BAB 5 IMPLEMENTASI

Pada bab ini membahas mengenai implementasi pembuatan aplikasi pencarian dan pemberian bantuan terhadap permasalahan kendaraan berdasakan lokasi terdekat berasal dari hasil yang diperoleh saat melakukan analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan implementasi terdiri dari penjelasan tentang spesifikasi lingkungan implementasi, batasan-batasan implementasi, implementasi basis data, implementasi kode program, dan implementasi antarmuka aplikasi.

5.1 Spesifikasi sistem

Hasil dari tahap analisis kebutuhan dan perancangan sistem menjadi dasar untuk dilakukan implementasi menjadi aplikasi pencarian dan pemberian bantuan terhadap permasalahan kendaraan berdasakan lokasi terdekat agar dapat berfungsi sesuai kebutuhan. Implementasi sistem bekerja pada lingkungan perangkat keras dan perangkat lunak. Perangkat keras yang digunakan pada lingkungan *client* adalah perangkat bergerak (*smartphone*).

5.1.1 Spesifikasi perangkat keras

Dalam pengembangan aplikasi pencarian dan pemberian bantuan terhadap permasalahan kendaraan berdasarkan lokasi terdekat menggunakan sebuah komputer dengan spesifikasi yang dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras komputer

Nama Komponen	**Spesifikasi**
Model Sistem	Laptop Lenovo Z40
Prosesor	Intel® Core™ i5 4210U CPU @1.70Ghz
Memori (HDD)	1 TB
Memori (RAM)	8 GB DDR3
Kartu Grafik	NVIDIA Geforce 840M

Proses instalasi dan pengujian perangkat yang digunakan menggunakan *smartphone* Android dengan spesifikasi perangkat keras yang ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat keras s*martphone* Android

No	**Nama Komponen**	**Spesifikasi**
1	Model Sistem	Oneplus One
	Prosesor	Qualcomm MSM8974AC Snapdragon 801 Quad-core 2.5 GHz

Tabel 5.2 Spesifikasi perangkat keras smartphone Android (lanjutan)

	Memori Internal	64 GB
	Memori (RAM)	3 GB
	Layar	5.5 Inchi, HD 1280x720, IPS
	Kamera	13 Megapiksel
	WLAN	802.11 b/g/n
2	Model Sistem	Lenovo P780
	Prosesor	Mediatek MT6589 Quad-core 1.2 GHz
	Memori Internal	4 GB
	Memori (RAM)	1 GB
	Layar	5 Inchi, HD 1280x720, IPS
	Kamera	8 Megapiksel
	WLAN	802.11 a/b/g/n
3	Model Sistem	Xiaomi Redmi 4A
	Prosesor	Qualcomm MSM8917 Snapdragon 425 Quad-core 1.4 GHz
	Memori Internal	16 GB
	Memori (RAM)	2 GB
	Layar	5 Inchi, HD 1280x720, IPS
	Kamera	13 Megapiksel
	WLAN	802.11 b/g/n

5.1.2 Spesifikasi perangkat lunak

Dalam pengembangan aplikasi pencarian dan pemberian bantuan terhadap permasalahan kendaraan mengunakan pendekatan *Location Based Service* (LBS) menggunakan sebuah komputer dengan spesifikasi perangkat lunak yang dijelaskan pada Tabel 5.3.

Tabel 5.3 Spesifikasi perangkat lunak komputer

Nama Komponen	**Spesifikasi**
Sistem Operasi	Windows 10 Pro © 2017 64-Bit
Bahasa Pemrograman	Java, XML, PHP
Software Development Kit	Java SE Development Kit 8 (64-Bit)

Tabel 5.3 Spesifikasi perangkat lunak komputer (lanjutan)

Programming Environtment	Java Runtime Environtment 8 PHP 7.1
Android SDK	API 16
Editor	Android Studio PHPStorm

Proses instalasi dan pengujian perangkat dilakukan pada perangkat bergerak *smartphone* Android dengan spesifikasi perangkat lunak yang ditunjukkan pada Tabel 5.4

Tabel 5.4 Spesifikasi perangkat lunak *smartphone* Android

No	Nama Komponen	Spesifikasi
1	Sistem Operasi	Android Versi 7.1.1 (Nougat)
2	Sistem Operasi	Android Versi 4.4 (Kitkat)
3	Sistem Operasi	Android Versi 6.0 (Marhsmallow)

5.2 Batasan implementasi

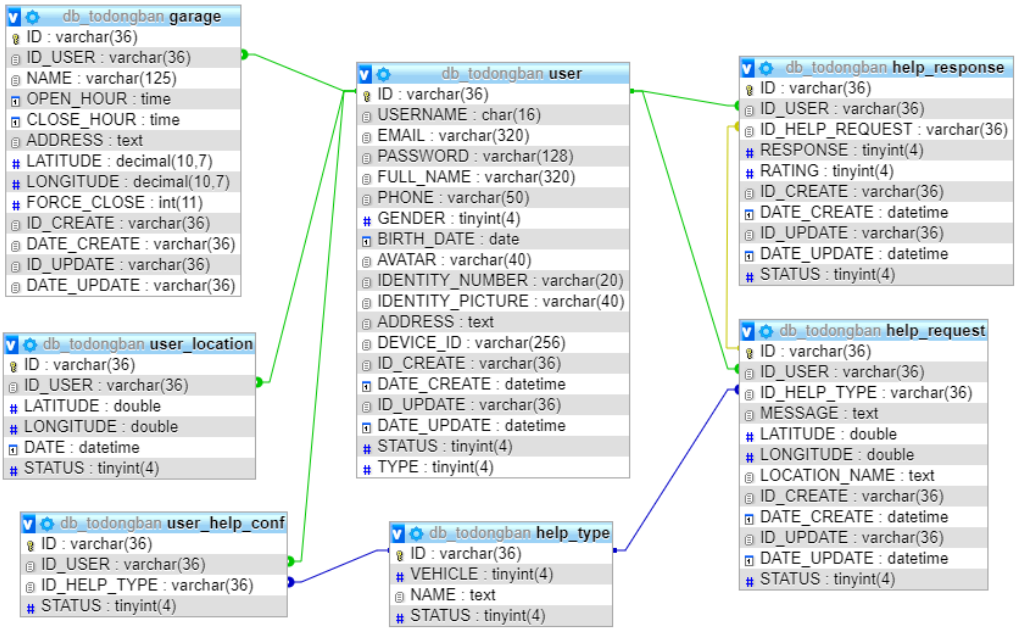
Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berikut:

1. Tampilan antarmuka aplikasi *server* atau web admin hanya berupa *prototype*.
2. Aplikasi harus berjalan dengan terkoneksi internet sebagai media pertukaran data.
3. Untuk mendapatkan lokasi dari pengguna aplikasi *client* dibutuhkan sensor GPS ataupun koneksi internet untuk mendapatkan lokasi latitude dan longtitude dari pengguna.
4. Akurasi dari hasil mendapatkan lokasi bergantung terhadap masing-masing perangkat, cuaca dan juga lokasi.
5. Implementasi peta menggunakan Google Maps Android API.

5.3 Implementasi basis data

Implementasi basis data perancangan perangkat lunak yang berjalan pada sisi *server* menggunakan *Database Management System* MySQL. Terdapat tujuh tabel pada perancangan basis data yaitu tabel *users*, *garage*, *user_location*, *help_type*, *user_help_conf*, *help_request*, dan *help_response*. Tabel-tabel tersebut dibuat

sesuai dengan perancangan basis data yang dibuat berdasarkan perancangan basis data. Implementasi basis data dapat ditunjukkan pada Gambar 5.1.



Gambar 5.1 Implementasi basis data

1. Implementasi tabel *user*

Gambar 5.2 menjelaskan struktur table *user*. Tabel *user* digunakan untuk menyimpan data pengguna.

#	Name	Type	Collation	Attributes	Null	Default
1	ID	varchar(36)	latin1_swedish_ci		No	None
2	USERNAME	char(16)	latin1_swedish_ci		No	None
3	EMAIL	varchar(320)	latin1_swedish_ci		No	None
4	PASSWORD	varchar(128)	latin1_swedish_ci		No	None
5	FULL_NAME	varchar(320)	latin1_swedish_ci		No	None
6	PHONE	varchar(50)	latin1_swedish_ci		No	None
7	GENDER	tinyint(4)			No	None
8	BIRTH_DATE	date			No	None
9	AVATAR	varchar(40)	latin1_swedish_ci		No	None
10	IDENTITY_NUMBER	varchar(20)	latin1_swedish_ci		No	None
11	IDENTITY_PICTURE	varchar(40)	latin1_swedish_ci		No	None
12	ADDRESS	text	latin1_swedish_ci		No	None
13	DEVICE_ID	varchar(256)	latin1_swedish_ci		Yes	NULL
14	ID_CREATE	varchar(36)	latin1_swedish_ci		Yes	NULL
15	DATE_CREATE	datetime			Yes	NULL
16	ID_UPDATE	varchar(36)	latin1_swedish_ci		Yes	NULL
17	DATE_UPDATE	datetime			Yes	NULL
18	STATUS	tinyint(4)			No	None
19	TYPE	tinyint(4)			No	None

Gambar 5.2 Implementasi tabel *user*

2. Implementasi tabel *garage*

Gambar 5.3 menjelaskan struktur tabel *garage*. Tabel *garage* digunakan untuk menyimpan data detail bengkel dari pengguna dengan tipe bengkel.

#	Name	Type	Collation	Attributes	Null	Default
1	ID	varchar(36)	latin1_swedish_ci		No	*None*
2	ID_USER	varchar(36)	latin1_swedish_ci		Yes	*NULL*
3	NAME	varchar(125)	latin1_swedish_ci		No	*None*
4	OPEN_HOUR	time			No	*None*
5	CLOSE_HOUR	time			No	*None*
6	ADDRESS	text	latin1_swedish_ci		No	*None*
7	LATITUDE	decimal(10,7)			Yes	*NULL*
8	LONGITUDE	decimal(10,7)			Yes	*NULL*
9	FORCE_CLOSE	int(11)			No	0
10	ID_CREATE	varchar(36)	latin1_swedish_ci		Yes	*NULL*
11	DATE_CREATE	varchar(36)	latin1_swedish_ci		Yes	*NULL*
12	ID_UPDATE	varchar(36)	latin1_swedish_ci		Yes	*NULL*
13	DATE_UPDATE	varchar(36)	latin1_swedish_ci		Yes	*NULL*

Gambar 5.3 Implementasi tabel *garage*

3. Implementasi tabel *user_location*

Gambar 5.4 menjelaskan struktur tabel *user_location*. Tabel *user_location* digunakan untuk menyimpan lokasi pengguna.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ID	varchar(36)			No	*None*	
2	ID_USER	varchar(36)			No	*None*	
3	LATITUDE	double			No	*None*	
4	LONGITUDE	double			No	*None*	
5	DATE	datetime			No	*None*	
6	STATUS	tinyint(4)			No	*None*	

Gambar 5.4 Implementasi tabel *user_location*

4. Implementasi tabel *help_type*

Gambar 5.5 menjelaskan struktur tabel *help_type*. Tabel *help_type* digunakan untuk menyimpan data jenis jenis bantuan yang dapat digunakan.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ID	varchar(36)			No	*None*	
2	VEHICLE	tinyint(4)			No	*None*	
3	NAME	text			No	*None*	
4	STATUS	tinyint(4)			No	*None*	

Gambar 5.5 Implementasi tabel *help_type*

5. Implementasi tabel *user_help_conf*

Gambar 5.6 menjelaskan struktur tabel *user_help_conf*. Tabel *user_help_conf* digunakan untuk menyimpan jenis bantuan yang dapat dibantu oleh pengguna.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ID	varchar(36)			No	None	
2	ID_USER	varchar(36)			No	None	
3	ID_HELP_TYPE	varchar(36)			No	None	
4	STATUS	tinyint(4)			No	None	

Gambar 5.6 Implementasi tabel *user_help_conf*

6. Implementasi tabel *help_request*

Gambar 5.7 menjelaskan struktur tabel *help_request*. Tabel *help_request* digunakan untuk menyimpan data permintaan bantuan yang diajukan pengguna.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ID	varchar(36)			No	None	
2	ID_USER	varchar(36)			No	None	
3	ID_HELP_TYPE	varchar(36)			No	None	
4	MESSAGE	text			Yes	NULL	
5	LATITUDE	double			No	None	
6	LONGITUDE	double			No	None	
7	LOCATION_NAME	text			Yes	NULL	
8	ID_CREATE	varchar(36)			No	None	
9	DATE_CREATE	datetime			No	None	
10	ID_UPDATE	varchar(36)			Yes	NULL	
11	DATE_UPDATE	datetime			Yes	NULL	
12	STATUS	tinyint(4)			No	None	

Gambar 5.7 Implementasi tabel *help_request*

7. Implementasi tabel *help_response*

Gambar 5.8 menjelaskan struktur tabel *help_response*. Tabel *help_response* digunakan untuk menyimpan data respon dari pengguna atas permintaan bantuan yang terjadi.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ID	varchar(36)			No	None	
2	ID_USER	varchar(36)			No	None	
3	ID_HELP_REQUEST	varchar(36)			No	None	
4	RESPONSE	tinyint(4)			No	None	
5	RATING	tinyint(4)			Yes	NULL	
6	ID_CREATE	varchar(36)			Yes	NULL	
7	DATE_CREATE	datetime			Yes	NULL	
8	ID_UPDATE	varchar(36)			Yes	NULL	
9	DATE_UPDATE	datetime			Yes	NULL	
10	STATUS	tinyint(4)			No	None	

Gambar 5.8 Implementasi tabel *help_response*

5.4 Implementasi kode program

Implementasi kode program ini merupakan tahap penulisan kode program berdasarkan fitur yang terdapat pada aplikasi pecari dan pemberian bantuan berdasarkan lokasi terdekat yang diberi nama Todong Ban. Implemetasi kode program didapat dari hasil analisis pada perancangan *class diagram* dan *sequence*

diagam. Kode program menggunakan bahasa pemrograman Java untuk aplikasi *client* dan PHP pada sisi *server*.

