AOT)

During development, a fast developer cycle is critical for iteration. The Dart VM offers a just-in-time compiler (JIT) with incremental recompilation (enabling hot reload), live metrics collections (powering DevTools), and rich debugging support.

When apps are ready to be deployed to production—whether you're publishing to an app store or deploying to a production backend—the Dart AOT compiler enables ahead-of-time compilation to native ARM or x64 machine code. Your AOT-compiled app launches with consistent, short startup time.

The AOT-compiled code runs inside an efficient Dart runtime that enforces the sound Dart type system and manages memory using fast object allocation and a generational garbage collector.

### **3.1.2 FIREBASE**

Cloud Functions for Firebase is a serverless framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your JavaScript or TypeScript code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.

Firebase developers can easily integrate with external services for tasks like processing payments and sending SMS messages. Also, developers can include custom logic that is either too heavyweight for a mobile device, or which needs to be secured on a server.

Cloud Functions for Firebase is optimized for Firebase developers:

- Firebase SDK to configure your functions through code
- Integrated with Firebase Console and Firebase CLI
- The same triggers as Google Cloud Functions, plus Firebase Realtime Database, Firebase Authentication, and Firebase Analytics triggers

Build powerful apps. Spin up your backend without managing servers. Effortlessly scale to support millions of users with Firebase databases, machine learning infrastructure, hosting and storage solutions, and Cloud Functions.

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

For developers that need a more full-featured backend, Cloud Functions provides a gateway to the powerful capabilities in Google Cloud Platform.

## **Flexibility**

The Cloud Firestore data model supports flexible, hierarchical data structures. Store your data in documents, organized into collections. Documents can contain complex nested objects in addition to subcollections.

## **Expressive querying**

In Cloud Firestore, you can use queries to retrieve individual, specific documents or to retrieve all the documents in a collection that match your query parameters. Your queries can include multiple, chained filters and combine filtering and sorting. They're also indexed by default, so query performance is proportional to the size of your result set, not your data set.

## **Realtime updates**

Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently.

## **Offline support**

Cloud Firestore caches data that your app is actively using, so the app can write, read, listen to, and query data even if the device is offline. When the device comes back online, Cloud Firestore synchronizes any local changes back to Cloud Firestore.

## **Designed to scale**

Cloud Firestore brings you the best of Google Cloud's powerful infrastructure: automatic multi-region data replication, strong consistency guarantees, atomic batch operations, and real transaction support. We've designed Cloud Firestore to handle the toughest database workloads from the world's biggest apps.

## **CHAPTER - 4**

### **SYSTEM ANALYSIS**

#### **4.1 EXISTING SYSTEM**

In chatting applications we have many features. Keep in touch with the groups of people that matter the most, like your family WhatsApp on the web and desktop, you can seamlessly sync all of your chats to your computer so that you can chat on whatever device is most convenient for you. WhatsApp voice and video calls use your phone's Internet connection, instead of your cell plan's voice minutes, so you don't have to worry about expensive calling charges. Send PDFs, documents, spreadsheets, slideshows and more, without the hassle of email or file sharing apps. Sometimes, your voice says it all. With just one tap you can record a Voice Message, perfect for a quick hello or a longer story. Some of your most personal moments are shared on WhatsApp, which is why we built end-to-end encryption into the latest versions of our app

#### **4.2 PROPOSED SYSTEM**

The proposed system is the computerized version of the existing system. We can introduce new features is chatbot, scheduling a message, location tracking. A chatbot or chatterbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. A chatbot is a type of software that can help customers by automating conversations and interact with them through messaging platforms. Chatbots are used in dialog systems for various purposes including customer service, request routing, or information gathering.

#### **4.3 FEASIBILITY STUDY**

A feasibility study is a preliminary study undertaken to determine and document a project's viability. The results of this project are used to make a decision whether to proceed with the project, or table it. If it indeed leads to a project being approved, the proposed project work starts.

##### **4.3.1 ECONOMICAL & FINANCIAL FEASIBILITY**

This involves questions whether the project started organization such as benefits should substantially exceed its costs, and whether the project has higher priority and profits than other projects that might use the same resources.

### **4.3.2 TECHNICAL FEASIBILITY**

This has such as the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. This can be qualified in terms of size of modules, coding, etc.

### **4.3.3 OPERATIONAL FEASIBILITY**

Operational feasibility is a test of feasibility that will check whether the system is working when it is developed and installed in place of the existing system. The Proposed system is beneficial only if it can be turned into information system that will meet the organization's operational requirements.

# CHAPTER - 5

## PROJECT DESCRIPTION

### 5.1 PROBLEM DEFINITION

Email, newsgroup and messaging applications provide means for communication among people but these are one-way mechanisms and they do not provide an easy way to carry on a real-time conversation or discussion with people involved. Chat room extends the one-way messaging concept to accommodate multi-way communication among a set of people.

### 5.2 OVERVIEW OF THE PROJECT

The **chatting application** is developed using firebase. This **chatting application** helps to users chat with anyone using this app.

### 5.3 MODULE

The project “JABBER” deals with the following modules.

- **Onboarding page**
  - This page show case the features of the app.
- **Login page**
  - This page is used to get the phone number form the user.
- **Message page**
  - This is the page where the user chat with their friends.
- **Create channel**
  - This page where user can create group for chatting.
- **Join channel**
  - This page is used to join a channel using QR-code or by using channel id.
- **Schedule message**
  - User can able to schedule a message for a certain time.
- **Profile page**
  - This page is used to showcase the user details like profile image, name etc.
- **Music page**

