NMAM Institute of Technology

(An Autonomous Institute Affiliated to VTU, Belagavi)

(A unit of NITTE Education Trust)

NITTE – 574110, UDUPI DIST., KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MINI PROJECT ON

LOCATE A FRIEND

Team Members:

1) Mr Manjunath Patkar (4NM17CS100)

2) Mr Maukashyap (4NM17CS101**)**

3) Mr Oliver Castelino (4NM18CS413)

Under the Guidance

Mr Ramesha Shettigar

Assistant Professor Gd-II

Department of Computer Science and Engineering

NMAMIT Nitte.

NMAM Institute of Technology

(An Autonomous Institute Affiliated to VTU, Belagavi)

(A unit of NITTE Education Trust)

NITTE – 574110, UDUPI DIST., KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

"Locate A Friend"

is a bonafide work carried out by

Manjunath Patkar - 4NM17CS100

Manukashayp U V - 4NM17CS101

Oliver Castelino - 4NM18CS413

in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in computer science and engineering prescribed by the Vishvesvaraya Technological University, Belagavi during the year 2019 - 2020

It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report.

The mini-project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to making this project successful.

First and foremost, we express our deep sense of gratitude and indebtedness to our guide Mr Ramesha Shettigar, Assistant Professor, Department of Computer Science and Engineering, for his inspiring guidance, constant encouragement, support and suggestions for improvement during the course of our project.

We sincerely thank Dr K.R. Udaya Kumar Reddy, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte.

Our sincere thanks to our beloved principal, Dr Niranjan N. Chiplunkar for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

We also thank all those who have supported us throughout the entire duration of our project.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout the course of our project.

Manukashyap U V - 4NM17CS101
Manjunath Patkar - 4NM17CS100
Oliver Castelino - 4NM18CS413

Table of Contents

Chapter 1: Introduction

1.1 Scope

Locate a friend is designed with simplicity and ease of use in mind. It is used to keep track of loved ones and keep oneself safe in dire situations. Along with the benefit we provide we also keep the user privacy in mind. A user can share the location with another user only if they are friends and both consent to do so.

1.2 Importance

In this day and age, there is danger lurking everywhere for people of any age. But if everyone had the ability to track each other with each others consent it would mitigate the risk we face by a small factor and prove to be a saviour in the long run. We aim to provide this service but how to utilise this is left to the user.

1.3 Objective

The objective of this mobile application to create a mobile application that can track the real-time location of a person and also enable either of the people to contact the security in case of necessary. Along with the core service we aim to enable the user secure authentication service that can be further secured in the future and the comfort in knowing that their location will only be shared with the people they provide consent to.

Along with the location of the friend, we also aim to provide the user with the ability to view the last date and time the location was updated and to navigate to that location or view it in more detail in google street view by connection directly to google maps.

Chapter 2: Literature Survey

2.1 Technical Background

The initial approach is the use a broadcast receiver to beam the location of the user at certain intervals to the firebase and the people who are on the friend list of that particular person can view it using the GUI provided by the app.

2.2 Existing System

Even though there are existing systems but they are flawed in their execution for this specific problem we are trying to solve since they offer these services as a sidecar for their main service. Thus these options are hidden in plain sight for most people and they tend to go unutilised. Some other similar tracking systems require us to carry a separate GPS module which is likely to be forgotten or lost, unlike the smartphone that is an integral part of our daily lives.

2.3 Proposed System

Our system is solely supposed to be a real-time location tracking and sharing system along with the ability to call the regional authority responsible for the security of the individuals. Our system provides a secure email-based authentication system. And the user has the ability to add only the people they are familiar to as friends and if and only if both parties agree then only they can view each others location.

The user can view their friend’s location along with accurate date and time and also they can be able to navigate to that place with the click of a button or have a better look at the place using google maps and street view.

Chapter 3: System requirement and specification

3.1 Functional Requirements

- The data should be secured.
- Reliable authentication system.
- The location should not be shared with any person other than the friend(s) of that person.
- It app should be easily sharable with friends enabling speedy install if required.
- User should be able to submit feedback and contact developer if required.

3.2 User Requirements

- Android Phone.
- Internet connection (need not be continuous and reliable)/
- Google Play services framework.
- Location enabled.