5.4.1 Kode program pendaftaran

Berikut ini pada Kode 5.1 adalah potongan kode program yang digunakan untuk melakukan proses pendaftaran, dimana proses pendaftaran memiliki 3 tampilan melalui *Fragment* dan 1 *Activity* untuk melakukan proses.

```
1   public class SignupActivity extends AppCompatActivity
2   implements
3   SignupBasicFragment.OnFragmentSignupBasicInteractionListener,
4           SignupTypeFragment.OnFragmentTypeInteractionListener,
5           SignupGarageFragment.OnFragmentInteractionListener {
…   ……
6       private void goToPageBasic() {
7           mFragmentManager.beginTransaction()
8                           .add(mFrameLayoutSignUp.getId(),
9   SignupBasicFragment.newInstance(null))
10                          .commit();
11      }
12      private void goToPageType(User user) {
13          mFragmentManager.beginTransaction()
14                          .replace(mFrameLayoutSignUp.getId(),
15  SignupTypeFragment.newInstance(user))
16                          .addToBackStack("SignupTypeFragment")
17                          .commit();
18      }
19      private void goToPageGarage(User user) {
20          mFragmentManager.beginTransaction()
21                          .replace(mFrameLayoutSignUp.getId(),
22                          SignupGarageFragment.newInstance(user))
23                          .addToBackStack("SignupGarageFragment")
24                          .commit();
25      }
26      private void submitNewAccountPersonal(User user) {
27          mFrameLayoutSignUp.setVisibility(View.GONE);
28          mDialog.show();
29          user = prepareUserToken(user);
30          List<MultipartBody.Part> partImages = new ArrayList<>(
31                  prepareSignupImage(user.getAvatarUri(),
```

```
user.getIdentityUri()));
        Map<String, RequestBody> signupData = new HashMap<>(
                NetworkHelpers.prepareMapPart(user));
        mApi.signup(signupData, partImages)
            .enqueue(new ApiResponse<>(
                    this::signupSuccess,this::signupFailure));
    }
    private void submitNewAccountGarage(Garage garage) {
        mFrameLayoutSignUp.setVisibility(View.GONE);
        mDialog.show();
        User user = prepareUserToken(garage.getUser());
        List<MultipartBody.Part> partImages = new ArrayList<>(
                prepareSignupImage(user.getAvatarUri(),
                    user.getIdentityUri()));
        garage.setUser(null);
        Map<String, RequestBody> signupData = new HashMap<>(
                 NetworkHelpers.prepareMapPart(garage));
        signupData.putAll(NetworkHelpers.prepareMapPart(user));
        mApi.signup(signupData, partImages)
            .enqueue(new ApiResponse<>(
               this::signupSuccess, this::signupFailure));
    }
    @Override
    public void onButtonBasicNextClicked(User userData) {
        goToPageType(userData);
    }
    @Override
    public void onButtonTypeClicked(User userData) {
        if (userData.getType() == User.TYPE_PERSONAL)
             submitNewAccountPersonal(userData);
        else if (userData.getType() == User.TYPE_GARAGE)
             goToPageGarage(userData);
    }
    @Override
    public void onButtonGarageSubmitClicked(Garage garage) {
        submitNewAccountGarage(garage);
    }
}
```

Kode 5.1 Kode program SignupActivity untuk proses pendaftaran pada client

Kode 5.1 merupakan potongan kode program untuk melakukan proses pendaftaran yang di proses dalam *SignupActivity* dan memiliki 3 *Fragment* yang akan menangani tampilan, dan berkomunikasi dengan *Activity* melalui implementasi interface pada *Fragment* tersebut. Penjelasan dari Kode 5.1 adalah sebagai berikut:

1. Pada *class SignupActivity* mengimplementasi *interface* yang berupa *listener* dari *fragment* yang digunakan untuk mendapatkan hasil interaksi dari *fragment* tersebut.
2. Baris 6-11 berfungsi untuk mengganti tampilan menjadi formulir awal / *basic form* yang pada kasus ini ditangani oleh *SignupBasicFragment*
3. Baris 12-18 berfungsi untuk mengganti tampilan menjadi formulir pemilihan jenis pendaftaran personal / bengkel yang pada kasus ini ditangani oleh *SignupTypeFragment*.
4. Baris 19-25 berfungsi untuk mengganti tampilan menjadi formulir bengkel yang pada kasus ini ditangani oleh *SignupGarageFragment.*
5. Baris 26-38 berfungsi untuk melakukan pendaftaran akun personal dengan parameter data dari pengguna. Dimana pada baris 29 data *user* diubah dengan data baru dengan tambahan token. Pada baris 30-34 adalah kode untuk mempersiapkan data yang akan dikirim menuju *server* dan kode pada baris 35-37 adalah proses koneksi pengiriman data menuju *server* dimana bila sukses akan dipanggil fungsi *signupSuccess* dan jika gagal dipanggil fungsi *signupFailure*.
6. Baris 39-53 berfungsi untuk melakukan pendaftaran akun bengkel dengan parameter data dari bengkel. Dimana pada baris 42 data *user* pada *garage* diubah dengan data baru dengan tambahan token. Pada baris 43-49 adalah kode untuk mempersiapkan data yang akan dikirim menuju *server* dan kode pada baris 50-52 adalah proses koneksi pengiriman data menuju *server* dimana bila sukses akan dipanggil fungsi *signupSuccess* dan jika gagal dipanggil fungsi *signupFailure*.
7. Baris 55-57 adalah fungsi dari *interface* yang terdapat pada *SignupBasicFragment* yang berguna untuk menangani interkasi dari fragment tersebut saat tombol selanjutnya ditekan, dan pada baris 56 akan mengganti tampilan dengan memanggil fungsi *goToPageType*
8. Baris 59-64 adalah fungsi dari *interface* yang terdapat pada *SignupTypeFragment* yang berguna untuk menangani interaksi dari fragment tersebut saat pengguna memilih tombol jenis pendafaran. Pada baris 60-61 akan berjalan saat pengguna memilih tombol pendaftaran personal dimana akan memanggil proses *submitNewAccountPersonal*, sedangkan pada baris 62-63 adalah kondisi lainnya jika pengguna memilih tombol pendaftaran bengkel dan akan mengganti tampilan dengan memanggil fungsi *goToPageGarage*.

9. Baris 66-68 adalah fungsi dari *interface* yang terdapat pada *SignupGarageFragment* yang berguna untuk menangani interkasi dari fragment tersebut saat tombol ditekan, dimana fungsi ini akan menjalankan fungsi *submitNewAccountGarage* untuk memproses pendaftaran bengkel.

```
1  public function signup_post()
2      {
3          $username = $this->input->post('username');
4          $email = $this->input->post('email');
5          $password = $this->input->post('password');
           $full_name = $this->input->post('full_name');
6          $phone = $this->input->post('phone');
7          $gender = $this->input->post('gender');
8          $birth_date = $this->input->post('birth_date');
9          $identity_number =
10           $this->input->post('identity_number');
11         $address = $this->input->post('address');
12         $device_id = $this->input->post('device_id');
13         $type = $this->input->post('type');
14         if (empty($username) || empty($email) ||
15 empty($password) || empty($full_name) || empty($phone) ||
16 empty($gender) || empty($identity_number) || empty($birth_date)
17 || empty($address) || empty($device_id) || empty($type)) {
18             $this->response_error(STATUS_CODE_KEY_EXPIRED,
19 'Data not complete');
20         }
21         if (strlen($username) > 16) {
22             $this->response_error(STATUS_CODE_KEY_EXPIRED,
23  'Username length exceeded');
24         }
25         $this->load->model('m_user');
26         $id = $this->m_user->generate_id();
27         if (!$this->m_user->check_username($username)) {
28             $this->response_error(STATUS_CODE_KEY_EXPIRED,
29  'Username exist');
30         }
31         $this->load->helper('string');
32         $this->load->library('upload');
```

```
33          $avatar_name = 'AVA_' . random_string('alnum', 16);
34          $upload_path = FCPATH . '/uploads';
35          if (!is_dir($upload_path)) {
36              mkdir($upload_path);
37          }
38          $upload_path = $upload_path . '/' . $username;
39          if (!is_dir($upload_path)) {
40              mkdir($upload_path);
41          }
42          $upload_config = [
43              'upload_path' => $upload_path,
44              'allowed_types' => '*',
45              'overwrite' => true,
46              'file_ext_tolower' => true
47          ];
48          $upload_config['file_name'] = $avatar_name;
49          $this->upload->initialize($upload_config);
50          if (!$this->upload->do_upload('avatar')) {
51              $this->response_error(STATUS_CODE_KEY_EXPIRED,
52  'Failed upload profile picture');
53          }
54          $avatar = $this->upload->data('file_name');
55          $identity_name = 'IDP_' . random_string('alnum', 16);
56          $upload_config['file_name'] = $identity_name;
57          $this->upload->initialize($upload_config);
58          if (!$this->upload->do_upload('identity_picture')) {
59              // Delete uploaded avatar and the directory
60              unlink($upload_path . '/' . $avatar);
61              rmdir($upload_path);
62              $this->response_error(STATUS_CODE_KEY_EXPIRED,
63              'Failed upload identity picture');
64          }
65          $identity = $this->upload->data('file_name');
66          $result = $this->
67            m_user->create($id, $username, $email, $password,
68                           $full_name, $phone, $gender,
69                           $birth_date, $avatar,
70                           $identity_number, $identity, $address,
```

```
71                          $device_id,
72                          User_data::$STATUS_NOT_ACTIVE, $type);
73          if ($result) {
74              if ($type == 2) {
75                  $garage_name = $this->
76                    input->post('garage_name');
77                  $garage_open = $this->input->
78                    post('garage_open');
79                  $garage_close = $this->input->
80                    post('garage_close');
81                  $garage_address = $this->input->
82                    post('garage_address');
83                  $garage_latitude = $this->input->
84                    post('garage_latitude');
85                  $garage_longitude = $this->input->
86                    post('garage_longitude');
87                  if (empty($garage_name) && empty($garage_open)
88           && empty($garage_close) && empty($garage_address)
89           && empty($garage_latitude) &&
90           empty($garage_longitude)) {
91                      $this->m_user->delete_pure($id);
92                      $this->
93          response_error(STATUS_CODE_SERVER_ERROR,
94          'Data Garage not complete');
95                  }
96                  $this->load->model('m_garage');
97                  $result_garage = $this->
98          m_garage->create($id, $garage_name, $garage_open,
99                  $garage_close, $garage_address,
100                 $garage_latitude, $garage_longitude);
101                 if (!$result_garage) {
102                     $this->m_user->delete_pure($id);
103                     $this->
104         response_error(STATUS_CODE_SERVER_ERROR,
105         'Something wrong when create user');
106                 }
107             }
…               ……
```

```
108             $this->response(['message' =>
109                 'Success create user',
110                 'token' => $this->create_token(
111                         [$username, $device_id])]);
112         } else {
113             $this->response_error(STATUS_CODE_SERVER_ERROR,
114         'Something wrong when create user');
115         }
116     }
```

Kode 5.2 Kode program untuk proses pendaftaran pada web service/*server*

Kode 5.2 merupakan potongan kode program untuk menangani proses pendaftaran pada sisi *server* / *web service*. Penjelasan dari Kode 5.2 adalah sebagai berikut:

1. Pada baris 1 menjelaskan bahwa fungsi ini dapat diakses dengan nama signup dan dengan metode POST.
2. Baris 3-13 adalah penanganan data pendaftaran pengguna yang diberikan dengan metode POST, dimana pada baris 14-20 bila data terdapat yang kosong akan memberikan pesan kesalahan dan pada baris 21-24 jika panjang *username* melebihi batas maksimal 16 maka juga akan meberikan pesan kesalahan.
3. Baris 27-30 berfungsi sebagai proses pengecekan ketersediaan *username* dan jika *username* telah digunakan maka akan memberikan pesan kesalahan.
4. Baris 33-54 berfungsi sebagai proses penanganan proses *upload avatar* dari pegguna dan pada baris 50-53 berfungsi sebagai pemberian pesan kesalahan jika gagal melakukan proses *upload avatar*.
5. Baris 55-64 berfungsi sebagai proses penanganan proses *upload* kartu identitas dari pegguna pegguna dan pada baris 58-64 berfungsi sebagai pemberian pesan kesalahan jika gagal melakukan proses *upload* kartu identitas dan menghapus *avatar* yang telah di *upload* sebelumnya.
6. Baris 66-72 berfungsi sebagai proses penyimpanan data pengguna.
7. Baris 74-107 adalah proses penanganan pendaftaran bengkel setelah proses penyimpanan data pengguna berhasil.
8. Baris 108-111 berfungsi untuk memberikan pesan sukses kepada *client* dengan isi data token yang dapat digunakan untuk penanda jika telah *login*.
9. 112-115 adalah proses yang dijalankan saat gagal dalam melakukan penyimpanan data, dan akan memberikan pesan kesalahan.

