Poznan University of Technology

Faculty of Computer Science and Telecommunications

Dawid Sienkiewicz

140773

Piotr Jankowski

141048

Application Security

Mobile chat application

Cybersecurity

November 22th, 2022

Table of contents:

1. General characteristics of the project

The presented documentation describes the project of the mobile application designed to send messages between users. The application consists of a separate login panel, a main page including all users ready to chat and a page with communication containing messages already sent by specified users and an opportunity to start or continue a conversation. All the functionalities are listed below:

- registration,
- authentication,
- viewing of users,
- sending messages,
- receiving messages,
- displaying of messages sent previously.

1.1 Roles of particular users

The specified actions are assigned to two types of users which are defined in the application: an unlogged user and a logged in user.

An unlogged user is able to create a new account and sign in to it.

A logged user has a possibility to view a list of users, send a message, receive a message and view previous communication.

In this application an administrator can be also mentioned. It is a user which has access to the database but not from the application. He has an opportunity to delete user accounts and delete messages.

The mentioned roles of the particular users are revealed in the pictures below.

1.2 Technologies

In order to accomplish the application the specified technologies will be used. The tools and their responsibilities are mentioned below:

- Android Studio with Kotlin to provide the application logic and XML to represent the data,
- Firebase to ensure the registration, authentication and data storage.

2. Requirements

In this section are presented functional and non-functional requirements. We can divide the first group due to types of actors.

2.1 Functional requirements

They mention services offered by the system and specific behaviors when receiving data. Here are listed functional requirements:

- for an unlogged user:
    - registration with name, email and password,

    - logging in using email and password,
- for a logged in user:
    - viewing of users,
    - sending messages,
    - receiving messages,
    - displaying of messages sent previously.

2.2 Non-functional requirements

They are used to define processing capabilities of the system and expectations towards it. Below are listed non-functional requirements:

- an intuitive interface not requiring the help in the form of additional tabs,
- the service maintains its functioning until the closure by the administrator,
- strong passwords containing letters and digits,
- the encryption for all sensitive data.

3. Diagrams

There are demonstrated two types of UML diagrams to reveal the structure of the application and its objectives. Introduced schemes are listed below:

- use case diagram,
- flow diagram.

3.1 Use case diagram

The use case diagram for the described application is presented below. There are three actors: unlogged user, logged in user and administrator who have a possibility to execute actions specified in the diagram.

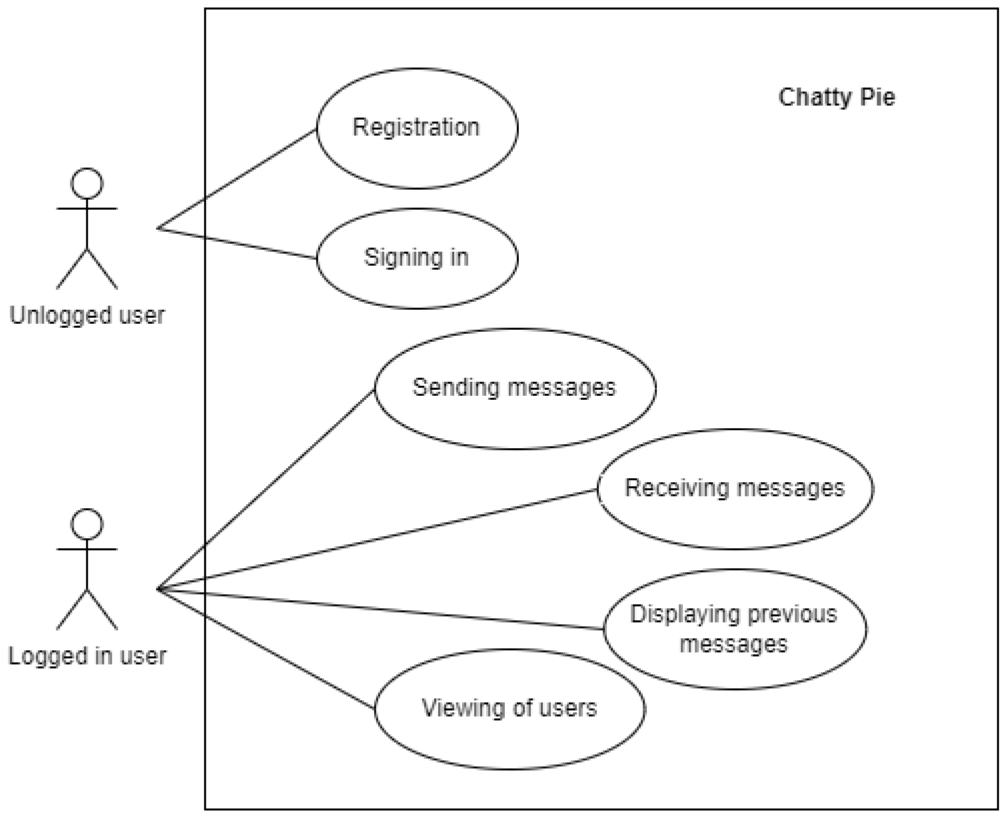


Diagram 1 - Use case diagram

3.2 Flow diagram

In the flow diagram there is presented a procedure of sending a message. There are distinguished particular activities to be performed by a user and by a system.