3.3 Software Requirements (For developers)

- Operating System (anyone out of the below options)
    - Microsoft® Windows® 7/8/10 (64-bit)
    - Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
    - GNOME or KDE desktop
    - Chrome OS
- Android Studio (Download Android Studio and SDK tools)
- Firebase
    - Realtime Database (Firebase Realtime Database)
    - Firebase Messaging service (FirebaseMessagingService)
- Libraries Used
    - Karumi/Dexter (Karumi)
    - Spots progress dialogue (spots-dialog)

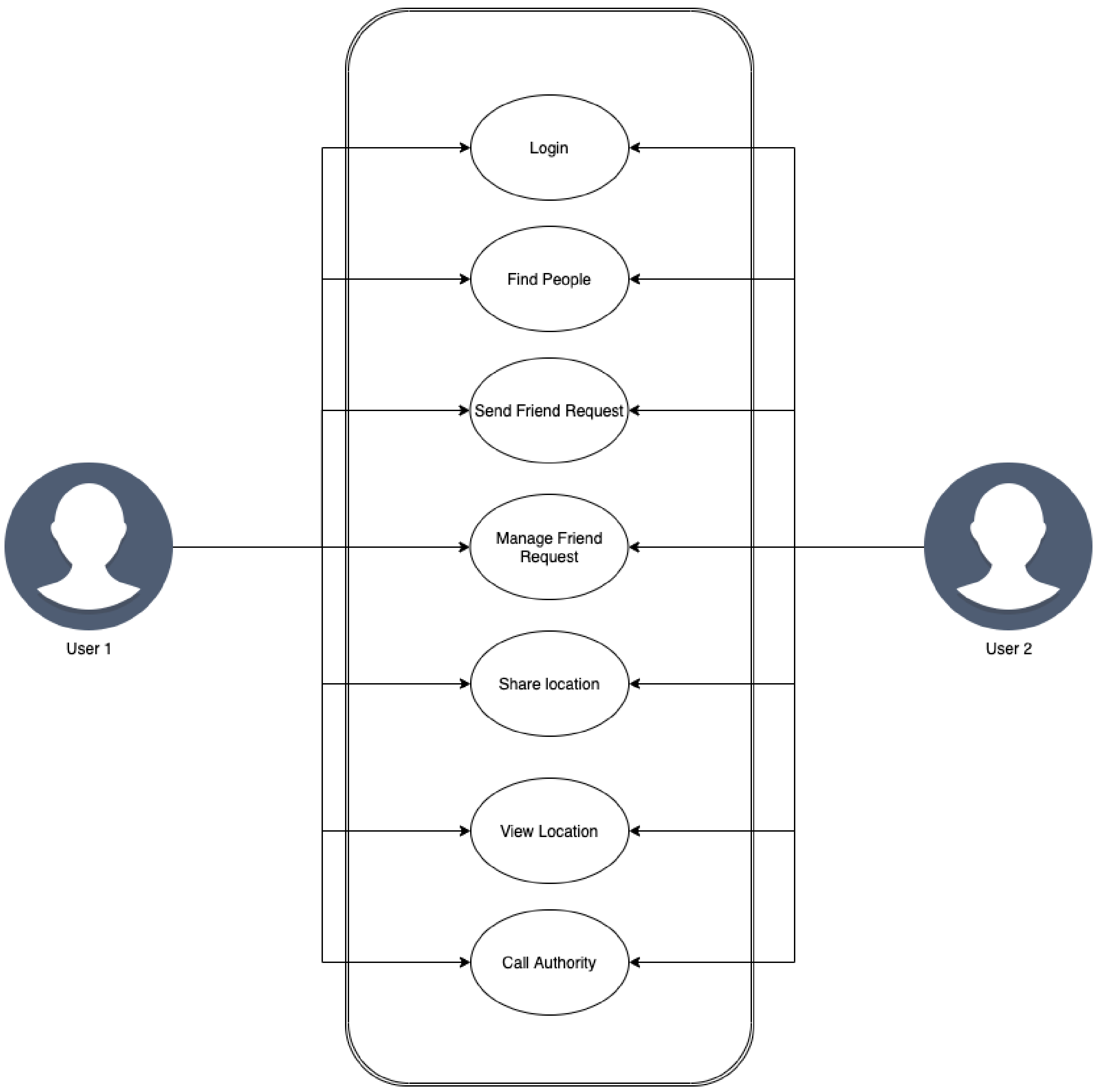
    - RxAndroid (ReactiveX)
    - RxJava2 Adapter (Retrofit RxJava)
    - Paper (PaperDB)
    - Android About Page (medyo)

