BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Pelaksanaan magang di divisi IT dilakukan dengan peran sebagai *Software Developer Intern*. Kegiatan ini berada di bawah pengawasan Bapak Adi Purnomo, Kepala Divisi IT, dan bimbingan Bapak Chaerul Sazali, staf bagian *Development*. Selama magang, tugas utama adalah mengembangkan aplikasi *Sales Force Automation* (SFA), dengan proses pengembangan yang melibatkan koordinasi melalui diskusi langsung bersama Bapak Chaerul Sazali dan tim *sales* Jakarta.

3.2 Tugas yang Dilakukan

Magang yang dilakukan di PT. Dirgaputra Ekapratama melibatkan tugas penting yang memberikan pengalaman dalam dunia kerja. Tugas yang diberikan adalah pengembagan fitur aplikasi *Sales Force Automation* (SFA). Terdapat beberapa fitur utama yang krusial untuk dikembangakan untuk membantu aktivitas tim *sales* PT. Dirgaputra Ekapratama melakukan penjualan. Berikut uraian tugas yang dilakukan selama kerja magang:

1. Membuat desain antarmuka pengguna (UI/UX) untuk fitur-fitur baru

   Meliputi fitur *order*, daftar faktur, *check-in*, *profile customer*,

2. Menggunakan bahasa pemrograman java sebagai dasar pegembangan aplikasi SFA

3. Melakukan *display* data menggunakan *volley* dari *webservices* yang disediakan mentor

3.3 Uraian Pelaksanaan Magang

Berikut adalah penjelasan mengenai pelaksanaan kegiatan magang yang dilakukan di PT. Dirgaputra Ekapratama

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Merancang tampilan untuk fitur daftar faktur serta faktur *detail* menggunakan file .xml pada Android Studio.
2	Melakukan perbaikan pada tampilan Daftar Kunjungan dan Daftar *Customer*.
3	Mengubah tampilan *customer profile* menjadi *dropdown*.
4	Membuat halaman *outstanding order* untuk *order* yang sedang berjalan, serta menambahkan *list, item, adapter,* dan *model* untuk *outstanding order*.
5	Menerapkan hasil kode ke halaman *transaction* menggunakan *fragment*, serta memperbaiki tampilannya.
6	Membuat tampilan dan kode untuk menampilkan *order* yang sudah berada pada *process counter*.
7	Membuat tampilan baru untuk item Daftar *Customer* dan Daftar Kunjungan, serta menambahkan *customer popup* dengan *button* ke *order*, *detail*, dan *check-in*.
8	Membuat fitur foto *customer* agar *sales* dapat menambahkan foto toko *customer*, serta melengkapinya dengan opsi *zoom* dan *slide* foto untuk menampilkan foto lainnya.
9	Membuat *field input* untuk *edit* nomor telepon, email, dan alamat pada bagian *profile*, serta menambahkan fitur *check-in* pada Daftar *Customer* dan Daftar Kunjungan.
10	Menambahkan kelengkapan fitur *check-in* pada Daftar *Customer* dan Daftar Kunjungan, serta menambahkan *permission checking* untuk akses lokasi, notifikasi, dan akses menyalakan lokasi.
11	Menambahkan *list detail order* pada *order* yang sedang berlangsung, serta membuat halaman *order* (Header) *form order*.
12	Membuat halaman *order* (Barang) *form order detail* untuk barang yang akan dipilih, serta menambahkan SQLite untuk menyimpan *order* sementara.
13	Membuat skema perhitungan harga *order* yang sudah terkena diskon dan pajak, serta menambahkan validasi pada fitur *order* agar tidak terjadi kesalahan pengisian data.
Lanjut pada halaman berikutnya	

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

Minggu Ke -	Pekerjaan yang dilakukan
14	Menambahkan *list merk* dan barang pada halaman *order*, serta membuat fitur *search* untuk *merk* dan barang berdasarkan nama dan kode.
15	Mengubah halaman *profile user* dengan menghapus *sales profile* dan memperbarui tampilan *fragment transaction*.
16	Menambahkan fitur *search* pada daftar faktur, menambahkan *days count* dengan warna merah untuk lebih dari 60 hari, serta menghapus *detail* barang faktur.
17	Menambahkan asset pada *"no result found"*, *"no ongoing order"*, dan *"location not turned on"*, serta membuat aplikasi tidak dapat digunakan jika lokasi belum diaktifkan.
18	Melakukan *request* lokasi secara *background* untuk mendukung fitur *tracking*, serta melakukan *meeting* dengan *sales* (*user*) untuk membahas fitur yang ada.
19	Membuat *job scheduler* untuk melakukan *post* lokasi *sales* sesuai dengan interval yang ditentukan, serta membuat fitur *change location customer* yang dapat diperbaharui oleh *sales*.
20	Mengubah tampilan *login*, merapikan tampilan*login page*, *homepage*, Daftar *Customer*, Daftar Kunjungan, dan Daftar Faktur.
21	Menambahkan fitur *change password* dan fitur *update* versi aplikasi.
22	Melakukan optimasi pada fitur *search* untuk *merk*, barang, dan faktur, serta memperbaiki *bug* kecil yang ditemukan saat *testing*.
23	Menyempurnakan fitur foto *customer* dengan menambahkan opsi penghapusan foto, serta melakukan *testing* menyeluruh pada fitur ini.
24	Memperbaiki desain halaman order untuk memastikan *user experience* yang lebih baik, serta mengoptimalkan performa SQLite untuk penyimpanan sementara.

3.3.1 Fitur Aplikasi Sales Force Automation

Fitur-fitur yang dikembangkan pada aplikasi *Sales Force Automation* (SFA) ini dirancang untuk mendukung aktivitas *sales* secara efisien dan terintegrasi. Aplikasi ini mencakup berbagai fitur utama seperti *Splash Screen*, *Login Page*, *Homepage*, dan beberapa *fragment* yang mendukung kebutuhan operasional *sales*.

A. Splash Screen

Sebelum user masuk ke dalam halaman *login*, user akan ditampilkan *splash screen* saat membuka aplikasi. Pada bagian ini terdapat beberapa proses di belakangnya, yang pertama adalah mendapatkan versi aplikasi yang digunakan menggunakan PackageInfo dan *PackageManager* lalu versi akan dibandingkan dengan versi yang ada di *webservice* menggunakan metode *post volley* seperti pada Potongan Kode 3.1. Pada fungsi *checkAppVersion* terdapat beberapa data yang dijadikan parameter untuk *checking* seperti token, imei ponsel, nama ponsel, *email* ponsel, versi aplikasi, dan kode aplikasi.

```
1  private void checkAppVersion(final String deviceImei_param,
       final String deviceName_param, final String
       deviceEmail_param, final String currentVersion_param) {
2       String tag_string_req = "req_version";
3       StringRequest strReq = new
       StringRequest(Request.Method.POST,
4               AppConfig.URL_UPDATE_VERSION, new
       ResponseListener<String>() {
5           @Override
6           public void onResponse(String response) {
7               Log.d(TAG, "Version Check Response: " +
       response);
8               if(response != null){
9                   try {
10                      JSONObject jObj = new
       JSONObject(response);
11                      int sukses = jObj.getInt("sukses");
12                      if (sukses == 1 ) {
13                          String latestVersion =
       jObj.getString("AppVersion");
```

```
14                          String downloadUrl =
    jObj.getString("FileUrl");
15                          Log.d(TAG, "Latest Version: " +
    latestVersion);
16                          showUpdateDialog(downloadUrl);
17                      } else if (sukses==2){
18                          //berhasil ini
19                          handler.postDelayed(() -> {
20                              sessionManagement.checkLogin();
21                              finish();
22                              }, SPLASH_DELAY);
23                      } else {
24                          String message =
    jObj.getString("message");
25
    Toast.make.Text(getApplicationContext(), "Error: " +
    message, Toast.LENGTH_LONG).show();
26                      }
27                  } catch (JSONException e) {
28                      e.printStackTrace();
29                  }
30              } else{
31
    Toast.make.Text(getApplicationContext(),"Server never
    retrive any data sended from you",
    Toast.LENGTH_LONG).show();
32              }
33          }
34      },
35              new Response.ErrorListener() {
36                  @Override
37                  public void onErrorResponse(VolleyError
    error) {
38                      Log.e(TAG, "Version Check Error: " +
    error.getMessage());
39
    Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
```

```
    Toast.LENGTH_LONG).show();
                }
            }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<>();
            params.put("token", "-");
            params.put("device_ime", deviceImei_param);
            params.put("device_name", deviceName_param);
            params.put("device_email", deviceEmail_param);
            params.put("versi_app", currentVersion_param);
            params.put("app_code", "100");
            Log.e(TAG, "Flag Checking"  + deviceImei_param
    + ", " + deviceName_param + ", " +deviceEmail_param + ",
    " +currentVersion_param);
            return params;
        }
     };
}

