PEMANFAATAN LIBRARY TESSERACT PADA APLIKASI APPKUNTAN BERBASIS ANDROID

Lugy Septian Catur Firmansyah

Universitas Komputer Indonesia
Jalan Dipatiukur No. 112-116 Kota Bandung, Jawa Barat 40132
E-mail : lugyscf@gmail.com

ABSTRAK

Teknologi yang sedang berkembang pesat saat ini, banyak orang membutuhkan teknologi yang dapat memenuhi kebutuhan yang tepat. Dari yang sebelumnya teknologi berawal dari *PC* atau *computer* hingga *smartphone* yang sedang berkembang pesat saat ini. Seperti pengelola keuangan pribadi dengan menggunakkan *library tesseract* yang terdapat pada *smartphone android*.

Library Tesseract merupakan mesin pengenal karakter optik yang ada pada *android*. Yang memungkinkan dapat mempermudah pengguna dalam mengatur pengeluaran keuangan belanja pribadi, dan juga dapat membantu memberikan pengingat untuk masalah pengeluaran keuangan belanja pribadi dengan cara mengambil gambar struk belanja menggunakan kamera *smartphone android* yang kemudian data belanja di *scan* dengan menggunakan *tesseract* yang terdapat pada *smartphone android*, sehingga data belanja yang telah di *scan* akan diproses oleh *tesseract* lalu data gambar yang telah diolah akan ditampilkan ke layar sebagai data keluaran berupa teks, yang kemudian data teks tersebut dapat di simpan dalam *database* dan akan memberikan notifikasi apabila pengeluaran keuangan melebihi batas maksimum yang telah ditentukan sebelumnya.

Kata Kunci : *Library Tesseract*, Kelola Keuangan, *Android*.

1. PENDAHULUAN

1.1 Latar Belakang

Pengelolaan keuangan dalam kehidupan sehari-hari merupakan salah satu kunci utama menjalani setiap kebutuhan hidup, dimulai dari anak sekolah, mahasiswa, pekerja, apalagi bagi setiap orang yang sudah berkeluarga, semuanya diharuskan mengelola keuangan mereka dengan baik. Bila pemakaian keuangan tidak terkontrol maka akan berakibat pemborosan yang sia-sia, yang pasti sangat merugikan diri sendiri.

Kenyataannya dalam hal keuangan, masih banyak diantara kita yang tidak dapat mengontrolnya, karena kebanyakan orang tanpa sadar akan membeli sesuatu yang tidak terlalu dibutuhkan hanya karena sekedar "menginginkan", padahal barang yang dibeli itu cukup mahal harganya. Sehingga kita pasti pernah berfikir "kemana perginya uang kita?" itulah yang akan mengakibatkan orang lupa terhadap kebutuhannya, dan pada akhirnya kita akan menyesalinya setelah sadar.

Berdasarkan data yang diperoleh dari kuesioner yang telah disebarkan melalui media sosial, ternyata dari 86 tanggapan, yang mempunyai *smartphone android* adalah 89,5% orang atau 9 orang yang tidak mempunyai *smartphone android*. Dan terbukti dari 77 tanggapan dari pertanyaan "Apakah anda pernah kebingungan kemana perginya uang anda?" 59,7% menjawab kadang-kadang, 31,2% menjawab sering, dan 9,1% menjawab tidak pernah. Yang berarti kebanyakan orang yang mengisi kuesioner tidak dapat mengontrol keuangannya. Itu dapat dibuktikan dari pertanyaan "Apakah pengeluaran keuangan anda sering tidak terkendali?" dan dari 77 tanggapan 55,8% menjawab kadang-kadang, 41,6% menjawab sering, dan 2,6% menjawab tidak pernah.

Kenapa setiap orang tidak mencatat pengeluaran keuangannya sehingga dapat membantu untuk mengelola keuangan pribadinya? Itu disebabkan karena dalam mencatat pengeluaran keuangan sedikit sulit dilakukan dan juga kebanyakan orang malas untuk mencatat pengeluarannya. Selain itu orang kesulitan dalam memasukkan data pengeluaran keuangannya, sehingga jika dalam memasukkan

datanya diambil melalui gambar maka dapat dipermudah dengan menggunakan *library tesseract*. Bagaimana cara memasukan data pengeluaran keuangan yang diambil melalui gambar? Yaitu dengan mengambil gambar struk belanja menggunakan kamera *smartphone android* yang kemudian data belanja di *scan* dengan menggunakan *tesseract* yang terdapat pada *smartphone android*.

