# Internet Practical Course

**Team Golf**
Maximilian Erich Michel
Sherzad Alimzhan
Sven Neubauer
Wassim Ben Salem
September 22, 2020

TECHNISCHE UNIVERSITÄT DARMSTADT

Telecooperation Lab (TK)

## Contents

## 1 Motivation

The coronavirus pandemic has infected people in as many as 213 countries and territories. Its spread resulted in unforeseen challenges that reordered the world in dramatic ways. It is indeed first and foremost a public health crisis, but with nearly 30 million confirmed cases around the globe, it had multiple ramifications on other aspects of the daily life in 2020. For instance CoViD-19 not only affected social interactions but also left the economy counting costs and wondering what recovery could look like. However times of crisis have always shown us moments of opportunities and this global pandemic was no exception. It was accompanied with a wave of innovation especially in the field of technology. Multiple tech-startups around the world are putting leading edge innovations at the service of front-line workers.

The use of an contract tracing application could help to minimize the impact to society and economy by preventing larger lockdown.

## 2 Technical documentation

### 2.1 Used Technologies in Back-end

In our Back-end we used different technologies like:

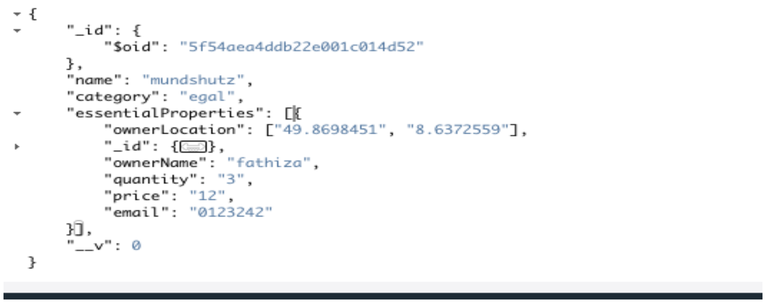
1. MongoDB: is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. (Version: last ) .
   See Figure 1
2. Mongoose: is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.
3. Node.js: Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser. we used the Version 14 , see Figure 2.
4. Express: is a web application framework for Node.js .It is designed for building web applications and APIs.
5. Docker : is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.
   In our Project we used docker to have images of mongodb and nodejs to avoid conflicts in versions and dependencies because we are working on different OS.
6. GitLab: is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration and deployment pipeline features, using an open-source license, developed by GitLab Inc. We used Gitlab to open issues which are mainly User stories , assign them to the developers and trace them .
   In every sprint we write user-Stories and push them to the 'Open' section in Gitlab. Every user in the Back-end or Front-end team takes one or multiple issues and works on them during the Sprint , for that we need to put them into the 'To Do' Field. The status of the issue needs to be continuously updated during the processing , so if the user starts the implementation he swipe it to the 'Doing' , and when he finishes he pushes it to the 'Closed' section . Moreover , we can discuss some details by writing some comments . See Figure 3

### 2.2 Backend Architecture

#### 2.2.1 Database Side

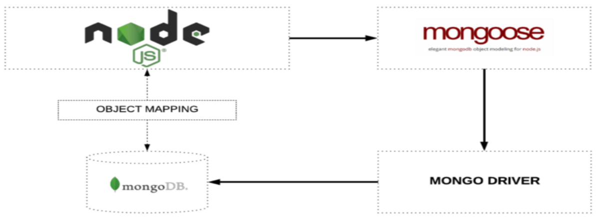
We separated our database into tables (Collections) to make that data easier to manage , fetch and to use. In our Database there are more than 10 Tables :

1. doctors : contains information about doctors and corona test centers ,Each input value contains attributes like address which is the address of the center , _id: which is assigned randomly by the DB to every object , properties which contain a list of attributes like city country postcode and name , and many other values.
   This table is manageable through the UI , in which we can add new values. See Figure 4



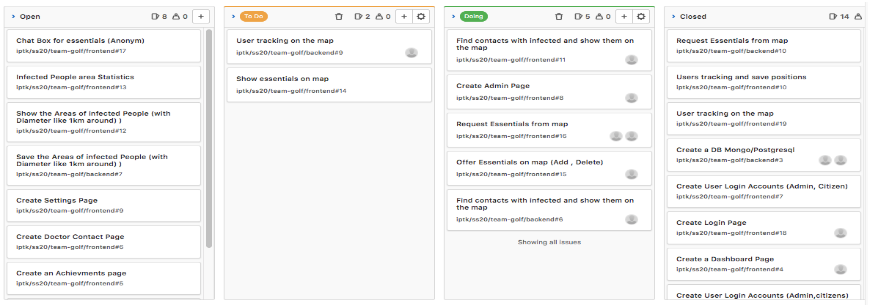
```
{
    "_id": {
        "$oid": "5f54aea4ddb22e001c014d52"
    },
    "name": "mundshutz",
    "category": "egal",
    "essentialProperties": [{
        "ownerLocation": ["49.8698451", "8.6372559"],
        "_id": {…},
        "ownerName": "fathiza",
        "quantity": "3",
        "price": "12",
        "email": "0123242"
    }],
    "__v": 0
}
```

Figure 1: MongoDB



Object Mapping between Node and MongoDB managed via Mongoose

Figure 2: Node.js



Open 8 0

- Chat Box for essentials (Anonym) iptk/ss20/team-golf/frontend#17
- Infected People area Statistics iptk/ss20/team-golf/frontend#13
- Show the Areas of infected People (with Diameter like 1km around) ) iptk/ss20/team-golf/frontend#12
- Save the Areas of infected People (with Diameter like 1km around) ) iptk/ss20/team-golf/backend#7
- Create Settings Page iptk/ss20/team-golf/frontend#9
- Create Doctor Contact Page iptk/ss20/team-golf/frontend#6
- Create an Achievments page iptk/ss20/team-golf/frontend#5

To Do 2 0

- User tracking on the map iptk/ss20/team-golf/backend#9
- Show essentials on map iptk/ss20/team-golf/frontend#14

Doing 5 0

- Find contacts with infected and show them on the map iptk/ss20/team-golf/frontend#11
- Create Admin Page iptk/ss20/team-golf/frontend#8
- Request Essentials from map iptk/ss20/team-golf/frontend#16
- Offer Essentials on map (Add , Delete) iptk/ss20/team-golf/frontend#15
- Find contacts with infected and show them on the map iptk/ss20/team-golf/backend#6

