|  |  |
| --- | --- |
| VIKsras  VILNIAUS KOLEGIJA / UNIVERSITY OF APPLIED SCIENCES  FACULTY OF ELECTRONICS AND INFORMATICS  SOFTWARE ENGINEERING COURSE | |
| **HYBRID MOBILE APPLICATIONS**  **FINAL PROJECT**  6531BX028 PI17E | |
|  | |
| STUDENT | DOMINYKAS JURKUS |
| (SIGNATURE)  06/01/2020 | |
| LECTURER | JUSTINAS ZAILSKAS |
| (SIGNATURE)  06/01/2020 | |

Content

[1. Introduction 3](#_Toc29198474)

[Goal 3](#_Toc29198475)

[Objectives 3](#_Toc29198476)

[Topic 3](#_Toc29198477)

[2. Formulation 4](#_Toc29198478)

[Functional requirements: 4](#_Toc29198479)

[Non-functional requirements: 4](#_Toc29198480)

[3. Analysis 4](#_Toc29198481)

[Use Case Diagram 4](#_Toc29198482)

[Requirements Traceability Matrix 5](#_Toc29198483)

[Use Case scenarios 5](#_Toc29198484)

[Databse logical design 7](#_Toc29198485)

[4. Software 8](#_Toc29198486)

[Development tools 8](#_Toc29198487)

[Main software files 8](#_Toc29198488)

[5. User Guide 8](#_Toc29198489)

# Introduction

## Goal

Create a mobile application using React Native or Flutter. The topic can be chosen by yourself.

## Objectives

The creation of a project consists of these subsystems:

1. Create Mobile Application (Api's are used, i.e gyro sensors).
2. Create Website (or other mobile app that communicates with each other).
3. Connected database (can be any SQL or NoSQL database).
4. Project report. The source code is stored in GitHub or GitLab.
5. Communication between the mobile application and the website can be done through WebSockets or WebServices

## Topic

In this final project I have decided to create a group chat application.

# Formulation

## Functional requirements:

1. The application should allow to post a message within a chat room.
2. The application should allow to register.
3. The application should allow to log in into an account.
4. The application should allow to create a chat room.
5. The application should display messages from different accounts.
6. The application should display message time.
7. The application should check if correct log in information is supplied.
8. The application should check if an account exist under an email address.

## Non-functional requirements:

1. The application shall be written in React Native or Flutter.
2. The application shall use Firebase for the database.
3. The application shall save messages within database.
4. The application shall use Firebase authentication for the authentication.
5. The application shall work on Android and iOS operating systems.

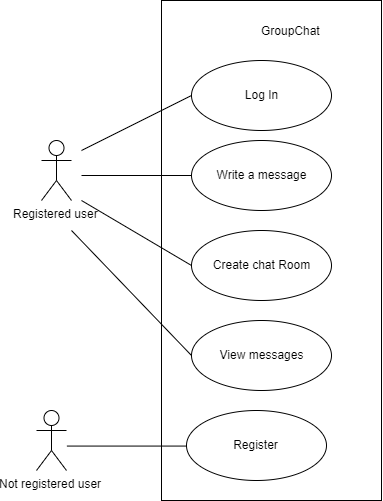
# Analysis

## Use Case Diagram

As we can see within the Use Case Diagram (Figure 2) there are two actors:

**Registered user** – is a user that has an account.

**Not registered user** – is a user that does not have an account.



**Figure 2 Use Case Diagram**

Registered users are able to log in, create a chat room, write a message and view messages from other users.

Not registered users are able to register.

## Requirements Traceability Matrix

In the table below we are able to see Requirements Traceability Matrix that allows us to check requirements against uses cases.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Use Cases | | | | | |
| UC1 | UC2 | UC3 | UC4 | UC5 |
| Requirements | R1 |  | x |  |  |  |
| R2 |  |  |  |  | x |
| R3 | x |  |  |  |  |
| R4 |  |  | x |  |  |
| R5 |  | x |  | x |  |
| R6 |  | x |  | x |  |
| R7 | x |  |  |  |  |
| R8 | x |  |  |  | x |

**Figure 3 Requirements Traceability Matrix**

## Use Case scenarios

In this part of the paper I am going to describe each use case with participating actors, preconditions, flow of events and postconditions.

|  |
| --- |
| **Log in** |
| **ID: UC1** |
| **Actors:**  Registered user |
| **Preconditions:**  1. The user has an account |
| **Flow of events:**  1. The Registered user inputs email address.  2. The Registered user inputs password.  3. The Registered user press Log in button.  4. The system checks if an account exists and correct password was written.  5. If an account exists and correct password was written, the system displays Room page. |
| **Postconditions:**  Registered user log in |

**Figure 4 Log In**

|  |
| --- |
| **Write a message** |
| **ID: UC2** |
| **Actors:**  Registered user |
| **Preconditions:**  1. The registered user is logged in.  2. There is at least one chat room created. |
| **Flow of events:**  1. The Registered user press on a room.  2. The system open the room and displays previous messages if any.  3. The Registered user writes a message.  4. The Registered user press “send”.  5. The system displays the message within the chat room. |
| **Postconditions:**  A message was created. |

