BAB IV
HASIL DAN PEMBAHASAN

4.1 Pembuatan *Barcode*

Barcode yang digunakan adalah UPC-A, *barcode* ini dibuat dengan menggunakan aplikasi *Barcode generator*. Pada pembuatan *barcode,* pertama-tama penulis menentukan dulu kode UPC-A, kode perusahaan,dan kode produk.

Kode UPC-A	001
Kode Perusahaan	4121
KodeProduk	30101
Barcode	001412130101

Selanjutnya menentukan *Check* digit untuk mengecek *barcode* telah terbaca dengan benar atau salah. Untuk menentukan *check* digit menggunakan cara sebagai berikut; Misal menentukan check digit 001412130101.

001412130101 dikalikan 1313131313 menjadi 0011216190303 total 36. Kemudian dibagi 10 = sisa 6. Setelah itu 10 dikurangi sisa = 4. Maka jadi *barcode*, 001412130101-4.

Setelah nomor *barcode* ditentukan *check* digit, *barcode* ini dipindahkan ke aplikasi *barcode generator* untuk mencetak *barcode* berupabalok/garis-garishitam- putih.

4.2 Pembuatan *Database*

Database yang digunakan adalah *Database* SQLite yang tersedia di aplikasi Android Studio sebagai *database* lokal. Pada pembuatan *database*, terlebih dahulu membuat kelas *Data.java* yang dimana kelas ini berfungsi untuk mendeskripsikan objek seperti *barcode*, namabarang, posisi, tanggal dan jam. Kode kelas *Data.java* dapat dilihat pada pada gambar.4.1

```
1   package com.example.aset;
2
3   import java.io.Serializable;
4
5   public class Data implements Serializable {
6
7       private String barcode;
8       private String namaBrg;
9       private String jenisBrg;
10      private String merk;
11      private String tgl;
12      private String jumBrg;
13      private String hrgBeli;
14      private String posisi;
15
16      public String getBarcode() { return barcode; }
19      public void setBarcode(String barcode) { this.barcode = barcode; }
22
23      public String getNamaBrg() { return namaBrg; }
26      public void setNamaBrg(String namaBrg) { this.namaBrg = namaBrg; }
29
30      public String getJenisBrg() { return jenisBrg; }
33      public void setJenisBrg(String jenisBrg) { this.jenisBrg = jenisBrg; }
36
37      public String getMerk() { return merk; }
40      public void setMerk(String merk) { this.merk = merk; }
43
44      public String getTgl() { return tgl; }
47      public void setTgl(String tgl) { this.tgl = tgl; }
50
51      public String getJumBrg() { return jumBrg; }
52      public void setJumBrg(String jumBrg) { this.jumBrg = jumBrg; }
55
56      public String getHrgBeli() { return hrgBeli; }
59      public void setHrgBeli(String hrgBeli) { this.hrgBeli = hrgBeli; }
62
63      public String getPosisi() { return posisi; }
66      public void setPosisi(String posisi) { this.posisi = posisi; }
69  }
70
```

Gambar 4.1 Kode Kelas *Data.Java*

Penjelasan gambar 4.1 adalah sebagai berikut:

1. Kode ini menyatakan atribut data untuk method getter dan setter.

```
public class Data implements Serializable {

private String barcode;
private String namaBrg;
private String jenisBrg;
private String merk;
private String tgl;
private String jumBrg;
private String hrgBeli;
private String posisi;
```

2. *Method getter* dan *setter* diberikan modifier `public` karena *method* ini diakses diluar *class*.

```
public String getBarcode() {
return barcode;}
public void setBarcode(String barcode) {
this.barcode = barcode;}

public String getNamaBrg() {
return namaBrg;}
public void setNamaBrg(String namaBrg) {
this.namaBrg = namaBrg;}

public String getJenisBrg() {
return jenisBrg;}
public void setJenisBrg(String jenisBrg) {
this.jenisBrg = jenisBrg;}

public String getMerk() {
return merk;}
public void setMerk(String merk) {
this.merk = merk;}

public String getTgl() {
return tgl;}
public void setTgl(String tgl) {
this.tgl = tgl;}

public String getJumBrg() {
return jumBrg; }
public void setJumBrg(String jumBrg) {
this.jumBrg = jumBrg;}

public String getHrgBeli() {
return hrgBeli;}
public void setHrgBeli(String hrgBeli) {
this.hrgBeli = hrgBeli;}

public String getPosisi() {
return posisi;}
public void setPosisi(String posisi) {
this.posisi = posisi;}
```

