**Qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjk lzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjk lzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Exam Replacement Component  BLUETOOTH CONTACT TRACING APP    23-Jul-20  JACOB DIMBO |

Contents

[ABSTRACT 2](#_Toc46440530)

[INTRODUCTION 3](#_Toc46440531)

[LITTERATURE REVIEW 4](#_Toc46440532)

[CHALLENGE DEFINITION 7](#_Toc46440533)

[WIREFRAMES AND MOCK-UPS 8](#_Toc46440534)

[DESIGN & ARCHITECTURE 12](#_Toc46440535)

[MVC ARCHITECTURE 21](#_Toc46440536)

[SYSTEM INVESTIGATION AND ANALYSIS 13](#_Toc46440537)

[FUNCTIONAL REQUIREMENTS 13](#_Toc46440538)

[NON-FUNCTIONAL REQUIREMENTS 13](#_Toc46440539)

[SYSTEM DESIGN 14](#_Toc46440540)

[DATABASE DEGIN 14](#_Toc46440541)

[HOW THE APPLICATION WILL WORK 17](#_Toc46440542)

[IMPLEMENTATION DIRECTIONS 19](#_Toc46440543)

[REFERENCES 23](#_Toc46440544)

# 

# ABSTRACT

Contact-tracing apps to fight the COVID-19 pandemic have increasingly been mentioned as useful tools to accompany and contribute to a return to normal life despite many ethical and legal questions they raise. The pressure exerted by business circles and lobbies to return to normality in order to save the economy has been intense. What started as a public health crisis morphed into an economic crisis and we are now faced with a ‘trick-or-treat’ choice: accept to ‘pay the price’ and use invasive contact tracing apps, and by so doing facilitate a gradual reopening of businesses and other economic activities, or fight for privacy and delay the return to normality.

# INTRODUCTION

Digital contact tracing applications are software that can be installed on a user’s personal device, such as a smartphone, to notify the user when they come into contact with person(s) infected with Covid-19. If contact is made, the user of the application is notified about the potential contact and provided with further instructions (e.g., to self-isolate). Some applications additionally notify the relevant public health authority.

A recent rapid evidence review from the Ada Lovelace Institute highlights a number of issues that may impact on the effectiveness of digital contact tracing, including whether the use of the application is mandatory or voluntary, the accuracy of the test in terms of detecting contact, how data is collected and accessed and by whom, and how data from the application will be used to inform appropriate actions.

# LITTERATURE REVIEW

As the covid-19 pandemic rages, technologists everywhere have been rushing to build apps, services, and systems for contact tracing: identifying and notifying all those who come in contact with a carrier. Some are lightweight and temporary, while others are pervasive and invasive: China’s system, for example, sucks up data including citizens’ identity, location, and even online payment history so that local police can watch for those who break quarantine rules. Some services are being produced locally by small groups of coders, while others are vast, global operations. Apple and Google are mobilizing huge teams to build their upcoming systems that notify people of potential exposure, which could be used by hundreds of millions of people almost immediately.

When we began comparing apps around the world, we realized there was no central repository of information; just incomplete, constantly changing data spread across a wide range of sources. Nor was there a single, standard approach being taken by developers and policymakers: citizens of different countries were seeing radically different levels of surveillance and transparency. So to help monitor this fast-evolving situation, we’re gathering the information into a single place for the first time with our Covid Tracing Tracker—a database to capture details of every significant automated contact tracing effort around the world. We’ve been working with a range of experts to understand what we need to look at, pulling sources including government documents, announcements, and media reports, as well as talking directly to those who are making these apps to understand the technologies and policies involved. So far we have documented 25 individual, significant automated contact tracing efforts globally, including details on what they are, how they work, and what policies and processes have been put in place around them. At the most basic level, we are compiling a list of automated contact tracing apps that are backed by national governments. These are apps designed to automatically tell users or public health officials whether somebody has potentially been exposed to covid-19; it’s what is generally known as “exposure notification (O’Neil et al 2020).

The efficacy of such apps has yet to be proved. Modeling suggests they can help to slow the spread of the virus — but only if enough of the population uses them. A preprint from a group at the University of Oxford, UK, suggests that a take-up threshold of 60% of the population can bring an outbreak under control1. The apps have also raised privacy concerns, because some need to store user data on central servers if people are to be identified and tracked. And even proponents of the apps say that, to be most effective, they still require human contact tracers in the loop to conduct follow-up interviews.

