BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Sebelum memulai penelitian, ditentukan dahulu metodologi penelitian yang akan digunakan. Dalam penelitian ini, metodologi penelitan yang dilakukan adalah sebagai berikut.

1. Telaah Literatur

Telaah literatur dilakukan guna memahami konsep dari metode *push notification* dan Firebase untuk diimplementasikan pada aplikasi notifikasi insiden berbasis Android. Pembelajaran dilakukan dengan cara membaca referensi dari berbagai sumber seperti skripsi, makalah, jurnal serta berbagai sumber lainnya baik *online* maupun media cetak. Tindakan ini dilakukan sepanjang masa penelitian, perancangan, dan pembangunan aplikasi.

2. Perancangan Aplikasi

Perancangan aplikasi dimulai dengan merancang alur kerja dari aplikasi notifikasi insiden ini. Alur kerja aplikasi digambarkan dengan menggunakan diagram alir (*flowchart*). Kemudian, dilakukan perancangan untuk perpindahan data yang terjadi di aplikasi notifikasi insiden yang dibuat dalam bentuk *Data Flow Diagram* (DFD). Hubungan antara satu tabel dengan tabel lain digambarkan dengan *Entity Relationship Diagram* (ERD). Struktur *database* yang digunakan dalam penelitian ini disesuaikan dengan data yang dibutuhkan. Dari sisi tampilan, dibuatlah rancangan tampilan aplikasi yang akan digunakan sehingga pengguna dapat melihat data mengenai insiden baik yang sedang terjadi maupun sudah terjadi.

3. Pemrograman Aplikasi

Setelah hasil rancangan aplikasi didapatkan, dilakukanlah pemrograman aplikasi. Tampilan aplikasi yang dibangun sesuai dengan rancangan yang telah dibuat pada tahap sebelumnya. Pengimplementasian metode *push notification* pada aplikasi notifikasi insiden berbasis Android dilakukan untuk memberikan notifikasi kepada rekan-rekan ketika terdapat rekan yang mengalami suatu insiden. Aplikasi notifikasi insiden akan dibuat dengan menggunakan bahasa pemrograman Java, PHP, dan MySQL.

4. Pengujian

Pada tahap pengujian akan dilakukan metode untuk mendapatkan *click ratio* dan *click time* dari notifikasi-notifikasi yang didapatkan oleh responden penelitian. Uji coba aplikasi akan dilakukan selama satu minggu.

5. Evaluasi

Pada tahap ini, data penelitian yang berupa jawaban UEQ, *click ratio* dan *click time* akan dianalisa menjadi bentuk kuantitatif agar dapat dilihat respon dari responden terhadap notifikasi yang didapatkan.

3.1.1 Skenario Pengujian

Skenario dalam melakukan uji coba aplikasi akan dibahas pada subbab ini. Pengujian dilakukan selama satu minggu, dimana pada hari pertama hingga ketiga, rekan-rekan komunitas GoJek Cikarang yang memiliki alat dapat menekan tombol pada alat tersebut ketika terjadi suatu insiden. Hari keempat hingga ketujuh, rekan-rekan yang memiliki alat diinstruksikan untuk menekan tombol pada alat supaya dapat mengirimkan notifikasi buatan. Rekan-rekan yang menerima notifikasi diinstruksikan untuk menekan setiap notifikasi yang didapatkan dengan

menganggap bahwa telah terjadi suatu insiden. Pada hari terakhir, rekan-rekan komunitas GoJek Cikarang diberikan kuesioner untuk mendukung analisa yang dilakukan.

3.1.2 Skenario Evaluasi

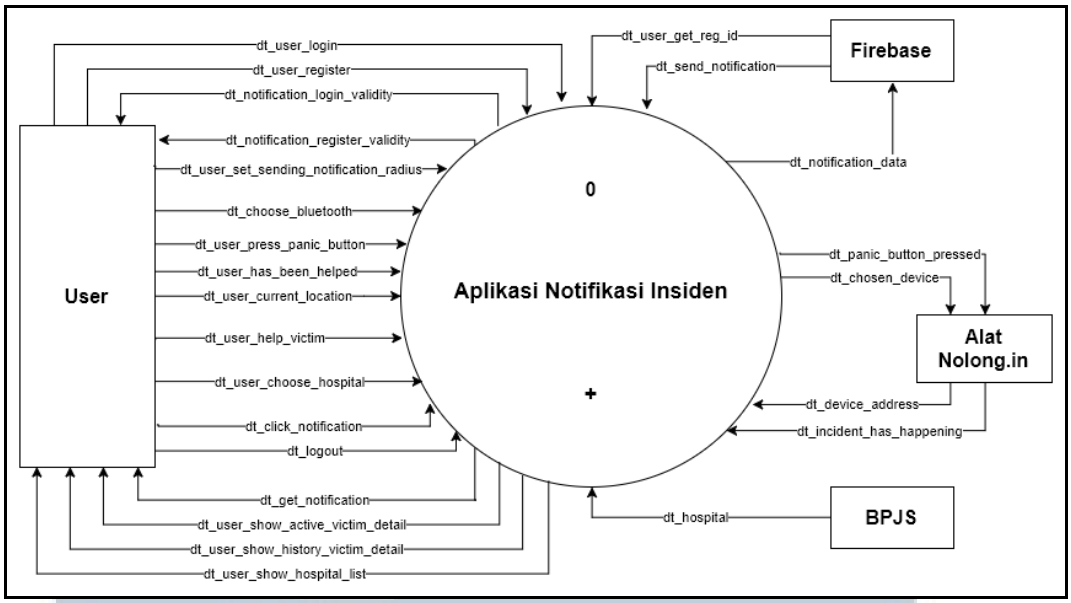
Pada subbab ini akan dibahas mengenai skenario dari evaluasi yang dilakukan. Setelah selesai melakukan pengujian, dilakukanlah evaluasi untuk menghitung *click time* dan *click ratio* berdasarkan data-data notifikasi yang ada. Selain itu, dilakukan juga evaluasi terhadap data-data yang didapatkan melalui kuesioner.

3.2 Perancangan Aplikasi

Perancangan aplikasi menghasilkan beberapa dokumen yang menjelaskan perpindahan data pada aplikasi yang tersaji dalam bentuk *Data Flow* Diagram, alur kerja aplikasi dalam bentuk *flowchart*, hubungan antara satu table dengan yang lainnya dalam bentuk *Entity Relationship* Diagram (ERD), *database schema*, struktur tabel, rancangan antarmuka aplikasi notifikasi insiden, dan arsitektur sistem.

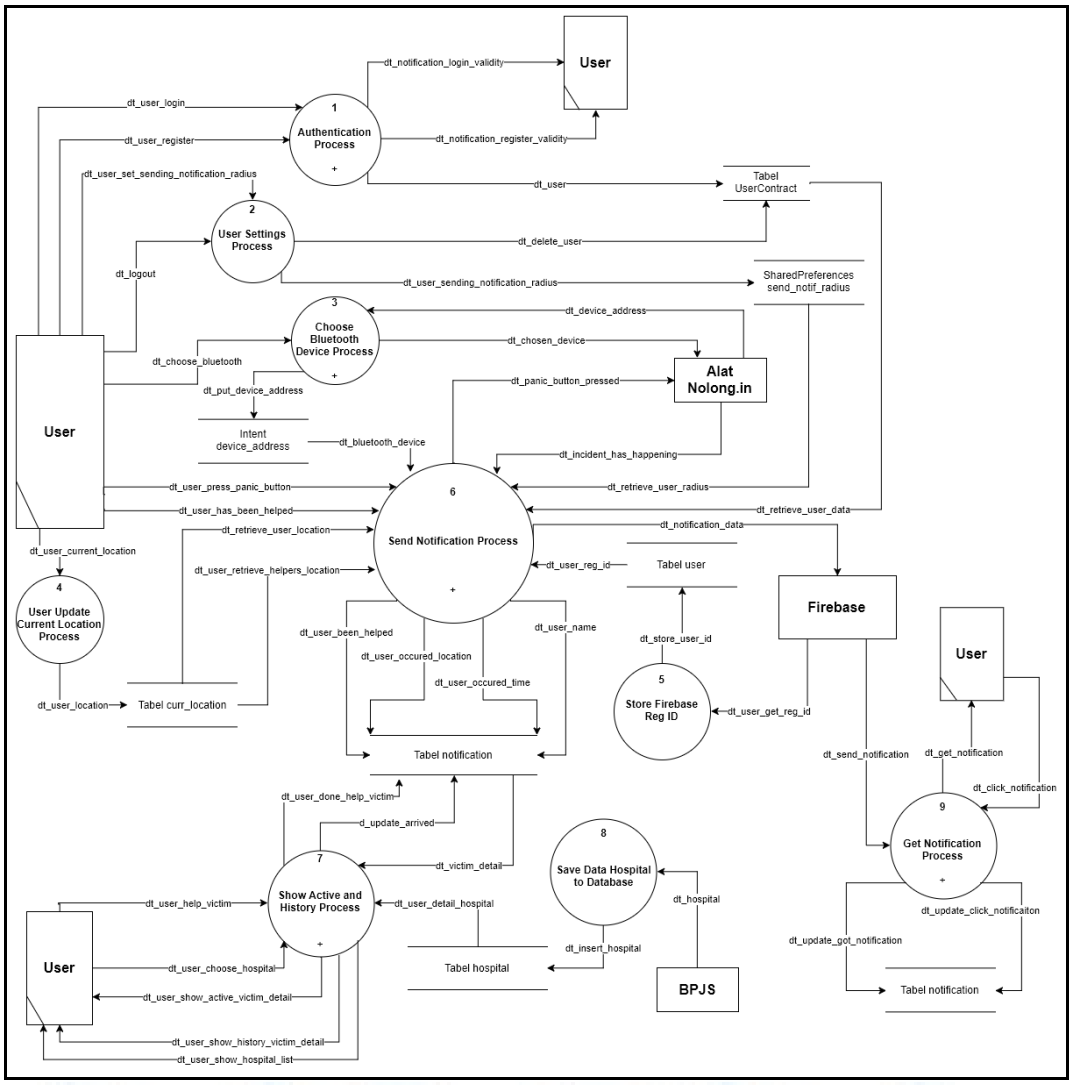
3.2.1 Data Flow Diagram

Alur perpindahan data yang ada dalam aplikasi digambarkan menggunakan *Data Flow Diagram* (DFD).



Gambar 3.1 DFD *Context Diagram* Aplikasi Notifikasi Insiden

Gambar 3.1 merupakan *context diagram* pada aplikasi notifikasi insiden yang menunjukkan garis besar perpindahan data yang terjadi dalam aplikasi. Dalam *context diagram* ini terdapat satu proses utama yaitu Aplikasi Notifikasi Insiden dan empat buah entitas, yaitu *user*, *firebase*, dan BPJS. Entitas *user* memiliki empat data masuk dan delapan data keluar. Entitas *firebase* hanya memiliki dua buah data masuk dan entitas BPJS hanya memiliki satu buah data masuk.



Gambar 3.2 DFD *Level* 1 Aplikasi Notifikasi Insiden

Gambar 3.2 merupakan DFD *level* 1 pada aplikasi notifikasi insiden. Pada diagram ini dapat dilihat terdapat sembilan buah proses yang menunjukkan apa yang terjadi di dalam aplikasi, yaitu *Authentication Process*, *User Settings Process*, *Choose Bluetooth Device Process*, *User Update Current Location Process*, *Store Firebase Reg ID*, *Send Notification Process*, *Show Active and History Process*, *Save Data Hospital to Database*, dan *Get Notification Process*.

