

**PRIVACY FOCUSED MESSENGER USING FLUTTER**

**A PROJECT REPORT**

Submitted by

RAKESH B. [REG NO:211417104215]

LAKSHMAN KUMAR M. [REG NO:211417104128]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2021

**BONAFIDE CERTIFICATE**

Certified that this project report”PRIVACY FOCUSED MESSENGER USING FLUTTER” is the Bonafide work of “RAKESH.B(211417104215) AND LAKSHMAN KUMAR M(211417104128)” who carried out the project under my supervision.

SIGNATURE SIGNATURE

Dr. S. MURUGAVALLI, M.E.,Ph.D. ..., Mr. .SHANMUGANATHAN., M.E.,

HEAD OF THE DEPARTMENT SUPERVISOR

PROFESSOR ASSOCIATE PROFESSOR

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE , PANIMALARENGINEERING COLLEGE,

NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voice Examination held on...........................

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent Dr.P.CHINNADURAI, M.A., Ph.D., for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors Tmt.C.VIJAYARAJESWARI, Thiru.C.SAKTHIKUMAR, M.E., and Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal Dr.K.Mani, M.E., Ph.D., for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, Dr. S. MURUGAVALLI, M.E.,Ph.D.., for the support extended throughout the project.

We would like to thank my Project Guide Mr. M.SHANMUGANATHAN

and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

ABSTRACT

One of the most important pieces of technology in our generation is messaging. The drawback with modern messengers is that users need a phone number tied to their name and which has their personal information. our application targets users who cannot carry their personal phone with them at all time or if their battery died and they are in an emergency situation and need to contact their family and friends they cannot use their account since they need their number and one-time password. Our application tackles the problem by using a simple login method such as email and password so users can login to their accounts at any time and wherever they want.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| CHAPTER NO | TITLE | PAGE NO |
|  | ABSTRACT  LIST OF FIGURES |  |
| 1. | CHAPTER 1: INTRODUCTION  OVERVIEW | 9 |
| 2. | CHAPTER 2: LITERATURE SURVEY | 10 |
| 3. | CHAPTER 3: SYSTEM ANALYSIS  3.1 EXISTING SYSTEM  3.2 PROPOSED SYSTEM  3.3 SYSTEM CONFIGURATION  3.4.1 HARDWARE CONFIGURATION  3.4.2 SOFTWARE CONFIGURATION | 13  13  14  16  16  16 |
| 4. | CHAPTER 4: ARCHITECTURE  4.1 SYSTEM ARCHITECTURE  4.2 UML DIAGRAMS  4.2.1 USECASE Diagram  4.2.2 activity Diagram | 18  18  18  19  20 |
| 5. | CHAPTER 5: SYSTEM DESIGN | 22 |
| 6. | CHAPTER 6 : SOFTWARE TESTING  6.1 INTRODUCTION  6.2 TYPES OF TESTS  6.2.1 UNIT TESTING  6.2.2 INTEGRATION TESTING  6.2.3 FUNCTIONAL TEST  6.2.4 SYSTEM TESTING  6.2.5 INTEGRATION TESTING  6.2.6 ACCEPTANCE TESTING | 30  30  31  31  32  32  34  36  36 |
| 7. | CHAPTER 7: CONCLUSION  7.1 APPLICATIONS  7.2 FUTURE ENHANCEMENTS  7.3 CONCLUSION  SCREENSHOTS  **REFERENCES** | 37  37  38  39  43 |

**CHAPTER 1**

**INTRODUCTION**

Messaging and texting are among the most popular methods of communication among children and teenagers. Messaging and texting can be much more than ways to communicate. They can also be tools that help young people learn and master important skills.

Online messaging apps are often used by young people to talk to their friend and peers. Staying connected with your friends is valuable, but online messaging also gives young people opportunities to develop their social skills in a range of formal and informal contexts.

Online messaging can help young people to:

* appreciate different perspectives
* understand the difference between appropriate and inappropriate behavior
* become more effective non-verbal communicators.

Privacy focused Messenger does not collect any. Adversaries cannot learn who is conversing with whom, or if anyone is conversing at all. Relationship unobservability does not exist in consumer-grade messengers.