**Figure 5 Write a message**

|  |
| --- |
| **Create a chat room** |
| **ID: UC3** |
| **Actors:**  Registered user |
| **Preconditions:**  1. The registered user is logged in. |
| **Flow of events:**  1. The Registered user writes a chat room name.  2. The Registered user press “create”.  3. The system saves the chat room within database and displays within a list. |
| **Postconditions:**  A chat room was created. |

**Figure 6 Create a chat room**

|  |
| --- |
| **View messages** |
| **ID: UC4** |
| **Actors:**  Registered user |
| **Preconditions:**  1. The registered user is logged in.  2. At least one chat room is created.  3. At least one message was written. |
| **Flow of events:**  1. The Registered user presses on a chat room  2. The system displays messages of that chat room. |
| **Postconditions:**  A message was viewed. |

**Figure 7 View messages**

|  |
| --- |
| **Register** |
| **ID: UC5** |
| **Actors:**  Not registered user |
| **Preconditions:** |
| **Flow of events:**  1. The Not Registered user presses on “Sign Up” button  2. The system displays “Sign UP” page.  3. The Not Registered user writes email address.  4. The Not Registered user writes password.  5. The Not Registered user press “Sign Up” button.  6. The system checks if an account with same email address does not exist.  7. If email address was not used, the system creates an account. |
| **Postconditions:**  Not registered user created an account. |

**Figure 8 Register**

From the use case scenarios and Requirements Traceability Matrix we can see that all functional requirements are covered.

## Databse logical design

For this project I have used NoSQL Firebase Realtime Database.

In the figure below (Figure 1) we can see that the database have two main paths – rooms and messages.

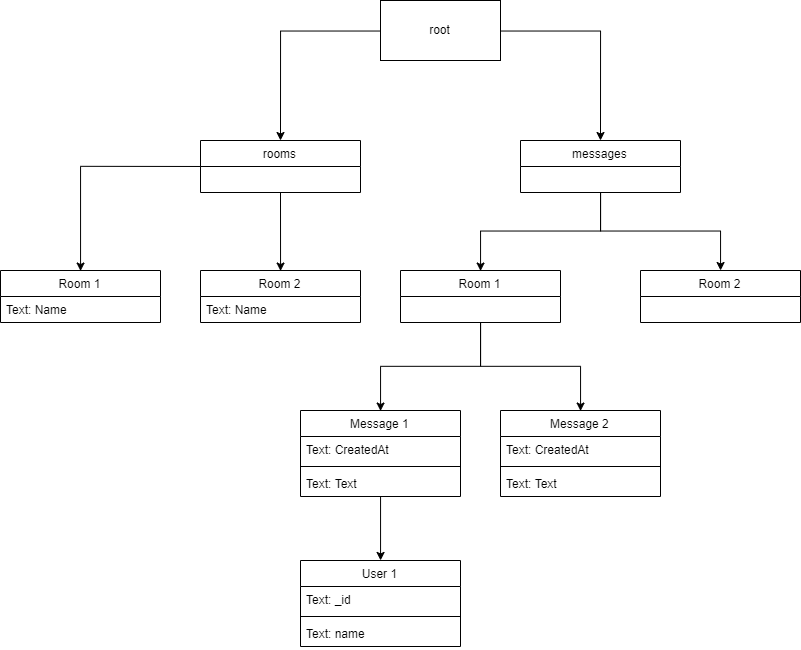
Rooms are used to store created chat rooms within the app. Each room entity has a property “name”.

Messages are used for storing user messages. Each message belongs to a room and has the user information (\_id and name) that wrote that message.

Moreover, each message contains:

CreatedAt – date when the message was created,

Text – text of the message.



**Figure 9 - Database Logical Design**

# Software

In this part of a paper I am going to write about development tools used, list main software files, and describe classes with class diagram.

## Development tools

In this project I have used React Native, React Native Gifted Chat, React Native Navigation, Visual Code, Firebase and Genymotion.