3.4 Hardware Requirement

- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

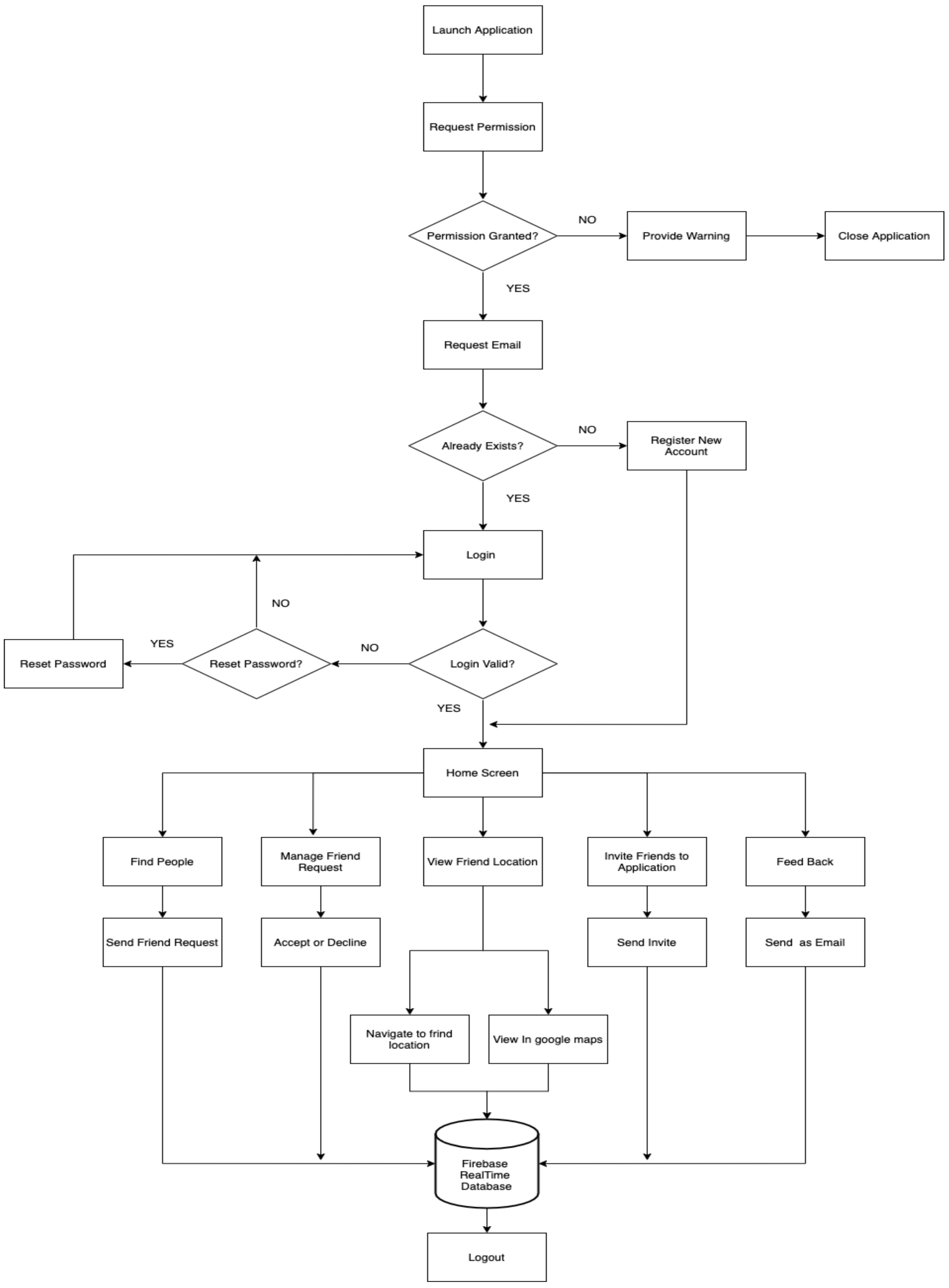
Chapter 4: System Design

4.1 Use case model



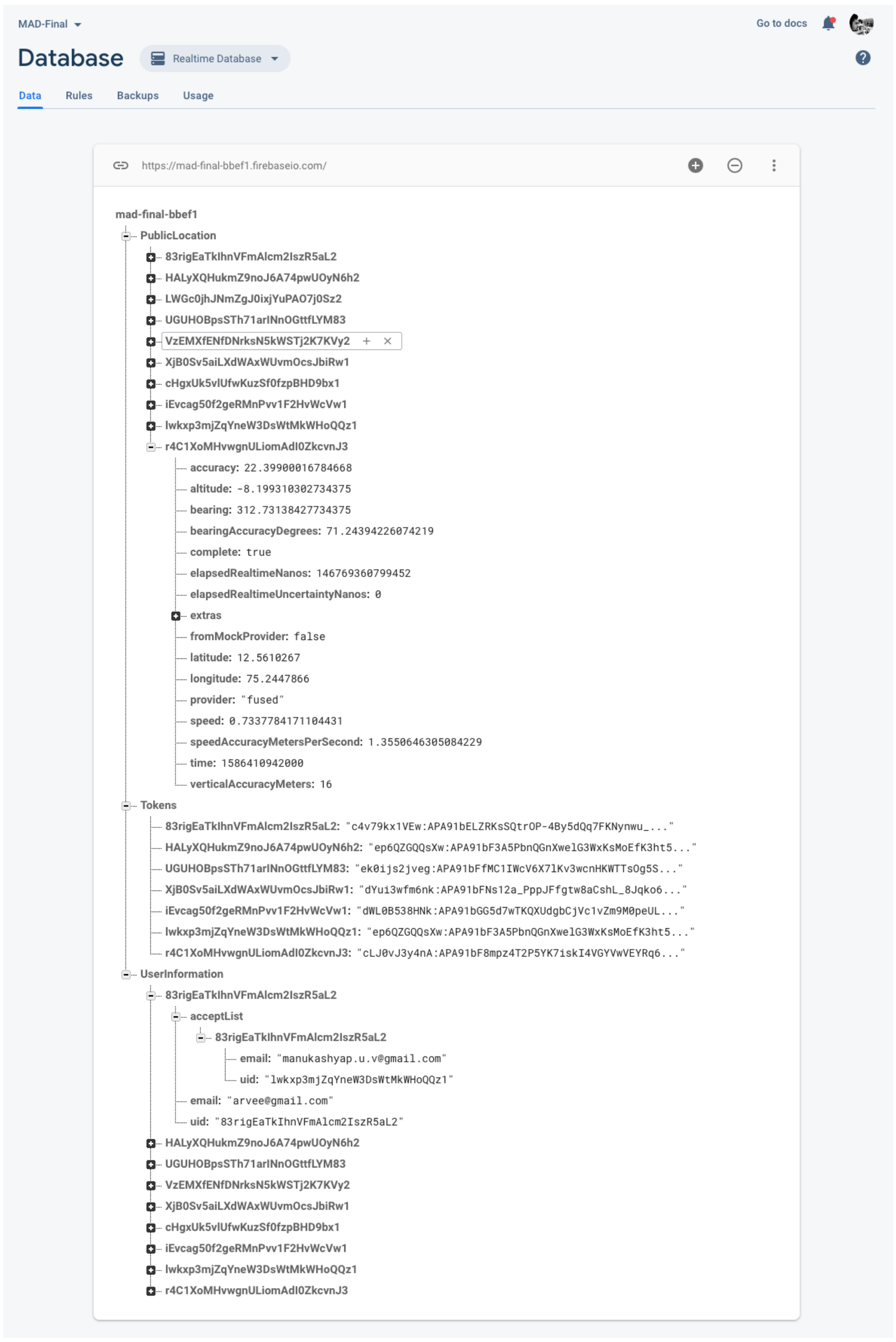
USE CASE MODEL - LOCATE. A FRIEND

4.2 Data flow diagram



4.3 Database Design

Snapshot of stored data from real-time database



```
https://mad-final-bbef1.firebaseio.com/

mad-final-bbef1
  PublicLocation
    83rigEaTkIhnVFmAlcm2IszR5aL2
    HALyXQHukmZ9noJ6A74pwUOyN6h2
    LWGc0jhJNmZgJ0ixjYuPAO7j0Sz2
    UGUHOBpsSTh71arINnOGttfLYM83
    VzEMXfENfDNrksN5kWSTj2K7KVy2
    XjB0Sv5aiLXdWAxWUvmOcsJbiRw1
    cHgxUk5vlUfwKuzSf0fzpBHD9bx1
    iEvcag50f2geRMnPvv1F2HvWcVw1
    lwkxp3mjZqYneW3DsWtMkWHoQQz1
    r4C1XoMHvwgnULiomAdI0ZkcvnJ3
      accuracy: 22.39900016784668
      altitude: -8.199310302734375
      bearing: 312.73138427734375
      bearingAccuracyDegrees: 71.24394226074219
      complete: true
      elapsedRealtimeNanos: 146769360799452
      elapsedRealtimeUncertaintyNanos: 0
      extras
      fromMockProvider: false
      latitude: 12.5610267
      longitude: 75.2447866
      provider: "fused"
      speed: 0.7337784171104431
      speedAccuracyMetersPerSecond: 1.3550646305084229
      time: 1586410942000
      verticalAccuracyMeters: 16
  Tokens
    83rigEaTkIhnVFmAlcm2IszR5aL2: "c4v79kx1VEw:APA91bELZRKsSQtrOP-4By5dQq7FKNynwu_..."
    HALyXQHukmZ9noJ6A74pwUOyN6h2: "ep6QZGQQsXw:APA91bF3A5PbnQGnXwelG3WxKsMoEfK3ht5..."
    UGUHOBpsSTh71arINnOGttfLYM83: "ek0ijs2jveg:APA91bFfMC1IWcV6X7lKv3wcnHKWTTsOg5S..."
    XjB0Sv5aiLXdWAxWUvmOcsJbiRw1: "dYui3wfm6nk:APA91bFNs12a_PppJFfgtw8aCshL_8Jqko6..."
    iEvcag50f2geRMnPvv1F2HvWcVw1: "dWL0B538HNk:APA91bGG5d7wTKQXUdgbCjVc1vZm9M0peUL..."