**CHAPTER 2**

**LITERATURE SURVEY**

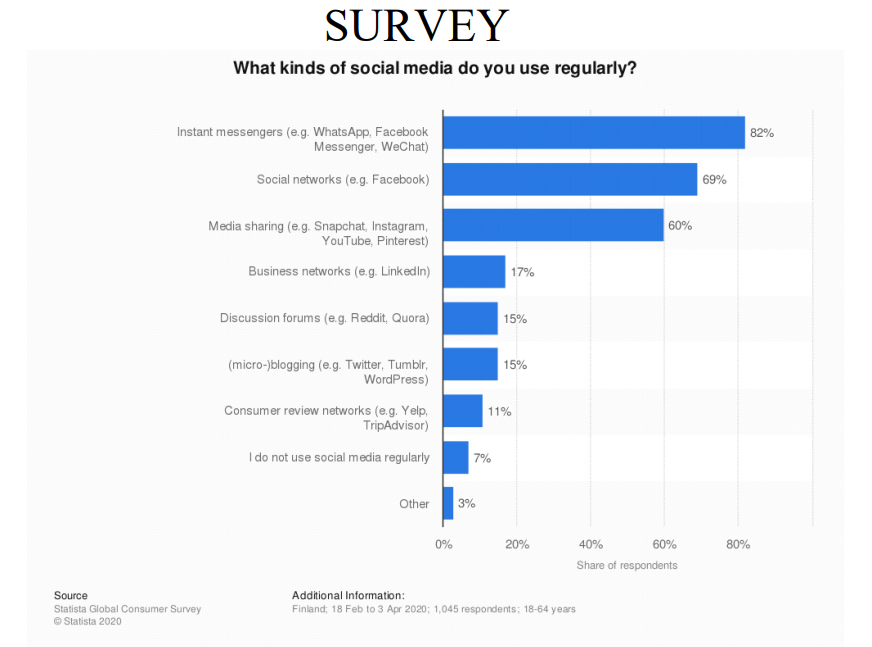
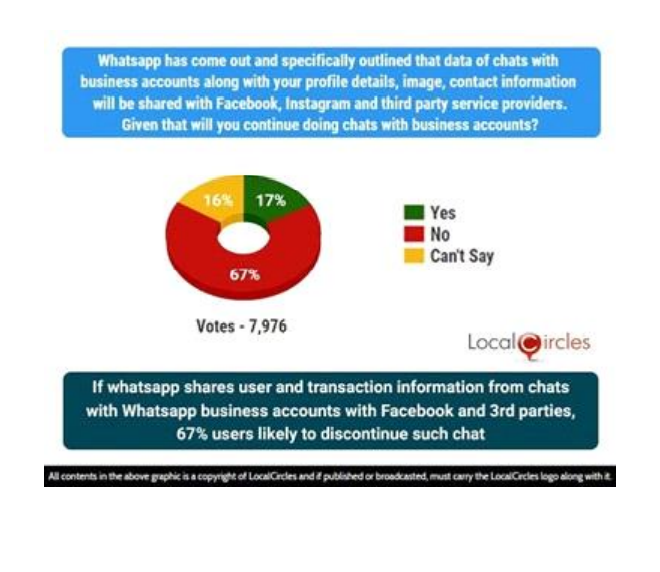
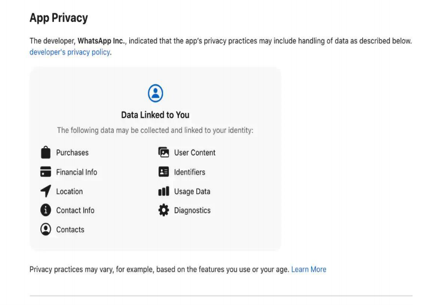
**IS METADATA-PRIVATE MESSAGING SO IMPORTANT?**

The rise of the data economy is driving enormous value by collecting, sharing, using and misuse of data. It has already enabled the building of global digital empires whose services we happily use and benefit from every day, throughout our day. As technology advances and the use increases, we become more and more dependent on it. Our dependence, however, makes us more vulnerable to security threats. From internet browsing to search engines, social media and messengers - even state-sponsored mass-surveillance programs - many outlets are collecting sensitive information about their users and the communication between them. And more often than not, the users know nothing about it. The world has become more connected, collaborative, proactive, and so on, making privacy increasingly harder to maintain.

**HAVE YOU READ ANY PRIVACY POLICY FOR MESSENGER APPLICATIONS?**

If one actually attempts to read the privacy policy of any given app, the digital advertising company third parties that receive your detailed profile are often not mentioned by name. If the third parties are actually listed, and there are typically hundreds of them, the user then has to read the privacy policies of these third parties to understand how the data is being used. Data privacy is not in the interest of the app providers because there is a lucrative financial benefit to be made by the use of your privacy.

Unfortunately, data privacy is often underestimated or passed over by IT and information security teams who routinely allow staff to use publicly available, consumer-grade messengers. Encryption alone is just simply not sufficient to ensure user or business privacy. It is essential to understand that security and privacy are two distinct things. As mentioned, security is about the safeguarding of data, whereas privacy is about safeguarding the user’s identity. Keep in mind that both are essential. Interestingly, you can have data protection without data privacy, but you cannot have data privacy without data protection. When privacy is enhanced, so is security.



**CHAPTER 3**

**SYSTEM ANALYSIS**

3.1 EXISTING SYSTEM

Encryption in messenger apps does not always mean that metadata of the communication is protected. Anyone capable of tapping into the communication can theoretically gain access to collected metadata such as the timing of the conversation, who is chatting with whom, the conversations’ traffic volumes and the identities of the devices. Metadata is sometimes just as important as the sensitive content of the messages.

Intelligence organizations have a substantial interest in metadata, even for encrypted conversations, since it often makes the need for the actual content irrelevant. Metadata reveals a lot about the underlying content. As an example, by combining and analyzing data points, government agents can deduce that just the existence of a conversation between two suspected criminals may be enough evidence to be considered actionable intelligence. If your adversaries are after your business secrets, they may be able to gain insight by accessing your metadata.

3.2 PROPOSED SYSTEM

The drawback with modern messengers is that users need a phone number tied to

their name and which has their personal information.

our application targets users who cannot carry their personal phone with them at all

time or if their battery died and they are in an emergency situation and need to

contact their family and friends they cannot use their account since they need their

number and one-time password.

Our application tackles the problem by using a simple login method such as email

and password so users can login to their accounts at any time and wherever they

want.

**3.3 FEASIBILITY STUDY**

A feasibility study is carried out to select the best system that meets performance

requirements. The main aim of the feasibility study activity is to determine that it

would be financially and technically feasible to develop the product.

**3.3.1 TECHNICAL FEASIBILITY**

This is concerned with specifying the software will successfully satisfy the user

requirement. Open source and business-friendly and it is truly cross platform, easily

deployed and highly extensible.

**3.3.2 ECONOMIC FEASIBILITY**

Economic analysis is the most frequently used technique for evaluating the

effectiveness of a proposed system. The enhancement of the existing system doesn’t

incur any kind of drastic increase in the expenses. Python is open source and ready

available for all users. Since the project is runned in python and jupyter notebook

hence is cost efficient.