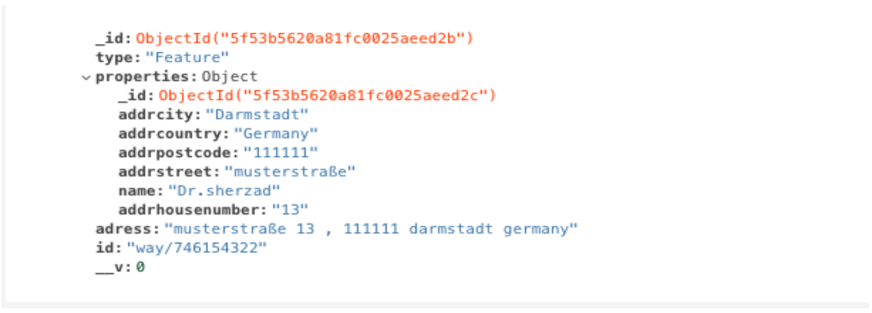
Showing all issues

Closed 14

- Request Essentials from map iptk/ss20/team-golf/backend#10
- Users tracking and save positions iptk/ss20/team-golf/frontend#10
- User tracking on the map iptk/ss20/team-golf/frontend#19
- Create a DB Mongo/Postgresql iptk/ss20/team-golf/backend#3
- Create User Login Accounts (Admin, Citizen) iptk/ss20/team-golf/frontend#7
- Create Login Page iptk/ss20/team-golf/frontend#18
- Create a Dashboard Page iptk/ss20/team-golf/frontend#4
- Create User Login Accounts (Admin,citizens)

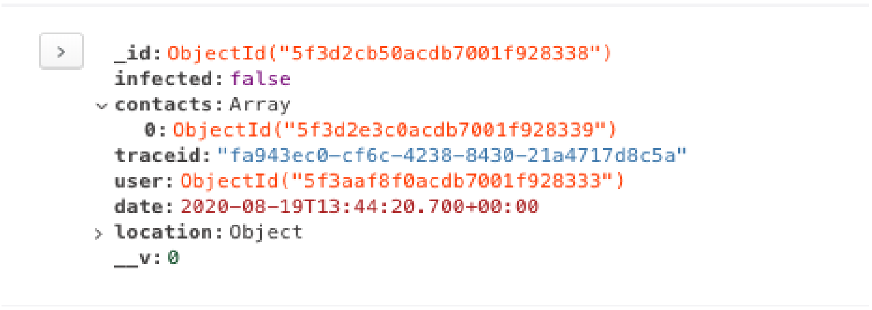
Figure 3: GitLab

2. GPS: contains the GPS information of users, like traces, contacts, infected or not, etc.See Figure 5
3. Users: contains information about the users that use our app . Each user is assigned a name, password, token, infection status, isAdmin to check if he is an admin or not.See Figure 6
4. Stores: is a table to save the stores location in our app, having a list of essentials assigned to every store . In fact every user can add through the UI a new store and save its location and push some essentials there . See Figure 7
5. Essentials: which are the Essentials requested by the users, and are added through the UI. See Figure 8 .Each essential value has:
    a) name: what is the name of the essential, e.g. toast;
    b) category: the category of the essential;
    c) owner Name : the username of the proprietor of the essential.
    d) quantity: quantity of the product;
    e) e-mail: the e-mail of the contact person



```
_id: ObjectId("5f53b5620a81fc0025aeed2b")
type: "Feature"
properties: Object
  _id: ObjectId("5f53b5620a81fc0025aeed2c")
  addrcity: "Darmstadt"
  addrcountry: "Germany"
  addrpostcode: "111111"
  addrstreet: "musterstraße"
  name: "Dr.sherzad"
  addrhousenumber: "13"
adress: "musterstraße 13 , 111111 darmstadt germany"
id: "way/746154322"
__v: 0
```

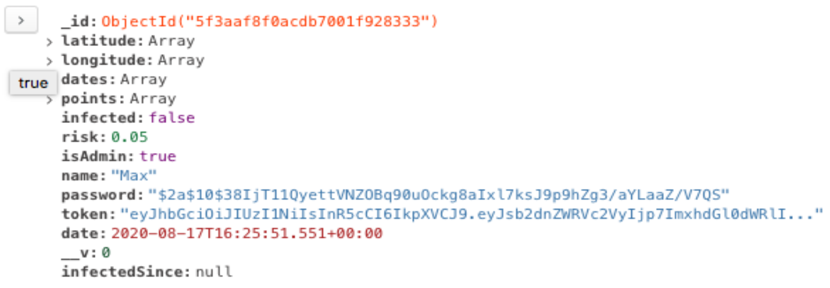
Figure 4: Docs



```
_id: ObjectId("5f3d2cb50acdb7001f928338")
infected: false
contacts: Array
  0: ObjectId("5f3d2e3c0acdb7001f928339")
traceid: "fa943ec0-cf6c-4238-8430-21a4717d8c5a"
user: ObjectId("5f3aaf8f0acdb7001f928333")
date: 2020-08-19T13:44:20.700+00:00
location: Object
__v: 0
```

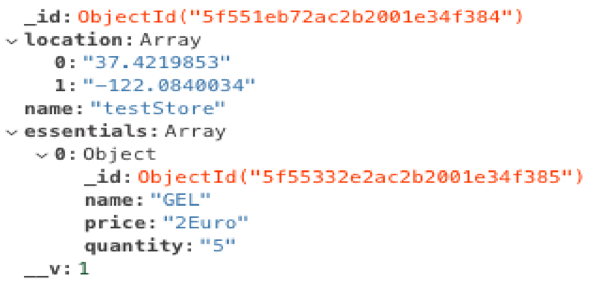
Figure 5: GPS

6. offerEssentials: which are the Essentials offered by the users , and have the same structure as the Essentials above.See Figure 9.
   a) name: what is the name of the essential, e.g. toast;
   b) category: the category of the essential.
   c) owner's name: the username of the proprietor of the essential.
   d) quantity: quantity of the product;
   e) e-mail: the e-mail of the contact person



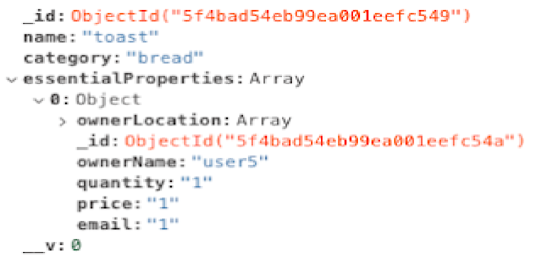
```
_id: ObjectId("5f3aaf8f0acdb7001f928333")
latitude: Array
longitude: Array
dates: Array
points: Array
infected: false
risk: 0.05
isAdmin: true
name: "Max"
password: "$2a$10$38IjT11QyettVNZOBq90uOckg8aIxl7ksJ9p9hZg3/aYLaaZ/V7QS"
token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRVc2VyIjp7ImxhdGl0dWRlI..."
date: 2020-08-17T16:25:51.551+00:00
__v: 0
infectedSince: null
```