5.4.2 Kode program pencarian bantuan

Berikut ini pada Kode 5.3 hingga Kode 5.5 adalah potongan kode program yang digunakan untuk melakukan proses pencarian bantuan, dimana proses pencarian bantuan ditangani sebuah *Activity* untuk melakukan proses dan *fragment* untuk menangani tampilannya.

```
1   public class MainActivityPersonal extends AppCompatActivity
2   implements
3   PersonalRequestHelpFragment.OnFragmentInteractionListener,
4   PersonalProcessHelpFragment.OnFragmentInteractionListener,
5   PersonalResponseHelpFragment.OnFragmentInteractionListener {
…   ……
6       private void initFragment() {
7           if (!mHelpProcess && !mHelpResponse) {
8               goToPageRequestHelp();
9           } else if (mHelpProcess) {
10              goToPageProcessHelp(null);
11          } else {
12              goToPageDetailResponseHelp(null);
13          }
14      }
15      private void goToPageRequestHelp() {
16          mFragment = PersonalRequestHelpFragment.newInstance();
17          changePage();
18      }
19      private void helpRequestSuccess(Response<SingleStringData>
20          response) {
21          SingleStringData body = response.body();
22          if (body != null) {
23              if (mFragment instanceof
24                      PersonalProcessHelpFragment) {
25                  mHelpProcessId = body.getData();
26                  mSharedPreferences.edit()
27      .putString(Values.SHARED_PREFERENCES_KEY_ID_HELP_PROCESS,
28          mHelpProcessId)
29                  .apply();
30                  ((PersonalProcessHelpFragment) mFragment)
31                  .setHelpRequestId(mHelpProcessId);
32              }
```

```
33          }
34      }
35      @Override
36      public void onHelpRequested(RequestHelp requestHelp) {
37          mHelpProcess = true;
38          registerReceiver(mBroadcastReceiver, mIntentFilter);
39          mSharedPreferences.edit()
40      .putBoolean(Values.SHARED_PREFERENCES_KEY_IN_HELP_PROCESS,
41        mHelpProcess)
42      .putString(Values.SHARED_PREFERENCES_KEY_TYPE_HELP_PROCESS,
43        requestHelp.getSelectedHelpType())
44          .apply();
45          goToPageProcessHelp(requestHelp);
46          mApi.requestHelp(requestHelp.getLocationLatitude(),
47            requestHelp.getLocationLongitude(),
48            requestHelp.getSelectedHelpType(),
49            requestHelp.getMessage(),
50            requestHelp.getLocationName())
51            .enqueue(new ApiResponse<>(
52          this::helpRequestSuccess, this::helpRequestFailure));
53      }
54      @Override
55      public void pushUpdateLocation(
56          double latitude, double longitude) {
57          mApi.updateLocation(latitude, longitude)
58              .enqueue(new ApiResponse<>());
59      }
60      private class FoundHelpersBroadcastReceiver
61          extends BroadcastReceiver {
62          @Override
63          public void onReceive(Context context, Intent intent) {
64              PeopleHelp peopleHelp = intent
65                  .getParcelableExtra(ARGS_BROADCAST_DATA);
66              if (mFragment instanceof
67                      PersonalProcessHelpFragment) {
68                  ((PersonalProcessHelpFragment) mFragment)
69                      .addData(peopleHelp);
70              }
```

```
71              }
72          }
…       ……
73      }
```

Kode 5.3 Kode program *MainActivityPersonal* untuk proses pencarian bantuan pada client

Kode 5.3 merupakan potongan kode program *Activity* untuk menangani proses pencarian bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1-5 menunjukan *MainActivityPersonal* mengimplementasi *interface* yang berupa *listener* dari *fragment* yang digunakan untuk mendapatkan hasil interaksi dari *fragment* tersebut.
2. Baris 6-14 berfungsi sebagai penentu tampilan yang akan ditampilkan dengan kondisi pada baris 7-8 jika tidak dalam proses apapun maka akan membuat tampilan pencarian bantuan dengan memanggil fungsi *goToPageRequestHelp,* sedangkan jika dalam proses pencarian bantuan seperti baris 9-10 akan memanggil tampilan *goToPageProcessHelp* yang digunakan untuk menampilkan daftar pemberi bantuan, selain itu jika dalam proses telah memilih respon maka akan memanggil tampilan *goToPageDetailResponseHelp* untuk menampilkan detail pemberi bantuan.
3. Baris 15-18 berfungsi untuk membuat fragment sebagai *PersonalRequestHelpFragment* yang dapat menampilkan tampilan pencarian bantuan dan merubah tampilannya pada baris 17 dengan memanggil fungsi *changePage*.
4. Baris 19-34 berfungsi sebagai respon sukses saat berhasil melakukan koneksi dengan *web service* untuk melakukan pencarian bantuan. Dimana pada baris 26-29 data yang didapatkan dari respon tersebut disimpan pada *sharedPreferences* agar dapat digunakan lain waktu. Pada baris 30-31 memanggil fragment yang sedang berjalan yang kemudian di *casting* menjadi *PersonalProcessFragment* dan memanggil fungsi *public setHelpRequestId* untuk memberikan data respon tersebut kepada fragment yang sedang berjalan.
5. Baris 36-53 adalah hasil implementasi dari *interface* milik *PersonalRequestHelpFragment* yang berfungsi sebagai jembatan komunikasi dari *fragment* menuju *activity* induknya untuk memberitahukan bahwa pengguna telah memilih jenis bantuan. Dimana pada barus 37 merubah status sekarang sebagai proses bantuan, baris 38 menjalankan *BroadCastReceiver* yang ada dalam *activity*. Barus 39-44 menyimpan status sekarang dan menyimpan jenis bantuan yang di pilih dalam *sharedPreferences*. Baris 45 merubah tampilan sekarang dengan memanggil *goToPageProcessHelp* dan pada baris 45-52 menjalankan proses koneksi menuju *web service*

menggunakan fungsi *requestHelp* dengan mengirimkan data *longitude, latitude,* jenis bantuan, pesan dan nama lokasi yang kemudian responnya akan diteruskan menuju fungis *helpRequestSuccess*.

6. Baris 55-59 berfungsi membuka koneksi menuju *web service* untuk melakukan perubahan lokasi dengan mengirimkan lokasi terkini berdasarkan *latitude* dan *longitude*, fungsi ini merupakan implementasi dari *interface* milik fragment *PersonalRequestHelpFragment*.
7. Baris 60-72 merupakan suatu *BroadCastReceiver* yang berfungsi menerima *broadcast* untuk melakukan perubahan data pada *PersonalProcessHelpFragment*. Broadcast ini dipanggil saat terdapat kiriman data dari *web service* berupa notifikasi.

```
1   public class PersonalRequestHelpFragment extends Fragment
2   implements MapFragment.OnFragmentCompleteInteractionListener {
…   ……
3       @Override
4       public void onMyLocationReady() {
5           if (mMyLocationFab.getVisibility() == View.GONE) {
6               mMyLocationFab.setVisibility(View.VISIBLE);
7           }
8           if (mListener != null) {
9               mListener.pushUpdateLocation(
10                  mMap.getLatitude(), mMap.getLongitude());
11          }
12      }
13      private void selectableVehicleType(View view) {
14          if (mBottomSheetBehavior.getState() ==
15              BottomSheetBehavior.STATE_COLLAPSED) {
16              showMyLocationButton(false);
17              mBottomSheetBehavior
18              .setState(BottomSheetBehavior.STATE_EXPANDED);
19          }
20          if (view.getId() == R.id.selectable_type_motorcycle) {
21              mSelectedVehicle = HelpType.VEHICLE_MOTORCYCLE;
22          } else if (view.getId() == R.id.selectable_type_car) {
23              mSelectedVehicle = HelpType.VEHICLE_CAR;
24          }
25          mAdapter.setVehicleType(mSelectedVehicle);
26      }
```

```
27      private void selectedHelpType(String helpTypeId) {
28          mRequestHelp = new RequestHelp(helpTypeId);
29          String helpConfirmationMessage = getString(
30                  R.string.help_confirmation_message,
31                  getString(mRequestHelp.getVehicleName()),
32                  getString(mRequestHelp.getHelpTypeName()));
33          new MaterialDialog.Builder(mContext)
34                  .title(R.string.help_confirmation_title)
35                  .content(helpConfirmationMessage)
36                  .input(R.string.help_confirmation_message_hint,
37                      0, true, (dialog, input) -> {})
38                  .positiveText(R.string.ok)
39                  .negativeText(R.string.cancel)
40                  .onNegative((dialog, which) -> {
41                      mRequestHelp = null;
42                      showBottomSheet(true);
43                      dialog.dismiss();
44                  })
45                  .onPositive((dialog, which) -> {
46                   mRequestHelp.setMessage(dialog
47                     .getInputEditText().getText().toString());
48                   mRequestHelp
49                     .setLocationLatitude(mMap.getLatitude());
50                   mRequestHelp
51                     .setLocationLongitude(mMap.getLongitude());
52                   mRequestHelp
53                     .setLocationName(mLocationName
54                        .getText().toString());
55                   dialog.dismiss();
56                   if (mListener != null) {
57                         mListener.onHelpRequested(mRequestHelp);
58                      }})
59                  .cancelable(false)
60                  .canceledOnTouchOutside(false)
61                  .autoDismiss(false)
62                  .show();
63      }
…   ……
```

```
64  }
```

Kode 5.4 Kode program *PersonalRequestHelpFragment* untuk proses pencarian bantuan pada client

Kode 5.4 merupakan potongan kode program *Fragment* untuk menampilkan halaman pencarian bantuan. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1-2 *PersonalRequestHelpFragment* mengimplementasi *interface* yang berupa *listener* dari *fragment* yang digunakan untuk mendapatkan hasil interaksi dari *fragment* tersebut.
2. Baris 4-12 adalah fungsi hasil implementasi *interface* milik *MapFragment* yang dijalankan saat telah berhasil mendapatkan lokasi sekarang, dimana pada baris 8-11 akan memanggil *listener* untuk melakukan update lokasi dengan memberikan masukan *latitude* dan *longitude* dari posisi sekarang.
3. Baris 13-26 merupakan fungsi yang dijalankan saat pengguna memilih jenis kendaraan, dimana barus 14-19 akan membuka *BottomSheet* yang berisi RecyclerView atau list jenis permasalahan. Baris 20-24 akan menyimpan jenis kendaraan yang dipilih kedalam variabel *mSelectedVehicle* dan baris 25 akan memberitahukan *adapter* yang menangani *RecyclerView* mengenai jenis kendaraan yang dipilih.
4. Baris 27-63 berfungsi sebagai penjalan proses saat pengguna memilih jenis permasalahan. Dimana pada baris 28 akan menginialisasi variabel *mRequestHelp* dengan jenis permasalahan yang dipilih sebagai parameternya, lalu pada baris 29-62 akan menyiapkan dialog konfirmasi beserta inputan pesan yang akan diberikan pengguna. Baris 45-58 merupakan proses yang dijalankan apabila pengguna menekan tombol konfirmasi sehingga pada baris 46-54 dilakukan pengisian nilai kepada variabel *mRequestHelp* yang kemudian dikirimkan menggunalan *listener* melalui fungsi *onHelpRequested*.

```
1   public function request_post(){
2       if (!$this->check_access()) {
3           $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
4           'Cant access');
5       }
6       $latitude = $this->input->post('latitude');
7       $longitude = $this->input->post('longitude');
8       $help_type = $this->input->post('help_type');
9       $message = $this->input->post('message');
10      $location_name = $this->input->post('location_name');
```

```
11      if (empty($latitude) || empty($longitude) ||
12          empty($help_type)) {
13          $this->response_error(STATUS_CODE_KEY_EXPIRED,
14            'Data not complete');
15      }
16      $user = $this->get_user();
17      $result = $this->m_help->insert_request($user->ID,
18        $latitude, $longitude, $help_type, $message,
19        $location_name);
20      $handler = new \GuzzleHttp\Handler\CurlMultiHandler();
21      $client = new \GuzzleHttp\Client(['curl' =>
22       [CURLOPT_SSL_VERIFYPEER => false, CURLOPT_TIMEOUT => 1],
23       'handler' => \GuzzleHttp\HandlerStack::create($handler)]);
24      $promise = $client->requestAsync('POST',
25        base_url('api/help/helper_search'),
26        ['headers' => ['Authorization' => 'Bearer ' .
27          $this->get_token()],
28          'form_params' => ['id_request' => $result,]]);
29      $handler->execute();
30      $this->response($result);
31  }
32  public function helper_search_post(){
33      ini_set('max_execution_time', 100);
34      if (!$this->check_access()) {
35          $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
36           'Cant access');
37      }
38      $id_request = $this->input->post('id_request');
39      if (empty($id_request)) {
40          $this->response_error(STATUS_CODE_KEY_EXPIRED,
41            'Data not complete');
42      }
43      $user = $this->get_user();
44      $helps = $this->m_help->get_request($id_request,
45        $user->ID);
46      if (!empty($helps)) {
47          $help = $helps[0];
48          $this->load->model('m_location');
```

```
49          $nearby_garages = $this->m_location->
50            get_nearby_garage($user->ID,
51            $help->ID_HELP_TYPE, $help->LATITUDE,
52            $help->LONGITUDE, $help->DATE_CREATE);
53          // Lets save help response from garage
54          $id_insert_garages = [];
55          $generated_id = $this->m_location->
56            generate_id(count($nearby_garages));
57          if (is_array($generated_id)) {
58              $id_insert_garages = $generated_id;
59          } else {
60              $id_insert_garages[] = $generated_id;
61          }
62          $insert_garages = [];
63          $date_now = date('Y-m-d H:i:s');
64          for ($i = 0; $i < count($nearby_garages); $i++) {
65              $garage = $nearby_garages[$i];
66              $garage->__cast();
67              $insert_garages[] = […];
68          }
69          $this->m_help->insert_response($insert_garages);
70          $this->load->library('fcm');
71          // Lets send notification to user and user
72          for ($i = 0; $i < count($nearby_garages); $i++) {
73              $garage = $nearby_garages[$i];
74              $garage->__cast();
75              $this->fcm->set_target($user->DEVICE_ID)
76                        ->set_code(Fcm::CODE_HELP_RESPONSE)
77                        ->set_payloads([
78                          'id' => $id_insert_garages[$i],
79                          'name' => $garage->NAME,
80                          'badge' => $this->m_help->
81                                  badge($garage->ID_USER),
82                          'distance' => $garage->DISTANCE,
83                          'user_type' => User_data::$TYPE_GARAGE,
84                          'accept' => false
85                          ])
86                        ->send();
```

```
87              $this->fcm->reset();
88              $this->fcm->set_target($garage->DEVICE_ID)
89                        ->set_code(Fcm::CODE_HELP_REQUEST)
90                        ->set_payloads([
91                          'id' => $id_insert_garages[$i],
92                          'name' => $user->FULL_NAME,
93                          'help_type' => $help->ID_HELP_TYPE,
94                          'distance' => $garage->DISTANCE
95                          ])
96                        ->send();
97              $this->fcm->reset();
98          }
99          $nearby_personal = $this->m_location->
100             get_nearby_personal($user->ID,
101               $help->ID_HELP_TYPE,
102               $help->LATITUDE, $help->LONGITUDE);
103         //  Lets save help response from personal
 …          ……
104     }
105 }
```