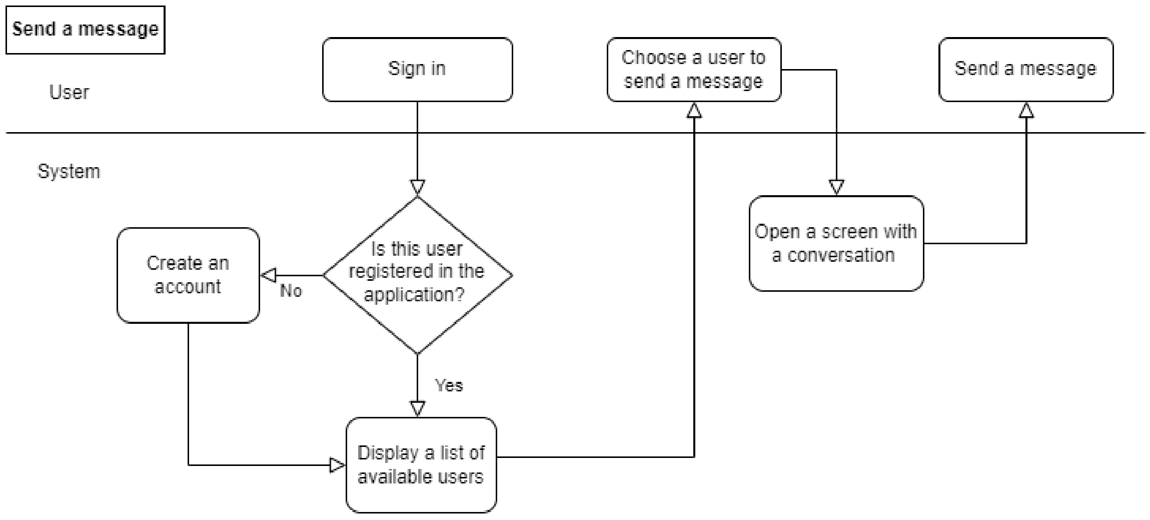
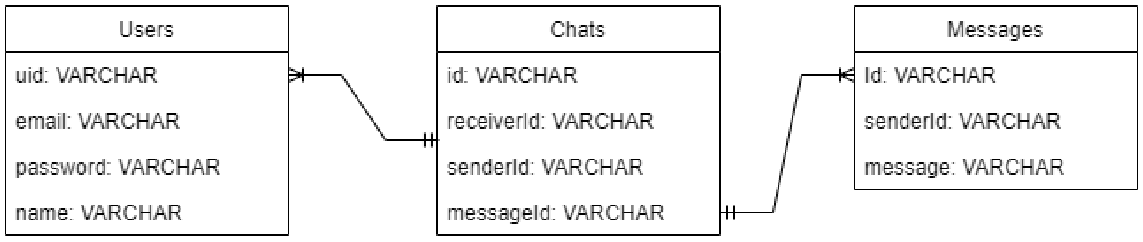

Send a message
User
System
Sign in
Choose a user to send a message
Send a message
Is this user registered in the application?
No
Yes
Create an account
Display a list of available users
Open a screen with a conversation

Diagram 2 - Flow diagram

4. Database schema

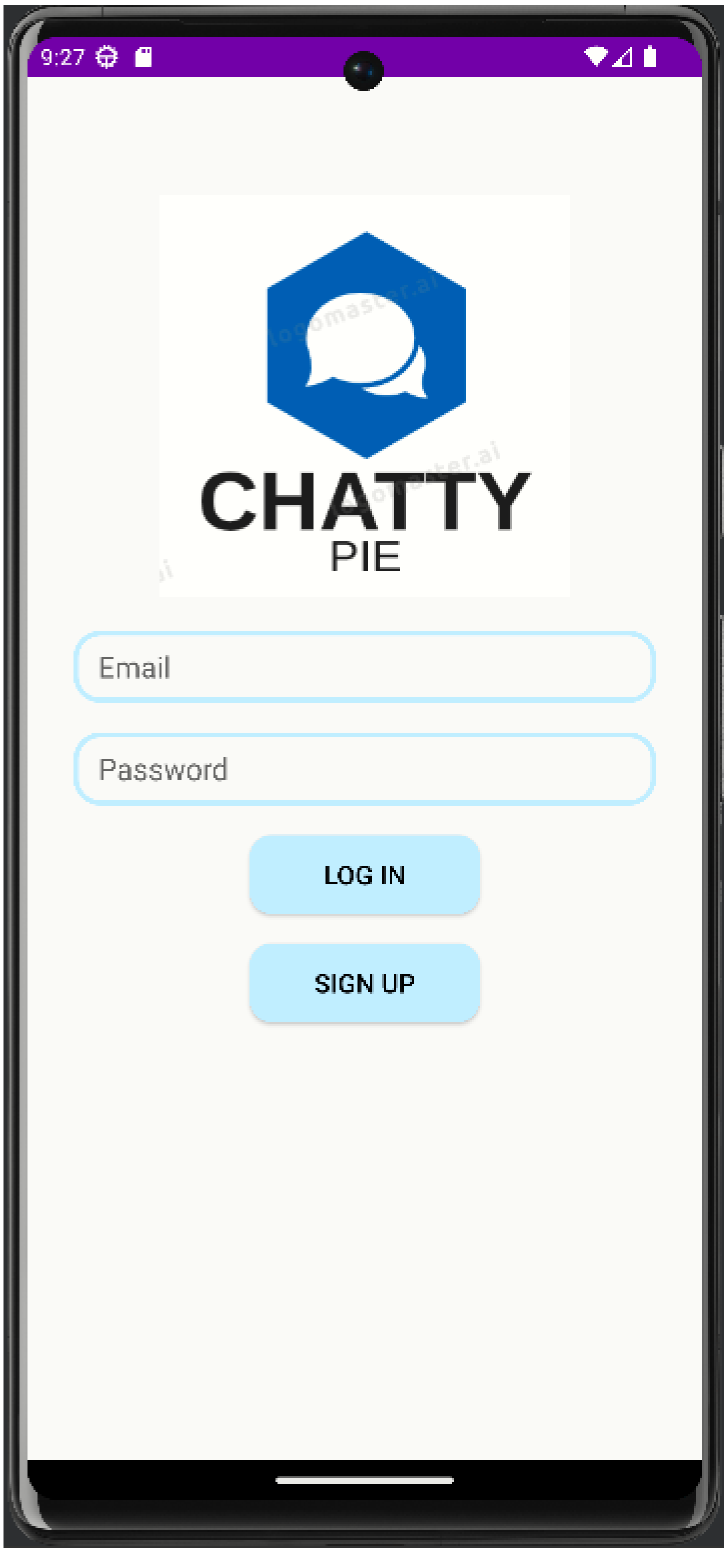
The database schema was prepared for the described project in order to reflect the structure of data and relationships between tables which must be provided to ensure the correct way of storing and updating data. The following scheme presents the database implemented in the application. There are mentioned three tables with relevant information about users having an account in the application and their chats containing particular messages.