```

Listing 3.1: Potongan kode untuk cek versi aplikasi

Selanjutnya jika aplikasi yang dimiliki *user* berbeda dengan versi terbaru yang ada maka *user* akan ditampilkan *alert dialog* seperti pada potongan Kode 3.2 yang bertujuan untuk *user* melakukan *update* aplikasi menggunakan *link* yang didapatkan dari *response webservice*.

```
private void showUpdateDialog(String downloadUrl) {
    AlertDialog`.Builder builder = new
    AlertDialog.Builder(this);
    builder.setTitle("Update Available");
    builder.setMessage("A new version of the app is
    available. Please update to continue.");
    builder.setCancelable(false);
    builder.setPositiveButton("Update", new
    DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int
    which) {
```

```
            Intent intent = new Intent(SplashActivity.this,
    UpdateAppActivity.class);
            intent.putExtra("url", downloadUrl);
            startActivity(intent);
        }
    });
    builder.setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int
    which) {
            dialog.dismiss();
        }
    });
    builder.show();
}

```

Listing 3.2: Potongan kode untuk menampilkan alert dialog

Url yang diterima akan digunakan pada *UpdateAppActivity* untuk melakukan *download file* yang dilakukan dalam *WebView*. *WebView* akan menampilkan url yang diterima. Tujuan penggunaan *WebView* sehingga user tidak perlu membuka aplikasi lain seperti *browser* mereka dan tetap berada pada aplikasi SFA. Potongan Kode 3.3 menunjukan kode untuk melakukan *update* dan *download* aplikasi.

```
private void downloadFile(String url, String mimeType) {
    Download`Manager.Request request = new
    Download`Manager.Request(Uri.parse(url));
    request.setMimeType(mimeType);
    request.set`Title("Downloading APK");
    request.set`Description("Downloading the application
    update...");

    request.set`NotificationVisibility(DownloadManager.Request
    .VISIBILITY_VISIBLE_NOTIFY_COMPLETED);
    if (android.os.Build.VERSION.SDK_INT >=
    android.os.Build.VERSION_CODES.Q) {
        request.setDestinationInExternalFilesDir
        (this, Environment.DIRECTORY_DOWNLOADS,
```

```
"SFA_V0.0.0.apk");
 } else {
 request.setDestinationInExternalPublicDir
 (Environment.DIRECTORY_DOWNLOADS, "SFA_V0.0.0.apk");
 }
 Download Manager download Manager = (DownloadManager)
getSystemService(Context.DOWNLOAD_SERVICE);
 if (downloadManager != null) {
     long downloadId = downloadManager.enqueue(request);
     // Monitor download progress in a separate thread
     new Thread(() -> {
         boolean downloading = true;
         while (downloading) {
             DownloadManager.Query query = new
DownloadManager.Query();
             query.setFilterById(downloadId);
             Cursor cursor =
downloadManager.query(query);
             try {
                 if (cursor != null &&
cursor.moveToFirst()) {
                     int statusIndex =
cursor.getColumnIndex(DownloadManager.COLUMN_STATUS);
                     if (statusIndex >= 0) {
                         int status =
cursor.getInt(statusIndex);
                         if (status ==
DownloadManager.STATUS_SUCCESSFUL) {
                             downloading = false;
                             runOnUiThread(() -> {

progressBar.setVisibility(View.GONE);
                                 Log.e(TAG, "Download
completed");
                             });
                         } else if (status ==
DownloadManager.STATUS_FAILED) {
                             downloading = false;
```

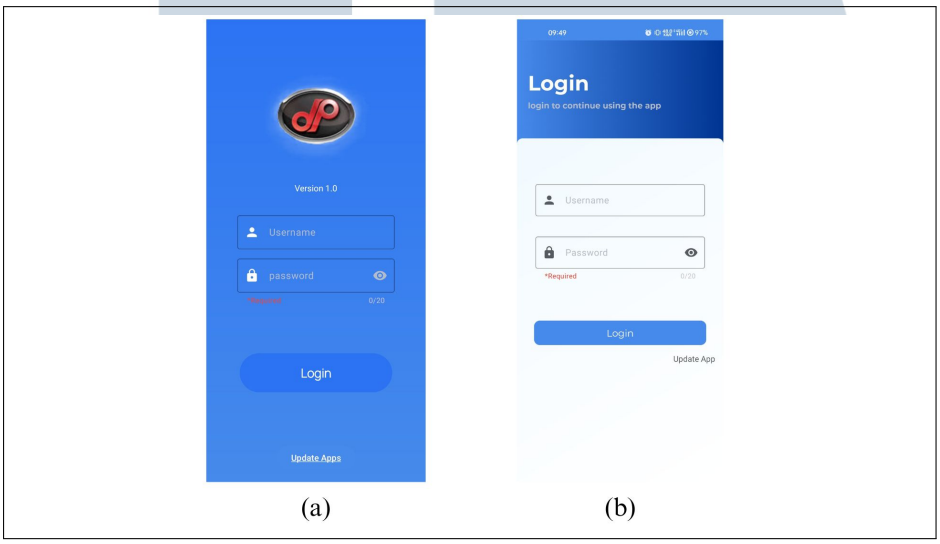
```
                                    runOnUiThread(() -> {
                                        progressBar.setVisibility(View.GONE);
                                        Log.e(TAG, "Download failed");
                                    });
                                }
                            } else {
                                Log.e(TAG, "COLUMN_STATUS index not found.");
                                downloading = false;
                            }
                        }
                    } finally {
                        if (cursor != null) {
                            cursor.close();
                        }
                    }
                    try {
                        Thread.sleep(500); // Poll every 500ms
                    } catch (InterruptedException e) {
                        Log.e(TAG, "Thread interrupted: " + e.getMessage());
                    }
                }
            }).start();
        } else {
            Log.e(TAG, "DownloadManager is not available");
            runOnUiThread(() -> progressBar.setVisibility(View.GONE));
        }
    }
```

Listing 3.3: Potongan kode untuk update aplikasi

B. Login Page

Tampilan halaman *login* telah dirancang ulang dengan mengganti latar belakang yang sebelumnya menggunakan warna biru muda *solid* seperti pada

Gambar 3.1 bagian (a) menjadi warna putih dengan sentuhan aksen biru seperti pada Gambar 3.1 bagian (b), untuk menciptakan tampilan yang lebih bersih, modern, dan profesional. Warna putih dipilih karena memberikan kesan sederhana, meningkatkan fokus pada elemen penting, serta memudahkan pembacaan teks. Hal ini sesuai dengan temuan Hall dan Hanna (2004) yang menunjukkan bahwa warna putih pada latar belakang lebih nyaman bagi user dan mengurangi kelelahan mata [2]. Penambahan aksen biru pada bagian atas menjaga identitas visual aplikasi.



