BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Pelaksanaan kerja magang di divisi IT dilakukan dengan peran sebagai *Software Developer Intern*. Kegiatan magang ini diawasi oleh Bapak Adi Purnomo, yang menjabat sebagai Kepala Divisi IT, dan dibimbing oleh Bapak Chaerul Sazali, staf di bagian Development. Selama masa magang, tanggung jawab utama adalah merancang dan membangun aplikasi *Sales Force Automation* (SFA). Dalam proses perancangan dan pengembangan aplikasi SFA, koordinasi dilakukan melalui diskusi langsung dengan Bapak Chaerul Sazali.

3.2 Tugas yang Dilakukan

Magang yang dilakukan di PT. Dirgaputra Ekapratama melibatkan berbagai tugas penting yang memberikan pengalaman dalam dunia kerja. Tugas utama yang diberikan adalah perancangan dan pembangunan aplikasi *Sales Force Automation* (SFA). Selain itu, ada juga tugas tambahan yaitu membantu staf infrastruktur dalam beberapa tugas harian. Pengalaman ini tidak hanya meningkatkan keterampilan teknis, tetapi juga kemampuan bekerja dalam tim dan memahami kebutuhan operasional perusahaan secara lebih mendalam. Berikut uraian tugas yang dilakukan selama kerja magang:

1. Membuat Aplikasi *Sales Force Automation*
    - Membuat desain antarmuka pengguna (UI/UX).
    - Menggunaan bahasa pemograman java sebagai dasar pembuatan aplikasi SFA
    - Melakukan *display* data menggunakan *volley* dari *webservices* yang disediakan mentor
2. Membantu staf infrastuktur melakukan instalasi
    - Instalasi aplikasi yang digunakan untuk melakukan pekerjaan di kantor
    - Melakukan *install* aplikasi *antivirus* pada laptop dan komputer yang ada di kantor pusat

3.3 Uraian Pelaksanaan Magang

Berikut adalah uraian pelaksanaan kerja magang yang dilakukan di PT. Dirgaputra Ekapratama, seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama magang

Minggu Ke -	Pekerjaan yang dilakukan
1	– *Onboarding* peserta magang pengenalan lingkungan kantor & gudang oleh *staf Human Capital* – *Training business process* PT. Dirgaputra Ekapratama oleh Bapak Adi Purnomo
2	– Training software di PT. Dirgaputra Ekapratama oleh Bapak Sofyan – Penjelasan rancangan aplikasi SFA oleh Bapak Chaerul Sazali. – Penentuan aplikasi dan bahasa pemrograman yang digunakan dalam pembuatan aplikasi
3	Pembuatan tampilan dan kode (.xml & .java) *Splash Screen, Login page, Activity Home*, dan Tampilan awal *Fragment (Home, Transaction, Report,*dan *More).*
4	– Membuat SessionManagement.java yang digunakan sebagai sharedPreferences untuk mendapatkan data *sales* melalui *webservices* dengan metode *volley post.* – Menambahkan *checkLogin* sehingga *user* yang sudah pernah melakukan *login* tidak perlu melakukan *login* lagi pada saat membuka aplikasi dilain waktu.
Lanjut pada halaman berikutnya	

Tabel 3.1: Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
5	Pembuatan tampilan *homepage* dan menghubungkan dengan *webservice* untuk mendapatkan *value* dari target, pencapaian, retur, dan persentase.
6	– Melakukan instalasi *software* kantor pusat pada laptop baru kepala cabang – Instalasi *antivirus* dengan *license* baru untuk seluruh komputer dan laptop kantor pusat besama dengan Bapak Dani.
7	– Menyambungkan *FragmentReport* dengan *website reporting* yang digunakan PT. Dirgaputra Ekapratama menggunakan *webview*. – Menambahkan *cookie manager* untuk menyimpan *user id* dan *password*.
8	– Membuat tampilan *FragmentMore* yang berisikan nama *sales*, id, tombol untuk melihat *profile user*, tombol untuk mengganti *password*, dan tombol untuk melakukan *logout*. – Membuat tampilan *Profile User* untuk meperlihatkan *User profile* dan *Sales profile*.
9	– Membuat tampilan *change password* dan *logout confirmation* dengan menggunakan *alertDialog* – Memperbaiki beberapa *error minor* pada saat change *fragment*

Lanjut pada halaman berikutnya

Tabel 3.1: Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
10	– Membuat tampilan *Fragment transaction* menggunakan *fragment*, – Membuat data *dummy* untuk *RecycleView Order* dan *Process*.
11	– Membuat *MatriksCustomer* untuk melakukan *display* seluruh *customer* yang dimiliki oleh *sales* menggunakan *RecycleView* – Menghubungkan *activity MatriksCustomer* dengan data yang diambil dari *webservice*.
12	Memperbaiki beberapa *error* pada *MatriksCustomer* dan *Recyclerview* untuk menampilkan data *customer*.
13	Menambakan fitur *search matriks customer* pada *header page* berdasarkan nama *customer* dan id *customer*
14	Membuat tampilan *alertdialog* saat *customer* dipilih dan dapat menuju ke *customer detail* dan *order* untuk melakukan pemesanan .
15	Membuat tampilan *customer detail* menggunakan *dropdown* yang menampilkan, *profile customer, profile pajak, dan ship to*.
Lanjut pada halaman berikutnya	

Tabel 3.1: Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
16	Membuat tampilan *outstanding order*, *ordering* dan 3 *fragment* untuk *header, ship to,* dan *detail* untuk melakukan order barang.

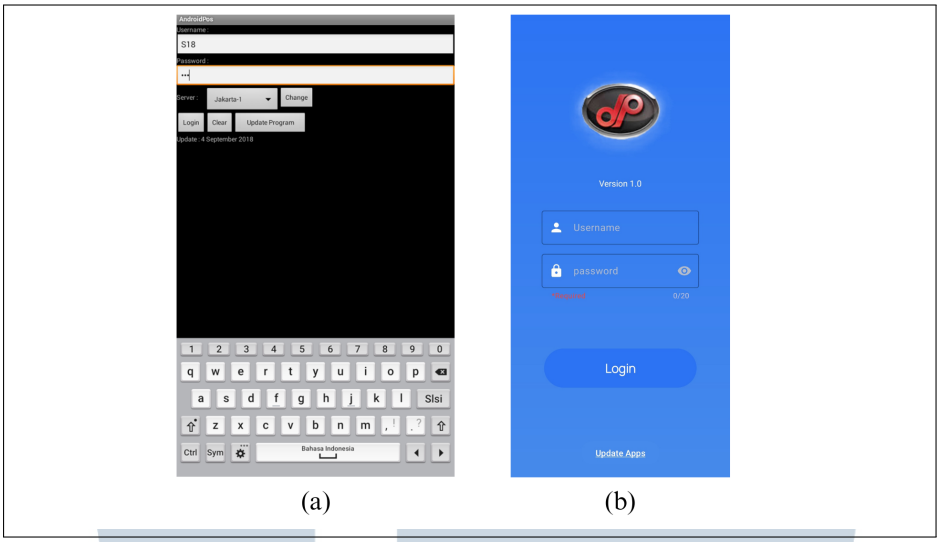
3.3.1 Pembuatan Aplikasi Sales Force Automation

Dalam pembuatan aplikasi SFA bahasa yang akan digunakan adalah bahasa pemrograman *Java* dan akan menggunakan *Android Studio* dalam pembuatannya. Tampilan dan kode yang digunakan dibuat berdasarkan diskusi dengan Bapak Chaerul Sazali. Berikut adalah tampilan dari aplikasi lama dan baru beserta potongan kode yang digunakan pada aplikasi baru:

A. Login Page

Pada Gambar 3.1 terdapat dua bagian yaitu bagian (a) dan (b). Gambar 3.1 bagian (a) adalah tampilan dari *login page* yang lama. Sedangkan bagian (b) adalah tampilan *login page* aplikasi SFA yang baru. Pada gambar (b) terdapat logo perusahaan PT. Dirgaputra Ekapratama saat ingin melakukan *login*, sedangkan pada Gambar A tidak terdapat apapun selain *input username, password* dan tombol lainnya. Selain itu pada bagian bawah dari Gambar B terdapat tombol untuk melakukan *update* aplikasi, sehingga tim IT PT. Dirgaputra Ekapratama tidak perlu melakukan instalasi ulang pada perangkat yang digunakan oleh *sales*.