    lwkxp3mjZqYneW3DsWtMkWHoQQz1: "ep6QZGQQsXw:APA91bF3A5PbnQGnXwelG3WxKsMoEfK3ht5..."
    r4C1XoMHvwgnULiomAdI0ZkcvnJ3: "cLJ0vJ3y4nA:APA91bF8mpz4T2P5YK7iskI4VGYVwVEYRq6..."
  UserInformation
    83rigEaTkIhnVFmAlcm2IszR5aL2
      acceptList
        83rigEaTkIhnVFmAlcm2IszR5aL2
          email: "manukashyap.u.v@gmail.com"
          uid: "lwkxp3mjZqYneW3DsWtMkWHoQQz1"
      email: "arvee@gmail.com"
      uid: "83rigEaTkIhnVFmAlcm2IszR5aL2"
    HALyXQHukmZ9noJ6A74pwUOyN6h2
    UGUHOBpsSTh71arINnOGttfLYM83
    VzEMXfENfDNrksN5kWSTj2K7KVy2
    XjB0Sv5aiLXdWAxWUvmOcsJbiRw1
    cHgxUk5vlUfwKuzSf0fzpBHD9bx1
    iEvcag50f2geRMnPvv1F2HvWcVw1
    lwkxp3mjZqYneW3DsWtMkWHoQQz1
    r4C1XoMHvwgnULiomAdI0ZkcvnJ3
```

The database is designed using Google Firebase Console in which data is stored in a popular data structure known as the JSON tree (JavaScript Object Notation). Every time when the data transfer happens from the client end, the information given to the UI is converted into JSON tree structure which is an efficient and faster way to retrieve and store data.

Sample JSON storing location of a user in the database

```
{
 "PublicLocation" : {
  "83rigEaTkIhnVFmAlcm2IszR5aL2" : {
   "accuracy" : 18.399999618530273,
   "altitude" : 0.34086214675636906,
   "bearing" : 167.195556640625,
   "complete" : true,
   "elapsedRealtimeNanos" : 124197982117742,
   "extras" : {
    "empty" : false,
    "parcelled" : false
   },
   "fromMockProvider" : false,
   "latitude" : 12.5608362,
   "longitude" : 75.2448044,
   "provider" : "fused",
   "speed" : 0.8584941625595093,
   "time" : 1585777028000
  }
 }
}
```

Chapter 5: Implementation

5.1 Solution Approach/Methodology