Sehingga penulis memberi saran untuk membuat aplikasi pembantu yang dapat mengingatkan pengeluaran keuangan pengguna. Aplikasi ini dapat mengambil teks dari sebuah gambar seperti gambar struk belanja, dengan menggunakan *library tesseract* yaitu mesin pengenal karakter optik yang ada pada *android*. Jika data teks sudah didapat maka data bisa diolah, seperti dapat memberikan pemberitahuan pengeluaran keuangan pribadi. Aplikasi akan membaca karakter yang sebelumnya sebuah struk

belanja yang di *foto* dan di *scan*, lalu berubah menjadi data teks sehingga aplikasi akan memproses data untuk memberikan notifikasi pengeluaran belanja kita pada setiap minggu atau setiap bulan tergantung pengaturannya. Cara menggunakan aplikasi ini, kita cukup memfoto struk belanja untuk mendapatkan pengingat pengeluaran keuangan dikemudian waktu, atau jika kita belanja di tempat yang tidak memberikan struk kita dapat memasukkan data barang yang dibeli secara manual dan memasukkan total belanja yang dikeluarkan secara manual juga.

Dan pembuatan aplikasi ini berdasarkan pada 77 tanggapan dari kuesioner yang peneliti tanyakan yaitu "Apakah anda setuju dengan dibuatnya aplikasi *smartphone android* yang dapat membantu mengingatkan masalah keuangan anda?" 42,9% menjawab sangat setuju, 41,6% menjawab setuju, 11,7% menjawab netral, dan 3,9% menjawab sangat tidak setuju.

Dengan ini penulis mengambil judul skripsi "Pemanfaatan Library Tesseract Pada Aplikasi Appkuntan Berbasis Android".

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dikemukakan diatas maka tedapat beberapa masalah, yaitu :

a. Bagaimana cara membuat aplikasi Appkuntan yang dapat memberikan suatu pengingat untuk mengatur keuangan pengguna dan juga memberikan suatu kemudahan dalam mencatat pengeluaran keuangan.
b. Bagaimana cara memanfaatkan *Library Tesseract* pada aplikasi Appkuntan sehingga dapat mempermudah setiap pengguna dalam mengolah data struk belanja.

1.3 Maksud dan Tujuan

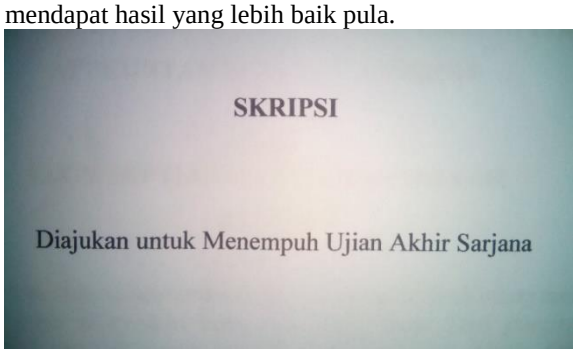
Berdasarkan permasalahan yang diteliti maka maksud dari penelitian ini adalah untuk menghasilkan sebuah aplikasi appkuntan yang membantu keuangan diri sendiri berbasis *android*.

Sedangkan tujuan yang akan dicapai pada penelitian ini adalah aplikasi appkuntan dapat mempermudah pengguna untuk mencatat pengeluaran keuangan dengan cara memindai struk belanja menggunakan *tesseract*, sehingga data teks yang didapat oleh aplikasi appkuntan dapat di olah untuk mengingatkan pengeluaran keuangan belanja pengguna.

1.4 Teori Memecah Karakter Gambar Menggunakan Tesseract

Teori memecahkan karakter gambar menggunakan *tesseract* pada *smartphone android*. Dalam memecahkan karakter gambar adalah dengan membuat *algoritma* yang baik sehingga *smartphone android* dapat membaca atau mengenali setiap karakter yang ada pada gambar seperti manusia yang melakukannya.