Proses *Authentication Process* merupakan proses yang digunakan oleh *user* untuk melakukan autentikasi ke dalam aplikasi. Proses ini menerima *input* dari *user*

yang berupa data *register* dan *login*. Kemudian, akan dilakukan penyimpanan data pada tabel *UserContract*.

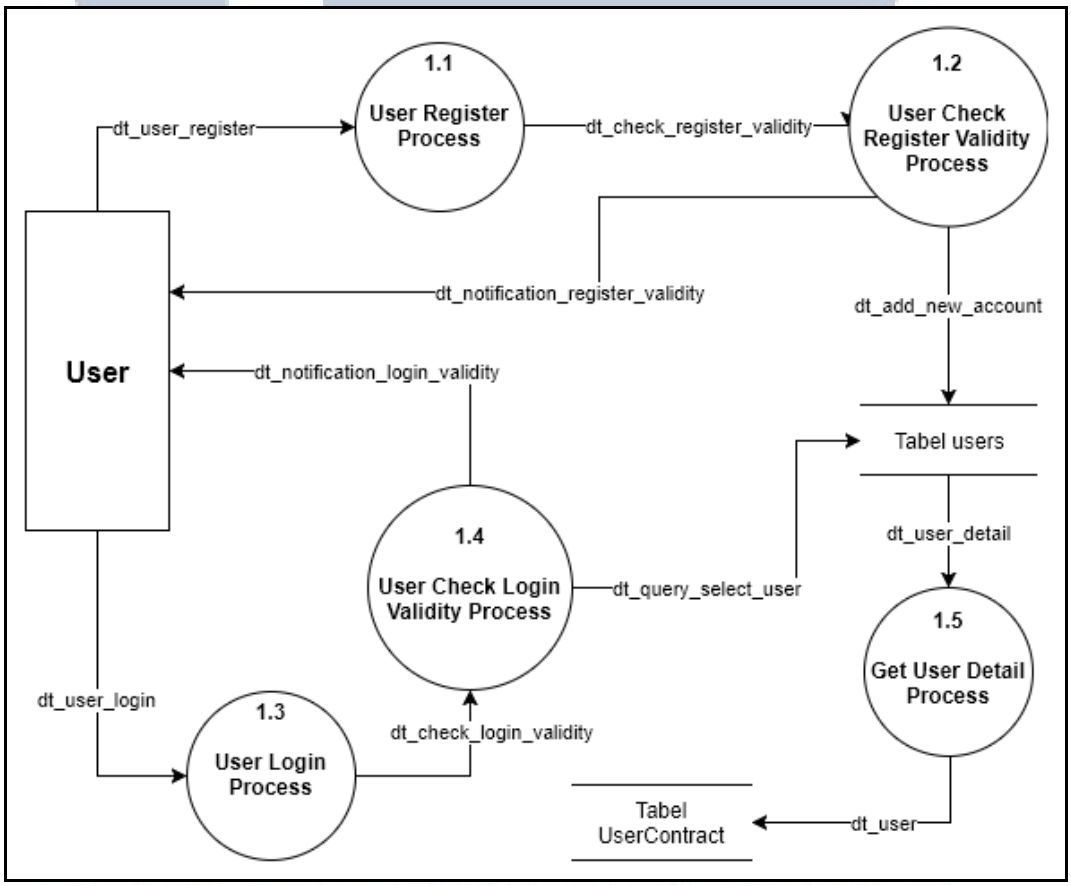
User Settings Process berfungsi untuk mengatur radius dari pengiriman notifikasi. Radius ini digunakan sebagai acuan jarak yang digunakan untuk melakukan pengiriman notifikasi ketika *user* mengalami insiden. Proses ini menerima *input* berupa radius yang sudah di-*set* sebelumnya atau radius *default* yang digunakan oleh aplikasi. Kemudian, data berupa nilai dari radius pengiriman ini akan disimpan dalam *SharedPreferences* untuk dapat digunakan ketika dibutuhkan. Selain itu, pada proses ini *user* juga dapat melakukan *logout* dari aplikasi. Ketika melakukan *logout*, akan dihapus data-datanya dari tabel *UserContract*.

Selanjutnya, terdapat proses *Choose Bluetooth Device Process*. Pada proses ini akan didapatkan *address* dari alat notifikasi insiden yang akan dimasukkan ke dalam *Intent device_address*. Proses ini digunakan untuk mengoneksikan aplikasi dengan alat notifikasi insiden. Tujuan dilakukannya pengoneksian ini adalah agar ketika terjadi suatu insiden, pengguna dapat menekan tombol yang ada pada alat dan notifikasi dapat dikirimkan kepada rekan-rekan. *User Update Current Location Process* berfungsi untuk meng-*update* data berupa lokasi terkini *user*. Data ini disimpan dalam tabel curr_location pada *database* untuk digunakan ketika *user* mengalami insiden.

Store Firebase Reg ID berfungsi untuk menyimpan Firebase *Registration* ID yang didapat dari Firebase ke dalam tabel *users*. Proses *Send Notification Process* digunakan untuk mengirimkan notifikasi kepada rekan-rekan yang berada dalam radius yang sudah ditentukan sebelumnya bahwa *user* atau *admin* sedang

mengalami insiden. Pada proses ini, juga didapatkan notifikasi yang dikirimkan oleh rekan yang mengalami insiden.

Kemudian, *Show Active and History Process* digunakan untuk melihat *list* yang berisi *detail* rekan yang sedang mengalami insiden maupun rekan yang pernah dibantu oleh *user*. Proses *Save Data Hospital to Database* merupakan proses untuk menyimpan data-data rumah sakit yang bekerja sama dengan BPJS dan didapatkan pada *website* resmi BPJS Kesehatan, yaitu https://faskes.bpjs-kesehatan.go.id/aplicares/#/app/dashboard. Proses ini dilakukan secara manual dengan memasukkannya ke dalam *database*.

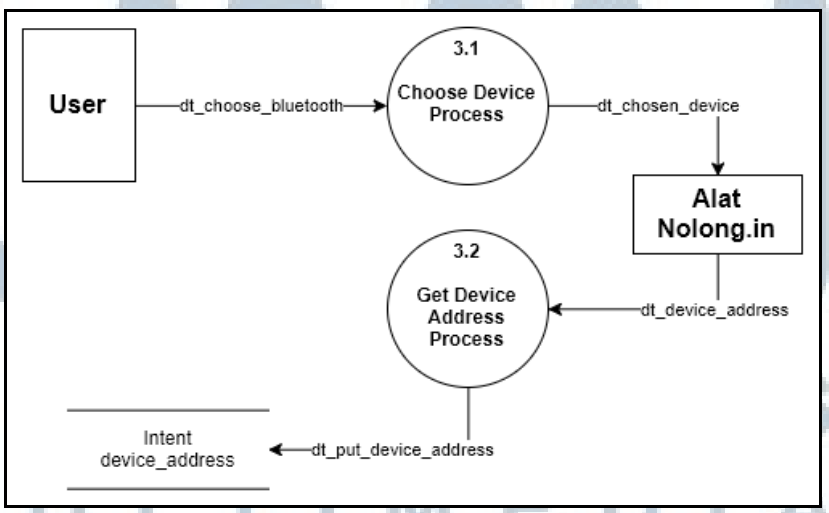


Gambar 3.3 DFD *Level* 2 Pada *Authentication Process*

Gambar 3.3 merupakan DFD *level* 2 pada turunan proses *Authentication Process* dari Gambar 3.2. Proses ini digunakan oleh *user* untuk melakukan

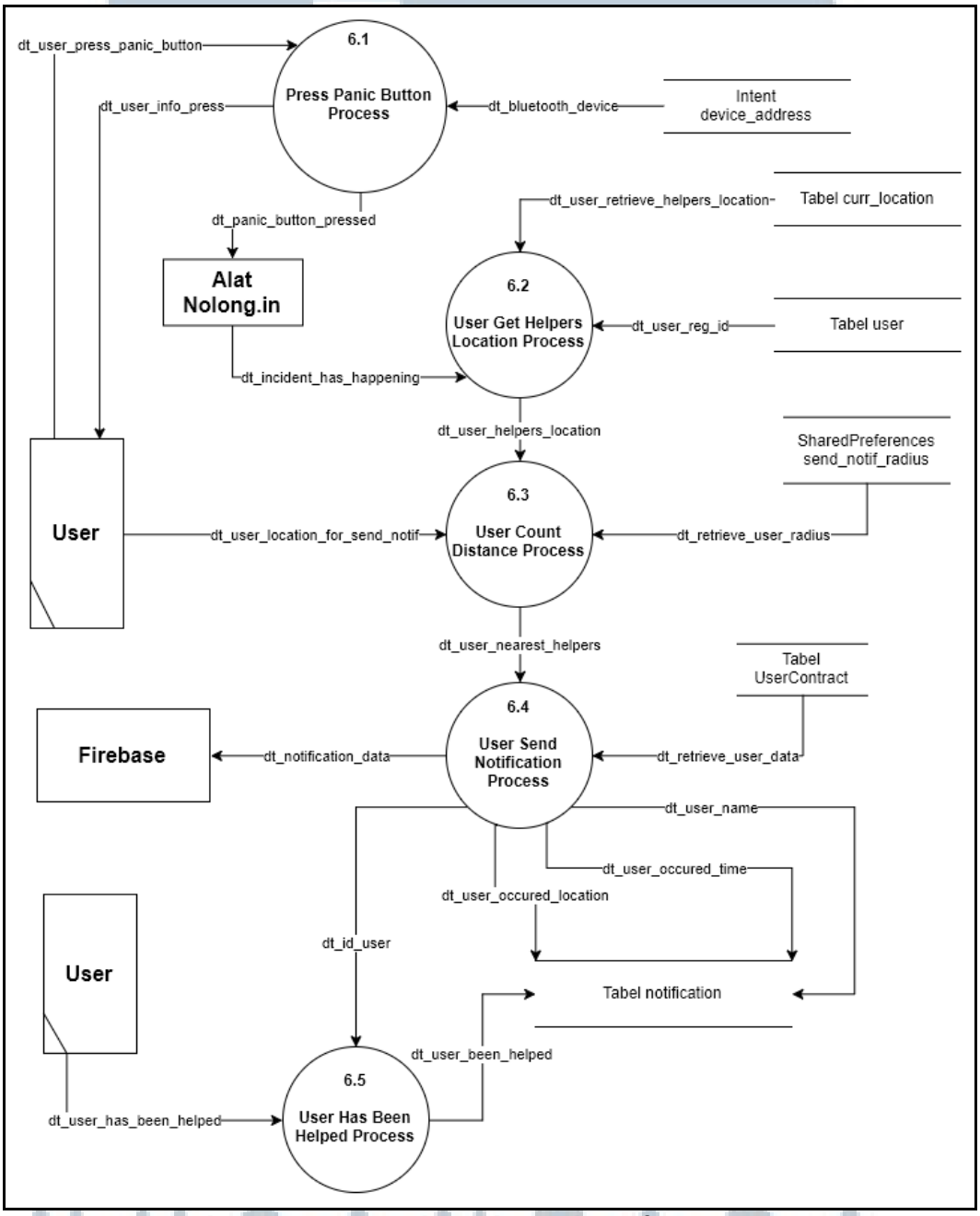
registrasi atau *login* ke dalam aplikasi. *User Register Process* digunakan untuk melakukan registrasi agar mendapatkan akun yang dapat digunakan untuk *login* ke dalam aplikasi. Proses ini menerima *input* berupa data *register* yang dimasukkan *user*. Kemudian, data tersebut akan masuk ke dalam proses *User Check Register Validity Process*, dimana akan dilakukan pengecekan validitas data tersebut. Setelah dicek bahwa data tersebut valid, akan dimasukkan ke dalam *database* pada tabel *users*. Namun, jika tidak, akan mengembalikan notifikasi kepada pengguna bahwa data yang diinput tidak valid.

