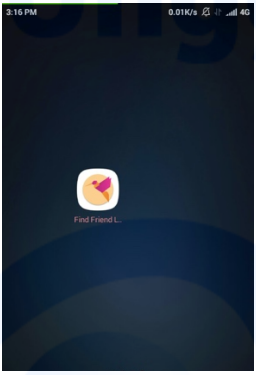
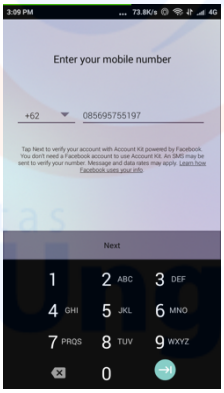
Lampiran 2. Tampilan Aplikasi

Lampiran 2.1 *Icon Launcher*



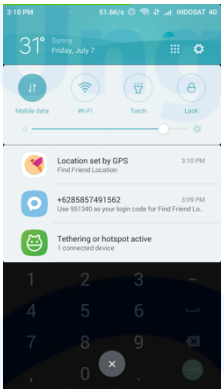
Gambar L2-1 Lampiran 2.1 - *Icon Laucher* Aplikasi

Lampiran 2.2 Tampilan *Login*



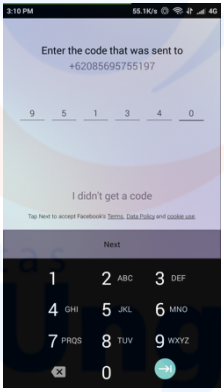
Gambar L2-2 Lampiran 2.2 - Tampilan *Login*

Lampiran 2.3 SMS Kode Verifikasi



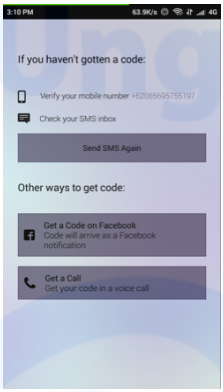
Gambar L2-3 Lampiran 2.3 - SMS Kode Verifikasi

Lampiran 2.4 Tampilan Memasukan Kode Verifikasi



Gambar L2-4 Lampiran 2.4 - Memasukan Kode Verifikasi

Lampiran 2.5 Tampilan Meminta Ulang SMS Kode Verifikasi

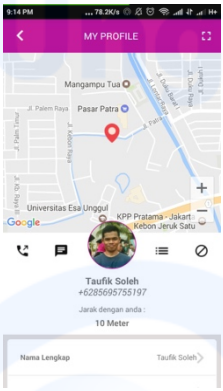


Gambar L2-5 Lampiran 2.5 - Tampilan Meminta Ulang SMS Kode Verifikasi

Lampiran 2.6 Tampilan *List* Teman

Gambar L2-6 Lampiran 2.6 – Tampilan *List* Teman

Lampiran 2.7 Tampilan Profile Penguna dan Profile Teman

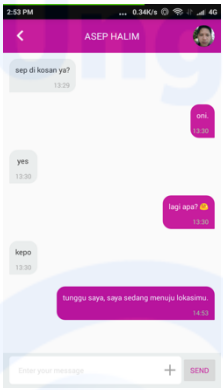


Gambar L2-7 Lampiran 2.7 - Tampilan Profile Pengguna dan Teman

Lampiran 2.8 Tampilan Mengganti Foto Profile

Gambar L2-8 Lampiran 2.8 - Tampilan Mengganti Foto Profile

Lampiran 2.9 Tampilan Percakapaan

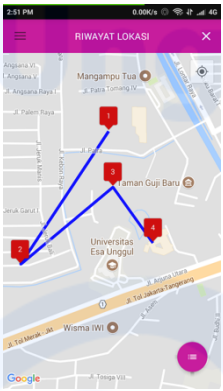


Gambar L2-9 Lampiran 2.9 - Tampilan Percakapaan

Lampiran 2.10 Tampilan Kalender Riwayat Lokasi

Gambar L2-10 Lampiran 2.10 - Tampilan Kalender Riwayat Lokasi

Lampiran 2.11 Tampilan Peta Riwayat Lokasi

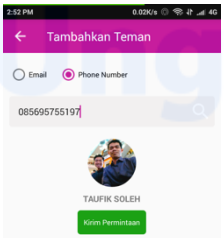


Gambar L2-11 Lampiran 2.11 - Tampilan Peta Riwayat Lokasi

Lampiran 2.12 Tampilan *List* Riwayat Lokasi

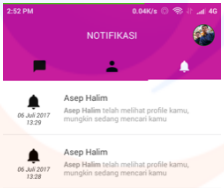
Gambar L2-12 Lampiran 2.12 - Tampilan List Riwayat Lokasi

Lampiran 2.13 Tampilan Hubungkan Teman



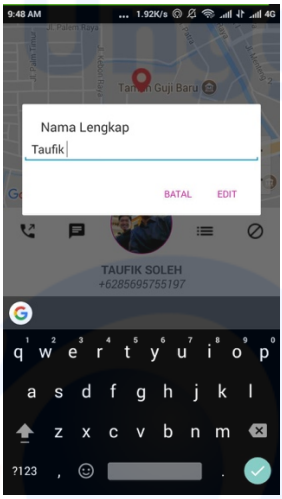
Gambar L2-13 Lampiran 2.13 - Tampilan Hubungkan Teman

Lampiran 2.14 Tampilan Notifikasi



Gambar L2-14 Lampiran 2.14 - Tampilan Notifikasi

Lampiran 2.15 Tampilan Edit Identitas Pengguna



Gambar L2-15 Lampiran 2.15 - Tampilan Edit Identitas Pengguna

Lampiran 3. Source Code Aplikasi

Untuk melihat lebih lengkap source code dari aplikasi ini, silahkan akses link dibawah ini.

Google Drive : https://drive.google.com/drive/folders/0B4cFjaWns-r5OG92QW1JX203aHM?usp=sharing