According to Zhang et al (2020), Experts suggest that at least 60% of the public would need to use these apps for them to be effective at limiting the spread of COVID-19. Yet fears that these apps would violate users' privacy by expanding governments' and tech companies' surveillance capacity may limit adoption. They studied Americans' attitudes toward smartphone contact tracing apps and public health surveillance policies using a large, nationally representative survey of U.S. adults (N=2,612). They found widespread reluctance among the public: support for contact tracing apps is lower than for expanding traditional contact tracing or introducing new measures like temperature checks and centralized quarantine. Using a conjoint analysis experiment embedded in the survey, they found that privacy-preserving features, including non-location tracking and decentralized data storage, increases the public's acceptance of contact tracing apps. Within the population, those with pre-existing health conditions or who know someone who had been COVID-19 positive were more likely to support the tool, suggesting that support will grow as cases increase. Despite significant partisan splits on most issues, Democrats and Republicans converge on levels of support for contact tracing apps, suggesting that bipartisan elite cues could work to augment support. Overall, we found sizable amounts of concern about privacy and misunderstanding about the technology used. Public education campaigns are much needed before states deploy contact tracing apps (Zhang et al 2020).

TraceTogether is the first national deployment of a Bluetooth-based contact tracing system in the world. It was developed by Singapore’s Government Technology Agency and the Ministry of Health to help the country better respond to epidemics. Following its release, more than 50 governments have expressed interest in adopting or adapting TraceTogether for their countries. Responding to this interest, they released an overview of BlueTrace, the privacy-preserving protocol that underpins TraceTogether, as well as OpenTrace, a reference implementation. OpenTrace comprises the source code for an iOS app, an Android app, a cloud-based backend, and baseline signal strength calibration data. That was made available to the open source community at github.com/opentrace-community on 9 April 2020 (Bay et al 2020). BlueTrace is a protocol for logging Bluetooth encounters between participating devices to facilitate contact tracing, while protecting the users’ personal data and privacy. When two participating devices encounter each other, they exchange non-personally identifiable messages that contain temporary identifiers. The identifiers rotate frequently to prevent third parties from tracking users. The user’s encounter history is stored locally on their user’s device; none of this data can be directly accessed by the health authority. If a user is infected or is the subject of contact tracing, they will be asked to share their encounter history with the relevant health authority with the use of a PIN. (A verification code may optionally be provided, to authenticate the health authority official’s request.) Only the health authority has the ability to decrypt the shared encounter history to obtain and use personally-identifiable information to filter for close contacts and contact potentially infected users. BlueTrace is designed to supplement manual contact tracing by addressing its key limitation: an infected person can only report contacts they are acquainted with and remember having met. BlueTrace could also allow for contact tracing to be more scalable and less resource-intensive. BlueTrace also allows a federated network of credentialed health authorities to each maintain distinct user bases, while allowing for contact tracing between users from different health authority jurisdictions (see Section 7: Federation and Interoperability).

# CHALLENGE DEFINITION

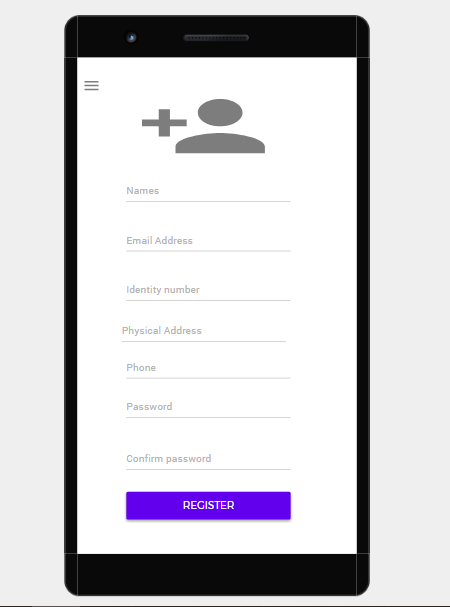
|  |  |
| --- | --- |
| CHALLENGE | Bluetooth contact tracing app |
| CATEGORY | HEALTH |
| DESCRIPTION | A mobile application designed to improve the processes of direct contact tracing by showing whether you might have been exposed to the noble coronavirus. |
| BACKGORUND | Contact tracing breaks the chains of transmission of an infectious disease and is thus an essential public health tool for controlling infectious disease outbreaks. Ever since the outbreak of corona virus contact tracing has become one of the fundamental activities that involve working with a patient (symptomatic and asymptomatic) who has been diagnosed with an infectious disease to identify and provide support to people (contacts) who may have been infected through exposure to the patient.  Contact tracing is conducted for [close contacts](https://www.cdc.gov/coronavirus/2019-ncov/php/contact-tracing/contact-tracing-plan/appendix.html#contact) (any individual within 6 feet of an infected person for at least 15 minutes) of laboratory-confirmed or probable COVID-19 patients. Since most of the countries were unprepared for the outbreak of COVID-19, this processes is manually done which makes the whole process long and hence the spread of the virus quickly multiplies.  The introduction of technology can speed up the slower manual processes and hence the reduced virus spreading rates. |
| SKILLS REQUIRED | Mobile application development  Databases |
| HARDWARE/SOFTWARE NEEDS | Android studio  Windows Pc I5 and above |
| Stakeholder | Public |

# WIREFRAMES AND MOCK-UPS

This section presents a simple overview of the mobile application interfaces and the major functionalities to be implemented.