User Login Process berfungsi untuk melakukan *login* ke dalam aplikasi. Proses ini menerima *input* berupa data *login* yang kemudian akan dicek validitas data tersebut pada proses *User Check Login Validity Process*. Apabila tidak valid, maka akan mengembalikan notifikasi kepada pengguna yang menyatakan bahwa data yang diinput tersebut tidak valid. Namun, jika valid, akan dilakukan pengambilan data dari *database* berupa detail dari *user* tersebut. Kemudian data tersebut akan dimasukkan ke dalam tabel *UserContract*.



Gambar 3.4 DFD *Level* 2 Pada *Choose Bluetooth Device Process*

Gambar 3.4 merupakan DFD *level* 2 pada turunan proses *Choose Bluetooth Device Process* dari Gambar 3.2. Proses ini digunakan oleh *user* untuk mengoneksikan alat dengan aplikasi dengan jaringan Bluetooth. Proses diawali dengan memilih koneksi Bluetooth dari alat. Kemudian, *address* yang didapat dari alat, dimasukkan ke dalam *Intent*.



Gambar 3.5 DFD *Level* 2 Pada *Send Notification Process*

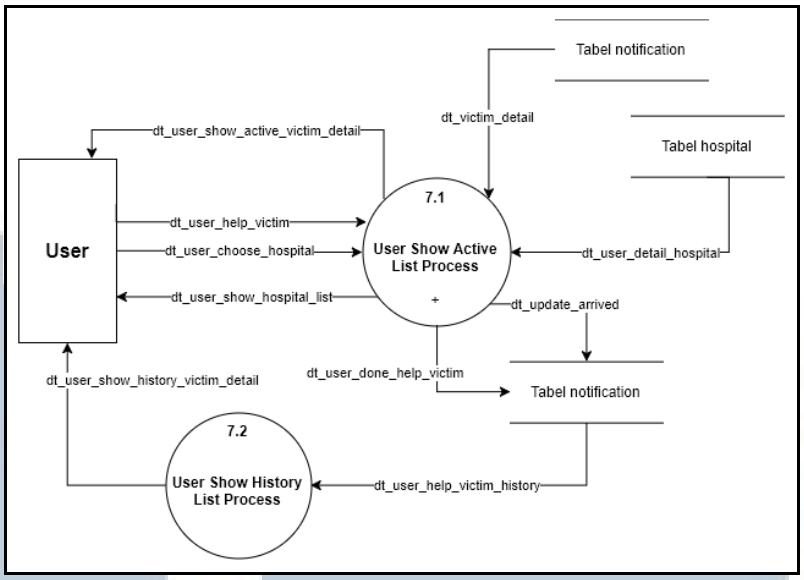
Gambar 3.5 merupakan DFD *Level* 2 pada turunan proses *Send Notification Process* dari Gambar 3.2. Pada proses ini terdapat lima buah sub proses, yaitu *Press Panic Button Process, User Get Helpers Location Process, User Count Distance Process, User Send Notification Process*, dan *User Has Been Helped Process*.

Proses *Press Panic Button Process* merupakan proses pengiriman data dari alat ke dalam aplikasi ketika tombol pada alat ditekan. Proses ini diawali dengan mendapatkan *address* dari alat. Kemudian, dikirimkan data dari alat ke aplikasi ketika tombol pada alat ditekan.

Selanjutnya, akan dijalankan proses *User Get Helpers Location Process* untuk mendapatkan data berupa lokasi dan *registration* ID dari helpers yang memiliki daerah lokasi yang sama dengan pengguna yang mengalami insiden. Proses *User Count Distance* berfungsi untuk melakukan penghitungan jarak antara pengguna dengan *helpers*. Proses ini akan dijalankan setelah mendapatkan lokasi dari *helpers*.

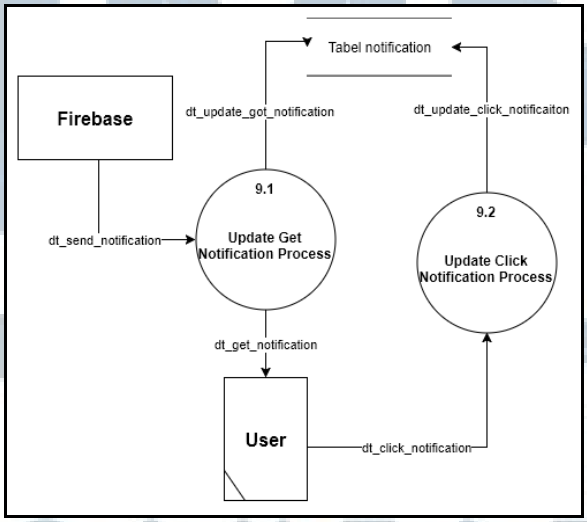
Setelah mendapatkan data *helpers* yang telah dilakukan penghitungan jarak, akan dikirimkan data notifikasi kepada Firebase pada proses *User Send Notification Process* jika hasil dari penghitungan tersebut lebih kecil dari atau sama dengan radius yang sudah ditentukan sebelumnya.

Pada proses *User Has Been Helped Process*, pengguna yang mengalami insiden akan menunggu *helpers* untuk datang dan membantu. Setelah *helpers* selesai membantu, pengguna akan memberi *flag* kepada setiap notifikasi yang dikirimkan agar menjadi selesai dibantu. Hal ini dilakukan agar menyatakan bahwa pengguna yang bersangkutan telah selesai dibantu oleh *helpers*, sehingga *helpers* yang lain tidak perlu datang ke lokasi kejadian untuk membantu.



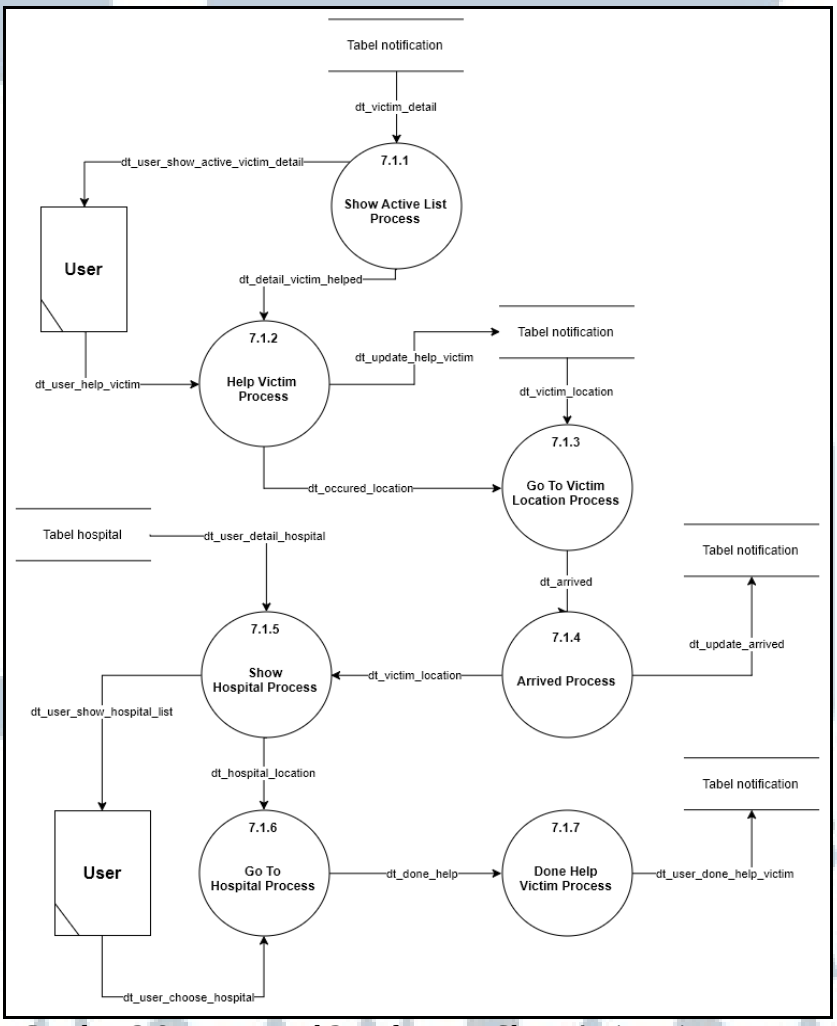
Gambar 3.6 DFD *Level* 2 Pada *Show Active and History Process*

Gambar 3.6 merupakan DFD *Level* 2 pada turunan proses *Show Active and History Process* dari Gambar 3.2. Pada proses ini terdapat dua proses, yaitu *User Show Active List Process* dan *User Show History List Process*. Proses *User Show Active List Process* digunakan untuk menampilkan rekan-rekan yang sedang mengalami insiden dan pada proses ini, pengguna juga dapat membantu rekan tersebut. Sedangkan, pada proses *User Show History List Process* akan ditampilkan *list* yang berisi rekan-rekan yang pernah dibantu oleh pengguna.



Gambar 3.7 DFD *Level* 2 Pada *Get Notification Process*

Gambar 3.7 merupakan DFD *Level* 2 pada turunan proses *Get Notification Process* dari Gambar 3.2. Proses diawali dengan Firebase mengirimkan data notifikasi yang kemudian akan diproses oleh aplikasi untuk menampilkan notifikasi. Ketika mendapatkan notifikasi, menjalankan proses *Update Get Notification Process* untuk menyimpan *timestamp* dari notifikasi yang diterima. Apabila pengguna mengklik notifikasi yang dikirimkan, maka akan menjalankan proses *Update Click Notification Process* untuk menyimpan *timestamp* dari notifikasi yang diklik. Kedua *timestamp* ini disimpan sebagai bahan dasar untuk analisa *click ratio* dan *click time.*



Gambar 3.8 DFD *Level* 3 Pada *User Show Active List Process*

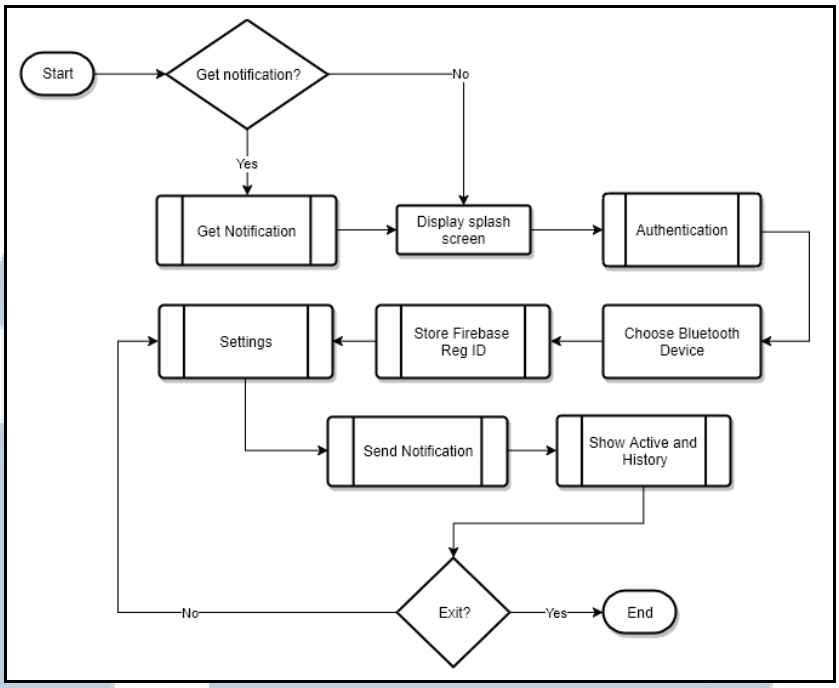
Gambar 3.8 merupakan DFD *Level* 3 pada turunan proses *User Show Active List Process* dari Gambar 3.6. Proses diawali dengan menerima data dari tabel *notification* yang berupa data *detail* dari rekan yang mengalami insiden. Kemudian, data tersebut masuk ke dalam proses *Show Active List Process*, dimana rekan tersebut akan ditampilkan ke *user* dalam bentuk *list*.