Selanjutnya membuat kelas *DBBarang.java* yang berfungsi membuat dan mengakses *database* SQLite. Kelas *DBBarang.java* berisi *method* untuk mengatur *database*. Kode kelas *DBBarang.java* dapat dilihat pada gambar 4.2.

```
package com.example.aset;

import android.app.DatePickerDialog;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.provider.BaseColumns;

public class DBBarang extends SQLiteOpenHelper {
    static abstract class MyColumns implements BaseColumns {
        static final String NamaTabel = "Barang";
        static final String edBarcode = "Barcode";
        static final String edNamaBrg = "NamaBrg";
        static final String edJenisBrg ="JenisBrg";
        static final String edMerk = "Merk";
        static final String edTgl = "TglBeli";
        static final String edJum = "Jumlah";
        static final String edHrg = "HrgBeli";
        static final String edPosisi = "PosisiBrg";
        }
    private static final String NamaDatabse = "dataBrg.db";
    private static final int VersiDatabase = 14;
    private static final String SQL_CREATE_ENTRIES = "CREATE TABLE "+MyColumns.NamaTabel+
            "("+MyColumns.edBarcode+" TEXT NOT NULL, " +
            ""+MyColumns.edNamaBrg+" TEXT NOT NULL, " +
            ""+MyColumns.edJenisBrg+" TEXT NOT NULL, " +
            ""+MyColumns.edMerk+" TEXT NOT NULL, " +
            ""+MyColumns.edTgl+" TEXT NOT NULL, " +
            ""+MyColumns.edJum+" TEXT NOT NULL, " +
            ""+MyColumns.edHrg+" TEXT NOT NULL, " +
            ""+MyColumns.edPosisi+ " TEXT NOT NULL)";

    private static final String SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS "+MyColumns.NamaTabel;
    DBBarang(Context context) { super(context, NamaDatabse,  factory: null, VersiDatabase); }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES); }
    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int il) {
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db); }
}
```

Gambar 4.2 Kode Kelas DBBarang.java

Penjelasan gambar 4.2 adalah sebagai berikut:

1. DBBarang meng- **`extends`** `SQLiteOpenHelper`

```
class DBBarang extends SQLiteOpenHelper
super(context, NamaDatabse, null, VersiDatabase);}
```

2. Mendefinisikan nama tabel, kolom dan baris sebagai konstanta.

```
static final String NamaTabel = "Barang";
static final String edBarcode = "Barcode";
static final String edNamaBrg = "NamaBrg";
static final String edJenisBrg ="JenisBrg";
static final String edMerk = "Merk";
```

```
static final String edTgl = "TglBeli";
static final String edJum = "Jumlah";
static final String edHrg = "HrgBeli";
static final String edPosisi = "PosisiBrg";
```

3. Membuat database menggunakan query.

```
private static final String SQL_CREATE_ENTRIES =
"CREATE TABLE "+MyColumns.NamaTabel+
"("+MyColumns.edBarcode+" TEXT NOT NULL, " +
""+MyColumns.edNamaBrg+" TEXT NOT NULL, " +
""+MyColumns.edJenisBrg+" TEXT NOT NULL, " +
""+MyColumns.edMerk+" TEXT NOT NULL, " +
""+MyColumns.edTgl+" TEXT NOT NULL, " +
""+MyColumns.edJum+" TEXT NOT NULL, " +
""+MyColumns.edHrg+" TEXT NOT NULL, " +
""+MyColumns.edPosisi+ " TEXT NOT NULL)";
```

4. Membuat database dan mengimplementasikan `onCreate`().

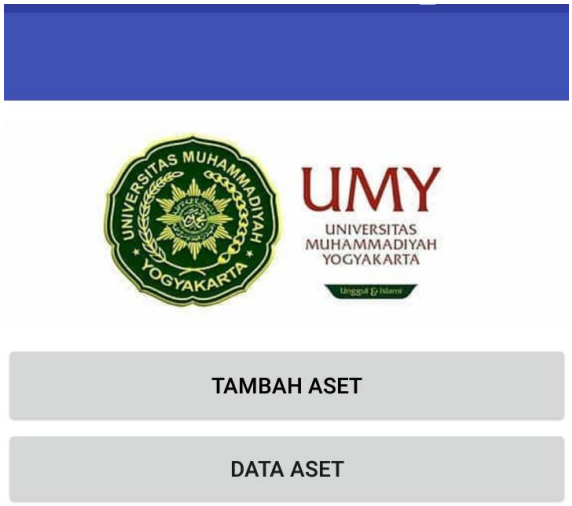
```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_ENTRIES); }
@Override
public void onUpgrade(SQLiteDatabase db, int i, int i1)
{
    db.execSQL(SQL_DELETE_ENTRIES);
    onCreate(db); }
```