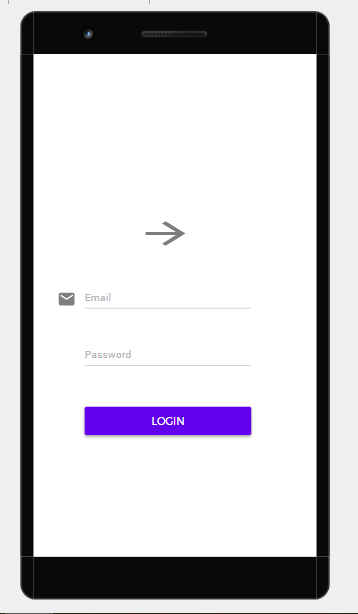
REGISTER

The register allows users to create account. Users are required to provide their basic details that will allow the admins to easily identify each user.

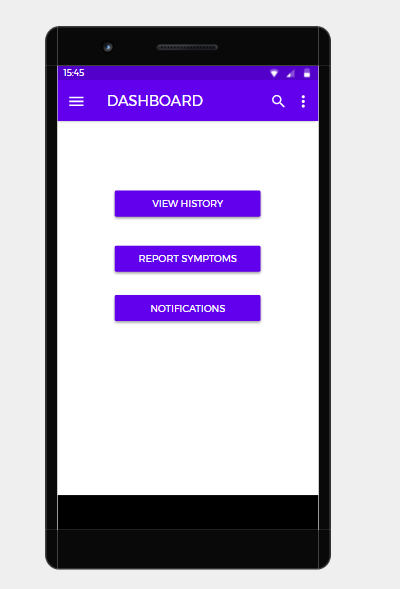


LOGIN

After successful registration a user can now login .Logging in avails the user to the major and fundamental operations.



DASHBOARD



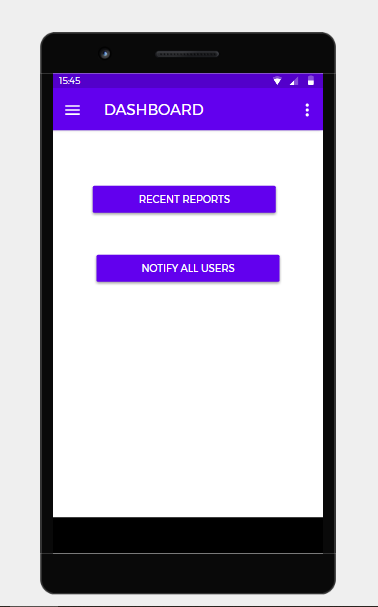
VIEW HISTORY

This operation allows a user to view their contact history, providing a list of the entire device id’s a user’s phone has discovered, location and time. We display the device id’s to protect other user’s privacy.

REPORT SYMPTOM

This is to allow a user to send a multi-cast notification to all the users that particular users has had contacted with, in case that user experiences covid-19 like symptoms.