Proses *Help Victim Process* digunakan untuk membantu rekan yang mengalami insiden. Pada proses ini akan dilakukan *update* bahwa pengguna membantu rekan tersebut ke dalam tabel *notification*. Pada proses *Go To Victim Location Process*, akan didapatkan lokasi korban dari tabel *notification*.

Arrived Process digunakan untuk meng-*update* data pada tabel *notification* bahwa pengguna telah sampai di lokasi korban. Kemudian, pada proses *Show Hospital Process* akan ditampilkan *list* rumah sakit yang terdekat dari tabel *hospital*. Pada proses *Go To Hospital Process*, pengguna akan memilih rumah sakit yang akan dituju, kemudian akan diberikan navigasi menuju lokasi tersebut. Proses *Done Help Victim Process* digunakan untuk menandakan bahwa pengguna telah selesai membantu rekan yang mengalami insiden tersebut dan akan dilakukan *update* data tersebut pada tabel *notification*.

3.2.2 Flowchart

Tahap awal dalam pengembangan aplikasi notifikasi insiden adalah pembuatan *flowchart* sebagai dasar alur jalannya aplikasi notifikasi insiden. *Flowchart* akan menjelaskan proses-proses pada aplikasi, dimana *flowchart* yang dibuat akan menjelaskan keseluruhan sistem dari aplikasi notifikasi insiden.



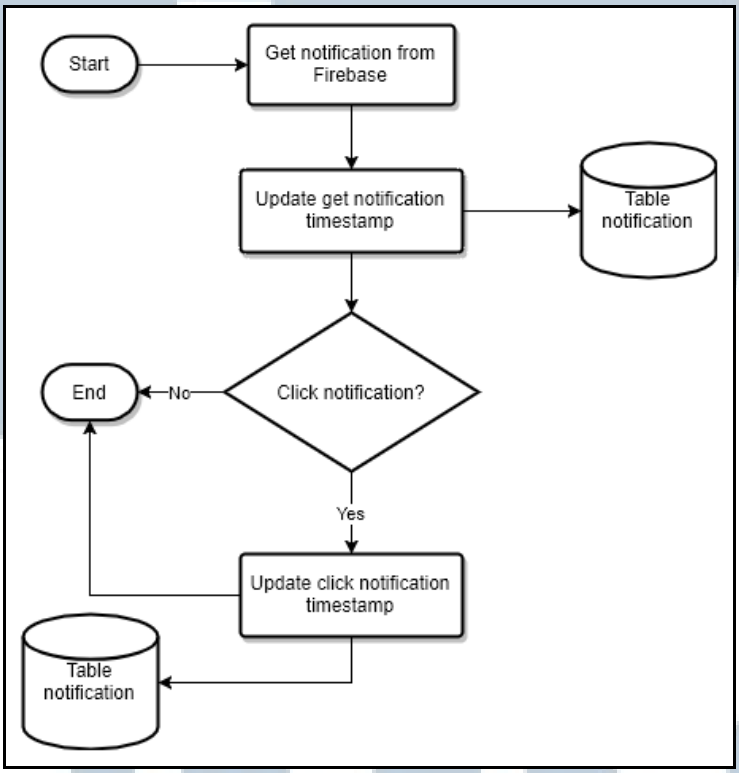
Gambar 3.9 *Flowchart* Aplikasi Notifikasi Insiden Secara Umum

Gambar 3.9 merupakan *flowchart* gambaran sistem aplikasi notifikasi insiden secara umum. Aplikasi dapat dibuka melalui dua cara, yaitu dengan membuka aplikasi secara langsung atau mengklik notifikasi yang didapat. Apabila mendapat notifikasi, akan masuk ke dalam proses *Get Notification*. Setelah melewati baik mendapat notifikasi atau tidak, akan menampilkan *splash screen* yang berisi logo dari aplikasi yang dibuat. Selanjutnya, akan masuk ke dalam proses *Authentication*. Proses ini digunakan untuk melakukan autentikasi ke dalam aplikasi.

Setelah itu, pada proses *Choose Bluetooth Device*, pengguna akan mengoneksikan Bluetooth dari *smartphone* yang dimiliki dengan koneksi Bluetooth yang dimiliki oleh alat dari aplikasi notifikasi. Setelah berhasil mengoneksikan *smartphone* dengan alat, akan masuk ke dalam proses *Store Firebase Reg ID*. Pada proses ini, akan disimpan *registration* ID yang didapatkan dari Firebase ke dalam

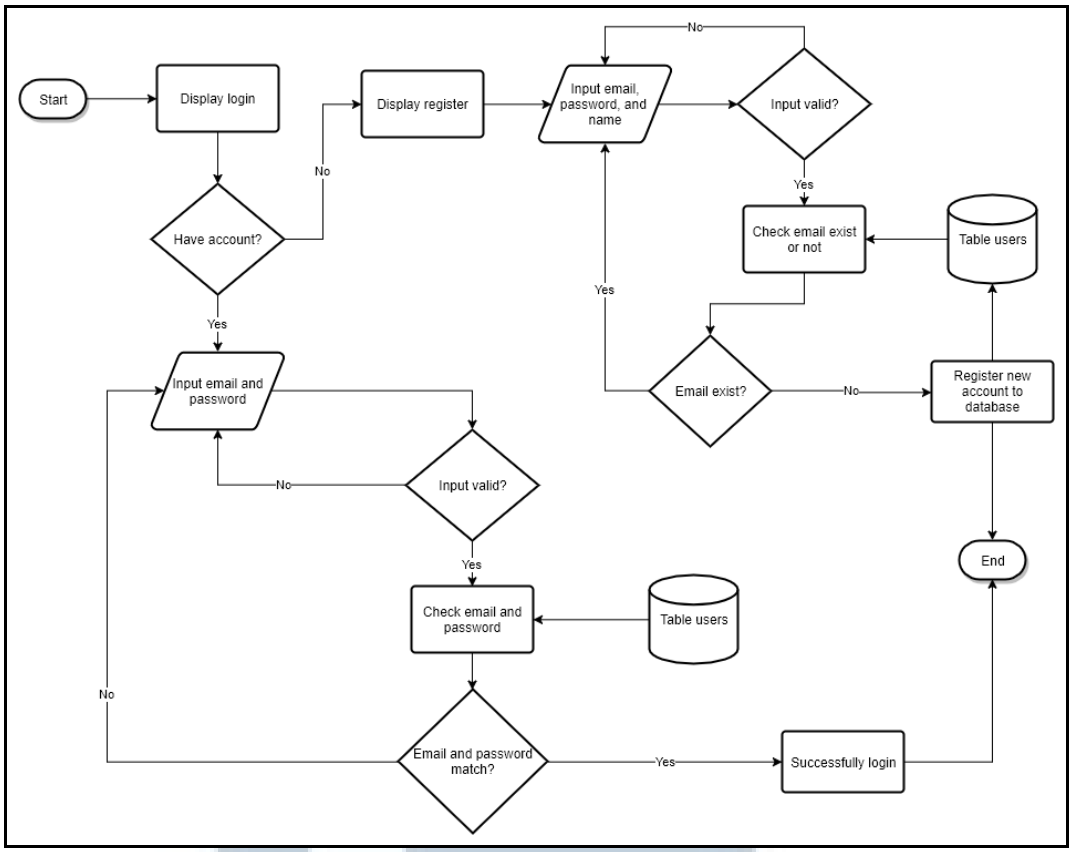
database. Selanjutnya, pada proses *Settings*, pengguna dapat mengatur radius dalam pengiriman notifikasi kepada rekan-rekan.

Pada proses *Send Notification*, akan dilakukan pengiriman notifikasi kepada rekan-rekan penolong ketika pengguna mengalami suatu insiden. Setelah itu, proses *Show Notification* akan jalankan ketika pengguna mendapatkan notifikasi insiden.



Gambar 3.10 *Flowchart Get Notification Process*

Gambar 3.10 merupakan *flowchart get notification* pada aplikasi notifikasi insiden. Proses ini diawali dengan mendapatkan notifikasi dari Firebase. Kemudian, akan disimpan *timestamp* notifikasi didapat ke dalam tabel *notification*. Apabila pengguna mengklik notifikasi, akan menyimpan *timestamp* notifikasi diklik ke dalam tabel *notification* juga.

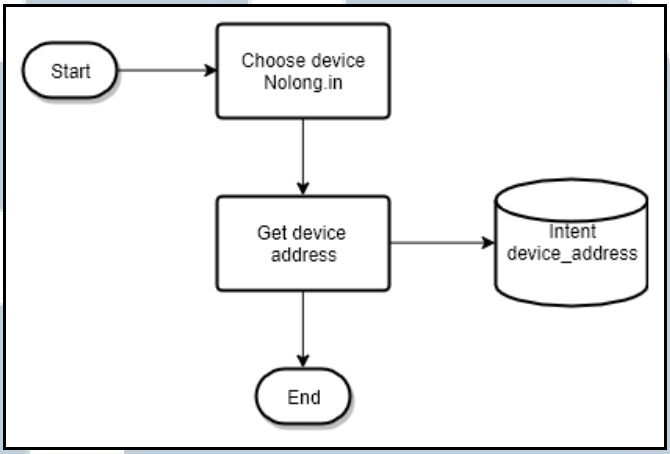


Gambar 3.11 *Flowchart Authentication Process*

Gambar 3.11 merupakan *flowchart authentication* pada aplikasi notifikasi insiden. Proses ini digunakan untuk melakukan autentikasi ke dalam aplikasi ini. Proses diawali dengan ditampilkan halaman untuk melakukan login. Jika pengguna belum memiliki akun, akan ditampilkan halaman *register*. Pada halaman ini, pengguna akan memasukkan *email*, *password*, dan nama, kemudian *input* yang dimasukkan akan dicek apakah valid atau tidak, jika valid, *email* yang di-*input* akan dicek di dalam *database* apakah tersedia atau tidak. Jika *email* tersebut tersedia, pengguna berhasil melakukan registrasi, namun jika tidak, pengguna harus memasukkan *email* yang lain, *password*, dan nama.

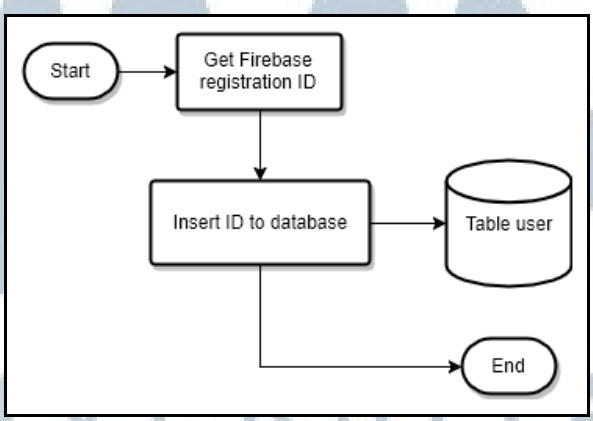
Apabila sudah memiliki akun, pengguna memasukkan *email* dan *password* yang sudah didaftarkan di dalam aplikasi notifikasi ini, kemudian akan dicek apakah *input* yang dimasukkan valid atau tidak. Jika valid, aplikasi akan melakukan

pengecekan apakah *email* dan *password* tersebut sesuai dengan data dalam *database*, jika sesuai, pengguna berhasil melakukan *login*. Namun, jika tidak, pengguna harus memasukkan *email* dan *password* yang sudah didaftarkan.