Kode 5.5 Kode program untuk proses pencarian bantuan pada *web service/server*

Kode 5.5 merupakan potongan kode program untuk menangani permintaan pencarian bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1 menjelaskan bahwa fungsi ini dapat diakses dengan nama request dan dengan metode POST.
2. Baris 2-5 menunjukan jika *request* yang dilakukan tidak memiliki akses baik berupa tidak terdapat token atau token tidak valid maka akan memberikan pesan kesalahan.
3. Baris 6-10 berfungsi untuk menangkap masukan yang diberikan dengan metode POST saat memanggil fungsi ini dimana akan dilakukan pengecekan pada baris 11-15 dan akan memberikan pesan kesalahan apabila data tidak memenuhi.
4. Baris 16 menunjukan pengambilan data *user* dari token yang mengakses fungsi ini.
5. Baris 17-19 akan menyimpan data permintaan bantuan dan mengembalikan *id* dari permintaan bantuan yang disimpan.

6. Baris 20-29 akan melakukan pemanggilan fungsi lain dengan melalui *HTTP Request* menggunakan *curl* secara *asyncrhonus* untuk mencari pengguna disekitar permintaan bantuan. Pemanggilan fungsi yang melalui *curl* bertujuan untuk proses pencarian pengguna untuk memberikan bantuan berjalan di proses berbeda sehingga tidak menunda proses pengembalian *id* dari pencarian bantuan yang akan dikembalikan melalui baris 30.
7. Baris 32 adalah fungsi yang digunakan untuk melakukan pengguna lain disekitar pencari bantuan.
8. Baris 33 dibutuhkan untuk meningkatkan batas waktu eksekusi PHP menjadi 100 detik.
9. Baris 34-36 menunjukan jika *request* yang dilakukan tidak memiliki akses baik berupa tidak terdapat token atau token tidak valid maka akan memberikan pesan kesalahan.
10. Baris 38 akan menangkap masukan dengan metode POST yang diberikan saat memanggil fungsi ini dan akan dilakukan pengecekan pada baris 39-42 yang akan memberikan pesan kesalahan saat data tidak memenuhi.
11. Baris 44 akan mendapatkan detail data dari pencarian bantuan.
12. Baris 49 akan mendapatkan data bengkel sekitar yang sedang buka dan menangani permasalahan sesuai yang dicari dengan menggunakan formula *Haversine* untuk menentukan jaraknya.
13. Baris 54-69 adalah proses penyimpanan data dari bengkel sekitar yang didapatkan sebelumnya kedalam tabel *help_response* dimana baris 54-61 akan membuat *id* terlebih dahulu sebelum memasukan datanya pada baris 69.
14. Baris 72-98 akan memberikan pemberitahuan kepada bengkel yang ditemukan dan pengguna yang mencari bantuan menggunakan FCM.
15. Baris 99 akan mendapatkan data pengguna personal yang berada disekitar pencari bantuan dengan pengaturan dapat menangani jenis bantuan yang diminta dan menggunakan formula *Haversine* untuk menentukan jaraknya.
16. Baris 103-104 akan melakukan hal yang sama dengan proses pada bengkel dengan perbedaan tidak dilakukan pengiriman pemberitahuan pada pengguna pencari bantuan melainkan hanya kepada pengguna sekitar yang ditemukan seblumnya.

5.4.3 Kode program pemilihan pemberi bantuan

Berikut ini pada Kode 5.6 hingga Kode 5.8 adalah potongan kode program yang digunakan untuk melakukan proses pencarian bantuan, dimana proses pencarian bantuan ditangani sebuah *Activity* untuk melakukan proses dan *fragment* untuk menangani tampilannya.

```
1   public class MainActivityPersonal extends AppCompatActivity
2   implements
3   PersonalRequestHelpFragment.OnFragmentInteractionListener,
4   PersonalProcessHelpFragment.OnFragmentInteractionListener,
5   PersonalResponseHelpFragment.OnFragmentInteractionListener {
…   ……
6       private void initFragment() {
7           if (!mHelpProcess && !mHelpResponse) {
8               goToPageRequestHelp();
9           } else if (mHelpProcess) {
10              goToPageProcessHelp(null);
11          } else {
12              goToPageDetailResponseHelp(null);
13          }
14      }
15      private void goToPageProcessHelp(RequestHelp requestHelp) {
16          if (requestHelp == null) {
17              String helpType = mSharedPreferences
18                  .getString(
19      Values.SHARED_PREFERENCES_KEY_TYPE_HELP_PROCESS, null);
20              requestHelp = new RequestHelp(helpType);
21              mHelpProcessId = mSharedPreferences
22                  .getString(
23      Values.SHARED_PREFERENCES_KEY_ID_HELP_PROCESS, null);
24          }
25          mFragment = PersonalProcessHelpFragment
26              .newInstance(requestHelp, mHelpProcessId);
27          registerReceiver(mBroadcastReceiver, mIntentFilter);
28          changePage();
29      }
30    private void responseSelectSuccess(Response<SingleStringData>
31          response) {
32          mDialog.dismiss();
33          mHelpProcess = false;
34          mHelpResponse = true;
35          mSharedPreferences.edit()
36                              .remove(Values
37        .SHARED_PREFERENCES_KEY_PEOPLE_HELP)
```

```
38                              .putBoolean(Values
39        .SHARED_PREFERENCES_KEY_IN_HELP_PROCESS, mHelpProcess)
40                              .putBoolean(Values
41        .SHARED_PREFERENCES_KEY_SELECTED_RESPONE, mHelpResponse)
42                              .apply();
43          goToPageDetailResponseHelp(mHelpProcessId);
44      }
45      @Override
46      public void onSelectedHelper(String responseId) {
47          mDialog.show();
48          mApi.responseSelect(responseId).enqueue(new
49              ApiResponse<>(this::responseSelectSuccess,
50              this::responseSelectFailed));
51      }
…   ……
52  }
```

Kode 5.6 Kode program *MainActivityPersonal* untuk proses pemilihan pemberi bantuan pada client

Kode 5.3 merupakan potongan kode program *Activity* untuk menangani proses pencarian bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1-5 menunjukan *MainActivityPersonal* mengimplementasi *interface* yang berupa *listener* dari *fragment* yang digunakan untuk mendapatkan hasil interaksi dari *fragment* tersebut.
2. Baris 6-14 berfungsi sebagai penentu tampilan yang akan ditampilkan dengan kondisi pada baris 7-8 jika tidak dalam proses apapun maka akan membuat tampilan pencarian bantuan dengan memanggil fungsi *goToPageRequestHelp,* sedangkan jika dalam proses pencarian bantuan seperti baris 9-10 akan memanggil tampilan *goToPageProcessHelp* yang digunakan untuk menampilkan daftar pemberi bantuan, selain itu jika dalam proses telah memilih respon maka akan memanggil tampilan *goToPageDetailResponseHelp* untuk menampilkan detail pemberi bantuan.
3. Baris 15-29 berfungsi untuk membuat fragment sebagai *PersonalProcessHelpFragment* yang dapat menampilkan tampilan pemilihan pemberi bantuan dan merubah tampilannya pada baris 28 dengan memanggil fungsi *changePage*. Namun apabila parameter yang diisikan kosong atau *null* maka pada baris 16-24 akan mengambil data dari dalam *sharedPreferences* terlebih dahulu.
4. Baris 30-34 berfungsi sebagai respon sukses saat berhasil melakukan koneksi dengan *web service* untuk melakukan pemilihan pemberi bantuan. Dimana

pada baris 33 -34 akan merubah status menjadi telah memilih respon dengan mengganti nilai *mHelpProcess* menjadi *false* dan *mHelpResponse* menjadi *true*, lalu nilai tersebut disampan di *sharedPreferences* pada baris 35-42 yang kemudian membuka halaman detail respon pada baris 43 dengan memberikan *id* dari request sebagai parameternya.

5. Baris 46-51 adalah hasil dari implementasi dari *interface* milik *PersonalProcessHelpFragment* yang berfungsi sebagai jembatan komunikasi dari *fragment* menuju *activity* induknya untuk memberitahukan bahwa pengguna memilih calon pemberi bantuannya dan memberikan nilai *id* dari respon dalam parameternya. Lalu baris 48-50 akan menjalankan proses koneksi menuju *web service* menggunakan fungsi *responseSelect* dengan mengirimkan data *id* dari respon yang jika sukses akan diteruskan kepada fungsi *responseSelectSuccess*.

```
1   public class PersonalProcessHelpFragment extends Fragment {
…   ……
2       @Override
3       public View onCreateView(LayoutInflater inflater,
4         ViewGroup container,  Bundle savedInstanceState) {
…           ……
5           mAdapterPersonal.setOnClickListener(helpResponseId ->
6                   mListener.onSelectedHelper(helpResponseId));
7           mAdapterGarage.setOnClickListener(helpResponseId ->
8                   mListener.onSelectedHelper(helpResponseId));
…           ……
9       }
10      public void addData(PeopleHelp data) {
11          if (data.getUserType() == User.TYPE_PERSONAL) {
12              mAdapterPersonal.addData(data);
13          } else { mAdapterGarage.addData(data); }
14          NotificationManagerCompat notificationManager =
15              NotificationManagerCompat.from(mContext);
16          notificationManager
17            .cancel(Values.NOTIFICATION_TAG_PEOPLE_HELP,
18                    Values.NOTIFICATION_ID_PEOPLE_HELP);
19          Vibrator vibrator = (Vibrator) mContext
20            .getSystemService(Context.VIBRATOR_SERVICE);
21          if (vibrator != null) { vibrator.vibrate(1000); }
22          toggleViewPeopleHelper();
```

```
23      }
…   ……
24  }
```

Kode 5.7 Kode program *PersonalProcessHelpFragment* untuk proses pemilihan pemberi bantuan pada client

Kode 5.7 merupakan potongan kode program *Fragment* untuk menampilkan halaman pemilihan pemberi bantuan. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Baris 3-9 merupakan potongan program yang dijalankan pertama kali oleh sebuah *fragment* untuk membuat tampilan, pada baris 6-8 ditunjukan bahwa terdapat 2 *adapter* yang berfungsi sebagai pengolah data pada *RecyclerView* yang berisi pemberi bantuan berupa personal dan bengkel melakukan pengaturan *clickListener* dimana akan berjalan saat pengguna menekan salah satu *item* pada *recyclerview* dan kemudian akan memanggil *listener* dengan menjalankan fungsi *onSelecedHelper* dengan parameter *id* dari respon yang dipilih.
2. Baris 10-23 berguna untuk menambahkan data kedalam daftar pemberi bantuan, dimana bersifat *public* karena akan perlu diakses oleh *BroadcastReceiver* yang ada di *class* lain. Pada baris 11-13 ditunjukan apabila data yang diterima melalui parameter merupakan data tipe personal maka data akan ditambahkan kedalam *adapter* pengolah data personal, sedangkan jika tidak akan ditambahkan kedalam *adapter* pengolah data bengkel.
3. Baris 14-18 berfungsi untuk menghapus notifikasi pemberitahuan jika tersedia pemberi bantuan.
4. Baris 19-22 berfungsi untuk menyalakan getaran pada *smartphone*.

```
1   public function select_response_post(){
2       if (!$this->check_access()) {
3           $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
4             'Cant access');
5       }
6       $id_response = $this->input->post('id_response');
7       if (empty($id_response)) {
8           $this->response_error(STATUS_CODE_KEY_EXPIRED,
9             'Data not complete');
10      }
11      $user = $this->get_user();
12      $requests = $this->m_help->
```

```
13          get_request_by_response($id_response);
14      $request = null;
15      if (!empty($requests)) {
16          $request = $requests[0];
17          $request->__cast();
18      }
19      $id_request = $request->ID;
20      if (empty($id_request)) {
21          $this->response_error(STATUS_CODE_KEY_EXPIRED,
22              'Data not complete');
23      }
24      $this->m_help
25          ->update_response($id_request, null, $request->ID_USER,
26                null, null, Help_response_data::$STATUS_SELECTED,
27                $id_response);
28      $this->m_help->process_request($id_request, $user->ID);
29      $this->response(null);
30  }
```