Saat *user(sales)* menekan tombol *login* setelah mengisi *username* dan *password*, maka aplikasi akan melakukan metode *loginLoad*. Aplikasi akan mengirimkan permintaan *login* ke server menggunakan *library Volley*. Jika respon berhasil, metode ini akan memproses data *sales* yang diterima dan menyimpannya dalam *SessionManagement* seperti pada potongan Kode 3.1. Jika terjadi kesalahan, baik dari server maupun jaringan, akan ditampilkan pesan kesalahan melalui *Toast*. Metode ini juga menampilkan *dialog* (*loading*) saat proses *login* berlangsung dan menutupnya setelah proses selesai.



Gambar 3.1. Tampilan *Login page* aplikasi lama dan baru

```
1  private void loginLoad(final String param_username, final
   String param_password) {
2      String tag_string_req = "req_login";
3      pDialog.setMessage("Process Login...");
4      showDialog();
5      StringRequest strReq = new
   StringRequest(Request.Method.POST,
6      AppConfig.URL_LOGIN, new Response.Listener<String>() {
7          @Override
8          public void onResponse(String response) {
9              Log.d(TAG, "Process Response: " +
   response.toString());
10             hideDialog();
11             try {
12                 JSONObject jObj = new JSONObject(response);
13                 String error = jObj.getString("error");
14                 if (error.equals("true")) {
15                     // Error in login. Get the error message
16                     String message =
   jObj.getString("error_msg");
17                     Toast.makeText(getApplicationContext(),
18                         "Error Message : " + message,
   Toast.LENGTH_LONG).show();
19                 } else {
```

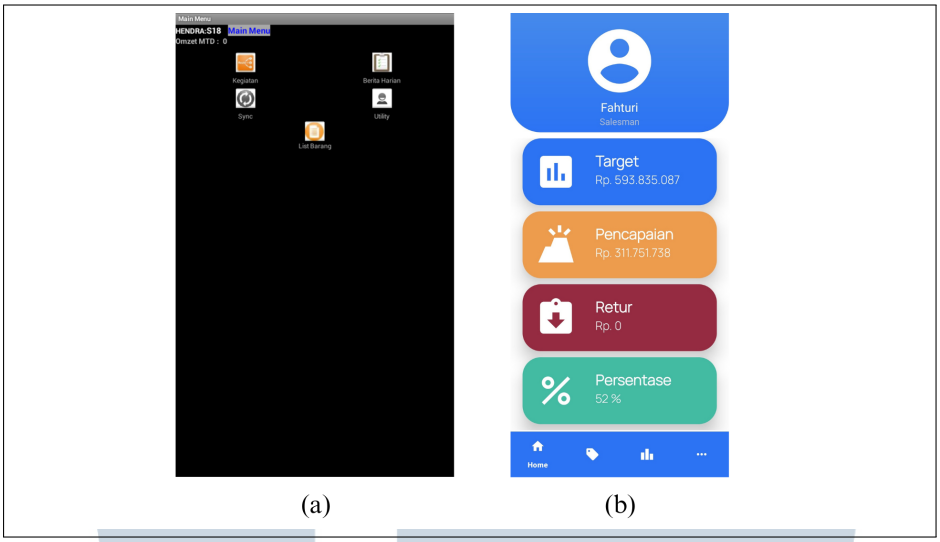
```
20                  String message =
    jObj.getString("error_msg");
21                  String Username =
    jObj.getString("Username");
22                  String Password =
    jObj.getString("Password");
23                  String NamaLengkap =
    jObj.getString("Example Name");
24                  ... // data lainnya
25                  Toast.makeText(getApplicationContext(),
26                      "Success Login : " + message,
    Toast.LENGTH_LONG).show();
27                  sessionManagement.setLogin(true,
    Username, Password, NamaLengkap);
28                  sessionManagement.checkLogin();
29              }
30          } catch (JSONException e) {
31              e.printStackTrace();
32              Toast.makeText(getApplicationContext(),
    "Json error: " + e.getMessage(),
    Toast.LENGTH_LONG).show();Toast.LENGTH_LONG).show();
33          }
34      }
35   }, new Response.ErrorListener() {
36       @Override
37       public void onErrorResponse(VolleyError error) {
38           hideDialog();
39           Toast.makeText(getApplicationContext(),
    "Network Error: " +
40               error.getMessage(),
    Toast.LENGTH_SHORT).show();
41       }
42   }) {
43       @Override
44       protected Map<String, String> getParams() {
45           Map<String, String> params = new
    HashMap<String, String>();
46           params.put("token", AppConfig.AS_TOKEN);
```

```
47              params.put("username", param_username);
48              params.put("password", param_password);
49              return params;
50          }
51      };
52      strReq.setRetryPolicy(new
    DefaultRetryPolicy(AppConfig.TIME_OUT_LOADING,
53          DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
54          DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
55      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);}
```

Kode 3.1: Potongan kode saat melakukan *login*

B. Fragment Home Page

Gambar 3.2 memiliki dua bagian yaitu bagian (a) dan (b). Gambar (a) adalah tampilan *homepage* dari aplikasi lama. Gambar (b) adalah tampilan *homepage* yang baru. Tampilan *homepage* dapat dilihat setelah *user* berhasil melakukan *login*, pada gambar (a) akan ditampilkan tombol kegiatan, berita harian, *sync*, *utility*, dan *list* barang. Sedangkan pada gambar (b) *user* dapat melihat nama, jabatan, target bulanan, pencapaian, retur, dan persentase dari pencapaiannya. Selain itu tampilan *homepage* ini juga menggunakan *fragment* untuk berpindah *activity*. Menampilkan data pada halaman ini menggunakan metode yang mirip seperti saat akan melakukan login yaitu *POST*. Metode *targetLoad* pada potongan Kode 3.2 mengirim permintaan *POST* ke server untuk memuat data target berdasarkan ”kodesales” dan ”kodecabang” yang diberikan. Metode ini menampilkan *loading* selama proses berlangsung. Setelah menerima respons dari server, data target, total pencapaian, retur, dan persentase ditampilkan di UI.