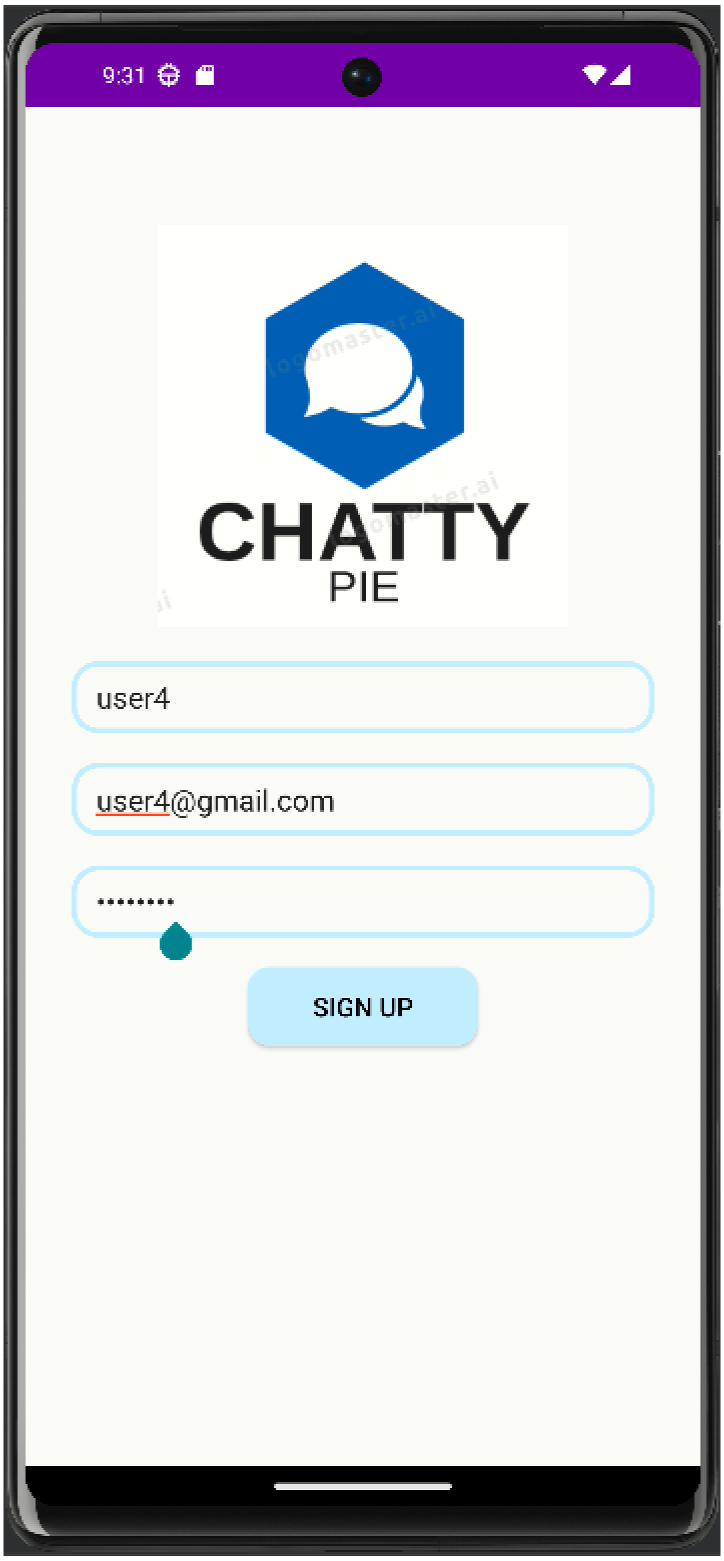
Scheme 1 - Database schema

5. Graphical interface designs

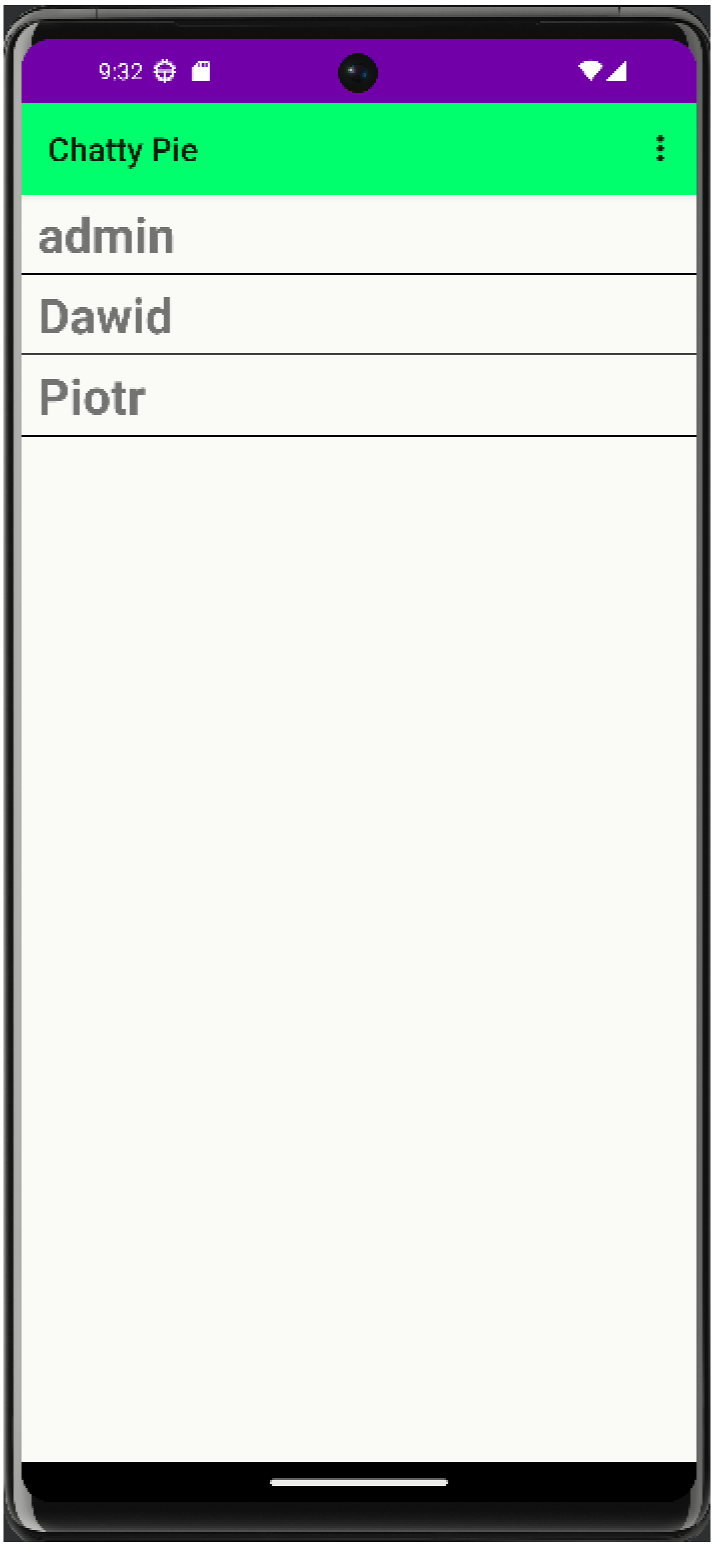
Pictures introduced below represent the specified views and functionalities of the application. The prepared interfaces include the start page (login page), the signup page, the main page with the list of users and the chat page.