4.3 Implementasi *Interface*

Untuk mengimplementasikan sebuah aplikasi diperlukan rancangan desain*user interface* (UI) yang menggunakan bahasa xml. Pada Android ada dua jenis user interface yaitu *view* yang digunakan untuk berinteraksi langsung dengan *user* dan *ViewGroups* digunakan untuk mengatur *view*. UI ini berfungsi untuk memudahkan dalam penggunaan aplikasi oleh *user*.

1. **Halaman*Menu***

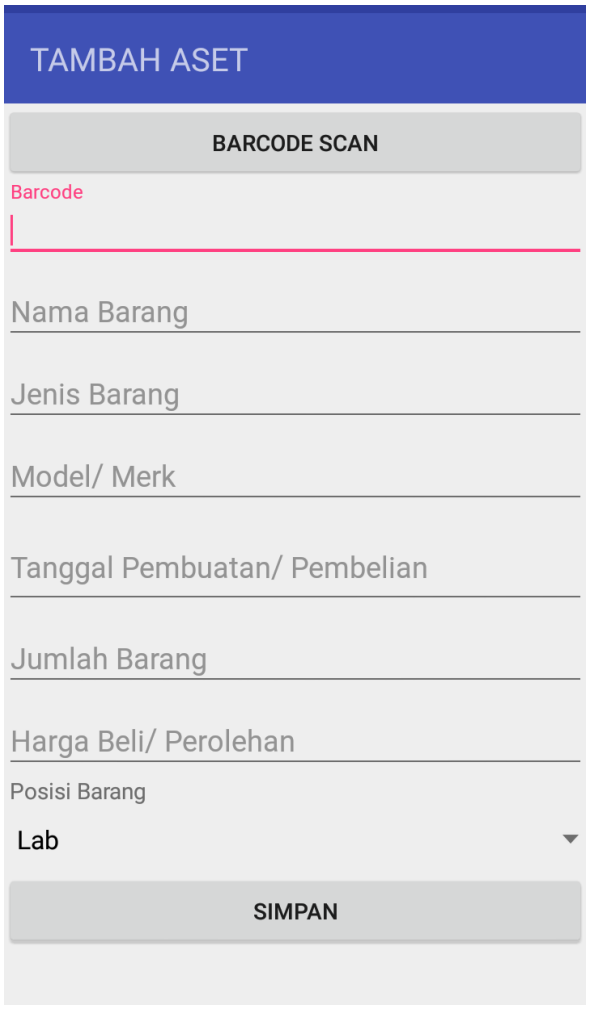
   Halaman menu merupakan tampilan utama aplikasi yang terdiri dari gambar yang berupa logo UMY dan duatombol *menu* yaitu tombol tambah aset yang dimana untuk memasukkan data dan tombol data aset untuk menampilkan data yang dimasukkan ke *database*. Tampilan halaman *menu* dapat dilihat pada gambar 4.3.