Kode 5.8 Kode program untuk proses pemilihan pemberi bantuan pada *web service/server*

Kode 5.8 merupakan potongan kode program untuk menangani pemilihan pemberi bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1 menjelaskan bahwa fungsi ini dapat diakses dengan nama select_response dan dengan metode POST.
2. Baris 2-5 menunjukan jika *request* yang dilakukan tidak memiliki akses baik berupa tidak terdapat token atau token tidak valid maka akan memberikan pesan kesalahan.
3. Baris 6 berfungsi untuk menangkap masukan yang diberikan dengan metode POST saat memanggil fungsi ini dimana akan dilakukan pengecekan pada baris 7-10 dan akan memberikan pesan kesalahan apabila data tidak memenuhi.
4. Baris 11 menunjukan pengambilan data *user* dari token yang mengakses fungsi ini.
5. Baris 12-18 adalah proses mengambil data request dari *id* respon yang diberikan.
6. Baris 24-27 akan merubah data respond dengan memberikan status sebagai respon yang dipilih
7. Baris 28 akan merubah data *request* menjadi berstatus telah diproses/telah memiliki perespon.

8. Baris 29 akan mengakhiri proses dengan mengirimkan pesan kosong kepada *client*.

5.4.4 Kode program akhiri pencarian bantuan

Berikut ini pada Kode 5.9 hingga Kode 5.11 adalah potongan kode program yang digunakan untuk melakukan melihat detail pemberi bantuan dan proses mengakhiri pencarian bantuan, dimana proses tersebut ditangani sebuah *Activity* untuk melakukan proses dan *fragment* untuk menangani tampilannya.

```
1  public class MainActivityPersonal extends AppCompatActivity
2  implements
3  PersonalRequestHelpFragment.OnFragmentInteractionListener,
4  PersonalProcessHelpFragment.OnFragmentInteractionListener,
5  PersonalResponseHelpFragment.OnFragmentInteractionListener {
…  ……
6      private void initFragment() {
7          if (!mHelpProcess && !mHelpResponse) {
8              goToPageRequestHelp();
9          } else if (mHelpProcess) {
10             goToPageProcessHelp(null);
11         } else {
12             goToPageDetailResponseHelp(null);
13         }
14     }
15     private void goToPageDetailResponseHelp(
16         String helpProcessId) {
17         if (helpProcessId == null) {
18             mHelpProcessId = mSharedPreferences
19         .getString(Values
20           .SHARED_PREFERENCES_KEY_ID_HELP_PROCESS, null);
21         } else {
22             mHelpProcessId = helpProcessId;
23         }
24         mFragment = PersonalResponseHelpFragment
25             .newInstance(mHelpProcessId);
26         changePage();
27     }
28   private void finishHelpSuccess(Response<SingleStringData>
29         response) {
```

```
30          Toast.makeText(this, "Pencarian bantuan telah selesai",
31            Toast.LENGTH_SHORT).show();
32          mHelpProcessId = null;
33          mHelpProcess = false;
34          mHelpResponse = false;
35          mSharedPreferences.edit()
36                              .putString(Values
37        .SHARED_PREFERENCES_KEY_ID_HELP_PROCESS, mHelpProcessId)
38                              .putBoolean(Values
39        .SHARED_PREFERENCES_KEY_IN_HELP_PROCESS, mHelpProcess)
40                              .putBoolean(Values
41        .SHARED_PREFERENCES_KEY_SELECTED_RESPONE, mHelpResponse)
42                              .apply();
43          mDialog.dismiss();
44          initFragment();
45      }
46      @Override
47      public void onFinishHelpProcess(String requestId,
48            int rating) {
49          mHelpProcessId = requestId;
50          mDialog.show();
51          mApi.finishHelp(mHelpProcessId, rating).enqueue(new
52              ApiResponse<>(this:: finishHelpSuccess,
53              this:: finishHelpFailure));
54      }
…   ……
55  }
```

Kode 5.9 Kode program *MainActivityPersonal* untuk proses akhiri pencarian bantuan pada client

Kode 5.9 merupakan potongan kode program *Activity* untuk menangani proses mengakhiri pencarian bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1-5 menunjukan *MainActivityPersonal* mengimplementasi *interface* yang berupa *listener* dari *fragment* yang digunakan untuk mendapatkan hasil interaksi dari *fragment* tersebut.
2. Baris 6-14 berfungsi sebagai penentu tampilan yang akan ditampilkan dengan kondisi pada baris 7-8 jika tidak dalam proses apapun maka akan membuat tampilan pencarian bantuan dengan memanggil fungsi *goToPageRequestHelp,* sedangkan jika dalam proses pencarian bantuan

seperti baris 9-10 akan memanggil tampilan *goToPageProcessHelp* yang digunakan untuk menampilkan daftar pemberi bantuan, selain itu jika dalam proses telah memilih respon maka akan memanggil tampilan *goToPageDetailResponseHelp* untuk menampilkan detail pemberi bantuan.

3. Baris 15-27 berfungsi untuk membuat fragment sebagai *PersonalResponseHelpFragment* yang dapat menampilkan tampilan detail pemberi bantuan pemilihan pemberi bantuan dan merubah tampilannya pada baris 26 dengan memanggil fungsi *changePage*. Namun apabila parameter yang diisikan kosong atau *null* maka pada baris 17-21 akan mengambil data dari dalam *sharedPreferences* terlebih dahulu jika tidak kosong akan langsung memberikan nilai dari parameter kepada *mHelpProcessId*.
4. Baris 28-44 berfungsi sebagai respon sukses saat berhasil melakukan koneksi dengan *web service* untuk mengakhhiri pencarian bantuan. Dimana pada baris 30-31 akan menampilkan pesan singkat bahwa proses pencarian bantuan berakhir, dan mereset ulang semua *variable* indicator proses pada baris 32-34 lalu menyimpan hasil reset tersebut kedalam *sharedPreferences* pada barus 35-43 yang kemudian mereset ulang tampilan dengan memanggil *initFragment* pada baris 44.
5. Baris 46-54 adalah hasil dari implementasi dari *interface* milik *PersonalResponseHelpFragment* yang berfungsi sebagai jembatan komunikasi dari *fragment* menuju *activity* induknya untuk memberitahukan bahwa pengguna mengakhiri proses pencarian bantuan dan memberikan nilai *id* dari *request* dan nilai *rating* dalam parameternya. Lalu baris 50-53 akan menjalankan proses koneksi menuju *web service* menggunakan fungsi *finishHelp* dengan parameter data yang dikirim adalah *id request* dan nilai *rating* yang kemudian akan diteruskan bila sukses menuju fungsi *finishHelpSuccess*.

```
1   public class PersonalResponseHelpFragment extends Fragment {
…   ……
2       public View initView(View view, View sheetView) {
…           ……
3           mResponseDetailButtonFinish.setOnClickListener(v ->
4             mRatingDialog.show());
5           mResponseDetailButtonSubmit.setOnClickListener(v -> {
6               int rating = (int) mResponseDetailRating
7                 .getRating();
8               if (rating <= 0) {
9                   Toast.makeText(mContext,
10                    "Berikan rating terlebih dahulu",
```

```
11                      Toast.LENGTH_SHORT)
12                          .show();
13                  return;
14              }
15              mResponseDetailRating.setRating(0);
16              mRatingDialog.dismiss();
17              submit(rating);
18          });
…           ……
19      }
…   ……
20  }
```

Kode 5.10 Kode program *PersonalResponseHelpFragment* untuk proses akhiri pencarian bantuan pada client

Kode 5.10 merupakan potongan kode program *Fragment* untuk menampilkan halaman detail pemberi bantuan yang terdapat tombol untuk mengakhiri proses pencarian bantuan. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Baris 3-19 merupakan potongan program yang dijalankan untuk membuat tampilan, dimana terdapat *mResponseDetailButtonFinish* sebagai tombol mengakhiri bantuan dan *mResponseDetaulSubmit* sebagai tombol mensubmit data nilai saat mengakhiri bantuan.
2. Pada baris 3-4 ditunjukan bahwa saat tombol *mResponseDetailButtonFinish* ditekan maka akan menampilkan dialog untuk memberikan *rating*.
3. Pada baris 5-18 ditunjukan bahwa saat tombol *mResponseDetailButtonSubmi* ditekan maka akan mengambil nilai *rating* yang diberikan, namun apabila nilai tersebut lebih kecil sama dengan 0 maka akan menampilkan pesan bahwa harus memberikan nilai terlebih dahulu dan mengakhirinya seperti yang ditunjukan pada baris 6-14. Jika kondisi pengecekan terpenuhi maka akan memanggil fungsi *submit.*

```
1   public function finish_request_post() {
2       if (!$this->check_access()) {
3           $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
4             'Cant access');
5       }
6       $id_request = $this->input->post('id_request');
7       $rating = $this->input->post('rating');
8       if (empty($id_request) || empty($rating)) {
9           $this->response_error(STATUS_CODE_KEY_EXPIRED,
10            'Data not complete');
```

```
11      }
12      $user = $this->get_user();
13      $user_responses = $this->m_help->
14        get_user_response($id_request, $user->ID);
15      $user_response = $user_responses[0];
16      $this->m_help->update_response($id_request,
17        $user_response->ID, $user->ID, null, $rating);
18      $this->m_help->finish_request($id_request, $user->ID);
19      if (!empty($user_response->DEVICE_ID)) {
20          $this->load->library('fcm');
21          $this->fcm
22               ->set_target($user_response->DEVICE_ID)
23               ->set_code(Fcm::CODE_HELP_REQUEST_FINISH)
24               ->send();
25          $this->fcm->reset();
26      }
27      $this->response(null);
28  }
```

Kode 5.11 Kode program untuk proses akhiri pencarianbantuan pada *web service/server*

Kode 5.11Kode 5.8 merupakan potongan kode program untuk menanganni proses pengakhiran pencarian bantuan pada sisi *server* / *web service*. Penjelasan dari kode tersebut adalah sebagai berikut:

9. Pada baris 1 menjelaskan bahwa fungsi ini dapat diakses dengan nama finish_request dan dengan metode POST.
10. Baris 2-5 menunjukan jika *request* yang dilakukan tidak memiliki akses baik berupa tidak terdapat token atau token tidak valid maka akan memberikan pesan kesalahan.
11. Baris 6-7 berfungsi untuk menangkap masukan yang diberikan dengan metode POST saat memanggil fungsi ini dimana akan dilakukan pengecekan pada baris 8-11 dan akan memberikan pesan kesalahan apabila data tidak memenuhi.
12. Baris 12 menunjukan pengambilan data *user* dari token yang mengakses fungsi ini.
13. Baris 13-15 adalah proses mengambil data *user* merspon pencarian bantuan dan terpilih untuk membantu.
14. Baris 16-17 akan merubah data respon dengan memberikan nilai rating.
15. Baris 18 akan merubah data *request* menjadi berstatus selesai.

16. Baris 19-26 akan mengirimkan pemberitahuan kepada pemberi bantuan mengenai *badge* atau peringkat barunya.
17. Baris 27 akan mengakhiri proses dengan mengirimkan pesan kosong kepada *client*.