**3.4 SYSTEM CONFIGURATION**

3.4.1 HARDWARE CONFIGURATION

* Processor - Intel Core i5
* RAM - 8 GB
* Hard Disk - 1 TB

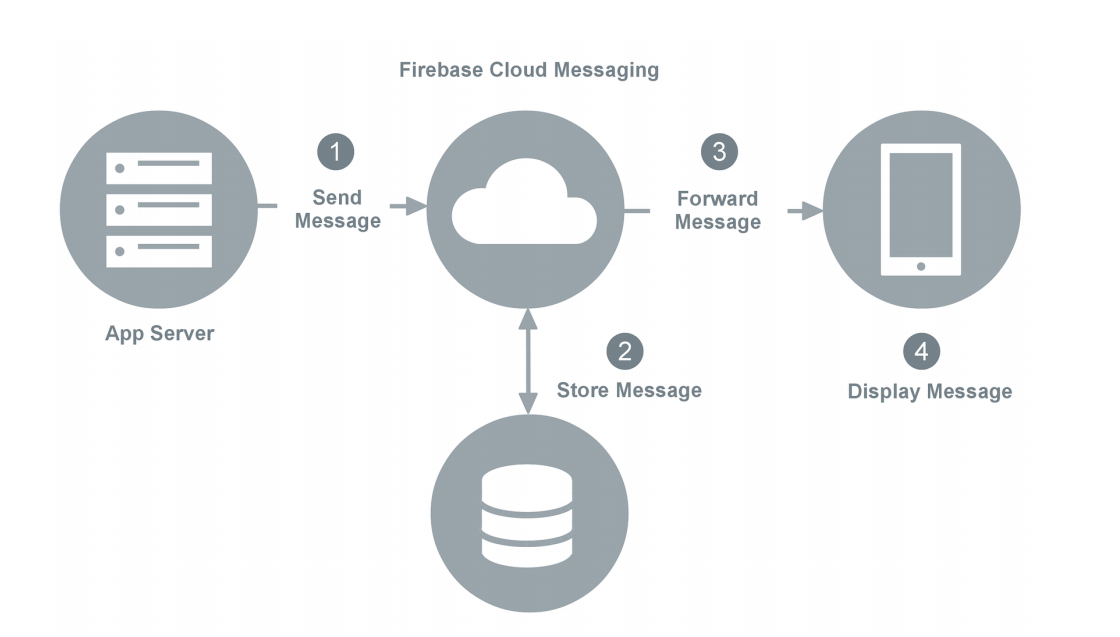
3.4.2 SOFTWARE CONFIGURATION

* Operating System - Mac OSX  Mojave
* Android Studio
* Programming Language : Flutter , Dart