Figure 6: User



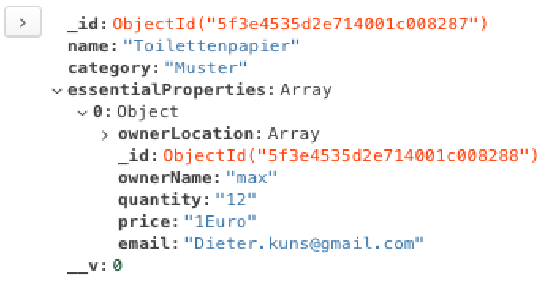
```
  _id: ObjectId("5f551eb72ac2b2001e34f384")
⌄ location: Array
    0: "37.4219853"
    1: "-122.0840034"
  name: "testStore"
⌄ essentials: Array
  ⌄ 0: Object
      _id: ObjectId("5f55332e2ac2b2001e34f385")
      name: "GEL"
      price: "2Euro"
      quantity: "5"
  __v: 1
```

Figure 7: Store



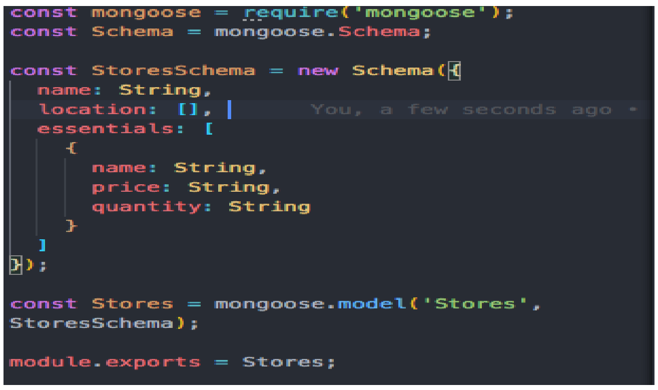
```
  _id: ObjectId("5f4bad54eb99ea001eefc549")
  name: "toast"
  category: "bread"
⌄ essentialProperties: Array
  ⌄ 0: Object
    › ownerLocation: Array
      _id: ObjectId("5f4bad54eb99ea001eefc54a")
      ownerName: "user5"
      quantity: "1"
      price: "1"
      email: "1"
  __v: 0
```

Figure 8: Request



```
  _id: ObjectId("5f3e4535d2e714001c008287")
  name: "Toilettenpapier"
  category: "Muster"
⌄ essentialProperties: Array
  ⌄ 0: Object
    › ownerLocation: Array
      _id: ObjectId("5f3e4535d2e714001c008288")
      ownerName: "max"
      quantity: "12"
      price: "1Euro"
      email: "Dieter.kuns@gmail.com"
  __v: 0
```

Figure 9: Offer



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const StoresSchema = new Schema({
  name: String,
  location: [], |        You, a few seconds ago •
  essentials: [
    {
      name: String,
      price: String,
      quantity: String
    }
  ]
});

const Stores = mongoose.model('Stores',
StoresSchema);