MainActivity.java

```
public class MainActivity extends BaseActivity {

    GPSTracker gps;
    private ViewPager viewPager;

    private Session session;
    private FloatingActionButton fab;
    private CircleImageView circleImageView;
    private String SRC;
    private int TAB_FROM_PUSH_NOTIFICATION;
    private String NOTIFICATION_KEY_ID;
    private Bundle bundle;
    private NotificationManager notificationManager;
    private Notification.Builder builder;
    private Notification n;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /** Session Active */
        sessoinStart();

        /** retrieve data from PUSH NOTIFICATION*/
        bundle = getIntent().getExtras();
        if (bundle != null) {
            TAB_FROM_PUSH_NOTIFICATION =
bundle.getInt("TAB");
            NOTIFICATION_KEY_ID =
bundle.getString("NOTIFICATION_KEY_ID");

            /** update status notification from unread to
read*/
            Notifications notifications = new
Notifications();

notifications.changeStatusNotificationToRead(session.getU
serId(), NOTIFICATION_KEY_ID);
        }
```

```
        //Coloring status bar
        systemBarTint();

        // set Layout Activity
        setContentView(R.layout.main_activity);

        //setup toolbar
        setupToolbar(R.id.toolbar, Title.FRIENDS);

        fab = (FloatingActionButton)
findViewById(R.id.floatingActionButton);
        setFab(R.drawable.ic_person_add_white_18dp, new
Intent(MainActivity.this, FriendInviteActivity.class));

        circleImageView = (CircleImageView)
findViewById(R.id.toolbar_image_profile);
        circleImageView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(MainActivity.this,
ProfileUserActivity.class);
                i.putExtra("USER_ID",
session.getUserId());
                startActivity(i);


Config.removeTransition(MainActivity.this, i);
            }
        });

        TabLayout tabLayout = (TabLayout)
findViewById(R.id.tabs);
        tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);

tabLayout.addTab(tabLayout.newTab().setIcon(R.drawable.ic
_chat));

tabLayout.addTab(tabLayout.newTab().setIcon(R.drawable.ic
_friends));

tabLayout.addTab(tabLayout.newTab().setIcon(R.drawable.ic
_notification));

//tabLayout.addTab(tabLayout.newTab().setIcon(R.drawable.
ic_star_black_24dp));

        Log.e("Print Debut ---------> ",
```

```
String.valueOf(tabLayout.getTabCount()));
        viewPager = (ViewPager)
findViewById(R.id.main_pager);
        viewPager.setOffscreenPageLimit(4);
        PagerMainAdapter pagerMainAdapter = new
PagerMainAdapter(getSupportFragmentManager(),
tabLayout.getTabCount());
        viewPager.setAdapter(pagerMainAdapter);
        viewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(tabLayout));
        if (bundle != null) {

viewPager.setCurrentItem(TAB_FROM_PUSH_NOTIFICATION);
        } else {
            viewPager.setCurrentItem(1);
        }
        tabLayout.setOnTabSelectedListener(new
TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab)
{

viewPager.setCurrentItem(tab.getPosition());
                Log.e("Print Debug 1 ----------> ",
String.valueOf(tab.getPosition()));

                System.out.println("User ID --------> " +
session.getUserId() + " | " + session.getUserId());

                getTabIndex(tab.getPosition());
            }

            @Override
            public void onTabUnselected(TabLayout.Tab
tab) {
                Log.e("Print Debug 2 ----------> ",
String.valueOf(tab.getPosition()));

            }

            @Override
            public void onTabReselected(TabLayout.Tab
tab) {
                Log.e("Print Debug 3 ----------> ",
String.valueOf(tab.getPosition()));

            }
        });
```

```
        //set profile photo in toobar
        toolbarProfileIcon();

        //check gps actived
        checkGPS();

        // make notification sticky
        //notificatoinLastLocation();

        checkConnection();
    }

    public void checkGPS() {
        if(ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
                    new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    1);
        } else {
            //start service
            startService();
        }

        gps = new GPSTracker(this);
    }

    public void startService() {
        //startService(new Intent(getBaseContext(),
LocationTranckerService.class));
        //startService(new Intent(getBaseContext(),
SystemService.class));
        startService(new Intent(getBaseContext(),
LocationService.class));
    }

    private void checkConnection(){
        final SweetAlertDialog pDialog = new
SweetAlertDialog(this, SweetAlertDialog.PROGRESS_TYPE);

pDialog.getProgressHelper().setBarColor(Color.parseColor(
"#A5DC86"));
        pDialog.setTitleText("Loading");
        pDialog.setCancelable(false);
        pDialog.show();
```

```
        DatabaseReference connectedRef =
FirebaseDatabase.getInstance().getReference(".info/connec
ted");
        connectedRef.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot
snapshot) {
                boolean connected =
snapshot.getValue(Boolean.class);
                if (connected) {
                    System.out.println("connected");
                    pDialog.dismiss();
                } else {
                    System.out.println("not connected");

pDialog.changeAlertType(SweetAlertDialog.ERROR_TYPE);
                }
            }

            @Override
            public void onCancelled(DatabaseError error)
{
                System.err.println("Listener was
cancelled");
            }
        });
    }

    private void toolbarProfileIcon() {

        DB.firebase()
                .child("users")
                .child(session.getUserId())
                .child("profilePicture")
                .addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
dataSnapshot) {

                        if (dataSnapshot.exists()) {
                            String SRC = (String)
dataSnapshot.getValue();
                            System.out.println("PROFILE
PICTURE : " + SRC);

                            Glide.with(MainActivity.this)
```

```
                        .load(SRC)
.placeholder(R.drawable.ic_user_64dp)
                        .dontAnimate()
.into(circleImageView);
                    }
                }

                @Override
                public void onCancelled(DatabaseError
databaseError) {

                }
            });

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action
bar will
        // automatically handle clicks on the Home/Up
button, so long
        // as you specify a parent activity in
AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_profile_user) {
            Intent i = new Intent(this,
ProfileUserActivity.class);
            i.putExtra("USER_ID", session.getUserId());
            startActivity(i);
            return true;
        }


        return super.onOptionsItemSelected(item);
    }

    public void getTabIndex(int index) {
        switch (index) {
            case 0:
                changeTitle(R.id.toolbar, Title.CHAT);

fab.setVisibility(View.GONE);
```

```
                break;
            case 1:
                changeTitle(R.id.toolbar, Title.FRIENDS);

setFab(R.drawable.ic_person_add_white_18dp, new
Intent(MainActivity.this, FriendInviteActivity.class));
                break;
            case 2:
                changeTitle(R.id.toolbar,
Title.NOTIFICATIONS);

                fab.setVisibility(View.GONE);
                break;
        }
    }

    private void setFab(int id, final Intent intent) {
        fab.setVisibility(View.VISIBLE);
        fab.setImageResource(id);
        fab.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                if (intent != null) {
                    startActivity(intent);

Config.removeTransition(MainActivity.this, intent);
                }
            }
        });
    }

    public void systemBarTint() {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.KITKAT) {
            setTranslucentStatus(true);
        }

        SystemBarTintManager tintManager = new
SystemBarTintManager(this);
        tintManager.setStatusBarTintEnabled(true);

tintManager.setStatusBarTintResource(R.color.colorPrimary
Dark);
    }

    @TargetApi(19)
```

```
    private void setTranslucentStatus(boolean on) {
        Window win = getWindow();
        WindowManager.LayoutParams winParams =
win.getAttributes();
        final int bits =
WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS;
        if (on) {
            winParams.flags |= bits;
        } else {
            winParams.flags &= ~bits;
        }
        win.setAttributes(winParams);
    }

    private void sessoinStart() {
        session = new Session(this);
        AccountKit.getCurrentAccount(new
AccountKitCallback<Account>() {
            @Override
            public void onSuccess(Account account) {
                String accountKitId = account.getId();
                PhoneNumber phoneNumber =
account.getPhoneNumber();
                String phoneNumberString =
phoneNumber.toString();

                session.setUserId(accountKitId);

session.setPhoneNumber(phoneNumberString);

            }

            @Override
            public void onError(final AccountKitError
error) {
                // Handle Error
            }
        });

    }

    public class BitmapAsync extends AsyncTask<Void,
Void, Bitmap> {

        @Override
        protected Bitmap doInBackground(Void... params) {

            try {
```

```
                URL url = new URL(SRC);
                HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
                connection.setDoInput(true);
                connection.connect();
                InputStream input =
connection.getInputStream();
                Bitmap myBitmap =
BitmapFactory.decodeStream(input);
                return getCircleBitmap(myBitmap);
            } catch (IOException e) {
                e.printStackTrace();
                return null;
            }

        }
    }

    private Bitmap getCircleBitmap(Bitmap bitmap) {
        final Bitmap output =
Bitmap.createBitmap(bitmap.getWidth(),
                bitmap.getHeight(),
Bitmap.Config.ARGB_8888);
        final Canvas canvas = new Canvas(output);

        final int color = Color.RED;
        final Paint paint = new Paint();
        final Rect rect = new Rect(0, 0,
bitmap.getWidth(), bitmap.getHeight());
        final RectF rectF = new RectF(rect);

        paint.setAntiAlias(true);
        canvas.drawARGB(0, 0, 0, 0);
        paint.setColor(color);
        canvas.drawOval(rectF, paint);

        paint.setXfermode(new
PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
        canvas.drawBitmap(bitmap, rect, rect, paint);

        bitmap.recycle();

        return output;
    }

    private void makeNotification(Context context, String
content) {
        notificationManager = (NotificationManager)
```

```
getSystemService(NOTIFICATION_SERVICE);
        Intent intent = new Intent(context,
MainActivity.class);

        PendingIntent pendingIntent =
PendingIntent.getActivity(context,
                1, intent,
PendingIntent.FLAG_UPDATE_CURRENT);

        builder = new Notification.Builder(context)
                .setContentTitle("Find Friend's
Location")
                .setContentText(content)
                .setContentIntent(pendingIntent)
                .setSmallIcon(R.mipmap.ic_launcher)

.setLargeIcon(BitmapFactory.decodeResource(getResources()
, R.mipmap.ic_launcher))
        ;

        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.JELLY_BEAN) {
            n = builder.build();
        } else {
            n = builder.getNotification();
        }

        n.flags |= Notification.FLAG_NO_CLEAR |
Notification.FLAG_ONGOING_EVENT;

        notificationManager.notify(1, n);
    }

    public void notificatoinLastLocation() {
        final LocationAddress locationAddress = new
LocationAddress(this);
        DB.firebase()
                .child("users")
                .child(session.getUserId())
                .addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
dataSnapshot) {

                        if (dataSnapshot.exists()) {
                            String lat = (String)
dataSnapshot.child("lastLatitude").getValue();
```

```
                            String lon = (String)
dataSnapshot.child("lastLongitude").getValue();

                            if ((lat != null && lon !=
null) && (!lat.isEmpty() && !lon.isEmpty())) {

locationAddress.setLatitude(Double.parseDouble(lat));

locationAddress.setLongitude(Double.parseDouble(lon));
                                String address =
locationAddress.getAddress() + " " +
locationAddress.getCity() + " " +
locationAddress.getState();


System.out.println("address : " + address);

makeNotification(getApplicationContext(), address);
                                }
                            }
                        }

                        @Override
                        public void onCancelled(DatabaseError
databaseError) {

                        }
                    });
    }
}
```