Karakter gambar perlu dipilih terlebih dahulu dengan melihat apakah kertasnya dalam keadaan baik, warna kertasnya putih dengan warna tinta yang hitam dan jelas, tidak ada huruf yang saling menimpa, kertas tidak terlipat-lipat atau rusak. Dengan demikian, *tesseract* dapat mengenali teks pada gambar tersebut dengan baik sehingga mendapat hasil yang lebih baik pula.



Gambar 1 Contoh Karakter Gambar

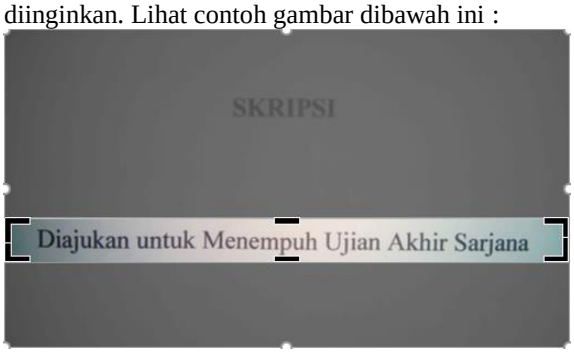
Berikut adalah langkah-langkah memecahkan data struk menggunakan *tesseract* pada *smartphone android* :

1. Ambil Gambar

Ambil gambar dengan menggunakan kamera *smartphone android* atau dengan cara mengambil gambar dari galeri.

2. Pemotongan Gambar (*Cropping*)

Setelah gambar diambil atau dipilih maka masuk ke proses pemotongan gambar. Pemotongan gambar dapat dilakukan dengan cara menggeserkan setiap titik potong, sehingga didapat data yang telah diinginkan. Lihat contoh gambar dibawah ini :



Gambar 2 Contoh Pemotongan Karakter Gambar

3. Proses Pengenalan Karakter

Setelah hasil pemotongan disimpan maka sistem *smartphone android* akan melakukan proses pengenalan karakter menggunakan *tesseract*. Dimana setelah karakter gambar berhasil dibaca atau dikenali maka sistem akan menampilkan hasil data karakter gambar. *Coding tesseract* untuk proses pengenalan karakter pada gambar hingga menampilkan hasil data yang telah dibaca atau dikenali, yaitu :

```
private void prepareTesseract() {
    try {
        prepareDirectory(DATA_PATH +
TESSDATA);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        copyTessDataFiles(TESSDATA);
    }

private void copyTessDataFiles(String path) {
        try {
            String fileList[] = getApplicationContext().getAssets().list(path);

            for (String fileName : fileList) {

                // open file within the assets folder
                // if it is not already there copy it to the sdcard
                String pathToDataFile = DATA_PATH + path + "/" + fileName;
                if (!(new File(pathToDataFile)).exists()) {

                    InputStream in = getApplicationContext().getAssets().open
                    (path + "/" + fileName);

                    OutputStream out = new FileOutputStream(pathToDataFile);

                    // Transfer bytes from in to out
                    byte[] buf = new byte[1024];
                    int len;

                    while ((len = in.read(buf)) > 0) {
                        out.write(buf, 0, len);
                    }
                    in.close();
                    out.close();

                    Log.d(TAG, "Copied " + fileName + "to tessdata");
                }
            }
        } catch (IOException e) {
            Log.e(TAG, "Unable to copy files to tessdata " + e.toString());
        }
    }

private void startOCR() {
        try {
            BitmapFactory.Options options = new BitmapFactory.Options();
            options.inSampleSize = 4; // 1 - means max size. 4 - means maxsize/4 size. Don't use value <4, because you need more memory in the heap to store your data.
            //Bitmap bitmap = BitmapFactory.decodeFile(imgUri.get
```

```
                Path(), options);

            result = extractText(thePic);
            textView.setText(result);
            parsingText();

        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
        }
    }

private String extractText(Bitmap bitmap) {
        try {
            tessBaseApi = new TessBaseAPI();
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
            if (tessBaseApi == null) {
                Log.e(TAG, "TessBaseAPI is null. TessFactory not returning tess object.");
            }
        }

        tessBaseApi.init(DATA_PATH, lang);

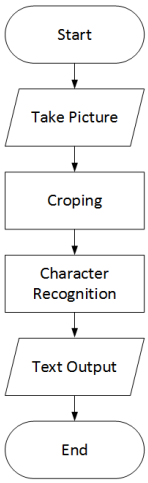
//      //EXTRA SETTINGS
//        //For example if we only want to detect numbers
//
            tessBaseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST,
            "1234567890");
//
//       //blackList Example

        //tessBaseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST,
        "!@#$%^&*()_       +=-,./<>?");

        Log.d(TAG, "Training file loaded");
        tessBaseApi.setImage(bitmap);
        String extractedText = "empty result";
        try {
            extractedText = tessBaseApi.getUTF8Text();
        } catch (Exception e) {
            Log.e(TAG, "Error in recognizing text.");
        }
        tessBaseApi.end();
        return extractedText;
    }
```