ADMIN DASHBOARD



# DESIGN & ARCHITECTURE

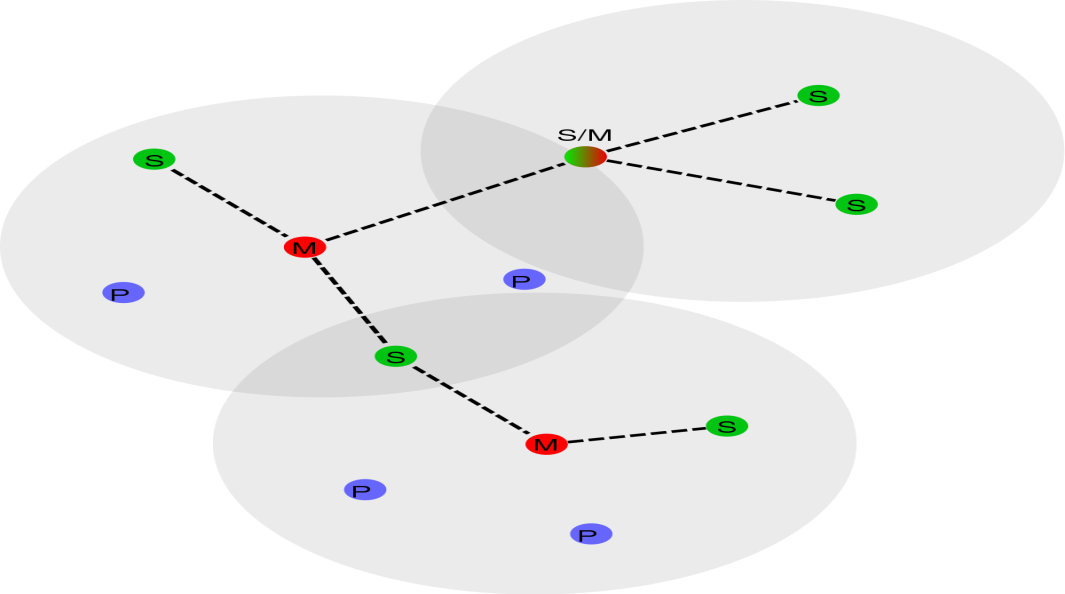
ARCHITECTURE

BLUETOOTH SCATTER NET

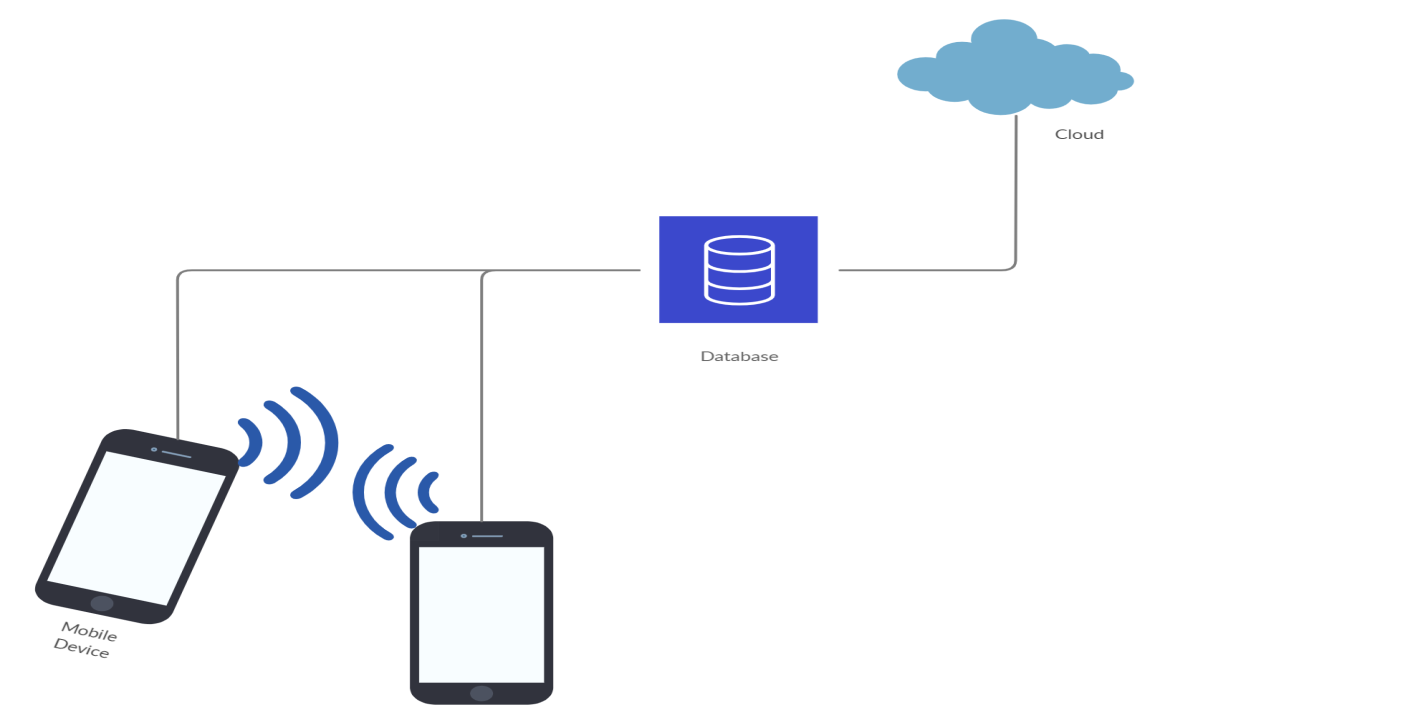
Since the application uses Bluetooth to discover other nearby devices, its architecture simulates the scatter net network.

A scatter net is a type of network that is formed between two or more Bluetooth-enabled devices, such as smartphones and newer home appliances. A scatter net is made up of at least two piconets. Bluetooth devices are peer units that act as slaves or masters. Scatter nets are formed when a device in a piconet, whether a master or a slave, decides to participate as a slave to the master of another piconet.

In our case the masters is the device that discovers the other devices, the slave is the discovered device. Each device has a chance to be a master to other devices.



SYSTEM ARCHITECTURE



# 

# SYSTEM INVESTIGATION AND ANALYSIS

# FUNCTIONAL REQUIREMENTS

Functional requirement defines the capabilities and functions that a system must be able to perform successfully. The Student project and problem set repository must be able to:

* Allow users to register.
* Allow users to report symptoms
* View notifications
* Allow admin to view reports.
* Notify users.
* Allow users to view notifications.