Application picture 1 - Start page



Application picture 2 - Signing up

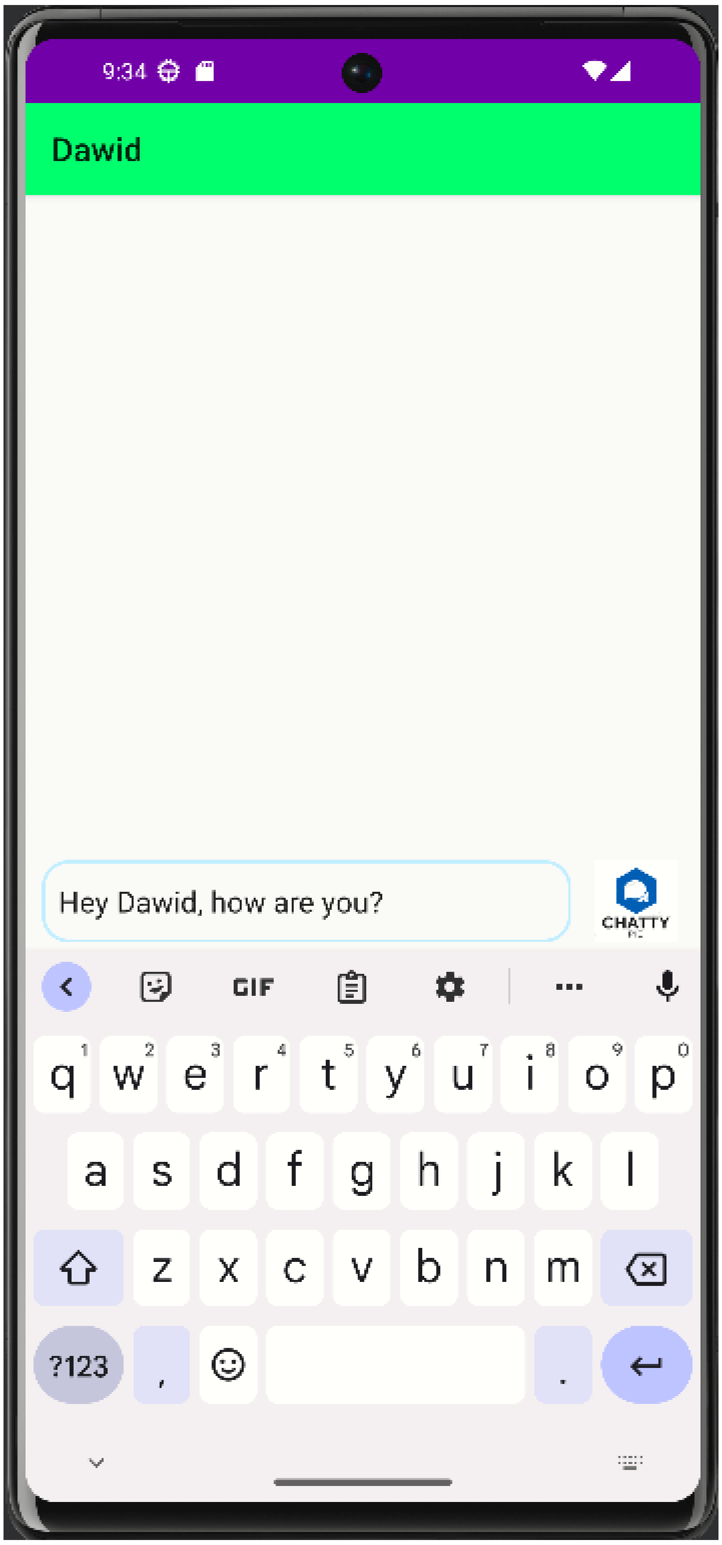


admin

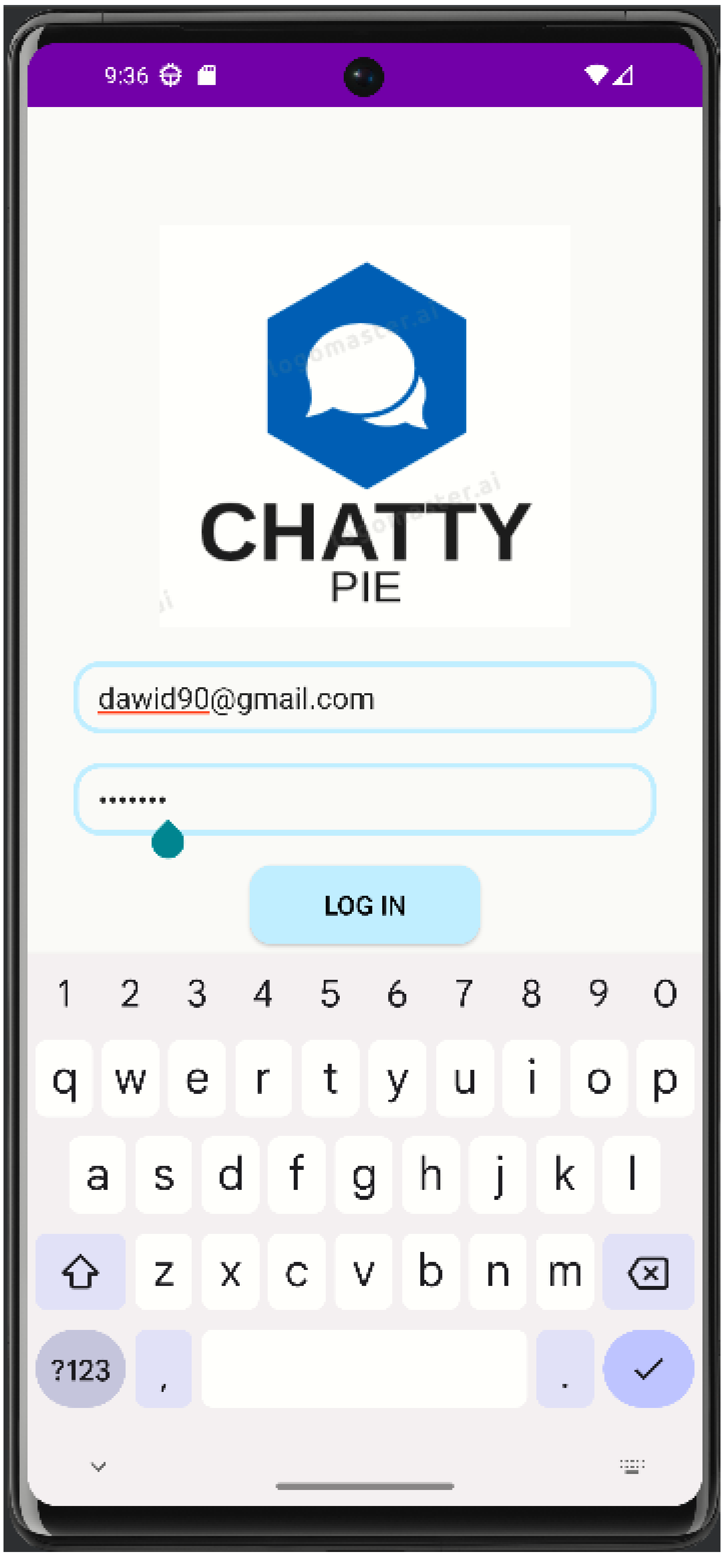
Dawid

Piotr