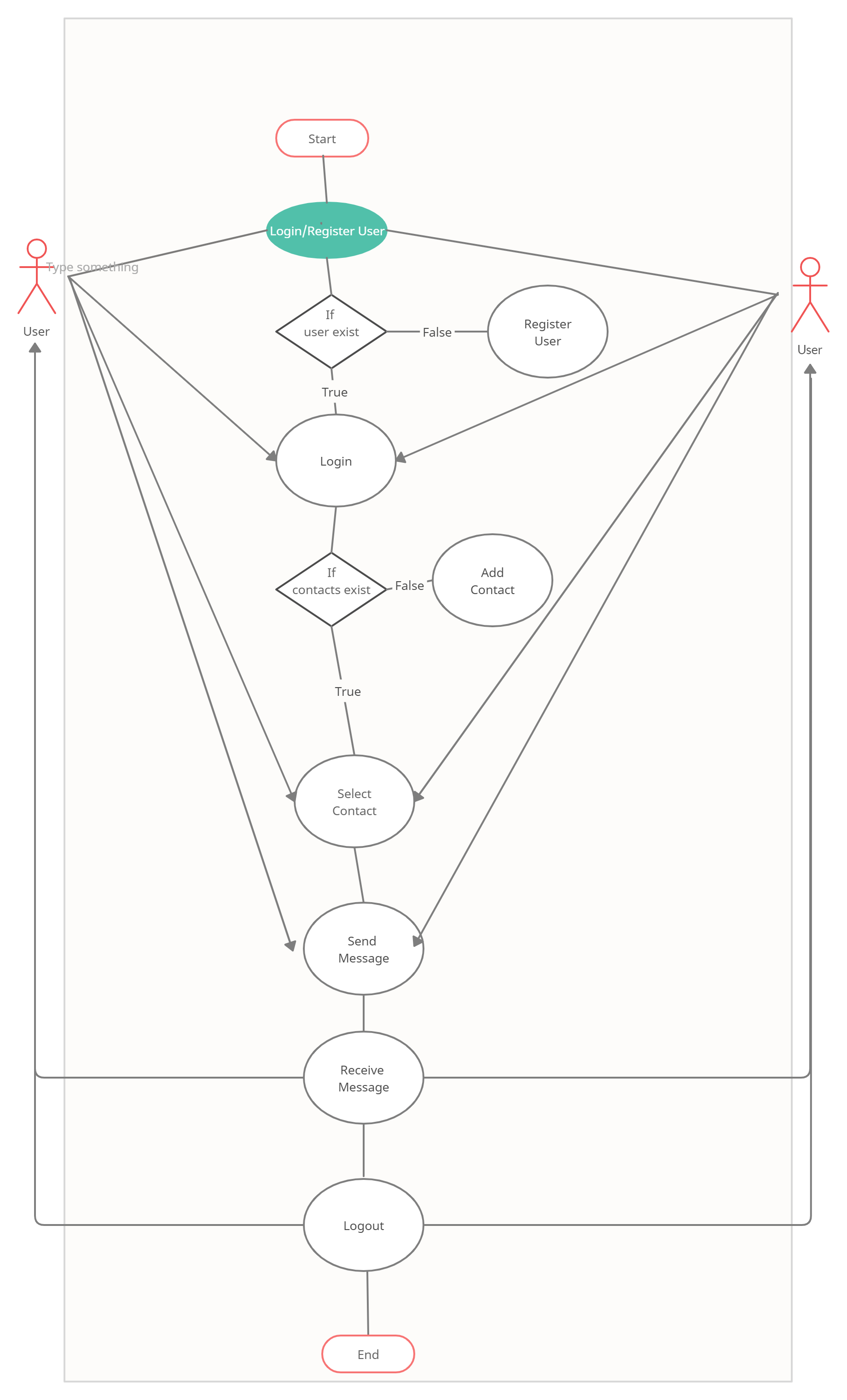
**CHAPTER 4**

ARCHITECTURE

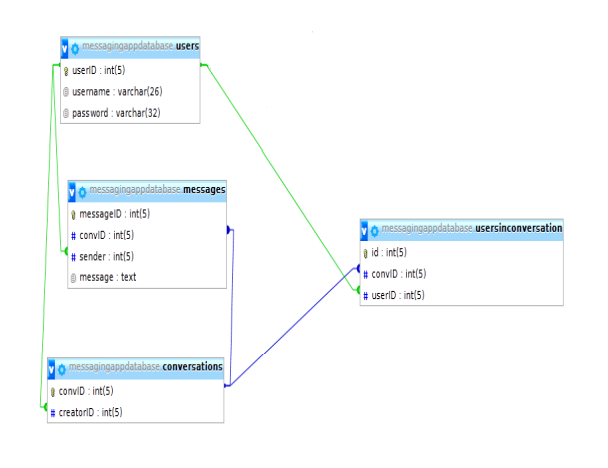
4.1 SYSTEM ARCHITECTURE



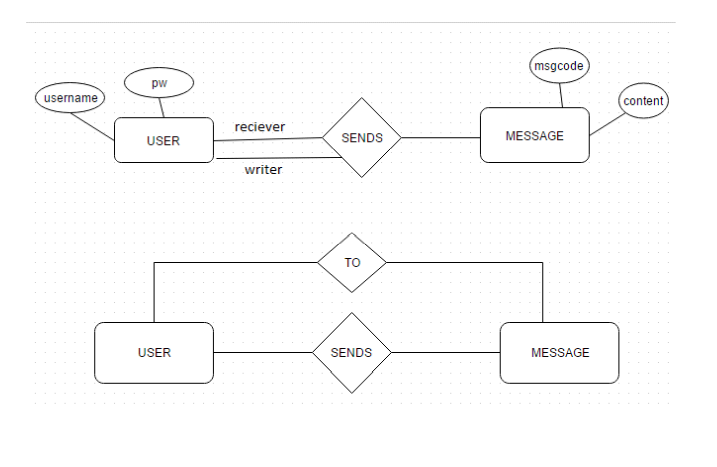
4.2 UML DIAGRAM



4.2.1 USECASE DIAGRAM



4.2.2 ACTIVITY DIAGRAM



CHAPTER 5

SYSTEM DESIGN

**WELCOME PAGE**

class \_WelcomeScreenState extends State<WelcomeScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.*white*,

      body: Padding(

        padding: EdgeInsets.symmetric(horizontal: 24.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: <Widget>[

            Row(

              children: <Widget>[

                Hero(

                  tag: 'logo',

                  child: Container(

                    child: Image.asset('images/logo1.png'),

                    height: 60.0,

                  ),

                ),

                Text(

                  'Privacy Messenger',

                  style: TextStyle(

                    fontSize: 30.0,

                    fontWeight: FontWeight.*w900*,

                  ),

                ),

              ],

            ),

            SizedBox(

              height: 48.0,

            ),

            Padding(

              padding: EdgeInsets.symmetric(vertical: 16.0),

              child: Material(

                elevation: 5.0,

                color: Colors.*lightBlueAccent*,

                borderRadius: BorderRadius.circular(30.0),

                child: MaterialButton(

                  onPressed: () {

                    //Go to login screen.

                    Navigator.*pushNamed*(context, LoginScreen.*id*);

                  },

                  minWidth: 200.0,

                  height: 42.0,

                  child: Text(

                    'Log In',

                  ),

                ),

              ),

            ),

            Padding(

              padding: EdgeInsets.symmetric(vertical: 16.0),

              child: Material(

                color: Colors.*blueAccent*,

                borderRadius: BorderRadius.circular(30.0),

                elevation: 5.0,

                child: MaterialButton(

                  onPressed: () {

                    //Go to registration screen.

                    Navigator.*pushNamed*(context, RegistrationScreen.*id*);

                  },

                  minWidth: 200.0,

                  height: 42.0,

                  child: Text(

                    'Register',

                  ),

                ),

              ),

            ),

          ],

        ),

      ),

    );

**LOGIN SCREEN**

Widget build(BuildContext context) {

  return Scaffold(

    backgroundColor: Colors.*white*,

    body: ModalProgressHUD(

      inAsyncCall: showSpinner,

      child: Padding(

        padding: EdgeInsets.symmetric(horizontal: 24.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: <Widget>[

            Hero(

              tag: 'logo',

              child: Container(

                height: 200.0,

                child: Image.asset('images/logo1.png'),

              ),

            ),

            SizedBox(

              height: 48.0,

            ),

            TextField(

              keyboardType: TextInputType.*emailAddress*,

              textAlign: TextAlign.center,

                style: TextStyle(color: Colors.*black*),

              onChanged: (value) {

                email = value;

                //Do something with the user input.

              },

              decoration: InputDecoration(

                floatingLabelBehavior: FloatingLabelBehavior.always,

                hintStyle: TextStyle(color: Colors.*grey*),

                hintText: 'enter you email',

                contentPadding:

                    EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),

                border: OutlineInputBorder(

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                enabledBorder: OutlineInputBorder(

                  borderSide:

                      BorderSide(color: Colors.*lightBlueAccent*, width: 1.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                focusedBorder: OutlineInputBorder(

                  borderSide:

                      BorderSide(color: Colors.*lightBlueAccent*, width: 2.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

              ),

            ),

            SizedBox(

              height: 8.0,

            ),

            TextField(

              obscureText: true,

              textAlign: TextAlign.center,

              style: TextStyle(color: Colors.*black*),

              onChanged: (value) {

                password = value;

                //Do something with the user input.