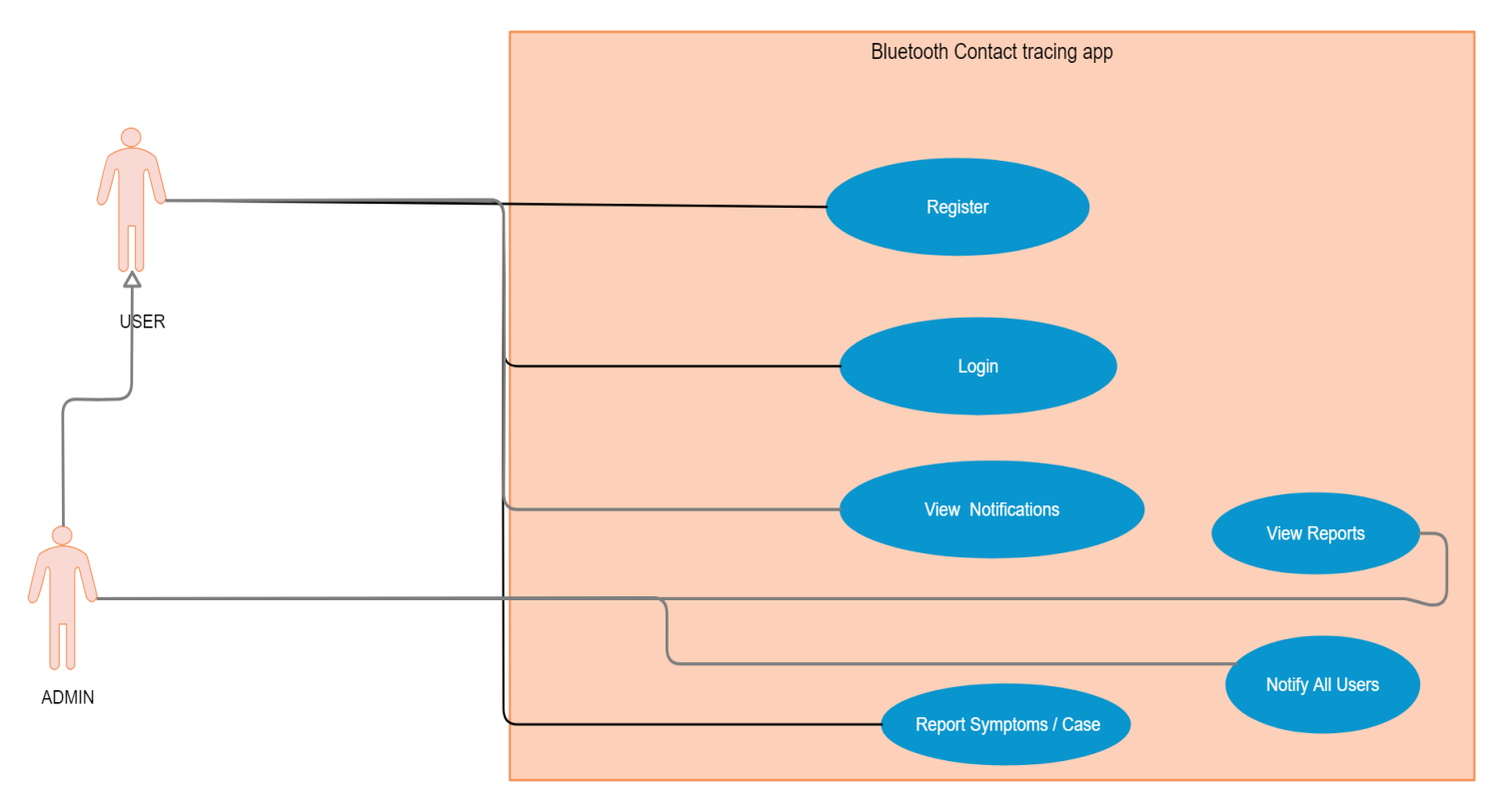
## 

## NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are things that are not necessary for the functioning of the system but they make it efficient. The researcher collected the non-requirements also and arranged them into the same document that provided the guideline for the project. The non-functional requirements are;