5.1.1 Firebase

Firebase is Google's mobile application development platform that helps you build, improve, and grow your app. In our app, we plan to use firebase for authentication and data storage. We also use it to send friend request but that will be mentioned later. We utilise the Email and Password login option provided by the firebase as our authentication system. And we will use the real-time database provided by the firebase to store our user's data in a secure manner.

Connecting firebase to the android app: Add Firebase to your Android project

5.1.2 Firebase Messaging

We need some means of enabling the user to send a friend request to the people they know. This is where the firebase messaging service comes into play.

Firebase Cloud Messaging Platform (formerly named as GCM) is a free mobile notification service by Google that enables (third-party) app developers to send notifications from GCM (Google Cloud Messaging) servers to their users.

Setting up FCM for Android: Set up a Firebase Cloud Messaging client app on Android

5.1.3 Local storage

Along with using real-time cloud database we also need to use in device storage to store some details such as user information, location etc. We use PaperDB to help us in that regard.

Know more about PaperDB: pilgr/Paper: Paper is a fast NoSQL-like storage for Java/Kotlin objects on Android with automatic schema migration support.

5.1.4 Broadcast Reciever

We need some kind of mechanism to update the user location in real-time to firebase. This is where the broadcast receiver comes into play. A broadcast receiver is an Android component which allows you to send or receive Android system or application events.