(a) (b)

Gambar 3.2. Tampilan *Home Page* aplikasi lama dan baru

```
private void targetLoad(final String kodesales, final
    String kodecabang) {
     String tag_string_req = "req_target";
     pDialog.setMessage("Loading Target Information...");
     showDialog();
     StringRequest strReq = new
    StringRequest(Request.Method.POST,
     AppConfig.URL_TARGET, new Response.Listener<String>() {
         @Override
         public void onResponse(String response) {
             hideDialog();
             try {
             JSONObject jObj = new JSONObject(response);
             boolean error = jObj.getBoolean("error");
             if (!error) {
                 String target = jObj.getString("Target");
                 String total = jObj.getString("Total");
                 String retur = jObj.getString("Retur");
                 String percentage =
    jObj.getString("Persentation");
                 targetValue.setText(target);
                 pencapaianValue.setText(total);
                 persentaseValue.setText(percentage);
                 returValue.setText(retur);
```

```
22              } else {
23                  String message =
    jObj.getString("error_msg");
24                  Toast.makeText(getContext(), "Error: " +
    message, Toast.LENGTH_SHORT).show();
25              }
26          } catch (JSONException e) {
27          e.printStackTrace();
28          Toast.makeText(getContext(), "JSON Error: " +
    e.getMessage(), Toast.LENGTH_SHORT).show();
29          }
30       }
31       }, new Response.ErrorListener() {
32          @Override
33          public void onErrorResponse(VolleyError error) {
34              hideDialog();
35              Toast.makeText(getContext(), "Network Error: "
    + error.getMessage(), Toast.LENGTH_SHORT).show();
36          }
37          }) {
38          @Override
39          protected Map<String, String> getParams() {
40              Map<String, String> params = new HashMap<>();
41              params.put("kodesales", kodesales);
42              params.put("kodecabang", kodecabang);
43              return params;
44          }
45          };
46          strReq.setRetryPolicy(new DefaultRetryPolicy(
47          AppConfig.TIME_OUT_LOADING,
48          DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
49          DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
50
    AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
51  }
52
```

Kode 3.2: Potongan kode *targetLoad*

C. Fragment Transaction

Selanjutnya adalah *Fragment Transaction* , *fragment* ini adalah inti utama dalam aplikasi ini. Pada Gambar 3.3 terdapat dua bagian yaitu bagian (a) dan bagian (b). Bagian (a) pada Gambar 3.3 adalah tampilan dari aplikasi yang lama, sedangkan gambar (b) adalah tampilan baru yang sudah dibuat. Pada gambar (b) *user* dapat melihat *outstanding order* yang sedang berlangsung pada pelanggannya. *RecyclerView* digunakan untuk menampilkan *outstanding order*, hal ini dipilih karena *RecyclerView* lebih efektif dan fleksibel dalam penggunaan dibandingkan *ListView* [**?**].

(a) (b)

Gambar 3.3. Tampilan *Transaction Page*

Selain *outstanding order*, pada bagian atas Gambar 3.3 terdapat dua tombol yaitu daftar kunjungan dan *matriks customer*. Untuk saat ini daftar kunjungan belum masuk ke dalam tujuan kerja magang. Sehingga peserta magang hanya perlu membuat *matriks customer* pada aplikasi SFA.

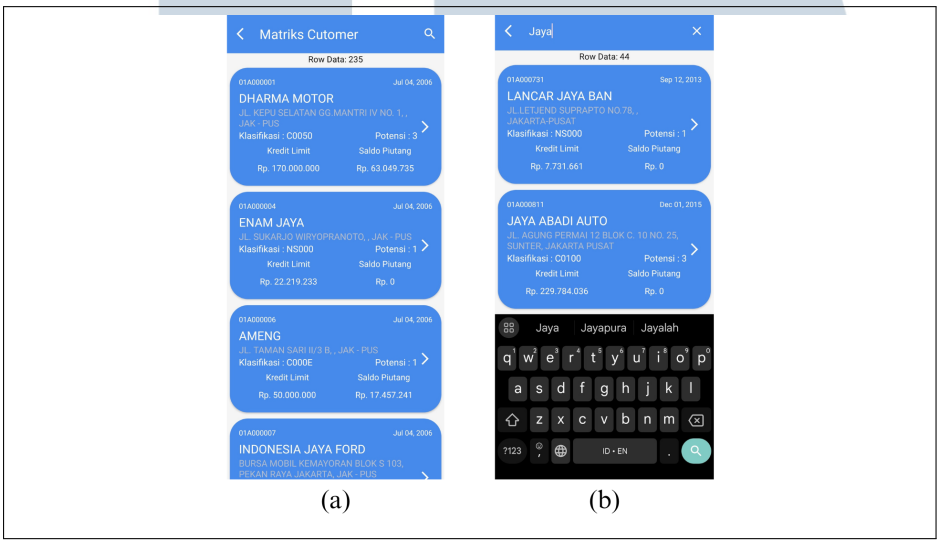
```
1 public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState) {
2     View view =
    inflater.inflate(R.layout.fragment_transaction,
    container, false);
3     btnDaftarKunjungan =
    view.findViewById(R.id.daftar_kunjungan);
4     btnDaftarKunjungan.setOnClickListener(v-> {
5         Intent intent = new Intent(getActivity(),
```

```
6           TransactionDaftarKunjungan.class);
7           startActivity(intent);
8       });
9       btnMatriksCustomer =
        view.findViewById(R.id.matriks_customer);
10      btnMatriksCustomer.setOnClickListener(v-> {
11          Intent intent = new Intent(getActivity(),
        TransactionMatriksCustomer.class);
        startActivity(intent);
12      });
13      tabLayout = view.findViewById(R.id.tabLayout);
14      viewPager = view.findViewById(R.id.viewPager);
15      VPAdapter vpAdapter = new
        VPAdapter(getChildFragmentManager(),
        FragmentPagerAdapter.BEHAVIOR_RESUME_ONLY_CURRENT
16      _FRAGMENT);
17      vpAdapter.addFragment
18      (new FragTransaction1(), "Ordering");
19      vpAdapter.addFragment(new FragTransaction2(),
        "Process");
20      viewPager.setAdapter(vpAdapter);
21      tabLayout.setupWithViewPager(viewPager);
22      return view;
23  }
24
```

C.1 Matriks Customer

Pada Gambar 3.4 adalah tampilan dari *matriks customer*. Pada *Matriks Customer user* dapat melihat seluruh *customer* yang terdaftar dengan kodesalesnya. Sehingga setiap *sales* memiliki customer yang berbeda. *Matriks customer* akan berguna sebagai *menu* dimana *user* dapat melihat data *customer*, data gudang *customer*, dan melakukan order jika *customer* tidak masuk kedalam daftar kunjungan. Dibagian atas terdapat "*Row Count*" ini adalah jumlah dari *customer* yang udah pernah bertransaksi dengan *user*. *List* pada *Matriks customer* menggunaan *RecycleView*. Fungsi *scrollRefresh* digunakan untuk *refresh* halaman dan menampilkan data, seperti pada potongan Kode 3.3. Di dalam fungsi

scrollRefresh terdapat fungsi *loadMartikCust* yang menggunakan *prefix* dan nama *user* agar bisa menampilkan data yang sesuai dengan *customer user* seperti pada potongan kode 3.4. Setelah melakukan *loadMatrikCust* maka data akan ditampilkan pada *RecyclerView*. Data tersebut yang berada dalam *RecyclerView* dapat di klik untuk menunjukan *CustomerPopUp*. *Customer* yang dipilih pada *RecyclerView* akan selalu sesuai dengan yang muncul pada *Pop Up* karena menggunakan metode *get(position)* pada potongan Kode 3.4.