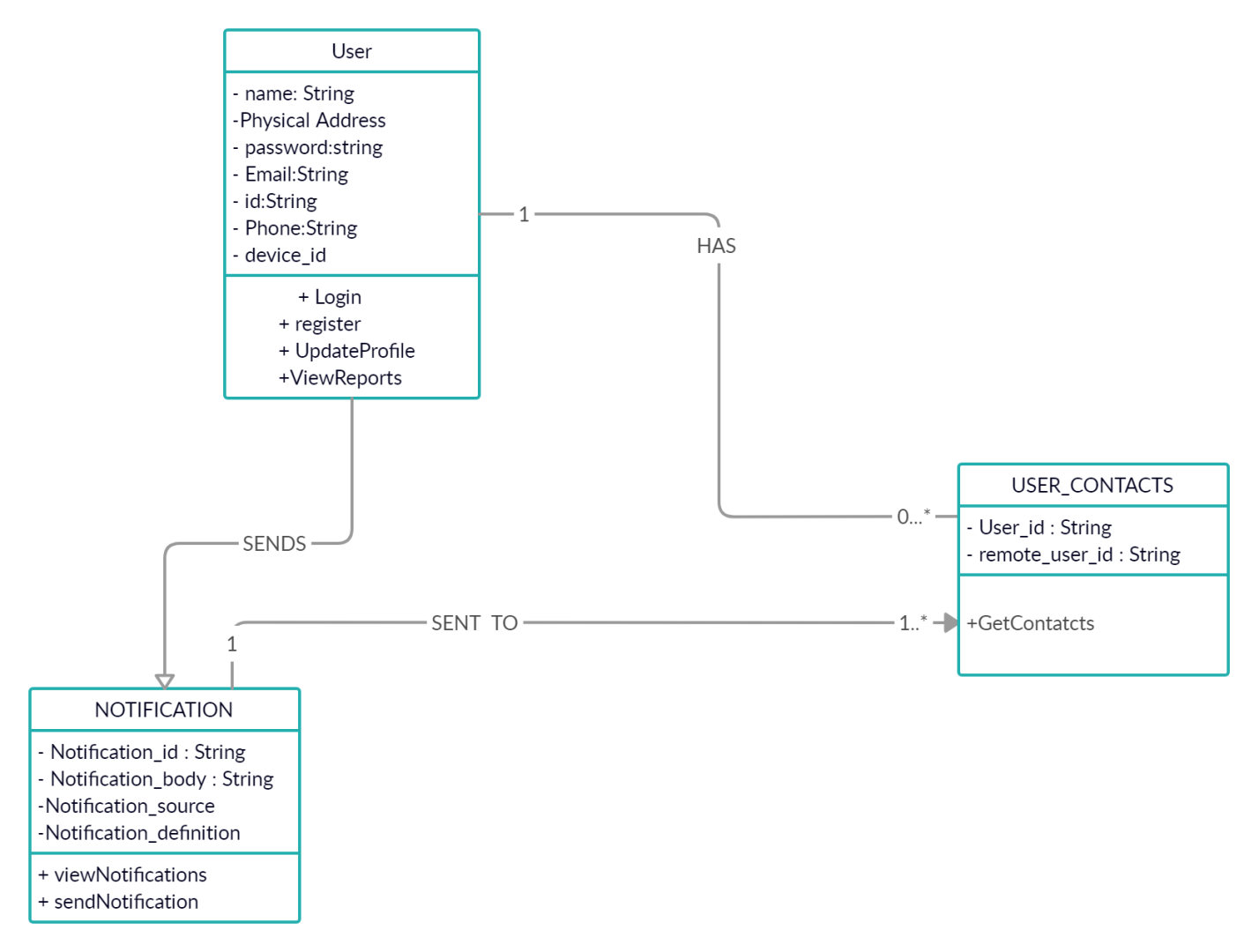
* **USER FRIENDLY** - The system should be user friendly such that it does not become difficult for the user to use.
* **USABILITY** - The system should be easy for a user to understand and use and with the right services.
* **SECURITY** - Only registered and authorized users can access the system resources.

# SYSTEM DESIGN

****

# DATABASE DEGIN

CLASS DIAGRAM



USER

This model is populated during registration. When users register, we also capture the mac-address of their devices.

|  |  |
| --- | --- |
| Name | VARCHAR(255) |
| Physical Address | VARCHAR(255) |
| Identity Number | VARCHAR(12) |
| Phone | VARCHAR(12) |
| Device\_ID | VARCHAR(255) |
| Password | VARCHAR(255) |

USER CONTACTS

This table is populated when a user’s device detects another device in a proximity distance range. Every device discovered, we populate the table with the user’s id matching the discovered mac-address in the database as the remote\_User\_id and the identity number associated with the local device as the identity number.

|  |  |
| --- | --- |
| Identity Number(foreign key) | VARCHAR(255) |
| Remote\_User\_id(foreign key) | VARCHAR(255) |
| Location | VARCHAR(255) |
| Time and date | Date |

NOTIFICATION

This table is populated when a user pushes a notification. The destination represents all the contacts that have a record associated with the notifying user in the USER CONTACTS TABLE. The notification body represents the message being communicated.

The id uniquely identifies a specific notification

|  |  |
| --- | --- |
| ID | INT(9) |
| Notification body | VARCHAR(255) |
| Destination | VARCHAR(255) |
| Source | VARCHAR(255) |

# ****HOW THE APPLICATION WILL WORK****

Each installation of the app has a single UUID.

The app requires high-level access to Bluetooth information from the device, specifically the device’s MAC address and the MAC address of nearby devices. The device with the app installed would be forced to broadcast its MAC.