              },

              decoration: InputDecoration(

                floatingLabelBehavior: FloatingLabelBehavior.always,

                hintStyle: TextStyle(color: Colors.*grey*),

                hintText: 'Enter your password.',

                contentPadding:

                    EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),

                border: OutlineInputBorder(

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                enabledBorder: OutlineInputBorder(

                  borderSide:

                      BorderSide(color: Colors.*lightBlueAccent*, width: 1.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                focusedBorder: OutlineInputBorder(

                  borderSide:

                      BorderSide(color: Colors.*lightBlueAccent*, width: 2.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

              ),

            ),

            SizedBox(

              height: 24.0,

            ),

            Padding(

              padding: EdgeInsets.symmetric(vertical: 16.0),

              child: Material(

                color: Colors.*lightBlueAccent*,

                borderRadius: BorderRadius.all(Radius.circular(30.0)),

                elevation: 5.0,

                child: MaterialButton(

                  onPressed: () async{

                    setState(() {

                      showSpinner = true;

                    });

                    //Implement login functionality.

                    try {

                      final user = await \_auth.signInWithEmailAndPassword(

                          email: email, password: password);

                      if (user != null) {

                        Navigator.*pushNamed*(context, ChatScreen.*id*);

                      }

                      setState(() {

                        showSpinner = false;

                      });

                    }

                    catch (e){

                      print(e);

                    }

                  },

                  minWidth: 200.0,

                  height: 42.0,

                  child: Text(

                    'Log In',

                  ),

**REGISTRATION SCREEN**

Widget build(BuildContext context) {

  return Scaffold(

    backgroundColor: Colors.*white*,

    body: ModalProgressHUD(

      inAsyncCall: showSpinner,

      child: Padding(

        padding: EdgeInsets.symmetric(horizontal: 24.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: <Widget>[

            Hero(

              tag: 'logo',

              child: Container(

                height: 200.0,

                child: Image.asset('images/logo1.png'),

              ),

            ),

            SizedBox(

              height: 48.0,

            ),

            TextField(

              keyboardType: TextInputType.*emailAddress*,

              textAlign: TextAlign.center,

              style: TextStyle(color: Colors.*black*),

              onChanged: (value) {

                //Do something with the user input.

                email = value;

              },

              decoration: InputDecoration(

                floatingLabelBehavior: FloatingLabelBehavior.always,

                hintStyle: TextStyle(color: Colors.*grey*),

                hintText: 'Enter your email',

                contentPadding:

                    EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),

                border: OutlineInputBorder(

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                enabledBorder: OutlineInputBorder(

                  borderSide: BorderSide(color: Colors.*blueAccent*, width: 1.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                focusedBorder: OutlineInputBorder(

                  borderSide: BorderSide(color: Colors.*blueAccent*, width: 2.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

              ),

            ),

            SizedBox(

              height: 8.0,

            ),

            TextField(

              obscureText: true,

              textAlign: TextAlign.center,

              style: TextStyle(color: Colors.*black*),

              onChanged: (value) {

                //Do something with the user input.

                password = value;

              },

              decoration: InputDecoration(

                floatingLabelBehavior: FloatingLabelBehavior.always,

                hintStyle: TextStyle(color: Colors.*grey*),

                hintText: 'Enter your password',

                contentPadding:

                    EdgeInsets.symmetric(vertical: 10.0, horizontal: 20.0),

                border: OutlineInputBorder(

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                enabledBorder: OutlineInputBorder(

                  borderSide: BorderSide(color: Colors.*blueAccent*, width: 1.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

                focusedBorder: OutlineInputBorder(

                  borderSide: BorderSide(color: Colors.*blueAccent*, width: 2.0),

                  borderRadius: BorderRadius.all(Radius.circular(32.0)),

                ),

              ),

            ),

            SizedBox(

              height: 24.0,

            ),

            Padding(

              padding: EdgeInsets.symmetric(vertical: 16.0),

              child: Material(

                color: Colors.*blueAccent*,

                borderRadius: BorderRadius.all(Radius.circular(30.0)),

                elevation: 5.0,

                child: MaterialButton(

                  onPressed: () async {

                    WidgetsFlutterBinding.*ensureInitialized*();

                    await Firebase.*initializeApp*();

                    setState(() {

                      showSpinner = true;

                    });

                    try {

                      final newUser = await \_auth

                          .createUserWithEmailAndPassword(

                          email: email, password: password);

                          if (newUser != null){

                            Navigator.*pushNamed*(context, ChatScreen.*id*);

                          }

                          setState(() {

                            showSpinner = false;

                          });

                      //Implement registration functionality.

                    }

                    catch (e){

                      print(e);

                    }

                  },

                  minWidth: 200.0,

                  height: 42.0,

                  child: Text(

                    'Register',

                    style: TextStyle(color: Colors.*white*),

                  ),

**CHAT SCREEN**

Widget build(BuildContext context) {

  return Scaffold(

    appBar: AppBar(

      leading: null,

      actions: <Widget>[

        IconButton(

            icon: Icon(Icons.*close*),

            onPressed: () {

              //Implement logout functionality

              \_auth.signOut();

              Navigator.*pop*(context);

            }),

      ],

      title: Text('Privacy Messenger'),

      backgroundColor: Colors.*lightBlueAccent*,

    ),

    body: SafeArea(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.spaceBetween,

        crossAxisAlignment: CrossAxisAlignment.stretch,

        children: <Widget>[

          StreamBuilder<QuerySnapshot>(

            stream: \_firestore.collection('messages').orderBy('date').snapshots(),

            builder: (context, snapshot){

              if (snapshot.hasData){

                final messages = snapshot.data.docs.reversed;

                List<MessageBubble> messageBubbles = [];

                for (var message in messages){

                  final messageText = message.data()['text'];

                  final messageSender = message.data()['sender'];

                  final currentUser = loggedInUser;

                  final messageBubble = MessageBubble(sender: messageSender, text: messageText, isMe: currentUser == messageSender,);

                  /\*Text('$messageText from $messageSender',

                  style: TextStyle(

                    fontSize: 20,

                  ),);\*/

                  messageBubbles.add(messageBubble);