1.5 Analisis Metode

Berikut adalah tahapan-tahapan beserta metode yang digunakan dalam aplikasi yang akan dibangun, mulai dari masukan hingga keluaran.



Gambar 3 Tahapan Pada Aplikasi

1. *Take Picture*

Tahap ini merupakan tahap awal yang digunakan untuk mengambil gambar melalui kamera yang terpasang pada perangkat *android*, sebagai masukkan yang akan diproses oleh tahap berikutnya.

2. *Cropping*

Tahap ini berfungsi untuk memotong gambar yang telah diambil oleh pengguna. Tahap *cropping* perlu dilakukan untuk memfokuskan gambar pada area teks saja agar tidak ada objek selain teks yang akan dikenali oleh *OCR*. Pengguna menentukan sendiri area teks pada gambar untuk dipotong, agar ukuran gambar lebih kecil sehingga mempercepat proses penengenalan karakter. Proses pemotongan gambar menggunakan fungsi yang sudah ada pada perangkat *android*.

3. *Character Recognition*

Proses pengenalan karakter menggunakan layanan yang sudah ada, yaitu *Tesseract*.

4. *Text Output*

Hasil pengenalan karakter oleh *OCR* akan dikirim kembali ke aplikasi berupa *file* teks yang selanjutnya akan disimpan ke dalam *database*.

1.6 Analisis Proses Parsing Data Struk

Untuk dapat menghasilkan data yang sesuai format struk belanja maka diperlukan proses *parsing* yaitu penguraian setiap kata atau angka pada struk belanja. *Coding* untuk proses penguraian karakter pada gambar struk hingga menampilkan hasil data yang telah dibaca atau dikenali, yaitu :

```
private void parsingText() {
      StringTokenizer tokens = new StringTokenizer (textView.getText().toString(), "\r\n");
      if (tokens.countTokens() == 1 ){
         String satu = tokens.nextToken();
         String harga1 = satu.substring(satu.length() - 5);
         String hargasat1 = satu.substring(satu.length() - 10);
         String hargasat1_1 = hargasat1.substring(0, 4);
         String qty = satu.substring(satu.length() - 12);
         String qty_1 = qty.substring(0, 1);

         satu = satu.replaceAll("[\\d.,]","");
         harga1 = harga1.replaceAll("[.,]","");

         aq.id(R.id.item1).text(satu);

aq.id(R.id.item2).visibility(View.GONE);

aq.id(R.id.item3).visibility(View.GONE);

aq.id(R.id.item4).visibility(View.GONE);

aq.id(R.id.item5).visibility(View.GONE);

aq.id(R.id.containerHasil).visibility(View.VISIBLE);

aq.id(R.id.textResult).visibility(View.GONE);

         aq.id(R.id.qty1).text(qty_1);
         aq.id(R.id.qty2).visibility(View.GONE);
         aq.id(R.id.qty3).visibility(View.GONE);
         aq.id(R.id.qty4).visibility(View.GONE);
         aq.id(R.id.qty5).visibility(View.GONE);

aq.id(R.id.containerQty).visibility(View.VISIBLE);

aq.id(R.id.textResultqty).visibility(View.GONE);

         aq.id(R.id.hrgitem1).text(harga1);

aq.id(R.id.hrgitem2).visibility(View.GONE);

aq.id(R.id.hrgitem3).visibility(View.GONE);

aq.id(R.id.hrgitem4).visibility(View.GONE);

aq.id(R.id.hrgitem5).visibility(View.GONE);

aq.id(R.id.containerHasilHrg).visibility(View.VISIBLE);

aq.id(R.id.textResultHrg).visibility(View.GONE);

         aq.id(R.id.hrg2item1).text(hargasat1_1);

aq.id(R.id.hrg2item2).visibility(View.GONE);

aq.id(R.id.hrg2item3).visibility(View.GONE);

aq.id(R.id.hrg2item4).visibility(View.GONE);

aq.id(R.id.hrg2item5).visibility(View.GONE);

aq.id(R.id.containerHasilHrg2).visibility(View.
```

```
VISIBLE);

aq.id(R.id.textResultHrg2).visibility(View.GONE);
```