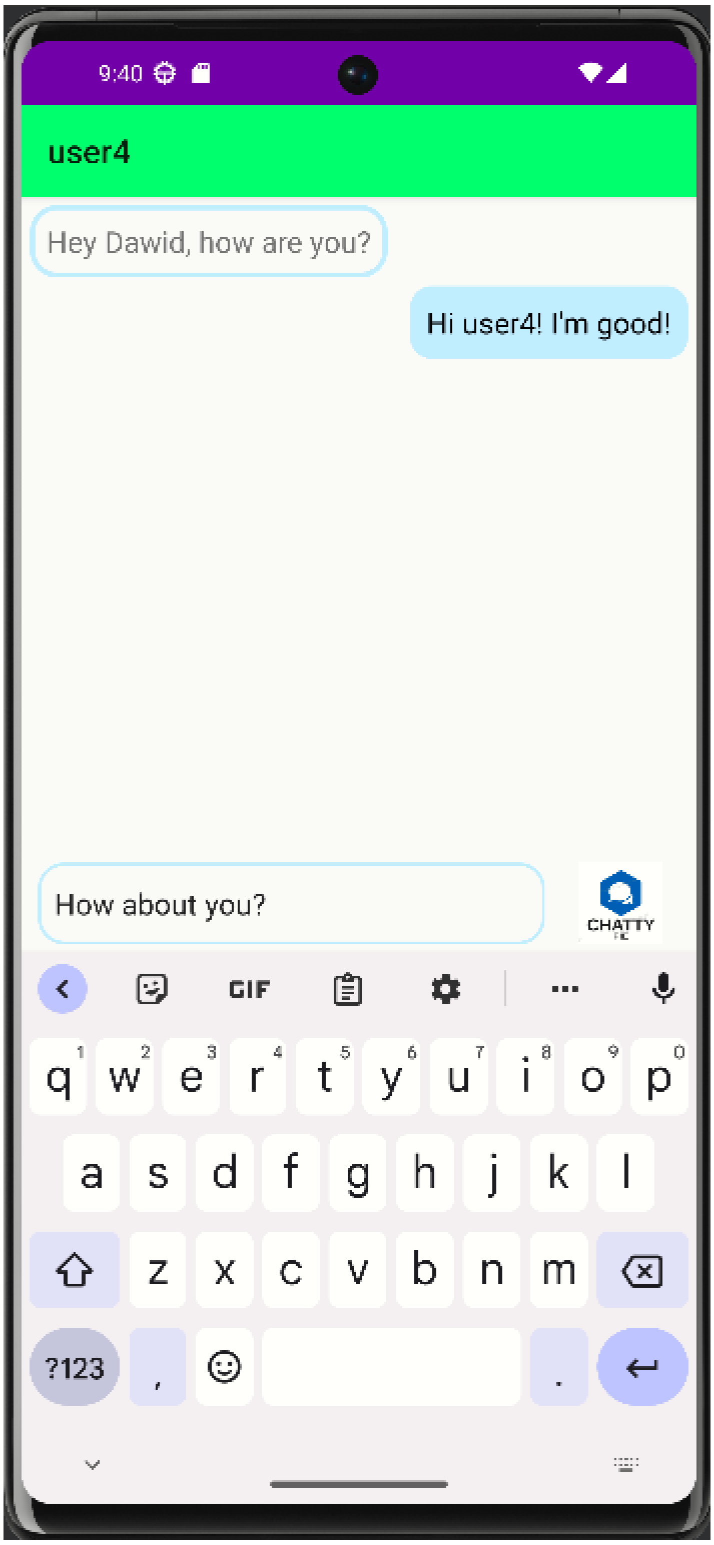
Application picture 3 - A list of users



Application picture 4 - Sending a message



Application picture 5 - Signing in



Hey Dawid, how are you?