Gambar 3.1. Halaman *Login page*

C. Home Page

Home page dalam aplikasi ini adalah bagian utama yang mengelola navigasi menggunakan *BottomNavigationView* untuk memuat empat *fragment* utama yaitu *Home*, *Transaction*, *Report*, dan *More*. Aktivitas ini juga menangani beberapa fitur penting, termasuk pemeriksaan dan permintaan izin akses lokasi serta GPS untuk memastikan fitur yang membutuhkan lokasi dapat berjalan dengan baik seperti pada Potongan Kode 3.4. Jika GPS tidak aktif, *dialog* khusus ditampilkan untuk meminta *user* mengaktifkan layanan lokasi. Selain itu, *ActivityResultLauncher* yang digunakan untuk menangani izin dengan lebih efisien, dan mekanisme *onActivityResult* memastikan *user* diingatkan untuk mengaktifkan GPS jika masih dinonaktifkan. Bagian ini juga mendukung navigasi *fragment* menggunakan *replace transactions* dan metode khusus untuk memuat *fragment* sesuai dengan kebutuhan seperti pada Potongan Kode 3.5. Untuk keperluan *background task*, aktivitas ini menyediakan metode *scheduleJob* yang menggunakan *JobScheduler*

untuk menjadwalkan tugas periodik seperti pelacakan lokasi user dengan interval minimum yang ditentukan sesuai batas Android (15 menit) [3]. Hal ini menjadikan aktivitas ini sebagai pusat pengelolaan navigasi, izin, dan tugas latar belakang dalam aplikasi.

```
mPermissionResultLauncher = registerForActivityResult(new
    ActivityResultContracts.RequestMultiplePermissions(),
    new ActivityResultCallback<Map<String, Boolean>>() {
     @Override
     public void onActivityResult(Map<String, Boolean>
    result) {
          if
    (result.get(Manifest.permission.ACCESS_FINE_LOCATION) !=
    null) {
               isLocationPermissionGranted =
    result.get(Manifest.permission.ACCESS_FINE_LOCATION);
          }
          if
    (result.get(Manifest.permission.ACCESS_BACKGROUND_LOCATION)
    != null) {
               isLocationPermissionGranted =
    result.get(Manifest.permission.ACCESS_BACKGROUND_LOCATION);
          }
          if
    (result.get(Manifest.permission.POST_NOTIFICATIONS) !=
    null) {
               isNotificationPermissionGranted =
    result.get(Manifest.permission.POST_NOTIFICATIONS);
          }
     }
});

```

Listing 3.4: Potongan kode untuk navigasi antar fragmen

```
public boolean onNavigationItemSelected(@NonNull MenuItem
    item) {
     Fragment selectedFragment = null;
     int itemId = item.getItemId();
     if (itemId == R.id.nav_item_home) {
```

```
27          selectedFragment = firstFragment;
28      } else if (itemId == R.id.nav_item_transaction) {
29          selectedFragment = secondFragment;
30      } else if (itemId == R.id.nav_item_report) {
31          selectedFragment = thirdFragment;
32      } else if (itemId == R.id.nav_item_more) {
33          selectedFragment = fourthFragment;
34      }
35      if (selectedFragment != null) {
36          getSupportFragmentManager()
37                  .beginTransaction()
38                  .replace(R.id.flFragment, selectedFragment)
39                  .commit();
40          return true;
41      }
42      return false;
43  }
44
```

Listing 3.5: Potongan kode untuk menampilkan alert dialog

```
1  @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
2  private void scheduleJob() {
3      ComponentName componentName = new ComponentName(this,
       MyJobService.class);
4      JobInfo.Builder builder = new JobInfo.Builder(JOB_ID,
       componentName)
5              .setRequiresCharging(false)
6              .setPeriodic(15 * 60 * 1000);
7      JobScheduler jobScheduler = (JobScheduler)
       getSystemService(Context.JOB_SCHEDULER_SERVICE);
8      if (jobScheduler != null) {
9          jobScheduler.schedule(builder.build());
10     }
11 }
12
```

Listing 3.6: Potongan kode untuk menampilkan alert dialog

D. Fragment Home

Pada tampilan *fragment Home*, sebelumnya persentase yang diterima dari *webservice* ditampilkan langsung, misalnya ”80%”. Namun, perubahan ini merubah nilai tersebut dengan menghapus simbol ”%” menggunakan replace, sehingga hanya angka integer yang tersisa. Nilai integer ini kemudian digunakan untuk memperbarui *progress bar* melalui metode *updateProgressBar(int value)* seperti pada Potongan Kode 3.7, yang mengatur *progress bar* dan menampilkan nilai persentase dalam format ”X%”, di mana X adalah angka yang sudah dibersihkan.

```
String cleanedPercent = percentage.replaceAll("[^0-9]", "");
int value = Integer.valueOf(cleanedPercent);
updateProgressBar(value);
private void updateProgressBar(int value) {
    progressBar.setProgress(value);
    progressText.setText(value + "%");
}

```

Listing 3.7: Potongan kode untuk update progress bar

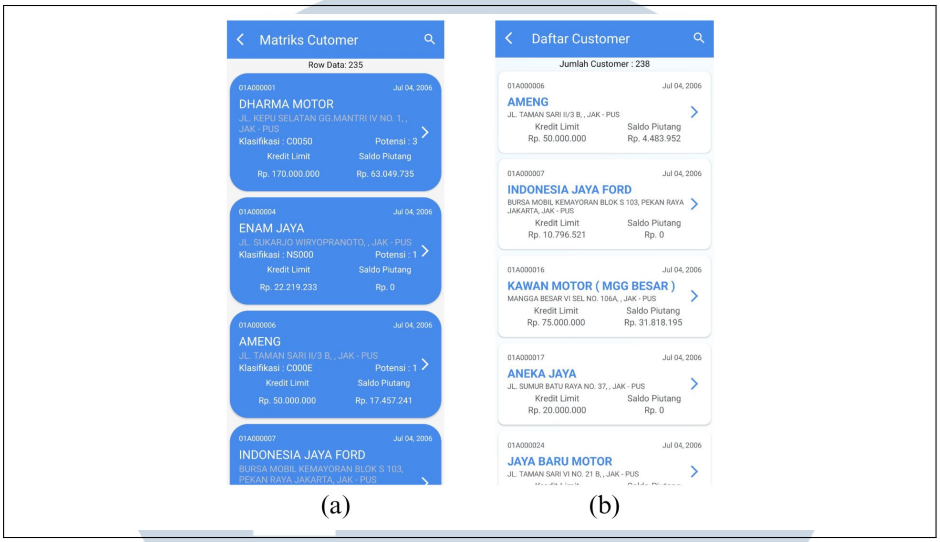
E. Fragment Transaction

Pada bagian ini, berbagai fitur dikembangkan, termasuk daftar kunjungan, daftar *customer*, daftar faktur, *ordering*, dan *process*. Setiap fitur ini juga mencakup berbagai fungsi tambahan yang mendukung operasional dan meningkatkan fungsionalitas aplikasi.