FriendListFragment.java

```
public class FriendListFragment extends Fragment {

    // Initialization RecyclerView
    private RecyclerView recyclerView;
    private FriendsRecyclerViewAdapter
friendsRecyclerViewAdapter;
    private ArrayList<Friends> friendsArrayList = new
ArrayList<>();

    // Session Manager
    private Session session;

    // Init SwipeRefresh Layout
    private SwipeRefreshLayout swipeRefreshLayout;
```

```
    // Main method on fragment
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view =
inflater.inflate(R.layout.friend_list_fragment, container,
false);

        // Get RecyclerView UI Component
        setComponentUI(view);

        // set Recyclerview
        setRecyclerView();

        // Mengambil data dari database online
        getDataFriends();

        // set Refresh
        onRefresh();

        return view;
    }

    // * Method Get Refresh Data
    private void onRefresh() {

        // Configuration Color on Swipe Refresh Loading
        swipeRefreshLayout.setColorSchemeResources(
                R.color.colorPrimary,
                android.R.color.holo_green_light,
                android.R.color.holo_orange_light,
                android.R.color.holo_red_light);

        swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {

                System.out.println("condition db : " +
DB.isConnect());

                // Check not connection
                //if(DB.isConnect()){
                // Clean List
                friendsArrayList.clear();
                // Get Data From Friend List
                getDataFriends();

friendsRecyclerViewAdapter.notifyDataSetChanged();

                System.out.println("Refresh");
                //}
```

```
                // * Hide Refresh Loading
                hideRefresh();
            }
        });
    } // End Method

    private void hideRefresh() {
        swipeRefreshLayout.setRefreshing(false);
    }

    private void setComponentUI(View view) {
        recyclerView = (RecyclerView)
view.findViewById(R.id.friends_Recyclerview);
        swipeRefreshLayout = (SwipeRefreshLayout)
view.findViewById(R.id.friends_swipeRefresh);

    }

    private void setRecyclerView() {
        // Set Recyclerview
        recyclerView.setLayoutManager(new
LinearLayoutManager(getContext()));
        recyclerView.hasFixedSize();
        recyclerView.setItemAnimator(new DefaultItemAnimator());

        // Set Adapter for RecyclerView
        friendsRecyclerViewAdapter = new
FriendsRecyclerViewAdapter(getContext(), friendsArrayList);
        recyclerView.setAdapter(friendsRecyclerViewAdapter);
    }


    // * Method Get Data Friend Feeds
    private void getDataFriends() {
        friendsArrayList.clear();
        session = new Session(getContext());

        // Firebase Query
        Query friendsQuery = DB.firebase().child("users")
                .child(new Session(getActivity()).getUserId()) //
this is need value for data
                .child("friends")
                .orderByChild("status")
                .equalTo("connected");

        friendsQuery.keepSynced(true);

        friendsQuery.addChildEventListener(new
ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot,
String s) {
                if (dataSnapshot.exists()) {
                    Friends friends =
dataSnapshot.getValue(Friends.class);
```

```
                    Query usersQuery = DB.firebase()
                            .child("users")
                            .child(friends.getFriendId());

                    usersQuery.keepSynced(true);

                    usersQuery.addListenerForSingleValueEvent(new
ValueEventListener() {
                        @Override
                        public void onDataChange(DataSnapshot
dataSnapshot) {
                            if (dataSnapshot.exists()) {

                                String friendId = (String)
dataSnapshot.child("userId").getValue();
                                String friendEmail = (String)
dataSnapshot.child("description").getValue();
                                String friendPhoneNumber =
(String) dataSnapshot.child("phone_number").getValue();
                                String friendName = (String)
dataSnapshot.child("displayName").getValue();
                                String friendProfilePicture =
(String) dataSnapshot.child("profilePicture").getValue();

                                Friends friends = new Friends();
                                friends.setFriendId(friendId);
                                friends.setEmail(friendEmail);

friends.setPhone_number(friendPhoneNumber);

friends.setDisplayName(friendName);

friends.setProfilePicture(friendProfilePicture);
                                friendsArrayList.add(friends);

friendsRecyclerViewAdapter.notifyDataSetChanged();
                            }
                        }

                        @Override
                        public void onCancelled(DatabaseError
databaseError) {

                        }
                    });
                }
            }

            @Override
            public void onChildChanged(DataSnapshot dataSnapshot,
String s) {
                onRefresh();
            }
```

```
            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot)
{

            }

            @Override
            public void onChildMoved(DataSnapshot dataSnapshot,
String s) {

            }

            @Override
            public void onCancelled(DatabaseError databaseError)
{

            }
        });
    } // End Method

    private void dataSample() {

        for (int i = 0; i < 30; i++) {
            Friends friends = new Friends();
            friends.setEmail("taufikhope@gmail.com");
            friends.setPhone_number("085695755197");
            friends.setDisplayName("Taufik Soleh");
            friends.setProfilePicture("");
            friendsArrayList.add(friends);
            friendsRecyclerViewAdapter.notifyDataSetChanged();
        }
    }
}
```