Gambar 3.12 *Flowchart Choose Bluetooth Device Process*

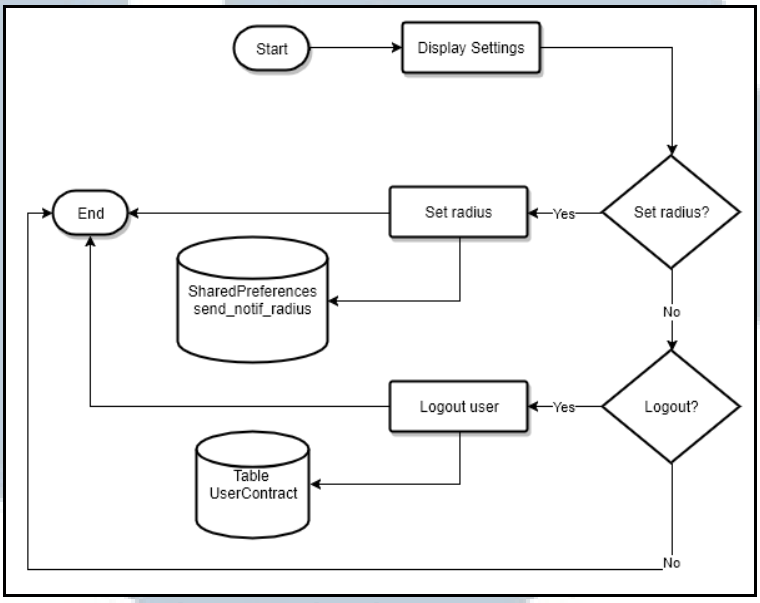
Gambar 3.12 merupakan *flowchart choose Bluetooth device* pada aplikasi notifikasi insiden. Proses ini digunakan untuk mengoneksikan alat dengan aplikasi dengan menggunakan Bluetooth. Proses diawali dengan memilih alat dari Nolong.in. Selanjutnya, *address* yang didapatkan dari alat tersebut dimasukkan ke dalam *Intent device_address*.



Gambar 3.13 *Flowchart Store Firebase Reg* ID *Process*

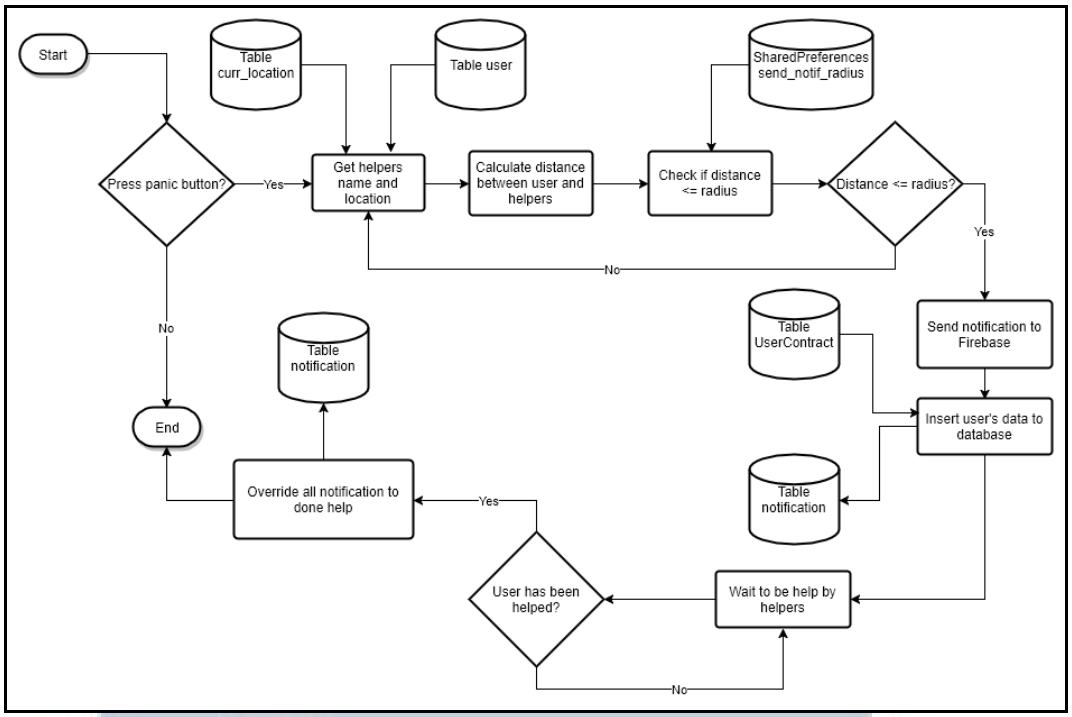
Gambar 3.13 merupakan *flowchart* fungsi *store Firebase registration* ID pada aplikasi notifikasi insiden. Fungsi ini digunakan untuk menyimpan *registration* ID yang sudah didapatkan sebelumnya dari Firebase ke dalam

database, yaitu tabel *user*. ID ini digunakan untuk membedakan setiap aplikasi notifikasi insiden yang diunduh oleh pengguna. Sehingga, pengiriman notifikasi oleh rekan yang mengalami insiden dapat dikirimkan kepada rekan yang dekat dengan lokasinya.



Gambar 3.14 *Flowchart Settings Process*

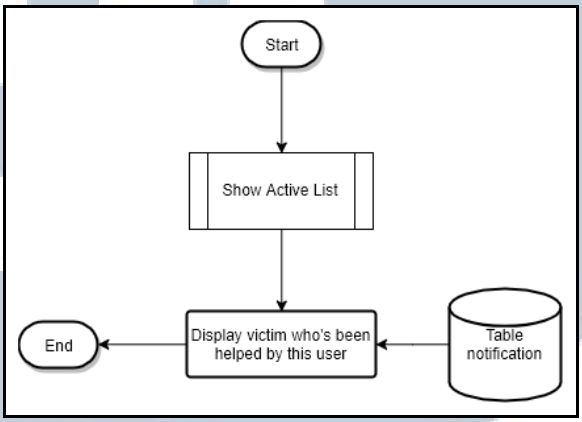
Gambar 3.14 merupakan *flowchart* fungsi *settings* pada aplikasi notifikasi insiden. Fungsi ini digunakan untuk mengatur radius dari pengiriman notifikasi yang diinginkan. Selain mengatur radius, juga dapat digunakan untuk melakukan *logout* dari aplikasi.



Gambar 3.15 *Flowchart Send Notification Process*

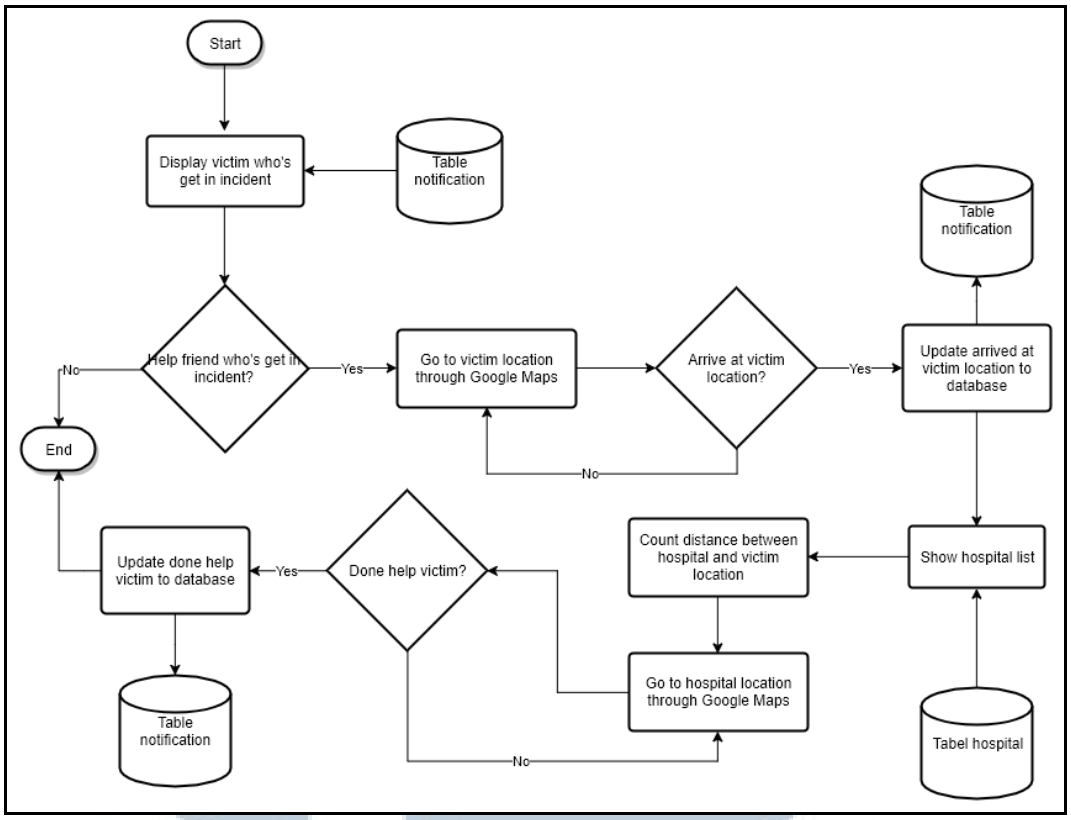
Gambar 3.15 merupakan *flowchart* untuk fungsi *send notification* pada aplikasi notifikasi insiden. Fungsi ini akan dijalankan ketika pengguna mengalami insiden. Proses dimulai jika pengguna menekan tombol *panic button* pada alat, kemudian akan didapatkan data yang berupa nama dan lokasi *helpers* atau rekan yang akan menolong dari tabel *curr_location* dan *user*. Setelah itu, akan dilakukan kalkulasi jarak terhadap lokasi *helpers* dengan rekan yang mengalami insiden. Jarak tersebut didapat dari radius yang sudah diatur sebelumnya pada bagian *Settings*, kemudian disimpan data radius tersebut pada *SharedPreferences send_notif_radius*. Jika hasil jarak tersebut lebih kecil dari atau sama dengan radius yang sudah ditentukan oleh pengguna, notifikasi akan dikirimkan ke Firebase. Kemudian, Firebase yang akan mengirimkan notifikasi ke *helpers* tersebut. Namun, jika tidak, akan dilanjutkan dengan *helpers* selanjutnya yang sebelumnya telah didapatkan.

Setelah notifikasi dikirimkan ke *helpers*, data rekan yang mengalami insiden akan dimasukkan ke dalam *database*, yaitu tabel *notification*. Kemudian, pengguna akan menunggu hingga *helpers* datang dan selesai membantu. Setelah *helpers* selesai membantu, pengguna yang mengalami insiden dapat melakukan *override* pada semua notifikasi yang dikirimkan agar menjadi selesai dibantu, hal ini dilakukan agar tidak ada *helpers* lain yang datang untuk menolong pengguna.



Gambar 3.16 *Flowchart Show Active and History Process*

Gambar 3.16 merupakan *flowchart* untuk fungsi *show active and history* pada aplikasi notifikasi insiden. Pada fungsi ini, terdapat proses *Show Active List* yang digunakan untuk menampilkan data rekan yang sedang mengalami insiden. Kemudian, terdapat proses untuk menampilkan data rekan yang pernah mengalami insiden dan dibantu oleh pengguna dari tabel *notification*.