E.1 Daftar Kunjungan dan Daftar Customer

Pada bagian ini, dilakukan penyesuaian pada tampilan *item* di *RecyclerView* untuk meningkatkan keterbacaan dan estetika antarmuka *user*. Sebelumnya, *item* dalam daftar menggunakan warna latar belakang biru *solid* yang terlalu mencolok atau kurang nyaman secara visual. Oleh karena itu, warna latar belakang diubah menjadi putih agar tampilannya lebih bersih, modern, dan mudah dibaca. Perubahan ini dapat dilihat secara visual pada Gambar 3.4 bagian (a) memperlihatkan desain lama dengan warna biru *solid*, sementara Gambar 3.4 bagian (b) menampilkan desain baru dengan warna putih sebagai latar belakang. Penyesuaian ini dilakukan

untuk memberikan pengalaman *user* yang lebih baik, terutama ketika menangani daftar data yang panjang.



Gambar 3.2. Halaman Daftar Customer

E.2 Check-in

Pada process *check-in user* akan ditampilkan *dialog* untuk mengkonfirmasi ulang, selama proses ini berlangsung dan *button* konfirmasi ditekan maka aplikasi akan mendapatkan tanggal, *latitude* dan *longitude* untuk lokasi, nama *device*, MCC,MNC, LAC, CID, dan IP. Data tersebut akan dibuat kedalam *JSONObject* untuk dikirim ke *webservice* menggunakan fungsi *processCheckIn()* seperti pada Potongan Kode 3.9. Pada *processCheckIn()* respon dari *webservice* akan menjadi penentu pesan apa yang akan di tampilkan ke *user*. Jika status = 0 maka akan ditampilkan . sedangkan sukses = 1 maka akan ditampilkan *check-in failed* seperti pada Potongan Kode 3.9.

```
1 private void showCheckInDialog(DKunjunganModel kunjungan) {
2     AlertDialog.Builder builder = new
  AlertDialog.Builder(TransactionDaftarKunjungan.this,
  R.style.AlertDialogTheme);
3     View view =
  LayoutInflater.from(getApplicationContext()).inflate(R.layout.checkin_p
  null);
4     builder.setView(view);
```

```
5       ((TextView)
        view.findViewById(R.id.header)).setText("Check-In
        Confirmation");
6       ((TextView)
        view.findViewById(R.id.message)).setText("Are you sure
        you want to check in to " + kunjungan.getNamaCust() +
        "?");
7       final AlertDialog alertDialog = builder.create();
8
        view.findViewById(R.id.first_btn).setOnClickListener(new
        View.OnClickListener() {
9           @Override
10          public void onClick(View v) {
11              alertDialog.dismiss();
12          }
13      });
14
        view.findViewById(R.id.second_btn).setOnClickListener(new
        View.OnClickListener() {
15          @Override
16          public void onClick(View v) {
17              getLocation();
18              sessionManagement = new
        SessionManagement(getApplicationContext());
19              HashMap<String, String> userDetails =
        sessionManagement.getUserDetails();
20              String prefixdb =
        userDetails.get(SessionManagement.KEY_IS_PREFIX);
21              String type = "INSERT";
22              String kode_sales=
        userDetails.get(SessionManagement.KEY_IS_USERSFA);
23              String nama_sales =
        userDetails.get(SessionManagement.KEY_IS_NAMALENGKAP);
24              String kode_customer = kunjungan.getKodeCust();
25              String nama_customer = kunjungan.getNamaCust();
26              Calendar calendar = Calendar.getInstance();
27              SimpleDateFormat dateFormat = new
        SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
```

```
28            SimpleDateFormat timeFormat = new
    SimpleDateFormat("HH:mm:ss:SSS", Locale.getDefault());
29            SimpleDateFormat dateTimeFormat = new
    SimpleDateFormat("yyyy-MM-dd HH:mm:ss:SSS",
    Locale.getDefault());
30            String currentDate =
    dateTimeFormat.format(calendar.getTime());
31            String tanggal =
    dateFormat.format(calendar.getTime());
32            String currentTime =
    timeFormat.format(calendar.getTime());
33            String latitudeStr = String.valueOf(latitude);
34            String longitudeStr = String.valueOf(longitude);
35            String accuracyStr = String.valueOf(accuracy);
36            String manufacturer = Build.MANUFACTURER;
37            String model = Build.MODEL;
38            String mcc = "";
39            String mnc = "";
40            String lac = "";
41            String cid = "";
42            TelephonyManager telephonyManager =
    (TelephonyManager)
    getSystemService(Context.TELEPHONY_SERVICE);
43            if
    (ActivityCompat.checkSelfPermission(getApplicationContext(),
    Manifest.permission.READ_PHONE_STATE) ==
    PackageManager.PERMISSION_GRANTED &&
44
    ActivityCompat.checkSelfPermission(getApplicationContext(),
    Manifest.permission.ACCESS_FINE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED) {
45                CellLocation cellLocation =
    telephonyManager.getCellLocation();
46                if (cellLocation instanceof
    GsmCellLocation) {
47                    GsmCellLocation gsmCellLocation =
    (GsmCellLocation) cellLocation;
```

```
48                  lac =
    String.valueOf(gsmCellLocation.getLac());
49                  mcc =
    telephonyManager.getNetworkOperator().substring(0, 3);
50                  mnc =
    telephonyManager.getNetworkOperator().substring(3);
51                  cid =
    String.valueOf(gsmCellLocation.getCid());
52              }
53          }
54          String localIp =
    IpAddressHelper.getLocalIpAddress();
55          new GetLocalIpTask().execute();
56          JSONObject jsonObject = new JSONObject();
57          try {
58              jsonObject.put("KodeSales", kode_sales);
59              jsonObject.put("NamaSales", nama_sales);
60              jsonObject.put("KodeCust", kode_customer);
61              jsonObject.put("NamaCust", nama_customer);
62              jsonObject.put("Tanggal", tanggal);
63              jsonObject.put("TransDate", currentDate);
64              jsonObject.put("TransTime", currentTime);
65              jsonObject.put("Latitude", latitudeStr);
66              jsonObject.put("Longitude", longitudeStr);
67              jsonObject.put("Akurasi", accuracyStr);
68              jsonObject.put("DeviceName", manufacturer +
    " " +  model);
69              jsonObject.put("MCC", mcc);
70              jsonObject.put("MNC", mnc);
71              jsonObject.put("LAC", lac);
72              jsonObject.put("CID", cid);
73              jsonObject.put("IP", localIp);
74          } catch (JSONException e) {
75              e.printStackTrace();
76          }
77          processCheckIn(prefixdb,type,
    jsonObject.toString(),currentDate,currentTime,
    nama_customer);
```

```
            alertDialog.dismiss();
        }
    });
    if (alertDialog.getWindow() != null) {
        alertDialog.getWindow().setBackgroundDrawable(new
    ColorDrawable(0));
    }
    alertDialog.show();
}