LocationHistoryCalendarActivity.java

```
public class LocationHistoryCalendarActivity extends BaseActivity
        implements
NavigationView.OnNavigationItemSelectedListener,
OnMapReadyCallback, OnDateSelectedListener {

    private String dateStr;

    private ViewPager mViewPager;

    private String dateTime;
    private String USER_ID;

    private GoogleMap googleMap;
    private ArrayList<LatLng> polyLine = new
```

```
ArrayList<>();
    private Marker marker;
    private LocationAddress locationAddress;
    private IconGenerator iconGenerator;
    private Session session;
    private SweetAlertDialog loading;
    private ArrayList<LocationHistory>
listLocationHistory;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.location_history_calendar_activit
y);

        // initialization
        session = new Session(this);
        locationAddress = new LocationAddress(this);
        locationAddress = new LocationAddress(this);
        iconGenerator = new IconGenerator(this);
        loading = new SweetAlertDialog(this,
SweetAlertDialog.PROGRESS_TYPE)
                .setTitleText("Loading...");

        //setup toolbar
        setupToolbarWithUpNav(R.id.toolbar, "Riwayat
Lokasi", R.drawable.ic_keyboard_arrow_left_white_24dp);

        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {

            USER_ID = bundle.getString("USER_ID");
            if (session.getUserId().equals(USER_ID)) {
                USER_ID = session.getUserId();
            }
        }

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
                this, drawer, toolbar,
R.string.navigation_drawer_open,
```

```
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        SupportMapFragment mapFragment =
(SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.history
_google_maps);
        mapFragment.getMapAsync(this);

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

navigationView.setNavigationItemSelectedListener(this);

        View headerView =
navigationView.getHeaderView(0);
        MaterialCalendarView calendarView =
(MaterialCalendarView)
headerView.findViewById(R.id.calendarView);

        calendarView.setOnDateChangedListener(this);
        calendarView.setDateSelected(CalendarDay.today(),
true);


gotoLocationHistoryList(USER_ID,Date.nowFormat("yyyyMMdd"
));
    }

    private void closeNavigation() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
```

```
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the
action bar if it is present.

getMenuInflater().inflate(R.menu.location_history_activit
y_calendar, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action
bar will
        // automatically handle clicks on the Home/Up
button, so long
        // as you specify a parent activity in
AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.history_close) {
            finish();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem
item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == R.id.nav_camera) {
            // Handle the camera action
        } else if (id == R.id.nav_gallery) {

        } else if (id == R.id.nav_slideshow) {

        } else if (id == R.id.nav_manage) {

        } else if (id == R.id.nav_share) {

        } else if (id == R.id.nav_send) {

        }
```

```
            closeNavigation();
            return true;
        }

        @Override
        public void onMapReady(GoogleMap googleMap) {
            this.googleMap = googleMap;
            googleMap.setBuildingsEnabled(true);
            googleMap.setIndoorEnabled(true);
            googleMap.setMyLocationEnabled(true);
            UiSettings setting = googleMap.getUiSettings();
            setting.setMapToolbarEnabled(false);

            //new LoadMapsInBackground(googleMap,
Date.nowFormat("yyyyMMdd")).execute();
            getLocationAsync(Date.nowFormat("yyyyMMdd"));
        }

        private void googleMapMarker(GoogleMap map, double
longitude, double latitude, int number, String address) {

            int Zoom = 17;
            if (number == 1) {

map.moveCamera(CameraUpdateFactory.newLatLngZoom(new
LatLng(longitude, latitude), Zoom));
            }

            // create custom icon bubble maps
            iconGenerator.setStyle(IconGenerator.STYLE_RED);

            marker = map.addMarker(new MarkerOptions()

//.icon(BitmapDescriptorFactory.fromResource(R.drawable.p
in_64dp))

.icon(BitmapDescriptorFactory.fromBitmap(iconGenerator.ma
keIcon(String.valueOf(number))))
                    .title(String.valueOf(address))
                    .anchor(0.5f, 1.0f) // Anchors the marker
on the bottom left
                    .position(new LatLng(longitude,
latitude)));
        }

        private void googleMapMarker(GoogleMap map, double
longitude, double latitude) {
```

```
        int Zoom = 17;
        map.setBuildingsEnabled(true);
        map.setIndoorEnabled(true);
        map.setMyLocationEnabled(true);

map.moveCamera(CameraUpdateFactory.newLatLngZoom(new
LatLng(longitude, latitude), Zoom));

        // create custom icon bubble maps
        iconGenerator.setStyle(IconGenerator.STYLE_RED);

        //You can customize the marker image using images
bundled with
        //your app, or dynamically generated bitmaps.
//
//        marker = map.addMarker(new MarkerOptions()
//
//.icon(BitmapDescriptorFactory.fromResource(R.drawable.p
in_64dp))
//
.icon(BitmapDescriptorFactory.fromBitmap(iconGenerator.ma
keIcon()))
//                .anchor(0.5f, 1.0f) // Anchors the
marker on the bottom left
//                .position(new LatLng(longitude,
latitude)));

        UiSettings setting = map.getUiSettings();
        setting.setMapToolbarEnabled(false);
//        setting.setCompassEnabled(true);
//        setting.setZoomGesturesEnabled(true);
    }

    private void getLocationHistoryAccordingDate(final
GoogleMap googleMap, final String tabTitle) {
        String DATE_LOCATION_HISTORY =
tabTitle.replace("-", "");

        DB.firebase()
                .child("users")
                .child(USER_ID)  // change to USER_ID
                .child("locationHistory")
                .child(DATE_LOCATION_HISTORY)
                .addValueEventListener(new
ValueEventListener() {
                    @Override
```

```
                public void onDataChange(DataSnapshot
dataSnapshot) {
                    System.out.println(dataSnapshot);
                    if (dataSnapshot.exists()) {
                        int number = 0;
                        for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                            LocationHistory data =
snapshot.getValue(LocationHistory.class);

data.setLatitude(data.getLatitude());

data.setLongtitude(data.getLongtitude());


System.out.println(data.getLatitude());

locationAddress.setLatitude(Double.valueOf(data.getLatitu
de()));

locationAddress.setLongitude(Double.valueOf(data.getLongt
itude()));


System.out.println("ADDRESS : " +
locationAddress.getAddress());

                            String address =
locationAddress.getAddress() + " " +
locationAddress.getCity() + " " +
locationAddress.getState();

                            System.out.println("LOG
FRAGMENT LATLING : " + tabTitle + " - " +
data.getLongtitude());

                            LatLng latLing = new
LatLng(Double.valueOf(data.getLatitude()),
Double.valueOf(data.getLongtitude()));
                            polyLine.add(latLing);

                            number++;


googleMapMarker(googleMap,
Double.valueOf(data.getLatitude()),
Double.valueOf(data.getLongtitude()), number, address);
```

```
                        drawPolyLine(googleMap);
                    }

                } else {
                    DB.firebase()
                            .child("users")
                            .child(USER_ID)

.addListenerForSingleValueEvent(new ValueEventListener()
{
                                @Override
                                public void
onDataChange(DataSnapshot dataSnapshot) {
                                    if
(dataSnapshot.exists()) {

                                        String
lat = (String)
dataSnapshot.child("lastLatitude").getValue();
                                        String
lon = (String)
dataSnapshot.child("lastLongitude").getValue();

                                        if
(!lat.isEmpty() && !lon.isEmpty()) {

googleMapMarker(googleMap, Double.valueOf(lat),
Double.valueOf(lon));
                                        }
                                    }
                                }

                                @Override
                                public void
onCancelled(DatabaseError databaseError) {

                                }
                            });


Toast.makeText(LocationHistoryCalendarActivity.this,
"Tidak ada riwayat lokasi", Toast.LENGTH_SHORT).show();
                    }

                    if (loading.isShowing()) {
                        loading.dismiss();
                    }
                }
```

```
                    @Override
                    public void onCancelled(DatabaseError
databaseError) {
                        if (loading.isShowing()) {
                            loading.dismiss();
                        }
                    }
                });
    }

    private void getLocationAsync(final String date) {
        String DATE_LOCATION_HISTORY = date.replace("-",
"");
        listLocationHistory = new ArrayList<>();

        DB.firebase()
                .child("users")
                .child(USER_ID)  // change to USER_ID
                .child("locationHistory")
                .child(DATE_LOCATION_HISTORY)
                .addValueEventListener(new
ValueEventListener() {

                    @Override
                    public void onDataChange(DataSnapshot
dataSnapshot) {
                        System.out.println(dataSnapshot);
                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                                LocationHistory data =
snapshot.getValue(LocationHistory.class);

data.setLatitude(data.getLatitude());

data.setLongtitude(data.getLongtitude());

listLocationHistory.add(data);
                            }


                            new
LoadMapsInBackground(googleMap,
listLocationHistory).execute();
                        } else {
                            DB.firebase()
```

```
                                .child("users")
                                .child(USER_ID)

.addListenerForSingleValueEvent(new ValueEventListener()
{
                                    @Override
                                    public void
onDataChange(DataSnapshot dataSnapshot) {
                                        if
(dataSnapshot.exists()) {
                                            String
lat = (String)
dataSnapshot.child("lastLatitude").getValue();
                                            String
lon = (String)
dataSnapshot.child("lastLongitude").getValue();

                                            if
(!lat.isEmpty() && !lon.isEmpty()) {

googleMapMarker(googleMap, Double.valueOf(lat),
Double.valueOf(lon));
                                            }
                                        }
                                    }

                                    @Override
                                    public void
onCancelled(DatabaseError databaseError) {

                                    }
                                });

Toast.makeText(LocationHistoryCalendarActivity.this,
"Tidak ada riwayat lokasi", Toast.LENGTH_SHORT).show();
                        }
                    }

                    @Override
                    public void onCancelled(DatabaseError
databaseError) {

                    }
                });


    }
```

```
    private void drawPolyLine(GoogleMap map) {
        // Polylines are useful for marking paths and
routes on the map.
        Polyline line = map.addPolyline(new
PolylineOptions().geodesic(true).color(Color.BLUE).width(
10));
        line.setPoints(polyLine);
    }

    @Override
    public void onDateSelected(@NonNull
MaterialCalendarView widget, @NonNull CalendarDay date,
boolean selected) {
        String strMonth, strDate;

        if (date.getMonth() < 10) {
            int mth = date.getMonth() + 1;
            strMonth = String.valueOf(0 + "" + mth);
        } else {
            int mth = date.getMonth() + 1;
            strMonth = String.valueOf(mth);
        }

        if (date.getDay() < 10) {
            strDate = String.valueOf(0 + "" +
date.getDay());
        } else {
            strDate = String.valueOf(date.getDay());
        }

        dateStr = date.getYear() + "-" + strMonth + "-" +
strDate;
        closeNavigation();

        polyLine.clear();
        googleMap.clear();

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {

                // get location according date
                getLocationAsync(dateStr);

            }
        }, 1000);
```