1.7 Analisis Kebutuhan Fungsional

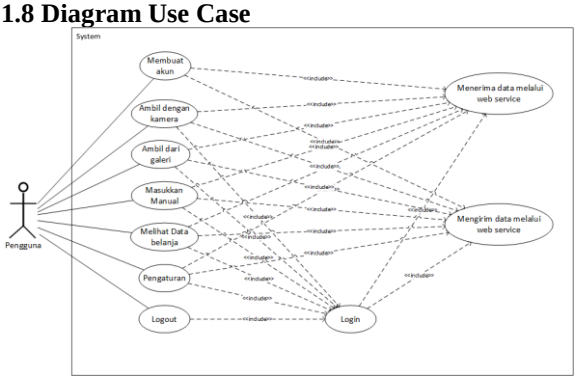
Tabel 1 Spesifikasi Perangkat Lunak Fungsional di Platform Mobile

Spesifikasi Kebutuhan Perangkat Lunak Fungsional di Platform Mobile	
Nama	**Keterangan**
SKPL-F-001	Sistem menyediakan fasilitas login
SKPL-F-002	Sistem menyediakan fasilitas ambil dengan kamera
SKPL-F-003	Sistem menyediakan fasilitas ambil dari galeri
SKPL-F-004	Sistem menyediakan fasilitas pengaturan
SKPL-F-005	Sistem menyediakan fasilitas masukkan manual
SKPL-F-006	Sistem menyediakan fasilitas data belanja
SKPL-F-007	Sistem menyediakan fasilitas mengirim data melaui web service
SKPL-F-008	Sistem menyediakan fasilitas menerima data melaui web service
SKPL-F-009	Sistem menyediakan fasilitas membuat akun
SKPL-F-010	Sistem menyediakan fasilitas logout

Tabel 2 Spesifikasi Perangkat Lunak Non Fungsional di Platform Mobile

Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional di Platform Mobile	
Nama	**Keterangan**
SKPL-NF-001	Sistem dapat diakses dimana saja
SKPL-NF-002	Sistem berbasis android
SKPL-NF-003	Sistem dapat dijalankan dengan versi minimum android 4.4.2 (Kit Kat)
SKPL-NF-004	Sistem dapat dijalankan optimal di smartphone yang memiliki kamera diatas 8 MP

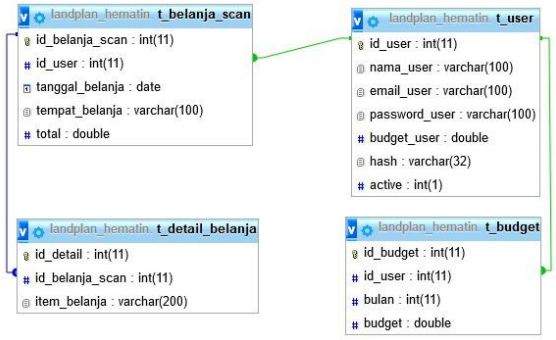
1.8 Diagram Use Case



Gambar 4 Diagram Use Case

1.9 Diagram Relasi

Diagram relasi merupakan rangkaian hubungan antara dua tabel atau lebih pada sistem basis data. Gambar Diagram Relasi merupakan gambaran rangkaian basis data pada pembangunan Appkuntan.



Gambar 5 Diagram Relasi

1.10 Struktur Tabel

Struktur tabel menggambarkan detail yang berisi field, tipe data, pangang data, dan keterangan lainnya. Tabel yang terdapat pada basis data yaitu admin, detail_pemesanan, jenis_makanan, menu, karyawan, dan Pemesanan.