## Main software files

**firebaseConfig.js** – manages firebase initialization,

**Messages.js** – handles Gifted chat, it initializes and displays,

**Rooms.js** – handles rooms page and their creation,

**SignIn.js** – handles sign in,

**SingUp.js** – handles registering,

**Styles.js** – defines styles.

# User Guide

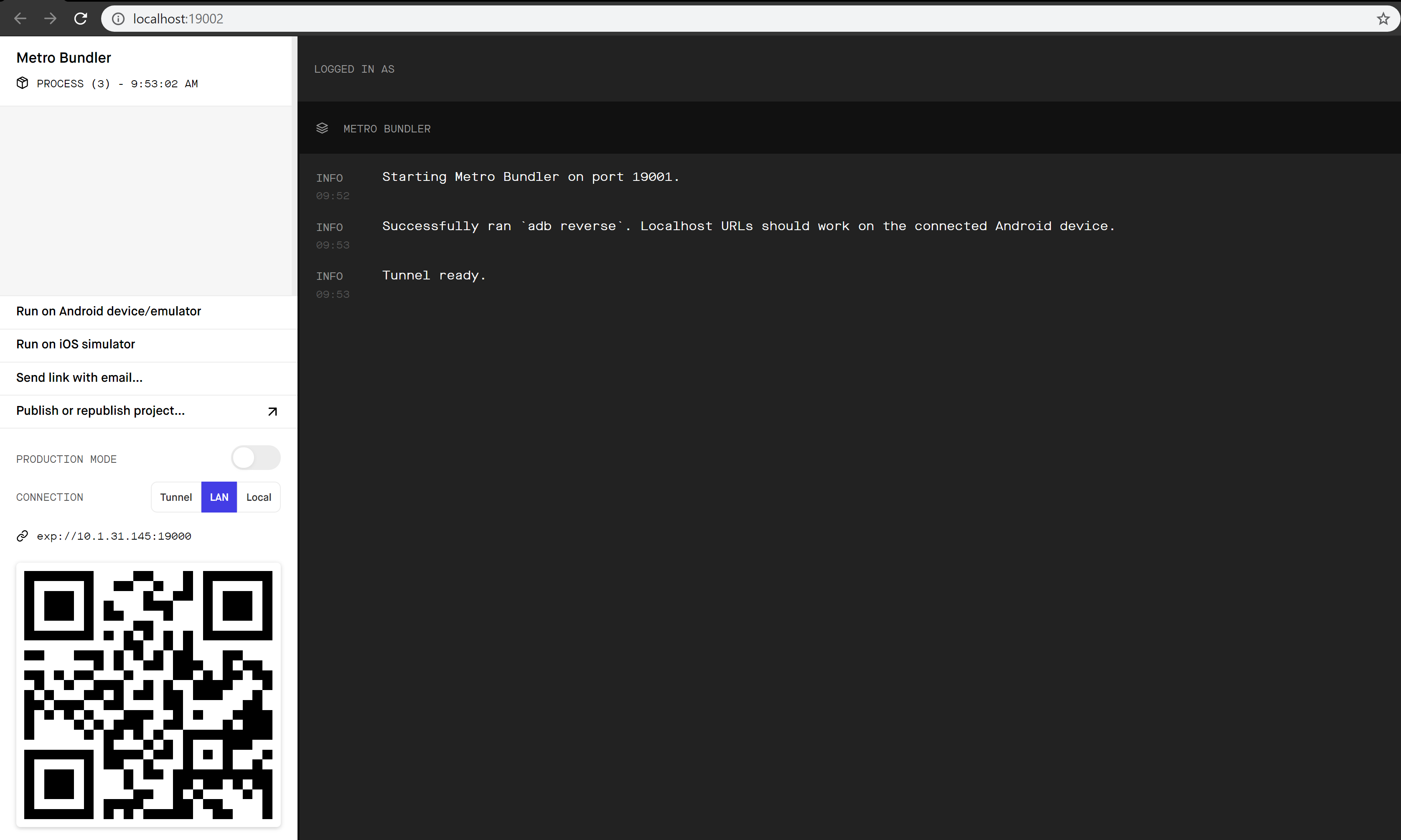
In order to use the application, the user needs to have Android or iOS device and email address.

The application is started with a command line from the project location in the file system.