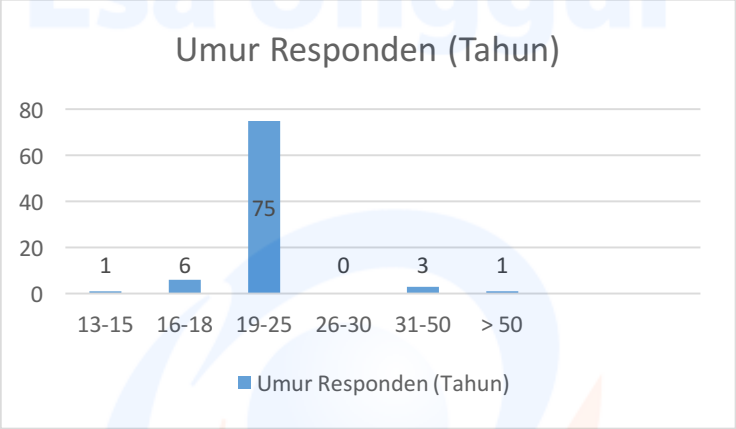
```
        gotoLocationHistoryList(USER_ID,dateStr);
        System.out.println(date.getYear() + strMonth +
strDate);
    }

    public void gotoLocationHistoryList(final String
USER_ID, final String DATE) {
        FloatingActionButton FAB = (FloatingActionButton)
findViewById(R.id.fabHistoryList);
        FAB.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(getApplicationContext(),
LocationHistoryListActivity.class);
                intent.putExtra("USER_ID", USER_ID);
                intent.putExtra("DATE", DATE);
                startActivity(intent);
            }
        });
    }


    private class LoadMapsInBackground extends
AsyncTask<String, String, String> {
        private GoogleMap googleMap;
        private ArrayList<LocationHistory>
listLocationHistory;
        private Handler handler = new Handler();
        private Thread thread = new Thread();

        public LoadMapsInBackground(GoogleMap googleMap,
ArrayList listLocationHistory) {
            this.googleMap = googleMap;
            this.listLocationHistory =
listLocationHistory;
        }

        @Override
        protected String doInBackground(String... params)
{
            handler.post(new Runnable() {
                @Override
                public void run() {
                    for (int i = 0; i <
listLocationHistory.size(); i++) {
```

```
                        googleMapMarker(googleMap,
Double.valueOf(listLocationHistory.get(i).getLatitude()),
Double.valueOf(listLocationHistory.get(i).getLongtitude()
), i + 1, "");
//
locationAddress.setLatitude(Double.valueOf(listLocationHi
story.get(i).getLatitude()));
//
locationAddress.setLongitude(Double.valueOf(listLocationH
istory.get(i).getLongtitude()));
//                          String address =
locationAddress.getAddress() + " " +
locationAddress.getCity() + " " +
locationAddress.getState();
                    }

                    for (int i = 0; i <
listLocationHistory.size(); i++) {
                        LatLng latLing = new
LatLng(Double.valueOf(listLocationHistory.get(i).getLatit
ude()),
Double.valueOf(listLocationHistory.get(i).getLongtitude()
));
                        polyLine.add(latLing);
                        drawPolyLine(googleMap);
                    }
                }
            });

            return null;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected void onProgressUpdate(String... values)
{
            super.onProgressUpdate(values);

        }
```