People who download and use the app will be instructed to register, users will provide their names, identity number, address, and phone and then the app captures the mac-address of the device and then added to the database.

Users will always have to switch on their Bluetooth, each device broadcasts its Bluetooth mac, for each Bluetooth mac discovered by your device, it creates a database record.

**REGISTERING USERS**

When registering users, we capture the device mac address. This mac-address is associated with the user details, since every device has a unique mac-address it is easy to protect user privacy, since this will be the value displayed to other users instead of the user’s real identity number.

**CREATING RECORDS**

User records are created during registration into the application, at this time the users table is populated.

The app uses Bluetooth to detect nearby devices. For every device we discover in close proximity we capture the mac address, extract the corresponding user identity number from the database and add to the **user contacts** table as the remote discovered id.

This table associates every user to the possible contacts they have made.

**NOTIFYING USERS**

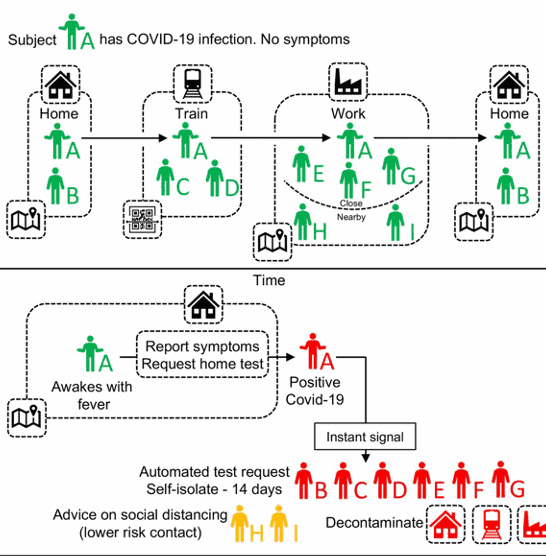
Users can push a notification, to every possible user with an existing record associated with the notifying user. For example when a person experiences any symptoms of covid-19, or has been found with covid-19. This is based on how much time the users were in the short distance proximity.

It can deliver a notification if the person has been in close proximity to someone who has contracted covid-19, provided that user pushes a notification on reporting symptoms. The receivers of the notification are selected based on the risk-scoring algorithm to decide who to alert. This algorithm takes into account how long the two people have been in proximity, and how close they got to each other. The notification will also be viewed by the admin as a recent report.

**DIAGRAM 1**

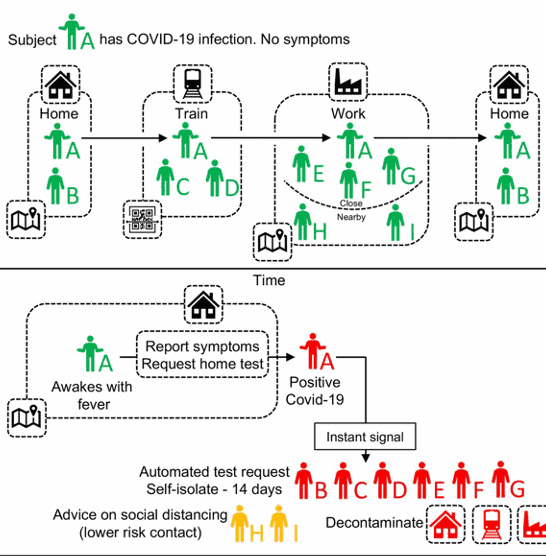
The above processes are further demonstrated in the diagrams below.

**CREATING CONTACT RECORDS**



**DIAGRAM 2**

**NOTIFYING USERS**



# IMPLEMENTATION DIRECTIONS

DEVELOPMENT TOOLS

This section gives a brief overview of the implementation tools and requirements that the Bluetooth contact tracing app will be developed with.

FLUTTER

Flutter is an app SDK for building high-performance, high-fidelity apps for iOS, Android, web ([beta](https://flutter.dev/web)), and desktop ([technical preview](https://flutter.dev/desktop)) from a single codebase.

BLUETOOTH API

The application uses Bluetooth in its major functionality. In this section we provide an overview of the Bluetooth API and how it works and why it is suitable for this task.

The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth functionality through the Android Bluetooth APIs. These APIs let applications wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

The Bluetooth APIs was chosen because of the following capabilities:

* Scan for other Bluetooth devices
* Query the local Bluetooth adapter for paired Bluetooth devices
* Establish RFCOMM channels
* Connect to other devices through service discovery
* Manage multiple connections

Classic Bluetooth

Classic Bluetooth is the right choice for more battery-intensive operations, which include streaming and communicating between Android devices. For Bluetooth devices with low power requirements, Android 4.3 (API level 18) introduces API support for Bluetooth Low Energy.

Bluetooth in Flutter:

FlutterBlue is a Bluetooth plugin for Flutter. This is a new mobile SDK to help developers build modern apps for iOS and Android to use Bluetooth functionality.