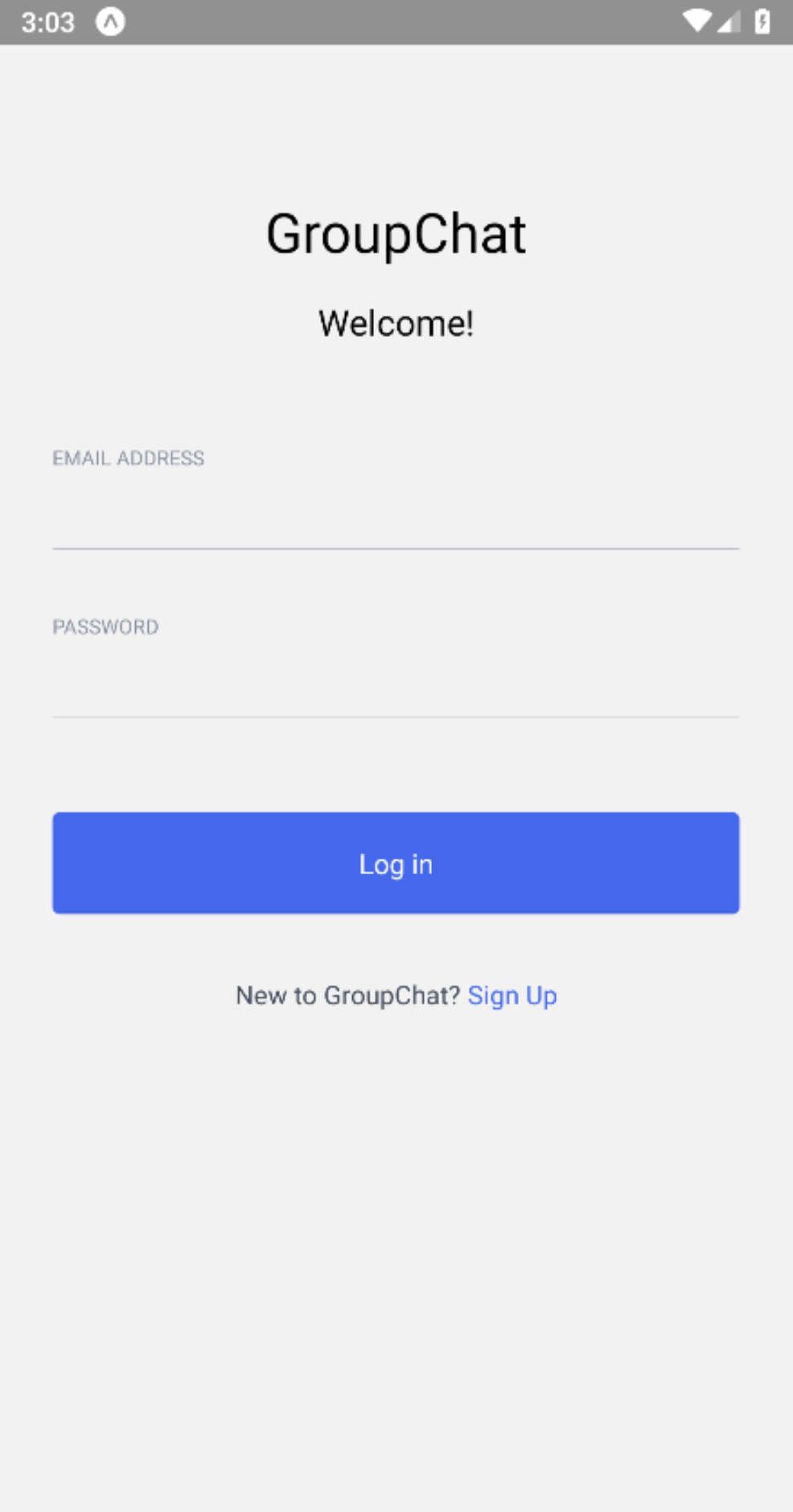
**Figure 10 console**

Once the project is started, a browser window will open.



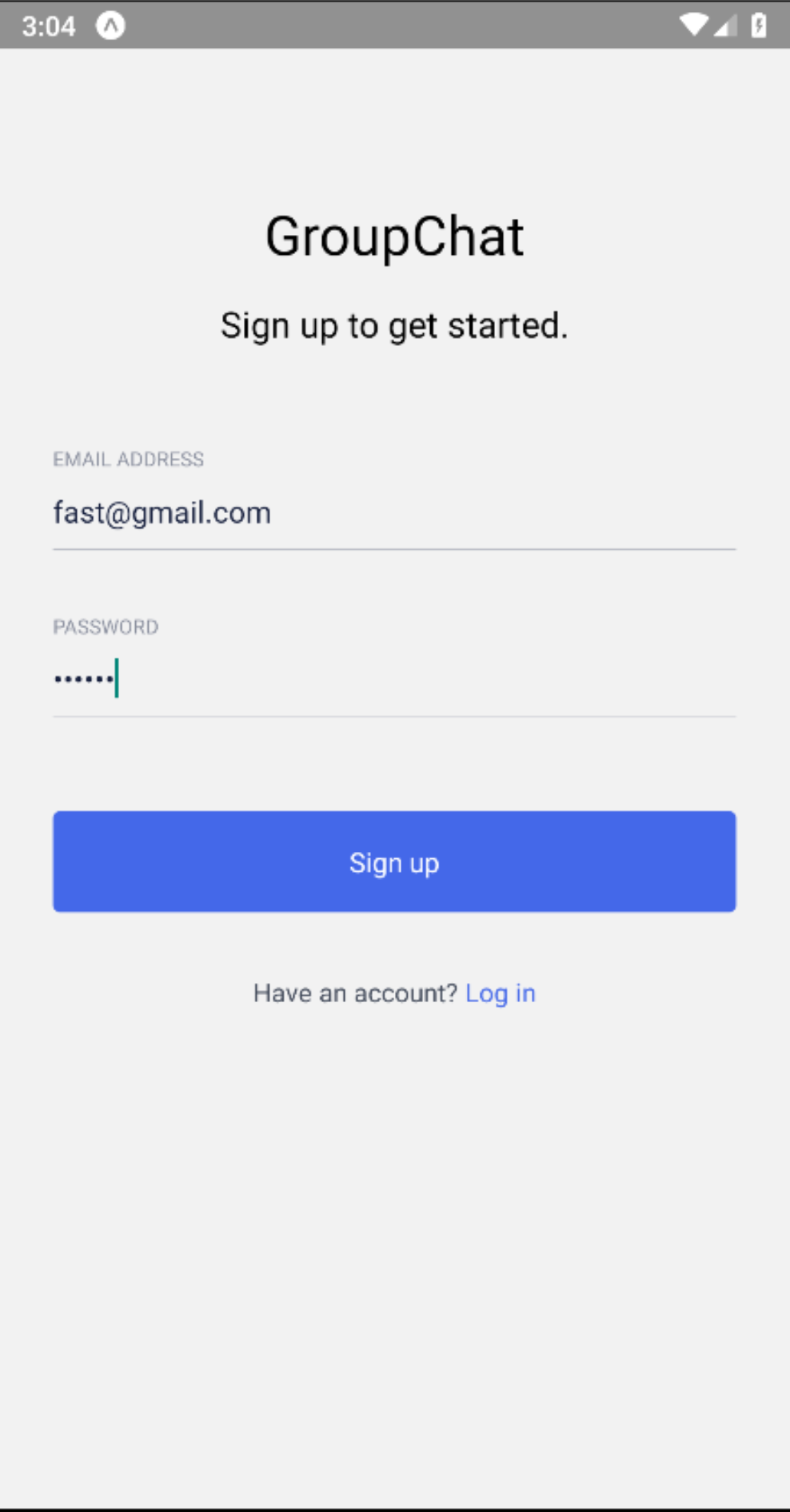
**Figure 11 web console**

Then the user can choose to use the expo app within their device or an emulator with expo app installed. For this manual, we are going to use the emulator. In order to run the emulator, press “Run on Android device/emulator”.