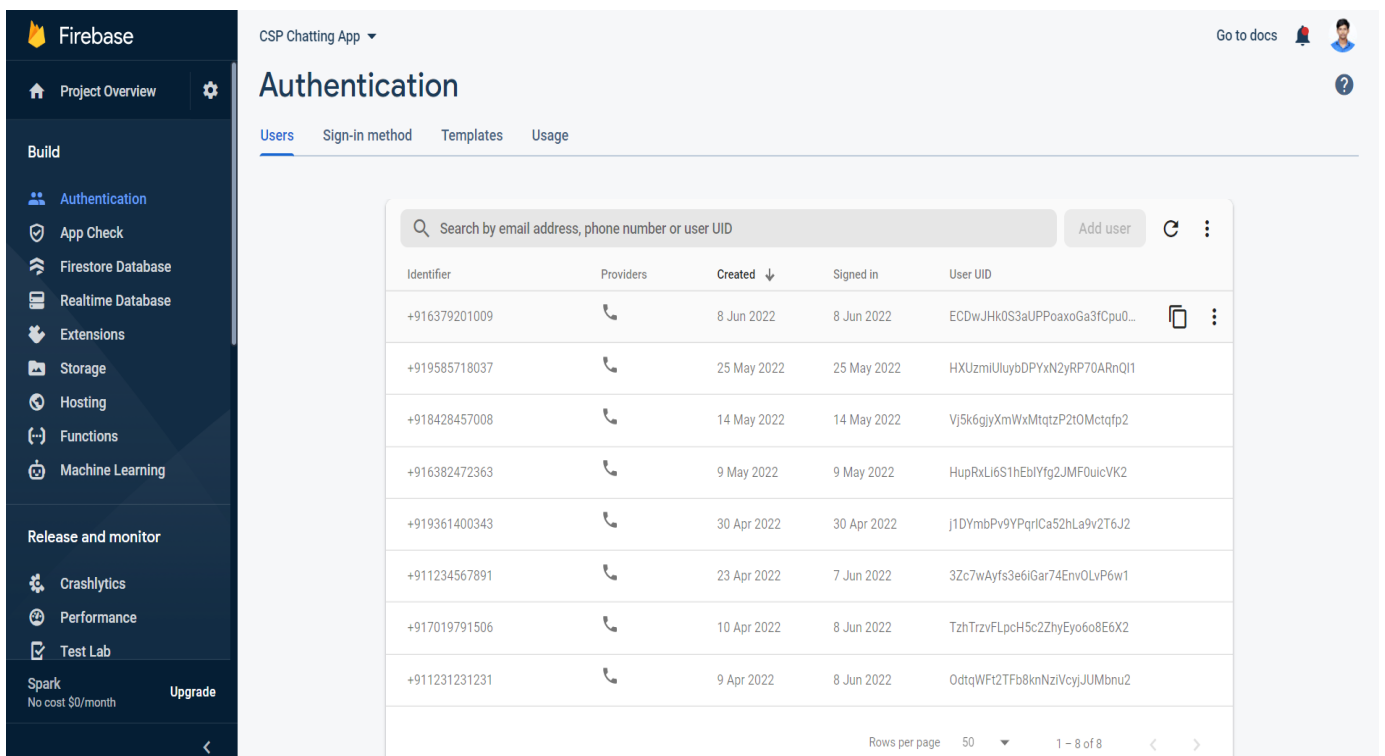


- It is an upcoming feature Which helps user to listen music.
- **News page**
  - It is an upcoming feature Which helps users to keep update them self with the latest news.
- **Settings page**
  - User can customize the app depends their needs
- **Chat Bot**
  - This page is uses to clear doubts and problems by chat with a bot when you are facing any issues. You can share your feeling without any fear.
- **Channel details page**
  - This page showcase the details of the channel.
- **Edit user profile page.**
  - User can change there profile and name using this page.
- **Contribution**
  - This page elaborate the details of the developers.

## DATA BASE

The database design plays a major role in the buildings of a project and also the table design is one of the important phases of a project

### 1. AUTHENTICATION



The screenshot shows the Firebase Authentication console for a project named "CSP Chatting App". The left sidebar contains navigation options: Project Overview, Build (Authentication, App Check, Firestore Database, Realtime Database, Extensions, Storage, Hosting, Functions, Machine Learning), Release and monitor (Crashlytics, Performance, Test Lab), and Spark (No cost \$0/month, Upgrade). The main content area is titled "Authentication" and has tabs for Users, Sign-in method, Templates, and Usage. The "Users" tab is active, displaying a table of users with columns: Identifier, Providers, Created, Signed in, and User UID. There are 8 users listed, each with a phone number as the identifier and a unique User UID. At the bottom, it shows "Rows per page: 50" and "1 - 8 of 8".

Identifier	Providers	Created	Signed in	User UID
+916379201009	Phone	8 Jun 2022	8 Jun 2022	ECDwJHk0S3aUPPoaxoGa3fCpu0...
+919585718037	Phone	25 May 2022	25 May 2022	HXUzmiUluYbDPYxN2yRP70ARnQ1
+918428457008	Phone	14 May 2022	14 May 2022	Vj5k6gijXmWxMtqtzP2tOMctqfp2
+916382472363	Phone	9 May 2022	9 May 2022	HupRxLi6S1hEbiYfg2JMF0ulcVK2
+919361400343	Phone	30 Apr 2022	30 Apr 2022	j1DYmbPv9YPqriCa52hLa9v2T6J2
+911234567891	Phone	23 Apr 2022	7 Jun 2022	3Zc7wAys3e6iGar74Env0LvP6w1
+917019791506	Phone	10 Apr 2022	8 Jun 2022	TzhTrzvFLpcH5c2ZhyEyo6o8E6X2
+911231231231	Phone	9 Apr 2022	8 Jun 2022	OdtqWfT2TFb8knNziVcyjJUMbnu2

## 2. MESSAGES

The screenshot shows the Firebase console interface for the 'CSP Chatting App' project. The left sidebar contains navigation options: Project Overview, Build (Authentication, App Check, Firestore Database, Realtime Database, Extensions, Storage, Hosting, Functions, Machine Learning), and Release and monitor (Crashlytics, Performance, Test Lab). The main content area displays the 'messages' collection in the 'csp-chatting-app' project. The '40pMZBKwDHUrgjUNW88' document is selected, showing the following fields:

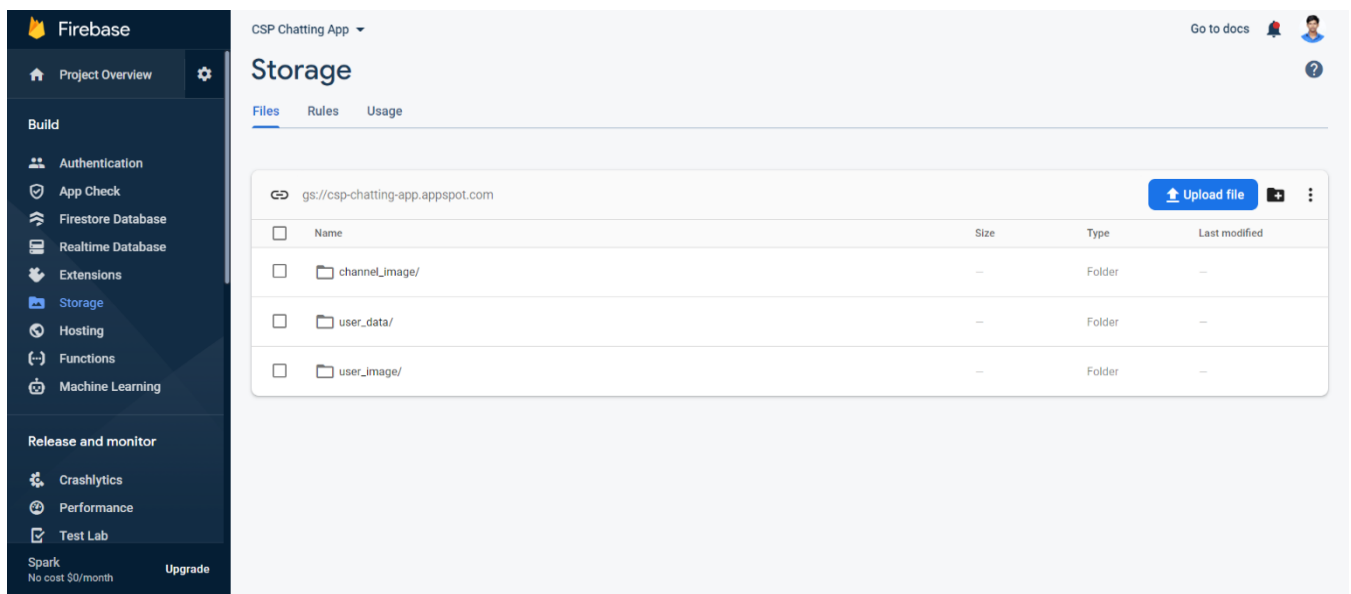
- channelId: "40pMZBKwDHUrgjUNW88"
- channelName: "Bca"
- channelOwner: "OdtqWFt2TFb8knNziVcyjJUMbnu2"
- channelProfile: null
- recentMessage: "Gi"
- time: 4 June 2022 at 10:39:03 UTC+5:30

## 3. USER DATA

The screenshot shows the Firebase console interface for the 'CSP Chatting App' project. The left sidebar contains navigation options: Project Overview, Build (Authentication, App Check, Firestore Database, Realtime Database, Extensions, Storage, Hosting, Functions, Machine Learning), and Release and monitor (Crashlytics, Performance, Test Lab). The main content area displays the 'users' collection in the 'csp-chatting-app' project. The 'Vj5k6gjjXmWxMtqtzP2tOMctqfp2' document is selected, showing the following fields:

- isOnline: true
- phoneNumber: "+918428457008"
- profileUrl: null
- uid: "Vj5k6gjjXmWxMtqtzP2tOMctqfp2"
- username: "Hi"

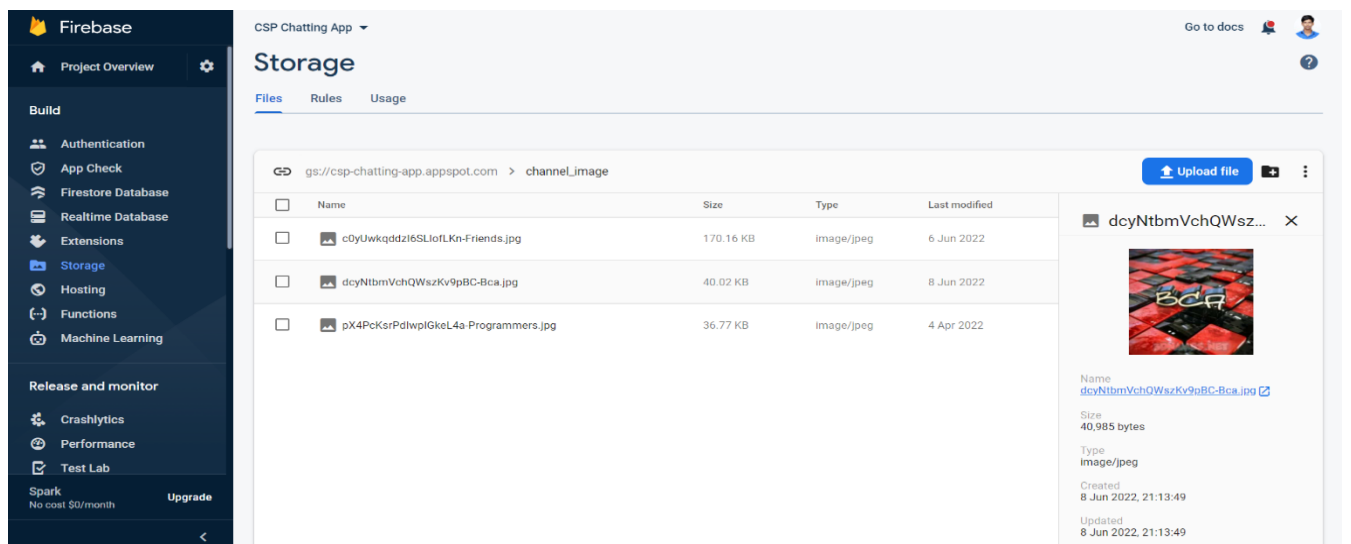
## 4. USER IMAGES



Storage console view for the CSP Chatting App. The left sidebar shows the project overview and build tools. The main area displays the Storage console with a table of folders:

Name	Size	Type	Last modified
channel_image/	—	Folder	—
user_data/	—	Folder	—
user_image/	—	Folder	—

## 5. USER CHANNEL IMAGES



Storage console view for the CSP Chatting App, showing the channel\_image directory. The left sidebar shows the project overview and build tools. The main area displays the Storage console with a table of files:

Name	Size	Type	Last modified
c0yUwkqddzleSLiofLKn-Friends.jpg	170.16 KB	image/jpeg	6 Jun 2022
dcyNtbmVchQWszKy9pBC-8ca.jpg	40.02 KB	image/jpeg	8 Jun 2022
pX4PcksrPdlwplGkeL4a-Programmers.jpg	36.77 KB	image/jpeg	4 Apr 2022

A file preview for dcyNtbmVchQWszKy9pBC-8ca.jpg is shown on the right, displaying the image and its metadata:

- Name: dcyNtbmVchQWszKy9pBC-8ca.jpg
- Size: 40,985 bytes
- Type: image/jpeg
- Created: 8 Jun 2022, 21:13:49
- Updated: 8 Jun 2022, 21:13:49

## **CHAPTER - 6**

### **SYSTEM TESTING**

Software testing is an important element of Software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and the costs associated with Software failure are motivating forces for well planned, through testing.

Though the test phase is often thought of as separate and distinct from the development effort--first develop, and then test--testing is a concurrent process that provides valuable information for the development team.

There are at least three options for integrating Project Builder into the test phase:

- ◆ Testers do not install Project Builder, use Project Builder functionality to compile and source-control the modules to be tested and hand them off to the testers, whose process remains unchanged.
- ◆ The testers import the same project or projects that the developers use.
- ◆ Create a project based on the development project but customized for the testers (for example, it does not include support documents, specs, or source), who import it.

### **TESTING OBJECTIVES**

There are several rules that can serve as testing objectives. They are

- ◆ Testing is a process of executing a program with the intent of finding an error.
- ◆ A good test case is one that has a high probability of finding an undiscovered error.
- ◆ A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software.

### **UNIT TESTING**

This is the first level of testing. In this different modules are tested against the specifications produced during the design of the module. During this testing the number of arguments is compared to input parameters, matching of parameter and arguments etc. It is also ensured whether the file attributes are correct, whether the Files are opened before using, whether Input/output errors are handled etc. Unit Test is conducted using a Test Driver usually.

Before We performed the unit testing, We are satisfied about 80% after that We have satisfied with our project of 100%.

## **VALIDATION TESTING**

This provides the final assurance that the software meets all functional, behavioral and performance requirements. The software is completely assembled as a package. Validation succeeds when the software functions in which the user expects.

---

## **CHAPTER - 7**

### **SYSTEM IMPLEMENTATION**

The project is developed by using FLUTTER frame work as the frontend and FIREBASE as the backend. Once the system has been designed, the next step is to convert the designed one into actual code, so as to satisfy the user requirements as expected. If the system is approved to be error free it can be implemented.

When the initial design was done for the system, the department was consulted for acceptance of the design so that further processing of the system development can be carried on. After the development of the system, a demonstration was given to them about working of the system. The aim of the system illustration was to identify any malfunctioning of the system.

Implementation includes proper training to end users. The implemented software should be maintained for prolonged running of the software.

Initially the system was run parallel with manual system. The system has been tested with data and has proved to error-free and user-friendly. Training was given to end user about the software and its features. Thus the project is running without any errors and warnings.

## **CHAPTER - 8**

### **CONCLUSION & FUTURE ENHANCEMENT**

#### **8.1 CONCLUSION**

The **CHATTING APPLICATION** is developed using FLUTTER and DART as frontend and backend is FIREBASE. All the modules are working properly.

The project has been completed successfully with the maximum satisfaction of the organization. The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. The project gives good idea on developing a full-fledged application satisfying the user requirements. The system is very flexible and versatile.

This software has a user-friendly screen that enables the user to use without any inconvenience. Validation checks induced have greatly reduced errors. Provisions have been made to upgrade the software.

#### **8.2 FUTURE ENHANCEMENT**

- In future, latest songs will be uploaded automatically.
- In future, latest songs will be updated automatically.
- In future, user can dedicate songs to their friends.
- In the future, users can check whether the news is correct or not.

## APPENDIX – A

### SOURCE CODE

#### pubspec.yaml

```
name: chatting_application
description: A new Flutter project.
publish_to: 'none'
version: 1.0.0+1
environment:
  sdk: ">=2.14.4 <3.0.0"
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
  introduction_screen: ^3.0.0
  get: ^4.6.1
  lottie: ^1.2.2
  country_pickers: ^2.0.0
  rive: ^0.8.1
  pin_code_fields: ^7.3.0
  image_picker: ^0.8.4+10
  firebase_core: ^1.13.1
  firebase_auth: ^3.3.10
  cloud_firestore: ^3.1.10
  firebase_storage: ^10.2.9
  grouped_list: ^4.2.0
  intl: ^0.17.0
  qr_flutter: ^4.0.0
  qr_code_scanner: ^0.7.0
  encrypt: ^5.0.1
  file_picker: ^4.5.1
  google_maps_flutter: ^2.1.5
  location: ^4.3.0
  fluttertoast: ^8.0.9
  http: ^0.13.4
  url_launcher: ^6.1.2
  emoji_picker_flutter: ^1.1.2
  workmanager: ^0.4.1
  flutter_local_notifications: ^9.5.3+1
```



```
shared_preferences: ^2.0.15
connectivity_plus: ^2.3.0
animations: ^2.0.2
flutter_spinkit: ^5.1.0
sound_stream: ^0.3.0
dialogflow_grpc: ^0.2.2
rxdart: ^0.27.4
```

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^1.0.0
  flutter_native_splash: ^2.1.2+1
  flutter_launcher_icons: ^0.9.2
```

```
flutter_icons:
  android: true
  ios: true
  image_path: "assets/app-icon.png"
```