```

Listing 3.8: Potongan kode dialog check-in

```
private void processCheckIn(final String
    param_prefixdb,final String param_textfilter,
                            final String param_json,final
    String param_curendate, final String param_curentime,
    final String param_namacust){
    String tag_string_req = "req_login";
    showDialog("Loading, please wait...");
    StringRequest strReq = new
    StringRequest(Request.Method.POST,
            AppConfig.URL_CHECKIN, new
    ResponseListener<String>() {
        @Override
        public void onResponse(String response) {
            Log.d(TAG, "Process Response: " +
    response.toString());
            hideDialog();
            try {
                JSONObject jObj = new JSONObject(response);
                int status = jObj.getInt("Status");
                String pesan = jObj.getString("msg");
                if (status == 0) {

    Toast.make.Text(TransactionDaftarKunjungan.this, pesan,
    Toast.LENGTH_LONG).show();
                    checkInSuccess(param_curendate,
    param_curentime);
```

```
18                  AppNotificationManager
    notificationManager = new
    AppNotificationManager(getApplicationContext());
19
    notificationManager.showNotificationSuccessCheckIn(param_namacust);
20              } else {
21                  checkInFailed(pesan);
22              }
23              } catch (JSONException e) {
24              // JSON error
25              e.printStackTrace();
26              Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
    Toast.LENGTH_LONG).show();
27          }
28      }
29   }, new Response.ErrorListener() {
30      @Override
31      public void onErrorResponse(VolleyError error) {
32          hideDialog();
33          String errorMessage = "Unknown error occurred";
34          if (error != null && error.getMessage() !=
    null) {
35              errorMessage = error.getMessage();
36          }
37
    Toast.make.Text(TransactionDaftarKunjungan.this,
    errorMessage, Toast.LENGTH_LONG).show();
38          checkInFailed(errorMessage);
39      }
40   }) {
41      @Override
42      protected Map<String, String> getParams() {
43          Map<String, String> params = new
    HashMap<String, String>();
44          params.put("prefixdb", param_prefixdb);
45          params.put("textfilter", param_textfilter);
46          params.put("jsondata", param_json);
```

```
47              Log.d(TAG, "Process : " + params);
48              return params;
49          }
50      };
51      strReq.setRetryPolicy(new
    DefaultRetryPolicy(AppConfig.TIME_OUT_LOADING,
52              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
53              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
54      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
55  }
56
```

Listing 3.9: Potongan kode process check-in

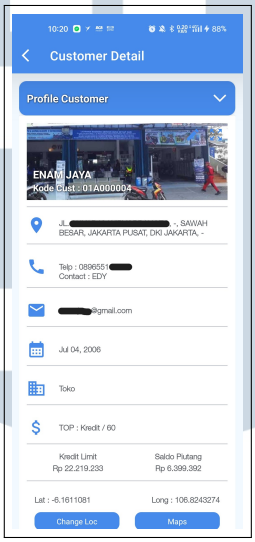
E.3 Customer Profile

Fitur ini dirancang untuk memberikan informasi mendetail tentang setiap *customer* kepada *sales*. Pada bagian atas halaman akan ditampilkan foto toko/lokasi *customer* disertai dengan informasi seperti nama *customer*, kode *customer*, nomor telepon, email, tanggal bergabung, jenis usaha, metode pembayaran, kredit *limit*, saldo piutang, alamat gudang, serta lokasi *customer* menggunakan koordinat *latitude* dan *longitude*. Data *customer* diambil ke *webservice* menggunakan metode *post* dengan memberikan kode *customer* sebagai parameternya. Selain itu, halaman ini juga dilengkapi dengan fitur pendukung, seperti mengambil foto lokasi *customer* (*get customer photo*), mengubah lokasi *customer* (*change location customer*), dan *map*. Berikut adalah tampilan dari *customer profile* yang menggunakan metode *dropdown* pada Gambar 3.3.

E.4 Get Customer Photo

Fitur *get customer photo* dibuat untuk memberikan kemudahan bagi tim *sales*, terutama saat melakukan kunjungan lokasi atau penagihan kepada *customer*. Dengan fitur ini *sales* dapat dengan cepat mengidentifikasi lokasi *customer*, terutama bagi sales baru yang belum pernah mengunjungi lokasi tersebut. Selain itu, fitur ini juga memungkinkan *sales* untuk menambahkan foto baru saat berada di lokasi *customer* sebagai dokumentasi tambahan.

Proses penambahan foto dilakukan dengan *ImagePicker*, *Imagepicker* dapat digunakan *user* untuk memilih gambar dari kamera langsung maupun dari galeri perangkatnya. Gambar yang dipilih akan diproses dan diubah ke dalam format *Base64* agar kompatibel untuk pengiriman data melalui jaringan. Format *Base64* memungkinkan gambar dikirim sebagai bagian dari *payload JSON*, memastikan file tetap utuh dan mudah diproses oleh server. Setelah itu, file gambar dalam format *Base64* ini akan dikirimkan ke server menggunakan metode *HTTP POST*, seperti yang ditunjukkan pada Potongan Kode 3.10.



Gambar 3.3. Halaman Customer Profile

```
try {
    InputStream inputStream =
   getContentResolver().openInputStream(uri);
    byte[] bytes = getBytes(inputStream);
    base64String = Base64.encodeToString(bytes,
   Base64.DEFAULT);
} catch (IOException e) {
    e.printStackTrace();
    Toast.make.Text(getApplicationContext(), "Failed to
   encode image to Base64", Toast.LENGTH_SHORT).show();
}
private void updateProfilePict() {
    String tag_string_req = "req_update_profile";
    showDialog("Updating Profile Pict...");
```

```
12      StringRequest strReq = new
    StringRequest(Request.Method.POST,
    AppConfig.URL_UPDATE_CUSTOMER, new
    ResponseListener<String>() {
13          @Override
14          public void onResponse(String response) {
15              Log.d("Response", response);
16              hideDialog();
17              try {
18                  JSONObject jsonObject = new
    JSONObject(response);
19                  String error =
    jsonObject.getString("error");
20                  String msg =
    jsonObject.getString("error_msg");
21                  if (error.equals("false")) {
22
    Toast.make.Text(getApplicationContext(), msg,
    Toast.LENGTH_LONG).show();
23                  } else {
24                      String message =
    jsonObject.getString("error_msg");
25
    Toast.make.Text(getApplicationContext(), "Error: " +
    message, Toast.LENGTH_LONG).show();
26                  }
27              } catch (JSONException e) {
28                  e.printStackTrace();
29                  Toast.make.Text(getApplicationContext(),
    "Json error: " + e.getMessage(),
    Toast.LENGTH_LONG).show();
30
31              }
32          }
33      }, new Response.ErrorListener() {
34          @Override
35          public void onErrorResponse(VolleyError error) {
36              Log.e("Data Error", error.getMessage());
```

```
37              hideDialog();
38              Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
    Toast.LENGTH_LONG).show();
39          }
40      }) {
41          @Override
42          protected Map<String, String> getParams() {
43              Map<String, String> params = new HashMap<>();
44              params.put("username", username);
45              params.put("kodecabang", kodecabang);
46              params.put("kodecust", kodecust_param);
47              params.put("foto", base64String);
48              return params;
49          }
50      };
51      strReq.setRetryPolicy(new DefaultRetryPolicy(
52              AppConfig.TIME_OUT_LOADING,
53              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
54              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
55      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
56  }
57
```