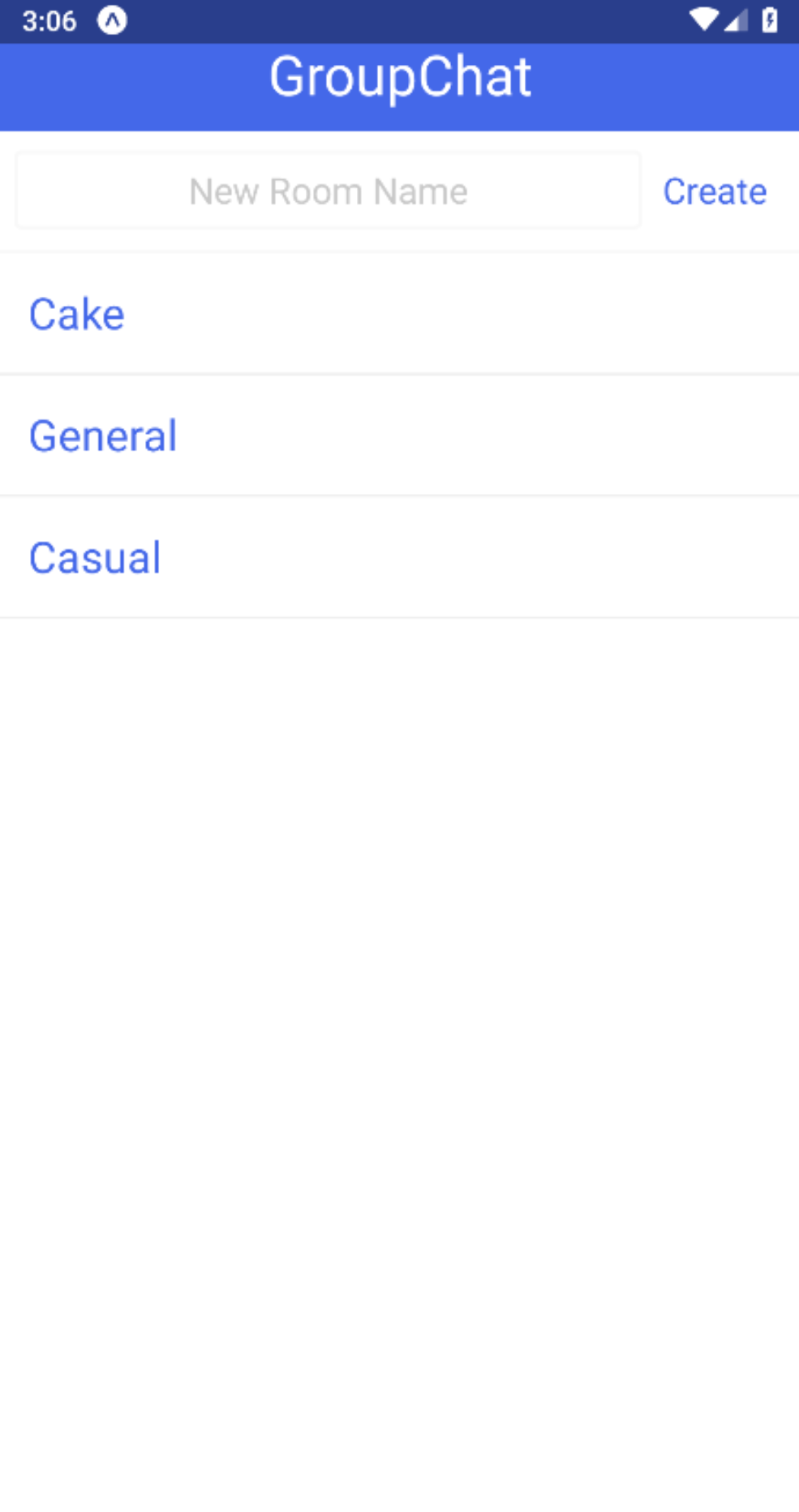
**Figure 11 Log in**

App will open and the Log in screen is presented. In order to register, we will need to press “Sing Up”.