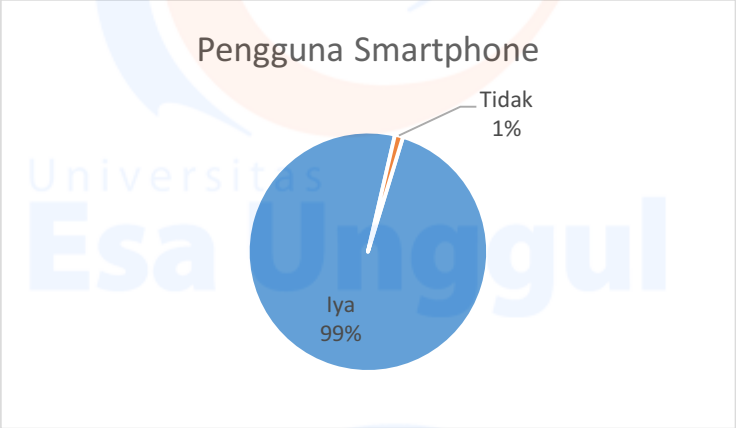
```
    }
}
```

Lampiran 4. Hasil Kuesioner

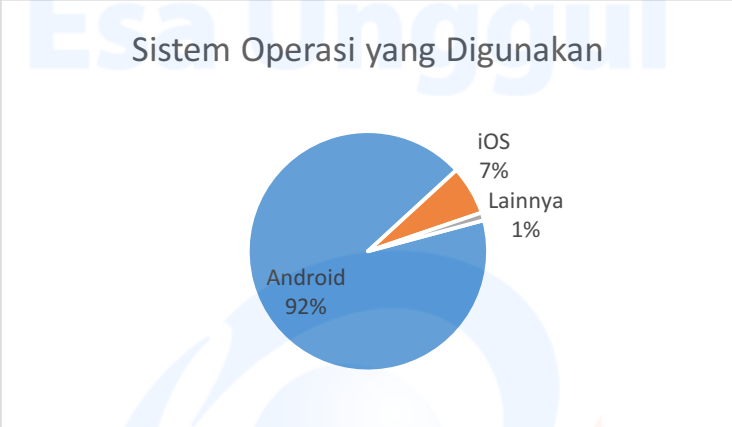
1. Berapa umur anda saat ini?



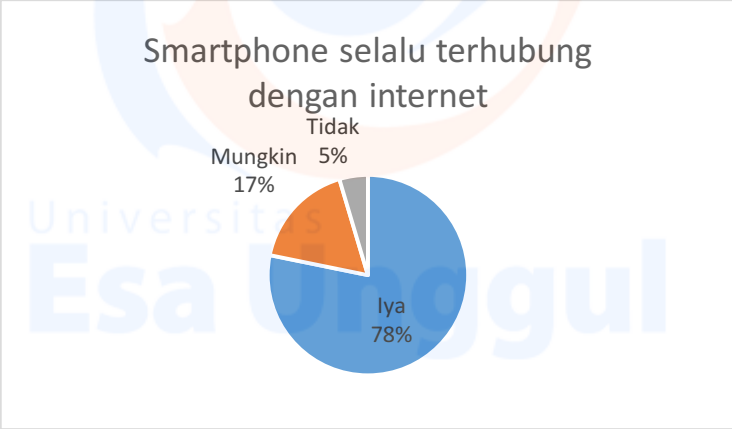
2. Apakah anda mempunyai *smartphone*?



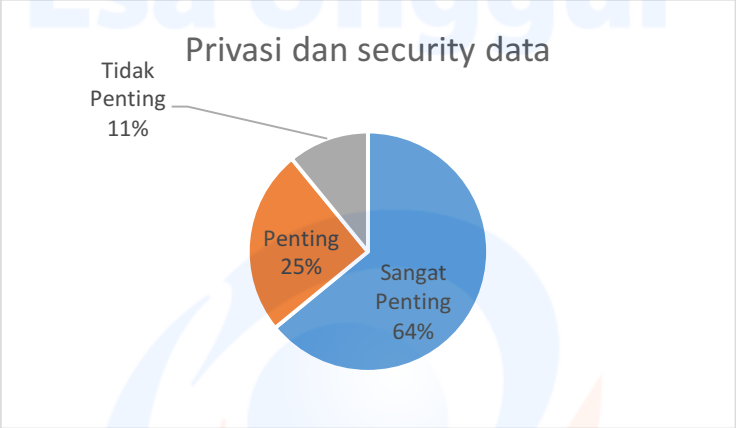
3. Apa sistem operasi *smartphone* anda?



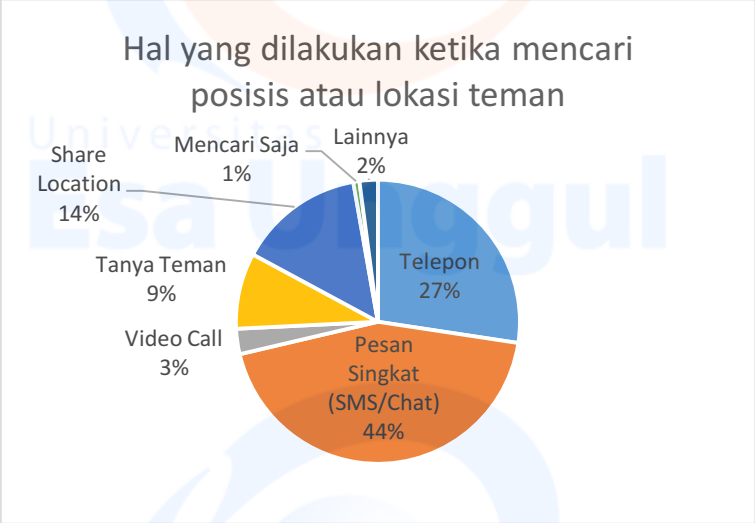
4. Apakah *smartphone* anda selalu terhubung dengan internet?



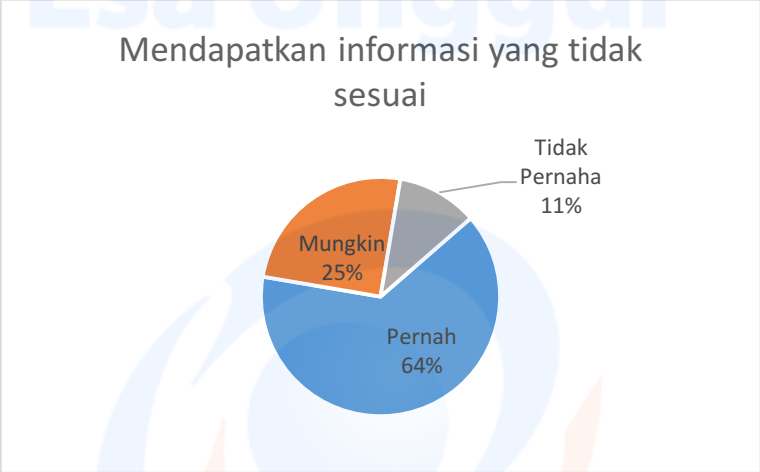
5. Apakah privasi dan *security* data pada aplikasi itu penting bagi anda?