Gambar 3.4. Tampilan *Matriks Customer*

```
public void scrollRefresh(String prefix, String usersfa){
    pDialog.setMessage("Loading...");
    pDialog.setCancelable(false);
    pDialog.show();
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            mCustModelList.clear();
            loadMartikCust(prefix,usersfa,"");
        }
    },1200);
}

```

Kode 3.3: Potongan kode *scrollRefresh*

```
1  private void loadMartikCust(final String prefixdb_param,
   final String kodesales_param, final String
   textfilter_param) {
2      String tag_string_req = "req_customer";
3      pDialog.setMessage("Loading...");
4      showDialog();
5      StringRequest strReq = new
   StringRequest(Request.Method.POST,
6      AppConfig.URL_MCUST, new Response.Listener<String>() {
7          @Override
8          public void onResponse(String response) {
9              Log.d(TAG, "Response: " + response);
10             hideDialog();
11             try {
12                 JSONObject jObj = new JSONObject(response);
13                 int sukses = jObj.getInt("sukses");
14                 if (sukses == 1) {
15                     JSONArray dataArray =
   jObj.getJSONArray("Data");
16                     int rowData = jObj.getInt("row_data");
17                     rowcount.setText("Row Data: " +
   rowData);
18                     for (int i = 0; i < dataArray.length();
   i++) {
19                         try {
20                     JSONObject custObject =
   dataArray.getJSONObject(i);
21                     MCustModel item = new MCustModel(
22                     custObject.getString("KodeCust"),
23                     custObject.getString("JoinDate"),
24                     custObject.getString("NamaCust"),
25                     custObject.getString("Alamat1") + ", " +
26                     custObject.getString("Alamat2") + ", " +
27                     custObject.getString("Kota"));
28                             mCustModelList.add(item);
29                         } catch (JSONException e) {
30                             e.printStackTrace();
31                         }
```

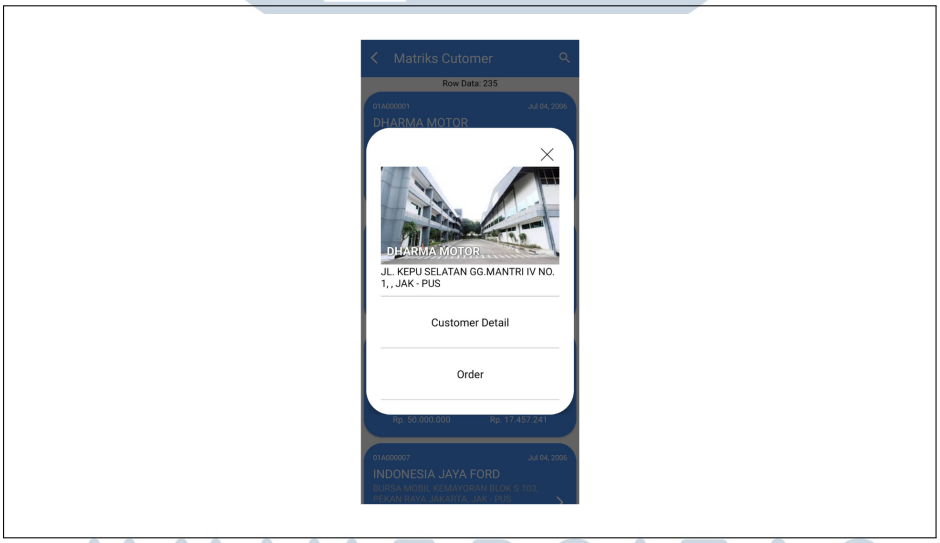
```
32                          mAdapter.notifyDataSetChanged();
33                      }
34                  } else {
35                      String message =
    jObj.getString("message");
36                      Toast.makeText(getApplicationContext(),
    "Error: " + message, Toast.LENGTH_LONG).show();
37                  }
38              } catch (JSONException e) {
39                  e.printStackTrace();
40                  Toast.makeText(getApplicationContext(),
    "Json error: " + e.getMessage(),
    Toast.LENGTH_LONG).show();
41              }
42          }
43      },
44              new Response.ErrorListener() {
45                  @Override
46                  public void onErrorResponse(VolleyError
    error) {
47                      Log.e(TAG, "Data Error: " +
    error.getMessage());
48                      Toast.makeText(getApplicationContext(),
    error.getMessage(), Toast.LENGTH_LONG).show();
49                      hideDialog();
50                  }
51              }) {
52          @Override
53          protected Map<String, String> getParams() {
54              Map<String, String> params = new HashMap<>();
55              params.put("prefixdb", prefixdb_param);
56              params.put("kodesales", kodesales_param);
57              params.put("textfilter", textfilter_param);
58              Log.e(TAG, params.toString());
59              return params;
60          }
61      };
62      strReq.setRetryPolicy(new DefaultRetryPolicy(
```

```
63              AppConfig.TIME_OUT_LOADING,
64              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
65              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
66      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
67  }
```

Kode 3.4: Potongan kode *loadMatrikCust*

C.2 Customer Pop Up

Setelah user melakukan klik pada salah satu *customer* maka muncul *Pop Up* seperti pada Gambar 3.5 yang menggunakan *AlertDialog* pada potongan Kode 3.5. Sehingga *CustomerPopUp* sebenarnya masih berada dalam satu *activity* yang sama dengan *matriks customer*, namun memiliki *file* xml yang berbeda. Pada *CustomerPopup user* dapat melihat data singkat dari *customer* dan dapat memilih untuk melihat *customer detail* atau *order*.



Gambar 3.5. Tampilan *Customer Pop Up*

```
1  private void showCustomerDialog(MCustModel customer){
2      AlertDialog.Builder builder = new AlertDialog.Builder
3
    (TransactionMatriksCustomer.this,R.style.AlertDialogTheme);
4      View view =
    LayoutInflater.from(getApplicationContext()).inflate(
5      R.layout.customer_popup,null);
```

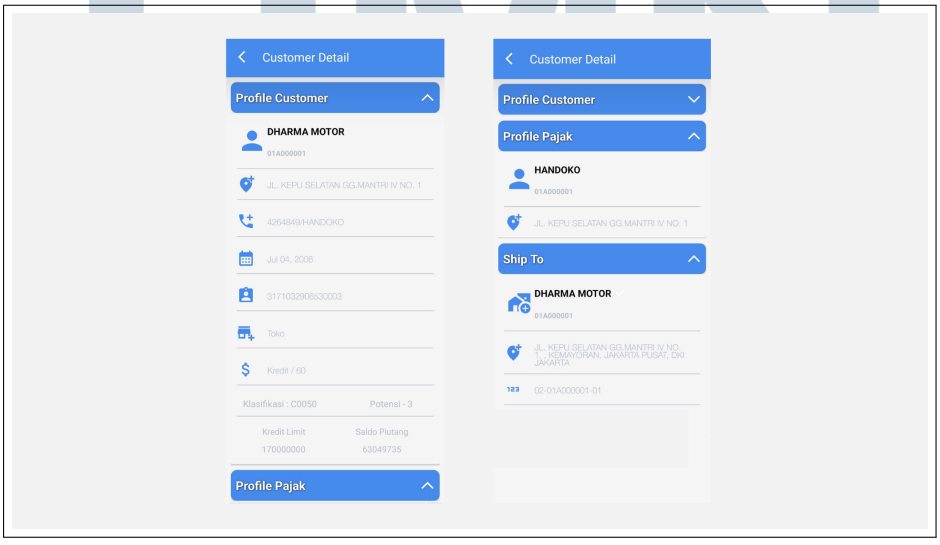
```
6     builder.setView(view);
7     String kodeCust = customer.getRes1();
8     ((TextView) view.findViewById(R.id.nama_customer)).
9     setText(customer.getRes3());
10    ((TextView) view.findViewById(R.id.alamat_customer)).
11    setText(customer.getRes4());
12    final  AlertDialog alertDialog = builder.create();
13    view.findViewById(R.id.close_button);
14    view.findViewById(R.id.close_button).
15    setOnClickListener(new View.OnClickListener() {
16        @Override
17        public void onClick(View v) {
18            alertDialog.dismiss();
19        }
20    });
    view.findViewById(R.id.cust_detail).setOnClickListener(new
    View.OnClickListener() {
21        @Override
22        public void onClick(View v) {
23            alertDialog.dismiss();
24            Intent intent = new
    Intent(TransactionMatriksCustomer.this,
    CustDetail.class);
25            intent.putExtra("kodeCust", customer.getRes1());
26            startActivity(intent);
27        }
28    });
29    view.findViewById(R.id.order).setOnClickListener(new
    View.OnClickListener() {
30        @Override
31        public void onClick(View v) {
32            alertDialog.dismiss();
33            Intent intent = new
    Intent(TransactionMatriksCustomer.this, Order.class);
34            intent.putExtra("kodeCust", customer.getRes1());
35            startActivity(intent);
36        }
37    });
```

```
38      if (alertDialog.getWindow()!= null){
39          alertDialog.getWindow().setBackgroundDrawable(new
    ColorDrawable(0));
40      }
41      alertDialog.show();
42  }
43
```