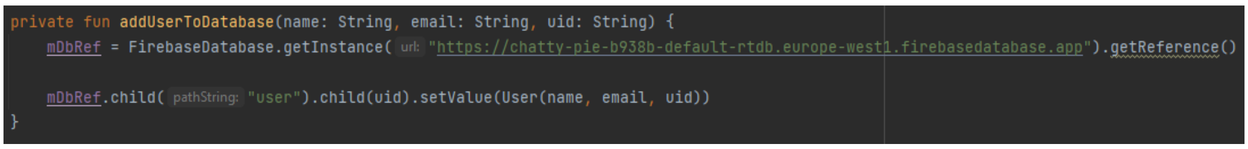
Hi user4! I'm good!

How about you?

Application picture 6 - Receiving a message and answering

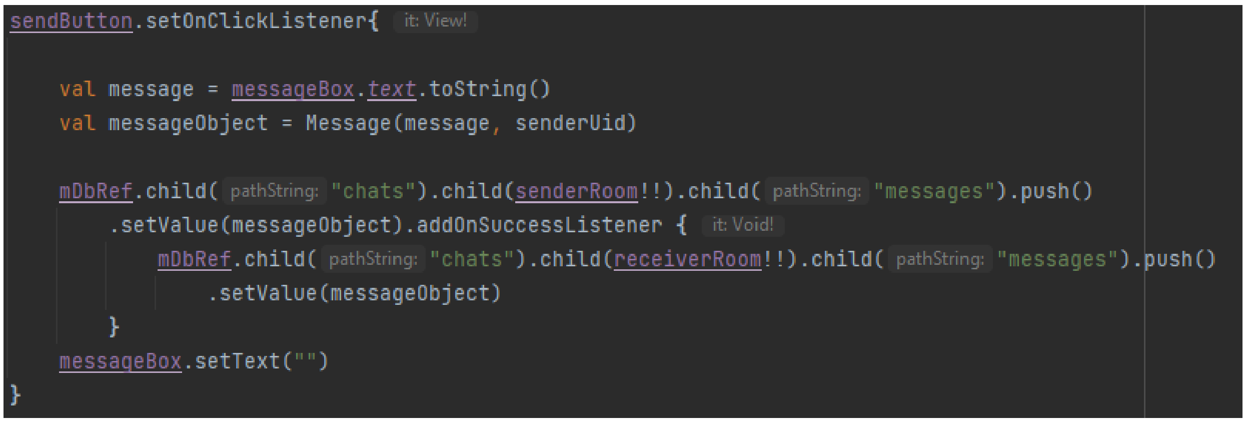
6. The most important methods and code fragments

Some functionalities are crucial to provide proper operation for clients using the application. Below are presented some code snippets with methods implementing some features like entering data into the database and user authentication.



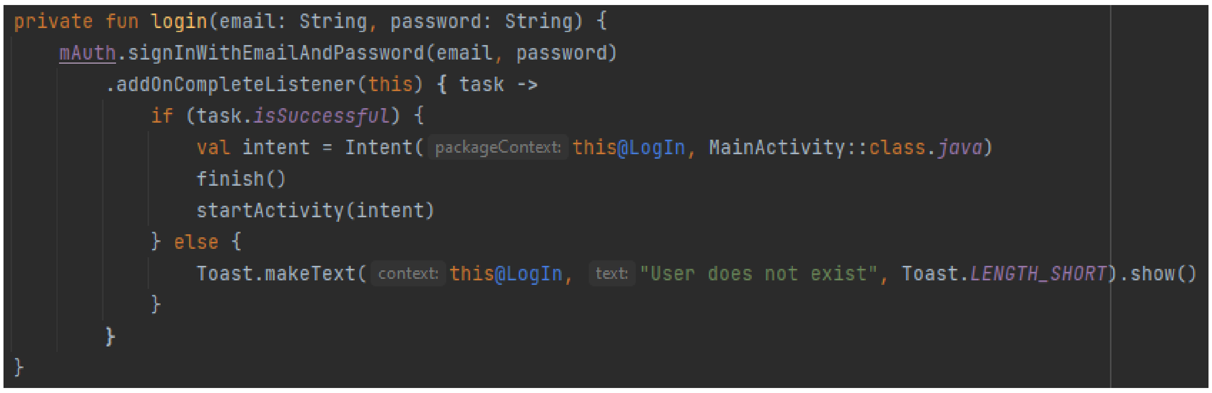
```kotlin
private fun addUserToDatabase(name: String, email: String, uid: String) {
    mDbRef = FirebaseDatabase.getInstance("https://chatty-pie-b938b-default-rtdb.europe-west1.firebasedatabase.app").getReference()

    mDbRef.child("user").child(uid).setValue(User(name, email, uid))
}
```

Code snippet 1 - Adding user to the database



```kotlin
sendButton.setOnClickListener{

    val message = messageBox.text.toString()
    val messageObject = Message(message, senderUid)

    mDbRef.child("chats").child(senderRoom!!).child("messages").push()
        .setValue(messageObject).addOnSuccessListener {
            mDbRef.child("chats").child(receiverRoom!!).child("messages").push()
                .setValue(messageObject)
        }
    messageBox.setText("")
}
```

Code snippet 2 - Adding message to the database

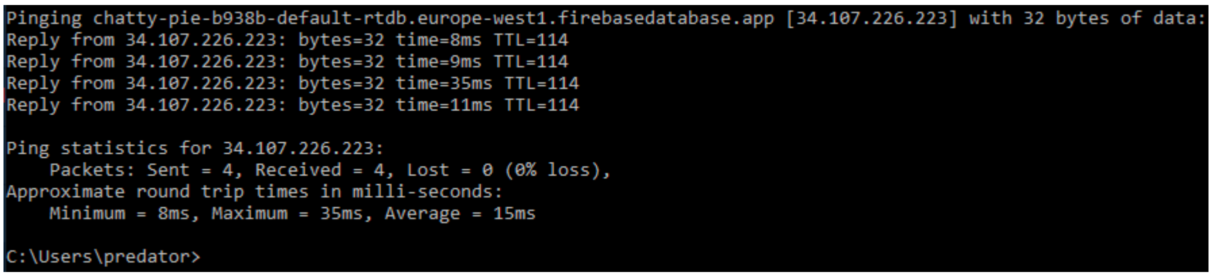


```kotlin
private fun login(email: String, password: String) {
    mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                val intent = Intent(this@LogIn, MainActivity::class.java)
                finish()
                startActivity(intent)
            } else {
                Toast.makeText(this@LogIn, "User does not exist", Toast.LENGTH_SHORT).show()
            }
        }
}
```