```
flutter_native_splash:
  color: "#ffffff"
  color_dark: "#ffffff"
  android: true
  ios: true
  image: "assets/splash.png"
  uses-material-design: true
  assets:
    - assets/animations/
    - assets/images/
    - assets/credentials.json
```

## **Main.dart**

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:connectivity_plus/connectivity_plus.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:workmanager/workmanager.dart';
```

```

import 'screens/home.dart';
import 'screens/onboarding_page.dart';

void callbackDispatcher() {
  Workmanager().executeTask((task, inputData) async {
    var connectivityResult = await (Connectivity().checkConnectivity());
    if (connectivityResult == ConnectivityResult.none) {
      return Future.value(false);
    } else {
      await Firebase.initializeApp();
      final userData = await FirebaseFirestore.instance
        .collection('users')
        .doc(inputData!['currentUserId'])
        .get();
      for (var a = 0; a < inputData['messages'].length; a++) {
        FirebaseFirestore.instance
          .collection('messages')
          .doc(inputData['cid'])
          .collection("channelChat")
          .add({
            'message': inputData['messages'][a],
            'messageType': inputData['type'][a],
            'createdTime': Timestamp.now(),
            'senderId': inputData['currentUserId'],
            'senderName': userData['username'],
          });
        FirebaseFirestore.instance
          .collection('messages')
          .doc(inputData['cid'])
          .update({
            'recentMessage': inputData['messages'][a],
            'time': Timestamp.now(),
          });
        FirebaseFirestore.instance
          .collection('users')
          .doc(inputData['currentUserId'])
          .collection("userChannels")
          .doc(inputData['cid'])
          .update({
            'recentMessage': inputData['messages'][a],
            'time': Timestamp.now(),
          });
      }
    }
  });
}

```

```

    });
  }
  return Future.value(true);
}
});
}

```

```

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  Workmanager().initialize(callbackDispatcher, isInDebugMode: true);
  runApp(MyApp());
}

```

```

class MyApp extends StatelessWidget {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      title: "Jabber",
      theme: ThemeData(primarySwatch: Colors.blue),
      home: _auth.currentUser == null ? const OnBoardingPage() : const Home(),
    );
  }
}

```

## **Home.dart**

```

import 'package:animations/animations.dart';
import
'package:chatting_application/screens/create_new_channel_or_join_channel.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

import '../controller/controller.dart';
import '../widget/customMaterialButton.dart';
import 'onboarding_page.dart';

```

```

class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  var data = Get.put(Controller());
  final _fabDimension = 56.0;
  @override
  void initState() {
    Future.delayed(Duration.zero, () async {
      var preProfileData;
      await FirebaseFirestore.instance
        .collection('users')
        .doc(FirebaseAuth.instance.currentUser?.uid)
        .get()
        .then((value) {
          preProfileData = value.data();
        });
      if (preProfileData == null) {
        FirebaseAuth.instance.signOut();
        Get.offAll(const OnBoardingPage(), transition: Transition.fade);
      }
    });
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFFfcfcfc),
      // body: GetBuilder<Controller>(
      //   builder: (controller) => controller.body,
      // ),
      body: GetBuilder<Controller>(
        builder: (controller) => PageTransitionSwitcher(
          transitionBuilder: (
            Widget child,
            Animation<double> animation,

```

```

    Animation<double> secondaryAnimation,
  ) {
    return FadeThroughTransition(
      animation: animation,
      secondaryAnimation: secondaryAnimation,
      child: child,
    );
  },
  child: controller.body,
),
),
bottomNavigationBar: bottomNavigationBar,
floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
floatingActionButton: Visibility(
  visible: MediaQuery.of(context).viewInsets.bottom == 0.0,
  child: OpenContainer(
    transitionType: ContainerTransitionType.fade,
    openBuilder: (BuildContext context, VoidCallback _) {
      return const CreateNewChannelOrJoinChannel();
    },
    closedElevation: 6.0,
    closedShape: RoundedRectangleBorder(
      borderRadius: BorderRadius.all(
        Radius.circular(_fabDimension / 2),
      ),
    ),
    closedColor: Theme.of(context).colorScheme.secondary,
    closedBuilder: (BuildContext context, VoidCallback openContainer) {
      return Container(
        color: const Color(0xFF006aff),
        child: SizedBox(
          height: _fabDimension,
          width: _fabDimension,
          child: Center(
            child: Icon(
              Icons.add,
              color: Theme.of(context).colorScheme.onSecondary,
            ),
          ),
        ),
      );
    },
  );

```

```

    },
  ),
),
);
}

```

```

Widget get bottomNavigationBar {
  var mediaData = MediaQuery.of(context).size;
  return GetBuilder<Controller>(
    builder: (controller) => BottomAppBar(
      clipBehavior: Clip.hardEdge,
      shape: const CircularNotchedRectangle(),
      notchMargin: 7.5,
      child: SizedBox(
        height: 60,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            CustomMaterialButton(() {
              controller.setScreen(0);
            }, controller.index == 0, Icons.messenger, "Chats"),
            CustomMaterialButton(() {
              controller.setScreen(1);
            }, controller.index == 1, Icons.music_note_rounded, "Music"),
            SizedBox(
              width: mediaData.width * .05,
            ),
            CustomMaterialButton(() {
              controller.setScreen(2);
            }, controller.index == 2, Icons.newspaper_rounded, "News"),
            CustomMaterialButton(() {
              controller.setScreen(3);
            }, controller.index == 3, Icons.account_box_rounded, "Profile"),
          ],
        ),
      ),
    );
}
}