module.exports = Stores;
```

Figure 10: Store scheme

### 2.2.2 Server Side

Now let's dive into the architecture of our back-end component . We tried as much as we can to make it simpler to use and easier to extend . That is why we composed our work into some separated folders:

1. node_modules: which contains the npm modules installed. You can get it by executing: npm install
2. models: contains the scheme of our tables in the database. We have 7 schemes Doctors,Users,Stores, etc.
   For example : Figure 10
3. routes: which contains our main Work . In fact routes contains 3 main Files :
   a) gps.js: which is used to handle all requests concerning the GPS , like setting your own location , analysing traces , getLocation . . .
   b) users.js : which is our main Class . Through it we can execute many features:
      i. the authentication for signing in and signing up
      ii. adding , deleting and updating essentials
      iii. adding , deleting , updating ,and mark as Admin for users
      iv. adding , deleting and updating Doctors (Corona test centers)
      v. adding and fetching Stores
      vi. adding essentials to specific stores
   c) index.js: used to ensure the authentication and redirection
4. app.js file : which contains our main configuration , for example the connection options to the DB , redirections, express Sessions . . .
5. the rest of files are just configuration files , for example package.json / dockerFile . . . , which are not mainly components of our architecture , that's why we will not specify them at the moment.

## 2.3 Front-end

To develop the front-end part of the application we used Android Studio IDE with Java. To create layouts we used XML. For the map and location tracking we used Google Map from Google Play Services. To scan the barcode we used ZXing ("Zebra Crossing") barcode scanning library for Java, Android.
There are mainly two types of classes in the app: Activities and Fragments. For example MainActivity consists of 5 Fragments: DashboardFragment, MapsFragment, InfoFragment, AdminFragment and DoctorsFragment. All this Fragments are shown on the figures below in the section "User Manual". When the app is started, the LoginActivity will be called. From here on the user can either register or log-in. Then respectively MainActivity or RegisterActivity will be started.
To make requests to the database we also created DTO (Data transfer object) classes and placeholder interfaces. To communicate with the database we used Retrofit.

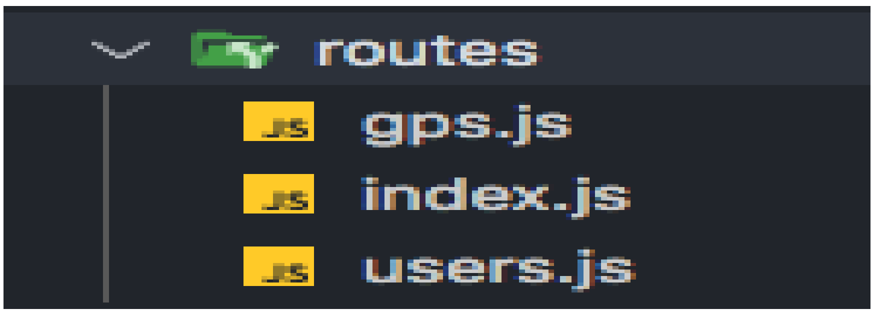
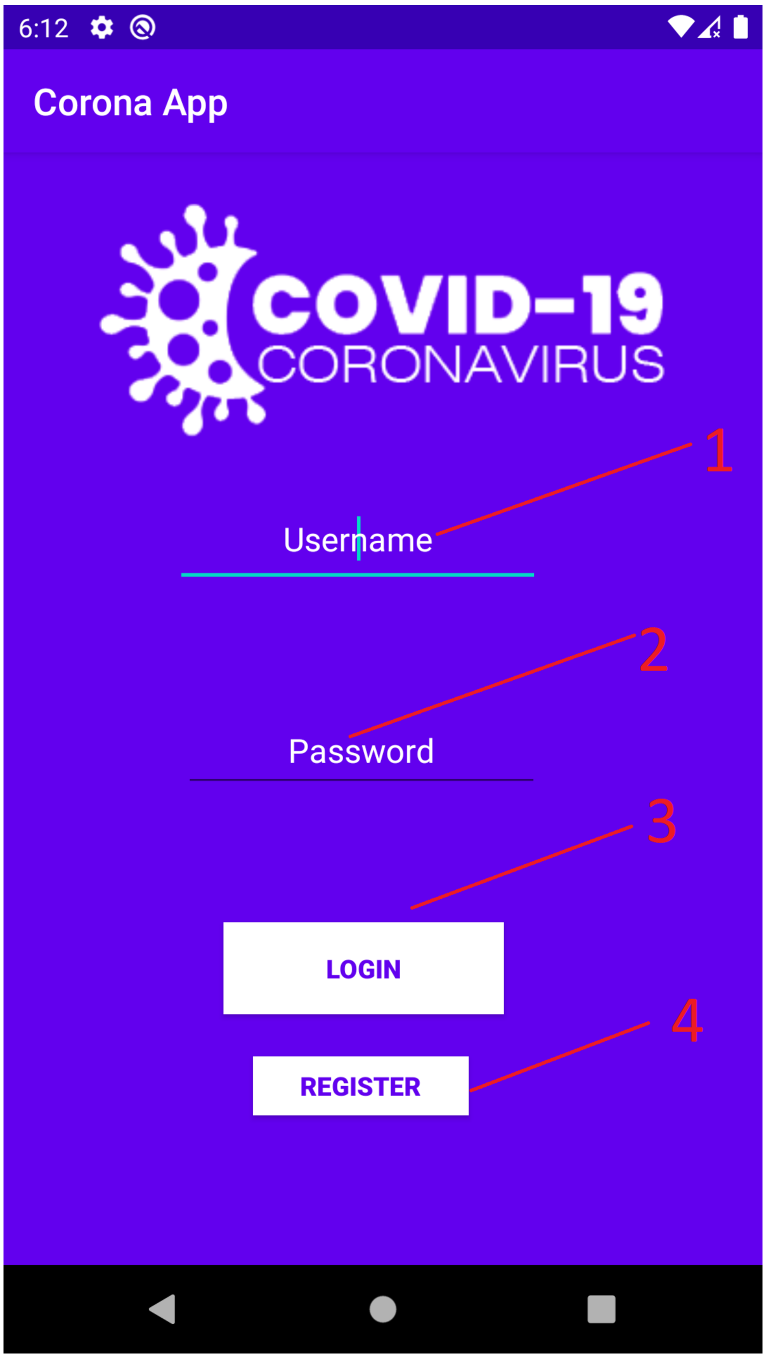


Figure 11: Files

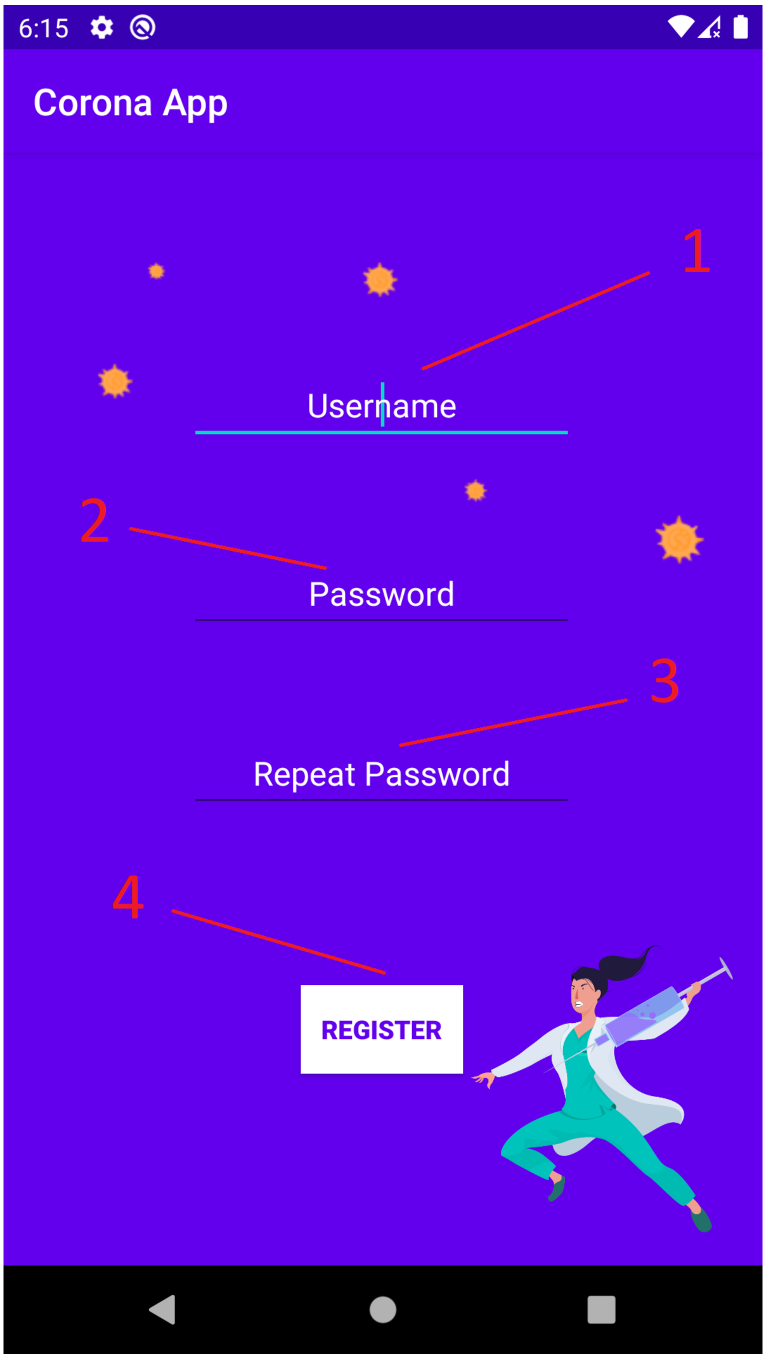
## 3 User manual

### 3.1 Login page



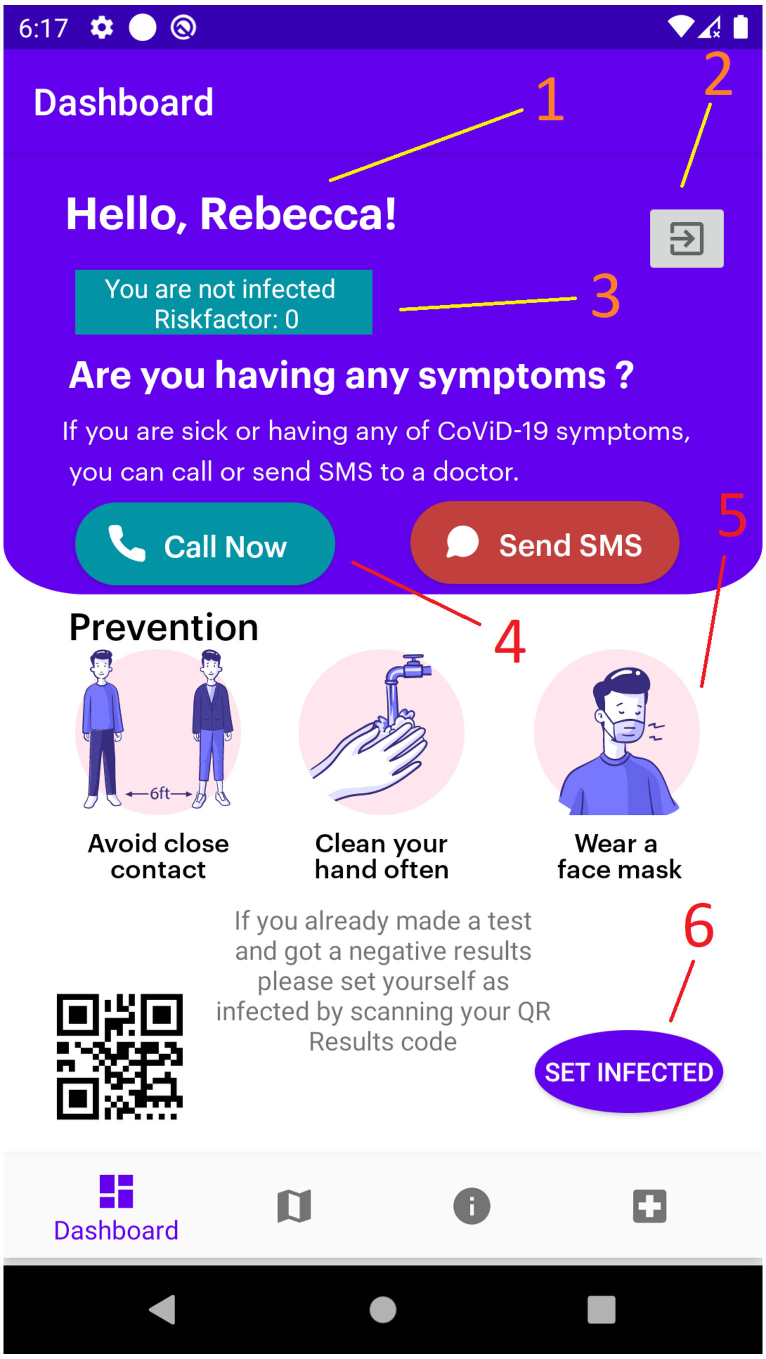
1. Insert username
2. Insert password
3. Login button, push it after the user wrote his username and password
4. Register button, user is redirected to the register page

## 3.2 Register page



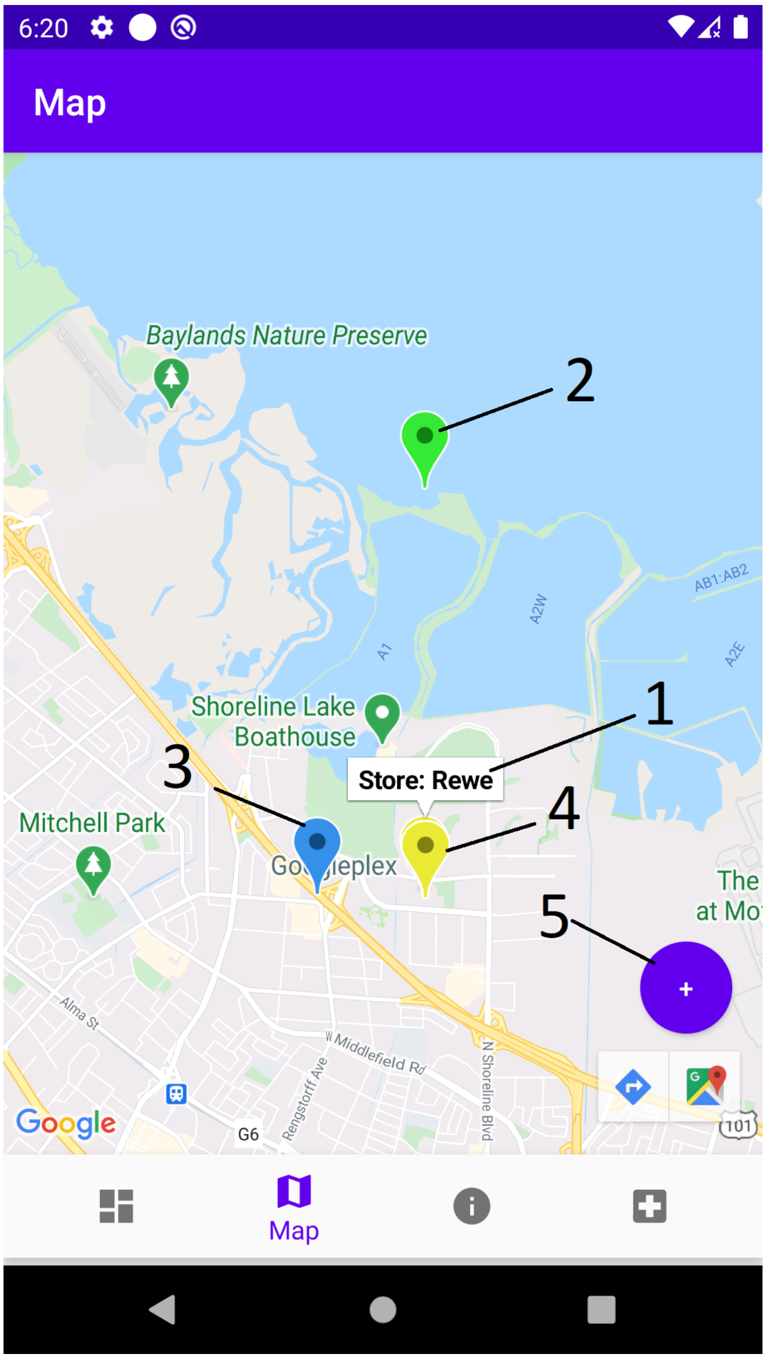
1. Insert username
2. Insert password
3. Insert password a second time
4. user will be registered if username does not exist

## 3.3 Dashboard


6:17
Dashboard
1
2
Hello, Rebecca!
You are not infected
Riskfactor: 0
3
Are you having any symptoms ?