Code snippet 3 - Logging in

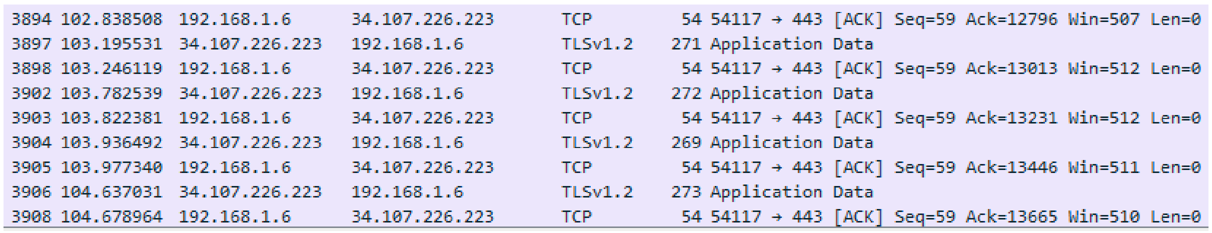
7. Security analysis

To provide sophisticated protection against the threats the specified requirements were met. All user data, like passwords, personal information and messages, are encrypted and stored in the database. Clients must use particular chars when creating their passwords. Each password should contain at least the combination of letters and digits. After every closing of the application a user will be logged out to prevent unauthorized access to the user account.



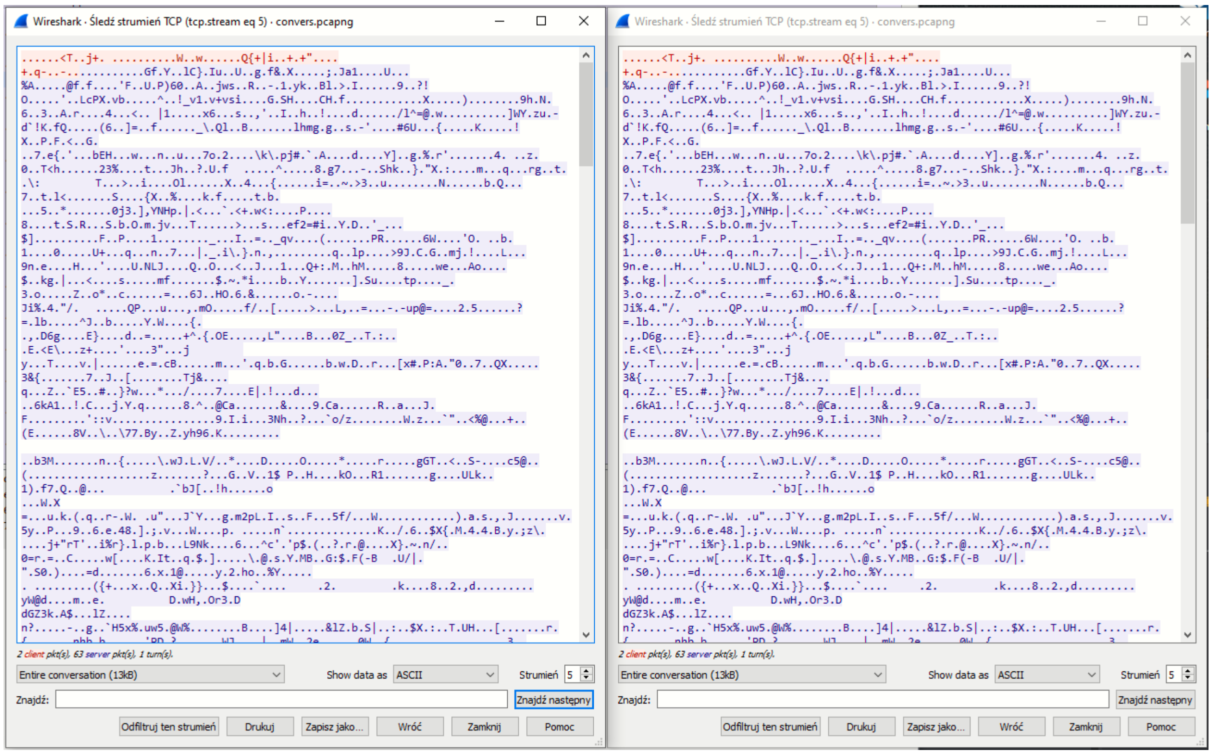
```
Pinging chatty-pie-b938b-default-rtdb.europe-west1.firebasedatabase.app [34.107.226.223] with 32 bytes of data:
Reply from 34.107.226.223: bytes=32 time=8ms TTL=114
Reply from 34.107.226.223: bytes=32 time=9ms TTL=114
Reply from 34.107.226.223: bytes=32 time=35ms TTL=114
Reply from 34.107.226.223: bytes=32 time=11ms TTL=114

Ping statistics for 34.107.226.223:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 35ms, Average = 15ms

C:\Users\predator>
```

Picture 1 - IP address of the database



```
3894 102.838508  192.168.1.6     34.107.226.223  TCP        54 54117 → 443 [ACK] Seq=59 Ack=12796 Win=507 Len=0
3897 103.195531  34.107.226.223  192.168.1.6     TLSv1.2   271 Application Data
3898 103.246119  192.168.1.6     34.107.226.223  TCP        54 54117 → 443 [ACK] Seq=59 Ack=13013 Win=512 Len=0
3902 103.782539  34.107.226.223  192.168.1.6     TLSv1.2   272 Application Data
3903 103.822381  192.168.1.6     34.107.226.223  TCP        54 54117 → 443 [ACK] Seq=59 Ack=13231 Win=512 Len=0
3904 103.936492  34.107.226.223  192.168.1.6     TLSv1.2   269 Application Data
3905 103.977340  192.168.1.6     34.107.226.223  TCP        54 54117 → 443 [ACK] Seq=59 Ack=13446 Win=511 Len=0
3906 104.637031  34.107.226.223  192.168.1.6     TLSv1.2   273 Application Data
3908 104.678964  192.168.1.6     34.107.226.223  TCP        54 54117 → 443 [ACK] Seq=59 Ack=13665 Win=510 Len=0
```