```

## Controller.dart

```
import 'dart:io';

import 'package:chatting_application/screens/music.dart';
import 'package:chatting_application/screens/chat_list.dart';
import
'package:chatting_application/screens/create_new_channel_or_join_channel.dart';
import 'package:chatting_application/screens/profile.dart';
import 'package:country_pickers/country.dart';
import 'package:country_pickers/country_pickers.dart';
import 'package:get/get.dart';

import '../screens/chat_bot.dart';
import '../screens/news.dart';

class Controller extends GetxController {
  var _index = 0;
  var isEmojiVisible = false;

  var _isLoading = false;

  get isLoading {
    return _isLoading;
  }

  setIsLoading(val) {
    _isLoading = val;
  }

  var _value = 0;

  get value {
    return _value;
  }

  setValue(val) {
    _value = val;
    update();
  }
}
```

```

final _screens = [
    ChatList(),
    Music(),
    News(),
    Profile(),
];

get body {
    return _screens[_index];
}

get index {
    return _index;
}

setScreen(index) {
    switch (index) {
        case 0:
            _index = index;
            break;
        case 1:
            _index = index;
            break;
        case 2:
            _index = index;
            break;
        case 3:
            _index = index;
            break;
    }
    update();
}

Country _selectedDialogCountry =
    CountryPickerUtils.getCountryByPhoneCode('91');

get selectedDialogCountry {
    return _selectedDialogCountry;
}

setCountry(country) {

```



```
    _selectedDialogCountry = country;
    update();
}
```

```
File? _storedImage;
File? _storedChannellImage;
```

```
get userProfileImage {
    return _storedImage;
}
```

```
get channelProfileImage {
    return _storedChannellImage;
}
```

```
setUserProfileImage(image) {
    _storedImage = image;
    update();
}
```

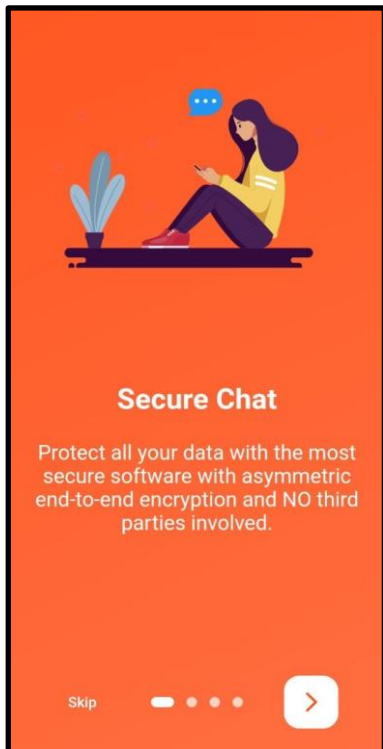
```
setChannelProfileImage(image) {
    _storedChannellImage = image;
    update();
}
}
```