6. Apakah yang anda lakukan saat mencari lokasi/poisis keberadaan teman?



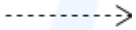
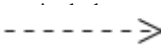
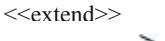
7. Apakah anda pernah mendapatkan informasi yang tidak sesuai saat mencari lokasi atau posisi teman?



Lampiran 5. Simbol – Simbol Pada Diagram UML

Lampiran 5.1 Tabel *Use Case Diagram*

Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya.
4		*Include*	Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri.
5		*Extend*	Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya

			atau sebagai syarat dijalankan *use case* ini.
6		*Association*	Komunikasi antar aktor dan *use case* yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Lampiran 5.2 Tabel *Activity Diagram*

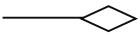
Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Aktivitas*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Decision /percabangan*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Lampiran 5.3 *Class Diagram*

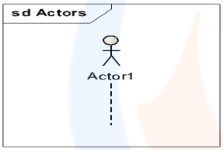
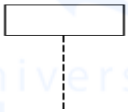
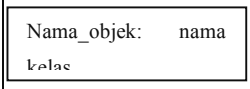
Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

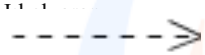
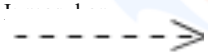
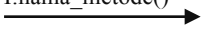
NO	GAMBAR	NAMA	KETERANGAN
1		*Association*	Hubungan antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
5		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
6		*Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah *use case* dimana fungsi yang satu

			adalah fungsi yang lebih umum dari yang lainnya.
7		*Agregasi/aggregation*	Hubungan antar kelas dengan makna semua-bagian *(whole part)*

Lampiran 5.4 *Sequence Diagram*

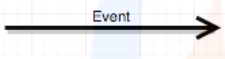
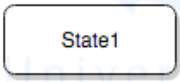
Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
2		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
3		*Objek*	Menyatakan objek yang berinteraksi oleh pesan.

4		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
5		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
6		*Pesan tipe return*	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
7		*Pesan tipe send*	Menyatakan bahwa suatu objek mengirim data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8		*Pesan tipe call*	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.

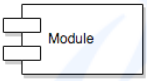
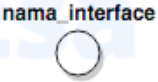
Lampiran 5.5 *Statechart Diagram*

Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Start* / Status Awal *(Initial State)*	*Start* atau *initial state* adalah *state* atau keadaan awal pada saat sistem mulai hidup
2		*End* / Status Akhir (*Final State)*	*End* atau *Final state* adalah *state* keadaan akhir dari daur hidup suatu sistem
3		*Event*	Event adalah kegiatan yang menyebabkan berubahnya status mesin.
4		*State*	*State* atau status adalah keadaan sistem pada waktu tertentu. *State* dapat berubah jika ada *event* tertentu yang memicu perubahan tersebut.

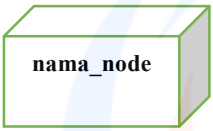
Lampiran 5.6 *Component Diagram*

Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
2		Komponen	Komponen sistem
3		Kebergantungan/ *dependency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
4		Antarmu*ka*/ *interface*	Sama dengan konsep interface pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
5		Link	Relasi antar komponen

Lampiran 5.7 *Deployment Diagram*

Sumber : Rekayasa Perangkat Lunak (A.S Rosa dan M. Shalahudin, 2016)

NO	GAMBAR	NAMA	KETERANGAN
1		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih *node*
2		*Node*	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan
3		*Kebergantungan / dependency*	Kebergantungan antar *node*, arah panah mengarahkan pada *node* yang dipakai
4		*Link*	Relasi antar *node*

Lampiran 6. Rancangan Basis Data

Lampiran 6.1 Tabel *Users*

Primary Key : *user_id*

Foreign Key : -

Tabel 7. Tabel *Users*

Field	**Tipe Data**	**Ukuran**	**Keterangan**
user_id	*varchar*	50	id user
username	*varchar*	50	username
display_name	*varchar*	100	nama tampilan
email	*varchar*	100	email
password	*varchar*	50	kata sandi
phone_number	*varchar*	15	nomor handphone
brith_date	*varchar*	30	tanggal lahir
address	*varchar*	100	alamat
status	*varchar*	30	status
register_date	*varchar*	30	tanggal pendaftaran
profile_picture	*varchar*	100	gambar profile
gender	*varchar*	15	jenis kelamin
register_method	*varchar*	30	metode pendaftaran

Lampiran 6.2 Tabel *Location History*

Primary Key : *location_history_id*

Foreign Key : *user_id*

Tabel 8. Tabel *Location History*

Field	**Tipe Data**	**Ukuran**	**Keterangan**
location_hisotry_id	*varchar*	30	id lokasi
user_id	*varchar*	30	Id pengguna
latitude	*varchar*	50	garis lintang/*horizontal*
longitude	*varchar*	50	garis bujur/*vertical*
datetime	*varchar*	30	tanggal penyimpanan lokasi
location_name	*varchar*	100	nama lokasi
city	*varchar*	100	kota
province	*varchar*	50	provinsi
address	*varchar*	100	alamat lokasi
country	*varchar*	50	negara

Lampiran 6.3 Tabel *Friends*

Primary Key : friend_id

Foreign Key : user_id

Tabel 9. Tabel *Friends*

Field	**Tipe Data**	**Ukuran**	**Keterangan**
friend_id	*varchar*	50	*id use*r (teman)
user_id	*varchar*	50	*Id user* (pengguna akun)
username	*varchar*	50	*username*
display_name	*varchar*	100	nama tampilan
email	*varchar*	100	email
phone_number	*varchar*	15	nomor *handphone*
birth_date	*varchar*	30	tanggal lahir
address	*varchar*	100	alamat
status	*varchar*	30	status
register_date	*varchar*	30	tanggal pendaftaran
profile_picture	*varchar*	100	gambar *profile*
friendship_date	*varchar*	30	tanggal pertemanan

Lampiran 6.4 Tabel *User Last Activity*

Primary Key : *last_acitivty_id*

Foreign Key : *user_id*

Tabel 10. Tabel *User Last Activity*

Field	**Tipe Data**	**Ukuran**	**Keterangan**
last_activity_id	*varchar*	30	id aktifitas terakhir
user_id	*varchar*	30	*id user*
last_login_datetime	*varchar*	30	tanggal login terakhir
last_latitude	*varchar*	50	garis lintang terakhir
last_longitude	*varchar*	50	garis bujur terakhir

Lampiran 6.5 Tabel *Notification*

Primary Key : Notification_id

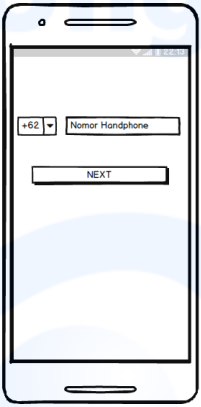
Foreign Key : Notification_sender

Tabel 11. Tabel *Notification*

Field	**Tipe Data**	**Ukuran**	**Keterangan**
notification_id	*varchar*	30	id pemberitahuan
notification_date	*varchar*	30	*tanggal pemberitahuan*
notification_type	*varchar*	30	*jenis pemberitahuan*
notification_description	*text*	-	deskripsi/keterangan
notification_receiver	*varchar*	30	penerima
notification_sender	*varchar*	30	pengirim
notification_status	*varchar*	30	status pemberitahuan