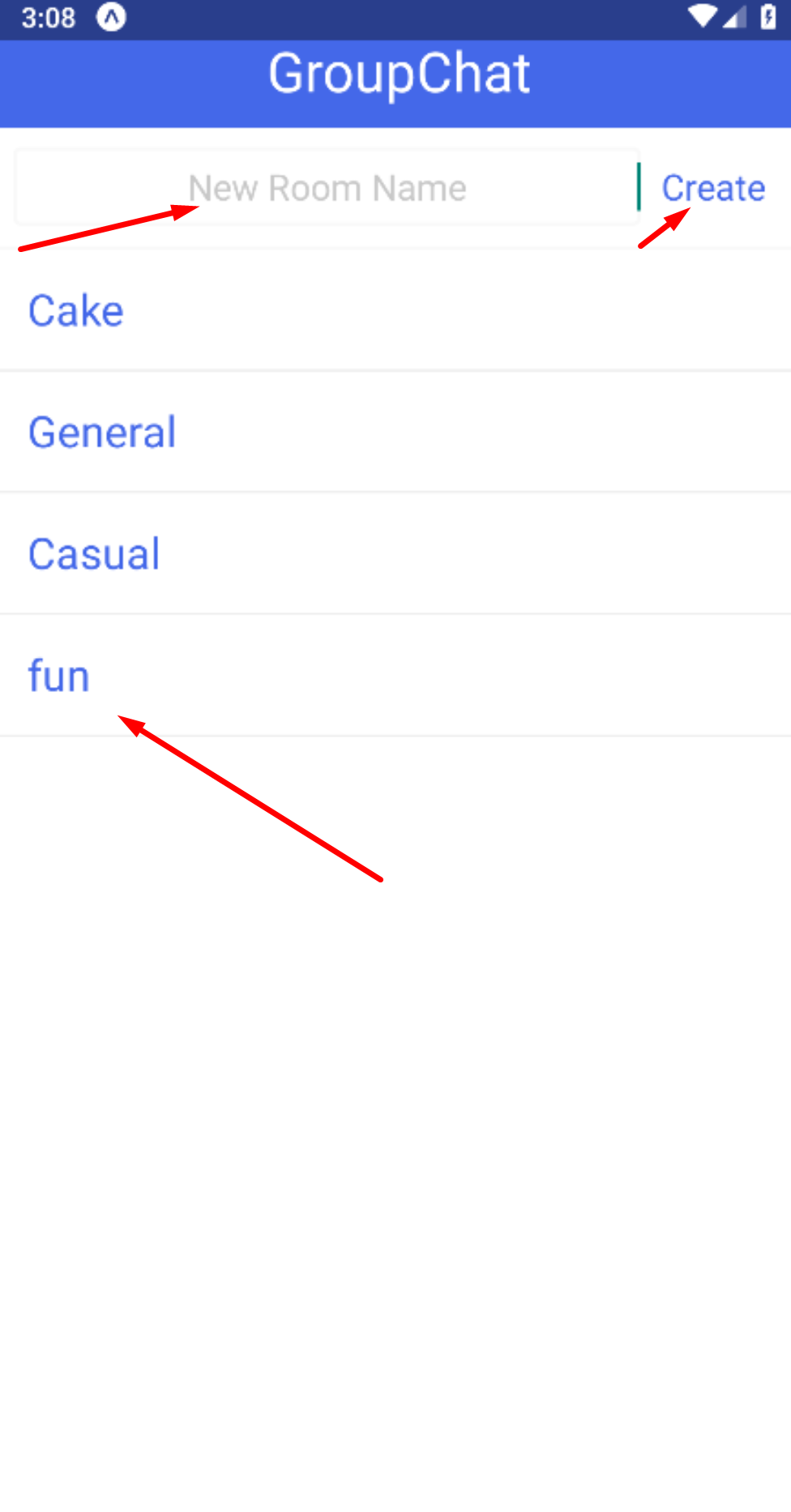
**Figure 12 register**

Here we will need to input email address and password. Once done, press “Sign up”.