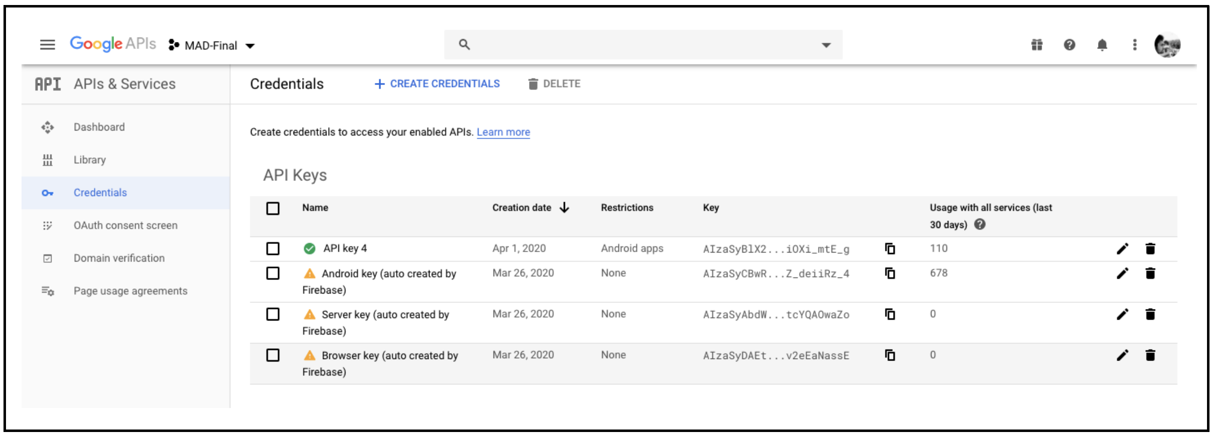
Know more about Broadcast Reciever: BroadcastReceiver

5.1.5 Google Maps API

Even though we have the user location to display it we need a maps overlay in our project. We can not use the services provided by google maps unless we have an API key.

Know more: Get an API Key | Maps JavaScript API

Google API consol



Create credentials to access your enabled APIs. Learn more

API Keys

Name	Creation date ↓	Restrictions	Key	Usage with all services (last 30 days)
API key 4	Apr 1, 2020	Android apps	AIzaSyBlX2...iOXi_mtE_g	110
Android key (auto created by Firebase)	Mar 26, 2020	None	AIzaSyCBwR...Z_deiiRz_4	678
Server key (auto created by Firebase)	Mar 26, 2020	None	AIzaSyAbdW...tcYQAOwaZo	0
Browser key (auto created by Firebase)	Mar 26, 2020	None	AIzaSyDAEt...v2eEaNassE	0

5.2 Code Implementation**

**Partial Implementation is shown here because of space constraints

5.2.1 User SignUp and SignIn

```
private void showSignInOptions() {
  startActivityForResult(AuthUI.getInstance()
         .createSignInIntentBuilder()
         .setAvailableProviders(providers)
         .build(), MY_REQUEST_CODE);
}

@Override
```

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  if (requestCode == MY_REQUEST_CODE) {
    IdpResponse response = IdpResponse.fromResultIntent(data);
    if (resultCode == RESULT_OK) {
      final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

      //Checking of the user is already registered
      user_information.orderByKey()
          .equalTo(firebaseUser.getUid())
          .addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
              if (dataSnapshot.getValue() == null) //User is not present
              {
                if (!dataSnapshot.child(firebaseUser.getUid()).exists()) {
                  Common.loggedUser = new User(firebaseUser.getUid(),
firebaseUser.getEmail());

                  //Adding to the database

user_information.child(Common.loggedUser.getUid()).setValue(Common.loggedUser);
                }
              } else {
                Common.loggedUser =
dataSnapshot.child(firebaseUser.getUid()).getValue(User.class);
              }

              //Save the Uid to storage and update the location from background
              Paper.book().write(Common.USER_UID_SAVE_KEY,
Common.loggedUser.getUid());
              updateToken(firebaseUser);
              setupUI();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
          });
    }
  }
}
```

5.2.2 Loading User List

```
private void loadUserList() {
  final List<String> lstUserEmail = new ArrayList<>();
  DatabaseReference userList =
FirebaseDatabase.getInstance().getReference(Common.USER_INFORMATION);
  userList.addListenerForSingleValueEvent(new ValueEventListener() {
     @Override
     public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for (DataSnapshot userSnapshot : dataSnapshot.getChildren()) {
           User user = userSnapshot.getValue(User.class);
           lstUserEmail.add(user.getEmail());
        }
        firebaseLoadDone.onFirebaseLoadUserNameDone(lstUserEmail);
     }

     @Override
     public void onCancelled(@NonNull DatabaseError databaseError) {
        firebaseLoadDone.onFirebaseLoadFailed(databaseError.getMessage());
     }
  });
}
```

5.3.3 Firebase Messaging Service