5.4.5 Kode program pemberitahuan pencarian bantuan

Berikut ini Kode 5.12 hingga Kode 5.15 adalah potongan kode program yang digunakan untuk menerima pemberitahuan pencarian bantuan dan melihat serta merespon pencarian bantuan tersebut. Dimana proses tersebut ditangani sebuah *Activity* dan 2 *class* yang berfungsi sebagai penangan notifikasi atau pemberitahuan.

```
1  public class NotificationHandlerService extends
2    FirebaseMessagingService {
3      @Override
4      public void onMessageReceived(RemoteMessage remoteMessage)
5      {
…            ……
6          if (remoteMessage.getData().size() > 0) {
7              Map<String, String> payload = remoteMessage
8                  .getData();
9              if (payload.containsKey(Values
10               .NOTIFICATION_DATA_GLOBAL_CODE)) {
11                 int code = Integer.parseInt(payload
12                   .get(Values.NOTIFICATION_DATA_GLOBAL_CODE));
13                 switch (code) {
14                     case Values
15                         .NOTIFICATION_CODE_REQUEST_PEOPLE_HELP:
16                        RequestSearchNotificationHandler
17                          .handle(this, payload);
18                        break;
…                      ……
19                 }
20             }
21         }
…          ……
22     }
23 }
```

Kode 5.12 Kode program *NotificationHandlerService* untuk menangani pemberitahuan pada client

Kode 5.12 merupakan potongan kode program untuk menangani pemberitahuan yang datang kedalam aplikasi. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1-2 menjelaskan bahwa *class* ini merupakan anak dari *FirebaseMessagingService* yang merupakan penyedia layanan notifikasi atau pemberitahuan.
2. Baris 4-22 merupakan fungsi yang dijalankan saat terdapat pemberitahuan yang masuk.
3. Baris 11-21 adalah potongan kode yang dijalankan saat notifikasi memiliki data atau *payload* dan jika punya akan dilakukan pengecekan apakah data tersebut memiliki elemen kode pemberitahuan yang digunakan untuk menentukan tindakan yang perlu dilakukan, dimana pada baris 14-18 menunjukan bahwa jika kode tersebut merupakan kode notifikasi pencarian bantuan maka akan menjalankan fungsi *handle* dari *RequestSearchNotificationHandler*.

```
1  public class RequestSearchNotificationHandler {
2      public static void handle(Context context, Map<String,
3        String> payload) {
4          String id = payload.get(Values
5            .NOTIFICATION_DATA_REQUEST_SEARCH_ID);
6          String name = payload.get(Values
7            .NOTIFICATION_DATA_REQUEST_SEARCH_NAME);
8          String helpType = payload.get(Values
9            .NOTIFICATION_DATA_REQUEST_SEARCH_TYPE);
10         float distance = Float.parseFloat(
11           payload.get(Values
12             .NOTIFICATION_DATA_REQUEST_SEARCH_DISTANCE));
13         saveData(context, id);
14         sendNotification(context, id, name, helpType,
15             distance);
16         autoCancelNotification(context);
17     }
18     private static void sendNotification(Context context,
19       String responseId, String name, String helpType,
20       float distance) {
21
```

```
        PendingIntent defaultPendingIntent =
NotificationBuilder
          .buildPendingIntent(context, ResponseActivity.class,
          NotificationBuilder.REQUEST_CODE_DEFAULT);
        Intent rejectIntent = new Intent(context,
          ResponseActionActivity.class);
        rejectIntent.putExtra(ResponseActionActivity
          .ACTION_REJECT, true);
        rejectIntent.putExtra(ResponseActionActivity
          .PARAMS_RESPONSE_ID, responseId);
        rejectIntent.setAction(ResponseActionActivity
          .ACTION_REJECT);
        String helpTypeName = context.getString(HelpType
          .getNameFromHelpType(helpType));
        String vehicleTypeName = context.getString(HelpType
          .getVehicleNameFromHelpType(helpType));
        String distanceText;
        if (distance < 1) {
            distance = distance * 1000;
            if (distance < 500) {
                distanceText = "dekat";
            } else {
                distanceText = distance + " meter";
            }
        } else {
            distanceText = distance + " kilometer";
        }
        PendingIntent rejectPendingIntent = NotificationBuilder
          .buildPendingIntent(context, rejectIntent,
          NotificationBuilder.REQUEST_CODE_DEFAULT);
        NotificationBuilder.sendNotification(context, Values
          .NOTIFICATION_ID_REQUEST_SEARCH,
          Values.NOTIFICATION_TAG_REQUEST_SEARCH,
          context.getString(R.string.appName),
          context.getString(
            R.string.notification_message_need_help, name,
            vehicleTypeName, helpTypeName, distanceText),
            defaultPendingIntent, rejectPendingIntent);
    }
```

```
60      private static void autoCancelNotification(Context context)
61      {
62          FirebaseJobDispatcher dispatcher = new
63              FirebaseJobDispatcher(
64                new GooglePlayDriver(context));
65          Bundle jobBundle = new Bundle();
66          Job job = dispatcher
67          .newJobBuilder()
68          .setService(ClearRequestSearchNotificationService.class)
69          .setTag(Values.NOTIFICATION_TAG_REQUEST_SEARCH)
70          .setConstraints(Constraint.ON_ANY_NETWORK)
71          .setLifetime(Lifetime.FOREVER)
72          .setRecurring(false)
73          .setReplaceCurrent(true)
74          .setTrigger(Trigger.executionWindow(60, 90))
75          .setRetryStrategy(RetryStrategy.DEFAULT_LINEAR)
76          .setExtras(jobBundle)
77          .build();
78          dispatcher.mustSchedule(job);
79      }
…       ……
80  }
```

Kode 5.13 Kode program *RequestSearchNotificationHandler* untuk menangani pemberitahuan mengenai pencarian bantuan pada client

Kode 5.13 merupakan potongan kode program yang menangani pemberitahuan khusus pencarian bantuan. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Baris 2-17 adalah fungsi *static* yang bersifat *public* dimana dapat dipanggil tanpa melakukan inisialisasi terlebiih dahulu, dimana pada baris 4-12 akan melakaukan pengambilan data dari notifikasi yang kemudian akan disimpan pada baris 13 dan dibuatkan notifikasi pada baris 14 dan dibuatkan pembatal otomatis pada baris 16.
2. Baris 18-59 merupakan fungsi untuk membuat notifikasi dengan memberika aksi bila ditekan akan membuka *class ResponseActivity* dan jika dihapus akan menjalankan fungsi *reject* pada.
3. Baris 60-79 merupakan fungsi untuk membuat fitur pembatalan otomatis yang menggunakan fitur dari *FirebaseJobDispatcher* dimana proses akan dijalankan di *background*.

```
1   public class ResponseActivity extends AppCompatActivity {
…       ……
2       private void rejectHelp() {
3           Intent intent = new Intent(this,
4             ResponseActionActivity.class);
5           intent.setAction(ResponseActionActivity.ACTION_REJECT);
6           intent.putExtra(ResponseActionActivity
7             .PARAMS_RESPONSE_ID, responseId);
8           startActivityForResult(intent, 100);
9       }
10      private void acceptHelp() {
11          Intent intent = new Intent(this,
12            ResponseActionActivity.class);
13          intent.setAction(ResponseActionActivity.ACTION_ACCEPT);
14          intent.putExtra(ResponseActionActivity
15            .PARAMS_RESPONSE_ID, responseId);
16          startActivityForResult(intent, 100);
17      }
…       ……
18  }
```

Kode 5.14 Kode program *ResponseActivity* untuk menangani detail pencarian bantuan dan merespon pencarian bantuan pada client

Kode 5.14 merupakan potongan kode program untuk menampilkan detail pencarian bantuan dan pemberian respon dimana respon menerima akan memanggil fungsi pada baris 2-19 dan respon tolan akan memanggil fungsi baris 10-17.

```
1   public function help_response_reject_post() {
2       if (!$this->check_access()) {
3           $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
4             'Cant access');
5       }
6       $id_response = $this->input->post('id_response');
7       if (empty($id_response)) {
8           $this->response_error(STATUS_CODE_KEY_EXPIRED,
9             'Data not complete');
10      }
11      $user = $this->get_user();
```

```
12      $responses = $this->m_help->
13        get_request_by_response($id_response, $user->ID);
14      $response = null;
15      if (!empty($responses)) {
16          $response = $responses[0];
17          $response->__cast();
18      }
19      $id_request = $response->ID;
20      if (empty($id_request)) {
21          $this->response_error(STATUS_CODE_KEY_EXPIRED,
22            'Data not complete');
23      }
24      $helps = $this->m_help->get_request($id_request);
25      if (!empty($helps)) {
26          $help = $helps[0];
27          $this->m_help->update_response($help->ID, $user->ID,
28            null, Help_response_data::$RESPONSE_REJECT);
29          if ($user->TYPE == User_data::$TYPE_GARAGE) {
30            $this->load->library('fcm');
31            $response = $this->m_help
32              ->get_response($id_request, $user->ID);
33            $response = $response[0];
34            $this->fcm->set_target($help->ID_USER)
35                 ->set_code(Fcm::CODE_HELP_RESPONSE_REJECTED)
36                 ->set_payload('id', $response->ID)
37                 ->send();
38             $this->fcm->reset();
39          }
40      }
41      $this->response(null);
42  }
43  public function help_response_accept_post() {
44      if (!$this->check_access()) {
45          $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
46            'Cant access');
47      }
48      $id_response = $this->input->post('id_response');
49      if (empty($id_response)) {
```

```
50          $this->response_error(STATUS_CODE_KEY_EXPIRED,
51            'Data not complete');
52      }
53      $user = $this->get_user();
54      $responses = $this->m_help->
55        get_request_by_response($id_response, $user->ID);
56      $response = null;
57      if (!empty($responses)) {
58          $response = $responses[0];
59          $response->__cast();
60      }
61      $id_request = $response->ID;
62      $distance = 0;
63      if (empty($id_request)) {
64          $this->response_error(STATUS_CODE_KEY_EXPIRED,
65            'Data not complete');
66      }
67      $helps = $this->m_help->get_request($id_request);
68      if (!empty($helps)) {
69          $help = $helps[0];
70          $this->m_help->update_response($help->ID, $user->ID,
71            null, Help_response_data::$RESPONSE_ACCEPT);
72          $name = $user->FULL_NAME;
73          $this->load->model('m_location');
74          if ($user->TYPE == User_data::$TYPE_GARAGE) {
75              $this->load->model('m_garage');
76              $garage = $this->m_garage->get($user->ID);
77              $garage = $garage[0];
78              $name = $garage->NAME;
79              $distance = $this->m_location->
80                distance_from_current_location_garage($user->ID,
81                $help->LATITUDE,$help->LONGITUDE);
82          } else {
83              $distance = $this->m_location->
84                distance_from_current_location_personal(
85                $user->ID, $help->LATITUDE, $help->LONGITUDE);
86          }
87          $this->load->library('fcm');
```

```
88          $this->fcm->set_target($help->DEVICE_ID)
89               ->set_code(Fcm::CODE_HELP_RESPONSE_ACCEPTED)
90               ->set_payloads([
91                    'id' => $id_response,
92                    'name' => $name,
93                    'badge' => $this->m_help->badge($user->ID),
94                    'distance' => (float)$distance,
95                    'user_type' => $user->TYPE,
96                    'accept' => true
97               ])->send();
98          $this->fcm->reset();
99          $this->response(null);
100     }
101 }
```

Kode 5.15 Kode program untuk proses respon pencarian bantuan pada *web service/server*

Kode 5.15 merupakan potongan kode program pada *server* yang akan melakukan proses respon dari pencarian bantuan, dimana respon tolak / *reject* dan respon menerima / *accept* memiliki tahapan yang kurang lebih sama dan hanya dibedakan dengan status yang dirubah dan jika tolak tidak akan memberikan pemberitahuan kepada pencari bantuan.

5.4.6 Kode program lihat statistik pencarian bantuan

Berikut ini Kode 5.16 dan Kode 5.17 adalah potongan kode program yang digunakan untuk memproses dan menampilkan data statistik pencarian bantuan. Dimana pada sisi *client* proses tersebut ditangani oleh sebuah *Activity*.

```
1   public class MainActivityGarage extends AppCompatActivity {
…   ……
2       public View initView() {
…           ……
3           mInputStatisticStartDate
4             .setOnClickListener(this::openCalendar);
5           mInputStatisticEndDate
6             .setOnClickListener(this::openCalendar);
7           mStatisticButtonSubmit
8             .setOnClickListener(view -> getStatistic());
…           ……
9       }
```

```
10      private void getStatistic() {
11          if (mStartDate == null || mEndDate == null) {
12              Snackbar.make(mToolbarMainGarage,
13      "Atur tanggal mulai dan tanggal selesai terlebih dahulu",
14              Snackbar.LENGTH_SHORT).show();
15              return;
16          }
17          mDialog.show();
18          mApi.statisticGarage(mStartDate, mEndDate)
19              .enqueue(new ApiResponse<>(
20      this::loadStatisticSuccess, this::loadStatisticFailed));
21      }
22      private void loadStatisticSuccess(Response
23        <StatisticGarageList> response) {
24          mData = new ArrayList<>(response.body().getData());
25          mDialog.dismiss();
26          mStatisticLayoutChart.setVisibility(View.VISIBLE);
27          createRequestHelpChart();
28          createMotorCycleChart();
29          createCarChart();
30      }
…   ……
31  }
```

Kode 5.16 Kode program *MainActivityGarage* untuk memproses dan menampilkan data statistik pencarian bantuan pada client

Kode 5.16 merupakan potongan kode program untuk menampilkan data statistik pencarian bantuan yang terjadi di sekitar dari bengkel. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Baris 2-9 berfungsi untuk menyiapkan tampilan pada *activity* dimana pada baris 3-6 terdapat *mInputStatisticStartDate* dan *mInputStatisticEndDate* yang merupakan masukan tanggal pencarian statistic yang jika ditekan akan membuka kalender untuk memilih tanggal. Sedangkan pada baris 7-8 ada tombol untuk melakukan proses yang akan memanggil fungsi *getStatstic* jika ditekan.
2. Baris 10-21 berfungsi sebagai proses pencarian bantuan dimana jika tanggal mulai dan tanggal akhir *range* pencarian bantuan belum diatur maka akan muncul pesan kesalahan seperti pada baris 11-16. Jika tidak maka pada baris 18-20 akan melakukan proses koneksi kepada *web service* melalui fugnsi *statisticGarage* dengan memberikan parameter tanggal mulai dan tanggal

akhir sebagai masukan datanya yang kemudian jika sukes akan diteruskan menuju fungsi *loadStatisticSuccess*.