Gambar 3.17 *Flowchart Show Active List Process*

Gambar 3.17 merupakan *flowchart* untuk fungsi *show active list* pada aplikasi notifikasi insiden. Proses ini diawali dengan mendapatkan data detail rekan yang sedang mengalami insiden dan yang pernah dibantu oleh pengguna dari *database*, yaitu tabel *notification*. Setelah itu, pengguna dapat memilih apakah ingin membantu rekan yang sedang mengalami insiden atau tidak, jika pengguna memilih untuk membantu, dapat mengikuti navigasi dari *third-party app*, yaitu Google Maps. Setelah pengguna sampai di lokasi rekan yang mengalami insiden, akan dilakukan *update* data ke dalam *database* untuk menunjukkan bahwa pengguna telah sampai di lokasi rekan tersebut.

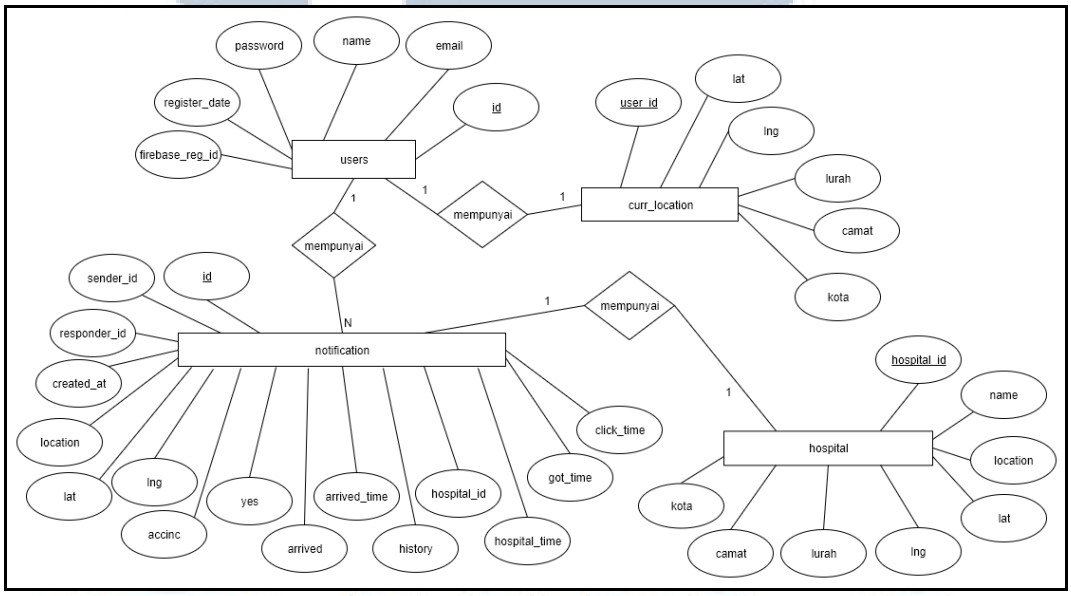
Selanjutnya, akan ditampilkan *list* rumah sakit yang tergabung dalam BPJS yang berada di sekitar lokasi kejadian dari tabel *hospital* beserta jarak antara lokasi

rumah sakit tersebut dengan lokasi kejadian menggunakan Google Maps Distance Matrix API. Isi dari tabel *hospital* terdapat pada lampiran.

Setelah itu, pengguna dapat memilih rumah sakit yang ingin dituju. Kemudian, akan diarahkan ke rumah sakit tersebut melalui *third-party app*, yaitu Google Maps. Apabila pengguna telah sampai di rumah sakit tersebut dan selesai membantu rekan, akan di-*update* bahwa rekan telah selesai dibantu oleh pengguna ke dalam tabel *notification*.

3.2.3 Entity Relationship Diagram

Sebelum melakukan perancangan *database*, harus melakukan terlebih dahulu perancangan *Entity Relationship Diagram* (ERD) sehingga dapat diketahui hubungan antar entitas dan atribut-atribut yang dimiliki oleh entitas tersebut.



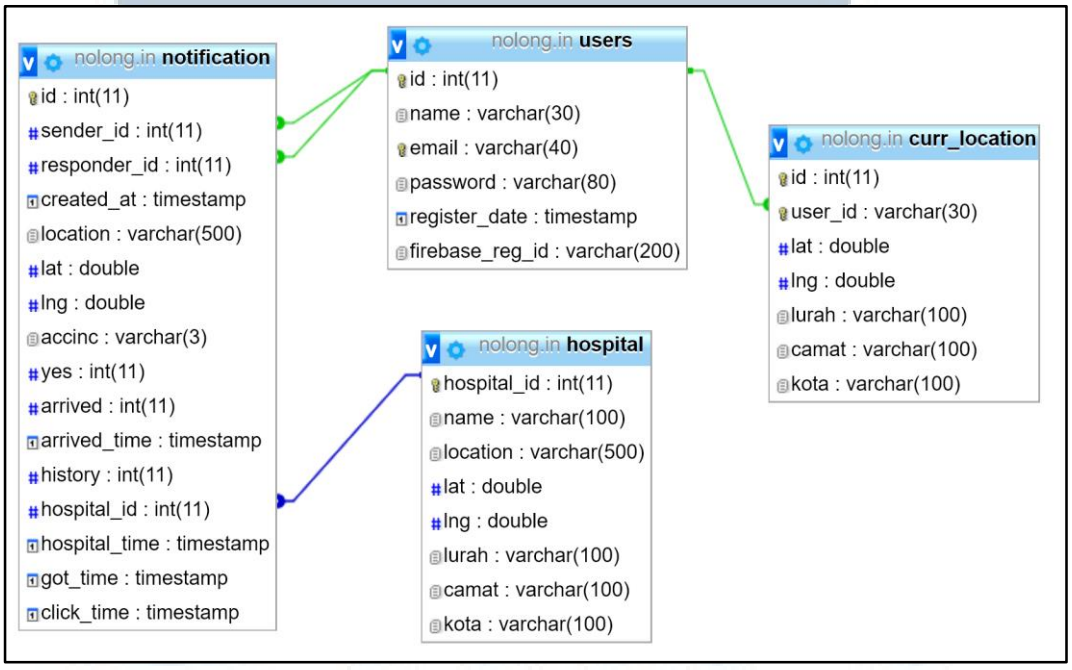
Gambar 3.18 ERD Aplikasi Notifikasi Insiden

Gambar 3.18 merupakan *entity relationship diagram* pada aplikasi notifikasi insiden. Terdapat empat buah entitas, yaitu *users*, *curr_location*, *notification*, dan *hospital*. Hubungan yang ada pada diagram ini adalah setiap *user* hanya memiliki sebuah *curr_location* dan setiap *user* dapat memiliki banyak

notification. Entitas *users* digunakan untuk menyimpan setiap *user* yang menggunakan aplikasi notifikasi insiden. Entitas *curr_location* digunakan untuk menyimpan lokasi terkini dari masing-masing *user* saat membuka aplikasi. Entitas *notification* digunakan untuk menyimpan data notifikasi yang dikirimkan oleh *user* yang mengalami insiden. Entitas *hospital* digunakan untuk menyimpan data-data rumah sakit.

3.2.4 Database Schema

Database schema dirancang sehingga terlihat hubungan antara kolom pada suatu tabel dengan kolom pada tabel lainnya. Dengan adanya hubungan tersebut, ketergantungan suatu tabel dengan tabel lainnya pun dapat dilihat.



Gambar 3.19 *Database Schema*

Gambar 3.19 merupakan *database schema* pada aplikasi notifikasi insiden. Pada *schema* ini, keempat tabel tersebut memiki hubungan. Tabel notification memiliki hubungan pada kolom sender_id dan responder_id dengan tabel users pada kolom id. Tabel notification juga memiliki hubungan pada kolom hospital_id

dengan tabel hospital pada kolom hospital_id. Tabel users memiliki hubungan pada kolom id dengan tabel curr_location pada kolom user_id.

3.2.5 Struktur Tabel

Dalam penelitian ini, terdapat dua macam *database* yang digunakan, yaitu *MySQL* dan *SQLite*. *Database MySQL* digunakan untuk melakukan *query database* pada *server* yang digunakan. Sedangkan, *SQLite* digunakan untuk melakukan *query database* yang ada di dalam *smartphone user*. Terdapat pula *SharedPreferences* yang digunakan untuk menyimpan data yang didapat dari pengaturan pada aplikasi. Berikut adalah struktur tabel yang digunakan dalam aplikasi notifikasi insiden.

A. MySQL

Pada *database* MySQL, terdapat empat buah tabel yang digunakan dalam penyimpanan dan penarikan data dalam aplikasi notifikasi insiden.

1. Nama Tabel : users

   Fungsi : Menyimpan data *user*

   Primary Key : id

   Foreign Key : -

Tabel 3.1 Struktur Tabel users

Nama Kolom	Tipe	Keterangan
id	int (11)	*Primary Key*
name	varchar (30)	Menyimpan data nama
email	varchar (40)	Menyimpan data *email*
password	varchar (80)	Menyimpan data *password*
register_date	timestamp	Menyimpan data *timestamp* pengguna melakukan registrasi
firebase_reg_id	varchar (200)	Menyimpan data Firebase *Registraiton* ID

2. Nama Tabel : curr_location

   Fungsi : Menyimpan data lokasi terkini *user*

Primary Key : id

Foreign Key : user_id

Tabel 3.2 Struktur Tabel curr_location

Nama Kolom	Tipe	Keterangan
id	int (11)	*Primary Key*
user_id	varchar (30)	*Reference key* kepada *id* dalam tabel *users*
lat	double	Menyimpan data nilai *latitude*
lng	double	Menyimpan data nilai *longitude*
lurah	varchar (100)	Menyimpan data nama kelurahan
camat	varchar (100)	Menyimpan data nama kecamatan
kota	varchar (100)	Menyimpan data nama kota

3. Nama Tabel : notification

Fungsi : Menyimpan data notifikasi dari rekan yang mengalami insiden

Primary Key : id

Foreign Key : sender_id, responder_id, dan hospital_id

Tabel 3.3 Struktur Tabel notification

Nama Kolom	Tipe	Keterangan
id	int (11)	*Primary Key*
sender_id	int (11)	*Reference key* kepada *id* dalam tabel *users*
responder_id	int (11)	*Reference key* kepada *id* dalam tabel *users*
created_at	timestamp	Menyimpan data *timestamp* notifikasi
location	varchar (500)	Menyimpan data nama lokasi
lat	double	Menyimpan data nilai *latitude*
lng	double	Menyimpan data nilai *longitude*
accinc	varchar (3)	Menyimpan data *flag* yang berupa *acc* atau *inc*

Tabel 3.3 Struktur Tabel notification (Lanjutan)

Nama Kolom	Tipe	Keterangan
yes	int (11)	Menyimpan data *flag* apakah rekan ingin membantu
arrived	int (11)	Menyimpan data *flag* apakah rekan sudah sampai di lokasi
arrived_time	timestamp	Menyimpan data *timestamp* rekan sampai di lokasi
history	int (11)	Menyimpan data *flag* apakah sudah selesai membantu rekan
hospital_id	int (11)	*Reference key* kepada *hospital_id* dalam tabel *hospital*
hospital_time	timestamp	Menyimpan data *timestamp* rekan sampai di rumah sakit
got_time	timestamp	Menyimpan data *timestamp* notifikasi diterima
click_time	timestamp	Menyimpan data *timestamp* notifikasi diklik

4. Nama Tabel : hospital

Fungsi : Menyimpan data rumah sakit yang tergabung dalam BPJS

Primary Key : hospital_id

Foreign Key : -

Tabel 3.4 Struktur Tabel hospital

Nama Kolom	Tipe	Keterangan
hospital_id	int (11)	*Primary Key*
name	varchar (100)	Menyimpan data nama rumah sakit
location	varchar (500)	Menyimpan data lokasi rumah sakit
lat	double	Menyimpan data nilai *latitude* rumah sakit
lng	double	Menyimpan data nilai *longitude* rumah sakit

Tabel 3.4 Struktur Tabel hospital (Lanjutan)

Nama Kolom	Tipe	Keterangan
lurah	varchar (100)	Menyimpan data nama keluarahan rumah sakit
camat	varchar (100)	Menyimpan data nama kecamatan rumah sakit
kota	varchar (100)	Menyimpan data nama kota rumah sakit

B. SQLite

Pada *database* SQLite, hanya terdapat satu tabel, yaitu *UserContract* yang digunakan untuk melakukan penyimpanan dan penarikan data *user* yang disimpan ketika melakukan *register* dan *login*.