Kode 3.5: Potongan kode *customerPopUp*

C.3 Customer Detail

Pada bagian *customer* detail terdapat 3 bagian yaitu *profile customer*, *profile* pajak, dan *ship to* seperti Gambar 3.6. Bagian *profile customer* akan menjukan data *customer* seperti nama, kode *customer*, alamat, nomor telepon, tanggal bergabung, dan lainnya. sedangkan pada *profile* pajak berisikan nama pembayar pajak dan alamatnya. Terakhir adalah *ship to*, memuat alamat gudang pengiriman barang. Jika *customer* memiliki beberapa gudang makan akan ditampilkan semuanya. Pada tampilan *customer* detail menggunakan metode *dropdown* yang memanfaatkan *view.VISIBLE* dan *view.GONE* seperti pada potongan Kode 3.6 sehingga *user* tidak perlu melakukan *scroll* ke bagian paling bawah untuk melihat *ship to*. Menampilkan data pada customer detail menggunakan metode yang sama yaitu *POST* seperti pada potongan Kode 3.7



Gambar 3.6. Tampilan *Customer Detail*

```
dropdown1.setOnClickListener(view -> {
    if(hiddenLayout1.getVisibility() == View.VISIBLE){
        TransitionManager.
        beginDelayedTransition(hiddenLayout1, new
    AutoTransition());
        hiddenLayout1.setVisibility(View.GONE);
        dropdown1.setImageResource
        (R.drawable.ic_arrow_down);
    }else{
        TransitionManager.
        beginDelayedTransition(hiddenLayout1, new
    AutoTransition());
        hiddenLayout1.setVisibility(View.VISIBLE);
        dropdown1.setImageResource
        (R.drawable.ic_arrow_up);
    }});
dropdown2.setOnClickListener(view -> {
    if(hiddenLayout2.getVisibility() == View.VISIBLE){
        TransitionManager.
        beginDelayedTransition(hiddenLayout2, new
    AutoTransition());
        hiddenLayout2.setVisibility(View.GONE);
        dropdown2.setImageResource
        (R.drawable.ic_arrow_down);
    }else{
        TransitionManager.
        beginDelayedTransition(hiddenLayout2, new
    AutoTransition());
    hiddenLayout2.setVisibility(View.VISIBLE);
        dropdown2.setImageResource
        (R.drawable.ic_arrow_up);
    }});
dropdown3.setOnClickListener(view -> {
    if(hiddenLayout3.getVisibility() == View.VISIBLE){
        TransitionManager.
        beginDelayedTransition(hiddenLayout3, new
    AutoTransition());
    hiddenLayout3.setVisibility(View.GONE);
```

```
32
    dropdown3.setImageResource(R.drawable.ic_arrow_down);
33      }else{
34          TransitionManager.
35          beginDelayedTransition(hiddenLayout3, new
    AutoTransition());
36          hiddenLayout3.setVisibility(View.VISIBLE);
37          dropdown3.setImageResource(R.drawable.ic_arrow_up);
38      }});
39
```

Kode 3.6: Potongan kode metode *dropdown*

```
1  private void LoadCustDetail(final String prefixdb_param,
       final String kodecust_param) {
2       String tag_string_req = "req_custdetail";
3       showDialog();
4       StringRequest strReq = new
       StringRequest(Request.Method.POST,
       AppConfig.URL_CUSTOMER, new Response.Listener<String>() {
5           @Override
6           public void onResponse(String response) {
7               Log.d(TAG, "Response: " + response);
          hideDialog();
8               try {
9                   JSONObject jsonObject = new
       JSONObject(response);
10                  int sukses = jsonObject.getInt("sukses");
11                  if (sukses == 1) {
       nama_customer.setText
12                  (jsonObject.getString("NamaCust"));
13                  kode_cust.setText
14                  (jsonObject.getString("KodeCust"));
       alamat_customer.setText
15                  (jsonObject.getString("Alamat1"));
16                  notelp.setText
17                  (jsonObject.getString("TelpNo") + "/" +
       jsonObject.getString("KontakPerson"));
18                  join_date.setText
```

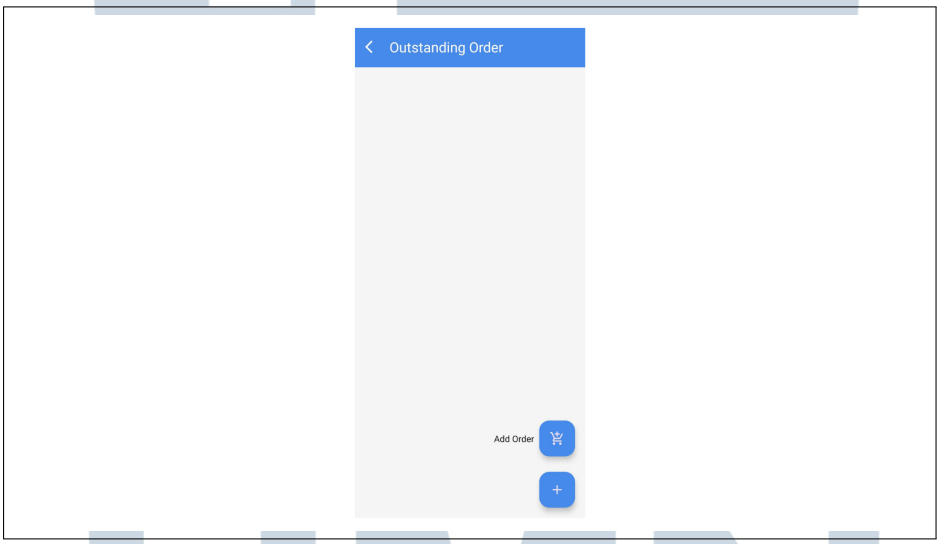
```
19                (jsonObject.getString("JoinDate"));
  no_ktp.setText
20                (jsonObject.getString("NoKTP"));
  jenis_usaha.setText
21                (jsonObject.getString("JenisUsaha"));
   String jenisUsahaCode =
  jsonObject.getString("JenisUsaha");
22                        String jenisUsahaMeaning =
  getJenisUsahaMeaning(jenisUsahaCode);
23
  jenis_usaha.setText(jenisUsahaMeaning);
24                        bayar_term.setText
25                        (jsonObject.getString("CaraBayar"));
26                        String caraBayarCode =
  jsonObject.getString("CaraBayar");
27                        String caraBayarMeaning =
  getCaraBayarMeaning(caraBayarCode);
28                        bayar_term.setText(caraBayarMeaning
  + " / " + jsonObject.getString("Term"));
29                        klasifikasi.setText("Klasifikasi :
  " + jsonObject.getString("Klasifikasi"));
30                        potensial.setText("Potensi - " +
  jsonObject.getString("Potensi"));kredit_limit.setText
31                        (jsonObject.getString("Limit"));
32                        saldo_piutang.setText
33
  (jsonObject.getString("SaldoPiutang"));
34                        nama_pajak.setText
35                        (jsonObject.getString("NamaPjk"));
36                        kode_cust_pajak.setText
37                        (jsonObject.getString("KodeCust"));
38                        alamat_pajak.setText
39                        (jsonObject.getString("Alamat1Pjk")
  + jsonObject.getString("Alamat2Pjk"));
40                    } else {
41                        String message =
  jsonObject.getString("pesan");
```

```
42
   Toast.makeText(getApplicationContext(), "Error: " +
   message, Toast.LENGTH_LONG).show();
43                  }
44              } catch (JSONException e) {
45                  e.printStackTrace();
46                  Toast.makeText(getApplicationContext(),
   "Json error: " + e.getMessage(),
   Toast.LENGTH_LONG).show();
47                  hideDialog();
48              }}
49          }, new Response.ErrorListener() {
50          @Override
51          public void onErrorResponse(VolleyError error) {
52              Log.e(TAG, "Data Error: " + error.getMessage());
53              Toast.makeText(getApplicationContext(),
54              error.getMessage(), Toast.LENGTH_LONG).show();
55          }}){
56          @Override
57          protected Map<String, String> getParams() {
58              Map<String, String> params = new HashMap<>();
59              params.put("prefixdb", prefixdb_param);
60              params.put("kodecust", kodecust_param);
61              Log.e(TAG, params.toString());
   return params;
62          }};
63     StrReq.setRetryPolicy(new DefaultRetryPolicy(
64              AppConfig.TIME_OUT_LOADING,
65              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
66              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
67     AppController.getInstance().addToRequestQueue(strReq,
68     tag_string_req);
69 }
70
```