Picture 2 - Intercepted communication between the mobile application and the database



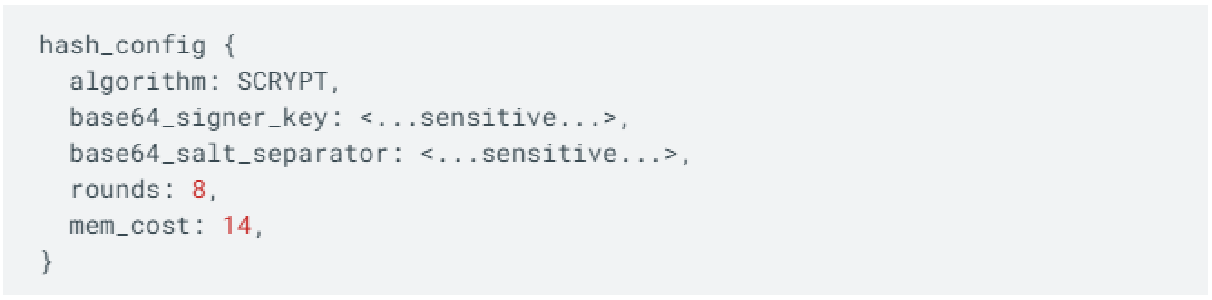
```
......<T..j+. ..........W..w......Q{+|i..+.+"....
+.q-..-...........Gf.Y..lC}.Iu..U..g.f&.X.....;.Ja1....U...
%A.....@f.f....'F..U.P)60..A..jws..R..-.1.yk..Bl.>.I......9..?!
O.....'..LcPX.vb.....^..!_v1.v+vsi....G.SH....CH.f...........X.....)........9h.N.
6..3..A.r....4...<.. |1.....x6...s..,'..I..h..!....d....../1^=@.w..........]WY.zu.-
d`!K.fQ.....(6..]=..f......_\.Ql..B.......lhmg.g..s.-'....#6U...{.....K.....!
X..P.F.<..G.
..7.e{.'...bEH...w...n..u...7o.2....\k\.pj#.`.A....d....Y]..g.%.r'.......4. ..z.
0..T<h......23%....t...Jh..?.U.f  .....^.....8.g7...-..Shk..}."X.:....m...q...rg..t.
.\:        T...>..i....Ol......X..4...{......i=..~.>3..u.........N......b.Q...
7..t.l<.......S....{X..%....k.f.....t.b.
...5..*.......0j3.],YNHp.|.<...`.<+.w<:....P....
8....t.S.R...S.b.O.m.jv...T......>...s...ef2=#i..Y.D..'_...
$].........F..P....1......._...I..=.._qv....(.......PR......6W....'O. ..b.
1....0.....U+...q...n..7...|._.i\.}.n.,.........q..lp....>9J.C.G..mj.!....L...
9n.e....H...'....U.NLJ....Q..O...<..J...1...Q+:.M..hM.....8.....we...Ao....
$..kg.|...<....s.....mf.......$.~.*i....b..Y.......].Su....tp...._.
3.o.....Z..o*..c......=...6J..HO.6.&......o.-....
Ji%.4."/.   .....QP...u...,.mO.....f/..[.....>...L,..=...-.-up@=....2.5......?
=.lb.....^J..b.....Y.W....{.
.,.D6g....E}....d..=.....+^.{.OE.....,L"....B...0Z_..T.:..
.E.<E\...z+....'....3"...j
y...T....v.|......e.=.cB......m...'.q.b.G......b.w.D..r...[x#.P:A."0..7..QX.....
3&{.......7..J..[........Tj&....
q...Z..`E5..#..}?w...*.../....7....E|.!...d...
..6kA1..!.C...j.Y.q......8.^..@Ca.......&....9.Ca......R..a...J.
F.........'::v................9.I.i...3Nh..?...`o/z.......W.z...`"..<%@...+..
(E......8V..\..\77.By..Z.yh96.K.........

..b3M.......n..{.....\.wJ.L.V/..*....D.....O.....*.....r.....gGT..<..S-....c5@..
(...................z.......?...G..V..1$ P..H....kO...R1.......g....ULk..
1).f7.Q..@...          .`bJ[..!h......o
...W.X
=...u.k.(.q..r-.W. .u"...J`Y...g.m2pL.I..s..F...5f/...W............).a.s.,.J.......v.
5y..P...9..6.e.48.].;.v...W....p. .....n`.............K../.6..$X{.M.4.4.B.y.;z\.
....j+"rT'..i%r}.l.p.b...L9Nk....6...^c'.'p$.(..?.r.@....X}.~.n/..
0=r.=..C.....w[....K.It..q.$.].....\.@.s.Y.MB..G:$.F(-B  .U/|.
".S0.)....=d.......6.x.1@.....y.2.ho..%Y.....
. .........({+...x..Q..Xi.}}...$....`....     .2.        .k....8..2.,d.........
yW@d....m..e.          D.wH,.Or3.D
dGZ3k.A$...lZ....
n?.....-..g..`H5x%.uw5.@W%........B....]4|.....&lZ.b.S|..:..$X.:..T.UH...[.......r.
```

2 client pkt(s), 63 server pkt(s), 1 turn(s).

Entire conversation (13kB) Show data as ASCII Strumień 5

Picture 3 - Encrypted communication in two directions

Additionally, Firebase is encoding passwords with a modified version of the hash function. Firebase Auth will rehash the password the first time that account successfully logs in. Accounts downloaded from Firebase Authentication will only ever contain a password hash if one for this version of script is available, or contain an empty password hash otherwise.



```
hash_config {
  algorithm: SCRYPT,
  base64_signer_key: <...sensitive...>,
  base64_salt_separator: <...sensitive...>,
  rounds: 8,
  mem_cost: 14,
}
```

Picture 4 - Firebase encryption

8. Summary

Created application meets the requirements mentioned in this documentation. All of the implemented features work correctly and present a proper user experience. All of the personal data is encrypted while transmitting. There is a possibility to enrich the application with other functionalities in the future, for example audio messages or video calls.