                }

                return Expanded(

                  child: ListView(

                    reverse: true,

                    padding: EdgeInsets.symmetric(horizontal: 10.0, vertical: 20.0),

                    children: messageBubbles,

                  ),

                );

              }

            },

          ),

          Container(

            decoration: kMessageContainerDecoration,

            child: Row(

              crossAxisAlignment: CrossAxisAlignment.center,

              children: <Widget>[

                Expanded(

                  child: TextField(

                    controller: messageTextController,

                    onChanged: (value) {

                      //Do something with the user input.

                      messageText = value;

                    },

                    decoration: kMessageTextFieldDecoration,

                  ),

                ),

                FlatButton(

                  onPressed: () {

                    //Implement send functionality.

                    messageTextController.clear();

                    \_firestore.collection('messages').add({

                      'text': messageText,

                      'sender': loggedInUser,

                      'date': DateTime.now().toIso8601String().toString(),

                  });

                  },

                  child: Text(

                    'Send',

                    style: kSendButtonTextStyle,

                  ),

CHAPTER 6

TESTING

6.1 INTRODUCTION

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 as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing

6.2TYPES OF TESTS

6.2.1UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

⦁ Flow graph notation

⦁ Kilometric complexity

⦁ Deriving test cases

⦁ Graph matrices Control

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.2.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.2.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 7**

CONCLUSION

**7.1 APPLICATIONS**

MESSENGER:

It can be used as a normal messenger for communication without worrying about

their privacy.

PROVIDING ANONYMOUS TIPS FOR LAW OFFICIALS

Our application can be used to provide anonymous tips to the law officials just by

registering with a mail id.

**7.2 FUTURE ENHANCEMENTS**

1. Implementation of video call functionality.
2. Adding video files to messages.
3. Adding GIFs support
4. adding file attachments
5. adding image attachments

**7.3 CONCLUSION**

We have successfully designed a PRIVACY FOCUSED MESSENGER.Our group

messenger is controlled entirely by the users and with a little technical knowledge

users can add their own server for their community group text.

We believe that privacy is a basic human right which should be respected by

everyone.We as people need privacy and it is a basic constitutional right that every

human should have. Many companies and organizations including private and

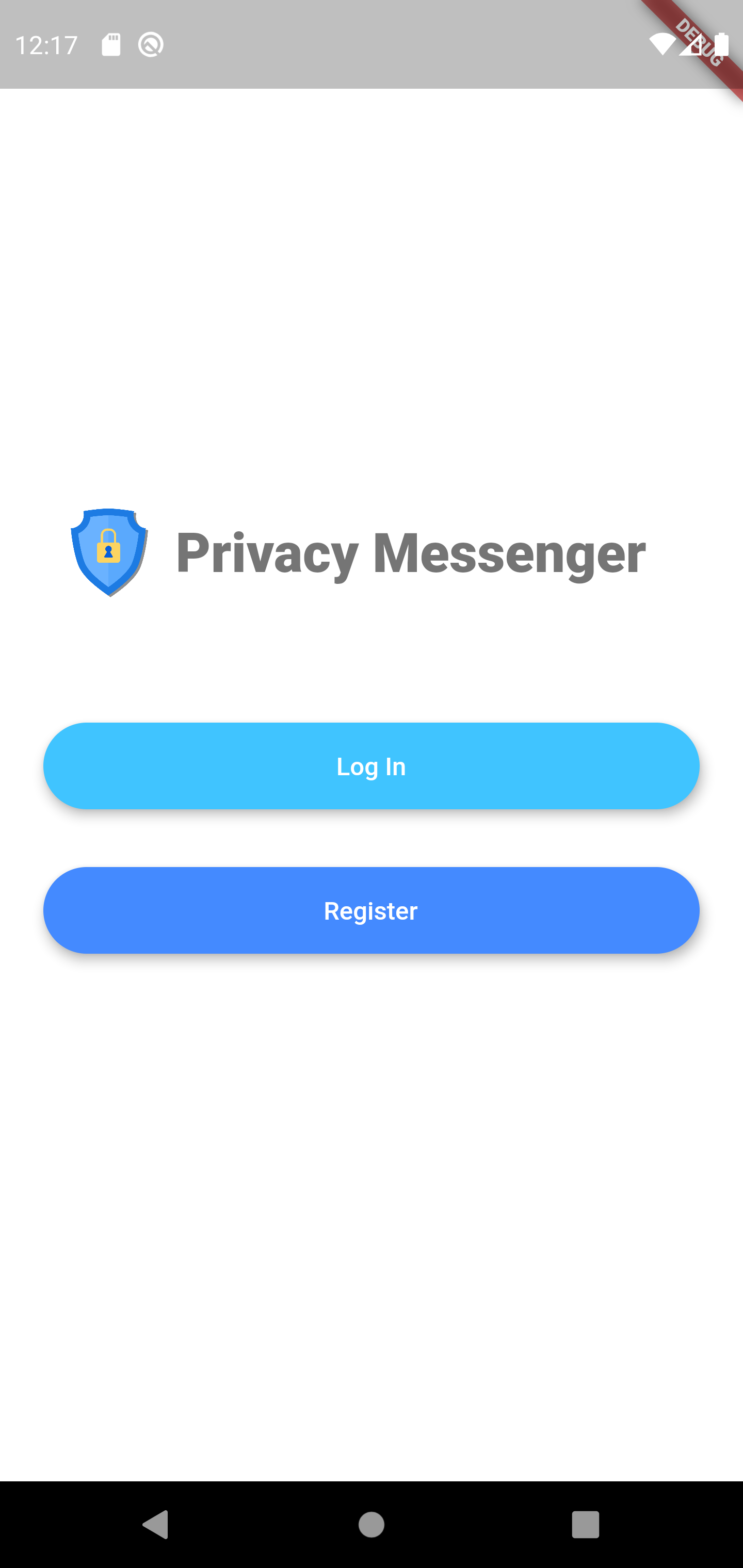
government organizations spy on their users in the name of national security.