Kode 3.7: Potongan kode *loadCustDetail*

C.4 Order

Saat *user* memilih tombol *order* pada *customer pop up* maka *user* akan masuk ke tampilan *order* seperti pada Gambar 3.7. Pada tampilan ini *user* dapat melihat *order* yang sedang berlangsung atau *outstanding order* pada *customer* tersebut. Pada bagian bawah kanan *user* dapat menekan tombol *add order* untuk menambahkan *order* untuk *customer*. Tombol *add order* menggunakan *floating button*. Hal ini dilakukan agar saat pengembang selanjutnya ingin menambahkan fitur, bisa ditambahkan pada *floating button* tersebut. *Floating button* dibuat menggunakan *view.VISIBLE* dan *view.GONE* seperti pada potongan Kode 3.8



Gambar 3.7. Tampilan *Order*

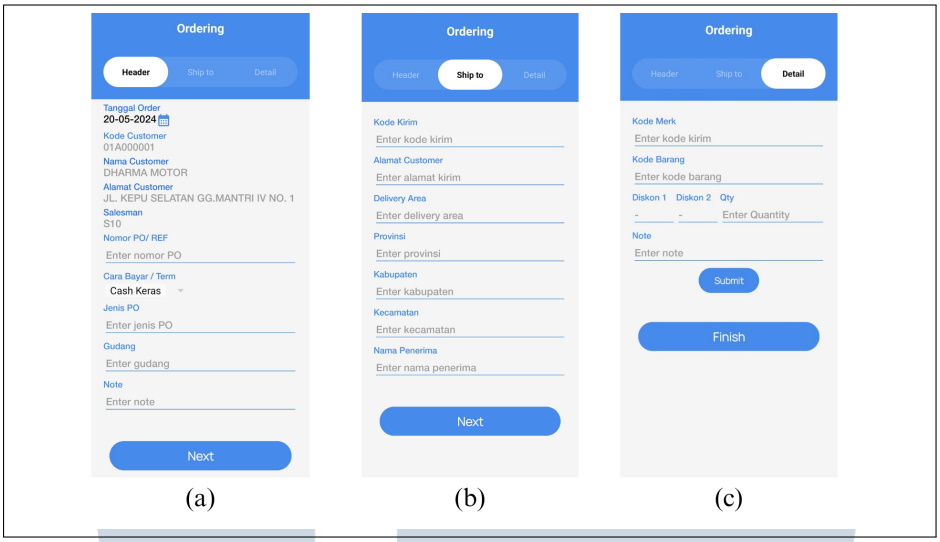
```
View view = toolbar.getChildAt(1);
view.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
    onBackPressed();
}
});
mAddFab = findViewById(R.id.add_fab);
mAddOrderFab = findViewById(R.id.add_order_fab);
addOrderActionText =
   findViewById(R.id.add_order_action_text);
mAddOrderFab.setVisibility(View.GONE);
addOrderActionText.setVisibility(View.GONE);
```

```
13 isAllFabsVisible = false;
14 mAddFab.setOnClickListener(v -> {
15 if (!isAllFabsVisible) {
16     mAddOrderFab.show();
17     addOrderActionText.setVisibility(View.VISIBLE);
18     isAllFabsVisible = true;
19 } else {
20     mAddOrderFab.hide();
21     addOrderActionText.setVisibility(View.GONE);
22     isAllFabsVisible = false;
23 }
24 });
25 mAddOrderFab.setOnClickListener(v -> {
26     Intent intent = new Intent(Order.this, Ordering.class);
       intent.putExtra("kodeCust", kodecust_param);
27     startActivity(intent);
28 });
29
```

Kode 3.8: Potongan kode *floating button*

C.5 Ordering

Pada Gambar 3.8 terdapat tiga bagian yaitu bagian (a) untuk fragment *header*, bagian (b) untuk bagian *ship to*, dan bagian (c) untuk bagian *detail*. Halaman ini dibuat menggunakan *fragment*, sehingga untuk berpindah *fragment* dibutuhkan sebuah fungsi untuk mengganti tampilan *tab* pada bagian atas tampilan. Maka dibuat fungsi *selectTab* pada potongan Kode 3.9 untuk berpindah fragment. Pada *fragment header, user* harus melakukan *input* berupa nomor itorder, nomor po, dan catatan. Selanjutnya pada bagian *ship to user* diminta untuk melakukan *input* kode kirim. Terakhir pada *fragment detail*. *User* akan memasukan kode merk, kode barang, diskon 1 dan 2 serta kuantitas dari pesanan *customer*. Selama proses *input* data yang diketik oleh *user* akan disimpan menggunakan *SQLite* metode yang digunakan dapat dilihat pada potongan Kode 3.10 dan harus membuat *DatabaseHelper*. Pada *DatabaseHelper* membuat tabel untuk data yang disimpan, potongan Kode **??**. Jika sudah selesai *user* harus menekan tombol *finish* untuk menyelesaikan *order*



Gambar 3.8. Tampilan *Ordering*

```
private void selectTab(int tabNumber){
    TextView tabItem1 =
   requireActivity().findViewById(R.id.tabitem1);
    TextView tabItem2 =
   requireActivity().findViewById(R.id.tabitem2);
    TextView tabItem3 =
   requireActivity().findViewById(R.id.tabitem3);
    TextView selectedTextView;
    TextView nonSelectedTextView1;
    TextView nonSelectedTextView2;
    if(tabNumber == 1){
        selectedTextView = tabItem1;
        nonSelectedTextView1 = tabItem2;
        nonSelectedTextView2 = tabItem3;
    } else if (tabNumber == 2) {
        selectedTextView = tabItem2;
        nonSelectedTextView1 = tabItem1;
        nonSelectedTextView2 = tabItem3;
    } else {
        selectedTextView = tabItem3;
        nonSelectedTextView1 = tabItem1;
        nonSelectedTextView2 = tabItem2;
    }
```

```
21      float slideTo = (tabNumber - 1) *
    selectedTextView.getWidth();
22      TranslateAnimation translateAnimation =
23      new TranslateAnimation(0,slideTo, 0, 0 );
24      translateAnimation.setDuration(1);
25      if(tabNumber == 1){
26      tabItem1.startAnimation(translateAnimation);
27      } else if (tabNumber == 2){
28      tabItem2.startAnimation(translateAnimation);
29      } else {
30      tabItem3.startAnimation(translateAnimation);
31      }
32      translateAnimation.setAnimationListener(new
    Animation.AnimationListener() {
33      @Override
34      public void onAnimationStart(Animation animation) {}
35      @Override
36      public void onAnimationEnd(Animation animation) {
37          selectedTextView.setBackgroundResource
38          (R.drawable.round_background_white100);
39          selectedTextView.setTypeface(null, Typeface.BOLD);
40          selectedTextView.setTextColor(Color.BLACK);
41          nonSelectedTextView1.setBackgroundColor
42
    (getResources().getColor(android.R.color.transparent));
43          nonSelectedTextView1.setTextColor
44          (getResources().getColor(R.color.white_shadow));
45
    nonSelectedTextView1.setTypeface(null,Typeface.NORMAL);
46          nonSelectedTextView2.setBackgroundColor
47
    (getResources().getColor(android.R.color.transparent));
48          nonSelectedTextView2.setTextColor
    (getResources().getColor(R.color.white_shadow));
49
    nonSelectedTextView2.setTypeface(null,Typeface.NORMAL);
50      }
51      @Override
```

```
52      public void onAnimationRepeat(Animation animation) {}});
53      }
54
```