Listing 3.10: Potongan kode *get customer photo*

E.5 Get Location dan Map

Fitur *get location* digunakan *sales* untuk merekam lokasi toko atau tempat *customer*. Dengan menggunakan fitur ini, aplikasi akan mengambil data koordinat berupa *latitude* dan *longitude* dari lokasi toko atau *customer* saat ini. Koordinat tersebut kemudian akan dikirimkan ke server menggunakan layanan *webservice* untuk disimpan sebagai referensi lokasi *customer*.*Latitude* dan *Longitude* dapat dihitung untuk membatasi *sales*, *sales* harus berada pada jarak 100 meter dari titik referensi lokasi yang sebelumnya sudah ada pada *customer*. Hal ini bertujuan agar *supervisor* dari *sales* dapat memantau pekerjaan *sales*. Untuk menentukan batas maksimum 100 meter, nilai *latitude* dapat ditambah atau dikurangi sebesar 0,000898°, dan nilai longitude ditambah atau dikurangi sebesar 0,000902°.Proses ini dapat dilihat pada Potongan Kode 3.11.

```
1  private void processChangeLoc(final String prefixdb_param,
   final String kodecust_param) {
2      String tag_string_req = "req_update_profile";
3      showDialog("Updating Profile Pict...");
4      StringRequest strReq = new
   StringRequest(Request.Method.POST,
   AppConfig.URL_UPDATE_LatLongCust, new
   ResponseListener<String>() {
5          @Override
6          public void onResponse(String response) {
7              Log.d("Response", response);
8              hideDialog();
9              try {
10                 JSONObject jsonObject = new
   JSONObject(response);
11                 String error =
   jsonObject.getString("error");
12                 String msg =
   jsonObject.getString("error_msg");
13                 if (error.equals("false")) {
14
   Toast.make.Text(getApplicationContext(), msg,
   Toast.LENGTH_LONG).show();
15                     changeLocationSuccess();
16                 } else {
17                     changeLocationFailed(msg);
18
   Toast.make.Text(getApplicationContext(), "Error: " +
   msg, Toast.LENGTH_LONG).show();
19                 }
20             } catch (JSONException e) {
21                 e.printStackTrace();
22
23                 Toast.make.Text(getApplicationContext(),
   "Json error: " + e.getMessage(),
   Toast.LENGTH_LONG).show();
24             }
25         }
```

```
26      }, new Response.ErrorListener() {
27          @Override
28          public void onErrorResponse(VolleyError error) {
29              Log.e("Data Error", error.getMessage());
30              hideDialog();
31              Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
    Toast.LENGTH_LONG).show();
32          }
33      }) {
34          @Override
35          protected Map<String, String> getParams() {
36              Map<String, String> params = new HashMap<>();
37              params.put("username", username);
38              params.put("prefixdb", prefixdb_param);
39              params.put("kodecust", kodecust_param);
40              params.put("latitude",
    String.valueOf(newLatitude));
41              params.put("longitude",
    String.valueOf(newLongitude));
42              return params;
43          }
44      };
45      strReq.setRetryPolicy(new DefaultRetryPolicy(
46              AppConfig.TIME_OUT_LOADING,
47              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
48              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
49      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
50  }
51
```

Listing 3.11: Potongan kode *get location*

Selain menyimpan lokasi, aplikasi juga menyediakan fitur *map* yang akan memudahkan *sales* untuk menavigasi ke lokasi *customer*. Dengan menekan tombol ini, aplikasi akan membuka aplikasi peta seperti Google Maps menggunakan *intent*, lengkap dengan parameter *latitude* dan *longitude* dari lokasi *customer* yang telah tersimpan sebelumnya. Fitur ini memungkinkan *sales* untuk langsung diarahkan ke lokasi *customer*. Bertujuan untuk kunjungan maupun untuk sekadar melihat posisi toko atau tempat *customer* di peta. Potongan Kode 3.12 menunjukan uri yang dibuat

akan dimasukan menggunakan *intent* ke aplikasi seperti Google Maps.

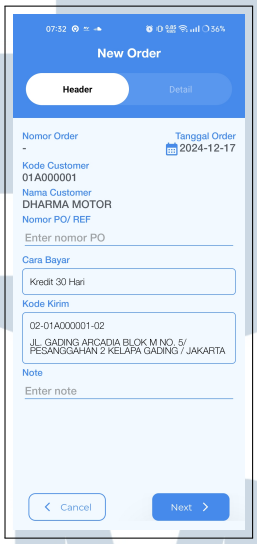
```
openMapBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Create a URI using latitude and longitude
        String geoUri = "geo:" + latitudeValue + "," + longitudeValue + "?q=" + latitudeValue + "," + longitudeValue;

        // Create an Intent to open Google Maps
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(geoUri));
        intent.setPackage("com.google.android.apps.maps"); // To ensure it opens with Google Maps

        // Check if there's an app available to handle this intent
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivity(intent);
        } else {
            Toast.make.Text(CustDetail.this, "Google Maps is not installed", Toast.LENGTH_SHORT).show();
        }
    }
});

```

Listing 3.12: Potongan kode *intent* Google Map

E.6 Order

Bagian Order pada aplikasi ini dibagi menjadi dua komponen utama, yaitu *Header* dan *Detail* (Barang). Pada bagian *Header*, informasi yang ditampilkan mencakup nomor *order*, tanggal *order*, kode *customer*, nama *customer*, nomor PO, cara bayar, kode kirim, dan catatan. Dari informasi tersebut, *sales* hanya perlu mengisi kolom cara bayar dan kode kirim sebagai isian wajib, sementara kolom nomor PO dan catatan bersifat opsional. Data lainnya seperti nomor *order*, kode

customer, dan nama *customer* sudah otomatis diisi oleh sistem. Sebelum *sales* memilih cara bayar, sistem secara otomatis akan menampilkan metode pembayaran yang biasa digunakan oleh *customer*. Namun, jika diperlukan, *sales* dapat mengubah cara bayar dengan ketentuan perubahan hanya dapat dilakukan ke arah yang lebih konservatif. Contohnya, jika *customer* memiliki metode pembayaran kredit 30 hari, mereka tidak dapat menggantinya menjadi kredit 60 hari, tetapi dapat mengubahnya menjadi *cash*. Untuk pemilihan cara bayar dan kode kirim, digunakan *AlertDialog* dengan *ListView* sebagai komponen untuk menampilkan daftar pilihan yang tersedia. Hal ini dirancang untuk memudahkan pengguna dalam memilih opsi yang sesuai secara intuitif dan cepat.