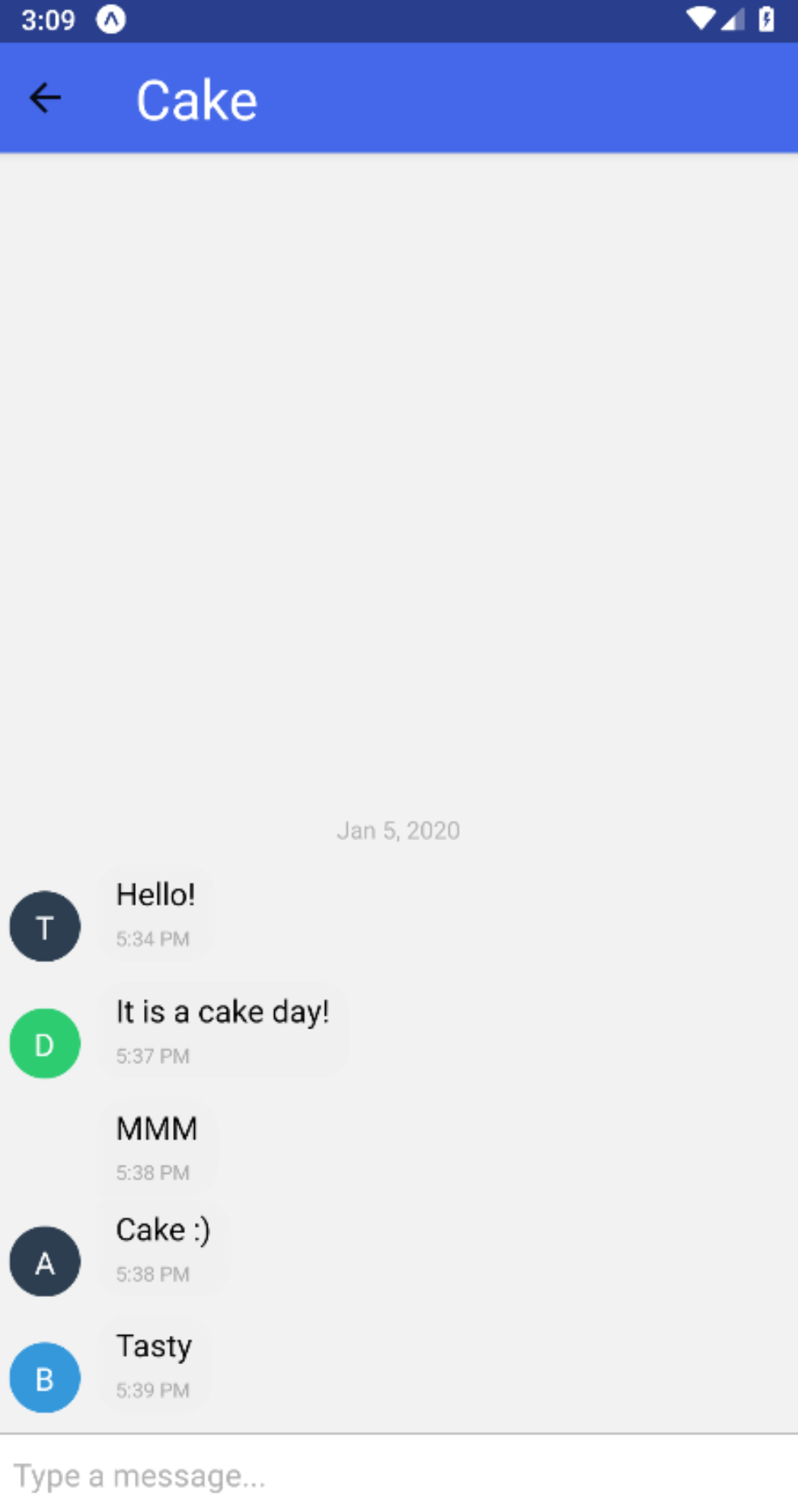
**Figure 13 Group Chat**

Group Chat window opens and we can see already created groups. We can press on a group or create one by typing in the “New Room Name” field and pressing “Create” button.