FlutterBlue used for Cross-Platform Bluetooth:

FlutterBlue offer the more for both platforms (iOS and Android) . Using the FlutterBlue instance, we can scan for and connect to nearby Bluetooth devices.

Once the Bluetooth device connects, then the Bluetooth Device object can discover services for Bluetooth services and Bluetooth characteristic and also for Bluetooth Descriptors. After Discovering and connecting the Bluetooth device then it is used to directly interact with characteristics and descriptors.

USING BLUETOOTH FUNCTIONALITY WITH FLUTTER

The plugin/dependency we will be using in this app to add Bluetooth is “[flutter Bluetooth serial](https://pub.dartlang.org/packages/flutter_bluetooth_serial)”, this plugin is implemented from another parent plugin called “flutter blue”. This is a very new plugin, the only plugin for Bluetooth available as of now. This plugin will only work for Android.

FIREBASE

Firebase will be used as the database for the mobile application.

It is an API that allows application data to be synchronized across clients and stored on **Firebase's** cloud. It provides a real-time database and back-end as a service for application developers.

AUTHENTICATION

This is a fundamental feature that verifies identity of user before accessing the privacy of a user. It will be implemented using firebase authentication. Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to mobile apps. It is chosen because it supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

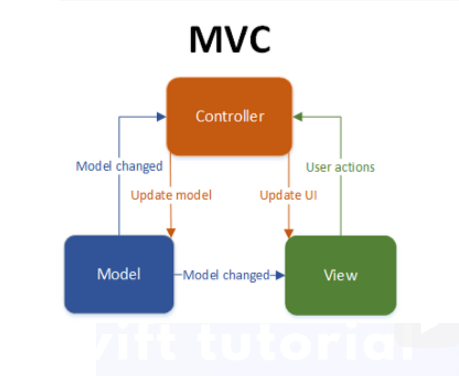
DEVELOPMENT ARCHITECTURE

MVC is the chosen approach to building this mobile app . Being a layer-based pattern, MVC separates user interface functionality from business/application/domain logic. As the name implies, this pattern consists of three main components and layers respectively:

* **Model** is a data holder, responsible for handling the business logic and defining rules to modify and operate data.
* **View** is a visual part and makes data from the model visible in the user interface.
* **Controller** is a front and back communicator between the model and the view. Based on the user behavior input from the view, it could change the model, process data through it and pass the results back to the view.

This structure helps mobile apps to focus on separate functions and tasks simultaneously with the help of these isolated layers. This way you can adjust or refactor your code much easier, as well as add new features, thus simplifying the process of app maintenance. The same rule doesn’t apply to iOS applications, however. Unlike this traditional model applied by Android, iOS decided to enhance MVC slightly.

# MVC ARCHITECTURE



# MODEL DEFINITION

Models define and set the structure and type of data that will be captured and stored in the database. This step also includes the definition of relationships between the models.

The system or application has the following models

* User
* Notification
* User Contacts

USER

The user model simulates a system user with all the required attributes of the system user. The database table associated with this model has the following fields

USER: Names, id number, physical address, phone password

NOTIFICATION

This model provides the mechanism of user notification. It simulates a notification sent and received by a user of the system. Fields: destination, message body, id.

USER CONTACTS

This model associates a user to all the possible contact made by a particular user. It is made up of all user\_id’s time and location .The has the following

User\_id, discovered\_user\_id, location, time.

Discovered user\_id represents the id of the discovered user.

User\_id represents the id of the detecting device.

VIEW DESIGN

This will involve the design, structuring and restructuring user interfaces to meet the purpose and type of user using the application, and also providing the rightful display information in the rightful formats. This involves login pages, display pages and dashboards.

CONTROLLER

In this system the controller represents the application backend, database operations, functionality.

LIMITATIONS

It does not work for users with their Bluetooth not switched on.

# REFERENCES

Bay, J, Kek, J., Tan, A., Hau, C., Yongquan, L., Ta, J., Quy,T.A. (2020). BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders; available at https://go.nature.com/2wwcwg4

O'Neill,P.H., Ryan-Mosley,T., Johnson,B. (2020) A flood of coronavirus apps are traking us, now its time to track them, archive page, MS TECH | GETTY

Zhang, B., Kreps, S. E., & McMurry, N. (2020). Americans' perceptions of privacy and surveillance in the COVID-19 Pandemic. https://doi.org/10.31219/osf.io/9wz3y

Thorneloe, R., Epton, T., Fynn, W., Daly, M., Stanulewicz, N., Kassianos, A., … Hart, J. (2020, April 30). SCOPING REVIEW OF MOBILE PHONE APP UPTAKE AND ENGAGEMENT TO INFORM DIGITAL CONTACT TRACING TOOLS FOR COVID-19. https://doi.org/10.31234/osf.io/qe9b6