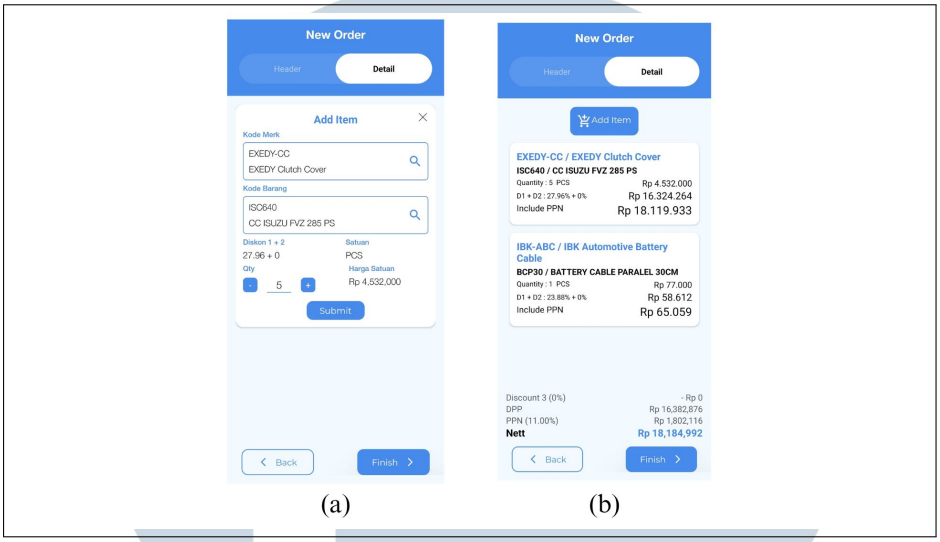


Gambar 3.4. Halaman Order Header

Pada bagian Barang atau *Detail*, *sales* dapat menambahkan barang yang ingin dipesan dengan langkah-langkah yang sistematis. Pertama, *sales* diharuskan memilih *merk* sebelum memilih barang. Pemilihan *merk* dan barang menggunakan metode yang sama, yaitu dengan menampilkan daftar melalui *ListView* yang dilengkapi dengan *SearchView*. Fitur *SearchView* ini bertujuan untuk mempermudah sales menemukan barang secara spesifik. Data untuk *merk* dan barang diperoleh dari respons *webservice* yang kemudian diproses oleh *adapter* dan *model* untuk ditampilkan dalam *ListView*.

Setelah *sales* memilih *merk* dan barang, informasi seperti diskon, satuan, dan harga barang akan otomatis ditampilkan. Langkah berikutnya, *sales* memasukkan *quantity* barang yang akan dipesan. Setelah semua data barang berhasil diisi, *sales* dapat menekan tombol *Submit* untuk menambahkan barang ke

dalam daftar pesanan. Data barang berikut perhitungan harganya akan disimpan secara lokal menggunakan SQLite sebagai penyimpanan sementara.



Gambar 3.5. Halaman Order Detail

Setelah proses input barang selesai, sales dapat menyelesaikan transaksi dengan menekan tombol *Finish*, yang akan membuat data kedalam format *JSON* dan akan dikirim ke sistem pusat melalui *webservice*. Proses ini memastikan data pesanan tersimpan secara aman dan terintegrasi ke sistem pusat. Proses pengiriman data terdapat pada Potongan Kode 3.13

```
1  private void paramDataJson(final String param_orderjson,
       final String param_orderdetailjson, final String
       textfilter_param) throws JSONException {
2       JSONObject orderJson = new JSONObject(param_orderjson);
3       orderJson.put("Discount", discount3ValueCleaned);
4       JSONArray detailJsonArray = new
       JSONArray(param_orderdetailjson);
5       orderJson.put("Detail", detailJsonArray);
6       Log.d(TAG, "Data Json : " + orderJson.toString());
7       System.out.println("Data Json : " +
       orderJson.toString());
8       String tag_string_req = "req_datajson";
9       showDialog("Loading, please wait...");
10      StringRequest strReq = new
       StringRequest(Request.Method.POST,
11              AppConfig.URL_INSERT_ORDER, new
       ResponseListener<String>() {
```

```
12          @Override
13          public void onResponse(String response) {
14              Log.d("Response", response);
15              hideDialog();
16              try {
17                  JSONObject jsonObject = new
    JSONObject(response);
18                  String status =
    jsonObject.getString("Status");
19                  if ("0".equals(status)) {
20                      Toast.make.Text(getContext(), "Success
    placing ordering data", Toast.LENGTH_LONG).show();
21                      //databaseHelper.deleteAllRecords();
22                      //Intent intent = new
    Intent(getContext(), Order.class);
23
    //intent.putExtra("kodeCust",kode_cust_h);
24                      //intent.putExtra("tipe",tipe);
25                      ((Ordering)
    getContext()).setResultAndFinishOK(
    event,kodecust,nodocument);
26                      AppNotificationManager
    notificationManager = new
    AppNotificationManager(getContext());
27
    notificationManager.showNotificationSuccessOrder(nama_cust_h);
28                  } else {
29                      String message =
    jsonObject.getString("msg");
30                      showWarning(message);
31                      Toast.make.Text(getContext(), "Error: "
    + message, Toast.LENGTH_LONG).show();
32                      Log.d(TAG, message);
33                  }
34              } catch (JSONException e) {
35                  e.printStackTrace();
36                  Toast.make.Text(getContext(), "Json error:
    " + e.getMessage(), Toast.LENGTH_LONG).show();
```

```
                showWarning(e.getMessage());
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e("Data Error", error.getMessage());
            hideDialog();
            Toast.make.Text(getContext(),
   getString(R.string.network_error_message),
   Toast.LENGTH_LONG).show();
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            // Posting parameters to the web service
            Map<String, String> params = new HashMap<>();
            params.put("prefixdb", prefix);
            params.put("textfilter", textfilter_param);
            params.put("bulan_rct", "10");
            params.put("tahun_rct", "2024");
            params.put("jsondata", orderJson.toString());
            return params;
        }
    };
    AppController.getInstance().addToRequestQueue(strReq,
   tag_string_req);
}

```

Listing 3.13: Potongan kode order

E.7 Daftar Faktur

Fitur Daftar Faktur dibuat untuk membantu sales memantau status pembayaran *customer* yang masih memiliki hutang. Melalui fitur ini *sales* dapat melihat jumlah tagihan yang harus dibayarkan oleh *customer* serta mengetahui jumlah hari hingga jatuh tempo berdasarkan nomor faktur. Informasi ini mempermudah *sales* dalam merencanakan kunjungan dan memastikan *customer*

membayar tepat waktu. Data pada halaman ini ditampilkan dengan metode *ListView* dan dilengkapi dengan metode *search* untuk mempermudah *sales* melakukan pencarian berdasarkan nama *customer* dan kode *customer*. Potongan Kode 3.14 menunjukan pengamblan data bedasarkan *response webservice*.

```
1  private void loadFakturData(final List<FakturModel>
   fakturList, final FakturAdapter adapter, final String
   prefix, final String usersfa) {
2      String tag_string_req = "req_faktur";
3      showDialog("Loading, please wait...");
4      StringRequest strReq = new
   StringRequest(Request.Method.POST, AppConfig.URL_AR, new
   ResponseListener<String>() {
5          @Override
6          public void onResponse(String response) {
7              Log.d("Response", response);
8              hideDialog();
9              try {
10                 JSONObject jsonObject = new
   JSONObject(response);
11                 int sukses = jsonObject.getInt("sukses");
12                 if (sukses == 1) {
13                     String totalValue =
   jsonObject.getString("Total");
14                     TotalAR.setText("Total AR: " +
   formatCurrency(totalValue));
15                     JSONArray dataArray =
   jsonObject.getJSONArray("Data");
16                     List<FakturModel> newFakturList = new
   ArrayList<>();
17                     for (int i = 0; i < dataArray.length();
   i++) {
18                         JSONObject dataObject =
   dataArray.getJSONObject(i);
19                         String kodeCust =
   dataObject.getString("KodeCust");
20                         String customerName =
   dataObject.getString("NamaCust");
```

```
21                          String arValue =
    dataObject.getString("TotalAR");
22                          int daysCount =
    dataObject.getInt("OverDue");
23                          FakturModel faktur = new
    FakturModel(kodeCust, customerName, arValue, daysCount);
24                          newFakturList.add(faktur);
25                      }
26                      adapter.updateList(newFakturList);
27                  } else {
28                      String message =
    jsonObject.getString("msg");
29
    Toast.make.Text(getApplicationContext(), "Error: " +
    message, Toast.LENGTH_LONG).show();
30                  }
31              } catch (JSONException e) {
32                  e.printStackTrace();
33                  Toast.make.Text(getApplicationContext(),
    "Json error: " + e.getMessage(),
    Toast.LENGTH_LONG).show();
34              }
35          }
36      }, new Response.ErrorListener() {
37          @Override
38          public void onErrorResponse(VolleyError error) {
39              Log.e("Data Error", error.getMessage());
40              hideDialog();
41              Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
    Toast.LENGTH_LONG).show();
42          }
43      }) {
44          @Override
45          protected Map<String, String> getParams() {
46              Map<String, String> params = new HashMap<>();
47              params.put("prefixdb", prefix);
48              params.put("kodesales", usersfa);
```

```
            return params;
        }
    };
    AppController.getInstance().addToRequestQueue(strReq,
   tag_string_req);
}