If you are sick or having any of CoViD-19 symptoms, you can call or send SMS to a doctor.
Call Now
Send SMS
5
4
Prevention
6ft
Avoid close contact
Clean your hand often
Wear a face mask
6
If you already made a test and got a negative results please set yourself as infected by scanning your QR Results code
SET INFECTED
Dashboard

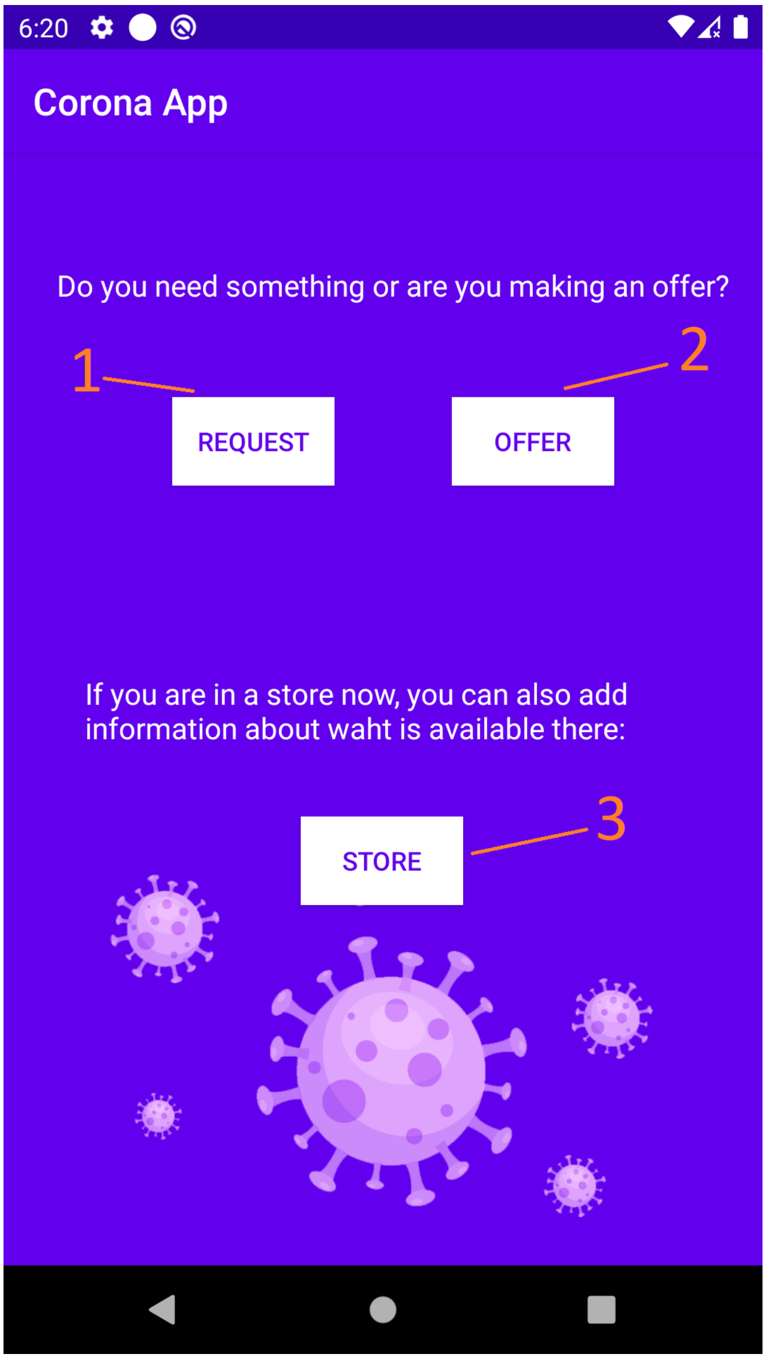
1. Username
2. log-out button
3. User status (infected/ not infected) and risk factor of the user between 0 and 1
4. Call the Corona help
5. Information about Corona
6. Opens the camera to mark the user infected

## 3.4 Map



1. Information about the marker, click on it to show all information
2. Marker offer(green)
3. Marker request(blue)
4. Marker shop(yellow)
5. Add new shops, offers and requests

## 3.5 Page for adding markers


6:20

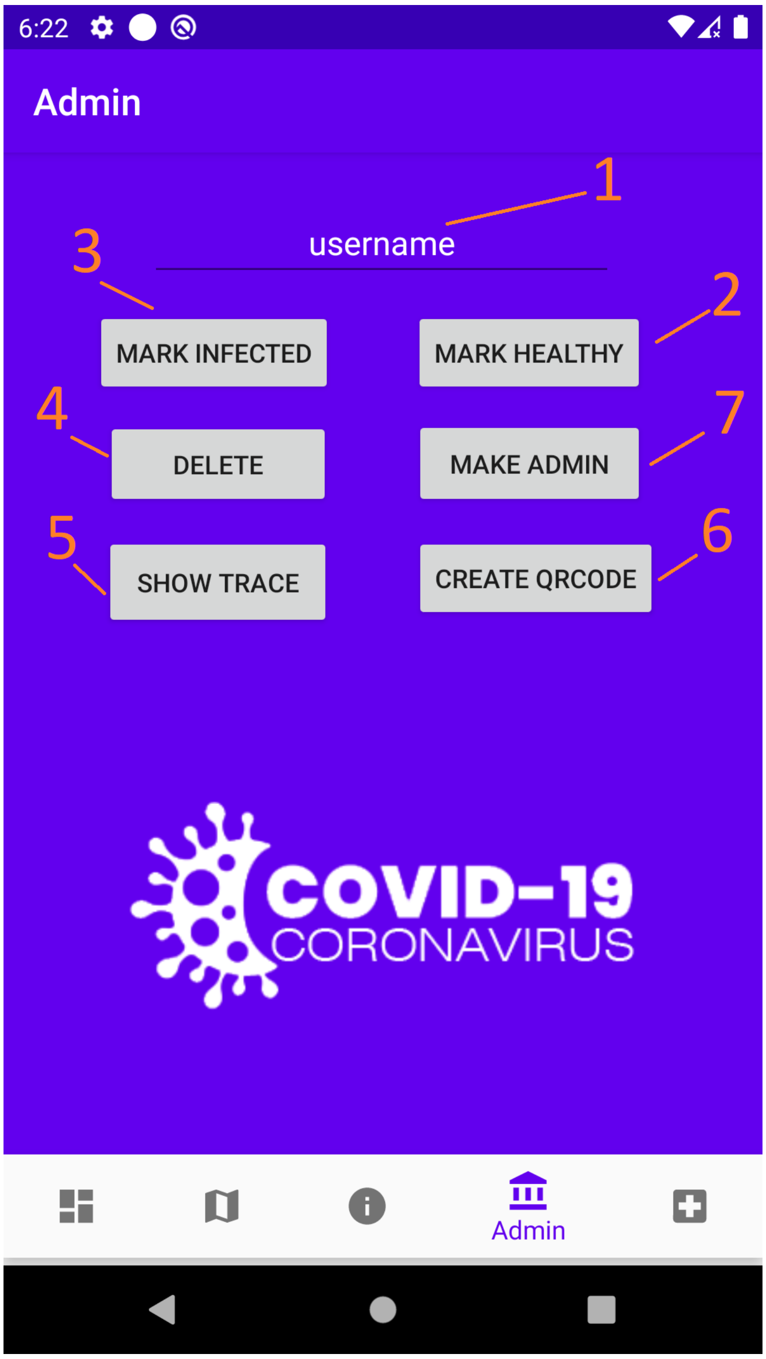
Corona App

Do you need something or are you making an offer?

If you are in a store now, you can also add information about waht is available there:

1. Add a request
2. Add an offer
3. Add a shop

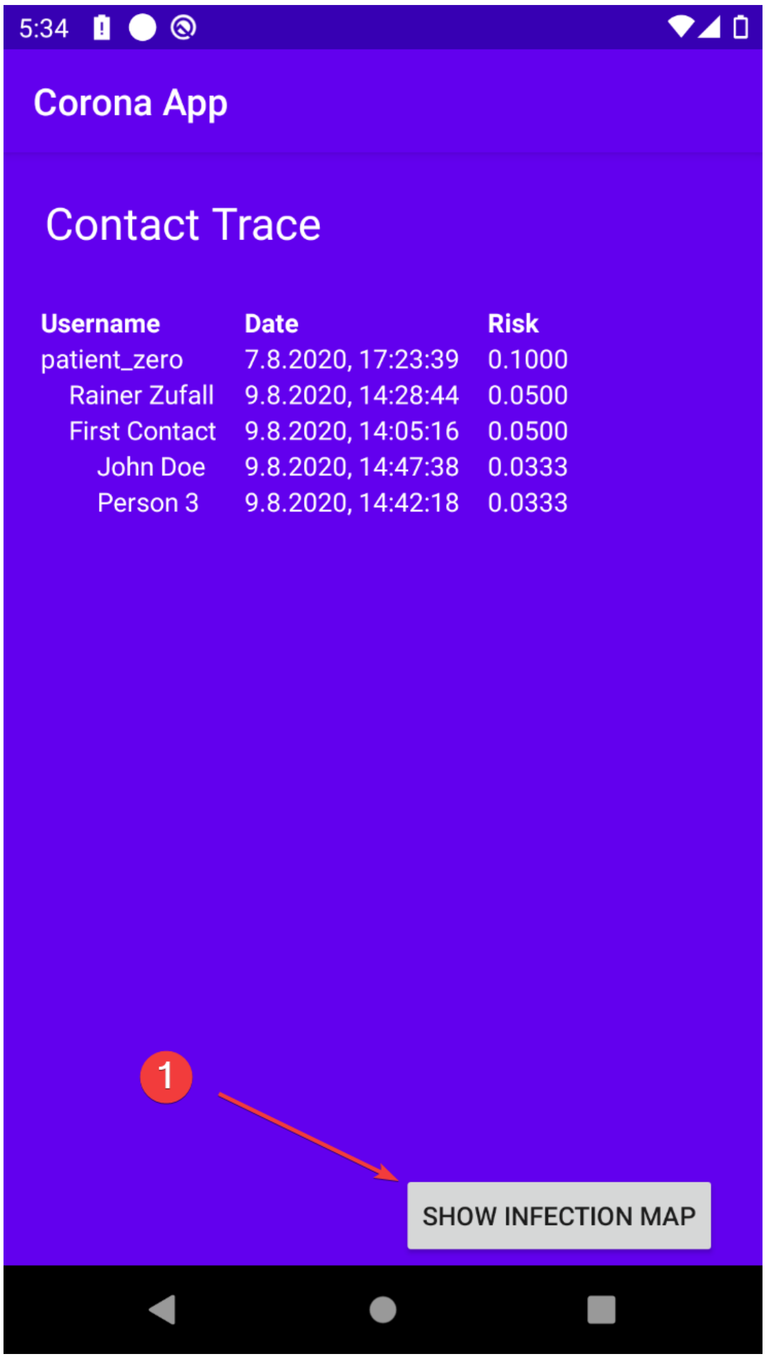
## 3.6 Admin page



1. Textfield, which is used for the buttons below
2. Mark the username in the textfield healthy
3. Mark the username in the textfield infected
4. Delete the user
5. Show the Trace of an infected user
6. Generate an QR code put of the text-field, which the user can read to mark himself infected
7. Make the user admin

## 3.7 Contact Tree

This page shows the tree of contacts from a chosen patient zero. The indent depth shows the number of contacts between the current user and patient zero. For each contact the date of contact is given. Also shown is the current risk factor, which is a Number between 0-1 with 0 meaning no risk and 1 most likely infected.