Tabel 3 Struktur Tabel t_user

Nama Field	Tipe Data	Panjang	Null	Ekstra	Kunci
id_user	Int	11	Tidak	Auto increment	Primary key
nama_user	Varchar	100	Tidak		
email_user	Varchar	100	Tidak		
password	Varchar	100	Tidak		

Tabel 4 Struktur Tabel t_budget

Nama Field	Tipe Data	Panjang	Null	Ekstra	Kunci
id_budget	Int	11	Tidak	Auto increment	Primary key
id_user	Int	11	Tidak		Foreign key

Nama Field	Tipe Data	Panjang	Null	Ekstra	Kunci
					references t_user (id_user)
bulan	Int	11	Tidak		
budget	double		Tidak		

Tabel 5 Struktur Tabel t_belanja_scan

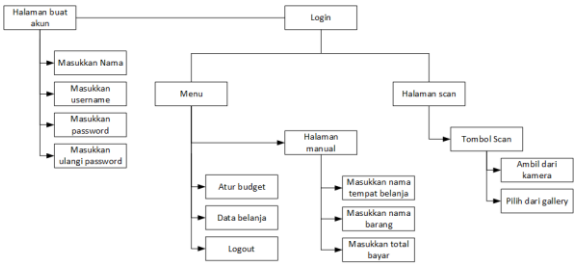
Nama Field	Tipe Data	Panjang	Null	Ekstra	Kunci
id_belanja_scan	Int	11	Tidak	Auto increment	Primary key
id_user	Int	11	Tidak		Foreign key references t_user (id_user)
tanggal_belanja	date		Tidak		
tempat_belanja	varchar	100	Tidak		

Tabel 6 Struktur Tabel t_detail_belanja

Nama Field	Tipe Data	Panjang	Null	Ekstra	Kunci
id_detail	Int	11	Tidak	Auto increment	Primary key
id_belanja_scan	Int	11	Tidak		Foreign key references t_belanja_scan (id_belanja_scan)
item_belanja	varchar	200	Tidak		
total_harga	Int	11	Tidak		
jumlah	Int	11	Tidak		
harga	int	11	tidak		

1.11 Struktur Menu Platform Android

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi sehingga aplikasi yang mudah dipahami dan mudah digunakan.



Gambar 6 Struktur Menu Platform Mobile

1.12 Implementasi Perangkat Keras

Perangkat keras (*hardware*) yang digunakan untuk membangun sistem di *platform mobile* untuk aplikasi Appkuntan sebagai asisten keuangan pribadi dapat dilihat pada tabel berikut.

Tabel 7 Implementasi Minimum Perangkat Keras di *Platform Mobile*

Nama Perangkat Keras	**Spesifikasi**
Proccessor	Quad-core 1.3 GHz Cortex-A7
Memory Internal	8 GB
Kamera belakang	8 MP
Layar	4.7 inches
Sistem Operasi	Android OS, v4.2 (Jelly Bean)

1.13 Implementasi Basis Data

Pembuatan basis data dilakukan dengan menggunakan aplikasi DBMS MySQL.

Tabel 8 Implementasi Basis Data

No	Nama Tabel	Perintah SQL
1	t_belanja_scan	CREATE TABLE IF NOT EXISTS `t_belanja_scan` (`id_belanja_scan` int(11) NOT NULL, `id_user` int(11) NOT NULL, `tanggal_belanja` date NOT NULL, `tempat_belanja` varchar(100) NOT NULL, `total` double NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1; ALTER TABLE `t_belanja_scan` ADD PRIMARY KEY (`id_belanja_scan`), ADD KEY `t_belanja_scan_ibfk_1` (`id_user`); ALTER TABLE `t_belanja_scan` ADD CONSTRAINT `t_belanja_scan_ibfk_1` FOREIGN KEY (`id_user`) REFERENCES `t_user`