1. Nama Tabel : UserContract

   Fungsi : Menyimpan data *user*

   Primary Key : id

   Foreign Key : -

Tabel 3.5 Struktur Tabel UserContract

Nama Kolom	Tipe	Keterangan
id	int (11)	*Primary Key*
user_id	varchar (10)	Menyimpan data ID pengguna
name	varchar (30)	Menyimpan data nama pengguna
email	varchar (40)	Menyimpan data *email* pengguna
register_date	text	Menyimpan data *timestamp* pengguna melakukan registrasi

2. Nama Tabel : LocationContract

   Fungsi : Menyimpan data lokasi *user*

   Primary Key : id

   Foreign Key : -

Tabel 3.6 Struktur Tabel LocationContract

Nama Kolom	Tipe	Keterangan
id	int (11)	*Primary Key*
lat	double	Menyimpan data nilai *latitude* pengguna
lng	double	Menyimpan data nilai *longitude* pengguna
lurah	varchar (80)	Menyimpan data nama keluarahan pengguna
camat	varchar (80)	Menyimpan data nama kecamatan pengguna
kota	varchar (80)	Menyimpan data nama kota pengguna
location	varchar (300)	Menyimpan data nama lokasi pengguna

C. SharedPreferences

Pada aplikasi notifikasi insiden ini, digunakan *SharedPreferences* untuk melakukan penyimpanan dan penarikan data pada pengaturan aplikasi.

Key : send_notif_radius

Default Value : 5

Fungsi : Menyimpan data radius dalam pengiriman notifikasi

Tabel 3.7 SharedPreferences send_notif_radius

Key	Default Value
send_notif_radius	5

D. Intent

Pada aplikasi notifikasi insiden ini, digunakan *Intent* untuk melakukan pengiriman data antar *activity* pada aplikasi.

Key : device_address

Default Value : null

Fungsi : Menyimpan data *address* dari alat notifikasi insiden

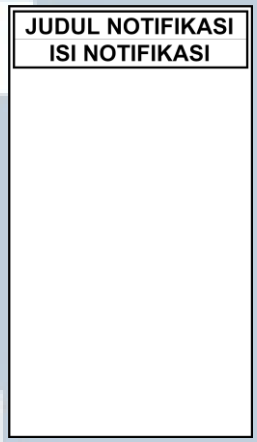
Tabel 3.8 Intent device_address

Key	Default Value
device_address	null

3.2.6 Rancangan Antarmuka

Sebelum membuat tampilan dari sebuah aplikasi, dibutuhkan sebuah rancangan antarmuka yang digunakan sebagai acuan dalam membangun aplikasi tersebut. Berikut merupakan tampilan untuk aplikasi notifikasi insiden ini.

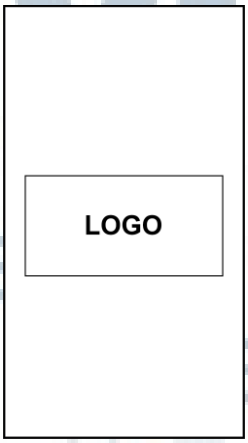
A. Rancangan Antarmuka Pada Proses Get Notification



Gambar 3.20 Rancangan Antarmuka Pada *Get Notification Process*

Gambar 3.20 merupakan rancangan antarmuka untuk proses *Get Notification* pada Gambar 3.10. Tampilan ini merupakan notifikasi yang akan didapatkan pengguna ketika terdapat pengguna yang mengalami insiden. Pada tampilan ini, terdapat judul notifikasi dan isi dari notifikasi tersebut.

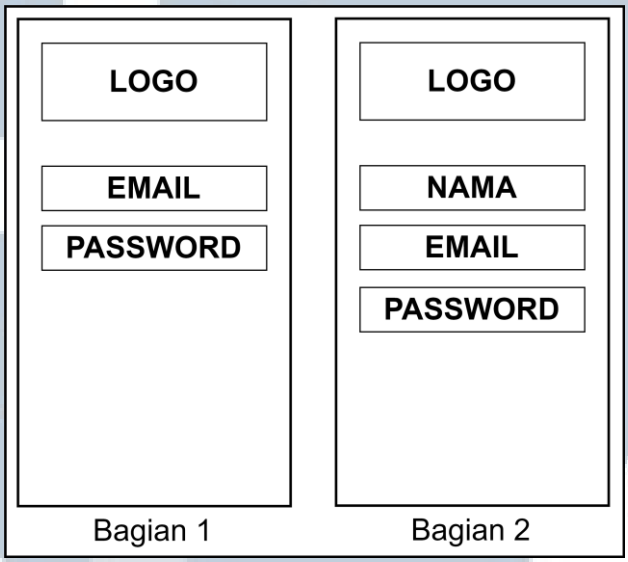
B. Rancangan Antarmuka Pada Proses Display SplashScreen



Gambar 3.21 Rancangan Antarmuka Pada *Display SplashScreen Process*

Gambar 3.21 merupakan rancangan antarmuka untuk proses *Display Splash Screen* pada Gambar 3.9. Tampilan ini merupakan tampilan awal ketika membuka aplikasi. Pada tampilan ini, terdapat logo dari aplikasi notifikasi insiden ini.

C. Rancangan Antarmuka Pada Proses Authentication



Gambar 3.22 Rancangan Antarmuka Pada *Authentication Process*

Gambar 3.22 merupakan rancangan antarmuka untuk proses *Authentication* pada Gambar 3.11. Pada rancangan ini terdapat dua bagian, Bagian 1 berisikan tampilan yang digunakan pengguna untuk melakukan *login* ke dalam aplikasi. Pada tampilan ini, terdapat logo dari aplikasi ini, kolom *email* dan *password*. Kolom *email* digunakan untuk memasukkan *email* yang sudah diregistrasikan sebelumnya pada aplikasi ini. Sedangkan, kolom *password* digunakan untuk memasukkan kata sandi yang sudah diregistrasikan sebelumnya bersama *email*.

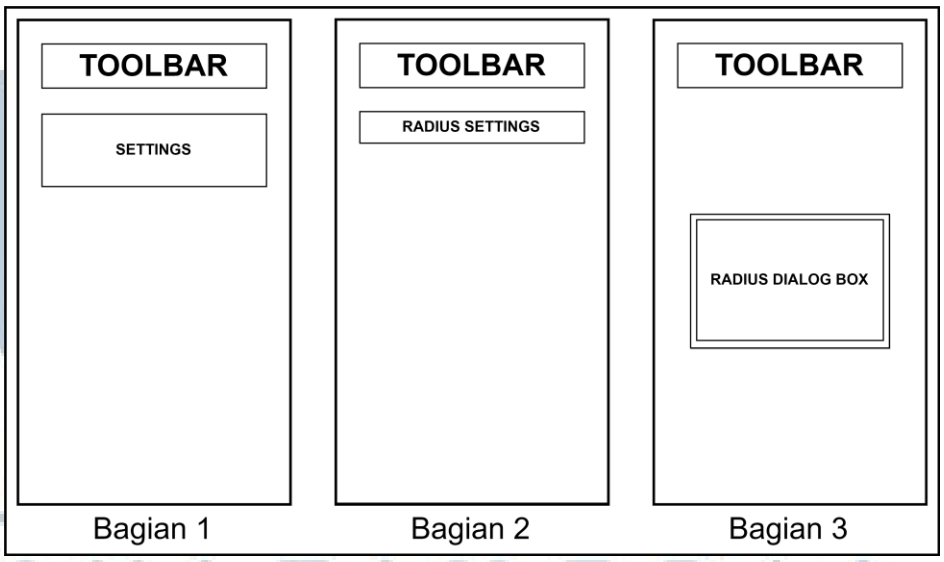
Bagian 2 merupakan tampilan untuk pengguna melakukan *register* ke dalam aplikasi. Pada tampilan ini, terdapat logo dari aplikasi ini, kolom nama, *email*, dan *password*. Ketiga kolom ini harus diisi dengan benar agar dapat melakukan registrasi ke dalam aplikasi ini.

D. Rancangan Antarmuka Pada Proses Choose Bluetooth Device

Gambar 3.23 Rancangan Antarmuka Pada *Choose Bluetooth Device Process*

Gambar 3.23 merupakan rancangan antarmuka untuk proses *Choose Bluetooth Device* pada Gambar 3.12. Tampilan ini digunakan untuk mengoneksikan Bluetooth pada *smartphone* dengan alat notifikasi insiden. Pada tampilan ini, terdapat *list* dari jaringan Bluetooth yang pernah dikoneksikan dan panduan untuk mengoneksikan *smartphone* dengan alat ini.

E. Tampilan Pada Proses Settings

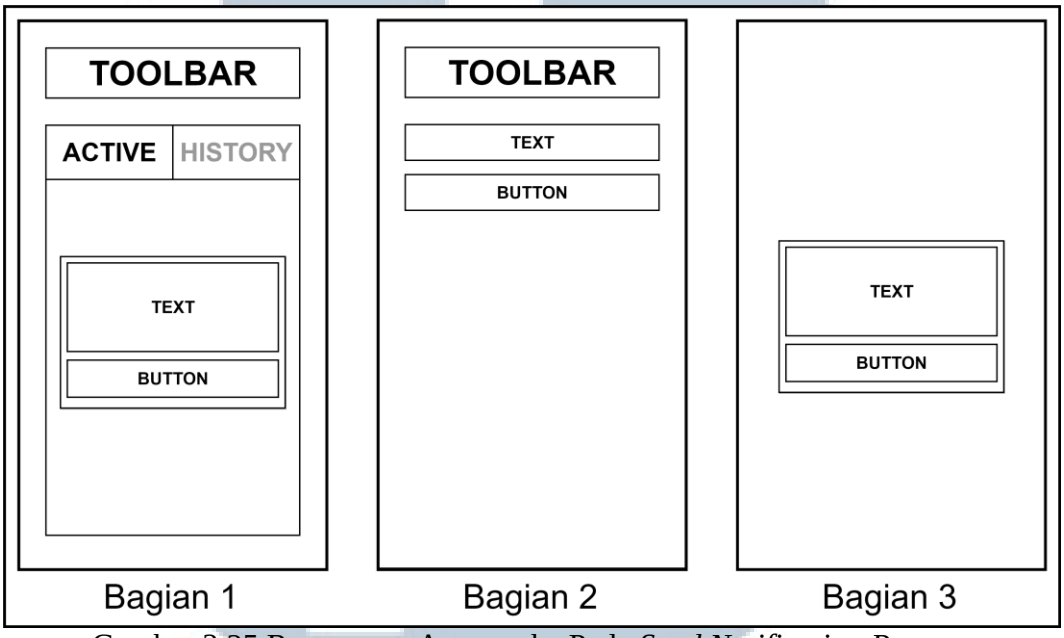


Gambar 3.24 Rancangan Antarmuka Pada *Settings Process*

Gambar 3.24 merupakan rancangan antarmuka untuk proses *Settings* pada Gambar 3.14. Tampilan ini digunakan untuk mengatur radius yang diinginkan

dalam pengiriman notifikasi kepada rekan-rekan. Selain untuk mengatur radius, terdapat pula pilihan untuk melakukan *logout*.