Kode 3.9: Potongan kode *selectTab*

```
viewModel.setNoOrder(editTextNoOrder.getText()
.toString().trim());
viewModel.setNoPO(editTextNoPO.getText().toString().trim());
viewModel.setNote(editTextNote.getText().toString().trim());
DatabaseHelper myDB = new DatabaseHelper(getContext());
myDB.addDataFrag1(viewModel.getNoOrder().getValue(),
    viewModel.getNoPO().getValue(),
    viewModel.getNote().getValue());
     // Perform fragment transaction to replace fragment 1
    with fragment 2
    getParentFragmentManager().beginTransaction()
     .replace(R.id.fragmentContainer, new F_OrderingTwo())
     .addToBackStack(null)
     .commit();
     // Call selectTab function to update the tab view
     selectTab(2);

```

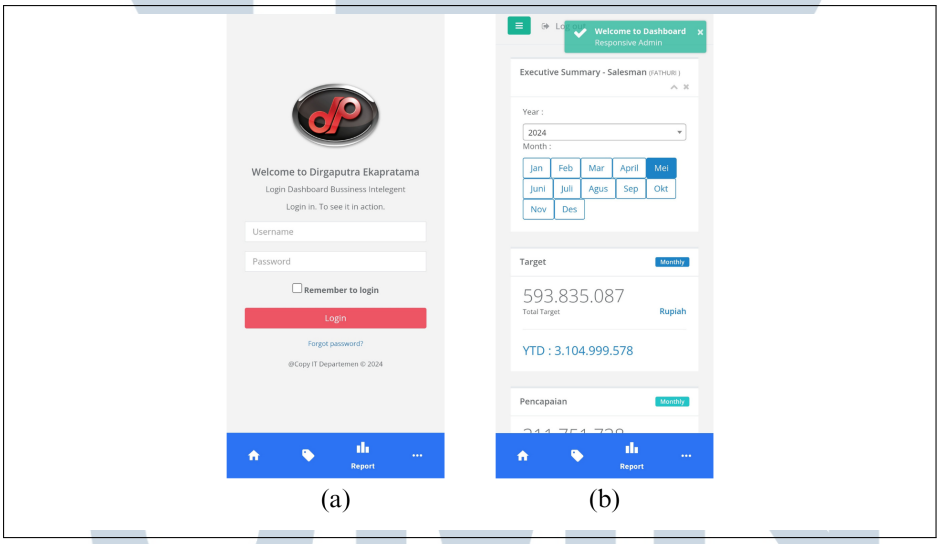
Kode 3.10: Potongan kode menyimpan data ke *SQLite*

```
@Override
public void onCreate(SQLiteDatabase db) {
     String query = "CREATE TABLE " + TABLE_NAME +
             " (" + COLUMN_ID + " INTEGER PRIMARY KEY
    AUTOINCREMENT, " +
         COLUMN_NOMOR_ORDER + " INTEGER, " +
         COLUMN_NOMOR_PO + " INTEGER," +
         COLUMN_NOTE + " TEXT," +
         COLUMN_KODE_KIRIM + " INTEGER);";
     db.execSQL(query);

     String createTableOrderingDetail = "CREATE TABLE " +
    TABLE_NAME2 +
             " (" + COLUMN_ID2 + " INTEGER PRIMARY KEY
    AUTOINCREMENT, " +
```

```
        COLUMN_KODE_MERK + " TEXT, " +
        COLUMN_KODE_BARANG + " TEXT, " +
        COLUMN_D1 + " REAL, " +
        COLUMN_D2 + " REAL, " +
        COLUMN_QTY + " INTEGER, " +
        COLUMN_NOTE_DETAIL + " TEXT);";
    db.execSQL(createTableOrderingDetail);
    }

```

Kode 3.11: Potongan kode *DatabaseHelper* label

D. Fragment Report



(a) (b)

Gambar 3.9. Tampilan *Report Page*

Reporting yang dilakukan pada PT. Dirgaputra Ekapratama masih menggunakan *website*, sehingga pada bagian *fragment reporting* akan dihubungkan dengan *website reporting*. Seperti pada Gambar 3.9 *user* akan melakukan *login* ulang di *website* tersebut, hal ini mempersulit karena harus melakukan *login* lagi pada aplikasi. Sehingga ditambahkan fitur yang akan menyimpan *cookie* dari *user* dengan menggunakan *CookieManager* dari *library android*. Aplikasi SFA dihubungkan dengan *web* menggunakan *WebView* seperti pada Kode 3.12.

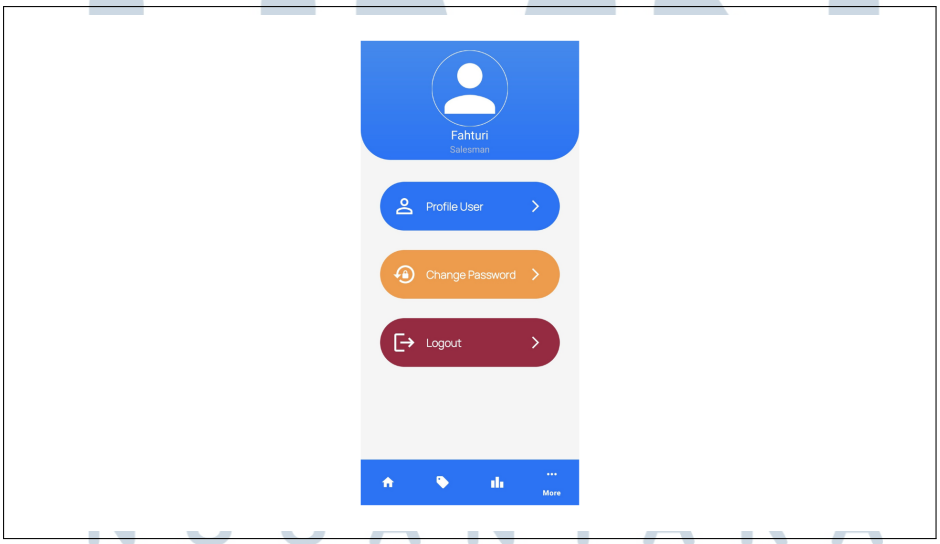
```
        webView = (WebView) view.findViewById(R.id.webview);
        webView.setWebViewClient(new WebViewClient());
        webView.loadUrl(
        "http://sfa.dirgaputra.co.id:85/dirgabi/index.php");
```

```
5         WebSettings webSettings = webView.getSettings();
6         webSettings.setJavaScriptEnabled(true);
7         webSettings.setDomStorageEnabled(true);
8         CookieManager cookieManager =
    CookieManager.getInstance();
9         cookieManager.setAcceptCookie(true);
10        cookieManager.acceptCookie();
11        String cookie =
    CookieManager.getInstance().getCookie("myWebsite");
    CookieManager.getInstance().getCookie
12
    ("http://sfa.dirgaputra.co.id:85/dirgabi/index.php");
```

Kode 3.12: Potongan kode reporting

E. Fragment More

Aplikasi lama SFA tidak memliki *menu more* sehingga *user* tidak dapat melihat *profile* dan tidak dapat mengganti *password* secara mandiri. Tampilan *profile user* meliputi foto *profile user*, nama, jabatan, tombol tombol *profile user*, tombol *change password*, dan tombol *logout*. Berikut adalah fungsi dan detail dari tombol pada *fragment more*.