3. Baris 22-30 merupakan fungsi yang dipanggil saat berhasil mendapatkan data statistic, dimana data tersebut akan dimasukan kedalam sebuah *List* pada baris 24 dan di buatkan grafik yang sesuai data seperti yang ditunjukan pada baris 27-29.

```
1  public function statistic_post() {
2      if (!$this->check_access()) {
3          $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
4            'Cant access');
5      }
6      $date_start = $this->input->post('date_start');
7      $date_end = $this->input->post('date_end');
8      $date_start_time = strtotime($date_start);
9      $date_end_time = strtotime($date_end);
10     $total_day = ceil(abs($date_end_time - $date_start_time)
11       / 86400);
12     $user = $this->get_user();
13     if ($user->TYPE == User_data::$TYPE_PERSONAL) {
14         $this->response_error(STATUS_CODE_NOT_AUTHORIZED,
15           'Cant access');
16     }
17     $this->load->model('m_garage');
18     $garages = $this->m_garage->get($user->ID);
19     $garage = $garages[0];
20     $garage->__cast();
21     $this->load->model('m_location');
22     $result = $this->m_location->get_nearby_request(
23       $garage->LATITUDE, $garage->LONGITUDE, $date_start,
24        $date_end);
25     for ($i = 0; $i < count($result); $i++) {
26         $result[$i]->__cast_date();
27     }
28     $data = [];
29     for ($i = 0; $i <= $total_day; $i++) {
30        $data[$i] = [
31        'motor_1' => 0,
32        'motor_1_id' => Config_data::$TYPE_MOTORCYCLE_FLAT_TIRE,
```

```
33          'motor_2' => 0,
34          'motor_2_id' => Config_data::$TYPE_MOTORCYCLE_NO_FUEL,
35          'motor_3' => 0,
36          'motor_3_id' => Config_data::$TYPE_MOTORCYCLE_BROKEN,
37          'car_1' => 0,
38          'car_1_id' => Config_data::$TYPE_CAR_FLAT_TIRE,
39          'car_2' => 0,
40          'car_2_id' => Config_data::$TYPE_CAR_NO_FUEL,
41          'car_3' => 0,
42          'car_3_id' => Config_data::$TYPE_CAR_BROKEN,
43          'total' => 0  ];
44      }
45      for ($i = 0; $i < count($data); $i++) {
46          $date_check_time = strtotime('+' . $i . ' day',
47            $date_start_time);
48          $date_check = date('Y-m-d', $date_check_time);
49          $data[$i]['date'] = date('Y-m-d H:i:s',
50            $date_check_time);
51          foreach ($result as $item) {
52              $item->__cast_date();
53              if ($date_check == $item->DATE_CREATE) {
54                  $data[$i]['total']++;
55                  switch ($item->ID_HELP_TYPE) {
56                    case Config_data::$TYPE_MOTORCYCLE_FLAT_TIRE:
57                        $data[$i]['motor_1']++; break;
58                    case Config_data::$TYPE_MOTORCYCLE_NO_FUEL:
59                        $data[$i]['motor_2']++; break;
60                    case Config_data::$TYPE_MOTORCYCLE_BROKEN:
61                        $data[$i]['motor_3']++; break;
62                    case Config_data::$TYPE_CAR_FLAT_TIRE:
63                       $data[$i]['car_1']++; break;
64                    case Config_data::$TYPE_CAR_NO_FUEL:
65                        $data[$i]['car_2']++; break;
66                    case Config_data::$TYPE_CAR_BROKEN:
67                        $data[$i]['car_3']++; break;
68                }
69              }
70          }
```

```
71      }
72      $this->response($data);
73  }
```

Kode 5.17 Kode program untuk mendapatkan data statistik pencarian bantuan pada web service/*server*

Kode 5.17 merupakan potongan kode program pada *server* yang berfungsi mendapatkan data statistik pencarian bantuan. Penjelasan dari kode tersebut adalah sebagai berikut:

1. Pada baris 1 menjelaskan bahwa fungsi ini dapat diakses dengan nama statistic dan dengan metode POST.
2. Baris 2-5 menunjukan jika *request* yang dilakukan tidak memiliki akses baik berupa tidak terdapat token atau token tidak valid maka akan memberikan pesan kesalahan.
3. Baris 6-7 berfungsi untuk menangkap masukan yang diberikan dengan metode POST saat memanggil fungsi ini dimana akan digunakan untuk menjadi batas waktu pencarian bantuan yang terjadi.
4. Baris 13-16 menunjukan bahwa fungsi ini hanya dapat digunakan oleh pengguna dengan tipe bengkel.
5. Baris 18-20 menunjukan pengambilan data bengkel milik pengguna.
6. Baris 22-27 merupakan proses pengambilan data pencarian bantuan terdekat dari bengkel.
7. Baris 28-44 merupakan proses pembuatan data sementara untuk menampung data statistic nantinya.
8. Baris 45-71 merupakan proses pengelompokan data statistik sesuai dengan tanggal dari proses pencarian bantuan.
9. Baris 72 menunjukan pengembalian data kepada *client* denga nisi data statistik.

5.4.7 Kode program validasi pengguna

Berikut ini Kode 5.18 dan Kode 5.19 adalah potongan kode program yang digunakan untuk proses validasi pendaftaran oleh admin.

```
1   class Register extends TDB_Controller{
…   ……
2   public function validation() {
…       ……
3       $data['registered_garage'] = $this->m_user->get(
4         null, null, User_data::$STATUS_NOT_ACTIVE, null,
```

```
5         User_data::$TYPE_GARAGE);
6       $data['registered_personal'] = $this->m_user->get(
7         null, null, User_data::$STATUS_NOT_ACTIVE,
8         null, User_data::$TYPE_PERSONAL);
9       $this->load->view('admin/base/header', $data);
10      $this->load->view('admin/manage/register/validation_list',
11        $data);
12      $this->load->view('admin/base/footer', $data);
13  }
14  public function validate($id) {
15      $user = $this->get_user();
16      $result = $this->m_user->update_status(
17        $id, User_data::$STATUS_ACTIVE, $user->ID);
18      if ($result) {
19         $registered = $this->m_user->get(null, null, null, $id);
20          if (!empty($registered)) {
21              $registered = $registered[0];
22              if (!empty($registered->DEVICE_ID)) {
23                  $this->load->library('fcm');
24                  $title = 'Status Verifikasi ' . $this->config
25                    ->item('app_name');
26                  $message =
27          'Pendaftaran anda telah di verifikasi, silahkan
28          menggunakan aplikasi dan membantu sesama';
29                  $this->fcm->set_target($registered->DEVICE_ID)
30                       ->set_key($this->config
31                          ->item('fcm_key', 'sensitive'))
32                       ->set_title($title)
33                       ->set_message($message)
34                       ->set_code(Fcm::CODE_REGISTER_COMPLETE)
35                       ->send();
36                  $message = $message . ' pada ' .
37                       date('d M Y H:i');
38                  $this->send_email($registered->EMAIL,
39                       $title, $message);
40              }
41          }
42      }
```

```
43      redirect('admin/manage/register');
44  }
…   ……
45  }
```

Kode 5.18 Kode program penampilan data pendaftar dan pemrosesan validasi pengguna oleh admin

Kode 5.18 merupakan potonga kode program *controller* pada web admin yang berfungsi menangani dan menampilkan data pendaftar. Penjelasan kode program tersbut dijelaskan sebagai berikut:

1. Baris 1 menunjukan nama dari *controller* yang bernama *register*.
2. Baris 2-13 merupakan fungsi dari *controller* yang akan menampilkan halaman validasi, dimana akan berisi daftar pengguna yang mendaftar pada aplikasi.
3. Baris 3 akan mengambil data pengguna dengan jenis bengkel yang mendaftar.
4. Baris 4 akan mengambil data pengguna dengan jenis personal yang mendaftar.
5. Baris 9-12 akan menampilkan data tersebut dalam *view*.
6. Baris 14-44 merupakan fungsi untuk proses validasi dengan parameter *id* dari pengguna yang akan divalidasi.
7. Baris 16 akan merubah status pengguna dengan *id* yang ada pada paremeter menjadi aktif.
8. Baris 19-41 akan dijalankan setelah proses perubahan status berhasil dan akan mengirim pemberitahuan menuju aplikasi beserta email mengenai status pendaftarannya.
9. Baris 43 akan mengembalikan menuju halaman mengelola validasi pendaftaran.

```
1   class User extends TDB_Controller {
…   ……
2   public function detail($id, $validation = null) {
3       $from_validation = (strtolower($validation)
4         == 'validation');
5       $registered = $this->m_user->get(
6           null, null, ($from_validation) ?
7     [User_data::$STATUS_NOT_ACTIVE,User_data::$STATUS_REJECTED]:
8     [User_data::$STATUS_ACTIVE, User_data::$STATUS_BANNED], $id);
9       if (!empty($registered)) {
```

```
10          $registered = $registered[0];
11          $registered->__cast();
12          $data = $this->basic_data();
…           ……
13          $data['from_validation'] = $from_validation;
14          if ($registered->ID != $registered->ID_UPDATE) {
15              $last_update_by = $this->m_user
16                ->get(null, null, null, $registered->ID_UPDATE);
17              if (!empty($last_update_by)) {
18                  $data['last_update_by'] = $last_update_by[0]
19                    ->FULL_NAME;
20              }
21          }
22          if ($registered->TYPE == User_data::$TYPE_GARAGE) {
23              $this->load->model('m_garage');
24              $registered_garages = $this->m_garage->
25                get($registered->ID);
26              $registered_garage = $registered_garages[0];
27              $registered_garage->__cast();
28              $data['registered_garage'] = $registered_garage;
29          }
30          $data['registered'] = $registered;
31          $this->load->view('admin/base/header', $data);
32          $this->load->view('admin/manage/user/detail', $data);
33          $this->load->view('admin/base/footer', $data);
34      } else {
35          $this->session->set_flashdata('error', 'User tidak
36  ditemukan'. ($from_validation ? ' atau telah berubah status' :
37  ''));
38          if (!$from_validation) {
39              redirect('admin/manage/register/validation');
40          } else {
41              redirect('admin/manage/user');
42          }
43      }
44  }
…   ……
45  }
```

Kode 5.19 Kode program penampilan detail pendaftar untuk admin

Kode 5.18 merupakan potonga kode program *controller* pada web admin yang berfungsi detail dari pendaftar. Penjelasan kode program tersbut dijelaskan sebagai berikut:

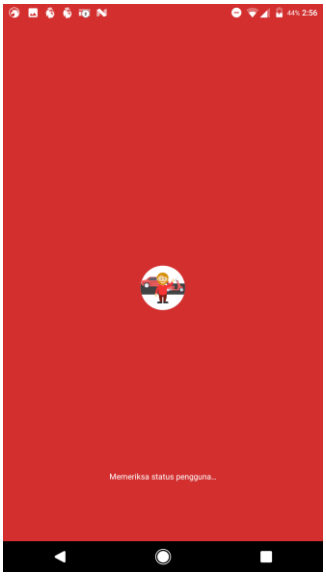
1. Baris 1 menunjukan *controller* bernama user
2. Baris 2-43 berfungsi menunjukan detail dari pengguna dengan parameter *id* dari user dan parameter opsional yaitu *validation* yang berguna untuk menunjukan halaman ini diakses dari halaman validasi atau tidak.
3. Baris 3 akan mengecek apakah halaman ini diakses dari halaman validasi pendaftaran atau tidak.
4. Baris 5-8 akan mengambil data pengguna yang jika diakses dari halaman validasi maka akan mengambil data pengguna dengan *id* dari parameter dan status tidak aktif atau ditolak, jika tidak maka akan mengambil data pengguna dengan *id* dari parameter dan status aktif atau *banned*.
5. Baris 22-29 akan dijalankan jika tipe pengguna adalah bengkel, dan akan mengabil data detail dari bengkel.
6. Baris 30-33 akan memanggil *view* untuk menampilkan data yang diambil sebelumnya dimana didalam *view* tersebut terdapat tombol validasi yang akan mengarahkan kehalaman proses validasi.

5.5 Implementasi antarmuka

Pada tahap ini dijelaskan mengenai berbagai hasil implementasi antarmuka pengguna yang telah diimplementasikan berdasarkan dari perancangan navigasi dan antarmuka yang telah dilakukan.

5.5.1 Halaman *Splash Screen*

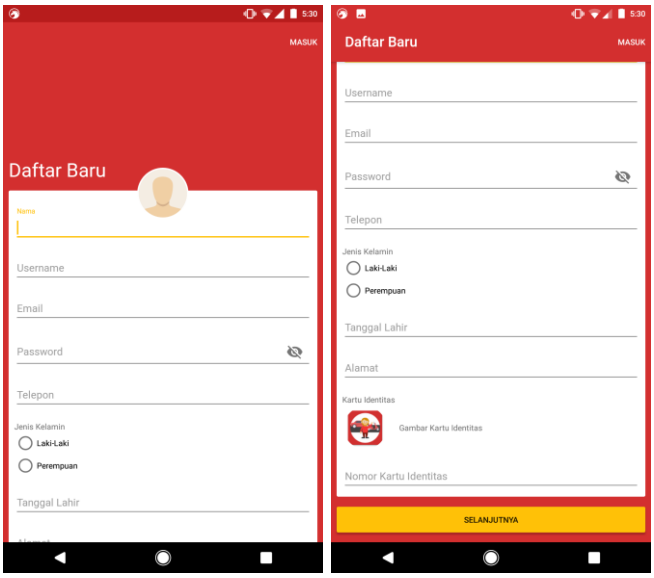
Pada halam ini menunjukan halaman berlatar belakan warna merah yang merupakan warna utama dari aplikasi disertai logo aplikasi ditengah dan pesan proses yang sedang berjalan dibawah dari logo. Implementasi antarmuka ini ditunjukan pada Gambar 5.9.