```
@Override
public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
  super.onMessageReceived(remoteMessage);
  if (remoteMessage.getData() != null) {
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        sendNotificationWithChannel(remoteMessage);
        addRequestToUserInformation(remoteMessage.getData());
     } else {
        sendNotification(remoteMessage);
        addRequestToUserInformation(remoteMessage.getData());
     }
  }
}
```

5.3.4 Broadcast Reciever for location

```
private void updateLocation() {
  buildLocationRequest();
  fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this);
  fusedLocationProviderClient.requestLocationUpdates(locationRequest,
getPendingIntent());

}

private PendingIntent getPendingIntent() {
  //Toast.makeText(this, "here", Toast.LENGTH_SHORT).show();
  Intent intent = new Intent(this, MyLocationReciever.class);
  intent.setAction(MyLocationReciever.ACTION);
  return PendingIntent.getBroadcast(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
}

private void buildLocationRequest() {
  locationRequest = new LocationRequest();
  locationRequest.setSmallestDisplacement(10f);
  locationRequest.setFastestInterval(3000);
  locationRequest.setInterval(5000);
  locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

5.3.5 Display friend location

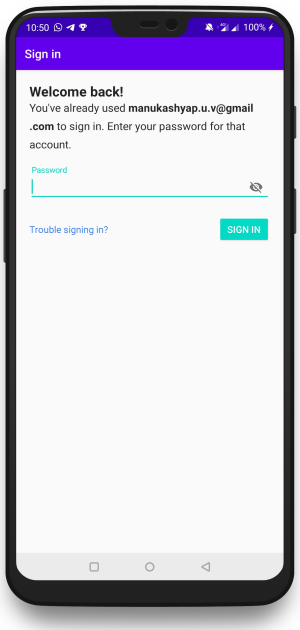
```
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
  if (dataSnapshot.getValue() != null) {
      MyLocation location = dataSnapshot.getValue(MyLocation.class);

      //Add Marker
      LatLng userMarker = new LatLng(location.getLatitude(), location.getLongitude());
      mMap.addMarker(new MarkerOptions().position(userMarker)
              .title(Common.trakingUser.getEmail())

.snippet(Common.getDateFormatted(Common.convertTimeStampToDate(location.getTim
e()))));

      mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(userMarker, 16f));
  } }
```

Chapter 6: Screenshots



10:50 100%

Sign in

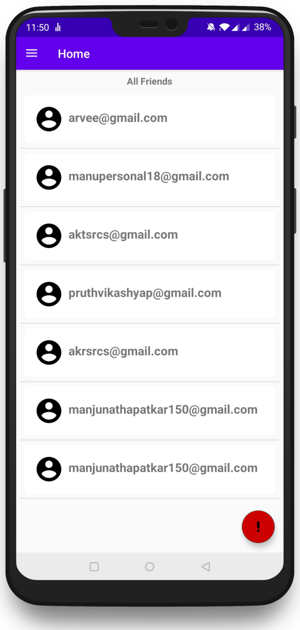
Welcome back!

You've already used **manukashyap.u.v@gmail.com** to sign in. Enter your password for that account.

Password

Trouble signing in? SIGN IN

LOCATE A FRIEND



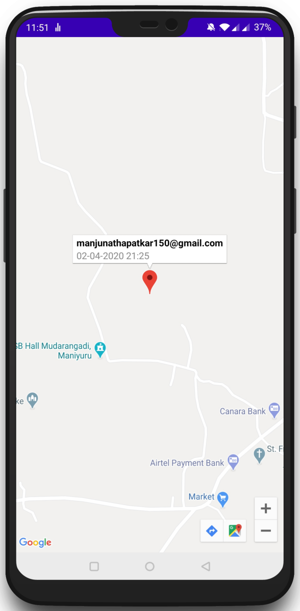
11:50 38%

Home

All Friends

- arvee@gmail.com
- manupersonal18@gmail.com
- aktsrcs@gmail.com
- pruthvikashyap@gmail.com
- akrsrcs@gmail.com
- manjunathapatkar150@gmail.com
- manjunathapatkar150@gmail.com

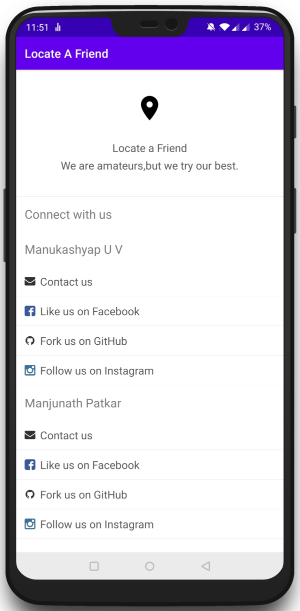
LOCATE A FRIEND



11:51 37%

manjunathapatkar150@gmail.com
02-04-2020 21:25

LOCATE A FRIEND



11:51 37%

Locate A Friend

Locate a Friend
We are amateurs,but we try our best.