Gambar 3.10. Tampilan *More Page*

E.1 Profile User

Bagian *profile user* akan melakukan menampilkan data *user profile* dan *salesman profile*. Data yang digunakan pada tampilan ini menggambil dari *session management* yang sebelumnya dibuat saat melakukan *login*. seperti pada Kode 3.13 fungsi *userSession()* akan melakukan *setText* pada id *TextView* yang berada di tampilan *layout*. Terdapat beberapa data yang bisa di *update* oleh *user* seperti nomor telepon, email, dan alamat.

(a) (b)

Gambar 3.11. Tampilan *Profile User*

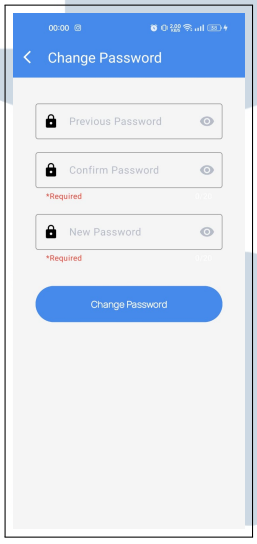
```
private void userSession() {
    sessionManagement = new SessionManagement(this);
    HashMap<String, String> user =
   sessionManagement.getUserDetails();
    username.setText
    (user.get(SessionManagement.KEY_IS_USERNAME));
    namalengkap.setText
    (user.get(SessionManagement.KEY_IS_NAMALENGKAP));
    prefixdb = user.get(SessionManagement.KEY_IS_PREFIX)
    .toString().trim();
    usersfa = user.get(SessionManagement.KEY_IS_USERSFA)
    .toString().trim();
    departemen.setText
    (user.get
    (SessionManagement.KEY_IS_DEPARTEMEN));
    jabatan.setText
```

```
16      (user.get(SessionManagement.KEY_IS_JABATAN));
17      cabang.setText
18      (user.get(SessionManagement.KEY_IS_KODECABANG));
19      level.setText
20      (user.get(SessionManagement.KEY_IS_KODELEVEL));
21  }
22
```

Kode 3.13: Potongan kode *profile user*

E.2 Change Password

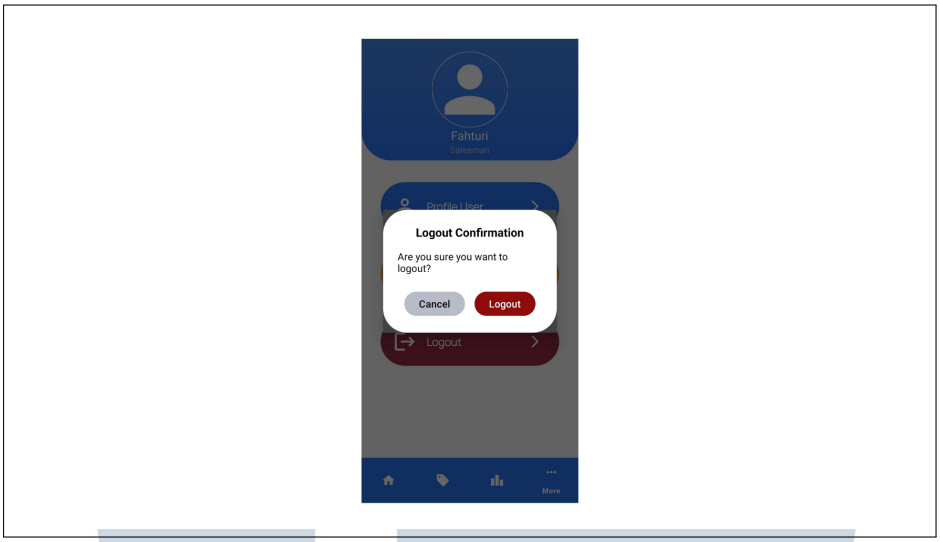
Tampilan halaman *change password* menggunaan *TextInputLayout* untuk menerima *input* dari *user*. Setelah *user* melakukan pengisian pada *previous password, confirm password,* dan *new password* maka *password user* untuk melakukan login akan terganti.



Gambar 3.12. Tampilan *Profile User*

E.3 Logout

Saat tombol logout ditekan, maka akan muncul *popUpAlert dialog* yang melakukan konfirmasi apakah *user* benar benar ingin melakukan *logout*. Jika *user* melakukan menekan tombol *logout* pada *alert dialog*, maka *session* akan dihapus dan *user* berhasil melakukan *logout*, hal ini dilakukan dengan fungsi *LogOut* yang sudah dibuat pada *SessionManagement*. Sedangkan jika user menekan tombol *cancel* maka *alert dialog* akan tertutup.



Gambar 3.13. Tampilan *alert dialog logout*

```
public void LogOut(){
    editor.clear();
    editor.commit();
    Intent i = new Intent(context, LoginActivity.class);
    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
    i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(i);
}

```

Kode 3.14: Potongan kode fungsi LogOut()

```
public Dialog onCreateDialog(@Nullable Bundle
    savedInstanceState) {
    AlertDialog.Builder builder = new
    AlertDialog.Builder(getActivity());
    builder.setTitle("Logout Confirmation")
    .setMessage("Are you sure you want to logout?")
    .setPositiveButton("Logout", new
    DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int
    which) {
            SessionManagement sessionManagement = new
    SessionManagement(requireActivity());
            sessionManagement.LogOut();
```

```
9           }
10      })
11      .setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
12          @Override
13          public void onClick(DialogInterface dialog, int
    which){
14              dialog.dismiss();
15          }
16      });
17      return builder.create();
18  }
19
```

Kode 3.15: Potongan kode alertdialog logout

3.3.2 Membantu staf infrastuktur melakukan instalasi

Untuk meningkatkan keamanan dan efisiensi operasional di kantor PT. Dirgaputra Ekapratama, dilakukan instalasi aplikasi *antivirus* ESET pada semua laptop dan komputer. ESET dikenal memiliki reputasi baik dalam melindungi sistem dari ancaman *malware*, *virus*, dan serangan *cyber* [**?**]. Proses instalasi ini dibimbing oleh Bapak Dhani Wijaya, seorang profesional IT bagian infrastruktur yang berpengalaman, yang memastikan setiap langkah dilakukan dengan benar dan sesuai standar keamanan. Selain itu, beliau juga memberikan panduan penggunaan dan pemeliharaan *antivirus*. Diharapkan, langkah ini akan melindungi sistem dan data kantor dari ancaman keamanan, menciptakan lingkungan kerja yang aman dan produktif bagi seluruh karyawan.

3.4 Kendala dan Solusi yang Ditemukan

Kendala-kendala yang ditemukan selama pelaksanaan magang di PT. Dirgaputra Ekapratama adalah:

1. Kurangnya pemahaman tentang bahasa *Java* dalam pengembangan aplikasi Android menjadi tantangan utama dalam proses pengembangan.

2. Revisi tampilan antarmuka pengguna (UI), terutama dalam pemilihan warna, yang memerlukan beberapa diskusi untuk mencapai desain yang diinginkan.

Upaya-upaya yang telah dilakukan dalam mengatasi kendala-kendala yang ditemukan antara lain:

1. Meningkatkan pemahaman bahasa pemrograman *Java* dengan memanfaatkan media *online* untuk belajar, seperti *video online* dan dokumentasi bahasa pemrograman *Java*.
2. Mencoba berbagai skema warna dan tampilan UI diiringi dengan diskusi bersama tim IT, hingga menemukan warna yang cocok.