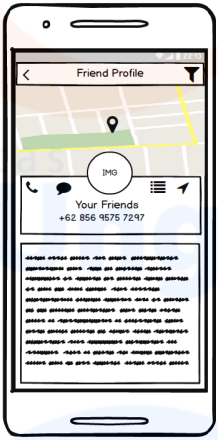
Lampiran 7. Perancangan *User Interface*

Lampiran 7.1 *Login*



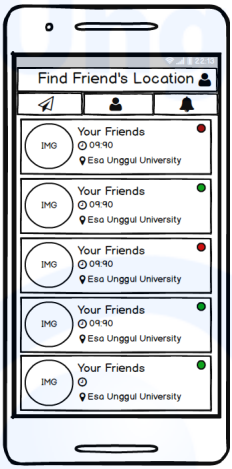
Gambar L7-1 *User Interface Login*

Lampiran 7.2 *Profile* Pengguna dan *Profile* Teman



Gambar L7-2 *User Interface Profile* Pengguna dan Teman

Lampiran 7.3 *List* Teman, *Chat* dan Pemberitahuan



Gambar L7-3 *User Interface List* Teman, *Chat* dan Pemberitahuan

Lampiran 7.4 Riwayat Lokasi Pengguna dan Teman

Gambar L7-4 *User Interface* Riwayat Lokasi Pengguna dan Teman

Lampiran 8. Tabel Pengujian Aplikasi dengan *Blackbox Testing*

No	Rancangan Proses	Hasil yang diharapkan	Hasil Pengujian	Keterangan
1	Klik *Icon Launcher* Aplikasi	- Jika belum *login* menampilkan halaman *Login*, - Jika telah *login* menampilkan halaman utama	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-1
2	Login dengan nomor *handphone*	Mendapatkan Kode Verifikasi melalui SMS	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-2
3	Mendapatkan SMS Kode Verifikasi	Menerima SMS Berisi Kode Verifikasi yang dikirim ke nomor handphone yang dimasukan	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-3
3	Memasukan kode verifikasi	-Jika berhasil masuk ke halaman utama - Jika gagal akan menampilkan pesan kesalahan	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-4
4	Klik **tidak mendapatkan kode**	Mendapatkan kode verifikasi yang berbeda	Sukses	Dapat dilihat di Lampiran

				2 Gambar L2-4
5	Klik **kirim SMS lagi**	Memasukan ulang nomor *handphone*	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-5
6	Klik *item* di *list* teman	Tampil halaman *profile* teman	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6
7	Klik *Tab Icon Friend*	Tampil halaman *list friend*s	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6
8	Klik *Tab Icon Notifikasi*	Tampil halaman pemberitahuan/*Notificati ons*	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-14
9	Klik *Tab Icon Chat*	Tampil halaman *Pesan*	Sukses	
10	Klik *Icon* Foto *Profile* Pengguna	Tampil halaman *profile* Pengguna	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-7
11	Klik *Icon* Hubungkan Teman	Tampil halaman *Hubungkan Teman*	Sukses	Dapat dilihat di Lampiran

				2 Gambar L2-6
13	Klik *icon Call* di *profile* teman	Melakukan panggilan telepon	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-7
14	Klik *icon chat* di *profile* teman	Menampilkan halaman percakapan dengan teman	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-7
15	Klik tombol *send* pada halaman percakapan	Mengirimkan pesan pada teman	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-9
16	Klik *icon* riwayat lokasi	Menampilkan halaman riwayat lokasi berupa peta	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6
	Klik *icon list* riwayat lokasi	Menampilkan halaman berupa list data riwayat lokasi	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-12
	Klik dan pilih tanggal pada kalender riwayat lokasi	Menampilkan kalender, lalu menampilkan data riwayat lokasi yang sesuai dengan tanggal	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-10

17	Klik *icon list block*	Menampilkan *list* teman yang diblok	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6
18	Klik foto *profile*	Menampilkan foto profile	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6
19	Klik tombol *edit* foto *profile*	Memilih foto yang tersedia di handphone	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-8
21	Klik identitias pengguna dan edit data pengguna	- Menampilkan *pop up form edit* identitas -mengubah identitas pengguna	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-15
22	Klik pemberitahuan yang diterima	Menampilkan *detail* pemberitahuan	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-14
23	Klik permintaan pertemanan	- Muncul *pop up* konfirmasi pertemanan	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-14

26	Klik *item* pada *list chat*	Menampilkan halaman percakapan dengan teman	Sukses	Dapat dilihat di Lampiran 2 Gambar L2-6

Lampiran 9. Kuesioner Kepuasan Pengguna

Keterangan pilihan jawaban : **SS** : Sangat Setuju, **S** : Setuju, **RR** : Ragu-Ragu, **TS**: Tidak Setuju, **STS**: Sangat Tidak Setuju						
No.	**Petanyaan**	**SS**	**S**	**RR**	**TS**	**STS**
1	Apakah tampilan, menu dan informasi yang disuguhkan dalam aplikasi ini mudah untuk dipahami?	20%	60%	10%	10%	0%
2	Apakah dengan adanya fitur melihat lokasi dan riwayat lokasi, anda lebih mudah mengetahui posisi teman dengan cepat?	70%	20%	10%	0%	0%
3	Apakah dengan adanya fitur *chating,* anda lebih mudah menghubungi teman?	10%	80%	10%	0%	0%
4	Apakah dengan aplikasi ini dapat membantu orang tua dalam mengawasi dan memantau aktifitas anak diluar rumah?	40%	50%	10%	0%	0%
5	Apakah informasi lokasi teman yang diberikan sudah akurat?	0%	60%	20%	20%	0%
6	Apakah fitur hubungkan teman,block teman dan notifikasi dapat menjaga privasi pengguna?	20%	50%	20%	10%	0%

Berdasarkan hasil kuesioner di atas, dapat disimpulkan beberapa hal sebagai berikut :

- Tampilan, menu dan informasi yang disuguhkan mudah dipahami. Hal ini ditunjukan dari 20% responden menjawab sangat setuju, 60% responden menjawab setuju, 10% responden menjawab ragu-ragu, dan 10% responden lainnya menjawab tidak setuju.
- Dengan adanya fitur melihat lokasi dan riwayat lokasi, pengguna merasa sangat lebih mudah mengetahui posisi teman dengan cepat. Hal ini dibuktikan oleh 70% responden menjawab sangat setuju, dan 30% responden lainnya menjawab setuju.
- Fitur *chating* mempermudah pengguna dalam menghubungi teman. Hal ini ditunjang oleh 10% responden menjawab sangat setuju, 80% responden menjawab setuju, dan 10% responden lainnya menjawab ragu-ragu.
- Orang tua dapat terbantu dalam mengawasi dan memantau aktifitas anak diluar rumah. Hal ini dibuktikan dengan 40% responden menjawab sangat setuju, 50% responden menjawab setuju, dan 10% responden menjawab ragu-ragu.
- Informasi lokasi teman yang ditampilkan akurat. Hal ini ditunjukan dari 0% responden menjawan sangat setuju, 60% responden menjawab setuju, 20% responden menjawab ragu-ragu, dan 20% responden lainnya menjawab tidak setuju.
- Dengan adanya fitur hubungkan teman, block teman dan notifikasi, pengguna merasa privasinya tetap terjaga. Hal ini dibuktikan dengan 20% responden menjawab sangat setuju, 50% responden menjawab setuju, 20% responden menjawab ragu-ragu, dan 10% responden lainnya menjawab tidak setuju.