**Rest of the code available in this website : -**

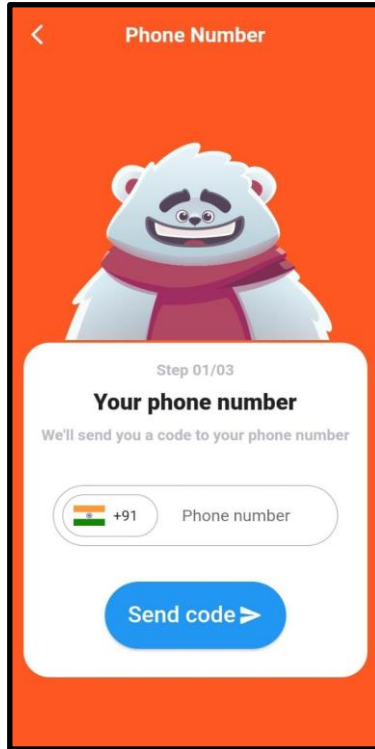
<https://github.com/chitraarasu/Jabber>

## APPENDIX – B

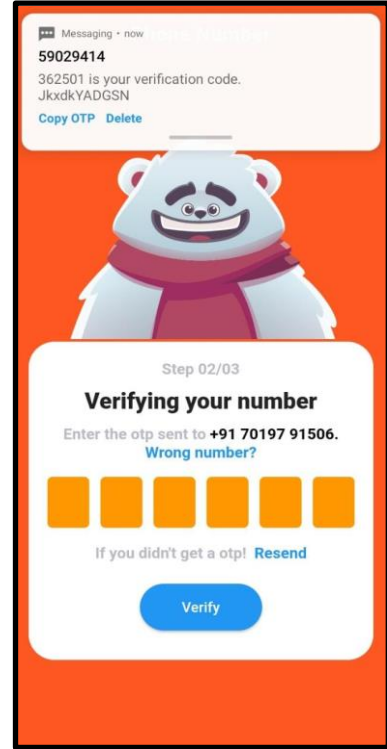
### OUTPUT



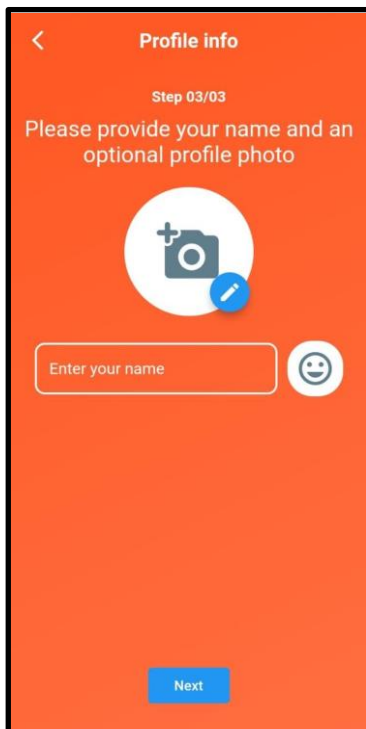
Onboarding



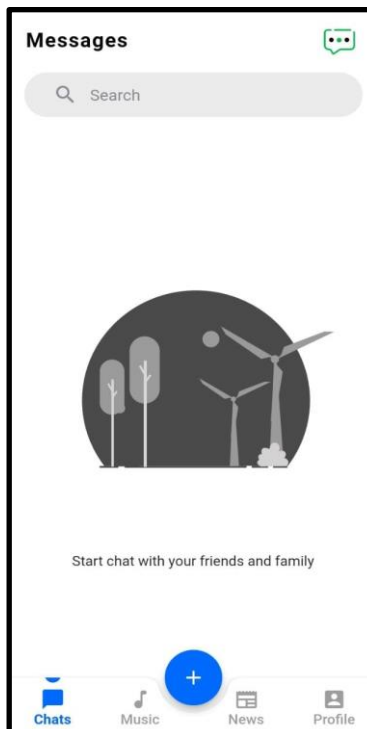
Phone number



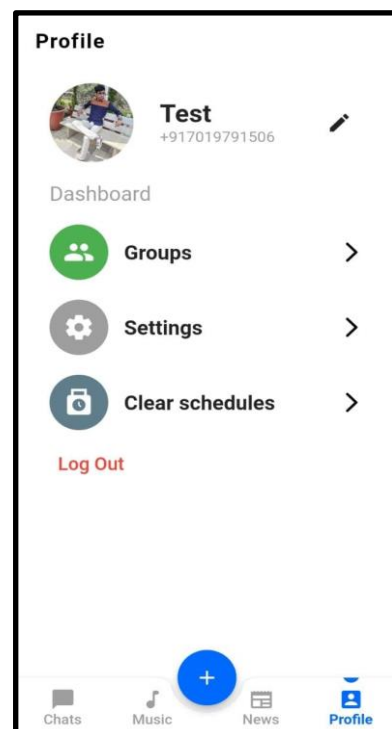
OTP page



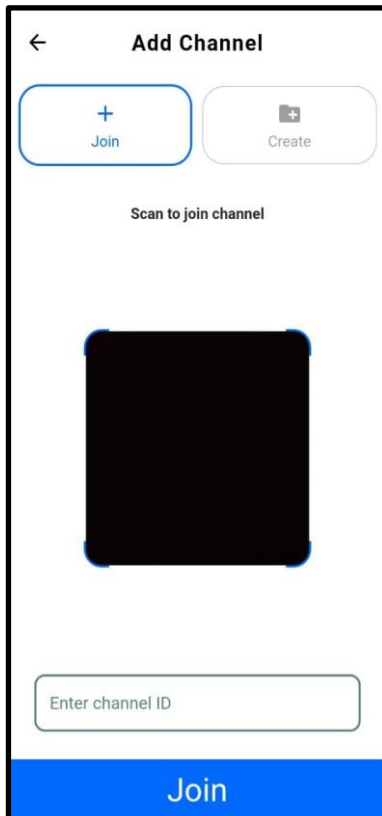
User profile



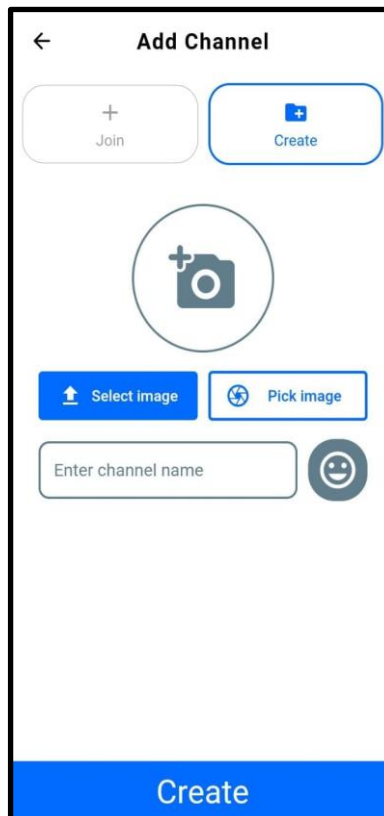
Home page



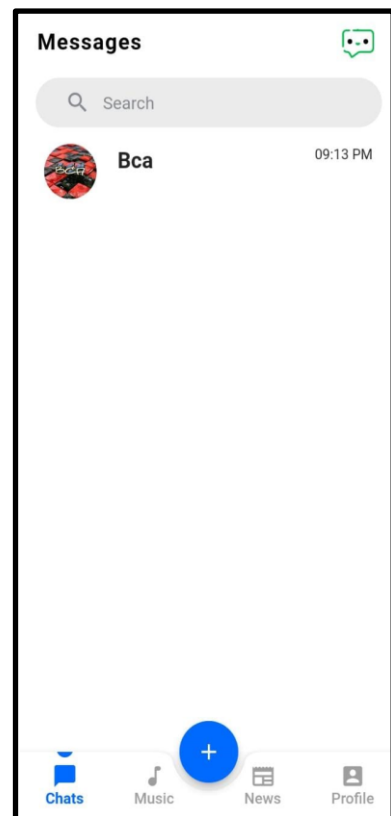
Profile page



Join channel page



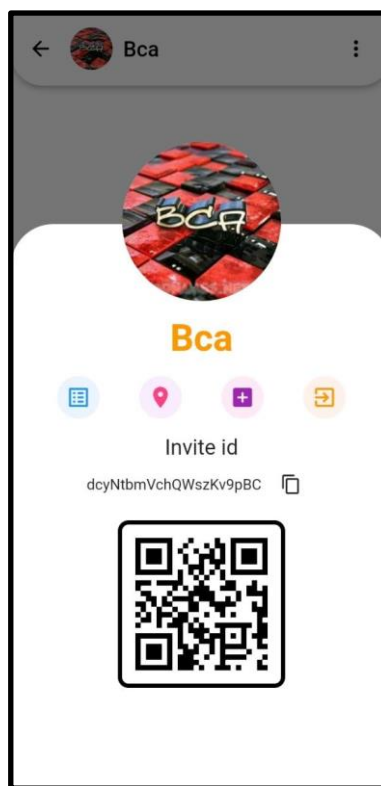
Create channel page



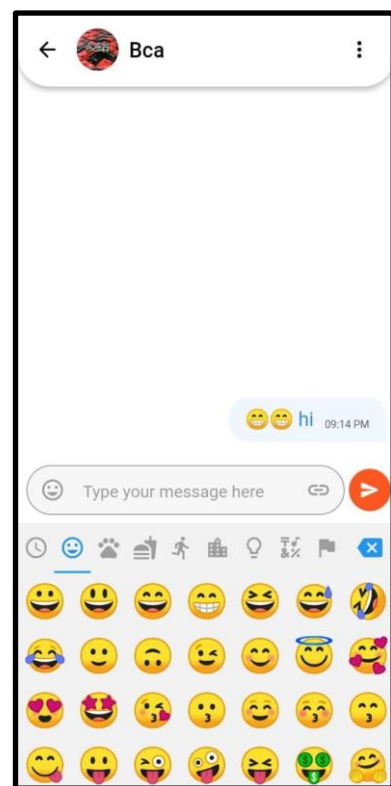
Home page



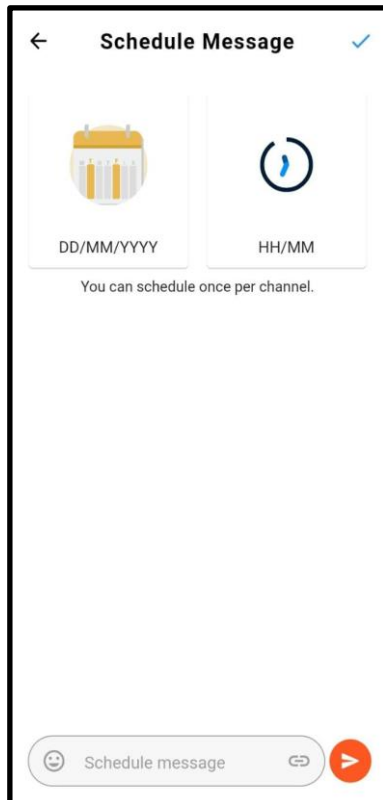
Chat screen



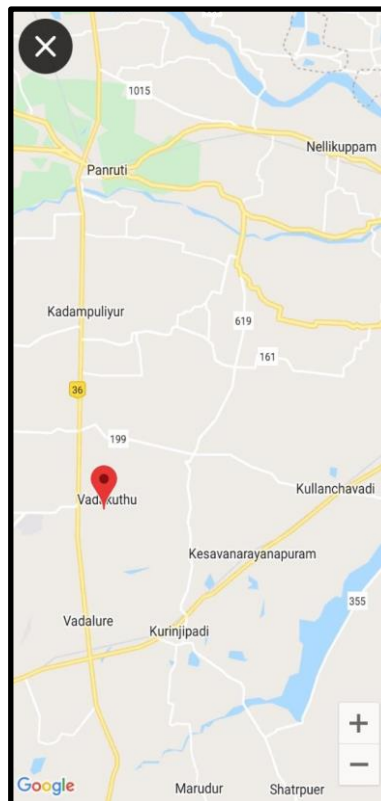