F. Tampilan Pada Proses Send Notification

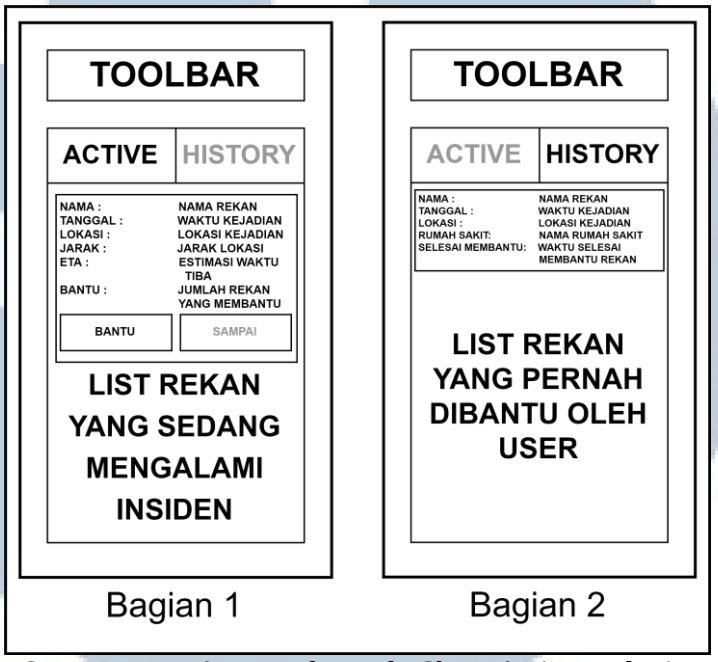


Gambar 3.25 Rancangan Antarmuka Pada *Send Notification Process*

Gambar 3.25 merupakan rancangan antarmuka untuk proses *Send Notification* pada Gambar 3.15. Terdapat tiga bagian pada tampilan ini. Pada Bagian 1 terdapat *dialog box* yang berisikan sebuah *text* dan *button*. *Dialog* ini muncul ketika pengguna menekan *panic button* yang ada pada alat. *Text* tersebut berisi penjelasan bahwa pengguna telah menekan tombol pada alat. Sedangkan, *button* tersebut digunakan untuk membatalkan pengiriman notifikasi kepada rekan-rekan. Selain itu, pada *button* tersebut juga berisi *count down* yang menghitung mundur dari angka 5 hingga 0. Jika mencapai angka 0, notifikasi akan dikirimkan kepada rekan-rekan dan akan ditampilkan Bagian 2. Tampilan pada Bagian 2 merupakan tampilan yang berisikan *text* dan *button*. Apabila pengguna telah selesai dibantu oleh rekan-rekannya, dapat menekan *button* ini. Bagian 3 merupakan

tampilan *dialog box* yang akan muncul ketika *button* pada Bagian 2 ditekan. *Dialog box* digunakan sebagai konfirmasi bahwa telah selesai dibantu.

G. Tampilan Pada Proses Show Active and History

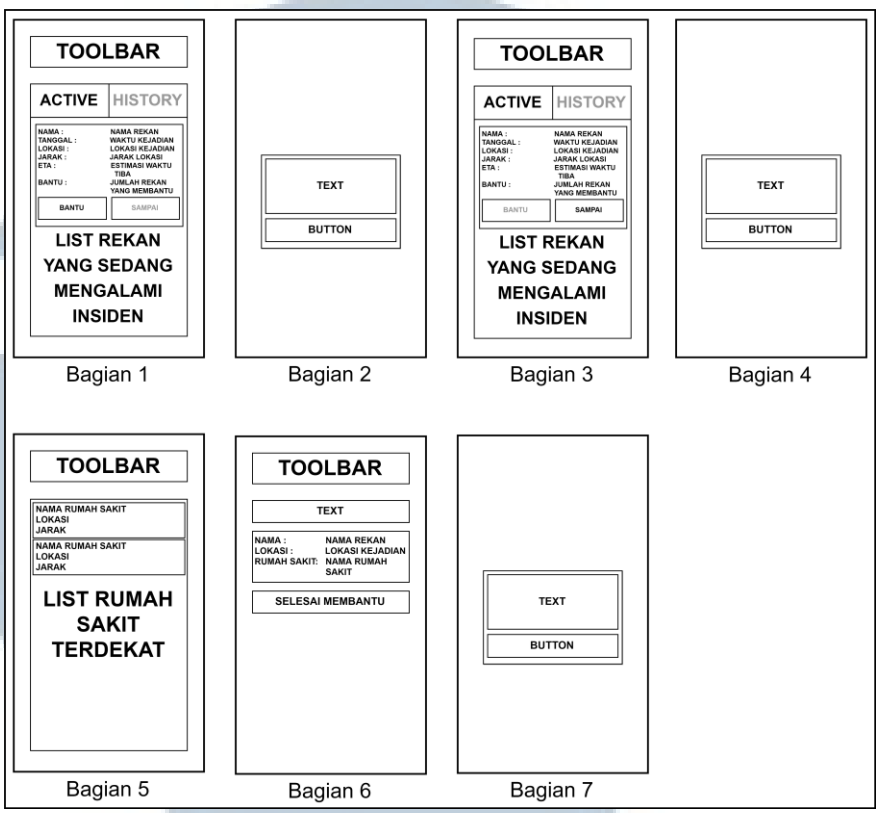


Gambar 3.26 Rancangan Antarmuka Pada *Show Active and History Process*

Gambar 3.26 merupakan rancangan antarmuka untuk proses *Show Active and History* pada Gambar 3.16. Pada tampilan ini terdapat dua bagian, Bagian 1 merupakan tampilan ketika terdapat rekan yang mengalami insiden. Pada bagian ini, terdapat detail rekan tersebut, yaitu nama, tanggal kejadian, lokasi kejadian, jarak antara pengguna ke lokasi kejadian, estimasi waktu tiba, dan jumlah berapa orang yang membantu, serta tombol BANTU dan SAMPAI yang akan dijelaskan lebih lanjut pada Gambar 3.25.

Bagian 2 merupakan tampilan yang berisikan rekan-rekan yang pernah mengalami insiden dan dibantu oleh pengguna. Bagian ini terdapat tampilan yang berisikan nama rekan, tanggal kejadian, lokasi kejadian, rumah sakit yang dituju, dan waktu selesai membantu rekan tersebut.

H. Tampilan Pada Proses Show Active



Gambar 3.27 Rancangan Antarmuka Pada *Show Active List Process*

Gambar 3.27 merupakan rancangan antarmuka untuk proses *Show Active List* pada Gambar 3.17. Pada tampilan ini terdapat tujuh bagian, Bagian 1 merupakan tampilan ketika terdapat rekan yang mengalami insiden. Tampilan ini sama seperti tampilan Bagian 1 pada Gambar 3.24. Pada tampilan ini terdapat nama rekan yang mengalami insiden, tanggal kejadian, lokasi kejadian, jarak antara lokasi pengguna dengan tempat kejadian, estimasi waktu sampai di lokasi, dan jumlah rekan yang membantu rekan yang mengalami insiden tersebut. Selain itu, terdapat juga tombol BANTU yang digunakan untuk menandakan bahwa pengguna ingin membantu rekan tersebut dan pergi ke lokasinya, serta tombol SAMPAI untuk menandakan bahwa pengguna telah sampai di lokasi kejadian. Pada tampilan ini hanya tombol BANTU yang dapat ditekan, tombol SAMPAI tidak dapat ditekan.

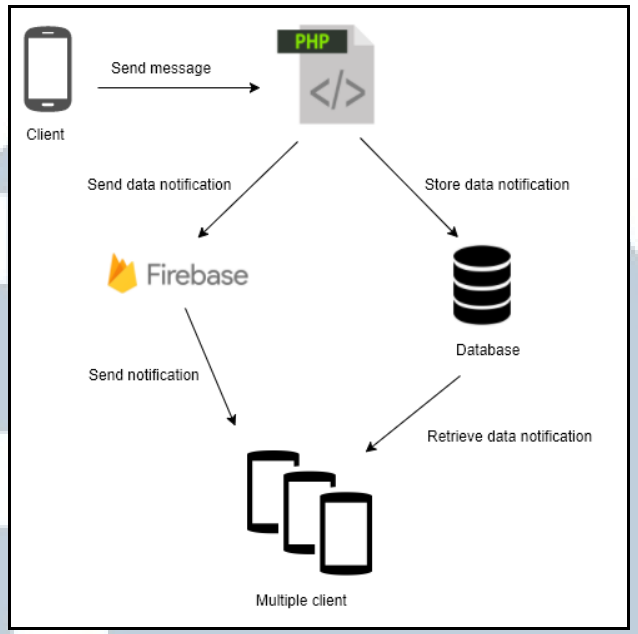
Pada tampilan Bagian 2 terdapat *dialog box* yang akan muncul ketika tombol BANTU pada Bagian 1 ditekan. *Dilaog box* ini digunakan sebagai konfirmasi bahwa pengguna menekan tombol BANTU.

Pada Bagian 3 sama seperti tampilan Bagian 1, hanya saja tombol SAMPAI dibuat menjadi dapat ditekan dan tombol BANTU dibuat menjadi tidak dapat ditekan. Seperti halnya tampilan dan kegunaan pada Bagian 2, Bagian 4 juga terdapat *dialog box*, hanya saja *dialog box* ini muncul ketika tombol SAMPAI ditekan.

Bagian 5 terdapat *list* rumah sakit yang berada di sekitar lokasi kejadian. *List* ini berisi nama rumah sakit, lokasi, dan jarak antara lokasi kejadian dengan rumah sakit. Setelah pengguna sampai di rumah sakit yang dituju, akan ditampilkan tampilan pada Bagian 6. Pada bagian ini, terdapat tampilan yang berisikan nama rekan yang mengalami insiden, lokasi kejadian, rumah sakit yang dituju, dan tombol SELESAI MEMBANTU. Tombol ini digunakan untuk menandakan bahwa pengguna telah selesai membantu rekan. Pada Bagian 7, terdapat *dialog box* yang muncul ketika tombol SELESAI MEMBANTU pada Bagian 6 ditekan. *Dialog box* ini digunakan sebagai konfirmasi bahwa pengguna telah menekan tombol SELESAI.

3.2.7 Arsitektur Sistem

Pada subbab ini akan dijelaskan arsitektur dari sistem yang digunakan. Berikut adalah arsitektur sistem Nolong.in.



Gambar 3.28 Arsitektur Sistem

Gambar 3.28 merupakan arsitektur dari sistem Nolong.in. Ketika pengguna menekan tombol pada alat, aplikasi akan mengirimkan data-data yang dibutukan Firebase untuk mengirimkan notifikasi ke PHP. Data tersebut berisi tujuan pengiriman yang berupa Firebase *registration* ID pengguna dan pesan yang akan dikirimkan. Kemudian, PHP akan mengirimkan data tersebut ke Firebase dan menyimpannya ke dalam *database*. Setelah itu, Firebase akan mengirimkan notifikasi ke *client* lain yang dituju dan *database* akan menampilkan data-data tersebut di dalam aplikasi *client*.