```

Listing 3.14: Potongan kode daftar faktur

F. Fragment More

F.1 Change Password

Pada aplikasi SFA, ditambahkan fitur Change Password yang memungkinkan user untuk mengganti kata sandinya dengan lebih aman. Sebelum mengirim data ke web service untuk memperbarui kata sandi, fitur ini melakukan beberapa validasi. Validasi mencakup pemeriksaan apakah semua kolom telah diisi, memastikan panjang kata sandi baru minimal enam karakter, verifikasi kecocokan antara kata sandi baru dan sandi konfirmasi, serta memastikan bahwa tidak ada simbol yang dilarang seperti tanda kurung, tanda petik, atau simbol lain yang dapat mengganggu proses. Jika semua validasi terpenuhi, aplikasi akan mengirimkan permintaan POST ke web service dengan parameter yang mencakup nama user, kata sandi lama, dan kata sandi baru. Setelah berhasil diperbarui, user akan diminta untuk log out sebagai langkah keamanan tambahan.

```
private void validateData() {
    String inp_prevPass =
   prevPassword.getText().toString().trim();
    String inp_newPass =
   newPassword.getText().toString().trim();
    String inp_CNewPass =
   confirmNewPassword.getText().toString().trim();
    if (inp_prevPass.isEmpty()) {
        prevPassword.setError("Enter current password");
        prevPassword.requestFocus();
    } else if (inp_newPass.isEmpty()) {
        newPassword.setError("Enter new password!");
        newPassword.requestFocus();
    } else if (inp_CNewPass.isEmpty()) {
```

```
12          confirmNewPassword.setError("Enter confirm
    password!");
13          confirmNewPassword.requestFocus();
14      } else if (inp_newPass.length() < 6) {
15          newPassword.setError("Password must be at least 6
    characters long");
16          newPassword.requestFocus();
17      } else if (!inp_newPass.equals(inp_CNewPass)) {
18          confirmNewPassword.setError("Passwords don't
    match");
19          confirmNewPassword.requestFocus();
20      } else if (containsForbiddenSymbols(inp_prevPass)) {
21          prevPassword.setError("Simbol ini tidak bisa
    digunakan");
22          prevPassword.requestFocus();
23      } else if (containsForbiddenSymbols(inp_newPass)) {
24          newPassword.setError("Simbol ini tidak bisa
    digunakan");
25          newPassword.requestFocus();
26      } else if (containsForbiddenSymbols(inp_CNewPass)) {
27          confirmNewPassword.setError("Simbol ini tidak bisa
    digunakan");
28          confirmNewPassword.requestFocus();
29      }
30      else {
31          updatePassword(inp_newPass);
32      }
33  }
34  private boolean containsForbiddenSymbols(String input) {
35      String[] forbiddenSymbols = {",",".", "[", "]", "{",
    "}", "'", "`", "\\", "\"", "=", "+"};
36      for (String symbol : forbiddenSymbols) {
37          if (input.contains(symbol)) {
38              showDialog("True");
39              hideDialog();
40              return true;
41          }
42      }
```

```
43      return false;
44  }
45  private void updatePassword(final String newPassword_param)
    {
46      String tag_string_req = "req_change_password";
47      showDialog("Updating Password...");
48      StringRequest strReq = new
    StringRequest(Request.Method.POST,
    AppConfig.URL_CHANGE_PASSWORD, new
    ResponseListener<String>() {
49          @Override
50          public void onResponse(String response) {
51              Log.d("Response", response);
52              hideDialog();
53              try {
54                  JSONObject jsonObject = new
    JSONObject(response);
55                  int status = jsonObject.getInt("Status");
56                  if (status == 0) {
57
    Toast.make.Text(getApplicationContext(), "Change
    Password Success", Toast.LENGTH_LONG).show();
58                      SessionManagement sessionManagement =
    new SessionManagement(getApplicationContext());
59                      sessionManagement.LogOut();
60                  } else {
61                      String message =
    jsonObject.getString("msg");
62
    Toast.make.Text(getApplicationContext(), "Error: " +
    message, Toast.LENGTH_LONG).show();
63                  }
64              } catch (JSONException e) {
65                  e.printStackTrace();
66                  Toast.make.Text(getApplicationContext(),
    "Json error: " + e.getMessage(),
    Toast.LENGTH_LONG).show();
67              }
```

```
68          }
69      }, new Response.ErrorListener() {
70          @Override
71          public void onErrorResponse(VolleyError error) {
72              Log.e("Data Error", error.getMessage());
73              hideDialog();
74              Toast.make.Text(getApplicationContext(),
    getString(R.string.network_error_message),
    Toast.LENGTH_LONG).show();
75          }
76      }) {
77          @Override
78          protected Map<String, String> getParams() {
79              Map<String, String> params = new HashMap<>();
80              params.put("username", usernameSession);
81              params.put("password_lama",
    prevPasswordSession);
82              params.put("password_baru", newPassword_param);
83              return params;
84          }
85      };
86      strReq.setRetryPolicy(new DefaultRetryPolicy(
87              AppConfig.TIME_OUT_LOADING,
88              DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
89              DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
90      AppController.getInstance().addToRequestQueue(strReq,
    tag_string_req);
91  }
92
```

Listing 3.15: Potongan kode change password

3.4 Kendala dan Solusi yang Ditemukan

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

1. Keterlibatan tim *sales* sebagai pengguna akhir yang kurang dalam tahap pengembangan aplikasi, sehingga fitur yang dibuat terkadang kurang sesuai

dengan kebutuhan *user*.

2. Fitur pelacakan lokasi dan sistem *order* berbasis *real-time*, memerlukan koneksi dengan *webservice* yang kompleks. Hal ini menyebabkan kendala teknis, seperti error dalam komunikasi antara aplikasi dan server.

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

1. Diskusi dengan tim *sales* dilakukan sejak awal proyek untuk memahami kebutuhan mereka dan memastikan fitur yang dikembangkan relevan dengan kebutuhan. Pendekatan ini juga meningkatkan keterlibatan tim *sales* dalam proses pengembangan, sehingga fitur yang dibuat lebih sesuai dengan kebutuhan pengguna akhir.

2. Masalah teknis diselesaikan melalui diskusi dengan tim IT, khususnya dalam mengidentifikasi dan memperbaiki *error* pada koneksi *webservice*. Penggunaan alat debugging dan logging juga digunakan untuk mempermudah identifikasi masalah.