Let us encourage free and open-source projects where the users can view the

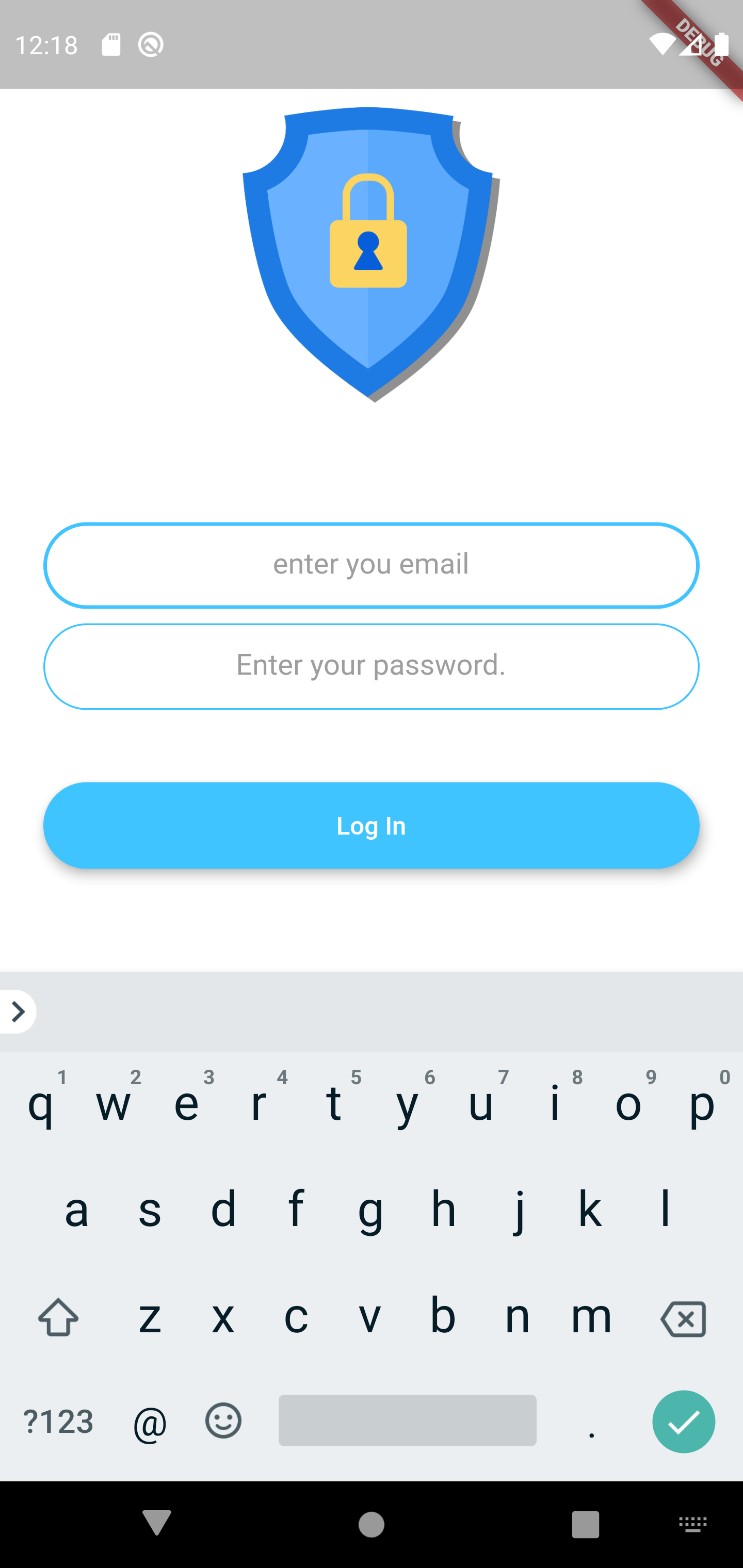
source code and everything is free from telemetry.