It is calculated by the number of contacts with a confirmed infected user either direct or indirect. With direct contacts adding 0.1 to the risk factor and indirect contacts adding to the risk factor depending on the number of steps.



5:34

Corona App

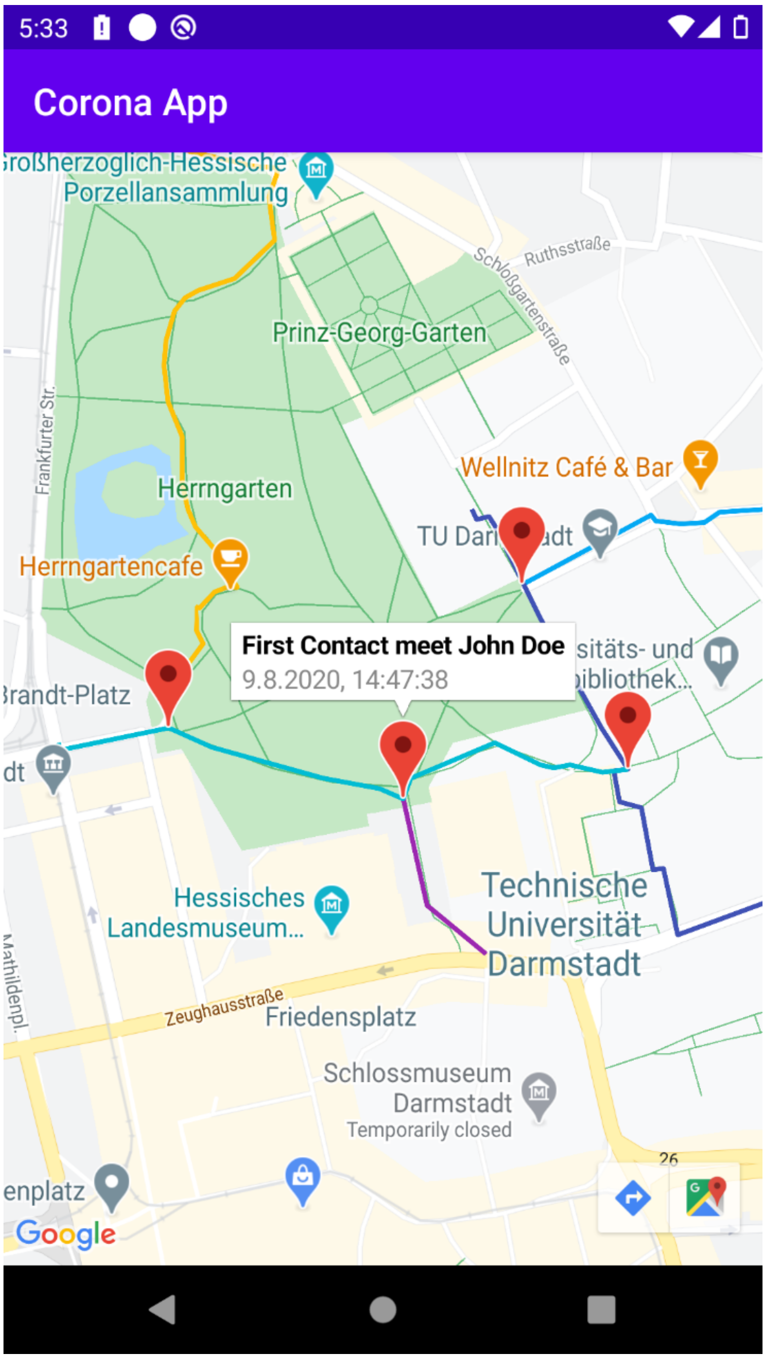
Contact Trace

| Username | Date | Risk |
|---|---|---|
| patient_zero | 7.8.2020, 17:23:39 | 0.1000 |
| Rainer Zufall | 9.8.2020, 14:28:44 | 0.0500 |
| First Contact | 9.8.2020, 14:05:16 | 0.0500 |
| John Doe | 9.8.2020, 14:47:38 | 0.0333 |
| Person 3 | 9.8.2020, 14:42:18 | 0.0333 |

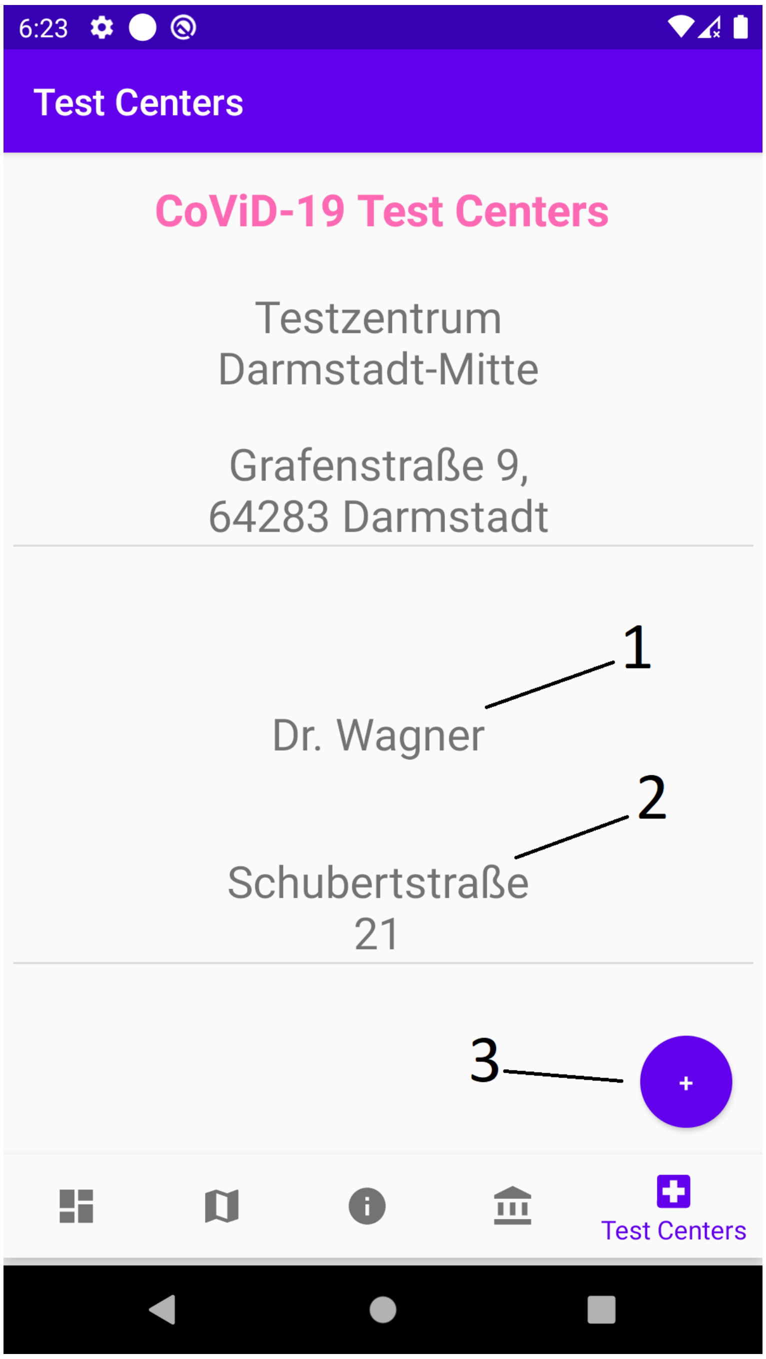
1. Show trace map of current tree

## 3.8 Trace Map

The trace map shows the movement profiles of users, it marks contact points and the trace beginning with that point. It starts with the patient zero and branches out with his contacts.



## 3.9 Doctor page



CoViD-19 Test Centers

Testzentrum Darmstadt-Mitte

Grafenstraße 9, 64283 Darmstadt

Dr. Wagner

Schubertstraße 21

1. Name of the doctor
2. Address of the doctor
3. Add a new doctor

# 4 Feature list

- Registration and authentication with pseudonym
- Dashboard with general guidance and current risk status

- Calculation of personal risk factor based on recent contacts
- QR-code reader to set yourself infected
- Background location tracking via GPS
- Upload of movement profile and search for contacts with other users
- Map view with current location trace
- Map view with current offers and request
- Form to add offers and requests with category and contact
- Form to create a store and report item availability
- Information page with current statistics
- Page of crowed sourced Covid-19 Test Centers with possibility to report missing ones
- Admin Page with Collection of features
  - Set Account infected/healthy
  - Delete account
  - Show contact tree of an infected person
  - Show trace map with traces and contact points of an infected person
  - QR-code generator on admin page
  - Mark account as admin
- Notify user when contact is detected

## 5 Summary

In this application we introduced key-features that may be needed during quarantine. Of course, we all hope that such situation will not happen again. Nevertheless we should be prepared, in case it repeats itself.

The application enables to reconstruct chains of infection to order those people into quarantine. This helps to limit the spread of the infection.

### 5.1 Future Work

Some features are only ruffed out and not polished to the state required in a general availability app these had to be redone.

One massive enhancement would be tracking contacts via the use of Bluetooth. This addition would make the tracing more reliable and would enable it in buildings.

### 5.2 Current Problems

As stated before one major disadvantage of using GPS to calculate contacts is the missing reception in buildings resulting in bad accuracy or even missing location information.

Another open issue is system performance as we only tested the system with some dummy users and could not test on a lager scale.