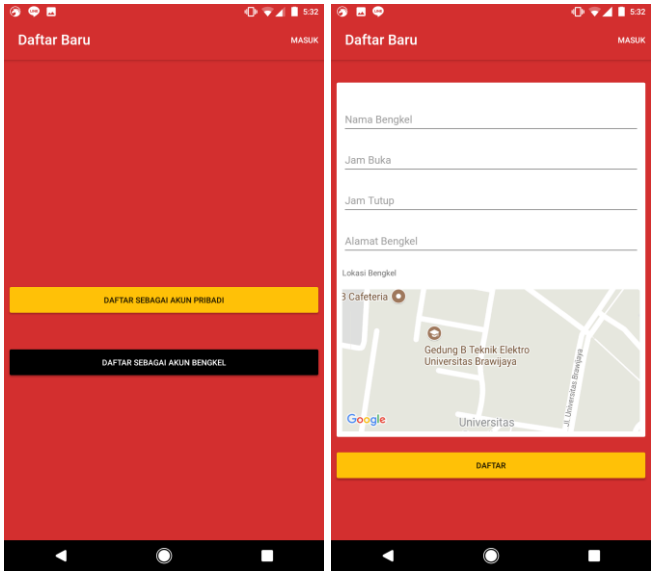
Gambar 5.9 Implementasi halaman *Splash Screen*

5.5.2 Halaman pendaftaran

Pada halam ini menunjukan halaman dengan formulir pendaftaran besera tombol navigasi pendaftaran, terdapat juga menu untuk melakukan *login* / masuk jika memiliki akun. Implementasi antarmuka ini ditunjukan pada Gambar 5.10 dan Gambar 5.11.



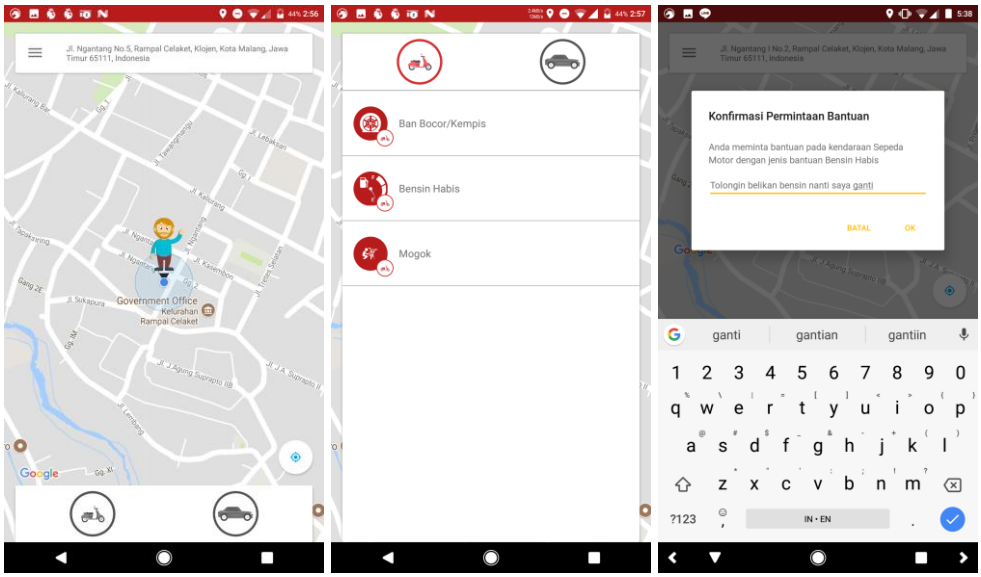
Gambar 5.10 Implementasi halaman pendaftaran (formulir dasar pengguna)



Gambar 5.11 Implementasi halaman pendaftaran (pemilihan jenis pengguna dan formulir bengkel)

5.5.3 Halaman pencarian bantuan

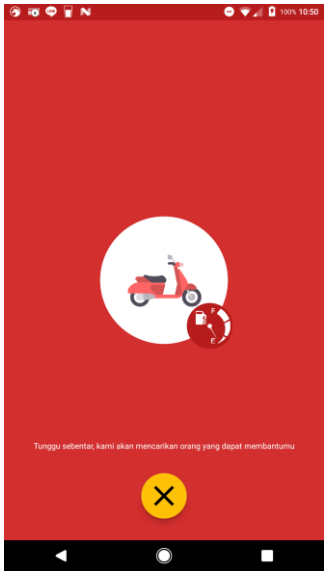
Pada halam ini menunjukan peta dengan marker gambar orang yang menunjukan lokasi sekarang, dimana marker itu dapat diubah lokasinya dengan menggerakan peta. Terdapat juga penjelasan nama lokasi keberadaan diatas beserta menu navigasi disebelahnya. DIbagian bawah terdapat tombol berupa lingkaran dengan gambar jenis kendaraan yang jika dipilih akan membuka jenis bantuan berdasarkan kendaraan yang dipilih dan setelah itu akan muncul kotak dialog konfirmasi beserta masukan pesan. Implementasi antarmuka ini ditunjukan pada Gambar 5.12.



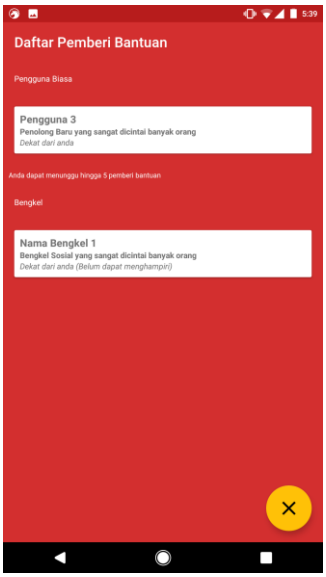
Gambar 5.12 Implementasi halaman pencarian bantuan

5.5.4 Halaman pemilihan pemberi bantuan

Pada halam ini menunjukan gambar jenis permasalahan dan kendaraan yang dipilih saat menunggu respon dari pengguna lain dengan pesan dibawahnya dan juga tombol batal ditengah seperti yang ditunjukan pada Gambar 5.13. Saat pengguna lain telah merespon akan muncul daftar yang dipisahkan sesuai jenis penggunanya yaitu personal diatas dan bengkel dibawah disertai nama, peringkat dan jarak dari lokasi sekarang. Selain itu tombol untuk batal berpindah kesebelah kanan seperti yang ditunjukan pada Gambar 5.14.



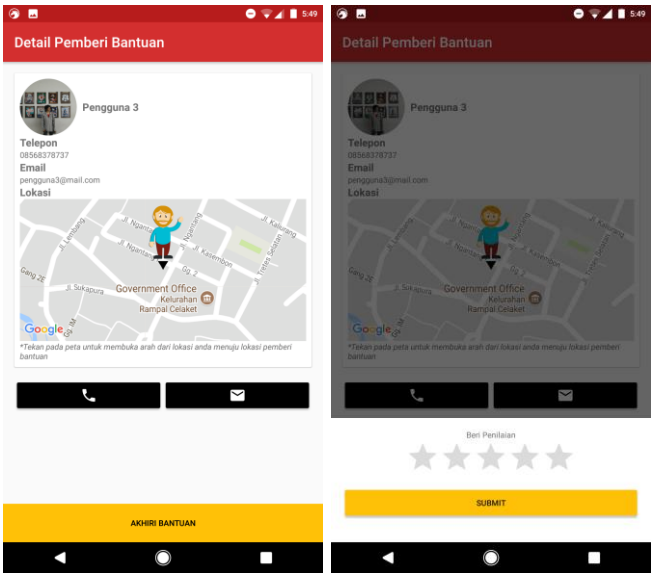
Gambar 5.13 Implementasi halaman pemilihan pemberi bantuan (proses menunggu respon)



Gambar 5.14 Implementasi halaman pemberi bantuan (daftar pemberi bantuan)

5.5.5 Halaman akhiri pencarian bantuan

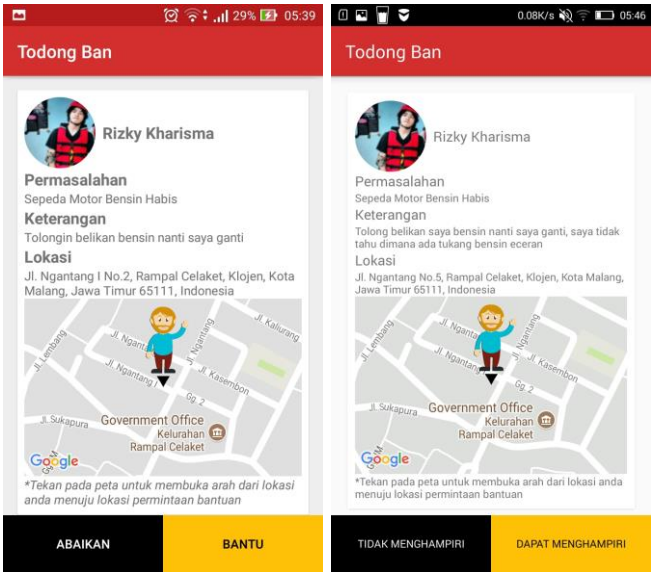
Pada halam ini menunjukan halaman dengan detail pemberi bantuan dan lokasi dari sang pemberi bantuan dalam bentuk peta yang jika ditekan akan membuka peta arah menuju tempat pemberi bantuan. Selain itu terdapat tombol untuk melakukan panggilan telepon kepada pemberi bantuan dan juga tombol mengirim email ke pemberi bantuan. Dibagian bawah terdapat tombol untuk mengakhiri yang jika ditekan akan muncul formulir rating dnegan bintang yang bisa dipilih dan tombol *submit* untuk memproses akhiri bantuan. Implementasi antarmuka ini ditunjukan pada Gambar 5.15.



Gambar 5.15 Implementasi halaman akhiri pencarian bantuan

5.5.6 Halaman detail pemberitahuan pencarian bantuan

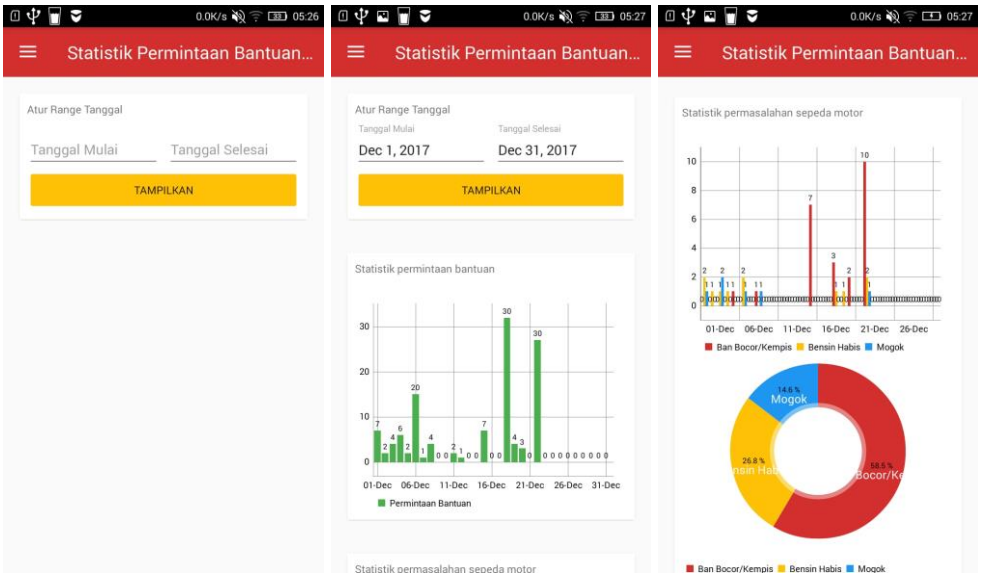
Pada halam ini menunjukan halaman dengan detail pencari bantuan beserta jenis bantuan yang diminta dan lokasi dari sang pencari bantuan dalam bentuk peta yang jika ditekan akan membuka peta arah menuju tempat pencari bantuan. Dibagian bawah terdapat tombol merespon berupa abaikan dan bantu untuk pengguna personal dan tidak menghampiri dan dapat menghampiri untuk pengguna bengkel. Implementasi antarmuka ini ditunjukan pada Gambar 5.16.



Gambar 5.16 Implementasi halaman detail pemberitahuan pencarian bantuan personal (kiri) & bengkel (kanan)

5.5.7 Halaman lihat statistik pencarian bantuan

Pada halam ini menunjukan halaman dengan formulir berisi masukan tanggak mulai dan tanggal selesai dan tombol *submit* pada awalnya, yang jika ditekan akan menampilkan data statistik yang disajikan dalam bentuk grafik dibawah formulir sebelumnya. Implementasi antarmuka ini ditunjukan pada Gambar 5.17.



Gambar 5.17 Implementasi halaman statistik pencarian bantuan