Channel profile



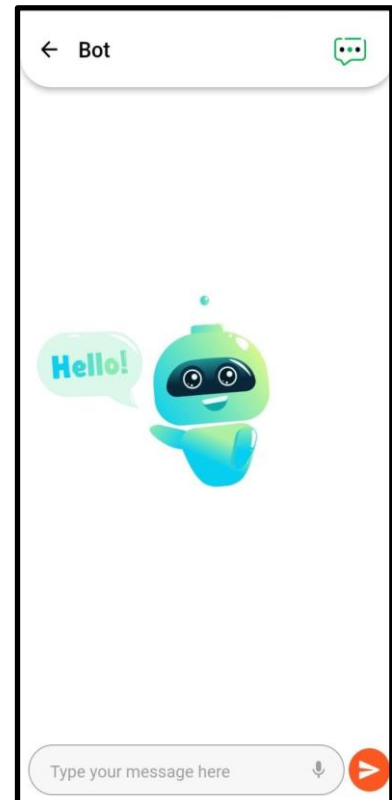
Chat screen



**Schedule message**



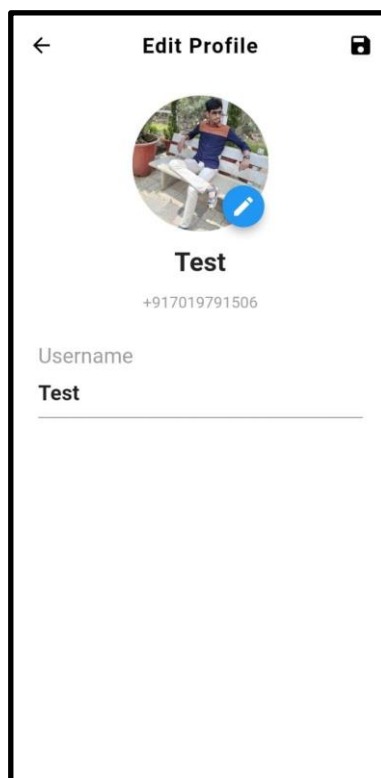
**Location tracker**



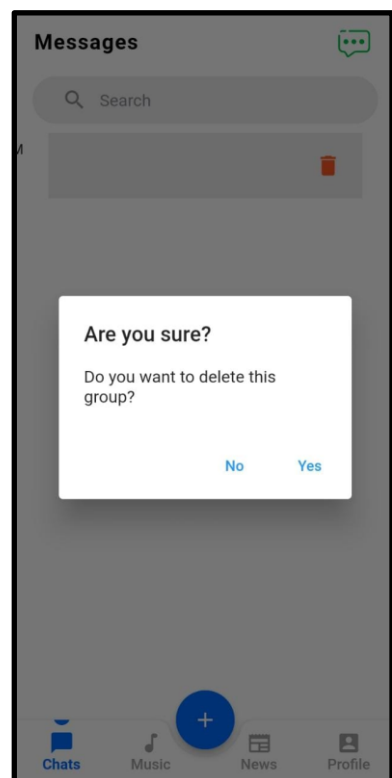
**Chat bot**



**Chatbot UI**



**Edit profile page**



**Leave channel**

# **BIBLIOGRAPHY AND REFERENCES**

## **BIBLIOGRAPHY**

1. Flutter UI succinctly, Ed Freitas.
2. Managing State in Flutter Pragmatically, Waleed Arshad.
3. Flutter Cookbook, Simone Alessandria.

## **REFERENCES**

1. <https://flutter.dev/>
2. <https://gallery.flutter.dev/>
3. <https://firebase.google.com/docs>
4. <https://dart.dev/>
5. <https://github.com/chitraarasu>
6. <https://github.com/kanagasabapathyR>
7. <https://github.com/prasathsaravanan>