Gambar 4.3 Tampilan menu

2. **Halaman Tambah Aset**

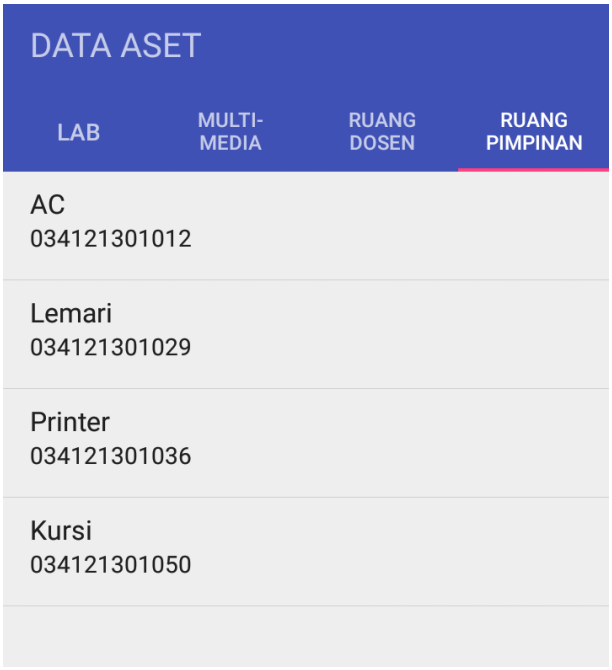
   Halaman tambah aset digunakan untuk menambahkan nama barang, jenis barang, model/ *merk*, tanggal pembuatan/ pembelian, jumlah barang, harga beli/ perolehan, posisi dan *barcode* yang dimana menambahkan *barcode* menggunakan tombol *barcodescan* untuk membaca *barcode*, setelah itu data disimpan ke *database* menggunakan tombol simpan. Tampilan halaman tambah aset dapat dilihat pada gambar 4.4



Gambar 4.4 Tampilan Tambah Aset

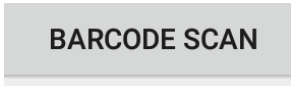
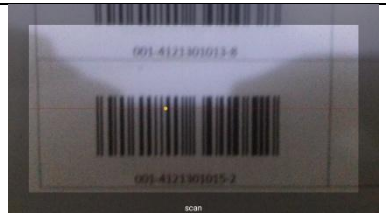
3. Halaman Data Aset

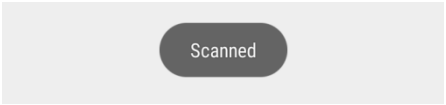
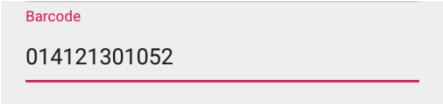
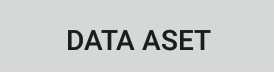
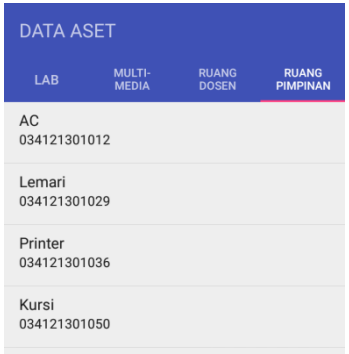
Halaman lihat data digunakan untuk melihat data yang dimasukkan ke *database*.Tampilan halaman lihat data dapat dilihat pada gambar 4.5.



Gambar 4.5 Tampilan Data Aset

Tabel 4.1 Hasil Pengujian Aplikasi

No	Kasus uji	Test Case	Hasil yang diharapkan	Hasil	Status
1.	Tombol Menu Tambah Aset pada halaman tambah aset		User masukkehalaman tambah aset		berhasil
2.	Tombol barcode scan pada halaman tambah aset dapatmasukke barcode reader		User masukke barcode reader		berhasil

3.	Barcode dapatdibaca		User dapat input barcode		berhasil
4.	Tombolsimpan pada halaman tambah aset dapat menyimpan data		User dapat menyimpan data		berhasil
5.	Tombol Menu Data Aset		User masuk kehalaman data aset		berhasil

4.1 Pembahasan

Aplikasi “Sistem Informasi Data Aset Menggunakan *Barcode Scanning* Berbasis Android” menyediakan dua fitur, yaitu tambah aset dan data aset. Fitur utama dari aplikasi ini adalah dapat menginput data aset berupa *barcode*, penambahan data selain *barcode* seperti nama barang, jenis barang, model/ merk, tanggal pembuatan/ pembelian, jumlah barang, harga beli/ perolehan dan posisi barang. Menurut hasil pengujian fungsional yang ada pada tabel 4.1 dapat disimpulkan aplikasi ini berjalan dengan baik sesuai fungsi yang dibuat dan telah lulus tahap pengujian.

4.2 Pengujian

Pengujian aplikasi dilakukan untuk mendapatkan hasil mengenai kualitas aplikasi yang diuji.Yang dimana bertujuan untuk menyesuaikan kebutuhan diperlukan. Metode pengujian yang digunakan “Sistem Informasi Data Aset Menggunakan *Barcode Scanner* Berbasis Android” adalah *Blackbox Testing*.

Blackbox Testing adalah teknik pengujian *software* yang hanya fokus pada fungsional dari sistem dan mengabaikan struktur *control internal.*