Connect with us

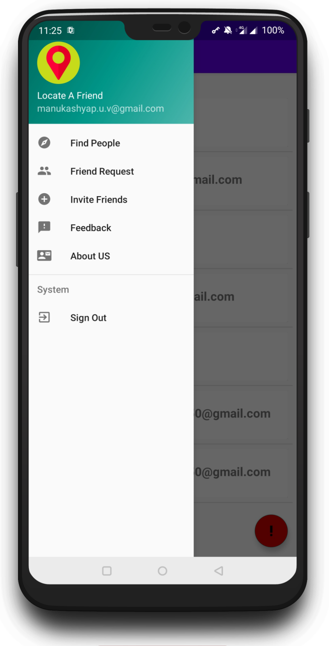
Manukashyap U V

- Contact us
- Like us on Facebook
- Fork us on GitHub
- Follow us on Instagram

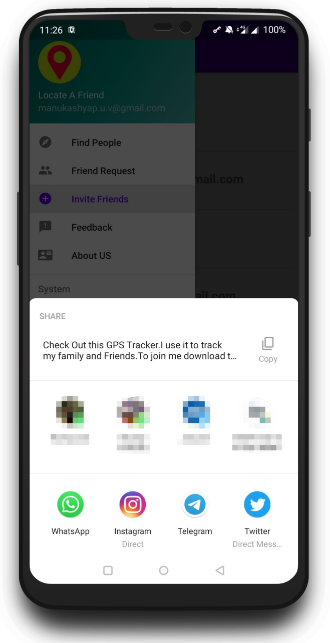
Manjunath Patkar

- Contact us
- Like us on Facebook
- Fork us on GitHub
- Follow us on Instagram

LOCATE A FRIEND

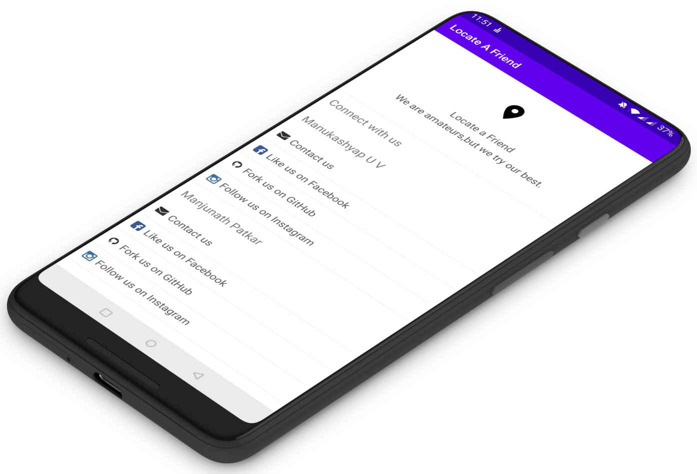


LOCATE A FRIEND



Check Out this GPS Tracker.I use it to track my family and Friends.To join me download t...

LOCATE A FRIEND



We are amateurs,but we try our best.

LOCATE A FRIEND

Chapter 7: Conclusion and Future Work

7.1 Conclusion/Result

- Locate A Friend is an android application that is intended to be used as a personal safety application which allows the user to track their friends and vice versa.
- This app respects the privacy of the user and the location is only shared with the people that the user consents to.
- As it uses the android platform it supports more than a million phones and can run on any Android phone of version marshmallow or higher and having google play services.
- Can be used anywhere in the world except the People's Republic of China.

7.2 Future Work

- More authentication options can be added.
- The existing authentication system can be made more robust.
- More personalization options can be given to the user with the option of having Display name, Display Picture, Secret Friend etc.
- Optimised search option can be added for ease of interaction with the GUI.
- Visual improvements can be added to the app such as themes and auto day and night mode.

Chapter 8: References and External Resources

8.1 References

- Android Studio Guide (Developer Guides)
- StackOverflow (Stack Overflow - Where Developers Learn, Share, & Build Careers)
- Youtube (Youtube)

8.2 External Resource

- Karumi/Dexter (Karumi)
- Spots progress dialogue (spots-dialog)
- RxAndroid (ReactiveX)
- RxJava2 Adapter (Retrofit RxJava)
- Paper (PaperDB)
- Android About Page (medyo)
- HiShoot2i Material (Releases · hishoot2i/Hishoot2i)