		(`id_user`) ON DELETE CASCADE ON UPDATE CASCADE;
2	t_budget	CREATE TABLE IF NOT EXISTS `t_budget` (`id_budget` int(11) NOT NULL, `id_user` int(11) NOT NULL, `bulan` int(11) NOT NULL, `budget` double NOT NULL) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1; ALTER TABLE `t_budget` ADD PRIMARY KEY (`id_budget`), ADD KEY `t_budget_ibfk_1` (`id_user`); ALTER TABLE `t_budget` MODIFY `id_budget` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=14; ALTER TABLE `t_budget` ADD CONSTRAINT `t_budget_ibfk_1` FOREIGN KEY (`id_user`) REFERENCES `t_user` (`id_user`) ON DELETE CASCADE ON UPDATE CASCADE;
3	t_detail_belanja	CREATE TABLE IF NOT EXISTS `t_detail_belanja` (`id_detail` int(11) NOT NULL, `id_belanja_scan` int(11) NOT NULL, `item_belanja` varchar(200) NOT NULL) ENGINE=InnoDB AUTO_INCREMENT=118 DEFAULT CHARSET=latin1; ALTER TABLE `t_detail_belanja` ADD PRIMARY KEY (`id_detail`), ADD KEY `t_detail_belanja_ibfk_1` (`id_belanja_scan`); ALTER TABLE `t_detail_belanja` MODIFY `id_detail` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=118; ALTER TABLE `t_detail_belanja` ADD CONSTRAINT `t_detail_belanja_ibfk_1` FOREIGN KEY (`id_belanja_scan`) REFERENCES `t_belanja_scan` (`id_belanja_scan`) ON DELETE
		CASCADE ON UPDATE CASCADE;
4	t_user	CREATE TABLE IF NOT EXISTS `t_user` (`id_user` int(11) NOT NULL, `nama_user` varchar(100) NOT NULL, `email_user` varchar(100) NOT NULL, `password_user` varchar(100) NOT NULL, `budget_user` double NOT NULL, `hash` varchar(32) NOT NULL, `active` int(1) NOT NULL DEFAULT '0') ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1; ALTER TABLE `t_user` ADD PRIMARY KEY (`id_user`); ALTER TABLE `t_user` MODIFY `id_user` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=20;

1.14 Implementasi Antarmuka

Implementasi antarmuka dilakukan terhadap setiap tampilan program yang dibangun dan pengkodeannya dalam bentuk file program.

Tabel 9 Implementasi Antar Muka di *Platform Mobile*

Menu	Deskripsi	Nama File
Login	Merupakan halaman untuk melakukan Login	activity_login.xml
Menu utama	Merupakan halaman awal ketika pengguna pertama login	activity_main.xml
Halaman Indomaret	Merupakan halaman yang menampilkan halaman scan untuk struk Indomaret	activity_indomaret.xml
Scan Struk	Merupakan halaman untuk memindai	activity_scan.xml

	struk	
Data Belanja	Merupakan halaman untuk melihat data belanja yang telah di *scan*.	activity_data_belanja_scan.xml
Pengaturan Budget	Merupakan halaman yang menampilkan pengaturan batas maksimal pengeluaran keuangan per bulan.	activity_budget.xml
Buat Akun	Merupakan halaman untuk membuat akun baru.	activity_daftar.xml

1.15 Kesimpulan

Berikut ini merupakan kesimpulan yang dapat diambil pada hasil akhir penelitian ini.

1. Aplikasi appkuntan pada platform android ini cukup dapat membantu pengguna dalam melakukan pencatatan keuangan.
2. Dan juga aplikasi appkuntan ini cukup dapat mengingatkan pengguna dalam jumlah pengeluaran belanjanya.

1.16 Saran

Perangkat lunak appkuntan yang telah dibangun sudah dapat menyelesaikan permasalahan yang dihadapi. Adapun saran-saran terhadap pengembangan perangkat lunak appkuntan ini adalah sebagai berikut :

1. Kendala pada kurangnya kemampuan *library tesseract* mengenali tulisan tangan yang cenderung tidak memiliki pola khusus.
2. Kesulitan mengenali tulisan struk yang huruf atau katanya saling menimpa.
3. Kesulitan mengenali tulisan struk yang huruf atau katanya tidak jelas.
4. Kesulitan mengenali tulisan struk jika kertas struk tidak berwarna putih.
5. Belum dapat membaca seluruh struk.

DAFTAR PUSTAKA

[1] Smith. Ray, "An Overview of Tesseract Engine," 2013.

[2] B. R. Sofian, "Perbandingan Algoritma Template Matching Dan Feature Extraction Pada Optical Character Recognition," vol. I, Bandung : s.n., 2012.

[3] K. Hamilton and R. Miles, Learning UML 2.0, O'Reilly, April 2006, p. 286.

[4] A. Solichin, MySQL 5 Dari Pemula Hingga Mahir, Jakarta: Achmatim.net, 2010.