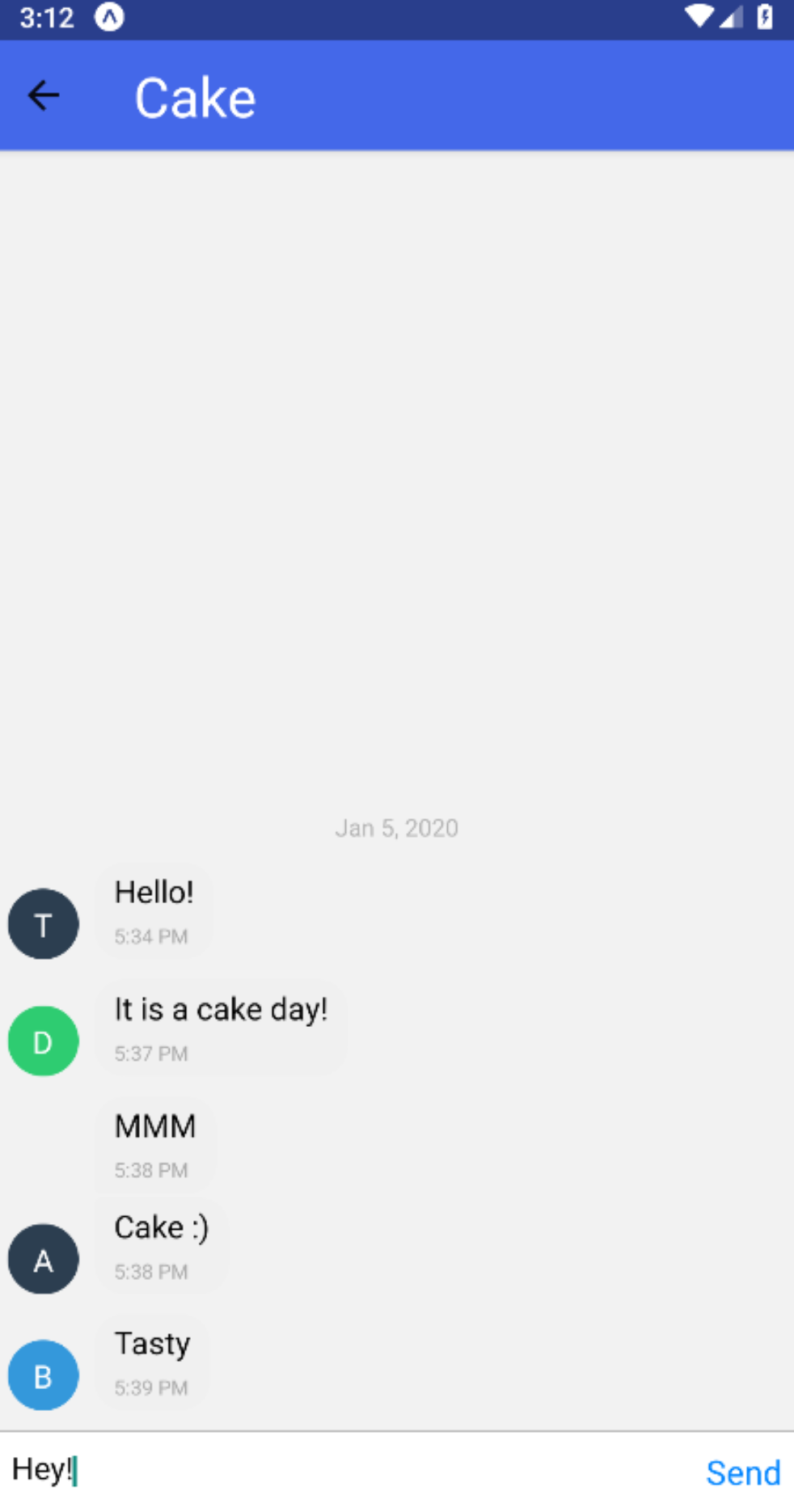
**Figure 14 create group**

Press on a group.



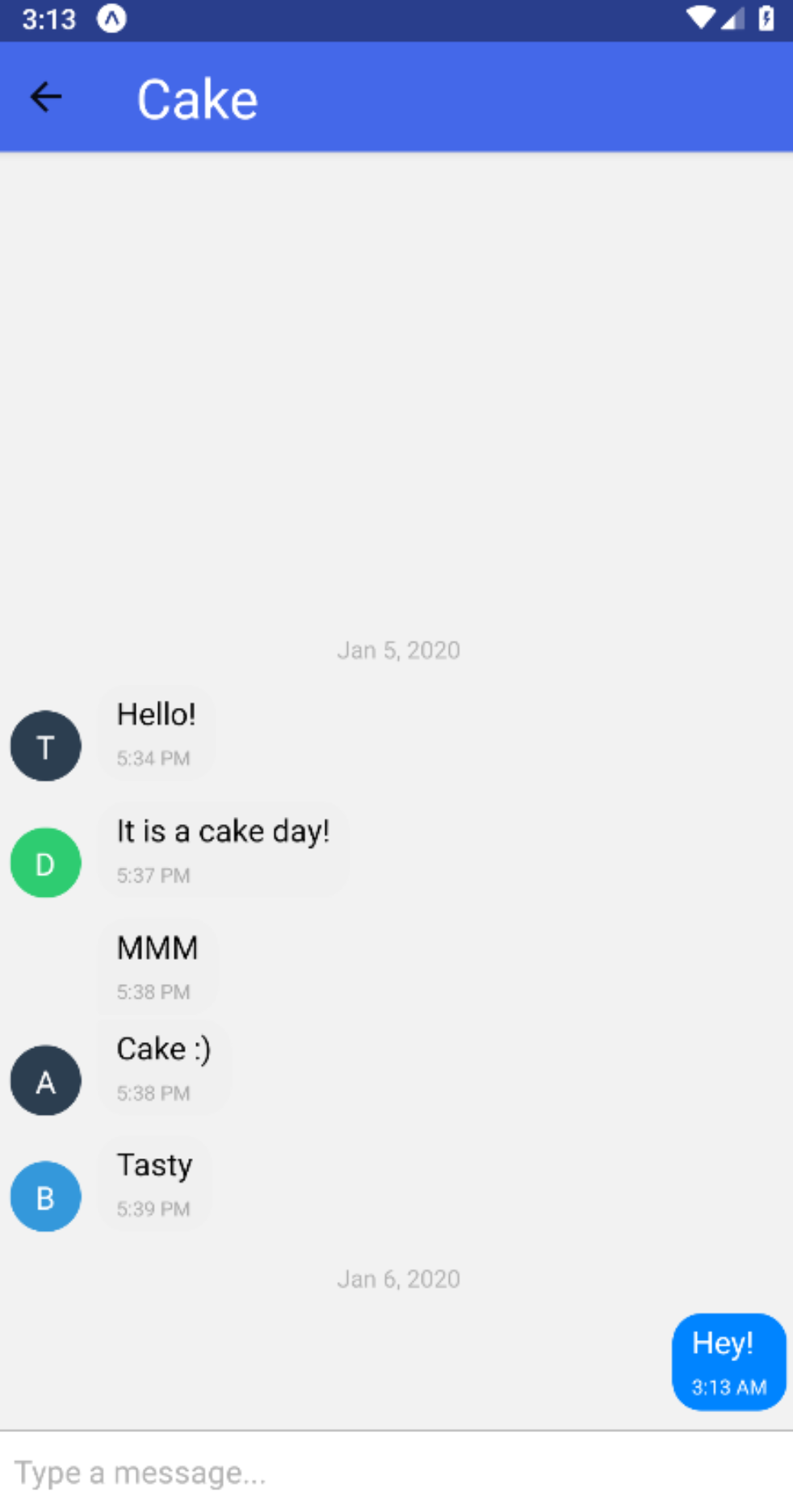
**Figure 15 messages**

Chat window opens and we can see previous messages if there any. In order to write a message we will need to type within the “Type a message…“ field and press “Send”.



**Figure 16 write a message**

And our message gets posted to the chat.



**Figure 17 posted message**

# Conclusion

In this project I have created a mobile group chat application with React Native and gifted chat UI library. I have defined functional and non-functional requirements, created use case diagram, Requirements Traceability Matrix and use cases. At the end I have create an user manual with steps and screenshots how to register, create a chat group and post a message.