**SCREENSHOTS**

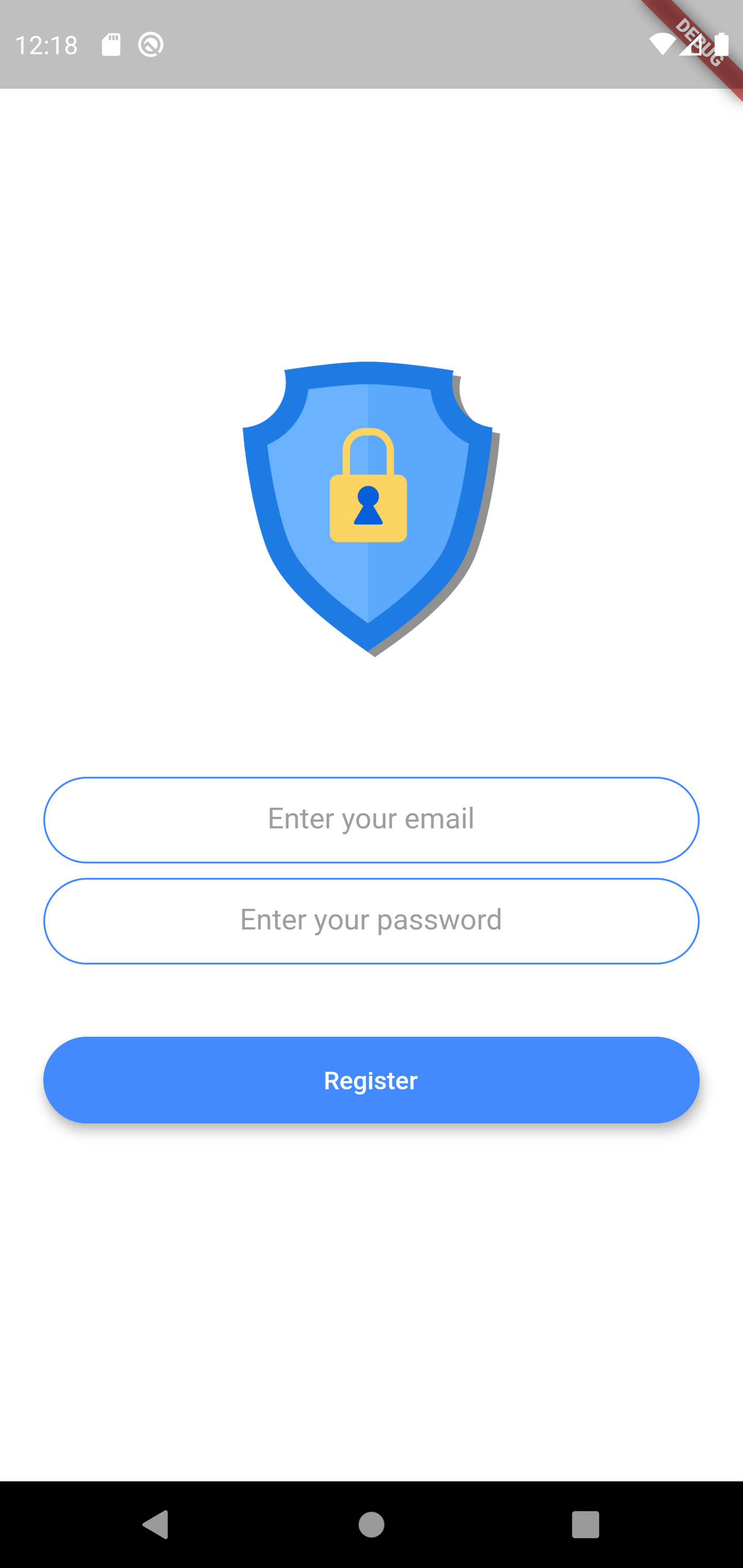
**WELCOME SCREEN**



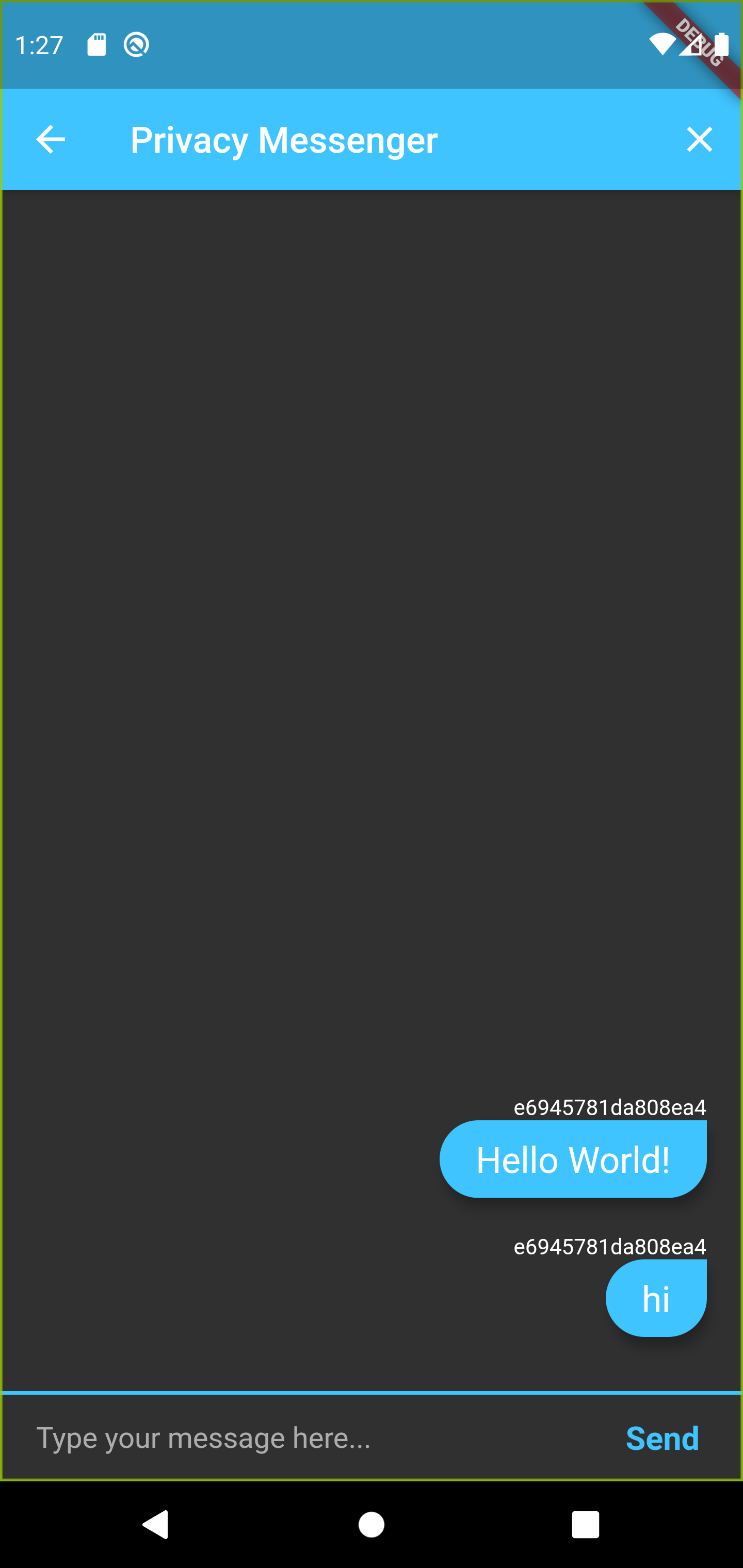
**LOGIN SCREEN**



**REGISTER SCREEN**



**CHAT SCREEN**



**REFERENCES**

1. Android studio documentation, google, https://developer.android.com/docs
2. Flutter documentation, google , https://flutter.dev/docs
3. Firebase documentation,google , https://firebase.google.com/docs
4. xcode developer documentation,apple, https://developer.apple.com/documentation/xcode/